神经网络 PA_3

——实验报告

闫力敏*

1 概述

本次试验，通过调用 tensorflow 框架 MLP 和 CNN 的实现，测试 BN 层对试验的影响。然后在此基础上，进行了一定程度的探索。

2 实验过程

本次试验，为了更好的测试 BN，分别在 MLP 和 CNN 的模型中进行了 BN 层的对比试验。

3 实验任务

3.1 MLP without BN

网络结构：input + Linear + ReLU + Linear + loss

```
W = weight_variable([28*28, 100]);
b = bias_variable([100]);
out1 = tf.matmul(self.x_, W) + b;
# out2 = batch_normalization_layer(out1, is_train);
out3 = tf.nn.relu(out1);
W1 = weight_variable([100, 10]);
b1 = bias_variable([10]);
out4 = tf.matmul(out3, W1) + b1;
logits = out4;
```

3.1.1 参数说明

采用的参数如下

```
config = {
    'learning_rate': 0.001,
    'learning_rate_decay_factor': 0.9995,
    'batch_size': 100,
    'max_epoch': 20,
}
```

*清华大学计算机系. 学号：2015011391. 邮编：100084

3.1.2 训练结果

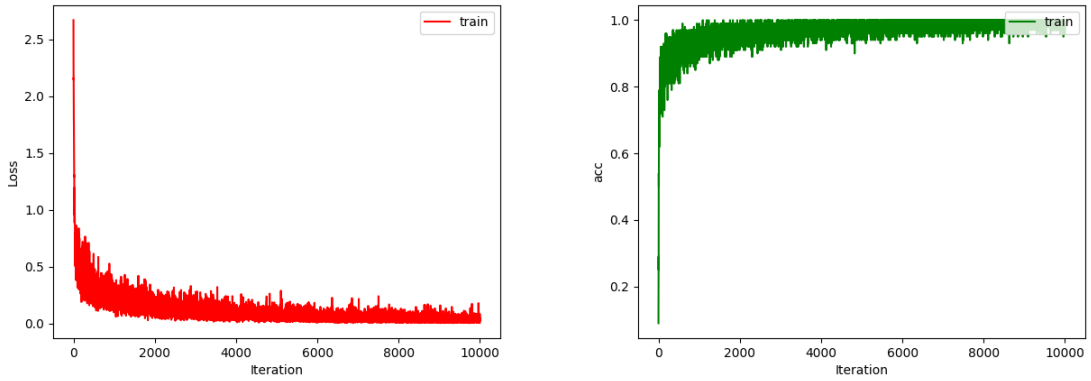

图 1: MLP without BN

测试的准确率：97.46%

3.1.3 小结

以上我们发现，不论 loss 还是 acc 的波动程度较为剧烈，因此我们引入 BN 层对比。

3.2 MLP with BN

网络结构：input + Linear + BN + ReLU + Linear + loss

```
W = weight_variable([28*28, 100]);
b = bias_variable([100]);
out1 = tf.matmul(self.x_, W) + b;
out2 = batch_normalization_layer(out1, is_train);
out3 = tf.nn.relu(out2);
W1 = weight_variable([100, 10]);
b1 = bias_variable([10]);
out4 = tf.matmul(out3, W1) + b1;
logits = out4;
```

3.2.1 参数说明

```
config = {
    'learning_rate': 0.001,
    'learning_rate_decay_factor': 0.9995,
    'batch_size': 100,
    'max_epoch': 20,
}
```

3.2.2 训练结果

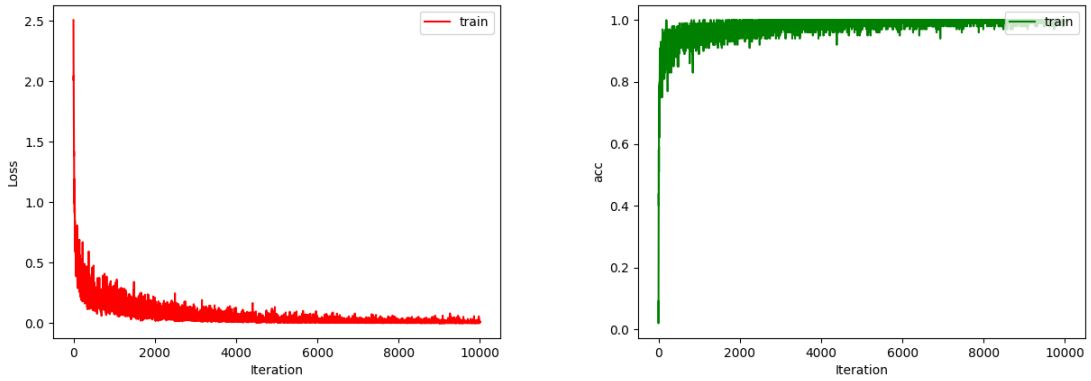

图 2: MLP with BN

测试的准确率：97.67%

3.2.3 小结

以上我们发现，loss 的波动程度明显减少，说明归一化处理确实是显著的提升了收敛的速率，归一化的一个好处是减少了某个图的权重过大的问题，使得每个图的地位尽可能的相等。于是我们在 CNN 模型下的测试，试验如下。

3.3 CNN without BN

网络结构：input + Conv + ReLU + MaxPool + Conv + ReLU + MaxPool + Linear + loss

为了增加 BN 对结果的影响上界，我们把参数设置的较为粗糙，我们首先采用的具体结构如下：

```
# conv1
W_conv1 = weight_variable([5,5, 1,4]);
b_conv1 = bias_variable([4]);
o_conv1 = tf.nn.relu((conv2d(x, W_conv1) + b_conv1)); # output size 28x28x8
o_pool1 = max_pool_2x2(o_conv1);

# conv2
W_conv2 = weight_variable([5,5, 4,8]);
b_conv2 = bias_variable([8]);
o_conv2 = tf.nn.relu((conv2d(o_pool1, W_conv2) + b_conv2)); # output size 14x14x16
o_pool2 = max_pool_2x2(o_conv2);
h_pool2_flat = tf.reshape(o_pool2, [-1, 7*7*8])
# liner
W1 = weight_variable([7*7*8, 10])
b1 = bias_variable([10])
out4 = tf.matmul(h_pool2_flat, W1) + b1;
logits = out4;
```

3.3.1 参数说明

```
config = {
    'learning_rate': 0.001,
```

```
    'learning_rate_decay_factor': 0.9995,
    'batch_size': 100,
    'max_epoch': 20,
}
```

3.3.2 训练结果

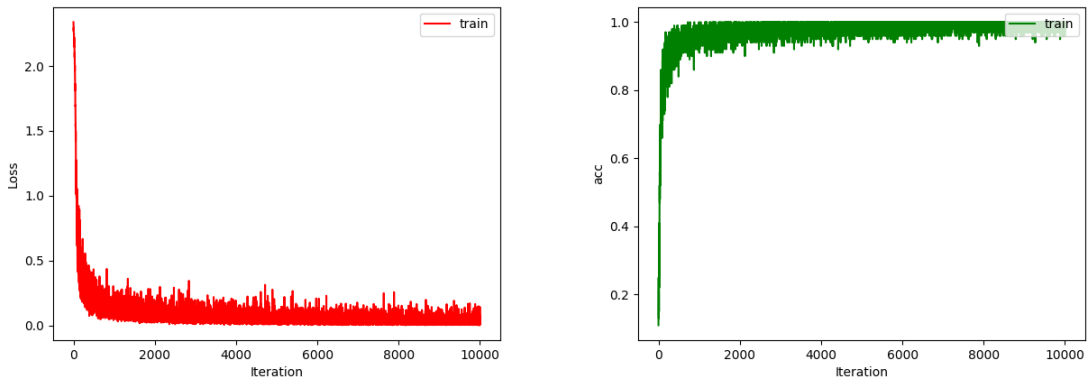

图 3: CNN without BN

测试的准确率：98.61%

3.3.3 小结

以上虽然在准确率上 CNN 网络确实较 MLP 具有优势，但是从图上分析可知数据的稳定性还是比较差的。在此基础上我们又增加了 BN 层。

3.4 CNN with BN

网络结构：input + Conv + BN + ReLU + MaxPool + Conv + BN + ReLU + MaxPool + Linear + loss

结构如下：

```
# conv1
W_conv1 = weight_variable([5,5, 1,4]);
b_conv1 = bias_variable([4]);
o_conv1 = tf.nn.relu(batch_normalization_layer(conv2d(x, W_conv1) + b_conv1)); # output size 28x28x8
o_pool1 = max_pool_2x2(o_conv1);

# conv2
W_conv2 = weight_variable([5,5, 4,8]);
b_conv2 = bias_variable([8]);
o_conv2 = tf.nn.relu(batch_normalization_layer(conv2d(o_pool1, W_conv2) + b_conv2)); # output size
                                                    14x14x16
o_pool2 = max_pool_2x2(o_conv2);
h_pool2_flat = tf.reshape(o_pool2, [-1, 7*7*8])
# liner
W1 = weight_variable([7*7*8, 10])
b1 = bias_variable([10])
```

```
out4 = tf.matmul(h_pool2_flat, W1) + b1;
logits = out4;
```

3.4.1 参数说明

```
config = {
    'learning_rate': 0.001,
    'learning_rate_decay_factor': 0.9995,
    'batch_size': 100,
    'max_epoch': 20,
}
```

3.4.2 训练结果

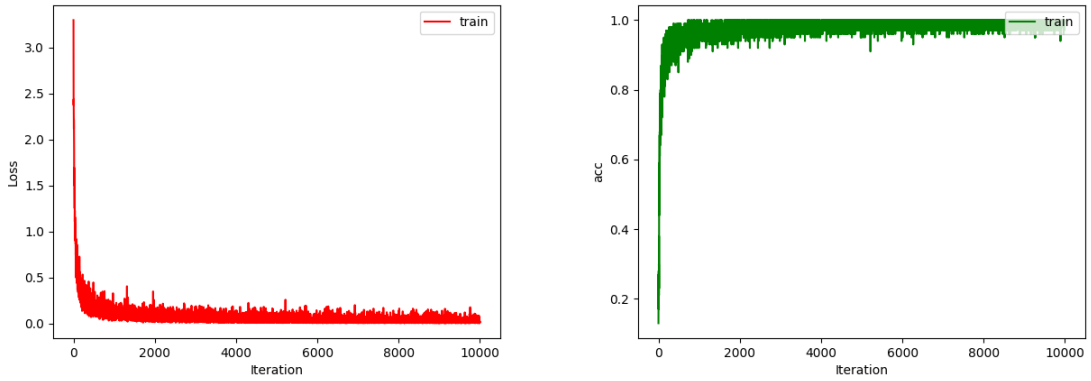

图 4: CNN with BN

测试的准确率：98.64%

3.4.3 小结

以上，我们只能说比较遗憾，在测试结果层面二者的区别不是很大，虽然在训练 acc 的比较中可以看出数据的波动稍微有所弱化，但是在 loss 层面，优化的情况并不明显。

3.5 探索尝试

通过上述的试验，我们发现 BN 层对 MLP 的优化比较明显，但是对于 CNN 就显得比较弱势。

3.5.1 BN 层策略对比动态调整 LearningRate 策略

而这种优化就体现在对收敛速率的提升上，同样，learningrate 的逐步衰减也可以达到提升收敛速率。在之前的试验中（PA_2）逐步衰减，不仅在 acc 训练集层面能有较为明显的收敛速率提升，而且在 loss 层面也有较好的表现，但是二者各自发挥的领域不同。二者是可以相结合的。

4 存在问题

事实上归一化的好处在于，让一个 batch_size 中每个训练数据对这个网络的贡献具有无偏性。BN 对 MLP 如此明显的可能在于，MLP 对图像的识别不具有局部性，因此对于同于数字在不同的位置所训练出的数据差别是比较大的，通过 BN 的归一化可能会减少这种位置偏移的影响，但是 CNN 却不同，卷积核的处理单元是遍布全图的，卷积的结果在一定程度上抵消掉了图像位置差异的问题，所以 BN 对 CNN 的影响不是非常大。但至于具体 BN 的真正影响的机制，自然还是后话了。

5 总结

通过本次试验，进一步熟悉了 tensorflow 的框架使用，也算有不少的收获吧。