A Peripheral Blood Cell Classification App

Correlation one. Data Science for All, Colombia. Team # 14.
Michael Mendez, Juan David Bejarano, Félix González, Julián Correa, Camilo Cortés, Santiago Alférez
Https://github.com/Fag42/Blood-Cells-Classifier

1. Introduction

Thirty trillion cells can be found within the human body which are divided into approximately two hundred different types. Some main cell types are stem cells, nerve cells (neurons), blood cells (red, white and platelets), muscle cells, and cartilage cells. Therefore, studying cells helps us to understand how the human body works. Peripheral blood cells, for instance, are the cellular component of blood such as red blood cells, which carry oxygen throughout our body, white blood cells, which helps to fight infections or participate in immune responses, and platelets, small cell fragments that are vital to proper blood clotting. Those cells have been studied from blood samples in labs over time to diagnose several diseases. Those diseases include rare genetic disorders, cancer, anemia, HIV, diabetes, and the current pandemic covid-19.

Blood cell classification is one of the most challenging tasks in blood diagnosis. Performing the classification of the cells through the manual procedure is difficult, prone to errors, and time-consuming due to the involvement of human labor. Also, for the manual segmentation, the experts make use of the advanced equipment, which cannot be adopted in rural areas.

The cell classification accuracy may also depend on the capabilities and experiences of the technical experts. Therefore, the necessity of an automated recognition system becomes inevitable. Deep learning methodology can help us to make this whole process more efficient and faster which potentially could save resources and lives. In addition, the experience gained in this work will be the basis for further extensions of convolutional neural networks (CNN) models to classify the broader classes of abnormal cells related to acute leukemia and lymphoma, such as myeloblasts, lymphoblasts, abnormal B or T lymphoid cells, and dysplastic cells that could help to diagnose.

Training a deep convolutional neural network (CNN) from scratch is challenging because it requires a large amount of labeled training data and a great deal of expertise to ensure proper convergence. A promising alternative is to fine-tune a CNN that has been pre-trained using, for instance, a large set of labeled natural images. This is the approach we used in this study, based on three different architectures: Resnet34, Resnet18 and EfficientNetB5.

We compare the three architectures and chose the one who offered the best accuracy on the validation set. Then, a fine-tuning process through Data Augmentation, finding the optimal Learning Rate and analyzing the deep features with methodologies like GradCam and t-SNE, PCA and K-means was made. As a result, we obtained a model with an accuracy on the test set of 99.94%.

1.1 Background

The peripheral blood smear is a fundamental part of the interpretation of the hemogram. It allows the morphological interpretation of the quantitative data produced by conventional hemogram. When it is properly interpreted and analyzed, it constitutes an element of great importance for medical personnel when making a correct diagnosis[1]. A peripheral blood smear is essential for the diagnosis of 80% of hematological diseases, hence the importance of making an accurate and rapid analysis that helps both laboratory and medical personnel[1].

In the hemogram, cells in the blood are evaluated. The main ones are Erythrocytes, Leukocytes and Platelets where different specific subtypes are found according to their function. In this project, we will focus on eight (08) specific subtypes (Figure 1) that through practical analysis are more frequently observed in infections and regenerative anemia[1].

We will first introduce the critical concept of Hematopoiesis followed by a description of the eight (08) different subtypes of peripheral blood cells. Hematopoiesis is the production of blood cells in the human body which is carried out in the bone marrow where erythrocytes, platelets and immature leukocytes are produced daily. These cells are differentiated and assigned with specific functions during cell maturation and hematopoiesis based on their cellular signaling. If failures are present in this process of cell maturation and signaling, we could observe abnormalities in the hemogram and peripheral blood smear. Throughout these diagnoses, one could diagnose the different pathologies associated with bloodstream cells.

Neutrophils: Neutrophils are polymorphonuclear leukocytes (PMN). They make up 50-70% of white blood cells and are considered the first line of defense against bacterial and fungal infections. They are 10-15 µm in size, their nucleus is seen as 2 to 5 lobes connected by small filaments, and their cytoplasm is colored pale pink or colorless[3].

Eosinophils: Eosinophils are white blood cells that are part of the so-called granulocytes, among their main functions are allergic responses, and defense against parasites. They measure between 12-15µm, it has a nucleus of several lobes joined by non-visible chromatin filaments, the cytoplasm is cream to pink in color, and sometimes it can have irregular edges.

Basophils: Basophils constitute less than 1% of leukocytes in humans, but they are the only circulating cells that contain histamine. They are important to trigger inflammatory reactions[3]. Its normal value is 0-150 per µL. Basophilia is defined as the increase with respect to the expected value where the increase is suggestive of myeloproliferative disease[4]. As for the decreased value which is called Basopenia, which like eosinopenia, it is difficult to detect, and is normally found in periods of ovulation[7], and in patients with urticaria[8].

Lymphocytes: Lymphocyte is a type of white blood cell found in the blood and lymphatic tissue. There are two subtypes of lymphocyte: B lymphocytes and T lymphocytes. They make antibodies and participate in immune and tumor responses. Their normal value is between 1000 to 4000 µL, being increased in newborns[4]. Morphologically small, medium and large lymphocytes can be found in a peripheral blood smear.

Monocytes: monocytes are white blood cells that in the blood stream becomes a macrophage, which encompass and destroy foreign microorganisms. They are the largest white blood cell measuring between 15-30 µm. The nucleus is slightly centered in the cytoplasm and is horseshoe-shaped, its cytoplasm is grayish blue[3]. Its normal value is between 50-900 µL. Monocytopenia is when its value is below what is expected, it is usually found in patients receiving corticosteroid therapy or in acute infections[4]. The increase above the expected value is known as monocytes, and it occurs in newborns, pregnant women, and chronic infections[4].

Immature Granulocytes (IG): immature cells (IG) are the cells in charge of the production in their different stages of blood cells and they are in the bone marrow. Although it is normal to find a minimum percentage in the bloodstream, a great proportion will suspect a malignant disease. Within these cells we can find Metamyelocyte which is normally observed in a percentage between 10% and 30% in the bone marrow It measures between 10-18 µm with a rounded or oval nucleus, and it is much smaller than that of the previous ones. Its cytoplasm is pink to light purple. If they are found in peripheral blood, it is associated with myeloproliferative disease or chronic leukemia[10].

Erythroblasts: Erythroblasts are precursor cells of erythrocytes with a measure of approximately 12µL. Its nucleus is rounded with a great size with respect to the cell. They are not normally circulating, but if they are present, it is because of malignant or premalignant diseases such as myelopathy anemia, myelodysplastic syndromes, or leukemias[11].

Platelets: Platelets are essential blood cells for hemostasis and are the main ones involved in thrombosis, bleeding disorders, and hereditary or acquired thrombotic events[4]. They are approximately 3 µm in diameter, and are observed as small biconvex discs[13]. Its normal value is between 150,000 and 450,000 platelets / mm3. When platelets are found at a lower value than expected, it is called thrombocytopenia. It is commonly caused by disorders such as leukemias, destruction by the spleen or liver. The increase above its normal value is known as thrombocytosis and is normally caused by acute bleeding, cancer, or hemolytic anemia[4].

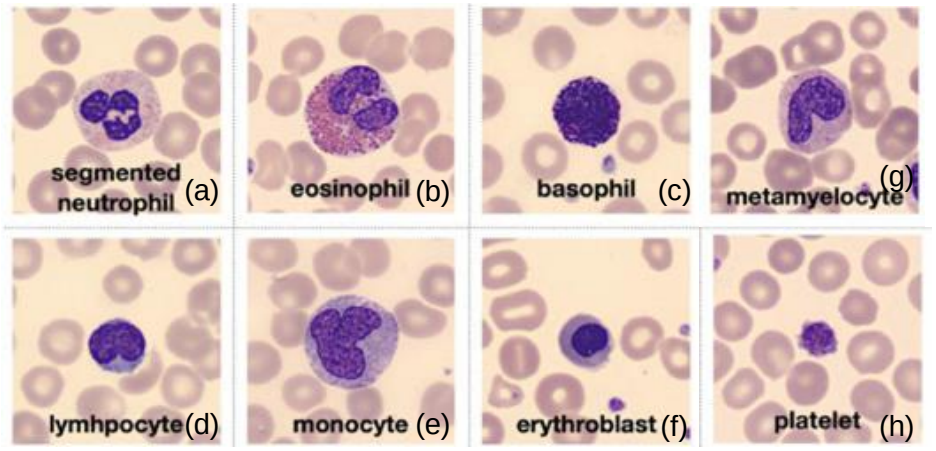

Figure 1. Images of the eight (08) different types of normal peripheral blood cells in the dataset[2].

2. Application Overview

The application allows loading the images to be classified. Once the images are loaded, the user will have three (3) tabs available to interact with the loaded images, and a help tab that explains the results of the classification. The interaction tabs for the user are:

Help: This section informs the user in general, how the application works and how each tab should be used.

Chart: Allows the user to quickly identify the amounts of images classified in each cell type with a bar chart and a pie chart.

Classification: Allows the user to view how the images were classified according to the types of cells available.

Visual Analysis: The most interactive tab since it shows the user through a scatter diagram the type of cells images loaded divided by zones. When clicking at each point, which represents a peripheral blood cell, shows the loaded image file already classified.

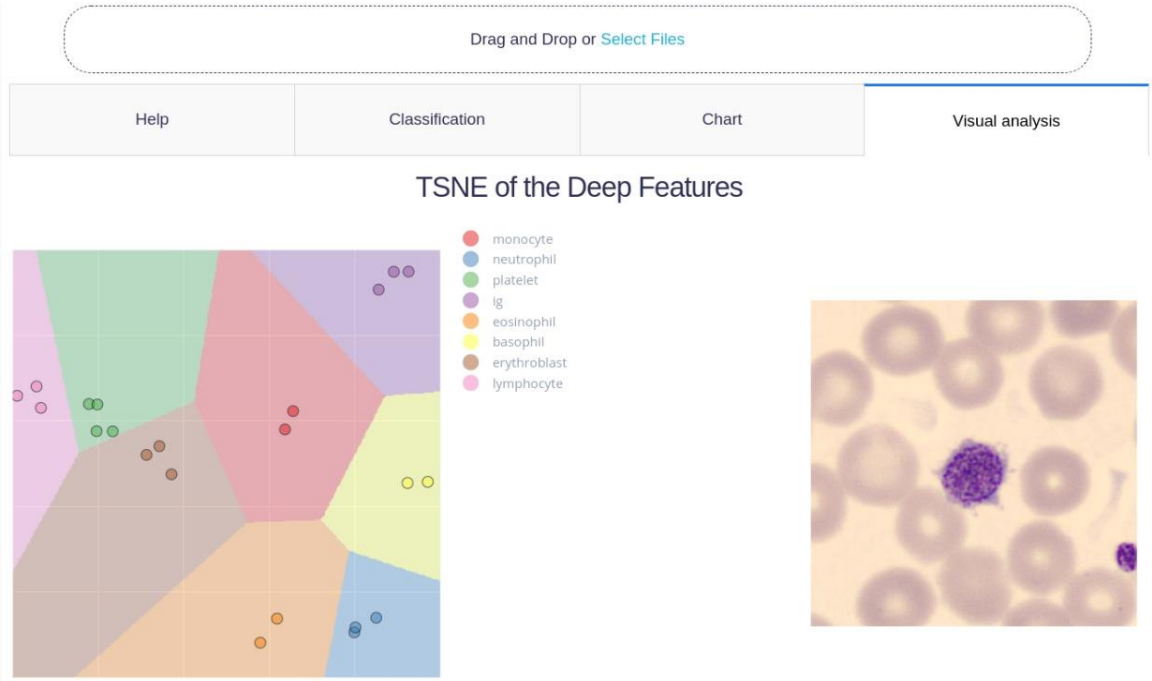

Figure 2. Visual Analysis tab of the Blood Cells Classifier App

3. Data Engineering

3.1 Interactive Front-end

The application interface was built in Dash with plot.ly including some components of seaborn and bootstrap for a better visualization. The interface allows images to be loaded which go through a transformation process and then pass them to the classification algorithm. The classification algorithm by predicting the type of cell to which each image belongs is a CNN, the display of the data is achieved through the features deep applying the technique t-SNE achieving segmenting images so that the borders dividing the cell types can be graphically identified, this is shown by means of a scatter plot.

3.2 Infrastructure

The application in production runs on a virtual machine hosted in Azure Web Services which has 4 processing cores and 16gb of RAM. In order to minimize host costs, we performed the classifications using CPU with models pre-trained locally with a Nvidia 1080 super graphics card.

The Azure virtual machine runs Ubuntu 20.04 LTS as operating system and manages the production model with Docker in a modified Docker image based on Ubuntu 20.04. It includes the software and libraries necessary to run the web application under the https protocol managed by the Gunicorn server library.

The virtual machine's internal control is carried out using the ssh protocol. The control of development and production versions is managed by Git. The application is externally accessible thanks to the dynamic NO-IP address control service and the security certificates provided by Let's Encrypt service.

4. Data Analysis and Computation

4.1 Dataset

The dataset consists of 17092 digital images corresponding to eight (08) different peripheral blood cell types as it is shown in Table 1.

Cell Type	Number of Images
Neutrophils	3329
Eosinophils	3117
Basophils	1218
Lymphocytes	1214
Monocytes	1420
Immature Granulocytes (metamyelocytes)	2895
Erythroblasts	1551
Platelets (thrombocytes)	2348
Total	17092

Table 1. Description of the blood cell images dataset

The blood smears were automatically stained using the May-Grunwald Giemsa technique which combines the effect of acidic eosin and alkaline methylene blue. This method has been widely used for morphological inspection and differential counting of blood cells. Then, the images were obtained from the analyzer CellaVision DM96 in the Core Laboratory at the Hospital Clinic of Barcelona. Clinical pathologists logged the images and their size is 360 x 362 pixels (high quality), in JPG format [2]. The cells images belong to people without infection, oncologic, or hematologic disease and without any pharmacological treatment by the time the samples were taken. An example of each cell type is illustrated in Figure 1.

4.2 Data Preprocessing

Data should be formatted into appropriately preprocessed floating-point tensors before being fed into the network. Currently, the data sits on a drive as JPG files, so the steps for getting it into the network are approximately as follows:

1. Read the picture files.
2. Decode the JPG content to RGB grids of pixels.
3. Convert these into floating-point tensors.
4. Rescale the pixel values (between 0 and 255) to the [0, 1] interval (as you know, neural networks prefer to deal with small input values).

It may seem a bit daunting, but fortunately fastAI has utilities to take care of these steps automatically. FastAI has a module with image-processing helper tools. In particular, it contains the class ImageDataBunch, which allows a quickly set up for Python generators that can automatically turn image files on disk into batches of preprocessed tensors.

4.3 Exploratory Data Analysis

We will begin validating the quality of the images and percentages of each class in the training set. There are 8 folders containing the images of every cell class. We will split randomly the data into 2 datasets: training set (80%) and validation set (20%). The function data.show_batch can be used to show some of the contents on the data set (Figure 2). They are all properly cropped for our model. A representation of the percentages is shown in Figure 2. We have a very proportionate dataset for each class. This concept is very critical since an unbalanced dataset could impact the accuracy of the neural network.

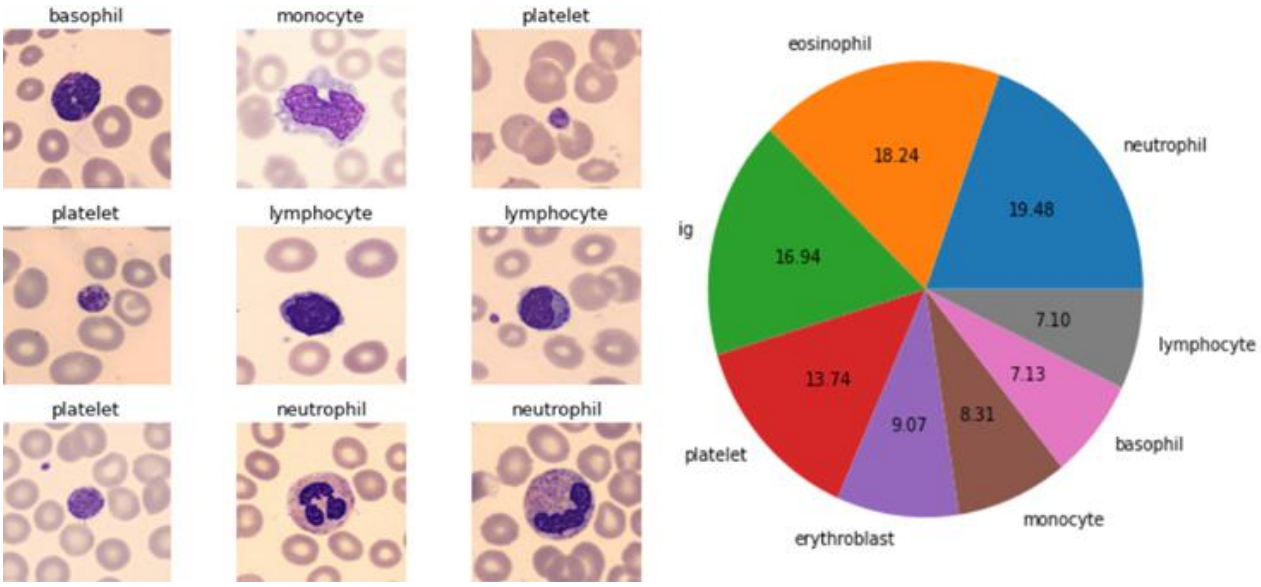

Figure 3. Peripheral blood cells and their respective distribution (%) in the dataset

t-SNE and PCA for Exploratory Data Analysis

A popular method for exploring high-dimensional data is called t-SNE (t-distributed Stochastic Neighbor Embedding), introduced by van der Maaten and Hinton in 2008. The goal is to take a set of points in a high-dimensional space and find a faithful representation of those points in a lower-dimensional space, typically the 2D plane.

A random sample of 1000 cell images were taken. Then, we created an array that contains all the images resized to 64 x 64 pixels, considering only the green component. Finally, we used t-SNE embedding to visualize each type of the 1000 cells in a scatter plot. Figure 4 represents a visualization of this technique. t-SNE is mostly used to understand high-dimensional data, and to project it into low-dimensional space (2D or 3D) for a more useful handling with CNN (convolutional neural network) . However, it is not possible to see relative sizes of clusters in a t-SNE plot. Likewise, distances between well-separated clusters in a t-SNE plot may mean nothing. This is why we combine the analysis with Principal Component Analysis (PCA), which preserves large pairwise distance maximize variance. A visual representation of PCA is shown in Figure 5.

Figure 4. t-SNE embedding over 1.000 cells images

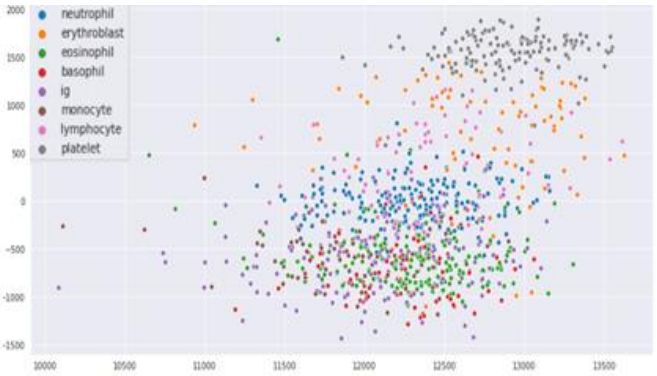

Figure 5. PCA from 1.000 cells images

At this point, if we look at the visualizations, and without the implementation of any machine learning technique, we can see that *platelets* and *erythroblast* may be different from the rest of the classes. Therefore, based on the previously explained and an extra research on the literature, both cell types are the most differentiated with respect to the others in size and shape. So far, we have just reduced the dimensionality from the images. Let's now use K-means clustering for the first 2 principal components (PCA) of the 1.000 blood cell images. Given we have 8 classes, we would use 8 clusters as it is seen in Figure 6.

Figure 7 represents the K-means algorithm with only 3 clusters. The orange points (cluster 1) seem to match the platelets, while the blue points seem to match the erythroblasts. The green points are the remaining blood cells. A first EDA over 1000 blood cells, resized to 64 x 64 pixels and green color component only, has shown a very nice separation between the platelets, erythroblasts and the other cells. We have used t-SNE to visualize in two dimensions, and PCA to observe the first two principal components and clustering purposes.

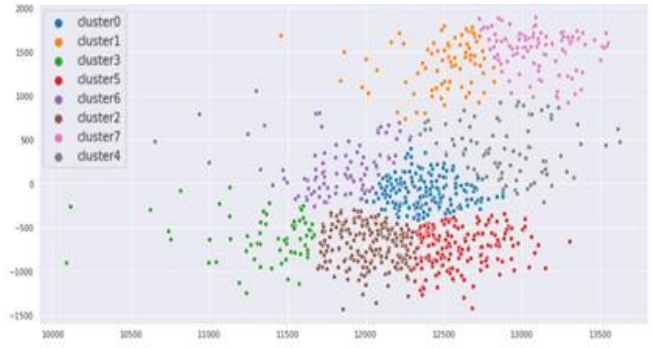

Figure 6. K-means clustering with 8 clusters

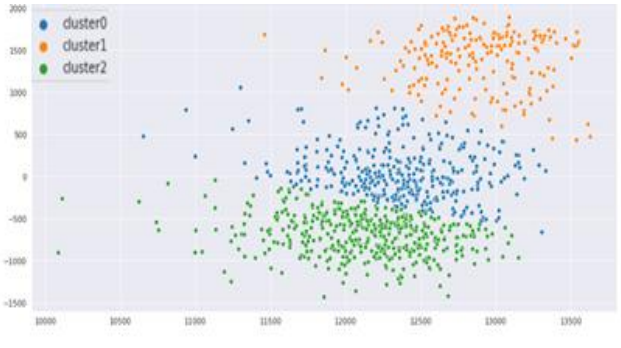

Figure 7. K-means clustering with 3 clusters

4.4 Model and Statistical Analysis

A promising alternative to training a CNN from scratch is to fine-tune a CNN that has been trained using a large labeled dataset from a different application. The pre-trained models have been applied successfully to various computer vision tasks as a feature generator or as a baseline for transfer learning. Therefore, we validated three different architectures that used pre-trained models and then we made the fine-tuning process to improve the accuracy of the model.

4.41 ResNet18

ResNet-18 is a CNN with eighteen (18) layers deep. We used the pre-trained version of the network which was trained on more than a million images from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, pencils, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. We will be using two different metrics to look at our training success; **accuracy:** validation accuracy and **error_rate**: validation error rate. The results are shown of Figure 8. The validation error was 2.8% after eight epochs.

epoch	train_loss	valid_loss	error_rate	time
0	0.733341	0.322197	0.100351	02:11
1	0.357591	0.181339	0.065535	02:09
2	0.232531	0.137488	0.050907	02:09
3	0.188233	0.109318	0.037449	02:08
4	0.144873	0.093134	0.033645	02:08
5	0.132003	0.083267	0.030427	02:09
6	0.100728	0.079354	0.030427	02:09
7	0.094805	0.076339	0.027794	02:09

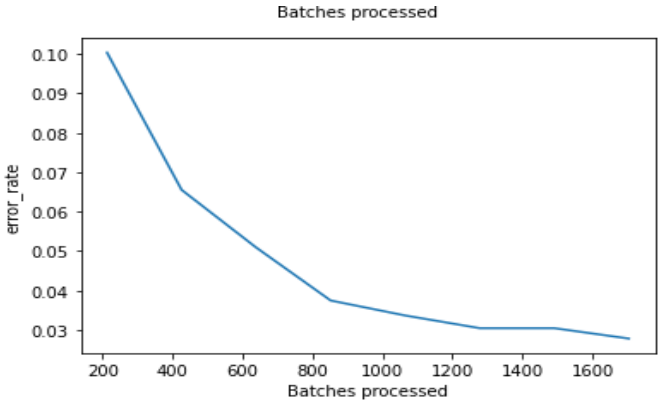

Figure 8. Validation error and performance of the ResNet18 architecture.

4.42 ResNet34

This is a backbone and a fully connected CNN with a single hidden layer as a classifier. It's called Residual Network with 34 layers (ResNet34). This particular powerful model with a proper size, and it has already been trained on looking at about one and a half million pictures of all kinds of objects and a thousand different categories using an image dataset called ImageNet. In the first run, we obtained an error rate of 3,5%. This is a good result for the first model without any tuning. the results of this model for our dataset are shown in Figure 9.

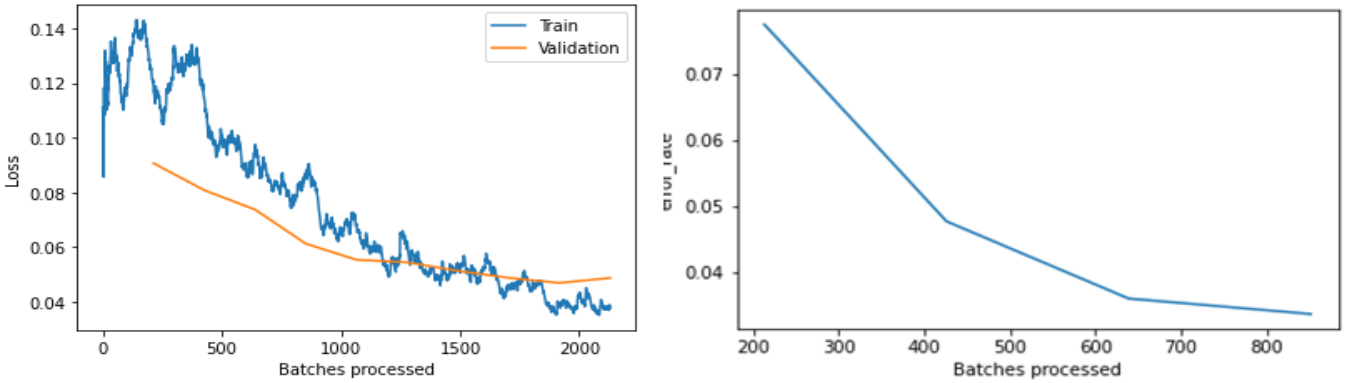

Fig 9. Results on the ResNet34 architecture without fine-tuning.

It's interesting that the error rate is higher than the ResNet18, which is a less complex model. Given the fact that it would be preferred to work with a less complicated model. Now, let's analyze the third architecture: EfficientNetB5.

4.43 EfficientNet

ResNet architecture has several variations ranging from 18 to 202. Although a deeper or wider CNN may increase the performance, it would come at a great computational cost. In 2019, Google experts proposed a novel model scaling method that uses a simple yet highly effective compound coefficient to scale up CNNs in a more structured manner called EfficientNet. The model surpasses state-of-the-art accuracy with up to 10 times better efficiency (smaller and faster). The baseline EfficientNet architecture is shown in figure 10.

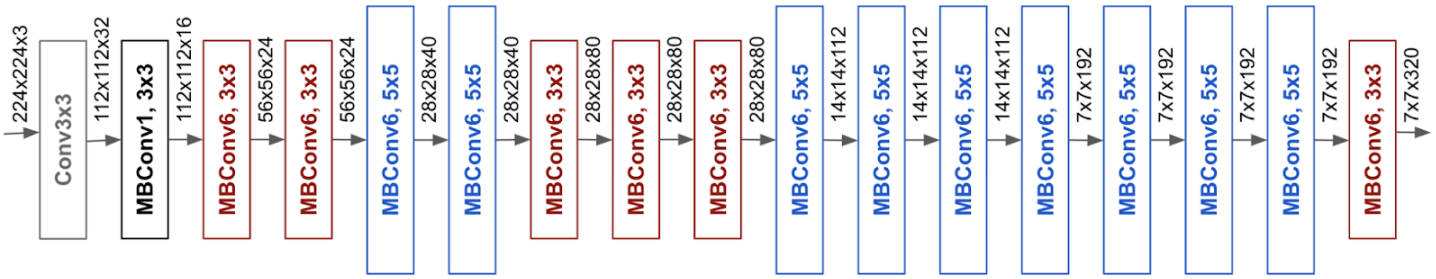

Figure 10. EfficientNet Architecture16

We trained three different models of the EfficientNet. EfficientNetB0 developed an improper and underfitted result for the cell classification. Therefore, we decide to work with a more robust model such as EfficientNetB5. We trained this model with the pre-procced weights it contains. In the first run, we obtained a better result than the previous models (1.7% validation error), so we decided to continue the fine-tuning process with this model. Results of the first run with EffcientNetB5 is shown in Figure 11.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.119028	0.090664	0.968403	0.031597	02:21
1	0.114325	0.080798	0.970158	0.029842	02:22
2	0.094860	0.073717	0.974254	0.025746	02:22
3	0.085690	0.061218	0.977472	0.022528	02:23
4	0.068156	0.055324	0.980690	0.019310	02:23
5	0.061133	0.054389	0.980398	0.019602	02:22
6	0.053241	0.051265	0.982738	0.017262	02:21
7	0.045715	0.048703	0.983031	0.016969	02:22
8	0.038766	0.046861	0.983616	0.016384	02:22
9	0.037610	0.048666	0.982446	0.017554	02:22

Figure 11. Performance of the first run on EfficientNet architecture without fine-tuning.

Fine-tuning Process

Our fine-tuning process is composed by the exploration and modification of three main elements: hyperparameters, overparameterization and regularization.

Hyperparameters

It's well known that the same hyperparameters don't work for all models, otherwise we would just use the globally "optimal" learning rate, batch size, etc. It could be the case that the larger models require higher/lower learning rates to perform well.

The best way to choose the learning rate for the next fitting could be consider "art". Some recommended methods

include choosing the LR at the steepest decline of loss or 10x prior to the minimum loss. We used an “interval slide rule” technique that shifts right to left on a flatter loss gradient plot of the learning rate finder. It progresses until the loss value of the right interval bound comes within a close-enough distance to the left interval bound. The left interval bound is then taken as the selected learning rate, with adjustment implementable as a multiplier argument. Learning Rates vs Loss of our baseline EfficientNet model is shown in Figure 12.

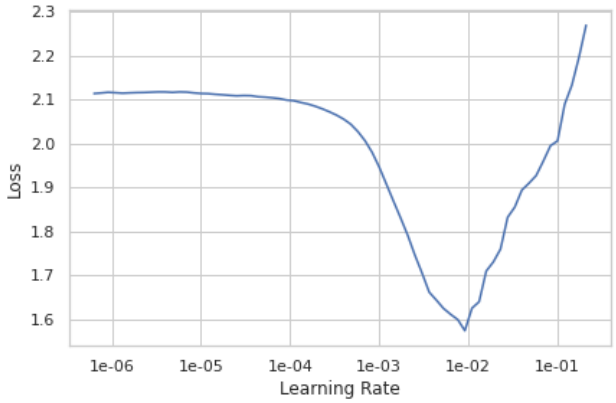

Figure 12. Learning Rate vs Losses to identify the optimal Learning Rate.

Based on the Figure above, we can use slice function to logarithmically distribute the learning rate between 1e-3 to 1e-2 for different layers in the network. Keeping the lowest learning rate for the initial layers and increasing it for later layers.

We experimented with different batch sizes (16, 32, 64 and 128). For sixteen (16) and one hundred twenty-eight (128) the model did not fit. The best result was with a batch size of 64. It’s also important to validate the number of Epochs we used in order to avoid underfitting and also overfitting. Consequently, we want to see our losses on the learning curve as shown in Figure 13. The model without tuning (left) shows the training loss higher than the validation loss which is an indication of underfitting. A tuned model with the proper Epochs has a training loss fewer than the Validation loss.

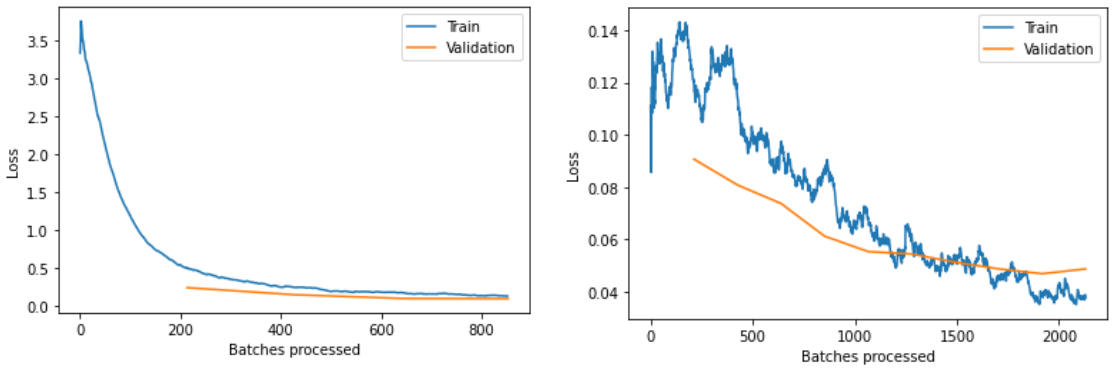

Figure 13. Underfitted model (left) with few Epochs vs correct model (right).

Overparameterization

The largest EfficientNet (EfficientNetB7) has over 60 million parameters. The ResNet architecture has over 23 million if we account for the trainable parameters. It was important to us to define which of the EfficientNet models were the most suitable for us. After iterating processes and analyzing, we identified that the best model for us taking into account the bias-variance trade-offs and the computational challenge was the EfficientNetB5.

Regularization

This is one the most important aspects of any machine Learning technique. This would help us to control the overparameterization issue. In our problem, we include some Data augmentation techniques that works as regularization elements. Data augmentation is perhaps the most important regularization technique when training a model for Computer Vision. Instead of feeding the model with the same pictures every time, we do small random transformations (a bit of rotation, zoom, translation, among others) that don't change what's inside the image (to the human eye) but do change its pixel values. Models trained with data augmentation will then generalize better. We added various data augmentation options during the training to reduce the chance of overfitting

- Flip vertically
- Random_scale
- Random brightness
- Add a reflection padding
- Rotation
- Randomly zoom and/or crop

Figure 13 shows how our cells look like after the Data Augmentation process. After several experiments, we concluded that high transformations like zooming and warping reduces the accuracy of our model. It makes sense the size and the shape of the cells are the most important features. Therefore, high modifications of these elements produce a reduction in accuracy. The best combination we found was to apply vertical flip, maximum zoom of 1.15, maximum rotation of 45 degrees, and a maximum warp of 0.1.

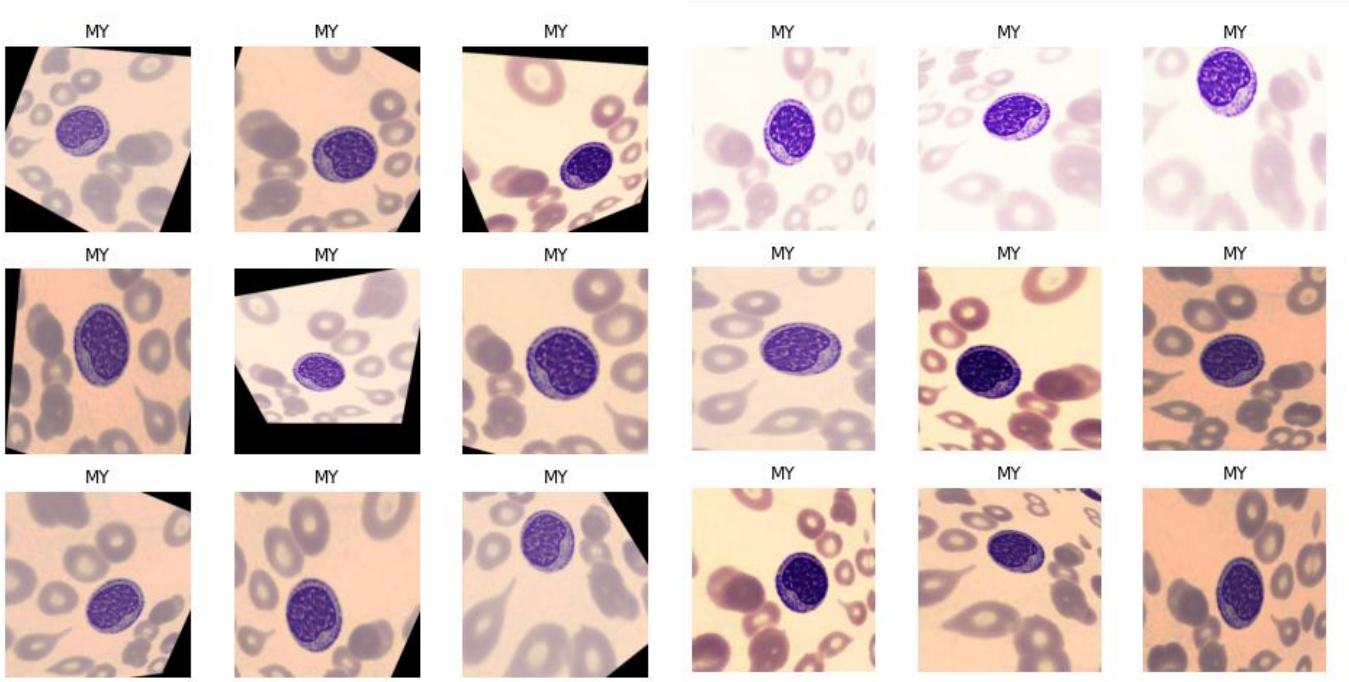

Figure 13. Data Bunch of the cells after different Data Augmentation configurations

4.5 Results and Interpretation

After working with the three different models mentioned above, we found that the EfficientNet is the best since it produces less error. The final model results are shown in Figure 14. Now, we have an error of 0.06% after fine-tuning, which is extremely extraordinary. Given the error rate, we don’t see signs of overfitting. We could continue training with more epochs, but it is good enough for us given the computational cost we already have.

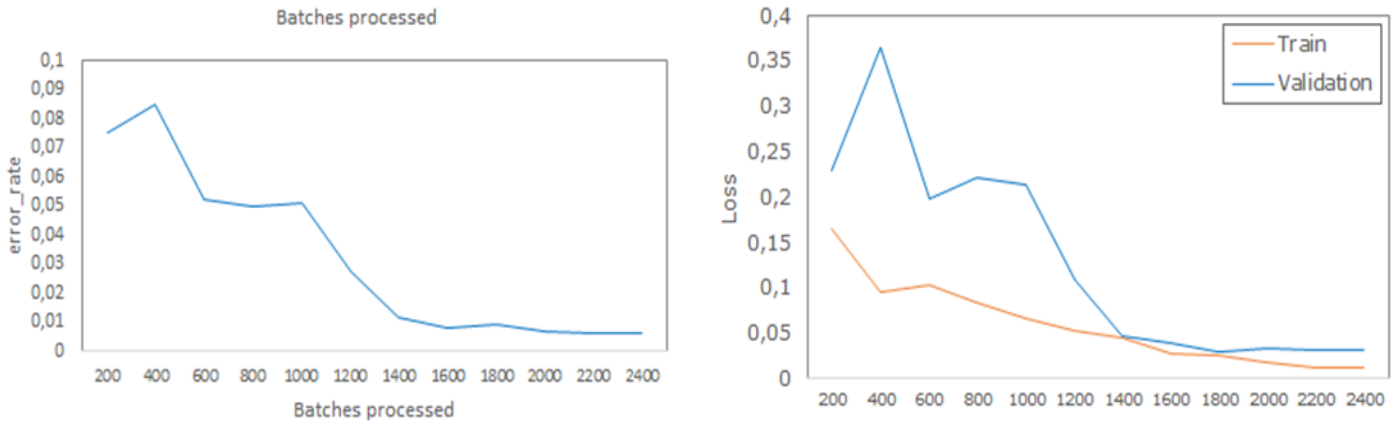

Figure 14. Results on the final model after fine-tuning. Validation error is 0.6%.

Now, for the interpretation of these results, one of the most useful procedures to do is called plot top losses. By plotting the top losses, we will find out what were the parameters that we were the most wrong on. A representation is shown in Figure 15. This can help us to identify the elements that are causing most of the errors. One important tool to consider is the Confusion Matrix as it is seen in Figure 16. We can identify that the model's top misclassifications are some neutrophils which are misclassified as IG.

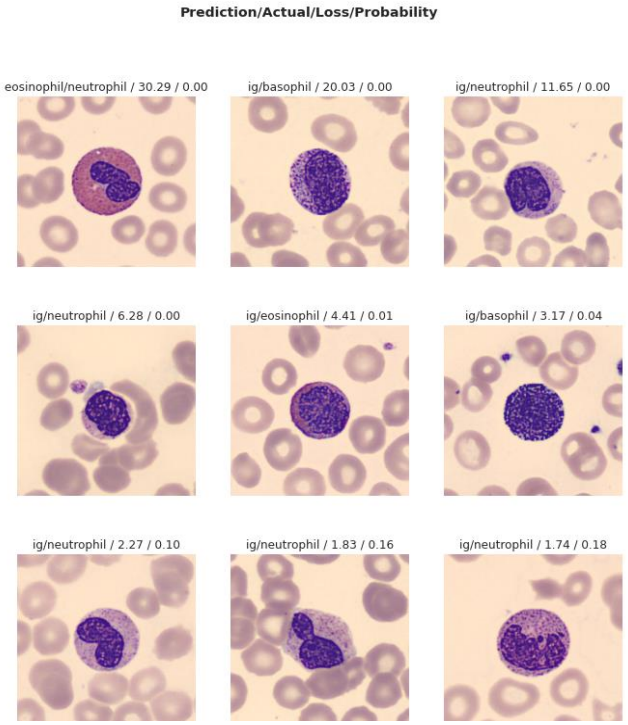

Fig 8. Tops losses

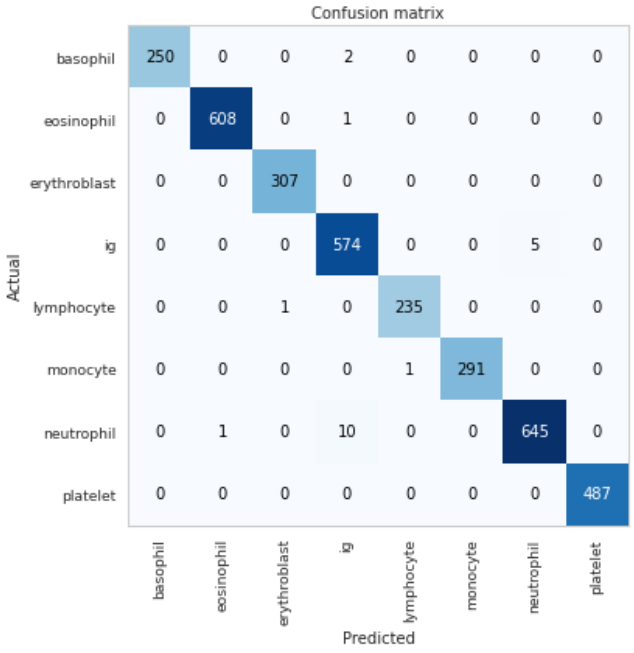

Fig 9. Confusion Matrix on the first experimental model

We propose a technique to produce "visual explanations" for decisions from a large class of CNN-based models making them more transparent. Our approach Gradient-weighted Class Activation Mapping (Grad-CAM) uses the gradients of the final convolutional layer to produce a coarse localization map highlighting important regions in the image for the cell classification.

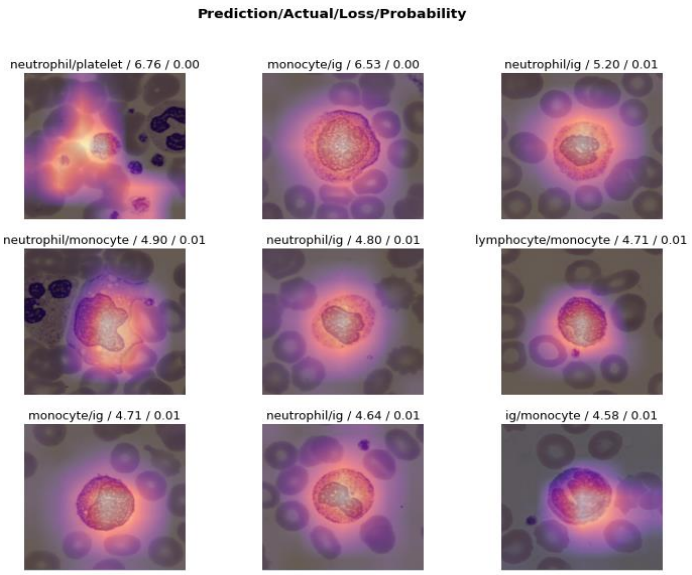

Fig 18. Grad-CAM of the top Losses which highlights important regions to predict the classes

Another technique for the visual interpretation of the model results is t-distributed Stochastic Neighbor Embedding (t-SNE). t-SNE was used to visualize the high-dimensional relationships of the eight (8) learned classes onto a two-dimensional plane. Specifically, we plotted a random selection of a thousand (1.000) training images for each class as seen in figure 19. At first, we thought we had a misclassification since some peripheral blood cells were assigned to an improper class. However, after making a review of those cells, we could identify that some of them were mislabeled on the dataset.

Further optimization was carried out to automate the removal of misclassified test images or tiles containing features of multiple classes. In order to remove these potentially anomalous points, we compared each point on the t-SNE plot to its nearest 50 neighbors to determine if points were substantially deviated from their labeled class cluster. This provided a refined visual plot that highlighted the learning relationship of representative tiles and classes to one another.

For each blood cell class, a prediction was generated using the probability distribution scores of the CNN's final SoftMax output layer or the tile distribution overlaid on the t-SNE or PCA plot of the CNN's final hidden layer. This was accomplished using hooks which is a method to extract the desired feature maps. Afterwards, we applied a clustering technique over the test set and now we have a representation of 1290 deep features in 2D as seen in figure 20. It's possible to see how near are the neutrophils with the IG cells which could explain the top losses between those two classes.

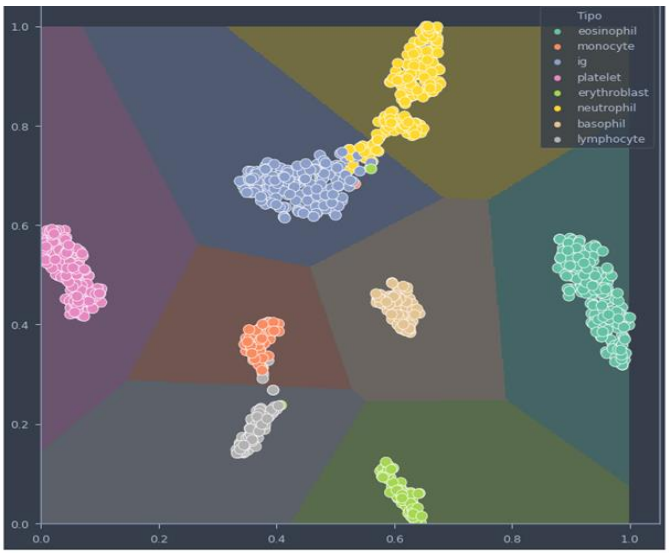

Figure 19. t-SNE over 1290 deep features (training set).

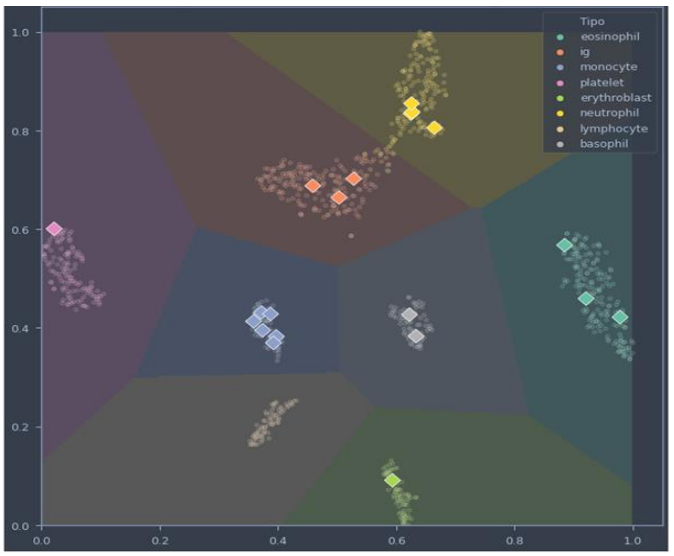

Figure 20. t-SNE embedding over 1290 deep features (test set)

5. Conclusions and Future Work

This work introduces an interactive and automated peripheral blood cell classification App. Blood cell classification is a challenging and time-consuming task. Due to technological advances, the process has become more efficient but it requires great capital expenditure and technical experience personnel. Some health care facilities, especially in rural areas, lack automated analyzer or advanced microscopes. Therefore, a manual and laborious procedure is required which could be prone to errors.

Our App allows users to load and classify peripheral blood cell images. The user will have three (3) tabs available to interact with the loaded images, and a help tab that explains the results of the classification. It only requires an internet connection. In addition. the current App works independently of the image type, so the interface will adapt to new classes, in other words, we could use a model that classifies tissues for instance and run it in the App.

We explored the baseline performance of eight different peripheral blood cell class through CNN on a prospective set of 17.000 datasets of peripheral blood cells. Initially, we were able to analyze the cells without applying any sophisticated machine learning algorithm. It was done through exploratory data analysis using methods such as t-SNE and PCA. At that point, we could state that platelets and erythroblast were different from the rest of the classes. Then, we performed experiments with three (3) CNN architectures: RestNet18, ResNet 34 and EfficienNet B5.

EfficienNetB5 performed better than the other two architectures with an initial accuracy on the test set of 98.3%. Then, we wanted to improve our results of the EfficientNet B5 so we applied different fine-tuning parameters like the Learning Rate, Epochs, and Bath Size. In order to control overfitting problems, we modified different parameters of the Data Augmentation transformations like flipping, ward, zooming, and rotations. We realized that one should be really careful about these transformations due to the fact that the shape and the size are the most important features to classify the cells.

Using the distribution of prediction scores across the eight (08) classes of our final model, we obtained an incredible accuracy of 99.94% on the test set. Analyzing the top losses, the confusion matrix and the clustering of the deep features, we conclude that the neutrophils and the IG cells are the most similar and more prone to misclassification.

The experience gained in this work could be the baseline for further extensions of convolutional neural networks (CNN) models to classify the broader classes of abnormal cells related to acute leukemia and different kinds of diseases. For instance, we could use the weights of our network to start a new training of abnormal blood cells. We could also extend this classification for more type of peripheral blood cells that will help us to identify diseases such as Malaria, Zika, Covid-19.

The following step will be to build an object detection model to automatically crop the blood cells (red blood cell, white blood cells and platelets). Then, we can assemble our current CNN model to have a more broad and automated classification

References

1. Campuzano-Maya G. Utilidad clínica del extendido de sangre periférica: los leucocitos. Medicina & Laboratorio 2008; 14: 311-357.
2. Acevedo, A., Merino, A., Alférez, S., Molina, Á., Boldú, L., & Rodellar, J. A dataset of microscopic Peripheral blood cell images for development of automatic recognition systems. Data in Brief, 30, 105474. 2020 https://doi.org/https://doi.org/10.1016/j.dib.2020.105474
3. Watts RG. Neutropenia. In: Greer JP, Foerster J,Lukens JN, Rodgers GM, Paraskevas F, Glader B,eds. Wintrobe's clinical hematology. 11th ed. Philadelphia, PA, USA; Lipincott Williams & Wilkins. 2004: 1777-1800.
4. Bain BJ. Quantitative changes in blood cells. In:Bain BJ, ed. Blood cells. A practical guide. 4th ed. Malden, Massachusetts USA; Blackwell Publishing.
   2006: 217-262.
5. Lacy P, Becker AB, MOqbel R. The human eosinophil. In: Greer JP, Foerster J, Lukens JN, Rodgers GM, Paraskevas F, Glader B, eds. Wintrobe's clinical hematology. 11th ed. Philadelphia, PA, USA; Lipincott Williams & Wilkins. 2004: 311-334.
6. Campuzano-Maya G. Eosinofilia: las 125 causas más frecuentes. Medicina & Laboratorio 2005; 11:321-361.
7. Soni R, Bose S, Gada D, Potnis V. Basopenia as an indicator of ovulation (a short-term clinical study). Indian J Physiol Pharmacol 1996; 40: 385-388.
8. Rorsman H. Basopenia in urticaria. Acta Allergol 1961; 16: 185-215.
9. Caldwell CW. Evaluation of peripheral blood lymphocytosis. Columbia, MO: Academic Information Systems, (AIS); 2000: p.
10. Skubitz KM. Neutrophilic leukocytes. In: Greer JP, Foerster J, Lukens JN, Rodgers GM, Paraskevas F, Glader B, eds. Wintrobe's clinical hematology.
    11th ed. Philadelphia, PA, USA; Lipincott Williams & Wilkins. 2004: 267-310.
11. Bain BJ. Blood cells. A practical guide. 4th ed ed Malden, Massachusetts USA: Blackwell Publishing; 2006.
12. Campuzano-Maya G. Utilidad clínica del extendido de sangre periférica: los eritrocitos. Medicina & Laboratorio 2008; 14: 311-357.
13. Campuzano-Maya G. Utilidad clínica del extendido de sangre periférica: las plaquetas. Medicina & Laboratorio 2008; 14: 511-528.
14. Merino A. Manual de citología de sangre periférica. Madrid: Acción Médica; 2005.
15. Acevedo, A., Alférez, S., Merino, A., Puigví, L., & Rodellar, J. (2019). Recognition of peripheral blood cell images using convolutional neural networks. Computer Methods and Programs in Biomedicine, 180, 105020. https://doi.org/https://doi.org/10.1016/j.cmpb.2019.105020
16. Tan M. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. Google AI Blog. 2009. https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html