POLITECNICO
MILANO 1863

AN2DL - First Homework Report
AI-pacapa

Luca Montiglio, Nizare El Ziani, Paolo Gennaro, Yigit Ali Selcuk

Codabench lucamontiglio, Codabench nizare, Codabench paologennaro, Codabench yigits

260144, 252591, 259522, 274471

September 24, 2025

1 Introduction

This project focuses on *image classification* using **deep learning** techniques to categorize 96x96 RGB images of blood cells into eight classes, addressing challenges like *visual variability* and *class imbalance*.

We employ *convolutional neural networks* (CNNs) for **accurate multi-class classification**, leveraging their capability to learn complex visual features. Our approach involves *data preprocessing*, *designing a CNN architecture*, and *refining the model* to optimize performance.

This report details the methodology and key findings, demonstrating the effectiveness of CNNs for this classification problem.

2 Problem Analysis

2.1 Data Characteristics

The dataset consists of 96x96 RGB images of blood cells in eight classes. Initial inspection revealed mislabeled images, outliers, irrelevant samples, and duplicates across classes, resulting in the removal of approximately 2,000 problematic images.

2.2 Main Challenges

Key challenges included *class imbalance*, which risks biasing the model, and the *limited dataset size*, increasing the likelihood of overfitting. Additionally, *visual variability* in blood cells shape, texture, and color added complexity to the task.

2.3 Initial Assumptions

The cleaned dataset was assumed to be representative and unbiased. After *model order selection* with k-fold cross-validation, the chosen architecture was expected to effectively extract relevant features while matching the problem's complexity.

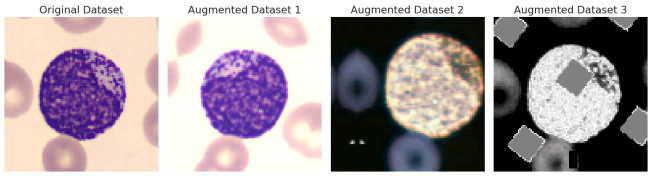
3 Method

We utilized *TensorFlow-Keras* for model training. Three model families were employed: *MobileNet, EfficientNet, and ConvNeXt.* Each model was fine-tuned using pre-trained weights from *ImageNet*, employing **transfer learning**. We initially froze the early layers of the models to retain generic features and progressively unfroze deeper layers for task-specific learning.

To enhance the generalization capability of the CNNs, we applied **data augmentation techniques.** Initially, we added an augmentation layer that performed simple transformations such as *flipping, rotations, and brightness adjustments.*

With the use of a local GPU, we were able to create several pre-processed datasets using *KerasCV*, applying different numbers of augmentations per

image, and also more advanced ones, such as MixUp, CutMix, etc..



This approach allowed us to effectively expand the dataset and improve model robustness.

In order to **prevent overfitting**, *L2 regularization* was used on the fully connected layers, and dropout layers were added after each dense layer, as well as *early stopping* on the training.
As loss function we choose *categorical cross-entropy*:

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

that was used for all models.
The Adam optimizer was applied as the baseline for training.
Class imbalance was addressed by calculating class weights, defined as $w_j = \frac{N}{C_j}$, where N is the total number of samples, and C_j is the count for class j. The loss function is adjusted by multiplying each class's contribution by its respective weight.
An *adaptive learning* rate was implemented in the final phase to overcome local plateaus.

4 Experiments

4.1 Experimental Setup

The dataset was pre-processed and split into training and validation sets, with multiple versions created using varying levels of augmentation: light, moderate, and heavy. Additionally, a version refined using K-means clustering was tested during the ***ConvNeXt*** experiments. Besides *K-means clustering*, other techniques were also tested to reduce image noise during prediction, including High-Pass filtering, *nucleus segmentation*, *noise elimination* based on the model's score for each cluster, and *test-time augmentation*. However, none of these methods led to improvements in accuracy, likely due to the strong noise resistance provided by heavy data augmentation during training. Training was monitored with metrics such as accuracy, precision, recall, and F1-score.
K-fold cross-validation was used to evaluate generalization potential.

4.2 Model Experiments

Gradually experimenting on different models, we traded an increase in the number of parameters and computational cost for the capacity to extract complex patterns, sufficient representation power, and generalization on large augmented datasets.

MobileNetV3Small - It was used as a baseline to establish initial performance. The model was trained with light augmentation (cropping, flipping, and rotation) by adding an augmentation layer and fine-tuned by progressively unfreezing deeper layers. *MobileNetV3Small* was chosen for its computational efficiency, making it suitable for our initial limited resources while still providing solid performance.

EfficientNetV2B3 - This model was selected for its balance between performance and efficiency. It was trained using pre-processed datasets created with *KerasCV* and *fine-tuned* in few rounds, with *L2 regularization* applied to prevent overfitting. *EfficientNetV2B3* led us to the steepest improvement, enhancing our team to the head of challenge standings. Optimizers such as AdamW, Ranger, and Lion were also used to assess their impact on robustness and generalization.

ConvNeXt - This model was chosen to explore newer, more advanced architectures, motivated by its promising design that offers enhanced robustness and performance on complex tasks. Models variants were trained on datasets with both light and heavy augmentations, with class weights recalculated to address class imbalance. A K-means clustered version of the dataset was also tested, although no significant improvements were observed. We also tried to do *Hyper-Parameters Tuning* with *KerasTuner*.

4.3 Evaluation Metrics and Validation

The performance of the models was evaluated using precision, recall, F1-score, and accuracy. The best-performing model, ***ConvNeXtXLarge***, achieved the highest accuracy during testing on our augmented datasets.

Table 1: Performance comparison of ***MobileNetV3Small***, ***EfficientNetV2B3***, and ***ConvNeXtXLarge***. The evaluation was conducted on the heavy augmented dataset.
Note: we didn't expect better accuracy on this dataset, yet was used as the benchmark for generalization capabilites.

Model	Accuracy	Macro Avg Precision	Macro Avg Recall	Macro Avg F1-Score
MobileNetV3Small	75% (± 1.0)	74% (± 1.5)	74% (± 1.3)	72% (± 2.0)
EfficientNetV2B3	81% (± 0.5)	81% (± 0.3)	79% (± 0.4)	79% (± 0.3)
ConvNeXtXLarge	**86%** (± 0.4)	**86%** (± 0.3)	**85%** (± 0.4)	**85%** (± 0.3)

Table 2: Classification report of the best-performing ***ConvNeXtXLarge*** model tested on the cleaned initial dataset.

Class	Precision	Recall	F1-Score	Support
0	0.92	**0.99**	**0.96**	430
1	**0.99**	**1.00**	**0.99**	1129
2	**0.99**	0.96	0.98	553
3	0.96	0.94	0.95	1047
4	0.94	**0.99**	**0.96**	431
5	0.95	0.93	0.94	466
6	**0.99**	0.97	0.98	1135
7	**1.00**	**0.99**	**0.99**	788
Accuracy	**0.97**			5979
Macro avg	**0.97**	0.97	0.97	5979
Weighted avg	**0.97**	0.97	0.97	5979

For classification reports of other models, tested with various configurations and datasets, please refer to the supplementary materials available in the project repository on GitHub.

5 Results

MobileNetV3Small served as the baseline. *EfficientNetV2B3* reached **86%**, while *ConvNeXtXLarge* achieved **89%** in earlier experiments, benefiting from augmented datasets, fine-tuning, and optimizations. The final Codabench evaluation, **ConvNeXtXLarge** reached **90%**.

6 Discussion

Initially, *EfficientNetV2B3* outperformed *ConvNeXt*, with **87%** accuracy compared to **86%**, possibly due to overfitting in *ConvNeXt* on earlier dataset configurations. However, with improved training and fine-tuning, *ConvNeXt* surpassed *EfficientNetV2B3*, as expected.

K-means clustering and other *noise filtering techniques* showed no significant benefits, highlighting the importance of effective augmentations.

Additionally, similar improvements to those made to *ConvNeXt* could have led to greater gains for *MobileNetV3Small* and probable modest improvements for *EfficientNetV2B3*. Although, at this stage, we decided to capitalize on the most promising model.

7 Conclusions

This work explored different models and optimization strategies to improve performance on the Codabench dataset. Starting with *MobileNetV3Small* as the baseline, we gradually applied dataset augmentations, fine-tuning, and optimizations with adaptive learning, leading to *ConvNeXtXLarge* achieving **90% accuracy**. Assuming we were being tested on an artificially augmented dataset, we consider this a good result for real application without over-fitting artifacts.

Future work could focus on refining optimization processes, exploring advanced filtering techniques, and automating hyper-parameter tuning to further enhance performance.

In general, we consider the results obtained to be transversal to many other image classification problems.

The project progressed through close collaboration among all team members, who participated equally in every phase of the work. There were no roles or tasks that were more burdensome for anyone, as everyone contributed equally, working together to achieve the set goals.