POLITECNICO
MILANO 1863

AN2DL - First Homework Report
CAMAL

Aleksa Mirkovic, Manuela Maroni, Carlotta Pecchiari

mikovicaleksa, manuelamaroni, cpecchia

244893, 252121, 233381

November 24, 2024

1 Introduction

This report describes our work done to solve the task of creating a model to classify a dataset made of 96x96 RGB images of blood cells into 8 classes: *Basophil, Eosinophil, Erythroblast, Immature Granulocytes, Lymphocyte, Monocyte, Neutrophil, Platelet.* The objective of this *image classification* project was to obtain the highest accuracy possible by using **deep learning** techniques, so to obtain the highest possible number of correctly classified images. Our approach consisted in trying out different optimizers, augmentation techniques and parameters to see what worked best for our dataset, trying also to take inspiration from papers we read regarding the topic.

2 Problem Analysis

The initial assumptions were:

- Problem is *well-defined*: the task involves a finite set of classes (labels), and each input image belongs to one of these classes.
- *Sufficient* and *representative* data: the dataset contains a sufficient number of labeled images for training.
- Classes are *independent*: each class is independent, and there is no overlap or ambiguity between them.

The dataset we were given consisted of 13759 images of blood cells with the respective labels corresponding to the 8 classes. The different classes did not present similar frequencies, ranging from 1000 to 2500 images per class. Also, when calculating the distribution of the mean pixel values we noticed it had a kind of *Gaussian* shape, except for two values that stood out. One was very high and far away compared to the others (mean pixel value: 0.43207592933006533), while the other one was noticed when we were closely inspecting the data (mean pixel value: 0.6524119746278141).

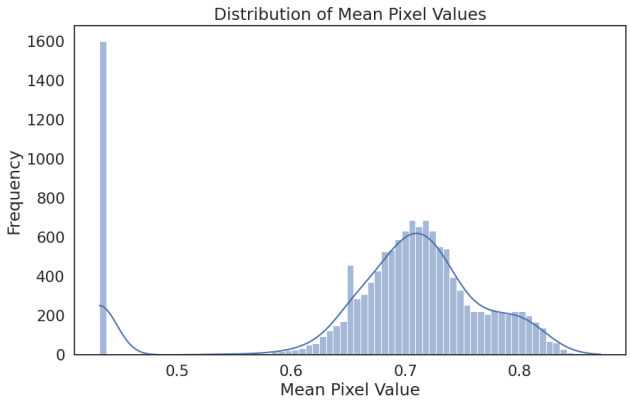

Figure 1: Mean Pixel Value distribution before outlier removal.

By taking a deeper look we discovered that these values corresponded each to an image that appeared many times in the dataset, causing confusion as the

two images were both also associated to different labels, so we decided to remove all the data corresponding to these two mean pixel values (1800 images were removed). Then, we looked for duplicates in the data and found 8 of them, and removed those as well.

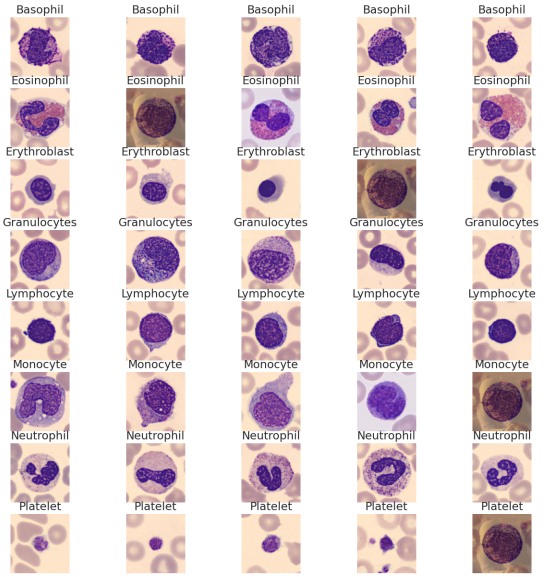

Figure 2: Random blood cells and the respective classes before outlier removal.

The main challenge was represented by the fact that we kept getting a good accuracy on our test data, but a very low one on Codabench. Since this highlighted a probable *overfitting* of the data, which is plausible as images are high-dimensional inputs that can lead to such computational challenges, we tried out many techniques to avoid the problem and finally got to a satisfying result. In particular, by looking at the *confusion matrix* we noticed that there were some problems regarding the class of *Immature Granulocytes*, as many of the images were being wrongly labeled as part of this class due to similarities between the different types of cells, especially between *Immature Granulocytes, Erythroblasts, Lymphocytes, Monocytes* and *Neutrophils*. Another problem was represented by the fact that training various deep learning models required significant computational resources and memory, especially for such a large dataset.

3 Method

First of all we divided all the images by 255, therefore rescaling the pixel values from the range [0, 255] to [0, 1], based on the assumption that neural networks perform better and converge faster when the input features are scaled to a smaller range. Then the images were flattened into a 1D array before applying the previously described *data preprocessing* techniques. Subsequently we performed the **reverse normalization**, before the splitting up of the data into *training, validation* and *test set*, to make sure the data format was consistent with the *pretrained model*. As for augmentation, we tried out many different techniques and ended up getting the best results by dividing the training set into batches of size 35 and applying **RandAugment** and **AugMix** to all of them. This double augmentation allowed the model to see images with different kinds of distortions and combinations, making it more robust to possible variations. To solve the problem of classes having different frequencies we applied **class weighting**, which consists in assigning higher weights to minority classes during training, therefore penalizing *misclassifications* of these classes more heavily.

We built a CNN using **DenseNet121** as the base model, which also allowed to freeze the layers of the base model or to fine-tune them. We then added two *fully connected dense layers* with 1024 and 512 neurons with L2 regularization (0.001 penalty) and *dropout layers* (0.5) on top of the pretrained base model for additional feature extraction. We used the **Relu (Rectified Linear Unit)** activation function for the dense layers, shown below:

$$f(x) = x + \mathrm{ReLU}(x) = \max(0, x) \tag{1}$$

The model also uses data augmentation if provided and applies pre-processing to the input data. We used the **Lion** optimizer and the **categorical cross-entropy** loss, hereby shown:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{2}$$

Finally, a **Softmax** output layer was added at the end to provide class probabilities for the 8 possible classes, according to equation 3:

$$f(x) = \mathrm{softmax}(Wx + b) \tag{3}$$

After creating the model through this technique of **transfer learning** we trained it first by freezing the layers of the base model, meaning their weights

were not updated during training, therefore training only the top layers (dense layers). We defined the **Early Stopping** and the **Learning Rate Reduction** callbacks in order to reduce *overfitting* and improve convergence. Afterwards we fine-tuned the layers that were previously frozen and retrained the model by using a learning rate corresponding to a value equal to 1/10 of the previous one to gradually fine-tune the model and avoid overshooting the optimal solution. The Test Accuracy obtained with the first model was of 71.39%, while the final Validation Accuracy of the second model was 97.21%.

4 Experiments

We tried out many different models with various parameters and techniques, and the main results are highlighted in table 1, which includes also to the baseline model:

Table 1: Table of the main results achieved.

Model	Val Acc	Test Acc	Score
Baseline model	0.1250	0.1250	0.13
MobileNetV2	0.9467	0.9289	0.11
DenseNet121	0.9621	0.9543	0.59
DenseNet121 + RandAugment and AugMix	**0.9721**	**0.9677**	**0.80**

5 Results

The first model we implemented was made by using **MobileNetV2** and did not perform any better than the baseline model, even though it presented promising results for what regards the validation and test accuracy. We tried to improve it by changing some parameters, especially by applying a more aggressive *augmentation*, but performance only improved by approximately 10% in accuracy. We also tried to avoid *overfitting* by applying *regularization* and *early stopping*, but the model still did not improve significantly. The following step consisted in the implementation of the **DenseNet121** model through the fine-tuning technique, which got us to reach a score of 0.59. The model that gave us the best score, the one described in this report, was able to reach a score of 0.80 thanks to the double augmentation technique of **RandAugment**and **AugMix**. The fact that the model got better and better by applying more significant augmentation strategies was a surprise to us, as we did not imagine there would be the need for such an aggressive augmentation, given the sufficient amount of data.

6 Discussion

The results we obtained were satisfactory considering the instruments we had to complete the task. By taking a look at different papers [1] and taking inspiration from some of them we tried to implement a model that could resemble the ones proposed [2], taking into consideration the memory constraint and high computational cost limitations. We believe our model to have a good prevention of *overfitting*, thanks to the techniques of fine-tuning, early stopping, weight balance, augmentation and dropout. The reason why we are not getting as good results on Codabench as the ones obtained in the validation set might be because we are assuming to have a representative training dataset, but maybe it does not actually represent the variations and diversity of the real-world data the model is being tested on.

7 Conclusions

Our work mainly focused on the improvement of the accuracy of our image classification model. The very first model was implemented by Mirkovic, and subsequently we all tried to improve it by making some changes in the parameters (learning rate, number of epochs, augmentation parameters, etc.) and in the used techniques, based also on the papers found by him and Pecchiari. Subsequently Mirkovic implemented the fine-tuning method, and Maroni improved it by adding the double augmentation technique that allowed us to get our highest score, finally the report was done by Pecchiari. The model could be improved by adding more dense layers for a better feature extraction, which we weren't able to implement due to the limited computational power of our computers. Nevertheless, every dataset has its own specific features that need to be taken into consideration in the model building process, together with the requirements of the task. It would also be interesting to implement a technique regarding the *explainability* of the model, as this is crucial in the medical field.

References

[1] Yuning Huang, Jingchen Zou, Lanxi Meng, Xin Yue, Qing Zhao, Jianqiang Li, Changwei Song, Gabriel Jimenez,Shaowu Li, Guanghui Fu, *Comparative Analysis of ImageNet Pre-Trained Deep Learning Models and DINOv2 in Medical Imaging Classification*, arXiv, 2024. [Online]. Available: https://arxiv.org/abs/2402.07595

[2] Rabia Asghar, Sanjay Kumar, Paul Hynds, Abeera Mahfooz, *Classification of All Blood Cell Images using ML and DL Models*, arXiv, 2024. [Online]. Available: https://doi.org/10.48550/arxiv.2308.06300