计14 汪隽立 2021012957

ANN-hw2

Explain how `self.training` work. Why should training and testing be different?

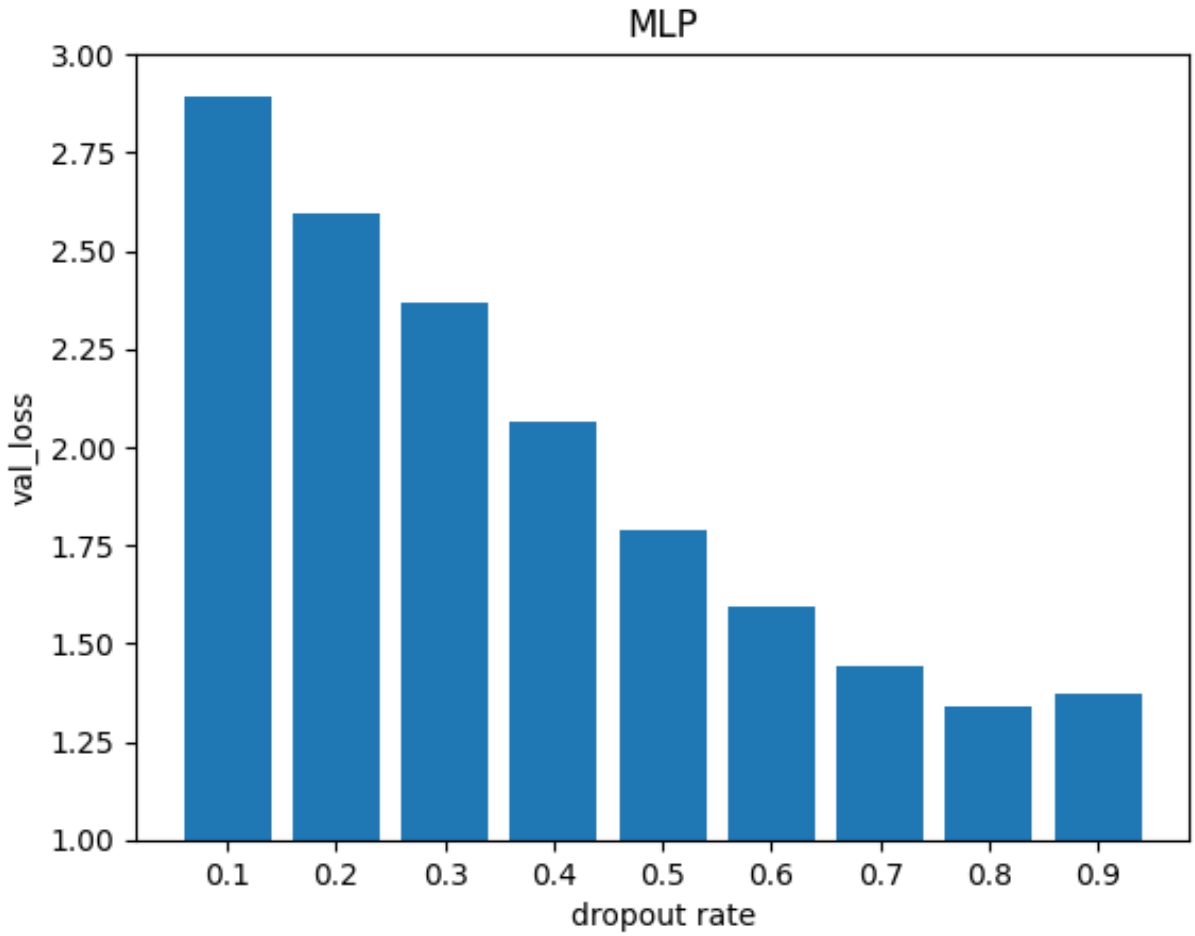
- 在`nn.module`框架下，`model.train()`会将`model.training`设置为`True`，而`model.eval()`不会，用以区分与验证、测试的过程。
- 在训练时，我们需要对模型中的参数进行反向传播的训练；而在验证或测试时，我们实际上是拿训练好的参数作用在数据上，无需再通过反向传播更新模型的参数。
- 除此以外，`batchNorm`操作不需要在测试过程中更新平均值和方差（我们是在训练过程中估计出均值和方差，然后作用在测试集上）；`dropout`操作也不需要在测试过程中使用，因为这只是帮助更好训练的方法。

MLP

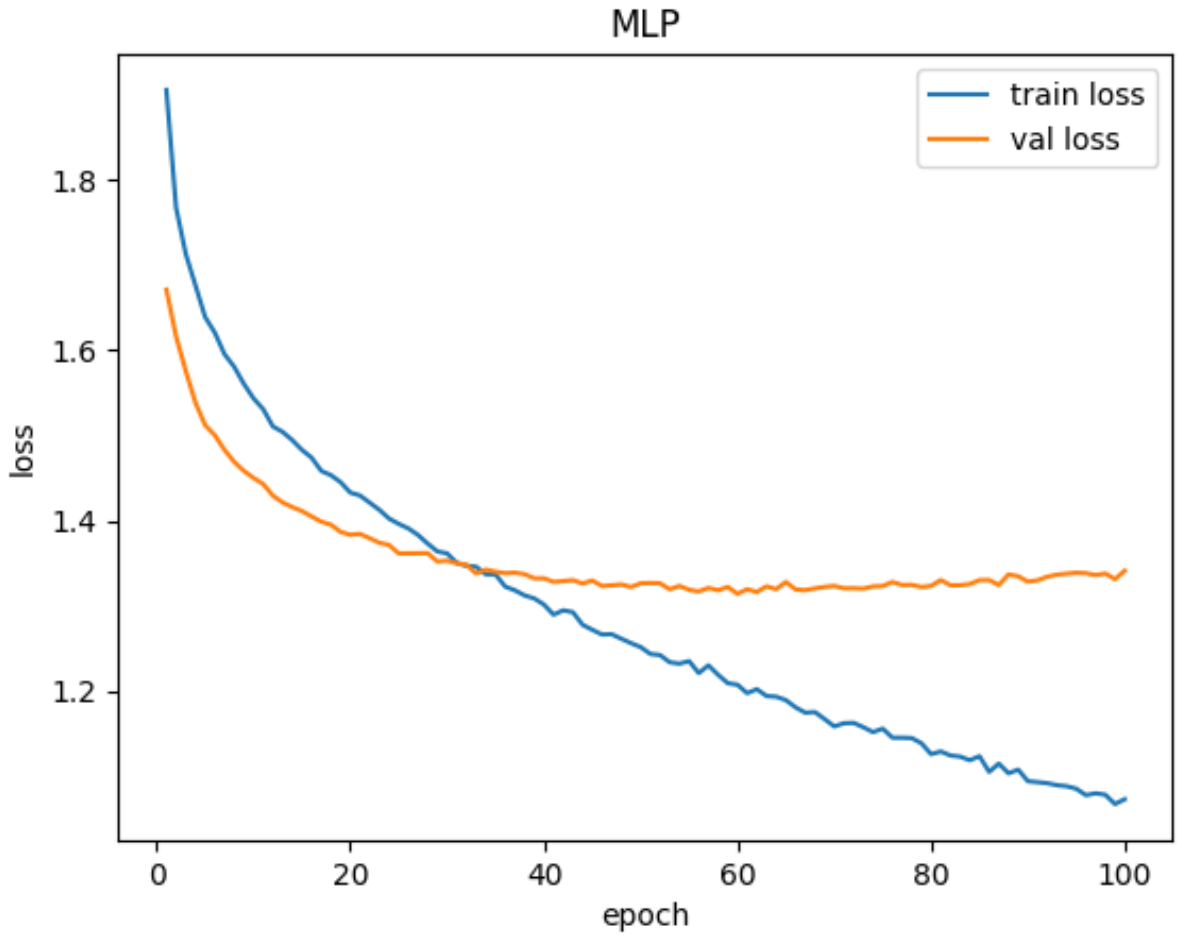
默认的参数列表：

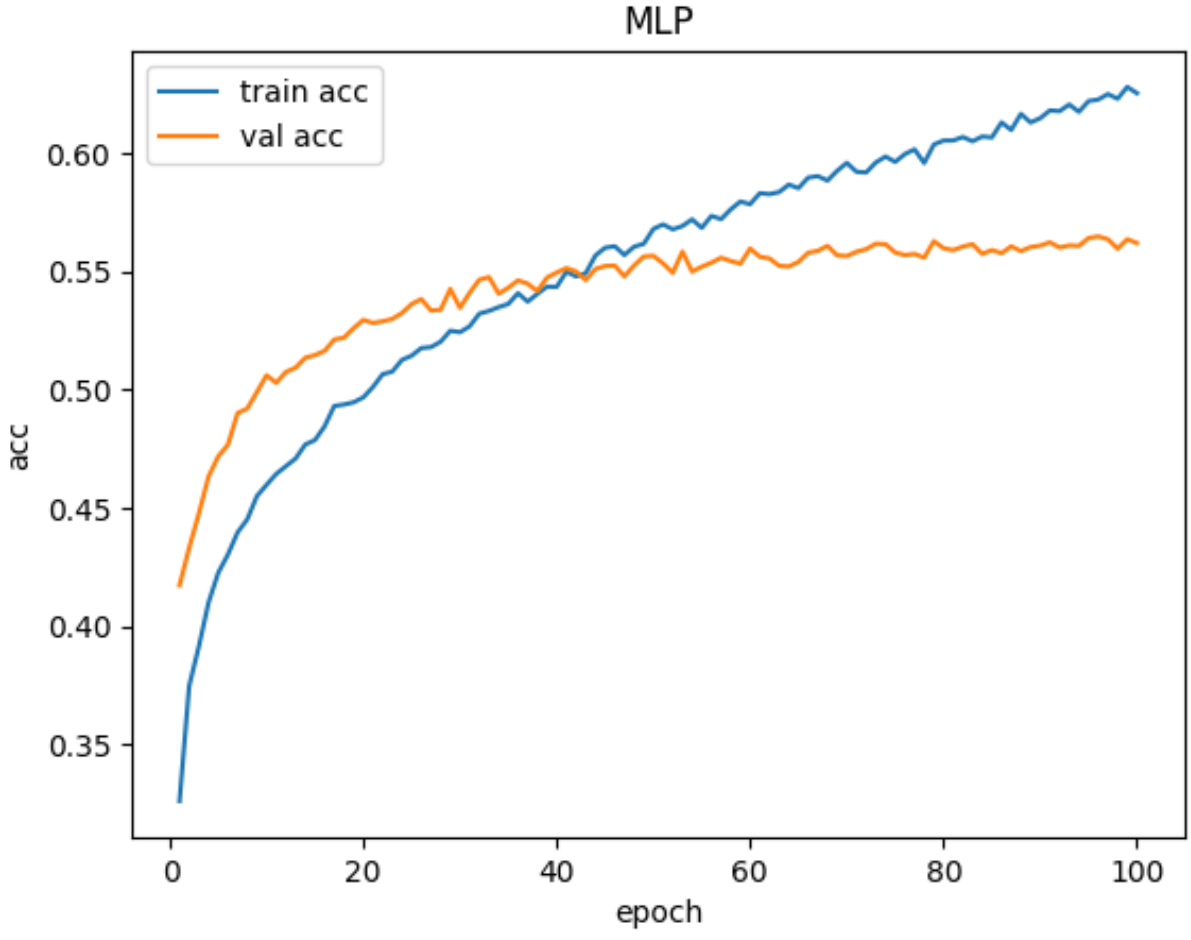
参数	值
batch_size	100
num_epoches	100
learning_rate	1e-3
hidden_layer	1024

对`dropout = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]`做了实验，得到在**验证集**上的loss为：



可以看出：在MLP实验中，当`dropout_rate`为0.8时，模型在验证集的表现最好。下面画出`dropout_rate=0.8`时的损失与准确率变化：


MLP
train loss
val loss
1.8
1.6
1.4
1.2
loss
0
20
40
60
80
100
epoch

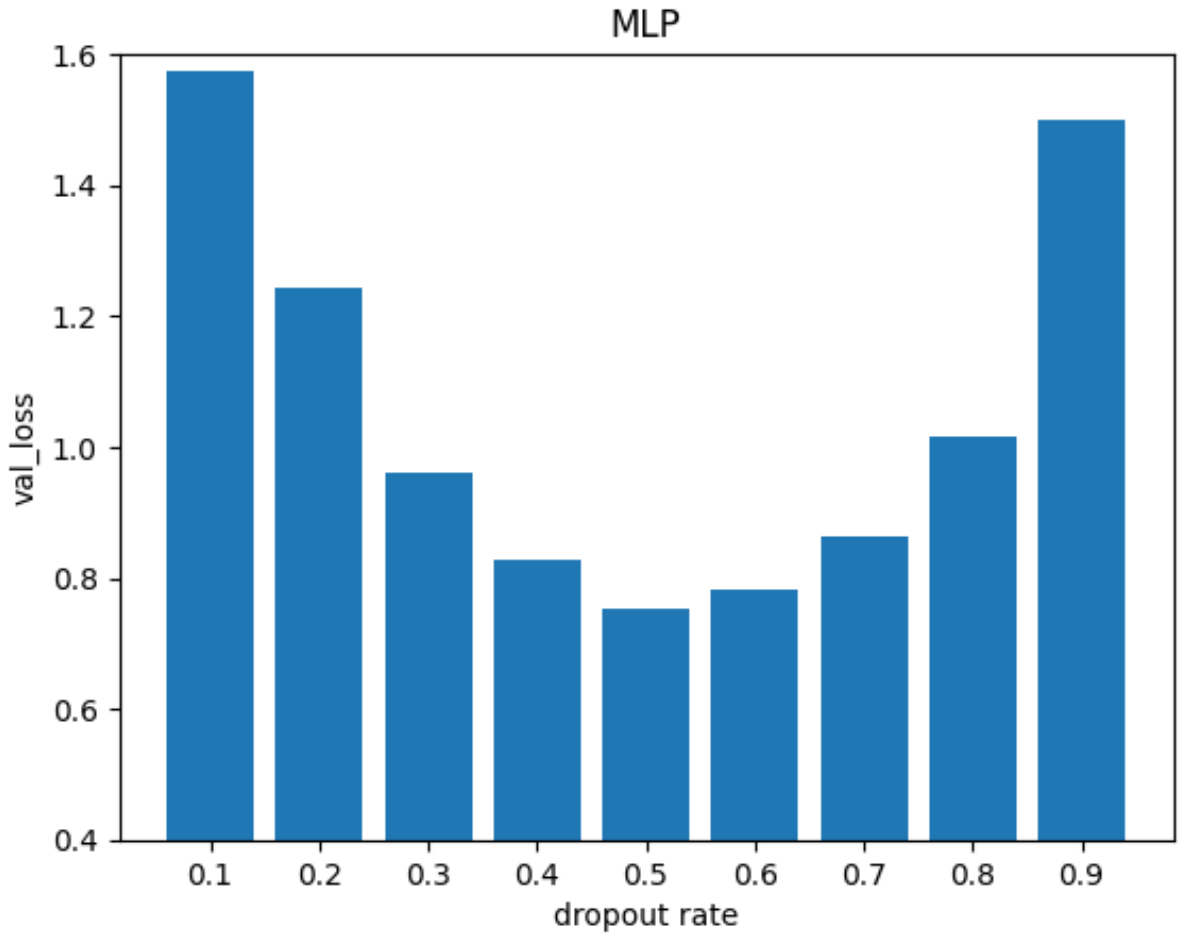

MLP
train acc
val acc
0.60
0.55
0.50
0.45
0.40
0.35
acc
0
20
40
60
80
100
epoch

CNN

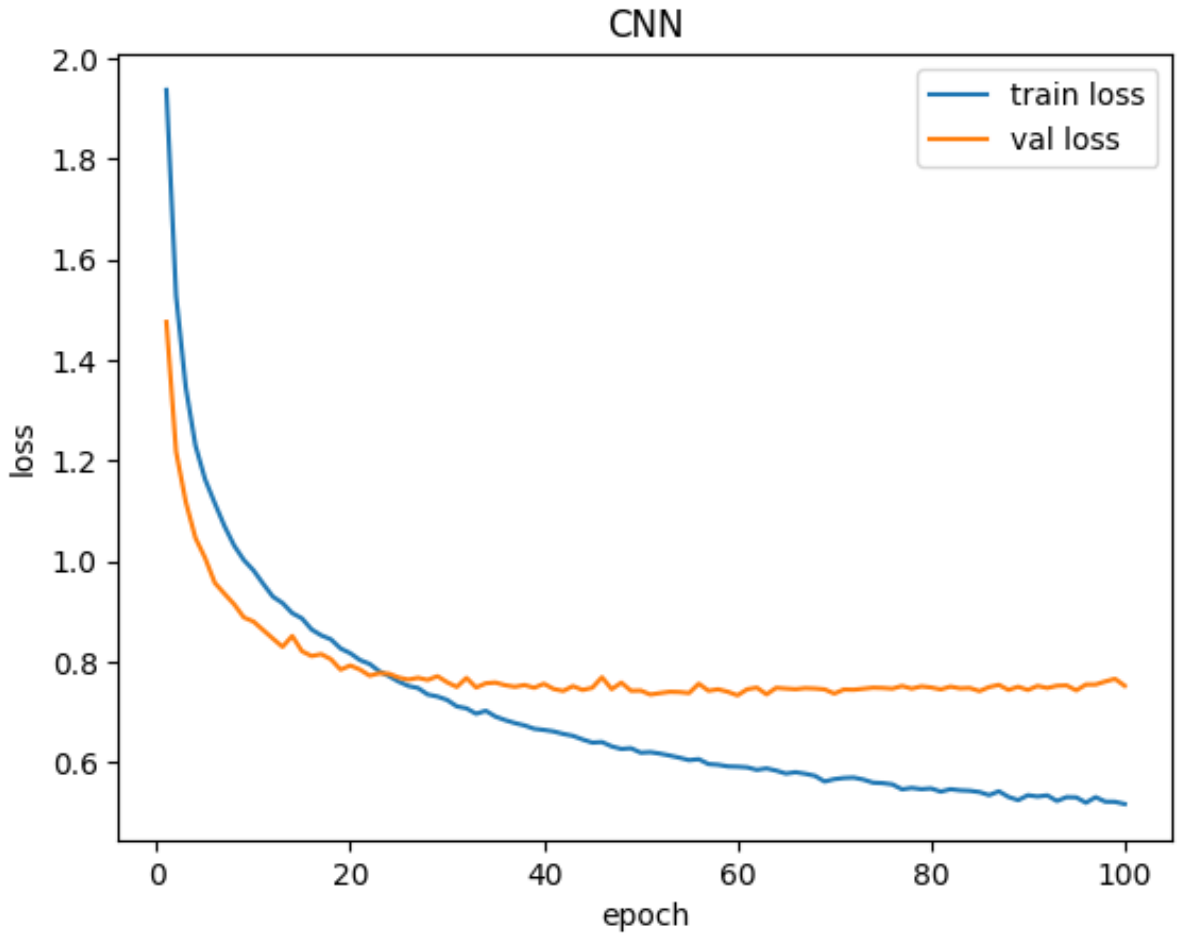
默认的参数列表：

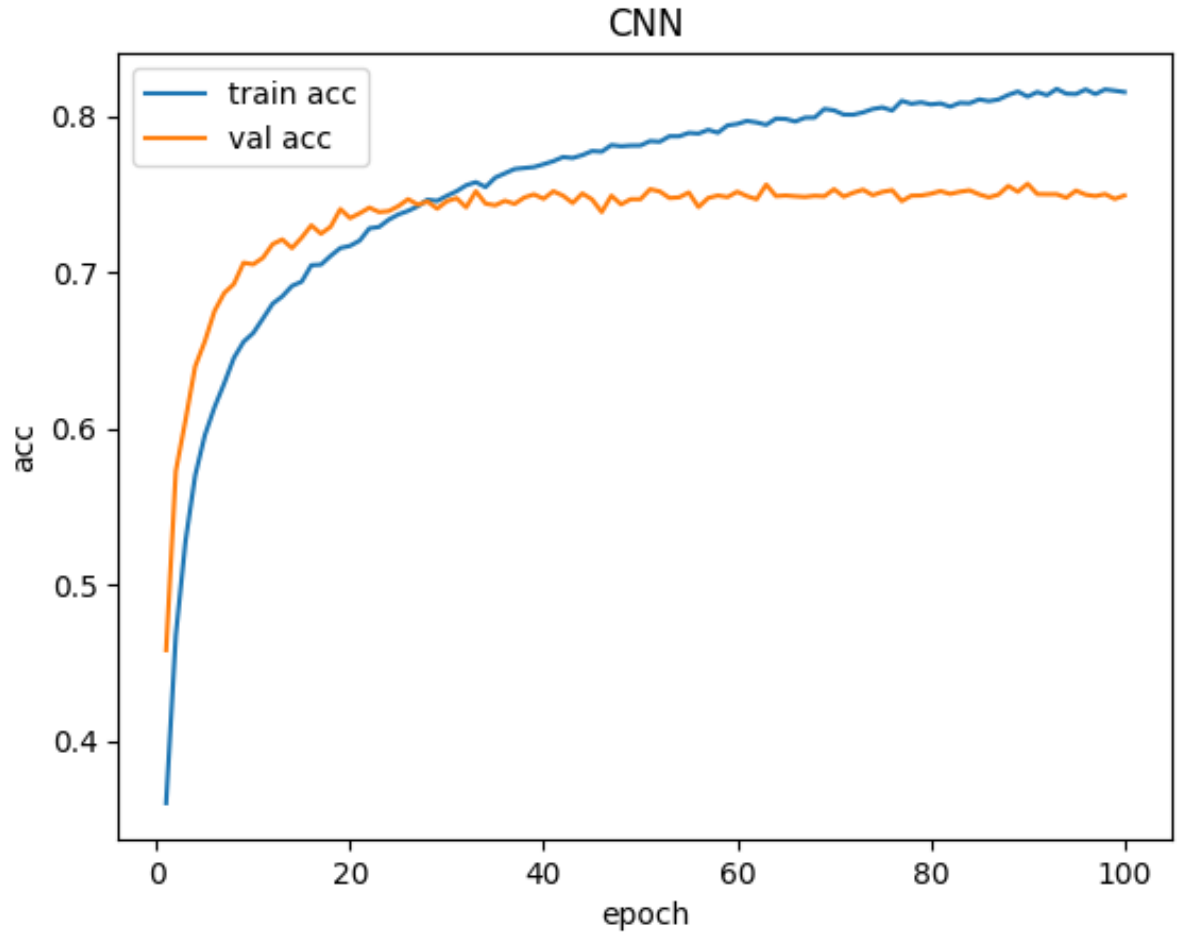
参数	值
batch_size	100
channels	[64, 128]
kernel_size	[3, 5], padding = [1, 2]
maxpool_size	[2, 2], padding = [1, 1]

对`dropout = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]`做了实验，得到在**验证集**上的loss为：



可以看出：在CNN实验中，当`dropout_rate=0.5`时，模型在验证集上的表现最好。下面画出`dropout=0.5`时的损失与准确率变化：


CNN
train loss
val loss
loss
epoch
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0
20
40
60
80
100

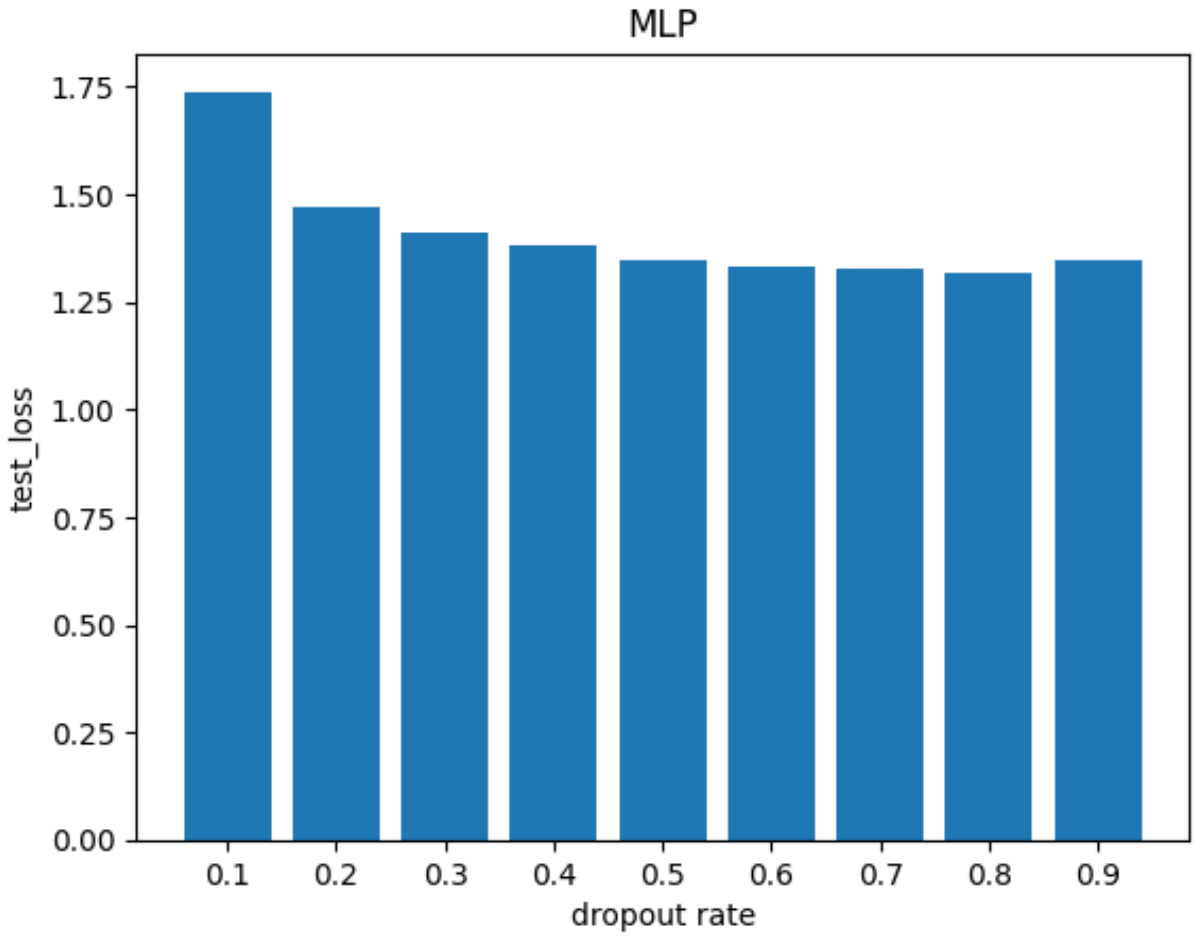

CNN
train acc
val acc
acc
epoch
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100

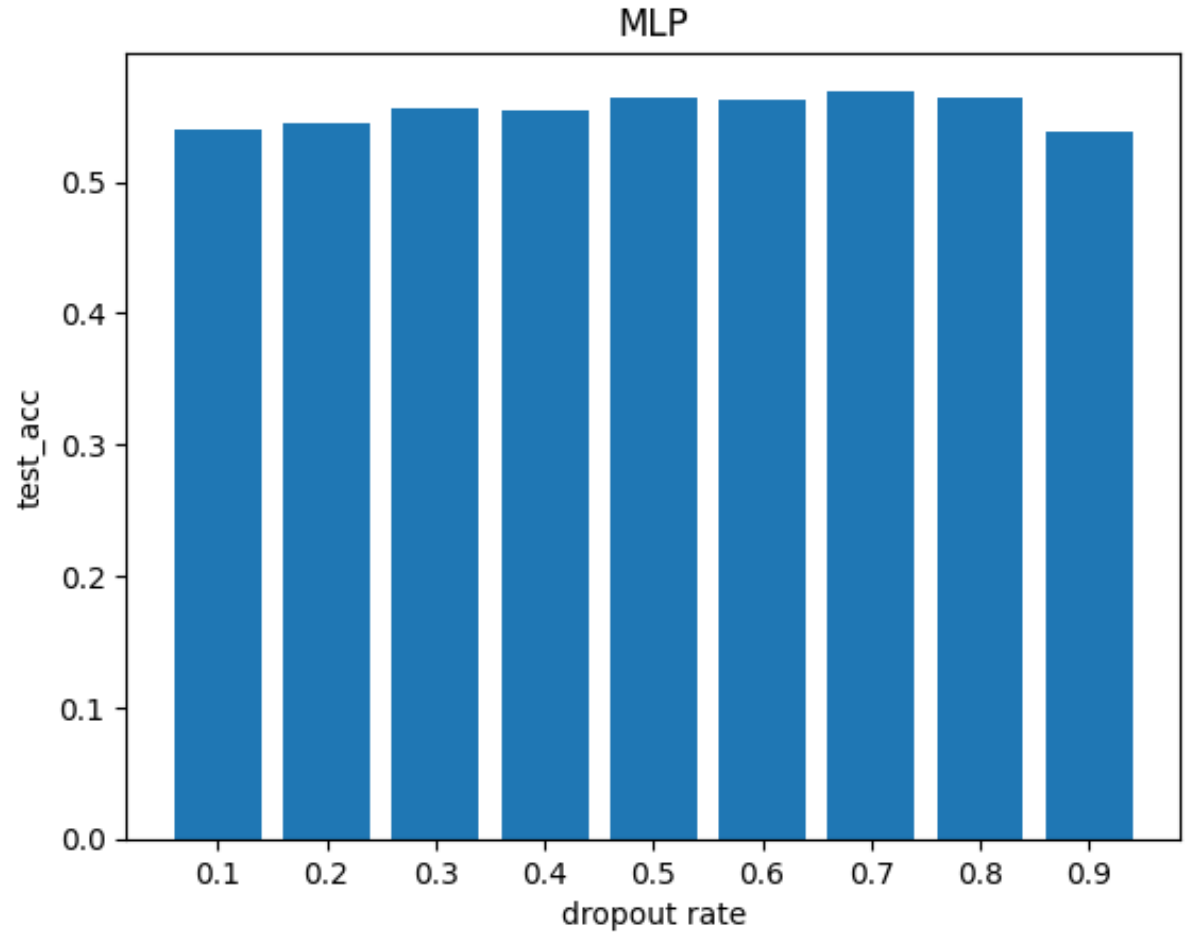
Explain why training loss and validation loss are different. How does the difference help you tuning hyper-parameters?

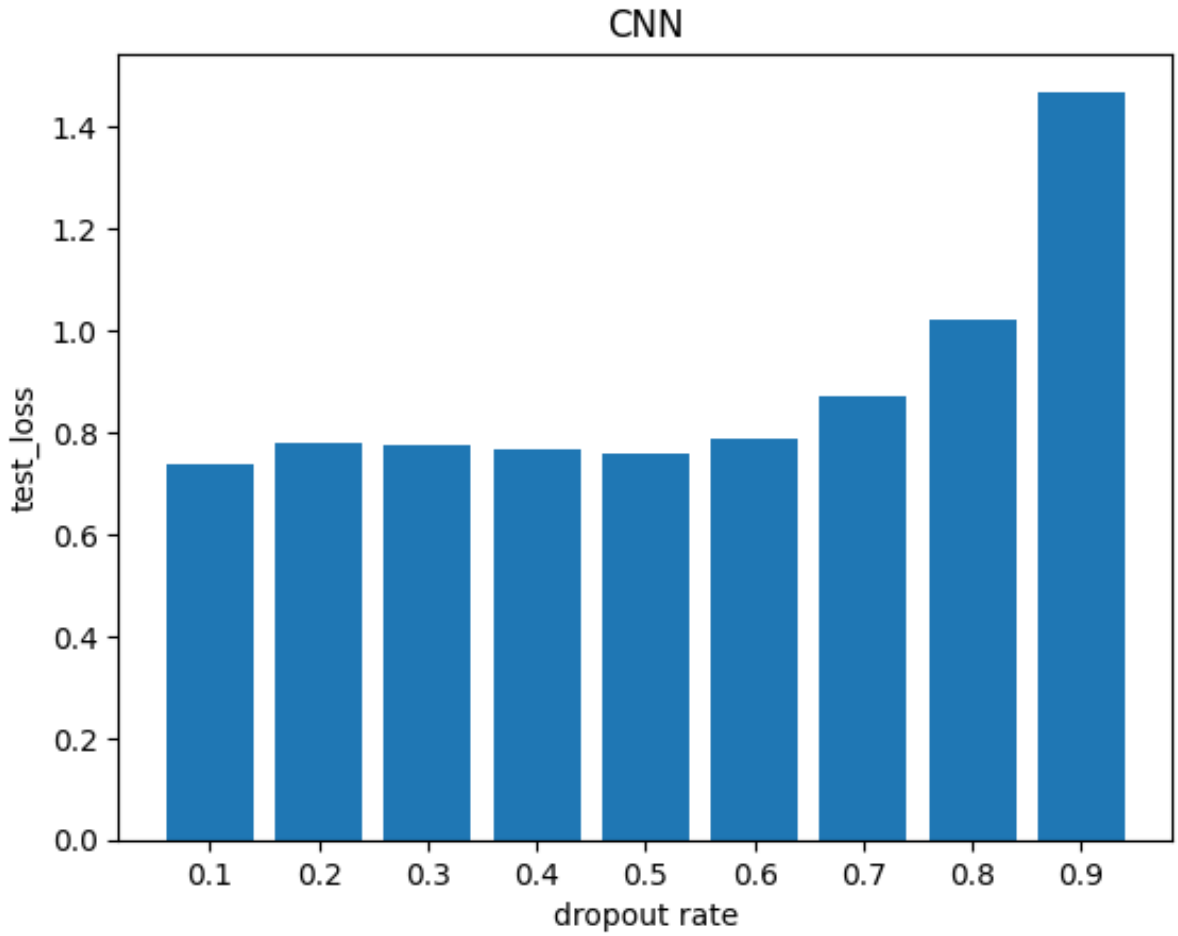
- 首先，训练集和验证集本身就是两个不同的集合。同一个模型作用在他们身上产生的损失自然不同。
- 其次，在训练过程中，我们利用的是训练集来进行反向传播，所以直接减少的是训练集上的loss。而验证集上的loss也有所减少，说明模型学到了数据的关键特征，并且泛化到了验证集上。这时模型所学习到的数据特征是有泛化能力的。
- 随着训练批次的上升，验证集的loss也开始上升。这说明模型正在学习训练集中不属于泛化特征的那些数据，也就是产生了过拟合现象。
- 总而言之，验证集上的loss呈先下降再上升的趋势，训练集上的loss则是一直下降。我们需要关注验证集上loss的最低点，这一点就是模型恰好没有过拟合训练集上数据的批次。我们应该在此停止训练，这样模型既学习到了泛化的数据特征，也没有对训练集上的数据进行过拟合。

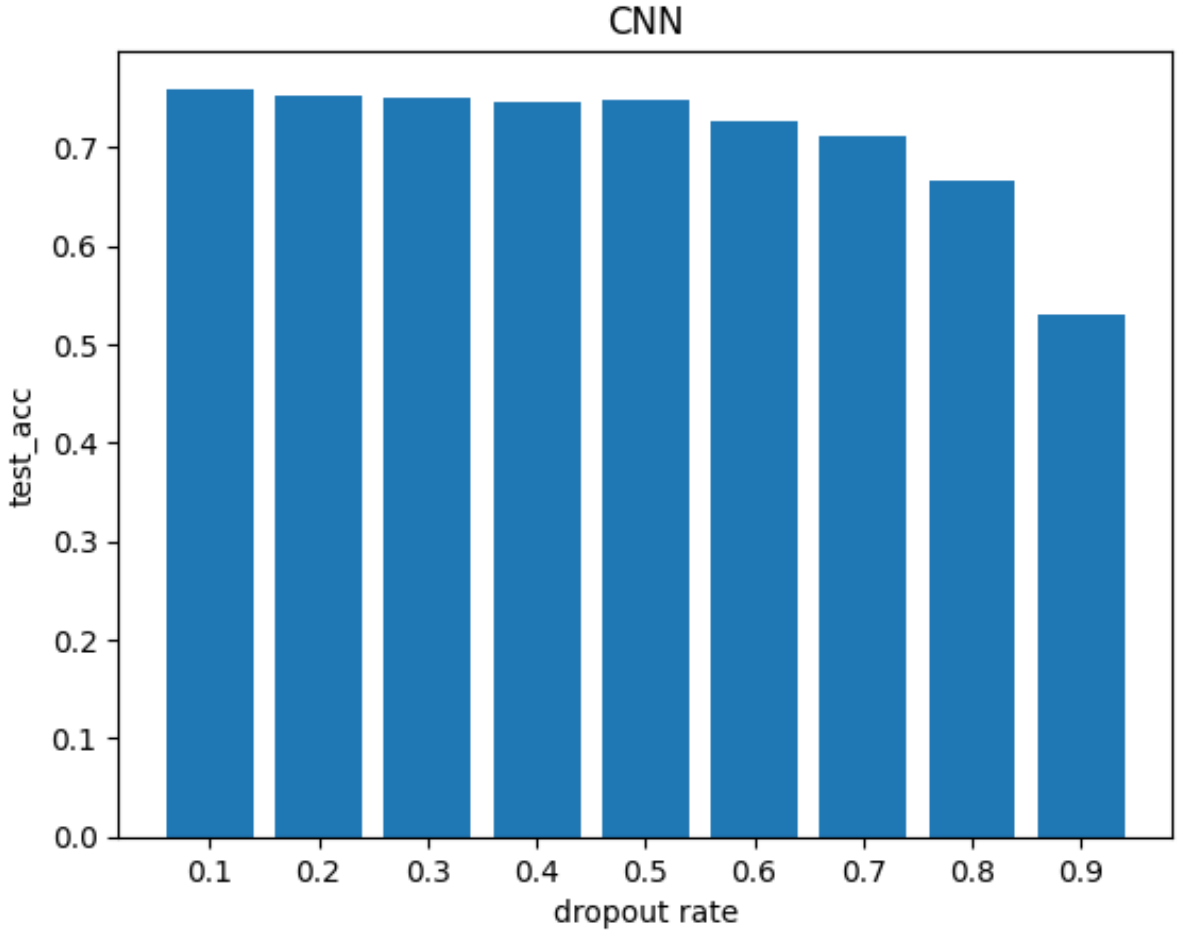
Final accuracy for testing

以下列出MLP和CNN不同dropout_rate在测试集上的表现结果：


MLP
test_loss
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
dropout rate


MLP
test_acc
0.5
0.4
0.3
0.2
0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
dropout rate


CNN
test_loss
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
dropout rate


CNN
test_acc
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
dropout rate

首先，可以明显看出CNN的效果远优于MLP。CNN在测试集上的准确率大致都在0.75左右，而MLP在测试集上的准确率只有0.55左右。这是模型本身带来的差异：对于图像这种数据结构而言，CNN比MLP有着更好的效果，因为CNN能够提取多通道的数据，以及图像局部的特征。

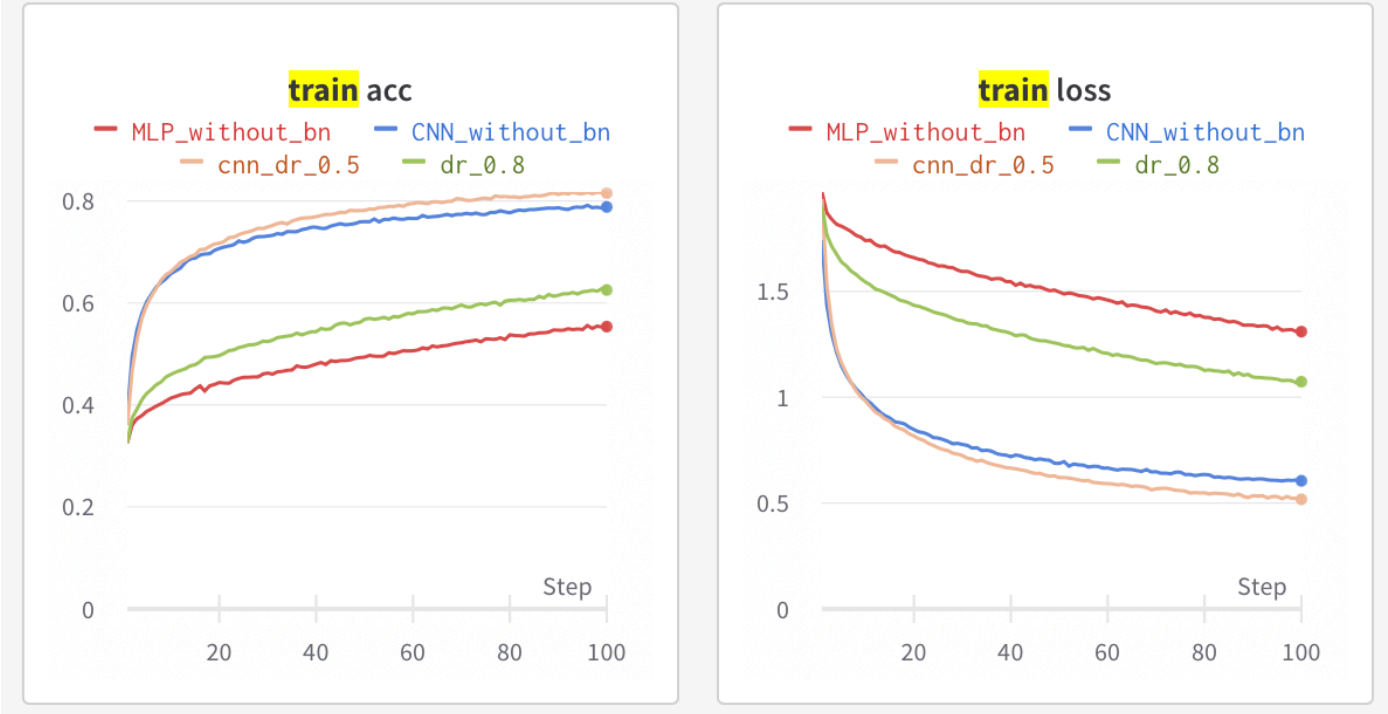
此外，在这个实验中，在验证集上表现最好的模型，在测试集上表现并非最好。例如MLP实验中，验证集上最好的模型是`dropout_rate=0.8`，而测试集上表现最好的模型却是`dropout_rate=0.7`。在CNN实验中，验证集上最好的模型是`dropout_rate=0.5`，而测试集上表现最好的模型却是`dropout_rate=0.1`。这可能是因为验证集不足够大的缘故。不过总体而言，在训练集上表现最好的模型在测试集上表现也是接近最好的。

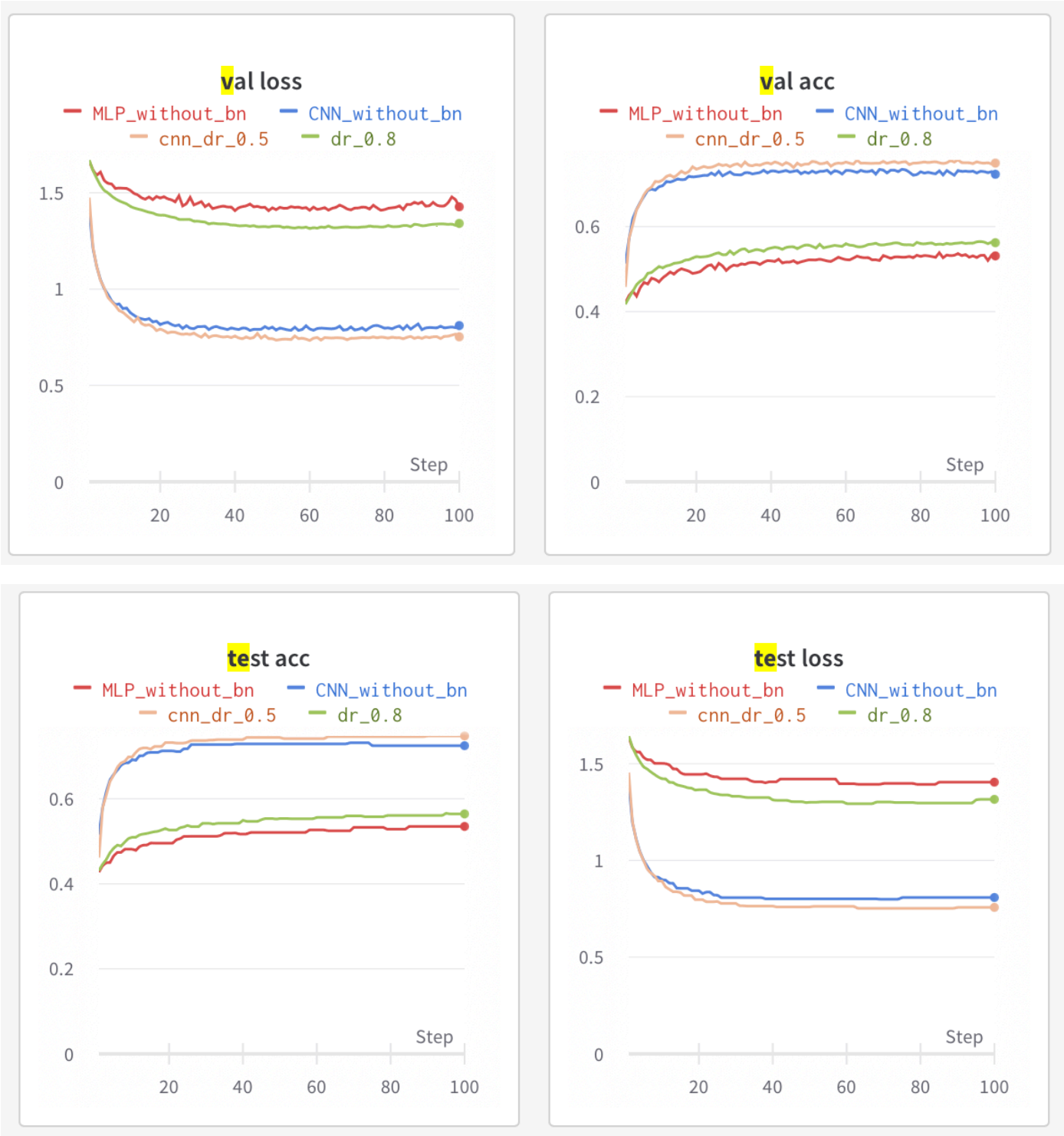
总之，在验证集上表现最好的两个模型的test loss和test acc为：

Model	Test loss	Test acc
MLP_dr=0.8	1.3156789135932923	0.5642999842762947
CNN_dr=0.5	0.7572290217876434	0.7477999848127365

Construct MLP and CNN without batch normalization, and discuss the effects of batch normalization

在`dropout_rate`为最佳的情况下，删去MLP和CNN的BN层，得到实验结果如下：

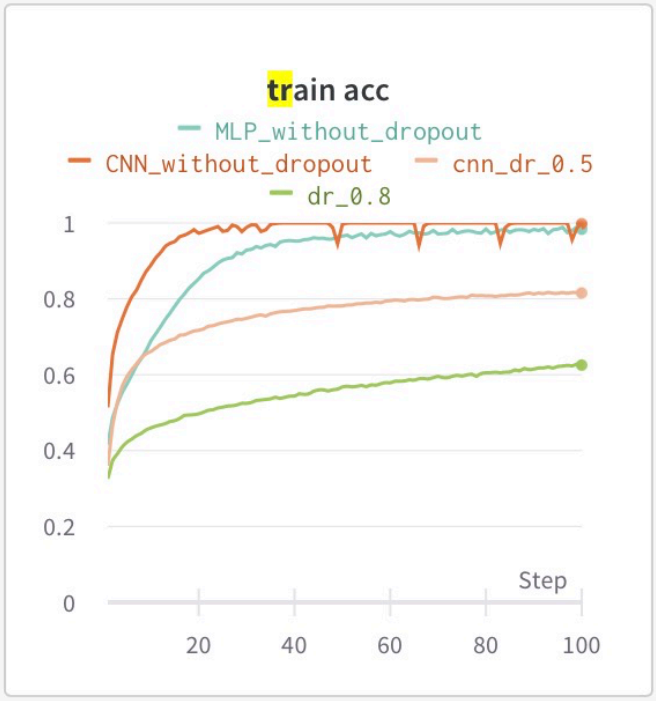


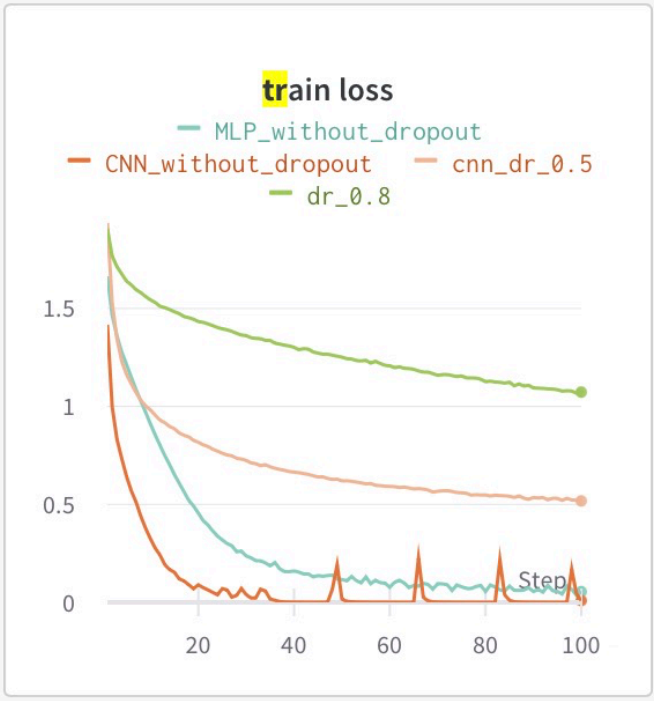


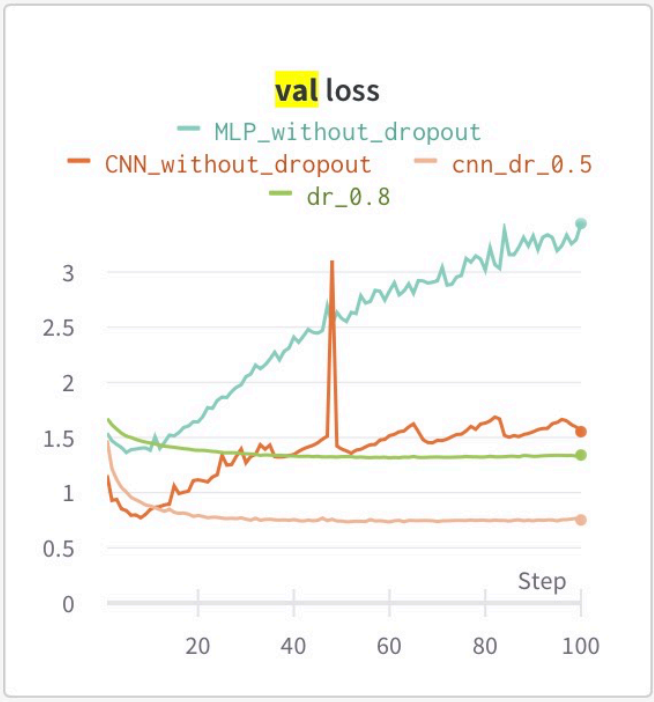
首先，在训练过程中，不论是MLP还是CNN，增加了BN层后模型的收敛速度，收敛效果都有所提升。这是因为BN可以解决**每个批次内部的数据偏移**：在训练过程中，如果各层分布存在差异，那么学习的难度将会大幅上升。增加了BN层以后，模型可以专注学习数据特征。

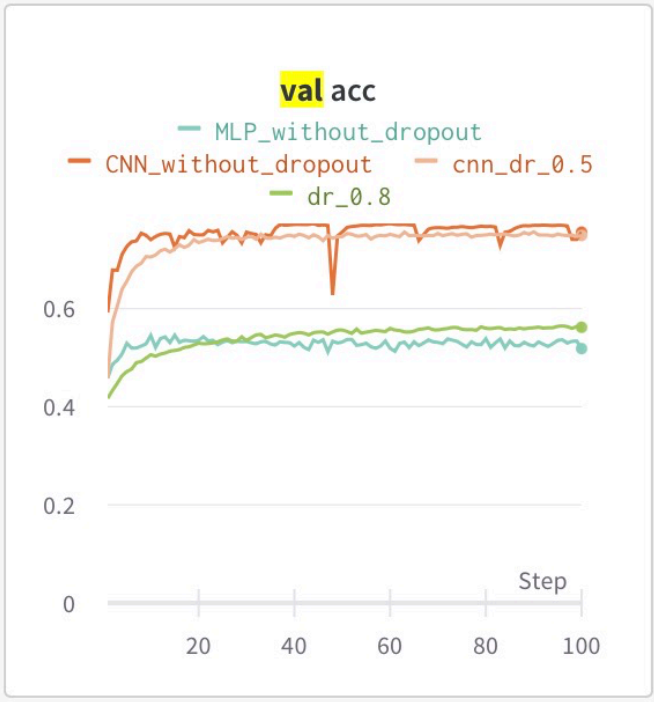
不过，在batch_size较小的情况下，BN的效果将有所下降；这时模型不能很好地估计输入样本的均值和方差。

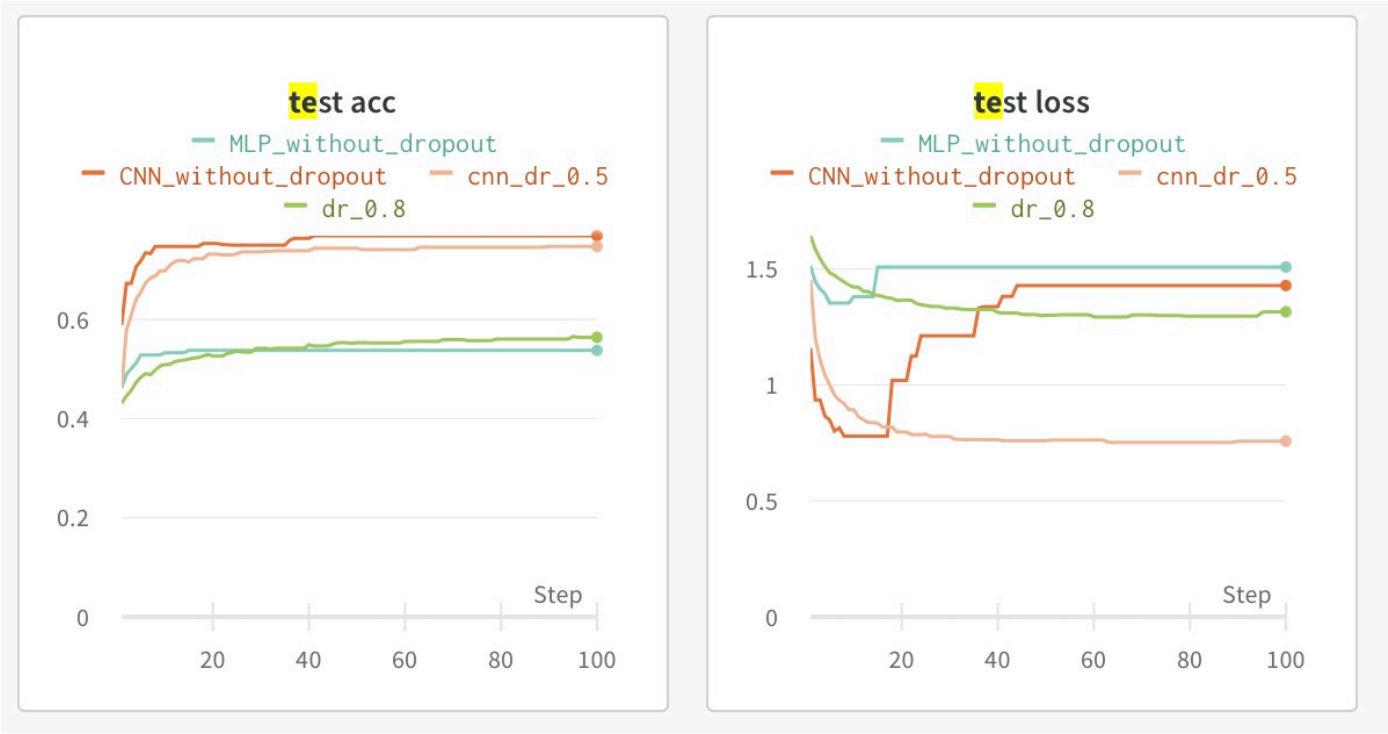
Construct MLP and CNN without dropout, and discuss the effects of dropout











可以看出，在**去掉了dropout以后**，模型在训练时的收敛速度和收敛效果得到了很大的提升。这说明模型拟合训练数据的能力得到了更大的提升。然而在验证集和测试集上的效果，MLP和CNN的表现有所不同：

- 对于MLP来说，增加dropout很明显改善了过拟合的问题。在验证集和测试集上，dropout都使得MLP有更好的泛化能力，取得了更高的准确率。
- 然而，对于CNN来说，增加dropout反而使模型在验证集和测试集上的表现有所下降。这可能是因为对于CNN模型而言，增加了dropout是对模型学习能力的劣势，使得数据中有待学习的特征并没有学习到。

总而言之，dropout的作用是：

- 在训练过程中，每次训练的网络结构实际上是各不相同的。类似于我们在训练很多个网络，但取的是这些网络的输出平均值，“不同结构的网络”输出的相互抵消可以有效解决过拟合问题。
- dropout过程可以减少某些神经元的联系关系，即不去学习类似于“一个特征生效，另一个特征才有效”的数据特征。即使数据中的某些特征丢失，模型仍然应该能给出正确的结果。这使得模型的泛化能力提高。

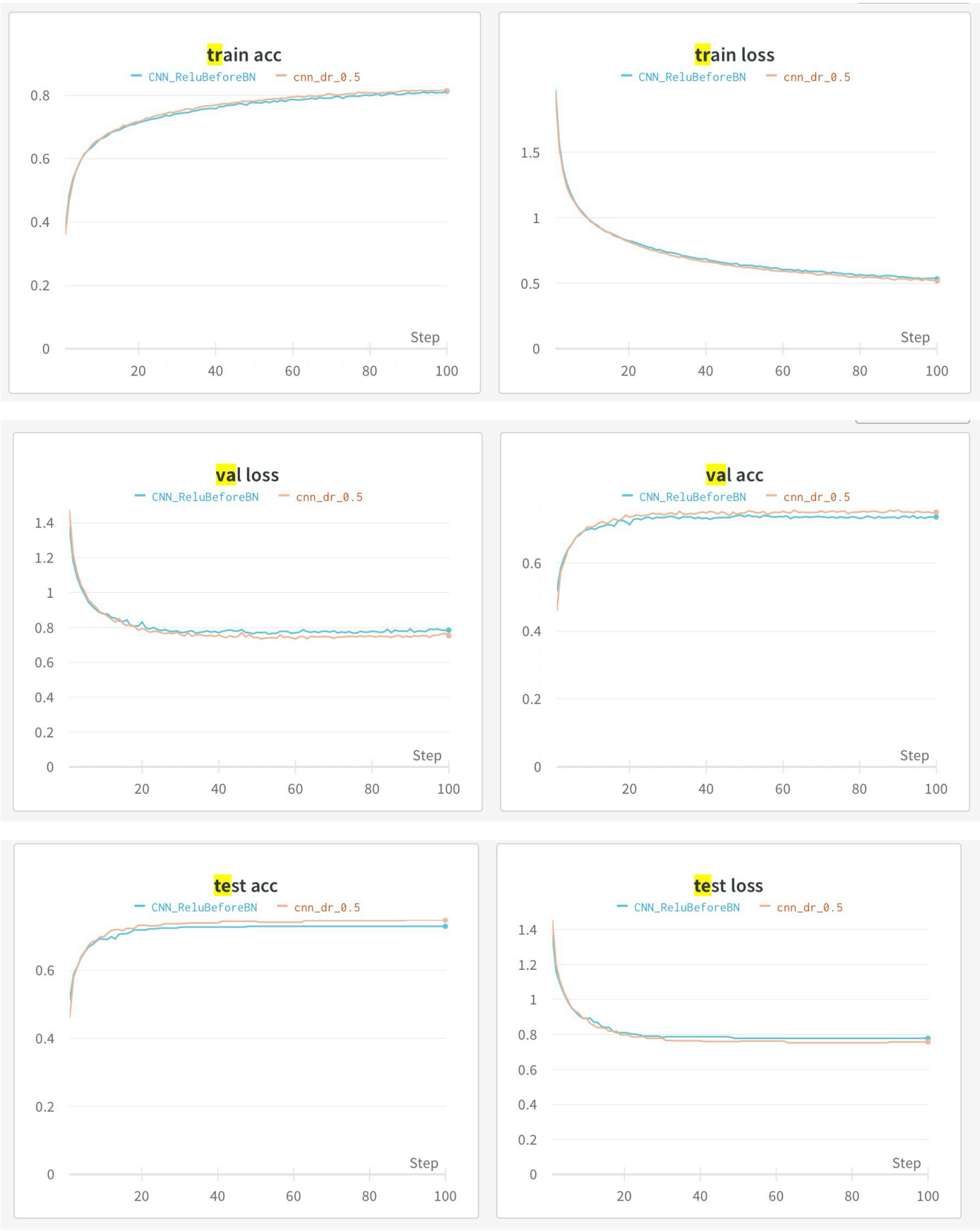
Consider changing the orders of different network blocks in CNN (e.g., Conv2d, BatchNorm2d, ReLU,Dropout2d and MaxPool2d). Show the results with different orders and explain why.

1. BN与Relu的位置关系

首先，我们考虑BN和Relu的位置关系。将BN和Relu的位置对调，将BN放到Relu以后，即模型的顺序为：

```
Model(
  (logits): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d()
    (3): Dropout()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=1, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (6): ReLU()
    (7): BatchNorm2d()
    (8): Dropout()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=1, dilation=1, ceil_mode=False)
    (10): Flatten(start_dim=1, end_dim=-1)
    (11): Linear(in_features=10368, out_features=10, bias=True)
  )
  (loss): CrossEntropyLoss()
)
```

得到的实验结果和对比如下：

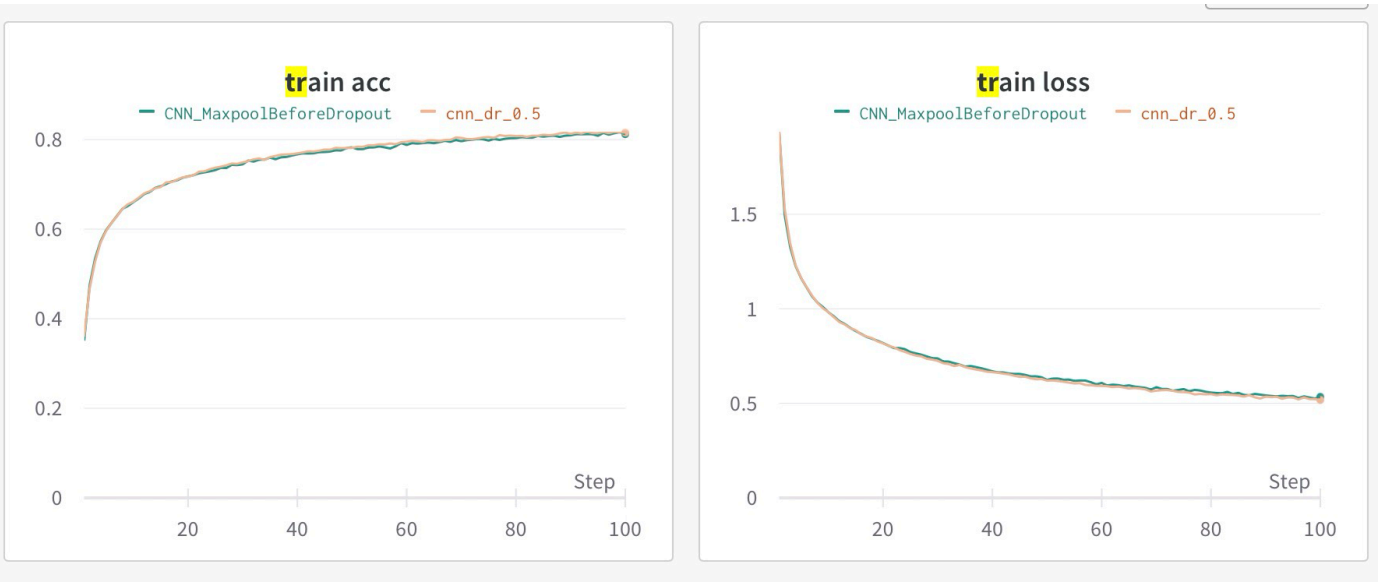


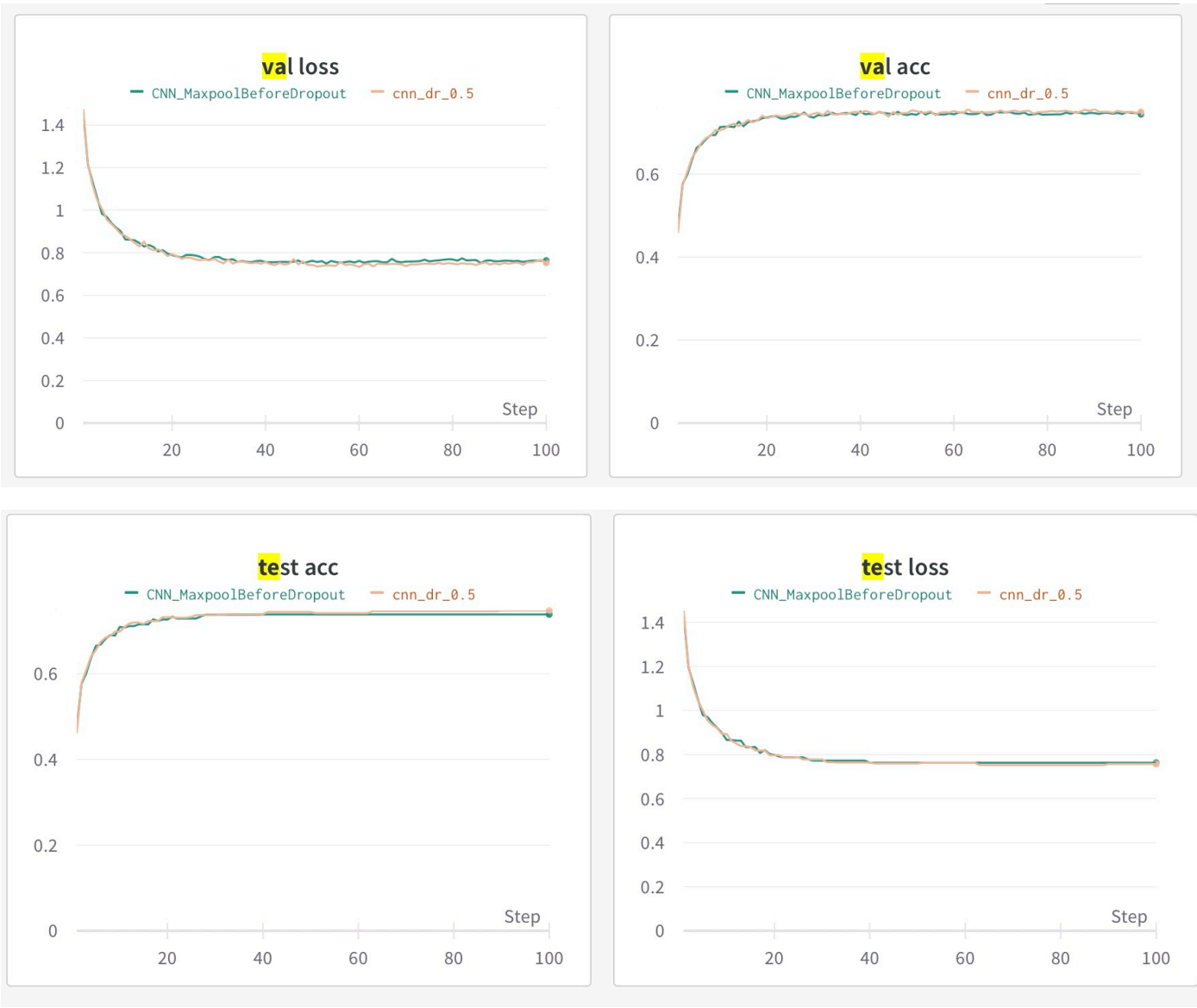
在最后的测试集上，最先的模型准确率为0.7478，而将ReLU提前的准确率为0.7293，可以看出将ReLU放在BN之后的效果略为更好。这可能是因为如果将BN放在ReLU以后，因为**ReLU的输出非负，不能在这之后再将数据变换为均值为0的高斯分布**。不过总体而言二者的效果相差无几，不同的数据可能会导致不同的结果。

2. maxpool与dropout的位置关系

然后，我们尝试将maxpool与dropout的位置对调。即：

```
Model(
  (logits): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d()
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=1, dilation=1, ceil_mode=False)
    (4): Dropout()
    (5): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (6): BatchNorm2d()
    (7): ReLU()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=1, dilation=1, ceil_mode=False)
    (9): Dropout()
    (10): Flatten(start_dim=1, end_dim=-1)
    (11): Linear(in_features=10368, out_features=10, bias=True)
  )
  (loss): CrossEntropyLoss()
)
```



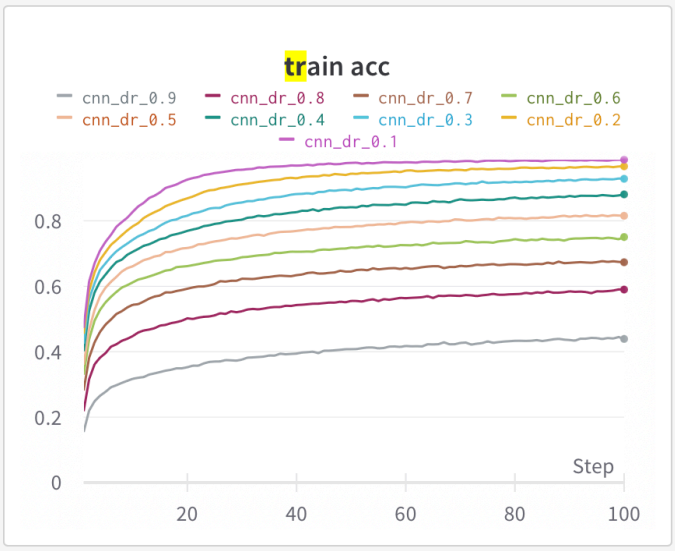


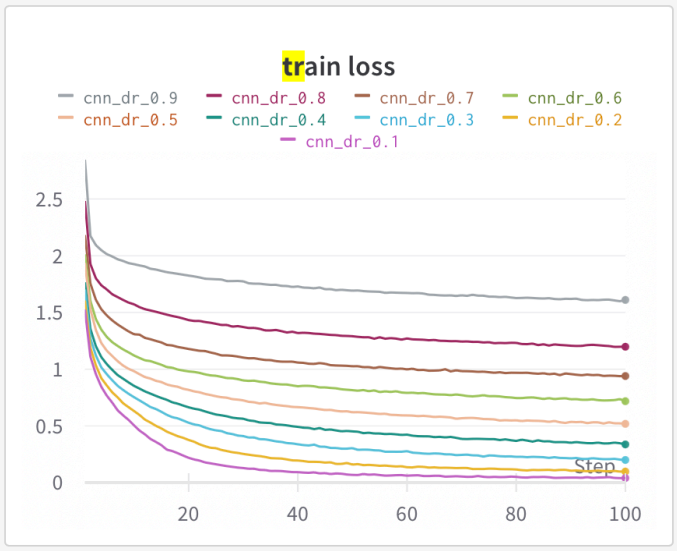
也即是观察**先dropout，后取最大**和**先取最大，后dropout**的效果比较。不过结果仍然没什么差异，可能的原因是同一通道之间相邻数据大小较为接近，故dropout的先后顺序对maxpool的影响不大。

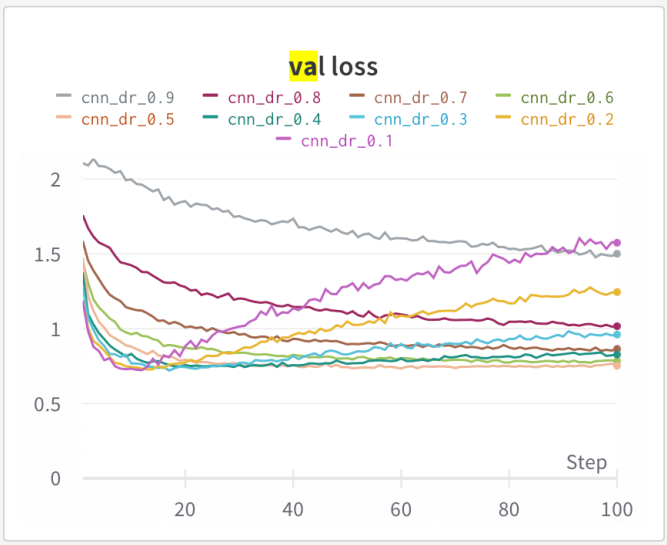
Tune the hyper-parameters dropout rate. Analyze how the hyper-parameters influencethe final performances.

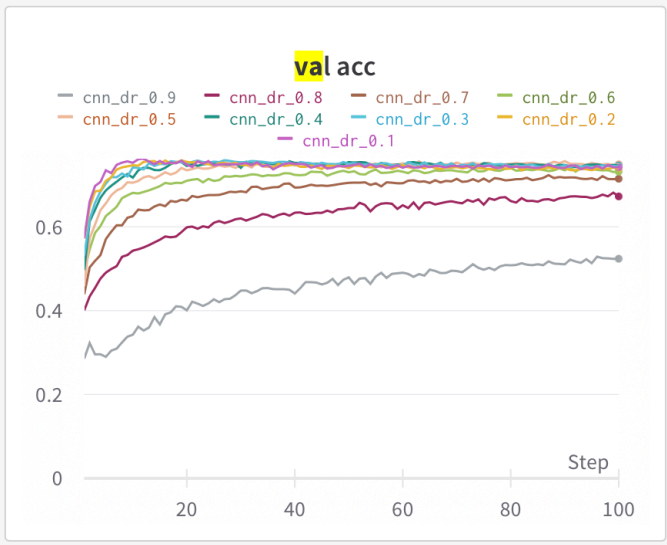
不同dropout_rate对模型的影响图表如下：

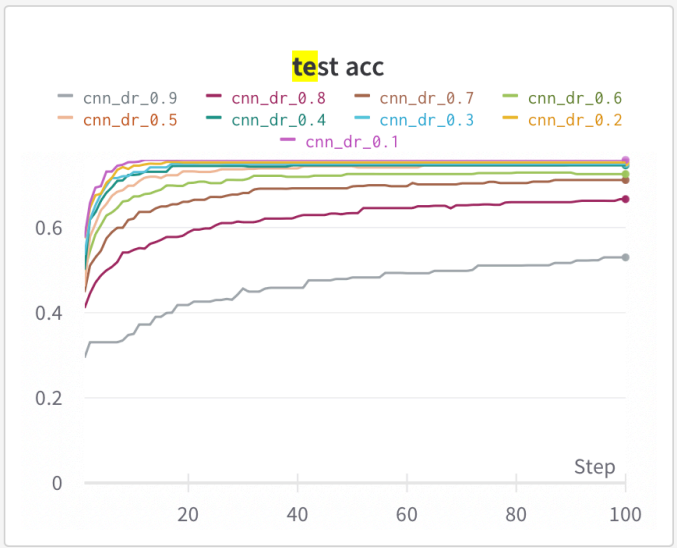
- CNN

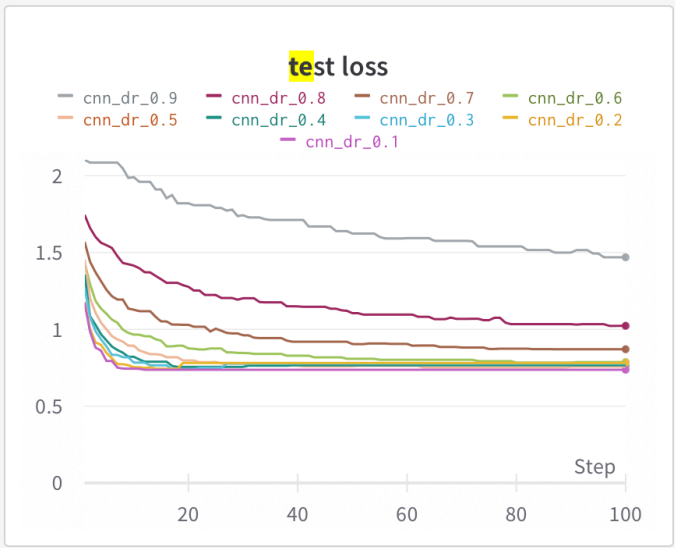

train acc
cnn_dr_0.9
cnn_dr_0.8
cnn_dr_0.7
cnn_dr_0.6
cnn_dr_0.5
cnn_dr_0.4
cnn_dr_0.3
cnn_dr_0.2
cnn_dr_0.1
0.8
0.6
0.4
0.2
0
Step
20
40
60
80
100


train loss
cnn_dr_0.9
cnn_dr_0.8
cnn_dr_0.7
cnn_dr_0.6
cnn_dr_0.5
cnn_dr_0.4
cnn_dr_0.3
cnn_dr_0.2
cnn_dr_0.1
2.5
2
1.5
1
0.5
0
Step
20
40
60
80
100

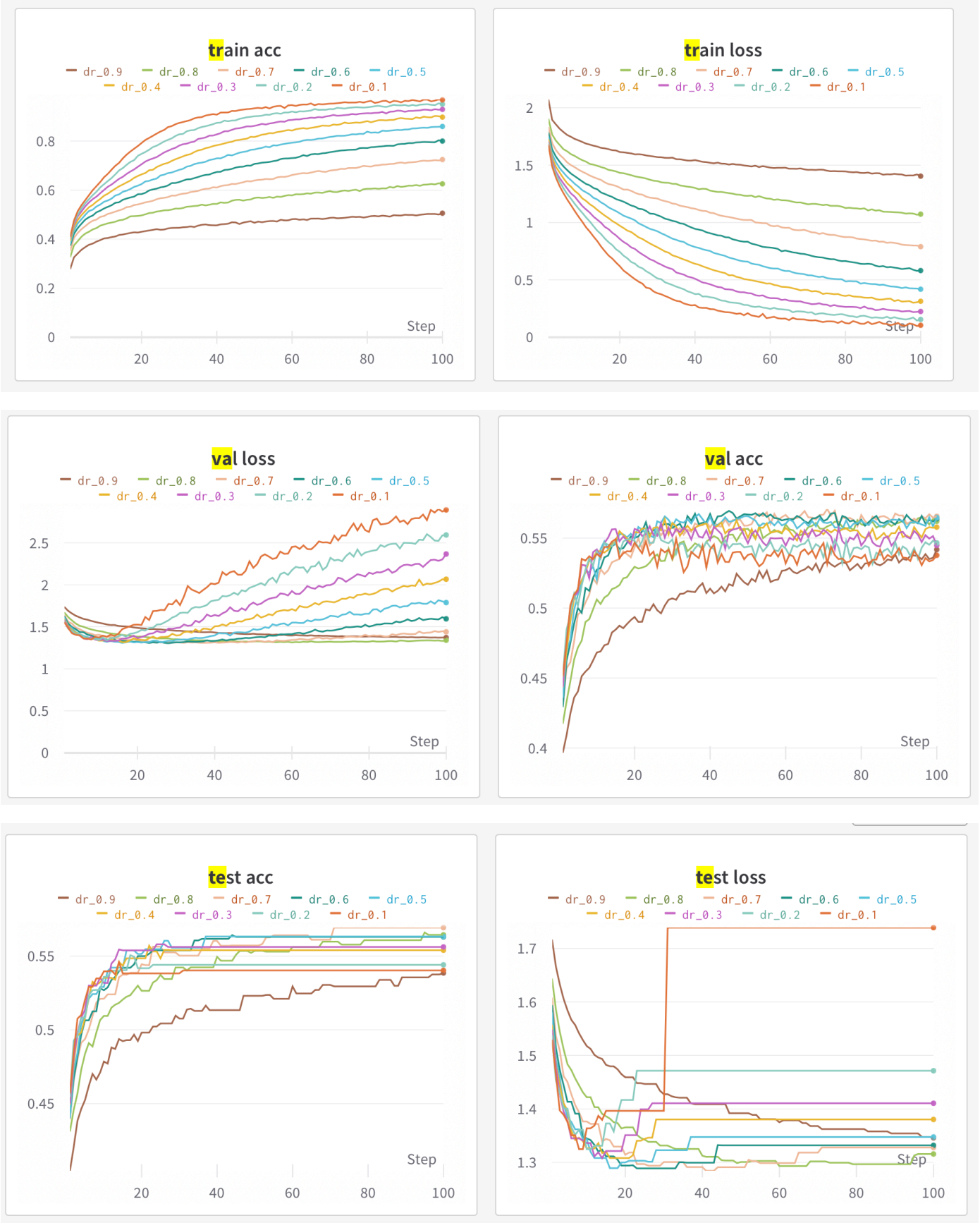

val loss
cnn_dr_0.9
cnn_dr_0.8
cnn_dr_0.7
cnn_dr_0.6
cnn_dr_0.5
cnn_dr_0.4
cnn_dr_0.3
cnn_dr_0.2
cnn_dr_0.1
2
1.5
1
0.5
0
Step
20
40
60
80
100


val acc
cnn_dr_0.9
cnn_dr_0.8
cnn_dr_0.7
cnn_dr_0.6
cnn_dr_0.5
cnn_dr_0.4
cnn_dr_0.3
cnn_dr_0.2
cnn_dr_0.1
0.6
0.4
0.2
0
Step
20
40
60
80
100


test acc
cnn_dr_0.9
cnn_dr_0.8
cnn_dr_0.7
cnn_dr_0.6
cnn_dr_0.5
cnn_dr_0.4
cnn_dr_0.3
cnn_dr_0.2
cnn_dr_0.1
0.6
0.4
0.2
0
Step
20
40
60
80
100


test loss
cnn_dr_0.9
cnn_dr_0.8
cnn_dr_0.7
cnn_dr_0.6
cnn_dr_0.5
cnn_dr_0.4
cnn_dr_0.3
cnn_dr_0.2
cnn_dr_0.1
2
1.5
1
0.5
0
Step
20
40
60
80
100

- MLP



- 在训练集上，dropout_rate基本与收敛速度和收敛效果成反比。如前所述，dropout的存在有效降低了模型对训练集的拟合能力。过小的dropout_rate可能会导致模型过拟合，而过大的dropout_rate可能会导致模型学习能力不足，最终欠拟合。在这次实验中，我们总是选择在验证集上表现最好的dropout_rate作为最终的参数选择。

- 除此以外，MLP和CNN拥有不同的最佳dropout_rate。这可能是因为不同模型本身的学习能力就不同，dropout层对其影响的程度也有所不同。对于CNN来说，较低的dropout_rate有优势（甚至没有dropout层可能会更好），而MLP则是更favour较高的dropout_rate。
- 另一个现象是：越高的dropout_rate往往导致更大的best_epoch（即val_loss的极值点）。这也和dropout影响学习能力相吻合，更大的dropout_rate会导致对数据特征的提取能力有所下降，故可能需要更多的epoches来训练最佳的模型。