AN2DL - First Homework Report
DL Bois

Alessandro Lorenzini, Manuela Benini, Daniel Giuseppe Boi

Codabench alelore6, Codabench manubenini25, Codabench danielboi

273343, 273307, 278900

November 24, 2024

1 Introduction

In this project, the aim was to create and train a neural network model, able to classify images of eight different types of blood cells with the **highest possible accuracy**.

2 Problem Analysis

Dataset characteristics

The first thing we did was analyze the given dataset to check the presence of outliers, corrupted images and duplicates.
To find trivial outliers, we first rapidly scanned the images of the dataset removing non unique images and more than a thousand of "Shreks" and "Rick Rolls".
Next, we used, as shown in figure 1, the Mahalanobis distance to find non-trivial outliers to get the final and cleaned dataset.

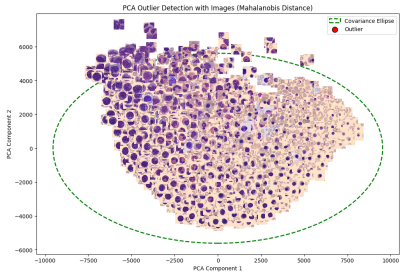

Figure 1: Distribution of the images with respect to the Mahalanobis distance

At this stage, we analyzed the distribution of the classes in the dataset, finding out that it was slightly unbalanced, as shown in figure 2. We simply faced this problem by calculating weights of each class to balance their importance in the training process.

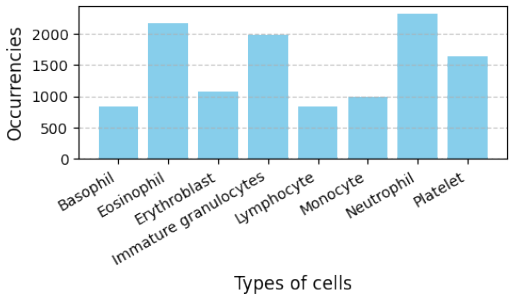

Figure 2: Graph of the classes distribution in the dataset

Main challenges

Since the beginning, the main problem we've encountered was the **discrepancy** between the accuracy obtained on our validation set and the one obtained on the test set provided. We attributed this issue to the fact that our model had **overfitted** the dataset and failed to generalize properly.
In order to overcome the problem, we used various techniques, such as *Data Augmentation*, *Transfer Learning* and *Fine Tuning*, which will be explained better in section 3. Moreover, another fundamental aspect was changing hyper-parameters to optimize

the model to this particular scenario, i.e., the classification of blood cells.

Initial assumptions

At the beginning, we implemented a very simple model: we started by splitting the dataset 80% for training and 20% for validation. We proceeded by normalizing data and setting the learning rate at 10^{-3}.
Next, we built the neural network layer by layer: 2 convolutional layers, 2 max pooling layers, ReLu as activation function, all of this followed by flatten and dense layers, where it's also been applied the early stopping.
With this model, we reached an accuracy of 80% circa on the validation set, but on the test set we only obtained 25%. We, thus, understood that more than a thing needed improvement and we started to work on our CNN. We implemented cross validation in order to verify if the problem was related to the particular splits of our dataset or to the discrepancy between ours and the one used in the test phase. What we noticed is that we reached always the same amount of accuracy on validation sets with a standard deviation close to 0 on all the fold splits we tried (3, 5 and 10 fold splits). Therefore, our conclusion was that the model was overfitting on our data and it wasn't capable of generalizing and making the correct predictions on the test set, so we decided to try different strategies to face this problem.

3 Our procedure

At first, we introduced dropout, in order to limit overfitting by switching off some neurons, and also L2 regularization. Furthermore, we added a data augmentation pipeline with some basic transformations, such as translation, rotation, zoom and brightness variation. Eventually, we inserted another convolutional layer and the accuracy increased on validation set but we didn't see any noteworthy improvement on test set.
Then, we decided to try **transfer learning**: at first, we used VGG16 and we immediately observed big improvement on accuracy (43% on test set), then we tried to use more recent and accurate models to maximize the result. All the scores we collected can be seen[1] in table 1.
While we were training all these pre-trained models, we also introduced some adjustments. We tried the L1-L2 regularization and we noticed a small improvement on accuracy but with an higher loss, hence actually nothing able to justify the choice of this over L2, so we came back to it.

Table 1

Model	Accuracy on test set
VGG19	62%
EfficientNetB2	**74%**
EfficientNetB3	67%
ResNet50	58%
Xception	65%

We built a more aggressive and specific data augmentation, adding to the already mentioned techniques the followings:

- *random erasing*: technique that puts small black rectangles on images in order to help the model to generalize.
- *changes in contrast and color*
- *random noise*

We, at first, implemented them using the OpenCv library but then, since the training was very slow (CPU side), we switched to TensorFlow (GPU side). Another modification was changing the optimizer from Adam to AdamW (with `weight_decay` at 10^{-4} at first and then 10^{-5} during the fine tuning), always with the aim of a better generalization.
At the end of all of these modifications, we reached an accuracy of 62% on test set, still pretty much far from the one obtained on our validation set (around 90%).
Finally, we decided to add **Fine Tuning**, unfreezing the last layers in order to make our model learn more specific characteristics. We, thus, trained just the "unlocked" part of the model with a lower learning rate (using also learning rate scheduler), because we didn't want to change so much the already acquired knowledge. We also lowered the number of epochs (about 30). In particular, we put as trainable our

[1] These are the accuracies obtained with the final model, including the use of Fine Tuning. We also tried ConvNextBase, but we didn't submit it because the accuracy was oscillating a lot.

custom layers[2] and the block 7 of EfficientNetB2. We tried to unfreeze even more layers, but at the end this was the best solution we found.
Actually, we expected a better improvement after applying Fine Tuning, indeed we obtained 74% of accuracy on final phase. Since on our validation set we reached 97% of accuracy, we hypothesized that the model went in overfitting during fine-tuning, but we truly didn't manage to overcome this obstacle.

4 Experiments

During our work, we constantly did a lot of tests changing parameters (see table[3] 2). The hyper-parameters that best fit our model are situated in the last column.

Table 2

Parameter	Range	Value chosen
Dropout rate	[0.2,0.5]	0.4
l2 lambda	[0, 10^{-3}]	10^{-3}
Learning rate	[10^{-1}, 10^{-5}]	10^{-4}
# of dense layers	[1,3]	1
Batch size	[16, 256]	16
Patience	[2,20]	15

Monitoring precision for every class, we noticed that it had values over 90%, except for monocytes, basophil and immature granulocytes, which were at about 60%. This happened in particular with ConvNextBase. To not make our model confuse these types of cells, we decided to increase contrast and also choose the model with the highest precision for each class, i.e., EffcientNetB2.

Table 3: Metrics

Type of cell	Precision	Recall	F1	Support
Basophil	0.86	0.98	0.92	108
Eosinophil	0.97	0.96	0.97	278
Erythroblast	0.98	0.94	0.96	138
Imm. granulocytes	0.92	0.81	0.86	253
Lymphocyte	0.92	0.96	0.94	108
Monocyte	0.81	0.90	0.86	126
Neutrophil	0.93	0.94	0.94	297
Platelet	0.99	1.00	0.99	210

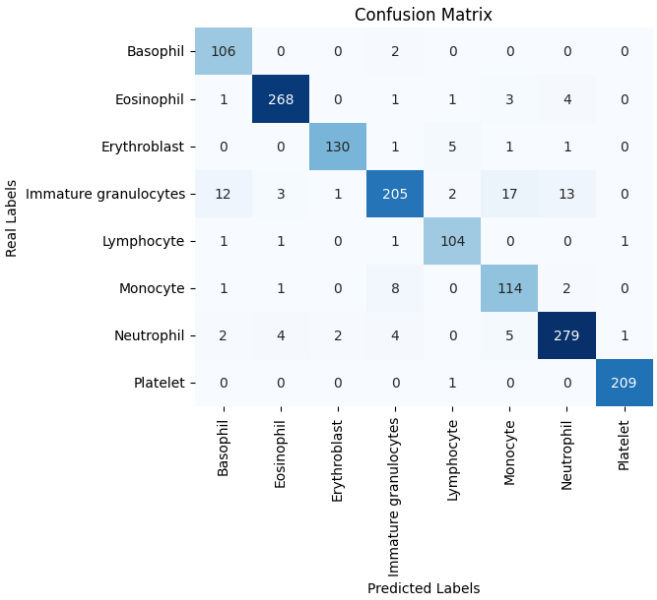

Figure 3: Confusion matrix

5 Conclusions

The contributions of each team member to the final work are:

- **Base model:** Manuela Benini
- **Data Augmentation:** Alessandro Lorenzini, Manuela Benini
- **Cross Validation:** Daniel Giuseppe Boi
- **Transfer Learning:** at first implemented by Alessandro Lorenzini, then all of us worked on two of the six written pre-trained models
- **Fine Tuning:** we worked all together to find the optimal configuration

Apart from this, we always cooperated and worked as a team in every aspect of the project: we at first developed a very simple model and then we proceeded together increasing step by step the complexity of our network and tuning our hyper-parameters, reaching the final result (see table 2).

[2]We put 256 neurons in the dense layer
[3]The range represents the interval of the values we used for that specific parameter)

References

[1] M. Matteucci. Lecture notes. PowerPoint presentation, 2024. Available here.

[2] G. Boracchi. Lecture notes. PowerPoint presentation, 2024. Available here.

[3] I. Rojas, Olga Valenzuela, F. Rojas, L. J. Herrera, F. Ortuño. Bioinformatics and Biomedical Engineering. From pag 727 to 738.