The Effects of Hyperparameters on Multilayer Perceptron Performance in Digit Classification

Rabbani Nur Kumoro
(21/472599/PA/20310)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
rabbani.nur.kumoro@mail.ugm.ac.id

Abstract — **This study delves into the intricate world of artificial neural networks, particularly focusing on the Multilayer Perceptron architecture, to analyze the profound impact of hyperparameters on the task of digit classification using the MNIST dataset. The investigation encompasses five key hyperparameters: the number of hidden layers, the nodes within the hidden layers, the learning rate, and the weight initialization. Through the experimentation, this analysis sheds light on how these hyperparameters influence the MLP's performance, including convergence speed and generalization capability. The findings from this study offer valuable insights into the critical role these hyperparameters play in optimizing the performance of neural networks for real-world classification tasks.**

Keywords — Artificial Neural Networks, Classification, Digits, Deep Learning, Hyperparameter, MLP

I. Introduction

Neural networks are systems of interconnected nodes, similar in concept to the neural networks found in biological organisms **[1]**. In biological systems, these networks consist of interconnected neurons that communicate through synapses, and they are crucial for the learning process in animals with advanced nervous systems. These networks enable organisms with cognitive abilities to develop learned behaviors that enhance their adaptation to the environment. Taking inspiration from the human brain, artificial neural networks empower computers to tackle cognitive tasks that typically require human expertise **[1]**. Similar to the human brain, machines can be trained to perform specific tasks by analyzing training datasets. At the core of neural networks is the perceptron, which serves as an artificial counterpart to biological neurons. **Figure 1** illustrates a model of a perceptron that calculates an output based on the weighted sum of its inputs. The weights assigned to inputs determine their individual influence on the perceptron's output, and the mathematical function used for this computation is referred to as an activation function.

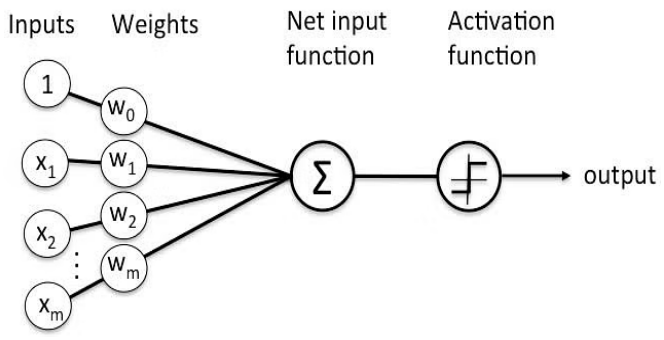

Figure 1. Perceptron

The Multilayer Perceptron (MLP) serves as an illustration of a neural network that comprises layers of fully connected perceptrons. This MLP structure involves organizing perceptrons into layers, as depicted in **Figure 2**. These neural networks undergo training using a technique called backpropagation, which employs gradient descent to minimize errors in the network's output by adjusting weight values **[2]**. Essentially, the algorithm learns from the data to understand a function. When provided with a set of features and a target variable, it acquires the ability to learn a nonlinear function, particularly for tasks such as classification.

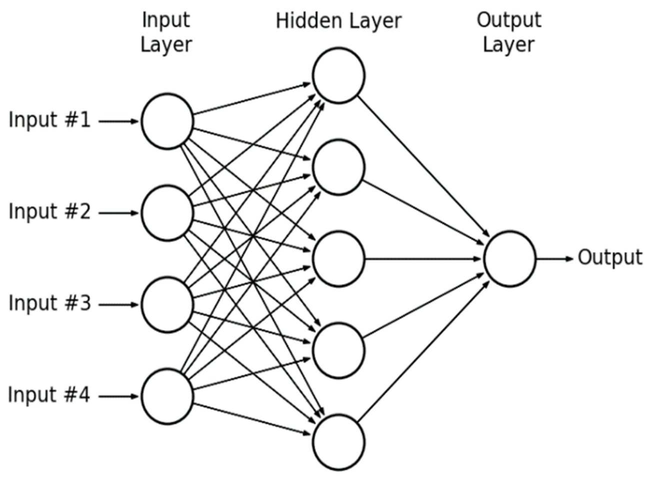

Figure 2. Multilayer Perceptron with 1 Hidden Layer

MLP exhibits its strength by progressively refining the input at each layer **[3]**. Its potency lies in its ability to enhance significant aspects of the input essential for classification while diminishing irrelevant details. This attribute proves highly beneficial in image classification tasks, where the complex task of identifying an object can be broken down into more manageable subtasks like detecting curves and edges. A pivotal aspect and advantage of MLP is that this decomposition isn't carried out manually, instead, it is autonomously learned through the training process **[3]**.

This proposed study aims to harness the potential of artificial neural networks, specifically utilizing MLP, to tackle the intricate task of digit classification in images, particularly on the widely recognized Modified National Institute of Standards and Technology (MNIST) dataset. The study seeks to explore how modifications in hyperparameters impact MLP's performance. By testing and adjusting these hyperparameters, the goal is to unlock the full potential of MLP in enhancing digit classification accuracy, ultimately contributing to the advancement of neural networks and their practical applications.

II. Dataset

MNIST holds a distinguished status as a benchmark dataset for handwritten digit recognition **[4]**. It is commonly featured as an introductory task in numerous deep-learning resources and research studies. It comprises a total of 70,000 images depicting handwritten digits. This extensive dataset is thoughtfully divided into two subsets: 60,000 images for training and 10,000 images for testing. Each image in MNIST possesses a compact size of 28 × 28, resulting in a total of 784 pixels per image, and is presented in grayscale. Furthermore, every image is assigned a label ranging from 0 to 9, signifying the specific digit it represents.

Figure 3. Handwritten Digit Images

This dataset's structured composition and well-defined objectives make it an excellent choice for exploring and evaluating various machine learning and neural network models **[4]**. MNIST stands out as an ideal choice due to its simplicity and widespread recognition in the realm of deep learning. Notably, **Figure 3** offers a glimpse into this dataset, showcasing a selection of images that have become emblematic in the field of computer vision.

III. Methodology

The methodology employed in this study comprises two primary phases: pre-processing and modeling. In the pre-processing phase, specific data enhancements are applied. These include the conversion of raw labels into one-hot vectors and the normalization of pixel values. These enhancements aim to improve the efficiency of the model's training process. In the subsequent modeling phase, an MLP architecture is constructed. Various hyperparameters, such as the number of hidden layers, learning rate, and weight initialization, are experimented with. The primary goal of this modeling phase is to perform Handwritten Digit classification using the MNIST dataset while exploring how these hyperparameters impact the model's performance.

Figure 4 illustrates the workflow of this study, beginning with the acquisition of the MNIST dataset. This dataset undergoes pre-processing procedures and is subsequently divided into training and testing subsets. These subsets serve as the foundation for training and assessing various MLP models. Once trained, the models are employed for classification tasks, successfully identifying the respective 10 digits.

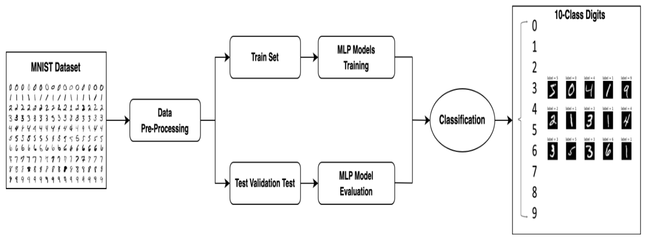

Figure 4. Methodology Flowchart Diagram

A. Pre-Processing

The pre-processing methods that are applied to the images include raw label conversion and normalization. These steps are essential to prepare the data to train the MLP model.

Initially, the dataset is imported directly from the TensorFlow library through Keras. This dataset comprises both a training set consisting of 60,000 examples and 60,000 corresponding labels, as well as a separate test set containing 10,000 examples and 10,000 associated labels. Each example within this dataset is represented as a 28 × 28 matrix, where each element of this matrix corresponds to a specific grayscale value within the 28 × 28 image.

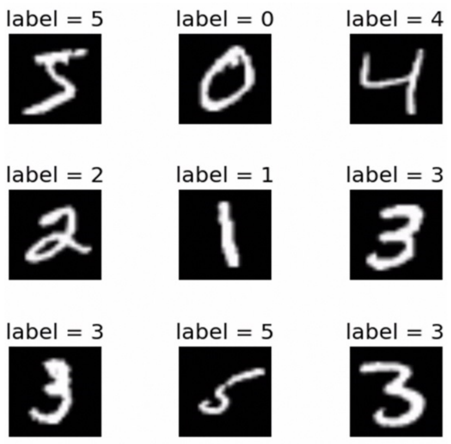

Figure 5. MNIST Labels

The dataset contains images in their raw grayscale format, with pixel values ranging from 0 to 255. Additionally, the labels within this dataset are represented in their raw digit form, ranging from 0 to 9. ignifying the handwritten digit depicted in the respective example, as illustrated in **Figure 5**. These raw formats are not conducive to training the MLP models effectively, necessitating pre-processing to transform both the input and output data.

The first pre-processing step involves converting the raw labels, originally represented as single digits (e.g., 0, 1, ..., 9), into a one-hot vector format. This transformation is necessary because it aligns the label data with the requirements of deep learning models, especially in tasks like classification. By converting the labels into one-hot vectors, the model can better understand and predict class probabilities, enhancing its ability to handle the classification task effectively.

Finally, the pre-processing step focuses on normalizing the pixel values within the images. Initially, the pixel values span the entire range from 0 to 255, which can introduce unnecessary variations during training. Normalization adjusts these pixel values to a standardized range of [0.0, 1.0]. This process ensures that the model's training proceeds smoothly, without encountering extreme weight and bias fluctuations, which can impede the learning process.

B. Modeling

MLP operates as a supervised artificial neural network learning model, following a feedforward structure to map a given set of input vectors onto a corresponding set of output vectors. It stands as a foundational architecture with the potential to tackle complex tasks. Unlike its single-layer perceptron counterpart, which can only address linearly separable problems, MLP leverages the power of multiple hidden layers to confront intricate challenges **[2]**.

Conceptually, the MLP can be visualized as a directed graph comprising multiple layers of nodes, with each layer fully interconnected to the subsequent one. Within the MLP model, it consists of distinct layers, beginning with an input layer, followed by one or more hidden layers, and concluding with an output layer. Notably, all nodes, with the exception of those in the input layer, function as neurons equipped with both summation and activation functions. The activation function, as denoted by **Equation (1)**, represents a non-linear operation that transforms the result of the summation function ($\boldsymbol{xw} + b$) into the output value $\boldsymbol{y}$ **[5]**. Here, the variables $\boldsymbol{x}$, $\boldsymbol{w}$, b, and $\boldsymbol{y}$ correspond to the input vector, weight vector, bias, and output value, respectively.

$$\boldsymbol{y} = \varphi(\boldsymbol{xw} + b) \qquad \textbf{(1)}$$

The flow of information within this network follows a unidirectional path, moving from the input layer to the output layer, fostering a hierarchical representation of data. The input layer faithfully reflects the dimensions of the problem space, with the number of neurons aligning with the input variables. In contrast, the output layer captures the network's ultimate predictions, its neuron count mirroring the number of distinct classes in the classification task at hand.

Within this study, two distinct MLP models will be investigated with each model having 2 and 3 hidden layers respectively. As a default configuration, both models will incorporate the Rectified Linear Unit (ReLU) activation function in their input and hidden layers, while employing the softmax activation function in their output layers to normalize outputs into a probability distribution across predicted output classes **[6]**. These models will form the foundation for the exploration of various hyperparameters, encompassing adjustments to the number of hidden layers, learning rates, and weight initialization techniques. This exploration aims to shed light on how these factors influence the performance of the models in the task of handwritten digit classification.

The initial model employed in this study is configured with 2 hidden layers, visualized in **Figure 6**. Each of these hidden layers is characterized by a distinct set of neurons, contributing to the creation of a distinctive neural architecture. In particular, the study investigates multiple variations in the number of nodes allocated to these two hidden layers, specifically considering configurations such as 256/256, 256/128, and 128/64.

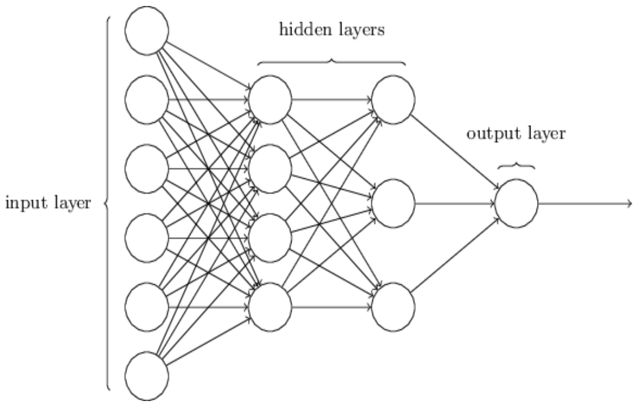

Figure 6. MLP Architecture with 2 Hidden Layers

In contrast, the second model within this study adopts a more intricate architecture, featuring a total of 3 hidden layers, as depicted in **Figure 7**. These layers are structured with varying configurations, showcasing the adaptability and flexibility of the MLP model. Specifically, the configurations explored encompass node allocations of 256/128/64 and 128/64/32 across the first, second, and third hidden layers, respectively. It's important to note that this multifaceted approach leads to the development of a total of 30 distinct MLP models, each distinguished by variations in hyperparameters, including learning rate and weight initialization. These models collectively form a comprehensive experimental framework aimed at assessing the influence of these hyperparameters on the performance of the multilayer perceptron (MLP) model.

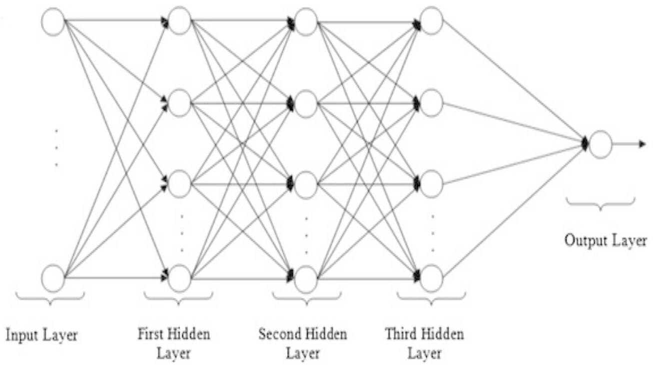

Figure 7. MLP Architecture with 3 Hidden Layers

Table 1 presents the hyperparameters under scrutiny in this study. These hyperparameters constitute the framework for comprehensive analysis in the experimentation, shaping the MLP model's performance in digit classification.

Table 1. Hyperparameter Settings

Hyperparameters	**Values**
Hidden Layer	[2, 3]
Number of Nodes near Input Layer	[256, 128]
Number of Nodes near Output Layer	[256, 128, 64, 32]
Learning Rate	[0.1, 0.01, 0.001]
Weight Initialization	[He, Random]

IV. Results and Analysis

This section presents the performance and hyperparameter analysis of the proposed MLP models. The model was performed in a rigorous experiment while checking all hyperparameters. It was experimented extensively and tested with a constant epoch = 20 for all of them. Firstly, an overview of the parameters used for validation and evaluation. The accuracy and loss functions for both the train and test samples are plotted for validation, while performance evaluation measures like accuracy are used for the evaluation of the testing validation data. These measures offer a detailed assessment of how effective the model is in accurately classifying handwritten digits.

While this study will not present all 30 models individually due to space constraints, several representative models will be showcased to illustrate the impact of varying hyperparameters on the MLP's performance. These selected models provide insights into the relationship between different parameter configurations and the model's accuracy and loss. However, for those interested in a more comprehensive overview of all 30 models and their corresponding results, the complete source code is available in the appendix.

A. Model A

The initial preset model, denoted as Model A, follows the configuration outlined in **Table 2**. These 6 models feature 2 hidden layers, each comprising 256 nodes in proximity to both the input and output layers.

Table 2. Model A Comparison with 2 Hidden Layers

Learning Rate	**Weight Initialization**	**Test Loss**	**Test Accuracy**
0.1	Random	0.067	0.981
0.01	Random	0.137	0.960
0.001	Random	0.428	0.882
0.1	He	**0.063**	**0.981**
0.01	He	0.106	0.968
0.001	He	0.292	0.919

Within the preset of Model A, consisting of 2 hidden layers, results of different hyperparameters reveals crucial insights. Notably, in the experimentation with different learning rates, specifically 0.1, 0.01, and 0.001. It can be seen that the best optimum learning rate is 0.1, giving the least loss and maximum accuracy. This finding implies that increased learning rates lead to improved performance metrics and it underscores the significance of selecting an appropriate learning rate, as it significantly influences the model's training and convergence speed **[7]**.

Furthermore, the investigation into weight initialization methods highlights the superiority of 'He' weight initialization over 'Random' weight initialization. 'He' initializes weights in a way that's better suited for deep networks, leading to improved convergence and accuracy during training. In contrast, 'Random' weight initialization may lead to slower convergence and less effective training, as it initializes weights randomly without considering network depth and architecture. The impact of weight initialization on model performance is particularly evident, underscoring the importance of this hyperparameter in effectively training an MLP for digit classification tasks **[8]**.

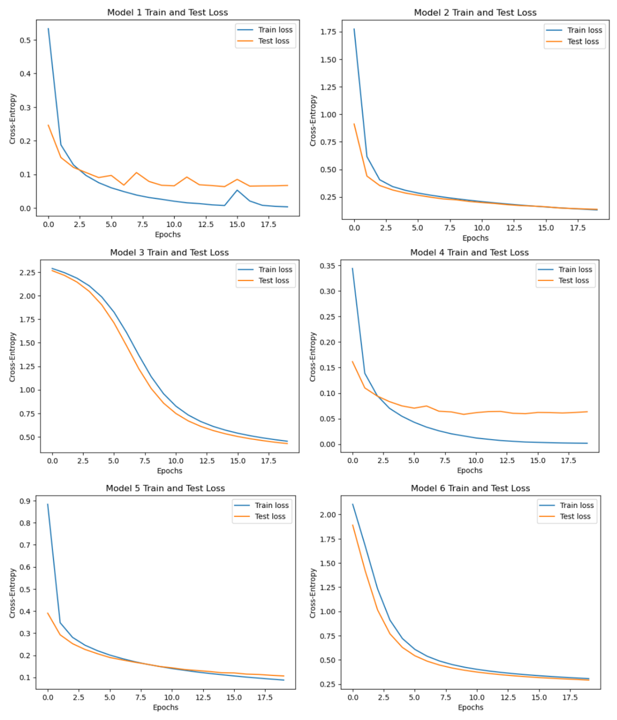

Figure 8. Model A Results

While 'He' weight initialization generally outperforms 'Random' weight initialization, it's important to note that not all instances exhibit the same behavior. In the case of Model 4, despite achieving the best accuracy and test loss scores, a slight overfitting pattern is observed. However, when considering the overall performance across various models, He Weight Initialization remains the preferred choice, as depicted in **Figure 8** and **Table 2**, due to its consistent advantages in convergence, loss functions, and accuracy.

B. Model B

The second preset in the experiment, Model B, adheres to the specifications outlined in **Table 3**. Selected from a pool of 30 models, these 5 models were particularly chosen based on the previous discovery that 'He' weight initialization in tandem with a 0.1 learning rate yields optimal results.

With the primary aim of assessing the influence of hidden layers and their respective node counts on overall model performance, the experiment considers two configurations: 2 and 3 hidden layers. These configurations, in turn, feature distinct node sizes to comprehensively evaluate their impact on the models' performance.

Table 3. Model B Comparisons between Hidden Layers

Number of Hidden Layers	Hidden Layer Nodes Configurations	Test Loss	Test Accuracy
2	[256, 256]	0.063	0.981
2	[256, 128]	**0.062**	0.982
2	[128, 64]	0.066	0.982
3	[256, 128, 64]	0.068	**0.983**
3	[128, 64, 32]	0.075	0.982

Within the Model B presets shown in **Table 3**, two configurations stand out with the best performance scores. The model featuring 2 hidden layers, comprising 256 nodes near the input layer and 128 nodes near the output layer, attains the lowest test loss score at 0.062. Conversely, the model with 3 hidden layers, consisting of 256 nodes near the input layer and 64 nodes near the output layer, achieves the highest test accuracy, scoring an impressive 0.983.

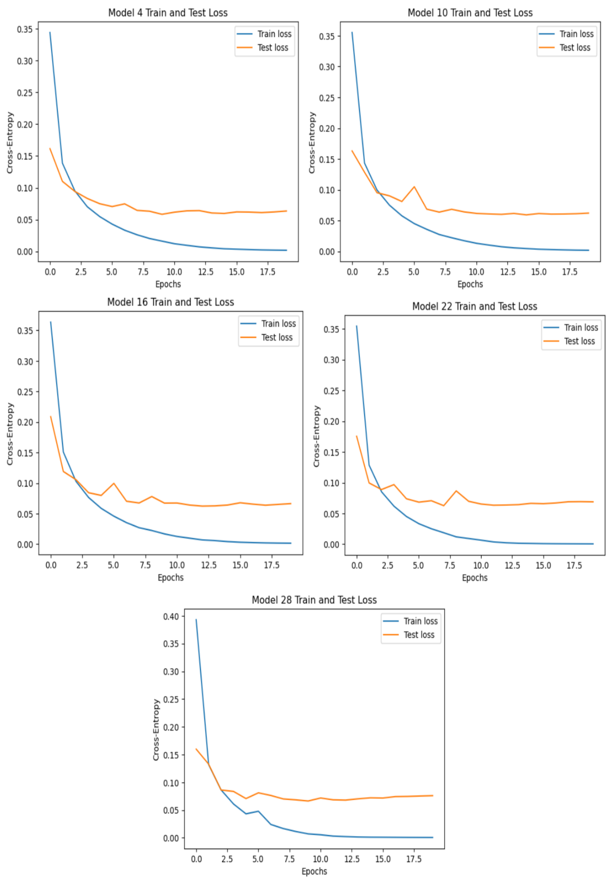

Figure 9. Model B Results

It's worth noting that these models exhibit nearly identical results due to the relatively straightforward nature of the classification task, as it was shown in the graph in **Figure 9**. Consequently, with the appropriate configuration of learning rate and weight initialization, the number of hidden layers appears to have minimal impact on model performance **[9]**.

However, it is interesting to see that models with a smaller number of nodes in the hidden layer close to the output layer tend to outperform their counterparts with larger numbers of nodes in this layer. This phenomenon can be attributed to the principle of network simplicity. When there are fewer nodes near the output layer, the model exhibits a reduced capacity for capturing complex patterns and fine-grained details in the data. This limitation, paradoxically, can lead to enhanced generalization performance, especially when the dataset and task are relatively straightforward. In such cases, the model is less prone to overfitting, where it essentially memorizes the training data but struggles to generalize to new, unseen data **[10]**. Instead, it focuses on capturing the essential features required for accurate classification, resulting in better overall performance.

The ideal quantity of nodes within each layer is contingent on factors such as problem complexity, dataset size, and network architecture. Determining the optimal number of nodes for each layer often involves a process of empirical exploration. This entails training the network using varying numbers of nodes and subsequently assessing its performance to identify the most effective configuration.

V. Conclusions

Overall, the MLP model has demonstrated remarkable accuracy in the classification of Handwritten Digit images. After conducting an extensive experiment involving 30 different models, it became evident that a hyperparameter configuration of a 0.1 Learning Rate and 'He' Weight Initialization consistently yielded the best results, with several models achieving approximately 98% accuracy. The role of hidden layers in the training process was notable, particularly for relatively straightforward classification tasks. Interestingly, irrespective of the number of hidden layers used, similar performance levels were achieved, showcasing that complexity in this context didn't significantly impact outcomes **[10]**.

However, the number of nodes within each layer did exert influence, affecting the network's generalization capability. Networks with more nodes had enhanced pattern-learning capabilities but also demonstrated a propensity to memorize training data, potentially limiting their ability to generalize to new examples. In summary, the number of nodes within each layer emerges as a pivotal hyperparameter, profoundly shaping the neural network's capacity to discern intricate patterns within data.

Acknowledgment

I would like to express my sincerest gratitude to Mr. Wahyono, S. Kom., Ph.D., for the invaluable guidance and advice provided throughout the Deep Learning course this semester. Thank you for imparting your extensive knowledge and expertise in the field, which has been instrumental in shaping my ideas and expanding my understanding of the subject matter.

Furthermore, I understand that the study may not have been as comprehensive as some of the other research in the field, which utilizes more advanced neural networks and hyperparameters. Nonetheless, I am immensely proud of what I was able to create and the potential practical applications it holds.

References

[1] D. M. D'Addona, "Neural network," CIRP Encyclopedia of Production Engineering, pp. 911–918, 2014. doi:10.1007/978-3-642-20617-7_6563

[2] T. Elansari, M. Ouanan, and H. Bourray, Modeling of Multilayer Perceptron neural network hyperparameter optimization and training, 2023. doi:10.21203/rs.3.rs-2570112/v1

[3] K.-C. Ke and M.-S. Huang, Enhancement of multilayer perceptron model training accuracy through the optimization of hyperparameters: A case study of the quality prediction of injection molded parts, 2021. doi:10.21203/rs.3.rs-685234/v1

[4] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. Available at: http://yann.lecun.com/exdb/mnist/.

[5] K.-C. Ke and M.-S. Huang, "Quality Prediction for injection molding by using a multilayer perceptron neural network," Polymers, vol. 12, no. 8, p. 1812, 2020. doi:10.3390/polym12081812

[6] D. Li, H. Wang, and Z. Li, "Accurate and fast wavelength demodulation for FBG reflected spectrum using multilayer perceptron (MLP) neural network," *2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, 2020. doi:10.1109/icmtma50254.2020.00066

[7] M. Bae, "Optimizing the hyper-parameters of Multi-layer Perceptron with greedy search," American Journal of Computer Science and Technology, vol. 4, no. 4, p. 90, 2021. doi:10.11648/j.ajcst.20210404.11

[8] Y. Liu, "the image classification of MNIST dataset by using machine learning techniques". UC Merced, 2021. Available at: https://escholarship.org/uc/item/0505d532

[9] T. Pricope, "A contextual analysis of multi-layer perceptron models in classifying hand-written digits and letters: limited resources". 2021. ArXiv, abs/2107.01782.

[10] S. Albahli, F. Alhassan, W. Albattah, and R. U. Khan, "Handwritten digit recognition: Hyperparameters-based analysis," Applied Sciences, vol. 10, no. 17, p. 5988, 2020. doi:10.3390/app10175988

Appendix

Dataset & Source Code: http://tiny.cc/GoogleColab-MLP