ARTIFICIAL INTELLIGENCE FOR APPLE BLACK ROT DISEASE IDENTIFICATION USING IMAGE PROCESSING

GROUP MEMBERS NAME : KARAN KHARKAR (20ET5001)
SHREESHAILYA MALI (19ET1139)
BHUSHAN MAHAJAN (19ET1105)
BATCH : B

Guide : Prajakta Papalkar

ABSTRACT

- This report holds a survey on fruit disease detection using image processing technique. Digital image processing is fast and accurate technique for detection of diseases in fruits. Identification and classification of diseases of fruits are done through various algorithms. Techniques include clustering and color based segmentation, artificial neural network and different classifiers based classification of diseases. The main focus of our work is obtaining the analysis of different fruit diseases detection techniques and also provides an overview of these techniques.

INTRODUCTION OF APPLE BLACK ROT, FROG EYE LEAF (Brown Spots)

- Black rot is an important disease of apple caused by the fungus Botryosphaeria obtusa.
- Black rot and frogeye leaf spot are phases of a widespread and damaging disease of apple.
- The fruit rot phase is called black rot and on the leaf it is called frogeye leaf spot.

SYMPTOMS

Leaf Symptoms

- “Frog-eye leaf ”are circular spots with brown or reddish edges and light tan interiors.
- Leaf infections result in a disease called frog-eye leaf spot.
- On leaves, the disease first appears as a tiny purple fleck which eventually enlarges into a circular lesion about 4-5 mm in diameter.
- The disease often first shows up one to three weeks after petal fall.
- The optimum temperature for leaf infections is around 26.6°C with 4.5 hours of leaf wetness.

Fruit symptoms

- Brown to black concentric rings can often be seen on larger infections.
- The flesh of the apple is brown but remains firm.
- Small, black spots can be seen on older fruit infections. These are fungal spore producing structures, called pycnidia.
- Some fruit dry out and remain attached to the tree.

Dataset (Program)

```
In [1]: import numpy as np
        import pandas as pd

        import matplotlib.pyplot as plt
        import seaborn as sns

        from tensorflow import keras
        from tensorflow.keras.preprocessing.image import ImageDataGenerator as Imgen
        from tensorflow.keras.models import Model,Sequential
        from tensorflow.keras.layers import Input,Conv2D,MaxPooling2D,Dropout,Flatten,Dense,GlobalAveragePooling2D,BatchNormalization
        from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint

        from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
        from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
        from tensorflow.keras.applications.densenet import DenseNet201
        import requests

        from sklearn.metrics import confusion_matrix,classification_report
```

```
In [2]: traingen = Imgen(preprocessing_function=preprocess_input,

                        shear_range = 0.2,
                        zoom_range = 0.2,
                        width_shift_range = 0.2,
                        height_shift_range = 0.2,
                        fill_mode="nearest",
                        validation_split=0.15)


        testgen = Imgen(preprocessing_function=preprocess_input)
```

```
In [3]: trainds = traingen.flow_from_directory("D:\Apple\Apple_black_rot\Train",
                                              target_size=(150,150),
                                               class_mode="categorical",
                                               seed=123,
                                               batch_size=32,
                                               subset="training"
                                              )
        valds = traingen.flow_from_directory("D:\Apple\Apple_black_rot\Train",
                                              target_size=(150,150)
```

```
valds = traingen.flow_from_directory("D:\Apple\Apple_black_rot\Train",
                                      target_size=(150,150),
                                       class_mode="categorical",
                                       seed=123,
                                       batch_size=32,
                                    subset="validation"
                                      )
testds = testgen.flow_from_directory("D:\Apple\Apple_black_rot\Valid",
                                    target_size=(150,150),
                                    class_mode="categorical",
                                    seed=123,
                                    batch_size=32,
                                    shuffle=False)
```

```
Found 3396 images belonging to 2 classes.
Found 599 images belonging to 2 classes.
Found 999 images belonging to 2 classes.
```

In [4]:

```
c = trainds.class_indices
classes = list(c.keys())
classes
```

Out[4]:

```
['Apple___healthy', 'Apple_black_rot']
```

In [ ]:

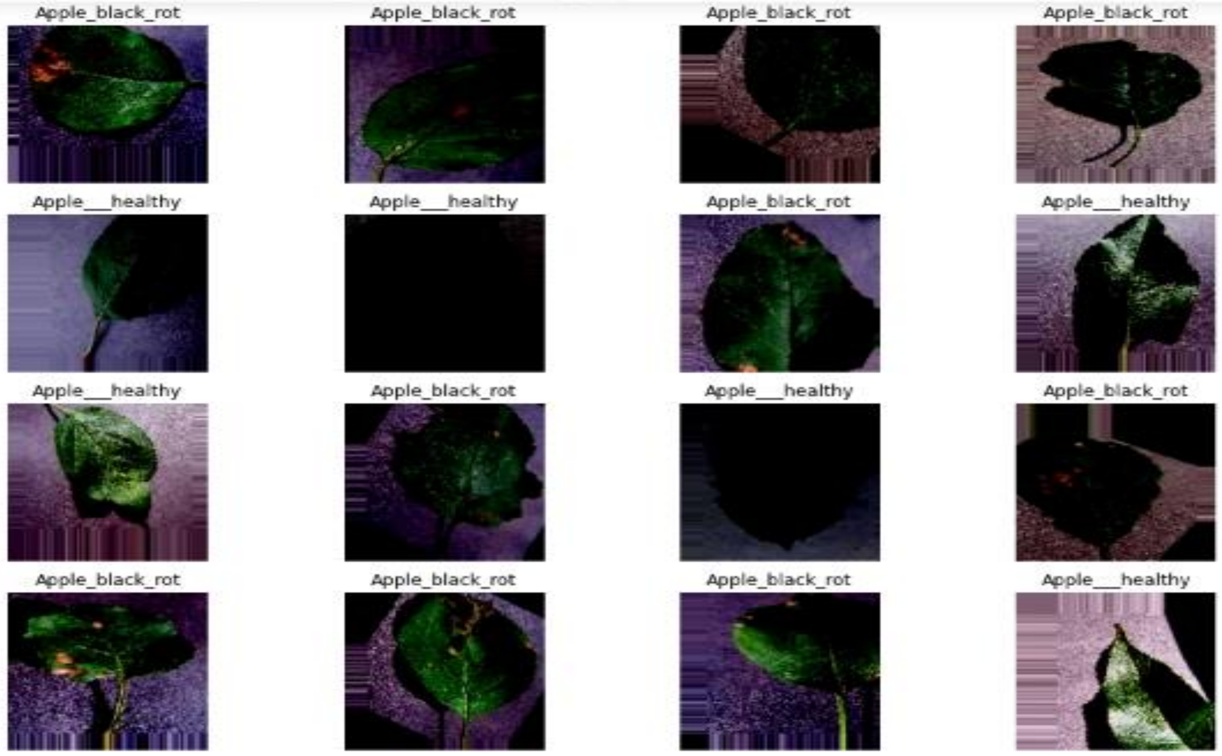
In [5]:

```
x,y = next(trainds)
def plotImages(x,y):
    plt.figure(figsize=[15,11])
    for i in range(16):
        plt.subplot(4,4,i+1)
        plt.imshow(x[i])
        plt.title(classes[np.argmax(y[i])])
        plt.axis("off")
    plt.show()
```

In [6]:

```
plotImages(x,y)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```




```
In [7]: resp = requests.get('http://www.google.com')
        base_model = DenseNet201(include_top=False,
                                 input_shape=(150,150,3),
                                  weights = "imagenet",
                                  pooling="avg"
```

```
In [8]: image_input = Input(shape=(150,150,3))

        x = base_model(image_input,training = False)

        x = Dense(512,activation = "relu")(x)

        x = Dropout(0.3)(x)

        x = Dense(128,activation = "relu")(x)

        image_output = Dense(2,activation="softmax")(x)

        model = Model(image_input,image_output)
```

```
In [9]: model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
densenet201 (Functional)     (None, 1920)              18321984
_________________________________________________________________
dense (Dense)                (None, 512)               983552
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 19,371,458
Trainable params: 1,049,474
Non-trainable params: 18,321,984
_________________________________________________________________
```

```
In [10]: model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy"])
```

```
In [11]: my_calls = [EarlyStopping(monitor="val_accuracy",patience=3),
                     ModelCheckpoint("Model.h5",verbose= 1 ,save_best_only=True)]
```

```
In [12]: hist = model.fit(trainds,epochs=5,validation_data=valds,callbacks=my_calls)
```

```
107/107 [==============================] - 220s 2s/step - loss: 0.1264 - accuracy: 0.9517 - val_loss: 0.0246 - val_accuracy: 0.
9933

Epoch 00001: val_loss improved from inf to 0.02456, saving model to Model.h5
Epoch 2/5
107/107 [==============================] - 201s 2s/step - loss: 0.0437 - accuracy: 0.9867 - val_loss: 0.0192 - val_accuracy: 0.
9883

Epoch 00002: val_loss improved from 0.02456 to 0.01922, saving model to Model.h5
Epoch 3/5
107/107 [==============================] - 194s 2s/step - loss: 0.0328 - accuracy: 0.9882 - val_loss: 0.0211 - val_accuracy: 0.
9917

Epoch 00003: val_loss did not improve from 0.01922
Epoch 4/5
107/107 [==============================] - 189s 2s/step - loss: 0.0335 - accuracy: 0.9879 - val_loss: 0.0199 - val_accuracy: 0.
9933

Epoch 00004: val_loss did not improve from 0.01922
```

In [13]:
```
model.evaluate(testds)
```

```
32/32 [==============================] - 46s 1s/step - loss: 0.0135 - accuracy: 0.9950
```

Out[13]:
```
[0.013549539260566235, 0.9949949979782104]
```

In [14]:
```
plt.figure(figsize=(15,6))

plt.subplot(1,2,1)
plt.plot(hist.epoch,hist.history['accuracy'],label = 'Training')
plt.plot(hist.epoch,hist.history['val_accuracy'],label = 'validation')

plt.title("Accuracy")
plt.legend()

plt.subplot(1,2,2)
plt.plot(hist.epoch,hist.history['loss'],label = 'Training')
plt.plot(hist.epoch,hist.history['val_loss'],label = 'validation')

plt.title("Loss")
plt.legend()
```

In [8]:
```
image_input = Input(shape=(150,150,3))

x = base_model(image_input,training = False)

x = Dense(512,activation = "relu")(x)

x = Dropout(0.3)(x)

x = Dense(128,activation = "relu")(x)

image_output = Dense(2,activation="softmax")(x)

model = Model(image_input,image_output)
```

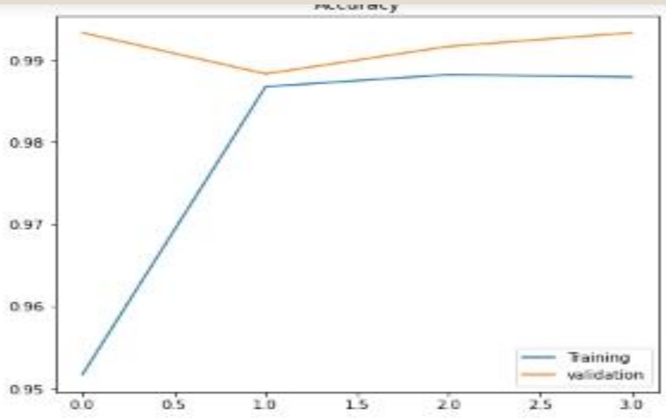
In [9]:
```
model.summary()
```

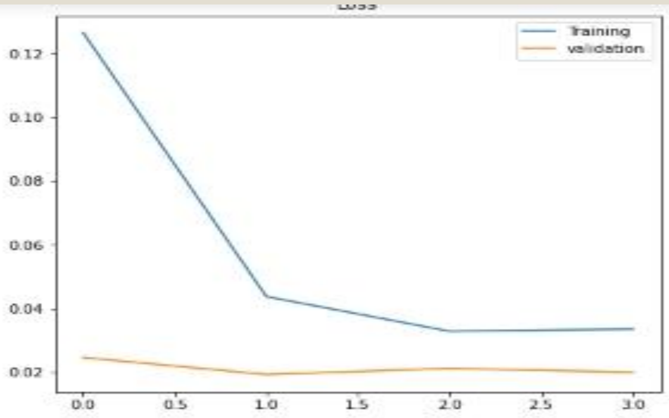
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
densenet201 (Functional)     (None, 1920)              18321984
_________________________________________________________________
dense (Dense)                (None, 512)               983552
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 19,371,458
Trainable params: 1,049,474
Non-trainable params: 18,321,984
_________________________________________________________________
```

In [10]:
```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy"])
```

In [11]:
```
my_calls = [EarlyStopping(monitor="val_accuracy",patience=3),
            ModelCheckpoint("Model.h5",verbose= 1 ,save_best_only=True)]
```

In [12]:
```
hist = model.fit(trainds,epochs=5,validation_data=valds,callbacks=my_calls)
```





In [15]:
```
pred = model.predict(testds)
```

In [16]:
```
pred = [np.argmax(i) for i in pred]
```

In [17]:
```
y_test = testds.classes
```

In [18]:
```
print(classification_report(pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       497
           1       1.00      0.99      0.99       502

    accuracy                           0.99       999
   macro avg       1.00      1.00      0.99       999
weighted avg       1.00      0.99      0.99       999
```

In [19]:
```
print(confusion_matrix(pred,y_test))
```

```
[[497   0]
 [  5 497]]
```

Conclusion

- This report gives the survey on fruit diseases detection and classification techniques by using image processing.
- The report discusses the methodology, results in each of the research work and future research directions.
- Different researchers used algorithms for image segmentation, feature extraction, training and classification of fruit disease.
- Among different methods, K-means clustering and SVM provides high accuracy and are widely used. All methods in this report provide efficient results and also save time.

Reference

- https://www.google.com/search?q=apple+black+rot&oq=apple&aqs=chrome.1.69i59l2j46i20i131i199i263i291i433i512j0i273j69i60.2206j0j9&client=ms-android-oppo-rvo3&sourceid=chrome-mobile&ie=UTF-8
- Kaggleandtensorflow.org
- http://www.omafra.gov.on.ca/english/crops/facts/blackrot.html
- httpss://www.google.com/search?q=apple+black+rot+disease+identification+using+image+processing&client=ms-android-oppo-rvo3&sxsrf=ALeKk03nLxPXhZpzEcYg_rQUCEhXbKIFgg%3A1628155566608&ei=rq4LYdHfJP7Q1sQPt_iBmA4&oq=apple+black+rot+di&gs_lcp=ChNtb2JpbGUtZ3dzLXdpei1zZXJwEAEYADIECCMQJzIFCAAQgAQyBggAEBYQHjIFCCEQoAEyBQghEKABMgUIIRCgATIFCCEQoAEyBQghEKABOgQIABBHOgoIABCABBCHAhAUOgQIIRAVUNYcWJElYPgyaABwAXgAgAGiAogB_gWSAQUwLjEuMpgBAKABAcgBCMABAQ&sclient=mobile-gws-wiz-serp

THE END