   float\_net, fixed\_net = quant\_model\_parameter(checkoutpoint1, config["feature"], checkpoint\_quant\_file,param\_Q\_file)

第一步模型参数量化。

输入checkoutpoint1为训练好的模型.pth,config为特征维数。

输出为量化好的模型参数，保存到文件checkpoint\_quant\_file里面，param\_Q\_file文件为每一层的Q值变化。

16 15 ----------conv0.weight的q值

16 15 ----------conv0.bias的q值

16 15 ----------conv1.weight的q值

16 15 ----------conv2.bias的q值

8 6 -------------rnn.weight\_ih\_l0和rnn.weight\_hh\_l0的q值

16 15 -----------rnn.bias\_ih\_l0和rnn.bias\_hh\_l0的q值

8 6 -----------rnn.weigh\_ih\_l0\_reverse和rnn.weight\_hh\_l0\_reverse的q值

16 15 ----------rnn.bias\_ih\_l0\_reverse和rnn.bias\_hh\_l0\_reverse的q值

8 6 ----------fc.weight的q值

16 15 --------fc.bias的q值

param\_Q\_file文件一共有十行两列。第一列为基础Q值，第二列则是根据实际的模型参数选出的真正的Q值。

Q值的计算原理：a.统计每个参数在基准值范围内属于哪个Q值跟基准Q值的比例范围，

计算公式：-2\*\*（Q基准值-1）/（2\*\*Q确定值）< 参数<2\*\*（Q基准值-1）/（2\*\*Q确定值）

每个参数会得到一个确定的Q值。

b。计算出该组参数中每个参数对应的Q值，统计该组参数的每个确定Q值数量,以此从高到底进行累加，大于参数总 数的95%的情况下，选择该Q值为这组参数的唯一确定Q值。

确定Q值之后，则将Q值应用到每个参数中，计算出模型中的每个参数的量化结果，保存到checkpoint\_quant\_flle中。

verify\_quant\_param(float\_net, fixed\_net, dev\_feature,dev\_label,target\_class)

第二步验证。

此函数用于模型量化前后的准确性对比测试

same as float ratio: 98.818970

total target num: 2

{'UNKNOWN\_WORD': 0, 'HeyMemo': 1}

target num: 0 length: 0

n\_wakeup: 0, n\_false\_wakeup 0

accurate\_rates: nan, false\_rate: nan

target num: 1 length: 21168

n\_wakeup: 16431, n\_false\_wakeup 4737

accurate\_rates: 0.776219, false\_rate: 0.223781

第一行的百分比为，浮点和定点的不同预测结果是否一致，基于总数的准确度计算。

后面的打印，则为pkl测试集中每个target具体预测结果的准确度计算，

此过程不影响量化结果。

 B\_list = crnn\_data\_flow\_quant(float\_net, pickle\_data[0], model\_config)

第三步pickledata的量化处理。

此步骤是将每一层的浮点结果做量化处理。

因此conv0 out, conv1 out, gru out, fc out, input

[[16, 12], [16, 14], [16, 15], [16, 11], [16, 13]]

如下的打印结果根据实际的Pkl文件的内容会有所区别。

Q值的计算与第一步略微有所区别。

此Q值的计算是统计不在基准Q值范围内的数据，抛除掉之后剩下正常范围内的值占总数的比例大于98%，则取该Q值为该组PKL数据的唯一确定值。

    prediction = run\_net(signal, checkpoint\_path, param\_Q\_file, model\_config, feature\_config, checkpoint\_quant\_file, 0)

第四步run\_net函数为验证和kws\_weight.c生成的最后步骤。

此步骤也是将前两部分的量化值融合到一起，应用于音频数据流和生成.weight.c文件。

fc\_out的输出Q值是根据最后一层FC的权重Q值加上数据流的卷积层的权重Q值相加得到的结果。

此过程也不影响量化结果。

无论信号长短多少，都回去1.6s的数据作为一次结果的输出，不够在后面补零，多了舍弃。