Blood Cell Classification Using Deep Learning: MLNN and CNN

Pedro Miguel Queiroga Trindade

University of Coimbra

Abstract. I study supervised image classification on the BloodMNIST dataset, a multi-class problem of 8 blood cell types. I compare two artificial neural network families: a Multi-Layer Neural Network and a Convolutional Neural Network, each under multiple architectural settings, loss functions and optimisers. Models are trained on the official MedMNIST training split and selected using the validation split. Performance is evaluated with precision, confusion matrix and macro-F1 because the dataset is unbalanced. My results show that CNNs consistently outperform MLPs.

1 Introduction

Image classification is one of the most fundamental problems in computer vision. Over the last decade, deep learning has become the dominant paradigm for this task, due to the ability of neural networks to learn hierarchical representations directly from raw data. In particular, Convolutional Neural Networks (CNNs) have shown remarkable performance in a wide range of image domains, replacing the need for manual feature engineering.

Medical image analysis is a domain where this impact has been especially strong, as deep learning has been increasingly applied to assist clinical workflows and support computer-aided diagnosis. However, medical datasets are often small, imbalanced and heterogeneous, which makes evaluation and model selection more challenging.

In this work, I study supervised image classification on a subset of MedMNIST focused on blood cell imagery. I compare different neural network architectures, explore multiple training configurations, and evaluate performance under standard metrics. Our goal is to analyse the effect of model family, loss function and optimiser choices, and to identify which configuration provides the best performance for this task.

2 Dataset

BloodMNIST is a subset of MedMNIST composed of peripheral blood images annotated by experts. It is a multi–class classification task with eight blood cell types, where each sample is represented as a 3×28×28 RGB image. The official dataset split contains 17,092 images, divided into 11,959 for training, 1,721 for validation and 3,412 for testing.

- **Image Size**: 3 x 28 x 28 pixels
- **Classes**: 8 Blood Cell Types
- **Training Set**: 11,959 images
- **Validation Set**: 1,721 images
- **Test Set**: 3,412 images
- **Total**: 17,092 images

As in real medical contexts, the class distribution is unbalanced: some cell types are much rarer than others. Because the dataset is already pre–processed and split into train/validation/test, this work does not focus on data cleaning or feature engineering. Instead, the main focus will be on implementing and comparing machine learning models — in particular different neural network architectures, losses and optimisers — and analysing how these choices impact performance on this imbalanced problem.

3 Architeture

3.1 Multi Layer Neural Network

The Multi-Layer Neural Network (MLNN) was implemented as a fully connected feed-forward model. The input vector of size 3×28×28=2352 is flattened and passed through two hidden layers, each containing 1180 neurons (the average between the input and output dimensions). Each hidden layer uses the ReLU activation function and is followed by a dropout layer with rate 0.2 to reduce overfitting. The final layer is a linear classifier with 8 output units, corresponding to the blood cell classes. Depending on the selected configuration, the network output is passed through log-softmax function, according to the loss function used (Cross-Entropy or Negative Log-Likelihood).

MLNN Hyperparameter Config Table 1 presents the complete hyperparameter configuration.

Table 1: MLNN Model Configuration for BloodMNIST

Param	**Config-1**	**Config-2**	**Config-3**	**Config-4**
Hidden Layers	2	2	2	2
Hidden Layer Size	1180	1180	1180	1180
Activation Function	ReLU	ReLU	ReLU	ReLU
Loss Function	CrossEntropy	CrossEntropy	NLLLoss	NLLLoss
Optimizer	SGD	Adam	SGD	Adam
Learning Rate	0.01	0.001	0.01	0.001
Dropout	0.2	0.2	0.2	0.2
Momentum	0.9	–	0.9	–

In this work, four distinct configurations of Multi-Layer Neural Networks (MLNN) were defined for the BloodMNIST blood cell classification problem, as presented in Table 1. All configurations maintain a consistent basic architecture in terms of the number of hidden layers (2) and the size of the layers (1180 neurons), as well as the ReLU activation function and a dropout of 0.2, ensuring structural consistency across the models.

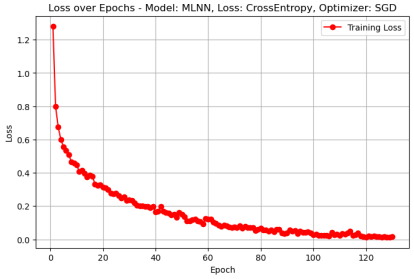
The differences between the configurations lie primarily in the training mechanisms, namely the choice of loss function, optimizer, learning rate, and momentum. These variations were designed to explore how different combinations of loss functions and optimization algorithms affect model performance:

Config-1 uses the CrossEntropy loss function combined with the SGD optimizer and a momentum of 0.9, with a learning rate of 0.01. This configuration represents a classical fully connected network training, allowing evaluation of the network behavior using stochastic gradient descent (SGD) with momentum.

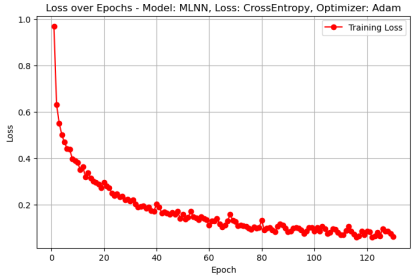
Config-2 retains CrossEntropy as the loss function but uses the Adam optimizer with a learning rate of 0.001, without explicit momentum. This configuration evaluates the impact of an adaptive optimizer, which automatically adjusts learning rates for each parameter, potentially accelerating convergence and improving performance.

Config-3 switches the loss function to NLLLoss, keeping SGD with a momentum of 0.9 and a learning rate of 0.01. This model allows analyzing the influence of the loss function choice on the training process, compared to the more common CrossEntropy for multi-class classification problems.

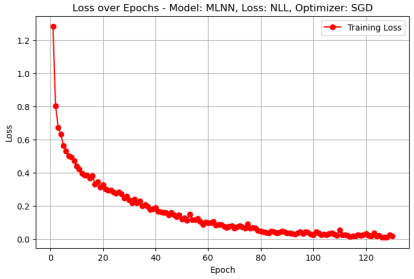
Config-4 combines NLLLoss with the Adam optimizer and a learning rate of 0.001, without momentum, allowing observation of how the NLL loss interacts with an adaptive optimizer.



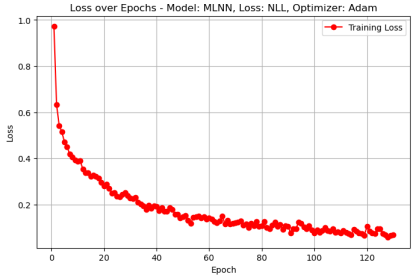
(a) Config-1: CrossEntropy + SGD



(b) Config-2: CrossEntropy + Adam



(c) Config-3: NLLLoss + SGD



(d) Config-4: NLLLoss + Adam

Fig. 1: Training loss curves for the four MLNN configurations on BloodMNIST.

The comparative analysis of the four MLNN configurations revealed clear differences in training behavior and convergence. Additionally, the SGD optimizer with momentum (**Config-1** and **Config-3**) produced smoother and more linear convergence curves, thanks to the stabilizing effect of momentum. In contrast, the Adam optimizer (**Config-2** and **Config-4**) allowed faster convergence but introduced more variability between epochs, resulting in less linear loss curves.

These observations suggest that, for BloodMNIST, SGD with momentum provides a more stable and predictable training trajectory, while Adam offers faster convergence at the cost of slightly more oscillations in the loss.

3.2 Convolutional Neural Network

The Convolutional Neural Network (CNN) was implemented to leverage spatial correlations in the BloodMNIST images. The network consists of three convolutional layers followed by ReLU activations and max-pooling operations.

- The first convolutional layer receives the input image with 3 channels and outputs 32 feature maps using a 3×3 kernel with padding 1, followed by a 2×2 max-pooling layer. - The second convolutional layer takes the 32 feature maps and outputs 64 feature maps using a 3×3 kernel with padding 1, followed by another 2×2 max-pooling layer. - The third convolutional takes the 64 layer

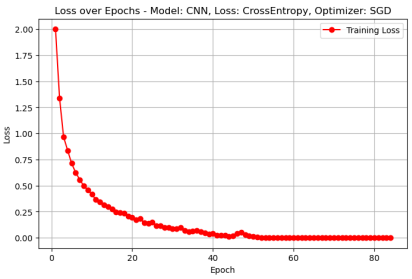
outputs 128 feature maps using a 3×3 kernel with padding 1, followed by a final 2×2 max-pooling layer.

The output of the last convolutional layer is flattened and passed through three fully connected layers with 256, 128, and 8 neurons, respectively. ReLU activations are applied after the first two fully connected layers to introduce non-linearity. The final layer is a linear classifier with 8 outputs corresponding to the blood cell classes. Depending on the selected configuration, the output can be passed through a log-softmax function, according to the loss function used (Cross-Entropy or Negative Log-Likelihood).

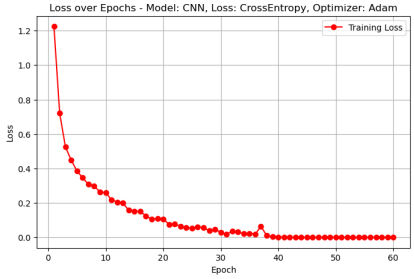
CNN Hyperparameter Config Table 2 presents the complete hyperparameter configuration.

Table 2: CNN Model Configuration for BloodMNIST

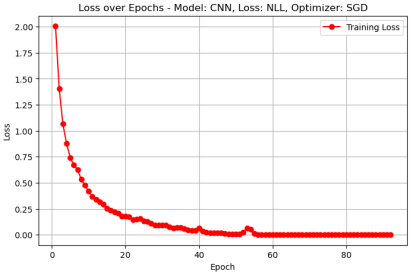
Param	**Config-1**	**Config-2**	**Config-3**	**Config-4**
Activation Function	ReLU	ReLU	ReLU	ReLU
Loss Function	CrossEntropy	CrossEntropy	NLLLoss	NLLLoss
Optimizer	SGD	Adam	SGD	Adam
Optimizer	0.01	0.001	0.01	0.001
Dropout	0.2	0.2	0.2	0.2
Momentum	0.9	–	0.9	–



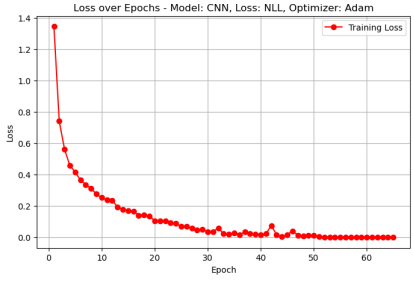
(a) Config-1: CrossEntropy + SGD



(b) Config-2: CrossEntropy + Adam



(c) Config-3: NLLLoss + SGD



(d) Config-4: NLLLoss + Adam

Fig. 2: Training loss curves for the four CNN configurations on BloodMNIST.

In both the MLNN and CNN models, using momentum-based SGD resulted in a smoother and more stable convergence of the loss function. The momentum term helps guide the weight updates in consistent directions by accumulating gradients over epochs, reducing abrupt fluctuations during training. In contrast, the Adam optimizer typically converges faster but can exhibit more variability in the loss, especially during the initial epochs.

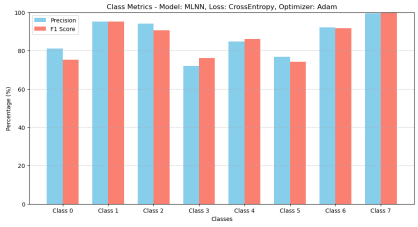
4 Results

In both neural network models, early stopping was used to prevent overfitting, and each configuration was carefully tuned to achieve the best possible results for the respective model.

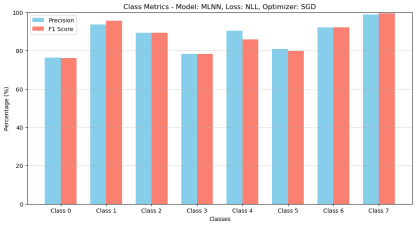
4.1 MLNN Results

Fig. 3: Precision and F1 Score for the four MLNN configurations on BloodMNIST.

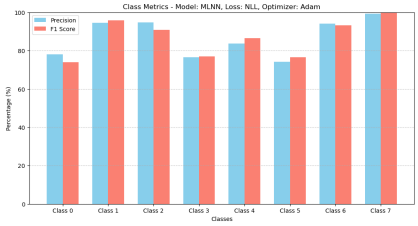
(a) Config-1: CrossEntropy + SGD



(b) Config-2: CrossEntropy + Adam



(c) Config-3: NLLLoss + SGD



(d) Config-4: NLLLoss + Adam

It was observed that when training for a very large number of epochs, the F1 scores of all configurations tend to converge to similar values. The weighted F1 Scores obtained were:

- CrossEntropy + SGD: 88.56%
- CrossEntropy + Adam: 87.67%

– NLLLoss + SGD: 88.57%
– NLLLoss + Adam: 88.38%

This indicates that, given enough training time, the networks are able to learn most patterns in the dataset, reducing observable differences between configurations. However, excessive training may increase the risk of overfitting if early stopping or regularization is not applied.

Fig. 4: Confusion Matrix

$$\begin{bmatrix} 176 & 1 & 0 & 51 & 2 & 14 & 0 & 0 \\ 0 & 593 & 0 & 9 & 1 & 1 & 19 & 1 \\ 3 & 1 & 278 & 11 & 7 & 2 & 5 & 4 \\ 24 & 8 & 7 & 470 & 10 & 34 & 26 & 0 \\ 5 & 0 & 12 & 16 & 207 & 1 & 2 & 0 \\ 5 & 0 & 2 & 48 & 2 & 226 & 1 & 0 \\ 1 & 13 & 9 & 27 & 3 & 1 & 612 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 470 \end{bmatrix}$$

Config-1: CrossEntropy + SGD

$$\begin{bmatrix} 171 & 1 & 1 & 44 & 5 & 22 & 0 & 0 \\ 3 & 594 & 0 & 11 & 1 & 2 & 13 & 0 \\ 2 & 0 & 272 & 12 & 12 & 3 & 8 & 2 \\ 27 & 8 & 4 & 468 & 16 & 31 & 25 & 0 \\ 3 & 0 & 5 & 15 & 212 & 4 & 4 & 0 \\ 5 & 0 & 1 & 68 & 4 & 204 & 2 & 0 \\ 0 & 21 & 6 & 32 & 0 & 0 & 607 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 470 \end{bmatrix}$$

Config-2: CrossEntropy + Adam

$$\begin{bmatrix} 185 & 1 & 0 & 38 & 2 & 18 & 0 & 0 \\ 1 & 609 & 0 & 3 & 1 & 1 & 9 & 0 \\ 4 & 1 & 278 & 9 & 3 & 3 & 8 & 5 \\ 33 & 18 & 9 & 453 & 8 & 28 & 30 & 0 \\ 10 & 0 & 14 & 14 & 199 & 2 & 4 & 0 \\ 7 & 0 & 1 & 45 & 5 & 224 & 2 & 0 \\ 2 & 21 & 9 & 17 & 2 & 1 & 614 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 470 \end{bmatrix}$$

Config-3: NLLLoss + SGD

$$\begin{bmatrix} 171 & 1 & 0 & 41 & 7 & 24 & 0 & 0 \\ 2 & 606 & 0 & 4 & 2 & 2 & 8 & 0 \\ 4 & 0 & 271 & 14 & 11 & 4 & 4 & 3 \\ 32 & 12 & 2 & 448 & 16 & 44 & 25 & 0 \\ 2 & 0 & 6 & 14 & 217 & 2 & 2 & 0 \\ 7 & 0 & 1 & 49 & 2 & 225 & 0 & 0 \\ 1 & 22 & 6 & 15 & 4 & 2 & 616 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 470 \end{bmatrix}$$

Config-4: NLLLoss + Adam

After analyzing the confusion matrices for each configuration, several general observations can be made.

Class 0 (first row) It is frequently confused with **Class 3**, as indicated by high values in column 3 (51, 44, 38, 41). This suggests that some samples of Class 0 share visual characteristics with Class 3, leading to misclassifications.

Class 1 (second row) Mostly classified correctly with high precision (590), but occasional confusions occur with Class 6 (19, 13, 9, 8). Class 6 appears to share some visual patterns with Class 1.

Class 2 (third row) Confused with Classes 3, 4, and 6 (values such as 11, 7, 5 and 12, 7, 8). Although there are misclassifications, they are less critical than those observed for Classes 0 and 3.

Class 3 (fourth row) While many samples are correctly classified, there are also confusions with Classes 0, 5, and 6. For example, notable values include 24, 27, 33, 32 in column 0 and 34, 31, 28, 44 in column 5. This indicates that Class 3 visually overlaps with multiple other classes.

Class 4 (fifth row) Some confusion with Classes 2 and 3 (values 12, 16, 14), but the majority of samples are correctly classified.

Class 5 (sixth row) Frequently confused with Class 3 (48, 68, 45, 49), indicating a significant visual similarity between these two classes.

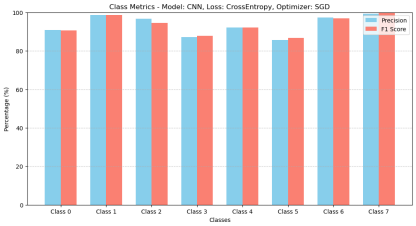
Class 6 (seventh row) Confusions occur with Classes 1, 2, and 3 (values 13, 9, 27; 21, 6, 32; 21, 6, 15), suggesting multiple overlaps with intermediate classes.

Class 7 (eighth row) Perfectly classified in all configurations (470/470). This class is easily identifiable with no observed confusions.

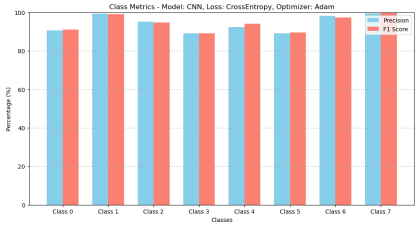
Overall, these confusion patterns remain consistent across all four configurations, confirming that the main source of classification errors is not the choice of loss function or optimizer, but rather the intrinsic visual similarity between specific classes in the dataset.

4.2 CNN Results

Fig. 5: Precision and F1 Score for the four CNN configurations on BloodMNIST.

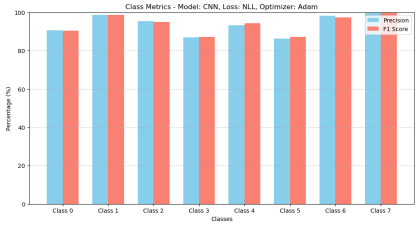


(a) Config-1: CrossEntropy + SGD



(b) Config-2: CrossEntropy + Adam

(c) Config-3: NLLLoss + SGD



(d) Config-4: NLLLoss + Adam

The respective weighted F1 scores obtained were:

- CrossEntropy + SGD: 94.20%
- CrossEntropy + Adam: 95.04%

– NLLLoss + SGD: 94.87%
– NLLLoss + Adam: 94.40%

As observed with the MLNN, when training for a sufficiently large number of epochs, the differences between configurations become minimal, indicating that all networks are able to learn the majority of patterns in the dataset.

Fig. 6: Confusion Matrix for CNN Configurations

$$\begin{bmatrix} 220 & 2 & 0 & 13 & 2 & 7 & 0 & 0 \\ 2 & 615 & 0 & 3 & 0 & 1 & 3 & 0 \\ 1 & 0 & 288 & 12 & 2 & 1 & 4 & 3 \\ 14 & 3 & 4 & 512 & 8 & 29 & 9 & 0 \\ 2 & 0 & 4 & 9 & 224 & 3 & 1 & 0 \\ 2 & 0 & 0 & 24 & 7 & 250 & 1 & 0 \\ 1 & 4 & 2 & 15 & 0 & 1 & 643 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 470 \end{bmatrix}$$

Config-1: CrossEntropy + SGD

$$\begin{bmatrix} 223 & 0 & 0 & 11 & 3 & 7 & 0 & 0 \\ 1 & 618 & 0 & 2 & 0 & 2 & 1 & 0 \\ 1 & 1 & 293 & 10 & 1 & 0 & 3 & 2 \\ 17 & 2 & 7 & 515 & 10 & 20 & 8 & 0 \\ 2 & 0 & 3 & 3 & 233 & 2 & 0 & 0 \\ 1 & 0 & 1 & 23 & 4 & 255 & 0 & 0 \\ 1 & 2 & 4 & 14 & 1 & 0 & 644 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 470 \end{bmatrix}$$

Config-2: CrossEntropy + Adam

$$\begin{bmatrix} 225 & 1 & 0 & 12 & 1 & 4 & 1 & 0 \\ 1 & 620 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 2 & 291 & 8 & 3 & 1 & 5 & 1 \\ 15 & 1 & 5 & 514 & 5 & 28 & 11 & 0 \\ 1 & 0 & 1 & 11 & 227 & 2 & 1 & 0 \\ 3 & 0 & 0 & 23 & 5 & 253 & 0 & 0 \\ 1 & 2 & 1 & 15 & 0 & 1 & 646 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 469 \end{bmatrix}$$

Config-3: NLLLoss + SGD

$$\begin{bmatrix} 220 & 2 & 0 & 16 & 1 & 5 & 0 & 0 \\ 3 & 616 & 0 & 4 & 0 & 1 & 0 & 0 \\ 1 & 1 & 294 & 9 & 1 & 1 & 3 & 1 \\ 15 & 2 & 7 & 505 & 10 & 31 & 9 & 0 \\ 0 & 0 & 2 & 7 & 232 & 2 & 0 & 0 \\ 2 & 0 & 2 & 26 & 4 & 250 & 0 & 0 \\ 2 & 4 & 3 & 14 & 1 & 0 & 642 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 470 \end{bmatrix}$$

Config-4: NLLLoss + Adam

After analyzing the confusion matrices for each configuration, several general observations can be made.

Class 0 (first row) Generally well classified, but occasionally confused with **Class 3**, as indicated by values like 13, 11, 12, 16 across the four configurations. This suggests that some Class 0 samples share visual features with Class 3, leading to misclassifications.

Class 1 (second row) Mostly classified correctly with high precision, but minor confusions occur with **Class 0** and **Class 6** (values such as 3, 4, 2, 3). This shows a slight visual overlap between Class 1 and Class 6.

Class 2 (third row) Confused primarily with **Class 3** (values: 12, 10, 8, 9), and to a lesser extent with Classes 1 and 6 (small values). Misclassifications exist but are less critical than for Classes 0 or 3.

Class 3 (fourth row) While many samples are correctly classified, there are confusions with **Classes 0, 2, 5, and 6** (values ranging roughly 13–31). This indicates that Class 3 overlaps visually with multiple other classes.

Class 4 (fifth row) Some confusions with **Classes 2 and 3** (values 4–11), but most samples are correctly classified.

Class 5 (sixth row) Confused mainly with **Class 3** (values: 24, 23, 26, 29), suggesting visual similarity between these classes.

Class 6 (seventh row) Occasional confusions with **Classes 3 and 2** (values: 14, 15), but the majority of samples are correctly identified.

Class 7 (eighth row) Perfectly classified in all configurations (470/470). This class is easy to identify with no observed confusion.

4.3 Comparison between MLNN and CNN

Overall, the CNN substantially reduces the magnitude of confusion compared to the MLNN, and most errors become marginal. The dominant pattern that persists is that Class 3 remains a central confusion source for multiple classes (especially 0, 2 and 5), while Class 7 is consistently trivial to classify. Comparing the overall performance of both models, it is evident that the CNN consistently outperforms the MLNN across all configurations. This demonstrates that convolutional architectures are better suited for image classification tasks, effectively capturing spatial patterns and features that fully connected networks struggle to represent. Consequently, CNNs achieve higher F1 scores, greater per-class precision, and more robust overall accuracy, confirming their superiority for this dataset.

5 Conclusion

In this project, I implemented and evaluated two neural network architectures, a Multi-Layer Neural Network (MLNN) and a Convolutional Neural Network (CNN), on the BloodMNIST dataset. Multiple configurations were tested, varying the loss functions (CrossEntropy and NLLLoss) and optimizers (SGD with momentum and Adam), with early stopping applied to prevent overfitting.

The results clearly indicate that CNNs outperform MLNNs across all configurations. CNNs not only achieve higher overall F1 scores and per-class precision but also exhibit more stable and consistent learning curves. The confusion matrices revealed that certain classes with visually similar features tend to be misclassified, highlighting the challenge of overlapping patterns in image data. Nonetheless, CNNs were more effective at minimizing these misclassifications.

Additionally, the choice of optimizer impacted convergence: SGD with momentum led to smoother loss curves and more stable training, while Adam converged faster but with more variability. Over long training periods, differences between configurations diminished, suggesting that the networks eventually learn the dominant patterns in the dataset.

In summary, the study demonstrates that convolutional architectures are better suited for image classification tasks involving complex spatial features. Future work could explore data augmentation or deeper CNNs to further improve performance and robustness.