AN2DL - First Homework Report
Alpaca

Ernesto Natuzzi, Flavia Nicotri, Luca Pagano, Giuseppe Vitello

ernesto, flanico, lp1807, peppisparrow

251284, 251671, 249359, 251129

November 24, 2024

***Abstract* - This project aims to classify 96x96 RGB images of blood cells into eight categories, each representing a specific cell state. To overcome the challenge of a small dataset, several techniques were used. A key challenge was training a model resilient to heavily transformed images generated through multiple layers of augmentation, leading to differing accuracy results on the unknown online test set.**

1 Introduction

The project, developed under the first homework provided by the *"Artificial Neural Networks and Deep Learning"* course offered by Politecnico di Milano, aims to build a model to **classify** correctly the type of blood cells under eight classes. The provided dataset is composed of 13759 RGB images of size 96×96 pixels, labeled as in Figure 1.

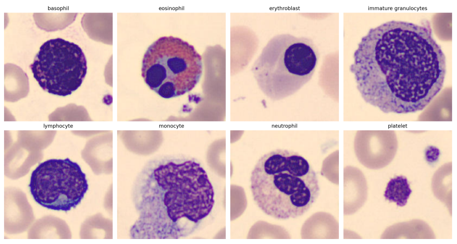

Figure 1: Cells classes

2 Data Analysis

The first step involved identifying and removing **outliers** and **duplicates**. After manually reviewing the images, we identified two main outliers that were repeated in the dataset. We then developed a Python script to remove the 1800 repeated images of these outliers, followed by the elimination of 8 additional duplicates from the cleaned dataset. This process resulted in a final dataset of 11,951 images, with the class distribution shown in Fig. 2.

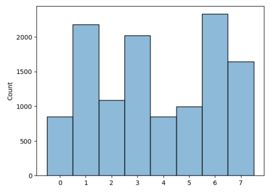

Figure 2: Distribution of each class

Our main focus was removing less obvious outliers that could hinder the model's generalization. To do this, we used a pre-trained MobileNetV3Large [1] with ImageNet [2] weights, excluding the dense layers, and utilized the feature extraction part as an image search engine. We extracted features for each class, calculated the centroid, and measured the squared distance of each image's features to the centroid. By setting an outlier threshold based on standard deviation, we flagged images exceeding this threshold as outliers.

However, this approach had little effect. The initial model, a simple custom CNN, correctly recognized these images. The discrepancy was due to the need for a more robust model, not a "dirty" dataset.

3 Experiments

In this section, we will discuss in chronological order the various experiments that allowed us to improve our model and how we achieved this.

3.1 Approach

Our efforts focused on finding the optimal combination of layers and hyperparameters to generate **augmented images** that were transformed but still recognizable. Initially, we used a simple custom **CNN**, but small networks pre-trained on ImageNet proved faster and more accurate. We progressively increased the complexity of augmentations until the model plateaued, then moved to a more complex architecture. Each iteration used the previous network as a baseline to evaluate if augmentations were improving performance, refining both augmentations and architecture for optimal results.

3.2 Augmentation

3.2.1 Initial Augmentation

The pipeline initially included basic transformations: translation, rotation, zoom, and flip. After several adjustments, we switched to using **RandAugment** [3] via **KerasCV**, experimenting with the number and intensity of augmentations per image. We used a **MobileNetV3Large** with ImageNet weights, establishing a new baseline. In our local non-augmented test, we achieved **93.98% accuracy**, while the online test during the development phase scored **0.62**.

3.2.2 Further Improvements

In the training we noticed a consistent **gap** between training and validation, as in Fig. 3, for the training of the above model, both in terms of accuracy and loss. Although it is not standard to augment the validation dataset, we chose to do so in subsequent tests since it no longer represented the real data distribution. This approach aligned training and validation results, improving the effectiveness of callbacks like EarlyStopping and ReduceLROnPlateau. We also began using **EfficientNetV2S** [4], which, based on its ImageNet accuracy and parameter efficiency [5], was ideal for our problem and resources. We added three dense layers (128, 64, 8 neurons) with dropouts, ending with a **softmax** activation.

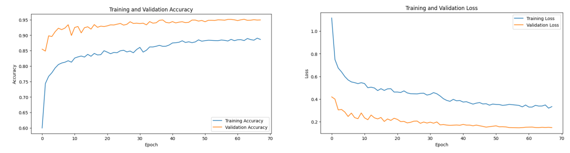

Figure 3: Accuracy and Loss of 0.62 model

3.2.3 Final Augmentation

With promising results, the pipeline was expanded to be more complete while. The final pipeline consists of:

- **Simple augmentations**: flip, rotation, and zoom;
- **AugMix [6] or RandAugment**: since these are invasive transformations, only one is applied per image;
- **GridMask [7] or RandomCutout**: both transformations mask parts of the image;
- **Color transformations**: ChannelShuffle, RandomChannelShift, or RandomHue.

By optimizing the augmentation pipeline and tuning hyperparameters, we significantly improved the network's generalization. In our local test, augmented for better comparison with the Codabench results, we achieved an accuracy of **84.48%**, and in the online test, an accuracy of **0.83**.

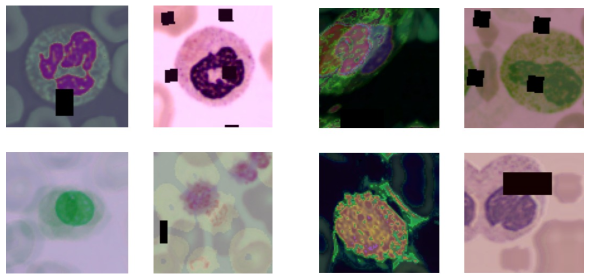
Figure 4: From left to right light to heavy augmentation

3.3 Training approach

During the research phase, we tested various models and **transfer learning** approaches. We found that with heavily augmented datasets, **fine-tuning** more than 50% of the convolutional layers improved performance and training both convolutional and dense layers together led to better results. From a baseline of 0.83, we achieved an accuracy of **0.91** in the final test. Unfreezing additional layers **worsened** performance, likely due to the pre-trained layers effectively extracting low-level features, while deeper layers struggled with the augmented dataset.

3.4 Segmentation

Uncertain about the optimal level of dataset augmentation, we shifted our approach to **preprocess** the data, enhancing its interpretability and reducing the load on the convolutional layers. We employed a **segmentation** technique to isolate the foreground from the background, using RGB values and spatial coordinates for pixel balancing. The segmentation was performed with predefined masks and the **K-Nearest Neighbors** classifier. While effective on non-augmented images, this method did not improve accuracy with augmented data.

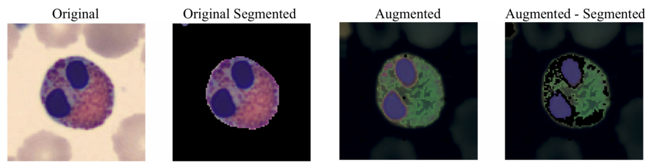

Figure 5: Segmented and augmented cell images

3.5 SMOTE

SMOTE (Synthetic Minority Over-sampling TEchnique) [8] is a preprocessing technique used to address class imbalance in a dataset. The algorithm works by calculating the difference between a sample and its nearest neighbor, multiplying this difference by a random number between 0 and 1, and adding it to the sample to create a new synthetic example in feature space. This process continues for the next nearest neighbors, up to a user-defined number. We employed this technique to address the lower accuracy in underrepresented classes (Fig. 2), such as classes 2, 4, and 5. Using the same EfficientNetV2s model, accuracy for these classes improved slightly, but overall accuracy declined. Therefore, we chose to use the original dataset, as it better reflects real-world images.

3.6 Loss Function and Optimizer

We tested various loss functions and found **Categorical Cross-Entropy** to be the most effective. Although incorporating class weights and using **Categorical Focal Cross-Entropy** showed similar results, they faced similar issues as SMOTE. For optimization, we started with Adam but switched to **AdamW** [9], which improved training efficiency. While AdamW is a common choice, we also explored Lion [10], which had excessive oscillations, and Ranger, which converged quickly but to a suboptimal minimum.

4 Results

Tab. 1 summarizes the experimental results. *Augmented Accuracy* reflects performance after applying data augmentation, while *Development Accuracy* and *Final Accuracy* represent development and final online test outcomes. The baseline model was fine-tuned with 50 convolutional layers, while all other models utilized 80% of the layers. The highest accuracy was achieved with an ensemble of two EfficientNetV2S models: one trained with standard methods and the other using AugMix with a slightly higher rate, demonstrating superior performance across all metrics.

Model/Configuration	Augmented Accuracy	Development Accuracy	Final Accuracy
MobileNetV3	60.04%	0.62	-
EfficientNetV2S baseline	84.48%	0.83	-
EfficientNetV2S (SMOTE)	93.50%	0.91	-
EfficientNetV2S (std. augment)	96.35%	0.92	0.91
EfficientNetV2S (Augmix)	95.63%	0.91	0.91
EfficientNetV2M	97.75%	0.89	-
Ensemble	96.65%	**0.93**	**0.92**

Table 1: Model Performance Comparison

5 Conclusion

A key challenge in this project was managing the fine-tuning process on heavily augmented datasets. It was difficult to identify the optimal threshold where models transitioned from improving generalization to causing overfitting. This issue was compounded by the lack of a dataset with the same distribution, requiring us to infer the transformations applied to the online data. Future work could focus on further optimizing augmentation strategies, which may facilitate the use of larger architectures, such as ConvNeXt [11], for improved performance.

References

[1] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[3] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719, 2019.

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

[5] Keras applications documentation: Comparison on imagenet between included keras models.

[6] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty, 2020.

[7] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *CoRR*, abs/2001.04086, 2020.

[8] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.

[9] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[10] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023.

[11] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.