AN2DL - First Homework Report
Team Name: FORmidable

Matteo Pompilio, Tanguy Rolland, Kalisto Willaey

tresp, tanguyrlld, kalisto

248966, 242816, 242848

November 24, 2024

1 Introduction

In this first challenge, we face an *image classification* problem on a blood cells dataset. Our goal was to reach good accuracy on the remote test set while achieving a great understanding of the methods to reach the best *robustness*.
We have used many **data augmentation** techniques and trained a variety of **CNNs**, while optimizing **hyperparameters** for a more efficient training.
Our approach to this challenge was to keep looking into the further improvements we could add to our process pipeline, starting from our naive intuitions and iteratively exploring more pertinent assets such as more efficient **loss functions** and **optimizers**.
We quickly understood that success wouldn't necessarily lie with the biggest models but with the most coherent and adaptative pipeline.

2 Problem Analysis

The dataset is initially composed of about 13800 cell images, distributed among 8 classes with non-balanced proportions. After looking at the data and as indicated in the literature [1], some cells types can be harder to differentiate than other. Immature Granulocytes are future Basophils, Eosinophils and Neutrophil and all look alike. We could anticipate difficulties in classification.

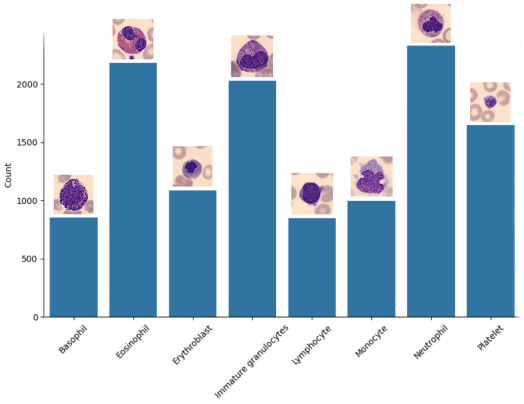

Figure 1: Class distribution in the dataset after cleaning

We performed Principal Component Analysis (PCA) on the image features extracted by a pretrained MobileNetV2 in order to visualise the classes in their latent representation (notebook attached).

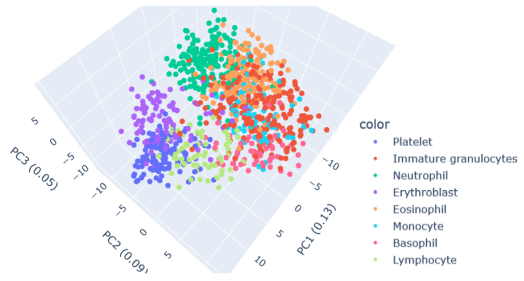

Figure 2: Images features from pretrained MobileNetV2 in the first principle plane of the PCA

This representation allows to estimate class separability and confirms that Immature Granulocytes have common features with many other classes as Monocytes and Basophils, while other classes like Platelets, Neutrophils or Erythroblast are expected to be easier to classify. We can adopt a class-weighing strategy to improve confusion scores.

3 Method

From the raw dataset, we had to apply the appropriate preprocessing steps to better understand the data and clean it before feeding it to the network. No missing values or NaN were identified; however, after visualizing random selections of images, we discovered about 1800 duplicates (especially repeated image having different labels, like in Figure 3), which we dropped.

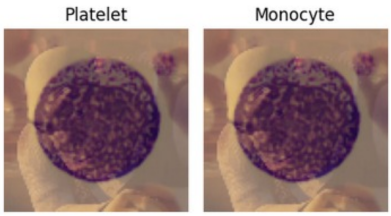

Figure 3: Shrek easter egg and duplicates

The class distribution wasn't balanced, so we tried balancing it through simple data augmentation (flips, crops, shifts...) and via class weighing (see next paragraphs).

We then split the dataset into train, test and validation sets, holding respectively 72%, 20% and 8% of the data.

Finally, we integrated these augmentations into a `tf.data` pipeline, to ensure efficient data loading and preprocessing. We applied, in order: the augmentation pipeline, the resizing of all images (only to meet the requirements of our pre-trained model, from 96x96 to 224x224, via bicubic interpolation), and, finally, normalization in the [-1, 1] range (also to match with MNV2, so we ended up using this range for all our models) .

Our initial modelling approach (basic model) involved building a simple neural network layer by layer. The model consisted of the following components: Input Layer, Convolutional Layers, Pooling Layers, Flatten Layer, Dense Layer, Output Layer. We compiled this model using the Adam optimizer with a specified learning rate and a categorical crossentropy loss, which formula is:

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

where N is the number of classes, y_i is the true label (one-hot encoded), and $\hat{y}_i$ is the predicted probability for class i. Note that we also tried a custom weighted version of this loss, setting higher weights for the classes that showed the lowest accuracy. However, this did not especially improve our overall accuracy and on the contrary, it reduced it (see table in chapter Experiments).

This initial model allowed us to understand the basic operations of convolutional layers, activation functions, and pooling layers, but its simplicity limited its performance for augmented datasets.

To enhance performance, we transitioned to using the MobileNetV2 architecture with pretrained weights from ImageNet, leveraging transfer learning, as we were looking a light, efficient and recent architecture. For fine-tuning, we unfroze the base model and only trained the top layers, while freezing the initial layers to retain the pretrained features.

For the final model, our decision was to implement a MobileNetV3 without pretrained weights. We took the architecture on the framework provided in the notebook of Lecture 4. Following the best practices advised for MobileNetV2 [2], we increased the number of filters as the network depth increased, as this principle also applies to the newer version. As recommended by its creators [3], we employed the Hard-Swish activation function:

$$h_{swish}(x) = x \cdot \frac{\min(\max(0, x+3), 6)}{6}$$

Additionally, we replaced Adam with the Lion optimizer, known for its adaptive learning rates and robust performance [4].

4 Experiments

First, as for the augmentation process, instead of preprocessing the data statically, we opted to create a pipeline that applies augmentations on-the-fly during training: this approach would indeed reduce memory usage and allow the model to see different variations of the same images across epochs. We

introduced specific augmentation layers to emulate medical imaging scenarios as close as possible:

- **RandAugment**: to maximize generalization by applying diverse augmentations, with a magnitude of 0.2
- **RandomSharpness**: although included in RandAugment, here emphasized, to simulate the varying focus in medical scans
- **GridMask**: to simulate real-world scenarios where cells might not be fully captured, with a factor of 0.4
- A **customized Gaussian noise layer**: reflecting the noise commonly found in digital medical images

As this final augmentation pipeline was defined (with respect to initial pipelines, now with higher magnitudes/factors, more augmentations per image and introduction of noise) the MobileNetV2 model was overfitting at early epochs (as you can see from our notebook), resulting in poorer accuracy scores on the hidden dataset; we hence decided to introduce the custom MobileNetV3 architecture, in a light version with around 2M parameters and a heavier one, composed of about 40M parameters.

Each model was compared on local test split accuracy, f1 score and codabench accuracy.

5 Results

Our incremental approach allowed us to compare many models, and the modern architectures wielded a clear increase in performance relatively to more basic CNNs. However, heavy models wouldn't necessarily show a huge improvement compared to lighter ones, gaining only few points of accuracy. Our final best result was of 80% of accuracy on the Codabench set, while performing as well as 98% locally, with a Heavy MobileNetV3. All results are summarized in 1).

Data Augmentation was a key point of this challenge, and our pipeline was refined all along. We understood how robustness was the main point to match hidden real-world data.

Even though we tried different strategies to better classify the hardest classes (via weighted loss function) our performance scores kept being undermined by misclassified Immature Granulocytes and Monocytes.

Figure 4: Confusion Matrix of our best performing models

6 Discussion

Our approach showcased several key strengths:

- Development and deployment of robust, modern architectures.
- The use of both pretrained and fully trained networks.
- Notably, the MobileNetV3 model, trained entirely without pretrained weights, achieved the highest local test set accuracy (97.62%) and a *Codabench* accuracy of 80%.

However, our approach had certain limitations. Despite extensive efforts in data augmentation and architetural adjustments, the disparity between the local test accuracy and Codabench performance indicates a mismatch between the hidden test dataset and our training augmentations. This highlights the need for more diverse or adaptive augmentation strategies to better account for unseen data variations.

Additionally, our attempts to address class imbalance, particularly for challenging categories like *Immature Granulocytes* and *Monocytes*, yielded only limited improvements, as evidenced by persistent misclassifications.

Table 1: Our testing progresses. Best results are highlighted in **bold**.

Model	Parameters	Accuracy	Recall	Codabench Accuracy
Basic model	314k	0.8989	0.8984	0.32
MobileNetV2 pre-trained	2.7M	0.8218	0.8231	0.68
MobileNetV3 light	1.7M	0.9640	0.9642	0.77
MobileNetV3 heavy	**42.3M**	**0.9762**	**0.9763**	**0.80**
MobileNetV3 heavy with custom loss function	42.3M	0.9762	0.9762	0.69

Finally, time constraints restricted our exploration of more advanced techniques beyond those covered in class. Incorporating insights from recent state-of-the-art papers could have helped us develop more sophisticated augmentation strategies and implement optimized custom loss functions for improved performance.

7 Conclusions

In summary, we had the opportunity to explore many different paths and optimize every element of our process, from data inspection to model training. Our main achievements were: an advanced data augmentation pipeline, reaching a very satisfying accuracy without using any big pretrained model, and refining our loss and activation function to really adapt to the challenge.
We would have liked to explore more into the custom loss and potentially other model architectures. Finding the right amount and type of data augmentation also seemed to remain an open question.
Finally, we state the team members' duties along this homework. Each one has contributed to the whole process during the challenge, and additionally: Matteo has trained the models on his laptop with a CUDA-compatible GPU and defined the augmentation techniques; Tanguy especially worked on data analysis and experimented the weighted loss function; finally, Kalisto mostly worked on building the models architecture.

References

[1] Alyssa Tigner, Sherif A. Ibrahim, and Ian V. Murray. *Histology, White Blood Cell*. StatPearls Publishing LLC, 2024.

[2] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE*, 2018.

[3] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. *ICCV*, 2019.

[4] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic Discovery of Optimization Algorithms. *arxiv*, 2023.