```python
# importing datetime module for now()
import datetime
# using now() to get current time
current_time = datetime.datetime.now()
# Printing value of now.
print("The current time is:", current_time)
```

```
The current time is: 2024-02-19 11:01:18.709227
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout

from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from sklearn.metrics import classification_report,confusion_matrix
import tensorflow as tf
import cv2
import os



labels = ['ELS', 'ER','HL','LLS','ND','RUST']
img_size = 212
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img))[...,::-
1] #convert BGR to RGB format
                resized_arr = cv2.resize(img_arr, (img_size,
img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)

#Now we can easily fetch our train and validation data.
train = get_data('F:\\Paper6_vit+cnn\\Groundnut_Leaf_dataset\\train1')
val = get_data('F:\\Paper6_vit+cnn\\Groundnut_Leaf_dataset\\Test1')
#F:\\Datasets\\rice_dataset\\Paper2_dataset\\Ref_ricedataset\\
Rice_dataset_final\\Train
#F:/Refined_dataset/EXP_dataset/Train
```

```
C:\Users\NITRR\AppData\Local\Temp\ipykernel_1520\1881809242.py:15: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  return np.array(data)
```

```
x_train = []
y_train = []
x_val = []
y_val = []

for feature, label in train:
  x_train.append(feature)
  y_train.append(label)

for feature, label in val:
  x_val.append(feature)
  y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255

x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)

datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range = 30,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip = True,  # randomly flip images
        vertical_flip=False)  # randomly flip images
```

```
datagen.fit(x_train)


#Let’s define a simple CNN model with 3 Convolutional layers followed
by max-pooling layers. A dropout layer is added after the 3rd maxpool
operation to avoid overfitting.

model = Sequential()
model.add(Conv2D(32,3,padding="same", activation="relu",
input_shape=(212,212,3)))
model.add(MaxPool2D())

model.add(Conv2D(32, 3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Conv2D(32, 3, padding="same", activation="relu")) #addred
one
model.add(MaxPool2D())

model.add(Conv2D(64, 3, padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(6, activation="softmax"))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 212, 212, 32)      896

 max_pooling2d_4 (MaxPooling  (None, 106, 106, 32)     0
 2D)

 conv2d_5 (Conv2D)           (None, 106, 106, 32)      9248

 max_pooling2d_5 (MaxPooling  (None, 53, 53, 32)       0
 2D)

 conv2d_6 (Conv2D)           (None, 53, 53, 32)        9248

 max_pooling2d_6 (MaxPooling  (None, 26, 26, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 26, 26, 64)        18496
```

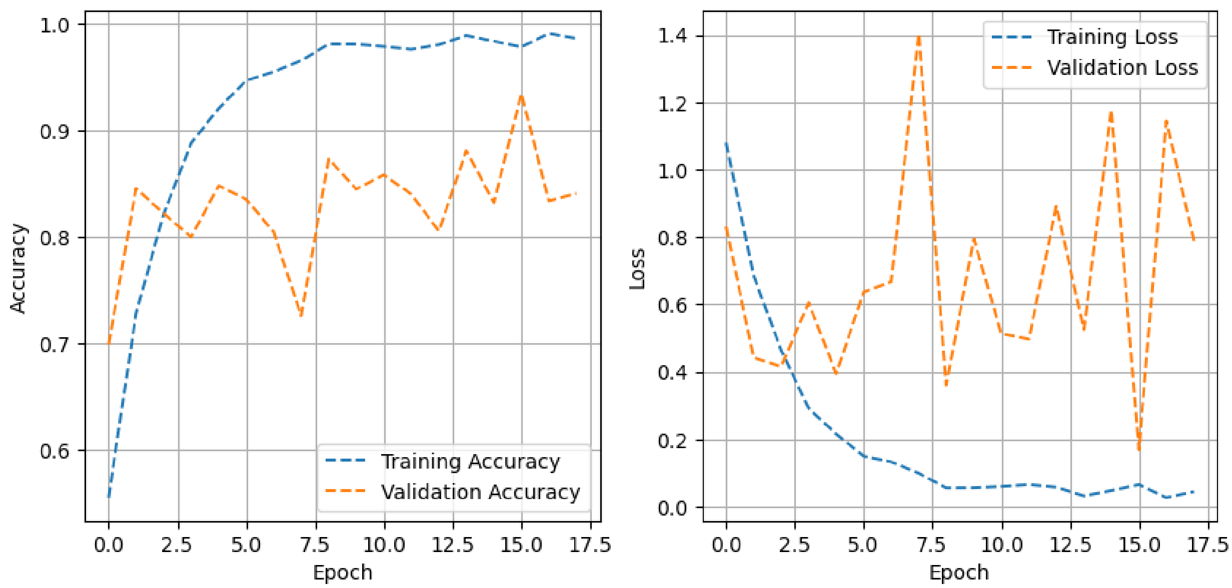
```
 max_pooling2d_7 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 13, 13, 64)        0

 flatten_1 (Flatten)         (None, 10816)             0

 dense_2 (Dense)             (None, 128)               1384576

 dense_3 (Dense)             (None, 6)                 774

=================================================================
Total params: 1,423,238
Trainable params: 1,423,238
Non-trainable params: 0
_________________________________________________________________
```

```
opt = Adam(lr=0.001) #lr=0.0001,0.001
model.compile(optimizer = opt , loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True) ,
metrics = ['accuracy'])

history = model.fit(x_train,y_train, batch_size=26,epochs = 18 ,
validation_data = (x_val, y_val))
```

```
Epoch 1/18

C:\Users\NITRR\AppData\Roaming\Python\Python310\site-packages\keras\
backend.py:5612: UserWarning: "`sparse_categorical_crossentropy`
received `from_logits=True`, but the `output` argument was produced by
a Softmax activation and thus does not represent logits. Was this
intended?
  output, from_logits = _get_logits(

305/305 [==============================] - 179s 582ms/step - loss:
1.0806 - accuracy: 0.5554 - val_loss: 0.8314 - val_accuracy: 0.6990
Epoch 2/18
305/305 [==============================] - 177s 582ms/step - loss:
0.6855 - accuracy: 0.7282 - val_loss: 0.4422 - val_accuracy: 0.8455
Epoch 3/18
305/305 [==============================] - 176s 577ms/step - loss:
0.4646 - accuracy: 0.8214 - val_loss: 0.4158 - val_accuracy: 0.8219
Epoch 4/18
305/305 [==============================] - 176s 577ms/step - loss:
0.2915 - accuracy: 0.8879 - val_loss: 0.6051 - val_accuracy: 0.7998
Epoch 5/18
305/305 [==============================] - 175s 575ms/step - loss:
0.2155 - accuracy: 0.9205 - val_loss: 0.3942 - val_accuracy: 0.8479
Epoch 6/18
```

```
305/305 [==============================] - 187s 612ms/step - loss: 
0.1497 - accuracy: 0.9466 - val_loss: 0.6372 - val_accuracy: 0.8353
Epoch 7/18
305/305 [==============================] - 188s 618ms/step - loss: 
0.1328 - accuracy: 0.9549 - val_loss: 0.6668 - val_accuracy: 0.8046
Epoch 8/18
305/305 [==============================] - 189s 619ms/step - loss: 
0.0990 - accuracy: 0.9654 - val_loss: 1.4045 - val_accuracy: 0.7258
Epoch 9/18
305/305 [==============================] - 195s 639ms/step - loss: 
0.0555 - accuracy: 0.9809 - val_loss: 0.3606 - val_accuracy: 0.8731
Epoch 10/18
305/305 [==============================] - 192s 631ms/step - loss: 
0.0559 - accuracy: 0.9808 - val_loss: 0.7949 - val_accuracy: 0.8448
Epoch 11/18
305/305 [==============================] - 192s 630ms/step - loss: 
0.0597 - accuracy: 0.9786 - val_loss: 0.5134 - val_accuracy: 0.8582
Epoch 12/18
305/305 [==============================] - 191s 628ms/step - loss: 
0.0654 - accuracy: 0.9760 - val_loss: 0.4970 - val_accuracy: 0.8400
Epoch 13/18
305/305 [==============================] - 190s 623ms/step - loss: 
0.0576 - accuracy: 0.9803 - val_loss: 0.8918 - val_accuracy: 0.8054
Epoch 14/18
305/305 [==============================] - 193s 632ms/step - loss: 
0.0320 - accuracy: 0.9890 - val_loss: 0.5255 - val_accuracy: 0.8810
Epoch 15/18
305/305 [==============================] - 192s 630ms/step - loss: 
0.0478 - accuracy: 0.9836 - val_loss: 1.1778 - val_accuracy: 0.8322
Epoch 16/18
305/305 [==============================] - 192s 629ms/step - loss: 
0.0651 - accuracy: 0.9781 - val_loss: 0.1685 - val_accuracy: 0.9346
Epoch 17/18
305/305 [==============================] - 193s 633ms/step - loss: 
0.0269 - accuracy: 0.9909 - val_loss: 1.1446 - val_accuracy: 0.8337
Epoch 18/18
305/305 [==============================] - 193s 631ms/step - loss: 
0.0446 - accuracy: 0.9862 - val_loss: 0.7892 - val_accuracy: 0.8408

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(18)

plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training 
Accuracy',linestyle='dashed',linewidth='1.4')
```

```
plt.plot(epochs_range, val_acc, label='Validation
Accuracy',linestyle='dashed',linewidth='1.4')
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
#plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training
Loss',linestyle='dashed',linewidth='1.4')
plt.plot(epochs_range, val_loss, label='Validation
Loss',linestyle='dashed',linewidth='1.4')
plt.legend(loc='upper right')
#plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.grid()
plt.show()
```



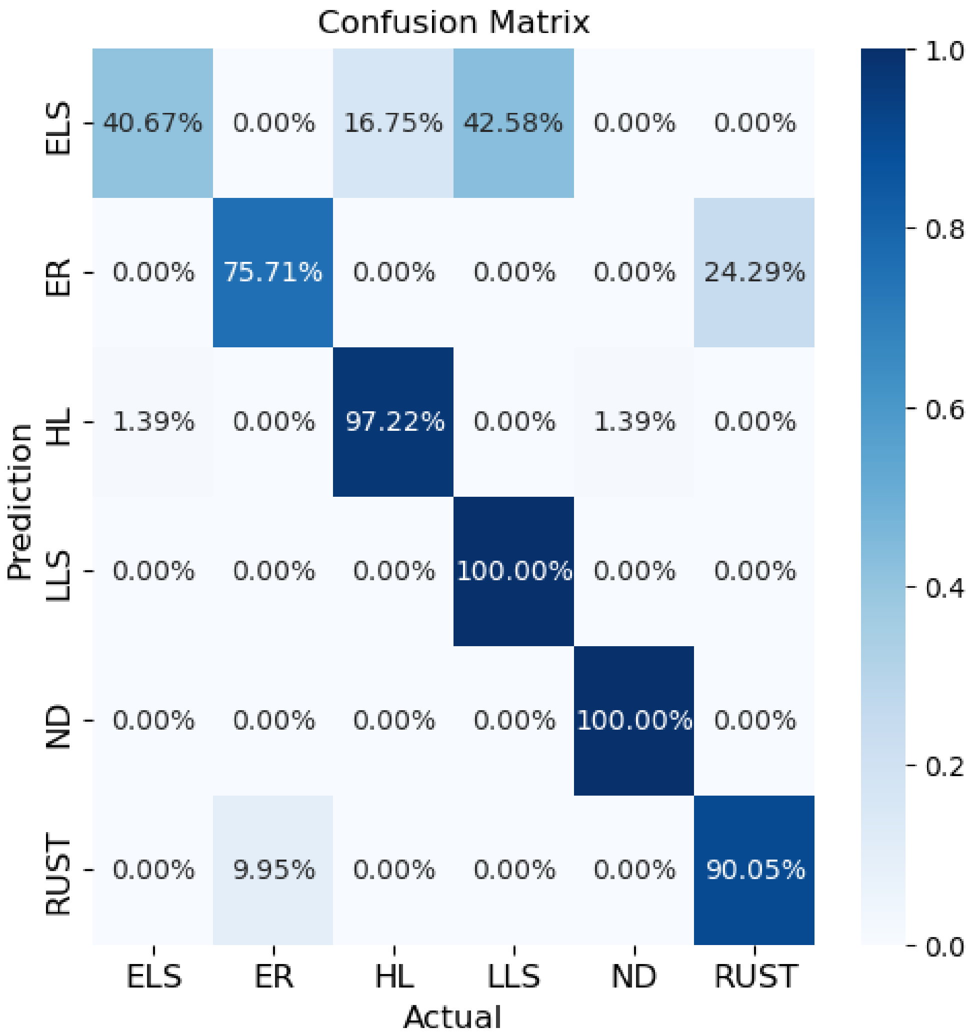

```
# predictions = model.predict_classes(x_val)
# predictions = predictions.reshape(1,-1)[0]
# print(classification_report(y_val, predictions, target_names =
['NK', 'NP','PK']))


predict_x=model.predict(x_val)
classes_x=np.argmax(predict_x,axis=1)
print(classification_report(y_val, classes_x, target_names = ['ELS',
'ER','HL','LLS','ND','RUST']))
```

```
40/40 [==============================] - 5s 133ms/step
              precision    recall  f1-score   support

         ELS       0.97      0.41      0.57       209
          ER       0.88      0.76      0.82       210
          HL       0.86      0.97      0.91       216
         LLS       0.70      1.00      0.82       206
          ND       0.99      1.00      0.99       217
        RUST       0.79      0.90      0.84       211

    accuracy                           0.84      1269
   macro avg       0.86      0.84      0.83      1269
weighted avg       0.86      0.84      0.83      1269
```

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_val, classes_x))
```

```
[[ 85   0  35  89   0   0]
 [  0 159   0   0   0  51]
 [  3   0 210   0   3   0]
 [  0   0   0 206   0   0]
 [  0   0   0   0 217   0]
 [  0  21   0   0   0 190]]
```

```
#Import the necessary libraries
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 6))
#compute the confusion matrix.
cm = confusion_matrix(y_val, classes_x)

#Plot the confusion matrix.
sns.heatmap(cm/np.sum(cm,axis=1).reshape(-
1,1),cmap='Blues',annot=True,fmt='.2%',xticklabels=['ELS',
'ER','HL','LLS','ND','RUST'],yticklabels=['ELS',
'ER','HL','LLS','ND','RUST'])
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.ylabel('Prediction',fontsize=12)
plt.xlabel('Actual',fontsize=12)
plt.title('Confusion Matrix',fontsize=12)
#plt.savefig('F:/paper_3_code_files_final/results/results1/TLCNN_CM.jp
g',dpi=600)
plt.show()
```




```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
precision_score(y_val, classes_x, average='micro')
```

```
0.8408195429472025
```

```
recall = recall_score(y_val, classes_x, average='micro')
print('recall=',recall)
```

```
recall= 0.8408195429472025
```

```
from sklearn.metrics import f1_score
f1_score(y_val, classes_x, average='micro')
```

```
0.8408195429472025
```

```
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import balanced_accuracy_score
a=matthews_corrcoef(y_val, classes_x)
```

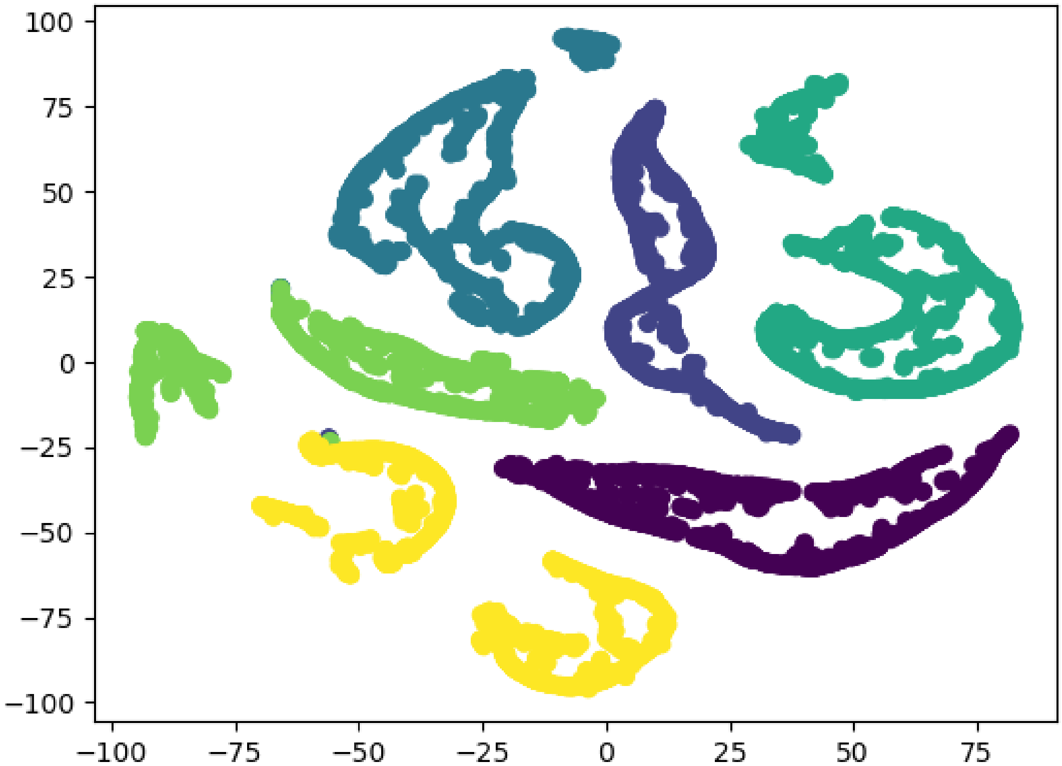
```
b=balanced_accuracy_score(y_val, classes_x)
print(a)
print(b)
```

```
0.8166805383625645
0.8394229296012784
```

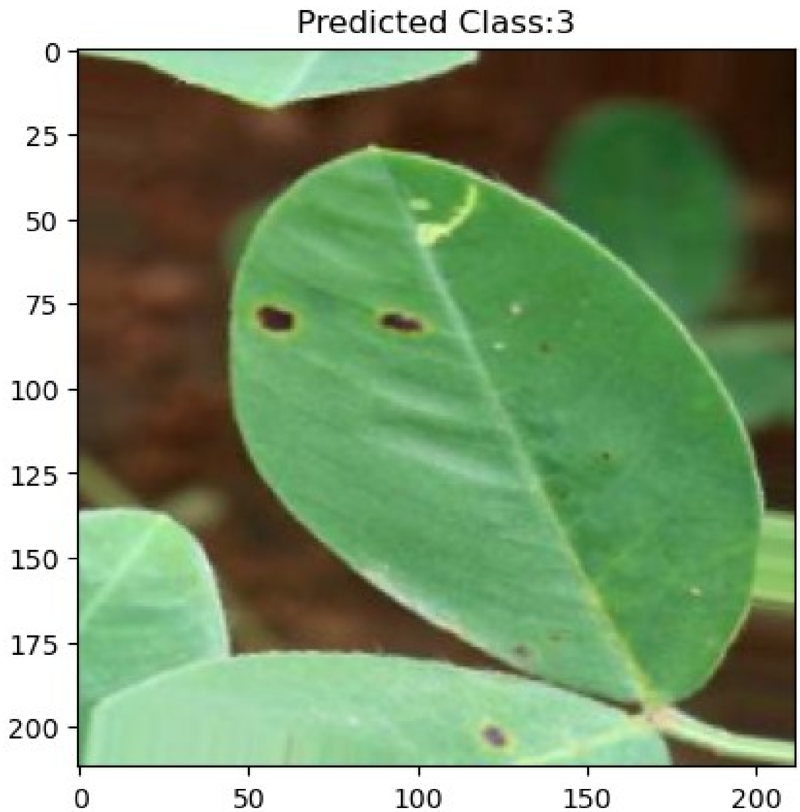
```
from sklearn.manifold import TSNE
# Extract the features from the last layer of the CNN
features = model.predict(x_train)
# Reduce the dimensionality of the features using t-SNE
tsne = TSNE(n_components=2, perplexity=30.0, early_exaggeration=12.0,
learning_rate=200.0, n_iter=1000, n_iter_without_progress=300,
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)
```

```
248/248 [==============================] - 31s 126ms/step
```

```
# Plot the results
plt.scatter(features_embedded[:, 0], features_embedded[:, 1],
c=y_train)
#plt.axis('off')
#plt.savefig('F:/Paper6_vit+cnn/results/Rice_dataset_results/P6_r_CNN_
TSNE.jpg',dpi=600)
plt.show()
```




```
import lime
from lime import lime_image

# Choose an image from the test set for explanation
image_index = 5
img = x_val[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(212, 212,3))
plt.title(f'Predicted Class:{predicted_class}')
plt.show()

1/1 [==============================] - 0s 28ms/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":"119f974bb00b43ecac21f4876d5861c4","version_major":2,"version_minor":0}

```
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 41ms/step
```

```
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 36ms/step
```

```
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 52ms/step
```

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


{"model_id":"db9d962d27ea4fa0b9d60a08510fe151","version_major":2,"vers
ion_minor":0}


```
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 54ms/step
```

```
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 48ms/step
```

```
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 76ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 49ms/step
```

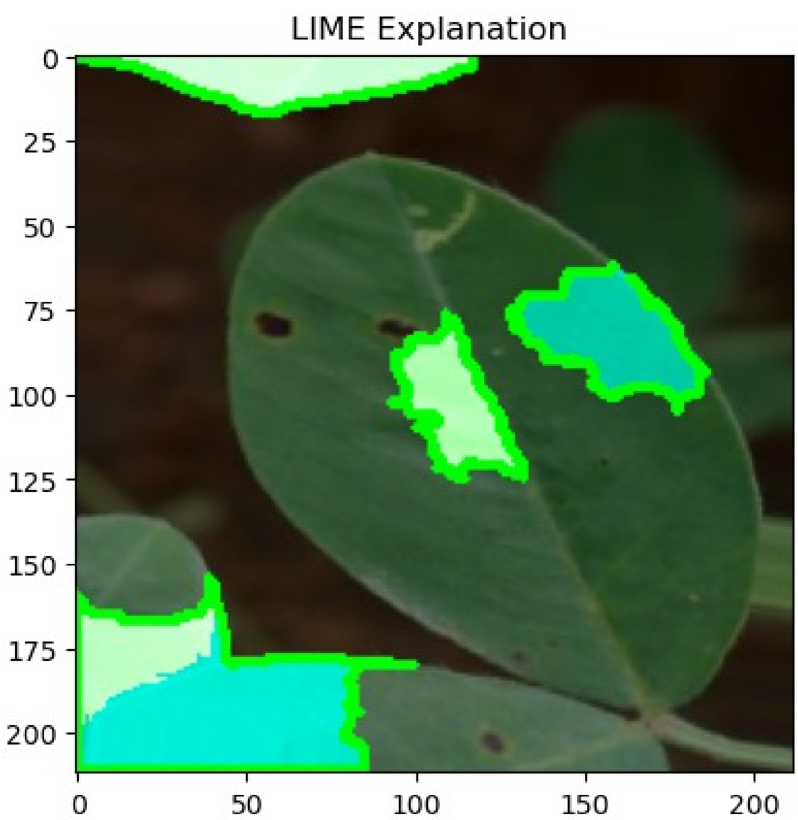
```python
#visualise LIME explanations with cv2.polylines
temp,mask=explanation.get_image_and_mask(explanation.top_labels[0],
positive_only=False,num_features=5,hide_rest=False)
```

```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([[[  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0],
        ...,
        [ 18,   9,   2],
        [ 18,   9,   2],
        [ 18,   9,   2]],

       [[  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0],
        ...,
        [ 18,   9,   2],
        [ 18,   9,   2],
        [ 18,   9,   2]],

       [[  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0],
        ...,
        [ 18,   9,   2],
        [ 18,   9,   2],
        [ 18,   9,   2]],

       ...,

       [[  0, 255,   0],
        [  0, 255,   0],
        [  0, 230, 202],
        ...,
        [100, 116,  86],
        [102, 118,  88],
        [103, 119,  89]],

       [[  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0],
        ...,
```

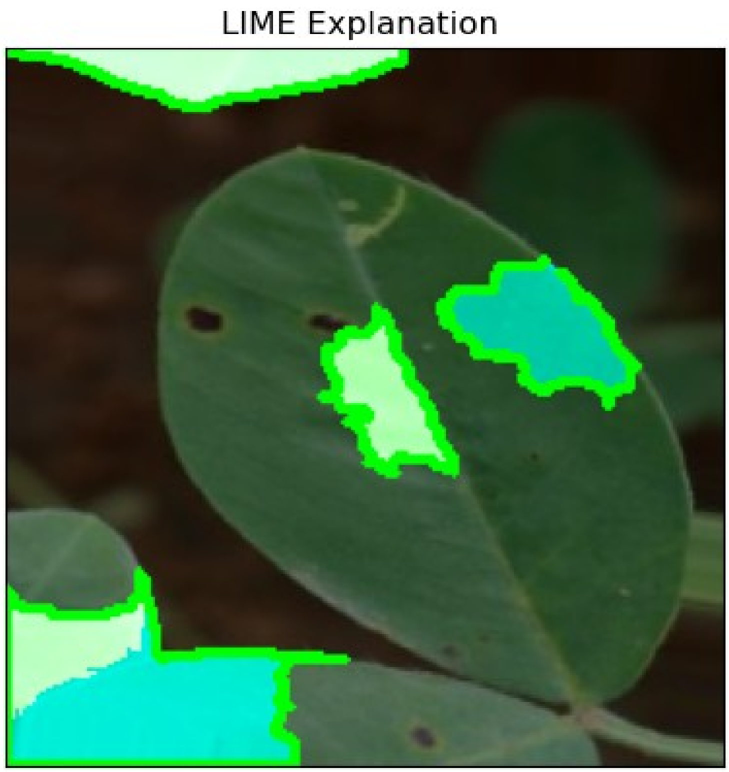
```
         [ 78,  90,  63],
         [ 81,  93,  66],
         [ 84,  96,  68]],

        [[  0, 255,   0],
         [  0, 255,   0],
         [  0, 255,   0],
         ...,
         [ 64,  73,  48],
         [ 67,  77,  52],
         [ 70,  80,  54]]], dtype=uint8)
```

```
#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()
```




```
# 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/Rice_dataset_results/P6_r_CNN_lime.pdf', dpi=600)
plt.show()
```




```
# importing datetime module for now()
import datetime
# using now() to get current time
current_time = datetime.datetime.now()
# Printing value of now.
print("The current time is:", current_time)
```

```
The current time is: 2024-02-19 12:25:51.252493
```