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： Homework 8		学号：201900130024
日期：2021.12.9	班级： 数据 19	姓名： 刘士渤
Email：liuburger@qq.com		

实验目的：

build a voice dataset and implement an algorithm for Spoken keyword spotting (sometimes called wake-up word or trigger word detection).
Implement a model which will beep every time you say "activate".

实验软件和硬件环境：

VScode JupyterNoteBook
联想拯救者 Y7000p

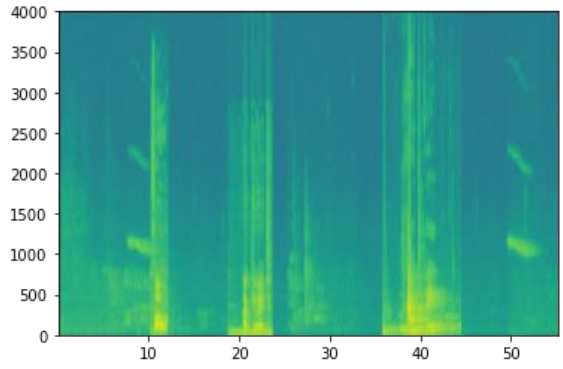
实验原理和方法：

GRU

实验步骤：（不要求罗列完整源代码）

1. 创建数据集：

主要思想是在一个 10s 的噪声背景中嵌入“active”和其他非“active”的音频（保证总时长还是 10s）。
观察样例的光谱图，黄绿色代表某个频段出现次数多，蓝色代表出现次数少：



- is_overlapping：
  判定两个片段重叠的依据是当前片段开始早于之前片段结束 且 当前片段结束晚于之前片段开始。

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

- insert_audio_clip:
  这个函数要用到之前写好的 get_random_time_segment（）生成随机的时间段并用 is_overlapping（）判断是否重叠；
  不重叠的话就将其加入到 previous_segments 中。

```
### START CODE HERE ###
# Step 1: Use one of the helper functions to pick a ra
# the new audio clip. (≈ 1 line)
segment_time = get_random_time_segment(segment_ms)
# Step 2: Check if the new segment_time overlaps with
# picking new segment_time at random until it doesn't
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 prev
previous_segments.append(segment_time)
### END CODE HERE ###
```

- insert_ones:
  在“active”语音结束后，需要将 50 个 t 的 label 置为 1，以表示识别到了“active”；
  这里我用循环的上界 min(Ty,segment_end_y+51)省去了原代码框架中每次循环都要判定。

```
### START CODE HERE ### (≈ 3 lines)
for i in range(segment_end_y+1,min(Ty,segment_end_y+51)):
    # if None < None:
        y[0, i] = 1
### END CODE HERE ###
```

- create_training_example:
  初始化 label 和 previous_segments：

```
### START CODE HERE ###
# Step 1: Initialize y
y = np.zeros((1,Ty))
# Step 2: Initialize seg
previous_segments = []
### END CODE HERE ###
```

背景噪声中需要插入若干个“active”和非“active”；
通过之前写好的 insert_audio_clip（）生成时间片段，再通过之前写好的 insert_ones（）更改 label：

```
### 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 ###
```

插入非“active”：

```
### 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 ###
```

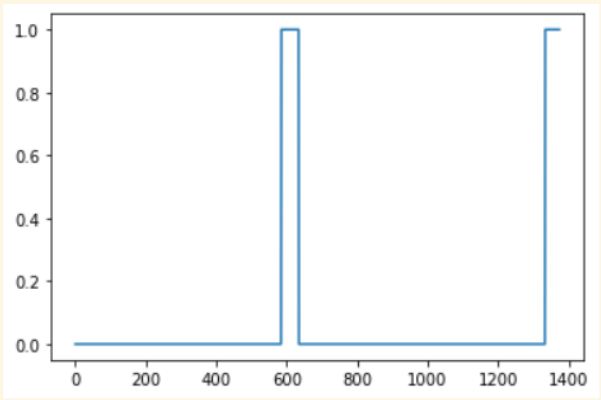
2. 完成模型：
通过提示我们可以很容易地写出代码；
提示中没有说明正则化的参数，那就用它默认的参数：

```
### START CODE HERE ###
# Step 1: CONV layer (≈4 lines)
X = Conv1D(filters=196,kernel_size=15,strides=4)(X_input)
X = BatchNormalization()(X)
X = Activation('relu')(X)
X = Dropout(0.8)(X)
# Step 2: First GRU Layer (≈4 lines)
X = GRU(units=128,return_sequences=True)(X)
X = Dropout(0.8)(X)
X = BatchNormalization()(X)
# Step 3: Second GRU Layer (≈4 lines)
X = GRU(units=128,return_sequences=True)(X)
X = Dropout(0.8)(X)
X = BatchNormalization()(X)
X = Dropout(0.8)(X)
# Step 4: Time-distributed dense layer (≈1 line)
X = TimeDistributed(Dense(1, activation = "sigmoid"))(X)
### END CODE HERE ###
```

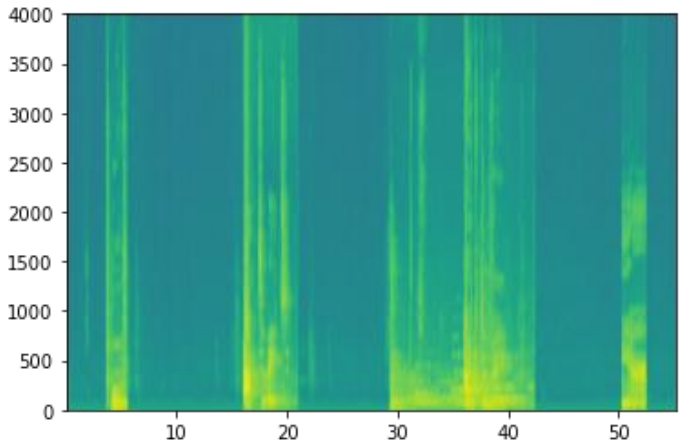
结论分析与体会：

1. 测试 insert_ones 函数，与样例对照是没有问题的：

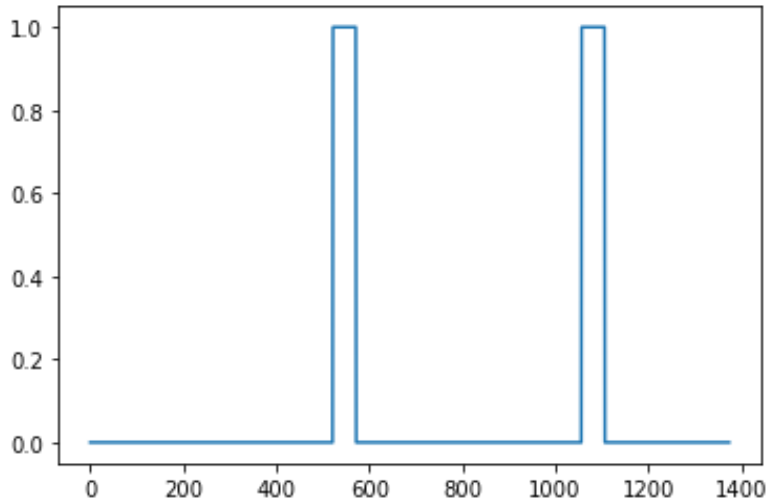
```
sanity checks: 0.0 1.0 0.0
```



2. 测试 create_training_example 函数，spectrogram 和样例不一样：



随后的 label 也和样例不一样，推测应该是随机种子的问题。



3. 模型的超参和样例不一样：

```
Total params: 522,561
Trainable params: 521,657
Non-trainable params: 904
```

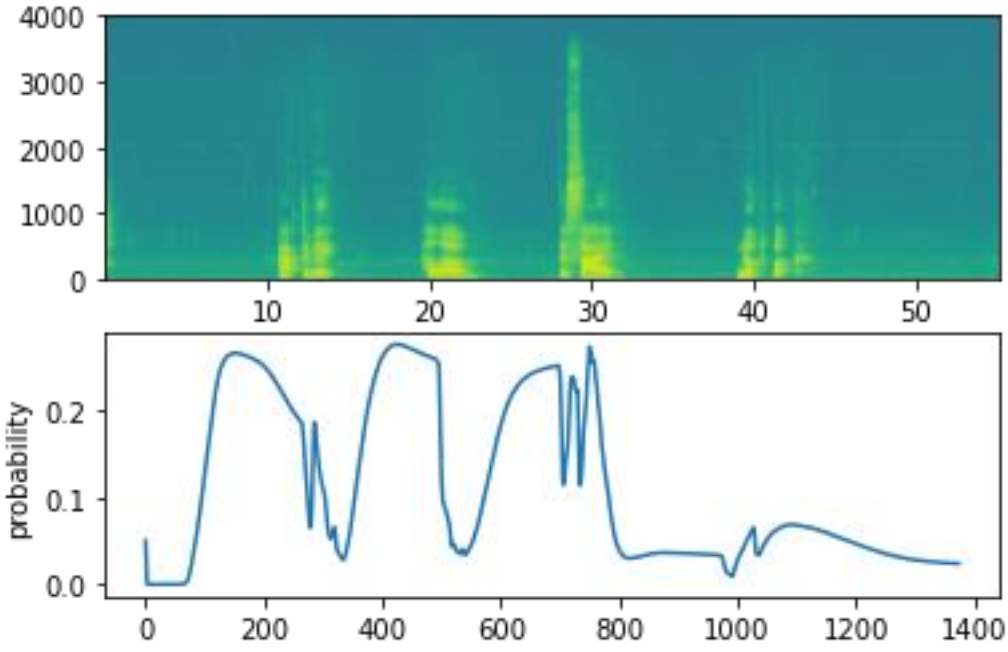
4. 用模型进行训练，得到了很低的 loss 和很高的准确率：

```
Epoch 1/1
26/26 [==============================] - 13s 506ms/step - loss: 0.0893 - acc: 0.9717
<keras.callbacks.History at 0x14299e8cdd8>
```

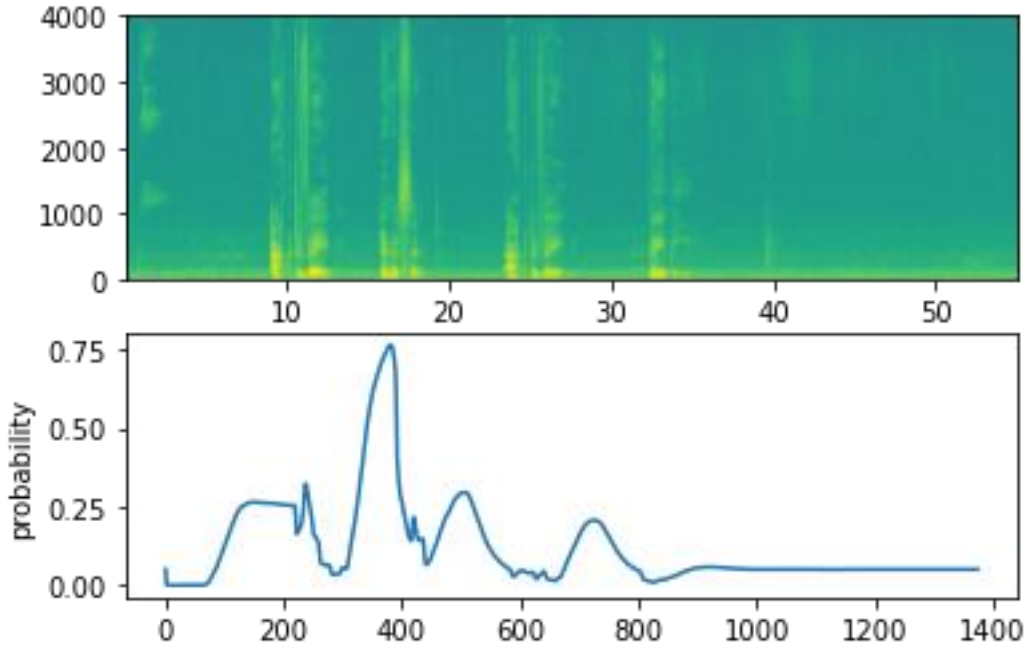
在 dev 上进行测试，准确率仍然很高：

```
25/25 [==============================] - 1s 45ms/step
Dev set accuracy =  0.9296872615814209
```

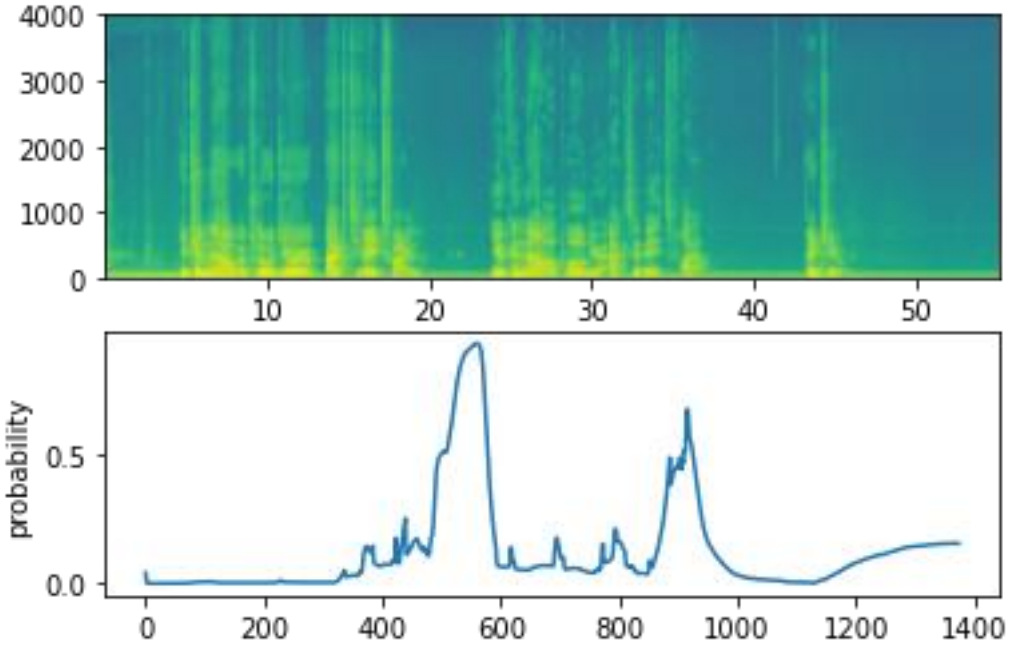
5. 用模型做预测，估计出现“active”的可能性，当 probability 超过阈值的时候，chime 一下，且为了避免在出现一次“active”的情况下 chime 两次，每 75step 最多有一次 chime；
阈值设置为 0.5，可以看出对于 1.wav，出现“active”的可能性很小，所以也不会 chime。



对于 2.wav，有一个峰值达到了 0.75，所以是有一次 chime 的。



6. 尝试自己的例子 my_audio.wav，有两次的峰值都超过了 0.5，所以有两次 chime：



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. 这里我用循环的上界 min(Ty,segment_end_y+51)省去了原代码框架中每次循环都要判定是否越界：

```
### START CODE HERE ### (≈ 3 lines)
for i in range(segment_end_y+1,min(Ty,segment_end_y+51)):
    # if None < None:
        y[0, i] = 1
### END CODE HERE ###
```

2. 首次加载模型 tr_model.h5 的时候，报了这样的错：‘str’ object has no attribute ‘decode’，因为它是 h5 文件，所以我检查了一下 pip list，原来自己没有 h5py，安装了 2.10.0 的 h5py 之后可以运行了。