AN2DL - First Homework Report
AXON Biotech

Matteo Balice, Antonio Giuseppe Doronzo, Alessandro Masini, Marco Muraro

matteobalice, imtonio, jumpit, marcomuraro4

251648, 251320, 232814, 103391

November 24, 2024

1 Introduction

Blood analysis is of fundamental importance in medical diagnostics for individual's physiological condition assessment and disease detection [2]. This project addresses this challenge by developing a **deep learning model** to classify blood cell images into eight predefined classes, depending on their nucleus, shape and cytoplasm [1]. The approach includes:

- **Data preparation**: The dataset was analyzed and preprocessed to enhance model performance and mitigate inconsistences and variability.
- **Model design and training**: A deep learning model was designed, trained, and optimized to effectively capture the distinguishing features of each blood cell type.
- **Evaluation**: The model's performance was evaluated on validation and test sets.

2 Problem Analysis

The dataset consists of 13,759 color images, each sized 96×96 pixels with 3 channels representing the *RGB color space*. Each image is labelled with one of eight *blood cell types*.

A significant challenge is the *variability in blood cell morphology*, such as differences in shape, size, and color due to staining techniques. Additionally, the dataset contains *noise*, including duplicate images with different labels and unrelated artifacts that are not cells, requiring **careful cleaning**. Another key challenge was finding the **optimal combination of augmentation techniques** to enhance generalization while preserving *biologically relevant features*. Finally, identifying the **most suitable model architecture** for this task required **extensive experimentation** to capture relevant and complex characteristics.

3 Method

Considering the small size of the available dataset, the chosen methodology is based on **Transfer Learning**. The pre-trained network serves as a feature extractor whose output is then propagated through the final fully connected part of the model developed, devoted to performing the final classification task. With reference to Figure 1, in the following, the macro steps of the methodology adopted are described in detail.

Firstly, accurate **data inspection** and **cleaning** were needed. Outliers (duplicate images with wrong labels and other noisy data) were removed. Hereafter, the cleaned dataset got split into *training set* (81%), *validation set* (9%), and *test set* (10%).

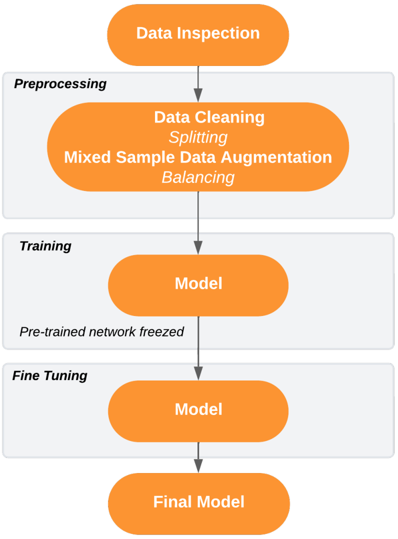

Figure 1: This high-level diagram describes the macro steps of the methodology.

At this stage, training data was augmented in such a way as to introduce noise, increase number of instances, and balance classes. For this purpose, **Mixed Sample Data Augmentation** techniques, such as MixUp, CutMix, and AugMix transformations, were found to be necessary to make the model more robust. Indeed, as the validation set was preserved as cleaned, validation process tells whether the model is still able to capture remarkable features or not. Thanks to this *train-validate* mechanism, the data augmentation pipeline was designed and incrementally improved by adding or removing transformations and meticulous parameters tuning.

The final model includes a preprocessing stage for further data augmentation and integrates **ConvNeXt Large** as *feature extractor*. A **GAP (Global Average Pooling)** layer connects the pre-trained network to the fully connected part, having the following structure: a **Batch Normalization** layer, followed by a **SE (Squeeze-and-Excitation)** block and a **GAP** layer, which is then connected to two **custom blocks**, each one consisting of **Dense layer**, **Batch Normalization**, **LeakyReLU activation** and **Dropout**, the latter to introduce *regularization*. The architecture ends with a **Softmax** layer giving the normalized probabilities for all eight classes.

SE block (*attention* mechanism) performs dynamic channel-wise feature recalibration.

Furthermore, the training was carried out in two stages. Firstly, only the fully connected part was trained, while the pre-trained network was freezed. **Fine Tuning** was then employed to improve model performance and leverage the wealth of knowledge of the pre-trained feature extractor. Lower layers were prevented from updating their weights so to keep taking advantage of the general features already learned when trained on ImageNet. Fine tuning was carried out multiple times with the purpose of performing **progressive unfreezing** from 150 freezed layers to 20 freezed layers with an incremental step of 50 and 30 at the last iteration. This process requires a lower learning rate with respect to the previous training, so as to preserve remarkable features that were previously learned by the fully connected part. Only **Conv2D** and **DepthwiseConv2D** layers were trained during fine tuning, whereas **Batch Normalization** layers remained freezed to retain statistics extracted from ImageNet, making the model more robust and reliable.

Moreover, the whole training process employs **Categorical Crossentropy** loss function and **Lion** optimizer. Alongside **Dropout** layers, **Early Stopping** was also applied to enhance **Regularization** and further *prevent from overfitting*.

For parameters and hyperparameters values, refer to the attached notebooks.

4 Experiments

The proposed model is the result of several experiments in which different kinds of pre-trained networks were used for transfer learning with different configurations for the fully connected part. As shown in Table 1, model performance was evaluated from predictions on both the clean test set and the augmented test set.

The very first experiments were conducted using a **model built from scratch** and subsequently performing transfer learning with pre-trained models from **VGG** and **MobileNet** families. These architectures show remarkable results when working with the clean dataset (**97-98%**), but their performance considerably decreases when introducing a complex data augmentation pipeline. For this reason, more sophisticated models were needed, such as networks from **ResNet** and **ConvNeXt** families.

In Figure 2, the confusion matrix for the **final model** based on **ConvNeXt Large** is reported.

Table 1: Metrics from predictions on the clean test set (left) and the augmented test set (right).

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
ResNet-50	85.82 / 71.69	88.14 / 73.70	85.82 / 71.69	86.23 / 72.33	98.78 / 95.19
ConvNeXt Base	95.23 / 80.74	95.43 / 80.69	95.23 / 80.74	95.25 / 80.59	99.82 / 97.61
ConvNeXt Large	**97.10 / 90.13**	**97.27 / 90.13**	**97.10 / 90.13**	**97.13 / 90.07**	**99.94 / 99.28**

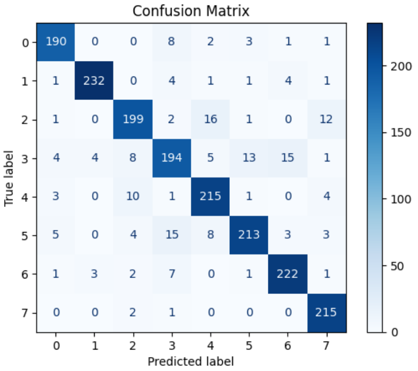

Figure 2: Confusion matrix related to predictions from the final model, based on ConvNeXt Large.

5 Results

Previous studies achieved accuracies such as 90.79% [3] with CNN-RNN and 91.7% [4] with ResNet.
At the early development phase, **ResNet**-based model achieved **58%** accuracy on Codabench. Then, the **ConvNeXt Base** model was trained using transfer learning only, achieving a score of **68%** with a simple data augmentation pipeline. Introducing fine-tuning resulted in performance increase to **73%**. Further gains were made by incorporating **Mixed Sample Data Augmentation**, which diversified the training set and elevated the score to **77%**. Finally, the fully connected part was enhanced by adding more complex layers (**SE block**, **Batch Normalization**, **Dropout**), resulting in a Codabench score of **80%**. **ConvNeXt Large**-based model with progressive fine-tuning led to further performance improvement to a final Codabench score of **87%**.

6 Discussion

Our final model demonstrated excellent performance, achieving a test accuracy of **97.10%** when classifying high-quality clean images of blood cells. However, we assumed that the Codabench test set contains noisier and less well-recognizable images. This hypothesis is supported by a subtle contrast in performance: our model excelled on our clean dataset, but slightly struggled with the more challenging test set from Codabench.
To fill this gap, larger models, such as **ConvNeXt XLarge**, could be employed. Nevertheless, in this work, **ConvNeXt XLarge** was not used because of GPU memory limitations. To overcome this problem one could decrease the batch size so as to load fewer images at each training iteration, but this would considerably increase the time needed for training.

7 Conclusions

The proposed model achieves **97.10%** accuracy on clean images, as we expect to work with in a real world blood analysis scenario. In contrast, the performance on noisy data slightly decreases to **90.13%** on local test set and **87%** on Codabench. To enhance the results, *future developments* could include employing a **larger model**, such as **ConvNeXt XLarge**, exploring **more complex augmentation techniques**, and considering a **bigger dataset**.

Contributions

Team working and meticulous organization of activities among teammates was fundamental to achieve the final goal.
Matteo Balice - CNN from scratch, ConvNeXt-based models, Fully connected part design
Antonio Giuseppe Doronzo - Data balancing, Augmentation, VGG and ResNet-based models
Alessandro Masini - Data augmentation and cleaning, ResNet models, ConvNeXt-based models
Marco Muraro - Research on state-of-the-art techniques, Preprocessing, MobileNet-based and ConvNeXt-based models

References

[1] R. Asghar, S. Kumar, and P. Hynds. Automatic classification of 10 blood cell subtypes using transfer learning via pre-trained convolutional neural networks. *Informatics in Medicine Unlocked*, 49:101542, 2024.

[2] R. Asghar, S. Kumar, and A. Mahfooz. Classification of blood cells using deep learning models. *Research Gate*, page 11, 08 2023.

[3] G. Liang, H. Hong, W. Xie, and L. Zheng. Combining convolutional neural network with recursive neural network for blood cell image classification. *IEEE Access*, 6:36188–36197, 2018.

[4] L. Ma, R. Shuai, X. Ran, W. Liu, and C. Ye. Combining dc-gan with resnet for blood cell image classification. *Medical & biological engineering & computing*, 58:1251–1264, 2020.