In [2]:

```
!pip install numpy pandas tqdm scikit-learn tensorflow pyaudio librosa
```

```
Requirement already satisfied: numpy in c:\users\dell\anaconda3\lib\site-packages
(1.23.5)
Requirement already satisfied: pandas in c:\users\dell\anaconda3\lib\site-packages
(1.5.3)
Requirement already satisfied: tqdm in c:\users\dell\anaconda3\lib\site-packages
(4.64.1)
Requirement already satisfied: scikit-learn in c:\users\dell\anaconda3\lib\site-pa
ckages (1.2.1)
Requirement already satisfied: tensorflow in c:\users\dell\anaconda3\lib\site-pack
ages (2.15.0)
Requirement already satisfied: pyaudio in c:\users\dell\anaconda3\lib\site-package
s (0.2.14)
Requirement already satisfied: librosa in c:\users\dell\anaconda3\lib\site-package
s (0.10.1)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\lib\site-pa
ckages (from pandas) (2022.7)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\dell\anaconda3\l
ib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: colorama in c:\users\dell\anaconda3\lib\site-packag
es (from tqdm) (0.4.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\dell\anaconda3\lib
\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\dell\anaconda3\lib\site-p
ackages (from scikit-learn) (1.1.1)
Requirement already satisfied: scipy>=1.3.2 in c:\users\dell\anaconda3\lib\site-pa
ckages (from scikit-learn) (1.10.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\dell\anaconda3
\lib\site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\dell\anaconda3\lib\site-
packages (from tensorflow-intel==2.15.0->tensorflow) (2.0.0)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\del
l\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.5.4)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\dell\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.60.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.
4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\dell\anaconda3\lib\site-packages (from t
ensorflow-intel==2.15.0->tensorflow) (4.23.4)
Requirement already satisfied: packaging in c:\users\dell\anaconda3\lib\site-packa
ges (from tensorflow-intel==2.15.0->tensorflow) (22.0)
Requirement already satisfied: setuptools in c:\users\dell\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.15.0->tensorflow) (65.6.3)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in c:\users\dell
\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\dell\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.15.0->tensorflow) (16.0.6)
Requirement already satisfied: tensorboard<2.16,>=2.15 in c:\users\dell\anaconda3
\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.1)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\dell\anaconda3
\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (4.4.0)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\dell\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (23.5.26)
Requirement already satisfied: keras<2.16,>=2.15.0 in c:\users\dell\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\de
ll\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.31.
0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\dell\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\dell\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.15.0->tensorflow) (2.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\dell\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (3.3.0)
Requirement already satisfied: six>=1.12.0 in c:\users\dell\anaconda3\lib\site-pac
kages (from tensorflow-intel==2.15.0->tensorflow) (1.16.0)
```

```
Requirement already satisfied: astunparse>=1.6.0 in c:\users\dell\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (1.6.3)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\dell\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.14.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in c:\users\dell\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\dell\anaconda3\lib\site-pac
kages (from tensorflow-intel==2.15.0->tensorflow) (3.7.0)
Requirement already satisfied: soxr>=0.3.2 in c:\users\dell\anaconda3\lib\site-pac
kages (from librosa) (0.3.7)
Requirement already satisfied: decorator>=4.3.0 in c:\users\dell\anaconda3\lib\sit
e-packages (from librosa) (5.1.1)
Requirement already satisfied: pooch>=1.0 in c:\users\dell\anaconda3\lib\site-pack
ages (from librosa) (1.4.0)
Requirement already satisfied: audioread>=2.1.9 in c:\users\dell\anaconda3\lib\sit
e-packages (from librosa) (3.0.1)
Requirement already satisfied: msgpack>=1.0 in c:\users\dell\anaconda3\lib\site-pa
ckages (from librosa) (1.0.3)
Requirement already satisfied: soundfile>=0.12.1 in c:\users\dell\anaconda3\lib\si
te-packages (from librosa) (0.12.1)
Requirement already satisfied: lazy-loader>=0.1 in c:\users\dell\anaconda3\lib\sit
e-packages (from librosa) (0.3)
Requirement already satisfied: numba>=0.51.0 in c:\users\dell\anaconda3\lib\site-p
ackages (from librosa) (0.56.4)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in c:\users\dell\anacond
a3\lib\site-packages (from numba>=0.51.0->librosa) (0.39.1)
Requirement already satisfied: requests in c:\users\dell\anaconda3\lib\site-packag
es (from pooch>=1.0->librosa) (2.28.1)
Requirement already satisfied: appdirs in c:\users\dell\anaconda3\lib\site-package
s (from pooch>=1.0->librosa) (1.4.4)
Requirement already satisfied: cffi>=1.0 in c:\users\dell\anaconda3\lib\site-packa
ges (from soundfile>=0.12.1->librosa) (1.15.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\dell\anaconda3\lib\s
ite-packages (from astunparse>=1.6.0->tensorflow-intel==2.15.0->tensorflow) (0.38.
4)
Requirement already satisfied: pycparser in c:\users\dell\anaconda3\lib\site-packa
ges (from cffi>=1.0->soundfile>=0.12.1->librosa) (2.21)
Requirement already satisfied: markdown>=2.6.8 in c:\users\dell\anaconda3\lib\site
-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.
4.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\dell\anaconda3\lib\site
-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.
2.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\dell\anaconda3\li
b\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflo
w) (2.25.2)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\d
ell\anaconda3\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==
2.15.0->tensorflow) (0.7.2)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in c:\users\dell\anaco
nda3\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->te
nsorflow) (1.1.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dell\anaconda3\lib\site-pa
ckages (from requests->pooch>=1.0->librosa) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\dell\anaconda3\li
b\site-packages (from requests->pooch>=1.0->librosa) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dell\anaconda3\lib\s
ite-packages (from requests->pooch>=1.0->librosa) (2023.7.22)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\dell\anaconda3
\lib\site-packages (from requests->pooch>=1.0->librosa) (2.0.4)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\dell\anaconda3\lib\site-p
ackages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.
15.0->tensorflow) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\dell\anaconda3\li
```

```
b\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-i
ntel==2.15.0->tensorflow) (0.2.8)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\dell\anaconda3\l
ib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-
intel==2.15.0->tensorflow) (5.3.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\dell\anaconda3
\lib\site-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->te
nsorflow-intel==2.15.0->tensorflow) (1.3.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\dell\anaconda3\lib\si
te-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.1
5.0->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\dell\anaconda3\lib
\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.1
6,>=2.15->tensorflow-intel==2.15.0->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\dell\anaconda3\lib\site
-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboar
d<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.2.2)
```

In [3]:

```python
import pandas as pd
import numpy as np
import os
import tqdm
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from sklearn.model_selection import train_test_split


label2int = {
    "male": 1,
    "female": 0
}


def load_data(vector_length=128):
    """A function to load gender recognition dataset from `data` folder
    After the second run, this will load from results/features.npy and results/labe
    as it is much faster!"""
    # make sure results folder exists
    if not os.path.isdir("results"):
        os.mkdir("results")
    # if features & labels already loaded individually and bundled, load them from
    if os.path.isfile("results/features.npy") and os.path.isfile("results/labels.np
        X = np.load("results/features.npy")
        y = np.load("results/labels.npy")
        return X, y
    # read dataframe
    df = pd.read_csv("balanced-all.csv")
    # get total samples
    n_samples = len(df)
    # get total male samples
    n_male_samples = len(df[df['gender'] == 'male'])
    # get total female samples
    n_female_samples = len(df[df['gender'] == 'female'])
    print("Total samples:", n_samples)
    print("Total male samples:", n_male_samples)
    print("Total female samples:", n_female_samples)
    # initialize an empty array for all audio features
    X = np.zeros((n_samples, vector_length))
    # initialize an empty array for all audio labels (1 for male and 0 for female)
    y = np.zeros((n_samples, 1))
    for i, (filename, gender) in tqdm.tqdm(enumerate(zip(df['filename'], df['gender
        features = np.load(filename)
        X[i] = features
        y[i] = label2int[gender]
    # save the audio features and labels into files
    # so we won't load each one of them next run
    np.save("results/features", X)
    np.save("results/labels", y)
    return X, y


def split_data(X, y, test_size=0.1, valid_size=0.1):
    # split training set and testing set
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size,
    # split training set and validation set
    X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_si
    # return a dictionary of values
    return {
        "X_train": X_train,
        "X_valid": X_valid,
        "X_test": X_test,
        "y_train": y_train,
        "y_valid": y_valid,
        "y_test": y_test
    }


def create_model(vector_length=128):
    """5 hidden dense layers from 256 units to 64, not the best model, but not bad.
    model = Sequential()
    model.add(Dense(256, input_shape=(vector_length,)))
    model.add(Dropout(0.3))
    model.add(Dense(256, activation="relu"))
    model.add(Dropout(0.3))
    model.add(Dense(128, activation="relu"))
    model.add(Dropout(0.3))
    model.add(Dense(128, activation="relu"))
    model.add(Dropout(0.3))
    model.add(Dense(64, activation="relu"))
    model.add(Dropout(0.3))
    # one output neuron with sigmoid activation function, 0 means female, 1 means m
    model.add(Dense(1, activation="sigmoid"))
    # using binary crossentropy as it's male/female classification (binary)
    model.compile(loss="binary_crossentropy", metrics=["accuracy"], optimizer="adam
    # print summary of the model
    model.summary()
    return model
```

```
WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\losse
s.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please u
se tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [4]:

```
import os
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard, EarlyStopping

from utils import load_data, split_data, create_model

# load the dataset
X, y = load_data()
# split the data into training, validation and testing sets
data = split_data(X, y, test_size=0.1, valid_size=0.1)
# construct the model
model = create_model()

# use tensorboard to view metrics
tensorboard = TensorBoard(log_dir="logs")
# define early stopping to stop training after 5 epochs of not improving
```

```
early_stopping = EarlyStopping(mode="min", patience=5, restore_best_weights=True)

batch_size = 64
epochs = 100

# train the model using the training set and validating using validation set
model.fit(data["X_train"], data["y_train"], epochs=epochs, batch_size=batch_size, v
          callbacks=[tensorboard, early_stopping])

# save the model to a file
model.save("results/model.h5")

# evaluating the model using the testing set
print(f"Evaluating the model using {len(data['X_test'])} samples...")
loss, accuracy = model.evaluate(data["X_test"], data["y_test"], verbose=0)
print(f"Loss: {loss:.4f}")
print(f"Accuracy: {accuracy*100:.2f}%")
```

```
WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\backen
d.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get
_default_graph instead.

WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\optimi
zers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.com
pat.v1.train.Optimizer instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 256)               33024     
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 dense_1 (Dense)             (None, 256)               65792     
                                                                 
 dropout_1 (Dropout)         (None, 256)               0         
                                                                 
 dense_2 (Dense)             (None, 128)               32896     
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
                                                                 
 dense_3 (Dense)             (None, 128)               16512     
                                                                 
 dropout_3 (Dropout)         (None, 128)               0         
                                                                 
 dense_4 (Dense)             (None, 64)                8256      
                                                                 
 dropout_4 (Dropout)         (None, 64)                0         
                                                                 
 dense_5 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 156545 (611.50 KB)
Trainable params: 156545 (611.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\utils
\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use t
f.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\engine
\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is depre
cated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

848/848 [==============================] - 10s 7ms/step - loss: 0.5754 - accuracy:
0.7635 - val_loss: 0.3938 - val_accuracy: 0.8437
Epoch 2/100
848/848 [==============================] - 8s 9ms/step - loss: 0.4217 - accuracy:
0.8318 - val_loss: 0.3592 - val_accuracy: 0.8622
Epoch 3/100
848/848 [==============================] - 6s 7ms/step - loss: 0.3800 - accuracy:
0.8503 - val_loss: 0.3142 - val_accuracy: 0.8757
Epoch 4/100
848/848 [==============================] - 9s 11ms/step - loss: 0.3616 - accuracy:
0.8606 - val_loss: 0.3017 - val_accuracy: 0.8800
Epoch 5/100
848/848 [==============================] - 9s 11ms/step - loss: 0.3488 - accuracy:
0.8664 - val_loss: 0.3141 - val_accuracy: 0.8787
Epoch 6/100
848/848 [==============================] - 8s 9ms/step - loss: 0.3408 - accuracy:
```

```
0.8691 - val_loss: 0.2885 - val_accuracy: 0.8817
Epoch 7/100
848/848 [==============================] - 6s 8ms/step - loss: 0.3287 - accuracy:
0.8747 - val_loss: 0.2777 - val_accuracy: 0.8951
Epoch 8/100
848/848 [==============================] - 5s 6ms/step - loss: 0.3204 - accuracy:
0.8797 - val_loss: 0.2829 - val_accuracy: 0.8923
Epoch 9/100
848/848 [==============================] - 5s 6ms/step - loss: 0.3218 - accuracy:
0.8769 - val_loss: 0.2653 - val_accuracy: 0.8959
Epoch 10/100
848/848 [==============================] - 6s 7ms/step - loss: 0.3157 - accuracy:
0.8791 - val_loss: 0.2605 - val_accuracy: 0.8976
Epoch 11/100
848/848 [==============================] - 5s 6ms/step - loss: 0.3112 - accuracy:
0.8837 - val_loss: 0.2613 - val_accuracy: 0.9019
Epoch 12/100
848/848 [==============================] - 7s 8ms/step - loss: 0.2967 - accuracy:
0.8886 - val_loss: 0.2517 - val_accuracy: 0.9036
Epoch 13/100
848/848 [==============================] - 6s 7ms/step - loss: 0.2955 - accuracy:
0.8876 - val_loss: 0.2522 - val_accuracy: 0.9029
Epoch 14/100
848/848 [==============================] - 7s 9ms/step - loss: 0.2920 - accuracy:
0.8896 - val_loss: 0.2663 - val_accuracy: 0.9026
Epoch 15/100
848/848 [==============================] - 8s 9ms/step - loss: 0.2871 - accuracy:
0.8918 - val_loss: 0.2597 - val_accuracy: 0.9006
Epoch 16/100
848/848 [==============================] - 6s 7ms/step - loss: 0.2894 - accuracy:
0.8907 - val_loss: 0.2514 - val_accuracy: 0.9077
Epoch 17/100
848/848 [==============================] - 10s 12ms/step - loss: 0.2822 - accurac
y: 0.8941 - val_loss: 0.2457 - val_accuracy: 0.9082
Epoch 18/100
848/848 [==============================] - 19s 22ms/step - loss: 0.2801 - accurac
y: 0.8946 - val_loss: 0.2475 - val_accuracy: 0.9071
Epoch 19/100
848/848 [==============================] - 14s 17ms/step - loss: 0.2774 - accurac
y: 0.8968 - val_loss: 0.2387 - val_accuracy: 0.9066
Epoch 20/100
848/848 [==============================] - 9s 10ms/step - loss: 0.2767 - accuracy:
0.8971 - val_loss: 0.2467 - val_accuracy: 0.9097
Epoch 21/100
848/848 [==============================] - 6s 7ms/step - loss: 0.2721 - accuracy:
0.8983 - val_loss: 0.2418 - val_accuracy: 0.9102
Epoch 22/100
848/848 [==============================] - 7s 9ms/step - loss: 0.2771 - accuracy:
0.8976 - val_loss: 0.2498 - val_accuracy: 0.9054
Epoch 23/100
848/848 [==============================] - 6s 7ms/step - loss: 0.2780 - accuracy:
0.8963 - val_loss: 0.2384 - val_accuracy: 0.9119
Epoch 24/100
848/848 [==============================] - 6s 7ms/step - loss: 0.2731 - accuracy:
0.8977 - val_loss: 0.2394 - val_accuracy: 0.9057
Epoch 25/100
848/848 [==============================] - 8s 9ms/step - loss: 0.2705 - accuracy:
0.8990 - val_loss: 0.2325 - val_accuracy: 0.9163
Epoch 26/100
848/848 [==============================] - 6s 8ms/step - loss: 0.2717 - accuracy:
0.8987 - val_loss: 0.2366 - val_accuracy: 0.9124
Epoch 27/100
848/848 [==============================] - 7s 8ms/step - loss: 0.2665 - accuracy:
0.8992 - val_loss: 0.2271 - val_accuracy: 0.9097
```

```
Epoch 28/100
848/848 [==============================] - 7s 8ms/step - loss: 0.2658 - accuracy:
0.9011 - val_loss: 0.2313 - val_accuracy: 0.9132
Epoch 29/100
848/848 [==============================] - 7s 8ms/step - loss: 0.2650 - accuracy:
0.9021 - val_loss: 0.2293 - val_accuracy: 0.9147
Epoch 30/100
848/848 [==============================] - 7s 8ms/step - loss: 0.2658 - accuracy:
0.9005 - val_loss: 0.2321 - val_accuracy: 0.9144
Epoch 31/100
848/848 [==============================] - 7s 8ms/step - loss: 0.2608 - accuracy:
0.9017 - val_loss: 0.2260 - val_accuracy: 0.9182
Epoch 32/100
848/848 [==============================] - 8s 9ms/step - loss: 0.2645 - accuracy:
0.9021 - val_loss: 0.2311 - val_accuracy: 0.9132
Epoch 33/100
848/848 [==============================] - 7s 8ms/step - loss: 0.2610 - accuracy:
0.9028 - val_loss: 0.2348 - val_accuracy: 0.9115
Epoch 34/100
848/848 [==============================] - 10s 12ms/step - loss: 0.2608 - accurac
y: 0.9038 - val_loss: 0.2371 - val_accuracy: 0.9130
Epoch 35/100
848/848 [==============================] - 8s 10ms/step - loss: 0.2577 - accuracy:
0.9038 - val_loss: 0.2435 - val_accuracy: 0.9105
Epoch 36/100
848/848 [==============================] - 8s 9ms/step - loss: 0.2618 - accuracy:
0.9012 - val_loss: 0.2287 - val_accuracy: 0.9170
Evaluating the model using 6694 samples...
```

```
C:\Users\Dell\anaconda3\lib\site-packages\keras\src\engine\training.py:3103: UserW
arning: You are saving your model as an HDF5 file via `model.save()`. This file fo
rmat is considered legacy. We recommend using instead the native Keras format, e.
g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
Loss: 0.2208
Accuracy: 91.75%
```

In [20]:

```python
import pyaudio
import os
import wave
import librosa
import numpy as np
from sys import byteorder
from array import array
from struct import pack


THRESHOLD = 500
CHUNK_SIZE = 1024
FORMAT = pyaudio.paInt16
RATE = 16000

SILENCE = 30

def is_silent(snd_data):
    "Returns 'True' if below the 'silent' threshold"
    return max(snd_data) < THRESHOLD

def normalize(snd_data):
    "Average the volume out"
    MAXIMUM = 16384
    times = float(MAXIMUM)/max(abs(i) for i in snd_data)

    r = array('h')
    for i in snd_data:
        r.append(int(i*times))
    return r

def trim(snd_data):
    "Trim the blank spots at the start and end"
    def _trim(snd_data):
        snd_started = False
        r = array('h')

        for i in snd_data:
            if not snd_started and abs(i)>THRESHOLD:
                snd_started = True
                r.append(i)

            elif snd_started:
                r.append(i)
        return r

    # Trim to the left
    snd_data = _trim(snd_data)

    # Trim to the right
    snd_data.reverse()
    snd_data = _trim(snd_data)
    snd_data.reverse()
    return snd_data

def add_silence(snd_data, seconds):
    "Add silence to the start and end of 'snd_data' of length 'seconds' (float)"
    r = array('h', [0 for i in range(int(seconds*RATE))])
    r.extend(snd_data)
    r.extend([0 for i in range(int(seconds*RATE))])
    return r

def record():
    """
    Record a word or words from the microphone and
    return the data as an array of signed shorts.
    Normalizes the audio, trims silence from the
    start and end, and pads with 0.5 seconds of
    blank sound to make sure VLC et al can play
    it without getting chopped off.
    """
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT, channels=1, rate=RATE,
        input=True, output=True,
        frames_per_buffer=CHUNK_SIZE)

    num_silent = 0
    snd_started = False

    r = array('h')

    while 1:
        # little endian, signed short
        snd_data = array('h', stream.read(CHUNK_SIZE))
        if byteorder == 'big':
            snd_data.byteswap()
        r.extend(snd_data)

        silent = is_silent(snd_data)

        if silent and snd_started:
            num_silent += 1
        elif not silent and not snd_started:
            snd_started = True

        if snd_started and num_silent > SILENCE:
            break

    sample_width = p.get_sample_size(FORMAT)
    stream.stop_stream()
    stream.close()
    p.terminate()

    r = normalize(r)
    r = trim(r)
    r = add_silence(r, 0.5)
    return sample_width, r

def record_to_file(path):
    "Records from the microphone and outputs the resulting data to 'path'"
    sample_width, data = record()
    data = pack('<' + ('h'*len(data)), *data)

    wf = wave.open(path, 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(sample_width)
    wf.setframerate(RATE)
    wf.writeframes(data)
    wf.close()



def extract_feature(file_name, **kwargs):
    """
    Extract feature from audio file `file_name`
        Features supported:
            - MFCC (mfcc)
            - Chroma (chroma)
            - MEL Spectrogram Frequency (mel)
            - Contrast (contrast)
            - Tonnetz (tonnetz)
        e.g:
        `features = extract_feature(path, mel=True, mfcc=True)`
    """
    mfcc = kwargs.get("mfcc")
    chroma = kwargs.get("chroma")
    mel = kwargs.get("mel")
    contrast = kwargs.get("contrast")
    tonnetz = kwargs.get("tonnetz")
    file_path = "C:/Users/Dell/Downloads/gender-recognition-by-voice-master (1)/ger
    X,  sample_rate = librosa.core.load(file_path)

    if chroma or contrast:
        stft = np.abs(librosa.stft(X))
    result = np.array([])
    if mfcc:
        mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axi
        result = np.hstack((result, mfccs))
    if chroma:
        chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis
        result = np.hstack((result, chroma))
    if mel:
        mel = np.mean(librosa.feature.melspectrogram(y=X, sr=sample_rate).T, axis=
        result = np.hstack((result, mel))
    if contrast:
        contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate
        result = np.hstack((result, contrast))
    if tonnetz:
        tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X), sr
        result = np.hstack((result, tonnetz))
    return result


if __name__ == "__main__":
    # load the saved model (after training)
    # model = pickle.load(open("result/mlp_classifier.model", "rb"))
    from utils import load_data, split_data, create_model
    import argparse
    parser = argparse.ArgumentParser(description="""Gender recognition script, this
                                    and perform inference on a sample you provide (
    parser.add_argument("-f", "--file", help="The path to the file, preferred to be
    args = parser.parse_args()
    file = args.file
    # construct the model
    model = create_model()
    # load the saved/trained weights
    model.load_weights("results/model.h5")
    if not file or not os.path.isfile(file):
        # if file not provided, or it doesn't exist, use your voice
        print("Please talk")
        # put the file name here
        file = r"C:\path\to\save\test.wav"
        # record the file (start talking)
        record_to_file(file)
    # extract features and reshape it
    features = extract_feature(file, mel=True).reshape(1, -1)
    # predict the gender!
    male_prob = model.predict(features)[0][0]
    female_prob = 1 - male_prob
    gender = "male" if male_prob > female_prob else "female"
    # show the result!
    print("Result:", gender)
    print(f"Probabilities:     Male: {male_prob*100:.2f}%    Female: {female_prob*1
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_60 (Dense)            (None, 256)               33024     
                                                                 
 dropout_50 (Dropout)        (None, 256)               0         
                                                                 
 dense_61 (Dense)            (None, 256)               65792     
                                                                 
 dropout_51 (Dropout)        (None, 256)               0         
                                                                 
 dense_62 (Dense)            (None, 128)               32896     
                                                                 
 dropout_52 (Dropout)        (None, 128)               0         
                                                                 
 dense_63 (Dense)            (None, 128)               16512     
                                                                 
 dropout_53 (Dropout)        (None, 128)               0         
                                                                 
 dense_64 (Dense)            (None, 64)                8256      
                                                                 
 dropout_54 (Dropout)        (None, 64)                0         
                                                                 
 dense_65 (Dense)            (None, 1)                 65        
                                                                 
=================================================================
Total params: 156545 (611.50 KB)
Trainable params: 156545 (611.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
1/1 [==============================] - 0s 418ms/step
Result: male
Probabilities:     Male: 95.74%    Female: 4.26%
```

In [21]:

```python
import glob
import os
import pandas as pd
import numpy as np
import shutil
import librosa
from tqdm import tqdm


def extract_feature(file_name, **kwargs):
    """
    Extract feature from audio file `file_name`
        Features supported:
            - MFCC (mfcc)
            - Chroma (chroma)
            - MEL Spectrogram Frequency (mel)
            - Contrast (contrast)
            - Tonnetz (tonnetz)
        e.g:
        `features = extract_feature(path, mel=True, mfcc=True)`
    """
    mfcc = kwargs.get("mfcc")
    chroma = kwargs.get("chroma")
    mel = kwargs.get("mel")
    contrast = kwargs.get("contrast")
    tonnetz = kwargs.get("tonnetz")
    X, sample_rate = librosa.core.load(file_name)
    if chroma or contrast:
        stft = np.abs(librosa.stft(X))
    result = np.array([])
    if mfcc:
        mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axi
        result = np.hstack((result, mfccs))
    if chroma:
        chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis
        result = np.hstack((result, chroma))
    if mel:
        mel = np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result = np.hstack((result, mel))
    if contrast:
        contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate
        result = np.hstack((result, contrast))
    if tonnetz:
        tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X), sr
        result = np.hstack((result, tonnetz))
    return result

dirname = "data"

if not os.path.isdir(dirname):
    os.mkdir(dirname)


csv_files = glob.glob("*.csv")

for j, csv_file in enumerate(csv_files):
    print("[+] Preprocessing", csv_file)
    df = pd.read_csv(csv_file)
    # only take filename and gender columns
    new_df = df[["filename", "gender"]]
    print("Previously:", len(new_df), "rows")
    # take only male & female genders (i.e droping NaNs & 'other' gender)
    new_df = new_df[np.logical_or(new_df['gender'] == 'female', new_df['gender'] ==
    print("Now:", len(new_df), "rows")
    new_csv_file = os.path.join(dirname, csv_file)
    # save new preprocessed CSV
    new_df.to_csv(new_csv_file, index=False)
    # get the folder name
    folder_name, _ = csv_file.split(".")
    audio_files = glob.glob(f"{folder_name}/{folder_name}/*")
    all_audio_filenames = set(new_df["filename"])
    for i, audio_file in tqdm(list(enumerate(audio_files)), f"Extracting features o
        splited = os.path.split(audio_file)
        # audio_filename = os.path.join(os.path.split(splited[0])[-1], splited[-1])
        audio_filename = f"{os.path.split(splited[0])[-1]}/{splited[-1]}"
        # print("audio_filename:", audio_filename)
        if audio_filename in all_audio_filenames:
            # print("Copyying", audio_filename, "...")
            src_path = f"{folder_name}/{audio_filename}"
            target_path = f"{dirname}/{audio_filename}"
            #create that folder if it doesn't exist
            if not os.path.isdir(os.path.dirname(target_path)):
                os.mkdir(os.path.dirname(target_path))
            features = extract_feature(src_path, mel=True)
            target_filename = target_path.split(".")[0]
            np.save(target_filename, features)
            # shutil.copyfile(src_path, target_path)
```

```
[+] Preprocessing balanced-all.csv
Previously: 66938 rows
Now: 66938 rows
```

```
Extracting features of balanced-all: 0it [00:00, ?it/s]
```

In [24]:
```
!python test.py --file "C:/Users/Dell/Downloads/gender-recognition-by-voice-master
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 256)               33024     
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 dense_1 (Dense)             (None, 256)               65792     
                                                                 
 dropout_1 (Dropout)         (None, 256)               0         
                                                                 
 dense_2 (Dense)             (None, 128)               32896     
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
                                                                 
 dense_3 (Dense)             (None, 128)               16512     
                                                                 
 dropout_3 (Dropout)         (None, 128)               0         
                                                                 
 dense_4 (Dense)             (None, 64)                8256      
                                                                 
 dropout_4 (Dropout)         (None, 64)                0         
                                                                 
 dense_5 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 156545 (611.50 KB)
Trainable params: 156545 (611.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________

1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 238ms/step
Result: male
Probabilities:     Male: 96.85%    Female: 3.15%
```

```
2023-12-09 02:05:53.611507: I tensorflow/core/util/port.cc:113] oneDNN custom oper
ations are on. You may see slightly different numerical results due to floating-po
int round-off errors from different computation orders. To turn them off, set the
environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\losse
s.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please u
se tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\backen
d.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get
_default_graph instead.

2023-12-09 02:06:05.550575: I tensorflow/core/platform/cpu_feature_guard.cc:182] T
his TensorFlow binary is optimized to use available CPU instructions in performanc
e-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX2 FMA, in oth
er operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:tensorflow:From C:\Users\Dell\anaconda3\lib\site-packages\keras\src\optimi
zers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.com
pat.v1.train.Optimizer instead.
```

In [ ]: