In [2]:
```
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 tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard, EarlyStoppir
from sklearn.model_selection import train_test_split
```

In [3]:
```
df = pd.read_csv("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-voice-m
```

In [4]:
```
df.head()
```

Out[4]:

	filename	gender
0	data/cv-other-train/sample-069205.npy	female
1	data/cv-valid-train/sample-063134.npy	female
2	data/cv-other-train/sample-080873.npy	female
3	data/cv-other-train/sample-105595.npy	female
4	data/cv-valid-train/sample-144613.npy	female

In [5]:
```
df.tail()
```

Out[5]:

	filename	gender
66933	data/cv-valid-train/sample-171098.npy	male
66934	data/cv-other-train/sample-022864.npy	male
66935	data/cv-valid-train/sample-080933.npy	male
66936	data/cv-other-train/sample-012026.npy	male
66937	data/cv-other-train/sample-013841.npy	male

In [6]:
```
# 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)
```

```
Total samples: 66938
Total male samples: 33469
Total female samples: 33469
```

In [7]:

```
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/la
    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 fro
    if os.path.isfile("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-vo
        X = np.load("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-voic
        y = np.load("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-voic
        return X, y
    # read dataframe
    df = pd.read_csv("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-voi
    # 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['gend
        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("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-voice-master
    np.save("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-voice-master
    return X, y
```

In [8]:
```
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_
    # 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
    }
```

In [9]:
```
# 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)
```

In [10]:
```
def create_model(vector_length=128):
    """5 hidden dense layers from 256 units to 64, not the best model."""
    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
    model.add(Dense(1, activation="sigmoid"))
    # using binary crossentropy as it's male/female classification (binary)
    model.compile(loss="binary_crossentropy", metrics=["accuracy"], optimizer="ad
    # print summary of the model
    model.summary()
    return model
```

In [11]:
```
model = create_model()
```

```
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: 156,545
Trainable params: 156,545
Non-trainable params: 0
_________________________________________________________________
```

In [12]:
```
# 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,
          callbacks=[tensorboard, early_stopping])
```

```
Epoch 1/100
848/848 [==============================] - 4s 3ms/step - loss: 0.5519 - accurac
y: 0.7716 - val_loss: 0.3795 - val_accuracy: 0.8446
Epoch 2/100
848/848 [==============================] - 3s 3ms/step - loss: 0.4148 - accurac
y: 0.8354 - val_loss: 0.3216 - val_accuracy: 0.8742
Epoch 3/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3797 - accurac
y: 0.8527 - val_loss: 0.3400 - val_accuracy: 0.8719
Epoch 4/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3583 - accurac
y: 0.8615 - val_loss: 0.2931 - val_accuracy: 0.8856
Epoch 5/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3487 - accurac
y: 0.8675 - val_loss: 0.2967 - val_accuracy: 0.8883
Epoch 6/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3337 - accurac
y: 0.8711 - val_loss: 0.2879 - val_accuracy: 0.8954
Epoch 7/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3244 - accurac
y: 0.8770 - val_loss: 0.2798 - val_accuracy: 0.8913
Epoch 8/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3210 - accurac
y: 0.8778 - val_loss: 0.2663 - val_accuracy: 0.8978
Epoch 9/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3129 - accurac
y: 0.8811 - val_loss: 0.2870 - val_accuracy: 0.8903
Epoch 10/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3034 - accurac
y: 0.8845 - val_loss: 0.2859 - val_accuracy: 0.8888
Epoch 11/100
848/848 [==============================] - 2s 3ms/step - loss: 0.3072 - accurac
y: 0.8830 - val_loss: 0.2619 - val_accuracy: 0.8964
Epoch 12/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2991 - accurac
y: 0.8878 - val_loss: 0.2719 - val_accuracy: 0.8885
Epoch 13/100
848/848 [==============================] - 3s 3ms/step - loss: 0.2947 - accurac
y: 0.8896 - val_loss: 0.2646 - val_accuracy: 0.8961
Epoch 14/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2931 - accurac
y: 0.8892 - val_loss: 0.2477 - val_accuracy: 0.9064
Epoch 15/100
848/848 [==============================] - 3s 3ms/step - loss: 0.2924 - accurac
y: 0.8908 - val_loss: 0.2590 - val_accuracy: 0.8993
```

```
Epoch 16/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2897 - accurac
y: 0.8906 - val_loss: 0.2500 - val_accuracy: 0.9007
Epoch 17/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2848 - accurac
y: 0.8932 - val_loss: 0.2546 - val_accuracy: 0.9039
Epoch 18/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2825 - accurac
y: 0.8937 - val_loss: 0.2348 - val_accuracy: 0.9120
Epoch 19/100
848/848 [==============================] - 3s 3ms/step - loss: 0.2810 - accurac
y: 0.8939 - val_loss: 0.2438 - val_accuracy: 0.9069
Epoch 20/100
848/848 [==============================] - 3s 4ms/step - loss: 0.2870 - accurac
y: 0.8964 - val_loss: 0.2551 - val_accuracy: 0.9034
Epoch 21/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2799 - accurac
y: 0.8946 - val_loss: 0.2402 - val_accuracy: 0.9097
Epoch 22/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2731 - accurac
y: 0.8977 - val_loss: 0.2371 - val_accuracy: 0.9095
Epoch 23/100
848/848 [==============================] - 3s 3ms/step - loss: 0.2705 - accurac
y: 0.8999 - val_loss: 0.2316 - val_accuracy: 0.9120
Epoch 24/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2700 - accurac
y: 0.8990 - val_loss: 0.2410 - val_accuracy: 0.9089
Epoch 25/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2719 - accurac
y: 0.8988 - val_loss: 0.2352 - val_accuracy: 0.9124
Epoch 26/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2748 - accurac
y: 0.8983 - val_loss: 0.2365 - val_accuracy: 0.9102
Epoch 27/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2672 - accurac
y: 0.8992 - val_loss: 0.2423 - val_accuracy: 0.9072
Epoch 28/100
848/848 [==============================] - 2s 3ms/step - loss: 0.2658 - accurac
y: 0.9004 - val_loss: 0.2406 - val_accuracy: 0.9049
```

Out[12]:

```
<keras.callbacks.History at 0x2afaa905e50>
```

In [13]:

```python
model.save("C:/Users/ASUS/Desktop/IOT/project/gender-recognition-by-voice-master/
```

In [14]:

```python
# 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}%")
```

```
Evaluating the model using 6694 samples...
Loss: 0.2301
Accuracy: 91.46%
```

In [ ]:

```

```