AN2DL - First Homework Report
ANNamo

Silvia Carone, Mattia Fioravanti, Antonio Fraccastoro, Federico Patacca

silviacarone, mattiasante0512, antoner , federicopatacca

249510, 243149, 252908, 250869

December 20, 2024

1 Introduction

In this challenge, We were asked to design a *neural network* that could classify eight distinct types of blood cells. The provided dataset consisted in 13759 96x96-pixel size images in RGB color format. To achieve this, we proposed two different models: one was a convolutional neural network (CNN) built from scratch, while the other exploited the transfer learning technique.

2 Problem Analysis

Our preliminary analysis focused on the dataset. First of all plotting some images, we noticed some **"outliers"** unrelated to our classification problem, therefore we removed them.
Then we explored the real dataset, composed of images of blood cells belonging to 8 different classes. After splitting the dataset into training (80%), validation (10%) and test (10%) preserving the class distribution, we noticed that they were **unbalanced** with 1887 samples of the most represented to the 688 of the least.
We then tried to leverage **augmentation** to both rebalance and increase the samples in the classes, the augmentation first consisted of geometric ones like translation, rotation and zoom, intentionally kept low to preserve the unique characteristic of the small size of the Platelet class, but then we moved to more aggressive ones for better generalization exploiting **RandAugmentation** [2] and **AugMix** [3] in the final model.
Some attempts worthy of mention, not included in the final model, were: the removal of the 1% extreme images selected with PCA and the addition of white noise in the training set which is a technique we ultimately left only for the test set in order to asses the **robustness** of our models.

3 Method

Our approach was first to see if with a **custom CNN** we were able to tackle this classification problem. First, we tried to overfit the dataset to make sure our model was complex enough, then we proceeded to regularize it.
This model consisted of five **convolutional layers** followed by **Global Average Pooling** to reduce the number of parameters to train. Finally, we inserted a dense layer to increment the capability of the network to generalize. To avoid overfitting and to improve the data flow within the model we have tried playing with different regularization methods, finding the optimal results in a configuration with a batch normalization layer for each convolutional layer and a dropout layer for the last dense layer before the output one. Furthermore, we decided to use the Lion[1] optimizer since the stochasticity given

by the update rule, which uses the sign operator, acts as a form of regularization, which can further improve our model's robustness. We then moved to a modern approach resorting to pre-trained models for image classification. We applied **transfer learning** with mainly one category of pre-trained models: EfficientNet. The EfficientNet[4] family is composed of recent models, faster to train and also trained with augmentation as in our case. We first used the B0 and B4 models which performed quite well, but their performances weren't high enough that's why we tried the last ones, namely the **V2S** and V2M, with the latter being too computationally expensive for training with our means, while the first proved to be efficient and fast, as we adopted it for our final model. To better adapt the EfficientNetV2S model to our task we decided to add two **dense layers** to this architecture with 512 and 256 neurons respectively, placing in between them a batch normalization layer and a dropout (0.2) to the second one; to both layers we applied an L2 (1e-4) regularization. The use of early stopping during training concludes the list of regularization techniques used during the transfer learning. Furthermore, we exploited **fine tuning** on the model carefully tuning the layers to unfreeze, keeping for example the batch normalization layers frozen to prevent overfitting and preserve the latent representation of the pre-trained model. To conclude, L2 regularization (1e-4) has been applied to the unfrozen **Conv2D** layers because it showed to have better behavior concerning generalization. Regarding the optimizer, we opted for **AdamW** which proved to be stable and performing with an initial learning rate of 4e-4 for transfer learning and 1e-5 for fine-tuning. The learning rate has been tuned successively exploiting the *ReduceLROnPlateau* method, a learning rate **scheduler** implemented in Keras, that enabled us to dig into lower losses by reducing the learning rate by a factor 2 when there was no improvement for 5 consecutive epochs.

4 Experiments

As previously stated, we identified a proper augmentation pipeline for the train set as a crucial factor in increasing the accuracy of the classification problem. Subsequently, we present several experiments we conducted on our model to achieve better results:

- **Oversampling**
  Initially, we tried to rebalance the classes through oversampling of the minority classes, in order to be able to provide our model with a greater number of samples for training. We thought this approach was overly simplistic and thus we opted to generate new samples exclusively through augmentation.
- **White noise on train set**
  We attempted to apply white noise on the training set to facilitate the model to recognize images in case of shift in several pixels.
- **Augmentation by AugMix**
  We tried first to augment the training set by means of AugMix, an algorithm implemented in Keras, which guarantees greater robustness thanks to the weighted combination of transformations it applies. However, it was deemed to be overly complex for our model and thus we decided to leverage both RandAug and Augmix, carefully tuning the amount of augmentation.

5 Results

From the table showing the results we can see that **unexpectedly** the custom CNN achieve a discrete result on the hidden test set which is well above the simplest **baseline** which is random guessing: 12.5%. As we expected the **pre-trained** models achieve better results than these two, in particular is remarkable the **final** one with EfficientNetV2S fine-tuned which performs excellently under every metrics and on every test set as shown also from the confusion matrix of the plain one, to which we add a 84.78% over the dirt test set with white noise, proving to have reached an impressive level of generalization.

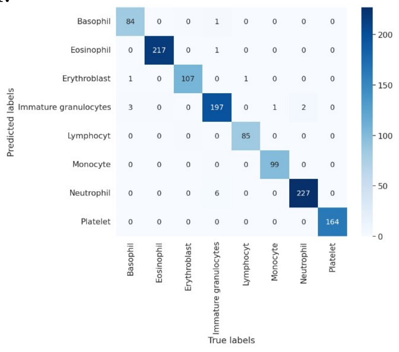

Figure 1: Confusion matrix of final model on plain test set

Table 1: Results achieved with different models.

Model	Training Accuracy	Validation Accuracy	Plain test accuracy	Hidden test set score
Custom CNN	98.49%	97.49%	96.99%	0.47
EfficientNet B0 (transfer-learning)	86.71%	86.26%	88.46%	-
EfficientNet B0 (fine-tuning)	93.61%	**9**4.06%	94.73%	0.49
EfficientNet B4 (transfer-learning)	84.93%	87.28%	87.45%	-
EfficientNet B4 (fine-tuning)	89.37%	90.71%	90.21%	0.54
EfficientNet V2S (transfer-learning)	90.51%	94.89%	94.23%	-
Final model	**98.30%**	**97.96%**	**98.66%**	**0.78**

6 Discussion

Our analysis of the results takes into account several metrics: we assess the accuracy of our final model on our **plain** test set, on the "dirt" test set with **white noise**, the hidden test set and the **confusion matrix**. The first 3 helped us to assess the **robustness** of our model while the last one gave us more insights about the performances on each class, in particular the most misclassified were often the immature granulocytes, besides our final model which was able to recognize basically all the classes indiscriminately.
This analysis helps us having a more complete overview of our model performance, especially in this case where we couldn't always see the real performances on the hidden test set which was our main limitation. This affected us at the beginning because there was an evident **discrepancy** between the local and the hidden test set which pushed us towards pre-trained models and heavy augmentation, which was our game changer. Also other important limitations concern our means, namely, the **RAM** of our computers which limited the number of augmented images we could generate as well as the other computational resources which prevented us from adopting larger and more powerful models as EfficientNetV2M and L or ConvNext.

7 Conclusions

Summary of Contributions

We addressed the classification challenge by leveraging pre-trained models, with EfficientNet V2S proving to be the most effective. Key techniques included fine-tuning with L2 regularization, dynamic learning rate scheduling, and advanced augmentation methods such as Augmix and RandAugmentation and white noise to improve model robustness and dataset balance.

Further Improvements

We aim to enhance the robustness of the model by further augmenting the dataset, ensuring more diverse samples are generated. One effective approach is leveraging the Keras **Dataset** data type, which is optimized for parallel computing, enabling more efficient data processing. In the thelastdance notebook, we explore a combination of RandAug and AugMix techniques to achieve a threefold increase in the dataset size, while maintaining balance using the same methodology outlined in the report. By utilizing the **Dataset** data type, we maximize parallelization, significantly improving computational efficiency. This allowed us to train with the EfficientNetV2M neural network. Unfortunately, we were unable to present the results within the time frame, but we are confident that the dedication and effort invested will be appreciated.

8 Contribution

Everyone in our group has contributed equally to the development of the model. It has been a process of trial and error and efficient implementation of numerous techniques.

- Silvia Carone: class rebalance, report
- Mattia Fioravanti: final notebook, report
- Antonio Fraccastoro: augmentation, custom-CNN, thelastdance, report
- Federico Patacca: implemented pre-trained models, report

References

[1] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le. Symbolic discovery of optimization algorithms. 36:49205–49233, 2023.

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

[3] 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.

[4] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.