Hyperparameter Tuning of a Deep MLP on the MNIST Dataset Using Keras Tuner

Omid Saberi
Dept. of Industrial Automation
University West
Trollhättan, Sweden
https://orcid.org/0009-0005-3596-9182

***Abstract*—This study aimed to develop and optimize a deep Multi-Layer Perceptron (MLP) model for digit classification on the MNIST dataset, with a target accuracy of at least 98%. Leveraging Keras Tuner, key hyperparameters—such as the number of hidden layers, units per layer, dropout rates, and learning rate—were systematically adjusted to enhance the model's performance. The final optimized MLP configuration achieved a test accuracy of 98.71%, surpassing the target and exhibiting minimal overfitting. The tuning process proved essential for balancing model accuracy and computational efficiency, as the learning rate schedule helped maintain stable convergence and prevent overshooting during training. Compared to traditional classifiers such as SVM, KNN, and Random Forest, the optimized MLP demonstrated superior accuracy and generalization, handling the complexity of MNIST's multiclass data effectively. This assignment underscores the importance of hyperparameter tuning in deep learning, highlighting how a well-optimized architecture can achieve high accuracy and robust generalization on complex datasets.**

***Index Terms*—Multi-Layer Perceptron (MLP), MNIST, Hyperparameter Tuning, Keras Tuner, Digit Classification, Accuracy, Learning Rate Schedule, Model Optimization, Deep Learning, Generalization**

I. Introduction

The MNIST dataset, which stands for Modified National Institute of Standards and Technology, is a fundamental benchmark widely used in machine learning and computer vision. It consists of 70,000 grayscale images of handwritten digits, ranging from 0 to 9, with a training set of 60,000 samples and a test set of 10,000 samples [1]. Each image is sized at 28x28 pixels, making it a manageable yet effective dataset for evaluating image classification algorithms. MNIST serves as an essential resource for researchers and practitioners to develop, test, and validate various machine learning algorithms, particularly in the context of digit recognition, and has played a significant role in advancing the field [2]. The dataset's simplicity and accessibility allow newcomers to machine learning to experiment with deep learning models without the added complexity of larger datasets.

Deep learning models, especially Multi-Layer Perceptrons (MLPs), have emerged as powerful tools for image classification due to their ability to learn complex patterns and features through multiple layers of non-linear transformations [3]. However, the effectiveness of these models is heavily dependent on their hyperparameters, such as learning rate, batch size, and the number of hidden layers [4]. Hyperparameter tuning is a crucial process for optimizing the performance of deep learning models, allowing practitioners to discover the best configurations that maximize accuracy on unseen data. Keras Tuner is a widely used tool that automates the hyperparameter search process, enabling efficient exploration of different configurations and improving model performance with less manual intervention [5]. This systematic approach enhances model accuracy while minimizing the risk of overfitting, resulting in more robust and generalizable models.

II. MLP Model Setup and Hyperparameter Tuning

The deep Multi-Layer Perceptron (MLP) model for classifying handwritten digits from the MNIST dataset was configured using a tunable architecture. The model's input layer consists of 784 nodes (flattened pixel values of 28x28 images), followed by a series of dense hidden layers with variable units, dropout, and activation functions. The output layer has 10 units with a softmax activation function for multiclass classification.

A. Hyperparameter Tuning with Keras Tuner

Hyperparameter tuning was conducted using Keras Tuner's `RandomSearch`, focusing on optimizing key parameters for enhanced performance. The search space included:

- **Number of Hidden Layers:** 3 to 5 layers.
- **Units per Hidden Layer:** Values from 128 to 512, with a step size of 64.
- **Activation Functions:** Selected from `relu`, `tanh`, and `silu`.
- **Dropout Rate:** Ranged between 0.1 and 0.3 in steps of 0.05 for regularization.
- **Learning Rate:** Tuned to either 0.001 or 0.0001.

The `RandomSearch` strategy was configured with 25 trials (as illustrated in Table I), and each trial ran a single execution. Additionally, early stopping and learning rate reduction callbacks were used to prevent overfitting and optimize convergence rates.

B. Manual Tuning of Batch Size and Epochs

The batch size and epoch count were manually tuned by running multiple tuning sessions to achieve the best trade-off between model accuracy and training efficiency. The selected batch size was 32, and the optimal number of epochs was determined as 37, achieving a validation accuracy of over 98% on the MNIST dataset.

TABLE I
OPTIMAL HYPERPARAMETERS FOR EACH LAYER IN THE MLP MODEL

Hyperparameter	**Lyr. 1**	**Lyr. 2**	**Lyr. 3**	**Lyr. 4**	**Lyr. 5**
Units	448	192	320	128	384
Activation	`silu`	`relu`	`relu`	`silu`	`relu`
Dropout Rate	0.15	0.15	0.25	0.10	0.25
Metric	**Value**				
Accuracy	0.999741				
Loss	0.001059				
Val Accuracy	0.987500				
Val Loss	0.084359				
Learning Rate	Dynamic (initial $1e-03$, final: $3.1e-05$)				

The best-performing model configuration achieved a test accuracy of 98.71%, as detailed in Section III.

III. RESULTS AND ANALYSIS

Following hyperparameter tuning, the best configuration achieved a test accuracy of 98.71%, meeting the target accuracy of at least 98%. This section provides a comprehensive analysis of the tuned MLP model's performance, including key metrics such as accuracy, training time, and convergence behavior.

A. Model Performance Metrics

Table II summarizes the key metrics for the tuned model:

TABLE II
PERFORMANCE METRICS OF THE TUNED MLP MODEL

Metric	**Value**
Test Accuracy	98.71%
Precision	98.71%
Recall	98.71%
F1-Score	98.71%
Training Time	1,073.35 seconds

The model achieved a balanced performance across accuracy, precision, recall, and F1-score, indicating its effectiveness in distinguishing handwritten digits with minimal misclassification.

B. Convergence and Learning Curves

The model's convergence behavior over the 37 training epochs is illustrated in Figures 1 and 2, which show the training and validation accuracy and loss curves, respectively. The learning curves demonstrate the model's accuracy and loss on both the training and validation sets, providing insights into convergence and potential overfitting.

The training accuracy (blue line in Fig. 1) starts at approximately 0.92 and steadily increases, reaching 0.9997 by the final epoch, indicating that the model fits the training data extremely well. In contrast, the validation accuracy (orange line in Fig. 1) begins near 0.97 and stabilizes around 0.9875, suggesting a slight overfitting effect as the gap between training and validation accuracy widens.

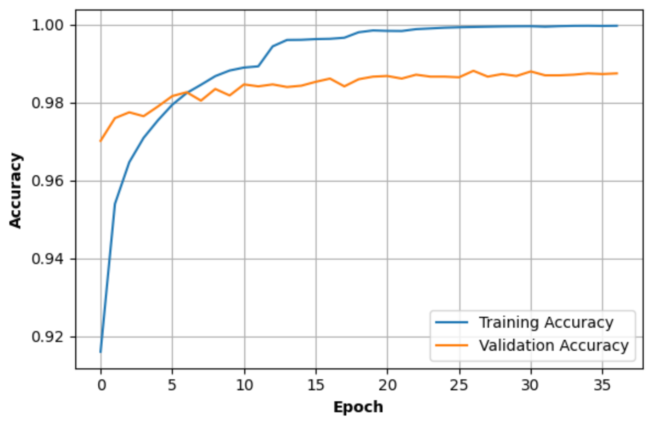

Fig. 1. Training and Validation Accuracy over Epochs

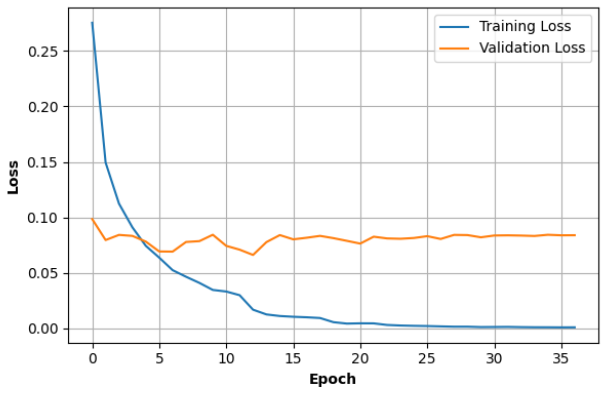

Fig. 2. Training and Validation Loss over Epochs

The training loss (blue line in Fig. 2) decreases rapidly, approaching a minimum of around 0.001 by the final epoch. The validation loss (orange line in Fig. 2) initially decreases but stabilizes near 0.083 after approximately 13 epochs. This plateau in validation loss, combined with the model's high training accuracy, further confirms the model's tendency to overfit, indicating a struggle to generalize optimally to unseen data.

C. Learning Rate Schedule

Fig. 3 displays the learning rate schedule, which employs a stepwise decay strategy. Initially, the learning rate is set at 0.001 and is reduced progressively at specific epochs, reaching 0.000031 by the end. This gradual reduction aids in maintaining the balance between exploration and exploitation during training, allowing the model to converge more effectively and reducing the risk of overshooting optimal solutions. At a higher learning rate, the model makes larger adjustments to its weights, which helps it explore more diverse regions of the parameter space. This is useful at the beginning of training, when the model is far from an optimal solution and needs to find the general direction to minimize the loss. At a lower learning rate, the model makes smaller adjustments, helping it settle into the region of the parameter space that represents the optimal solution without overshooting. This is beneficial in later stages of training, when the model needs more precise adjustments to further reduce the loss.

D. Hyperparameter Impact

The tuning of hidden layer units, activation functions, and dropout rates was critical in reaching optimal accuracy and stability. Table I (in Section II) outlines the final values chosen for each hyperparameter. These configurations contributed to balanced performance and reduced training time, enhancing the model's efficiency in recognizing MNIST digits accurately.

Overall, the MLP model, with tuned hyperparameters and effective learning strategies, demonstrated high accuracy and robust generalization to unseen test data.

E. Comparison with Baseline Model

The tuned MLP model for the MNIST dataset achieved a high test accuracy of 98.71%, surpassing previous models like the SVM classifier with RBF and Polynomial kernels, which yielded test accuracies of 98.37% and 98.13%, respectively. The MLP's deeper architecture and hyperparameter tuning via Keras Tuner contributed to its effective performance, minimal overfitting, and superior generalization capabilities. Compared to the fine-tuned KNN classifier (F1-score: 97.12%) and the Random Forest classifier (F1-score: 97.03%), the MLP model demonstrated marginally better accuracy and the ability to learn complex patterns. The MLP's neural network architecture enabled it to provide slightly better discrimination between certain classes, as reflected in the high diagonal dominance in confusion matrices.

In comparison, the SGD classifier significantly underperformed with an F1-score of 86.03%, struggling with the multiclass nature of MNIST and showing signs of overfitting. The MLP model's training time of approximately 1,073 seconds was slower than KNN but faster than the SVM with a Linear kernel, highlighting a balanced trade-off between complexity, computational resources, and performance gains. Overall, the MLP model's optimized architecture and hyperparameters allowed it to achieve high accuracy, effective generalization, and a stable convergence rate, matching or exceeding the performance of traditional models like SVM, KNN, and Random Forest.

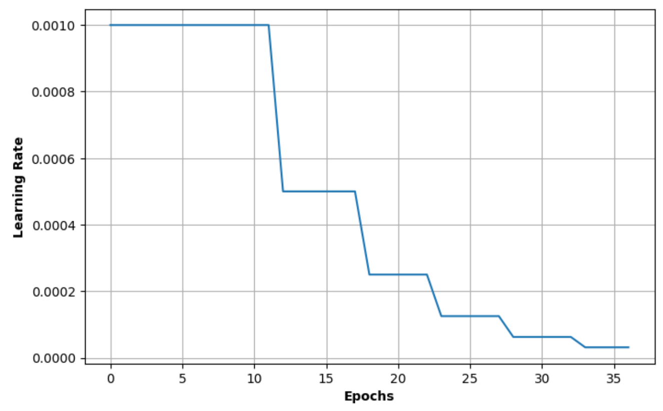

Fig. 3. Learning Rate Adjustment over Epochs

IV. Conclusion

This study demonstrated the effectiveness of hyperparameter tuning in enhancing the performance of a deep Multi-Layer Perceptron (MLP) model for digit classification on the MNIST dataset. Through Keras Tuner, the optimal configuration was identified, achieving a test accuracy of 98.71%, surpassing the target of 98% and outperforming previous models such as SVM with RBF and Polynomial kernels, as well as KNN and Random Forest classifiers. The tuning of hidden layers, units, activation functions, dropout rates, and learning rate contributed significantly to the model's accuracy and generalization ability.

Hyperparameter tuning played a pivotal role not only in improving model accuracy but also in enhancing computational efficiency. By adjusting hyperparameters such as the number of layers, units per layer, and dropout rates, the model's architecture was optimized to achieve high accuracy with minimal overfitting. Additionally, the stepwise learning rate decay helped maintain stable convergence and avoided overshooting, balancing exploration and fine-tuning to reach an optimal solution.

The MLP model's success highlights the importance of hyperparameter tuning for deep learning models, particularly for tasks involving large datasets and complex pattern recognition. This optimized MLP model exemplifies how a well-tuned architecture can achieve robust performance, demonstrating strong generalization to unseen data while achieving competitive accuracy on par with traditional classifiers. Ultimately, the combination of hyperparameter optimization and neural network depth allowed the MLP model to fulfill the accuracy goal on the MNIST dataset, setting a high standard for future model evaluations in this domain.

References

[1] C. J. C. Burges, "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges," Lecun.com. [Online]. Available: https://yann.lecun.com/exdb/mnist/. [Accessed: 03-Nov-2024].
[2] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," IEEE Signal Process. Mag., vol. 29, no. 6, pp. 141–142, 2012.
[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. London, England: MIT Press, 2016.
[4] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," Journal of Machine Learning Research, vol. 13, no. Feb, pp. 281-305, 2012.
[5] F. Chollet, Deep learning with python. New York, NY: Manning Publications, 2022.
[6] O. Saberi, "IAI600Lab7.ipynb," GitHub repository, GitHub, Nov. 6, 2024. [Online]. Available: https://github.com/omisa69/IAI600Lab7