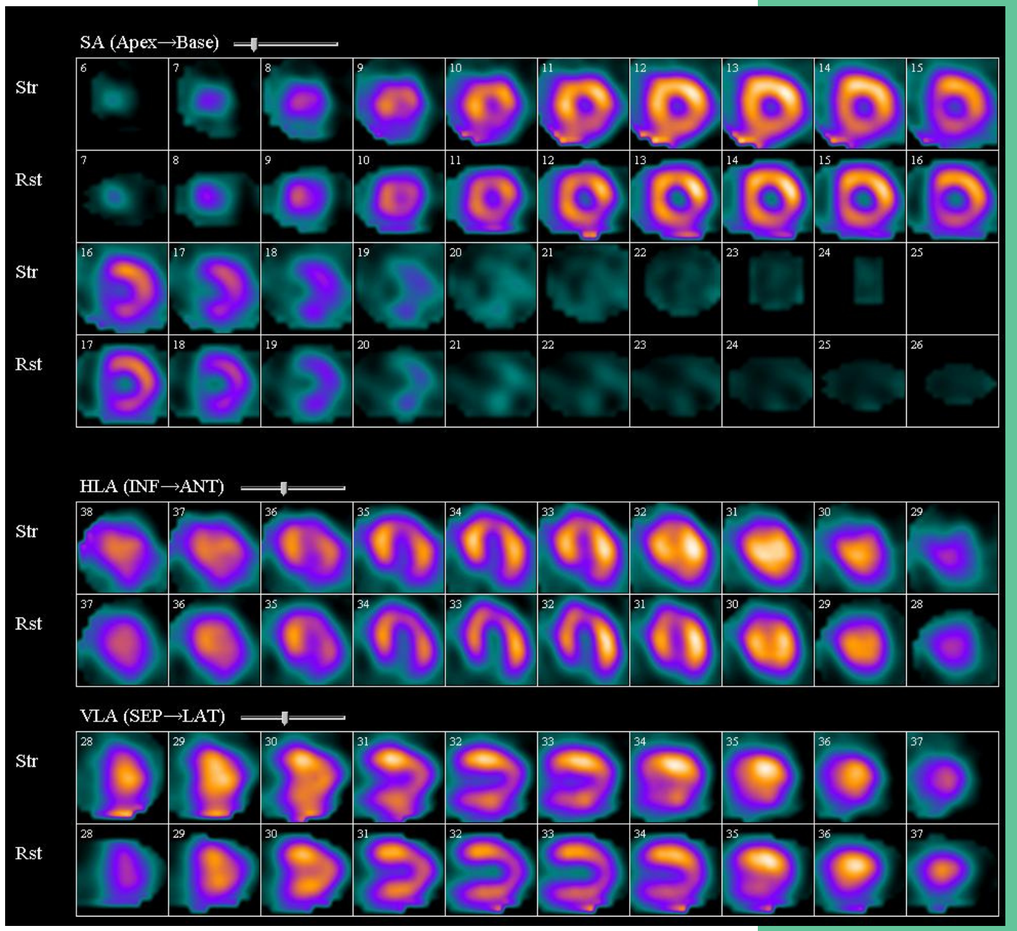
CARDIAC

SPECT image Classification By ML.

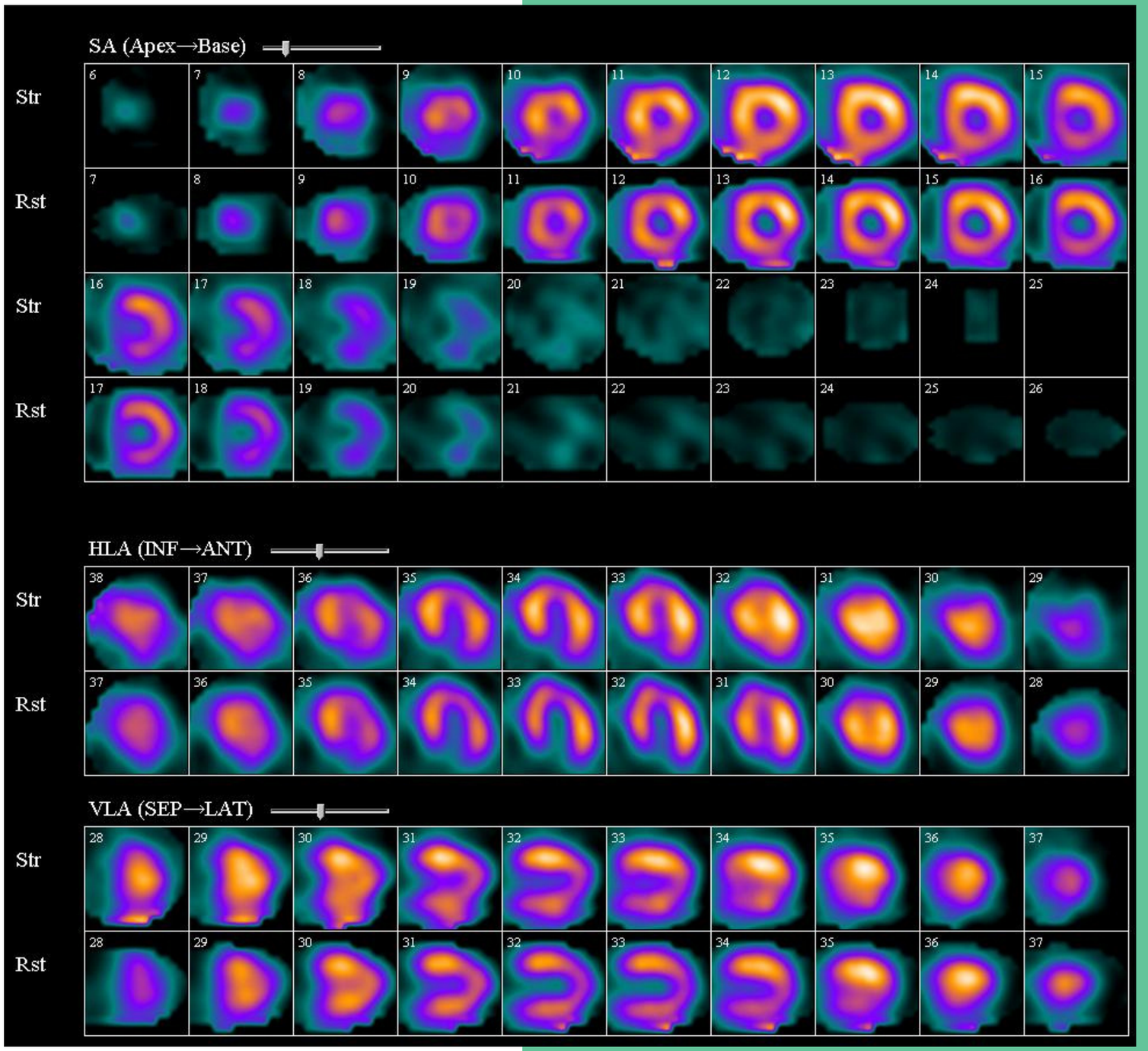


BY:
M.HASEEB
M.SHAHEER
M.ZAID

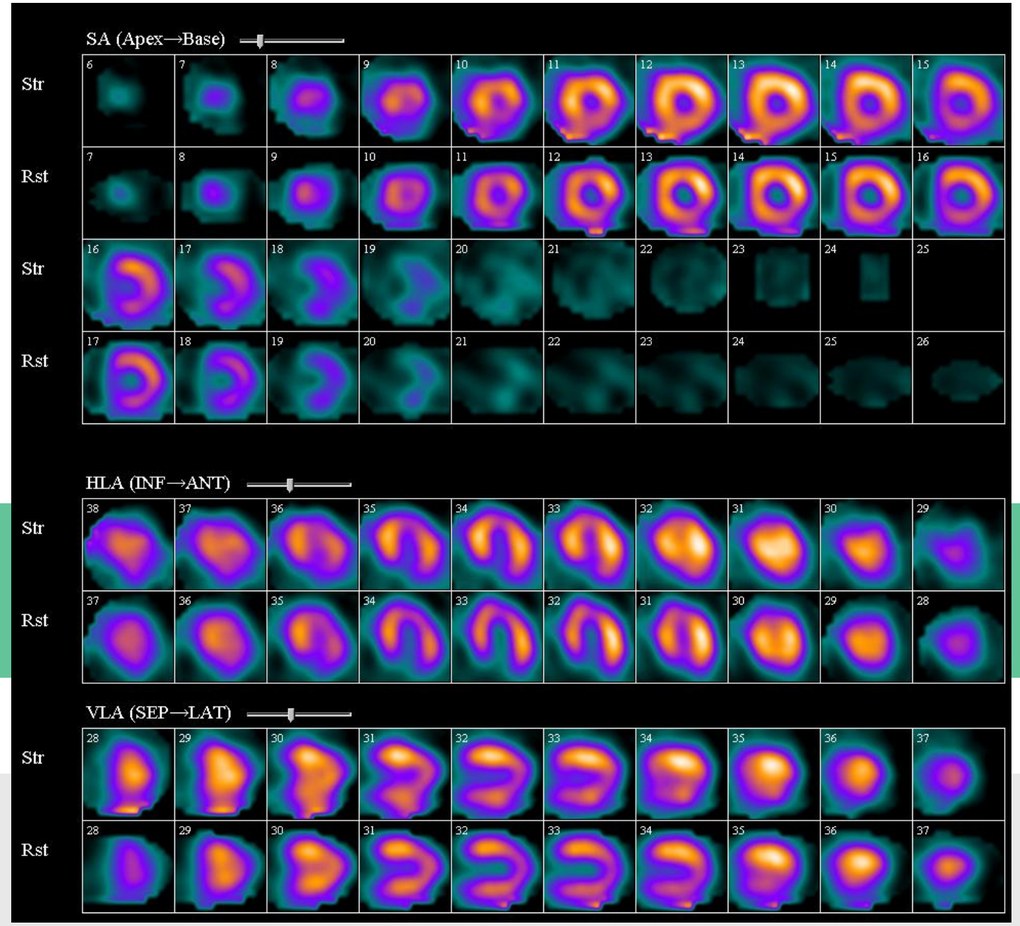
We Perform 3 Tasks

- Dataset Generation
- Apply CNN
- Apply GoogleNet V3

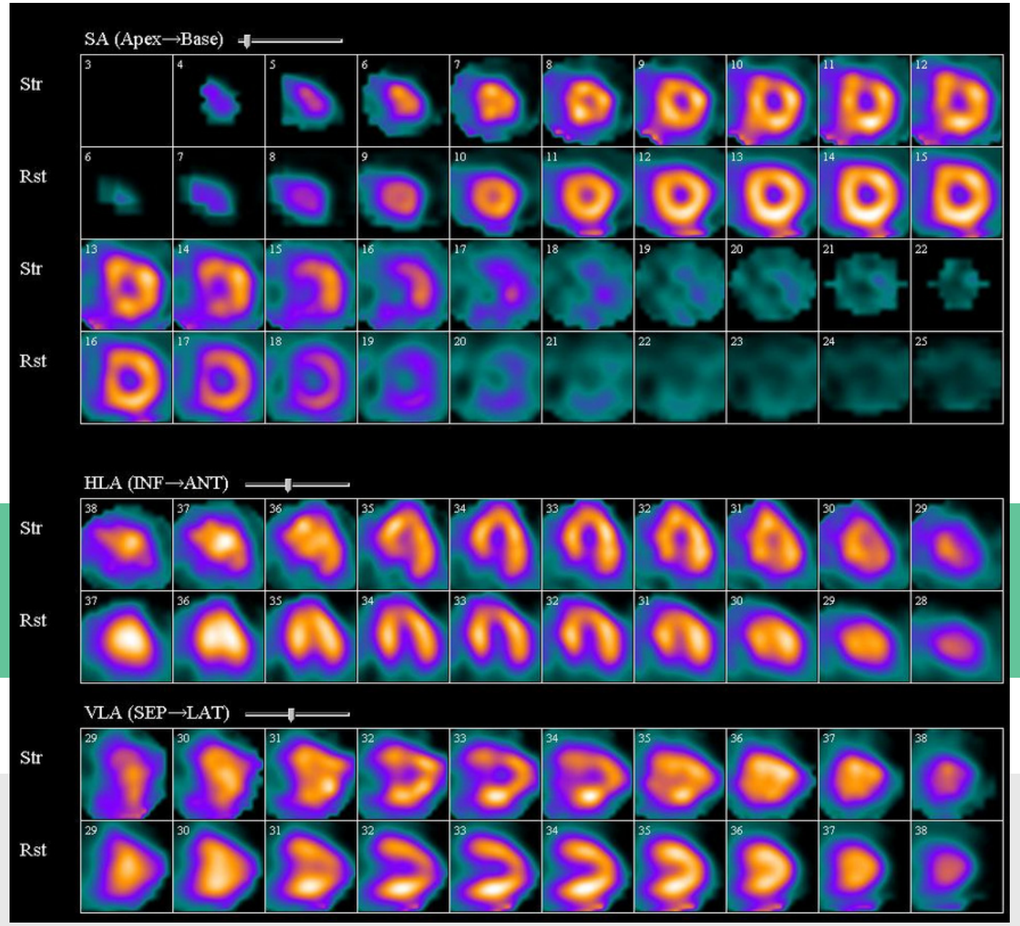
Dataset and it’s Difficulties.



NORMAL

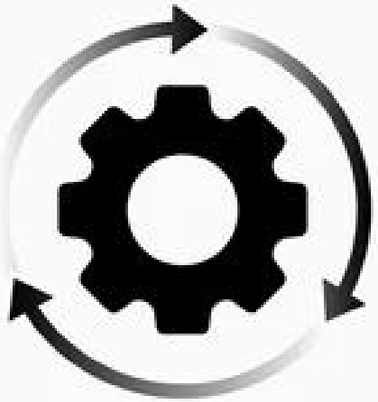


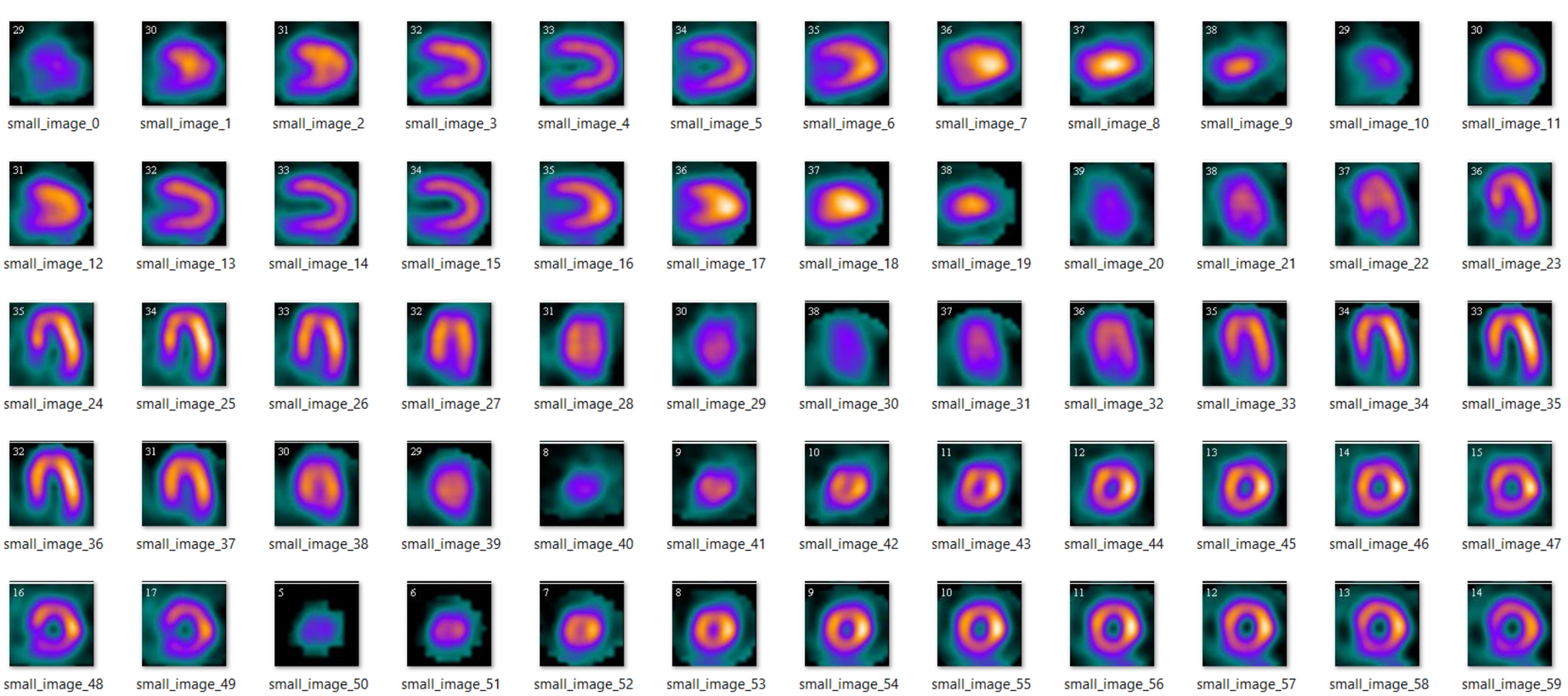
ABNORMAL



3

Solution




small_image_0
small_image_1
small_image_2
small_image_3
small_image_4
small_image_5
small_image_6
small_image_7
small_image_8
small_image_9
small_image_10
small_image_11
small_image_12
small_image_13
small_image_14
small_image_15
small_image_16
small_image_17
small_image_18
small_image_19
small_image_20
small_image_21
small_image_22
small_image_23
small_image_24
small_image_25
small_image_26
small_image_27
small_image_28
small_image_29
small_image_30
small_image_31
small_image_32
small_image_33
small_image_34
small_image_35
small_image_36
small_image_37
small_image_38
small_image_39
small_image_40
small_image_41
small_image_42
small_image_43
small_image_44
small_image_45
small_image_46
small_image_47
small_image_48
small_image_49
small_image_50
small_image_51
small_image_52
small_image_53
small_image_54
small_image_55
small_image_56
small_image_57
small_image_58
small_image_59

CNN Implementation

```
52 # Define the CNN model
53 model = models.Sequential()
54
55 model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, img_channels)))
56 model.add(layers.MaxPooling2D((2, 2)))
57
58 model.add(layers.Conv2D(64, (3, 3), activation='relu'))
59 model.add(layers.MaxPooling2D((2, 2)))
60
61 model.add(layers.Conv2D(128, (3, 3), activation='relu'))
62 model.add(layers.MaxPooling2D((2, 2)))
63
64 model.add(layers.Flatten())
65 model.add(layers.Dense(256, activation='relu'))
66 model.add(layers.Dropout(0.5))
67 model.add(layers.Dense(len(class_names), activation='softmax'))
```

Results Of CNN

```
38/38 [==============================] - 15s 391ms/step - loss: 0.3047 - accuracy: 0.8700 -
Epoch 23/30
38/38 [==============================] - 15s 389ms/step - loss: 0.2903 - accuracy: 0.8650 -
Epoch 24/30
38/38 [==============================] - 15s 390ms/step - loss: 0.2593 - accuracy: 0.8800 -
Epoch 25/30
38/38 [==============================] - 15s 390ms/step - loss: 0.2849 - accuracy: 0.8983 -
Epoch 26/30
38/38 [==============================] - 15s 389ms/step - loss: 0.2826 - accuracy: 0.8817 -
Epoch 27/30
38/38 [==============================] - 15s 390ms/step - loss: 0.2635 - accuracy: 0.9000 -
Epoch 28/30
38/38 [==============================] - 15s 404ms/step - loss: 0.2370 - accuracy: 0.9067 -
Epoch 29/30
38/38 [==============================] - 15s 389ms/step - loss: 0.2195 - accuracy: 0.9050 -
Epoch 30/30
38/38 [==============================] - 14s 368ms/step - loss: 0.2564 - accuracy: 0.8850 -
Found 600 images belonging to 2 classes.
38/38 [==============================] - 5s 119ms/step - loss: 0.1432 - accuracy: 0.9533
Test Accuracy: 0.95333331823349
```

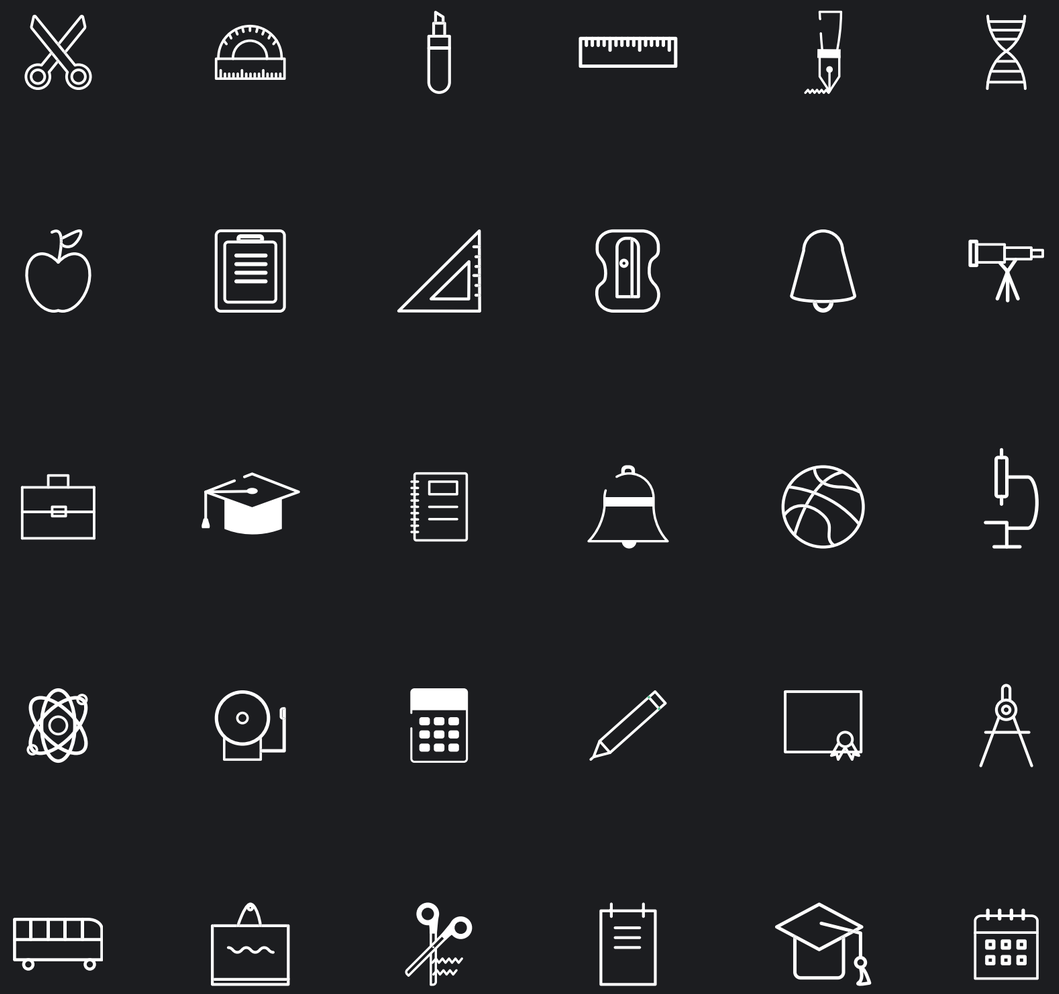
GoogleNET Implementation

```
# Define your custom model
custom_model = models.Sequential()
custom_model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(img_height, img_width, img_channels)))
custom_model.add(layers.MaxPooling2D((2, 2)))
custom_model.add(layers.Conv2D(128, (3, 3), activation='relu'))
custom_model.add(layers.MaxPooling2D((2, 2)))
custom_model.add(layers.Conv2D(256, (3, 3), activation='relu'))
custom_model.add(layers.MaxPooling2D((2, 2)))
custom_model.add(layers.Flatten())
custom_model.add(layers.Dense(512, activation='relu'))
custom_model.add(layers.Dropout(0.5))

# Load InceptionV3 (GoogLeNet) model with pre-trained weights (excluding the top layers)
inception_model = InceptionV3(input_shape=(img_height, img_width, img_channels), include_top=False, weights='imagenet')

```

Results Of GoogleNET

```
Epoch 40/50
100/100 [==============================] - 130s 1s/step - loss: 0.6519 - accuracy: 0.6066 - val_loss: 0.6553 - val_accuracy: 0.6084
Epoch 41/50
100/100 [==============================] - 116s 1s/step - loss: 0.6492 - accuracy: 0.6424 - val_loss: 0.6545 - val_accuracy: 0.6135
Epoch 42/50
100/100 [==============================] - 128s 1s/step - loss: 0.6573 - accuracy: 0.6035 - val_loss: 0.6557 - val_accuracy: 0.6009
Epoch 43/50
100/100 [==============================] - 114s 1s/step - loss: 0.6472 - accuracy: 0.6261 - val_loss: 0.6543 - val_accuracy: 0.6130
Epoch 44/50
100/100 [==============================] - 112s 1s/step - loss: 0.6492 - accuracy: 0.6173 - val_loss: 0.6543 - val_accuracy: 0.6130
Epoch 45/50
100/100 [==============================] - 128s 1s/step - loss: 0.6512 - accuracy: 0.6261 - val_loss: 0.6530 - val_accuracy: 0.6114
Epoch 46/50
100/100 [==============================] - 128s 1s/step - loss: 0.6448 - accuracy: 0.6274 - val_loss: 0.6567 - val_accuracy: 0.6024
Epoch 47/50
100/100 [==============================] - 128s 1s/step - loss: 0.6464 - accuracy: 0.6368 - val_loss: 0.6519 - val_accuracy: 0.6130
Epoch 48/50
100/100 [==============================] - 128s 1s/step - loss: 0.6490 - accuracy: 0.6299 - val_loss: 0.6496 - val_accuracy: 0.6205
Epoch 49/50
100/100 [==============================] - 133s 1s/step - loss: 0.6465 - accuracy: 0.6299 - val_loss: 0.6577 - val_accuracy: 0.6165
Epoch 50/50
100/100 [==============================] - 127s 1s/step - loss: 0.6494 - accuracy: 0.6217 - val_loss: 0.6475 - val_accuracy: 0.6210
Found 1992 images belonging to 2 classes.
125/125 [==============================] - 27s 213ms/step - loss: 0.6475 - accuracy: 0.6210
Test Accuracy: 0.6209839582443237
```

Any Questions?



THANKS ALOT FOR YOUR PATIENCE