POLITECNICO
MILANO 1863

AN2DL - First Homework Report

8pesto

Elisa Nordera, Sara Redaelli, Sveva Zanetti, Rachele Zanin

eluuu, sararedaelli, svevazanetti, rachelezanin

279637, 275651, 275953, 277461

November 24, 2024

1 Introduction

The task we were assigned consisted in developing a Convolutional Neural Network to solve a multi-class classification problem. In particular, the goal was to classify RGB images of blood cells belonging to 8 different classes.

Our first aim was to **inspect the dataset** to better understand the problem and make changes to it if needed. We then focused on **developing a deep learning architecture** able to assign the correct labels to unknown images. Eventually we **evaluated the model's performance** based on its accuracy in making previsions on a hidden test set on the Codabench platform.
This is the path we followed: starting from a simple model we added complexity and changed features to achieve as better results as possible.

2 Problem Analysis

The provided dataset was made of 13759 RGB images of size 96x96 categorized into 8 distinct classes, namely: 'Basophil' (1052 images), 'Eosinophil' (2381 images), 'Erythroblast' (1285 images), 'Immature granulocytes' (2226 images), 'Lymphocyte' (1049 images), 'Monocyte' (1393 images), 'Neutrophil' (2530 images), 'Platelet' (1843 images).

The first challenge we had to face concerned the **dataset preprocessing**. By inspecting our dataset we indeed detected the presence of some **outliers** (as the ones shown in Figure 1) which could adulterate the goodness of the model, hence we removed them, obtaining a new dataset made of 11959 images. We also noticed a significant unbalance between the classes, four of them being underrepresented, so at first we decided to use augmentation techniques to readjust them, but then we realized that working on the classes' weights led to better results.

Figure 1: Example of the outliers detected in the dataset.

After this preprocessing phase we began to develop our CNN. Starting it from scratch would have required too much effort and it would have probably led to poor results, therefore we adopted a pretrained network to perform feature extraction and we combined it with fine-tuning to meet the task.

Choosing a proper feature extractor was one of the main challenges we had to face and it required several modifications, as better explained in the "**Method**" paragraph.

3 Method

In order to make the CNN work better we preprocessed the input images and applied AugMix to add variability to the dataset.
Before the training phase we split the dataset into two subsets, one used for training the model and the other one used for validation purposes. After this, the training set was given as input to a pretrained network which could act as a feature extractor.

The first supernets we considered for transfer learning came from the MobileNet and VGG families. We soon realized that these kinds of networks were not suited for our goal, as the scoring was disappointing even if we applied different augmentation techniques on the training dataset, because our model could make the right precision only half of the time. Thus, we looked for a pretrained model which could boost the performance and moved to EfficientNet and Inception networks. In particular, we achieved far better results by adopting **EfficientNetV2M**, which became our final choice.

After applying transfer learning to our inputs we inserted a dense layer with a ReLu activation function, followed by an output layer with softmax activation. However, since the improvement was still not satisfying enough, we introduced more layers. In particular, we added a batch normalization layer after the feature extractor to standardize the input and make learning faster and two more dense layers, each followed by a dropout layer to prevent overfitting.
Thanks to these modifications, we observed a better result in the scoring phase, meaning that our model was less prone to overfitting and more able to categorize unknown images. Moreover, we replaced the Categorical Crossentropy Loss with the **Categorical Focal Crossentropy** to take into account the imbalance within the classes, as this loss function used the weights we had previously assigned to each class based on their size. This change proved to be significantly better than trying to balance the dataset by using augmentation during the preprocessing phase, so we decided to include it in our final model.

In order to adjust the pretrained model to our dataset and classification task we used fine tuning. We froze 640 layers, which were excluded from training, out off 740. Indeed, since we wanted to avoid overfitting, after some tries we realized that putting 100 layers as trainable was not too demanding in terms of memory usage and training time, but could lead to good performances.

3.1 Augmentation

The provided dataset was made of images where cells were clearly distinguishable. We wanted our architecture to be able to classify in more general and uncertain scenarios, so we introduced an augmentation pipeline before transfer learning.
We decide to include several types of augmentation to prevent overfitting, such as Random Flip, Random Brightness, Random Zoom, Random Contrast and Random Rotation.
After observing that it contributed to raise accuracy, to improve the model we implemented a **Test Time Augmentation** that could be applied to the test set. More in detail, TTA predicted the test set labels, it generated four different augmented versions for each input image and then deduced their labels. In the end, it obtained the input image's class by combining these five results and choosing the label associated with the highest value.

3.2 Optimizers

During the last phase of the model's development we tested the entire architecture with different optimizers. After using Adam with parameter 0.001 in our first implementations, we tried to switch to Lion, but we observed suboptimal results in terms of accuracy. We then reintroduced Adam, this time setting the parameter to 1e-4, and eventually switched to the **AdamW optimizer** with parameter 1e-4, as we noticed better accuracy achievements without a relevant difference regarding the time needed for training.

Table 1: Results achieved with two different versions of EfficientNet and different parameters. Final model highlighted in **bold**.

Model's features	Scoring on test set
V2B3 + Adam + augmented dataset + manual classes rebalance + TTA	0.71
V2B3 + Adam + Focal Crossentropy + TTA	0.69
V2M + Adam + augmented dataset + Focal Crossentropy + TTA	0.73
V2M + AdamW + augmented dataset + Focal Crossentropy + TTA	**0.76**
V2M + AdamW + Focal Crossentropy + TTA	0.67

4 Experiments

To have an overview of the results achieved through different models, see Table 1 and Table 2.

Table 2: Results obtained by using different supernets for transfer learning (adopting Adam as optimizer).

Model	Scoring on test set
MobileNetV3Small	0.54
VGG16	0.30
InceptionV3	0.36
EfficientNetV2M	0.62
EfficientNetV2M with TTA	0.66

5 Results

Our final model proved to be discretely good at classifying images, obtaining an **accuracy of 0.9721** on the validation set and a score of 0.76 on the hidden test set.

6 Discussion

We are well aware of the fact that, even if this is a rather satisfying architecture, there could be several improvements that could raise the accuracy and the model's general quality.
First of all, the restrictions due to our devices' hardware and to the limitations in the use of memory and GPU made the training process quite slow, so having more powerful tools would maybe help in developing a more accurate model.
We also recognize that there may be more optimal combinations of parameters and that increasing the patience and number of epochs could lower the loss and raise the accuracy, but the amount of time required would go far beyond a couple of weeks. Moreover, the problems on the Codabench platform during the last days of the development phase prevented us from checking if some of the solutions we had hypothesized could have been helpful.

7 Conclusions

We all came up with some ideas to improve the model's performance and we discussed them as a group before trying to implement them.
Sveva was the main responsible for the coding part of the project, as she was the one who could solve many technical aspects and debug errors, but also she came up with several brilliant ideas which proved to be successful.
Elisa made a lot of effort in finding the best supernet for transfer learning and dedicated her time to searching for an effective solution, patiently waiting for the code to run.
Sara contributed to the development of the model by finding the best parameters and additional layers to add. She also took part in the writing of the final report.
Rachele contributed to the preprocessing and augmentation of the initial dataset and reorganized the path by writing this report.
This CNN was the final result of a teamwork in which each member gave their contribution and participated actively, never giving up and always cooperating with one another.

8 Bibliography

We based our hypotheses and our model's implementation on the lectures from the AN2DL course and on the following articles and papers:

@Keras Keras Applications at https://keras.io/api/applications/
@Keras Keras Layers at https://keras.io/api/layers/
@arXiv M.Kimura, Understanding Test-Time Augmentation at https://arxiv.org/html/2402.06892v1