LAUTECH Journal of Engineering and Technology 19 (5) 2025: 180-190
10.36108/laujet/5202.91.0551

Impact of Hyperparameter Tuning on Hybridised Convolutional Neural Networks for Pathloss Modelling in Mobile Communication Systems

Jimoh A. A., Ojo F. K. and *Adeyemo Z. K.

Department of Electronic and Electrical Engineering, Faculty of Engineering and Technology, Ladoke Akintola University of Technology, Ogbomoso.

Article Info

Article history:

Received: Oct. 18, 2025
Revised: Dec. 22, 2025
Accepted: Jan. 07, 2026

Keywords:

Convolutional Neural Networks,
Hyperparameters,
Mobile Network Systems,
Pathloss Modelling

Corresponding *Author:*
zkadeyemo@lautech.edu.ng

ABSTRACT

The performance of machine learning models, particularly Convolutional Neural Networks (CNNs), is profoundly influenced by effective hyperparameter tuning. However, a comprehensive understanding of how these hyperparameters affect the predictive accuracy of CNN-based pathloss models has not been adequately carried out. This study explores the role of hyper-parameter tuning in a hybridised CNN architecture that integrates DenseNet121 and ResNet50 to enhance pathloss prediction in mobile network environments. Field measurements were conducted along strategically selected urban and suburban routes in Ilorin, Kwara State, Nigeria. The results revealed the critical influence of key hyperparameters, such ashidden layers, batch size, training epochs, and computational efficiency, on model performance. Initially, with only two (2) hidden layers, the model showed suboptimal predictive accuracy, characterised by an MAE of 25.15, a MSE of 34.43, and a highly negative R² value of 6.01. However, increasing the hidden layers to seventeen(17) yielded a substantial improvement, with the MAE reducing to 2.08, the MSE decreasing to 7.35, and the R² shifting positively to 0.80. Further analysis of batch sizes revealed that smaller sizes resulted in poor model performance, increasing it to 8 significantly enhanced accuracy. Additionally, an increase in training epochs from 50 to 200 led to a marked reduction in prediction errors, albeit at the expense of extended training time per iteration. These findings underscore the pivotal role of strategic hyperparameter selection in optimising CNN-based pathloss modelling, offering valuable insights for enhancing predictive performance in mobile network systems.

INTRODUCTION

Pathloss modelling plays a vital role in the development and optimisation of mobile communication networks (Abdulkarim *et al.*, 2022), as accurate pathloss prediction enables efficient network planning, resource allocation, and signal coverage estimation (Tushar and Jadon, 2013). Traditional empirical pathloss models, while widely used, often struggle to adapt to the diverse and dynamic nature of mobile environments, limiting their predictive accuracy (Oyelaran *et al.*, 2024). Machine Learning (ML)-based pathloss models have appeared as promising alternatives, exploring the ability of ML to find complex relationships within input data. In particular, deep learning approaches such as Convolutional Neural Networks (CNNs) have demonstrated superior performance by extracting both linear and nonlinear features from the input data during machine training (Al-Hakim and Prasetiyo, 2024). Unlike conventional empirical models, CNNs can learn spatial dependencies and generalise better across varying network conditions. Recent studies have explored various ML models for pathloss prediction, including Support Vector Machines, Random Forest, Extreme Gradient Boosting, and Transformers (Abdollahzadeh *et al.*, 2024). While these models offer advantages in different contexts, they lack the deep feature extraction capabilities inherent in CNNs. The CNNs, with their hierarchical architecture, can process large input datasets, capturing intricate patterns that influence signal propagation (Benz *et al.,* 2021). Despite the promising potential of CNNs, optimising their performance requires careful tuning of hyperparameters such as epoch count, batch size, learning rate, activation functions, and the number of hidden layers. Understanding how these hyperparameters affect the predictive accuracy of CNN-based models remains crucial (Arnold *et al.,* 2024). This study investigates the impact of hyper-parameter tuning on a hybridised CNN architecture combining DenseNet and ResNet to enhance pathloss prediction in mobile network systems. However, CNNs have various architectural variants, each designed with improved features and layer structures tailored for specific applications. The choice of a CNN variant depends on the nature of the input data, research objectives, and the intended area of application (Christopher *et al.,* 2023). For instance, deep learning models, including CNNs, often encounter the challenge of vanishing gradients, which can hinder optimal learning and model convergence (Ahmed *et al.,* 2024). One of the CNN variants designed to address this issue is DenseNet, which introduces densely connected layers that enhance gradient flow and improve feature propagation. DenseNet's architecture mitigates the vanishing gradient problem by establishing direct connections between layers, facilitating efficient learning and parameter optimization (Hasan *et al.,* 2021). This design is particularly helpful for deep networks that require complex feature extraction without gradient decay. However, to further enhance feature reuse and model stability, integrating DenseNet with ResNet's residual blocks, which are thoughtfully incorporated to complement DenseNet by preserving essential features and mitigating gradient vanishing through shortcut connections (Gao *et al.,* 2022). This hybridisation improves predictive performance, making it well-suited for pathloss modelling and other deep learning tasks requiring robust feature extraction. Proper tuning of hyperparameters, such as learning rate and activation functions, further ensures efficient model convergence (Belete and Huchaiah, 2022). This research therefore hybridised the variants of DenseNet121 and ResNet50of CNNs, where the residual network connection in the ResNet50 layers allows a feature reuse capability for the hybridised architecture to mitigate any gradient that may vanish during training while sustaining the deep learning requirements for excellent feature extraction. The performance of this new hybridised architecture requirescareful tuning of the hyperparameters, including epoch count, batch size, learning rate, activation functions, and the number of hidden layers. The impact of these hyperparameters on the learning dynamics and predictive accuracy of CNN-based pathloss models is still an area of active research.This study,therefore,aims to assess the influence of hyperparameter tuning on the performance of a hybridised CNN model combining DenseNet121 and ResNet50 for pathloss prediction by systematically evaluating the hyperparameter configurations. The study looks to provide insights into optimizing CNN-based pathloss models for improved prediction accuracy and generalization in diverse mobile network environments.The remaining section in this article isorganised as follows: Section 2 presents the Evolution of hyper-parameter tuning in machine learning, while Section 3 presents the methodology. In section 4, the results and discussions were presented. Section 5 concludes the article and then presents the recommendations.

EVOLUTION OF HYPERPARAMETER TUNING IN MACHINE LEARNING

The development of hyperparameter tuning strategies has evolved alongside advancements in Machine Learning (ML) architectures and computational techniques. Between the 1950s and 1980s, we had the foundation of ML algorithms, including perceptrons (Weerts *et al.,* 2020) and early decision tree models, which relied on manually defined hyperparameters with limited computational resources. However,the models were small, and tuning was primarily heuristic-based. With the resurgence of neural networks in the 1990s, backpropagation became a critical development, requiring careful selection of learning rates and weight initialisation strategies to avoid vanishing gradients(Halbouni *et al.,* 2022). Gradient descent-based optimisation methods like stochastic gradient descent (SGD) were manually tuned using fixed learning rates and batch sizes.However, the early 2000s saw the introduction of hyperparameter search techniques, moving beyond manual choice. Grid search emerged as a systematic approach, iterating over predefined hyperparameter values (Belete and Huchaiah, 2022). Random search followed, offering better efficiency by sampling from a hyperparameter space rather than exhaustively searching every combination (Arafat *et al.,* 2024). In furtherance, Bayesian optimisation improved search efficiency by modelling the objective function and selecting promising hyperparameter values iteratively (Gao *et al.,* 2022). During this period, adaptive optimisers such as AdaGrad, RMSprop, and Adam optimiserrevolutionised tuning by dynamically adjusting learning rates, improving convergence and generalization (Ng and Ghahfarokhi, 2022). Between2015 and 2020,when CNNs gained prominence in image recognition and complex pattern learning, hyperparameter tuning focused on depth, kernel size, activation functions, and dropout rates (Ennibras *et al.,* 2024). ResNet50, for instance, introduced skip connections to mitigate vanishing gradients, reducing the need for excessive hyperparameter tuning,which further enhanced feature reuse with densely connected layers, DenseNet121, requiring adjustments in learning rate schedules and batch sizes. Recently, advancements have used automated hyperparameter tuning through Neural Architecture Search (NAS) and AutoML frameworks. Google's NASNet and EfficientNet demonstrated that Artificial Intelligence (AI)-driven hyperparameter search could optimise CNN architectures beyond human-designed models (Ahmed *et al.,* 2024; El-Maksoud *et al.,* 2021). Transformer-based models, such as Vision Transformers (ViTs), introduced novel tuning challenges, including sequence length and attention head optimization (Maurício *et al.,* 2023).The trend towards higher-frequency neural networks, larger datasets, and self-optimising ML architectures suggests an increasing reliance on deep learning-based hyperparameter tuning, meta-learning, and federated optimisation strategies (Ethier and Châteauvert, 2024). However, there is a significant research gap in understanding the specific impact of hyperparameter tuning on the architectural structure of complex CNN models, particularly hybridised architectures that integrate multiple CNN variants. Existing studies have largely focused on either individual CNN architectures or general tuning techniques without a comprehensive analysis of how hyperparameter adjustments influence hybrid models' learning efficiency, convergence rate, and generalisation performance.This study, therefore, bridges this gap by systematically analysing the effects of hyperparameter tuning on a hybridised CNN architecture combining DenseNet121 and ResNet50. The key contributions of this research are as follows:

- It examines the impact of key hyperparameters, including epoch size, batch size, learning rate, activation functions, and the number of hidden layers, on hybridised CNN models.
- It optimises strategies for hybridised CNNs through the study of optimised hyperparameter configurations that enhance convergence speed, reduce overfitting, and improve pathloss prediction accuracy.
- Benchmarking of the hybridised CNN-based pathloss model's performance. This researchsets upempirical insights into the effectiveness of hyperparameter tuning for complex deep learning frameworks.

METHODOLOGY

The systematic approach adopted for the research in the hyperparameters impact analyses on a hybridised CNN-based pathloss prediction model through deep learning techniques encompasses data collection, preprocessing, training, testing, and evaluation of the developed hybridised CNN-based pathloss model.

The development of the hybridised model

The model was developed using the libraries and tools for ML, such as scikit-learn and TensorFlow, which are open-source tools on Jupiter Notebook on Google Colaboratory. The architectural structural layers of DenseNet121 and ResNet50 variants of CNN were hybridised through layer compilation and compression as depicted in the flow chart of Figure 1.

Data acquisition

The following procedures were taken for the field measurement of the input data through a drive test.

i. Four measurement routes of GRA – Unilorin, Taiwo – Airport – Otte, Post Office - ARMTI, and Emir Palace - KWASU routes were chatted to cover urban and suburban areas of Ilorin, where all the mobile Base Transceiver Station (BTS) of MTN, Glo, Airtel, and 9Mobile are actively present.

ii. A test drive was done to study human activities and vehicle movement along each route, and an average speed of 40km/hr. It was adopted for the drive test to mitigate the effect of Doppler on the field measurement data.

iii. RantCell Professional version was installed on anInfinix Hot30 mobile phone with an interface as shown in Figure 2. The Global Positioning System for the mobile phone was enabled, and each mobile network's Serial Identification Module (SIM) was sequentially enabled for the drive test along each of the four measurement routes

iv. Mobile signal pathloss values, altitude, latitude, longitude, and satellite images of the measurement points were measured and captured.

Data Preprocessing

The field measurement data were preprocessed using a Microsoft Excel sheet, where outliers in the field data were sorted and filtered, and the data were divided into training data, testing data, and evaluation data. The spherical law of Cosine model, expressed by (Van-Brummelen, 2013) in Equation 1, was used to determine the Radial Distance $R_{Distance}$ of each measurement point from the reference BTS, with information in Table 1 for each route.

$$R_{Distance}(km) = Acos\left((cos(\emptyset_1)cos(\emptyset_2) + sin(\emptyset_1)sin(\Delta\lambda) * cosE_R)\right) \qquad 1$$

Where $\emptyset_1$ and $\emptyset_2$ are the latitude and longitude of the reference BTS, $\Delta\lambda$ is the change in longitude of each measurement point, and E_R is the Earth's radius in km. This model delivers high-precision results with 15 significant figures, enabling accurate calculations for distances as small as approximately 1 meter.

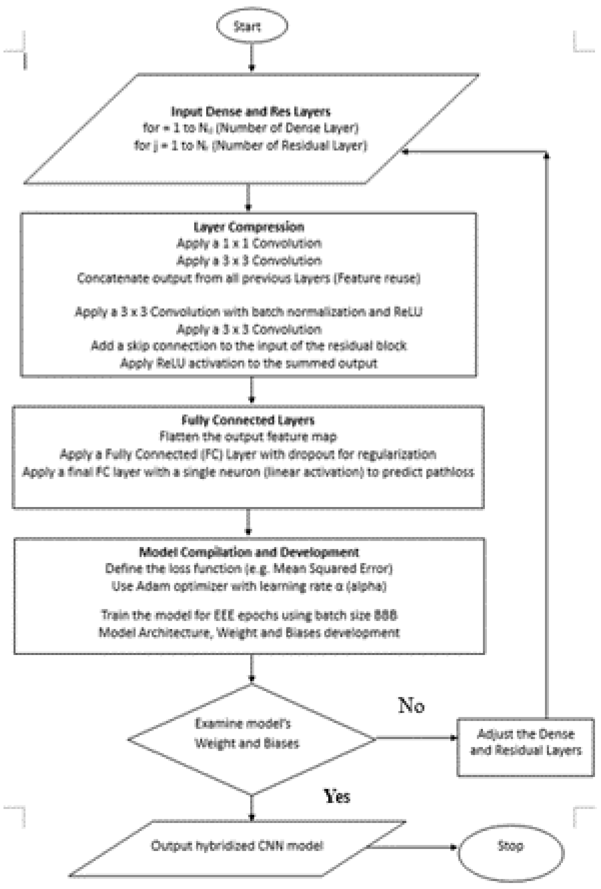
Start
Input Dense and Res Layers
for = 1 to N₁ (Number of Dense Layer)
for j = 1 to N₂ (Number of Residual Layer)
Layer Compression
Apply a 1 x 1 Convolution
Apply a 3 x 3 Convolution
Concatenate output from all previous Layers (Feature reuse)
Apply a 3 x 3 Convolution with batch normalization and ReLU
Apply a 3 x 3 Convolution
Add a skip connection to the input of the residual block
Apply ReLU activation to the summed output
Fully Connected Layers
Flatten the output feature map
Apply a Fully Connected (FC) Layer with dropout for regularization
Apply a final FC layer with a single neuron (linear activation) to predict pathloss
Model Compilation and Development
Define the loss function (e.g. Mean Squared Error)
Use Adam optimizer with learning rate α (alpha)
Train the model for EEE epochs using batch size BBB
Model Architecture, Weight and Biases development
Examine model's Weight and Biases
No
Adjust the Dense and Residual Layers
Yes
Output hybridized CNN model
Stop

Figure 1: The Flow Chart of the hybridised CNN-based pathloss Model's Architectural Structure.

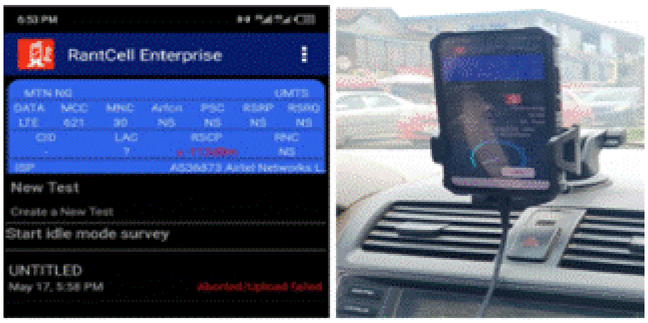

Figure 2: The RantCell Logger Interface and Field Measurement setup.

Table 1: The summary of Reference BTS for the four measurement routes

Routes	Altitude	Latitude	Longitude	RSRP(dB)	Net-Data/Type
GRA-Unilorin	299	8.48513	4.56636	-74	LTE/MTN
Post-office-Armti	340	8.45223	4.58196	-80	EDGE/MTN
Taiwo-Otte	284	8.49952	4.56169	-50	HSPA+/MTN
Emir-Kwasu	248	8.49379	4.56680	-75	HSPA+/MTN

Routes	Altitude	Latitude	Longitude	RSRP(dB)	Net- Data/Type
GRA-Unilorin	302	8.49997	4.57865	-64	LTE/GLO
Post-office-Armti	334	8.45302	4.58098	-61	LTE/GLO
Taiwo-Otte	340	8.44737	4.50745	-91	LTE/GLO
Emir-Kwasu	278	8.49361	4.56689	-73	LTE/GLO

Routes	Altitude	Latitude	Longitude	RSRP(dB)	Net-Data/Type
GRA-Unilorin	287	8.48887	4.56586	-69	LTE/AIRTEL
Post-office-Armti	288	8.49901	4.57336	-75	LTE/AIRTEL
Taiwo-Otte	294	8.43722	4.49921	-75	LTE/AIRTEL
Emir-Kwasu	323	8.43732	4.49925	-76	LTE/AIRTEL

Routes	Altitude	Latitude	Longitude	RSRP (dB)	Net-Data/Type
GRA-Unilorin	286	8.48524	4.56606	-76	LTE/9MOBILE
Post-office-Armti	330	8.45316	4.56606	-86	LTE/9MOBILE
Taiwo-Otte	290	8.48381	4.55826	-87	LTE/9MOBILE
Emir-Kwasu	330	8.45316	4.58111	-86	LTE/9MOBILE

Training and testing of the hybridised CNN-based pathloss models

The preprocessed data were separated into 70% training data, 15% testing data, and evaluation data of 15%, to train the hybridisedCNN-basedpathloss architecture developed using the Python code of Figure 3.

Performance metrics

The performance metrics used in the gauging of the hybridised convolutional neural network to study the hyperparameters' impact on pathloss modeling for mobile communication systems are mean error, absolute mean error, Absolute error, mean squared error, and R-squared coefficient of determination.

Mean error $(\mathbf{M_{error}})$

This is the average of the errors spread between the measured pathloss values and the predicted path loss values. It helps quantify the average amount of errors in a set of predicted pathloss values (McGrath *et al.*, 2020) and it is as expressed in Equation 2.

$$\text{Mean Error}(\mathrm{M_{error}}) = \frac{1}{n} \times \sum_{v=0}^{n}(\mathrm{M_v} - \mathrm{P_v}) \qquad 2$$

where $M_v$and $P_v$present the measured and predicted pathloss values.

Absolute mean error $(\mathbf{A_{m\text{-}error}})$

This is the measure of the average magnitude of the errors between the predicted path loss values and the measured path loss values. It is similar to the mean error, but it takes the absolute value of the differences, making it always positive (Fabián *et al.*, 2021).

The mathematical expression is as expressed in equation 3

$$(\mathrm{A_{m\text{-}error}}) = \frac{1}{n} \times |\textstyle\sum_{v=0}^{n}(\mathrm{M_v} - \mathrm{P_v})| \qquad 3$$

where n is the number of occurrences, $Mv$and Pv are as defined in equation 2.

Mean absolute error (MAE)

This measures the average magnitude of errors in path loss predictions, without considering their direction. It gives an overview of how much the predicted path loss values deviate from the actual measured path loss values (Patro and Ranjan, 2014).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\lceil y_i - \hat{y}_i \rceil \qquad 4$$

Where: n is the number of data points, y_i is the actual path loss measured value, $\hat{y}_i$ is the predicted path loss value, and $\lceil y_i - \hat{y}_i \rceil$represents the absolute error for each path loss predicted. The lower MAE values indicate better model performance. It's easy to interpret as it reflects the average error in the same units as the output variable.

Mean squared error (MSE)

This computes the average squared difference between predicted and actual values. It penalises larger errors more heavily than MAE due to squaring (Heydarian*et al.*, 2022)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad 5$$

where: n, y_i, and $\hat{y}_i$ are as defined in equation 4. $(y_i - \hat{y}_i)^2$ represents the squared error for each predicted path loss value. The Lower MSE values indicate better predictions. However, it can amplify the impact of outliers because errors are squared.

```
# Import the necessary modules
from tensorflow.keras.applications import DenseNet121, ResNet50
from tensorflow.keres import layers, Model

# Build the hybrid CNN model
def build_hybrid_cnn ():

# Load pre-trained DenseNet and ResNet models
densenet = DenseNet121 (weights=:'imagenet', include_top=False, input_shape=(224, 224, 3))
resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

#Freeze the base models
densenet.trainable = False
resnet.trainable = False

#Image input layer
Input_image = layers.Input(shape=(224, 224, 3)

#DenseNet and ResNet feature extraction
densenet_features = densenet (input_image)
resnet_features = densenet = resnet (input_image)

#Concatenate the features
Combined_features = layers.Concatenate() ([
layers.GlobalAveragePooling2D()(densenet_features),
layers.GlobalAveragePooling2D()(resnet_features),
])
```

Figure 3: The Snapshot of the Training and Testing of the hybridised CNN-based pathloss model on Google Colaboratory.

R-squared – coefficient of determination (R^2) measures how well the model explains the variability of the dependent variable. It ranges from 0 to 1, where 1 indicates a perfect fit, 0 indicates a poor fit (Deng *et al.*, 2016; Xu *et al.*, 2020).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \qquad 6$$

Where: n, y_i, and $\hat{y}_i$ are as defined in equation 4, $\bar{y}$ is the mean of the actual path loss measured value, $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ is the residual sum of squares, and $\sum_{i=1}^{n}(y_i - \bar{y})^2$ is the total sum of squares. The values range between 0 and 1. Where 1 indicates a perfect fit, and 0 indicates a poor fit.

RESULTS AND DISCUSSIONS

The results depicted in Figure 4 show the relationship between the number of Hidden Layers and three key performance metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2). At 2 hidden layers, the model performs poorly, with high MAE (25.15), high MSE (34.43), and a highly negative R^2 value (-6.01), indicating that the model is significantly underperforming compared to a baseline prediction. As the number of hidden layers increases to 10 and 12, there is a gradual performance improvement. The MAE and MSE values decrease, and the R^2 value becomes less negative (-5.3 to -0.1).

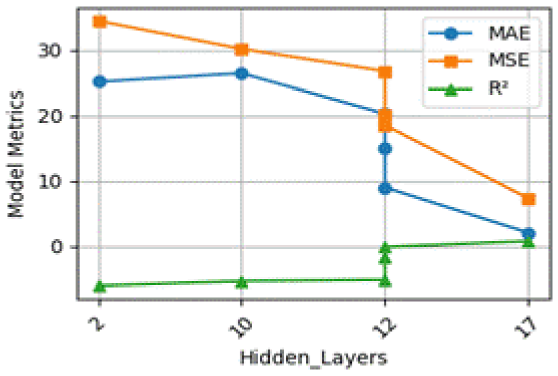

Figure 4: The Graphical depiction of the HiddenLayer's tuning with reference to the Model Evaluation Metrics.

This suggests that adding more hidden layers improves the model's ability to learn patterns, reducing prediction errors.With 17 hidden layers, the model achieves its best performance, with MAE dropping to 2.08 and MSE reducing to 7.35. Additionally, the R² value turns positive (0.8), indicating that the model now explains a significant portion of the variance in the data.

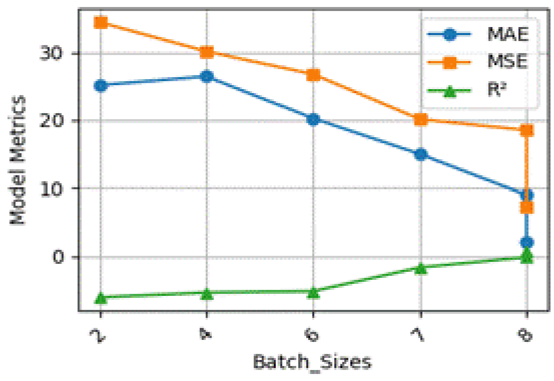

Figure 5: The Graphical depiction of the Batch_Size's tuning with reference to the Model Evaluation Metrics

The graphical depiction of Figure 5 shows the relationship between the Batch Size and three key performance metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R²).At Batch Size = 2, the model exhibits poor performance, with a high MAE (25.15), high MSE (34.43), and a highly negative R² value (-6.01). This indicates that the model performs significantly worse than a simple baseline prediction, suggesting inadequate learning due to the small batch size.As the Batch Size increases from 4 to 7, the MAE and MSE values decrease, and the R² value becomes less negative, improving from -5.3 to -1.6. This trend suggests that increasing the batch size enhances the model's ability to generalize and reduce errors.At Batch Size = 8, the model's performance improves significantly. The MAE drops to 9.021, the MSE reduces to 18.54, and the R² value approaches zero (-0.1), indicating that the model is becoming more reliable. When the batch size remains at 8, but additional factors (hidden layers) likely influence training, the MAE drops further to 2.08, the MSE decreases to 7.35, and the R² value turns positive (0.8), suggesting an optimal configuration for accurate predictions.

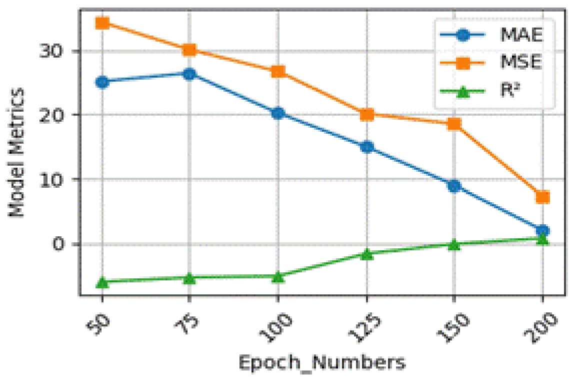

Figure 6: The Graphical depiction of the Number of epochs tuning with reference to the Model Evaluation Metrics

Figure 6 illustrates the relationship between the number of training epochs and the model's performance. At 50 epochs, the model performs poorly, with high MAE (25.15), high MSE (34.43), and a highly negative R² value (-6.01). This suggests that the model is underfitting the data, failing to learn meaningful patterns due to insufficient training.As the number of epochs increases from 75 to 125, the model shows progressive improvement, with MAE and MSE decreasing, and the R² value moving closer to zero (-5.3 to -1.6). This indicates that more training allows the model to refine its predictions and capture underlying data patterns better.At 150 epochs, the MAE and MSE further decrease to 9.021 and 18.54, respectively, and the R² value approaches -0.1, suggesting that the model is nearing optimal training.At 200 epochs, the model achieves its best performance, with MAE dropping to 2.08, MSE reducing to 7.35, and R² becoming positive (0.8). This means the model now explains a significant portion of the variance in the data, indicating a strong predictive capability.

In Figure 7, the results show that the Training Time Per Step remains constant at 549 milliseconds from epoch 50 to 150, despite an increase in Batch Size from 2 to 8 and variations in the number of Hidden Layers. This suggests that increasing the batch size and hidden layers within this range did not significantly affect the training time.

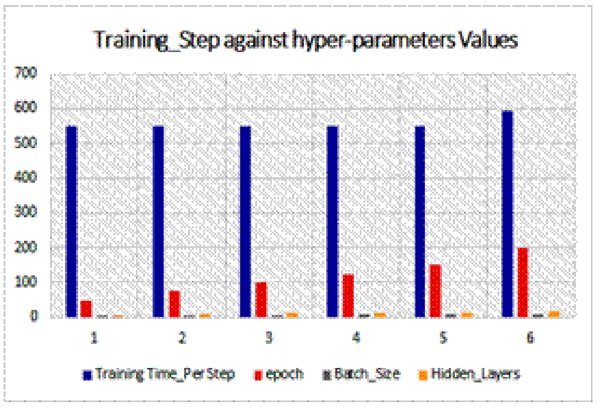

Figure 7: The Graphical depiction of the Training Step against hyperparameters Values

However, at epoch 200, the Training Time Per Step increases to 593 milliseconds. This increase coincides with a rise in Hidden Layers from 12 to 17, while theBatch Size remains constant at 8. This suggests that the increase in model complexity due to additional hidden layers led to a higher computational cost, resulting in a longer training time per step.

CONCLUSION AND RECOMMENDATIONS

The optimal model configuration is achieved with a specific combination of hidden layers, batch size, and training epochs, resulting in improved accuracy and generalisation. Increasing the number of hidden layers enhances the model's performance, with a particularly notable improvement observed within a certain range. A smaller batch size proves effective, balancing error reduction with computational efficiency. The model's accuracy continues to improve with more training epochs, reaching its best performance after enough iterations. While moderate increases in batch size and hidden layers do not substantially impact training time, excessive additions beyond a certain threshold lead to increased computational demands. Therefore, when designing neural networks for similar tasks, it is crucial to focus on optimising these parameters: hidden layers, batch size, and training epochs to achieve the best performance. Ultimately, there must be a careful balance between enhancing model complexity and managing computational cost, as deeper networks can yield better results but may also require more training resources. Future research directions should investigate other hyperparameters, such as learning rate and activation functions, to further improve the model's performance.

REFERENCES

Abdollahzadeh, B., Khodadadi, N., Barshandeh, S., Trojovský, P., Gharehchopogh, F. S., El-kenawy, E.-S. M., Abualigah, L., and Mirjalili, S. (2024). Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning. Cluster Computing, 27(4), 5235–5283. https://doi.org/10.1007/s10586-023-04221-5

Abdulkarim, A., Faruk, N., Alozie, E., Sowande, Olugbenga. A., Olayinka, I.-F. Y., Usman, A. D., Adewole, K. S., Oloyede, A. A., Chiroma, H., Garba, S., Imoize, A. L., Musa, A., and Taura, L. S. (2022). Application of Machine Learning Algorithms to Path Loss Modeling: A Review. 2022 5th Information Technology for Education and Development (ITED), 1–6. https://doi.org/10.1109/ITED56637.2022.1005 1448

Ahmed, W., Massoud, M., and El-Bouridy, M. (2024). Optimizing Mri-Based Medical Diagnosis: Comparative Analysis of Efficientnet Performance with varying Learning Rates. Journal of the Egyptian Society of Tribology, 21(2), 105–119. https://doi.org/10.21608/jest.2024.278570.108 5

Al Hakim, M. F., and Prasetiyo, B. (2024). CNN-ML Stacking for better Classification of Rice

Leaf Diseases. 2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS), 1–5. https://doi.org/10.1109/AIMS61812.2024.10512454.

Arafat, M. Y., Hossain, M. J., and Alam, M. M. (2024). Machine learning scopes on microgrid predictive maintenance: Potential frameworks, challenges, and prospects. Renewable and Sustainable Energy Reviews, 190, 114088. https://doi.org/10.1016/j.rser.2023.114088

Arnold, C., Biedebach, L., Küpfer, A., and Neunhoeffer, M. (2024). The role of hyperparameters in machine learning models and how to tune them. Political Science Research and Methods, 12(4), 841–848. https://doi.org/10.1017/psrm.2023.61

Belete, D. M., and Huchaiah, M. D. (2022). Grid search in hyperparameter optimization of machine learning models for the prediction of HIV/AIDS test results. International Journal of Computers and Applications.https://www.tandfonline.com/doi/abs/10.1080/1206212X.2021.1974663

Benz, P., Ham, S., Zhang, C., Karjauv, A., and Kweon, I. S. (2021). Adversarial Robustness Comparison of Vision Transformer and MLP-Mixer to CNNs. https://doi.org/10.48550/ARXIV.2110.02797

Christopher, S. S., Thakur, A. K., Hazra, S. K., Sharshir, S. W., Pandey, A. K., Rahman, S., Singh, P., Sunder, L. S., Raj, A. K., Dhivagar, R., and Sathyamurthy, R. (2023). Performance evaluation of an external compound parabolic concentrator integrated with a thermal storage tank for a domestic solar refrigeration system. Environmental Science and Pollution Research, 30(22), 62137–62150. https://doi.org/10.1007/s11356-023-26399-2

Deng, X., Liu, Q., Deng, Y., and Mahadevan, S. (2016). An improved method to construct a basic probability assignment based on the confusion matrix for a classification problem. Information Sciences, 340–341, 250–261.

El-Maksoud, A. J. A., Ebbed, M., Khalil, A. H., and Mostafa, H. (2021). Power Efficient Design of High-Performance Convolutional Neural Networks Hardware Accelerator on FPGA: A Case Study With GoogLeNet. IEEE Access, 9, 151897–151911. https://doi.org/10.1109/Access.2021.3126838

Ennibras, F., Aoula, E.-S., and Bouihi, B. (2024). AI in Preventing Dropout in Distance Higher Education: A Systematic Literature Review. 2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), 1–7. https://doi.org/10.1109/Iraset60544.2024.10548954

Ethier, J., and Châteauvert, M. (2024). Machine Learning-Based Path Loss Modeling With Simplified Features. IEEE Antennas and Wireless Propagation Letters, 23(11), 3997–4001. IEEE Antennas and Wireless Propagation Letters. https://doi.org/10.1109/LAWP.2024.3388253

Fabián, Z. (2021). Mean, Mode, or Median? The Score Mean. Communications in Statistics - Theory and Methodsfor Engineering Application50(10), 2360–2370.

Gao, H., Zhong, S., Zhang, W., Igou, T., Berger, E., Reid, E., Zhao, Y., Lambeth, D., Gan, L., Afolabi, M. A., Tong, Z., Lan, G., and Chen, Y. (2022). Revolutionizing Membrane Design Using Machine Learning-Bayesian

Optimization. Environmental Science & Technology, 56(4), 2572–2581. https://doi.org/10.1021/acs.est.1c04373

Gao, L., Huang, Y., Zhang, X., Liu, Q., and Chen, Z. (2022). Prediction of Prospecting Target Based on ResNet Convolutional Neural Network. Applied Sciences, 12(22), 11433. https://doi.org/10.3390/app122211433

Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., and Ahmad, R. (2022). CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System. IEEE Access, 10, 99837–99849. https://doi.org/10.1109/Access.2022.3206425

Hasan, N., Bao, Y., Shawon, A., and Huang, Y. (2021). DenseNet Convolutional Neural Networks Application for Predicting COVID-19 Using CT Image. SN Computer Science, 2(5), 389. https://doi.org/10.1007/s42979-021-00782-7

Heydarian, M., Doyle, T. E., and Samavi, R. (2022). MLCM: Multi-Label Confusion Matrix. IEEE Access, 10, 19083–19095.

Maurício, J., Domingues, I., and Bernardino, J. (2023). Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review. Applied Sciences, 13(9), 5521. https://doi.org/10.3390/app13095521

McGrath, S., Zhao, X., Steele, R., Thombs, B. D., Benedetti, A., Levis, B., Riehm, K. E., Saadat, N., Levis, A. W., Azar, M., Rice, D. B., Sun, Y., Krishnan, A., He, C., Wu, Y., Bhandari, P. M., Neupane, D., Imran, M. and Zhang, Y. (2020). Estimating the sample mean and standard deviation from commonly reported quantiles in meta-analysis. Statistical Methods in Medical Research, 29(9), 2520–2537.

Ng, C. S. W., and Jahanbani Ghahfarokhi, A. (2022). Adaptive Proxy-based Robust Production Optimization with Multilayer Perceptron. Applied Computing and Geosciences, 16, 100103. https://doi.org/10.1016/j.acags.2022.100103

Oyelaran, O. P., Adeyemo, Z. K., Ojo, S. I., & Ojedokun, I. A. (2024). Empirical Mode Decomposition-Based Amplify and Forward Technique for Cooperative Cognitive Radio System. ABUAD Journal of Engineering Research and Development (AJERD), 7(2), 455–466. https://doi.org/10.53982/ajerd.2024.0702.43-j

Patro, V. M. and Ranjan P. M. (2014). Augmenting Weighted Average with Confusion Matrix to Enhance Classification Accuracy. Transactions on Machine Learning and Artificial Intelligence, 2(4). 34 - 38.

Tushar Saxena and Jadon J. S. (2013). Review on 2G, 3 G, and 4G Radio Network Planning. International Journal of Engineering, Business and Enterprise Application, 6(1), 84–89.

Van Brummelen, G. (2013). Heavenly mathematics: The forgotten art of spherical trigonometry. Princeton University Press.

Weerts, H. J. P., Mueller, A. C., and Vanschoren, J. (2020). Importance of Tuning Hyperparameters of Machine Learning Algorithms (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2007.07588

Xu J., Zhang Y., and Miao D. (2020). Three-way confusion matrix for classification: A measure-driven view. Information Sciences and Technology Systems (IS), 5(7), 772 – 794.