AN2DL - First Homework Report
Hidden Players

Agnese Negroni, Alberto Pola, Fabio Romagnoli, Niccolò Signorelli

February 23, 2025

1 Introduction

In the medical field, artificial neural networks, and more specifically convolutional neural networks, have facilitated significant progress in the detection and classification of abnormalities. This project aims to develop a Convolutional Neural Network (**CNN**) capable of classifying microscope images of eight different types of blood cells. We leverage deep learning, image analysis, and multi-class classification techniques to design a robust CNN capable of categorizing blood cells with high accuracy. Initially, our approach was guided by a few scientific articles we found online, which proved to be instrumental in helping us take the first steps ([1], [4]).

In the following analysis, the scores listed in the column "Cb acc." (Codabench[1] accuracy) represent the models' accuracy when evaluated on a **highly corrupted dataset**, which was used by the professors to assess the performance of our models. The nature of this dataset was revealed only after the end of the challenge.

Throughout the entire challenge, we used Python and several of its libraries, with TensorFlow being the main tool for all the deep learning tasks.

2 Problem Analysis

The problem involves the classification of 96x96 RGB images of blood cells belonging to eight different classes, as shown in Figure 1. While the images in the dataset appear highly regular and exhibit minimal variance, further research revealed that, in real-world applications, blood cell images often display significant variability. This insight, combined with the poor results obtained from our first models trained on the basic dataset, reframed the problem by emphasizing the importance of building a model that is not only accurate in classifying the available dataset but also robust and reliable when applied to more diverse datasets. Specifically, the goal was to develop a model that is **domain invariant**, ensuring its effectiveness across varying image characteristics. This objective led us to dedicate a significant effort to the process of **data augmentation**.

The initial dataset consisted of 11,959 images and exhibited class imbalance. To address this problem, we created a balanced training set through data augmentation, rather than using class weights in the loss function, as the latter method did not yield satisfactory results.

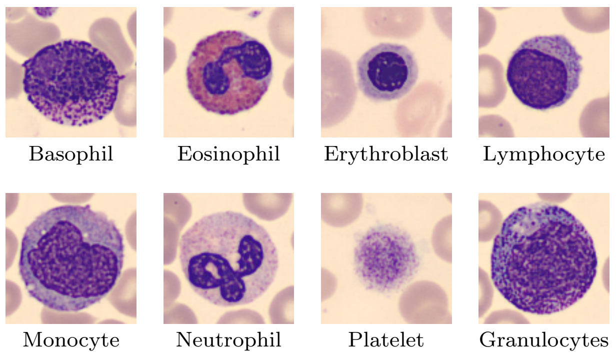

Figure 1: Overview of Blood Cell Types

[1]This is the name of the platform that was used for the challenge.

3 Method

To establish a benchmark for our models and verify their accuracy in classifying images from a diverse domain, we created a test set with randomized augmentations (Bmk). This test set was used to evaluate the generalization ability of our models throughout the challenge and allowed us to validate their accuracy locally.

Various data augmentation techniques exist, including geometric transformations, color adjustments, noise-based modifications, and more. These techniques are widely used to improve a model's ability to classify images across diverse styles, lighting conditions, textures, and resolutions.

Due to computational and time constraints, building and training an effective model from scratch was beyond our resources, as demonstrated by some initial tests. Therefore, we adopted **transfer learning**, a technique that leverages pre-trained models with proven classification performance. In particular, we utilized the weights of the initial layers of these models, which are responsible for the complex task of large-scale feature extraction. Subsequently, through **fine-tuning**, we adjusted the weights of selected later layers to adapt the model to our specific task.

Our focus was on two pre-trained architectures: **ResNet** and **EfficientNet**. This choice was guided by our initial tests and supported by research in papers such as [1] and [4]. These architectures outperformed alternatives like VGG, DenseNet, and MobileNet in terms of accuracy and overall performance.

4 Experiments and Results

4.1 Data Augmentation

Having identified generalization as our main challenge, we focused heavily on data augmentation. This process involves modifying images, either through a single transformation or a sequence of transformations applied in a pipeline.

Our initial approach was inspired by [2]. Multiple datasets were created by applying specific transformations to every image in the original dataset. These datasets were then concatenated into groups based on the type of transformation applied and used to train a custom small CNN for comparison. The results of these experiments are summarized in Table 1, in the next paragraph. While these results were promising, the actual models still struggled to generalize to the previously unseen augmented data of the Codabench dataset.

This prompted us to progressively enhance our augmentation strategies by applying more complex and aggressive transformations. A summary of the efficacy of these transformations on pretrained models is presented in the next paragraph, specifically in Table 2. The key insight from this approach was that transfer learning demonstrated greater robustness in extracting essential features for our classification problem, even when working with increasingly transformed images.

Moving forward, we decided to adopt more randomized types of data augmentation. We leveraged the KerasCV library ([3]), a supplementary library for computer vision that is integrated with Keras. Using this library, we developed our final data augmentation pipeline, which incorporated functions such as **RandAugment**, **AugMix**, and others. A scheme of the main pipeline is illustrated in Figure 2. The subsequent models were all trained on two copies of the original datasets passed through the pipeline, along with an additional copy that was augmented with the functions **CutMix** and **MixUp**.

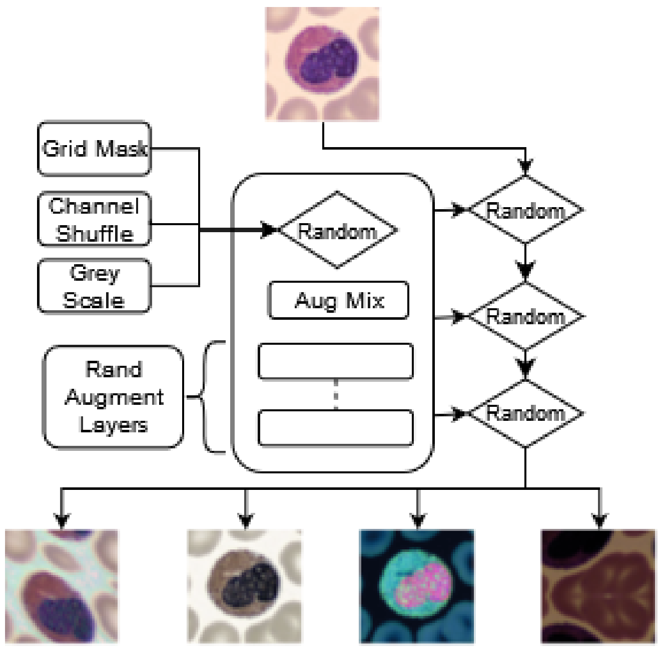

Figure 2: KerasCV Pipeline to diversify images

4.2 Model Selection

Regarding the choice of the model, we initially built a custom Convolutional Neural Network to familiarize ourselves with the TensorFlow/Keras library. This model was trained on the original dataset using various forms of slight augmentation. Despite making repeated modifications to the network's architecture in an attempt to improve performance, the results remained suboptimal, as illustrated in Table 1.

Table 1: A1: flip, scale; A2: rotate, translate, shear; A3: contrast, brightness, color shift

Aug.	Val. acc.	Bmk acc.
A1	0.955	0.204
A2	0.987	0.196
A3	0.991	0.385

Given the poor accuracies, we decided to transition to models based on pre-trained architectures, training them on a more extensively augmented dataset, as discussed in the previous paragraph. During this phase, no fine-tuning was applied, as we aimed to compare their raw capabilities in order to make a choice. The results of their evaluation are shown in Table 2.

Table 2: Goemetric, chromatic and blurring transformation together

Backbone	Val. acc.	Bmk acc.
VGG16	0.904	0.254
ResNet50	0.943	0.601
MobileNet	0.925	0.552
EfficientNetB5	0.921	0.580

Observing these results, we decided to perform fine-tuning on the **ResNet** and **EfficientNet** architectures. For each architecture, we trained multiple versions with different hyperparameter configurations, leveraging also the Keras Tuner package for this task. The main results of this new trial are summarized in Table 3.

Table 3: Tuned models' performance on augmented datasets with two final dense layer with 1024 and 128 neurons.

Backbone	Val. acc.	Bmk acc.
ResNet50	0.963	0.677
EfficientNetB5	0.978	0.789

Noticing an improvement in performance, we proceeded to train different versions of these models on the output of the augmentation pipeline shown in Figure 2. The key results from this phase are summarized in Table 4.

Table 4: Final models performance

Backbone	Val. acc.	Bmk acc.	Cb acc.
ResNet50	0.924	0.931	0.88
ResNet152	0.938	0.940	0.89
EfficientNetB6	0.935	0.945	0.90

As a final attempt, we decided to create an ensemble of the best-performing versions of our **ResNet50**, **ResNet152**, and **EfficientNetB6**-based models (the ones outlined in Table 4), aiming to achieve an even better performance by combining their prediction capabilities. As anticipated, the ensemble delivered slightly improved results, achieving a score of **0.917** on Codabench. After the end of the challenge, the Codabench test set was revealed to us. A sample of its images is shown in Figure 3.

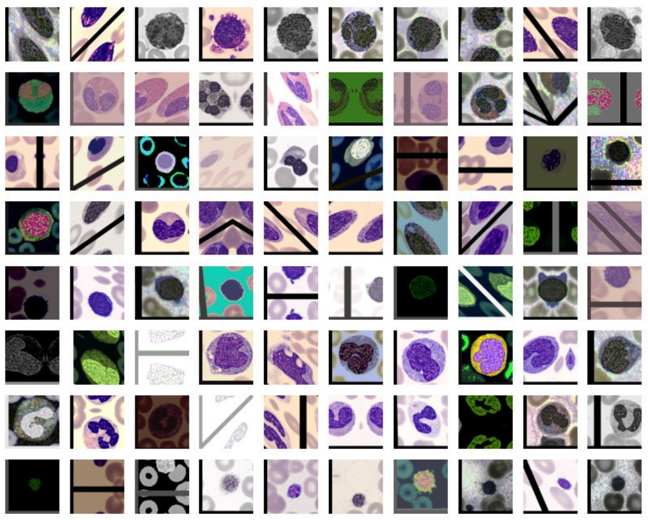

Figure 3: Sample of Codabench test set.

4.3 Optimizer

Throughout the whole challenge, we used the *Adam* optimizer as our reference optimization algorithm, but we also experimented with other optimization algorithms such as *SGD* and *Lion*. However, these attempts yielded either no significant improvements (SGD) or even worse results (Lion, which converged too quickly). As a result, we decided to continue using **Adam** as our primary optimizer.

5 Conclusions

Through the development of this project, we understood the crucial role of data augmentation in training robust Image Classifiers. Moreover, having limited computational resources, we could see how important it is to leverage pre-trained architectures and how to tailor them to a specific problem. In particular, we observed that unfreezing too many layers of the pre-trained backbones can lead, at some point, to a decrease in performance.

References

[1] R. Asghar, S. Kumar, and P. Hynds. Automatic classification of 10 blood cell subtypes using transfer learning via pre-trained convolutional neural networks. *Informatics in Medicine Unlocked*, 49:101542, 2024.

[2] L. Nanni, M. Paci, S. Brahnam, and A. Lumini. Feature transforms for image data augmentation. *Neural Computing and Applications*, 34(24):22345–22356, Aug. 2022.

[3] L. Wood, Z. Tan, I. Stenbit, J. Bischof, S. Zhu, F. Chollet, D. Sreepathihalli, R. Sampath, et al. Kerascv, 2022.

[4] Z. Zhu, Z. Ren, S. Lu, S. Wang, and Y. Zhang. Dlbcnet: A deep learning network for classifying blood cells. *Big Data and Cognitive Computing*, 7(2), 2023.