AN2DL - First Homework Report riconvoluzione informatica

Valeria de Gennaro, Donato Fiore, Lorenzo Fonnesu, Gabriele Lorenzetti

valr0109, donatofiore01, fonzy, gabriele

251450, 248399, 249663, 259410

February 10, 2026

1 Introduction

In this project, we tackled the problem of *classifying images of blood cells* into eight different classes. The objective was to develop a **deep learning model** capable of analysing these images and correctly assigning each cell type to the corresponding class. The dataset used was first analysed and cleaned of any duplicate or misleading images. As for the construction of the model, we relied on a pre-trained model using the **transfer learning** technique, followed by a **fine-tuning** phase. *To improve performance and reduce overfitting*, the model was supplemented with **data augmentation** strategies and the application of **dropout**. These measures allowed us to achieve an accuracy of up to **80%**.

2 Problem Analysis

2.1 Dataset characteristics

The problem was presented with an initial dataset as described above. The first step was the analysis of the latter so as to have a deeper understanding of the problem.
The analysis showed:

- **Distribution** The dataset was composed by a total of 13759 images, divided in a total of 8 classes;
- **Disruptive Elements** The dataset included 1800 out-of-context images (all gathered at the end of the dataset) and 8 duplicate images that add no value to the training process, lowering accuracy and performances.

2.2 Main challenges

The analysis of the dataset brought the following conclusions:

- **pre-processing** with the goal of cleaning the dataset of disturbing elements needed;
- addition of **image generality** needed, due to the limited number of images, in order to increase the generality of the model out of the training phase;
- use of a **proper model** capable to learn efficiently from the resulting dataset.

2.3 Model development

The starting point of the model was a simple **feed-forward neural network**, in which we applied **dropout layers**, **global average pooling** and **early stopping**. However, due to poor results, this approach was replaced by the use of **transfer learning** and **fine tuning**. In this phase, we experimented different models, which included **MobileNetV2** [3], **ResNet50** [2], **EfficientNetB0**

[6] and **ConvNeXtBase** [4], obtaining our better results with the last one. In parallel, we also introduce **data augmentation** techniques such as **random augmentation** [1], **cut mix** [7] and **cut out** to improve the accuracy of the model.

3 Method

The following section describe all the techniques and approaches used to obtain the final model.

3.1 Data Preparation

Given the reasons described above, the data preparation phase of the dataset is important to achieving optimal results even outside the training environment. The process itself consists of two steps: **Cleaning**, where the out-of-context and duplicated images were removed, and **Normalization**, where the images were normalised by dividing each value by 255.0, in order to improve stability and effectiveness of training and allows the model to achieve target accuracy in less time.

3.2 Pre-Processing

Pre-processing aims to enhance both the size and diversity of the dataset. This is usually achieved by applying **Augmentation technique**: various transformations to the images, increasing their generality and making the model more robust to real-world variations. For our purpose, we adopted the following approach: the dataset was duplicated and split in different ways. Each slice of the dataset was then processed using a distinct technique, as illustrated in Figure 1

- **DATASET_1** The first dataset was divided and processed as follows: **40%** designated for the VAL_TEST portion, which was then split evenly into validation and test dataset, without applying any other operation; **60%** was used for data augmentation, evenly applying the **cut mix** [7] and **cut out** techniques (50% each)
- **DATASET_2** The second dataset was divided and processed as follows: **50%** allocated for **random data augmentation** [1], with 70% of this portion undergoing augmentation using multiple random layers (four layers per image), introducing variability among the images. The remaining 30% of this portion was kept unaltered to ensure that some images remain original; **50%** dedicated to a pipelined data augmentation process applied uniformly to the entire portion.

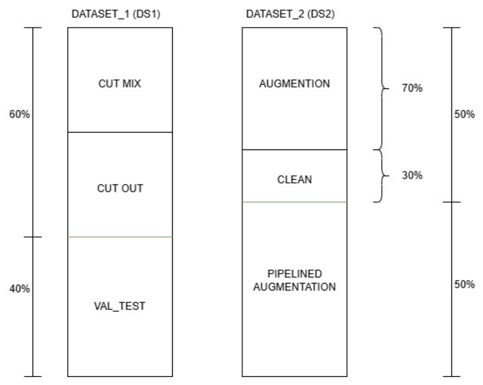

Figure 1: splitting of the dataset

At the end of this step, the two datasets (without the VAL_TEST portion) were concatenated, obtaining the final dataset, ready for training.

The augmentation techniques were implemented using the **keras_cv** library, in particular: RandAugment [1], RandomTranslation, RandomContrast, RandomHue, RandomSharpness, RandomColorDegeneration, RandomChannelShift, Solarization, RandomBrightness, CutMix [7] and CutOut.

3.3 Transfer learning and fine tuning

With the aim to achieve better accuracy, the following techniques were used: **Transfer learning** Using ConvNeXtBase as a pre-trained model, three additional layers were added (GAP, dense and dropout) and exclusively trained using 10 epochs, early stopping and **AdamW optimizer**, with a learning rate of 1e-3 and weight decay of 1e-4, while keeping the pre-trained model's layers frozen; **Fine Tuning** Achieved by unfrozing all the conv2D and depthwiseConv2D layers starting from the 150th layer onward, followed by training the model for 20 epochs with early stopping and **Adam optimizer**, with a **cosine annealing** for the learning rate starting from 1e-4. In particular, as to the cosine annealing, from [5] we have:

$$\eta_t = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})(1 + cos(\frac{T_{cur}}{T_i}\pi)) \quad (1)$$

with η being the learning rate and T_{curr} accounting for how many epochs have been performed since the last restart.

Finally, regarding the activation function and the categorical cross-entropy loss, we used in both cases the softmax activation function and the categorical cross-entropy function shown respectively in equations 2 and 3:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \quad (2)$$

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \quad (3)$$

This approach allowed us to greatly improve our performances and accuracy.

3.3.1 ConvNeXtBase model

In our final result we used ConvNeXt, a modern convolutional network which introduces architectural improvements over traditional models, guaranteeing more accuracy over them.

4 Experiments

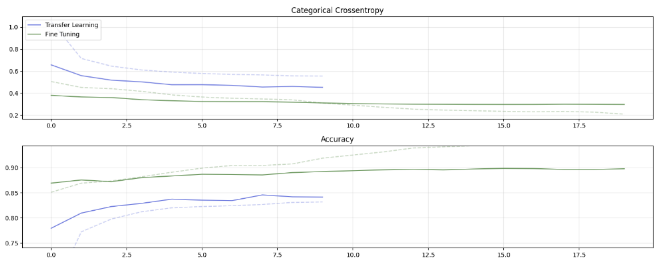

Figure 2: fine-tuning

The graph of Categorical Crossentropy shows that the model with Fine Tuning achieves constant improvement over Transfer Learning, with lower values on both training and validation. The Accuracy graph shows how Fine Tuning outperforms Transfer Learning in terms of accuracy, achieving more stable and higher results as the epochs progress.

5 Results

The proposed model achieved a final score of 80% on Codabench, significantly outperforming the baseline training score of just 20%. This result highlights its good **predictive accuracy** and **generalization** capabilities.

Additionally, certain attempts to improve the model yielded unexpected results: applying k-fold cross-validation or MixUp techniques did not lead to any improvements and, in some cases, even worsened performance.

6 Discussion

The proposed model demonstrates a solid general approach, particularly in terms of dataset expansion and generalization, making it more versatile and suitable for recognizing a diverse range of images. However, despite the strength of the concept, the approach does not perform optimally due to the risk of overfitting. This issue is likely caused by suboptimal configurations in the preprocessing step, which ultimately limits the model's accuracy.

7 Conclusions

In terms of future work, it would be interesting to be able to incorporate k-fold cross-validation. This could lead to improvements in the model, resulting in better results through increased robustness and improved generalization ability. Another possible strategy may be to implement MixUp techniques, which allow creating synthetic examples by combining images and their labels, reducing the risk of overfitting and improving the model's ability to learn more general representations.

Model	**Score**	**CDB score**	**Val_accuracy**	**Val_loss**	**Accuracy**	**Loss**
EfficientNetB0 before FineTuning			88.91	0.3654	71.99	0.8444
EfficientNetB0 after Fine-Tuning		0.64	90.38	0.3439	73.74	0.8061
ConvNeXtBase before Fine-Tuning			84.56	0.4523	78.16	0.5686
ConvNeXtBase after Fine-Tuning		**0.80**	**89.83**	**0.2968**	**94.89**	**0.2201**

References

[1] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:*, 1909.13719, 2019.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:*, 1512.03385v1, 2015.

[3] A. Howard, M. Sandler, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *arXiv preprint arXiv:*, 1801.04381v4, 2019.

[4] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *arXiv preprint arXiv:*, 2201.03545v2, 2022.

[5] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:*, 1608.03983v5, 2017.

[6] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:*, 1909.13719, 2019.

[7] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *arXiv preprint arXiv:*, 1905.04899, 2019.