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：Spoken Keyword spotting		学号：201900301174
日期：2021-12-20	班级： 智能 19	姓名： 韩旭
Email：hanx@mail.sdu.edu.cn		

实验目的：

Introduction

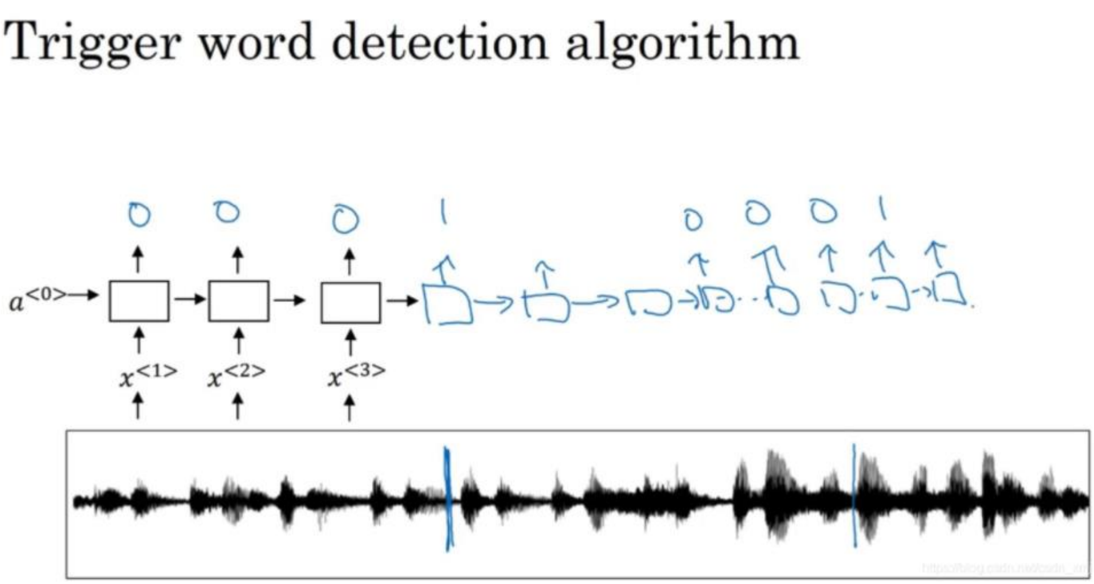
In this assignment, you will build a voice dataset and implement an algorithm for Spoken keyword spotting (sometimes called wake-up word or trigger word detection).

- You will 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：

Termius
Jupyter notebook
RTX3090
Xeon Gold 6226R
Tensorflow 1.15.5
Keras 2.2.5
Numpy 1.18.5
H5py 2.10.0

实验原理和方法：



构建一个 RNN，把一个音频片段计算出它的声谱图特征得到特征向量，然后把它放到 RNN 中，最后定义我们的目标标签 y。假如音频片段中的这一点是某人刚刚说完一个触发字，那么在这一点之前，可以在训练集中把目标标签都设为 0，然后在这个点之后把目标标签设为 1。假如在一段时间之后，触发字又被说了一次，那么就可以再次在这个点之后把目标标签设为 1。不过该算法一个明显的缺点就是它构建了一个很不平衡的训练集，0 的数量比 1 多太多了。

代码：is_overlapping：检查是否有重叠

```
### START CODE HERE ### (≈ 4 line)
# Step 1: Initialize overlap as a "False" flag. (≈ 1 line)
overlap = False

# Step 2: loop over the previous_segments start and end times.
# Compare start/end times and set the flag to True if there is an overlap (≈ 3 lines)
for previous_start, previous_end in previous_segments:
    if segment_start<=previous_end and segment_end>=previous_start:
        overlap = True
### END CODE HERE ###
```

```
In [13]:
    overlap1 = is_overlapping((950, 1430), [(2000, 2550), (260, 949)])
    overlap2 = is_overlapping((2305, 2950), [(824, 1532), (1900, 2305), (3
    print("Overlap 1 = ", overlap1)
    print("Overlap 2 = ", overlap2)
```

executed in 6ms, finished 18:03:39 2021-12-20

```
Overlap 1 =  False
Overlap 2 =  True
```

Expected Output:

Overlap 1	False
Overlap 2	True

代码：insert_audio_clip：根据提示写代码，插入音频

```
### START CODE HERE ###
# Step 1: Use one of the helper functions to pick a random time segment onto which to insert
# the new audio clip. (≈ 1 line)
segment_time = get_random_time_segment(segment_ms)

# Step 2: Check if the new segment_time overlaps with one of the previous_segments. If so, keep
# picking new segment_time at random until it doesn't overlap. (≈ 2 lines)
while is_overlapping(segment_time,previous_segments):
    segment_time = get_random_time_segment(segment_ms)

# Step 3: Add the new segment_time to the list of previous_segments (≈ 1 line)
previous_segments.append(segment_time)
### END CODE HERE ###
```

结果：由于随机数或者 numpy 版本不完全一样，导致结果不完全相同，但结果正确

In [15]:
```
np.random.seed(5)
audio_clip, segment_time = insert_audio_clip(backgrounds[0], activates
audio_clip.export("insert_test.wav", format="wav")
print("Segment Time: ", segment_time)
IPython.display.Audio("insert_test.wav")
```
executed in 61ms, finished 18:03:39 2021-12-20

```
Segment Time:  (7286, 8864)
```

0:00 / 0:10

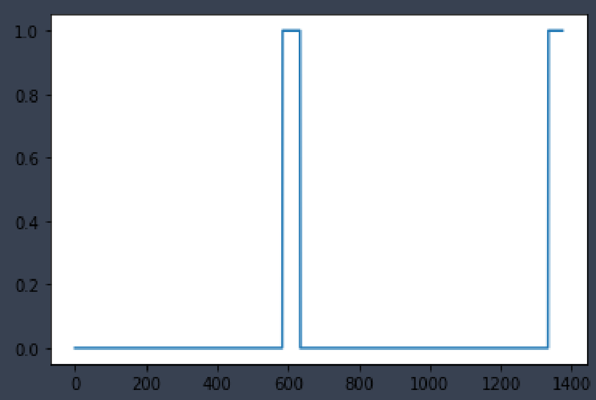
代码：insert_ones 正确的地方插入 1 作为 label

```
# Add 1 to the correct index in the background label (y)
### START CODE HERE ### (≈ 3 lines)
for i in range(segment_end_y+1,segment_end_y+51):
    if i < Ty:
        y[0, i] = 1.
### END CODE HERE ###
```

结果正确

In [18]:
```
arr1 = insert_ones(np.zeros((1, Ty)), 9700)
plt.plot(insert_ones(arr1, 4251)[0,:])
print("sanity checks:", arr1[0][1333], arr1[0][634], arr1[0][635])
```
executed in 132ms, finished 18:03:39 2021-12-20

```
sanity checks: 0.0 1.0 0.0
```



Expected Output

sanity checks:	0.0 1.0 0.0

代码：create_training_example 根据提示写代码，制作训练样本

```python
### START CODE HERE ###
# Step 1: Initialize y (label vector) of zeros (≈ 1 line)
y = np.zeros((1,Ty))

# Step 2: Initialize segment times as empty list (≈ 1 line)
previous_segments = []
### END CODE HERE ###

# Select 0-4 random "activate" audio clips from the entire list of "activates" recordings
number_of_activates = np.random.randint(0, 5)
random_indices = np.random.randint(len(activates), size=number_of_activates)
random_activates = [activates[i] for i in random_indices]

### START CODE HERE ### (≈ 3 lines)
# Step 3: Loop over randomly selected "activate" clips and insert in background
for random_activate in random_activates:
    # Insert the audio clip on the background
    background, segment_time = insert_audio_clip(background,random_activate,previous_segments)
    # Retrieve segment_start and segment_end from segment_time
    segment_start, segment_end = segment_time
    # Insert labels in "y"
    y = insert_ones(y,segment_end)
### END CODE HERE ###
```

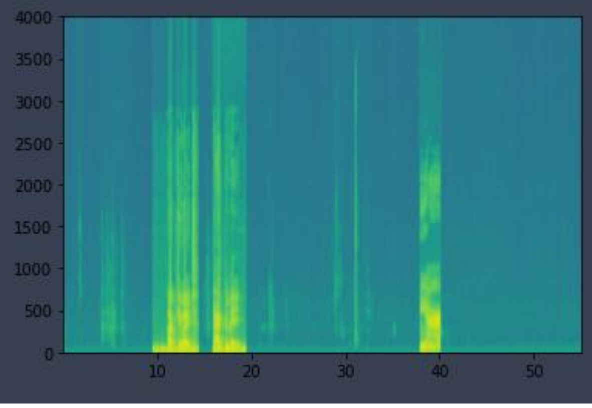
```python
# Select 0-2 random negatives audio recordings from the entire list of "negatives" recordings
number_of_negatives = np.random.randint(0, 3)
random_indices = np.random.randint(len(negatives), size=number_of_negatives)
random_negatives = [negatives[i] for i in random_indices]

### START CODE HERE ### (≈ 2 lines)
# Step 4: Loop over randomly selected negative clips and insert in background
for random_negative in random_negatives:
    # Insert the audio clip on the background
    background, _ = insert_audio_clip(background,random_negative,previous_segments)
### END CODE HERE ###
```

结果正确

```
In [20]:  x, y = create_training_example(backgrounds[0], activates, negatives)
```

executed in 648ms, finished 18:03:40 2021-12-20

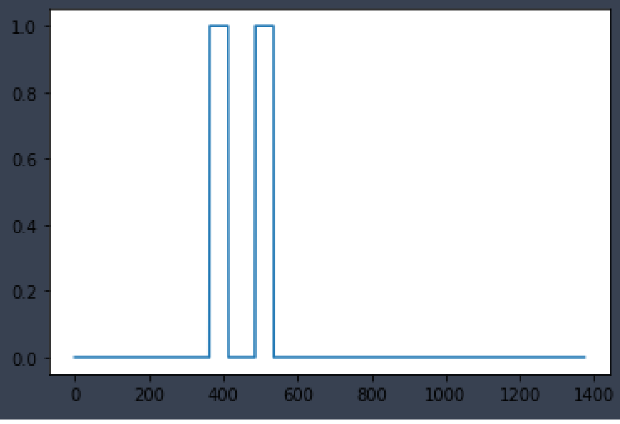
```
File (train.wav) was saved in your directory.
```




```
plt.plot(y[0])
```

executed in 381ms, finished 18:03:40 2021-12-20

```
[<matplotlib.lines.Line2D at 0x7f2441fe3ee0>]
```



代码：model，根据模型图和提示写代码

```python
### START CODE HERE ###

# Step 1: CONV layer (≈4 lines)
X = Conv1D(196,15,strides=4)(X_input)                         # CONV1D
X = BatchNormalization()(X)                          # Batch normalization
X = LeakyReLU(alpha=0.2)(X)                          # ReLu activation
X = Dropout(0.8)(X)                         # dropout (use 0.8)

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

### END CODE HERE ###
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 5511, 101)]       0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 1375, 196)         297136
_________________________________________________________________
batch_normalization_3 (Batch (None, 1375, 196)         784
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 1375, 196)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1375, 196)         0
_________________________________________________________________
gru_2 (GRU)                  (None, 1375, 128)         124800
_________________________________________________________________
dropout_5 (Dropout)          (None, 1375, 128)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 1375, 128)         512
_________________________________________________________________
gru_3 (GRU)                  (None, 1375, 128)         98688
_________________________________________________________________
dropout_6 (Dropout)          (None, 1375, 128)         0
_________________________________________________________________
batch_normalization_5 (Batch (None, 1375, 128)         512
_________________________________________________________________
dropout_7 (Dropout)          (None, 1375, 128)         0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 1375, 1)           129
=================================================================
Total params: 522,561
Trainable params: 521,657
Non-trainable params: 904
_________________________________________________________________
```

```
Train on 26 samples
Epoch 1/5
26/26 [==============================] - 30s 1s/sample - loss: 0.0757 - acc: 0.9759
Epoch 2/5
26/26 [==============================] - 29s 1s/sample - loss: 0.0522 - acc: 0.9820
Epoch 3/5
26/26 [==============================] - 29s 1s/sample - loss: 0.0517 - acc: 0.9811
Epoch 4/5
26/26 [==============================] - 17s 668ms/sample - loss: 0.0580 - acc: 0.9784
Epoch 5/5
26/26 [==============================] - 23s 876ms/sample - loss: 0.0501 - acc: 0.9838


<tensorflow.python.keras.callbacks.History at 0x7f22302b0070>
```

验证准确度 0.945

```
In [63]:
loss, acc = model.evaluate(X_dev, Y_dev)
print("Dev set accuracy = ", acc)
```

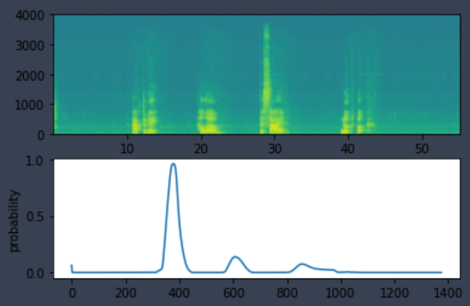
executed in 1.53s, finished 18:31:40 2021-12-20

```
25/25 [==============================] - 2s 61ms/sample - loss: 0.3897 - acc: 0.9451
Dev set accuracy =  0.94507635
```

测试数据：

```
In [38]:
filename = "./raw_data/dev/1.wav"
prediction = detect_triggerword(filename)
chime_on_activate(filename, prediction, 0.5)
IPython.display.Audio("./chime_output.wav")
```

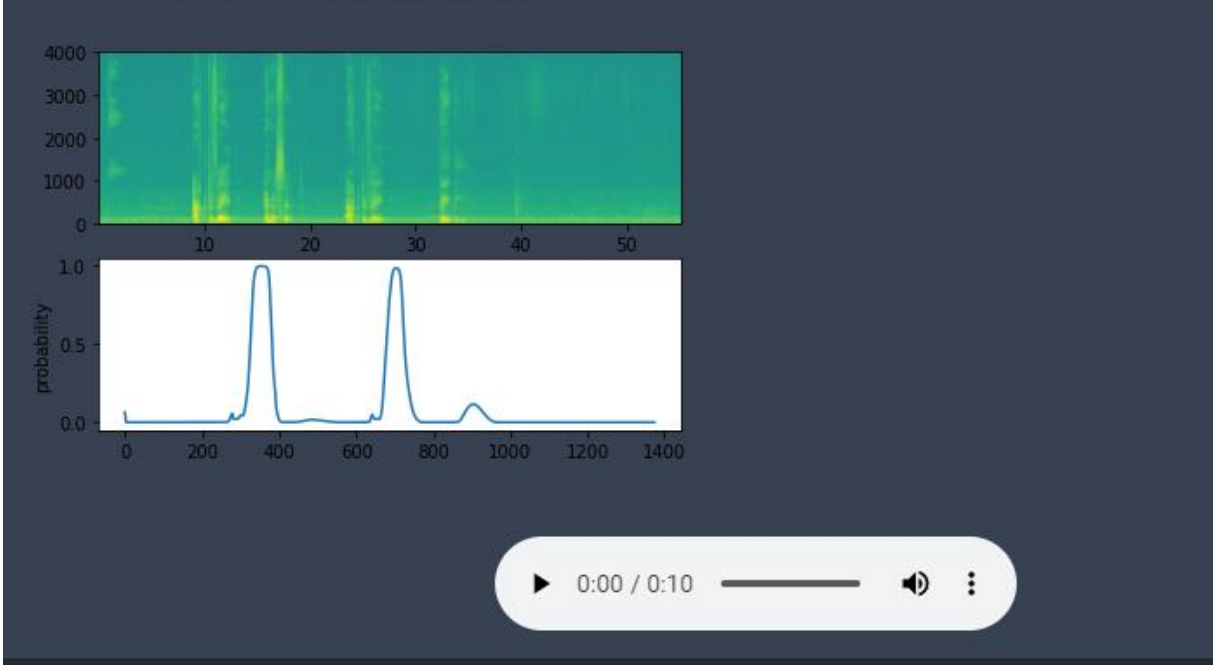
executed in 3.96s, finished 18:04:42 2021-12-20



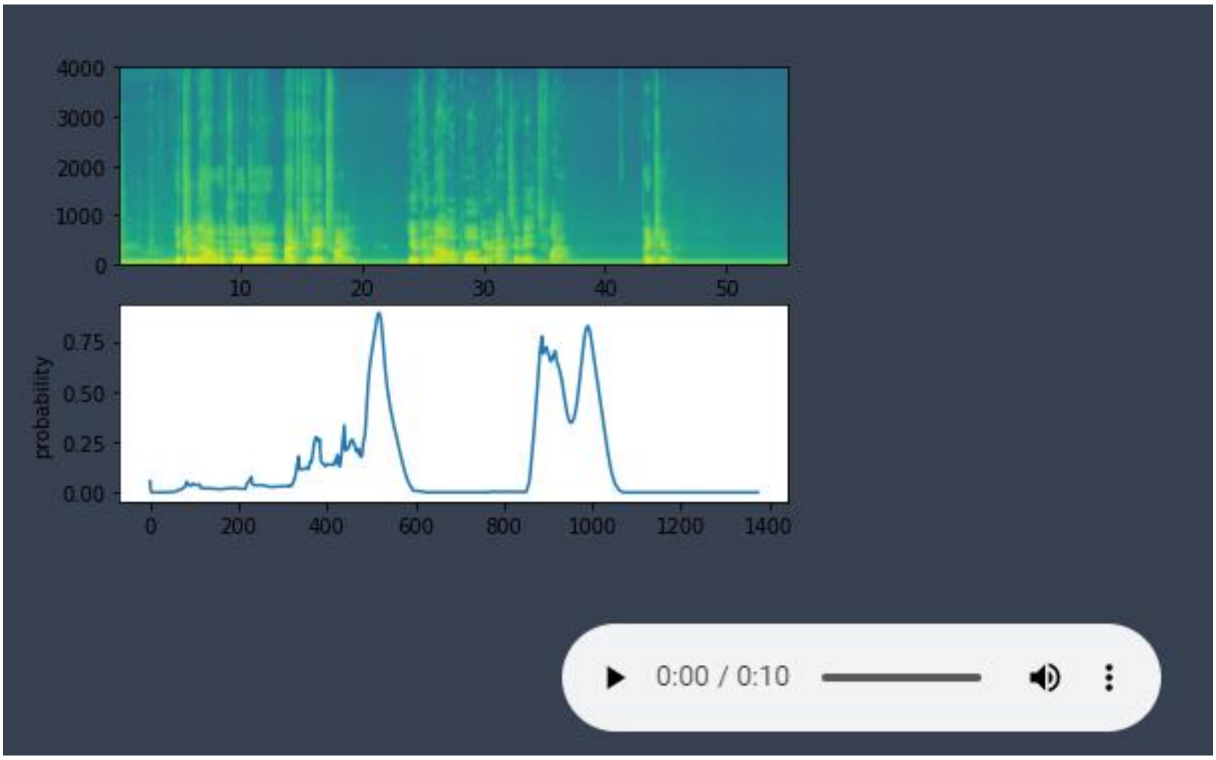
0:00 / 0:10

```
filename  = "./raw_data/dev/2.wav"
prediction = detect_triggerword(filename)
chime_on_activate(filename, prediction, 0.5)
IPython.display.Audio("./chime_output.wav")
```

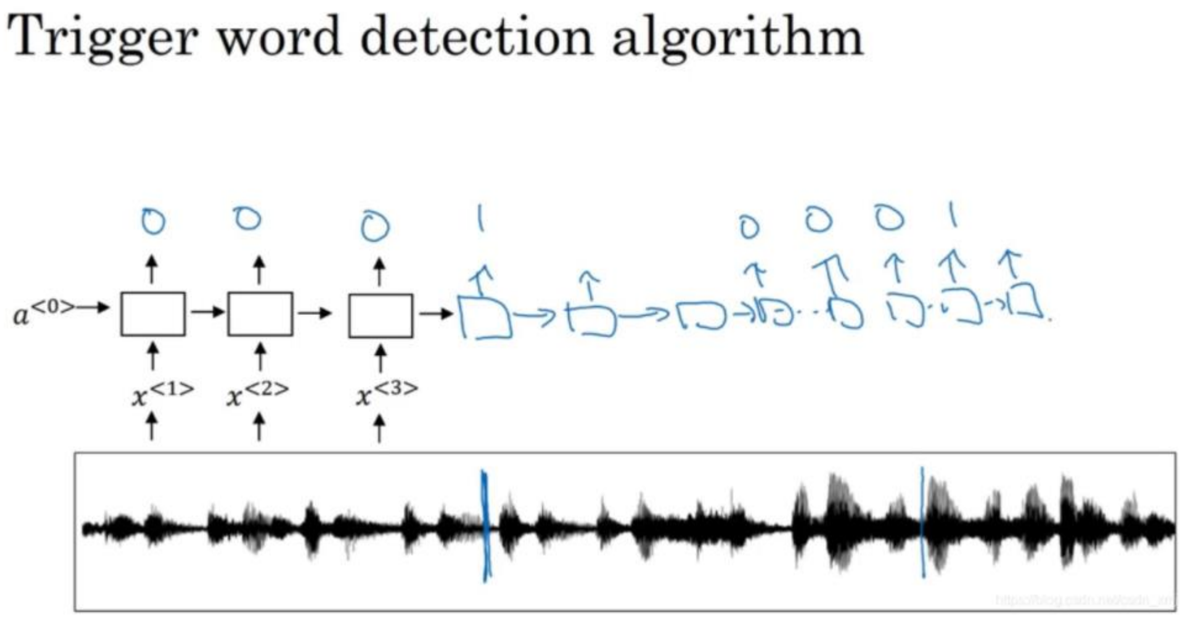
executed in 2.70s, finished 18:04:45 2021-12-20



自己的示例



结论分析与体会：



1， 构建一个 RNN，把一个音频片段计算出它的声谱图特征得到特征向量，然后把它放到 RNN 中，最后定义我们的目标标签 y。假如音频片段中的这一点是某人刚刚说完一个触发字，那么在这一点之前，可以在训练集中把目标标签都设为 0，然后在这个点之后把目标标签设为 1。假如在一段时间之后，触发字又被说了一次，那么就可以再次在这个点之后把目标标签设为 1。不过该算法一个明显的缺点就是它构建了一个很不平衡的训练集，0 的数量比 1 多太多了。

2， 遇到问题的时候要多 Google，在这里遇到的更多是环境不兼容的问题，tf2.X 到 tf1.X 的改头换面，keras 被收到 tf2，还有 numpy 的问题，都是版本不同的原因，以后还是多使用 torch。。。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1—3 道问答题：

1， Tf1 和 tf2 的不兼容已经是老生常谈了，keras 被收进了 tf2，所以不能直接调用 keras。

2， 还有加载模型的时候遇到了 numpy 版本的问题，于是降版本到 1.18.5。

3， 然后又遇到了 h5py 的问题，又降级到 2.10.0，问题解决。