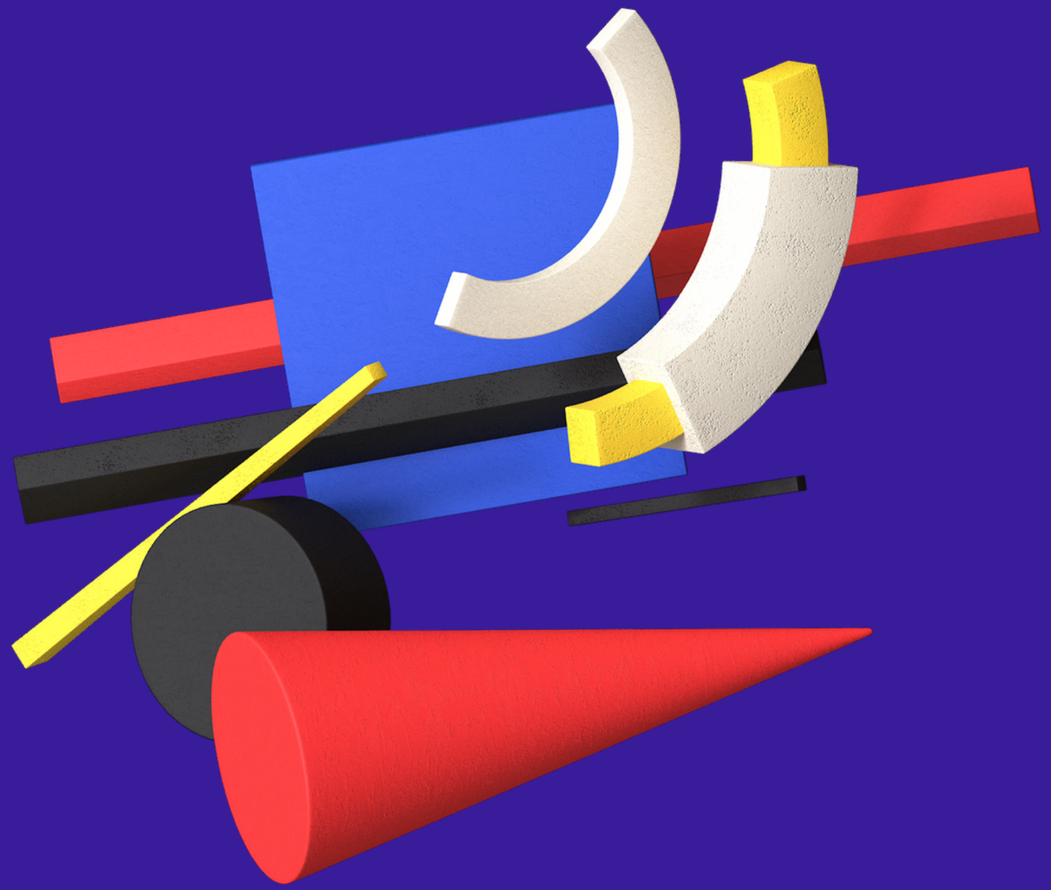
CA4

DEEP LEARNING

PLANT DISEASE DETECTION



Team Members

Siva Kumar CH V (E0119014)

Sai Ganesh A (E0119037)

Banari Babu B (E0119029)

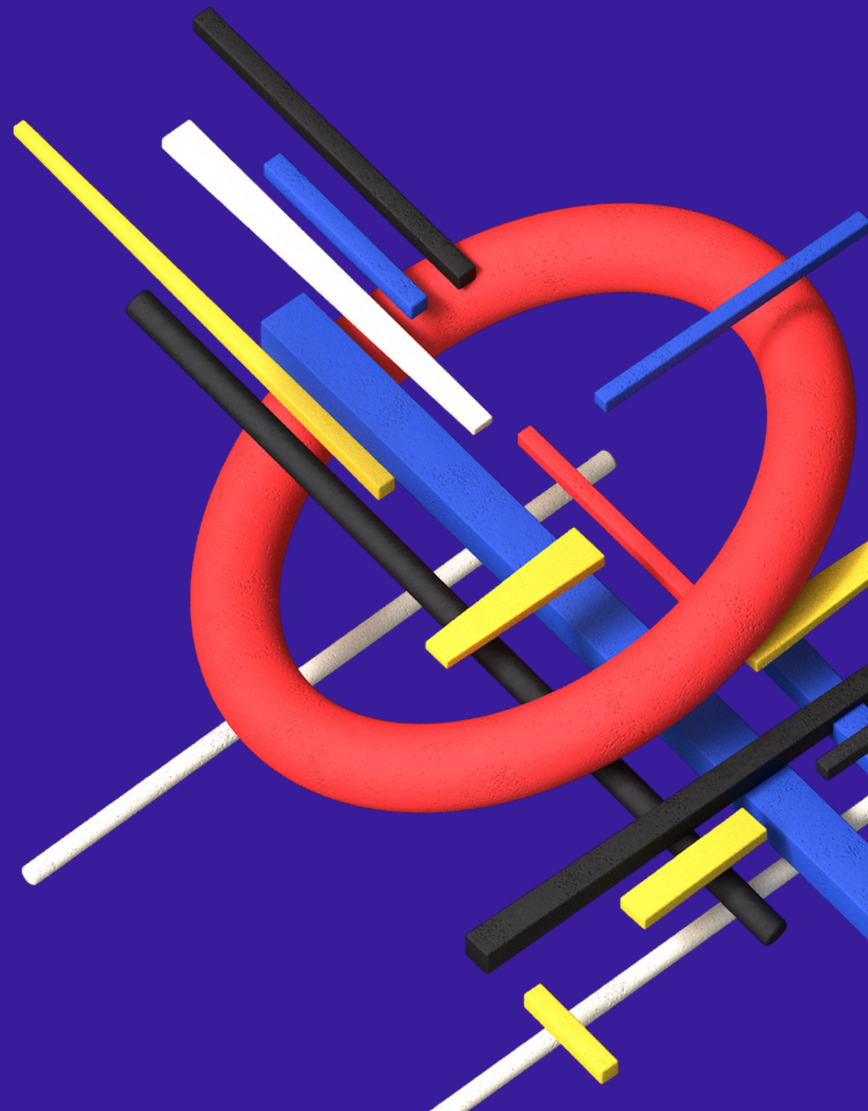
Dileep A (E0119033)

Akash Reddy Y (E0119037)

Table Of Contents

Problem Statement

To detect some commonly occurring diseases in plants



ABOUT DATASET

The Dataset contains 15 directories and total of 20,639 images.

The images span 3 Crop Species : Pepper , Potato , Tomato

The Data set contains 15 classess of crop disease pairs and is listed below :

Pepper Bell Bacterial Spot , Pepper Bell Healthy , Potato Early Blight , Potato Late Blight , Potato Healthy etc ..

Kaggle Link : https://www.kaggle.com/emmarex/plantdisease

ABOUT DATASET

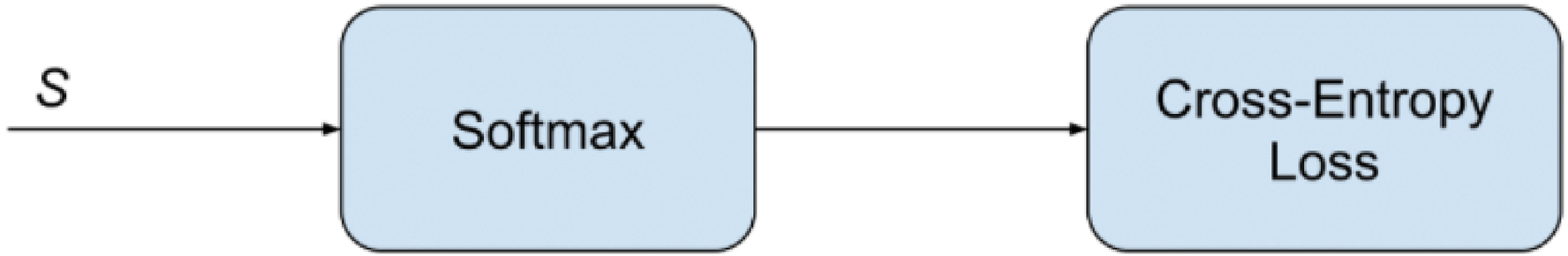
	0
Pepper__bell___Bacterial_spot	997
Potato___healthy	152
Tomato_Leaf_Mold	952
Tomato__Tomato_YellowLeaf__Curl_Virus	3209
Tomato_Bacterial_spot	2127
Tomato_Septoria_leaf_spot	1771

Tomato_healthy	1591
Tomato_Spider_mites_Two_spotted_spider_mite	1676
Tomato_Early_blight	1000
Tomato__Target_Spot	1404
Pepper__bell___healthy	1478
Potato___Late_blight	1000
Tomato_Late_blight	1909
Potato___Early_blight	1000
Tomato__Tomato_mosaic_virus	373

Metrics Used

- Precision
  - It expresses the proportion of the data points our model says was relevant actually were relevant
- Recall
  - It refers to percentage of total relevant results correctly classified by the model
- F1 Score
  - Weighted Average of precision and Recall

Loss Function :



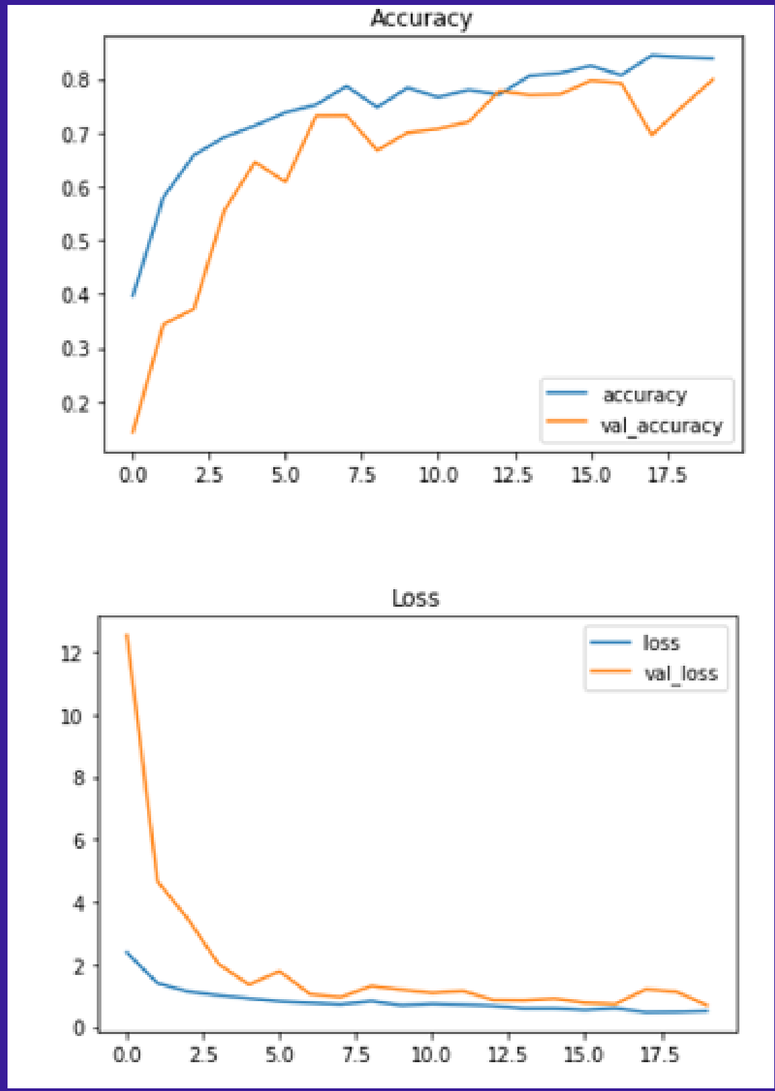
$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

Custom Model

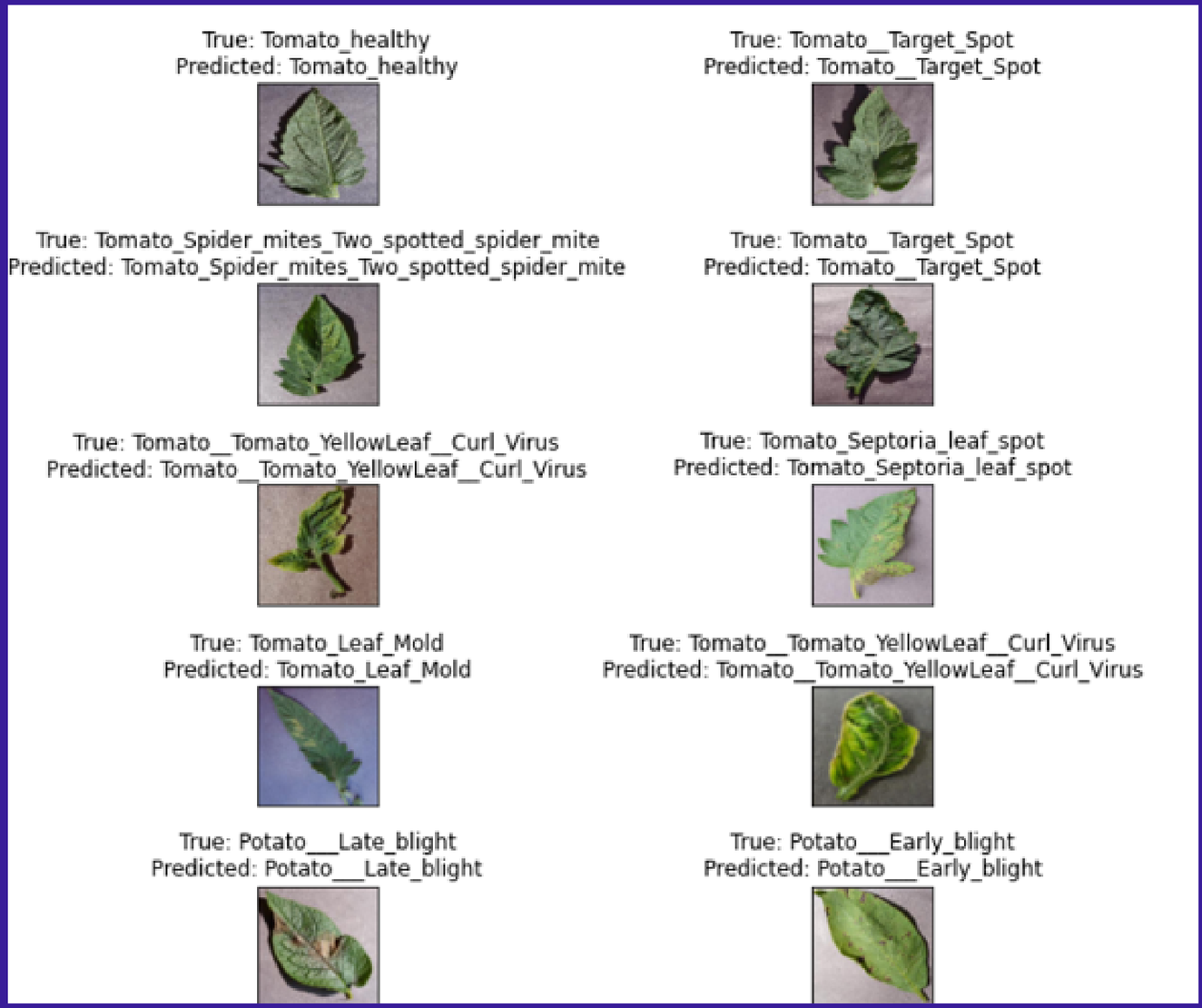
```
n_classes = 15
custom_model = Sequential()
custom_model.add(Conv2D(32, (3, 3), padding="same",input_shape=(100,100,3)))
custom_model.add(Activation("relu"))
custom_model.add(BatchNormalization())
custom_model.add(MaxPooling2D(pool_size=(3, 3)))
custom_model.add(Dropout(0.25))
custom_model.add(Conv2D(64, (3, 3), padding="same"))
custom_model.add(Activation("relu"))
custom_model.add(BatchNormalization())
custom_model.add(Conv2D(64, (3, 3), padding="same"))
custom_model.add(Activation("relu"))
custom_model.add(BatchNormalization())
custom_model.add(MaxPooling2D(pool_size=(2, 2)))
custom_model.add(Dropout(0.25))
custom_model.add(Conv2D(128, (3, 3), padding="same"))
custom_model.add(Activation("relu"))
custom_model.add(BatchNormalization())
custom_model.add(Conv2D(128, (3, 3), padding="same"))
custom_model.add(Activation("relu"))
custom_model.add(BatchNormalization())
custom_model.add(MaxPooling2D(pool_size=(2, 2)))
custom_model.add(Dropout(0.25))
custom_model.add(Flatten())
custom_model.add(Dense(1024))
custom_model.add(Activation("relu"))
custom_model.add(BatchNormalization())
custom_model.add(Dropout(0.5))
custom_model.add(Dense(n_classes))
custom_model.add(Activation("softmax"))
```

Custom Model

loss : 0.50
Accuracy : 0.83
validation loss : 0.57
validation accuracy : 0.79



Custom Model Predictions

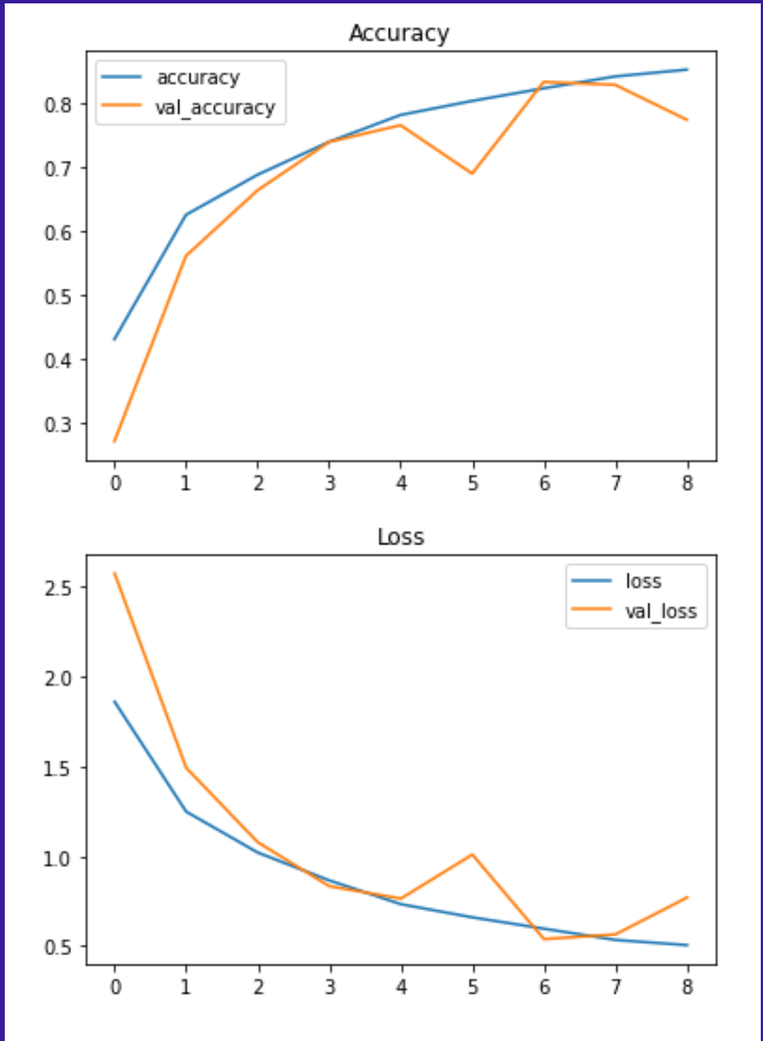


AlexNet

```
Alexnet.add(Flatten())
Alexnet.add(Dense(4096,input_shape=(100,100,3,)))
Alexnet.add(BatchNormalization())
Alexnet.add(Activation('relu'))
Alexnet.add(Dropout(0.4))

Alexnet.add(Dense(4096))
Alexnet.add(BatchNormalization())
Alexnet.add(Activation('relu'))
Alexnet.add(Dropout(0.4))

Alexnet.add(Dense(1000))
Alexnet.add(BatchNormalization())
Alexnet.add(Activation('relu'))
Alexnet.add(Dropout(0.4))

Alexnet.add(Dense(15))
Alexnet.add(BatchNormalization())
Alexnet.add(Activation('softmax'))
```

```
Alexnet.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
my_callbacks  = [EarlyStopping(monitor='val_accuracy',
                               min_delta=0,
                               patience=2,
                               mode='auto')]

AlexNet_history=Alexnet.fit(
    train_gen,
    validation_data=valid_gen,
    epochs=10,
    callbacks=my_callbacks
)
```

AlexNet



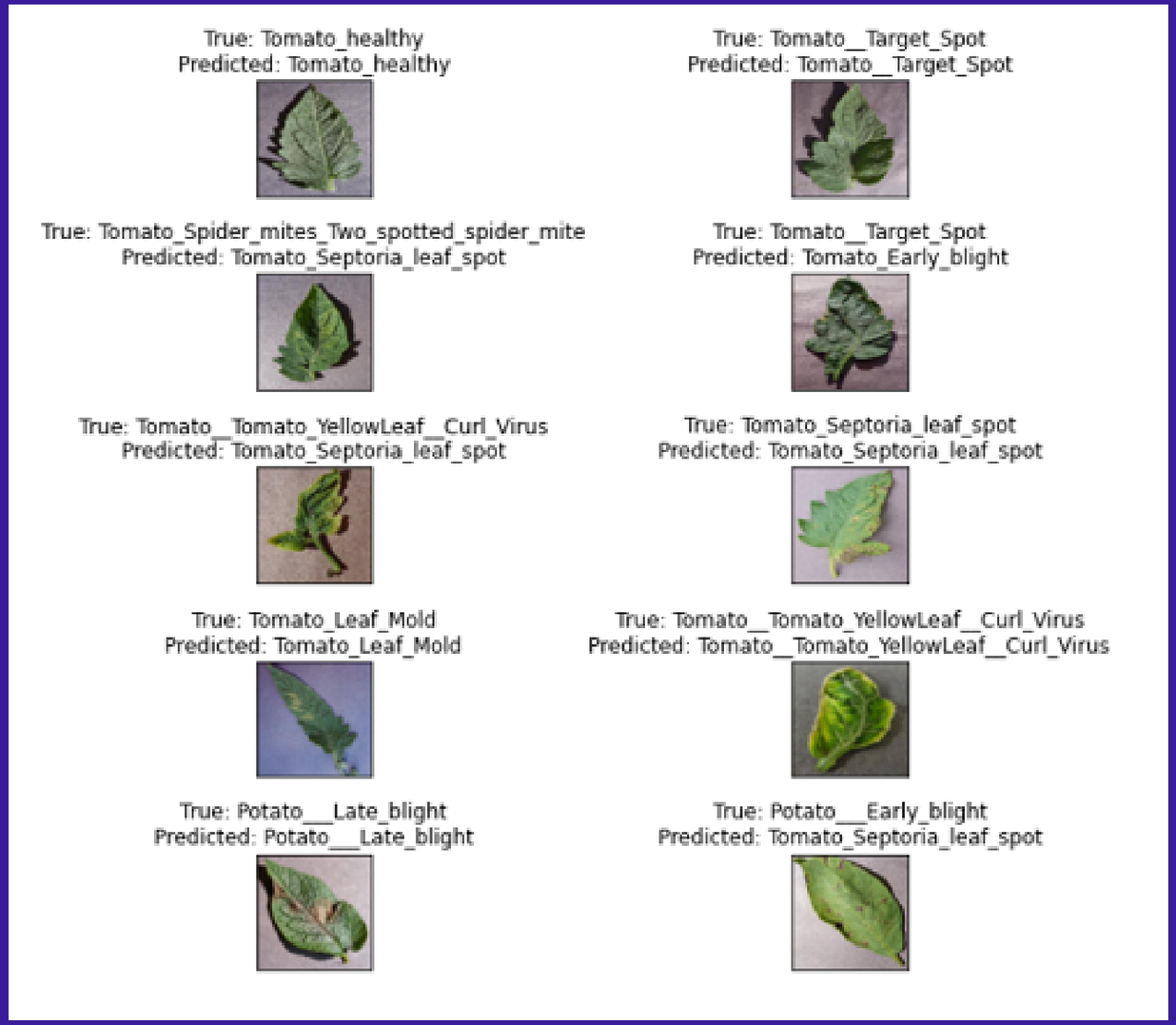
loss : 0.50

Accuracy : 0.78

validation loss : 0.60

validation accuracy : 0.77

AlexNet Model Predictions

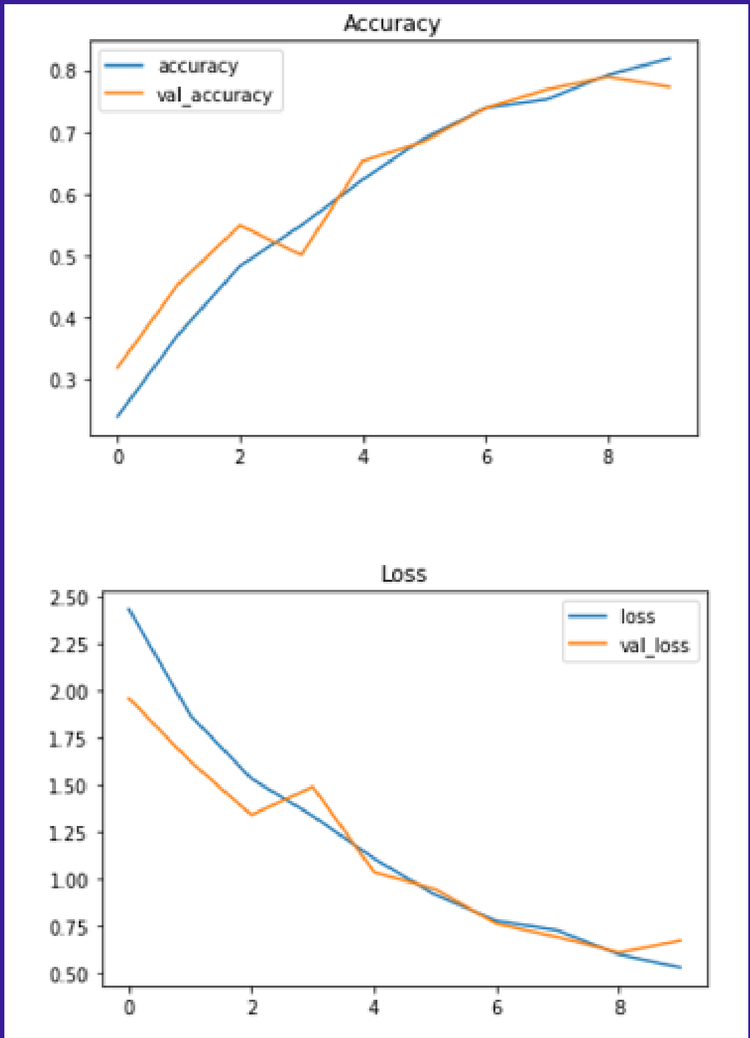


VGG19 Model

```
vgg_model = VGG19(
    input_shape=(100,100, 3),
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

inputs = vgg_model.input

x = Dense(128, activation='relu')(vgg_model.output)
x = Dense(128, activation='relu')(x)

outputs = Dense(15, activation='softmax')(x)

vgg19_model = Model(inputs=inputs, outputs=outputs)
```

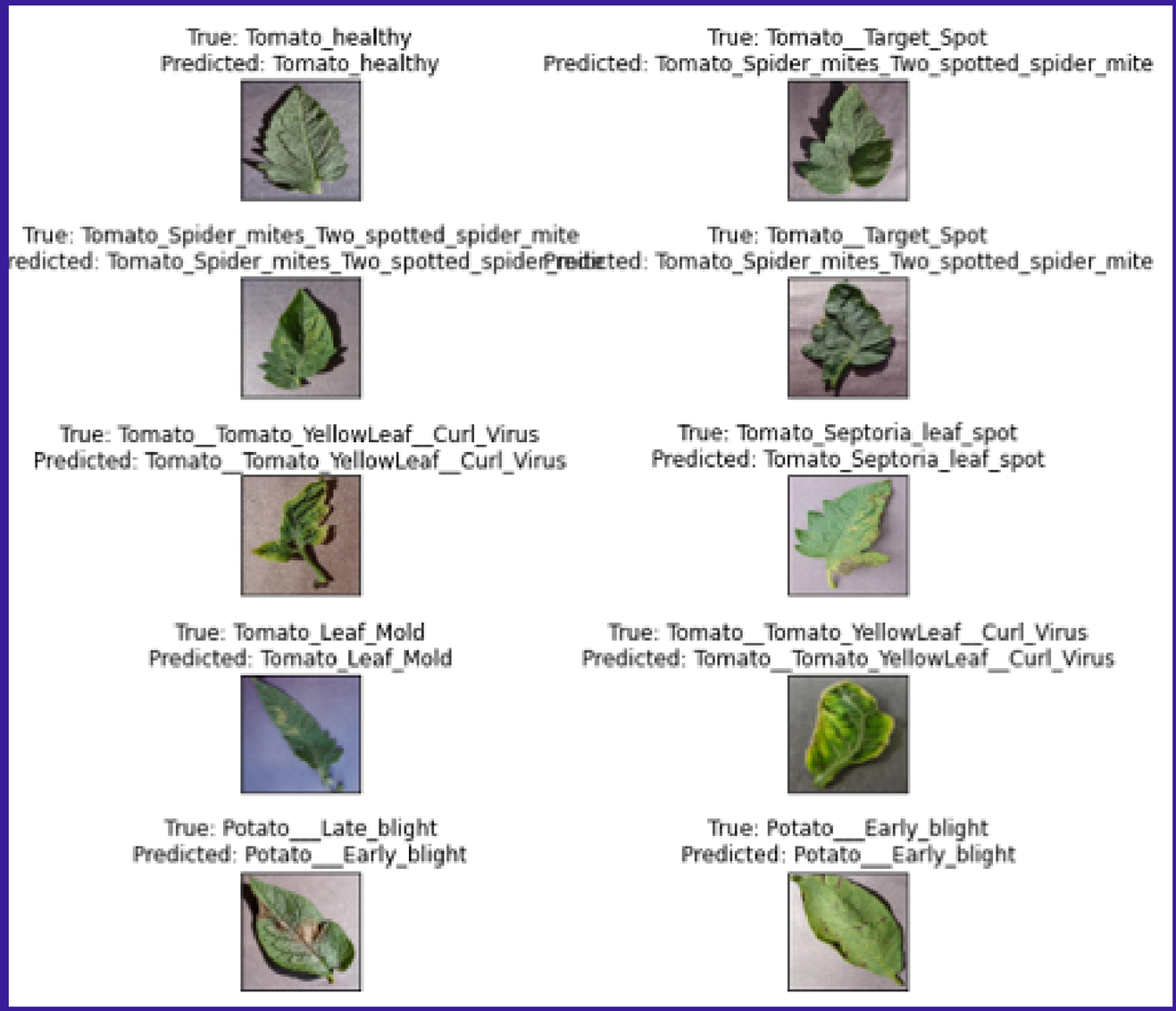
```
vgg19_model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

my_callbacks  = [EarlyStopping(monitor='val_accuracy',
                               min_delta=0,
                               patience=2,
                               mode='auto')]
```

VGG19 and Metrics



loss	:	0.65
Accuracy	:	0.78
validation loss	:	0.67
validation accuracy	:	0.77

VGG 19 Model Predictions



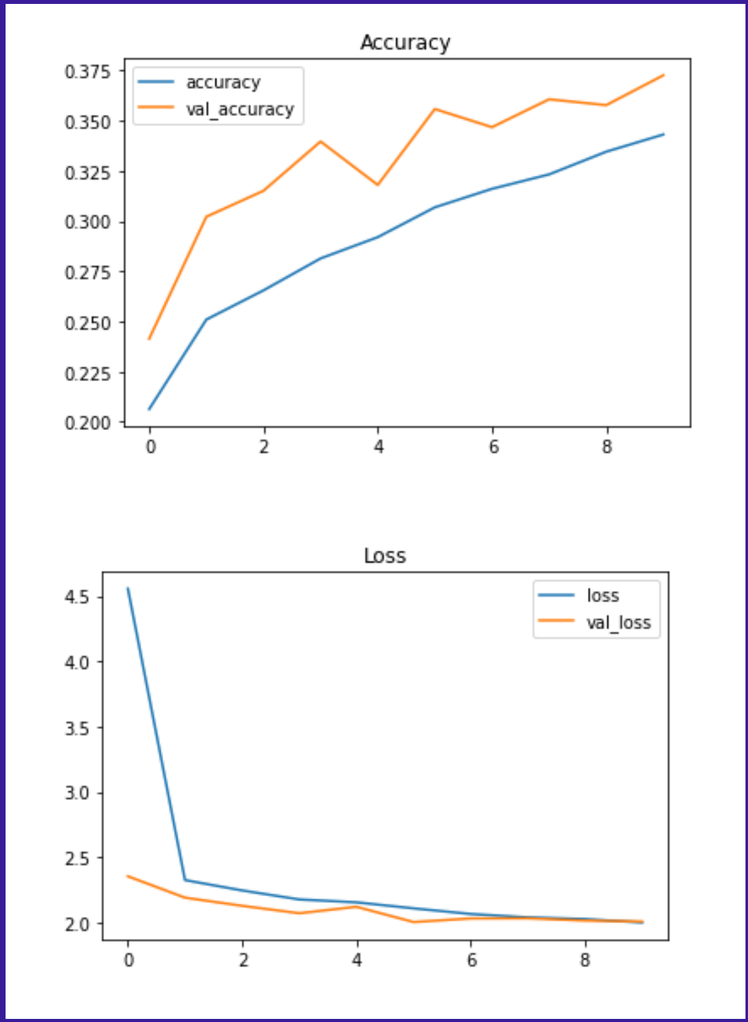
Google Net

GoogLeNet is a 22-layer deep convolutional neural network that's a variant of the Inception Network, a Deep Convolutional Neural Network

The GoogleNet Architecture is 22 layers deep, with 27 pooling layers included. There are 9 inception modules stacked linearly in total.

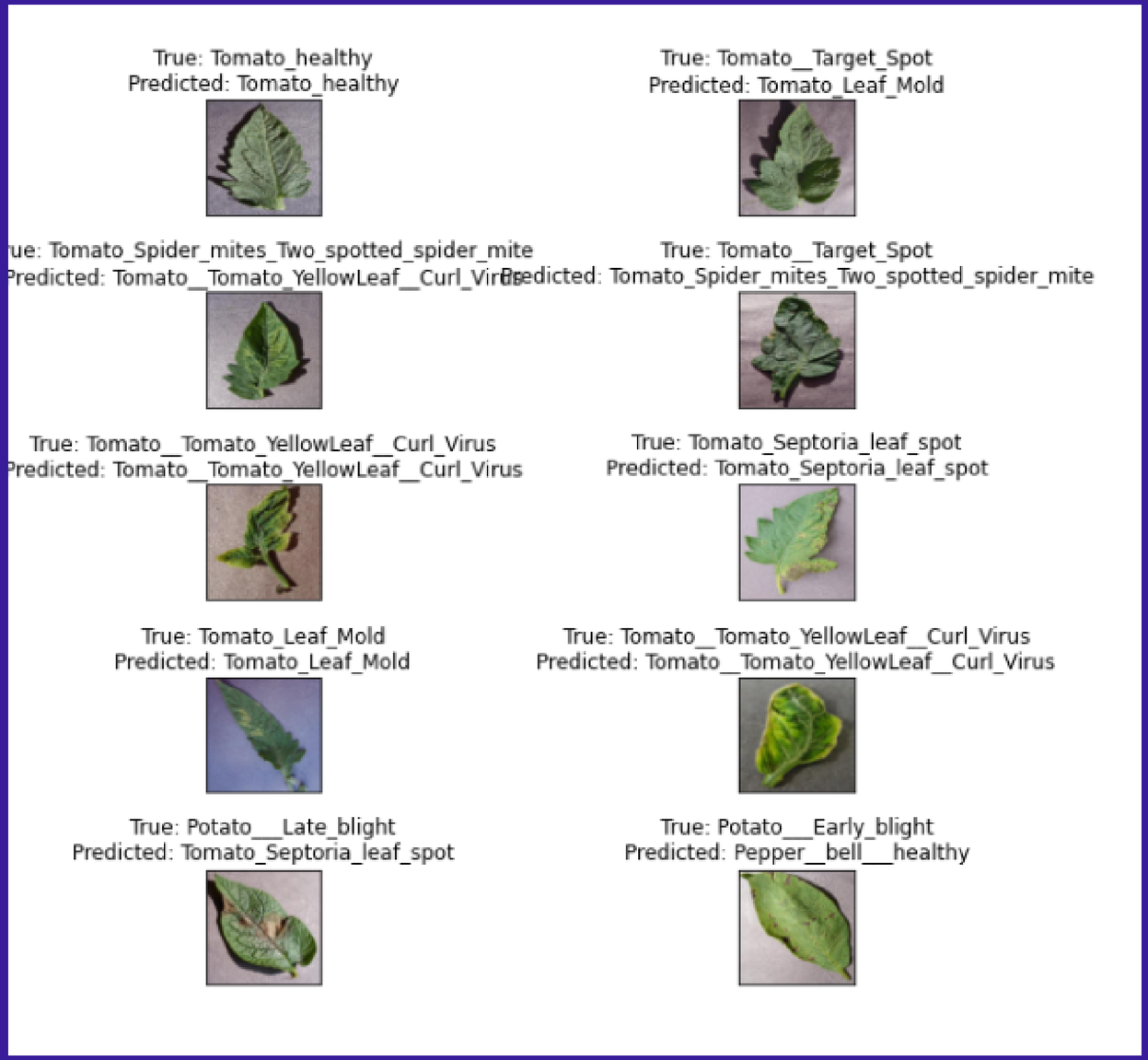
```
base_model=InceptionV3(input_shape=(100,100,3),include_top=False,weights='imagenet')
N_Class=15
x=base_model.output
x=GlobalAveragePooling2D()(x)

for layer in base_model.layers:
    layer.trainable=False

x=Flatten()(x)
x=Dense(1024,activation='relu')(x)
x=BatchNormalization()(x)
x=Dense(512,activation='relu')(x)
x=Dropout(0.25)(x)
predictions=Dense(N_Class,activation='softmax')(x)
google_net=Model(base_model.input,predictions)
```

Google Net (Inception V3)



loss	:	0.65
Accuracy	:	0.78
validation loss	:	0.80
validation accuracy	:	0.71

Google Net Model Predictions



ResNet

ResNet, short for Residual Networks is a classic neural network

Skip Connnections technique is introduced in ResNet Model.

Concepts behind ResNet allowed us train models with Deep Network without failing

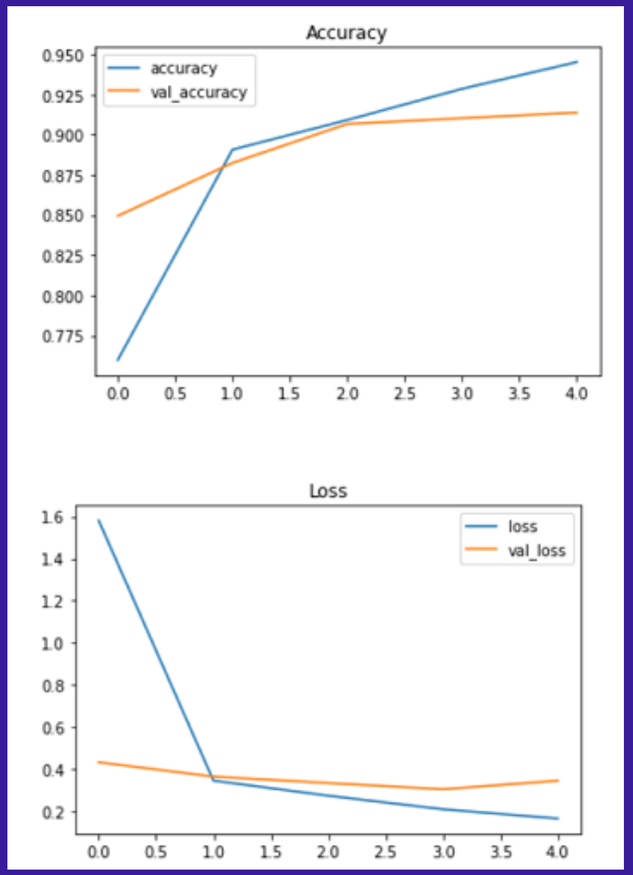
It helped to overcome the problem of vanishing gradient.

ResNet and Metrics

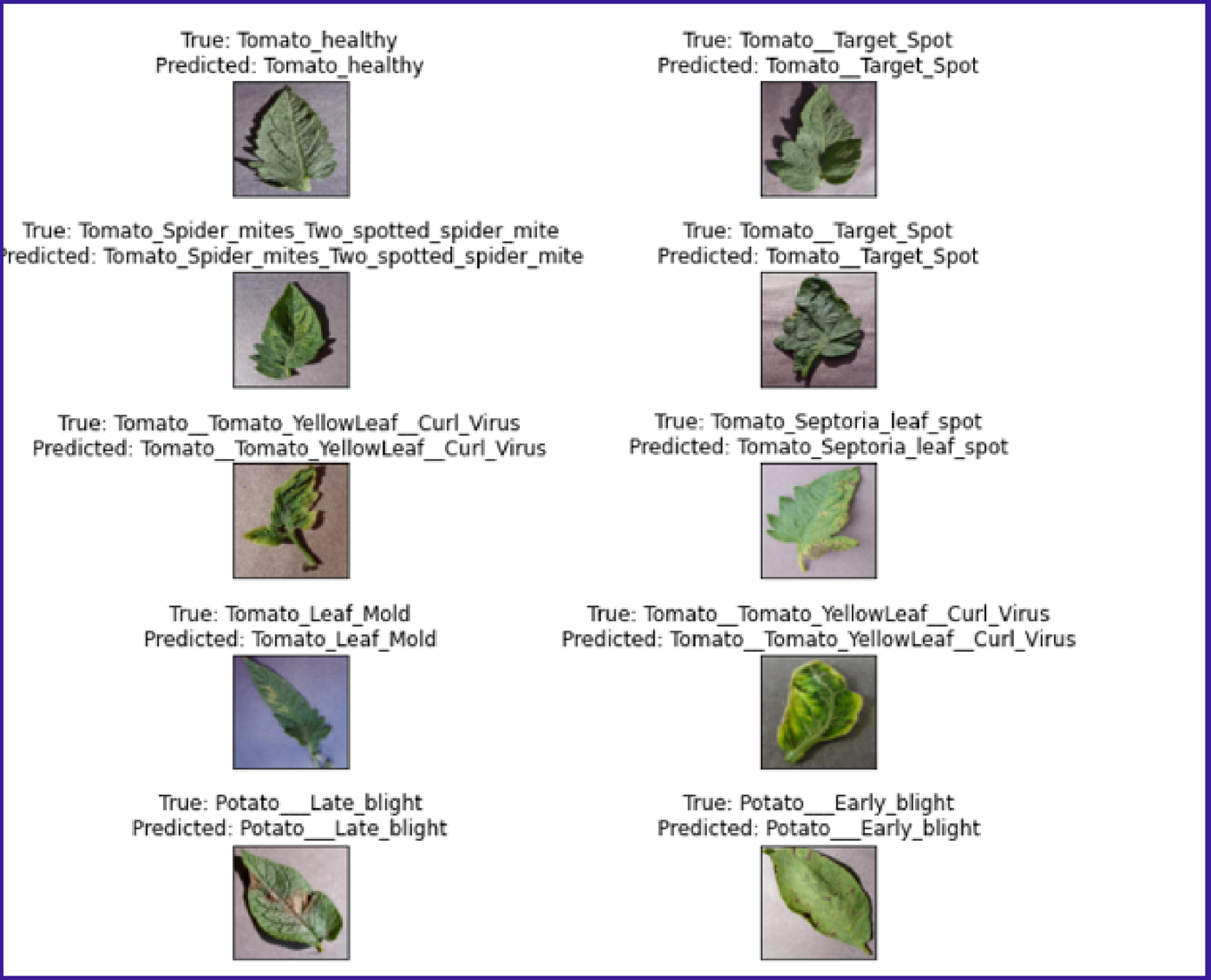
```
base_model=ResNet50(input_shape=(100,100,3),include_top=False,weights='imagenet')

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/re
snet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 0s 0us/step
94781440/94765736 [==============================] - 0s 0us/step
```

```
x=base_model.output
x=Flatten()(x)
x=Dense(512,activation='relu')(x)
x=Dropout(0.2)(x)
x=Dense(15,activation='softmax')(x)
resnet=Model(base_model.input,x)
```

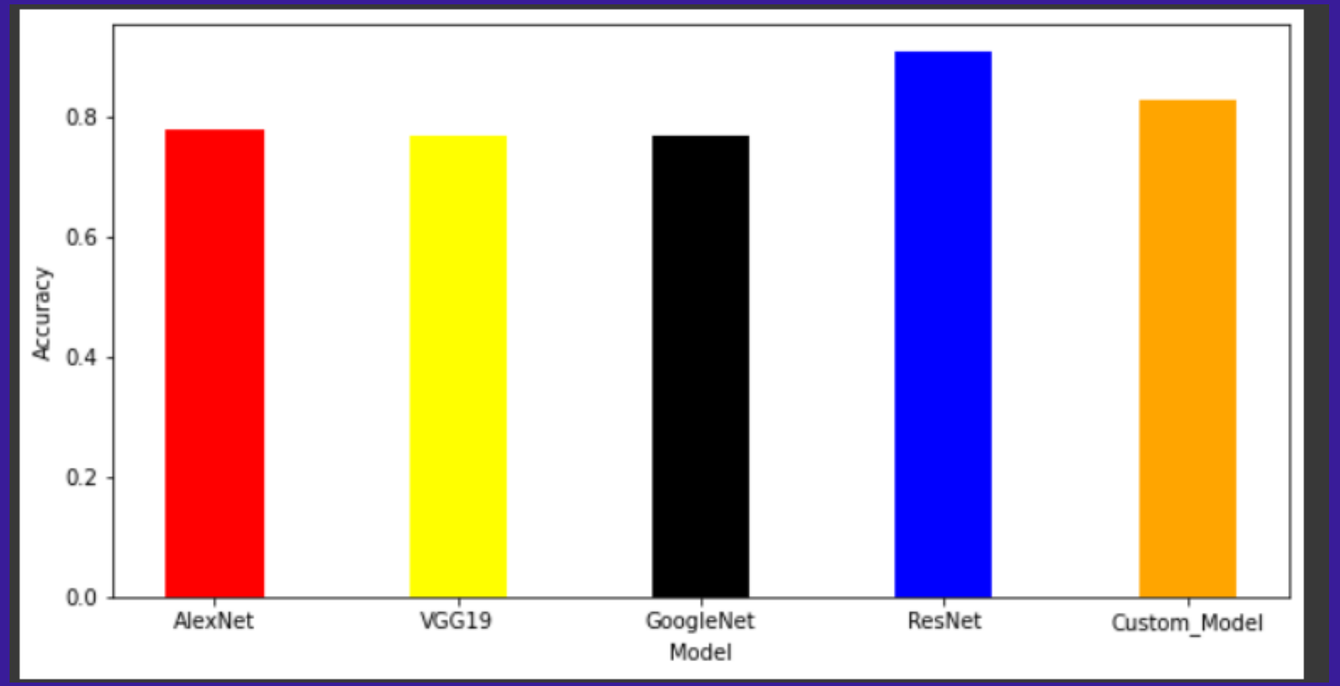


ResNet Model Predictions



Comparison Of Models

	Model	Model Accuracy
1	Alex_Net	0.78
2	VGG_19	0.78
3	Goog_Net	0.77
4	Res_Net	0.91
5	Custom_Model	0.83



Conclusion

- Implemented Deep Learning Model for detection of plant diseases
- Out of the 5 models implemented
    - ResNet Model performed best compared to other

Thank You