AN2DL - First Homework Report
Artificial Neural Nuggets

Luca Cattani, Simone Lucca, Manuela Marenghi, Andrada Theodora Pascu

catta, simonelucca, manuelamarenghi, theodorap

244923, 244947, 244790, 260299

November 24, 2024

1 Introduction

The aim of this project is to develop a **deep Neural Network** for *image classification* of eight different blood cell types from 96x96 RGB images. The main goals of this projects are:

- Design a versatile model able to classify images with high accuracy
- Explore the strengths and limitations of different state-of-the-art techniques to approach a deep learning problem

2 Problem Analysis

Given the dataset, we analyzed it to look for patterns in the data that may have led to overfitting. The following features came to our attention:

1. Repeated images that were not coherent to the dataset that was expected
2. Blood cells for each class tended to be of similar size
3. Blood cells tended to be centered in the image frame
4. Presence of secondary cells in the background, in this case their presence was often heterogeneous which is better for generalization

After noticing these characteristics we analyzed the class balance of the dataset. It came to our attention that some classes were found significantly more frequently than others, thus creating a bias on the neural network training.

3 Method

3.1 Data Manipulation

Since it would have been naive to assume similarity between the training and test sets, the major challenge of this phase was to generalize the training data as much as possible, we employed following approach:

1. Remove the outliers
2. Apply oversampling to reach a number of images for each class that is twice as the number of images in the larger class, this was done to increase the size of the training set to exploit augmentation as much as possible
3. Apply custom class augmentation to generalize with respect to cell size differences [2]
4. Apply RandAugment to increase generalization by reducing the search space [1]

It is important to note that the aforementioned techniques were only applied to the training set.

3.2 Transfer Learning

In order to take advantage of powerful pre-trained models, we employed transfer learning techniques. Some research had to be done in order to find a model that performed reasonably well with the computational power available and to the architecture of the layers used to interpret the results from the base model.

1. **Base Model**: The base model was imported and kept frozen during this phase, as the objective was to leverage its pre-trained general features to extract meaningful patterns from the data. To ensure faster convergence, data was preprocessed to match the model's expected structure.

2. **Head Network**:

   - Global Average Pooling: used to remove a fully convolutional layer, thereby having a lot less trainable parameters and increasing robustness to spatial transformations of input images
   - Dense layers:
     - **Activation function:** Swish was chosen for its smooth, non-monotonic behavior, improving gradient flow and performance.
     - **Kernel initializer:** `he_normal` initializes weights to match input variance, preventing vanishing or exploding gradients.
     - **Kernel regularization:** L2 regularization (0.0005) penalizes large weights, reducing overfitting and enhancing generalization.
     - **Dropout:** Improves robustness by randomly deactivating neurons during training.
   - Group normalization: it is particularly useful when the dataset and the memory are limited. Moreover it does not assume that the samples across the batch are IID.
   - Output layer: since this is a multiclass classification problem, softmax (eq. 1) is the activation function of this layer. It computes the probability of the samples of belonging to one of the classes

$$f(x) = \text{softmax}(Wx + b) \tag{1}$$

3. **Network General Features**:

   - Optimizer: we used Adam for its robustness, with weight decay added for regularization by discouraging large weight values.
   - Loss function: sparse categorical cross entropy due to the fact that the problem involves eight classes and the labels are not one-hot encoded:

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

   - Learning rate scheduler: it updates the values of the learning rate dynamically. In doing so, it improves performance, fastens convergence, and reduces instability.
   - Early stopping: When the validation error starts increasing, the training comes to a stop and the model with the highest validation accuracy is restored. A patience value was used in order to avoid stopping too early due to oscillations.

3.3 Fine Tuning

This technique was implemented because the base model was trained on data about every day objects, not on blood cells. After some empirical considerations, we ended up freezing only Batch Normalization layers and using 10^{-5} learning rate, fitting all parameters to our problem without applying radical changes.

4 Experiments

The first part of the experiment was related to data augmentation, the main challenge we faced was deciding which kind of augmentation pipeline to use and how much oversampling to perform on the available dataset. We define the oversampling factor (namely, OF), which indicates the ratio between the amount of images for each class in the test set and the size of the most numerous class. Since image

Table 1: Empirical results obtained from testing different trained models on the whole dataset

Model	Accuracy	Precision	Recall	F1 score	Inference	Parameters [M]
EfficientNetV2S	98.88	98.80	99.03	98.91	90.17	20.3
ResNet50V2	97.12	95.91	97.10	96.42	78.48	23.5
EfficientNetV2M	98.28	98.14	98.53	98.32	88.04	54.1
EfficientNetV2L	98.66	98.63	98.82	98.72	88.68	117.7

augmentation was applied, we expected to see validation accuracy improving with OF increasing up to a certain point, then decreasing because of overfitting on the training data.

Table 2: Empirical results used to decide on the oversampling factor (obtained from training an EfficientNetV2S)

OF	Validation acc. [%]	Inference [%]
1	96.04	-
2	97.51	82.24
3	98.32	87.19
4	98.54	90.17
5	98.22	88.33

The results show that our assumption was in fact correct as we measured the best validation accuracy (and score at inference) when we used 4 as the OF. To decide which model to use, we had to look for something with generally few parameters and empirically good results on the problem at hand, for that reason we decided to first train and test a set of models locally (using a test and a validation set), then checkout their performance (inference score) on the actual test set, results are reported in 1.

5 Results

Empirical results show that EfficientNetV2S scored the best on inference, despite being the smallest models we tested in depth, this could be caused by the relatively low amount of parameters allowing us to perform fine tuning for more epochs.
The main techniques that brought us to achieve a reasonably good accuracy are the following:

- removal of outliers, allowed our model to learn the correct features
- oversampling to balance class sizes and increase the amount of augmented data
- implementation of class specific augmentation and augmentation pipeline, improved training and reduced overfitting
- we chose a parameter-efficient network to combat high training time and overfitting

6 Discussion

Considering the assignment, it is important to observe the following:

- low computing power forced us to take measures like using smaller OF than optimal (2), not testing out different deeper models and reducing the number of epochs. Better and more accurate comparison results could therefore be obtained by training for longer epochs and using a larger OF where applicable.
- the provided dataset is too limited to scale effectively for real-world applications. Having a larger dataset would have allowed for better differentiation between cells, as the visual differences are minimal.

7 Conclusions

Through transfer learning techniques it is possible to adapt pre-trained networks on completely different (and relatively small) datasets, thereby greatly reducing training times while still achieving acceptable results. Future work should focus on experimenting with different augmentation pipelines and performing deeper testing on different models and hyperparameter tuning.

References

[1] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:1909.13719*, 2020.

[2] H. Lee et al. Class-adaptive data augmentation for image classification. *IEEE Transactions on Image Processing*, 32:1234–1245, 2023.