AN2DL - First Homework Report
Deepmindset

Giacomo Giovanni Papa, Marta Zecchini, Leonardo Salvucci, Alessandro Torazzi

giacomopapa, martazecchini, leonsalvu, atorazzi

252855, 246790, 259585, 252725

November 24, 2024

Contents

1 Introduction

The goal of this project was to **design and train a deep learning architecture** for *multi-class image classification* problem, with a focus on achieving both efficiency and high accuracy on the test dataset. Specifically, the classification task involved identifying blood cells: the dataset consisted of nearly 13,000 images divided into 8 classes, each representing a distinct cell type or state within the human circulatory system.

2 Pre-processing

We started out with a dataset of 13,759 96x96 RGB images, which we carefully inspected through data visualization, identifying groups of duplicate images. Concerned about other outliers we used techniques such as *t-SNE analysis* (figure 1), to detect distorted images, including some with backgrounds featuring Shrek or Rick, which could have compromised the neural network's ability to learn effectively. To ensure a more reliable dataset for training, we subsequently removed these images.
Additionally, the *t-SNE analysis* revealed other types of outliers, such as ones containing two different cells but assigned to a single label. We chose to retain these samples since similar cases might appear in the test set as well, and we wanted the network to be robust to such scenarios.

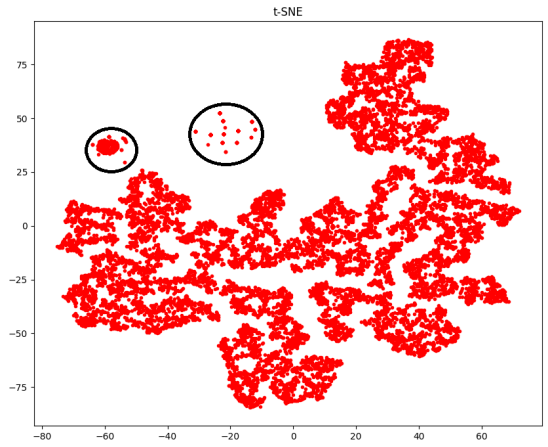

Figure 1: t-SNE Results

After these steps, the dataset resulted reduced to 11,959 images. At this point we analyzed class distribution and we found a significant imbalance, with number of samples per class ranging from approximately 850 to 2,330 images. **After splitting the dataset into training, validation, and test sets**, we addressed this situation by balancing the training set to prevent biased learning and poor performance on minority classes. Instead of simple oversampling with identical copies, we applied **geometric augmentations**—including rotations, translations, and flips, to increase the sample size. The training set was adjusted to 2,500 images per class, ensuring an equal representation across all categories.

3 Model construction and adopted strategy

After several unsuccessful attempts with a *custom-built CNN*, it became evident that employing a pre-designed network was essential for effectively learning the data features. Consequently, we adopted a transfer learning approach with fine-tuning.
Our primary objective was to design a neural network that was as lightweight as possible. We hypothesized that a light and agile architecture would enable faster training times, allowing us to perform extensive trials to fine-tune hyperparameters and better evaluate the impact of our modifications.

3.1 Model description

We tested several pre-trained ImageNet networks, such as EfficientNetB0 and MobileNetV3Small, reaching the best performance with **MobileNetV3Large**: a neural network specifically designed for mobile and edge applications, which strikes a great balance between computational efficiency and accuracy. This model integrates *Inverted Residual Blocks* with *Squeeze-and-Excitation modules*, further optimized through *Neural Architecture Search (NAS)*. The core of MobileNetV3Large relies on the classical MobileNet layers, subdivided into depth-wise and point-wise convolutional blocks [2].
These features are specifically designed to ensure optimal feature extraction while keeping computational costs low, thanks to the limited number of weights and parameters. Following the convolutional base, we included an optional *dropout* layer, a dense layer with 128 units using *Relu* activation, and an output layer with the option to apply *L2-Regularization*. For backpropagation we chose the Adam optimizer.

3.2 Training

We started with the Transfer Learning (TL) phase, where we used a pre-trained model and applied techniques like L2-regularization ($\lambda = 5 \cdot 10^{-3}$) and the Adam optimizer [3]. We also used two callbacks: *Early stopping* to conclude training when validation accuracy stopped improving, and *ReduceLROnPlateau* to adjust the learning rate when needed. This phase resulted in a validation accuracy of around 90%. We then proceeded with Fine-Tuning (FT), nowadays considered a must when approaching Trasfer Learning[4]. We kept frozen the first 124 out of 187 total layers to preserve their pre-trained weights

Table 1: Comparison of different model performances. Values are weighted and expressed as percentages (%).

Model:	$\mathbf{ValAcc}_{\text{local}}$	$\mathbf{Accuracy}_{\text{test}}$	**Recall**	**F1-score**
Custom CNN	95.03	24	95	95
EfficientNetB0	91.63	56	92	92
MobileNetV3Large	**97.66**	**82**	**97.6**	**97.6**

while unfreezing only the Conv2D and Depth-wise Conv2D layers in the remaining part, keeping BatchNorm layers frozen. For the training we confirmed the same callbacks, but we chose to increase the λ to 10^{-2}, since in these 50 layers was condensed the great part of the weights. This approach improved the validation accuracy by 6-7%, with the best model achieving a final accuracy of 97.66%.

3.3 Other attempts

Before finding our optimal combination, we made several attempts with other Networks and other optimizers. The table 1 presents results highlighting the superior performance of the MobileNetV3Large model compared to other pre-trained networks. Regarding the optimizers, as shown in Table 2, we compared Nesterov SGD, Lion and Adam. The first one performed surprisingly well during the TL phase but fell short during Fine-Tuning, delivering lower results than the others. Lion, on the other hand, produced good results but still not better than Adam ones.

Table 2: Comparison between different optimizers before and after fine tuning (FT).

Optimizer	$\mathbf{TL}_{\text{accuracy}}$	$\mathbf{FT}_{\text{accuracy}}$
SGD	**91.32**	95.51
Lion	88.17	96.73
Adam	88.79	**97.66**

We also experimented with the Dropout regularization technique, both in combination with L2-regularization and on its own. However, the final accuracy was consistently worse than our previous results, possibly because we were unable to find the optimal hyperparameter settings.

4 Data Augmentation

Data augmentation[7] and in particular the RandAugment layer turned out to be fundamental in our solution, as it was in fact the aspect that gave us the greatest step up (30%) in the general test-accuracy. This technique, as highlighted in [1], should increase model robustness by exposing it to diverse and realistic data variations, effectively reducing overfitting. Despite that, for long time we used only the standard keras augmentations, as we were worried about the amount of distorsion that layers like `RandAugment` would have brought to the images. Moreover, trying to introduce this layer in the structure of the NN seemed to lead us to the impossibility to train the model, because of its great computational costs. The step forward was made when, studying on keras examples, we found a way to parallelize the application of this layer in a pipeline, so that the augmentation would have been done by batches. We were then pleased to notice that, not only the Network was still able to learn the right features, but also that our model had become much more robust to realistic variations in the data, as proved by the test accuracy achieved. Specifically, in our case we obtained the best results applying two random augmentations per image with a magnitude of 0.4 (stddev 0.15) and an 80% application rate, including geometric augmentations like shearing. To further improve spatial diversity, we added more `RandomFlip` and `RandomRotation` layers to the pipeline. The augmented dataset was then **shuffled** so that at every epoch a different version of the same data was seen by the network.

4.1 Alternative unsuccessful data augmentation approaches

We also experimented with alternative augmentations, which proved to be unsuccessful for us:

Salt and Pepper Noise introduces random black and white pixels into the image. We were unable to achieve good results with this technique, and we supposed that the cause may be that it can obscure critical details such as edges and internal structures, making accurate classification more challenging.

Gaussian Noise simulates sensor noise by adding random values to pixel intensities, following a normal distribution. Disrupting biological patterns, reducing contrast, and blurring contours, we concluded that this augmentation didn't work with our model because it could have complicated the recognition of fine structures in cellular images.

CutMix: This augmentation combines two images by replacing a rectangular region of one image with a patch

from another. The labels are also mixed proportionally to the area of the patch. While CutMix encourages the model to learn features from different parts of the images and reduces overfitting by regularizing the network [8], its applicability to cellular images is limited. In our case, we applied CutMix to the classes with the highest error rates, achieving excellent local performance (99%). However, this performance did not generalize well, with accuracy dropping to 76% on unseen test data.

4.2 Validation Set Augmentation

We applied also some augmentation to the validation set, focusing specifically on classes 3, 5, and 6, as these were the most confused by the network. Geometric transformations, such as rotation and flipping, were applied to these classes. The goal of this procedure was mostly to have a more accurate estimate of how well the network managed to correctly predict those problematic classes

4.3 Test Time Augmentation (TTA)

Test-Time Augmentation (TTA) is a technique that applies augmentations to images during the testing phase, generating multiple augmented versions of each image. The predictions for these augmented images are then averaged to produce the final class probabilities. As demonstrated in [6] this strategy should guarantee the expected error to be less than or equal to the one of the same model without TTA. In our application, TTA has been employed to further enhance the robustness and accuracy of the model. Specifically, for our dataset of 8 cell classes, each image was augmented 5 times during testing with geometric transformations and changes in brightness and contrast. Among the options used, contrast augmentation proved to be the most effective on the output, as variations in its magnitude parameter had the most significant impact on the network's performance. In general the application of TTA significantly contributed to aligning the validation error with the hidden test set error. This improvement was particularly important for ensuring model generalization, given the complexity and variability of cell images, which can present unique challenges to accurate classification.

5 Discussion

Despite having achieved satisfactory results with our network, we noticed recurrent discrepancies between the accuracy on our test set (figure 2) and the hidden one. In fact we never managed to completely close the accuracy gap between the two, despite always ensuring our test set independency from training and validation set. We think it is possible that different and more advanced data augmentation combinations could have produced better results. Another point to be mentioned is that the vast array of possible pretrained models, limited resources, and unexpected testing issues prevented us from exploring many other architectures. This leaves us with the open question of whether a more complex network could have further improved classification accuracy.

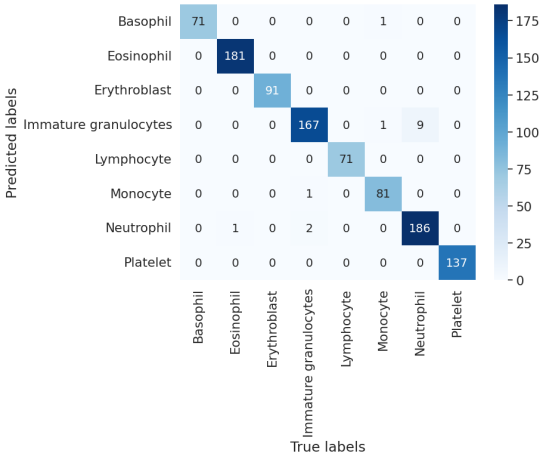

Figure 2: Confusion matrix on local test set

6 Conclusions

In conclusion, our best model consists of a pre-trained Mobilenet convolutional network, combined with additional layers and fine-tuning and other startegies, that made this model tailored for this specific task, allowing us to achieve a promising accuracy of 82%. While this result is already significant, there is potential for further improvement by exploring alternative solutions. Our proposals for future work include:

- **k-fold cross-validation**: This technique could enhance the robustness of our model and improve its generalization capabilities. However, due to limited resources and the high computational cost, we were unable to implement it. Nevertheless, it remains a promising avenue for future exploration.

- **Lion optimizer**: According to the literature, this optimizer has demonstrated superior performance compared to Adam [5]. A future direction could involve a deeper analysis of this optimizer and its successful integration into our network. The Lion optimizer is particularly effective with a high batch size and a low learning rate. Although we conducted preliminary experiments, the results were not outstanding. With further studies and greater computational resources, better performance could potentially be achieved.

Contributions

Each team member contributed to the final work as follows:

- **Leonardo Salvucci**: data preprocessing, model development and best model selection, data augmentation, hyperparameter tuning, results comparison, and report writing.
- **Giacomo Giovanni Papa**: data preprocessing, model development and best model selection, dataset oversampling, data augmentation, test-time augmentation (TTA), and report writing.
- **Marta Zecchini**: model development and best model selection, data augmentation, test-time augmentation (TTA), model inference, results comparison, and report writing.
- **Alessandro Torazzi**: data preprocessing (t-SNE), model development and best model selection, data augmentation, test-time augmentation (TTA) and report writing.

References

[1] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv*, 2019.

[2] A. Howard, M. Sandler, G. Chu, et al. Searching for mobilenetv3. *arXiv*, 2019.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[4] Z. Liu, L. Shen, and A. B. Chan. Improved fine-tuning by better leveraging pre-training data. *arXiv*, 2021.

[5] D. J. Morrow et al. Symbolic discovery of optimization algorithms. *arXiv*, 2023.

[6] K. Shanmugam, T. Zhang, A. Jaimes, and K. Grauman. Better aggregation in test-time augmentation. *arXiv*, 2020.

[7] Z. Wang, Y. Yang, Y. Tai, X. Liu, C. Wang, J. Li, and F. Huang. A comprehensive survey on data augmentation. *arXiv*, 2023.

[8] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *arXiv*, 2019.