AN2DL - First Homework Report
DreamTeam

Alisa Pesotskaia, Safuan Iusupov, Davide Paltrinieri, Dario Napolitano
allythe2003, safuanlip, dpaltrinieri, darionapolitano
272533, 272547, 276139, 275560

November 24, 2024

1 Introduction

This project aims to *classify* blood cells by implementing **deep learning** techniques.

Our approach began with **data inspection**. Subsequently, we tested models with increasing complexity, starting from a simple CNN designed by us. Additionally, advanced data augmentation techniques were applied to enhance the model's robustness and generalization capabilities.

2 Problem Analysis

2.1 Dataset

The task involves supervised classification of microscopic images of blood cells, categorized into 8 classes representing different cell types. The dataset comprises a total of **13,759** RGB images, each with three channels and a resolution of 96×96 pixels.

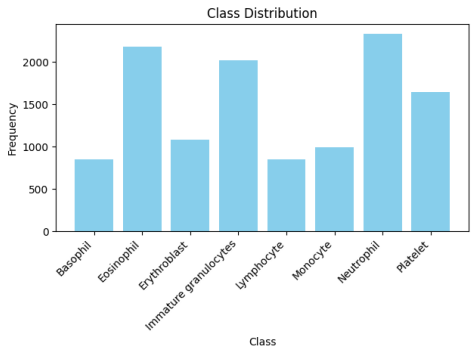

Figure 1: Class Distribution

2.2 Method

To inspect the dataset, we visualized some sample images from each class and analysed the class distribution. As displayed in Figure 1 a slight class imbalance was found and corrected using `sklearn`. Generally, the following techniques have been implemented for models training:

- Data normalization by 255.0;
- Cross-entropy loss function (Equation 1);

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

- 80-20% dataset split for training-validation;
- Dropout.

3 First attempt

3.1 Simple CNN

As the first step, we trained a simple CNN model with around **50,000 parameters**. To reduce the number of parameters, the images were resized to 36×36 pixels and no augmentation was applied.

The validation accuracy reached **99%**, whereas the Codabench submission revealed an accuracy of **26%**. This significant discrepancy highlighted that the test data differs dramatically from the provided dataset. Two hypotheses were made:

1. The given dataset might contain **outliers**;
2. The test dataset might be **augmented**;

3.2 Outliers

We suspected that Eugenio Lomurno had some surprises for us, especially when he shared links to 10 hours of "*Shreksophone*" and "*Never Gonna Give You Up*". After a closer look, we noticed some **outliers**, specifically, some images of the dataset were altered with *unconventional backgrounds*, as displayed in Figure 2.

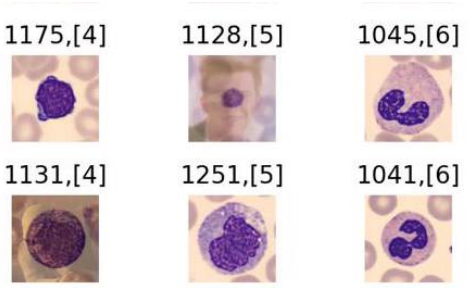

Figure 2: Example of **outliers** in the given dataset (image n. 1128 and image n. 1131).

After removing all duplicated and altered images, the dataset was reduced to **11,943** images.

A simple CNN was then re-trained on the refined dataset. However, the test accuracy surprisingly decreased, suggesting that the outliers may have contributed to better generalization and higher test performance, at least, under that specific training setup and network configuration.

3.3 Wrong path

As an alternative approach to the problem, we attempted to extract images such those shown in Figure 3 to utilize them to augment the dataset by incorporating them as backgrounds.

Figure 3: Extracted shots, similar to those in the outliers, featuring *Shrek* and *Rick Astley*.

To implement this augmentation, designing a custom augmentation layer was necessary. However, unfortunately, training the model (simple CNN) on the dataset with augmented backgrounds **did not lead to an improvement** in the test score.

Given such an outcome, from this point forward, we opted to work with the data without outliers and experimented with training models of varying complexity. Some of used models were suggested in [1].

We also kept the original image size unchanged. Additionally, we applied **simple augmentations** [8] and, for some models, **complex augmentations**, as illustrated in Figure 4 and Figure 5.

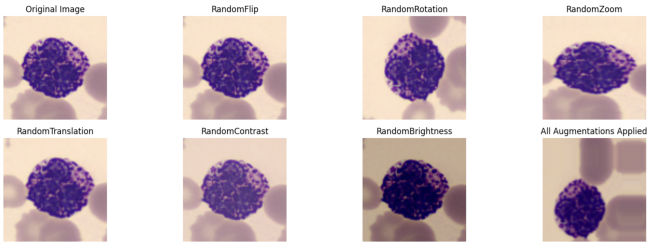

Figure 4: Simple augmentations: random flip, rotation, zoom, translation, contrast and brightness.

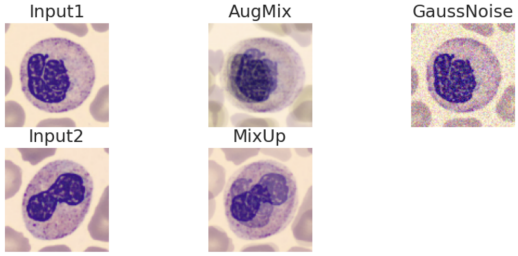

Figure 5: Complex augmentations: AugMix [3], Gaussian noise [4] and MixUp [5].

4 Experiments

During the experiments with different approaches, our team members did the following:

1. **Alisa** explored the data and found **outliers**, focused on **fine-tuning and transfer-learning** using InceptionNet, MobileNet , ResNet and EfficientNet with simple and complex augmentations. Implemented **ensembles** of trained models.

2. **Safuan** focused on **fine-tuning and transfer-learning** using EfficientNet, Xception and ConvNextBase with simple and complex augmentations. **Balanced** the class distribution by over-sampling, suggested in [2]. Implemented a **custom augmentation layer**.

3. **Davide** implemented **Inception CNN** with varying number of layers, used **fine-tuning** and **transfer learning** techniques in his experiments, searched for **complex augmentations**, used GaussianNoise and AugMix and trained a family of MobileNets. Used **class balancing** from Keras.

4. **Dario** implemented and tested **Simple CNN**, after focused on **transfer-learning and fine-tuning** using MobileNetV3Large and NASNetMobile, **prepared MixUp** and **RandAugment** data for training models with complex augmentations.

Table 1: Summary of main experiments

Model Name	Parm.	Augm.	Train.	Score
SimpleCNN	52K	B	B	0.21
SimpleCNN+IncLayers	201K	B	B	0.30
IncNet (20 layers)	204K	S	T	0.41
MobileNetV2	2M	S	F	**0.62**
MobileNetV2	2M	C	F	0.52
MobileNetV3Large	3M	AugMix	TCR	0.41
MobileNetV3Large	3M	S+AugMix	TCR	0.45
MobileNetV3Small	1M	S+MixUp	FCR	0.33
EfficientNetV2s	20M	S	TR	**0.62**
EfficientNetB2	8M	S	F	0.42
EfficientNetB2	8M	C	B	0.27
Xception	20M	C	T	0.44
Xception	20M	S	F	0.49
NASNetMobile	4M	S	F	0.54
ConvNeXtBase	87M	B	TR	0.32
ConvNeXtBase	87M	MixUp	T	0.41
MobileNetV2+EffNetV2s	22M	S	FCR	**0.70**

Referring to Table 1, which displays the outcomes we achieved, please note the following:

Augmentations

- S = Simple: rotation, flip, translation, zoom, brightness, contrast;
- C = S + MixUp;
- B = S + Class balancing [7].

Training Techniques

- B = Basic: Adam optimizer, learning rate decay, early stopping;
- T = B + Transfer-learning;
- F = T + Fine-tuning;
- TR = T + Learning rate scheduler (LRS);
- TCR = TR + Class Weighting (CW);
- FCR = F + CW + LRS;
- E = Ensemble.

Score: Refers to **test accuracy** achieved during the development phase on Codabench submissions.

5 Results

Our main findings can be summarized as follows:

- Our best model is an **ensemble** that performs the average between two models which showed **the best** results. Such technique allowed us to achieve an **8% increase in accuracy** (from 62% to 70%).

  This improvement can be attributed to the diversity in the features learned by the two models. As suggested in [6], shallower models learn generalised features, while deeper networks learn more semantically meaningful features. The ensemble combines these complementary strengths.

- **Simple** augmentations **improve** model performance, whereas **complex** augmentations tend to **degrade** it.

- Using a pre-trained model and performing transfer-learning and fine-tuning resulted in the **20% jump in accuracy** on the **test dataset** (from 42% to 62%). However, the fine-tuning phase often led to **marginal improvements**, particularly for large models.

  This stems from large pre-trained models having highly generalized feature representations. When fine-tuned on a smaller dataset, these models are prone to overfitting. As a result, the benefit of adjusting the pre-trained weights is sometimes negligible compared to the performance already achieved through transfer learning alone.

- Increasing the number of parameters of the model did not always translate to better performance. For instance, MobileNetV2 with 2M parameters **outperformed** ConvNeXtBase with 87M parameters by **21%**.

6 Discussion

At times, the performance differences between models seemed inconsistent and counterintuitive. For instance, while increasing parameters led to a 20% performance improvement when transitioning from EfficientNetB2 (8M) to EfficientNetV2s (20M), it did not result in any noticeable gains with MobileNetV2 (2M) and MobileNetV3Large (3M).

7 Conclusions

This challenge allowed us to gain practical experience with several network training techniques and improve our understanding of their impact. There is still much to explore, including testing more advanced augmentations and optimizing ensembling methods. Although the process was challenging, it was both engaging and rewarding, leaving us enthusiastic for further competitions and opportunities to apply these skills.

References

[1] Rabia Asghar, Sanjay Kumar, and Paul Hynds. "Automatic classification of 10 blood cell subtypes using transfer learning via pre-trained convolutional neural networks". In: *Informatics in Medicine Unlocked* 49 (2024), p. 101542.

[2] Xinjian Guo et al. "On the Class Imbalance Problem". In: *2008 Fourth International Conference on Natural Computation*. Vol. 4. 2008, pp. 192–201. DOI: 10.1109/ICNC.2008.871.

[3] Keras. *AugMix*. 2024. URL: https://keras.io/api/keras_cv/layers/augmentation/aug_mix/.

[4] Keras. *GaussianNoise*. 2024. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/GaussianNoise.

[5] Keras. *MixUp*. 2024. URL: https://keras.io/examples/vision/mixup/.

[6] Ashnil Kumar et al. "An Ensemble of Fine-Tuned Convolutional Neural Networks for Medical Image Classification". In: *IEEE Journal of Biomedical and Health Informatics* 21.1 (2017), pp. 31–40. DOI: 10.1109/JBHI.2016.2635663.

[7] scikit-learn. *ClassBalancing*. 2024. URL: https://scikit-learn.org/dev/modules/generated/sklearn.utils.class_weight.compute_class_weight.html.

[8] TensorFlow. *Data augmentation*. 2024. URL: https://www.tensorflow.org/tutorials/images/data_augmentation.