In [1]:
```
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 cv2

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, LSTM, Dense, BatchNormalization
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import roc_curve, auc, roc_auc_score


from IPython.display import clear_output
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
train_dir =r"C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\vad\train"
test_dir =r"C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\vad\test"

SEED = 12
IMG_HEIGHT = 64
IMG_WIDTH = 64
BATCH_SIZE = 64
EPOCHS = 1
LR =  0.00003
NUM_CLASSES = 4
CLASS_LABELS = ['Abuse','Explosion','Fighting',"Normal"]
```

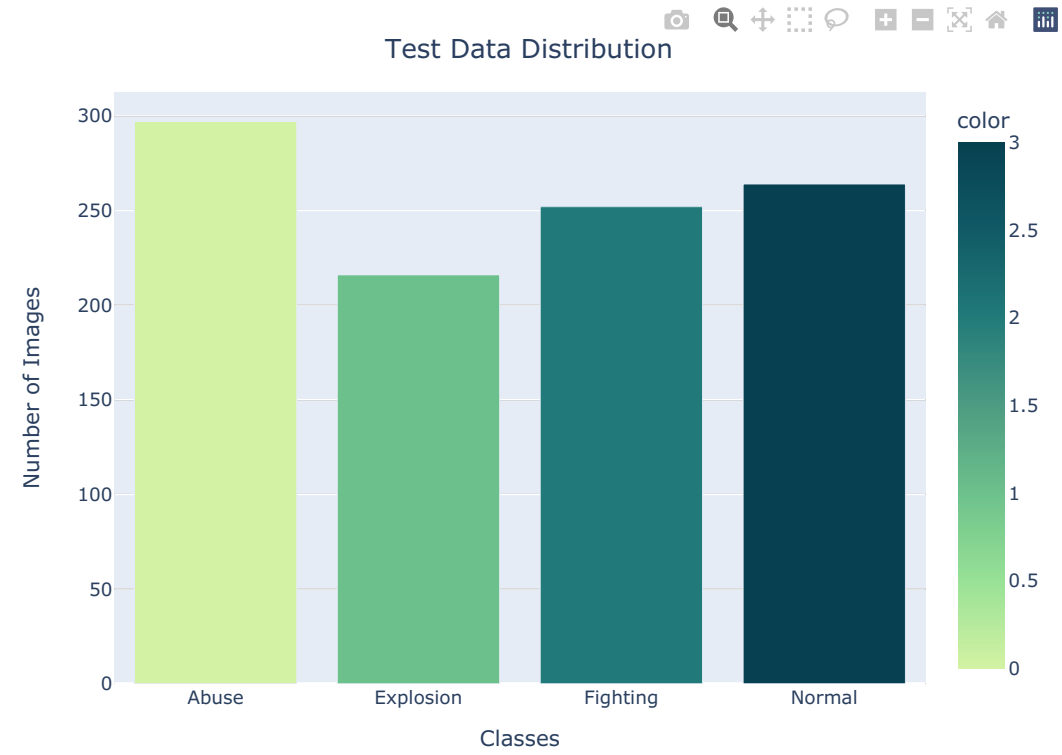
In [3]:
```
preprocess_fun = tf.keras.applications.densenet.preprocess_input

train_datagen = ImageDataGenerator(horizontal_flip=True,
                                   width_shift_range=0.1,
                                   height_shift_range=0.05,
                                   rescale = 1./255,
                                   preprocessing_function=preprocess_fun
                                  )
test_datagen = ImageDataGenerator(rescale = 1./255,
                                  preprocessing_function=preprocess_fun
                                 )
```

In [4]:
```
train_generator = train_datagen.flow_from_directory(directory = train_dir,
                                                    target_size = (IMG_HEIGHT ,IMG_WIDTH),
                                                    batch_size = BATCH_SIZE,
                                                    shuffle  = True ,
                                                    color_mode = "rgb",
                                                    class_mode = "categorical",
                                                    seed = SEED
                                                   )
test_generator = test_datagen.flow_from_directory(directory = test_dir,
                                                  target_size = (IMG_HEIGHT ,IMG_WIDTH),
                                                   batch_size = BATCH_SIZE,
                                                   shuffle  = False ,
                                                   color_mode = "rgb",
                                                   class_mode = "categorical",
                                                   seed = SEED
                                                  )
```

```
Found 3057 images belonging to 4 classes.
Found 1029 images belonging to 4 classes.
```

In [5]:
```
fig = px.bar(x = CLASS_LABELS,
             y = [list(train_generator.classes).count(i) for i in np.unique(train_generator.classes)] ,
             color = np.unique(train_generator.classes) ,
             color_continuous_scale="Emrld")
fig.update_xaxes(title="Classes")
fig.update_yaxes(title = "Number of Images")
fig.update_layout(showlegend = True,
    title = {
        'text': 'Train Data Distribution ',
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
fig.show()
```

In [6]:
```
fig = px.bar(x = CLASS_LABELS,
             y = [list(test_generator.classes).count(i) for i in np.unique(test_generator.classes)] ,
             color = np.unique(train_generator.classes) ,
             color_continuous_scale="Emrld")
fig.update_xaxes(title="Classes")
fig.update_yaxes(title = "Number of Images")
fig.update_layout(showlegend = True,
    title = {
        'text': 'Test Data Distribution ',
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
fig.show()
```



In [7]:
```
def feature_extractor(inputs):
    feature_extractor = tf.keras.applications.DenseNet121(input_shape=(IMG_HEIGHT,IMG_WIDTH, 3),
                                                 include_top=False,
                                                 weights="imagenet")(inputs)
```

```
    return feature_extractor

def classifier(inputs):
    x = tf.keras.layers.GlobalAveragePooling2D()(inputs)
    x = tf.keras.layers.Dense(256, activation="relu")(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(1024, activation="relu")(x)
    x = tf.keras.layers.Dropout(0.5)(x)
    x = tf.keras.layers.Dense(512, activation="relu")(x)
    x = tf.keras.layers.Dropout(0.4) (x)
    x = tf.keras.layers.Dense(NUM_CLASSES, activation="softmax", name="classification")(x)

    return x

def final_model(inputs):
    densenet_feature_extractor = feature_extractor(inputs)
    classification_output = classifier(densenet_feature_extractor)

    return classification_output

def define_compile_model():

    inputs = tf.keras.layers.Input(shape=(IMG_HEIGHT ,IMG_WIDTH,3))
    classification_output = final_model(inputs)
    model = tf.keras.Model(inputs=inputs, outputs = classification_output)

    model.compile(optimizer=tf.keras.optimizers.SGD(LR),
                loss='categorical_crossentropy',
                metrics = [tf.keras.metrics.AUC()])

    return model

model = define_compile_model()
clear_output()
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 64, 64, 3)]       0

 densenet121 (Functional)    (None, 2, 2, 1024)        7037504

 global_average_pooling2d (  (None, 1024)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               262400

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1024)              263168

 dropout_1 (Dropout)         (None, 1024)              0

 dense_2 (Dense)             (None, 512)               524800

 dropout_2 (Dropout)         (None, 512)               0

 classification (Dense)      (None, 4)                 2052

=================================================================
Total params: 8089924 (30.86 MB)
Trainable params: 8006276 (30.54 MB)
Non-trainable params: 83648 (326.75 KB)
_________________________________________________________________
```

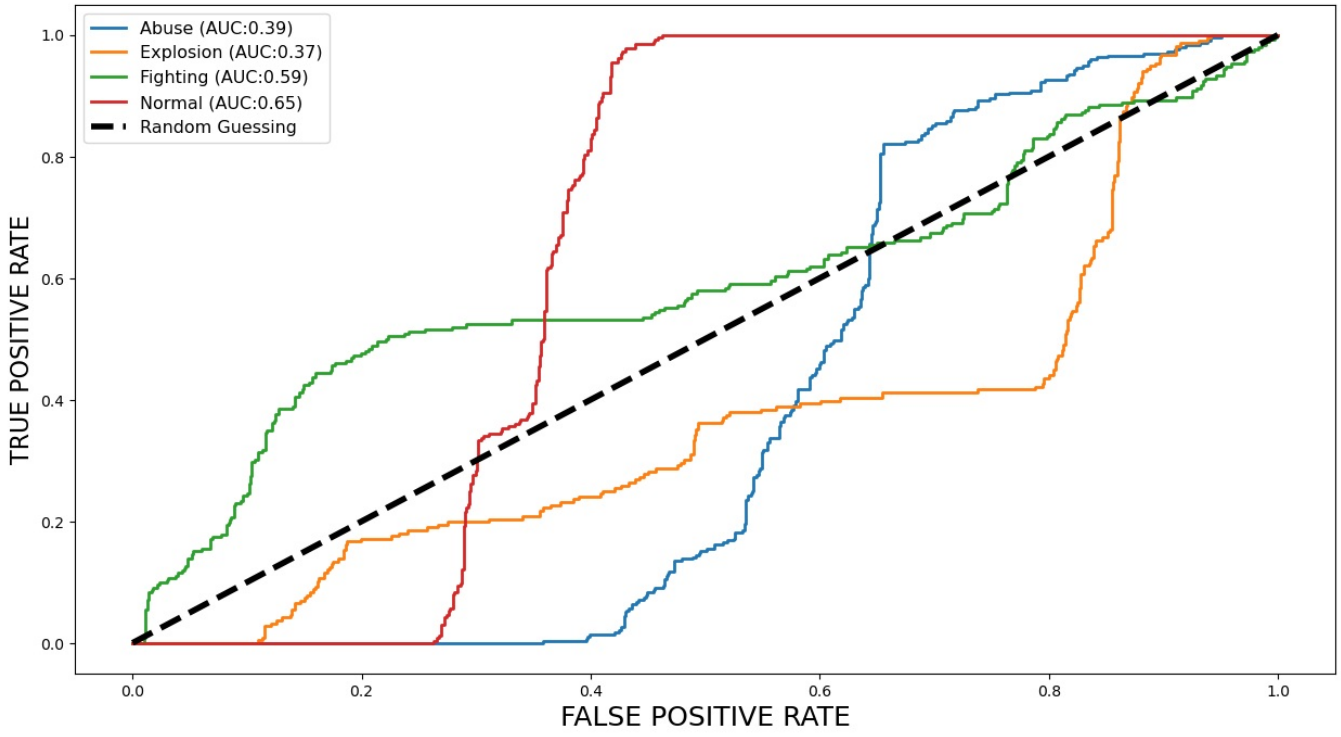
In [8]:
```
history = model.fit(x = train_generator,validation_data=test_generator,epochs = EPOCHS)
```

```
48/48 [==============================] - 218s 4s/step - loss: 1.7233 - auc: 0.5185 - val_loss: 1.5689 - val_auc
: 0.4769
```

In [9]:
```
preds = model.predict(test_generator)
y_test = test_generator.classes
fig, c_ax = plt.subplots(1,1, figsize = (15,8))

def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    for (idx, c_label) in enumerate(CLASS_LABELS):
        fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), y_pred[:,idx])
        c_ax.plot(fpr, tpr,lw=2, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
    c_ax.plot(fpr, fpr, 'black',linestyle='dashed', lw=4, label = 'Random Guessing')
    return roc_auc_score(y_test, y_pred, average=average)

print('ROC AUC score:', multiclass_roc_auc_score(y_test , preds  , average = "micro"))
plt.xlabel('FALSE POSITIVE RATE', fontsize=18)
plt.ylabel('TRUE POSITIVE RATE', fontsize=16)
```

```
plt.legend(fontsize = 11.5)
plt.show()
```

```
17/17 [==============================] - 18s 780ms/step
ROC AUC score: 0.47789233699866174
```



In [15]:
```
import smtplib
def send_email_alert(anomaly):
    from_email ="2004081ece@cit.edu.in"
    to_email = "mrrahul21333@gmail.com"
    password = "#cit@21333!"



    subject = "Anomaly Detected in Model Performance"
    message = "The detected anomaly is "+ str(anomaly)

    try:
        server = smtplib.SMTP("smtp.gmail.com", 587)
        server.starttls()
        server.login(from_email, password)
        server.sendmail(from_email, to_email, f"Subject: {subject}\n\n{message}")
        server.quit()
        print("Alert email sent successfully.")
    except Exception as e:
        print("Error sending email:", str(e))
```

In [16]:
```
input_image_dir = r"C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred"
model = define_compile_model()
class_labels = ['Abuse', 'Explosion', 'Fighting', 'Normal']
predictions = []

for image_filename in os.listdir(input_image_dir):

    image_path = os.path.join(input_image_dir, image_filename)
    img = cv2.imread(image_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (IMG_WIDTH, IMG_HEIGHT))
    img = img / 255.0

    prediction = model.predict(np.expand_dims(img, axis=0))
    predicted_class = class_labels[np.argmax(prediction)]
    predictions.append({
        "image_path": image_path,
        "predicted_class": predicted_class,
        "class_probabilities": {class_labels[i]: float(prediction[0][i]) for i in range(len(class_labels))}
    })


for prediction in predictions:
    print(f"Image: {prediction['image_path']}")
    print(f"Predicted Class: {prediction['predicted_class']}")
    print("Class Probabilities:")
    for label, probability in prediction['class_probabilities'].items():
        print(f"{label}: {probability:.4f}")
    send_email_alert(predicted_class)
    print("\n")
```

```
1/1 [==============================] - 5s 5s/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 89ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 104ms/step
Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img1.png
Predicted Class: Explosion
Class Probabilities:
Abuse: 0.3694
Explosion: 0.4253
Fighting: 0.1137
Normal: 0.0916
Alert email sent successfully.


Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img2.png
Predicted Class: Abuse
Class Probabilities:
Abuse: 0.4750
Explosion: 0.4745
Fighting: 0.0098
Normal: 0.0407
Alert email sent successfully.


Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img3.png
Predicted Class: Abuse
Class Probabilities:
Abuse: 0.3959
Explosion: 0.2681
Fighting: 0.0458
Normal: 0.2902
Alert email sent successfully.


Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img4.png
Predicted Class: Abuse
Class Probabilities:
Abuse: 0.3769
Explosion: 0.2233
Fighting: 0.1049
Normal: 0.2948
Alert email sent successfully.


Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img5.png
Predicted Class: Abuse
Class Probabilities:
Abuse: 0.8223
Explosion: 0.1159
Fighting: 0.0256
Normal: 0.0362
Alert email sent successfully.


Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img6.png
Predicted Class: Explosion
Class Probabilities:
Abuse: 0.3161
Explosion: 0.3662
Fighting: 0.1026
Normal: 0.2151
Alert email sent successfully.


Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img7.png
Predicted Class: Abuse
Class Probabilities:
Abuse: 0.4259
Explosion: 0.1963
Fighting: 0.1274
Normal: 0.2504
Alert email sent successfully.


Image: C:\Users\RAHUL M\Desktop\Mini Project\mini project 2\pred\img8.png
Predicted Class: Abuse
Class Probabilities:
Abuse: 0.4713
Explosion: 0.2298
Fighting: 0.1072
Normal: 0.1917
Alert email sent successfully.
```