POLITECNICO
MILANO 1863

AN2DL - First Homework Report
DeepDrive

Alberto Eusebio, Martina Riva, Lorenzo Rosati

albertoeusebio, martina6, rosze

244659, 249408, 233465

November 24, 2024

1 Introduction

This project focuses on classifying **96x96 RGB blood cell images** into eight categories using deep learning. The main goal is to achieve high accuracy and generalization through *transfer learning* and *fine-tuning*. Key steps include:

- Dataset preprocessing and augmentation.
- Selection of a robust pre-trained model.
- Model training, evaluation, and optimization.

Despite challenges like class imbalance and dataset variability, the approach aims to deliver a highly accurate and generalizable model.

2 Problem Analysis

The dataset consists of 13'759 labeled images, with an **imbalanced class distribution** and a significant presence of **outliers**. Key challenges include:

- **Low intraclass variability** and **high interclass similarity**, making classification difficult.
- Differences between the training dataset and the unseen test set on Codabench, complicating generalization.

These factors increase the risk of **overfitting** and hinder model performance. To address these challenges, we made the assumption that the dataset is **insufficient to train a deep learning model from scratch**. Based on this, we adopted *transfer learning* and *fine-tuning* techniques, leveraging pre-trained models to retain general features while focusing on task-specific adaptation.

3 Method

This section outlines the preprocessing steps, model architecture, training strategy, and evaluation process adopted for the blood cell classification task.

3.1 Preprocessing

To prepare the dataset for training, the following steps were undertaken:

- **Outlier removal**: Ensured the dataset's quality by eliminating anomalous samples.
- **Data splitting**: Divided the dataset into training, validation, and local test sets. The local test set was used to evaluate model performance before submitting to Codabench, which used an independent test set.
- **Data augmentation**: Applied transformations such as rotation and translation using *AugMix* [2] and Keras preprocessing tools

(e.g., `ImageDataGenerator`). These techniques expanded the training set, improved generalization, and addressed class imbalance.

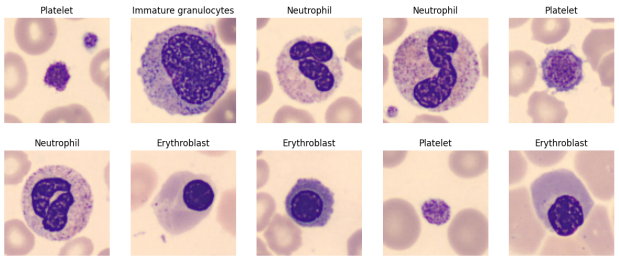

Figure 1: Original dataset

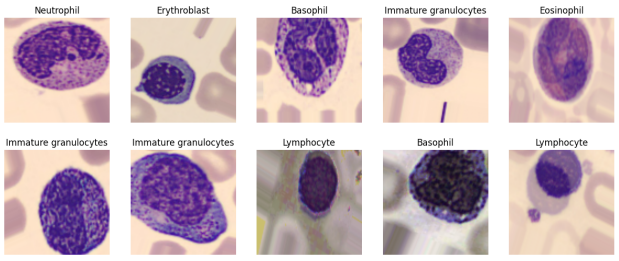

Figure 2: Augmented dataset

3.2 Model

To leverage the power of pre-trained models, we adopted a transfer learning approach using *EfficientNetV2S*[6] as the base model. The key steps are as follows:

1. **Preprocessing:** The input data was passed through the base model's preprocessing function immediately after the input layer.
2. **Model Architecture Enhancements:**
   - Added a *Global Average Pooling* layer to reduce spatial dimensions.
   - Followed it with *Batch Normalization* and *Dropout* to improve generalization and prevent overfitting.
   - Included six blocks, each comprising *Dense*, *Dropout*, and *Batch Normalization layers*, with varying parameters. All Dense layers used ReLU activation, as shown in Equation 1:

$$f^{(l)}(x) = \text{ReLU}\left(W^{(l)}x^{(l-1)} + b^{(l)}\right) \quad (1)$$

3. **Output Layer:** A softmax activation function with 8 neurons was used to output class probabilities, as defined in Equation 2:

$$f(x) = \text{softmax}(Wx + b) \quad (2)$$

4. **Freezing Layers:** To retain pre-trained knowledge, we froze the first 120 Conv2D and DepthwiseConv2D layers of the base model.

3.3 Training

To optimize the model, we implemented the following strategies:

- **Callbacks:**
  - *ReduceLROnPlateau:* Decreased the learning rate.
  - *EarlyStopping:* Halted training when no improvement was observed in validation accuracy for a specified number of epochs.
- **Loss Function:** The model was trained using the *categorical cross-entropy loss* function, defined in Equation 3:

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

  where N is the number of samples, y_i is the true label, and $\hat{y}_i$ is the predicted probability.
- **Optimizer:** The *Adam* optimizer was employed to minimize the loss function effectively.

3.4 Evaluation

The trained model was evaluated on the test set using the *accuracy* metric as shown in the following equation 4.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (4)$$

4 Experiments

The development process was divided into three key stages, each focusing on critical aspects of the project:

- **Data Augmentation**: Various augmentation methods were tested, including SMOTE [1], oversampling, and *AugMix* [2]. Among these, AugMix outperformed the others, achieving a **4% accuracy improvement** over **SMOTE** and a **3.7% improvement** over **oversampling**.

- **Model Architecture**: Several pre-trained models were evaluated as base architectures, including MobileNetV3Small[3], EfficientNetB0[5], EfficientNetV2S[6]. EfficientNetV2S emerged as the best-performing model, achieving superior accuracy and generalization. EfficientNetV2L[7] was also tested but excluded due to its computational cost and marginal performance gains.

- **Fine-Tuning**: The number of trainable layers was varied to identify the optimal configuration. In order, we tried freezing 120, 220, 420, 513 layers. Freezing the first 120 layers of EfficientNetV2S resulted in the best performance, balancing pre-trained knowledge retention with task-specific learning.

5 Results

The proposed model based on EfficientNetV2S achieved an accuracy of 98.95% on the local test set, significantly outperforming EfficientNetB0 and MobileNetV3Small. The use of AugMix provided a notable accuracy improvement of 4% compared to SMOTE. Freezing the first 120 layers during fine-tuning proved optimal, ensuring a balance between pre-trained knowledge retention and task-specific adaptation. These results highlight the model's strong performance and generalization capabilities.

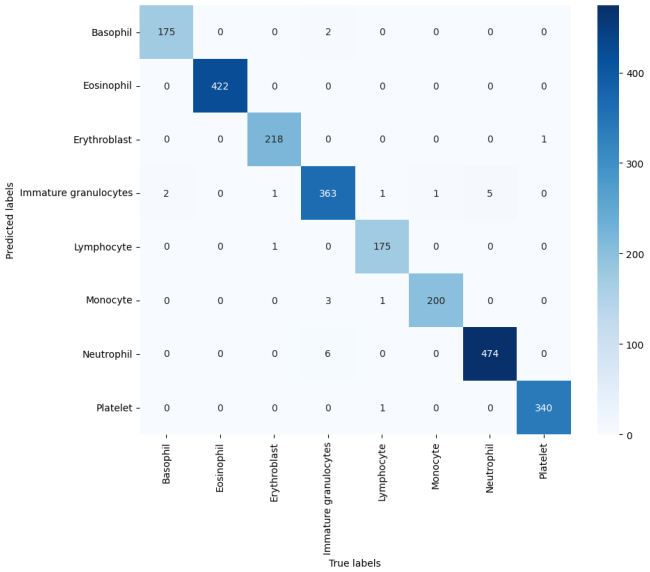

Figure 3: the confusion matrix for EfficientNetV2S

6 Discussion

The results highlight the effectiveness of *EfficientNetV2S* combined with *AugMix* and systematic fine-tuning in addressing class imbalance and dataset variability. The model demonstrates strong generalization capabilities and achieves high accuracy. However, limitations include potential overfitting due to dataset size and reliance on augmentation. Including additional external data or region-of-interest cropping (e.g., using YOLO[4]) could further improve performance but was not pursued due to time constraints.

7 Conclusions

This project demonstrates the effectiveness of transfer learning and fine-tuning for blood cell classification. EfficientNetV2S, coupled with AugMix and optimized layer freezing, delivered strong performance.

8 Contribution

All members of the team contributed equally to the development of the models and writing of this report.

Table 1: The metrics obtained on the local test dataset

Model	Accuracy	Precision	Recall	F1 score
EfficientNetV2S	**98.95 ± 0.12**	**98.96 ± 0.14**	**98.95 ± 0.10**	**98.96 ± 0.13**
EfficientNetB0	98.37 ± 0.18	98.38 ± 0.22	98.37 ± 0.15	98.37 ± 0.17
MobileNetV3Small	85.65 ± 0.80	90.99 ± 0.72	85.65 ± 0.85	86.97 ± 0.78

References

[1] D. Elreedy and A. F. Atiya. A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance. *Information Sciences*, 505:32–64, 2019.

[2] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty, 2020.

[3] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3, 2019.

[4] M. Hussain. Yolov5, yolov8 and yolov10: The go-to detectors for real-time vision, 2024.

[5] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[6] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training, 2021.

[7] U. Waqas, J. Visser, H. Choe, and D. Lee. Multimodal fused deep learning networks for domain specific image similarity search. *Computers, Materials Continua*, 75:243–258, 01 2023.