```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_rem
```

```
from __future__ import print_function
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from time import time
#np.random.seed(1337)  # for reproducibility

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Convolution1D, MaxPooling1D
from keras.utils import np_utils
from keras.callbacks import TensorBoard
```

```
# set parameters:
test_dim = 499
maxlen = 100
nb_filter = 256
filter_length_1 = 10
filter_length_2 = 5
hidden_dims = 750
nb_epoch = 12
nb_classes = 2
split_ratio = 0.15

print('Loading data...')

# X = np.load('/content/drive/My Drive/Colab Notebooks/data/numpy_vectors/x_test_mfcc_500_50:50_samples_sliced_out.npy')
```

```
# y = np.load('/content/drive/My Drive/Colab Notebooks/data/numpy_vectors/y_label_500_50:50_samples_sliced_out.npy')
X = np.load('/content/drive/My Drive/Colab Notebooks/data/numpy_vectors/x_3:1_samples_out.npy')
y = np.load('/content/drive/My Drive/Colab Notebooks/data/numpy_vectors/y_3:1_samples_out.npy')
print(X.shape)
print(y.shape)
```

```
Loading data...
(3155, 499, 13)
(3155,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_ratio)
Y_train = y_train
Y_test = y_test


import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D
from keras.layers import Activation, Dropout, Flatten, Dense
nb_train_samples = X.shape
input_shape = (test_dim, 13)
for batch_size in range(25, 26, 5):
    print('Build model...')
    model = Sequential()

    model = Sequential()
    model.add(Conv1D(32, (3), input_shape=input_shape))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=(2)))

    model.add(Conv1D(32, (3)))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=(2)))

    model.add(Conv1D(64, (3)))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=(2)))
```

```
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

model.fit(X_train, Y_train, steps_per_epoch=nb_train_samples[0] // batch_size,
        nb_epoch=10, shuffle='true', verbose=1)

Y_preds = model.predict(X_test)
for i in range(len(Y_preds)):
    print(Y_preds[i], Y_test[i])
score = model.evaluate(X_test, Y_test, verbose=1)
print(score)
```

```
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[0.] 0
[1.] 1
[0.] 0
[0.] 0
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 0
[1.] 1
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[0.] 0
[1.] 1
```

```
[0.] 0
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[1.] 1
[0.] 0
[1.] 1
[1.] 1
[0.] 0
[0.] 0
474/474 [==============================] - 0s 294us/step
[0.07283044824621568, 0.9936708860759493]
```