UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

UNIVERSITÀ DEGLI STUDI ROMA TRE

Department of Industrial, Electronic, and Mechanical Engineering

PHD IN PHOTOVOLTAICS

Curriculum D: Power Electronics and Control

DOTTORATO DI INTERESSE NAZIONALE IN "PHOTOVOLTAICS"

DEEP LEARNING MODEL WITH AN ADAPTIVE HARRIS HAWK OPTIMIZATION FOR MULTI-STEP AHEAD SOLAR POWER FORECASTING

Supervisor:
Prof. Francesco Riganti-Fulginei

PhD Program Director:
Prof. Giovanni Spagnuolo

PhD Candidate:
Rafiq Asghar
MAT. 8860800024

XXXVIII Cycle

2022 - 2025

Funded by the European Union

MUR Ministero dell'Università e della Ricerca

Dottorato di Interesse Nazionale

PhD PV



Abstract

The growing integration of Photovoltaic (PV) power production continuously stresses electricity networks due to its unpredictable and unstable nature. The impacts are more critical, particularly for highly PV-penetrated power systems. Accurate multistep prediction can help utility companies to handle the PV intermittency issue and balance the supply and demand more effectively. However, achieving precise multistep prediction remains a significant challenge, particularly for real-world datasets. In real-world datasets, PV power output is influenced by complex, unpredictable, and nonlinear variables, including solar irradiance, wind speed, cloud movements, temperature, seasonal changes, and PV operational conditions. All these factors change over time, resulting in highly unpredictable and non-stationary patterns of power production, making the accurate prediction of real-world PV datasets extremely challenging. To overcome these challenges, this thesis explores the efficiency of a deep learning-based method for multistep PV power forecasting.

The proposed framework employed an adaptive Harris Hawk-optimized dual-stream attention-based hybrid deep learning model capable of capturing short- and long-term dependencies in a real-world PV power dataset. By integrating local feature extraction, temporal modeling, dynamically assigning weights, and improved optimization capabilities, the proposed method can learn intricate nonlinear and complex relationships in historical PV power datasets. A direct multi-step forecasting method is used, which predicts future time steps using historical data, reducing error transmission and improving long-term prediction accuracy.

The proposed model is validated using two real-world datasets acquired from Alice Springs, Australia, and ENEA in Anguillara Sabazia, Italy. For the Alice Springs dataset, the proposed method is evaluated using 1-step, 3-step, and 6-step forecasting horizons to determine its ability to maintain reliability and precision as the prediction window increases. Conversely, the ENEA dataset is employed to assess the model's capability for real-time forecasting, with an emphasis on both accuracy and computational efficiency. The experimental findings demonstrate that the proposed model outperforms the baseline models in terms of forecasting

accuracy and performance on the Alice Springs dataset. Additionally, on the ENEA dataset, the proposed model demonstrated high forecasting precision with low computational expenses, which are the main requirements for real-time forecasting. The results validate the applicability and usefulness of the proposed model for intricate multistep and real-time PV power forecasting problems. Hence, the proposed design is robust and adaptable, making it ideal for energy predictions, complex engineering design, deep learning parameter tuning, and various other real-world problems.

Keywords: Solar energy, PV power forecasting, dual-stream deep learning, multi-head attention, adaptive Harris Hawk Optimization, performance analysis.

List of Publications

Journal Articles

J1. R. Asghar, F. Riganti Fulginei, M. Quercio, and A. Mahrouch “Artificial Neural Networks for Photovoltaic Power Forecasting: A Review of Five Promising Models”, *IEEE Access*, Vol. 12, pp. 90461–90485, July 2024. DOI: 10.1109/ACCESS.2024.3420693.

J2. R. Asghar, M. Quercio, L. Sabino, A. Mahrouch, and F. Riganti Fulginei, "A Novel Dual-Stream Attention-Based Hybrid Network for Solar Power Forecasting," *IEEE Access*, Vol. 13, pp. 59596-59609, March 2025, DOI: 10.1109/ACCESS.2025.3555810.

J3. R. Asghar, M. Quercio, D. Milillo, Misbah Ullah, Syed Zarak Shah, and F. Riganti Fulginei, " Adaptive Harris Hawk optimization algorithm for solving global optimization problems," Accepted, *Engineering Computations*, Emerald, 25 March 2026.

Conference Publications

C1. R. Asghar, F. Riganti Fulginei, M. Quercio, A. Ahmad, and M. Abusara “Application of deep learning algorithms for BIPV power forecasting in Italy”, *2024 IEEE International Conference on Environment and Electrical Engineering and 2024 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe),* Rome, Italy, 20 November 2024. 10.1109/EEEIC/ICPSEurope61470.2024.10750971.

C2. R. Asghar, F. Riganti Fulginei, M. Quercio, M. Maoz, S. Sabino, and M. Abusara “Day-ahead photovoltaic power forecasting using a hybrid BiLSTM-CNN model”, *2024 IEEE Sixth International Conference on Intelligent Computing in Data Sciences*, 25 November 2024, Marrakech, Morocco. 10.1109/ICDS62089.2024.10756380.

C3. R. Asghar, F. Riganti Fulginei, M. Quercio, S. Sabino, and M. Abusara “A hybrid LSTM-CNN model for short-term photovoltaic power forecasting

in Italy", *2024 IEEE International Symposium on Consumer Technology (ISCT)*, Kuta, Bali, Indonesia, 17 December 2024.10.1109/ISCT62336.2024.10791191.

C4. R. Asghar, M. Quercio, L. Sabino, D. Milillo, A. Mahrouch, and F. Riganti Fulginei "Solar irradiance estimation using a hybrid deep learning model with metaheuristic algorithms", *IEEE 9th International Symposium on Innovative Approaches in Smart Technologies*, Gaziantep–Türkiye, 12 August 2025. 10.1109/ISAS66241.2025.11101966.

Contents

Abstract ..... **iv**

List of Publications ..... **vi**

List of Tables ..... **xi**

List of Figures ..... **xii**

Nomenclature ..... **xv**

1 Introduction **1**

1.1 Research Background 1

1.2 Problem Statement 4

1.3 Research Objectives 7

1.4 Significance of the Study 8

1.5 Thesis Structure 10

2 Fundamentals and Related Work **13**

2.1 Solar Power Forecasting: Concepts and Challenges 13

2.2 Classical and Data-Driven Forecasting Models 16

2.2.1 Physical Models 16

2.2.2 Classical Statistical Models 21

2.2.3 Machine Learning Models 25

2.3 Overview of Deep Learning Architectures 28

2.4 Deep Learning Models for Time Series Forecasting 31

2.5 Attention mechanisms and their applications in forecasting 34

2.5.1 Attention in Natural Language Processing 34

2.5.2 Attention in Time Series Forecasting 36

2.6 Metaheuristic Algorithms for Hyperparameter Tuning 37

2.6.1 Harris Hawk Optimization and Its Variants 40

2.7 Summary and Research Gaps 42

3 Dual-Stream Attention-based Hybrid Deep Learning Model **45**

3.1 Overview of the Proposed Framework 45

3.2 CNN for Local Feature Extraction 48

3.3 BiLSTM for Long-Term Temporal Dependencies 50

3.4 Multi-head Attention Mechanism for Dynamic Weighting of Feature 54
3.5 Integration of BiLSTM-CNN–Multi-head Attention 56
3.6 Workflow of the Proposed BiLSTM-CNN-MHA Model....59
3.7 Summary....64
4 Adaptive Harris Hawk Optimization Algorithm (ADHHO) 67
4.1 Nature-Inspired Metaheuristics: A Review 67
4.2 Mathematical Frameworks of the Original HHO....70
4.2.1 Exploration Phase 70
4.2.2 Transition Phase 72
4.2.3 Exploitation Phase....72
4.3 Adaptive Harris Hawk Optimization (ADHHO) 75
4.3.1 Chaotic Optimizations....76
4.3.2 Nonlinear Control Parameter 77
4.3.3 Adaptive Inertia Weight 78
4.3.4 Adaptive Decay Factor....79
4.4 ADHHO Algorithmic Workflow 80
4.5 Convergence Characteristics on IEEE CEC Benchmark Suites 81
4.5.1 Convergence Curves on CEC 2014 Benchmark Suite 82
4.5.2 Convergence Curves on CEC 2017 Benchmark Suite.... 89
4.6 Summary....96
5 Experimental Evaluation of the Alice Springs Dataset 100
5.1 Dataset Requirement and Characteristics for Deep Learning Models 100
5.2 Alice Springs Dataset: Characteristics and Visualization 102
5.3 Data Cleaning: Missing Values and Outliers 107
5.4 Data Normalization and Sequence Generation 111
5.5 Proposed Model Architectures 113
5.6 Baseline Models for Analysis and Comparison 116
5.7 Performance Metrics 119
5.8 Results and Discussions.... 121

5.9 Summary 133
6 Assessment of the Proposed Model on the ENEA Dataset 136
6.1 Description of ENEA and Energy Community 136
6.2 Data Collection and Visualization 139
6.3 Data Analysis and Preprocessing 140
6.4 Baseline Models for Analysis and Comparison 144
6.5 Performance Metrics for the ENEA dataset 146
6.6 Description of the problem strategy 147
6.7 Results and Discussions 148
6.8 Summary 155
7 Conclusions and Future Direction 157
7.1 Conclusions 157
7.2 Future Direction 161
Acknowledgement 163

List of Tables

2.1	Advantages and disadvantages of various physical models.....	20
2.2	Advantages and disadvantages of various statistical models.....	25
2.3	Advantages and disadvantages of various machine learning models	28
4.1	Initial parameter settings of all tested algorithms.....	82
4.2	Description of the CEC2014 benchmark Suite.....	84
4.3	Performance analysis of ADHHO and HHO on the CEC2014 benchmark suite.....	85
4.4	Performance analysis of ADHHO and other algorithms on CEC2014 test functions (F1-F22)	86
4.5	Description of the CEC2017 benchmark suite.....	91
4.6	Performance analysis of ADHHO and HHO on the CEC2017 benchmark suite.....	92
4.7	Performance analysis of ADHHO and other algorithms on the CEC2017 benchmark suite.....	93
5.1	Description of the dataset.....	105
5.2	Detailed description of the single baseline models for comparison..	117
5.3	Detailed description of the hybrid baseline models for comparison.	118
5.4	One-step ahead forecasting results of the studied models.....	122
5.5	Three-step ahead forecasting results of the studied models.....	126
5.6	Six-step ahead forecasting results of the studied models.....	130
6.1	Profiles of the individuals participating in the trial of the smart community.....	138
6.2	Description of the users and their annual electricity consumption...	139
6.3	Detailed description of the baseline models for comparison on the ENEA dataset.....	145
6.4	Performance comparison of the forecasting models on the ENEA dataset.....	149
6.5	Comparison of computational complexity and training time for all forecasting models.....	150

List of Figures

1.1	The impact of the PV installations on the power systems.....	2
1.2	Workflow of Hyperparameter optimization of a deep learning model.....	4
1.3	Content and outcomes of the thesis.....	10
2.1	Worldwide solar power cumulative installations and yearly increment. The data for the bar chart is obtained from the IEA..	14
2.2	Overview of the worldwide solar power installation capacity in 2024 alone. Sourced by: International Energy Agency	14
2.3	Number of publications on physical models (source: Data are obtained from the Scopus database from 2018 to 2024).....	17
2.4	Spatial resolution Vs. time horizon for different physical modes	18
2.5	Relationship between Machine Learning and Deep Learning.....	29
2.6	Basic workflow of the Metaheuristic Optimization Algorithm.....	38
2.7	Four Classes of Metaheuristic Algorithms, based on their principle of operation.....	39
3.1	A two-layer structure of the CNN model.....	48
3.2	The flow Operation of the LSTM unit.....	52
3.3	The BiLSTM basic structure, showing two LSTMs operating simultaneously on each time step.....	53
3.4	Dual-stream BiLSTM-CNN with multi-head attention structure	56
3.5	The flowchart of the dual-stream BiLSTM-CNN with multi-head attention.....	60
4.1	Description of the three phases of HHO.....	71
4.2	Comparison of escape energy using different nonlinear functions.....	77
4.3	Behavior of escape energy during 500 iterations.....	78
4.4	Flowchart of the ADHHO algorithm technique.....	81
4.5	Convergence curve for ADHHO and other evaluated algorithms on CEC2014.....	89
4.6	Convergence curve for ADHHO and other evaluated algorithms on CEC2017.....	96
5.1	The map of Alice Springs and specifications of the Trina Solar 1A site.....	103
5.2	Historical data on PV output and weather variables at the BP solar power installation.....	105
5.3	Daily and hourly distribution of PV power output.....	106
5.4	Heatmap correlation of weather variables and PV power.....	107
5.5	Detecting and removing outliers from Trina Solar data.....	111
	(a) Box plots of PV power and weather variables before IQR.....	110

	(b) Box plots of PV power and weather variables after IQR.....	111
5.6	Architecture of the proposed hybrid deep learning model with an adaptive Harris Hawk optimization.....	114
5.7	Radar chart of the studied models using 1-step ahead forecasting	122
5.8	Comparison of the studied models' predictive capabilities using 1-step ahead forecasting.....	124
5.9	Comparison of the studied models on a sunny day from 1-step ahead forecasting.....	124
5.10	Comparison of the studied models on a cloudy day from 1-step ahead forecasting.....	124
5.11	Regression analysis between forecasted and observed values of the studied models on a 1-step ahead forecasting.....	125
5.12	Radar chart of the studied models using 3-step ahead forecasting	126
5.13	Comparison of the studied models' predictive capabilities using 3-step ahead forecasting.....	128
5.14	Comparison of the studied models on a sunny day from 3-step ahead forecasting.....	128
5.15	Comparison of the studied models on a cloudy day from 3-step ahead forecasting.....	128
5.16	Regression analysis between forecasted and observed values of the studied models on a 3-step ahead forecasting.....	129
5.17	Radar chart of the studied models using 6-step ahead forecasting.....	131
5.18	Comparison of the studied models' predictive capabilities using 6-step ahead forecasting.....	132
5.19	Comparison of the studied models on a sunny day from 6-step ahead forecasting.....	132
5.20	Comparison of the studied models on a cloudy day from 6-step ahead forecasting.....	132
5.21	Regression analysis between forecasted and observed values of the studied models on a 6-step ahead forecasting.....	133
6.1	Basic working of the DHOMUS software for controlling the flow of information.....	137
6.2	Heatmap of the Pearson's correlation matrix of weather variables and energy.....	141
6.3	Anomaly detection in the ENEA dataset, for demonstration, only one week (06-05-2024 to 12-05-2024) is presented.....	143
6.4	Removing anomaly from the ENEA dataset, for demonstration, only one week (06-05-2024 to 12-05-2024) is presented.....	144
6.5	Comparison of the forecasting models' predictive capabilities for PV users (DU_1, DU_2, DU_7, and DU_9).....	152
6.6	Scatter plot between actual and forecasted values of the studied models for energy production of PV users.....	153

6.7 Comparison of the performance of all forecasting models for simulated PV user..... 155

Nomenclature

ADALINE	Adaptive Linear Neuron
ADHHO	Adaptive Harris Hawk Optimization
ANN	Artificial Neural Networks
ARIMA	Autoregressive Integrated Moving Average
AROME	Application of Research to Operations at Mesoscale
BiLSTM	Bidirectional Long Short-Term Memory
CEC	Congress on Evolutionary Computation
CI	Cloud Index
CIRACast	Cooperative Institute for Research in the Atmosphere
CNNs	Convolutional Neural Networks
DA	Differential Evolution
DBN	Deep Belief Networks
DHOMUS	Data HOMes and USers
DKASC	Desert Knowledge Australia Solar Centre
DLMs	Deep Learning Models
DT	Decision Tree
EA	Evolutionary algorithms
ENEA	Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile
FARMS	Fast All-sky Radiation Model for Solar Applications
FiTs	Feed-in Tariffs
GAN	Generative Adversarial Network
GAs	Genetic Algorithms
GELU	Gaussian Error Linear Unit
GMDH	Group Method of Data Handling
GRU	Gated Recurrent Unit
GWO	Grey Wolf Optimizer
HHO	Harris Hawk Optimization
IEA	International Energy Agency
k-NN	K-Nearest Neighbors
LF	Levy Flight

LOCE	Levelized Cost of Electricity
LR	Linear Regression
LSTM	Long Short-Term Memory
MAD	Median Absolute Deviation
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MBE	Mean Bias Error
MHA	Multi-head Attention
MOAs	Metaheuristic Optimization Algorithms
MLMs	Machine Learning Models
MLP	Multilayer perceptron
MLR	Multiple Linear Regression
MSE	Mean Squared Error
NLP	Natural Language Processing
NRMSE	Normalized Root Mean Squared Error
NWP	Numerical Weather Prediction
PLCC	Power Line Carrier Communication
PSO	Particle Swarm Optimization
PV	Photovoltaics
R	Correlation Coefficient
R^2	Coefficient of Determination
RBF	Radial Basis Function
RF	Random Forests
RLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
RTM	Radiative Transfer Model
S2S	Sequence-to-Sequence
SSA	Salp Swarm Algorithm
SVM	Support Vector Machine
SVR	Support Vector Regression
WOA	Whale Optimization Algorithm
WMAPE	Weighted Mean Absolute Percentage Error
WRF	Weather Research and Forecasting

1

Introduction

1.1 Background and Motivation

Over the past decades, the demand for renewable energy has increased significantly due to rising concerns about global warming, health issues, and depletion of non-conventional energy sources [1], [2]. Solar power has emerged as a viable alternative that can provide an unlimited, sustainable, and cost-effective energy supply [3]. This energy source can be deployed on various scales, ranging from small rooftop installations to huge utility-scale plants, and is more adaptable in addressing both local and regional energy demands. However, PV output power is inherently erratic and unstable due to its dependence on the weather conditions [4]. Its unstable nature can significantly impact the operation and stability of utility grids. The output power can drop from the rated value to a minimum value in minutes or seconds. Such instant power reductions can lead to voltage fluctuations, infrastructure damage, and power failures [5]. The impacts can be more critical when the PV power represents a large portion of the total generated power [6]. Fig. 1.1 provides details on how the size of the PV systems impacts the power systems. To avoid these negative impacts on the entire power system, precise PV power forecasting is essential. Accurate forecasting can help utility companies balance the supply and demand more effectively. It provides dependable and efficient approaches for fulfilling the minimum generation and scaling requirements [3], [7]. It promotes the integration of more renewable energy resources, lowering dependence on fossil fuels. Additionally, it can enhance the system's reliability and reduce operational costs.

Earlier, forecasting techniques used physical, statistical, and regression-based models. These models are transparent, interpretable, and computationally less expensive [9]. However, they struggle to learn sudden variations in the real world

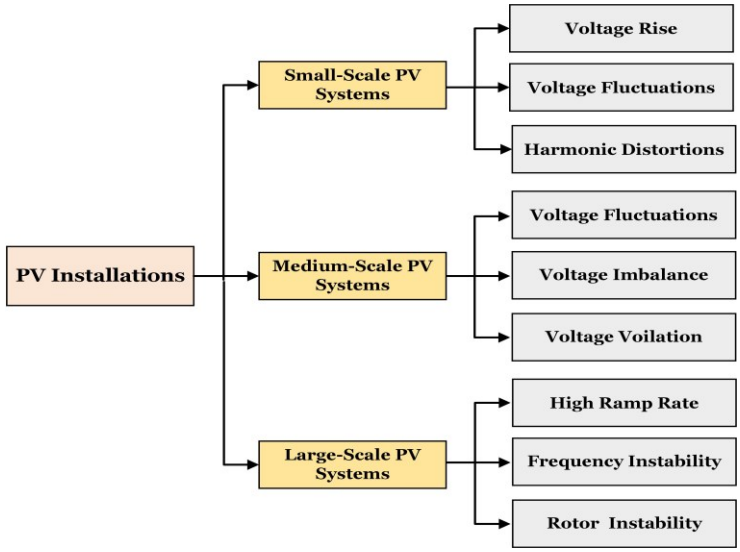


Figure 1.1: The impact of the PV installations on the power systems[6], [8]

solar irradiance and PV power datasets [10]. Additionally, these models presume linearity and stationarity in datasets and may fail to capture nonlinear relationships. MLMs have overcome some of the limitations of these models by learning complex and nonlinear relationships. These models can present more intricate, data-driven approaches compared to classical and regression-based models [11]. Nevertheless, conventional MLMs, such as SVM, RF, and DT, possess a shallow structure and limited learning capacity [3]. These models struggle to generalize, particularly on high-dimensional or noisy datasets. Recently, DLMs, particularly CNNs, LSTMs, and BiLSTMs, have shown superior performance for time series predictions. Studies have shown that these models outperform traditional ML models, including SVM, LR, RF, and SVR [12], [13]. These models can handle multivariate and high-dimensional inputs [14]. Hence, these models have become increasingly adapted for forecasting tasks.

Despite these benefits, DLMs still have limitations in detecting complex patterns in modern PV power datasets. This is because modern real-world PV data contains a combination of linear patterns, complex relationships, noise, and seasonality; single models, such as CNN, LSTM, and BiLSTM, may not be enough to capture these factors. Recently, the research focus has shifted toward integrating multiple DLMs, which has greatly improved pattern recognition capabilities, forecasting accuracy, and convergence rate. In hybrid models, the

most feasible choice for PV power forecasting is CNN-BiLSTM [15], [16]. This is because CNNs can better capture sudden or local patterns, while BiLSTM can excel in capturing long-term temporal dependencies. Some studies suggest that altering the sequence order in CNN-BiLSTM can enhance the accuracy and performance. For instance, the BiLSTM-CNN hybrid model can better explain the complex and nonlinear patterns in PV power datasets compared to CNN-BiLSTM [17], [18]. While BiLSTM-CNN achieved better performance and is computationally less expensive compared to CNN-BiLSTM, it still treats the time series datasets equally. In contrast, in a real-world scenario, not every timestep adds the same contribution to the final output. At this stage, the application of an attention mechanism is vital. The attention mechanism decides which timesteps or attributes are crucial for PV power forecasting. It assigns different weights to each timestep [19] in the fused features of the BiLSTM-CNN, improving robustness and interpretability. Hence, the combination of BiLSTM-CNN and attention mechanism creates a robust design that can capture both short and long-term dependencies and dynamically focus on important features in PV datasets.

Although the hybrid combination of BiLSTM-CNN and multi-head attention has shown promising results in multiple fields, its efficiency is still highly reliant on suitable hyperparameter settings. Studies showed that suitable hyperparameters can enhance the model's overall performance by 8% [20]. Hence, several methods are available for hyperparameter tuning, although MOAs can handle complex optimization problems and often exhibit acceptable solutions [21]. These algorithms are grouped into four main classes, including evolutionary, human-based, physics-based, and swarm-based algorithms [22]. However, swarm intelligence algorithms gained greater attention for solving nonlinear optimization problems. These algorithms get inspiration from the collaborative behavior of decentralized systems [23], including bacteria foraging, fireflies, insect swarms, and others [24]. Some well-known examples of these algorithms are PSO, GWO, and WOA. However, similar to other algorithms, swarm intelligence algorithms have various limitations, including slow convergence, particularly in the exploitation phase. These algorithms can lose diversity and experience a decline in performance in multi-dimensional optimization tasks. HHO, another prominent MOA, gets inspiration from the cooperative and hunting nature of the Harris hawks. This algorithm has multiple distinctive features

compared to various swarm intelligence algorithms. Firstly, this algorithm has a strong equilibrium between exploration and exploitation. Secondly, this algorithm employs a time-varying method that improves with iterations, resulting in a smooth transition from exploration to exploitation. Finally, it exhibits a gradual convergence behavior, which increases the possibility of finding the most favorable solutions. Hence, HHO can better optimize the hyperparameters of the single and hybrid deep learning models. Fig. 1.2 illustrates the workflow of how the HHO optimized the DLMs to enhance precision and accuracy.

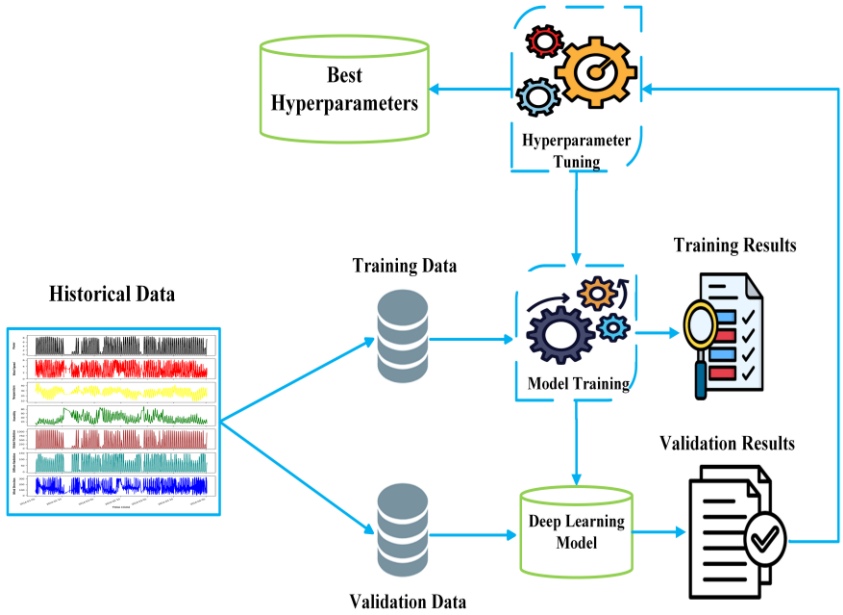


Figure 1.2: Workflow of Hyperparameter optimization of a deep learning model

1.2 Problem Statement

Despite recent advancements in forecasting techniques, achieving precise and reliable prediction of PV power remains a significant challenge. PV power output depends on various weather-related variables, including solar irradiance, wind speed, wind direction, moving clouds, temperature, and relative humidity [25], [26]. All these variables show complex, nonlinear, and time-varying behaviors. These nonlinear and intricate interactions make the prediction more challenging, particularly in erratic and dynamic environmental conditions. Additionally,

geographical factors, forecasting horizons, and technical factors further contribute to the complexity of forecasting [3]. Earlier, forecasting techniques, including physical, statistical, and machine learning models, struggled to capture sudden variations such as cloud-induced variations, weather fluctuations, and seasonality changes in PV power production. Additionally, these models struggle to generalize, particularly on high-dimensional or noisy datasets. Recently, hybrid deep learning models, such as BiLSTM-CNN, have shown greater success in time series datasets, particularly PV power forecasting. However, most studies use sequential stacking of BiLSTM-CNN in which the first model processes the data first, followed by the second. The sequential stacking results in slower computation and may not be suitable for multistep and real-time forecasting tasks due to their computational burden. These models perform multiple transformations, increasing the possibility of information degradation or loss of important aspects. To overcome these limitations, a parallel processing approach is applied in which both BiLSTM and CNN analyze the input data independently.

Another main problem is that DLMs, including BiLSTM-CNN, treat all time series datasets equally. In contrast, in a real-world scenario, not every timestep adds the same contribution to the final output. For PV power forecasting, this may result in low accuracy during abrupt weather conditions. Some studies address this problem by employing self-attention mechanisms, which assign different weights to each timestep in the fused features of the BiLSTM-CNN. However, self-attention cannot fully detect patterns within datasets and is also less resilient to noisy data. It employs a single head, which allows it to learn only one relation across the entire sequence. This reduces the diversity of the learned attributes due to the model's inability to capture multiple unique dependencies. In real-world datasets, a single learning feature can lead to overfitting due to anomalies. To overcome these limitations, in our work, we employ multi-head attention. The multi-head attention provides a more detailed and richer representation of the sequence. Compared to self-attention, multi-head attention can reduce information loss. Additionally, it can improve the model's capacity to interpret various temporal dependencies, thereby enhancing forecasting accuracy and interpretability.

A further significant challenge arises from the hyperparameter tuning of DLMs. DLMs, including BiLSTM-CNN-attention, are highly sensitive to

hyperparameter settings, such as LSTM units, CNN filters, number of layers, epochs, batch size, learning rate, number of heads, and even the selection of activation function. Without proper hyperparameter settings, these models can lead to lower forecasting accuracy and performance [27]. Additionally, poorly chosen hyperparameters result in longer training time and increased risk of overfitting [28]. MOAs, particularly HHOs, are considered the best choice for hyperparameter tuning of DLMs. Although original HHOs can address multimodal and continuous optimization problems, they are still prone to getting trapped in local optima for multi-dimensional problems. Another problem of HHO is the shift between exploration and exploitation, which is regulated by escape energy. This rigid control may cause exploitation to begin too late, resulting in slow convergence. Additionally, HHO's exploitation phase relies on random jumps, and inappropriate jumps near optimum solutions may result in overshoot. These shortcomings may result in inferior design and prevent the DLMs from attaining their full potential. Thus, to achieve low final fitness values, a better convergence rate, and higher efficiency, an adaptive HHO is proposed in this work. The proposed adaptive HHO improves the original HHO in four phases. Firstly, two chaotic maps are used, one for initializing the hawks' population and another for generating random numbers. Secondly, a nonlinear energy factor is used to maintain a balance between global and local search. Thirdly, an adaptive inertia weight mechanism is introduced to enhance the convergence rate and search stability. Lastly, an adaptive decay factor is used to regulate the steps as the algorithm converges.

The main research problem highlighted in this work can be summarized as follows: ***how can a hybrid DLM combining BiLSTM, CNN, and multi-head attention be tuned using adaptive Harris Hawk Optimization to attain higher forecasting accuracy, improved reliability, with low computational burden on a real-world dataset?*** The proposed research work hypothesizes that an adaptive Harris Hawk Optimization (ADHHO), combined with chaotic maps, a nonlinear energy factor, an adaptive inertia weight, and an adaptive decay factor, may address the shortcomings of traditional HHO and enhance the hyperparameter tuning of DLMs. Thus, the key research problem is to design a robust hybrid model ADHHO-BiLSTM-CNN-MHA that can effectively capture multidimensional dependencies, dynamically focus on

important features in PV datasets using multi-head attention, and adaptively adjust its hyperparameters using ADHHO. The proposed architecture is intended to handle linear and nonlinear patterns, complex relationships, seasonality, and sudden weather fluctuations in real-world datasets. Additionally, the proposed architecture aims to reduce the gap between model precision and computational efficiency by designing a framework that is both intelligent and adaptable for practical and real-time forecasting applications.

1.3 Research Objectives

The primary objective of this work is to design a dual-stream hybrid deep learning model that combines BiLSTM, CNN, and a multi-head attention mechanism optimized by an adaptive Harris Hawk optimization algorithm, for enhanced multistep PV power forecasting. This key objective stems from the shortcomings and drawbacks of conventional forecasting models, which often fail to capture the complex and nonlinear patterns inherent in real-world PV power datasets. Hence, by integrating the short-term local patterns capability of CNN, the long-term temporal dependencies of BiLSTM, the dynamically weight-assigning ability of MHA, and improved optimization capabilities of ADHHO, the proposed model aims to achieve superior forecasting accuracy under rapidly changing situations, such as abrupt changes in weather conditions or other environmental factors.

Overall, the research objective of this study can be summarized as follows,

1. To design and evaluate a dual-stream hybrid model that integrates BiLSTM, CNN, and multi-head attention for capturing local patterns, temporal dependencies, and focuses on the most important part of the input sequence of the real-world PV power datasets.
2. To develop a parallel structure that reduces the dimensionality, the number of trainable parameters for the BiLSTM-CNN model, and also requires less computation, consumes less memory, and is less prone to overfitting.
3. To design a robust framework that is suitable for multistep-ahead PV power forecasting and can provide high accuracy and performance with low computational expense.

4. To implement an adaptive Harris Hawk Optimization algorithm integrating chaotic optimizations, a nonlinear control parameter, adaptive inertia weight, and adaptive decay factor, for accurate and efficient hyperparameter optimization of the proposed dual-stream attention-based hybrid model, improving accuracy and resilience in diverse forecasting conditions.
5. To assess the performance of the proposed ADHHO optimized BiLSTM-CNN-MHA using the real-world PV power dataset acquired from Alice Springs, Australia, and ENEA Italy, and compare the performance against multiple baseline models and optimization techniques.
6. To formulate a generalized and adaptable system suitable for various time-series prediction problems besides PV power forecasting.

1.4 Significance of the Study

The significance of this research lies in its contribution to time-series forecasting by introducing an adaptive Harris Hawk-optimized dual-stream attention-based hybrid model to address the challenges of PV power forecasting. By integrating BiLSTM for detecting long-term temporal dependencies and the CNN for detecting short-term local patterns, the proposed research considers intricate, complex, and sequential attributes of various prediction problems. The multi-head attention, after reducing the dimensionality and hence the number of trainable parameters, further enhances the model's capacity by assigning attention weights to specific data in the feature space, thereby improving comprehension and helping prevent the parallel structure BiLSTM-CNN from being misled by erroneous sequences. The proposed design is robust and adaptable, making it ideal for energy predictions, complex engineering design, deep learning parameter tuning, and various other real-world problems. The experimental findings demonstrate that the proposed model outperforms the baseline models on two real-world datasets. Since the real-world datasets often reflect linear patterns, nonlinear trends, seasonality, sudden variations, and periodic fluctuation, the proposed model can effectively capture these variations, demonstrating its robustness on complex datasets.

A noteworthy advancement in this study is the implementation of an adaptive Harris Hawk Optimization algorithm for tuning the hyperparameters of the dual-stream attention-based hybrid model. Conventional metaheuristic algorithms, including HHO, are prone to getting trapped in local optima for multi-dimensional problems. They are sensitive to parameter settings and require precise parameter adjustments to achieve high-quality results. ADHHO proposes adaptive search strategies that integrate chaotic optimization, a nonlinear control parameter, an adaptive inertia weight, and an adaptive decay factor to enhance convergence and robustness. The experimental findings demonstrate that the proposed ADHHO achieved the lowest fitness values compared to state-of-the-art MOAs on multiple unimodal, multimodal, and fixed dimension optimization problems. The strong exploitation and exploration capabilities support the proposed algorithm in providing high-quality solutions, which are critical in real-life applications. Additionally, the proposed ADHHO not only demonstrated promising potential in addressing complex optimization problems but also improved the scalability and reliability of deep learning models, as validated through various experimental results.

By combining a dual-stream attention-based hybrid model and an adaptive Harris Hawk optimization algorithm, this research offers valuable practical and theoretical contributions. The proposed model provides academics and practitioners a novel, efficient, and robust approach for addressing complex forecasting tasks where the datasets are complex, multifaceted, erratic, and nonlinear. The practical implication of the proposed model is equally important. Accurate forecasting can help utility companies balance supply and demand, providing more dependable approaches for meeting minimum generation and scaling requirements. It promotes the integration of more renewable energy resources and lowers dependence on fossil fuels. Additionally, it reduces operational costs and persuades plant managers to create more accurate bids regarding energy planning and energy management. Ultimately, this research work addresses the current limitations of the forecasting methods, defines a novel benchmark for forecasting accuracy and model's reliability, and opens the opportunity for future research direction.

1.5 Thesis Structure

The dissertation is based on the outcomes of the PhD research work. This dissertation is divided into seven chapters, and its layout is presented in Fig. 1.3 with accompanying publications.

Chapter 1: Introduction discusses the background, problem statements, and research objectives of this work. It outlines how a dual-stream BiLSTM-CNN-MHA model optimized with adaptive Harris Hawk optimization can improve the forecasting accuracy.

Chapter 2: Fundamentals and Related Work presents a detailed overview of the importance and challenges of time series forecasting. It discusses classical and data-driven models, highlighting their main shortcomings in PV power forecasting. It explores the role of deep learning models and explains how the hybrid combination of a deep learning model and multi-head attention can improve the capacity to learn complicated trends, seasonal fluctuations, and long-

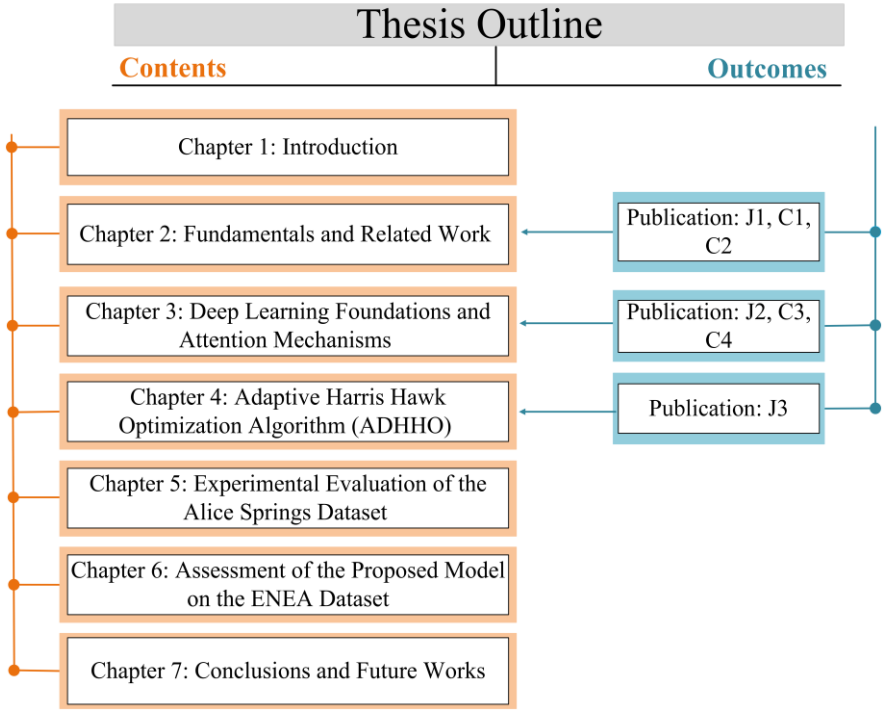


Figure 1.3: Content and outcomes of the thesis

term dependencies in PV power data. This chapter then highlights the importance of hyperparameter tuning using metaheuristic algorithms for improved accuracy and faster convergence. Ultimately, this literature review highlights the research gaps that support the need for an ADHHO optimized dual-stream attention-based hybrid model for PV power forecasting.

Chapter 3: Deep Learning Foundations and Attention Mechanisms outlines the mathematical structure and underlying concepts of the key strategies employed in this work. It explains local feature extraction in CNN, long-term temporal dependencies in BiLSTM, and multi-head attention for dynamic weighting of features. Additionally, it provides the workflow of parallel processing BiLSTM-CNN with multi-head attention mechanisms.

Chapter 4: Adaptive Harris Hawk Optimization Algorithm (ADHHO) discusses the design and implementation of the adaptive Harris Hawk optimization algorithm. The adaptive HHO introduces a four-step improvement in the standard HHO, including chaotic optimizations, a nonlinear control parameter, adaptive inertia weight, and adaptive decay factor. Then, the workflow and pseudocode explaining the operation of the adaptive HHO are presented. Finally, the performance of the proposed adaptive HHO is validated against standard HHO and other advanced algorithms on two benchmark suites: CEC 2014 and CEC 2017.

Chapter 5: Experimental Evaluation of the Alice Springs Dataset provides a research framework, such as a description of the dataset, data cleaning procedure, and assessment methods. It explains PV power output and various weather-related variables employed for analysis, including global horizontal radiation (W/m^2), diffuse horizontal radiation (W/m^2), temperature (°C), relative humidity (%), wind speed (m/sec^2), and wind direction (°). It also discusses data normalization, the sliding window approach, and the data splitting method. Additionally, this chapter describes the proposed model architecture, baseline models, and various performance metrics for analysis and comparison. Finally, the results of the proposed model are analyzed in comparison to the baseline models for 1-step, 3-step, and 6-step ahead forecasting.

Chapter 6: Assessment of the Proposed Model on the ENEA Dataset presents the performance results of the proposed model on the ENEA dataset. Initially, data is prepared by combining energy and static datasets from the ENEA and historical weather data from the Open-Meteo API. Then, the dataset is cleaned, normalized, and split to improve model performance. Similar to Chapter 5, the baseline models and performance metrics for the ENEA dataset are selected and described. Ultimately, the performance of the proposed model is compared to the baseline models, and the results are discussed.

Chapter 7: Conclusions and Future Work summarizes the thesis's conclusions and contributions and provides further recommendations for future study.

2

Fundamentals and Related Work

This chapter provides a detailed description of classical and data-driven models used for solar applications. In addition, the operation and mathematical models for the widely used models are presented.

2.1 Solar Power Forecasting: Concepts and Challenges

Over the past decades, fossil fuels have remained the leading energy source, representing nearly 83% of the world's energy output in 2020 [29]. These resources contribute significantly to greenhouse gas emissions and global warming. They release vast amounts of carbon dioxide, methane, and other hazardous gases that contribute to air pollution and health issues. These resources are non-renewable and depleted at an alarming rate [1]. Once extracted, they cannot be replenished instantly. This limited supply can lead to future energy security issues, particularly as energy demand continues to rise. Solar power may be a viable option for the future energy supply. This energy source can provide an unlimited, sustainable, and cost-effective energy supply. Its operating and maintenance costs are much lower compared to other energy sources. In addition, it can be installed in different sizes, ranging from small rooftop to huge utility-scale plants. Hence, it is more adaptable in addressing both local and regional energy demands. In the last few decades, many countries have increased the power generation from PV systems through various policies and incentive programs. These incentives include capital subsidies, FiTs, net metering, research and development incentives, technology production incentives, and portfolio standards, etc. The incentive schemes have resulted in technological advancements, PV system cost reduction, growing participation of the private

sector, and lower energy costs. From 2010 to 2023, solar PV experienced a 90% decrease in LCOE, becoming 56% less expensive than the cheapest fossil fuel [30]. The lower overall cost of PV installations has dramatically increased its installed capacity compared to both renewable and non-renewable energy sources. According to the IEA, the accumulated PV power capacity reached 2.2 TW in 2024, accounting for 10% of the global energy consumption [31]. Fig. 2.1 illustrates the cumulative PV installations from 2005 to 2024. As shown in the figure, in 2024 alone, almost 602 GW of PV systems were installed worldwide, representing more than 75% of the new renewable energy capacity. The overview of one-year installed capacity worldwide is shown in Fig. 2.2. Moreover, the IEA predicted that solar PV power will surpass wind and hydropower generation by

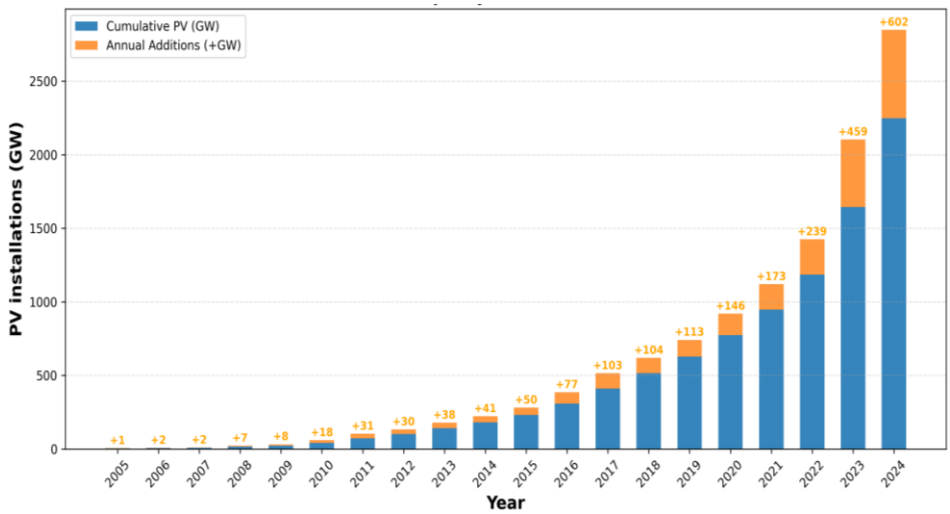


Figure 2.1: Worldwide solar power cumulative installations and yearly increment. The data for the bar chart is obtained from the International Energy Agency [31]

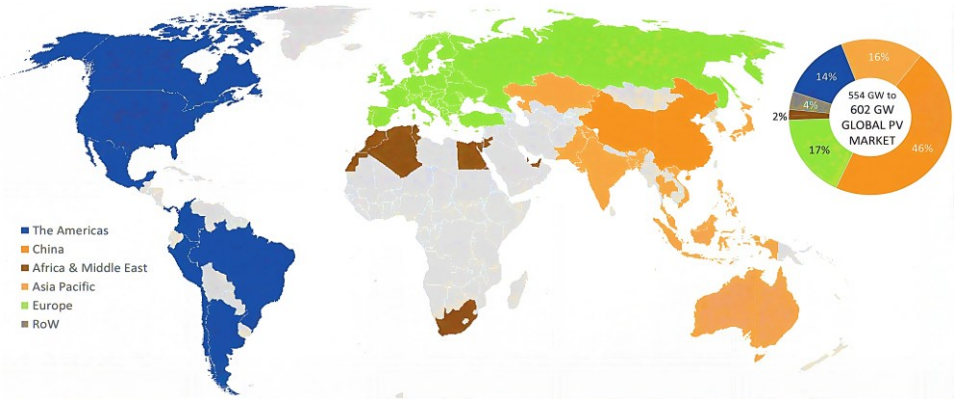


Figure 2.2: Overview of the worldwide solar power installation capacity in 2024 alone. Sourced by: International Energy Agency [31]

2027 and 2029, respectively [32]. Bloomberg estimated that by 2050, PV power will supply 50% of the world's energy needs, with an accumulated capacity of 14,000 GW [33]. The staggering rise in PV power production presents a challenge regarding its extensive integration into the power systems. For the reliable and safe operation of utility grids, the total power injected into them must be balanced by the total consumption. However, PV output power is inherently erratic and unstable due to its dependence on the weather conditions. The unstable nature of PV power can significantly impact the operation and stability of utility grids. The output power can drop from the rated value to a minimum value in minutes or seconds. Such instant power reductions can lead to voltage and frequency instability. The impacts can be more critical when the PV power represents a large portion of the total generated power [34]. In high PV-penetrated power systems, fluctuations in PV power can lead to voltage changes, power outages, infrastructure damage, and even catastrophic failures [5].

Hence, to avoid these negative impacts on the entire power system, precise PV power forecasting is essential. Accurate forecasting can help utility companies balance the supply and demand more effectively. It provides more dependable and efficient approaches for fulfilling the minimum generation and scaling requirements. It promotes the integration of more renewable energy resources, lowering dependence on fossil fuels. Additionally, it can enhance the system's reliability and reduce operational costs. Precise prediction can persuade plant managers to create accurate bids and plan maintenance downtime effectively [35]. Despite its numerous advantages, attaining accurate forecasting poses significant obstacles. The main obstacle is the availability of high-quality data for training forecasting models. Sometimes, the data is insufficient to train the forecasting models, while in other cases, the data contains missing values and outliers. Missing values and outliers can significantly reduce the model's performance and provide inaccurate forecasts. Missing values can disrupt temporal patterns and trends in datasets, affecting the learning process. Outliers can cause poor generalization, inaccurate gradient updates, and result in convergence issues. Similarly, various other factors can greatly reduce the forecasting accuracy. These include the forecasting horizons, sudden changes in weather-related variables, improper hyperparameters, inappropriate model structure, and performance indicators [3]. Inaccurate forecasting results can cause financial losses, ineffective

operations, and safety hazards. In some power markets, inaccuracies can result in penalties if the difference between the forecast and actual power production exceeds a specific tolerance range. For instance, the authors estimated the cost of forecasting errors for a solar thermal power plant in Spain [36]. Forecasting error is divided into groups ranging from 10% to 100%. According to the findings, penalties per MWh increase from 10-20% to 60-70% for relative prediction errors while remaining relatively constant at about 6 €/MWh for larger forecast errors. An overestimated PV power would require a different set of generators to respond than initially planned. In contrast, underestimating PV power output can lead to transmission congestion, curtailment, and require an adjustment to the scheduled generator configuration.

2.2 Classical and Data-Driven Prediction Models

Accurate PV power forecasting remains a considerable challenge, owing to various contributing factors [37]. Weather conditions, geographical factors, forecasting horizons, and technical aspects all contribute to the complexity of forecasting. Over the years, multiple forecasting models, including Physical, Statistical, and hybrid, have been proposed [38]. The application of these models is highly reliant on the natural setting of the region. Selecting the optimal models that accurately reflect the local climate and specifications of the PV systems is crucial. Choosing the best model can minimize the financial and safety uncertainties. This section provides a concise overview of the various physical, statistical, and machine learning models used for solar system applications.

2.2.1 Physical Models

Physical models use scientific concepts and physical laws to predict future outcomes. These models are designed using precise site data (coordinates, altitude) and weather-related variables (irradiance, temperature, humidity). Additionally, creating these models also required panel orientations (azimuth and tilt), panel descriptions (rated values and efficiency), and inverter technical details. These models are reliable and transparent because they depend on physical laws and exact site information. They do not require any previous or

historical PV power data and can be used for power estimation of a new power plant [39]. Additionally, physical models can be employed when there is insufficient PV power data due to power interruptions and blackouts. PV plant owners can use these models when they have complete design information about the plant. In the literature, multiple physical models are proposed, including sky-image models, clear-sky models, NWP models, satellite models, etc. Fig. 2.3 presents the five widely used physical models for solar energy applications. The number of publications is taken from the Scopus database between 2018 and 2024. The figure illustrates that NWP, clear-sky, and satellite-based models are commonly employed for solar system applications due to their intrinsic properties. Hence, we present the advantages and disadvantages of these three models in Tab. 2.1. It is essential to note that physical models do not directly forecast PV power; instead, they predict solar irradiance or other important weather variables. Then, the solar irradiance or other weather variables can be used to forecast PV power. The detailed explanation of these models is presented below.

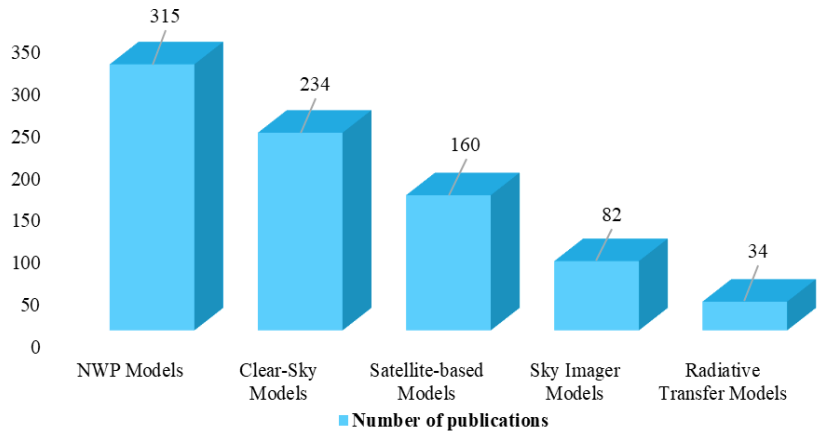


Figure 2.3: Number of publications on physical models (source: Data are obtained from the Scopus database from 2018 to 2024)

NWP Models: These models are used to forecast important weather-related variables such as irradiance, temperature, humidity, and others. These models are considered the most precise for solar irradiance prediction when the forecasting horizon exceeds 5 hours [40]. Initially, these models were used to predict rainfall, storms, and hurricanes. Recently, they have been employed for solar irradiance

forecasting ahead of several hours [41]. WRF-solar is a well-known NWP model, first implemented in 2015 for solar irradiance forecasting [42]. It is the early mesoscale NWP model that was designed to provide precise forecasts of the variables essential for solar energy. It provides very short-interval data for all components of irradiance, which helps mimic sudden variations in PV power generation. In [42], WRF-solar and the global forecasting system are compared for solar irradiance forecasts. The high-resolution WRF-solar produces the lowest relative MAE of 23.14% compared to the coarse domain (24.51%) and the global forecasting system (25.75%). In [43], two NWP models, the European Centre for Medium-Range Weather Forecasts (global) and AROME, are compared for solar irradiance and power forecasting. The data for analysis are taken from 32 radiometry stations and 94 solar power plants. The findings indicate that AROME's high resolution allows it to capture local GHI variations more effectively. On the other hand, the global NWP model gives a more uniform forecast and can outperform AROME on point-based forecasting.

Satellite-based Models: Some studies indicated that satellite-derived cloud motion can attain better results compared to NWP models for 4-5hours ahead forecasting [41]. These models offer a better spatial resolution, as shown in Fig. 2.4. These models employ geometric corrections to accurately predict shadows.

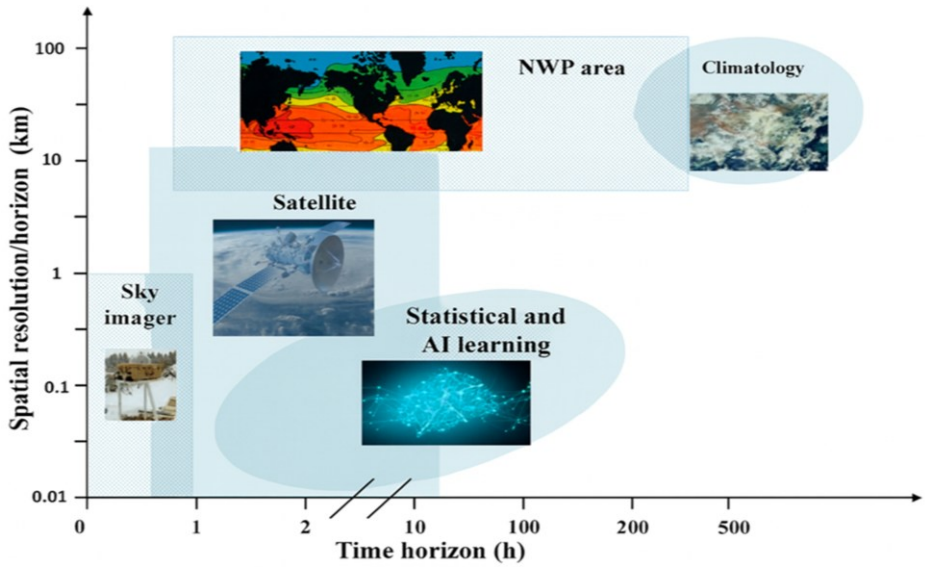


Figure 2.4: Spatial resolution Vs. time horizon for different physical modes[44]

These models first calculate the reflected solar irradiance from the Earth's surface. Then, a minimum and maximum reflectance are defined to normalize the actual reflectance values. Next, the CI is calculated using the equation below,

$$CI = \frac{\rho - \rho_{cloudy}}{\rho_{clear} - \rho_{cloudy}} \tag{2.1}$$

Where ρ, ρ_{clear}, and ρ_{cloudy} indicate the current, minimum, and maximum reflectance. satellite reflectance. The CI is multiplied by the clear-sky Irradiance (G_{CS}) to estimate the all-sky irradiance (G) as,

$$G = G_{CS} \, . \, CI \tag{2.2}$$

The first satellite-based model was used by J. D. Tarpley in 1979 to measure daily insolation [45]. His model achieved a 10% error, indicating a reasonable precision for PV power and agricultural use. Over the years, satellite models have been widely researched for solar applications. In 2016, FARMS was developed by NREL for GHI estimation under various sky scenarios [46]. This model achieved higher accuracy than a two-stream approximation and can be used for real-time applications. In 2022, Tournadre et al. proposed the Heliosat-V for solar irradiance estimation at Earth's surface [47]. The proposed model leverages RTM to predict overcast and clear sky satellite landscapes of the Earth's reflectance. The Heliosat-V model achieved a bias of 20 W/m^2, an RMSE value of 93 W/m^2, and an R value of 0.948.

Clear-Sky Models: These models are also widely used for solar power applications. These models forecast solar irradiance during clear sky conditions; hence, they define the upper limit for other forecasting models, such as satellite models. The first clear-sky model was proposed by NREL in 1981, which is well-known as the Bird Model [48]. This model mainly relies on algebraic formulas, which makes it suitable for simple applications. Over the years, various clear-sky models have been proposed, including Ineichen–Perez, SOLIS, REST2, McClear, and others. These models are normally classified based on the input requirements. Almost all clear-sky models require the solar zenith angle (θ) and air mass (m) for measuring solar geometry for a particular location [49]. Some models also require precise information on various atmospheric variables, such as aerosols,

atmospheric pressure, ozone, carbon dioxide, etc. The air mass (m) is normally calculated using the following equation [49],

$$m = \frac{1-e^{-4}.ele^{sl}}{cos\,\theta+0.50572\cdot(96.07995-\theta)^{-1.6364}} \tag{2.3}$$

$$\theta = 90° - ele^{sl} \tag{2.4}$$

Where ele^{sl} represents the elevation above sea level. Among the atmospheric variables, aerosol extinction is the main contributor to the atmospheric attenuation. The aerosol optical depth is normally used to measure the aerosol extinction.

$$AOD^{\lambda} = \beta\lambda^{-\alpha} \tag{2.5}$$

Where β, α, and λ represent the Ångström turbidity, Ångström exponent, and solar radiation wavelength, respectively.

Types	Advantages	Disadvantages
NWP Models	• Gives extended prediction horizons (day-multiday ahead) • Useful for broad regions • Capture large-scale weather events	• Lower spatial resolution (global NWP >10km) • Lower temporal resolution (1 to 3 h) • Cannot capture small-scale clouds such as Cumulus • Unable to predict accurate cloud shadow locations. • Computationally expensive and high processing times
Satellite-based Models	• Better spatial resolution (1-5km) • Capture precisely small-scale clouds • Excellent for short-term forecasting	• Assume a static motion of the clouds into the near future • Unable to capture dynamic cloud conditions such as sudden cloud formation or dissipation • Accuracy decreases as the forecasting horizon increases
Clear-Sky Models	• Provides a baseline or reference for multiple forecasting models • Minimum input data requirements • Computationally less expensive compared to other models	• Limited to clear skies only and does not consider cloud cover • Not appropriate for actual forecasting • Require precise estimations of aerosol optical depth, precipitable water vapor, and other variables, which are difficult to measure accurately

Table 2.1: Advantages and disadvantages of various physical models[40], [41], [50]

2.2.2 Classical Statistical Models

Classical statistical models are suitable for short-term PV power forecasting. These models used previously collected data to capture patterns and correlations, thereby forecasting solar irradiance or PV power output. Compared to physical models, these models do not require complex atmospheric physics; instead, they depend on high-quality past data. Hence, these models are computationally less expensive. Typically, statistical models employ statistical theories and make specific assumptions about the data's layout. Based on underlying assumptions, they predict the future values of the target variable. When these assumptions coincide with the true features of the dataset, these models are more likely to make accurate predictions. In contrast, if the assumptions are inaccurate or overlooked, then statistical models may generate false or deceptive predictions. The reliability of the assumptions has a significant effect on the accuracy of the forecasting. Statistical models are normally transparent and interpretable [9]. This makes them particularly valuable in fields where transparency and interpretability are important. In the literature, various advanced statistical models are employed for solar applications. However, in this section, we discussed commonly used statistical models for solar power forecasting. Tab. 2.2 also provides the detailed advantages and disadvantages of these statistical models for forecasting tasks.

Linear Regression: A popular example of the statistical model is the least squares linear regression, which was introduced in the early 19th century. Mathematically, it can be expressed as,

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \qquad i = \{1,2,3 \dots N\} \tag{2.6}$$

Where y_i, x_i, β_0, β_1, and ε represent the dependent variable, predictor variables, intercept, slope coefficient, and error term, respectively. The linear regression estimates the unknown parameters β_0, β_1 by minimizing the sum of squared differences between the actual and predicted values.

$$S = \sum_{i=1}^{N} [y_i - (\hat{\beta}_0 - \hat{\beta}_1 x_i)]^2 \tag{2.7}$$

$$\hat{\beta}_1 = \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}) / \sum_{i=1}^{N} (x_i - \bar{x})^2 \tag{2.8}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{2.9}$$

Since linear regression is a type of statistical model, it is normally based on various assumptions. These include linearity, homoscedasticity, weak exogeneity, etc. Violating these assumptions may result in skewed forecasts, high average errors, and incorrect conclusions.

Principal Component Analysis: This model is potentially the most widely used multivariate statistical model, having applications in literally all scientific domains, including PV power forecasting. PCA was proposed by K. Pearson in 1901 to detect and extract trends in the dataset [51]. Nonetheless, this approach received wide recognition in the mid-20th century, during the "Quantitative Revolution". The delayed acceptance of this method was due to the high computational burden of matrix decompositions [52]. However, with the emergence of digital computers, this method has gained great acceptance in many fields. Currently, PCA has been mainly used for dimensionality reduction without compromising key information. This feature can eliminate irrelevant data, allowing feature extraction and improving PV power forecasting accuracy. This is because reducing dimensionality can improve the data handling capability of the forecasting model. Additionally, considering only relevant features can reduce the likelihood of overfitting. Consider a dataset $X \epsilon \mathbb{R}^{n \times p}$, then the center of data can be expressed as [53],

$$\tilde{X} = X - \bar{x}, \quad \bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{2.10}$$

The covariance matrix C_M and the eigenvector a_k can be computed,

$$C_M = \frac{1}{n}\tilde{X}.\tilde{X}^T \tag{2.11}$$

$$C_M a_k = \lambda_k a_k \tag{2.12}$$

Where n, $\bar{x}$, and λ_k represent the number of samples, column means, and the kth eigenvalue, respectively. Next, the eigenvalues are sorted as $\lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots \geq \lambda_r$ to select the top r vectors for reducing the dimensionality.

$$A_k = [a_1\ a_2\ a_3 \ldots a_r] \tag{2.13}$$

Now, the original centered dataset is transformed into new coordinates as,

$$S = \tilde{X} A_k \tag{2.14}$$

Where S represents the principal component scores.

Exponential Smoothing: Exponential smoothing is a viable prediction method that employs weighted averages of previous data, and the weight decreases exponentially with time. In simpler terms, the most recent data receives precedence over the older data. This model has three main variants: 1) simple exponential smoothing, 2) double exponential smoothing, and 3) triple exponential smoothing. Simple exponential smoothing is used for datasets that have a stable mean or exhibit no trends. Double exponential smoothing, also known as Holt's Method, can handle datasets with trends. Triple exponential smoothing, also known as the Holt-Winters Method, can be useful for datasets that have trends and seasonal variations. Triple exponential smoothing is further classified into two types, multiplicative and additive seasonal models. Additive seasonality assumes constant seasonal trends, whereas multiplicative seasonality assumes that seasonal fluctuations are not constant but vary with trends [54]. Mathematically, triple exponential smoothing with additive seasonality can be expressed as [55],

$$l_t = \alpha(y_t - s_{t-l}) + (1-\alpha)(l_{t-1} + d_{t-1}) \tag{2.15}$$

$$d_t = \beta(l_t - l_{t-1}) + (1-\beta)(d_{t-1}) \tag{2.16}$$

$$s_t = \gamma(y_{t-1} - l_t) + (1-\gamma)(s_{t-l}) \tag{2.17}$$

The forecast can be determined using the level (l_t), trend (d_t), and seasonality (s_t) factors, as shown below,

$$f_{t+h} = l_t + d_{th} + s_{t-l+h}) \tag{2.18}$$

Where α, β, γ represent level, trend, and seasonal smoothing constant, respectively. l is the seasonality length and h represents the forecast horizon.

Autoregressive Integrated Moving Average: ARIMA is the most widely used class of parametric time series. It is an efficient statistical model for solar irradiance and PV power forecasting. ARIMA is made of three important parts: 1) the autoregressive part, 2) the integrated part, and 3) the moving average part. The autoregressive model predicts the future value (y_t) by assuming that the

current value of the time series is a linear combination of its previous values. The pth order autoregressive model can be expressed as [56],

$$y_t = \sum_{i=1}^{p} \varphi_i y_{t-i} + e_t \tag{2.19}$$

Where $\varphi_1, \varphi_2, \dots, \varphi_p$ represent the unknown coefficients, and e_t represent the white noise. The integrated part converts a non-stationary time series into a stationary one. This is because non-stationary time series exhibit patterns and seasonal fluctuations that can defy the assumptions of the autoregressive and moving average models. Hence, differencing is applied to normalize the series. The integrated part of order d can be expressed as [54],

$$y_t = (1 - B)^d y_t \tag{2.20}$$

Where B is the lag operator. The moving average part, also referred to as error feedback terms, measures the impact of previous forecast errors on the current value. This part updates the forecasting according to the size and direction of past errors. The qth order moving average can be expressed as,

$$y_t = e_t + \sum_{i=1}^{q} \vartheta_i e_{t-i} \tag{2.21}$$

Where ϑ_i represents the moving average coefficients. Combining the equations of all three parts gives the full ARIMA(p, d,q) equation as,

$$\nabla^d y_t = \varphi_1 \nabla^d y_{t-1} + \varphi_2 \nabla^d y_{t-2} + \cdots + \varphi_p \nabla^d y_{t-p} + e_t + \vartheta_1 e_{t-1} + \vartheta_2 e_{t-2} + \cdots + \vartheta_q e_{t-q} \tag{2.22}$$

Types	Advantages	Disadvantages
LR Models	• Fast and simple to comprehend • Works excellently if PV power has a linear relationship with weather variables, such as irradiance or temperature • Computationally less expensive • Widely used multivariate statistical model	• Unable to capture nonlinear relationships, such as cloud-induced variations, weather fluctuations, and seasonality changes • Sensitive to outliers • Variations may affect the PV power forecasts
PCA Models	• Reduces dimensionality without compromising key information • Eliminate irrelevant data, allowing feature extraction • Consider only relevant features and reduce the probability of overfitting	• Consider linear correlation between features • May not detect nonlinear relationships in solar datasets • Select features that reflect the maximum variance, which might not always be the best predictors of PV power output

Exponential Smoothing Models	• Fast and simple to comprehend • Most recent data receives precedence over the older data • Have high accuracy and durability • Particularly effective for short-term forecasting	• Performs poorly when datasets exhibit complex structures such as patterns and seasonality • The initialization problem reduces the accuracy • Not suitable for medium and long-term forecasting • The performance is highly dependent on the selection of smoothing parameters
ARIMA Models	• Specially designed for time series forecasting • Provide accurate results on stationary time series datasets •	• Getting a stationary dataset can be challenging owing to its unpredictable nature • Struggle to capture sudden variation in time series datasets • Provide low accuracy for long-term forecasting

Table 2.2: Advantages and disadvantages of various statistical models[10], [53], [57], [58]

2.2.3 Machine Learning Models

Machine learning is a subclass of AI that enables computers to learn patterns and trends from datasets without explicit programming. It entails employing models for data analysis, pattern recognition, and predictions. These models used labeled or unlabeled datasets for training. Once trained on these datasets, they can be used for either classification tasks or prediction tasks. Machine learning has a long tradition, dating back to the mid-1900s. In 1950, English mathematician A. M. Turing first introduced the concept of “Can machines think” in his paper [59]. A.M. Turing proposed the Imitation Game to check if a machine could demonstrate the same intelligent behavior as humans. In 1957, Rosenblatt introduced the perceptron [60]. The perceptron was the first machine learning model that was used for binary classification tasks. However, over time, various advanced MLMs have been proposed for solving complex regression and classification tasks. These MLMs are classified into four groups: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Among the four types, supervised learning is widely used for various real-world applications, including PV power forecasting [61]. These models utilized labeled datasets for predicting the target variable. In labeled datasets, each input has a corresponding output; thus, the model learns an exact relationship between inputs and outputs. This section outlines the fundamental principles of several widely used MLMs for forecasting tasks, and Tab. 2.3 provides further explanations of the advantages and disadvantages.

k-Nearest Neighbors: k-NN was the first non-parametric MLM used for classification tasks. E. Fix and J. Hodges initially proposed the concept of k-NN, and later expanded it by T. Cover [62]. Today, this model is commonly used in data mining, machine industries, robotics, forecasting, and sensor networks. K-NN normally works on the principle of probability of resemblance. It suggests that data points that are similar can cluster together in space. Hence, k-NN predicts the new instance based on its proximity to the existing training data. For time series datasets, this proximity or closeness can be determined using the dynamic time warping method. The summarized version of standard k-NN regression can be presented as,

$$\hat{y} = \frac{1}{k}\sum_{i \in N_k(\hat{x})} y_i \tag{2.23}$$

Where $\hat{y}$, k, and $N_k(\hat{x})$ represent the predicted value, the number of neighbors, and the set of indices of k closet of x. The standard k-NN gives the same importance to all neighbors, irrespective of their distance from x. In contrast, another version, weighted k-NN, gives more weight to the closer neighbors.

Decision Trees: Decision Trees were initially proposed by Morgan and Sonquist in 1963. They normally take the shape of a flowchart, wherein each node indicates a logic test, and each leaf indicates a possible forecast. The primary benefit of these models is that the method of inference is completely transparent [63]. The path from root to leaf is simple to track and understandable. Due to their simplicity, robustness, and transparency, these models are widely used in various fields, including data analysis, text classification, library systems, healthcare, medical diagnosis, stock market analysis, and forecasting. Suppose S represents the complete dataset, with n unique class labels. s_j represents the number of samples in the class C_j. The entropy can be calculated as,

$$I(s_1, s_2, \dots, s_n) = -\sum_{j=1}^{n} p_j log_2 p_j \tag{2.24}$$

Where $p_j = {}^{s_j}/_{S}$ represents the probability of a sample corresponding to a class C_j. Now suppose attribute A has m unique values: $A = \{a_1, a_2, \dots, a_m\}$, resulting in m subsets: $S = \{S_1, S_2, \dots, S_m\}$, then the anticipated entropy can be calculated as,

$$E_s(A) = \sum_{k=1}^{m} \frac{s_{1k} + s_{2k} + \dots + s_{mk}}{S} I(s_{1k}, s_{2k}, \dots, s_{mk}) \tag{2.25}$$

The information gain of A is measured as,

$$Gain(A) = I(s_1, s_2, \dots, s_n) - E_s(A) \tag{2.26}$$

Hence, the $Gain(A)$ indicates the reduction in the amount of uncertainty using the attribute A for splitting the data.

Support Vector Machine: SVM, developed by V. N. Vapnik, is undoubtedly one of the most widely used machine learning models. The classic SVM works on the principle of finding an optimal hyperplane that can easily separate data points of various classes. The optimal hyperplane is the one that has the maximum margin (distance between the hyperplane and support vectors) [64]. Sometimes, the hyperplane cannot separate the datasets into various classes. In such cases, SVM employs a non-parametric kernel function. A prominent kernel function is the RBF, which can model complex nonlinear relationships. The RBF transformed the input dataset into a high-dimensional feature space, enabling SVM to create a linear separation. In this way, SVM can address datasets that are not linearly separable. SVM is commonly used for classification tasks, though it can also be applied to regression tasks. For regression tasks, including PV power forecasting, it is referred to as SVR, an extension of SVM. SVR has many comparable features with SVM, such as kernel methods for nonlinear mappings and margin maximization. However, SVR employs $f(x)$ to estimate the desired value y_i with a tolerance ϵ, rather than determining the hyperplane as [65],

$$f(x) = \sum_{j=1}^{m}(\alpha_j - \grave{\alpha}_j)k(x, x_j) + b \tag{2.27}$$

Where α_j, $\grave{\alpha}_j$ represent the weight learned, $k(x, x_j)$ represents the kernel function, and b is the bias term. The RBF kernel is expressed as,

$$k(x, x_j) = \exp\left[-\frac{1}{\sigma^2}\|x - x_j\|^2\right] \tag{2.28}$$

Where σ is the RBF kernel parameter.

Types	Advantages	Disadvantages
k-Nearest Neighbors Models	• Simple to comprehend and intuitive • Easy to implement • Perform well when the features are carefully selected	• Limited ability to address complex information in a large dataset • Presents significant performance issues, particularly in selecting the value of k • Highly sensitive to the existence of non-relevant variables

		• Slow when the number of samples in training is increased
Decision Trees Models	• Reliable and simple to understand • May handle categorical and numerical variables to predict • Faster training performance • Can model a nonlinear relation between the actual and predicted attributes	• Very prone to overfitting • May struggle to model temporal dependencies • Require complete datasets and cannot address missing feature values • Struggle in categorizing several output classes • Not suitable for noisy datasets due to their sensitivity to errors
SVM/SVR Models	• Can address datasets that are not linearly separable • Kernel function adds nonlinearities, enhancing its forecasting capabilities • Kernel function presents efficient and useful outcomes while modeling different relationships	• Fail to perform well on large datasets due to their training complexity • Possess limited capacity for modelling temporal dependencies • Sensitive to hyperparameters (ϵ, kernel width), and inaccurate tuning may result in overfitting

Table 2.3: Advantages and disadvantages of various machine learning models[63], [66]

2.3 Overview of Deep Learning Architectures

Deep learning is a family of ML, as shown in Fig. 2.5, that uses ANNs with multiple layers. They can outperform traditional MLMs, including SVM, LR, RF, and others. Traditional ML models possess a shallow structure that limits their learning ability. They struggle to generalize, particularly on high-dimensional or noisy datasets. In contrast, DLMs possess many hidden layers and can learn complex and hierarchical patterns. These models can handle multivariate and high-dimensional inputs. These features become critical when dealing with the dynamic nature of environmental conditions. Hence, DLMs have become increasingly prominent in many fields, including object detection, healthcare, machine translation, NLP, and forecasting. In the literature, multiple DLMs have been proposed, including CNN, RNN, GAN, Autoencoders, Transformer, and hybrid models. Additionally, two variants of RNN, LSTM and GRU, are also used due to their unique structure and exceptional performance. In this section, we will discuss the operation, properties, and structure of some of the widely used DLMs.

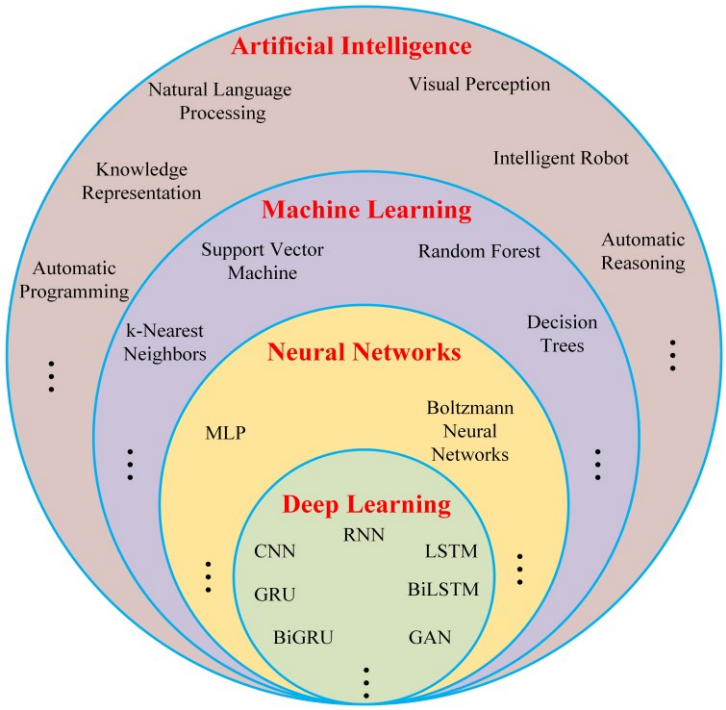


Figure 2.5: Relationship between Machine learning and Deep Learning

Convolutional Neural Network: CNN is specifically designed to analyze datasets with a grid-like architecture. CNN's main strength is its capacity to reduce the total number of parameters, allowing scholars to explore more complicated models [67]. The input layer of CNN receives the raw data and passes it to the convolutional layer. This layer employs a kernel that slides over the input datasets, producing feature maps that illustrate distinct trends as [68],

$$x_n^s = f\left(\sum_{r \in M_n} x_r^{s-1} * k_{rn}^s + b_n^s\right) \tag{2.32}$$

Where x_n^s, f, s, k, and b represent the nth output feature map, activation function, number of layers, the convolution kernel, and the bias. The output of the convolutional layer is passed to the pooling layer, which performs a down-sampling operation that reduces the dimensions of the feature maps [69].

$$x_r^s = f(B_r^s * dow(x_r^{s-1}) + b_r^s) \tag{2.33}$$

Where dow, x_r^{s-1}, denote the pooling function and the feature map of the $s-1^{\text{th}}$ layer. Finally, outputs of the pooling layer are passed to a fully connected layer that represents the predicted results.

$$y^s = f(w^s * y^{s-1} + b^s) \tag{2.34}$$

Where y^s, w^s, and b^s denote the final output vector, the weight, and the bias.

Recurrent Neural Network: RNN was initially proposed in the 1980s to address sequential or time-series datasets. RNN possesses a form of memory that enables it to store information from previous time steps. This feature makes it ideal for applications requiring text, audio, or time-series datasets. A high-dimensional hidden state and nonlinear development provide the RNN's hidden state with remarkable demonstrating potential, allowing it to integrate information over multiple timesteps and present accurate predictions [70]. Let us consider a standard RNN receiving an input vector sequence $(x_1, x_2, \dots, x_n)$ and estimates the hidden states (h_n) and outputs (y_n) using the equations below [71].

$$h_n = f_{hh}(x_n w_{hx} + h_{n-1} w_{hh} + b_{hh}) \tag{2.35}$$

$$y_n = f_{yy}(h_n w_{yh} + b_{yy}) \tag{2.36}$$

f_{hh} and f_{yy} denote activation functions, b_{hh} and b_{yy} denote bases, and w denotes the weight matrix between the corresponding layers.

Long Short-Term Memory Neural Network: LSTM is an advanced variant of the RNN that was especially designed to overcome RNN's vanishing gradient issues [72]. Its structure is the same as RNNs, with the exception that memory cells—specialized units—replace the RNN hidden layer. The memory cells store and maintain information, and the three gates—input, output, and forget—control information flow. Initially, the forget gate analyses both past state information and new data using the sigmoid function (σ) to determine which data should be deleted. Then, the input gate uses the σ and hyperbolic tangent functions($tanh$) functions to decide the new data that needs to be retained in the cell state. Subsequently, the cell state is updated using the information obtained from the previous two gates. Finally, the output gate decides the next hidden state. The overall LSTM unit's process can be explained by the following equations [73].

Forget gate: $$f_t = \sigma_f(w_f . [h_{t-1}, x_t] + b) \tag{2.37}$$

Input gate: $$i_t = \sigma_i(w_i . [h_{t-1}, x_t] + b) \tag{2.38}$$

Update vector: $N_t = tanh(w_c . [h_{t-1}, x_t] + b)$ (2.39)

Cell state: $C_t = C_{t-1} * f_t + N_t * i_t$ (2.40)

Output gate: $O_t = \sigma(w_o[h_{t-1}, X_t] + b))$ (2.41)

Hidden state: $h_t = O_t * tanh(C_t)$ (2.42)

Here, x_t, w, and b represent the input vectors, weights, and biases, respectively.

Gated Recurrent Unit: GRUs, proposed in 2014, were also developed to overcome long-term dependency issues. Like an LSTM, a GRU comprises gating units that regulate the flow of information; however, it does not have separate memory cells [74]. The gate structures of a GRU are simpler than those of an LSTM. The update gate in a GRU combines the functions of both the forget gate and input gate in an LSTM. The update gate (u_t) decides which information from the previous state should be moved to the current state, whereas the reset gate (r_t) decides whether new information will be added to the prior state. Next, the candidate's hidden state ($\acute{h_t}$) is calculated using the ($tanh$), and the final hidden state (h_t) is then determined using linear interpolation. Mathematically, the overall process can be explained by the following equations [75].

$$\begin{cases} r_t = \sigma(w_r \cdot [h_{t-1}, x_t] + b) & (2.43) \\ u_t = \sigma(w_u \cdot [h_{t-1}, x_t] + b) & (2.44) \\ \acute{h_t} = tanh(w_{\acute{h}} \cdot [h_{t-1}, x_t] + b) & (2.45) \\ h_t = (1 - u_t) * h_{t-1} + u_t * \acute{h_t} & (2.46) \end{cases}$$

Where w and b represent the weights and biases, respectively.

2.4 Deep Learning Models for Time Series Forecasting

Recently, the application of deep learning models for time series forecasting has increased due to the rapid growth of artificial intelligence. Their data-driven approaches greatly improve the extraction and analysis of critical information

from large datasets. These models can handle complex, nonlinear problems; thus, they are useful for forecasting, modelling, and optimization [3]. For instance, RNNs and their variants, such as LSTM, GRU, BiLSTM, and BiRNN, provide a notable advancement in managing time-series data, which is essential for forecasting that is inherently temporal. These are especially proposed to capture sequential dependencies [3]. Compared to conventional neural networks, RNNs maintain hidden states that contain information from past timesteps. This allows the RNN to capture temporal patterns in which past data points impact future predictions [70]. Additionally, they can capture highly nonlinear relationships that classical models such as ARIMA may not detect [76]. Along with RNNs, CNNs and their variants are also widely researched for time series prediction due to their global sharing and local connection capabilities, which may drastically decrease model training parameters and time. These models may sometimes outperform LSTM and GRU for time series forecasting. For instance, a temporal-CNN (TCNN) is developed to analyze time series data and estimate PV power production [77]. The proposed TCNN's performance is compared to that of FFNN, RNN, LSTM, and GRU. The TCNN achieved the lowest MAE values of 70.015 and 0.510 as well as the lowest RMSE values of 98.118 and 0.721 for both the University of Queensland and the Sanyo datasets. Despite the benefits of DLMs over classical models, these models still have limitations in detecting various complex patterns in contemporary datasets. Since actual datasets, such as PV power output, may include a combination of linear patterns, complex relationships, noise, and seasonality, single models may not be enough to capture these factors. For instance, actual PV power output is notably multifaceted and impacted by multiple and unpredictable factors [37]. Weather variables, seasonal effects, and sudden fluctuations can significantly affect the PV power output [78]. Single models may struggle to mimic these nonlinearities, seasonality, and sudden shifts [17]. Hence, several research studies have combined multiple DLMs to increase pattern recognition capabilities, improve forecasting accuracy, and convergence rate. The main advantage of the hybrid model is that it can combine the key features of different models. These models can then capture various data features with higher forecasting accuracy than single models. Hence, these models have gained significant attention in solar power and irradiance forecasting.

In the literature, hybrid models are proposed by integrating multiple single models, such as LSTM, CNN, RNN, MLP, GRU, Autoencoder, and Transformer. CNN-LSTM is a commonly used hybrid model for solar power and irradiance forecasting [3]. This hybrid model combines the features of both CNN and LSTM in a single model [14]. CNNs excel in examining and interpreting spatial features, whereas LSTMs are superior for capturing temporal features. Combining these two models can improve the forecasting accuracy compared to both standalone CCN and LSTM models [79]. Some studies suggest that changing the sequence of the model in CNN-LSTM can significantly enhance forecasting accuracy. For instance, compared to CNN-LSTM, LSTM-CNN can perform better on time series datasets [14]. Both CNN and LSTM capture different aspects of the input datasets, and for PV power datasets, LSTM-CNN has an edge over CNN-LSTM. Meteorological variables, seasonal shifts, and nonlinear trends are various temporal factors that can significantly influence PV power output. Introducing LSTM layers in the beginning allows the models to capture these factors effectively. Once LSTM layers extract these factors, CNNs can be used as a feature enhancer. CNN layers examine the LSTM outputs for detecting local patterns and sudden shifts. Hence, LSTM-CNN can be a better choice for PV power forecasting than CNN-LSTM [80]. However, BiLSTM can better detect long-term dependencies than LSTM [81]. Compared to LSTM, BiLSTM simultaneously processes the sequence in both forward and backward directions. The two-way perspective of the dataset offers deeper contextual representations. Thus, this model may provide better accuracy and reliability compared to the unidirectional counterpart (LSTM). Multiple studies suggested that the hybrid combination of BiLSTM and CNN is a better choice for PV power forecasting [15], [18]. In [18], BiLSTM-CNN is evaluated for PV power forecasting using the BP solar plant datasets. The findings of the proposed model are compared to single models (CNN, LSTM, BiLSTM) and hybrid models (LSTM-CNN, BiLSTM-CNN). The experimental results show that BiLSTM-CNN achieved better scores on most of the performance metrics on both sunny and cloudy days. BiLSTM-CNN achieved a low average RMSE of 77.73W and a high Coefficient of Determination of 0.9966. Hence, BiLSTM-CNN is a vital option for solar power forecasting [3].

2.5 Attention mechanisms and their applications in forecasting

Attention mechanisms have appeared as a key advancement in deep learning and AI algorithms, enabling them to focus on the relevant aspects of the input dataset [19]. This mechanism is motivated by humans' capacity to focus on the most important features at a specific time and disregard less important information. This selective focus on important parts not only preserves the relevant information but also effectively utilizes the limited resources and time. Initially, attention determines the parts of the input dataset that have greater influence on the output values, in this case, predictions. Attention determines the influence by comparing each part (Query) to the other parts (Key) in the sequence [17]. Typically, dot products are used to determine similarity scores. These scores are then converted into an understandable format using the softmax function. The softmax function converts the initially assigned scores into probabilities that sum to 1. Now, the resulting value, also known as attention weights, decides the importance of a specific element compared to others. Assigning attention weights to specific data in the feature space facilitates the detection of complex dependencies and interactions. This focused attention on specific data is important because not all features in the input dataset contribute equally to the output value. This focused attention is equally important when the computing power is limited. It may evaluate more vital data with fewer computing resources.

2.5.1 Attention in Natural Language Processing

An attention mechanism was originally proposed by Bahdanau et al. in 2014 [82]. It was used for neural machine translation tasks and to address the limitations of fixed-sized context vectors. This method trains neural networks to learn how an input sentence can be translated from the source to the target language. This method uses an RNN Encoder-Decoder architecture for S2S tasks. Generally, the encoder and decoder are fully connected neural networks that are designed to map the input sequence to the output sequence. They may use RNN, LSTM, GRU, LSTM/GRU, CNN, or a transformer model. The encoder transforms

the input sequence into a compressed feature representation, also referred to as context vectors. The decoder then utilizes the compressed representation to produce the output sequence. Hence, the encoder-decoder architecture maps sequences to sequences, even if the input and output have different lengths. The complete architecture consists of an encoder and decoder, trained in combination to increase the prospect of a precise translation [82]. The first neural network-based encoder-decoder architecture was introduced by Cho et al. in 2014 [83]. In his model, the encoder utilized an RNN to process the complete input sequence sequentially, producing a single fixed-sized vector. The RNN-based decoder utilized this vector to give the output sequence word by word. The original RNN encoder-decoder was suitable for short sentences; however, it struggles with lengthy sentences. This is because the encoder cannot store all the critical information for lengthy sentences. This may result in incorrect translations for lengthy and complicated sentences. Hence, to address these issues, the attention mechanism utilizes a bidirectional RNN for the encoder that creates a set of hidden states $(h_1, h_2, \dots, h_n)$ depending on the input sequences $(x_1, x_2, \dots, x_n)$ rather than just a single vector h. The BiRNN can capture both past and future information at each time step. For instance, for the input word x_j, the encoder combines the forward and backward RNNs' hidden states $h_j = [\overrightarrow{h_\iota}, \overleftarrow{h_\iota}]$. The decoder prioritizes the important components of the input sequences of the translation. It comprises an attention block and a unidirectional RNN. It calculates the alignment score ϱ_{tj} for h_j and decoder state d_{t-1} as [84],

$$\varrho_{tj} = e(h_j, d_{t-1}) \tag{2.50}$$

Where e represents the learnable function. Next, the attention weights are calculated by normalizing the alignment score as,

$$\alpha_{tj} = \frac{\exp(\varrho_{tj})}{\sum_{k=1}^{n} \exp(\varrho_{tk})} \tag{2.51}$$

The decoder then calculates the context vector as the weighted sum of all encoders' hidden states.

$$c_t = \sum_{k=1}^{n} \alpha_{tj} h_j \tag{2.52}$$

Finally, the decoder RNN determined the output distribution y_t as,

$$p(y_t|y_1, y_2, \dots, y_{t-1}, x) = RNN\ (c_t) \tag{2.53}$$

2.5.2 Attention in Time Series Forecasting

The early attention mechanism used weighted averages of the input parts. Although its applications are growing in various fields such as language translations, text classification, voice detection, caption creation, time series forecasting, and sequential recommendation systems, numerous advanced variants have been proposed. However, the most important breakthrough occurred when Vaswani et al. (2017) introduced the Transformer Model [85]. The Transformer Model completely replaced the recurrence mechanism with self-attention. This development enhances the model's capacity to capture long-term dependencies more efficiently. The self-attention mechanism used in the Transformer can execute all time-steps concurrently, eliminating the information bottleneck. Hence, following the success of self-attention in NLP, attention mechanisms are achieving prominence in time series forecasting. This is because time series datasets consist of sequential observations, with particular values having a bigger impact on future forecasts. MLMs (ARIMA, SVR, regression, etc.) and conventional DLMs (RNN, LSTM, CNN, etc.) may struggle to capture these influences, particularly for extended time frames. These models have limited capacity to learn complicated trends, seasonal fluctuations, and long-term dependencies. Hence, an attention layer was added to the DLMs, such as RNN and LSTM, to capture these trends most effectively. Qin et al. proposed the first RNN with an attention mechanism for time series prediction [86]. His model introduced a two-stage attention mechanism: 1) input attention, and 2) temporal attention. The input attention mechanism was used to dynamically capture important input attributes, depending on past hidden states. The temporal attention mechanism chooses important encoder hidden states over all time steps. The two separate attentions are then combined and trained using backpropagation algorithms. With growing interest in self-attention, it was increasingly incorporated into more sophisticated time series forecasting models. Lim et al. integrated recurrent and self-attention layers to capture both local patterns and long-term dependencies [87]. The integrated model presents high-

performance results for multi-step time series forecasting while also providing interpretability. Zhou et al. introduced the probsparse self-attention to minimize the computational burden while solving lengthy sequences [88]. The proposed model can significantly enhance inference speed, making it suitable for large-scale prediction applications.

Recently, studies have been using a hybrid combination of attention mechanisms and deep learning models for time series forecasting tasks. For instance, deep learning LSTM and attention mechanisms are integrated for single-step-ahead time series forecasting [19]. The hybrid model (Multi-head ATT-LSTM) combined the capabilities of attention in concentrating on relevant information and LSTM in detecting long-term dependencies. The hybrid model outperformed both single and hybrid models on most of the time series datasets. Currently, CNN-LSTM combined with self-attention has gained prominence in various domains, including PV power, wind energy, and solar irradiance. The hybrid model can leverage the benefits of CNN to capture local patterns, LSTM to learn the temporal dependencies, and self-attention to focus on important patterns. For instance, CNN-attention-LSTM is proposed for next-day solar irradiance forecasting [89]. The hybrid model achieved better scores compared to the single and hybrid models, producing the lowest error rates for most performance metrics. In addition, the attention mechanism is also employed with LSTM, multichannel CNN, CNN-GRU, and CNN-BiLSTM to enhance accuracy and minimize the variable nature of PV power production [17]. Most of the research employed self-attention techniques to identify trends or patterns within the datasets. These studies demonstrated that attention mechanisms have improved forecasting accuracy compared to models without attention.

2.6 Metaheuristic Algorithms for Hyperparameter Tuning

Besides achieving higher forecasting accuracy using hybrid models, the optimal selection of hyperparameters is also crucial. Hyperparameters can have a significant impact on model performance, and selecting optimal parameters can improve forecasting accuracy compared to an unoptimized model. Several

methods are available for hyperparameter tuning, though MOAs can handle complex optimization problems and often exhibit acceptable solutions. MOAs provide a precise and effective technique for exploring the solution space, enabling the model to identify the optimal parameter combinations. These algorithms are stochastic, and at each iteration, they modify the individuals in the population using an intended fitness function to optimize their performance till the stopping criteria are reached. The basic flow of the population-based metaheuristic algorithm is presented in Fig. 2.6. The stochastic nature allows them to avoid local optima, and hence, these algorithms can provide robust solutions to insufficient and noisy data. Additionally, these algorithms are population-based, which helps efficient parallelization. Over the years, various MOAs have been proposed for diverse engineering and computer-related problems. These algorithms are grouped into four main classes based on their principle of operation, as shown in Fig. 2.7 [22]. These include evolutionary, human-based, physics-based, and swarm-based algorithms. Evolutionary algorithms take inspiration from the Darwinian theory of natural selection. These algorithms simulate the behavior of living organisms. GA, DE, and genetic programming are a few popular examples of this class. These algorithms have proven useful for global and engineering optimization problems [90]. However, these algorithms frequently require precise parameter tuning and are computationally more expensive than other MOAs. Additionally, they can show slow convergence, particularly in complex and multi-

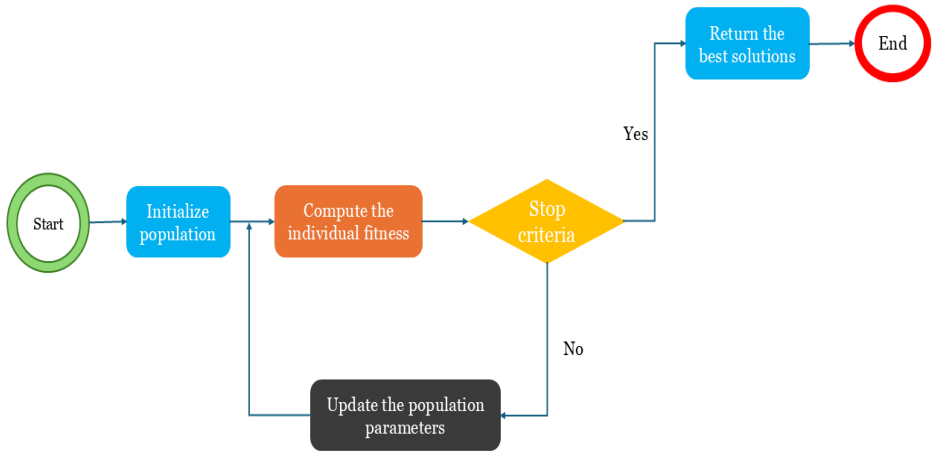


Figure 2.6: Basic workflow of the Metaheuristic Optimization Algorithm

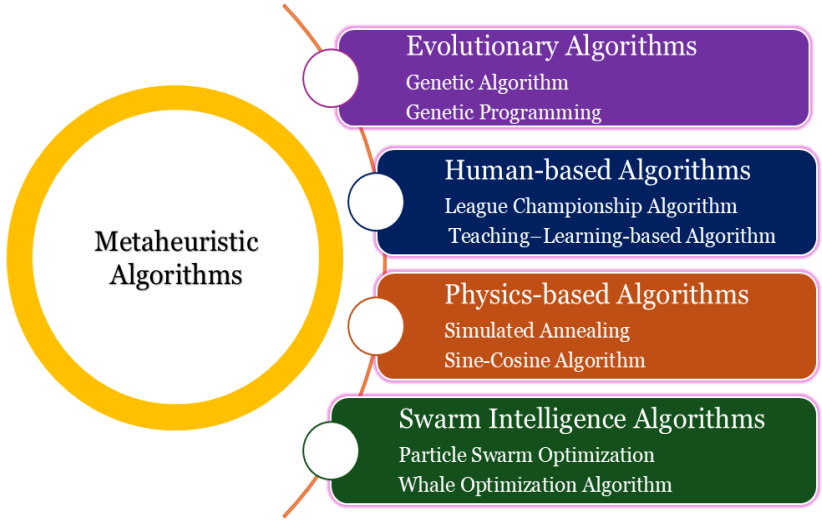


Figure 2.7: Four Classes of Metaheuristic Algorithms, based on their principle of operation

modal optimization problems. The second class of MOAs is human-based algorithms that get inspiration from the thinking, absorbing, and social interaction of humans. Due to humans' excellent cognitive capability, these algorithms may present notable results [91]. League championship algorithm, teaching–learning-based algorithms, and cultural algorithms are some popular examples of these algorithms. Human behavior is highly complex and multifaceted, though these algorithms oversimplify and can impede performance while addressing complex optimization tasks. These algorithms are susceptible to population initialization and parameters [92]. Additionally, these algorithms have an inadequate theoretical background compared to other algorithms.

The third class of MOAs gets inspiration from basic concepts of physical or chemical processes found in nature. Physics-based algorithms are proposed by simulating certain principles of physics and chemistry. The popular examples of these algorithms include simulated annealing, sine-cosine algorithms, and black hole algorithms. Physics-based algorithms maintain a strong equilibrium between exploration and exploitation and are typically scalable to different problem domains. However, these algorithms have a low convergence rate and limited search space exploitation capabilities [93]. Additionally, slower physical processes can result in prolonged processing times compared to other algorithms. The

fourth and most popular class of the MOAs is the swarm intelligence algorithms. These algorithms get inspiration from the collaborative behavior of decentralized systems [23]. They include bacteria foraging, fireflies, insect swarms, and others. These algorithms rely on multiple interactive agents that can efficiently explore and exploit the search space. Each agent in the group obeys simple principles and constantly interacts with another member in the group. In this way, these agents may collectively accomplish complicated tasks without the need for a centralized control system. These algorithms have recently gained greater attention for solving nonlinear optimization problems. They can locate global optima in a complex and multi-dimensional search space. Some well-known examples of these algorithms are PSO, GWO, and WOA. These algorithms are robust and adaptable, which makes them ideal for energy predictions, complex engineering design, deep learning parameter tuning, and various other real-world problems. However, most of these algorithms have a slow convergence, particularly during the exploitation phase. These algorithms can lose diversity and experience a decline in performance in multi-dimensional optimization tasks.

2.6.1 Harris Hawk Optimization and Its Variants

HHO, another popular swarm-based algorithm that gets inspiration from the cooperative and hunting nature of the Harris hawks. This algorithm has multiple distinctive features compared to other prominent swarm intelligence-based algorithms. Firstly, this algorithm exhibits a strong balance between exploration and exploitation [94]. Secondly, this algorithm employs a time-varying method that improves with iterations, resulting in an easy switch from exploration to exploitation. A further distinguishing attribute of this algorithm is its gradual convergence behavior. Such gradual shifts can increase the possibility of finding the most favorable solutions. Additionally, this algorithm's exploitation stage employs a greedy strategy to prioritize the best solutions over those of lower quality, rendering it an appealing method [95]. The principle working of this algorithm is divided into six steps: two explorations and four exploitations. This six-step working principle enables it to evade local optima and enhance convergence speed. Thus, it has shown greater success in various domains, such as the power sector, DLMs parameter tuning, public health, and the environment.

Although HHO can be efficient for solving various optimization problems, due to its stochastic nature, the original HHO may get trapped in local optima, leading to premature convergence. Another problem of HHO is the shift between exploration and exploitation, which is regulated by escape energy. This rigid control may cause exploitation to begin too late, resulting in slow convergence. Additionally, HHO's exploitation phase relies on random jumps, and inappropriate jumps near optimum solutions may result in overshoot. Thus, to achieve low final fitness values, better convergence rate, and higher efficiency, improvement in the exploration and exploitation phases is important.

Various studies have proposed an updated or enhanced version of HHO to address the shortcomings of the original HHO, particularly its inclination for early convergence, slow convergence rate, and fixed exploration/exploitation control. These enhancements include introducing nonlinear escape energy to improve the shift between exploration and exploitation phases and opposition-based learning to increase the convergence rate. For instance, an integrated enhanced variant of HHO is presented that employs an energy regulator, a vector adjustment technique, and an upgraded L_f method to achieve better accuracy and evade premature convergence [96]. An improved version is proposed that integrates HHO with opposition-based learning to enhance exploration and boost convergence curves [97]. Additionally, a self-adaptive method is included to increase diversity and leverage on already explored regions. Some studies used chaotic initialization and mutations to enhance diversity and improve exploration. In [98], HHO efficiency is improved by introducing chaotic maps at three different stages. First, a chaotic map is used to initialize the population of HHO. A second chaotic map is used to generate random numbers for the HHO exploration phase. Finally, a chaotic map is used to control the energy equation in the transition phase. In [99], a boosted mutation-based HHO is proposed to determine the parameters of a PV module. The boosted approach enables HHO to accelerate the convergence speed and explore a wider area of the search basins. Similarly, multiple studies combine HHO with other advanced MOAs to enhance the solution accuracy of different research areas. The hybrid algorithms are then tested on real-world optimization and engineering problems, and the results are compared with the original HHO and other single MOAs. For instance, SSA is integrated with HHO for image segmentation problems [100]. The HHO is used

for the exploration and exploitation phases, while the SSA is employed to update the solutions. Finally, the outputs of both algorithms are combined and processed for the next iterations. Similarly, to improve the exploration ability of HHO, Wunnava et al. used a hybrid combination of DE and HHO for image thresholding [101]. The proposed methods updated the escape energy within the range [0,2].

2.7 Summary and Research Gaps

This chapter examines the evolution of time series forecasting approaches from physical models to classical data-driven models to modern deep learning models. Physical models are reliable and transparent because they rely on physical laws and precise site information. They do not require any previous or historical PV power data and can be used for power estimation of a new power plant [14]. However, these models require accurate PV system specifications, including details about PV panels, panel orientations, module characteristics, and inverter descriptions. Additionally, their accuracy significantly decreases during rapidly changing situations. Classical statistical models are suitable for short-term PV power forecasting. Compared to physical models, these models do not require complex atmospheric physics; instead, they depend on high-quality past data. However, similar to physical models, these models struggle to learn sudden variations in solar irradiance and PV power output data. Additionally, these models presume linearity and stationarity in datasets and may fail to capture nonlinear relationships. MLMs have overcome some of the limitations of the statistical and physical models by learning complex and nonlinear relationships between variables. These models can present more intricate, data-driven approaches compared to physical and statistical methods. Nevertheless, conventional MLMs, such as SVM, RF, and DT, possess a shallow framework and limited learning capacity. These models struggle to generalize, particularly on high-dimensional or noisy datasets.

Recently, DLMs, particularly CNNs, LSTMs, and BiLSTMs, have demonstrated superior performance for time series datasets. They can outperform traditional ML models due to their capacity to learn complex and hierarchical patterns. They can handle multivariate and high-dimensional inputs. However, these models still

have limitations in detecting various complex patterns in modern PV power datasets. This is because PV power output may include a combination of linear patterns, complex relationships, noise, and seasonality; single models may not be enough to capture these factors. Weather variables, seasonal effects, and sudden fluctuations can significantly affect the PV power output, and single DLMs may struggle to mimic these nonlinearities, seasonality, and sudden shifts. Hence, to increase pattern recognition capabilities, improve forecasting accuracy, and the convergence rate, several research studies have combined multiple DLMs. The hybrid combinations have shown promising results over single deep learning models. The most feasible choice for PV power forecasting is combining BiLSTM and CNN. This is because BiLSTM can excel in capturing long-term temporal dependencies, while CNNs can better capture sudden or local patterns. The BiLSTM-CNN hybrid model can better explain the complex and nonlinear patterns in PV power datasets. However, most studies use sequential stacking of BiLSTM-CNN, which is computationally expensive and may not be suitable for multi-task learning, therefore restricting real-time responses. Additionally, the sequential model performs multiple transformations, increasing the possibility of information degradation or loss of important aspects. Recent studies increasingly employed self-attention mechanisms to enhance the forecasting accuracy of both single and hybrid models. For instance, CNN-attention-LSTM is proposed for next-day solar irradiance forecasting [89]. The hybrid model achieved better scores compared to the single and hybrid models, producing the lowest error rates for most performance metrics. The attention mechanism is also integrated with other hybrid models, including CNN-GRU and CNN-BiLSTM, to improve accuracy and minimize the variable nature of PV power production [17]. However, self-attention cannot fully detect patterns within datasets and is also less resilient to noisy data. It employs a single head, which allows it to learn only one relation across the entire sequence. This reduces the diversity of the learned attributes due to the model's inability to capture multiple unique dependencies. Sometimes the single learn feature can lead to overfitting, particularly in noisy datasets.

Another challenge is selecting suitable hyperparameter settings for DLMs. Without proper hyperparameter settings, even the best hybrid deep learning models may fail to achieve better accuracy. Studies suggest that DLMs tuned by metaheuristic algorithms can attain 8% higher results than untuned models.

Recently, MOAs, particularly HHOs, have shown greater success in different optimization problems, particularly for hyperparameter tuning of DLMs. However, these studies used original HHO, which is prone to getting trapped in local optima for multi-dimensional problems. Additionally, in the original HHO, the shift between exploration and exploitation is regulated by linear escape energy. This rigid control may cause exploitation to begin too late, resulting in slow convergence. Furthermore, HHO's exploitation phase relies on random jumps, and inappropriate jumps near optimum solutions may result in overshoot. Hence, there is a need for a robust forecasting model that combines temporal dependencies modeling, attention-based weighting, and adaptive optimization capabilities to improve accuracy and generalization. Hence, to cover this gap, this dissertation proposes a hybrid deep model, BiLSTM-CNN-MHA, with an adaptive Harris Hawk optimization for multi-step ahead solar power forecasting.

3

Dual-Stream Attention-based Hybrid Deep Learning Model

3.1 Overview of the Proposed Framework

Accurate PV power forecasting remains a major issue due to its dependence on various dynamic environmental factors. Weather conditions, geographical factors, forecasting horizons, and technical aspects all contribute to the complexity of forecasting. Physical, classical statistical, and machine learning models may fail to capture these complex relationships in PV power output. These models struggle to generalize, particularly on high-dimensional or noisy datasets. Deep learning, a subset of ML, possesses many hidden layers and can learn complex and hierarchical patterns. They can outperform physical, classical statistical, and traditional MLMs. These models can handle multivariate and high-dimensional inputs. These features become critical when dealing with the dynamic nature of environmental conditions. Hence, DLMs have become prominent in many fields, particularly for PV power forecasting. Over time, multiple DLMs have been proposed; however, RNN, CNN, and Transformer are widely used for PV power forecasting. These models are highly researched and mature for forecasting tasks. RNNs provide precise forecasting when evaluating sequential datasets [102]. They can capture temporal relationships more effectively than other ML models. However, standard RNNs may struggle when processing long sequences due to vanishing and exploding gradient problems. To overcome these problems, two advanced variants, LSTM and GRU, were introduced in 1997 and 2014, respectively [103]. Both LSTM and GRU possess short- and long-term memory. These features allow them to learn long-term dependencies more effectively than

a standard RNN. However, GRU and LSTM still have shortcomings, particularly for large datasets with long-term horizon prediction. These models primarily focus on a unidirectional flow of information; hence, their capacity to fully capture patterns and trends is limited. Advanced RNN variants, including BiLSTM, can be suitable since they process data in both forward and backward directions. Compared to unidirectional LSTM and GRU, BiLSTM extracts more information from a sequence by considering both past and future contexts simultaneously [104]. Another DLM, CNN, that excels in capturing local patterns, has recently gained wide attention. This is because PV power data contain both local patterns, such as sudden spikes or short weather events, and long-term temporal patterns, such as daily cycle or seasonal variations. While BiLSTM is designed to capture long-term temporal patterns, the CNN can learn local patterns more effectively compared to other models. The convolutional filter in CNN slides over the PV dataset to capture small-scale variation. Similar to temporal patterns, local patterns are important for accurate PV power forecasting.

Since PV power data contains both temporal and local patterns, a single DLM may not fully extract detailed information from PV datasets. Hence, in the last few years, hybrid models have been increasingly employed for PV power forecasting. These models combine the characteristics of two or more DLMs, which allows them to capture multiple data features with higher forecasting accuracy. For instance, in [105], GRU, LSTM, and BiLSTM combine to predict short-term solar energy using an Amherst dataset. Among different combinations, LSTM-BiLSTM achieved high forecasting accuracy, demonstrating low MSE, MAE, and RMSE at timestep 24, with values of 0.00163, 0.01821, and 0.0404, respectively. The integration of CNN with RNN and its variants is a suitable choice for PV power forecasting. This hybrid combination can effectively learn both local and temporal patterns within PV power datasets [79]. Hence, multiple studies have combined CNN and different variants of RNN. A hybrid model integrating CNN and LSTM is widely researched due to its capacity to detect both local and temporal patterns. However, few studies suggest that changing the order of the sequence in the CNN-LSTM model yields better forecasting outcomes [80], [106]. For instance, a hybrid LSTM-CNN model is proposed for solar power prediction using a dataset from Alice Springs, Australia, and the model's results are compared with those of CNN, LSTM, and CNN-LSTM [80]. Some studies have

integrated CNN with BiLSTM, demonstrating superior forecasting performance compared to both LSTM-CNN and CNN-LSTM. A CNN-BiLSTM hybrid model is introduced for solar radiation prediction using datasets from three distinct geographical locations [107]. The CNN-BiLSTM model outperforms the CNN-LSTM and CNN-GRU for univariate and multivariate forecasting. Nonetheless, there is evidence that BiLSTM-CNN can outperform CNN-BiLSTM [18], [108]. These analyses and comparisons were based on real-time PV power data collected from Alice Springs. Hence, BiLSTM-CNN has the capacity to capture both temporal and spatial features, outperforming single and hybrid combinations of deep learning models. Recently, some studies have utilized attention mechanisms to enhance the capture of temporal dependencies. These studies used an attention mechanism with both single and hybrid DLMs. The attention mechanism is employed with LSTM [109], multichannel CNN [110], and CNN-GRU [111]. The results show that the attention-based DLMs can provide enhanced generalization, improve accuracy, and mitigate the variable nature of PV power production. Since BiLSTM, CNN, and attention mechanisms are suitable for extracting detailed information from PV power output, most studies combined them for forecasting tasks. For instance, attention mechanisms are used with CNN-BiLSTM [112], [113] and BiLSTM-CNN [108], [114]. However, these studies employed a sequential stacking of CNN and BiLSTM, in which the first model processes the data first, followed by the second. The sequential stacking of CNN and BiLSTM results in slower computation speeds and may not be suitable for multi-task learning. Additionally, most of these studies employed self-attention mechanisms for capturing spatiotemporal patterns; however, self-attention cannot fully detect patterns within datasets and is also less resilient to noisy data. To overcome the limitations of previous studies, this chapter presents a dual-stream attention-based hybrid model for PV power forecasting. The proposed model employs a parallel processing approach in which both BiLSTM and CNN analyze the input data separately to learn temporal and spatial patterns. The outputs of these models are concatenated and passed to the MHA layer, which improves forecasting accuracy by assigning different weights to specific data. It enhances the model's durability by focusing on important input variables. Additionally, it accounts for variations in PV power and weather. Hence, the proposed model can achieve better results compared to single and hybrid models.

3.2 CNN for Local Feature Extraction

CNN is a popular deep learning model that excels in data analysis with a specific grid-like structure. It is primarily used for image-focused applications such as image classification, object detection, face recognition, and image segmentation. CNN is designed to replicate the visual perception mechanisms of living beings. It can perform both supervised and unsupervised learning. The main advantage of this model is that it reduces the number of parameters, allowing it to solve multifaceted and complex tasks. Additionally, it can effectively handle noisy data by eliminating noise at each layer. Hence, these features further extended their applications in multiple areas, including medical imaging, computer vision, art creation, natural language processing, forecasting, and environmental monitoring. For forecasting tasks, it has been widely researched due to faster training and automatic feature extraction capabilities. It is also highly effective for capturing local patterns. Hence, over the years, various CNN architectures have been proposed; yet the basic structure of most of them remains similar. Fig. 3.1 shows the basic structure of a CNN. The figure shows that CNN consists of three main layers: convolutional, pooling, and fully connected.

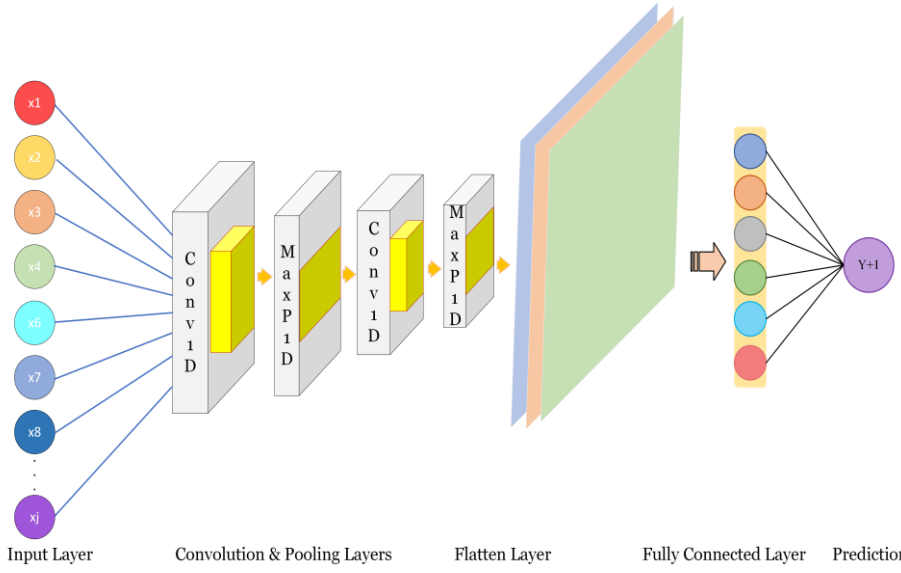


Figure 3.1: A two-layer structure of the CNN model

Convolutional layer: The convolutional layer is the first and most important layer in CNN architecture. The main function of this layer is to detect short-term dependencies in PV datasets. Since abrupt weather fluctuations, shading effects, and panel heating greatly impact the PV power output. The convolutional layer applies sliding filters over the PV datasets to detect these local patterns. Each filter has a specific weight that is initially assigned at random. Then these weights are modified throughout the training process to capture more useful local patterns. Each filter captures different local patterns. For instance, one filter may capture sudden spikes in irradiance due to moving clouds, while another filter may detect a sharp decline in PV power output due to rainfall. After employing these filters on the input data, the output obtained is known as a feature map. Hence, multiple filters in the convolutional layer generate multiple feature maps, each focusing on a distinct feature of the PV data. These attributes make the convolutional layer effective for PV power prediction, where it is important to capture both weather-induced variations and more regular daily patterns. Mathematically, a convolutional layer can be expressed as,

$$x_r^m = f\left(\sum_{s \in M_r} x_s^{m-1} * k_{sr}^m + b_r^m\right) \tag{3.1}$$

Where x_r^m denotes the rth output feature map of the convolutional layer. f represents the activation function, m represents the number of layers, and k denotes the kernel. The activation function plays a vital role in the convolutional layer. It introduces nonlinearity into the model, allowing CNN to capture more complex relationships in the PV datasets accurately. Without using an activation function in the convolutional layer, the convolution operation will be just a linear mapping of the input datasets.

Pooling layer: The pooling layer is employed after the convolutional layer to perform a down-sampling operation. This downsampling process reduces the dimension of the feature maps while retaining the important information. Hence, the pooling layer reduces the number of parameters, which indirectly reduces the computational cost. Additionally, by lowering the parameters, it prevents the CNN model from getting too complicated, hence avoiding model overfitting. Normally, two methods are employed in the pooling layer: max pooling and average pooling. In max pooling, the maximum value in each window is selected. This allows the

CNN model to focus on the most dominant and important features, such as sudden drops in temperature or irradiance. In average pooling, the mean of each window is selected. This results in better representation of the data, focusing on general patterns rather than local sudden variations. Both max and average pooling can be useful for model generalization; however, for PV power data, which is highly influenced by sudden variations, max pooling is normally preferred. Mathematically, the pooling layer can be expressed as,

$$x_s^m = f(B_s^m * pool(x_s^{m-1}) + b_s^m) \tag{3.2}$$

Where $pool$ indicates the pooling function, x_s^{m-1} indicates the feature map of the $m-1$th layer, and B_s^m indicates the bias.

Fully connected layer: This is the final layer of the CNN architecture, which provides the model's output. The convolutional and pooling layers function as feature extractors, detecting essential patterns in the input dataset. The fully connected layer combines these extracted features to learn relationships that are important for forecasting. In other words, this layer functions as a classifier or regressor. Mathematically, the fully connected layer can be expressed as,

$$y^C = f(w^m * y^{m-1} + b^m) \tag{3.3}$$

Where y_c^m, w^m, and b^m indicate the output vector, the weight, and the bias.

3.3 BiLSTM for Long-Term Temporal Dependencies

BiLSTM is an advanced variant of RNNs that enhances the learning capability of traditional LSTMs. It is designed to overcome some of the limitations of the conventional LSTM. Traditional LSTM allows the flow of information in only one direction, which limits its ability to capture detailed information from input datasets. BiLSTM, on the other hand, analyses the input dataset in both forward and backward directions. This two-directional flow enables the BiLSTM to explore both past and future contexts simultaneously. This attribute is useful to capture long-term dependencies more efficiently compared to a unidirectional LSTM. Hence, over the years, BiLSTM has been increasingly employed in multiple domains, including NLP, speech recognition, time series forecasting [18], protein

classification [115], voice activity detection [116], biomedical [117], and others. The two-way flow of information is predominantly effective for PV power forecasting, as both past trends and future forecasts can greatly influence model performance. Since BiLSTM is made of two LSTMs, its operation can be understood by analyzing the LSTM unit. An LSTM unit employs an advanced gating framework to control the flow of information. It retains a cell state, usually known as ${C_t}^L$, serving as long-term memory. The cell state transmits the accumulated information about the sequences from one timestep to the subsequent one until the sequence ends. During the flow of information, there are minor interactions that maintain the information unchanged. In the LSTM unit, three gates are used to retain the stability of cell states. These include input, output, and forget gates.

Forget Gate (${f_t}^L$): Determines the information that needs to be removed from the cell state.
Input Gate (${i_t}^L$): Determines the information that needs to be preserved in the cell state.
Update Gate (${O_t}^L$): Determines the information that is sent from the cell state.

Hence, these gates are used to either remove or preserve data from previous time steps, enabling the LSTM to sustain long-term dependencies. The output of each gate is a number between 0 and 1, where 0 indicates 'remove' and 1 indicates 'preserve'. At first, the forget gate examines both past state information and new data using the sigmoid function (σ), as shown in Fig. 3.2. The forget gate looks at h_{t-1} and x_t to decide which data or value should be removed from the cell state. Once the unwanted information is removed from the cell state, the next phase is to add the values that need to be preserved in the cell state. This is performed by the input gate, which uses the σ and hyperbolic tangent functions($tanh$) to decide the new data that needs to be retained in the cell state. After determining the new data, the cell state (${C_t}^L$) is updated using data from the two previous gates: forget and input gates. The output of the forget gate is multiplied by the previous cell state (${C_{t-1}}^L$) and then it is added to the output of the input gate, as shown in Fig. 3.2. The new cell state is passed to the output of the LSTM unit. Finally, the output gate determines what will be the new hidden state (${h_t}^L$). The output gate first employs σ similar to the input gate to determine which data should pass through.

This is done using the previous hidden state ($h_{t-1}{}^{L}$) and current input (x_t). Then$C_t{}^{L}$ is passed through *tanh* to transform the value between -1 and +1 and

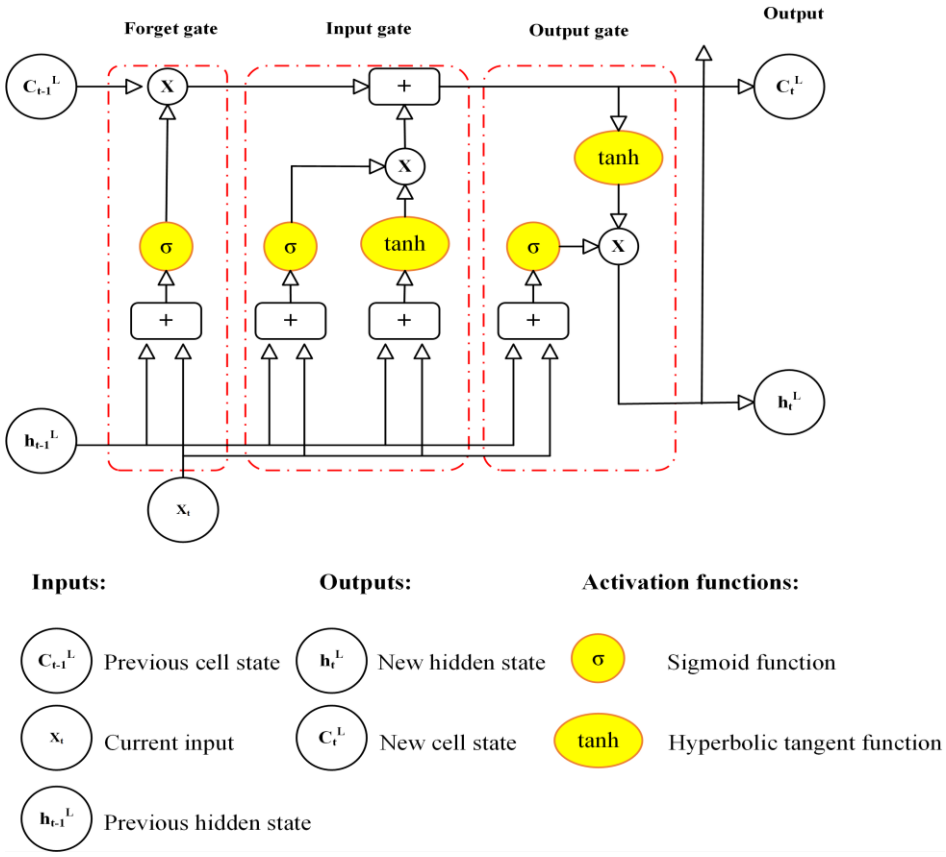


Figure 3.2: The flow Operation of the LSTM unit

multiply it by the sigmoid function output. Mathematically, the complete process of the LSTM unit can be expressed using the following equations,

$$f_t{}^{L} = \sigma_{fg}\left(w_{fg}.\left[h_{t-1}{}^{L}, x_t\right] + b_{fg}\right) \quad (3.4)$$

$$i_t{}^{L} = \sigma_{ig}\left(w_{ig}.\left[h_{t-1}{}^{L}, x_t\right] + b_{ig}\right) \quad (3.5)$$

$$\check{C}_t{}^{L} = tanh\left(w_{ct}.\left[h_{t-1}{}^{L}, x_t\right] + b_{ct}\right) \quad (3.6)$$

$$C_t{}^{L} = C_{t-1}{}^{L} * f_t{}^{L} + \check{C}_t{}^{L} * i_t{}^{L} \quad (3.7)$$

$$O_t{}^{L} = \sigma_{og}\left(w_{og}\left[h_{t-1}{}^{L}, x_t\right] + b_{og}\right) \quad (3.8)$$

$$h_t{}^{L} = O_t{}^{L} * tanh\left(C_t{}^{L}\right) \quad (3.9)$$

From Fig. 3.2, the LSTM examines data chronologically from $t = 1$ to $t = N$. This one-way flow of data can be considered a shortcoming of the traditional LSTM, particularly for capturing long-term dependencies in large datasets. Additionally, in some real-world situations, the data from the future instances is equally pertinent. For example, an abrupt fluctuation in PV power output at 2:30 pm can be due to passing clouds before 2:30 pm. Nevertheless, the forecasting model's ability can be further enhanced if it learns that PV power was completely restored at 3:00 pm, suggesting an abrupt fluctuation in PV power rather than a long-term change. Hence, to overcome these shortcomings, BiLSTM is used, which comprises two LSTMs; one operates in the forward direction (from $t = 1$ to $t = N$) while the other operates in the backward direction (from $t = N$ to $t = 1$), as shown in Fig. 3.3. The forward LSTM ($LSTM_{forward}$) encodes the sequence from all the preceding steps, and the backward LSTM ($LSTM_{backward}$) encodes the sequence from all future steps. Evaluating data in both directions could identify hidden correlations that might not be visible when using a single LSTM unit. Hence, BiLSTM can provide high forecasting accuracy compared to unidirectional LSTM. Mathematically, the $LSTM_{forward}$ and $LSTM_{backward}$ can be expressed as,

$$\overrightarrow{h_t} = LSTM_{forward}(x_t, \overrightarrow{h_{t-1}}) \tag{3.10}$$

$$\overleftarrow{h_t} = LSTM_{backward}(x_t, \overleftarrow{h_{t+1}}) \tag{3.11}$$

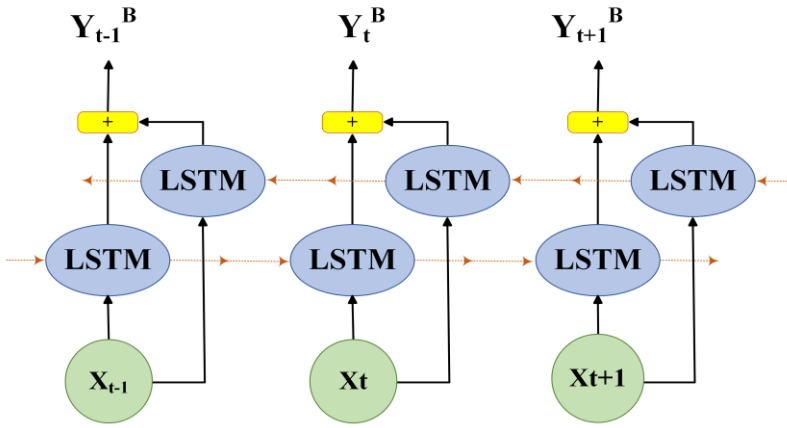


Figure 3.3: The BiLSTM basic structure, showing two LSTMs operating simultaneously on each time step

The two hidden states provide different views of the same input. By concatenating these two hidden states, the BiLSTM can offer a single, yet richer representation of the input data. This single concatenated hidden state (${h_t}^B$) combines data from both the past ($\overrightarrow{h_t}$) and future ($\overleftarrow{h_t}$). Mathematically, ${h_t}^B$ can be expressed as,

$$ {h_t}^B = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{3.12} $$

The final output of BiLSTM (${y_t}^B$) can be expressed as,

$$ {y_t}^B = w_y {h_t}^B + b_y \tag{3.13} $$

Where $w_y = w_{yf} + w_{yb}$ is the concatenated weight. For PV power forecasting, BiLSTMs offer a competitive advantage compared to other models. They allow the model to learn both the structure and importance of the events. Hence, knowing detailed information about the data can yield more precise and stable predictions.

3.4 Multi-head Attention Mechanism for Dynamic Weighting of Feature

Deep learning models, including RNNs, LSTMs, and BiLSTMs, examine the dataset sequentially, which makes them an excellent choice for PV power forecasting. Although BiLSTM has improved performance compared to LSTM, its limited memory capacity restricts it from storing detailed information in the hidden states. Similar to other RNN variants, it updates the hidden states sequentially, which may deteriorate while capturing very lengthy sequences. Hence, in such lengthy sequences, BiLSTM may lose important information. An attention mechanism was proposed in 2014 to overcome the information degradation problem. An attention mechanism assigns attention weights to specific data of the feature space, enabling the detection of complex dependencies and interactions in lengthy sequences. Instead of storing all previous data in the hidden state, it computes attention scores between the query vector and a set of key vectors. Typically, dot products are used to determine these scores. For query (Q), key (K), and value (V), the attention scores can be computed as,

$$ Score\ (Q, K) = {Q.K^T}\Big/{\sqrt{d_k}} \tag{3.14} $$

Where d_k is the key vector dimensionality. These attention scores show the influence of each past timestep on the current prediction value. For instance, a low attention score indicates that the corresponding timestep offers less important information for the current prediction value. On the other hand, a high attention score illustrates that the corresponding timestep provides meaningful information for prediction tasks. Once the attention scores are attained, a softmax function is employed to convert them into a probability distribution. By doing this, the generated attention weights are ensured to be non-negative and sum to one.

$$AW = \text{softmax}\left(Q.K^T \Big/ \sqrt{d_k}\right) \tag{3.15}$$

Where AW represents the attention weight. The normalized weights after applying a softmax function are used to produce a weighted sum of values, also known as the context vector.

$$CV = \sum_{j=1}^{T} AW^j V^j \tag{3.16}$$

Where CV is the context vector. The context vector acts as a customized memory, focusing on the important timesteps rather than equally handling all inputs. This allows it to concentrate adaptively on the information that is highly valuable for precise prediction. In the multi-head attention, this overall process is repeated across multiple heads. These heads are working in parallel, and each head learn a different aspect of the input dataset. Each head concentrates on a unique query, key, and value matrices. For instance, one head may learn short-term variations while another may learn long-term patterns. Mathematically, each head is computed as,

$$head_i = softmax\left(Q^j.K^j \Big/ \sqrt{d_k}\right)V^j \tag{3.17}$$

Where $Q^j = QW^{q,j}$, $K^j = KW^{k,j}$, and $V^j = VW^{v,j}$ are the projected queries, keys, and values. The output of multi-head attention is obtained by combining the outputs of all h heads.

$$MHA = Concat(head_1, head_2, \dots, head_h)W^h \tag{3.18}$$

Where W^h is the output projection matrix. The multi-head attention provides a more detailed and richer representation of the sequence. Compared to RNN and

its variants, as well as self-attention, multi-head attention can reduce information loss. It can improve the model's capacity to interpret various temporal dependencies, thereby enhancing forecasting accuracy and interpretability.

3.5 Integration of BiLSTM-CNN–Multi-Head Attention

The hybrid deep learning model integrating BiLSTM, CNN, and multi-head attention is presented in Fig. 3.4. As shown in the figure, the proposed model introduced a parallel processing technique in which both BiLSTM and CNN receive the preprocessed data separately. Both these models analyze the preprocessed input data concurrently. Parallel input processing to BiLSTM and CNN is an effective design for handling time series data. This structure enables the proposed model to learn both short and long-term dependencies more effectively. This reduces the shortcomings of sequential feature extraction. Additionally, the parallel processing approach allows each model to focus on its unique abilities. For instance, CNN captures complex spatial relationships and local patterns from the input dataset. These features are effective for detecting short-term fluctuations and changes that can instantly impact PV power output.

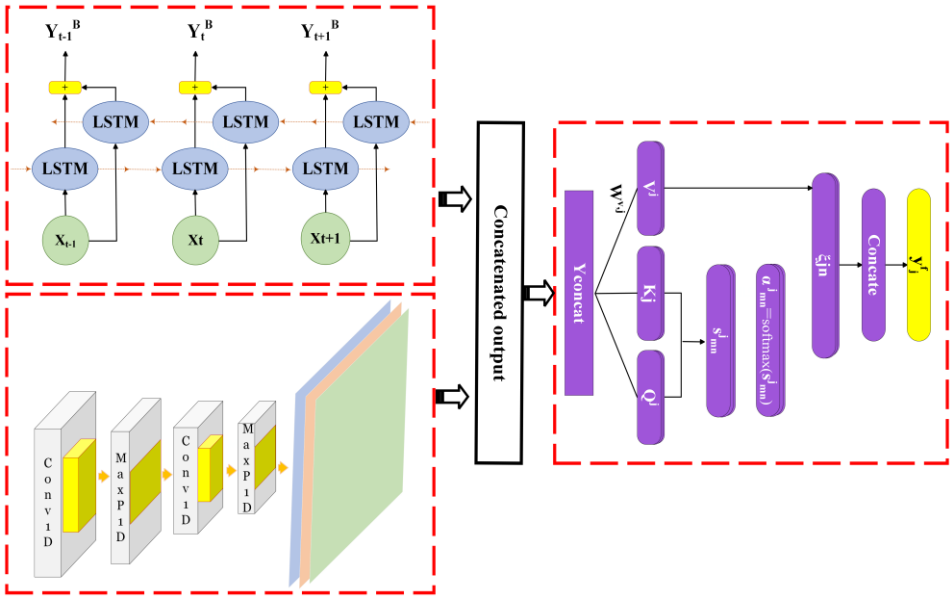


Figure 3.4: Dual-stream BiLSTM-CNN with multi-head attention structure

Additionally, the parallel processing approach allows each model to focus on its unique abilities. For instance, CNN captures complex spatial relationships and local patterns from the input dataset. These features are effective for detecting short-term fluctuations and changes that can instantly impact PV power output. Detecting these features can enhance the model's ability to understand the local interactions between multiple variables. The BiLSTM, on the other hand, learns temporal relationships and long-term dependencies from the same input dataset. The two-way flow of information allows it to capture recurring patterns and the delayed impacts of weather and atmospheric factors on PV power production. It provides a deeper contextual representation that describes the way environmental factors evolve, supplementing the local patterns captured by the CNN. Hence, the parallel processing can learn spatial and temporal patterns simultaneously, enhancing accuracy and representational depth compared to the sequential BiLSTM-CNN model. Additionally, a parallel model can effectively use the GPUs' parallel computing capacity. The proposed approach can reduce training time compared to serial processing, in which the BiLSTM must first process the input stream before the CNN examines it, resulting in a computational bottleneck. In contrast, the parallel approach executes multiple computations simultaneously, resulting in faster convergence, especially with extended sequences. Moreover, the parallel model performs minimal transformation on the input data stream. Compared to the sequential model, which employs multiple transformations (BiLSTM→CNN→Dense), the parallel model reduces such types of transformations. Each model in a parallel architecture processes the input stream directly, reducing information degradation or loss of important aspects. Such an approach generalizes better since it includes multiple learning perspectives. It also provides an efficient and reliable framework for combining multi-head attention layers. After concatenating the output of both BiLSTM and CNN, the multi-head attention layer can accurately identify important parts of the sequences, resulting in improved comprehension and reduced complexity.

Let $Y_B = \{{y_1}^B, {y_2}^B, \dots, {y_\mathrm{T}}^B, \}$ represent the outputs of the BiLSTM and $Y_C = \{{y_1}^C, {y_2}^C, \dots, {y_\mathrm{T}}^C, \}$ represent the outputs of the CNN. The BiLSTM model extracts temporal dependencies, while the CNN captures local patterns from the dataset. Upon extracting both temporal and local patterns, the outputs of the BiLSTM and

CNN are concatenated and then sent to the MHA layer. The outputs of the concatenated layer Y_{Concat} can be expressed as,

$$Y_{concat} = Concat(Y_B, Y_C) \tag{3.19}$$

$Y_{concat} \in \mathbb{R}^{T(d_C+d_B)}$, where T, d_C, d_B represent the sequence length, CNN, and BiLSTM outputs dimensionality. The MHA captures complex temporal correlations and trends across different features, enabling the model to focus on key data points for improved forecasting precision. The multi-head attention processed the outputs of the concatenated layer ($Y_{concat} = \{y_{concat}{}^1, y_{concat}{}^2, \dots, y_{concat}{}^T\}$ to generate a query Q^j, key K^j, and value V^j vectors.

$$\begin{cases} Q^j = Y_{concat} W^{q,j} = [q_1{}^j, q_2{}^j, \dots, q_T{}^j] & (3.20) \\ K^j = Y_{concat} W^{k,j} = [k_1{}^j, k_2{}^j, \dots, k_T{}^j] & (3.21) \\ V^j = Y_{concat} W^{v,j} = [v_1{}^j, v_2{}^j, \dots, v_T{}^j] & (3.22) \end{cases}$$

Where $W^{q,j}$, $W^{k,j}$, and $W^{v,j}$ represent the weight metrics for query, key, and value, respectively. The attention scores $s^j{}_{nm}$ can be determined by taking the dot product of Q^j and K^j at the n^{th} and m^{th} time step for head j, followed by normalization to get the attention weight $\alpha^j{}_{nm}$.

$$S^j{}_{nm} = q_n{}^j . k_m{}^j \Big/ \sqrt{d_k} \tag{3.23}$$

$$\alpha^j{}_{nm} = softmax(s_{nm}{}^j) \tag{3.24}$$

$$\alpha^j{}_{nm} = exp(s^j{}_{nm}) \Big/ \sum_{k=1}^{T} exp\ (s^j{}_{nk}) \tag{3.25}$$

The output (attention coefficient) of the j[th] head for the n[th] position is subsequently obtained by aggregating all weighted value vectors. The final output of a single attention head is determined by combining the attention coefficients across all positions.

$$\xi^j{}_n = \sum_{s=1}^{T} (\alpha^j{}_{ns} . v^j{}_s) \tag{3.26}$$

$$head^j = [\xi^j{}_1, \xi^j{}_2, \dots, \xi^j{}_T] \tag{3.27}$$

In multi-head attention, multiple heads operate simultaneously, allowing each head to concentrate on a unique query, key, and value matrices. Hence, the output of multi-head attention is obtained by combining the outputs of the h-heads as:

$$y^m = Concat(head_1, head_2, \dots, head_h)W^h \tag{3.28}$$

Following the concatenated output, the global pooling function is used to generate a single vector of predicted results, which is then passed to the final dense layer.

$$y^f{}_j = \sum_{i=1}^{dcon} W^f{}_{ij} \,.\, \sigma(GlobalPooled_j) + b^f{}_j \tag{3.29}$$

Where $y^f{}_j$, $GlobalPooled_j$, $\sigma, W^f{}_{ij}$, and $b^f{}_j$ denote the output of the dense layer, output from the GlobalPooled, activation function, weight matrix, and bias.

3.6 Workflow of the Proposed BiLSTM-CNN-MHA Model

The flowchart of the proposed BiLSTM-CNN-MHA is presented in Fig. 3.5. The process starts with the data preparation phase, in which the unprocessed input data is converted into an optimized form. Multiple transformation phases are used for temporal data sequences $x_t \in \mathbb{R}^{N\times T\times D}$, where N, T, and D indicate batch size, length of sequence, and input dimension. In most cases, the input data contains various missing and inaccurate values that can result in poor model performance. DLMs learn trends and patterns in the input datasets. When the data contains missing or noisy values, DLMs may learn incorrect relationships, resulting in overfitting [21]. These models minimize the loss function using the gradient descent technique, which indicates how much each parameter should be updated during training. Missing or noisy values can result in distorted feature values, incorrect loss estimation, and erroneous gradient updates. When these values occur irregularly, DLMs may learn biased relationships. The data cleaning step also includes removing outliers or anomalies present in the datasets. Outliers are observations that differ drastically from the general dataset's patterns and trends. These can arise from multiple reasons [118], including sensor malfunctions, instrument calibration problems, shading effects, power inverter faults, preventive maintenance periods, and sudden weather fluctuations [21]. Similar to

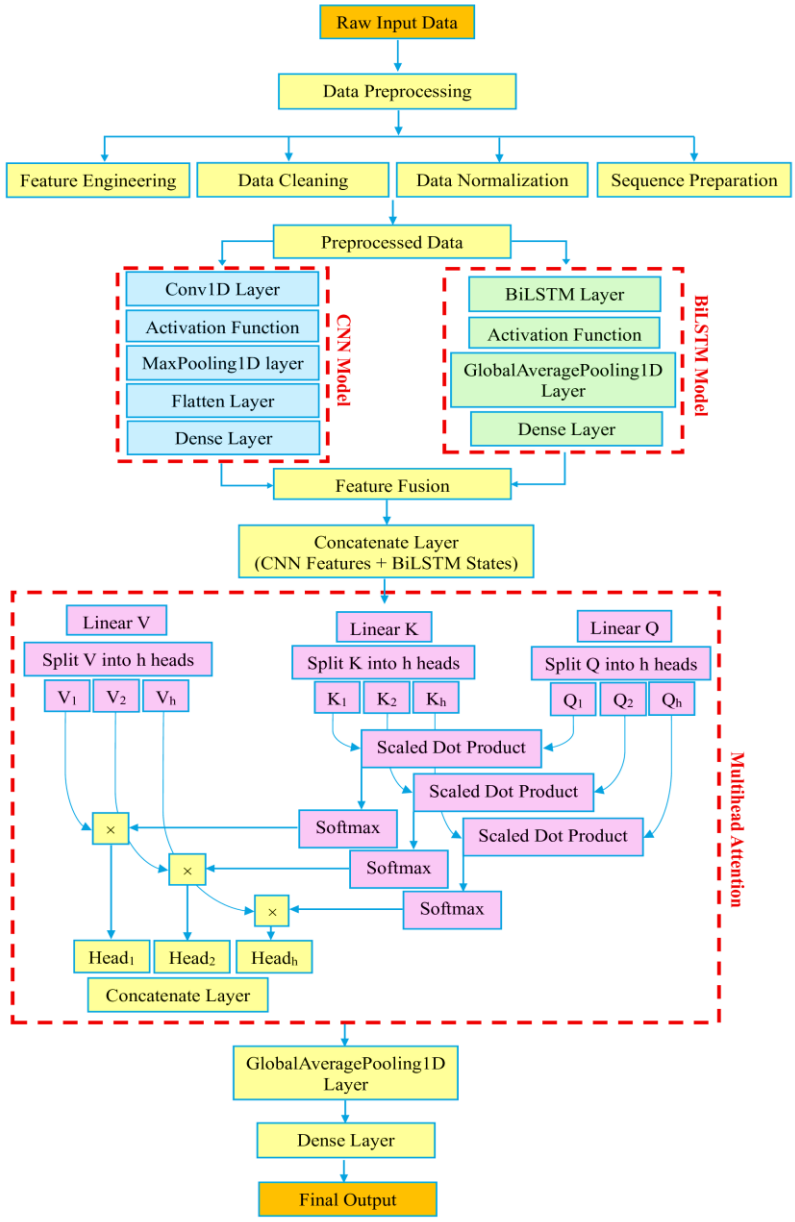


Figure 3.5: The flowchart of the dual-stream BiLSTM-CNN with multi-head attention

missing values, outliers or anomalies can significantly impact the model performance. They may extend the range of the measured values, including average and standard deviation, resulting in inaccurate analysis. Additionally, they may distort the patterns and trends in the dataset, resulting in low forecasting

accuracy. Hence, removing outliers is important for enhanced interpretability, faster convergence, and better generalization. In the literature, various techniques are employed to remove outliers or anomalies. These include interpolation, the Z-score method, the wavelet transform, the interquartile range, isolation forest, and others [21]. Among them, the interquartile range and Z-score are widely used techniques due to their robustness and simple implementation.

After removing outliers and missing values, dataset normalization is an important step in PV data analysis. The PV power dataset contains multiple parameters, including PV power (kW), solar irradiance (W/m^2), relative humidity (%), wind speed (m/sec^2), temperature (°C), and wind direction (°A). These parameters have different scales and units. Supplying an unnormalized dataset, particularly to deep learning models such as CNN and BiLSTM, may result in low forecasting accuracy and poor performance. Data normalization transforms the datasets to a common range, usually 0 to 1, without affecting the correlation between these parameters. Typically, MinMaxScaler is used for data normalization. Data normalization allows DLMs to handle all parameters equally. This feature prevents the model from being dominated by the parameters that have a wider numeric range, such as solar irradiance or PV power output. Additionally, data normalization facilitates faster convergence, enhanced model performance, and improved generalization. Following the data normalization, sequence preparation is also crucial for deep learning models. In sequence preparation, the dataset is transformed into a structure that is compatible with the model's structural needs, especially for LSTM, BiLSTM, or any hybrid combination of these models. These models analyze data through gating mechanisms that retain temporal relationships across the sequence. Hence, they require a suitable data structure to utilize their sequential modeling capacity. Typically, a sliding window method is used, in which input-output pairs are created. The data structure for recurrent models is commonly a three-dimensional tensor of shape[batch size, sequence length, features]. The batch size indicates the number of training samples in one forward or backward pass. It regulates the learning rate, memory utilization, and learning stability. A small batch size can provide uniform learning performance, but the model will require a longer training time. On the other hand, a high batch size can provide faster training but may use a large amount of memory. A sequence length is the number of timesteps

input to the model. In other words, it indicates the number of previous data points used for predicting future values. An extended sequence length may detect both short and long-term temporal patterns; however, it can be computationally more expensive. Features may be referred to as input variables or parameters. In PV power data, features can be wind speed, humidity, temperature, or irradiance. They are important for the model's performance and validation because they help the model understand the factors that affect PV power production.

The pre-processed data is fed into both CNN and BiLSTM separately, as shown in the figure. The CNN branch employs four layers: Conv1D, MaxPooling1D, Flatten, and Dense, while the BiLSTM branch uses three layers: BiLSTM, GlobalAveragePooling1D, and Dense. The Conv1D layer applies sliding filters over the PV datasets to detect local patterns, while the MaxPooling1D layer is employed to perform a down-sampling operation. The Flatten layer transforms multi-channel feature maps into a 1D vector without affecting the original data. This layer acts as a bridge between the max pooling layer and the final dense layer. The final dense layer, also referred to as a fully connected layer, cannot support multi-dimensional output directly from the pooling or convolutional layers. It can only process a one-dimensional array. Hence, the Flatten layer performs a transformation suitable for the dense layer. In the BiLSTM branch, the $LSTM_{forward}$ encodes the sequence from all the preceding steps (from $t = 1$ to $t = N$), and the $LSTM_{backward}$ encodes the sequence from all future steps (from $t = N$ to $t = 1$). Both $LSTM_{forward}$ and $LSTM_{backward}$ use the same mathematical equations. However, each LSTM processes it in a different direction; hence, they learn different parameters. On each timestep (t), the output of both forward and backward LSTM is concatenated $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$ and passed to the GlobalAveragePooling1D layer. The global pooling layer converts the entire temporal sequence into a single vector. This layer takes the average of all hidden states in the sequence, reducing dimensionality. In other words, the global pooling layer determines the average of each feature or attribute over all time-steps. For instance, let us consider timesteps = 10 and LSTM unit size = 32. The output shape of the BiLSTM layer will be (None, 10, 64), and the output shape of the GlobalAveragePooling1D layer will be (None, 64). The overall temporal behavior captured by the global pooling layer is then transferred to the final dense layer.

The final dense layer then transforms the average temporal attributes into a dimension supported by a concatenated layer.

The proposed parallel structure reduces the dimensionality and hence the number of trainable parameters for the BiLSTM-CNN model. Therefore, the proposed model requires less computation, consumes less memory, and is less prone to overfitting. Additionally, it can make the proposed model simpler, faster, and suitable for real-time forecasting. The output of CNN (features) and BiLSTM (states) is concatenated and passed to multi-head attention. The multi-head attention decides which timesteps or attributes are crucial for PV power forecasting. The BiLSTM and CNN models treat the time series datasets equally; however, in a real-world scenario, not every timestep adds the same contribution to the final output. For instance, the temperature at midday on a peak sunny day may have a greater influence on PV power output compared to the temperature recorded at midnight. At this stage, the application of multi-head attention is vital. The multi-head attention assigns weights to each timestep in the fused features of the BiLSTM-CNN. For computing weights, the attention mechanism first calculates the attention scores using the scaled dot product method. The attention scores show the influence of each past timestep on the current prediction value. A high attention score indicates that the corresponding timestep provides meaningful information for prediction tasks. In contrast, a low attention score indicates that the corresponding timestep offers less important information for the current prediction value. Once the attention scores are determined, the softmax activation function is used to normalize attention scores. By doing this, the generated attention weights are ensured to be non-negative and sum to one. The normalized weights after applying a softmax function are used to produce a weighted sum of values, also known as the context vector. The context vector acts as a customized memory, focusing on the important timesteps rather than equally handling all inputs. In the multi-head attention, multiple heads work in parallel. Each head captures a different feature from the BiLSTM-CNN output. All these heads are then combined, representing a comprehensive fusion of various learned relationships. The output of the multi-head attention layer is passed to the GlobalAveragePooling1D layer, which converts the entire temporal sequence into a fixed-length vector. The multi-head attention and GlobalAveragePooling1D layers provide both focus and performance. The multi-head layer identifies the

important features, while the GlobalAveragePooling1D layer compresses this information into a compact representation. This compact representation is then passed to the Dense layer for PV power forecasting.

3.7 Summary

This chapter provides a detailed overview of the hybrid deep learning model. The proposed hybrid model integrates BiLSTM, CNN, and multi-head attention for PV power forecasting. BiLSTM is an advanced variant of RNNs that consists of two LSTMs; one operates in the forward direction while the other operates in the backward direction. The forward LSTM ($LSTM_{forward}$) encodes the sequence from all the preceding steps, and the backward LSTM ($LSTM_{backward}$) encodes the sequence from all future steps. The two-way flow enables the BiLSTM to learns temporal relationships and long-term dependencies from the PV dataset. Additionally, it provides a deeper contextual representation that describes the way environmental factors evolve, supplementing the local patterns captured by the CNN. CNN is another popular model that captures complex spatial relationships and local patterns from the PV dataset using sliding filters. It employs multiple filters to generate multiple feature maps, each focusing on a distinct feature of the PV data. These attributes make it highly effective for PV power prediction, where it is important to capture both weather-induced variations and more regular daily patterns. Additionally, these attributes are effective for detecting short-term fluctuations and changes that can instantly impact PV power output. Detecting these features can enhance the model's ability to understand the local interactions between multiple variables. Hence, combining it with BiLSTM can be a feasible choice for PV power forecasting. This combination can effectively capture both short-term local patterns and long-term temporal dependencies. It can also better explain the complex and nonlinear patterns in PV power datasets.

Over the years, various studies have employed the hybrid combination of BiLSTM and CNN for multiple fields. However, these studies used a sequential stacking of BiLSTM and CNN, which are computationally expensive. In sequential stacking, the first model processes the data first, followed by the second. The sequential stacking of these models results in slower computation and may not be

suitable for multi-task learning, therefore restricting real-time responses. Additionally, the sequential model performs multiple transformations, increasing the possibility of information degradation or loss of important aspects. To overcome these limitations, we proposed a hybrid deep learning model that employs a parallel model, which is computationally less expensive. In the proposed model, both BiLSTM and CNN analyze the input data independently. This structure is particularly effective for handling time series data because it allows the model to learn both short and long-term dependencies more effectively. Each model in this structure focuses on its unique abilities, enhancing accuracy and representational depth. Additionally, the parallel model can effectively use the GPUs' parallel computing capacity. It can execute multiple computations simultaneously, resulting in faster convergence, especially with extended sequences. Moreover, the parallel model performs minimal transformation on the input data stream. Each model in this architecture processes the input stream directly, reducing information degradation or loss of important aspects.

Another advantage of the proposed model is that it reduces the dimensionality and the number of trainable parameters for the BiLSTM-CNN model. The proposed model requires less computation, consumes less memory, and is less prone to overfitting. These features can make the proposed model simpler, faster, and suitable for real-time forecasting. Additionally, the proposed structure provides an efficient and reliable framework for combining multi-head attention layers. This is because a Flatten layer followed by a Dense layer in CNN and a global pooling layer followed by a Dense layer in BiLSTM reduce the dimensionality of both CNN and BiLSTM, respectively. Without reducing the dimensionality of both BiLSTM and CNN, passing to the multi-head attention layer will make the model computationally expensive, particularly for a high number of parameters (LSTM units, CNN filters). For instance, consider a hybrid model with three layers of both BiLSTM and CNN models. Each layer in the BiLSTM branch has 32 units, and each layer in the CNN branch has 32 filters. Without reducing dimensionality, the proposed model will require 668,256 parameters to train. On the other hand, reducing the dimensionality of both BiLSTM and CNN will require 70,768 trainable parameters. Hence, the proposed structure will require less time to train and will be computationally less expensive.

After concatenating the output of the CNN (features) and BiLSTM (states) using a fusion layer, it is passed through a multi-head attention. The multi-head attention decides which timesteps or attributes are crucial for PV power forecasting. The BiLSTM and CNN models treat the time series datasets equally; however, in a real-world scenario, not every timestep adds the same contribution to the final output. For instance, the temperature at midday on a peak sunny day may have a greater influence on PV power output compared to the temperature recorded at midnight. At this stage, the application of multi-head attention is vital. The multi-head attention assigns weights to each timestep in the fused features of the BiLSTM-CNN. For computing weights, the attention mechanism first calculates the attention scores using the scaled dot product method. The attention scores show the influence of each past timestep on the current prediction value. A high attention score indicates that the corresponding timestep provides meaningful information for prediction tasks. In contrast, a low attention score indicates that the corresponding timestep offers less important information for the current prediction value. Once the attention scores are determined, the softmax activation function is used to normalize attention scores. By doing this, the generated attention weights are ensured to be non-negative and sum to one. The normalized weights after applying a softmax function are used to produce a weighted sum of values, also known as the context vector. The context vector acts as a customized memory, focusing on the important timesteps rather than equally handling all inputs. In the multi-head attention, multiple heads work in parallel. Each head captures a different feature from the BiLSTM-CNN output. All these heads are then combined, representing a comprehensive fusion of various learned relationships. The output of the multi-head attention layer is passed to the GlobalAveragePooling1D layer, which converts the entire temporal sequence into a fixed-length vector. The multi-head attention and GlobalAveragePooling1D layers provide both focus and performance. The multi-head layer identifies the important features, while the GlobalAveragePooling1D layer compresses this information into a compact representation. This compact representation is then passed to the Dense layer for PV power forecasting.

4

Adaptive Harris Hawk Optimization Algorithm (ADHHO)

4.1 Nature-Inspired Metaheuristics: A Review

The term “optimization” indicates finding the best possible solution (minimum or maximum) for an intended objective function under certain constraints. It can be applied in multiple domains, including transportation (finding the optimal route), energy (improving efficiency), manufacturing (minimizing costs), machine learning (training networks), robotics (path planning), and others. Over the years, the importance of these optimization problems has significantly increased due to advances in technology. The growing integration of various fields, such as electrical engineering, energy, computer science, communication system, and artificial intelligence, has further increased the complexity of these optimization problems. Classical methods, which include Gradient Descent, the Gauss–Newton method, Linear Programming, the Levenberg–Marquardt method, and others, may fail to solve these complex optimization problems. These methods can provide high efficiency when the objective function is smooth and unimodal. However, today’s real-world problems are multimodal, non-differentiable, and nonlinear [119]. These methods often rely on strong assumptions, which may not give promising results in these complex scenarios. These methods may experience stagnation of local optima. For solving these complex problems, stochastic algorithms, such as heuristics and metaheuristics optimization algorithms, are preferred [120]. Heuristic means “to find” or “discover a solution by trial and error”. Heuristic algorithms provide high-quality solutions within a suitable time, although they may not be optimal. These algorithms are useful when we are looking for accessible solutions rather than the best ones. Hill Climbing, Greedy

Algorithm, and Dynamic Programming are some of the examples of heuristic algorithms [121]. Normally, these algorithms are developed using problem-specific knowledge. This problem-specific knowledge can lead them towards suitable sections of the search space and avoid exploring whole areas. However, these algorithms are problem-specific and may not be generalized to different optimization problems. They also do not guarantee finding an optimal solution and are prone to getting stuck in local optima, particularly in complex optimization problems.

Metaheuristic algorithms, another stochastic algorithm, are used to solve various optimization problems. Like heuristic algorithms, these algorithms can provide an acceptable solution within a reasonable time frame [120], though they have a high probability of avoiding local optima. They have high efficiency and can present robust solutions to noisy datasets. Their population-based design enables efficient parallelization, which improves computation for complex problems. Over time, multiple MOAs have been proposed, and based on their working principle, they are divided into four main groups. These include evolutionary algorithms, human-based algorithms, physics-based algorithms, and swarm-based algorithms [119]. Among these groups, swarm-based algorithms are widely researched due to their flexibility and high problem-solving abilities. These algorithms mimic the cooperative behavior of decentralized systems, such as fish schooling, foraging behavior, and insect swarms. Since swarm-based algorithms are population-based, they rely on many cooperative agents to explore the search space. Agents in the groups update their positions following algorithm-specific movement rules. For instance, in PSO, each particle updates its velocity toward its individual best and swarm best positions. The main advantage of swarm-based algorithms is that they maintain a balance between exploration and exploitation. This balance is normally adjusted using different parameters, including step size, velocity, and energy depletion. These algorithms are robust and adaptable, which makes them ideal for energy predictions, complex engineering design, deep learning parameter tuning, and various other real-world problems. Despite these advantages, most swarm-based algorithms are prone to getting stuck in local optima. They are sensitive to parameter settings; hence, they require precise parameter adjustments to achieve high-quality results. HHO, another popular swarm-based algorithm that gets inspiration from the cooperative and hunting

nature of the Harris hawks. This algorithm has multiple distinctive features compared to other prominent swarm intelligence-based algorithms. Firstly, this algorithm has a strong equilibrium between exploration and exploitation [94]. Secondly, this algorithm employs a time-varying method that improves with iterations, resulting in a smooth transition from exploration to exploitation. Thirdly, it exhibits a gradual convergence behavior that increases the possibility of finding the most favorable solutions. Finally, this algorithm's exploitation stage employs a greedy strategy to prioritize the best solutions over the low-quality ones, rendering it an appealing method [95].

Although HHO can be efficient for solving various optimization problems, due to its stochastic nature, the original HHO may still be prone to getting trapped in local optima for multi-dimensional problems. Another problem of HHO is the shift between exploration and exploitation, which is regulated by escape energy. This rigid control may cause exploitation to begin too late, resulting in slow convergence. Additionally, HHO's exploitation phase relies on random jumps, and inappropriate jumps near optimum solutions may result in overshoot. Thus, to achieve low final fitness values, better convergence rate, and higher efficiency, improvement in the exploration and exploitation phases is important. Multiple and advanced variants of HHO have been proposed to address the shortcomings of the original HHO, particularly its inclination for early convergence, slow convergence rate, and fixed exploration/exploitation control. These enhancements include introducing a nonlinear energy factor to improve the shift between exploration and exploitation phases and opposition-based learning to increase the convergence rate. Additionally, a self-adaptive method is included to increase diversity and leverage on already explored regions. Moreover, some studies used chaotic initialization and mutations to enhance diversity and improve exploration. Although the updated or improved versions have addressed certain features of the original HHO, they still have major shortcomings that restrict their applications in various optimization problems. The improvements using these variants are only useful in certain stages of the search process. For instance, chaotic versions may enhance early exploration yet struggle to maintain diversity in final iterations. Opposition-based learning increases the number of fitness evaluation, resulting in a high computational cost. Levy flight-improved HHO can lead to overshooting and may disturb fine-tuning close to global optima. Similarly,

multiple studies combine HHO with other advanced MOAs to enhance the solution accuracy of different research areas. However, hybrid methods can inherit the drawbacks of the second algorithm and can be computationally more expensive, particularly for multi-dimensional problems. Hence, it is vital to develop an adaptive-HHO for maintaining a balance between exploration and exploitation, enhancing convergence speed, and boosting robustness without increasing computational costs.

4.2 Mathematical Frameworks of the Original HHO

The HHO gets inspiration from the cooperative and hunting nature of the Harris hawks. Harris hawks, also called Dusky hawks, are intelligent birds in nature with unique hunting abilities. These birds hunt in cooperative family groups of two to six [122]. These birds typically perch on tall trees near their home, and when they find other family members, they gather at the same perch site. In groups, these birds leapfrog all over their home range, and this movement becomes more intense when they are vigorously looking for prey. Once they locate their prey, they employ various tactics, including tracing, encircling, and attacking [97]. In 2019 [123], mimicked these cooperative and hunting behaviors and designed a powerful metaheuristic algorithm called Harris Hawks Optimization. Heidari et al. divided the principal working of HHO into three phases: exploration, transition, and exploitation. Fig. 4.1 illustrates the details of these three phases of HHO. Initially, a population of individuals within boundary constraints is initialized using random numbers. Each individual in the population represents a candidate solution, whereas the best candidate solution is the intended prey (rabbits). An objective function, depending on the type of problem, is also initialized to evaluate the fitness of each individual in the population.

4.2.1 Exploration Phase

In the exploration phase, hawks randomly perch on a certain location and wait to find prey (rabbits). Here, hawks employ two different strategies, and each strategy has an equal chance. In the first strategy ($q_{rn} < 0.5$), the hawk perch depends on

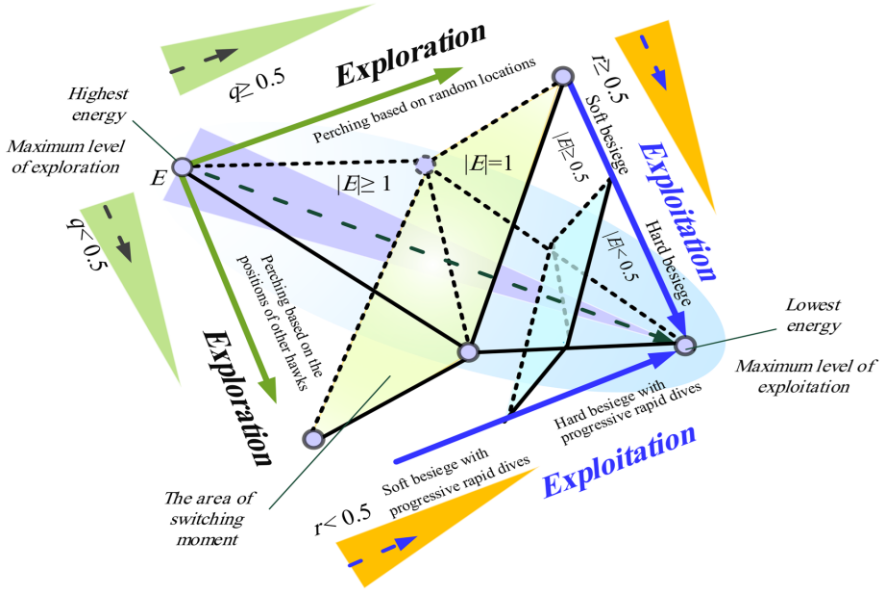


Figure 4.1: Description of the three phases of HHO [95]

the locations of other family members. In the second strategy, such as when $q_{rn} \geq 0.5$, the hawk chose a random position for detecting the intended prey. These two strategies are expressed in the equations below [123],

$$X(t+1) = \begin{cases} X_{rn}(t) - r_{rn1}|X_{rn}(t) - 2r_{rn2}X(t)|, & q_{rn} \geq 0.5 \\ \left(X_{rb}(t) - X_{mn}(t)\right) - r_{rn3}(L_{bs} + r_{rn4}(U_{bs} - L_{bs}), & q_{rn} < 0.5 \end{cases} \quad (4.1)$$

Where r_{rn1}, r_{rn2}, r_{rn3}, r_{rn4}, and q_{rn} are the random numbers in the range [0, 1]. $X(t)$ and $X(t+1)$ are the position vectors of hawks in the current and next iteration. $X_{rn}(t)$, $X_{rb}(t)$, and $X_{mn}(t)$ represent the position vector of a randomly chosen hawk, the rabbit position, and the mean position vector of the hawks, respectively. Similarly, U_{bs} and L_{bs} represent the upper and lower bounds of the search space. $X_{mn}(t)$ can be expressed using the equation below,

$$X_{mn}(t) = \frac{1}{S}\sum_{j=1}^{S} X_j(t) \quad (4.2)$$

Where S, and $X_j(t)$ represent all possible solutions and each hawk's position vector.

4.2.2 Transition Phase

The transition phase of HHO involves the shift from exploration to the exploitation phase. This transition is normally determined by the escape energy (E_{ep}) of the prey. The E_{ep} can be mathematically expressed as,

$$E_{ep} = 2E_o(1 - \frac{t}{T_m}) \tag{4.3}$$

In the above equation, E_o, $Iter$, and N_{iters} denote the prey's initial energy, current iteration, and maximum iterations, respectively. E_o is a random number in the range [-1, 1], and mathematically it can be expressed as,

$$E_o = 2 * rand() - 1 \tag{4.4}$$

When E_o rises from 0 to 1, the prey becomes strong, while when the value drops from 0 to -1, the prey becomes weak. In equation (3), the term $\frac{Iter}{N_{iters}}$ increases over iterations, hence E_{ep} decreases over iterations. When the absolute value of E_{ep} is greater than 1, the hawks continue exploring the search space. However, when the absolute value of E_{ep} is below 1, they start the exploitation phase or the hunting phase.

4.2.3 Exploitation Phase

During this phase, the hawks strike the prey that had been revealed in the prior phase, performing a surprise pounce. However, the prey displays a tendency to evade such situations. Hence, the hawks use different strategies to catch prey depending on the E_{ep}. In HHO, four different strategies are used to model the behavior of hawks during the hunting phase. Here, a new random number (r_{rn}) is used to describe the probability of prey escaping from dangerous situations. When $r_{rn} \geq 0.5$ the prey fails to escape, and hawks will perform soft or hard besieges. On the other hand, when $r_{rn} < 0.5$, the prey may successfully escape, and hawks will perform soft or hard besieges with progressively rapid dives. The hawks' besiege also depends on the escape energy of the prey. When the absolute value of $E_{ep} \geq 0.5$, the hawks will perform soft besieges, while $|E_{ep}| < 0.5$, the hawks will perform

hard besieges. Hence, two parameters are used to decide the soft and hard besieges: 1) escape energy of prey, and 2) probability of escaping.

Soft Besiege: When $r_{rn} \geq 0.5$ and $|E_{ep}| \geq 0.5$, the prey tends to escape, and the hawks softly encircle the prey, forcing it to lose more of its energy. The behavior of hawks during a soft besiege can be modeled as,

$$X(t+1) = X_{\Delta}(t) - E_{ep}|J_{sr}X_{rb}(t) - X(t)| \tag{4.5}$$

Where J_s is the prey's jumping strength and $X_{\Delta}(t)$ is the difference between the position vector of the prey and the hawk. The J_{sr} and $X_{\Delta}(t)$ can be expressed as,

$$J_{sr} = 2(1 - r_{rn5}) \tag{4.6}$$

$$X_{\Delta}(t) = X_{rb}(t) - X(t) \tag{4.7}$$

Where r_{rn5} is a random number in the range[0,1]. Due to r_{rn5}, the J_{sr} changes over each iteration to mimic the dynamic nature of prey jumping.

Hard Besiege: When $r_{rn} \geq 0.5$ and $|E_{ep}| < 0.5$, the prey has less energy and is exhausted after consuming much of its energy. In hard besieges, the hawks hardly surround the prey before performing the surprise pounce. The behavior of hawks during this stage can be modeled as,

$$X(t+1) = X_{rb}(t) - E_{ep}|X_{rb}(t) - X(t)| \tag{4.8}$$

Soft Besiege with Progressive Rapid Dives: When $r_{rn} < 0.5$ and $|E_{ep}| \geq 0.5$, the prey has sufficient energy and tends to escape. The hawks in this stage smartly encircle the prey and start gentle dives. The hawks' gentle and irregular dives made the prey perform a zigzag pattern, and to model these patterns, an L_f technique was introduced. The L_f can resemble the actual zigzag movement of the prey and the hawks' sudden dives toward the escaping prey. The equation below expresses the hawks' next move towards the prey.

$$Y = X_{rb}(t) - E_{ep}|J_{sr}X_{rb}(t) - X(t)| \tag{4.9}$$

Next, the hawks compare the potential outcome of this movement with the prior dive. If the result is not satisfactory, then they increase the rapid dives. These hawks' rapid dives can be modeled using the following equation,

$$Z = Y + S_{rn} \times L_f\ (D_i) \tag{4.10}$$

Where D_i and S_{rn} represent the problem's dimension and a random number of size $1 \times D_i$. The L_f can be calculated as,

$$L_f(x) = 0.01 \times \frac{u_{rn} \times \sigma}{|v_{rn}|^{\frac{1}{\beta}}} \tag{4.11}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}} \tag{4.12}$$

Where u_{rn} and v_{rn} are the random numbers inside[0,1] and β is fixed at 1.5. After including the Levy flight, the updated equation in soft besiege can be expressed as,

$$X(t+1) = \begin{cases} Y, & if\ f(Y) < f(X(t)) \\ Z, & if\ f(Z) < f(X(t)) \end{cases} \tag{4.13}$$

Hard Besiege with Progressive Rapid Dives: When $r_{rn} < 0.5$ and $|E_{ep}| < 0.5$, the prey does not have sufficient energy, and hawks perform a hard besiege. The hwks, in this case, attempt to shorten the gap with the intended prey before the surprise pounce. The equations below modeled the behavior of the hawks in the next iteration,

$$X(t+1) = \begin{cases} Y, & if\ f(Y) < f(X(t)) \\ Z, & if\ f(Z) < f(X(t)) \end{cases} \tag{4.14}$$

Where Y and Z can be calculated as,

$$Y = X_{rb}(t) - E_{ep}|J_{sr}X_{rb}(t) - X_{mn}(t)| \tag{4.15}$$

$$Z = Y + S_{rn} \times L_f\ (D_i) \tag{4.16}$$

Where $X_{mn}(t)$ is the mean position vector, which can be calculated from Eq. (2).

Pseudocode of HHO

Inputs: The P_{pop} and N_{iters}

Outputs: Location of Prey and its fitness value

Initialize the random population $X_j(j = 1, 2, 3 \ldots N)$

While $(Iter < N_{iters})$ **do**

  Evaluate the fitness of hawks using the objective function

  Set X_{rb} = the best prey position

  for each hawk X_j **do**

    Update the escape energy E_{ep} using Eq. (4.3) and (4.4)

    Update the jump strength J_{sr} using Eq.(4.6)

    If $|E_{ep}| \geq 1$ **then**

      Update hawk position using Eq.(4.1)

    end if

    If $|E_{ep}| < 1$ **then**

      If $r_{rn} \geq 0.5$ and $|E_{ep}| \geq 0.5$ **then**

        Update hawk position using Eq. (4.5) and (4.7)

      else If $r_{rn} \geq 0.5$ and $|E_{ep}| < 0.5$ **then**

        Update hawk position using Eq. (8)

      else If $r_{rn} < 0.5$ and $|E_{ep}| \geq 0.5$ **the**

        Update hawk position using Eq. (13)

      else If $r_{rn} < 0.5$ and $|E_{ep}| < 0.5$ **then**

        Update hawk position using Eq. (14)

      end if

    end if

  end for

end

return X_{rb}

4.3 Adaptive Harris Hawk Optimization (ADHHO)

The original HHO can provide superior performance, high efficiency, and efficient computation compared to sophisticated and advanced metaheuristic algorithms, as verified by various experimental results. However, this algorithm still has shortcomings and constraints, including slow and premature convergence. The

reason is that some approaches used are simple and may not truly represent the actual behavior of the hawks. For example, the transition between exploration and exploitation is linear, and it does not consider the problem's landscape. The original HHO does not employ any mechanism to adjust the conditions based on the hawks' fitness or performance. Additionally, the hawk's movement is largely dominated by randomness, and there is no guided movement. These shortcomings can limit the applications of original HHO in real-world optimization problems. Thus, to improve the performance and overcome these shortcomings, this study presents an ADaptive Harris Hawk Optimization (ADHHO). The proposed method improves the original HHO in four phases. Firstly, two chaotic maps are used, one for initializing the hawks' population and another for generating random numbers. Secondly, a nonlinear energy factor is used to maintain a balance between global and local search. Thirdly, an adaptive inertia weight mechanism is introduced to enhance the convergence rate and search stability. Lastly, an adaptive decay factor S_{rn} is used to regulate the steps as the algorithm converges. All these phases will be explained in this section.

4.3.1 Chaotic Optimizations

In the original HHO, the hawks' population is initialized using a random function, which may result in premature convergence and affect the algorithm's performance. Random initialization may not thoroughly explore the search space. Hence, to enhance global search capabilities, chaotic maps are used for population initialization in metaheuristic algorithms. They generate a sequence that can completely explore the search space with greater probability. This helps the metaheuristic algorithm to avoid local optimum and preserve the diversity of the population. Additionally, it can improve the convergence speed of the optimization algorithm. In this paper, two chaotic maps are used. A tent map is employed for the hawks' population initialization, while a logistic map is utilized to generate chaotic random numbers during the exploration phase. It is important to note that the initial value may have a major influence on the oscillatory pattern of chaotic maps [124]. A value between 0 and 1 is typically used as an initial value for most chaotic maps. The following equations represent the mathematical formulas for the tent and logistic maps.

$$X_{n+1}{}^{tent} = \begin{cases} \frac{X_n}{0.7}, & X_n < 0.7 \\ \frac{10}{3}(1 - X_n), & X_n \geq 0.7 \end{cases} \tag{4.17}$$

$$X_{n+1}{}^{log} = bX_n(1 - X_n) \tag{4.18}$$

Where $X_n \in (0,1)$ and b is the control parameter ranging from 0 to 4.

4.3.2 Nonlinear Control Parameter

The transition between exploration and exploitation is controlled by E_{ep}. The value of E_{ep} in the original HHO decreases linearly from 2 to 0. This linear version of E_{ep} may not fully explain the actual behavior of hawks. In addition, the linear energy factor may significantly shrink the search space [125] and result in inefficient search. Moreover, the linear version of E_{ep} may lead to over-exploration or exploitation, resulting in lower efficiency, particularly in complicated and multimodal optimization problems. Various studies replace the linear version of E_{ep} with nonlinear functions or adaptive control parameters [124], [125]. Fig. 4.2 presents a comparison between linear escape energy and function-based

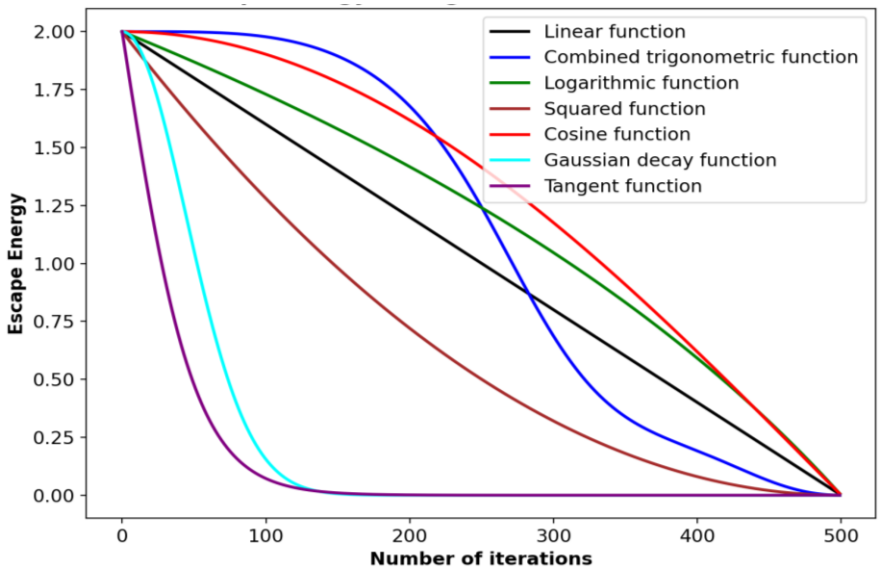


Figure 4.2: Comparison of escape energy using different nonlinear functions

nonlinear escape energy. In this paper, we used the squared function to illustrate the dynamics of escape energy. The squared function-based escape energy can be calculated as follows:

$$E_s = 2E_o\ (1 - \frac{Iter}{N_{iters}})^2 \tag{4.19}$$

Where $Iter$ and N_{iters} represent the current and maximum iterations. The time-dependent behavior of linear and proposed escape energy can be shown in Fig. 4.3. The figure shows that the proposed E_s can sustain a large value at the start for effective exploration. This value remains low for extended periods in the final phase to promote local exploitation.

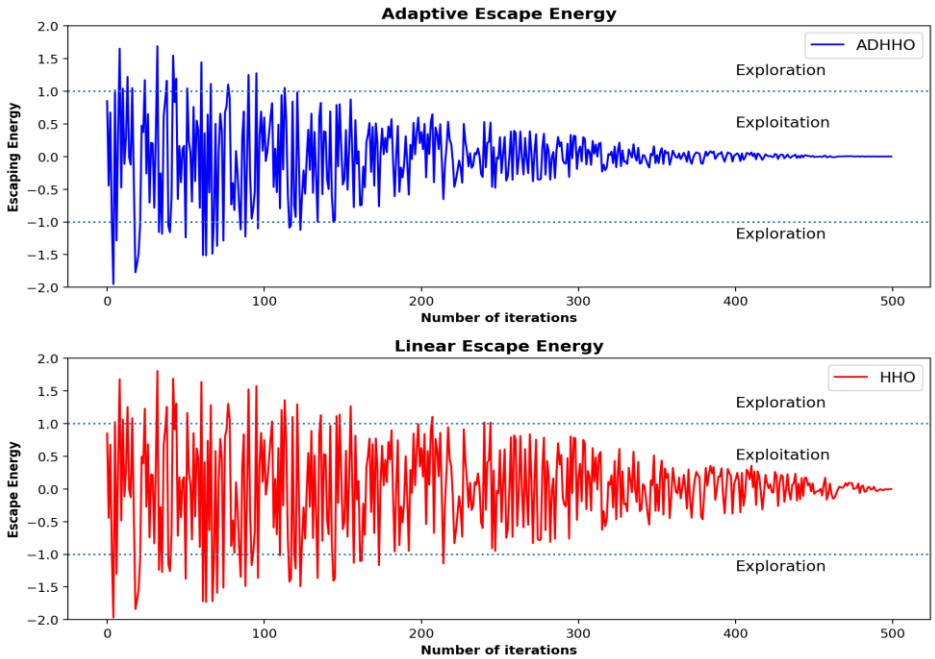


Figure 4.3: Behavior of escape energy during 500 iterations

4.3.3 Adaptive Inertia Weight

Motivated by the PSO velocity update strategy, this paper introduced an Adaptive Inertia Weight (AIW) mechanism in the original HHO. Inertia weight plays a crucial part in the velocity update equation of PSO. It maintains a balance between global exploration and local exploitation. In the ADHHO, the inertia weight can scale the entire movement of hawks in the direction of the best solution, which is

the position of the rabbit $X_{rb}(t)$. This movement is updated based on the fitness value of the hawks. In other words, the hawk's movement is influenced by the fitness factor (f_f). The f_f reflects the best position $X_{rb}(t)$ and the worst position of the hawk. Hence, the f_f facilitates the inertia weight to adapt according to the performance of the hawks. The f_f can be defined as,

$$f_f(t) = (f\big(X(t)\big) - f(X_{rb}(t)))/(f_{max}(t) - f(X_{rb}(t))) \quad (4.20)$$

$$f_{max}(t) = \max_{1 \le j \le M} f\left(X_j(t)\right) \quad (4.21)$$

Where $f(X_{rb}(t))$ and $f_{max}(t)$ the best fitness and the worst fitness of the hawk. Using the performance of the hawks, the AIW can be calculated as:

$$w_{ad} = w_{max} - (w_{max} - w_{min}) \times \sqrt{\frac{Iter}{N_{iters}}}.mean(f_f(t)) \quad (4.22)$$

$$mean(f_f(t)) = \frac{1}{M}\sum_{j=1}^{M} \frac{(f\left(X_j(t)\right) - f(X_{rb}(t)))}{(f_{max}(t) - f(X_{rb}(t)))} \quad (4.23)$$

In the above equation, w_{max} and w_{min} represent the maximum and minimum inertia weight. $mean(f_f(t))$ indicates the collective progress of the hawks. Using $mean(f_f(t))$ allows the hawks to move more unifiedly, preventing random individual adjustments.

4.3.4 Adaptive Decay Factor

The original HHO used a random number S_{rn} in the L_f of the exploitation phase, as mentioned in Eq.(10). This random number may impact the position update of the hawks. This is because both S_{rn} and L_f are random, and their results can be unconstrained randomness. Multiplying S_{rn}, when its value is close to 1, with L_f, particularly in the later phases of iterations, can lead to large displacement. This large displacement may result in overshoots or bypass better solutions. Even a small S_{rn} in the later phases can weaken the exploitation. To address this issue, a decay or damping factor (D_f)is introduced in the proposed ADHHO, which can be mathematically expressed as follows:

$$D_f = (1 - \frac{Iter}{N_{iters}})^2 \quad (4.24)$$

The decay factor is multiplied by the S_{rn}, which can facilitate adjusting the L_f dynamically over the iterations. This damping factor reduces the influence of randomness on the L_f, particularly in the final iterations when the algorithm is close to convergence. In the early iterations, D_f is close to 1, which allows for bigger jumps and promotes exploration. However, in the later iterations, D_f steadily decreases towards 0. These gradual decreases enable smaller steps, resulting in more targeted and focused updates. Hence, the D_f can reduce the possibilities of overshooting and improve convergence near optimum solutions.

4.4 ADHHO Algorithmic Workflow

The flowchart of the proposed ADHHO is shown in Fig. 4.4. The ADHHO takes six input parameters, including population size, maximum iterations, problem dimension, upper bounds, lower bounds, and objective function. The output parameters of the ADHHO are the best prey position, best fitness score, and convergence curve. The proposed ADHHO algorithm improved all three phases of HHO compared to other variants that improved either exploration or exploitation. In the exploration, chaotic maps are utilized to enhance the search range and avoid clustering. A tent chaotic map is used to initialize the hawk populations. This chaotic map uniformly distributes solutions over the search space. Similarly, a logistic chaotic map is used to generate random numbers in equation (4.1). Next is the transition phase, which is highly influenced by the value of escaping energy. For a smooth transition, a square-based function escape energy is used. The third phase (exploitation phase) is crucial for fine-tuning solutions, accelerating convergence, and ensuring excellent results. Weak exploitation can prevent HHO from reaching the global optimum, while strong exploitation may result in early convergence. As shown in the figure, at each iteration, the adaptive inertia weight, adaptive decay factor, and jumping factor are updated. Similarly, to improve the movement of hawks toward the global best position or prey location, the adaptive inertia weight also considers the best and worst performance of the hawks. With this guided movement, the hawk can easily find the best location and can avoid getting stuck in local minima. Hence, in the exploitation phase, the adaptive

inertia weight and adaptive decay factors are used with L_f to take guided steps towards the prey locations.

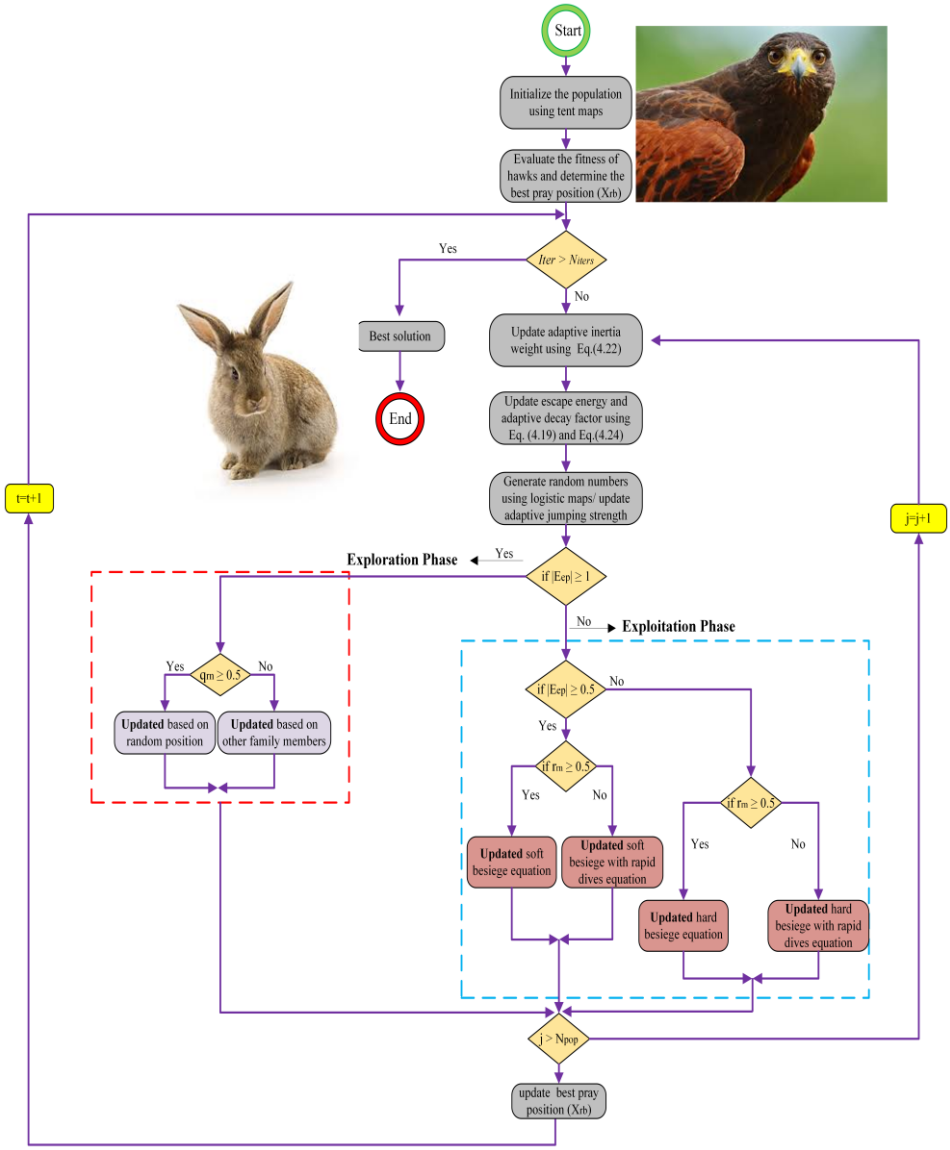


Figure 4.4: Flowchart of the ADHHO algorithm technique

4.5 Convergence Characteristics on IEEE CEC Benchmark Suites

To assess the performance and convergence characteristics of the proposed ADHHO, this section presents a comprehensive evaluation of experimental findings. Multiple well-known benchmark suites from the IEEE CEC 2014 and 2017 are employed for validation purposes. The performance results on these benchmark functions are then assessed against prominent metaheuristic algorithms, including PSO, GA, GWO, WOA, and HHO. Tab 4.1 presents the detailed descriptions of these algorithms. These six algorithms are separately evaluated on each benchmark function, and their results are recorded. For a fair comparison, these algorithms employed the same population size (50) and number of iterations (500). For coding and experimentation, Jupyter Notebook with Python and TensorFlow versions 3.10.0 and 2.10.0 is employed. These experiments are performed on a computer that has an Intel (R) Core (TM) i9-13900HX CPU@2.20GHz, 32GB of RAM, and a 64-bit operating system.

Algorithms	Parameters	Values
PSO	c_1	1.5
	c_2	1.5
	w_{max}	0.9
	w_{min}	0.4
GA	Mutation rate	0.3
	Crossover rate	0.8
	Tournament size	3
	Elitism rate	0.05
GWO	A	[2 0]
WOA	A	[2 0]
HHO	β	1.5
	E_e	[2 0]
ADHHO	β	1.5
	B	4
	X_n	0.7
	w_{max}	0.9
	w_{min}	0.3

Table 4.1: Initial parameter settings of all tested algorithms

4.5.1 Convergence Curves on CEC 2014 Benchmark Suite

The CEC 2014 benchmark suite is widely used for evaluating the performance and reliability of optimization algorithms [127]. These are single-objective optimization problems, and the level of complexity rises with the increase in the

number of dimensions. Thus, this benchmark suite helps assess new or improved optimization algorithms. The CEC 2014 is primarily divided into four main groups, including unimodal, multimodal, hybrid, and composite. For our analysis, we selected 22 benchmark functions: unimodal (F1-F3), multimodal (F4-F16), and hybrid (F17-F22) functions. The complete details of these 22 benchmark functions, including the type, dimension, range, and optimal values, are listed in Tab. 4.2. The CEC 2014 functions are highly complex and multidimensional; hence, the final score of each benchmark function is calculated based on 10 runs. Additionally, the same random seed is utilized for all tested algorithms. These benchmark functions are minimization problems; consequently, the algorithm that achieves the lowest score will be more efficient. We performed two experiments on the CEC 2014 benchmark functions. Initially, we validated the improvement in ADHHO by comparing the results with the original HHO. In the second phase, we compared the results of ADHHO with all five studied algorithms, namely PSO, GA, GWO, WOA, and HHO, to rank the best algorithm.

For the first experiments, Tab. 4.3 presents the experimental results for ADHHO and HHO on the CEC 2014 benchmark suite. For in-depth analysis, four important metrics are used, and for each metric, the best score is presented in bold. The findings of this experiment show that the ADHHO produced the most promising results on most of the CEC2014 test functions. The adaptive-HHO beat the original HHO on all four key metrics except for the two standard deviations, F9 and F14. Both ADHHO and HHO achieved similar results on only three test functions (F4, F5, and F20). Overall, ADHHO produced the best scores on 86 out of 88 performance metrics, while HHO achieved the best scores on 19 out of 88 metrics. In the second experiment, the performance and the convergence curves are compared with all five studied algorithms. All these algorithms are evaluated on two key metrics: mean and standard deviation. The outcomes of all algorithms on these two metrics are presented in Tab. 4.4 and Fig. 4.5. Tab. 4.4 shows that ADHHO once again performs better than all other competing algorithms. The adaptive-HHO produced the low mean value for all test functions. Additionally, ADHHO beats all competing algorithms on standard deviations except for functions F9, F10, F12, and F14. Among the other algorithms, HHO shows a reasonable performance on multiple benchmark functions. Overall, the ADHHO obtained the best score on 40 out of 44 metric values, followed by HHO (9), WOA

(5), GA (3), GWO (1), and PSO (0). Similarly, in terms of convergence curves, listed in Fig. 4.5, GWO converges faster in the initial iterations. However, its convergence curves remained flat for the rest of the iterations without improving. In contrast, the proposed ADHHO achieved the lowest fitness values on all test functions at the end of iterations. This indicates that the adaptive version can locate superior and near-optimal solutions more effectively. Additionally, the proposed algorithm can exhibit superior exploitation capabilities compared to other competing algorithms. Therefore, the proposed adaptive HHO has demonstrated promising potential in addressing complex optimization problems.

Type	Fun.	Description	Dim.	Range	Optimal
Unimodal	F1	Rotate High-Conditioned Elliptic	30	[-100 100]	100
	F2	Rotated Bent Cigar	30	[-100 100]	200
	F3	Rotated Discus	30	[-100 100]	300
Multimodal	F4	Shifted and Rotated Rosenbrock's	30	[-100 100]	400
	F5	Shifted and Rotated ACKLEY's	30	[-100 100]	500
	F6	Shifted and Rotated Weierstrass's	30	[-100 100]	600
	F7	Shifted and Rotated Griewank's	30	[-100 100]	700
	F8	Shifted Rastrigin's	30	[-100 100]	800
	F9	Shifted and Rotated Rastrigin's	30	[-100 100]	900
	F10	Shifted Schwefel's	30	[-100 100]	1000
	F11	Shifted and Rotated Schwefel's	30	[-100 100]	1100
	F12	Shifted and Rotated Katsure	30	[-100 100]	1200
	F13	Shifted and Rotated HappyCat	30	[-100 100]	1300
	F14	Shifted and Rotated HGBat	30	[-100 100]	1400
	F15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's	30	[-100 100]	1500
	F16	Shifted and Rotated Expanded Scaffer's	30	[-100 100]	1600
Hybrid	F17	Hybrid Function 1 (N = 3)	30	[-100 100]	1700
	F18	Hybrid Function 2 (N = 3)	30	[-100 100]	1800
	F19	Hybrid Function 3 (N = 4)	30	[-100 100]	1900
	F20	Hybrid Function 4 (N = 4)	30	[-100 100]	2000
	F21	Hybrid Function 5 (N = 5)	30	[-100 100]	2100
	F22	Hybrid Function 6 (N = 5)	30	[-100 100]	2200

Table 4.2: Description of the CEC2014 benchmark Suite

Fun.	Optimizers	MEAN	STD	MIN	MAX
F1	ADHHO	**3.77×10^{04}**	**1.43×10^{04}**	**1.50×10^{04}**	**7.34×10^{04}**
	HHO	2.74×10^{05}	$9.60E \times 10^{04}$	1.57×10^{05}	4.31×10^{05}
F2	ADHHO	**1.61×10^{07}**	**3.64×10^{07}**	**3.98×10^{04}**	**1.25×10^{08}**
	HHO	1.57×10^{08}	1.03×10^{08}	6.31×10^{07}	3.49×10^{08}

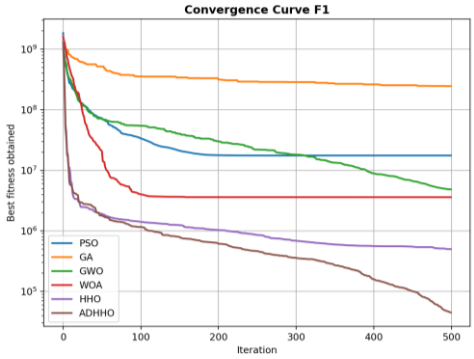
F3	ADHHO	**3.00 × 10^02**	**2.17 × 10^-02**	**3.00 × 10^02**	**3.00 × 10^02**
	HHO	3.05 × 10^02	5.84 × 10^-01	3.04 × 10^02	3.06 × 10^02
F4	ADHHO	**4.00 × 10^02**	**0.00 × 10^0**	**4.00 × 10^02**	**4.00 × 10^02**
	HHO	**4.00 × 10^02**	**0.00 × 10^0**	**4.00 × 10^02**	**4.00 × 10^02**
F5	ADHHO	**5.00 × 10^02**	**0.00 × 10^0**	**5.00 × 10^02**	**5.00 × 10^02**
	HHO	**5.00 × 10^02**	**0.00 × 10^0**	**5.00 × 10^02**	**5.00 × 10^02**
F6	ADHHO	**6.02 × 10^02**	**9.59 × 10^-01**	**6.01 × 10^02**	**6.03 × 10^02**
	HHO	6.09 × 10^02	2.47 × 10^0	6.04 × 10^02	6.13 × 10^02
F7	ADHHO	**7.00 × 10^02**	**8.01 × 10^-03**	**7.00 × 10^02**	**7.00 × 10^02**
	HHO	7.01 × 10^02	6.87 × 10^-02	7.01 × 10^02	7.01 × 10^02
F8	ADHHO	**1.07 × 10^03**	**5.54 × 10^01**	**9.33 × 10^02**	**1.15 × 10^03**
	HHO	1.96 × 10^03	3.72 × 10^02	1.57 × 10^03	2.85 × 10^03
F9	ADHHO	**9.06 × 10^02**	2.76 × 10^-01	**9.05 × 10^02**	**9.06 × 10^02**
	HHO	9.07 × 10^02	**1.71 × 10^-01**	9.06 × 10^02	9.07 × 10^02
F10	ADHHO	**1.09 × 10^04**	**1.28 × 10^02**	**1.08 × 10^04**	**1.12 × 10^04**
	HHO	1.11 × 10^04	2.49 × 10^02	**1.08 × 10^04**	1.18 × 10^04
F11	ADHHO	**1.18 × 10^04**	**6.17 × 10^-03**	**1.18 × 10^04**	**1.18 × 10^04**
	HHO	**1.18 × 10^04**	1.88 × 10^-02	**1.18 × 10^04**	**1.18** × 10^04
F12	ADHHO	**1.201 × 10^03**	**5.243 × 10^-01**	**1.200 × 10^03**	**1.202 × 10^03**
	HHO	1.202 × 10^03	6.015 × 10^-01	1.201 × 10^03	1.203 × 10^03
F13	ADHHO	**1.301 × 10^03**	**1.071 × 10^-01**	**1.301 × 10^03**	**1.301 × 10^03**
	HHO	1.305 × 10^03	1.113 × 10^0	1.303 × 10^03	1.308 × 10^03
F14	ADHHO	**1.4010×10^03**	9.069×10^-02	**1.4009×10^03**	**1.4012×10^03**
	HHO	1.4011×10^03	**8.545×10^-02**	1.4010×10^03	**1.4012×10^03**
F15	ADHHO	**2.24 × 10^04**	**4.11 × 10^04**	**1.51 × 10^03**	**1.42 × 10^05**
	HHO	3.16 × 10^06	3.02 × 10^06	2.51 × 10^04	9.48 × 10^06
F16	ADHHO	**1.88 × 10^03**	**8.06 × 10^01**	**1.79 × 10^04**	**2.05 × 10^03**
	HHO	1.96 × 10^03	9.32 × 10^01	1.84 × 10^03	2.13 × 10^03
F17	ADHHO	**1.75 × 10^03**	**1.29 × 10^01**	**1.73 × 10^03**	**1.77 × 10^03**
	HHO	1.80 × 10^03	2.45 × 10^01	1.77 × 10^03	1.84 × 10^03
F18	ADHHO	**1.84 × 10^03**	**1.13 × 10^01**	**1.82 × 10^03**	**1.86 × 10^03**
	HHO	1.90 × 10^03	2.50 × 10^01	1.86 × 10^03	1.95 × 10^03
F19	ADHHO	**1.95 × 10^03**	**6.56 × 10^0**	**1.94 × 10^03**	**1.96 × 10^03**
	HHO	1.98 × 10^03	1.97 × 10^01	1.96 × 10^03	2.02 × 10^03
F20	ADHHO	**2.00 × 10^03**	**0.00 × 10^0**	**2.00 × 10^03**	**2.00 × 10^03**
	HHO	**2.00 × 10^03**	**0.00 × 10^0**	**2.00 × 10^03**	**2.00 × 10^03**
F21	ADHHO	**2.116×10^03**	**7.703×10^-01**	**2.115×10^03**	**2.117×10^03**
	HHO	2.118×10^03	8.592×10^-01	2.116×10^03	2.119×10^03
F22	ADHHO	**2.25 × 10^03**	**8.05 × 10^0**	**2.24 × 10^03**	**2.26 × 10^03**
	HHO	2.30 × 10^03	2.71 × 10^01	2.28 × 10^03	2.37 × 10^03

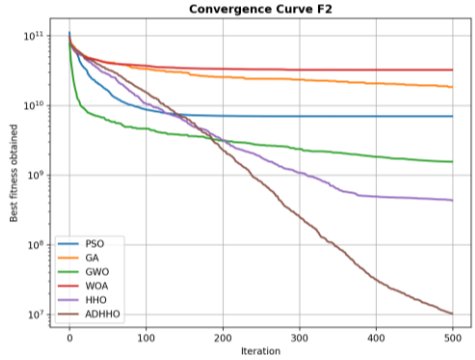
Table 4.3: Performance analysis of ADHHO and HHO on the CEC2014 benchmark suite

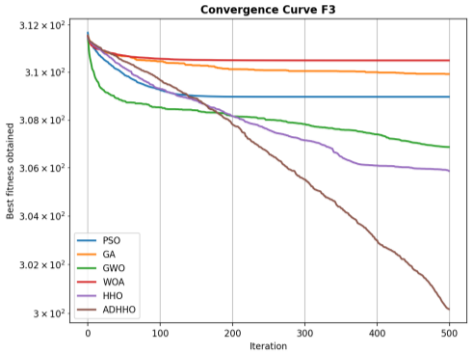
Fun.	Metric	ADHHO	HHO	GWO	PSO	GA	WOA
F1	MEAN	**3.77 × 10^04**	2.74 × 10^05	2.34 × 10^06	1.04 × 10^07	2.08 × 10^08	2.17 × 10^06

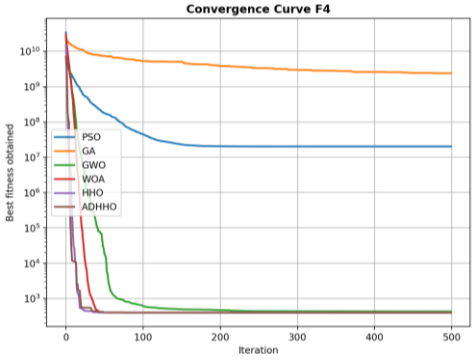
F2	STD	**1.43 × 10^04**	9.60 × 10^04	1.98 × 10^06	7.62 × 10^06	3.51 × 10^07	1.66 × 10^06
	MEAN	**1.61 × 10^07**	1.57 × 10^08	5.92 × 10^08	4.60 × 10^09	1.34 × 10^10	2.81 × 10^10
F3	STD	**3.64 × 10^07**	1.03 × 10^08	6.46 × 10^08	2.29 × 10^09	4.92 × 10^09	1.06 × 10^10
	MEAN	**3.00 × 10^02**	3.05 × 10^02	3.06 × 10^02	3.08 × 10^02	3.09 × 10^02	3.09 × 10^02
F4	STD	**2.17 × 10^-02**	5.84 × 10^-01	2.19 × 10^0	8.25 × 10^-01	2.95 × 10^0	3.11 × 10^0
	MEAN	**4.00 × 10^02**	**4.00×10^02**	4.37 × 10^02	7.06 × 10^05	1.63 × 10^09	**4.00×10^02**
F5	STD	**0.00 × 10^0**	**0.00×10^0**	6.39 × 10^01	5.64 × 10^05	6.16 × 10^08	**0.00×10^0**
	MEAN	**5.00 × 10^02**	**5.00×10^02**	5.21 × 10^02	5.21 × 10^02	5.21 × 10^02	**5.00×10^02**
F6	STD	**0.00 × 10^0**	**0.00×10^0**	7.08 × 10^-02	2.43 × 10^-01	4.70 × 10^-02	4.02 × 10^-14
	MEAN	**6.02 × 10^02**	6.09 × 10^02	6.16 × 10^02	6.10 × 10^02	6.16 × 10^02	6.14 × 10^02
F7	STD	**9.59 × 10^-01**	2.47 × 10^0	5.55 × 10^0	2.74 × 10^0	5.50 × 10^0	4.96 × 10^0
	MEAN	**7.00 × 10^02**	7.01 × 10^02	7.01 × 10^02	7.02 × 10^02	7.05 × 10^02	7.08 × 10^02
F8	STD	**8.01 × 10^-03**	6.87 × 10^-02	4.33 × 10^-01	6.06 × 10^-01	1.80 × 10^0	2.91 × 10^0
	MEAN	**1.07 × 10^03**	1.96 × 10^03	2.27 × 10^03	5.32 × 10^03	1.77 × 10^04	3.40 × 10^04
F9	STD	**5.54 × 10^01**	3.72 × 10^02	1.04 × 10^03	1.98 × 10^03	6.15 × 10^03	1.16 × 10^04
	MEAN	**9.06 × 10^02**	9.07 × 10^02	9.07 × 10^02	9.08 × 10^02	9.10 × 10^02	9.09 × 10^02
F10	STD	2.76 × 10^-01	**1.71 × 10^-01**	4.97 × 10^-01	3.09 × 10^-01	**1.71×10^-01**	7.45 × 10^-01
	MEAN	**1.09 × 10^04**	1.11 × 10^04	1.14 × 10^04	1.20 × 10^04	1.16 × 10^04	1.21 × 10^04
F11	STD	1.28 × 10^02	2.49 × 10^02	1.99 × 10^02	1.15 × 10^02	**7.52×10^01**	1.80 × 10^02
	MEAN	**1.18 × 10^04**	**1.18 × 10^04**	1.25 × 10^04	1.24 × 10^04	1.23 × 10^04	**1.18×10^04**
F12	STD	**6.17 × 10^-03**	8.78 × 10^-03	1.41 × 10^02	1.20 × 10^02	6.93 × 10^01	5.95 × 10^0
	MEAN	**1.201×10^03**	1.202×10^03	1.203×10^03	1.202×10^03	1.203×10^03	1.202×10^03
F13	STD	5.24×10^-01	6.01×10^-01	8.73×10^-01	3.85×10^-01	**1.87×10^-01**	5.10×10^-01
	MEAN	**1.301×10^03**	1.305×10^03	1.322×10^03	1.382×10^03	1.587×10^03	1.768×10^03
F14	STD	**1.071×10^-01**	1.113×10^0	1.606×10^01	5.466×10^01	1.023×10^02	1.860×10^02
	MEAN	**1.4010×10^03**	1.4011×10^03	1.4027×10^03	1.7113×10^03	1.5051×10^04	1.5302×10^03
F15	STD	9.07 × 10^-02	**8.54×10^-02**	3.20 × 10^0	1.49 × 10^02	2.16 × 10^03	2.05 × 10^01
	MEAN	**2.24 × 10^04**	3.16 × 10^06	1.18 × 10^07	2.17 × 10^08	2.73 × 10^09	1.08 × 10^10
F16	STD	**4.11 × 10^04**	3.02 × 10^06	1.46 × 10^07	3.41 × 10^08	1.30 × 10^09	5.64 × 10^09
	MEAN	**1.88 × 10^03**	1.96 × 10^03	2.13 × 10^03	2.01 × 10^03	2.30 × 10^03	2.19 × 10^03
F17	STD	**8.06 × 10^01**	9.32 × 10^01	3.00 × 10^02	9.15 × 10^01	2.42 × 10^02	2.04 × 10^02
	MEAN	**1.75 × 10^03**	1.80 × 10^03	1.81 × 10^03	2.04 × 10^03	1.88 × 10^03	2.69 × 10^03
F18	STD	**1.29 × 10^01**	2.45 × 10^01	1.45 × 10^02	3.59 × 10^02	4.32 × 10^01	6.78 × 10^02
	MEAN	**1.84 × 10^03**	1.90 × 10^03	1.86 × 10^03	2.27 × 10^03	1.92 × 10^03	2.78 × 10^03
F19	STD	**1.13 × 10^01**	2.50 × 10^01	1.45 × 10^01	3.51 × 10^02	5.38 × 10^01	6.66 × 10^02
	MEAN	**1.95 × 10^03**	1.98 × 10^03	**1.95×10^03**	2.12 × 10^03	1.98 × 10^03	2.21 × 10^03
F20	STD	**6.56 × 10^0**	1.97 × 10^01	1.97 × 10^01	3.10 × 10^02	3.14 × 10^01	3.91 × 10^02
	MEAN	**2.00 × 10^03**	**2.00×10^03**	2.02 × 10^03	2.05 × 10^03	2.06 × 10^03	**2.00×10^03**
F21	STD	**0.00 × 10^0**	**0.00 × 10^0**	1.20 × 10^01	9.83 × 10^0	1.21 × 10^01	7.19 × 10^-14
	MEAN	**2.116×10^03**	2.118×10^03	2.126×10^03	1.424×10^05	1.109×10^04	2.131×10^03
F22	STD	**7.70 × 10^-01**	8.59 × 10^-01	7.39 × 10^0	3.03 × 10^05	3.50 × 10^03	1.18 × 10^01
	MEAN	**2.25 × 10^03**	2.30 × 10^03	2.25 × 10^03	3.45 × 10^03	2.31 × 10^03	3.61 × 10^03
	STD	**8.05 × 10^0**	2.71 × 10^01	1.83 × 10^01	1.81 × 10^03	4.03 × 10^01	1.31 × 10^03

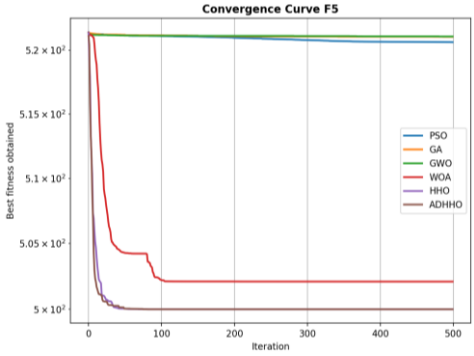
Table 4.4: Performance analysis of ADHHO and other algorithms on the CEC2014 test functions

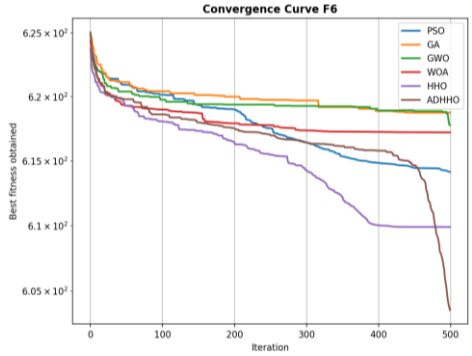

Convergence Curve F1
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

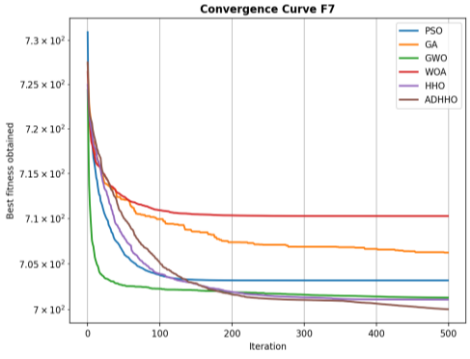

Convergence Curve F2
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

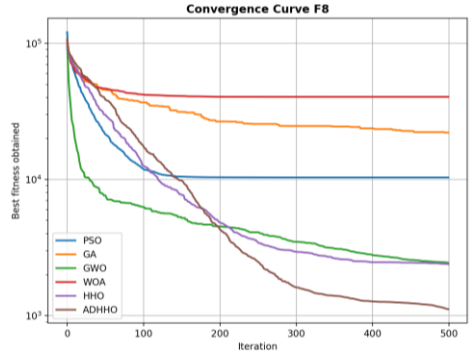

Convergence Curve F3
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

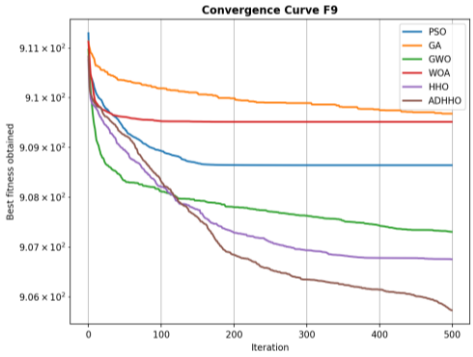

Convergence Curve F4
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

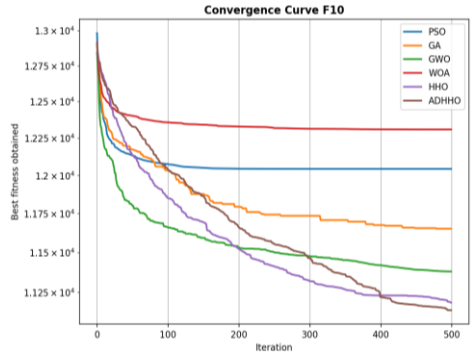

Convergence Curve F5
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

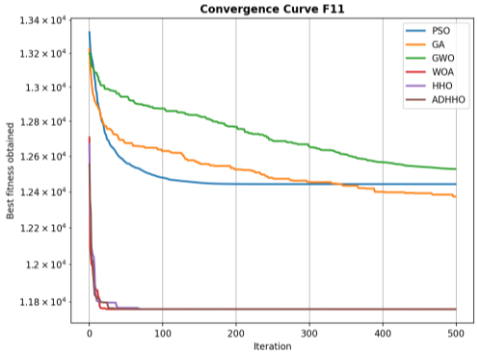

Convergence Curve F6
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

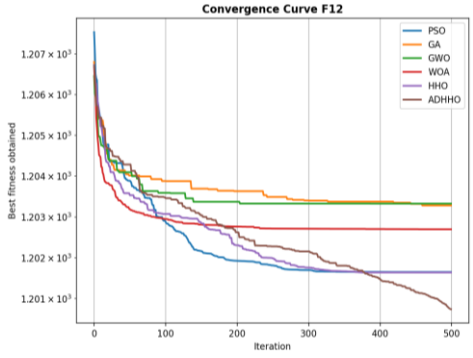

Convergence Curve F7
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

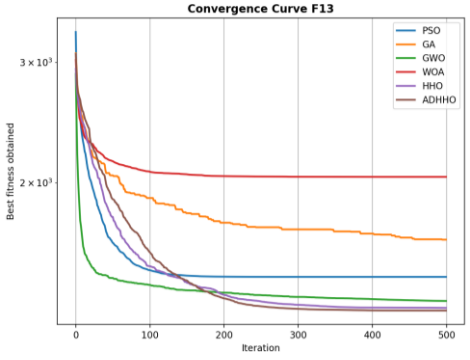

Convergence Curve F8
Best fitness obtained
Iteration
PSO
GA
GWO
WOA
HHO
ADHHO

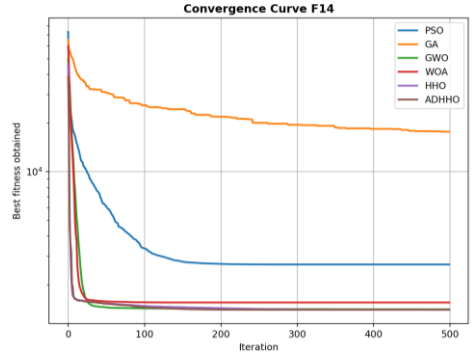

Convergence Curve F9
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration

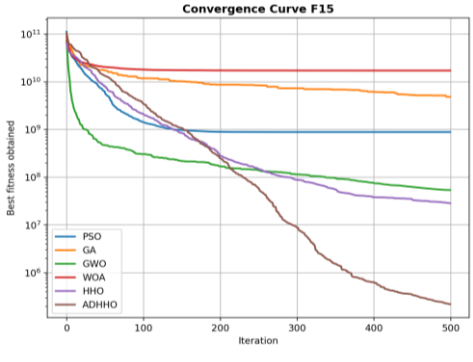

Convergence Curve F10
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration

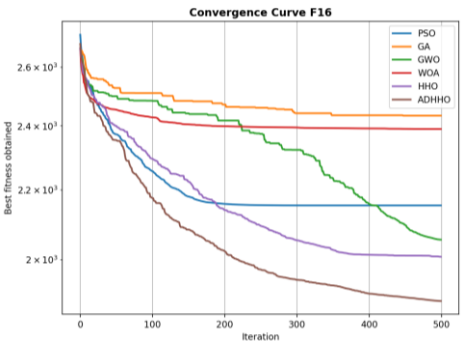

Convergence Curve F11
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F12
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F13
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F14
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F15
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F16
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration

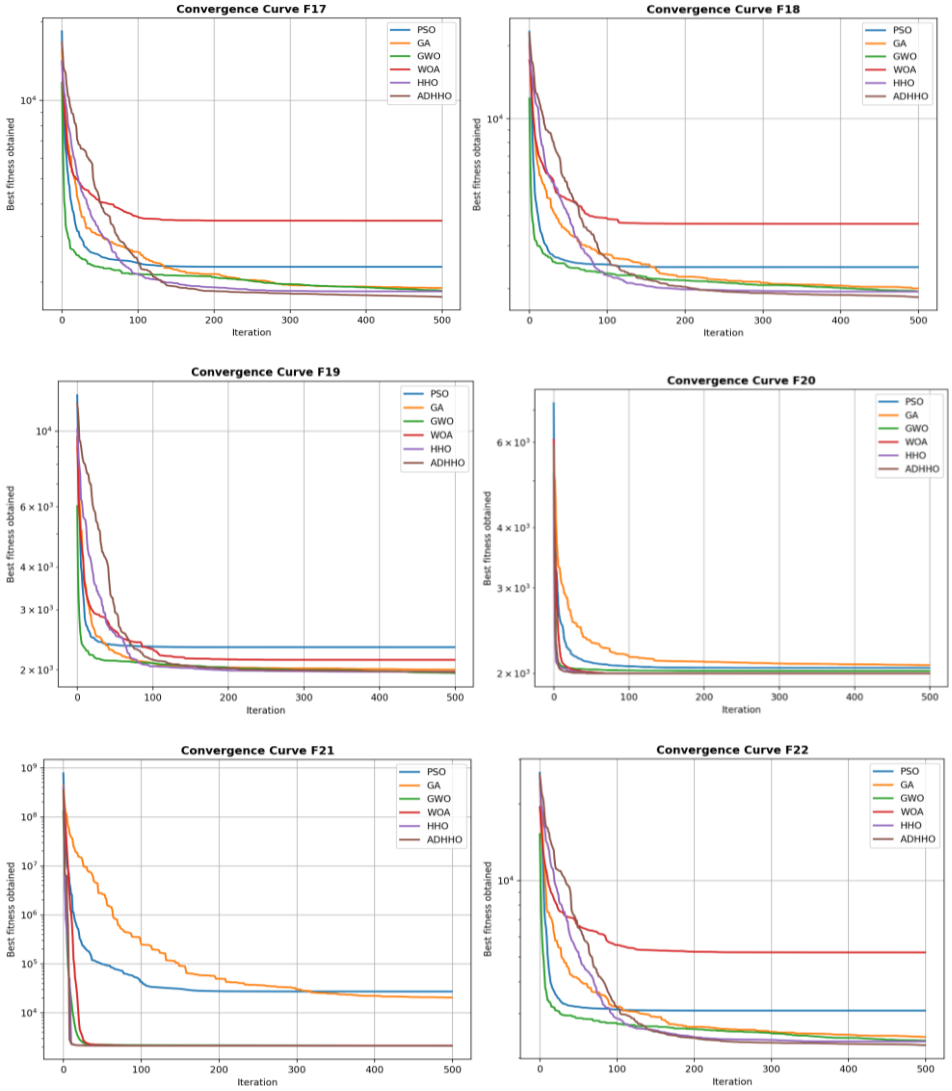


Figure 4.5: Convergence curve for ADHHO and other evaluated algorithms on CEC2014

4.5.2 Convergence Curves on CEC 2017 Benchmark Suite

The performance of the adaptive-HHO is also tested on the CEC2017 benchmark suite. This benchmark suite is commonly used in the literature to assess the optimization capacity of new or updated algorithms. The CEC 2017 functions offer multi-dimensional and multimodal landscapes for testing algorithms. This benchmark suite also includes hybrid and composite functions with shifted and rotated settings. These functions are useful for evaluating the adaptation

capability of a new algorithm to diverse environments. Hence, CEC 2017 can present challenging settings for testing algorithms. The CEC 2017 benchmark suite is normally divided into four main groups: unimodal, multimodal, hybrid, and composition. However, for our experiments, we selected 20 functions from the first three groups: unimodal (F1-F3), multimodal (F4-F10), and hybrid (F11-F20). The detailed descriptions of these functions are presented in Tab. 4.5. The dimensions for all functions are fixed at 30, and the theoretical minimum varies from 100 to 2000. Similar to the CEC 2014 benchmark suite, the CEC 2017 benchmark suite is also used to validate the improvement in HHO and compare its performance with that of other well-known algorithms. Initially, the performance of ADHHO and HHO is assessed using four metrics. Tab. 4.6 presents the comparison of both ADHHO and HHO on all test functions. The table shows that adaptive-HHO outperforms original HHO on all test functions and achieves the best scores on three key metrics (mean, minimum, and maximum). On one function, F5, both adaptive-HHO and original HHO produced the same results for all four metrics. Similarly, ADHHO achieved the most promising results on all standard deviations, except for two functions: F4 and F6, where the HHO outperformed ADHHO. Overall, ADHHO performed well on 78 out of 80 metric values, while HHO achieved the best scores on 7 out of 80 metric values.

In the second experiment, the performance and convergence curves of ADHHO and five competing algorithms are assessed on 20 test functions. Two performance metrics, mean and standard deviation, are used, and the results of the analysis are shown in Tab. 4.7. The analysis results indicated the superiority of the ADHHO. The adaptive-HHO achieved the lowest mean values on 19 test

Type	Fun.	Description	Dim.	Range	Optimal
Unimodal	F1	Shifted and Rotated Bent Cigar	30	[-100 100]	100
	F2	Shifted and Rotated the Sum of Different Power	30	[-100 100]	200
	F3	Shifted and Rotated Zakharov	30	[-100 100]	300
Multimodal	F4	Shifted and Rotated Rosenbrock's	30	[-100 100]	400
	F5	Shifted and Rotated Rastrigin's	30	[-100 100]	500
	F6	Shifted and Rotated Expanded Scaffer's	30	[-100 100]	600
	F7	Shifted and Rotated Lunacek Bi_Rastrigin	30	[-100 100]	700

Hybrid	F8	Shifted and Rotated Non-Continuous Rastrigin's	30	[-100 100]	800
	F9	Shifted and Rotated Levy	30	[-100 100]	900
	F10	Shifted and Rotated Schwefel's	30	[-100 100]	1000
	F11	Hybrid Function 1 (N = 3)	30	[-100 100]	1100
	F12	Hybrid Function 2 (N = 3)	30	[-100 100]	1200
	F13	Hybrid Function 3 (N = 3)	30	[-100 100]	1300
	F14	Hybrid Function 4 (N = 4)	30	[-100 100]	1400
	F15	Hybrid Function 5 (N = 4)	30	[-100 100]	1500
	F16	Hybrid Function 6 (N = 4)	30	[-100 100]	1600
	F17	Hybrid Function 6 (N = 5)	30	[-100 100]	1700
	F18	Hybrid Function 6 (N = 5)	30	[-100 100]	1800
	F19	Hybrid Function 6 (N = 5)	30	[-100 100]	1900
	F20	Hybrid Function 6 (N = 6)	30	[-100 100]	2000

Table 4.5: Description of the CEC2017 benchmark suite

Fun.	Optimizers	MEAN	STD	MIN	MAX
F1	ADHHO	**2.20×10^{04}**	**6.19×10^{03}**	**9.38×10^{03}**	**3.33×10^{04}**
	HHO	1.73×10^{07}	9.74×10^{06}	7.10×10^{06}	4.43×10^{07}
F2	ADHHO	**2.00×10^{02}**	**5.11×10^{-06}**	**2.00×10^{02}**	**2.00×10^{02}**
	HHO	2.00×10^{02}	8.54×10^{-04}	2.00×10^{02}	2.00×10^{02}
F3	ADHHO	**8.73×10^{02}**	**1.72×10^{02}**	**6.33×10^{02}**	**1.19×10^{03}**
	HHO	2.71×10^{03}	5.33×10^{02}	1.50×10^{03}	3.54×10^{03}
F4	ADHHO	**4.277×10^{02}**	4.036×10^{-01}	**4.272×10^{02}**	**4.288×10^{02}**
	HHO	4.286×10^{02}	**3.220×10^{-01}**	4.280×10^{02}	4.292×10^{02}
F5	ADHHO	**5.00×10^{02}**	**0.00×10^{0}**	**5.00×10^{02}**	**5.00×10^{02}**
	HHO	**5.00×10^{02}**	**0.00×10^{0}**	**5.00×10^{02}**	**5.00×10^{02}**
F6	ADHHO	**6.07×10^{02}**	9.24×10^{-01}	**6.06×10^{02}**	**6.09×10^{02}**
	HHO	6.08×10^{02}	**7.76×10^{-01}**	6.07×10^{02}	6.10×10^{02}
F7	ADHHO	**7.47×10^{02}**	**6.08×10^{0}**	**7.42×10^{02}**	**7.61×10^{02}**
	HHO	7.87×10^{02}	1.89×10^{01}	$7.64E \times 10^{02}$	8.23×10^{02}
F8	ADHHO	**1.19×10^{03}**	**6.08×10^{01}**	**1.13×10^{03}**	**1.34×10^{03}**
	HHO	2.37×10^{03}	7.46×10^{02}	1.66×10^{03}	4.06×10^{03}
F9	ADHHO	**9.58×10^{02}**	**5.44×10^{0}**	**9.49×10^{02}**	**9.67×10^{02}**
	HHO	9.75×10^{02}	7.67×10^{0}	9.60×10^{02}	9.86×10^{02}
F10	ADHHO	**1.83×10^{03}**	**2.85×10^{02}**	**1.28×10^{03}**	**2.49×10^{03}**
	HHO	1.20×10^{04}	2.75×10^{03}	8.23×10^{03}	1.65×10^{04}
F11	ADHHO	**1.21×10^{03}**	**1.69×10^{01}**	**1.18×10^{03}**	**1.24×10^{03}**
	HHO	1.42×10^{03}	1.29×10^{02}	1.28×10^{03}	1.69×10^{03}
F12	ADHHO	**1.23×10^{03}**	**7.67×10^{0}**	**1.22×10^{03}**	**1.25×10^{03}**
	HHO	1.31×10^{03}	3.91×10^{01}	1.25×10^{03}	1.37×10^{03}

F13	ADHHO	**1.32 × 10^03**	**9.39 × 10^-01**	**1.32 × 10^03**	**1.33 × 10^03**
	HHO	1.37 × 10^03	4.49 × 10^01	1.34 × 10^03	1.48 × 10^03
F14	ADHHO	**1.30 × 10^04**	**1.43 × 10^04**	**2.91 × 10^03**	**5.07 × 10^04**
	HHO	1.22 × 10^05	4.71 × 10^04	4.18 × 10^04	2.11 × 10^05
F15	ADHHO	**4.18 × 10^03**	**1.32 × 10^02**	**3.89 × 10^03**	**4.37 × 10^03**
	HHO	4.61 × 10^03	3.17 × 10^02	4.14 × 10^03	5.17 × 10^03
F16	ADHHO	**1.65 × 10^03**	**9.68 × 10^0**	**1.64 × 10^03**	**1.67 × 10^03**
	HHO	1.67 × 10^03	1.17 × 10^01	1.65 × 10^03	1.69 × 10^03
F17	ADHHO	**5.94 × 10^03**	**1.28 × 10^02**	**5.71 × 10^03**	**6.11 × 10^03**
	HHO	6.42 × 10^03	1.60 × 10^02	6.21 × 10^03	6.70 × 10^03
F18	ADHHO	**6.04 × 10^03**	**9.34 × 10^01**	**5.85 × 10^03**	**6.18 × 10^03**
	HHO	6.48 × 10^03	1.04 × 10^02	6.36 × 10^03	6.67 × 10^03
F19	ADHHO	**1.93 × 10^03**	**1.99 × 10^0**	**1.93 × 10^03**	**1.94 × 10^03**
	HHO	1.94 × 10^03	2.41 × 10^0	1.94 × 10^03	1.94 × 10^03
F20	ADHHO	**2.26 × 10^03**	**1.04 × 10^0**	**2.26 × 10^03**	**2.26 × 10^03**
	HHO	2.34 × 10^03	1.96 × 10^01	2.31 × 10^03	2.37 × 10^03

Table 4.6: Performance analysis of ADHHO and HHO on the CEC2017 benchmark suite

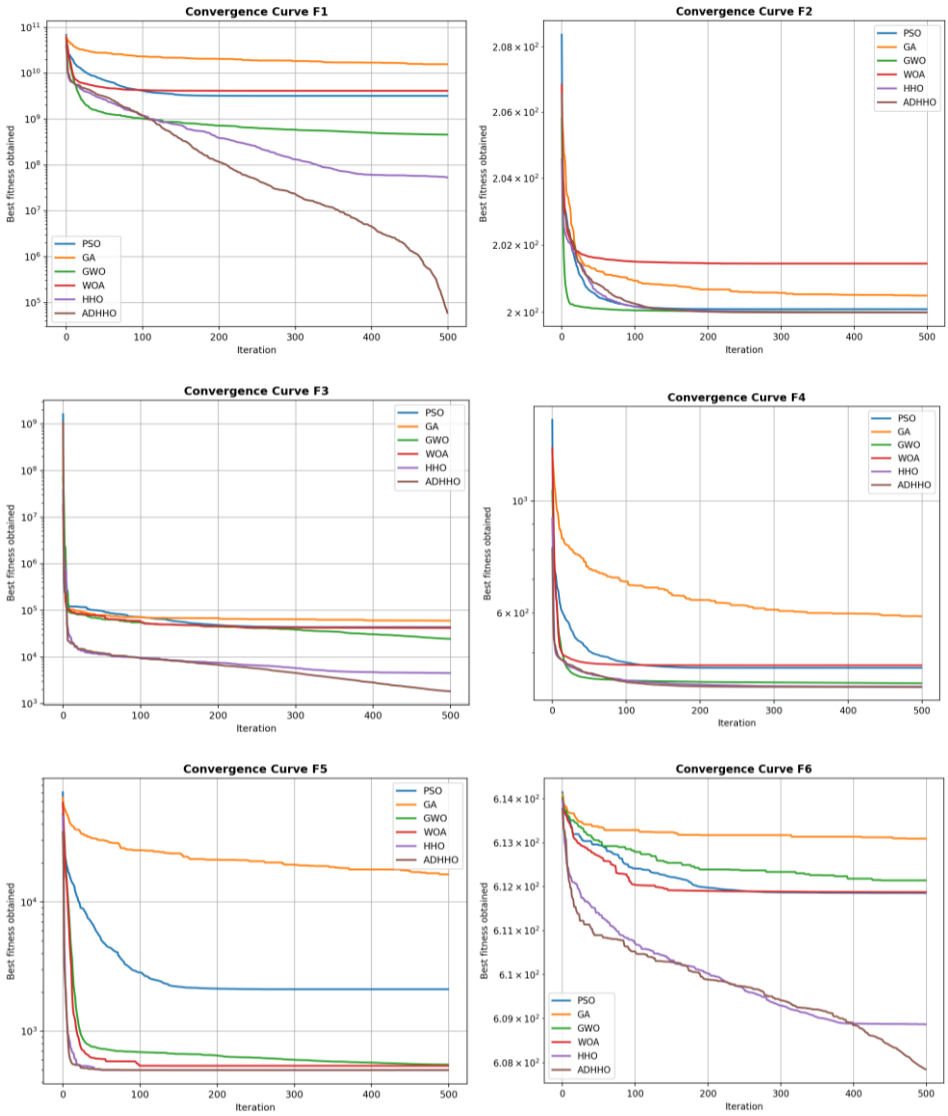
Fun	Metric	ADHHO	HHO	GWO	PSO	GA	WOA
F1	MEAN	**2.20 × 10^04**	1.73 × 10^07	2.30 × 10^08	7.35 × 10^08	1.19 × 10^10	3.14 × 10^09
	STD	**6.19 × 10^03**	9.74 × 10^06	1.76 × 10^08	3.52 × 10^08	1.54 × 10^09	1.15 × 10^09
F2	MEAN	**2.00 × 10^02**	**2.00×10^02**	**2.00 × 10^02**	**2.00 × 10^02**	**2.00×10^02**	2.01 × 10^02
	STD	5.11 × 10^-06	8.54 × 10^-04	3.54 × 10^-03	8.78 × 10^-03	1.27 × 10^-01	**4.02×10^-01**
F3	MEAN	**8.73 × 10^02**	2.71 × 10^03	9.11 × 10^03	2.94 × 10^04	5.64 × 10^04	2.26 × 10^04
	STD	**1.72 × 10^02**	5.33 × 10^02	3.84 × 10^03	1.86 × 10^04	7.30 × 10^03	2.30 × 10^04
F4	MEAN	**4.28 × 10^02**	4.29 × 10^02	4.30 × 10^02	4.37 × 10^02	5.84 × 10^02	4.61 × 10^02
	STD	4.04 × 10^-01	**3.22×10^-01**	1.46 × 10^0	3.60 × 10^0	3.47 × 10^01	1.13 × 10^01
F5	MEAN	**5.00 × 10^02**	**5.00×10^02**	5.54 × 10^02	1.33 × 10^03	1.43 × 10^04	5.41 × 10^02
	STD	**0.00 × 10^0**	**0.00×10^0**	1.70 × 10^01	2.90 × 10^02	1.75 × 10^03	8.16 × 10^01
F6	MEAN	**6.07 × 10^02**	6.08 × 10^02	6.12 × 10^02	6.12 × 10^02	6.13 × 10^02	6.12 × 10^02
	STD	9.24 × 10^-01	7.76 × 10^-01	4.48 × 10^-01	9.79 × 10^-01	**1.96×10^-01**	8.25 × 10^-01
F7	MEAN	**7.47 × 10^02**	7.87 × 10^02	7.76 × 10^02	1.48 × 10^03	1.26 × 10^04	9.18 × 10^02
	STD	6.08 × 10^0	1.89 × 10^01	1.82 × 10^01	2.95 × 10^02	1.84 × 10^03	3.80 × 10^01
F8	MEAN	**1.19 × 10^03**	2.37 × 10^03	2.06 × 10^03	6.06 × 10^03	1.80 × 10^04	2.71 × 10^04
	STD	**6.08 × 10^01**	7.46 × 10^02	7.86 × 10^02	2.96 × 10^03	6.50 × 10^03	1.02 × 10^04
F9	MEAN	9.58 × 10^02	9.75 × 10^02	**9.18 × 10^02**	1.69 × 10^03	6.55 × 10^03	1.07 × 10^03
	STD	**5.44 × 10^0**	7.67 × 10^0	9.15 × 10^0	4.67 × 10^02	1.07 × 10^03	4.33 × 10^01
F10	MEAN	**1.83 × 10^03**	1.20 × 10^04	9.67 × 10^03	1.92 × 10^04	3.91 × 10^04	5.54 × 10^04
	STD	**2.85 × 10^02**	2.75 × 10^03	2.55 × 10^03	1.13 × 10^04	8.36 × 10^03	2.25 × 10^04
F11	MEAN	**1.21 × 10^03**	1.42 × 10^03	1.48 × 10^03	2.17 × 10^03	1.99 × 10^03	3.78 × 10^03

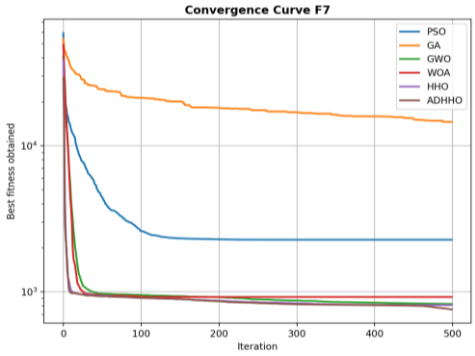
F12	STD	**1.69×10^{01}**	1.29×10^{02}	1.96×10^{02}	3.89×10^{02}	1.99×10^{02}	1.37×10^{03}
	MEAN	**1.23×10^{03}**	1.31×10^{03}	1.33×10^{03}	1.86×10^{03}	1.90×10^{03}	3.81×10^{03}
F13	STD	**7.67×10^{0}**	3.91×10^{01}	1.52×10^{02}	1.51×10^{03}	2.47×10^{02}	1.35×10^{03}
	MEAN	**1.32×10^{03}**	1.37×10^{03}	1.56×10^{03}	2.01×10^{03}	2.12×10^{03}	3.65×10^{03}
F14	STD	**9.39×10^{-01}**	4.49×10^{01}	1.57×10^{02}	4.60×10^{02}	1.71×10^{02}	1.18×10^{03}
	MEAN	**1.30×10^{04}**	1.22×10^{05}	6.49×10^{04}	5.14×10^{04}	1.11×10^{05}	6.13×10^{05}
F15	STD	**1.43×10^{04}**	4.71×10^{04}	4.45×10^{04}	3.27×10^{04}	4.03×10^{04}	3.05×10^{05}
	MEAN	**4.18×10^{03}**	4.61×10^{03}	4.82×10^{03}	6.40×10^{03}	7.15×10^{03}	1.16×10^{04}
F16	STD	**1.32×10^{02}**	3.17×10^{02}	2.84×10^{02}	7.36×10^{02}	8.70×10^{02}	2.95×10^{03}
	MEAN	**1.65×10^{03}**	1.67×10^{03}	1.70×10^{03}	1.74×10^{03}	1.69×10^{03}	1.86×10^{03}
F17	STD	**9.68×10^{0}**	1.17×10^{01}	1.15×10^{02}	2.32×10^{02}	1.27×10^{01}	1.83×10^{02}
	MEAN	**5.94×10^{03}**	6.42×10^{03}	6.58×10^{03}	6.82×10^{03}	9.05×10^{03}	1.19×10^{04}
F18	STD	**1.28×10^{02}**	1.60×10^{02}	2.96×10^{02}	1.70×10^{02}	8.87×10^{02}	2.06×10^{03}
	MEAN	**6.04×10^{03}**	6.48×10^{03}	6.75×10^{03}	6.95×10^{03}	9.16×10^{03}	1.19×10^{04}
F19	STD	**9.34×10^{01}**	1.04×10^{02}	2.29×10^{02}	1.30×10^{02}	8.89×10^{02}	2.04×10^{03}
	MEAN	**1.932×10^{03}**	1.938×10^{03}	**1.932×10^{03}**	**1.932×10^{03}**	1.935×10^{03}	1.935×10^{03}
F20	STD	1.99×10^{0}	2.41×10^{0}	1.90×10^{0}	2.40×10^{0}	**1.35×10^{0}**	2.31×10^{0}
	MEAN	**2.26×10^{03}**	2.34×10^{03}	3.29×10^{03}	3.30×10^{03}	1.12×10^{04}	1.56×10^{04}
	STD	**1.04×10^{0}**	1.96×10^{01}	8.99×10^{02}	4.26×10^{02}	1.14×10^{03}	2.18×10^{03}

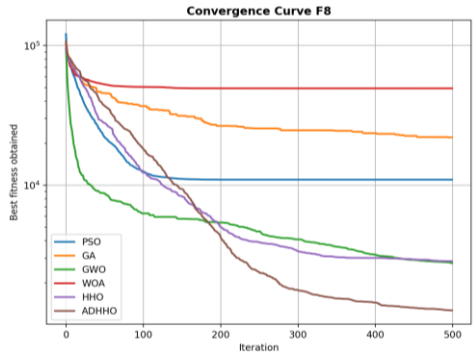
Table 4.7: Performance analysis of ADHHO and other algorithms on the CEC2017 benchmark suite

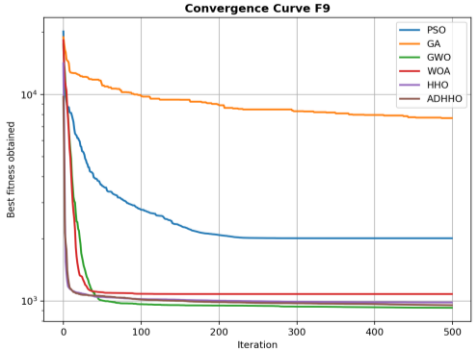
functions, except on F9, on which GWO outperforms all studied algorithms. On standard deviations, ADHHO achieved the best scores on 15 out of 20 test functions. Overall, the ADHHO achieved the best score on 34 out of 40 metric values, followed by HHO (4), GWO (3), GA (3), PSO (2), and WOA (1). The convergence curves for all algorithms on 20 test functions are presented in Fig. 4.6. The figure shows that GWO initially converges faster than other algorithms on most of the test functions. However, its value remains flat or improves very little after 100 iterations. The GWO gives high final fitness scores on almost all test functions. In contrast, the proposed ADHHO achieved the lowest final value on all test functions and converged faster. The low final fitness scores, especially for multi-dimensional and hybrid functions, indicate that the adaptive-HHO has strong exploitation ability. Due to these abilities, the ADHHO can avoid local optima stagnation. Additionally, the lowest final values indicate that ADHHO can escape local optima and maintain exploration. Hence, the strong exploitation and

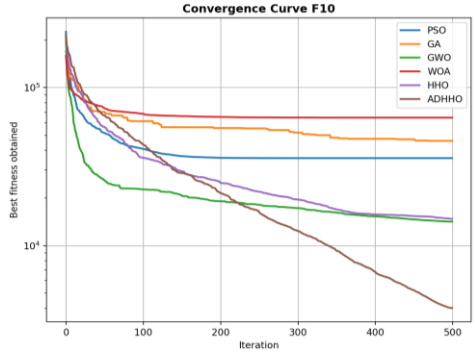
exploration capabilities support the proposed algorithm in providing high-quality solutions, which are critical in real-life applications. Among other algorithms, WOA performed the worst on most of the test functions. This algorithm exhibits slow convergence and achieves high final fitness values. The overall results revealed that ADHHO can be employed for multi-dimensional and complex optimization problems.

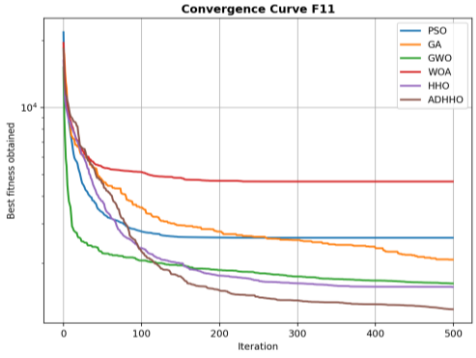


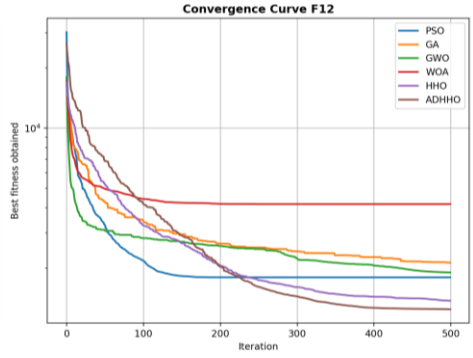

Convergence Curve F7
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration

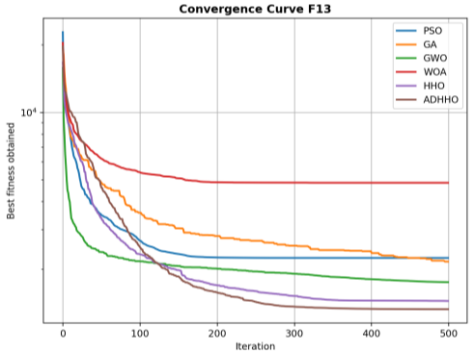

Convergence Curve F8
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration

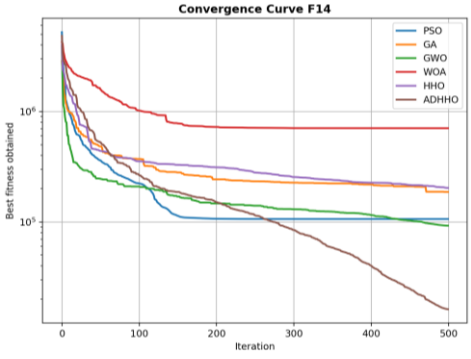

Convergence Curve F9
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F10
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F11
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F12
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F13
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration


Convergence Curve F14
PSO
GA
GWO
WOA
HHO
ADHHO
Best fitness obtained
Iteration

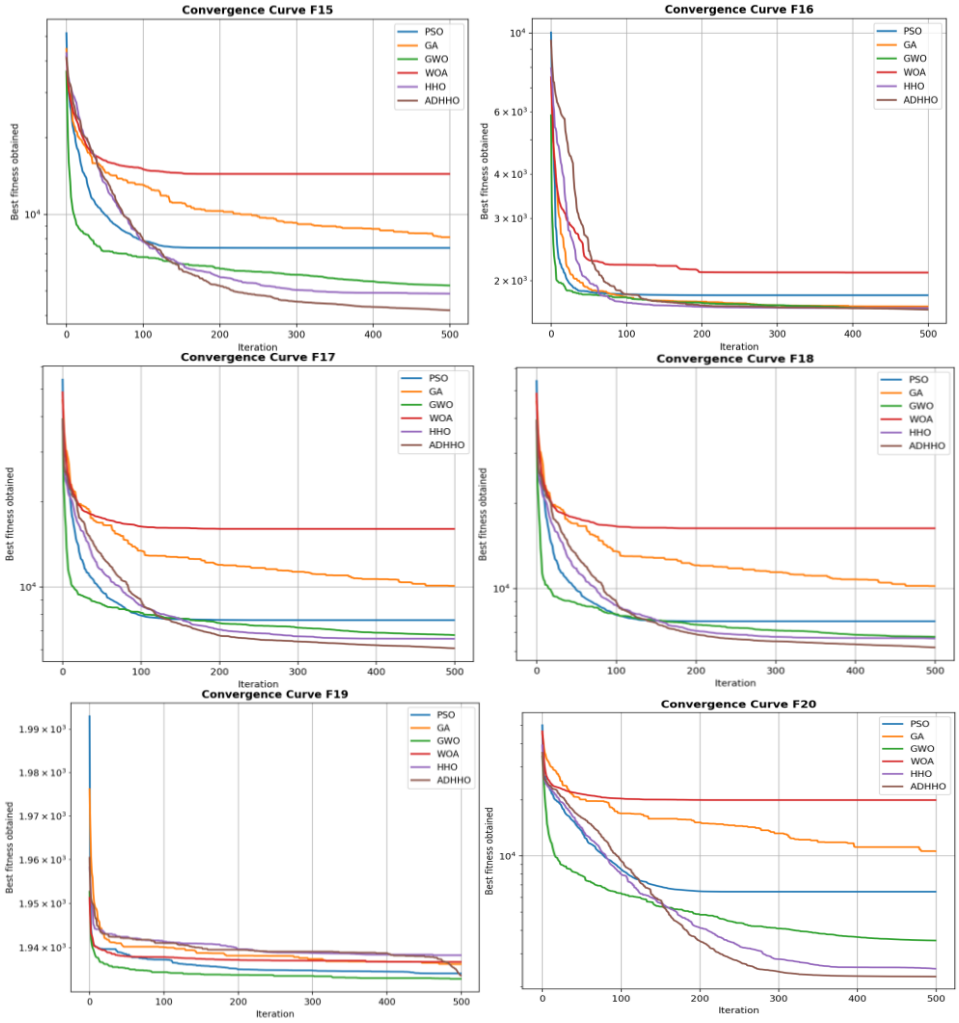


Figure 4.6: Convergence curve for ADHHO and other evaluated algorithms on CEC2017

4.6 Summary

The importance of optimization problems has significantly increased due to advances in technology and the integration of various fields. Classical methods, which are suitable for smooth and unimodal functions, may fail to solve the complex optimization problems. These methods may experience stagnation of local optima, particularly in complex, non-differentiable, and nonlinear optimization problems. Hence, metaheuristic algorithms are widely applied to these problems. These algorithms can provide an acceptable solution within a

reasonable time frame. They have high efficiency and can present robust solutions to noisy datasets. Their population-based design enables efficient parallelization, which improves computation for complex problems. Over time, multiple MOAs have been proposed, though swarm-based algorithms are widely researched due to their flexibility and high problem-solving abilities. These algorithms are robust and adaptable, which makes them ideal for energy predictions, complex engineering design, deep learning parameter tuning, and various other real-world problems. However, most swarm-based algorithms are prone to getting stuck in local optima. They are sensitive to parameter settings; hence, they require precise parameter adjustments to achieve high-quality results. HHO, another swarm-based algorithm that gets inspiration from the cooperative and hunting nature of the Harris hawks. This algorithm has multiple distinctive features compared to other prominent swarm intelligence-based algorithms. Firstly, this algorithm has a strong equilibrium between exploration and exploitation. Secondly, this algorithm employs a time-varying method that improves with iterations, resulting in a smooth transition from exploration to exploitation. Thirdly, it exhibits a gradual convergence behavior that increases the possibility of finding the most favorable solutions. Finally, this algorithm's exploitation stage employs a greedy strategy to prioritize the best solutions over the low-quality ones.

Although HHO can be efficient for solving various optimization problems, due to its stochastic nature, the original HHO may still be prone to getting trapped in local optima for multi-dimensional problems. Another problem of HHO is the shift between exploration and exploitation, which is regulated by escape energy. This rigid control may cause exploitation to begin too late, resulting in slow convergence. Additionally, HHO's exploitation phase relies on random jumps, and inappropriate jumps near optimum solutions may result in overshoot. Thus, to achieve low final fitness values, better convergence rate, and higher efficiency, improvement in the exploration and exploitation phases is important. Various studies have proposed advanced and enhanced variants of HHO that overcome, somehow, the limitations of the original algorithm. These enhancements include introducing a nonlinear energy factor to improve the shift between exploration and exploitation phases and opposition-based learning to increase the convergence rate. Additionally, a self-adaptive method is included to increase diversity and leverage on already explored regions. Moreover, some studies used

chaotic initialization and mutations to enhance diversity and improve exploration. Although the enhanced versions have addressed certain features of the original HHO, they still have major shortcomings that restrict their applications in various optimization problems. The improvements using these variants are only useful in certain stages of the search process. For instance, chaotic versions may enhance early exploration yet struggle to maintain diversity in final iterations. Opposition-based learning increases the number of fitness evaluation, resulting in a high computational cost. Levy flight-improved HHO can lead to overshooting and may disturb fine-tuning close to global optima. Similarly, multiple studies combine HHO with other advanced MOAs to enhance the solution accuracy of different research areas. However, hybrid methods can inherit the drawbacks of the second algorithm and can be computationally more expensive, particularly for multi-dimensional problems.

Hence, it is vital to develop an adaptive-HHO for maintaining a balance between exploration and exploitation, enhancing convergence speed, and boosting robustness without increasing computational costs. Thus, this research presents an Adaptive Harris Hawk Optimization (ADHHO). The proposed method improves the original HHO in four phases. Firstly, two chaotic maps are used, one for initializing the hawks' population and another for generating random numbers. Secondly, a nonlinear energy factor is used to maintain a balance between global and local search. Thirdly, an adaptive inertia weight mechanism is introduced to enhance the convergence rate and search stability. Lastly, an adaptive decay factor *Sr* is used to regulate the steps as the algorithm converges. To assess the performance and convergence characteristics of the proposed ADHHO, some experiments are conducted on well-known IEEE CEC benchmark suites, including 2014 and 2017. The performance results on these benchmark functions are then assessed against prominent metaheuristic algorithms, such as PSO, GA, GWO, WOA, and HHO. The findings indicate that the proposed ADHHO achieves superior results compared to all studied algorithms. The ADHHO achieved the theoretical best scores for most of the benchmark functions. These functions belong to all three groups: unimodal, multimodal, and fixed dimension. On the CEC 2014 benchmark functions, the proposed ADHHO obtained the best score on 40 out of 44 metric values, followed by HHO (9), WOA (5), GA (3), GWO (1), and PSO (0). Finally, on CEC 2017, the

ADHHO achieved the best score on 34 out of 40 metric values, followed by HHO (4), GWO (3), GA (3), PSO (2), and WOA (1).

5

Experimental Evaluation of the Alice Springs Dataset

5.1 Dataset Requirement and Characteristics for Deep Learning Models

Deep learning models have complex structures, consisting of multiple layers and thousands of trainable parameters. Each parameter (weight or bias) indicates a small amount of learned information. Together, these parameters present an intricate feature space, allowing DLMs to capture complex relationships between input datasets. However, DLMs normally require large amounts of diverse datasets to achieve this [128], [129]. This is because each parameter requires enough information to learn useful patterns. Since these models have thousands or millions of parameters, they need large datasets to acquire precise representations of the input datasets. In other words, DLMs are heavily dependent on datasets to understand hidden relationships, capture important features, and then provide precise forecasts using these learned representations. Additionally, the size of the datasets can impact the model's generalization [130]. When trained on enough datasets, DLMs with an appropriate size are exposed to multiple fluctuations and conditions, enabling them to learn meaningful connections with the datasets. As a result, they may provide precise forecasts on unseen datasets, minimizing the overfitting issue. In overfitting, DLMs become heavily dependent on the training dataset, memorize certain trends rather than understanding basic concepts. When these models are exposed to unseen datasets, they provide very low results [131]. The availability of sufficient data is also crucial for evaluating the

performance of the DLMs in comparison to other baseline models. When enough data is available, it can easily be divided into training, validation, and testing datasets. This allows the precise assessment of the forecasting models.

Another important factor is the overall quality of the dataset. DLMs are data-driven models able to learn complex and nonlinear relationships between input datasets. The performance and accuracy of these models highly depend on the correctness and quality of the input datasets. A high-quality dataset is associated with high accuracy and performance, whereas a low-quality dataset can result in accuracy degradation [132]. Quality datasets allow DLMs to learn important and generalizable patterns. For instance, when the dataset is uniformly distributed and free from anomalies, DLMs may easily capture the true correlations between the input and target features. On the other hand, when datasets contain outliers, missing values, or erroneous values, it may result in poor model performance [132]. This is because these datasets may not fully assess the capacity of the DLMs and can negatively affect the model's generalization. In addition, outliers, missing values, or erroneous values in the training dataset can lead to unstable learning patterns. Learning these unstable patterns may reduce the performance and generalization of DLMs. This decline in performance can be worse when the testing datasets contain missing or false values and DLMs are not trained on such datasets. Hence, in short, the quality of datasets determines the quality of forecasting models [133]. Normally, outliers, missing values, or erroneous values may arise from multiple reasons. For instance, missing values can be due to sensor faults, data transmission issues, human errors, and unavailability. Erroneous values may occur due to improper sensor calibration, sensor malfunctions, software bugs, and data logging errors. Outliers may arise due to sensor malfunctions, failures in communication and measurement apparatus, irregularities in maximum power tracking, and array shutdowns [118]. Hence, to improve data quality, data preprocessing and postprocessing are essential, which enable models to identify key traits and eliminate outliers from the dataset.

Similarly, data diversity is equally important for enhancing model accuracy, efficiency, and reliability [134]. Data diversity is defined as the range and depth of data points within a given dataset, such as their categories, features, situations, and representations. The data diversification aims to enhance the range of information within the datasets. This allows forecasting models to capture more

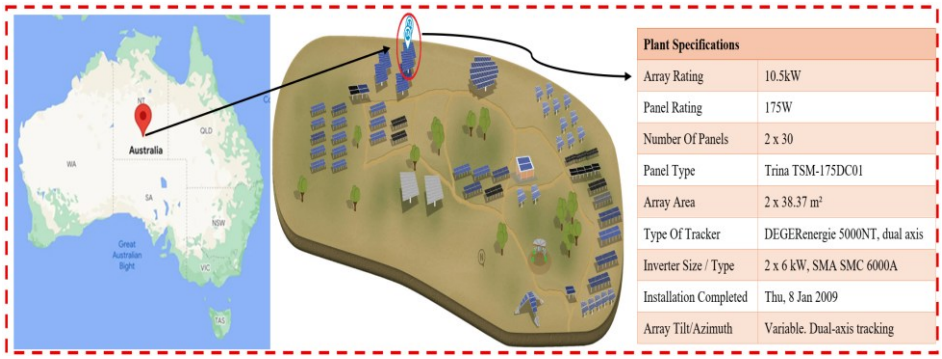
detailed knowledge through the learning process. Once forecasting models are trained on these datasets, they can perform well with limited training samples. Studies show that enhancing data diversity can reduce overfitting, similar to conventional regularization approaches [135]. Enhancing diversity in the dataset can affect the distribution of the model's weight matrix, thus impacting the model's learning and generalization process. This effect is similar to applying dropout for regularization. Normally, when a dataset has low diversity, a data augmentation method is used. This method applies various transformations to the input dataset, including scaling, permutation, window slicing, and adding noise. Data augmentation increases variability in the training dataset, which allows forecasting models to perform better on unseen data. Hence, data augmentation improves robustness, generalization, and model adaptability.

5.2 Alice Springs Dataset: Characteristics and Visualization

The dataset used for training and analysis of the proposed BiLSTM-CNN-MHA optimized with ADHHO was obtained from DKASC in Alice Springs. Alice Springs is an isolated town in the Northern Territory of Australia, located at about an equal distance from Darwin (1,499 km) and Adelaide (1,532 km) [136]. Köppen climate classifies this region as a hot desert climate (BWh). The annual average rainfall in this region is about 285.9mm, representing a semi-desert environment. Due to the semi-desert climate, the temperature in this region varies widely. During the summer season (December-February), the average temperature in Alice Springs ranges from 20°C to 35°C, with temperatures potentially reaching 40°C [137]. In winter (June to August), the temperature can vary between 4.4°C and 20°C, and may fall below 0°C. The climate in this region is erratic, with extremely hot summers and freezing winters. Hence, Alice Springs has high PV power potential and is considered an ideal location for the installation and development of a PV plant. Over the years, almost 40 PV systems have been installed at the DKASC research facility in Alice Springs. These PV systems feature a wide range of designs and module technologies. Hence, this research facility provides a prospective testbed for assessing the reliability and efficiency of different PV technologies.

Additionally, this facility constantly measures various weather variables and the real-time PV power data from multiple sites. These measured data are publicly available on https://dkasolarcentre.com.au/ [138]. Researchers worldwide use these datasets for several key objectives. These include assessing the efficiency and stability of different PV technologies, testing and validating ML and DL models, analyzing the impact of weather variables on PV power generation, and comparing fixed and tracking system configurations.

For our analysis, we obtained data from site 1A (Trina, 10.5kW, mono-Si, Dual, 2009). The location and specifications of this site are presented in Fig. 5.1. Trina Solar is the manufacturer of the PV modules used on this site, which has an installed capacity of 10.5 kW. The PV modules of this site use monocrystalline silicon technology. This technology achieves high conversion efficiency [139] and exhibits superior performance in various environmental scenarios. As mentioned in the figure, the PV modules of this site are mounted on dual-axis trackers, which precisely track the sun's position throughout the day. The dual-axis tracker can give higher performance compared to a single-axis tracker. This is because the dual-axis tracker adjusts the modules' position on two separate axes, optimizing the sunlight exposure. This tracker can rotate the PV module an entire 360° [140], enhancing energy yield. The 1A site has 2 x 30 panels, each with a power rating of 175W. It is equipped with two 6 kW SMA SMC 6000A inverters, and the installation was completed on January 8, 2009. The chosen dataset contains various attributes, including active energy, current, active power, wind speed, temperature, relative humidity, global horizontal radiation, diffuse horizontal



Plant Specifications	
Array Rating	10.5kW
Panel Rating	175W
Number Of Panels	2 x 30
Panel Type	Trina TSM-175DC01
Array Area	2 x 38.37 m²
Type Of Tracker	DEGERenergie 5000NT, dual axis
Inverter Size / Type	2 x 6 kW, SMA SMC 6000A
Installation Completed	Thu, 8 Jan 2009
Array Tilt/Azimuth	Variable. Dual-axis tracking

Figure 5.1: The map of Alice Springs and specifications of the Trina Solar 1A site

radiation, wind direction, rainfall, global tilted radiation, and diffuse tilted radiation. For our analysis, we select PV power and six variables that show strong positive and negative correlations with PV power. The six variables are global horizontal radiation (W/m²), diffuse horizontal radiation (W/m²), temperature (°C), relative humidity (%), wind speed (m/sec²), and wind direction (°). The detailed description of these six variables and PV power is presented in Tab. 5.1. To enhance the interpretability and visual illustration of the temporal patterns in the chosen dataset, Fig. 5.2 presents the Mat plot of the first three months of the PV power and all six variables. Additionally, the daily and hourly distribution of PV output is presented in Fig. 5.3. Fig. 5.2 shows that PV power, global horizontal radiation, and diffuse horizontal radiation follow similar daily cycles. There are multiple variations in power and both radiations, which represent periodic shading or moving clouds. These moving clouds or shading can significantly affect the PV power output, and in some cases, the PV power may fall by more than 50%. Other weather variables, such as temperature, humidity, and wind speed, fluctuate throughout the day. The temperature follows a steady cycle, rising during the day and falling at night, while the humidity increases during the cold night hours. The wind speed, as shown in the figure, varies throughout the day, peaking around midday. To more
clearly illustrate the impact of weather variables and PV power, we use a heatmap function from the scikit-learn library, as shown in Fig. 5.4. The heatmap shows the pairwise correlation between all parameters (weather variables and PV power). Normally, the range of correlation is between +1 and -1. The positive one indicates a strong correlation, while the negative one indicates a weak correlation. Fig. 5.4 illustrates that PV power has a strong correlation with global horizontal radiation (r = 0.97) and diffuse horizontal radiation (r = 0.75). Wind speed and temperature have a moderate positive correlation with PV power, with r values of 0.60 and 0.44, respectively. In contrast, relative humidity has a moderate negative correlation with r = -0.50, whereas wind direction shows a weak negative correlation of r = -0.20 with PV power.

Table 5.1: Description of the dataset

Variables	Power	Wind Speed	Temperature	Humidity	Global Radiation	Diffuse Radiation	Wind Direction
Unit	kW	m/s²	°C	%	W/s²	W/s²	°
Mean	2.454	2.372	21.405	39.132	258.297	56.403	143.173
STD	3.099	3.174	9.572	24.462	354.133	93.946	242.778
Min	-0.033	-919.076	-3.086	3.126	1.140	0.319	-14732.871
25 %	0	1.144	14.511	19.022	2.872	1.226	100.718
50 %	0	2.104	22.022	32.639	8.335	6.578	139.984
75 %	5.637	3.412	28.475	56.170	506.087	72.568	223.578
Max	11.331	13.351	45.286	101.889	1749.500	708.572	337.518

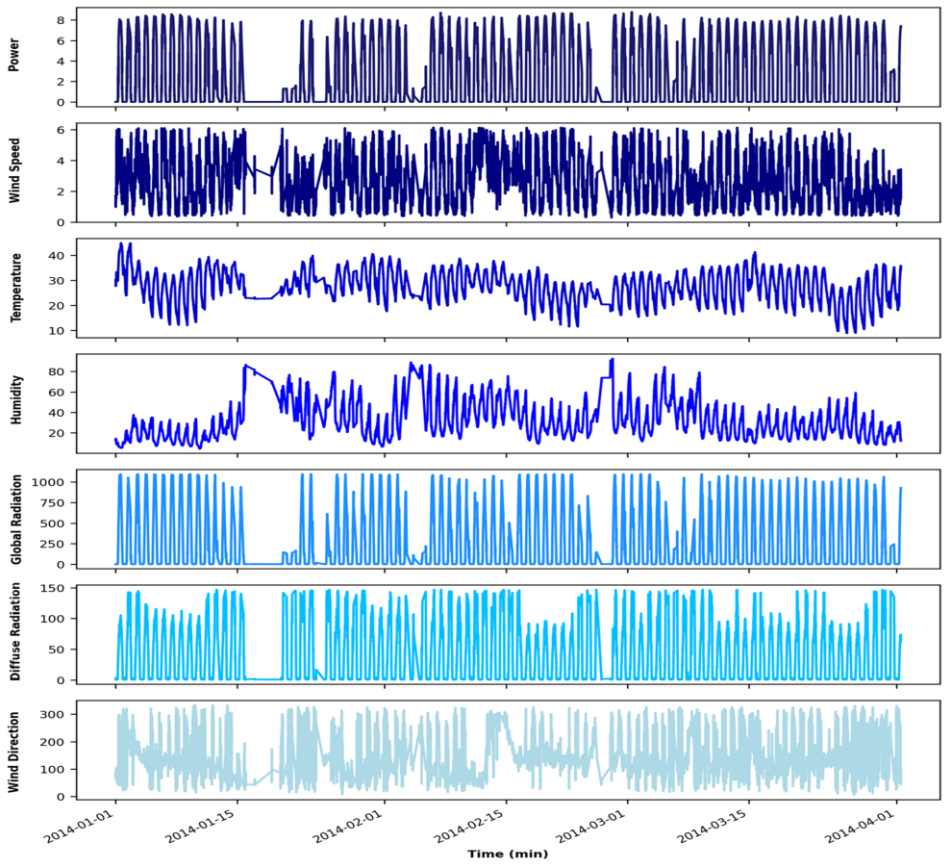


Figure 5.2: Historical data on PV output and weather variables at the BP solar power installation

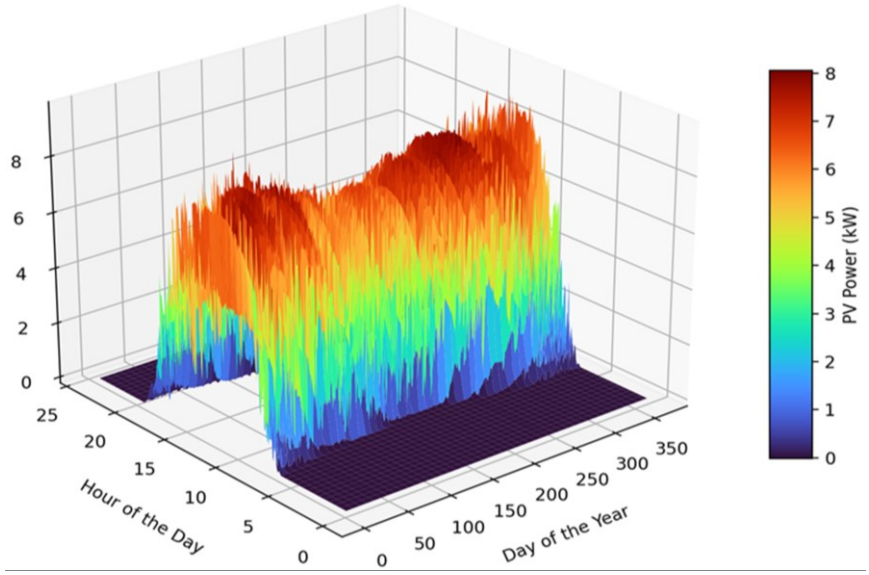


Figure 5.3: Daily and hourly distribution of PV power output

Fig. 5.2 shows that PV power, global horizontal radiation, and diffuse horizontal radiation follow similar daily cycles. There are multiple variations in power and both radiations, which represent periodic shading or moving clouds. These moving clouds or shading can significantly affect the PV power output, and in some cases, the PV power may fall by more than 50%. Other weather variables, such as temperature, humidity, and wind speed, fluctuate throughout the day. The temperature follows a steady cycle, rising during the day and falling at night, while the humidity increases during the cold night hours. The wind speed, as shown in the figure, varies throughout the day, peaking around midday. To more clearly illustrate the impact of weather variables and PV power, we use a heatmap function from the scikit-learn library, as shown in Fig. 5.4. The heatmap shows the pairwise correlation between all parameters (weather variables and PV power). Normally, the range of correlation is between +1 and -1. The positive one indicates a strong correlation, while the negative one indicates a weak correlation. Fig. 5.4 illustrates that PV power has a strong correlation with global horizontal radiation (r = 0.97) and diffuse horizontal radiation (r = 0.75). Wind speed and temperature have a moderate positive correlation with PV power, with r values of 0.60 and 0.44, respectively. In contrast, relative humidity has a moderate negative

correlation with r = -0.50, whereas wind direction shows a weak negative correlation of r = -0.20 with PV power.

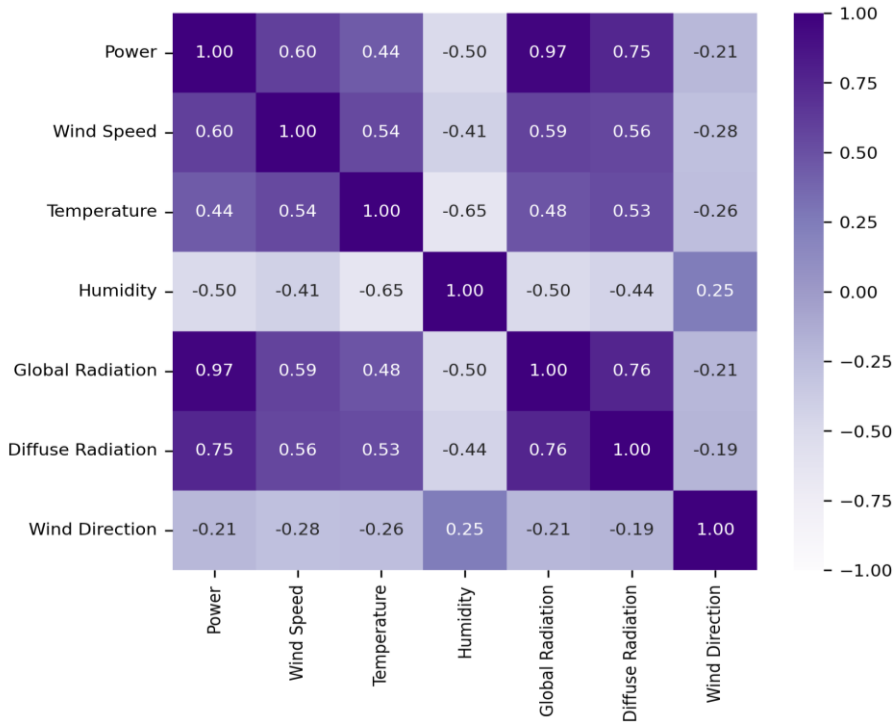


Figure 5.4: Heatmap correlation of weather variables and PV power

5.3 Data Cleaning: Missing Values and Outliers

After data visualization, the dataset is checked for missing values and unwanted values. Missing values [141] and unwanted characters are common problems in real-world datasets. These issues may occur due to multiple reasons, including human error, sensor faults, communication failures, and equipment malfunctions. These issues can significantly impact the learning process, resulting in poor accuracy. DLMs normally learn trends and patterns in the input datasets. When these trends and patterns are distributed, these models may learn incorrect relationships. Additionally, DLMs minimize the loss function [142] using the gradient descent technique, which indicates how much each parameter should be updated during training. Missing values or unwanted characters can result in distorted feature values, incorrect loss estimation, and erroneous gradient

updates. When these values occur irregularly, DLMs may learn biased relationships. Hence, to remove missing values and unwanted characters, we used a simple linear interpolation function. Linear interpolation is computationally less expensive [143] and can work well for time series with regular interruptions. The Python code for demonstrating missing values and unwanted characters is shown below.

```
# Missing and unwanted values analysis

def missing_unwanted (df, char):

    char=char.replace(".", "").replace("-", "")
    values="[" + re.escape(char) +"]"
    df=df.map(lambda x:re.sub(values, "", x) if isinstance (x, str) else x)
    df=df.apply (pd.to_numeric, errors="coerce")
    df=df.interpolate(method="linear", limit_direction="both")
    return df
df=missing_unwanted(df,"/#$" )
```

After filling missing values and removing unwanted characters, the dataset is checked for outliers or anomalies. Outliers are observations that differ drastically from the general dataset's patterns and trends [144]. In other words, these are unusual events that vary considerably from the normal instances. Outliers can arise from multiple reasons, including experimental errors [145], sensor malfunctions, instrument calibration problems, shading effects, power inverter faults, preventive maintenance periods, and sudden weather fluctuations. Similar to missing values, outliers or anomalies can significantly impact the model performance [144], [145]. They may extend the range of the measured values, including average and standard deviation, resulting in inaccurate analysis. Additionally, they may distort the patterns and trends in the dataset, leading to low forecasting accuracy. Moreover, outliers can impact the decision of selecting a suitable model and the predicted values of the target variables [146]. Hence, removing outliers is important for enhanced interpretability, faster convergence, and better generalization. In the literature, various techniques are employed to remove outliers or anomalies. These include the Z-score method, K-nearest neighbors, the interquartile range, polynomial fit, wavelet transforms, isolation forest, and others [144], [146]. Among them, the interquartile range and Z-score

are widely used techniques due to their robustness and simple implementation. In this study, we utilize the interquartile range function from the Pandas Library for removing outliers or anomalies. The Python code for detecting and removing outliers is presented below. For visualization, we use box plots from Seaborn as shown in Fig. 5.5. The figure highlights the outliers with red "+" markers in various weather-related variables, including global horizontal radiation, diffuse horizontal radiation, wind speed, wind direction, and rainfall. The wind speed and wind direction exhibit negative values, which are far from the actual values. These negative values indicate outliers due to measurement faults because wind speed and wind direction can never be negative in practice. After applying the interquartile range function from the Pandas Library, the outliers in all weather-related variables are removed as shown in Fig. 5.5 (b). Now there are no "+" markers in the boxplots, and the dataset can be used for training and testing DLMs.

```
# Outlier detections using box plot
def box_plot(df, cols ):
    flierprops = dict(marker='+', markeredgecolor='red', markersize=10, linestyle='none')
    boxprops = dict(linestyle='-', facecolor='lightgray', edgecolor='black', linewidth=1)
    capprops = dict(linestyle='--', linewidth=2)
    medianprops = dict(linestyle='--', linewidth=2)
    meanprops = dict(linestyle='--', linewidth=2)
    whiskerprops =dict(linestyle='--', linewidth=2)

    fig, axes=plt.subplots(2,4, sharey=False, figsize=(8,4), dpi=300)
    axes=axes.flatten()
    for ax, col in zip(axes, cols ):
        sns.boxplot(data=df, y = col, width=0.2, saturation=0.75, linewidth=2, medianprops=medianprops,
                    whiskerprops= whiskerprops, meanprops = meanprops,  capprops = capprops, boxprops=boxprops, flierprops=flierprops, ax=ax)
        ax.set_ylabel(col, color="black", weight="bold")
        plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

```
# Outlier removal
def outliers (df, threshold):

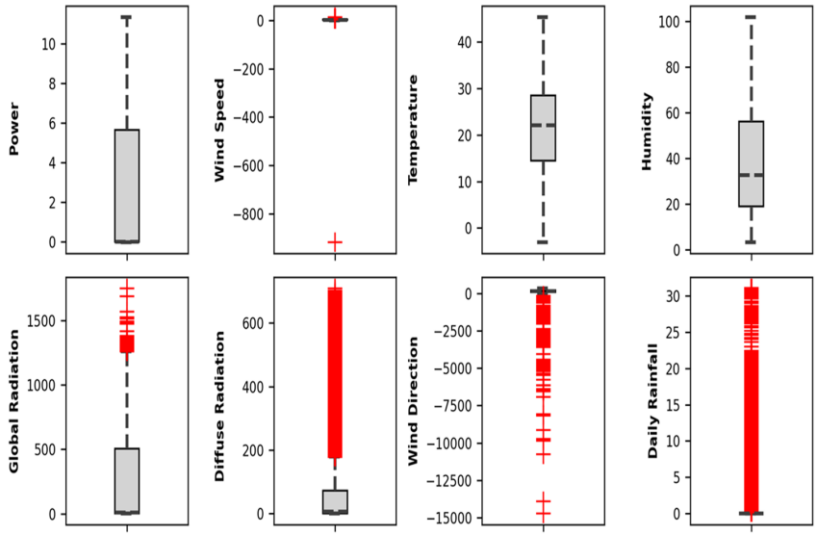
    df_data = df.copy()

    for col in df_data.columns:
        if col == "timestamp":
            continue

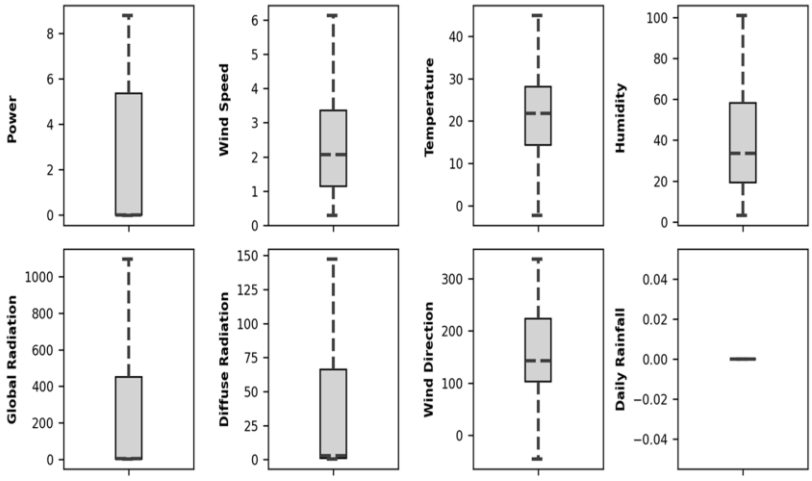
        if pd.api.types.is_numeric_dtype(df_data[col]):
            Q1 = df_data[col].quantile(0.25)
            Q3 = df_data[col].quantile(0.75)
            IQR = Q3 - Q1
            lower = Q1 - threshold * IQR
            upper = Q3 + threshold * IQR

            df_data = df_data.where((df_data[col] >= lower) & (df_data[col] <= upper), float ("nan"))
    df_data = df_data.interpolate(method = "linear", limit_direction = "both")
    return df_data

cols=["Power","Wind Speed", "Temperature", "Humidity", "Global Radiation", "Diffuse Radiation", "Wind Direction", "Rainfall"]
box_plot(df, cols)
df1=outliers(df, 1.2)
print("\n\033[1mBox Plot after applying outlier function\033[0m\n")
box_plot(df1, cols)
```



(a) Box plots of PV power and weather variables before IQR



(b) Box plots of PV power and weather variables after IQR

Figure 5.5: Detecting and removing outliers from Trina Solar data

5.4 Data Normalization and Sequence Generation

After removing outliers and missing values, dataset normalization is an important step in PV data analysis. The PV power dataset contains multiple parameters, including PV power (kW), solar irradiance (W/m²), relative humidity (%), wind speed (m/sec²), temperature (°C), and wind direction (°A). These parameters have different scales and units. Supplying an unnormalized dataset, particularly to deep learning models such as CNN and BiLSTM, may result in low forecasting accuracy and poor performance. Data normalization transforms the datasets to a common range [147], without affecting the correlation between these parameters. This transformation allows all attributes to make an equal contribution to the final output. It assists DLMs in handling all parameters equally. This feature prevents the model from being dominated by parameters that have a wider numeric range [148], such as solar irradiance or PV power output. Additionally, data normalization facilitates faster convergence, enhanced model performance, and improved generalization. In the literature, MinMaxScaler is a widely used method for data normalization due to its various advantages. For instance, it can improve forecasting accuracy by reducing various error metrics,

particularly when utilized with recurrent networks [147]. Additionally, it reduces the maximum and minimum error rates between true and estimated values. Hence, in this study, we employed MinMaxScaler for data normalization. Mathematically, MinMaxScaler can be expressed as [149],

$$\acute{x} = \frac{x - min_o}{max_o - min_o} \times (max_n - min_n) + min_n \tag{5.1}$$

Where $\acute{x}$, x, max, and min indicate the normalized, original, maximum, and minimum values of the attributes. The subscripts o and n represent the old and new values. Following the data normalization, sequence preparation is also crucial for deep learning models. In sequence preparation, the dataset is transformed into a structure that is compatible with the model's structural needs, especially for LSTM, BiLSTM, or any hybrid combination of these models. These models analyze data through gating mechanisms that retain temporal relationships across the sequence. Hence, they require a suitable data structure to utilize their sequential modeling capacity. Typically, a sliding window method is used, in which input-output pairs are created. The data structure for recurrent models is commonly a three-dimensional tensor of shape[batch size, sequence length, features]. The batch size indicates the number of training samples in one forward or backward pass. It regulates the learning rate, memory utilization, and learning stability. A small batch size can provide uniform learning performance, but the model will require a longer training time. On the other hand, a high batch size can provide faster training but may use a large amount of memory. A sequence length is the number of timesteps input to the model. In other words, it indicates the number of previous data points used for predicting future values. An extended sequence length may detect both short and long-term temporal patterns; however, it can be computationally more expensive. Features may be referred to as input variables or parameters. In PV power data, features can be wind speed, humidity, temperature, or irradiance. They are important for the model's performance and validation because they help the model understand the factors that affect PV power production. The following are the Python codes for data normalization and sequence generation. Here, we put the step ahead forecasting equal to one; however, the experimental evaluation is carried out for 1-step ahead, 3-step ahead, and 6-step ahead forecasting.

```
# Data transformation
targets = df1[["Power"]]
features = df1[["Wind Speed", "Temperature", "Humidity", "Global Radiation", "Diffuse Radiation", "Wind Direction"]]

targetsScaler = MinMaxScaler()
featuresScaler = MinMaxScaler()

Y_data = targetsScaler.fit_transform(targets)
X_data = featuresScaler.fit_transform(features)

# Sequence Genaration for DLMs
def model_data(X_data, Y_data, window_size, future):
    #first split the target and features into X_train, X_test, Y_train, Y_test
    X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size=0.1, shuffle=False)
    # initialize the X_train_r, Y_train_r using the sliding window method which will be used for training the model
    X_train_r, Y_train_r = [], []

    for i in range (len(X_train)-window_size-future + 1):
        X_train_r.append(X_train[i : i + window_size])
        Y_train_r.append(Y_train[i + window_size: i + window_size + future, 0])
     # initialize the X_test_r, Y_test_r using the sliding window method which will be used for testing the model
    X_test_r, Y_test_r = [], []

    for i in range (len(X_test)-window_size-future + 1):
        X_test_r.append(X_test[i : i + window_size])
        Y_test_r.append(Y_test[i + window_size: i + window_size + future, 0])

    X_train_r = np.array(X_train_r)
    Y_train_r = np.array(Y_train_r)
    X_test_r = np.array(X_test_r)
    Y_test_r = np.array(Y_test_r)

    return X_train_r, Y_train_r, X_test_r, Y_test_r

# Now pass the data and assign it to X_train_r, Y_train_r, X_test_r, Y_test_r
X_train_r, Y_train_r, X_test_r, Y_test_r = model_data(X_data, Y_data, 10, 1 )
#print the shape to check the available data for model training
print(X_train_r.shape, Y_train_r.shape, X_test_r.shape, Y_test_r.shape)
```

5.5 Proposed Model Architectures

The architecture of the proposed dual-stream attention-based hybrid model optimized with adaptive Harris hawk optimization is presented in Fig. 5.6. Following the data processing stage, the data is fed into both BiLSTM and CNN separately. The BiLSTM branch consists of three layers: BiLSTM, GlobalAveragePooling1D, and Dense, whereas the CNN branch comprises four layers: Conv1D, MaxPooling1D, Flatten, and Dense. In the BiLSTM branch, one

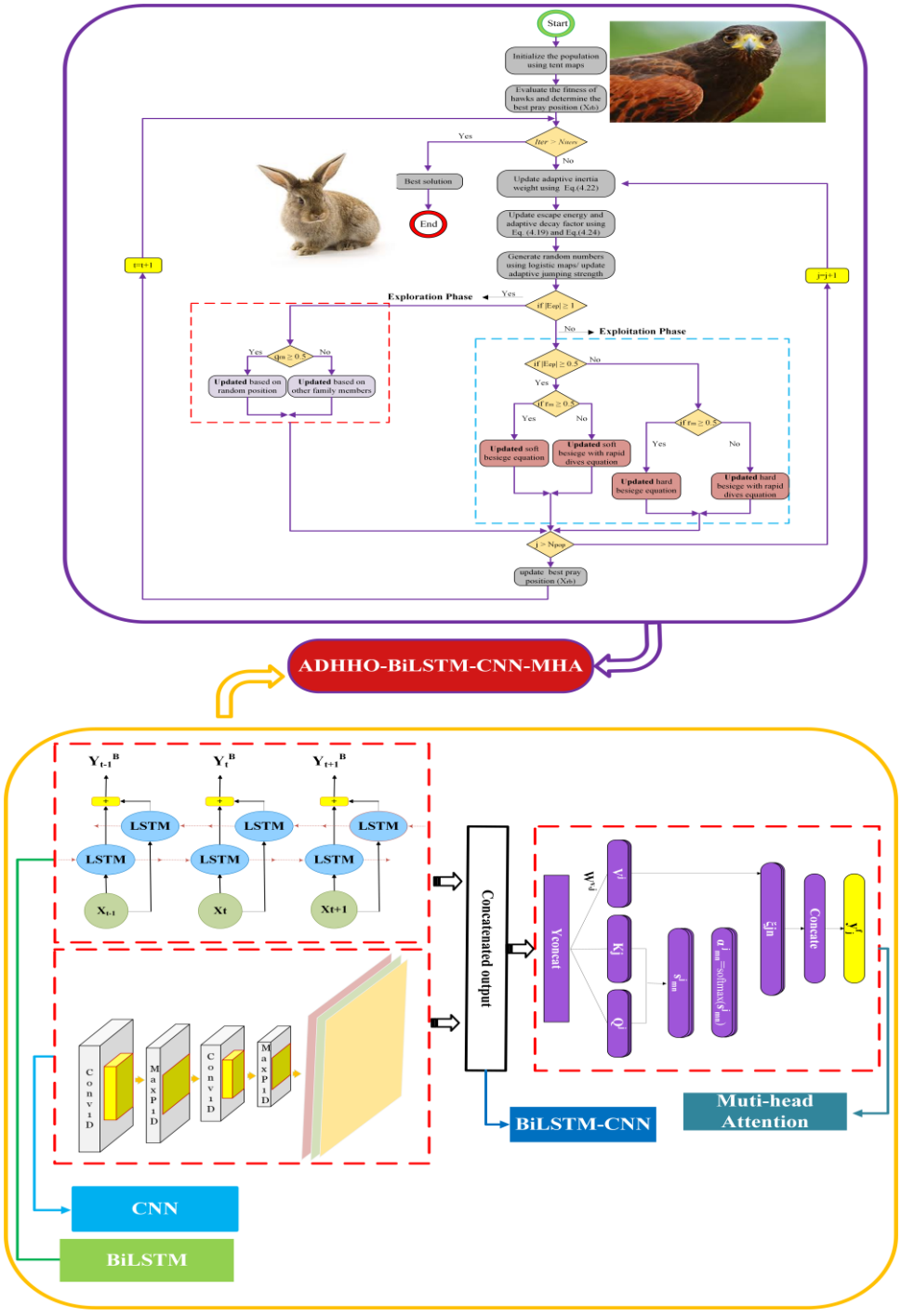


Figure 5.6: Architecture of the proposed hybrid deep learning model with an adaptive Harris Hawk optimization

$LSTM_{forward}$ encodes the sequence from all the preceding steps, and the other $LSTM_{backward}$ encodes the sequence from all future steps. On each timestep (t), the output of both forward and backward LSTM is concatenated $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$ and passed to the GlobalAveragePooling1D layer. The global pooling layer converts the entire temporal sequence into a single vector. The overall temporal behavior captured by the global pooling layer is then transferred to the final dense layer. The final dense layer then transforms the average temporal attributes into a dimension supported by a concatenated layer. In the CNN branch, initially, the Conv1D layer applies sliding filters over the PV datasets to detect local patterns, and then the MaxPooling1D layer is employed to perform a down-sampling operation. Next, the Flatten layer transforms multi-channel feature maps into a 1D vector, preserving the original data. The output of the Flatten layer is then passed to the fully connected layer, which only processes a one-dimensional array. The Flatten layer performs a transformation suitable for the dense layer.

The proposed parallel structure reduces the dimensionality and hence the number of trainable parameters for the BiLSTM-CNN model. Therefore, the proposed model requires less computation, consumes less memory, and is less prone to overfitting. Additionally, it can make the proposed model simpler, faster, and suitable for real-time forecasting. The output of CNN (features) and BiLSTM (states) is concatenated and passed to multi-head attention. The multi-head attention decides which timesteps or attributes are crucial for PV power forecasting. It assigns different weights to each timestep in the fused features of the BiLSTM-CNN. As the name suggests, multi-head attention consists of multiple heads that are working in parallel. Each head captures a different feature from the BiLSTM-CNN output. All these heads are then combined, representing a comprehensive fusion of various learned relationships. The output of the multi-head attention layer is passed to the GlobalAveragePooling1D layer, which converts the entire temporal sequence into a fixed-length vector. The multi-head attention and GlobalAveragePooling1D layers provide both focus and performance. The multi-head layer identifies the important features, while the GlobalAveragePooling1D layer compresses this information into a compact representation. This compact representation is then passed to the Dense layer for PV power forecasting.

As shown in Fig. 5.6, ADHHO is employed for hyperparameter optimization of the proposed dual-stream attention-based hybrid model. ADHHO proposes adaptive search strategies that integrate chaotic optimization, a nonlinear control parameter, an adaptive inertia weight, and an adaptive decay factor to enhance convergence and robustness. Compared to other HHO variants, the proposed ADHHO algorithm improved all three phases of HHO. In the exploration, chaotic maps are utilized to enhance the search range and avoid clustering. A tent chaotic map is used to initialize the hawk populations. This chaotic map uniformly distributes solutions over the search space. Similarly, a logistic chaotic map is used to generate random numbers. Next is the transition phase, which is highly influenced by the value of escaping energy. For a smooth transition, a square-based function escape energy is used. The third phase (exploitation phase) is crucial for fine-tuning solutions, accelerating convergence, and ensuring excellent results. Hence, the proposed ADHHO can be highly effective for optimizing the hyperparameters of the BiLSTM-CNN-MHA. Since our model consists of two layers of CNN branch and two of BiLSTM branch, we selected six parameters in BiLSTM-CNN-MHA. Hence, the hyperparameter search includes CNN filter 1, CNN filter 2, LSTM units 1, LSTM units 2, batch size, and number of heads for multi-head attention.

5.6 Baseline Models for Analysis and Comparison

The performance of the proposed model is compared with that of the single and hybrid models. In the single model, we used a statistical model (LR), a machine learning model (SVR), and multiple deep learning models: MLP, CNN, LSTM, and BiLSTM, as listed in Tab. 5.2. In hybrid models, we employed different combinations of deep learning models, including CNN-BiLSTM, BiLSTM-CNN, BiLSTM-CNN-MHA, and HHO-optimized BiLSTM-CNN-MHA, as mentioned in Tab. 5.3. Hence, a total of ten models are used for analysis and comparison, including six single models and four hybrid models. All these models are created in Python 3.10.0, using the TensorFlow library version 2.10.0. For fair comparison, all DLMs, including single and hybrid, used the same activation function (ReLU), same learning rate (0.001), same optimizer (Adam), same

Table 5.2: Detailed description of the single baseline models for comparison

Types	Model	Hidden layers	Number of neurons	Activation	Flatten/Pooling	Number of heads	Final Layers	Descriptions
Single Models	LR	-	-		-	-	Output only	• Employ a normal least squares method • Presents linear relations between the input attributes and the predicted outputs
	SVR	-	-		-	-	Regression function	• Kernel function: “rbf” • Regularization parameter (C): 1.0 • An epsilon-insensitive margin: 0.1
	MLP	4	16, 32, 64, 32	ReLU	-	-	Dense(8)+Dense(1)	-
	CNN	4	16, 32, 64, 32	ReLU	Flatten	-	Dense(8)+Dense(1)	• Kernel size: 2, pool size: 2, strides: 1
	LSTM	4	16, 32, 64, 32	ReLU	-	-	Dense(8)+Dense(1)	-
	BiLSTM	4	16, 32, 64, 32	ReLU	GlobalAveragePooling1D	-	Dense(8)+Dense(1)	-

Table 5.3: Detailed description of the hybrid baseline models for comparison

Types	Model	Hidden layers	Number of neurons	Activation	Flatten/Pooling	Number of heads	Final Layers	• Descriptions
Hybrid Models	CNN-BiLSTM	4	16, 32, 64, 32	ReLU	Flatten + GlobalAveragePooling1D	-	Dense(8)+ Dense(1)	• Dual stream • Two layers of CNN and two layers of BiLSTM
	BiLSTM-CNN	4	16, 32, 64, 32	ReLU	Flatten + GlobalAveragePooling1D	-	Dense(8)+ Dense(1)	• Dual stream • Two layers of BiLSTM and two layers of CNN
	BiLSTM-CNN-MHA	4	16, 32, 64, 32	ReLU	Flatten + GlobalAveragePooling1D	4	Dense(8)+ Dense(1)	• Dual stream • Two layers of BiLSTM, two layers of CNN • Multi-head attention layer + Global Average Pooling
	HHO-BiLSTM-CNN-MHA	4	HHO-Optimized Units and filters	ReLU	Flatten + GlobalAveragePooling1D	HHO-Optimized heads	Dense(8)+ Dense(1)	• Dual stream • Two layers of BiLSTM, two layers of CNN • Multi-head attention layer + Global Average Pooling

epochs (20), and same batch size (64); though, HHO and ADHHO optimized models employed the optimized hyperparameters. Additionally, all these models are analyzed on three different step-ahead forecasts: 1-step ahead, 3-step ahead, and 6-step ahead.

5.7 Performance Metrics

The performance metrics are also known as evaluation metrics or error metrics. These metrics are important for analysing model performance and optimization [17]. They suggest the optimal values for model hyperparameters and network configurations. These metrics give a fair comparison between various forecasting models and optimization algorithms. Additionally, these metrics may identify certain shortcomings in the model's forecasting. Moreover, these metrics evaluate the accuracy with which the trained models predict values from the testing dataset. In the literature, multiple performance metrics are used to analyze the performance of DLMs. However, in this work, we employed six performance metrics to compare the proposed hybrid model's performance with other studied models. These include RMSE, MSE, MAE, NRMSE, MBE, and R^2. These six metrics are combined to give a detailed examination of the studied models. For instance, RMSE and MSE metrics penalize large errors more severely. This feature is particularly beneficial in PV power forecasting, where abrupt fluctuations in weather-related variables can significantly impact the grid operations [17]. MAE determines the average magnitude of the error between the estimated and true values. This metric treats all errors equally, which makes it less prone to outliers. A low value of MAE shows that the estimated PV value is close to the true value of PV power output, suggesting high precision and performance. This enables power companies to make more predictable decisions regarding operations. On the other hand, a high MAE indicates a poor model design, an insufficient dataset, or erratic weather conditions.

The MBE metric gives insight into a model's long-term performance. It shows the tendency of the proposed model to under- or overestimate the predicted values [150]. The value of MBE can be positive or negative. A positive MBE indicates that the forecasting model has overestimated the PV power production. This can lead

to operational inefficiencies and supply shortages. Overestimated values indicate overreliance on power output from PV power plants when auxiliary backup sources are required. On the other hand, a negative MBE indicates that the forecasting model has underestimated the PV power production. Underestimated values can lead to excessive energy storage or dependence on auxiliary power sources, thus increasing operational costs. Therefore, the MBE helps grid operators to detect systematic bias in PV power predictions, which can improve energy scheduling. Likewise, R^2 evaluates the overall fitness of DLMs for a specific dataset. It reflects the capacity of DLMs to explain the variation in the PV dataset. The value of R^2 can vary between 1 and 0. A value of 1 means that DLMs precisely capture all variations in the PV power dataset. On the other hand, a value of 0 indicates that DLMs are unable to explain the variations in the PV dataset, resulting in inaccurate forecasting. Hence, this metric is useful for grid operators to assess the reliability and feasibility of DLMs. Similarly, NRMSE is the normalized version of RMSE [151], allowing comparison of errors across different datasets and regions. NRMSE is essential for adapting DLM to the various areas and power systems. Mathematically, all six performance metrics can be expressed as [151], [152],

$$\text{MAE} = \left. \sum_{i=1}^{N} |y_{ai} - y_{fi}| \right/ N \tag{5.2}$$

$$\text{MSE} = \left. \sum_{i=1}^{N} (y_{ai} - y_{fi})^2 \right/ N \tag{5.3}$$

$$\text{MBE} = \left. \sum_{i=1}^{N} (y_{ai} - y_{fi}) \right/ N \tag{5.4}$$

$$\text{RMSE} = \left. \sqrt{\left(\sum_{i=1}^{N} \left(y_{ai} - y_{fi}\right)^2\right)} \right/ N \tag{5.5}$$

$$R^2 = 1 - \left. \left(\sum_{i=1}^{N} (y_{ai} - y_{fi})^2\right) \right/ \left(\sum_{i=1}^{M} (y_{ai} - \overline{Mean})^2\right) \tag{5.6}$$

$$\text{NRMSE} = \left. RMSE \times 100\% \right/ (y_{max} - y_{min}) \tag{5.7}$$

In the about equations, $y_{ai}, y_{fi}, \overline{mean}, y_{max}$, and y_{min} represent the true, predicted, mean, minimum, and maximum values of PV power output, respectively. N denotes the sample size.

5.8 Results and Discussions

This section provides a comprehensive assessment of the proposed model's performance. The Alice Springs dataset (90% for training and 10% for testing) is used for assessment purposes. The performance of the proposed model is assessed for 1-step-ahead, 3-step-ahead, and 6-step-ahead forecasting. The results on each step-ahead forecasting are compared with both single and hybrid baseline models discussed in Section 5.6. These include LR, SVR, MLP, CNN, LSTM, BiLSTM, CNN-BiLSTM, BiLSTM-CNN, BiLSTM-CNN-MHA, and HHO optimized BiLSTM-CNN-MHA. In addition, six performance metrics—MSE, MAE, RMSE, R^2, MBE, and NRMSE—were used to evaluate the comparative outcomes of the anticipated and actual values of PV power production. Tab. 5.4 presents the results of the studied models for 1-step-ahead forecasting. The best measures on each performance metric are shown in bold. The table indicates that the proposed model exhibits the highest predictive accuracy across most performance metrics. The ADHHO-BiLSTM-CNN-MHA attained an MSE value of 0.00323, an RMSE value of 0.0568, an R^2 value of 0.9671, an MBE value of 0.0058, and an NRMSE value of 0.0597. On the MAE metric, HHO-optimized BiLSTM-CNN-MHA achieved the lowest value of 0.023. The HHO-BiLSTM-CNN-MHA is the second most effective in terms of accuracy and performance. However, on the MBE metric, BiLSTM attained the best score of -0.0026. Overall, the results of the 1-step-ahead forecasting indicate that all hybrid models, optimized or unoptimized, outperform the single models on almost all evaluation metrics. To improve visualization and highlight the superiority of the ADHHO-optimized dual-stream attention-based hybrid model compared to the baseline models, Fig. 5.7 illustrates a Radar chart for all performance metrics. The Radar chart provides a better visual comparison of multiple performance metrics simultaneously. It provides a simple

	Model	MSE (kW)	MAE (kW)	RMSE (kW)	R2 -	MBE (kW)	NRMSE -
1-timestep (5 minutes)	Linear Regression	0.00550	0.0507	0.0743	0.9438	-0.0277	0.0780
	SVR	0.00510	0.0485	0.0714	0.9481	-0.0134	0.0750
	MLP	0.00475	0.0374	0.0670	0.9534	0.0249	0.0710
	CNN	0.00370	0.0244	0.0609	0.9623	-0.0036	0.0639
	LSTM	0.00382	0.0241	0.0618	0.9611	-0.0053	0.0649
	BiLSTM	0.00384	0.0245	0.0620	0.9608	**-0.0026**	0.0651
	CNN-BiLSTM	0.00369	0.0238	0.0607	0.9624	-0.0043	0.0638
	BiLSTM-CNN	0.00369	0.0238	0.0607	0.9624	-0.0036	0.0638
	BiLSTM-CNN-MHA	0.00355	0.0252	0.0596	0.9638	0.0113	0.0626
	HHO-BiLSTM-CNN-MHA	0.00343	**0.0231**	0.0586	0.9650	0.0047	0.0615
	Proposed Model	**0.00323**	0.0238	**0.0568**	**0.9671**	0.0058	**0.0597**

Table 5.4: One-step ahead forecasting results of the studied models

way to determine which model is better overall. Since we normalized the value for the Radar chart, the lower the value of the performance metrics, the closer their graphs are to the outer edges. In contrast, the higher the value of the performance metrics, the closer their graphs are to the center. As shown in Fig. 5.7, the proposed model achieves lower values on most performance metrics; thus, its graph appears near the edges of the radar chart.

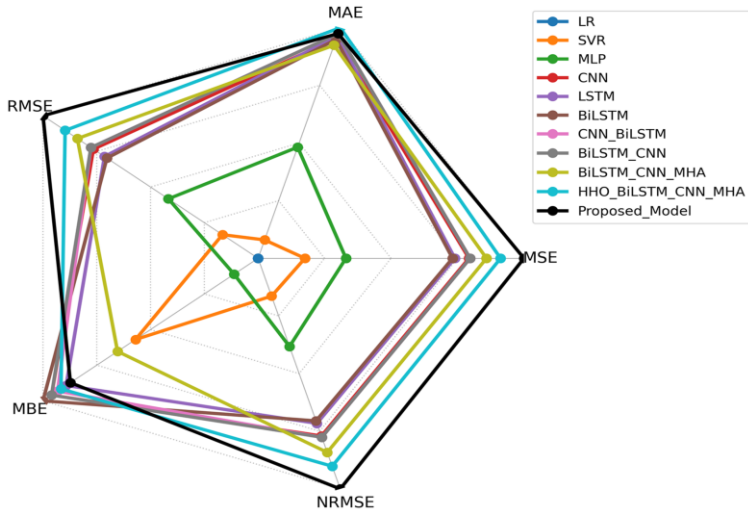


Figure 5.7: Radar chart of the studied models using 1-step ahead forecasting

Fig. 5.8 illustrates the graphical representation of the actual and predicted values of the studied models for PV power output. Overall, the figure shows that all forecasting models closely follow the actual PV power output. LR (statistical model) and SVR (MLM) demonstrate poor performance on 1-step-ahead forecasting, and their predicted values significantly deviate, particularly for zero and peak PV power values, as indicated in blue and green colours. For better visualization and a clearer understanding, we selected two different days, sunny and cloudy, for analysis from Fig. 5.8. A portion of both these figures is further magnified to enhance visibility. Fig. 5.9 shows the forecasting results for a sunny day, which exhibits some fluctuations at the start of the day. The studied models, particularly the single models, struggle to capture these fluctuations. For the sunny day forecast, LR largely overestimated, and MLP underestimated the PV power output during the peak times from about 9:30 to almost 4:00 pm. The over- and underestimated PV power output can create operational and economic issues for the utility grids. It can significantly impact the scheduling of energy storage and generators, resulting in higher operational expenses and reduced reliability of the power system. Hence, these models may not be suitable for complex PV datasets that exhibit high fluctuations. On the other hand, the proposed model somehow provides superior predictive performance at the steep curves. The proposed model closely followed the actual PV power output for most of the time, leading to improved accuracy and performance. Fig. 5.10 shows the forecasting results for a cloudy day, which exhibit fluctuations for the entire day. On a cloudy day, MLP and BiLSTM-CNN-MHA underestimated, while LR overestimated the PV power output. There are some fluctuations at the start of the day, and almost all models fail to capture these fluctuations. Overall, all studied models perform well on cloudy day forecasts; however, as shown in the magnified portion of the figure, the proposed model almost closely followed the actual data for most of the time. Fig. 5.11 shows a comparison of the regression analysis for all studied models. It explains how well the predicted values match the actual values. When the predicted values strongly correlated with the actual values, they formed a cluster near the regression line. As shown in the figure, most of the values are near the regression line, which indicates that the studied models are performing very well. Hence, the results of the 1-step forecasting indicate that all forecasting models achieved a reasonable accuracy; however, hybrid models, particularly the

ADHHO-optimized dual-stream attention-based hybrid model, attained the best scores and accuracy on most of the performance indicators.

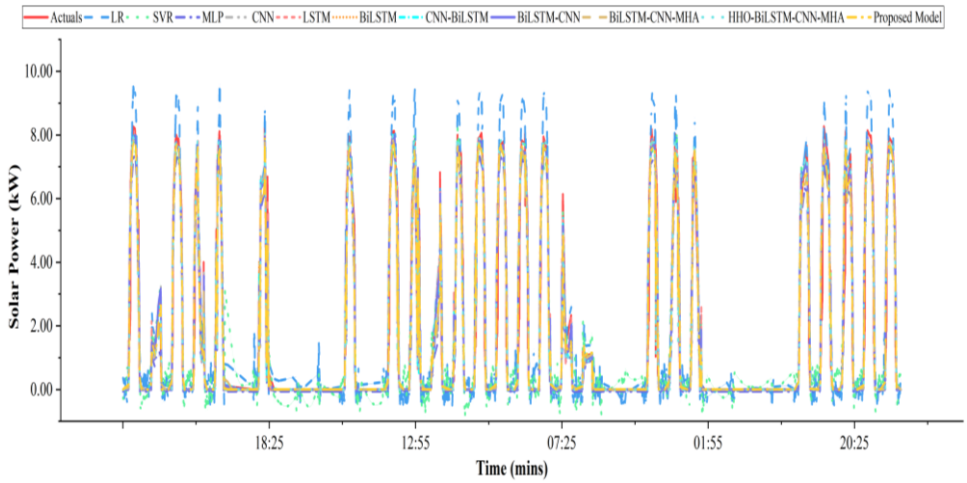


Figure 5.8: Comparison of the studied models' predictive capabilities using 1-step ahead forecasting

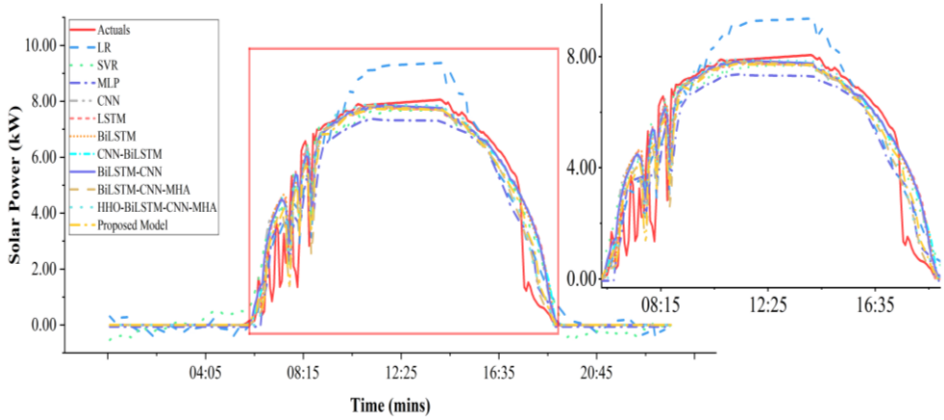


Figure 5.9: Comparison of studied models on a sunny day from 1-step ahead forecasting

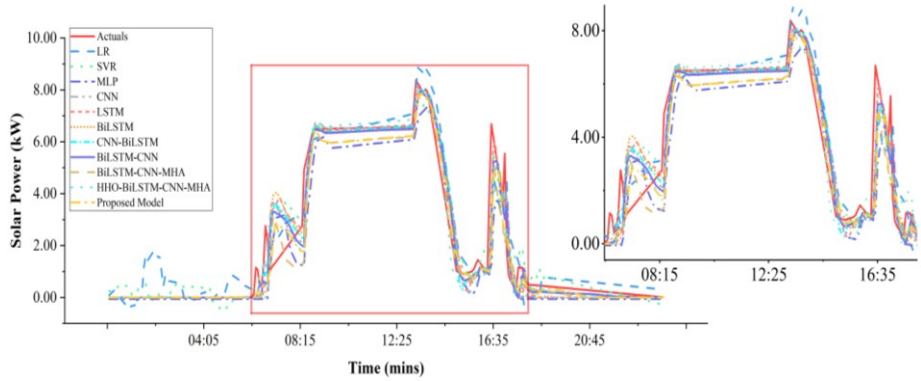


Figure 5.10: Comparison of studied models on a cloudy day from 1-step ahead forecasting

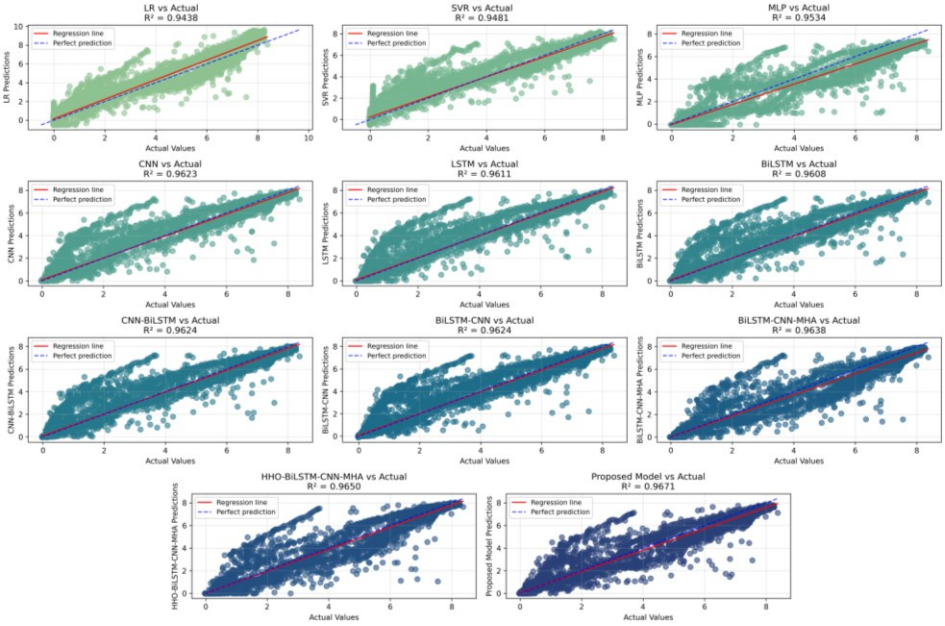


Figure 5.11: Regression analysis between forecasted and observed values of the studied models on a 1-step ahead forecasting

The performance of the proposed models is also tested on 3-step-ahead forecasting, and the results are presented in Tab. 5.5. The findings on 3-step-ahead forecasting indicate that ADHHO-BiLSTM-CNN-MHA outperforms other studied models on five out of six evaluation metrics; nonetheless, HHO-optimized BiLSTM-CNN-MHA excelled on two metrics (MAE and MBE). The proposed model achieved an MSE of 0.00338, an RMSE of 0.0581, an R^2 of 0.9656, and an NRMSE of 0.0610. On the MAE performance metric, both ADHHO-BiLSTM-CNN-MHA and HHO-BiLSTM-CNN-MHA attained the best score of 0.0233, whereas on the MBE metric, the HHO-optimized BiLSTM-CNN-MHA outperforms all models, attaining the lowest value of 0.00004. The HHO-optimized BiLSTM-CNN-MHA is also the second best performing model after the proposed model, followed by BiLSTM-CNN-MHA. Among the single models, BiLSTM achieved better scores on all performance metrics, even on MAE; it also outperformed hybrid models, such as CNN-BiLSTM, BiLSTM-CNN, and BiLSTM-CNN-MHA. From the table, it is evident that hybrid models achieved better performance on most metrics compared to single models due to their capacity to model both spatial and temporal dependencies, particularly on complex datasets.

This can also be validated from the Radar chart, as illustrated in Fig. 5.12. The figure shows that the proposed model, mentioned in black, attained the lowest error rates on most metrics; its chart appears near the edges of the radar chart. The chart for the second best performing model, HHO-optimized BiLSTM-CNN-MHA, lies just below the proposed model, mentioned in cyan. On the other hand, the statistical model (LR) and machine learning model (SVR) perform poorly, producing high error rates on all metrics; their charts appear at the center of the chart.

	Model	MSE (kW)	MAE (kW)	RMSE (kW)	R2 -	MBE (kW)	NRMSE -
3-timesteps (15 minutes)	Linear Regression	0.00586	0.0517	0.0766	0.9402	0.0225	0.0803
	SVR	0.00544	0.0487	0.0737	0.9446	0.0119	0.0774
	MLP	0.00411	0.0319	0.0641	0.9581	-0.0032	0.0672
	CNN	0.00402	0.0251	0.0634	0.9591	0.0038	0.0666
	LSTM	0.00395	0.0247	0.0628	0.9598	0.0029	0.0659
	BiLSTM	0.00386	0.0245	0.0621	0.9607	0.0016	0.0652
	CNN-BiLSTM	0.00387	0.0292	0.0622	0.9607	-0.0131	0.0653
	BiLSTM-CNN	0.00385	0.0265	0.0620	0.9608	-0.0102	0.0651
	BiLSTM-CNN-MHA	0.00376	0.0256	0.0613	0.9617	-0.0012	0.0644
	HHO-BiLSTM-CNN-MHA	0.00366	**0.0233**	0.0605	0.9627	**0.00004**	0.0634
	Proposed Model	**0.00338**	**0.0233**	**0.0581**	**0.9656**	-0.0027	**0.0610**

Table 5.5: Three-step ahead forecasting results of the studied models

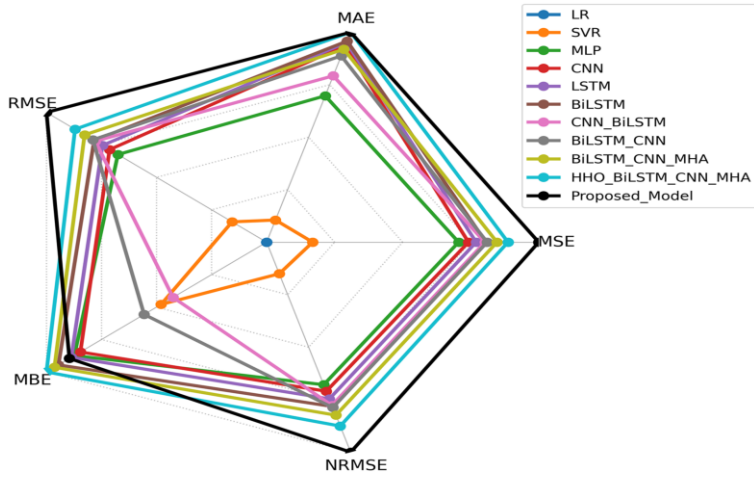


Figure 5.12: Radar chart of the studied models using 3-step ahead forecasting

Fig. 5.13 provides the performance of all studied models on 3-step-ahead (15-minute) forecasting. Similar to 1-step-ahead forecasting, the LR and SVR exhibit high variability near the peak and zero PV power values. All DLMs, both single and hybrid, followed the general patterns of the actual PV power. To closely visualize the forecasted results at 3-step-ahead forecasting, we used the same sunny and cloudy days for analysis, as displayed in Fig. 5.14 and 5.15. The actual PV power exhibits some fluctuations for a sunny day, as shown in Fig. 5.14. In this case, both MLP and SVR underestimated, and LR overestimated the PV power output during the peak times from about 9:30 to almost 4:00 pm. From the magnified portion of the sunny day, it can be seen that the proposed model very closely followed the actual PV values during the peak hours. Although the proposed model didn't capture some fluctuations at the start, it still outperforms both single and hybrid models. Similarly, the forecasting results of all studied models on a cloudy day are presented in Fig. 5.15. On this day, CNN-BiLSTM underestimated, while LR overestimated the PV power output. Overall, the proposed model demonstrates better performance on the entire day forecast, except for fluctuations from 6:00 am to 8:15 am. On these fluctuations, most of the studied models didn't follow the actual PV power values, as shown in the magnified portion of the cloudy day. Furthermore, Fig. 5.16 displays the regression plot for 3-step-ahead forecasting. The figure shows that the actual and predicted values are highly correlated, forming a cluster near the regression line. This indicates that all studied models have precisely captured complex and nonlinear patterns in the PV dataset. However, the figure shows that the proposed model provides a better fit, particularly in the areas where the PV power exhibits significant nonlinearity. Thus, the results of the 3-step-ahead forecasting demonstrate that the ADHHO-BiLSTM-CNN-MHA achieved superior performance and precision compared to all studied models.

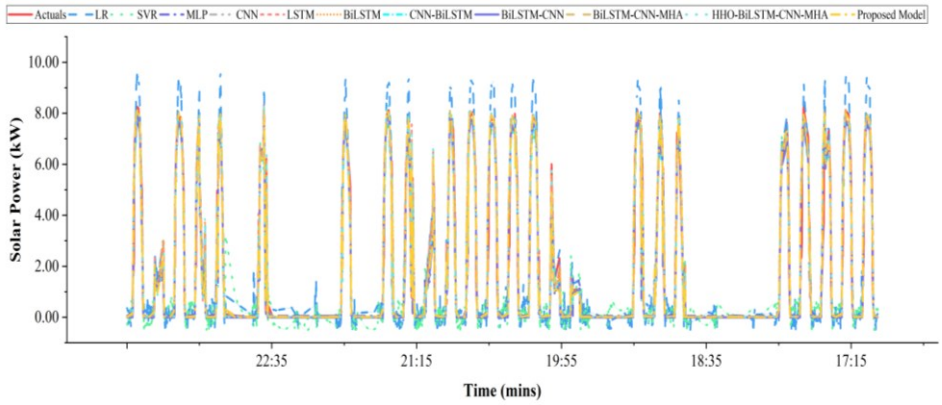


Figure 5.13: Comparison of the studied models' predictive capabilities using 3-step ahead forecasting

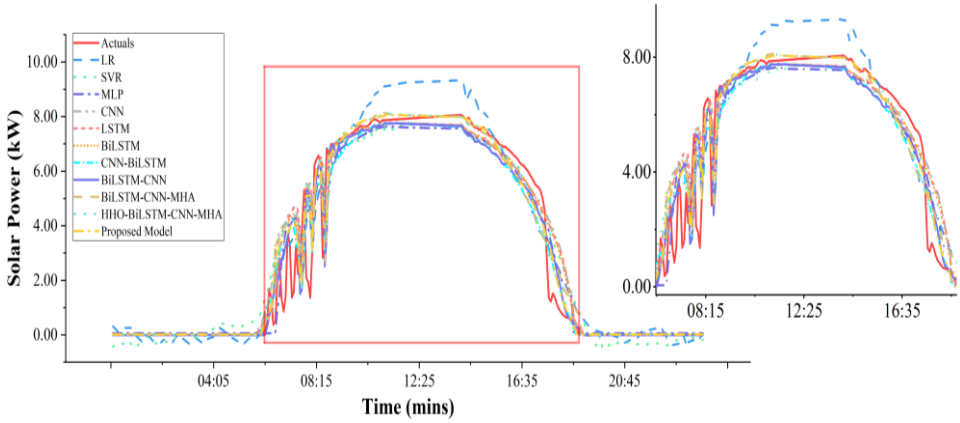


Figure 5.14: Comparison of the studied models on a sunny day from 3-step ahead forecasting

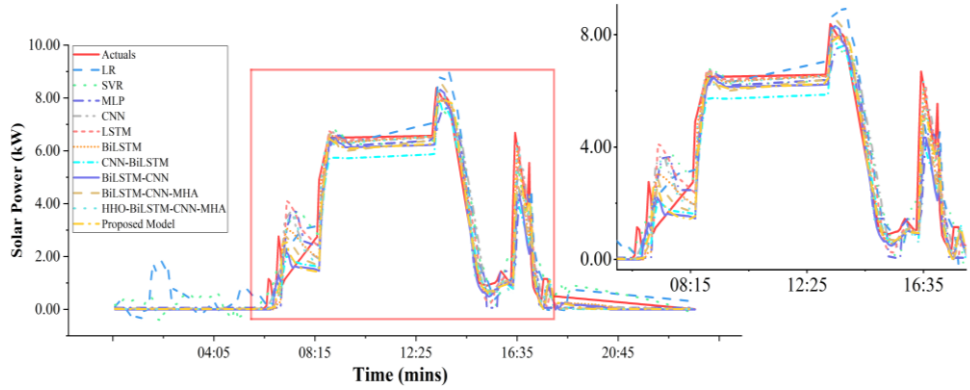


Figure 5.15: Comparison of the studied models on a cloudy day from 3-step ahead forecasting

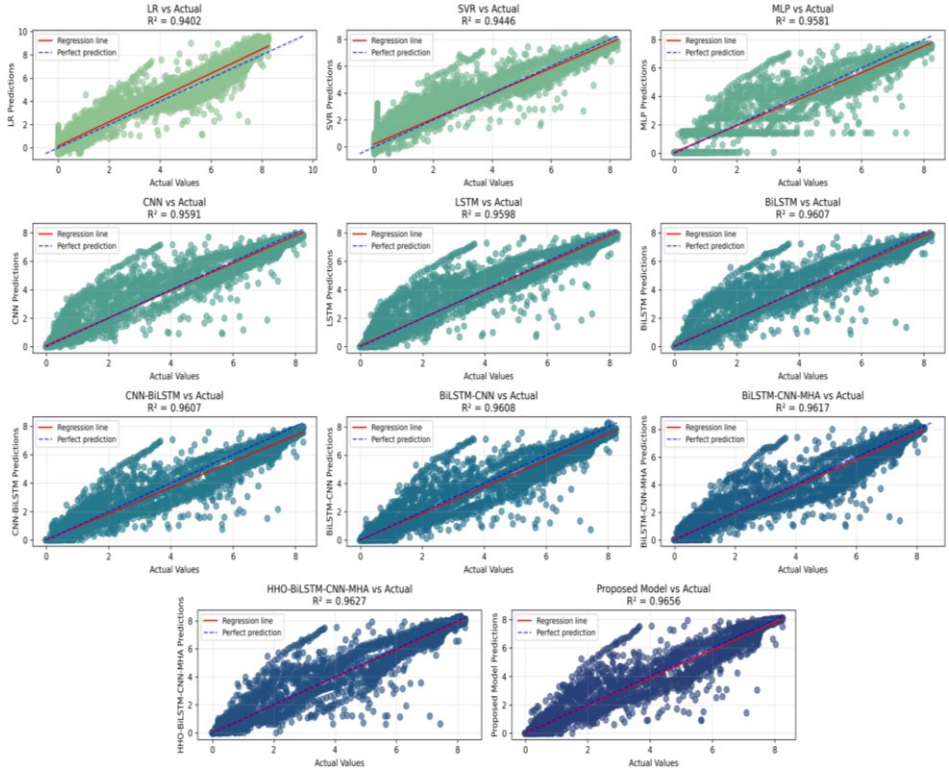


Figure 5.16: Regression analysis between forecasted and observed values of the studied models on a 3-step ahead forecasting

Next, we further increase the forecasting horizon to 6 timesteps (30 minutes) and assess the performance of the studied models. The findings on 6-step ahead forecasting are presented in Tab. 5.6. The table indicates that error rates increase with the increase in the forecasting horizons. The reason for this can be that increasing the forecasting horizons reduces the temporal relationship between the weather variables and PV power output. This makes it more challenging for the studied models to capture sudden variations and changes. However, the table's findings indicate that despite the higher forecasting horizon, the proposed model outperforms all other studied models. The ADHHO-optimized BiLSTM-CNN-MHA achieved the lowest error rates on four out of six evaluation metrics, while both HHO-BiLSTM-CNN-MHA and CNN attained the best score on one metric. The proposed ADHHO-BiLSTM-CNN-MHA yielded the lowest MSE, RMSE, R^2, and NRMSE values of 0.00392, 0.0626, 0.9600, and 0.0657, respectively.

	Model	MSE (kW)	MAE (kW)	RMSE (kW)	R2 -	MBE (kW)	NRMSE -
6-timesteps (30 minutes)	Linear Regression	0.00637	0.0532	0.0798	0.9351	0.0223	0.0837
	SVR	0.00609	0.0515	0.0780	0. 9379	0.0164	0.0819
	MLP	0.00482	0.0295	0.0694	0.9508	-0.0057	0.0729
	CNN	0.00444	0.0270	0.0666	0.9548	**-0.0002**	0.0699
	LSTM	0.00451	0.0269	0.0672	0.9540	0.0022	0.0705
	BiLSTM	0.00449	0.0269	0.0670	0.9543	-0.0041	0.0703
	CNN-BiLSTM	0.00442	0.0271	0.0665	0.9550	0.0009	0.0698
	BiLSTM-CNN	0.00430	0.0262	0.0656	0.9562	0.0019	0.0688
	BiLSTM-CNN-MHA	0.00422	0.0294	0.0650	0.9570	-0.0063	0.0682
	HHO-BiLSTM-CNN-MHA	0.00403	**0.0259**	0.0634	0.9590	0.0009	0.0666
	Proposed Model	**0.00392**	0.0265	**0.0626**	**0.9600**	-0.0041	**0.0657**

Table 5.6: Six-step ahead forecasting results of the studied models

Similarly, the HHO-optimized BiLSTM-CNN-MHA achieved the best score on MAE (0.0259), while CNN attained the lowest score of -0.0002 on the MBE metric. The findings show that ADHHO-BiLSTM-CNN-MHA outperforms other studied models in forecasting accuracy. The proposed model is followed by HHO-BiLSTM-CNN-MHA and BiLSTM-CNN-MHA, which attained somehow slightly higher values than the proposed model. Among the single models, CNN outperforms other models, producing lower error rates. In contrast, LR and SVR provide higher forecasting errors, as indicated in the Radar chart for 6-step ahead forecasting. The Radar chart, in Fig. 5.17, shows that the LR achieved the highest error rates; hence, its chart is located at the center. SVR is the second-worst performer; its chart is located just above the center. In contrast, the proposed model yielded the lowest errors; its chart is located near the outer edges of the chart, followed by HHO-BiLSTM-CNN-MHA and BiLSTM-CNN-MHA.

Fig. 5.18 presents a comparison of all studied models on the 6-step (30-minute) forecasting. Both LR and SVR exhibited high variability near the peak and zero PV power values. These models may not be capable of capturing nonlinear behavior in the PV dataset. Overall, all forecasting models followed the general patterns of the actual PV power. Similar to 1-step and 3-step forecasting, we also selected the same sunny and cloudy days for analysis. Fig. 5.19 and 5.20 present the forecasting performance of the studied models on these days. On a sunny day, BiLSTM-CNN-MHA underestimated, and LR overestimated the PV power output

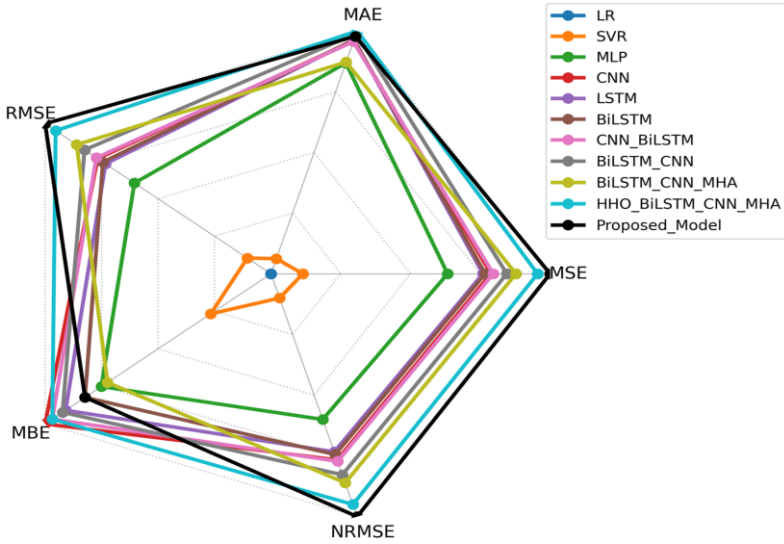


Figure 5.17: Radar chart of the studied models using the 3-step ahead forecasting

during the peak times from about 9:30 to almost 4:00 pm. Again, most of the models didn't capture the initial fluctuations in PV power output; however, all these models perform well on the rest of the day. The magnified portion of the figure shows that the proposed model mostly captures the actual PV power output. Compared to other models, it exhibits fewer deviations from the actual values. Similarly, a comparison of all studied models on a cloudy day is illustrated in Fig. 5.20. On this day, MLP and BiLSTM-CNN-MHA underestimated, while LR overestimated the PV power output. All forecasting models perform well on this day, except for fluctuations from 6:00 am to 8:15 am. However, as illustrated in the magnified portion, the proposed ADDHO-BiLSTM-CNN-MHA very closely followed the actual values during peak hours and steep curves. This indicates that the proposed model performs even better as the forecasting horizons increase. Furthermore, Fig. 5.21 presents the regression plot for 6-step-ahead forecasting. The figure reveals that, in this case, the actual and predicted values are slightly scattered compared to 1-step and 3-step forecasting. However, they still formed a cluster near the regression line,

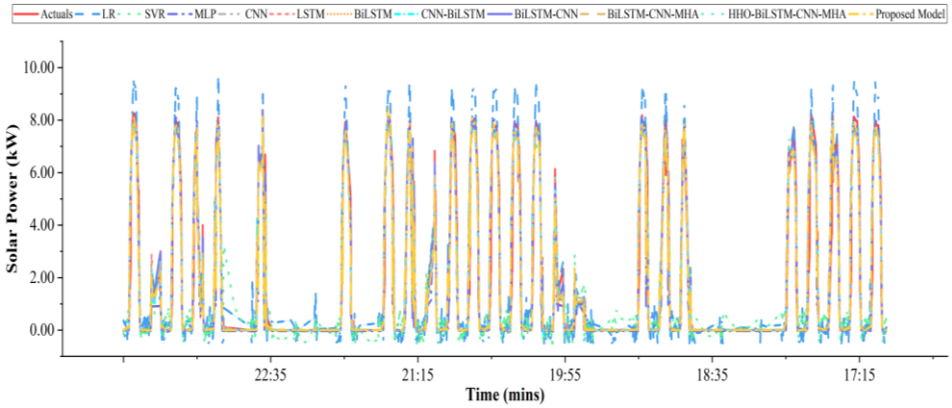


Figure 5.18: Comparison of the studied models' predictive capabilities using 6-step ahead forecasting

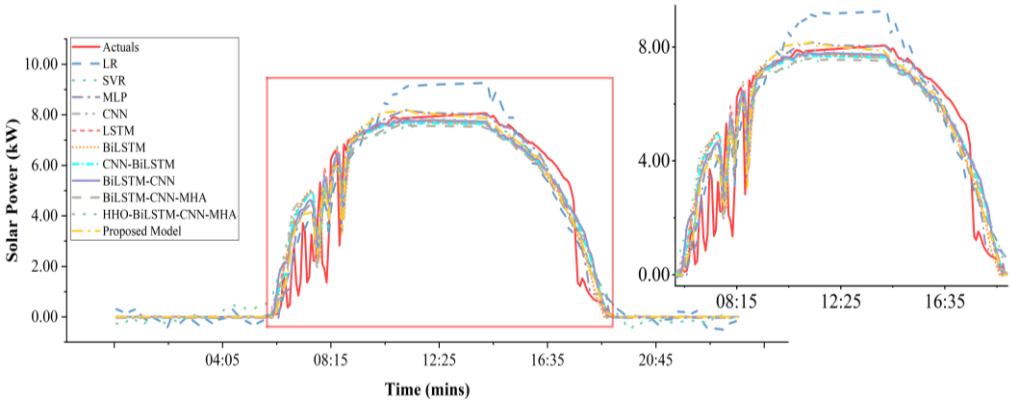


Figure 5.19: Comparison of the studied models on a sunny day from 6-step ahead forecasting

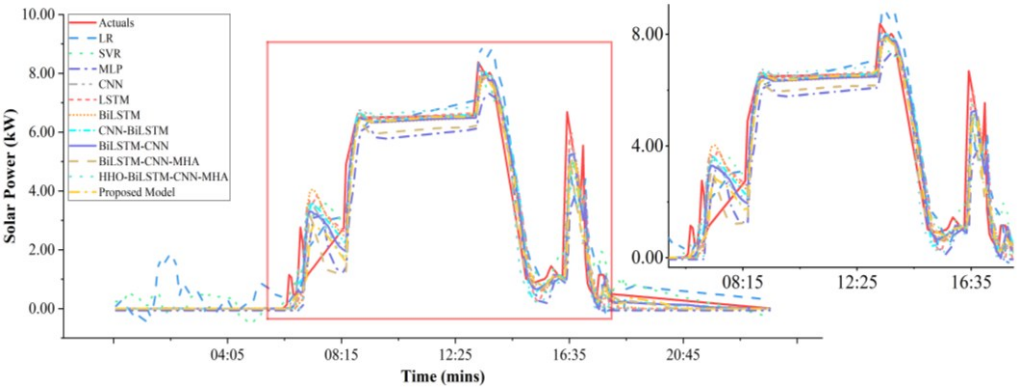


Figure 5.20:Comparison of the studied models on a cloudy day from 6-step ahead forecasting

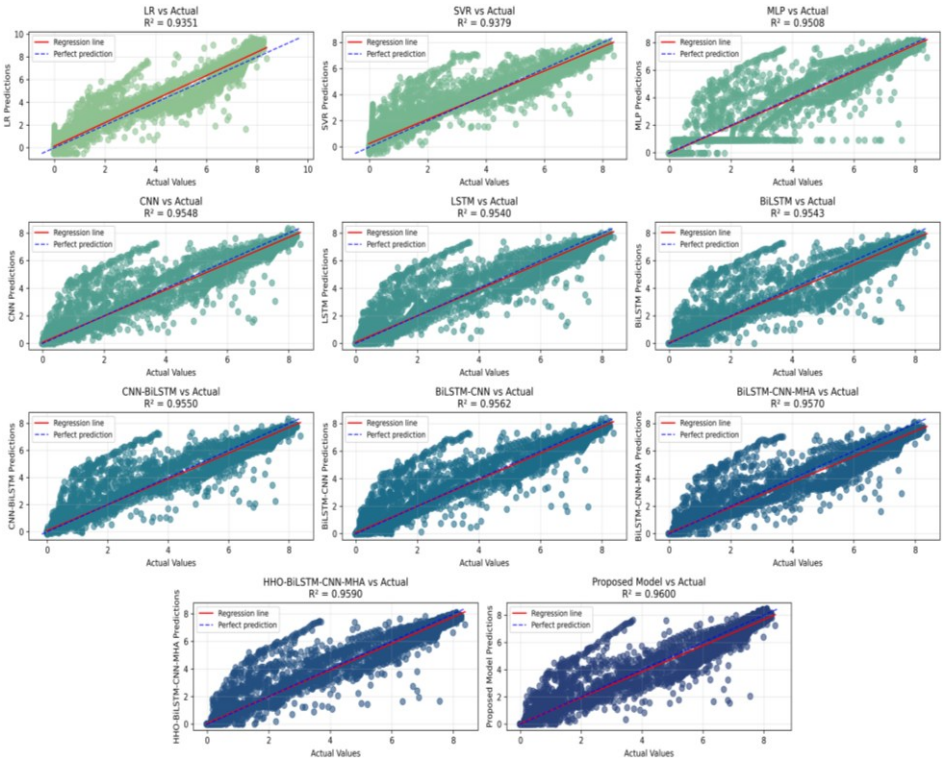


Figure 5.21: Regression analysis between forecasted and observed values of the studied models on a 6-step ahead forecasting

indicating that these models can capture complex and nonlinear patterns in the PV dataset. The higher value of the coefficient of determination reveals that the proposed model provides a better fit, particularly in the areas where the PV power exhibits significant nonlinearity. Consequently, the findings of the 6-step-ahead forecasting illustrate that the proposed model outperforms all studied models, even with high forecasting horizons, achieving superior performance and accuracy.

5.9 Summary

In this chapter, the performance of the proposed ADHHO-optimized BiLSTM-CNN-MHA is tested and validated for multi-step ahead forecasting. The openly available dataset from site 1A in Alice Springs, Australia, is used for experimental

analysis. The site 1A dataset (Trina, 10.5 kW, mono-Si, Dual, 2009) has an installed capacity of 10.5 kW, and the PV modules of this site are mounted on dual-axis trackers. The chosen dataset contains various attributes; however, for our analysis, we select PV power and six variables that show strong positive and negative correlations with PV power. The six variables are global horizontal radiation (W/m^2), diffuse horizontal radiation (W/m^2), temperature (°C), relative humidity (%), wind speed (m/sec^2), and wind direction (°). The performance of the proposed model is assessed for 1-step-ahead, 3-step-ahead, and 6-step-ahead forecasting. The results of each step-ahead forecasting are compared with both single and hybrid baseline models. These include LR, SVR, MLP, CNN, LSTM, BiLSTM, CNN-BiLSTM, BiLSTM-CNN, BiLSTM-CNN-MHA, and HHO optimized BiLSTM-CNN-MHA. In addition, six performance metrics—MSE, MAE, RMSE, R2, MBE, and NRMSE—were used to evaluate the comparative outcomes of the anticipated and actual values of PV power production.

On 1-step ahead forecasting, the proposed model demonstrates the highest predictive accuracy on four out of six performance metrics. The ADHHO-BiLSTM-CNN-MHA attained an MSE value of 0.00323, an RMSE value of 0.0568, an R^2 value of 0.9671, an MBE value of 0.0058, and an NRMSE value of 0.0597. On the MAE metric, HHO-optimized BiLSTM-CNN-MHA achieved the lowest value of 0.023. The HHO-BiLSTM-CNN-MHA is the second most effective in terms of accuracy and performance. However, on the MBE metric, BiLSTM attained the best score of -0.0026. On 3-step forecasting, the ADHHO-BiLSTM-CNN-MHA outperforms other studied models on five out of six evaluation metrics; nonetheless, HHO-optimized BiLSTM-CNN-MHA excelled on two metrics (MAE and MBE). The proposed model achieved an MSE of 0.00338, an RMSE of 0.0581, an R^2 of 0.9656, and an NRMSE of 0.0610. On the MAE performance metric, both ADHHO-BiLSTM-CNN-MHA and HHO-BiLSTM-CNN-MHA attained the best score of 0.0233, whereas on the MBE metric, the HHO-optimized BiLSTM-CNN-MHA outperforms all models, achieving the lowest value of 0.00004. Similarly, on 6-step forecasting, the ADHHO-optimized BiLSTM-CNN-MHA achieved the lowest error rates on four out of six evaluation metrics, while both HHO-BiLSTM-CNN-MHA and CNN attained the best score on one metric. The proposed ADHHO-BiLSTM-CNN-MHA yielded the lowest MSE, RMSE, R^2, and NRMSE values of 0.00392, 0.0626, 0.9600, and 0.0657, respectively. Similarly, the HHO-

optimized BiLSTM-CNN-MHA achieved the best score on MAE (0.0259), while CNN attained the lowest score of -0.0002 on the MBE metric.

Overall, the results of the three experiments, including tables and figures, confirmed that the proposed ADHHO-optimized BiLSTM-CNN-MHA outperforms all single and hybrid models in forecasting accuracy. The proposed model is followed by HHO-BiLSTM-CNN-MHA and BiLSTM-CNN-MHA, which attained somehow slightly higher values than the proposed model. Additionally, all hybrid models, optimized or unoptimized, outperform the single models on almost all evaluation metrics. The Trina Solar plant has a real-world dataset that includes linear and nonlinear patterns, complex relationships, seasonality, and sudden weather fluctuations. The proposed model not only captures steady PV power outputs during sunny days but also nonlinear and complex patterns, seasonality, and sudden weather fluctuations during cloudy days. Hence, the proposed model can be adopted as a reliable tool for PV power forecasting, where accuracy, reliability, and stability are important.

6

Assessment of the Proposed Model on the ENEA Dataset

6.1. Description of ENEA and Energy Community

This chapter provides a detailed analysis of the proposed model on the ENEA dataset. ENEA, located in Rome, is an Italian research organization that operates in the field of energy, environment, and cutting-edge technology, promoting productivity and environmentally friendly policies. Initially, it was created to gather and spread the scientific knowledge related to the applications of nuclear energy. However, the organization's objective has gradually evolved to include renewable energy, energy efficiency, ecological research, and technology advancements to support industries and social growth. Currently, ENEA manages multiple research and testing facilities throughout Italy, providing advanced technical and operational support to government institutions, universities, and companies. Additionally, it maintains a major part in Italy's scientific and technical growth in the solar power sector. ENEA undertakes cutting-edge research in PV materials, including high-performance solar cells, thin-film technology, and integrated solar energy solutions. It also enhances the reliability of PV panels and optimizes the PV power plant performance through mathematical modelling, simulation, and various online monitoring systems.

Besides PV research, ENEA also supports Italy's energy strategy by developing tools for PV power forecasting and analyzing the impact of large-scale solar power integration on the utility grids. For instance, it inaugurated a smart community in Anguillara Sabazia, an Italian town located 32 km from Rome [153], which allows

users to produce, utilize, and exchange energy with the utility grid, thereby enhancing the standard of living and environmental sustainability. For that purpose, ENEA designed an online platform that serves as the brain of the smart community. Fig. 6.1 provides the basic working of the DHOMUS platform. As shown in the figure, it combines power generation, usage, communications, and control to regulate energy at both home and community levels. At the core of the DHOMUS platform is the intelligent home that includes a user device. The user device gathers information from two attached meters, M1 (Exchange Meter) and M2 (Production Meter), to decide the flow of energy inside the home. The M2 meter is connected to the output of the locally installed renewable energy sources, and it provides information about the total power available from these resources. The user device first utilizes the power available from PV power plants to meet the consumption requirements of the home. Any excess power is then stored in the battery or sent to the grid. Similarly, the M1 meter provides real-time data on the amount of energy taken or fed to the utility grid. When the available power from local renewable sources is not sufficient, the power is taken from the utility grid to meet the home's consumption requirements. On the other hand, when excess power is available from local renewable sources, it is sent back to the utility grid.

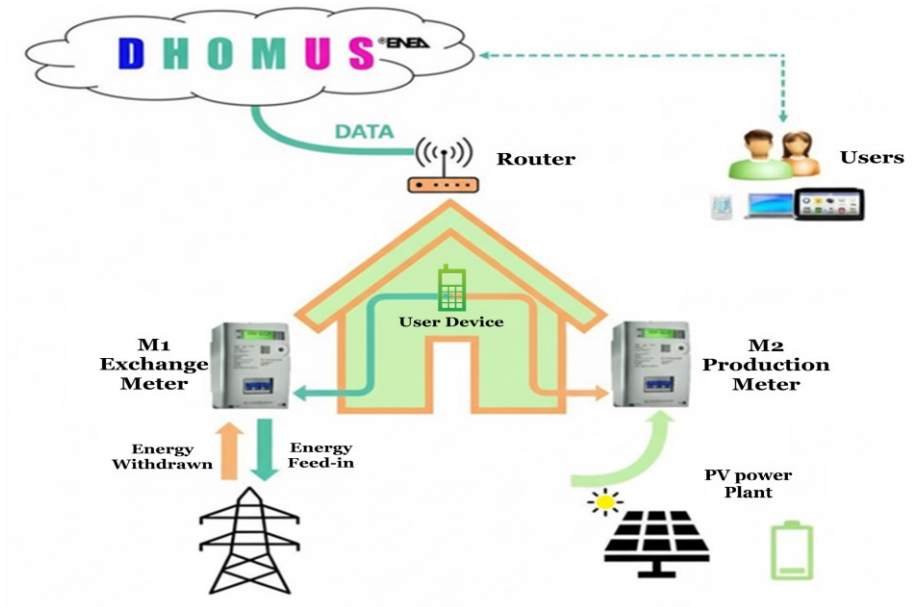


Figure 6.1: Basic working of the DHOMUS software for controlling the flow of information

Communication is essential in the energy community to optimize the flow of energy, engage participants, and achieve the common goal of energy conservation and self-sufficiency. In this case, the meters and user devices communicate with each other via PLCC, while the user devices and routers are connected through Wi-Fi. The router then sends the data from the user devices to the DHOMUS platform, which can be viewed by the end user for decision-making. The users can track real-time energy consumption and generation, compare expenses and savings, and continuously engage in energy conservation efforts. The platform also permits end users to coordinate, which is an important feature of the energy community. The DHOMUS platform collects data from multiple users in the energy community in Anguillara Sabazia. In this trial, ten users from the nearby community are selected for participation. The complete description of these ten users is presented in Tab. 6.1. As indicated in the table, some users (DU_1, DU_2, DU_4, DU_7, and DU_9) have an installed PV capacity and an M2 meter at their premises, and they act as prosumers. On the other hand, DU_3, DU_5, DU_6, DU_8, and DU_10 do not have an installed PV capacity and an M2 meter at their premises, and they act as consumers.

ID User	Total residents	Installed PV capacity	Tilt	Orientation (to the south)	Battery storage	M2 meter
Units	-	(kWp)	[°]	[°]	(kWh)	-
DU_1	2	6	29	20	-	Yes
DU_2	2	3.2	18	0	5	Yes
DU_3	3	4.8	8	120	-	No
DU_4	2	6	29	35	-	Yes
DU_5	2	-	-	-	-	No
DU_6	1	-	-	-	-	No
DU_7	2	3.45	25	45	-	Yes
DU_8	3	-	-	-	-	No
DU_9	1	3	30	40	-	Yes
DU_10	3	-	-	-	-	No

Table 6.1: Profiles of the individuals participating in the trial of the smart community

6.2. Data Collection and Visualization

The data employed for training and testing the proposed ADHHO-BiLSTM-CNN-MHA was obtained from ENEA in Rome. The dataset, spanning from January 1, 2024, to January 1, 2025, was used for analysis. For PV users, the dataset includes imported energy, exported energy, and energy obtained from the PV power plant. Hence, for PV users (DU_1, DU_2, DU_4, DU_7, and DU_9), both M1 and M2 meters were used for the data collection. For non-PV users (DU_3, DU_5, DU_6, DU_8, and DU_10), the data only includes imported energy details, which are collected using an M1 meter. The data for both PV and non-PV users are sampled every 15 minutes. Similarly, the ENEA also provides a static dataset, which includes information about each user's annual consumption and contracted power or maximum power, as shown in Tab. 6.2. The static data is combined with the energy data to provide the model with an additional feature. However, deep learning models require large amounts of diverse datasets for better forecasting accuracy and performance. They are heavily dependent on datasets to understand hidden relationships, capture important features, and then provide precise forecasts using these learned representations. Additionally, the overall quality of the dataset also impacts the performance and accuracy of these models. The dataset obtained from ENEA mostly contained missing or null values, which is not enough to train or test the model. Additionally, there are a few parameters

ID User	Total residents	Residents at daytime	Surface area	Year of construction	Annual energy consumption	Maximum power
Units	-	-	(m^2)	(year)	(kWh)	(kW)
DU_1	2	1	140	1982-1991	4,279	6
DU_2	2	2	165	After 2015	1,210	3
DU_3	3	2	108	1972-1981	2,293	3
DU_4	2	0	297	1972-1981	10,455	6
DU_5	2	1	50	1991-2005	1,106	3
DU_6	1	1	109	1972-1981	1,098	3
DU_7	2	2	116	1982-1991	2,114	4.5
DU_8	3	1	300	1982-1991	3,194	6
DU_9	1	0	156	1982-1991	1,252	3
DU_10	3	1	274	1991-2005	2,557	3

Table 6.2: Description of the users and their annual electricity consumption

available, with no strong correlation among them. Hence, to improve the overall quality and quantity of the input dataset for training, we obtained the historical weather variables from the Open-Meteo API [154]. The historical weather variables, including temperature, cloud cover, shortwave radiation, direct normal irradiance, and diffuse radiation, are obtained for the same timespan (January 1st, 2024, to January 1st, 2025). However, weather variables obtained from the Open-Meteo API give a 1-hour resolution while the main dataset is sampled every 15 minutes. To get the 15-minute interval data for all weather variables, we used an interpolation function. The interpolation function can increase the temporal resolution without requiring the collection of additional observations. Additionally, it helps to determine missing or incomplete values. Although the converted temporal resolution may not provide accurate results, historical weather variables such as solar radiation, temperature, and others are important in understanding the fluctuations in PV power output. This is because weather variables have a strong positive or negative correlation with PV power, as we already noticed in the Alice Springs dataset. These correlations are important for the deep learning model to learn during the training process and provide accurate predictions on future datasets. Hence, after improving the temporal resolution of the weather variables, it is further integrated with the main energy dataset. Fig. 6.2 indicates the correlation between energy and weather variables. The DU_4 data exhibit multiple abnormalities and cannot demonstrate an appropriate correlation with weather variables, as confirmed in Fig. 6.2. Hence, we excluded DU_4 from the analysis to avoid affecting other parameters. Another problem is the time shift between the energy and weather variables. We obtained the weather variables from the Open-Meteo API for the same timespan as we get the energy dataset. However, after visualization, we found that the weather variables and energy display a one-hour time lag. This problem was rectified by giving a one-hour shift to the weather variables.

6.3. Data Analysis and Preprocessing

Once the ENEA data is collected and visualized, it is further pre-processed and prepared for model training. The ENEA dataset contains multiple missing values

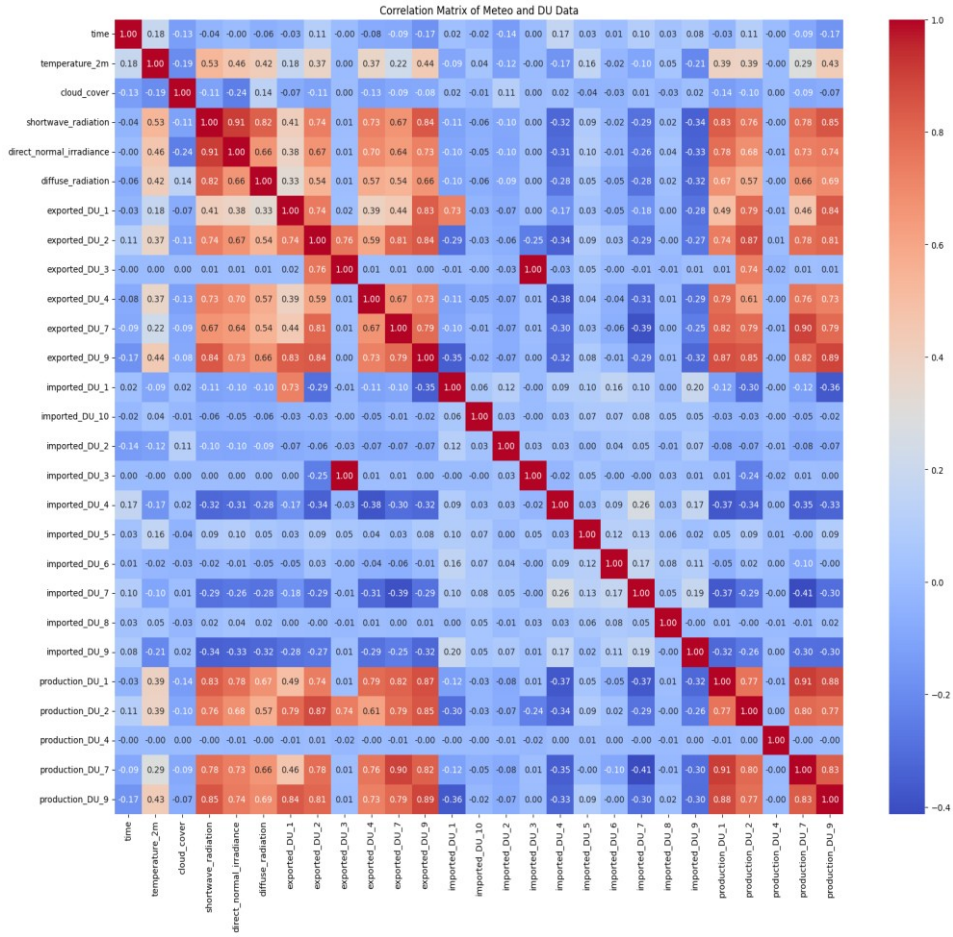


Figure 6.2: Heatmap of the Pearson's correlation matrix of weather variables and energy

and anomalies; removing these values is important for improving data quality and model reliability. These values occur in actual datasets due to various reasons, including sensor faults, human error, and equipment malfunctions. They can significantly impact the performance of deep learning models, resulting in distorted feature values, incorrect loss estimation, and erroneous gradient updates. In the ENEA dataset, numerous missing values appear within the existing rows. Additionally, in some instances, the entire time intervals were missing, and standard interpolation may not capture these gaps. To resolve this problem, we used a median imputation method, which is not influenced by spikes and is also less prone to anomalies. In this method, the missing values are replaced

by the median of all available values in the same column. Consider $X = X_1, X_2, X_3, \dots, X_n$ is any numerical feature with n number of observations. The median imputation can be expressed using the following equations,

$$X_{\text{imput},j} = \begin{cases} X_j, & \text{when } X_j \text{ is not missing} \\ \text{median}(X_1, X_2, X_3, \dots, X_N), & \text{when } X_j \text{ is missing} \end{cases} \quad (6.1)$$

X_j and N denote the value for the i$^{\text{th}}$ observation and the total number of samples.

Similarly, removing anomalies is important in enhancing the stability and generalization of DLMs. Anomalies can disturb the patterns and trends in the dataset, resulting in poor accuracy. Additionally, outliers can impact the decision of selecting a suitable model and the predicted values of the target variables. To detect and eliminate anomalies from the ENEA dataset, we used MAD, a powerful statistical method. MAD is designed to identify anomalies in time series datasets, such as PV power data, and is not affected by anomalies themselves. In this method, initially, a local baseline is established using a rolling median. For every timestep t, a rolling window size $w_s = 5$ is used as,

$$E_t = \text{median}\ (X_{t-2}, X_{t-1}, X_t, X_{t+1}, X_{t+2}\ ) \quad (6.2)$$

Where E_t is the anticipated PV output. Following that, each sample's deviation (S_t) from the local baseline is computed as,

$$S_t = \mid X_t - E_t \mid \quad (6.3)$$

Next, MAD at the time t for a wider window is computed as,

$$MAD_t = \text{median}(S_{t-7}, S_{t-6}, \dots, S_t, \dots, S_{t+6}, S_{t+7}) \quad (6.4)$$

The larger window size is crucial since it enhances the stability and accuracy of the variability estimate. Then the MAD is transformed into a robust standard deviation-like metric ($\sigma_{robustMAD,t}$) using a scaling constant. For normal distribution data, the scaling constant is 1.4826.

$$\sigma_{robustMAD,t} = 1.4826 \times \text{MAD}_t \quad (6.5)$$

After estimating the $\sigma_{robustMAD,t}$, a threshold ($T_{threshold,t}$) is specified for the detection of anomalies as,

$$T_{threshold,t} = E_t + 10 \times \sigma_{robustMAD,t} + 2.0 \tag{6.6}$$

In the above equation, a factor of 10 ensures that only undesirable spikes are detected. Equation (6.6) indicates that when $X_t > T_{threshold,t}$ it is considered a True value (anomaly); otherwise, it will be treated as a false (normal) value. After detecting the anomalies, it is then replaced with a rolling median (E_t) as,

$$X_{clean,t} = \begin{cases} X_t, & X_t \leq T_{threshold,t} \\ E_t, & X_t > T_{threshold,t} \end{cases} \tag{6.7}$$

Since medians are utilized at all stages, the MAD method is resistant to variability induced by large spikes. This makes it extremely useful for a real-world PV sensor dataset containing small yet extreme abnormalities. Fig. 6.3 and 6.4 show the

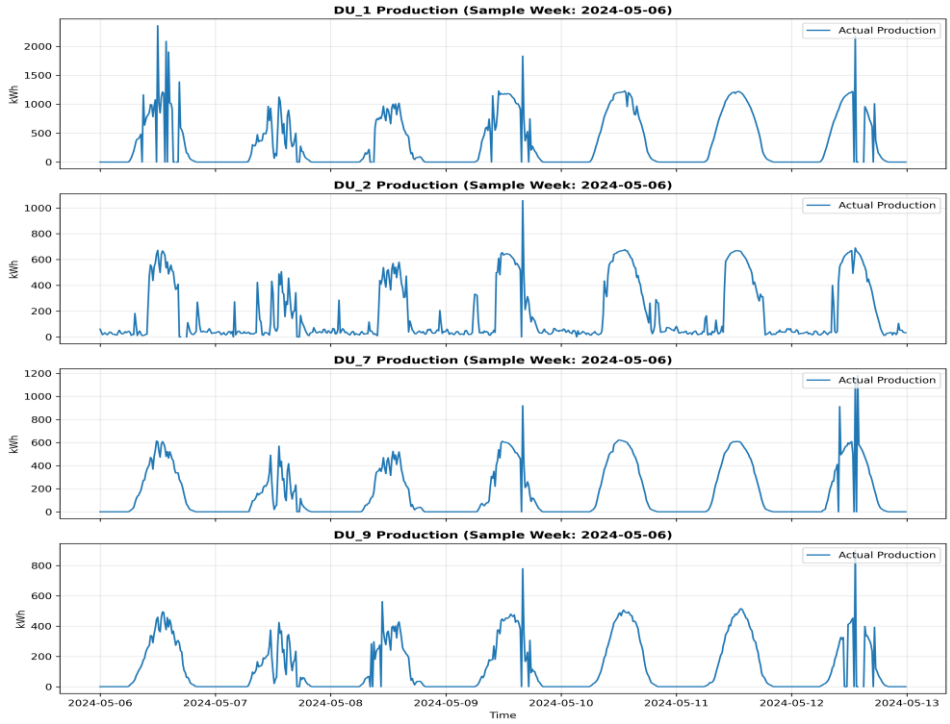


Figure 6.3: Anomaly detection in the ENEA dataset, for demonstration, only one week (06-05-2024 to 12-05-2024) is presented

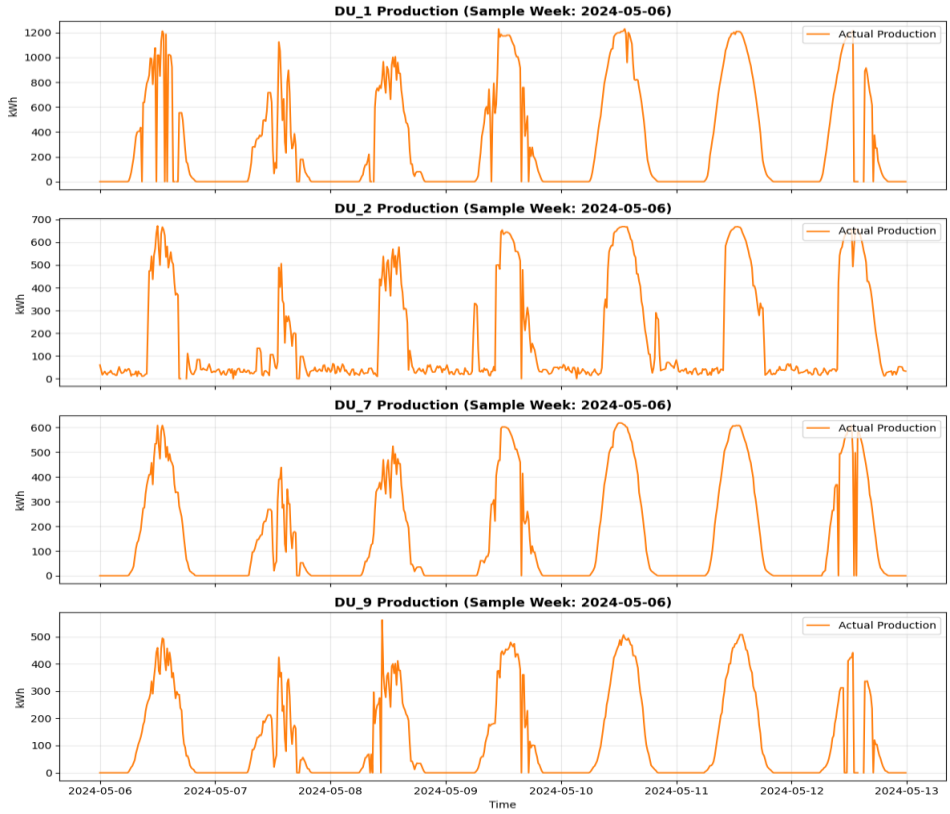


Figure 6.4: Removing anomaly from the ENEA dataset, for demonstration, only one week (06-05-2024 to 12-05-2024) is presented

graphs for PV users before and after applying the anomaly detection method. As displayed in Fig. 6.3 and 6.4, the anomalies distorted the results and provided a misleading visualization. They raise or lower the average and exaggerate the standard deviations, affecting data distribution.

6.4. Baseline Models for Analysis and Comparison

For comparison and analysis, we employed six deep learning models, including both single and hybrid models, as indicated in Tab. 6.3. The reason for selecting these models is that the ENEA dataset is slightly noisy and has a large number of missing values. Statistical models may not be suitable for these data due to their sensitivity to noise and anomalies. Additionally, these models have limited

Table 6.3: Detailed description of the baseline models for comparison on ENEA dataset

Model	Hidden layers	Neurons' numbers	Activation	Flatten/Pooling	Number of heads	Final Layers	Descriptions
CNN	4	4, 8, 16, 8		Flatten	-	Dense(8)+Dense(1)	• Kernel size: 2, pool size: 2, strides: 1
LSTM	4	4, 8, 16, 8		-	-	Dense(8)+Dense(1)	-
BiLSTM	4	4, 8, 16, 8	ReLU	-	-	Dense(8)+Dense(1)	-
BiLSTM-CNN-MHA	4	4, 8, 16, 8	ReLU	Flatten+ GlobalAveragePooling1D	8	Dense(8)+Dense(1)	• Dual stream • Two layers of BiLSTM, two layers of CNN • Multi-head attention layer + Global Average Pooling
HHO-BiLSTM-CNN-MHA	4	HHO-Optimized Units and Filters	ReLU	Flatten+ GlobalAveragePooling1D	HHO-Optimized heads	Dense(8)+Dense(1)	• Dual stream • Two layers of BiLSTM, two layers of CNN • Multi-head attention layer + Global Average Pooling

adaptability and capacity to complex patterns. All these baseline models are developed in Python 3.10.0, using the TensorFlow library version 2.10.0. As indicated in the table, all the baseline models use four hidden layers, containing 4, 8, 16, and 8 neurons, respectively. For fair comparison, all DLMs, including single and hybrid, used the same activation function (ReLU), same learning rate (0.001), same optimizer (Adam), same epochs (20), and same batch size (64); though, HHO and ADHHO optimized models employed the optimized hyperparameters. Additionally, all these models include the same number of fully connected layers and neurons (8 and 1). For BiLSTM-CNN-MHA, we selected 8 heads because, after some attempts, 8 heads produce satisfactory results.

6.5. Performance Metrics for the ENEA dataset

Similar to the Alice Springs dataset, to evaluate the performance and give a fair comparison between various forecasting models on the ENEA dataset, we used multiple performance metrics. These metrics include MAE, RMSE, R^2, and WMAPE. MAE determines the average magnitude of the error between the estimated and true values, treating all errors equally, which makes it less prone to outliers. RMSE is a commonly used metric for evaluating forecasting models, particularly for the PV power forecasting task. RMSE can be expressed as the square root of the average of the squared deviations between anticipated and actual values. This metric penalizes large errors more severely compared to MAE. Similarly, R^2 assesses the overall fitness of forecasting models for a particular dataset. It shows the ability of the forecasting model to explain the variation in a dataset. The R^2 normally ranges between 0 and 1, whereas 0 indicates that the model cannot capture any variations in the available dataset, and 1 indicates that the forecasting model precisely captures all variations in the available dataset. WMAPE is another metric employed to assess the performance of DLMs. It is the upgraded version of MAPE that incorporates the idea of weights. It computes the sum of absolute errors as a percentage of the sum of the true values. Mathematically, WMAPE can be expressed as,

$$WMAPE = \sum_{i=1}^{N} |y_{ai} - y_{fi}| \Big/ \sum_{i=1}^{N} |y_{ai}| \times 100 \quad (6.8)$$

Where, y_{ai} and y_{fi} represent the true and predicted values of PV power output. N denotes the sample size.

6.6. Description of the problem strategy

The task for the ENEA dataset was to initially train the forecasting models to predict the production of PV users and then use the same model to simulate the production for the non-PV users. The forecasting method for PV users employs the normal prediction strategy, such that the model is first trained and then assessed on the validation dataset. The PV users are those residences that have installed PV capacity with recorded energy production data. The method starts with data preparation, in which energy data and historical weather data are combined. In addition, a static dataset that includes unique features of each PV system is also incorporated. Then, a sliding window approach with a 96-step window is used to generate sequences for the forecasting models. Each trained model processes these sequences and produces prediction outcomes, which are then evaluated using performance metrics.

The forecasting strategy for simulated users followed a different approach: forecasting probable energy generation for residences without PV systems. These users represent non-PV users whose energy production is estimated using "what-if" conditions. As these users have no PV installations and thus do not possess any energy production information, an adaptive learning method is used. The static PV characteristics are obtained from the DU_7, thus mimicking that all simulated users have installed a similar PV system as DU_7. In other words, DU_7 static PV system configurations were theoretically adapted to other non-PV users' scenarios. User DU_7 was chosen as a benchmark as this user already has a PV system and can be used as a baseline. This method enables a reliable and unbiased PV power production across several users. The weather variables for simulated users are obtained from the same Open-Meteo API, maintaining temporal uniformity across all forecasts. The combined dataset (weather variables and borrowed static properties) is handled using the same normalizing and sequences

creation pipeline. These sequences reflect the general situation that a potential PV system might encounter at each non-PV user home location. The forecasting models will then produce prediction results using these sequences. The generated forecasts indicate the amount of energy each simulated user would produce if they installed an identical PV system(DU_7). However, it is worth noting that the simulated prediction cannot be evaluated using the performance metrics since we do not have any actual energy production for it. Instead, these forecasting results may help in the decision-making process, allowing the stakeholders to examine the possible benefits of PV installations for residences without PV energy systems.

6.7. Results and Discussions

This section provides the performance of the proposed model on the ENEA dataset. The results of the proposed model are then evaluated against the baseline models as described in Section 6.4. The baseline models include single, hybrid, and optimized deep learning models. All these models are developed in Python, using the TensorFlow library. As described in Section 6.4, all these models employ the same layers, number of neurons, activation functions, optimizer (Adam), and learning rate (0.001). To train these models, we used 80 percent of the ENEA dataset, and the remaining dataset was used to validate their performance. Additionally, a 24-hour lookback window (96 × 15 minutes = 1440 minutes) is used to allow the forecasting models to capture both daily trends and the daily solar cycle. The 24-hour lookback window size covers both day and night hours, allowing the forecasting models to capture the on/off cycle of the PV power output. Tab. 6.4 provides a detailed description of the performance of these models on the ENEA dataset for PV users. As shown in the table, the proposed model attained the best overall scores on all performance metrics. The ADDHO-BiLSTM-CNN-MHA achieved an overall MAE of 0.02411, an MSE of 0.00302, an RMSE of 0.05495, an R^2 of 0.8040, and a WMAPE of 33.95%. The proposed model is followed by BiLSTM-CNN-MHA, yielding an overall MAE, MSE, RMSE, R^2, and WMAPE values of 0.02473, 0.00304, 0.05514, 0.8026, and 34.83%, respectively. In single models, BiLSTM produces better overall results compared to both CNN and LSTM due to its capacity to simultaneously process the sequence in both

Models	Users	MAE	MSE	RMSE	R2	WMAPE
		(kWh)	(kWh)	(kWh)	-	(%)
CNN	Overall	**0.03406**	**0.00421**	**0.06487**	**0.7268**	**47.96%**
	DU_1	0.04067	0.00607	0.07792	0.8051	35.58%
	DU_2	0.03989	0.00479	0.06918	0.4866	59.96%
	DU_7	0.02635	0.00254	0.05041	0.6685	49.01%
	DU_9	0.03167	0.00351	0.05925	0.2161	79.06%
LSTM	Overall	**0.02969**	**0.00356**	**0.05970**	**0.7686**	**41.81%**
	DU_1	0.04275	0.00670	0.08183	0.7851	37.40%
	DU_2	0.03435	0.00414	0.06435	0.5558	51.63%
	DU_7	0.02073	0.00166	0.04074	0.7834	38.57%
	DU_9	0.02144	0.00159	0.03991	0.6443	53.53%
BiLSTM	Overall	**0.02635**	**0.00334**	**0.05776**	**0.7834**	**37.10%**
	DU_1	0.03147	0.00509	0.07135	0.8366	27.53%
	DU_2	0.03412	0.00478	0.06916	0.4870	51.28%
	DU_7	0.01809	0.00161	0.04016	0.7896	33.65%
	DU_9	0.02536	0.00239	0.04887	0.4667	63.32%
BiLSTM-CNN-MHA	Overall	**0.02473**	**0.00304**	**0.05514**	**0.8026**	**34.83%**
	DU_1	0.03059	0.00490	0.07003	0.8426	26.76%
	DU_2	0.02806	0.00300	0.05472	0.6788	42.17%
	DU_7	0.01645	0.00141	0.03755	0.8160	30.61%
	DU_9	0.02517	0.00262	0.05122	0.4143	62.85%
HHO-BiLSTM-CNN-MHA	Overall	**0.02506**	**0.00309**	**0.05554**	**0.7998**	**35.29%**
	DU_1	0.02956	0.00468	0.06840	0.8498	25.86%
	DU_2	0.03121	0.00403	0.06351	0.5674	46.91%
	DU_7	0.01503	0.00109	0.03303	0.8577	27.95%
	DU_9	0.02765	0.00289	0.05374	0.3551	69.02%
Proposed Model	Overall	**0.02411**	**0.00302**	**0.05495**	**0.8040**	**33.95%**
	DU_1	0.03062	0.00512	0.07153	0.8358	26.79%
	DU_2	0.03259	0.00376	0.06133	0.5966	48.99%
	DU_7	0.01548	0.00132	0.03633	0.8279	28.80%
	DU_9	0.02149	0.00199	0.04463	0.5553	53.64%

Table 6.4: Performance comparison of the forecasting models on the ENEA dataset

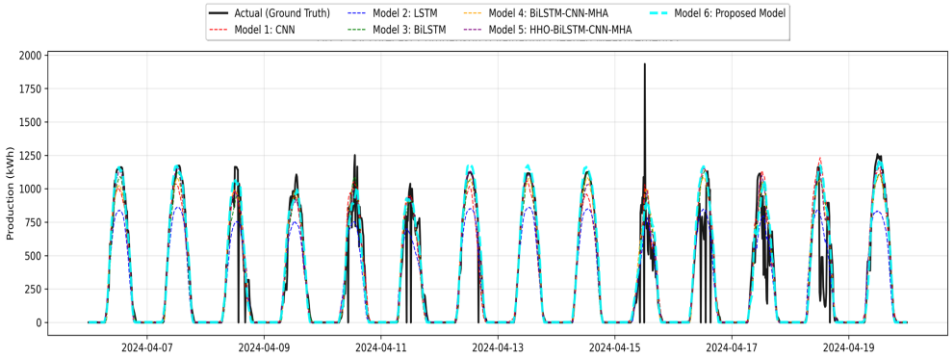
Model	Training time	Trainable Parameters
	(sec)	-
CNN	243.50	1,077
LSTM	4366.89	3,105
BiLSTM	8015.05	8,497
BiLSTM-CNN-MHA	2556.17	6,297
HHO-BiLSTM-CNN-MHA	4338.57	4,992
Proposed Model	4254.75	2,029

Table 6.5: Comparison of computational complexity and training time for all forecasting models'

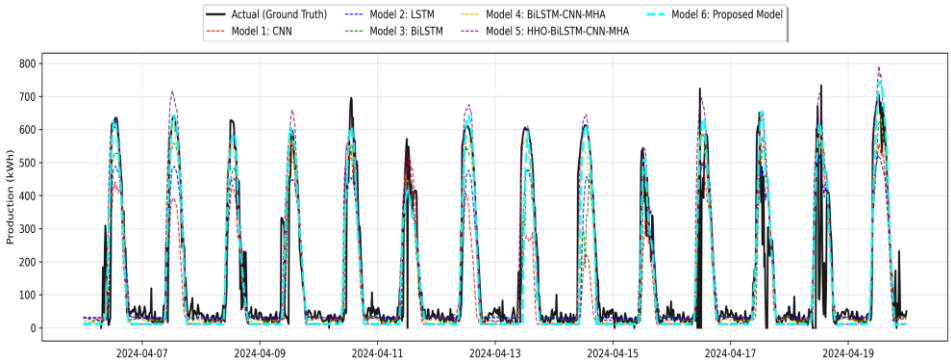
forward and backward directions. In contrast, CNN produces the worst results for all performance metrics with MAE, MSE, RMSE, R^2, and WMAPE values of 0.03406, 0.00421, 0.06487, 0.7268, and 47.96%, respectively. Tab. 6.5 presents the computational complexity and training time requirements for all studied models. The table shows that CNN is computationally less expensive due to its architectural layout. However, CNN is not suitable for capturing temporal dependencies, particularly in real-world and noisy datasets, as shown by experimental results from both Alice Springs and ENEA datasets. However, compared to all recurrent single and hybrid models, the proposed model is computationally less expensive, requires less training time, and achieves the best scores on all performance metrics. The ADHHO-BiLSTM-CNN-MHA has 20,29 trainable parameters and takes 4254.75 seconds to complete training. In contrast, BiLSTM is computationally more expensive among all studied models, with 8,497 trainable parameters and a training time of about 8015.05 seconds.

Fig. 6.5 illustrates the performance results for PV users using all six forecasting models. For clear visualization, we used sample data for two weeks (14days) from 06-04-2024 to 19-04-2024. The figure shows that the actual PV energy exhibits significant fluctuations in most of the sample data. This indicates that the weather in ENEA may range from generally overcast to rainy. Overall, all studied models, both single and hybrid, followed the general patterns of the actual energy production. Although the proposed model didn't capture some fluctuations, it still outperforms both single and hybrid models. As indicated by the figures, the proposed model mentioned in "cyan" closely followed the actual energy data for all DU_1, DU_2, DU_7, and DU_9. In contrast, CNN provides poor performance

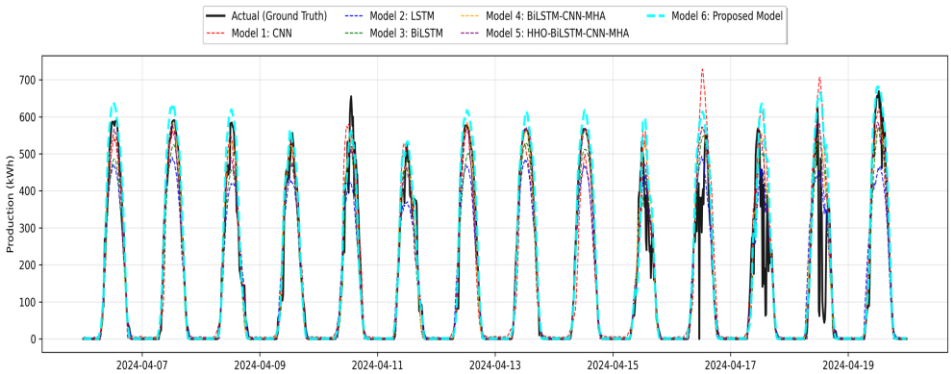
for most of the PV users, as indicated in Fig. 6.5. In DU_1, there is still an anomaly, which is not captured by any forecasting models. Similarly, the DU_2 actual energy data exhibits various fluctuations near zero values, which are not captured by any model, and this is also a positive aspect of the trained forecasting models.



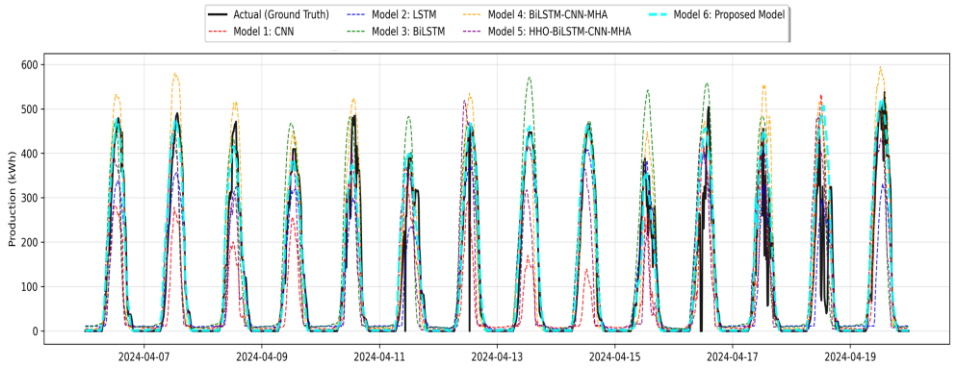
a) DU_1: PV forecast comparison



b) DU_2: PV forecast comparison



c) DU_7: PV forecast comparison



d) DU_9: PV forecast comparison

Figure 6.5: Comparison of the forecasting models' predictive capabilities for PV users (DU_1, DU_2, DU_7, and DU_9)

Fig. 6.6 displays the regression plot or scatter plot for all PV users. The figure shows that the actual and predicted values are highly correlated, forming a cluster near the regression line. This indicates that all studied models have precisely captured complex and nonlinear patterns in the energy production dataset. However, the figure shows that the proposed model, which is indicated in "cyan" provides a better fit, particularly in the areas where the energy production exhibits significant nonlinearity. Consequently, the results of the forecasting indicate that the ADHHO-BiLSTM-CNN-MHA achieved superior performance and precision compared to all studied models. As we indicated, the DU_1 still has some anomalies, which are also visible in the scatter plot. The scatter plot for DU_1, DU_7, and DU_9 form a close cluster near the regression line, while for DU_2, the actual and predicted values are more scattered away from the regression line. This can be confirmed from Tab. 6.4, where all forecasting models yielded high error rates compared to other PV users.

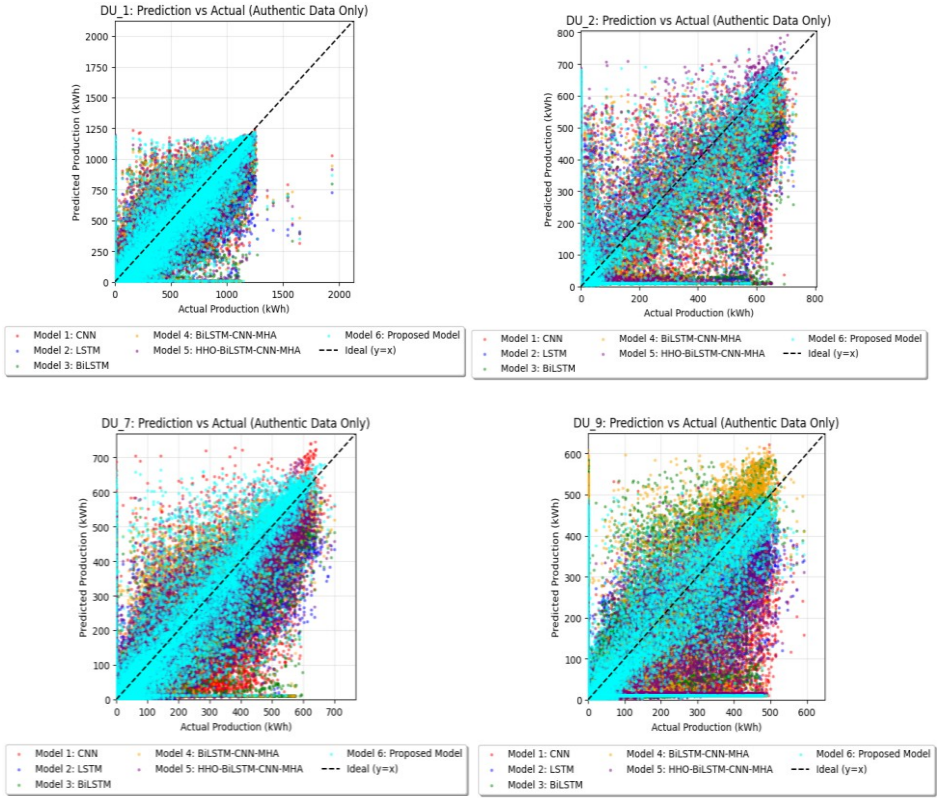
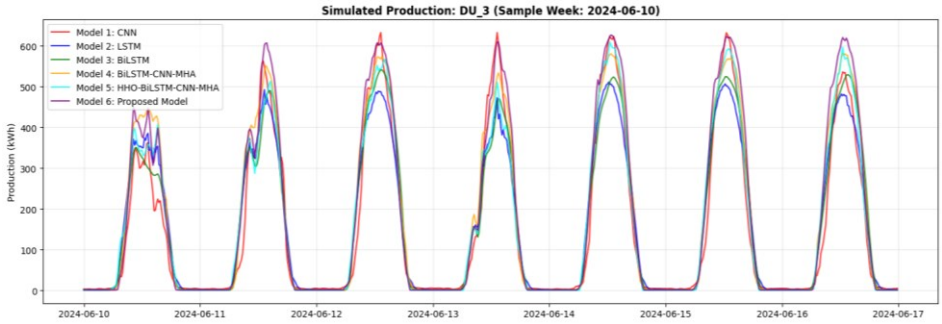


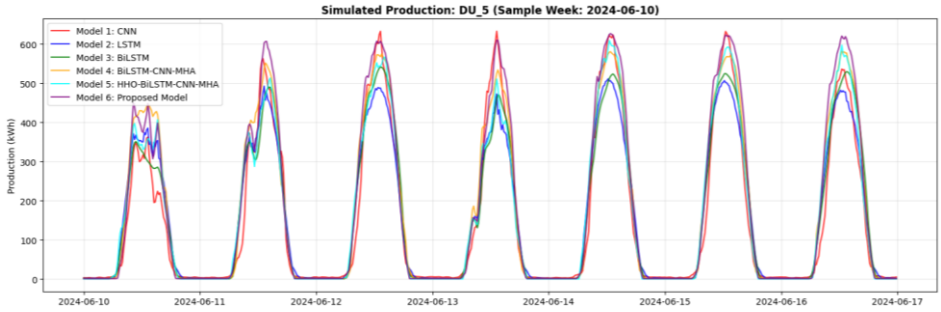
Figure 6.6: Scatter plot between actual and forecasted values of the studied models for the energy production of the PV users

line, while for DU_2, the actual and predicted values are more scattered away from the regression line. This can be confirmed from Tab. 6.4, where all forecasting models yielded high error rates compared to other PV users.

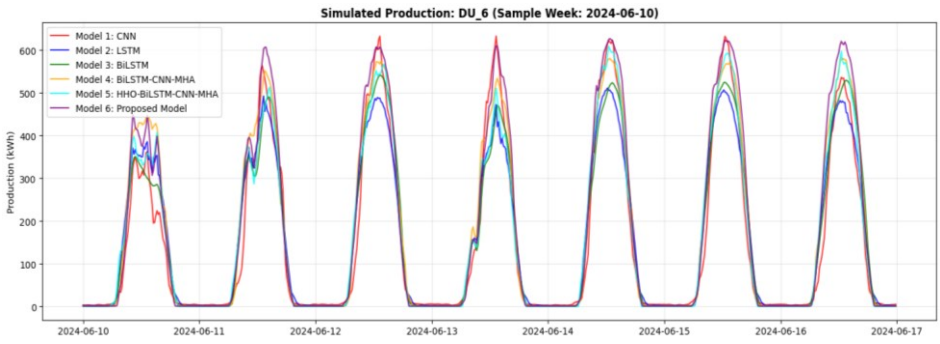
Fig. 6.7 shows the forecasting results of all studied models for simulated users or non-PV users. Since these users do not have installed PV capacity or energy production information, a transfer learning method is employed. We used DU_7 static PV system configurations for these users, combined with weather data. The figures for all users are mostly identical because we used the same input static PV system configurations (DU_7) for all non-PV users. Additionally, all these non-PV users are living in the same weather timeframe, and hence, this part of the input dataset is also the same for all users. Hence, feeding the forecasting models the same weather and PV system configurations will result in identical energy production.



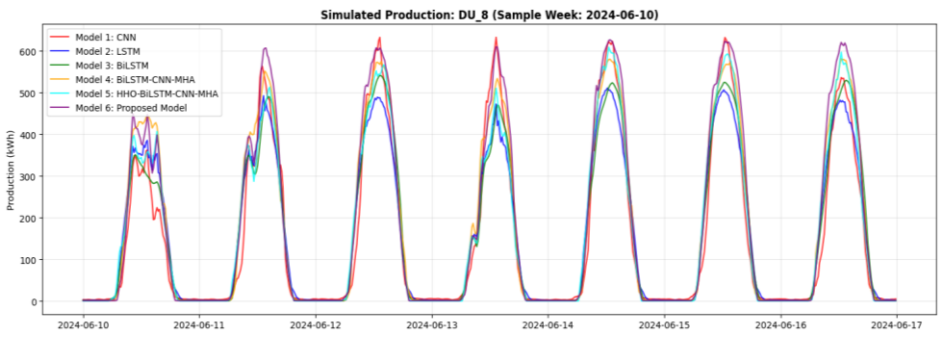
a) DU_3: simulated energy production



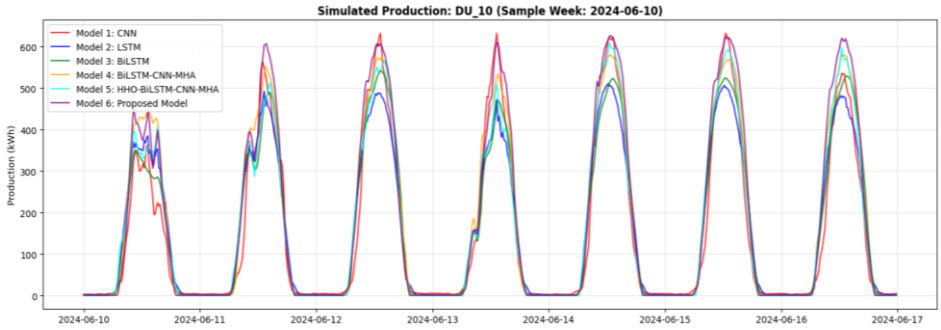
b) DU_5: simulated energy production



c) DU_6: simulated energy production



d) DU_8: simulated energy production



e) DU_8: simulated energy production

Figure 6.7: Comparison of the performance of all forecasting models for the simulated PV user

6.8. Summary

In this chapter, the performance of the proposed ADHHO-BiLSTM-CNN-MHA is validated using the ENEA dataset. The dataset, spanning from January 1, 2024, to January 1, 2025, was used for analysis. For PV users, the dataset includes imported energy, exported energy, and energy obtained from the PV power plant. For non-PV users, the data only includes imported energy details. The data for both PV and non-PV users are sampled every 15 minutes. Similarly, the ENEA also provides a static dataset, which includes information about each user's annual consumption and contracted power or maximum power. The static data is combined with the energy data to provide the model with an additional feature. Additionally, we also obtained historical weather data from the Open-Meteo API. The task for the ENEA dataset was to initially train the forecasting models to predict the production of PV users and then use the same model to simulate the production for the non-PV users. The forecasting method for PV users employs the normal prediction strategy, such that the model is first trained and then assessed on the validation dataset. All three datasets, including energy, static, and historical weather, are combined for forecasting the energy production for PV users. Then, the data is cleaned, normalized, and sequences are generated to train all studied models. To validate the performance of the proposed model, baseline models and performance metrics were carefully selected. The experimental results

show that the proposed ADDHO-BiLSTM-CNN-MHA outperforms all baseline models on all performance metrics. The proposed model achieved an overall MAE of 0.02411, an MSE of 0.00302, an RMSE of 0.05495, an R^2 of 0.8040, and a WMAPE of 33.95%. The proposed model is followed by BiLSTM-CNN-MHA, yielding an overall MAE, MSE, RMSE, R^2, and WMAPE values of 0.02473, 0.00304, 0.05514, 0.8026, and 34.83%, respectively. In single models, BiLSTM produces better overall results compared to both CNN and LSTM due to its capacity to simultaneously process the sequence in both forward and backward directions. In contrast, CNN produces the worst results for all performance metrics with MAE, MSE, RMSE, R^2, and WMAPE values of 0.03406, 0.00421, 0.06487, 0.7268, and 47.96%, respectively.

Similarly, the energy production for non-PV users is also predicted using a transfer learning method. In this case, the static PV characteristics of DU_7 are combined with weather variables obtained from the same Open-Meteo API. The combined dataset (weather variables and borrowed static properties) is handled using the same normalizing and sequences creation pipeline. However, it is worth noting that the simulated prediction cannot be evaluated using the performance metrics since we do not have any actual energy production for it. Instead, these forecasting results may help in the decision-making process, allowing the stakeholders to examine the possible benefits of PV installations for residences without PV energy systems. The experimental results show that the figures for all users are mostly identical because we used the same input static PV system configurations (DU_7) for all non-PV users. Additionally, all these non-PV users are living in the same weather timeframe, and hence, this part of the input dataset is also the same for all users. Hence, feeding the forecasting models the same weather and PV system configurations will result in identical energy production.

7

Conclusions and Future Directions

7.1. Conclusions

This thesis investigated the application of deep learning and optimization techniques for multistep real-time PV power forecasting. The proposed framework integrates a **dual-stream BiLSTM-CNN architecture, a multi-head attention mechanism, and an adaptive Harris Hawk optimization algorithm** to improve the performance and accuracy of PV power forecasting. The proposed model was primarily developed to address the fundamental issues of multistep PV power forecasting, including nonlinearity, data variability, seasonality, and real-time operational limitations. These issues are often encountered in real-world datasets. Additionally, the proposed model was evaluated for real-time forecasting capability, which required high precision and low computational expense. The experimental results demonstrated that the ADHHO-BiLSTM-CNN-MHA achieved superior performance on both forecasting tasks. The proposed model not only outperforms all baseline models on multi-step-ahead forecasting but is also suitable for real-time forecasting tasks.

In Chapter 2, we investigated earlier forecasting methods, including physical, statistical, machine learning, and deep learning models, as well as their limitations for solar power and irradiance forecasting. These models are not suitable for multistep ahead forecasting on real-world datasets due to their structural inability to handle sequential prediction complexity. Statistical models rely on recursive one-step predictions, resulting in continuous error propagation across a larger forecasting horizon. MLM models possess a shallow structure that considers input data as static and stationary feature vectors, ignoring sequential dependencies in

multistep forecasting. Even single DLMs lose crucial preliminary context information over long sequences and may not handle the stochastic nature of multistep-ahead forecasting. Hybrid BiLSTM-CNN models can provide better forecasting results; however, they condensed all the captured spatio-temporal information into a fixed-length latent representation. This reduces the model's capacity to maintain dynamic and transitory attributes, resulting in a significant drop in accuracy over longer time periods. Some studies used self-attention to overcome these limitations; however, its inability to detect simultaneous temporal patterns restricts the accuracy over longer horizons. Additionally, combining BiLSTM-CNN and self-attention sequentially is computationally expensive and may not be suitable for multistep forecasting, thus restricting real-time responses.

In Chapter 3, a dual-stream attention-based hybrid model is proposed to provide a robust multistep forecasting using real-world PV datasets. This approach provides more accurate generalization on diverse datasets, including real-world datasets, and can be applied to any forecasting task. The proposed deep learning model consists of two parts: a dual-stream BiLSTM-CNN and a multi-head attention. The proposed dual-stream framework offers several important advantages over the sequential stacking presented in the literature. Firstly, this structure can capture both spatial and temporal patterns in real-world PV power datasets simultaneously, enhancing accuracy and representational depth. Secondly, each model processes the input stream directly, thereby reducing the need for multiple transformations and minimizing information degradation. Thirdly, the proposed dual stream provides an efficient and reliable framework for combining multi-head attention layers. Lastly and importantly, the proposed dual stream reduces the model's trainable parameters and computational requirements. The low number of trainable parameters enables the model to consume less memory and thus be less prone to overfitting, which is critical for real-time PV power forecasting. The second part is the use of multi-head attention after the fused features of the BiLSTM-CNN. This framework is superior compared to others due to its ability to concentrate on diverse feature spaces that represent different temporal properties. Additionally, the proposed framework can dynamically and progressively fuse multi-scale data, enabling each head to concentrate on a crucial aspect of each subsequent time step. Such frameworks preclude error accumulation over longer forecasting horizons.

In Chapter 4, an adaptive Harris Hawk Optimization is designed for optimal tuning of the hyperparameters of the proposed dual-stream attention-based model. The proposed model, presented in Chapter 3, indicates a complex and non-convex optimization problem with a large and complicated set of hyperparameters. The demand for a precise, reliable optimization approach, such as ADHHO, can not only improve performance, but is also an essential requirement for this study due to the three main problems: lack of adaptability in conventional tuning techniques, 2) possibility of poor convergence and generalization, and 3) adjusting the framework for PV-specific dynamics. The proposed ADHHO uses adaptive search strategies that integrate chaotic optimization, a nonlinear control parameter, an adaptive inertia weight, and an adaptive decay factor to enhance convergence and robustness. To assess the performance and convergence characteristics of the proposed ADHHO, some experiments are conducted on well-known IEEE CEC benchmark suites, including 2014 and 2017. The performance results on these benchmark functions are then assessed against prominent MOAs, including PSO, GA, GWO, WOA, and HHO. The experimental results show that the proposed ADHHO achieved the theoretical best scores for most of the benchmark functions. These functions belong to all three groups: unimodal, multimodal, and fixed dimension. On the CEC 2014 benchmark functions, the proposed ADHHO obtained the best score on 40 out of 44 metric values, while on CEC 2017, it achieved the best score on 34 out of 40 metric values.

In Chapter 5, the dual stream attention-based model proposed in Chapter 3 and the adaptive Harris Hawk optimization algorithm developed in Chapter 4 are combined for multi-step ahead PV power forecasting. The data for experiments were obtained from the Trina Solar Plant (Trina, 10.5 kW, mono-Si, Dual, 2009) in Alice Springs, Australia. Three separate experiments were conducted: 1-step-ahead, 3-step-ahead, and 6-step-ahead forecasting. The results of each experiment are compared with well-known statistical (LR), machine learning (SVR), deep learning without optimization (MLP, CNN, LSTM, BiLSTM, CNN-BiLSTM, BiLSTM-CNN, BiLSTM-CNN-MHA), and deep learning with optimization (HHO-BiLSTM-CNN-MHA). Additionally, for comprehensive analysis, six performance metrics with different requirements and evaluation criteria, MSE, MAE, RMSE, R^2, MBE, and NRMSE, were used. The experimental

results show that on 1-step ahead forecasting, the proposed model demonstrates the highest predictive accuracy on four out of six performance metrics. The ADHHO-BiLSTM-CNN-MHA attained an MSE value of 0.00323, an RMSE value of 0.0568, an R^2 value of 0.9671, an MBE value of 0.0058, and an NRMSE value of 0.0597. On the MAE metric, HHO-BiLSTM-CNN-MHA achieved the lowest value of 0.023, while on the MBE metric, BiLSTM attained the best score of -0.0026. Similarly, on 3-step forecasting, the ADHHO-BiLSTM-CNN-MHA outperforms other studied models on five out of six evaluation metrics. The proposed model achieved an MAE of 0.0233, an MSE of 0.00338, an RMSE of 0.0581, an R^2 of 0.9656, and an NRMSE of 0.0610. On the MBE performance metric, the HHO-optimized BiLSTM-CNN-MHA outperforms all models, achieving the lowest value of 0.00004. Finally, on 6-step forecasting, the ADHHO-optimized BiLSTM-CNN-MHA achieved the lowest error rates on four out of six evaluation metrics. The proposed ADHHO-BiLSTM-CNN-MHA yielded the lowest MSE, RMSE, R^2, and NRMSE values of 0.00392, 0.0626, 0.9600, and 0.0657, respectively. Similarly, the HHO-optimized BiLSTM-CNN-MHA achieved the best score on MAE (0.0259), while CNN attained the lowest score of -0.0002 on the MBE metric. Overall, the results of the three experiments, including tables and figures, confirmed that the proposed ADHHO-BiLSTM-CNN-MHA outperforms all single and hybrid models on multi-step PV power forecasting on a real-world dataset.

In Chapter 6, the proposed ADHHO-optimized dual-stream attention-based model is evaluated for real-time PV power forecasting. The objective of this task was to initially train the forecasting models for PV users and then use the same models to predict the PV power output for non-PV users. However, the ENEA dataset was insufficient for model training, so we combined three datasets: energy and static datasets obtained from ENEA, and a historical weather dataset from the Open-Meteo API. After combining these datasets, we found various irregularities using data visualization. Hence, we performed rigorous data preprocessing to remove missing values, anomalies, time shifts, and correlation issues. After data preparation, we carefully selected the baseline models for comparisons. This is because some models in Chapter 5 showed overfitting behavior, while others were computationally expensive when applied to the ENEA dataset. Additionally, five performance metrics with different evaluation criteria were selected, including

MAE, MSE, RMSE, R^2, and NMAPE. The experimental findings show that the proposed model achieved the highest predictive accuracy on all performance metrics. The ADDHO-BiLSTM-CNN-MHA achieved an overall MAE of 0.02411, an MSE of 0.00302, an RMSE of 0.05495, an R^2 of 0.8040, and a WMAPE of 33.95%. The proposed model is followed by BiLSTM-CNN-MHA, yielding an overall MAE, MSE, RMSE, R^2, and WMAPE values of 0.02473, 0.00304, 0.05514, 0.8026, and 34.83%, respectively.

7.2. Future Directions

This thesis presents a comprehensive analysis of an ADHHO-optimized dual-stream attention-based hybrid deep learning model for multistep PV power forecasting. The proposed model shows superior performance on diverse datasets with low computational burden. Hence, the proposed model can be adopted as a reliable tool for any forecasting tasks, where accuracy, reliability, and stability are important. The findings of this thesis open multiple research avenues for further improvements.

Synthesis of a high-dimensional spatial-temporal dataset

Existing prediction models depend heavily on sequential inputs such as historical PV power and weather variables. These inputs may not accurately reflect the spatial nature of the weather variables, which have a significant impact on PV power output. Thus, combining the high-dimensional spatial dataset, including satellite-derived cloud images, sky camera measurements, or a dataset from multiple sensor networks, can be a primary future direction. Employing improved spatial encoders, the forecasting models can capture cloud movements and spatial radiation patterns directly. Such improvements will allow the models to precisely capture physical mechanisms triggering PV power fluctuations.

Probabilistic PV power forecasting

This thesis focused on the PV power forecasting for both Alice Springs and ENEA using a point-based or deterministic forecasting approach. The primary future direction is to extend the existing deterministic approach into a probabilistic

forecasting framework. The probabilistic forecasting approach provides a range of probabilities corresponding to all possible future results. It presents a detailed analysis of the uncertainties associated with future outcomes, thus improving the decision-making process. In practical energy systems, understanding uncertainties is crucial for grid operators to control backup generators and minimize operational and monetary hazards associated with forecasting inaccuracies. Hence, extending the current approach into probabilistic forecasting will establish the basis for uncertainty-informed PV power prediction.

Applicability to real-time forecasting

The proposed model is suitable for real-time forecasting due to its lower trainable parameters. Future research can further reduce the computational complexity while retaining high forecasting accuracy. Multiple compression methods, like network pruning, may be employed to eliminate ineffective neurons and reduce the number of attention heads. Particularly, the application of adaptive HHO can be extended to optimize forecasting accuracy and computational expense measures, including model size and training time, simultaneously.

Application to other renewable and load forecasting tasks

This research work employed the proposed model to predict multistep ahead PV power forecasting; however, it can be easily adapted to other renewable energy sources and load forecasting tasks. Particularly, the same forecasting model can be used for wind energy forecasting, where weather fluctuations can highly impact the output power production. Similarly, its applications can be extended to a load forecasting task, which is affected by spatiotemporal trends, weather fluctuations, and human behavior. The proposed dual-stream attention-based model and adaptive HHO can be easily adapted to handle diverse data characteristics.

Acknowledgement

The last three years have been a challenging but gratifying experience. Firstly, I would like to express my deepest appreciation to my supervisor, Professor Francesco Riganti-Fulginei, for his ongoing support and guidance. His expertise and insight have assisted me through both academic and professional experiences, transforming me into a more committed and devoted researcher. His enthusiasm and determination have been a continual source of encouragement, and I am deeply thankful for his assistance during this journey.

Additionally, I am deeply grateful to my coordinator, Professor Giovanni Spagnuolo, for his kindness, advice, and support. His courteous yet perceptive guidance has motivated me to improve my research and professional career. I would also like to express my sincere gratitude to the board of professors and faculty members for their valuable academic advice, insightful suggestions, and unwavering encouragement throughout my PhD studies.

Last but not least, I want to express my heartfelt appreciation to my parents, uncle, brothers, and all other family members for their unwavering love, devotion, and encouragement. Their support enabled me to overcome challenges and develop my own way in life.

Bibliography

[1] R. Asghar, M. H. Sulaiman, Z. Mustaffa, N. Ullah, and W. Hassan, "The important contribution of renewable energy technologies in overcoming Pakistan's energy crisis: Present challenges and potential opportunities," *Energy Environ.*, vol. 34, no. 8, pp. 3450–3494, Dec. 2023, doi: 10.1177/0958305X221134110.

[2] F. Zheng *et al.*, "A Highly Sensitive CRISPR-Empowered Surface Plasmon Resonance Sensor for Diagnosis of Inherited Diseases with Femtomolar-Level Real-Time Quantification," *Adv. Sci.*, vol. 9, no. 14, p. 2105231, May 2022, doi: 10.1002/advs.202105231.

[3] R. Asghar, F. R. Fulginei, M. Quercio, and A. Mahrouch, "Artificial Neural Networks for Photovoltaic Power Forecasting: A Review of Five Promising Models," *IEEE Access*, vol. 12, pp. 90461–90485, 2024, doi: 10.1109/ACCESS.2024.3420693.

[4] H. Wang *et al.*, "Taxonomy research of artificial intelligence for deterministic solar power forecasting," *Energy Convers. Manag.*, vol. 214, p. 112909, June 2020, doi: 10.1016/j.enconman.2020.112909.

[5] B. Uzum, A. Onen, H. M. Hasanien, and S. M. Muyeen, "Rooftop Solar PV Penetration Impacts on Distribution Network and Further Growth Factors—A Comprehensive Review," *Electronics*, vol. 10, no. 1, p. 55, Dec. 2020, doi: 10.3390/electronics10010055.

[6] H. Wen, "DEEP LEARNING-BASED SOLAR FORECASTING METHODS FOR BETTER PHOTOVOLTAIC INTEGRATION," University of Liverpool, UK, 2023. [Online]. Available:

https://livrepository.liverpool.ac.uk/3170859/1/PhD_Thesis_HaoranWen_201026343.pdf

[7] A. Jakoplić, D. Franković, V. Kirinčić, and T. Plavšić, "Benefits of short-term photovoltaic power production forecasting to the power system," *Optim. Eng.*, vol. 22, no. 1, pp. 9–27, Mar. 2021, doi: 10.1007/s11081-020-09583-y.

[8] E. A. F. Nunes, D. K. Alves, E. S. Dantas, T. O. A. Rocha, R. L. A. Ribeiro, and F. B. Costa, "Impact of PV systems on microgrids under different levels of penetration and operational scenarios," in *2017 Brazilian Power Electronics Conference (COBEP)*, Juiz de Fora: IEEE, Nov. 2017, pp. 1–6. doi: 10.1109/COBEP.2017.8257336.

[9] L. Shpaner, "When Inference Meets Prediction: Navigating the Line Between Statistical Models and Machine Learning," University of San Diego online. [Online]. Available: https://onlinedegrees.sandiego.edu/when-inference-meets-prediction-navigating-the-line-between-statistical-models-and-machine-learning/?utm_source=chatgpt.com

[10] E. Chodakowska, J. Nazarko, Ł. Nazarko, H. S. Rabayah, R. M. Abendeh, and R. Alawneh, "ARIMA Models in Solar Radiation Forecasting in Different Geographic Locations," *Energies*, vol. 16, no. 13, p. 5029, June 2023, doi: 10.3390/en16135029.

[11] F. Haselbeck, J. Killinger, K. Menrad, T. Hannus, and D. G. Grimm, "Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions," *Mach. Learn. Appl.*, vol. 7, p. 100239, Mar. 2022, doi: 10.1016/j.mlwa.2021.100239.

[12] Y. Ledmaoui, A. El Maghraoui, M. El Aroussi, R. Saadane, A. Chebak, and A. Chehri, "Forecasting solar energy production: A comparative study of machine learning algorithms," *Energy Rep.*, vol. 10, pp. 1004–1012, Nov. 2023, doi: 10.1016/j.egyr.2023.07.042.

[13] D. Lee, J. Jeong, S. H. Yoon, and Y. T. Chae, “Improvement of Short-Term BIPV Power Predictions Using Feature Engineering and a Recurrent Neural Network,” *Energies*, vol. 12, no. 17, p. 3247, Aug. 2019, doi: 10.3390/en12173247.

[14] R. Asghar, F. R. Fulginei, M. Quercio, L. Sabino, F. Crescimbini, and M. Abusara, “A Hybrid LSTM-CNN Model for Short-Term Photovoltaic Power Forecasting in Italy,” in *2024 IEEE International Symposium on Consumer Technology (ISCT)*, Kuta, Bali, Indonesia: IEEE, Aug. 2024, pp. 67–73. doi: 10.1109/ISCT62336.2024.10791191.

[15] M. Sabri and M. El Hassouni, “Predicting photovoltaic power generation using double-layer bidirectional long short-term memory-convolutional network,” *Int. J. Energy Environ. Eng.*, vol. 14, no. 3, pp. 497–510, Sept. 2023, doi: 10.1007/s40095-022-00530-4.

[16] A. Rai, A. Shrivastava, and K. C. Jana, “Differential attention net: Multi-directed differential attention based hybrid deep learning model for solar power forecasting,” *Energy*, vol. 263, p. 125746, Jan. 2023, doi: 10.1016/j.energy.2022.125746.

[17] R. Asghar, M. Quercio, L. Sabino, A. Mahrouch, and F. R. Fulginei, “A novel dual-stream attention-based hybrid network for solar power forecasting,” *IEEE Access*, pp. 1–1, 2025, doi: 10.1109/access.2025.3555810.

[18] R. Asghar, F. R. Fulginei, M. Quercio, M. Maoz, L. Sabino, and M. Abusara, “Day-Ahead Photovoltaic Power Forecasting Using a Hybrid BiLSTM-CNN Model,” in *2024 Sixth International Conference on Intelligent Computing in Data Sciences (ICDS)*, Marrakech, Morocco: IEEE, Oct. 2024, pp. 1–8. doi: 10.1109/ICDS62089.2024.10756380.

[19] H. Abbasimehr and R. Paki, “Improving time series forecasting using LSTM and attention models,” *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 1, pp. 673–691, Jan. 2022, doi: 10.1007/s12652-020-02761-x.

[20] M. Neshat, M. M. Nezhad, S. Mirjalili, D. A. Garcia, E. Dahlquist, and A. H. Gandomi, “Short-term solar radiation forecasting using hybrid deep residual learning and

gated LSTM recurrent network with differential covariance matrix adaptation evolution strategy," *Energy*, vol. 278, p. 127701, Sept. 2023, doi: 10.1016/j.energy.2023.127701.

[21] R. Asghar, M. Quercio, L. Sabino, D. Milillo, A. Mahrouch, and F. R. Fulginei, "Solar irradiance estimation using a hybrid deep learning model with metaheuristic algorithms," in *2025 9th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, Gaziantep, Turkiye: IEEE, June 2025, pp. 1–8. doi: 10.1109/ISAS66241.2025.11101966.

[22] M. Shehab *et al.*, "Harris Hawks Optimization Algorithm: Variants and Applications," *Arch. Comput. Methods Eng.*, vol. 29, no. 7, pp. 5579–5603, Nov. 2022, doi: 10.1007/s11831-022-09780-1.

[23] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges," *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 13187–13257, Nov. 2023, doi: 10.1007/s10462-023-10470-y.

[24] O. Zedadra, A. Guerrieri, N. Jouandeau, G. Spezzano, H. Seridi, and G. Fortino, "Swarm intelligence-based algorithms within IoT-based systems: A review," *J. Parallel Distrib. Comput.*, vol. 122, pp. 173–187, Dec. 2018, doi: 10.1016/j.jpdc.2018.08.007.

[25] H. Zhu, X. Li, Q. Sun, L. Nie, J. Yao, and G. Zhao, "A Power Prediction Method for Photovoltaic Power Plant Based on Wavelet Decomposition and Artificial Neural Networks," *Energies*, vol. 9, no. 1, p. 11, Dec. 2015, doi: 10.3390/en9010011.

[26] N. Son and M. Jung, "Analysis of Meteorological Factor Multivariate Models for Medium- and Long-Term Photovoltaic Solar Power Forecasting Using Long Short-Term Memory," *Appl. Sci.*, vol. 11, no. 1, p. 316, Dec. 2020, doi: 10.3390/app11010316.

[27] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a Python library for model selection and hyperparameter optimization," *Comput. Sci. Discov.*, vol. 8, no. 1, p. 014008, July 2015, doi: 10.1088/1749-4699/8/1/014008.

[28] S. Hanifi, A. Cammarono, and H. Zare-Behtash, “Advanced hyperparameter optimization of deep learning models for wind power prediction,” *Renew. Energy*, vol. 221, p. 119700, Feb. 2024, doi: 10.1016/j.renene.2023.119700.

[29] J. L. Holechek, H. M. E. Geli, M. N. Sawalhah, and R. Valdez, “A Global Assessment: Can Renewable Energy Replace Fossil Fuels by 2050?,” *Sustainability*, vol. 14, no. 8, p. 4792, Apr. 2022, doi: 10.3390/su14084792.

[30] IRENA, “Renewable power generation costs in 2023,” International Renewable Energy Agency, Abu Dhabi, 2024. [Online]. Available: https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2024/Sep/IRENA_Renewable_power_generation_costs_in_2023.pdf

[31] International Energy Agency, “Task 1 Strategic PV Analysis and Outreach – 2025 Snapshot of Global PV Markets,” International Energy Agency, Apr. 2025. [Online]. Available: https://iea-pvps.org/wp-content/uploads/2025/04/Snapshot-of-Global-PV-Markets_2025.pdf

[32] IEA, “Renewables 2024, Analysis and forecast to 2030,” Paris, Oct. 2024. [Online]. Available: https://www.iea.org/reports/renewables-2024

[33] Bloomberg News, “Solar May Generate Half of World’s Power by 2050,” Bloomberg. Accessed: Apr. 20, 2022. [Online]. Available: bloomberg.com/news/articles/2022-04-20/solar-energy-may-generate-half-of-world-s-power-by-2050#xj4y7vzkg

[34] J. Zhang *et al.*, “A suite of metrics for assessing the performance of solar power forecasting,” *Sol. Energy*, vol. 111, pp. 157–175, Jan. 2015, doi: 10.1016/j.solener.2014.10.016.

[35] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. J. Martinez-de-Pison, and F. Antonanzas-Torres, “Review of photovoltaic power forecasting,” *Sol. Energy*, vol. 136, pp. 78–111, Oct. 2016, doi: 10.1016/j.solener.2016.06.069.

[36] B. Kraas, M. Schroedter-Homscheidt, and R. Madlener, “Economic merits of a state-of-the-art concentrating solar power forecasting system for participation in the Spanish electricity market,” *Sol. Energy*, vol. 93, pp. 244–255, July 2013, doi: 10.1016/j.solener.2013.04.012.

[37] A. M. Assaf, H. Haron, H. N. Abdull Hamed, F. A. Ghaleb, S. N. Qasem, and A. M. Albarrak, “A Review on Neural Network Based Models for Short Term Solar Irradiance Forecasting,” *Appl. Sci.*, vol. 13, no. 14, p. 8332, July 2023, doi: 10.3390/app13148332.

[38] L. Wang and J. Shi, “A Comprehensive Application of Machine Learning Techniques for Short-Term Solar Radiation Prediction,” *Appl. Sci.*, vol. 11, no. 13, p. 5808, June 2021, doi: 10.3390/app11135808.

[39] C. Cui, Y. Zou, L. Wei, and Y. Wang, “Evaluating combination models of solar irradiance on inclined surfaces and forecasting photovoltaic power generation,” *IET Smart Grid*, vol. 2, no. 1, pp. 123–130, Mar. 2019, doi: 10.1049/iet-stg.2018.0110.

[40] P. Mathiesen, C. Collier, and J. Kleissl, “A high-resolution, cloud-assimilating numerical weather prediction model for solar irradiance forecasting,” *Sol. Energy*, vol. 92, pp. 47–61, June 2013, doi: 10.1016/j.solener.2013.02.018.

[41] R. Perez *et al.*, “Comparison of numerical weather prediction solar irradiance forecasts in the US, Canada and Europe,” *Sol. Energy*, vol. 94, pp. 305–326, Aug. 2013, doi: 10.1016/j.solener.2013.05.005.

[42] J. Thaker and R. Höller, “Evaluation of High Resolution WRF Solar,” *Energies*, vol. 16, no. 8, p. 3518, Apr. 2023, doi: 10.3390/en16083518.

[43] M. J. Mayer, D. Yang, and B. Szintai, “Comparing global and regional downscaled NWP models for irradiance and photovoltaic power forecasting: ECMWF versus AROME,” *Appl. Energy*, vol. 352, p. 121958, Dec. 2023, doi: 10.1016/j.apenergy.2023.121958.

[44] H. Ye, B. Yang, Y. Han, and N. Chen, “State-Of-The-Art Solar Energy Forecasting Approaches: Critical Potentials and Challenges,” *Front. Energy Res.*, vol. 10, p. 875790, Mar. 2022, doi: 10.3389/fenrg.2022.875790.

[45] J. D. Tarpley, “Estimating Incident Solar Radiation at the Surface from Geostationary Satellite Data,” *J. Appl. Meteorol.*, vol. 18, no. 9, pp. 1172–1181, Sept. 1979, doi: 10.1175/1520-0450(1979)018%3C1172:eisrat%3E2.0.co;2.

[46] Y. Xie, M. Sengupta, and J. Dudhia, “A Fast All-sky Radiation Model for Solar applications (FARMS): Algorithm and performance evaluation,” *Sol. Energy*, vol. 135, pp. 435–445, Oct. 2016, doi: 10.1016/j.solener.2016.06.003.

[47] B. Tournadre, B. Gschwind, Y.-M. Saint-Drenan, X. Chen, R. Amaro E Silva, and P. Blanc, “An alternative cloud index for estimating downwelling surface solar irradiance from various satellite imagers in the framework of a Heliosat-V method,” *Atmospheric Meas. Tech.*, vol. 15, no. 12, pp. 3683–3704, June 2022, doi: 10.5194/amt-15-3683-2022.

[48] R. E. Bird and R. L. Hulstrom, “A Simplified Clear Sky Model for Direct and Diffuse Insolation on Horizontal Surfaces,” Solar Energy Research Institute, Feb. 1981. [Online]. Available: https://www.nrel.gov/media/docs/libraries/grid/tr-642-761.pdf?sfvrsn=dbe556c7_1

[49] F. Antonanzas-Torres, R. Urraca, J. Polo, O. Perpiñán-Lamigueiro, and R. Escobar, “Clear sky solar irradiance models: A review of seventy models,” *Renew. Sustain. Energy Rev.*, vol. 107, pp. 374–387, June 2019, doi: 10.1016/j.rser.2019.02.032.

[50] N. Straub, S. Karalus, W. Herzberg, and E. Lorenz, “Satellite-Based Solar Irradiance Forecasting: Replacing Cloud Motion Vectors by Deep Learning,” *Sol. RRL*, vol. 8, no. 24, Dec. 2024, doi: 10.1002/solr.202400475.

[51] K. Pearson, “LIII. *On lines and planes of closest fit to systems of points in space*,” *Lond. Edinb. Dublin Philos. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, Nov. 1901, doi: 10.1080/14786440109462720.

[52] A. Maćkiewicz and W. Ratajczak, “Principal components analysis (PCA),” *Comput. Geosci.*, vol. 19, no. 3, pp. 303–342, Mar. 1993, doi: 10.1016/0098-3004(93)90090-r.

[53] J. Shlens, “A Tutorial on Principal Component Analysis,” 2014, *arXiv*. doi: 10.48550/ARXIV.1404.1100.

[54] M. A. Hammad, B. Jereb, B. Rosi, and D. Dragan, “Methods and Models for Electric Load Forecasting: A Comprehensive Review,” *Logist. Sustain. Transp.*, vol. 11, no. 1, pp. 51–76, Feb. 2020, doi: 10.2478/jlst-2020-0004.

[55] V. Prema and K. U. Rao, “Development of statistical time series models for solar power prediction,” *Renew. Energy*, vol. 83, pp. 100–109, Nov. 2015, doi: 10.1016/j.renene.2015.03.038.

[56] K. Ullah *et al.*, “Short-Term Load Forecasting: A Comprehensive Review and Simulation Study With CNN-LSTM Hybrids Approach,” *IEEE Access*, vol. 12, pp. 111858–111881, 2024, doi: 10.1109/access.2024.3440631.

[57] H. H. Hameed, “Smoothing Techniques for Time Series Forecasting,” Eastern Mediterranean University, North Cyprus, 2015. [Online]. Available: https://i-rep.emu.edu.tr/xmlui/bitstream/handle/11129/2326/hameedhaifaa.pdf?sequence=1

[58] N. T. Thanh Loan, D. Ngoc Hung, and V. T. Thuy Van, “Application of ARIMA model and deep learning in forecasting stock price in Vietnam,” *Salud Cienc. Tecnol. - Ser. Conf.*, vol. 4, p. 1320, Jan. 2025, doi: 10.56294/sctconf20251320.

[59] A. M. Turing, “COMPUTING MACHINERY AND INTELLIGENCE,” *Mind*, vol. 49, pp. 433–460, Oct. 1950, [Online]. Available: https://courses.cs.umbc.edu/471/papers/turing.pdf

[60] T. J. Sejnowski, “The unreasonable effectiveness of deep learning in artificial intelligence,” *Proc. Natl. Acad. Sci.*, vol. 117, no. 48, pp. 30033–30038, Dec. 2020, doi: 10.1073/pnas.1907373117.

[61] Z. Ersozlu, S. Taheri, and I. Koch, “A review of machine learning methods used for educational data,” *Educ. Inf. Technol.*, vol. 29, no. 16, pp. 22125–22145, Nov. 2024, doi: 10.1007/s10639-024-12704-0.

[62] R. K. Halder, M. N. Uddin, Md. A. Uddin, S. Aryal, and A. Khraisat, “Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications,” *J. Big Data*, vol. 11, no. 1, Aug. 2024, doi: 10.1186/s40537-024-00973-y.

[63] A. Navada, A. N. Ansari, S. Patil, and B. A. Sonkamble, “Overview of use of decision tree algorithms in machine learning,” in *2011 IEEE Control and System Graduate Research Colloquium*, Shah Alam, Malaysia: IEEE, June 2011, pp. 37–42. doi: 10.1109/icsgrc.2011.5991826.

[64] “Estimating aboveground stand carbon by combining Sentinel-1 and Sentinel-2 satellite data: a case study from Turkey,” in *Forest Resources Resilience and Conflicts*, Elsevier, 2021, pp. 117–126. doi: 10.1016/b978-0-12-822931-6.00008-3.

[65] Z. Ramedani, M. Omid, A. Keyhani, S. Shamshirband, and B. Khoshnevisan, “Potential of radial basis function based support vector regression for global solar radiation prediction,” *Renew. Sustain. Energy Rev.*, vol. 39, pp. 1005–1011, Nov. 2014, doi: 10.1016/j.rser.2014.07.108.

[66] R. Nisbet, M. Gary, and K. Yale, *Handbook of Statistical Analysis and Data Mining Applications*. Elsevier, 2018. doi: 10.1016/C2012-0-06451-4.

[67] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, Antalya: IEEE, Aug. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

[68] M. Coskun, A. Ucar, O. Yildirim, and Y. Demir, “Face recognition based on convolutional neural network,” in *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, Kremenchuk: IEEE, Nov. 2017, pp. 376–379. doi: 10.1109/MEES.2017.8248937.

[69] H. Gholamalinejad and H. Khosravi, “Vehicle classification using a real-time convolutional structure based on DWT pooling layer and SE blocks,” *Expert Syst. Appl.*, vol. 183, p. 115420, Nov. 2021, doi: 10.1016/j.eswa.2021.115420.

[70] I. Sutskever, J. Martens, and G. E Hinton, “Generating text with recurrent neural networks,” in *Proceedings of the 28th international conference on machine learning*, pp. 1017–1024. [Online]. Available: https://dl.acm.org/doi/10.5555/3104482.3104610

[71] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks,” 2018, doi: 10.48550/ARXIV.1801.01078.

[72] P. Kumari and D. Toshniwal, “Deep learning models for solar irradiance forecasting: A comprehensive review,” *J. Clean. Prod.*, vol. 318, p. 128566, Oct. 2021, doi: 10.1016/j.jclepro.2021.128566.

[73] C. Olah, “Understanding LSTM Networks,” colah’s blog. Accessed: Aug. 27, 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[74] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” 2014, doi: 10.48550/ARXIV.1412.3555.

[75] P. Jia, H. Zhang, X. Liu, and X. Gong, “Short-Term Photovoltaic Power Forecasting Based on VMD and ISSA-GRU,” *IEEE Access*, vol. 9, pp. 105939–105950, 2021, doi: 10.1109/ACCESS.2021.3099169.

[76] N. Q. Nguyen, L. D. Bui, B. V. Doan, E. R. Sanseverino, D. D. Cara, and Q. D. Nguyen, “A new method for forecasting energy output of a large-scale solar power plant

based on long short-term memory networks a case study in Vietnam," *Electr. Power Syst. Res.*, vol. 199, p. 107427, Oct. 2021, doi: 10.1016/j.epsr.2021.107427.

[77] Y. Lin, I. Koprinska, and M. Rana, "Temporal Convolutional Neural Networks for Solar Power Forecasting," in *2020 International Joint Conference on Neural Networks (IJCNN)*, Glasgow, United Kingdom: IEEE, July 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9206991.

[78] J. Zhu *et al.*, "Time-Series Power Forecasting for Wind and Solar Energy Based on the SL-Transformer," *Energies*, vol. 16, no. 22, p. 7610, Nov. 2023, doi: 10.3390/en16227610.

[79] H. Alharkan, S. Habib, and M. Islam, "Solar Power Prediction Using Dual Stream CNN-LSTM Architecture," *Sensors*, vol. 23, no. 2, p. 945, Jan. 2023, doi: 10.3390/s23020945.

[80] K. Wang, X. Qi, and H. Liu, "Photovoltaic power forecasting based LSTM-Convolutional Network," *Energy*, vol. 189, p. 116225, Dec. 2019, doi: 10.1016/j.energy.2019.116225.

[81] S. Khan, F. Shaikh, M. M. Siddiqui, T. Hussain, L. Kumar, and A. Nahar, "Hourly Forecasting of Solar Photovoltaic Power in Pakistan Using Recurrent Neural Networks," *Int. J. Photoenergy*, vol. 2022, pp. 1–11, Jan. 2022, doi: 10.1155/2022/7015818.

[82] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," 2014, *arXiv*. doi: 10.48550/ARXIV.1409.0473.

[83] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," 2014, doi: 10.48550/ARXIV.1406.1078.

[84] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, Sept. 2021, doi: 10.1016/j.neucom.2021.03.091.

[85] A. Vaswani *et al.*, “Attention Is All You Need,” 2017, *arXiv*. doi: 10.48550/ARXIV.1706.03762.

[86] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, “A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction,” 2017, *arXiv*. doi: 10.48550/ARXIV.1704.02971.

[87] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, “Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting,” 2019, *arXiv*. doi: 10.48550/ARXIV.1912.09363.

[88] H. Zhou *et al.*, “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting,” 2020, *arXiv*. doi: 10.48550/ARXIV.2012.07436.

[89] X. Hou, C. Ju, and B. Wang, “Prediction of solar irradiance using convolutional neural network and attention mechanism-based long short-term memory network based on similar day analysis and an attention mechanism,” *Heliyon*, vol. 9, no. 11, p. e21484, Nov. 2023, doi: 10.1016/j.heliyon.2023.e21484.

[90] R. Asghar, F. Riganti Fulginei, H. Wadood, and S. Saeed, “A Review of Load Frequency Control Schemes Deployed for Wind-Integrated Power Systems,” *Sustainability*, vol. 15, no. 10, p. 8380, May 2023, doi: 10.3390/su15108380.

[91] R. Rai, A. Das, S. Ray, and K. G. Dhal, “Human-Inspired Optimization Algorithms: Theoretical Foundations, Algorithms, Open-Research Issues and Application for Multi-Level Thresholding,” *Arch. Comput. Methods Eng.*, vol. 29, no. 7, pp. 5313–5352, Nov. 2022, doi: 10.1007/s11831-022-09766-z.

[92] M. M. Saber, M. A.Elsayed, A. Meshref, and M. A Elsisy, “Human-Based Optimization Algorithms and Their Applications,” *Benha J. Appl. Sci.*, vol. 9, no. 5, pp. 289–320, May 2024, doi: 10.21608/bjas.2025.280508.1389.

[93] S. Sridharan, R. K. Subramanian, and A. K. Srirangan, “Physics based meta heuristics in manufacturing,” *Mater. Today Proc.*, vol. 39, pp. 805–811, 2021, doi: 10.1016/j.matpr.2020.09.775.

[94] H. M. Alabool, D. Alarabiat, L. Abualigah, and A. A. Heidari, “Harris hawks optimization: a comprehensive review of recent variants and applications,” *Neural Comput. Appl.*, vol. 33, no. 15, pp. 8939–8980, Aug. 2021, doi: 10.1007/s00521-021-05720-5.

[95] A. G. Hussien *et al.*, “Recent Advances in Harris Hawks Optimization: A Comparative Study and Applications,” *Electronics*, vol. 11, no. 12, p. 1919, June 2022, doi: 10.3390/electronics11121919.

[96] C. Ouyang, C. Liao, D. Zhu, Y. Zheng, C. Zhou, and T. Li, “Integrated improved Harris hawks optimization for global and engineering optimization,” *Sci. Rep.*, vol. 14, no. 1, p. 7445, Mar. 2024, doi: 10.1038/s41598-024-58029-3.

[97] A. G. Hussien and M. Amin, “A self-adaptive Harris Hawks optimization algorithm with opposition-based learning and chaotic local search strategy for global optimization and feature selection,” *Int. J. Mach. Learn. Cybern.*, vol. 13, no. 2, pp. 309–336, Feb. 2022, doi: 10.1007/s13042-021-01326-4.

[98] S. Almotairi, E. Badr, M. Abdul Salam, and A. Dawood, “Three Chaotic Strategies for Enhancing the Self-Adaptive Harris Hawk Optimization Algorithm for Global Optimization,” *Mathematics*, vol. 11, no. 19, p. 4181, Oct. 2023, doi: 10.3390/math11194181.

[99] H. M. Ridha, A. A. Heidari, M. Wang, and H. Chen, “Boosted mutation-based Harris hawks optimizer for parameters identification of single-diode solar cell models,” *Energy Convers. Manag.*, vol. 209, p. 112660, Apr. 2020, doi: 10.1016/j.enconman.2020.112660.

[100] M. A. Elaziz, A. A. Heidari, H. Fujita, and H. Moayedi, “A competitive chain-based Harris Hawks Optimizer for global optimization and multi-level image thresholding problems,” *Appl. Soft Comput.*, vol. 95, p. 106347, Oct. 2020, doi: 10.1016/j.asoc.2020.106347.

[101] A. Wunnava, M. Kumar Naik, R. Panda, B. Jena, and A. Abraham, “A differential evolutionary adaptive Harris hawks optimization for two dimensional practical Masi entropy-based multilevel image thresholding,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3011–3024, June 2022, doi: 10.1016/j.jksuci.2020.05.001.

[102] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, “Robust Online Time Series Prediction with Recurrent Neural Networks,” in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Montreal, QC, Canada: IEEE, Oct. 2016, pp. 816–825. doi: 10.1109/DSAA.2016.92.

[103] L. N. Boucetta, Y. Amrane, and S. Arezki, “Comparative Analysis of LSTM, GRU, and MLP Neural Networks for Short-Term Solar Power Forecasting,” in *2023 International Conference on Electrical Engineering and Advanced Technology (ICEEAT)*, Batna, Algeria: IEEE, Nov. 2023, pp. 1–6. doi: 10.1109/ICEEAT60471.2023.10426346.

[104] R. Gupta and A. Chandel, “A bidirectional gated recurrent unit based novel stacking ensemble regressor for foretelling the global horizontal irradiance,” *Sustain. Comput. Inform. Syst.*, vol. 44, p. 101041, Dec. 2024, doi: 10.1016/j.suscom.2024.101041.

[105] A. Zameer, F. Jaffar, F. Shahid, M. Muneeb, R. Khan, and R. Nasir, “Short-term solar energy forecasting: Integrated computational intelligence of LSTMs and GRU,” *PLOS ONE*, vol. 18, no. 10, p. e0285410, Oct. 2023, doi: 10.1371/journal.pone.0285410.

[106] P. Kumari and D. Toshniwal, “Long short term memory–convolutional neural network based deep hybrid approach for solar irradiance forecasting,” *Appl. Energy*, vol. 295, p. 117061, Aug. 2021, doi: 10.1016/j.apenergy.2021.117061.

[107] A. Rai, A. Shrivastava, and K. C. Jana, “A CNN-BiLSTM based deep learning model for mid-term solar radiation prediction,” *Int. Trans. Electr. Energy Syst.*, vol. 31, no. 9, Sept. 2021, doi: 10.1002/2050-7038.12664.

[108] W. Liu, Q. Liu, and Y. Li, “Ultra-short-term photovoltaic power prediction based on modal reconstruction and BiLSTM-CNN-Attention model,” *Earth Sci. Inform.*, vol. 17, no. 3, pp. 2711–2725, June 2024, doi: 10.1007/s12145-024-01308-4.

[109] M. Aslam, S.-J. Lee, S.-H. Khang, and S. Hong, “Two-Stage Attention Over LSTM With Bayesian Optimization for Day-Ahead Solar Power Forecasting,” *IEEE Access*, vol. 9, pp. 107387–107398, 2021, doi: 10.1109/ACCESS.2021.3100105.

[110] J. Heo, K. Song, S. Han, and D.-E. Lee, “Multi-channel convolutional neural network for integration of meteorological and geographical features in solar power forecasting,” *Appl. Energy*, vol. 295, p. 117083, Aug. 2021, doi: 10.1016/j.apenergy.2021.117083.

[111] Z. A. Khan, T. Hussain, and S. W. Baik, “Dual stream network with attention mechanism for photovoltaic power forecasting,” *Appl. Energy*, vol. 338, p. 120916, May 2023, doi: 10.1016/j.apenergy.2023.120916.

[112] W. Liu and Z. Mao, “Short-term photovoltaic power forecasting with feature extraction and attention mechanisms,” *Renew. Energy*, vol. 226, p. 120437, May 2024, doi: 10.1016/j.renene.2024.120437.

[113] T. Bashir, H. Wang, M. Tahir, and Y. Zhang, “Wind and solar power forecasting based on hybrid CNN-ABiLSTM, CNN-transformer-MLP models,” *Renew. Energy*, vol. 239, p. 122055, Feb. 2025, doi: 10.1016/j.renene.2024.122055.

[114] G. Li, S. Guo, X. Li, and C. Cheng, “Short-term Forecasting Approach Based on bidirectional long short-term memory and convolutional neural network for Regional Photovoltaic Power Plants,” *Sustain. Energy Grids Netw.*, vol. 34, p. 101019, June 2023, doi: 10.1016/j.segan.2023.101019.

[115] U. K. Lilhore, S. Simiaya, M. Alhussein, N. Faujdar, S. Dalal, and K. Aurangzeb, "Optimizing protein sequence classification: integrating deep learning models with Bayesian optimization for enhanced biological analysis," *BMC Med. Inform. Decis. Mak.*, vol. 24, no. 1, p. 236, Aug. 2024, doi: 10.1186/s12911-024-02631-y.

[116] N. Wilkinson and T. Niesler, "A Hybrid CNN-BiLSTM Voice Activity Detector," 2021, *arXiv*. doi: 10.48550/ARXIV.2103.03529.

[117] H. Cho and H. Lee, "Biomedical named entity recognition using deep neural networks with contextual information," *BMC Bioinformatics*, vol. 20, no. 1, p. 735, Dec. 2019, doi: 10.1186/s12859-019-3321-4.

[118] G. Li *et al.*, "Outlier data mining method considering the output distribution characteristics for photovoltaic arrays and its application," *Energy Rep.*, vol. 6, pp. 2345–2357, Nov. 2020, doi: 10.1016/j.egyr.2020.08.034.

[119] V. Tomar, M. Bansal, and P. Singh, "Metaheuristic Algorithms for Optimization: A Brief Review," in *RAiSE-2023*, MDPI, Mar. 2024, p. 238. doi: 10.3390/engproc2023059238.

[120] X.-S. Yang, *Nature-Inspired Optimization Algorithms*. London, United Kingdom: Elsevier, 2014. doi: 10.1016/C2013-0-01368-0.

[121] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World: A Survey," *Int. J. Comput. Eng. Res. TRENDS*, vol. 2, no. 5, pp. 296–304, May 2015, [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=039da802e218c89a3c49e235af8e555b2fdfa063

[122] J. C. Bednarz, "Cooperative Hunting Harris' Hawks (*Parabuteo unicinctus*)," *Science*, vol. 239, no. 4847, pp. 1525–1527, Mar. 1988, doi: 10.1126/science.239.4847.1525.

[123] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, “Harris hawks optimization: Algorithm and applications,” *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019, doi: 10.1016/j.future.2019.02.028.

[124] A. A. Dehkordi, A. S. Sadiq, S. Mirjalili, and K. Z. Ghafoor, “Nonlinear-based Chaotic Harris Hawks Optimizer: Algorithm and Internet of Vehicles application,” *Appl. Soft Comput.*, vol. 109, p. 107574, Sept. 2021, doi: 10.1016/j.asoc.2021.107574.

[125] Y. H. Choo, Z. Cai, V. Le, M. Johnstone, D. Creighton, and C. P. Lim, “Enhancing the Harris’ Hawk optimiser for single- and multi-objective optimisation,” *Soft Comput.*, vol. 27, no. 22, pp. 16675–16715, Nov. 2023, doi: 10.1007/s00500-023-08952-w.

[126] D. T. Akl, M. M. Saafan, A. Y. Haikal, and E. M. El-Gendy, “IHHO: an improved Harris Hawks optimization algorithm for solving engineering problems,” *Neural Comput. Appl.*, vol. 36, no. 20, pp. 12185–12298, July 2024, doi: 10.1007/s00521-024-09603-3.

[127] W. Li, G.-G. Wang, and A. H. Alavi, “Learning-based elephant herding optimization algorithm for solving numerical optimization problems,” *Knowl.-Based Syst.*, vol. 195, p. 105675, May 2020, doi: 10.1016/j.knosys.2020.105675.

[128] E.-M. Nikolados, A. Wongprommoon, O. M. Aodha, G. Cambray, and D. A. Oyarzún, “Accuracy and data efficiency in deep learning models of protein expression,” *Nat. Commun.*, vol. 13, no. 1, p. 7755, Dec. 2022, doi: 10.1038/s41467-022-34902-5.

[129] X. Song, L. Deng, H. Wang, Y. Zhang, Y. He, and W. Cao, “Deep learning-based time series forecasting,” *Artif. Intell. Rev.*, vol. 58, no. 1, p. 23, Nov. 2024, doi: 10.1007/s10462-024-10989-8.

[130] Q. Yao, C.-H. H. Yang, R. Jiang, Y. Liang, M. Jin, and S. Pan, “Towards Neural Scaling Laws for Time Series Foundation Models,” 2024, *arXiv*. doi: 10.48550/ARXIV.2410.12360.

[131] S. Aburass and M. A. Rumman, “Quantifying Overfitting: Introducing the Overfitting Index,” in *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET*, Sydney, Australia: IEEE, July 2024, pp. 1–7. doi: 10.1109/ICECET61485.2024.10698575.

[132] W. Wang, Y. Li, A. Li, J. Zhang, W. Ma, and Y. Liu, “An Empirical Study on Noisy Label Learning for Program Understanding,” Dec. 31, 2023, *arXiv*: arXiv:2307.08990. doi: 10.48550/arXiv.2307.08990.

[133] T. He, S. Yu, Z. Wang, J. Li, and Z. Chen, “From Data Quality to Model Quality: an Exploratory Study on Deep Learning,” June 10, 2019, *arXiv*: arXiv:1906.11882. doi: 10.48550/arXiv.1906.11882.

[134] T. Zhou, S. Ruan, and H. Hu, “A literature survey of MR-based brain tumor segmentation with missing modalities,” *Comput. Med. Imaging Graph.*, vol. 104, p. 102167, Mar. 2023, doi: 10.1016/j.compmedimag.2022.102167.

[135] Y. Ba, M. V. Mancenido, and R. Pan, “Data Diversity as Implicit Regularization: How Does Diversity Shape the Weight Space of Deep Neural Networks?,” 2024, *arXiv*. doi: 10.48550/ARXIV.2410.14602.

[136] wikipedia, “Alice Springs.” [Online]. Available: https://en.wikipedia.org/wiki/Alice_Springs

[137] Australia, “The climate of Alice Springs,” Tourism Australia. [Online]. Available: https://www.australia.com/it-it/facts-and-planning/weather-in-australia/alice-springs-weather.html

[138] DKA, “DKA Solar Centre.” [Online]. Available: https://dkasolarcentre.com.au/download?location=alice-springs

[139] J. Sun, Y. Zuo, R. Sun, and L. Zhou, “Research on the conversion efficiency and preparation technology of monocrystalline silicon cells based on statistical distribution,”

Sustain. Energy Technol. Assess., vol. 47, p. 101482, Oct. 2021, doi: 10.1016/j.seta.2021.101482.

[140] M. Hasanuzzaman, Ed., *Technologies for solar thermal energy: theory, design and optimization*. London, UK: Academic Press, 2022.

[141] M. Alabadla *et al.*, "Systematic Review of Using Machine Learning in Imputing Missing Values," *IEEE Access*, vol. 10, pp. 44483–44502, 2022, doi: 10.1109/ACCESS.2022.3160841.

[142] H. Seo, M. Bassenne, and L. Xing, "Closing the Gap Between Deep Neural Network Modeling and Biomedical Decision-Making Metrics in Segmentation via Adaptive Loss Functions," *IEEE Trans. Med. Imaging*, vol. 40, no. 2, pp. 585–593, Feb. 2021, doi: 10.1109/TMI.2020.3031913.

[143] T. Blu, P. Thevenaz, and M. Unser, "Linear Interpolation Revitalized," *IEEE Trans. Image Process.*, vol. 13, no. 5, pp. 710–719, May 2004, doi: 10.1109/TIP.2004.826093.

[144] H. Alimohammadi and S. Nancy Chen, "Performance evaluation of outlier detection techniques in production timeseries: A systematic review and meta-analysis," *Expert Syst. Appl.*, vol. 191, p. 116371, Apr. 2022, doi: 10.1016/j.eswa.2021.116371.

[145] A. Ur Rehman and S. B. Belhaouari, "Unsupervised outlier detection in multidimensional data," *J. Big Data*, vol. 8, no. 1, p. 80, Dec. 2021, doi: 10.1186/s40537-021-00469-z.

[146] Y. Yu, Y. Zhu, S. Li, and D. Wan, "Time Series Outlier Detection Based on Sliding Window Prediction," *Math. Probl. Eng.*, vol. 2014, no. 1, p. 879736, Jan. 2014, doi: 10.1155/2014/879736.

[147] Y.-S. Kim, M. K. Kim, N. Fu, J. Liu, J. Wang, and J. Srebric, "Investigating the impact of data normalization methods on predicting electricity consumption in a building

using different artificial neural network models," *Sustain. Cities Soc.*, vol. 118, p. 105570, Jan. 2025, doi: 10.1016/j.scs.2024.105570.

[148] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Appl. Soft Comput.*, vol. 97, p. 105524, Dec. 2020, doi: 10.1016/j.asoc.2019.105524.

[149] P. J. Muhammad Ali, "Investigating the Impact of Min-Max Data Normalization on the Regression Performance of K-Nearest Neighbor with Different Similarity Measurements," *ARO- Sci. J. KOYA Univ.*, vol. 10, no. 1, pp. 85–91, June 2022, doi: 10.14500/aro.10955.

[150] N. Caliskan, E. Jadraque, Y. Tham, and T. Muneer, "Evaluation of the accuracy of mathematical models through use of multiple metrics," *Sustain. Cities Soc.*, vol. 1, no. 2, pp. 63–66, July 2011, doi: 10.1016/j.scs.2011.02.001.

[151] T. Tanyıldızı Ağır, "Very Short-Term Solar Power Forecasting Using Hybrid LSTM-SVM," *Bitlis Eren Üniversitesi Fen Bilim. Derg.*, vol. 14, no. 2, pp. 677–696, June 2025, doi: 10.17798/bitlisfen.1581731.

[152] J. Qu, Z. Qian, and Y. Pei, "Day-ahead hourly photovoltaic power forecasting using attention-based CNN-LSTM neural network embedded with multiple relevant and target variables prediction pattern," *Energy*, vol. 232, p. 120996, Oct. 2021, doi: 10.1016/j.energy.2021.120996.

[153] V. Pfister, M. Botticelli, F. Lauro, and S. Romano, "Case Study Analysis of Energy Monitoring for Prosumer Engagement and Energy Awareness," in *2024 IEEE International Humanitarian Technologies Conference (IHTC)*, Bari, Italy: IEEE, Nov. 2024, pp. 1–6. doi: 10.1109/IHTC61819.2024.10855059.

[154] P. Zippenfenig, *Open-Meteo.com Weather API*. (Dec. 31, 2024). Zenodo. doi: 10.5281/ZENODO.7970649.