Systems & Biomedical Engineering Department

Blood cells Dataset

Prepared by:

Mohamed Kamal

Donia Abdel-Salam

Gehad Mohamed

Merna Saleh

Submitted to:

Dr. Inas Yassin

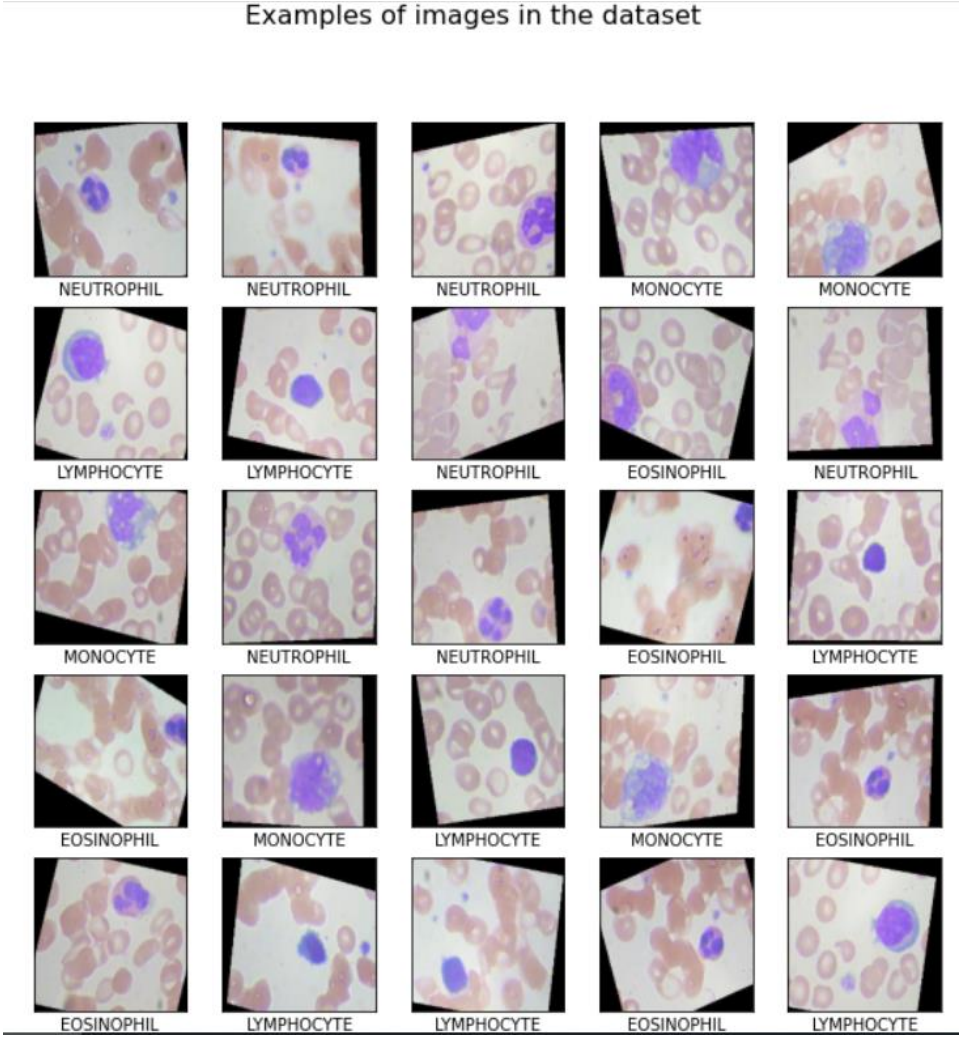
Eng. Christen Ramses

Problem:

Establishing an accurate count and classification of leukocytes commonly known as WBC (white blood cells) is crucial in the assessment and detection of illness of an individual, which involves complications on the immune system that leads to various types of diseases including infections, leukemia, cancer, anemia, AIDS, etc. The accuracy of these existing methods could still be improved. We proposed a method that could segment various types of WBCs: monocytes, lymphocytes, eosinophils, basophils, and neutrophils from a microscopic blood image using CNN (Convolutional Neural Network) for counting which in turn generates more accurate results. Also, we had used SVC and random forest models and we will describe why we had selected these three models to work with.

Description for data-set:

This dataset contains 12,500 augmented images of blood cells (JPEG) with a CSV file that contains the cell labels. There are approximately 3,000 images for each of 4 different cell types that are Eosinophil, Lymphocyte, Monocyte, and Neutrophil. This dataset is accompanied by an additional dataset containing the original 410 images (pre-augmentation).



Why we choose each model:

- **CNN:**

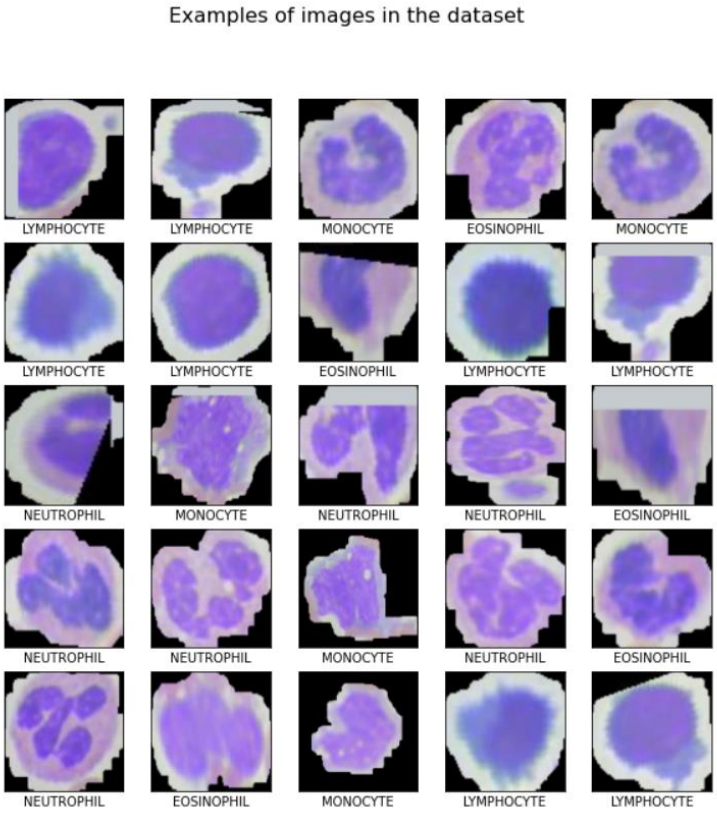
  Deep Learning has already shown power in many application fields and is accepted by more and more people as a better approach than the traditional machine learning models. In particular, the implementation of deep learning algorithms, especially Convolutional Neural Networks (CNN), brings huge benefits to the medical field, where a huge number of images are to be processed and analyzed.

- **SVC & Random Forest:**

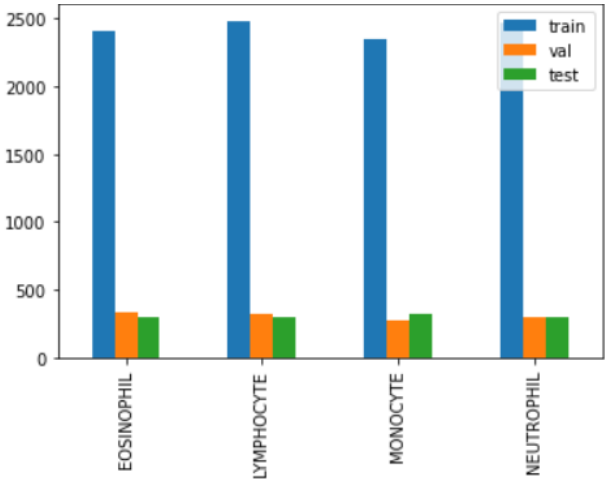
  In our data-set, we have a lot of features and they are the best choice to deal with such a dataset.

Pre-processing steps:

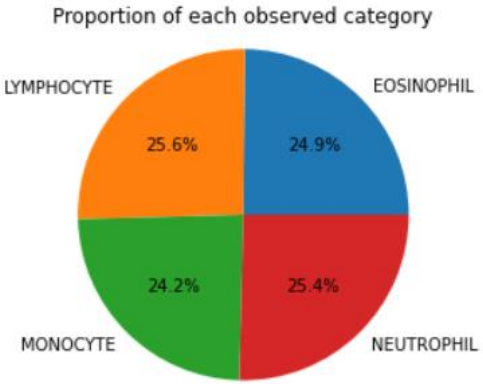
- Import necessary packages and libraries
- Load data
  1. Write some helper functions (getImgContours, getBoxes, findEdges)
  2. We read the file of data
  3. Iterate through training and test sets
  4. Iterate through folders in each dataset
  5. Iterate through each image in folder
  6. Get pathname of each image
  7. Add padding to the image to better detect cell at the edge
  8. Threshold the image to get the target cell
  9. Opening erosion then dilation
  10. Detecting the blood cell
  11. Get the large box and get its coordinate.
  12. Draw the contour and fill it
  13. Extract the blood cell
  14. Resize the image
  15. Append the image and its corresponding label to the output
  16. Return images and its labels



- We had split the data into train, validation, and test data by ratio 80%, 10%, 10% respectively.



- Data exploration:



Training the model:

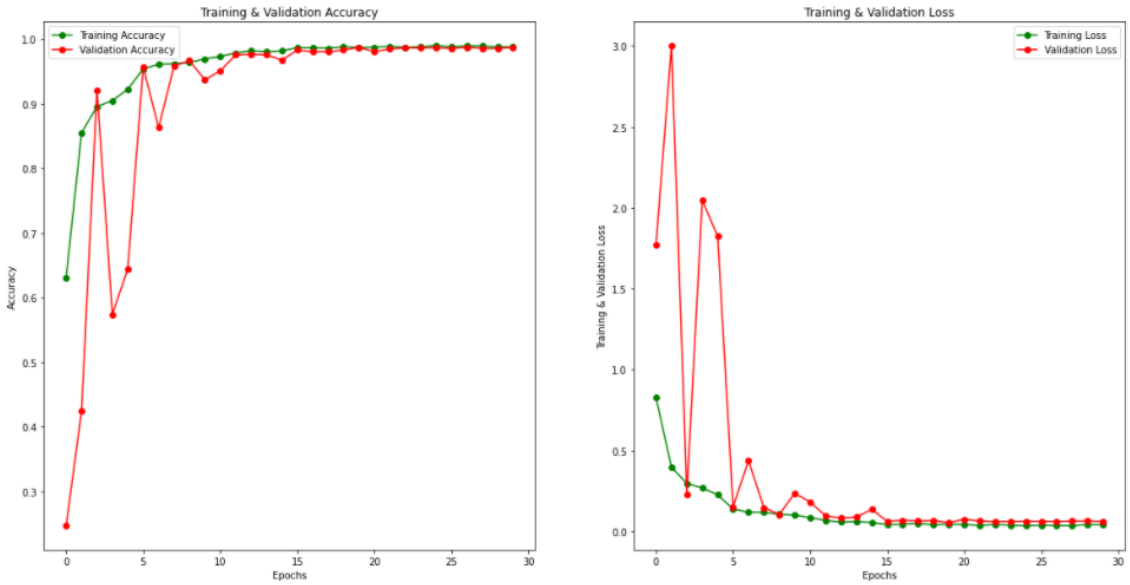
1. **CNN:** This model contains a sequence of five Conv blocks containing combinations of SeparableConv2D, Batch Normalization, MaxPooling and Dropout layers. The output of the final Conv block is flattened and followed by three Fully Connected (FC) layers each with its own Dropout layer. A final FC layer is added with four units and a softmax activation for multiclass classification.
2. **SVC:**

   SVM is a supervised machine learning algorithm which can be used for classification problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. Simply put, it does some extremely complex data transformations, then figures out how to separate your data based on the labels or outputs you've defined.
3. **Random Forest:**

   We had used Scikit-Learn package. Random forest is a type of supervised machine learning algorithm based on ensemble learning. We pick N random records from the dataset. Then, we build a decision tree based on these N records. After that, We Choose the number of trees you want in your algorithm and repeat the previous two steps.

Results:

CNN



```
38/38 [==============================] - 0s 7ms/step - loss: 0.0602 - accuracy: 0.9802
Loss of the model  is - test  0.06021144613623619
Accuracy of the model is - test 98.0246901512146 %
38/38 [==============================] - 0s 7ms/step - loss: 0.0620 - accuracy: 0.9868
Loss of the model  is - val  0.06197246536612511
Accuracy of the model is - val 98.68312478065491 %
304/304 [==============================] - 2s 7ms/step - loss: 0.0155 - accuracy: 0.995
6
Loss of the model  is - train  0.015450972132384777
Accuracy of the model is - train 99.55742955207825 %
```

Random Forest:

Getting Accuracy Score

```
from sklearn.metrics import accuracy_score
print("Test accuracy",accuracy_score(test_labels, pred))
print("Train accuracy",accuracy_score(train_labels,pred2))
```

```
Test accuracy 0.9209551255660766
Train accuracy 1.0
```

SVC:

```
print("test_score",clf1.score(test_images,test_labels))
print("train_score",clf1.score(train_images,train_labels))
```

```
test_score 0.7366255144032922
train_score 0.9976327706875258
```

Why we prefer the CNN model:

A CNN-based framework is built to automatically classify the blood cell images into subtypes the cells. Experiments are conducted on a dataset of 13k images of blood cells with their subtypes, and the results show that the CNN model provides better results in terms of evaluation parameters. When we took a look at the results that we had got, we found that the CNN model provides us with the best accuracy.

Problems we had faced and how we overcome:

We face the overfitting problem and feedback from users on other Kaggle notebooks suggesting poor data shuffling, so we decided to combine the Train/Test folders and recreate the dataset using our own shuffling and splits.