AN2DL - First Homework Report
DeepL

Matteo Bonfadini, Elena Lippolis, Lorenzo Cossiga, Michele Baggi

50mgk, elenali, lorenzocossiga, mik01

243786, 252310, 242309, 252119

November 24, 2024

1 Introduction

The project aims to implement *multi-class image classification* using **deep learning** techniques for blood cell analysis. This *supervised learning* task requires balancing model complexity to ensure generalization to unseen data while minimizing *overfitting*. The primary **goal** is to design a solid neural network that classifies blood cells into the correct categories with high accuracy.

The **approach** involves dataset analysis to identify potential issues such as class imbalance, image quality, and the need for preprocessing. We then developed simple deep learning models to establish a performance benchmark, which was subsequently refined into more complex models.

2 Problem Analysis

The dataset consists of **13759** RGB images designed for the classification of different types of blood cells. Each image has size 96x96 pixels and it is labeled with one of eight (from 0 to 7) classes, representing the following blood cell types: basophils (0), eosinophils (1), erythroblasts (2), immature granulocytes (3), lymphocytes (4), monocytes (5), neutrophils (6), and platelets (7). We show in Figure 1 a sample for each class.

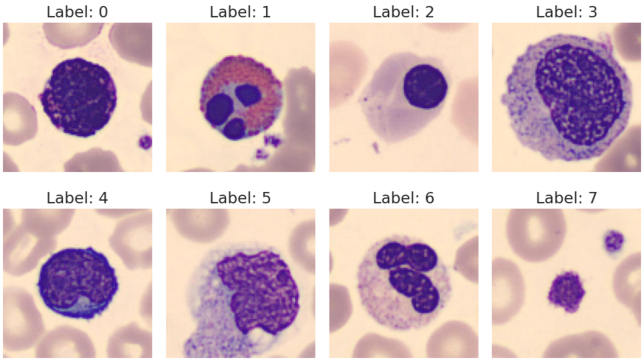

Figure 1: Example of an image for each class

We tackled the problem in a 3-phase approach.

2.1 Phase 1: Data Inspection

Firstly, we inspected the dataset and printed its shape. Secondly, a deeper inspection revealed the presence of *obvious* outliers, as shown in Figure 2, which could skew the learning process: 200 samples of the very same blood cell for each class and 200 rick roll. Therefore we proceed by erasing those images.

We also noticed some images with grayscale backgrounds, but further analysis through *PCA* based on the Mahalanobis distance [2] (normalizing the dataset) showed this pattern did not negatively influence our performance. This indicates that the grayscale background images do not im-

pact the model's ability to learn effectively.

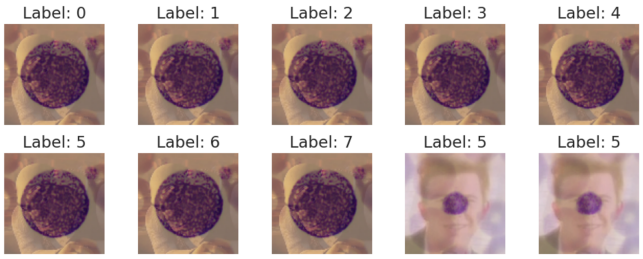

Figure 2: Samples of *obvious* outliers.

2.2 Phase 2: Class Imbalance

The new dataset, for a total of **11959** blood cell images, has shape reported in Figure 3.

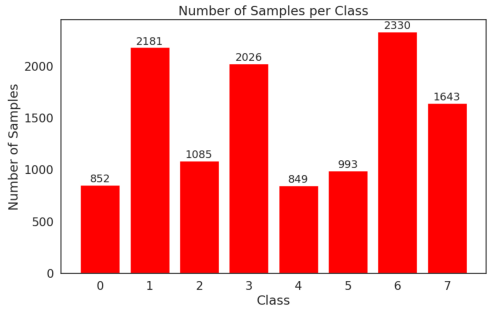

Figure 3: Number of samples in each class.

Upon closer inspection, one might observe that the dataset exhibits class imbalance. This problem poses additional challenges for training a model that generalizes effectively across all categories.

To address this, after the splitting of the dataset into training, validation and test sets according to an 83:13:4 ratio, we balanced the training one by replicating images in the underrepresented classes. The result is a new training set with 2000 images for each class.

Throughout the challenge we noticed that the validation accuracy and the (local) test accuracy were almost identical, as a matter of fact in some training session we just split the data 83:17, therefore we are not going to show the test accuracy for the various models as it would be redundant with respect to the validation one.

Finally, we applied **data augmentation techniques** to create variations of these images (*oversampling*) [5]. In particular, after trials and errors, we decided to use `AugMix` and `RandAugment` [3].

2.3 Phase 3: Mixed Precision

One of the main challenges we encountered was the low training speed, which was solved by implementing mixed precision [1]. This technique uses lower precision (16-bit floating point numbers) for computations while retaining single precision (32-bit) for weight updates. Mixed precision accelerated training and reduced energy consumption, all while maintaining a reasonable accuracy level.

2.4 Overfitting and Assumptions

Overfitting of the training set reduced the model's ability to perform well on unseen (and possibly *modified*) test data. We intentionally allowed initial overfitting assuming that we needed to verify that the model was able to fit the training data before addressing generalization issues, to employ only later techniques like early stopping and dropout layers to mitigate overfitting and enhance performance on unseen data [4].

Finally, we assumed the dataset to be solid, with minimal labeling error or inconsistencies. Moreover, we hypothesized that a simple baseline model would provide reasonable performance before moving to more complex architectures.

3 Method

To tackle the classification task, we employed a **transfer learning** approach, leveraging pretrained models as feature extractors. Specifically, we experimented with three different architectures: VGG16, EfficientNetV2B0, and an additional model structurally identical to VGG16.

For the VGG16-based model, the architecture consisted of a dense layer with linear transformation, batch normalization to stabilize and accelerate training, ReLU activation to introduce nonlinearity, a dropout layer (dropout rate = 0.3) for regularization, and a final dense layer with softmax activation for multi-class classification:

vgg16
Dense(256)
Batch Normalization
Dropout
Dense(8)

For EfficientNetV2B0, we used a more complex setup to improve generalization and robustness:

effNetV2B0
Augmentation
Dense(256)
Batch Normalization
Dropout
Dense(256)
Batch Normalization
Dropout
Dense(8)

The models were built using the Keras Functional API, linking the input layer to the custom head via the frozen vgg16/effNet feature extractor. The model were later compiled with a loss function (Categorical Cross-Entropy, designed for multi-class classification tasks), and the **Lion optimizer** [6]. Furthermore, during the training phase, we exploited the callbacks **early stopping**, **reduce learning rate** and **module checkpoint** to prevent overfitting and adjust the learning rate.

4 Experiments and Results

Initially, we implemented a simple model based on the CIFAR-10 architecture. While it achieved acceptable performance on the validation set, its accuracy on the hidden test set was suboptimal.

To address this, we developed a more complex model identical to VGG16 but trained from scratch with randomly initialized weights. Although this approach slightly improved test accuracy, the gains were limited.

Subsequently, we shifted to transfer learning, incorporating pre-trained models (VGG16 and EfficientNetV2B0). For VGG16, we added a custom classification head as described earlier and fine-tuned the model. This approach achieved the best test set accuracy of 82%.

We also experimented with EfficientNetV2B0 with additional dense and augmentation layers.

The results on both validation and hidden test set are shown in Table 1. These results emphasize that simplicity, combined with fine-tuning of higher convolutional layers in pre-trained models, can yield better results in our scenario.

Table 1: Various models accuracy.

Model	Val. Acc.	Hidden Test Acc.
Cifar10	83.24%	25%
vgg16_{RI}	82.59%	39%
vgg16_{TL}	74.63%	49%
$\mathbf{vgg16}_{FT}$	**97.13%**	**82%**
effnet_{FT}	96.20%	75%

5 Conclusions

This project aimed to solve a multi-class classification problem with a dataset of blood cell images using transfer learning and fine-tuning with VGG16. The algorithm includes augmentation, regularization, and training techniques, achieving high accuracy in both validation and test datasets, and demonstrating its potential applicability in medical image analysis.

The final VGG16 model achieved high performance on validation and test accuracy, demonstrating the effectiveness of transfer learning and fine-tuning. Leveraging the pre-trained VGG16 model allowed a more effective extraction and use of complex visual features, reducing the need for training from scratch and speeding up convergence. Augmentation and regularization techniques helped mitigate overfitting and improve generalization across all blood cell classes, and the use of mixed precision reduced computational overhead and accelerated training.

Several areas for improvement in our models include addressing some **weaknesses**, such as the drop in accuracy from validation to test data, indicating the model could still struggle with generalizing to unseen data. In addition, given the dataset's initial class imbalance, it is possible that certain underrepresented blood cell types experienced lower performance metrics. A further improvement in our algorithm could be adjusting the loss function to prioritize minimizing critical misclassification and using more sophisticated methods for dealing with class imbalance.

To conclude, even though we did most of work together, we highlight here the main contribution of each team member: Lorenzo and Michele developed the VGG16 and EfficentNet models, while Elena and Matteo dealt with data inspection and augmentation as well as the writing of this report.

References

[1] M. Dörrich, M. Fan, and A. M. Kist. Impact of mixed precision techniques on training and inference efficiency of deep neural networks. *IEEE Access*, 11:57627–57634, 2023.

[2] H. Ghorbani. Mahalanobis distance and its application for detecting multivariate outliers. *Facta Universitatis Ser. Math. Inform.*, 34, 2019.

[3] lukewood. Cutmix, mixup, and randaugment image augmentation with kerascv. Online available at `https://keras.io/guides/keras_cv/cut_mix_mix_up_and_rand_augment`, April 2022.

[4] M. Matteucci. Facing overfitting lecture. October 2024.

[5] C. Shorten and T. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 2019.

[6] TensorFlow Developers. Lion optimizer, 2024. `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Lion`.