AN2DL - First Homework Report
FeedForward Force

Aman Saini, Marine Peuzet, Carlos Ruiz Aguirre, Ariadna García Lorente

Amannnammm, mapeuzet7, irondog421, ariadnagarcia

276667, 276769, 276661, 276831

November 24, 2024

1 Introduction

This project, part of the Artificial Neural Networks and Deep Learning course, applies deep learning techniques to classify 96x96 RGB images of blood cells into eight distinct classes: basophils, eosinophils, erythroblasts, immature granulocytes, lymphocytes, monocytes, neutrophils, and platelets. These cell types are critical for immune function, inflammation regulation, and infection defense. Accurate classification aids in diagnosing conditions like infections, blood disorders, and certain cancers.

The task involves **multi-class classification**, requiring a robust model to generalize effectively from diverse training data. Using **pre-trained models** and **image processing techniques**, our goal is to optimize model architecture for high-accuracy classification.

2 Problem Analysis

The initial dataset *training_set.npz* contained 13,759 96x96 RGB images, divided into eight classes and distributed as shown in the Figure 1

After inspecting the given dataset, we identified some easily distinguishable **out-of-distribution** samples. We categorized these into **two types of anomalous samples**, images with a Shreck or Rickroll on the background:

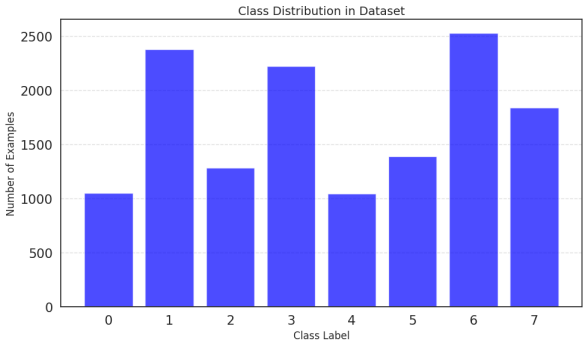

Figure 1: Class distribution

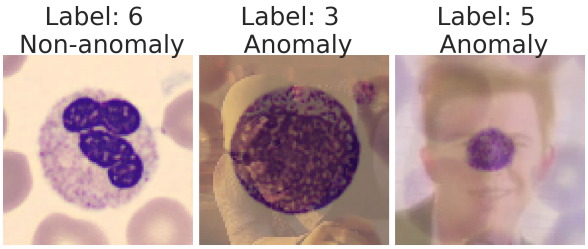

Figure 2: Normal and anomalous samples

The main challenges in this task involve addressing the **unbalanced class** distribution and effectively **managing anomalies** present in the dataset.

Moreover we assume that **the model will handle distorted images if they are included in the training** through data augmentation

3 Preprocessing Data

Handling and preparing the data to feed the model requires several preprocessing steps:

- **Removing anomalies.**

All abnormal images belong to the previously identified Shrek or Rickroll categories. To eliminate these anomalies, the perceptual hash (pHash) of the two samples was calculated, and all images with a matching hash were removed. This process successfully filtered out **1,800 abnormal samples**..

- **Data augmentation.**

For training a robust model and improving generalization, we have applied a combination of augmentation techniques, **AugMix** [4] and **RandAugment** [6].

From the original clean dataset (11,959 samples) we have generated low (s=0.1, m=0.1), medium (s=0.5, m=0.5) and hard (s=0.9, m=0.9) augmented datasets. s indicates severity (AugMix) and m represents magnitude (RandAugment). Each augmented dataset contains the same number of samples as the original, resulting in a total of 47,836 images in the preprocessed dataset.

4 Method

4.1 Feeding the model

- **Normalization:**

Pixel values are scaled using TensorFlow's `EfficientNet`[5] preprocessing function:

$$\tilde{x}_{ij} = \frac{x_{ij}}{255}$$

where x_{ij} represents the pixel value at location (i, j), and $\tilde{x}_{ij}$ is the normalized value.

- **Label Encoding:**

Labels are one-hot encoded, transforming class indices into a binary matrix format:[3]

$$y_k = \begin{cases} 1 & \text{if the sample belongs to class } k \\ 0 & \text{otherwise} \end{cases}$$

- **Train-Test Split:**

The dataset is divided into training (70%) and testing (30%) subsets to evaluate model generalization.

- **Class Weight Computation:**

To address the issue of unbalanced class samples, class weights are computed as:

$$w_k = \frac{N}{n_k \cdot C}$$

where N is the total number of samples, n_k is the number of samples in class k, and C is the total number of classes.
These weights are passed to the loss function during training to penalize misclassifications of underrepresented classes more heavily.

4.2 Model Architecture

The model leverages **EfficientNetV2**[5] with pretrained weights from **ImageNet** for **transfer learning**. An **input layer** defines the dataset shape, and **data augmentation** includes **horizontal flipping**, **random rotation (0.1)**, **zoom(0.1)**, **contrast adjustment(0.1)**, and **Gaussian noise(0.05)** to improve robustness.

For **fine-tuning**, 50% of the layers are **frozen** to retain features, while the rest are **unfrozen** for adaptation. The **output layer** matches the number of dataset classes.

4.3 Training Configuration

- **Loss Function:**

The categorical cross-entropy loss function, weighted by class, is used for multi-class classification:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} w_j \cdot y_{ij} \log(\hat{y}_{ij})$$

where w_j represents the weight for class j, y_{ij} is the true label, and $\hat{y}_{ij}$ is the predicted probability. [1]

- **Optimizer:**

AdamW is a stochastic gradient descent method that combines the adaptive estimation of first-order and second-order moments (as in Adam) with decoupled weight decay regularization. [9]
Gradient Update Rule:

$$\theta_t = \theta_{t-1} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \cdot \theta_{t-1} \right)$$

Weight Decay:

$$g_t = \nabla f(\theta_t) + w_t \theta_t$$

- **Batching and Epochs:**

Training uses a **batch size** of **16** and a **learning rate** of **0.00005** to balance stability and effective convergence.

5 Experiments

This section describes the experiment set up, methodology and evaluation metrics

5.1 The Set Up:

Objective: The aim of these experiments was to evaluate the performance of the model when classifying images of blood cells into their respective class.

Data: To keep a fair environment to evaluate the models, we used the data described before and the same seed for all experiments and models.

Models: We conducted our experiments on the following models: VGG-19, ResNet101V2 and EfficientNetV2S. For each model we focused on changing the number of frozen layers, the number of the dropout, the learning rate and optimizer. However, for sake of clarity, we are only going to show the best-performing results if each model.

Metrics: To measure the performance we used the following evaluators: Accuracy, Precision, Recall, and F1 score.

5.2 Data Obtained

Table 1: Results of experiments

Model	Accuracy	Precision	Recall	F1 score	Codabench Accuracy
VGG-19	98.75	98.75	98.75	98.75	60
ResNet101V2	98.92	98.92	98.92	98.92	59
EfficientNetV2S	**99.04**	**99.04**	**99.04**	**99.04**	**77**

6 Results

Comparison: EfficientNetV2S with fine tunning achieved the highest accuracy of 77%, significantly surpassing our basline VGB of 60%. Eventhough the the rest of the parameters are rather similar. **Key Achievements:** The implementation of EfficientNet_V2 and it's fine tunning resulted in a 17% improvemnet compared to our base model. Furthermore the advanced data augmentation techniques contributed to the generalization of the network. **Unexpected Outcomes:** What surprised us the most was the size we ended up with of the data set. Additionally, we were once again remained that sometimes, the simplest solution is the best.

7 Discussion

The model, based on **EfficientNetV2S** [5], balances performance and efficiency, with techniques like **Batch Normalization** [7], **Dropout** [11], and **transfer learning** [13] improving generalization to biomedical data. Optimization is enhanced through the **AdamW optimizer** [9], a dynamic scheduler (**ReduceLROnPlateau**) [2], and **class weights** to address imbalance, while **early stopping** and **data augmentation** [4, 6] further improve robustness. However, the model's accuracy of **0.77** (23% error rate) raises concerns in medical contexts, and its **complexity**, combined with **long training times** and **high memory demands**, presents practical challenges. The fixed image size (**96x96**) limits detail capture, and increasing resolution would exacerbate **resource constraints**, highlighting the need for careful optimization of training strategies.

8 Conclusions

The project involved a balanced team effort. Ariadna focused on dataset analysis and outlier removal, Carlos developed a data augmentation strategy, Marine optimized model parameters, and Aman researched performance enhancements. Ariadna and Marine prepared scripts while Carlos and Aman also handled the computation. The report sections were fairly divided.

Suggestions for Improvement: Implement a progressive unfreezing strategy [13] to optimize fine-tuning, explore curriculum learning [8] or cyclic learning rates [10] to enhance convergence, use k-fold cross-validation [12] for more reliable evaluations and to prevent overfitting.

Propose future work: Implement real-time data processing for practical performance evaluation, use multi-modal data inputs for richer predictions, apply transfer learning from larger, diverse datasets to boost robustness.

References

[1] T. Developers. Categorical crossentropy, 2024. Accessed: 2024-11-24.

[2] T. Developers. Learning rate schedules in tensorflow, 2024. Accessed: 2024-11-24.

[3] T. Developers. One hot encoding, 2024. Accessed: 2024-11-24.

[4] K. A. Documentation. Augmix, 2024. Accessed: 2024-11-24.

[5] K. A. Documentation. Efficientnetv2, 2024. Accessed: 2024-11-24.

[6] K. A. Documentation. Randaugment, 2024. Accessed: 2024-11-24.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[8] S. Liu et al. Learning from multiple data sources: A survey. *arXiv*, 2021. Accessed: November 2024.

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

[10] PyImageSearch. Cyclical learning rates with keras and deep learning, 2019. Accessed: November 2024.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[12] StackOverflow. Cross-validation in keras, 2018. Accessed: November 2024.

[13] TensorFlow. Transfer learning with keras, 2023. Accessed: November 2024.