ANN-HW-1

计11 周韧平 2021010699

实验概括

本次实验为实现多层感知机，并在 MINIST 数据集上进行训练验证

多层感知机实现分为线性层、激活函数和损失函数，代码中定义了三部分的前向和后向传播过程。

具体来说，本次实验室实现了一下模块

线性层 `Linear`：实验中，数据 x 被组织成 $(batch\_size, in\_dims)$ 的形式输入，经过线性层 W 计算得到u

$u = xW^T + b$ (这里和讲义略有不同，从线性代数定义来说应该有一个转置)

激活函数：本次实验中我实现了 `Selu`, `Swish`, `Gelu` 三种激活函数，加上给的激活函数 `Relu`, `Sigmoid` 共五种

损失函数：本次实验中我实现了基本功能要求的 `MSELoss`， `SoftmaxCrossEntropyLoss` 和 `HingeLoss` 函数，此外，我还实现了 `FocalLoss` 函数

本次实验中我使用的是自己的笔记本，4核处理器，型号为11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz

实验结果

经过多组超参数组合的尝试，最终本实验中超参数设定为

```
{
    learning_rate = 1e-4,
    weight_decay = 1e-3,
    momentum=0.9,
    batch_size=100,
    max_epoch=100,
    dispfreq=50,
    test_epoch=5
}
```

如果不是特别提及对超参数的探究，都采用上述设定。

单隐藏层实验

本节中，我将线性层设定为 (784,784),(784,10) 不变，探究不同激活函数和损失函数对训练效果的影响

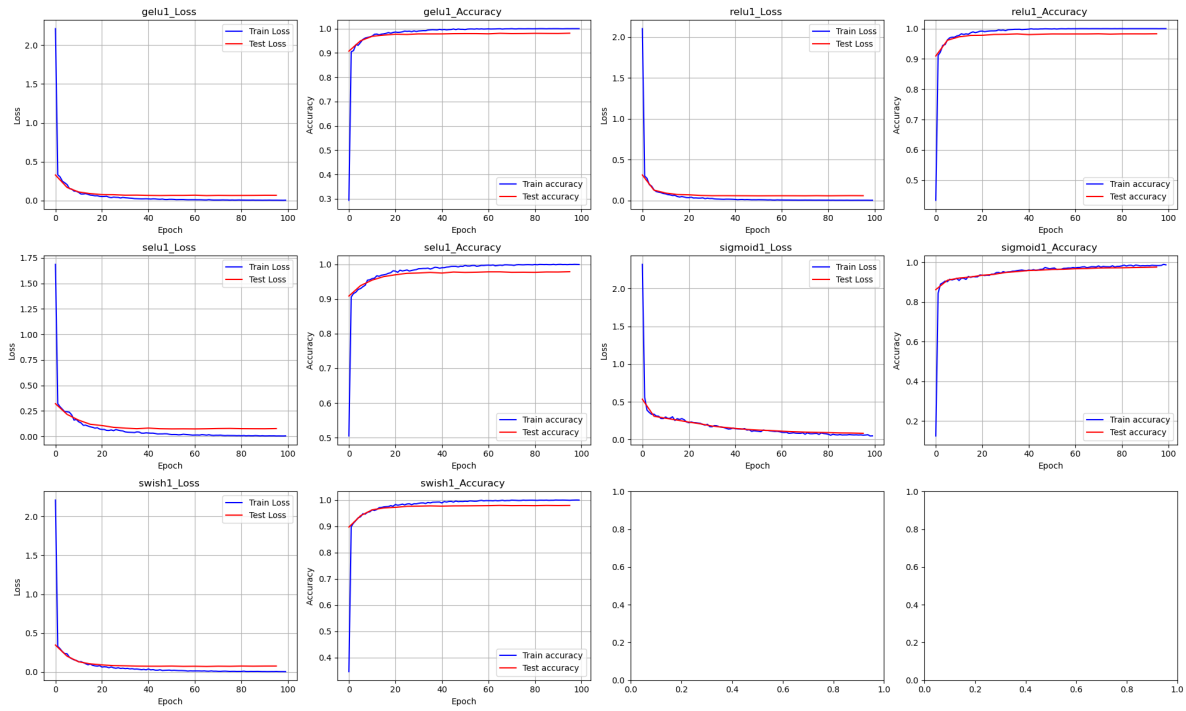
激活函数

本小节中，我用 `SoftmaxCrossEntropyLoss` 作为损失函数，通过调节隐藏层的激活函数 `Selu`, `Swish`, `Gelu` , `Relu`, `Sigmoid` 得到结果如下

激活函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
Selu	0.0045	0.9998	0.0771	0.9788	1523

激活函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
Swish	0.0057	0.0769	0.9996	0.9795	1383
Gelu	0.0042	0.0668	0.9994	0.9814	1749
Relu	0.0026	0.0584	1.0000	0.9831	1155
Sigmoid	0.0603	0.0820	0.9830	0.9758	1252

可视化结果



实验分析

- 单层隐藏层训练稳定，且在超参数设置合适情况下，所有激活函数设定都表现良好，能得到97%以上的准确率结果
- 在所有激活函数中，sigmoid 收敛最慢，且最终在测试集上的准确率也较低
- 从训练用时来看，相同 Epoch 下 GELU 函数用时最长，RELU 最短。分析原因可能是在后向传播时 RELU 函数本身比较简单（大于0返回1，小于0返回0），而 GELU 不仅公式复杂，还参照讲义指导，采用了微元方法求解梯度，而其他激活函数用的都是解析解求梯度，因此导致其速度较慢。

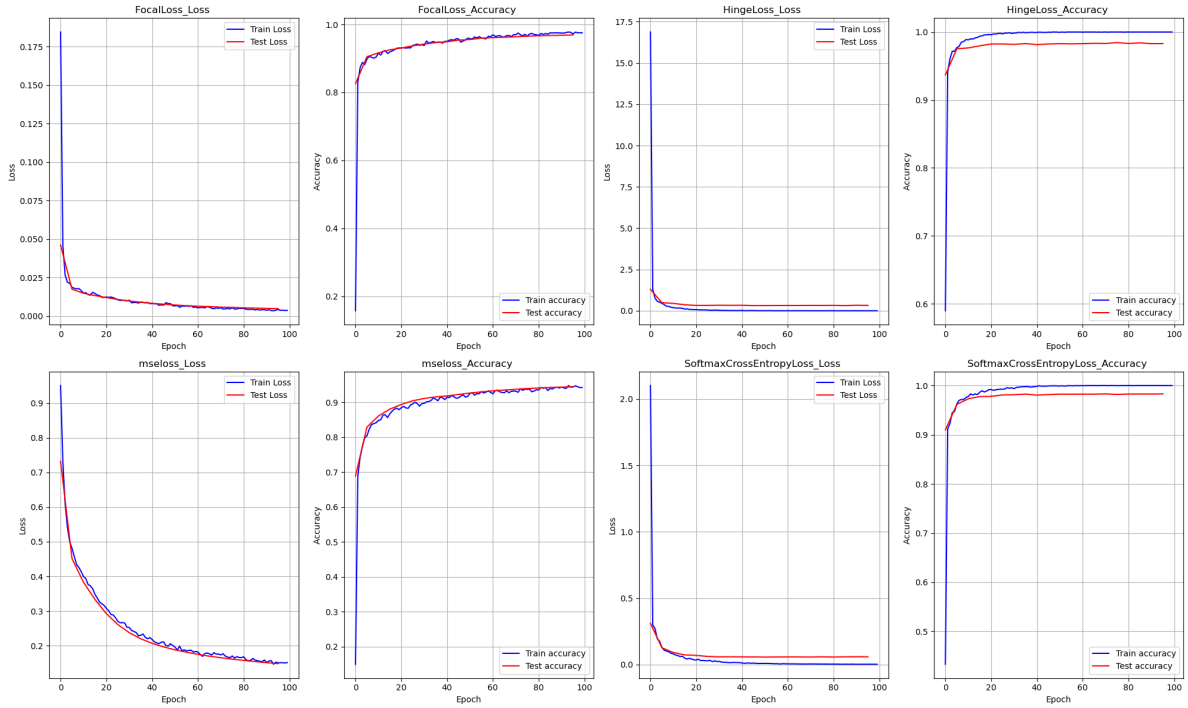
损失函数

本小节中，我用 `relu` 作为隐藏层的激活函数，通过调节损失函数 `MSELoss`，`SoftmaxCrossEntropyLoss`， `HingeLoss` 和 `FocalLoss` 得到结果如下

损失函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
`MSELoss`	0.1520	0.1482	0.9442	0.9444	1172
`SoftmaxCrossEntropyLoss`	0.0026	0.0584	1.0000	0.9831	1155
`HingeLoss`	0.0010	0.3235	1.0000	0.9830	1101

损失函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
`FocalLoss`	0.0031	0.0048	0.9790	0.9691	1105

可视化结果



实验分析

- `HingeLoss` 和 `SoftmaxCrossEntropyLoss` 效果由于另外两个损失函数，能够达到98%以上的准确率
- `MSELoss` 效果较差，且通过和其它激活函数组合实验得到结果表明，`mseloss` 在和 `swish` 激活函数组合时仅能得到 89.1% 的准确率，而在与 `selu` 激活函数组合时仅能得到 91.8% 的准确率。分析其原因可能是这样：考虑到这是一个10分类任务，而 `mseloss` 的设计本身并没有考虑结果在（0，1）之间这个特性，因此相比交叉熵这类做过归一化操作的损失函数设计，在该任务上表现较差

双隐藏层实验

本节中，我将线性层设定为 (784,784),(784,784),(784,10)，并保持其它超参数不变，将单隐藏层中的所有实验重新进行一遍（两层隐藏层采用相同的激活函数），探究不同两层隐藏层下激活函数和损失函数对训练效果的影响，为了便于对比，我将一层隐藏层的数据也列在下面

单层隐藏层

激活函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
`Selu`	0.0045	0.9998	0.0771	0.9788	1523
`Swish`	0.0057	0.0769	0.9996	0.9795	1383
`Gelu`	0.0042	0.0668	0.9994	0.9814	1749

激活函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
`Relu`	0.0026	0.0584	1.0000	0.9831	1155
`Sigmoid`	0.0603	0.0820	0.9830	0.9758	1252

双层隐藏层

激活函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
`Selu`	0.0007	0.0932	1.0000	0.9797	2984
`Swish`	0.0016	0.1119	0.9994	0.9785	2610
`Gelu`	0.0044	0.1026	1.0000	0.9802	3400
`Relu`	0.0003	0.0701	1.0000	0.9819	2308
`Sigmoid`	0.0456	0.0840	0.9868	0.9749	2371

单层隐藏层

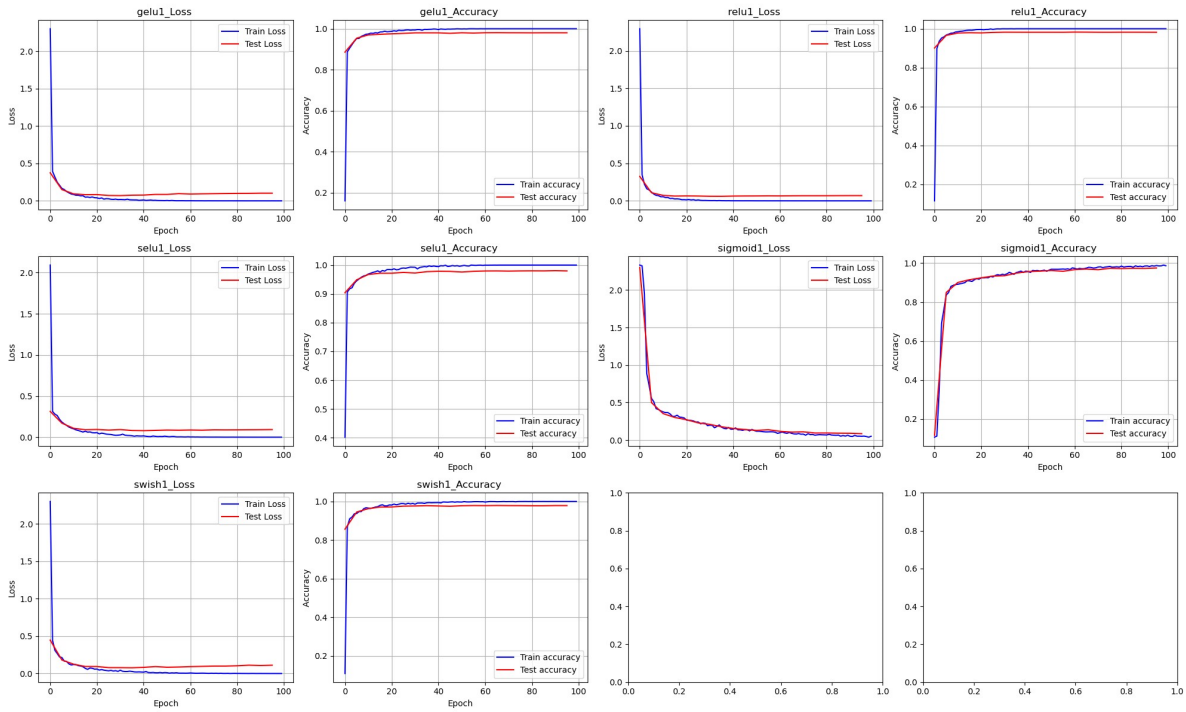
损失函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
`MSELoss`	0.1520	0.1482	0.9442	0.9444	1172
`SoftmaxCrossEntropyLoss`	0.0026	0.0584	1.0000	0.9831	1155
`HingeLoss`	0.0010	0.3235	1.0000	0.9830	1101
`FocalLoss`	0.0031	0.0048	0.9790	0.9691	1105

双层隐藏层

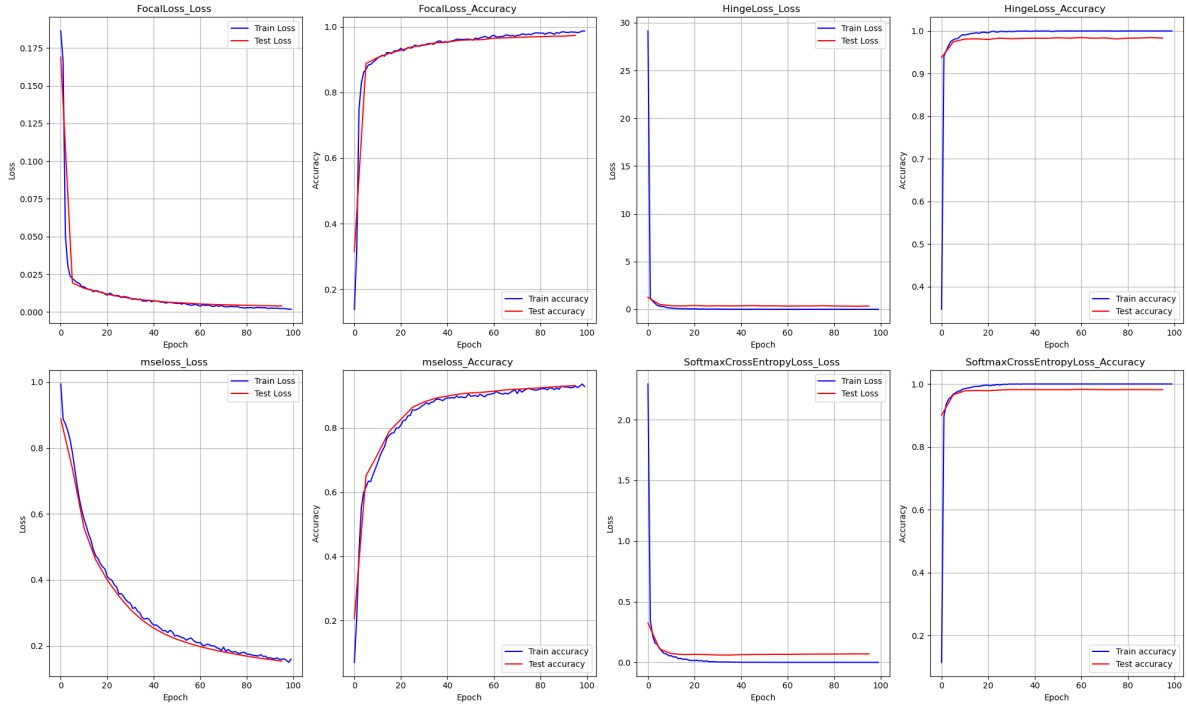
损失函数	train loss	test loss	train accuracy	test accuracy	训练用时（s）
`MSELoss`	0.1481	0.1533	0.9334	0.9320	2235
`SoftmaxCrossEntropyLoss`	0.0003	0.0701	1.0000	0.9819	2308
`HingeLoss`	0.0028	0.3512	0.9998	0.9831	2207
`FocalLoss`	0.0025	0.0040	0.9818	0.9738	2246

可视化结果

不同激活函数对比



不同损失函数对比



实验分析

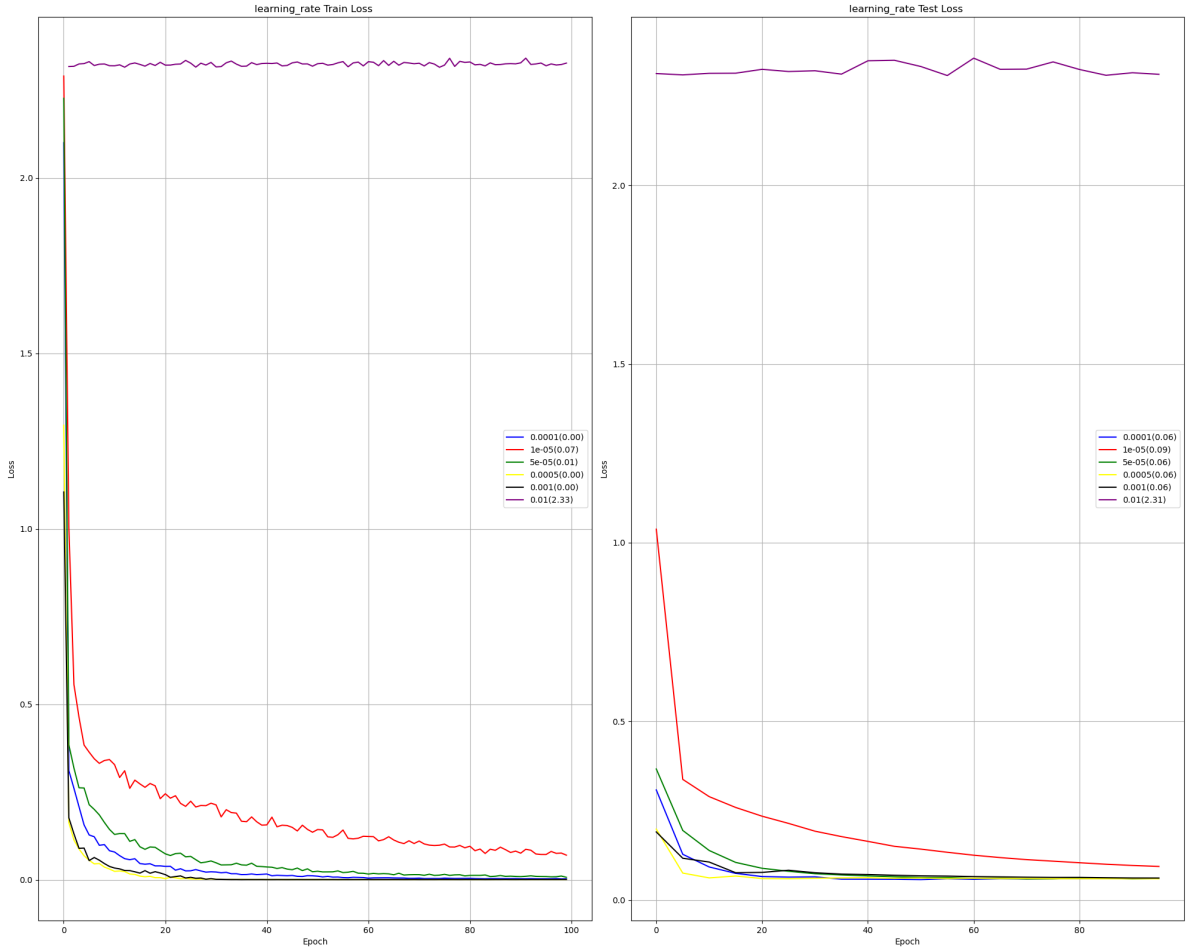
- 和单隐藏层相比，双隐藏层设定在该分类问题上并没有明显提升，这可能是两个原因导致：1.单隐藏层实验并没有体现出明显的过拟合现象，而双层隐藏层相比单隐藏层有更多参数，能够处理更复杂的数据这一特点并没有体现出来。2.本实验仅对隐藏层设定做了复制粘贴的操作，可能需要更加精心设计的网络结构以及不同的激活函数搭配才能使网络实现更高的准确率
- 和单隐藏层相比，双隐藏层增加了训练的时间复杂度，约为单隐藏层的两倍，这无疑大大增加了训练的成本。考虑到准确率没有明显的提升，在当前参数设定下，增加网络深度的设计对于该任务并没有太大的意义。

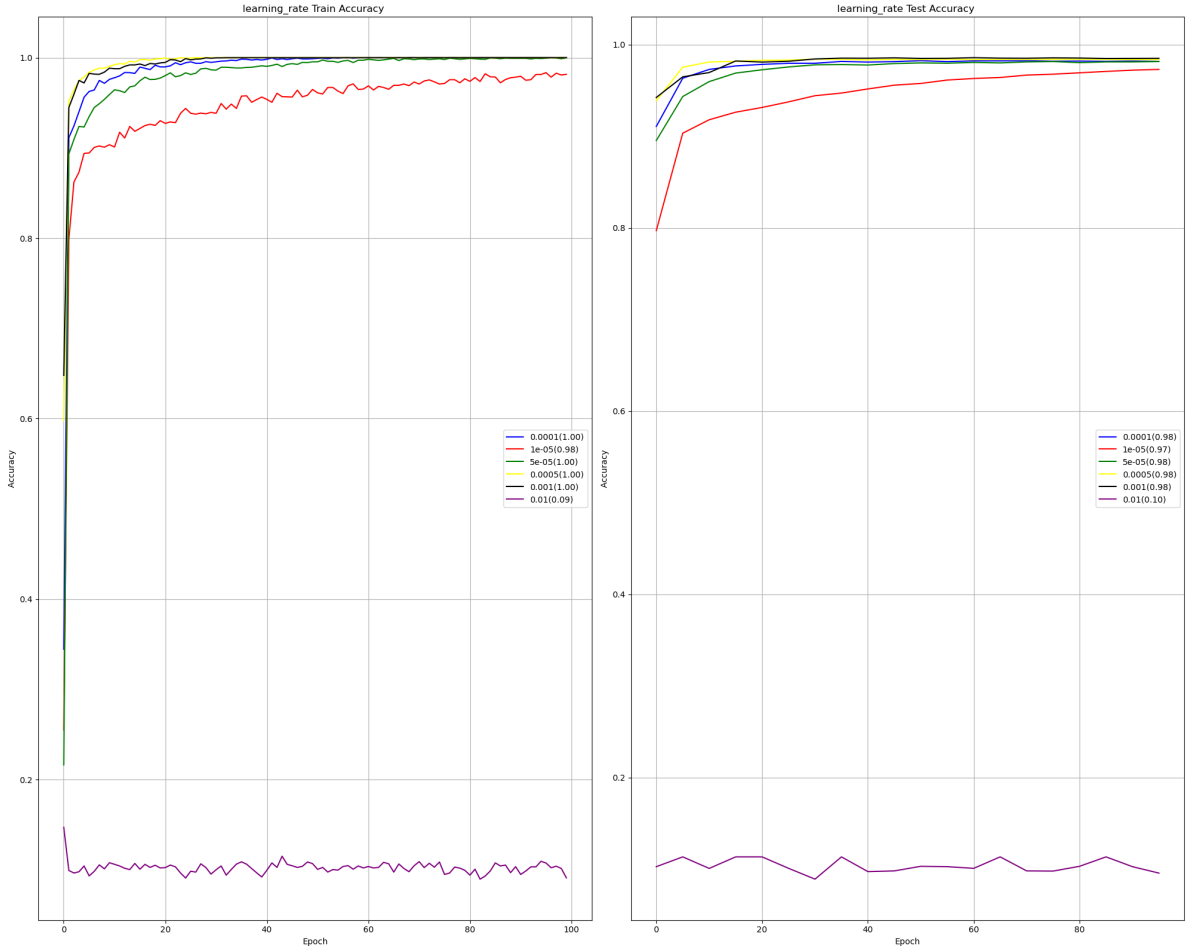
超参数实验

在本节中，我选用本大节开始提到的超参数作为 baseline，并使用单隐藏层 + `RELU` + `SoftmaxCrossEctropy` 的网络结构，选用该组设定的原因是在之前的实验中这组设定具有较高的稳定性和较快的训练速度。在此基础上，本小节中我探究了学习率 (learning rate)，动量 (momentum)，权重衰减 (weight decay)，一次训练所抓取的数据样本数量 (batch size)，隐藏层神经元个数 (hidden size) 这些超参数设定对模型训练的影响

学习率 (learning rate)

在学习率取 $0.01, 0.001, 0.005, 1e-4, 5e-5, 1e-5$ 下，训练过程如下



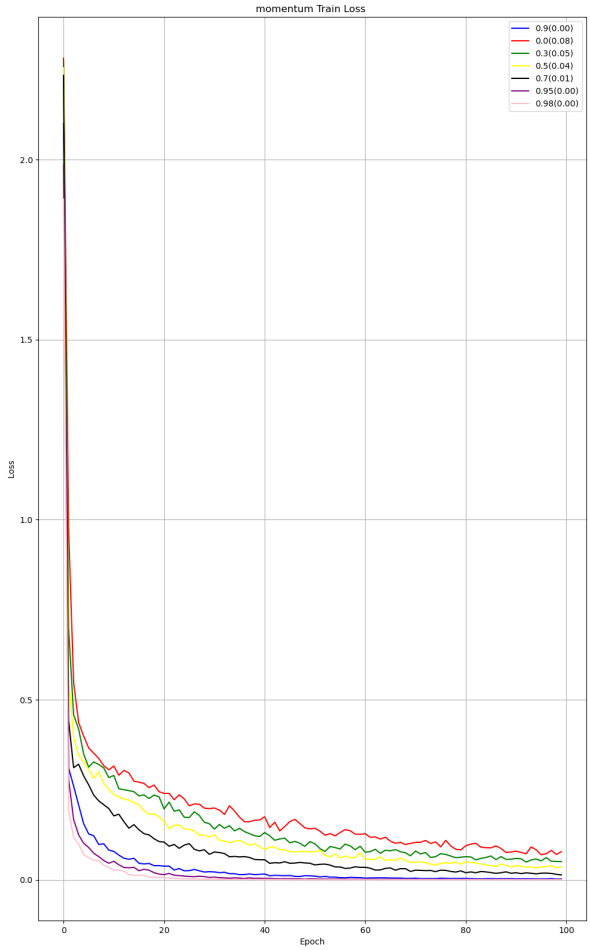


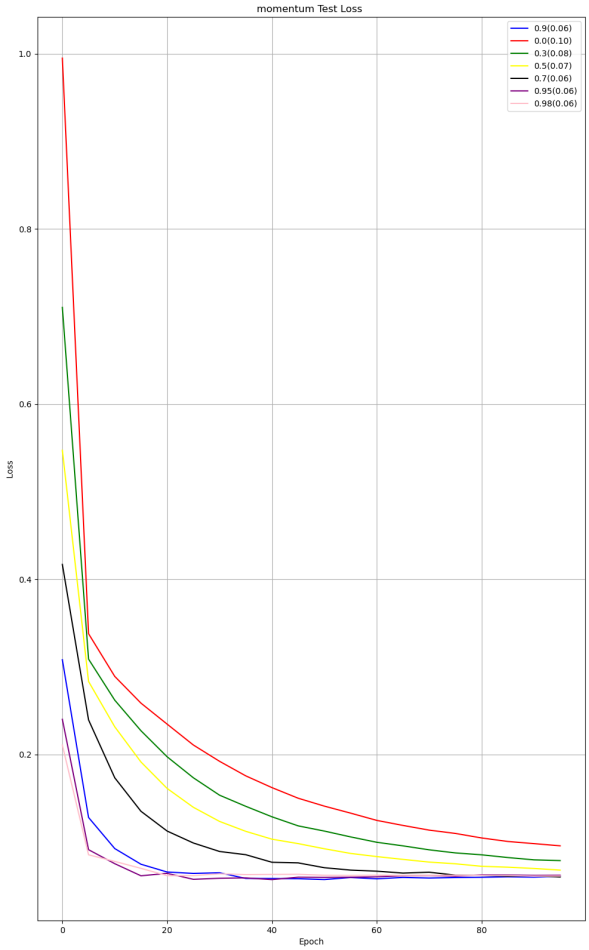
- 学习率小的模型收敛速度慢且准确率较低，这是因为学习率小的模型每次梯度更新小，同时不易逃离鞍点或局部最小点。

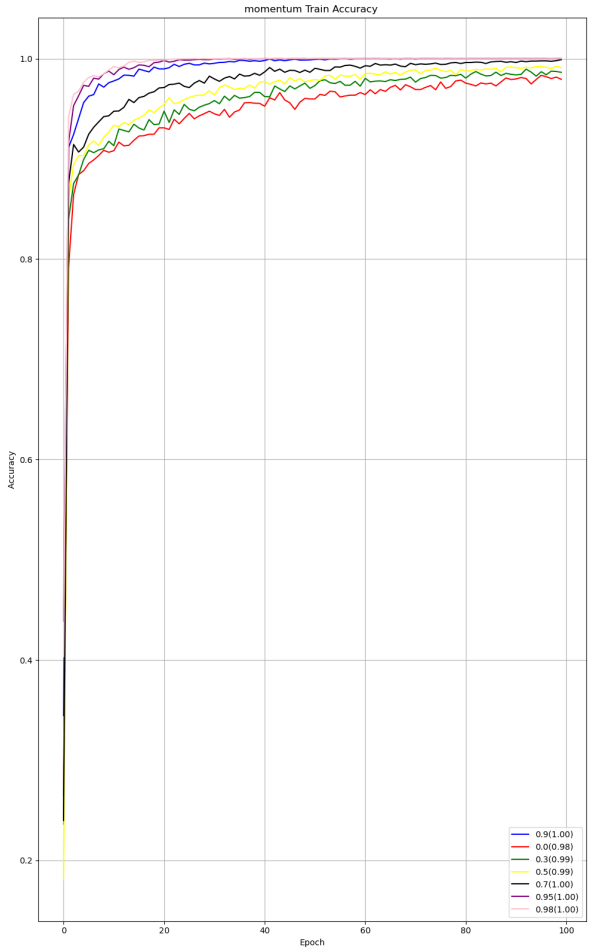
- 在 $[1e-5, 1e-3]$ 范围内模型在训练集和测试集上的收敛速度随着学习率的增加而增加，但在 $[5e-5, 1e-3]$ 这一区间内，最终收敛的准确率基本一致。这可能是因为在学习率不是很小的情况下，模型都能逃离局部极小点或者鞍点。从训练成本来看，较大的学习率可以更快的收敛到较高的准确率，这使得模型更加高效
- 在学习率很大时（$1e-2$），模型的loss并不会下降，很难接近极小值，这时就需要考虑调低学习率，或者调整网络设定

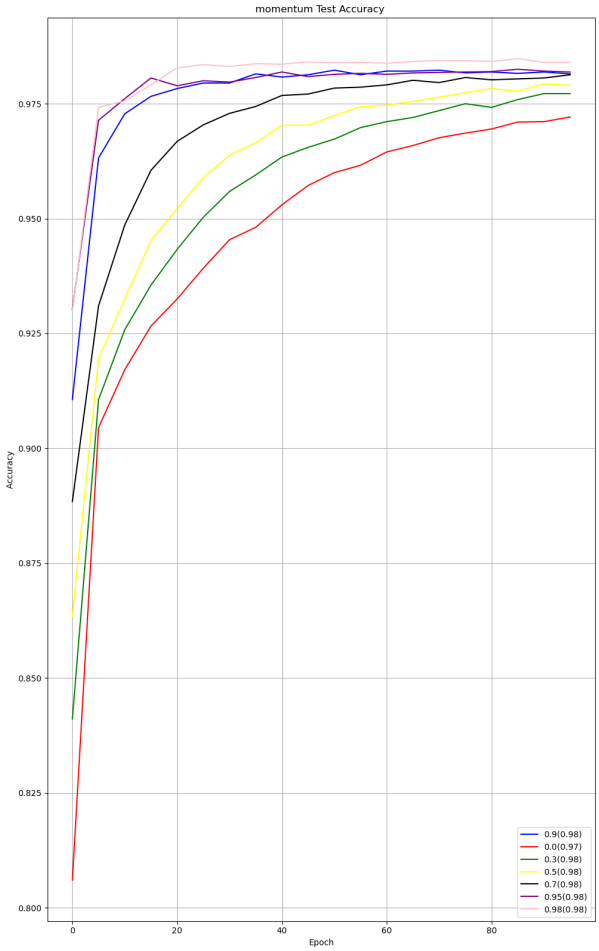
动量 (momentum)

在动量取 $0, 0.3, 0.5, 0.7, 0.95, 0.98$ 下，训练过程如下





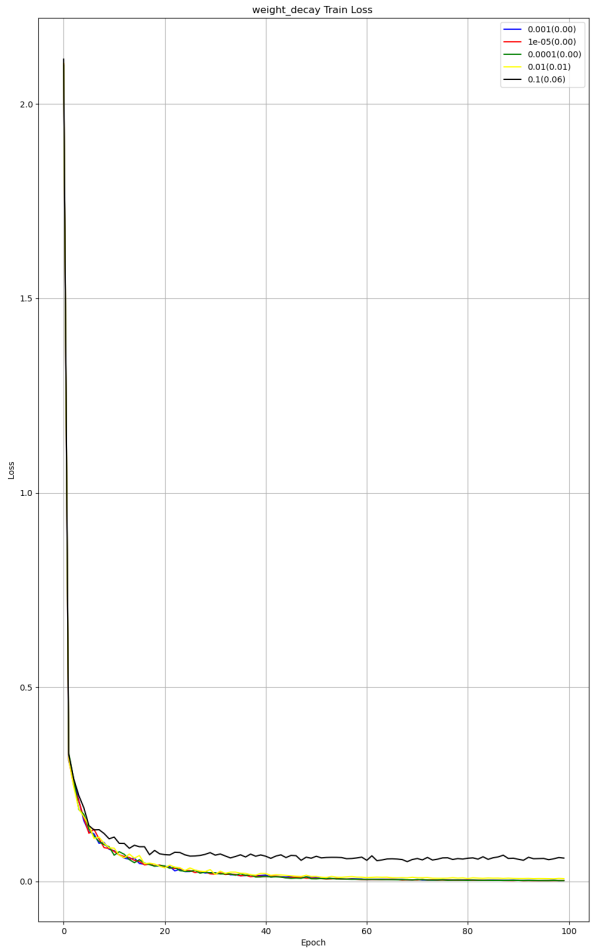


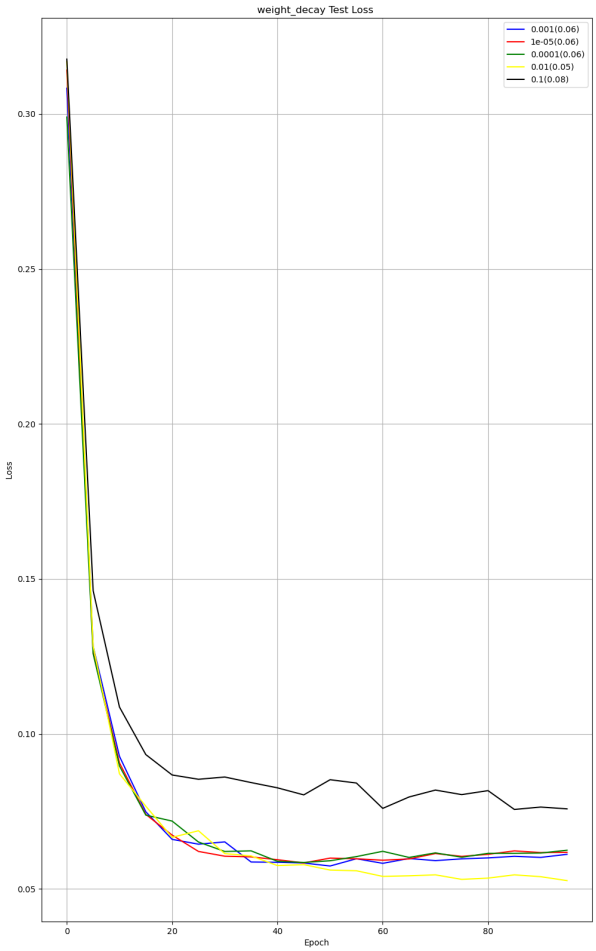


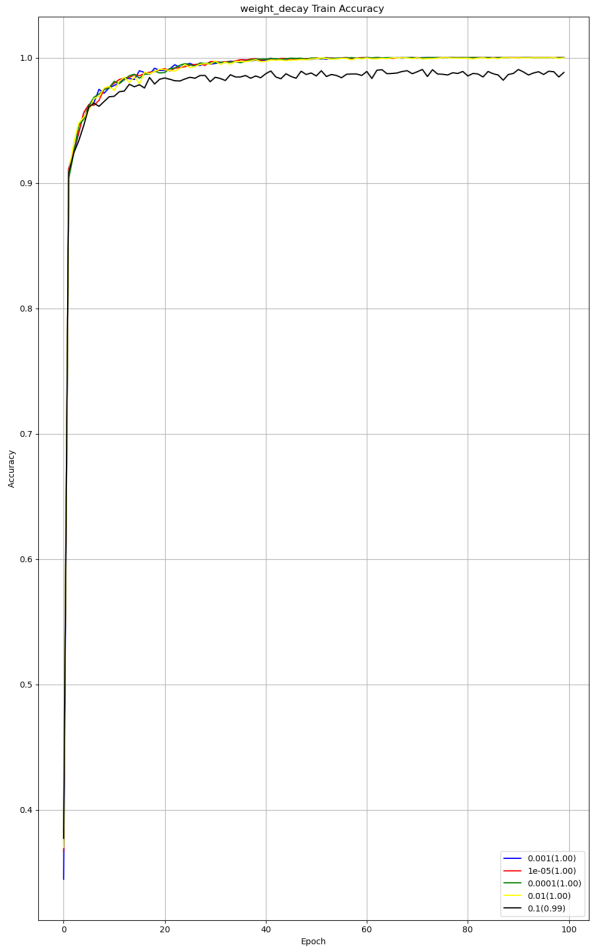
- 随着动量的增加，模型收敛变得更快，且在最终准确率上也有一定的增长。这是由于动量本质上是对多次的梯度进行了指数级的加权，在加权求和的情况下，有利于收敛的分量得到放大，而不利于收敛的分量在一定程度上被多次梯度相互抵消，从而加快模型的收敛，使其更快的走向全局最小点。

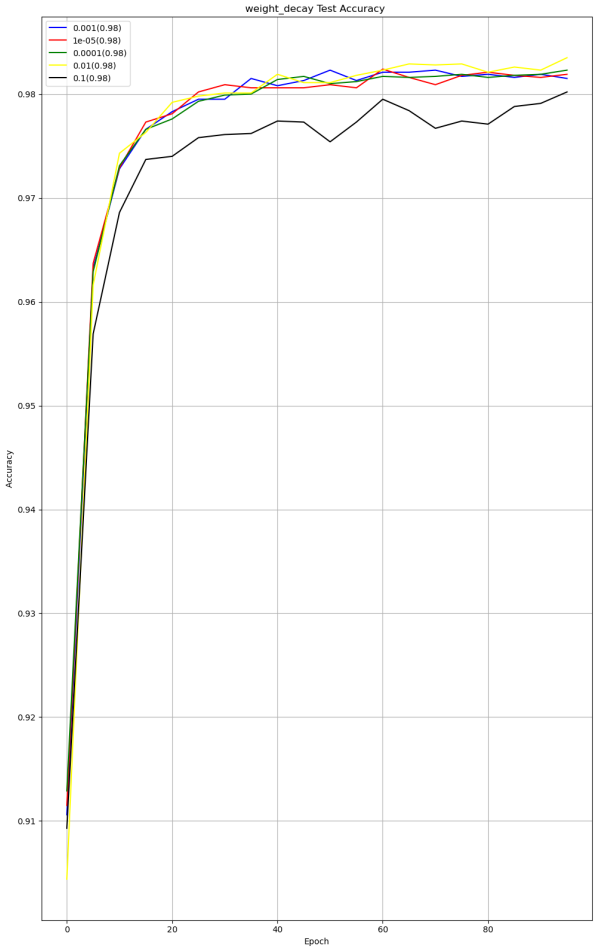
权重衰减 (weight decay)

在权重衰减取 $0.1, 0.01, 0.001, 0.0001, 1e-5$ 下，训练过程如下





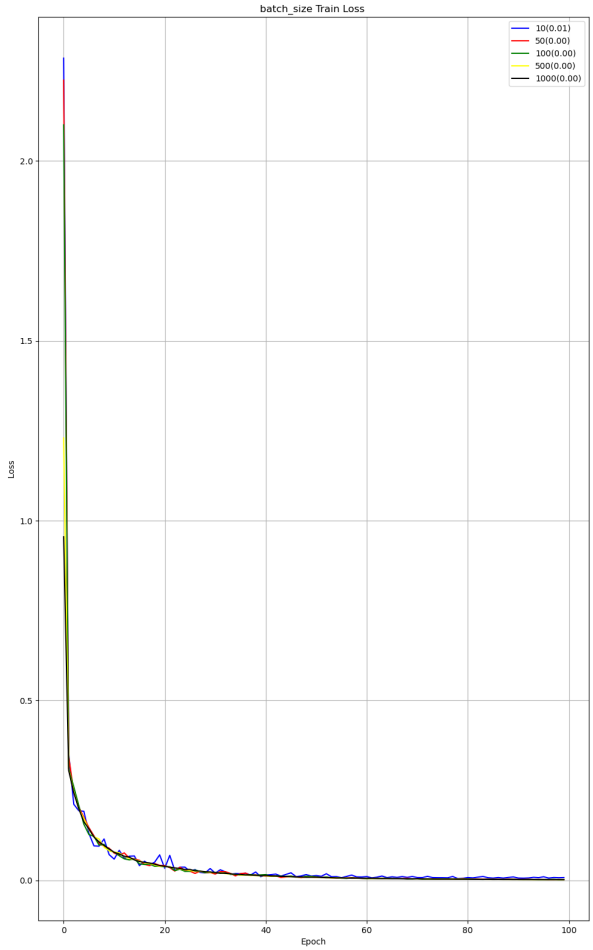


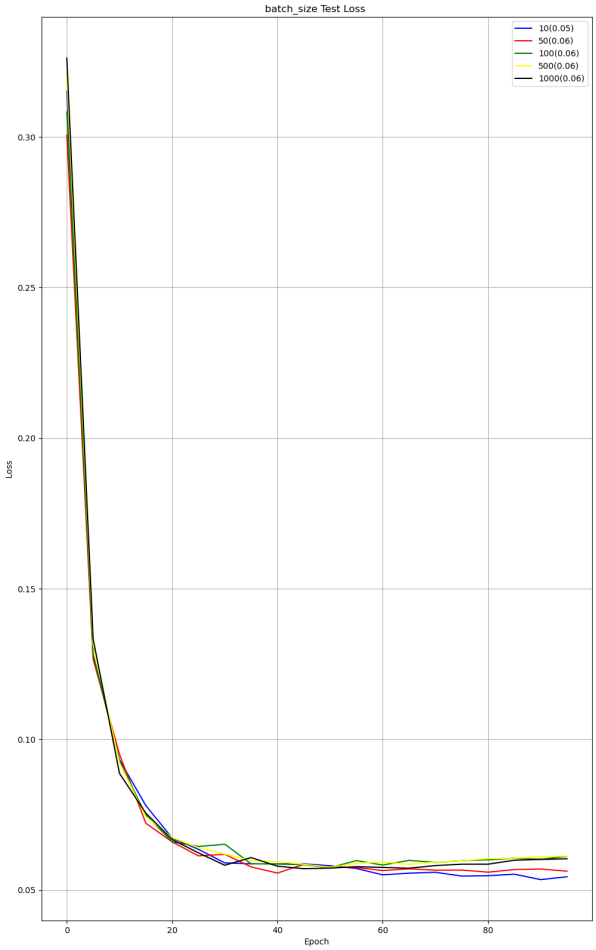


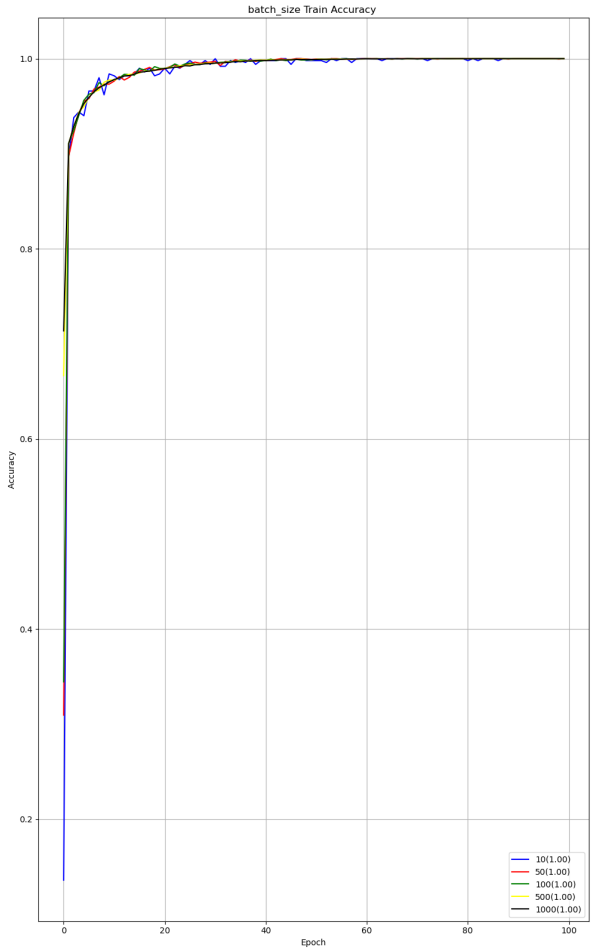
- weight decay较大（0.1）时，模型准确率明显降低，这是由于作为 baseline 的网络复杂性较低，而 weight decay 作为一种正则化手段，过高的权重衰减会大大降低模型的复杂性，最终限制模型的认知能力。
- weight decay较小时，模型普遍表现较好，随着 weight decay 增加模型有轻微的过拟合现象出现，但对整体的准确率影响不大。这是因为本任务数据量充足，对于模型的泛化性要求不是很高，因此 weight decay 并没有体现出明显的作用。

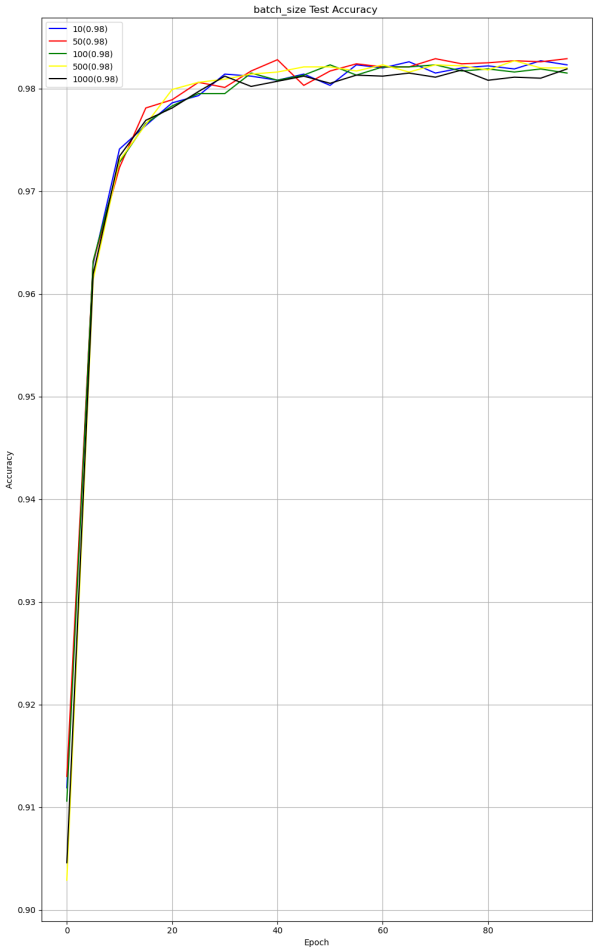
一次训练所抓取的数据样本数量 (batch size)

在一次训练所抓取的数据样本数量取 $10, 50, 100, 500, 1000$ 下，训练过程如下





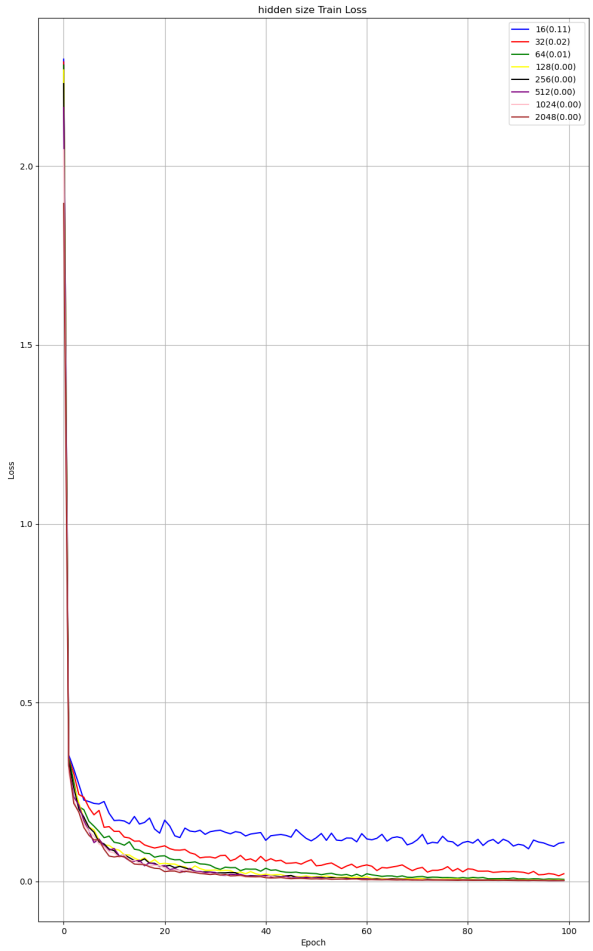


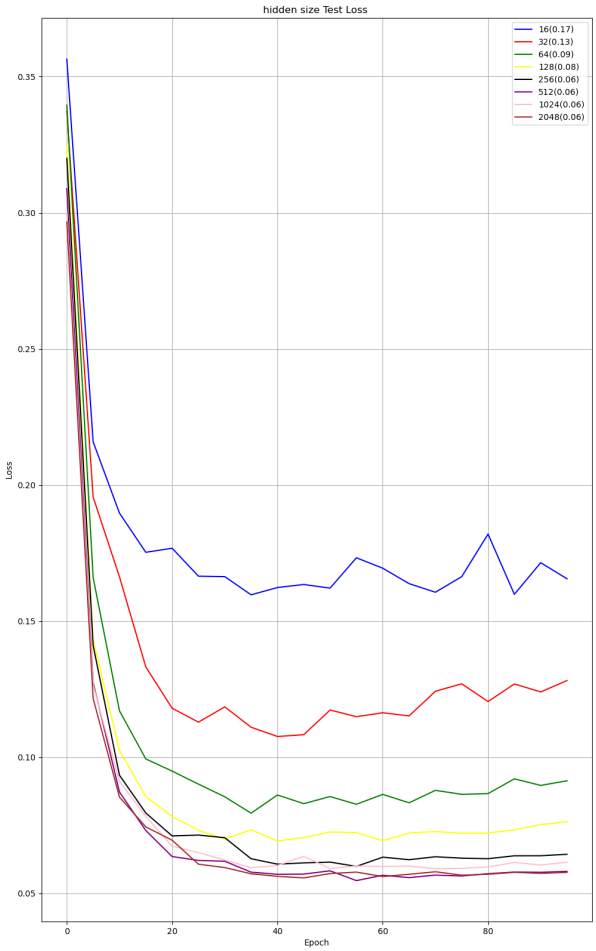


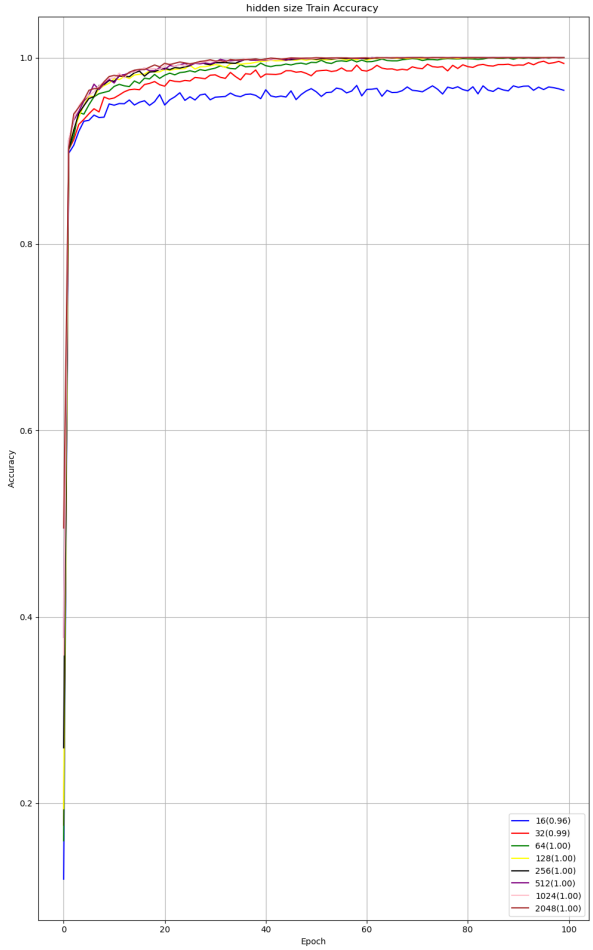
- batch size 较小时，模型训练时的震荡明显变大（$bs = 10$），这是由于每次梯度下降时随机性较大，模型受到了更多的噪声影响。但整体来看，最终模型都成功收敛到了较高的准确率上，这可能是因为数据集本身的随机噪声并不是很大，单次训练的样本量即使较少，也能和整体数据集的特征分布较为一致，并没有出现随机噪声使得模型完全无法收敛的情况。

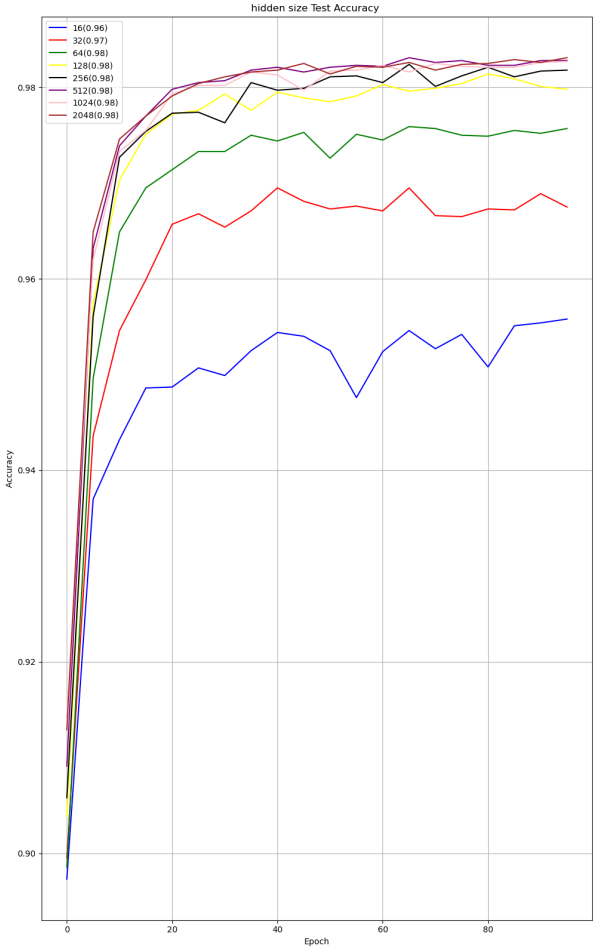
隐藏层神经元个数 (hidden size)

在隐藏层神经元个数取 $16, 32, 64, 128, 256, 512, 1024, 2048$ 下，训练过程如下

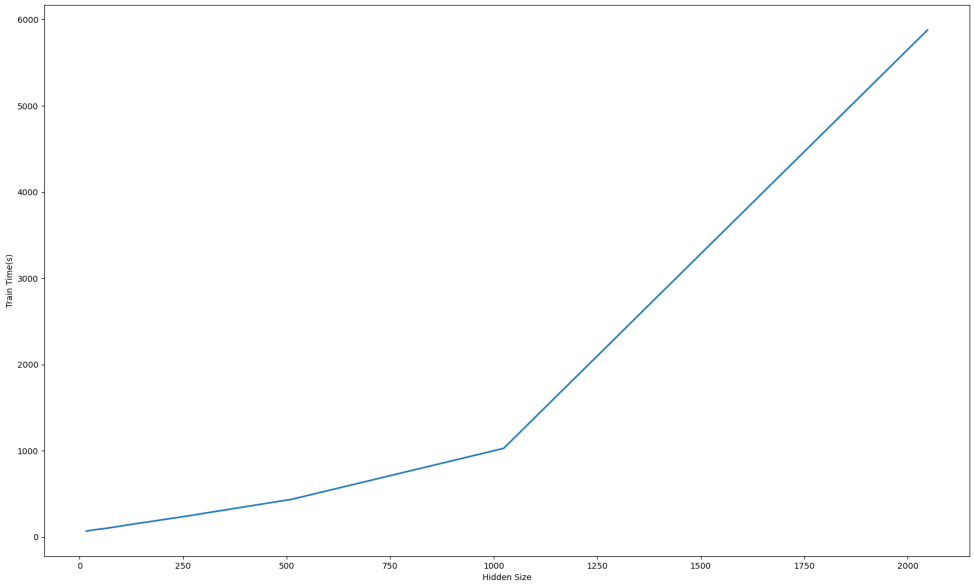








- 在 $[16, 512]$ 范围内，随着隐藏层的增加，模型收敛更快且准确率更高，这是由于在架构相同的情况下，更多的参数有利于增强模型对数据的认知能力，学到更多的数据集特征
- 在 $[512, 2048]$ 范围内，模型准确率不再变化，且训练时间成本随着隐藏层神经元数增加而呈倍增长（如下图），这是因为随着隐藏层神经元增加，模型复杂度不再是限制模型认知能力的因素，而过多的神经元会增加计算成本，从而导致训练时间的增加

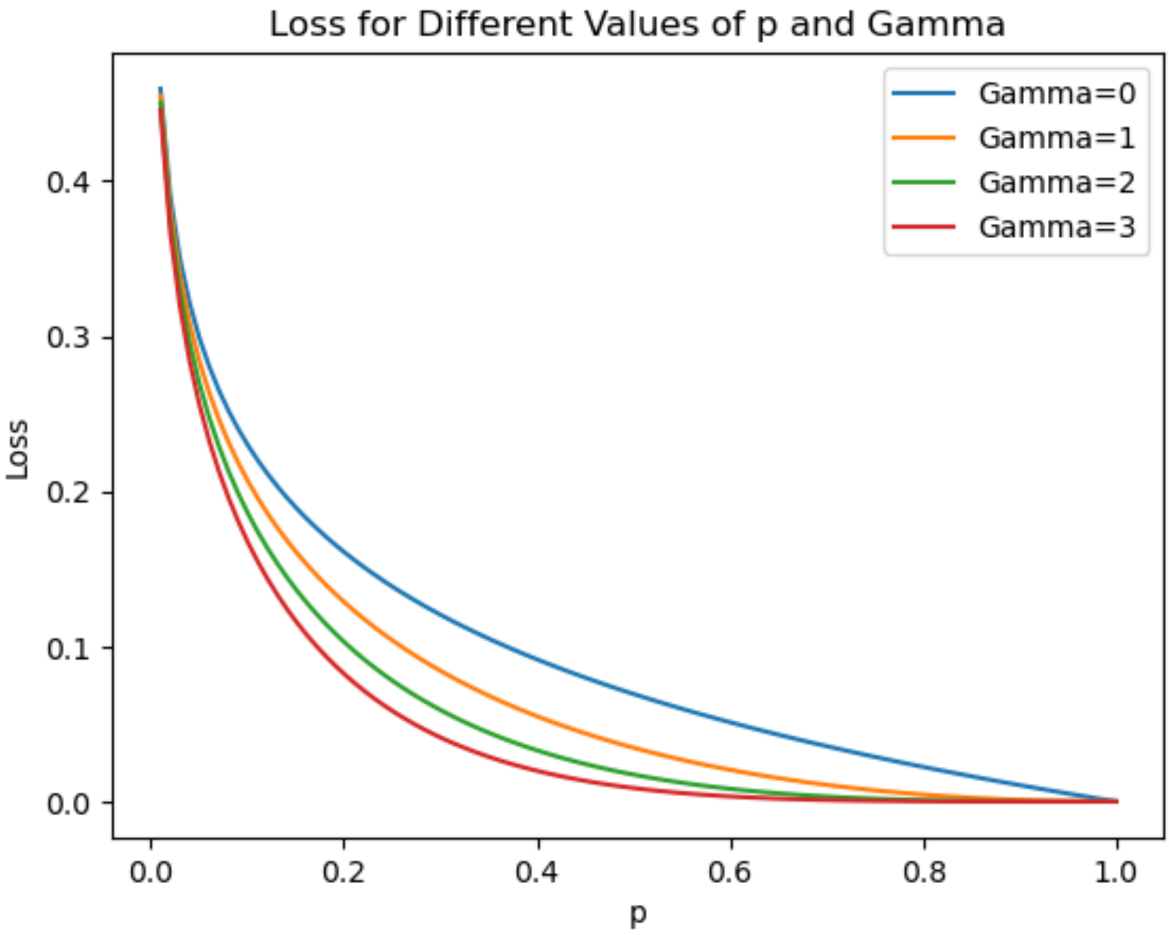


讨论

Focal Loss vs Soft max Cross Entropy Loss

Focal Loss 的提出是为了解决训练中样本不均衡的问题，其核心思路为：样本非平衡造成的结果是样本量较少的类别不容易分类正确，也就是所谓的“难分类样本”，而如果能让模型聚焦于这些难分类的样本，可以进一步提升整体的分类准确率。和交叉熵函数相比，Focal Loss 引入了 modulating factor $(1-h_k^{(n)})^{\gamma}$，当处理易分类样本时，$h_k^{(n)}$ 作为置信度衡量的指标，$h_k^{(n)} \to 1$，此时损失函数值中这一项会很小，因而其在模型训练中造成的影响也会比较小，反之，如果样本不易分类，其对loss的贡献较大，因此更容易左右模型训练的结果。

基于作业要求中给出的公式，我绘制了在十分类任务中不同超参数 γ 设定下，$loss$ 和真实标签置信度 p 的关系，可以看到，随着 γ 的增大，置信度高的样本的 $loss$ 与置信度低的样本的 $loss$ 之比在下降，这也形象的说明了为什么置信度低的样本会对模型训练产生更大的影响。



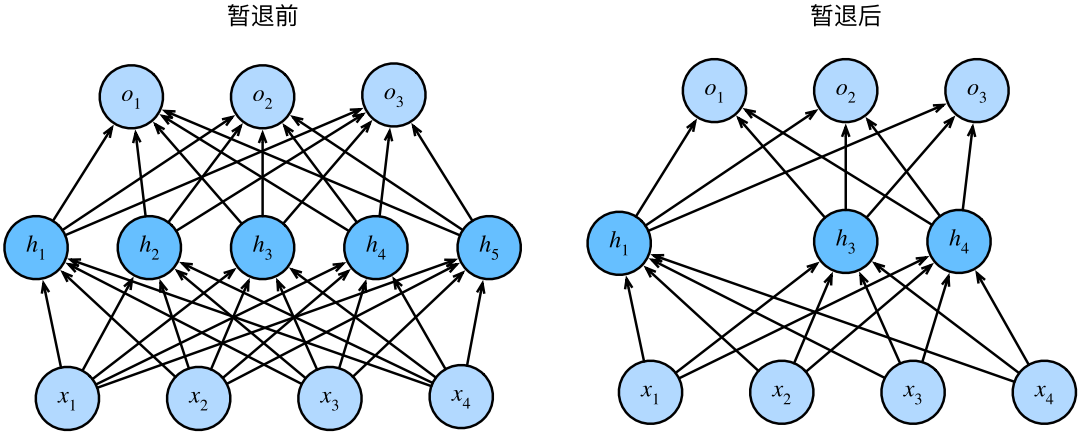
避免过拟合方法

过拟合是指模型在训练集上表现很好但泛化性能很差的情况，具体表现为损失函数很小但测试集上表现并没随着训练集损失函数下降而提升，避免过拟合的方法有很多，包括对数据集进行特征提取，对模型实行正则化等，本此实验中，我才用了暂退法 (Dropout) 来探究避免过拟合的方法。

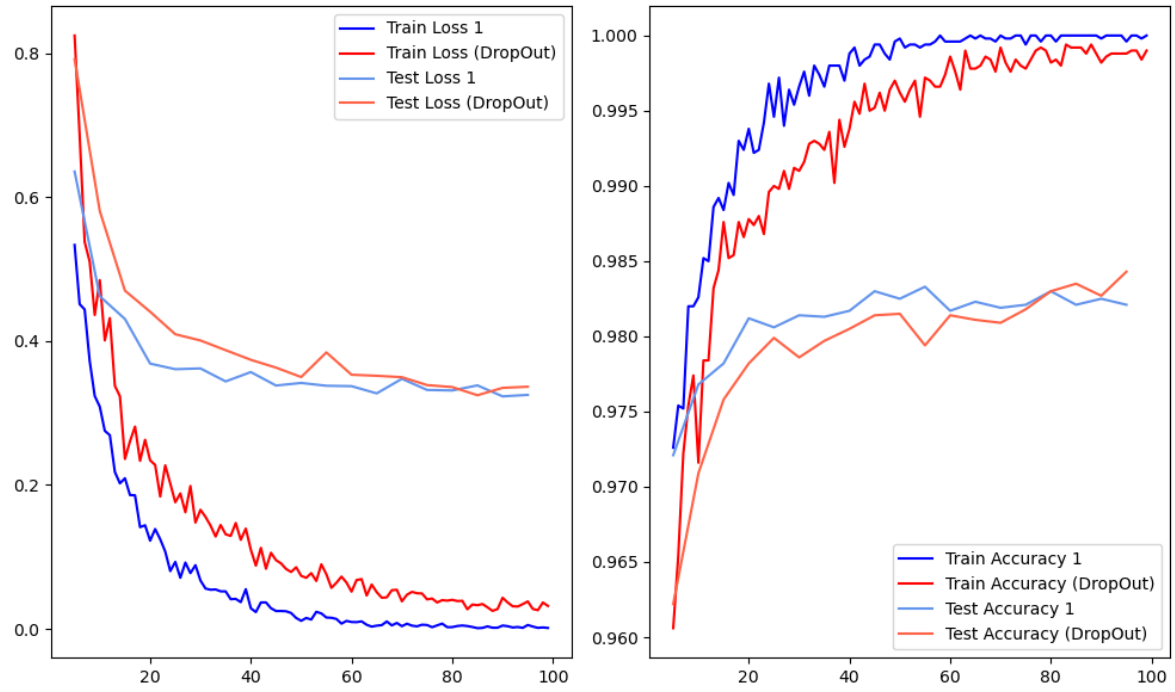
暂退法是指在前向传播过程中，计算每一内部层的同时注入噪声，具体的操作是在训练过程中丢弃（drop out）一些神经元。在整个训练过程的每一次迭代中，标准暂退法包括在计算下一层之前将当前层中的一些节点置零，从而达到只激活一部分神经元的目的。

具体来说，在具有隐藏层的神经网络中，每轮训练中对于隐藏层 h 令

$$h_i(x) = \begin{cases} 0 & \text{概率为} 1-p \\ h_i(x) & \text{概率为} p \end{cases}$$



在单隐藏层，`learning_rate` = 5e-5, `weight_decay` = 1e-3, `momentum` =0.9, `batch_size` =100, `max_epoch` =100, `dispfreq` =50, `test_epoch` =5， `dropout_p` =0.8 条件下，使用 Relu 激活函数和 Focal loss 损失函数（针对这一组合进行探究的主要原始之前的实验结果中出现了轻微过拟合的现象）下实验结果如下，为了能更好的看到训练的细微差别，下面从第5个epoch开始绘制训练结果，可以看到，使用暂退法后训练集loss以及accuracy的收敛速度明显变慢，但在测试集上得到了更好的结果（提升大约0.3%），也大大缩小了模型在测试集合和训练集上的差距（降低0.5%），一定程度上起到了避免过拟合的目的



参考文献

Focal Loss for Dense Object Detection [1708.02002] Focal Loss for Dense Object Detection (arxiv.org)

4.4. 模型选择、欠拟合和过拟合 — 动手学深度学习 2.0.0 documentation (d2l.ai)