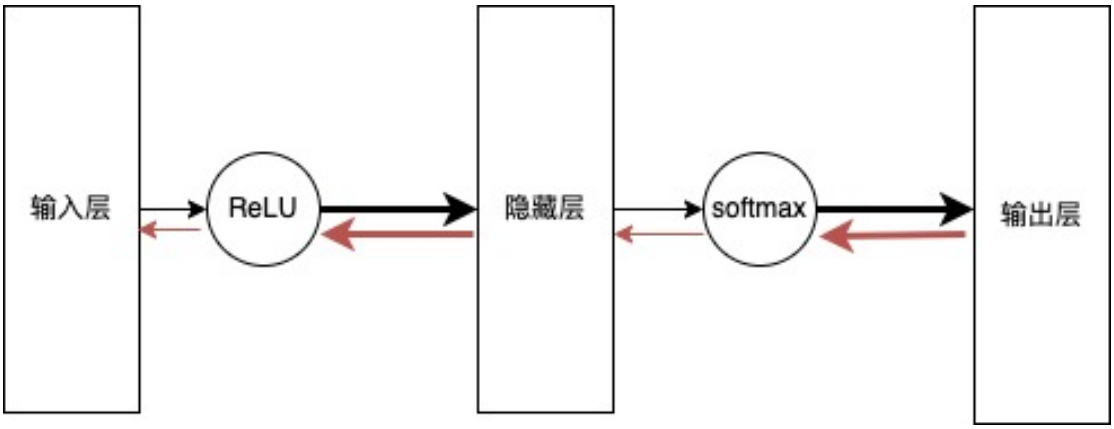
计算机图像处理大作业实验报告

报告人： 何峙
学号： 21215122
专业： 大数据与人工智能
实验步骤：

1. 全连接神经网络

本实验主要实现一个两层的全连接神经网络，基本架构如下所示：



其中黑色箭头为正向传播，红色箭头为反向传播（线条粗细只是用来区分不同层的关系）。

- 正向传播即输入向量进行线性变换后接一个非线性变换的运算，本实验使用 ReLU 和 Softmax 函数作为非线性变换，过程如下：

(1) $Z_1 = W_1 \odot X + B_1$

(2) $A_1 = ReLU(Z_1)$

(3) $Z_2 = W_2 \odot A_1 + B_2$

(4) $\widetilde{Y} = Softmax(Z_2)$

(5) 计算损失函数，本实验使用交叉熵作为损失函数：

$$Loss(\widetilde{Y}, Y) = -\widetilde{Y} * logY - (1 - \widetilde{Y}) * log(1 - Y)$$

反复套用（1）、(2)、(3）步，可搭建更多层的全连接神经网络。

- 反向传播，计算 Loss 损失函数，Softmax 函数、ReLU 函数，还有线性函数 Z = W·X +B 各函数分别对 W，对 B 和对 X 的偏导数，其目的是根据下式更新权重 W 和 B：

$$W := W - lr * \frac{\partial F}{\partial W}$$

$$B := B - lr * \frac{\partial F}{\partial B}$$

其中 F 为各层对应使用的函数，lr 为学习率，本实验使用随机梯度下降方法（SGD）。

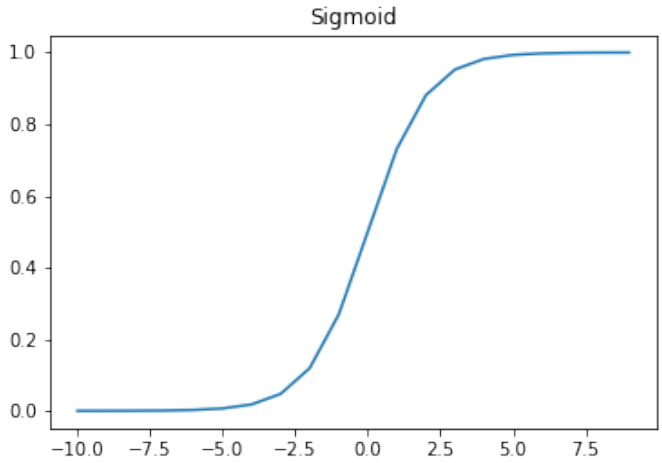
综上，不断的使用正向传播——反向传播——正向传播——反向传播……，迭代到一定的轮数 Loss 函数收敛到某个值，即可停止网络的训练，至此完成全连接神经网络权值的学习。

本实验还进行了三个小试验：

- 激活函数 Sigmoid、ReLU 和 Leaky ReLU 的讨论

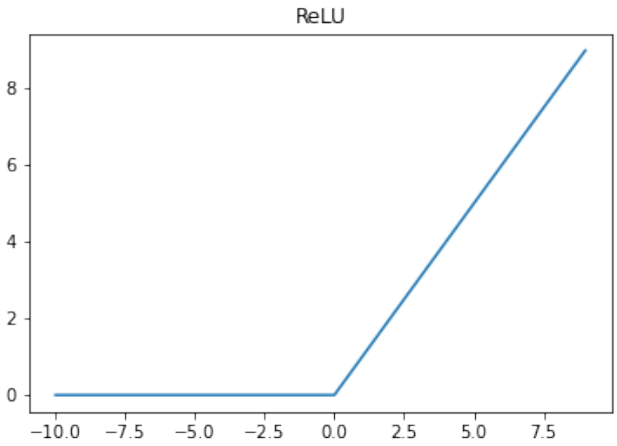
对比三个函数：

（1）Sigmoid



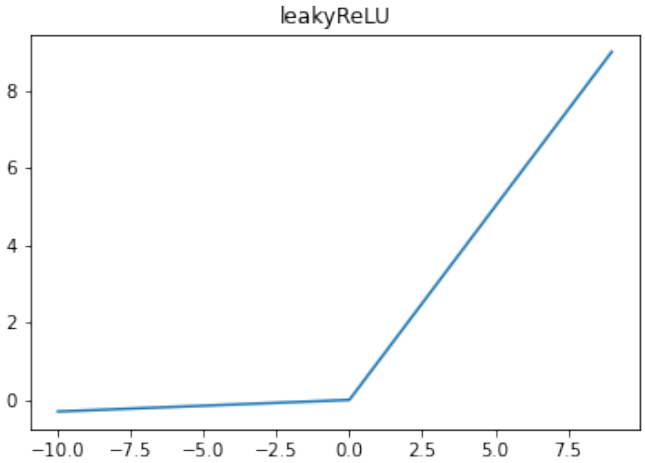
靠近 0 的值导数较大，趋于负无穷和正无穷时，导数越趋于 0。所以使用其作为激活函数时，在 0 附近，有较好的激活性，但在正负无穷区域容易发生梯度消失现象。

（2）ReLU



由其图像可知，大于 0 时，梯度为常数，不会出现梯度消失。但小于 0 时，梯度为 0，这是神经元不会被激活，即不会被训练。

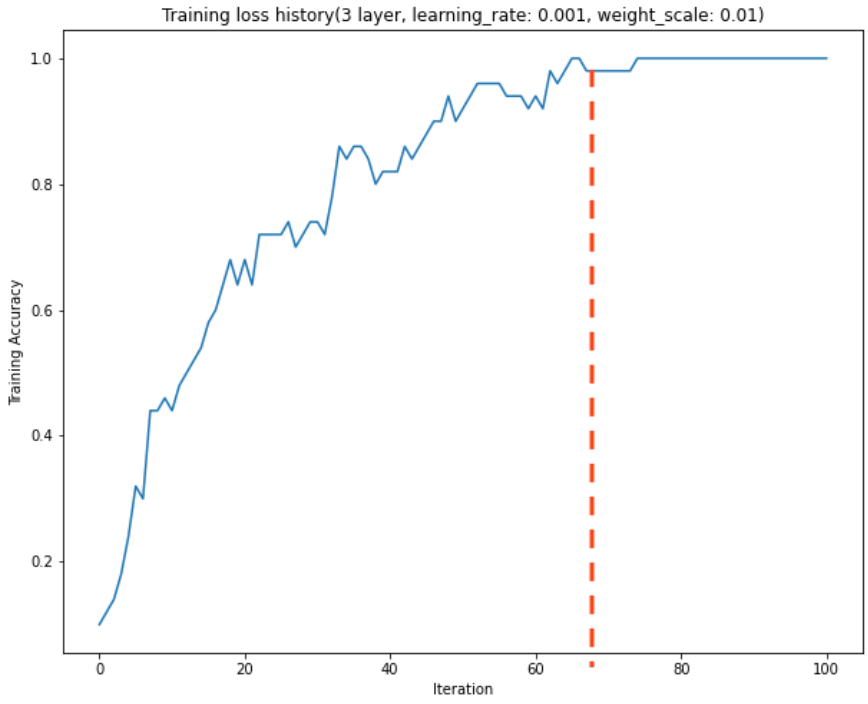
（3）LeakyReLU



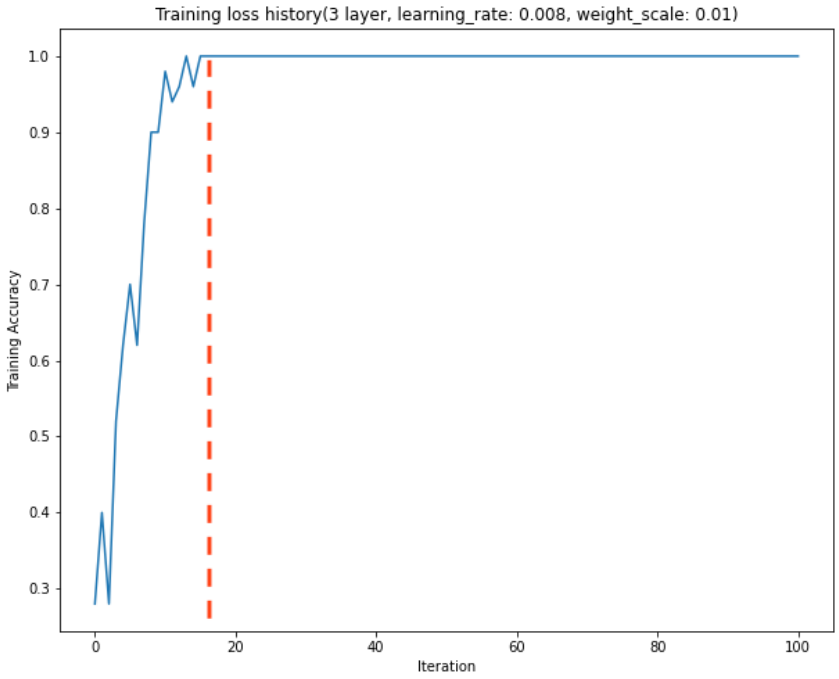
大于 0 的时候与 ReLU 类似，会保留一个比较小的负的值，使得此时梯度也不会消失。
本实验使用 LeakyReLU 也可以，只是效果不会太明显。

- 使用 3 层全连接网络，在 20 轮内用 50 个样本，使训练准确度达到 100%

最初使用的学习率为 1e-3，发现 20 轮里无法达到 100%的训练准确度，便尝试迭代 100 轮，结果如下：



发现大概在 70~72 轮，网络训练准确度才能达到 100%，猜测是迭代速度过慢所致，于是尝试将学习率增大为 8e-3，可达到要求：

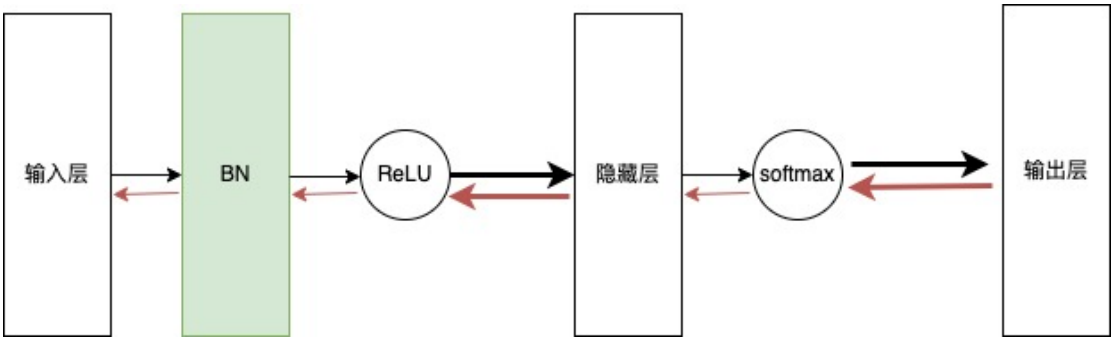


- 使用 5 层全连接网络，在 20 轮内用 50 个样本，使训练准确度达到 100%：

随着网络深度的增加，参数调整比之前难度增加。这里调节学习率后，发现影响不大，调节 weight_scale 发现对结果影响比较大，需要经过多次调整后才能达到 100%准确率。说明对于单纯增加网络深度，其参数和权值的调整会变得特别艰难。

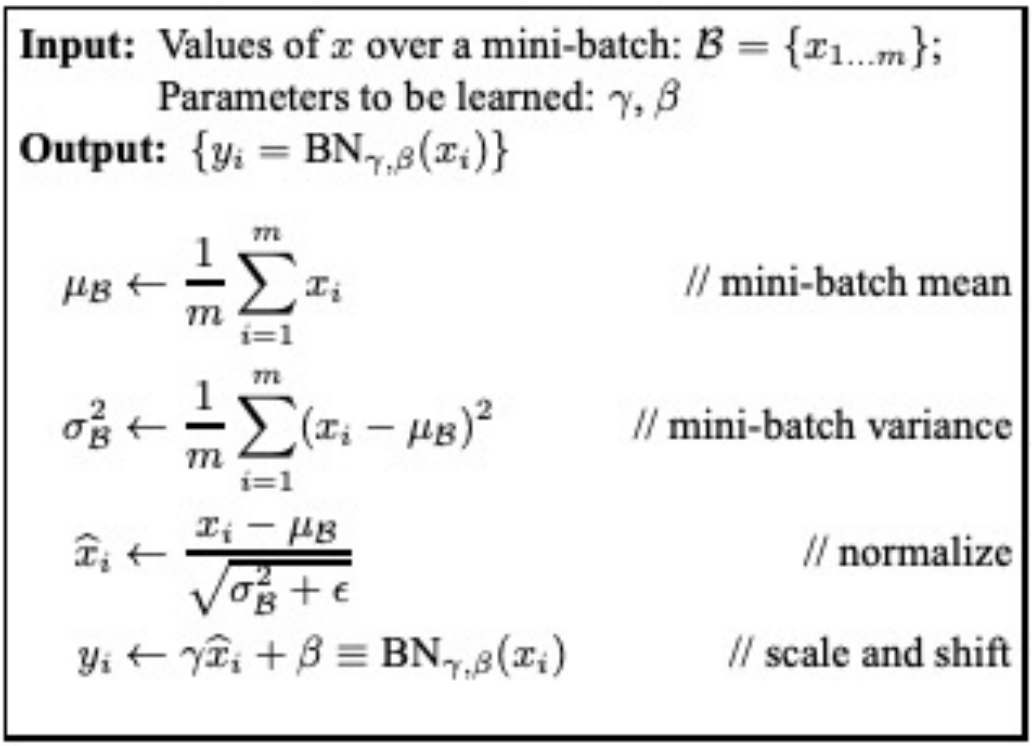
2. 归一化

当我们训练深度神经网络的时候，数据不断的通过网络层的处理也会使得其原始分布发生改变。更严重的是，随着权重得不断更新，每一层得输入特征的分布也会不断地发生漂移。归一化的引入使得数据经过网络层后继续保持均值为 0、方差为 1 的分布，本实验使用的 batch normalization 基本架构如下：



数据经过全连接层后，在流入激活函数之前，先进行 batch normalization（上图绿色部分）处理。在增加更多网络层时，可以套用这种模式，按需进行 batch normalization。

batch normalization 流程如下：



（图 1，来自“batch normalization”论文[1] 的算法说明）

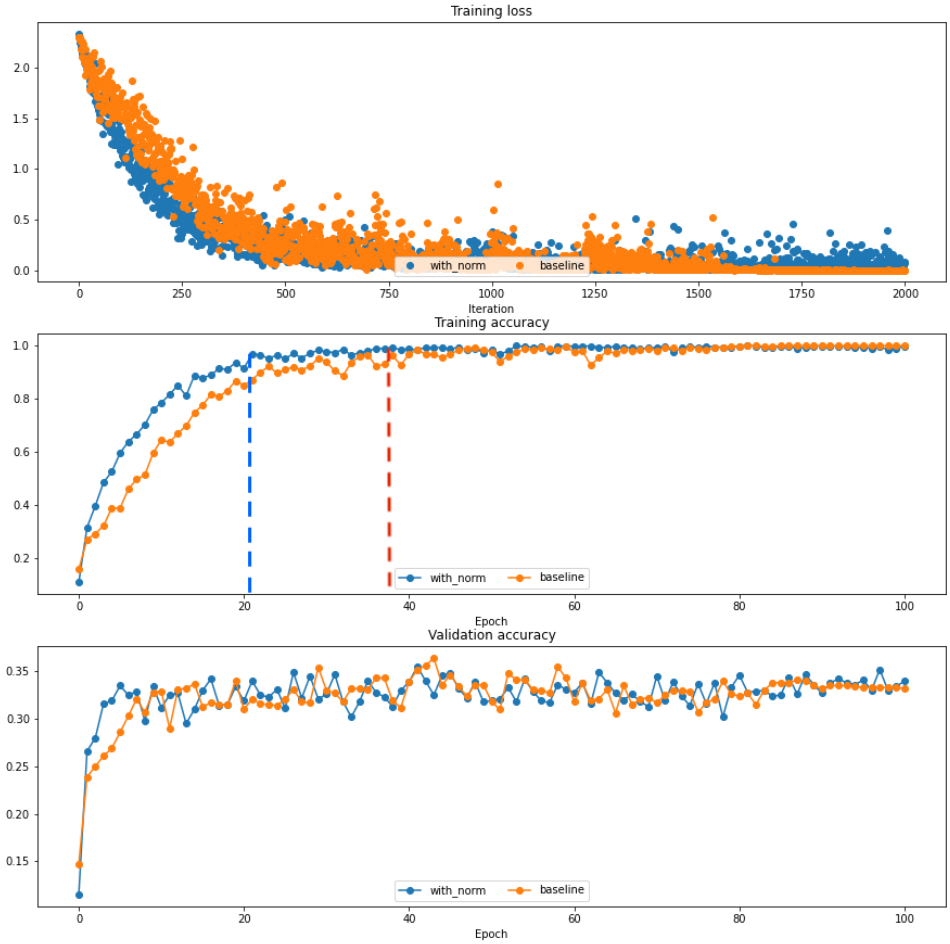
其中，参数 γ 和 β 一起参与训练。

这里进行了四个小实验：

- 使用 batch normalization 和不使用 batch normalization 的区别

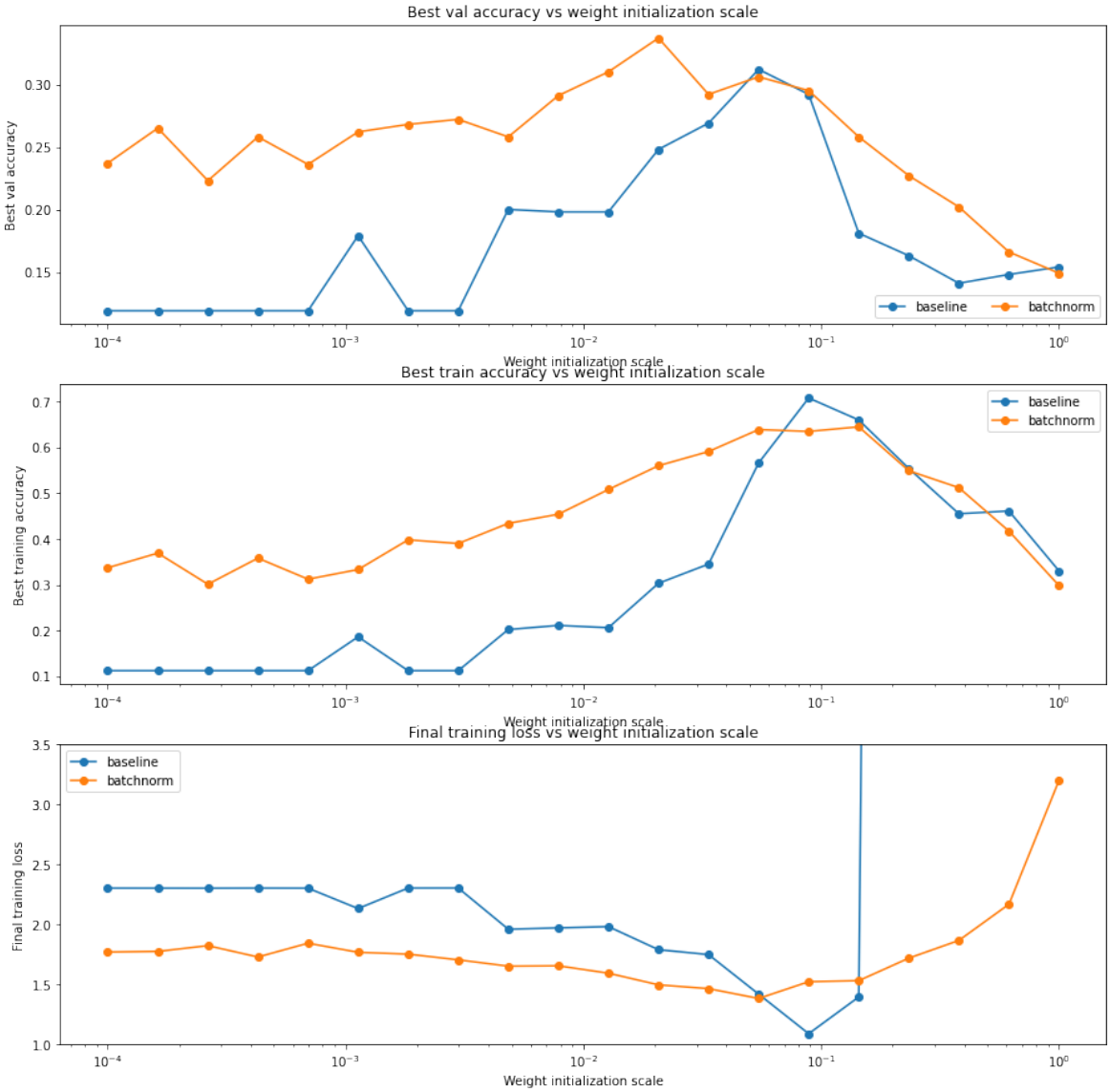
如下图所示，可见使用 batch normalization 时算法收敛的更快（垂直蓝线标记），收敛的也更稳定：

[1] 《Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift》https://arxiv.org/abs/1502.03167



- batch normalization 与初始化权重的联系

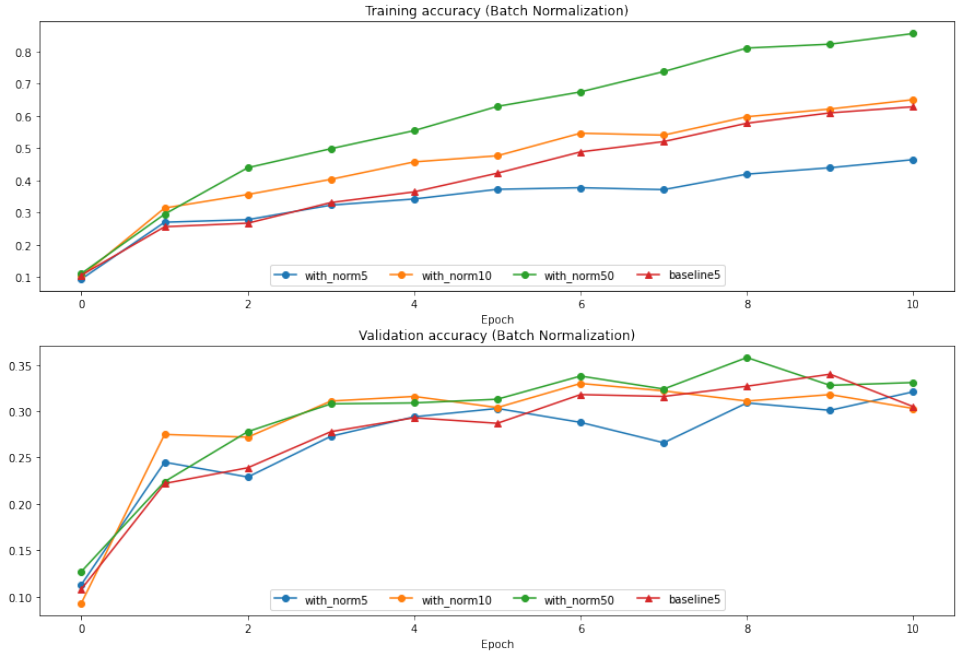
本实验定义一个 8 层的神经网络，然后分别比较不同的权重初始化参数下，带 batch normalization 和不带 batch normalization 时的网络的性能差异，结果如下图：



可见，batch normalization 使得网络的训练对网络参数初始化变得不那么敏感。而不带 batch normalization，对于权重初始化过小，则参数分布逐渐集中在 0 附近，导致回传的梯度乘以参数之后变得非常小。对于权重初始化过大，则参数分布逐渐两极化，出现梯度消失现象。

- batch normalization 与 batch size 的联系

初始化一个带 BN 层的 6 层神经网络，然后使用不同 batch size 大小的参数进行训练。对比如下图：

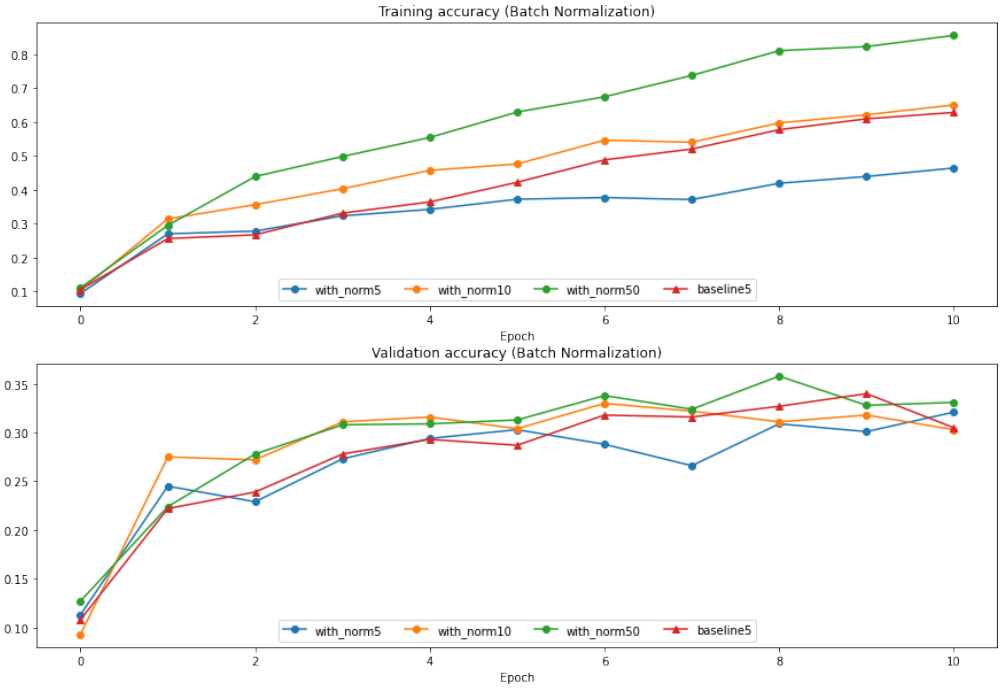


可见，随着 batch size 的增加，模型收敛的更快，说明 batch normalization 层适合大的 batch size，可能是大的批量使得样本 batch 的均值和方差估计得更准确。

- 使用 layer normalization

batch normalization 是对 batch 内样本的每个特征做归一化，而 layer normalization 是对一个样本中所有特征做归一化。

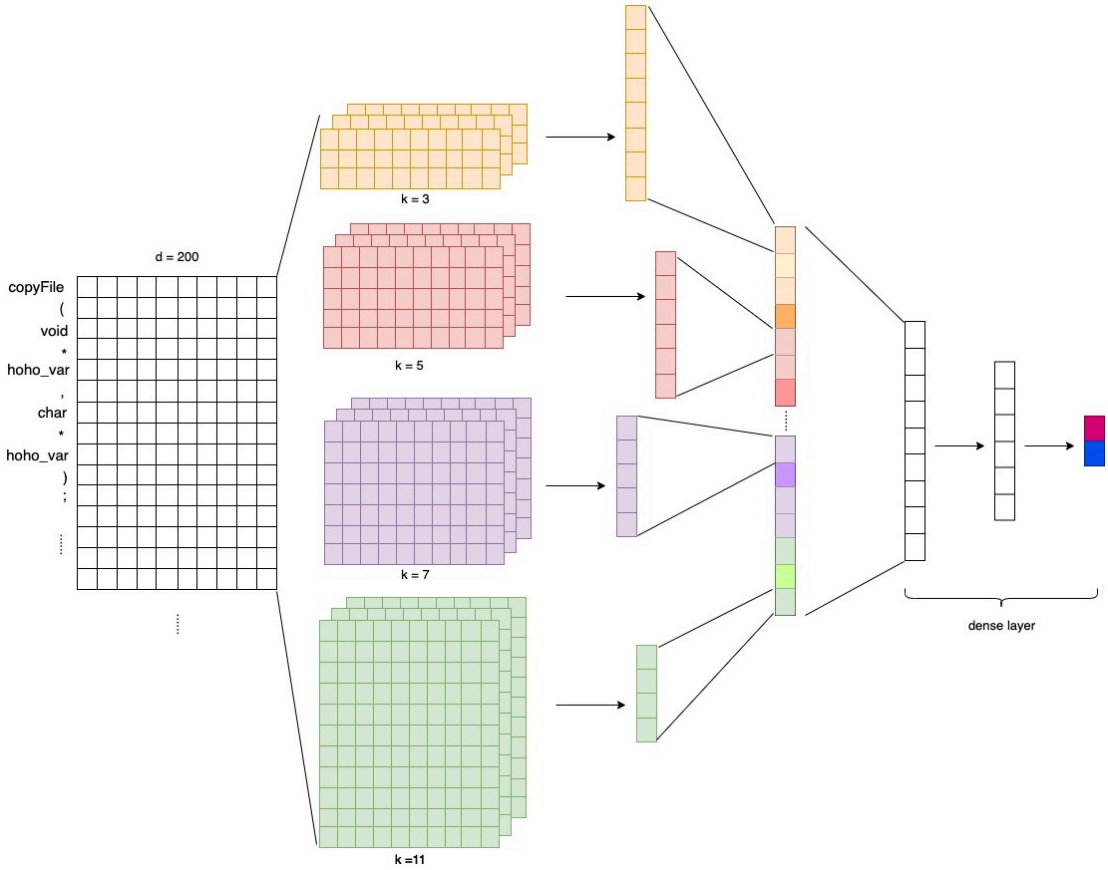
本实验主要比较不用 batch size 对使用 layer normalization 性能的影响，如下图：



可以发现，batch size 对 layer normalization 的影响比 batch normalization 影响小。

3. CNN

卷积神经网络基本架构为卷积层、池化层、全连接层等，如下图（这里现实了 4 种不同大小的卷积核）：



本实验搭建一个三层神经网络，关键是卷积计算和池化计算：

（1） 卷积计算即每个卷积核与图像每个像素的乘积的加和

（2） 池化计算是对卷积计算的结果进行取最大化或平均值的运算。

其余操作（正向运算，反向传播，等）跟实验 1 的全连接层类似，这里不再累述。

最后可视化卷积核，可见其可以提前图像的各类特征，如横竖轮廓、色彩饱和度等。