计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：Trigger Word Detection		学号：201918130222
日期：2021/12/28	班级： 智能	姓名： 魏江峰
Email：2257263015@qq.com		

实验目的：

Implement a model which will beep every time you say "activate". After the model is completed, you will be able to record your own speech clips and trigger a prompt tone when the algorithm detects that you say "activate".

实验软件和硬件环境：

硬件环境：

处理器：Intel core i7 9750-H

电脑：神州 z7m-ct7nk

软件环境：

Pycharm 与 jupyter notebook

实验原理和方法：

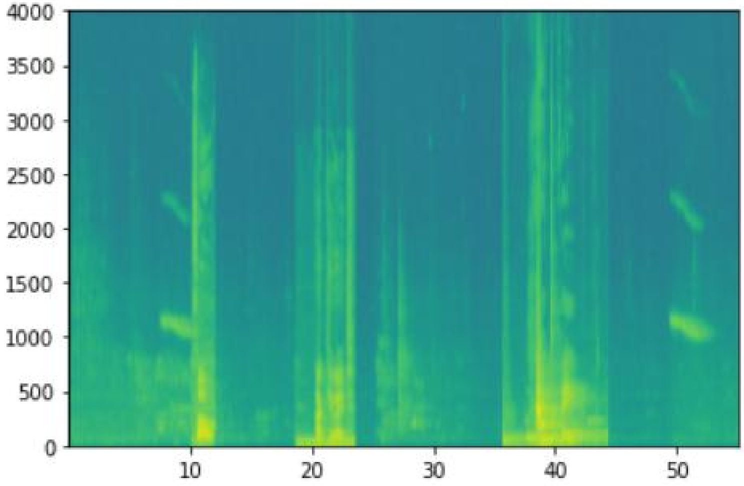
构建语音数据集并实现触发词检测算法（有时也称为关键字检测或唤醒字检测）。触发词检测技术允许 Amazon Alexa，Google Home，Apple Siri 和 Baidu DuerOS 等设备听到某个单词后唤醒。

实验步骤：（不要求罗列完整源代码）

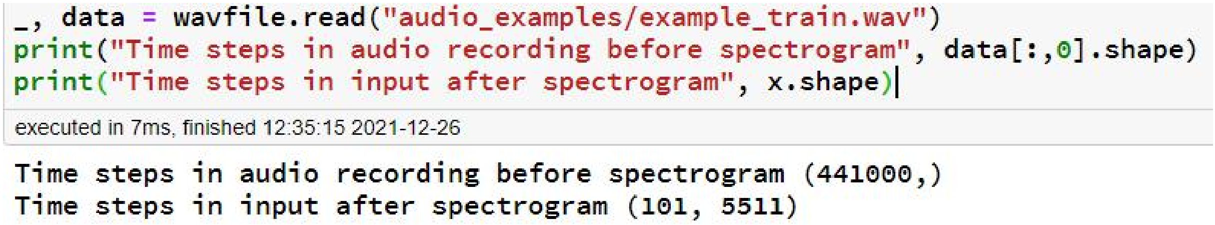
1.1 - Listening to the data

1.2 - From audio recordings to spectrograms¶

根据音频文件绘制出波形图：



加载音频数据：



```
_, data = wavfile.read("audio_examples/example_train.wav")
print("Time steps in audio recording before spectrogram", data[:,0].shape)
print("Time steps in input after spectrogram", x.shape)
```

executed in 7ms, finished 12:35:15 2021-12-26

```
Time steps in audio recording before spectrogram (441000,)
Time steps in input after spectrogram (101, 5511)
```

1.3 - Generating a single training example

实现 is_overlapping 函数

```
    overlap = False
    for previous_start, previous_end in previous_segments:
        if segment_start<= previous_end and segment_end >= previous_start:
            overlap = True
    return overlap
```

实现 insert_audio_clip：

```
segment_time = get_random_time_segment(segment_ms)
 while is_overlapping(segment_time,previous_segments):
  segment_time = get_random_time_segment(segment_ms)
  previous_segments.append(segment_time)
```

实现 insert_ones：

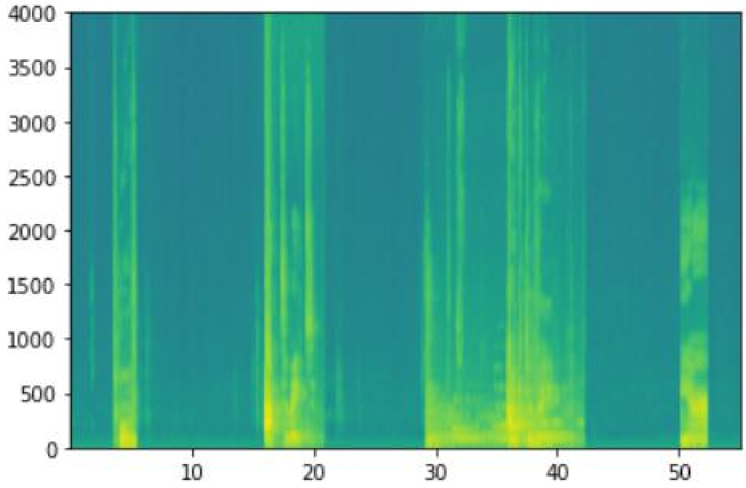
```
if (Ty-segment_end_y)>= 50:
      y[0][segment_end_y+1:segment_end_y+51] = 1
    else :
        y[0][(segment_end_y + 1):] = 1
```

实现 create_training_example：

```
  y = np.zeros((1,Ty))
  previous_segments = []

for random_activate in random_activates:
   background, segment_time =
    insert_audio_clip(background,random_activate,previous_segments)
        segment_start, segment_end =   segment_time
        y = insert_ones(y,segment_end)

for random_negative in random_negatives:
        background, _ =
```

insert_audio_clip(background,random_negative,previous_segments)

建立的训练样例：



1.4 - Full training set：加载已经预处理后的样例

1.5 - Development set：加载已经预处理后的 dev set

2 - Model

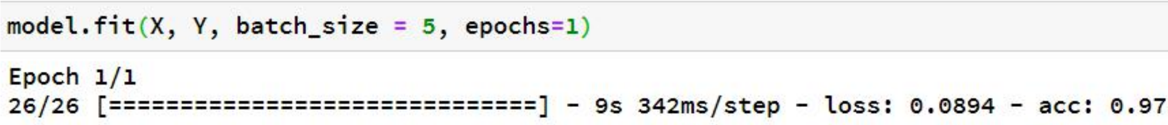
2.1 - Build the model

实现 model 函数：建立模型

```
# Step 1: CONV layer (≈4 lines)

    X = Conv1D(196,15,strides= 4)(X_input)                              # CONV1D

    X = BatchNormalization()(X)                        # Batch normalization

    X = Activation('relu')(X)                      # ReLu activation

    X = Dropout(0.8)(X)                                 # dropout (use 0.8)



    # Step 2: First GRU Layer (≈4 lines)

    X =GRU(128,return_sequences= True)(X)
```

```
     # GRU (use 128 units and return the sequences)

    X = Dropout(0.8)(X)                                    # dropout (use 0.8)

    X = BatchNormalization()(X)                                      # Batch normalization



    # Step 3: Second GRU Layer (≈4 lines)

    X = GRU( 128,return_sequences= True)(X)                                    # GRU (use 128 units and return the sequences)

    X = Dropout(0.8)(X)                                    # dropout (use 0.8)

    X = BatchNormalization()(X)                                     # Batch normalization

    X = Dropout(0.8)(X)                                     # dropout (use 0.8)



    # Step 4: Time-distributed dense layer (≈1 line)

    X = TimeDistributed(Dense(1, activation = "sigmoid"))(X) # time distributed    (sigmoid)
```

2.2 - Fit the model

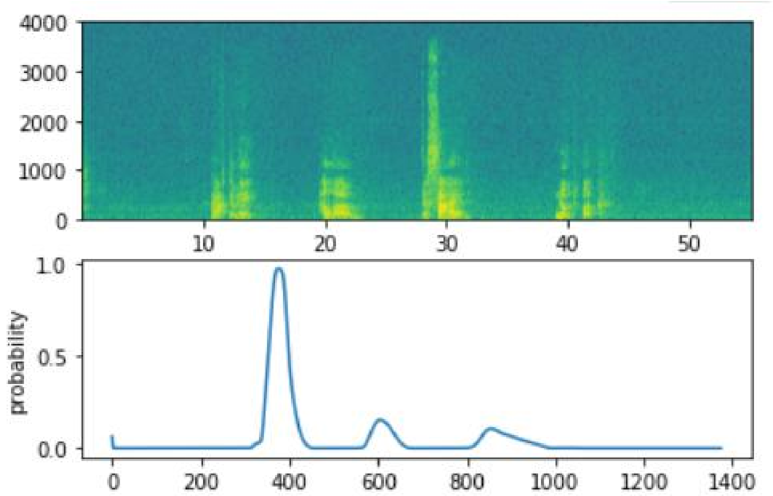


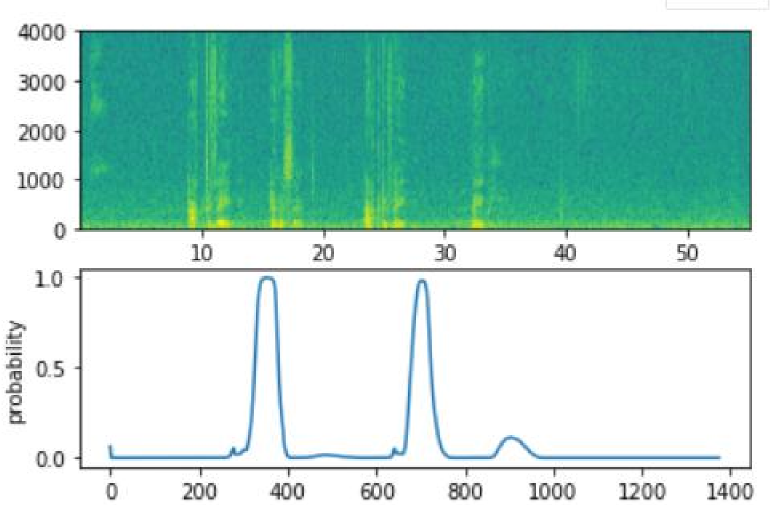
```
model.fit(X, Y, batch_size = 5, epochs=1)
```

```
Epoch 1/1
26/26 [==============================] - 9s 342ms/step - loss: 0.0894 - acc: 0.97
```

2.3 - Test the model

端到端 end-to-end 深度学习方法可用于构建非常有效的触发字检测系统。





可以看出模型大致正确预测了位置。

结论分析：

数据合成是为语音问题创建大型训练集的有效方法，特别是触发单词检测。

在将音频数据传递到 RNN，GRU 或 LSTM 之前，使用频谱图和可选的 1D conv layer 是常见的预处理步骤。

端到端 end-to-end 深度学习方法可用于构建非常有效的触发字检测系统。