Understanding and Contrasting Multilayer Perceptrons and Convolutional Neural Networks for Image Classification

Ling-Yun, Huang

Huanglingyun510@gmail.com

Abstract—*This study compares the performance of two methods, Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs), in a image classification task. Through the different hyperparameter settings, CNNs consistently outperform MLPs. The findings show the importance of spatial feature extraction design, particularly in tasks that involve image data. Additionally, this study highlights the significant impact of parameter optimisation, such as learning rate adjustment, on model performance. Future research directions include exploring techniques like k-fold cross-validation and experiencing pre-trained CNN models to further enhance accuracy and address biases in image classification. Overall, this research provides valuable insights MLPs and CNNs in image analysis and recognition.*

1 Introduction

In this section, we will explore the strengths and weaknesses of two methods, Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs), particularly for their application in image classification tasks. Additionally, we will provide an overview of the image dataset used in this study.

1.1 Multilayer Perceptrons and Convolutional Neural Networks[1]

Multilayer Perceptrons (MLPs) are commonly used in many applications, including pattern recognition and model fitting. They consist of multiple layers of interconnection neurones, including an input layer, one or more hidden layers, and an output layer. Through the process of backpropagation, MLPs adjust the weights and biases of the neurones and optimise the performance of the model prediction.

Convolutional Neural Networks (CNNs) are specifically designed to learn the spatial features from data. This makes them good with image analysis and recognition tasks. CNNs consist of multiple layers, including conventional, pooling, and fully connected layers. The conventional layers use filters to extract features from data, while the pooling layers down-sample these features, retaining only the relevant information. Through backpropagation, CNNs adjust their parameters and optimise feature extraction, enabling them to make accurate predictions in data.

In the image classification task, MLPs treat each pixel in the image as an input neurone. However, with this approach the networks do not capture spatial relationships among neighbouring pixels, making it hard to increase the test accuracy. On the other hand, with conventional and pooling layers, CNNs could extract features by combining information from the neighbour pixels. This enables CNNs to capture the spatial patterns and effectively deal with the slight shift in the image.

Since spatial patterns are significant in image recognition, the hypothesis of this comparison is that MLPs may struggle to reach high accuracy by only relying on individual pixel values for

classification. In contrast, CNNs are expected to perform better in image classification tasks due to their ability for spatial pattern recognition.

1.2 Dataset

The dataset used in this study is “Intel Image Classification” sourced from Kaggle[2]. It contains images distributed in six different categories: building, forest, glacier, mountain, sea, and street. Figure 1 showcases representative images from each class. The dataset includes two subsets, with the first subset containing approximately 14,000 images and the second subset containing around 3,000 images. The larger set was used for model training and performance evaluation, while the smaller remained unseen until the best model was selected.

Figure 1. The representative images for each class.

2 Methods

In this section, we will describe the image preprocessing steps, the training and testing processes, and the architectures and hyperparameters setting of the networks.

2.1 Methodology

Before the training process, image preprocessing is performed, which includes resizing the image to 32x32 pixels with 3 channels. The images were then converted to tensor format and organized into batches of size 32.

Two subsets of the dataset were used in this study. The larger subset was used for the training process while the smaller subset remained unseen until two best models were selected. In the training process, the larger subset was randomly split into 80/10/10 percentages for training, validation, and testing respectively. The training set contains 80 percent of the dataset is used to train the neural network model. During training, 10 percent of the dataset is used to monitor the performance of the network. If the network did not show improvement within 5 epochs, the training would stop to avoid overfitting. Additionally, training will also terminate when it reaches the maximum epoch limit, which in this study is set to 20. Once the training was completed, the performance of the network was evaluated by the remaining 10 percent of the dataset.

The model with the highest accuracy on the testing set was selected as the best model for each method. The smaller subset was then used to assess the performance of these two best models.

2.2 MLPs Architecture and Hyperparameters

In MLPs, the architecture consists of an input layer, hidden layers, and an output layer. The input layer consists of neurons representing the flattened input image size of 32x32 pixels with 3 channels. The number of neurones in the hidden layers is determined by a parameter. ReLU activation functions are applied across all layers except the output layer, where SoftMax

activation function is used. The output layer consists of 6 neurons, each corresponding to one of the 6.

For the hidden layers, two settings are defined: one or two hidden layers. The hyperparameters are adjusted with varying numbers of neurons (64, 128, and 256) and different learning rates (0.0001, 0.001, and 0.01) to observe their impact on model performance.

2.3 CNNs Architecture and Hyperparameters

In CNNs, the architecture consists of convolutional layer followed by max pooling, a fully connected layer, and an output layer. The convolutional layer experiences different numbers of filters and kernel sizes. The max pooling layer performs max pooling with a kernel size of 2 and a stride of 2. The fully connected layer contains 64 neurones with ReLU activation, and the output layer consists of 6 neurons, each corresponding to one of the 6 classes.

The hyperparameters are adjusted with varying numbers of filters (10, and 20), different kernel size (3x3, 5x5, and 7x7), and different learning rates (0.0001, 0.001, and 0.01) to observe their impact on model performance.

3 Result

In this section, we will provide the evaluation of model selection and the comparison of two best models.

3.1 Two Methods Comparison and Best Models Selection

Table 1 shows the results of exploring different hyperparameters in MLPs and CNNs are presented, with the best model highlighted for each method.

Table 1: Comparison of MLPs and CNNs with Different Hyperparameters

MLPs with one or two hidden layers				CNNs with one convolutional layers				
Hidden Layer(s)	Leraning Rate	Epochs	Test Accuracy	Kernel Size	Filters	Leraning Rate	Epochs	Test Accuracy
[64]	0.0001	20	54.49%	3x3	10	0.0001	20	67.59%
[128]	0.0001	20	58.40%	3x3	20	0.0001	20	70.73%
[256]	0.0001	20	58.48%	5x5	10	0.0001	20	68.59%
[64,64]	0.0001	20	57.62%	5x5	20	0.0001	20	71.01%
[128,128]	0.0001	20	59.40%	7x7	10	0.0001	20	67.81%
[256,256]	0.0001	20	58.40%	7x7	20	0.0001	20	70.23%
[64]	0.001	14	56.84%	3x3	10	0.001	12	71.79%
[128]	0.001	20	59.05%	3x3	20	0.001	15	71.87%
[256]	0.001	20	57.19%	5x5	10	0.001	13	72.93%
[64,64]	0.001	20	59.54%	5x5	20	0.001	15	73.72%
[128,128]	0.001	20	59.40%	7x7	10	0.001	15	72.22%
[256,256]	0.001	15	55.48%	7x7	20	0.001	10	73.22%
[64]	0.01	6	16.74%	3x3	10	0.01	8	61.25%
[128]	0.01	9	17.52%	3x3	20	0.01	9	66.17%
[256]	0.01	9	17.52%	5x5	10	0.01	9	58.12%
[64,64]	0.01	6	16.03%	5x5	20	0.01	13	61.82%
[128,128]	0.01	6	16.74%	7x7	10	0.01	11	53.99%
[256,256]	0.01	6	16.03%	7x7	20	0.01	10	50.00%

In MLPs, no significant difference was observed between using one or two hidden layers. When the learning rate was set to 0.0001 or 0.001, the accuracy of testing set ranged approximately from 55% to 60%. However, the lack of a clear pattern in these variations could be due to the random initialisation of weights and biases. Some models triggered the early stopping criterion before reaching the maximum number of epochs, especially with a learning rate of 0.001. On the other hand, with a learning rate of 0.01, not only did the models stop early, but they also showed minimal learning, leading to accuracies comparable to random guessing, at around 17%.

In CNNs, most models performed better than those in MLPs, indicating the superior performance of convolutional architectures in image classification tasks. However, there was no significant difference observed with different kernel sizes and filters when the learning rate was kept the same. When the learning rate was set to 0.0001, none of the models reached the early stopping criterion, suggesting they might not have had a chance to converge before reaching the maximum number of epochs. Conversely, with a learning rate of 0.001, the models reached the early stopping criterion and achieved the best performance in terms of test accuracy. However, when the learning rate was set to 0.01, the models triggered early stopping but performed worse than those with a learning rate of 0.001.

Regarding the different kernel sizes, the 5x5 kernel size appeared to perform slightly better than others. Similarly, when using smaller learning rate setting, the use of 20 filters produced better performance compared to 10 filters. However, this observed trend may also have been influenced by various factors, including the initial weight and bias choices, or the data split used for training, validation, and testing. These factors could have impacted the network's performance and should be considered when interpreting the results.

Overall, CNNs outperformed MLPs in this study. The best model in MLPs consists of two hidden layers with [64,64] neurones and a learning rate of 0.001, and the best model in CNNs featured a kernel size of 5x5, 20 filters, and a learning rate of 0.001.

3.2 Comparison of Two Best Models

Figure 2 illustrates the training and validation losses of the two best models. It is notable that while MLPs show gradual improvement in losses with each epoch, CNNs start with lower losses and exhibit more significant improvement over time compared to MLPs.

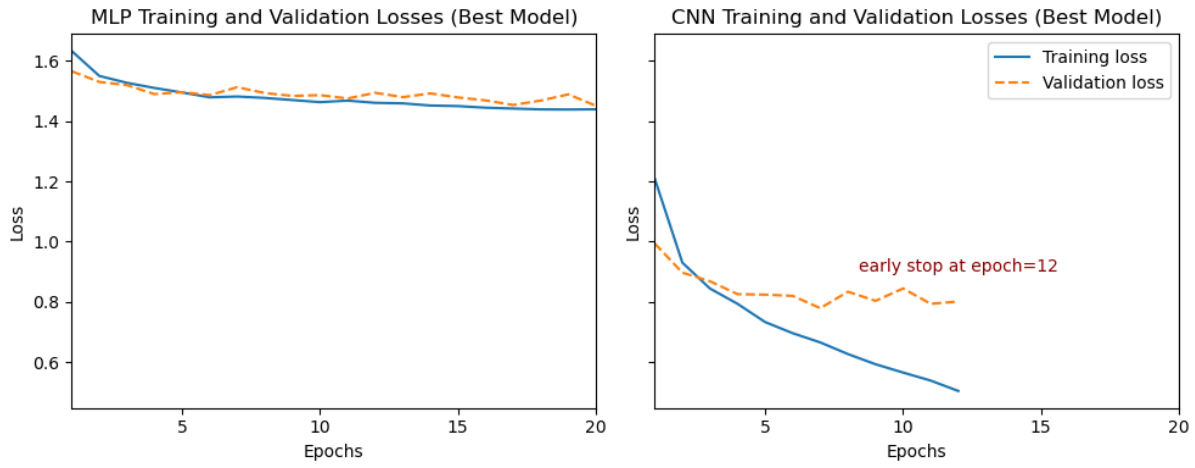


Figure 2. Training and validation losses in two best models.

After selecting the two best models, we proceeded to evaluate their performance using unseen data. Figure 3 presents the metrics, including precision, recall, and F1-score, along with the confusion matrix, illustrating the performance of the two best models on the unseen data. The accuracy results with MLPs achieving 0.58 and CNNs achieving 0.71, aligning with our observations during the model training process. The comparison of metrics clearly indicates that CNNs outperformed MLPs across all evaluated criteria. This emphasizes the effectiveness of CNNs over MLPs for the given task.

The confusion matrix in a multi-class task provides a comprehensive overview of the models' predictions compared to the groud truth across all classes. It also allows us to examine the models' performances in distinguishing between any two classes, revealing patterns of misclassification and providing valuable insights into these models' behaviours.

In our two best models' performance, we observed that many images classified as class 0 (building) were mistakenly predicted as class 5 (street), likely because street images often contain buildings within them. Moreover, within classes 2 to 4 (glacier, mountain, and sea), there were a lot of misclassifications among these classes, suggesting similarities in the visual feature of these landscapes.

MLPs best model performance on unseen data			
Class	Precision	Recall	F1-score
0	0.53	0.32	0.40
1	0.70	0.77	0.73
2	0.56	0.67	0.61
3	0.60	0.64	0.62
4	0.46	0.33	0.38
5	0.56	0.70	0.63
Model Accuracy : 0.58			

CNNs best model performance on unseen data			
Class	Precision	Recall	F1-score
0	0.69	0.60	0.65
1	0.82	0.91	0.86
2	0.75	0.63	0.69
3	0.71	0.63	0.66
4	0.60	0.73	0.66
5	0.72	0.78	0.75
Model Accuracy : 0.71			

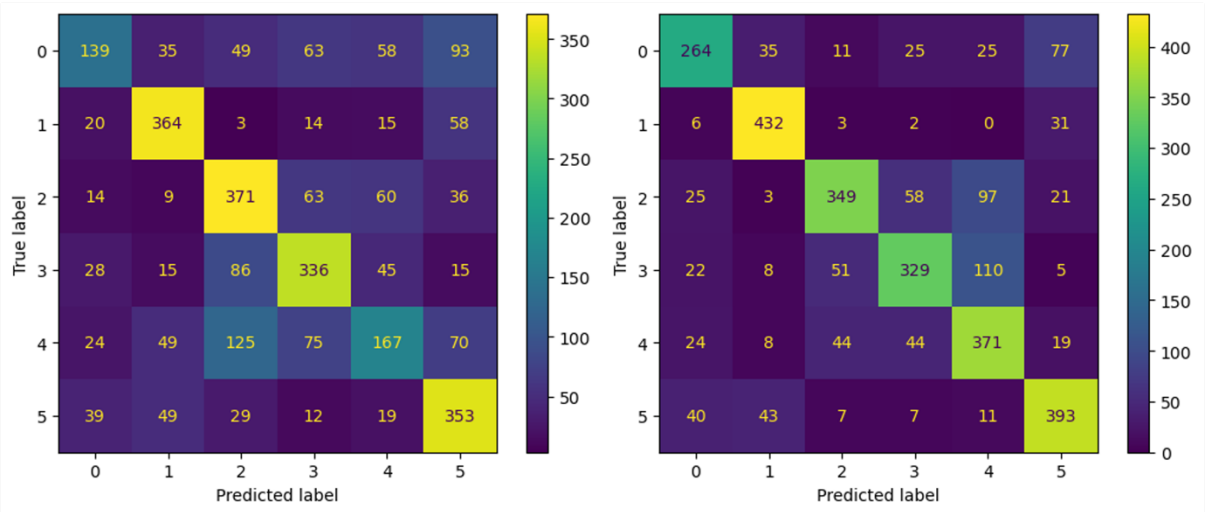


Figure 3. Two best models' performance on unseen dataset.

4 Lesson Learn and Future Work

Through this study, it became evident that CNNs outperform MLPs in image classification tasks. This highlights how important it is to use architectures specifically designed to extract spatial features when working with image data. Additionally, the study shows the significant impact of the learning rate setting on model performance during the training process. Exploring

different learning rates within the range of 0.0001 to 0.001, or training with more epochs in the smaller learning rate setting, could potentially improve results. However, it's important to recognize that while these methods might improve accuracy, MLPs are limited in their ability to recognize spatial patterns, which could restrict how much they can improve.

Moreover, the study showed that neural networks can be influenced by the initial values of weights and biases. With only one data split, there's a risk of biased results. To address this, using k-fold cross-validation could provide more reliable results, even though it would take longer to get them. Also, making use of pre-trained CNN models could be a promising approach. Applying these pre-trained algorithms, such as VGG, ResNet[3] or models trained on ImageNet[4] and applying them to the images in this study could provide insights into their performance and potentially improve classification accuracy.

In conclusion, CNNs prove to be more effective than MLPs in image classification due to their ability to extract spatial features effectively. This study shows the importance of optimising parameters like learning rate, while also suggesting future research directions such as using k-fold cross-validation and pre-trained CNN models to enhance accuracy and deal with biases. Overall, this research provides valuable insights for improving both MLPs and CNNs methods in image analysis and recognition.

References

[1] Haykin Simon, *Neural networks and learning machines*, 3rd ed. Upper Saddle River, N.J. : Pearson Education, 2009.

[2] PUNEET BANSAL, “Intel Image Classification,” Kaggle. Accessed: Apr. 13, 2024. [Online]. Available: https://www.kaggle.com/datasets/puneet6060/intel-image-classification

[3] J. Liang, “Image classification based on RESNET,” *J Phys Conf Ser*, vol. 1634, no. 1, p. 012110, Sep. 2020, doi: 10.1088/1742-6596/1634/1/012110.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

Appendix 1 – glossary

Multilayer Perceptrons (MLPs): Neural networks consisted of multiple layers, including input, hidden, and output layers, commonly used for pattern recognition and model fitting tasks.

Convolutional Neural Networks (CNNs): Deep learning architectures specifically designed for analysing visual data, by learning spatial features through convolutional and pooling layers.

Hyperparameters: Parameters that are set before the training process begins, such as learning rates and layer sizes, which influence the model's learning process.

Learning Rate: A hyperparameter that determines the step size at which the model adjusts its parameters during training, influencing the speed and stability of the learning process.

Backpropagation: An algorithm used to train neural networks by iteratively adjusting the model's parameters to minimize the difference between predicted and actual outputs.

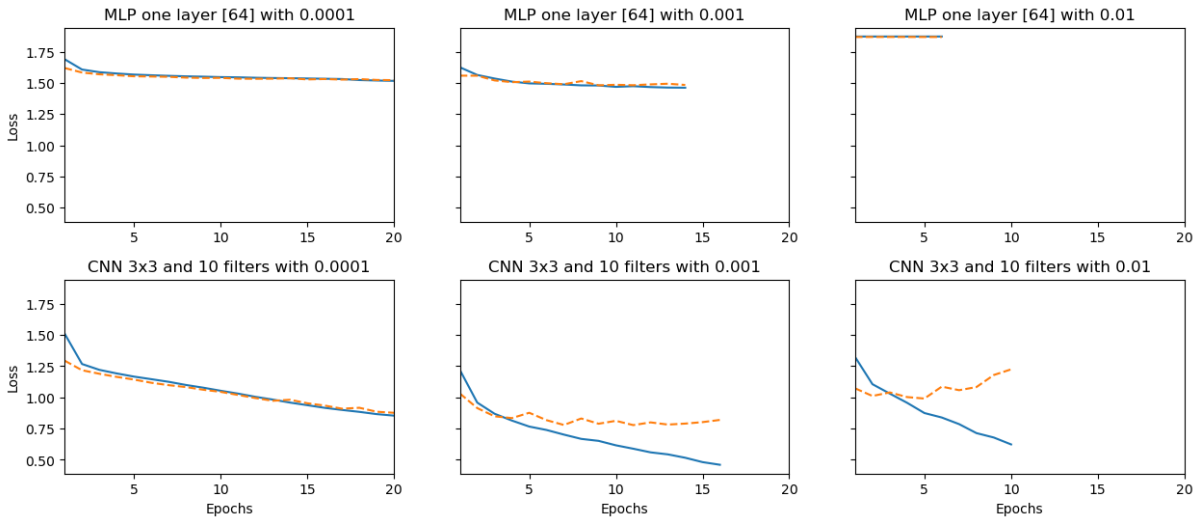
Max Pooling: A technique used in CNNs to reduce the spatial dimensions of features and only the most important information is remained.

Early Stopping: A technique used to prevent overfitting by stopping the training process when the model's performance on a validation dataset begins to decrease.

Pre-trained Models: Neural network models that have been trained on large datasets for specific tasks and can be fine-tuned or used directly for similar tasks.

Appendix 2 – Intermediate results

The training and validation losses



Since different parameters with the same learning rate perform similarly, only the simplest model settings are shown here. This figure shows the loss for training and validation at different learning rates. For CNNs, it is obvious that its starting loss is lower compared to MLPs. When the learning rate is 0.001, the model shows a better learning process compared to the other two learning rate settings (0.0001 and 0.01).

Appendix 3 – Implementation details

The models were implemented using Python 3.9.13 with PyTorch library version 2.2.1. Additional libraries included torchvision, matplotlib, and scikit-learn for data processing and visualisation.

Image data was preprocessed by resizing images to 32x32 pixels with 3 channels and converting them to tensor format. During the training of neural network models, the Adam optimizer was employed with learning rate at (0.0001, 0.001, and 0.01). A categorical cross-entropy loss function was used, along with a fixed batch size of 32. Training continued for a maximum of 20 epochs, with early stopping implemented if no improvement was observed within 5 epochs on the validation set. Model performance was evaluated using an 80/10/10 train-validation-test split.

For MLPs, the input layer was flattened images of size 32x32 pixels with 3 channels, followed by one or two hidden layers with varying numbers of neurones (64, 128, and 256). ReLU activation functions were applied across all layers, with a SoftMax activation function at the output layer comprising 6 neurons corresponding to the classes.

For CNNs, a convolutional layer followed by max pooling, a fully connected layer, and an output layer. The convolutional layer used different filters (10, and 20) and kernel sizes (3x3, 5x5, 7x7) to extract features, while the fully connected layer comprised 64 neurons with ReLU activation.