```
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
import os
import cv2
import matplotlib.pyplot as plt
```

```
2023-02-21 01:00:18.880633: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use t
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-02-21 01:00:19.206157: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.
2023-02-21 01:00:19.206219: I tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2023-02-21 01:00:20.949393: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: ca
2023-02-21 01:00:20.950070: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer_plugin.so.7'; dlerror: libnvinfer_
2023-02-21 01:00:20.950104: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot dlopen some TensorRT libraries. If you would like to use Nvidia GPU wi
```

```
#Function to extract labels for both real and altered images
def extract_label(img_path,train = True):
  filename, _ = os.path.splitext(os.path.basename(img_path))

  subject_id, etc = filename.split('__')
  #For Altered folder
  if train:
      gender, lr, finger, _, _ = etc.split('_')
  #For Real folder
  else:
      gender, lr, finger, _ = etc.split('_')

  gender = 0 if gender == 'M' else 1
  lr = 0 if lr == 'Left' else 1

  if finger == 'thumb':
      finger = 0
  elif finger == 'index':
      finger = 1
  elif finger == 'middle':
      finger = 2
  elif finger == 'ring':
      finger = 3
  elif finger == 'little':
      finger = 4
  return np.array([gender], dtype=np.uint16)


img_size = 96
#Function to iterate through all the images
def loading_data(path,train):
    print("loading data from: ",path)
    data = []
    for img in os.listdir(path):
        try:
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
            img_resize = cv2.resize(img_array, (img_size, img_size))
            label = extract_label(os.path.join(path, img),train)
            data.append([label[0], img_resize ])
        except Exception as e:
            pass
    data
    return data



Real_path = "archive/socofing/SOCOFing/Real"
Easy_path = "archive/socofing/SOCOFing/Altered/Altered-Easy"
Medium_path = "archive/socofing/SOCOFing/Altered/Altered-Medium"
Hard_path = "archive/socofing/SOCOFing/Altered/Altered-Hard"


Easy_data = loading_data(Easy_path, train = True)
Medium_data = loading_data(Medium_path, train = True)
Hard_data = loading_data(Hard_path, train = True)
test = loading_data(Real_path, train = False)

data = np.concatenate([Easy_data, Medium_data, Hard_data], axis=0)
```

```
del Easy_data, Medium_data, Hard_data
```

```
loading data from:  archive/socofing/SOCOFing/Altered/Altered-Easy
loading data from:  archive/socofing/SOCOFing/Altered/Altered-Medium
loading data from:  archive/socofing/SOCOFing/Altered/Altered-Hard
loading data from:  archive/socofing/SOCOFing/Real
<__array_function__ internals>:180: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different
```

```
import random
random.shuffle(data)
random.shuffle(test)


img, labels = [], []
for label, feature in data:
  labels.append(label)
  img.append(feature)
train_data = np.array(img).reshape(-1, img_size, img_size, 1)
train_data = train_data / 255.0
from keras.utils.np_utils import to_categorical
train_labels = to_categorical(labels, num_classes = 2)
del data


#Import necessary libraries
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
from tensorflow.keras import layers
from tensorflow.keras import optimizers

model = Sequential([
        Conv2D(32, 3, padding='same', activation='relu',kernel_initializer='he_uniform', input_shape = [96, 96, 1]),
        MaxPooling2D(2),
        Conv2D(32, 3, padding='same', kernel_initializer='he_uniform', activation='relu'),
        MaxPooling2D(2),
        Flatten(),
        Dense(128, kernel_initializer='he_uniform',activation = 'relu'),
        Dense(2, activation = 'softmax'),
        ])
```

```
2023-02-21 01:41:38.610499: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot
2023-02-21 01:41:38.610886: W tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:265] failed call to cuInit: UNKNOWN ERROR (303)
2023-02-21 01:41:38.611004: I tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (lapputoppu): /proc/driv
2023-02-21 01:41:38.784035: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use t
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

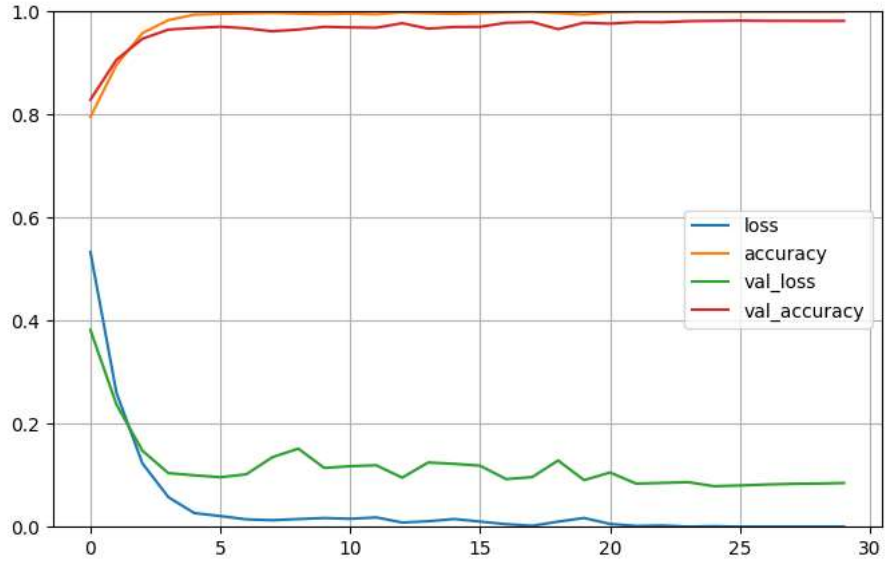
```
model.compile(optimizer = optimizers.Adam(1e-3), loss = 'categorical_crossentropy', metrics = ['accuracy'])
early_stopping_cb = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)


history = model.fit(train_data, train_labels, batch_size = 128, epochs = 30,
        validation_split = 0.2, callbacks = [early_stopping_cb], verbose = 1)
```

```
Epoch 2/30
308/308 [==============================] - 266s 862ms/step - loss: 0.2604 - accuracy: 0.8961 - val_loss: 0.2375 - val_accuracy: 0.9062
Epoch 3/30
308/308 [==============================] - 221s 717ms/step - loss: 0.1232 - accuracy: 0.9579 - val_loss: 0.1477 - val_accuracy: 0.9469
Epoch 4/30
308/308 [==============================] - 211s 686ms/step - loss: 0.0576 - accuracy: 0.9833 - val_loss: 0.1042 - val_accuracy: 0.9648
Epoch 5/30
308/308 [==============================] - 212s 688ms/step - loss: 0.0267 - accuracy: 0.9939 - val_loss: 0.1000 - val_accuracy: 0.9678
Epoch 6/30
308/308 [==============================] - 209s 678ms/step - loss: 0.0209 - accuracy: 0.9954 - val_loss: 0.0963 - val_accuracy: 0.9704
Epoch 7/30
308/308 [==============================] - 212s 688ms/step - loss: 0.0146 - accuracy: 0.9964 - val_loss: 0.1020 - val_accuracy: 0.9673
Epoch 8/30
308/308 [==============================] - 209s 680ms/step - loss: 0.0129 - accuracy: 0.9972 - val_loss: 0.1351 - val_accuracy: 0.9614
Epoch 9/30
308/308 [==============================] - 210s 682ms/step - loss: 0.0151 - accuracy: 0.9958 - val_loss: 0.1518 - val_accuracy: 0.9649
Epoch 10/30
308/308 [==============================] - 215s 697ms/step - loss: 0.0172 - accuracy: 0.9948 - val_loss: 0.1143 - val_accuracy: 0.9702
Epoch 11/30
```

```
Epoch 15/30
308/308 [==============================] - 260s 843ms/step - loss: 0.0152 - accuracy: 0.9952 - val_loss: 0.1223 - val_accuracy: 0.9699
Epoch 16/30
308/308 [==============================] - 228s 740ms/step - loss: 0.0102 - accuracy: 0.9967 - val_loss: 0.1186 - val_accuracy: 0.9702
Epoch 17/30
308/308 [==============================] - 209s 680ms/step - loss: 0.0052 - accuracy: 0.9985 - val_loss: 0.0926 - val_accuracy: 0.9781
Epoch 18/30
308/308 [==============================] - 211s 684ms/step - loss: 0.0020 - accuracy: 0.9996 - val_loss: 0.0965 - val_accuracy: 0.9795
Epoch 19/30
308/308 [==============================] - 212s 687ms/step - loss: 0.0101 - accuracy: 0.9968 - val_loss: 0.1288 - val_accuracy: 0.9655
Epoch 20/30
308/308 [==============================] - 208s 676ms/step - loss: 0.0171 - accuracy: 0.9939 - val_loss: 0.0906 - val_accuracy: 0.9782
Epoch 21/30
308/308 [==============================] - 213s 692ms/step - loss: 0.0057 - accuracy: 0.9985 - val_loss: 0.1055 - val_accuracy: 0.9764
Epoch 22/30
308/308 [==============================] - 212s 689ms/step - loss: 0.0017 - accuracy: 0.9996 - val_loss: 0.0839 - val_accuracy: 0.9794
Epoch 23/30
308/308 [==============================] - 210s 682ms/step - loss: 0.0026 - accuracy: 0.9997 - val_loss: 0.0853 - val_accuracy: 0.9790
Epoch 24/30
308/308 [==============================] - 211s 685ms/step - loss: 5.6878e-04 - accuracy: 0.9998 - val_loss: 0.0867 - val_accuracy: 0.9811
Epoch 25/30
308/308 [==============================] - 206s 670ms/step - loss: 0.0012 - accuracy: 0.9996 - val_loss: 0.0789 - val_accuracy: 0.9817
Epoch 26/30
308/308 [==============================] - 215s 697ms/step - loss: 4.2742e-04 - accuracy: 0.9996 - val_loss: 0.0803 - val_accuracy: 0.9821
Epoch 27/30
308/308 [==============================] - 214s 694ms/step - loss: 3.9160e-04 - accuracy: 0.9997 - val_loss: 0.0821 - val_accuracy: 0.9817
Epoch 28/30
308/308 [==============================] - 263s 855ms/step - loss: 3.6940e-04 - accuracy: 0.9998 - val_loss: 0.0835 - val_accuracy: 0.9817
Epoch 29/30
308/308 [==============================] - 383s 1s/step - loss: 3.7563e-04 - accuracy: 0.9998 - val_loss: 0.0840 - val_accuracy: 0.9815
Epoch 30/30
308/308 [==============================] - 389s 1s/step - loss: 3.4882e-04 - accuracy: 0.9998 - val_loss: 0.0851 - val_accuracy: 0.9816
```

```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize = (8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
```

```
(0.0, 1.0)
```




```
test_images, test_labels = [], []

for label, feature in test:
    test_images.append(feature)
    test_labels.append(label)

test_images = np.array(test_images).reshape(-1, img_size, img_size, 1)
test_images = test_images / 255.0
del test
test_labels  = to_categorical(test_labels, num_classes = 2)


model.evaluate(test_images, test_labels)
```

```
188/188 [==============================] - 12s 60ms/step - loss: 0.0539 - accuracy: 0.9938
[0.05391285568475723, 0.9938333630561829]
```