POLITECNICO
MILANO 1863

AN2DL - First Homework Report
Gan's n Roses

Denis Sanduleanu, Jonatan Sciaky, Alessio Caggiano, Mattia Repetti

sanduleanud24, jonardan, falcro, mattiarepetti

251564, 252155, 258744, 251626

November 24, 2024

1 Introduction

Our goal was to classify images of blood cells belonging to 8 different classes using **Deep Learning** techniques. Following an incremental approach, we implemented various models. We approached the problem by creating a **custom CNN**, but given the complexity of the problem and the discrepancy between the results obtained locally and those on the verification platform, we moved on to the use of more advanced techniques, namely a **pre-trained network** (Transfer Learning) and optimizing performance through the use of **advanced data augmentation** and **regularization techniques.**

2 Problem Analysis

Dataset Characteristics
The training data contains:

- A Numpy array of shape (13759, 96, 96, 3) containing the RGB images.
- A Numpy array of shape (13759,), with class values ranging from 0 to 7, corresponding to the classes of the blood cells.

By analyzing random plots, we identified two main outliers in our dataset, namely Shrek and Rickroll, and removed them along with all duplicate images. Starting with 13,759 images,The dataset was reduced by approximately 13% after cleaning. Finally, we checked for any NaN values that could be problematic but found none.

Challenges
The dataset presented two major challenges:

- **Class Imbalance:** Some classes were underrepresented, making it difficult for the model to learn equally across all categories. To manage this problem we try two main approaches:
  - **Oversampling** only the minority classes until all the classes reach the same number of instances.
  - Usage of **Categorical Weight**. To each image is assigned a weight based on the number of instances of that class in relation to the total number of instances in the dataset, and these weights influence the loss function during model training, ensuring that the model pays more or less attention to specific classes
- **Data Augmentation and Generalization:** Finding the right augmentation techniques was critical to improve the model's generalization capability. At the same time, balancing the model's ability to generalize without overfitting required careful fine-tuning of augmentation parameters and regularization techniques.

3 Method

Our approach involved using transfer learning techniques by leveraging the pre-trained weights from **ImageNet**. We utilized mainly three different networks: **convNeXtBase**, **convNeXtLarge**, and **ConvNeXtSmall**, each followed by dense and dropout layers, with a softmax activation layer at the end. Following the results of the initial tests concerning the difference between local accuracy and platform accuracy, we decided to focus on Data Augmentation as shown in Figure 1, and try to find the best parameters for our models using **trial and error** approach. For image augmentation, we applied various methods explained in the following chapter. Given the use of very large and complex models, one of our priorities from the very beginning was to prevent overfitting. To achieve this, we implemented techniques such as **early stopping**, **reducing the learning rate on plateau** and also the addition of **dropout layers**.

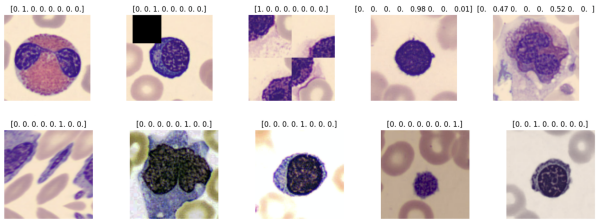

Figure 1: Example of augmetend images.

4 Experiments

The initial approach involved designing a custom CNN from scratch, drawing inspiration from the foundational concepts of convolutional neural networks (CNNs) for computer vision and elements from popular architectures. However, the results were not satisfactory, leading us to adopt a transfer learning approach instead. Among the **augmentation techniques** employed were RandAug, CutMix, MixUp, GridMask and CutOut. These techniques were applied to train the network on a diverse range of images and scenarios, making it robust to various challenges and capable of generalizing effectively.

Additionally, we attempted to isolate the cells from the rest of the background to reduce unnecessary content in the images. This preprocessing step was designed to enhance the model's focus on relevant features, minimizing distractions caused by extraneous elements.

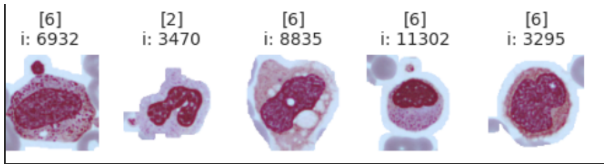

Figure 2: Example of masked images.

In terms of optimizers, we experimented with **Adam**, **Nadam**, **AdamW**, and **Lion**, each with unique characteristics, to evaluate their impact on model performance. These optimizers differ in their update rules and mechanisms, offering varied approaches to learning stability and convergence speed.

For regularization, we incorporated dropout layers to mitigate overfitting. Multiple trials were conducted to fine-tune the dropout rates at different layers, achieving a balance that prevented both overfitting and underfitting.

Additionally, we tried to apply**L1**, **L2**, and **L1L2 regularization**.

We also conducted several experiments by making the last layers of the network trainable. This approach aimed to **fine-tune** the model by updating the parameters of the final layers, enabling it to adapt better to the specific features of our dataset while leveraging the pre-trained knowledge from earlier layers. The number of trainable layers was gradually increased during testing to find the optimal balance between performance improvement and overfitting prevention.

5 Results

Improved Score with Augmented Dataset: The score increased significantly when using an augmented dataset compared to the original one. The augmentation techniques that yielded the best results were **CutMix**, **MixUp**, and **RandAug**. Among these, **RandAug** delivered the most significant improvements during the testing phase.
Improved Score with Dropout Layers: The score improved further with the inclusion of dropout layers to prevent overfitting. Through multiple trials, we identified dropout values that ensured the model's performance was adequately balanced, avoiding both excessive overfitting and underfitting.

Table 1: Refers to local test.

Model	Accuracy	Precision	Recall	f1-score
convNeXtLarge	88	89 ± 8.5	85 ± 13	87 ±9
convNeXtBase	97	96 ±4	95 ± 4	97 ± 3
convNeXtSmall	92.8	92.9 ± 2.8	92.8 ± 2.9	92.8 ± 2.9

Further Improvement by Training the Last Layers: The score was further enhanced by making the parameters of the last layers trainable. The best results were achieved by making the **last 25% to 40% of the network's layers trainable,** allowing the model to fine-tune the deeper representations while leveraging the pre-trained features from earlier layers.

One of the most unexpected results was the significant discrepancy in performance between the local test set and the platform's test set. This led us to experiment with various augmentation and regularization techniques to maximize the model's generalization capabilities.
We achieved the best results with a ConvNeXtLarge with the following parameters and features:
-Batch size: 256
-Data split: 60% (7100 images) training, 36%(4300 images) validation and 4% for test set (500 images)
We create a training set augmenting each initial image with CutMix, MixUp, RandAugment and Split-and-Rotate, and then concatenated it to the training test for a total of 42000 images
Validation set has been augmented with RandAugment on the 70% of the images.
-EarlyStopping and patience: 8 epochs decreased to 4 in the following fine-tuning phase
-Fine-tuning layers: 204 frozen layers
-Optimizer: AdamW with learning rate:$1e^{-3}$ and weight decay:$1e^{-5}$
-Regularizer: L1L2 Regularization with $1e^{-3}$
-Dropout rate: 35%

6 Discussion

Our model generalizes well across all classes, achieving a nearly diagonal confusion matrix and consistent accuracy on validation and test sets. While initial overfitting was a challenge, we mitigated it through **dropout**, **regularization**, and **early stopping**, ensuring a balanced fit. **Class weights** and **stratified splits** addressed dataset imbalance, though the limited size of our dataset, combined with discrepancies between our local data and the provided test data, such as differences in augmentations applied to the test images on the platform, introduced significant challenges. **Extensive data augmentation and careful tuning** were key to overcoming these issues, enabling robust performance despite these limitations.

7 Conclusions

We developed a robust classifier achieving 0.83 accuracy on the platform, significantly outperforming a random guesser's 0.125 for 8 classes. The **confusion matrix** confirmed consistent performance across all classes, with no class dominating or being neglected. **Data augmentation** had the greatest impact, improving the model's ability to generalize effectively. Additionally, **transfer learning** played a key role by leveraging pre-trained features from larger datasets, allowing the model to learn complex representations efficiently despite the limited size of our dataset.

While the model performed well, future work could explore advanced techniques like **model ensembling** to enhance accuracy further. Additionally, this classifier could be adapted for similar tasks, such as classifying other cell types, expanding its utility beyond the current dataset.

8 Member Contributions

We began by working together on data cleaning and selecting augmentations. Then, Alessio and Mattia focused on building a CNN from scratch, while Jonatan and Denis explored transfer learning. After initial tests, we shifted entirely to transfer learning, each developing a model. In the final four days, we all focused on refining the most promising model, ConvNeXt Large.

References

- *An explainable AI-based blood cell classification using optimized convolutional neural network*
- *White Blood Cell Classification Using Multi-Attention Data Augmentation*
- *CutMix, MixUp, and RandAugment image augmentation with KerasCV*
- *RandAugment layer*
- Optimizers