Audio files - Deep learning

```
#audio files
!wget 'http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands.zip
#unzip
!unzip mini_speech_commands.zip
#delete unnecessary files
!rm '/content/mini_speech_commands/README.md'
```

```
  inflating: mini_speech_commands/yes/28ed6bc9_nohash_1.wav
  inflating: __MACOSX/mini_speech_commands/yes/._28ed6bc9_nohash_1.wav
  inflating: mini_speech_commands/yes/e805a617_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._e805a617_nohash_0.wav
  inflating: mini_speech_commands/yes/d197e3ae_nohash_3.wav
  inflating: __MACOSX/mini_speech_commands/yes/._d197e3ae_nohash_3.wav
  inflating: mini_speech_commands/yes/bd2db1a5_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._bd2db1a5_nohash_0.wav
  inflating: mini_speech_commands/yes/50f55535_nohash_1.wav
  inflating: __MACOSX/mini_speech_commands/yes/._50f55535_nohash_1.wav
  inflating: mini_speech_commands/yes/f550b7dc_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._f550b7dc_nohash_0.wav
  inflating: mini_speech_commands/yes/1aeef15e_nohash_1.wav
  inflating: __MACOSX/mini_speech_commands/yes/._1aeef15e_nohash_1.wav
  inflating: mini_speech_commands/yes/a0f93943_nohash_1.wav
  inflating: __MACOSX/mini_speech_commands/yes/._a0f93943_nohash_1.wav
  inflating: mini_speech_commands/yes/ab7b5acd_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._ab7b5acd_nohash_0.wav
  inflating: mini_speech_commands/yes/baeac2ba_nohash_3.wav
  inflating: __MACOSX/mini_speech_commands/yes/._baeac2ba_nohash_3.wav
  inflating: mini_speech_commands/yes/28ce0c58_nohash_3.wav
  inflating: __MACOSX/mini_speech_commands/yes/._28ce0c58_nohash_3.wav
  inflating: mini_speech_commands/yes/617de221_nohash_2.wav
  inflating: __MACOSX/mini_speech_commands/yes/._617de221_nohash_2.wav
  inflating: mini_speech_commands/yes/d0faf7e4_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._d0faf7e4_nohash_0.wav
  inflating: mini_speech_commands/yes/e649aa92_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._e649aa92_nohash_0.wav
  inflating: mini_speech_commands/yes/e7ea8b76_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._e7ea8b76_nohash_0.wav
  inflating: mini_speech_commands/yes/459345ea_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._459345ea_nohash_0.wav
  inflating: mini_speech_commands/yes/b97c9f77_nohash_3.wav
  inflating: __MACOSX/mini_speech_commands/yes/._b97c9f77_nohash_3.wav
  inflating: mini_speech_commands/yes/ec201020_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._ec201020_nohash_0.wav
  inflating: mini_speech_commands/yes/24c9f572_nohash_2.wav
  inflating: __MACOSX/mini_speech_commands/yes/._24c9f572_nohash_2.wav
  inflating: mini_speech_commands/yes/7d8babdb_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._7d8babdb_nohash_0.wav
  inflating: mini_speech_commands/yes/3006c271_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._3006c271_nohash_0.wav
  inflating: mini_speech_commands/yes/7799c9cd_nohash_0.wav
  inflating: __MACOSX/mini_speech_commands/yes/._7799c9cd_nohash_0.wav
  inflating: mini_speech_commands/yes/b7a0754f_nohash_1.wav
  inflating: __MACOSX/mini_speech_commands/yes/._b7a0754f_nohash_1.wav
  inflating: mini_speech_commands/yes/ad63d93c_nohash_0.wav
```

```
        inflating: mini_speech_commands/yes/ad63d93c_nohash_0.wav
        inflating: __MACOSX/mini_speech_commands/yes/._ad63d93c_nohash_0.wav
        inflating: mini_speech_commands/yes/c2aeb59d_nohash_0.wav
        inflating: __MACOSX/mini_speech_commands/yes/._c2aeb59d_nohash_0.wav
        inflating: mini_speech_commands/yes/7cbf645a_nohash_0.wav
        inflating: __MACOSX/mini_speech_commands/yes/._7cbf645a_nohash_0.wav
        inflating: mini_speech_commands/yes/30802c5d_nohash_0.wav
        inflating: __MACOSX/mini_speech_commands/yes/._30802c5d_nohash_0.wav
        inflating: mini_speech_commands/yes/da2c5f1b_nohash_2.wav
        inflating: __MACOSX/mini_speech_commands/yes/._da2c5f1b_nohash_2.wav
        inflating: mini_speech_commands/yes/c0c0d87d_nohash_0.wav
        inflating: __MACOSX/mini_speech_commands/yes/._c0c0d87d_nohash_0.wav
```

```
#pip install librosa
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import librosa
from scipy.io import wavfile
import IPython.display as ipd
```

```
ipd.Audio('/content/mini_speech_commands/down/004ae714_nohash_0.wav')
```

0:00 / 0:01

```
ipd.Audio('/content/mini_speech_commands/no/0132a06d_nohash_1.wav')
```

0:00 / 0:01

```
samples, samplingrate = librosa.load('/content/mini_speech_commands/down/004ae714_nohash_0
```

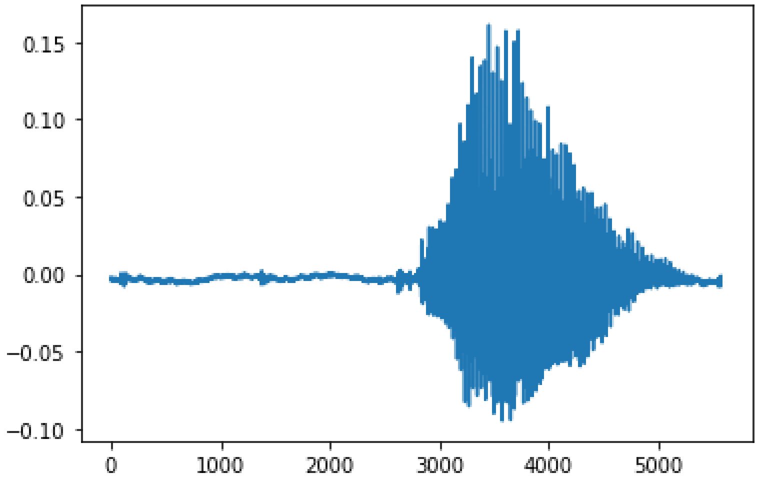
```
plt.plot(samples)
```

```
[<matplotlib.lines.Line2D at 0x7f1318021040>]
```

```
s,sr=librosa.load('/content/mini_speech_commands/go/004ae714_nohash_0.wav', sr=8000)
```

```
plt.plot(s)
```

```
[<matplotlib.lines.Line2D at 0x7f1318155fa0>]
```



read audio

```
#samplingrate=X intervals/sec
#samplingrate=8000
#sound length=1sec
#samples=8000

#sound length=2secs
#samples=16000
```

```
X = []
y = []
```

```
def collectAudio(foldername):
  basefolder='/content/mini_speech_commands/'
  folderpath=os.path.join(basefolder,foldername)
  for audio in os.listdir(folderpath):
    filepath = os.path.join(folderpath,audio)
    samples, _ = librosa.load(filepath, sr=8000)
    if(len(samples)==8000):
      X.append(samples)
      y.append(foldername)
```

```
collectAudio('down')
```

```
collectAudio('go')
collectAudio('left')
```

```
collectAudio('no')
```

```
collectAudio('right')
```

```
collectAudio('stop')
```

```
collectAudio('up')
```

```
collectAudio('yes')
```

Feature extraction

```
import librosa

def feature_chromagram(waveform, sample_rate):
    # STFT computed here explicitly; mel spectrogram and MFCC functions do this under the
    stft_spectrogram=np.abs(librosa.stft(waveform))
    # Produce the chromagram for all STFT frames and get the mean of each column of the re
    chromagram=np.mean(librosa.feature.chroma_stft(S=stft_spectrogram, sr=sample_rate).T,a
    return chromagram

def feature_melspectrogram(waveform, sample_rate):
    # Produce the mel spectrogram for all STFT frames and get the mean of each column of t
    # Using 8khz as upper frequency bound should be enough for most speech classification
    melspectrogram=np.mean(librosa.feature.melspectrogram(y=waveform, sr=sample_rate, n_me
    return melspectrogram

def feature_mfcc(waveform, sample_rate):
    # Compute the MFCCs for all STFT frames and get the mean of each column of the resulti
    # 40 filterbanks = 40 coefficients
    mfc_coefficients=np.mean(librosa.feature.mfcc(y=waveform, sr=sample_rate, n_mfcc=40).T
    return mfc_coefficients


def get_features(waveform):
    # load an individual soundfile
        chromagram = feature_chromagram(waveform, sample_rate=8000)
        melspectrogram = feature_melspectrogram(waveform, sample_rate=8000)
        mfc_coefficients = feature_mfcc(waveform, sample_rate=8000)

        feature_matrix=np.array([])
        # use np.hstack to stack our feature arrays horizontally to create a feature matri
        feature_matrix = np.hstack((chromagram, melspectrogram, mfc_coefficients))

        return feature_matrix
```

```
X_feats = []
```

```
for audio in X:
  features = get_features(audio)
  X_feats.append(features)
```

```
    /usr/local/lib/python3.8/dist-packages/librosa/filters.py:238: UserWarning: Empty fi
      warnings.warn(
    /usr/local/lib/python3.8/dist-packages/librosa/core/pitch.py:153: UserWarning: Tryin
      warnings.warn("Trying to estimate tuning from empty frequency set.")
```

```
len(X_feats)
```

```
    7804
```

```
len(y)
```

```
    7804
```

```
X[0].shape
```

```
    (8000,)
```

```
X_np = np.array(X_feats)
```

```
X_np.shape
```

```
    (7804, 180)
```

```
#audiocommands = ['down','go','left','no','right','stop','up','yes']
#for i in audiocommands:
#  collectAudio(i)
```

```
#audiocommands = os.listdir('mini_speech_commands')
#for i in audiocommands:
#  collectAudio(i)
```

```
len(X_feats)
```

```
    7804
```

```
X_np = np.array(X_feats)
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from tensorflow.keras.utils import to_categorical
```

```
enc = LabelEncoder()
enc.fit(y)
y_le = enc.transform(y)
y_one = to_categorical(y_le)
```

```
enc.classes_
```

```
     array(['down', 'go', 'left', 'no', 'right', 'stop', 'up', 'yes'],
           dtype='<U5')
```

```
X_np.shape
```

```
     (7804, 180)
```

```
y_one
```

```
     array([[1., 0., 0., ..., 0., 0., 0.],
            [1., 0., 0., ..., 0., 0., 0.],
            [1., 0., 0., ..., 0., 0., 0.],
            ...,
            [0., 0., 0., ..., 0., 0., 1.],
            [0., 0., 0., ..., 0., 0., 1.],
            [0., 0., 0., ..., 0., 0., 1.]], dtype=float32)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_np, y_one, test_size=0.2, shuffle=Tr
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, strati
```

```
sc.fit(X_train)
```

```
     StandardScaler()
```

```
sc.transform(X_train)
sc.transform(X_val)
sc.transform(X_test)
```

```
     array([[-2.9342518 , -2.8860927 , -2.965853  , ..., -1.235823  ,
             -1.5579255 , -1.0386353 ],
            [ 1.1621727 ,  1.7134429 ,  0.67530197, ..., -0.33271417,
              1.7156307 , -0.23227105],
            [-0.5614193 ,  0.5132362 ,  0.7873666 , ...,  0.4343924 ,
              0.1843935 ,  0.9201225 ],
            ...,
            [-0.11356499,  0.62283176, -0.09169139, ...,  1.1163362 ,
             -0.4500865 ,  1.2802444 ],
            [ 0.4219551 ,  0.17488007, -0.8309027 , ...,  1.0534976 ,
              1.2747871 ,  0.24491036],
            [-0.00848142, -0.20074311, -0.6544333 , ..., -0.10672616,
             -0.74048847, -1.3599845 ]], dtype=float32)
```

```
X_train.shape
```

```
(4682, 180)
```

```
X_val.shape
```

```
(1561, 180)
```

```
X_test.shape
```

```
(1561, 180)
```

ANN

```
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout
```

```
audioANN = Sequential()
```

```
audioANN.add(Dense(units=1024, activation='relu',input_dim=180))
audioANN.add(Dropout(rate=0.25))
audioANN.add(Dense(units=8, activation='softmax'))
```

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

```
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

es = EarlyStopping(monitor='val_accuracy', min_delta=0, patience=30, verbose=1, mode='autc
mc = ModelCheckpoint(filepath='bestweights.h5', monitor='val_accuracy', verbose=1, save_be
rd = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1, patience=15, verbose=1, mode='a
```

```
history = audioANN.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=100, callb
```

```
147/147 [==============================] - 1s 4ms/step - loss: 0.2763 - accuracy:
Epoch 87/100
134/147 [==========================>...] - ETA: 0s - loss: 0.2881 - accuracy: 0.9
Epoch 87: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2859 - accuracy:
Epoch 88/100
138/147 [===========================>..] - ETA: 0s - loss: 0.3158 - accuracy: 0.9
Epoch 88: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.3159 - accuracy:
Epoch 89/100
```

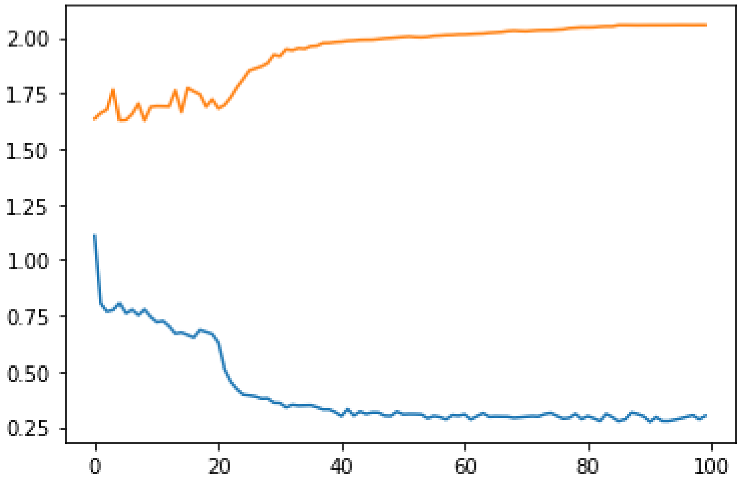
```
136/147 [==========================>...] - ETA: 0s - loss: 0.3049 - accuracy: 0.9
Epoch 89: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.3090 - accuracy:
Epoch 90/100
134/147 [==========================>...] - ETA: 0s - loss: 0.2921 - accuracy: 0.9
Epoch 90: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2986 - accuracy:
Epoch 91/100
144/147 [============================>.] - ETA: 0s - loss: 0.2733 - accuracy: 0.9
Epoch 91: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2740 - accuracy:
Epoch 92/100
144/147 [============================>.] - ETA: 0s - loss: 0.2932 - accuracy: 0.9
Epoch 92: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2957 - accuracy:
Epoch 93/100
143/147 [============================>.] - ETA: 0s - loss: 0.2780 - accuracy: 0.9
Epoch 93: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2778 - accuracy:
Epoch 94/100
136/147 [==========================>...] - ETA: 0s - loss: 0.2817 - accuracy: 0.9
Epoch 94: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2776 - accuracy:
Epoch 95/100
136/147 [==========================>...] - ETA: 0s - loss: 0.2792 - accuracy: 0.9
Epoch 95: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2833 - accuracy:
Epoch 96/100
136/147 [==========================>...] - ETA: 0s - loss: 0.2898 - accuracy: 0.9
Epoch 96: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2896 - accuracy:
Epoch 97/100
133/147 [==========================>...] - ETA: 0s - loss: 0.2990 - accuracy: 0.9
Epoch 97: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2972 - accuracy:
Epoch 98/100
136/147 [==========================>...] - ETA: 0s - loss: 0.3119 - accuracy: 0.9
Epoch 98: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.3037 - accuracy:
Epoch 99/100
133/147 [==========================>...] - ETA: 0s - loss: 0.2912 - accuracy: 0.9
Epoch 99: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.2854 - accuracy:
Epoch 100/100
134/147 [==========================>...] - ETA: 0s - loss: 0.2990 - accuracy: 0.9
Epoch 100: val_accuracy did not improve from 0.53555
147/147 [==============================] - 1s 4ms/step - loss: 0.3014 - accuracy:
```

```
newmodel = load_model('bestweights.h5')
```

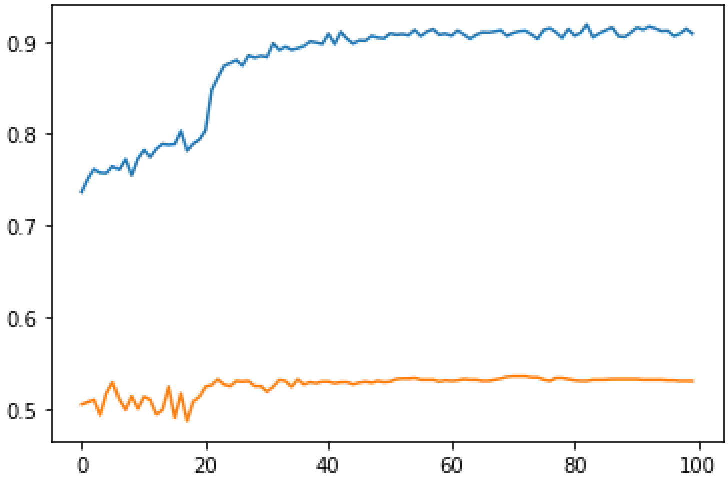
```
newmodel.evaluate(X_test, y_test)
```

```
49/49 [==============================] - 0s 3ms/step - loss: 2.0215 - accuracy: 0.54
[2.0214686393737793, 0.5432415008544922]
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.show()
```



```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.show()
```



Predicting on Xtest

```
ypred = audioANN.predict(X_test)
```

```
49/49 [==============================] - 0s 2ms/step
```

```
ypred
```

```
array([[5.5824575e-04, 6.3896581e-04, 1.3610960e-03, ..., 4.7067594e-04,
        1.3329077e-04, 9.9547452e-01],
       [2.2470087e-03, 1.4630765e-01, 3.7246179e-03, ..., 1.0094283e-03,
        1.9410166e-03, 1.1379361e-03],
       [6.8626833e-01, 1.0694664e-02, 6.2532448e-03, ..., 9.2954123e-03,
        2.0370705e-03, 2.9694315e-03],
       ...,
       [5.3913653e-04, 5.6991680e-04, 1.3286794e-03, ..., 4.5361184e-04,
        1.2835462e-04, 9.9566984e-01],
       [6.9005467e-04, 9.9137259e-01, 6.3308270e-04, ..., 6.5012230e-04,
        3.8716369e-04, 6.5987580e-04],
       [8.4950286e-04, 7.5332663e-04, 9.9284667e-01, ..., 5.9558294e-04,
        4.3656596e-04, 1.9244319e-03]], dtype=float32)
```

```
import numpy as np
ypredclasses = np.argmax(ypred, axis=-1)
```

```
ypredclasses
```

```
array([7, 3, 0, ..., 7, 1, 2])
```

```
y_actual = enc.inverse_transform(ypredclasses)
```

```
y_actual
```

```
array(['yes', 'no', 'down', ..., 'yes', 'go', 'left'], dtype='<U5')
```

```
audioANN.evaluate(X_test, y_test)
```

```
49/49 [==============================] - 0s 2ms/step - loss: 2.0417 - accuracy: 0.54
[2.0417487621307373, 0.5413196682929993]
```

```
newmodel.evaluate(X_test, y_test)
```

```
49/49 [==============================] - 0s 3ms/step - loss: 2.0215 - accuracy: 0.54
[2.0214686393737793, 0.5432415008544922]
```

Deployment - website

Connecting webpage with ANN

✓ 0s completed at 12:55 PM ● ×