tomato-disease-classification-2-1

October 18, 2024

[1]:
```
#imported the Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from keras import layers,Sequential
from keras.layers import␣
↪Conv2D,MaxPooling2D,Flatten,Dense,Dropout,BatchNormalization
import os
import cv2 as cv
from keras.models import Model
from keras.applications.resnet50 import ResNet50
from keras.applications.vgg16 import preprocess_input
from keras.models import Sequential
from glob import glob
```

[2]:
```
path=r"C:\Users\RAHUL PATIL\Downloads\archive (11)\tomato\train"#path of upto␣
↪the training folder
```

[3]:
```
dis=os.listdir(path)#list of that inside the folder
dis
```

[3]:
```
['Tomato___Bacterial_spot',
 'Tomato___Early_blight',
 'Tomato___healthy',
 'Tomato___Late_blight',
 'Tomato___Leaf_Mold',
 'Tomato___Septoria_leaf_spot',
 'Tomato___Spider_mites Two-spotted_spider_mite',
 'Tomato___Target_Spot',
 'Tomato___Tomato_mosaic_virus',
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus']
```

[4]:
```
dis.index('Tomato___Early_blight')#disease
```

[4]:
```
1
```

[55]:
```
Data=[]
for i in dis:  # ALL FOLDERS INSIDE PARENT PATH
    A = os.path.join(path, i)  # FOR JOINING PATHS
    count = 0  # Initialize a counter for images loaded from the current folder

    for j in os.listdir(A):  # FOR GETTING ALL CONTENT FROM FOLDER
        if count >= 100:  # Check if 100 images have already been loaded
            break

        B = os.path.join(A, j)  # JOINed up to that photo
        img = cv.imread(B)  # CONVERTING IMAGE TO PIXEL INTENSITY MATRIX

        if img is not None:  # Check if the image was loaded successfully
            C = cv.resize(img, (224, 224))  # RESIZING PIXEL INTENSITY MATRIX
            T = dis.index(i)  # FOR GETTING TARGET VARIABLE
            Data.append([C, T])  # TO STORE
            count += 1  # Increment the counter
```

[56]:
```
i=cv.imread(r"C:\Users\RAHUL PATIL\Downloads\archive␣
 ↪(11)\tomato\train\Tomato___healthy\fde29807-8223-4fc5-a06b-7cc93101a6d1___GH_HL␣
 ↪Leaf 174.JPG")
#taken the img with coverted into pixel
```

[57]:
```
i.shape
```

[57]:
```
(256, 256, 3)
```

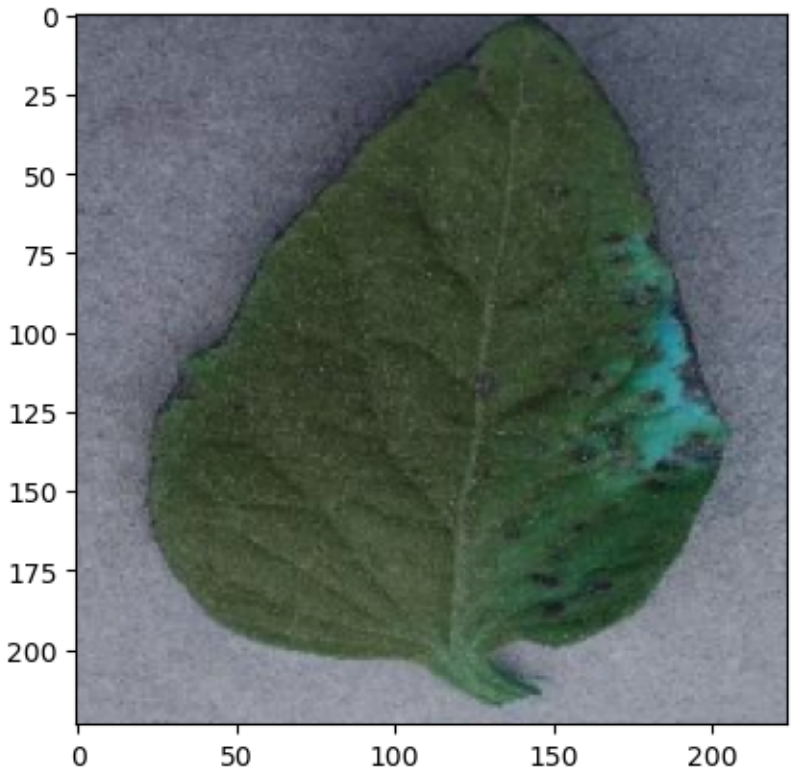
[58]:
```
Data[1]
```

[58]:
```
[array([[[114, 117, 132],
         [109, 112, 127],
         [105, 108, 123],
         …,
         [ 89,  96, 111],
         [ 84,  91, 106],
         [ 85,  92, 107]],

        [[107, 110, 125],
         [106, 109, 124],
         [102, 105, 120],
         …,
         [100, 107, 122],
         [110, 117, 132],
         [103, 110, 125]],

        [[110, 113, 128],
         [112, 115, 130],
```

```
        [109, 112, 127],
        …,
        [ 99, 106, 121],
        [108, 115, 129],
        [108, 115, 130]],

       …,

       [[143, 145, 156],
        [146, 148, 159],
        [147, 149, 160],
        …,
        [140, 144, 155],
        [143, 147, 158],
        [147, 151, 162]],

       [[143, 145, 156],
        [144, 146, 157],
        [144, 146, 158],
        …,
        [147, 151, 162],
        [151, 154, 166],
        [155, 159, 170]],

       [[142, 144, 155],
        [159, 160, 171],
        [150, 152, 163],
        …,
        [144, 148, 159],
        [141, 145, 156],
        [140, 144, 155]]], dtype=uint8),
 0]
```

[59]:
```
plt.imshow(Data[1][0])#IMAGE SHWING CODE
```

[59]:
```
<matplotlib.image.AxesImage at 0x257f7bccf80>
```



[60]:
```
len(Data)
```

[60]: 1000

[61]:
```
len(Data[1][0])
```

[61]: 224

[64]:
```
import random #for shuffle
random.shuffle(Data)
```

[65]:
```
F=[]
T=[]
for i,j in Data:#FOR SEPERATING FEATURES AND TARGETS
    F.append(i)
    T.append(j)
```

[66]:
```
len(F)
```

[66]: 1000

[67]:
```
T
```

[67]: [8,
0,
4,
7,
8,
9,
4,
9,
9,
1,
4,
6,
8,
5,
3,
7,
8,
2,
5,
3,
9,
9,
3,
4,
1,
4,
7,
3,
7,
5,
0,
4,
2,
0,
8,
5,
9,
8,
9,
7,
4,
5,
8,
2,
2,
0,
7,

```
4,
6,
1,
7,
0,
0,
0,
7,
0,
8,
4,
6,
9,
0,
5,
8,
6,
6,
3,
8,
3,
0,
6,
9,
3,
9,
4,
9,
7,
3,
7,
1,
6,
9,
4,
7,
2,
7,
5,
7,
8,
6,
9,
2,
5,
1,
5,
```

4,
9,
9,
0,
0,
0,
3,
7,
2,
8,
8,
3,
4,
4,
0,
5,
4,
7,
0,
4,
5,
5,
8,
5,
9,
6,
3,
8,
4,
7,
5,
8,
2,
6,
4,
1,
0,
1,
8,
0,
0,
2,
0,
1,
3,
9,
1,

2,
9,
5,
8,
5,
6,
0,
4,
8,
4,
5,
5,
4,
5,
3,
1,
3,
7,
0,
8,
3,
8,
5,
0,
9,
0,
2,
5,
1,
4,
3,
2,
6,
7,
4,
1,
9,
3,
7,
3,
6,
8,
9,
4,
3,
0,
5,

9,
2,
4,
5,
6,
7,
5,
5,
9,
6,
7,
8,
6,
3,
0,
1,
8,
4,
2,
6,
7,
7,
0,
3,
0,
0,
2,
1,
2,
5,
8,
1,
1,
1,
1,
6,
1,
7,
2,
2,
6,
6,
0,
5,
5,
7,
4,

```
1,
1,
8,
4,
2,
5,
9,
7,
5,
5,
0,
9,
7,
5,
1,
7,
4,
8,
4,
0,
0,
3,
0,
8,
5,
2,
0,
1,
5,
7,
6,
7,
2,
8,
8,
5,
1,
0,
4,
2,
5,
7,
7,
1,
2,
2,
0,
```

```
2,
8,
3,
3,
3,
1,
8,
7,
4,
1,
2,
9,
2,
5,
0,
5,
0,
7,
1,
2,
6,
0,
4,
8,
2,
4,
2,
0,
1,
8,
2,
9,
7,
9,
2,
1,
3,
7,
4,
5,
2,
2,
6,
6,
1,
8,
3,
```

6,
4,
9,
4,
0,
4,
7,
2,
9,
8,
3,
4,
4,
8,
5,
1,
6,
9,
6,
9,
7,
7,
2,
9,
2,
2,
2,
2,
6,
1,
8,
0,
4,
8,
4,
5,
7,
9,
7,
4,
9,
3,
8,
2,
9,
2,
1,

5,
4,
2,
1,
1,
9,
2,
4,
7,
4,
6,
7,
5,
2,
5,
6,
6,
0,
2,
9,
9,
9,
9,
9,
0,
1,
2,
8,
0,
8,
5,
1,
3,
7,
6,
1,
8,
1,
0,
1,
5,
3,
4,
3,
3,
9,
8,

8,
9,
5,
1,
4,
8,
8,
4,
5,
7,
8,
6,
7,
5,
2,
4,
4,
0,
7,
3,
3,
8,
9,
9,
4,
7,
4,
3,
1,
3,
3,
0,
1,
8,
8,
2,
3,
1,
2,
0,
8,
5,
4,
5,
4,
2,
2,

```
7,
7,
3,
4,
0,
0,
1,
0,
4,
4,
3,
0,
8,
6,
2,
1,
1,
6,
1,
5,
2,
4,
6,
6,
6,
3,
5,
5,
4,
8,
8,
4,
6,
3,
7,
3,
9,
6,
1,
1,
4,
6,
7,
3,
3,
7,
9,
```

```
5,
9,
1,
8,
4,
0,
0,
4,
1,
6,
2,
0,
1,
1,
2,
3,
1,
5,
3,
0,
2,
6,
8,
8,
2,
7,
0,
6,
5,
6,
7,
6,
4,
3,
2,
1,
1,
4,
9,
0,
5,
6,
6,
3,
6,
4,
8,
```

7,
0,
4,
1,
8,
0,
1,
0,
2,
6,
9,
9,
7,
5,
2,
3,
2,
9,
6,
0,
1,
2,
5,
6,
4,
3,
3,
1,
2,
5,
8,
8,
4,
8,
2,
1,
1,
4,
0,
6,
1,
3,
0,
4,
1,
8,
7,

```
2,
3,
1,
0,
5,
7,
8,
3,
3,
7,
2,
6,
5,
7,
0,
0,
1,
9,
7,
8,
0,
0,
7,
1,
2,
3,
1,
7,
7,
5,
3,
3,
3,
3,
6,
6,
3,
4,
2,
1,
5,
0,
6,
0,
7,
5,
2,
```

```
2,
3,
4,
5,
1,
3,
7,
9,
0,
6,
3,
1,
5,
9,
4,
8,
9,
6,
5,
8,
5,
4,
9,
6,
7,
1,
1,
6,
7,
9,
3,
9,
0,
4,
8,
4,
5,
0,
9,
3,
2,
9,
5,
6,
4,
8,
0,
```

```
1,
6,
4,
4,
6,
7,
6,
0,
9,
5,
8,
2,
1,
7,
8,
0,
6,
4,
9,
6,
0,
8,
9,
3,
5,
0,
7,
8,
6,
5,
6,
7,
1,
5,
9,
5,
0,
5,
7,
0,
9,
5,
1,
8,
7,
8,
6,
```

9,
1,
6,
3,
8,
3,
3,
3,
6,
4,
1,
0,
3,
2,
8,
0,
2,
8,
1,
2,
3,
2,
7,
8,
1,
3,
1,
3,
6,
4,
7,
9,
3,
7,
7,
0,
3,
9,
3,
8,
5,
0,
9,
5,
3,
1,
5,

```
9,
8,
8,
9,
6,
1,
9,
1,
4,
5,
6,
1,
5,
0,
6,
4,
7,
4,
2,
7,
1,
6,
6,
6,
9,
5,
8,
5,
5,
9,
7,
2,
6,
6,
8,
8,
6,
5,
7,
6,
4,
2,
4,
2,
1,
8,
0,
```

```
6,
7,
9,
9,
9,
9,
4,
3,
6,
3,
7,
5,
9,
0,
3,
1,
5,
6,
7,
3,
9,
5,
0,
7,
2,
5,
9,
1,
5,
0,
8,
5,
3,
2,
3,
2,
3,
6,
2,
7,
9,
5,
2,
2,
7,
7,
4,
```

6,
2,
0,
9,
8,
8,
9,
8,
5,
6,
9,
4,
9,
9,
2,
2,
9,
6,
3,
5,
9,
2,
2,
1,
4,
7,
3,
9,
6,
9,
7,
3,
4,
2,
4,
2,
1,
2,
9,
6,
0,
2,
4,
3,
7,
8,
0,

3,
8,
2,
6,
3,
2,
7,
4,
7,
0,
8,
8,
3,
9,
3,
4,
1,
4,
4,
9,
8,
6,
1,
9,
6,
9,
2,
0,
3,
3,
2,
7,
7,
6,
5,
7,
8,
3,
7,
6,
1,
3,
7,
0,
5,
0,
8,

```
 9,
 6,
 6,
 5,
 0,
 0,
 8,
 1,
 1,
 6,
 9,
 1,
 8]
```

[68]:
```
T1=pd.get_dummies(T).replace({True:1,False:0}) # THESE TARGETS ARE NOMINAL SO␣
 ↪GETTING DUMMIES FOR MAINTAING THE IN BETWEEN RELATION WITHOUT AFFECTING THE␣
 ↪MODEL
T1
```

```
C:\Users\RAHUL PATIL\AppData\Local\Temp\ipykernel_101308\4243940820.py:1:
FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  T1=pd.get_dummies(T).replace({True:1,False:0}) # THESE TARGETS ARE NOMINAL SO
GETTING DUMMIES FOR MAINTAING THE IN BETWEEN RELATION WITHOUT AFFECTING THE
MODEL
```

[68]:

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	1	0
..	..	..	..	..	..	..	..	..	..	..
995	0	1	0	0	0	0	0	0	0	0
996	0	0	0	0	0	0	1	0	0	0
997	0	0	0	0	0	0	0	0	0	1
998	0	1	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	1	0

[1000 rows x 10 columns]

[69]:
```
F=np.array(F)#FOR FASTER CALCULATION
```

[70]:
```
F1=F/255 #FOR MINMAX SCALER
```

[71]:
```
F[1]#FIRST IMAGE WITHOUT MINMAX
```

[71]:
```
array([[[147, 148, 158],
        [147, 148, 158],
        [147, 148, 158],
        …,
        [159, 160, 170],
        [156, 157, 167],
        [154, 155, 165]],

       [[148, 149, 159],
        [148, 149, 159],
        [148, 149, 159],
        …,
        [151, 152, 163],
        [151, 152, 162],
        [150, 151, 161]],

       [[146, 147, 156],
        [146, 147, 157],
        [146, 147, 157],
        …,
        [150, 151, 161],
        [150, 151, 161],
        [150, 151, 161]],

       …,

       [[ 97,  98, 112],
        [ 99, 100, 114],
        [ 96,  97, 111],
        …,
        [ 99, 101, 112],
        [106, 108, 119],
        [110, 112, 123]],

       [[ 90,  91, 105],
        [102, 103, 117],
        [ 95,  96, 110],
        …,
        [111, 113, 124],
        [107, 108, 119],
        [ 96,  99, 109]],

       [[ 84,  85,  99],
        [105, 106, 120],
        [ 97,  97, 111],
```

```
        …,
        [112, 114, 125],
        [109, 111, 122],
        [ 96,  98, 109]]], dtype=uint8)
```

[72]:
```
F1[1]#FIRST IMAGE AFTER MINMAX
```

[72]:
```
array([[[0.57647059, 0.58039216, 0.61960784],
        [0.57647059, 0.58039216, 0.61960784],
        [0.57647059, 0.58039216, 0.61960784],
        …,
        [0.62352941, 0.62745098, 0.66666667],
        [0.61176471, 0.61568627, 0.65490196],
        [0.60392157, 0.60784314, 0.64705882]],

       [[0.58039216, 0.58431373, 0.62352941],
        [0.58039216, 0.58431373, 0.62352941],
        [0.58039216, 0.58431373, 0.62352941],
        …,
        [0.59215686, 0.59607843, 0.63921569],
        [0.59215686, 0.59607843, 0.63529412],
        [0.58823529, 0.59215686, 0.63137255]],

       [[0.57254902, 0.57647059, 0.61176471],
        [0.57254902, 0.57647059, 0.61568627],
        [0.57254902, 0.57647059, 0.61568627],
        …,
        [0.58823529, 0.59215686, 0.63137255],
        [0.58823529, 0.59215686, 0.63137255],
        [0.58823529, 0.59215686, 0.63137255]],


       …,

       [[0.38039216, 0.38431373, 0.43921569],
        [0.38823529, 0.39215686, 0.44705882],
        [0.37647059, 0.38039216, 0.43529412],
        …,
        [0.38823529, 0.39607843, 0.43921569],
        [0.41568627, 0.42352941, 0.46666667],
        [0.43137255, 0.43921569, 0.48235294]],

       [[0.35294118, 0.35686275, 0.41176471],
        [0.4       , 0.40392157, 0.45882353],
        [0.37254902, 0.37647059, 0.43137255],
        …,
        [0.43529412, 0.44313725, 0.48627451],
        [0.41960784, 0.42352941, 0.46666667],
```

```
        [0.37647059, 0.38823529, 0.42745098]],

       [[0.32941176, 0.33333333, 0.38823529],
        [0.41176471, 0.41568627, 0.47058824],
        [0.38039216, 0.38039216, 0.43529412],
        …,
        [0.43921569, 0.44705882, 0.49019608],
        [0.42745098, 0.43529412, 0.47843137],
        [0.37647059, 0.38431373, 0.42745098]]])
```

[73]:
```
F1.shape#1000 IMAGES HAVING WIDTH OF 224 ,HEIGHT OF 224 AND 3 RGB CHANNEL
```

[73]:
```
(1000, 224, 224, 3)
```

[74]:
```
T1.shape #TARGET VARIABLE SHAPE
```

[74]:
```
(1000, 10)
```

[75]:
```
T=np.array(T)#faster Calculation
```

0.1 MODEL BUILDING

[85]:
```
model=Sequential()
#DATA AUGMENTATION

model.add( Conv2D( 160, (3,3), activation='relu') )#160 IS FILTER COUNT␣
 ↪,(3*3) IS FILTER SIZE
model.add( MaxPooling2D( (2,2) ,strides=(1,1)))#(2*2)IS THE MAXPOOLING␣
 ↪MATRIX,

model.add(Conv2D(140,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2),strides=(2,2)))

model.add(Conv2D(120,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2),strides=(2,2)))

model.add(Flatten())#get matrices in a series way..

model.add(Dense(90,input_shape=(224,224,3),activation='relu'))#input layer
model.add(Dense(80,activation='relu'))#hidden Layer
model.add(Dense(10,activation='softmax'))#output Layer

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

```
C:\Users\RAHUL PATIL\AppData\Roaming\Python\Python312\site-
```

```
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

[29]:
```
model.fit(F1,T1, epochs=1, validation_split=0.15, batch_size=32)
```

```
  6/266              1:49:33 25s/step -
accuracy: 0.1030 - loss: 4.2996
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[29], line 1
----> 1 model.fit(F1,T1, epochs=1, validation_split=0.15, batch_size=32)

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\keras\src\utils\traceback_ut ls.
 ↪py:117, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\keras\src\backend\tensorflow trainer.
 ↪py:320, in TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose,␣
 ↪callbacks, validation_split, validation_data, shuffle, class_weight,␣
 ↪sample_weight, initial_epoch, steps_per_epoch, validation_steps,␣
 ↪validation_batch_size, validation_freq)
    318 for step, iterator in epoch_iterator.enumerate_epoch():
    319     callbacks.on_train_batch_begin(step)
--> 320     logs = self.train_function(iterator)
    321     logs = self._pythonify_logs(logs)
    322     callbacks.on_train_batch_end(step, logs)

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\util\trace back_utils.
 ↪py:150, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:833, in Function.__call__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
```

```
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:878, in Function._call(self, *args, **kwds)
    875 self._lock.release()
    876 # In this case we have not created variables on the first call. So we ca n
    877 # run the first trace but we should fail if variables are created.
--> 878 results = tracing_compilation.call_function(
    879     args, kwds, self._variable_creation_config
    880 )
    881 if self._created_variables:
    882   raise ValueError("Creating variables on a non-first call to a functio "
    883                    " decorated with tf.function.")

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:139, in call_function(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:1322, in ConcreteFunction._call_flat(self, tensor_inputs, captured_inputs)
   1318 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1319 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NO E
   1320     and executing_eagerly):
   1321   # No tape is watching; skip to running the function.
-> 1322   return self._inference_function.call_preflattened(args)
   1323 forward_backward = self._select_forward_and_backward_functions(
   1324     args,
   1325     possible_gradient_type,
   1326     executing_eagerly)
   1327 forward_function, args_with_tangents = forward_backward.forward()

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:216, in AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) -> Any:
    215   """Calls with flattened tensor inputs and returns the structured␣
 ↪output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)
```

```
File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:251, in AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\cont xt.
 ↪py:1552, in Context.call_function(self, name, tensor_inputs, num_outputs)
   1550 cancellation_context = cancellation.context()
   1551 if cancellation_context is None:
-> 1552   outputs = execute.execute(
   1553       name.decode("utf-8"),
   1554       num_outputs=num_outputs,
   1555       inputs=tensor_inputs,
   1556       attrs=attrs,
   1557       ctx=self,
   1558   )
   1559 else:
   1560   outputs = execute.execute_with_cancellation(
   1561       name.decode("utf-8"),
   1562       num_outputs=num_outputs,
   (…)
   1566       cancellation_manager=cancellation_context,
   1567   )

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\exec te.
 ↪py:53, in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:
```

```
KeyboardInterrupt:
```

1 Problem:-Beacause it consumes more time and low accuracy so i used Transfer learning model.i.e:-RESTNET50 Model to get better accuracy and performance .

[104]:
```
resnet50=ResNet50(input_shape=(224,224,3),#input and weights fot model
         weights='imagenet',
         include_top=False)
```

[105]:
```
for i in resnet50.layers:
    i.trainable=False
```

[106]:
```
x=Flatten()(resnet50.output)
```

[107]:
```
y=Dense(100,activation='relu')(x)#added Layers
z=Dense(90,activation='relu')(y)
hh=Dense(80,activation='relu')(z)
```

[108]:
```
gg=Dense(10,activation='softmax')(hh)#output layer
```

[109]:
```
model=keras.Model(resnet50.input,gg)
```

[110]:
```
resnet50.trainable=False
```

[111]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[ ]:
```

```

[115]:
```
model.fit(F1,T1, epochs=42, validation_split=0.15, batch_size=32)
```

```
Epoch 1/42
27/27             121s 5s/step -
accuracy: 0.1816 - loss: 2.2391 - val_accuracy: 0.1867 - val_loss: 2.1315
Epoch 2/42
27/27             122s 5s/step -
accuracy: 0.2173 - loss: 2.0802 - val_accuracy: 0.2333 - val_loss: 2.0261
Epoch 3/42
27/27             122s 5s/step -
accuracy: 0.2902 - loss: 2.0204 - val_accuracy: 0.2800 - val_loss: 2.0000
Epoch 4/42
27/27             122s 5s/step -
accuracy: 0.2922 - loss: 1.9266 - val_accuracy: 0.3267 - val_loss: 1.9327
```

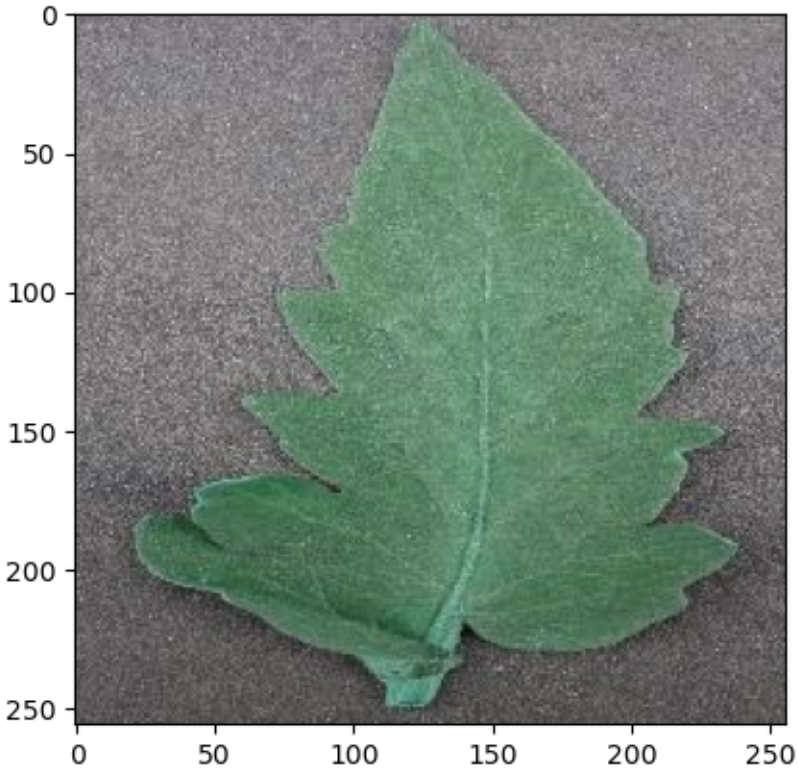
```
Epoch 5/42
27/27             121s 5s/step -
accuracy: 0.3029 - loss: 1.8620 - val_accuracy: 0.3267 - val_loss: 1.8025
Epoch 6/42
27/27             121s 5s/step -
accuracy: 0.3557 - loss: 1.8164 - val_accuracy: 0.3533 - val_loss: 1.8529
Epoch 7/42
27/27             122s 5s/step -
accuracy: 0.3261 - loss: 1.8357 - val_accuracy: 0.3400 - val_loss: 1.7254
Epoch 8/42
27/27             109s 4s/step -
accuracy: 0.3740 - loss: 1.6998 - val_accuracy: 0.2933 - val_loss: 1.9344
Epoch 9/42
27/27             89s 3s/step -
accuracy: 0.3285 - loss: 1.8045 - val_accuracy: 0.3800 - val_loss: 1.6960
Epoch 10/42
27/27             87s 3s/step -
accuracy: 0.4565 - loss: 1.5947 - val_accuracy: 0.3200 - val_loss: 1.7649
Epoch 11/42
27/27             87s 3s/step -
accuracy: 0.3936 - loss: 1.6732 - val_accuracy: 0.4467 - val_loss: 1.5267
Epoch 12/42
27/27             90s 3s/step -
accuracy: 0.4547 - loss: 1.5259 - val_accuracy: 0.4267 - val_loss: 1.5067
Epoch 13/42
27/27             86s 3s/step -
accuracy: 0.4954 - loss: 1.4908 - val_accuracy: 0.4333 - val_loss: 1.5225
Epoch 14/42
27/27             89s 3s/step -
accuracy: 0.4717 - loss: 1.4647 - val_accuracy: 0.3533 - val_loss: 1.8429
Epoch 15/42
27/27             90s 3s/step -
accuracy: 0.4785 - loss: 1.4855 - val_accuracy: 0.3267 - val_loss: 1.8054
Epoch 16/42
27/27             85s 3s/step -
accuracy: 0.4940 - loss: 1.4626 - val_accuracy: 0.4267 - val_loss: 1.6418
Epoch 17/42
27/27             89s 3s/step -
accuracy: 0.4856 - loss: 1.4077 - val_accuracy: 0.4267 - val_loss: 1.5752
Epoch 18/42
27/27             87s 3s/step -
accuracy: 0.5320 - loss: 1.3007 - val_accuracy: 0.3933 - val_loss: 1.5939
Epoch 19/42
27/27             88s 3s/step -
accuracy: 0.4652 - loss: 1.4365 - val_accuracy: 0.3800 - val_loss: 1.7774
Epoch 20/42
27/27             89s 3s/step -
accuracy: 0.5149 - loss: 1.2949 - val_accuracy: 0.3733 - val_loss: 1.6221
```

```
Epoch 21/42
27/27             86s 3s/step -
accuracy: 0.5560 - loss: 1.2000 - val_accuracy: 0.4533 - val_loss: 1.4643
Epoch 22/42
27/27             89s 3s/step -
accuracy: 0.4912 - loss: 1.3297 - val_accuracy: 0.4000 - val_loss: 1.5088
Epoch 23/42
27/27             89s 3s/step -
accuracy: 0.5493 - loss: 1.2494 - val_accuracy: 0.4000 - val_loss: 1.6857
Epoch 24/42
27/27             86s 3s/step -
accuracy: 0.5208 - loss: 1.3094 - val_accuracy: 0.4600 - val_loss: 1.4620
Epoch 25/42
27/27             89s 3s/step -
accuracy: 0.6322 - loss: 1.0233 - val_accuracy: 0.3867 - val_loss: 1.5808
Epoch 26/42
27/27             85s 3s/step -
accuracy: 0.5842 - loss: 1.1037 - val_accuracy: 0.4200 - val_loss: 1.5302
Epoch 27/42
27/27             85s 3s/step -
accuracy: 0.6211 - loss: 1.0775 - val_accuracy: 0.5133 - val_loss: 1.4727
Epoch 28/42
27/27             86s 3s/step -
accuracy: 0.5783 - loss: 1.1555 - val_accuracy: 0.3000 - val_loss: 2.1071
Epoch 29/42
27/27             89s 3s/step -
accuracy: 0.5221 - loss: 1.2701 - val_accuracy: 0.4133 - val_loss: 1.8483
Epoch 30/42
27/27             88s 3s/step -
accuracy: 0.5461 - loss: 1.3801 - val_accuracy: 0.4667 - val_loss: 1.5871
Epoch 31/42
27/27             87s 3s/step -
accuracy: 0.6552 - loss: 1.0074 - val_accuracy: 0.4400 - val_loss: 1.5169
Epoch 32/42
27/27             84s 3s/step -
accuracy: 0.6285 - loss: 1.0195 - val_accuracy: 0.4333 - val_loss: 1.5108
Epoch 33/42
27/27             87s 3s/step -
accuracy: 0.6191 - loss: 1.0332 - val_accuracy: 0.4667 - val_loss: 1.8618
Epoch 34/42
27/27             88s 3s/step -
accuracy: 0.5225 - loss: 1.3162 - val_accuracy: 0.4400 - val_loss: 1.5306
Epoch 35/42
27/27             84s 3s/step -
accuracy: 0.6322 - loss: 1.0642 - val_accuracy: 0.4800 - val_loss: 1.4010
Epoch 36/42
27/27             87s 3s/step -
accuracy: 0.6839 - loss: 0.9044 - val_accuracy: 0.5200 - val_loss: 1.4669
```

```
Epoch 37/42
27/27             88s 3s/step -
accuracy: 0.6996 - loss: 0.8679 - val_accuracy: 0.4867 - val_loss: 1.4917
Epoch 38/42
27/27             85s 3s/step -
accuracy: 0.6530 - loss: 0.9941 - val_accuracy: 0.4933 - val_loss: 1.4260
Epoch 39/42
27/27             88s 3s/step -
accuracy: 0.6465 - loss: 0.9804 - val_accuracy: 0.5067 - val_loss: 1.3854
Epoch 40/42
27/27             88s 3s/step -
accuracy: 0.6935 - loss: 0.8080 - val_accuracy: 0.4667 - val_loss: 1.5595
Epoch 41/42
27/27             85s 3s/step -
accuracy: 0.6700 - loss: 0.9380 - val_accuracy: 0.3800 - val_loss: 1.6921
Epoch 42/42
27/27             89s 3s/step -
accuracy: 0.6798 - loss: 0.9723 - val_accuracy: 0.5067 - val_loss: 1.4420
```

[115]:
```
<keras.src.callbacks.history.History at 0x2534eef8a70>
```

[131]:
```
img_path = r"C:\Users\RAHUL PATIL\Downloads\archive␣
 ↪(11)\tomato\train\Tomato___healthy\fe8f8808-2631-491e-a46b-bd2a1a4958e7___GH_HL␣
 ↪Leaf 213.1.JPG"
img = cv.imread(img_path,)#converted photo into pixels

img_resized = cv.resize(img, (224, 224))#resized the image
img_resized=img_resized.reshape(1,224, 224, 3)

plt.imshow(img,)
plt.show()
```



[132]:
```
prediction = model.predict(img_resized)
prediction
```

```
1/1             5s 5s/step
```

[132]:
```
array([[1.1455600e-07, 3.5983673e-03, 4.3499790e-14, 9.8521912e-01,
        1.1182067e-02, 2.0146469e-09, 1.3004514e-16, 9.5956576e-10,
        7.0784707e-18, 3.7950588e-07]], dtype=float32)
```

[133]:
```
dis[1]
```

[133]:
```
'Tomato___Early_blight'
```

[134]:
```
Test=r"C:\Users\RAHUL PATIL\Downloads\archive (11)\tomato\val"
```

[173]:
```
Data1=[]
for i in dis:  # ALL FOLDERS INSIDE PARENT PATH
    A = os.path.join(path, i)  # FOR JOINING PATHS
    count = 0  # Initialize a counter for images loaded from the current folder

    for j in os.listdir(A):  # FOR GETTING ALL CONTENT FROM FOLDER
        if count >= 100:  # Check if 100 images have already been loaded
```

```
            break

        B = os.path.join(A, j)  # JOINed up to that photo
        img = cv.imread(B)  # CONVERTING IMAGE TO PIXEL INTENSITY MATRIX

        if img is not None:  # Check if the image was loaded successfully
            C = cv.resize(img, (224, 224))  # RESIZING PIXEL INTENSITY MATRIX
            T = dis.index(i)  # FOR GETTING TARGET VARIABLE
            Data1.append([C, T])  # TO STORE
            count += 1  # Increment the counter
```

[174]:
```
import random #for shuffle
random.shuffle(Data)
```

[175]:
```
Data1[1]
```

[175]:
```
[array([[[114, 117, 132],
         [109, 112, 127],
         [105, 108, 123],
         …,
         [ 89,  96, 111],
         [ 84,  91, 106],
         [ 85,  92, 107]],

        [[107, 110, 125],
         [106, 109, 124],
         [102, 105, 120],
         …,
         [100, 107, 122],
         [110, 117, 132],
         [103, 110, 125]],

        [[110, 113, 128],
         [112, 115, 130],
         [109, 112, 127],
         …,
         [ 99, 106, 121],
         [108, 115, 129],
         [108, 115, 130]],

        …,

        [[143, 145, 156],
         [146, 148, 159],
         [147, 149, 160],
         …,
         [140, 144, 155],
```

```
          [143, 147, 158],
          [147, 151, 162]],

         [[143, 145, 156],
          [144, 146, 157],
          [144, 146, 158],
          …,
          [147, 151, 162],
          [151, 154, 166],
          [155, 159, 170]],

         [[142, 144, 155],
          [159, 160, 171],
          [150, 152, 163],
          …,
          [144, 148, 159],
          [141, 145, 156],
          [140, 144, 155]]], dtype=uint8),
 0]
```

[176]:
```
F3=[]
T2=[]
for i,j in Data1:#FOR SEPERATING FEATURES AND TARGETS
    F3.append(i)
    T2.append(j)
```

[177]:
```
T2
```

[177]:
```
[0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,

```
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
```

```
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
2,
2,
2,
2,
2,
2,
2,
```

```
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
```

```
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
2,
3,
```

```
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
```

```
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
3,
```

```
3,
3,
3,
3,
3,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
```

```
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
```

```
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
4,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
```

```
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
```

```
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
5,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
```

```
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
```

6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
6,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,

7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,

7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
7,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,

```
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
```

```
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
8,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
```

```
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
9,
```

```
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9,
 9]
```

[178]:
```
F3=np.array(F3)#minMax Scaler between 0-1
F3=F3/255
```

[179]:
```
predd=model.predict(F3)
predd
```

```
32/32              104s 3s/step
```

[179]:
```
array([[7.86572874e-01, 8.86464417e-02, 3.75992781e-03, …,
        1.08184870e-02, 3.74505507e-05, 2.52712518e-02],
       [1.26339614e-01, 4.28561820e-03, 7.63513327e-01, …,
        4.09386680e-03, 6.84540370e-04, 1.95089020e-02],
       [2.13623375e-01, 1.65229514e-01, 8.33290920e-04, …,
        1.17455767e-02, 1.43176620e-03, 8.13737586e-02],
       …,
       [5.14214523e-02, 1.11942336e-01, 4.20497119e-04, …,
        4.61030602e-02, 1.03383347e-01, 2.07632810e-01],
       [2.85049691e-03, 8.18455871e-03, 4.12059684e-08, …,
        1.56013430e-05, 1.46434204e-05, 9.80562687e-01],
       [7.47538283e-02, 6.79405853e-02, 7.96059612e-05, …,
        3.45701678e-03, 2.10112669e-02, 5.05622327e-01]], dtype=float32)
```

[180]:
```
A=[]
for i in predd:
    A.append(np.argmax(i))
```

[182]:
```
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(T2,A))
```

```
[[52  3  9  1 11 19  0  0  0  5]
 [ 7 54  1  6  5 12  0  6  2  7]
 [ 0  0 96  0  0  0  0  3  1  0]
 [ 0 10  3 31 31 17  1  1  3  3]
 [ 0  0  1  1 91  5  0  0  2  0]
 [ 0  2  3  2 19 70  0  0  4  0]
 [ 1  0  8  0  0  2 60 17 12  0]
 [ 1  1  6  0  0 10 13 63  5  1]
 [ 0  0  1  0  4  0  0  1 94  0]
 [ 4  3  1  1  9  5  2  0 12 63]]
```

[183]:
```
classification_report(T2,A)
```

[183]:
```
'              precision    recall  f1-score   support\n\n           0
0.80      0.52      0.63       100\n           1       0.74      0.54      0.62
100\n           2       0.74      0.96      0.84       100\n           3
0.74      0.31      0.44       100\n           4       0.54      0.91      0.67
100\n           5       0.50      0.70      0.58       100\n           6
0.79      0.60      0.68       100\n           7       0.69      0.63      0.66
100\n           8       0.70      0.94      0.80       100\n           9
0.80      0.63      0.70       100\n\n    accuracy
0.67      1000\n   macro avg       0.70      0.67      0.66      1000\nweighted
avg       0.70      0.67      0.66      1000\n'
```

[184]:
```
print(classification_report(T2,A))
```

	precision	recall	f1-score	support
0	0.80	0.52	0.63	100
1	0.74	0.54	0.62	100
2	0.74	0.96	0.84	100
3	0.74	0.31	0.44	100
4	0.54	0.91	0.67	100
5	0.50	0.70	0.58	100
6	0.79	0.60	0.68	100
7	0.69	0.63	0.66	100
8	0.70	0.94	0.80	100
9	0.80	0.63	0.70	100
accuracy			0.67	1000
macro avg	0.70	0.67	0.66	1000
weighted avg	0.70	0.67	0.66	1000

2 The Accuracy of Model is 67%

[ ]: