train1

吴琪

July 2024

这个题目还是比较熟悉的，不仅是因为在本科课程认知科学与类脑计算的大作业里面学习了图像分割与分类，并且在实验室前段时间的测试当中也选取了相同的题目，所以在这里的培训就引用了上次的测试结果，并进行了一些改正。

网络模型参考了 LeNet 模型，首先是输入层，大小为 32*32*3，这样方便表达出图像潜在的明显特征，然后对输入层进行第一次卷积，通过 6 个 5*5 的卷积核将图像变为 28*28*6，再加入一个池化层，变为 14*14*6，以便于后续检测更多的特征信息。接下来再通过有 16 个 5*5 的卷积核的卷积层和池化层，将输入变为 5*5*16。为了方便，后面全部采用了全连接层，将输出变为线性 120，再加入新的全连接，变为线性 84，最后通过全连接变为 10 并输出。使用 PlotNeutralNet 绘制的网络框架如图 1：

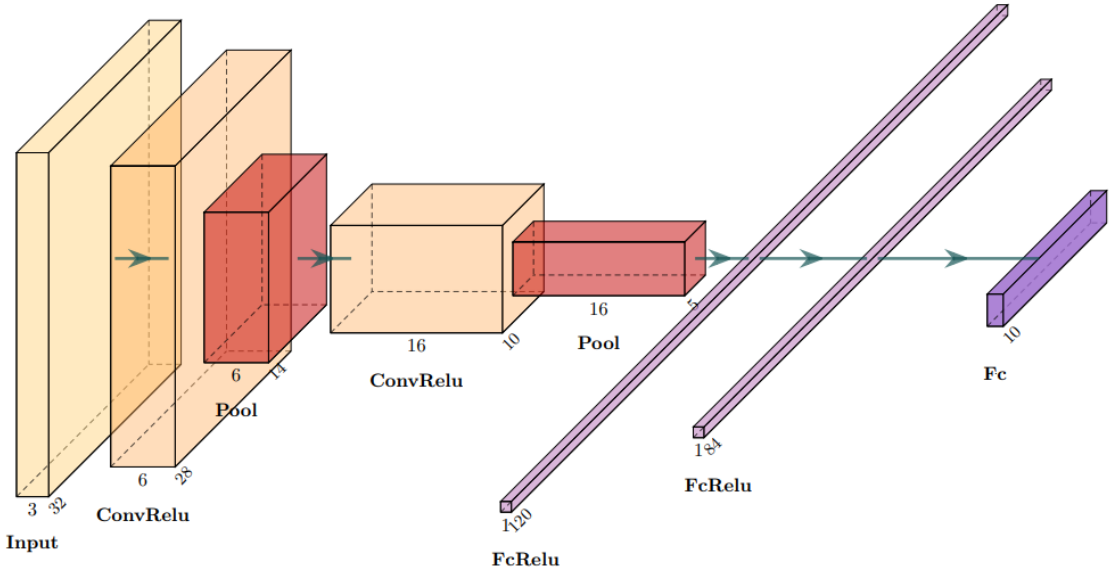

图 1: The structure of the net

损失函数采用了交叉熵损失函数，即 nn.CrossEntropyLoss。交叉熵用来表达两个概率分布之间的相似性，熵越大则表示差别越大，损失值也就越大；交叉熵的值越小，两个概率分布就越接近。假设概率分布 p 为期望输出，概率分布 q 为实际输出，则交叉熵定义为：

$$H(p,q) = -\sigma_x(p(x)logq(x) + (1-p(x))log(1-q(x))) \tag{1}$$

不过引用的交叉熵损失函数的计算方式采用了另一种方式：

$$loss(x, y) = L = \{l_{1,\dots,l_N}\}^T \tag{2}$$

$$l_n = -\Sigma_{c=1}^{C} \omega_c log \frac{e^{x_{n,c}}}{\Sigma_{i=1}^{C} e^{x_{n,i}}} \cdot y_{n,c} \tag{3}$$

它相当于 softmax+log+nllloss，前面是熟悉的计算，后者用于实现负对数似然函数中的负号。

接下来加载训练集与测试集，获取设计好的 Net 和损失函数。首先测试 optim 中不同的优化器，设计了五种带有不同优化器的 Net：net_SGD, net_Momentum, net_RMSprop, net_Adam, net_Adagrad。在训练中，记录每 100 次产生的 loss，并在每个 epoch 后对测试集进行预测，记录预测准确率，下面是在 15 个 epoch 内产生的 loss 和每个 epoch 对应的预测准确率：

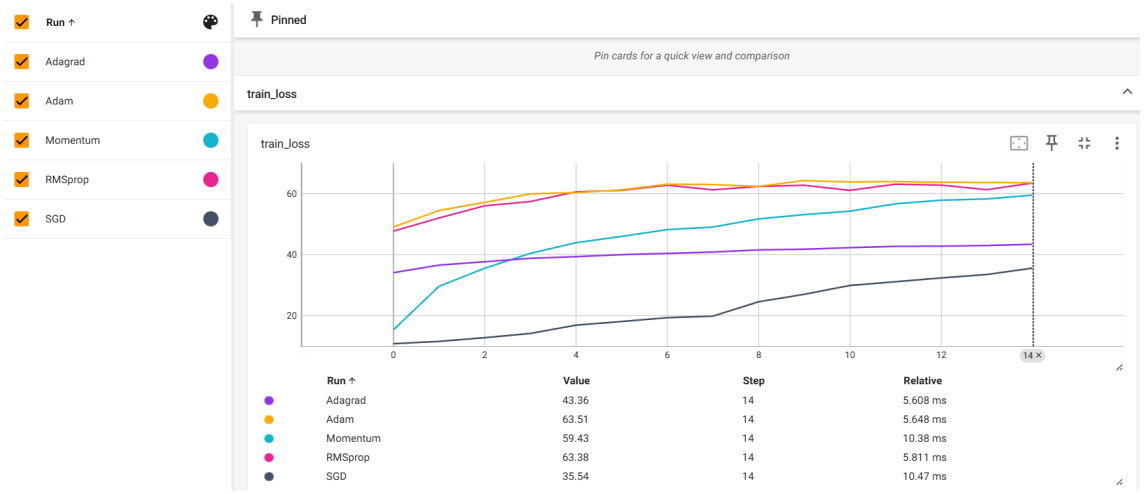

图 2: Different optim's train loss of each batch

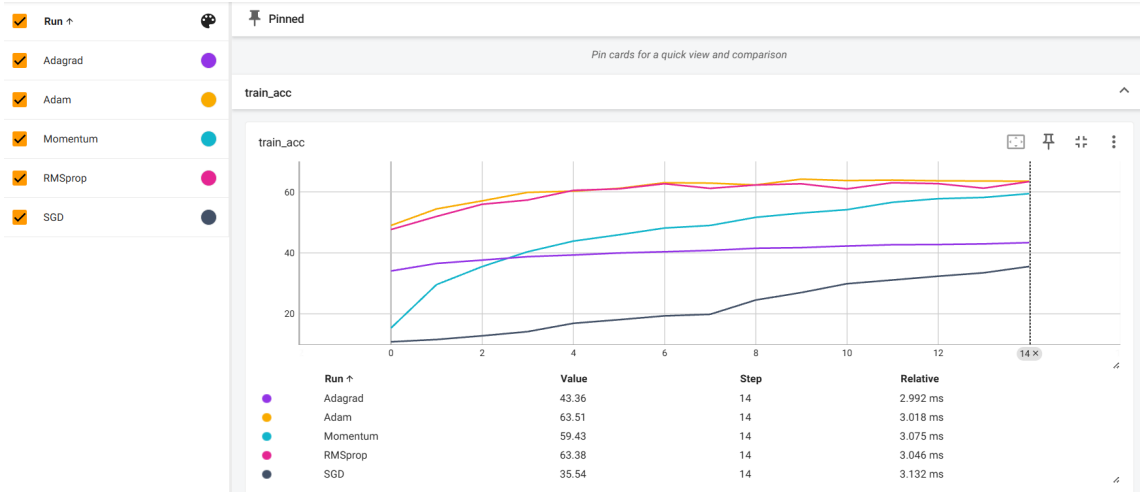

图 3: Different optim's accuracy of each batch

PyTorch 中的 optimizer 这五种优化方法，可以分为两类：SGD 及其改进 Momentum 和逐参数适应学习率方法，即 AdaGrad、RMSProp、Adam。SGD（随机梯度下降）是最普通的

优化器，计算训练集中的小批量，基本没有加速效果。Momentum 是 SGD 的改进版，它加入了动量原则，在当前 step 的参数更新中加入了部分上一个 step 的梯度。AdaGrad 在迭代过程中为每个参数维护一个学习率缩放因子，该因子根据该参数的梯度更新进行自适应调整。RMSProp 算法引入了一个衰减系数来控制梯度信息的衰减速度。Adam 算法不仅利用梯度的一阶矩估计，还利用梯度的二阶矩估计。经过不同优化器的比较，可以看出 RMSprop 和 Adam 在短 batch 内效果最好，Momentum 算法则比较流畅。

接下来考虑了不同的 learning rate 下的训练效果。尝试了 0.001，0.005，0.01, 0,1 四种情况下的 loss 与预测准确率，下面是在 10 个 epoch 内产生的 loss 和每个 epoch 后的预测率：

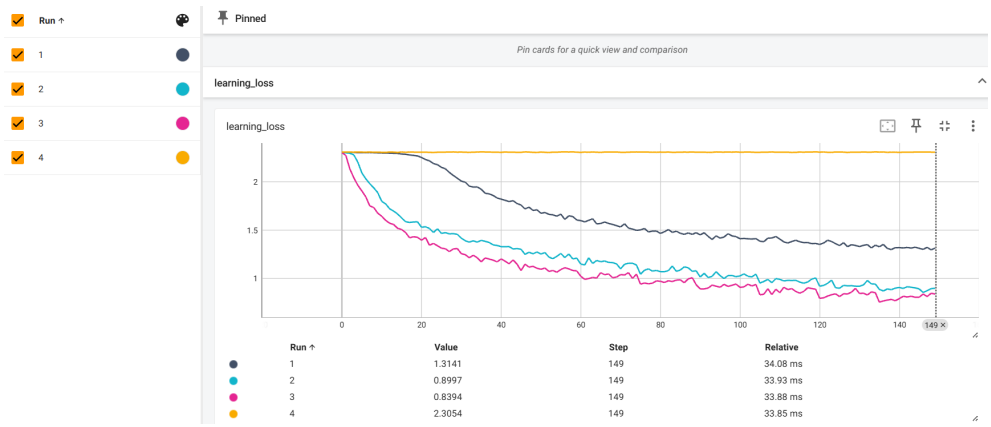

图 4: Different learning rate's train loss of each batch

图 5: Different learning rate's accuracy of each batch

可以发现，在初始 learning_rate=0.001 的基础上，逐渐增加，得到的效果是逐渐变好的，但是差异在减弱，当 learning_rate 持续增加时，效果会出现下降的趋势。而当 learning_rate=0.1 时，训练过程中没有出现迭代效果，应该是学习率设置过大导致的参数调整失衡。最好的参数应该在小 epoch 内多次测量，观察参数的变换或者曲线的增长率，选择较平滑的设定为初始 learning_rate。或者直接采用自适应梯度优化器，在训练过程中能够自动调整参数的学习率。

最后是选取图片进行测试，设定好标签，从训练集中选取几张照片，然后预测出它们的标签并输出。

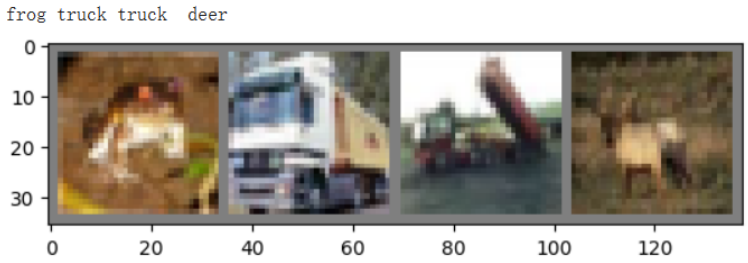

图 6: Precesion of the pictures

思考时间，和目前更先进的算法差距还是挺大的，LeNet 架构主要是卷积和下采样相结合，网络深度有限导致网络性能不高，并且网络参数有限，不能学习到很复杂的特征。如果提升效果的话可以先在网络结构上进行优化，例如增加卷积层和全连接层的数量，增强网络的学习能力，或者引入激活层。在训练策略上，采用更优化的算法、更合理的超参数设置等，进一步提升模型的泛化能力，相比之下模型总体上还是有很大的不足的。

之前学习过程中接触过一些简单的图像分类和一些简单的网络模型，经过前段时间的测试和这次培训之中，这些知识又得到了回顾，输入图像和对应的标签，把图像输入到网络中，计算网络的输出和对应标签之间的损失再反向传播，或者更新权重。其实在完成框架后，还有很多细节要做，如何绘制 Net 模型，如何使用 tensorboard 绘制图像，而且还接触到了 gradio 库，这些能够直观化的工具对于学习都有很大的帮助。此外，如何系统的比较多个优化器、多个超参数构成的网络之间的效果，这也要重新去学习。尽管之前接触过类似数据集的图像分类，但是在完善这个测试的过程中还是可以学习到新的东西。题目的开放性很高，可以给我们很多发挥的地方，而且在学长的指导下，也是学到了很多应用工具，对完成实验和报告有很大的帮助。