2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Differential Evolution Algorithm based Hyper-Parameters Selection of Transformer Neural Network Model for Load Forecasting

1st Anuvab Sen
Electronics and Telecommunication
Indian Institute of Engineering
Science and Technology, Shibpur
Howrah, India
sen.anuvab@gmail.com

2rd Arul Rhik Mazumder
Computer Science
School of Computer Science
Carnegie Mellon University
Pittsburgh, United States of America
arul.rhik@gmail.com

3rd Udayon Sen
Computer Science and Technology
Indian Institute of Engineering
Science and Technology, Shibpur
Howrah, India
udayon.sen@gmail.com

***Abstract*—Accurate load forecasting plays a vital role in numerous sectors, but accurately capturing the complex dynamics of dynamic power systems remains a challenge for traditional statistical models. For these reasons, time-series models (ARIMA) and deep-learning models (ANN, LSTM, GRU, etc.) are commonly deployed and often experience higher success. In this paper, we analyze the efficacy of the recently developed Transformer-based Neural Network model in load forecasting. Transformer models have the potential to improve load forecasting because of their ability to learn long-range dependencies derived from their Attention Mechanism. We apply several metaheuristics namely Differential Evolution to find the optimal hyperparameters of the Transformer-based Neural Network to produce accurate forecasts. Differential Evolution provides scalable, robust, global solutions to non-differentiable, multi-objective, or constrained optimization problems. Our work compares the proposed Transformer-based Neural Network model integrated with different metaheuristic algorithms by their performance in load forecasting based on numerical metrics such as Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE). Our findings demonstrate the potential of metaheuristic-enhanced Transformer-based Neural Network models in load forecasting accuracy and provide optimal hyperparameters for each model.**

***Index Terms*—Deep Learning, Differential Evolution, Particle Swarm Optimization, Genetic Algorithm, Metaheuristics**

I. Introduction

Load forecasting is the application of science and technology to predict the future demand for electricity or power in a given geographical location, for some specific future time. It plays a crucial role in various sectors, such as energy, trading and markets, infrastructure planning, disaster management, etc., to name a few. Traditional load prediction methods rely on historical data and models that simulate patterns of electricity consumption, but such models often face challenges in accurately capturing the complex dynamics of power systems [1]. To model this complexity, time series models like Auto-Regressive Moving Average (ARIMA) [2] various deep learning techniques have been introduced such as Artificial Neural Networks (ANN) [3], Recurrent Neural Networks (RNN) [4], Long Short-Term Memory (LSTM) [5], and Gate Recurrent Units (GRU) [6]. The models work to improve the accuracy of load forecasts by leveraging large datasets and discovering hidden patterns to predict future values.

Recently Transformer models [7] have revolutionized machine learning due to their unique architecture. Because of the capability to run parallelly across multiple GPUs, they perform more efficiently compared to other deep learning models and take less time to train compared to sequential models such as LSTMs [8]. Furthermore, as the Transformer model generates results after training through backpropagation, they can generate future results using a larger reference window in comparison to RNNs, LSTMs, and GRUs [9]. This window gives Transformers a better ability to identify long-range dependencies in sequences and better resistance towards the vanishing gradient problem [10] compared to other deep learning models. The Transformer's strength in identifying long-range dependencies made them the optimal model for natural language processing and they are used in machine translation, text generation, speech recognition, and more.

Like any other deep learning model, their performance depends on the chosen hyperparameters. In this work we utilized metaheuristics Genetic Algorithm [11], Differential Evolution [12], and Particle Swarm Optimization [13] to identify ideal hyperparameters. Although hyperparameter search techniques like Grid Search [14], Random Search [15], and Bayesian Optimization [16] are substantial improvements to manual tuning, they are inferior to the metaheuristics discussed this paper. The metaheuristics are more efficient than grid search and random search and more robust and scalable than Bayesian Optimization. Furthermore, these algorithms can be applied to nonlinear, nonconvex, and noncontinuous functions [17] [18] [19].

Traditional Transformers take a sequence of tokenized inputs. For Natural Language Processing these inputs are words but can be generalized to other sequential data for other tasks. These tokens are then run through several

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

encoder and decoder layers. Encoders process the input using the self-attention mechanism to find dependencies between tokens and positional encoding to maintain the ordering of tokens. The decoders then generate output token sequences using similar self-attention mechanisms, but also a unique encoder-decoder attention layer that allows it to read the encoded information.

In this work, we created a custom Transformer Neural network model. Our model only uses the encoder of the Transformer and uses it to enhance Deep Learning Models for load forecasting. This research is unique by investigating the Transformer's Attention Mechanism capabilities outside of the usual scope of natural language processing. We identify that the Transformer's abilities in long-range dependencies can be applied to load forecasting.

Our work seeks to fill the void and propose Differential Evolution optimized custom Transformer Neural Networks specifically designed for load forecasting. To evaluate the results we also integrated Particle Swarm Optimization and Genetic Algorithm with the Transformer Neural Networks to benchmark against our proposed Differential Evolution integrated Transformer Neural Network. In particular, our work is the first to propose a Differential Evolution-based hyperparameter tuning scheme for a Transformer-based Neural Network model for load forecasting.

II. Preliminaries

A. Differential Evolution

Differential Evolution (DE) is a stochastic population-based optimization algorithm developed by Rainer Storn and Kenneth Price in 1997. It is used to find approximate solutions to a wide class of challenging objective functions. DE can be used on functions that are nondifferentiable, non-continuous, non-linear, noisy, flat, multi-dimensional, possess multiple local minima, contain constraints, or are stochastic [20]. A general problem formulation that DE could solve is:

$$\text{For objective function } f : X \subseteq \mathbb{R}^n \rightarrow \mathbb{R} \text{ where } X \neq \emptyset$$
$$\text{find } s \in X \text{ s.t. } f(s) \leq f(x)\ \forall x \in X \text{ where } f(s) \neq -\infty$$

Its versatility comes from its unique implementation that does not require the gradient of the function. DE obtains a minimum solution by initializing a set of candidate solutions and iteratively improving each solution by applying various genetic operators [21].

1) Initialization: Suppose f has D parameters. An N-sized candidate solution population is initialized, with each candidate solution modeled as x_i, a D-parameter vector.

$$x_{i,G} = [x_{1,i,G}, x_{2,i,G} ... x_{D,i,G}] \text{ where } i = 1, 2...N$$
$$\text{and } G \text{ is the generation number}$$

Each index $x_{j,i,G}$ with $j = 1, 2...D$ represents a parameter to be manipulated approximate a solution to the objective function [22]. During the initialization of the first generation, each parameter for all candidate solutions is set randomly from bounds $[x_j^L, x_j^U]$.

$$x_j^L \leq x_{j,i,1} \leq x_j^U$$

2) Mutation: A mutation is a stochastic change that expands the candidate solution search space. Mutations are used in DE to prevent the algorithm from converging upon a local optimum [23]. In the original mutation scheme devised by Storn, a mutant vector v_i is created from randomly sampling three candidate solution vectors v_{r_1}, v_{r_2}, v_{r_3} such that r_1, r_2, r_3 and i are distinct. The mutant vector is obtained by adding the weighted difference of two of the vectors to the third.

$$v_{i,G+1} = v_{r_1,G} + F \times (v_{r_2,G} - v_{r_3,G})$$

$F \in [0, 2]$ represents the scale factor controlling the magnitude of the mutation.

3) Crossover: Crossover is how successful candidate solutions pass their characteristics to the following generations. A trial vector $u_{i,G+1}$ is created by combining the original vector $x_{i,G}$ and its corresponding mutant vector $v_{i,G+1}$. A widely used crossover scheme is described below: [24].

$$u_{j,i,G+1} = \left\{ \begin{array}{ll} v_{j,i,G+1}, & \text{if } p_{rand}\ U(0,1) \leq CR \\ x_{j,i,G} & \text{else} \end{array} \right\}$$

Each $j = 1, 2....D$ and $v_{i,G+1} \neq x_{i,G}$

4) Selection: Given both the initial target vector and generated trial vector, the fitness of each is evaluated using the initial objective or cost function f. The vector with the lower cost is passed to the next generation.

$$x_{i,G+1} = \left\{ \begin{array}{ll} u_{i,G+1}, & \text{if } f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} & \text{else} \end{array} \right\}$$

The Differential Evolution Algorithm is illustrated in Figure 1 below.

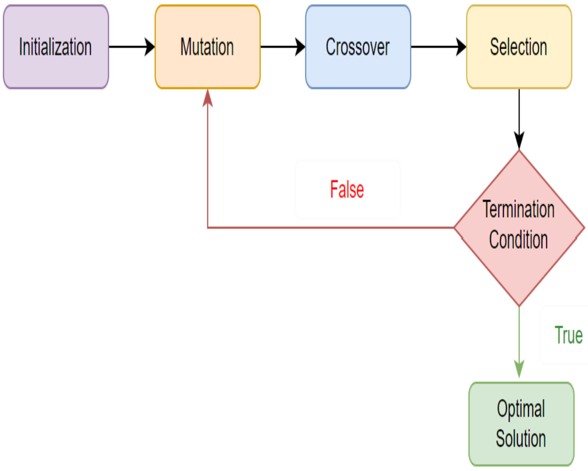

Fig. 1. Differential Evolution Algorithm

Mutation, crossover, and selection are cycled until either the maximum number of generations is attained or the candidate solutions meet a predefined accuracy threshold. Differential evolution (DE) thus operates by

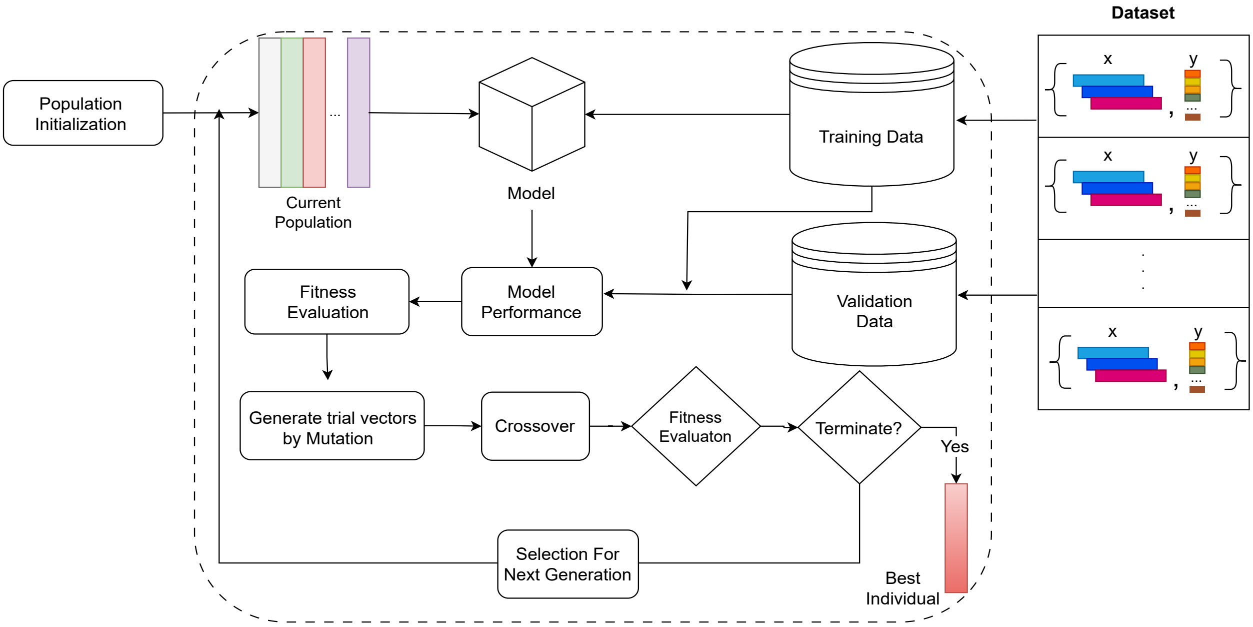

Fig. 2. Mechanism of the Differential algorithm based hyper-parameters selection approach for the Load Forecasting task.

iteratively updating the candidate solutions and evolving the population over multiple generations iteratively. The process is repeated for specific number of generations, until a termination criterion is satisfied, or a desired level of convergence (global minimum) is achieved eventually.

III. Proposed Approach

This section describes the implementation of integrating metaheuristics with the Transformer-based Neural Network for load forecasting. Each metaheuristic is used to identify the optimal set of hyperparameters and the efficacy of the hyperparameters is measured using the Mean Squared Error (MSE) and Mean Average Percentage Error (MAPE) metrics. The integrated Differential Evolution mechanism selection strategy for the hyperparameters is outlined in Figure 2 above.

A. Transformer-based Neural Network

We implemented the Transformer-based Neural Network by building a sequential model and sequentially adding layers. The input layer is initialized with 36 nodes and then passed through a dense time-distributed Transformer-based Neural Network layer of 64 nodes. Next, we have an 8-headed attention layer with a dimension of 64 and a dropout rate of 0.1, where the output of the previous operation is applied. The result is then flattened and run through 2 dense layers containing 64 nodes before returning through a 24-node output layer. The activation function used for all cases, except for the output layer, is Rectified Linear Units (ReLU) [25] (modeled below) except for the output layer.

$$f(x) = max(0, x)$$

The output uses a Linear Activation Function.

$$f(x) = x$$

All Transformer-based Neural Networks use the implemented form of metaheuristics algorithms to optimize the batch size, learning rate hyperparameters, and epoch. Optimization is done by minimizing loss using Mean Squared Error. The metaheuristic-optimized Transformer-based Neural Networks are assessed by comparing the MAPE for each set of hyperparameters found.

The entire architecture of the proposed model is portrayed in Figure 3 below.

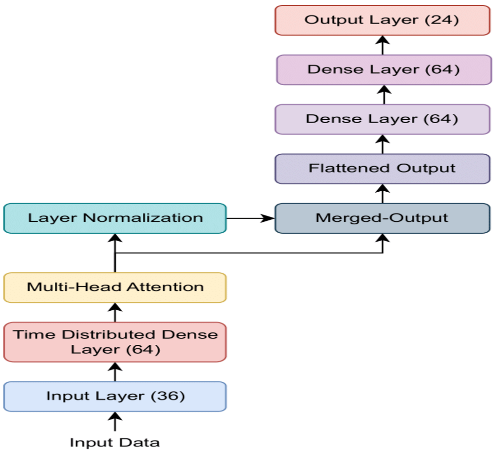

Fig. 3. The proposed Transformer-based deep learning model

The multi-headed attention is primarily used for the model to simultaneously operate different parts of the input sequence, improving the performance. The normalization layer is then passed on top of the attention layer to make the model robust. This avoids the scenario of the model relying too much on specific features, which reduces over-fitting. The best-optimized set of hyperparameters obtained from each metaheuristic algorithm is then applied to each Transformer-based Neural Network and subsequently tested on the test dataset to generate the results.

Each model's best set of batch size epoch and learning rate are summarized in Table I below:

TABLE I
BEST SET OF HYPERPARAMETERS FOR TRANSFORMER

Metaheuristics	Batch Size	Epoch	Learning Rate
Genetic Algorithm	80	844	0.0001
Particle Swarm	173	35	0.3109
Differential Evolution	24	1000	0.1

After passing the attention layer, the output is flattened. This means it is reshaped from a 3D to a 2D tensor, which allows the subsequent layers to treat the output as a sequence of 2D inputs. The flattened output is then passed through two dense layers, each consisting of 64 layers. These dense layers and increased nodes allow the model to capture more complex patterns and relationships in the data.

The output layer contains 24 nodes, as the model is designed to produce a prediction 24 hours ahead. The custom transformer model architecture developed is intended for the specific task of short-term load forecasting. Its primary aim is to aid the industry by operating on load data to predict variations in various load parameters.

IV. EXPERIMENTAL DETAILS

A. Dataset Description

For this project, the *Load Dataset*[1] was curated using meteorological data scraped from the official website of the Government of Canada [26]. The dataset covers the period from 1$^{\text{st}}$ January 2017 to 4$^{\text{th}}$ July 2023 in Ottawa, Ontario. It contains 19 variables, capturing details such as date, time (in 24 hours), year, quarter, month, week of the year, day of the year, state holiday, hour of the day, day of the week, day type, temperature (in °C), dew point temperature (in °C), relative humidity (%), wind speed (in km/h), visibility (in km), precipitation amounts (in mm), daily peak (in MW), and hourly demand (in MW). In total, there are 96,432 rows, with each row representing data for a specific hour.

B. Preprocessing

During the preprocessing stage, we addressed missing data in the compiled dataset. Since the precipitation column had significant missing information, it was excluded from the analysis. Regarding the temperature, only 0.03% of the data was missing. To forecast predictions up to 24 hours into the future, we used 3 hours of past data. The data was standardized using the StandardScaler function from the sklearn.preprocessing library [27].

The dataset was then split into three subsets: the training dataset, denoted as D_{train}, the validation dataset, denoted as D_{val}, and the testing dataset, denoted as D_{test}. The training dataset covers the period from January 1st, 2017, to December 31st, 2020. Within this dataset, 25% of the data was allocated to the validation dataset. The remaining data, extending until July 14th, 2023, constitutes the testing dataset.

C. Experimental Setups

The experiments of this work are implemented in Python 3.10.11 using three libraries : Tensorflow 2.11.0, Tensorflow built in Keras, and Numpy 1.21.

V. RESULTS AND DISCUSSION

We obtained the mean absolute percentage error (MAPE) using the proposed approach to implement the differential evolution-based hyperparameter tuning of the Transformer-based deep Neural Network offered in the preceding section. This MAPE was compared to the MAPEs generated from the proposed approach with the genetic algorithm and particle swarm optimization-based hyperparameter tuning of the custom architecture. The codes used in this paper are linked below[2].

The Standard scaler has been used to improve the convergence and stability of seasonal data during model training. This scaler prevents features of larger sizes from dominating the training process and also normalizes the dataset, allowing the model to learn effectively from the data. These steps are necessary for improving forecasting models.

The mean squared error (MSE) is used to measure the fitness of the differential evolution algorithm.

MSE serves as the loss function and is plotted against the number of epochs for the entire training duration as shown in Figure 4.

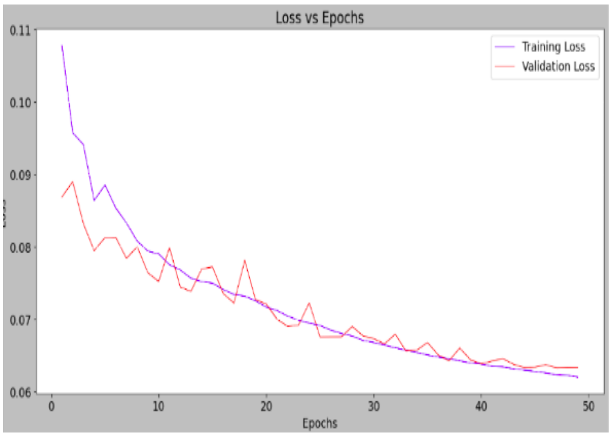

Fig. 4. Training & Validation Loss vs Epochs plots for the Transformer-based Neural Network DE model

The plot helps us to observe how the loss changes over time and whether the model is optimizing or overfitting.

Mean Absolute Percentage Error (MAPE) is used to gauge the accuracy of the entire model. It provides a

[1] Dataset Link: https://doi.org/10.7910/DVN/O8QA5H

[2] Code Link: https://github.com/AnuvabSen1/Meta-Transformer

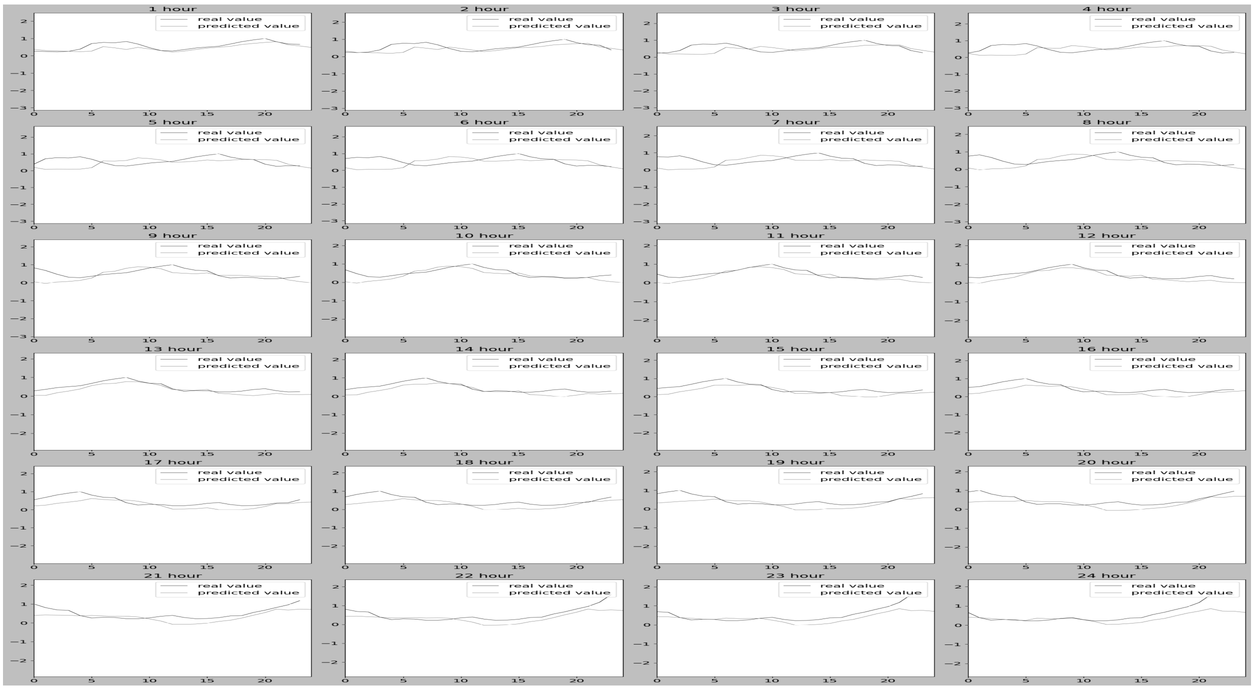

Fig. 5. Predicted plots for hourly demand for next 24 hours starting from the N-th hour for Transformer-based Neural Network DE Model.

measure of the average percentage difference between predicted values and the actual values.

Table II below provides us with a comparison of MAPE among various metaheuristic optimization algorithms used here.

TABLE II
COMPARISON OF METAHEURISTICS AND MAPE FOR BEST SET OF HYPERPARAMETERS FOR TRANSFORMER

Metaheuristics	MAPE
Manual Selection	2.07
Genetic Algorithm	1.31
Particle Swarm	1.28
Differential Evolution	1.11

The results prove that Differential Evolution (DE) algorithm outperforms the Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) in terms of mean absolute percentage error (MAPE).

Differential Evolution's superior ability can be attributed to a few factors. DE can more effectively explore the search space and exploit the promising regions for optimal solutions using its various genetic operators, therefore producing most desirable results. To visually understand the results and accuracy of the load forecasting model proposed here we have used two plots.

The first plot provides us with a 24-hour prediction for the best-performing DE on the Transformer-based Neural Network model as shown in Figure 6.

This shows that DE on Transformer-based Neural Network gives a fairly accurate prediction on Test data. The second graph plots the hourly demand variation for 24 hours starting from the N th hour shown in Figure 5.

The plots indicate that the accuracy decreases as N increases or as further in time we want to predict the less accurate results we obtain. The mutation operator sets random disturbances to ensure the prevention of early convergence towards a local minimum. The crossover operator passes on successful attributes to accelerate the convergence process even further. The selection operator preserves the fittest candidate solutions to improve the quality of results.

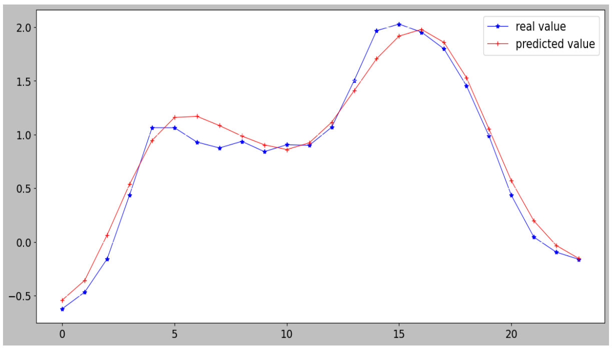

Fig. 6. 24-hours ahead forecast plot for the Transformer-based Neural Network DE model

The results ascertain that metaheuristic optimization algorithms consistently outperform the manual selection method of selecting hyperparameters. Particle Swarm Optimization (PSO) performs better than Genetic Algorithm (GA) but falls short behind Differential Evolution (DE). PSO suffers from rapid convergence, limiting its ability to reach the global optimum, which could be an explanation for its performance.

VI. CONCLUSION AND FUTURE WORK

This paper applies several metaheuristic algorithms to a custom Transformer-based Neural Network to find the optimal hyperparameters. This selection method was proven to be far more efficient and accurate than manual selection. Amongst the metaheuristics tested, Differential Evolution proved to be the best because of its mutation and selection operators which not only allowed the algorithm to thoroughly search the sample space but the filter and refine the best solutions. Differential Evolutions performance was then followed by Particle Swarm Optimization and finally Genetic Algorithm.

Due to possessing limited computational resources, each metaheuristic algorithm couldn't be applied to sufficiently large populations over many generations. If this research is extended with more powerful devices, future studies over larger populations and more generations will corroborate our findings. Future study may investigate the performance of other alternative metaheuristic algorithms on hyperparameter tuning for similar deep learning models, across a wide range of forecasting tasks.

VII. Acknowledgement

This work was made possible through the generous backing of Mitacs and the dedicated support of Dr. Chi Tang, Associate Professor at McMaster University. Their invaluable support enabled Anuvab Sen to embark on an enriching journey of undergraduate research at McMaster University, Canada.

References

[1] Sofi, S. & Oseledets, I. A case study of spatiotemporal forecasting techniques for Load forecasting. (2022)

[2] Smith, T. & Others. pmdarima: ARIMA estimators for Python. Available at http://www.alkaline-ml.com/pmdarima. [Online; accessed October 5, 2023]

[3] McCulloch, W. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin Of Mathematical Biophysics*. **5**, 115-133 (1943)

[4] Rumelhart, D., Hinton, G. & Williams, R. Learning Internal Representations By Error Propagation. (1985)

[5] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation*. **9**, 1735-1780 (1997)

[6] Gao, Y. & Glowacka, D. Deep Gate Recurrent Neural Network. (2016)

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. & Polosukhin, I. Attention Is All You Need. *CoRR*. **abs/1706.03762** (2017), http://arxiv.org/abs/1706.03762

[8] Miao, X., Wang, Y., Jiang, Y., Shi, C., Nie, X., Zhang, H. & Cui, B. Galvatron. *Proceedings Of The VLDB Endowment*. **16**, 470-479 (2022,11), https://doi.org/10.14778%252F3570690.3570697

[9] Devlin, J., Chang, M., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805

[10] Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal Of Uncertainty, Fuzziness And Knowledge-Based Systems*. **6**, 107-116 (1998)

[11] Holland, J. Genetic algorithms. *Scientific American*. **267**, 66-72 (1992)

[12] Storn, R. & Price, K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal Of Global Optimization*. **11**, 341-359 (1997)

[13] Kennedy, J. & Eberhart, R. Particle swarm optimization. *Proceedings Of ICNN'95 - International Conference On Neural Networks*. **4**, pp. 1942-1948 vol.4 (1995)

[14] LaValle, S., Branicky, M. & Lindemann, S. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal Of Robotics Research*. **23**, 673-692 (2004)

[15] Bergstra, J. & Bengio, Y. Random Search for Hyper-Parameter Optimization. *Journal Of Machine Learning Research*. **13**, 281-305 (2012)

[16] Snoek, J., Larochelle, H. & Adams, R. Practical Bayesian Optimization of Machine Learning Algorithms. (2012)

[17] A. Sen, V. Gupta, and C. Tang, *Differential Evolution Algorithm Based Hyperparameter Selection of Gated Recurrent Unit for Electrical Load Forecasting*, *arXiv preprint arXiv:2309.13019*, 2023.

[18] Sen, A., Rhik Mazumder, A., Dutta, D., Sen, U., Syam, P., Dhar, S. 2023. Comparative Evaluation of Metaheuristic Algorithms for Hyperparameter Selection in Short-Term Weather Forecasting. arXiv e-prints. doi:10.48550/arXiv.2309.02600, https://arxiv.org/abs/2309.02600

[19] A. Mazumder, A. Sen, and U. Sen, *Benchmarking Metaheuristic-Integrated Quantum Approximate Optimisation Algorithm against Quantum Annealing for Quadratic Unconstrained Binary Optimization Problems*, *arXiv preprint arXiv:2309.16796*, 2023.

[20] Rocca, P., Oliveri, G. & Massa, A. Differential Evolution as Applied to Electromagnetics. *IEEE Antennas And Propagation Magazine*. **53**, 38-49 (2011)

[21] Storn, R. On the usage of differential evolution for function optimization. *On The Usage Of Differential Evolution For Function Optimization*. pp. 519 - 523 (1996,7)

[22] Price, K., Storn, R. & Lampinen, J. Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series). (Springer-Verlag,2005)

[23] Zaharie, D. Critical values for the control parameters of differential evolution algorithms. *Critical Values For The Control Parameters Of Differential Evolution Algorithmss*. **2** pp. 62-67 (2002,1)

[24] Georgioudakis, M. & Plevris, V. A comparative study of differential evolution variants in constrained structural optimization. *Frontiers*. (2020,6), https://www.frontiersin.org/articles/10.3389/fbuil.2020.00102/full

[25] Agarap, A. Deep learning using rectified linear units (relu). *ArXiv Preprint ArXiv:1803.08375*. (2018)

[26] Canada, E. & Change, C. Government of Canada / gouvernement du Canada. *Climate*. (2023,5), https://climate.Load.gc.ca/historical_data/search_historic_data_e.html

[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. & Others. Scikit-learn: Machine learning in Python. *Journal Of Machine Learning Research*. **12**, 2825-2830 (2011)