Training&Quantization notes

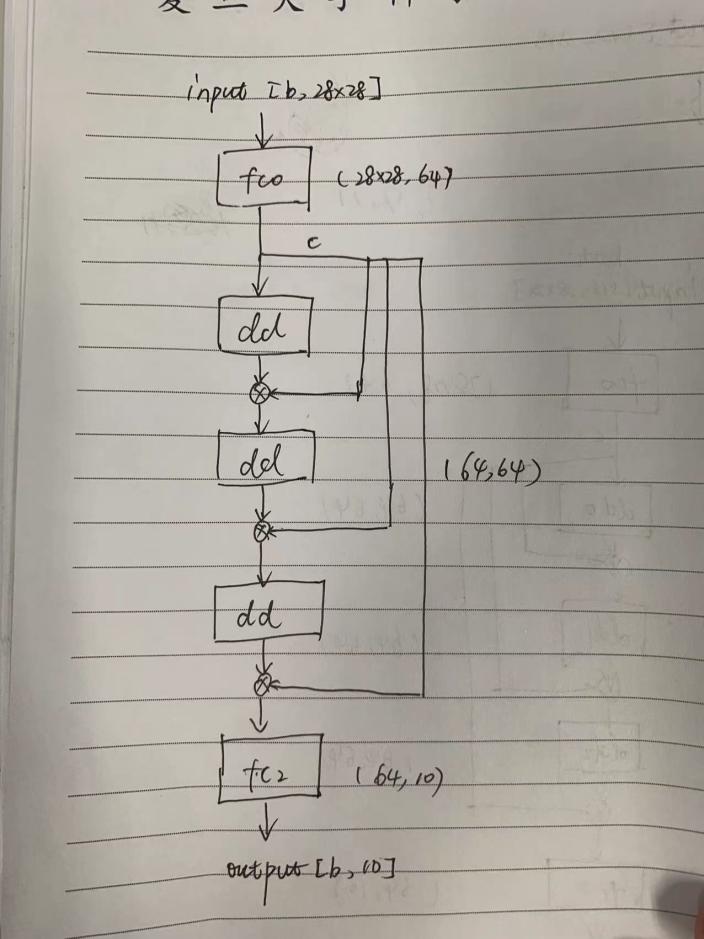
模型

模型采用树突网络DDNet（Dendrite Net），参考资料：

<https://blog.csdn.net/xiaoxiao_ziteng/article/details/111410269>

<https://www.bilibili.com/video/BV1Dp4y1a7Bk?from=search&seid=2721252407753649106&spm_id_from=333.337.0.0>

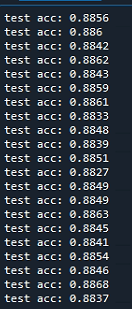
采用DD2，结构如下图：



有三层FC： fc0, dd, fc2, 第一层fc0的输出被暂存为c，每次dd计算过后，输出与c相乘作为下一级输入，最后输出10位output作为分类结果。

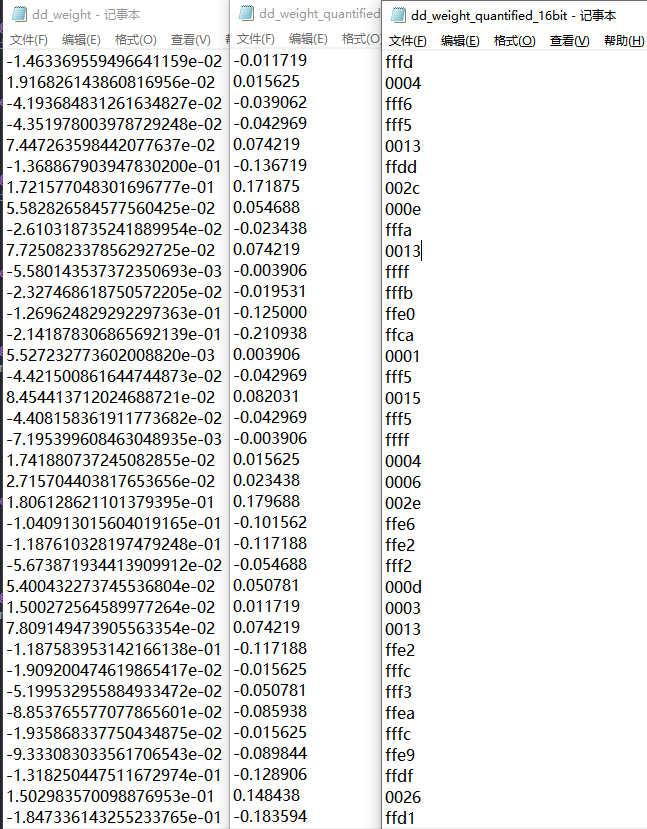
网络亮点：没有bias，没有激活函数，只需要实现pe的fc运算和一个ppe乘法器。参数量54912，比较少。

分类效果：88%+的准确率，只训练了300个epoch，并没有完全收敛，后续可以跑长一点。



量化

Input以及weight数据被量化为fixpoint(1,7,8)，即1位符号位，7位整数位，8位小数位，最小精度为0.00390625，并存为16进制和10进制，如下图：



从左到右依次为量化前的float数据，量化后的fixpoint，以及存为16bit hex的量化后的fixpoint，发现存在量化误差，不确定是否影响最终结果，考虑改为(1,4,11)，精度可达到4.8828\*10e-4

Why C model

C model旨在实现基于量化后参数的forward运算，并提供与硬件对应的中间结果，作为硬件运算结果的reference。Pytorch对量化训练的支持性不好，如果用32/64bit的未量化的数据直接传给硬件，会造成数倍不必要的开销，所以硬件上是采用量化后的16bit fixpoint格式的数据进行计算的，因此需要c model，实现基于16bit量化后数据的神经网络forward运算，作为硬件模型的reference。

硬件需要解决的问题（部分）：

1. 运算FC的PE
2. DDR文件的排列，WEIGHT和INPUT需要排列到DDR文件（.txt），硬件通过读取ddr文件来获取数据（与软硬件协同一起解决），算完后写回到ddr给pc。
3. DDR2MEM，需要按照DDR的排列顺序，从ddr里对应的地址拿取对应长度的数据，放到对应的MEM里
4. 乘法运算模块mul（c\*dd\_out）
5. 顶层的状态机，来控制pe和mul，从mem里取数，运算，再送到输出的mem，等等。

软硬件协同需要解决的问题：

1. 板子与PC端数据的传输，运算结果的display，整个流程走通，如何实现real time
2. 是否考虑通过指令集来控制硬件的模块，（需要和硬件的同学讨论，评估一下工作量，这个方案仅供参考）