Energy Prediction Using Evolutionary Lean Neural Networks

Yong Wee Foo

Submitted in fulfillment of the requirements for the

Degree of Doctor of Philosophy

James Watt School of Engineering

College of Science and Engineering

University of Glasgow

July 2022

Abstract

The demand for data center services, driven by the surge in online applications and services, has propelled energy consumption to unprecedented levels. While renewable energy provides an attractive and more environmentally friendly alternative to existing energy resources, renewable intermittency is a major issue for grid operators. Accurate energy predictions are thus paramount to maintaining optimal services and energy provisions amidst a shift towards greener energy for more sustainable data centers.

Artificial Neural Networks (ANN) are powerful learning machines adopted for several decades for prediction problems. Recent years have seen increased interest in ANN, led by advancements in AI and computing hardware. Despite the significant progress, ANNs are notoriously hard to train and extremely difficult to interpret as the relationships between the input variables and the output responses are often hard to tease apart. The structure of ANN can considerably impact its performance as it has a direct dependency on the model architecture and parameters. Achieving high performance accuracy and the ability to generalize across different problem sets remains a big challenge for ANNs. For example, an over-trained model becomes too large and complex, is more prone to overfitting, and cannot make accurate predictions as it does not generalize well to new data. Additionally, the more complex a network, the more difficult it is to explain the relationships learned by the model.

Traditionally, most research focus on model parameter learning, where gradient-based methods are frequently applied to optimize connection weights and biases. In contrast, model architecture learning is manually set based on experience or trial-and-error experimentation. However, this approach suffers several constraints, including limiting the search space of candidate solutions with a predefined number of neurons and connections.

To address these limitations, a novel ANN called the Evolutionary Lean Neural Network (EVLNN) is developed in this thesis. EVLNN uses an improved Genetic Algorithm (GA) to optimize ANN architecture and parameters, offering greater training flexibility than traditional approaches. The proposed approach has the advantage of simplifying energy

prediction tasks by allowing one to specify parameters such as the minimum and maximum network size, the transfer functions, feedforward architecture, or architecture with feedback for time series forecasting. In this approach, structural optimality properties of the problem are formulated and solved with an implementation of an improved GA that includes species parallelism, intra-and-inter species crossover, and a two-stage mutation. EVLNN serves as a global search algorithm by locating a parsimonious ANN that can provide a more generalized solution. Sensitivity analysis mechanisms are designed into the algorithm to help with interpretability and understanding of the model.

In developing the EVLNN algorithm, a set of benchmark functions was used to empirically evaluate and compare the algorithm's performance with other well-established algorithms - Particle Swarm Optimization (PSO), Differential Evolution (DE), and the standard Genetic Algorithm (GA). The results showed EVLNN's ability to generalize well by locating the peaks in all the test functions, whereas the other algorithms have located the peaks in all but one test function.

The EVLNN algorithm was applied to two energy prediction problems in this thesis. The first application is in predicting the energy consumption of a Hadoop testbed. Using variables related to energy consumption from the Hadoop system, EVLNN accurately predicted its energy consumption and helped identify key energy influencing factors. It also performed more favorably than networks trained by PSO-NN, DE-NN, and GA-NN. The second application is in the forecasting of solar irradiance. EVLNN showed accurate forecasts in different settings of time resolutions (sample size) and using a different number of input variables. In most of those settings, EVLNN outperformed PSO-NN, DE-NN, GA-NN, and the fully-connected Time Delay Backpropagation neural network (TD-BPNN).

Accurate energy predictions underpin the essential improvements required in energy resource management for both data center owners and grid operators. Furthermore, the ability to explain and interpret the model behavior provides a basis for understanding the dynamics of energy consumption. This work has provided a simplified and flexible approach to ANN architecture design and parameter optimization to achieve interpretable models with high accuracy and good generalization properties for energy prediction

problems. The findings demonstrated that EVLNN could create parsimonious models for accurate energy prediction, which are also capable of discovering the relationships between key determinants of energy consumption.

Contents

List of Tables

List of Figures

List of Abbreviations

ACO	Ant Colony Optimization
AI	Artificial Intelligence
AIS	All-In Strategy
ALNM	Active Learning Based Niching Method
ANFIS	Adaptive Network-based Fuzzy Inference System
ANN	Artificial Neural Network
ANSGAII	NSGAII with variable-space niching
AR	Additive Regression
ARMA	Autoregressive Moving Average
ARMAX	ARMA with Exogenous variables
ARX	Autoregressive with Exogenous inputs
AT	Ambient Temperature
AWS	Amazon Web Services
BP	Backpropagation
CAMS	Copernicus Atmosphere Monitoring Service
CEC	Congress on Evolutionary Computation
CEP	Complex Event Processing
CLSTM	Convolutional Neural Network with Long Short-Term Memory
CMA-ES	Covariance matrix adaptation evolution strategy
CMC	Canadian Meteorological Centre
CMV	Cloud Motion Vectors
CNN	Convolutional Neural Network
Crowding DE/rand/1/bin	Classic DE algorithm extended with the crowding scheme
CS	Covering Subset
CW	Connection Weights
dADE/nrand/1/bin	Dynamic Archive Niching Differential Evolution Algorithm 1
dADE/nrand/2/bin	Dynamic Archive Niching Differential Evolution Algorithm 1
DE	Differential Evolution
DECG	DE algorithm using crowding and gradient descent

DELG	DE algorithm using local selection and gradient descent
DELS_ajitter	DE algorithm using local selection and ajitter global mutation
DE/nrand/1/bin	Classic Differential Evolution algorithm with dynamic and clustering 1
DE/nrand/2/bin	Classic Differential Evolution algorithm with dynamic and clustering 2
DE-NN	Differential Evolution based Neural Network
DHI	Diffused Horizontal Irradiance
DNI	Direct Normal Irradiance
DNN	Deep Neural Network
DT	Decision Tree
DVFS	Dynamic Voltage and Frequency Scaling
EA	Evolutionary Algorithm
EAGA	Energy-Aware Greedy Algorithm
EANN	Evolutionary Artificial Neural Network
ECMWF	European Centre for Medium-Range Weather Forecasts
EFS	Energy-aware Fair Scheduling
eGMC	enhanced Green MapReduce Cluster
ELM	Extreme Learning Machine
EMRSA	Energy-aware MapReduce Scheduling Algorithm
ENN	Elman Neural Network
EU	European Union
EURECA	European Union Resource Efficiency Coordination Action
EVLNN	Evolutionary Lean Neural Network
FCNN	Fully Connected Neural Network
FWNN	Fuzzy Wavelet Neural Networks
GA	Genetic Algorithm
GABPNN	Genetic Algorithm optimized BPNN
GA-CuDNNGRU	Genetic Algorithm-Cuda Deep Neural Network Gated Recurrent Unit
GA-DNN	Genetic Algorithm-Deep Neural Network
GA-GRU	Genetic Algorithm-Gated Recurrent Unit
GA-LSTM	Genetic Algorithm-Long-Short Term Memory

GA-NN	Genetic Algorithm based Neural Network
GA-SVR	Genetic Algorithm based Support Vector Regression
GB	Gigabyte
Gbps	Gigabit per second
GBRT	Gradient Boot Regression Tree
GFS	Global Forecast System
GHI	Global Horizontal Irradiance
GMC	Green MapReduce Cluster
GP	Genetic Programming
GRNN	Generalized Regression Neural Network
GRU	Gated Recurrent Unit
GSR	Global Solar Radiation
GS-SVR	Grid Search-Support Vector Regression
GW	Gigawatt
GWh	Gigawatt hour
HDFS	Hadoop Distributed File System
IEA	International Energy Agency
IoT	Internet of Things
iPDU	Intelligent Power Distribution Unit
iPOP-CMA-ES	CMA-ES with increasing population size
kbps	Kilobits per second
KL	Kullback-Leibler
KNN	K-Nearest Neighbor
KW	Kilowatt
KWh	Kilowatt Hour
LBNL	Lawrence Berkeley National Laboratory
LC	Linguistic Complexity
LDPC	Low-Density Parity Check
Leaky ReLU	Leaky Rectified Linear Unit
LM	Levenberg-Marquardt
LM-BPNN	Levenberg-Marquardt Backpropagation Neural Network
$LSEA_{EA}$	Localised search evolutionary algorithm using EAs for local search

$LSEA_{GP}$	LSEA using Gaussian process as its local search mechanism
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MBE	Mean Bias Error
MEA	Multinational Evolutionary Algorithm
MERF	Multiple Energy-Related Features
MIB	Management Information Base
MKP	Multidimensional Knapsack Problem
ML	Machine Learning
MLP	Multi-Layer Perceptron
MOS	Model Output Statistics
MPC	Model Predictive Control
MPP	Massively Parallel Processing
MSE	Mean Square Error
MSSPSO	Multi-Sub-Swarm Particle Swarm Optimisation algorithm
NAM	North American Mesoscale
NARXNN	Nonlinear Autoregressive Neural Network with Exogenous inputs
NCEP	National Centers for Environmental Prediction
NEA1	Niching Evolutionary Algorithm 1
NEA2	Niching Evolutionary Algorithm 2
nMAE	normalized MAE
NMMSO	Niching Migratory Multi-Swarm Optimizer algorithm
NOAA	National Oceanic and Atmospheric Administration
nRMSE	normalized RMSE
N-VMO	Niching Variable Mesh Optimization algorithm
NWP	Numerical Weather Prediction
OID	Object Identity
PaD	Partial Derivative
PB	Petabytes
PCC	Pearson Correlation Coefficient
PCNN	Partially-Connection Neural Networks

PNA-NSGAII	Parameterless niching assisted NSGAII
PPA	Power Purchase Agreement
PR	Peak Ratio
PSO	Particle Swarm Optimization
PSO-NN	Particle Swarm Optimization based Neural Network
PSO-SVR	PSO-Support Vector Regression
PT	Panel Temperature
PUE	Power Usage Effectiveness
PV	Photovoltaic
QoS	Quality of Service
R^2	Coefficient of Determination
RAID	Redundant Array of Independent Disk
RBF	K-means Radial Base Function
RDPS	Regional Deterministic Prediction System
REC	Renewable Energy Credits
ReLU	Rectified Linear Unit
RH	Relative Humidity
RI	Relative Importance
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SA	Sensitivity Analysis
SaDE-ELM	Self-adaptive Differential Evolutionary Extreme Learning Machine
SNMP	Simple Network Management Protocol
SR	Success Rate
SSD	Sum of the Square Partial Derivatives
SUS	Stochastic Universal Sampling
SVR	Support Vector Regression
Tanh	Hyperbolic Tangent
TD-BPNN	Time Delay Backpropagation Neural Network
TSI	Total Sky Imagery
TWh	Terrawatt hour
US	United States

WRF	Weather Research and Forecasting
XML	eXtensible Markup Language
YARN	Yet Another Resource Negotiator

List of Symbols

B_I	A single-bias nodes at the input layer
b_j	Input bias at node *j*
B_H	A single-bias nodes at the hidden layer
$D(P)$	Population diversity of population *P*
E_H	Shannon's equitability index
EL_p	Elitism percentage
f_{min}	Function minima
f_{opt}	Function optima
G_{max}	Maximum number of generations
G_{opt}	Number of optima
H	Shannon diversity index
h_j	Hidden node *j*
$h_{j,\ in}$	Sum of weighted inputs at hidden node *h*
$h_{j,out}$	Output value at hidden node *j*
i_j	Individual genotype
$MaxFE$	Maximum number of function evaluations
MU_p	Mutation percentage
MU_r	Mutation value range
MX_p	Mutation matrix probability
N	Individual's chromosome matrix
N'	New individual's chromosome matrix

N''	New individuals with re-enabled links in N'
N_P	Population size
N_s	Number of species
NN_i	The *ith* individual in population *P*
$NN_{sp_{i,j}}$	Speciated *jth* individual within the subpopulation of species *i*
NPF_i	Number of global optima found during the i^{th} run
NR	Number of runs
NSR	Number of successful runs
$o_{k,in}$	Input at output node k
o_k	Output node k
$o_{k,out}$	Output at output node node k
P	Population size
P'	Speciated population
P_L	Probability matrix for link-node mutation
P_W	Probability matrix for weights mutation
Q_{ih}	Ratio of the absolute value of the connection weight and the sum of the absolute value of the connection weights of all input neurons
R_L	A vector created with normally distributed random generated elements of values between 0 and 1 for link-node mutation
R_{L2}	A second vector created with normally distributed random generated elements of values between -0.5 and 0.5 for link-node mutation
R_W	A normally distributed random generated matrix with elements of values between 0 and 1
s_{i_j}	Genome string of individual i_j
S_{i_j}	The set of substrings of s_{i_j}

$S_{\{i_1,i_2,\ldots,i_n\}}$	Total number of substrings, s_{i_j} in the population
SP_i	Subpoulation of species *i*
w_{ij}	Connection weight between the input node *i* and the hidden node *j*
W_p	Peak Watt of Solar Panel
W_W	Weights-change matrix
x_{ij}	Represents the ith row and jth column of the matrix for the connection between nodes *i* and *j*
XO_p	Intra-species crossover percentage
XS_p	Inter-species crossover probability
$\sigma_j(\cdot)$	Activation function at node *j*
$\sigma_{j,tanh}(\cdot)$	Hyperbolic Tangent (Tanh) activation function at node *j*
$\sigma_{k,linear}(\cdot)$	Linear activation function at node *k*
$\sigma_{k,lrelu}(\cdot)$	Leaky ReLU activation function at node *k*
$\sigma_{k,relu}(\cdot)$	Rectified Linear Unit (ReLU) activation function at node *k*
$\sigma_{k,sigmoid}(\cdot)$	Sigmoid activation function at node *k*

List of Publications

1. Y. W. Foo and C. Goh, “Solar Irradiance Forecasting with Fewer Features and Small Dataset using EVLNN", IEEE Open Access J. Power Energy, June 2022 (Under Review).

2. Y. W. Foo and C. Goh, “Solar Irradiance Forecasting in Tropical Weather using an Evolutionary Lean Neural Network,” 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 490-497, doi: 10.1109/CEC45853.2021.9504875.

3. Y. W. Foo, C. Goh, L. Chan, and Y. Li, “Generalized Hybrid Evolutionary Algorithm Framework with a Mutation Operator Requiring no Adaptation,” International Conference on Simulated Evolution and Learning (SEAL) 2017, Springer International Publishing AG 2017, pp. 486-498.

4. Y. W. Foo, C. Goh, Y. Li, “Speciation and Diversity Balance for Genetic Algorithms and Application to Structural Neural Network Learning,” IEEE International Joint Conference on Neural Networks (IJCNN), 2016, pp. 1283-1290.

5. Y. W. Foo, C. Goh, Li, Yun, “Machine Learning with Sensitivity Analysis to Determine Key Factors Contributing to Energy Consumption in Cloud Data Centers,” International Conference on Cloud Computing Research and Innovation (ICCCRI), 2016, pp. 107-113.

6. Y. W. Foo, C. Goh, H.C. Lim, Z-H Zhan, and Y. Li, “Evolutionary Neural Network Based Energy Consumption Forecast for Cloud Computing,” International Conference on Cloud Computing Research and Innovation (ICCCRI), 2015, pp. 53-64.

7. Y. W. Foo, C. Goh, H.C. Lim, and Y. Li, “Evolutionary Neural Network Modeling for Energy Prediction of Cloud Data Centers,” International Symposium on Grids

and Clouds 2015 (ISGC2015) - Highly Distributed Computing Systems, Proceedings of Science Vol. 239.

8. Z-G Chen, Z-H Zhan, H-H Li, K-J Du, J-H Zhong, Y.W. Foo, Y. Li, and J. Zhang, “Deadline Constrained Cloud Computing Resources Scheduling through an Ant Colony System Approach,” International Conference on Cloud Computing Research and Innovation (ICCCRI), 2015, pp. 112-119.

9. Y. Wei, Y.W. Foo, K.C. Lim, F. Chen, “The Auto-configurable LDPC Codes for Distributed Storage,” IEEE 17th International Conference on Computational Science and Engineering (ICCSE), 2014, pp. 1332-1338.

10. Y. Wei, Y. W. Foo, “A Cost-Effective and Reliable Cloud Storage,” IEEE 7th International Conference on Cloud Computing (ICCC), 2014, pp. 938-939.

Acknowledgment

"Two roads diverged in a wood, and I,

I took the one less traveled by,

And that has made all the difference."

- An excerpt from "The Road Not Taken" by Robert Frost

Making a choice is simple; sticking to it is hard. However, the irony is that you will never know you have made the "right" choice. Since "way leads on to way," one will never get the chance to experience the other road and can never know which was less traveled. At the end of the journey, we always rewrite our own histories to justify our decisions. And that makes all the difference.

The completion of this thesis has been a bittersweet journey. Without the tireless prayers, boundless love and support from my wife Jane, and unflagging encouragement from my daughters, Gladys and Natalie, I would never have come this far. Their love, patience, understanding, and tolerance of my shortcomings have been my comfort and strength.

I am grateful for my brother, Yong Chean, who is an example and inspiration in my life and keeps me motivated in my research work. Many thanks to Steven Tan, who lent me his quiet but essential support in this journey.

Special thanks to my supervisor, Dr. Cindy Goh, for the guidance and advice. I am grateful and honored to have had the opportunity to work with such an excellent supervisor. Thanks to Joo Hock, Cheng Leong, Leslie, and Lai Meng for the company. They have made my journey all the more memorable with their waves of laughter during relaxation time and sympathetic ears at times of difficulty.

Last but not least, I want to thank my God and Father in Heaven, who is my pillar, fortress, and deliverer, my rock, in whom I take refuge, my shield, and salvation. I am nothing without Him. And that makes all the difference.

Author’s Declaration

I hereby declare that this thesis was composed and originated from work entirely carried out by myself. The work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted elsewhere in consideration for a higher degree or professional qualifications.

Rights Statement

Chapter 1

1. Introduction

1.1 The Nexus between Data Center Energy Efficiency and Renewables

The demand for data center services has risen worldwide to support big data and digital services [1] [2]. This demand has propelled data center energy consumption to unprecedented levels. Based on the International Energy Agency (IEA) 2020 report, energy usage by global data centers in 2019 was approximately 200 Terrawatt hour (TWh) [3]. The consumption puts the data center at 1% of global electricity demand [4] and around 0.3% of overall carbon emissions. The continued upsurge of big data-driven by a plethora of social media applications, eCommerce websites, mobile gaming platforms, the Internet of Things (IoT), autonomous systems, and cryptocurrency is a growing concern for data center electricity use and its potential impact on the environment [5].

With existing efficiency resources almost fully tapped and the projected global data center compute instances potentially doubling within the next 3 to 4 years [6], an intentional effort to manage sustainable energy growth for data centers is pertinent. Instead of seeing data centers as an environmental threat, they provide the much-needed push to accelerate progression in renewables. In its Innovation Landscape for the Power Sector study, IRENA showed that data centers sit at the nexus of energy efficiency, renewable energy, and the burgeoning data economy in an increasingly digitalized world [7]. Amidst broader climate goals and the push for renewables and green data centers, accurate energy predictions can help address the challenges and opportunities for a more sustainable future.

There have been proposals to design sustainable data centers with facilities enabled by supply and demand-side management to take advantage of efficiency gain by predicting and shifting workload demands to exploit renewables' time variations and availabilities [8] – [10]. However, integrating renewable energy sources in data centers is non-trivial as they

are far more complicated than traditional data centers [11] [12]. The complication can be viewed from both sides. Firstly, predicting demand-side requires precise accounting and knowledge of how the energy consumption of various workloads and server utilization varies non-linearly with localized conditions like temperature and humidity [13]. Secondly, predicting the supply-side generation requires an accurate renewable energy forecast that would allow grid operators to optimize clean electricity generation. For example, solar irradiance's intermittency and stochastic behavior mean that solar power generation is not guaranteed, making it hard to predict its availability. Though sunrise and sunset events can be anticipated, as when the sun rises or sets each day is known, solar irradiance forecasting during the day remains challenging due to fluctuations in solar radiation from frequent cloud formation and changing weather patterns. Hence, advanced forecasting techniques and AI will become critical to integrating higher renewable energy shares into the grid to support greener data centers [14].

1.1.1. Big Data and the Data Center Transformation

In the last decade, the data center has transformed its computing platform to adapt to the era of data deluge, one of the most popular being the Hadoop platform. Hadoop consists of the Hadoop Distributed File System (HDFS) [15] with MapReduce [16] or Yet Another Resource Negotiator (YARN) [17] as software frameworks and has been widely adopted as an open-source Massively Parallel Processing (MPP) platform for big data applications. Internet media companies like Meta (formerly known as Facebook), Google, and Microsoft use platforms like Hadoop to process and analyze big data [18] – [20]. The Hadoop platform can scale into thousands of nodes in a single cluster, supporting hundreds of Petabytes (PB) of data to meet growing demands. The sheer scale and high fault-tolerant nature of Hadoop constitute a significant piece of the overall data center system. Inevitably, the intense energy usage from the Hadoop platform would be massive.

1.1.2. Data Center Energy Demand

In Meta's data center alone, energy consumption surged by 33%, from 509 GigaWatt hour (GWh) in 2012 to 678 GWh from the year before. The company's total data center energy

consumption increased by another 16% in 2013 to 822 GWh as it massively scaled up its cloud infrastructure to meet intense customer demands [21]. In 2019, the company's electricity usage reached a new high of 5.1 TWh [22], a 50% increase from the previous year of 3.2 TWh in 2018 [23].

Energy consumption of Google's data centers has also increased from 2.86 TWh in 2011 to 10.6 TWh in 2018 as computing power skyrocketed and data center capacity expanded to meet robust demand for data services [24] [25]. In 2019, the company announced that an investment of more than $13 billion would go into building new data centers and expanding existing ones [26]. Similar announcements were also made by top-ranked cloud service providers such as Amazon Web Services (AWS), Microsoft, and Tencent to continue their investment and building of data centers at hyper-scale [27] – [29].

Consequently, studies have forecasted that data center electricity use could reach 2,967 TWh/year by 2030 [30], reaching a level of consumption equivalent to one-quarter of worldwide electricity consumed in 2010 [31]. The European Union's (EU's) EURECA (EU Resource Efficiency Coordination Action) Project put the energy consumption of the European data center at 130 TWh in 2017 [32]. The Lawrence Berkeley National Laboratory (LBNL) in the United States (US) projected that US data center energy consumption at approximately 70 TWh in 2020 [3] [4]. Greenpeace cited that China's data centers' energy consumption was 160 TWh in 2018 and is projected to reach 367 TWh by 2023 [5]. At the same time, these data centers represent 3.8% of CO_2 emissions [33], and it is estimated that by 2025, greenhouse gas emissions by data centers will have tripled since 2010 [34].

1.1.3. Data Center Energy Efficiency Gains

Most studies claiming enormous energy used by data centers often overlook strong countervailing energy efficiency trends that have occurred in parallel. Some of the considerable energy improvements came from technological advancements such as increased server, storage, networking equipment efficiencies, and greater use of virtualization technologies. Others came from increased gains in data center operational

efficiency. Together, it is possible to achieve substantial growth in digital services with much smaller growth in the energy footprint [35]. However, given the ever-growing demands for data center services, current strategies for energy efficiency improvement will not likely keep up with the rate of data center expansion.

The industry's most notable data center energy efficiency metric is the Power Usage Effectiveness (PUE). PUE is calculated using the equation,

$$PUE = \frac{Total\ Power\ into\ Data\ Center}{IT\ Equipment\ Power} \tag{1.1}$$

where, *Total Power into Data Center* is the energy dedicated solely to the data center, and *IT Equipment Power* is the energy consumed by equipment used to manage, process, store, or route data within the compute space. A PUE of 1, which would be ideal, means all the power going into the data center is being used to power IT equipment. World-class data centers are approaching a practical minimum in operating PUE of 1.1 or lower [36] [37], as they take advantage of economies of scale and leverage the latest technology and practices. However, the industry data centers' PUE in a 12-year study has flattened out with an average of 1.67 in 2019 [38]. It is observed in the study that the reduction in the average PUE from 2007 to 2013 is mainly due to the adoption of methods such as hot and cold air separation, raising data center temperatures, or applying more control on computer room air-conditioning and power distribution. The PUE numbers indicate that energy efficiency gains leveraging conventional methods may have been dampened.

1.1.4. Renewables and AI for Efficient Data Center Energy Management

Explosive growth in data center infrastructure does not need to mean growth in emissions. Transitioning energy from thermal-based sources to renewables to power data centers could be a game-changer. By leveraging the attractive economics of renewables, and the increased efficiencies made possible by Artificial Intelligence (AI), energy-intensive data centers could be self-sustaining [39]. Research in AI and renewables-enabled data center sustainability is growing [40] – [43]. A recent example is the tropical data center testbed set up as a state-of-the-art facility for energy efficiency research [44]. The Uptime Institute

Intelligent Report on 'Five data center trends for 2021' concurs with this development, citing sustainability and AI as the leading trends for data centers in 2021 [45]. Operators that can successfully harness this energy source coupled with AI advancement might change how data centers operate in the future. Big technology companies such as AWS, Google, Microsoft, and Meta, are already integrating higher shares of renewable power (e.g., wind, solar, hydro, marine, and geothermal power) into their data centers or purchasing renewable energy credits (REC) to offset their fossil fuel usage [46] – [49].

Among the renewable energies, clean electricity systems based on solar photovoltaic (PV) power generation is the fastest-growing energy source worldwide [50]. The accelerated deployment of solar PV globally, combined with the rapid development of the solar energy industry, has driven costs down. With the cost of renewable electricity falling, transition effort could be further boosted as power represents as much as 70% of the data center's total operating costs [51]. Currently, the issue limiting its growth is the intermittency of renewable energy. The intermittent nature of renewables means that electricity from these sources will not be continuously available. It also creates technical challenges in integrating renewable energy into the grid. Therefore, intermittent availability has been at the forefront of renewable energy research for a number of years to resolve the impacts of intermittent generation. With the advancements in AI, it is possible to alleviate the impacts of intermittent generation through accurate solar energy forecasting to further integrate solar capacity into the grid for a predictable generation.

1.2 Energy Modeling and Prediction

Energy prediction is of paramount significance for the optimal operation of systems and plants to meet their energy needs [52]. Accurate models of underlying systems are pivotal to predicting the systems' behaviors. Such predictions are integral to business planning, resource management, and energy efficiency improvement. For example, data center planners can use energy consumption predictions to improve workload scheduling, resource allocation, and data replication or placement to achieve energy proportionality.

Grid operators require accurate forecasts of solar variability for reliable dispatch, ramp forecasting, or spinning of reserves from additional sources. Most works consider solving these problems separately. This research aims to lay the groundwork to bridge the gap between these two engineering problems by improving the generalizability of a predictive model capable of both demand-side and supply-side energy prediction.

In general, the approaches used in modeling energy systems can be categorized into the physical and Machine Learning (ML) approaches.

1.2.1 Physical Modeling Approach for Energy Prediction

Most early studies and existing work that focus on physical models are based on mathematical equations describing a dynamic system's physical state. In the energy consumption modeling of data centers, [53] developed a general power consumption model for the Hadoop cluster, and [54] proposed an energy model for Hadoop workloads. In solar irradiance forecasting, [55] presented a physical satellite model based on participating atmospheric components, radiometers, and meteorological data. Global spectral numeric weather prediction models [56] are also popular models used to predict solar irradiance through weather phenomena occurring in the Earth’s atmosphere. These approaches to describing and analyzing energy predictions require a precise and clearly defined mathematical model that involves selecting appropriate meteorological features and collecting vast data.

Energy modeling based on physical models is the most accurate. They also have the advantage of not requiring any historical data, making them flexible as they can simulate a future system so long as its physical properties are known. However, physical models are demanding since they must include all the necessary mathematical equations and data. Using the underlying physics to solve the equations numerically requires an expensive process of abstracting the full underlying properties of a nonlinear energy system to a high degree of accuracy, making this method unamenable.

1.2.2 Machine Learning Approach for Energy Prediction

On the contrary, ML, a class of AI, uses a data-driven approach that seeks to match input-output predictions to data that does not require the abstraction of complex underlying properties of the systems as in physical models. Its approach reverse engineers existing data to learn and discover patterns or hidden information in the system. While ML techniques have been adopted for several decades, interest in the field has grown in recent years, led by advancements in computing hardware and revolutionary development in AI. In particular, a class of ML techniques known as Artificial Neural Networks (ANN) and Deep Neural Networks (DNN) has seen a resurgence in energy prediction research [52]. In energy consumption modeling, Fuzzy Wavelet Neural Networks (FWNNs) were proposed as a control model to improve the energy efficiency of the Hadoop cluster [57]. Liang and Hu [58] presented a deep learning model to predict energy consumption using multiple energy-related features acquired from a Hadoop cluster. The use of ANN or deep learning for solar energy forecasting is also widespread. Convolutional Neural Networks (CNNs) were employed to predict the sunshine duration [59], Recurrent Neural Network (RNN) was employed for solar radiation forecast [60], and Long Short-Term Memory (LSTM) models were successfully deployed to predict solar radiation [61].

While ANN-based ML methods can create models from data, it has several disadvantages. Firstly, the model training process can be time-consuming and expensive due to a large number of parameters and hyperparameters to optimize and often relies on expert knowledge and trial-and-error to determine the optimal structure. Secondly, while the powerful and versatile ANN is capable of learning the non-linear and intricate interactions between features, it is also more prone to overfitting with its complex structure, limiting its generalization ability. Thirdly, the more complex a network, the more difficult it is to interpret the cause of the results in relation to the inputs. Lack of interpretability prevents the model from being queried to understand which specific features are relevant for making predictions causing them to be less desirable for real-world applications.

1.3 Methods for Neural Network Structural Learning

ANN's performance depends on two aspects – the model architecture and the model parameters. The model architecture consists of the number of hidden neurons and their connectivity, whereas the model parameters consist of connection weights and biases. Theoretically, an infinite set of model architecture associated with the model parameters representing the ANN structure exists, making it hard to locate an optimal model. Hence, selecting an appropriate network structure is non-trivial [62].

The neural network structural learning problem can be viewed as an optimization or search problem where structural representations form the search landscape. In ANN design, a cost function can be formulated as the objective function to minimize, with the structural representations embodying learnable parameters for the optimization. Various optimization methods have been proposed in the past years to train ANNs. These methods can be categorized into gradient-based, reinforcement learning-based, and nature-inspired search [63].

1.3.1. Gradient-Based Methods

In gradient-based methods, the search moves towards the optimum solution in a continuous space using a gradient descent method. Candidate model architecture is sampled from the search space, trained on the training dataset by gradient-based methods, and evaluated on the validation dataset, guided by the objective (or cost) function. The target is to reduce the model's predicted error averaged over the entire training dataset. The Backpropagation (BP) algorithm is commonly used to guide the training of ANN, where the error of the cost function is propagated backward through the network [64]. The derivatives of this error component are used to evaluate the search direction iteratively to help reproduce model architectures with higher performance [65]. The disadvantage of this method is that the objective function must be differentiable, limiting the selection of objective functions used to train the network [66]. Since model architecture is often discrete, this method necessitates converting the search space to continuous values if the gradient-based method

is used [67]. Another shortcoming with the gradient-based method is that it can easily get trapped in local minimas of the objective function.

1.3.2. Reinforcement Learning-based Methods

Reinforcement learning (RL) is a technique that trains intelligent agents to solve a specific task using a set of actions to maximize some accumulative rewards [68]. RL-based methods in network structural learning train an RNN controller using RL to generate DNN architectures represented by variable-length strings [69] [70]. During the DNN training, the accuracy or reward signal computed is used by RL to train the RNN. The RNN controller then determines a sequence of operator and connection tokens to construct the networks. Over time, the RNN will learn to improve the search as DNN architecture with higher accuracy will be given higher probabilities. However, the RL controller needs to try tens of actions to get a positive reward as a supervisory signal, making the training process inefficient [71].

1.3.3. Nature-Inspired Search Methods

Nature-inspired search methods are stochastic approaches that only use cost function values to drive the search process. They do not require the cost functions to be continuous or differentiable and use a population-based approach to search large spaces for candidate solutions [72]. Examples of nature-inspired search methods are Evolutionary Algorithms (EA), such as the Genetic Algorithm (GA) [73] and Differential Evolution (DE) [74], and Swarm Intelligence, such as the Particle Swarm Optimization (PSO) [75].

GA is a bio-inspired search algorithm that allocates resources to explore new regions for solutions by successfully exploiting randomness and subsequently competing for survival based on an estimate of the fitness of the competing regions. A general process of this simulation comprises defining the solutions to the problem using genetic representation called chromosomes. These chromosomes collectively form a population of individuals. An objective function is then designed to evaluate the health of these individuals, where the solutions are subsequently ranked according to their fitness value. Next, healthier

individuals recombine to form new solutions through the generic operators of selection, crossover, and mutation designed to alter their genetic composites during reproduction. With iterations, the concept of survival of the fittest in simulated evolution is expected to lead to improved solutions. The iteration continued and converged when the fittest individual is found, or a stop conditions are met.

DE is a heuristic algorithm for global optimization that uses a parallel direct search approach to optimizing system properties by appositely selecting the system parameters. Essentially, the steps in DE involve initializing a population where each individual is represented by a parameter vector of D dimensions. An objective function is defined, which is then employed to evaluate every generation in search of the best parameter vector. Selecting parameter vectors for the next generation is based on the Darwinian evolution rule, whereby the parameter vectors with better fittest are to be selected. The process converges when the best parameter vectors are found, or the stop conditions are met.

PSO is a bio-inspired algorithm based on bird flock or fish school, where organisms teem in search of food (optima) by assessing their velocity and location (own best). This is done while following the optimum organism (global best) and at the same time interacting with each other (cognitive factor) and their environment (social factor). The PSO simulates this behavior by initializing a population of random solutions called particles. Each particle is a potential solution that searches for optima characterized by the particle's velocity and position as it moves through the solution space. At each iteration, the particle determines its own best solution so far and the population's global best solution. With these values, the particle updates its velocity and positions at each iteration with learnings from the previous best particle and the global best particle. The iterative process converges when the global best particle is found, or the stop conditions are met.

Due to their population-based, highly parallelizable, stochastic search approach, these algorithms have been known to overcome challenges presented by multiple objectives and multiple local optimal, making them more general than gradient-based methods [76]. In addition, they also have fewer meta-parameters than RL-based methods, making the algorithms simpler to implement [77]. For example, DE was successfully applied to

optimize LSTM for electricity prices prediction [78], and a combination of GA and DE was proposed to train ANN for short-term load forecasting [79]. Although EAs are prone to locate the global optimum, there is no guarantee of convergence to an optimal solution.

1.4 Energy Prediction Challenges and the Evolutionary-based ANN Approach

Accurate energy prediction is challenging due to its dependency on environmental conditions. For example, solar irradiance forecasting is affected by atmospheric conditions and changing cloud formation, and data center energy consumption by changes in applications and workloads. While the power of ANNs to approximate any function is well-documented as these models are superior at recognizing patterns in data and generating accurate predictions, they are also prone to overfitting. ANN architecture design is also challenging, requiring human expertise and expensive trial-and-error efforts. Moreover, ANN's complex structure often makes it unclear how the model approximates functions.

Automating ANN design using an evolutionary-based approach can remove human intervention and eliminate the tedious trial-and-error process. The design can consider parsimony while capturing the meaning of nonlinear relationships between the inputs and the outputs, generalized for predictions of data center energy consumption on the demand side and forecasting of solar irradiance on the supply side.

1.5 Aims of Research

This research aims to make the problem of energy prediction simpler using an evolutionary-based approach to ANN learning to produce an interpretable and generalized model. This will be achieved with a learning algorithm that uses several novel mechanisms of an improved GA for the structural optimization of ANNs based on parsimony. The algorithm

will include the design of a matrix encoding scheme allowing for learning feedforward and feedback ANNs. Two crossover strategies will be created to maintain species parallelism and intensify the search for promising basins of interest while exploring new parts of the search landscape. A two-stage mutation will be introduced to avoid local optima in complex problems by making small gene pool variations. Diversity measures will be implemented to track population diversity and provide information about the search behavior. The algorithm will also include an ensemble-based approach to sensitivity analysis to improve model interpretability and make it easier to identify input features that most affect the outputs.

1.6 Key Contributions

- A novel Evolutionary Lean Neural Network (EVLNN) has been designed to improve the interpretability and generalization of energy predictions. It uses an improved GA to optimize network parameters automatically, minimizing human experts' involvement and the costly, inefficient trial-and-error effort.
- A unique structurally inclusive matrix encoding scheme is designed for feedforward and feedback propagation ANN models based on parsimony while not having complications of extra parameters in the representation, offering higher accuracy with fewer features and a smaller number of samples.
- Intra-species and inter-species crossover strategies and a two-stage mutation involving weights and link-node mutation are developed to provide species parallelism amidst the explorative search for novel landscapes that aims to converge towards global optima.
- An ensemble-based approach to sensitivity analysis is proposed to improve model interpretability, making it easier to identify input features that most affect the outputs and providing valuable insight into the underlying system, which cannot be obtained using typical black-box models.

- A diversity measure inspired by the linguistic complexity approach is implemented to track population diversity and provide information to aid in understanding evolutionary search behavior.

1.7 Outline of Thesis

Chapter 2 examines related work in energy prediction of Hadoop data centers and solar irradiance forecasting. Techniques introduced by these studies are appraised, and their limitations are highlighted. Gaps in the broader literature are discussed, and essential questions are raised in these areas.

The fundamental concept and design of the EVLNN architecture are detailed in Chapter 3. An illustration of the EVLNN framework and a description of the search algorithm are also discussed.

Chapter 4 investigates EVLNN's search capability through a set of benchmark test functions and compares the model's performance to modern meta-heuristic algorithms and the state-of-the-art niching EAs in the Congress on Evolutionary Computation (CEC) 2013 and 2015 competitions. The results are analyzed and discussed in this chapter.

In Chapter 5, EVLNN is applied to predict the energy consumption of a Hadoop cluster. This chapter also describes the Hadoop testbed setup and data collection process. The training of EVLNN is explained, and the energy consumption prediction by the model is evaluated and compared to ANNs trained using other EAs, namely PSO-NN, DE-NN, and GA-NN, respectively.

In Chapter 6, EVLNN is applied to forecast solar irradiance in tropical weather. Meteorological data such as wind speed, ambient temperature and relative humidity, and solar PV surface temperature are used as input variables. The model is trained on datasets with four resolutions to provide predictions at time steps of 1-minute, 15 minutes, 30 minutes, and 1-hour over a horizon of seven days. The use of a smaller scale of feature

subsets was investigated, with the performance of EVLNN in most prediction time-steps significantly better than PSO-NN, DE-NN, GA-NN, and TD-BPNN.

Finally, Chapter 7 concludes the research findings and recommends the direction for future research.

Chapter 2

2. Related Work

2.1 Introduction

Sustainability is becoming a growing concern for data centers due to their enormous energy demand. Although the past decades have seen significant improvements in data center energy efficiency through academia, research, and industry efforts [80], data centers still generate a high carbon-intense footprint. Transiting existing data centers energy demands to low-carbon energy sources and renewables is essential to combat climate and environmental effects. Decarbonizing data centers can boost a sustainable digital world and mitigate the impact of climate change. As more data centers are expanding their renewable energy strategy through Renewable Energy Certificates (RECs) and considerable Power Purchase Agreements (PPA) [81], grid operators must guarantee the physical delivery of clean electricity on the local grid. However, weather and atmospheric changes affect energy production planning, necessitating renewable energy forecasts. With their variability and limited predictability, it is ever more critical to study renewable energy forecasting for reliable and smooth integration of clean energy production into the existing power grid [82].

While data center energy research covers an extensive area, this chapter reviews existing literature on data center energy efficiency-related, specifically in the Hadoop cluster. In light of the increasing emphasis on solar energy due to its rapid growth over the last decade, this chapter also examines related work on time-series solar irradiance forecasting and its complex relationships between weather and meteorological variables [83].

2.2 Hadoop Energy Efficiency Studies

Hadoop data centers can scale into thousands of nodes in a single cluster, supporting hundreds of Petabytes (PB) data. The energy consumed by such a cluster would be massive. The intensive energy consumption of the Hadoop cluster presents significant opportunities for energy optimization and research. The data center energy consumption process is shown in Figure 2.1. The total energy consumed by a data center can be determined from its Information and Communication Technology (ICT) load and the facility load. The ICT load comprises energy consumption from servers, storage, and networks. The facility comprises heating, ventilation, air conditioning (HVAC) load, and power distribution load. Alongside Figure 2.1 are techniques for energy efficiency studies broadly discussed in the literature primarily classified into five categories [80], [84], [85], depending on the emphasis of the methodology used to address the energy problem. The categories are energy-aware workload placement and scheduling, energy proportionality, Dynamic Voltage and Frequency Scaling (DVFS), data replication and storage efficiency, and modeling using machine learning techniques. Approaches using energy-aware workload placement and scheduling, and energy proportionality make up the majority of existing studies. However, machine learning techniques are gaining popularity.

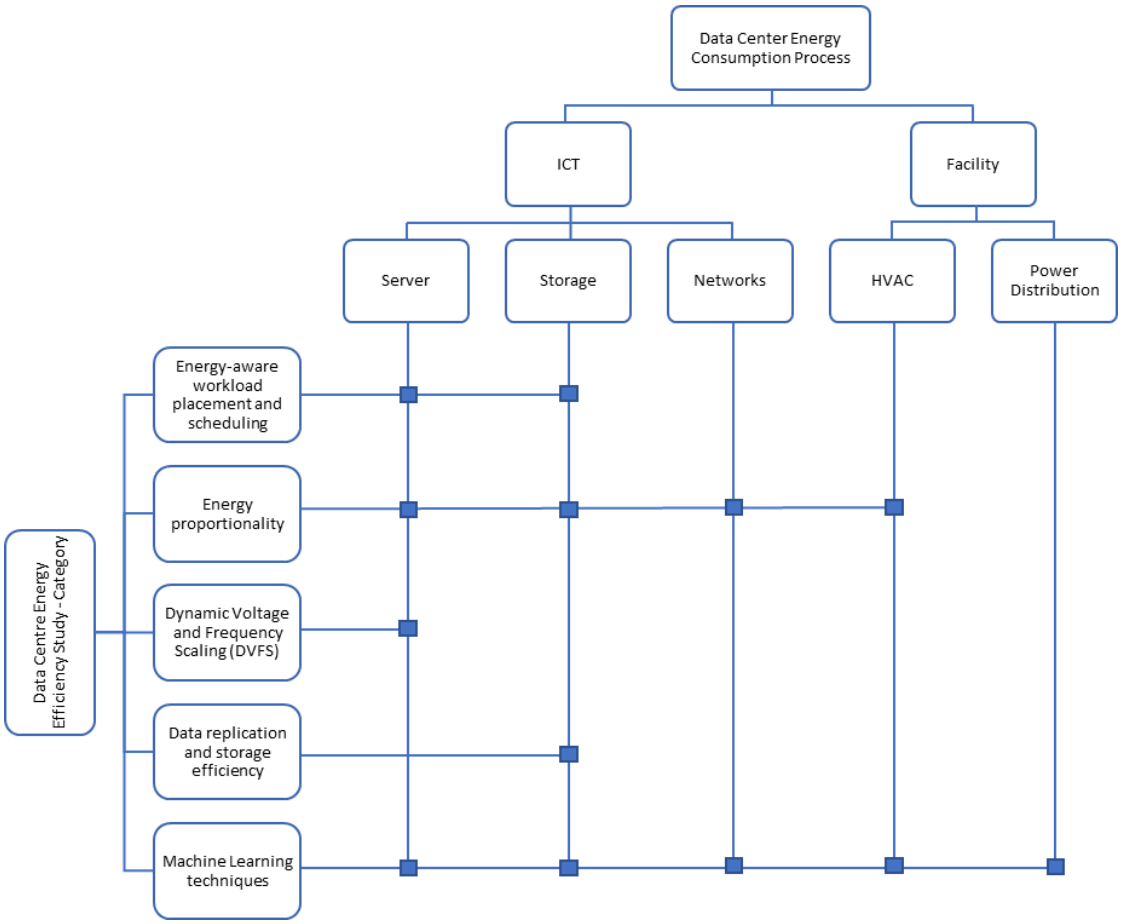


Figure 2.1 Categories of energy-efficient studies and their application in the data center energy consumption process.

2.2.1. Energy-Aware Workload Placement Scheduling

Researchers have long recognized the importance of energy-aware workload placement scheduling and energy proportionality. Most of the work in these fields focuses on using various techniques to schedule or place workloads on the most suitable nodes to minimize node overloads [86] [87].

Krish et al. [88] proposed a heterogeneity-aware and power-conserving task scheduler for the Hadoop workload scheduling algorithm. The energy performance and characteristics of Hadoop applications on different hardware platforms were first profiled using metrics such as CPU, memory, storage, and network usage by the Energy Profiler. The power

characteristics based on workloads of different sizes were extrapolated from the Energy Profiler to be analyzed statistically to determine a suitable resource-application match. A scheduler subsequently allocates the resources in the optimal sub-cluster to the workloads. A placement policy was proposed to ensure that at least one replica of the data is available in a small subset of nodes called the Covering Subset (CS). This strategy allows the turning off of other sub-clusters not part of the CS to conserve energy, and the approach helped reduce energy consumption by 21%. This approach poses a scalability issue as the ability to profile a dynamic and changing environment accurately becomes challenging.

Kulkarni [89] proposed scheduling tasks based on the thermal model in the data center to minimize the heat dissipated by the nodes and the cooling power overheads. The proposed method uses two thermal-aware schedulers to balance the temperature and reduce power consumption costs in the data center. The first is a dynamic scheduler that schedules jobs based on CPU and disk temperatures and utilization feedback given by the slave nodes. The second is a static scheduler that assigns the job to the slave nodes based on the jobs' profile, such as a CPU-intensive application, Disk intensive application, or both. When assigning the tasks to the nodes, the static scheduler would consider the CPU and Disk temperature at the time of scheduling to maintain the average CPU and Disk temperature across the Hadoop cluster. The new schedulers were implemented on top of Hadoop's FIFO queue system and improved energy costs by 10-15%. However, these two schedulers worked on different principles and were not integrated. Integrating the schedulers using job profile knowledge with utilization would offer greater potential for energy savings.

Mashayekhy et al. [90] modeled the MapReduce scheduling problem as a linear integer program and designed a greedy algorithm for solving this problem. The greedy algorithm called Energy-aware MapReduce Scheduling Algorithm (EMRSA) finds an energy-efficient assignment of the MapReduce tasks to the compute nodes. The EMRSA solution achieved an average energy savings of 32% and 40% for large and small-scale MapReduce jobs, respectively. While the approach can gain higher energy savings, it does so at the expense of job completion time. The drawback is that time-sensitive and deadline-driven tasks would be adversely impacted.

Shao et al. [91] introduced the Multidimensional Knapsack Problem (MKP) technique to model the Energy-aware Fair Scheduling framework (EFS) for MapReduce job placement. An Energy-Aware Greedy Algorithm (EAGA) is applied to solve the MKP problem and realize task placement on energy-efficient nodes. The approach combined workload scheduling with energy proportionality to achieve more significant energy savings by transiting inactive nodes to an idle state or turning them off after a threshold duration. The experimental results showed a savings of 2% to 14%. Nonetheless, this approach's weakness is that the job's execution time relies on the user request deadline to accurately compute the number of resources needed in the cluster. Without knowing the job completion time, the performance of the dynamic node management strategy will be greatly impacted.

2.2.2. Energy Proportionality

Other authors leverage the concept of energy proportionality and utilize techniques to 'right size' the data center. For example, a power-proportional server with 10% utilization should draw 10% of its maximum power. Leverich and Kozyrakis [92] presented Covering Subset (CS) strategy concept paper. CS is a group of nodes containing immediate data availability, even when all nodes outside this subset are disabled. An energy model was designed based linearly on CPU utilization to evaluate the power characteristics of the servers. It uses an energy-aware fair scheduler to assign Hadoop jobs to be performed over the minimum availability of all requested data in all nodes in the covering subset. At the same time, the rest of the nodes not in the subset can be gracefully powered down to conserve energy. This strategy had attained an energy reduction of between 9-50%. However, the high energy penalty for "awakening" the nodes from a powered down state countervails energy efficiency, reducing or neutralizing the energy savings.

Building on Leverich's work, Lang and Patel [93] proposed aggregating "live" data into subset nodes of the cluster and turning off nodes outside the CS subset. The proposed All-In Strategy (AIS) takes an extreme approach to the CS strategy to power down an entire cluster to achieve deeper energy savings. The CS strategy designates that the CS nodes are to be kept online to maintain at least one copy of each unique data block on these nodes.

Hence the authors altered the HDFS data placement policy so that during periods of low utilization, some or all of the non-CS nodes can be powered down to save energy. For example, if 33% of the nodes are CS nodes, then only the CS nodes are online at this utilization. Different offline states were experimented with, like hibernate, stop grant (low power state), and shutdown. Offline states still draw power, albeit low, because the motherboard and network card are still powered on for remote management. It resulted in the hibernate offline state being the most energy-efficient overall due to its low power consumption in that offline state and low power transitioning cost to the online state. The authors presented the energy consumption model of the MapReduce cluster, denoted as $E(\omega, \nu, \eta)$ is expressed as,

$$E(\omega, \nu, \eta) = (P_{tr}T_{tr}) + (P_{\omega}^{n} + P_{\omega}^{\bar{n}})T_{\omega} + (P_{idle}^{m} + P_{idle}^{\bar{m}})T_{idle} \quad (2.2)$$

where ω is the simplified workload characteristics, ν is the specified time windows when running the workload, η is the hardware characteristics, n and is the number of nodes running the job, $\bar{n}$ is *the* number of offline nodes during job processing, *m and* $\bar{m}$ are the number of online nodes and offline nodes in the idle period, respectively, P_{tr} is the average transitioning power, $P_{\omega}^{[n,\bar{n}]}$ is the on/offline workload power, $P_{idle}^{[n,\bar{n}]}$ is the on/offline idle power, T_{tr} is the time to transition nodes between power-up and power-down state, T_{ω} is the execution time of the workload, and T_{idle} is the idle time of the nodes. From Equation 2.1, energy consumption can be reduced by reducing idle energy consumption $\left(P_{idle}^{m} + P_{idle}^{\bar{m}}\right)T_{idle}$. However, it is observed in Equation 2.1 that T_{tr} can have a significant impact on energy consumption, if T_{ω} is small. It was reported that the AIS method achieved 26% more energy savings than the CS method by Leverich and Kozyrakis [92]. Nonetheless, a crucial aspect of AIS is the cost of transition T_{tr} and the negative secondary effects of degraded performance. Another drawback is that the workloads take longer to run as fewer worker nodes are available. However, the impact of the shortcomings can be mitigated with an accurate energy modeling and workload forecast system guiding the AIS energy management framework to keep optimal data availability at the highest energy savings possible. Such as how many nodes to power down and for how long.

Amur et al. [94] presented RABBIT, a power-proportional distributed file system (PPDFS) that uses data distribution in the HDFS to achieve power performance efficiency by ensuring a minimal number of powered-up nodes are active. RABBIT uses an equal-work data-layout policy for equal load sharing formulated as an optimization problem. The equal-work data-layout policy first established p nodes as primary nodes, with one replica of the dataset, called the primary replica, distributed evenly over the primary nodes. Thus, keeping the p nodes on is sufficient for data availability, and where $p << N$, the total number of nodes, gives RABBIT a low minimum power setting. The results showed that this approach had achieved power-proportionality as low as 5% of the nodes in an active state to service MapReduce jobs in the entire Hadoop cluster. However, the efficiency of an equal-work data-layout policy may suffer in a dynamic data center environment where adding or removing nodes for maintenance is expected. This approach will lead to high energy overheads due to its equal-work data-layout policy.

Lin et al. [95] proposed an energy-efficient and resilient data layout policy called eStor to address data disruption problems when turning off nodes. Le et al. [96] proposed a data placement method called Accordion, which uses data replication to arrange the data layout comprehensively and provide efficient gear-shifting. However, the approach will incur energy consumption overhead as the shifting of gears requires node reactivation during the transition. Kaushik et al. proposed GreenHDFS [97] and Predictive GreenHDFS [98], respectively, which rely on insightful data classification and energy-conserving placement policy to cluster nodes into various ‘hot’ and ‘cold’ activity levels and power down those ‘cold’ nodes with substantially long periods of idleness. However, the approach suffers from performance impact when transitioning nodes from off mode to active mode.

2.2.3. Dynamic Voltage and Frequency Scaling

Dynamic voltage and frequency scaling (DVFS) techniques are closely related to energy proportionality. These techniques adjust scalable power components such as CPUs by manipulating the operational frequency and voltage to regulate the power utilization in the processor based on the current load. The power consumption is mainly governed by the expression [99],

$$P = CV^2F \tag{2.3}$$

where P is the power, C is the switching capacitance, V is the supplied voltage, and F is the working frequency. Wirtz et al. [100] studied the effectiveness of DVFS in reducing energy costs by experimenting with three MapReduce workloads and applied three different DVFS scheduling policies with four frequencies: 0.8GHz, 1.3GHz, 1.8GHz, and 2.5GHz. Based on the results, DVFS scheduling could effectively improve energy. In particular for clusters with dominating idle power. However, DVFS does impose a performance penalty in return for energy cost savings, as extensive performance and energy profiling are required to achieve the balance.

Hou et al. [101] investigated the impact of DVFS on energy saving in MapReduce/Hadoop implementation on cloud platforms. The proposed method uses distributed online optimization algorithms to optimize the time-averaged energy consumption of MapReduce jobs performed in two scenarios. These scenarios are fog-assisted with DVFS (limited data processing at the edge) and fog-coordinated with DVFS (dispatching data at the edge to the cloud) simulated using MATLAB software. The results indicated that the second scenario offered 32% higher energy savings than the first. Nonetheless, these works have assumed that the processors of the active servers operate at the same frequency and that raw data arriving at the fog nodes are identically distributed. Real-world deployment of Hadoop clusters in data centers is expected to host inhomogeneous computing elements with varying compute demands. Hence, it is unclear if the simulation results are close to or similar to the real world.

2.2.4. Data Replication and Storage Efficiency

Replication is a common method applied in the Hadoop cluster to ensure redundancy. However, making multiple copies (the standard is three) of data blocks in the HDFS expands the storage capacity, which leads to increased energy consumption. Fan et al. [102] proposed replacing HDFS default replication of three with erasure coding to improve spatial and energy efficiency. The approach combines Redundant Array of Independent Disk 5 (RAID 5) and mirroring encoding with RAID 6 erasure encoding to lower triplicated

overheads to RAID-class redundancy. In this approach, storage overheads are reduced to $1 + \frac{1}{N}$ and $\frac{2}{N}$ respectively, where N is the number of repica blocks. However, the encoding period can be delayed if "hot" files in the Hadoop cluster are continually accessed. Cheng et al. [103] applied the Complex Event Processing (CEP) technique to analyze HDFS audit logs. Data are first distinguished into 'hot' and 'cold' types and then placed into active or standby storage in HDFS using replication or erasure coding. The erasure coding approach of adding redundancy in HDFS provides high reliability at a fraction of the cost of replication. However, erasure coding requires complex central management systems that affect HDFS's overall performance and interoperability. Wei and Foo [104] successfully integrated Low-Density Parity Check (LDPC) coding into HDFS to improve storage efficiency with a less complex optimized equation-based repair algorithm. The basic principle behind this technique is to incorporate the Quality of Service (QoS) policy from the users as the first criterion for automatically adjusting and configuring the parameters related to the LDPC code [105]. The approach allows the existing resources to be used more efficiently to meet the time-sensitive requests allowing other applications access to the storage. While LDPC coding can reduce the storage footprint much further than erasure coding, the tradeoff is repair or decode overhead during node failures. The decoding complexity increases as the code length become longer, adding to the data recovery time.

2.2.5. Modeling using Machine Learning Techniques

The use of machine learning (ML) modeling techniques such as Artificial Neural Networks (ANN) for energy consumption modeling and prediction has gained renewed interest among data center energy researchers in recent years. Wang and Cao [106] proposed a control model that dynamically adjusts the energy ratio (power budget) at a predefined level to ensure performance goal is met. The control model's core is the Model Predictive Control (MPC) strategy and a feedback mechanism to keep the operation within the power budget. The MPC is based on fuzzy wavelet neural networks (FWNNs) to predict the nonlinear relationship to adapt to the dynamic workload. A large Hadoop cluster computing state sample is collected to form the training dataset. The trained model is deployed for energy efficiency ratio prediction. The control system uses the forecast information to conduct

CPU frequency scaling for each node in the Hadoop cluster. The combination of prediction and control allows this approach to increase energy efficiency by 5% to 12.6%. However, the structural design of FWNN is reliant on professional knowledge and practical experience, which can be time-consuming.

Liang and Hu [107] established a deep learning model called Multiple Energy-Related Features (MERF) to predict energy consumption using multiple energy-related features acquired from a Hadoop testbed. The energy-related features are *CPU utilization*, *Average Load*, *Memory Use*, *Map Read Task*, *Reduce Read Task*, *Map Write Task*, *Reduce Write Task*, *Network I/O Speed*, *Shuffle Size*, *File Size*, *Number of MapReduce Instructions*, *Disk Utilization*, *Transmission and Read/Write Ratio*, *Available Space in File System*, *Page Faults/sec*, *Byte Consumed per CPU second*, *Context Switches Rate*, *HDFS R/W Throughput*, *Disk Traffic*, and *Energy Consumption*. The energy-related data is acquired by running MapReduce jobs and combining several open-source performance monitoring applications and tools such as Ganglia, Nagios, Zabbix, and Hadoop build-in counter for the acquisition work. The architecture of the deep learning model using a Deep Neural Network (DNN) was empirically determined, consisting of one input layer, three hidden layers, and one output layer. Each layer consists of hidden neurons 12-100-100-100-1. The energy-related features were selected using the Kullback-Leibler (KL) divergence, which measures the difference between two probability distributions. The MERF model outperformed five other machine learning models with higher prediction accuracy to offer a solution for improving energy efficiency. Nonetheless, DNN requires a large dataset for training, and the manually designed architecture can be inefficient and sub-optimal.

Toha et al. [108] investigated different machine learning techniques, such as K-Nearest Neighbor (KNN), Support Vector Regression (SVR), and Additive Regression (AR), to predict the MapReduce cluster's total energy consumption, consisting of computational energy and cooling energy. The techniques using SVR and AR had achieved 97% accuracy. Using the prediction information, two methods, namely the Green MapReduce Cluster (GMC) and enhanced GMC (eGMC), were proposed to determine the number of active servers achieving an energy reduction of up to 47% to 76%. In their experiment, the authors

used MapReduce Wordcount applications to generate workload on the cluster and provided the Wikimedia dataset [109] as payloads for the Wordcount application. A total of 68 jobs with an average job size of about 150 MB were used in the experiment. By executing the workloads, data such as the response time, computational power, and cooling power over the same series of jobs were collected. The datasets collected were used to train the machine learning models to determine the minimum number of computing nodes needed to be activated. Though the high energy savings presented were apparent, the results were based on simulated data and achieved at the expense of performance degradation in some cases. The metrics used for performance measurement were response time, throughput, and waiting time. GMC experienced lower throughput, longer response time, and longer waiting time than other methods in their experiment.

A summary of prior work reviewed is shown in Table 2.1.

Table 2.1 A summary of prior work in Hadoop energy efficiency studies.

Author	Category	Method	Results	Remarks
Krish [88]	Energy-aware job scheduling	Applied application profiling and statistical analysis to find a suitable application-resource match.	Power savings of 21%.	This approach poses a scalability issue as the ability to profile a dynamic and changing environment accurately becomes challenging.
Kulkarni [89]	Energy-aware job scheduling	Applied thermal modeling and performed dynamic scheduling based on CPU, disk temperature, and utilization of the nodes, and static scheduling based on job profiles.	Improvement of 10-15% of energy cost.	The two schedulers worked on different principles and were not integrated. Integrating the schedulers would offer more significant potential for energy savings.
Mashayekhy [90]	Energy-aware job scheduling	Modeled the scheduling problem as an Integer Program, then designed a greedy algorithm called EMRSA to solve the problem.	40% less energy consumption on average for small MapReduce jobs.	The approach would affect time-sensitive and deadline-driven tasks.
Shao [91]	Energy-aware job scheduling	Modeled the scheduling problem as a Multidimensional Knapsack Problem, then designed a greedy algorithm called EAGA to solve the problem.	2% - 14% less energy consumption.	Requires knowing the workload completion time to calculate the resources required in the cluster.
Leverich [92]	Power-proportionality	Introduced covering subset (CS) for servers and applied energy-aware fair scheduling to assign tasks to the CS.	Energy savings of between 9% to 50%.	High energy penalty for “awakening” the nodes from a powered down state.
Lang [93]	Power-proportionality	Modeled the energy consumption of Hadoop and proposed an ‘All-in-Strategy’ (AIS) to power down the entire cluster.	Achieved 26% more energy savings than in [92].	High transition overhead to wake up nodes and performance degradation during the transition phase.
Amur [94]	Power-proportionality	Used a load balancer to complement the equal-work policy that ensures each active	Power proportionality achieved as low as	Incur management and performance overhead in data rebuild parallelism.

Author	Category	Method	Results	Remarks
		node services the same number of blocks.	5% of active nodes.	
Lin [95]	Power-proportionality	Proposed replication strategy.	Demonstrated 40% of energy savings.	Require analysis of data rebuild parallelism to reposition lost data
Le [96]	Power-proportionality	Applied data placement method.	Improve by 20% compared with [94].	Shifting of gears requires node reactivation during the transition.
Kaushik [97]	Power-proportionality.	Applied data-classification technique known as GreenHDFS by partitioning cluster servers into hot and cold zones then putting cold servers to sleep.	Achieved 26% energy savings.	Access to files residing in a cold zone may suffer performance degradation as servers transition from off mode to active mode. There is also an energy penalty cost to wake up the servers.
Kaushik [98]	Power-proportionality	Incorporated predictive ability into GreenHDFS using ridge regression to guide file placement.	Improved average response time by 40% compared to GreenHDFS in [97].	An aggressive policy can result in a higher number of access to the cold zone decreasing energy savings.
Wirtz [100]	DVFS	Applied DVFS scheduling policy.	Reduce energy by up to 23%	Do not have a method to evaluate performance versus power in scheduling.
Hou [101]	DVFS	Proposed a data dispatch strategy at the edge of the cloud.	Increased energy efficiency by 32%.	The method might not work in an inhomogeneous computing data center.
Fan [102]	Data replication and storage	Applied Erasure coding.	Energy savings of 70%.	The encoding period can be delayed if "hot" files are continually accessed.
Cheng [103]	Data replication and storage	Used Complex Event Processing.	Reduce storage overhead.	Need to operate in real-time to schedule the placement of data effectively.
Wei & Foo [104] [105]	Data replication and storage	Applied LDPC coding.	Achieved 56.7% storage savings.	A promising alternative to erasure coding such as Reed-Solomon
Wang [106]	Machine learning	Proposed Fuzzy wavelet neural network (FWNN) trained to predict changing workloads to scale and control CPU frequency dynamically	Energy efficiency increased by 5% to 12.6%.	The FWNN structure design relies on professional knowledge and practical experience, which can be time-consuming.
Liang [107]	Machine learning	Applied Deep neural network (DNN) trained with energy-related features to predict the energy consumption of Hadoop cluster	High predictive accuracy using KLIC for essential feature selection	Requires large training dataset and the trial-and-error approach to DNN architecture search is time-consuming and sub-optimal.
Toha [108]	Machine learning	Proposed Green MapReduce Cluster (GMC) and eGMC (enhanced) methods using Support Vector Regression (SVR) and Additive Regression (AR) for predictive analytics.	Models achieved an energy reduction of between 47% to 76%.	The results were based on simulated data and achieved at the expense of performance degradation in some cases.

2.3 Summary of Hadoop Energy Efficiency Studies

Data centers already consume vast amounts of energy. The rapid acceleration of digital transformation will multiply this further. The state-of-the-art energy efficiency methods in

Hadoop are discussed, and their respective advantages and disadvantages are presented. The energy-aware job scheduling method improves the utilization of resources through workload placement to reduce the idleness of nodes. In some approaches, job profiling techniques are incorporated to obtain the essential job information in energy-efficient scheduling algorithms. In other approaches, it is accompanied by shutting down idle nodes to achieve deeper energy savings. However, most studies lack the usage of accurate predictions to essentially anticipate the changing workload environment or avoid unnecessarily high energy overheads in starting or waking up nodes.

The DVFS method adjusts the CPU state for optimal energy performance according to the changing workload. Nevertheless, in a heterogeneous environment, the offered resources and energy performance ratio may differ from machine to machine, making the DVFS tuning complex and challenging. It may be possible that the complexity can be mitigated by introducing a predictive model to offer insights into changing workload and utilization trends. The methods proposed in data replication and storage efficiency have distinguished properties such as high data locality and low node idleness. However, if data can also be marked or predicted as ‘hot’ or ‘cold,’ data availability and parallelism in Hadoop data centers could be further improved with differentiated replication strategies and erasure coding implementation.

Machine learning techniques have shown great potential in enhancing energy efficiency. ANNs models for energy prediction for data centers have become increasingly popular. Whether using a deep network with many hidden layers like CNN or a shallow network with one hidden layer largely depends on the problem instance. For CNN, there is the issue of solution complexity and the possibility of over-parameterization of the network. In the energy prediction applications for this work, shallow ANNs are more effective.

Nonetheless, it is recognized that these networks are powerful learning machines that tend to overfit the training dataset. Implementing an evolutionary architecture search approach could prevent this drawback by locating a parsimonious ANN that can provide a more generalized solution. In addition, the reduced architectural complexity of shallow ANNs increases model transparency making these networks more interpretable than deep neural

networks. Model interpretability is essential in studying feature importance that affects energy consumption. Finally, a closer look reveals a paucity of literature on EA-based ANNs techniques for Hadoop energy efficiency study. To address the literature gap, a novel EVLNN design based on parsimony has been proposed to model the Hadoop system's energy consumption and gain insights into the underlying factors influencing its energy consumption.

2.4 Solar Irradiance Forecasting Studies

Despite the rapid increase in solar power penetration, solar power's intermittency remains an issue that must be addressed. Solar power intermittency is due to two factors. Firstly, the "variability" or the fluctuation in solar irradiance caused by changing atmospheric conditions and cloud formations. Secondly, the "uncertainty" of the electricity generation which is ascribed to the inability to accurately forecast solar energy generation by the minutes, hours, days, or longer. Therefore, solar energy prediction at various temporal resolutions and forecast horizons is essential in scheduling and optimizing energy production.

The methodologies for solar power forecasting involve a two-stage approach. The first stage is to forecast the solar irradiance at the surface level. The second stage is to transform the predicted irradiance into power production by modeling the solar PV power system or using the information to directly calculate the system's power output. Hence the challenge of solar PV power forecasting has been essentially recognized as similar to solar irradiance forecasting. Many studies only focus on solar irradiance forecasting since it is the most challenging element to model. The other reason is that solar radiation forecasting may be applied in fields other than solar PV energy planning and optimization.

Another important concept of solar irradiance forecasting is the temporal aspect of forecasts. There are three concepts in the time definition of forecasts in the literature; the forecast horizon, the forecast resolution, and the forecast intervals [110]. The forecast horizon is the

length of time ahead of the predictions. The forecast resolution, or time-steps, describes the forecast's frequency, and the forecast interval denotes the time range of predictions. The predictive accuracy of different forecasting models is different, influenced by the different temporal resolution of input data and forecast horizon of output data [111]. Different forecasting classification is also related to the types of applications, and it is efficient within its range of time resolution and forecast horizons. For example, ramp forecasting and spinning reserves require an intra-hour forecast horizon with seconds to minutes resolution. Grid operations require an intra-day forecast horizon with minutes to hours resolution, and scheduling and electricity market trading and hedging require a day-ahead forecast horizon with hourly resolution.

The literature review shows that solar forecasting methods can be broadly classified into Numerical Weather Prediction (NWP) models, satellite imaging, Total Sky Imagery (TSI), statistical analysis, and machine learning and AI. Several works, namely by Antonanzas et al. [110], Inman et al. [112], Diagne et al. [113], and Yang et al. [114], have provided a concise review and trend on solar forecasting techniques. The primary use of existing methods and their respective applications are illustrated in Figure 2.2 [113]. Different methods are used for different forecast horizons where each method is efficient. TSI uses images of cloud cover taken from the ground camera at intervals of several minutes. Image processing and cloud tracking techniques are applied to these images for real-time or now-time forecasting for up to 30 minutes. Satellite imaging uses cloud images observed by satellites and then applies the Heliosat method to estimate solar surface irradiance [115]. The temporal resolution is about 30 minutes due to the time required to download the images. Image processing using satellite imaging is slower than the TSI, with a forecast horizon of an hour to several hours, up to a day ahead. Statistical analysis methods use past solar irradiance time series and meteorological data to predict future solar irradiance. Depending on the data acquired, temporal resolution can range from minutes to an hour for a forecast horizon from minutes to hours. NWP methods use current weather observations to predict the future states of the weather, where the outputs include temperature and irradiance. NWP models are the best fit for longer forecast horizons. Machine learning and AI, such as ANN and deep learning models, use a data-driven approach to learn hidden

patterns in data. Such models have complex structures that can learn nonlinear relationships between the input and output data to allow the prediction of output data for a given input. These powerful models have a wide range of applications and forecast horizons.

The growing interest in applying machine learning and AI models for solar irradiance forecasting is an interesting observation in the literature. These methods, which rely on a data-driven approach, can draw on extensive data sets to derive insights into the correlations between all available parameters, inputs, and outputs, making them suitable for deployment across various time resolutions and forecast horizons.

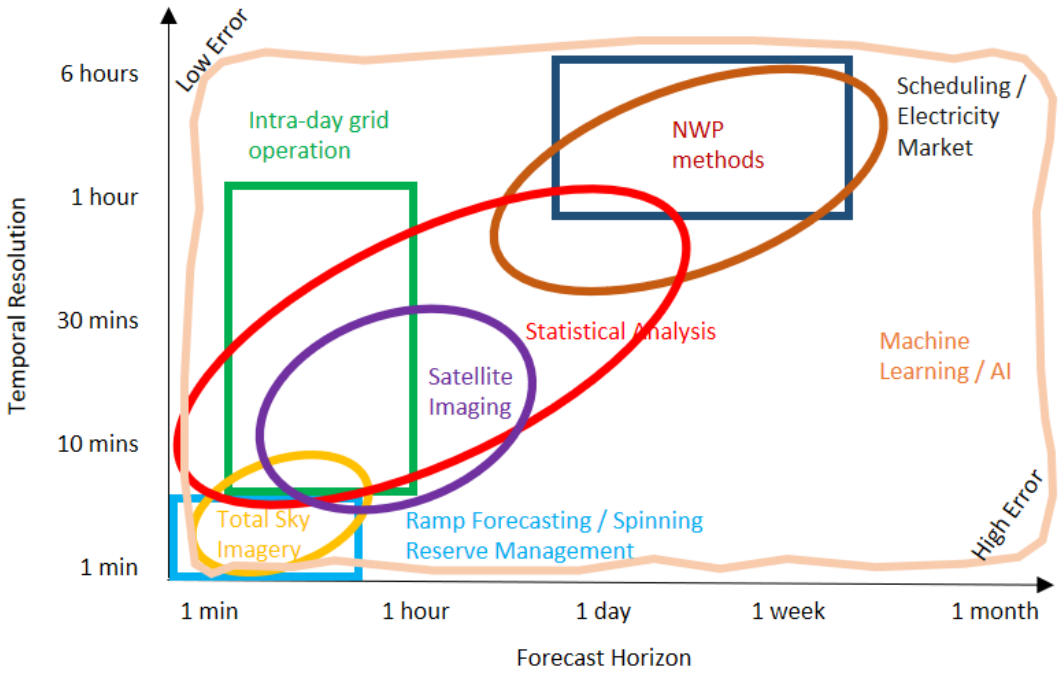


Figure 2.2 The figure shows the classification of the forecasting methods and their application in the temporal resolution and forecast horizon coverage. A higher forecast horizon leads to a higher error rate.

2.4.1 Numerical Weather Prediction Methods

NWP provides solar irradiance forecast weather conditions up to several days ahead using current weather conditions as input into mathematical models. The core methodology in NWP-based forecasting uses atmospheric data and the law of physics to govern the transition of the atmosphere from one state to another to predict solar radiation. Input

variables are meteorological measurements such as wind, temperature, humidity, and surface pressure; the forecast variable is solar irradiance. Selected well-known NWP models are the Global Forecast System (GFS) model and the North American Mesoscale (NAM) model by the National Centers for Environmental Prediction (NCEP), Regional Deterministic Prediction System (RDPS) model by the Canadian Meteorological Centre (CMC), and the Weather Research and Forecasting (WRF) model by the National Oceanic and Atmospheric Administration (NOAA). Yang et al. [116] applied the GFS model to capture the movement of clouds. Values of meteorological variables in the lower atmosphere were collected to build mathematical models for cloud predictions. The author observed that surface downward longwave flux had a positive bias tracked back to errors in the surface air temperature and proposed a method to correct the errors and improve the cloud fraction estimation. Larson et al. [117] evaluated the NAM and the RDPS models to forecast cloud fractions over a 12 x 12 km^2 spatial grid with hourly temporal resolution.

Additionally, predicted water vapor, Ozone, Carbon Dioxide, and aerosol concentrations were used to forecast GHI. An averaging method was adopted where irradiance forecast within a set distance of 100 km was spatially averaged to reduce forecast errors. Jimenez et al. [118] augmented the WRF model by incorporating cloud-aerosol interaction and cloud-radiation feedback to estimate surface irradiance accurately. Meteorological variables and effects of atmospheric aerosol were used to estimate direct and diffuse surface irradiance. The study showed that the influence of atmospheric aerosol components in clear-sky conditions is evident.

NWP's temporal resolution ranges from three to six hours, and spatial resolution ranges from 1 km to tens and hundreds of km. While NWP's forecast horizon can reach a day or several days, the accuracy is generally low, with significant biases and random errors in the irradiance estimates as the atmospheric conditions are chaotic [119]. In addition, the method, relative to other methods, is more sensitive to initial conditions resulting in models' forecasts which can diverge widely from each other. Improving NWP models requires good, relevant observation data combined with field campaigns [82]. Therefore, new models of NWP are still being tested and applied to operational NWP [120].

2.4.2 Satellite Imaging

The extraterrestrial solar radiation incident outside the Earth's atmosphere is a constant known by the symbol, G_{sc} where $G_{sc} = 1{,}361 \frac{W}{m^2}$. This radiation from the Sun is attenuated as it passes through the Earth's atmosphere [121]. The presence of clouds, water vapor, and aerosols in the atmosphere diffuses part of the incoming solar radiation resulting in the division of the incident extraterrestrial beam irradiance into two distinct components, Direct Normal Irradiance (DNI) or beam irradiance and Diffused Horizontal Irradiance (DHI). The DNI radiation reaches the Earth's surface without being absorbed or scattered. In contrast, the DHI is radiation scattered or diffused by the atmospheric constituents. Some radiation is also scattered and reflected off the Earth's ground or water surfaces which is then reflected or re-scattered by the atmosphere to the observer. This also forms part of the DHI. The Global Horizontal Irradiance (GHI) received at the Earth's surface is then the sum of the DHI and the DNI incident on the normal surface expressed as [122],

$$GHI = DHI + DNI * \cos\theta \tag{2.4}$$

where θ represents the solar zenith angle when the Sun is directly overhead, θ is zero. Therefore at $\theta = 0$, the radiation exposure to the surface is most extensive. The incident radiation attenuated as it negotiates its way to the ground level through a complex series of multiple reflections, absorptions, and re-emissions due to interactions with atmospheric constituents is illustrated in Figure 2.3. From the figure, cloud cover and cloud optical depth (thick or thin) have the most decisive influence on solar irradiance at the surface level [113].

In satellite imaging, images are provided as grayscale images. Each image pixel represents the radiance signal of solar radiation backscattered to space by the atmospheres and clouds. Pixels are then assigned to the respective classes based on their pixel brightness. Tarpley [123] defined three cloud classes: clear, partly cloudy, and cloudy. The number of pixels in each class is weighted to obtain fractional cloud cover and thus the estimated cloud thickness at the targeted area. The amount of solar radiation can then be forecasted as absorbed and scattered by clouds (Rayleigh scattering) and aerosols (Mie scattering), which is a function of clock thickness. Cloud motion vector fields were also determined to predict

irradiance for the next time-step. Hammer et al. [115] analyzed the satellite images for information on cloudiness. They used statistical means to derive the Cloud Motion Vectors (CMV), where the cloud movement is extrapolated to forecast GHI. Satellite imaging can provide up to several hours of solar irradiance forecasting with high spatial resolution. Rigollier et al. [124] proposed Heliosat-2, a method of analyzing Meteosat images that takes various empirical parameters presented in the Heliosat-1 method and expresses them into physical laws. This approach ensures a broader application as no ground measurement is required. While cloud movement obeys physical rules, atmospheric processes are turbulent, making them stochastic and challenging to model.

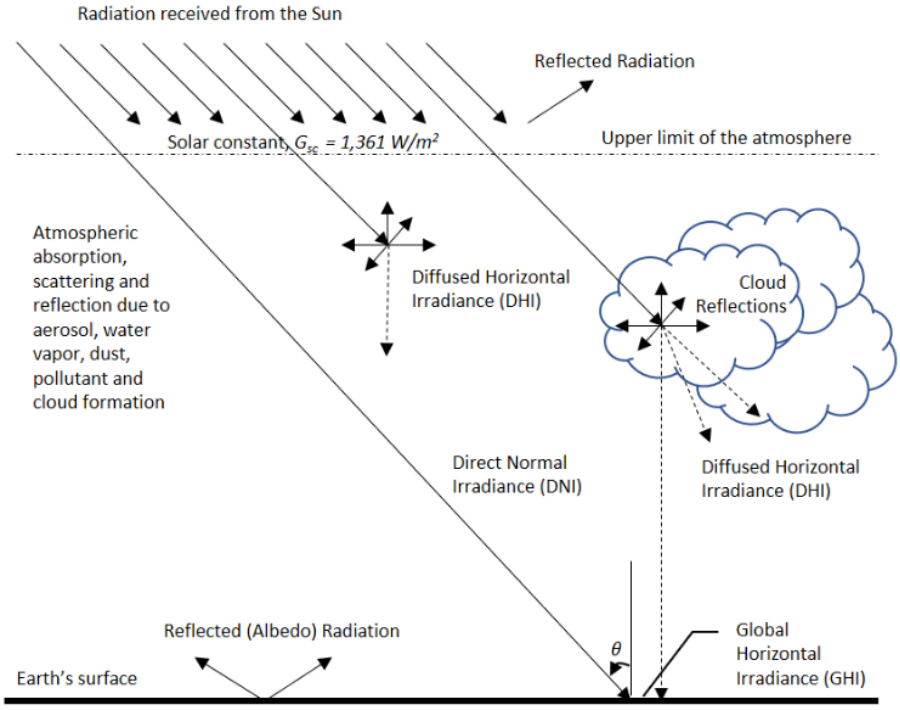


Figure 2.3 The solar radiation components consist of GHI, DHI, and DNI. The solar zenith angle θ is the angle of the Sun relative to the line normal to the Earth's surface.

2.4.3 Total Sky Imager

Like satellite imaging methods, Total Sky Imager (TSI) methods analyze the cloud by combining sky photos with image processing and cloud tracking techniques for real-time forecasts of GHI and DNI up to 30 minutes ahead. Instead of using satellites to obtain cloud imagery, ground-based TSI cameras are usually mounted on building roof-top to take a

360° view of the images of the sky above. The sky images are processed, and information on the ratio of the blue and red colors is used to distinguish whether the pixel presents a clear sky, thin clouds, or opaque clouds through the estimation of cloud cover fraction [125]. Successive images are analyzed for sky cover, cloud motion, and cloud shadows to derive cloud motion vectors where these cloud images are propagated forward in time to establish a correlation with solar irradiance resulting in a forecast [126]. Marquez and Coimbra [127] proposed a method to process sky images, classify them into clear, thin, opaque clouds, and compute cloud velocity to generate intra-hour irradiance prediction. Chow et al. [128] presented a cloud detection technique using the red and blue color ratio to estimate whether the dominant source on a pixel is the clear sky or cloudy. Sky images taken every 30 seconds were processed to determine cloud cover. Solar irradiance is then estimated using the derived cloud cover coupled with an empirical clear sky model [129]. CMV was generated from consecutive sky images to predict cloud locations up to 5 minutes ahead through cross-correlating consecutive sky images using a network of 6 ground pyranometers over 2 km^2. Yang et al. [130] processed the sky images taken at 30-second intervals to determine cloud cover, optical depth, and mean cloud field velocity. Information from several deployed pyranometers and frozen cloud advection were used to forecast cloud locations for up to 15 min horizons.

Central to the methodology is the camera's resolution of the pictures taken, such as the pixel elements, that would determine the accuracy of the cloud's distance from the imager. In addition, a limited range view from the groud-based imager provides only local meteorological information restricting the forecast horizon to approximately 30 minutes. It is possible to deploy an array of ground imagers to gather more information on local cloud formation and extend the forecast horizon. However, the stochastic nature of the local cloud formation may limit the correlation of successive cloud images and pose difficulties for ground-based imagery methods. In addition, to the challenges with cloud detection, forecast errors are more pronounced within the circumsolar region due to the distortion of the pixel's blue and red color ratio caused by the forward scattering of sunlight by the presence of aerosols [131].

2.4.4 Statistical Models for Solar Energy Forecasting

Statistical models are commonly used to investigate variable relationships, patterns, and trends using historical solar-energy-related data. Bacher et al. [132] presented an approach first using a statistical smoothing technique to derive a clear sky model, then applying Autoregressive (AR) and the AR with Exogenous input (ARX) models to forecast solar power. The AR model had only the lagged values of solar power as inputs, whereas the ARX model added exogenous data from the NWP model as inputs. The ARX model produced better results, using solar power observations and NWPs as inputs. Li et al. [133] investigated the Autoregressive Moving Average (ARMA) model and the ARMA with Exogenous variables (ARMAX) model, in which the latter used the average of the past data and external variables as the predictors such as temperature, precipitation amount, insolation duration and humidity, to forecast solar power output. Chu et al. [134] proposed the Smart Autoregressive Moving Average (ARMA) model using lagged past values and errors to forecast solar irradiance. The Smart ARMA was optimized using a Genetic Algorithm (GA) trained ANN to determine the variables as inputs. Although the approach had corrected forecast bias and yielded better results than the ARMA model, particularly at higher solar variability, the performance was not significantly superior to the reference model at moderate to low solar variability.

Gagne et al. [135] evaluated different statistical learning models, including gradient boosting, random forest, and linear regression models for solar irradiance forecasting. Cloud cover and aerosol information from NWP models were applied as input variables to the Model Output Statistics (MOS) approach. The authors incorporated other climatological information concerning spatial and temporal variability of each variable into the statistical learning models for correction at each observation. Nonetheless, the parameter configuration is an issue in this approach as it dramatically impacts model performance, leading to significant investment in model tuning. Diagne et al. [136] downloaded the forecasts of the GFS from the NCEP website and used them as input to the WRF model to predict solar irradiance. The predictions were compared with the observations on the ground to find the statistical relation and correct the error. However,

the approach required large and high-quality ground measurements relative to other models to evaluate the forecast accuracy and refine the model. Noia et al. [137] compared four models, Tarpley [123], Hay and Hanson [138], Justus et al. [139], and Cano et al. [55], that utilized the MOS approach in the satellite imaging methods. This method includes independent predictors such as meteorological data, atmospheric transmissivity, and satellite image pixel brightness in regression equations to predict solar radiation. The drawbacks of these approaches are that it requires high-quality ground-based data for verification, and tuning may cause forecasting errors due to the localization of the pyranometer sites on the satellite images.

2.5 Machine Learning and AI for Solar Energy Prediction

2.5.1 Artificial Neural Network Models

The research and application of ANN models for solar irradiance forecasting have gained renewed interest in recent years [140]. Crisosto et al. [141] trained a Levenberg-Marquardt (LM) Backpropagation Neural Network (LM-BPNN) with four years of data using all-sky images and measured global irradiance as input variables. The model could predict the irradiance in the first 10-30 minutes better than the reference persistence model. However, the accuracy is dependent mainly on the weather conditions beyond this forecast horizon. Faceira et al. [142] trained Multi-Layer Perceptron (MLP) based on the LM algorithm using cloudiness level and historical solar radiation data to predict hourly Global Solar Irradiance (GSI) by one day ahead. Model accuracy was measured using Mean Bias Error (MBE) and Root Mean Square Error (RMSE) with a t-test. Although architectures with up to 3 hidden layers were experimented with, no significant differences were observed in the performance of ANN with deeper layers. Nurcahyo et al. [143] presented a weather forecast system built using hybrid GA to optimize the weights and connections of Partially-Connection Neural Networks (PCNN).

The study showed that trained PCNN had a Mean Absolute Percentage Error (MAPE) testing result of 35.53% compared to a Fully Connected Neural Network (FCNN) with

results of 38.82%. From this standpoint, it can be considered that PCNNs exhibited better generalization ability than their dense counterparts. Su et al. [144] proposed a hybrid ANN ensemble approach that uses a weighted average to combine predictions of the other ANN techniques into a single forecast, as shown in Table 2.2. The performance of hybrid ANN and the other ANN techniques was evaluated based on a six-day-ahead PV power forecast at half-hour resolution. The results were as follows: Hybrid ANN, NARXNN, ENN, ANFIS, ELM, GRNN, GABPNN, and BPNN. It must be pointed out that ANNs with dynamic feedback mechanisms such as the NARXNN and ENN were superior and that an evolved NN like GABPNN had provided a more accurate forecast than the standalone BPNN. It is also observed that non of the ten individual methods was superior in all four seasons.

Table 2.2 Su et al. [144] proposed a hybrid ANN method compared to seven other ANN models.

Comparison with Seven other ANN models	**Hidden Neurons**
1. Extreme Learning Machine (ELM)	4 to 20
2. Backpropagation Neural Networks (BPNN)	4 to 14
3. Nonlinear Autoregressive Neural Network with Exogenous Inputs (NARXNN) with 1:2 time-delay	10
4. GA optimization BPNN (GABPNN)	11
5. Adaptive Network-based Fuzzy Inference System (ANFIS)	Randomly initialized
6. Generalized Regression Neural Network (GRNN)	Same number as inputs
7. Elman Neural Network (ENN)	4 to 8

2.5.2 Deep Learning Models

Another popular ANN-based method in the literature includes the use of deep learning for energy forecasting [145] [146], such as the Convolutional Neural Networks (CNNs) and the Long Short-Term Memory (LSTM) networks. Mulyadi and Djamal [59] employed CNN to predict sunshine duration with weather features consisting of temperature, humidity, and solar radiation length. The model achieved high accuracy of 98.84% for the training dataset but dropped to 54.55% for the new dataset. The apparent gap suggests strong evidence of overfitting. In addition, CNN training remains time-consuming because most parameters, such as the filter size, the number of filters, and the pooling size and

methods, are determined by empirical means. Different configurations can affect accuracy by as much as 50%. Capizzi et al. [60] employed wavelet-transformed to enhance Recurrent Neural Network (RNN) predictions for a 2-day solar radiation forecast. The model exploits the correlations between time-series solar radiation and meteorological variables such as wind speed, humidity, and temperature. The model produced a very low RMSE error compared to models obtained by hybrid ANNs. However, RNN is constrained by the vanishing gradient problem, especially when the dataset is large. Abayomi-Alli et al. [147] applied LSTM models successfully to predict solar radiation using weather and geographical variables. The inputs used were Longitude and Latitude, elevation, maximum and minimum temperature, precipitation, wind speed, relative humidity, tropospheric ozone, and solar radiation. High R^2 values between actual and weather variables were also exhibited for solar radiation, maximum temperature, wind speed, relative humidity, and precipitation. The models achieved a prediction accuracy between 99.3% to 99.9%. Muhammad et al. [148] applied LSTM models to predict hourly and daily solar irradiance for a year ahead. The model achieved an MSE value of 10.4% in the hourly prediction for a year ahead. Nonetheless, LSTM models face similar challenges as the other deep learning methods, requiring manual tuning of many hyperparameters, such as epochs, layers, neurons, and optimizers, which is time-consuming.

2.5.3 Hybridized Deep Learning Models

Hybridized deep learning approaches in global solar radiation and PV forecasting have also gained popularity, for example, the integration of CNN with the LSTM network to form a CLSTM network. Ghimire et al. [149] proposed a hybrid CLSTM model trained using half-hourly interval global radiation data for multi-step forecast horizons of 1-day up to 8 months. The results showed that the hybrid CLSTM model outperformed the standalone CNN and LSTM. It also performed better than other models such as Gated Recurrent Unit (GRU), Recurrent Neural Network (RNN), Deep Neural Network (DNN), MLP, and Decision Tree (DT), with the lowest RMSE, MAE, and MAPE values for forecast horizons of 1-day, 1-week, 2-week, and 1-month. Similar outcomes were also seen by Wang et al. [150], who examined the hybrid CLSTM model for 1-day ahead PV power forecasting, where the best results were produced using three years of the input data sequence.

Compared with the LSTM model, the RMSE, MAE, and MAPE of the CLSTM model decreased by 13.82%, 30.39%, and 31.25%, respectively. Compared with the CNN model, RMSE, MAE, and MAPE decreased by 6.54%, 10.00%, and 12.00%, respectively. However, one of the CLSTM model's shortcomings is the high computational cost, which increases significantly with the filter size. Additionally, the training data size must increase to be large enough to cover all features.

2.5.4 Evolutionary-based ANN models

The Evolutionary Algorithm (EA), with its bio-inspired metaheuristic search ability, has been widely used to find an optimal set of network parameters from a population of candidate solutions. Particle Swarm Optimization (PSO), Differential Evolution (DE), and Genetic Algorithm (GA) are well-known classes of EA applied to various optimization problems [131]. Dong et al. [151] proposed a novel CNN combined with a hybrid chaotic GA/PSO algorithm to optimize the CNN networks' hyperparameters for solar irradiance prediction. Named CHA-CNN, the algorithm includes the PSO process that searches for optimal parameters such as kernel sizes, output feature maps, learning rate, number of epochs, and batch size. It introduced the notion of 'chaos' as a population initialization technique and employed GA to evolve the PSO particles' update process. The hybrid CHA-CNN with GA/PSO algorithm is compared to the manual parameter adjustment CNN, K-means Radial Base Function (RBF), and Gradient Boot Regression Tree (GBRT). CHA-CNN outperformed the other techniques, with the annual averaged MAE of the proposed method reduced by 49.47%, 47.6%, and 20.34%, respectively, compared with manual CNN, K-means-RBF, and GBRT.

Ghimire et al. [152] proposed a Self-adaptive Differential Evolutionary ELM (SaDE-ELM) hybridized with swarm-based Ant Colony Optimization (ACO) for the prediction of Global Solar Radiation (GSR). The predictor dataset consists of 67 atmospheric, land, and oceanic climate variables. The SaDE-ELM is benchmarked against nine different models, including three variants of ELM, Genetic Programming (GP), PSO-NN, GA-NN, PSO-Support Vector Regression (PSO-SVR), GA-SVR, and Grid Search-SVR (GS-SVR). Test results showed that the SaDE-ELM model in terms of Coefficient of Determination (R^2), RMSE,

and MAE performance indices were better than those nine models at 0.99, 0.405, and 0.506, respectively. The success of SaDE-ELM can be attributed to the ACO, which proved to be an effective feature selection algorithm. It is then combined with DE to select the network parameters for ELM to enable SaDE-ELM to overcome the slow weight updating process of classical ELM.

Guijo-Rubio et al. [153] proposed an Evolutionary ANN (EANN) using satellite data as features for solar radiation prediction without the need for ground measurements or data based on atmospheric variables. The predictive variables used in the experiments were reflectivity, clear sky radiance, cloud index, solar radiation data from Copernicus Atmosphere Monitoring Service (CAMS) [154], and solar data from SolarGIS [155]. The target was global solar radiation. The author examined different activation functions, including the sigmoid function, the radial basis function, and the product function, and compared the performance of the evolutionary ANN with other ML regressors. This approach achieved a 2% improvement compared to the results obtained by an ELM and over 6% by numerical models based on satellite measurements. Meng et al. [156] utilized GA to optimize the weights and bias values of the BPNN with an artificial classification of history day to predict solar PV power. The input predictors were meteorological data, including ground irradiance, temperature, total cloud amount, total cloud amount, wind speed, and humidity, from the European Centre for Medium-Range Weather Forecasts (ECMWF). As variations in weather conditions are proportional to the PV power generation, the weather types were pre-classified into four different templates: sunny, cloudy, showers, and heavy rain. The proposed method trained the model and then searched historical sample data closely correlated to the forecast weather types, improving the PV power prediction accuracy. The normalized RMSE (nRMSE) and normalized MAE (nMAE) values obtained using this GA-BPNN with historical day classification improved by 3.45% and 11.6%, respectively, over GA-BPNN. Jaidee and Pora [157] proposed using GA to optimize the parameters of ANNs for very short-term solar power forecasting. The network types examined include GA-DNN, GA-LSTM, GA-Gated Recurrent Unit (GA-GRU), and GA-Cuda Deep Neural Network Gated Recurrent Unit (GA-CuDNNGRU). Each network has 23 inputs, eight outputs, and three hidden layers. The eight outputs provided eight time-

steps with 30 mins each to produce a 4-hour ahead forecast. Test results showed that the GA-GRU and the GA-CuDNNGRU performed well, with RMSE values at 7.83% and 7.87%, respectively.

A summary of prior work reviewed in this section is shown in Table 2.3.

Table 2.3 A summary of the prior work in solar irradiance forecasting.

Author	Classification	Model	Temporal and Spatial Resolution	Remarks
Yang [116]	Numerical Weather Prediction	Applied the Global Forecast System (GFS) model and uses	3 to 6-h resolution for a 180 h horizon over a 70 x 70 km^2 spatial grid.	The NWP model often resorts to different strategies for developing and evaluating new parameterization of physical models.
Larson [117]	Numerical Weather Prediction	Applied the North American Mesoscale (NAM) model.	1-h resolution for a 1 to 36-h horizon and a 3-h resolution for a 39 to 84-h horizon, over a 12 x 12 km^2 spatial grid.	Tend to overpredict the solar irradiance due to biases in the model predicting insufficient cloud cover.
Jimenez [118]	Numerical Weather Prediction	Applied the Weather Research and Forecasting (WRF) model.	3-h resolution interpolated to hourly for up to 24-h horizon over a 13 x 13 km^2 spatial grid.	The inclusion of aerosol characterization emphasized the climatological impact, but it requires precise instrumentation to measure aerosol properties accurately.
Tarpley [123]	Satellite Imaging	Applied image processing to estimate solar flux depletion and determine cloud motion vector fields.	30 mins or 1-h resolution with a forecast horizon of up to 24-h over a 2.5 x 2.5 km^2 to 50 x 50 km^2 spatial grid.	Ground pyranometers measurements are needed to tune parameters through statistical methods.
Hammer [115]	Satellite Imaging	Applied image processing to analyze information on cloudiness and calculated cloud motion vectors	30 mins resolution at a horizon of ten days, with a spatial resolution of 2.5 x 2.5 km^2.	The formula relies on cloudiness as the sole atmospheric parameter for surface irradiance, ignoring the presence of atmospheric aerosol and particles as these are not considered.
Rigollier [124]	Satellite Imaging	Presented the Heliosat-2 model integrating satellite data with an improved Heliosat-1 model	1-h resolution at 5 and ten days horizon with a spatial resolution of 10 x 10 km^2.	It required accurate and frequent calibration of the radiometers.
Marquez [127]	Total Sky Imager	Applied image processing to classify clear, thin, and opaque clouds.	1 min resolution with 3 to 15 min horizon.	Requires the sky imager to be co-located with the solar PV plant with a larger plant requiring multiple imagers.
Chow [128]	Total Sky Imager	Applied cloud detection technique and cloud motion vectors.	30-sec resolution with a 5 min horizon over a 2 x 2 km^2 spatial resolution.	Circumsolar glare on the optics from the sky imager could affect image processing accuracy.
Yang [130]	Total Sky Imager	Applied image processing and cloud field velocity to predict cloud motion.	30-sec resolution for up to 15 min horizons over a limited spatial resolution.	Required deployment of multiple pyranometers to increase the forecast horizon. Higher errors within the circumsolar region due to scattering of sunlight caused by aerosols.
Bacher [132]	Statistical Analysis	Presented statistical smoothing and applied AR and ARX models.	15 mins resolution with a horizon starting from 1-h to 36-h.	The ARX model that used solar power observations and NWPs as input produced better results with a 35% improvement of RMSE value over the reference model.
Li [133]	Statistical Analysis	Applied time-series ARMA model and ARMAX model	Forecast 1-day ahead power output of the PV system using 6 months of the training dataset.	The ARMAX model outperformed other statistical methods and Radial Base Function (RBF) network.

Author	Classification	Model	Temporal and Spatial Resolution	Remarks
Chu [134]	Statistical Analysis	Proposed a Smart ARMA model using a GA-trained ANN to optimize the input variables	30-sec resolution with a forecast horizon of 5, 10, and 15 mins over a spatial resolution of 4 x 4 km^2.	Although the forecast bias was correct and higher accuracy was achieved at higher solar variability, forecast errors were not consistently reduced at moderate to low solar variability.
Gagne [135]	Statistical Analysis	Utilized NWP and MOS	5 min resolution of hourly forecast up to 14 to the 24-hour horizon.	Parameter configuration is an issue as it dramatically impacts model performance, leading to significant investment in model tuning.
Diagne [136]	Statistical Analysis	Utilized WRF and MOS	Hourly resolution with up to 24 h forecast horizon over a spatial resolution of 3 km.	Required large and high-quality ground measurements to evaluate the forecast accuracy and refine the model.
Noia [137]	Statistical Analysis	Performed solar radiation forecasting using the MOS approach	Hourly resolution at a 1-day forecast horizon with a spatial resolution of 1.4 km x 1.4 km.	Required high-quality ground-based data for verification and tuning may cause forecasting errors due to the localization of the pyranometer sites on the satellite images.
Crisosto [141]	Machine Learning and AI – Shallow Network	Applied LM-BPNN model using irradiance and sky images are inputs.	1 min resolution at 1-h forecast horizon.	Achieved a 40% reduction in RMSE and MAE values compared to the reference persistence model. Predicted first 10 to 30 mins well, but accuracy is affected beyond this horizon.
Faceira [142]	Machine Learning and AI – Shallow Network	Applied LM Multi-Layer Perceptron (MLP)	Hourly resolution at 1-day forecast horizon.	Achieved a low MAPE value of 5.1%. Utilized MLP with 3 hidden layers with no significant performance improvement in ANN with deeper layers.
Nurcahyo [143]	Machine Learning and AI – Shallow Networks	Applied hybrid GA and Partially-Connection Neural Networks (PCNN)	Weather prediction at 7-day forecast horizon	Achieved higher accuracy and lower MAPE values than GA-FCNN.
Su [144]	Machine Learning and AI – Shallow Network	Proposed hybrid ANN using an ensemble approach to average the forecast skills	Half-hour resolution at a 6-day horizon.	Achieved lowest nRMSE value at 6.74% averaged over four seasons. None of the ten uncorrelated methods was superior in all four seasons, but combining them exhibited better results.
Mulyadi [59]	Machine Learning and AI – Deep Network	Applied Convolutional Neural Networks (CNN) model	Daily resolution at 1-month forecast horizon	Achieved a high accuracy of 98.84% for the training dataset but moderate accuracy of 54.55% for the new dataset suggesting strong evidence of overfitting.
Capizzi [60]	Machine Learning and AI – Deep Network	Proposed Recurrent Neural Network (RNN) model with wavelet transform	1-day resolution with a 2-day forecast horizon	Produced a very low RMSE error compared to models obtained by other hybrid ANNs. However, RNN is constrained by the vanishing gradient problem.
Abayomi-Alli [147]	Machine Learning and AI – Deep Network	Applied Long Short-Term Memory (LSTM) model	1-day resolution with 35 years of dataset split into 70% training and 30% testing data.	The model achieved an accuracy of 99.3% to 99.9%. Nevertheless, LSTM training requires tuning many hyperparameters, such as epochs, layers, neurons, and optimizers.
Muhammad [148]	Machine Learning and AI – Deep Network	Applied LSTM model	Hourly and daily resolutions for a 1-year forecast horizon	The model achieved an MSE value of 10.4% in the hourly prediction for a year ahead. Requires time-consuming manual tuning of hyperparameters.
Ghimire [149]	Machine Learning and AI – Deep Network	Proposed a hybrid CNN with LSTM (CLSTM) model	30 mins resolution for several forecast horizons of 1-day, 1-week, 2-week, 1-month up to 8-month.	The model achieved RMSE values of 8.2%, 16.0%, 14.2%, and 32.8%, for the forecast horizon, of 1-d, 1-w, 2-w and 1-m, respectively.
Wang [150]	Machine Learning and AI – Deep Network	Proposed a hybrid CLSTM model	5 mins resolution for a 1-day forecast horizon using 3 years of data were split into 90% for	High computational cost, which increases significantly with the filter size. Additionally, the training data

Author	Classification	Model	Temporal and Spatial Resolution	Remarks
			training and 10% for testing.	size must increase to be large enough to cover all features.
Dong [151]	Machine Learning and AI – Evolutionary Deep Network	Applied CNN with a chaotic GA/PSO hybrid algorithm (CHA-CNN) to optimize the CNN networks' hyperparameters	3-hour resolution and prediction using nine years of the training dataset, one year of the testing dataset	CHA-CNN equipped with a chaotic hybrid GA/PSO algorithm using 15 meteorological attributes achieved an MAE reduction of 49.5%, 47.6%, and 20.3%, respectively, compared with manually configured CNN, K-means-RBF, and GBRT.
Ghimire [152]	Machine Learning and AI - Evolutionary Shallow Network	Proposed Self-adaptive differential evolutionary Extreme Learning Machine (SaDE-ELM) hybridized with ant colony optimization (ACO)	6-hour resolution to predict monthly daily average. 176 months of the dataset were used, where 130 months were for training and 46 months were for testing.	ACO was used for feature selection, and DE was used for hyperparameter optimization to develop the 3-layer ELM model. The R^2, RMSE, and MAE values performed at 0.99, 0.405, and 0.506, respectively, outperforming nine evolutionary ANNs.
Guijo-Rubio [153]	Machine Learning and AI - Evolutionary ANN	Presented Evolutionary ANN (EANN) trained using satellite data	1-h resolution for 1-h forecast horizon with a spatial resolution of 1 km^2 to 3 km^2.	The model achieved a 2% improvement compared to the results obtained by an ELM and over 6% by numerical models based on satellite measurements.
Meng [156]	Machine Learning and AI - Evolutionary ANN	Applied GA-BPNN with weather type classification	15 mins resolution at a 1-day forecast horizon. 6- month dataset was used with 5-m for training and 1-m for testing.	Values of nRMSE and nMAE obtained using GA-BPNN with weather classification improved by 3.45% and 11.6% over GA-BPNN.
Jaidee [157]	Machine Learning and AI	Applied Genetic Algorithm Neural Network (GA-NN)	30-mins resolution at 4-h forecast horizon.	The models GA-GRU and the GA-CuDNNGRU performed well with RMSE values at 7.83% and 7.87%, respectively.

2.6 Summary of Solar Irradiance Forecasting Studies

Solar irradiance and PV power forecasting are well discussed in the literature. The physical method relies on the law of physics to establish mathematical models to predict solar irradiance and calculate solar PV power generation. The prediction accuracy of the physical method is strongly dependent on the accuracy of the NWP information, but currently, it exhibits limitations in improving the accuracy of NWP.

The statistical analysis method establishes the correlation mapping relationship between input-output data employing regression equations, parameter estimation, and correlation analysis of the processed historical data such as solar radiation and meteorological-related data to predict irradiance and PV power generation. Unlike the physical method, the

statistical analysis does not require a clear and complete understanding of the physical laws and their relationship with the PV system, but only a partial understanding and realization through various regression techniques.

Hence, compared with the physical method, the statistical analysis method has the advantages of a more straightforward modeling approach. However, the prediction accuracy of the statistical method is strongly related to the quality of the historical data, which may be challenging to acquire in implementation. The prediction accuracy generally depends on the ability to process a higher-dimensional data set to ensure the effect, increasing the complexity and slowing down the prediction speed.

Machine learning can efficiently extract high-dimensional complex nonlinear features and map them directly to the output. Due to this advantage, the machine learning-based prediction method has become one of the most commonly used methods in recent studies for solar irradiance forecasting. The literature review shows that the ANN-based prediction model significantly improved RMSE, MAE, and MAPE values compared to some typical physical or statistical prediction models. In addition, hybrid evolutionary-based ANN models have reported promising results in the literature.

Many of the predictive models relied on a large number of satellite images and meteorological and atmospheric variables. These variables include solar radiation at the earth's surface and top of the atmosphere, latent and sensible heat, wind stresses, surface rainfall and snowfall, cloud fraction, cloud condensate, atmospheric temperature, and humidity may be expensive to obtain. Moreover, the data from these variables requires to be accurate and of vast quantity for these models to be effective. To address this challenge, a novel EVLNN model is proposed that tries the reverse direction of what most forecast models do by selecting fewer and essential features while working with datasets with smaller sample sizes to achieve a model with higher accuracy.

Chapter 3

3. Evolutionary Lean Neural Network

3.1 Introduction

This chapter details a novel machine learning (ML) approach, EVLNN, that combines the strength of several novel mechanisms of an improved GA to optimize a set of ANNs based on parsimony. This approach allows the modeling of nonlinear functions capturing the relationships between the inputs and the outputs while not having complications of extra parameters in the representation. The result is a generalized ANN model with a lean architecture offering higher accuracy, particularly when the number of features and samples is small [158]. Conventional ANN has a large amount of redundancy [159]. The unnecessary connections increase network complexity, leading to poor generalization and difficulty understanding the input-out relationships [160]. The EVLNN design based on parsimony contains only a subset of the entire set of possible connections of the ANN. By considering parsimony, network complexity can be reduced to help improve generalization and offer a better interpretation of the prediction outcome [161].

3.2 The EVLNN Framework

The EVLNN framework comprises two main components – the encoding scheme and the neural architecture search algorithm. EVLNN adopts a direct encoding method for genotype-phenotype mapping of the ANN architecture, where the mapping is achieved using a structurally inclusive chromosome matrix. Phenotypic information such as the number of neurons, the connection weights, and the type of connections (feedforward or feedback) are directly mapped onto a chromosome matrix. An improved GA with species parallelism introduced into the search algorithm categorically distinguishes these solution candidates into their respective genotypically similar species. Two crossover strategies,

namely intra-species and inter-species crossovers, are presented to allow species to exchange genetic information in search of global solutions. Intra-species crossover maintains parallelism by a careful recombination of similar species, while inter-species crossover seeks to discover new landscapes in the solution space. The differentiated search strategies aim to achieve both intensification and diversification of search. A two-stage mutation is proposed to avoid locally optimal solutions through incremental changes to the weights, bias, and network connections. The Mean Square Error (MSE) is employed as the objective function to guide the evolutionary process toward global optimality. An ensemble-based approach to sensitivity analysis is designed for insightful interpretations of the relationships between the input variables and the output responses. During the evolutionary process, species diversity and richness are tracked through three diversity measures. The information is used for studying EVLNN's search behavior and parameter tuning. Figure 3.1 shows the framework of EVLNN, and the following sections describe the framework in detail.

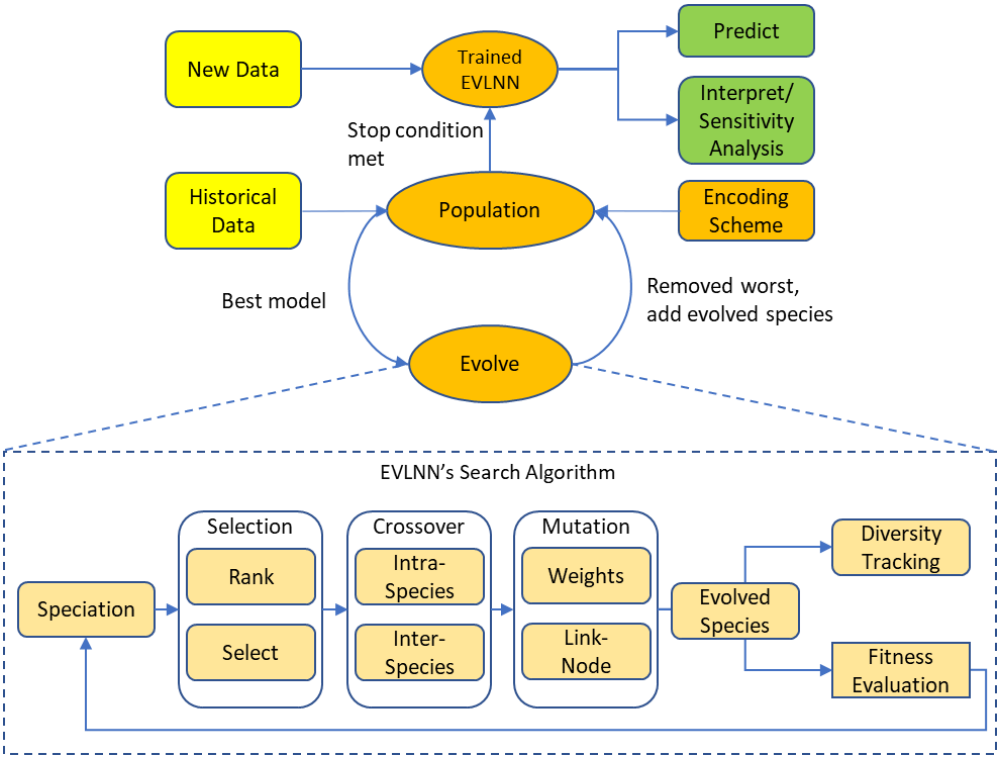


Figure 3.1 The EVLNN framework.

3.2.1 Encoding Scheme

Encoding is the process of formulating the possible solutions to a given problem to a search space representation in the form of a chromosome [162]. The chromosome should carry the necessary information representing the solution's characteristics. The encoding scheme's design thus depends on the problem' nature and can impact the efficiency of a search algorithm [163]. Structural optimization is a complex problem as many design parameters must be considered [164].

The proposed encoding scheme in EVLNN uses a structurally inclusive chromosome matrix to convey information like the feedforward and feedback connectivity types, number of hidden layers, number of neurons, connection weights and connectivities. While the scheme is well-suited for a three-layer feedforward or feedback ANN, its basic encoding structure can be scaled to accommodate deeper layers. EVLNN's search algorithm is applied to optimize the learnable parameters embodied in the chromosomes by simulating genetic evolution to propagate parent genetic properties in more highly fit chromosomes to the successor generations [165].

3.2.2 Matrix-Based Chromosome Encoding

A three-layer ANN structure can be represented by an m x n chromosome matrix as expressed in Equation 3.1,

$$x_{i,j} \qquad \forall_j = \{1, \dots, n\}\ \forall_i = \{1, \dots, m\} \tag{3.1}$$

where $x_{i,j}$ represents the *ith* row and *jth* column of the matrix for the connection between nodes i and j, and its value is the weight of that connection. To describe the encoding scheme, any given feedforward three-layer ANN structure with I input nodes, H hidden nodes, and O output nodes can be represented by an m x n chromosome matrix where m and n are expressed in Equation 3.2 and 3.3, respectively,

$$m = I + B_I + B_H + O \tag{3.2}$$

$$n = H \tag{3.3}$$

where B_I and B_H are single-bias nodes at the input layer and hidden layer, respectively.

For example, Figure 3.2 illustrates the encoding of a given 3-layer feedforward ANN with two input nodes (*I=2)*, five hidden nodes (*H=5)*, and two output nodes (*O*=2). The dimension of the chromosome matrix representation for the ANN is determined by applying Equations 3.2 and 3.3, where,

$$\mathrm{M} = I + B_I + B_H + O = 2 + 1 + 1 + 2 = 6 \tag{3.4}$$

and

$$n = H = 5 \tag{3.5}$$

The sample chromosome matrix is shown in Figure 3.3, where it is formed through a collection of several vectors, namely the input vector (Figure 3.4a), the input bias vector (Figure 3.4b), the hidden bias vector (Figure 3.4c), and the output vector (Figure 3.4d).

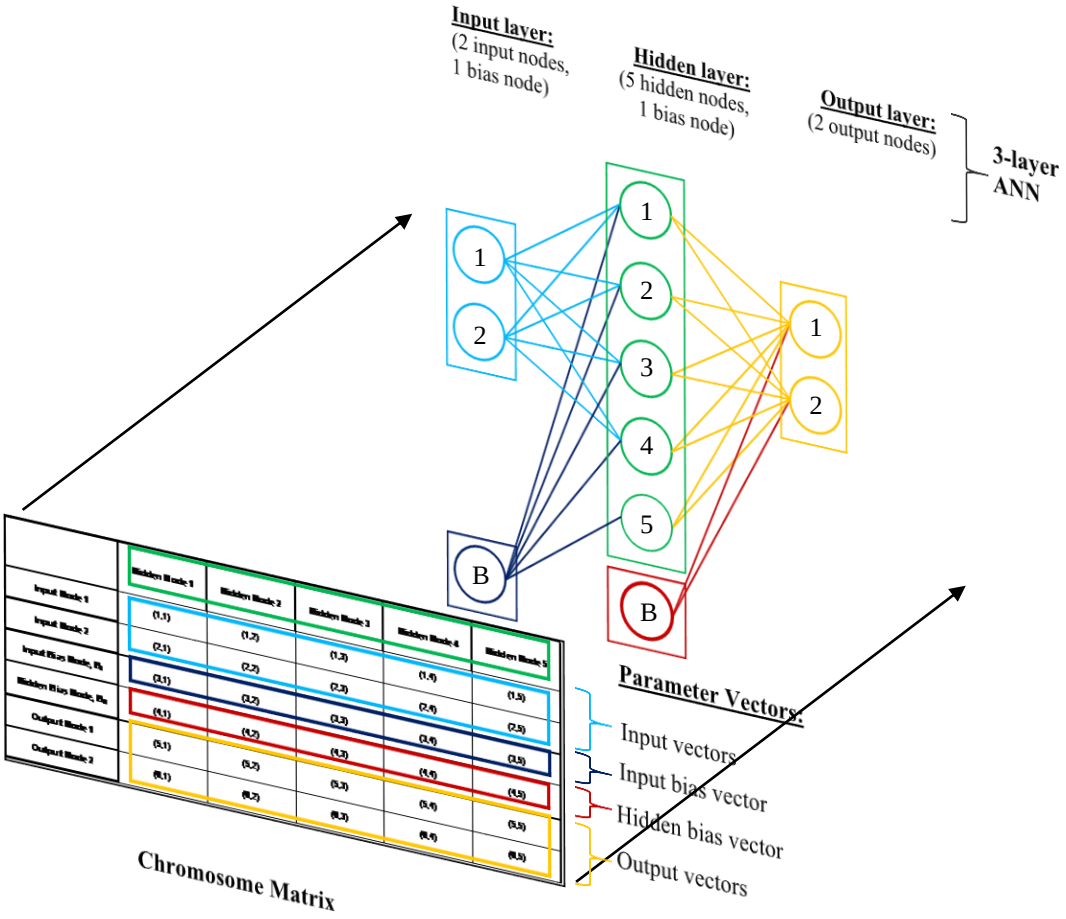


Figure 3.2 ANN encoding using a chromosome matrix.

	Hidden Node 1	Hidden Node 2	Hidden Node 3	Hidden Node 4	Hidden Node 5
Input Node 1	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
Input Node 2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
Input Bias Node, B_I	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)
Hidden Bias Node, B_H	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)
Output Node 1	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)
Output Node 2	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)

Chromosome Matrix

Figure 3.3 A sample chromosome matrix.

The input vector's element $x_{i,j}$ corresponds to a connection weight, $w(i,j)$ between the input node *i* and the hidden node *j*. Similar element mapping applies to the input bias vector and the output vector. For the hidden bias vector, each element corresponds to a connection weight between the hidden layer bias node *i* and the output layer node *j*. The remaining elements in the hidden bias vector are not used in the calculation.

	Hidden Node 1	Hidden Node 2	Hidden Node 3	Hidden Node 4	Hidden Node 5
Input Node 1	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
Input Node 2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)

(a) Input vectors.

	Hidden Node 1	Hidden Node 2	Hidden Node 3	Hidden Node 4	Hidden Node 5
Input Bias Node B_I	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)

(b) Input bias vector.

	Hidden Node 1	Hidden Node 2	Hidden Node 3	Hidden Node 4	Hidden Node 5
Hidden Bias Node B_H	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)

(c) Hidden bias vector.

	Hidden Node 1	Hidden Node 2	Hidden Node 3	Hidden Node 4	Hidden Node 5
Output Node 1	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)
Output Node 2	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)

(d) Output vectors.

Figure 3.4(a-d) A sample breakdown of a chromosome matrix.

3.2.3 Model Representation

With the formation of the chromosome matrix, a further step is needed to transform the chromosome matrix into its genotype form to present a traceable genotype-phenotype mapping. In EVLNN, genotypes are building blocks that represent feasible solutions to the problem in the fitness landscape. Model representation allows fitness information of the candidate solutions to be calculated and a set of genetic operators to influence the possible movement of the feasible solutions towards global optima in a minimum number of steps [165]. In this case, the 'problem' is optimizing the EVLNN model. Good genetic properties in the building blocks are propagated from parent to child as the phenotypes undergo spatial and weight changes in the fitness landscape [166]. At convergence, the fittest individual would have the optimized model found [167].

The explanation of the genotype-phenotype mapping in EVLNN is illustrated in Figures 3.5 and 3.6. Figure 3.5 shows a sample chromosome matrix with connection weight values. The matrix is transformed to its genotype by removing the columns if the output vector column contains zero values. Elements with zero values in the genotype shown in Figure 3.6a represent inactive connections in the phenotype (Figure 3.6b).

	Hidden Node 1	**Hidden Node 2**	**Hidden Node 3**	**Hidden Node 4**	**Hidden Node 5**
Input Node 1	0.832	0	0.526	0.975	0.080
Input Node 2	0	0	-0.818	0.604	0
Input Bias Node 1	0.021	-0.705	0.082	0.679	-0.070
Hidden Bias Node 1	-0.610	0.774	-0.193	0.700	-0.424
Output Node 1	0.243	0	0	-0.703	0
Output Node 2	0.174	-0.456	-0.085	-0.223	0

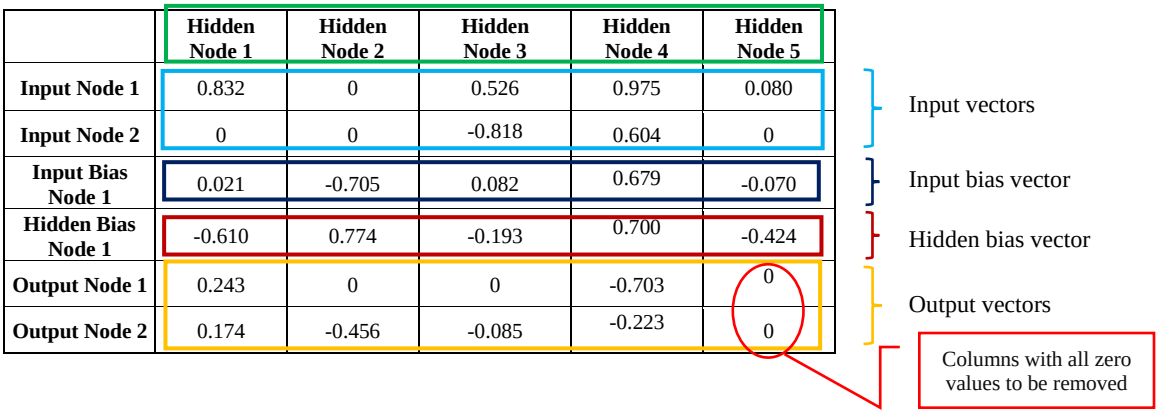
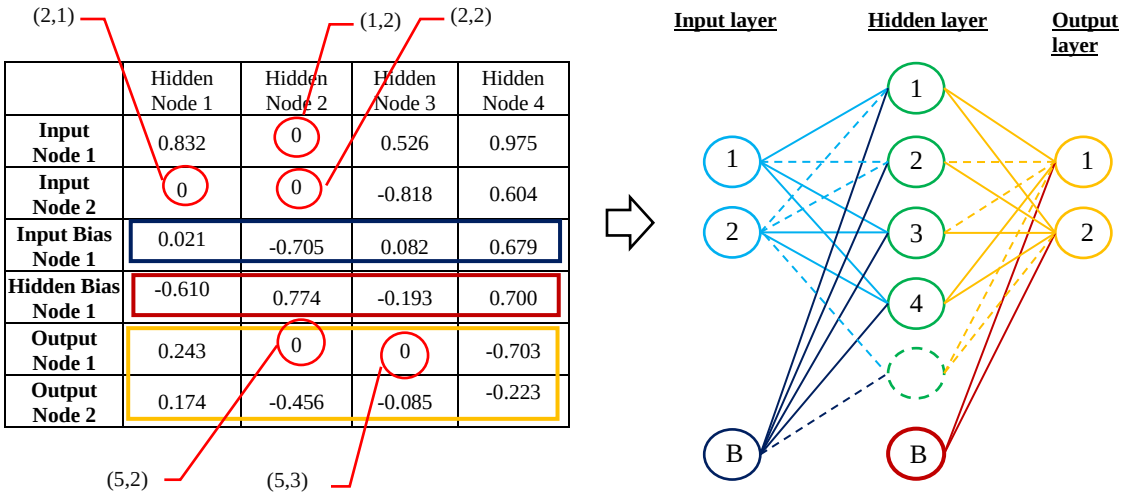


Figure 3.5 A sample *6* x *5* chromosome matrix for a 3-layer EVLNN with two input variables and one output variable with a potential of up to 5 hidden neurons.

For example, elements at indices (1, 2), (2,1), and (2,2) of the input vectors have zero values, and so are elements at indices (5,2) and (5,3) in the output vector. An element with zero values represents an inactive connection.



	Hidden Node 1	Hidden Node 2	Hidden Node 3	Hidden Node 4
Input Node 1	0.832	0	0.526	0.975
Input Node 2	0	0	-0.818	0.604
Input Bias Node 1	0.021	-0.705	0.082	0.679
Hidden Bias Node 1	-0.610	0.774	-0.193	0.700
Output Node 1	0.243	0	0	-0.703
Output Node 2	0.174	-0.456	-0.085	-0.223

(a) Genotype (b) Phenotype

Figure 3.6(a-b) Genotype-to-phenotype mapping for EVLNN. A zero value in the genotype corresponds to an inactive connection on the phenotype.

3.2.4 EVLNN Architecture

Figure 3.7 illustrates the relationship between the nodes, weights, connections, and activation functions and the information flow carried through the EVLNN artificial neurons using a set of equations presented in Equations 3.6 to 3.14. The hidden nodes $h_1, h_2, \ldots, h_j$ are related to the input nodes $x_1, x_2, \ldots, x_i$ via the respective weighted connections, $w_{11}, w_{12}, \ldots, w_{ji}$, where w_{ji} represents the weighted connection from the input node x_i to the hidden node h_j.

Likewise, the hidden nodes $h_1, h_2, \ldots, h_j$ are related to the output nodes $o_1, o_2, \ldots, o_k$, via the respective weighted connections, $w_{11}, w_{21}, \ldots, w_{jk}$, where w_{jk} represents the weighted connection from the hidden node h_j to the output node o_k. At the input side of the hidden node $h_{j,\,in}$, weighted inputs are summed and combined with the input bias b_j into a single value (Equation 3.6), which then undergoes an activation function, $\sigma_j(\cdot)$ in the hidden node (Equation 3.7), to obtain c. Similarly, at the input side of the output node, $h_{j,out}$ signals are weighted and summed, then combined with the output bias b_k into a single value to obtain $o_{k,in}$ (Equation 3.8), which then undergoes an activation function, $\sigma_k(\cdot)$ in the output node to obtain $o_{k,out}$ (Equation 3.9), the predicted value of the network.

$$h_{j,in} = \sum_i w_{ji} * x_i + b_j \tag{3.6}$$

$$h_{j,out} = \sigma_j(h_{j,in}) \tag{3.7}$$

$$o_{k,in} = \sum_j w_{jk} * h_{j,out} + b_k \tag{3.8}$$

$$o_{k,out} = \sigma_k(o_{k,in}) \tag{3.9}$$

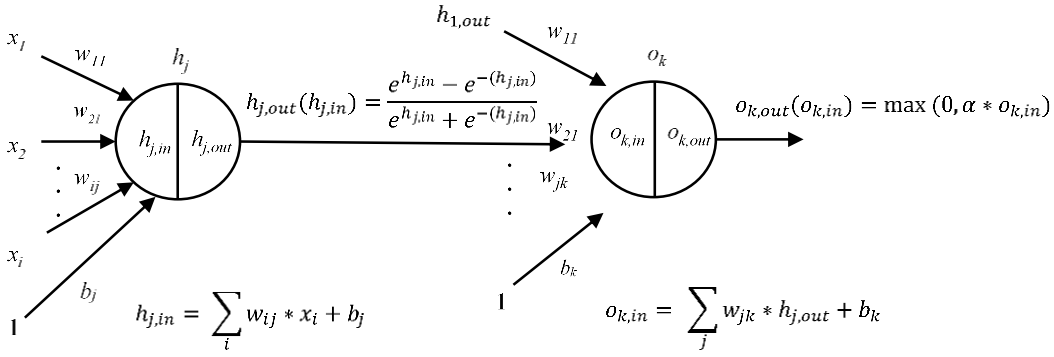


Figure 3.7 Relationships between nodes, weights, connections, and activation functions for a sample EVLNN showing Tanh and ReLU activation functions for the hidden and output nodes.

As activation functions have their strengths and weaknesses [168], EVLNN allows the selection of various activation functions for the hidden and output nodes depending on the problem and the training dataset. These activation functions include the Hyperbolic Tangent (Tanh) function $\sigma_{j,tanh}(\cdot)$, the Sigmoid function $\sigma_{k,sigmoid}(\cdot)$, the Rectified Linear Unit (ReLU) function $\sigma_{k,relu}(\cdot)$, the Leaky ReLU function $\sigma_{k,lrelu}(\cdot)$ and the linear activation function $\sigma_{k,linear}(\cdot)$. The function equations are expressed in Equations 3.10 to 3.14, respectively.

$$h_{j,out} = \sigma_{j,tanh}(h_{j,in}) = \frac{e^{(h_{j,in})} - e^{-(h_{j,in})}}{e^{(h_{j,in})} + e^{-(h_{j,in})}} \tag{3.10}$$

$$h_{j,out} = \sigma_{j,sigmoid}(h_{j,in}) = \frac{1}{1 + e^{-(h_{j,in})}} \tag{3.11}$$

$$o_{k,out} = \sigma_{k,relu}(o_{k,in}) = \max(0,\ o_{k,in}) = \begin{cases} 0 & if\ o_{k,in} < 0 \\ o_{k,in} & if\ o_{k,in} \geq 0 \end{cases} \tag{3.12}$$

$$o_{k,out} = \sigma_{k,lrelu}(o_{k,in}) = a(o_{k,in}) + (o_{k,in}) = \begin{cases} a(o_{k,in}) \; if \; o_{k,in} & < 0 \\ o_{k,in} \quad if \; o_{k,in} & \geq 0 \end{cases} \tag{3.13}$$

$$o_{k,out} = \sigma_k(\cdot) \tag{3.14}$$

3.3 The EVLNN Search Algorithm

Various concepts of speciation in nature are being investigated in the literature [169]. The essence of speciation is contingent on the isolation of gene flow between subgroups [170]. There are two dominant views of speciation. The first is allopatric speciation. In allopatric speciation, the original population is split into isolated subpopulations caused by geographical barriers. This split prevented gene exchange between the isolated subpopulations [171]. The subpopulations, therefore, can evolve independently of the other, forming evolutionary independent new species. The second is sympatric speciation, where speciation occurs due to changes in population genetics under dissimilar selective pressure, specifically mating preference [172]. The evolution of mating preference continues after the post-mating isolation instigated further gene changes among the subgroups and subsequent differentiation in phenotype [173]. The divergence in phenotype increases the reproductive barrier between subgroups and gene flow, leading to evolutionary independent sympatric species [174].

Inspired by the concept of ecological speciation in which species adapt and breed to locate resource basins or niches in a fitness landscape, the proposed framework is an improved GA for a multimodal search to locate the global optima. The EVLNN search algorithm, based on the framework in Figure 3.1, is described in the following. The algorithm source code is developed using the MATLAB R2020a software, and the pseudo-code can be found in Appendix A. A step-by-step outline of the EVLNN search algorithm is presented below, followed by a detailed description.

1. **Population Initiation**: A population of candidate solutions with m x n chromosome matrices is initiated with uniformly distributed random values between -1 to 1 representing the connection weights.
2. **Speciation**: The new chromosome matrices created are transformed into their respective genotypes of varying dimensions. Genotypes with the same dimension are subsequently speciated into their respective sub-populations.
3. **Ranking and Selection**: The individuals are evaluated by a fitness function and subsequently ranked within their respective species. Selection pressure using the Stochastic Universal Sampling method is applied where individuals with higher fitness have more chance to be selected for crossover.
4. **Crossover:** Two strategies are adopted to maintain species parallelism in the search process while exploring new landscapes in the solution space. These are the intra-species crossover and the inter-species crossover.
    a. **Intra-species crossover**: The intra-species crossover restricts mating within individual species to maintain species parallelism and intensifies the search by propagating good genotypic properties to the next generation. This strategy allows species to parallel explore the search space around promising basins of interest.
    b. **Inter-species crossover**: The inter-species crossover is interspersed in the evolutionary process to diversify the search. Mating pairs are randomly selected from diverse species for crossover. If successful, the pair will produce offspring with novel structures.
5. **Mutation:** After the selected individuals have undergone the crossover, a two-stage mutation is implemented to avoid local optima through incremental changes to its architecture. The stages are the weights mutation and the link-node mutation.
    a. **Weights mutation**: The weights mutation attempts to perturb selected connection weights with uniformly distributed random values between -0.5 and 0.5.
    b. **Link-node mutation**: The link-node mutation aims to change the neural network structure incrementally by re-enabling disabled links on the network.

6. **Fitness Evaluation and Termination Criteria**: The fitness of the new individuals (offspring) is evaluated, and healthier individuals continue onto the next generation propagating good genotypic properties. The algorithm terminates if the healthiest individual is found. Otherwise, the cycle repeats steps 2 to 6 until the termination criteria are met.
7. **Diversity Tracking**: A diversity tracking mechanism is introduced to measure species richness, species evenness, and population diversity. These diversity measures are built into the EVLNN algorithm to provide much-needed insights into the algorithm's search behavior.
8. **Ensemble-based Sensitivity Analysis**: An ensemble-based approach to sensitivity analysis is incorporated into the algorithm to analyze the relationship between the input variables and the non-linear transformations the network learned at its output. By combining several uncorrelated sensitivity analysis methods, the sub-categorical contribution of the input variables to the output is ranked according to their relative importance to identify factors influencing the output.

3.3.1 Population Initialization and Speciation

At initialization, a population *P* of size *p* is generated as presented in Equation 3.15,

$$P = \{NN_1, NN_2, NN_i, \dots, NN_p\} \tag{3.15}$$

where NN_i is the *ith* individual in population *P*. The individual NN_i is an m x n chromosome matrix formed using Equation 3.1, with the matrix dimension determined using Equations 3.2 and 3.3. For example, Equation 3.16 shows a chromosome matrix where $x_{i,j}$ stores a uniformly distributed random value between -1 and 1. These values represent the connection weights in the respective phenotype.

$$NN_i = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix} \tag{3.16}$$

where,

$$x_{i,j} = [-1\ 1] \tag{3.17}$$

A sample chromosome matrix of an individual NN_i is shown partially in Equation 3.18.

$$NN_1 = \begin{bmatrix} -0.8049 & 0.4121 & \cdots & 0.5381 \\ 0.4430 & -0.9363 & \cdots & 0.1629 \\ \vdots & \vdots & \ddots & \vdots \\ -0.6576 & -0.7620 & \cdots & -0.4363 \end{bmatrix} \tag{3.18}$$

A set of m x n uniformly distributed random binary number matrices, $b_{i,j}$ are then generated and used to perform element-wise matrix multiplication with $x_{i,j}$ to create the basic structure of parsimonious networks using Equation 3.19,

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix} \odot \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,1} & b_{m,2} & \cdots & b_{m,n} \end{bmatrix} = \begin{bmatrix} x_{1,1} \cdot b_{1,1} & x_{1,2} \cdot b_{1,2} & \cdots & x_{1,n} \cdot b_{1,n} \\ x_{2,1} \cdot b_{2,1} & x_{2,2} \cdot b_{2,2} & \cdots & x_{2,n} \cdot b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} \cdot b_{m,1} & x_{m,2} \cdot b_{m,2} & \cdots & x_{m,n} \cdot b_{m,n} \end{bmatrix} \tag{3.19}$$

A sample of the binary number matric, $b_{1,1}$ is shown partially in Equation 3.20,

$$b_{1,1} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 1 \end{bmatrix} \tag{3.20}$$

A sample of the resultant chromosome matrix after applying Equation 3.19 to Equations 3.18 and 3.20 is shown partially in Equation 3.21,

$$\begin{bmatrix} -0.8049 & 0.4121 & \cdots & 0.5381 \\ 0.4430 & -0.9363 & \cdots & 0.1629 \\ \vdots & \vdots & \ddots & \vdots \\ -0.6576 & -0.7620 & \cdots & -0.4363 \end{bmatrix} \odot \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.4121 & \cdots & 0.5381 \\ 0.4430 & -0.9363 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -0.6576 & 0 & \cdots & 0.4363 \end{bmatrix} \tag{3.21}$$

Two impacts resulted from this multiplication. Firstly, a chromosome matric that contains uniformly distributed zero values is generated. As elements in the chromosome matrix represent connection weights in the individual's phenotype, a link is eliminated or disabled when the value is zero, making the network parsimonious. Secondly, the zero values in the chromosome matrix in the output vectors mean that individuals with different numbers of hidden nodes will be created. The EVLNN algorithm acted on this variation to separate the individuals into species. This speciation process is performed through a matrix operation

by removing the columns containing zero values in the output vector of the chromosome matrix. The result is a matrix, called a genotype, consisting of varying dimensions. The new population P' can be expressed as a collection of a subpopulation of unique species expressed in Equation 3.22,

$$P' = \{SP_1, SP_3, \dots, SP_i \dots, SP_H\} \tag{3.22}$$

where SP_i refers to species i given $1 \leqslant i \leqslant$ H. In the phenotype context, i also identify the number of hidden nodes. In the speciation process, each SP_i contains zero of more individuals expressed in Equation 3.23,

$$SP_i = \left\{NN_{sp_{i,1}}, NN_{sp_{i,2}}, \dots, NN_{sp_{i,j}} \mid null\right\} \tag{3.23}$$

where $NN_{sp_{i,j}}$ is the j^{th} individual in *species i* given $1 \leq j \leq 100$. The chromosome matrix NN_1 of Equation 3.18 is illustrated in Figure 3.8 with all its elements. Its corresponding speciated individual genotype matrix, $NN_{sp13,1}$, is shown in Figure 3.9.

0	0.4121	0	-0.2967	0	-0.1372	-0.0215	0	0	0	0.2481	0	0	-0.4160	0.3331	-0.4964	0.3998	0	0.5403	0	-0.7616	0	-0.9608	0.0288	0	0.5381
-0.4430	0	0.9195	0	0	0.8213	0	0	0.7519	0	0	0	0.6363	0	0	0	0.2771	0.8872	0	0	0.8797	0	0	0	0	0
0	-0.4462	0	0.1705	0	0	0.8001	0.4894	0	0	-0.2090	0	0.6351	0	0	0.2342	-0.9328	0.2754	0	0	0.2911	0.0508	-0.1514	0	0.9778	0.8566
0.9150	-0.9077	0.1705	0.0994	0.3784	0	-0.2615	0	0	0	0	0	0.4449	0.9681	0	-0.4694	-0.8624	0.9154	-0.0573	0	0	0.0607	0	0	0	0.1602
0	-0.8057	0	0.8344	0.4963	-0.7089	-0.7776	0	0	0	0.9760	0.8094	0	-0.6657	0	0.6488	-0.3608	-0.5186	-0.9285	-0.5821	0.2786	0.7223	-0.6059	-0.6003	0.7309	0
0	0.6469	0.5025	0	-0.0989	0	0.5605	0	-0.5845	0	0	0.2197	0.3192	-0.7876	0	0.9653	0	0.3522	0	0.4186	0.0894	-0.0303	0.6434	-0.1861	0.2251	0
0	0.3897	-0.4898	0.5144	0	0.7386	0	-0.2630	0	0	0	0.2353	0.0372	-0.2552	0.1224	0	0	-0.4219	0	-0.5275	0	0	-0.1402	0.4974	0.9799	0.7254
0.9143	-0.3658	0.0119	0	0	0.1594	-0.5166	0	0	0	0	0	0	0	0	-0.3122	0	0	-0.0530	0	0.0878	0	0	0.6512	0	-0.0314
-0.0292	0	0.3982	0	0.8267	0	-0.1922	0	-0.5390	-0.1517	0.5924	0	0.2980	0	0.3384	0.1681	0	0.3903	0	0	0.4421	0	-0.2176	0.5799	0	0
0	0	0	0	-0.6952	0	-0.8071	0	0	0.0157	-0.8026	0.1534	0	-0.3210	-0.6191	-0.7845	0	-0.8640	-0.3178	0	0.0450	0	0.5382	0	0	-0.5812
0	0	0	-0.8483	0.6516	0.7061	0	0	0	-0.8290	0	-0.6342	-0.0924	0	-0.2622	0	0.9373	0	0	-0.0825	0	-0.3046	0	0	0	0.1046
-0.1565	0	0	0	0	0	0	0.5514	-0.5482	0	-0.3293	-0.5201	0	0.8407	0	0	0.0627	-0.5519	0	0	-0.5626	-0.7000	0	0	-0.0038	0.2598
0.8315	0.5310	-0.7228	0	0.9923	-0.2981	0	0	-0.6586	0	0	0.7730	0	-0.8946	0.9633	0	0	0	0.4769	0.5406	0	0	0.5102	0	0.8017	0
0	0	-0.7014	0	0	0.0265	0.1504	-0.1283	0	0	-0.7269	0	0	0.4757	0	0	0	0.6888	-0.5143	-0.2996	-0.7806	-0.4757	-0.2452	-0.7274	0.1493	0.2294
0	0	0	0	0	-0.1964	-0.8804	-0.1064	0	0.8577	0.4425	-0.0202	0	-0.4618	0.7110	0.1887	0.2219	0	0.8348	0	0	0	0	0.3573	0	-0.2752
0.3115	0	0	0	-0.7867	0	-0.5304	0	0	0.4607	-0.7865	0	0	-0.1543	0	-0.9550	0.5576	0	0	-0.1677	0	0.5099	0	-0.0096	0	-0.9009
0	0	0	0.1376	0	0	0	0.0170	0	-0.0228	0	0	-0.2181	0.0957	0	0	-0.1531	0	0	0	-0.1033	0	0	-0.6206	0.1720	-0.0209
0	0.2926	0	0	0	0	0	0.0215	0	0	0	0	0.6628	0	0	-0.3746	-0.8184	0	0	0	-0.2684	-0.1152	0	0	0	0
0	0	-0.5130	0	0.5498	-0.6322	0	0	-0.6304	0	0	0.0009	0.6067	0	-0.1435	0	-0.4671	0	0	-0.4871	0	0	0	-0.7048	0.3328	-0.7538
0	0.5094	0	0	0.6346	-0.5201	-0.9140	0.5897	0.8098	-0.0823	0	0	0	0.9661	0	-0.6425	0	-0.2265	0	0.2269	0.2558	-0.2815	0	-0.8901	-0.8330	0
0.5155	-0.4479	0	0	0.7374	-0.1655	0	0	0.9595	0.9262	0	0	-0.2015	-0.3971	0	-0.1542	-0.4380	0.8320	0.1524	0.1645	0.5440	0.4727	0	0	0	0
0.4863	0.3594	0	0	-0.8311	0	0	-0.2428	0	0	0.7818	0.3639	0.0538	0	0.1790	0	0	-0.9977	0	0.0815	0.8657	0	0	0.1211	0.3219	-0.6219
0	0	-0.4978	-0.3776	-0.2004	0.8054	0.4634	0	-0.7778	0	-0.3317	-0.9151	-0.1664	0.3327	0	0.1970	0	0	0	0	0	0.3668	0	0.8592	0.4595	-0.9147
0.2028	0.5792	0.5984	-0.9009	-0.4336	0.3069	-0.0207	0.9457	0.4970	0.1357	-0.4021	-0.4878	0.7731	-0.1064	0.6320	-0.8033	0.7192	-0.9447	0.7983	0.7999	0.0482	-0.7596	-0.6444	0.4122	0.6627	-0.9303
-0.6576	0	0	0	0	0	-0.0982	-0.2985	0	0	0	0.0433	0	0.3962	0	0.3919	0.7507	-0.0782	0.2889	0	-0.7223	0	0	0.1656	0.9646	0.4363

Figure 3.8 The chromosome matrix of NN_1 of dimension *25x26* displaying all its element values.

0	-0.0215	0	0	-0.4160	-0.4964	0.3998	0	0.5403	-0.7616	0.0288	0	0.5381
-0.4430	0	0	0	0	0	0.2771	0.8872	0	0.8797	0	0	0
0	0.8001	0.4894	0	0	0.2342	-0.9328	0.2754	0	0.2911	0	0.9778	0.8566
0.9150	-0.2615	0	0	0.9681	-0.4694	-0.8624	0.9154	-0.0573	0	0	0	0.1602
0	-0.7776	0	0.8094	-0.6657	0.6488	-0.3608	-0.5186	-0.9285	0.2786	-0.6003	0.7309	0
0	0.5605	0	0.2197	-0.7876	0.9653	0	0.3522	0	0.0894	-0.1861	0.2251	0
0	0	-0.2630	0.2353	-0.2552	0	0	-0.4219	0	0	0.4974	0.9799	0.7254
0.9143	-0.5166	0	0	0	-0.3122	0	0	-0.0530	0.0878	0.6512	0	-0.0314
-0.0292	-0.1922	0	0	0	0.1681	0	0.3903	0	0.4421	0.5799	0	0
0	-0.8071	0	0.1534	-0.3210	-0.7845	0	-0.8640	-0.3178	0.0450	0	0	-0.5812
0	0	0	-0.6342	0	0	0.9373	0	0	0	0	0	0.1046
-0.1565	0	0.5514	-0.5201	0.8407	0	0.0627	-0.5519	0	-0.5626	0	-0.0038	0.2598
0.8315	0	0	0.7730	-0.8946	0	0	0	0.4769	0	0	0.8017	0
0	0.1504	-0.1283	0	0.4757	0	0	0.6888	-0.5143	-0.7806	-0.7274	0.1493	0.2294
0	-0.8804	-0.1064	-0.0202	-0.4618	0.1887	0.2219	0	0.8348	0	0.3573	0	-0.2752
0.3115	-0.5304	0	0	-0.1543	-0.9550	0.5576	0	0	0	-0.0096	0	-0.9009
0	0	0.0170	0	0.0957	0	-0.1531	0	0	-0.1033	-0.6206	0.1720	-0.0209
0	0	0.0215	0	0	-0.3746	-0.8184	0	0	-0.2684	0	0	0
0	0	0	0.0009	0	0	-0.4671	0	0	0	-0.7048	0.3328	-0.7538
0	-0.9140	0.5897	0	0.9661	-0.6425	0	-0.2265	0	0.2558	-0.8901	-0.8330	0
0.5155	0	0	0	-0.3971	-0.1542	-0.4380	0.8320	0.1524	0.5440	0	0	0
0.4863	0	-0.2428	0.3639	0	0	0	-0.9977	0	0.8657	0.1211	0.3219	-0.6219
0	0.4634	0	-0.9151	0.3327	0.1970	0	0	0	0	0.8592	0.4595	-0.9147
0.2028	-0.0207	0.9457	-0.4878	-0.1064	-0.8033	0.7192	-0.9447	0.7983	0.0482	0.4122	0.6627	-0.9303
-0.6576	-0.0982	-0.2985	0.0433	0.3962	0.3919	0.7507	-0.0782	0.2889	-0.7223	0.1656	0.9646	0.4363

Figure 3.9 The genotype matrix of speciated individual $NN_{sp_{13,1}}$, with a matrix dimension of *25*x*13*.

In neural network architecture design, a typical three-layer network would enable the network to model any arbitrary function [175]. However, there is no precise approach to determining the appropriate network size, such as the number of hidden nodes, to prevent underfitting or overfitting. Most authors use trial-and-error estimation, an arbitrary scaling factor, or general rule-of-thumb methods to provide a starting point. One rule-of-thumb as a starting point to determine the number of hidden nodes would be two-thirds the size of the input nodes plus the output nodes. In applying EVLNN to the real-world problems in this work, the value of H or maximum evolvable hidden node size is set to 26.

An illustration of an EVLNN population at initialization is shown in Figures 3.10 and 3.11. Figure 3.10 shows a scatter plot of a speciated population P' consists of 100 individuals. Figure 3.11 shows a histogram plot of P' with a normal distribution with P'=*{SP5, SP6 ,..., SP18}*. It is observed that *SP1* to *SP4* and *SP19* to *SP26* have zero individuals. *SP13* is the most populous, with 15 individuals, *SP13* =$\{NN_{sp_{13,1}}, NN_{sp_{13,2}}, \ldots, NN_{sp_{13,15}}\}$ while *SP7* and *SP18* are the least populous with only one individual each, where *SP7* =$\{NN_{sp_{7,1}}\}$ and *SP18* = $\{NN_{sp_{18,1}}\}$.

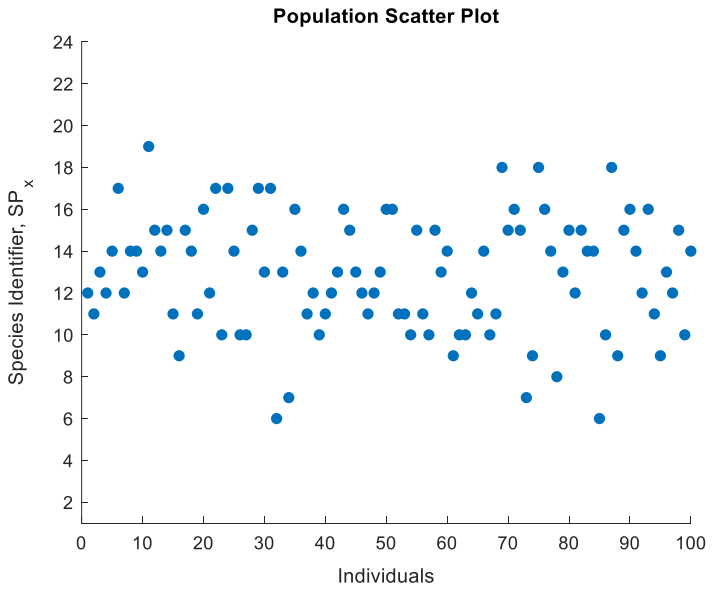


Figure 3.10 Species distribution in a population as seen in a scatter plot.

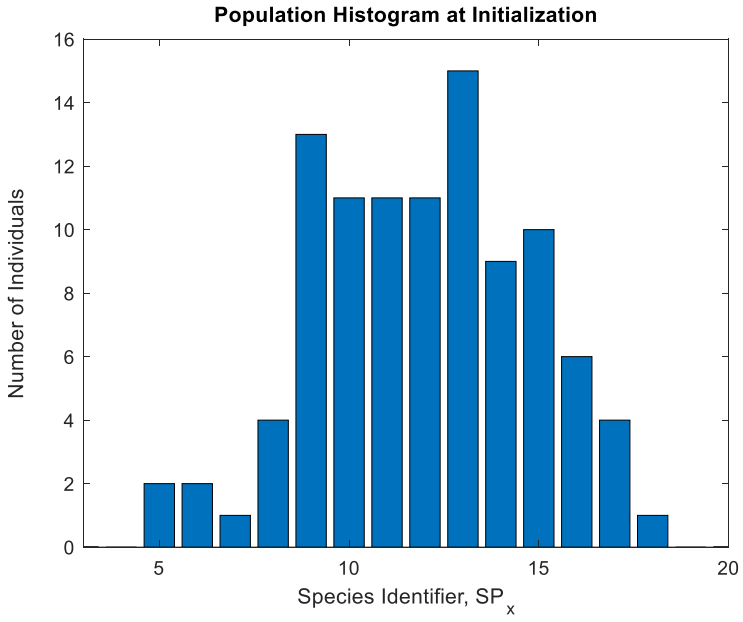


Figure 3.11 Species distribution of a population as seen in a histogram.

The number of hidden nodes in an ANN determines the dimensionality of the search landscape. It is, therefore, a direct indicator of the complexity of the function the network is capable of modeling [176]. The specific connectivity pattern of the phenotype will determine what portion of this space is being explored. The search space for the ANN phenotype can range from a network with a high number of hidden nodes and a high number

of connections (dense network) to a network with a low number of hidden nodes and a low number of network connections (parsimonious network). EVLNN is designed to model complex nonlinear functions with good generalization keeping the network parsimonious. The model achieves this by maintaining species parallelism, searching for promising solutions while locating new solutions in the landscape. The aim is to enhance the search ability by intensifying the search toward regions of interest while diversifying the search to learn different parts of the landscape.

3.3.2 Ranking and Selection

With the population genotypically speciated, individuals are then ranked according to their fitness within their respective species. The fitness of the individual F_i, is defined as

$$F_i = \frac{1}{1+M_i} \tag{3.24}$$

where M_i represents individual MSE expressed as

$$M_i = \frac{1}{n} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2 \tag{3.25}$$

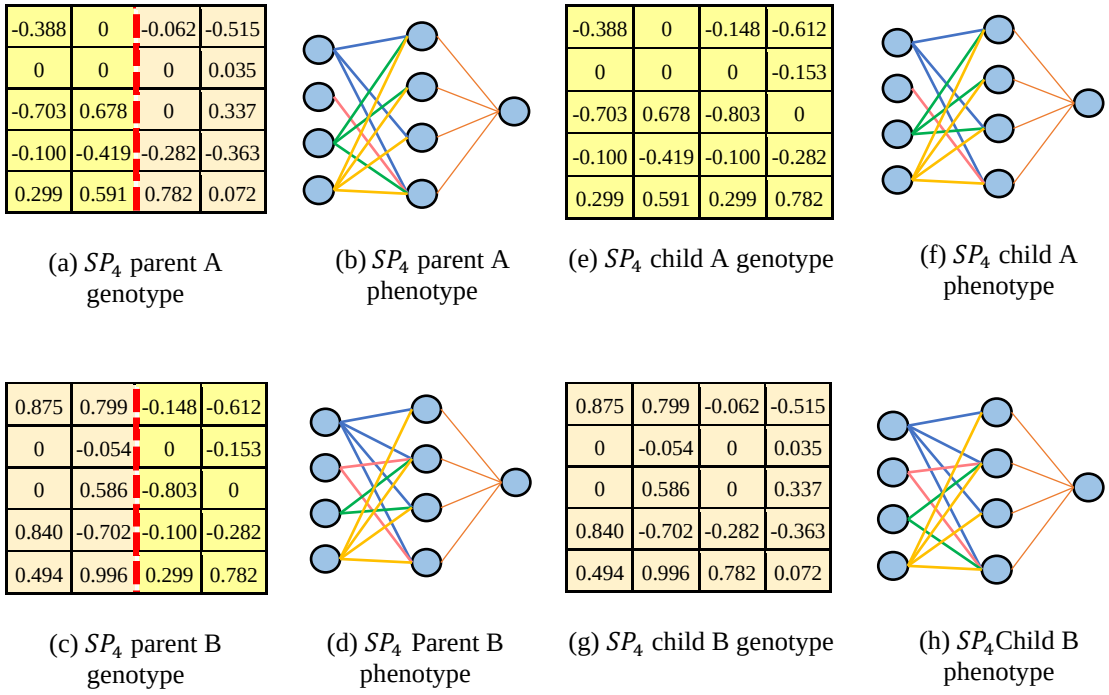
where y_t is the known target and $\hat{y}_t$ is the prediction at time *t.*

In applying selection pressure, well-known selection methods for GA were studied [177]. These methods can be broadly categorized into proportional and elitist. The former maintains genetic diversity by preventing the population from converging to a local minimum but increasing convergence time. While the latter increases convergence speed, it is more likely to converge on a local minimum due to loss of genetic diversity. This work used Stochastic Universal Sampling (SUS) for its minimum spread and non-bias sampling scheme [178]. This sampling scheme aligns with the EVLNN's species parallelism strategy to protect early structural innovation and prevent the fittest individuals from prematurely dominating the candidate solution space.

3.3.3 Intra-Species Crossover

The intra-species crossover takes advantage of species parallelism to investigate multiple basins simultaneously. In the intra-species crossover, breeding is restricted to genotypically similar individuals to preserve the genetic differentiation and parallelism of the population. By limiting the genetic drift, genotypically similar individuals reproduce after their kind and improve exploitation within the respective solution basins [179] [180].

The intra-species crossover process in the EVLNN algorithm is illustrated in Figure 3.12, using the recombination of Species_4 as an example. The genotypes of Parent A and Parent B belong to SP_4, respectively is shown in Figure 3.12(a) and Figure 3.12(c). When these parents recombined at the mid-point of their genotypes, Child A and Child B are produced, with their genotypes shown in Figures 3.12(e) and 3.12(g), respectively. It is observed that intra-species crossover produces offspring that belongs to Species_4, as shown in Figures 3.12(f) and 3.12(h). The intra-species crossover process preserves a differentiated landscape while refining the solution at multiple basins.



(a) SP_4 parent A genotype

(b) SP_4 parent A phenotype

(e) SP_4 child A genotype

(f) SP_4 child A phenotype

(c) SP_4 parent B genotype

(d) SP_4 Parent B phenotype

(g) SP_4 child B genotype

(h) SP_4Child B phenotype

Figure 3.12(a-h) Example of intra-species recombination process for Species_4.

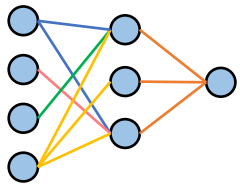
3.3.4 Inter-Species Crossover

A second recombination strategy, the inter-species crossover, is proposed to complement intra-species crossover. In particular, this strategy allows the exploration of new ANN structures in the architecture landscape for search optimization. Inter-species crossover strategy sporadically recombines dissimilar genotypes to generate novel solutions across genotypic boundaries. The intent is to create new possible solutions in the fitness landscape, contributing to exploration capability [181].

The inter-species crossover process is illustrated in Figure 3.13, using the recombination of Species_3 and Species_6 as an example. The genotypes of Parent A and Parent B belong to SP_3 and SP_6, respectively is shown in Figure 3.13(a) and Figure 3.13(c). When these parents recombined at the mid-point of their genotypes, Child A and Child B are produced, with their genotypes shown in Figures 3.13(e) and 3.13(g), respectively. It is observed that inter-species crossover produces offspring with phenotypes shown in Figure 3.13(f) and Figure 3.13(h) that do not resemble the phenotype of their parents shown in Figure 3.13(b) and Figure 3.13(d) in terms of the networks' hidden node size. Thus the algorithm has created new fitness landscapes to explore.

0.076	0	-0.439
0	0	0.697
0.820	0	0
-0.150	0.383	-0.302
0.018	0.986	0.234

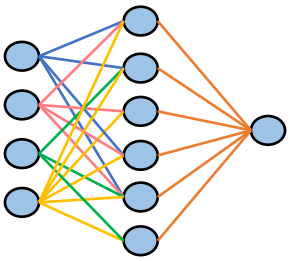
(a) SP_3 parent A genotype



(b) SP_3 parent A phenotype

-0.762	-0.370	0	0.771	0.110	0
-0.975	0	0.142	-0.636	-0.866	0
0	-0.163	0	0	0.265	-0.871
-0.743	0.674	0.120	-0.340	0.611	-0.313
-0.308	-0.682	-0.851	-0.213	-0.271	0.606

(c) SP_6 parent B genotype

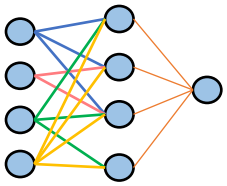


(d) SP_6 parent B phenotype

Figure 3.13(a-d) Inter-species crossover for Species_3 and Species_6 resulting in Species_4 and Species_5.

0.076	0.771	0.110	0
0	-0.636	-0.866	0
0.820	0	0.265	-0.871
-0.150	-0.340	0.611	-0.313
0.018	-0.213	-0.271	0.606

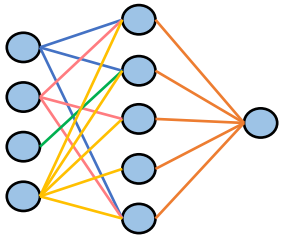
(e) SP_4 child A genotype



(f) SP_4 child A phenotype

-0.762	-0.370	0	0	-0.439
-0.975	0	0.142	0	0.697
0	-0.163	0	0	0
-0.743	0.674	0.120	0.383	-0.302
-0.308	-0.682	-0.851	0.986	0.234

(g) SP_5 child B genotype



(h) SP_5 child B phenotype

Figure 3.13(e-h) Inter-species crossover for Species_3 and Species_6 resulting in Species_4 and Species_5.

3.3.5 Weights Mutation

A two-stage mutation is introduced to the EVLNN algorithm to avoid the traps of local optima in complex problems. The aim is to inject small variations in the gene pool by producing incremental changes to the ANN architecture. The selected individual or mutant first undergoes a weights mutation followed by a link-node mutation.

The weights mutation stage perturbs the connection weights of the selected chromosome matrix by a small value between -0.5 and 0.5. A weights-change matrix, W_W, is first created with uniformly distributed random values between -0.5 and 0.5. A weights mutation probability matrix, P_W, is then formed when a matrix, R_W, a matrix with uniformly distributed random values between 0 to 1, is compared to the mutation probability. A value of '1' (or the Boolean value of 'TRUE') is assigned to an element in P_W if the respective R element is less than the mutation probability. A value of '0' (or the Boolean value of 'FALSE') is assigned if otherwise, as expressed in Equation 3.26,

$$P_W = \begin{cases} 1 & R_w \leq Mutation\ Probability \\ 0 & otherwise \end{cases} \tag{3.26}$$

After the weights mutation, a new individual, N' is resulted using the following matrix operations shown in Equation 3.27,

$$N' = N + (W_W \cdot P_W) \tag{3.27}$$

To illustrate, a sample matrix R_W, is shown in Equation 3.28,

$$R_W = \begin{bmatrix} 0.930 & 0.468 & 0.630 & 0.645 & 0.070 \\ 0.327 & 0.127 & 0.059 & 0.369 & 0.719 \\ 0.798 & 0.147 & 0.409 & 0.255 & 0.679 \\ 0.344 & 0.063 & 0.102 & 0.647 & 0.601 \\ 0.201 & 0.539 & 0.212 & 0.008 & 0.293 \\ 0.501 & 0.832 & 0.945 & 0.499 & 0.784 \end{bmatrix} \tag{3.28}$$

In EVLNN, the purpose of mutation is to introduce a slight adjustment to the chromosome matrix. Hence its probability is set to a low value, usually in the range of 1% to 2%. Supposing the *Mutation Probability* rate is 0.015, applying Equation 3.26, the weights mutation probability matrix, P_W is,

$$P_W = \begin{bmatrix} FALSE & FALSE & FALSE & FALSE & FALSE \\ FALSE & FALSE & FALSE & FALSE & FALSE \\ FALSE & FALSE & FALSE & FALSE & FALSE \\ FALSE & FALSE & FALSE & FALSE & FALSE \\ FALSE & FALSE & FALSE & TRUE & FALSE \\ FALSE & FALSE & FALSE & FALSE & FALSE \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.29}$$

Note that the element selected for weights mutation has a value of 0.008, less than the *Mutation Probability* rate of 0.015. If, given that the weights change matrix, W_W is,

$$W_W = \begin{bmatrix} 0.224 & 0.372 & -0.373 & 0.075 & -0.312 \\ 0.094 & 0.099 & 0.163 & -0.368 & -0.330 \\ 0.139 & -0.152 & 0.412 & -0.343 & 0.234 \\ -0.500 & -0.101 & -0.080 & 0.500 & 0.389 \\ -0.280 & -0.275 & -0.013 & 0.096 & 0.356 \\ -0.347 & 0.480 & -0.0372 & -0.152 & 0.464 \end{bmatrix} \tag{3.30}$$

and assuming the selected individual in SP_5 has a chromosome matrix, N given as,

$$N = \begin{bmatrix} 0.832 & 0 & 0.526 & 0.975 & 0.080 \\ 0 & 0 & -0.818 & 0.604 & 0 \\ 0.021 & -0.705 & 0.082 & 0.679 & -0.070 \\ -0.610 & 0.774 & -0.193 & 0.700 & -0.424 \\ 0.243 & 0 & 0 & -0.703 & 0 \\ 0.174 & -0.456 & -0.085 & -0.223 & 0 \end{bmatrix} \tag{3.31}$$

then the new individual, N', is obtained after applying Equation 3.27,

$$N' = \begin{bmatrix} 0.832 & 0 & 0.526 & 0.975 & 0.080 \\ 0 & 0 & -0.818 & 0.604 & 0 \\ 0.021 & -0.705 & 0.082 & 0.679 & -0.070 \\ -0.610 & 0.774 & -0.193 & 0.700 & -0.424 \\ 0.243 & 0 & 0 & -0.703 & 0 \\ 0.174 & -0.456 & -0.085 & -0.223 & 0 \end{bmatrix} + \left(\begin{bmatrix} 0.224 & 0.372 & -0.373 & 0.075 & -0.312 \\ 0.094 & 0.099 & 0.163 & -0.368 & -0.330 \\ 0.139 & -0.152 & 0.412 & -0.343 & 0.234 \\ -0.500 & -0.101 & -0.080 & 0.500 & 0.389 \\ -0.280 & -0.275 & -0.013 & \mathbf{0.356} & 0.096 \\ -0.347 & 0.480 & -0.0372 & -0.152 & 0.464 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right) \quad (3.32)$$

$$N' = \begin{bmatrix} 0.832 & 0 & 0.526 & 0.975 & 0.080 \\ 0 & 0 & -0.818 & 0.604 & 0 \\ 0.021 & -0.705 & 0.082 & 0.679 & -0.070 \\ -0.610 & 0.774 & -0.193 & 0.700 & -0.424 \\ 0.243 & 0 & 0 & -0.703 & 0 \\ 0.174 & -0.456 & -0.085 & -0.223 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{0.356} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.33)$$

$$N' = \begin{bmatrix} 0.832 & 0 & 0.526 & 0.975 & 0.080 \\ 0 & 0 & -0.818 & 0.604 & 0 \\ 0.021 & -0.705 & 0.082 & 0.679 & -0.070 \\ -0.610 & 0.774 & -0.193 & 0.700 & -0.424 \\ 0.243 & 0 & 0 & \mathbf{-0.347} & 0 \\ 0.174 & -0.456 & -0.085 & -0.223 & 0 \end{bmatrix} \quad (3.34)$$

EVLNN then performs an integrity check on boundary limits of the changed weights to see if it is between -1 and 1. If the mutated weight value exceeds the set limits, this weight value is assigned -1 or 1, depending on which end of the boundary is exceeded.

3.3.6 Link-Node Mutation

In the link-node mutation stage, a vector $V = [v_1, v_2, \ldots, v_n]$ is formed and derived from matrix N' where v_i is the i^{th} index location of the disabled links, and n is the number of disabled links in the matrix N'. A vector R_L of length n with uniformly distributed random values between 0 and 1 is created, and similarly, a second vector R_{L2} with uniformly distributed random values between -0.5 and 0.5. The R_{L2} vector is a placeholder for connection weights for the links to be re-enabled. A probability matrix for link-node mutation, P_L, is then formed using the expression in Equation 3.35,

$$P_L = R_{L2} \cdot (R_L \leqslant \textit{Link-node Mutation Probability}) \quad (3.35)$$

where *Link-Node Mutation Probability*, like the *Mutation Probability* in Equation 3.26, is kept small. Subsequently, a new individual, N'', is formed through re-enabling the disabled links in N' by assigning the weights from V to N' expressed in Equation 3.36,

$$N'' = N'(V) = P_L \tag{3.36}$$

To illustrate the link-node mutation process, Equation 3.37 shows the vector V of an individual in SP_5. The vector elements indicate the index location of the disabled links of that individual, N'.

$$V = \begin{bmatrix} 2 \\ 7 \\ 8 \\ 11 \\ 17 \\ 26 \\ 29 \\ 30 \end{bmatrix} \tag{3.37}$$

If, given that the matrix, R_L and R_{L2}, respectively, are,

$$R_L = \begin{bmatrix} 0.784 \\ 0.398 \\ 0.107 \\ 0.003 \\ 0.703 \\ 0.997 \\ 0.924 \\ 0.477 \end{bmatrix} \tag{3.38}$$

and,

$$R_{L2} = \begin{bmatrix} -0.051 \\ 0.228 \\ -0.253 \\ 0.010 \\ -0.295 \\ 0.005 \\ -0.402 \\ 0.083 \end{bmatrix} \tag{3.39}$$

And supposing that the *Link-Node Mutation Probability* rate is set to 0.015, applying Equation 3.35, the probability matrix P_L is,

$$P_L = \begin{bmatrix} -0.051 \\ 0.228 \\ -0.253 \\ 0.010 \\ -0.295 \\ 0.005 \\ -0.402 \\ 0.083 \end{bmatrix} \cdot \left(\begin{bmatrix} 0.784 \\ 0.398 \\ 0.107 \\ 0.003 \\ 0.703 \\ 0.997 \\ 0.924 \\ 0.477 \end{bmatrix} \leq 0.015 \right) = \begin{bmatrix} -0.051 \\ 0.228 \\ -0.253 \\ 0.010 \\ -0.295 \\ 0.005 \\ -0.402 \\ 0.083 \end{bmatrix} \cdot \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.010 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.40}$$

By applying Equation 3.36, the element in the fourth index position of P_L replaces the fourth disabled link in N' resulting in a new individual N''. Therefore, after a two-stage mutation, N'' is,

$$N'' = \begin{bmatrix} 0.832 & 0 & 0.526 & 0.975 & 0.080 \\ 0 & 0 & -0.818 & 0.604 & 0 \\ 0.021 & -0.705 & 0.082 & 0.679 & -0.070 \\ -0.610 & 0.774 & -0.193 & 0.700 & -0.424 \\ 0.243 & \mathbf{0.010} & 0 & \mathbf{-0.347} & 0 \\ 0.174 & -0.456 & -0.085 & -0.223 & 0 \end{bmatrix} \tag{3.41}$$

3.3.7 Fitness Evaluation and Termination Criteria

The cost function in Equation 3.25 guides the evolutionary process towards convergence. At the end of one evolution cycle, the health of the genotypes is evaluated, and the population is ranked globally following their fitness evaluation using Equation 3.24. The top five percent of individuals with the best fitness are chosen to pass to the next generation (elitist solutions), replacing the weaker ones. This approach ensures that the mean population fitness advances gradually and that any potential early stagnation is avoided [182]. The best models form the next generation, and the iterations continue with the evolutionary process if the termination conditions are not met.

The EVLNN algorithm checks the stop criteria at the end of each iteration. If the fittest individual meets a predefined value, or the maximum iteration is reached the evolutionary process tops. Else the whole evolution cycle repeats itself. For example, Figure 3.14 shows that EVLNN has achieved population convergence over 500 generations (or iterations) for the Himmelblau benchmark function used to evaluate EVLNN's performance in Chapter 4. The plot shows the falling average MSE converging towards the solution during the search process.

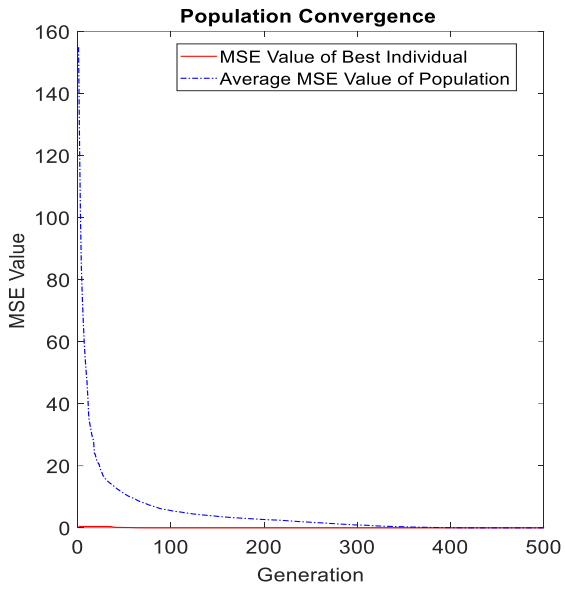


Figure 3.14 Population convergence plot for the search for global optima for the Himmelblau benchmark function.

3.3.8 Diversity Tracking

During the evolutionary process, chromosomes are subjected to reorganizations. Hence, measuring the diversity in a population of individuals is essential to better understand EVLNN's search characteristics, which can help to improve the algorithm's performance. In EA, individuals' and populations' diversity can be measured at either the genotypic or phenotypic levels. Phenotype constituted by a fixed number p of real parameters in the p-dimensional real space, $\mathbb{R}^p$ can be mathematically analyzed by diversity measures [183]. However, phenotypic level diversity measure does not lend itself to a straightforward approach for phenotype structure with variable topology and a number of parameters like the ANN [184].

In this regard, the genotype space's focus on genotypic diversity measures is more appropriate for EVLNN. Nonetheless, conventional diversity measures for the whole population can be obtained from the diversity measure for pairs of individuals and subsequently combining all the pairwise distances between individuals. Examples are the hamming distance, gene-level entropy, and chromosome-level hamming distance [185]. These approaches require genomes that have fixed lengths and uniform structures. Defining

a measure of diversity for a population with variable-length genomes becomes more complicated as conventional approaches lack the mechanisms to effectively handle EVLNN's variable-length genomes. This shortcoming has led to considering the Linguistic Complexity (LC) approach to population diversity measures [184].

The concept of LC is to afford appropriate models to measure a language's structural complexity. It generalizes from single strings to populations using the substring count to define the distance between strings. As population diversity is closely related to the concept of distance between individuals, a measure of diversity for the population can be obtained by estimating the number of different individuals that the population contains. In words, the diversity of the population is defined as *n* times the ratio of the total number of substrings in the population genome to the cumulative number of substrings in the genome of the individuals considered separately. The derived genome strings, s_{i_j} from all individuals within the population are then used to calculate the population diversity, *D(P)* of a population $P = \{i_1, i_2, \dots, i_n\}$ expressed in Equation 3.42 [184],

$$D(P) = D(\{i_1, i_2, \dots, i_n\}) = n \frac{\left|S_{\{i_2, i_2, \dots, i_n\}}\right|}{\sum_{j=1}^{n} \left|S_{i_j}\right|} \tag{3.42}$$

where n is the number of individuals in population P, $S_{\{i_1, i_2, \dots, i_n\}}$ is the total number of substrings, s_{i_j} in the population (that is, considering only once those appearing in the genome string of multiple individuals) and $\left|S_{\{i_1, i_2, \dots, i_n\}}\right|$ its cardinality, and S_{i_j} is the set of substrings of s_{i_j} with $\left|S_{i_j}\right|$ its cardinality.

To explain this relationship using an example, supposing there are three individuals, i_1, i_2, and i_3, with genomes $s_{i_1} = abc$, $s_{i_2} = bcb$, and $s_{i_3} = abad$. We have:

$$S_{i_1} = \{a, ab, abc, b, bc, c\}, \left|S_{i_1}\right| = 6 \tag{3.43}$$

$$S_{i_2} = \{b, bc, bcb, c, cb\}, \left|S_{i_2}\right| = 5 \tag{3.44}$$

$$S_{i_3} = \{a, ab, aba, abad, b, ba, bad, ad, d\}, \left|S_{i_3}\right| = 9 \tag{3.45}$$

$$S_{\{i_1,i_2,i_3\}} = \{a, b, ab, bc, abc, bcb, aba, c, abad, b, cb, c, ba, bad, ad, d\}, \left|S_{\{i_1,i_2,i_3\}}\right| = 16 \quad (3.46)$$

Therefore, applying Equation 3.42, the measure *D(P)* of diversity for the population constituted by the three individuals, i_1, i_2, *and* i_3, is,

$$D(\{i_1, i_2, i_3\}) = 3\frac{|S_{\{i_1,i_2,i_3\}}|}{|S_{i_1}|+|S_{i_2}|+|S_{i_3}|} = 3\frac{16}{6+5+9} = 2.4 \quad (3.47)$$

LC's string-based approach overcomes the shortcomings of conventional methods by supporting highly reorganizable genomes of variable length in the calculation. This feature is essential because of its speciation process and accedes to EVLNN's variable-length genomes. Hence, the possibility of using LC's substring count approach to define a distance between individuals belonging to populations with variable length genomes is explored. The approach is adapted and described below for EVLNN's implementation.

A step-by-step outline of the adapted approach is explained below:

1. **Convert genotype to its genome structure**: Individual genotype, i_j of the population $P = \{i_1, i_2 \ldots i_n\}$, where n is the number of individuals in the population temporarily converted to their respective genome structure by locating the non-zero elements in the genotype and replacing them with ones ('1').
2. **Convert genome structure to a string**: The element positions with the values of ones ('1') in the genome structure are retrieved to form a vector which constitutes the genome string, s_{i_j} of that individual, i_j.
3. **Compute population diversity**: The derived genome strings, s_{i_j} from all individuals within the population are then used to calculate the population diversity, *D(P)* using Equation 3.42.

To illustrate EVLNN's population diversity computation process adapted from the LC concept, supposing there are two genotypic individuals, i_1 and i_2 and their genome strings, S_{i_1} and S_{i_2} are shown in Equations 3.48 and 3.49, respectively,

$$i_1 = \begin{bmatrix} 0 & -0.348 & 0 \\ 0 & 0 & 0.492 \\ 0.628 & 0 & 0.914 \\ -0.583 & -0.569 & 0.239 \\ 0.023 & -0.345 & 0.295 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow S_{i_1} = \{3,4,5,6,9,10,12,13,14,15\} \quad (3.48)$$

and

$$i_2 = \begin{bmatrix} 0.865 & 0 & 0 & -0.521 \\ 0 & -0.649 & 0.211 & 0 \\ 0.170 & 0 & 0.914 & -0.688 \\ 0.873 & -0.109 & -0.311 & 0.129 \\ 0.723 & 0.052 & 0.295 & 0.566 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow S_{i_2} = \{1,3,4,5,7,9,10,12,13,14,15,16,18,19,20\} \quad (3.49)$$

The cardinality of S_{i_1} and S_{i_2}, are,

$$\left|S_{i_1}\right| = |\{3,4,5,6,9,10,12,13,14,15\}| = 10 \quad (3.50)$$

$$\left|S_{i_2}\right| = |\{1,3,4,5,7,9,10,12,13,14,15,16,18,19,20\}|] = 15 \quad (3.51)$$

and $S_{\{i_1,i_2\}}$ the total number of substrings is,

$$|S_{\{i_1,i_2\}}| = |\{1,3,4,5,6,7,9,10,12,13,14,15,16,18,19,20\}| = 16 \quad (3.52)$$

Therefore, applying Equation 3.42, the population diversity, *D(P)* for $P = \{i_1, i_2\}$ is,

$$D(\{i_1, i_2\}) = 2\frac{|S_{\{i_1,i_2\}}|}{|S_{i_1}|+|S_{i_2}|} = 2\frac{16}{10+15} = 1.28 \quad (3.53)$$

The explanation of the result above means that the two individuals correspond to 1.28 equivalent individuals. If the two individuals, S_{i_1} and S_{i_2} are identical, then,

$$D(\{i_1, i_2\}) = 2\frac{|S_{\{i_1,i_2\}}|}{|S_{i_1}|+|S_{i_2}|} = 1 \quad (3.54)$$

If the two individuals are completely diverse, *D(P)*=2. Hence $1 \leq D(P) \leq n$, where *n* is the size of the population.

Although the adapted LC diversity measure provides insight into the diversity of the individuals within the population at time *t*, it could not show the relationships between the number of species and the number of individuals present. To achieve this, additional

diversity measures were introduced to EVLNN. Learning from studies in ecological speciation, a simple and effective practice is to use Shannon's diversity and Shannon's equitability. The Shannon Diversity index, *H,* provides information about species richness, specifically, the number of species present and the relative abundances of these different species. Here, *H* is defined as,

$$H = -\sum_{i=1}^{S} p_i \ln(p_i) \tag{3.55}$$

where p_i is the proportion of the total sample represented by species *i*, ln is the Natural log, and S is the total number of unique species. The higher the value of *H*, the higher the diversity of species in the population. A higher *H* value also means a higher number of species and the evenness of their abundance. The lower the value of *H*, the lower the diversity. A value of *H*=0 indicates no diversity or that the population only has one species.

The Shannon Equitability Index, E_H provides information about species evenness, specifically, how similar the abundances of different species are, is defined as,

$$E_H = \frac{H}{H_{max}} \tag{3.56}$$

where $H_{max} = \ln(S)$ is the maximum diversity possible, and *H* is the Shannon diversity index. The E_H value ranges from 0 to 1, where 1 indicates complete evenness, that is, all groups have the same frequency. To provide some perspective, the following is an example that shows the calculation *H* and E_H for two given populations with the data shown in Table 3.1.

Table 3.1 Calculation of H and E_H for two population samples.

Species	**Frequency**	p_i	$ln(p_i)$	$p_i * ln(p_i)$	$H = -\sum_{i=1}^{5} p_i \ln(p_i)$	$E_H = \frac{H}{H_{max}}$
Population A (100 individuals, five species)						
A	40	0.40	-0.92	-0.37		
B	21	0.21	-1.56	-0.33		
C	18	0.18	-1.71	-0.31	1.47	0.92
D	9	0.09	-2.41	-0.22		
E	12	0.12	-2.12	-0.25		
Population B (100 individuals, five species)						
F	20	0.20	-1.61	-0.32		
G	20	0.20	-1.61	-0.32		
H	20	0.20	-1.61	-0.32	1.61	1.00
I	20	0.20	-1.61	-0.32		
J	20	0.20	-1.61	-0.32		

In Table 3.1, though Population A and Population B have the same number of individuals and species, H is higher for Population B than Population A. The higher H value indicates that Population B is more diverse in the evenness and abundance of species than Population A. For Population B, the E_H value is 1.0 shows that there is complete evenness of species in this population.

Figure 3.15 shows a sample diversity measure chart where EVLNN tracks three diversity indices, the population diversity (Equation 3.47) using the LC approach, the Shannon Diversity Index (Equation 3.55), and the Shannon Equitability Index (Equation 3.56).

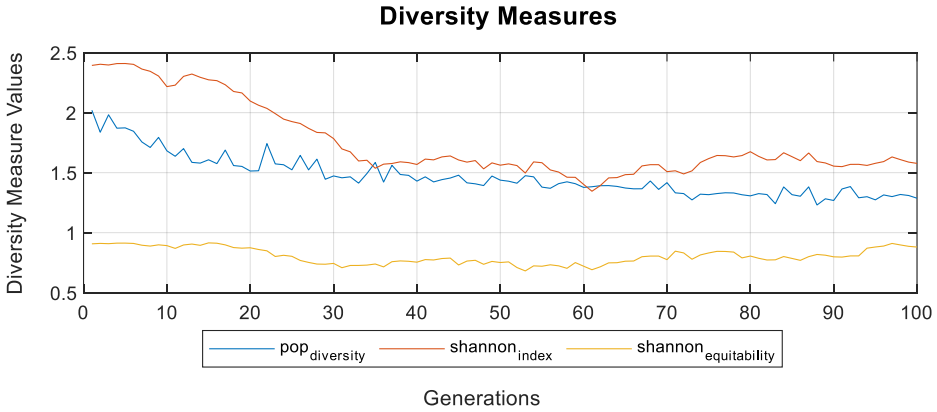


Figure 3.15 The diversity measurements chart produced by the EVLNN algorithm.

The plot provides insight into EVLNN species diversity, richness, and evenness. The overall population diversity reduces from a value of two to about 1.3, indicating the population converges. However, the solutions are not entirely identical at the end of the

evolution cycles. The Shannon Diversity index declines from a value of about 2.5 to about 1.5 at generation 35 onwards, indicating the number of species is reducing but stabilizes at generation 35. Shannon Equitability remains about 0.7 to 0.9, indicating changing species unevenness throughout the evolutionary process. These measures suggest that EVLNN has maintained diversity.

3.3.9 Interpretability of EVLNN

ANN are black-box models, and interpreting them is a significant challenge. With interpretability, ANN can considerably increase their adoption, particularly in the field of energy predictions, where valuable insights into the underlying system structure can be advantageous.

Sensitivity Analysis (SA) approaches are often applied to black-box neural networks to attribute the output responses' importance to the input variables' contribution. However, these approaches are not straightforward, as some degree of inconsistency would be expected when interpreting the causal relationship between the output variable and a predictor of interest. In most cases, it has not been possible to reach a consensus on the best-performing method as SA methods can produce varying outcomes [183] – [185]. A closer look at the above literature revealed that instability exists due to the SA methods applied. Most prior work has applied SA methods to a single trained neural network for analysis, which can have varying results. This inconsistency arises from the stochastic nature of the neural network modeling approach. The final trained state is determined by several factors, such as the network architecture and the initial random weights used [186].

To overcome this inconsistency, several authors have proposed using ensemble ANN and averaging the network errors [184] – [186]. Pentoś [186] proposed the use of three SA methods, namely the Partial Derivatives (PaD) method, the Connection Weights (CW) method, and the Statistical Methods to SA on an ensemble of 20 ANN architecture to reduce the inherent instability. Luíza da Costa et al. [187] proposed a voting approach to evaluate the importance of rankings in the contribution of the input to the output of a trained ANN generated by five weight-based sensitivity analysis methods, namely Garson [188], Yoon et al. [189], Tsaur et al. [190], Howes and Crook [191], and Olden and Jackson [192].

Instead of using the magnitude of variable importance, the most voted importance orders were considered to determine the final importance order. However, the approach is biased towards a homogeneous ensemble of weight-based sensitivity analysis methods. J. de Ona and Garrido [193] proposed applying heterogeneous sensitivity analysis methods to a set of trained ANNs to obtain a ranking of relative importance for each ANN and each method. These methods are the Perturb, the Profile, the CW, and the PaD. Then an approach based on calculating the ranking of variable's relative importance for each method as a function of average importance values is obtained from every ANN in the set. However, using the magnitude of the variable's relative importance in the calculation can be misleading. It may indicate an erroneous importance order due to the differences in the ANN topologies and connection weights.

The aim of enabling interpretability in the EVLNN is to help identify major factors influencing the prediction. However, ensuring the outcome of SA methods can produce a reliable result is of great importance. To the author's knowledge, the proposed EVLNN's ensemble-based approach to sensitivity analysis is evaluated for the first time in this study [194]. The approach involves applying an ensemble SA methods described in [195]; namely, the PaD method [196] [197], the Perturb method [198], the Profile method [199] [200], and the CW method [192] [201], to a set of 50 identified EVLNN models based on parsimony. Subsequently, a data aggregation technique is employed to determine the relative importance of the input variable that influences the output. Figure 3.16 shows the block diagram of the proposed approach.

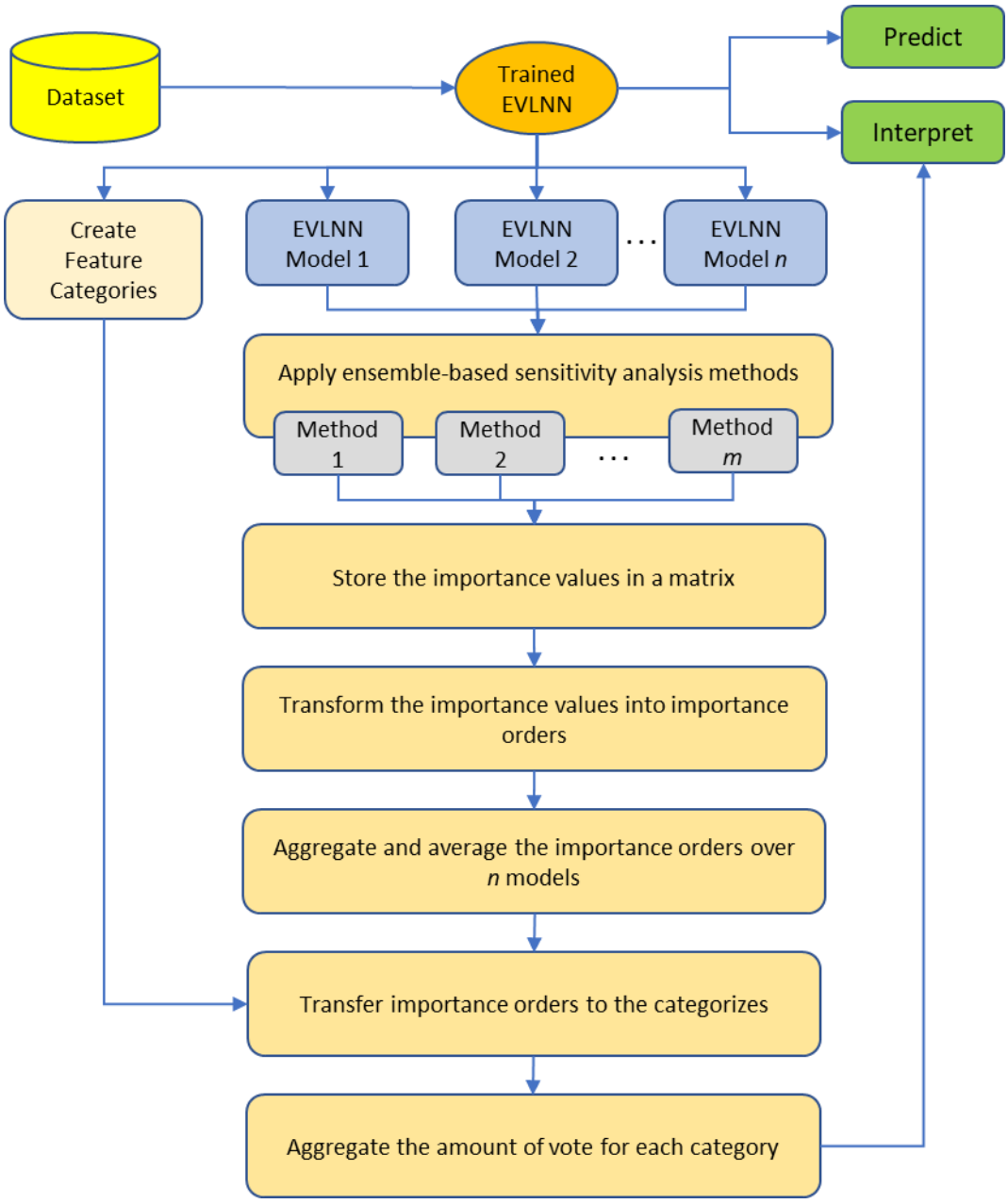


Figure 3.16 The proposed ensemble-based approach to sensitivity analysis

The approach is described below, with its pseudo-code and detailed implementation explained in Appendix B.

A step-by-step outline of EVLNN's ensemble-approached to sensitivity analysis is explained below:

1. **Trained EVLNN with the dataset**: Trained a set of *n* parsimonious EVLNN models.
2. **Analyze variables and create categories:** Analyze the input variables to identify and differentiate significant categories.
3. **Apply ensembled-based SA method**: Apply the PaD, Perturb, Profile, and CW methods to every one of the *n* models to compute the importance values of each variable.
4. **Store importance values in a matrix**: Store the calculated importance values of each variable by each method into a matrix.
5. **Transform the importance values into importance orders**: Based on the importance values, transform them into importance orders with the variables ranked.
6. **Aggregate and average the importance orders over *n* models**: Aggregate the importance orders over n models for each variable and method.
7. **Transfer importance orders to the categories**: Transfer the importance orders to their respective categories and rank them.
8. **Aggregate the amount of voting for each category**: Use a voting approach to evaluate the importance of ranking by aggregating the amount of voting. If there is a tie, the number of voting for the following positions is considered to break the tie.

This approach is applied to a real-world problem to determine factors influencing the energy consumption of the Hadoop system. The results are presented and discussed in Chapter 5, demonstrating better stability in EVLNN's interpretability, drastically reducing the potential inconsistency.

3.4 The EVLNN Algorithm for Handling Multimodal Functions

In order to validate and compare its performance to other optimization algorithms, the EVLNN algorithm design is subjected to a suite of benchmark test functions in Chapter 4. Benchmark test functions are artificially created to represent the nature of many real-world problems [202] to help validate the performances of optimization algorithms [203]. These

functions often consist of multimodal and multi-dimensional landscapes with multiple global optima. The steps below explain how the EVLNN algorithm handles multimodal function optimization, mainly how EVLNN's speciation concept and novel intra-species and inter-species crossovers and mutation are applied in function optimization.

A step-by-step outline of EVLNN's approach to multimodal function optimization is explained below:

1. Define the parameter values of EVLNN, such as the population size N_P, the maximum number of generations G_{max}, the number of species N_s, the intra-species crossover percentage XO_p, the inter-species crossover probability XS_p, the mutation percentage MU_p, the mutation value range MU_r, and the mutation matrix probability MX_p.
2. Initialize a population $\mathbf{X}$ of N_P solution candidates, $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{N_P}\}$ where each solution candidate $\boldsymbol{x}_i$ is a D-dimensional vector containing the variable values to be optimized, which are randomly and uniformly distributed between $[x_j^{low}\ x_j^{high}]$ for the range of the input domain such that,

$$x_{j,i} = x_j^{low} + \left(rand(0,1).\left(x_j^{high} - x_j^{low}\right)\right) \tag{3.57}$$

   where j=1, 2, …, D; i=1, 2, …, N_P with j and i being the variable and individual indexes, respectively. That is, $x_{j,i}$ is the j^{th} variable of the i^{th} individual.
3. Encode each solution candidate $\boldsymbol{x}_i$ of D-dimensional vector, into the form of a candidate chromosome matrix $\mathbf{Y}$ of variable length s, where $2 \leq s \leq N_s$, (N_s is the number of species),

$$\mathbf{Y} = \begin{bmatrix} x_{1,i} \\ \vdots \\ x_{j,i} \end{bmatrix} = \begin{bmatrix} x_{1,i}^1 & \dots & x_{1,i}^s \\ \vdots & \vdots & \vdots \\ x_{j,i}^1 & \dots & x_{j,i}^s \end{bmatrix} \tag{3.58}$$

   where the elements of the chromosome matrix are derived from dividing $x_{j,i}$ into s parts such that,

$$x_{j,i} = \sum(x_{j,i} .* \mathbf{W}) \tag{3.59}$$

   where $\mathbf{W}$ is a vector of a weighted coefficient given by,

$$\mathbf{W} = \frac{s}{\sum s} \tag{3.60}$$

where S is a uniformly distributed random integer with a vector of size between 2 and N_s using Equation 3.61 (the integers are generated through the function '*randi*' in MATLAB).

$$S = rand\big(1, randi([2, N_s])\big) \tag{3.61}$$

4. Speciate the population of candidate solutions differentiated by the length of the chromosome matrix s and obtain the size of each species.
5. Compute the function values of the i^{th} solution candidate and calculate its absolute error, ε_i is,

$$\varepsilon_i = |f_{min} - f(x_1, x_2, \dots, x_D)| \tag{3.62}$$

where f_{min} is the given function minima, and D is the number of variables or dimensions. The fitness F_i of the i^{th} chromosome matrix is subsequently calculated using the value of ε_i from Equation 3.62,

$$F_i = \frac{1}{1+\varepsilon_i} \tag{3.63}$$

6. Obtain the number of species and species size.
7. For each species, rank the solution candidates based on their fitness value.
8. Apply the Stochastic Universal Selection (SUS), a selection scheme employed to choose individuals for recombination and mutation. SUS is non-bias and ensures a minimum spread is maintained using a single random value to select the candidates at equally spaced intervals. Hence, it is likely that in SUS, weaker individuals of the sub-population have equal chances to be chosen and, therefore, do not allow the fittest individuals to dominate the candidate space, prematurely killing the diversity. The SUS example is illustrated in Figures 3.17 and 3.18. Using the individual's fitness, a selection probability for that individual is calculated as shown in Table 3.2 and mapped to a contiguous segment of a line such that the individual's segment is equal in size to its fitness. The mapping is illustrated in Figure 3.17 using the fitness information in Figure 3.18(b). The starting position of the first pointer is given by a uniformly distributed random number in the range [0, 1]. For ten individuals to be selected, the

distance between the pointers is 1/10=0.1. A modulo function is applied to ensure an even number of individuals are selected.

Table 3.2 Selection probability and fitness value.

Number of Individuals	1	2	3	4	5	6	7	8	9	10
Fitness Value	0.717	0.666	0.629	0.587	0.476	0.462	0.435	0.099	0.062	0.014
Selection Probability	0.17	0.16	0.15	0.14	0.11	0.11	0.10	0.02	0.01	0.00
Probability Cummulation	0.17	0.33	0.49	0.63	0.74	0.85	0.96	0.98	0.99	1.00

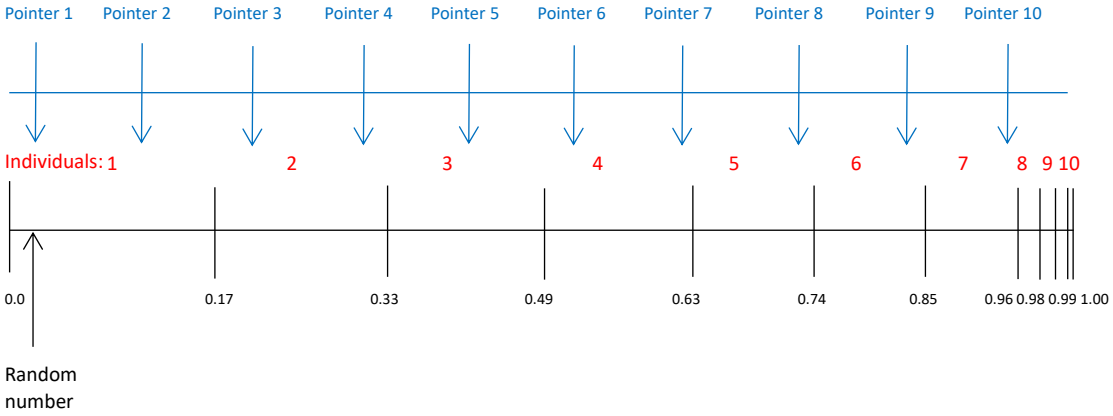


Figure 3.17 The SUS mapping of an individual's fitness to a contiguous segment. The selected individuals consist of the 1, 1, 2, 2, 3, 4, 4, 5, 6, and 7.

Figure 3.18 illustrates steps 6 to 8 using a sample species of ten individuals undergoing the selection process for recombination and mutation.

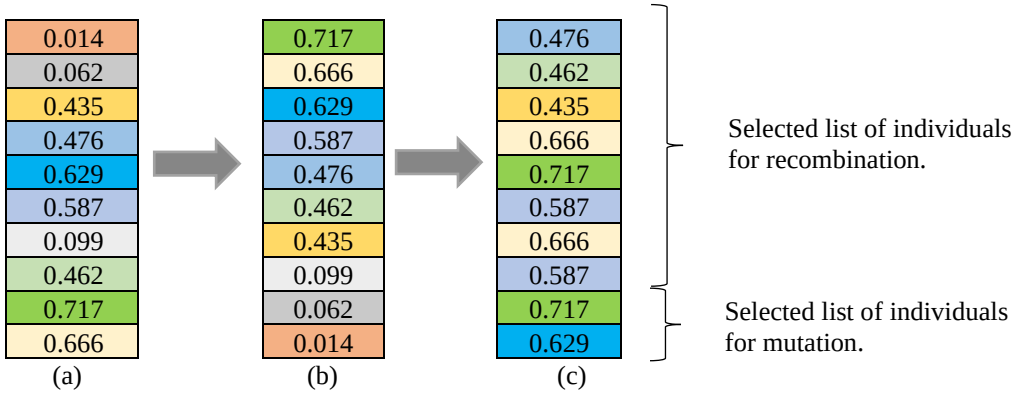


Figure 3.18(a-c) (a) A subpopulation of individuals within a species with corresponding fitness. (b) Individuals are ranked within the species according to their fitness. (c) Apply SUS to select potential candidates for recombination and mutation.

9. Perform recombination on the selected list by sequential pairing down the list.
   a. If there is only one individual in the species, then perform mutation.
   b. Generate a random number between 0 and 1 with normal distribution, and if the number $\leq XS_p$, perform inter-species crossover.
   c. Else perform intra-species crossover.
10. Perform mutation on a selected list of individuals using the expression,

$$\boldsymbol{Y_{new}} = \boldsymbol{Y} + (\boldsymbol{C} .* \boldsymbol{V}) \tag{3.64}$$

where $\boldsymbol{Y_{new}}$ is the new mutated individual of the original $\boldsymbol{Y}$, the candidate chromosome matrix in Equation 3.58, $\boldsymbol{C}$ is the change matrix with element values within the mutation range MU_r,

$$\boldsymbol{C} = 2 * M_r * rand(0,1) - MU_r \tag{3.65}$$

and $\boldsymbol{V}$ is the mutation matrix obtained from performing a logical comparison between a randomly generated matrix and the mutation matrix probability, MX_p,

$$\boldsymbol{V} = rand(0,1) < MX_p \tag{3.66}$$

11. Perform integrity checks on boundary limits.

12. Evaluate the offspring by comparing the fitness of the offspring with their parents. Weaker individuals are replaced with healthier ones. Repeat steps 7 to 12 until all species are evaluated.
13. Rank individual fitness population-wise, and maintain elitism EL_p of 5%, replacing individuals ranked in the bottom 5% with individuals from the top 5%.
14. Create a next-generation parent population.
15. If maximum generation reaches or stops, condition met, then stop. Otherwise, go to step 4.

3.4.1 Intra-Species Crossover for Low-Dimensionality Problems

Low-dimensionality test functions generally have two variables, x_1 and x_2 to optimized. To demonstrate the crossover process, Figure 3.19(a-b) shows two sample parents from *Species_6*, namely Parent_1 and Parent_2. Crossover occurs at the parents' chromosome matrix mid-point, indicated by the red dotted lines in Figure 3.18. The offspring produced is shown in Figure 3.20(a-b).

-0.5461	-0.1997	-0.4918	-0.4740	-0.1177	-0.0861
0.2917	0.1067	0.2627	0.2532	0.0629	0.0460

(a) Parent_1

1.5713	0.5747	1.4152	1.3640	0.3386	0.2478
0.1615	0.0591	0.1455	0.1402	0.0348	0.0255

(b) Parent_2

Figure 3.19(a-b) Sample chromosome matrices of Parent_1 and Parent_2 in *Species_6*.

-0.5461	-0.1997	-0.4918	1.3640	0.3386	0.2478
0.2917	0.1067	0.2627	0.1402	0.0348	0.0255

(a) Child_1

1.5713	0.5747	1.4152	-0.4740	-0.1177	-0.0861
0.1615	0.0591	0.1455	0.2532	0.09629	0.0460

(b) Child_2

Figure 3.20(a-b) Chromosome matrices of new offspring, Child_1, and Child_2, after crossover of Parent_1 and Parent_2 in *Species_6*.

For odd-numbered species, the crossover point in the chromosome matrix is determined by rounding up the mid-point value. For example, in *Species_9*, the midpoint crossover is $\frac{9}{2} =$ 4.5 round up to 5, that is, between the fifth and sixth column of the chromosome matrix.

3.4.2 Intra-Species Crossover for High-Dimensionality Problems

For the EVLNN algorithm optimizing functions with high dimensionality, the formation of the chromosome matrix is modified to take into account the high number of variables of the test functions by increasing the number of rows in the chromosome matrix. For example, Figure 3.21(a-b) features a pair of parents from *Species_4* where the respective chromosome matrix contains the values for variables x_1, x_2, x_3, x_4 *and* x_5.

-2.2016	-4.3602	-3.8046	-3.4728
2.6440	-4.5854	0.7075	3.0252
2.6083	0.3166	-0.3147	-1.9407
-1.2290	2.8698	-5.0176	0.2933
0.6972	4.4616	-1.6744	-3.4371

(a) Parent_1

1.0484	-0.5084	3.3494	-4.0436
-2.4367	-4.2783	0.3942	4.7483
1.5839	-2.7861	5.1003	-5.0924
1.9451	4.2491	-4.3364	2.8261
2.5510	-3.5736	-0.5893	3.2619

(b) Parent_2

Figure 3.21(a-b) Sample chromosome matrix of Parent_1 and Parent_2 in Species_4 for solving high dimensionality problems.

Similarly, recombination occurs at the parents' chromosome matrix mid-point, indicated by the red dotted lines in Figure 3.21(a-b). The offspring produced is shown in Figure 3.22(a-b).

-2.2016	-4.3602	3.3494	-4.0436
2.6440	-4.5854	0.3942	4.7483
2.6083	0.3166	5.1003	-5.0924
-1.2290	2.8698	-4.3364	2.8261
0.6972	4.4616	-0.5893	3.2619

(a) Child_1

1.0484	-0.5084	-3.8046	-3.4728
-2.4367	-4.2783	0.7075	3.0252
1.5839	-2.7861	-0.3147	-1.9407
1.9451	4.2491	-5.0176	0.2933
2.5510	-3.5736	-1.6744	-3.4371

(b) Child_2

Figure 3.22(a-b) Chromosome matrix of Child_1 and Child_2 are new offspring after recombination of Parent_1 and Parent_2 in Species_4.

3.4.3 Function Optimization using EVLNN – An Example

For illustration, the Himmelblau benchmark function is presented as an example of function optimization to demonstrate EVLNN's species parallelism and search capabilities. A series of runs were performed, extensive data were collected during the experiment, and the

algorithm's search characteristics were investigated and analyzed with visualization plots to explain EVLNN's steps in the search optimization process.

The Himmelblau-2D function has the mathematical equation expressed as,

$$f(x) = ({x_1}^2 + x_2 - 11)^2 + (x_1 + {x_2}^2 - 7)^2 \tag{3.67}$$

The function's 3D plot is shown in Figure 3.23, where the contour depicts a multimodal and multiple global optima landscape. Figure 3.24 shows the contour in 2D. The search is evaluated on $x_i \in$ [-6 6], for all i =1, …, d where d=2. The function has four global minima at f(x*)=0, at x* = (3, 2), x* = (-3.779, -3.283), x* = (-2.805, 3.131) and x* = (3.584, -1.848). Four large red squares mark the locations of the global optima in the search landscape.

Figure 3.25 shows the EVLNN's species distribution at initialization. There are in total 125 individuals and 14 species. As observed, individuals are not distributed evenly among the different species. SP_3 or *Species_3* has 13 individuals, the largest species, whereas *Species_6* and *Species_15* have five individuals each. They form the smallest species.

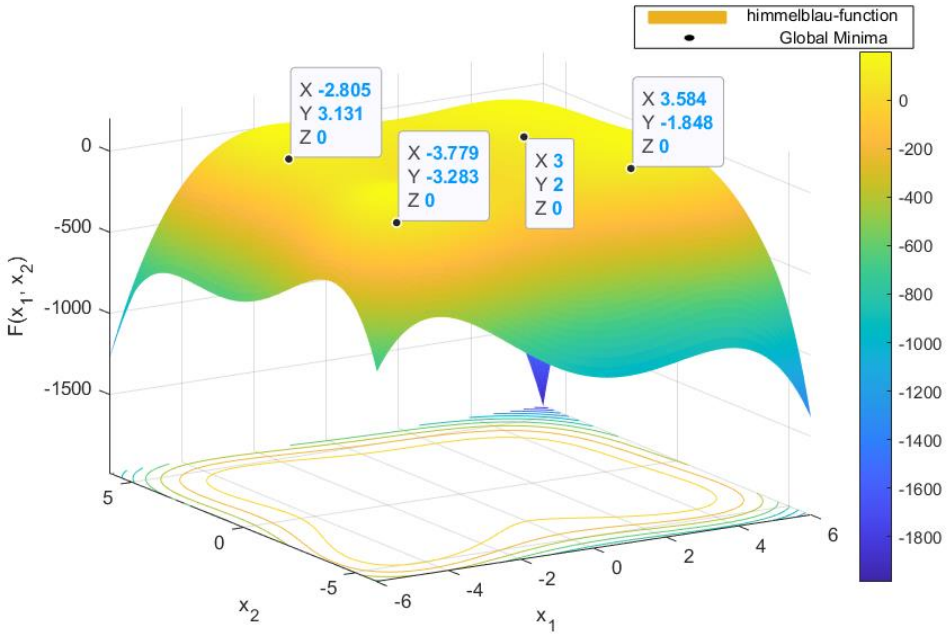


Figure 3.23 3D plot of the Himmelblau function with four global minima.

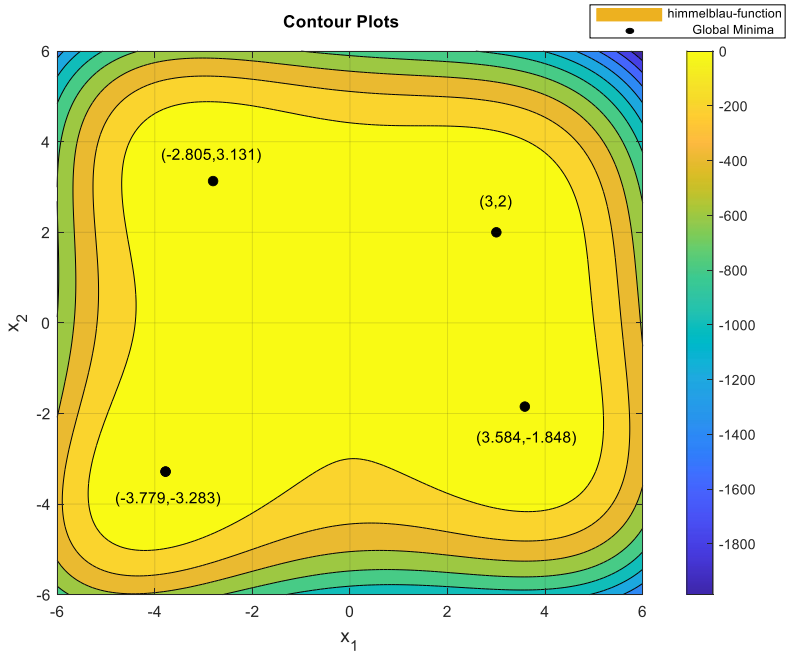


Figure 3.24 Contour plot of the Himmelblau function with locations of the four global minima.

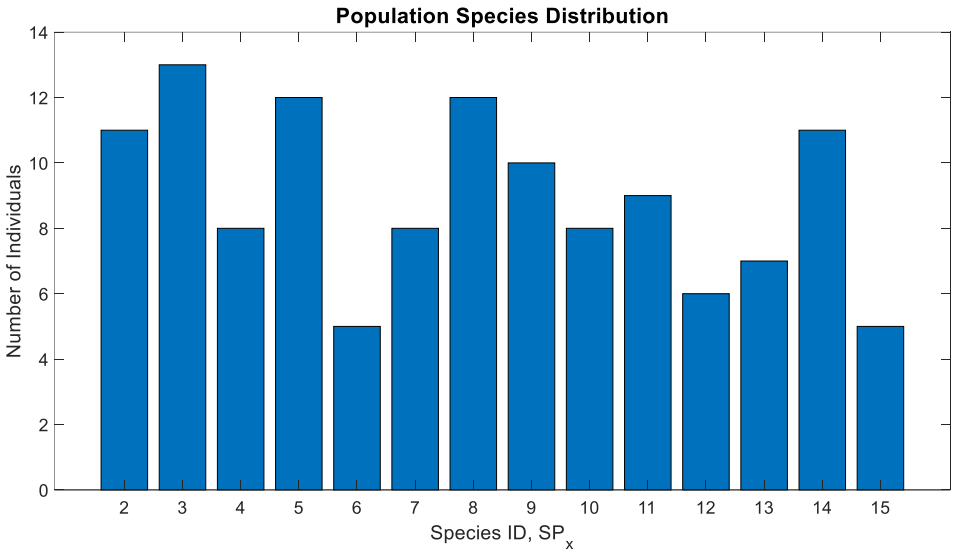


Figure 3.25 Species distribution at the initialization.

Figure 3.26 shows the population at initiation presented on a 2D search landscape of the Himmelblau function in the range [-6 6]. Potential solutions are represented by various colored shapes on the landscape with similar colored shapes belonging to the same species. The species are initially scattered over the search landscape, searching for attractive basins shown by the four red squares in this multimodal contour.

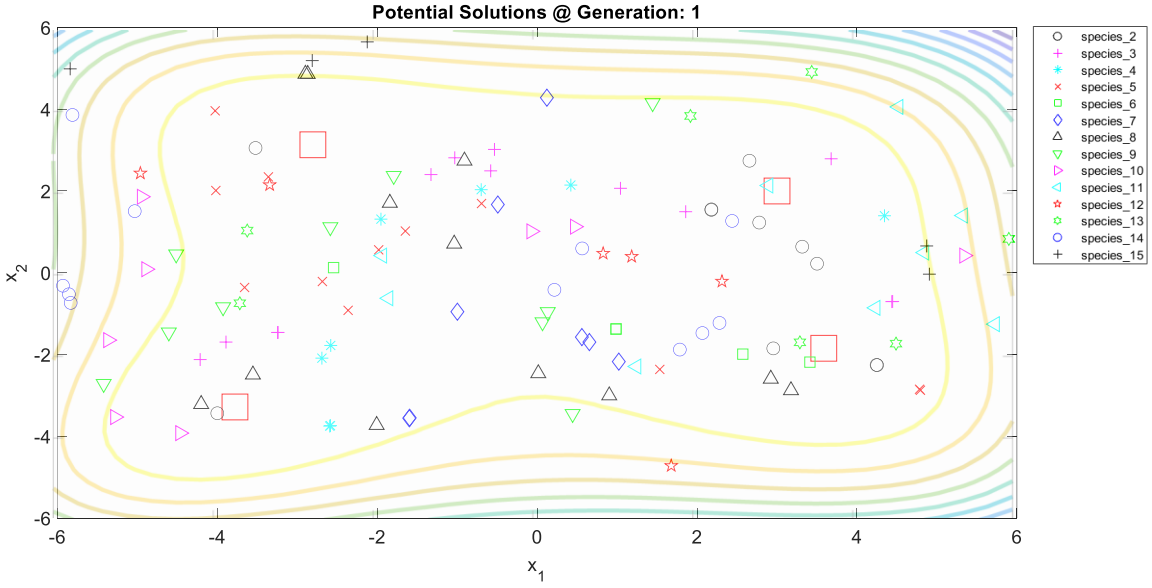


Figure 3.26 Landscape showing speciated solution candidates in generation one.

Figure 3.27 to 3.33 show the evolutionary map of the EVLNN algorithm captured in steps of ten generations from generation 20, 40, 60, to 100, then in steps of 200 generations from 100 to 500. These plots visually depict how the population evolves on the objective function surface where the various species search for the global optima. More granular search steps are illustrated in Appendix C.

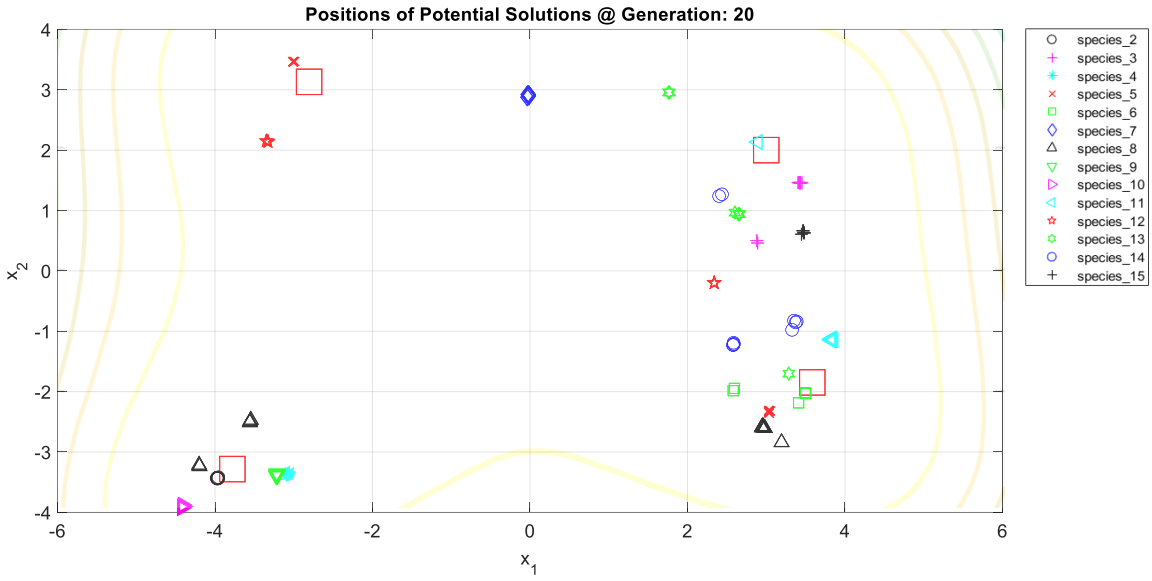


Figure 3.27 At generation 20, species are seen drawing closer to the minima.

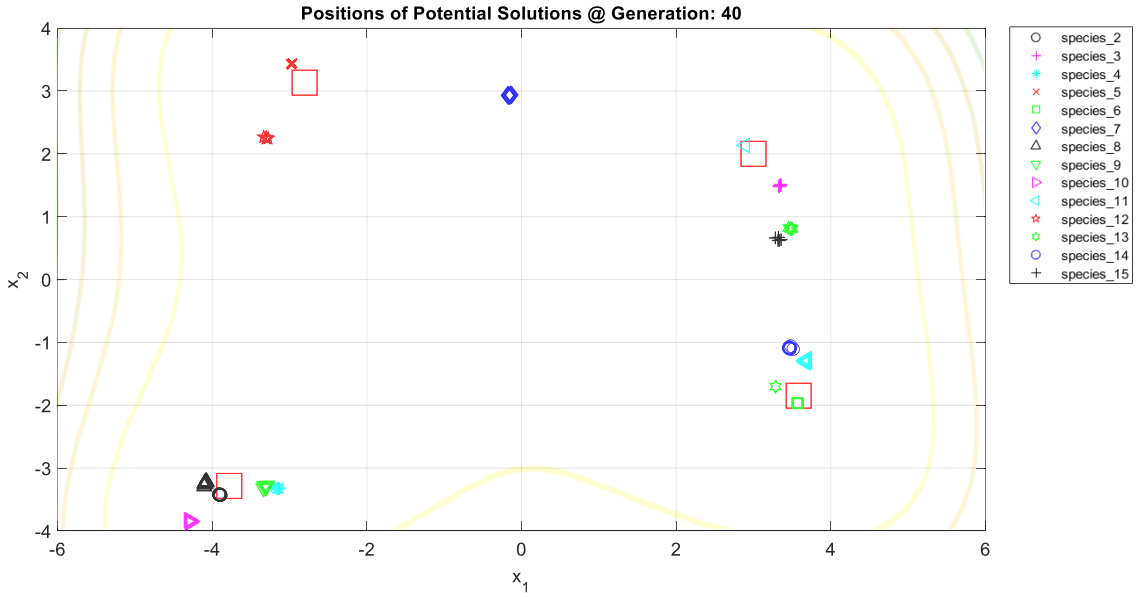


Figure 3.28 At generation 40, Species_8 has become identical to the other species' search positions.

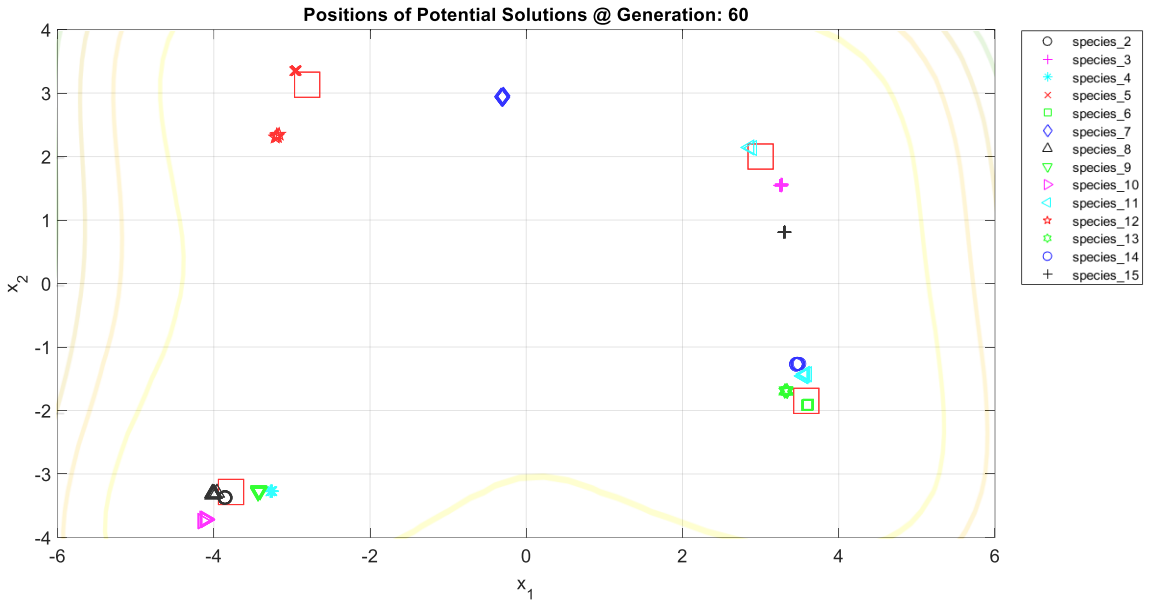


Figure 3.29 At generation 60, the search continues.

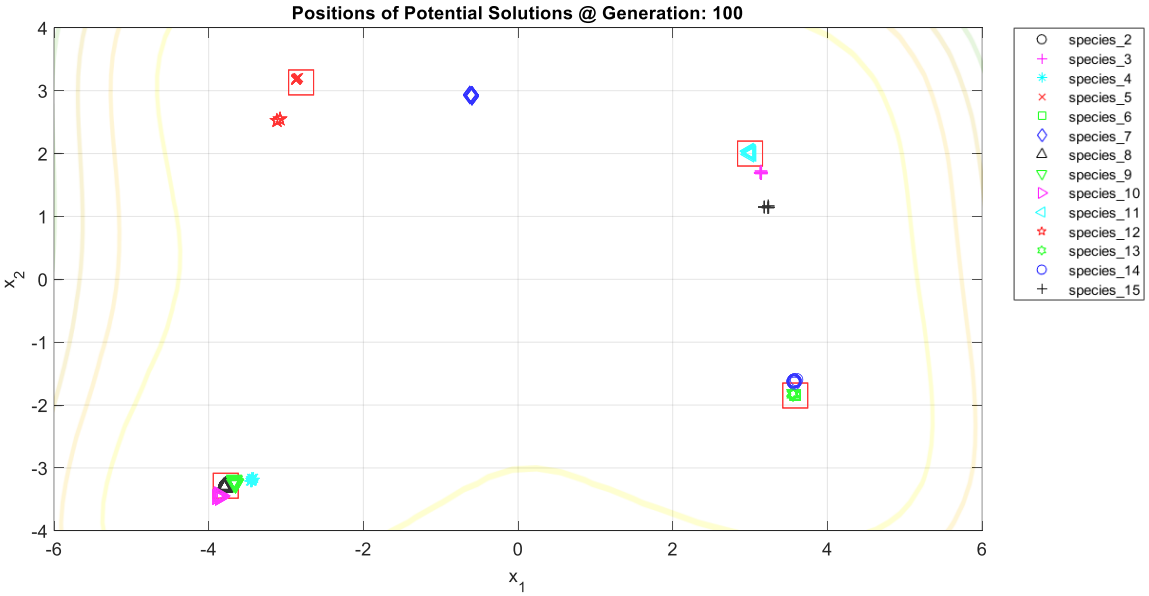


Figure 3.30 At generation 100, there is a clear path on the movement of these remaining species.

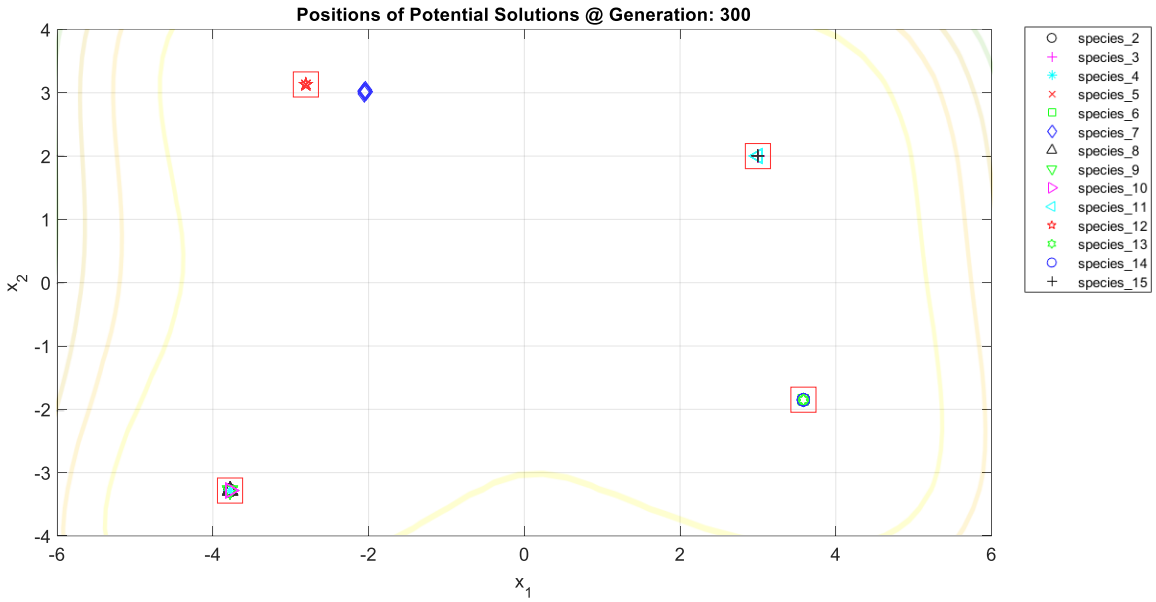


Figure 3.31 At generation 300, all the species except Species_7 are seen inside one of the red squares.

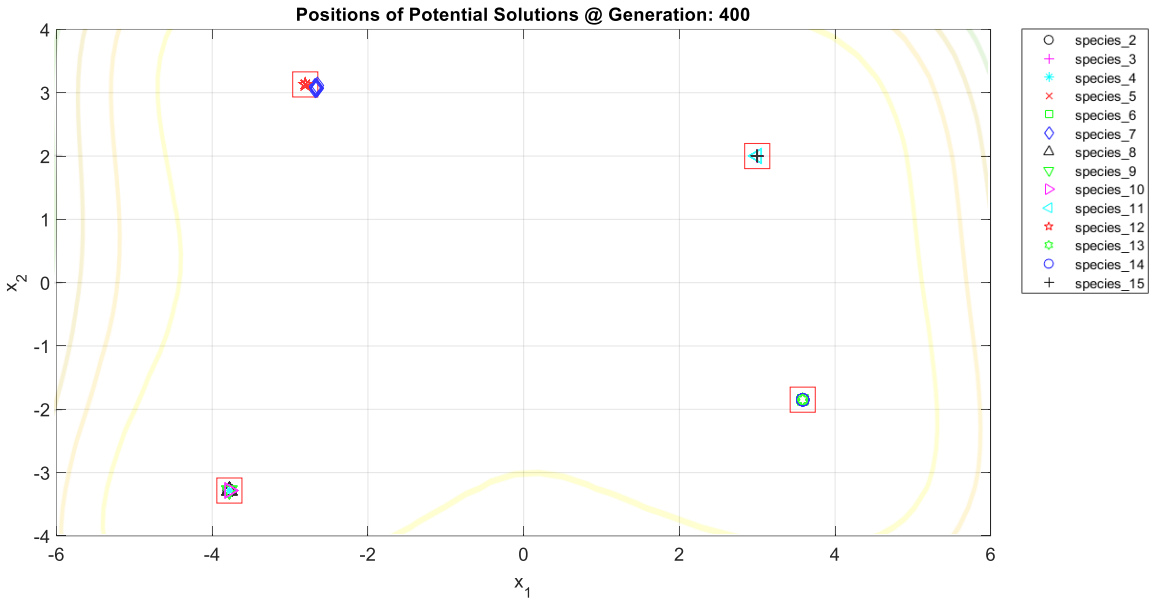


Figure 3.32 At generation 400, Species_7 has located one of the global minima.

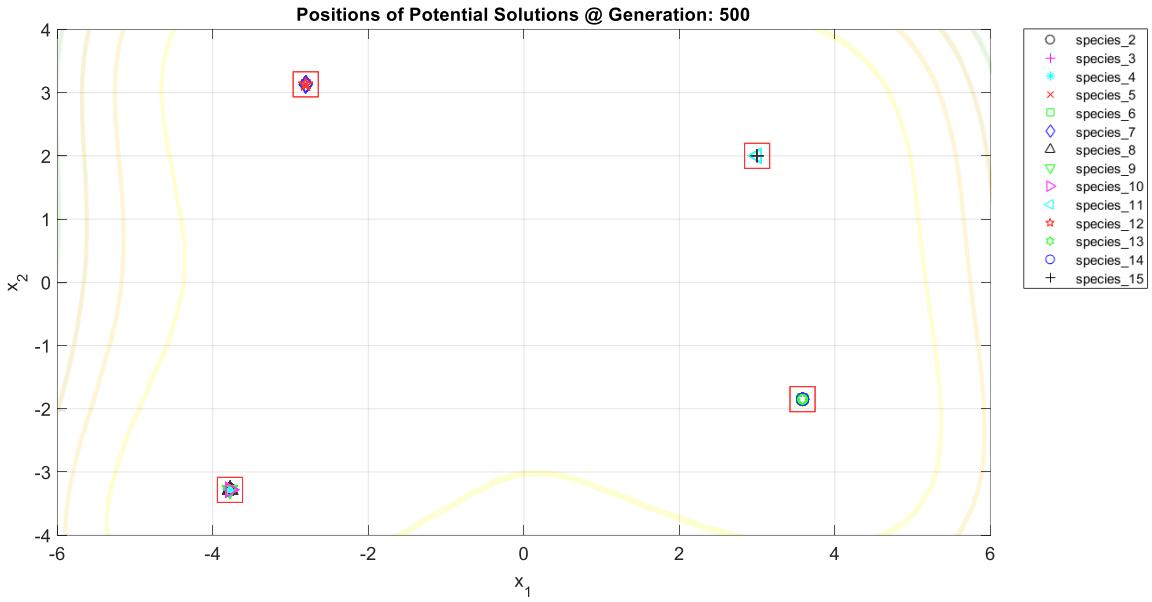


Figure 3.33 At generation 500, all the species are inside one of the red squares where the global minima are located.

Figure 3.34 shows the convergence rate of each of the 14 species, converging at a different rate to the global minima. At convergence, the global optima are found by SP_3, SP_2, SP_5, and SP_6, with the lowest error at 2.6998 x 10^{-6}, 1.1689 x 10^{-6}, 1.1638 x 10^{-5}, and 2.7659 x 10^{-7}, respectively.

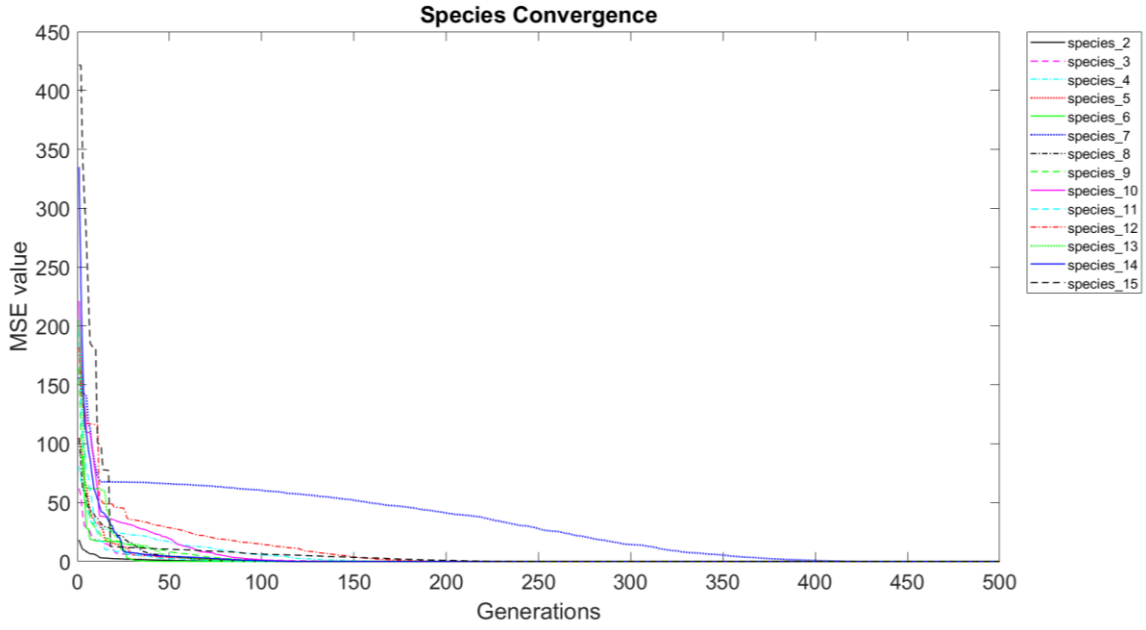


Figure 3.34 Individual species convergence over 500 generations.

3.5 Chapter Summary

In this chapter, the EVLNN framework is explained along with several novel mechanisms. Firstly, a structurally inclusive matrix encoding scheme is designed for model representation in the fitness landscape. Network information is encoded in a chromosome matrix, allowing the weights and structures to evolve simultaneously. Secondly, species parallelism is proposed for preserving structural innovation and giving potential solutions within the species a chance to thrive. It is achieved by introducing two crossover strategies, using intra-species crossover as the predominant recombination strategy but interspersing the computation with an occasional inter-species crossover. The strategies have maintained parallelization and intensified the search for promising regions of interest while exploring new parts of the fitness landscape. Thirdly, a two-stage mutation called the weights mutation and link-node mutation is introduced to prevent the search from being trapped in local optima by making incremental changes to the ANN weights and structures. The mutations are performed separately, injecting new alleles into the gene pool to prevent the population from stagnating too quickly. Fourthly, a diversity tracker mechanism is incorporated into the EVLNN algorithm to provide insights into the evolutionary search behavior that would benefit the fine-tuning of the GA parameters. Finally, an ensemble-based approach to sensitivity analysis is implemented into the structure of the EVLNN to help explain the relationship between the input variables and the output.

In addition, the EVLNN algorithm for handling multimodal function optimization is presented. While the purpose of EVLNN is to automate neural architecture search for energy prediction, its central concept of speciation can be applied to function optimization to evaluate the global search characteristic of EVLNN and the algorithm's generalization ability.

Chapter 4

4. Model Evaluation and Comparison

4.1. Introduction

This chapter describes the evaluation of the EVLNN algorithm using a broad set of benchmark test functions, including those recommended in the CEC 2013 and 2015 Competitions [204] [205]. As far as possible, the test suite covers various problems, including unimodal, multimodal, multi-dimensional separable, and non-separable, for a comprehensive assessment of the EVLNN algorithm. The performance of the EVLNN is then compared to classic GA, meta-heuristic methods such as PSO and DE, and the state-of-the-art niching algorithms from the CEC 2013 and 2015 Competitions.

It is worth noting that benchmark test functions are artificially created and are often well designed. In contrast, real-world problems are much more diverse and can be very different from these test functions [206]. Thus, optimization algorithms that work well for test functions may not work well in real-world applications. As such, additional experiment using real-world time-series energy load data is also considered in this chapter to assess the effectiveness of EVLNN in handling real-world prediction problems.

4.2. Test Methodology and Assumptions

The implementation of the benchmark functions has been chiefly based on the source code from the IEEE CEC 2013 competition [207] and public domain resources from [208] [209]. PSO, DE, and GA are implemented using public domain codes [210]. Standard parameter settings are used with no additional tuning when implemented in the PSO, DE, and GA algorithms.

The test made several assumptions. Firstly, the number of global optima is known. Secondly, the different distances between the global optima are known. Thirdly, modality information on the problem landscape is known. The EVLNN algorithm is executed for each problem from f_1 to f_{16} in Tables 4.1 to locate all global optima at a given accuracy level, ε where $\varepsilon \in \{1.0E\text{-}01, 1.0E\text{-}02, \ldots, 1.0E\text{-}05\}$. The accuracy ε, specifies the maximum allowable difference between a known global optima f_{opt} (or minima, f_{min}) and the predicted solution $f(x)$ expressed as,

$$\left|f_{opt} - f(x)\right| \leq \varepsilon \tag{4.1}$$

Since the number of global optima is known, the algorithm's performance in locating all global optimas over multiple runs can be measured in terms of the *peak ratio* (PR) *success rate* (SR) and the averaged number of evaluations to achieve a required accuracy ε. Given a fixed maximum number of function evaluations (MaxFE), and required accuracy level ε, PR, expressed in Equation 4.2, measures the average percentage of all known global optima found over multiple runs,

$$PR = \frac{\sum_{i=1}^{NR} NPF_i}{NR * NKP} \tag{4.2}$$

where NPF_i represents the number of global optima found during the i^{th} run, NKP represents the number of known global optima, and NR represents the number of runs. SR is expressed in Equation 4.3, which measures the success rate in the percentage of runs out of all runs,

$$SR = \frac{NSR}{NR} \tag{4.3}$$

where NSR is the number of successful runs and NR is the number of runs. The recommended value is NR=50 from the CEC benchmark standard [202]. Therefore, a value of 1.0 indicates that all fifty runs found all global peaks, whereas a value of 0.0 means none of the peaks are found in any of the fifty runs. The best solutions found after a given number of evaluations in multiple runs are then checked if it is within the niche radius, r to be considered the global optima. The mean global optima found averaged over fifty runs on

these test functions are then compared with the CEC 2013 and 2015 competition algorithms to assess EVLNN's competitiveness.

Table 4.1 presents the parameters for performance measurement for the benchmark test functions f_1 to f_{16} to determine if a niching algorithm has located all the global optima in the competition. The value of the niche radius r, the peak height, the number of global optima, and the maximum number of function evaluations *MaxFE* are referenced from [202].

Table 4.1 Parameters used for performance measurement.

Function	Description	*r*	Peak height	No. of global optima	*MaxFE*
f_1	Bohachevsky N.1 (2D)	0.5	0	1	2.0E+05
f_2	Booth (2D)	0.5	0	1	2.0E+05
f_3	Sphere (30D)	0.5	0	1	2.0E+05
f_4	Brown (30D)	0.5	0	1	2.0E+05
f_5	Ackley (2D)	0.5	0	1	2.0E+05
f_6	Rosenbrock (2D)	0.5	0	1	2.0E+05
f_7	Rastrigin (30D)	0.5	0	1	2.0E+05
f_8	Griewank (30D)	0.5	0	1	2.0E+05
f_9	Five-uneven-peak trap (1D)	0.01	200.0	2	5.0E+04
f_{10}	Equal Maxima (1D)	0.01	1.0	5	5.0E+04
f_{11}	Uneven Decreasing Maxima (1D)	0.01	1.0	1	5.0E+04
f_{12}	Himmelblau (2D)	0.01	200.0	4	5.0E+04
f_{13}	Six-hump Camel Back (2D)	0.5	1.031628453	2	5.0E+04
f_{14}	Shubert (2D)	0.5	186.7309088	18	2.0E+05
f_{15}	Vincent (2D)	0.2	1.0	36	2.0E+05
f_{14}	Shubert (3D)	0.5	2709.093505	81	4.0E+05
f_{15}	Vincent (3D)	0.2	1.0	216	4.0E+05
f_{16}	Modified Rastrigin (2D)	0.01	-2.0	12	2.0E+05

4.2.1. Genetic Parameter Tuning for EVLNN

Generic parameter settings can significantly impact the performance of EAs, and EVLNN is not an exception and must be tuned. The genetic parameters for EVLNN to be tuned are

the population size N_P, the maximum number of generations G_{max}, the number of species N_S, the intra-species crossover percentage XO_p, the inter-species crossover probability XS_p, the mutation percentage MU_p, the mutation value range MU_r, the mutation matrix probability MX_p, and the elitism percentage EL_p. A summary of the genetic parameters is shown in Table 4.2.

Table 4.2 The EVLNN genetic parameters and their description.

EVLNN Genetic Parameters	Description
N_P	Population size
G_{max}	Max generation
N_s	Number of species
XO_p	Intra-species crossover percentage
XS_p	Intra-species crossover percentage
MU_p	Mutation percentage
MX_p	Mutation Matrix Probability
MU_r	Mutation value range
EL_p	Elitism percentage

Generally, genetic parameters adopt different values under given environments. To avoid ad-hoc speculation of these parameter values with no guideline on what values might work better, fixing them requires a study of the possible effects of these parameters and their interaction with the quality of the solution. Therefore, a set of experiments is designed to establish the values of these genetic parameters. Given the fixed *MaxFE* budget and the number of global optima G_{opt} of a function under test, the process includes first establishing the values of N_p and $G_{max,}$ then proceeding to determine XO_p and MU_p, and, finally, the values of XS_p, MU_r, and MX_p.

A) Tuning of N_p and G_{max}

The parameters N_p and G_{max} are the first values to be determined. For three different values of *MaxFE*={5.0E+04, 2.0E+05, 4.0E+05} [202], the respective N_p and G_{max} are different. The most common value pairs for N_p and G_{max} used during the experiments are,

$$\{N_p, G_{max}\} = \begin{cases} \{125, 400\}, \{100, 500\} & for\ MaxFE = 5.0\text{E} + 04 \\ \{500, 400\}, \{400, 500\} & for\ MaxFE = 2.0\text{E} + 05 \\ \{1000, 400\}, \{800, 500\} & for\ MaxFE = 4.0\text{E} + 05 \end{cases} \quad (4.4)$$

As for N_s, the value is set using the equation,

$$N_s = x * G_{opt} \quad (4.5)$$

where $1.5 \leq x \leq 4.0$ and G_{opt} is the number of optima. This range is determined empirically, with the value $x = 1.5$ being the most consistently used. However, if the search landscape has a low number of optima, the value for x is generally higher.

B) Tuning of XO_p and MU_p and Result Analysis

XO_p and MU_p are set to 80% and 20%, respectively, to provide a stable response. XS_p, MU_r, and MX_p are then tuned collectively to provide an adequate balance in explorative and exploitative search. These three parameters are the most sensitive as slight variations in their values could lead to very different results.

Table 4.3 presents three parameter sets used in EVLNN to optimize f_{15}, Shubert (2D) function, as an example to explain the interactions between the parameters and the effects of the responses. Parameter set 2 yielded better PR results at higher accuracy of ε = 1.0E-03 to ε = 1.0E-05, parameter set 3 yielded better PR results at lower accuracy of ε = 1.0E-01 to ε = 1.0E-02. Parameter set 1 was the worst performer, all other parameters being held constant. Function f_{15} is thus fine-tuned with, $N_p = 400$, $G_{max} = 500$, $N_s = 27$ (see Equation 4.5, where x is chosen as 1.5). This configuration provides an average of 15 individuals per species, given by, $\frac{N_p}{N_s} = \frac{400}{27} \approx 15$. Parameter set 2 is therefore chosen.

Table 4.3 EVLNN experiment for Shubert (2D) function for *MaxFE*=2.0E+05. N_p, $N_{s,}$ and G_{max} yielded different Peak Ratio (PR) results over a range of accuracy 1.0E-01 to 1.0E-0.5 for 50 runs. The best values for each accuracy level are in bold.

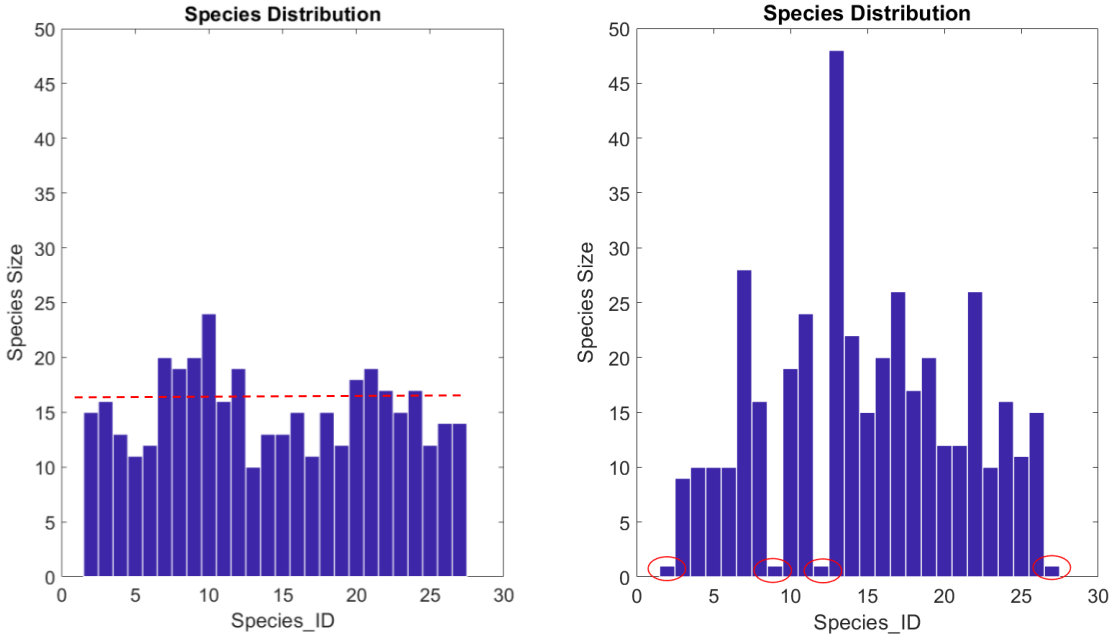
Parameters		N_p	G_{max}	N_s	G_{opt}	$\frac{N_s}{G_{opt}}$	$\frac{N_p}{N_s}$	**PR**	ε
Experiments	Set 1	250	800	18	18	1	~14	0.372	1.0E-01
								0.326	1.0E-02
								0.020	1.0E-03
								0.000	1.0E-04
								0.000	1.0E-05
	Set 2	400	500	27	18	1.6	~15	0.504	1.0E-01
								0.504	1.0E-02
								0.498	1.0E-03
								0.450	1.0E-04
								0.276	1.0E-05
	Set 3	400	500	36	18	2	~11	**0.778**	1.0E-01
								0.648	1.0E-02
								0.213	1.0E-03
								0.042	1.0E-04
								0.008	1.0E-05

C) Tuning of XS_p, $MX_{p,}$ and MU_r and Result Analysis

The values of XS_p, $MX_{p,}$ and MU_r are subsequently determined for the Shubert (2D) function with *MaxFE*=2.0E+0. It was observed that a stable population was produced when XS_p was set to between 1% to 2%. If XS_p is set too high, inter-species recombination activities increase, reducing species size and eventually leading to extinction. Depending on the application, this outcome may not be desirable.

This phenomenon can be illustrated with two separate populations configured with different XS_p settings, at 1.5% and 4%, respectively, for the Shubert (2D) function evaluation. After 500 generations, their respective species distributions are shown in Figure 4.1(a-b). For the population where XS_p is set to 4%, it is observed that four species have had their species sizes reduced to 1, namely *Species_2*, *Species_9*, *Species_12,* and *Species_27* (see Figure 4.1b highlighted by red circles). A unity species size means the lone individual would not participate in recombination. Mutation only would significantly reduce the effectiveness of explorative search. Contrast this to the population where XS_p is set to 1.5%, and species size remained an average of about 15 (see Figure 4.1a highlighted by a red dotted line).

This number remained relatively stable from the beginning. The experiment found evidence that a low value of XS_p helps maintain the species population. This finding is significant as the tuning of XS_p can determine EVLNN's search behavior. For solving multimodal and multi-optima problems, the XS_p value is recommended to remain small.



(a) (b)

Figure 4.1(a-b) Species distribution and XS_p values at the end of 500 generations for the evaluation of the Shubert (2D) function. In (a), XS_p set to 1.5% resulted in species with an average size of 15 holding stable from the start. In (b), XS_p is set higher to 4% resulting in some species (highlighted in red circle) having a species size of 1.

The value of MX_p determines the number of genes in the change matrix participating in the mutation process. This parameter can take a broader range of values from 5% to 60%. Depending on the problem to be solved, the value of MX_p can be adjusted. For example, a higher MX_p value will expand the search in the vicinity of a basin, and a lower value will deepen the search toward a basin. Therefore, if the value is set too high, convergence may not happen; likewise, solutions may not be optimal if the values are too low.

In conjunction with MX_p, the parameter MU_r determines the values to be added to the genes of the chromosome matrix in the selected mutant. The range of values for MU_r is -0.1 ≤

$MU_r \leq 0.1$ for small incremental changes. A narrower range of $-0.02 \leq MU_r \leq 0.02$ is set for a small search range. A summary of the EVLNN algorithm's parameter settings derived from the Shubert (2D) function evaluation is described in Table 4.4. The parameter tuning of EVLNN is problem-specific and needs to be fine-tuned for different function optimization. Nonetheless, the empirical rule-of-thumb derived from Table 4.4 provides a framework for EVLNN parameter tuning.

Table 4.4 Lesson learned on the EVLNN parameters and settings derived from evaluating the Shubert (2D) function.

EVLNN Parameters	**Values**	**Empirical Lesson Learned**
N_P	100, 400, 800	Higher N_p values for search landscape with a high number of global optima.
G_{max}	400, 500	A higher value allows sufficient generations for species to evolve and converge to better solutions.
N_s	$x * G_{opt}$	G_{opt} is the number of global optima and $1.5 \leq x \leq 4.0$ with a higher value of x for multimodal high-dimensional functions. For low G_{opt}, x is set to a higher value.
XO_p	80%	This percentage provides a stable evolutionary process.
XS_p	1% to 2%	Higher XS_p for multimodal functions with multiple global optima. With higher N_s, XS_p needs to reduce accordingly to avoid species extinction.
MU_p	20%	This percentage provides a stable evolutionary process.
MX_p	5% to 60%	Higher MX_p to avoid stagnation and local minima trap in challenging multimodal high dimensionality landscape.
MU_r	-0.1 to 0.1	A narrower range is selected if the search space is small.
EL_p	5%	This percentage provides a stable evolutionary process.

4.3. Evaluation using Benchmark Test Functions

Many benchmark functions for numerical optimization have been reported in the literature. However, there is no standard list or set of benchmark functions to validate and compare the performance of optimization algorithms [203]. In order to evaluate EVLNN's search

abilities, a test suite is compiled to expose EVLNN to a wide variety of problems such as unimodal, multimodal, separable, non-separable, and multi-dimensional problems. The problem of function optimization is essentially finding the function's minima (or maxima).

Unimodal test functions have only one global solution in the search landscape, whereas multimodal test functions have more than one local optimum in the search space. Single model optimization algorithms are the basis of more complex optimization algorithms such as multi-objective optimization and constrained optimization [211]. If the landscape is unimodal, the efficiency of EVLNN for local search can be tested.

Multimodal test functions have multiple global optima or one global optima with many local minima in the search landscape. They are designed to test the ability of an algorithm to avoid local minima, often known as “traps.” If an algorithm encounters many minima in a search space, it could be trapped in one of them. Once trapped in the deceptive minima, the search algorithm becomes significantly hampered and may not recover by directing the search away from these deceptive minima to the proper optimal solution. The landscape modality is essential to test the speciation characteristic of EVLNN and the robustness of EVLNN’s crossover and mutation strategies for global search. The following equation shows a general optimization problem,

$$\underset{x}{\text{minimize}}\, f(x) \tag{4.6}$$

where the optima may be a single value or a set of values which $x^* \in D$ for all feasible points D in a search space.

Separable functions have variables that are independent of each other. Hence each variable can be independently optimized. On the contrary, non-separable functions have variables that are interrelated or are not independent. Boyer et al. [212] described that a function of p variable is separable if it can be written as a sum of p functions of just one variable, that is,

$$f(x_1, x_2, \dots, x_p) = \sum_{i=1}^{p} f_i(x_i) \tag{4.7}$$

Separable functions are relatively easier to solve than their non-separable counterpart. Non-separable functions are problems where a portion of the variables interact amongst themselves [213]. Separability in the function landscape is thus considered for testing if EVLNN is capable of dealing with complicated fitness landscapes with mixed variable separability.

Dimensionality is an important issue in a search landscape. It refers to the number of variables in the function. The search space increases exponentially as the number of variables or dimensions increases [214]. Test functions with varying dimensionality could present problems in such a way that contains few global minima but closely locate them to the local minima. Such problems are considered for effectively testing EVLNN's exploration characteristics and performance around narrow basins in the search space.

There is no fixed number of benchmark functions to test the reliability and performance of optimization algorithms. Most papers varied from a few up to about 20. Typically, a diverse test suite is selected to provide a good representation of the variety of unimodal, multimodal, separable, non-separable, and multi-dimensional problems. Tables 4.5 and 4.6 presents the test functions employed to evaluate the EVLNN algorithm. These are well-known optimization benchmark functions [204], [211] – [213] widely used to test out new optimization algorithms.

Table 4.5 Benchmark test functions to evaluate the EVLNN algorithm. *d* is the number of dimensions, and *Range* is the input domain where the function is evaluated and f_{min} is the global minimum.

Test function	Description	Variable	*d*	Range	f_{min}
Unimodal, low dimensionality					
$f_1(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi x_2) + 0.7$	Bohachevsky N.1	Separable	2	[-10 10]	0
$f_2(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	Booth	Non-separable	2	[-2π 2π]	0
Unimodal, high dimensionality					
$f_3(x) = \sum_{i=1}^{d} x_i^2$	Sphere	Separable	30	[-5.12 5.12]	0
$f_4(x) = \sum_{i=1}^{d-1} (x_i^2)^{(x_{i+1}^2+1)} (x_{i+1}^2)^{(x_i^2+1)}$	Brown	Non-Separable	30	[-1 4]	0
Multimodal, low dimensionality					
$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)\right) + 20 + exp(1)$	Ackley	Separable	2	[-35 35]	0
$f_6(x) = \sum_{i=1}^{d} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	Rosenbrock	Non-separable	2	[-5 10]	0
Multimodal, high dimensionality					
$f_7(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)]$	Rastrigin	Separable	30	[-5.12 5.12]	0
$f_8(x) = f(x) = 1 + \sum_{i=1}^{d} \frac{x_i{}^2}{4000} - \prod \cos\left(\frac{x_i}{\sqrt{i}}\right)$	Griewank	Non-separable	30	[-100 100]	0

Figure 4.2 illustrates each function's different modality, dimensionality, and separability aspects.

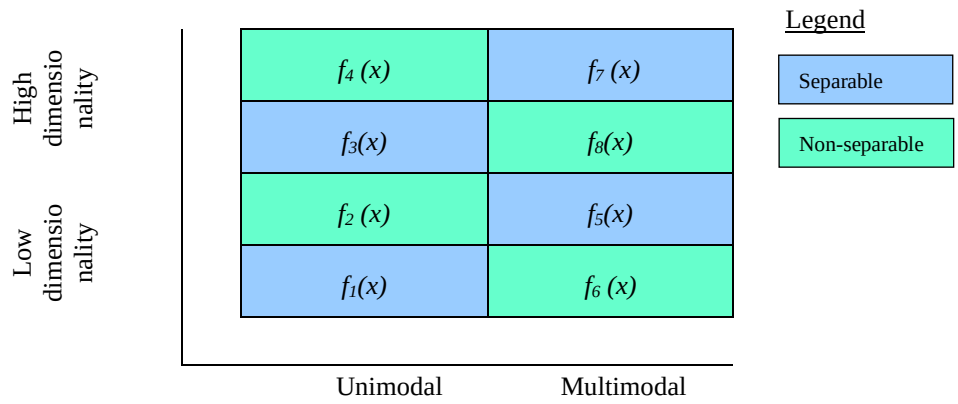


Figure 4.2 A representative spread of test functions to evaluate EVLNN's search capability.

The Bohachevsky N.1-2D (f_1) function is shaped like a bowl and evaluated on $x_i \in$ [-10 10], for all i = 1, 2. The Booth-2D (f_2) function has non-separable variables and is evaluated on $x_i \in [-2\pi\ 2\pi]$, for all i = 1, 2. The Sphere-30D (f_3) function is evaluated on $x_i \in$ [-5.12 5.12], for all i = 1, 2, …, 30 with a high dimensionality of 30 variables. The Brown-30D (f_4) function is characterized by fine seesaw edges in the vicinity of the minima. The almost fractal landscape is extremely challenging to optimize by any global or local optimization methods. The f_4 function is evaluated on $x_i \in$ [-1 4], for all i = 1, 2, …, 30 with a high dimensionality of 30 variables. The Ackley-2D (f_5) function consists of a global minima with many local minima evaluated on $x_i \in$ [-35 35], for all i = 1, 2. The landscape of f_5 has an almost flat outer surface with a large center aperture that plunges to 0 at the minima position. This landscape presents a risk for optimization algorithms to be trapped in one of its many local minima. The Rosenbrock-2D (f_6) function is evaluated on $x_i \in [-2\pi\ 2\pi]$, for all i = 1, 2. The function is characterized by its valley shape landscape where the global minima lie in a narrow, parabolic valley, and convergence to the minima is difficult [215]. The Rastrigin-30D (f_7) function is extremely multimodal. The many local minima are spaced evenly, forming a highly deceptive landscape for optimization algorithms. The f_7 function is evaluated on $x_i \in$ [-5.12 5.12], for all i = 1, 2, …, 30 with a high dimensionality of 30 variables. The Griewank-30D (f_8) function is characterized by many regularly distributed local minima, posing a risk to the optimization algorithms trapped in the local optima. The f_8 function is evaluated on $x_i \in$ [-100 100], for all i = 1, 2, …, 30 with a high dimensionality of 30 variables.

Single objective problems can be transformed into the dynamic, niching composition as real-world problems with a complex optimization environment [211]. Table 4.6 is a list of test functions recommended by the CEC in 2013 and 2015 to test new algorithms [207] involving multiple satisfactory solutions and a large number of search variables to mimic the real-world environment. These benchmark test functions are employed in the experiment to test the search capability of EVLNN. Besides assessing the performance of EVLNN, these benchmark functions also act as a common platform for comparing EAs that incorporate niching methods to locate multiple optima, through the definition of

standard performance metrics, the maximum number of function evaluations, and accuracy levels.

Functions $f_9 - f_{11}$ are one dimensional (1D) functions. The Five-Uneven-Peak Trap-1D (f_9) function is evaluated on $x \in [0\ 30]$, characterized by three local optima and two global optima. The global optima are located at the search space's margin, making the function difficult to solve. Both the Equal Maxima-1D (f_{10}) and Uneven Decreasing Maxima-1D (f_{11}) functions are evaluated on $x \in [0\ 1]$. The Equal Maxima-1D (f_{10}) function is characterized by five evenly distributed global optima and is designed to test the stability of the niching method. The Uneven Decreasing Maxima-1D (f_{11}) functions are characterized by one global optima and four local optima. The 1D test functions are important in studying the search characteristics of EVLNN. They can also be easily visualized in a 2D plot.

Table 4.6 CEC 2013 and 2015 test functions for evaluating the EVLNN algorithm. *d* is the number of dimensions, and *Range* is the input domain where the function is evaluated and f_{min} is the global minimum.

CEC Benchmark Test Functions					
Test function	**Description**	**Characteristic**	***d***	**Range**	f_{min}
$f_9(x)=\begin{cases} 80(2.5-x) & for\ 0 \le x \le 2.5, \\ 64(x-2.5) & for\ 2.5 \le x \le 5.0, \\ 64(7.5-x) & for\ 5.0 \le x \le 7.5, \\ 28(x-7.5) & for\ 7.5 \le x \le 12.5, \\ 28(17.5-x) & for\ 12.5 \le x \le 17.5, \\ 32(x-17.5) & for\ 17.5 \le x \le 22.5, \\ 32(27.5-x) & for\ 22.5 \le x \le 27.5, \\ 80(x-27.5) & for\ 27.5 \le x \le 30. \end{cases}$	Five-Uneven-Peak Trap	2 global optima, 3 local optima	1	[0 30]	200
$f_{10}(x) = sin^6(5\pi x)$	Equal Maxima	5 global optima, 0 local optima	1	[0 1]	1
$f_{11}(x) = exp\left(-2\log(2)\left(\frac{x-0.08}{0.854}\right)^2\right) sin^6\left(5\pi\left(x^{\frac{3}{4}}-0.05\right)\right)$	Uneven Decreasing Maxima	1 global optima, 4 local optima	1	[0 1]	1
$f_{12}(x) = 200 - ({x_1}^2 + x_2 - 11)^2 + (x_1 + {x_2}^2 - 7)^2$	Himmelblau	4 global optima, 0 local optima	2	[-6 6]	200
$f_{13}(x) = -4\left[\left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (4x_2^2 - 4)x_2^2\right]$	Six-hump Camel	2 global optima, 2 local optima	2	[-5 5]	-1.0316
$f_{14}(x) = -\prod_{i=1}^{d}\sum_{j=1}^{5} jcos[(j+1)x_i + j]$	Shubert	$d.3^d$ global optima, 760 local optima	2 3	[-10 10]^d	-186.7309 2709.094
$f_{15}(x) = \frac{1}{d}\sum_{i=1}^{d} \sin(10\log(x_i))$	Vincent	6^d global optima, 0 local optima	2 3	[0.25 10]^d	1
$f_{16}(x) = -\sum_{i=1}^{d}(10 + 9\cos(2\pi k_i x_i)))$	Modified Rastrigin	$\prod_{i=1}^{d} k_i$ optima, k_i=1 for i=1-3, 0 local optima	2	[0 1]^d	-2.0

The Himmelblau-2D (f_{12}) function has four equal optima, with two closer to each other than the other 2. Its landscape has 0 local optima and is evaluated on $x_i \in [-6\ 6]$, for all $i = 1, 2$. The Six-hump Camel-2D (f_{13}) function is characterized by 2 global optima and 2 local optima, is evaluated on $x_i \in [-6\ 6]$, for all $i = 1, 2$. The inverted Shubert-2D/3D (f_{14}) and the inverted Vincent-2D/3D (f_{15}) functions are difficult and intriguing. The inverted Shubert-2D (f_{14}) function is characterized by 760 local optima and 18 global optima in 9 pairs where each pair is very close to the other, but the distance between each pair is much greater. As the dimensionality d increases, the number of global optima increases by $d.3^d$. The inverted Vincent-2D (f_{15}) function has 6^d global optima, but unlike the even distances between the global optima in f_{14}, in the Vincent-2D (f_{15}) function, the global optima have vastly different spacing between them. In addition to that, the Vincent function has no local optima. The Modified Rastrigin-2D (f_{16}) function consists of 0 local optima and $\prod_{i=1}^{d} k_i$ global optima, where k_1=3, and k_2=4, for d=2, the number of global optima is 12.

The landscapes of functions $f_{12} - f_{16}$ with multiple global optima and zero or more local or deceptive optima are essential in testing the performance of EVLNN, particularly its speciation ability in the global search for multiple optimas.

4.3.1. Function Evaluations for f_1 to f_8

The results from experiments using EVLNN for the benchmark test functions f_1 to f_8 given in Table 4.5 are presented in Table 4.7. The peak locations found by EVLNN at various iterations during the function evaluations of 2D landscapes f_1, f_2, f_5, and f_6 are shown in Figure 4.3. Figures 4.4 and 4.5 display the convergence characteristics of EVLNN for functions f_1 to f_8.

From Table 4.7, EVLNN achieved 100% success for PR and SR for Bohachevsky N.1-1D (f_1) and Booth-2D (f_2) functions for all ε. It also performed well for the Ackley-2D (f_5) and Rosenbrock-2D (f_6) functions, with an overall average of 81.2% and 93.2%, respectively, for locating the peaks for all ε. Figure 4.3 presents the contour plots of the benchmark functions Bohachevsky N.1-2D (f_1), Booth-2D (f_2), Ackley-2D (f_5), and Rosenbrock-2D (f_6). The search pattern of EVLNN is illustrated by the red dots whose distribution is traced

at the 1st, 10th, 100th, 300th, and 500th iterations. The figure shows that EVLNN had converged at the optima of these benchmark functions.

For the Sphere-30D (f_3) and Brown-30D (f_4) functions, EVLNN had 100% success for PR and SR at accuracy ε = 1.0E-01 and ε = 1.0E-02. However, it failed to locate the peaks where $\varepsilon \leq$ 1.0E-03. For the Rastrigin-30D (f_7) function, EVLNN had 100% success for PR and SR at ε = 1.0E-01 but could not locate any peaks beyond the next accuracy level. For the Griewank-30D (f_8) function, EVLNN achieved a PR of 32% at ε = 1.0E-01, dropping by an average of 7% at each accuracy level to 4% at ε = 1.0E-05. Benchmark functions with high dimensionality caused the size of the chromosome matrix to increase, leading to a higher number of gene elements to be handled. The crossover process thus became more complex, making EVLNN less efficient and capable of reaching an optimal result.

Table 4.7 Peak ratios and success rates of EVLNN for test functions f_1 to f_8.

Accuracy level, ε	**f_1 Bohachevsky N.1-2D**		**f_2 Booth-2D**		**f_3 Sphere-30D**		**f_4 Brown-30D**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
1.0E-01	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.0E-02	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.0E-03	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000
1.0E-04	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000
1.0E-05	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000
Accuracy level, ε	**f_5 Ackley-2D**		**f_6 Rosenbrock-2D**		**f_7 Rastrigin-30D**		**f_8 Griewank-30D**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
1.0E-01	1.000	1.000	1.000	1.000	1.000	1.000	0.320	0.000
1.0E-02	1.000	1.000	0.980	0.980	0.000	0.000	0.220	0.000
1.0E-03	1.000	1.000	0.940	0.940	0.000	0.000	0.140	0.000
1.0E-04	1.000	1.000	0.880	0.880	0.000	0.000	0.080	0.000
1.0E-05	0.060	0.060	0.860	0.860	0.000	0.000	0.040	0.000

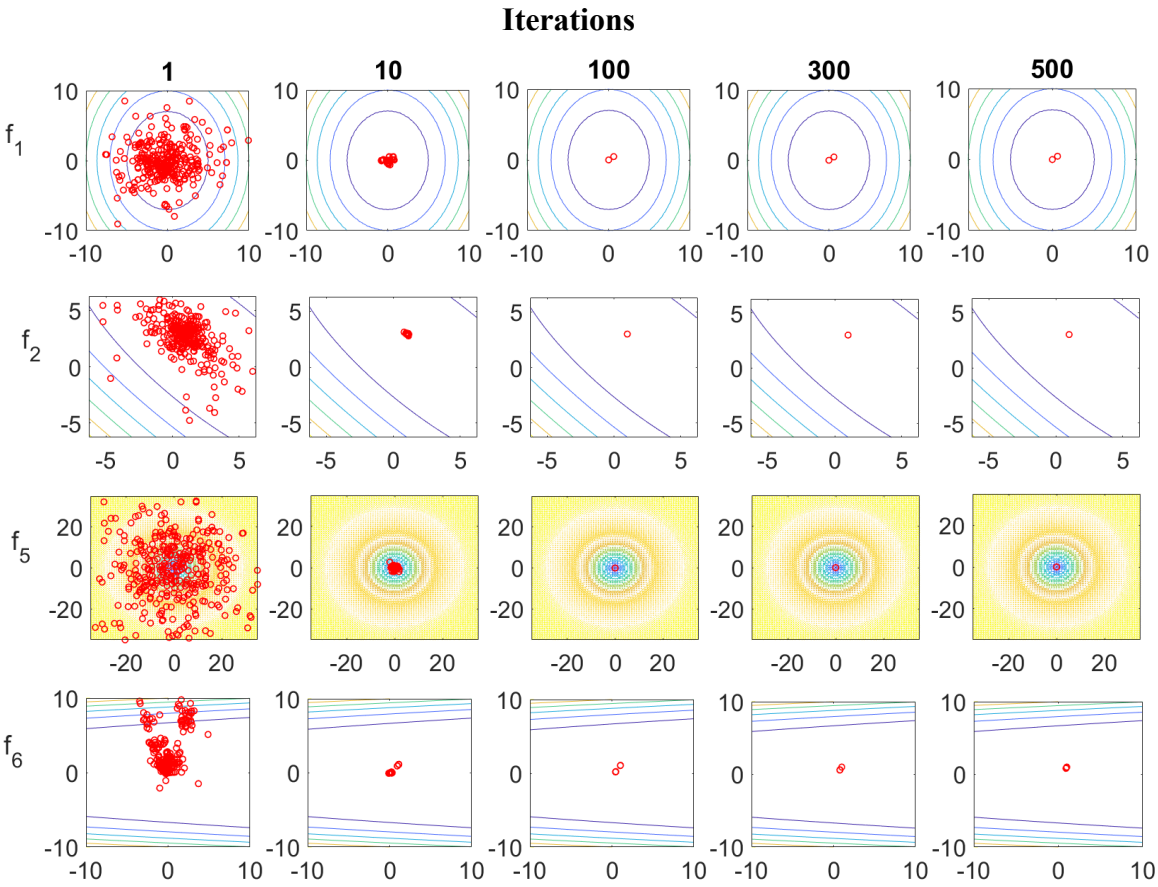


Figure 4.3 Red dots illustrate the search patterns of EVLNN at the 1st, 10th, 100th, 300th, and 500th iterations of the function evaluations on the 2D landscapes of Bohachevsky N.1-2D (f_1), Booth-2D (f_2), Ackley-2D (f_5), and Rosenbrock-2D (f_6), respectively.

Figures 4.4 and 4.5 illustrate the algorithm's convergence characteristics for functions f_1 to f_5 and f_6 to f_8, respectively. Slower convergence is observed for high dimensionality functions like the Sphere-30D (f_3) (orange dotted plot) and Brown-30D (f_4) (purple plot). EVLNN took about 400 generations and 200 generations, respectively, to reach good solutions. For the Rastrigin-30D (f_7) function (cyan dotted plot) in Figure 4.5, EVLNN achieved an error value $\varepsilon < 0.05$ within the first 100 generations; however, beyond this value, it could not converge to better solutions shown by a prolonged period of 'stagnation' from generations 230 onwards.

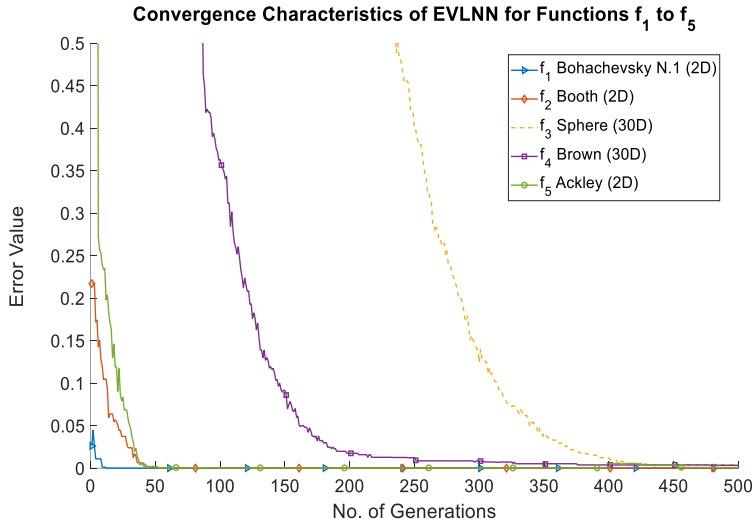


Figure 4.4 Convergence characteristics of EVLNN algorithm for f_1 to f_5.

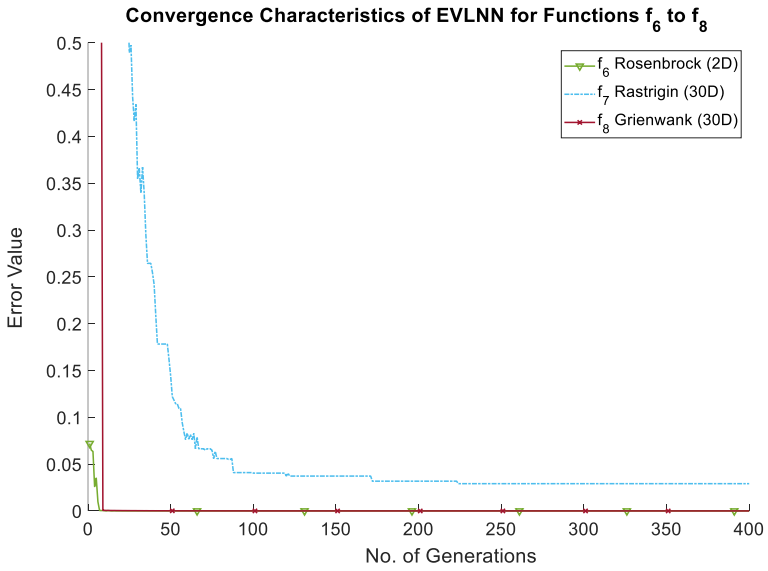


Figure 4.5 Convergence characteristics of EVLNN algorithm for f_6 to f_8.

Table 4.8 compares the PR and SR measurements between EVLNN and PSO, DE, and GA.

Table 4.8 Peak Ratios (PR) and Success Rates (SR) of EVLNN, PSO, DE, and GA.

f_1, Bohachevsky N.1-2D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-02	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-03	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-04	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-05	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_2, Booth-2D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-02	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-03	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-04	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-05	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_3, Sphere-30D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-02	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-03	0.000	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-04	0.000	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-05	0.000	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_4, Brown-30D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-02	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-03	0.000	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-04	0.000	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-05	0.000	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_5, Ackley-2D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-02	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-03	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-04	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-05	0.060	0.060	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
f_6, Rosenbrock-2D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-02	0.940	0.940	**1.000**	**1.000**	**1.000**	**1.000**	0.960	0.960
ε = 1.0E-03	0.940	0.940	**1.000**	**1.000**	**1.000**	**1.000**	0.920	0.920

ε = 1.0E-04	0.860	0.860	**1.000**	**1.000**	**1.000**	**1.000**	0.880	0.880
ε = 1.0E-05	0.820	0.820	**1.000**	**1.000**	**1.000**	**1.000**	0.860	0.860
f_7, Rastrigin-30D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	**1.000**	**1.000**	0.000	0.000	0.000	0.000	0.000	0.000
ε = 1.0E-02	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ε = 1.0E-03	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ε = 1.0E-04	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ε = 1.0E-05	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
f_8, Griewank-30D	EVLNN		PSO		DE		GA	
	PR	SR	PR	SR	PR	SR	PR	SR
ε = 1.0E-01	0.320	0.320	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ε = 1.0E-02	0.220	0.220	0.640	0.640	**1.000**	**1.000**	0.980	0.980
ε = 1.0E-03	0.140	0.140	0.280	0.280	**1.000**	**1.000**	0.920	0.920
ε = 1.0E-04	0.080	0.080	0.280	0.280	**1.000**	**1.000**	0.920	0.920
ε = 1.0E-05	0.040	0.040	0.280	0.280	**1.000**	**1.000**	0.920	0.920

From Table 4.8, it is seen that the performance of EVLNN is comparable to PSO, DE, and GA for Bohachevsky N.1-1D (f_1) and Booth-2D (f_2) functions, where PR and SR are both at 100% across all ε. For the Rosenbrock-2D (f_6) function, PSO and DE performed well, with 100% of the peaks found at all ε. The performance of EVLNN in these functions is comparable to GA, with an average PR of 91.2% and 92.4%, respectively, across all ε. For the Sphere-30D (f_3), Brown-30D (f_4), and Ackley-2D (f_5) functions, the peaks were found at all ε by the PSO, DE, and GA.

However, this was not so for EVLNN, which only managed to find all peaks for Sphere-30D (f_3) and Brown-30D (f_4) at ε = 1.0E-02. For the Ackley-2D (f_5) function, EVLNN has achieved an accuracy at ε = 1.0E-04, whereas PSO, DE, and GA had a higher accuracy at ε = 1.0E-05. It may be possible that the landscape for f_5, with its many local minima around the global minima, had trapped the EVLNN species preventing them from finding better solutions. For Rosenbrock-2D (f_6) function, EVLNN's performance is comparable with GA but loses out to PSO and DE. EVLNN's less desirable performance in high dimensionality problems, particularly for functions f_3 and f_4, is due to increased crossover complexity in handling larger chromosome matrix sizes. As dimensionality increases, the chromosome matrix size increases row-wise. This scenario presents a limitation in the search efficiency of EVLNN during crossover for real parameter optimization. Nonetheless, EVLNN is a

neural architecture search algorithm that aims to locate optimal parsimonious structures that could accurately predict the output of a time-series dataset. Hence the results should not impact its real-world application in structural optimization for predictions.

Nonetheless, a bright spark for EVLNN is in the Rastrigin-30D (f_7) function evaluation. It also has a high dimensionality of 30 variables. EVLNN was the only algorithm that found all peaks for the f_7 function at accuracy level $\varepsilon = 1.0\text{E-}01$, whereas no peaks were found by PSO, DE, and GA. It is worth discussing the interesting facts revealed by the superior results of EVLNN. The main difference in the Rastrigin-30D (f_7) function compared to the other high dimensional functions such as f_3 and f_4, where EVLNN did not fair well, is that f_7 is hugely multimodal, with many local minima. However, the location of the minima is regularly distributed, forming a highly deceptive landscape for optimization algorithms. It is possible that the speciation capability of EVLNN helped navigate the search through this type of landscape. A similar conclusion was reached for the Shubert-2D (f_{14}) test function, which has 18 global optima in 9 pairs with many local optima. EVLNN was the only algorithm that could locate the function peak at $\varepsilon = 1.0\text{E-}05$. Details of the Shubert-2D (f_{14}) result are discussed in section 4.4.1. In the real world, the strength of EVLNN's search algorithm could be applied to detect the source of hotspots (global optima) in data centers where there are multiple regions of hot spots (local optima) that are close to one another (multimodal).

For Griewank-30D (f_8) function, DE has the best performance, with 100% success in finding the peaks (PR=1.000) for all runs (SR=1.000) and at all ε. Next are GA, PSO, and EVLNN, which achieved an average PR of 94.8%, 49.6%, and 14.4%, respectively, at all ε. As EVLNN is designed for neural architectural search, its application as a function optimizer had a limited overall performance. However, EVLNN had located the peaks of functions f_1 to f_8, demonstrating that the algorithm generalizes better than PSO, DE, and GA for this set of test functions.

Figure 4.6 compares the convergence characteristics of EVLNN with PSO, DE, and GA for this function. The plots show that EVLNN converged to a lower error, whereas PSO, DE, and GA were stagnant at a higher error for f_7. The trend from these plots suggests that PSO, DE, and GA would not converge to reasonable solutions even if given a higher number of iterations.

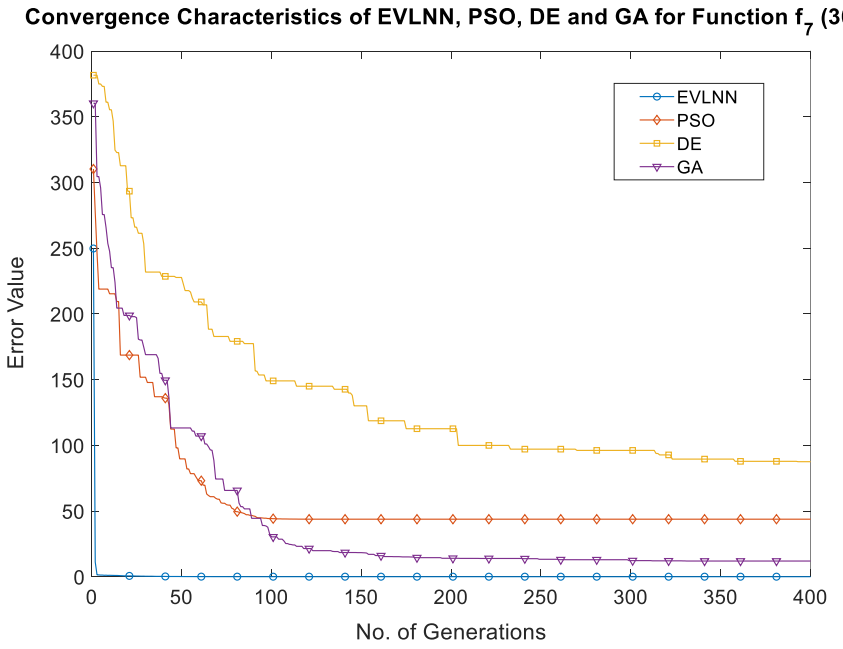


Figure 4.6 The convergence characteristics of EVLNN, PSO, DE, and GA algorithms for the Rastrigin-30D (f_7) function.

While EVLNN exhibited some weaknesses in high-dimensional problems, it still outperformed PSO, DE, and GA for the Rastrigin-30D (f_7) function by successfully locating the peak at ε = 1.0E-01, where these algorithms had failed to do so. EVLNN, as a function optimizer for f_7, had performed well specific to this function landscape.

4.4. Comparative Analysis of other State-of-the-Art EAs

The performance of the EVLNN algorithm was evaluated using test functions for (i) single optima, multimodal problems, and (ii) multi-optima, multimodal problems in Tables 4.5 and 4.6, respectively. The results for (i) are compared to modern population-based meta-heuristic algorithms, namely PSO and DE, and the classic GA is used as a reference. These algorithms generally converge to a single solution and are well-suited compared to

EVLNN's search capability for solving single optima multimodal problems. The results for (ii) are compared to 21 state-of-the-art EAs from the CEC 2013 and CEC 2015 competitions. The EAs are presented in Table 4.9.

Table 4.9 State-of-the-art EAs in the CEC 2013 and CEC 2015 competitions.

	Niching Algorithms	Description	Year of CEC Competition
1.	DE/nrand/1/bin [216]	Classic DE algorithm with dynamic and clustering 1	2013
2.	DE/nrand/2/bin [216]	Classic DE algorithm with dynamic and clustering 2	2013
3.	Crowding DE/rand/1/bin [217]	Classic DE algorithm extended with the crowding scheme	2013
4.	NMMSO$^{\&+}$ [218]	Niching Migratory Multi-Swarm Optimizer algorithm	2015
5.	N-VMO [219]	Niching Variable Mesh Optimization algorithm	2013
6.	dADE/nrand/1/bin$^{\#}$ [220]	Dynamic Archive Niching Differential Evolution Algorithm 1	2013
7.	dADE/nrand/2/bin [220]	Dynamic Archive Niching Differential Evolution Algorithm 2	2013
8.	NEA1 [216]	Niching Evolutionary Algorithm 1	2013
9.	NEA2$^{\#\&+}$ [216]	Niching Evolutionary Algorithm 2	2013
10.	DECG [221]	DE algorithm using crowding and gradient descent	2013
11.	DELG [221]	DE algorithm using local selection and gradient descent	2013
12.	DELS_ajitter [221]	DE algorithm using local selection and ajitter global mutation	2013
13.	CMA-ES$^{\#}$ [222]	Covariance matrix adaptation evolution strategy	2013
14.	iPOP-CMA-ES [223]	CMA-ES with increasing population size	2013
15.	ANSGAII [224]	NSGAII with variable-space niching	2013
16.	PNA-NSGAII [224]	Parameterless niching assisted NSGAII	2013
17.	$LSEA_{EA}$ [225]	Localised search evolutionary algorithm using EAs for local search	2015
18.	$LSEA_{GP}{}^{\&+}$ [226]	LSEA using Gaussian process as its local search mechanism	2015
19.	ALNM [227]	Active Learning Based Niching Method	2015
20.	MEA [228]	Multinational Evolutionary Algorithm	2015
21.	MSSPSO [229]	Multi-Sub-Swarm Particle Swarm Optimisation algorithm	2015

$^{\#}$*Top three winners of CEC 2013*
$^{\&}$*Top three winners of CEC 2015*
$^{+}$*Top three winners of the overall CEC 2013 and CEC 2015*

According to the competition ranking [204] [205], the top-performing optimizer in the CEC 2013 competition is the NEA2 algorithm proposed by Preuss [216]. The NEA2 hybridizes the CMA-ES algorithm with a control strategy that employs a simple heuristic to detect different search spaces and avoid multiple local searches in the same area. The dADE/nrand/1 algorithm proposed by Epitropakis et al. [220] came in second. The algorithm enhances the classic DE mutation strategies with local information from the current population to efficiently locate and maintain global optima. The classic CMA-ES

[222] was ranked third in the same competition. According to [216], the CMA-ES is likely the best method in Evolutionary Computation (EC) field. The algorithm uses a 'restart' strategy to detect stagnation by setting up a new population in a different area in the search space.

The top-performing optimizer in the CEC 2015 competition is the NMMSO algorithm proposed by Fieldsend [218]. The sub-swarms in NMMSO provided the niching mechanism to optimize separate local modes dynamically. The NMMSO was also the winner averaged over the two competitions, followed by the NEA2, $LSEA_{GP}$ [226], and $LSEA_{EA}$ [225]. The $LSEA_{GP}$ employs a surrogate-based approach that uses the Gaussian correlation model, also known as a Gaussian Process, from the MATLAB Kriging toolbox [230] as a local surrogate model to automatically adapt to the number of niches depending on the characteristics of the landscape discovered. Distinct from the $LSEA_{GP}$, the $LSEA_{EA}$ presents exploitative hill-climbing EAs rather than the surrogate models to drive the local search [225]. The test results and comparative analysis are discussed in the later sections.

4.4.1. Function Evaluations for f_9 to f_{16}

Table 4.10 presents the PR and SR results achieved by EVLNN for the benchmark test functions listed in Table 4.6 evaluated for all five levels of accuracy. From Table 4.10, EVLNN has achieved 100% success in locating the peaks for the Uneven Decreasing Maxima-1D (f_{11}) and Six-hump Camel Back-2D (f_{13}) functions for all ε. It also attained an average PR of 79.4% and 84.3% for Equal Maxima-1D (f_{10}) and Himmelblau-2D (f_{12}) functions, respectively, for all ε. EVLNN has done moderately well for Five-Uneven-Peak Trap-1D (f_9), Shubert-2D (f_{14}), and Modified Rastrigin-2D (f_{16}), with an average PR at 45.1%, 44.6%, and 59.1%, respectively. However, EVLNN had performed poorly with Vincent-2D (f_{15}), Shubert-3D (f_{14}), and Vincent-3D (f_{15}), with an average PR of 37.3%, 1.24%, and 8.8%, respectively. A common characteristic of these functions, f_{15} (2D), f_{14} (3D), and f_{15} (3D), is that they have a high number of global optima, G_{opt} at 36, 81, and 216, respectively. Given that $G_{opt} >> 1$, with reference to Table 4.2 and Equation 4.5, N_s, the number of species will be large. At a fixed budget *MaxFE*, the average species size $\frac{N_p}{N_s}$

would be small. A small $\frac{N_p}{N_s}$ might not contain sufficient genetic variability within the EVLNN species to generate good solutions during the crossover and mutation process for that species.

Table 4.10 Peak ratios and success rates of EVLNN for test functions f_9 to f_{16}.

Accuracy level, ε	**f_9 Five-Uneven-Peak Trap-1D**		**f_{10} Equal Maxima-1D**		**f_{11} Uneven Decreasing Maxima-1D**		**f_{12} Himmelblau-2D**		**f_{13} Six-hump Camel Back-2D**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
1.0E-01	0.850	0.700	0.836	0.360	1.000	1.000	0.905	0.620	1.000	1.000
1.0E-02	0.830	0.660	0.836	0.360	1.000	1.000	0.905	0.620	1.000	1.000
1.0E-03	0.470	0.200	0.828	0.360	1.000	1.000	0.905	0.620	1.000	1.000
1.0E-04	0.100	0.000	0.788	0.300	1.000	1.000	0.885	0.540	1.000	1.000
1.0E-05	0.003	0.000	0.684	0.160	1.000	1.000	0.615	0.100	1.000	1.000
Accuracy level, ε	**f_{14} Shubert-2D**		**f_{15} Vincent-2D**		**f_{14} Shubert-3D**		**f_{15} Vincent-3D**		**f_{16} Modified Rastrigin-2D**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
1.0E-01	0.504	0.000	0.387	0.000	0.057	0.000	0.140	0.000	0.878	0.140
1.0E-02	0.504	0.000	0.387	0.000	0.005	0.000	0.101	0.000	0.872	0.140
1.0E-03	0.498	0.000	0.387	0.000	0.000	0.000	0.091	0.000	0.708	0.000
1.0E-04	0.450	0.000	0.380	0.000	0.000	0.000	0.071	0.000	0.373	0.000
1.0E-05	0.276	0.000	0.338	0.000	0.000	0.000	0.037	0.000	0.125	0.000

For function f_9 at accuracy ε = 1.0E-05, it is observed that EVLNN produced a weak performance with a PR of 0.3%. One possible explanation for EVLNN's low PR of 0.3% for function f_9 may be attributed to EVLNN's stochastic approach to solving large-scale optimization problems while ensuring generality.

Figures 4.7 and 4.8 examine the algorithm's search pattern on the contour map of functions f_9 to f_{13} and f_{14} to f_{16}, respectively. Figure 4.7 consists of 1D and 2D multimodal functions f_9 to f_{13}, and Figure 4.8 consists of scalable 2D multimodal functions f_{14} to f_{16}. The search patterns of EVLNN are illustrated by the red dots whose distribution is traced at the 1st, 100th, 200th, 400th, and 500th iterations of the function evaluations on these landscapes. These figures show most of the peak locations returned by EVLNN during the function by the 100th generation.

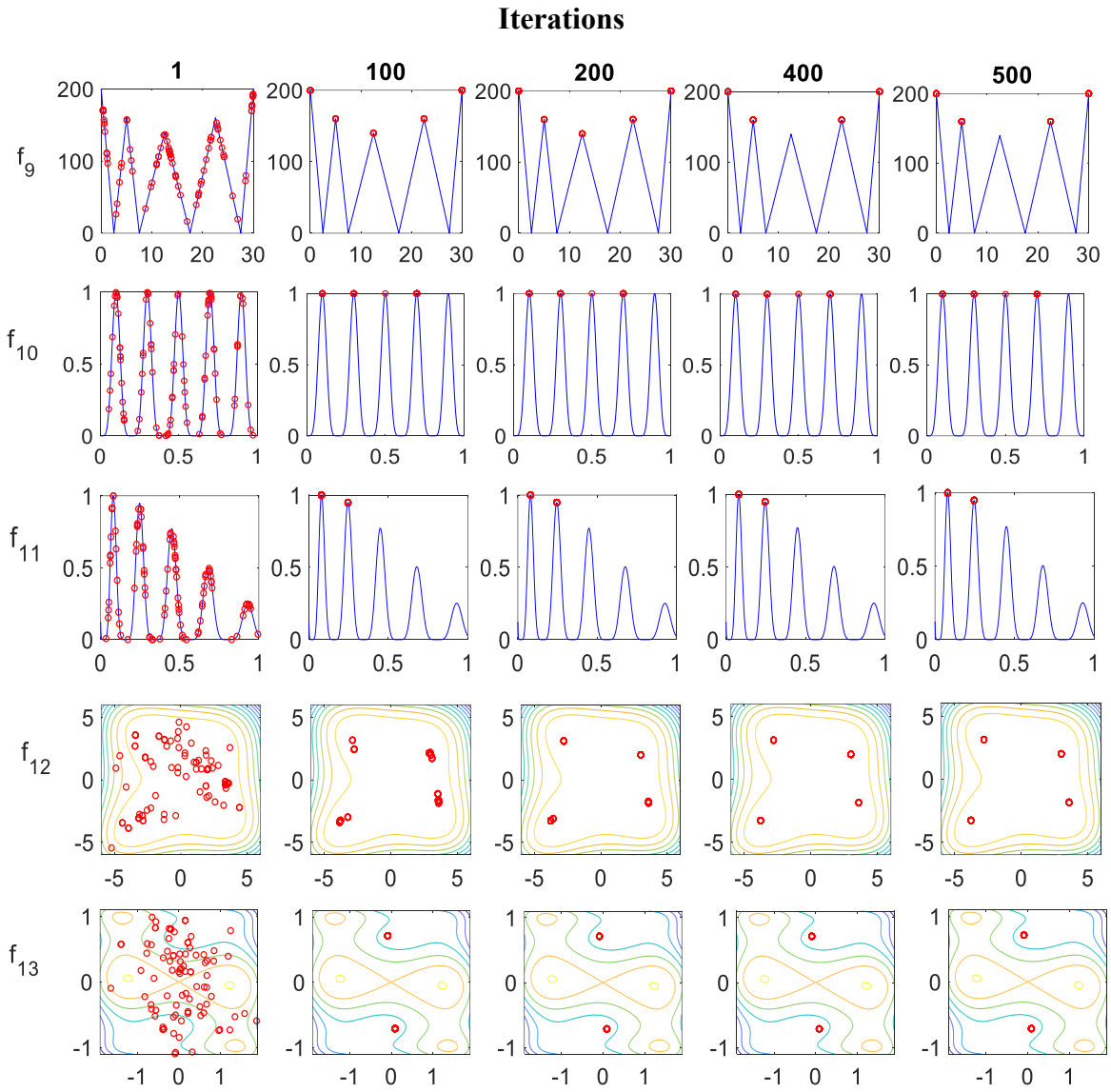


Figure 4.7 The search patterns of EVLNN for functions f_9, f_{10}, f_{11}, f_{12}, and f_{13}.

In Figure 4.8, the Shubert-2D (f_{14}) function shows nine pairs of optima distributed evenly apart. In this landscape, EVLNN can converge to about half of them, although some species appeared trapped in the local optima located around the global optima. For the Vincent-2D (f_{15}) function, out of the 36 global optima unevenly distributed on the landscape, EVLNN can locate approximately half of them. For the Modified Rastrigin-2D (f_{16}) function, EVLNN can locate all the 12 basins with high PR at 87.8%, 87.2%, and 70.8% for accuracy at ε = 1.0E-1 to 1.0E-3, respectively, but lower PR at 37.3% and 12.5% for ε ={1.0E-4, 1.0E-5}, respectively.

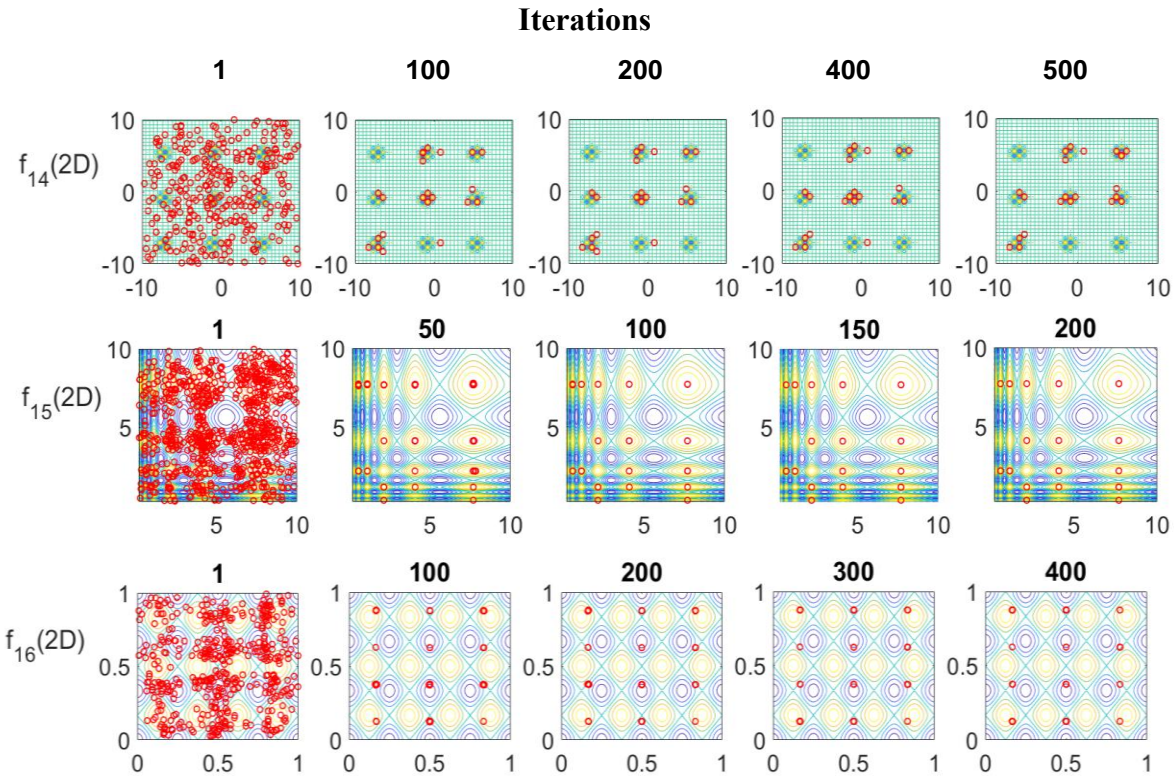


Figure 4.8 The search patterns of EVLNN for functions f_{14}, f_{15}, and f_{16}, respectively.

Figures 4.9 and 4.10 examine the convergence characteristics of EVLNN for functions f_9 to f_{13} and f_{14} to f_{16}, respectively. For the Shubert-3D (f_{14}) function (dotted lines in orange color), EVLNN took more than 250 iterations to locate better solutions. The iterations before were a period of 'stagnation' between generations 100 to 250. In evaluating the Shubert-3D (f_{14}) function, small species sizes may have caused the lack of genetic variation, failing to produce sufficient 'new alleles' for the population.

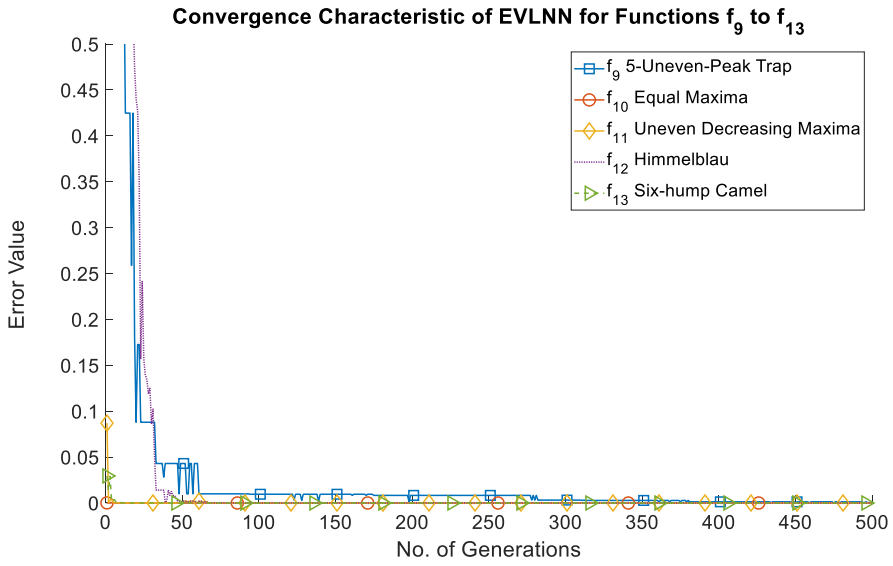


Figure 4.9 Convergence characteristics of EVLNN for functions f_9 to f_{13}

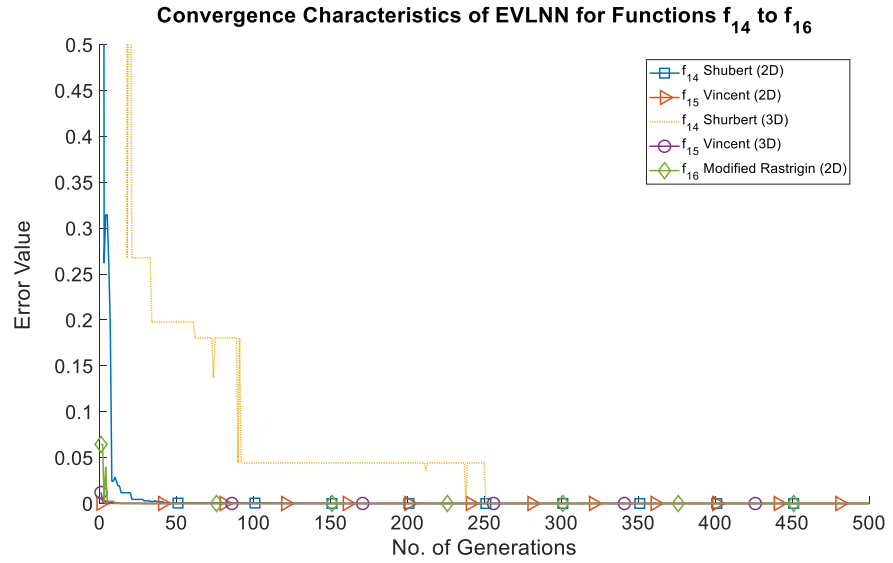


Figure 4.10 Convergence characteristics of EVLNN for functions f_{14} to f_{16}

The performance of EVLNN is compared to the results of a wide range of multimodal optimization algorithms listed in Table 4.8 for ε = 1.0E-03 to 1.0E-5. Table 4.11 shows that the state-of-the-art niching algorithms have superior performance, but EVLNN is also competitive. For example, EVLNN's performance is on par with all the algorithms for the Uneven Decreasing Maxima-1D (f_{11}) and Six-hump Camel-2D (f_{13}) functions with an average PR score of 100% for all ε. For the Five-Uneven-Peak Trap-1D (f_9) function,

EVLNN performed better than Crowding DE/rand/1/bin and MEA for ε = 1.0E-03 and better than DE/rand/1/bin for ε = 1.0E-05. For the Equal Maxima-1D (f_{10}) function, EVLNN edged out iPOP-CMA-ES for ε = 1.0E-03 and ε = 1.0E-04. For the Himmelblau-2D (f_{12}) function, EVLNN outperformed iPOP-CMA-ES, ANSGAII, MEA, and MSSPSO at ε = 1.0E-03 and ε = 1.0E-04 and bettered ANSGAII, MEA, and MSSPSO at ε = 1.0E-05.

It is interesting to note that for the Shubert-2D (f_{14}) function, EVLNN outperformed all the algorithms at ε = 1.0E-05, with a PR of 27.6%. None of the state-of-the-art EAs can locate the peak at this accuracy level. Examining the landscape of f_{14} (see Figure 4.8) shows that the landscape is characterized by many local optima, with 18 global optima positioned in 9 pairs. The individual pairs are closely located together, but the distance between the pairs is much more significant and evenly distributed. A similar conclusion was reached for Rastrigin-30D (f_7), where EVLNN was the only algorithm that found all peaks for the f_7 function at accuracy level ε = 1.0E-01. The results highlight the strength of EVLNN in locating global optima in a landscape where basins are close to each other. As the algorithm does not restrict species flocking to a basin if basins exist close to each other, the probability of EVLNN successfully locating the closely paired basins will be high.

In a real-world scenario, EVLNN can be applied to detect the source of hotspots (global optima) in data centers where there are multiple regions of hot spots (local optima) close to one another (multimodal). EVLNN can also be used to locate the region with the highest solar irradiance (global optima) receiving more solar irradiation over a nearby region (local optima) with the highest potential to produce solar energy.

For the Vincent-2D (f_{15}) function, EVLNN's performance surpassed DE/nrand/1/bin, DE/nrand/2/bin, iPOP-CMA-ES, and MSSPSO at ε = 1.0E-03 and ε = 1.0E-04, and DE/nrand/2/bin, iPOP-CMA-ES, MEA and MSSPSO at ε = 1.0E-05. For the Vincent-3D (f_{15}) function, EVLNN outdid DE/nrand/2/bin, DELG, iPOP-CMA-ES and MSSPSO at ε = 1.0E-03 and ε = 1.0E-04 and DELG, iPOP-CMA-ES and MSSPSO at ε = 1.0E-05. For the Modified Rastrigin-2D (f_{16}) function, EVLNN's performance topped iPOP-CMA-ES and MSSPSO at ε = 1.0E-03 and ε = 1.0E-04, and MSSPSO at ε = 1.0E-05.

Table 4.11 Comparison of Peak Ratios (PR) and Success Rates (SR) between EVLNN and the other state-of-the-art niching algorithms.

1. Function f_9, Five-Uneven-Peak Trap	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.470	0.140	0.100	0.000	0.003	0.000
DE/nrand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Crowding DE/rand/1/bin	0.090	0.000	0.020	0.000	0.000	0.000
NMMSO[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
N-VMO	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/1/bin[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA2[#&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DECG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELS_ajitter	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CMA-ES[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
iPOP-CMA-ES	0.780	0.560	0.780	0.560	0.780	0.560
ANSGAII	0.930	0.860	0.930	0.860	0.900	0.800
PNA-NSGAII	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{EA}$[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
MEA[*]	0.050	0.000	0.050	0.000	0.050	0.000
MSSPSO[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
2. Function f_{10} Equal Maxima	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.828	0.360	0.788	0.300	0.684	0.160
DE/nrand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Crowding DE/rand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NMMSO[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
N-VMO	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/1/bin[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA2[#&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DECG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELS_ajitter	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CMA-ES[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
iPOP-CMA-ES	0.772	0.180	0.752	0.160	0.732	0.140
ANSGAII	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
PNA-NSGAII	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{EA}$[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
MEA[*]	**1.000**	**1.000**	0.996	0.980	0.980	0.900
MSSPSO[*]	**1.000**	**1.000**	**1.000**	**1.000**	0.952	0.760

3. Function f_{11} Uneven Decreasing Maxima	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DE/nrand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Crowding DE/rand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NMMSO[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
N-VMO	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/1/bin[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA2[#&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DECG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELS_ajitter	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CMA-ES[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
iPOP-CMA-ES	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ANSGAII	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
PNA-NSGAII	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{EA}$[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
MEA[*]	**1.000**	**1.000**	**1.000**	**1.000**	0.960	0.960
MSSPSO[*]	**1.000**	**1.000**	**1.000**	**1.000**	0.940	0.940
4. Function f_{12} Himmelblau	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.905	0.620	0.885	0.540	0.615	0.100
DE/nrand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Crowding DE/rand/1/bin	**1.000**	**1.000**	0.995	0.980	0.420	0.040
NMMSO[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
N-VMO	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/1/bin[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA1	**1.000**	**1.000**	**1.000**	**1.000**	0.990	0.960
NEA2[#&+]	**1.000**	**1.000**	**1.000**	**1.000**	0.990	0.960
DECG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELS_ajitter	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CMA-ES[#]	**1.000**	**1.000**	**1.000**	**1.000**	0.990	0.960
iPOP-CMA-ES	0.725	0.160	0.725	0.160	0.725	0.160
ANSGAII	0.470	0.020	0.320	0.000	0.155	0.000
PNA-NSGAII	0.995	0.980	0.985	0.960	0.805	0.420
$LSEA_{EA}$[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
MEA[*]	0.055	0.000	0.005	0.000	0.005	0.000
MSSPSO[*]	0.010	0.000	0.005	0.000	0.000	0.000
5. Function f_{13} Six-hump Camel	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DE/nrand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

DE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Crowding DE/rand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NMMSO*&+	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
N-VMO	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/1/bin#	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA2#&+	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DECG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELS_ajitter	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CMA-ES#	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
iPOP-CMA-ES	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ANSGAII	0.940	0.880	0.900	0.800	0.680	0.380
PNA-NSGAII	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{EA}$*	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$*&+	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM*	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
MEA*	**1.000**	**1.000**	0.640	0.420	0.070	0.000
MSSPSO*	0.650	0.460	0.050	0.000	0.000	0.000

6. Function f_{14} Shubert (2D)	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.498	0.000	0.450	0.000	**0.276**	**0.000**
DE/nrand/1/bin	0.440	0.000	0.434	0.000	0.000	0.000
DE/nrand/2/bin	0.669	0.000	0.669	0.000	0.000	0.000
Crowding DE/rand/1/bin	0.972	0.740	0.107	0.000	0.000	0.000
NMMSO*&+	0.998	0.960	0.997	0.940	0.000	0.000
N-VMO	0.940	0.360	0.670	0.000	0.000	0.000
dADE/nrand/1/bin#	**1.000**	**1.000**	0.984	0.780	0.000	0.000
dADE/nrand/2/bin	**1.000**	**1.000**	0.833	0.020	0.000	0.000
NEA1	0.622	0.000	0.612	0.000	0.000	0.000
NEA2#&+	0.958	0.440	0.950	0.380	0.000	0.000
DECG	0.997	0.940	0.997	0.940	0.000	0.000
DELG	0.993	0.880	0.993	0.880	0.000	0.000
DELS_ajitter	**1.000**	**1.000**	0.999	0.980	0.000	0.000
CMA-ES#	0.782	0.020	0.776	0.020	0.000	0.000
iPOP-CMA-ES	0.094	0.000	0.090	0.000	0.000	0.000
ANSGAII	0.041	0.000	0.001	0.000	0.000	0.000
PNA-NSGAII	0.523	0.000	0.473	0.000	0.000	0.000
$LSEA_{EA}$*	0.996	0.920	0.993	0.880	0.000	0.000
$LSEA_{GP}$*&+	0.997	0.960	0.996	0.940	0.000	0.000
ALNM*	**1.000**	**1.000**	**1.000**	**1.000**	0.000	0.000
MEA*	0.004	0.000	0.000	0.000	0.000	0.000
MSSPSO*	0.000	0.000	0.000	0.000	0.000	0.000

7. Function f_{15} Vincent (2D)	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.387	0.000	0.380	0.000	0.338	0.000
DE/nrand/1/bin	0.349	0.000	0.337	0.000	0.333	0.000
DE/nrand/2/bin	0.276	0.000	0.276	0.000	0.275	0.000
Crowding DE/rand/1/bin	0.715	0.000	0.709	0.000	0.716	0.000
NMMSO*&+	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
N-VMO	0.945	0.140	0.901	0.000	0.806	0.000
dADE/nrand/1/bin#	0.892	0.020	0.823	0.000	0.732	0.000

dADE/nrand/2/bin	0.872	0.000	0.757	0.000	0.644	0.000
NEA1	0.691	0.000	0.657	0.000	0.640	0.000
NEA2[#&+]	0.918	0.060	0.914	0.040	0.911	0.040
DECG	0.659	0.000	0.656	0.000	0.646	0.000
DELG	0.582	0.000	0.582	0.000	0.582	0.000
DELS_ajitter	0.467	0.000	0.462	0.000	0.452	0.000
CMA-ES[#]	0.521	0.000	0.518	0.000	0.516	0.000
iPOP-CMA-ES	0.112	0.000	0.111	0.000	0.111	0.000
ANSGAII	0.668	0.000	0.509	0.000	0.346	0.000
PNA-NSGAII	0.726	0.000	0.709	0.000	0.683	0.000
$LSEA_{EA}$[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM[*]	0.822	0.000	0.811	0.000	0.792	0.000
MEA[*]	0.402	0.000	0.383	0.000	0.302	0.000
MSSPSO[*]	0.202	0.000	0.030	0.000	0.004	0.000

8. Function f_{14} Shubert (3D)	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.000	0.000	0.000	0.000	0.000	0.000
DE/nrand/1/bin	0.113	0.000	0.112	0.000	0.113	0.000
DE/nrand/2/bin	0.365	0.000	0.365	0.000	0.363	0.000
Crowding DE/rand/1/bin	0.716	0.000	0.290	0.000	0.038	0.000
NMMSO[*&+]	0.983	0.180	0.981	0.180	0.980	0.180
N-VMO	0.270	0.000	0.198	0.000	0.027	0.000
dADE/nrand/1/bin[#]	0.545	0.000	0.431	0.000	0.356	0.000
dADE/nrand/2/bin	0.724	0.000	0.660	0.000	0.613	0.000
NEA1	0.059	0.000	0.055	0.000	0.054	0.000
NEA2[#&+]	0.240	0.000	0.240	0.000	0.239	0.000
DECG	0.309	0.000	0.308	0.000	0.224	0.000
DELG	0.611	0.000	0.611	0.000	0.510	0.000
DELS_ajitter	0.000	0.000	0.000	0.000	0.000	0.000
CMA-ES[#]	0.115	0.000	0.115	0.000	0.115	0.000
iPOP-CMA-ES	0.020	0.000	0.020	0.000	0.020	0.000
ANSGAII	0.000	0.000	0.000	0.000	0.000	0.000
PNA-NSGAII	0.310	0.000	0.275	0.000	0.252	0.000
$LSEA_{EA}$[*]	0.893	0.000	0.886	0.000	0.886	0.000
$LSEA_{GP}$[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM[*]	0.822	0.000	0.811	0.000	0.792	0.000
MEA[*]	0.000	0.000	0.000	0.000	0.000	0.000
MSSPSO[*]	0.000	0.000	0.000	0.000	0.000	0.000

9. Function f_{15} Vincent (3D)	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.091	0.000	0.071	0.000	0.037	0.000
DE/nrand/1/bin	0.099	0.000	0.095	0.000	0.094	0.000
DE/nrand/2/bin	0.066	0.000	0.066	0.000	0.065	0.000
Crowding DE/rand/1/bin	0.274	0.000	0.274	0.000	0.270	0.000
NMMSO[*&+]	0.920	0.000	0.917	0.000	0.913	0.000
N-VMO	0.399	0.000	0.275	0.000	0.192	0.000
dADE/nrand/1/bin[#]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
dADE/nrand/2/bin	0.479	0.000	0.335	0.000	0.260	0.000
NEA1	0.407	0.000	0.381	0.000	0.359	0.000
NEA2[#&+]	0.584	0.000	0.581	0.000	0.579	0.000
DECG	0.242	0.000	0.240	0.000	0.237	0.000
DELG	0.012	0.000	0.012	0.000	0.012	0.000

DELS_ajitter	0.157	0.000	0.157	0.000	0.154	0.000
CMA-ES[#]	0.274	0.000	0.273	0.000	0.272	0.000
iPOP-CMA-ES	0.027	0.000	0.027	0.000	0.026	0.000
ANSGAII	0.345	0.000	0.140	0.000	0.038	0.000
PNA-NSGAII	0.318	0.000	0.298	0.000	0.276	0.000
$LSEA_{EA}$[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$[*&+]	0.744	0.000	0.668	0.000	0.556	0.000
ALNM[*]	0.289	0.000	0.264	0.000	0.150	0.000
MEA[*]	0.130	0.000	0.112	0.000	0.054	0.000
MSSPSO[*]	0.001	0.000	0.000	0.000	0.000	0.000

10. Function f_{16} Modified Rastrigin (2D)	ε = 1.0E-03		ε = 1.0E-04		ε = 1.0E-05	
	PR	SR	PR	SR	PR	SR
EVLNN	0.708	0.000	0.373	0.000	0.125	0.000
DE/nrand/1/bin	0.998	0.998	1.000	1.000	1.000	1.000
DE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Crowding DE/rand/1/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NMMSO[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
N-VMO	**1.000**	**1.000**	**1.000**	**1.000**	0.968	0.660
dADE/nrand/1/bin[#]	0.981	0.280	0.967	0.140	0.947	0.020
dADE/nrand/2/bin	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
NEA1	0.983	0.840	0.973	0.740	0.960	0.660
NEA2[#&+]	**1.000**	**1.000**	0.988	0.860	0.980	0.760
DECG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELG	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
DELS_ajitter	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CMA-ES[#]	0.998	0.980	0.992	0.900	0.978	0.760
iPOP-CMA-ES	0.343	0.000	0.313	0.000	0.303	0.000
ANSGAII	0.998	0.980	0.953	0.580	0.695	0.060
PNA-NSGAII	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{EA}$[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
$LSEA_{GP}$[*&+]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ALNM[*]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
MEA[*]	0.990	0.900	0.965	0.760	0.707	0.100
MSSPSO[*]	0.048	0.000	0.007	0.000	0.000	0.000

[*] *Algorithms participated in the CEC 2015 Competition. The rest are in the CEC 2013 Competition.*
[#] *Top three winners of CEC 2013*
[&] *Top three winners of CEC 2015*
[+] *Top three winners of CEC 2013 and CEC 2015 combined*

A re-rank is performed to assess the position of EVLNN against the state-of-the-art niching algorithms based on the PR scores at three accuracy levels, $\varepsilon = \{10^{-3}, 10^{-4}, 10^{-5}\}$ averaged over the ten multimodal benchmark functions f_9 to f_{16} including f_{14} (3D) and f_{15} (3D). The ranking result is shown in Table 4.12.

Table 4.12 Overall performance of EVLNN is ranked together with the state-of-the-art niching algorithms from the CEC 2013 and CEC 2015 competitions based on the average PR score at three accuracy levels, $\varepsilon = \{10^{-3}, 10^{-4}, 10^{-5}\}$ over ten multimodal benchmark functions f_9 to f_{16}.

S/N	Algorithms	Statistics for PR		Rank
		Mean	Std Dev	
1	EVLNN	**0.500**	0.369	**19**
2	DE/nrand/1/bin	0.684	0.398	16
3	DE/nrand/2/bin	0.715	0.374	15
4	Crowding DE/rand/1/bin	0.654	0.394	17
5	NMMSO	**0.956**	0.179	**1**
6	N-VMO	0.786	0.339	8
7	dADE/nrand/1/bin	0.889	0.238	4
8	dADE/nrand/2/bin	0.839	0.262	6
9	NEA1	0.748	0.344	12
10	NEA2	0.836	0.287	7
11	DECG	0.784	0.329	9
12	DELG	0.783	0.346	10
13	DELS_ajitter	0.728	0.400	14
14	CMA-ES	0.741	0.352	13
15	iPOP-CMA-ES	0.480	0.390	20
16	ANSGAII	0.565	0.400	18
17	PNA-NSGAII	0.754	0.318	11
18	$LSEA_{EA}$	**0.955**	0.180	**2**
19	$LSEA_{GP}$	**0.932**	0.203	**3**
20	ALNM	0.852	0.279	5
21	MEA	0.397	0.425	21
22	MSSPSO	0.330	0.447	22

Based on the average PR score at three accuracy levels, $\varepsilon = \{10^{-3}, 10^{-4}, 10^{-5}\}$ over ten multimodal benchmark functions f_9 to f_{16}, the overall performance of EVLNN is ranked 19 out of 22 algorithms with an average PR score of 0.500 and a standard deviation of 0.369. The top three algorithms are NMMSO, $LSEA_{EA}$, and LSEAGP. The bottom three are iPOP-CMA-ES, MEA, and MSSPSO.

As observed, the performance of EVLNN in this set of benchmark tests is relatively weaker than the other state-of-the-art EAs presented in the CEC 2013 and CEC 2015 competitions. It is important to note that the CEC algorithms are designed for the competition focusing on real-parameter optimization, whereas EVLNN is a parsimonious ANN built for forecasting. The CEC algorithms performed better as they are tailored to lower or higher dimension problems, and the algorithms' parameters need to be re-tuned in case of different benchmark functions [231]. On the other hand, the objective of EVLNN is to evolve

parsimonious ANN models that result in improved generalization and interpretability for practical applications. This distinct difference between EVLNN and the compared CEC algorithms lies in the nature of the problem.

The CEC test functions are well-known benchmarks in the evolutionary computing research community, and it serves as a gold standard for search algorithm design and comparison. The functions are well-designed and are typically unconstrained with a multimodal landscape consisting of minima, optima, and saddle points. The stringent standard presented by the CEC test functions with their parameter settings set a high benchmark for comparing different niching methods. In the algorithm design process, the strength and limitations of EVLNN are better understood through the set of test functions. Nonetheless, real-world problems are more diverse, with isolated regions that can be very different from these test functions. Thus an algorithm's good results for the test functions may not translate to good results in the real world. In this regard, an added experiment using open-access real-world time-series data for benchmark testing is introduced for comparison and to help assess the applicability of EVLNN.

4.5. Time-series Electricity Load Data as Benchmark for Forecasting

Test functions are generally "well behaved" functions with regular domains, while realistic problems have many nonlinear complex constraints, and the domains can be formed by many isolated regions or islands [206]. Various open-access data platforms have made datasets available to be analyzed to advance machine learning technologies. One example is the time-series dataset for electricity load forecasting made available by MATLAB for researchers and AI engineers to develop predictive models in the energy sector. It consists of historical electricity loads of Sydney, electricity prices for the Australian market, the Sydney temperature, and New South Wales weather data sampled at 30 minutes intervals from 2006 to 2010 [232]. In this experiment, a subset of one month's data from 1st to 31st July 2010, consisting of a sample size of 1,490 out of a potential 87,600, was extracted for

model training. The split of 70% for training and 30% for testing resulted in the training datasets of 1,043 samples. This sample size is not extensive considering the high variability of electricity load. The identified model is evaluated for a day-ahead forecast on 1 August 2010. The aim of selecting a one-month dataset from a five-year dataset is to test the predictive accuracy of EVLNN with a small sample size. Reducing the reliance on large datasets has potential for real-world applications where models that generalize well on smaller sample sizes can reduce the cost of implementation. A summary of the input and response variables is shown in Table 4.13, and their descriptive statistics are shown in Table 4.14.

Table 4.13 Input features and response variable used.

Input Features	**Abbreviation**
$x_1(t)$: Dry Bulb Temperature at time *t*	DB
$x_2(t)$: Dew Point Temperature at time *t*	DP
$x_3(t)$: Wet Bulb Temperature at time *t*	WB
$x_4(t)$: Relative Humidity at time *t*	RH
$x_5(t)$: Electricity prices at time *t*	EP
Response Variable	**Abbreviation**
$y_1(t)$: Electricity load at time *t*	EL

Table 4.14 A summary of the descriptive statistics of a real-world dataset used to evaluate EVLNN's performance for electricity load forecasting.

Input	**Feature**	**Unit and Symbol**	**Range**	**Min**	**Max**	**Mean**	**Std Dev**
$x_1(t)$	DB	degree Celsius, ºC	15.00 ºC	6.00 ºC	21.00 ºC	12.59 ºC	± 2.91 ºC
$x_2(t)$	DP	degree Celsius, ºC	14.90 ºC	-0.30 ºC	14.60 ºC	7.94 ºC	± 3.40 ºC
$x_3(t)$	WB	degree Celsius, ºC	12.50 ºC	4.30 ºC	16.80 ºC	10.37 ºC	± 2.47 ºC
$x_4(t)$	RH	Percentage, %	66.00 %	33.00 %	99.00 %	75.04 %	± 15.39 %
$x_5(t)$	EP	Australian Dollar, AUD	104.44 AUD	3.96 AUD	108.40 AUD	27.40 AUD	± 7.30 AUD
Output	**Feature**	**Unit and Symbol**	**Range**	**Min**	**Max**	**Mean**	**Std Dev**
$y_1(t)$	EL	MegaWatt hour, MWh	6290.40 MWh	6617.72 MWh	12908.12 MWh	9688.56 MWh	± 1356.72 MWh

Figure 4.11 shows the dataset consisting of a sample size of 1,490, where 70% is used for training (blue plot) and 30% for testing (red plot) EVLNN, PSO-NN, DE-NN, and GA-NN models. The identified models are then deployed to forecast a one-day electricity load on

1st August 2010 (green plot). The dataset is first normalized to values between 0 and 1. Subsequently, 50 runs of the experiments are conducted, and their performances are averaged and shown in Table 4.15. While PSO-NN has the lowest training MSE value, EVLNN has the lowest testing MSE value and standard deviation. The identified models from each algorithm are applied to forecast a one-day electricity load in Sydney, Australia, on 1st August 2010.

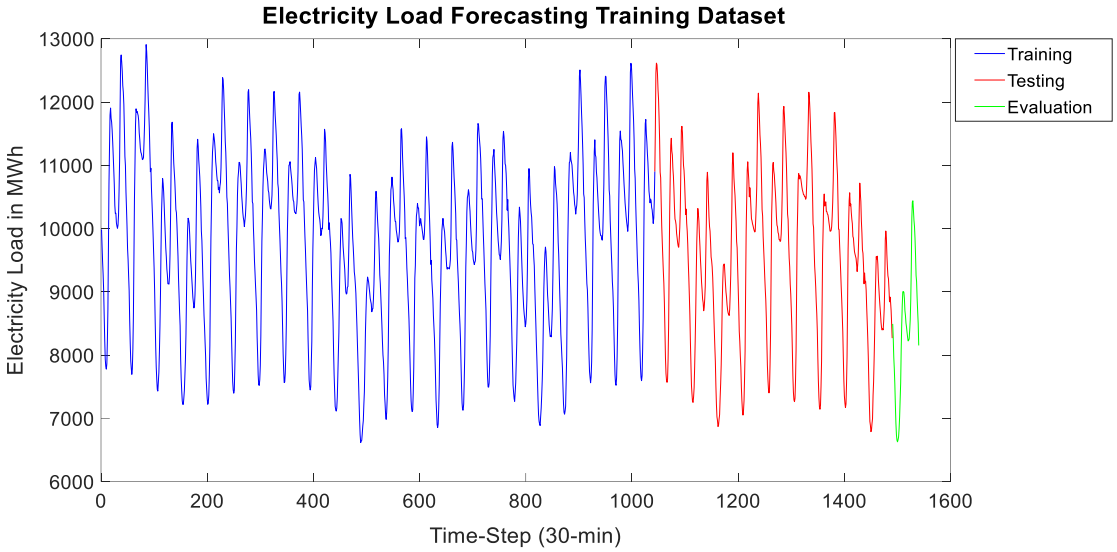


Figure 4.11 Time-series electricity load from 1st July 2010 to 1st August 2010. The blue and red plots indicate the training and testing dataset. The green plot is an out-of-sample dataset used to evaluate the models.

Table 4.15 Comparing the training and testing MSE scores averaged over 50 runs.

Models	Training MSE		Testing MSE	
	Mean	**Std Dev**	**Mean**	**Std Dev**
EVLNN	0.02936	0.00322	**0.03228**	**0.00332**
PSO-NN	**0.02496**	**0.00155**	0.04202	0.00599
DE-NN	0.02593	0.00153	0.03899	0.01005
GA-NN	0.03004	0.00304	0.03368	0.00400

The predicted plot is shown in Figure 4.12. EVLNN predicted well at the first upturn but undercompensated at the first downturn and the second upturn. DE-NN and GA-NN

predicted well at the first upturn but overcompensated at the first downturn and the second upturn. PSO-NN had a better prediction at the first downturn but overcompensated at the first and second upturn.

The results indicated that PSO-NN, DE-NN, and GA-NN have higher positive biases, over-compensating at the upturns. In contrast, EVLNN has a lower negative bias, under-compensating at the upturns. Overall, EVLNN has the lowest test MSE compared to PSO-NN, DE-NN, and GA-NN, whose values are 0.0356, 0.0578, 0.0189, and 0.0401, respectively. The result demonstrated that EVLNN is closer to detecting the peaks than troughs in a time-series dataset. Another crucial statistical measure is the standard deviation value. All the other models, except EVLNN, suffer from higher testing standard deviation values. While EVLNN tends to have a lower negative bias, it does not suffer from a high standard deviation. It can be reasoned that these models represent the features of the training dataset very well but failed to generalize to the testing dataset compared to EVLNN. To achieve a low standard deviation value, search algorithms must produce near-optimal models to consistently converge to the global optima of the cost function. Therefore, EVLNN has performed comparatively well in this real-world application.

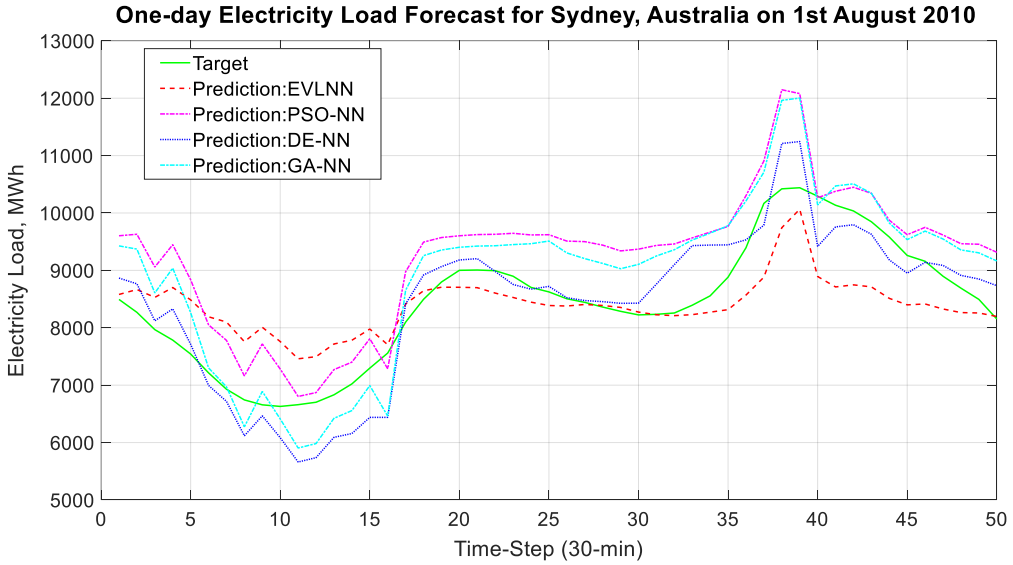


Figure 4.12 One-day electricity load forecast for Sydney, Australia, for an out-of-sample dataset from 1st August 2010 at 0000H to 2nd August 2010 at 0000H, at 30 mins time resolution.

4.6. Chapter Summary

This chapter evaluated EVLNN with a comprehensive set of benchmark functions to assess the algorithm's performance. EVLNN's search capability tested with benchmark functions f_1 to f_8 demonstrated that EVLNN had a strong ability to locate the global optima in low-dimensional unimodal and multimodal landscapes. In particular, EVLNN had achieved an average PR of 100% for Bohachevsky N.1-1D (f_1), and Booth-2D (f_2), and 81.2% and 93.2% for Ackley-2D and Rosenbrock-2D (f_6) functions across all $\varepsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. This result is comparable to the performance of PSO, DE, and GA. However, for high-dimensional optimization problems, such as the Sphere-30D (f_3), Brown-30D (f_4), Rastrigin-30D (f_7), and Griewank-30D (f_8) functions, EVLNN's average PR and SR values were lower compared to PSO, DE, and GA. EVLNN's less desirable performance in high dimensionality problems is due to a drop in search efficiency caused by increased crossover complexity due to a larger chromosome matrix size. Nonetheless, the bright spot is that for the Rastrigin-30D (f_7) function, EVLNN can locate 100% of the optima at $\varepsilon=\{10^{-1}\}$ for all runs. In contrast, PSO, DE, and GA are unsuccessful, with a PR of 0%. inferior

Test results from benchmark functions, f_9 to f_{16}, again demonstrated that EVLNN had performed well for 1D and 2D multimodal test functions f_9 to f_{13} across all $\varepsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. For the Uneven Decreasing Maxima (f_{11},) and Six-hump Camel Back (f_{13}) functions, EVLNN recorded a PR of 100% for all ε. The algorithm also produced a high PR of 84.3% and 79.4% for the Himmelblau (f_{12}) and Equal Maxima (f_{10}) functions. For the scalable multimodal functions f_{14} to f_{16}, EVLNN performed moderately well for the Modified Rastrigin (f_{16}), Shubert-2D (f_{14}), and Vincent-2D (f_{15}) functions, with PR of 59.1%, 44.6%, and 37.6%, respectively. However, for the Five-Uneven-Peak Trap (f_9) function, EVLNN could not find any peaks at $\varepsilon = \{10^{-4}, 10^{-5}\}$. EVLNN also suffered from locating the peaks associated with Shubert-3D (f_{14}) and Vincent-3D (f_{15}) functions, coping only with PR of 1.24% and 8.8%, respectively. These optimization functions exposed EVLNN's weaknesses in high-dimension problems. One possible explanation may be that the smaller species have fewer genetic variability to evolve to better solutions.

Nonetheless, it is interesting to note that EVLNN had outperformed all the CEC 2013 and CEC 2015 niching algorithms in the Shubert-2D (f_{14}) function, where it managed a PR score of 27.6% at $\varepsilon = \{10^{-5}\}$. In contrast, all the other niching algorithms were unsuccessful with 0% at this accuracy level. Overall, EVLNN's average PR score of 0.500 across the ten benchmark functions f_9 to f_{16} (including Shubert-3D and Vincent-3D) is ranked 19 out of 22 CEC 2013 and CEC 2015 state-of-the-art niching algorithms, with its performance ahead of iPOP-CMA-ES, MEA, and MSSPSO. Despite the relatively weaker results by EVLNN compared to other CEC algorithms, a distinct difference lies in the algorithm design purpose. That is, the CEC algorithms are focused on real-parameter optimization, whereas EVLNN is an ANN that is built for forecasting. The performance of EVLNN was thus validated using the open-access real-world time-series electricity load data as a benchmark for forecasting. The results demonstrated that EVLNN is promising and applicable for real-world energy prediction problems.

Chapter 5

5. Energy Consumption Prediction in Hadoop Cluster

5.1. Introduction

Data centers provide several platforms for managing and processing big data; the Hadoop platform is one of the most popular [233]. With its rich ecosystem composed of a set of feature-rich development tools, Hadoop's popularity grew and is extensively used today by Corporations, Enterprises, and Internet companies to analyze data-intensive problems [234] [235] [236]. While Hadoop is highly scalable and fault-tolerant for processing massive data, the energy consumed by Hadoop data centers is intense. This phenomenon has presented significant optimization opportunities and attracted extensive interest in its energy efficiency research. Energy consumption prediction methods are critical to data center sustainability efforts to improve the energy efficiency of Hadoop data centers. However, due to complexity and heterogeneity in data center scenarios, it is difficult to estimate energy consumption accurately using conventional approaches. This chapter presented EVLNN as an ML model for Hadoop energy consumption prediction using multiple energy-related features. System environment, applications, and hardware-related data are collected from a Hadoop testbed for model training. The identified models were compared with neural networks trained using other EAs, namely Particle Swarm Optimization (PSO-NN), Differential Evolution (DE-NN), and Genetic Algorithm (GA-NN). The results showed that EVLNN had outperformed the other models, verifying EVLNN's predictive accuracy and the capability to generalize to new data. Further experiments were conducted to determine the factors contributing to energy consumption using the ensemble-based approach to sensitivity analysis, where input variables from the identified models were analyzed to assess their relative importance.

5.2. Hadoop – Background

Hadoop is a distributed file system architecture for massively parallel processing (MPP). The architecture combines the Hadoop File System (HDFS) [15] in the data layer and MapReduce version one [16] or MapReduce version two [237] in the software, forming a new computing paradigm. Hadoop emerged as an open-source Apache project in 2009 and constituted a new area in academic research in the MPP system [233] [238] [239] [240]. In Hadoop's Client-Master-Slave architecture, the master in the data layer is known as the NameNode, and the slaves are known as the DataNodes. The files stored in HDFS are first divided into fixed-size blocks by the NameNode and subsequently replicated and distributed across the DataNodes in the Hadoop cluster for enhanced performance and reliability. In the software layer, the JobTracker daemons manage the cluster resources and the parallel processing of the HDFS data via the MapReduce jobs submitted by the client. The client is any machine interacting with and requesting services from the Hadoop cluster.

Upon receiving the MapReduce job, JobTracker then communicates to the NameNode to determine the DataNodes where the data is located. It then divides the jobs to MapReduce tasks and assigns them to the TaskTracker daemons that reside in the DataNodes to process them. Each TaskTracker is configured with a static allocation of fixed-size "slots" where a map slot, known as a mapper, and a reduced slot, known as a reducer, are used by the map task or reduce task, respectively. In addition, the TaskTracker provides job progress information to the JobTracker that monitors the overall status of the job. The number of mappers created is dependent on the number of input file splits, and the number of reducers is configured by the system administrator. The map task processes the data and outputs a list of key-value pairs as intermediate data. The intermediate data will then be shuffled and sorted using the keys and subsequently merged. The reducer task processes the intermediate data, further reducing them into smaller units and writing the result in HDFS. This Client-Master-Slave architecture allows Hadoop to scale and handle big data well. Figures 5.1 and 5.2 depict the execution of a MapReduce job. Details of the map and reduce phases can be found in [241].

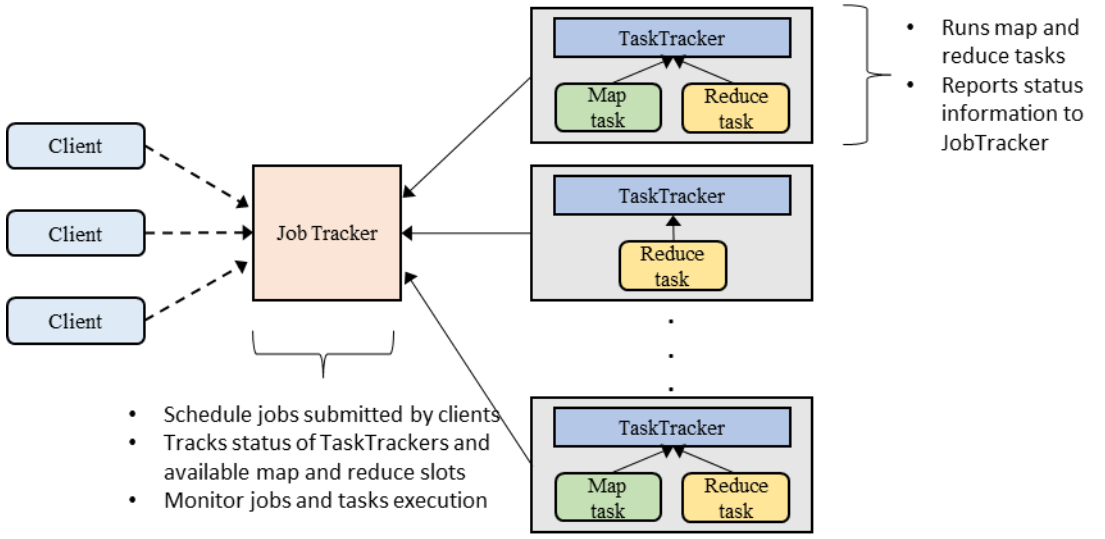


Figure 5.1 The MapReduce software layer architecture.

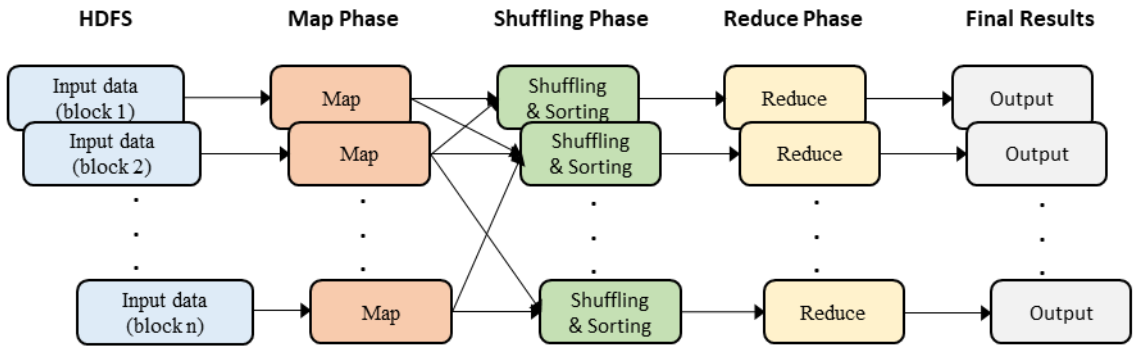


Figure 5.2 The MapReduce job's computation phases.

5.3. The Hadoop Testbed

A local testbed was set up to study the energy performance of Hadoop. The setup consists of a 120-core Hadoop cluster, an intelligent power distribution unit to measure energy consumption, and software monitoring tools to collect energy-related data from the Hadoop cluster to analyze and train the EVLNN model.

5.3.1. Physical Testbed Setup

Figure 5.3 shows the physical infrastructure and connectivity of the Hadoop testbed set up for the experiment. The testbed comprises five servers configured to form one Hadoop NameNode and four Hadoop DataNodes. The NameNode is running on an HP ProLiant DL360P Generation 8 server that comes with 64 Gigabyte (GB) memory, dual-socket Intel(R) Xeon(R) CPU E5-2667 @ 2.90GHz with a 6-core CPU cum hyper-threading technology.

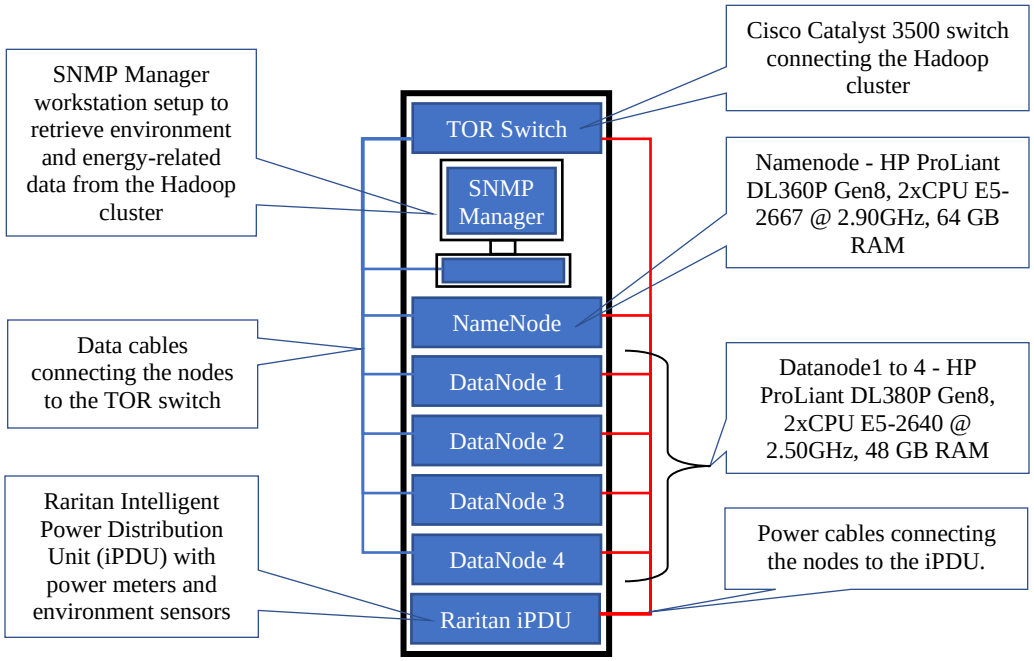


Figure 5.3 The Hadoop testbed.

The DataNodes runs on HP ProLiant DL380P Generation 8 servers with 48 GB memory, dual-socket Intel(R) Xeon(R) CPU E5-2640 @ 2.50GHz 6-core CPU cum hyper-threading technology. Hyper-threading is a process by which a CPU divides up its physical cores into virtual cores that are treated as physical cores by the operating system. CPUs with six cores use this process to create four threads or four virtual cores. Hence, the Hadoop cluster consists of five servers with dual-core sockets with 6-core CPUs, forming a total of 120-core with hyper-threading enabled. All nodes were installed with the open-source CentOS version 6.5 Linux operating system and Apache Hadoop version 0.20.1. The nodes are

interconnected via a top-of-rack (TOR) Gigabit Ethernet switch with 1 Gigabit per second (Gbps) connections. In addition, a Raritan intelligent Power Distribution Unit (iPDU) provides power connection to the clusters and is used to measure and record the power consumption by the nodes. The JobTracker daemon was installed in the NameNode. All DataNodes were installed with the TaskTracker daemons.

5.3.2. Hadoop Software Configuration

The Hadoop daemons were configured via four primary eXtensible Markup Language (XML) files, namely, *hdfs-site.xml*, *core-site.*xml, *mapred-site.xml*, and *yarn-site.xml*. Most of the parameters in the XML files are left in their default settings except for the following parameters configured, as shown in Table 5.1, that ensured a stable system performance.

Table 5.1 Hadoop configuration parameters and values.

Hadoop Configuration Parameter	**Description**	**Value**	**Configuration File**
dfs.replication	Replication factor	2	hdfs-site.xml
mapred.child.java.opts	Java virtual machine (JVM) heap size for the MapReduce processes	512 MB	mapred-site.xml
mapreduce.task.io.sort.mb	The total amount of buffer memory to use while sorting files, in megabytes	200 MB	mapred-site.xml
mapreduce.map.sort.spill.percent	The soft limit in the serialization buffer. Once reached, a thread will begin to spill the contents to the disk in the background.	0.9	mapred-site.xml
mapreduce.task.io.sort.factor	The number of streams to merge at once while sorting files. This parameter determines the number of open file handles.	20	mapred-site.xml
dfs.blocksize	The default block size for new files, in bytes.	128 MB	hdfs-site.xml

5.3.3. Monitoring Tools for Data Acquisition

A set of monitoring tools was installed to collect system parameters data and analyze energy consumption. These parameters can be classified into the application, system hardware, power and environment, which require separate acquisition tools. Application data were

acquired using built-in Hadoop job-related counters [242]. Data from system hardware parameters were acquired using Ganglia (version 3.7.1) [243], an open-source performance monitoring software that measures the utilization of various system metrics such as CPU, memory, and networks [244]. Data from the power and environmental parameters were acquired through the Simple Network Management Protocol (SNMP) software. The SNMP software communicates with the intelligent Power Distribution Unit (iPDU) to extract power consumption, temperature, and humidity information. The iPDU contains the SNMP Management Information Base (MIB), which can be queried through their respective SNMP Object Identities (OIDs). Table 5.2 shows the SNMP OID strings configured in the SNMP Manager software to retrieve those data.

Table 5.2 SNMP OID strings and polling interval for power consumption data.

Hadoop Cluster Components	iPDU Outlet	SNMP OID String	Description	Polling Interval (secs)
NameNode	1	1.3.6.1.4.1.13742.6.5.4.3.1.4.1.1.5	Active power	5
DataNode1	2	1.3.6.1.4.1.13742.6.5.4.3.1.4.1.2.5	Active power	5
DataNode2	3	1.3.6.1.4.1.13742.6.5.4.3.1.4.1.3.5	Active power	5
DataNode3	4	1.3.6.1.4.1.13742.6.5.4.3.1.4.1.4.5	Active power	5
DataNode4	5	1.3.6.1.4.1.13742.6.5.4.3.1.4.1.5.5	Active power	5
Network Switch	6	1.3.6.1.4.1.13742.6.5.4.3.1.4.1.6.5	Active power	5
Temperature sensor	10	1.3.6.1.4.1.13742.6.3.5.3.1.3.1.12.10	Temperature	5
Humidity sensor	11	1.3.6.1.4.1.13742.6.3.5.3.1.3.1.12.11	Humidity	5

5.4. Predictive Modeling for the Hadoop System

5.4.1. Payload Generation, Workload Simulation, and Data Acquisition

Two commonly used Hadoop applications, Wordcount and Terasort, were used to simulate workloads on the Hadoop testbed. These applications are commonly employed for energy efficiency and performance studies in Hadoop research [245] [246]. Wordcount is a simple application that counts the number of times each word (key) appears (value) in the input

dataset. Wordcount has many real-world applications in text analytics. For example, by analyzing the word frequency of Twitter data related to weather, one can learn about the liveliness of climate discussion on social media. Terasort comes with Hadoop, and it is used to sort key-value tuples. This application is generally used for benchmarking the performance of Hadoop systems.

In order to simulate the workloads, payloads must be generated for the Wordcount and Terasort applications. Various payload sizes from 100 MB to over 100 GB were created. For Wordcount, the payloads are generated using the Linux command-line interface's '>>' redirection symbol to append multiple text files into a large file. For Terasort, the payloads are generated using the Teragen application in Hadoop. Teragen can generate large datasets to be sorted by Terasort.

During workload simulation, the Wordcount and Terasort were run the payload files in sizes 100MB, 200MB, 300MB, …, 100GB, and beyond. In the process, energy-related data such as disk I/O, network transfer, memory, and computational activities [247] were then acquired through tools such as Ganglia, SNMP, and Hadoop built-in counters. The data were subsequently used to train the EVLNN model for energy consumption prediction.

Most energy consumption research for Hadoop focuses only on acquiring a few input features. EVLNN uses multiple energy-related input features to provide insight into the feature's importance relative to the system energy consumption. A total of 23 input features and the energy consumption data were acquired from the Hadoop testbed. These features and their methods of acquisition are presented in Table 5.3.

Table 5.3 A list of 23 input features and one output variable for data acquisition from the Hadoop testbed.

Category	Parameters	Unit	Description	Method of Acquisition
System Utilization	1. Cluster CPU utilization (system)	%	Percentage of CPU time used by the kernel	Ganglia
	2. Cluster CPU utilization (user)	%	Percentage of CPU time used by user space processes to run applications	Ganglia
	3. Cluster CPU utilization (wait)	%	Percentage of CPU time spent waiting for input or output operations, like reading or writing to disk.	Ganglia
	4. Cluster CPU time spent	Seconds	Cluster CPU time spent during the MapReduce process	Hadoop counters
	5. Memory use	GB	The part of the memory used for processing out of the total available physical memory	Ganglia
	6. Memory cache	GB	The part of the memory used to cache the contents of frequently-used disk data	Ganglia
	7. Memory buffer	MB	Memory buffered during MapReduce process	Ganglia
	8. System process	Number	Number of processes running	Ganglia
Disk I/O Activities	9. File: Map byte read	GB	Data read by Mapper from local disk	Hadoop counters
	10. File: Reduce byte read	GB	Data read by Reducer from local disk	Hadoop counters
	11. File: Map byte written	GB	Data written by Mapper to local disk	Hadoop counters
	12. File: Reduce byte written	GB	Data written by Reducer to local disk	Hadoop counters
	13. HDFS: Reduce byte written	GB	Data written by Reducer to HDFS	Hadoop counters
	14. Reduce Shuffle bytes (Total)	GB	Data transferred from Map to Reduce	Hadoop counters
Network Transfer	15. Network (In)	Gbps	Data received	Ganglia
	16. Network (Out)	Gbps	Data transmitted	Ganglia
Job Profile	17. File size	GB	Size of MapReduce jobs	Hadoop counters
	18. Job completion time	Hour	Time taken to finish a MapReduce job	Hadoop counters
	19. Number of Mappers	Number	Job's instruction number	Hadoop counters
	20. Number of Reducers	Number	Job's instruction number	Hadoop counters
	21. Workload type	Number	0 for Wordcount, 1 for Terasort	N.A.
Environment	22. Rack Relative Humidity	%	Relative humidity measured within the rack	SNMP
	23. Rack Temperature	°C	Rack temperature hosting the cluster	SNMP
Energy	1. Energy consumption	kWh	Energy consumed by cluster (output variable)	SNMP

5.4.2. Exploratory Data Analysis

Data exploration was initially performed to analyze and better understand the energy consumption of the Hadoop cluster. Figure 5.4 presents the instantaneous power of a Terasort MapReduce job on a 50 GB payload file. The data was acquired via SNMP polling at an interval of 5 seconds. It was observed that individually, the DataNodes (also called the worker nodes) had almost similar power consumption patterns. The consumed power of the DataNodes was also higher than the NameNode (also known as the head node). This pattern was expected as DataNodes executes the Mapper and Reducer programs, whereas NameNode only manages the metadata and job scheduling. Figure 5.5 shows the aggregated power where there was an initial surge in the power consumption to almost 1400 W before it receded to a mean value of 1037.86 W. The elapsed time to execute the Terasort application was 36 minutes and 18 seconds (or 2,178 secs) with a 50 GB payload.

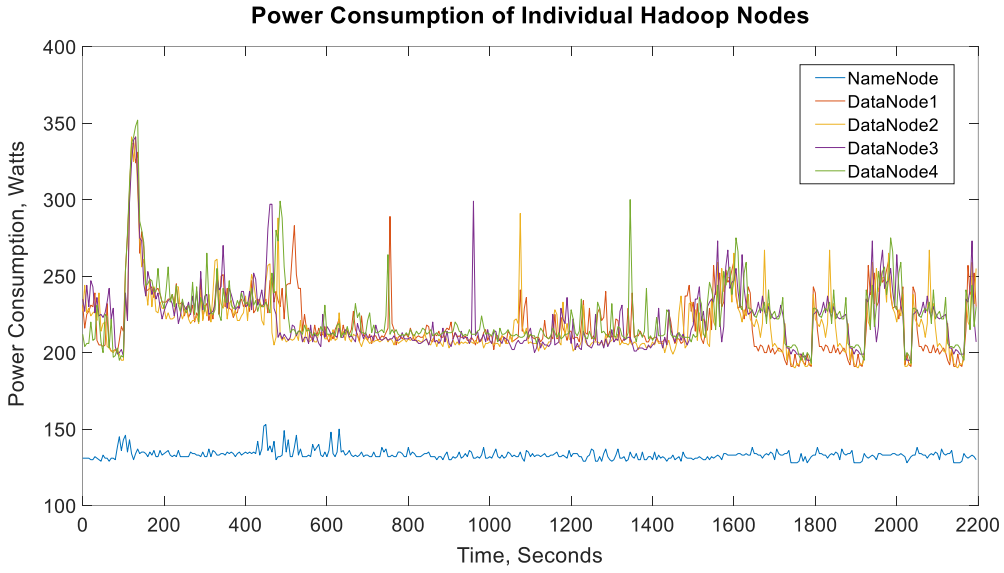


Figure 5.4 Instantaneous power chart for Terasort application with a 50 GB payload.

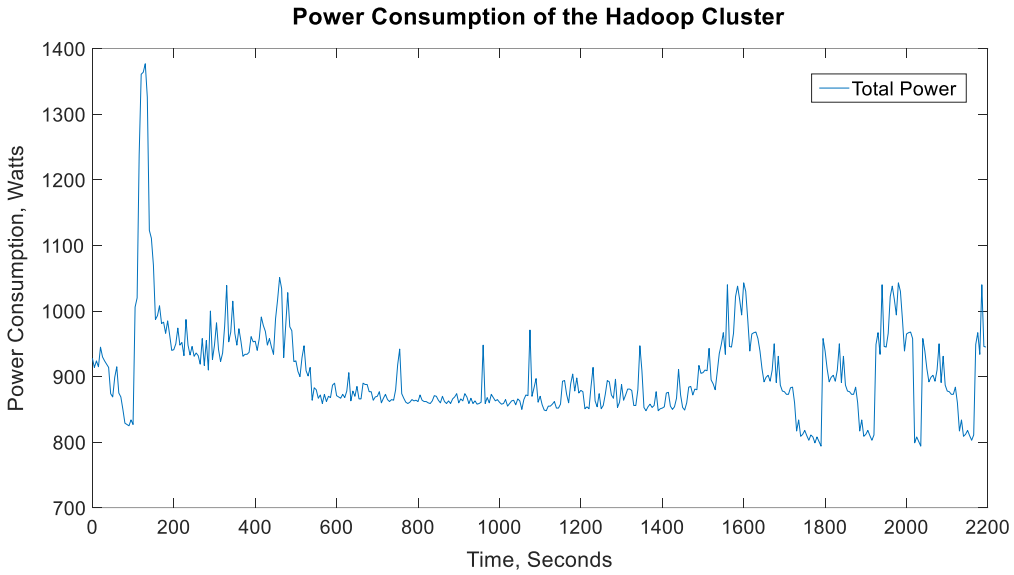


Figure 5.5 Aggregated power chart for Terasort application with a 50 GB payload.

A similar analysis was performed for a Wordcount MapReduce job with a 50 GB payload. However, the chart in Figure 5.6 depicts a distinctly different power consumption pattern where higher power is consumed but at a shorter elapsed time. Figure 5.7 shows the aggregated power, sustained at approximately 1520 W for at least 200 seconds. The mean power consumed was 1330.23 W over the elapsed time of the Wordcount application of 697 seconds.

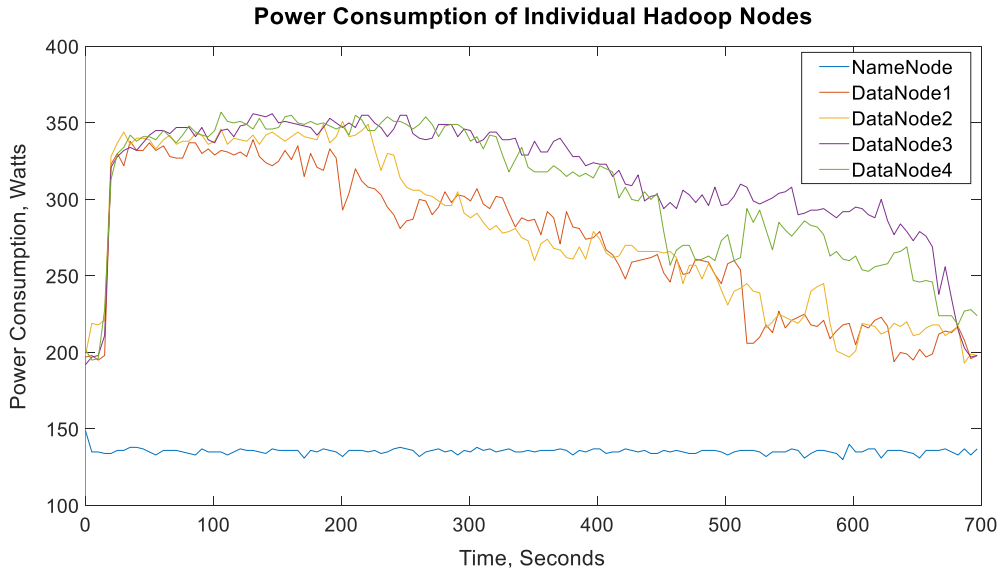


Figure 5.6 Instantaneous power chart for Wordcount application with a 50 GB payload.

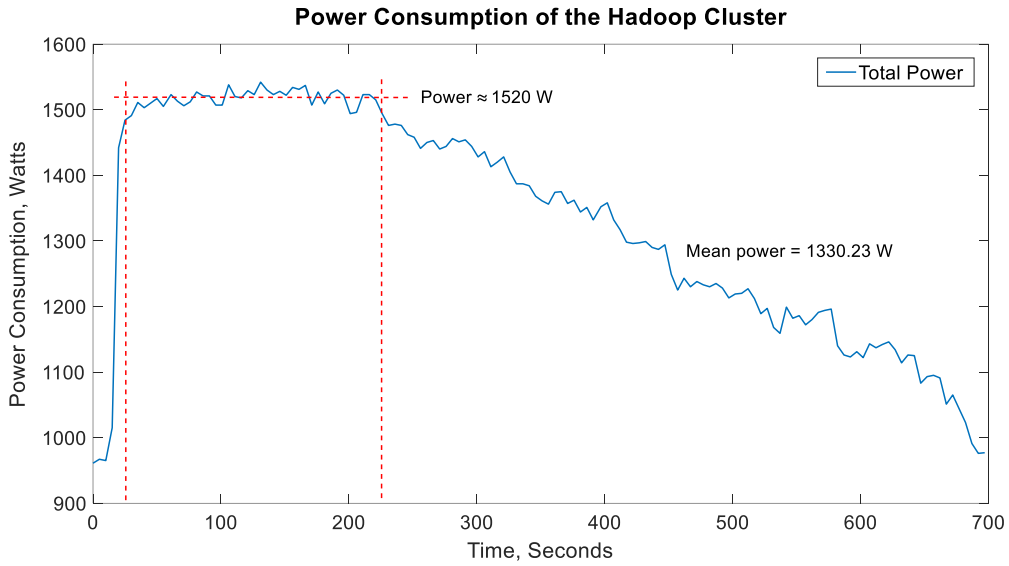


Figure 5.7 Aggregated power chart for Wordcount application with a 50 GB payload.

Instantaneous power varies all the time; hence the dataset is considered too noisy for model training. There are ways to smooth out the constant fluctuation of instantaneous power. One of them is to work with mean power or energy data. Mean power represents the power consumed over a specific period, and mean energy represents the mean load performed over that period. The cluster's energy consumption *E,* therefore, can be calculated using Equation 5.1,

$$E = PT \tag{5.1}$$

where *P* is the power measured in Watts and *T* is the period measured in seconds to perform the work. Equation 5.1 could also be enhanced when dealing with cumulative energy change by integrating the instantaneous power, *P(t),* in a period between t_1 and t_2, given in Equation 5.2,

$$E = \int_{t_1}^{t_2} P(t)\, dt \tag{5.2}$$

However, the function of the power consumption curve is generally not known. Hence the energy consumed by Terasort and Wordcount workloads over an interval $[t_1\ t_2]$ can be approximated by the area under the curve for that period. The computed result was thus 0.62 kWh and 0.25 kWh, respectively. Comparatively, the Terasort application, while

needing lower power but a longer elapsed time, consumed more than twice the energy level of the Wordcount application, which needed higher power but shorter elapsed time.

With that initial understanding, three additional workloads of payload sizes, 55 GB, 60 GB, and 65 GB, were generated. The energy consumption pattern was further analyzed with data collected from the energy-related features. The result is presented in Table 5.4. Several observations can be drawn for the Wordcount application; firstly, energy consumption generally increases with the file size. However, I/O activities counters were low, such as *map byte read*, *map byte written*, *reduce byte read*, *reduce byte written*, and *reduce shuffle byte*. This observation suggests the Wordcount application is light on disk I/O activities. Secondly, the CPU and memory utilization counters were high, such as *cluster CPU (user)*, *memory (use)*, and *system (process)*. This observation suggests that the Wordcount application is compute-intensive.

Table 5.4 Energy-related features and their respective data obtained from executing the MapReduce Wordcount and Terasort workloads with different payload sizes.

Energy-Related Metric	**MapReduce Workloads**							
	50 GB		**55 GB**		**60 GB**		**65 GB**	
	WC*	**TS^**	**WC**	**TS**	**WC**	**TS**	**WC**	**TS**
Map Byte Read (GB)	3.16	50.35	3.48	55.38	3.80	61.42	4.12	66.46
Map Byte Written (GB)	3.36	100.15	3.69	110.17	4.03	122.19	4.37	132.20
Reduce Byte Read (GB)	0.17	50.07	0.19	55.07	0.21	61.08	0.22	66.09
Reduce Byte Written (GB)	0.17	50.08	0.19	55.08	0.21	61.09	0.22	66.10
Reduce Byte Written HDFS (GB)	0.00	50.17	0.00	55.18	0.00	61.20	0.00	66.22
CPU Time Spent (min)	288.15	154.85	313.80	129.58	351.70	150.69	381.03	148.80
Reduce Shuffle Byte Total (GB)	0.17	50.07	0.19	55.07	0.21	61.08	0.22	66.09
Number of Mappers	187	200	205	220	224	244	243	264
Number of Reducers	1	115	1	115	1	115	1	115
Cluster CPUs (Systems) (%)	1.9	4.3	2.0	4.0	2.4	4.5	1.9	4.6
Cluster CPUs (User) (%)	49.6	8.2	47.9	9.3	50.1	9.0	44.5	9.8
Cluster CPU (wait) (%)	0.5	11.2	0.5	16.2	1.6	17.8	0.5	22.7
Memory (Use) GB	85.2	93.9	84.7	92.8	81.4	95.3	83.3	93.7
Memory (Cache) GB	68.3	75.3	69.1	74.4	70.1	70.1	67.2	39.6
Memory (Buffer) MB	1.0	10.3	1.0	12.3	1.0	14.1	1.0	46.2
System (Process)	226.8	69.9	246.8	58.7	231.8	81.0	252.9	77.0
Network (IN) (kbps)	9.6	274.4	8.6	431.6	11.4	340.8	9.0	406.3
Network (OUT) (kbps)	9.6	277.3	8.6	436.5	11.3	352.5	8.9	418.0
Elapsed Time in mins	10.6	36.6	10.4	28.0	12.8	32.3	14.1	30.8
Rack Relative Humidity (%)	50.0	50.0	51.0	50.0	50.0	50.0	46.0	50.0
Rack Temperature (ºC)	23.3	23.4	23.4	23.9	23.9	23.9	26.5	23.9
Mean Active Power (kW)	1.33	1.04	1.42	1.04	1.38	1.03	1.46	1.05
Cumulative Energy (kWh)	0.26	0.63	0.25	0.48	0.27	0.56	0.34	0.54

**WC is Wordcount job, ^TS is Terasort job.*

In comparison, energy consumption for the Terasort application generally doubled that of the Wordcount application for the same payload. I/O activity counters such as *reduce shuffle byte*, the *number of mappers*, and the *number of reducers* were high. The *CPU time spent* counter was half the Wordcount application and did not increase with payload size. The *cluster CPU (user)* counter was low, with a mean of around 9.08%. The *cluster CPU (wait)* counter increased with file size, and the *system (process)* counter was comparatively lower and generally stable. The *memory use* counter was comparatively higher, and the *network (IN/OUT)* counters were high in both directions. These observations suggest that the Terasort application is light on CPU demand, high in I/O, and high in network traffic; hence,

the Terasort application is I/O-intensive. Table 5.5 summarizes the feature analysis and characterization of the different MapReduce workloads and their impact on energy consumption.

Table 5.5 Analysis and characterization of the WordCount and Terasort workloads.

Features	**Wordcount Application**	**Terasort Application**
Energy consumption	Increases with file size	Constant
Mapper I/O activities	Low	High
Reducer I/O activities	Low	High
Shuffle activities	Low	High
CPU time spent	Increases with file size	Constant
Number of mappers	Increases with file size	Increases with file size
Number of reducers	Low, constant	High, constant
CPU (user) – used by applications	High	Low
CPU (system) - used by the kernel	Very low	Low
CPU (wait) - waiting for I/O	Very low	Low
Memory (use)	High	High
Memory (cache)	High	High
Memory (buffer)	Very low	Very low, can be unpredictable
System processes	High	Moderate
Network traffic	Low	High
Elapsed time	Low	High

To further explore the characteristics of the energy-related features, the plots of these features for Terasort and Wordcount workloads for payloads from 50 GB to 65 GB at one GB increment. The plots consist of CPU, Memory, Network utilization, Number of System Processes, Temperature, Humidity, I/O activities (Number of Mappers and Reducers), CPU time spent, Elapsed Time, and Energy Consumption. It can be observed from Figure 5.8 that the energy consumption for Terasort workloads was higher than Wordcount workloads. This observation is due to the longer elapsed time taken by Terasort workloads to complete the job for the same payload. However, when computed at a per-second level, Terasort and Wordcount workloads consumed an average of 17.52 kWh/s and 23.23 kWh/s, respectively. The analysis found evidence that compute-intensive workloads consumed higher energy than I/O intensive workloads due to higher *CPU (user)* utilization, *CPU time spent*, and *system (process)*.

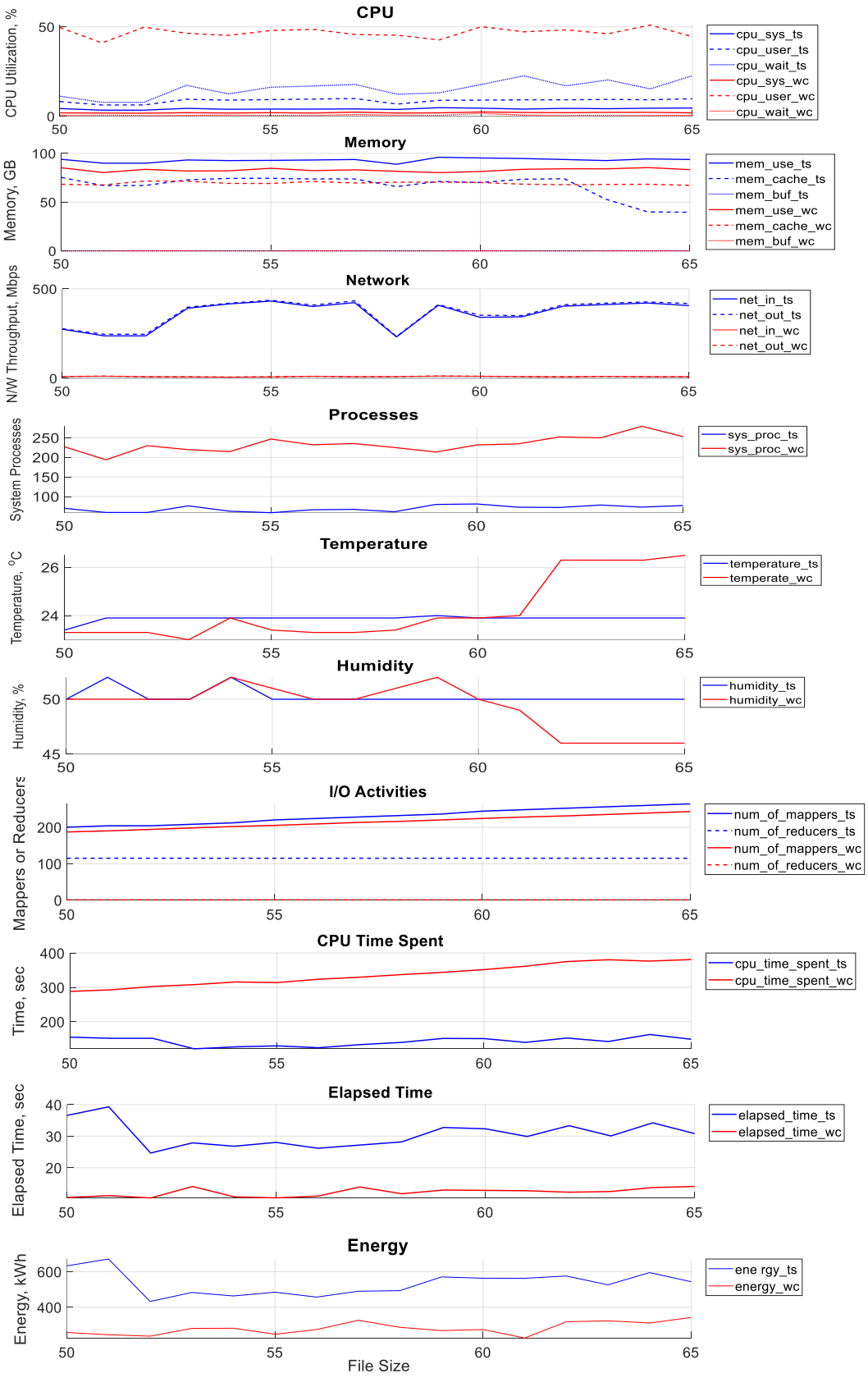


Figure 5.8 Energy-related features for MapReduce and Terasort workloads with various payload sizes.

5.4.3. Data Transformation and Normalization

The data acquired in their raw format was first transformed to their respective standard units, such as file size to Gb, network traffic to kilobits per second (kbps), and energy to kWh. After that, the data is normalized using min-max scaling as shown in Equation 5.3 to values between 0 and 1 before model training,

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{5.3}$$

where $x = (x_1, \ldots x_n)$, $min(x)$, and $max(x)$ are the minimum and maximum values of variable x, respectively, and z_i is the normalized data of x_i in the i^{th} sample.

5.4.4. Model Training

The energy-related data acquired from executing the Wordcount and Terasort jobs, used for model training, is split into 70% training and 30% testing set. The model training was executed in 50 trials. The best individuals from each trial are identified by their lowest training MSE values. These 50 identified individuals are then applied to predict the output using the testing dataset. EVLNN's performance is computed by averaging the MSE results of the training and testing dataset obtained in the 50 trials. Then, the results are compared to the MSE results produced by models trained using PSO-NN, DE-NN, and GA-NN.

The EVLNN model is shown in Figure 5.9. It consists of 23 input features, and the output to predict is the energy consumption.

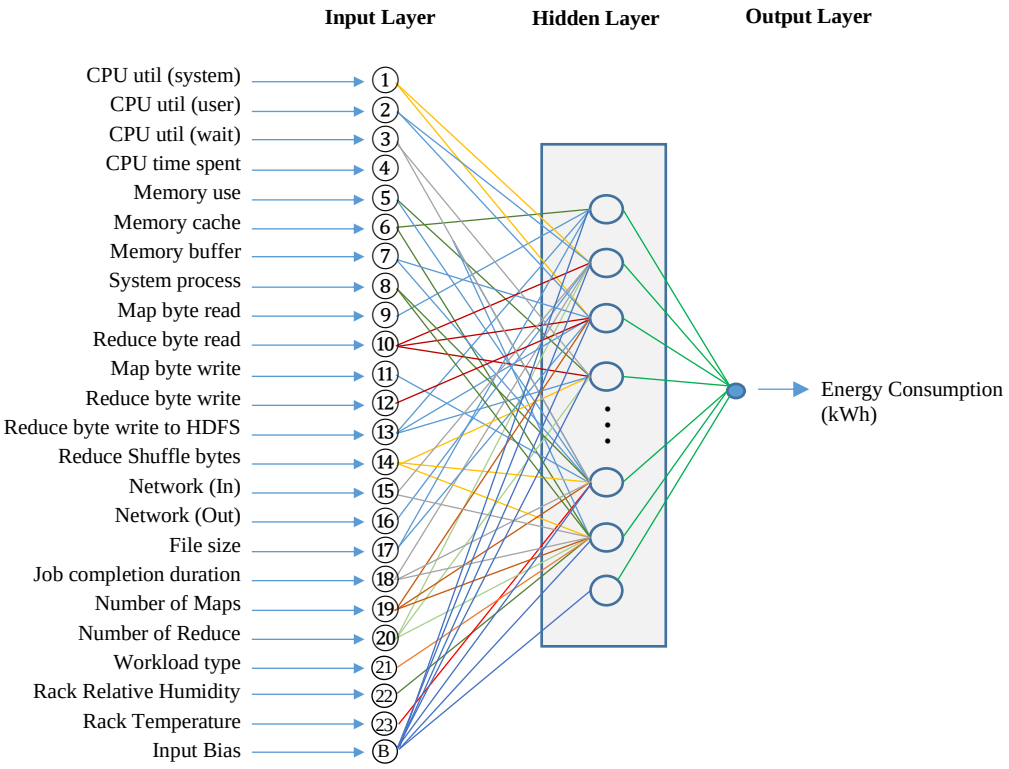


Figure 5.9 The EVLNN model with 23 input variables and one response variable.

The hyperparameter settings for EVLNN are given in Table 5.6, with its genetic operators shown in Table 5.7. The EVLNN search algorithm is applied to identify an optimal parsimonious ANN for Hadoop energy consumption prediction.

Table 5.6 EVLNN's hyperparameter settings.

EVLNN's Hyper Parameter	**Description**
Input layer neuron	23
Hidden layer neuron	1 to 26
Output layer neuron	1
Hidden layer activation function	Sigmoid function
Output layer activation function	Pureline
Network type	Feedforward MLP
Network connections	Partially connected

Table 5.7 EVLNN's operators and values.

EVLNN Operators	**Value**
Population size	100
Max generation	100
Intra-species crossover probability	0.9
Inter-species crossover probability	0.01
Mutation probability	0.01
Link-node mutation probability	0.01
Weights Mutation range	-0.5 to 0.5
Replacement probability	0.05

At initialization, a population of 100 individuals is created and subsequently speciated. Figure 5.10 shows a normal species distribution at initialization, where the horizontal axis represents species *ID,* and the vertical axis represents the size of the species. For example, *Species_13* has the highest number of individuals at the start with 23 individuals, and *Species_8* and *Species_19* have only one individual each.

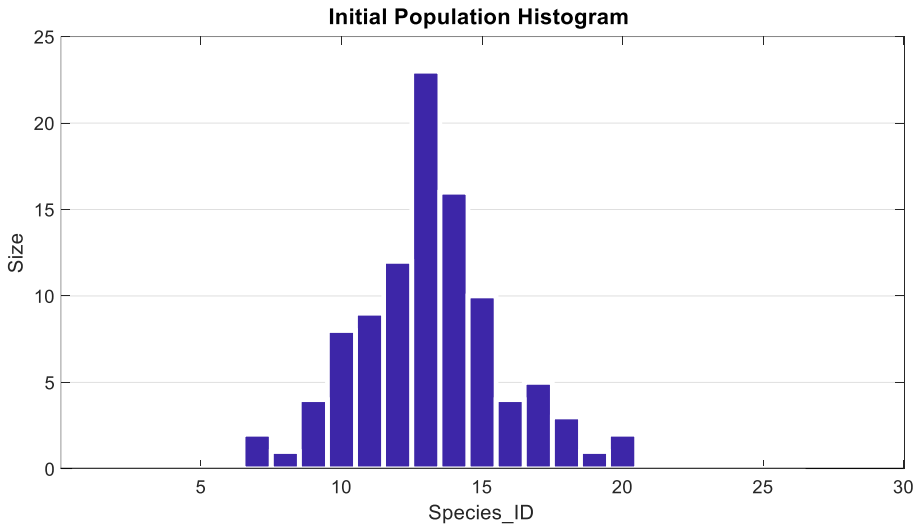


Figure 5.10 Species distribution at population initialization.

As crossover and mutation occur through the EVLNN process to search for optimal parsimonious structures, the species growth pattern also changes, as seen in Figure 5.11. This changing pattern is depicted by the varying height of each bar in the population histogram captured from generation 20 to generation 100 at an interval of 20 generations.

For example, at the 20th generation, most species have around the same number, but *Species_7, Species*_8, and Species*_10* have extinct, and a new *Species_22* has spawned. In the 40th generation, the population is concentrated around *Species_13*, and another new *Species_21* has spawned. In the 60th generation, there was a shift in species concentration from *Species_12* to *Species_14*. This trend continued until the 100th generation, with slight movement in between. The varying species size and the emergence and disappearance of species highlighted the effects of speciation, crossovers, and mutations of EVLNN in a continual search for the fittest landscape.

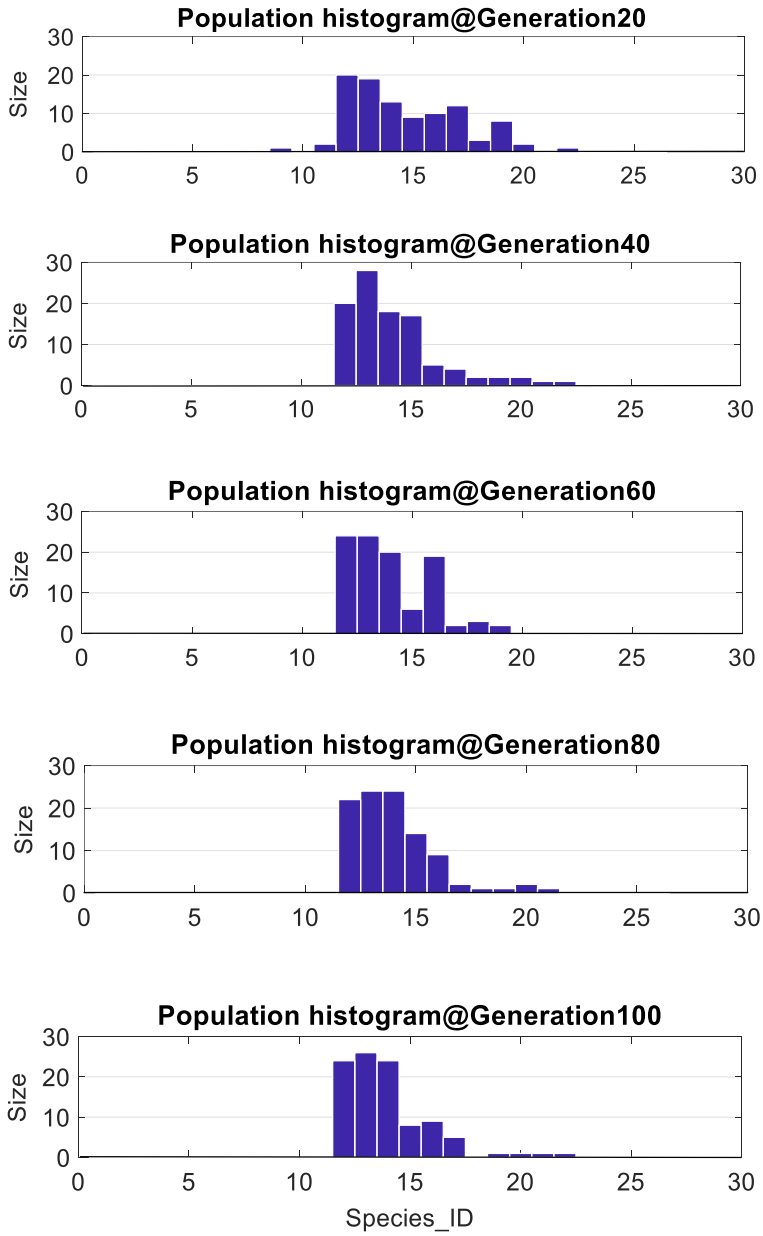


Figure 5.11 Histogram showing the species distribution for the EVLNN population from 20th generation to 100th generation at intervals of 20 generations.

Figure 5.12 examines the growth pattern of each species in the EVLNN population in detail. The diagram reflects the growth and shrinks trend among species 1 to 26 during the evolutionary search for the optimal global structure. As more individuals with higher fitness evolve in that species, the species size grows at the expense of other species as the total population remains constant. This phenomenon can be observed from *Species_12* to *Species_16*. *Species_12* to *Species_14* eventually grew to an average size of 20, and

Species_15 and *Species_16* to ten. Generally, optimal solutions would come from species with a larger subpopulation. A declining trend is observed for *Species_17* to *Species_19,* starting from around the 10th generation. These species ended in a very small population or extinction, signifying that the species' fitness could not compete globally. Similarly, *Species_7* to *Species_11* had a brief growth for around the first 20 generations. After that, these species could not thrive as they were being replaced with healthier individuals in the population. Almost a flat line trend is observed for *Species_20* to *Species_26,* highlighting zero or marginal growth, while *Species_1* to *Species_6* did not spawn any individuals.

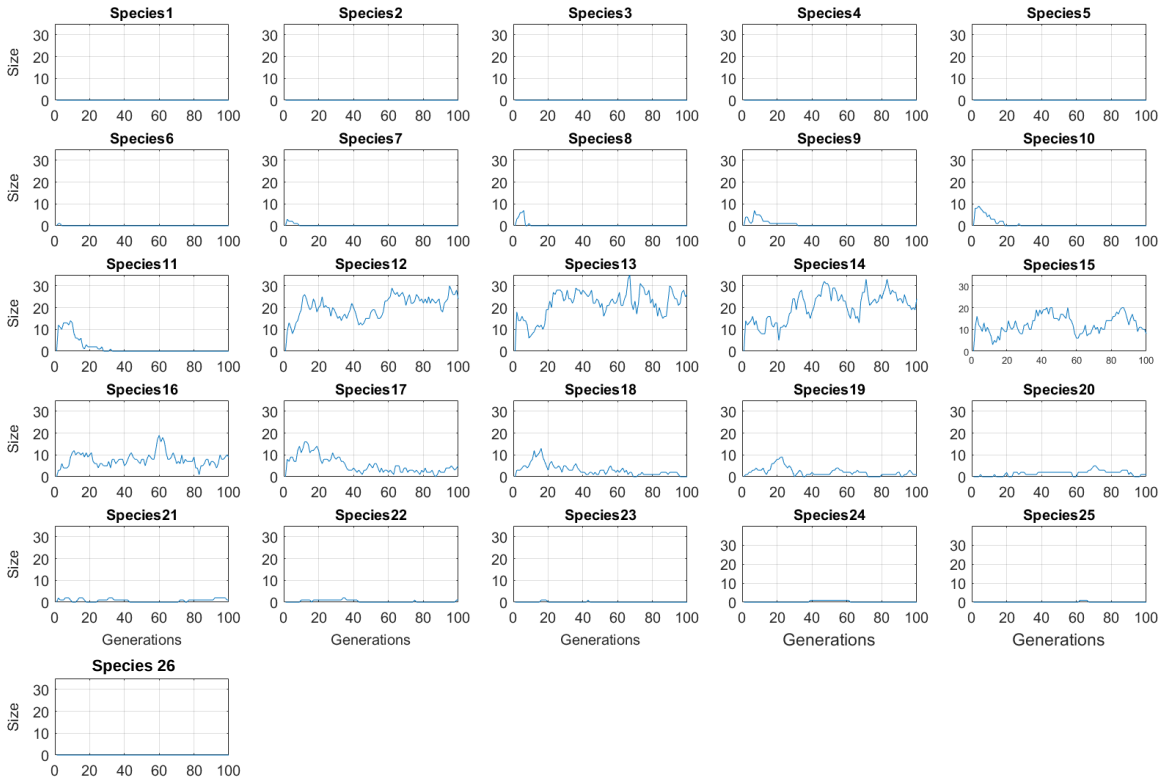


Figure 5.12 Species growth charts depicting their respective growth patterns.

The EVLNN algorithm implemented a diversity tracker with three indices, $pop_{diversity}$, $shannon_{diversity}$, and $shannon_{equitability}$, to track population diversity for insights into its search behavior. The $pop_{diversity}$ tracks how diverse the population is through individuals' phenotype structures, the $shannon_{diversity}$ tracks the species abundance, and the

$shannon_{equitability}$ tracks the species evenness or how close in numbers each species is. Figure 5.13 shows the chart of these three indices during one of the experiments.

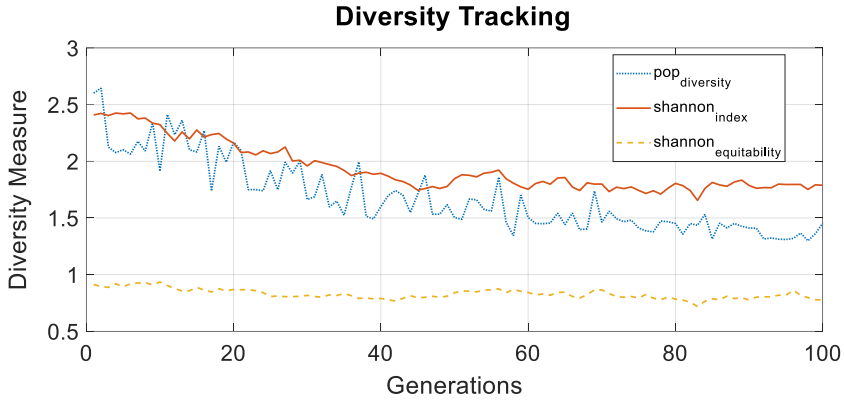


Figure 5.13 Tracking for solution diversity.

The observed constant variability for the $pop_{diversity}$ plot in Figure 5.13 implied that individuals' phenotype structure within the population is continually changing. The $pop_{diversity}$ value measured at 2.60 at the start and 1.45 at the end of 100 generations indicated a convergence trend as individuals within the population became more similar. The observed downward slope of the $shannon_{index}$ plot started at an initial value of 2.41 and ended at 1.79 after 100 generations. The gradual slope implied that the population had converged to fewer species than it started. This result is expected as species thrive in a good landscape towards convergence. In addition, the observed $shannon_{equitability}$ plot reduced from a value of 0.91 to 0.78. The reducing value signifies that species unevenness has decreased slightly. The results showed that while the abundance of species decreases towards convergence, EVLNN's species parallelism characteristics attempt to maintain species evenness during the search process.

Figure 5.14 shows the convergence of EVLNN. It is observed that the fittest individual was located in the 98th generation with a training MSE of 0.00164. The solution's architecture was a parsimony ANN with 12 hidden nodes and 243 connections from *Species_12*. The model was subsequently scored using the testing dataset, where the results are discussed in the next section.

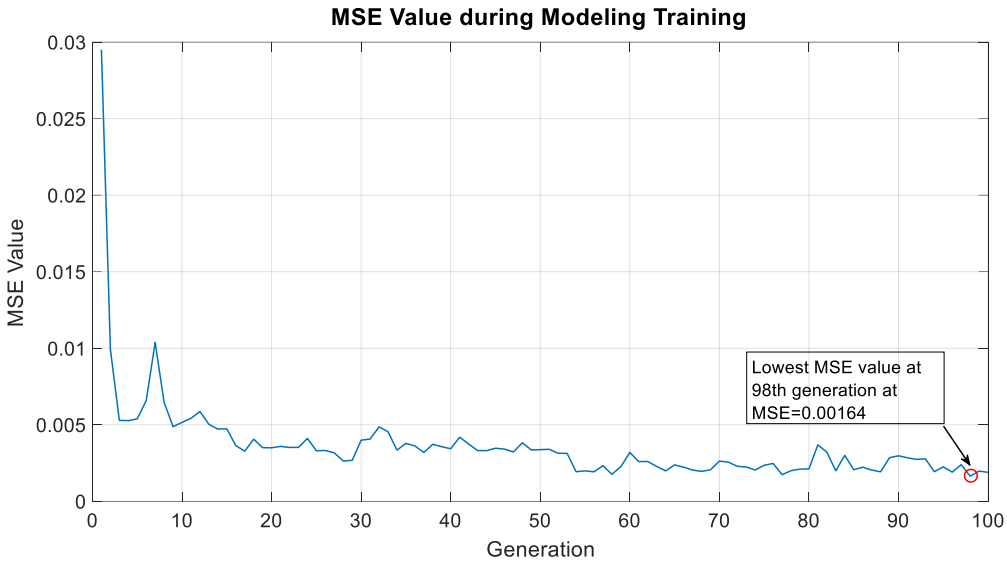


Figure 5.14 Convergence of EVLNN with the lowest MSE value of 0.00164 at the 98th generation.

5.5. Results and Discussion

5.5.1. Model Testing and Comparison

EVLNN was trained using the energy-related dataset acquired from executing Wordcount and Terasort workloads. Its performance is subsequently compared with those ANNs trained using other modern metaheuristic methods, Particle Swarm Optimization (PSO-NN), Differential Evolution (DE-NN), and the conventional Genetic Algorithm (GA-NN). The operators and values of these EA-based learning techniques are shown in Table 5.8(a-c). The population size, maximum number of iterations, and the hyperparameters of the EA-based networks were kept the same as EVLNN's.

Table 5.8(a-c) EA-based ANN with their learning techniques, operators, and values.

(a)

Particle Swarm Optimization

Operators	Values
Population size	100
Max Iterations	100
C0	0.1
C1	1.5
C2	2.5

(b)

Differential Evolution

Operators	Values
Population size	100
Max Iterations	250
Upper bound	1
Lower bound	-1

(c)

Classic Genetic Algorithm

Operators	Values
Population size	100
Max Generation	250
Crossover probability	0.8
Mutation probability	0.01
Link-node mutation probability	0.01
Weights Mutation range	-0.5 to 0.5

Table 5.9 presents the MSE results of EVLNN, PSO-NN, DE-NN, and GA-NN. The training and testing MSE scores are computed by taking the best MSE results obtained in each of the 50 trials and computing the mean, respectively. While PSO-NN's training error was lower than EVLNN, its testing error was much higher. It also had a higher standard deviation for the testing MSE. One possible reason could be that PSO-NN was overfitting the training dataset. As such, it could not generalize to the new dataset resulting in a much higher MSE for the testing dataset. EVLNN's average testing MSE score of 0.00230 was superior to PSO-NN, DE-NN, and GA-NN testing MSE scores of 0.00310, 0.01041, and 0.01071, respectively. In another observation, the higher standard deviation values of all the other models, except EVLNN, indicated that these models suffer from a higher spread of MSE values from the mean. The results showed that EVLNN had higher accuracy with better consistency when predicting new unseen data.

Table 5.9 Comparing the training and testing, MSE scores averaged over 50 runs.

Models	Training MSE Score		Testing MSE Score	
	Mean	Std Dev	Mean	Std Dev
EVLNN	0.00180	0.00038	**0.00230**	**0.00042**
PSO-NN	**0.00146**	**0.00041**	0.00310	0.00195
DE-NN	0.01015	0.00287	0.01041	0.00307
GA-NN	0.01060	0.00441	0.01071	0.00416

Figure 5.15 shows the EVLNN's prediction output plotted against the target output. It can be seen that the predicted plot tracked the target plot indicating that the trained EVLNN generalized well to the existing system. The results confirm the findings that EVLNN is capable of accurately predicting the Hadoop energy consumption.

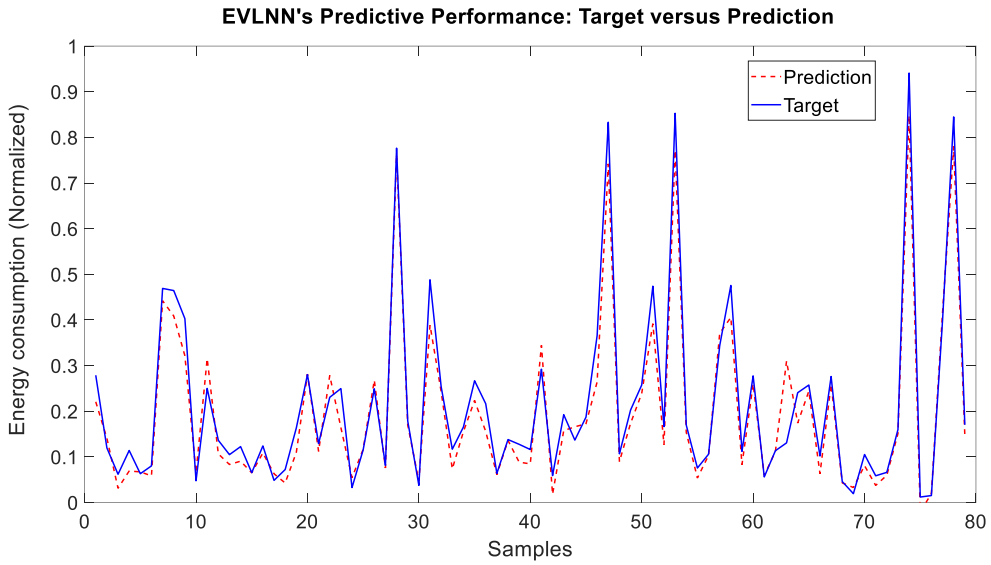


Figure 5.15 EVLNN's Energy consumption prediction for the Hadoop testbed.

5.5.2. Model Convergence Characteristics

The convergence characteristics of EVLNN were compared with those from PSO-NN, DE-NN, and GA-NN, as shown in Figure 5.16. From the figure, PSO-NN, DE-NN, and GA-NN displayed similar convergence patterns whereby long periods of stagnation (or plateaus) existed during the evolutionary search process before a better solution was found.

The plateaus signify that these algorithms may be trapped in deceptive local minima and could not find 'a way out' to better solutions for an extended period. In EVLNN's case, however, the convergence pattern showed a trend of decreasing MSE values without any period of stagnation. The results demonstrate two things. First, species parallelism enables EVLNN to diversify the search to different parts of the landscape. Second, EVLNN's two-

fold crossover strategies enable the algorithm to explore other basins of interest for optimal solutions reducing the chance of the individuals becoming trapped in local minima.

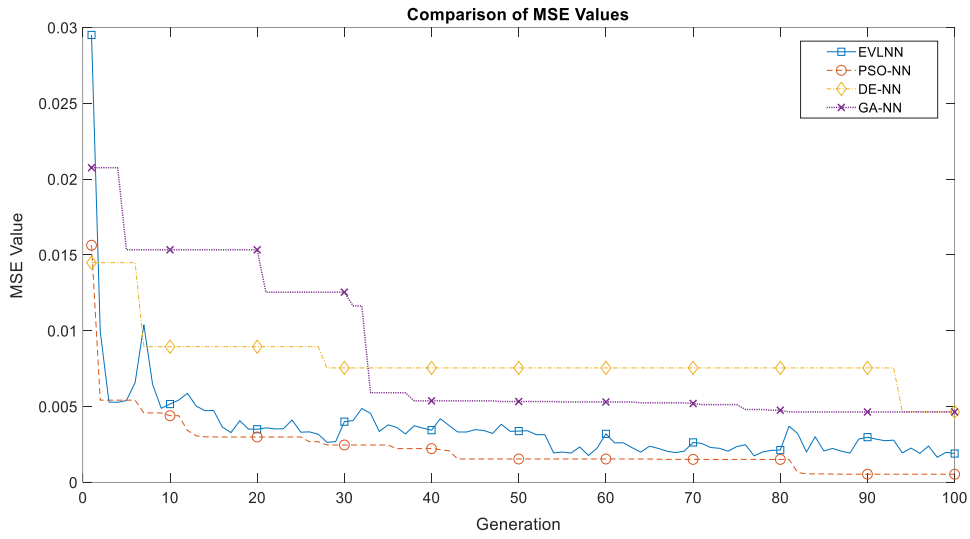


Figure 5.16 Convergence characteristics of EVLNN, PSO-NN, DE-NN, and GA-NN models.

5.5.3. Structural Comparison of the Identified Networks

The learned structures from EVLNN and the various EAs are shown in Table 5.10. Using a naïve method for assessing the physical complexity of network structures based on the number of hidden neurons and connections, DE-NN had the most complex structure, with an average of 20 hidden neurons and 348 connections. In contrast, GA-NN had the simplest, with an average of 11 hidden neurons and 149 connections. However, neither of these structures was able to predict the energy consumption accurately. EVLNN, with an average of 15 hidden neurons and 314 connections, had achieved the best performance reflected in the testing MSE score. Based on the results obtained, it could be inferred that networks with a higher number of hidden neurons. However, they could present additional representational power and have a downside as their excessive number of free parameters will increase the risk of overfitting. In the same context, networks that trim back excessively, leaving fewer hidden neurons, will adversely limit their predictive ability due to the insufficient capacity of the ANN. In another observation, it appears that the number of connections has a lesser impact on predictive accuracy. EVLNN and DE-NN had around

the same number of connections, but the performance of EVLNN was about 4.5 times better, with an average of five fewer hidden neurons. Finally, while DE-NN had the most complex structure and GA-NN had the simplest, they produced almost similar performance. This result verifies that different phenotypic representations can produce behaviorally equivalent ANNs [248].

Table 5.10 Comparison of the trained neural network structures averaged over 50 runs.

Models	**Number of Hidden Neurons**				**Number of Connections**			
	Min	**Max**	**Mean**	**SD**	**Min**	**Max**	**Mean**	**SD**
EVLNN	9	20	15	2	208	438	310	46
PSO-NN	8	20	13	3	97	219	153	28
DE-NN	12	25	20	3	158	538	348	86
GA-NN	8	15	11	2	95	216	149	28

The result agrees with the literature since networks with large neuronal structures could be counter-productive. The excessive hidden neurons will probably encourage each to memorize the relationships between input and output dataset, decreasing error on the training set but not necessarily generalizing well to new data on the testing set. Similarly, insufficient hidden neurons would lead to poor performance. However, it remains unclear to which degree the number of hidden neurons is 'excessive.' The findings in Table 5.10 suggest that EVLNN had outperformed the other models. EVLNN delivers significantly better results due to its novel search approach combining speciation and crossover strategies and mutation to perform search parallelly from multiple hidden nodes, emphasizing architecture search rather than just evolving its behavior. The stochastic search nature of EVLNN has few constraints on the species to evolve ANN of different numbers of hidden nodes within the solution space. EVLNN is a promising algorithm for optimizing neural network architecture that can reduce human intervention in the training process. Neural network design faces a vast search space of different ANN architectures. EVLNN could effectively explore the architectural landscape and discover optimal structures that generalize well, minimizing the need for expert knowledge and time-consuming trial-and-error effort.

5.5.4. Ensemble-based Sensitivity Analysis Approach to Determine Input Variable Importance

An ensemble-based approach to SA was applied to investigate and interpret the contributing factors to energy consumption. The goal of this approach is to achieve stability in model interpretation. The details of the ensemble SA method, consisting of the Connection Weights (CW) method [188], the Pad method [192], the Perturb method [198], and the Profile method [200] and their pseudo-codes, are explained in detail in Appendix B.

The CW method calculates the relative importance of the inputs to the neural network output (see Equations B.17 and B.18). The method is based on the concept that the neurons' output depends on the input neurons' contributions subjected to the connection weights' magnitude and direction. Inputs with higher connection weights represent a higher excitation level of activation at the output of the neurons. They, therefore, are relatively more important in predicting than inputs with lower connection weights.

The PaD method calculates the relative contribution of the neural network outputs using the Sum of the Square Partial Derivatives (SSD) (see Equation B.14). The method is based on the Backpropagation (BP) algorithm, which is used to compute the partial derivatives of the cost function with respect to each weight.

The Perturb method computes the relative importance of each variable by predicting the output at the network by progressively applying white noise to each input variable while keeping the other variables constant. The predicted output is subsequently used to calculate the new MSE, which is then compared to the original MSE. The relatively more important variable is expected to significantly influence the network's output, exhibited by a more significant difference between the MSEs.

The Profile method calculates the relative importance of each input variable by varying the values of the input variable while the remaining input variables are kept constant. The input values change with a scale range fixed with an initial setting to their minimum value, the first quartile, median, third quartile, and maximum values, resulting in five output values. The median values over the scale range are identified to obtain a profile curve. The

difference between the maximum and minimum values from the profile curve is computed. The more significant the difference, the more influence that input variable has on the output variable.

The 23 input features (See Table 5.11) are mapped according to their affinity into five major energy-related categories, as shown in Table 5.12. These categories are System Utilization, Disk I/O Activities, Network Transfer, Job Profile, and the Environment. The ensemble SA method is applied to the 50 identified EVLNN models from the 50 runs of EVLNN training. These are the best individuals or optimal solutions from each of the runs. The relative importance of each input variable for each EVLNN model is calculated and stored in a matrix.

Table 5.11 Input Variables of the EVLNN model.

Input Number	Variables
1	Cluster CPUs (Systems)
2	Cluster CPUs (User)
3	Cluster CPUs (wait)
4	CPU Time Spent
5	Memory (Use)
6	Memory (Cache)
7	Memory (Buffer)
8	System (Process)
9	File: Map byte read
10	File: Reduce byte read
11	FILE: Map byte written
12	File: Reduce byte written
13	HDFS: Reduce byte written
14	Reduce Shuffle Bytes (Total)
15	Network (IN)
16	Network (OUT)
17	File Size
18	Job Completion Time
19	Number of mappers
20	Number of reducers
21	Workload type
22	Humidity
23	Temperature

The importance values were subsequently transformed into importance orders and averaged over the 50 models in each SA method. Table 5.13 shows the orders; however, no consensus among the four methods was observed at this stage. For example, the CW and Profile methods ranked input variable 21 (*workload type*) as the most important contributor to

energy consumption. The PaD and Perturb methods ranked input variable 17 (*file size*) as the most important. Comparatively, the input variable 17 (*file size*) was ranked second and third in the Profile and CW methods, respectively. In contrast, input variable 21 (*workload type*) was ranked fourth in the Perturb method but 19th in the PaD method.

Table 5.12 Five energy-related categories (differentiated by their respective colors).

	Categories
1	Job Profile
2	System Utilization
3	Disk I/O
4	Network Transfer
5	Environment

Table 5.13 Ranking of input variable importance averaged over 50 identified EVLNN models.

Rank	**Sensitivity Analysis Method**			
	CW	**PaD**	**Perturb**	**Profile**
1	21	17	17	21
2	5	2	18	17
3	17	7	4	9
4	19	9	21	4
5	1	18	2	18
6	20	11	14	10
7	16	19	3	13
8	12	15	12	14
9	2	6	19	11
10	14	4	20	12
11	10	22	13	1
12	8	1	10	19
13	7	23	11	20
14	11	5	5	2
15	3	20	1	6
16	4	8	6	16
17	23	14	8	7
18	18	12	9	3
19	13	21	16	22
20	6	16	15	5
21	15	13	7	15
22	22	3	23	8
23	9	10	22	23

Next, the importance orders were transferred to the categories, shown in Table 5.14. It can be observed that the method has resulted in a more consistent model interpretation in which all four SA methods agreed with the ranking of *Job Profile* as the most critical factor

contributing to energy consumption. The *Job Profile* category, as presented in Table 5.14, consists of the input features: *file size*, *workload type*, *job completion time*, the *number of mappers, and reducers*. If there is a tie, the amount of voting positions is adopted to break the tie (See Table 5.15). For example, in the case of position two, System Utilization and Disk I/O categories garnered two votes each for position two, which indicated a tie. Then the amount of voting for the following position is considered to break the tie. Thus, in the final result, the order of importance was Job Profile, followed by System Utilization (most votes for positions two and three), Disk I/O, Network , and finally the Environment. The resulting descriptions are interpretable, providing insight into the system of interest. While it is expected that the categories were not all ranked the same by all four methods, this approach has exhibited good stability by dramatically reducing the inconsistency.

Table 5.14 Ranking of factors contributing to energy consumption by categories.

Rank	CW	PaD	Perturb	Profile
1	1	1	1	1
2	2	2	3	3
3	4	5	2	2
4	3	4	4	4
5	5	3	5	5

Table 5.15 Amount of votes received by each category.

		Voting Count					
	Categories	**Position 1**	**Position 2**	**Postion 3**	**Position 4**	**Position 5**	**Final Position**
1	Job Profile	4	0	0	0	0	1
2	System Utilization	0	2	2	0	0	2
3	Disk I/O	0	2	0	1	1	3
4	Network Transfer	0	0	1	3	0	4
5	Environment	0	0	1	0	3	5

5.6. Chapter Summary

The Hadoop cluster is a parallel distributed system with dynamic and nonlinear behavior. The energy consumption of a Hadoop cluster using EVLNN was successfully modeled. The performance of EVLNN is better than ANNs evolved using other modern state-of-the-art metaheuristics EAs such as PSO-NN, DE-NN, and conventional GA-NN. The testing MSE scores for EVLNN, PSO-NN, DE-NN, and GA-NN were 0.00230 ($\pm$0.00042), 0.00310 ($\pm$0.00195), 0.01041 ($\pm$0.00307), and 0.01071 ($\pm$0.00416), respectively. The experimental results showed that EVLNN could effectively explore the architecture landscape and discover optimal structures using several mechanisms of an improved GA, such as species parallelism, intra-and-inter species crossovers, and a two-stage mutation. In addition, the EVLNN architecture with reduced network complexity was interpretable, using an ensemble-based approach to sensitivity analysis combined with a data aggregation technique. The approach was successfully implemented to extract the underlying energy characteristics of the system by informing how various input parameters affect the system's output. The results highlighted that the most important contribution to the Hadoop energy consumption is the *Job Profile* category. This category consists of the *workload types (compute-intensive or I/O intensive)*, *file size (size of the payload)*, *job completion time*, and *the number of mappers (which is dependent on the payload size and the HDFS block size) and reducers*. This aspect of the research suggests that EVLNN is a competitive and promising method for energy prediction. EVLNN exhibited good generalizability and interpretability. Its ensemble-based approach to sensitivity analysis has produced a more consistent interpretation of the energy consumption influencing factors, which is essential in contributing to the research in data center sustainability.

Chapter 6

6. Solar Irradiance Forecasting in Tropical Region

6.1. Introduction

Clean electricity system based on solar photovoltaic (PV) power generation is rapidly growing worldwide. Cumulative solar PV capacity reached almost 400 Gigawatt (GW) and generated over 460 Terawatt hours (TWh) in 2017 [249]. This capacity represents around 2% of global power output, and by 2023, the world will have one trillion watts of installed solar PV capacity. Solar power intermittency remains a significant issue for data centers' transition to renewable energy despite its rapid penetration. Solar power intermittency can be due to two factors. Firstly, the "variability" of solar power generation is caused by fluctuations in solar radiation from changing cloud conditions during the day. Secondly, the "uncertainty" of the electricity generation is due to grid operators not knowing the electricity production at multiple timescales with perfect accuracy. Therefore, integrating solar electricity into the data center electricity grid will be particularly challenging as the variability of solar resources means solar power generation is not guaranteed. The solar irradiance forecast data can be used to calculate the PV power output, assisting grid operators in determining the power output capacity. Hence accurate forecasting cum better planning can mitigate the effects of variability and uncertainty associated with intermittency and is essential for further integration of renewable energy into the grid [250].

In this chapter, EVLNN was applied to forecast time series solar irradiance. The training dataset covers a four-year database between 2013 to 2016 consisting of meteorological variables. A subset of data was extracted from the database to train the EVLNN model to predict solar irradiance over a forecast horizon of seven days at four different time steps: 1-min, 15-min, 30-min, and hour. The performance of EVLNN is compared to well-known EA-based neural networks modeled with Particle Swarm Optimization (PSO-NN),

Differential Evolution (DE-NN), and the classic Genetic Algorithm (GA-NN). A fully connected nonlinear time-delay neural network (TD-BPNN) trained using Levenberg-Marquardt (LM) backpropagation (BP) algorithm is included as a reference to evaluate the performance of the models.

6.2. The Solar Photovoltiac Testbed

A small-scale 1 kW PV system grid-connected Solar PV testbed located on the rooftop of the School of Engineering at Nanyang Polytechnic was used for the experiment. The experimental dataset covers the period from 2013 to 2016.

6.2.1. Experimental Testbed

The PV system testbed is mounted at 1.38° N 103.85° E, 40 m above ground level. The solar panels are tilted at approximately 15° to the horizontal plane to optimize the energy harvest from the equatorial sunshine. It also prevents rainwater from being trapped by the panel frame, which could cause dirt deposited on the panel after evaporation. Weather instruments are also installed, along with a data logger that collects and records data at 15-second intervals. Table 6.1 summarizes the information on the Solar PV testbed setup.

Table 6.1 Solar PV test panel and location information.

Parameters	**Specifications**
Site Name	School of Engineering, Nanyang Polytechnic
Site Latitude	1.38^{o} N
Site Longitude	103.85^{o} E
Solar PV Panel (W_p)	1 kW
Solar Panel Type	MultiCrystal PV modules
Solar Panel Efficiency	16%
Panel Tilt Angle	15^{o}

Figure 6.1 shows the PV panels and the associated meteorological measurement instruments. These instruments include the pyrometer, wind vane, anemometer, barometer, rain gauge, temperature sensor, and humidity sensors. The schematic diagram of the PV monitoring system is shown in Figure 6.2.

Figure 6.1 Left: Meteorological Measuring Instruments for rainfall, wind direction, and wind speed. Right: Rooftop solar panel testbed.

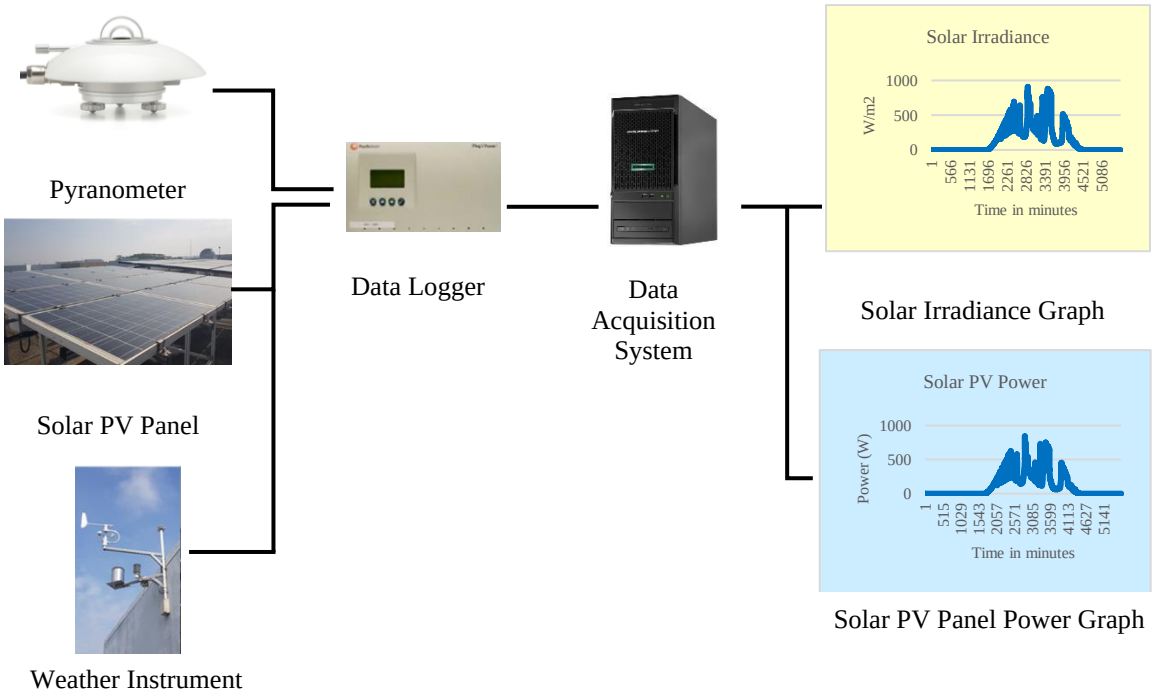


Figure 6.2 Schematic diagram of the PV monitoring system.

6.3. Data Preparation

6.3.1. Initial Data Exploration

Figure 6.3 shows a box plot of the distribution of daily irradiance observed at the PV location for each month in 2016. It was noted that the median daily irradiation was the lowest in June, recorded at 2.95 kWh/m^2, whereas the median daily irradiation was the highest in March, recorded at 4.36 kWh/m^2. February and September were months with high daily irradiation too.

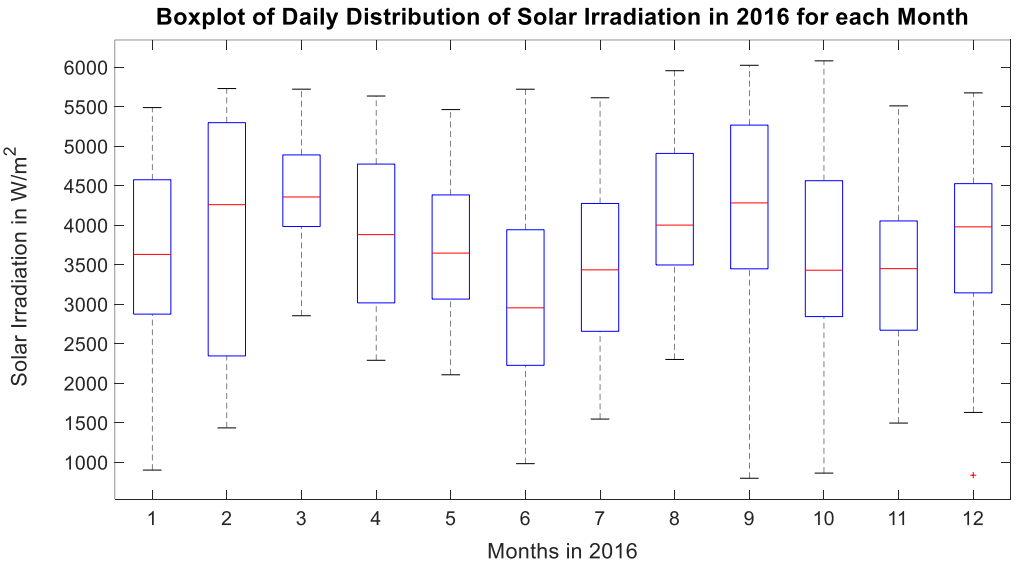


Figure 6.3 Distribution of daily irradiance in 2016 for each month at latitudes 1.38ºN and 103.85ºE.

A scatter plot in Figure 6.4 shows high irradiance variability throughout the year. This phenomenon is expected as the tropical region experiences frequent cloud formation and unexpected weather changes. The annual daily average irradiation in 2016 was about 3.78 kWh/m^2 ($\pm$ 1.15 kWh/m^2), and the total annual irradiation was 1,383 kWh/m^2. Figure 6.5 shows a box plot of the hourly irradiance distributed throughout 2016. The hourly mean irradiance is shown in a red curve. The chart depicts that irradiance is highly correlated to the hours of the day. A typical day follows the pattern where it starts with a low GHI, reaches the peak value around solar noon, and finally descends to a low again in the evening.

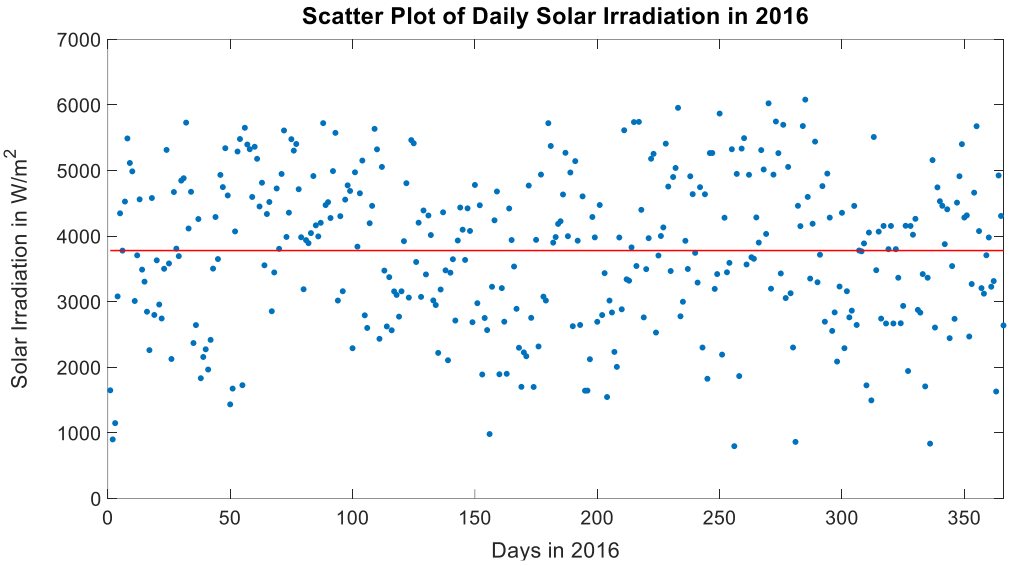


Figure 6.4 Scatter plot of the daily irradiance with the red line indicating the mean irradiance in 2016.

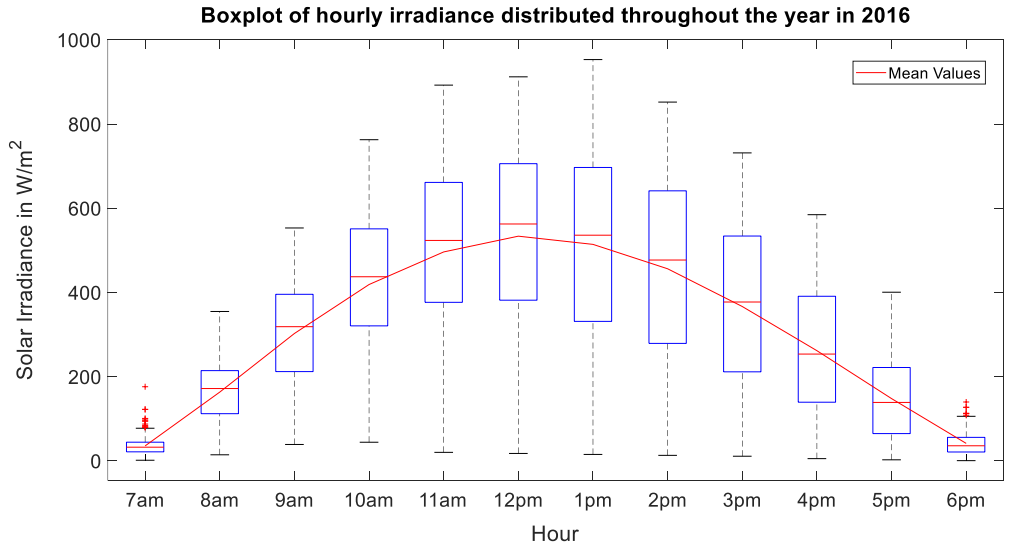


Figure 6.5 Hourly irradiance distribution in 2016 with the red curve indicating the mean.

To further explore and analyze irradiance variability, a typical day was chosen on 1st March 2016, where the clear sky GHI, DHI, and DNI values are plotted at the exact location (1.38°N, 103.85°E) using the McClear Clear Sky Model from the Copernicus Atmosphere Monitoring Service (CAMS) McClear Clear-Sky Irradiation service [154]. Figure 6.6 shows the GHI, DHI, and DNI on 1st March plotted in red, magenta, and green, respectively. The per-minute solar irradiance on 1st March is plotted in blue. From Figure 6.6, it can be seen that solar irradiance in the tropics is generally associated with high variability in the

time series, where high irradiance values were often preceded and followed by large drops in irradiance that resulted in a large magnitude of the fluctuation.

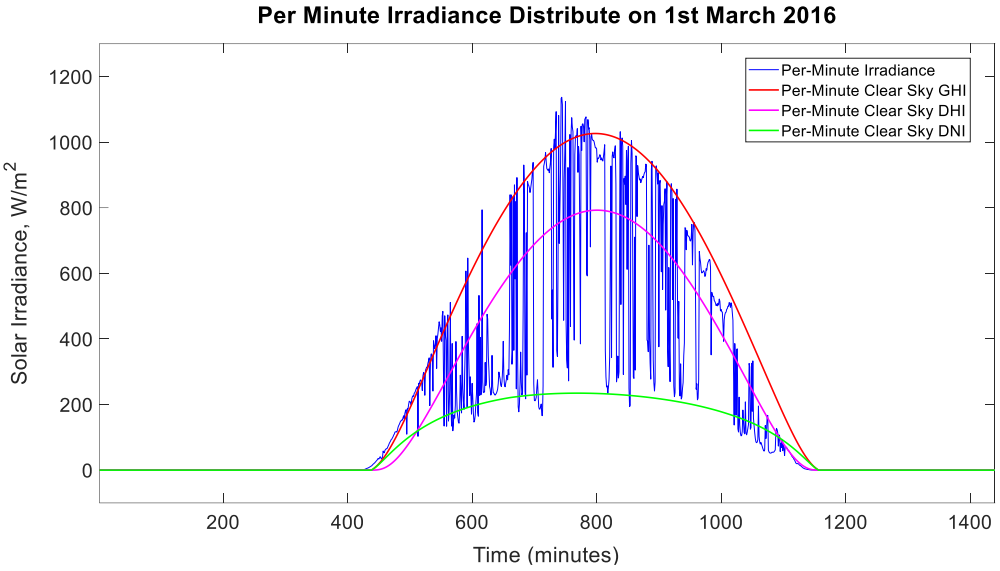


Figure 6.6 Irradiance profile on 1st March 2016.

6.3.2. Selecting Training Data Length

After the initial data exploration from our database, the next step is to decide the length of the training. For practical reasons, the best combination for the training set was *N=30* days, with fewer days proving insufficient and more days did not yield better results [251]. This approach is adopted where *N=30* days is used as a window period to explore the database further. The 2013 to 2016 database revealed that most days are similarly characterized by high irradiance variability with differing magnitudes on different days, except for those days with rain, where the irradiance patterns were different.

6.3.3. Determining Forecast Horizon and Time-Step for Predicting Solar Irradiance

A forecast horizon is how far into the future a sufficiently good prediction can be made. The concept of forecast horizon in solar can be categorized as long-term (1-10 years), medium-term (one month to one year), short-term (one hr or several hours to one day or several days), and very short-term (one min to several mins) [252] depending on the application. A forecast horizon of several days and an hourly time-step prediction are common in the literature [119], [253], [254]. In tropical regions, intra-hour and inter-hour

solar intermittency can be high caused by the temporal change of cloud structure influenced by cloud motion or frequent tropical rainfall. This work investigates the intermittency at four different time steps: 1-min, 15-min, 30-min, and hourly, with a forecast horizon of seven days. Based on findings during the initial data exploration stage, a forecast horizon of seven days would likely contain days with dry and wet weather conditions that can sufficiently provide an indication of how well the model generalizes to data into the future.

6.3.4. Pre-processing of Data

The dataset for 2016 was explored, and the training set in March 2016 was eventually selected. An important consideration when selecting a representative month is its good mix of weather conditions. Although February's daily solar irradiance distribution holds a good spread (see Figure 6.3), it only had one day of rain for the entire month, which may not be representative. The selected dataset has a 15-second time resolution. Figure 6.7 shows the time-series solar irradiance for March 2016, consisting of 178,560 data samples. The month experienced ten wet days of various rainfall intensity on the fourth, fifth, sixth, seventh, eighth, tenth, thirteenth, fourteenth, twentieth, and twenty-first day of March indicated by the rain gauge readings of 0.4mm/hr, 3.6mm/hr, 0.2mm/hr, 3.6mm/hr, 2.0mm/hr, 5.2mm/hr, 11.6mm/hr, 0.2mm/hr, 4.8mm/hr, 12.2mm/hr, respectively. Of the ten days of rain, four were light (< 2.5 mm/hour), three were moderate (> 2.5 mm/hour, < 7.5 mm/hour), and two were heavy (> 7.5 mm/hour), while the rest were dry days.

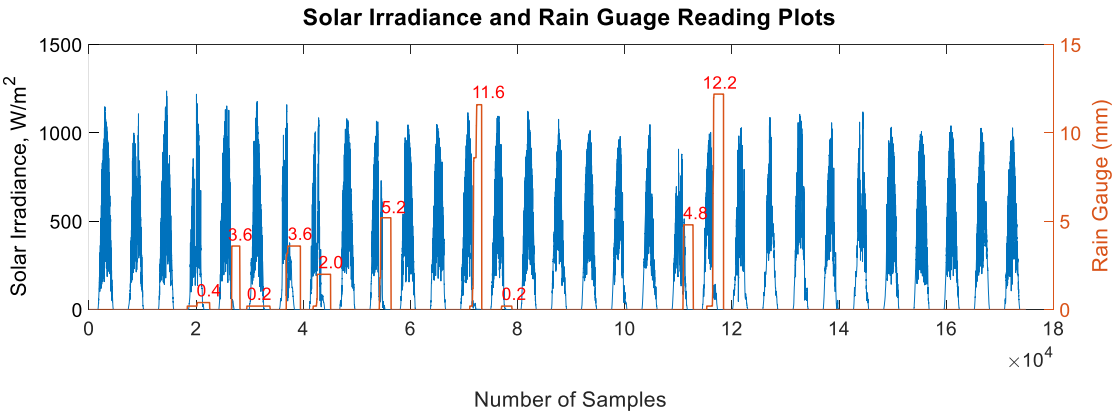


Figure 6.7 Solar irradiance raw data for March 2016 with rain gauge reading indicated.

Four training sets of 1-min, 15-min, 30-min, and hourly time resolution were extracted from this raw dataset to train the predictive models for short-term irradiance forecasting. The testing set is taken from the first seven days of April 2016.

Table 6.2 summarizes the descriptive statistics for the four training sets. It can be observed that the lower the time resolution, the higher the information lost. For example, the maximum irradiance of 1,237.67 W/m^2 sampled at an hourly resolution compared to the maximum irradiance of 1,066.90 W/m^2 sampled at the 1-min resolution has a reduction of 13.8%.

Table 6.2 Solar irradiance statistics at various time resolutions.

Training Set	Resolution	Minimum (W/m^2)	Maximum (W/m^2)	Mean (W/m^2)	Std Dev (W/m^2)	Training Samples	Testing Samples
1	1 min	0.29	1,237.67	183.84	275.59	43,710	9,870
2	15 min	0.32	1,113.97	182.72	273.69	2,914	658
3	30 min	0.33	1,079.04	185.31	277.43	1,457	329
4	1 hour	0.35	1,066.90	183.36	273.12	729	165

It is observed that the hourly dataset has a relatively small sample size of 729. Unlike large datasets, small datasets rendered most machine learning techniques impractical for predictive modeling as the lack of data makes it hard for models to map the input and output relationship in the dataset. Nonetheless, it is essential to investigate the performance of EVLNN trained with a small dataset of a few hundred samples. From an implementation viewpoint, reducing the reliance on extensive training datasets for solar irradiance forecasting has its advantages, as meteorological data are complex and expensive to acquire due to the high cost of weather and atmospheric measurement instrument.

Figure 6.8 shows the per minute irradiance testing dataset from 1st to 7th April 2016. Two of the seven days were rainy days. The first was measured at 43.2 mm/hr on 3rd April between 14:29 to 15:29, with relative humidity (RH) reaching 96.84%. After this time, the rain gauge measurement did not increase, but solar irradiance increased to a high of 276.7 W/m^2 at 18:00 before decreasing to almost zero at 19:13. The second rainy day was measured at 13.8 mm/hr on 5th April between 16:25 to 17:25, with RH reaching 80.21% during this period. The non-rainy days during this period exhibited the irradiance of a typical sunny day in the tropics with high variability.

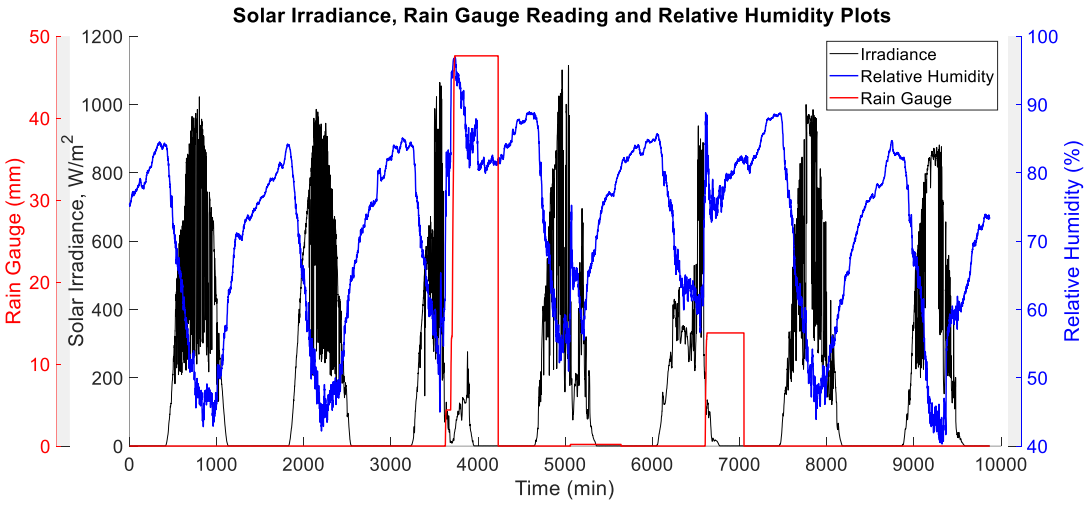


Figure 6.8 Irradiance testing dataset from 1st to 7th April 2016.

In summary, the testing dataset presents interesting variability and challenges with a mix of wet and dry days for solar irradiance forecasting.

6.3.5. Exploring and Selecting the Features

There are broadly three models of ANN in the literature; those using exogenous inputs [255] [256] [257], those using univariate endogenous input [258], and those using both [259]. Exogenous inputs usually include meteorological data such as temperature, humidity, rainfall, and wind speed, and endogenous inputs include solar irradiance, temperature, or PV output power. The exogenous time-series inputs features $\{x_1(t), x_2(t), \ldots, x_7(t),\}$ collected from the dataset for training the models are shown in Table 6.3 and the statistical description of the features in Table 6.4.

Table 6.3 Input features for EVLNN.

Input Features	Abbreviation
$x_1(t)$: Ambient Temperature at time t	AT
$x_2(t)$: Relative Humidity at time t	RH
$x_3(t)$: Rain Gauge reading at time t	RG
$x_4(t)$: Wind Speed at time t	WS
$x_5(t)$: Wind Direction at time t	WD
$x_6(t)$: Atmospheric Pressure at time t	AP
$x_7(t)$: PV panel surface Temperature at time t	PT

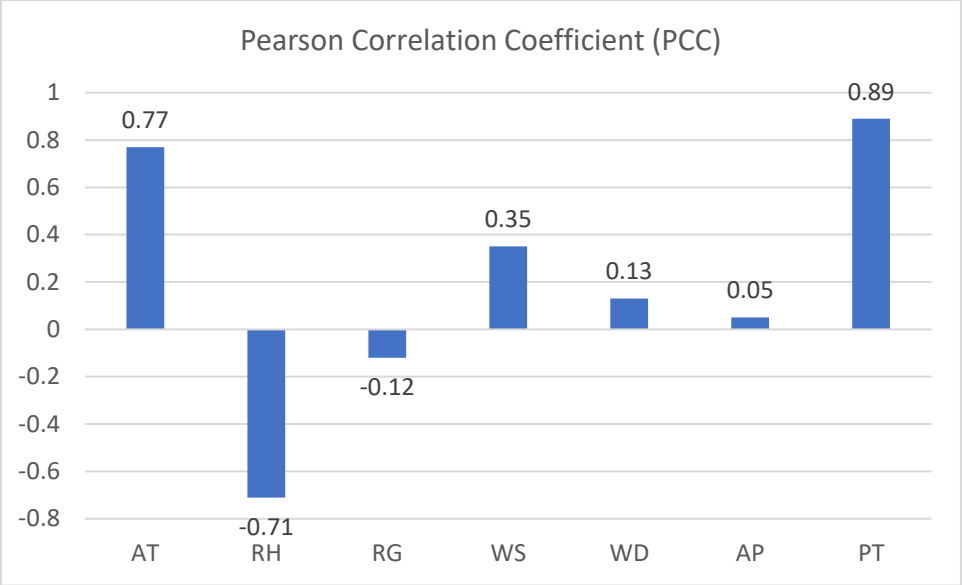
Table 6.4 Statistical Analysis of Input Feature Data Obtained for the Period in March 2016.

Input	Feature	Unit, Symbol	Range	Min	Max	Mean	Std Dev
$x_1(t)$	AT	Degree Celsius, ºC	12.4ºC	25.1ºC	37.5ºC	29.7ºC	2.5ºC
$x_2(t)$	RH	Percentage, %	55.8%	38.0%	93.8%	71.2%	11.8%
$x_3(t)$	RG	millimeter per hour, mm/h	12.2 mm/h	0 mm/h	12.0 mm/h	0.5 mm/h	1.8 mm/h
$x_4(t)$	WS	meter per second, m/s	8.0 m/s	0 m/s	8.0 m/s	0.9 m/s	± 0.7 m/s
$x_5(t)$	WD	Degree, º	360º	0º	360º	91.1º	± 101.5º
$x_6(t)$	AP	millibar, mbar	5.8 mbar	1008.0 mbar	1013.8 mbar	1010.9 mbar	± 0.9 mbar
$x_7(t)$	PT	Degree Celsius, ºC	33.3ºC	24.3ºC	57.6ºC	31.9ºC	± 7.3ºC

The Pearson Correlation Coefficient (PCC) method is applied to explore the correlation between the individual input variables with the solar irradiance response variable. The PCC measures the strength of a linear association between two variables, *x* and *y*, expressed in Equation 6.1.

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2(y-\bar{y})^2}} \tag{6.1}$$

Figure 6.9 shows the PCC result plotted in a comparison bar chart. *PT* and *AT* input variables have strong positive correlations with irradiance with 0.89 and 0.77, respectively. On the other hand, the input variable *RH* has a strong negative correlation with a value of -0.71. All the input variables are meteorological data except *PT*. Meteorological data are expensive to acquire as they require costly weather instruments. On the contrary, PV is a temperature sensor data that is inexpensive to acquire due to its affordability and low setup cost.



	AT	RH	RG	WS	WD	AP	PT
r	0.77	-0.71	-0.12	0.35	0.13	0.05	0.89

Figure 6.9 Pearson correlation between the input variables, AT, RH, RG, WS, WD, AP, and PT, and the response variable, solar irradiance.

6.3.6. Normalizing the Dataset

Data normalization, also known as feature scaling, is subsequently performed to ensure the input variables' magnitude is the same as inputs of different scale affects learning speed. Specifically, the min-max scaling is used to normalize the training set between the range of -1 to 1 governed by Equation 6.2, where x' is the transformed value of x and x_{min} and x_{max} is the minimum and maximum value of x in the dataset, respectively.

$$x' = 2 * \left[\frac{x - x_{min}}{x_{max} - x_{min}}\right] - 1 \qquad (6.2)$$

6.3.7. Preparing the Feature Sets for Model Training

The training of the models is conducted in two phases, as depicted in Table 6.5 and Figure 6.10. In the first phase, feature set one comprising seven input features, as listed in Table 6.5, is used. In the second phase, a smaller scale of features *AT*, *RH*, and *PT* has a stronger

correlation where $|PCC| \geq 0.7$ (Figure 6.9) was selected to form three subsets consisting of *AT*, *RH*, and *PT,* in subset one, *AT* and *PT* in subnet two, and *PT* in subset three. The combination of features in the subsets is based on the features' $|PCC|$ where the feature with a lower value is eliminated when reducing the feature size in the following subsets, eventually having only one feature in the last subnet. The four feature sets are then used to train the models to forecast solar irradiance at four different time-steps. Experimentation with various feature sets is designed to evaluate the generalization capability of EVLNN.

Table 6.5 Input and target features of phases 1 and 2 for model training.

Input features				**Target Feature for both PHASES 1 and 2**
PHASE 1	**PHASE 2**			
	Feature Subset 1	**Feature Subset 2**	**Feature Subset 3**	
AT	AT	AT	PT	
RH	RH	PT		
RG	PT			
WS				IRR
WD				
AP				
PT				

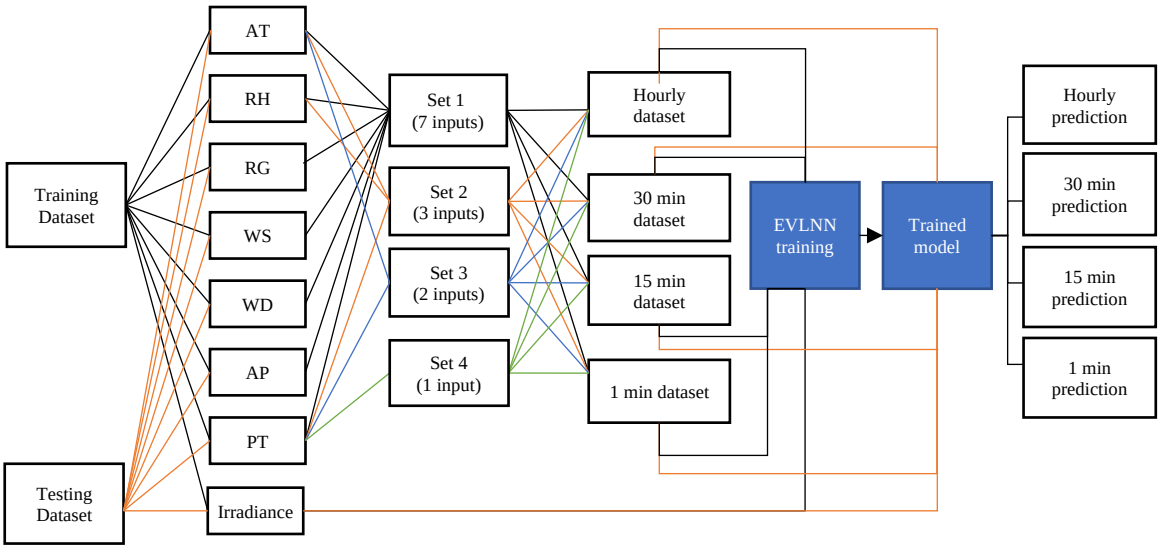


Figure 6.10 *F*our feature *sets* to train the EVLNN model for multiple time-step predictions.

6.4. Model Training

6.4.1 Designing EVLNN Architecture for Time-Series Forecasting

Figure 6.11 presents the EVLNN architecture with feedback for time series forecasting. The feedback structure allows lagged values, *t-h* at the output (where *h* is the number of time-steps), to be used as inputs to the network. The future values of a time-series response variable, y_k can be forecasted based on their past values. In the EVLNN architecture, the solar irradiance output at the previous time $y_{t-h}, \ldots, y_{t-3}, y_{t-2}, y_{t-1}$ is fed back as input to predict the solar irradiance output at the current time steps, $y_t, y_{t+1}, y_{t+2}, \ldots, y_{t+m}$ alongside the rest of the exogenous inputs, $x_{i(t-h)}$ where i=1, 2, …, 7. These input variables are described in Table 6.3.

In the model training, the lagged delay of two predicted time-step is taken and made available to the model for the forecast on the next step. Predicted values are cleansed of the noise at the network's output before feeding back to the inputs.

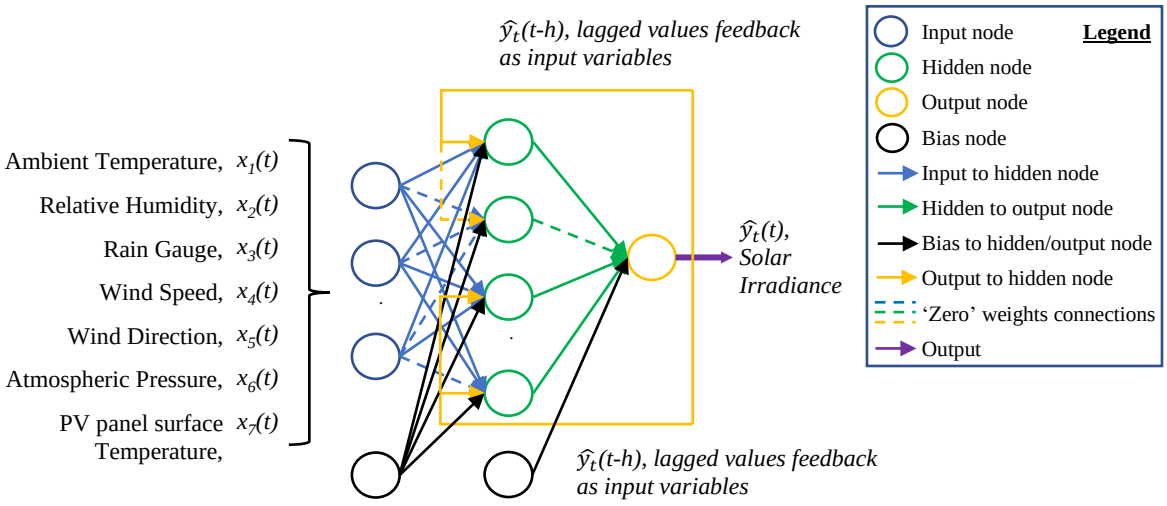


Figure 6.11 The figure illustrates four feature sets comprising various input features used to evaluate EVLNN's performance.

6.4.2 Error Metrics for Model Performance Comparison

A large number of error metrics have been used by authors to evaluate and compare the different methods and their solar energy forecast accuracies. As each statistical metric focuses on a specific aspect of point distribution, no unique metric is valid for all situations. Instead, researchers usually include several metrics to assess model performance as each one adds information about the model's accuracy [260]. Table 6.6 lists the widely used forecast error metrics and their respective descriptions and computations. Equations 6.3 to 6.7 in the list are Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Bias Error (MBE), Mean Absolute Percentage Error (MAPE), coefficient of determination or R-squared (R^2), respectively. The error performance of EVLNN is evaluated against ANNs trained with other EAs. The ANNs are PSO-NN, DE-NN, and GA-NN, and a fully connected Time-delay BPNN (TD-BPNN) trained using the Levenberg-Marquardt (LM) algorithm is used as a benchmark. In Table 6.6, the forecast error is expressed at time *t* as $e_t = y_t - \hat{y}_t$ where y_t *and* $\hat{y}_t$ are the actual and predicted values, respectively, n is the number of samples, and k is the number of predictors.

Table 6.6 Error metrics used to evaluate EVLNN against other models.

Metric	Description	Computation	
RMSE	RMSE squares the errors before calculating their mean, followed by taking the square root of the mean. It provides a good measure of the model's accuracy.	$\sqrt{\frac{1}{n}\sum_{t=1}^{n}(e_t)^2}$	(6.3)
MAE	MAE averages the absolute error of each predicted and measured pair. It measures how big an error from the forecast is on average.	$\frac{1}{n}\sum_{t=1}^{n}\lvert e_t\rvert$	(6.4)
MBE	MBE estimates the average bias in the model between over and underprediction.	$-\frac{1}{n}\sum_{t=1}^{n} e_t$	(6.5)
MAPE	MAPE compares the error ratio over the actual in percentage terms from different series on different scales.	$\frac{100}{n}\sum_{t=1}^{n}\left\lvert\frac{e_t}{y_t}\right\rvert$	(6.6)
R^2	Coefficient of Determination (R^2) measures the strength of the relationship between the model and the predictors. It measures how well the predictors explain future responses on a scale of 0 to 1.	$1-\left[\frac{\sum_{t=1}^{n} e_t^2}{\sum_{t=1}^{n}(y_t-\bar{y})^2}\right]$	(6.7)

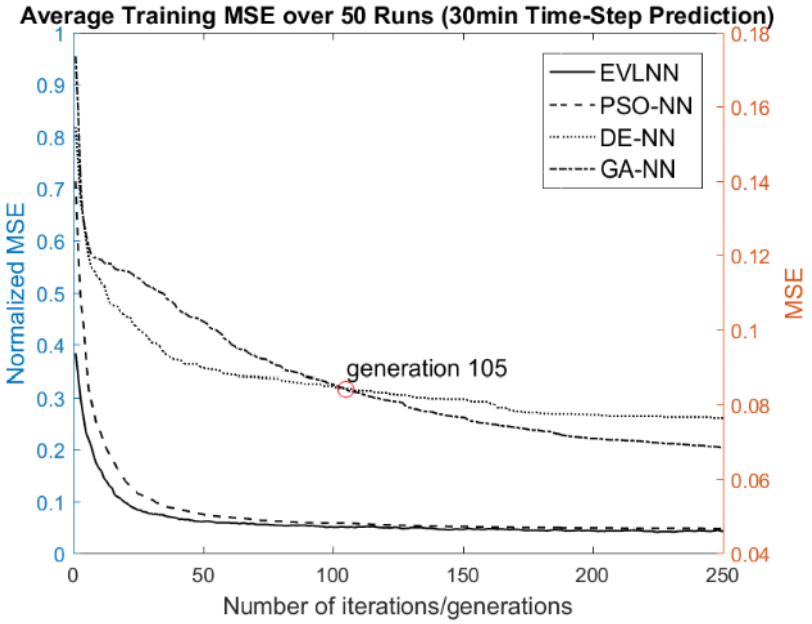
6.4.3 Training of EVLNN

Fifty trials were conducted with 250 iterations to train the models for four time-step predictions. The identified models are then applied to forecast solar irradiance using out-of-sample data as a test dataset, and the results are summarized in Table 6.7. In Table 6.7, it is observed that EVLNN had outperformed all the other EAs reaching low MSE values of 0.0480, 0.0446, 0.0441, and 0.0350 for each of the hourly, 30-min, 15-min, and 1-min time-step predictions, respectively, demonstrating good generality across diverse sizes of training datasets. The superior results can be attributed to the EVLNN algorithm for neural architecture search. TD-BPNN, a fully connected network, only outperformed EVLNN in the 1-min time-step prediction with a lower MSE value of 0.0175 compared to EVLNN's 0.0350. However, TD-BPNN could not generalize to other datasets and time-step predictions. Figure 6.12(a-d) compares the MSE convergence values reached by the EA-based learning algorithms at various time-step predictions. The left and right y-axes of the graphs show the normalized MSE and actual MSE values, whereas the x-axes show the number of iterations or generations.

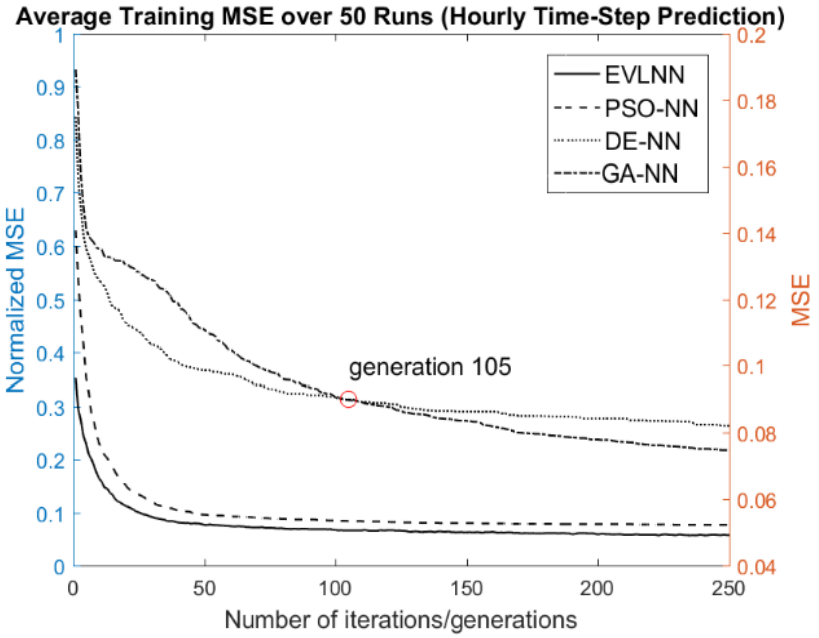
Table 6.7 Sample statistics of training MSE values averaged over N=50 runs tested for various models at each time-step prediction. Embolden figures to represent better results.

Time-step	Models	Mean	N	Std Dev	Std. Error Mean
Hourly	EVLNN	**0.0480**	50	0.0011	0.0002
	PSO-NN	0.0501	50	0.0007	0.0001
	DE-NN	0.0700	50	0.0080	0.0011
	GA-NN	0.0623	50	0.0068	0.0010
	TD-BPNN	0.1305	50	0.0112	0.0016
30-min	EVLNN	**0.0446**	50	0.0009	0.0001
	PSO-NN	0.0448	50	0.0006	0.0001
	DE-NN	0.0657	50	0.0079	0.0011
	GA-NN	0.0581	50	0.0084	0.0012
	TD-BPNN	0.0786	50	0.0067	0.0010
15-min	EVLNN	**0.0441**	50	0.0009	0.0001
	PSO-NN	**0.0441**	50	0.0007	0.0001
	DE-NN	0.0623	50	0.0071	0.0010
	GA-NN	0.0543	50	0.0054	0.0008
	TD-BPNN	0.0534	50	0.0038	0.0005
1-min	EVLNN	0.0350	50	0.0007	0.0001
	PSO-NN	0.0371	50	0.0007	0.0001
	DE-NN	0.0529	50	0.0054	0.0008
	GA-NN	0.0451	50	0.0038	0.0005
	TD-BPNN	**0.0175**	50	0.0035	0.0005

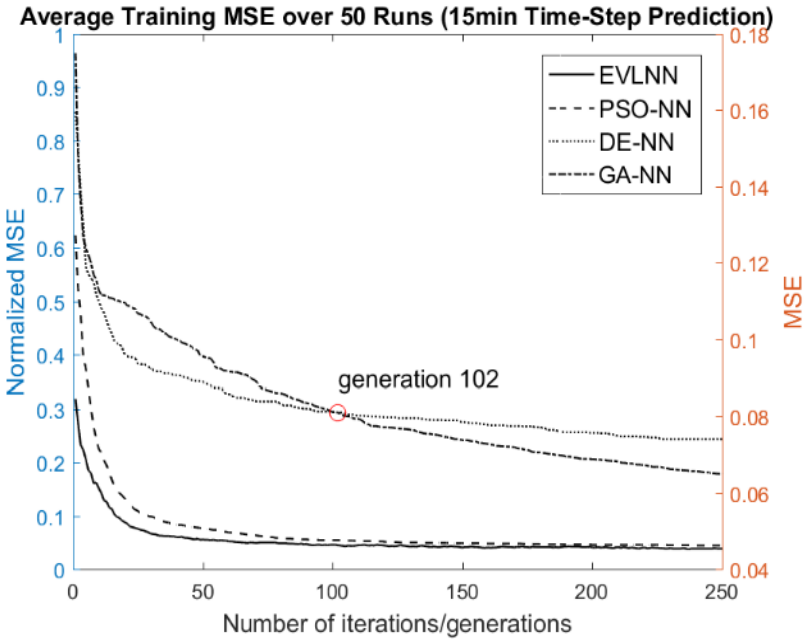
It is observed that EVLNN and PSO-NN converged faster to lower MSE. GA-NN has a slower convergence rate at the start. However, it overtook DE-NN after around 100 generations and converged to a lower MSE than DE-NN for all time-step predictions. The superiority of getting a lower MSE by EVLNN and PSO-NN demonstrated that EVLNN, with its species parallelism technique, and PSO-NN, with its swarm-based approach, had achieved superior results in navigating the architecture landscape to locate the optimal networks. However, DE-NN and GA-NN could not escape from local structural minima and hence were trapped at higher MSEs. The result is consistent with the experiments performed in Chapter 5 when predicting the Hadoop energy consumption, where EVLNN and PSO-NN also outperformed DE-NN and GA-NN.



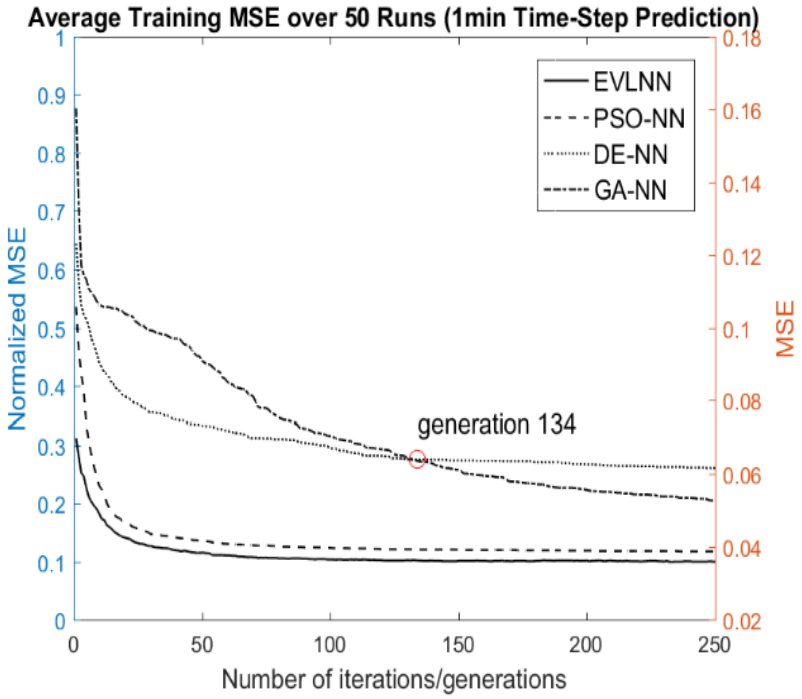
(a) Hourly time-step prediction.



(b) 30-min Time-Step Prediction.



(c) 15-min Time-Step Prediction.



(d) 1-min Time-Step Prediction.

Figure 6.12(a-d) Average training MSE repeated over 50 runs for various time-steps predictions

6.4.4 Model Convergence Speed and Rate

The experiment found that EVLNN has converged to a lower MSE faster, outpacing the other EA-based algorithms in all time-step predictions. The convergence speed is gauged based on the number of iterations each algorithm takes to reach specific checkpoint MSE values. In contrast, the average convergence rate is based on Equation 6.8, where Δy is the change in the dependent variable y and Δx is the change in independent variable x for a function f on a given interval $[x_1, x_2]$.

$$Average\ Convergence\ Rate = \frac{\Delta y}{\Delta x} = \frac{f(x_2)-f(x_1)}{x_2-x_1} \tag{6.8}$$

In calculating convergence speed, the values chosen are from $10\text{x}10^{-2}$ to $4\text{x}10^{-2}$ in steps of 0.01. Our findings are summarized in Table 6.8. The results demonstrated that EVLNN convergences to these values faster than the other EA-based algorithms. For instance, in the per-minute time-step prediction, EVLNN took 1 generation (or iteration) to reach the

MSE value of $7x10^{-2}$, whereas PSO, DE-NN, and GA-NN took 6, 69, and 102 iterations, respectively. In the hourly time-step prediction, EVLNN took nine iterations to reach the MSE value of $7x10^{-2}$, and PSO-NN took 15 iterations. However, DE-NN and GA-NN stagnated after 108 and 172 iterations, respectively, making minimal improvement. When calculating the average convergence rate, the entire iteration length is applied before stagnation sets in, up to 250 iterations. The results in Table 6.8 showed that EVLNN has the fastest convergence rate for 1-min, 15-min, and 30-min time-step predictions, whereas PSO-NN has the fastest convergence rate for the hourly time-step prediction.

Table 6.8 Comparison of convergence speed and rate. Embolden figures to indicate the best results.

Time-step (training sample)	**EVLNN**	**PSO-NN**	**DE-NN**	**GA-NN**	**MSE Value**	**Convergence Rate** $\left(\frac{\Delta y}{\Delta x}\right)$
	No. of iterations or generations					
1-min (53,580)	**1**	2	7	29	10 x 10^{-2}	EVLNN = $\frac{10x10^{-2}-4x10^{-2}}{32-1} = \mathbf{19.35x10^{-4}}$
	1	3	11	53	9 x 10^{-2}	
	1	5	23	71	8 x 10^{-2}	PSO-NN = $\frac{10x10^{-2}-4x10^{-2}}{90-2} = 6.82x10^{-4}$
	1	6	69	102	7 x 10^{-2}	
	4	9	-	156	6 x 10^{-2}	DE-NN = $\frac{10x10^{-2}-7x10^{-2}}{69-7} = 4.84x10^{-4}$
	10	15	-	-	5 x 10^{-2}	
	32	90	-	-	4 x 10^{-2}	GA-NN = $\frac{10x10^{-2}-6x10^{-2}}{156-29} = 3.15x10^{-4}$
15-min (3,572)	**1**	4	16	40	10 x 10^{-2}	EVLNN = $\frac{10x10^{-2}-5x10^{-2}}{28-1} = \mathbf{18.51x10^{-4}}$
	1	6	43	69	9 x 10^{-2}	
	2	8	118	107	8 x 10^{-2}	PSO-NN = $\frac{10x10^{-2}-5x10^{-2}}{57-4} = 9.43x10^{-4}$
	5	12	-	186	7 x 10^{-2}	
	11	16	-	-	6 x 10^{-2}	DE-NN = $\frac{10x10^{-2}-8x10^{-2}}{118-16} = 1.96x10^{-4}$
	28	57	-	-	5 x 10^{-2}	
	-	-	-	-	4 x 10^{-2}	GA-NN = $\frac{10x10^{-2}-7x10^{-2}}{186-40} = 2.05x10^{-4}$
30-min (1,786)	**1**	5	26	55	10 x 10^{-2}	EVLNN = $\frac{10x10^{-2}-5x10^{-2}}{38-1} = \mathbf{13.51x10^{-4}}$
	2	6	49	81	9 x 10^{-2}	
	4	8	163	128	8 x 10^{-2}	PSO-NN = $\frac{10x10^{-2}-5x10^{-2}}{56-5} = 9.80x10^{-4}$
	7	12	-	222	7 x 10^{-2}	
	15	22	-	-	6 x 10^{-2}	DE-NN = $\frac{10x10^{-2}-8x10^{-2}}{163-26} = 1.46x10^{-4}$
	38	56	-	-	5 x 10^{-2}	
	-	-	-	-	4 x 10^{-2}	GA-NN = $\frac{10x10^{-2}-7x10^{-2}}{222-55} = 1.80x10^{-4}$
Hourly (894)	**1**	5	42	70	10 x 10^{-2}	EVLNN = $\frac{10x10^{-2}-5x10^{-2}}{164-1} = 3.06x10^{-4}$
	2	7	108	104	9 x 10^{-2}	
	4	9	-	172	8 x 10^{-2}	PSO-NN = $\frac{10x10^{-2}-6x10^{-2}}{29-5} = \mathbf{4.16x10^{-4}}$
	9	15	-	-	7 x 10^{-2}	
	17	29	-	-	6 x 10^{-2}	DE-NN = $\frac{10x10^{-2}-9x10^{-2}}{108-42} = 1.52x10^{-4}$
	164	-	-	-	5 x 10^{-2}	
	-	-	-	-	4 x 10^{-2}	GA-NN = $\frac{10x10^{-2}-8x10^{-2}}{172-70} = 1.96x10^{-4}$

Note: A 'hypen' ('-') means stagnation with minimal improvement.

6.5. Results and Discussion

6.5.1. Model Testing and Analysis for Phase 1 – Use of Multiple Features

Table 6.9 shows the sample statistics of the MSE score for the testing dataset averaged over 50 runs for each predictive time-step for phase one of the experiment. All seven features were used to train the models. It is observed that PSO-NN has achieved a low MSE score for hourly, 30-min, and 15-min time-step predictions, followed by EVLNN. As for the 1-min time-step prediction, TD-BPNN was ranked top.

Table 6.9 Sample statistics of MSE scores averaged over N=50 runs for each time-step prediction. Embolden figures represent the best results.

Time-step (Sample Size)	Models	Mean	N	Std Deviation	Std Error Mean
Hourly (894 samples)	EVLNN	0.0524	50	0.0056	0.0008
	PSO-NN	**0.0482**	50	0.0044	0.0006
	DE-NN	0.0673	50	0.0194	0.0027
	GA-NN	0.0581	50	0.0134	0.0019
	TD-BPNN	0.0610	50	0.0136	0.0019
30-min (1,786 samples)	EVLNN	0.0542	50	0.0038	0.0005
	PSO-NN	**0.0518**	50	0.0034	0.0005
	DE-NN	0.0689	50	0.0187	0.0026
	GA-NN	0.0597	50	0.0111	0.0016
	TD-BPNN	0.0623	50	0.0046	0.0007
15-min (3,572 samples)	EVLNN	0.0425	50	0.0027	0.0004
	PSO-NN	**0.0403**	50	0.0015	0.0002
	DE-NN	0.0593	50	0.0094	0.0013
	GA-NN	0.0545	50	0.0098	0.0014
	TD-BPNN	0.0473	50	0.0043	0.0006
1-min (53,580 samples)	EVLNN	0.0376	50	0.0013	0.0002
	PSO-NN	0.0381	50	0.0011	0.0002
	DE-NN	0.0567	50	0.0081	0.0011
	GA-NN	0.0479	50	0.0052	0.0007
	TD-BPNN	**0.0264**	50	0.0027	0.0004

In using multiple features for prediction, EVLNN generally performed well in all the experiments. Its ranking is below PSO-NN, except for the 1-min time-step prediction where EVLNN is below TD-BPNN but above PSO-NN. The results demonstrated EVLNN's generalization capability with sparse or dense datasets. The findings confirmed that EVLNN is a competitive technique outperforming DE-NN and GA-NN while comparable to PSO-NN for 30-min and 1-min time-step predictions. EVLNN's technique with intra-

species and inter-species recombination strategy provides diversity balance to give the algorithm a generality when working with sparse or dense datasets. It is observed that TD-BPNN with fully connected layers performed poorly against the partially connected networks trained using EAs in the hourly predictions. This result may be attributed to the fact that a small dataset might be sufficient for low complexity models but insufficient to train a complex model such as the TD-BPNN. However, the strength of TD-BPNN is seen as the size of the dataset increased with shorter time-step predictions. In the 1-min time-step prediction, it was observed that TD-BPNN had outperformed all the other EA-based techniques.

A multiple paired sample t-test was conducted to compare EVLNN's performance with the other learning techniques to investigate if there were statistical differences between condition means. Bonferroni method is used to correct the *p*-value using $p = \frac{\alpha}{n}$, where α is the original *p*-value and *n* is the number of paired samples tests performed. At α=0.01 and $n = 10$, we obtained $p \leqslant 0.001$. Table 6.10 shows the results of the paired samples t-test.

Table 6.10 Paired Samples Test. Embolden figures denotes the paired differences between EVLNN's average testing MSE scores and the other model, which are lower and statistically significant.

			Paired Differences							
						Confidence Interval of the Difference (99%)				
			Mean	**Std. Deviation**	**Std Error Mean**	**Lower**	**Upper**	**t**	**df**	**Sig. (2-tailed)**
Hourly	Pair 1	EVLNN – PSO-NN	0.0042	0.0069	0.0010	0.0016	0.0068	4.314	49	0.000
	Pair 2	EVLNN – DE-NN	**-0.0149**	0.0205	0.0029	-0.0227	-0.0071	-5.142	49	0.000
	Pair 3	EVLNN_– GA-NN	-0.00580	0.0142	0.0020	-0.0111	-0.0004	-2.864	49	0.006
	Pair 4	EVLNN – TD-BPNN	**-0.0086**	0.0148	0.0021	-0.0142	-0.0029	-4.075	49	0.000
30-min	Pair 5	EVLNN – PSO-NN	0.0024	0.0059	0.0008	0.0002	0.0047	2.896	49	0.006
	Pair 6	EVLNN – DE-NN	**-0.0147**	0.0197	0.0028	-0.0221	-0.0072	-5.272	49	0.000
	Pair 7	EVLNN – GA-NN	-0.0055	0.0120	0.0017	-0.0100	-0.0009	-3.227	49	0.002
	Pair 8	EVLNN – TD-BPNN	**-0.0080**	0.0060	0.0009	-0.0103	-0.0058	-9.493	49	0.000
15-min	Pair 9	EVLNN – PSO-NN	0.0022	0.0030	0.0004	0.0011	0.0034	5.257	49	0.000
	Pair 10	EVLNN – DE-NN	**-0.0168**	0.0099	0.0014	-0.0205	-0.0130	-12.016	49	0.000
	Pair 11	EVLNN – GA-NN	**-0.0119**	0.0101	0.0014	-0.0157	-0.0081	-8.369	49	0.000
	Pair 12	EVLNN – TD-BPNN	**-0.0048**	0.0047	0.0007	-0.0065	-0.0030	-7.249	49	0.000
1-min	Pair 13	EVLNN – PSO-NN	-0.0005	0.0019	0.0003	-0.0012	0.0002	-1.813	49	0.076
	Pair 14	EVLNN – DE-NN	**-0.0191**	0.0083	0.0012	-0.0222	-0.0159	-16.287	49	0.000
	Pair 15	EVLNN – GA-NN	**-0.0103**	0.0057	0.0008	-0.0125	-0.0082	-12.895	49	0.000
	Pair 16	EVLNN – TD-BPNN	0.0112	0.0031	0.0004	0.0101	0.0124	25.799	49	0.000

The results highlighted that for hourly time-step prediction, there was a significant average difference between the MSE scores ($t_{49} = 4.314, p < 0.001$) of EVLNN and PSO-NN. On average, EVLNN's testing MSE scores were 0.0042 points higher (99% CI [0.0016, 0.0068]). In comparison with DE-NN and TD-BPNN, EVLNN's average MSE scores were 0.0149 (99% CI [-0.0227, -0.0071]) and 0.0086 (99% CI [-0.0142, -0.0029]) points lower, respectively. These differences were statistically significant (($t_{49} = -5.142, p < 0.001$) and ($t_{49} = -4.075, p < 0.001$)). In comparison with GA-NN, the average difference in the testing MSE scores between EVLNN and GA-NN were not significant ($t_{49} = -2.864, p < 0.001$). In the 30-min time-step prediction, the difference between EVLNN's average MSE scores and PSO-NN was not statistically significant. In the 15-min time-step prediction, EVLNN's average MSE scores were significantly higher than PSO-NN but lower than DE-NN, GA-NN, and TD-BPNN. In the per min time-step prediction, EVLNN's average MSE scores were significantly lower than DE-NN and GA-NN but significantly higher than TD-BPNN.

6.5.2. Comparison of 7-Day Forecasting Horizon Plots

The forecast horizon of seven days from 1st to 7th April 2016 is used to evaluate the forecast proficiency of the models. The predicted solar irradiance values by the models are denormalized to W/m^2 and plotted against the actuals for comparison in Figure 6.13 to Figure 6.16. Figure 6.13 shows the forecast result for the hourly time-step resolution. In general, the models could reasonably predict the hourly irradiance, as demonstrated by the prediction trend that tracked the actual values, except for the first and third day, highlighted by the red circles, where the prediction did not match up against the actual values. The predictions overcompensated the actual values towards the peak in the first red circle. The models predicted an upturn in the second circle, but the actual data was a downturn. Further analysis was later conducted, in particular, to investigate EVLNN's predictive accuracy on those days.

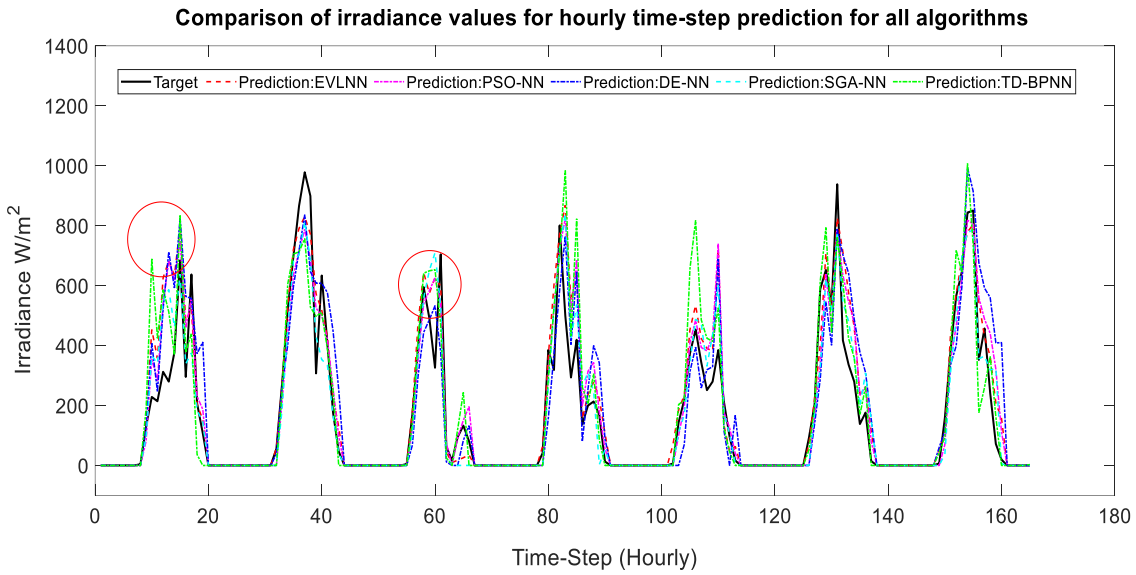


Figure 6.13 Hourly time-step predictions where circled portion indicates prediction with little success.

Figure 6.14 shows the results of the 30-min time-step predictions. It is observed that DE-NN had over-predicted parts of days three, four, and five. TD-BPNN had over-predicted parts of days three, five, and seven, and EVLNN overcompensated the peak on day five. However, unlike the hourly prediction, the overcompensation issue did not occur on day one of this experiment. Nonetheless, prediction for day three solar irradiance remains challenging for all the models.

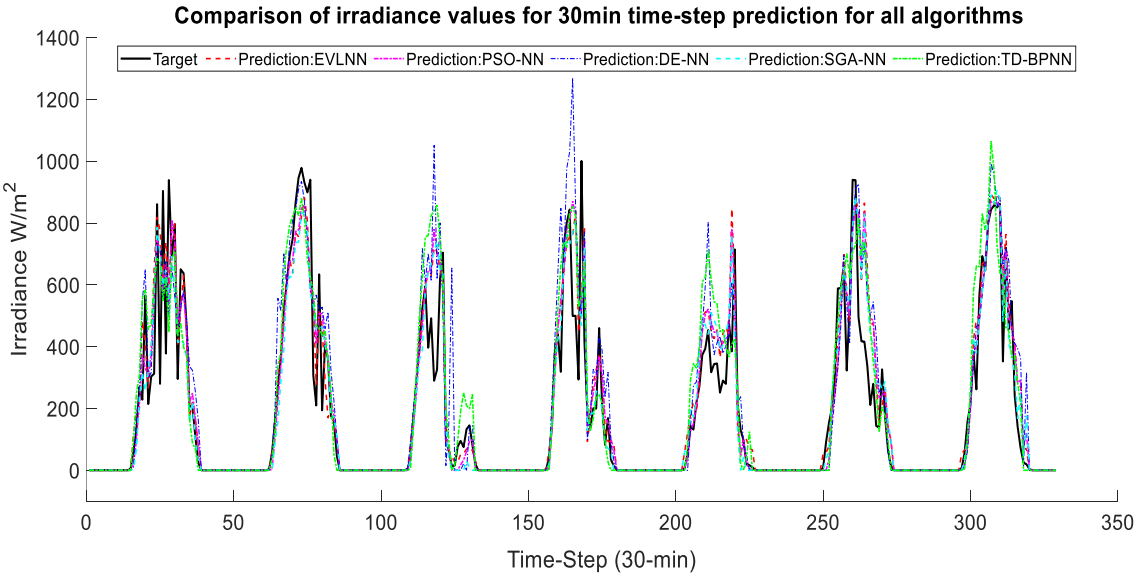


Figure 6.14 30-min time-step predictions.

Figure 6.15 shows the results of the 15-min time-step predictions. The models' predictions for part of day three and day five did not match up to the actual. The models overpredicted the solar irradiance in those regions. Nonetheless, EVLNN's prediction was the least overcompensation for day five compared to the other models.

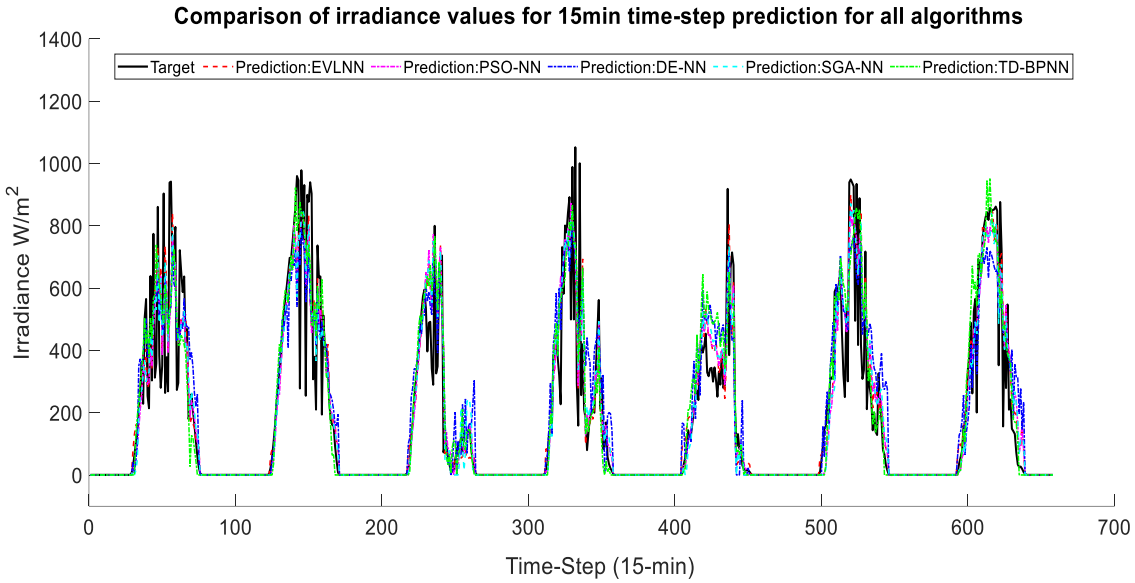


Figure 6.15 15-min time-step predictions.

Figure 6.16 shows the results of the per min time-step predictions. The charts are separated by the respective protocols for clarity due to the huge number of samples. It is observed that the models were able to track the actual plot in general, although each had varying accuracy. DE-NN overpredicted days three and five and underpredicted days two, four, six, and seven. All the models overpredicted the solar irradiance for part of day five.

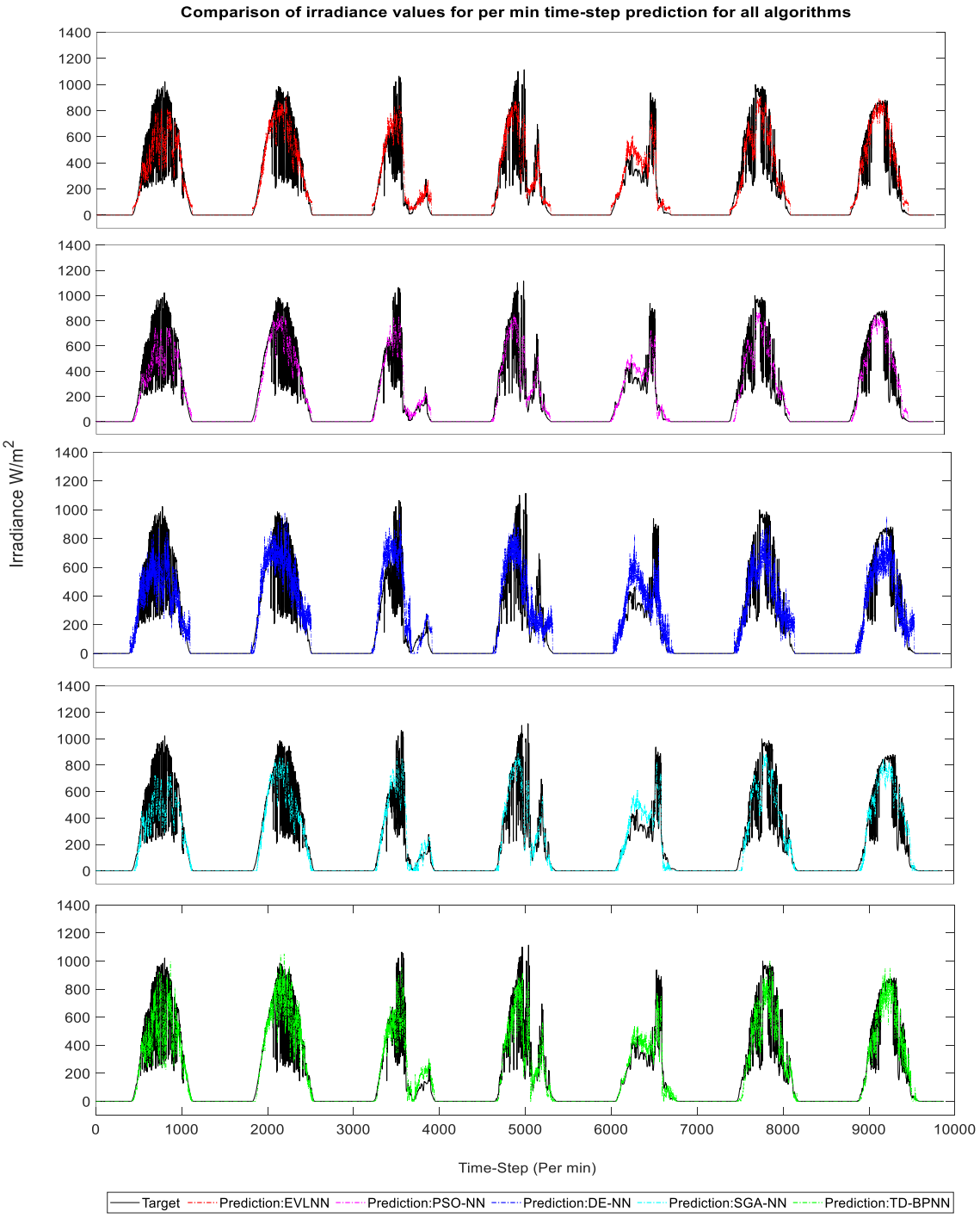


Figure 6.16 Per min time-step predictions.

Further analysis was performed for predictions on 1st and 3rd April 2016. Figures 6.17 and 6.18 compare the individual algorithms' target and predicted irradiance. The irradiance on 1st April 2016 was observed to have much higher variability than the other days. A heavy downpour occurred on 3rd April, which disrupted solar irradiance for at least an hour.

Despite these phenomena, EVLNN has generally performed well. For 1-min and 15-min time-step predictions, EVLNN, with the other EA-based ANNs, has successfully predicted the target curve. While TD-BPNN has the best per-minute prediction, it performed poorly against EVLNN for the other time-step predictions. The results proved that EA-based ANNs with partial connectivity are capable of modeling complex non-linear functions like high variability solar irradiance. EVLNN's species parallelism, crossovers strategies, and two-stage mutation have helped maintain an effective search producing good generalization across all time-step predictions.

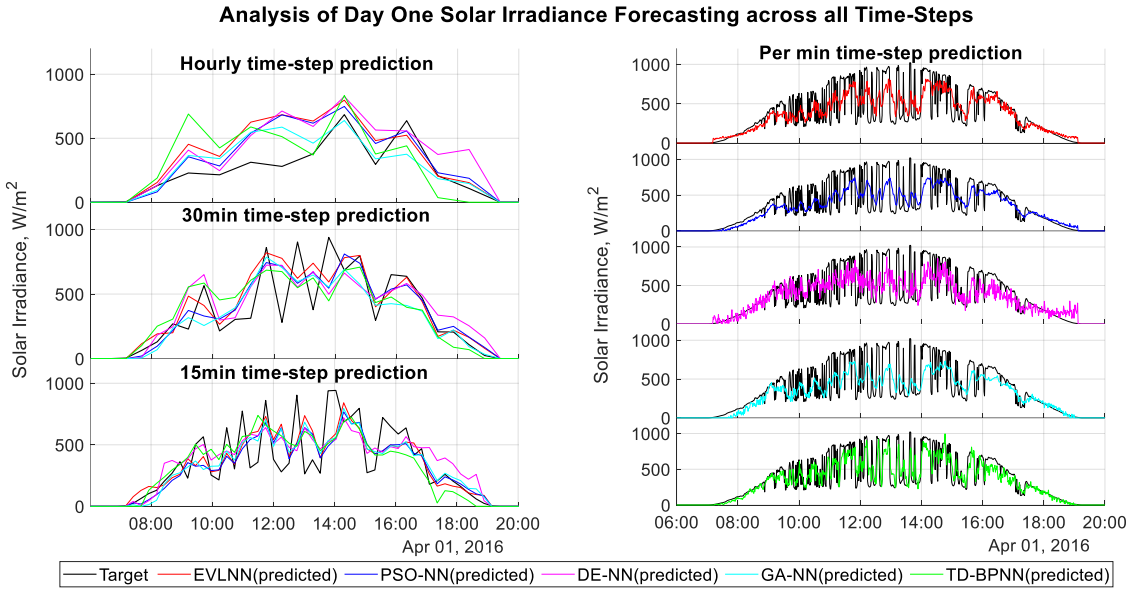


Figure 6.17 Comparison of target and predicted irradiance on 1st April 2016 for all time-steps.

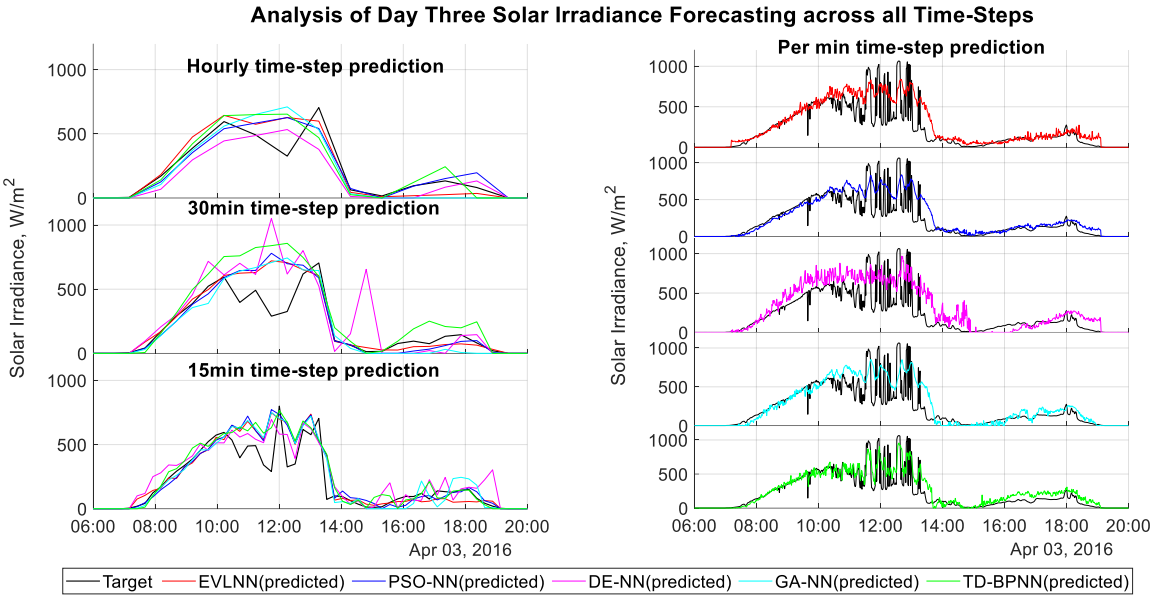


Figure 6.18 Comparison of target predicted irradiance on 3rd April 2016 for all time-steps.

6.5.3. Model Testing and Analysis for Phase 2 – Use of Smaller Number of Input Features

Feature data for solar irradiance forecasting are generally expensive to collect due to the high weather and meteorological instrument cost. The selection of fewer and essential attributes can reduce the dimensionality of the data, resulting in a simpler model and making model implementation less expensive. This experiment investigates EVLNN's performance and ability to generalize using a smaller scale of feature subsets for training. In the selection, features with a higher absolute PCC value were chosen from the original seven features to form three subsets. These features are solar panel surface temperature (PT), ambient temperature (AT), and relative humidity (RH), whose PCC values are 0.89, 0.77, and -0.71, respectively. The feature subsets were grouped as shown in Table 6.11.

Table 6.11 Smaller scale of feature subsets are used for training the EVLNN models.

Experiments	Number of input features	Features
Set 1 (original)	7	$x_1(t) = AT$, $x_2(t) = RH$, $x_3(t) = RG$, $x_4(t) = WS$, $x_5(t) = WD$, $x_6(t) = AP$, $x_7(t) = PT$
Set 2 (new)	3	$x_1(t) = PT$, $x_2(t) = AT$, $x_3(t) = RH$
Set 3 (new)	2	$x_1(t) = PT$, $x_2(t) = AT$
Set 4 (new)	1	$x_1(t) = PT$

The experiment results in Table 6.12 showed that EVLNN continued to perform well, if not better, under fewer input features. In particular, the EVLNN model trained with a single input feature had the lowest testing MSE scores for hourly, 30-min, and 15-min time-step predictions compared to the other models. Trained with two and three input features, the EVLNN models have also performed well with the lowest testing MSE scores for both hourly and 30-min time-step predictions. The results found clear support for the EVLNN model trained with fewer input features. It is notable that EVLNN trained with a single feature, specifically using only solar panel surface temperature as input, performed well, giving good results in the hourly, 30 min, and 15 min time-step predictions. Moreover, solar panel surface sensor temperature is more inexpensive to acquire than meteorological data. This result compares favorably to ANN models in current studies that predominantly use meteorological variables as inputs.

Table 6.12 Sample statistics of MSE scores averaged over N=50 runs for all models using a smaller scale of feature subsets. Embolden figures to mean the best result for that time-step prediction.

		Models	Mean MSE Score	N	Std. Deviation	Std. Error Mean
Three input features	Hourly	EVLNN	**0.0459**	50	0.0028	0.0004
		PSO-NN	0.0476	50	0.0031	0.0004
		DE-NN	0.0501	50	0.0066	0.0009
		GA-NN	0.0491	50	0.0051	0.0007
		TD-BPNN	0.0610	50	0.0138	0.0020
	30-min	EVLNN	**0.0504**	50	0.0015	0.0002
		PSO-NN	0.0532	50	0.0025	0.0004
		DE-NN	0.0552	50	0.0062	0.0009
		GA-NN	0.0553	50	0.0060	0.0008
		TD-BPNN	0.0623	50	0.0047	.0007
	15-min	EVLNN	0.0422	50	0.0010	0.0001
		PSO-NN	**0.0410**	50	0.0011	0.0002
		DE-NN	0.0437	50	0.0027	0.0004
		GA-NN	0.0438	50	0.0032	0.0005
		TD-BPNN	0.0473	50	0.0043	0.0006
	1-min	EVLNN	0.0361	50	0.0005	0.0001
		PSO-NN	0.0386	50	0.0007	0.0001
		DE-NN	0.0413	50	0.0025	0.0004
		GA-NN	0.0416	50	0.0032	0.0005
		TD-BPNN	**0.0264**	50	0.0027	0.0004
Two input features	Hourly	EVLNN	**0.0454**	50	0.0016	0.0002
		PSO-NN	0.0459	50	0.0032	0.0005
		DE-NN	0.0459	50	0.0048	0.0007
		GA-NN	0.0486	50	0.0056	0.0008
		TD-BPNN	0.0541	50	0.0095	0.0013

	30-min	EVLNN	**0.0499**	50	0.0014	0.0002
		PSO-NN	0.0523	50	0.0026	0.0004
		DE-NN	0.0526	50	0.0041	0.0006
		GA-NN	0.0529	50	0.0048	0.0007
		TD-BPNN	0.0573	50	0.0037	0.0005
	15-min	EVLNN	0.0409	50	0.0007	0.0001
		PSO-NN	**0.0402**	50	0.0006	0.0001
		DE-NN	0.0415	50	0.0016	0.0002
		GA-NN	0.0417	50	0.0016	0.0002
		TD-BPNN	0.0474	50	0.0028	0.0004
	1-min	EVLNN	0.0355	50	0.0004	0.0001
		PSO-NN	0.0383	50	0.0006	0.0001
		DE-NN	0.0391	50	0.0012	0.0002
		GA-NN	0.0394	50	0.0013	0.0002
		TD-BPNN	**0.0288**	50	0.0005	0.0001
One input feature	Hourly	EVLNN	**0.0453**	50	0.0012	0.0002
		PSO-NN	0.0497	50	0.0016	0.0002
		DE-NN	0.0493	50	0.0023	0.0003
		GA-NN	0.0493	50	0.0028	0.0004
		TD-BPNN	0.0671	50	0.0172	0.0024
	30-min	EVLNN	**0.0497**	50	0.0010	0.0001
		PSO-NN	0.0525	50	0.0008	0.0001
		DE-NN	0.0520	50	0.0020	0.0003
		GA-NN	0.0528	50	0.0018	0.0003
		TD-BPNN	0.0563	50	0.0078	0.0011
	15-min	EVLNN	**0.0418**	50	0.0004	0.0001
		PSO-NN	0.0421	50	0.0003	0.0000
		DE-NN	0.0424	50	0.0008	0.0001
		GA-NN	0.0421	50	0.0001	0.0001
		TD-BPNN	0.0477	50	0.0024	0.0003
	1 min	EVLNN	0.0376	50	0.0003	0.0000
		PSO-NN	0.0416	50	0.0032	0.0005
		DE-NN	0.0408	50	0.0007	0.0001
		GA-NN	0.0404	50	0.0007	0.0001
		TD-BPNN	**0.0308**	50	0.0006	0.0002

6.5.4. Statistical Analysis of Models' Forecasting Accuracy with Fewer Inputs

A multiple paired sample t-test with Bonferroni correction was conducted to compare significant differences between the testing MSE scores averaged over 50 runs for the various learning techniques. At $\alpha = 0.01$, the adjusted *p*-value for pairwise comparison where the *p*-value is required for significance would be $p \leqslant 0.001$. Table 6.13 shows the results of the paired samples t-test.

Table 6.13 Paired samples t-test. Embolden figures denotes the paired differences between EVLNN's average testing MSE scores and the other model, which are lower and statistically significant.

				Paired Differences							
							99% Confidence Interval of the Difference				
		Pair	Paired Samples	Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	Sig. (2-tailed)
3 inputs	Hourly	1	EVLNN – PSO-NN	-0.0017	0.0045	0.0006	-0.0034	-0.0000	-2.704	49	0.009
		2	EVLNN – DE-NN	**-0.0042**	0.0071	0.0010	-0.0069	-0.0015	-4.199	49	0.000
		3	EVLNN – GA-NN	**-0.0032**	0.0057	0.0008	-0.0054	-0.0010	-3.923	49	0.000
		4	EVLNN – TD-BPNN	**-0.0151**	0.0148	0.0021	-0.0207	-0.0095	-7.210	49	0.000
	30-min	5	EVLNN – PSO-NN	**-0.0028**	0.0031	0.0004	-0.0039	-0.0016	-6.232	49	0.000
		6	EVLNN – DE-NN	**-0.0047**	0.0064	0.0009	-0.0072	-0.0023	-5.217	49	0.000
		7	EVLNN – GA-NN	**-0.0048**	0.0066	0.0009	-0.0073	-0.0023	-5.191	49	0.000
		8	EVLNN – TD-BPNN	**-0.0118**	0.0050	0.0007	-0.0137	-0.0099	-16.696	49	0.000
	15-min	9	EVLNN – PSO-NN	0.0012	0.0014	0.0002	0.0007	0.0017	6.331	49	0.000
		10	EVLNN – DE-NN	**-0.0014**	0.0029	0.0004	-0.0026	-0.0003	-3.499	49	0.001
		11	EVLNN – GA-NN	**-0.0016**	0.0032	0.0005	-0.0028	-0.0003	-3.415	49	0.001
		12	EVLNN – TD-BPNN	**-0.0051**	0.0044	0.0006	-0.0067	-0.0034	-8.252	49	0.000
	1-min	13	EVLNN – PSO-NN	**-0.0026**	0.0008	0.0001	-0.0029	-0.0022	-21.669	49	0.000
		14	EVLNN – DE-NN	**-0.0053**	0.0026	0.0004	-0.0063	-0.0043	-14.643	49	0.000
		15	EVLNN – GA-NN	**-0.0056**	0.0031	0.0004	-0.0068	-0.0044	-12.599	49	0.000
		16	EVLNN – TD-BPNN	0.0097	0.0027	0.0004	0.0086	0.0107	24.995	49	0.000
2 inputs	Hourly	17	EVLNN – PSO-NN	-0.0005	0.0033	0.0005	-0.0017	0.0008	-.996	49	0.324
		18	EVLNN – DE-NN	-0.0004	0.0052	0.0007	-0.0024	0.0015	-.584	49	0.562
		19	EVLNN – GA-NN	**-0.0032**	0.0056	0.0008	-0.0053	-0.0010	-3.958	49	0.000
		20	EVLNN – TD-BPNN	**-0.0087**	0.0093	0.0013	-0.0122	-0.0051	-6.571	49	0.000
	30-min	21	EVLNN – PSO-NN	**-0.0024**	0.0032	0.0005	-0.0036	-0.0012	-5.211	49	0.000
		22	EVLNN – DE-NN	**-0.0027**	0.0044	0.0006	-0.0043	-0.0010	-4.299	49	0.000
		23	EVLNN – GA-NN	**-0.0030**	0.0047	0.0007	-0.0048	-0.0013	-4.599	49	0.000
		24	EVLNN – TD-BPNN	**-0.0074**	0.0040	0.0006	-0.0089	-0.0059	-13.238	49	0.000
	15-min	25	EVLNN – PSO-NN	0.0008	0.0008	0.0001	0.0004	0.0011	6.542	49	0.000
		26	EVLNN – DE-NN	-0.0005	0.0018	0.0003	-0.0012	0.0002	-2.043	49	0.046
		27	EVLNN – GA-NN	-0.0008	0.0018	0.0003	-0.0014	-0.0001	-2.979	49	0.004
		28	EVLNN – TD-BPNN	**-0.0064**	0.0029	0.0004	-0.0075	-0.0053	-15.446	49	0.000
	1-min	29	EVLNN – PSONN	**-0.0028**	0.0007	0.0001	-0.0030	-0.0025	-27.663	49	0.000
		30	EVLNN – DE-NN	**-0.0036**	0.0012	0.0002	-0.0041	-0.0032	-21.208	49	0.000
		31	EVLNN – GA-NN	**-0.0039**	0.0014	0.0002	-0.0044	-0.0034	-19.874	49	0.000
		32	EVLNN – TD-BPNN	0.0067	0.0006	0.0001	0.0065	0.0069	84.695	49	0.000
1 input	Hourly	33	EVLNN – PSONN	**-0.0044**	0.0020	0.0003	-0.0052	-0.0036	-15.598	49	0.000
		34	EVLNN – DE-NN	**-0.0040**	0.0027	0.0004	-0.0051	-0.0030	-10.443	49	0.000
		35	EVLNN – GA-NN	**-0.0040**	0.0029	0.0004	-0.0051	-0.0029	-9.616	49	0.000
		36	EVLNN – TD-BPNN	**-0.0218**	0.0171	0.0024	-0.0283	-0.0153	-9.005	49	0.000
	30-min	37	EVLNN – PSONN	**-0.0028**	0.0012	0.0002	-0.0033	-0.0024	-16.592	49	0.000
		38	EVLNN – DE-NN	**-0.0023**	0.0023	0.0003	-0.0032	-0.0014	-7.054	49	0.000
		39	EVLNN – GA-NN	**-0.0030**	0.0021	0.0003	-0.0038	-0.0023	-10.241	49	0.000
		40	EVLNN – TD-BPNN	**-0.0066**	0.0079	0.0011	-0.0096	-0.0036	-5.848	49	0.000
	15-min	41	EVLNN – PSONN	**-0.0003**	0.0005	0.0001	-0.0005	-0.0001	-3.937	49	0.000
		42	EVLNN – DE-NN	**-0.0006**	0.0008	0.0001	-0.0010	-0.0003	-5.320	49	0.000
		43	EVLNN – GA-NN	**-0.0003**	0.0006	0.0001	-0.0005	-0.0001	-3.383	49	0.001
		44	EVLNN – TD-BPNN	**-0.0059**	0.0023	0.0003	-0.0068	-0.0050	-17.837	49	0.000
	1-min	45	EVLNN – PSONN	**-0.0039**	0.0032	0.0005	-0.0051	-0.0027	-8.783	49	0.000
		46	EVLNN – DE-NN	**-0.0031**	0.0008	0.0001	-0.0034	-0.0028	-27.447	49	0.000
		47	EVLNN – GA-NN	**-0.0027**	0.0008	0.0001	-0.0030	-0.0024	-24.537	49	0.000
		48	EVLNN – TD-BPNN	0.0069	0.0006	0.0001	0.0067	0.0071	85.453	49	0.000

When using PV as a single input feature, superior results were seen in the hourly, 30-min, and 15-min time-step predictions, where EVLNN's average testing MSE scores are

significantly better than PSO-NN, DE-NN, GA-NN, and TD-BPNN. In the 1 min prediction, there were significant differences in the testing MSE scores between EVLNN and PSO-NN ($t_{49} = -8.783, p < 0.001$), EVLNN and DE-NN ($t_{49} = -27.447, p < 0.001$) and, EVLNN and GA-NN ($t_{49} = -24.537, p < 0.001$). Extensive results of this analysis confirmed that EVLNN is a competitive model for solar irradiance prediction. More precisely, in the presence of fewer features, EVLNN achieved the most consistent and robust results.

6.5.5. Comparison of Error Metrics

The error metrics RMSE, MAE, adjusted R2 (Coefficient of Determination), MBE, and MAPE, are computed using the denormalized W/m^2 predicted values. Their values are evaluated to compare the model performance. Figure 6.19 presents the scores of the various error metrics using a bar chart for comparison. For hourly and 30-min time-step predictions, EVLNN trained with a smaller scale of feature subsets obtained better results with lower RMSE and MAE scores and higher Adjusted R^2 scores than other models. EVLNN trained with a single input feature (solar PV surface temperature) has performed exceptionally well, showing better results for RMSE, MAE, and adjusted R^2 in the hourly, 30-min, and 15-min time-step predictions than other models.

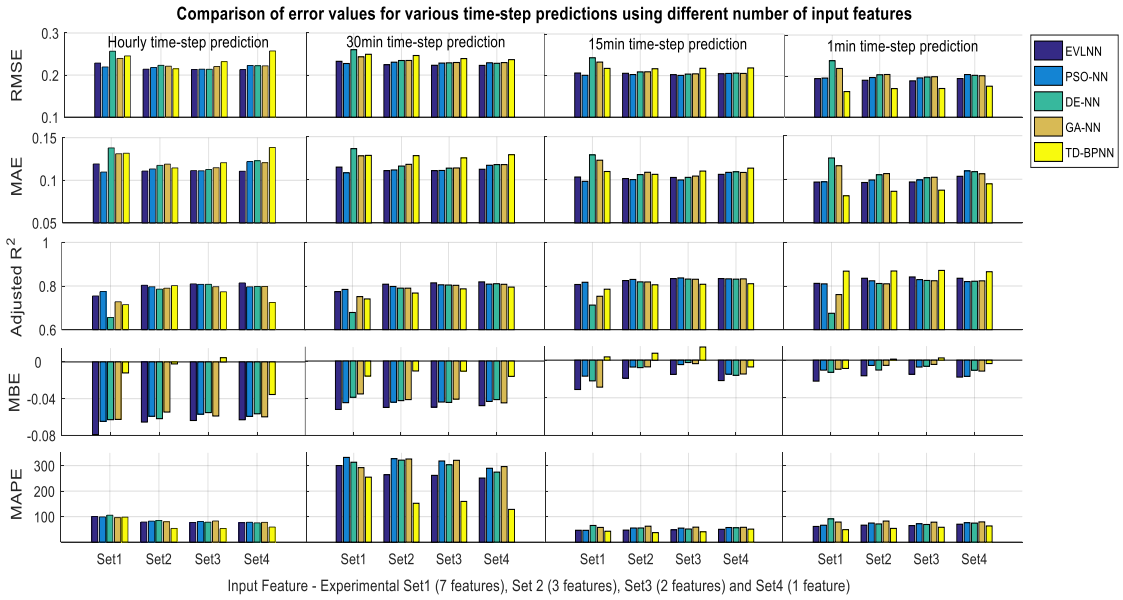


Figure 6.19 Error matrices are presented in a bar chart for comparison. In separate experiments, models were trained using seven, three, two, and single input features.

Table 6.14 shows the error scores averaged over 50 runs for the models trained with a single input feature for various time-step predictions. The values of error metrics, RMSE, and MAE are displayed in a denormalized form in W/m^2. It is observed that EVLNN has performed better than PSO-NN, DE-NN, GA-NN, and TD-BPNN in the hourly time-step prediction. In terms of percentage, EVLNN's RMSE values improved by 4.5%, 4.1%, 4.1%, and 17.3%, respectively, MAE value improved by 9.2%, 9.9%, 8.2%, and 20.2%, respectively and Adjusted R^2 value improved by 2.5%, 2.5%, 2.5%, and 12.3%, respectively. Similar results were also seen in the 30-min time-step prediction, where EVLNN's RMSE, MAE, and Adjusted R^2 values are better than PSO-NN, DE-NN, GA-NN, and TD-BPNN. For the 15-min time-step prediction, EVLNN's values for RMSE and MAE were better than the other models but tied with PSO-NN and GA-NN for the Adjusted R^2 value. In the 1 min time-step prediction, EVLNN error scores are better than PSO-NN, DE-NN, and GA-NN except for TD-BPNN. In this time-step prediction, TD-BPNN has outperformed EVLNN taking advantage of the power of fully connected networks in the presence of a large dataset. However, when the sample interval increases, TD-BPNN performance decreases. The results suggest that a fully connected ANN suffers from difficulties with generalization because of overfitting. Another interesting observation is

EVLNN's MBE scores, which were more negative than the other models. This finding suggests that the EVLNN's search characteristic is more conservative and biased toward under-predicting. For the TD-BPNN, though smaller overall negative MBE scores were observed, it tends to over predict, as indicated by the RMSE scores for hourly, 30-min, and 15-min time-step prediction for larger sample intervals.

Table 6.14 Comparison of error matrices averaged over 50 runs for models trained with a single input feature. Embolden figures to mean the best result for that time-step prediction.

Prediction Time-Step		Error Metrics									
		RMSE (W/m^2)		MAE (W/m^2)		Adjusted R^2		MBE (W/m^2)		MAPE (%)	
		Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Hourly	EVLNN	**104.08**	1.37	**54.12**	0.83	**0.82**	0.00	-30.79	2.15	50.17	2.81
	PSO-NN	109.00	1.70	59.61	0.68	0.80	0.01	-29.00	1.92	**47.96**	0.87
	DE-NN	108.58	2.46	60.08	1.00	0.80	0.01	-27.61	3.85	48.05	1.44
	GA-NN	108.50	3.04	58.98	0.80	0.80	0.01	-29.26	3.45	48.40	1.46
	TD-BPNN	125.79	14.73	67.78	6.18	0.73	0.07	**-17.36**	8.45	281.21	89.58
30=min	EVLNN	**111.51**	1.05	**56.13**	0.63	**0.82**	0.00	-24.03	1.76	52.12	1.83
	PSO-NN	114.62	0.91	58.44	0.26	0.81	0.00	-21.77	1.39	**47.79**	0.55
	DE-NN	114.04	2.13	58.80	0.74	0.81	0.01	-20.78	4.06	47.93	1.19
	GA-NN	114.87	1.92	58.84	0.66	0.81	0.01	-22.49	2.42	49.29	1.18
	TD-BPNN	118.41	7.33	64.59	2.51	0.80	0.03	**-8.27**	2.69	78.27	19.88
15-min	EVLNN	**107.53**	0.50	**55.91**	0.58	**0.84**	0.00	-11.40	1.07	55.17	1.57
	PSO-NN	107.90	0.34	57.11	0.60	**0.84**	0.00	-7.82	1.17	**50.98**	0.53
	DE-NN	108.34	0.98	57.47	0.95	0.83	0.00	-8.47	3.17	51.36	1.23
	GA-NN	107.92	0.73	57.07	1.03	**0.84**	0.00	-7.72	2.20	51.70	1.25
	TD-BPNN	114.85	2.74	59.71	1.08	0.81	0.01	**-3.80**	2.67	51.53	5.77
1-min	EVLNN	108.06	0.43	57.61	0.50	0.84	0.00	-10.02	1.30	60.04	1.31
	PSO-NN	111.38	1.21	59.87	0.73	0.82	0.01	-9.00	8.23	56.94	3.57
	DE-NN	112.46	0.96	60.52	1.38	0.83	0.00	-5.96	3.48	58.85	1.23
	GA-NN	111.88	0.89	59.19	1.27	0.83	0.00	-6.44	2.43	58.85	1.69
	TD-BPNN	**97.71**	0.86	**52.79**	0.67	**0.87**	0.00	**-2.00**	0.83	**50.56**	0.65

These results suggest that EVLNN has the superiority of getting lower RMSE and MAE and higher R^2 requiring fewer features. The findings show that the strength of EVLNN lies in the ability to predict accurately despite a smaller amount of features. It is also important to note that using only one error indicator may be insufficient, and more than one error indicator should be considered when objectively investigating the performance of predictive models.

6.6. Chapter Summary

Solar power intermittency threatens to reduce power generation from solar radiation, making data center energy transition to renewables challenging. Accurate prediction of solar irradiance is a critical research topic that can help manage risk, compensate for the variations of solar PV power and accelerate data center energy transition to renewables.

In this chapter, the EVLNN model is presented for solar irradiance forecasting. EVLNN's predictive performance for 1-min, 15-min, 30-min, and hourly time-step using fewer input features has been investigated. In the model development, MSE was employed as the objective function to minimize, and test statistics were adopted to investigate statistical differences in the condition means between paired samples. The predictive accuracy of EVLNN was measured using a range of five error matrices: RMSE, MAE, adjusted R^2, MBE, and MAPE.

It was observed that EVLNN trained with a single input feature had performed better than PSO-NN, DE-NN, GA-NN, and TD-BPNN in the hourly time-step prediction, where EVLNN's RMSE values improved by 4.5%, 4.1%, 4.1%, and 17.3%. Also, EVLNN's MAE values improved by 9.2%, 9.9%, 8.2%, and 20.2%, respectively and Adjusted R^2 values improved by 2.5%, 2.5%, 2.5%, and 12.3%, respectively. Similar results were also seen in the 30-min time-step prediction, where EVLNN's RMSE, MAE, and Adjusted R^2 values were better than PSO-NN, DE-NN, GA-NN, and TD-BPNN. For the 15-min time-step prediction, EVLNN's values for RMSE and MAE were better than the other models but tied with PSO-NN and GA-NN for the Adjusted R^2 values.

The experiments showed that EVLNN performed substantially better than the other models in three error indicators. The results also confirmed that EVLNN is a preferred model requiring a smaller scale of feature subsets and training datasets. When trained with a single input feature using the solar PV surface temperature, EVLNN performed well, giving superior results to the models trained by other EAs. This is an important finding as acquiring non-meteorological data is more straightforward and less expensive. This means that EVLNN compares favorably to most ANNs in the literature, generally trained with

meteorological data, which is expensive due to the setup and maintenance cost of weather instruments or stations. These results show that EVLNN is a promising solution for solar irradiance forecasting with resilient qualities to cope with the solar variability in the tropics.

Chapter 7

7. Conclusion and Future Work

The rapid acceleration of digital services has skyrocketed data center energy consumption to unprecedented levels. Accurate energy prediction can improve data center energy consumption and accelerate the transition to renewables for a more sustainable future. While ANNs are powerful learning machines for modeling complex nonlinear systems, they are non-interpretable and prone to overfitting due to their redundant architecture. Moreover, the design of ANN relies heavily on human experts and trial-and-error efforts, making it a time-consuming and expensive task. Methods that can make this task more efficient and provide insights into the relationships between the variables and energy consumption are desirable.

A comprehensive review of Hadoop data center energy efficiency in Chapter 2 led to an observed gap in the literature where the application of EA-based ANN is still limited for energy consumption predictions of Hadoop systems. Similarly, a review of time-series solar irradiance forecasting techniques reflects a general trend in the literature requiring a relatively large number of variables and datasets.

In this research, the problem of energy prediction is made efficient using an evolutionary-based approach to ANN learning, known as EVLNN, to produce parsimonious ANN with interpretability and improved generalizability. Automating the parameter learning of ANN has also minimized the dependency on human experts and costly trial-and-error efforts. The result is a relatively frugal ANN model, which also captures the meaning of nonlinear relationships between the inputs and the outputs.

7.1. The EVLNN Model for Energy Prediction

7.1.1 EVLNN Framework, Architecture, and Algorithm

The EVLNN model consisting of its framework, architecture, and search algorithm, was described in Chapter 3. The design of EVLNN has achieved its goal of a generalized and interpretable model for energy prediction applications. This is realized through EVLNN's several novel mechanisms that support species parallelism amidst the explorative search. A structurally inclusive matrix encoding scheme based on parsimony has been designed to accommodate problem representation of a feedforward and feedback ANN. The intra-species and inter-species crossover strategies and a two-stage mutation with weights and link-node have been created to automate the search for global solutions of parsimonious networks. An ensemble-based approach to sensitivity analysis has been proposed to improve model interpretability, making it easier to identify input features that most affect the outputs, and providing valuable insight into the factors influencing the predictions.

7.1.2 EVLNN's Search Capability and Performance in Benchmark Test Functions in Comparison with other EAs

The search capability of EVLNN was investigated in Chapter 4 through a set of 16 benchmark test functions [207] [261]. The first set of benchmark functions (see Table 4.4) demonstrated EVLNN's ability to locate the global optima in low-dimensional unimodal and multimodal landscapes. In particular, EVLNN had achieved an average Peak Ratio (PR) of 100% for the Bohachevsky N.1-2D (f_1) and Booth-2D (f_2) functions across all $\varepsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. EVLNN also achieved an average PR of 100% for the Ackley-2D function (f_5) at $\varepsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$, but only a PR of 6% at $\varepsilon = \{10^{-5}\}$. For the Rosenbrock-2D (f_6) function, PSO and DE performed better than EVLNN for all $\varepsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ with an average PR of 100%, whereas EVLNN and GA had an average PR of 91.2% and 92.4%, respectively. For high-dimensionality functions, EVLNN's performance was less desirable. For example, for the Griewank-30D (f_8) function, DE had the best performance, followed by GA, PSO, and EVLNN. For the Sphere-30D (f_3) and Brown-30D (f_4) functions, EVLNN could only locate the peaks at $\varepsilon = \{10^{-1}, 10^{-2}\}$.

While EVLNN had a moderate overall performance, it is the only algorithm that had located the peaks of all the test functions, f_1 to f_8, including f_7, Rastrigin-30D, demonstrating that the algorithm generalizes better than PSO, DE, and GA in this set of test functions.

The performance of EVLNN was compared to the state-of-the-art EAs presented in the CEC 2013 and CEC 2015 competitions using a second set of benchmark functions from the competitions (see Tables 4.5 and 4.11). EVLNN recorded an average PR score of 100% for the Uneven Decreasing Maxima-1D (f_{11}) and Six-hump Camel Back-2D (f_{13}) functions across all $\varepsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. EVLNN performed moderately well, achieving an average PR score of 59.1% and 44.6%, and 37.6% for the Modified Rastrigin-2D (f_{16}), Shubert-2D (f_{14}), and Vincent-2D (f_{15}) functions, respectively. For the Himmelblau-2D (f_{12}) and Equal Maxima-1D (f_{10}) functions, EVLNN attained an average PR score of 84.3% and 79.4% for all ε, respectively. Nonetheless, EVLNN is the only algorithm that had located the peak for the Shubert-2D (f_{14}) function at ε = 1.0E-05, with a PR of 27.6%. At the same time, the other EAs failed at this accuracy level, demonstrating the generalizability of EVLNN. In the overall assessment, EVLNN was ranked 19 out of 22 algorithms presented in the CEC competitions, outperforming iPOP-CMA-ES, MEA, and MSSPSO. The ranking is based on the average PR score across 10 CEC multimodal benchmark functions, where EVLNN achieved an average PR score of 50%. Overall, EVLNN had demonstrated the ability to locate the global optima in low-dimensional unimodal and multimodal landscapes and, on average, 50% of the global optima in high-dimensional multimodal landscapes.

It is important to note that the state-of-the-art algorithms developed for the CEC competition focus on real-parameter optimization. In contrast, EVLNN is a parsimonious ANN built for forecasting. Therefore, the EVLNN's search capability and applicability to real-world problems were further assessed in Chapter 4. Using open-access real-world time-series data as a benchmark to forecast electricity load for the experiment, EVLNN had achieved a better performance in terms of a lower testing MSE score than ANNs trained using PSO, DE, and GA.

7.2. Application to Hadoop Energy Consumption Prediction

For a more thorough experiment, a physical testbed was set up to test the ability of EVLNN to model data center energy consumption, which was discussed in Chapter 5. The testbed consisted of a 120-core Hadoop cluster where a combination of CPU-intensive and I/O-intensive MapReduce jobs, such as the WordCount and TeraSort applications typically used in the literature, with varying payloads were executed to simulate real-world workloads. Energy-related data from 23 parameters, the highest number of variables examined compared to similar studies performed in the literature, were collected via monitoring tools like Ganglia, SNMP protocol, and build-in Hadoop counters. The EVLNN model was trained using 70% of the dataset and tested with the remaining 30% to predict the Hadoop energy consumption. Its performance was compared to ANN trained with other EAs such as PSO-NN, DE-NN, and the classic GA-NN. EVLNN's testing MSE score of 0.00230 ($\pm$0.00042) outperformed the testing MSE score of PSO-NN, DE-NN, and GA-NN, which were recorded at 0.00310 ($\pm$0.00195), 0.01041 ($\pm$0.00307), and 0.01071 ($\pm$0.00416), respectively. The result demonstrated that EVLNN is a competitive search algorithm capable of predicting the energy consumption of a Hadoop cluster.

In addition, factors influencing energy consumption were also successfully investigated using EVLNN's ensemble sensitivity analysis. 23 energy-related input variables were classified into five categories: job profile, system utilization, disk I/O, network transfer, and the environment. The relative importance of individual variables was aggregated and averaged across four chosen sensitivity analysis methods. The outcome of the category ranking placed *job profile* as the most critical factor contributing to energy consumption, followed by *system utilization*, d*isk I/O*, *network transfer*, and finally, *the environment*. The findings attributed *file size*, *workload type*, *job completion time*, and the *number of mappers and reducers* as significant factors influencing energy consumption. This insight can provide potential ways to manage data center energy efficiency better.

7.3. Application to Solar Irradiance Forecasting

A small-scale PV system setup with weather instruments was used as a testbed to evaluate EVLNN's solar irradiance forecasting ability in the real world. One month of raw data in March 2016 was sampled at 1-min, 15-min, 30-min, and hourly time resolutions to create four datasets. These datasets were used to train the EVLNN model to forecast solar irradiance over seven days. The 1-min dataset had the largest sample size of 53,580, and the hourly data had the smallest sample size of 894. The input features were ambient temperature, relative humidity, rain gauge reading, wind speed, wind direction, atmospheric pressure, and PV panel surface temperature. EVLNN's performance was investigated through two experimental phases. The first phase involved seven input features, while the second involved fewer input features. The approach of selecting fewer and essential attributes was an attempt to achieve the reverse of what most forecast models do, that is, relying on many meteorological and atmospheric input features. Phase two investigated EVLNN's generalization capability using fewer features. In analyzing the results from the experiments, the best scenarios and applications of EVLNN are in forecasting solar irradiance at the hourly and 30-min time-step resolutions with a smaller subset of input features of three and fewer.

EVLNN's performance in phase one was superior to DE-NN, GA-NN, and TD-BPNN for the hourly, 30-min, and 15-min predictions. EVLNN was also superior to PSO-NN, DE-NN, and GA-NN for the 1-min prediction. However, EVLNN's performance was below PSO-NN in the hourly, 30-min, and 15-min predictions and below TD-BPNN in the 1-min prediction. The results demonstrated EVLNN's generalization capability with sparse or dense datasets, and the findings confirmed that EVLNN is a competitive technique.

EVLNN's performance improved in phase two, showing the approach can generalize better with few features. For the hourly and 30-min time-step predictions, EVLNN trained with three or fewer features obtained better results with lower RMSE and MAE scores and higher Adjusted R^2 scores than the other models. In particular, in the hourly time-step prediction, EVLNN trained with PV as the single input feature performed better than PSO-NN, DE-NN, GA-NN, and TD-BPNN. In terms of percentage, EVLNN's RMSE values

improved by 4.5%, 4.1%, 4.1%, and 17.3%, respectively, MAE value improved by 9.2%, 9.9%, 8.2%, and 20.2%, respectively and Adjusted R^2 value improved by 2.5%, 2.5%, 2.5%, and 12.3%, respectively. Improved results were also seen in the 30-min time-step prediction. For the 15-min time-step prediction, EVLNN's RMSE and MAE scores were better than the other models and comparable to PSO-NN and GA-NN's Adjusted R^2 scores. In the 1-min time-step prediction, EVLNN's error scores were better than PSO-NN, DE-NN, and GA-NN except for TD-BPNN. The TD-BPNN had outperformed EVLNN taking advantage of the power of fully connected networks in the presence of a large dataset. However, TD-BPNN's performance decreases in tandem with sample size, suggesting that a fully connected ANN suffers from difficulties with generalization because of overfitting. The findings provided evidence of EVLNN's generalized capability as the algorithm can attain good results for sparse or dense datasets. By selecting fewer and essential features, EVLNN achieved even higher accuracy.

The experiments tested EVLNN's ability to generalize and predict accurately. It is clear that the strength of EVLNN lies in a smaller scale of feature subsets, especially when EVLNN was trained with PV as the single input feature, the model's performance was more superior. In scenarios where acquiring meteorological data can be time-consuming and costly, EVLNN is a preferred approach. Its ability to generalize well with fewer features makes it more straightforward and less expensive to implement.

7.4. Future Work

EVLNN is designed to locate parsimonious ANN with improved generalizability and interpretability. However, there are shortcomings in the EVLNN framework if it is used to solve optimization problems based on real parameters. One area of improvement concerns the handling of higher dimensionality problems. The design of EVLNN's matrix encoding scheme does not scale well for high-dimensional problems, as observed in EVLNN's low peak ratio score at an accuracy level of 1.0E-03 and better for Sphere-30D, Brown-30D, Rastrigin-30D, Griewank-30D, Shubert-3D, and Vincent-3D benchmark functions. The

genotypic crossovers performed for real parameter optimization at the matrices level do not generate sufficient novel genetic material to create new solutions. One improvement for future work is to take the higher dimensional multimodal problem and reduce it to multiple low-dimensional multimodal problems. Instead of recombining two single large chromosome matrices, the main chromosomes matrix can be subdivided into smaller sub-chromosome structures or subsets for recombination at the matrix subset level. The recombined matrix at the subset level can then be recombined at a higher level to form their original dimension as the new offspring. This way, more alleles in the chromosome matrix can be enhanced with new genetic materials to search for novel solutions.

Another area of improvement is enhancing the diversity tracking mechanism for adaptive crossover and mutation. The diversity measure provides insight into the species' evolution and can be used to automatically tune selective genetic parameters to improve the algorithm's search behavior making the algorithm adaptive. This can be achieved by modifying the fitness function to incorporate elements of the diversity information, species size, and MSE into the learning to influence the search operation.

The third area for future work is integrating EVLNN's solar irradiance forecast ability with Hadoop's energy prediction. The research has laid the groundwork with a predictive model capable of both demand-side and supply-side energy prediction. Future work can bridge the gap between these two engineering problems by using EVLNN to aid the data center transition to renewables. An integrated solution that can predict and accurately balance power consumption demand with renewable energy supply can accelerate the decarbonization of data centers and begin the transition of this industry to a more sustainable future.

The fourth area is the further analysis and comparison of EVLNN with the other state-of-the-art time-delay models. Time-Delay Neural Networks (TDNN) allow the networks to have a finite dynamic response to time series data by introducing a delay at the network's input. The EVLNN design is a special case of TDNN that channels its output back into the input nodes at a delay of up to several time-steps. Competitive analysis of these two

architectures over diverse datasets and input features deserves a thorough investigation in future work.

Lastly, further investigation on the predictive accuracy of EVLNN for other renewable sources such as wind and geothermal energy could be conducted. A model that can predict a diverse mix of renewable sources could help to accelerate the higher penetration of renewables into the traditional grid.

References

[1] A. Shehabi et al., “United States Data Center Energy Usage Report,” 2016. https://eta.lbl.gov/publications/united-states-data-center-energy

[2] A. Shehabi, S.J. Smith, E. Masanet, J. Koomey “Data center growth in the United States: decoupling the demand for services from electricity use,” Environmental Research Letters, Vol. 13, No. 12, 2018, p. 124030. DOI: https://doi.org/10.1088/1748-9326/aaec9c

[3] George Kamiya, “Data Centres and Data Transmission Networks,” International Energy Agency, June 2020. https://www.iea.org/reports/data-centres-and-data-transmission-networks

[4] https://yearbook.enerdata.net/electricity/electricity-domestic-consumption-data.html

[5] J. Nicola. “How to stop data centres from gobbling up the world's electricity.” Nature, vol. 561, no. 7722, Sept. 2018, pp. 163+.

[6] Cisco, “Cisco Global Cloud Index: Forecast and methodology, 2016–2021 white paper” (Cisco, document 1513879861264127, 2018).

[7] https://www.irena.org/publications/2020/Jun/Renewable-Power-Costs-in-2019

[8] P. Banerjee, C. D. Patel, C. Bash and P. Ranganathan, “Sustainable data centers: Enabled by supply and demand side management,” 2009 46th ACM/IEEE Design Automation Conference, 2009, pp. 884-887, doi: 10.1145/1629911.1630138.

[9] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and Chris Hyser, “Renewable and cooling aware workload management for sustainable data centers,” SIGMETRICS Perform. Eval. Rev. 40, 1 (June 2012), pp. 175–186. DOI:https://doi.org/10.1145/2318857.2254779

[10] S. Kwon, “Ensuring renewable energy utilization with quality of service guarantee for energy-efficient data center operations,” Applied Energy, vol. 276, 2020, 115424, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2020.115424.

[11] L. Bird, M. Milligan, and D. Lew, “Integrating Variable Renewable Energy: Challenges and Solutions,” National Renewable Energy Laboratory, Technical Report Prepared under Task No. WE11.0820, 2013.

[12] T. N. Le, Z. Liu, Y. Chen, and C. Bash, “Joint capacity planning and operational management for sustainable data centers and demand response,” In Proceedings of the

Seventh International Conference on Future Energy Systems (e-Energy '16). Association for Computing Machinery, New York, NY, USA, Article 16, pp. 1–12. DOI:https://doi.org/10.1145/2934328.2934344

[13] S. Pelley, D. Meisner and T. F. Wenisch, and J. W. Vangilder, "Understanding and abstracting total data center power," In Workshop on Energy-Efficient Design, 2009.

[14] IRENA (2019), Solutions to Integrate High Shares of Variable Renewable Energy. A Report from the IRENA to the G20 Energy Transitions Working Group (ETWG). International Renewable Energy Agency.

[15] S. Ghemawat, H. Gobioff, S-T Leung, "The Google File System," Proceedings of the 19th ACM Symposium on Operating Systems Principles, ACM, Bolton Landing, NY (2003), pp. 20-43.

[16] J. Dean J, S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun. ACM, 51 (1) (2008), pp. 107-113.

[17] V. K. Vavilapalli et al., "Apache Hadoop YARN: yet another resource negotiator," SOCC '13: Proceedings of the 4th annual Symposium on Cloud ComputingOctober 2013 Article No.: 5 pp. 1–16. https://doi.org/10.1145/2523616.2523633

[18] A. Thusoo et al., "Data Warehousing and Analytics Infrastructure at Facebook," SIGMOD'10, Indianapolis, Indiana, USA, June 6–10, 2010.

[19] Bernard Marr, Big Data in Practice: How 45 Successful Companies Used Big Data Analytics to Deliver Extraordinary Results. Chapter 38, Google: How Big Data is at the Heart of Google's Business Model. Book Editor(s):Bernard Marr. First published: 01 April 2016. https://doi.org/10.1002/9781119278825.ch38

[20] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, 2010, pp. 1-10, doi: 10.1109/MSST.2010.5496972.

[21] Carbon and energy impact, https://www.fb-carbon.com/pdf/download2013.pdf, 2013.

[22] N. Sönnichsen, "Facebook electricity use worldwide 2011-2019," Oct 15, 2020.

[23] https://sustainability.fb.com/wp-content/uploads/2019/08/2018-Sustainability-Data-Disclosure.pdf

[24] https://services.google.com/fh/files/misc/google_2019-environmental-report.pdf

[25] N. Sönnichsen, "Google's energy consumption 2011-2018," Oct 10, 2019.

[26] https://www.fiercetelecom.com/telecom/google-plans-to-invest-13-billion-u-s-2019,

[27] https://datacenternews.asia/story/aws-and-azure-spending-big-bucks-on-data-centres-to-get-ahead

[28] https://www.crn.com/news/data-center/amazon-s-data-center-offensive-continues-in-world-s-largest-market

[29] https://www.datacenterdynamics.com/en/news/tencent-plans-70bn-spree-giant-data-centers-and-infrastructure-cloud-and-ai/

[30] A. Andrae and T. Edler, "On global electricity usage of communication technology: trends to 2030," Challenges 2015, 6 117–57; doi:10.3390/challe6010117

[31] 2012 Key World Energy Statistics, International Energy Agency, 2012.

[32] European Union's (EU's) EURECA (EU Resource Efficiency Coordination Action) Project, 2017. https://www.dceureca.eu

[33]Internet usage worldwide - statistics & facts (2021). https://www.statista.com/topics/1145/internet-usage-worldwide/

[34]F. Bordage, GreenIT (2019), https://www.greenit.fr/wp-content/uploads/2019/11/GREENIT_EENM_etude_EN_accessible.pdf

[35] E. Masanet, N. Shehabi, N. Lei, S. Smith and J. Koomey, "Recalibrating global data center energy-use estimates," DOI: 10.1126/science.aba3758, Science 367 (6481), pp. 984-986, 2020.

[36] https://www.google.com/about/datacenters/efficiency/

[37]https://web.archive.org/web/20160312205123/http://www.opencompute.org/learn/energy-efficiency/

[38] https://journal.uptimeinstitute.com/is-pue-actually-going-up/

[39] S. Ratka and F. Boshell, "The nexus between data centres, efficiency and renewables: a role model for the energy transition," June 26, 2020.

[40] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and Chris Hyser, "Renewable and cooling aware workload management for sustainable data centers," SIGMETRICS Perform. Eval. Rev. 40, 1 (June 2012), pp. 175–186. DOI:https://doi.org/10.1145/2318857.2254779

[41] E. Oró, V. Depoorter, A. Garcia and J. Salom, "Energy efficiency and renewable energy integration in data centres," Renewable and Sustainable Energy Reviews, vol. 42, 2015, pp. 429-445, https://doi.org/10.1016/j.rser.2014.10.035.

[42] C. Koronen, M. Åhman and L.J. Nilsson, "Data centers in future European energy systems—energy efficiency, integration and policy," Energy Efficiency, 2020. vol. 13(1), pp. 129-144.

[43] Z. Liu, H. Yu, R. Liu, M. Wang and C. Li, "Configuration optimization model for data-center-park-integrated energy systems under economic, reliability, and environmental considerations," Energies, 2020, vol. 13(2), 448, doi:10.3390/en13020448.

[44]https://news.nus.edu.sg/nus-and-ntu-launch-first-of-its-kind-tropical-data-centre-testbed/

[45] R. Ascierto, A. Lawrence, "Five data center trends for 2021", UI Intelligence report 42, Dec 2020.

[46]https://sustainability.aboutamazon.com/environment/sustainable-operations/renewable-energy

[47] https://www.google.com/about/datacenters/renewable/

[48]https://azure.microsoft.com/en-gb/global-infrastructure/sustainability/#carbon-benefits

[49] https://sustainability.fb.com/data-centers/

[50] Centre for Climate and Energy Solutions, https://www.c2es.org/content/renewable-energy/

[51] SSG (2017), "Power in the Data Center and its Cost Across the U.S.," Site Selection Group. https://info.siteselectiongroup.com/blog/power-in-the-data-center-and-its-costs-across-the-united-states

[52] T. Hong, P. Pinson, Y. Wang R. Weron, D. Yang, and H. Zareipour, "Energy Forecasting: A Review and Outlook," IEEE Open Access Journal of Power and Energy, 2020, 7. 10.1109/OAJPE.2020.3029979.

[53] N. Zhu, L. Rao, X. Liu, and J. Liu, "Handling more data with less cost: Taming power peaks in MapReduce clusters," in Proc. APSYSWorkshop, 2012, pp. 3:1–3:6. DOI: 10.1145/2349896.2349899

[54] B. Feng, J. Lu, Y. Zhou, and N. Yang, "Energy efficiency for MapReduce workloads: An in-depth study," in Proc. 23rd ADC, Darlinghurst, Australia, 2012, vol. 124, pp. 61–70.

[55] D. Cano, J. M. Monget, M. Albuisson, H. Guillard, N. Regas, and L. Wald, "A method for the determination of the global solar radiation from meteorological satellite data," Solar Energy, 1986, vol. 37, pp. 31-39.

[56] M. Ehrenforfer, "Spectral numerical weather prediction models," Society for Industrial and Applied Mathematics 2012.

[57] H. Wang and Y. Cao, "An Energy Efficiency Optimization and Control Model for Hadoop Clusters," in IEEE Access, vol. 7, pp. 40534-40549, 2019, doi: 10.1109/ACCESS.2019.2907018.

[58] Y. Liang and Z. Hu, "Prediction Method of Energy Consumption Based on Multiple Energy-Related Features in Data Center," 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 2019, pp. 140-146, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00030.

[59] A. Mulyadi and E. C. Djamal, "Sunshine Duration Prediction Using 1D Convolutional Neural Networks," 2019 6th International Conference on Instrumentation, Control, and Automation (ICA), Bandung, Indonesia, 2019, pp. 77-81, doi: 10.1109/ICA.2019.8916751.

[60] G. Capizzi, C. Napoli and F. Bonanno, "Innovative Second-Generation Wavelets Construction with Recurrent Neural Networks for Solar Radiation Forecasting," IEEE Transactions on neural networks and learning systems, vol. 23, no. 11, pp. 1805-1815, 2012.

[61] A. Abayomi-Alli, M. O. Odusami, O. Abayomi-Alli, S. Misra and G. F. Ibeh, "Long Short-Term Memory Model for Time Series Prediction and Forecast of Solar Radiation and other Weather Parameters," 19th International Conference on Computational Science and Its Applications (ICCSA), 2019, pp. 82-92.

[62] M. Hagan, H. Demuth, M. Beale and O. De Jesús. Neural Network Design (2nd Edition), 2014.

[63] X. Zhou, A. K. Qin, Y. Sun and K. C. Tan, "A Survey of Advances in Evolutionary Neural Architecture Search," 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 950-957, doi: 10.1109/CEC45853.2021.9504890.

[64] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning representations by back-propagating errors," Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0

[65] H. Pham and M. Y. Guan, "Efficient neural architecture search via parameter sharing," arXiv preprint arXiv:1802.03268, 2018.

[66] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, vol. 14, pp. 76-86.

[67] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," arXiv preprint arXiv:1806.09055, 2018.

[68] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," in IEEE Transactions on Neural Networks, vol. 9(5), pp. 1054-1054, Sept. 1998, doi: 10.1109/TNN.1998.712192.

[69] B. Zoph, and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," Nov 2016, 2016arXiv161101578Z

[70] H. Pham and M. Y. Guan, "Efficient neural architecture search via parameter sharing," arXiv preprint arXiv:1802.03268, 2018.

[71] Y. Chen, G. Meng, Q. Zhang, and S. Xiang, "Renas: Reinforced evolutionary neural architecture search," in Proc. of the IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4787–4796.

[72] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," IEEE Trans. Evol. Comput., vol. 20, no. 4, pp. 606–626, 2015

[73] J. Holland, Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.

[74] R. Storn and K. Price, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, pp. 341–359, 1997.

[75] J. Kennedy and R. Eberhart, "Particle swarm optimization," in IEEE International Conference on Neural Networks. Perth, Australia: IEEE Service Center, 1995, pp. 1942–1948.

[76] J.S Arora, "Introduction to Optimum Design," 4th Ed. Academic Press, 2017, https://doi.org/10.1016/C2013-0-15344-5

[77] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in Proceedings of the AAAI conference on artificial intelligence, vol. 33, 2019, pp. 4780–4789.

[78] L. Peng, S. Liu, R. Liu, and L. Wang, “Effective long short-term memory with differential evolution algorithm for electricity price prediction,” Energy, vol. 162, pp. 1301–1314, 2018.

[79] C. P. Joy, G. Pillai, Y. Chen and K. Mistry, “Micro-Genetic Algorithm Embedded Multi-Population Differential Evolution based Neural Network for Short-Term Load Forecasting,” 2021 56th International Universities Power Engineering Conference (UPEC), 2021, pp. 1-4, doi: 10.1109/UPEC50034.2021.9548262.

[80] M. Dayarathna, Y. Wen and R. Fan, “Data Center Energy Consumption Modeling: A Survey,” in IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 732-794, Firstquarter 2016, doi: 10.1109/COMST.2015.2481183.

[81] D. Mytton, “Renewable energy for data centers – renewable energy certificates, power purchase agreements and beyond,” Uptime Institute Intelligence Report 44, Feb 2021.

[82] C. Sweeney, R.J. Bessa, J. Browell, and P. Pinson, “The future of forecasting for renewable energy,” Wiley Interdiscip Rev Energy Environ, vol. 9 (2) (2020), p. e365. DOI: 10.1002/wene.365

[83] A. Lebedys, D.Akande, N. Elhassan, G. Escamilla, A. Whiteman and I. Arkhipova, “Renewable energy statistics 2021,” https://www.irena.org/publications/2021/Aug/Renewable-energy-statistics-2021

[84] F. Shabestari, A. M. Rahmani, N. J. Navimipour, and S. Jabbehdari, “A taxonomy of software-based and hardware-based approaches for energy efficiency management in the Hadoop,” Journal of Network and Computer Applications, 2019, vol. 126, pp. 162-177. https://doi.org/10.1016/j.jnca.2018.11.007.

[85] A. Thakkar, K. Chaudhari, M. Shah, "A Comprehensive Survey on Energy-Efficient Power Management Techniques,"(2020) Procedia Computer Science, 167, pp. 1189-1199. doi: 10.1016/j.procs.2020.03.432.

[86] M. Usama, M. Liu, M. Chen, “Job schedulers for Big data processing in Hadoop environment: testing real-life schedulers using benchmark programs,” Digital Commun. Netw., vol. 3(4), (2017), pp. 260-273.

[87] M. Varga, A. Petrescu-Nita, F. Pop, “Deadline scheduling algorithm for sustainable computing in Hadoop environment,” Comput. Secur., 76 (2018), pp. 354-366.

[88] K. Krish, M. Iqbal, M. Rafique, and A. Butt, “Towards energy awareness in Hadoop,” Fourth International Workshop on Network-Aware Data Management (NDM), 2014, pp. 16-22.

[89] S. Kulkarni, “Cooling hadoop: temperature aware schedulers in data centers,” Auburn University, Alabama, 2013.

[90] L. Mashayekhy, M. Nejad, D. Grosu, D. Lu, and W. Shi, “Energy-Aware Scheduling of MapReduce Jobs,” Proceedings - 2014 IEEE International Congress on Big Data, DOI:10.1109/BigData.Congress.2014.15.

[91] Y. Shao, C. Li, J. Gu, J. Zhang, and Y. Luo, “Efficient jobs scheduling approach for big data applications,” Computers and Industrial Engineering, vol. 117, March 2018, pp. 249-261.

[92] J. Leverich and C. Kozyrakis, “On the energy (in)efficiency of Hadoop clusters,” ACM SIGOPS Operating Systems Review 2010, vol. 44, no. 1, pp. 61-65.

[93] W. Lang and J. M. Patel, “Energy Management for MapReduce Clusters,” Proc. VLDB Endow., vol. 3, no. 1-2, pp. 129-139, 2010.

[94] H. Amur, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, “Robust and flexible power proportional storage,” SoCC '10 Proceedings of the 1st ACM symposium on Cloud computing, 2010, pp. 217-228.

[95] B Lin, S Li, X Liao, Q Wu, and S Yang, “eStor energy efficient and resilient data center storage,” International Conference on Cloud and Service Computing (CSC), 2011, pp. 366 – 371.

[96] H. H. Le, S. Hikida and H. Yokota, “Efficient gear-shifting for a power-proportional distributed data-placement method,” 2013 IEEE International Conference on Big Data, Silicon Valley, CA, 2013, pp. 76-84, doi: 10.1109/BigData.2013.6691557.

[97] R. Kaushik, M. Bhandarkar, and K. Nahrstedt, “Evaluation and analysis of GreenHDFS: a self-adaptive, energy-conserving variant of the Hadoop distributed file system,” IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), 2010, pp. 274 - 287.

[98] R. Kaushik, T. Abdelzaher, R. Egashira, and K. Nahrstedt, "Predictive data and energy management in GreenHDFS," International Green Computing Conference and Workshops (IGCC), 2011, pp. 1-9.4

[99] K.M. Attia, M.A. El-Hosseini, H.A. Ali, "Dynamic power management techniques in multi-core architectures: a survey study," Ain Shams Eng. J., vol. 8 (3) (2017), pp. 445-456. https://doi.org/10.1016/j.asej.2015.08.010

[100] T. Wirtz and R. Ge, "Improving MapReduce energy efficiency for computation intensive workloads," 2011 International Green Computing Conference and Workshops, Orlando, FL, 2011, pp. 1-8, doi: 10.1109/IGCC.2011.6008564.

[101] S. Hou, W. Ni, S. Zhao, B. Cheng, S. Chen, and J. Chen, "Frequency-Reconfigurable Cloud Versus Fog Computing: An Energy-Efficiency Aspect," in IEEE Transactions on Green Communications and Networking, vol. 4, no. 1, pp. 221-235, March 2020, doi: 10.1109/TGCN.2019.2953891.

[102] B. Fan, W. Tantisiriroj, L. Xiao and G. Gibson, "DiskReduce: RAID for data-intensive scalable computing," PDSW '09 Proceedings of the 4th Annual Workshop on Petascale Data Storage, 2009, pp. 6-10.

[103] Z. Cheng, Z. Luan, Y. Meng, Y. Xu, D. Qian, A. Roy, N. Zhang and G. Guan, "ERMS: an elastic replication management system for HDFS," IEEE International Conference on Cluster Computing Workshops 2012, pp. 32-40.

[104] Y. Wei and Y. W. Foo, "A Cost-Effective and Reliable Cloud Storage," 2014 IEEE 7th International Conference on Cloud Computing, 2014, pp. 938-939, doi: 10.1109/CLOUD.2014.132.

[105] Y. Wei, Y. W. Foo, K. C. Lim and F. Chen, "The Auto-configurable LDPC Codes for Distributed Storage," 2014 IEEE 17th International Conference on Computational Science and Engineering, 2014, pp. 1332-1338, doi: 10.1109/CSE.2014.254.

[106] H. Wang and Y. Cao, "An Energy Efficiency Optimization and Control Model for Hadoop Clusters," in IEEE Access, vol. 7, pp. 40534-40549, 2019, doi: 10.1109/ACCESS.2019.2907018.

[107] Y. Liang and Z. Hu, "Prediction Method of Energy Consumption Based on Multiple Energy-Related Features in Data Center," 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing &

Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 2019, pp. 140-146, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00030.

[108] T. R. Toha, A. S. M. Rizvi, J. Noor, M. A. Adnan and A. B. M. A. Al Islam, "Towards Greening MapReduce Clusters Considering Both Computation Energy and Cooling Energy," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 4, pp. 931-942, 1 April 2021, doi: 10.1109/TPDS.2020.3029724.

[109] Wikimedia, "Wikimedia dump service." Accessed: Oct. 28, 2017. [Online]. Available: https://dumps.wikimedia.org/enwiki/

[110] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F.J. Martinez-de-Pison, F. Antonanzas-Torres, "Review of photovoltaic power forecasting," Solar Energy, 2016, vol. 136, pp. 78-111, https://doi.org/10.1016/j.solener.2016.06.069.

[111] S. Monjoly, M. André, R. Calif, T. Soubdhan, "Forecast Horizon and Solar Variability Influences on the Performances of Multiscale Hybrid Forecast Model," Energies 2019, 12, 2264. https://doi.org/10.3390/en12122264

[112] R. H. Inman, H. T.C. Pedro, C. F.M. Coimbra, "Solar forecasting methods for renewable energy integration," Progress in Energy and Combustion Science, 2013, Vol. 39, pp. 535-576.

[113] M. Diagne, M. David, P. Lauret, J. Boland, N. Schmutz, "Review of solar irradiance forecasting methods and a proposition for small-scale insular grids," Renewable and Sustainable Energy Reviews 2013, Vol. 27, pg. 65–76.

[114] D. Z. Yang, J. Kleissl, C. A. Gueymard, H. T.C. Pedro, C. F.M. Coimbra, "History and trends in solar irradiance and PV power forecasting: A preliminary assessment and review using text mining," Solar Energy, 2018, vol. 168, pp. 60–101.

[115] A. Hammer, D. Heinemann, E. Lorenz, "Short term forecasting of solar radiation based on satellite data," In:EUROSUN2004 (ISES Europe Solar Congress), 2004, pp. 841–8.

[116] F. Yang, H-L Pan, SK Krueger, "Evaluation of the NCEP global forecast system at the ARM SGP site," Monthly Weather Review, 2006, vol. 134(12), pp. 3668-3690.

[117] D.P. Larson, L. Nonnenmacher, C.F.M Coimbra, "Day-ahead forecasting of solar power output from photovoltaic plants in the American Southwest," Renew. Energy vol. 91, 2016, pp. 11–20. http://dx.doi.org/10.1016/j.renene.2016.01.039.

[118] P.A. Jimenez, J.P. Hacker, J. Dudhia, S.E. Haupt, J.A. Ruiz-Arias, C.A. Gueymard, G. Thompson, T. Eidhammer, A. Deng, "WRF-Solar: Description and clear-sky assessment of an augmented NWP model for solar power prediction," Bull. Am. Meteorol. Soc., 2016, vol. 97, pp. 1249-1264, 10.1175/BAMS-D-14-00279.1

[119] E. Lorenz, J. Hurka, D. Heinemann, H.G. Beyer, "Irradiance forecasting for the power prediction on grid-connected photovoltaic systems," IEEE J. Select. Top. Appl. Earth Observ. Remote Sensing, 2009, vol. 2, no. 1, pp. 2-10.

[120] A. C. Lorenc, and M. Jardak, "A comparison of hybrid variational data assimilation methods for global nwp," Quarterly Journal of the Royal Meteorological Society, 2018, vol. 144(717), pp. 2748–2760. DOI: https://doi.org/10.1002/qj.3401

[121] RM Goody, YL Yung, "Atmospheric radiation: theoretical basis," 2nd ed. Oxford University Press; 1995.

[122] J. A. Duffie and A. B. William, "Solar Engineering of Thermal Processes 4th Edition". John Wiley & Sons, Inc. 2013.

[123] JD Tarpley, "Estimating incident solar radiation at the surface from geostationary satellite data," Journal of Applied Meteorology 1979, vol. 18(9), pp. 1172-81.

[124] C. Rigollier, M. Lefèvre, L. Wald, "The method heliosat-2 for deriving short wave solar radiation from satellite images," Solar Energy, 2004, vol. 77, pp. 159–69.

[125] D.W. Slater, C.N. Long, T.P. Tooman, "Total sky imager/whole sky imager cloud fraction comparison," In: Eleventh ARM ScienceTeam Meeting Proceedings, March 2001, Atlanta, Georgia.

[126] S. Quesada-Ruiz, Y. Chu, J. Tovar-Pescador, H.T.C. Pedro, C.F.M. Coimbra, "Cloud-tracking methodology for intra-hour DNI forecasting," Solar Energy, vol. 102, 2014, pp. 267-275. https://doi.org/10.1016/j.solener.2014.01.030.

[127] R. Marquez and C.F.M. Coimbra, "Intra-hour DNI forecasting based on cloud tracking image analysis," Solar Energy vol. 91, 2013, pp. 327-336.

[128] C. W. Chow, B. Urquhart, J. Kleissl, M. Lave, A. Dominguez, J. Shields, B. Washom, "Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed," Solar Energy vol. 85(11), 2011, pp. 2881-2893.

[129] C. Rigollier, O. Bauer, L. Wald, "On the clear sky model of the ESRA - European Solar Radiation Atlas - with respect to the Heliosat method," Solar Energy 2000, vol. 68(1), pp. 33-48.

[130] H. Yang, B. Kurtz, D. Nguyen, B. Urquhart, C.Wai Chow, M. Ghonima, J. Kleissl, "Solar irradiance forecasting using a ground-based sky imager developed at UC San Diego," Solar Energy, vol. 103, 2014, pp. 502-524.

[131] X-S Yang, "Genetic Algorithms," Nature-Inspired Optimization Algorithms, Elsevier, 2014, pp. 77-87. https://doi.org/10.1016/B978-0-12-416743-8.00005-1.

[132] P. Bacher, H. Madsen, H. Nielsen, "Online short-term solar power forecasting," Solar Energy, 2004, vol. 83, pp. 1772–1783.

[133] Y. Li, Y. Su, L. Shu, "An ARMAX model for forecasting the power output of a grid connected photovoltaic system," Renew. Energy, 2014, vol. 66, pp. 78–89.

[134] Y. Chu, B. Urguhart, S. Gohari, H. Pedro, J. Kleissl, C. Coimbra, "Short-term reforecasting of power output from a 48MWe solar PV plant," Solar Energy, 2015, vol. 112, pp. 68–77.

[135] D.J. Gagne II, A. McGovern, S.E. Haupt, J.K Williams, "Evaluation of statistical learning configurations for gridded solar irradiance forecasting," Solar Energy vol. 150, 2017, pp. 383–393. http://dx.doi.org/10.1016/j.solener.2017.04.031.

[136] H. M. Diagne, M. David, J. Boland, N. Schmutz and P. Lauret, "Post-processing of solar irradiance forecasts from WRF model at Reunion Island," Solar Energy, 2014, vol. 105, pp. 99–108.

[137] M. Noia, C Ratto, and R. Festa, "Solar irradiance estimation from geostationary satellite data: I. statistical models," Solar Energy, 1993, vol. 51(6), pp. 449 – 456.

[138] J. E. Hay and K. J. Hanson, "A satellite-based methodology or determining solar irradiance at the ocean surface during GATE," Bull. American Meteorol. Soc., 1987, vol. 59, 1549.

[139] C. Justus, M. V. Paris, and J. D. Tarpley, “Satellite-measured insolation in the United States, Mexico, and South America,” Remote Sensing of Environment, 1986, vol. 20, pp. 57-83.

[140] C. Voyant, G. Notton, S. Kalogirou, et al., “Machine learning methods for solar radiation forecasting: a review Renewable Energy,” 2017, vol. 105, pp. 569-582.

[141] C. Crisosto, M. Hofmann, R. Mubarak and G. Seckmeyer, “One-Hour Prediction of the Global Solar Irradiance from All-Sky Images Using Artificial Neural Networks,” Energies, MDPI, Open Access Journal, vol. 11(11), pg. 1-16, Oct 2018. DOI: 10.3390/en11112906.

[142] J. Faceira, P. Afonso, P. Salgado, “Prediction of Solar Radiation Using Artificial Neural Networks,” Proceedings of the 11th Portuguese Conference on Automatic Control, pp. 397-406, CONTROLO’2014.

[143] S. Nurcahyo, F. Nhita, Adiwijaya, “Rainfall Prediction in Kemayoran Jakarta Using Hybrid Genetic Algorithm (GA) and Partially Connected Feedforward Neural Network (PCFNN),” International Conference on Information and Communication Technology (ICoICT), 2014.

[144] D. Su, E. Batzelis, B. Pal, “Machine Learning Algorithms in Forecasting of Photovoltaic Power Generation,” International Conference on Smart Energy Systems and Technologies (SEST), 2019.

[145] H. Wang, Z. Lei, X. Zhang, B. Zhou and J. Peng, “A review of deep learning for renewable energy forecasting,” Energy Conversion and Management, vol. 198, 2019, Article number 111799.

[146] K. Kaba, M. Sarıgül, M. Avcı, H. M. Kandırmaz, “Estimation of daily global solar radiation using deep learning model,” Energy 162 (2018) pp. 126-135.

[147] A. Abayomi-Alli, M. O. Odusami, O. Abayomi-Alli, S. Misra and G. F. Ibeh, “Long Short-Term Memory Model for Time Series Prediction and Forecast of Solar Radiation and other Weather Parameters,” 19th International Conference on Computational Science and Its Applications (ICCSA), 2019, pp. 82-92.

[148] A. Muhammad, J. M. Lee, S. W. Hong, S. J. Lee and E. H. Lee, “Deep Learning Application in Power System with a Case Study on Solar Irradiation Forecasting,” 2019

International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Okinawa, Japan, 2019, pp. 275-279, doi: 10.1109/ICAIIC.2019.8668969.

[149] S. Ghimire, R.C. Deo, N. Raj, J. Mi, "Deep solar radiation forecasting with convolutional neural network and long short-term memory network algorithms," Applied Energy, vol. 253 (2019).

[150] K. Wang, X. Qi, H. Liu, "A comparison of day-ahead photovoltaic power forecasting models based on deep learning neural network," Applied Energy vol. 251 (2019).

[151] N. Dong, J-F Chang, A-G Wu, Z-K Gao, "A novel convolutional neural network framework based solar irradiance prediction method," International Journal of Electrical Power & Energy Systems 114, Jan 2020, 105411. https://doi.org/10.1016/j.ijepes.2019.105411.

[152] S. Ghimire, R.C. Deo, N.J. Downs and N. Raj, "Self-adaptive differential evolutionary extreme learning machines for long-term solar radiation prediction with remotely-sensed MODIS satellite and Reanalysis atmospheric products in solar-rich cities," Remote Sensing of Environment, vol. (212), June 2018, pp. 176-198.

[153] D. Guijo-Rubio, A.M. Durán-Rosal, P.A. Gutiérrez, A.M. Gómez-Orellana, C. Casanova-Mateo, J. Sanz-Justo, S. Salcedo-Sanz, C. Hervás-Martínez, "Evolutionary artificial neural networks for accurate solar radiation prediction," Energy, vol. 210, 2020.

[154] http://www.soda-pro.com/web-services/radiation/cams-mcclear

[155] https://solargis.com/

[156] X. Meng, A. Xu, W. Zhao, H. Wang, C. Li and H. Wang, "A new PV generation power prediction model based on GA-BP neural network with artificial classification of history day," International Conference on Power System Technology, POWERCON 2018.

[157] S. Jaidee and W. Pora, "Very Short-Term Solar Power Forecasting Using Genetic Algorithm Based Deep Neural Network," 4th International Conference on Information Technology (InCIT), Nov 2019.

[158] K. Bashtova et al. (2022) Application of the Topological Gradient to Parsimonious Neural Networks. In: Tuovinen T., Periaux J., Neittaanmäki P. (eds) Computational Sciences and Artificial Intelligence in Industry. Intelligent Systems, Control and Automation: Science and Engineering, vol 76. Springer, Cham. https://doi.org/10.1007/978-3-030-70787-3_5.

[159] D. Elizondo, E. Fiesler, "A survey of partially connected neural networks," Int J Neural Syst. 1997 Oct-Dec;8(5-6):535-58. doi: 10.1142/s0129065797000513.

[160] S. Kang and C. Isik, "Partially connected feedforward neural networks structured by input types," in IEEE Transactions on Neural Networks, vol. 16(1), pp. 175-184, Jan. 2005, doi: 10.1109/TNN.2004.839353.

[161] S. Desai, A. Strachan, "Parsimonious neural networks learn interpretable physical laws," Scientific Reports 11, Article number: 12761 (2021). https://doi.org/10.1038/s41598-021-92278-w.

[162] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, Reading, MA, 1989.

[163] A. Eiben and J. Smith, "Introduction to Evolutionary Computing." Springer. 2007.

[164] M. Ahmad, M. Abdullah, and D. Han, "A Novel Encoding Scheme for Complex Neural Architecture Search," *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, JeJu, Korea (South), 2019, pp. 1-4, doi: 10.1109/ITC-CSCC.2019.8793329.

[165] S. Ronald, "Robust encodings in genetic algorithms: a survey of encoding issues," IEEE International Conference on Evolutionary Computation, 1997, pp. 43-48.

[166] T. Hu, J. L. Payne, W. Banzhaf, J. H. Moore, "Evolutionary dynamics on multiple scales: a quantitative analysis of the interplay between genotype, phenotype, and fitness in linear genetic programming," Genet Program Evolvable Mach (2012) vol. 13, pp. 305–337. DOI 10.1007/s10710-012-9159-4.

[167] G. Bakırlı, D. Birant, A. Kut, "An incremental genetic algorithm for classification and sensitivity analysis of its parameters," Expert Systems with Applications, vol. 38(3), March 2011, pg. 2609-2620. doi: 10.1016/j.eswa.2010.08.051.

[168] Siddharth Sharma, Simone Sharma, A. Athaiya, "Activation functions in neural networks," International Journal of Engineering Applied Sciences and Technology, 2020, vol. 4(12), ISSN No. 2455-2143, pp. 310-316.

[169] J.A. Coyne and H.A. Orr, "Speciation," 2004, Sunderland, MA: Sinauer Associates vol. 545.

[170] H. D. Rundle and P. Nosil, "Ecological speciation," Ecology Letters, (2005) vol. 8, pp. 336–352. https://doi.org/10.1111/j.1461-0248.2004.00715.x

[171] W. D. Allmon, "A causal analysis of stages in allopatric speciation," Oxford surveys in evolutionary biology, vol. 8, pp. 219-219.

[172] D. I. Bolnick and B. M. Fitzpatrick, "Sympatric Speciation: Models and Empirical Evidence," Annual Review of Ecology, Evolution, and Systematics, 2017, vol. 38, pp. 459-487.

[173] K. Kaneko, "Symbiotic sympatric speciation: consequence of interaction-driven phenotype differentiation through developmental plasticity," The Society of Population Ecology and Springer-Verlag Tokyo, 2002, vol. 44, pp. 71–85.

[174] G. S. van Doorn, "Sexual selection and sympatric speciation," CEES Progress Report, 2004.

[175] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Edition, Prentice-Hall International, Upper Saddle River, N.J, 1999

[176] A. Hadjiivanov and A. Blair, "Complexity-based speciation and genotype representation for neuroevolution," *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, BC, 2016, pp. 3092-3101, doi: 10.1109/CEC.2016.7744180.

[177] K. Jebari, "Selection Methods for Genetic Algorithms," International Journal of Emerging Sciences, 2013, vol. 3, pp. 333-344.

[178] T. Pencheva, K. Atanassov, and A. Shannon, "Modelling of a stochastic universal sampling selection operator in genetic algorithms using generalized nets," Proceedings of the 10th International Workshop on Generalized Net, 2009.

[179] W. M. Spears, "Simple Subpopulation Schemes," Proceedings of the Third Annual Conference on Evolutionary Programming, 1994, pp. 296-307.

[180] S. F. Galan, O. J. Mengshoel, and R. Pinter, "A Novel Mating Approach for Genetic Algorithms," Evolutionary Computation, 2012, vol. 21(2).

[181] E. Osaba, R. Carballedo, F. Diaz, E. Onieva, I. de la Iglesia and A. Perallos, "Crossover versus Mutation: A Comparative Analysis of the Evolutionary Strategy of Genetic Algorithms Applied to Combinatorial Optimization Problems," The Scientific World Journal, Hindawi Publishing Corporation, vol. 2014, Article ID 154676, https://doi.org/10.1155/2014/154676.

[182] D. Bhandari, C. Murthy, & S.K. Pal, "Genetic algorithm with elitist model and its convergence," Int. J. Pattern Recogn. vol. 10, pp. 731–747 (1996). https://doi.org/10.1142/S0218001496000438

[183] S. Theodoridis and K. Koutroumbas, (2003) Pattern Recognition. 2nd Edition, Academic Press, San Diego.

[184] C. Mattiussi, M. Waibel and D. Floreano, "Measures of diversity for populations and distances between individuals with highly reorganizable genomes," Evolutionary Computation Journal, 2004, vol. 12(4) pp. 495-515.

[185] M. Črepinšek, S-H Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," ACM Comput. Surv. 45, 3, Article 35 (June 2013). DOI:https://doi.org/10.1145/2480741.2480752

[186] K. Pentoś, "The methods of extracting the contribution of variables in artificial neural network models – Comparison of inherent instability," Computers and Electronics in Agriculture, vol. 127, 2016, pp. 141-146, ISSN 0168-1699. https://doi.org/10.1016/j.compag.2016.06.010.

[187] N. Luíza da Costa, M. Dias de Lima, R. Barbosa, "Evaluation of feature selection methods based on artificial neural network weights," (2021) Expert Systems with Applications, 168, art. no. 114312. https://www.journals.elsevier.com/expert-systems-with-applications. doi: 10.1016/j.eswa.2020.114312.

[188] G. D. Garson, "Interpreting Neural Network Connection Weights," Artificial Intelligence Expert, 1991, vol. 6, pp. 47-51.

[189] Y. Yoon, T. Guimaraes and G. Swales, "Integrating artificial neural networks with rule-based expert systems," Decision Support Systems, 1994, vol. 11(5), pp. 497–507. https://doi.org/p10.1016/0167-9236(94)90021-3.

[190] S.-H. Tsaur, Y.-C. Chiu, and C.-H. Huang, "Determinants of guest loyalty to international tourist hotels—a neural network approach," Tourism Management, 2002, vol. 23(4), pp. 397–405. https://doi.org/10.1016/S0261-5177(01)00097-8.

[191] P. Howes, and N. Crook, "Using input parameter influences to support the decisions of feedforward neural networks," Neurocomputing, 1999. https://doi.org/10.1016/S0925-2312(98)00102-7.

[192] J. Olden and D. Jackson, “Illuminating the “black box”: a randomization approach for understanding variable contributions in artificial neural networks,” Ecological Modelling, vol. 154(1–2), 2002, pp. 135-150, ISSN 0304-3800, https://doi.org/10.1016/S0304-3800(02)00064-9

[193] J. de Ona and C. Garrido, “Extracting the contribution of independent variables in neural network models: a new approach to handle instability,” Neural Comput & Applic, 25:859-869.

[194] Y. W. Foo, C. Goh and Y. Li, “Machine Learning with Sensitivity Analysis to Determine Key Factors Contributing to Energy Consumption in Cloud Data Centers,” 2016 International Conference on Cloud Computing Research and Innovations (ICCCRI), 2016, pp. 107-113, doi: 10.1109/ICCCRI.2016.24.

[195] M. Gevrey, I. Dimopoulos, S. Lek, “Review and comparison of methods to study the contribution of variables in artificial neural network models,” Ecological Modelling, vol. 160(3), 2003, pp. 249-264. https://doi.org/10.1016/S0304-3800(02)00257-0.

[196] I. Dimopoulos, J. Chronopoulos, A. Chronopoulou-Sereli, S. Lek, “Neural network models to study relationships between lead concentration in grasses and permanent urban descriptors in Athens city (Greece),” Ecological Modelling, vol. 120(2–3), 1999, pp. 157-165. https://doi.org/10.1016/S0304-3800(99)00099-X

[197] Y. Dimopoulos, P. Bourret and S. Lek, “Use of Some Sensitivity Criteria for Choosing Networks with Good Generalization Ability,” Neural Processing Letter, 1995, vol.2(6), pp. 1-4.

[198] X. Zeng, D.S. Yeung, “A quantified sensitivity measure for multilayer perceptron to input perturbation,” Neural Computation. 2003, vol. 15(1), pp. 183–212. DOI: 10.1162/089976603321043757

[199] S. Lek, A. Belaud, P. Baran, I. Dimopoulos, and M. Delacoste, “Role of some environmental variables in trout abundance models using neural networks,” Aquatic Living Resources, (1996), vol. 9(1), pp. 23-29. doi:10.1051/alr:1996004

[200] S. Lek, A. Belaud, I. Dimopoulos, J. Lauga, J. Moreau, “Improved estimation, using neural networks, of the food consumption of fish populations,” Marine and Freshwater Research. 1995, vol. 46(8), pp. 1229–1236. DOI: 10.1071/MF9951229

[201] A. T. C. Goh, "Back-propagation Neural Networks for Modelling Complex Systems," Artificial Intelligence in Engineering, 1995, vol. 9, pp. 143-151.

[202] X-D Li, K. Tang, M. N. Omidvar, Z-Y Yang, and K. Qin, "Benchmark Functions for the CEC' 2013 Special Session and Competition on Large-Scale Global Optimization," 2013.

[203] M. Jamil and X-S Yang, "A literature survey of benchmark functions for global optimization problems," Int. Journal of Mathematical Modelling and Numerical Optimisation, vol. 4(2), pp. 150-194 (2013).

[204] X. Li, A. Engelbrecht, and M.G. Epitropakis, "Results of the 2013 IEEE CEC Competition on Niching Methods for Multimodal Optimization," IEEE Congress on Evolutionary Computation, 20-23 June, Cancun, Mexico, 2013.

[205] X. Li, A. Engelbrecht, and M.G. Epitropakis, "Results of the 2015 IEEE CEC Competition on Niching Methods for Multimodal Optimization," IEEE Congress on Evolutionary Computation, 25-28 May, Sendai, Japan, 2015.

[206] Nature-inspired computation and swarm intelligence algorithms, theory and applications, Yang - Elsevier/Academic Press - 2020.

[207] X. Li, A. Engelbrecht, and M.G. Epitropakis, "Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization," Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

[208] https://github.com/mazhar-ansari-ardeh

[209] https://www.sfu.ca/~ssurjano/index.html

[210] http://www.yarpiz.com

[211] J. Liang, B. Y. Qu, P. N. Suganthan and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report, vol. 2012(12), pp. 3–18, 2013.

[212] D. O. Boyer, C. H. Martfnez, N. G. Pedrajas, ”Crossover Operator for Evolutionary Algorithms Based on Population Features,” Journal of Artificial Intelligence Research, vol. 24, pp. 1-48, 2005.

[213] M. N. Omidvar, X. Li and X. Yao, “Cooperative Co-evolution with delta grouping for large scale non-separable function optimization,” 2010, pp. 1-8. DOI: 10.1109/CEC.2010.5585979.

[214] X. Yao, Y. Liu, “Fast Evolutionary Programming,” Proc. 5th Conf. on Evolutionary Programming, 1996.

[215] V. Picheny, T. Wagner, D. Ginsbourger, “A benchmark of kriging-based infill criteria for noisy optimization,” Struct. Multidiscip. Optim., 48 (3) (2013), pp. 607-626

[216] M. Preuss. “Niching the CMA-ES via nearest-better clustering,” In Proceedings of the 12th annual conference companion on Genetic and evolutionary computation (GECCO ’10). ACM, New York, NY, USA, pp. 1711-1718, 2010.

[217] R. Thomsen, “Multimodal optimization using crowding-based differential evolution,” In the IEEE Congress on Evolutionary Computation, 2004. CEC2004, vol.2, pp. 1382-1389, 19-23 June 2004.

[218] J. E. Fieldsend, “Running Up Those Hills: Multimodal search with the niching migratory multi-swarm optimiser,” 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, 2014, pp. 2593-2600, doi: 10.1109/CEC.2014.6900309.

[219] D. Molina, A. Puris, R. Bello and F. Herrera, “Variable mesh optimization for the 2013 CEC Special Session Niching Methods for Multimodal Optimization,” 2013 IEEE Congress on Evolutionary Computation, Cancun, 2013, pp. 87-94, doi: 10.1109/CEC.2013.6557557.

[220] M. G. Epitropakis, Li, X., and Burke, E. K., “A Dynamic Archive Niching Differential Evolution Algorithm for Multimodal Optimization,” IEEE Congress on Evolutionary Computation, 2013. CEC 2013. Cancun, Mexico, pp. 79-86, 2013.

[221] J. Ronkkonen, “Continuous Multimodal Global Optimization with Differential Evolution-Based Methods, Ph.D. thesis,” Lappeenranta University of Technology, 2009.

[222] N. Hansen and A. Ostermeier, “Completely Derandomized Self-Adaptation in Evolution Strategies,” Evolutionary Computation, 2001, 9(2), pp. 159-195.

[223] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," In the 2005 IEEE Congress on Evolutionary Computation, 2005. vol.2, pp.1769-1776, 2-5 Sept. 2005.

[224] S. Bandaru and K. Deb, "A parameterless-niching-assisted bi-objective approach to multimodal optimization," 2013 IEEE Congress on Evolutionary Computation, Cancun, 2013, pp. 95-102, doi: 10.1109/CEC.2013.6557558.

[225] J. E. Fieldsend, "Using an adaptive collection of local evolutionary algorithms for multimodal problems," Soft Comput (2015) 19:1445–1460. DOI 10.1007/s00500-014-1309-6.

[226] J. E. Fieldsend, "Multimodal Optimisation using a Localised Surrogates Assisted Evolutionary Algorithm," in UK Workshop on Computational Intelligence (UKCI 2013), 2013, pp. 88–95.

[227] Y. Zhou and K. Saitou, "An Active Learning Based Niching Method with Sequential Binary Probabilistic Classification and Class Split Threshold Updating," University of Michigan, Ann Arbor.

[228] R. K. Ursem, "Multinational evolutionary algorithms," in Proceedings of the Congress on Evolutionary Computation, 1999, pp. 1633–1640.

[229] J. Zhang, D.-S. Huang, and K.-H. Liu, "Multi-Sub-Swarm Optimization Algorithm for Multimodal Function Optimization," in IEEE Congress on Evolutionary Computation, 2007, pp. 3215–3220.

[230] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, "DACE A Matlab Kriging Toolbox," Technical Report IMM-TR-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, Tech. Rep., 2002.

[231] U. Škvorc, T. Eftimov and P. Korošec, "CEC Real-Parameter Optimization Competitions:Progress from 2013 to 2018," 2019, DOI: 10.1109/CEC.2019.8790158.

[232] D. Willingham (2021). Electricity Load Forecasting for the Australian Market Case Study (https://www.mathworks.com/matlabcentral/fileexchange/31877-electricity-load-forecasting-for-the-australian-market-case-study), MATLAB Central File Exchange. Retrieved December 16, 2021.

[233] N. Maleki, A. Rahmani and M. Conti, “MapReduce: an infrastructure review and research insights,” The Journal of Supercomputing, 2019, vol. 75 pp. 1-69. DOI: 10.1007/s11227-019-02907-5.

[234] S. Mazumdar and S. Dhar, “Hadoop as Big Data Operating System -- The Emerging Approach for Managing Challenges of Enterprise Big Data Platform,” Proceedings of the 2015 IEEE First International Conference on Big Data Computing Service and Applications, pp. 499-505.

[235] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H-A Jacobsen, “Bigbench: Towards an industry standard benchmark for big data analytics,” ACM SIGMOD International Conference on Management of Data, ACM (2013), pp. 1197-1208. DOI:https://doi.org/10.1145/2463676.2463712

[236] V. Chang, “An overview, examples, and impacts offered by emerging services and analytics in cloud computing virtual reality,” Neural Comput. Appl. (2015)

[237] https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html

[238] J. Li, Y. Liu, J. Pan, P. Zhang, W. Chen, and L. Wang, “Map-Balance-Reduce: An improved parallel programming model for load balancing of MapReduce,” Future Generation Computer Systems, vol. 105, 2020, pp. 993-1001, ISSN 0167-739X, https://doi.org/10.1016/j.future.2017.03.013.

[239] C. Xu and W. Zhuang, “Parallel Computing Framework Based on MapReduce and GPU Clusters,” In Proceedings of the 4th International Conference on Computer Science and Application Engineering, 2020. Association for Computing Machinery, New York, NY, USA, Article 73, pp. 1-5. DOI:https://doi.org/10.1145/3424978.3425051.

[240] L. Belcastro, F. Marozzo and D. Talia, “Programming models and systems for Big Data analysis,” International Journal of Parallel, Emergent and Distributed Systems, vol. 34(6), 2019, pp. 632-652, DOI: 10.1080/17445760.2017.1422501.

[241]https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

[242] Faraz, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar, “Tarazu: optimizing MapReduce on heterogeneous clusters,” SIGARCH Comput. Archit. News 40, 1 (March 2012), pp. 61–74. DOI:https://doi.org/10.1145/2189750.2150984

[243] Monitoring with Ganglia, by Alex Dean et al., November 2012, Publisher(s): O'Reilly Media, Inc. ISBN: 9781449329709.

[244] M. L. Massie, B. N. Chun, D. E. Culler, “The ganglia distributed monitoring system: design, implementation, and experience,” Parallel Computing, vol. 30(7), 2004, pp. 817-840, ISSN 0167-8191, https://doi.org/10.1016/j.parco.2004.04.001.

[245] Linh T.X. Phan, Z. Zhang, B.T. Loo, Insup Lee, "Real-time MapReduce Scheduling", 2010, University of Pennsylvania Department of Computer and Information Science, Technical Report No. MSCIS-10-32.

[246] J. Veiga, R. R. Expósito, X. C. Pardo, G. L. Taboada and J. Tourifio, "Performance evaluation of big data frameworks for large-scale data analytics," 2016 IEEE International Conference on Big Data (Big Data), 2016, pp. 424-431, doi: 10.1109/BigData.2016.7840633.

[247]https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.0.0.2/bk_cluster-planning-guide/content/typical-workloads.html

[248] X. and Y. Liu, “A new evolutionary system for evolving artificial neural networks,” in IEEE Transactions on Neural Networks, vol. 8, no. 3, pp. 694-713, May 1997, doi: 10.1109/72.572107.

[249] https://www.iea.org/topics/renewables/solar/

[250] D. Lew, N. Miller, K. Clark, G. Jordan and Z. Gao, “Impact of High Solar Penetration in the Western Interconnection,” NREL Technical Report (NREL/TP-5500-49667), December 2010.

[251] M. P. Almeida, O. Perpiñáń, L. Narvarte, “PV power forecast using a nonparametric PV model,” Solar Energy 115 (2015), pp. 354-368.

[252] M. Q. Raza, M. Nadarajah, C. Ekanayake, “On recent advances in PV output power forecast,” Solar Energy 136 (2016) pp. 125–144.

[253] X. Qing, Y. Niu, “Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM,” Energy 148 (2018) pp. 461-468.

[254] S. Leva, A. Dolara, F. Grimaccia, M. Mussetta and E. Ogliari, “Analysis and validation of 24 hours ahead neural network forecasting of photovoltaic output power,” Mathematics and Computers in Simulation, 2017, vol. 131, pp. 88-100. http://dx.doi.org/10.1016/j.matcom.2015.05.010.

[255] M. Abuella, B. Chowdhury, “Solar Power Forecasting Using Artificial Neural Networks,” North American Power Symposium (NAPS), 2015.

[256] C. Paoli, C. Voyant, M. Muselli, M-L Nivet, “Use of exogenous data to improve an Artificial Neural Networks dedicated to daily global radiation forecasting,” 9th International Conference on Environment and Electrical Engineering, 2010.

[257] G. Vanderstar, P. Musilek, A. Nassif, “Solar Forecasting using Remote Solar Monitoring Stations and Artificial Neural Networks,” IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), 2018.

[258] H. Pedro, C. Coimbra, “Assessment of forecasting techniques for solar power production with no exogenous inputs,” Solar Energy 86 (2012) pp. 2017–2028.

[259] A. F. Romero, F. L. Quilumba, H. N. Arcos, “Short-Term Active Power Forecasting of a Photovoltaic Power Plant using an Artificial Neural,” IEEE Second Ecuador Technical Chapters Meeting (ETCM), 2017.

[260] T.E Hoff, R. Perez, J. Kleissl, D. Renne, J. Stein, “Reporting of irradiance modeling relative prediction errors,” Prog. Photovolt.: Res. Appl. 2013, vol. 21, pp. 1514–1519. http://dx.doi.org/10.1002/pip.2225.

[261] K. Hussain, M. N. B. Mohd Salleh, S. Cheng, R. Naseem, “Common Benchmark Functions for Metaheuristic Evaluation: A Review,” International Journal on Informatics Visualization, vol. 1 (2017), No. 4-2. DOI: 10.30630/joiv.1.4-2.65.

[262] M. Hadzima-Nyarko, E. K. Nyarko, D. Morić, "A neural network based modelling and sensitivity analysis of damage ratio coefficient," Expert Systems with Applications, vol. 38(10), 2011, pp. 13405-13413, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2011.04.169.

[263] Y. Sewsynker,E. B. Gueguim Kana, A. Lateef, "Modelling of biohydrogen generation in microbial electrolysis cells (MECs) using a committee of artificial neural networks (ANNs)," Biotechnology & Biotechnological Equipment, 2015, vol. 29(6), pp. 1208-1215. https://doi.org/10.1080/13102818.2015.1062732.

[264] M. Causse, J. Cameron, M. S. Masmoudi and T. Houcine, “Parsimonious Neural Networks,” 2019.

[265] J. D. Olden, M. K Joy, R. G Death, “An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data,”

Ecological Modelling, vol. 178(3–4), 2004, pp. 389-397, ISSN 0304-3800. https://doi.org/10.1016/j.ecolmodel.2004.03.013.

[266] Y. Dhebar, K. Deb, "Effect of a Push Operator in Genetic Algorithms for Multimodal Optimization," In: Mandal J., Dutta P., Mukhopadhyay S. (eds) Computational Intelligence, Communications, and Business Analytics. CICBA 2017. Communications in Computer and Information Science, vol 775. Springer, Singapore. https://doi.org/10.1007/978-981-10-6427-2_1.

[267] J. Rönkkönen, X. Li, V. Kyrki, et al., "A framework for generating tunable test functions for multimodal optimization," Soft Comput 15, pp. 1689–1706, (2011). https://doi.org/10.1007/s00500-010-0611-1.

[268] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in Proc. Workshop Mach. Learn. High Perform. Comput. Environ, 2015, p. 4.

[269] P. Linardatos, V. Papastefanopoulos, S. Kotsiantis, "Explainable AI: A Review of Machine Learning Interpretability Methods," *Entropy* 2021, vol. *23(1)*, 18. https://doi.org/10.3390/e23010018

[270]https://www.greenpeace.org/international/press-release/24112/electricity-consumption-from-chinas-internet-industry-to-increase-by-two-thirds-by-2023-greenpeace/

[271] Q. Tang, S. Gupta, and G. Varsamopoulos, "Thermal-aware task scheduling for data centers through minimizing heat recirculation," In Cluster Computing, 2007 IEEE International Conference on, pages 129 –138, sept. 2007.

[272] X. Li, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," in Proc. Genet. Evol. Comput. Conf., Seattle, WA, USA, 2004, pp. 105–116.

[273] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Multimodal optimization using niching differential evolution with index-based neighborhoods," in Proc. IEEE Congr. Evol. Comput. (CEC), Brisbane City, QLD, Australia, Jun. 2012, pp. 1–8.

[274] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 28, no. 3, pp. 417–425, Jun. 1998.

[275] J. L. Noyes, B3.5 Handbook of Neural Computation. London, U.K.: Chapman & Hall, 1997.

[276] M. Srinivas, L. M. Patnaik, "Genetic algorithms: A Survey," Computer, June 1994. https://doi.org/10.1109/2.294849.

[277] K. De Jong, W. Spears, "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," 2006. 10.1007/BFb0029729.

[278] J. J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 16, no. 1, pp. 122-128, Jan. 1986, doi: 10.1109/TSMC.1986.289288.

[279] Y. W. Foo, C. Goh and Y. Li, "Speciation and diversity balance for Genetic Algorithms and application to structural neural network learning," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 2016, pp. 1283-1290.

[280] Z. Zhang, et al., "Opening the black box of neural networks: methods for interpreting neural network models in clinical applications," Annals of translational medicine vol. 6,11 (2018): 216. doi:10.21037/atm.2018.05.32

[281] M. Scardi, L.W. Harding, "Developing an empirical model of phytoplankton primary production: a neural network case study," Ecological Modelling, vol. 120(2–3), 1999, pp. 213-223. https://doi.org/10.1016/S0304-3800(99)00103-9

[282] M. Tsang, D. Cheng and Y. Liu, "Detecting Statistical Interactions from Neural Network Weights," 2017. https://arxiv.org/abs/1705.04977

Appendices

A. The EVLNN Pseudo Code

Figure A.1 Pseudo-code for EVLNN

Algorithm 1 The EVLNN Pseudo Code

```
                                                    // Initialization
1.  main(MAX_Iteration, popSize)
2.  set control parameters
3.  generate a set of feasible solutions uniform random distribution, called popVector of size popSize
4.  popVector=rand(popSize)
    // Map phenotype (or real values) to chromosomes vectors
5.  chromVector=map(popVector)
                                                    // Speciation
6.  for i=1 to size of chromVector do
    // Assign chromosomes to species vector
7.      species_id=length(chromVector(i))
8.      spVector(species_id)=chromVector(i)
9.  end for
    // Loop until condition is met
10. while globalBestVector not found AND iteration < MAX_Iteration do
                                                    // Evaluate and rank species
11.     for each spVector do
    // Evaluate fitness of feasible solutions and rank solutions within species
12.       spVectorRanked=rank(eval(spVector))
13.    end for
                                                    // Mating selection and recombination
14.    for each species do
    // Intra-species recombination
15.       for i = 1 to size of spVectorRanked do
16. // Select pair uniformly and stochastically
17.           ParentA=SUS(spVectorRanked)
18.           ParentB=SUS(spVectorRanked)
19. // Inter-species recombination
20. // Check if inter-species crossover probability is met
21. // if so randomly select 2nd parent from the general population
22.           if crossspecies_crossover_probability > rand()
23.              ParentB=rand(1, popVector)
24.           end if
    // Perform single-point crossover
25.     ChildA=Parent1A+Parent2B
26.     ChildB=Parent2A+Parent2B
    // Compare fitness of parent-child
27.       if fit(ChildA,ChildB) > fit(ParentA, Parent B)
28. // Store new feasible solutions
29.           newVector(ChildA, ChildB)
30.       else
31.           newVector(ParentA, ParentB)
32.       end if
    // Check feasible solutions
33.       if newVector is unfeasible
34.          Repair and Update newVector
```

```
35.          end if
36.       end for
                                                    // Mutation
    // Select remaining solutions as mutants
37.       for i = 1 to size of Remain(spVectorRanked) do
38.          generate changeVector
39.          generate probabilityVector
    // Prepare mutation vector for link and weights mutation
40.          mutationVector=changeVector*probabilityVector
41.          mutant=Remain(spVectorRanked)
42.          mutant=mutant*mutationVector
    // Store new feasible solutions
43.         newVector(mutant)
    // Check feasible solutions
44.             if newVector is unfeasible
45.                Repair and Update newVector
46.             end if
47.       end for
    // Update chromosome vector and species vector and evaluate global best
48.    for each newVector do
                          // Evaluate fitness of feasible solutions and rank solutions globally
49.         newVectorRanked=rank(eval(newVector))
50.    end for
    // Replace bottom individuals with healthier individuals
51.    bottomIndex=bottomIndex(newVectorRanked)
52.    topIdex=topIndex(newVectorRanked)
53.    newVectorRanked(bottomIndex)=newVectorRanked(topIndex)
54. // Update chromosome vector
55.    chromVector=newVectorRanked
56.    spVector=chromVector
57. // Obtain genome map of each individual
58.    genome[]=getGenome(spVector)
                                                    // Evaluate global best
59.    globalBestVector=maxfit(chromVector)
                                                    // Diversity Calculation
60. // Perform diversity calculation and plot the trends
61. diversity_calc(genome[], popSize)
62. plot_diversity()
63. end while
                                                    // Ensemble-based Sensitivity Analysis
64. SA=sensitivity_analysis_calc()
65. return(globalBestVector, SA)
66. end main
```

Figure A.2 Pseudo-code for diversity calculation

Algorithm 2 Diversity Calcuation Method

```
1:  function diversity_calc (genome, popSize)
    // Find non-zero elements in the genome_array and return the indices corresponding to the non-zero
    // entries. This is to convert the genome to number string for diversity calculation
2:  num_genome=0
3:  populationSize=popSize
4:  for i=1 to size of individual genome do
5:      num_genome=num_genome+(genome(i))
6:  genome[]=find_non_zero(genome)
7:  end for
8:  unique_genome=genome[]
9:  for i=1 to population_size do
10:     unique_genome=union(unique_genome, genome[i])
11:     total_genome=total_genome+size(genome[i])
12:     i=i+1;
13: end for
14: // Calculate population diversity
15: pop_diversity=population_size*(size(unique_genome)/total_genome);
16: // Calcuate Shannon's diversity
17: species=unique(sort(genome[]))
18: individuals=genome[];
19: shannon=[species,histcount(individuals(),species)]
20: species_diversity_index=0;
21: for i=1 to size of shannon array do
22:     temp=-[(shannon[i]/sum(shannon[])*log(shannon[i]/sum(shannon[])];
23:     species_diversity_index=species_diversity_index+temp;
24: end for
25: species_equitability_index=species_diversity_index/log(size(species[]))
26: return(pop_diversity,species_diversity_index,species_equitability_index)
27: end function
```

B. Sensitivity Analysis Methods

B.1 Methodology for the PaD Method

In implementing the Partial Derivatives (PaD) method, the relative contribution of the Multi-Layer Perceptron (MLP) outputs is computed using the Sum of the Square Partial Derivatives (SSD) obtained per input variable [192] [197]. This method is based on the Backpropagation (BP) algorithm, which is used to compute the partial derivatives of the cost function with respect to each weight parameter [280]. The expression for the SSD implemented in the EVLNN algorithm is,

$$s_{ik,SSD} = \sum_{j=1}^{N} (s_{ik}|_{x_j})^2 \tag{B.1}$$

where $s_{ik}|_{x_j}$ refers to the sensitivity of the output of the k^{th} neuron in the output layer to the input of the i^{th} neuron evaluated in x_j and N refers to the number of samples in the dataset. Figure B.1 shows the Pseudo-code for the PaD method.

Figure B.1 Pseudo-code for the PaD method.

Algorithm 1 Partial Derivative Method

```
function PaD_METHOD (W, B, N, H, I, O, X)
// INPUT: Parameter values of the fittest individual EVLNN including weights W, bias B, sample size
// N, number of hidden neurons H, number of inputs I, number of outputs O and the testing dataset, X
// OUTPUT: Ranking of the relative contribution of the input variables to the output variable
for n = 1 to N do
    for j = 1 to H do
        for i = 1 to I do
            //  Calculate the weighted sum (z_ij) at the input i to the hidden neurons, j
            z_ij = Σ_{i,j}(w_ij · x_i) + b_j
        end for
        // Calculate the output of the activation function (h_j) at the hidden neuron, j
        h_j = 1 / (1+exp(-z_ij))     // for Sigmoid activation function
    end for
    for k = 1 to O do
        // Calculate the weighted sum z_jk at the input of k^th output neuron
        z_jk = Σ_{j,k}(w_jk · h_j) + b_k
    end for
        //  Calculate the output of EVLNN (y_k) at the k^th output neuron
        y_k = z_jk      // for Pure line, f(x)=x activation function
    // Calculate the derivation, f'(h_j) at the hidden neuron, j
    for j = 1 to H do
        f'(h_j) = h_j · (1 - h_j)
```

end for
// Calculate the derivation, $f'(y_k)$ at the output neuron, k
for l = 1 to O **do**
$f'(y_k)$= 1
end for
// Calculate the variation of the output variable y_k, with respect to the input variables x_i
// using the chain rule
for i = 1 to I **do**
$var\_y\_x_i = \sum_i \left(f'(y_k) \cdot w_{jk} \cdot f'(h_j) \cdot w_{ij} \right)$
end for
for i = 1 to I **do**
// Calculate the mean sensitivity, S_i
$S[i] = \frac{\sum_i (var_{y_{x_i}})}{N}$
// Calculate the sum of the squared partial derivatives (SSD) for each input variable
$SSD[i] = \sum_i [(S_i).^2]$
end for
for i = 1 to I **do**
// Calculate the relative importance of each input variables in percentage terms
$RI[i]=(SSD[i]/\sum SSD) \times 100$
end for
// Sort the relative importance in ascending order
ranked_RI=sort(RI)
return(*ranked_RI***)**
end function

The algorithm first imports the optimized parameters of the trained EVLNN. Then the values of h_j and z_{ij} are calculated. h_j is the output value of the j^{th} hidden neuron with a Sigmoid activation function expressed in Equation B.2, and z_{ij} is the weighted sums at the input of the j^{th} hidden neuron given by Equation B.3,

$$h_j = \frac{1}{1+e^{-z_{ij}}} \tag{B.2}$$

$$z_{ij} = \sum_{i,j} (w_{ij} \cdot x_i) + b_j \tag{B.3}$$

where w_{ij} is the connection weight between the j^{th} hidden neuron and the i^{th} input neuron and x_i is the value of the i^{th} input variables, and b_j is the bias at the j^{th} hidden neuron. Subsequently, y_k and z_{jk} are calculated. y_k is the output value of the k^{th} output neuron with a Pure Line activation function ($f(x)=x$) expressed in Equation B.4 and z_{jk} is the weighted sums at the input of the k^{th} output neuron given by Equation B.5,

$$y_k = z_{jk} \tag{B.4}$$

$$z_{jk} = \sum_{j,k} w_{jk} \cdot h_j + b_k \tag{B.5}$$

where w_{jk} is the connection weight between the k^{th} output neuron and the j^{th} hidden neuron and h_j is the output of the j^{th} hidden neuron, and b_k is the bias at the k^{th} hidden neuron. Next, the variation in the output value y_k with respect to the variation in the input variables x_i is calculated by applying the chain rule given by Equation B.6,

$$\frac{\partial y_k}{\partial x_i}(\boldsymbol{X}_n) = \sum_{i,j,k} \frac{\partial y_k}{\partial z_{jk}} \cdot \frac{\partial z_{jk}}{\partial h_j} \cdot \frac{\partial h_j}{\partial z_{ij}} \cdot \frac{\partial z_{ij}}{\partial x_i} \tag{B.6}$$

where

$$\frac{\partial y_k}{\partial z_{jk}} = f'(y_k) = 1 \tag{B.7}$$

$$\frac{\partial z_{jk}}{\partial h_j} = w_{jk} \tag{B.8}$$

$$\frac{\partial h_j}{\partial z_{ij}} = f'(h_j) = h_j \cdot (1 - h_j) \tag{B.9}$$

and

$$\frac{\partial z_{ij}}{\partial x_i} = w_{ij} \tag{B.10}$$

from which Equation B.11 and B.12 are obtained,

$$\frac{\partial y_k}{\partial x_i}(\boldsymbol{X}_n) = \sum_{i,j,k} f'(y_k) \cdot w_{jk} \cdot f'(h_j) \cdot w_{ij} \tag{B.11}$$

$$= \sum_{i,j,k} 1 \cdot w_{jk} \cdot h_j \cdot (1 - h_j) \cdot w_{ij} \tag{B.12}$$

Following, the mean sensitivity value S_i can be calculated using the output value y_k with respect to the variation for each input variable x_i expressed in Equation B.13,

$$S_i = \frac{1}{N} \sum_n \frac{\partial y_k}{\partial x_i} \tag{B.13}$$

With the sensitivity obtained for each variable, the relative contribution of each input variable can be acquired by calculating the sum square partial derivatives (SSD) shown in Equation B.14,

$$SSD_i = \sum_i^N (S_i)^2 \quad \text{(B.14)}$$

The input variable with the highest SSD value has the most influence on the output variable. This value is then divided by the sum of SSD from which the relative importance of each input variable in percentage terms is computed and subsequently ranked.

B.2 Methodology for the Profile Method

The profile method analyzes the relative importance of each input variable by varying the values of the input variable along with a scale range. In contrast, the remaining input variables are kept at fixed values [200]. The scale consists of a chosen number of intervals between its minimum and maximum initial data value. Each input variable x_i is increased in steps following the scale, keeping all other variables fixed with an initial setting to their minimum value, the first quartile, median, third quartile, and maximum values. Hence, for each variation x_i in the scale point, five output values will be obtained, one from each quantile. The median output is measured to obtain a curve that displays the variation profile for every variable. The difference between the maximum and minimum values from the profile curve is computed. The more significant the difference, the more influence that input variable has on the response variable. Figure B.2 shows the pseudo-code for the Profile method.

Figure B.2 Pseudo-code for the Profile method.

Algorithm 2 Profile Method

```
function PROFILE_METHOD (W, B, N, H, I, O, X)
//
// INPUT: Parameter values of the fittest individual EVLNN including weights W, bias B, sample size
// N, number of hidden neurons H, number of inputs I, number of outputs O and the testing dataset, X
// OUTPUT: Ranking of the relative contribution of the input variables to the output variable
//
// Sort input values in ascending order rowwise
sorted_x_i=sort(x_i)
for i = 1 to 9 do
    // Create a scale with 11 steps for the input variable x_i, starting from minimum to maximum
    // in steps of 10% to 90% , where scale[0]=x_min , scale[10]=x_max , and scale[1,...,9] is given below
    scale[i]= x_i × (0.1 × i)
end for
for j = 1 to 3 do
    // Create 5 quantiles starting from minimum sample size of 1, to 1st quantile, median, 3rd quantile
    // and maximum sample size, where quantiles[0]=1, quantiles[4]=size(X),
    // and quantiles[1,..,3] is given below
    quantiles[j]=[size(X) × (0.25 × j)]
end for
for i = 1 to I do
    for q = 1 to 5 do
        sorted_x_i=sorted_x_i(:,quantiles[j])
        for s = 1 to 11 do
            sorted_x_i=sort_x_i(:,scale[i])
            // Calculate the new predicted output, y_p of the EVLNN model with the new input, x_i
```

```
            y_p[s,q]=EVLNN(W, B, N, H, I, O, sorted_x_i)
        end for
    end for
    // Sort rowwise so that the median value can be extracted
    for s = 1 to 11 do
        sorted_y_p[s,:]=sort(y_p[s,:])
    end for
    // The median value is in the third column
    MEDIAN[i]=sorted_y_p[:,3]
end for
for i = 1 to I do
    // Calculate the difference between the max and min output values and rank the input variables
    // that have the most influence on the output variable
    RI[i]=max(MEDIAN[i])-min(MEDIAN[i])
end for
// Sort the relative importance in ascending order
ranked_RI=sort(RI)
return(ranked_RI)
end function
```

The first step of the Profile method is to sort the input values in ascending order. Next, a scale of 11 steps, *scale[1,..,11]* is created to form new input values consisting of the minimum, 10%, 20%, 30% 40%, 50%, 60%, 70%, 80%, 90% and maximum value of the input variable. At each input value x_i from its minimum to its maximum, the values of the rest of the input variables are held fixed, first at their minimum value, followed by first quartile, median, third quantile, and finally maximum values. With the new input values, the predicted output at the EVLNN model is subsequently computed, resulting in five output values for each variation of $x_{i,}$ which are then ranked. The median values are identified over the scale range to obtain a profile curve with variation for every variable. The difference between the maximum and minimum median values is determined to identify the input variables relative to their importance. The higher the difference, the more influence the input has on the output. The Profile method is illustrated with a schema shown in Figure B.3.

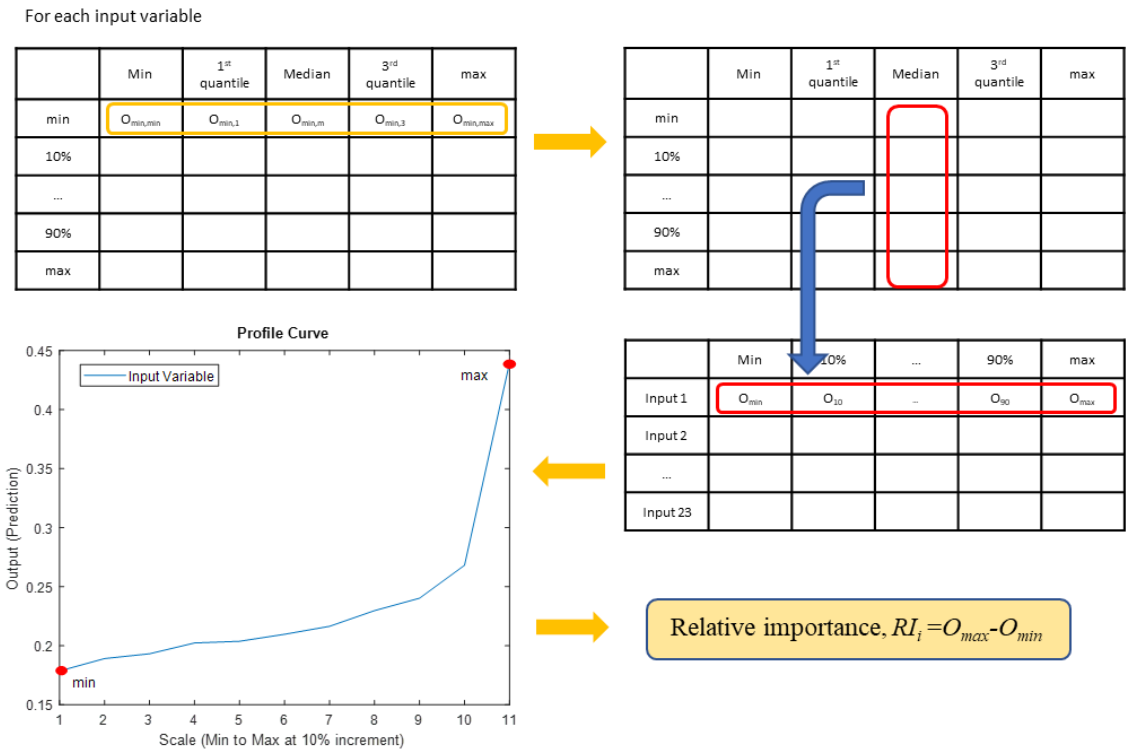


Figure B.3 Profile method schema.

B.3 Methodology for the Perturb Method

In implementing the perturb method, the value of the selected variable is perturbed while the other variables are kept at their original values [198]. The formula is expressed as,

$$x_i = x_i + \delta_j \tag{B.15}$$

where δ_j or white noise is progressively applied to each variable of the identified model at incremental steps (j=1, 2, ...) of the original value, x_i, and the output is then measured. Injecting white noise at the input variables causes the response variable at the output to increase [281]. This increase in the neural network output is subsequently assessed to determine which predictor is relatively more important than the rest. The predictor who affects the change the most is the most influential. Though the relationship between the predictors and the response variables is not strictly monolithic, less sensitive predictors should not significantly affect the neural network outputs with considerable white noise. In contrast, predictors with high relative importance should cause material variation to the neural network outputs when white noise is injected. Figure B.4 shows the pseudo-code for the Perturb method.

Figure B.4 Pseudo-code for the Perturb method.

Algorithm 3 Perturb Method

```
function PERTURB_METHOD (W, B, N, H, I, O, X, mse_ori)
//
// INPUT: Parameter values of the fittest individual EVLNN including weights W, bias B, sample size
// N, number of hidden neurons H, number of inputs I, number of outputs O, the testing dataset, X and
// the original results of the Mean Square Error (MSE)
// OUTPUT: Ranking of the relative contribution of the input variables to the output variable
//
// Sort input values in ascending order rowwise
sorted_xi=sort(xi)
for i = 1 to I do
    for j = 1 to 5 do
        //  Perturb input variable xi in 5 steps at 10% to 50% of the original value
        p[i]= [xi + (xi × (0.1 × j))
        // Copy the original values of input data set
        perturbed_X[:]=X[:]
        // Replace the value of input variable xi with new perturbed values
        perturbed_X[i]=p[i]
        // Calculate the new predicted output, yp of the EVLNN model with the new input, xi
        yp=EVLNN(W, B, N, H, I, O, perturbed_X)
```

74: // Calculate the MSE using the function mse_cal
75: *perturbed_mse[i,j] = mse_cal(X_{target}, y_p)*
17: **end for**
18: // Calculate the mean MSE with the perturbed input values
19: a*ve_mse=[sum(perturbed_mse)]/5*
20: // Calculate the relative importance of each input variable by computing the absolute difference
21: // of the new MSE with the original MSE
22: *RI[i]=|mse_mean−mse_ori|*
23: **end for**
24: // Sort the relative importance in ascending order
25: *ranked_RI=sort(RI)*
26: **return(***ranked_RI*)
27: **end function**

The steps in the Perturb method include applying a perturbation, δ to each input, $x_{i,}$ as expressed in Equation B.14.

$$x_i = x_i + \delta \tag{B.16}$$

Five steps of perturb amount equivalent to 10%, 20%, 30%, 40%, and 50% of the original value are applied progressively to compute the predicted output at the EVLNN model while keeping the other inputs to their original value. The result of the predicted output is subsequently used to calculate the MSE. After the five steps of perturbation, the mean MSE *mse_mean* is computed. This *mse_mean* is then compared with the original MSE *mse_ori* (obtained before the perturbation). The difference obtained represents the relative contribution of the input variable. The input variable that has a more significant influence on the output variable than the rest would exhibit a more significant difference between the MSEs. The Perturb method is illustrated with a schema shown in Figure B.5.

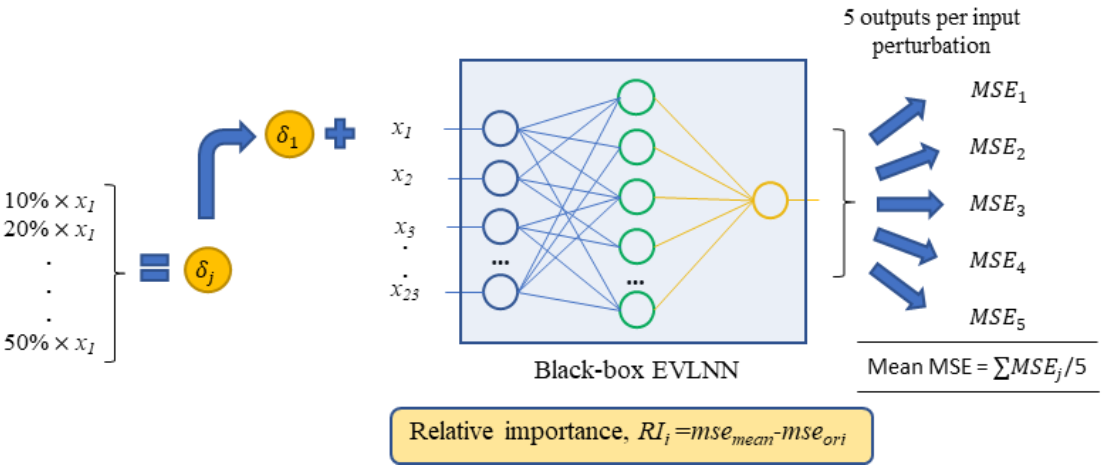


Figure B.5 Perturb method schema.

B.4 Methodology for the Connection Weights Method

The importance of neural network connection weights has been investigated in the works of [188], [192], [201], [282]. The connection weights in a neural network are the links between the neurons from the inputs to the outputs. Hence, its outputs depend on the contributions of the inputs subjected to the connection weights' magnitude and direction. Inputs with higher connection weights represent a higher excitation level of activation at the output of the neurons. Therefore, they are relatively more important in predicting than inputs with lower connection weights. Inputs with positive connection weights increase the value of the predictive response, and inputs with negative connection weights decrease the value of the prediction response. In this study, the Weights method was applied to calculate the relative importance of the inputs to the neural network output, using Equation B.17 and Equation B.18 [195] [201],

$$Q_{ih} = \frac{|w_{ih}|}{\sum_{i=1}^{ni}|w_{ih}|} \tag{B.17}$$

where w_{ih} refers to the absolute value of the connection weight between the input neuron, *i,* and the hidden neuron, *h*, and Q_{ih} is the ratio of the absolute value of the connection weight and the sum of the absolute value of the connection weights of all input neurons *i*, and,

$$RI(\%)_i = \frac{\sum_{h=1}^{nh} Q_{ih}}{\sum_{h=1}^{nh}\sum_{i=1}^{ni} Q_{ih}} \text{ x } 100 \tag{B.18}$$

where *RI* is the percentage relative importance of all output weights attributable to the input variables, is the ratio of the sum of Q_{ih} for each hidden neuron and the sum for each hidden neuron of the sum for each input neuron of Q_{ih}. Figure B.6 shows the pseudo-code for the Connection Weights method.

Figure B.6 Pseudo-code for the Connection Weights method.

Algorithm 4 Connection Weights Method

```
function CONN_WEIGHTS_METHOD (W, B, N, H, I, O, X)
// INPUT: Parameter values of the fittest individual EVLNN including weights W, bias B, sample size
// N, number of hidden neurons H, number of inputs I, number of outputs O, the testing dataset, X
// OUTPUT: Ranking of the relative contribution of the input variables to the output variable
// Take the absolute values of the neural network weights
W_positive=|W|
// For each hidden neuron and each input variable x_i, multiply the absolute value of the hidden-
// output layer W_positive_jk connection weight by the absolute value of the hidden-input layer connection
// weight W_positive_ij.
for j = 1 to H do
    for i = 1 to I do
        P[i,j] = W_positive[i,j] × W_positive[j,k]
    end for
end for
for j = 1 to H do
    for i = 1 to I do
    // Divide P by the sum for all the input variables Q_ij
    Q[i,j]=P[i,j]/∑P[i,j]
    end for
end for
// Compute the sum of the influence for each input
S[i] = ∑Q[i,j]
S2 = ∑S[i]
for i = 1 to I do
// Divide S_i by the sum for all the input variables, S2, and express as a percentage to compute relative
// importance which is the distribution of all output weights attributable to the given input variable.
RI[i]=(S[i]/S2) × 100
end for
// Sort the relative importance in ascending order
ranked_RI=sort(RI)
return(ranked_RI)
end function
```

The first step in the Connection Weights method is to take the absolute values of the connection weights of the identified EVLNN model. Then, for each hidden neuron, compute the product of the connection weight from the input to the hidden neuron, with the connection weight from the hidden to the output neuron, to obtain the array P_{ij},

$$P_{ij} = w_{ij} \cdot w_{jk} \tag{B.19}$$

where j is the number of hidden neurons, w_{ij} is the connection weight between the input neuron i and the hidden neuron j, and w_{jk} is the connection weight between the hidden

neuron j and the output neuron k. Subsequently, for each hidden neuron, divide P_{ij} by the sum of all input variables P_{ij}, to obtain Q_{ij},

$$Q_{ij} = \frac{P_{ij}}{\sum P_{ij}} \tag{B.20}$$

Following, for each input neuron, the product Q_{ij} is summed to obtain S_i,

$$S_i = \sum_i Q_{ij} \tag{B.21}$$

Finally, S_i is divided by the sum of all the input variables and then multiplied by 100 to compute the percentage relative importance, *RI(%),*

$$RI_i(\%) = \left(\frac{S_i}{\sum S_i}\right) \times 100 \tag{B.22}$$

A higher *RI* value represents the input variable's more significant influence over the output variable. The Connection Weights method is illustrated with a schema shown in Figure B.7.

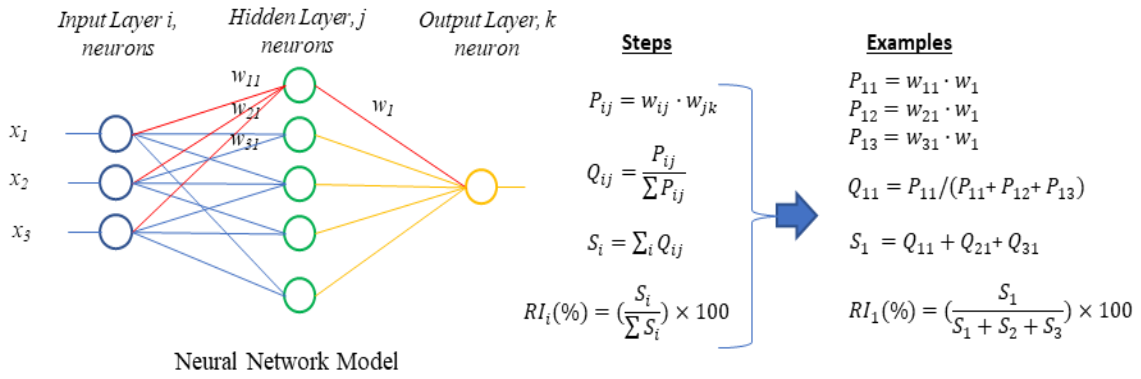


Figure B.7 Connection Weights schema.

C. Analysis of EVLNN's Search Pattern

C.1 The Himmelblau-2D Function

The Himmelblau benchmark function is used to illustrate EVLNN's search operation. Detailed data during the experiment was collected. With the help of visualization plots, the algorithm's search characteristics were investigated and analyzed to explain EVLNN's search process. The Himmelblau-2D (f_{11}) function has the mathematical equation expressed as,

$$f_{11}(x) = ({x_1}^2 + x_2 - 11)^2 + (x_1 + {x_2}^2 - 7)^2 \tag{C.1}$$

The 3D plot of the function is shown in Figure C.1, where the figure depicts four global optima. The global minima $f(x^*) = 0$, are at x^*= (3, 2), (-3.779, -3.283), (-2.805, 3.131) and (3.584, -1.848). Figure C.2 shows the contour landscape in 2D. The search is evaluated on $x_i \in$ [-6 6], for all i =1, …, d where d=2.

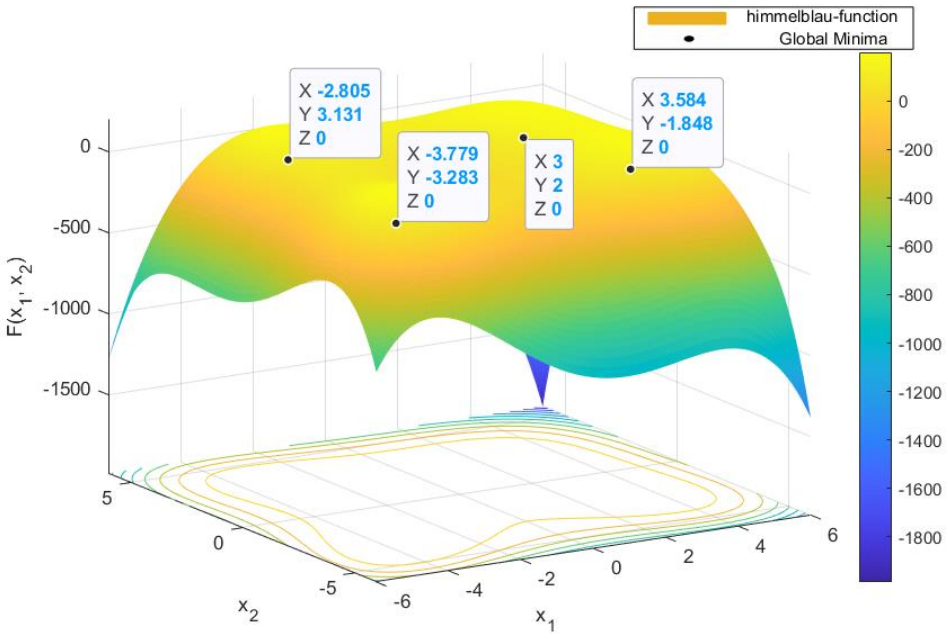


Figure C.1 3D plot of the Himmelblau function with four global minima

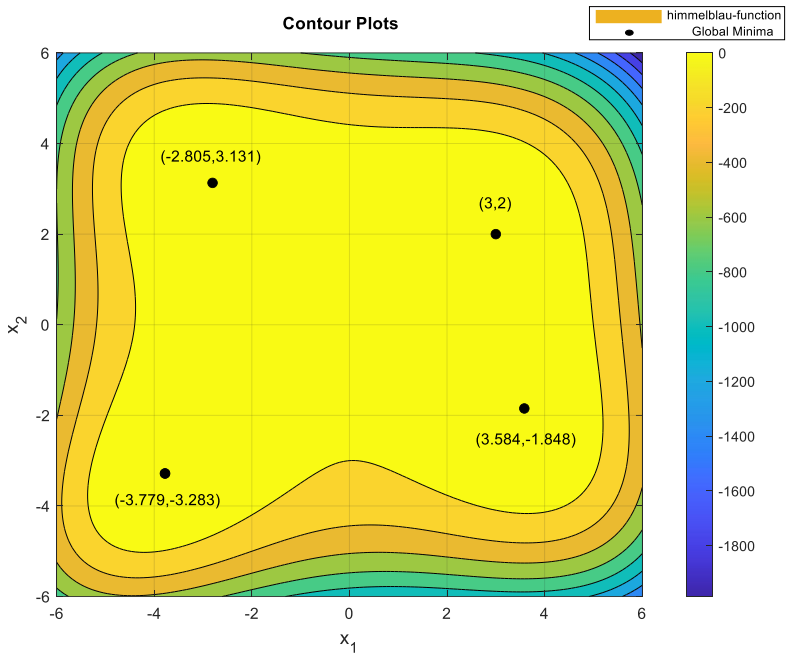


Figure C.2 Contour plot of the Himmelblau function with locations of the four global minima

Figure C.3 shows the population at initiation presented on a 2D search landscape of the Himmelblau function in the range [-6 6]. Potential solutions are represented by various colored shapes on the landscape with similar colored shapes belonging to the same species. The species are initially scattered over the search landscape, searching for attractive basins. The function has four global minima at $f(x^*)=0$, at $x^* = (3, 2)$, $x^* = (-3.779, -3.283)$, $x^* = (-2.805, 3.131)$ and $x^* = (3.584, -1.848)$. Four large red squares mark these locations in the search landscape. Figure C.4(a) shows the species distribution at initialization. There are in total 125 individuals and 14 species. As observed, individuals are not distributed evenly among the different species. SP_3 or *Species_3* has 13 individuals, the largest species, whereas *Species_6* and *Species_15* have five individuals each. They form the smallest species. Figure C.4(b) shows the population convergence rate over 500 generations. The average MSE falls steeply in the first 20 generations and needs about 400 generations to converge to good solutions. Figure C.5 shows the convergence rate of each of the 14 species, converging at a different rate to the global minima. *Species_15* has the slowest convergence rate suggesting that this species was stuck at a local minimum in the search landscape. Nonetheless, the species eventually converged along with the other species.

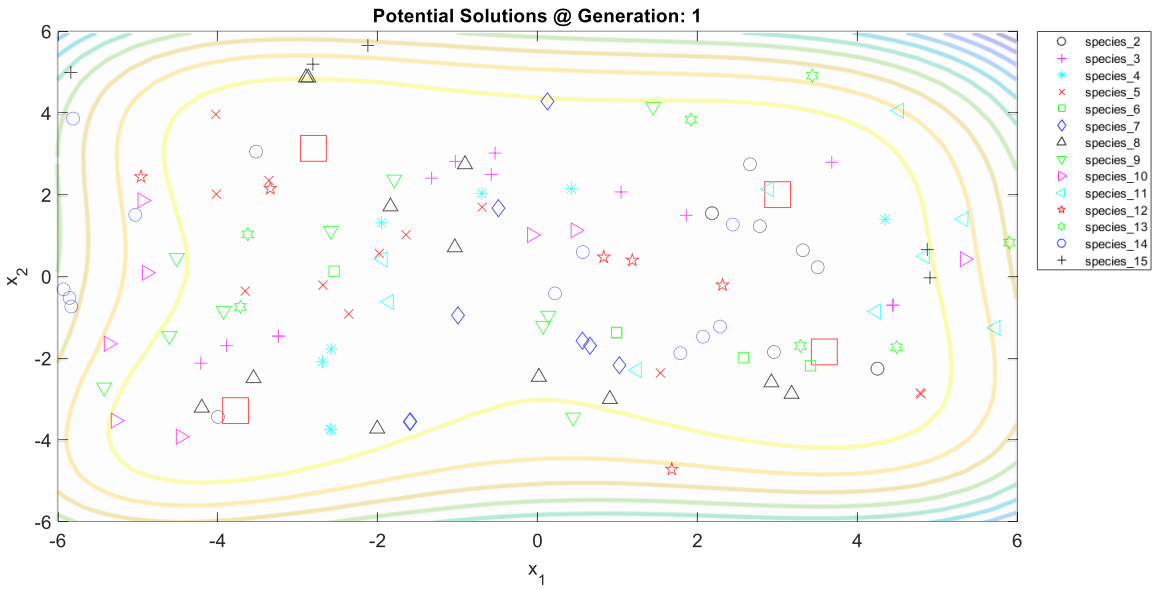
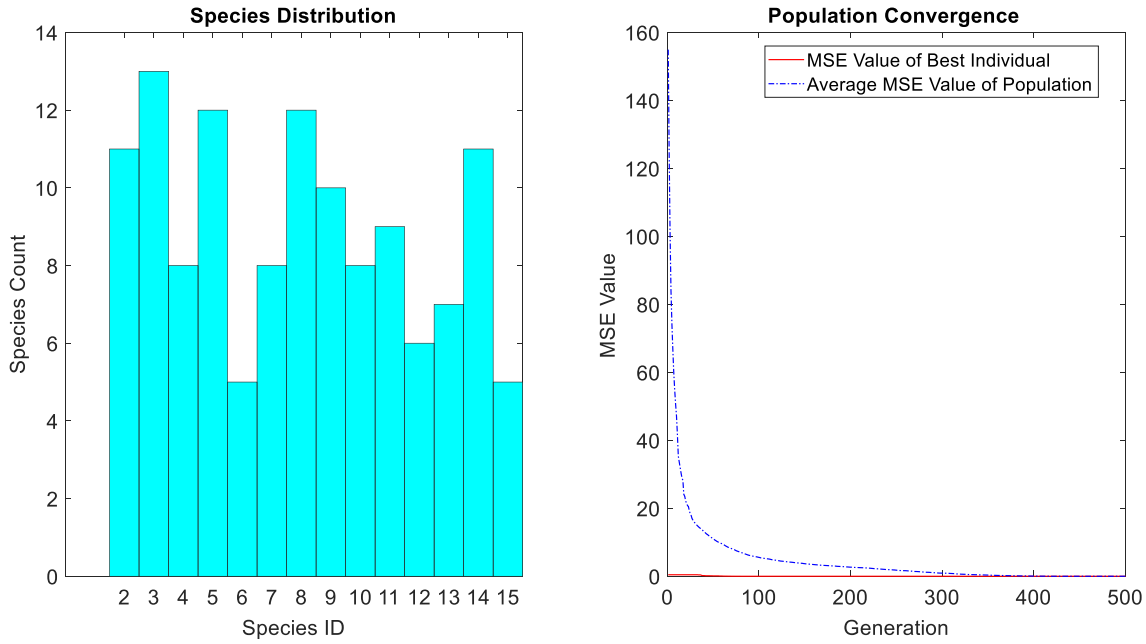


Figure C.3 Landscape showing speciated solution candidates in generation 1.



(a) (b)

Figure C.4(a-b) (a) Species distribution at the initialization. (b) Population convergence over 500 generations.

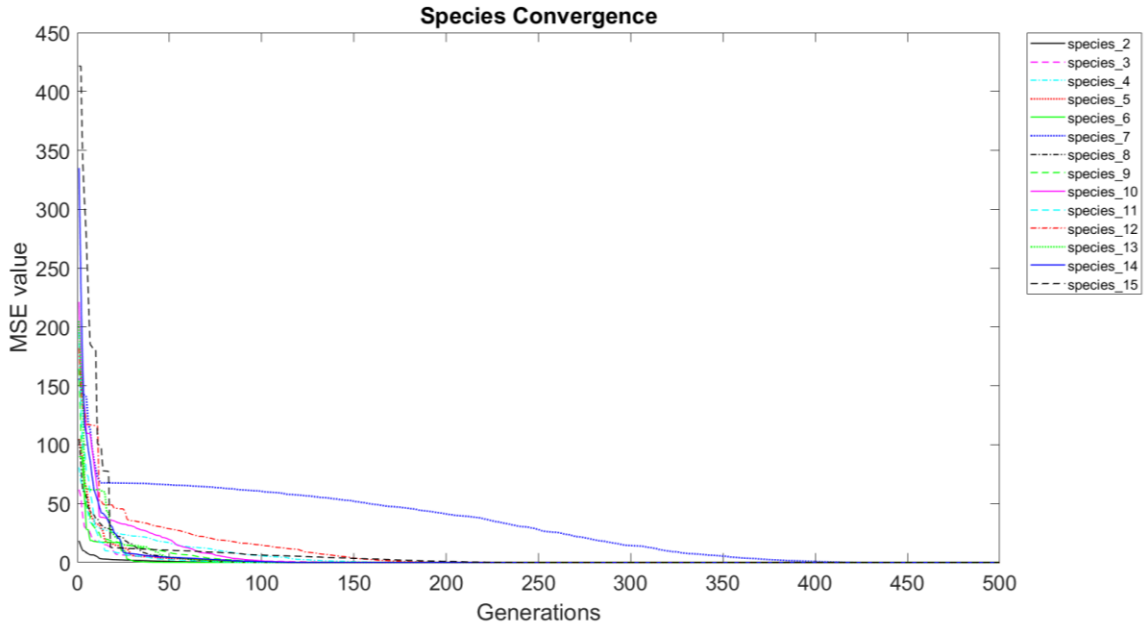


Figure C.5 Individual species convergence over 500 generations.

Figure C.6 to Figure C.20 shows the evolutionary map of the EVLNN algorithm captured in steps of ten generations from generations 10, 20, 30, … to 100, 150, 200, then in steps of 100 generations to 500. These plots visually depict how the population evolves on the objective function surface where the various species are highlighted using different colored shapes. The four global optima are highlighted using red squares.

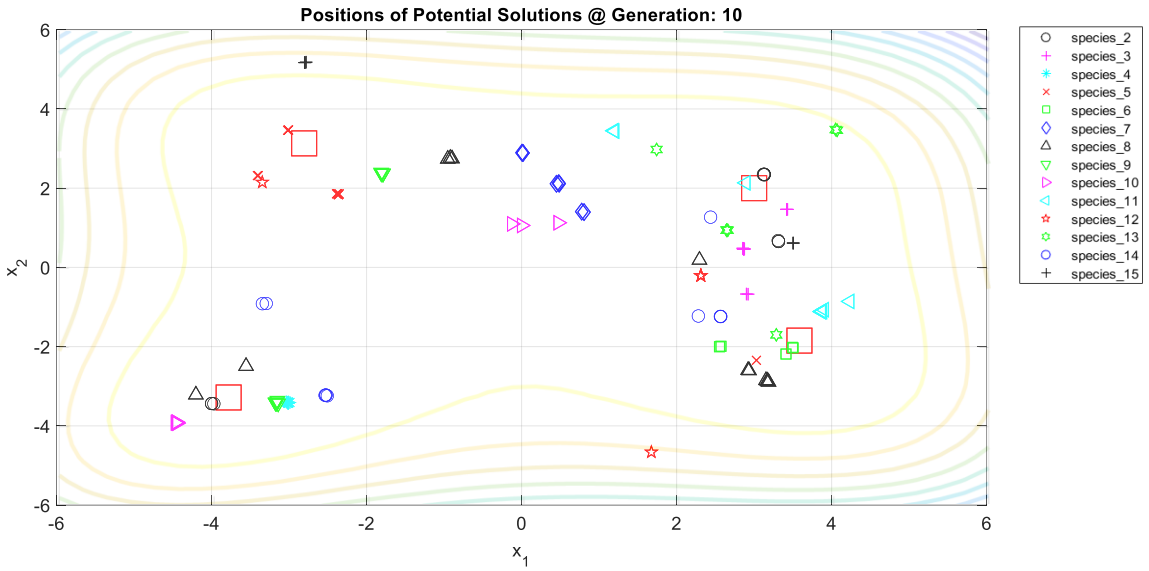


Figure C.6 At generation 10, species move towards the basins of interest depicted by the four red squares.

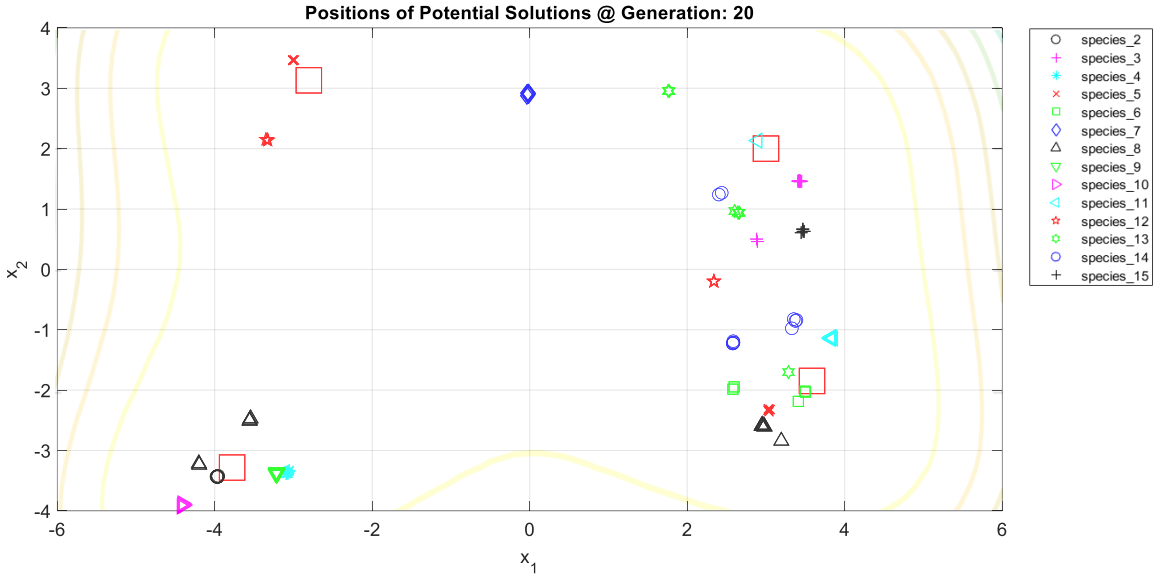


Figure C.7 At generation 20, species are seen drawing closer to the minima

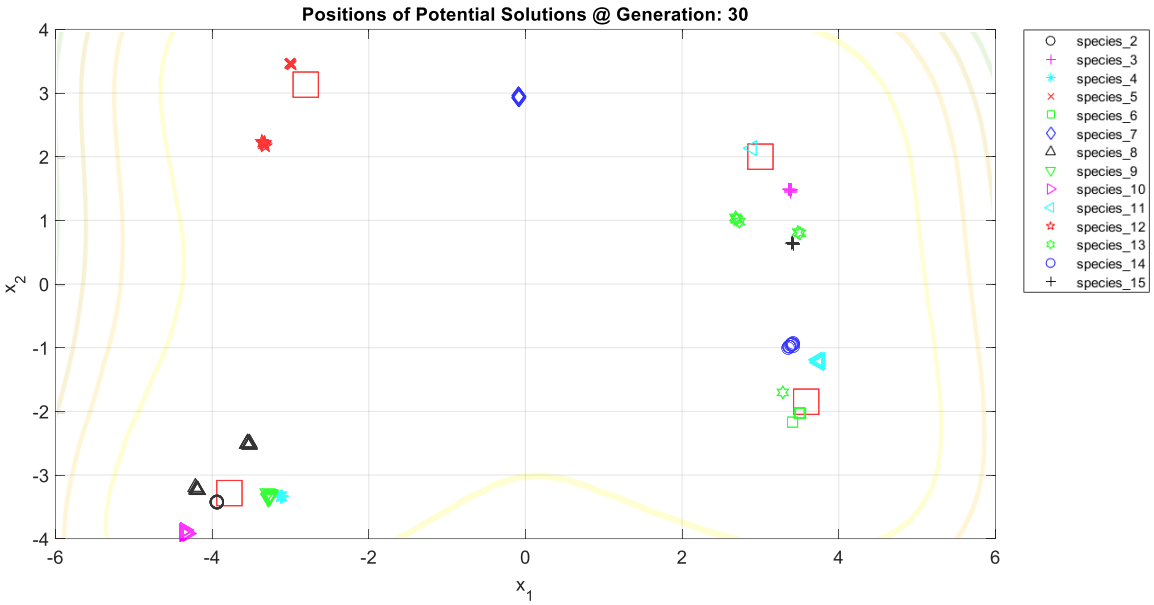


Figure C.8 At generation 30, species are becoming similar based on their positions.

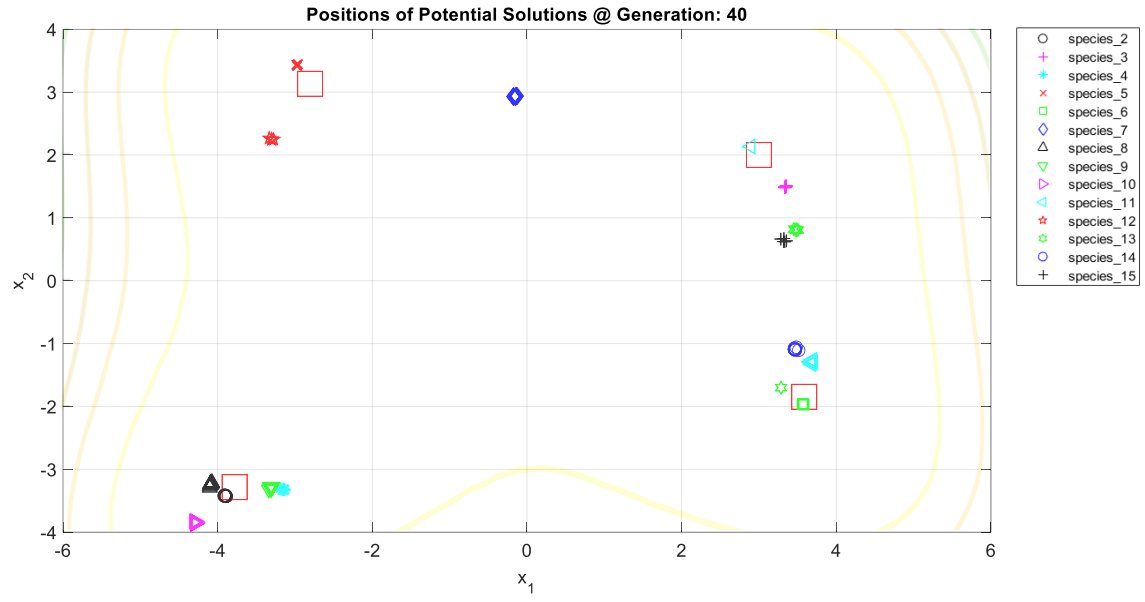


Figure C.9 At generation 40, Species_8 has become identical in their search positions like the other respective species.

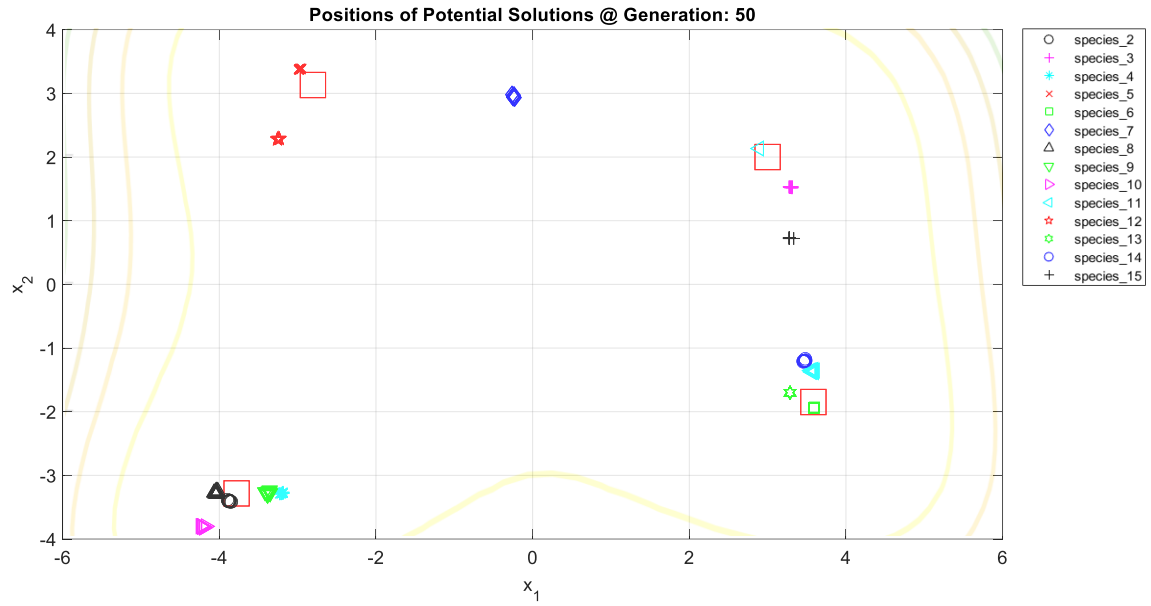


Figure C.10 At generation 50, most species are near global minima except Species_7, which seems stuck in a local minima.

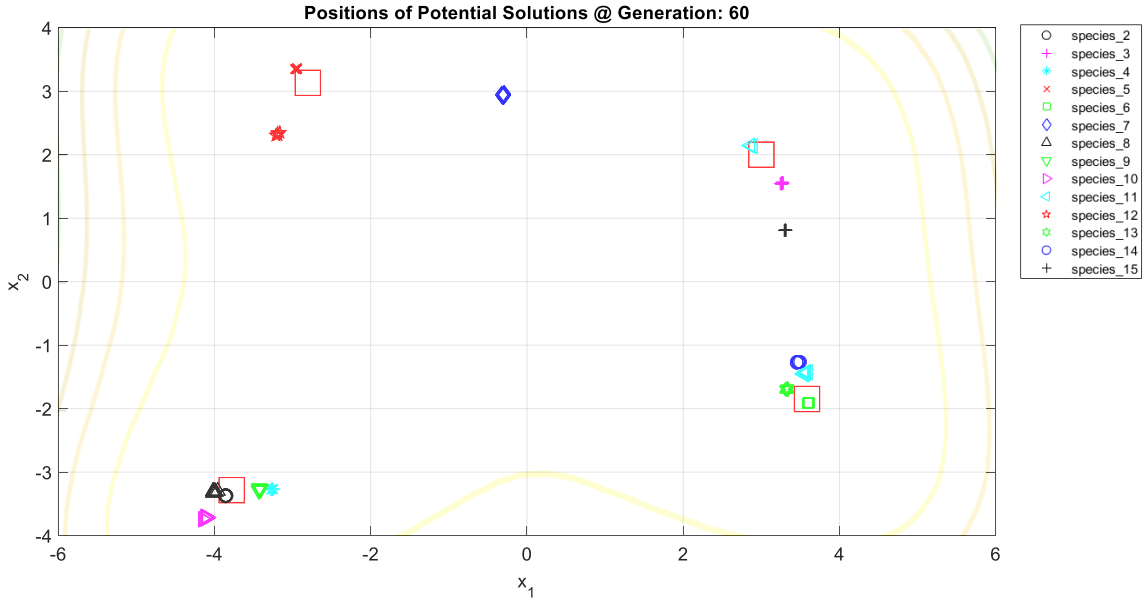


Figure C.11 At generation 60, the search continues.

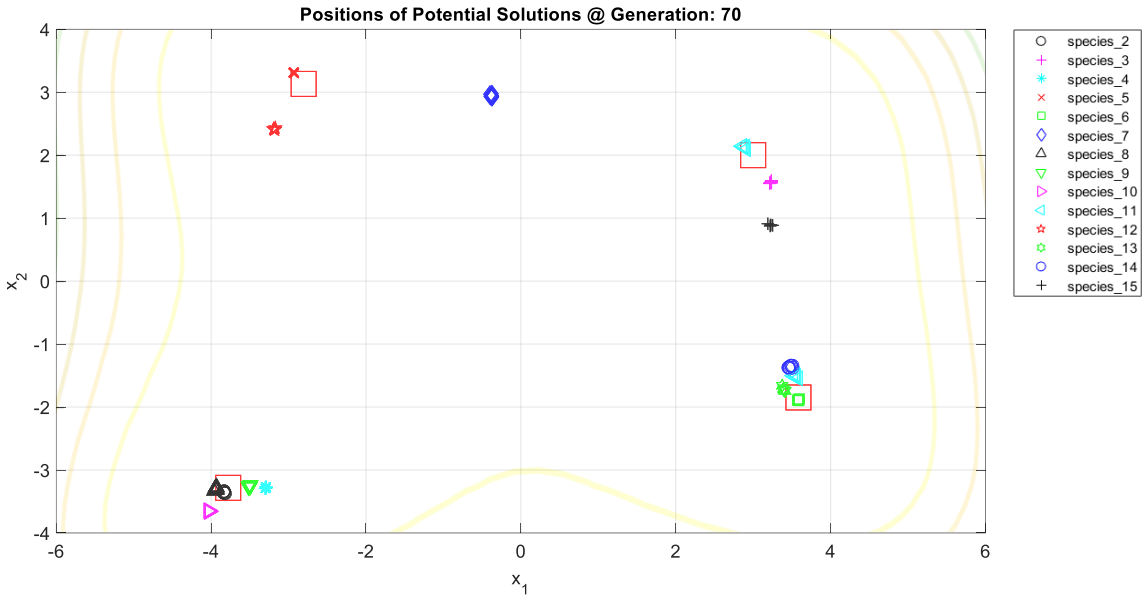


Figure C.12 At generation 70, some species are seen inside the global minima's red square.

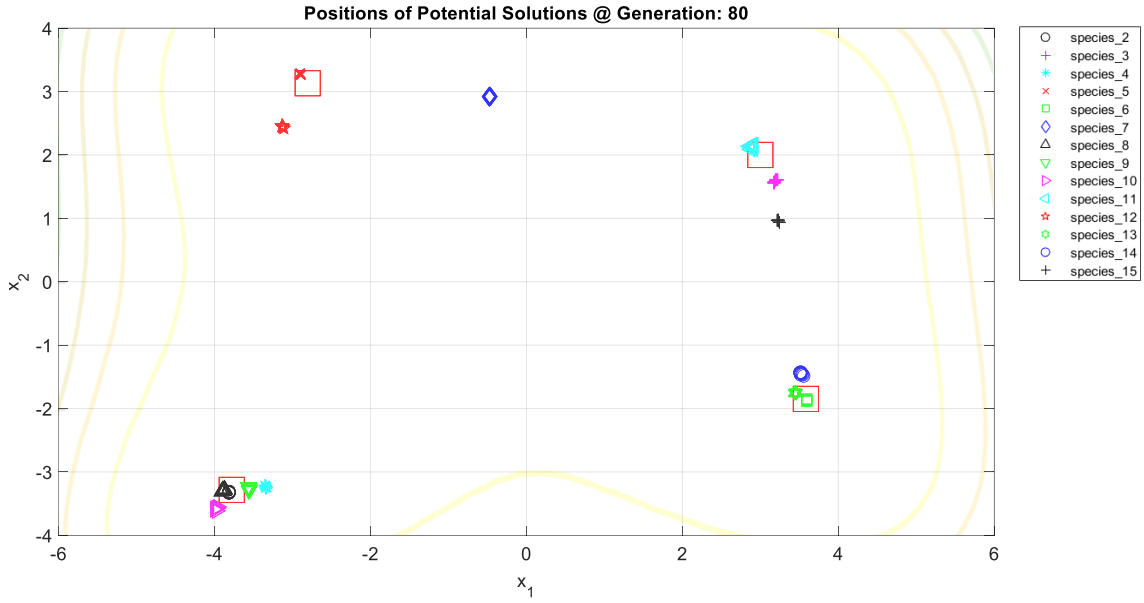


Figure C.13 At generation 80, Species_11 are now identical in their search positions.

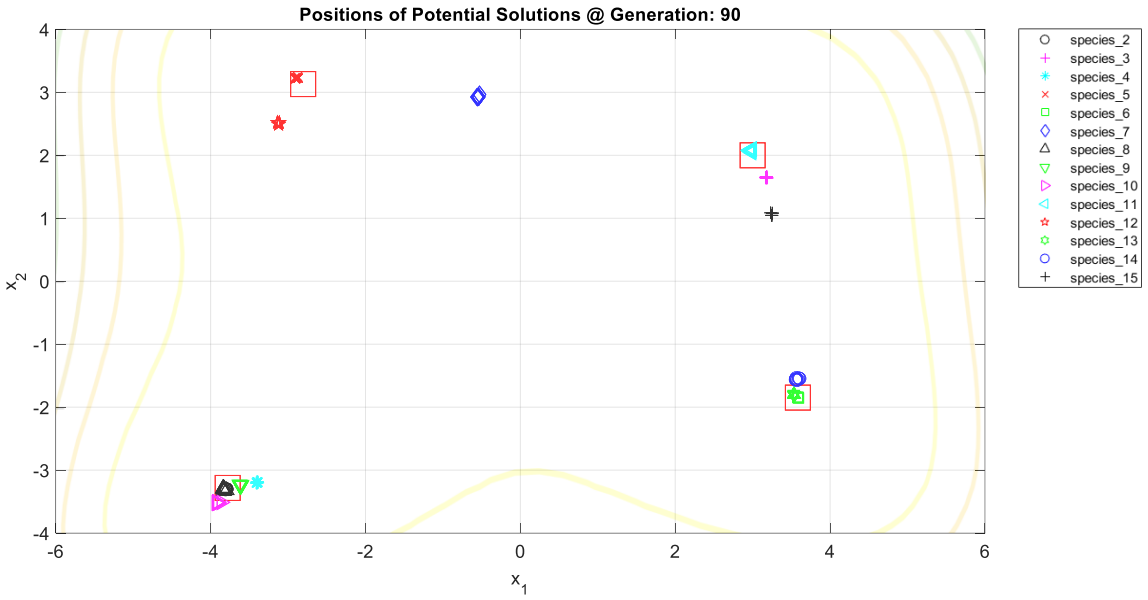


Figure C.14 At generation 90, more species are inside one of the red squares, with the remaining six species still trying to locate the minima.

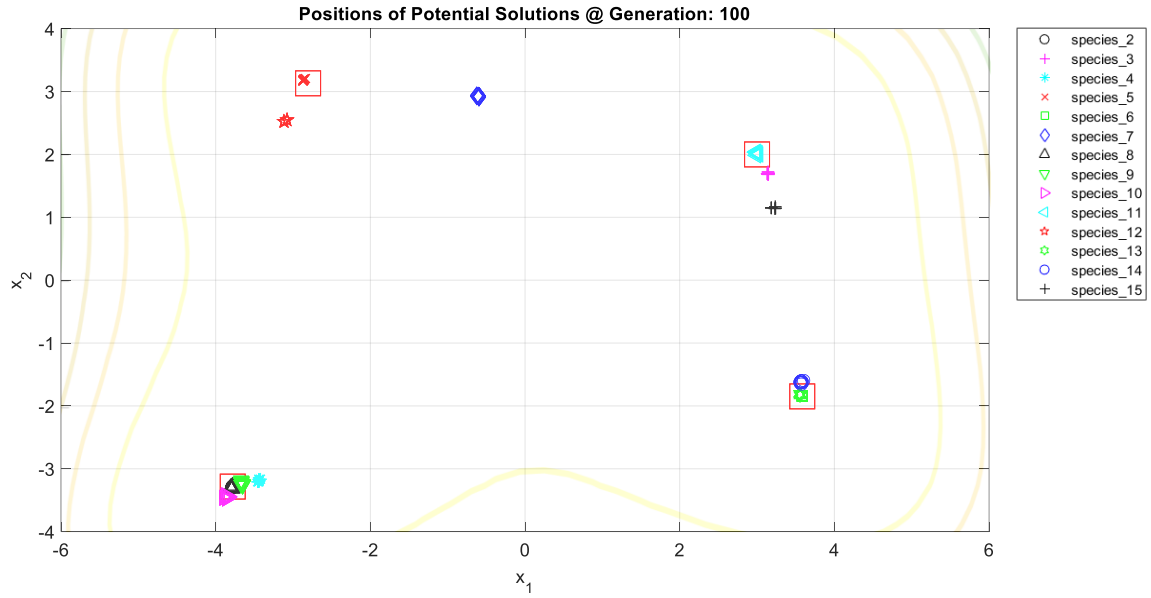


Figure C.15 At generation 100, the remaining species can be seen converging towards the minima.

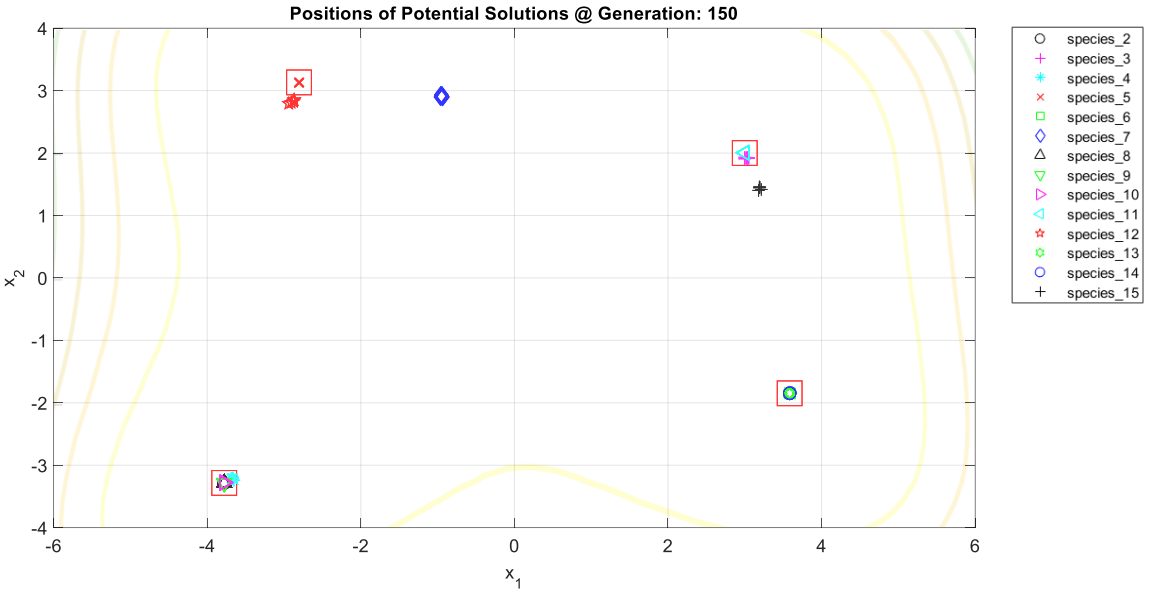


Figure C.16 At generation 150, Species_7 has moved out of the local minima and towards the global minima.

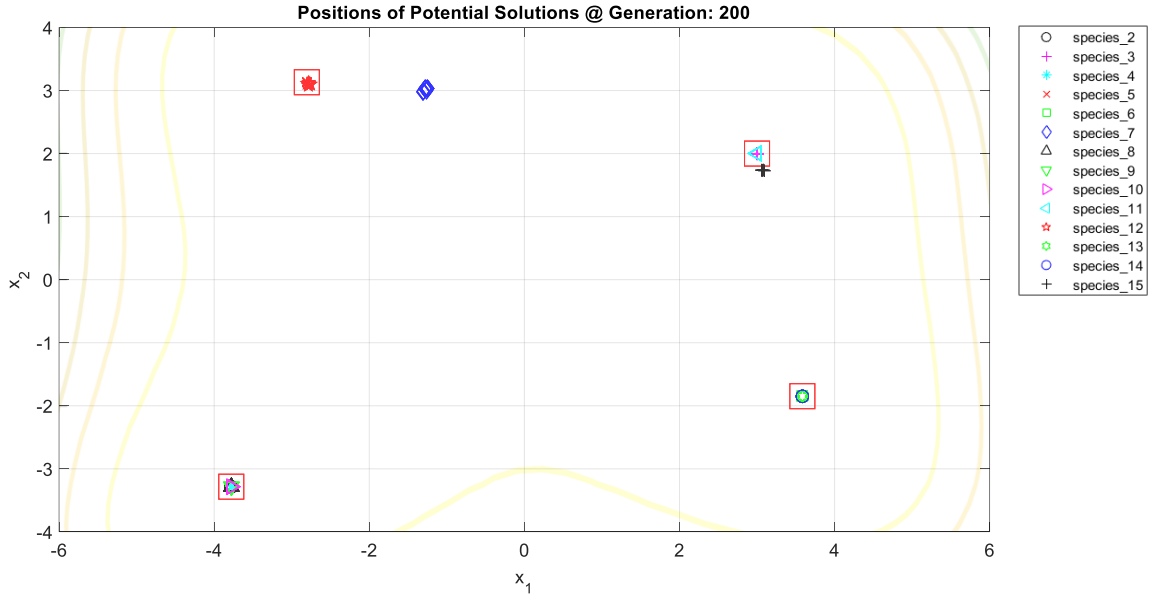


Figure C.17 At generation 200, most species have landed inside one of the red squares.

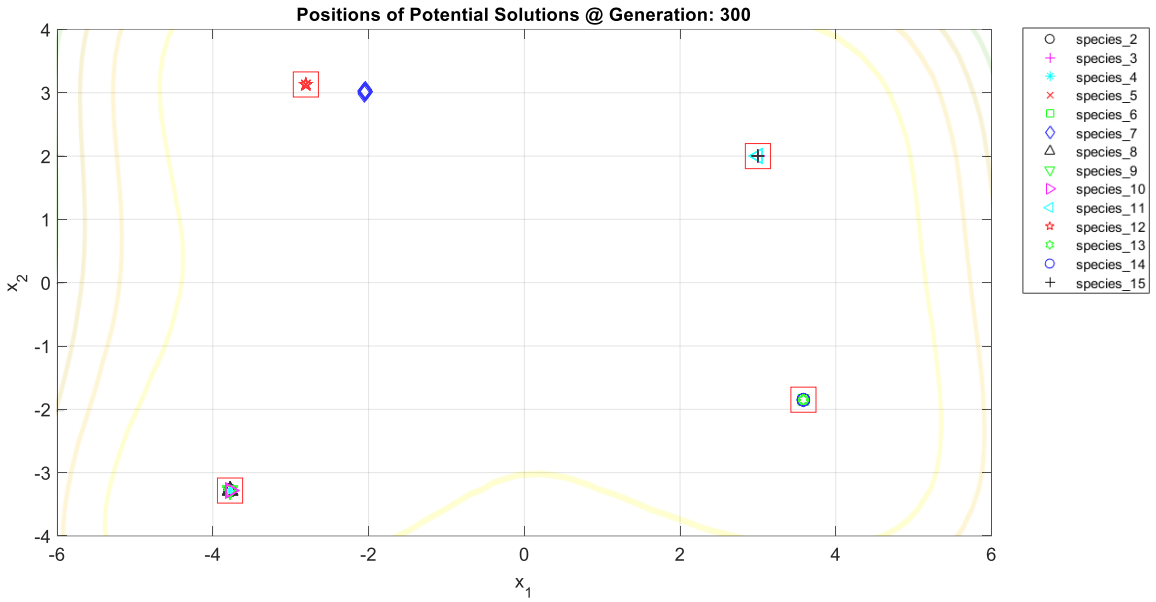


Figure C.18 At generation 300, all the species except Species_7 can be seen inside one of the red squares.

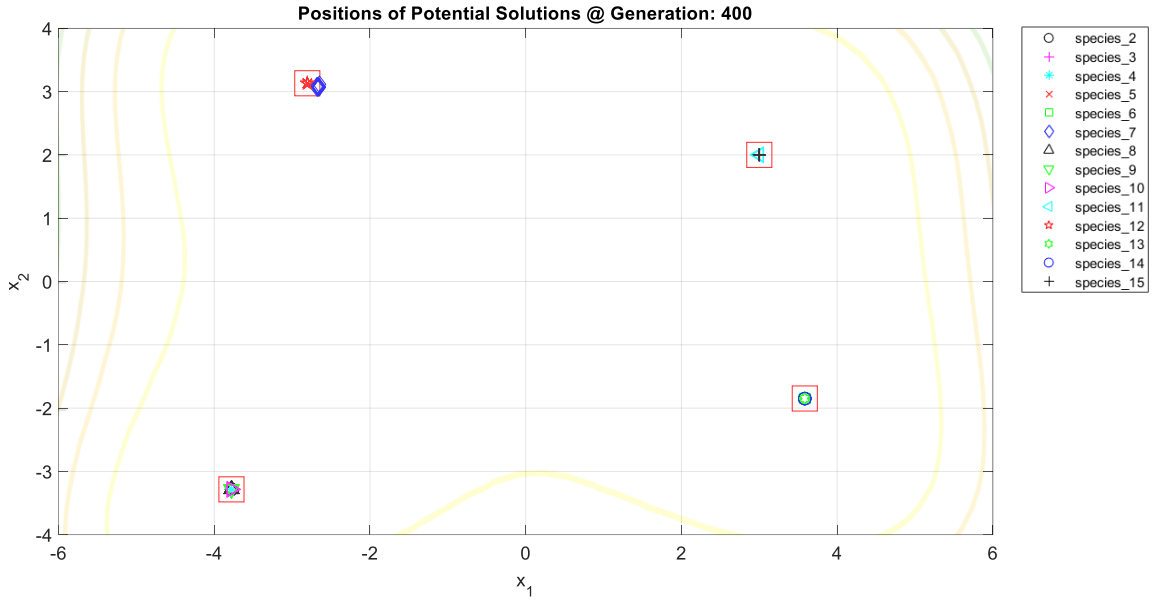


Figure C.19 At generation 400, Species_7 has located one of the global minima.

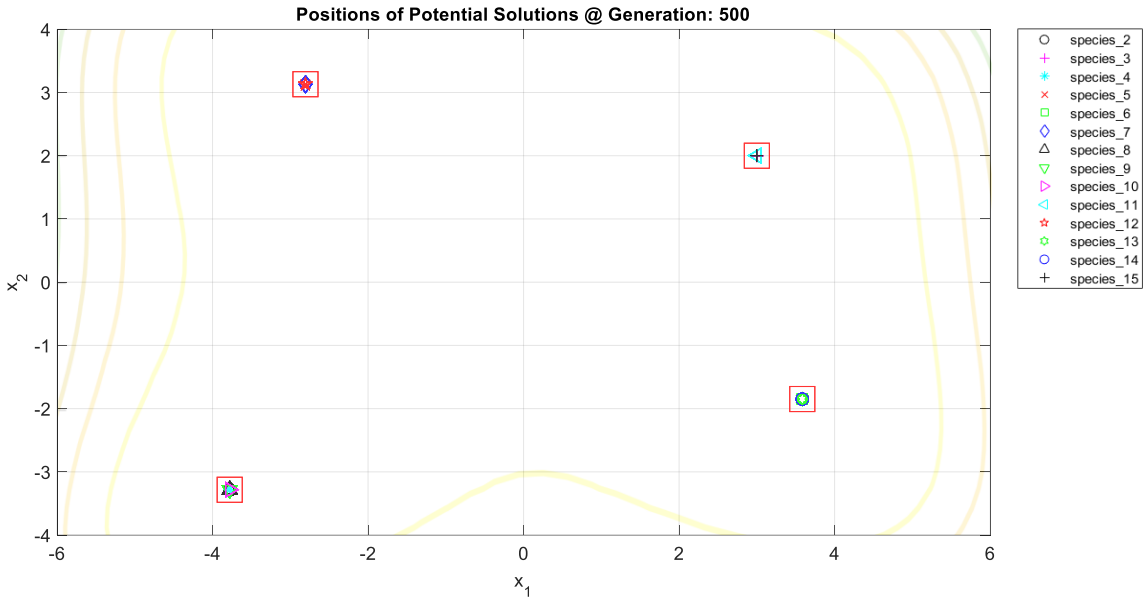


Figure C.20 At generation 500, all the species have found the global minima indicated by the red squares.

C.2 Heatmap Visualization

Additional analysis of EVLNN's search behavior for the Himmelblau-2D (f_{11}) function is performed using a visual heatmap. The active heatmap visualization is written in MatLab R2020a software. Figure C.21 shows the heatmap at the end of EVLNN's search operation for f_{11}. The numbers in columns two to five represent the proximity of the species' Euclidean distance to the global minima. The species' Mean Absolute Error (MAE) is also calculated and indicated in the heatmap in the first column. The heatmap's warmer colors represent that the species is further away from the global minima, whereas the cooler colors represent that the species is nearer to the global minima. From Figure C.21, Species_2 is the best performer in the diagram with the lowest MAE of 0.0000 and a mean Euclidean distance of 0.00016 to global minima two (-3.779, -3.283). *Species_7* is the worst performer with the MAE of 0.00024 and a mean Euclidean distance of 0.0020 to global minima three (-2.805, -3.131). *Species_2*, *Species_4*, *Species_8*, *Species_9,* and *Species_10* have located the global-minima-two with the Euclidean distance measured at 0.00016, 0.00086, 0.00170, 0.00117, and 0.00118, respectively. *Species_3*, *Species_11,* and *Species_15* have located

global-minima-one at (3,2) with the Euclidean distance measured at 0.00056, 0.00242, and 0.00319, respectively. *Species_5*, *Species_7,* and *Species_12* have located global-minima-three at (-2.805, -3.131) with the Euclidean distance measured at 0.00072, 0.00220, and 0.00078, respectively. *Species_6*, *Species_13,* and *Species_14* have located global-minima-four at (3.584, -1.848) with the Euclidean distance measured at 0.00071, 0.00271, and 0.00205, respectively. The heatmap illustrates an important finding that EVLNN's speciation approach has achieved the parallelism required to locate all the global optimas with low MAE values. The best solutions for the 4 global minima are found in *Species_3*, *Species_2*, *Species_5* and *Species_6* at (3,2), (-3.779, -3.283), (-2.805, 3.132) and (3.584, -1.848), respectively.

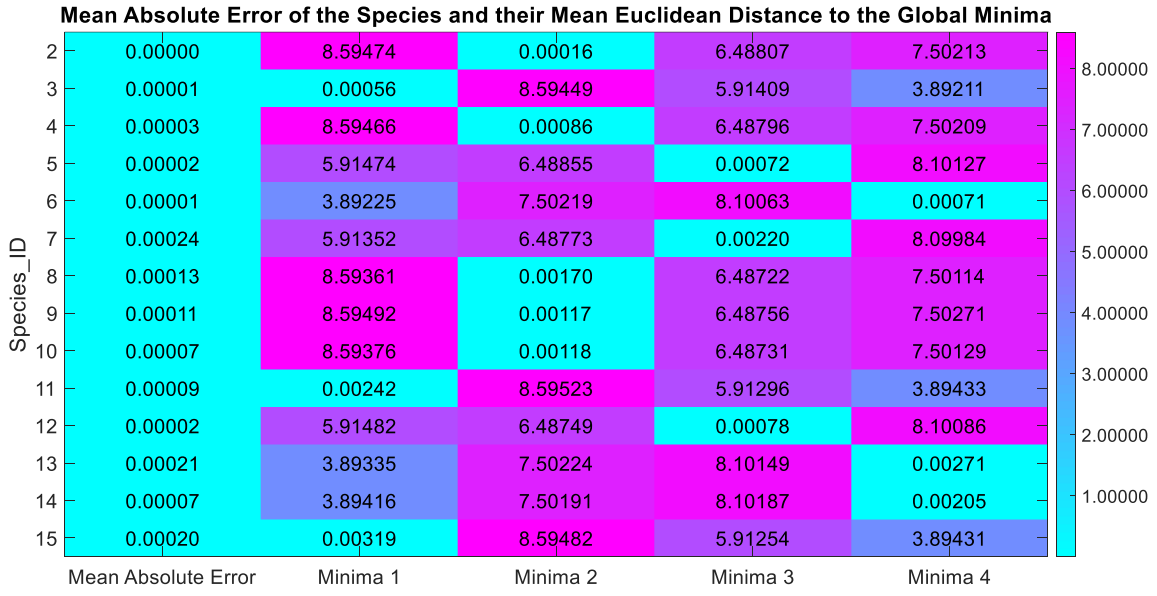


Figure C.21 Analysis of EVLNN search behavior using a visual heatmap.

In addition, a bubble chart, written in MatLab R2020a software, is used to analyze and visualize the species density around the respective global basins. Figure C.22 shows the bubble chart at the end of EVLNN's search operation with a larger bubble size corresponding to a higher species density and a higher number of solution candidates. The number of solution candidates present at the global minima 1, 2, 3, and 4 is 27, 49, 26, and 23, respectively, resulting in different bubble sizes. From Figure C.22, is observed that the distribution of the species on each basis was uneven. This outcome is expected as EVLNN takes a stochastic approach in the explorative search process, resulting in an even number of solution candidates at the various basins.

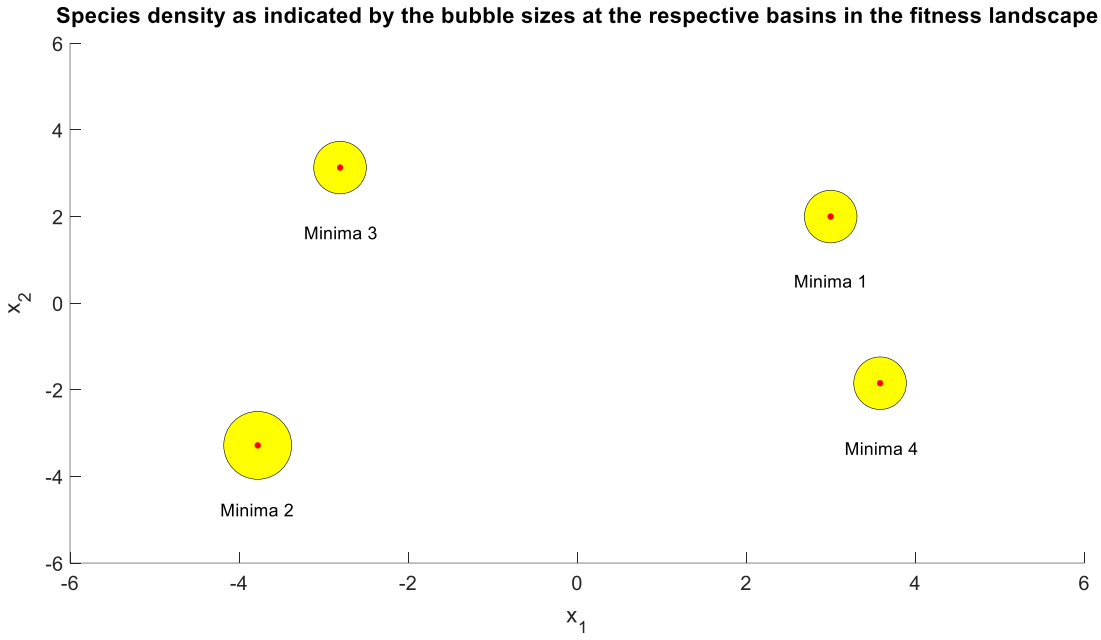


Figure C.22 Analysis of EVLNN search behavior using a bubble chart.