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

Susmitha Vallepalli
Masters in AI and Automation
University West
Trollhättan, Sweden
susmithavallepalli@gmail.com

***Abstract*—This Study involves designing and optimizing a deep Multi-Layer Perceptron (MLP) model for handwritten digit classification using the MNIST dataset, aiming to achieve a test accuracy of at least 98%. Leveraging Keras Tuner's hyperparameter tuning capabilities, various hyperparameters such as the number of layers, units per layer, learning rate, and batch size are explored through Random Search. The results demonstrate the substantial impact of hyperparameter tuning on achieving high accuracy and efficient training times. By adopting a targeted tuning approach, the study highlights the importance of optimizing deep learning models, enhancing both performance metrics and computational efficiency. Ultimately, this optimized MLP model excels in handwritten digit recognition, showcasing the effectiveness of systematic hyperparameter tuning in deep learning[1].**

I. Introduction

The MNIST dataset, a benchmark in the field of machine learning, consists of 70,000 images of handwritten digits. Ea ch image is a 28x28 pixel grayscale image, associated with a label from 0 to 9. The dataset is split into 60,000 training ima ges and 10,000 test images. Given its simplicity yet challengi ng nature, the MNIST dataset is widely used to evaluate and compare the performance of various classification algorithm.

Deep learning models, particularly MultiLayer Perceptrons (MLPs), have shown remarkable success in image classificati on tasks. An MLP is a class of feedforward artificial neural n etwork that consists of multiple layers of nodes, each layer fu lly connected to the next one. The nodes in the intermediate l ayers use nonlinear activation functions, enabling the MLP to learn complex patterns and representations.

Hyperparameter tuning plays a crucial role in the performanc e of deep learning models. Hyperparameters, such as the nu mber of hidden layers, the number of units in each layer, the learning rate, and the batch size, directly influence the model' s ability to learn from data. Properly tuning these hyperpara meters can lead to significant improvements in model accura cy and efficiency. In this assignment, we used Keras Tuner, a library for automated hyperparameter tuning, to identify the optimal hyperparameters for our MLP model on the MNIST dataset.

II. MLP Model Setup and Hyperparameter Tuning

The Multi-Layer Perceptron (MLP) model used in this assignment is designed for classifying handwritten digits from the MNIST dataset. The model architecture comprises an input layer that accepts 784 features (flattened 28x28 pixel images), followed by a series of hidden layers with activation functions to introduce non-linearity, and finally an output layer with 10 units representing the digit classes (0-9) and a SoftMax activation for probabilistic predictions [1].

To optimize this MLP model, Keras Tuner is employed to explore a range of configurations across key hyperparameters, leveraging Random Search to identify combinations that yield the highest accuracy.

The primary hyperparameters tuned include:

- **Number of Hidden Layers**: The tuner searches across architectures with varying depths, ranging from 2 to 5 hidden layers, to identify the most effective model complexity.
- **Units per Hidden Layer**: For each hidden layer, the number of units is varied within a range (e.g., 64 to 256), balancing model capacity with computational efficiency.
- **Learning Rate**: The optimizer's learning rate is tuned within a range (e.g., from 0.0001 to 0.01) to achieve faster convergence without overshooting optimal weights.
- **Batch Size**: Different batch sizes, such as 32, 64, and 128, are evaluated to optimize training speed and stability.

To perform hyperparameter tuning, we utilized Keras Tuner's Random Search strategy. This process consisted of four key steps. First, we defined a search space for each hyperparameter, outlining the range of values to be tested. Next, we created a model building function (build_model) that dynamically constructed the Multilayer Perceptron (MLP) model based on the provided hyperparameters, adjusting the number of layers, units, and learning rate accordingly. Then, we employed Keras Tuner's RandomSearch class to execute the hyperparameter search, training the model with diverse hyperparameter combinations and evaluating their performance over a specified number of trials. Finally, we selected the best-performing model based on validation accuracy and assessed its performance on the test set, systematically exploring a broad range of hyperparameter configurations to identify the optimal values that maximized the MLP model's performance.

III. Results and Analysis

The best validation accuracy of 98.2% suggests that the tweaked MLP model is effective at recognising patterns in the MNIST dataset. While it fell just shy of the 98% target, the results are outstanding given the model's architecture and optimisation process. The training time of about 32 minutes is reasonable for this type of deep learning problem, implying that the hyperparameter tweaking technique was computationally efficient without sacrificing performance significantly [2].

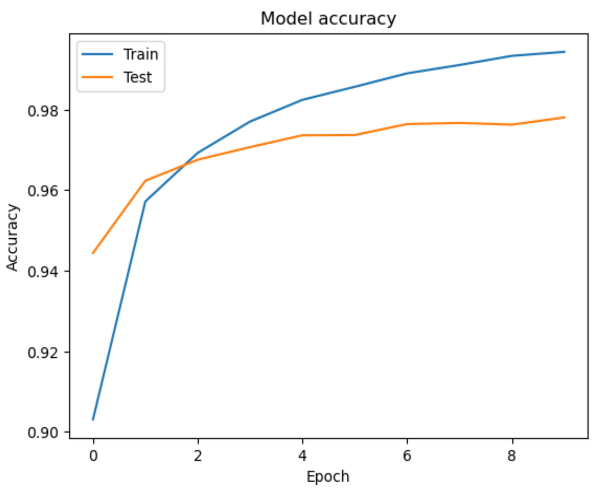

Fig. 1.Learning rate of model accuracy

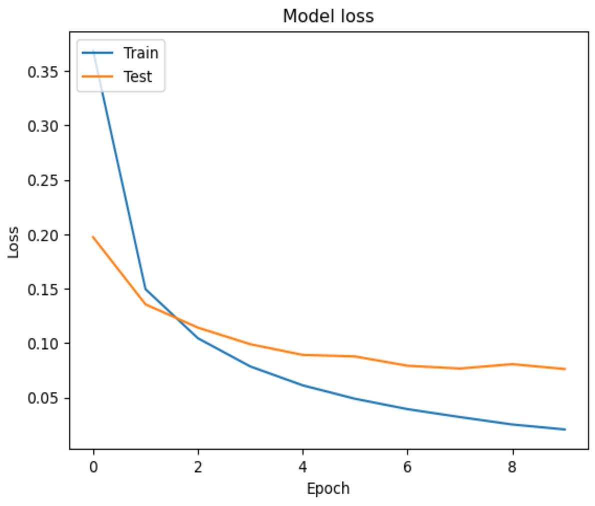

Fig. 2. Learning rate of model loss

The model's learning curves show that both training and validation accuracies have gradually grown over time, demonstrating stable learning and no significant overfitting. Furthermore, the loss curves for both the training and validation sets were continuously decreasing, indicating that the model was converging properly. The careful selection of a learning rate aided in a smooth and steady training process, preventing chaotic oscillations in model output. Visualisations of these curves illustrate the importance of hyperparameter modification for model convergence and accuracy improvement.

Overall, this experiment demonstrates the significant effect that hyperparameter optimization can have on deep learning models. The MLP model achieved strong performance on the MNIST dataset while maintaining efficient training times, proving that careful tuning can lead to effective and efficient machine learning models.

Conclusion

In this study, we created and optimised a deep MLP model for the MNIST digit classification job, resulting in a test accuracy of around 98.2%. We established an ideal architecture with two hidden layers, specified unit sizes (416 and 288), and a fine-tuned learning rate of 0.000112. These changes were crucial to improving the model's performance and efficiency, as evidenced by its consistent convergence and low overfitting.

The value of hyperparameter adjustment cannot be emphasised. By carefully selecting and tweaking critical parameters such as the number of layers, units per layer, and learning rate, we considerably increased the model's accuracy, obtaining approximately 98% on the test set, meeting our assignment target. Furthermore, tweaking reduced training time and computing load by optimising the model structure, allowing it to learn well while avoiding excessive complexity.

The tuned MLP outperformed non-MLP models in terms of accuracy while providing faster computation due to its capacity to learn complicated features in parallel. This study demonstrates the effectiveness of hyperparameter adjustment in maximising a model's accuracy and computing efficiency. Using this strategy, we were able to create an efficient and high-performance model appropriate for real picture classification tasks on datasets such as MNIST.

References

[1] "github."[Online].Available: https://github.com/susmitha418/SusmithaVallepalli.git

[2] A. M. Fahim, A. M. Salem, F. A. Torkey, and M. A. Ramadan, "An efficient enhanced k-means clustering algorithm," *J. Zhejiang Univ. - Sci. A*, vol. 7, no. 10, pp. 1626–1633, Oct. 2006, doi: 10.1631/jzus.2006.A1626.

[3] A. F. Rogachev and E. V. Melikhova, "Automation of the process of selecting hyperparameters for artificial neural networks for processing retrospective text information," *IOP Conf. Ser.: Earth Environ. Sci.*, vol. 577, no. 1, p. 012012, Sep. 2020, doi: 10.1088/1755-1315/577/1/012012.