AN2DL - First Homework Report
The Backpropagators

Arianna Procaccio, Francesco Buccoliero, Kai-Xi Matteo Chen, Luca Capoferri

ariii, frbuccoliero, kaiximatteoc, luke01

246843, 245498, 245523, 259617

November 24, 2024

This is a report of the first homework for the course **Artificial Neural Network and Deep Learning** at Politecnico di Milano.[1]

The focus of this homework is to correctly assign to each *RGB image* of **blood cells** one of eight given labels, referring to the type of cell. The problem is commonly known as *Image Classification.*

1 Problem Analysis

The dataset provided counts 13759 **images** as tensors of size $(96, 96, 3)$ and as many labels, encoded as numbers from 0 to 7.

Following a first inspection it was noted that a total of 1800 ***"junk"* images** were maliciously added to the dataset. Even by ignoring the *superimposed* pattern, such images didn't match the assigned label, thus stopping any attempt to recover any information from them. We then assumed they could be discarded prior to any further work. See Fig.1 for a visual reference.

The dataset has a certain amount of **class unbalance**, with the smallest class being just over 1000 samples and the biggest just over 2500. Since the imbalance isn't extreme (e.g. credit card frauds or cancer detection) we didn't address it directly with techniques like *SMOTE*[1] but rather moved to losses like *FocalLoss*[7] that can help with that over training.

The metric to maximize as homework score, as declared in the rules, is the **accuracy** over a different test set, not given to us. The accuracy score is computed by a submission engine run over the internet. See Eq.1

$$\text{Accuracy} = \frac{\sum(y_{\text{true}} = y_{\text{pred}})}{\text{len}(y_{\text{true}})} \tag{1}$$

Given the specifics of the problem and the rules given, and following a first "manual" attempt, we decided to spend most of our work time working with *pre-trained models* available on Keras and applying **Transfer Learning** and **Fine Tuning** to fit them to our needs, rather than building our own model manually from scratch.

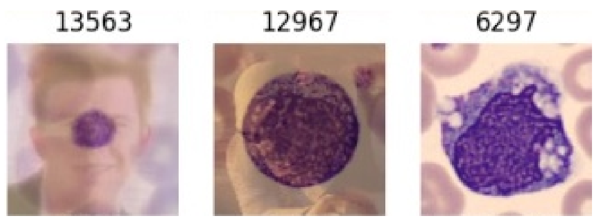

Figure 1: Two junk images and a valid one. The two are obtained by superimposition of images of singer Rick Astley and cartoon character Shrek.

[1]`http://chrome.ws.dei.polimi.it/index.php?title=Artificial_Neural_Networks_and_Deep_Learning`

2 Method

Our methodology followed the principle of **modularity**. Since the beginning we worked over flexibility and built our notebook so that anyone could easily swap *model*, *optimizer*, *augmentation* or any other relevant setting. With the same ease any new insightful idea (or suggestion from the *Logbook*) could be easily added as an option without disrupting others' workflow. This helped to easily **test many different configurations** and ultimately led to our best results.

2.1 Preprocessing and augmentation

We promptly found all indicies of *junk* images over the dataset and added them to a **blacklist** file to be used to discard them at any run of the notebook.

We then worked on building an **augmentation pipeline** that resulted being the key to achieving higher accuracy. In its final form it triplicates the dataset processing it trough **three parallel augmentation** paths and a final general augmentation:

- A third of the data was passed to the final augmentation stage without any modification
- Another third of the data was passed trough **RandAug**[3], then **CutMix**[12] and finally **MixUp**[13]
- The final third of the data was passed trough **AugMix**[4]

The **final augmentation** step before training consisted of a *Sequential* Keras Layer with classic geometric transformations such as *rotation* and *translation*.

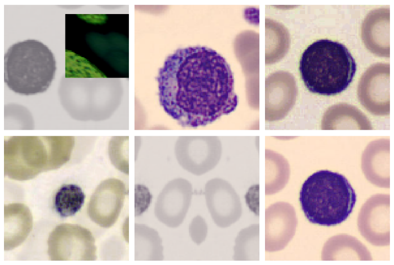

Figure 2: Six augmentation pipelines outputs.

2.2 The model

We tried different models over the first phase:

- The first attempt involved building a **CNN** from scratch, consisting of four *convolutional* blocks, *ReLU* activations and *Dropout* for regularization. Finally, a *Global Average Pooling* layer. The model has a total of over $400k$ parameters to train.
- We then moved to **VGG19**[9] and obtained some improvements, particularly when applying *finetuning* over the imagenet weights.
- The best model (we eventually used in the end for most of our tests) was the Large version of **EfficientNetV2L**[10] which, thanks to its more advanced and modern topology, yielded better results over both Transfer Learning and Fine Tuning, and faster fitting too.
- Then we also tried an **ensembling method** by taking the average prediction from multiple models (VGG19+EffNet + EffNet with a custom GAP only, *à la* [6]) to produce a more robust prediction. However, this method's accuracy was **only as good as the best model** considered, leading to its discard.

2.3 Optimizers, Losses

We employed different optimizers, including more exotic ones like **LION**[2] and **RANGER**[11] but despite the effort with parameter tuning the best results came with simpler optimizers like **AdamW**[8], that extends ordinary **Adam**[5] to weight decay, and **SGD** (Stochastic Gradient Descent).

Over losses: we spent most of our attempts using a classical **Categorical Crossentropy** loss, the standard for multi-class classification problems, see Eq.2. Then we had better results switching to **FocalLoss**[7] with **Label Smoothing**, to further help with overfitting. See Eq.3 for details. Note $p_{t,i}$ is the predicted probability for the true class of the i-th sample and the *hyperparameters*: α deals with class imbalance and γ is a "focusing" parameter.

$$\text{CCE} = -\frac{1}{N}\sum_{i=1}^{N}\log(p_{t,i}) \tag{2}$$

$$\text{FL} = -\frac{1}{N}\sum_{i=1}^{N}\alpha_{t,i}(1-p_{t,i})^{\gamma}\log(p_{t,i}) \tag{3}$$

Table 1: The accuracy results obtained with our tests. Only the best outcome is reported per each attempt. The augmentation methods used don't necessarly match, as we moved towards more complex solutions while also changing models. Best results are highlighted in **bold**.

Model	Train Acc.	Validation Acc.	Platform Test Acc.
Random predictor (expected)	-	0.125	0.125
Custom CNN (Adam)	0.89	0.93	0.24
VGG19	0.97	0.96	0.62
VGG19+Aug (SGD+WD)	0.97	0.98	0.61
EfficientNetV2	0.84	0.83	0.52
EfficientNetV2 (GAP only)	0.88	0.92	0.53
EfficientNetV2+Aug*	**0.98**	**0.99**	**0.90**

3 Experiments and results

We applied the above mentioned models, with and without augmentation, kept the best models and tried to improve by **tuning hyperparameters**. All the best results have been achieved with *AdamW*, unless specified. As shown in Table 1, the EfficientNetV2 model achieved the **highest accuracy**, but other observations can be made:

- The test accuracy on Codabench[2] is **consistently lower** than the accuracy obtained on a held out test set from our dataset. This suggests a **lack of generalization** of the network that nudges further work over the topology.

- When augmentation was applied the train accuracy required much **more epochs** and **aggressive** learning rate to match validation accuracy. This behaviour was expected but motivated some adjustments in our notebook.

4 Conclusions

In this homework we employed different techniques to address the classification problem of blood cell images. Through various experiments, we identified two key elements to achieve a high performance:

- **Impact of Data Augmentation:** Augmentation proved to be the most impactful technique for improving NN performance, especially in mitigating *overfitting* and enhancing *generalization.* Our augmentation pipeline, including *CutMix, AugMix*, and traditional augmentations, significantly **boosted model robustness and accuracy**.

- **Transfer Learning and Fine Tuning:** The use of pretrained models showed the advantages of leveraging publicly available weights when training data is limited, requiring fewer training epochs to converge with respect to training the network from scratch. A final **Fine Tuning** step was then the key to reaching even better performances.

While achieving satisfactory results, there are still some points that could've been explored:

- Despite being a popular approach, our attempts at **ensembling** did not outperform the best individual model, suggesting potential room for improvement. Due to the time constraints we decided to **prioritize local improvements** over the same known reliable models rather than exploring different model alternatives and their ensembles.

- While **focal loss** with label smoothing helped address *class imbalance*, exploring alternative imbalance handling techniques, such as data resampling or cost-sensitive learning, might yield additional benefits.

The work has been evenly divided among the group but in order to identify contributions: **Matteo** was the group leader and worked mostly on the building blocks of the notebook, **Arianna** and **Luca** spent most effort over augmentation and **Francesco** on hyperparameter tuning, testing and writing the report.

[2]Our submission platform. `https://www.codabench.org/`

References

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[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 automated data augmentation with a reduced search space, 2019.

[4] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty, 2020.

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[6] M. Lin, Q. Chen, and S. Yan. Network in network, 2014.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

[8] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[10] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training, 2021.

[11] L. Wright. Ranger - a synergistic optimizer. `https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer`, 2019.

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

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