ANN实验报告2

张天祺 2021010719 计12

`self.training`如何工作，为什么训练和测试不一样

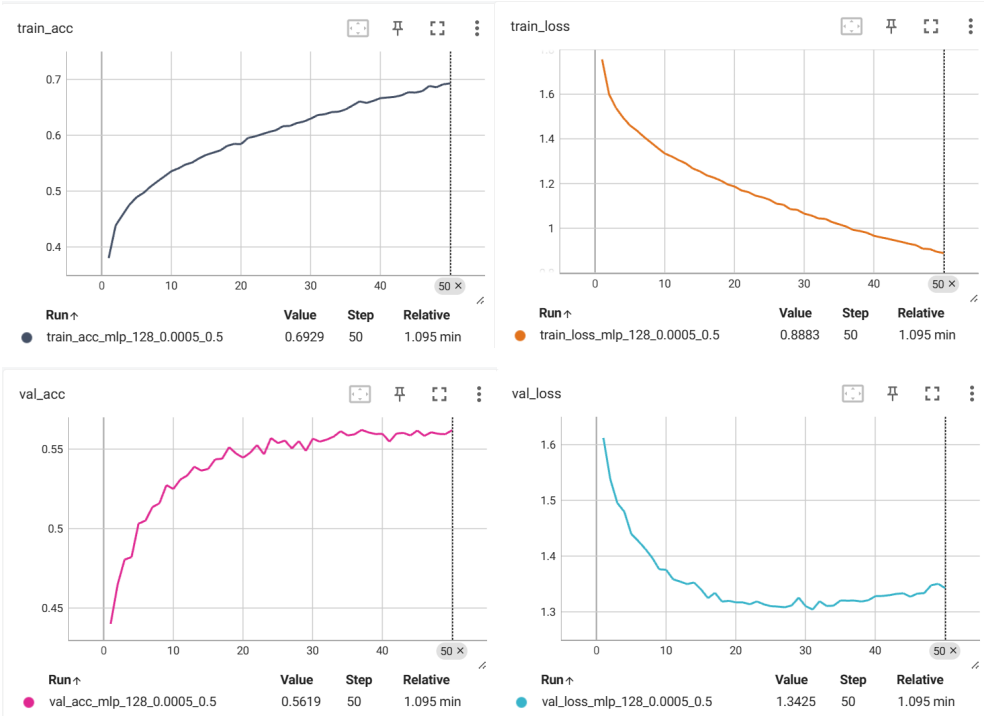
在`torch.nn.Module`中存在成员变量`training`，这个变量在成员函数`train()`中被设置为`True`，而在成员函数`eval()`中被设置成`False`。因此在对`Module`进行继承的类都拥有这个成员变量和成员函数，可以用于区分在训练过程还是验证过程。

在训练时，往往需要计算loss、梯度然后进行回传更新参数，而在测试的过程中则指执行计算loss的过程，并不会回传梯度更新模型参数。因此，这也是训练和测试不同的原因。

MLP和CNN模型的实验效果

MLP模型

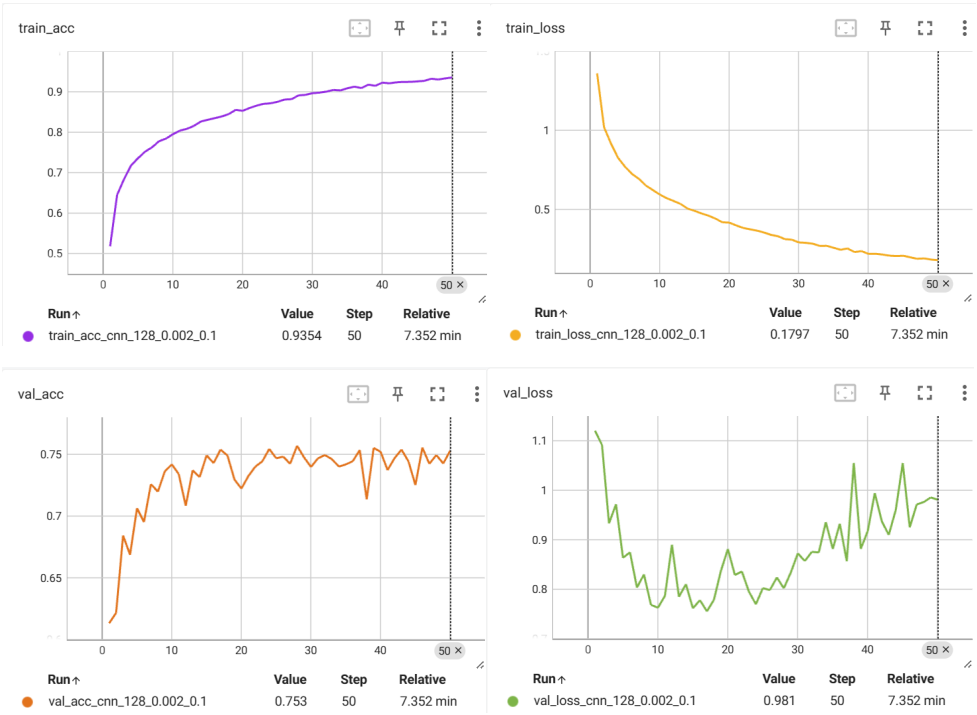
对于`MLP`模型，在`BatchNorm1d`的建构过程中，我选择了`exponential moving average`，`bias`选择了1e-5。实验得出的最好的超参数设置为：隐藏层神经元数量512，`batch_size`为128，`learning_rate`为5e-3，，`drop_rate`为0.5，具体实验表现如下：



CNN模型

对于`CNN`，实验得出的最好的超参数设置为：`batch_size`为128，`learning_rate`为2e-3，`drop_rate`为0.1

在构建`CNN`模型时，我设置了两个卷积层的输出维数分别是128和64，`BatchNorm2d`的具体实现与`BatchNorm1d`一致，最大池化选择核为3*3，步长为2，最后通过`1600*10`的全连接层得到`logits`。具体实验表现如下：



解释`training loss`和`validation loss`为什么不同，对调节超参数有什么帮助

通过`mlp`和`cnn`的实验可以看出来，`training loss`随着训练epoch的增加基本会保持持续下降的趋势，表示模型对于训练集的拟合程度越来越好，同时也可以看到训练的准确率也是在逐步提升。而`validation loss`则并不是这样，随着训练量的提升，`validation loss`先下降，然后进入平台期，此时继续训练，`validation loss`不降反升，同时验证集准确率也会一直在一个值附近波动。

对于这种情况，可能的原因是：

- 训练集和测试集上数据的分布并不一致，模型在训练集上表现较好可能验证集上效果不佳。
- 训练的时间过长后模型对于训练集的拟合效果较好，但也有可能陷入过拟合中，模型的泛化能力下降。

这对我们调节超参数的帮助是：通过比较`training loss`和`validation loss`各自的变化情况，我们可以判断模型的拟合情况，如果有些过拟合，我们就可以适当降低训练的`epoch`数量或者适当调低学习率等，同时，也可以增加`weight_decay`这种惩罚过拟合的超参数。

测试集的最终准确率

`mlp`模型在上文选择的超参数下，测试集上的最好准确率为0.545，最低loss为1.43

`cnn`模型在上文选择的超参数下，测试集上的最好准确率为0.748，最低loss为0.86

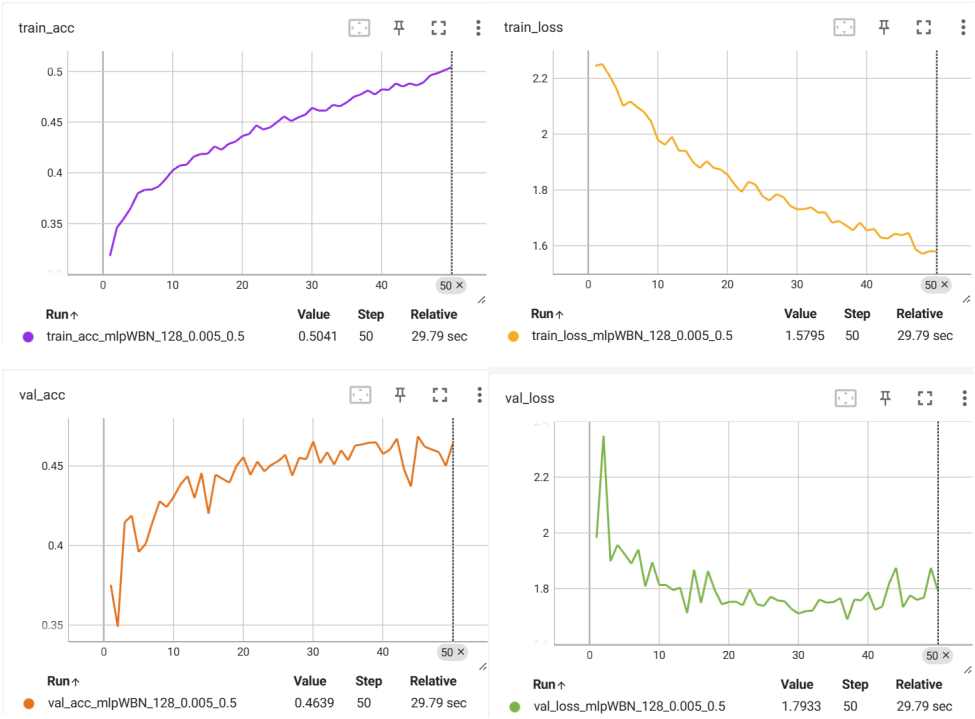
可以看出`cnn`模型显著优于`mlp`模型，我认为原因可能有以下几点：

- `cnn`的模型计算复杂度更高，而卷积核这一设置又可以在一定程度上降低参数量，也就使`cnn`使用更少的参数量也可以达到更好的模型泛化效果。同时从模型内部看，`cnn`有着局部连接性，可以更好提供图片的局部特征。
- `mlp`的参数虽然相对更多，但是全连接性使得它更容易陷入过拟合的问题，也容易陷入到局部最优中。

无`batch normalization`的模型效果

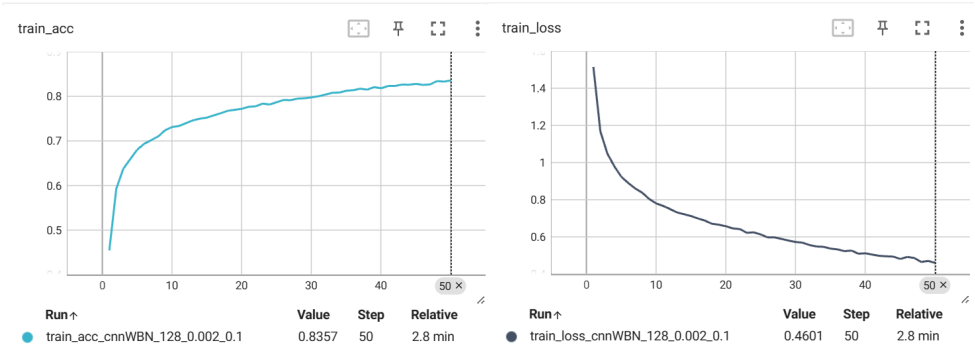
`mlp`模型

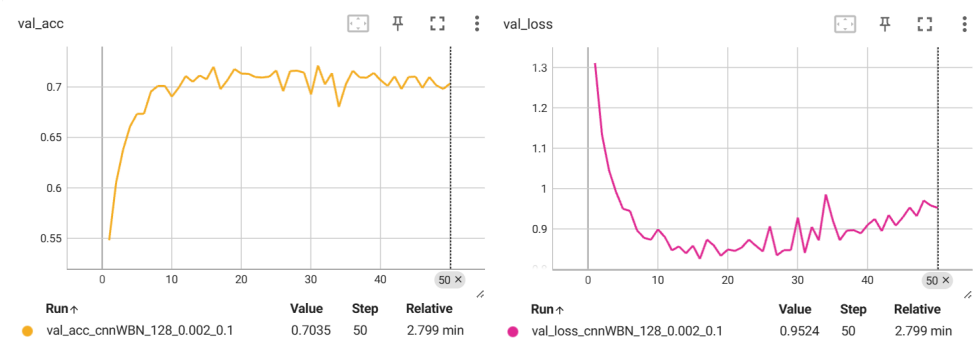
选择上文中的超参数，去除`batch normalization`之后，训练的效果如下：



`cnn`模型

选择上文中的超参数，去除`batch normalization`之后，训练的效果如下：





结果分析

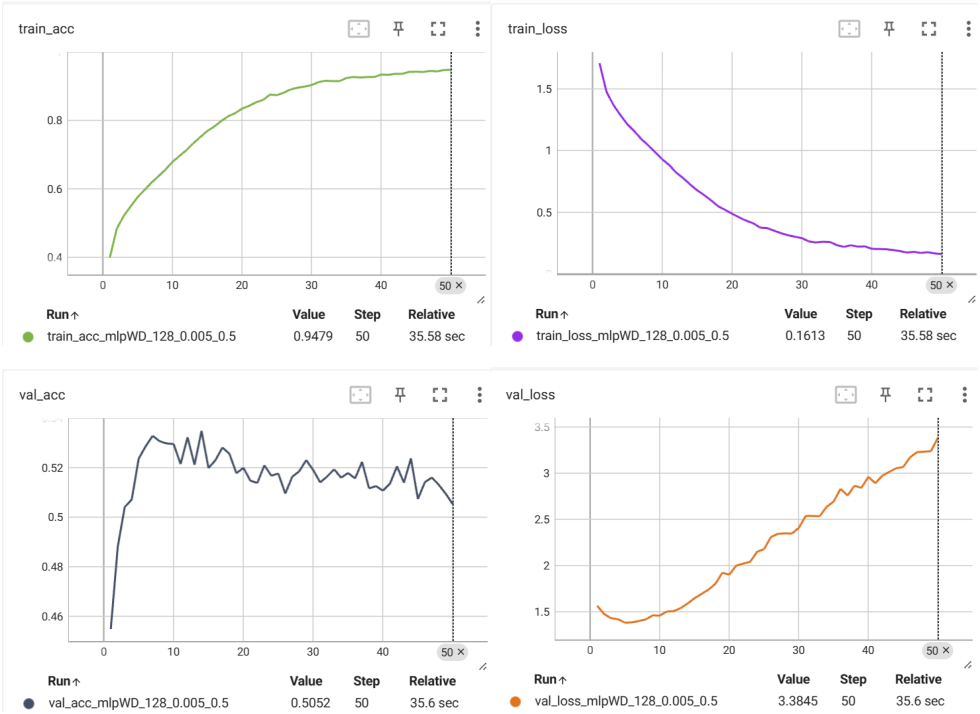
可以看出，去除掉batch normalization之后，两种模型的效果都有一定的下降，我认为可能的原因如下：

- batch normalization通过对小批量的数据归一化处理，可以减少每一层输入的分布差异，从而加快收敛的过程。
- batch normalization通过归一化输入可以稳定梯度的传播，降低梯度消失和梯度爆炸发生的概率。

无`Dropout`的模型效果

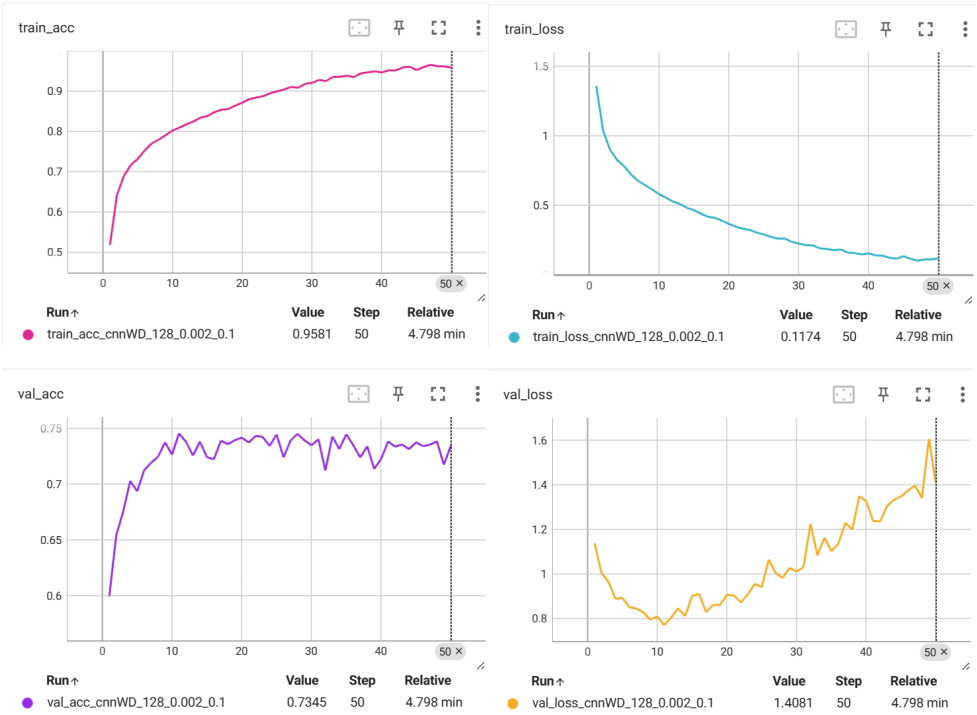
`mlp`模型

选择上文中的超参数，去除`Dropout`之后，训练的效果如下：



可以看出结果有一定的下降。

cnn模型



总体影响不大，结果有一点点下降

结果分析

对于以上实验现象，我认为有如下原因：

- `Dropout`通过随机丢弃神经元可以实现对于过拟合现象的有效缓解，而`mlp`模型因为神经元之间是全连接的，更容易出现过拟合现象，因此去除`Dropout`对于`mlp`模型的影响相对较大。
- 对于`cnn`而言，由于`cnn`的局部连接性导致其相对全连接的`mlp`而言本身就更不容易出现过拟合的现象，而且`Dropout`虽然能在原理上环节过拟合的现象，但是随机丢弃神经元也会导致`cnn`图像数据的不完整性，这也是去除掉`Dropout`后模型在验证集上稍有下降但是在训练集上准确率反而有所提升的原因。

调整CNN结构的顺序

控制实验的超参数与上文中最好的超参数一致，调整`cnn`模型的层的顺序，由于模型层之间组合的种类过多，因此简要抽出几种组合方法进行实验，得到如下结果：

> 部分实验修改了线性层的输入维度以与其所连接的层的维度匹配

	train loss	train acc	validation loss	validation acc	test loss	test acc
交换`BatchNorm`和`Relu`	0.1766	0.9361	1.2656	0.7228	1.0602	0.728
将`MaxPool`放在`Conv`前	0.7826	0.8241	1.2519	0.6258	1.0482	0.642
将`Dropout`放在`BatchNorm`前	0.1920	0.9313	0.9881	0.7300	0.8510	0.746

对于这三种调整结构的尝试，可以看出模型的能力都有着不同程度的下降，原因分别如下：

- `Relu`层的作用是在神经网络中引入非线性性质，将负数输入变为零，而保持正数输入不变。这种非线性性使神经网络能够学习非线性关系，从而更好地拟合复杂的数据分布和模式。但是如果将`Relu`层放在`BatchNorm`层之前，由于负数的输入全部变成了0，所以会导致模型学习到的特征存在部分偏差，同时也有可能导致梯度消失的问题。
- `MaxPool`层的作用是最大池化，对输入特征图进行下采样，以减小特征图的尺寸，降低计算复杂性，并减少过拟合的风险。但是如果将`MaxPool`层放在`Conv`层之前，可能会导致模型丧失空间信息太快，这也显著影响了模型的性能。
- `Dropout` 层通常用于模型的正则化，以减少过拟合。但是如果将 `Dropout` 层放置在 `Conv` 和 `BatchNorm2d` 之间，可能会导致模型在正则化方面的效果下降，但这对于本次实验任务或许并不明显，这也是这一顺序的调整对于模型的最终表现影响不大的原因。

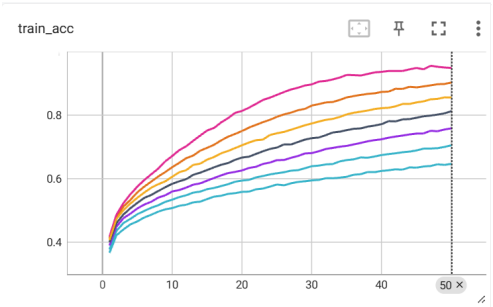
调整超参数实验效果

> 对于下面所有实验结果图像的命名，均为
> `{metric}_{model_name}_{batchsize}_{learningrate}_{Dropout}`

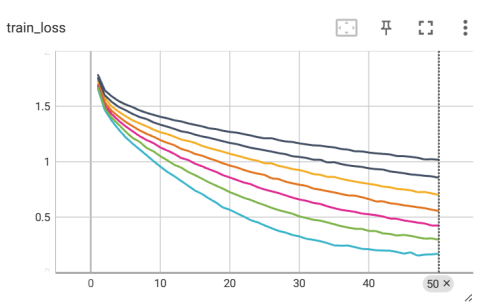
调整`Dropout`超参数对于实验的影响

`mlp`模型

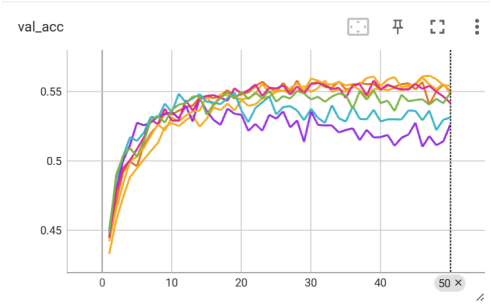
控制模型结构和其他超参数与上文实验设置相同的条件下，调节`Dropout`超参数，分别使`Dropout`数值为0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6，具体得到实验结果如下：



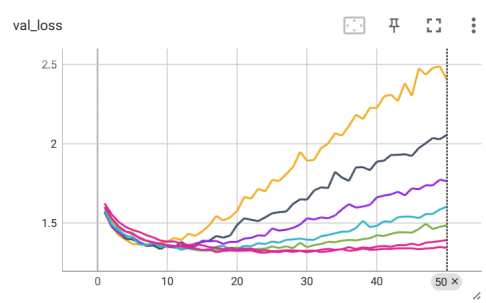
Run	Value	Step	Relative
train_acc_mlp_128_0.001_0.0	0.9485	50	56.71 sec
train_acc_mlp_128_0.001_0.1	0.9026	50	57.52 sec
train_acc_mlp_128_0.001_0.2	0.8555	50	50.95 sec
train_acc_mlp_128_0.001_0.3	0.8126	50	52.37 sec
train_acc_mlp_128_0.001_0.4	0.7584	50	57.27 sec
train_acc_mlp_128_0.001_0.5	0.7052	50	52.5 sec
train_acc_mlp_128_0.001_0.6	0.6467	50	1.113 min



Run	Value	Step	Relative
train_loss_mlp_128_0.001_0.0	0.1698	50	56.71 sec
train_loss_mlp_128_0.001_0.1	0.2965	50	57.52 sec
train_loss_mlp_128_0.001_0.2	0.4244	50	50.95 sec
train_loss_mlp_128_0.001_0.3	0.556	50	52.37 sec
train_loss_mlp_128_0.001_0.4	0.7016	50	57.27 sec
train_loss_mlp_128_0.001_0.5	0.856	50	52.5 sec
train_loss_mlp_128_0.001_0.6	1.0181	50	1.113 min



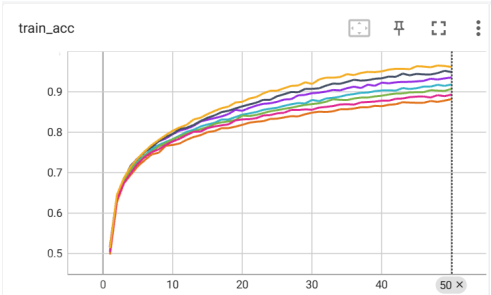
Run	Value	Step	Relative
val_acc_mlp_128_0.001_0.0	0.526	50	56.7 sec
val_acc_mlp_128_0.001_0.1	0.5313	50	57.49 sec
val_acc_mlp_128_0.001_0.2	0.5489	50	50.95 sec
val_acc_mlp_128_0.001_0.3	0.5419	50	52.38 sec
val_acc_mlp_128_0.001_0.4	0.5501	50	57.26 sec
val_acc_mlp_128_0.001_0.5	0.5487	50	52.51 sec
val_acc_mlp_128_0.001_0.6	0.5543	50	1.112 min



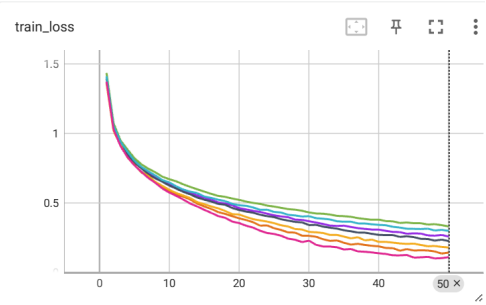
Run	Value	Step	Relative
val_loss_mlp_128_0.001_0.0	2.4054	50	56.7 sec
val_loss_mlp_128_0.001_0.1	2.0556	50	57.49 sec
val_loss_mlp_128_0.001_0.2	1.7641	50	50.95 sec
val_loss_mlp_128_0.001_0.3	1.6036	50	52.38 sec
val_loss_mlp_128_0.001_0.4	1.4856	50	57.26 sec
val_loss_mlp_128_0.001_0.5	1.3925	50	52.51 sec
val_loss_mlp_128_0.001_0.6	1.3439	50	1.112 min

`cnn`模型

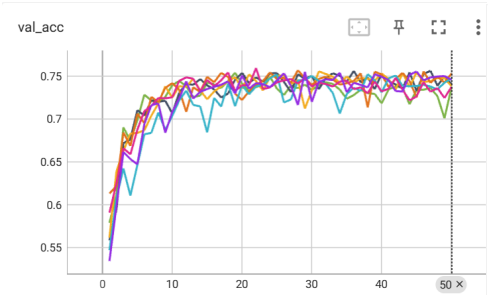
控制模型结构和其他超参数与上文实验设置相同的条件下，调节`Dropout`超参数，分别使`Dropout`数值为0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3，具体得到实验结果如下：



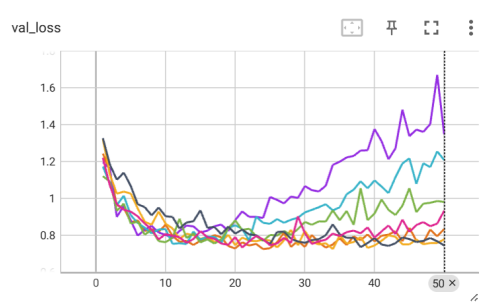
Run	Value	Step	Relative
train_acc_cnn_128_0.002_0.0	0.9613	50	7.368 min
train_acc_cnn_128_0.002_0.05	0.9487	50	7.364 min
train_acc_cnn_128_0.002_0.1	0.9354	50	7.352 min
train_acc_cnn_128_0.002_0.15	0.918	50	5.502 min
train_acc_cnn_128_0.002_0.2	0.9072	50	5.461 min
train_acc_cnn_128_0.002_0.25	0.893	50	5.46 min
train_acc_cnn_128_0.002_0.3	0.882	50	5.456 min



Run	Value	Step	Relative
train_loss_cnn_128_0.002_0.0	0.1103	50	7.368 min
train_loss_cnn_128_0.002_0.05	0.1434	50	7.364 min
train_loss_cnn_128_0.002_0.1	0.1797	50	7.352 min
train_loss_cnn_128_0.002_0.15	0.2262	50	5.502 min
train_loss_cnn_128_0.002_0.2	0.2581	50	5.461 min
train_loss_cnn_128_0.002_0.25	0.299	50	5.46 min
train_loss_cnn_128_0.002_0.3	0.331	50	5.456 min



Run	Value	Step	Relative
val_acc_cnn_128_0.002_0.0	0.7366	50	7.368 min
val_acc_cnn_128_0.002_0.05	0.7374	50	7.364 min
val_acc_cnn_128_0.002_0.1	0.753	50	7.352 min
val_acc_cnn_128_0.002_0.15	0.7383	50	5.501 min
val_acc_cnn_128_0.002_0.2	0.7406	50	5.461 min
val_acc_cnn_128_0.002_0.25	0.7458	50	5.46 min
val_acc_cnn_128_0.002_0.3	0.7473	50	5.456 min



Run	Value	Step	Relative
val_loss_cnn_128_0.002_0.0	1.3471	50	7.368 min
val_loss_cnn_128_0.002_0.05	1.2077	50	7.364 min
val_loss_cnn_128_0.002_0.1	0.981	50	7.352 min
val_loss_cnn_128_0.002_0.15	0.9337	50	5.502 min
val_loss_cnn_128_0.002_0.2	0.8346	50	5.461 min
val_loss_cnn_128_0.002_0.25	0.7788	50	5.46 min
val_loss_cnn_128_0.002_0.3	0.7449	50	5.456 min

结果分析

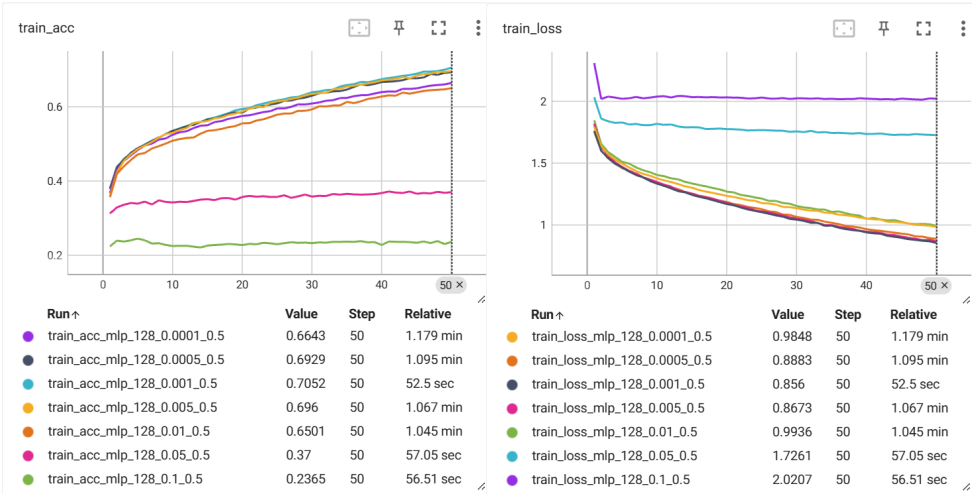
- 可以从两种模型中看出，随着`Dropout`数值的增加，模型的训练集准确率逐渐下降，这与上文的分析也保持一致，而验证集的准确率`mlp`上升而`cnn`出现波动。
- 也正如上文分析所言，`mlp`模型对于`Dropout`的依赖程度要高于`cnn`，敏感度也会更高，因此随着`Dropout`数值的增加`mlp`在验证集上的准确率逐渐上升，当然这个上升肯定不是无限度的上升，因为`Dropout`掉过多的神经元必然会导致模型无法正常学习到特征。而对于`cnn`而言，由于

模型自身对于过拟合的抵抗程度就较好，所以对于`Dropout`的依赖就不高，它所提供的更多的是一种随机性。

调整`learningrate`超参数对于实验的影响

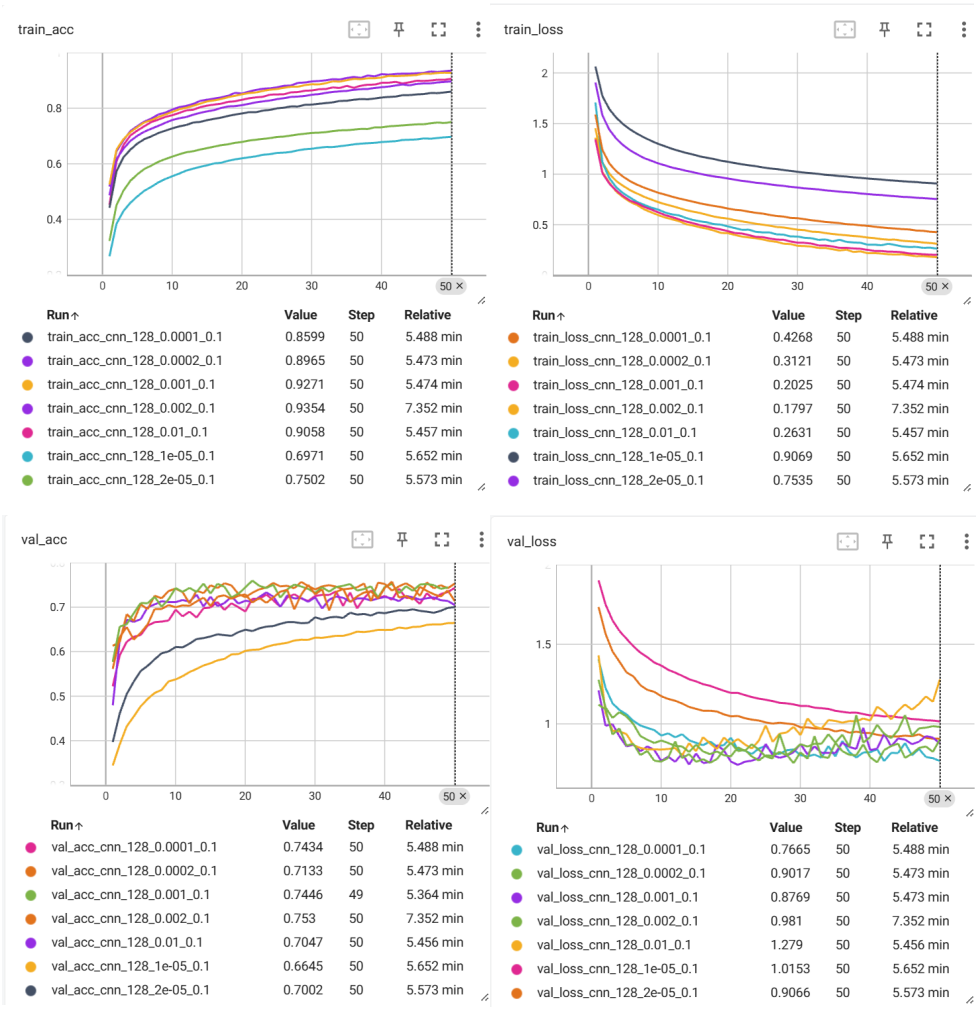
`mlp`模型

控制模型结构和其他超参数与上文实验设置相同的条件下，调节`learningrate`超参数，分别使`learningrate`数值为1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4，具体得到实验结果如下：



`cnn`模型

控制模型结构与上文实验设置相同的条件下，控制超参数`batch_size`为128，`Dropout rate`为0.1，调节`learningrate`超参数，分别使`learningrate`数值为1e-2, 2e-3, 1e-3, 2e-4, 1e-4, 2e-5, 1e-5，具体得到实验结果如下：



Run	Value	Step	Relative
train_acc_cnn_128_0.0001_0.1	0.8599	50	5.488 min
train_acc_cnn_128_0.0002_0.1	0.8965	50	5.473 min
train_acc_cnn_128_0.001_0.1	0.9271	50	5.474 min
train_acc_cnn_128_0.002_0.1	0.9354	50	7.352 min
train_acc_cnn_128_0.01_0.1	0.9058	50	5.457 min
train_acc_cnn_128_1e-05_0.1	0.6971	50	5.652 min
train_acc_cnn_128_2e-05_0.1	0.7502	50	5.573 min

Run	Value	Step	Relative
train_loss_cnn_128_0.0001_0.1	0.4268	50	5.488 min
train_loss_cnn_128_0.0002_0.1	0.3121	50	5.473 min
train_loss_cnn_128_0.001_0.1	0.2025	50	5.474 min
train_loss_cnn_128_0.002_0.1	0.1797	50	7.352 min
train_loss_cnn_128_0.01_0.1	0.2631	50	5.457 min
train_loss_cnn_128_1e-05_0.1	0.9069	50	5.652 min
train_loss_cnn_128_2e-05_0.1	0.7535	50	5.573 min

Run	Value	Step	Relative
val_acc_cnn_128_0.0001_0.1	0.7434	50	5.488 min
val_acc_cnn_128_0.0002_0.1	0.7133	50	5.473 min
val_acc_cnn_128_0.001_0.1	0.7446	49	5.364 min
val_acc_cnn_128_0.002_0.1	0.753	50	7.352 min
val_acc_cnn_128_0.01_0.1	0.7047	50	5.456 min
val_acc_cnn_128_1e-05_0.1	0.6645	50	5.652 min
val_acc_cnn_128_2e-05_0.1	0.7002	50	5.573 min

Run	Value	Step	Relative
val_loss_cnn_128_0.0001_0.1	0.7665	50	5.488 min
val_loss_cnn_128_0.0002_0.1	0.9017	50	5.473 min
val_loss_cnn_128_0.001_0.1	0.8769	50	5.473 min
val_loss_cnn_128_0.002_0.1	0.981	50	7.352 min
val_loss_cnn_128_0.01_0.1	1.279	50	5.456 min
val_loss_cnn_128_1e-05_0.1	1.0153	50	5.652 min
val_loss_cnn_128_2e-05_0.1	0.9066	50	5.573 min

结果分析

通过以上对于两个模型的实验，我们可以得出以下结论：

- 学习率的设置会影响到模型训练的收敛速度，从上图中可以很容易看出，当学习率过大时，在训练loss的角度，模型很快就会达到收敛的过程，但是收敛太快往往表现效果就不佳，甚至会出现随着训练量的提升loss反而增加的情况。
- 但是反过来，当学习率过小时，模型的收敛速度过慢，甚至在上文实验中出现了50个epoch仍然没有收敛的情况。同时，过小的学习率也很容易导致模型陷入局部最优中。
- 学习率影响着参数更新的速度，较大的学习率将导致较大的参数更新，而较小的学习率将导致较小的参数更新，太大或太小的学习率都可能导致模型性能下降。在实验中需要选择合适的学习率可以确保模型参数在训练中能够适当地调整。