Name - Satyapriya

Roll No - 22103057

Kaggle User Name - satyapriya100

Assignment 2

Crack Detection Image Classification Given dataset consists of images obtained from concrete bridge decks, pavements, and walls, images can contain either cracks or no cracks.

Challenge is to develop a binary image classification model to detect cracks in the concrete.

We will use a Transfer learing CNN model to make our predictions.

```
# This command is used to check the availability of GPU on the system
!nvidia-smi
```

```
Fri Apr 14 17:02:33 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   58C    P8    10W /  70W |      3MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Extracting dataset from Kaggle

```
#to install the Kaggle package
!pip install -q kaggle
```

```
#command is used to mount the Google Drive account for linking my google drive for kaggle.json file
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# creating creates a new directory named .kaggle.
! mkdir ~/.kaggle
```

```
#copies the Kaggle API credentials from the Google Drive to the newly created .kaggle directory.
!cp /content/drive/MyDrive/Colab_Notebooks/Kaggle_Credential/kaggle.json ~/.kaggle/
```

```
#command changes the permission of the copied Kaggle API credential file read and write the file
! chmod 600 ~/.kaggle/kaggle.json
```

```
#command to downloads the dataset of the competition "crack-detection-image-classification-2023" from Kaggle.
! kaggle competitions download -q -c crack-detection-image-classification-2023
```

```
#command unzips the downloaded dataset.
! unzip -q crack-detection-image-classification-2023.zip
```

Importing all the Libraries

```
# import the libraries as shown below

import os
import cv2
from glob import glob
import tensorflow as tf   # import TensorFlow
from tensorflow import keras
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten, Conv2D, MaxPooling2D, GlobalAveragePooling2D, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential, load_model
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from pathlib import Path
```

```
#Check the version of TensorFlow you are using
print(tf.__version__)
print(tf.config.list_physical_devices('GPU'))
```

```
     2.12.0
     [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

Creating DataFrames

```
TRAINING_EPOCHS = 20
BATCH_SIZE = 32

image_height = 256
image_width = 256
```

```
from tensorflow.keras.utils import image_dataset_from_directory
base_dir = 'train/'

base_dataset = image_dataset_from_directory(base_dir,
                                            image_size = (image_height, image_width),
                                            crop_to_aspect_ratio = True,
                                            shuffle = False,
                                            batch_size = 32)
```

```
     Found 14968 files belonging to 2 classes.
```

```
base_df = pd.DataFrame(base_dataset.file_paths.copy())
base_df.columns = ['fullpaths']
base_df['labels'] = base_df.apply(lambda x: base_dataset.class_names[1] if (base_dataset.class_names[1] in x.fullpaths)
base_df['filepaths'] = base_df.apply(lambda x: str(x.fullpaths).replace(base_dir, ''), axis=1)
```

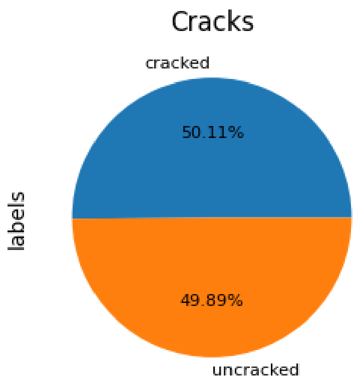
```
base_df.head(-5)
```

	fullpaths	labels	filepaths
0	train/cracked/1000.jpg	cracked	cracked/1000.jpg
1	train/cracked/10000.jpg	cracked	cracked/10000.jpg
2	train/cracked/10003.jpg	cracked	cracked/10003.jpg
3	train/cracked/10004.jpg	cracked	cracked/10004.jpg
4	train/cracked/10005.jpg	cracked	cracked/10005.jpg
...	...	...	...
14958	train/uncracked/9985.jpg	uncracked	uncracked/9985.jpg
14959	train/uncracked/9986.jpg	uncracked	uncracked/9986.jpg
14960	train/uncracked/9988.jpg	uncracked	uncracked/9988.jpg

```
freq = base_df['labels'].value_counts()
print(freq)
```

```
cracked      7501
uncracked    7467
Name: labels, dtype: int64
```

```
freq.plot(kind='pie', figsize=(3,3), title='Cracks', autopct='%1.2f%%',  shadow = False, fontsize=8);
```



Loading Image Data

```
# 80% - train set,
# 10% - validation set,
# 10% - test set

from sklearn.model_selection import train_test_split

train_df, valid_df, test_df = np.split(base_df.sample(frac=1, random_state=42), [int(.8*len(base_df)), int(.9*len(base_d

train_df.head(-5)
```

	fullpaths	labels	filepaths
6560	train/cracked/8308.jpg	cracked	cracked/8308.jpg
1139	train/cracked/1199.jpg	cracked	cracked/1199.jpg
2478	train/cracked/14420.jpg	cracked	cracked/14420.jpg
5747	train/cracked/6782.jpg	cracked	cracked/6782.jpg

```
# Use the Image Data Generator to import the images from the dataset
# Data agumentation and pre-processing using tensorflow
datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.05,
                                   horizontal_flip = True,
                                   vertical_flip   = True,
                                   rotation_range  = 25)

training_set = datagen.flow_from_dataframe(train_df, # dataframe
                                          directory   = base_dir, # images data path / folder in which images are there
                                          x_col       = 'filepaths',
                                          y_col       = 'labels',
                                          color_mode  = "rgb",
                                          target_size = (image_height, image_width), # image height , image width
                                          class_mode  = "categorical",
                                          batch_size  = BATCH_SIZE,
                                          shuffle     = True,
                                          seed        = 42)
```

```
     Found 11974 validated image filenames belonging to 2 classes.
```

```
test_datagen = ImageDataGenerator(rescale = 1./255)

val_set = test_datagen.flow_from_dataframe(valid_df, # dataframe
                                      directory   =   base_dir, # images data path / folder in which images are there
                                      x_col       =   'filepaths',
                                      y_col       =   'labels',
                                      color_mode  =   "rgb",
                                      target_size =   (image_height, image_width), # image height , image width
                                      class_mode  =   "categorical",
                                      batch_size  =   BATCH_SIZE,
                                      shuffle     =   True,
                                      seed        =   42)
```

```
     Found 1497 validated image filenames belonging to 2 classes.
```

```
test_set  =  test_datagen.flow_from_dataframe(test_df, # dataframe
                                              directory   = base_dir, # images data path / folder in which images are th
                                              x_col       = 'filepaths',
                                              y_col       = 'labels',
                                              color_mode  = "rgb",
                                              target_size = (image_height, image_width), # image height , image width
                                              class_mode  = "categorical",
                                              batch_size  = BATCH_SIZE,
                                              shuffle     = False)
```

```
     Found 1497 validated image filenames belonging to 2 classes.
```

```
# Get labels in dataset
a = training_set.class_indices
class_names = list(a.keys())  # storing class names in a list
a
```

```
     {'cracked': 0, 'uncracked': 1}
```

```
def plot_images(img, true_labels, predictions = None):
    plt.figure(figsize=[6, 8])
    for i in range(16):
        plt.subplot(4, 4, i+1)
        plt.imshow(img[i])
        plt.axis('off')
        if (predictions is not None):
            plt.title("{}\n  {} {:.1f}%".format(class_names[np.argmax(true_labels[i])], class_names[np.argmax(prediction
        else:
            plt.title(class_names[np.argmax(true_labels[i])])
```

```
x, y = next(training_set)
plot_images(x, y)
```

Loading Xception model

```
# Import the InceptionV3 library as shown below and add preprocessing layer to the front of InceptionV3
# Here we will be using imagenet weights
# command creates an InceptionV3 model with the specified input shape and pre-trained weights.
xception_base_model = tf.keras.applications.xception.Xception(include_top = False,
                                                              weights = 'imagenet',
                                                              input_shape = (image_height, image_width, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_di
83683744/83683744 [==============================] - 1s 0us/step
```

```
xception_base_model.summary()
```

```
 block13_sepconv1_bn (BatchNorm  (None, 16, 16, 728)  2912       ['block13_sepconv1[0][0]']
 alization)

 block13_sepconv2_act (Activati  (None, 16, 16, 728)  0          ['block13_sepconv1_bn[0][0]']
 on)

 block13_sepconv2 (SeparableCon  (None, 16, 16, 1024  752024     ['block13_sepconv2_act[0][0]']
 v2D)                           )

 block13_sepconv2_bn (BatchNorm  (None, 16, 16, 1024  4096       ['block13_sepconv2[0][0]']
 alization)                     )

 conv2d_3 (Conv2D)              (None, 8, 8, 1024)   745472     ['add_10[0][0]']

 block13_pool (MaxPooling2D)    (None, 8, 8, 1024)   0          ['block13_sepconv2_bn[0][0]']

 batch_normalization_3 (BatchNo  (None, 8, 8, 1024)  4096       ['conv2d_3[0][0]']
 rmalization)

 add_11 (Add)                   (None, 8, 8, 1024)   0          ['block13_pool[0][0]',
                                                                 'batch_normalization_3[0][0]']

 block14_sepconv1 (SeparableCon  (None, 8, 8, 1536)  1582080    ['add_11[0][0]']
 v2D)

 block14_sepconv1_bn (BatchNorm  (None, 8, 8, 1536)  6144       ['block14_sepconv1[0][0]']
 alization)

 block14_sepconv1_act (Activati  (None, 8, 8, 1536)  0          ['block14_sepconv1_bn[0][0]']
 on)

 block14_sepconv2 (SeparableCon  (None, 8, 8, 2048)  3159552    ['block14_sepconv1_act[0][0]']
 v2D)

 block14_sepconv2_bn (BatchNorm  (None, 8, 8, 2048)  8192       ['block14_sepconv2[0][0]']
 alization)

 block14_sepconv2_act (Activati  (None, 8, 8, 2048)  0          ['block14_sepconv2_bn[0][0]']
 on)

==================================================================================================
Total params: 20,861,480
Trainable params: 20,806,952
Non-trainable params: 54,528
__________________________________________________________________________________________________
```

```
def create_model(base_model):

    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation = 'relu')(x)
    x = Dropout(0.4)(x)
    x = Dense(64, activation = 'relu')(x)
    x = Dropout(0.2)(x)

    outputs = Dense(len(class_names), activation='softmax')(x)

    model = Model(base_model.inputs, outputs)

    return model
```

```
xception_model = create_model(xception_base_model)
```

```
xception_model.summary()
```

```
 conv2d_3 (Conv2D)              (None, 8, 8, 1024)   745472      ['add_10[0][0]']

 block13_pool (MaxPooling2D)    (None, 8, 8, 1024)   0           ['block13_sepconv2_bn[0][0]']

 batch_normalization_3 (BatchNo  (None, 8, 8, 1024)  4096        ['conv2d_3[0][0]']
 rmalization)

 add_11 (Add)                   (None, 8, 8, 1024)   0           ['block13_pool[0][0]',
                                                                  'batch_normalization_3[0][0]']

 block14_sepconv1 (SeparableCon  (None, 8, 8, 1536)  1582080     ['add_11[0][0]']
 v2D)

 block14_sepconv1_bn (BatchNorm  (None, 8, 8, 1536)  6144        ['block14_sepconv1[0][0]']
 alization)

 block14_sepconv1_act (Activati  (None, 8, 8, 1536)  0           ['block14_sepconv1_bn[0][0]']
 on)

 block14_sepconv2 (SeparableCon  (None, 8, 8, 2048)  3159552     ['block14_sepconv1_act[0][0]']
 v2D)

 block14_sepconv2_bn (BatchNorm  (None, 8, 8, 2048)  8192        ['block14_sepconv2[0][0]']
 alization)

 block14_sepconv2_act (Activati  (None, 8, 8, 2048)  0           ['block14_sepconv2_bn[0][0]']
 on)

 global_average_pooling2d (Glob  (None, 2048)        0           ['block14_sepconv2_act[0][0]']
 alAveragePooling2D)

 dense (Dense)                  (None, 128)          262272      ['global_average_pooling2d[0][0]'
                                                                 ]

 dropout (Dropout)              (None, 128)          0           ['dense[0][0]']

 dense_1 (Dense)                (None, 64)           8256        ['dropout[0][0]']

 dropout_1 (Dropout)            (None, 64)           0           ['dense_1[0][0]']

 dense_2 (Dense)                (None, 2)            130         ['dropout_1[0][0]']

==================================================================================================
Total params: 21,132,138
Trainable params: 21,077,610
Non-trainable params: 54,528
__________________________________________________________________________________________________
```

Training Data

```
def fit_model(model, base_model, epochs, fine_tune = 0):
    # early stopping call back
    # monitors the validation loss during training and stops the training early
    es = tf.keras.callbacks.EarlyStopping(
                          monitor='val_loss',
                          min_delta = 0.02,
                          patience=6,
                          verbose=0,
                          mode='auto',
                          baseline=None,
                          start_from_epoch=10,
                          restore_best_weights=True)

    # saves the best model during training based on the validation loss
    model_cp = tf.keras.callbacks.ModelCheckpoint(filepath = 'best_model.h5',
                                              monitor='val_loss',
                                              save_best_only = True,
                                              verbose=1)


    # Defines how many layers to freeze during training.
    # Layers in the convolutional base are switched from trainable to non-trainable
    # depending on the size of the fine-tuning parameter.
    print("Training number of layers in model = ", fine_tune)

    if fine_tune > 0:
        base_model.trainable = True
        for layer in base_model.layers[:fine_tune]:
            layer.trainable = False
        # small learning rate for fine tuning
        # tell the model what cost and optimization method to use
        model.compile(optimizer=tf.keras.optimizers.Adam(1e-5),
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])
        for layer in model.layers:
          print(layer.name,layer.trainable)
    else:
        base_model.trainable = False
        # tell the model what cost and optimization method to use
        model.compile(optimizer=tf.keras.optimizers.Adam(),
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])
        for layer in model.layers:
          print(layer.name,layer.trainable)

    # fit the model
    # Run the cell. It will take some time to execute
    history = model.fit(training_set,
                        validation_data=val_set,
                        epochs= epochs,
                        steps_per_epoch=len(training_set),
                        validation_steps=len(val_set),
                        callbacks=[es,model_cp])

    return history
```

```
base_layers = len(xception_base_model.layers)
print("xception base layers = ", base_layers)
```

```
xception base layers =  132
```

```
r = fit_model(xception_model,
              xception_base_model,
              epochs = TRAINING_EPOCHS,
              fine_tune = int(base_layers/5))
```

```
375/375 [==============================] - 244s 649ms/step - loss: 0.3489 - accuracy: 0.8518 - val_loss: 0.2978
Epoch 7/20
375/375 [==============================] - ETA: 0s - loss: 0.3404 - accuracy: 0.8517
Epoch 7: val_loss improved from 0.29781 to 0.29135, saving model to best_model.h5
375/375 [==============================] - 243s 649ms/step - loss: 0.3404 - accuracy: 0.8517 - val_loss: 0.2913
Epoch 8/20
```

```
375/375 [==============================] - ETA: 0s - loss: 0.2923 - accuracy: 0.8765
Epoch 14: val_loss did not improve from 0.26884
375/375 [==============================] - 242s 644ms/step - loss: 0.2923 - accuracy: 0.8765 - val_loss: 0.2756
Epoch 15/20
375/375 [==============================] - ETA: 0s - loss: 0.2847 - accuracy: 0.8793
Epoch 15: val_loss improved from 0.26884 to 0.26853, saving model to best_model.h5
375/375 [==============================] - 242s 644ms/step - loss: 0.2847 - accuracy: 0.8793 - val_loss: 0.2685
Epoch 16/20
375/375 [==============================] - ETA: 0s - loss: 0.2855 - accuracy: 0.8783
Epoch 16: val_loss did not improve from 0.26853
375/375 [==============================] - 241s 642ms/step - loss: 0.2855 - accuracy: 0.8783 - val_loss: 0.2721
Epoch 17/20
375/375 [==============================] - ETA: 0s - loss: 0.2765 - accuracy: 0.8828
Epoch 17: val_loss did not improve from 0.26853
375/375 [==============================] - 241s 642ms/step - loss: 0.2765 - accuracy: 0.8828 - val_loss: 0.2717
Epoch 18/20
375/375 [==============================] - ETA: 0s - loss: 0.2792 - accuracy: 0.8814
Epoch 18: val_loss did not improve from 0.26853
375/375 [==============================] - 242s 645ms/step - loss: 0.2792 - accuracy: 0.8814 - val_loss: 0.2711
Epoch 19/20
375/375 [==============================] - ETA: 0s - loss: 0.2733 - accuracy: 0.8858
Epoch 19: val_loss did not improve from 0.26853
375/375 [==============================] - 242s 644ms/step - loss: 0.2733 - accuracy: 0.8858 - val_loss: 0.2714
Epoch 20/20
375/375 [==============================] - ETA: 0s - loss: 0.2680 - accuracy: 0.8856
Epoch 20: val_loss did not improve from 0.26853
```

```
xception_model.summary()
```

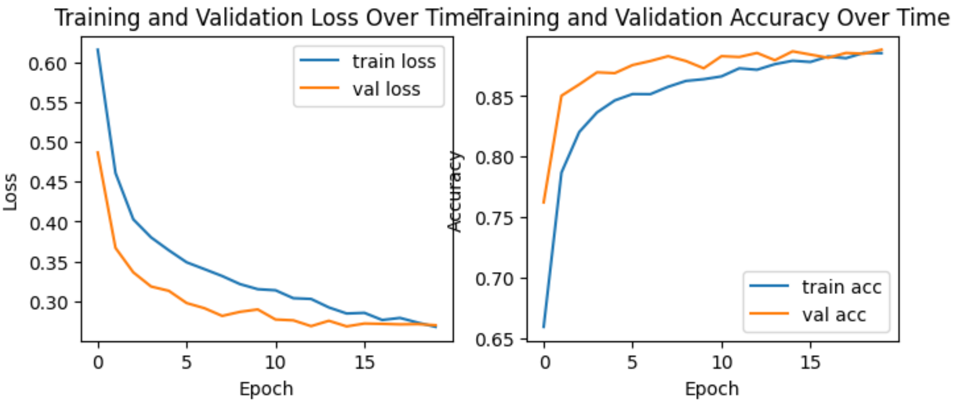
```
 dropout_1 (Dropout)            (None, 64)           0           ['dense_1[0][0]']

 dense_2 (Dense)                (None, 2)            130         ['dropout_1[0][0]']

==================================================================================================
Total params: 21,132,138
Trainable params: 20,886,794
Non-trainable params: 245,344
__________________________________________________________________________________________________
```

```
# from tensorflow.keras.models import load_model

# # load best model
# model = load_model('best_model.h5')
```

Plots

```
# creating a function for ploting the loss and Accuracy

def plot_model(r):

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 3))

    ax1.plot(r.history['loss'], label='train loss')
    ax1.plot(r.history['val_loss'], label='val loss')
    ax1.set_xlabel('Epoch')
    ax1.set_ylabel('Loss')
    ax1.legend()
    ax1.set_title("Training and Validation Loss Over Time")


    ax2.plot(r.history['accuracy'], label='train acc')
    ax2.plot(r.history['val_accuracy'], label='val acc')
    ax2.set_xlabel('Epoch')
    ax2.set_ylabel('Accuracy')
    ax2.legend()
    ax2.set_title("Training and Validation Accuracy Over Time")

    plt.show()
    plt.savefig('Plot')
```

```
plot_model(r)
```

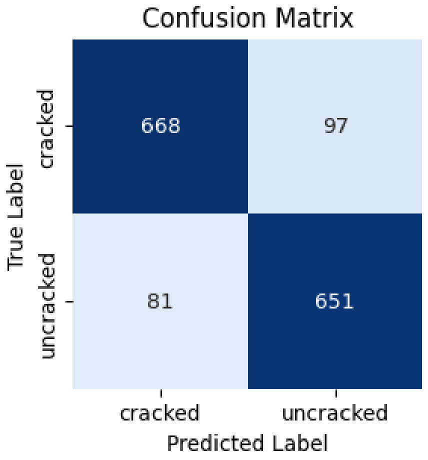


```
<Figure size 640x480 with 0 Axes>
```

Results

```
# Results
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import f1_score

def evaluate_model(model, test_data):

    results = model.evaluate(test_data, verbose=0)
    loss = results[0]
    acc = results[1]

    print("    Test Loss: {:.5f}".format(loss))
    print("Test Accuracy: {:.2f} %".format(acc * 100))

    y_true = test_data.classes

    y_pred = np.argmax(model.predict(test_data), axis=1)
    cm = confusion_matrix(y_true, y_pred)
    print(cm)
    clr = classification_report(test_data.labels, y_pred, target_names=class_names) #["POSITIVE", "NEGATIVE"])

    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, square=True, fmt='g', vmin=0, cmap='Blues', cbar=False)
    plt.xticks(ticks=np.arange(2) + 0.5, labels=class_names)
    plt.yticks(ticks=np.arange(2) + 0.5, labels=class_names)
    plt.xlabel("Predicted Label", fontsize= 10)
    plt.ylabel("True Label", fontsize= 10)
    plt.title("Confusion Matrix")
    plt.show()

    print("Classification Report:\n----------------------\n", clr)

    f1 = f1_score(test_data.labels, y_pred)
    print(f1)



evaluate_model(xception_model, test_set)
```

```
    Test Loss: 0.28024
Test Accuracy: 88.11 %
47/47 [==============================] - 9s 175ms/step
[[668  97]
 [ 81 651]]
```



```
Classification Report:
----------------------
               precision    recall  f1-score   support

     cracked       0.89      0.87      0.88       765
   uncracked       0.87      0.89      0.88       732

    accuracy                           0.88      1497
   macro avg       0.88      0.88      0.88      1497
weighted avg       0.88      0.88      0.88      1497

0.8797297297297298
```

Predictions

```
test_path = 'test'
test_filenames = os.listdir(test_path)


test_predictions = np.array([])

for img in os.listdir(test_path):
    image_path = os.path.join(test_path,img)
    image_path
    image = tf.keras.utils.load_img(image_path, target_size = (256,256))
    image = tf.keras.utils.img_to_array(image)
    image = np.array([image])
    image = image/255
    y_pred = xception_model.predict(image)
    y_pred_classes = np.argmax(y_pred)
    test_predictions = np.append(test_predictions, class_names[y_pred_classes])
    #break
#print(test_predictions[0])
#print(test_labels[0])
```

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
```

```
1/1 [==============================] - 0s 22ms/step
```

Output file

```
# Create a DataFrame with the filename and predicted values
results_df = pd.DataFrame({
    "filename": test_filenames,
    "class": test_predictions
})

# Save the DataFrame to a CSV file
results_df.to_csv("A2_22103057_Satyapriya.csv", index=False)
```

Uploading to Kaggle

```
! kaggle competitions submit -c crack-detection-image-classification-2023 -f /content/A2_22103057_Satyapriya.csv -m test
```

```
100% 34.2k/34.2k [00:01<00:00, 22.3kB/s]
Successfully submitted to Crack Detection: Image Classification 2023
```

```
! kaggle competitions submissions -c crack-detection-image-classification-2023
```

```
fileName                         date                 description                     status    publicScore  privateS
-------------------------------  -------------------  ------------------------------  --------  -----------  --------
A2_22103057_Satyapriya.csv       2023-04-14 18:38:52  test_submission_1               complete  0.89000
newfile                          2023-04-14 17:03:52  Notebook newfile | Version 2    complete  0.80200
submission.csv                   2023-04-14 16:50:00  new file                        complete  0.80100
A2_22103057_Satyapriya.csv       2023-04-13 14:53:16  test_submission_1               complete  0.78300
A2_22103057_Satyapriya.csv       2023-04-12 16:05:45  test_submission_1               complete  0.81400
A2_22103057_Satyapriya (1).csv   2023-04-12 16:00:49  new file                        complete  0.75600
A2_22103057_Satyapriya.csv       2023-04-12 14:01:43  test_submission_1               complete  0.76900
A2_22103057_Satyapriya.csv       2023-04-11 19:10:03  test_submission_1               complete  0.54200
A2_22103057_Satyapriya.csv       2023-04-11 18:51:18  test_submission_1               complete  0.48400
A2_22103057_Satyapriya.csv       2023-04-11 18:35:51  test_submission_1               complete  0.56200
A2_22103057_Satyapriya.csv       2023-04-11 17:08:07  test_submission_1               complete  0.74500
A2_22103057_Satyapriya.csv       2023-04-11 16:04:44  test_submission_1               complete  0.74700
A2_22103057_Satyapriya.csv       2023-04-11 15:38:54  second submission               complete  0.67300
A2_22103057_Satyapriya.csv       2023-04-11 15:29:43  first prediction                complete  0.32700
```

```
! kaggle competitions leaderboard -s -c crack-detection-image-classification-2023
```

```
  teamId  teamName               submissionDate       score
--------  ---------------------  -------------------  -------
10200611  Satyapriya             2023-04-14 18:38:52  0.89000
10193103  Akshit Singh Chauhan   2023-04-09 11:30:29  0.88400
10188189  Shubhi Kant            2023-04-12 17:33:42  0.88000
10181370  Padam Sharma           2023-04-06 17:34:30  0.87000
10188870  Ayush Gupta            2023-04-10 11:15:23  0.84800
10191342  Huzaifa0498            2023-04-12 20:26:15  0.81000
10182251  apsingh007             2023-04-10 20:12:42  0.80900
10181135  LALIT CHOUDHARY        2023-04-13 06:51:31  0.79400
10211836  ABHISHEK_MOURYA_04     2023-04-14 17:31:25  0.79400
10185034  Gowri Naidu            2023-04-11 17:30:37  0.76800
10194748  Ranjan9779             2023-04-13 20:21:06  0.73100
10182822  priyanshu maddheshiya  2023-04-13 16:02:26  0.71400
10219377  Analyst573             2023-04-14 12:53:59  0.71400
10215427  SATISH KUMAR           2023-04-14 14:04:03  0.65600
10217667  Sawarmal               2023-04-14 10:48:19  0.61900
10203077  Rahul Taank            2023-04-13 08:54:37  0.59800
10197145  Azad prajapat          2023-04-14 15:23:17  0.55200
10181314  Kula vardhan Reddy     2023-04-11 06:05:50  0.52100
10182657  Pankajkmr22            2023-04-14 10:22:47  0.51600
10191735  Nitin Jangir           2023-04-11 05:26:34  0.51300
10185744  anay nagar             2023-04-09 14:16:54  0.50600
10185064  Gourav Jaiswal         2023-04-10 12:45:12  0.48500
10180964  himanshu berad         2023-04-09 07:51:23  0.48400
10188567  Dhanya Sagar           2023-04-11 12:47:34  0.47700
10202643  divyavani gunturu      2023-04-14 13:32:11  0.45300
```

✓ 0s completed at 12:08 AM ● ×