Cifar-10 Classification with MLP and CNN 实验报告

计81 肖光烜 2018011271

MLP

网络结构

```
Model(
  (fc1): Linear(in_features=3072, out_features=256, bias=True)
  (bn1): BatchNorm1d()
  (act): ReLU()
  (dropout): Dropout()
  (fc2): Linear(in_features=256, out_features=10, bias=True)
  (loss): CrossEntropyLoss()
)
```

最佳超参数设定

经过几次尝试发现初始的`1e-3`确实是一个比较合适的学习率，因此没有调整。隐层维数选择了`256`，也没什么理由，并且这次实验的重点也不是MLP，就不调整了。

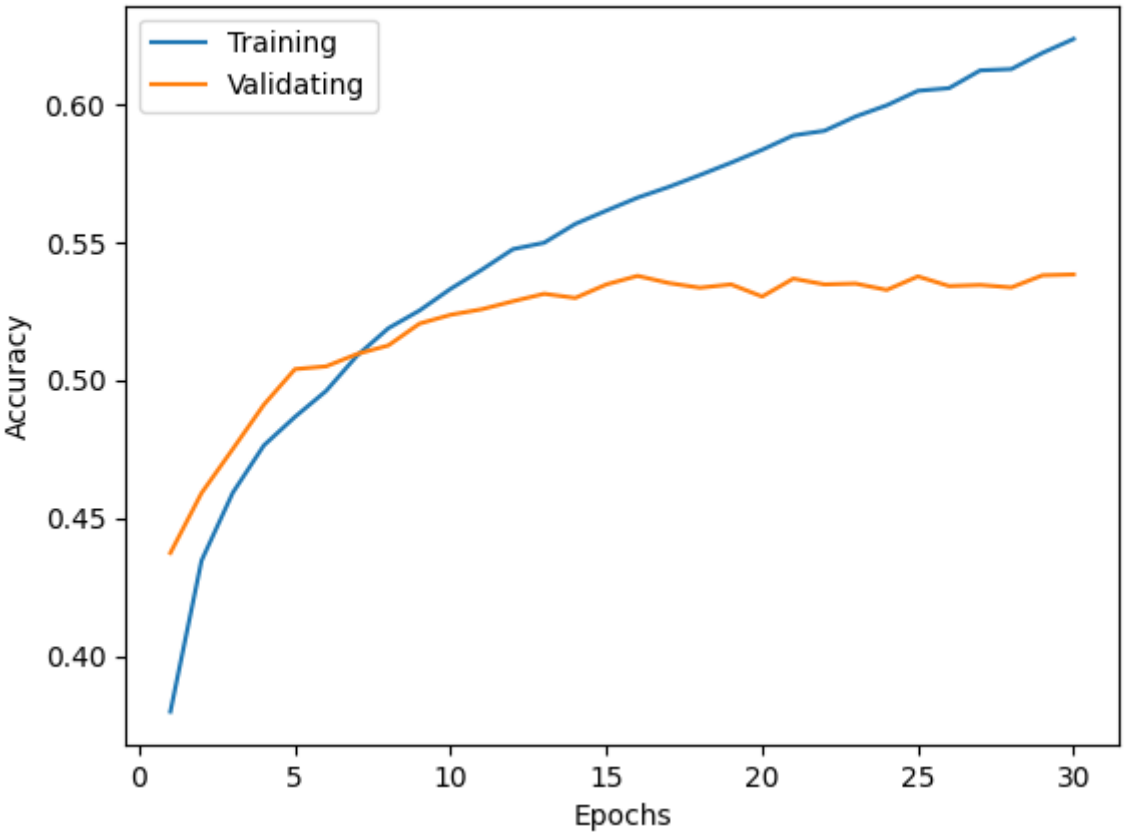
```
batch_size = 100
learning rate = 1e-3
batch_norm_momentum = 0.1
dropout_rate = 0.4
hidden_dim = 256
num_epochs = 30
```

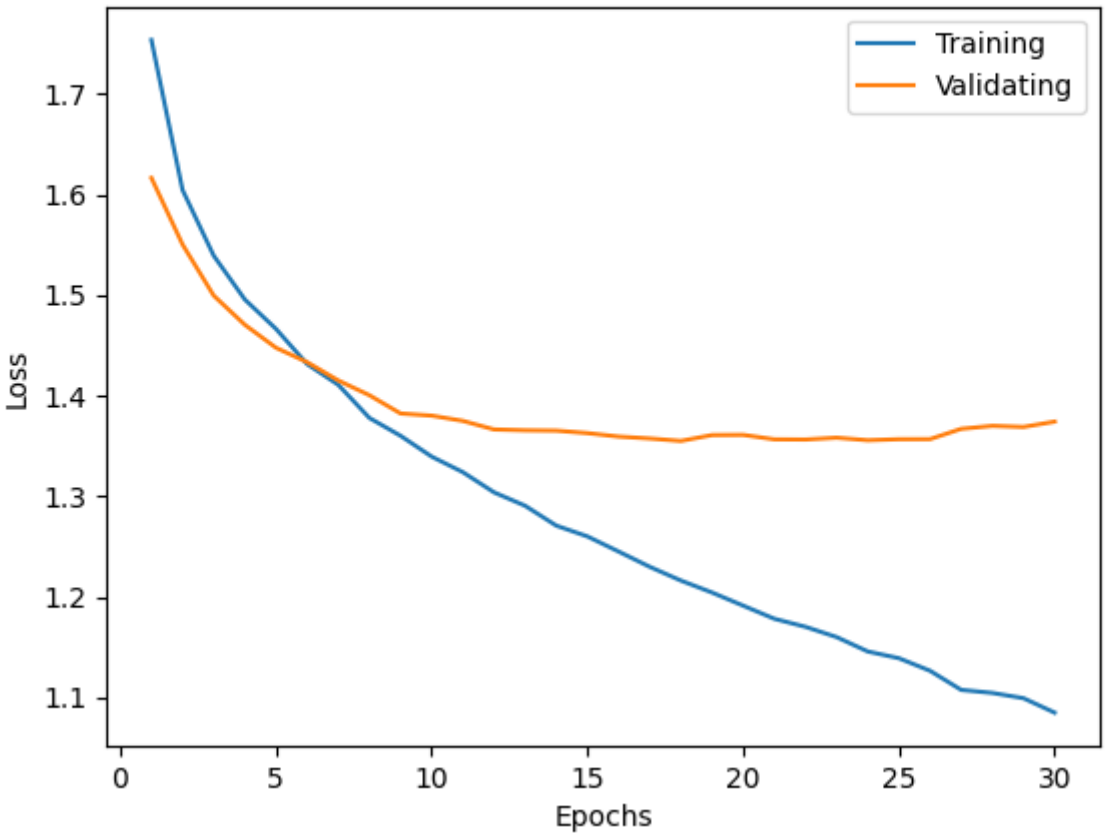
最佳模型的训练结果

在验证集上取得最佳准确率的模型在测试集上取得53.48%的准确率。

```
Epoch 30 of 30 took 2.989814519882202s
  learning rate:                 0.001
  training loss:                 1.0850399507582187
  training accuracy:             0.6237499874830246
  validation loss:               1.3743274533748626
  validation accuracy:           0.5383999875187874
  best epoch:                    30
  best validation accuracy:      0.5383999875187874
  test loss:                     1.3500653505325317
  test accuracy:                 0.5347999873757362
```

训练曲线：


Training
Validating
Accuracy
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30
Epochs


Training
Validating
Loss
1.7
1.6
1.5
1.4
1.3
1.2
1.1
0
5
10
15
20
25
30
Epochs

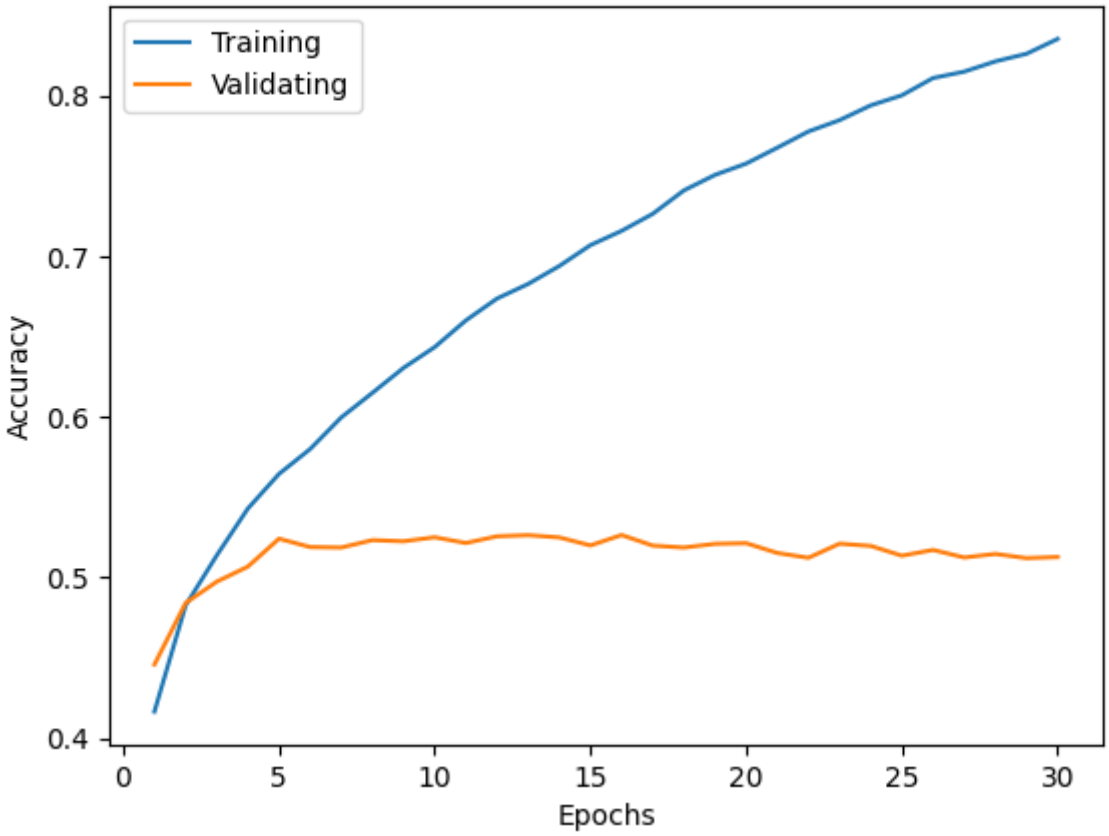
不同Dropout Rate的模型表现

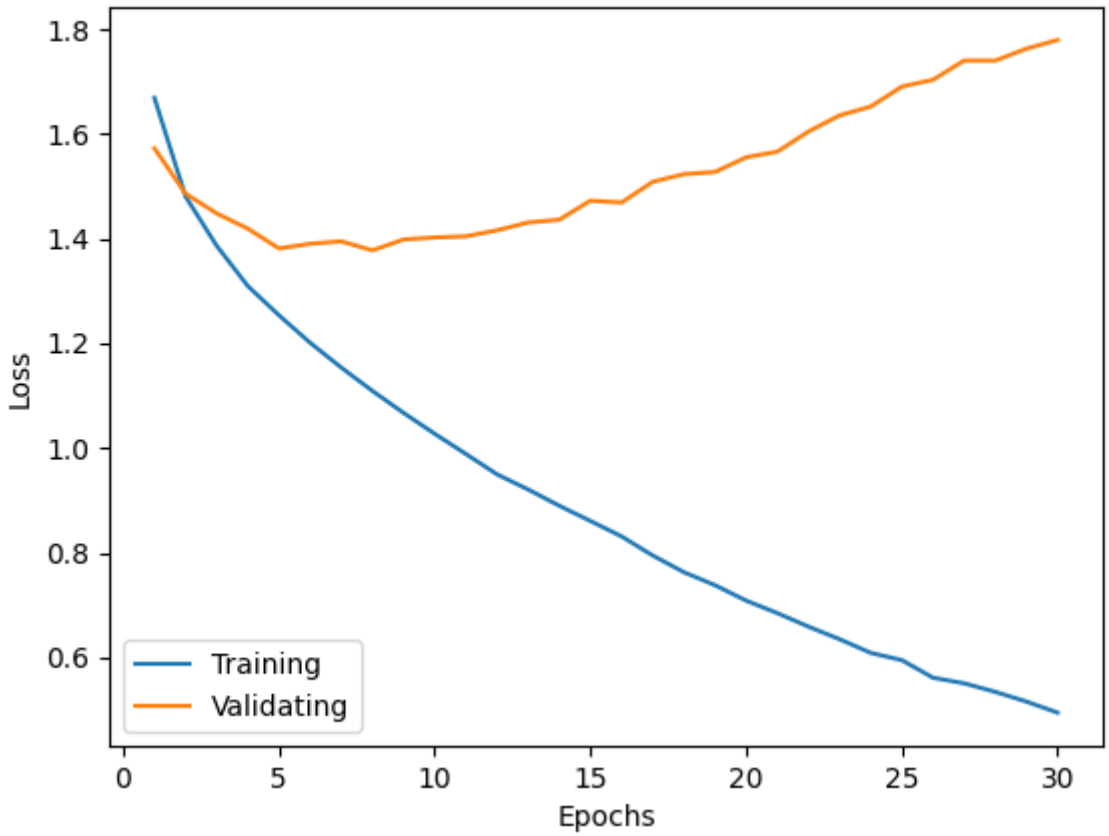
我在0~100%每隔10%取值，测试不同Dropout Rate情况下模型的表现，结果如下：

Dropout Rate (%)	Test Accuracy of the Best Validation Model (%)
0	51.82
10	53.09
20	53.26
30	53.39
40	**53.48**
50	53.02
60	52.47
70	51.60
80	49.51
90	44.96

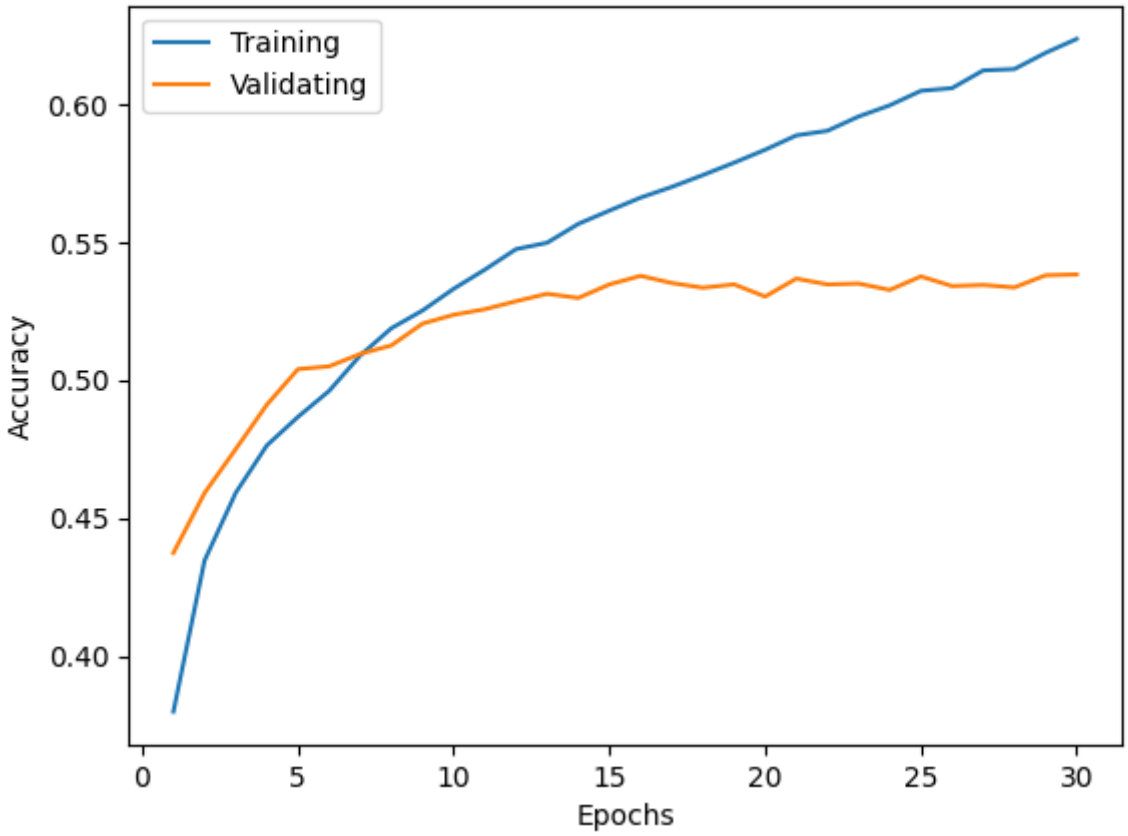
可以明显看出模型最终在测试集上的表现随着Dropout Rate的提高呈现先增加后减小的趋势，这是可以理解的。当Dropout Rate很小乃至没有时，模型很容易过拟合到训练集上，导致在测试集上表现不佳；当Dropout Rate过高时，模型中的参数在训练时不能被充分训练，相当于训练过程中模型参数量过小，不能有效学习数据集的分布。

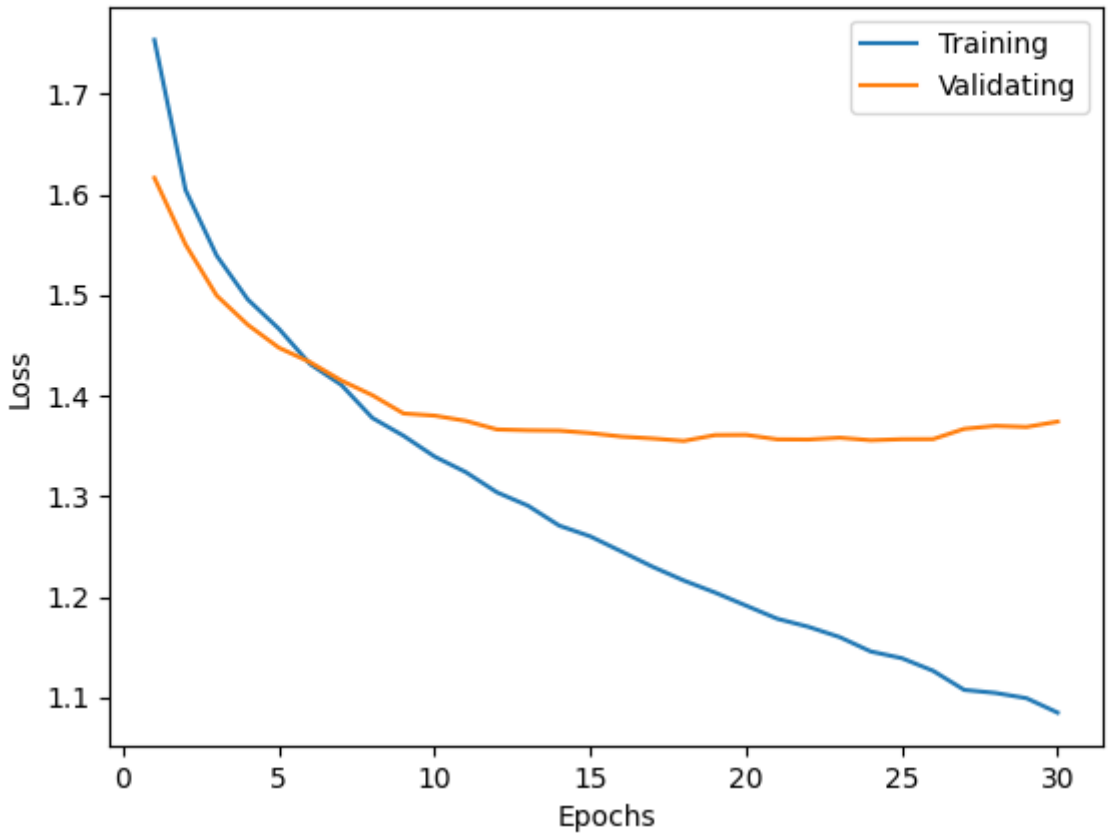
无Dropout的MLP训练曲线



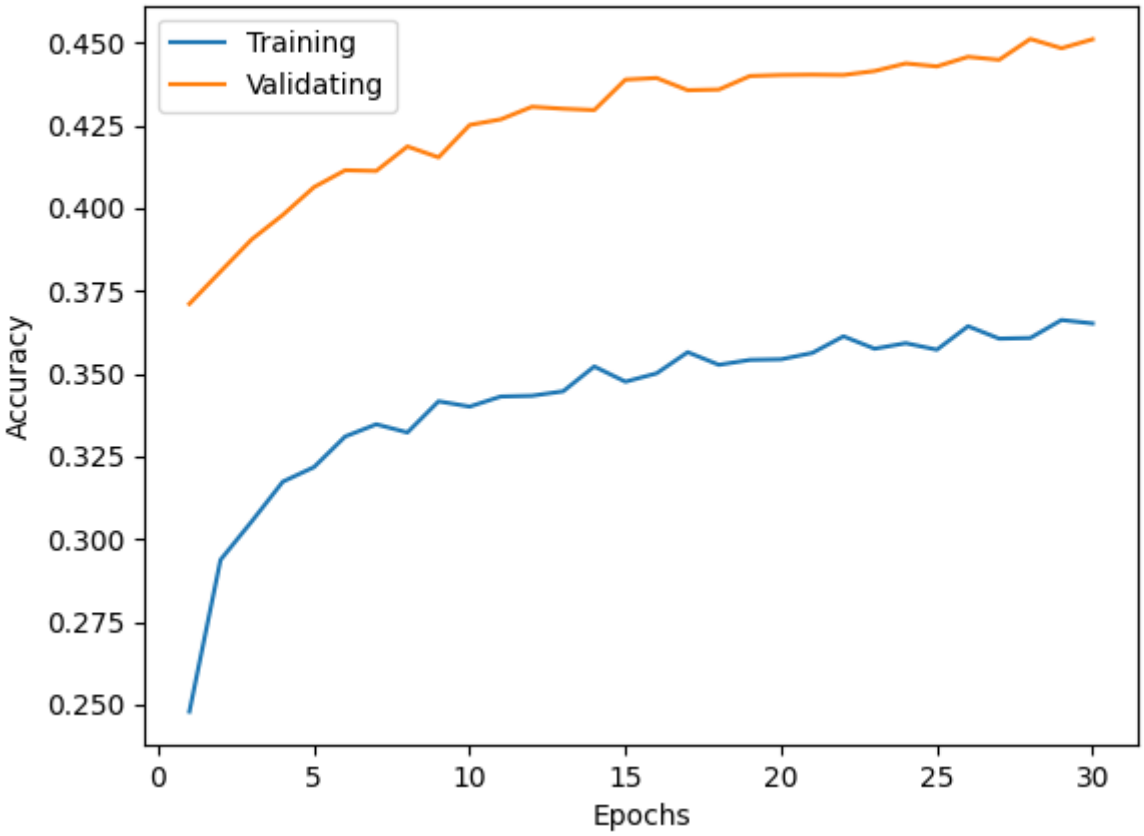


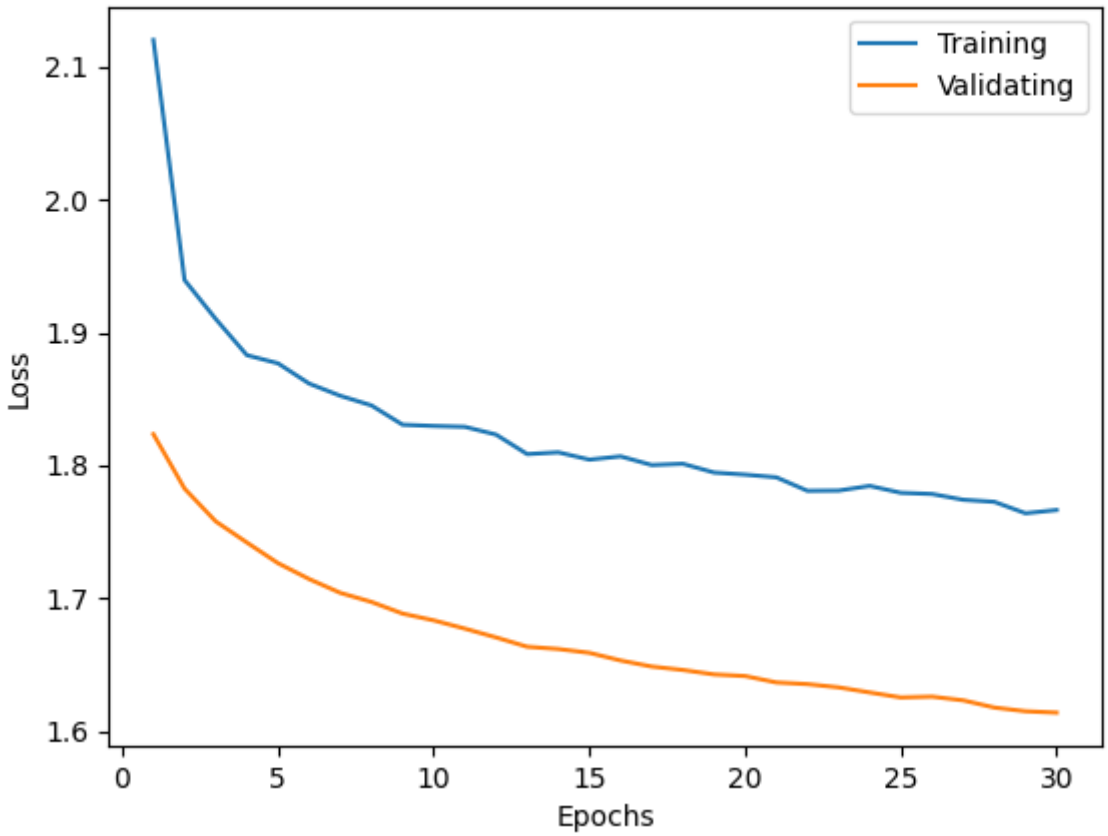
40% Dropout的MLP训练曲线





90% Dropout的MLP训练曲线


Training
Validating
Accuracy
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0
5
10
15
20
25
30
Epochs


Training
Validating
Loss
2.1
2.0
1.9
1.8
1.7
1.6
0
5
10
15
20
25
30
Epochs

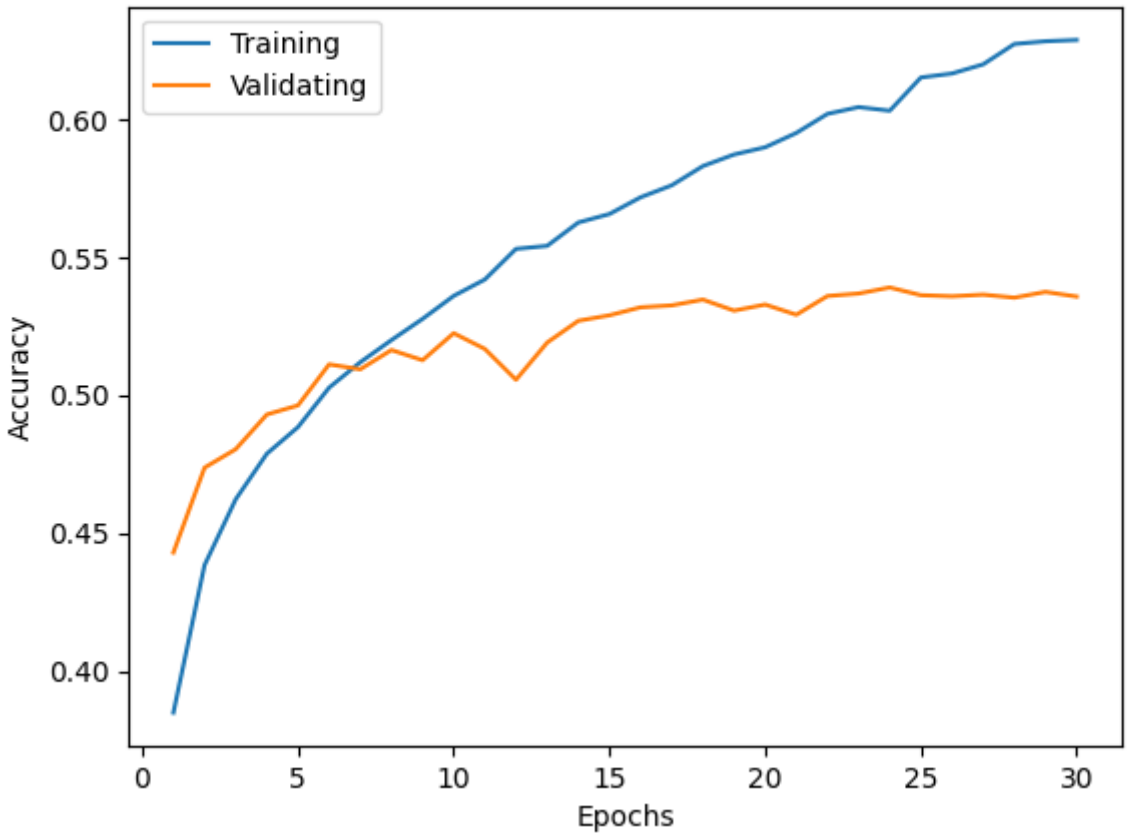
根据以上三种训练曲线可以清晰地看出，当没有Dropout时，模型在训练集上收敛得最快，能够最快拟合到训练集的分布上，但同时也会最快过拟合，导致在测试集上效果不佳；当Dropout Rate过大时，训练时的参数量过小，无法有效学习数据的分布，收敛最慢，严重欠拟合；当Dropout Rate合适时，模型以合适的速度收敛，能够在测试集上获得较好的表现。

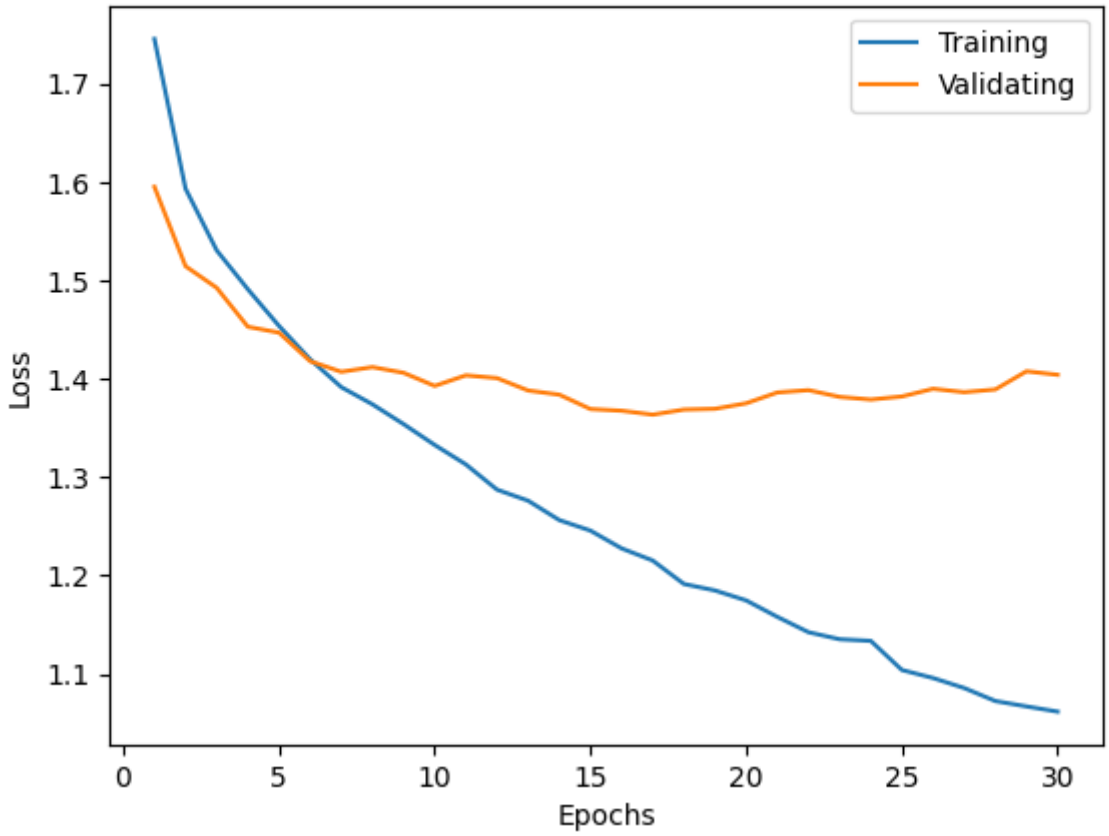
有无Batch Normalization的模型表现

在与最佳模型的超参数设定完全相同的情况下，我进行了无Batch Norm的实验，实验结果对比如下：

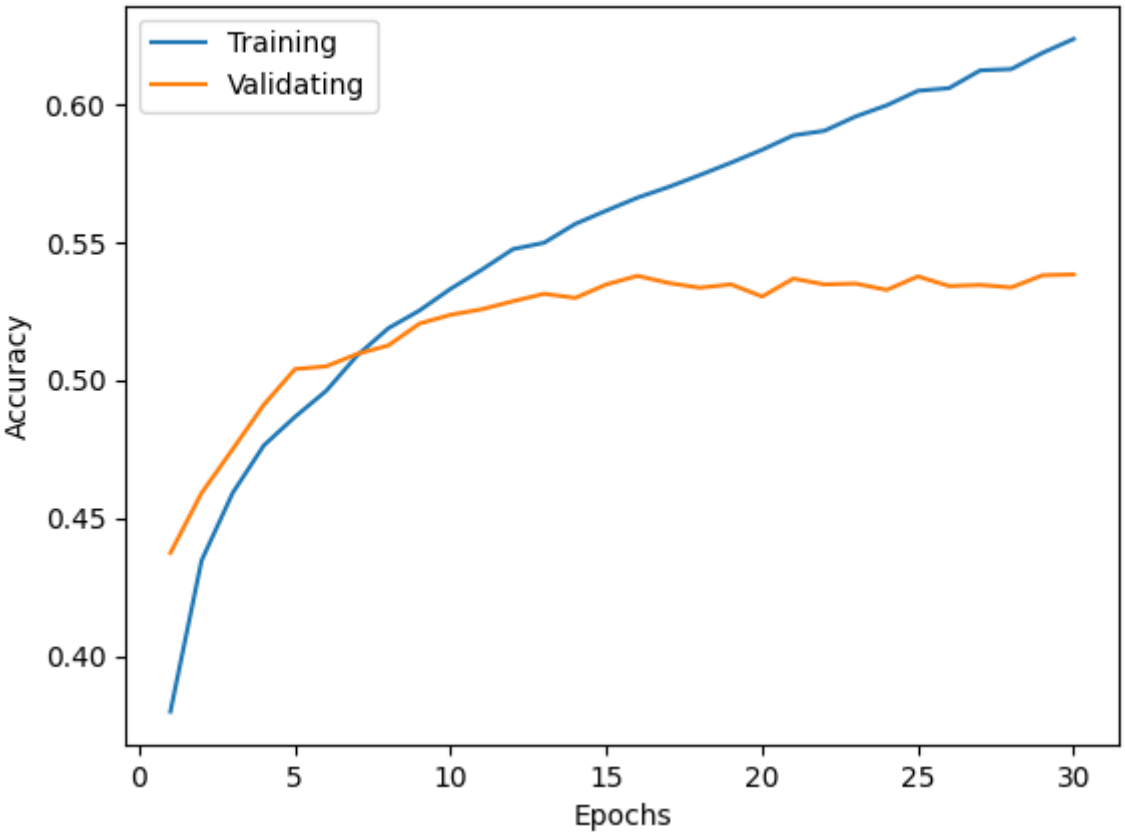
Model	**Test Accuracy of the Best Validation Model (%)**
MLP Drop 40% with Batch Normalization	**53.48**
MLP Drop 40% without Batch Normalization	53.07

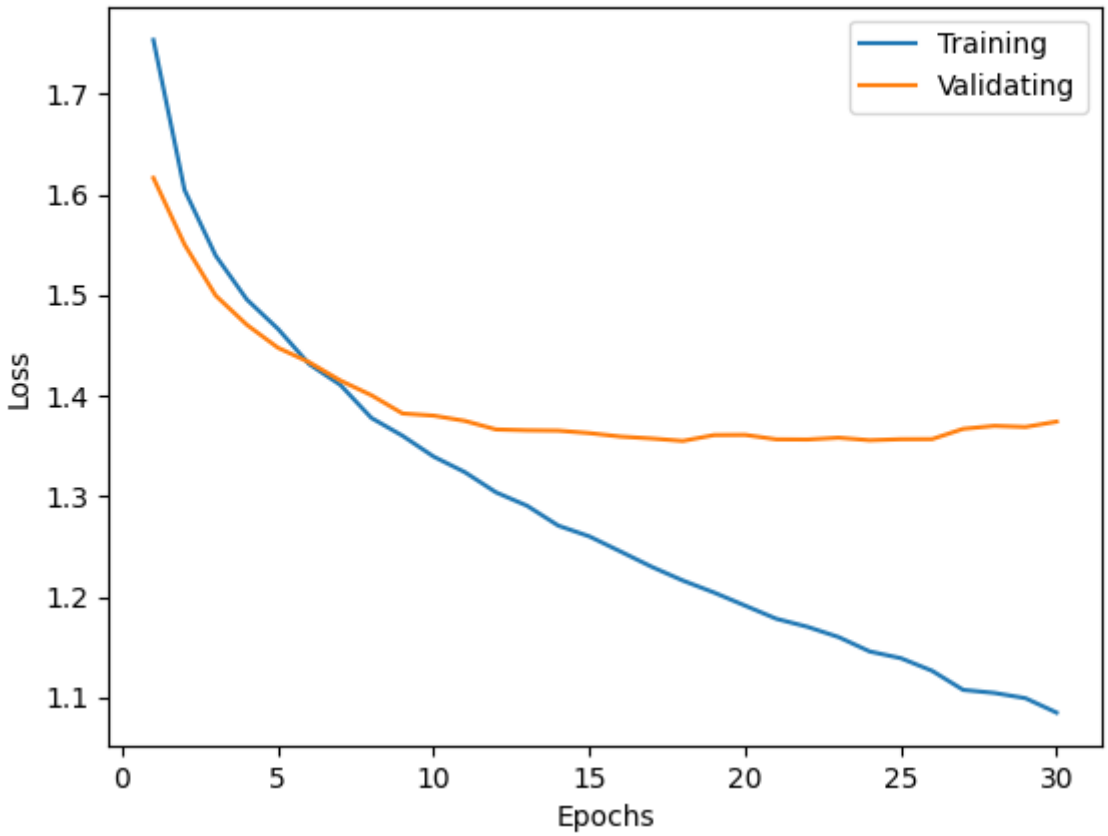
无 Batch Normalization 训练曲线





有 Batch Normalization 训练曲线


Training
Validating
Accuracy
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30
Epochs


Training
Validating
Loss
1.7
1.6
1.5
1.4
1.3
1.2
1.1
0
5
10
15
20
25
30
Epochs

可以明显的看出，加入Batch Normalization之后，训练曲线更加稳定，几乎没有抖动，这也使得我们能够使用更高的学习率进行优化，使得训练和收敛都加快。

CNN

网络结构

第一层卷积我采用的卷积核直径为5，Feature Map数为100；第二层卷积采用的卷积核直径为3，Feature Map 60。在卷积核尺寸的设计上，我认为在模型的底层应当捕捉更加整体上位的特征，因此采用较大的直径；在模型的高层特征维数已经十分凝练且维数较小，因此选用更小的卷积核来捕捉更细致的特征。在Feature Map数量的选择上，小的特征数会导致严重的欠拟合，过大的特征数使得模型能够捕捉到更多信息，但是也由于参数过多导致训练过慢和容易过拟合，我经调试和对比选择了100、60这个适中的Feature Map个数。

在池化层的尺寸选择方面，我没有作过多的尝试，就选用了尺寸和步长均为2的最大值池化。

```
Model(
  (conv_layers): Sequential(
    (0): Conv2d(3, 100, kernel_size=(5, 5), stride=(1, 1))
    (1): BatchNorm2d()
    (2): ReLU()
    (3): Dropout()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (5): Conv2d(100, 60, kernel_size=(3, 3), stride=(1, 1))
    (6): BatchNorm2d()
    (7): ReLU()
    (8): Dropout()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (fc): Linear(in_features=2160, out_features=10, bias=True)
  (loss): CrossEntropyLoss()
)
```

最佳超参数设定

经过几次尝试发现初始的`1e-3`确实是一个比较合适的学习率，因此没有调整。最佳的Dropout Rate是实验测试出来的。

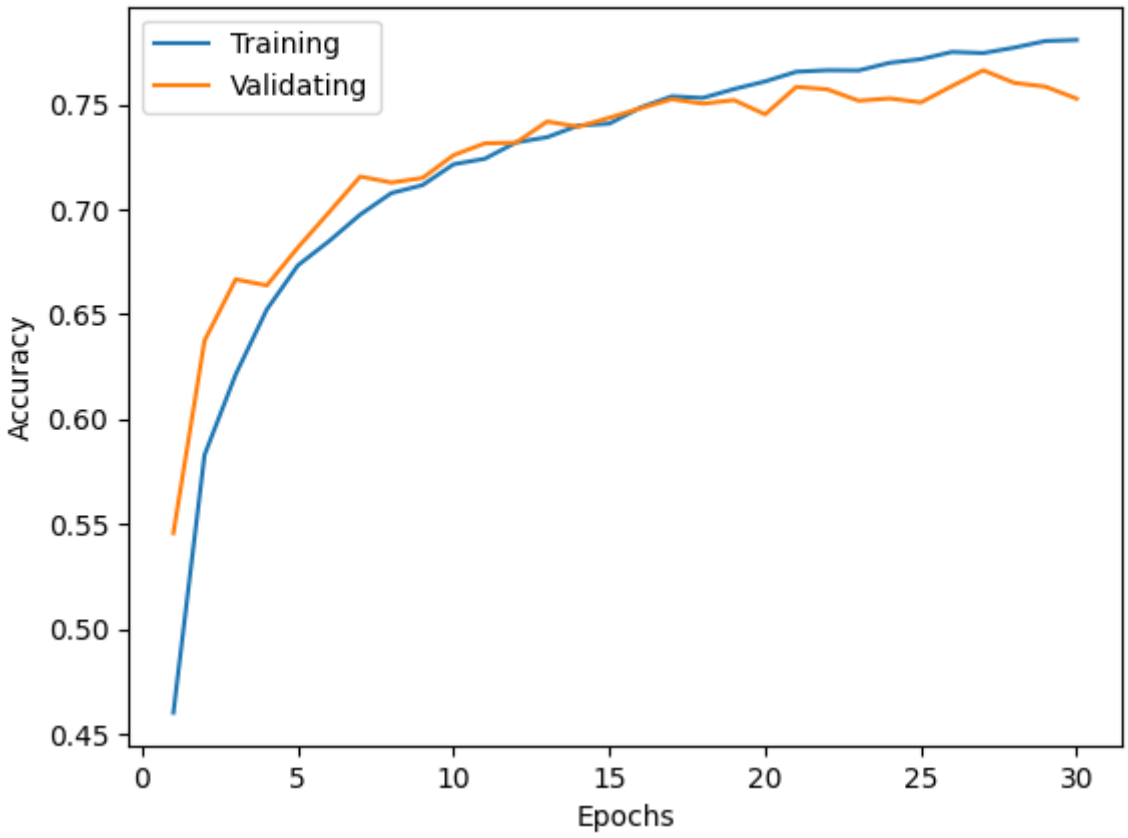
```
batch_size = 100
learning rate = 1e-3
batch_norm_momentum = 0.1
dropout_rate = 0.6
num_epochs = 30
```

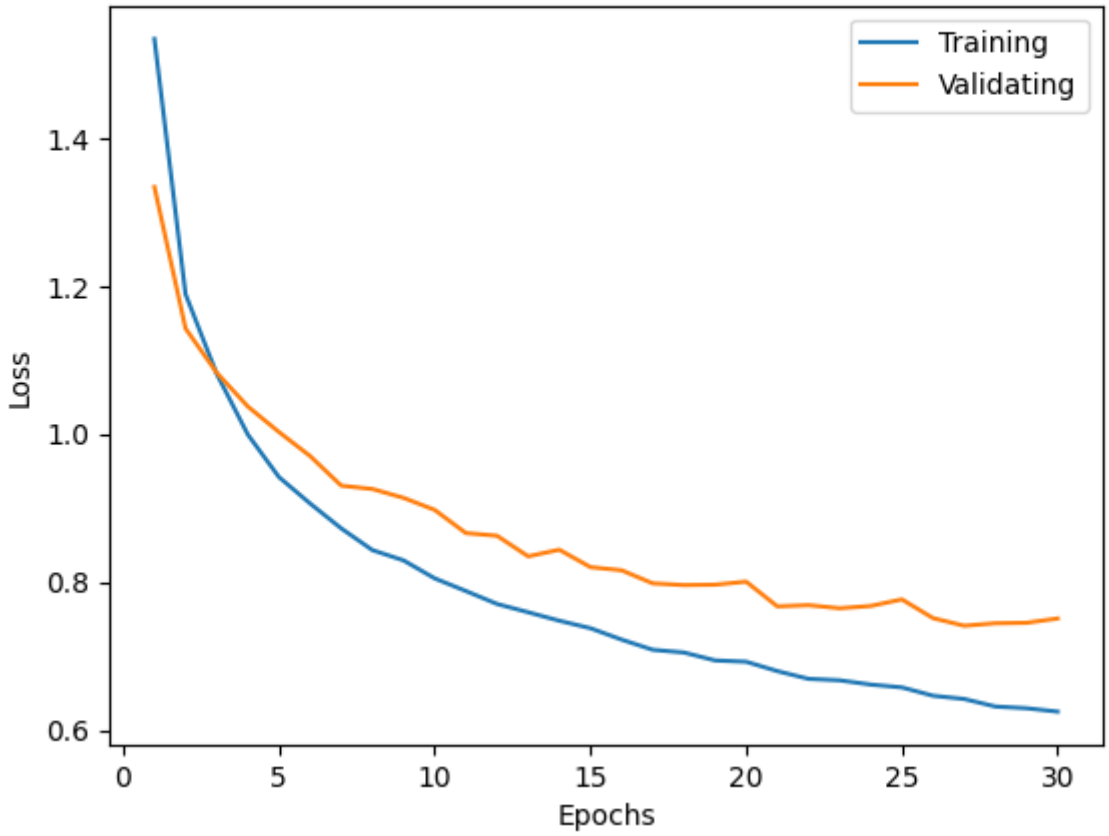
最佳模型的训练结果

在验证集上取得最佳准确率的模型在测试集上取得76.05%的准确率。

```
Epoch 30 of 30 took 5.5184783935546875s
  learning rate:                 0.001
  training loss:                 0.6250202737748622
  training accuracy:             0.7807499799132347
  validation loss:               0.7510512465238571
  validation accuracy:           0.7527999830245972
  best epoch:                    27
  best validation accuracy:      0.7662999790906906
  test loss:                     0.747600220143795
  test accuracy:                 0.7604999810457229
```

训练曲线：





不同Dropout Rate的模型表现

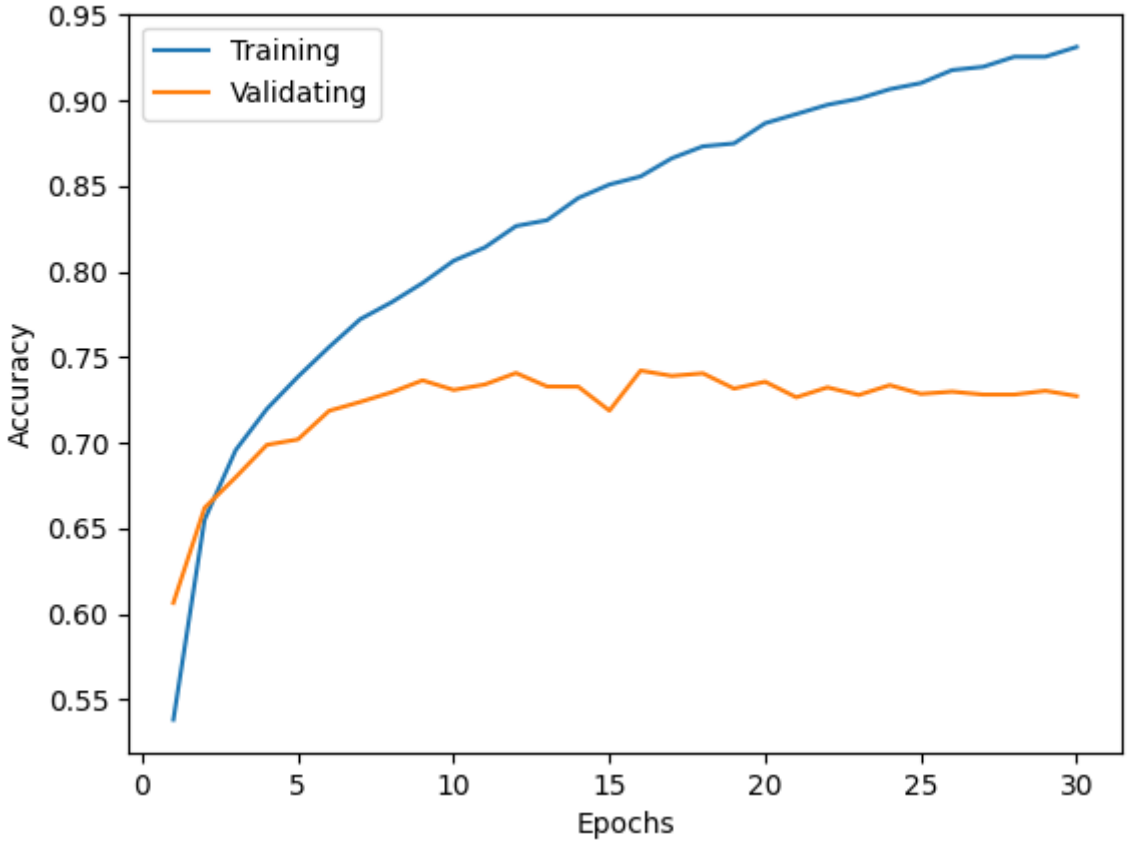
我在0~100%每隔10%取值，测试不同Dropout Rate情况下模型的表现，结果如下：

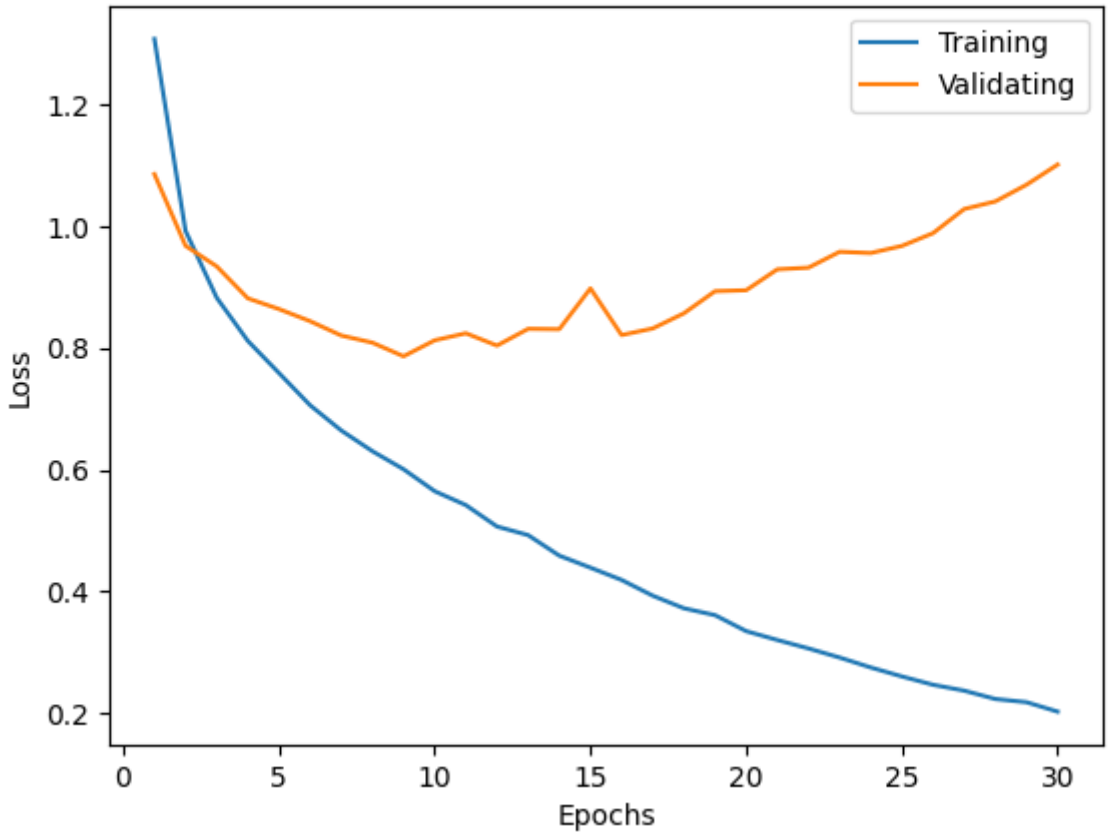
Dropout Rate (%)	Test Accuracy of the Best Validation Model (%)
0	73.77
10	73.90
20	74.63
30	74.93
40	75.39
50	75.73
60	**76.05**
70	74.04
80	72.78
90	63.00

可以明显看出模型最终在测试集上的表现随着Dropout Rate的提高呈现先增加后减小的趋势，这是可以理解的。当Dropout Rate很小乃至没有时，模型很容易过拟合到训练集上，导致在测试集上表现不佳；当Dropout

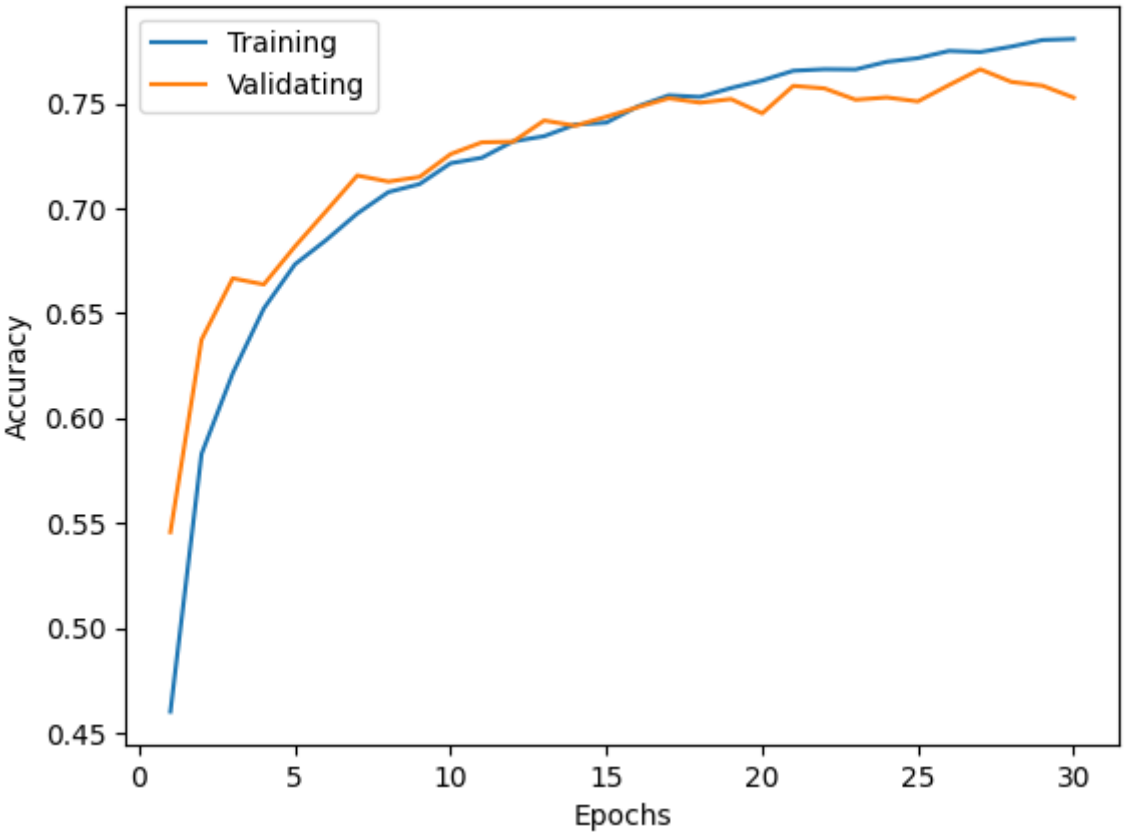
Rate过高时，模型中的参数在训练时不能被充分训练，相当于训练过程中模型参数量过小，不能有效学习数据集的分布。

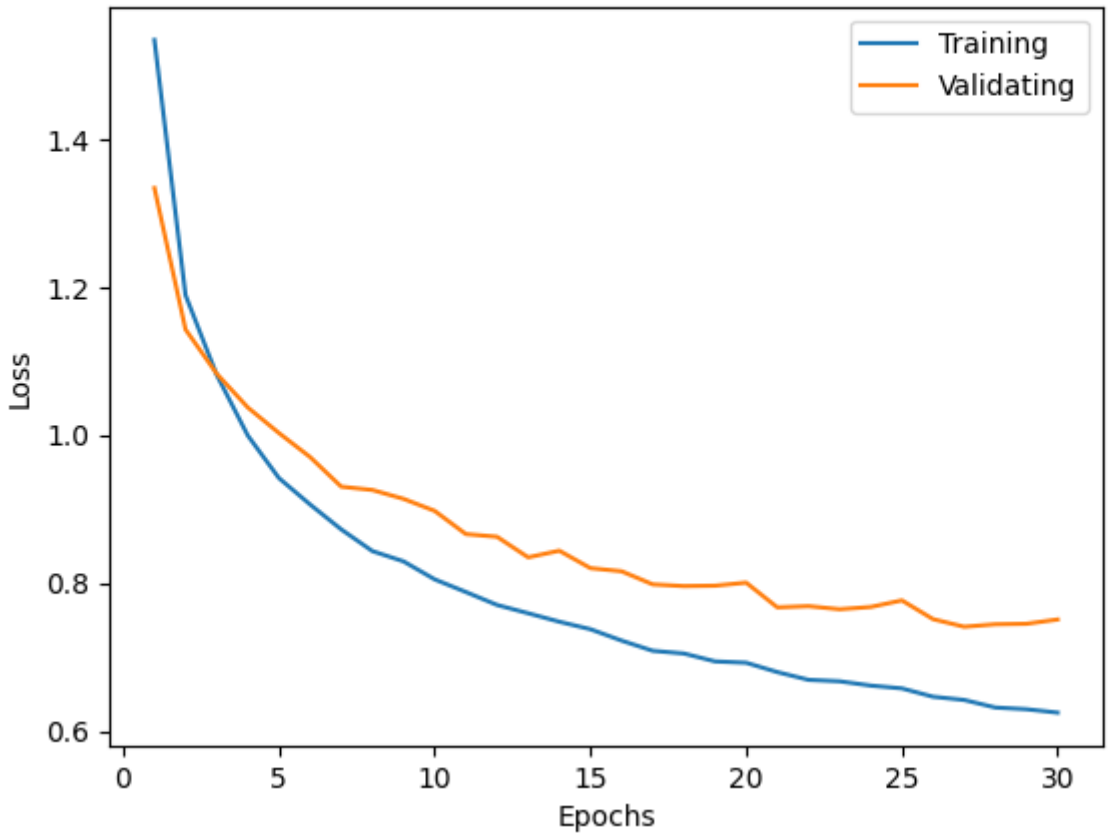
无Dropout的CNN训练曲线



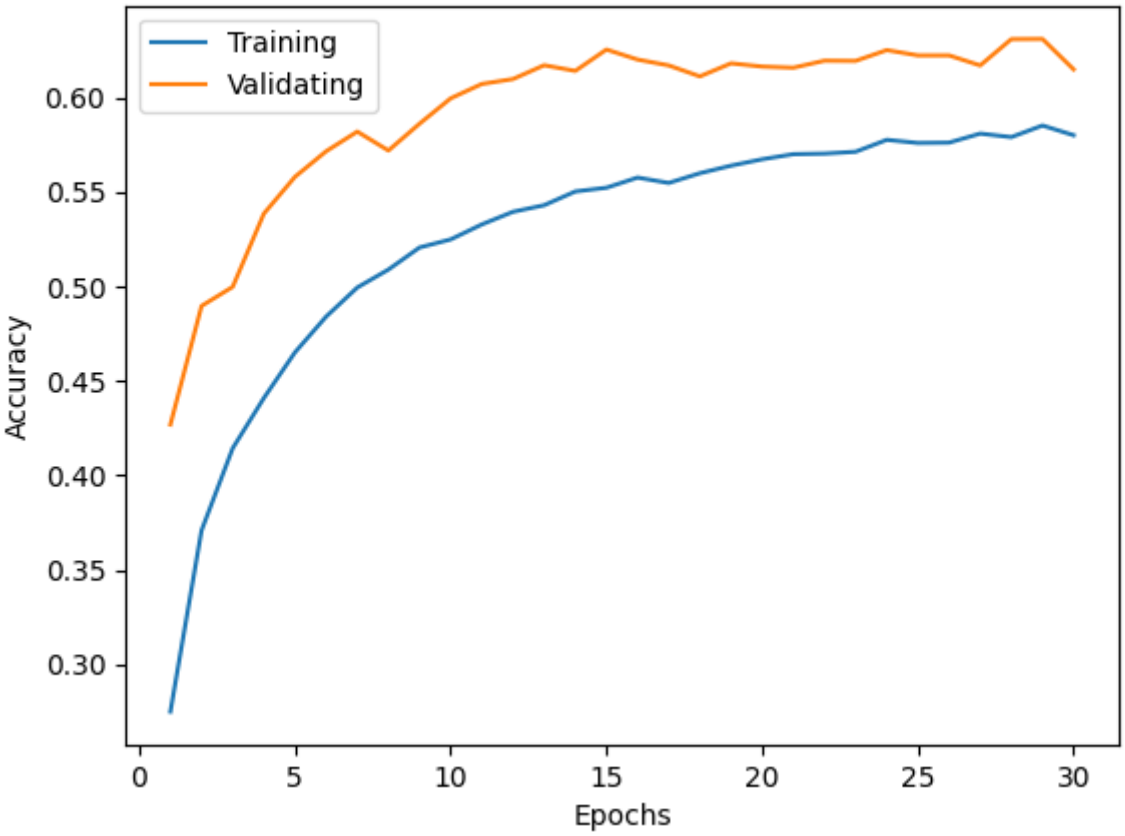


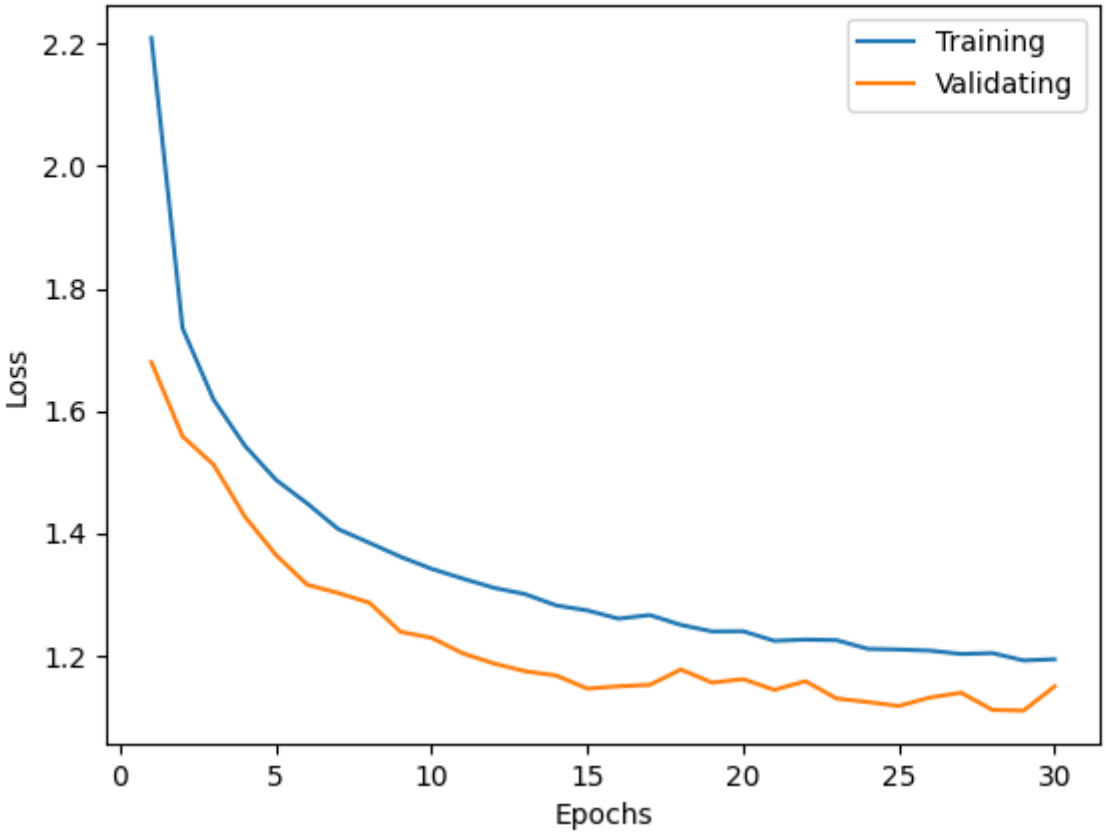
60% Dropout的CNN训练曲线





90% Dropout的CNN训练曲线


Training
Validating
Accuracy
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
30
Epochs


Training
Validating
Loss
2.2
2.0
1.8
1.6
1.4
1.2
0
5
10
15
20
25
30
Epochs

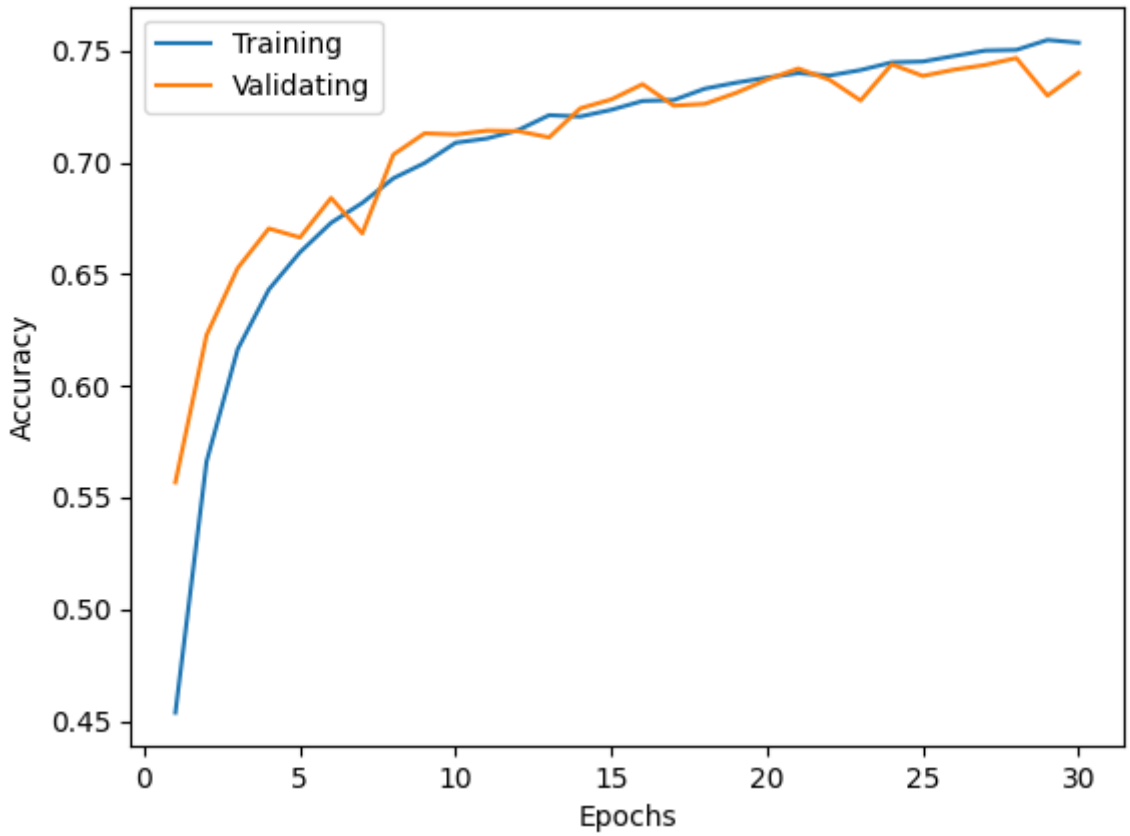
根据以上三种训练曲线可以清晰地看出，当没有Dropout时，模型在训练集上收敛得最快，能够最快拟合到训练集的分布上，但同时也会最快过拟合，导致在测试集上效果不佳；当Dropout Rate过大时，训练时的参数量过小，无法有效学习数据的分布，收敛最慢，严重欠拟合；当Dropout Rate合适时，模型以合适的速度收敛，能够在测试集上获得较好的表现。

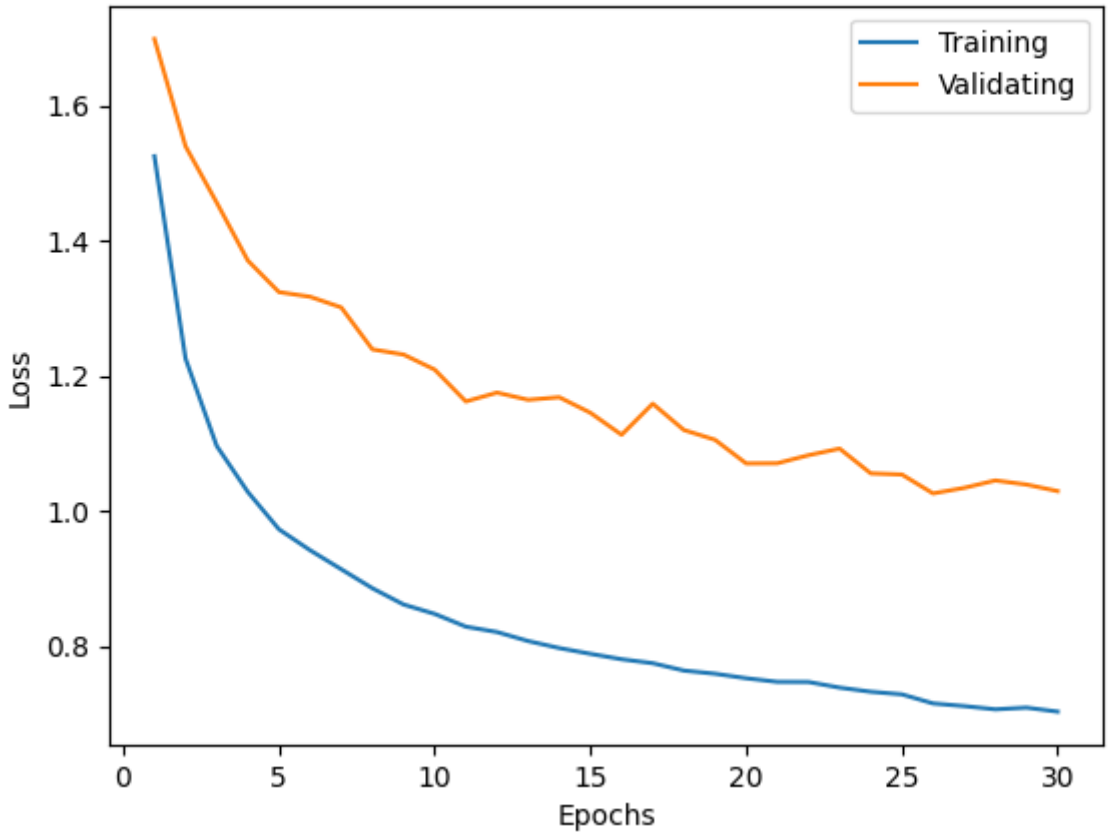
有无Batch Normalization的模型表现

在与最佳模型的超参数设定完全相同的情况下，我进行了无Batch Norm的实验，实验结果对比如下：

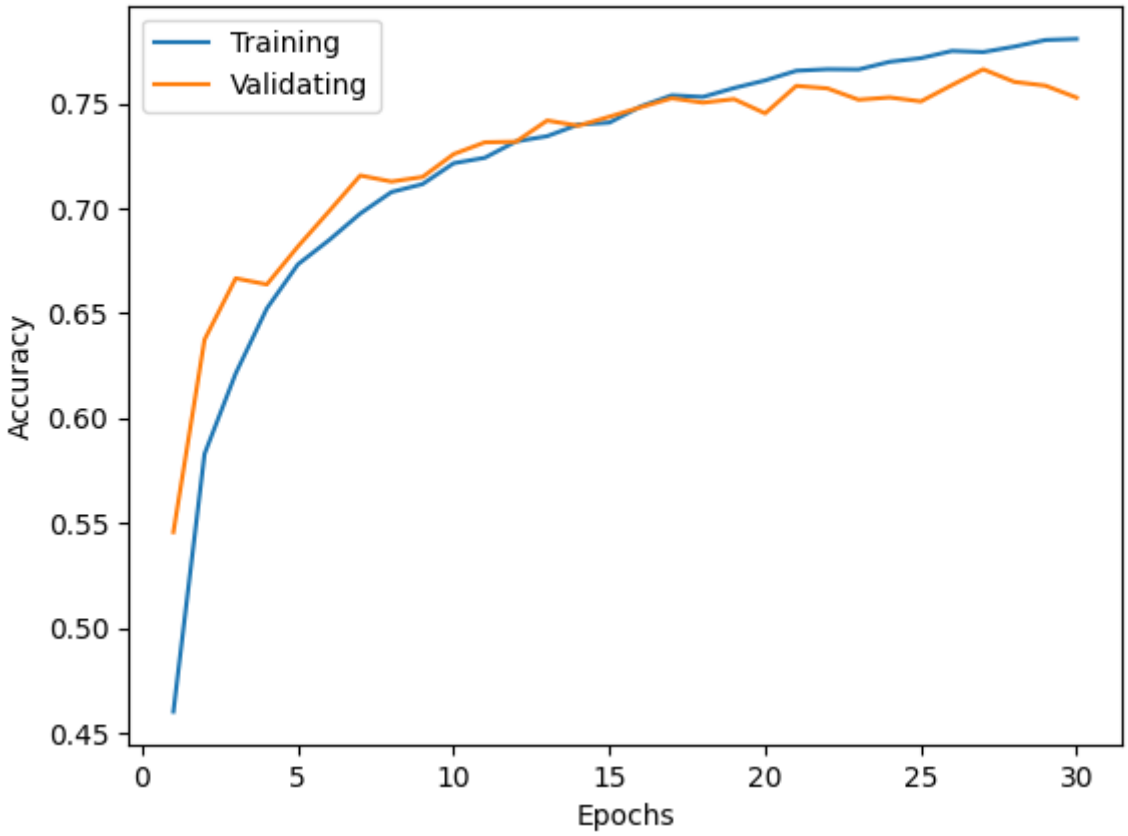
Model	**Test Accuracy of the Best Validation Model (%)**
CNN Drop 60% with Batch Norm	**76.05**
CNN Drop 60% without Batch Norm	73.60

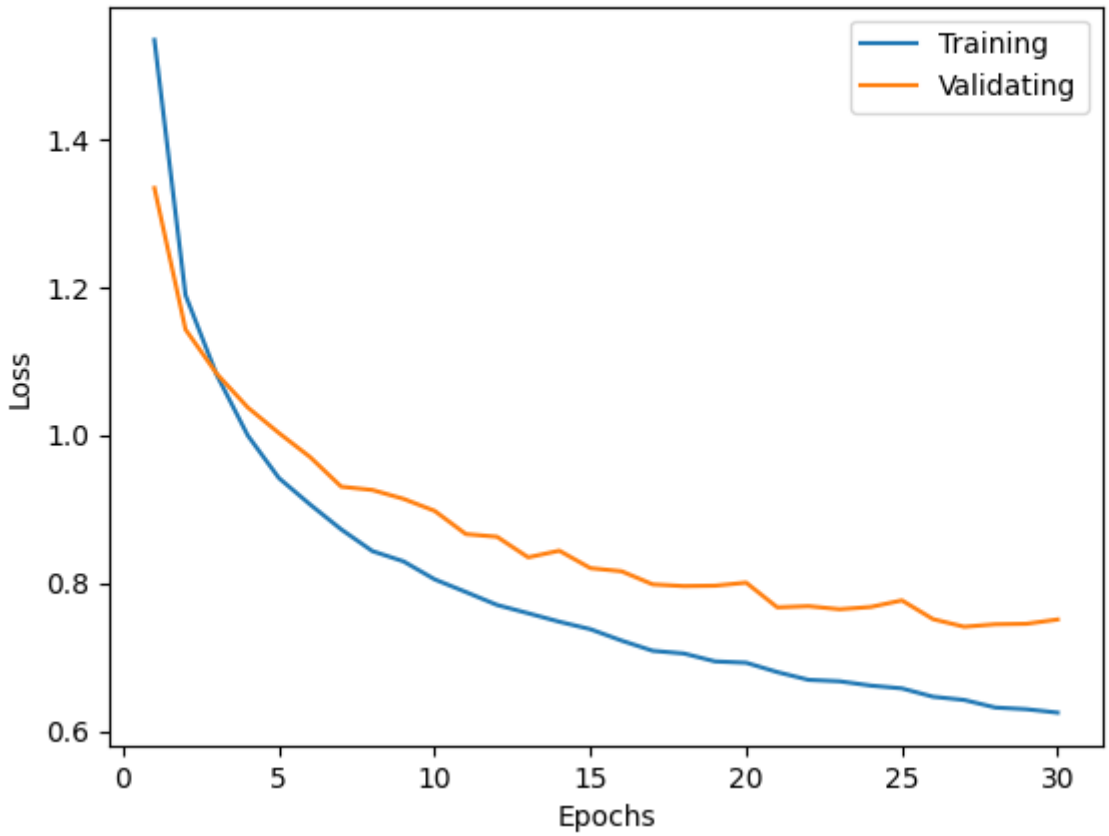
无 Batch Normalization 训练曲线





有 Batch Normalization 训练曲线


Training
Validating
Accuracy
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
5
10
15
20
25
30
Epochs


Training
Validating
Loss
1.4
1.2
1.0
0.8
0.6
0
5
10
15
20
25
30
Epochs

可以明显看出，有Batch Normalization时模型收敛显然更加稳定，曲线抖动减小，收敛变快。并且由于在测试数据上也根据训练数据的参数估计进行归一化，一定程度上Batch Normalization还能提高模型的泛化能力，这从有Batch Normalization模型过拟合现象小于无BN的模型可以得到。

思考

`self.training` 是如何工作的？为什么训练和测试要有所不同？

调用`model.train()`时，`self.training = True`。

调用`model.eval()`时，`self.training = False`。

在模型中用到`Dropout`、`BatchNorm`这种训练和测试场景行为不同的层时，这个属性就十分重要，帮助这些层做不同的行为。

训练曲线和验证曲线为什么不同？这种不同如何帮助你调整超参数？

训练曲线和验证曲线不同的根本原因在于训练数据集和验证数据集的分布不同，这种分布上的差异可能是我们不在意的“次要差异”，这种不同就是我们所说的“过拟合”现象，即模型学习到了训练集上某些不重要的统计痕迹；这种分布上的的差异也有可能是根本上的“主要差异”，这种情况表示我们使用的训练数据集本身就是有偏的，不能很好表达测试数据的分布，此时通过优化模型就很难奏效了。

但一般情况来说都是过拟合现象，表现于训练误差不断减小，但验证误差先减小后增大，呈现“V”字形。这种不同可以告知我们模型何时发生何种程度的过拟合现象，促使我们采取一系列提高模型泛化性能的措施，比如提高Dropout Rate，也能指导我们调整学习率、训练轮数等参数。

MLP与CNN效果差异的原因？

通过这次对比实验能够清晰看出MLP在图像分类上的表现明显不如CNN，这和二者的结构差异有紧密的联系。

MLP的缺点

- MLP由于所有神经元之间都是紧密连接的，因此当图片和类别增大时，其参数量将很快增大，这导致其难以训练且容易过拟合，泛化能力较差。
- MLP不具有平移不变性，比如说训练集中的猫的图片，猫如果都出现在左上角，那么测试集中一个出现在右下角的猫的图片对于MLP来说就很难分类正确，这不符合我们对图片分类问题平移不变性的预期。
- MLP损失了图片像素之间空间临近性的信息。在图片中往往相邻的像素比距离较远的像素之间更容易有语义上的联系，但是MLP输入时将所有像素都压缩成一个一维向量，因此图像二维上的临近性丢失，这其实是损失了很大一部分的信息。

CNN的改进

根据MLP的缺点，人们引入了**空间平移不变性**这一归纳偏置，设计出了CNN，CNN在处理图像这类具有空间关系的数据具有以下优点：

- CNN中的处理图片中不同位置卷积核是共享参数的，这不仅引入了空间平移不变性，并且极大减小了参数中的冗余，使得网络更容易收敛和训练，并且网络也可以更深，能够更好的捕捉高层抽象的语义信息。
- CNN是**稀疏连接**的，这不仅减小了参数量，并且引入了一个归纳偏置：邻近的像素比远离的像素关系更密切。这使得我们的网络更能收敛到我们所希望的解上，而不是拟合到训练数据中奇怪的统计特征。

- CNN的输入是一个二维矩阵，输出也是一个二维矩阵，这保留了图片数据中的空间临近性信息。