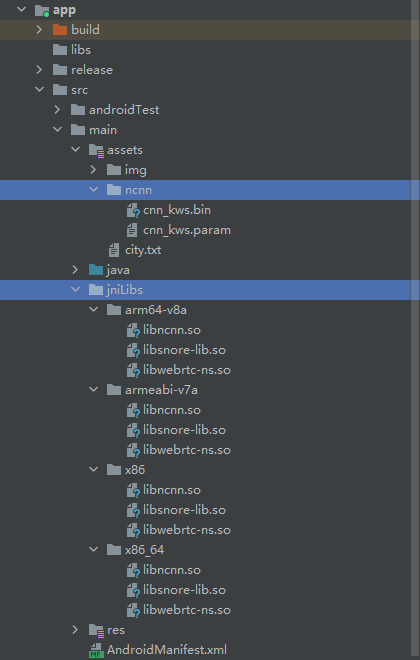
1. 在/src/main目录下创建jniLibs文件夹，并将文档中jniLibs目录下的so录复制进去。
2. 创建assets目录，并将文档中的/ncnn目录复制进去。



1. 新建包名org/snore，并复制文档中../org/snore目录下的Result.java和SnoreLib.java两个文件复制到新建的包中。
2. 初始化算法：

private void checkInit() {

if (SnoreRecognizer.getInstance().isInited()) {

return;

}

cachedThreadPool = Executors.newFixedThreadPool(4);

snoreThreadPool = Executors.newSingleThreadExecutor();

AppConfig config = new AppConfig(context);

int version = config.getKwsVersion();

File ncnn = context.getDir("ncnn", Context.MODE\_PRIVATE);

File bin = new File(ncnn, "cnn\_kws.bin");

File param = new File(ncnn, "cnn\_kws.param");

if (version < KWS\_VERSION || !bin.exists() || !param.exists()) {

try {

copy("ncnn/cnn\_kws.bin", bin);

copy("ncnn/cnn\_kws.param", param);

config.editor().setKwsVersion(KWS\_VERSION).apply();

} catch (IOException e) {

e.printStackTrace();

}

}

try {

SnoreRecognizer.getInstance().init(param.getPath(), bin.getPath(), TYPE\_NUMBER);

} catch (Exception e) {

e.printStackTrace();

｝

1. 音频识别

private final int AUDIO\_SIZE = 112000;//输入数据大小：3.5秒（16000 \* 2 \* 3.5f）

byte[] buffer = new byte[AUDIO\_SIZE \* 2];

int pos = 0;

public void parse(byte[] audioData) {

System.arraycopy(audioData, 0, buffer, pos, audioData.length);//缓存数据

pos += audioData.length;//累计数据长度

if (pos > AUDIO\_SIZE) {//缓存超过3.5秒的数据

byte[] data = new byte[AUDIO\_SIZE];//创建3.5秒的字节数组

System.arraycopy(buffer, 0, data, 0, AUDIO\_SIZE);//从缓存里取出3.5秒的数据

resetBuffer();//整理缓存

int result = SnoreRecognizer.getInstance().process(data);//识别这段音频

handlerResult(result);//处理识别结果

}

}