POLITECNICO
MILANO 1863

AN2DL - First Homework Report
dAI che è venerdì

Alessandro Annechini, Riccardo Fiorentini, Lorenzo Vignoli

annechini, riccardo01, vigno

250905, 249012, 252085

November 24, 2024

1 Introduction

The First Homework for the Artificial Neural Networks and Deep Learning (AN2DL) course involves implementing a deep learning model for image classification. In particular, the goal of the model is to correctly classify blood cells images into eight categories, based on their shape, contour, color, and other factors.

The main objective of this project is to maximize the predictor's *accuracy*:

$$accuracy = \frac{correctly\ predicted\ samples}{all\ samples} \quad (1)$$

To achieve this goal, we employed several state-of-the-art techniques, including data augmentation, advanced model architectures, and hyperparameter tuning. These steps were crucial to build a robust model capable of generalizing to unseen data.

The report is structured as following: in Section 2 we introduce the main challenges, while in Section 3 we detail the methodology we employed to tackle the problem. Section 4 reports the experiments performed while designing our model, and Section 5 shows the final results on the test set. Finally, in Section 6 we list our most important takeaways, along with the main individual constributions.

2 Problem Analysis

The creation of a Deep Learning model requires three main steps: *training*, *validation* and *testing*. Assessing quality and diversity of the images provided for training the model is pivotal for maximizing its performance, as they constitute the set of information over which the model learns to recognize different classes. Some of the challenges that we needed to tackle in this image classification problem are:

- **Data diversity**: the dataset provided for training contained images with the same color palette. In a real world scenario, these models are used in various different settings, and such a complexity may not be captured when training on an homogeneous dataset.

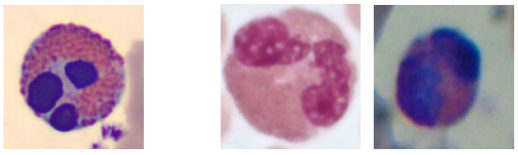

Figure 1: *Eosinophil* image from the provided dataset compared to *eosinophil* images from [14, 17]

- **Incorrect samples**: the pre-labeled dataset for training included incorrect images, that needed to be removed in order for the training process to work correctly.

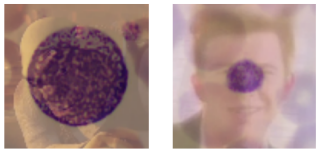

Figure 2: Incorrect images in the dataset

3 Method

After a review of the literature [11, 10, 13, 1, 12], we established a pipeline to design an optimal model for the problem. This process involved iterative testing and improvements, in order to find the optimal components and parameters.

3.1 Transfer Learning Model

Following a series of experiments, we decided to adopt transfer learning as the main strategy for our classification model. We employed `DenseNet121` [7] model from `tensorflow.keras` as our pre-trained model, with the weights initialized using ImageNet. `DenseNet121` is composed of several layers, including: convolutional layers, activation layers (ReLU), and batch normalization layers, for stabilizing the training process. Next, we applied a dropout layer to reduce overfitting, and a Dense layer with softmax activation for the final classification.

3.2 Data Augmentation

To address the limited variability in the training set, we employed data augmentation to increase the model's generalization capability, identifying the most suitable transformations. To make classification independent of specific cells positions and sizes, we applied transformations such as translation, zoom, and rotation. To modify the color palette, we introduced brightness, contrast, saturation and hue adjustments, among others. To simulate realistic noise, we introduced additional transformations, including sharpness adjustment and Gaussian blur. Moreover, different classes were "mixed" using `MixUp` [18], and we applied `RandAugment`[3] successfully.

To ensure dataset differentiation without compromising sample quality, we applied each transformation to a fraction of the data, gradually expanding it iteratively. This approach prevented excessive transformations on individual samples and allowed us to increase both the training test size and the accuracy of the model in realistic scenarios. Additionally, both transfer learning and the first four fine tuning stages used a dataset enriched with all these transformations, while the final fine tuning was performed using a dataset completely augmented with `RandAugment`. This allowed us to train the model on variations that are challenging to generate manually, enhancing the overall accuracy.

3.3 Training Parameters

After evaluating various optimizers, we selected `Lion` [2] for transfer learning and `Nadam` [4] for fine-tuning. We managed the learning rate with `ReduceLROnPlateau` [9], enabling gradual adjustments as the model improved. To prevent overfitting, we used `EarlyStopping` [8].

3.4 Fine Tuning

After transfer learning, where the `DenseNet` layers are frozen to train the model's FC layers, fine-tuning is implemented in 4+1 steps with progressive unfreezing of layers (last one using the all-`RandAugment` dataset). A layer with real-time data augmentations, including flips, rotations, translations, and brightness adjustments, is applied throughout the whole training process.

3.5 Test Time Augmentation

As a final technique, we employed test time augmentation (TTA) to enhance our model's performance. TTA involves making multiple predictions on slightly modified versions of the input image during inference. By averaging these predictions, we can achieve a more robust and accurate final prediction.

4 Experiments

The classification model described in Section 3 is the result of several trials. In this section, we highlight some of the experiments we conducted, ranging from the selection of the initial model to the usage of preprocessing layers.

4.1 Transfer Learning Models

Table 1 lists the pre-trained models we explored as starting models for transfer learning. In addition to those, we performed some experiments with a simple `CustomModel` composed of 4 convolutional layers and 3 dense layers.

Model Name	#Weights	Storage (MB)
`DenseNet121` [7]	7,037,504	26.85
`InceptionResNetV2` [15]	54,336,736	207.28
`ResNet50` [5]	23,587,712	89.98
`MobileNet` [6]	3,228,864	12.32
`NASNet` [19]	84,916,818	323.93
`EfficientNetB0` [16]	4,049,571	15.45

Table 1: Transfer learning models

4.2 Data Preprocessing

In our final training strategy, we decided to employ data augmentation in order to account for the diversity of the test samples. Following a different approach, we tried to apply some preprocessing layers in order to remove redundant information, both at training and inference time. We conducted experiments with preprocessing filters such as **gray scale** to remove color dependency, **background elimination** to isolate the cells and **Sobel filter** for edge detection.

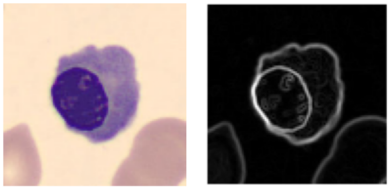

Figure 3: Sobel filter

What we noticed during our evaluations is that this kind of preprocessing boosts the performances of small, simple models (such as our `CustomModel`), without impacting significantly the accuracy of more complex models.

5 Results

Our model is lightweight compared to current state-of-the-art structures (7,037,504 weights with a storage requirement of 26.85 MB), yet it is able to achieve 92% accuracy on the official test set. Table 2 reports the accuracy of the models we submitted to the coding platform.

We highlight that we did not test the whole methodology described in Section 3 for each model, as all the experiments we performed were driven by incremental improvements of the training pipeline along with more detailed research for the best pre-trained model to be employed.

Model Name	Accuracy
`DenseNet121` [7]	**92%**
`InceptionResNetV2` [15]	85%
`NASNet` [19]	71%
`ResNet50` [5]	63%
`CustomModel`	57%

Table 2: Models accuracies on the official test set

6 Conclusions

The most influential step in our training methodology was **data augmentation**, boosting the model's generalization capability. We noticed that the application of the correct augmentation layers yielded important performance boosts, allowing us to reach 92% accuracy with a relatively small model. By contrast, our attempts to increase accuracy through model complexity did not produce the same improvements. Finally, test time augmentation proved to be a reasonable choice for all the models we employed, consistently increasing the final accuracy by 1 to 5%.

6.1 Individual contributions

Alessandro Annechini performed several experiments on data preprocessing and augmentations, and introduced test time augmentation to increase the accuracy of the implemented classifiers.
Riccardo Fiorentini selected the `DenseNet121` model and perfected the transfer learning procedure by testing optimizer parameters, training callbacks, and the number of layers to freeze during fine-tuning.
Lorenzo Vignoli performed the majority of the experiments on large transfer learning models and designed the augmentation layers adopted in the final version of the model.

References

[1] M. Ayyathurai, J. Ruth, K. V R, and U. Ranganathan. Automatic classification of white blood cells using deep features based convolutional neural network. *Multimedia Tools and Applications*, 2022. Accessed November 2024.

[2] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le. Symbolic discovery of optimization algorithms, 2023.

[3] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719, 2019.

[4] T. Dozat. Incorporating nesterov momentum into adam, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[7] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[8] Keras. Earlystopping, 2024. `https://keras.io/api/callbacks/early_stopping`.

[9] Keras. Reducelronplateau, 2024. `https://keras.io/api/callbacks/reduce_lr_on_plateau`.

[10] V. Kuhn et al. Red blood cell function and dysfunction: Redox regulation, nitric oxide metabolism, anemia. *Antioxidants & Redox Signaling*, 26(12):718–742, 2017. Accessed November 2024.

[11] S. Manik, L. M. Saini, and N. Vadera. Counting and classification of white blood cells using artificial neural networks (ann). *IEEE Xplore*, 2017. Accessed November 2024.

[12] J. Mitra, K. Vijayran, K. Verma, and A. Goel. Blood cell classification using neural network models. *IEEE Xplore*, 2023. Accessed November 2024.

[13] J. Pernow, A. Mahdi, J. Yang, and Z. Zhou. Red blood cell dysfunction: A new player in cardiovascular disease. *Cardiovascular Research*, 115(11):1596–1605, 2019. Accessed November 2024.

[14] U. Poggemann, T. Kern, and A. Koenen. Microscopic examination of blood, September 2018. Carl Zeiss Microscopy GmbH.

[15] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[16] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[17] A. W. Tessema, M. A. Mohammed, G. L. Simegn, and T. C. Kwa. Quantitative analysis of blood cells from microscopic images using convolutional neural network. *Medical & Biological Engineering & Computing*, 59(1):143–152, 2021.

[18] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

[19] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.