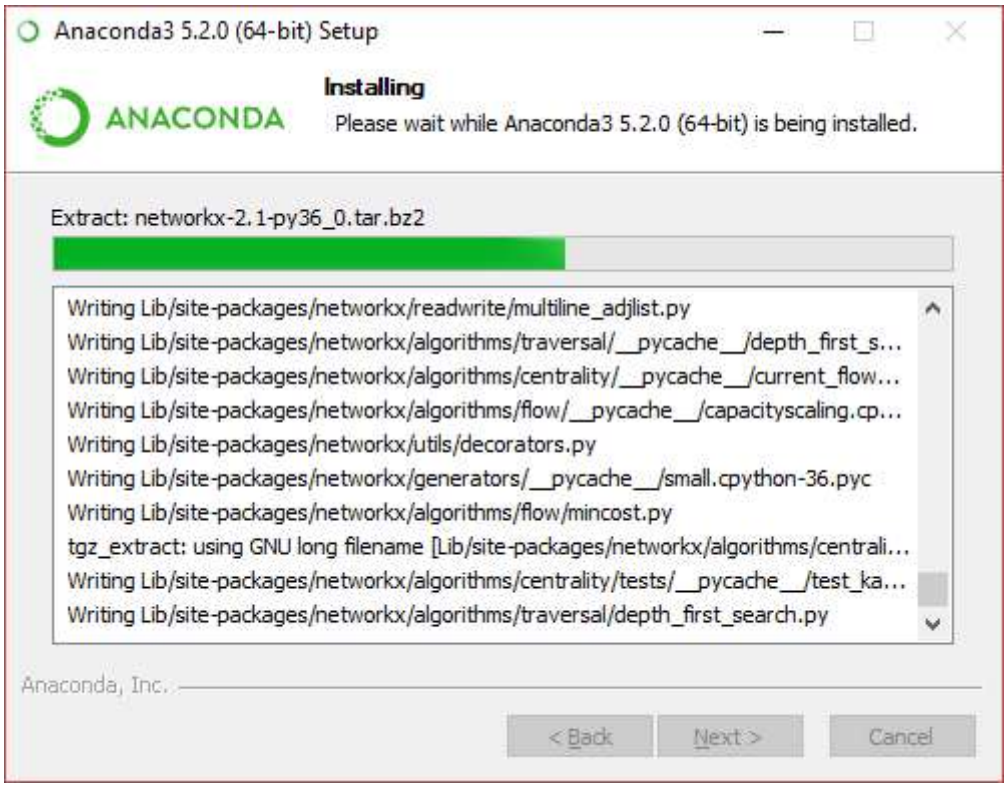
Χρησιμοποιήσαμε το anaconda για τη χρήση tensorflow σε windows



Στην python με τις βιβλιοθήκες pydub , pyaudio , keras , numpy, librosa, tensorflow, wave, random , glob, sklearn υλοποιήσαμε ένα νευρωνικό δίκτυο.

Μέσω αρχείων dataset spoken_numbers τους 16bit από το pannous
και υλοποιήσαμε μετατροπή για πιο μεγάλη ομοιομορφία.

```python
from pydub import AudioSegment
from pydub.silence import split_on_silence
import glob

DATA_DIRECTORY = './data/spoken_numbers_pcm/'
empty = AudioSegment.silent(1000)
fnames = glob.glob(DATA_DIRECTORY + "*.wav")

for fname in fnames:
    audioFile = AudioSegment.from_wav(fname)
    audioFile = empty + audioFile + empty

    chunk = split_on_silence(audioFile, min_silence_len=150, silence_thresh=-50)[0]
    chunk.export(fname, format='wav')
```

Με το audacity ηχογραφήσαμε σε 16bit 176400hz τα οποία ήταν τα metadata των δεδομένων μας από 0 μέχρι 9 ψηφία.

Φτιάξαμε ένα random output generator το οποίο έπαιρνε και τα ήδη υπάρχον data αλλα χωρίς την τροποποίηση.

```python
from pydub import AudioSegment
from pydub.silence import split_on_silence
import glob
import random

data_dir = './data/16bit/'
empty = AudioSegment.silent(1000)
fnames = glob.glob(data_dir + "*.wav")
n = len(fnames) - 1
digits = random.randint(4, 10) # number of digits
output_dir = "./data/tests/output.wav"

# generate a wav file with random digits(4-10)
output = empty
for i in range(0, digits):
    idx = random.randint(0, n) # random index
    audio = AudioSegment.from_wav(fnames[idx]) # random audio
    output = output + audio + empty
output.export(output_dir, format='wav')
```

Φτιάξαμε ένα αρχείο για record αν δεν έχει ο χρήστης audacity το οποίο λειτουργεί για 15 δευτερόλεπτα.

```python
import pyaudio
import wave

FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 176400
CHUNK = 512
RECORD_SECONDS = 15
WAVE_OUTPUT_FILENAME = "./data/tests/result.wav"
audio = pyaudio.PyAudio()

# start Recording
stream = audio.open(
    format=FORMAT,
    channels=CHANNELS,
    rate=RATE, input=True,
    frames_per_buffer=CHUNK
)
print("recording for 15 seconds")

frames = []
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)
print("finished recording")

# stop Recording
stream.stop_stream()
stream.close()
audio.terminate()

# save results
waveFile = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
waveFile.setnchannels(CHANNELS)
waveFile.setsampwidth(audio.get_sample_size(FORMAT))
waveFile.setframerate(RATE)
waveFile.writeframes(b''.join(frames))
waveFile.close()
```

Για test size χρησιμοποιούμε το 20% και για random seed το έτος.
Με τη glob διαβάζουμε τα αρχεία.

```python
import glob
import numpy as np # n-dimentional array math lib
import random
import librosa # music and audio analysis lib
import pyaudio
import wave
import os.path
# pydub: audio analysis lib
from pydub import AudioSegment
from pydub.silence import split_on_silence
# import scikit-learn related helper function train_test_split
from sklearn.model_selection import train_test_split
# importing keras related helper classes:
#  LSTM, Dense, Dropout, Flatten, Sequential, Adam, EarlyStopping and ModelCheckpoint
import keras
from keras.layers import LSTM, Dense, Dropout, Flatten, Conv2D
from keras.models import Sequential, load_model
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping, ModelCheckpoint

SEED = 2018
DATA_DIRECTORY = './data/spoken_numbers_pcm/'

# split all .wav files in a training set and validation set
wav_files = glob.glob(DATA_DIRECTORY + "*.wav")
X_train, X_val = train_test_split(wav_files, test_size=0.2, random_state=SEED)

# print training set's size and validation set's size
print("\ntraining set size: {}".format(len(X_train)))
print("validation set size: {}\n".format(len(X_val)))

N_FEATURES = 20
MAX_LEN = 120  # 120 -> test val , 80 old val
N_CLASSES = 10

# encoding function
def one_hot_encode(labels_dense, n_classes=10):
    return np.eye(n_classes)[labels_dense]

# read .wav files and transform them to usable output
```

Ο batch generator μεσω librosa τροποποιεί με mfcc τα wav ώστε να μπορούν να εισαχθούν στο νευρωνικό με το κατάλληλο format.

```
# read .wav files and transform them to usable output
def batch_generator(data, batch_size=16):
    while 1:
        random.shuffle(data)
        X, y = [], []
        for i in range(batch_size):
            wav_file = data[i]
            wave, sampling_rate = librosa.load(wav_file, mono=True)
            num = int(wav_file[26])
            label = one_hot_encode(num, N_CLASSES)
            y.append(label)
            mfcc = librosa.feature.mfcc(wave, sampling_rate)
            M = MAX_LEN - len(mfcc[0])
            # if M < 0:
            #     M = 0
            mfcc = np.pad(
                mfcc,
                ((0, 0), (0, M)),
                mode='constant',
                constant_values=0
            )
            X.append(np.array(mfcc))
        yield np.array(X), np.array(y)


# read .wav files and transform them to usable output *non generator function*
def take_nXs(data, n):
    random.shuffle(data)
    X, y = [], []
    for i in range(n):
        wav_file = data[i]
        wave, sampling_rate = librosa.load(wav_file, mono=True)
        label = int(wav_file[26])
        y.append(label)
        mfcc = librosa.feature.mfcc(wave, sampling_rate)
        M = MAX_LEN - len(mfcc[0])
        # if M < 0:
        #     M = 0
        mfcc = np.pad(
            mfcc,
            ((0, 0), (0, M)),
            mode='constant',
            constant_values=0
        )
        X.append(np.array(mfcc))
    return np.array(X), y
```

Παρόμοια με από πάνω αλλά δεν είναι generator συνάρτηση η take_nXs.
Η take_nXs2 όμοια δεν βάζει labels.
Εκτυπώνουμε τη μορφή dataset αφού παραχθεί από το generator.

```
# take_nXs2 is the same as take_nXs but it doesn't return the labels
def take_nXs2(data, n):
    X = []
    for i in range(n):
        wav_file = data[i]
        wave, sampling_rate = librosa.load(wav_file, mono=True)
        mfcc = librosa.feature.mfcc(wave, sampling_rate)
        M = MAX_LEN - len(mfcc[0])
        # print(len(mfcc[0]))
        # if M < 0:
        #     M = 0
        mfcc = np.pad(
            mfcc,
            ((0, 0), (0, M)),
            mode='constant',
            constant_values=0
        )
        X.append(np.array(mfcc))
    return np.array(X)


# print the shapes of training set and label set
X_set, y_set = next(batch_generator(X_train, batch_size=1))
print('shape of data set: {}'.format(X_set.shape))
print('shape of label set: {}\n'.format(y_set.shape))


# define hyperparameters
LEARNING_RATE = 0.0025 # 0.001
BATCH_SIZE = 16 # 64
N_EPOCHS = 5 # 50
DROPOUT = 0.3 # 0.5
INPUT_SHAPE = X_set.shape[1:]
STEPS_PER_EPOCH = 2240 // BATCH_SIZE  # SAMPLES // BATCH_SIZE # 50
```

Στη συνέχεια ορίζουμε υπερ παραμέτρους για το νευρωνικό με learning rate 0.0025 , batch size 16, 5 εποχές, dropout 0.3, και steps per epoch βάλαμε τα δείγματα // batch size.

Όταν έχουμε κάνει εκπαίδευση το αποθηκεύουμε και αν υπαρχει την επόμενη φορά να το φορτώσει.
Εδώ βλέπουμε τη δομή του μοντέλου.

```
model_path = './model.h5'
model = None
# check if the model is already trained
if not os.path.exists(model_path):
    # define neural network's architecture
    model = Sequential()
    model.add(
        LSTM(256, return_sequences=True, input_shape=INPUT_SHAPE, dropout=DROPOUT)
    ) # 256 old val, 50 new val
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))  # 128 old, 100 test val
    model.add(Dropout(DROPOUT))
    model.add(Dense(N_CLASSES, activation='softmax'))


    # we set the loss function, compile our model and output the model's summary
    opt = Adam(lr=LEARNING_RATE)
    model.compile(
        loss='categorical_crossentropy',
        optimizer=opt,
        metrics=['accuracy']
    )
    model.summary()


    # show training and validation accuracy on terminal
    history = model.fit_generator(
        generator=batch_generator(X_train, BATCH_SIZE),
        steps_per_epoch=STEPS_PER_EPOCH,
        epochs=N_EPOCHS,
        verbose=1,
        validation_data=batch_generator(X_val, 32),
        validation_steps=2240 // BATCH_SIZE
    )

    model.save('model.h5')  # save model's state(architecture, weights, etc..)
else:
    # load existing model
    model = load_model('model.h5')
```

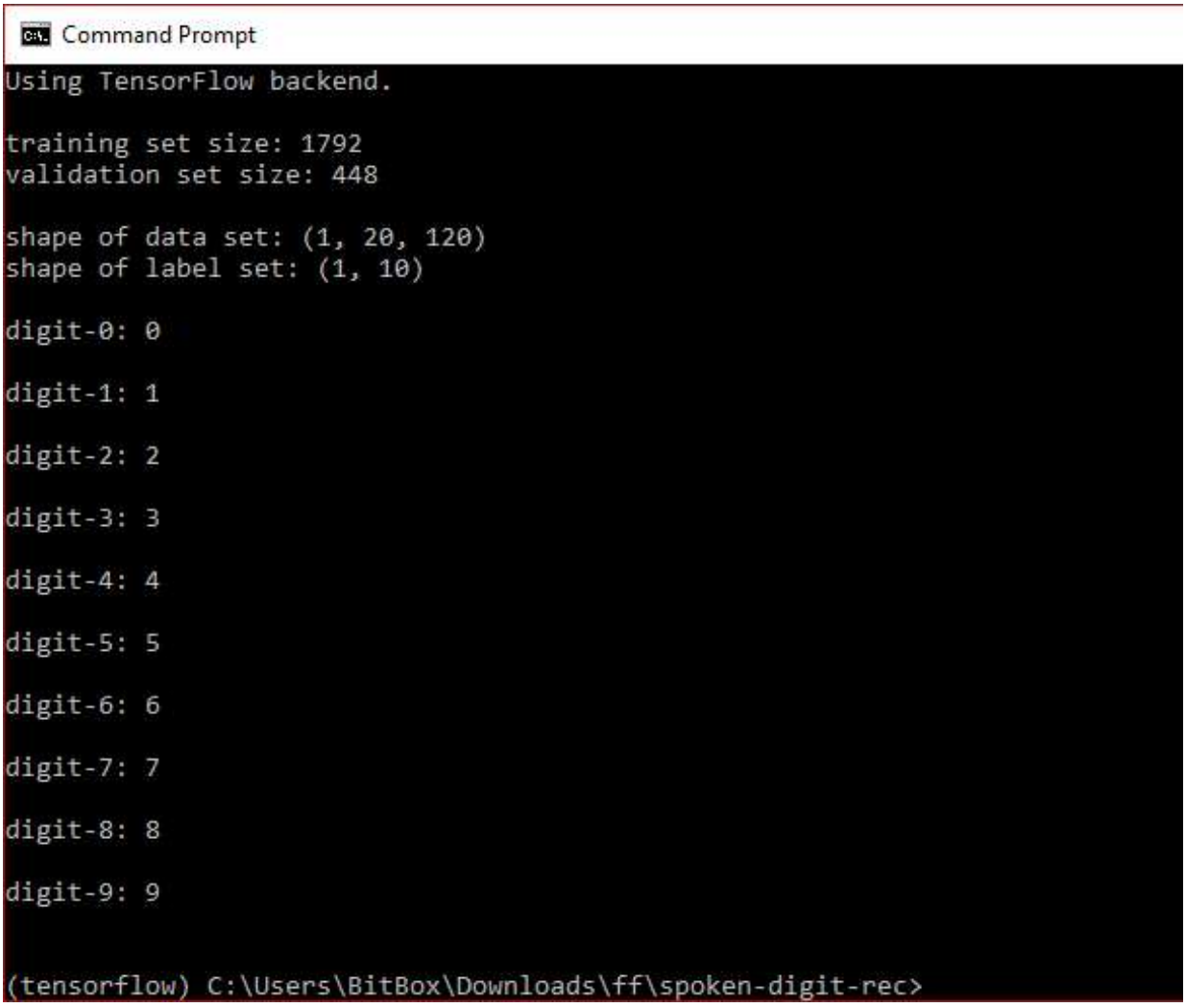
Εδώ δοκιμάζουμε την ηχογράφηση που κάναμε με το audacity. Την κόβουμε σε chunks με ελάχιστο μήκος σιωπής 150 milliseconds και όριο σιωπής -50.
Τα κάνει output μετα σε wav χωρισμένα.
Τα διαβάζει παλι με glob τα τα ταξινομεί και τεστάρει την πρόβλεψη στο δίκτυο που εκπαιδεύσαμε.
Έπειτα εκτυπώνει τα αποτελέσματα , των chunks και διπλά το προβλεπόμενο νούμερο.

```
# Using data from data/tests folder
file_path = "./data/tests/panos.wav"
audio = AudioSegment.from_wav(file_path) # read wav file

# split recording to chunks
chunks = split_on_silence(
    audio,
    min_silence_len=150,
    silence_thresh=-50
)

# save each chunk as .wav file
for i, chunk in enumerate(chunks):
    chunk.export("./data/chunks/chunk{0}.wav".format(i), format="wav")

# read all chunks
wav_chunks = glob.glob("./data/chunks/" + "*.wav") # find all .wav chunks
wav_chunks = sorted(wav_chunks)

X3_tests = take_nXs2(wav_chunks, len(wav_chunks)) # take all samples
predictions2 = model.predict_classes(X3_tests, batch_size=1, verbose=0) # predict output

# show predictions
idx = 0
for p in predictions2:
    print("digit-{0}: {1}\n".format(idx, p))
    idx += 1
```

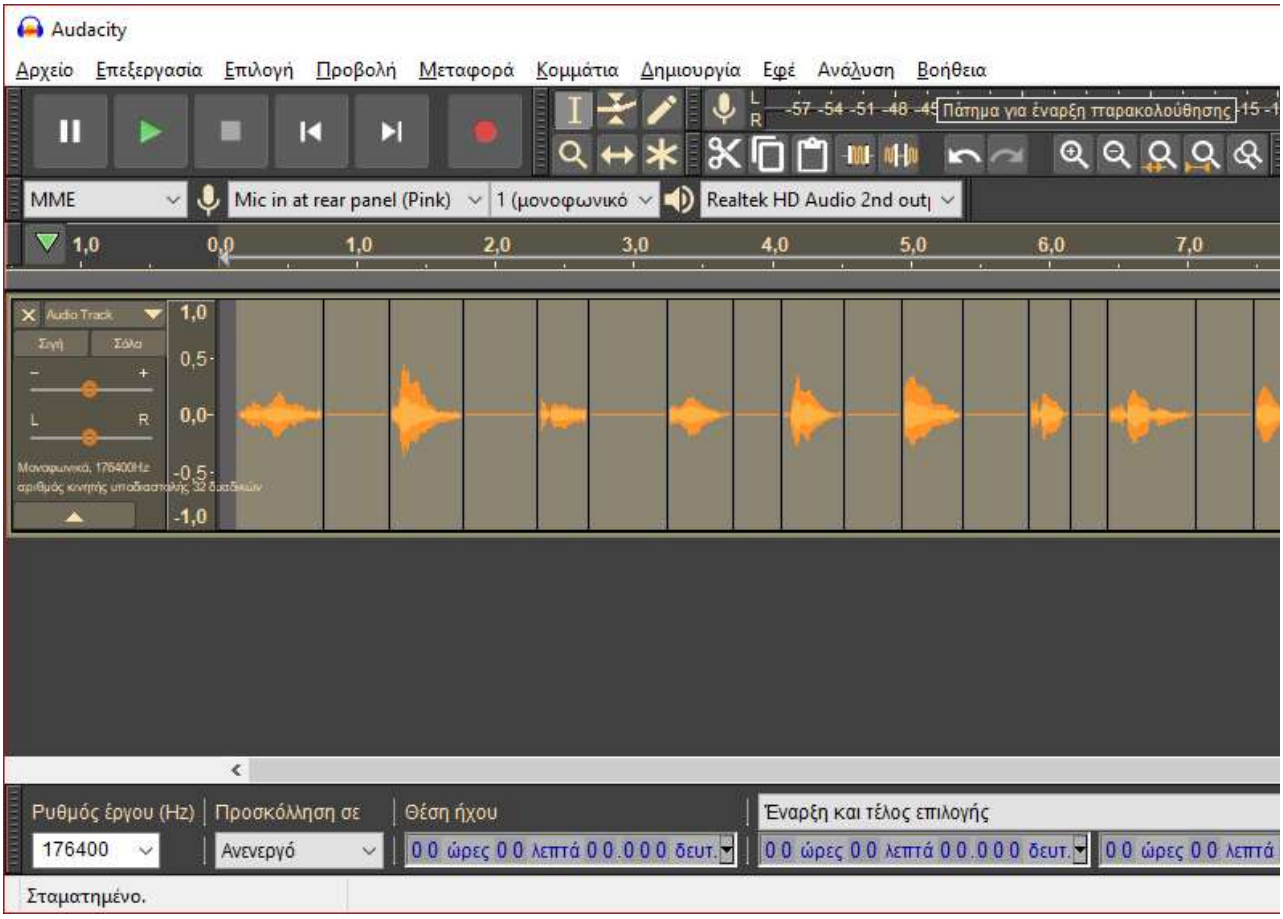
Τα αποτελέσματα για το output αρχείο που είναι generated από τα μη επεξεργασμένα αρχεία.

```
digit-0: 7

digit-1: 4

digit-2: 8

digit-3: 3

digit-4: 6

digit-5: 4

digit-6: 2

MacBook-Pro-Basiles:spoken-digit-rec vasilis$
```

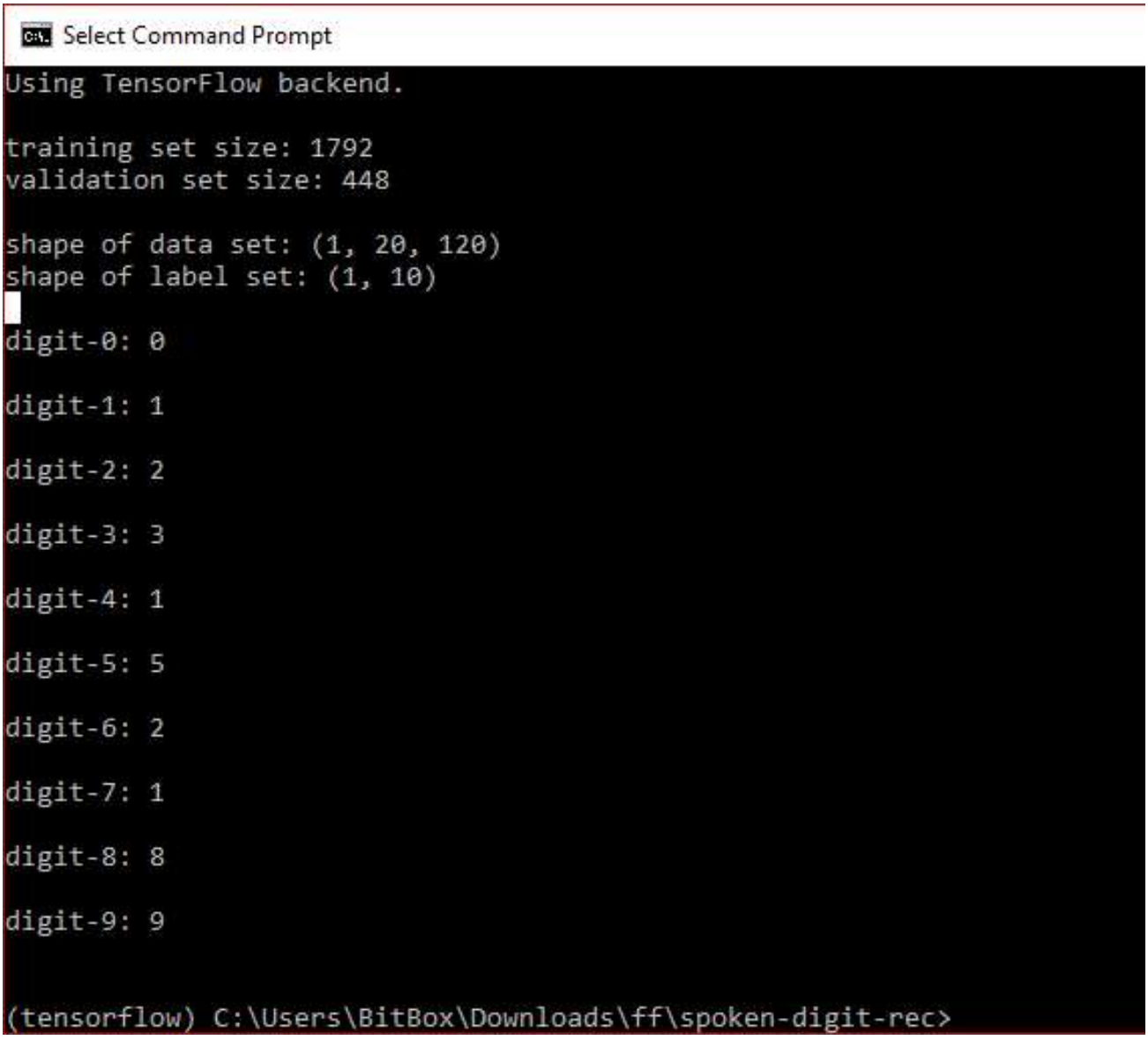
Και από την ηχογράφηση που κάναμε οι ιδιοι.



Δεύτερη ηχογράφηση με τη φωνή μας στο αρχείο panos2.wav



Τα αποτελέσματα από κάτω. Σε όλες τις ηχογραφήσεις έχει υποστεί μείωση θορύβου από audacity.


Select Command Prompt
Using TensorFlow backend.
training set size: 1792
validation set size: 448
shape of data set: (1, 20, 120)
shape of label set: (1, 10)
digit-0: 0
digit-1: 1
digit-2: 2
digit-3: 3
digit-4: 1
digit-5: 5
digit-6: 2
digit-7: 1
digit-8: 8
digit-9: 9
(tensorflow) C:\Users\BitBox\Downloads\ff\spoken-digit-rec>