Laboratory Assignment and Report

Comparative Analysis of SLP, MLP, and CNN Models: Performance Optimization and Overfitting Mitigation

Student Number：u5521142

MA Digital Media and Cultures

University of Warwick
IM931 Interdisciplinary Approaches to Machine Learning
Centre for Interdisciplinary Methodologies
November 2024

Words Count：2880

Introduction

This essay explores the application of three machine learning models: Single-Layer Perceptron (SLP), Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN)—conducting a total of six experiments using the MNIST dataset. The research aims to examine the performance optimization of these three algorithmic models through parameter tuning, identifying the model that achieves the highest accuracy on the reserved test set. Given the architectural differences among the three models, the hyperparameters that influence their performance vary accordingly. Therefore, the hyperparameters adjusted differ across experiments for each model. This essay will provide detailed explanations of the hyperparameter choices for each experiment.

The dataset used in this study is the course-provided mnist2.json, a standard dataset of handwritten digits. It consists of 50,000 grayscale images for training and 10,000 for testing, each with a resolution of 28x28 pixels. The images are divided into 10 categories, representing the digits 0 through 9. The goal of the experiments is to classify each digit image into its correct category. All experiments were conducted using GPU acceleration.

The experiments will employ Grid Search as the method for hyperparameter selection. Given that the primary goal of machine learning is to enable models to accurately predict new data, both model selection and hyperparameter tuning serve this overarching purpose. Exploring all possible combinations of hyperparameters becomes a key evaluation strategy. As Bergstra and Bengio (2012) point out, "choosing hyperparameters to minimize generalization error is an indispensable part of machine learning." Grid Search offers a straightforward and exhaustive approach to addressing this issue. This essay will focus on hyperparameter experiments across six tasks: one SLP experiment, two MLP experiments, and three CNN experiments.

A. SLP Experiment

A Single-Layer Perceptron (SLP) is a simple feedforward neural network consisting of a single layer of weighted inputs connected to an output layer, typically used for linear classification tasks.Since the SLP model has the fewest hyperparameters among the three models, the experiment uses an exhaustive search method to explore all combinations of the two hyperparameters.

Experiment 1: Hidden Layer Optimization

The objective of the first experiment is to optimize the performance of the SLP model by fine-tuning two hyperparameters: epochs and learning rate. Epochs refer to the number of times the model passes through the entire training dataset. Too few epochs may result in underfitting, where the model fails to fully learn the features of the data, while too many epochs can lead to overfitting, reducing the model's ability to generalize to new data. In this experiment, the search range for epochs is set to 10, 20, 30, 50, and 100.

Based on these epoch values, the learning rate hyperparameter is then tuned. The learning rate determines the step size for updating the model's weights during training. A learning rate that is too high can cause the model to be unstable or even prevent it from converging, while a learning rate that is too low may slow down the training process or lead to convergence to a suboptimal local minimum. Therefore, the search range for the learning rate is set to 0.001, 0.01, 0.05, 0.1, and 0.5.

B. MLP Experiments

A Multi-Layer Perceptron (MLP) is a feedforward neural network composed of multiple layers, including one or more hidden layers, that enables the modeling of complex, non-linear relationships; in essence, an MLP is an SLP structure with added hidden layers.The goal of the MLP experiments is to optimize model performance by fine-tuning the combination of hidden layers and Dropout layers.

Experiment 2: Hidden Layer Optimization

The second experiment focuses on finding the optimal combination of the number of hidden layers and the number of neurons per layer. The number of hidden layers is set to 1, 2, and 3, representing different model depths. For each layer configuration, three neuron counts are tested: 64, 128, and 256, representing different model capacities. Using Grid Search, all possible combinations of hidden layer configurations will be tested, and their performance will be evaluated on the validation set. The validation accuracy for each configuration will be recorded.

Experiment 3: Dropout Layer Optimization

Based on the results from hidden layer optimization, the third experiment aims to optimize the Dropout layers. Dropout is a common regularization technique that randomly disables a fraction of neurons during training to reduce the model's reliance on specific neurons, thereby preventing overfitting. This experiment tests the placement and dropout rate of Dropout layers by adding a Dropout layer after each hidden layer configuration.

The dropout rates are chosen based on Hinton's findings from experiments on a standard feedforward neural network. In his study, the test set error was reduced from 160 errors without any regularization to 130 errors by applying 50% Dropout to hidden layers and L2 regularization. Further reduction to 110 errors was achieved by adding a 20% Dropout to the input layer (Hinton, 2012). Therefore, this experiment will test dropout rates of 0.4, 0.5, and 0.6.

C. CNN Experiments

A Convolutional Neural Network (CNN) is a specialized type of deep neural network designed to process structured grid-like data, such as images, using convolutional layers to automatically and efficiently extract spatial features. Three hyperparameter

experiments were conducted on the CNN model: activation function optimization, convolution kernel size and quantity optimization, and pooling strategy optimization.

Experiment 4: Activation Function Optimization

The activation function is a core component of the non-linear mapping in CNNs, directly influencing training speed, gradient propagation, and overall performance. This experiment explores the impact of two activation functions on model performance: ReLU and Tanh. ReLU is the most commonly used activation function due to its ability to effectively mitigate the vanishing gradient problem. Tanh, on the other hand, outputs zero-centered data, making it suitable for tasks requiring symmetric distributions. Through Grid Search, various combinations of these activation functions will be tested in both convolutional and fully connected layers. The configuration yielding the best performance on the validation set will be used for final testing.

Experiment 5: Convolution Kernel Size and Quantity Optimization

The convolutional layer is the most critical feature extraction component in CNNs. The size and number of convolution kernels directly affect the model's ability to perceive features. Smaller kernels are effective for capturing local details, while larger kernels are better suited for extracting global features. Since 3x3 kernels are commonly used in MNIST tasks, this experiment will test kernel sizes of 2x2, 3x3, and 7x7. The number of kernels will be set to 16, 32, and 64.

Experiment 6: Pooling Strategy Optimization

Pooling layers are used in CNNs for feature dimensionality reduction and down-sampling, lowering computational complexity while retaining essential information. The size of the pooling window affects the collection of local feature information and the feature map size, while the pooling method determines the type of features captured. For example, MaxPooling focuses on prominent local features,

while AveragePooling emphasizes overall smoothness and balanced feature distribution. Both approaches can impact generalization performance. This experiment tests pooling window sizes of (2, 2) and (3, 3) with two pooling methods: MaxPooling2D and AveragePooling2D.

In the experiments above, the SLP model employed an exhaustive search to optimize the number of epochs and learning rate, focusing on the impact of fundamental hyperparameters on model performance. The MLP experiments aimed to optimize hidden layer configurations and Dropout rates, striking a balance between network capacity and regularization. The CNN model adopted the most complex optimization strategies, involving experiments on activation functions, convolution kernel size and quantity, and pooling strategies, to fully explore the model's potential in feature extraction and generalization.

Experiments on MNIST

Experiment 1: Hidden Layer Optimization

The experimental results, shown in table format, indicate that moderate learning rates (0.01, 0.05) provide the best stability and achieve higher validation accuracy at mid-to-high epochs. Low learning rates improve accuracy gradually but are limited (maxing at 0.8993), while high learning rates show faster early convergence but greater fluctuations and slight performance drops at higher epochs. The highest validation accuracy, 0.9185, was achieved with 50 epochs and a learning rate of 0.05. However, loss and accuracy curves indicate overfitting, as validation loss rises while validation accuracy drops. Each experiment took 49 minutes.

Epochs	Learning Rate	Validation Accuracy
50	0.05	0.9185
100	0.01	0.9177

100	0.05	0.9173
30	0.05	0.917
30	0.1	0.9169
50	0.1	0.9162
20	0.05	0.9159
20	0.1	0.9159
50	0.01	0.915
10	0.1	0.9147
10	0.05	0.9143
100	0.1	0.9143
30	0.01	0.9115
10	0.5	0.9091
30	0.5	0.9087
20	0.01	0.9085
20	0.5	0.9085
50	0.5	0.9079
100	0.5	0.9055
10	0.01	0.8996
100	0.001	0.8993
50	0.001	0.8915
30	0.001	0.885
20	0.001	0.8792
10	0.001	0.8587

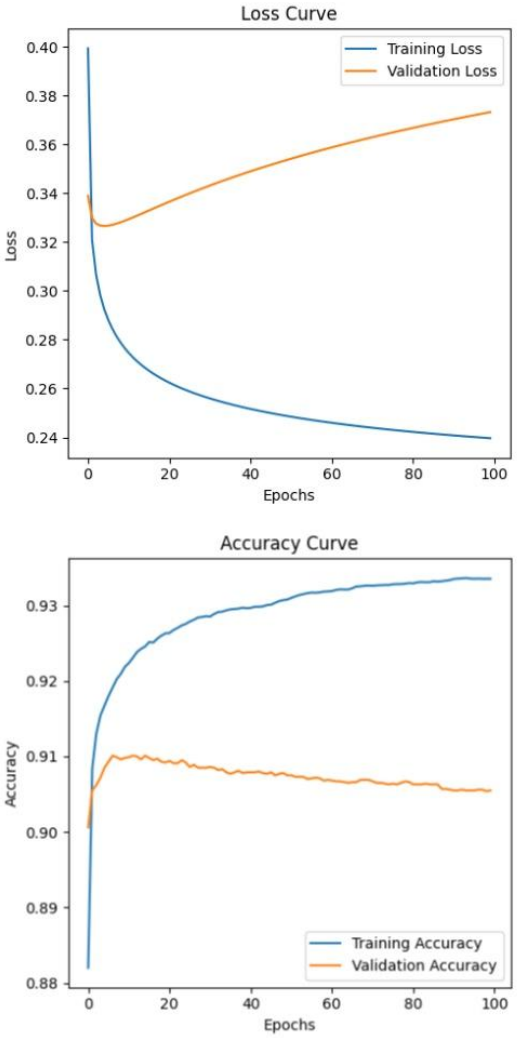


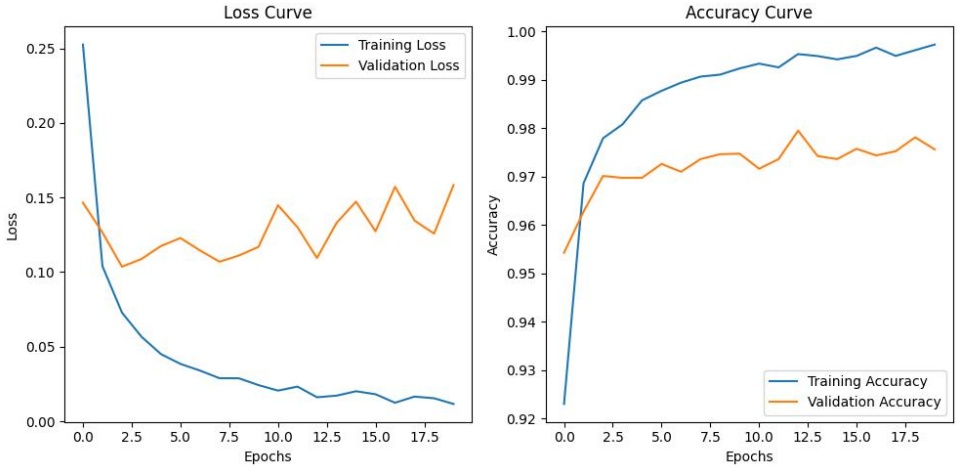
Fig.1 Loss and accuracy curves for the Single-Layer Perceptron (SLP) model trained with epochs = 10 and learning rate = 0.5, showing clear signs of overfitting as validation loss increases while validation accuracy stagnates.

Experiment 2: Hidden Layer Optimization

As the number of hidden layers increases, the model's validation accuracy shows slight improvements in certain configurations, but overall changes are limited. Across all hidden layer configurations, validation accuracy fluctuates between 0.9650 and 0.9737. The best accuracy, 0.9756, was achieved with three hidden layers, each

containing 256 neurons. Regardless of the number of hidden layers, increasing the number of neurons generally enhances validation accuracy.

In summary, deeper configurations (2 or 3 layers) and higher neuron counts (256) improve model performance. However, the performance gains diminish as depth and capacity increase, particularly given the small accuracy differences in this experiment. Notably, the loss curve for the best-performing configuration shows a clear upward trend over time, indicating that the overfitting issue persists.



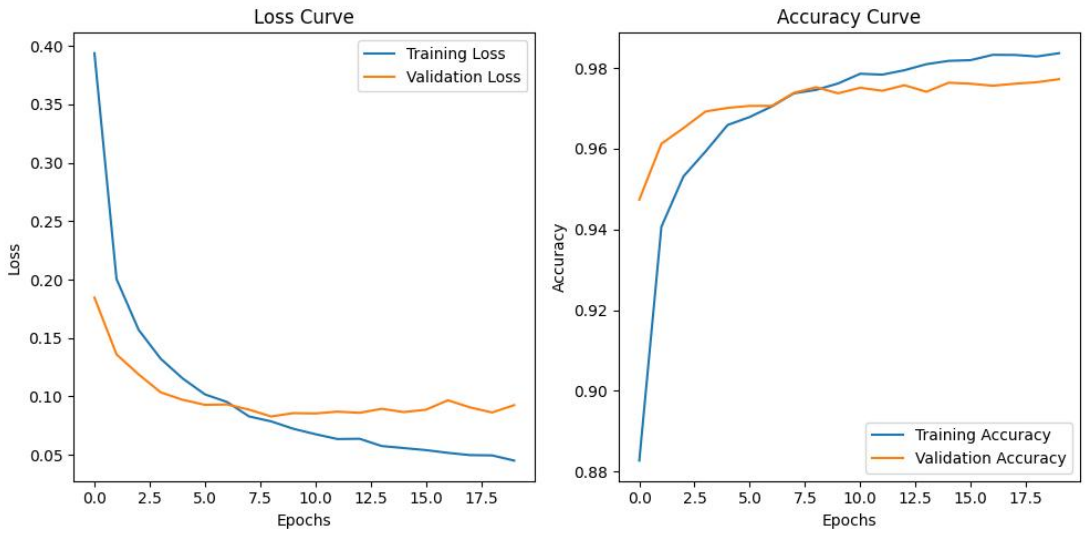
Hidden Layers	Units per Layer	Validation Accuracy
3	256	0.9756
2	256	0.9753
3	128	0.9747
1	128	0.9737
1	256	0.9734
2	128	0.9718
3	64	0.9695
1	64	0.965
2	64	0.9626

Fig.2 Loss and accuracy curves for the Multi-Layer Perceptron (MLP) model trained with 3 hidden layers and 256 units per layer, showing persistent overfitting as validation loss continues to rise while validation accuracy plateaus.

Experiment 3: Dropout Layer Optimization

Introducing Dropout layers significantly mitigates the overfitting issue(Figure2 & 3), with validation accuracy ranging between 0.968 and 0.9772, showing a narrower range compared to models without Dropout. The results indicate that a high Dropout rate (0.6) generally leads to lower validation accuracy across all hidden layer configurations, dropping to 0.9680 for three hidden layers, suggesting that excessive Dropout may cause underfitting.

Medium and low Dropout rates performed better, with a 0.4 Dropout rate excelling in shallow networks but insufficient to prevent overfitting in deeper ones. The best-performing configuration was a single hidden layer with a Dropout rate of 0.5, achieving a validation accuracy of 0.9772. This suggests that adding Dropout layers offers greater performance gains in MLP models than merely increasing the number of hidden layers.



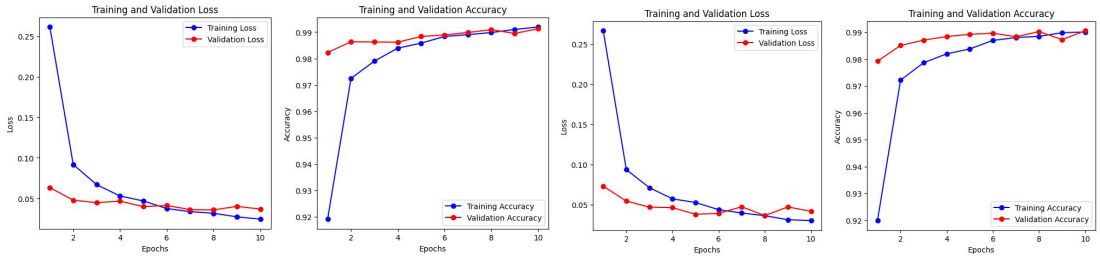
Hidden Layers	Dropout Rate	Validation Accuracy
1	0.5	0.9772
1	0.6	0.977

Fig.3 Loss and accuracy curves for the Multi-Layer Perceptron (MLP) model trained with 1 hidden layer and a 0.5 dropout rate, showing effective mitigation of overfitting, though the result reflects performance in a shallow network configuration.

1	0.4	0.9769
2	0.5	0.9761
3	0.4	0.9761
2	0.4	0.9758
3	0.5	0.9739
2	0.6	0.9718
3	0.6	0.968

Experiment 4: Activation Function Optimization

Before starting the CNN experiments, a baseline model was established using ReLU as the activation function. This model achieved a validation accuracy of 0.9931, higher than any of the previous experiments, and showed no significant signs of overfitting in the loss curve visualization. However, a notable drawback was its runtime of 11 minutes per experiment, nearly double that of the MLP.



Loss and accuracy curves for the CNN model

Fig. 4.(Left) Conv1 activation = ReLU and Conv2 activation = ReLU.

Fig. 5.(Right) Conv1 activation = ReLU and Conv2 activation = Tanh.

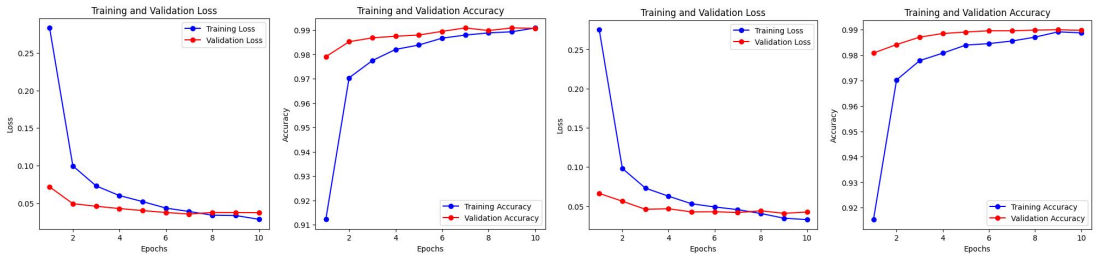


Fig. 6.(Left) Conv1 activation = Tanh and Conv2 activation = ReLU.

Fig. 7.(Right) Conv1 activation = Tanh and Conv2 activation = Tanh.

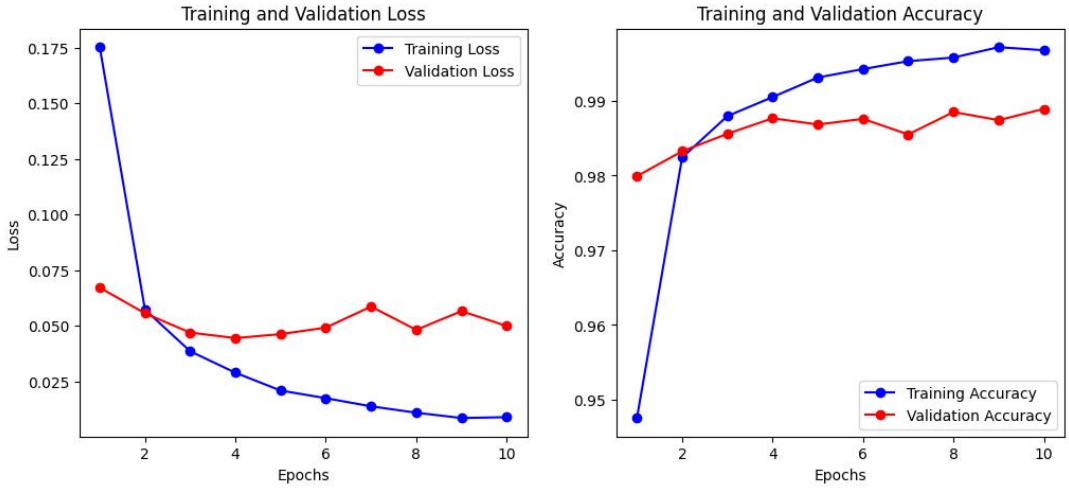
Conv1 Activation	Conv2 Activation	Learning Rate	Test Loss	Test Accuracy
ReLU	ReLU	0.001	0.0256	0.9928
Tanh	ReLU	0.001	0.0282	0.9917
ReLU	Tanh	0.001	0.0324	0.9912
Tanh	Tanh	0.001	0.0321	0.9902

The subsequent experiments aimed to optimize CNN performance by testing different activation function combinations (ReLU and Tanh) in the convolutional layers. The goal was to assess whether introducing Tanh could complement ReLU or enhance accuracy. However, the results showed that ReLU remained the best performer, achieving the highest test accuracy of 99.28% with the lowest test loss of 0.0256. The Tanh + ReLU combination yielded a test accuracy of 99.17%, closely matching the ReLU + ReLU configuration. This indicates that even when ReLU is used only in the second layer, it remains highly effective.

Experiment 5: Convolution Kernel Size and Quantity Optimization

The experiment evaluated the combined impact of different convolution kernel sizes and filter counts on model performance, with test accuracy ranging between 0.9849 and 0.9909. The (3, 3) kernel consistently performed best, achieving the highest test accuracy of 99.09% and the lowest test loss of 0.0364 with 16 filters. The (2, 2) kernel also performed well with 32 filters, but showed higher losses in other configurations, indicating that smaller kernels may be more sensitive to fine-grained features.

The (7, 7) kernel resulted in slightly lower test accuracy and higher loss, particularly on small datasets like MNIST, where it appears less effective. Additionally, 32 filters provided a good balance between computational efficiency and model performance, making it a practical choice under resource constraints.



Kernel Size	Filters	Test Loss	Test Accuracy
(3, 3)	16	0.0364	0.9909
(2, 2)	32	0.0434	0.9905
(3, 3)	64	0.0439	0.99
(7, 7)	64	0.0499	0.9898
(7, 7)	32	0.0487	0.9888
(7, 7)	16	0.043	0.9886
(3, 3)	32	0.0472	0.9883
(2, 2)	64	0.0648	0.9855
(2, 2)	16	0.0574	0.9849

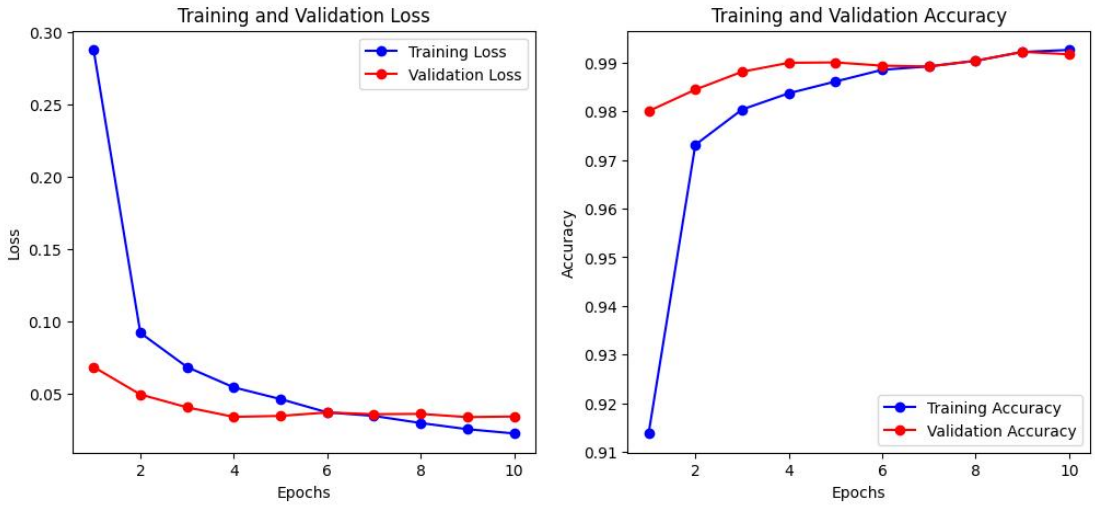
Fig.8 Loss and accuracy curves for the Convolutional Neural Network (CNN) model trained with 16 filters and a kernel size of (3, 3), showing less effective handling of overfitting compared to the configurations in Experiment 4.

Experiment 6: Pooling Strategy Optimization

The experiment evaluated the impact of pooling strategies on model performance, with MaxPooling2D demonstrating superior and more stable results. The best configuration was MaxPooling2D with a (2, 2) window, achieving a test accuracy of 99.21% and the lowest test loss of 0.0259. This pooling method effectively extracts key features while suppressing irrelevant noise.

The (2, 2) window consistently produced high test accuracy across all pooling methods. In contrast, the larger (3, 3) window performed worse, particularly with

AveragePooling2D, where test accuracy declined more significantly, likely due to the loss of local information. AveragePooling2D, which prioritizes feature smoothness, may be better suited for tasks requiring balanced feature distribution, explaining its inferior performance compared to MaxPooling2D in this experiment.



Pooling Type	Pool Size	Test Loss	Test Accuracy
MaxPooling2D	(2, 2)	0.027	0.9926
AveragePooling2D	(2, 2)	0.0279	0.9908
MaxPooling2D	(3, 3)	0.0409	0.9881
MaxPooling2D	(4, 4)	0.0462	0.9848
AveragePooling2D	(3, 3)	0.0524	0.9823
AveragePooling2D	(4, 4)	0.1068	0.9671

Fig.9 Loss and accuracy curves for the Convolutional Neural Network (CNN) model trained with MaxPooling2D and a pool size of (3, 3), showing improved handling of overfitting compared to Experiment 5.

Comparative analysis

Model type	Hyperparameter(s)	training time（min）	Test Accuracy Interval	
SLP	Epochs & Learning rate	49	0.9185	0.8587
MLP	Hidden Layers and units per Layer	55	0.9756	0.9626
MLP	Dropout Rate	62	0.9772	0.968
CNN	Conv1 Activation	50	0.9928	0.9902
CNN	Kernel Size & Filters	110	0.9909	0.9849
CNN	Pooling Type & Pool Size	45	0.9926	0.9671

In this essay, systematic tests were conducted on SLP, MLP, and CNN models to explore the impact of different parameter configurations on model performance. The results demonstrate that the CNN model outperforms the others in overall performance, while the optimized MLP model offers a strong alternative under resource constraints. In contrast, the SLP model, being a single-layer network, lags significantly behind the other two in terms of performance.

In the MLP experiments, Experiment 2 validated the approach proposed by researchers in 1986, where increasing the number of hidden units was suggested as a solution to the issue of poor local minima in lower-dimensional subspaces (David). Across configurations, the smallest Units per Layer value (64) consistently resulted in lower accuracy compared to 128 and 256 units. However, the number of hidden layers did not exhibit a proportional relationship with final accuracy, and overfitting was observed across all configurations in this experiment.

To address this, Experiment 3 introduced dropout rates, building on the results of Experiment 2 with 256 units per layer and three hidden layers. The results

successfully narrowed accuracy fluctuations, raised the lower accuracy bound, and effectively mitigated overfitting. This insight was applied to the CNN baseline model (Experiment 0), which included dropout layers from the outset, enabling the model to achieve high performance. This aligns with Hinton's (2012) observation that using non-convolutional higher layers with many parameters leads to significant improvements when dropout is applied but worsens performance without it.

Among the various hyperparameters tested, Experiment 5, which focused on optimizing the convolution kernel size and filter count in the CNN model, required the longest computation time but yielded the narrowest accuracy fluctuation range and the highest accuracy lower bound. Filters of 16 and 32 offered an ideal balance between computational cost and model performance. While 64 filters slightly improved feature extraction, they led to performance drops in some kernel configurations and significantly increased runtime.

In Experiment 4, ReLU demonstrated clear superiority over Tanh, achieving the highest test accuracy of 99.28% and the lowest test loss of 0.0256. As Dubey (2022) notes, “The Logistic Sigmoid and Tanh activation functions should be avoided for Convolutional Neural Networks as they lead to poor convergence. However, these activation functions are commonly used as gates in recurrent neural networks.” In light of this, it is reasonable to suspect that Sigmoid would also underperform compared to ReLU in the current setting.

Experiment 6, which focused on optimizing pooling layers, exhibited the widest accuracy fluctuation range among the CNN experiments, underscoring the critical role of feature extraction and noise suppression in CNNs. The observed trends in *Pool Size* contrasted with those of *Kernel Size* in Experiment 5, highlighting the differing priorities of pooling and convolutional layers in feature extraction and information compression.

Pooling layers primarily serve to compress features and reduce dimensionality, improving computational efficiency and generalization by minimizing redundant information. Smaller windows (e.g., (2, 2)) strike a balance by retaining more detail during compression, preventing excessive information loss. Conversely, convolutional layers focus on feature extraction, capturing both local patterns and global relationships. Medium-sized kernels (e.g., (3, 3)) excel in balancing local and global information, enhancing the model's representational capacity.

This aligns with LeCun et al.'s early assertion: “A large degree of invariance to geometric transformations of the input can be achieved with this progressive reduction of spatial resolution compensated by a progressive increase of the richness of the representation” (1998).

To sum up, the CNN model stands out for its ability to mitigate overfitting, improve accuracy, and enhance feature extraction, making it particularly well-suited for tasks with high precision requirements. Its main drawback is the longer training time, but its exceptional performance makes it the preferred choice for accuracy-critical tasks. Conversely, the optimized MLP model achieves a solid balance between validation accuracy and computational efficiency, especially with the inclusion of Dropout layers, making it a viable alternative in resource-constrained environments. In contrast, the SLP model is largely regarded as a limited version of MLP, with the lowest accuracy and the least promising application prospects among the three.

Conclusion

This study systematically evaluated SLP, MLP, and CNN models, focusing on the impact of different hyperparameter configurations on their performance. The results highlight CNN's superior accuracy and feature extraction capabilities, making it the ideal choice for tasks requiring high precision. However, its longer training time remains a notable drawback. The optimized MLP model demonstrated a good balance between validation accuracy and computational efficiency, especially with

the introduction of Dropout layers, positioning it as a viable alternative in resource-constrained environments. In contrast, the SLP model, as a simpler structure, exhibited significantly lower accuracy and limited practical applications compared to the other two.

Despite these findings, the study has some limitations. The experiments were conducted solely on the MNIST dataset, which may restrict the generalizability of the results to other data types. Additionally, the hyperparameter search was constrained to a predefined range, leaving room for further exploration. The lack of a more diverse dataset and broader experimentation could have influenced the robustness of the conclusions.

Future work could focus on applying these models to more diverse datasets to test their generalization capabilities. Further collaboration with technical experts might also help optimize model structures and training processes. Additionally, exploring practical applications of these models in fields such as finance and education could provide valuable insights into their real-world utility.

Reference

[1] Bergstra.J and Bengio.Y (2012). Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, null (3/1/2012), 281–305.

[2] Dubey, S.R., Singh, S.K. and Chaudhuri, B.B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503,92–108. doi:https://doi.org/10.1016/j.neucom.2022.06.111.

[3] Hinton, G. (2012) Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

[4] Lecun, L. Bottou, Y. Bengio and P. Haffner, (1998) "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, doi: 10.1109/5.726791.

[5] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986) Learning representations by back propagating errors. *nature* 323, 533–536.