Blood Cell Type Classification with Machine Learning

Nuno Batista[1] and Francisco Lapa Silva[1]

University of Coimbra, Portugal
nuno.batista@uc.pt
francisco.lapamsilva@gmail.com

1 Introduction

In this project we address the multi-class classification of blood cells using the BloodMNIST dataset. This task is complicated by significant class imbalance. We developed and systematically evaluated two fundamental neural network architectures: the Multi-Layer Perceptron (MLP) and the Convolutional Neural Network (CNN). Through a series of experiments, we compared their performance and investigated strategies, such as weighted loss functions, to mitigate the effects of the imbalanced data distribution. This report details our methodology, experimental setup, and a comparative analysis of the results to identify the most effective approach for this diagnostic challenge.

2 Problem Description

2.1 The BloodMNIST Dataset

The project utilizes the BloodMNIST dataset [1], a collection of 17,092 microscopic blood cell images derived from the Blood Cell Count and Detection (BCCD) dataset. Each color image has a $3 \times 28 \times 28$ resolution and is categorized into one of eight distinct blood cell types, as shown in Figure 1.

The distribution of samples across these eight classes, as detailed in Table 1, is highly imbalanced. For instance, the 'Neutrophil' class constitutes 34.01% of the dataset, whereas the 'Basophil' class represents only 2.36%. This imbalance poses a significant challenge, as a naive model could achieve a misleadingly high accuracy by simply predicting the majority class, while failing to identify rare but medically significant cell types. Consequently, standard accuracy is an inappropriate metric for evaluation, necessitating the use of metrics that are robust to imbalanced data.

3 Proposed Approach

3.1 Multi-Layer Perceptron (MLP)

As a foundational baseline, we first evaluate the Multi-Layer Perceptron. By flattening the $3 \times 28 \times 28$ image data into a 1D vector of 2352 features, the

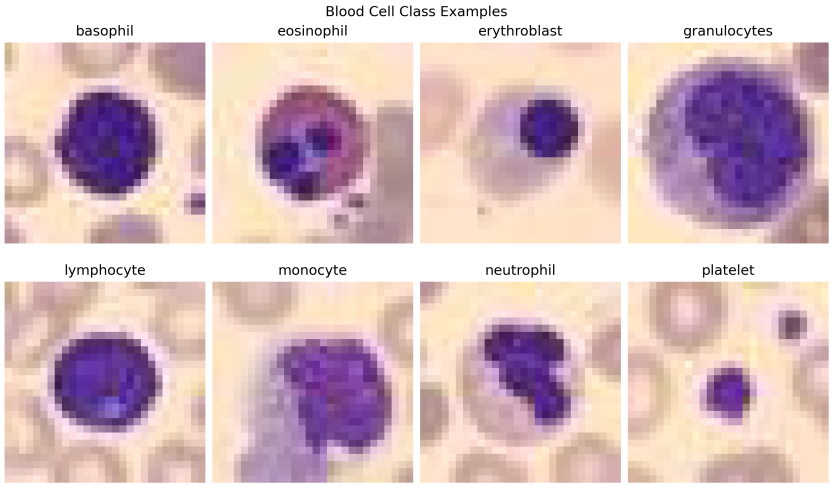

Fig. 1: Example images from the BloodMNIST dataset, illustrating the visual characteristics of the different cell types.

Table 1: Class distribution in the BloodMNIST dataset (MedMNIST v2).

Class	Cell Type	Number of Samples (%)
0	Basophil	404 (2.36%)
1	Eosinophil	1,048 (6.13%)
2	Erythroblast	1,924 (11.26%)
3	Immature Granulocyte	1,826 (10.68%)
4	Lymphocyte	3,769 (22.06%)
5	Monocyte	1,659 (9.71%)
6	Neutrophil	5,812 (34.01%)
7	Platelet	650 (3.80%)
Total		17,092 (100%)

MLP processes pixel information without spatial context. Our experiments explore how its performance is influenced by architectural depth and the choice of optimizer.

3.2 Convolutional Neural Network (CNN)

To leverage the spatial structure of the image data, we next implement Convolutional Neural Networks. Unlike MLPs, CNNs use convolutional and pooling layers to learn hierarchical feature maps, a design inherently suited for computer vision tasks. We conduct a parallel set of experiments to determine the optimal CNN architecture and training strategy.

3.3 Choice of Activation and Loss Functions

Two fundamental choices in our network design were the activation function for hidden layers and the loss function for training.

For activation, we exclusively used the Rectified Linear Unit (ReLU). Unlike older functions like Sigmoid, which saturates for large inputs and causes the "vanishing gradient" problem, ReLU does not. Its simple $\max(0, x)$ operation is computationally efficient and promotes sparsity in the network, making it the standard and superior choice for hidden layers in modern deep learning.

For the loss function, we used Cross-Entropy Loss. This function is standard for multi-class classification as it is designed to measure the divergence between the predicted probability distribution from the model's softmax output and the true distribution of the labels. It is the mathematically appropriate choice for penalizing incorrect categorical predictions, whereas alternatives like Mean Squared Error are designed for regression tasks involving continuous values.

3.4 Handling Class Imbalance

To mitigate the effects of the imbalanced class distribution, our primary strategy is the use of a weighted cross-entropy loss function. The weight for each class is calculated as the inverse of its frequency in the training dataset. This method assigns a higher penalty to misclassifications of rare classes, forcing the model to pay more attention to them during training and preventing it from becoming biased towards the majority classes.

4 Experimental Setup

4.1 Implementation Details

All models were implemented in Python3 using the PyTorch framework. The MedMNIST library was used for dataset loading. Model evaluation and metric calculations were performed using scikit-learn. As a preprocessing step, all images were transformed into tensors and each pixel's channels were normalized to a range of [-1, 1]. The dataset was divided into 11,959 training, 1,712 validation, and 3,421 test samples.

4.2 Hyperparameter Search Space

To identify the optimal configuration for our models, we performed a grid search over a range of hyperparameters, including optimizers (Adam, AdamW, SGD with momentum), loss functions (Cross-Entropy, Weighted Cross-Entropy), learning rates (10^{-3}, 10^{-4}), batch sizes (64, 128), and epoch count (5, 10, 15 and early stopping), and various model architectures. The validation set performance was used to select the best configuration for final evaluation. Early stopping was employed to prevent overfitting, with a patience of 5 epochs. The best hyperparameters found for each one of the models are aggregated in Table 2.

4.3 Evaluation Metrics

Given the significant class imbalance in the BloodMNIST dataset, overall accuracy is a misleading metric. A model could ignore minority classes entirely and still achieve a high accuracy score. For this reason, we chose not to use accuracy as a primary evaluation metric. Instead, our analysis focuses on metrics that provide a fair and comprehensive assessment across all classes:

- **Macro-Averaged F1-Score:** This is our primary decision metric. As the unweighted mean of the F1-scores for each class, it treats all classes equally, regardless of their sample size.
- **Confusion Matrix:** Used to visually inspect the model's performance on a per-class basis and identify specific misclassification patterns.

By focusing on the F1-score macro average, we can effectively diagnose whether a model is succeeding on all classes or only on the majority classes.

Table 2: Optimal hyperparameter configurations for the best-performing MLP and CNN models, as determined by performance on the validation set.

Hyperparameter	**Best MLP Configuration**	**Best CNN Configuration**
Architecture	1 Hidden Layer (1024 neurons)	3 Conv Blocks (32, 64, 128 filters) 1 Linear Layer (128 neurons)
Loss Function	Cross-Entropy Loss (unweighted)	Cross-Entropy Loss (**weighted**)
Optimizer	Adam	Adam
Learning Rate	0.001	0.001
Dropout Rate	0.2	0.3
Batch Size	128	128
Epochs	Early Stopping	Early Stopping

5 Results and Analysis

This section presents our findings, organized thematically. We first compare the foundational architectures, then analyze the impact of a different loss function

as our primary strategy for handling class imbalance, we also analyze the effects of the model complexity, and conclude with an error analysis of our best model.

5.1 Comparative Analysis: MLP vs. CNN

The first set of experiments aimed to determine which architecture is fundamentally better suited for this image classification task. As shown in Table 3, the CNN significantly outperforms the MLP.

Table 3: Performance comparison of the best MLP and CNN models on the test set. Both models were trained with a weighted loss function.

Model	Macro F1-Score	Early Stopping Epoch	Training Time
Best MLP	0.845	13	17.1s
Best CNN	**0.938**	9	1m5.7s

The performance gap of over 9% in the Macro F1-score is substantial. In addition to this gap in classification performance, there was a notable difference in training time. The best MLP model completed training in just 17.1 seconds, whereas the more complex CNN model required 1 minute and 5.7 seconds, even though the CNN trained was on fewer epochs. This disparity is expected, as the convolutional operations in a CNN are computationally more intensive than the matrix multiplications of an MLP. This highlights a classic trade-off: the CNN's superior ability to learn spatial features comes at the cost of increased computational resources.

5.2 Impact of Weighted Loss on Performance

This experiment is crucial for demonstrating the effectiveness of our strategy to handle class imbalance. For our best CNN architecture, applying a weighted loss function yielded significant gains, as summarized in Table 4.

Table 4: Comparison of CNN performance with unweighted vs. weighted loss.

Loss Function	Macro F1-Score	Early Stopping Epoch
Standard Cross-Entropy	0.923	9
Weighted Cross-Entropy	**0.938**	9

The Macro F1-score increases by a significant 1.5%. This confirms that the weighted loss successfully improved the model's ability to classify minority classes, which is the primary goal.

5.3 Effect of Model Complexity

We investigated how architectural complexity affects performance and monitored training to prevent overfitting. For the MLP, we found a "sweet spot" in complexity, where increasing model capacity progressively improved performance up to a point, after which returns stagnate, as shown in Figure 2. A similar trend was observed for the CNN. Figure 3 illustrates the training dynamics of our final models, showing that training was stopped effectively before significant overfitting occurred. The CNN's validation loss shows higher volatility compared to the MLP, which is characteristic of a more complex model's sensitivity to batch variations on a small validation set. However, the overall downward trend indicates successful learning and generalization.

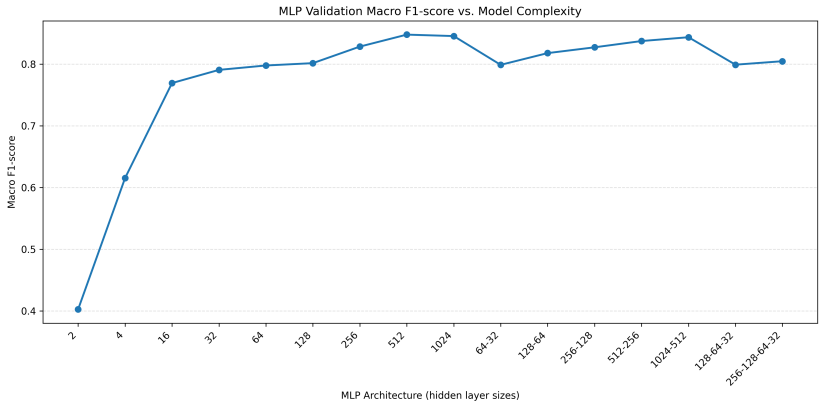

Fig. 2: Line graph of MLP Validation F1-Score vs. Model Complexity. This plot should show the Macro F1-score on the y-axis and different MLP architectures on the x-axis.

5.4 Impact of Optimizer Choice

The choice of optimizer also had a notable impact. For both architectures, the standard **Adam** optimizer provided the best performance in comparison to the Stochastic Gradient Descent with a momentum value of 0.9 and learning rates varying from 0.1 to 0.001 (which yielded a macro F1 score of 0.72 and 0.87 for the best MLP and CNN architectures, respectively). This confirms Adam's robustness for vision tasks.

5.5 Final Model Performance and Error Analysis

Our final, best-performing model is the CNN trained with a weighted cross-entropy loss and the Adam optimizer. Figure 4 shows its confusion matrix on the test set, and a detailed per-class breakdown is provided in Table 5.

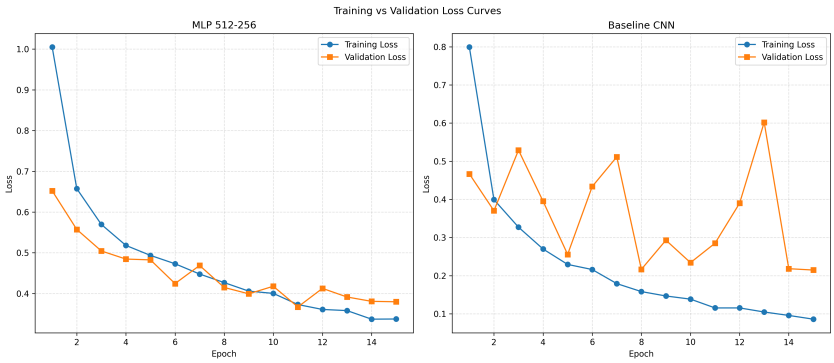

Fig. 3: Training and validation loss curves for the final MLP and CNN models. The validation loss stabilizes, indicating that early stopping prevented overfitting.

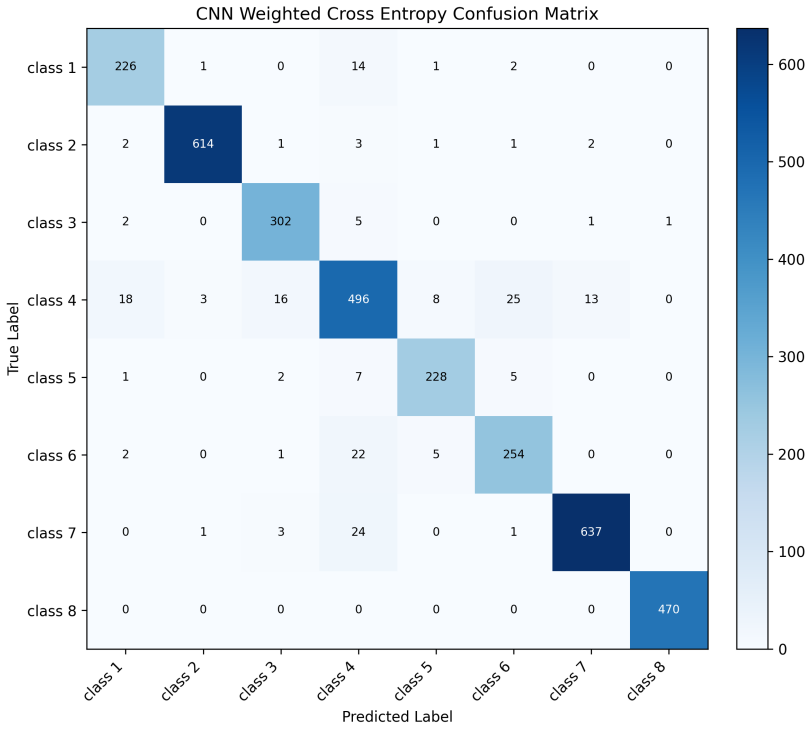

Fig. 4: Confusion matrix for the final CNN model on the test set.

The model achieves excellent F1-scores above 0.90 for most classes. From the confusion matrix, we observe that the most common errors occur when dealing with the 'Immature Granulocytes' class, according to a surveyed medical student, this makes sense, as that class is just an umbrella term for cells in a long maturation process, where each stage of the process is very visually distinct. This can be verified by looking at some samples from that class in Figure 5.

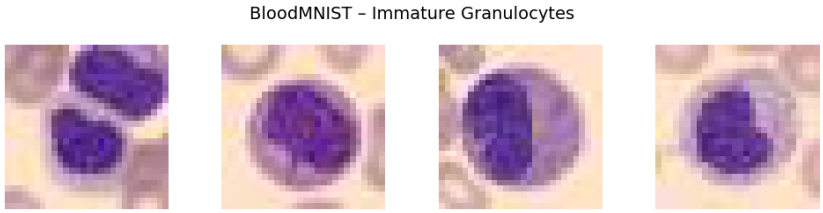

Fig. 5: Examples from the 'Immature Granulocytes' class, confirming the visual disparities between the different instances of this class.

We also observe that the two rarest classes, 'Basophil' and 'Platelet' also achieved competent F1-scores, suggesting that the use of a weighted loss function mitigated the class imbalance problem.

Table 5: Per-class performance of the final CNN model on the test set.

Cell Type	Precision	Recall	F1-Score
Basophil	0.91	0.90	0.90
Eosinophil	0.98	0.99	0.98
Erythroblast	0.98	0.93	0.95
Immature Granulocyte	0.90	0.82	0.86
Lymphocyte	0.98	0.91	0.94
Monocyte	0.83	0.93	0.88
Neutrophil	0.92	0.99	0.95
Platelet	0.99	1.00	0.99

6 Conclusion

This project successfully developed and evaluated deep learning models for blood cell classification on the unbalanced BloodMNIST dataset. Our results demonstrate that a Convolutional Neural Network (CNN) is significantly superior to a Multi-Layer Perceptron (MLP) for this image-based task, achieving a final Macro F1-score of 0.938. We further established that employing a class-weighted

loss function was a critical strategy, particularly for the CNN, as it ensured fair performance across all classes, including rare ones, and improved the Macro F1-score compared to an unweighted approach.

References

1. J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "MedMNIST v2 - A large-scale lightweight benchmark for 2D and 3D biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, Jan. 2023.