POLITECNICO
MILANO 1863

AN2DL - First Homework Report
acque del friuli

Filippo Riva, Christian Rossi, Kirolos Sharoubim, Antonio Sulfaro

filipporiva, christianrossi, carlosharu, antoniosulfaro

260296, 245631, 250813, 259954

November 24, 2024

1 Introduction

The primary goal of this project is to address an *eight-class classification problem* using 96×96 RGB images of blood cells, leveraging a labeled dataset containing 13,756 samples.

To achieve the highest possible classification **accuracy**, we systematically experimented with various techniques and neural network architectures discussed during the lectures, comparing the performance of each approach.

As a secondary objective, we focused on enhancing the classifier's **generalization** ability, aiming to improve its accuracy on Codabench evaluation data.

2 Problem Analysis

The main issues we faced during this homework where:

1. The presence of **outliers**, which were maliciously inserted into the dataset.

2. A considerable **class imbalance**, which might have led to biased models.

3. A huge **performance mismatch** between the local and real test sets, implying that our data didn't properly represent the distribution of the test.

3 Method

To streamline the implementation process, we organized the work into a series of Jupyter notebooks, each dedicated to a specific set of tasks:

- *Dataset Analysis*: used to explore the dataset, identify and remove outliers, and address the class imbalance issues.

- *Model Development and Training*: a structured environment for building neural networks and performing training experiments.

- *Deployment*: necessary for generating the submission files for Codabench, and additional prediction time logic.

Additionally, we grouped auxiliary functions into a `utils.py` script to promote code reuse and consistency.

To allow parallel development, each team member created a personalized copy of a notebook to implement their unique approach. This strategy even enabled us to easily track intermediate results, helping identify the techniques that were best suited for the problem.
Furthermore, we adopted an **incremental approach**, starting with simple models and gradually progressing to more advanced architectures. This step-by-step refinement allowed us to build on our insights and systematically improve performance.

3.1 Problem Resolution

The challenges outlined in the second section were addressed as follows:

1. Outlier removal: all images identical to the two outliers identified in the dataset (blood cells with an altered background) were removed. We opted not to use a k-Nearest Neighbor approach since the outliers were identical and easily identifiable.

2. Class balancing, through several strategies:

   (a) Under-sampling and over-sampling.

   (b) Under-sampling with MixUp augmentation interpolating images within the same class.

   (c) Class-weighted categorical crossentropy during training.

3. Generalization improvement by augmentation techniques. After extensive testing we ended up using:

   (a) CutMix: combining images by overlaying random sections.

   (b) MixUp: interpolating pairs of images to generate new samples.

   (c) GridMask: randomly occluding portions of images with grid-like patterns.

   (d) RandAugment: applying a random selection of basic transformations to images.

   These techniques were randomly combined to create a high-variance training set, ensuring diversity while avoiding overly distorted images.

4 Experiments

Below, we present only the models and configurations that provided meaningful results:

1. Multi Layer Perceptron (MLP):

   - *Basic MLP* : Used as a comparison baseline.
   - *MLP + Augmentation* : performed worse than the basic MLP, as MLPs lack the ability to handle spatial invariance effectively.

2. Convolutional Neural Network (CNN):

   - *Basic CNN* : delivered improved performance compared to MLPs, achieving a 96.7% local test set accuracy, highlighting the difference between training and real test sets.
   - *CNN + Augmentation*: integrated *Global Average Pooling* (GAP) layers, simple *data augmentation*, *class-weighted loss functions*, and *batch normalization* to enhance generalization. However, the improvement on accuracy was modest.

3. EfficientNetB0: we have decided to start from this architecture since it is lightweight, but has also a good performance on the imagenet dataset. The main configurations are:

   - *Basic EfficientNetB0*: the first transfer learning solution using custom top-layers. Exploiting GAP layers, augmentation, and class-weighted loss function. This significantly boosted performance even without major changes on augmentation techniques, highlighting the efficiency of transfer learning.
   - *EfficientNetB0 improved*: enhanced with *ELU* activation functions, the *Lion* optimizer, *mixed precision* training, and batch normalization. Showed minor training time improvements during fine-tuning and slightly better performance on test data.
   - *EfficientNetB0 with custom layer*: introduced a custom network layer for image thresholding to remove backgrounds during both training and prediction. It did not significantly improve performance, neither locally nor on the Codabench test set.

4. EfficientNetV2B3:

   - *EfficientNetV2B3*: a more complex model paired with *CutUp*, *Mixup* and *GridMask* augmentations. This configuration was more robust to noisy and augmented data, maintaining solid performance on the original dataset.

- *EfficientNetV2B3 improved*: focusing more on augmented approaches and hyperparameter tuning. All augmentations were applied as preprocessing before training showing a considerable improvement in training speed. This setup delivered higher validation accuracy and better generalization to augmented test data.

5. *EfficientNetV2L*: this larger architecture aimed to maximize performance by carefully adjusting the augmentation techniques we developed on EfficientNetV2B3 and applying extensive fine tuning. It delivered the best results overall. This approach had the best performance even locally, scoring a 98.6% accuracy, showing how augmentation can help even on really standard data.

The following table summarizes the performance of the models on Codabench:

Model	Accuracy
Basic - Augmented MLP	19% - 16%
Basic - Augmented CNN	22% - 26%
EfficientNetB0	63%
EfficientNetB0 improved	64%
EfficientNetB0 w/ custom layer	63%
Basic - Improved EfficientNetV2B3	68% - 84%
EfficientNetV2L	**86%**

Throughout training we extensively used *early stopping* and *dropout* to avoid overfitting, and ensuring good performance on test data.
In the final stages, we considered applying test-time augmentation to further boost performance on the test dataset, even if we were not able to test it on Codabench.

5 Results

The reason why EfficientNetV2L achieved such a good performance compared to previous models was due to mainly two reasons:

- Aggressive usage of augmentation techniques, expanding the size of the dataset to 3 times it's original size, deeply exploiting the feature extractor complexity.
- Deep Fine Tuning, since the problem was highly specialized and way simpler than the original image-net problem, on which our weights were being initialized.

Besides that, the network's architecture followed a standard transfer learning approach, substituting pre-trained classification layers with a *Global Average Pooling* (GAP) layer, followed by dense layers with $ReLU$ activations.

The optimizer we tried were *Adam* and *Lion*, although they didn't much differ, both in performance and training speed.

6 Final Considerations

One of the most surprising findings was the limited impact of our initial augmentation techniques due to their lack of complexity. This highlighted (contrary to our initial beliefs) the effectiveness of using advanced augmentations, even for simpler problems.

This epiphany made us realize that the performance of even simpler models, could have been highly improved by knowledgeable use of the augmentations techniques on which we focused too little on the initial part of this homework. We are sure, in fact, that by extensive Fine Tuning and a lot of experimentation on augmentation approaches the performance of such a big model like EfficientNetV2L can be matched and surpassed by way simpler models as proven by the improved EfficientNetV2B3, reducing needed training time and prediction time performance.

7 Conclusions

Each team member contributed equally to modifications and ideas that cumulatively led to the development of our final model.
Future improvements could involve:

- Testing additional augmentation techniques to further increase robustness.
- Implementing an ensemble of two or three high-performing models to combine their strengths and achieve better overall accuracy.
- Trying smaller models with advanced augmentation techniques, to increase deployment efficiency both in size and speed.

References

1. M. Matteucci, G. Boracchi. AN2DL slides. 2024.
2. C. Chola, et al. BCNet: A Deep Learning Computer-Aided Diagnosis Framework for Human Peripheral Blood Cell Identification. 2022.
3. M. Tan, Q. V. Le. EfficientNetV2: Smaller Models and Faster Training. 2021.
4. E. D. Cubuk, B. Zoph, J. Shlens, Q. V. Le, RandAugment: Practical automated data augmentation with a reduced search space. 2019.