```
import datetime
cur_date=datetime.datetime.now()
print("current date and time",cur_date)
```

```
current date and time 2024-02-18 18:22:30.148991
```

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications import ResNet50V2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from keras.models import Sequential

from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

import matplotlib.pyplot as plt
import numpy as np
import random
import cv2
import os

from imutils import paths

imagePaths = sorted(list(paths.list_images("F:\\Paper6_vit+cnn\\
Groundnut_Leaf_dataset\\train1")))
#D:\G_Dataset_final\Train
# random shuffle
random.seed(42)
random.shuffle(imagePaths)

data = []
labels = []
image_dims = (224, 224, 3)# 212 for xception model

for imagePath in imagePaths:
    image = cv2.imread(imagePath)
    image = cv2.resize(image, (image_dims[1], image_dims[0]))
    image = img_to_array(image)
    data.append(image)
    l = label = imagePath.split(os.path.sep)[-2].split("_")
    labels.append(l)

data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)
print("{} images ({:.2f}MB)".format(len(imagePaths), data.nbytes /
(1024 * 1000.0)))
```

```
7910 images (9302.16MB)
```

```
data = np.array(data)
label = np.array(labels)
print(data.shape)
```

```
(7910, 224, 224, 3)
```

```
mlb = MultiLabelBinarizer()
labels = mlb.fit_transform(labels)
# total 4 labels
print("class labels:")
for (i, label) in enumerate(mlb.classes_):
    print("{}. {}".format(i + 1, label))
```

```
class labels:
1. ELS
2. ER
3. HL
4. LLS
5. ND
6. RUST
```

```
trainX, testX, trainY, testY = train_test_split(data, labels,
test_size=0.40) # (0.20)
```

```
from keras.applications import ResNet50V2 #InceptionResNetV2
NASNetMobile InceptionV3 VGG19
inc=ResNet50V2(input_shape=(224,224,3),weights='imagenet',include_top=
False)
```

```
for i in inc.layers:
    i.trainable=False
```

```
from tensorflow.keras import layers
x=Flatten()(inc.output)
#x=layers.Dropout(0.2)(x)
```

```
pred=Dense(6,activation='softmax')(x)
```

```
from keras.models import Model
model=Model(inputs=inc.input,outputs=pred)
```

```
#model.summary()
```

```
from tensorflow import keras
opt = keras.optimizers.Adam(learning_rate=0.1)
model.compile(optimizer=opt,loss='categorical_crossentropy',metrics=['
accuracy'])
#model.compile(optimizer='adam',loss='categorical_crossentropy',metric
s=['accuracy'])
```

```
history=model.fit(trainX, trainY,
          batch_size=52,
          epochs=25,
          verbose=1,
          validation_data=(testX, testY))
```

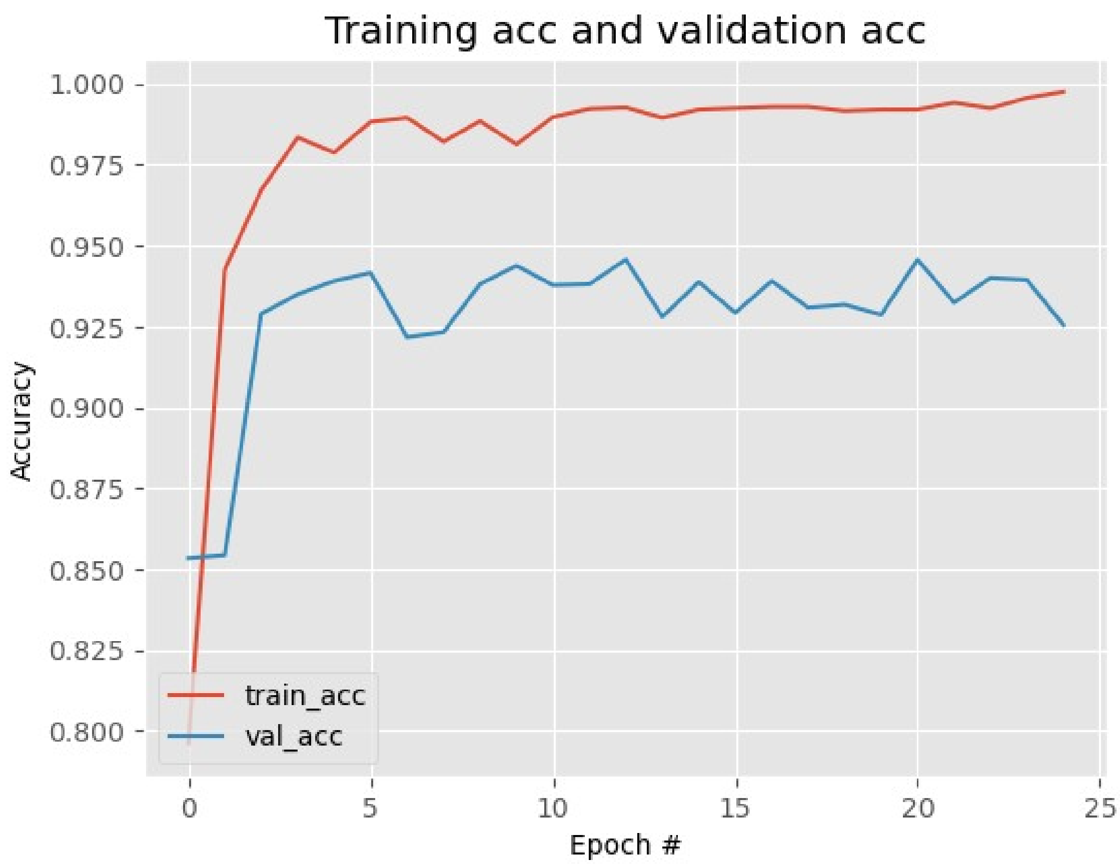
```
Epoch 1/25
92/92 [==============================] - 311s 3s/step - loss: 151.0061
- accuracy: 0.7960 - val_loss: 93.7447 - val_accuracy: 0.8534
Epoch 2/25
92/92 [==============================] - 292s 3s/step - loss: 27.6724
- accuracy: 0.9425 - val_loss: 107.6520 - val_accuracy: 0.8543
Epoch 3/25
92/92 [==============================] - 275s 3s/step - loss: 17.2647
- accuracy: 0.9671 - val_loss: 50.9723 - val_accuracy: 0.9289
Epoch 4/25
92/92 [==============================] - 274s 3s/step - loss: 7.3123 -
accuracy: 0.9834 - val_loss: 40.6326 - val_accuracy: 0.9349
Epoch 5/25
92/92 [==============================] - 274s 3s/step - loss: 12.3406
- accuracy: 0.9787 - val_loss: 49.0459 - val_accuracy: 0.9390
Epoch 6/25
92/92 [==============================] - 273s 3s/step - loss: 4.6845 -
accuracy: 0.9882 - val_loss: 48.0991 - val_accuracy: 0.9415
Epoch 7/25
92/92 [==============================] - 274s 3s/step - loss: 6.2735 -
accuracy: 0.9895 - val_loss: 73.8171 - val_accuracy: 0.9216
Epoch 8/25
92/92 [==============================] - 273s 3s/step - loss: 11.2403
- accuracy: 0.9821 - val_loss: 77.7963 - val_accuracy: 0.9232
Epoch 9/25
92/92 [==============================] - 273s 3s/step - loss: 5.9967 -
accuracy: 0.9884 - val_loss: 58.1889 - val_accuracy: 0.9381
Epoch 10/25
92/92 [==============================] - 273s 3s/step - loss: 12.5210
- accuracy: 0.9812 - val_loss: 64.6022 - val_accuracy: 0.9437
Epoch 11/25
92/92 [==============================] - 273s 3s/step - loss: 7.6821 -
accuracy: 0.9897 - val_loss: 67.8581 - val_accuracy: 0.9377
Epoch 12/25
92/92 [==============================] - 273s 3s/step - loss: 6.4005 -
accuracy: 0.9922 - val_loss: 85.9565 - val_accuracy: 0.9381
Epoch 13/25
92/92 [==============================] - 273s 3s/step - loss: 6.0932 -
accuracy: 0.9926 - val_loss: 67.6839 - val_accuracy: 0.9456
Epoch 14/25
92/92 [==============================] - 273s 3s/step - loss: 7.4764 -
accuracy: 0.9895 - val_loss: 114.9252 - val_accuracy: 0.9279
Epoch 15/25
92/92 [==============================] - 273s 3s/step - loss: 7.7026 -
```

```
accuracy: 0.9920 - val_loss: 93.1443 - val_accuracy: 0.9387
Epoch 16/25
92/92 [==============================] - 276s 3s/step - loss: 8.5866 -
accuracy: 0.9924 - val_loss: 90.7842 - val_accuracy: 0.9292
Epoch 17/25
92/92 [==============================] - 274s 3s/step - loss: 4.9188 -
accuracy: 0.9928 - val_loss: 82.9417 - val_accuracy: 0.9390
Epoch 18/25
92/92 [==============================] - 275s 3s/step - loss: 5.0494 -
accuracy: 0.9928 - val_loss: 95.4457 - val_accuracy: 0.9308
Epoch 19/25
92/92 [==============================] - 275s 3s/step - loss: 7.2624 -
accuracy: 0.9916 - val_loss: 100.8819 - val_accuracy: 0.9317
Epoch 20/25
92/92 [==============================] - 274s 3s/step - loss: 8.1351 -
accuracy: 0.9920 - val_loss: 111.1013 - val_accuracy: 0.9286
Epoch 21/25
92/92 [==============================] - 274s 3s/step - loss: 9.8789 -
accuracy: 0.9920 - val_loss: 90.4479 - val_accuracy: 0.9456
Epoch 22/25
92/92 [==============================] - 275s 3s/step - loss: 5.1206 -
accuracy: 0.9941 - val_loss: 126.2789 - val_accuracy: 0.9324
Epoch 23/25
92/92 [==============================] - 275s 3s/step - loss: 10.7417
- accuracy: 0.9924 - val_loss: 102.6714 - val_accuracy: 0.9399
Epoch 24/25
92/92 [==============================] - 275s 3s/step - loss: 5.9879 -
accuracy: 0.9956 - val_loss: 100.8897 - val_accuracy: 0.9393
Epoch 25/25
92/92 [==============================] - 274s 3s/step - loss: 5.2674 -
accuracy: 0.9975 - val_loss: 125.2883 - val_accuracy: 0.9254

print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=16)

# for each image in the testing set we need to find the index of the
label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1),
predIdxs,target_names=mlb.classes_))

[INFO] evaluating network...
198/198 [==============================] - 119s 598ms/step
              precision    recall  f1-score   support

         ELS       0.94      0.82      0.87       517
          ER       1.00      0.99      0.99       433
          HL       0.94      0.91      0.93       597
```

```
         LLS       0.94      0.90      0.92       613
          ND       0.83      0.95      0.89       502
        RUST       0.93      1.00      0.96       502

    accuracy                           0.93      3164
   macro avg       0.93      0.93      0.93      3164
weighted avg       0.93      0.93      0.93      3164
```

```
N = 25
plt.style.use("ggplot")
plt.figure()
#plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
#plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), history.history["accuracy"],
label="train_acc")
plt.plot(np.arange(0, N), history.history["val_accuracy"],
label="val_acc")
plt.title("Training acc and validation acc")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="lower left")
```

```
<matplotlib.legend.Legend at 0x1c226a6f100>
```




```
N = 25
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"],
label="val_loss")
#plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
#plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and validation loss")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="lower left")
```

```
<matplotlib.legend.Legend at 0x1c226b8c490>
```


```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(testY.argmax(axis=1), predIdxs))
```

```
[[423   0  11  35  35  13]
 [  0 428   0   0   0   5]
 [  0   0 544   0  53   0]
 [ 27   1   2 553  10  20]
 [  0   0  21   1 479   1]
 [  0   0   0   1   0 501]]
```

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
precision_score(testY.argmax(axis=1), predIdxs, average='micro')
```

```
0.9254108723135271
```

```
recall = recall_score(testY.argmax(axis=1), predIdxs, average='macro')
print('recall=',recall)
```

```
recall= 0.9286948679136069
```

```
from sklearn.metrics import f1_score
f1_score(testY.argmax(axis=1), predIdxs, average='micro')
```

```
0.9254108723135271
```

```
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import balanced_accuracy_score
a=matthews_corrcoef(testY.argmax(axis=1), predIdxs)
b=balanced_accuracy_score(testY.argmax(axis=1), predIdxs)
print(a)
print(b)
```

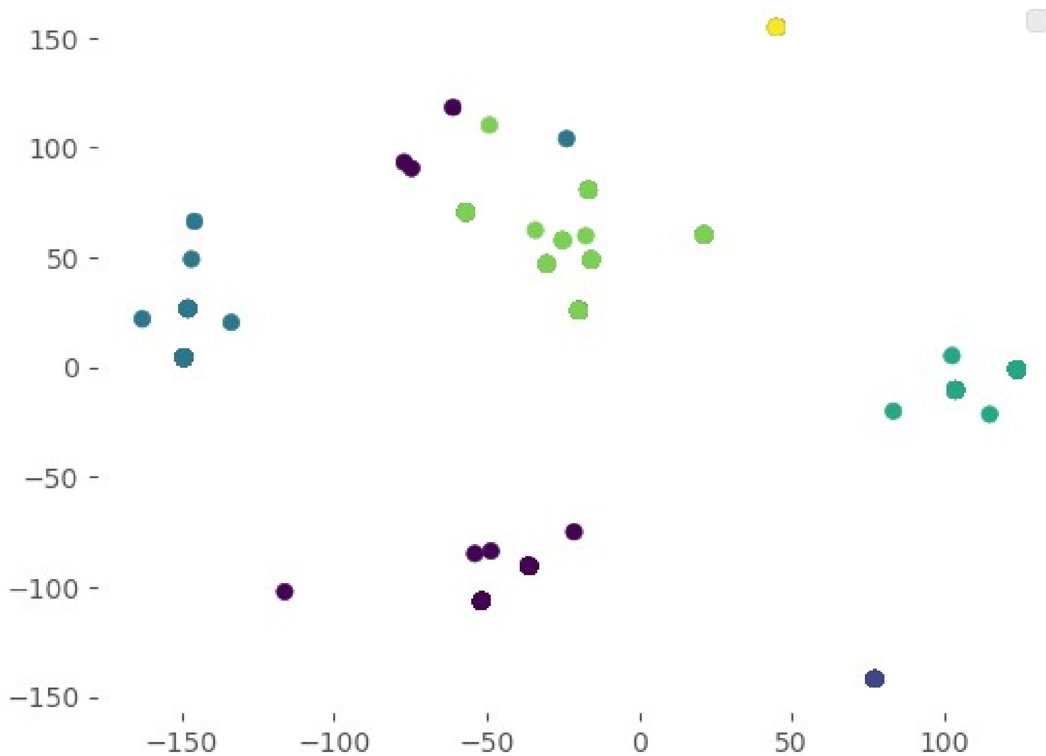
```
0.9109879928149365
0.9286948679136069
```

```
from sklearn.manifold import TSNE
# Extract the features from the last layer of the CNN
features = model.predict(trainX)
# Reduce the dimensionality of the features using t-SNE
tsne = TSNE(n_components=2, perplexity=30.0, early_exaggeration=12.0,
learning_rate=200.0, n_iter=1000, n_iter_without_progress=300,
min_grad_norm=1e-07, metric='euclidean', metric_params=None,
init='pca', verbose=0, random_state=None, method='barnes_hut',
angle=0.5, n_jobs=None)
features_embedded = tsne.fit_transform(features)
```

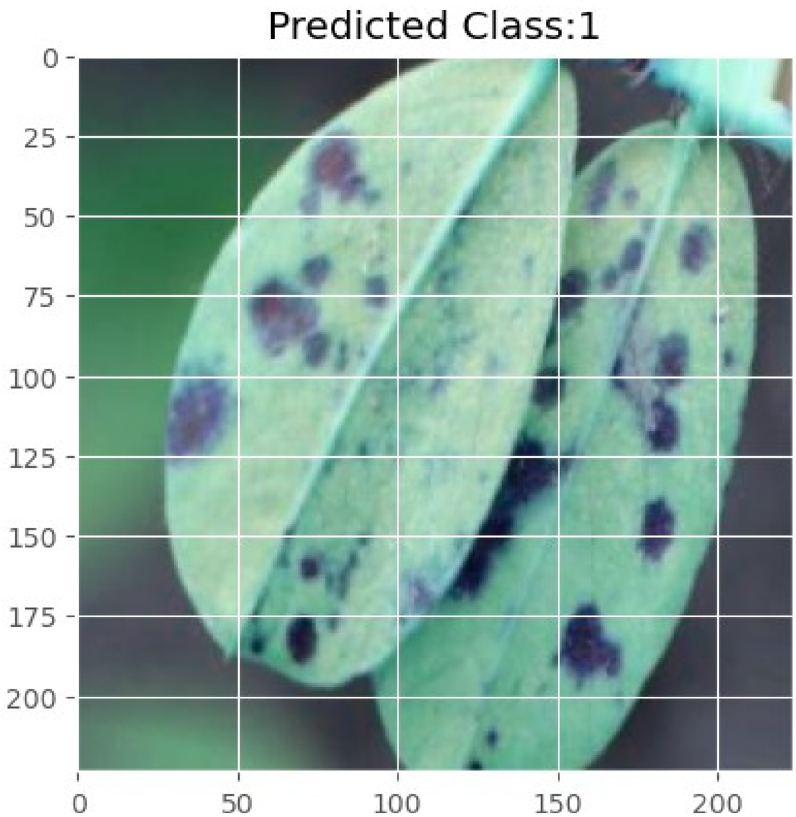
```
149/149 [==============================] - 178s 1s/step
```

```
# Plot the results
ax=plt.axes()
ax.set_facecolor("white")
plt.scatter(features_embedded[:, 0], features_embedded[:, 1],
c=trainY.argmax(axis=1))

#plt.axis('off')
plt.legend()
plt.grid(False)
#plt.savefig('F:/Paper6_vit+cnn/results/updated_results/P6_ResNet50V2_
TSNE.pdf',dpi=600)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```




```
import lime
from lime import lime_image

# Choose an image from the test set for explanation
image_index = 5
img = testX[image_index]
img_array = np.expand_dims(img, axis=0)

# Make predictions
predictions = model.predict(img_array)
predicted_class = np.argmax(predictions)

# Display the original image and predictions
plt.imshow(img.reshape(224, 224,3))
plt.title(f'Predicted Class:{predicted_class}')
plt.show()

1/1 [==============================] - 0s 76ms/step
```




```
# Create a LIME explainer for image classification
explainer = lime_image.LimeImageExplainer()

# Explain predictions
explanation = explainer.explain_instance(img_array[0], model.predict,
top_labels=1, hide_color=0, num_samples=1000)
```

{"model_id":"4eb4254093c74aa7b2de1f2f662d4478","version_major":2,"version_minor":0}

```
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 366ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 353ms/step
1/1 [==============================] - 0s 344ms/step
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 343ms/step
1/1 [==============================] - 0s 358ms/step
1/1 [==============================] - 0s 366ms/step
1/1 [==============================] - 0s 373ms/step
1/1 [==============================] - 0s 368ms/step
1/1 [==============================] - 0s 364ms/step
```

```
1/1 [==============================] - 0s 363ms/step
1/1 [==============================] - 0s 375ms/step
1/1 [==============================] - 0s 377ms/step
1/1 [==============================] - 0s 373ms/step
1/1 [==============================] - 0s 362ms/step
1/1 [==============================] - 0s 366ms/step
1/1 [==============================] - 0s 377ms/step
1/1 [==============================] - 0s 369ms/step
1/1 [==============================] - 0s 375ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 344ms/step
1/1 [==============================] - 0s 347ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 346ms/step
1/1 [==============================] - 0s 349ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 349ms/step
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 343ms/step
1/1 [==============================] - 0s 342ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 344ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 352ms/step
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 361ms/step
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 350ms/step
1/1 [==============================] - 0s 352ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 350ms/step
1/1 [==============================] - 0s 367ms/step
1/1 [==============================] - 0s 373ms/step
1/1 [==============================] - 0s 368ms/step
1/1 [==============================] - 0s 378ms/step
1/1 [==============================] - 0s 371ms/step
1/1 [==============================] - 0s 363ms/step
1/1 [==============================] - 0s 377ms/step
1/1 [==============================] - 0s 371ms/step
1/1 [==============================] - 0s 369ms/step
1/1 [==============================] - 0s 369ms/step
1/1 [==============================] - 0s 373ms/step
1/1 [==============================] - 0s 388ms/step
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 351ms/step
```

```
1/1 [==============================] - 0s 362ms/step
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 358ms/step
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 362ms/step
1/1 [==============================] - 0s 349ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 354ms/step
1/1 [==============================] - 0s 353ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 396ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 350ms/step
1/1 [==============================] - 0s 344ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 350ms/step
1/1 [==============================] - 0s 347ms/step
1/1 [==============================] - 0s 360ms/step
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 355ms/step
1/1 [==============================] - 0s 364ms/step
1/1 [==============================] - 0s 387ms/step
1/1 [==============================] - 0s 361ms/step
1/1 [==============================] - 0s 364ms/step
1/1 [==============================] - 0s 376ms/step
1/1 [==============================] - 0s 369ms/step
1/1 [==============================] - 0s 391ms/step
1/1 [==============================] - 0s 371ms/step
1/1 [==============================] - 0s 378ms/step
1/1 [==============================] - 0s 403ms/step
1/1 [==============================] - 0s 368ms/step
1/1 [==============================] - 0s 422ms/step
1/1 [==============================] - 0s 366ms/step
1/1 [==============================] - 0s 360ms/step
1/1 [==============================] - 0s 353ms/step
```

```python
# explain predictions
explanation=explainer.explain_instance(img_array[0],model.predict,top_labels=1,hide_color=0,num_samples=1000)
```

{"model_id":"7fdf03e3f24741f6bd7ad9024608feba","version_major":2,"version_minor":0}

```
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 383ms/step
1/1 [==============================] - 0s 371ms/step
1/1 [==============================] - 0s 407ms/step
1/1 [==============================] - 0s 363ms/step
```

```
1/1 [==============================] - 0s 360ms/step
1/1 [==============================] - 0s 360ms/step
1/1 [==============================] - 0s 353ms/step
1/1 [==============================] - 0s 364ms/step
1/1 [==============================] - 0s 368ms/step
1/1 [==============================] - 0s 362ms/step
1/1 [==============================] - 0s 382ms/step
1/1 [==============================] - 0s 391ms/step
1/1 [==============================] - 0s 422ms/step
1/1 [==============================] - 0s 365ms/step
1/1 [==============================] - 0s 379ms/step
1/1 [==============================] - 0s 385ms/step
1/1 [==============================] - 0s 385ms/step
1/1 [==============================] - 0s 381ms/step
1/1 [==============================] - 0s 394ms/step
1/1 [==============================] - 0s 372ms/step
1/1 [==============================] - 0s 406ms/step
1/1 [==============================] - 0s 375ms/step
1/1 [==============================] - 0s 377ms/step
1/1 [==============================] - 0s 385ms/step
1/1 [==============================] - 0s 363ms/step
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 395ms/step
1/1 [==============================] - 0s 382ms/step
1/1 [==============================] - 0s 380ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 395ms/step
1/1 [==============================] - 0s 358ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 350ms/step
1/1 [==============================] - 0s 376ms/step
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 364ms/step
1/1 [==============================] - 0s 376ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 382ms/step
1/1 [==============================] - 0s 346ms/step
1/1 [==============================] - 0s 362ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 353ms/step
1/1 [==============================] - 0s 361ms/step
1/1 [==============================] - 0s 366ms/step
1/1 [==============================] - 0s 381ms/step
1/1 [==============================] - 0s 355ms/step
1/1 [==============================] - 0s 406ms/step
1/1 [==============================] - 0s 378ms/step
1/1 [==============================] - 0s 361ms/step
1/1 [==============================] - 0s 372ms/step
1/1 [==============================] - 0s 374ms/step
```

```
1/1 [==============================] - 0s 402ms/step
1/1 [==============================] - 0s 414ms/step
1/1 [==============================] - 0s 395ms/step
1/1 [==============================] - 0s 373ms/step
1/1 [==============================] - 0s 406ms/step
1/1 [==============================] - 0s 391ms/step
1/1 [==============================] - 0s 379ms/step
1/1 [==============================] - 0s 364ms/step
1/1 [==============================] - 0s 360ms/step
1/1 [==============================] - 0s 367ms/step
1/1 [==============================] - 0s 373ms/step
1/1 [==============================] - 0s 376ms/step
1/1 [==============================] - 0s 348ms/step
1/1 [==============================] - 0s 397ms/step
1/1 [==============================] - 0s 359ms/step
1/1 [==============================] - 0s 358ms/step
1/1 [==============================] - 0s 360ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 392ms/step
1/1 [==============================] - 0s 364ms/step
1/1 [==============================] - 0s 380ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 406ms/step
1/1 [==============================] - 0s 360ms/step
1/1 [==============================] - 0s 356ms/step
1/1 [==============================] - 0s 351ms/step
1/1 [==============================] - 0s 364ms/step
1/1 [==============================] - 0s 358ms/step
1/1 [==============================] - 0s 378ms/step
1/1 [==============================] - 0s 406ms/step
1/1 [==============================] - 0s 358ms/step
1/1 [==============================] - 0s 397ms/step
1/1 [==============================] - 0s 375ms/step
1/1 [==============================] - 0s 377ms/step
1/1 [==============================] - 0s 381ms/step
1/1 [==============================] - 0s 371ms/step
1/1 [==============================] - 0s 393ms/step
1/1 [==============================] - 0s 396ms/step
1/1 [==============================] - 0s 404ms/step
1/1 [==============================] - 0s 367ms/step
1/1 [==============================] - 0s 399ms/step
1/1 [==============================] - 0s 375ms/step
1/1 [==============================] - 0s 381ms/step
1/1 [==============================] - 0s 361ms/step
1/1 [==============================] - 0s 353ms/step
1/1 [==============================] - 0s 349ms/step
```

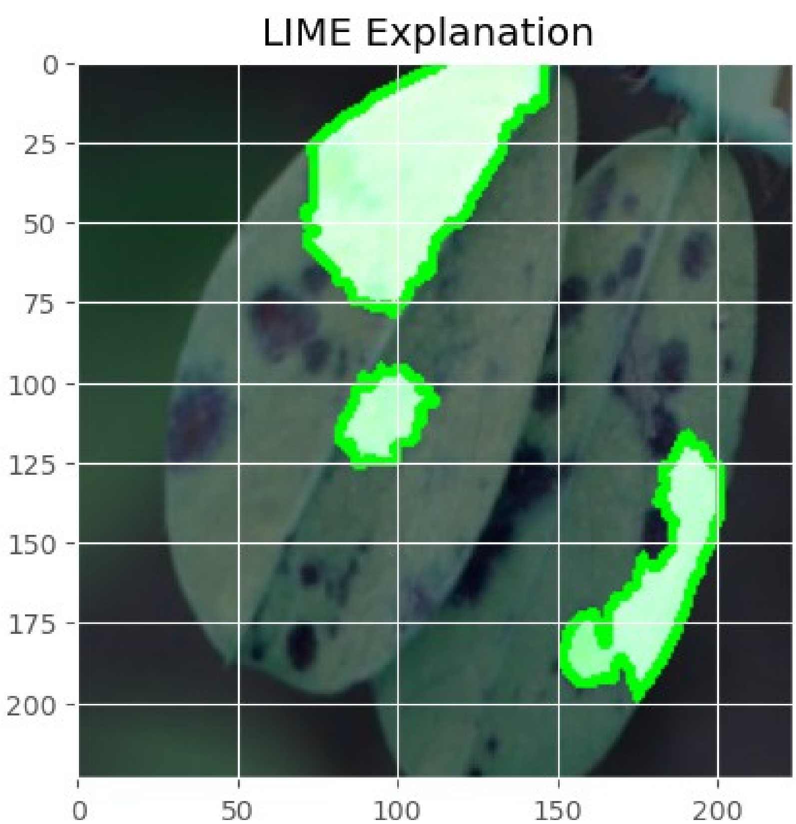
```python
#visualise LIME explanations with cv2.polylines
temp,mask=explanation.get_image_and_mask(explanation.top_labels[0],
positive_only=False,num_features=5,hide_rest=False)
```

```python
#convert temp to uint8 for cv2.polylines
temp=(temp/2+0.5*mask[:,:,np.newaxis])*255
temp=temp.astype(np.uint8)

#find contours in the mask
contours,_=cv2.findContours(mask.astype(np.uint8),cv2.RETR_EXTERNAL,cv
2.CHAIN_APPROX_SIMPLE)

#Draw contours on the image
cv2.polylines(temp,contours,isClosed=False,color=(0,255,0),thickness=2
)
```

```
array([[[27, 31, 34],
        [27, 31, 34],
        [27, 31, 34],
        ...,
        [74, 80, 68],
        [67, 70, 57],
        [69, 69, 56]],

       [[27, 31, 34],
        [27, 31, 34],
        [27, 31, 34],
        ...,
        [71, 77, 65],
        [66, 69, 56],
        [69, 70, 57]],

       [[27, 31, 35],
        [27, 31, 35],
        [27, 31, 35],
        ...,
        [68, 73, 61],
        [66, 68, 55],
        [72, 72, 59]],

       ...,

       [[46, 63, 55],
        [46, 63, 55],
        [46, 63, 55],
        ...,
        [41, 44, 48],
        [41, 43, 48],
        [41, 43, 48]],

       [[46, 63, 55],
        [46, 63, 55],
        [47, 63, 56],
        ...,
```

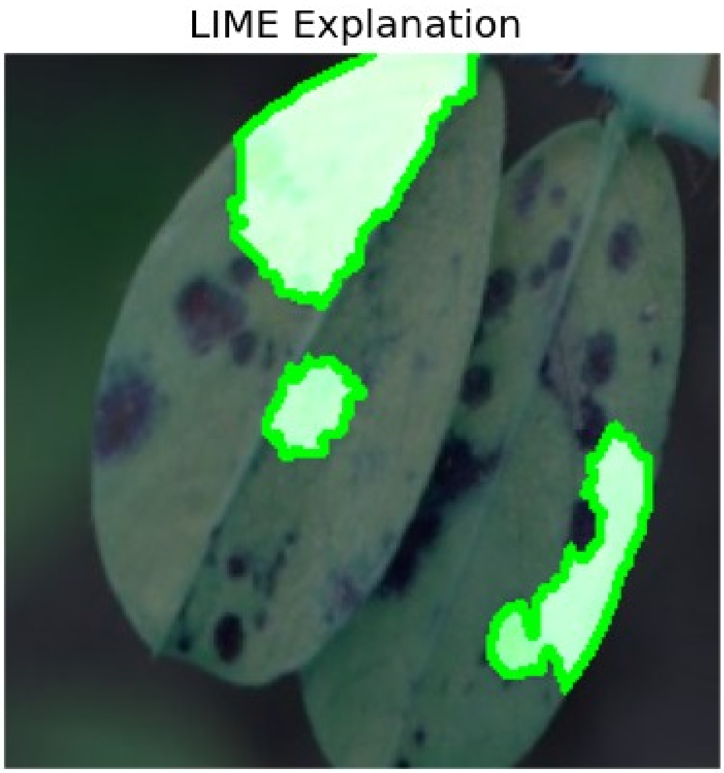
```
         [42, 44, 49],
         [42, 44, 49],
         [42, 44, 49]],

        [[46, 63, 55],
         [46, 63, 55],
         [47, 63, 56],
         ...,
         [43, 45, 50],
         [42, 45, 49],
         [42, 45, 49]]], dtype=uint8)
```

```python
#display the image
plt.imshow(temp)
plt.title('LIME Explanation ')
#plt.savefig('F:/Paper6_vit+cnn/results/P6_proposed_lime1.jpg',dpi=600
)
plt.show()
```




```python
# Code for turning off x and y labels
plt.imshow(temp)
plt.title('LIME Explanation ')
plt.xticks([])  # Turn off x labels
plt.yticks([])  # Turn off y labels
```

```
#plt.savefig('F:/Paper6_vit+cnn/results/P6_ResNet50V2_lime.jpg',
dpi=600)
plt.show()
```



```
import datetime
cur_date=datetime.datetime.now()
print("current date and time",cur_date)
```

```
current date and time 2024-02-19 10:07:34.709927
```