SE125: 机器学习 2021 年 12 月

实验报告——CNN 图像分类

小组成员 和敬凯，钱昱辰，葛煜

1 概述

在本实验中, 我们基于卷积神经网络对 CIFAR10 图集进行了图像分类实验. 我们使用了 torch 框架进行模型构建和训练, 并使用了 tensorboard 可视化训练流程并记录和可视化训练数据 (如准确率和 loss 曲线). 经过多次测试和参数调整, 我们的 100 次迭代的各项分类平均准确率可以稳定在 96% 左右.

在本报告中，我们首先介绍系统的详细设计，包括数据预处理，模型设计和训练方法，接下来我们将介绍实验过程和进行的调参实验，我们进行的调参实验包括对不同学习率、有无 dropout 层、不同卷积层个数和有无 BN 层的测试，最后，我们展示了实验得出的最佳参数的实验结果，并对本作业做了简要总结.

本小组成员有和敬凯 (519021910187)，钱昱辰 (519021910671)，葛煜 (519021910553).

2 系统设计

2.1 数据预处理

为了加速训练过程, 使梯度下降过程更稳定并抑制过拟合, 我们首先使用 torch 的 transforms.ToTensor 和 transforms.Normalize 方法对数据进行预处理，transforms.ToTensor 方法将图片的各通道参数归一化到 [0,1] 区间, transforms.Normalize 将各通道参数进一步使用正态分布归一化到 [-1,1], 参考现有资料, 我们将参数设置为 mean = [0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5].

预处理方法如下:

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

2.2 模型设计

图 1 展现了我们的 cnn 网络的可视化流程. 我们设置了三层卷积层来提取数据特征，数据在每一层会经过卷积-池化-BN 处理. 如图所示，输入已预处理的 CIFAR10 图集后，数据会经过一层卷积层处理，之后进行池化，在数据池化之后会进行一次 BN，再重复 2 次相同的操作. 之后，数据会经过 dropout 层进行 40% 的 dropout 来抑制过拟合，以及 2 次全连接层来对数据特征进行非线性组合，最终得到输出.

经过测试, 3 层卷积层可以较好的提取图像的特征, 使用 3 层以上卷积层会减缓训练过程, 并出现过拟合; 而使用三层以下卷积层则无法较好的提取图像特征, 分类的准确率较低, 调参实验的详细描述将在下节展示.

在每层卷积和池化后我们设置了 BN 层, 这有利于稳定训练过程, 抑制梯度爆炸和梯度消失.

Dropout 有效防止了过拟合的发生, 这在我们的测试中得到了证明. 经过实验, 我们将 dropout rate 设置为 0.4.

全连接层对卷积层得到的特征进行非线性整合, 并最终输出分类结果. 我们设置了两层全连接层, 数据在进入每层全连接层前会进行 dropout.

模型类代码如下:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=(3, 3), padding=1)
        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=(3, 3), padding=1)
        self.conv3 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=(3, 3), padding=1)
        self.bn1 = nn.BatchNorm2d(16)
        self.bn2 = nn.BatchNorm2d(32)
        self.bn3 = nn.BatchNorm2d(64)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(64 * 4 * 4, 500)
        self.fc2 = nn.Linear(500, 10)
        self.dropout = nn.Dropout2d(0.4)

    def forward(self, x):
        x = self.bn1(self.pool(torch.nn.functional.relu(self.conv1(x))))
        x = self.bn2(self.pool(torch.nn.functional.relu(self.conv2(x))))
        x = self.bn3(self.pool(torch.nn.functional.relu(self.conv3(x))))
        x = x.view(-1, 64 * 4 * 4)
        x = self.dropout(x)
        x = torch.nn.functional.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

2.3 训练方法

我们使用梯度下降和反向传播的方法对模型进行训练. 经过多次测试, 我们将学习率设置为 0.005. 梯度下降使用了 torch 的 optim.SGD 函数.

```
optimizer = torch.optim.SGD(model.parameters(), lr=0.005)
```

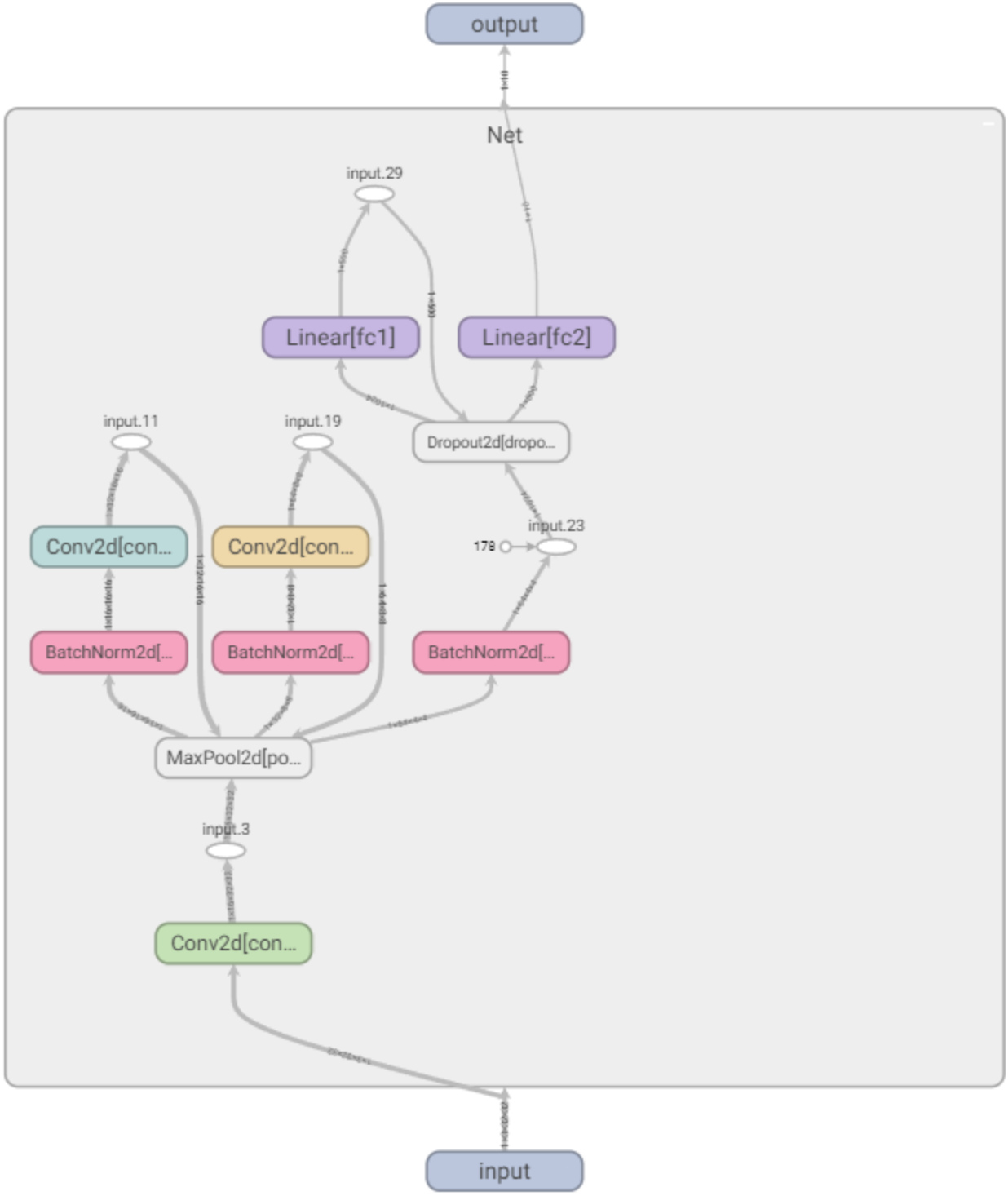

output
Net
input.29
Linear[fc1]
Linear[fc2]
input.11
input.19
Dropout2d[dropo...
input.23
179
Conv2d[con...
Conv2d[con...
BatchNorm2d[...
BatchNorm2d[...
BatchNorm2d[...
MaxPool2d[po...
input.3
Conv2d[con...
input

图 1: 实验模型流程概览

训练过程如下:

```
for epoch in range(1, n_epochs + 1):
    train_loss = 0.0
    model.train()
    for data, target in train_loader:
        data, target = data.cuda(), target.cuda()
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()*data.size(0)

    train_loss = train_loss/len(train_loader.sampler)
```

每轮训练后将在 tensorboard 中记录 loss 数据用于模型评估. 训练轮次设置为 100 次.

2.4 训练过程监控和可视化

为了更好地分析模型和优化参数，我们使用 tensorboard 记录中间参数 (如分类准确率和 loss) 并做可视化. 在每轮迭代后我们将训练集的 loss 记录在 tensorboard 中，每经过 5 轮迭代后我们分析测试集的分类准确率和 loss 并记录，这有利于我们分析参数设置和监控训练过程.

相关代码示例如下，相关参数的可视化结果将在下节展示并分析.

```
writer.add_scalar('Loss of Train Data',train_loss,epoch)
```

2.5 结果评估

我们使用测试集的分类准确率 (包括平均分类准确率和各类图片的分类准确率) 和 loss 来评估分类结果，具体代码如下，我们使用 testAccuracy 来分析最终分类结果和监控模型训练过程.

```
def testAccuracy(loader, model):
    testLoss = 0.0
    classCorrect = list(0. for i in range(10))
    class_total = list(0. for i in range(10))
    model.eval()
    for data, target in loader:
        if train_on_gpu:
            data, target = data.cuda(), target.cuda()
        output = model(data)
        loss = criterion(output, target)
        testLoss += loss.item() * data.size(0)
        _, pred = torch.max(output, 1)
        correct_tensor = pred.eq(target.data.view_as(pred))
```

```
        correct = np.squeeze(correct_tensor.numpy()) if not train_on_gpu else np.squeeze(correct_
        for i in range(batch_size):
            label = target.data[i]
            classCorrect[label] += correct[i].item()
            class_total[label] += 1

    test_loss = testLoss / len(loader.dataset)
    writer.add_scalar('Loss of Test Data',
                      test_loss,
                      epoch)

    for i in range(10):
        className = classes[i]
        writer.add_scalar('Accuracy of ' + className+'/%',
                          100. * classCorrect[i] / class_total[i],
                          epoch)
    writer.add_scalar('Overall Accuracy/%',
                      100. * np.sum(classCorrect) / np.sum(class_total),
                      epoch)
```

3 实验过程和调参实验

在实验过程中，我们设定了一系列的参数，包括训练学习率 (lr)、dropout 的比率、卷积层的层数、是否使用 BN 层。为了方便对比分析参数对实验的影响，我们设定了一组对照用的参数，为学习率 lr=0.005，dropout 比率 0.4，卷积层 3 层，使用 BN，将这种参数条件下的数据作为对照组。按照实验要求，每次实验都终止于第 100 次迭代。

Definition 1. *对照组：学习率 lr=0.005，dropout 比率 0.4，卷积层 3 层，使用 BN，这是我们最终选择的最优参数设置*

注：由于我们的疏忽，考察训练集数据的 loss 时，计算了平均 loss，而在考察测试集数据的 loss 时，没有将计算出的 loss 取平均值，所以两个 loss 的数值相差了 100000 倍，由于这个问题在撰写报告时才被发现，如果返工工作量会比较大，所以我们还是采用了原始数据，如果对读者造成困扰我们十分抱歉。

3.1 训练过程

为方便起见，我们在此使用对照组实验来展示实验的训练过程，由于训练过程的理论逻辑以及代码实现已经在本实验报告 2.3 模块中详细说明，所以在本模块中我们仅展示训练过程的数据变化过程。

我们利用 tensorboard 的图形化工具图形化了实验过程中 loss、测试集准确率随 epoch 的变化过程。

对照组（lr=0.005,dropout 比率 0.4，卷积层 3 层，使用 BN）的训练集 loss、测试集 loss、测试集准确率随 epoch 变化的曲线图分别如图 2，图 3，图 4 所示。

可以看出变化趋势，训练过程中测试集和训练集 loss 都是呈先快速下降，后缓慢下降，最后趋于收敛的趋势，而测试集的准确率则呈先快速上升，后缓慢上升，最后趋于收敛的趋势。这组实验在训练过程中没有出现过拟合。

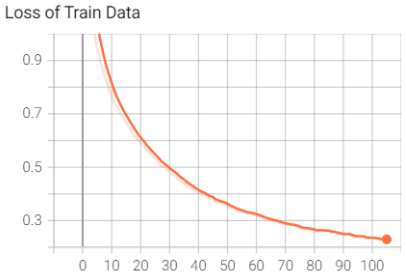


图 2: 训练集数据 loss-epoch

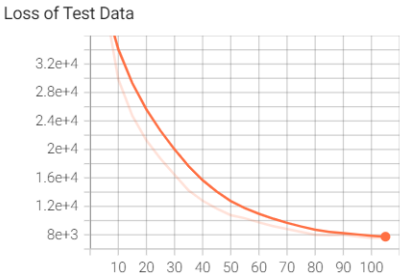


图 3: 测试集数据 loss-epoch

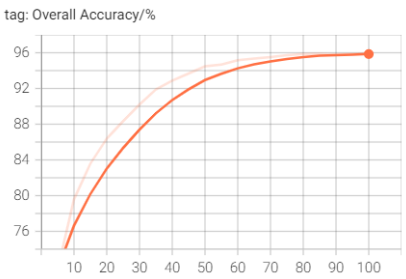


图 4: 测试集 accuracy-epoch

3.2 调参实验及分析比较

在本实验中，我们对上述四种参数训练学习率 (lr)、dropout 的比率、卷积层的层数、是否使用 BN 进行了调整，分析比较了调参后模型和对照组模型的 loss、准确率以及其变化趋势，提出了一些我们的看法。

3.2.1 学习率对训练过程和结果的影响

在其他三种参数保持和对照组不变时，更改学习率 learning rate 的值。将参数学习率 lr 改为 0.003。测试集准确率变化趋势如图 5 所示。

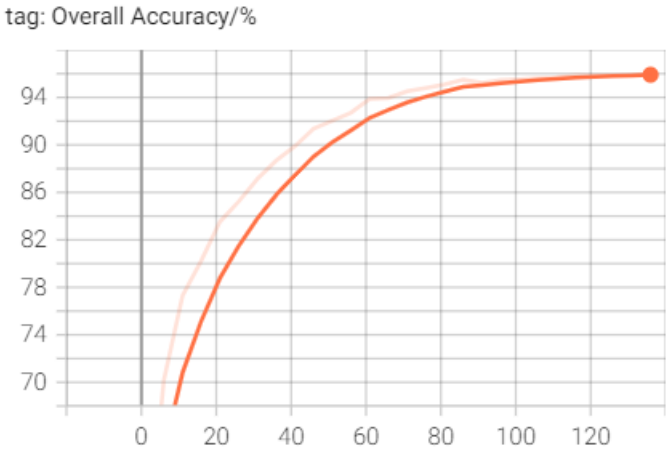


图 5: 学习率为 0.003 时的测试集准确率趋势曲线

为了完整地展示测试集准确率的变化规律，我们将最大 epoch 设置成了 130，可以看到，当把学习率调低时，在 epoch=100 时准确率曲线并未收敛，而在 epoch 接近 120 时才收敛，训练过程中准确率收敛需要的迭代次数会增加。同时由于 epoch=100 时仍未收敛，此时的训练集准确率为 95.54%，低于对照组。

Observation 2. 学习率过低会影响模型在训练时的收敛速度，且模型最高准确率提升不明显.

下面将参数学习率 lr 改为 0.008.

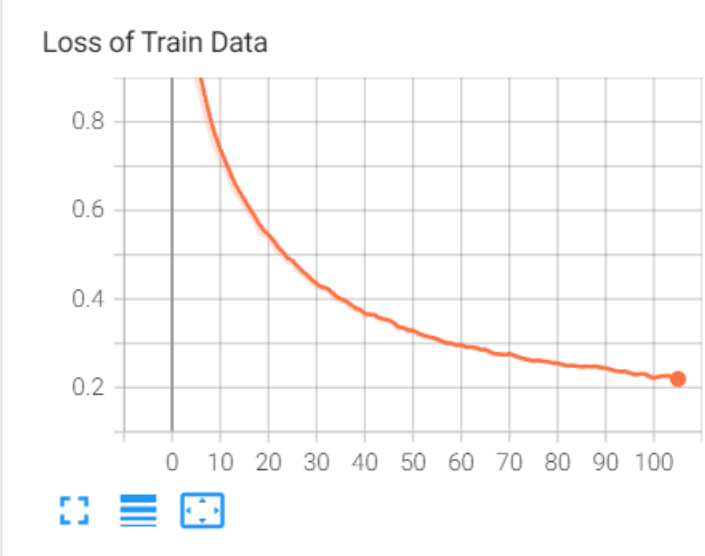


图 6: 学习率为 0.008 时的测试集 loss 趋势曲线

可以看到，在训练过程中 train loss 经常起伏波动，说明学习率过高会影响训练的稳定性，导致训练过程难以较快收敛。epoch=100 时的训练集准确率为 96.02%。

Observation 3. *学习率过高会影响训练过程的稳定性，影响模型收敛速度，使模型很难较快收敛到局部最优解.*

3.2.2 dropout 层的有无对训练过程和结果的影响

在本小节中，我们保持其他三种参数保持和对照组相同，更改 dropout 的比率。将 dropout 比率改为 0.5。测试集准确率变化趋势如下图所示。

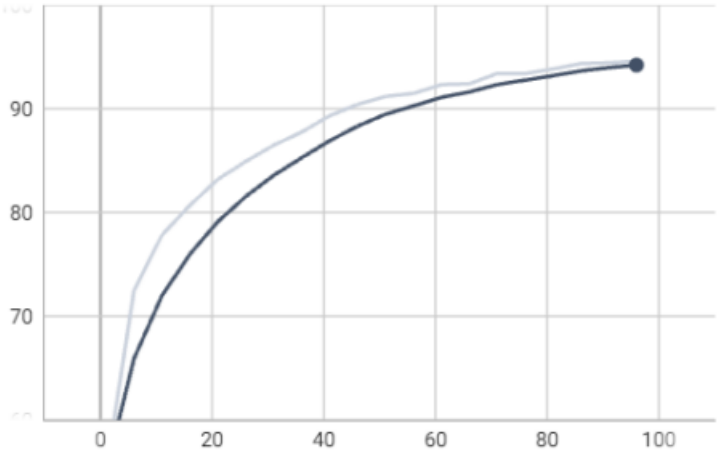
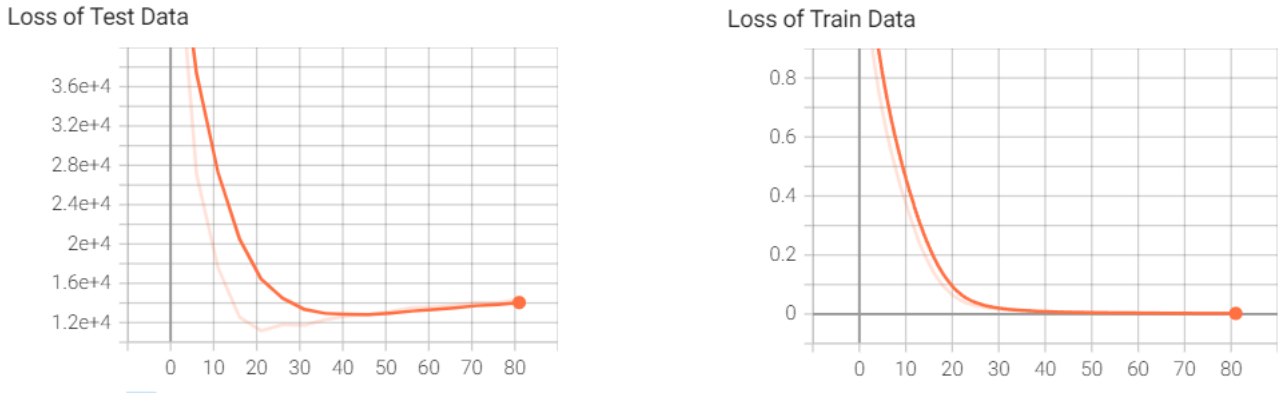


图 7: dropout 为 0.5 时的测试集准确率趋势曲线

可以看到在 epoch=100 时并未收敛，说明 dropout 的比率过大时，需要更长的训练过程来完成训练。在 epoch=100 时，测试集准确率为 94.54%，低于对照组。

Observation 4. *dropout 率过高会影响模型收敛速度和准确率.*

我们还测试了没有 dropout 层的情况下的情况。



(a) 测试集 loss　　(b) 训练集 loss

图 8: 没有 dropoout 层情况下训练情况

可以看到虽然测试集 loss 逐渐收敛，但是测试集的 loss 在 epoch=30 附近出现了不减反增的情况，说明训练过程中出现了过拟合现象 (模型对训练集的个性化特征过于重视，导致模型无法学习到数据的普遍规律，从而导致测试集的 loss 上升)，说明 dropout 可以有效防止过拟合。

Observation 5. *dropout 率过低或没有 dropout 层会导致模型训练容易出现过拟合.*

3.2.3 卷积层个数对训练过程和结果的影响

在本节中，我们在其他三种参数保持和对照组相同的条件下，更改卷积层的层数。

将卷积层改为 2 层，发现各种评估指标的趋势没有明显变化，但 epoch=100 时测试集准确率为 95.45%，低于卷积层为 3 层时的准确率，同时每轮训练所需的时间减少，但达到收敛所需的训练轮次增加。

将卷积层改为 4 层，发现模型训练过程中出现一定的过拟合现象，epoch=100 时测试集准确率为 95.31%，低于卷积层为 3 层时的准确率，同时每轮训练所需时间增加。

Observation 6. *卷积层过少时模型参数过于简单，无法提取到数据的全部特征，分类准确率低；卷积层过多时模型参数过于复杂，容易出现过拟合，也会影响分类准确率.*

同时卷积层较少时每轮训练所需时间短，但达到收敛所需的迭代次数多，卷积层较多时每轮训练所需时间长，但达到收敛所需的迭代次数少.

3.2.4 BN 层对训练过程和结果的影响

在本节中，我们在其他三种参数保持和对照组相同的条件下，去掉 BN 层。

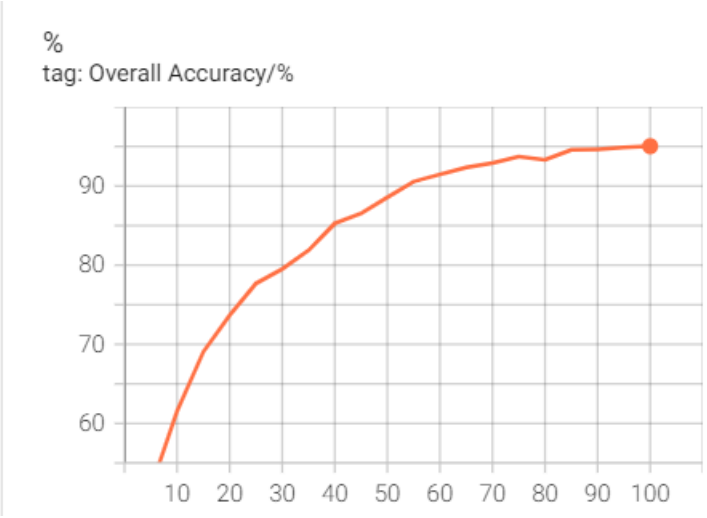


图 9: 去掉 BN 之后的测试集准确率趋势曲线

可以看到测试集准确率在训练过程中产生了较大起伏，说明 BN 层有助于稳定训练过程。此外，发现去掉 BN 之后程序的运行时间比去掉 BN 之前长，说明 BN 也可以有效地缩短训练时间。

Observation 7. *BN 层有助于稳定训练过程，并缩短训练时间*

4 实验结果和最高准确率

4.1 实验得到的最佳参数设置

经过实验，我们发现采用学习率 lr=0.005，dropout 比率 0.4，卷积层为 3 层，使用 BN 层的参数设置时的模型分类准确率最高.

4.2 最高准确率

在 epoch=100 时，使用上述模型得到的测试集数据分类准确率为 96.02%，不限定训练次数的模型最高分类准确率也为 96.02%，因为 100 次迭代后模型训练达到收敛.

下面是使用上述模型 100 次迭代后各类图像的分类准确率.

我们发现 dog 和 cat 的分类准确率明显低于其他类别，这可能是由于 dog 和 cat 长相上相似度较高造成的，在之后的改进模型中，我们可以根据 dog 和 cat 的特征对模型做针对性改进来提高 dog 和 cat 的分类准确率.

此外，各类图像分类准确率变化的趋势并不完全相同，有些类别的图像的分类准确率在训练结束时尚未达到收敛 (如 cat)，但是有些类别的图像的分类准确率在训练结束时已经在达到峰值后下降 (如 airplane，可能是过拟合引起的)，最高的平均分类准确率并不意味着所有图像的分类准确率都达到了最高，这也为我们改进模型提供了思路.

表 1: 各类图像分类准确率

类别名	分类准确率
airplane	96.8%
automobile	98.5%
bird	93.52%
cat	91.92%
deer	95.5%
dog	93.52%
frog	97.78%
horse	96.94%
ship	98.34%
truck	97.52%
average	96.03%

5 实验总结

在本次实验中，我们调研并学习了使用 CNN 实现图像分类的知识，完成了训练和测试的代码，并调整和测试了了一些参数，对结果进行了比较分析。在调参、比较、分析的过程中，我们总结了一些规律和结论，如

- 合适的学习率对于模型和训练十分重要，学习率过低时模型收敛缓慢，学习率过高时模型训练不稳定并很难达到局部最优解.
- dropout 可以有效防止过拟合，但 dropout 比例过大会导致模型 underfit 从而影响模型分类准确率.
- BN 可以有效稳定训练过程并加快训练速度.
- 搭建 CNN 模型时为了取得最高分类准确率应该不断增加卷积层数目直到出现过拟合为止，但是卷积层过多也会导致训练速度缓慢，某些情况下需要平衡这两者之间的关系.
- 在 CNN 模型中前几个卷积层应该尽量宽以便充分提取数据特征，后几个卷积层应该逐渐变窄以便充分组合各类特征.

本模型还存在很多不足之处，可以在以后加以改进，如

- 本模型在每层卷积后都添加了 BN，但是一些资料显示，在前几个卷积层后无需添加 BN，且在前几个卷积层添加 BN 可能会影响样本特征的充分提取和模型的分类准确率.
- 在平均分类准确率达到最大时一些类别的分类准确率尚未达到收敛，而一部分类别的分类准确率已经出现了过拟合，这可能影响模型在不同场景下的应用，可以针对不同样本的特征学习情况对模型做针对性的改进.
- 部分特征相近的类别的图像的分类准确率显著较低 (如 dog 和 cat)，可以针对这类情况做针对性的改进.
- 可以充分利用现有硬件的并行计算能力对训练做并行化改进来加快训练速度，此外，可以利用现有的分布式计算框架对模型做改进.

本次实验帮助我们巩固了知识，培养了调参、分析的能力，使我们受益匪浅。

参考文献

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Radio Engineers*, 86(11):2278–2323, 1998.

[4] Yue Ding Xiaodong Gu. 课程资料. 2021.

[5] 周志华. 机器学习. 2016.