```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import os
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.applications import DenseNet121
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout, MaxPooling2D, Con
from tensorflow.keras.models import Sequential
from IPython.display import clear_output
import warnings
warnings.filterwarnings('ignore')
```

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.13
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (fron
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kag
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (
```

```
!mkdir ~/.kaggle
```

```
!cp kaggle.json ~/.kaggle
```

```
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d odins0n/ucf-crime-dataset
```

```
Downloading ucf-crime-dataset.zip to /content
100% 11.0G/11.0G [02:53<00:00, 73.1MB/s]
100% 11.0G/11.0G [02:53<00:00, 68.2MB/s]
```

```
!unzip ucf-crime-dataset.zip
```

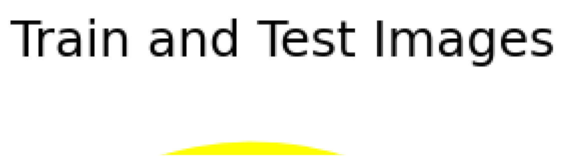
```
  inflating: Train/Vandalism/Vandalism050_x264_870.png
  inflating: Train/Vandalism/Vandalism050_x264_880.png
  inflating: Train/Vandalism/Vandalism050_x264_890.png
  inflating: Train/Vandalism/Vandalism050_x264_90.png
```

```
train_dir = "/content/Train"
test_dir = "/content/Test"
SEED = 12
IMG_HEIGHT = 64
IMG_WIDTH = 64
BATCH_SIZE = 128
EPOCHS = 5
LR = 0.00003
crime_types=os.listdir(train_dir)
n=len(crime_types)
print("Number of crime categories : ",n)
```

```
Number of crime categories :  14
```

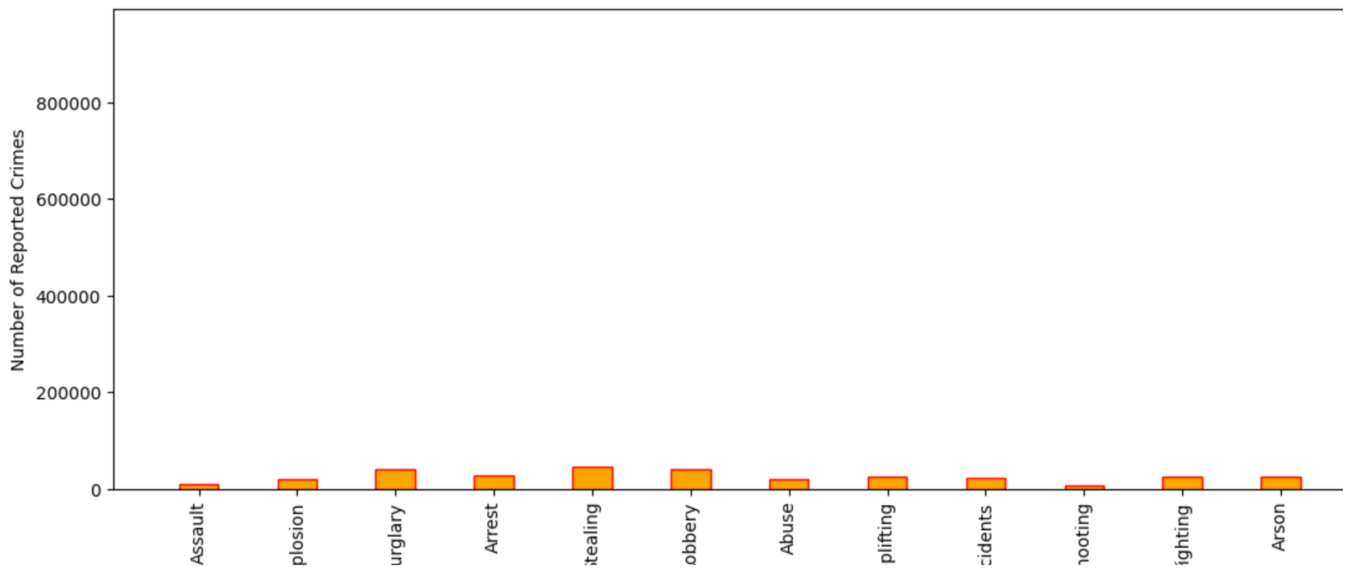
```
crimes={}
train=test=0
for clss in crime_types:
    num=len(os.listdir (os.path.join(train_dir,clss)))
    train+=num
    test+=len(os.listdir (os.path.join(test_dir,clss)))
    crimes[clss]=num


plt.figure(figsize=(8, 5))
plt.pie(x=np.array([train, test]), autopct="%.1f%%", explode=[0.1, 0.1], labels=["Training Da
plt.title("Train and Test Images", fontsize=18);
```



```
plt.figure(figsize=(15,5))
plt.bar(list(crimes.keys()), list (crimes.values()), width=0.4, align="center",edgecolor=["re
plt.xticks(rotation=90)


plt.xlabel("Reported Crimes")
plt.ylabel("Number of Reported Crimes")
plt.show()
```



```
IMG_SHAPE=(64,64)
#Apply Image_Dataset_from_directory Functionality To Train Set And Test Set
train_set=image_dataset_from_directory(
    train_dir,
    label_mode="categorical",
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=True,
    seed=SEED,
    validation_split=0.2,
    subset="training",
)



val_set=image_dataset_from_directory(
    train_dir,
    label_mode="categorical",
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=True,
    seed=SEED,
    validation_split =0.2,
    subset="validation",
)


test_set=image_dataset_from_directory(
    test_dir,
    label_mode="categorical",
    class_names=None,
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=False,
    seed=SEED,
)
```

```
Found 1266345 files belonging to 14 classes.
Using 1013076 files for training.
Found 1266345 files belonging to 14 classes.
Using 253269 files for validation.
Found 111308 files belonging to 14 classes.
```

```
# Create Transfer Learning Function
INPUT_SHAPE=(64,64,3)
def transfer_learning():
    base_model=DenseNet121(include_top=False, input_shape=INPUT_SHAPE, weights="imagenet")

    thr=149

    for layers in base_model.layers[:thr]:
        layers.trainable=False

    for layers in base_model.layers[thr:]:
        layers.trainable=False

    return base_model


# Adding Dense Layers

def create_model():
    model=Sequential()

    base_model=transfer_learning()
    model.add(base_model)

    model.add(GlobalAveragePooling2D())

    model.add(Dense(256, activation="relu"))
    model.add(Dropout(0.2))

    model.add(Dense(512, activation="relu"))
    model.add(Dropout (0.2))

    model.add(Dense(1024, activation="relu"))

    model.add(Dense (n, activation="softmax"))

    model.summary()

    return model



model=create_model()

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/dense
29084464/29084464 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 densenet121 (Functional)    (None, 2, 2, 1024)        7037504   
                                                                 
 global_average_pooling2d (G  (None, 1024)             0         
 lobalAveragePooling2D)                                          
                                                                 
 dense (Dense)               (None, 256)               262400    
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 dense_1 (Dense)             (None, 512)               131584    
                                                                 
 dropout_1 (Dropout)         (None, 512)               0         
                                                                 
 dense_2 (Dense)             (None, 1024)              525312    
                                                                 
 dense_3 (Dense)             (None, 14)                14350     
                                                                 
=================================================================
Total params: 7,971,150
Trainable params: 933,646
Non-trainable params: 7,037,504
_________________________________________________________________
```

```
# Configure The Learning Process
```

```
model=create_model()

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

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 2, 2, 1024)        7037504

 global_average_pooling2d_1   (None, 1024)             0
 (GlobalAveragePooling2D)

 dense_4 (Dense)             (None, 256)               262400

 dropout_2 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 512)               131584

 dropout_3 (Dropout)         (None, 512)               0

 dense_6 (Dense)             (None, 1024)              525312

 dense_7 (Dense)             (None, 14)                14350

=================================================================
Total params: 7,971,150
Trainable params: 933,646
Non-trainable params: 7,037,504
_________________________________________________________________
```

```
# Train The Model
history = model.fit(x = train_set,
                    validation_data=val_set,
                    epochs = 5)
```

```
# Save model
model.save('crime.h5')
```

```
#Load the saved model using load_model

from tensorflow.keras.models import load_model
model=load_model('crime.h5')
```

```
model.load_weights('crime.h5')
```

```
y_true = np.array([])
for x, y in test_set:
    y_true = np.concatenate([y_true, np.argmax(y.numpy(), axis=-1)])
```

```
y_pred=model.predict(test_set)
```

```
     870/870 [==============================] - 1027s 1s/step
```

```
from tensorflow.keras.preprocessing import image

# Testing 1
img = image.load_img("/content/Test/Burglary/Burglary005_x264_1030.png",target_size=(64,64))
x = image.img_to_array(img) # Converting Image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax (model.predict(x)) # Predicting the higher probablity Index
op = ['Fighthing', 'Arrest', 'Vandalism', 'Assault', 'Stealing' , 'Arson', 'Normalvideos', 'B
op[pred] # tist indexing with output
```

```
     1/1 [==============================] - 0s 40ms/step
     'Burglary'
```

```
# Testing 2
img = image.load_img("/content/Test/Fighting/Fighting003_x264_1020.png",target_size
x = image.img_to_array(img) # Converting Image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax (model.predict(x)) # Predicting the higher probablity Index
op = ['Fighthing', 'Arrest', 'Vandalism', 'Assault', 'Stealing' , 'Arson', 'Normalv
op[pred] # tist indexing with output
```

```
     1/1 [==============================] - 0s 41ms/step
     'Robbery'
```

```
# Testing 3
img = image.load_img("/content/Test/NormalVideos/Normal_Videos_003_x264_1020.png",t
x = image.img_to_array(img) # Converting Image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax (model.predict(x)) # Predicting the higher probablity Index
op = ['Fighthing', 'Arrest', 'Vandalism', 'Assault', 'Stealing' , 'Arson', 'Normalv
op[pred] # tist indexing with output
```

```
     1/1 [==============================] - 4s 4s/step
     'Vandalism'
```

✓ 3s completed at 9:34 PM ● ✕