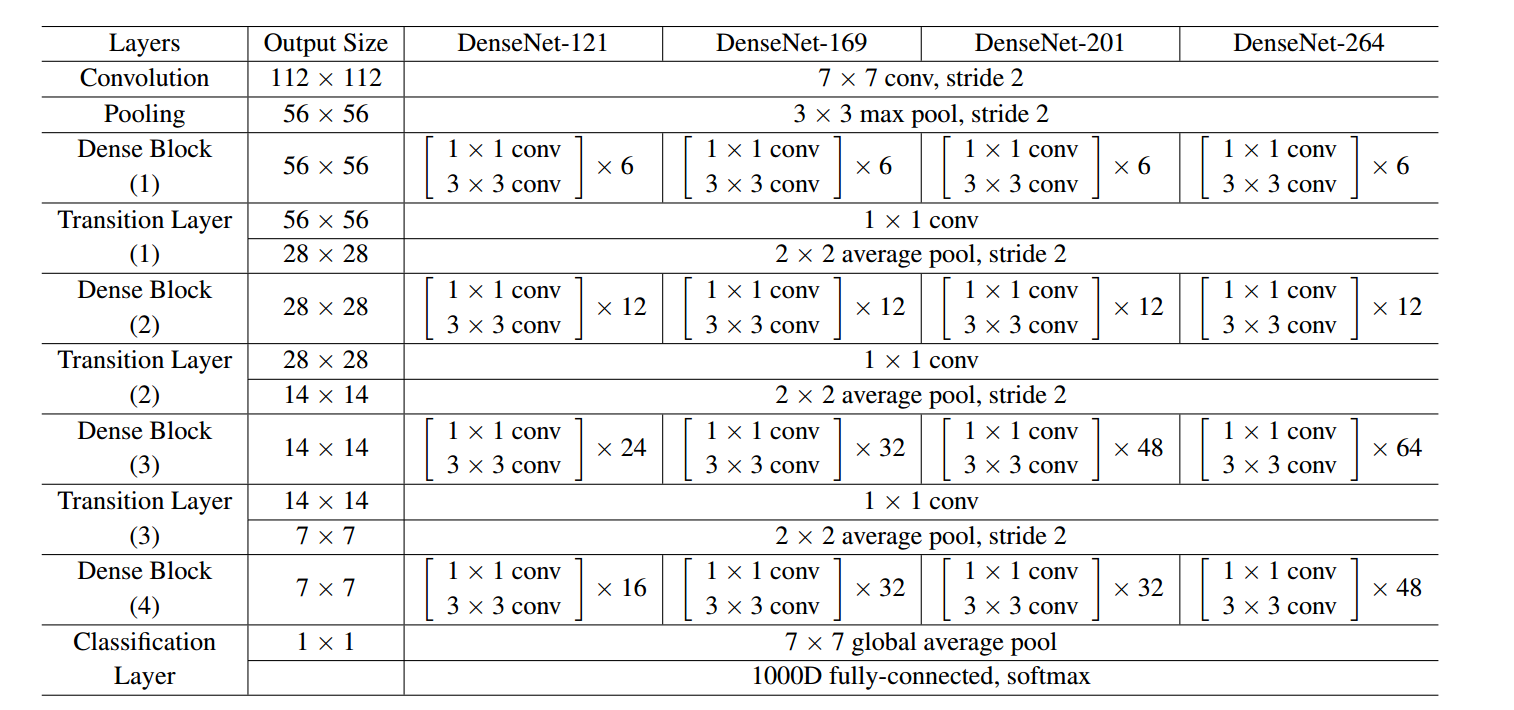
CSDN第八周densenet作业文档

关于densenet的实现

基本上参照论文中的实现过程



请详见代码

关于growth的思考：

试验结果，growth越大，训练的时间越长，基本比例为线性n增长。

以下结果为tinymind上gpu运算的结果，Layer = 12

Gwoth为12的时候，

INFO:tensorflow:global step 3550: loss = 4.6672 (0.494 sec/step)

INFO:tensorflow:global step 3560: loss = 4.9283 (0.476 sec/step)

INFO:tensorflow:global step 3570: loss = 4.7057 (0.496 sec/step)

INFO:tensorflow:global step 4730: loss = 4.6554 (0.497 sec/step)

INFO:tensorflow:global step 4740: loss = 4.5184 (0.481 sec/step)

INFO:tensorflow:global step 5170: loss = 4.4322 (0.472 sec/step)

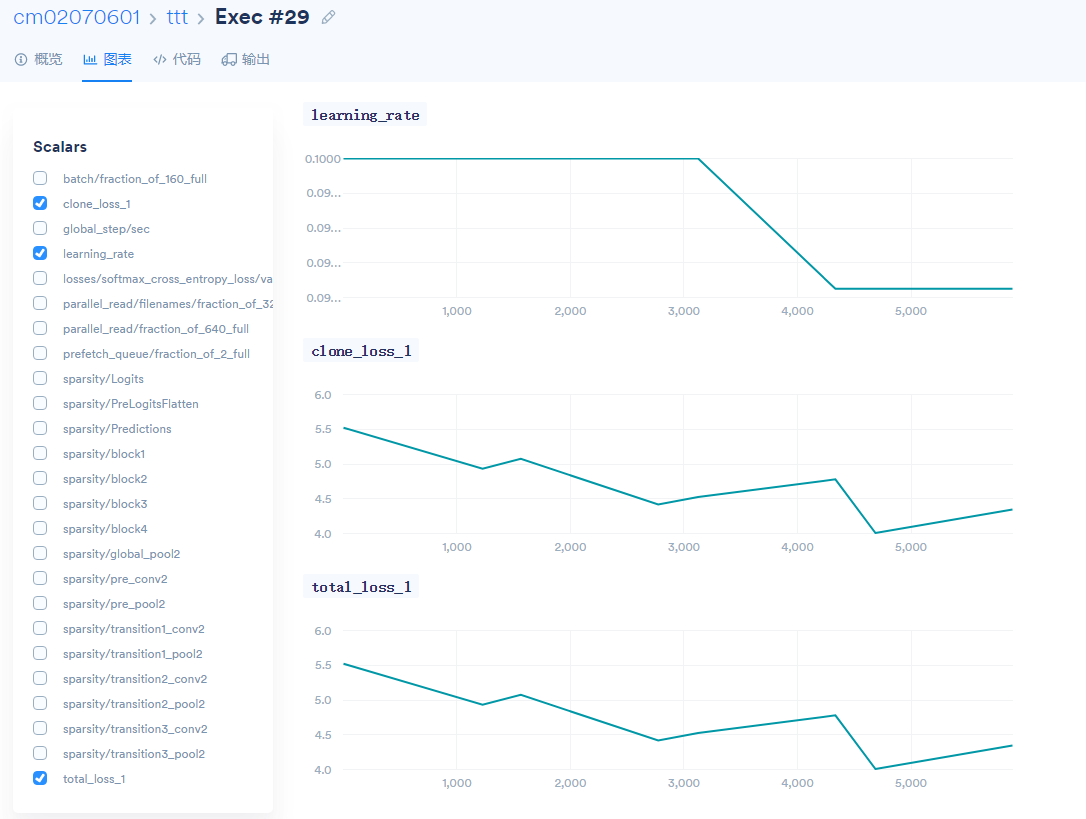
INFO:tensorflow:global step 6330: loss = 4.7005 (0.489 sec/step)

INFO:tensorflow:global step 6340: loss = 4.1280 (0.492 sec/step)

INFO:tensorflow:global step 6350: loss = 4.5695 (0.498 sec/step)

eval/Accuracy[0.0600585938]

eval/Recall\_5[0.194091797]

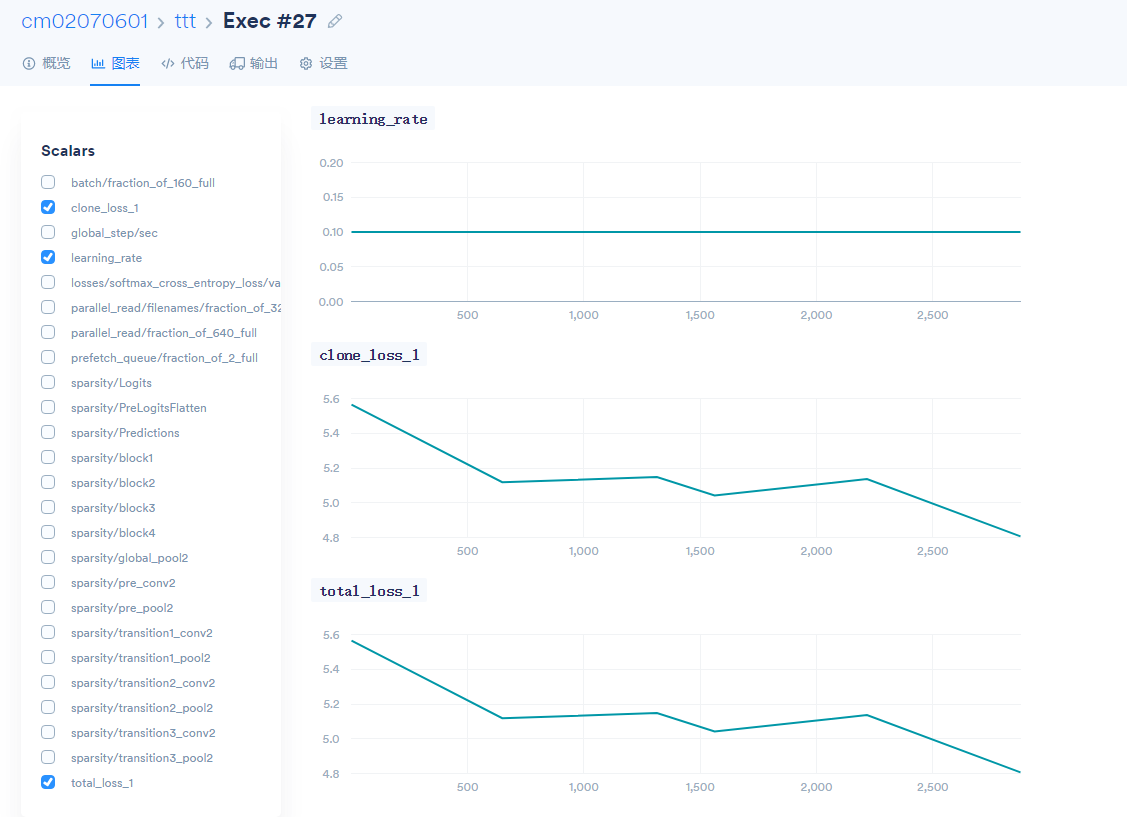


Gwoth为24的时候，时间基本翻倍了，基于时间就是金钱的事实，做出一个大致的推论。

INFO:tensorflow:global step 3570: loss = 4.7716 (0.915 sec/step)

INFO:tensorflow:global step 3580: loss = 4.8238 (0.908 sec/step)

INFO:tensorflow:global step 3590: loss = 4.4735 (0.904 sec/step)



下面为growth =32时的运算结果

INFO:tensorflow:global step 2710: loss = 5.1972 (1.201 sec/step)

INFO:tensorflow:global step 2720: loss = 4.9001 (1.189 sec/step)

INFO:tensorflow:global step 2730: loss = 5.1091 (1.191 sec/step)

INFO:tensorflow:global step 4650: loss = 4.7004 (1.185 sec/step)

INFO:tensorflow:global step 4660: loss = 4.6911 (1.198 sec/step)

INFO:tensorflow:global step 4670: loss = 4.7573 (1.198 sec/step)

INFO:tensorflow:global step 4930: loss = 4.4841 (1.188 sec/step)

INFO:tensorflow:global step 4940: loss = 4.8129 (1.195 sec/step)

INFO:tensorflow:global step 4950: loss = 4.6191 (1.187 sec/step)

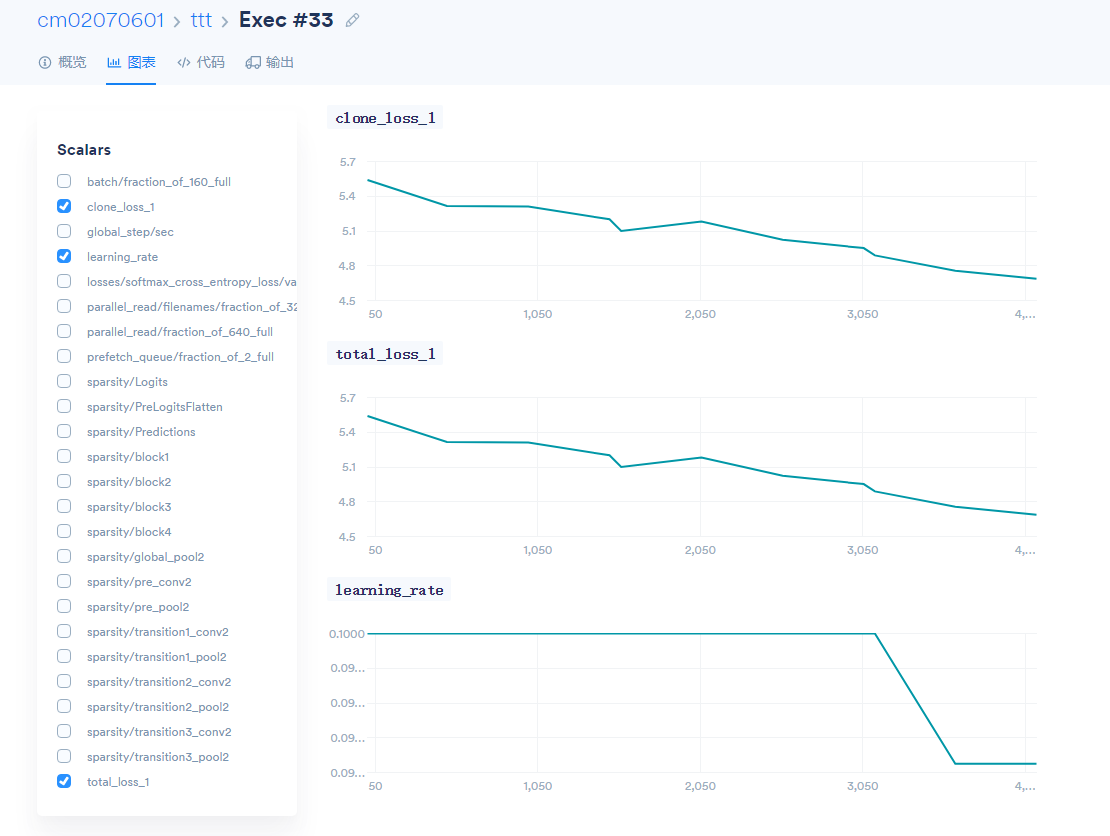
eval/Recall\_5[0.107910156]

eval/Accuracy[0.0329589844]

可以看出随着growth的增加，loss下降速度并没有快很多，然而因为参数量的增加，运算时间是多了很多。

由于经费问题，没有花很长的时间去获得最终稳定的收敛结果，所以只从收敛速度上观察。

从论文上看，增加growth是可以得到更好的识别结果的。



尝试过增加layer的数量，但是tinymind似乎跑不了，报的错误类似占用闪存较大。

Growth为论文中的主要两个参数，一个是layer，一个是k（growthrate）。

If each function H` produces k featuremaps, it follows that the l` th layer has k0 +k ×(l^th−1) input feature-maps, where k0 is the number of channels in the input layer.

个人理解如下：

Layer是densenet没个block向前获取feature时的layer深度。

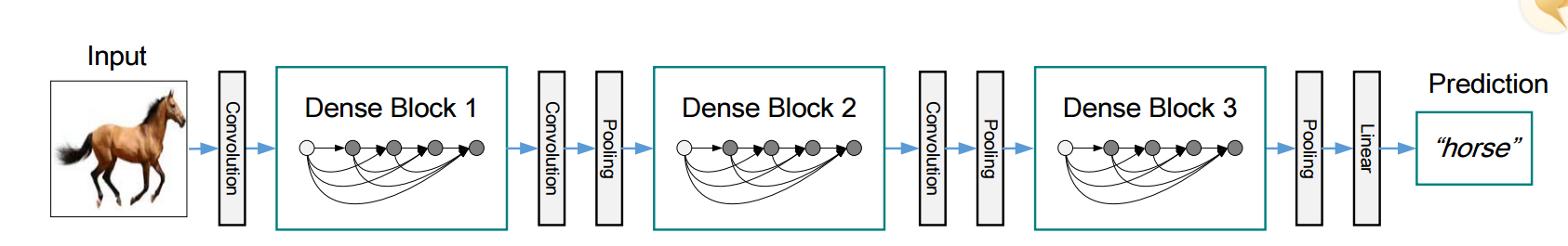
k是输出的chanel数量，随着layer的增加，输出的feature也随之增加。

Densenet整体参数的数量会被这两个超参数限制约束。从论文结果看，densenet的参数数量比resnet或densenet前的一些神经网络有明显的下降，但是却获得了较好的结果。

从个人的实验结果看，运算时间也和growthRate成正比增加。基本上是参数量的增加导致的运算时间的增加。

关于稠密的解释：

稠密这个词主要来源于densenet的“dense”，即稠密的意思。之前的神经网络连接上比较相近（远近的近）。即每一层的输入只来源于上一层，输出只影响下一层。但是densenet由于网络的设计，在一个block内部，做到了每一层都影响前面若干层，同样也影响后面若干层。如下图：



这样层和层之前的链接也更加的紧密，需要的参数量也大范围的下降，

巧妙的地方还在于用transitionLayer进行feature数量的控制，经过一个block后，随着growthRate的增加（feature的输出也增加），用一个transitionLayer去对输出的feature进行瓶颈式的收敛。

可以说dense（稠密）让feature之间的联系更加紧密，但是却没有通过“增加参数量”的方式实现。

后续思考：

1. 有文章说densenet缓解了梯度消失的现象，但是由于实验的时间限制，并没有观测到结果，也许是缓解了吧……
2. Densenet主要应用于图片识别，或者说卷积的应用大致在图像这个层面，对于声音及文字的识别，或者人类其他感知源的识别，densenet的方式有没有在其他方式的模式识别中应用呢，期待后面的学习。比如之前学到的机器学习，也采用densenet的方式进行优化？