POLITECNICO
MILANO 1863

AN2DL - First Homework Report
Bio.log(y) Team

Luca Lepore, Arianna Rigamonti, Michele Sala, Jacopo Libero Tettamanti

hif1a, aririgamonti, micheles, jacoposheep

252309, 252321, 252325, 252329

November 25, 2024

1 Introduction

The first homework assignment focuses on a multi-class image classification problem using deep learning techniques. The goal is to develop a robust model able to accurately classify blood cells images into their respective categories. To address this task, we adopted a stepwise strategy, starting from a simple baseline model and incrementally increasing complexity to assess the impact of each enhancement on performance.

2 Problem Analysis

The dataset consists of 13,759 RGB images (96x96 pixels) of blood cells categorized into eight different classes, each representing a specific cell type: Basophils, Eosinophils, Erythroblasts, Immature granulocytes, Lymphocytes, Monocytes, Neutrophils, and Platelets. The dataset shows class imbalance and identical images that had to be be taken into account. The most important features to distinguish blood cells are the overall dimension, nucleus-to-cytoplasm ratio, the number of nucleus lobes, its form and position and the presence and size of granules [1]. The background, dominated by the presence of erythrocytes, does not provide meaningful features for distinguishing cell types. When smeared onto slides, blood cells can exhibit a wide rage of variability. They may appear in any orientation, some may be partially cutoff within the microscope's field view, and their size depends on the zoom setting. Additionally, staining may be uneven or poor, cells can become elongated or damaged, and their shape may vary depending on the pathological state of the patient [1]. To simulate these variations, data augmentation techniques are essential. Furthermore, pretrained models from Keras Applications were implemented to improve model generalization reducing the computational time for the training process.

3 Methods

3.1 Data Preprocessing

The first step was to inspect the dataset to understand the type of images provided. This revealed the presence of many duplicate images that were removed to minimize the risk of overfitting. To achieve this, we used the **Perceptual Hashing (pHash) algorithm**, which generates a unique hash based on the image's visual content using Discrete Cosine Transformation (DCT) [2]. As this hash value acts as a digital fingerprint, images with identical hashes were considered duplicates, with only one copy retained. This reduced the dataset size from 13,759 to 11,953 images (1,806 duplicates). Pixel values were preserved in the 0–255 range without normalization. Following this, the dataset was

split into training (60%), validation (20%), and test sets (20%).

3.2 Data Augmentation

Data augmentation was a key part of our preprocessing pipeline, designed to increase dataset size and variability, reducing the risk of overfitting and improving generalization. The augmentation techniques we considered were RandAugment and AugMix. **RandAugment** applies a predefined number of random transformations, such as rotations, and flips, to each image. The number of transformations per image is controlled by the n parameter and their intensity by the magnitude parameter m. For training and validation sets, we defined $n = 3$ and $m = 0.7$, whereas for the test set, we defined $n = 4$ and $m = 1.0$. **AugMix** generates multiple augmentation chains, linearly combines their outputs, and mixes them with the original image. For training and validations sets, we defined $k = 3$ and $m = 0.7$, whereas for the test set we defined $k = 4$ and $m = 0.8$. The mixing weights w_j were uniformly distributed. Through this process, the training and validation sets were expanded to 3× the original size, whereas the test set was expanded to 4×. Actually, we used two versions of the test set: one with the original images and the other with the augmented ones. Original images in the training and validation sets were excluded to prevent bias instead.

3.3 Models

3.3.1 Baseline & Regularised CNNs

Our **baseline model** was a simple Convolutional Neural Network (CNN) composed of two blocks, each containing a Conv2D layer with ReLU activation function followed by a MaxPooling2D layer. Before the output layer, a single Dense layer was included. Then, we developed its **regularised version**, introducing Batch Normalization and Dropout layers within both blocks (Dropout = 0.3) and after the Dense layer (128 neurons, Dropout = 0.4).

3.3.2 Pretrained Models

As the next step, we explored pretrained models to implement Transfer Learning. The models we selected include MobileNetV3, EfficientNetV2M, ConvNeXtBase, VGG16, chosen among the options provided by Keras Applications. The initial selection criteria included top-1% accuracy, model size, and inference speed. To test the complexity of our dataset, we started with simpler pretrained models and gradually moved to more complex ones, searching a compromise between accuracy and size. For each model we followed a standardized pipeline comprising two phases: an initial **feature extraction** phase followed by a **fine-tuning** phase. Fine-tuning was initially performed unfreezing only the final layers (depending on the model) with a reduced learning rate of 10^{-4}. Then we retrained the entire model, excluding the BatchNormalization layers, with a further lowered learning rate of 10^{-5}. The same pipeline for each model was executed with three empirically determined parameter configurations (table 1).

Table 1: Configurations tested to choose the best performing models

Parameters	Cfg. 1	Cfg. 2	Cfg.3
N. Dense*	0	1	2
N. Neurons	N/A	512	First: 512 Second: 256
Batch Norm.	N/A	Applied	Applied
Dropout	0.3	0.3	First: 0.3 Second: 0.3
Learning rate	10^{-3}	10^{-3}	10^{-3}

*N. Dense refers to the number of Dense layers before the output layer.

3.3.3 Hyperparameter Tuning

Subsequently, we selected the models with the best performance on the augmented test set and proceeded with hyperparameters tuning using the **Hyperband algorithm** [3] from Keras Tuner. We focused on optimizing combination of Dense layers, the number of neurons, the Dropout rate, and the learning rate prior performing fine-tuning. Hyperparameter tuning is a resource-intensive and time-consuming process, which is why we applied it to the best performing models and did not consider to tune additional parameters. Finally, we performed again feature extraction and fine-tuning with the new parameters.

4 Experiments

The experimental process involved evaluating each model's performance on the test sets and the Cod-

abench platform. For each model, we considered the results from its best configuration (Table 2).

Table 2: Performance Summary of Different Models

Model	**Cfg.**	**Test (%)**	**Aug Test (%)**	**Codabench (%)**
Baseline CNN	-	57.9	30.9	19
Regularized CNN	-	79.9	42.2	30
MobileNetV3	3	92.5	62.2	70
VGG16	1	96.4	71.0	80
EfficientNetV2M	**3**	**98.4**	**80.1**	**92**
ConvNeXtBase	**1**	**98.1**	**80.0**	**92**

Following this, we performed hyperparameters tuning using the Hyperband algorithm on the best performing models (table 3).

Table 3: Hyperparameter Tuning Results

Model	Dense Layers	Dropout (per Layer)	Learning Rate
EfficientNetV2M	2 (384, 128)	0.1, 0.1	0.001
ConvNeXtBase	1 (128)	0.3	0.001

The results of the new transfer learning process with the optimal parameters are shown in table 4.

Table 4: Post-Tuning Results for the Best Models

Model	Val (%)	Aug Test (%)	Codabench (%)
EfficientNetV2M	**89**	**78**	**91**
ConvNeXtBase	89.6	76.8	90

5 Results

The best results in terms of accuracy on the test sets were obtained after the final stage of fine-tuning, where all layers were unfrozen. The application of a pretrained model through transfer learning significantly outperformed the baseline CNN (accuracy: 80% vs. 31% on our augmented test set). Furthermore, even with hyperparameter tuning, the best absolute results on the Codabench test set were achieved by ConvNeXtBase (92%, Cfg. 1) and EfficientNetV2M (92%, Cfg. 3).

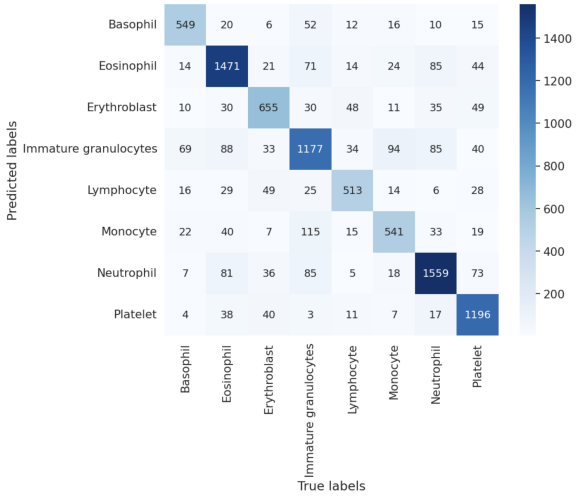

Figure 1: Confusion matrix for the best version of EfficientNetV2M on the augmented test set.

6 Discussion

Thanks to robust data augmentation techniques and transfer learning, we achieved optimal results on both our test sets and the Codabench test set. However, the best performances were achieved with the empirically chosen configurations and not the hyperband algorithm. This could be due to the algorithm not converging to a global minimum or the limited computational resources preventing the tuning of all parameters. For ConvNeXtBase, a second hyperparameter tuning also included weight decay (AdamW optimizer), but it did not improve performance, achieving 90% accuracy on the Codabench test set.

7 Conclusions

In this project, we developed an image classification model, but there are several areas where the model's performance could be refined. Future improvements could include experimenting with longer training patience and refining the number of frozen convolutional layers. Additionally, exploring a wider range of data augmentation strategies could enhance the model's generalization capabilities and improve its performance by further expanding the training set. Finally, future work could focus on exploring alternative architectures, more advanced regularization techniques, and integrating more diverse datasets to increase the model's robustness.

Additional Materials

Each group member's contribution as well as all runned notebooks and codabench submissions are available at this link

References

[1] Vinay Kumar, Abul K. Abbas, and Jon C. Aster. *Robbins & Cotran Pathologic Basis of Disease*, chapter 13 Diseases of White Blood Cells, Lymph Nodes, Spleen, and Thymus; chapter 14: Red Blood Cells and Bleeding Disorders, pages 579–668. Elsevier, Philadelphia, PA, 9th edition, 2015.

[2] P. Subudhi and K. Kumari. A fast and efficient large-scale near duplicate image retrieval system using double perceptual hashing. *Signal, Image and Video Processing (SIViP)*, 18(12):8565–8575, 2024.

[3] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.