AN2DL - First Homework Report
AwesomeChallengers

Sara Jo Agostino, Andrea Faccioli, Giacomo Falcone

sarajoagostino, andreafacciolii, giacomofalcone

280760, 271316, 271005

January 10, 2025

1 Introduction

Developing efficient methods to identify and categorize different cell states is crucial for advancing human health outcomes. For instance, it can be vital for disease prevention and early diagnosis, as well as for monitoring disease progression and treatment response. This project is focused on a **multi-class classification** problem: the goal is to classify blood cell images into **8 different classes** [Basophil (0), Eosinophil (1), Erythroblast (2), Immature granulocytes (3), Lymphocyte (4), Monocyte (5), Neutrophil (6), Platelet (7)] using **deep learning** techniques.

2 Problem Analysis

Our aim is to train a model that can accurately predict the class of an image. Formally, we have to learn a function f such that *prediction=f(image)*. Furthermore, the performance metric employed is **accuracy**, defined as the ratio of correct predictions over the total number of predictions made (equal to N, the size of the test set).

2.1 Data Inspection

The given dataset contains 13759 samples. Each image has a size of 96×96 pixels, represented in RGB channels. First of all we analyzed the dataset in order to identify outliers and duplicate images. Indeed, by removing noise and redundancy from the data, we can work with a more representative dataset, enabling models to learn effectively and make more accurate predictions. We found and removed 1806 duplicates from the dataset, including the following 2 outliers:

Figure 1: Outliers

Additionally, we noted that visual similarity among cell images may presents challenges due to inter-class resemblance and intra-class variability in shape and size. Using a pre-trained EfficientNetB0 to extract features and compare them between images, we identified 116 similar image pairs ($\geq$ 95% similarity) but found no significant imbalance or bias in the dataset. Thus, we left it unchanged.

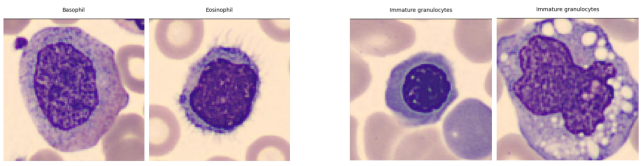

Figure 2: Inter-class similarity and intra-class variability

Returning to the issue, the new dataset now contains 11951 samples. At our first attempts, we split it in training, validation and test set, in the ratio: training set 9559 (80%), validation set 1196 (10%) and test set 1196 (10%).

Then we examined the samples distribution among the classes and we noticed that some blood cells were less represented in the dataset:

Class	Occurences
Basophil	680
Eosinophil	1743
Erythroblast	867
Immature granulocytes	1618
Lymphocyte	679
Monocyte	794
Neutrophil	1864
Platelet	1314

Table 1: Training set occurrences per class

3 Method

3.1 Class Imbalance

We addressed the issue of class imbalance by trying two different techniques: **class weighting** technique (*it gives adequate importance to the less frequent classes during training*) and a **weighted image augmentation** (to bring the number of occurrences per class to the level of the maximum represented one).

3.2 Choosing the Right Model

Even knowing that the way of transfer learning could be probably take us to a better results, we decided to use a bottom-up approach by starting with some simple **custom CNN**. Only after a couple of days of experimentation we decided to opt for **transfer learning**.

3.3 Regularization

Another crucial problem was **regularization**. This is a critical component in training deep learning models, that aims to mitigate **overfitting**. *Overfitting occurs when a model becomes too specialized in learning patterns from the training data, so this leads to poor performances in unseen data.* Regularization helps the model generalize better by introducing constraints or modifications during training. In our models we tried different regularization techniques: class weightening, image augmentation, **early stopping** (patience numbers in range 10-20), **dropout** (dropout rate in range 0.3-0.4), L1-L2 regularization, global average pooling.

4 Experiments

4.1 Class Imbalance

We experimented with different combinations of the two techniques described in section 3.1. Using only class weighting could limit the model's ability to learn from underrepresented classes effectively. On the other hand, relying solely on data augmentation might excessively alter the features of some classes compared to others that are less augmented.
To equalize class occurrences we tried different settings of AugMix [2] and at the end we opted for a small severity to preserve information of original images, a high alpha and a small number of chains to avoid overlapping images that may guide to an unclear overlapped image.

4.2 Custom CNN

In the early stages of the development phase we implemented a simple convolution neural network from scratch, starting from 3 convolutional layers followed by ReLu activations and fully connected layers for classification. We trained this model many times, tuning the depth of the network and other parameters (e.g. kernel size, padding). This approach led us to a fairly good performance on the public dataset ($\simeq$ 80% accuracy), while turned out to be weak on Codabench hidden set ($\simeq$ 20% accuracy). This behaviour made us thinking about the importance of data augmentation: it was clear that our model was **overfitting** the training data and wasn't able to generalize on a different set. Simple augmentation layers (e.g. RandomZoom, RandomRotation, RandomFlip) slighty improved the performance ($\simeq$ 35% accuracy).

4.3 Augmentation and Transfer Learning

In the following days we started a parallel investigation of advanced data augmentation techniques and transfer learning methods.
We decided to exploit the power of pre-trained models, available in Keras Applications. Each of us examined a different famous neural network architecture: ResNet, MobileNet and Xception. After few hours of testing we decided to gather our efforts on **MobileNetV3** [1] family because it provided us with a good compromise between accuracy and computational complexity.
With a fixed network architecture, we conducted a bunch of experiments on different types of data augmentation. We scanned `Keras CV` library augmentation layers to identify the ones that could have had the better impact on

Table 2: Accuracy of our relevant models.

Model	**Accuracy** (local test)	Precision (weighted)	Recall (weighted)	F1 (weighted)	**Accuracy** (Codabench)
Custom CNN	81	-	-	-	35.4
MobileNetV3Large	97.66	97.68	97.66	97.65	65
ConvNeXt	98.24	98.26	98.24	98.25	77

the classification robustness. After few tests we decided to preprocess our training set by applying a `RandAugment` [3] transformation to the images. RandAugment randomly chooses an operation over a set of them and applies it to the image. This behaviour helped us to tackle the fact that we couldn't know how the hidden test set was processed. We played around with its parameters (e.g. augmentations_per_image, magnitude) as we wanted to avoid an excessive alteration of the original images.

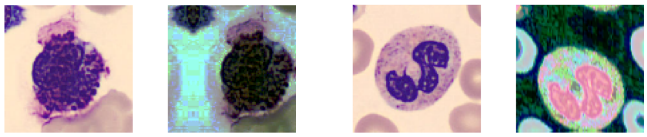

Figure 3: RandAugment(#augm=5, magn=0.2)

4.4 Optimizer Choice

We decided to use **Lion** [4], because it combines computational efficiency with faster convergence. Moreover, it is expected to achieve a better generalization compared to Adam.

4.5 Final model

After several experiments with different data augmentation strategies, as the results show limited improvement ($\simeq$ 65% accuracy), we decided to switch to a more powerful pretrained model. We selected **ConvNeXt** [5] because it offers a superior performance on complex image classification tasks. We immediately get a better performance (73% accuracy) even keeping the augmentation part simple (so without RandAugment) and by balancing classing trough a class weightened function during training.

5 Results

From table 2 is quite clear how the main problem was regarding generalization. To face this problem, we developed a final model (achieving 77% accuracy) finding a trade-off between the amount of available data and possible distortions in the data through weighted image augmentation. We opted for an arbitrary selection of the number of transformed images to add to the minority classes, preserving the weight balancing mechanism.

6 Discussion

While class weighting and augmentation techniques addressed data imbalance effectively, the introduction of severe augmentations sometimes led to feature distortions, impacting classification for specific cell types. Such as the model was focusing on less informative regions of images (e.g. edges) for certain predictions.

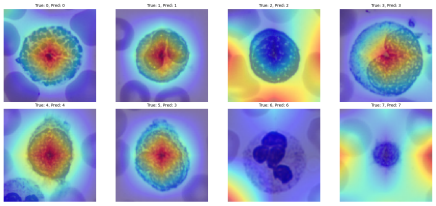

Figure 4: CAM of MobileNet

The choice of ConvNeXt provided a significant performance boost, showcasing its ability to handle complex visual tasks. However, limitations persisted, computational costs for training that larger architectures posed practical challenges, emphasizing the need for efficient augmentation and regularization strategies.

7 Conclusions

In this project, we successfully tackled the multi-class classification of blood cell images using by leveraging ConvNeXt and advanced augmentation techniques. However, further improvements, such as targeted augmentations (e.g. center cropping and Gaussian blurring for the edges) and optimization of feature extraction, could enhance performance.

References

[1] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. Le, H. Adam. **Searching for MobileNetV3**. link, 2019. Accessed: 2024-11-13.

[2] D. Hendrycks, N. Mu, E. Cubuk, B. Zoph, J. Gilmer, B. Lakshminarayanan. **AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty**. link, 2019. Accessed: 2024-11-14.

[3] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, Quoc V. Le. **RandAugment: Practical automated data augmentation with a reduced search space**. link, 2019. Accessed: 2024-11-14.

[4] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C. Hsieh, Y. Lu, Q. Le. **Symbolic Discovery of Optimization Algorithms**. link, 2023. Accessed: 2024-11-16.

[5] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, S. Xie. **A ConvNet for the 2020s**. link, 2022. Accessed: 2024-11-19.

Contributions

Sara Jo Agostino:

- Dataset analysis - Balancing
- Dataset augmentation
- Transfer learning and fine-tuning on MobileNet and ConvNextLarge

Andrea Faccioli:

- Custom CNN training
- Transfer learning and fine-tuning on Xception and MobileNet

Giacomo Falcone:

- Dataset inspection (finding outliers and similarity analysis)
- MobileNet training with various augmentation
- Transfer learning and fine-tuning on InceptionV3 and EfficientNetB4
- Applied CAM analysis on MobileNet