Digital Image Processing

From

Image processing

To

Deep-learning using CNN

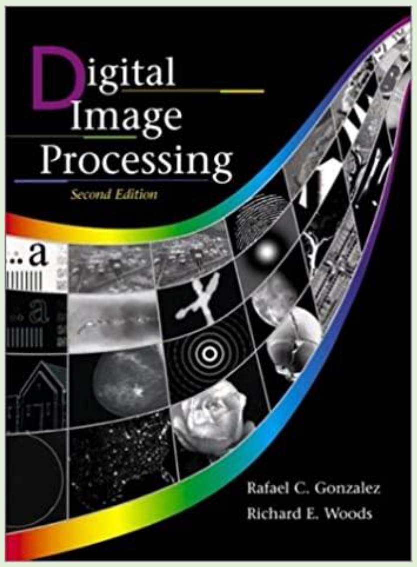


IMAGE ➔ PROCESSING ➔ ANALSYSIS ➔ RECOGNIZNG

AASTMT
Smart Village

Presented by: Mohamed Hazem **Supervised by: Engy Ehab**

Acknowledgement

I would like to thank our advisor Assoc. Prof. Dr Aliaa Youssif who really directed me to produce this project. I would like to give special thanks to our Professor for:

- Providing me with the needed technical information throughout the project.

- Not only the technical information but also for the soft skills; by giving me the chance to present every step of project in a formal way.

- Giving me the chance to search for the needed information in design.

Nonetheless, I would also thank Engineer Engy Ehab who gives me much knowledge in

- CNN (Convolutional neural network).

- Deep-learning as all

- Python

- Techniques

Our Project Split to 3 Phases 2 of them Image Processing and Last one using CNN Deep-Learning.

No worry everything will be present clearly in Table of Contents

Table of Contents

Table of Contents

Chapter One 3

Analysis of MRI Images Using Image Processing Technique 3

Example ONE 3

Example Two 4

Chapter Two 5

Analysis of Breast Cancer Images Using Image Processing Technique 5

Chapter Three & Last 8

White Blood cells Classification using CNN Deep learning & TensorFlow & Keras & Real-life Dataset 8

Chapter One

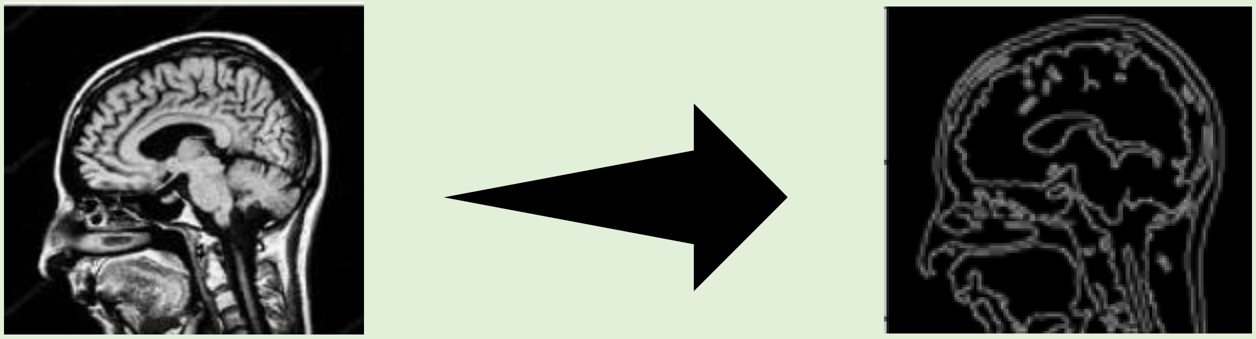
Analysis of MRI Images Using Image Processing Technique

Example ONE

So, here's our image and we going to use some process on it to get it out

Original Image

Target Image



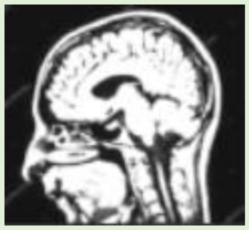
Let's go ahead and see what need to happen

Process & Pipeline:

Step 1 Contrast:

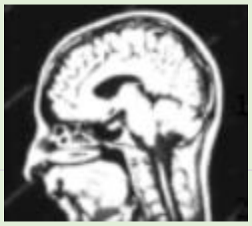
```
imgcon=cv2.addWeighted(imgabs,alpha,np.zeros(imgabs.shape, imgabs.dtype),0,beta)
```

Note (Alpha control in Degree of Contrast)



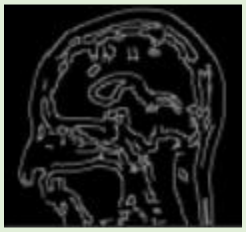
Step 2 Blurring:

```
imggau2 = cv2.GaussianBlur(imgcon,(3,3),sigmaX=1,borderType=cv2.BORDER_CONSTANT)
```



Step 3 Edge Detection using Canny:

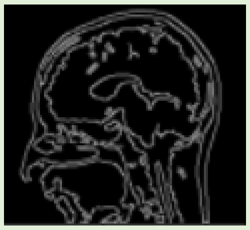
```
imgcanny= cv2.Canny(imggau2,300,800)
```



Step 4 Repeat Step 3 with different values:

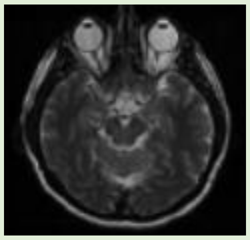
After applying many of canny but with different values we got our Target Finally

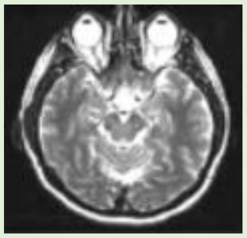
As you can See ➔

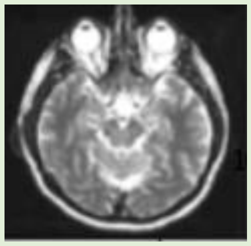


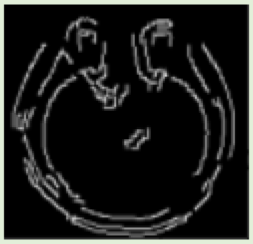
Example Two

We know What happens from Example one so we will move Quickly on this one 😊

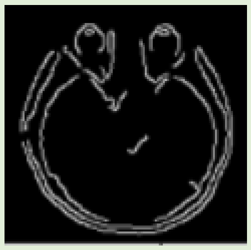






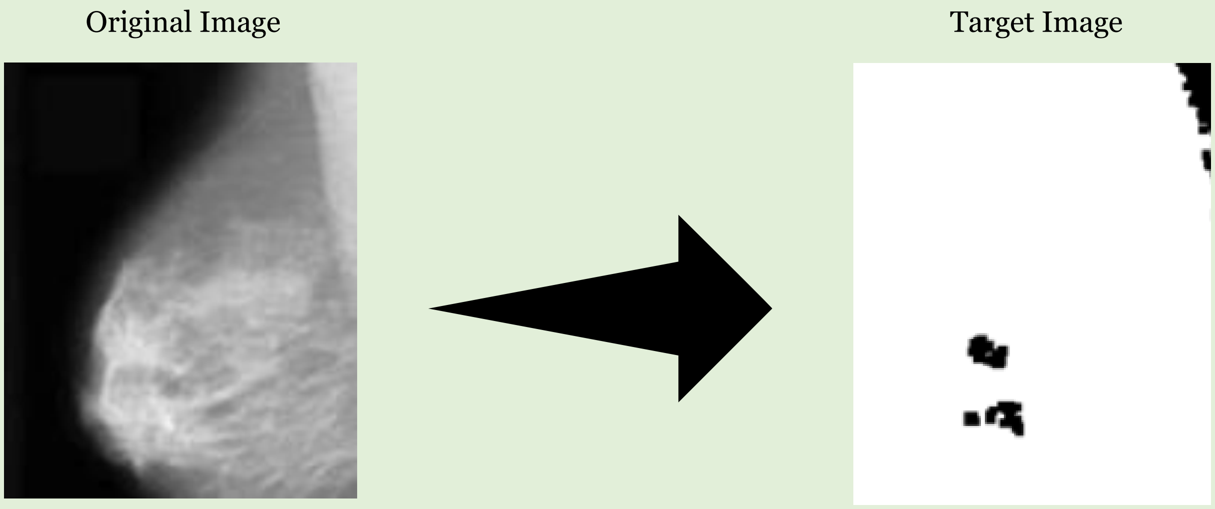


Final Output



Chapter Two

Analysis of Breast Cancer Images Using Image Processing Technique

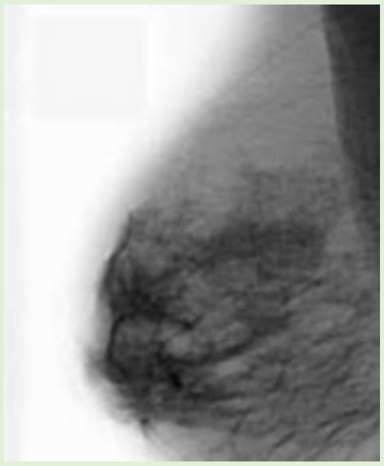


At Target Image you can see the cancer clearly after Applying some Image Processing Techniques

Process & Pipeline:

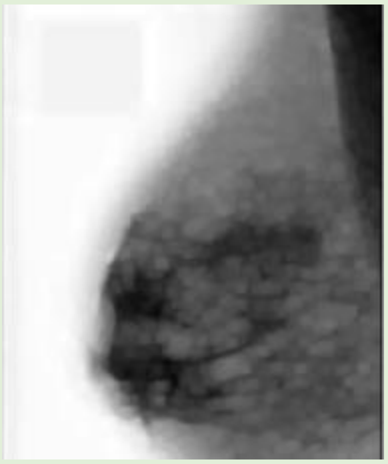
Step 1 Negative:

```
imgneg = 255-img
```



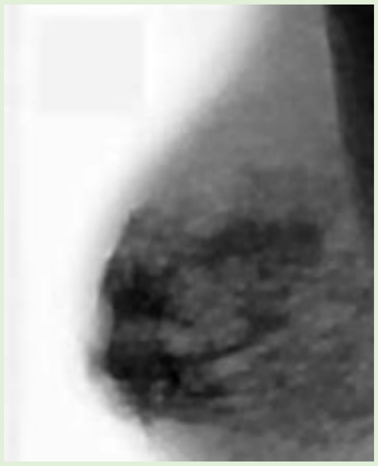
Step 2 Dilation:

```
imgdil = cv2.dilate(imgneg, kernel, iterations=1)
```



Step 3 erosion:

```
imgero = cv2.erode(imgdil, kernel, iterations=1)
```



Step 4 Threshold:

```
imgthreshold = cv2.threshold(imgero,50,90,cv2.THRESH_BINARY)
```

Here at Threshold, you can see Cancer cells clearly in the breast

Chapter Three & Last

White Blood cells Classification using CNN Deep learning & TensorFlow & Keras & Real-life Dataset

First, Let's Define our Dataset

It's 9956 Real images for white Blood Cells

(EOSINOPHIL, LYMPHOCYTE, MONOCYTE, NEUTROPHIL)

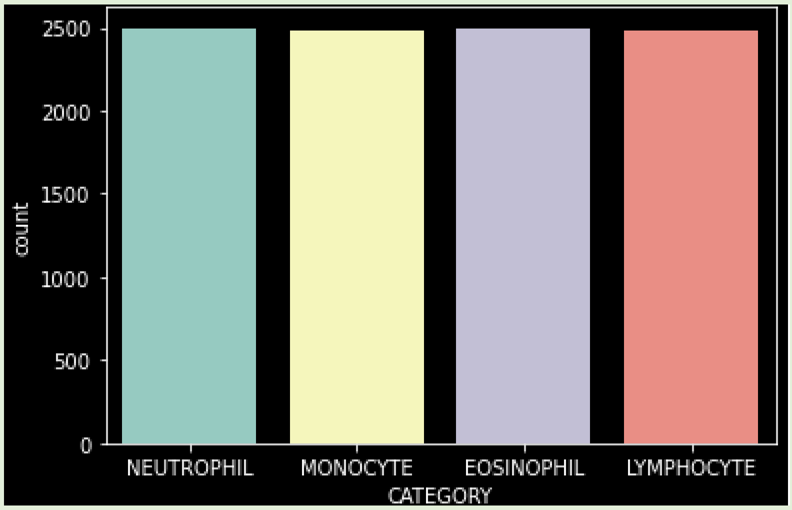
We now will pick some of it to train and other to test and also a few as validation

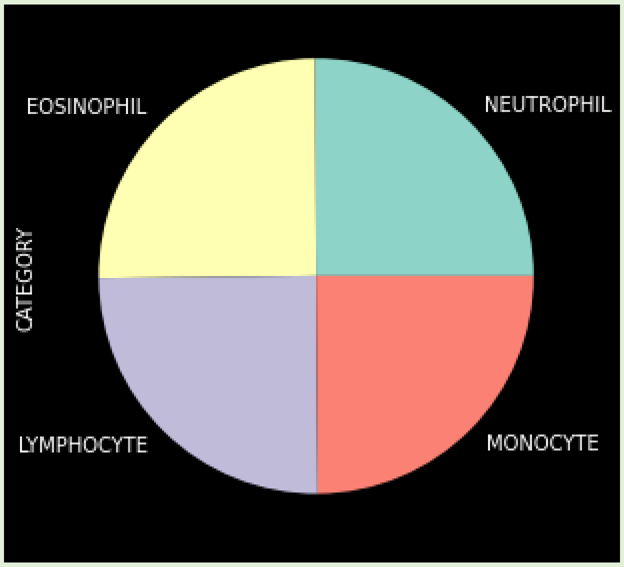
Train

```
Train_Data_Path = Path("C:/Users/moham/Downloads/dataset2-master/dataset2-master/images/TRAIN")
```

```
print("EOSINOPHIL: ", Train_JPG_Labels.count("EOSINOPHIL"))
print("LYMPHOCYTE: ", Train_JPG_Labels.count("LYMPHOCYTE"))
print("MONOCYTE: ", Train_JPG_Labels.count("MONOCYTE"))
print("NEUTROPHIL: ", Train_JPG_Labels.count("NEUTROPHIL"))
```

```
EOSINOPHIL:  2497
LYMPHOCYTE:  2483
MONOCYTE:  2478
NEUTROPHIL:  2499
```



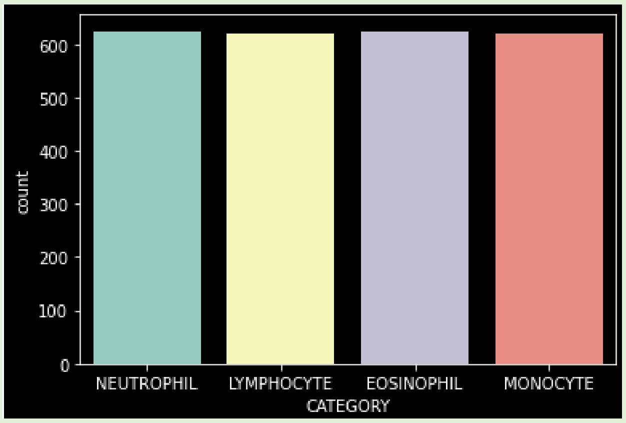


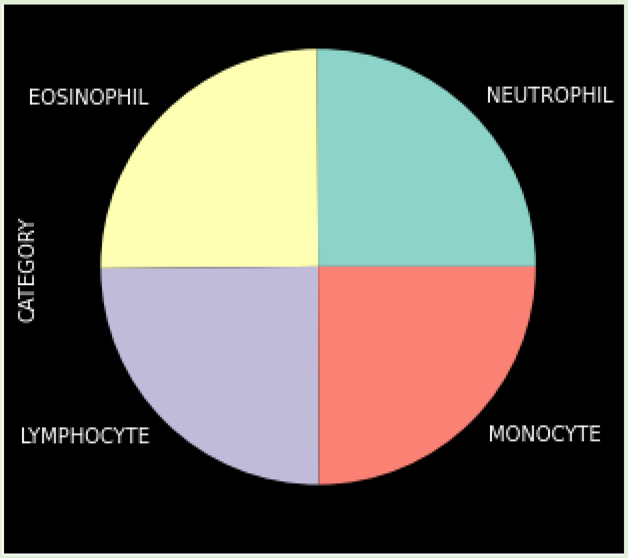
Test

```
Test_Data_Path = Path("C:/Users/moham/Downloads/dataset2-master/dataset2-master/images/TEST")
```

```
print("EOSINOPHIL: ", Test_JPG_Labels.count("EOSINOPHIL"))
print("LYMPHOCYTE: ", Test_JPG_Labels.count("LYMPHOCYTE"))
print("MONOCYTE: ", Test_JPG_Labels.count("MONOCYTE"))
print("NEUTROPHIL: ", Test_JPG_Labels.count("NEUTROPHIL"))
```

```
EOSINOPHIL:  623
LYMPHOCYTE:  620
MONOCYTE:  620
NEUTROPHIL:  624
```



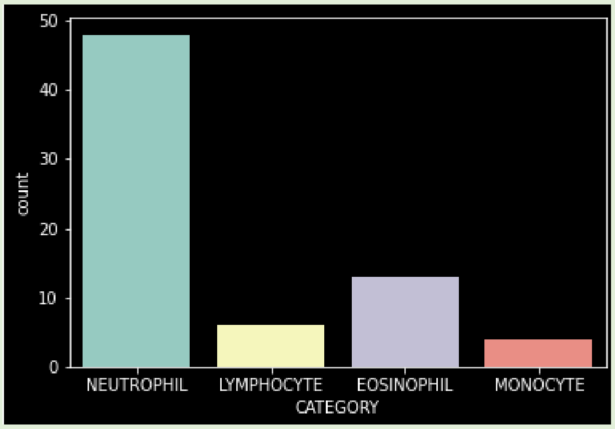


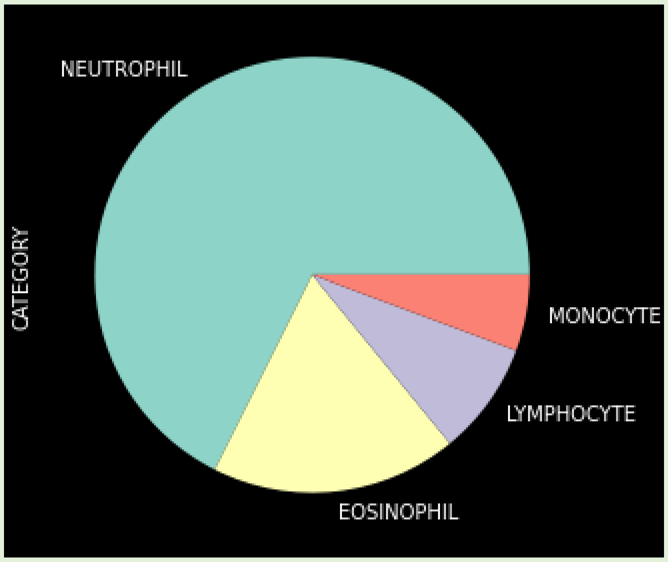
Validation

```
Validation_Data_Path = Path("C:/Users/moham/Downloads/dataset2-master/dataset2-master/images/TEST_SIMPLE")
```

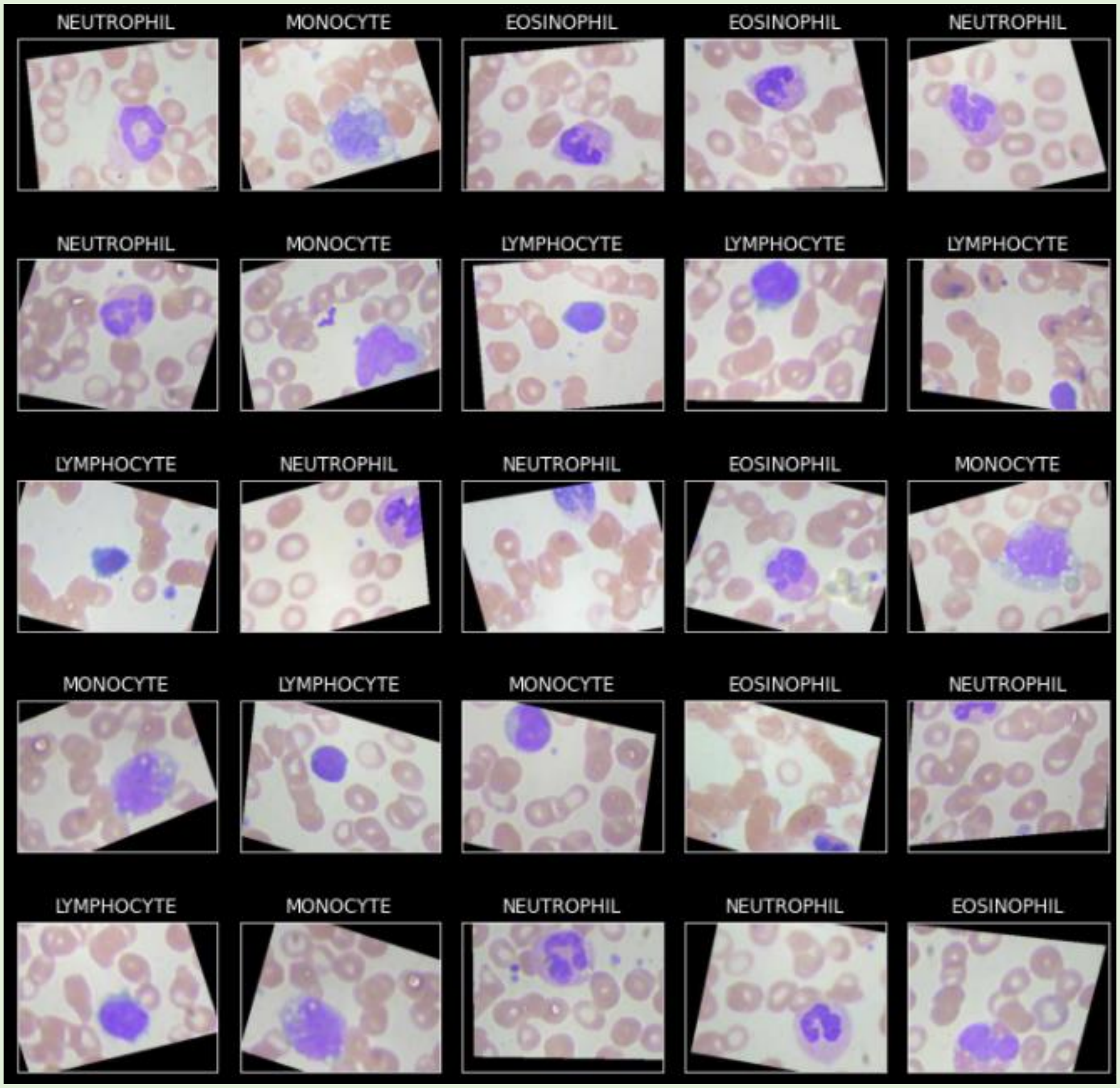
```
print("EOSINOPHIL: ", Validation_JPG_Labels.count("EOSINOPHIL"))
print("LYMPHOCYTE: ", Validation_JPG_Labels.count("LYMPHOCYTE"))
print("MONOCYTE: ", Validation_JPG_Labels.count("MONOCYTE"))
print("NEUTROPHIL: ", Validation_JPG_Labels.count("NEUTROPHIL"))
```

```
EOSINOPHIL:  13
LYMPHOCYTE:  6
MONOCYTE:  4
NEUTROPHIL:  48
```

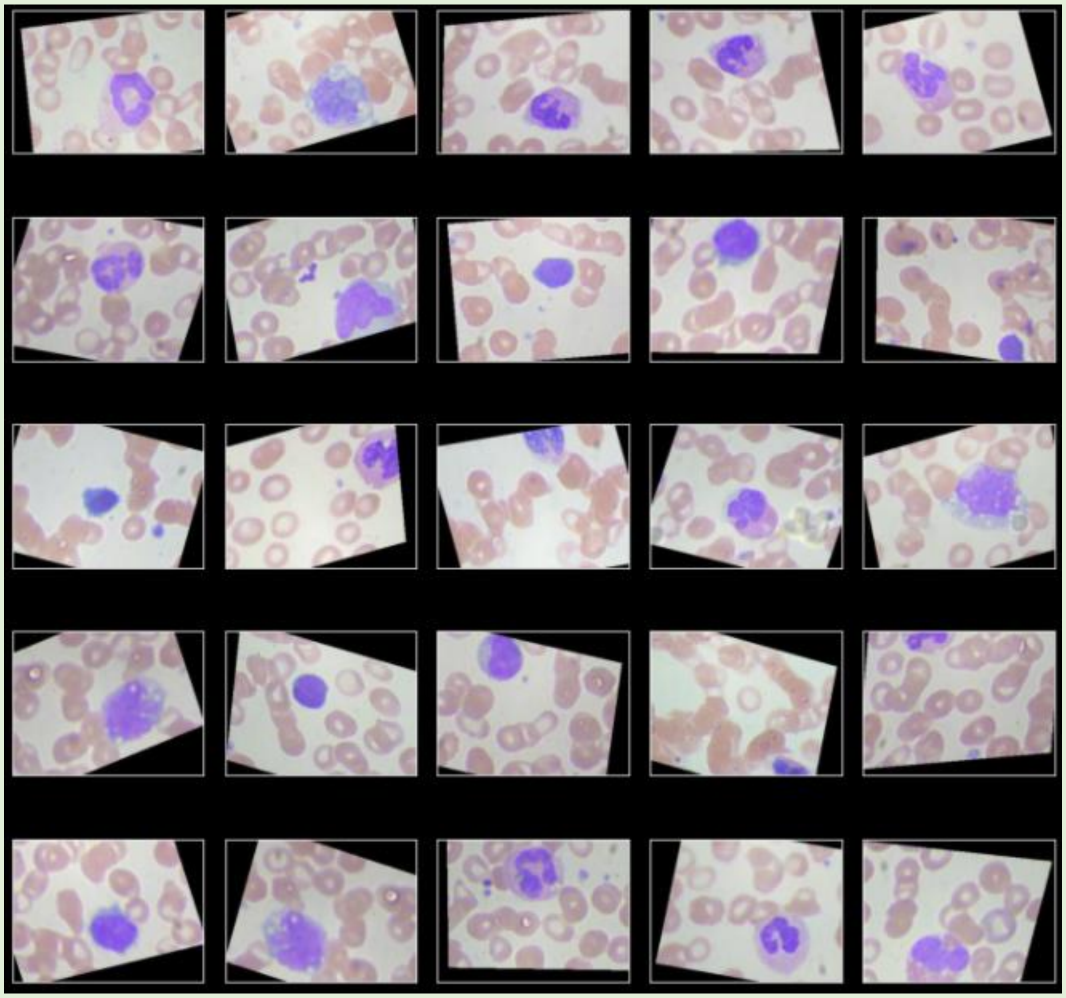




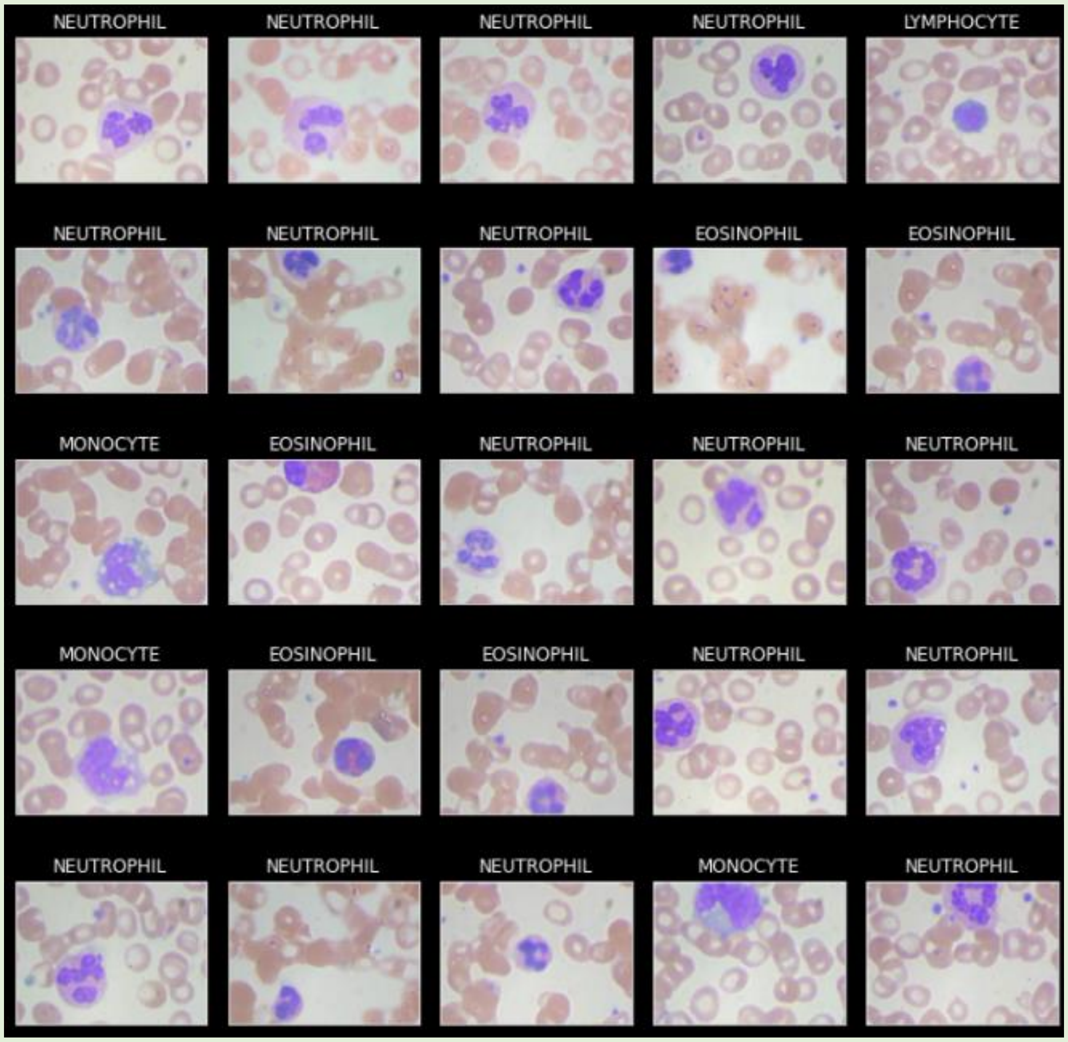
Let's take a look on Our images that we will train



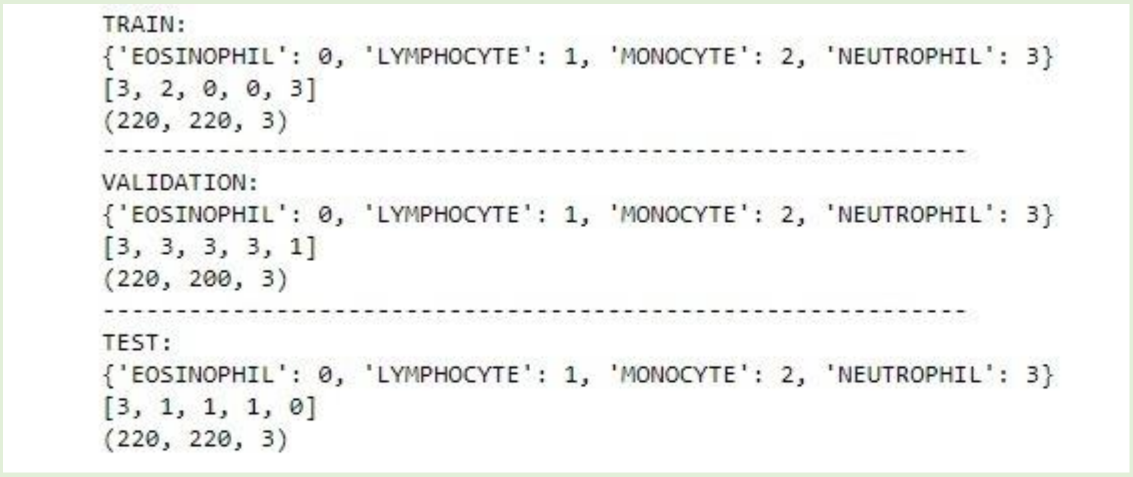
And Our images that we will test



Here also Our Validation images



Now we will class Our Data as numbers



```
TRAIN:
{'EOSINOPHIL': 0, 'LYMPHOCYTE': 1, 'MONOCYTE': 2, 'NEUTROPHIL': 3}
[3, 2, 0, 0, 3]
(220, 220, 3)
---------------------------------------------------------
VALIDATION:
{'EOSINOPHIL': 0, 'LYMPHOCYTE': 1, 'MONOCYTE': 2, 'NEUTROPHIL': 3}
[3, 3, 3, 3, 1]
(220, 200, 3)
---------------------------------------------------------
TEST:
{'EOSINOPHIL': 0, 'LYMPHOCYTE': 1, 'MONOCYTE': 2, 'NEUTROPHIL': 3}
[3, 1, 1, 1, 0]
(220, 220, 3)
```

So, we can train it and get result as Predictions of (0,1,2,3)

Though…

Things will Happens at next pages

Let's the Deep-learning Start

Activation of CNN

CNN STRUCTURE WITH SeparableConv2D

```
Model = Sequential()

Model.add(SeparableConv2D(32,3,
                          activation="relu",
                 input_shape=(220,220,3)))
Model.add(BatchNormalization())
Model.add(MaxPooling2D((2)))

#
Model.add(SeparableConv2D(64,3,
                 activation="relu"))
Model.add(SeparableConv2D(128,(3,3),
                 activation="relu"))
Model.add(Dropout(0.5))
Model.add(MaxPooling2D((2)))

#
Model.add(SeparableConv2D(64,3,
                 activation="relu"))
Model.add(SeparableConv2D(128,3,
                 activation="relu"))
Model.add(Dropout(0.5))
Model.add(GlobalAveragePooling2D())

#
Model.add(Flatten())
Model.add(Dense(256,
                activation="relu"))
Model.add(Dropout(0.5))
Model.add(Dense(4,
                activation="softmax"))
```

Then we add our TensorFlow Keras

```
Call_Back = tf.keras.callbacks.EarlyStopping(monitor="val_accuracy",patience=5,mode="max")
```

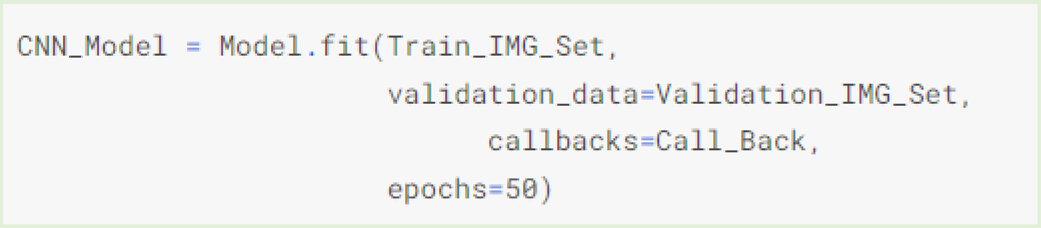
And here just to define loss & accuracy print in our cmd

```
Model.compile(optimizer="rmsprop",loss="categorical_crossentropy",metrics=["accuracy"])
```

So, Let's train it up

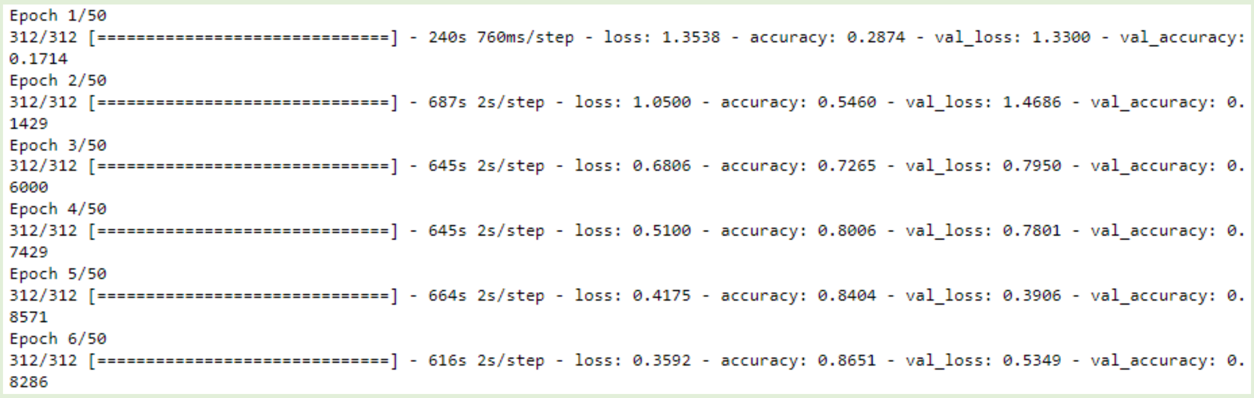
Normal we use between 10 epochs to 28 epochs

But to get best result & accuracy we will go for 50 epochs



```
CNN_Model = Model.fit(Train_IMG_Set,
                       validation_data=Validation_IMG_Set,
                              callbacks=Call_Back,
                       epochs=50)
```

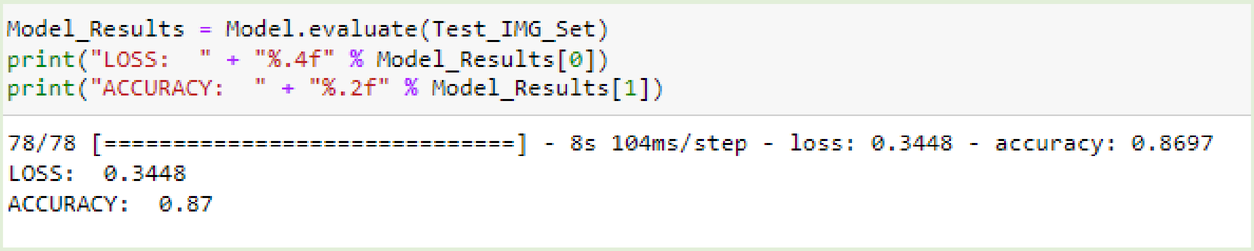
Small look on training...



```
Epoch 1/50
312/312 [==============================] - 240s 760ms/step - loss: 1.3538 - accuracy: 0.2874 - val_loss: 1.3300 - val_accuracy:
0.1714
Epoch 2/50
312/312 [==============================] - 687s 2s/step - loss: 1.0500 - accuracy: 0.5460 - val_loss: 1.4686 - val_accuracy: 0.
1429
Epoch 3/50
312/312 [==============================] - 645s 2s/step - loss: 0.6806 - accuracy: 0.7265 - val_loss: 0.7950 - val_accuracy: 0.
6000
Epoch 4/50
312/312 [==============================] - 645s 2s/step - loss: 0.5100 - accuracy: 0.8006 - val_loss: 0.7801 - val_accuracy: 0.
7429
Epoch 5/50
312/312 [==============================] - 664s 2s/step - loss: 0.4175 - accuracy: 0.8404 - val_loss: 0.3906 - val_accuracy: 0.
8571
Epoch 6/50
312/312 [==============================] - 616s 2s/step - loss: 0.3592 - accuracy: 0.8651 - val_loss: 0.5349 - val_accuracy: 0.
8286
```

So, after 8 hours of training, we finally back.........

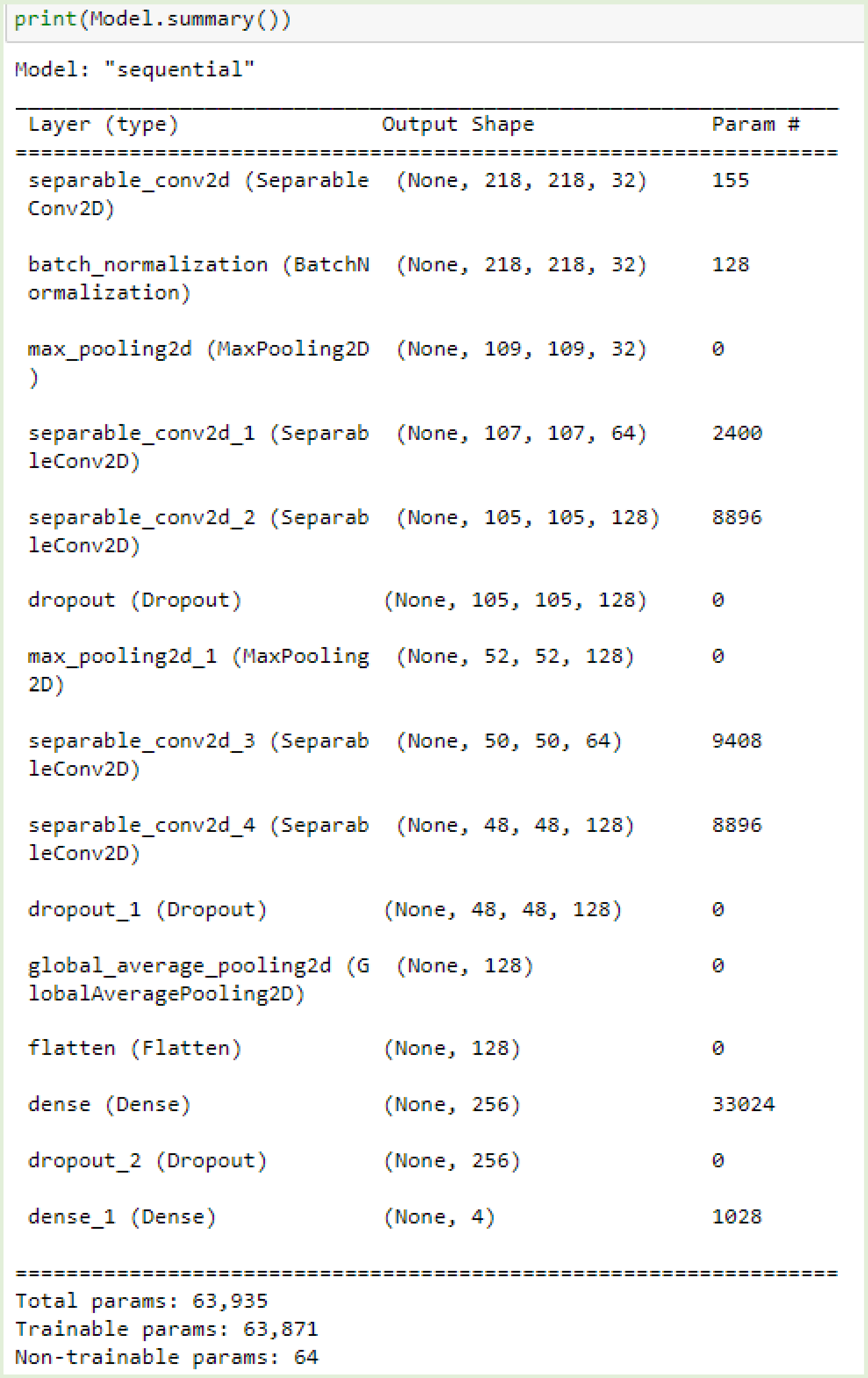
Let's check accuracy of our model now



```
Model_Results = Model.evaluate(Test_IMG_Set)
print("LOSS:  " + "%.4f" % Model_Results[0])
print("ACCURACY:  " + "%.2f" % Model_Results[1])
```

```
78/78 [==============================] - 8s 104ms/step - loss: 0.3448 - accuracy: 0.8697
LOSS:  0.3448
ACCURACY:  0.87
```

And here's model summary if anyone interest



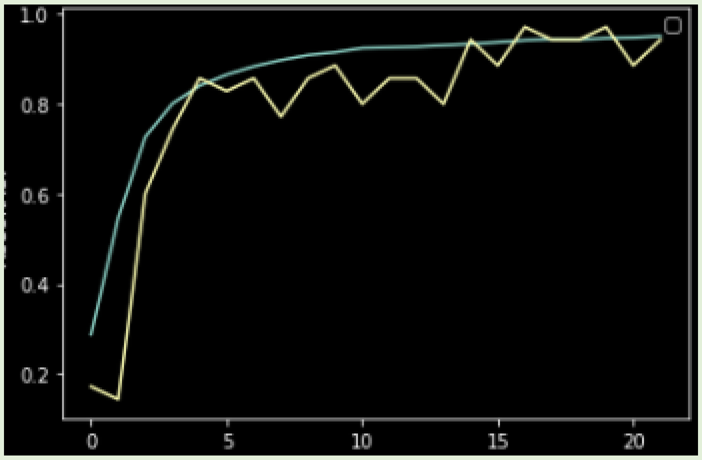
```
print(Model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 separable_conv2d (Separable  (None, 218, 218, 32)     155
 Conv2D)

 batch_normalization (BatchN  (None, 218, 218, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 109, 109, 32)     0
 )

 separable_conv2d_1 (Separab  (None, 107, 107, 64)     2400
 leConv2D)

 separable_conv2d_2 (Separab  (None, 105, 105, 128)    8896
 leConv2D)

 dropout (Dropout)           (None, 105, 105, 128)     0

 max_pooling2d_1 (MaxPooling  (None, 52, 52, 128)      0
 2D)

 separable_conv2d_3 (Separab  (None, 50, 50, 64)       9408
 leConv2D)

 separable_conv2d_4 (Separab  (None, 48, 48, 128)      8896
 leConv2D)

 dropout_1 (Dropout)         (None, 48, 48, 128)       0

 global_average_pooling2d (G  (None, 128)              0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 256)               33024

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 4)                 1028

=================================================================
Total params: 63,935
Trainable params: 63,871
Non-trainable params: 64
```

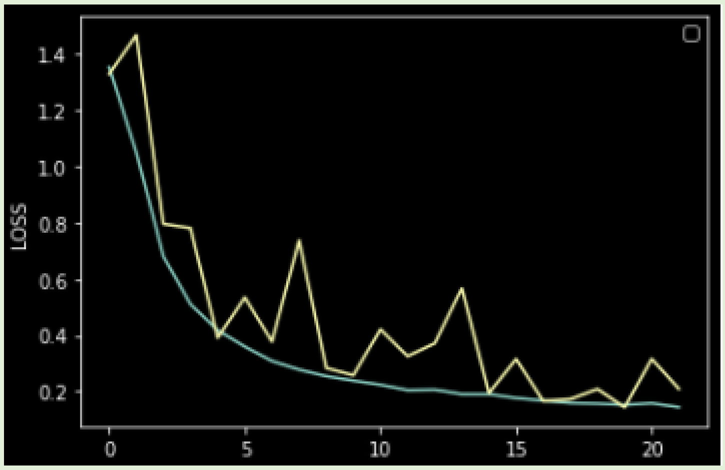
So, let's start now with some of CNN Benefits

Note (we got best result at 23 epochs so it didn't continue)

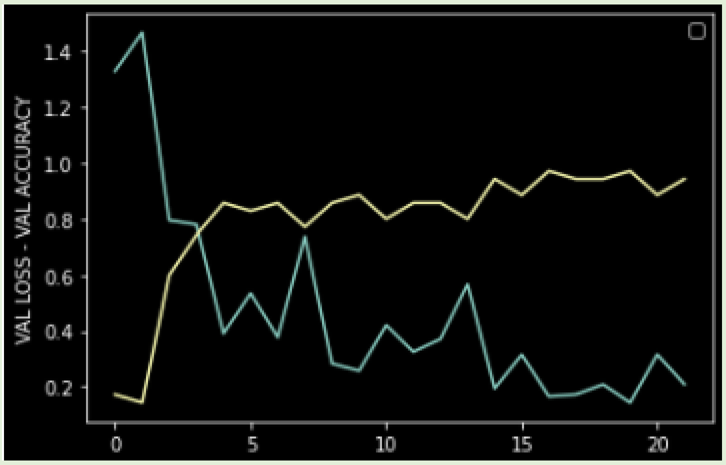
Accuracy Plot



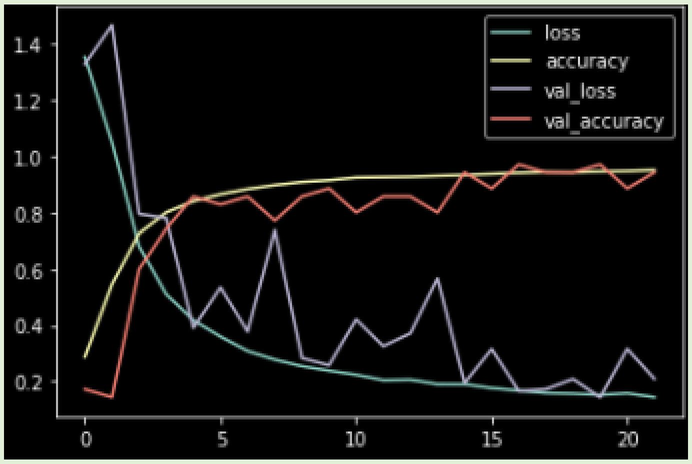
Loss Plot



Val Acc vs Val Loss

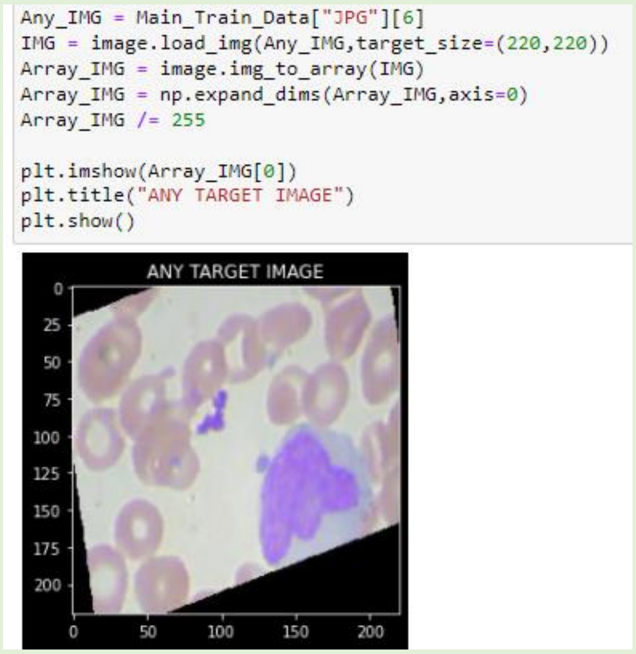


And here's everything



So, let's go ahead and have some Image Processing

Let's Pick a random Image and Play with it

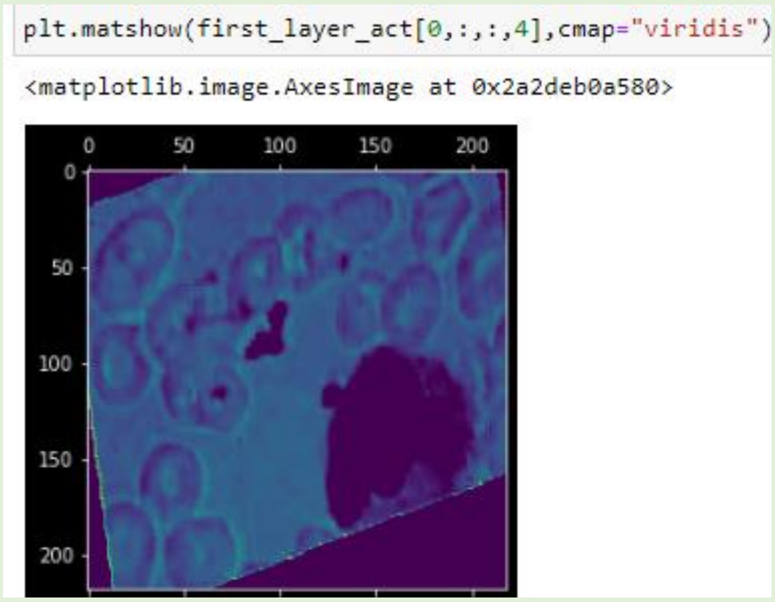


```
Any_IMG = Main_Train_Data["JPG"][6]
IMG = image.load_img(Any_IMG,target_size=(220,220))
Array_IMG = image.img_to_array(IMG)
Array_IMG = np.expand_dims(Array_IMG,axis=0)
Array_IMG /= 255

plt.imshow(Array_IMG[0])
plt.title("ANY TARGET IMAGE")
plt.show()
```

Apply some layers?

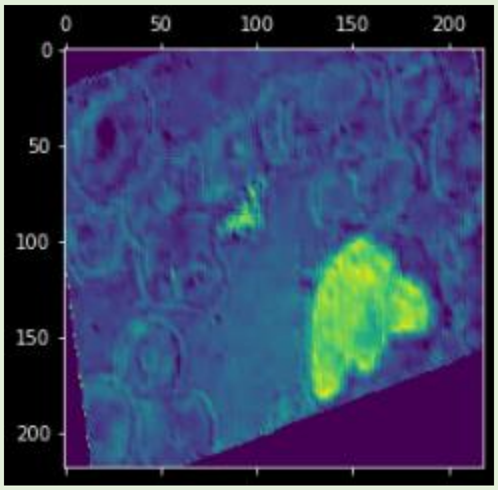
We can apply 0-15 different layer

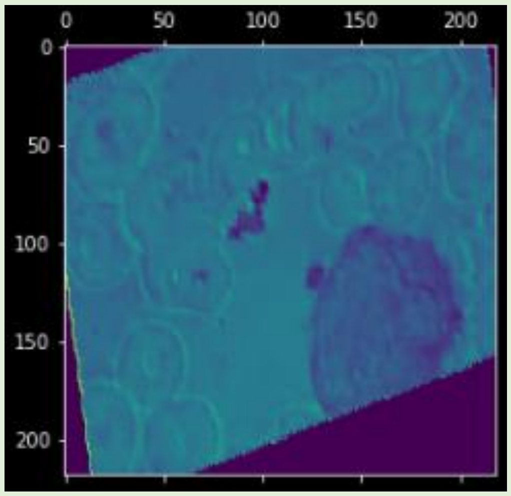
Here's some

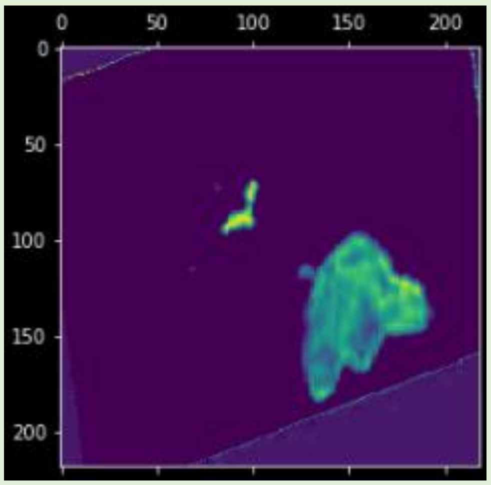


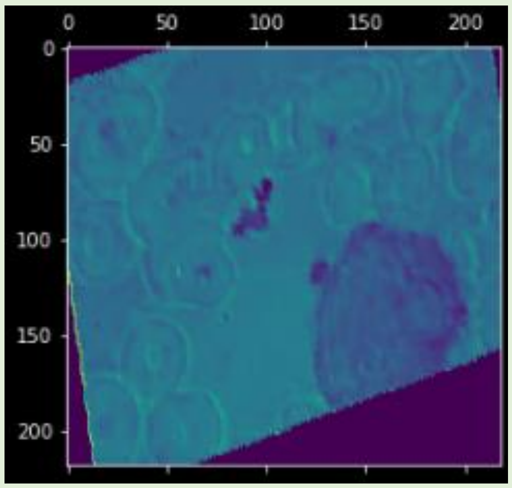
```
plt.matshow(first_layer_act[0,:,:,4],cmap="viridis")
```

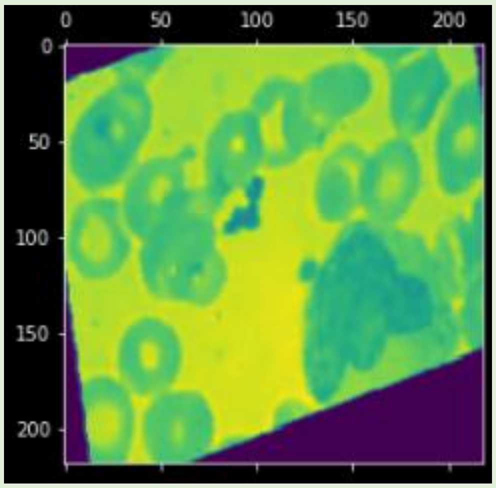
```
<matplotlib.image.AxesImage at 0x2a2deb0a580>
```

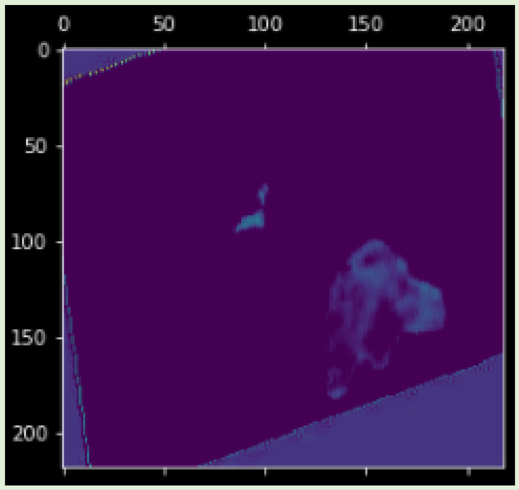


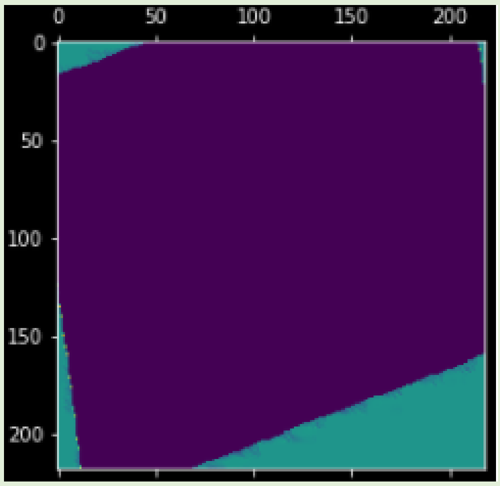


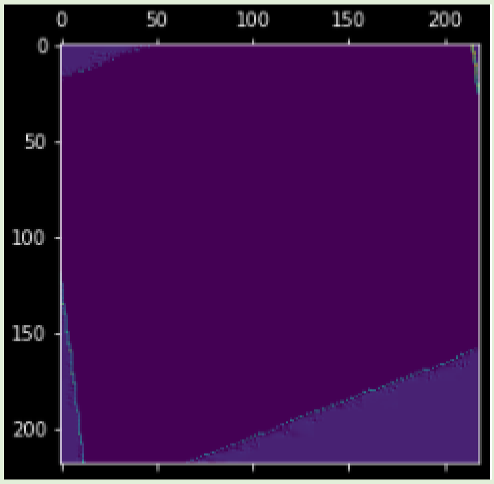


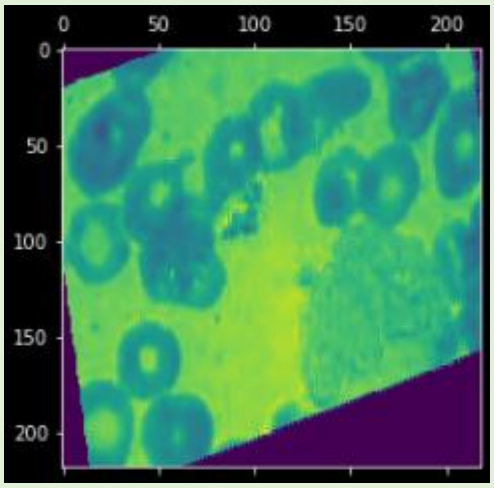


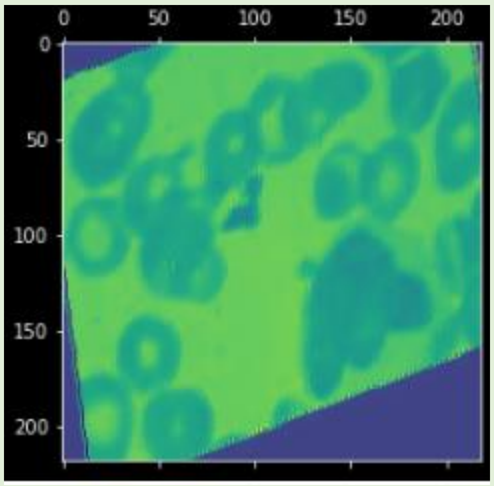










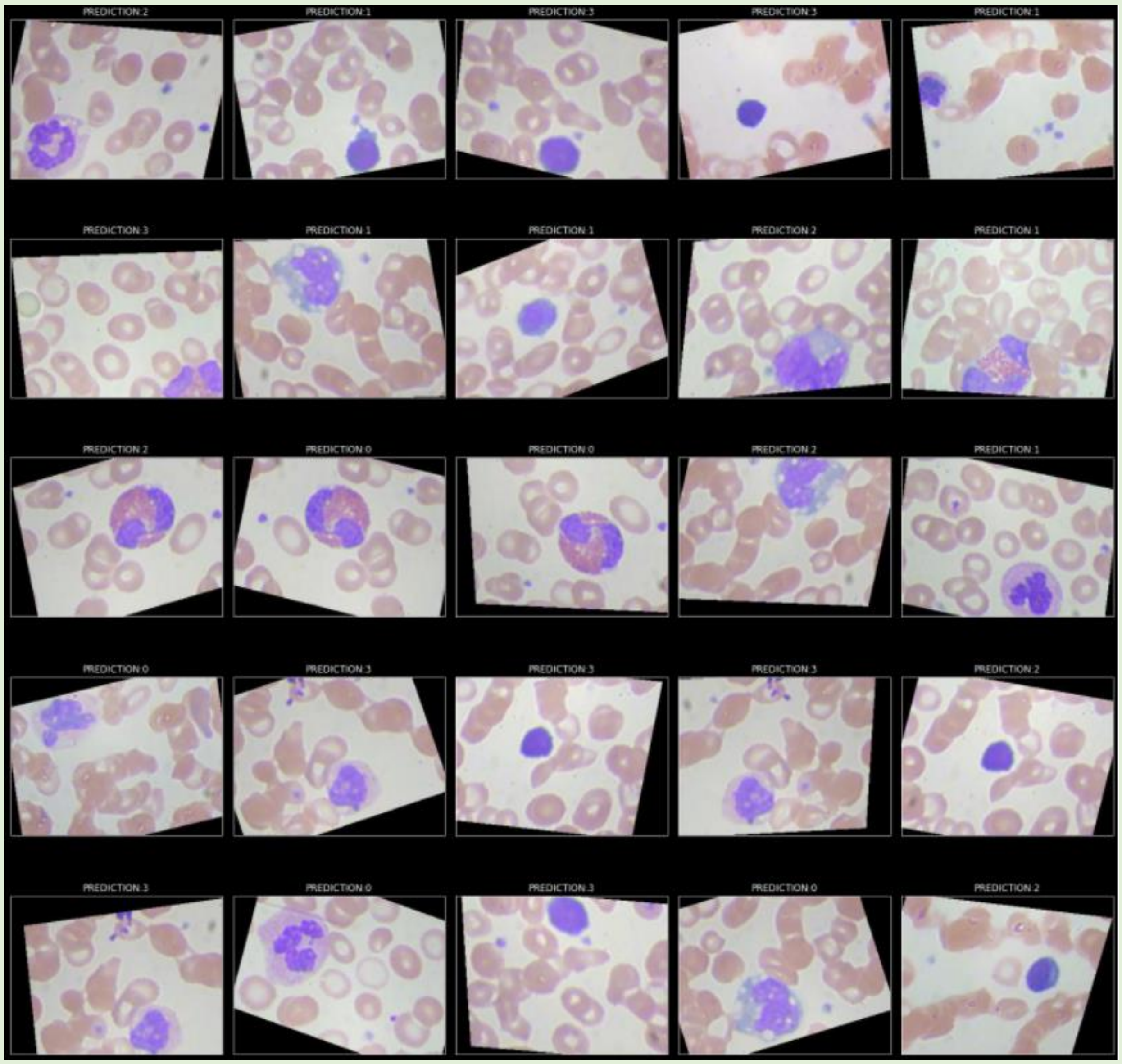


So, we Almost at the End ☹

If you remember we Classed our White Blood Cells as 0,1,2,3

So, here's the Predication & Classification now

(EOSINOPHIL 0, LYMPHOCYTE 1, MONOCYTE 2, NEUTROPHIL 3)



Thank You!