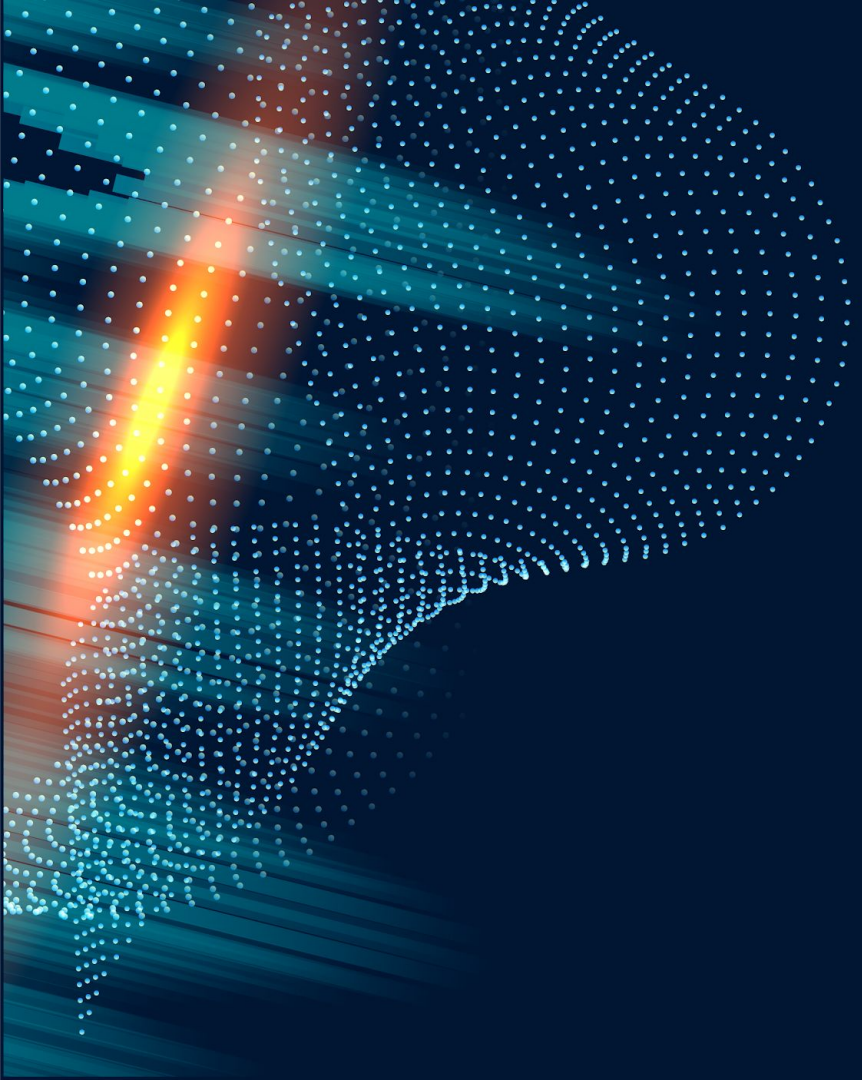
IMAGE PROCESSING MINI PROJECT

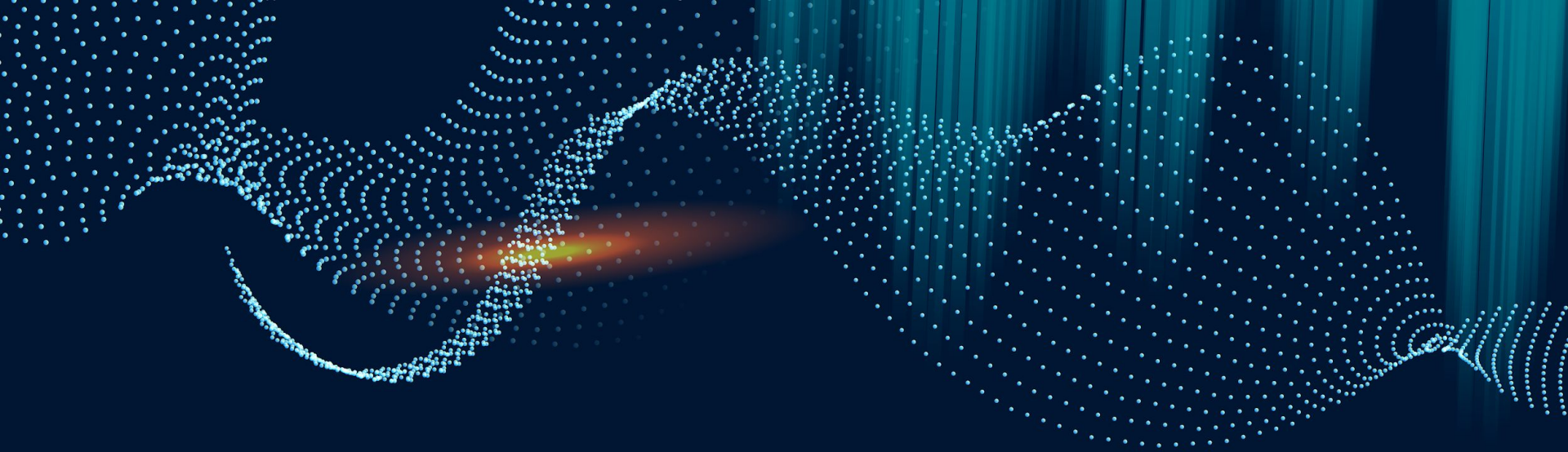
EC386

PM Prasanna 201EC242
Abhinav Raghunandan 201EC102



CAPSULE ENDOSCOPY

Image Classification Techniques

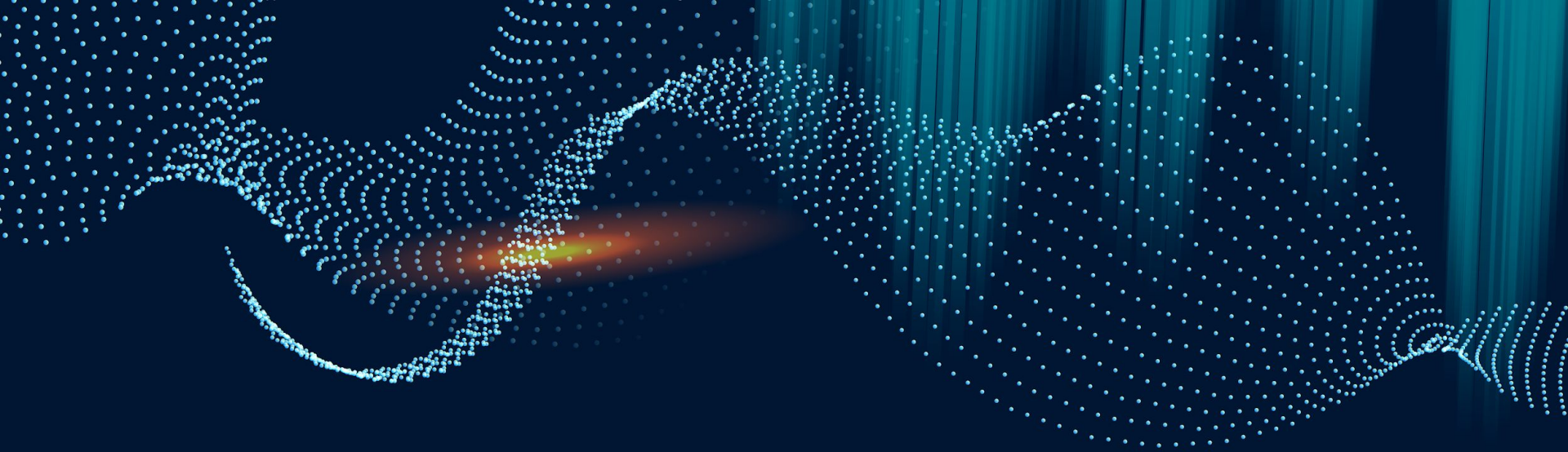


01 PROBLEM STATEMENT

Brief description about the Aim of the this project.

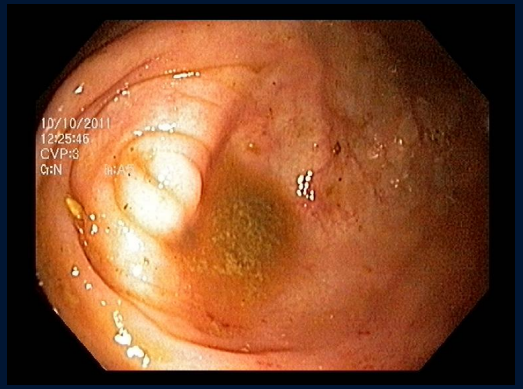
Perform Anatomical Classification using interior images of the intestine obtained from the Capsule Endoscopy Technique

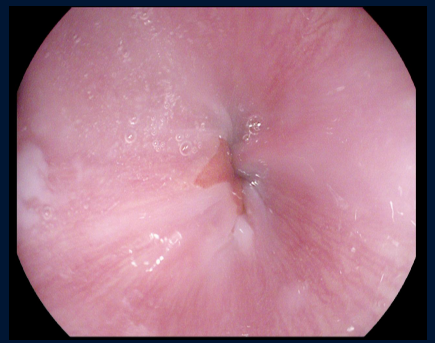
The intestine contains 3 major regions - Pylorus, Cecum and the Z- Line. Perform Image Classification based on texture, size and other features which is useful for medical treatments.



02 DATASET

- The dataset contains images of the different sections of the colon captured using Capsule Endoscopy
- Source of Dataset : Dr. Aparna P Dinesh (Project Mentor)





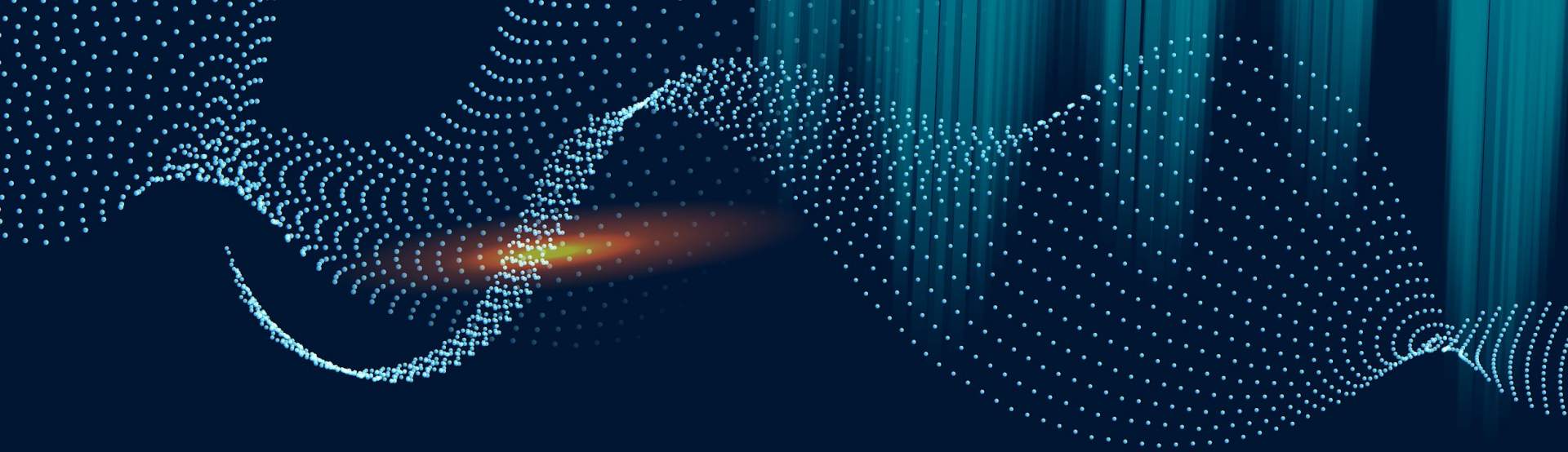
Data Augmentation

- Rotation, Zooming in and out, shifting height and width and horizontal flipping of the image

- Performed Data augmentation and finally obtained 131 images belonging to 3 classes as ‘Training’ data
- And 31 images belonging to 3 classes as Validation data.

```
# create a data generator
datagen = ImageDataGenerator(
        samplewise_center=True,  # set each sample mean to 0
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False)
```

```
[ ] train_it = datagen.flow_from_directory('/content/gdrive/MyDrive/Mini_Project/Train',
                                           target_size=(224, 224),
                                           color_mode='rgb',
                                           class_mode='categorical',
                                           batch_size=5)
    # load and iterate validation dataset
    valid_it = datagen.flow_from_directory('/content/gdrive/MyDrive/Mini_Project/Test',
                                          target_size=(224, 224),
                                          color_mode='rgb',
                                          class_mode='categorical',
                                          batch_size=5)
```



03 PREVIOUS WORK

Transfer Learning Model Implementation using VGG- 16

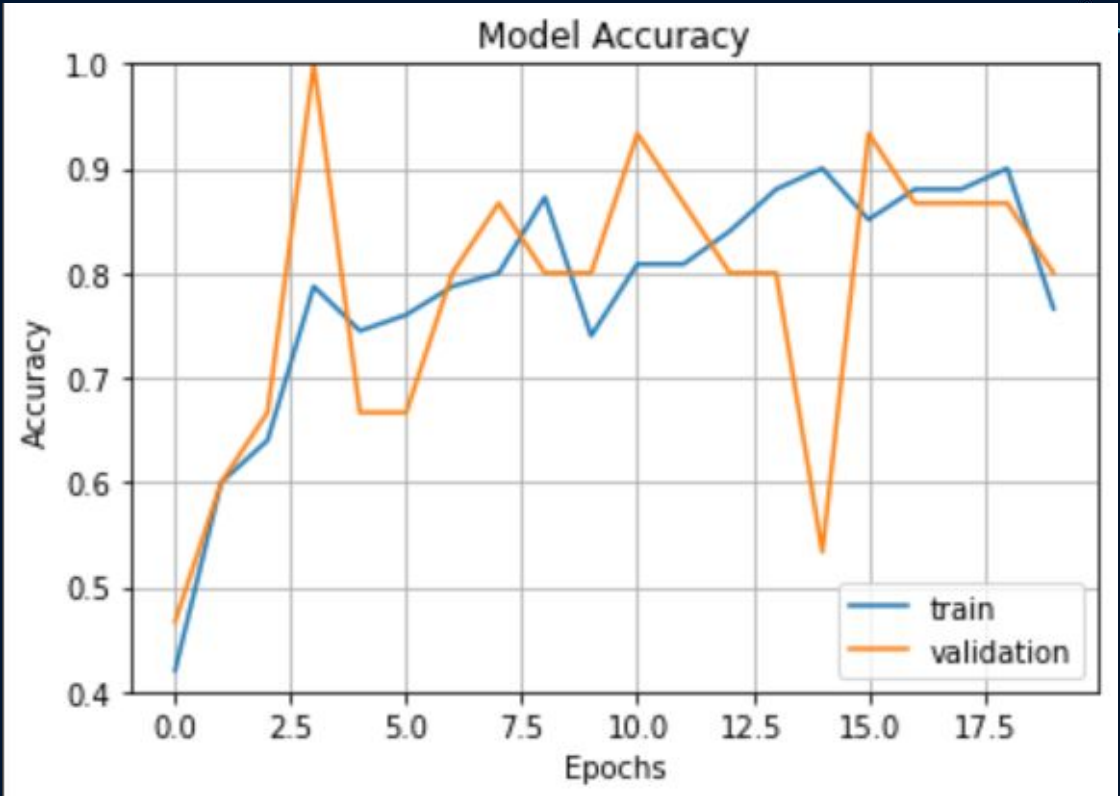
- Classified the image dataset using VGG-16 model.

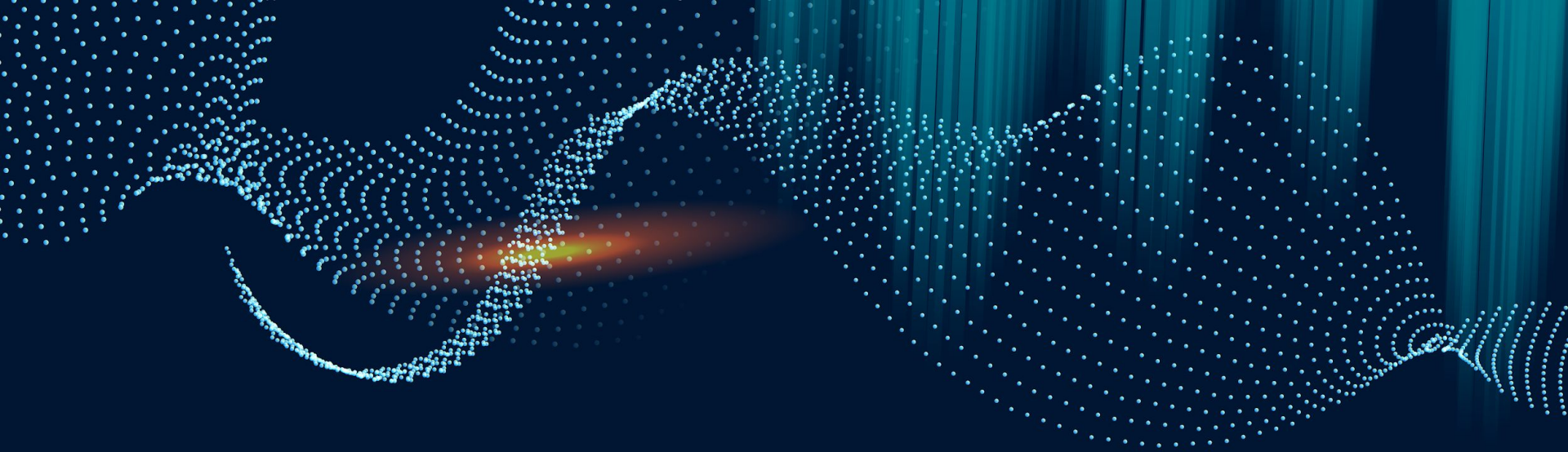
Reasons for choosing VGG-16

- Small dataset used
- Highly efficient and accurate as it is a pre-trained model

GitHub Repo Code Link VGG16

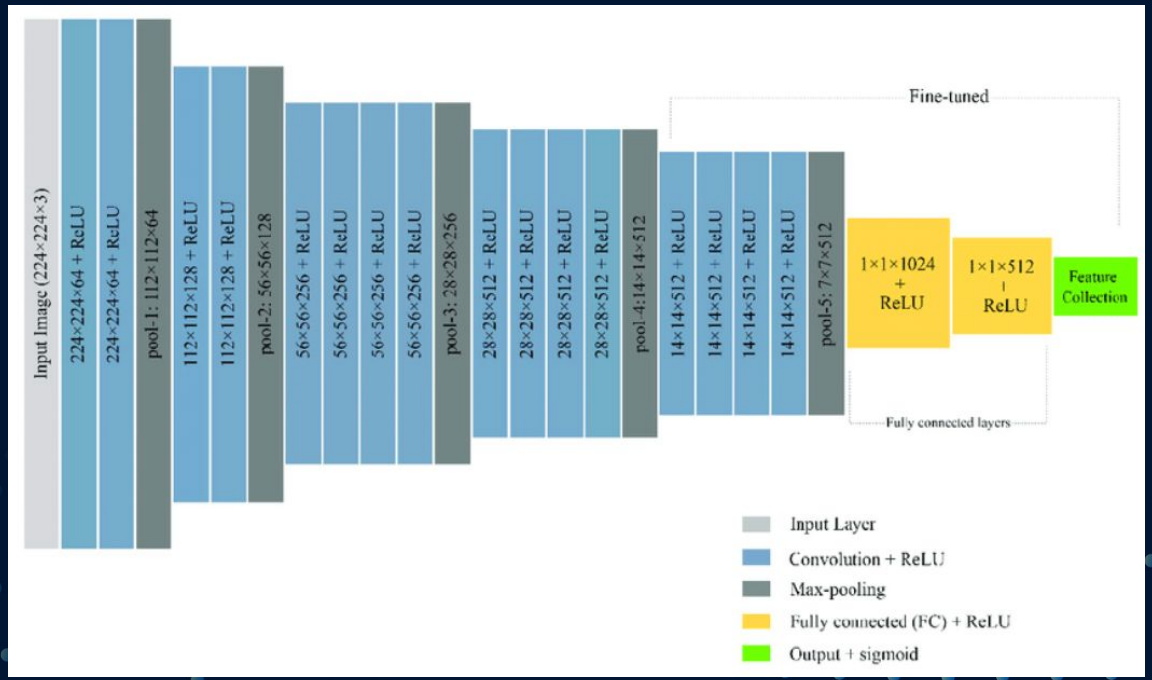
Accuracy Graph Plot





04 RECENT WORK

VGG-19 MODEL


Input Image (224×224×3)
224×224×64 + ReLU
224×224×64 + ReLU
pool-1: 112×112×64
112×112×128 + ReLU
112×112×128 + ReLU
pool-2: 56×56×128
56×56×256 + ReLU
56×56×256 + ReLU
56×56×256 + ReLU
56×56×256 + ReLU
pool-3: 28×28×256
28×28×512 + ReLU
28×28×512 + ReLU
28×28×512 + ReLU
28×28×512 + ReLU
pool 4:14×14×512
14×14×512 + ReLU
14×14×512 + ReLU
14×14×512 + ReLU
14×14×512 + ReLU
pool-5: 7×7×512
Fine-tuned
1×1×1024 + ReLU
1×1×512 + ReLU
Feature Collection
Fully connected layers
Input Layer
Convolution + ReLU
Max-pooling
Fully connected (FC) + ReLU
Output + sigmoid

Using VGG- 19 Model

- Classified the image dataset using VGG-19 model.
- Model was suggested by our Project Mentor

Advantages of VGG-19

- 3 more Convolutional Layers than the VGG-16 Model
- Bound to result in Higher Accuracy

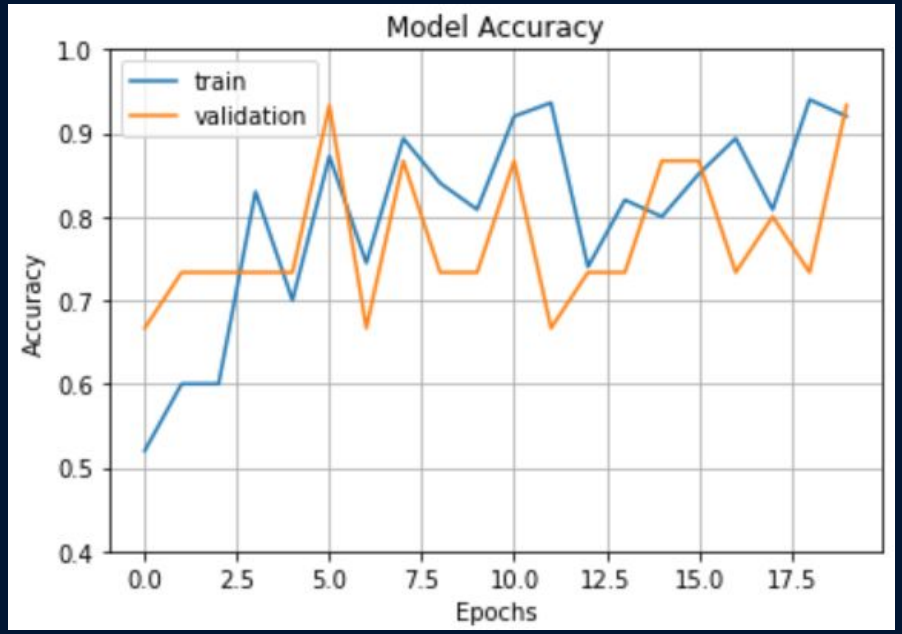
GitHub Repo Code Link VGG19

VGG-19 Model Summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 3)                 387

=================================================================
Total params: 32,936,003
Trainable params: 12,911,619
Non-trainable params: 20,024,384
_________________________________________________________________
```

Accuracy Graph Plot

```
2s 226ms/step - loss: 1.3921 - accuracy: 0.9200 - val_loss: 4.9863 - val_accuracy: 0.8667

3s 308ms/step - loss: 0.7570 - accuracy: 0.9362 - val_loss: 4.5534 - val_accuracy: 0.6667
```



Model Implementation

```
VGG19_model = Sequential()
pretrained_model = keras.applications.vgg19(
    weights='imagenet',  # Load weights pre-trained on ImageNet.
    input_shape=(224, 224, 3),
    include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/v
80134624/80134624 [==============================] - 4s 0us/step
```

```
[ ] for layer in pretrained_model.layers:
        layer.trainable=False

    VGG19_model.add(pretrained_model)
```

```
[ ] VGG19_model.add(Flatten())
    VGG19_model.add(Dense(512, activation='relu'))
    VGG19_model.add(Dense(128, activation='relu'))
    VGG19_model.add(Dense(3, activation='softmax'))
```

EfficientNet B0

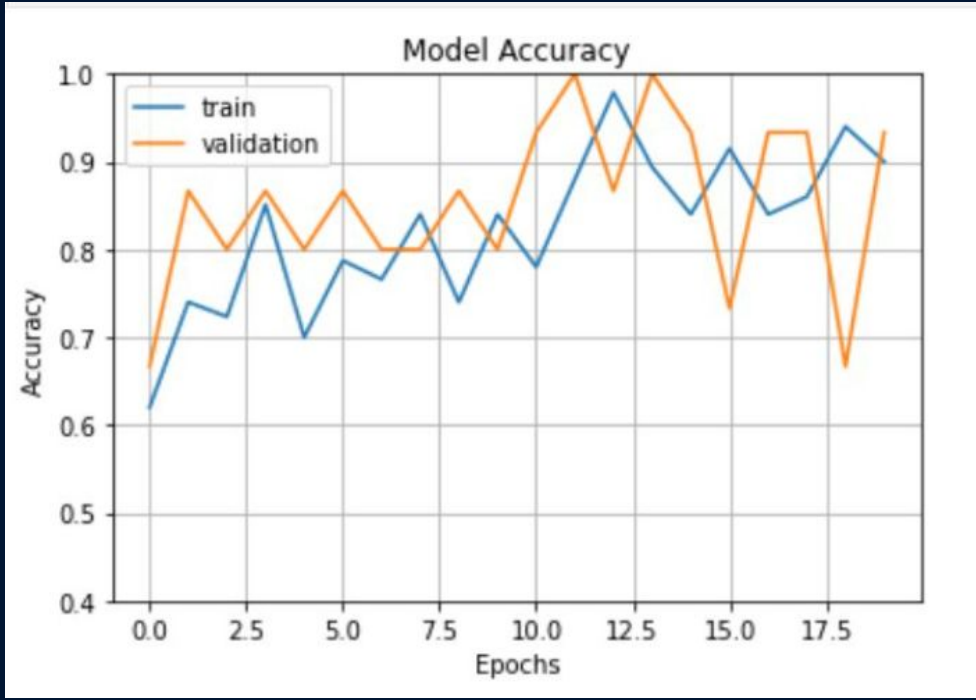
EfficientNet B0

- Paper published on EfficientNet and ResNet Networks

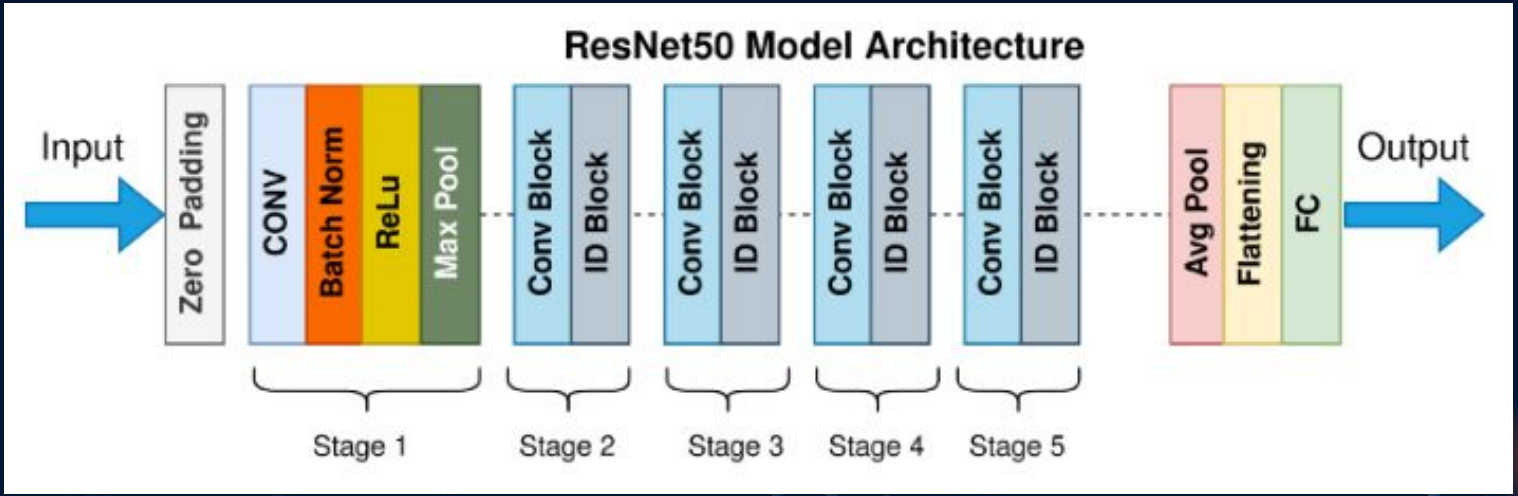
- Lower accuracy than expected since we run the risk of Overfitting when we use EfficientNet with small dataset applications

GitHub Repo Code Link EfficientNet

Code and Implementation



ResNet-50 MODEL



Using ResNet-50 Model

- ResNet-50 is a Convolutional Neural Network which is 50 layers deep
- Pre- trained model has trained over a million images using the ImageNet Database

Reasons for choosing ResNet-50

Article about ResNet enhancing performance

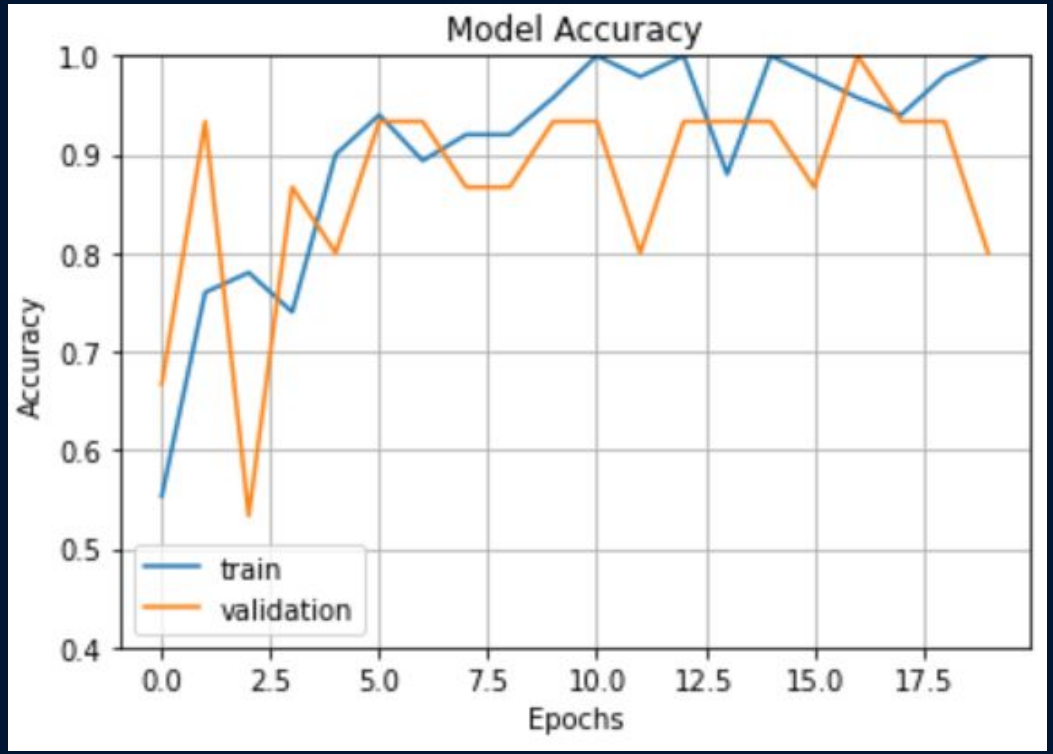
- Significant enhancement of performance due to more convolutional layers
- Maintain low error rate in the deep layers of the network

GitHub Repo Code Link ResNet50

ResNet-50 Model Summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 3)                 387

=================================================================
Total params: 24,702,851
Trainable params: 1,115,139
Non-trainable params: 23,587,712
```

Accuracy Graph Plot



```
- 13s 1s/step - loss: 0.1354 - accuracy: 0.9200 - val_loss: 0.1416 - val_accuracy: 0.8667

- 12s 1s/step - loss: 0.0945 - accuracy: 0.9574 - val_loss: 0.0584 - val_accuracy: 0.9333

- 12s 1s/step - loss: 0.0498 - accuracy: 1.0000 - val_loss: 0.1323 - val_accuracy: 0.9333

- 12s 1s/step - loss: 0.0749 - accuracy: 0.9787 - val_loss: 0.4320 - val_accuracy: 0.8000

- 13s 1s/step - loss: 0.0152 - accuracy: 1.0000 - val_loss: 0.0845 - val_accuracy: 0.9333

- 13s 1s/step - loss: 0.1623 - accuracy: 0.8800 - val_loss: 0.2113 - val_accuracy: 0.9333

- 12s 1s/step - loss: 0.0662 - accuracy: 1.0000 - val_loss: 0.2632 - val_accuracy: 0.9333

- 12s 1s/step - loss: 0.0482 - accuracy: 0.9787 - val_loss: 0.5581 - val_accuracy: 0.8667
```

Model Implementation

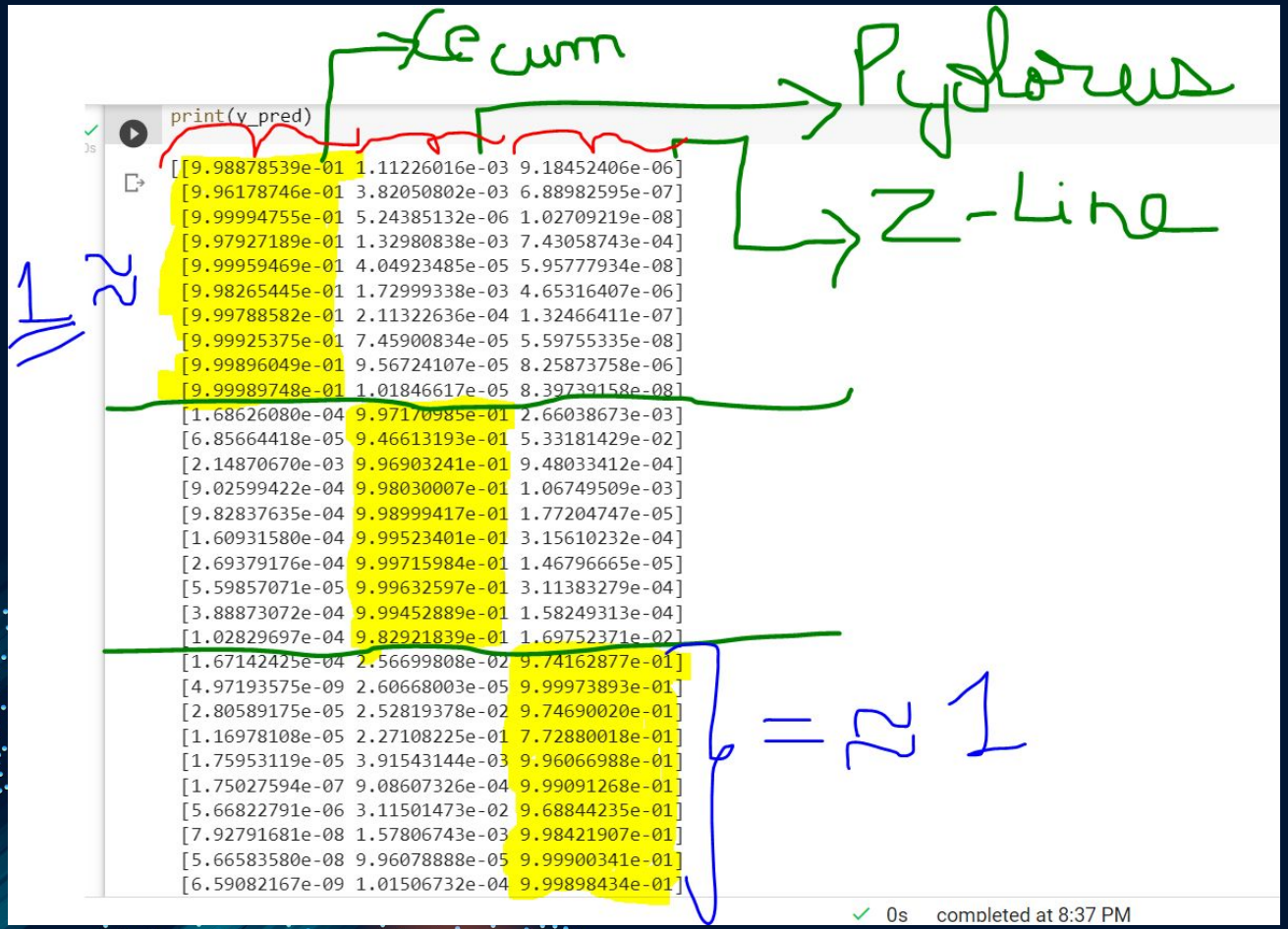
```
conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5_block1_2_bn[0][0]']
n)

conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4_block6_out[0][0]']

conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5_block1_2_relu[0][0]']
```

```
for layer in pretrained_model.layers:
        layer.trainable=False

resnet_model.add(pretrained_model)
```

```
resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation='relu'))
resnet_model.add(Dense(128, activation='relu'))
resnet_model.add(Dense(3, activation='softmax'))
```

Confusion Matrix for the ResNet-50 Model

	CECUM	PYLORUS	Z- LINE
cecum1.png	0.998	1.65×10^{-8}	5.22×10^{-14}
pylorus4.png	7.07×10^{-9}	0.999	10^{-12}
z_line7.png	8.91×10^{-14}	2.08×10^{-8}	0.999


print(y_pred)
[[9.98878539e-01 1.11226016e-03 9.18452406e-06]
[9.96178746e-01 3.82050802e-03 6.88982595e-07]
[9.99994755e-01 5.24385132e-06 1.02709219e-08]
[9.97927189e-01 1.32980838e-03 7.43058743e-04]
[9.99959469e-01 4.04923485e-05 5.95777934e-08]
[9.98265445e-01 1.72999338e-03 4.65316407e-06]
[9.99788582e-01 2.11322636e-04 1.32466411e-07]
[9.99925375e-01 7.45900834e-05 5.59755335e-08]
[9.99896049e-01 9.56724107e-05 8.25873758e-06]
[9.99989748e-01 1.01846617e-05 8.39739158e-08]
[1.68626080e-04 9.97170985e-01 2.66038673e-03]
[6.85664418e-05 9.46613193e-01 5.33181429e-02]
[2.14870670e-03 9.96903241e-01 9.48033412e-04]
[9.02599422e-04 9.98030007e-01 1.06749509e-03]
[9.82837635e-04 9.98999417e-01 1.77204747e-05]
[1.60931580e-04 9.99523401e-01 3.15610232e-04]
[2.69379176e-04 9.99715984e-01 1.46796665e-05]
[5.59857071e-05 9.99632597e-01 3.11383279e-04]
[3.88873072e-04 9.99452889e-01 1.58249313e-04]
[1.02829697e-04 9.82921839e-01 1.69752371e-02]
[1.67142425e-04 2.56699808e-02 9.74162877e-01]
[4.97193575e-09 2.60668003e-05 9.99973893e-01]
[2.80589175e-05 2.52819378e-02 9.74690020e-01]
[1.16978108e-05 2.27108225e-01 7.72880018e-01]
[1.75953119e-05 3.91543144e-03 9.96066988e-01]
[1.75027594e-07 9.08607326e-04 9.99091268e-01]
[5.66822791e-06 3.11501473e-02 9.68844235e-01]
[7.92791681e-08 1.57806743e-03 9.98421907e-01]
[5.66583580e-08 9.96078888e-05 9.99900341e-01]
[6.59082167e-09 1.01506732e-04 9.99898434e-01]]
0s completed at 8:37 PM
Ecum
Pylorus
Z-Line
1 ≈
= ≈ 1

98.84%

Accuracy of our ResNet- 50 Model

SUMMARY OF MODELS

VGG-16

Accuracy
88.45%

VGG-19

Accuracy
93.62%

EfficientNet B0

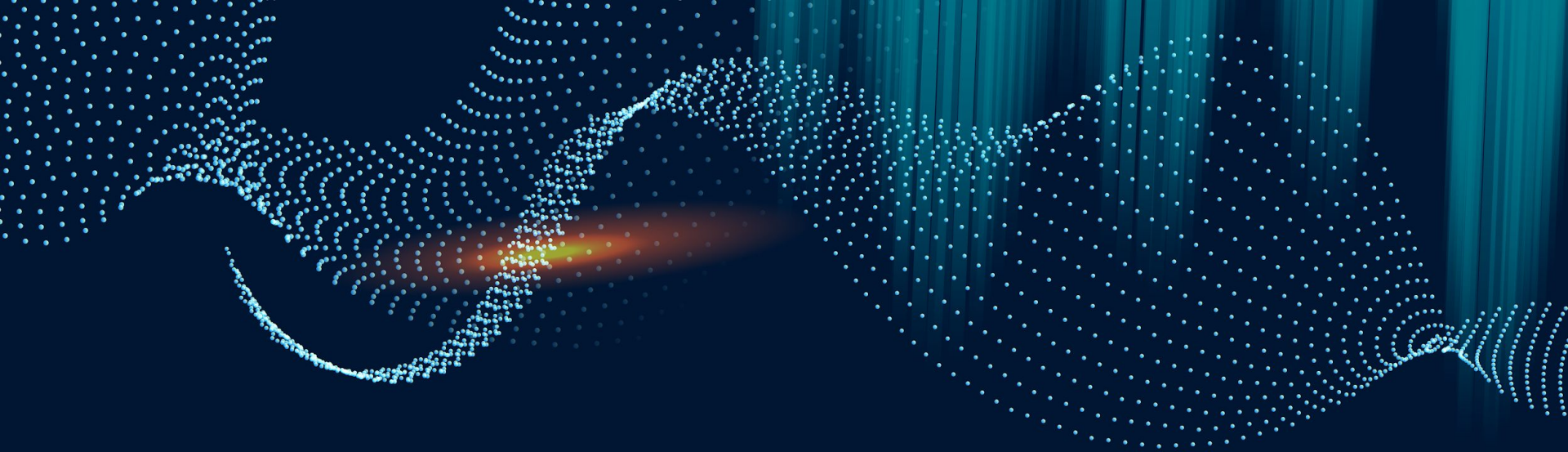
Accuracy
95.79%

ResNet-50

Accuracy
98.84%

REFERENCES

1. Introduction to Image Classification
2. About Data Augmentation
3. About Activation Functions
4. Transfer Learning
5. VGG-19 Model
6. Paper on EfficientNet and ResNets
7. Parameters in Convolutional Neural Networks
8. StackOverflow for General Error Handling



THANK YOU

PM Prasanna 201EC242
Abhinav Raghunandan 201EC102