Keyword wake-up

Keyword wake-up
- Introduction to the results of routine experiments
- Code Analysis

Introduction to the results of routine experiments

In this section, we will introduce the voice wake-up function on K230

The example code of this section is located in: [Source code/10.Media/05.keyword_spotting.py]

We use CanMV IDE to open the sample code and connect K230 to the computer via USB.

Click the Run button in the lower left corner of CanMV IDE

At this point, you can see the serial terminal continuously outputting Deactivated!

```
Deactivated!
total took 73.87 ms
Deactivated!
total took 298.68 ms
Deactivated!
total took 299.70 ms
Deactivated!
total took 299.86 ms
Deactivated!
total took 299.87 ms
Deactivated!
total took 299.83 ms
Deactivated!
total took 299.98 ms
Deactivated!
total took 299.88 ms
```

When we shout [Xiaonan Xiaonan] to K230, the serial terminal will output ====Detected XiaonanXiaonan!====, indicating that the wake-up word is detected.

```
total took 299.98 ms
Deactivated!
total took 299.88 ms
Deactivated!
total took 299.94 ms
====Detected XiaonanXiaonan!====
total took 1307.87 ms
Deactivated!
total took 22.50 ms
Deactivated!
```

> This voice wake-up model is provided by Canaan Technology

Code Analysis

[For complete code & bilingual comments, please refer to [Source Code/10.Media/05.keyword_spotting.py]]

Let's analyze the structure of this voice wake-up (KWS) code:

1. Class Design Architecture

- Inheritance structure: `KWSApp` inherited from `AIBase`
- Core methods:

```
class KWSApp(AIBase):
    def __init__(self, kmodel_path, threshold, debug_mode=0)
    def preprocess(self, pcm_data)
    def postprocess(self, results)
```

2. Audio preprocessing process

```
def preprocess(self, pcm_data):
    # PCM数据转换为float列表
    for i in range(0, len(pcm_data), 2):
        int_pcm_data = struct.unpack("<h", pcm_data[i:i+2])[0]  # 使用小端格式解析16位有符号整数 Parse 16-bit signed integer using little-endian format
        float_pcm_data = float(int_pcm_data)                    # 转换为浮点数 Convert to floating point number
        pcm_data_list.append(float_pcm_data)                    # 添加到列表中 Add to the list

    # Feature extraction and tensor conversion
    # 特征提取和张量转换
    mp_feats = aidemo.kws_preprocess(fp, pcm_data_list)[0]      # 调用aidemo模块的预处理函数提取音频特征 all the preprocessing function of the aidemo module to extract audio features
    mp_feats_np = np.array(mp_feats).reshape((1, 30, 40))       # 重塑数组形状为模型所需的输入维度 Reshape the array to the input dimension required by the model
```

3. Post-processing logic

```
def postprocess(self, results):
    logits_np = results[0]                                          # 获取第一个输出结果（预测概率） Get the first output result (predicted probability)
    self.cache_np = results[1]                                      # 更新缓存数组，用于下一次推理 Update the cache array for the next inference
    max_logits = np.max(logits_np, axis=1)[0]                       # 获取每个样本的最大概率值 Get the maximum probability value of each sample
    #Threshold determination and result output
    # 阈值判断和结果输出
    if max_p > self.threshold and idx == 1:
        return 1
```

4. Audio Stream Configuration

```
input_stream = p.open(format=FORMAT,                                # 打开输入音频流，配置格式、通道数、采样率等参数 Open the input audio stream and configure parameters such as format, number of channels, sampling rate, etc.
                      channels=CHANNELS,
                      rate=SAMPLE_RATE,
                      input=True,
                      frames_per_buffer=CHUNK)
```

5. Main loop structure

```
while True:
    os.exitpoint()                                                  # 检查是否有退出信号 Check if there is an exit signal
    pcm_data = input_stream.read()                                  # 读取音频数据 Read audio data
    res = kws.run(pcm_data)                                         # 运行关键词检测 Run keyword detection
    if res:
        # 播放回复音频 Play reply audio
        wf = wave.open(reply_wav_file, "rb")                        # 打开回复音频文件 Open the reply audio
```