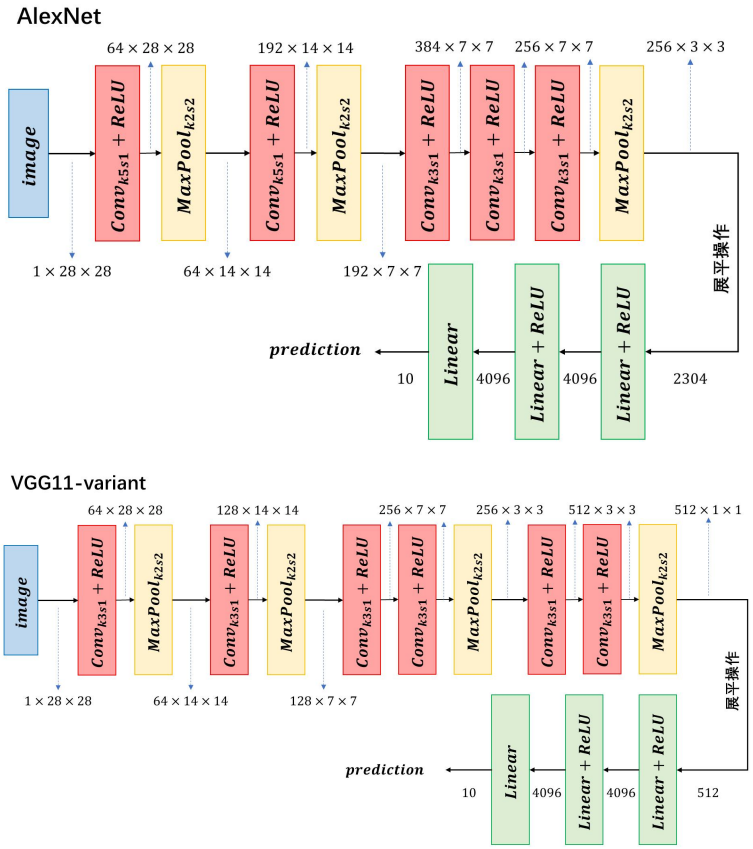
Homework5 卷积神经网络

3220104685 赵一锟

1 实验要求

经过课堂上的学习和课后的阅读，相信大家对卷积神经网络的构建和训练有了一定的了解。在本次作业中，我们将尝试使用 PyTorch 构建和训练两个经典的卷积神经网络 AlexNet 和 VGGNet。



2 实验原理

2.1 卷积神经网络

2.1.1 AlexNet

AlexNet 是一个深度卷积神经网络，包含 8 个学习层：5 个卷积层 和 3 个全连接层。它能够自动学习图像从低级特征（边缘、纹理）到高级语义特征（物体部件、整体对象）的层次化表示，最终用于图像分类。

AlexNet 采用 ReLU。ReLU 计算简单 (f(x) = max(0, x))，在正区间梯度恒为 1，有效缓解了梯度消失问题，显著加快了训练速度（比使用 tanh 快数倍），并允许训练更深的网络。

2.1.2 VGGNet

VGGNet 是一个深度卷积神经网络，包含 9 个学习层：6 个卷积层 和 3 个全连接层。其核心设计理念是：使用堆叠的小型 3x3 卷积核构建深度网络。它证明了深度（即使是相对适度的增加）对提升 CNN 性能的有效性，其简洁、规则、模块化的结构使其成为理解 VGGNet 思想和卷积神经网络基础架构的绝佳范例。

2.2 损失函数

2.1.1 均方误差损失（MSE）

均方误差是机器学习和统计建模中常用的损失函数，特别适用于回归问题。它通过计算预测值与真实值之间差异的平方平均值，来衡量模型的预测性能。MSE 的计算公式为：

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

由于平方操作，MSE 对较大的误差更加敏感，有助于模型重点优化这些误差，与此同时，它还是一个连续且可导的函数，适合使用梯度下降等优化算法进行模型训练，但其对离群点（异常值）非常敏感，可能导致模型过于关注这些点，影响整体性能。

2.1.2 交叉熵损失（cross_entropy）

交叉熵损失是深度学习中分类问题常用的损失函数，特别适用于多分类问题。它通过度量预测分布与真实分布之间的差异，来衡量模型输出的准确性。交叉熵损失的计算公式为：

$$\mathrm{Loss} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

相对于均方误差损失主要用于回归问题，对异常值敏感，不适合分类任务；交叉熵损失函数可以通过度量预测分布与真实分布之间的差异，有效指导模型在分类任务中学习更准确的概率分布。

3 模型参数调整与比较

在完成 VGG 神经网络搭建代码补全及测试过程代码补全后，我们进入训练参数选取及模型比较的部分。本实验使用 3080Ti 进行模型训练，单个 epoch 训练时长在 11s 左右。以下为全部的实验训练参数及训练结果，下面我将逐个参数进行分析，并对最终的训练模型进行比较。(注：所有序号均按照 log 的时间顺序排列)

AlexNet交叉熵损失									
实验次数\参数	batch_size	epochs	learning_rate	optimizer	dropout_prob	学习率调节器	准确率	loss	训练集与验证集准确率差
1	128	20	0.01	sgd	0.5	FALSE	91.66%	0.279	4.73%
2	128	20	0.01	sgd	0.5	TRUE	91.69%	0.2617	4.76%
3	128	20	0.01	adam	0.5	FALSE	10.00%	2.303	85.34%
4	128	20	1.00E-03	adam	0.5	FALSE	90.52%	0.3845	5.90%
5	128	20	8.00E-04	adam	0.5	FALSE	90.99%	0.3409	6.67%
6	128	20	1.00E-04	adam	0.5	FALSE	91.60%	0.3041	6.21%
7	128	20	8.00E-05	adam	0.5	FALSE	91.48%	0.2651	5.83%
8	128	15	1.00E-04	adam	0.5	TRUE	91.24%	0.2555	4.01%
9	128	20	1.00E-04	adam	0.5	TRUE	91.56%	0.2928	6.12%
10	64	20	1.00E-04	adam	0.5	TRUE	91.82%	0.4437	7.71%
11	256	20	1.00E-04	adam	0.5	TRUE	91.41%	0.2545	3.95%
12	192	20	1.00E-04	adam	0.5	TRUE	91.49%	0.2639	4.66%
13	128	20	1.00E-04	adam	0.4	TRUE	91.74%	0.3031	6.32%
14	128	20	1.00E-04	adam	0.6	TRUE	91.72%	0.2807	5.77%
15	128	20	1.00E-04	adam	0.7	TRUE	91.78%	0.2709	5.35%
16	128	20	1.00E-04	adam	0.8	TRUE	91.51%	0.2623	4.54%
17	128	20	0.05	sgd	0.5	TRUE	92.04%	0.4741	7.50%
18	128	20	0.008	sgd	0.5	TRUE	91.29%	0.2561	3.88%
19	128	25	0.008	sgd	0.5	TRUE	91.45%	0.2719	5.70%
20	256	20	0.01	sgd	0.5	TRUE	90.38%	0.2750	2.17%
21	256	25	0.01	sgd	0.5	TRUE	90.42%	0.2683	3.24%
22	256	30	0.01	sgd	0.5	TRUE	91.12%	0.2682	4.22%
23	256	40	0.01	sgd	0.5	TRUE	91.00%	0.3097	6.74%
24	64	20	0.008	sgd	0.5	TRUE	91.64%	0.2884	6.62%
25	128	20	0.01	sgd	0.7	TRUE	91.52%	0.2598	5.02%
26	128	20	0.01	sgd	0.6	TRUE	91.36%	0.2623	4.74%
27	128	20	0.01	sgd	0.4	TRUE	91.63%	0.2586	4.73%
模型与损失函数比较分析（batch_size = 128, epochs = 20, lr = 1e-4, adam, prob = 0.7, use_scheduler）									
实验次数\参数	模型			损失函数			准确率	loss	训练集与验证集准确率差
28	VGG			cross_entropy			91.75%	0.2576	4.87%
29	VGG			mse			92.61%	0.0124	6.47%
30	alex			mse			92.56%	0.0120	6.41%
更换数据：(batch_size = 128, epochs = 20, lr = 0.01, sgd, prob = 0.5, use_scheduler)									
实验次数\参数	模型			损失函数			准确率	loss	训练集与验证集准确率差
31	alex			mse			79.71%	0.0310	0.17%
32	VGG			mse			77.48%	0.0335	0.23%
33	VGG			cross_entropy			91.80%	0.2471	3.73%

图 1 训练结果总表

3.1 使用 AlexNet 在交叉熵损失函数的条件下进行参数调整

3.1.1 learning rate 学习率

•控制变量：batch_size = 128, epochs = 20, optimizer = adam, dropout_prob = 0.5, use_scheduler

序号\指标	learning rate	准确率	loss	训/验准确率差
3	0. 01	10. 00%	2. 303	85. 34%
4	0. 001	90. 52%	0. 3845	5. 90%
5	0. 0008	90. 99%	0. 3409	6. 67%
6	0. 0001	91. 60%	0. 3041	6. 21%
7	0. 00008	91. 48%	0. 2651	5. 83%

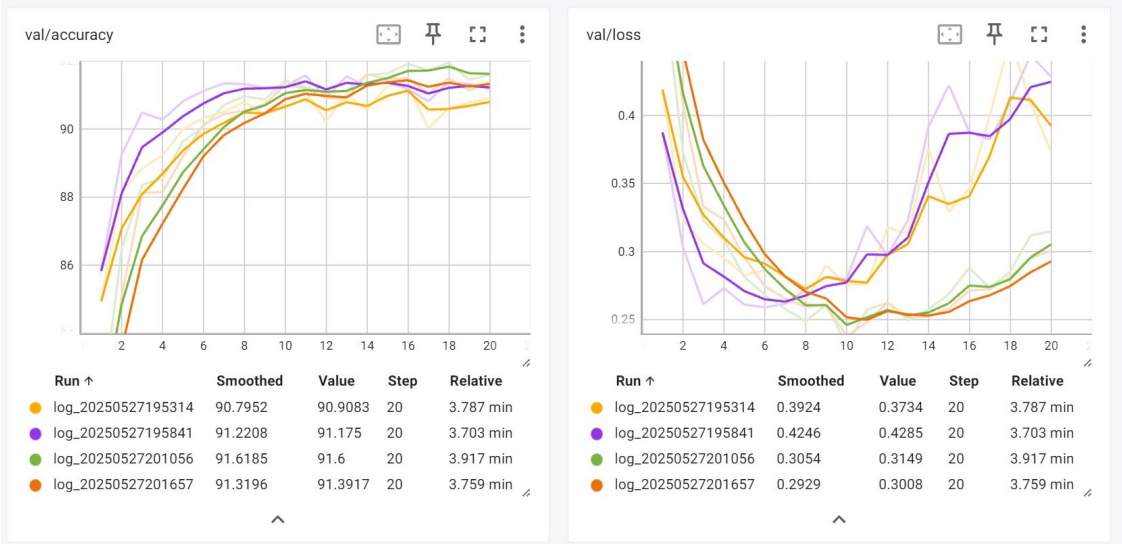


图 2 learning_rate adam

通过以上数据，我们可以首先观察到使用 adam 优化器，所需要的学习率相对较低；因此序号 3（lr = 0.01)学习率较大，训练时会跳出最优区域，结果出现了严重的欠拟合现象。

观察与比较验证集中序号 4—7 的 tensorboard，我们可以发现当 lr = 1e-4 时，训练的得分最高，相比于序号 4、5，序号 6、7 的 loss 的收敛性相对较好，在 20 epochs 内没有出现明显的上升，但整体呈上升趋势，说明可能有过拟合现象，其他参数仍需要调整。因此 adam 优化器的学习率选在 lr = 1e-4 比较合适。

• 控制变量：batch_size = 128, epochs = 20, optimizer = sgd, dropout_prob = 0.5, use_scheduler

序号\指标	learning rate	准确率	loss	训/验准确率差
2	0. 01	91. 69%	0. 2617	4. 76%
17	0. 05	92. 04%	0. 4741	7. 50%
18	0. 008	91. 29%	0. 2561	3. 88%
19 (epoch = 25)	0. 008	91. 45%	0. 2719	5. 70%

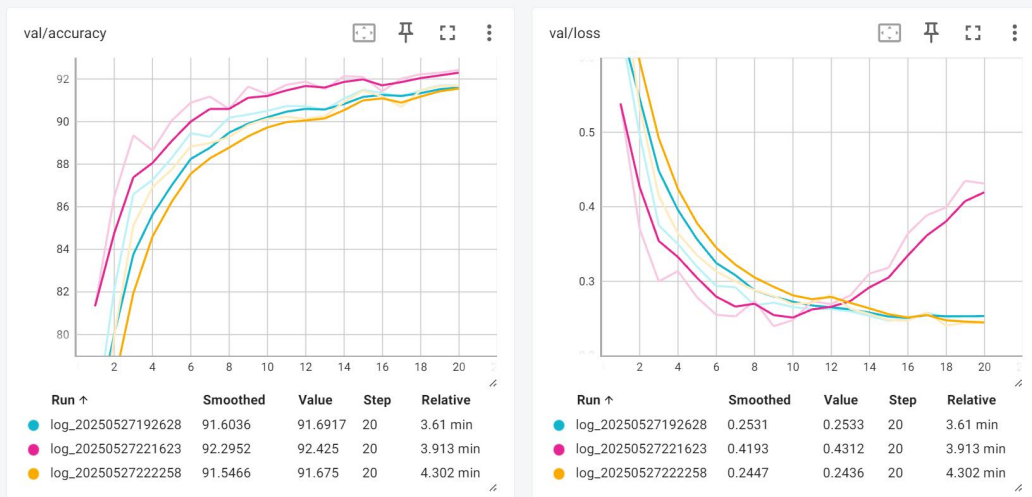


图 3 learning_rate sgd

根据以上表格，我们可以看到在 lr = 0.05 时，loss 曲线明显不收敛，这是由于学习率过高而导致的发散震荡现象严重。而将 lr 减小至 0.01 或 0.008 时，loss 都是收敛的，而 0.01 的学习率最终准确率比 lr = 0.008 高(包含 epoch = 25 的情况)，因此充分训练的前提下，lr = 0.01 是最佳选择。

3.1.2 batch size 批量大小

• 控制变量：epochs = 20, learning_rate = 1e-4, optimizer = adam, prob = 0.5, use_scheduler\

序号\指标	batch size	准确率	loss	训/验准确率差
9	128	91. 56%	0. 2928	6. 12%
10	64	91. 82%	0. 4437	7. 71%
11	256	91. 41%	0. 2545	3. 95%
12	192	91. 49%	0. 2639	4. 66%

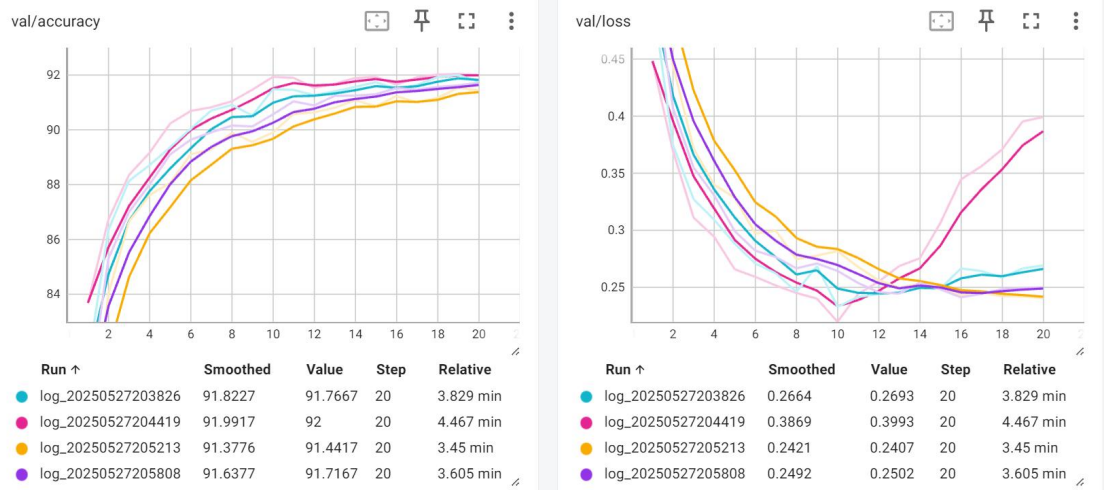


图 4 batch size adam

batch_size 是深度学习训练过程中的一个关键超参数，它定义了每次迭代（iteration）中用于更新模型参数的样本数量。根据图标分析可知，使用 adam 优化器，较小的 batch_size（如 64）容易出现过拟合现象（训练集与验证集准确率相差太大，loss 在多次迭代后变为增大）；中等的 batch_size（如 128）拥有相对最高的准确率，但泛化能力有待进一步优化；较大的 batch_size（如 192、256）拥有着更好的泛化能力。

• 控制变量：epochs = 20/30, learning_rate = 0.01, optimizer = sgd, prob = 0.5, use_scheduler\

序号\指标	batch size	准确率	loss	训/验准确率差
2	128	91. 69%	0. 2617	4. 76%
22(epoch = 30)	256	91. 12%	0. 2682	4. 22%
24(lr = 0.008)	64	91. 64%	0. 2884	6. 62%

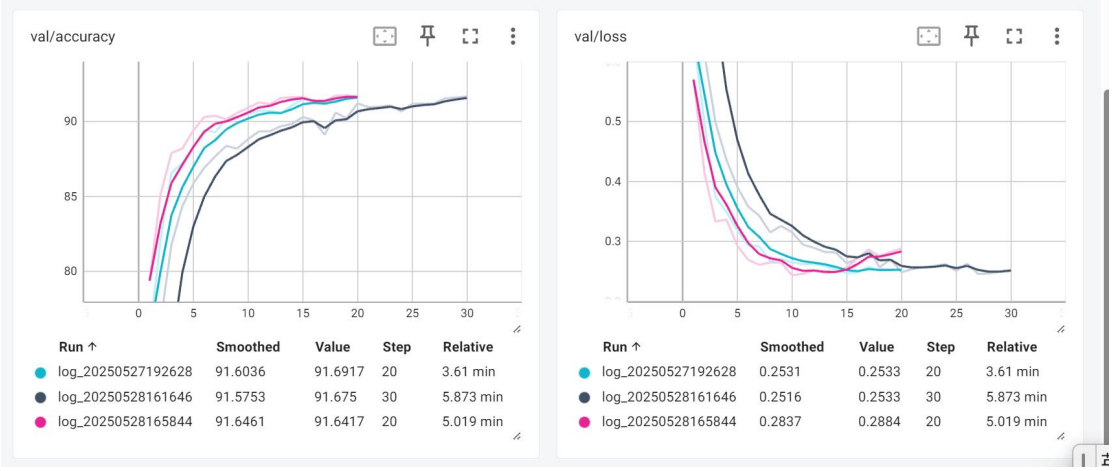


图 5 batch size sgd

根据图表信息分析可知，在 sgd 优化器训练的条件下，batch size 的表现与 adam 大体相似：中高的 batch size（如 128，256）的准确率与泛化能力俱佳，但高 batch size 需要更高的迭代次数；而较低的 batch size（如 64），虽然准确率较高，但即使在更低的学习率下（lr = 0.008）,泛化能力仍较差，过拟合现象较为严重。

3.1.3 dropout_prob 舍弃概率

•控制变量：batch size=128, epochs=20, learning_rate = 0.0001, optimizer = adam, use_scheduler

序号\指标	dropout_prob	准确率	loss	训/验准确率差
9	0. 5	91. 56%	0. 2928	6. 12%
13	0. 4	91. 74%	0. 3031	6. 32%
14	0. 6	91. 72%	0. 2807	5. 77%
15	0. 7	91. 78%	0. 2709	5. 35%
16	0. 8	91. 51%	0. 2623	4. 54%

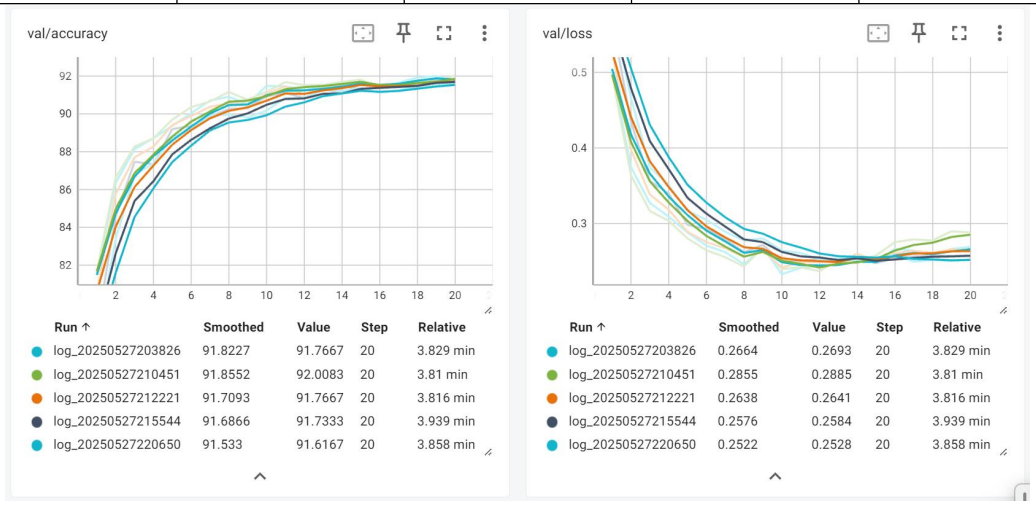


图 6 drop prob adam

dropout_prob 这个参数定义了在训练过程中，随机“舍弃”（暂时从网络中移除）某个神经元（节点）的概率。根据图表数据，我们可以发现除 prob = 0.4 外，其他概率的 loss 都基本收敛。整体而言随着舍弃概率增大准确率变化不大，在 prob = 0.7 时达到最佳，泛化能力逐渐提升。综合来看在使用 adam 优化器时，使用 prob = 0.7 为最佳的选择。

• 控制变量：batch size=128, epochs=20, learning_rate = 0.01, optimizer = sgd, use_scheduler

序号\指标	dropout_prob	准确率	loss	训/验准确率差
2	0. 5	91. 69%	0. 2617	4. 76%
25	0. 7	91. 52%	0. 2598	5. 02%
26	0. 6	91. 36%	0. 2623	4. 74%
27	0. 4	91. 63%	0. 2586	4. 73%

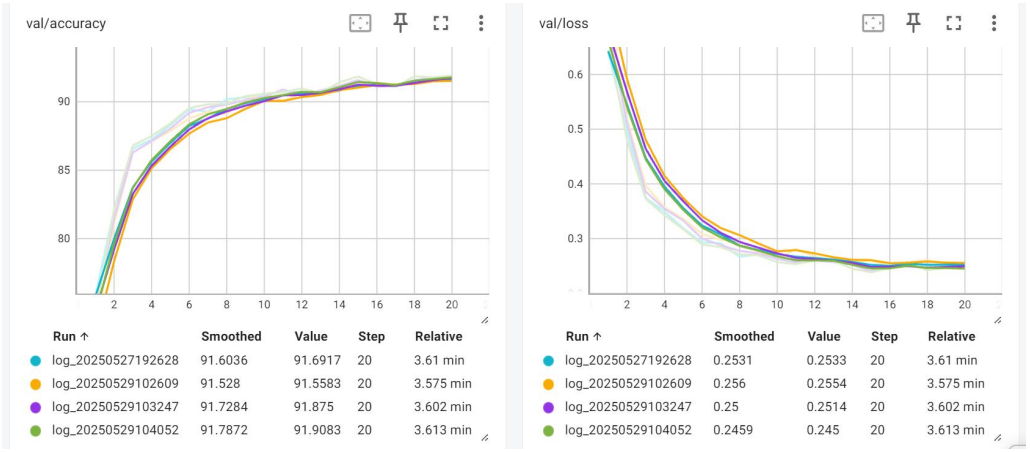


图 7 drop prob adam

根据图表数据，在 sgd 优化器下，我们可以分析得到：不同的 prob 对模型的训练结果影响不大，模型均可以有较好的表现，loss 也均收敛。对测试集而言，prob = 0.5 时，准确率最高，因此选定 prob = 0.5 作为最佳结果。

3.1.4 epoch 训练轮次

• 控制变量：batch size=256, prob = 0.5, learning_rate = 0.01, optimizer = sgd, use_scheduler

序号\指标	epoch	准确率	loss	训/验准确率差
20	20	90. 38%	0. 275	2. 17%
21	25	90. 42%	0. 2683	3. 24%
22	30	91. 12%	0. 2682	4. 22%
23	40	91. 00%	0. 3097	6. 74%

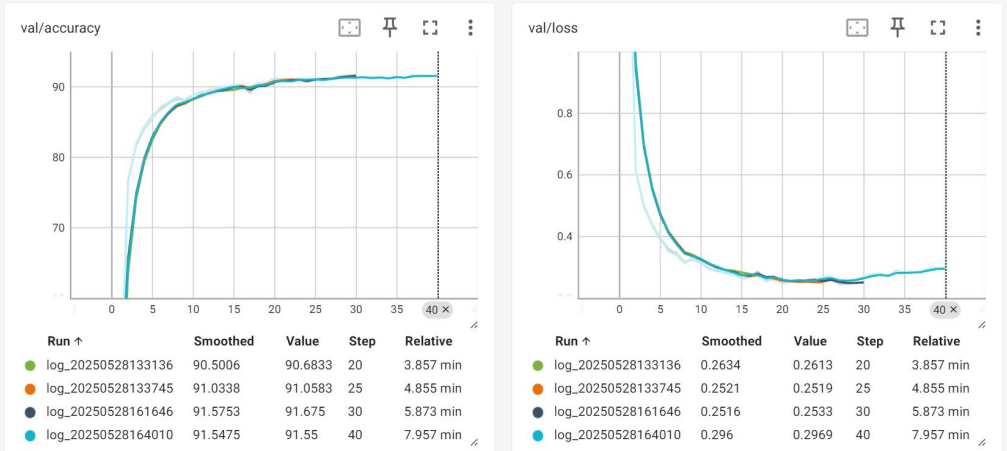


图 8 epoch

epoch 代表训练的轮次，它的设定对模型性能、训练时间和过拟合风险有显著影响。当 epoch 不足时，模型未充分训练；当 epoch 过多时，可能导致模型陷入过拟合，导致 loss 有震荡或回升。因此，在调节其他参数时实时跟进调节 epoch 有助于我们对不同参数的最佳训练状态进行合理比较。

由上图表可知，在探索 sgd 优化器下，合适的 batch_size 时，我们使用 epoch 的调节寻找相同 batch_size 条件下的最优解。在低 epoch（20、25）状态下，模型未进行充分训练；而在高 epoch（40）状态下，模型的 loss 出现回升，有明显的过拟合现象；中等训练轮次（30）则拥有最佳的准确率以及较好的泛化能力。因此针对 batch size = 256，我们可以知道最佳的训练轮次是 epoch = 30。

除上表示例之外，实际调参过程中还多次进行微调 epoch 来探索当前参数下的最佳训练结果（如序号 8、19）。

3.1.5 学习率调节器的引入

• 控制变量：batch size=128, epoch = 20, prob = 0.5, learning_rate = 0.01, optimizer = sgd

序号\指标	use_scheduler	准确率	loss	训/验准确率差
1	False	91.66%	0.279	4.73%
2	True	91.69%	0.2617	4.76%

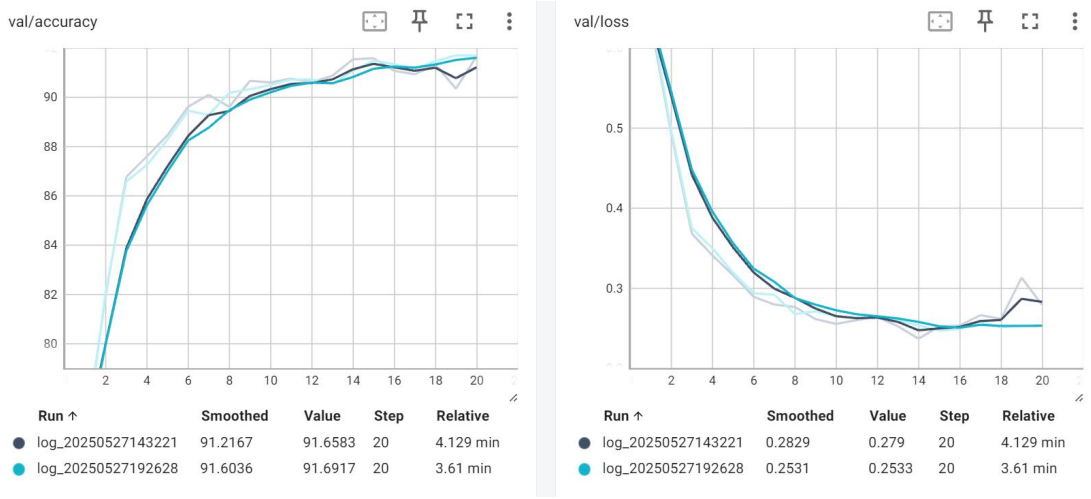


图 9 use_scheduler sgd

学习率调节器是一种在训练过程中动态调整学习率的策略，本模型主要采用的是多项式衰减学习率调节器。由上表可知，使用 sgd 优化器时，学习率调节器对减少震荡、促进 loss 收敛有着显著作用，并能在一定程度上提高模型的准确率。

• 控制变量：batch size=128, epoch = 20, prob = 0.5, learning_rate = 0.0001, optimizer = adam

序号\指标	use_scheduler	准确率	loss	训/验准确率差
6	False	91.60%	0.3041	6.21%
9	True	91.56%	0.2928	6.12%

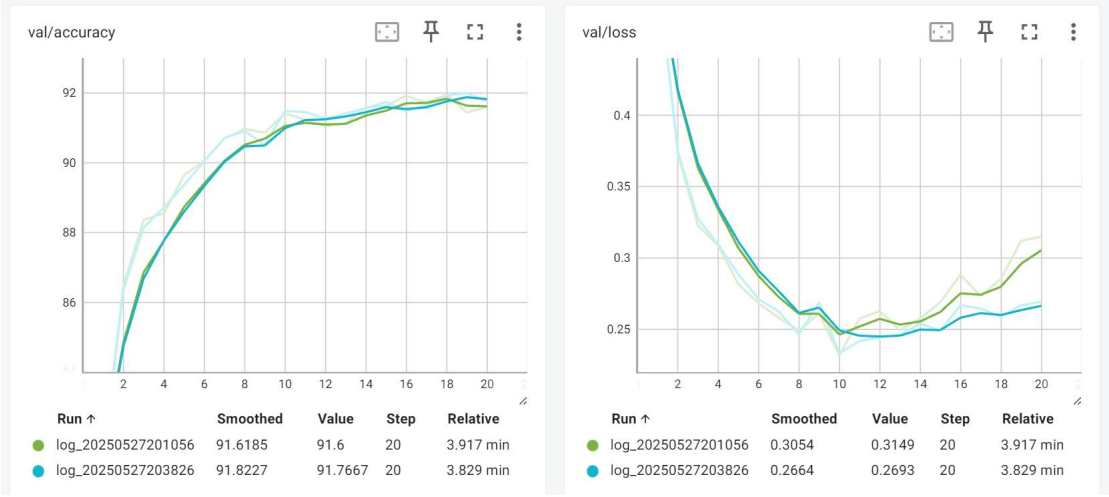


图 10 use scheduler adam

在使用 adam 优化器的条件下，虽然表中测试集的数据我们不能看出明显的性能提升，但是在验证集的曲线走势我们可以看到，无论是准确率还是泛化能力（loss 的收敛性），引入学习率调节器后，模型的性能都有明显的提升。因此无论对于哪一种优化器，学习率调节器均有着提升模型准确率与泛化能力的性能。

3.1.6 优化器的选择

在上述的参数比较过程中，我对两种不同的优化器均进行了分别讨论，可以发现两种优化器的最佳参数不同，一些性质也不相同，但是总体的训练结果最佳值相差不大，下面我将简要对比一下两种优化在各自最佳参数下的训练结果对比（分别为图一总表中标记的两行）：

序号\指标	use_scheduler	准确率	loss	训/验准确率差
2	sgd	91.69%	0.2617	4.76%
15	adam	91.78%	0.2709	5.35%

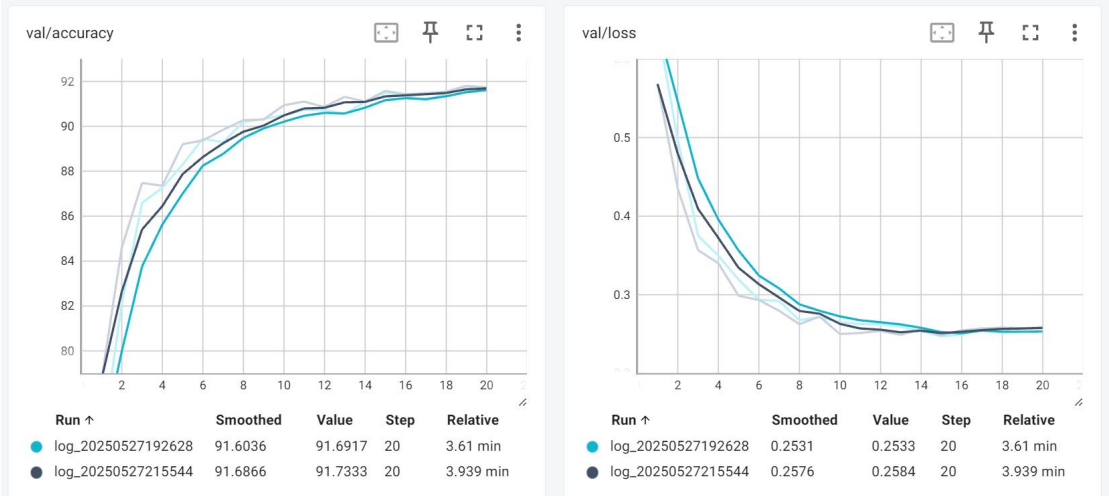


图 11 优化器

从图表中我们可以看到，两个模型的训练最佳情况均具有较好的性能，adam 最终优化的准确率略高于 sgd，但 sgd 在泛化能力（loss 收敛性）上略胜一筹，总体而言二者无明显的差距，都是较好的模型训练结果。

3.2 AlexNet 与 VGGNet 在不同损失函数下的模型比较

• 使用 adam 优化器的最佳训练结果参数

batch size=128, epochs=20, learning_rate = 0.0001, prob = 0.7, optimizer = adam, use_scheduler

序号\指标	模型	损失函数	准确率	loss	准确率差
15	Alex	cross_entropy	91. 78%	0. 2709	5. 35%
28	VGG	cross_entropy	91. 75%	0. 2576	4. 87%
29	VGG	mse	92. 61%	0. 0124	6. 47%
30	alex	mse	92. 56%	0. 012	6. 41%

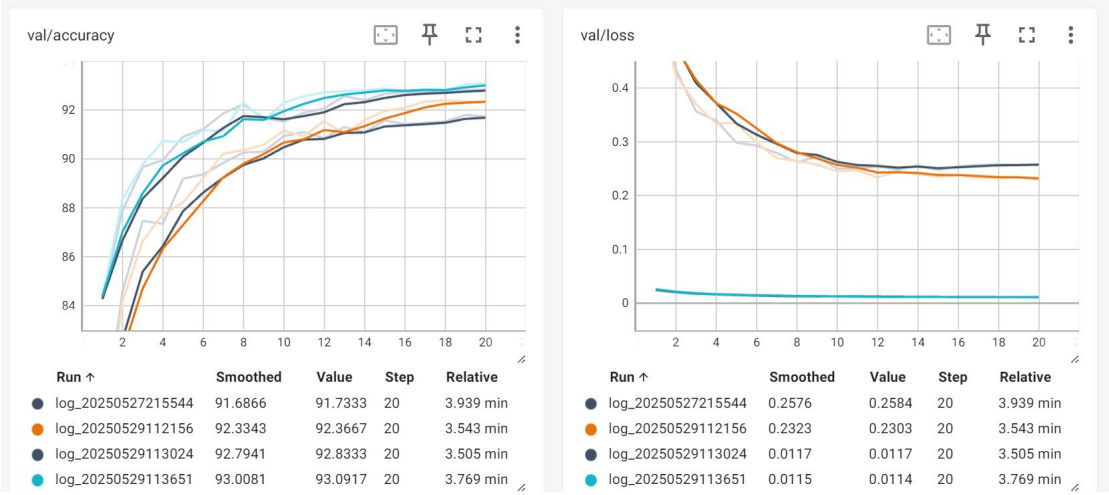


图 12 模型对比 adam 最优参数

根据图表结果，我们看到 Alex 与 VGG 模型的不同对实验结果的影响很小；而对比两个损失函数，我们发现均方误差损失的训练效果要好于交叉熵损失，这与我们在实验原理部分分析的两个损失函数的性能差距很大：交叉熵损失相比于均方误差更擅长处理分类问题。对此，我通过查找资料分析可能有以下两点原因：

1 实验中 shirt 标签的训练结果持续低于 80%，与其他标签的训练结果差距较大，数据集中可能存在一定的噪声导致 shirt 难以分辨。

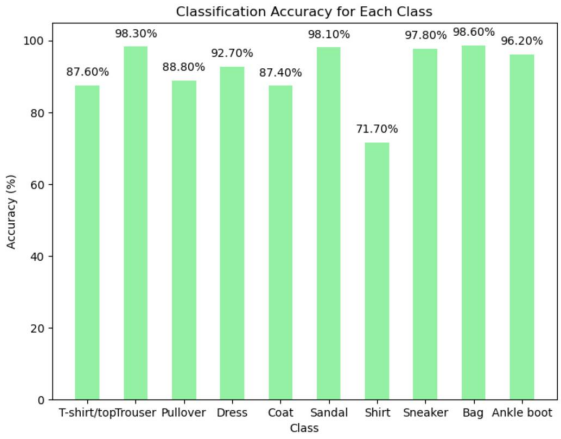


图 13 各类别标签测试准确率

2 交叉熵损失函数的优势并不绝对，在特定合适的参数下，mse 的表现可能更优。

针对以上两点分析，我们使用 sgd 的最优参数继续进行测试与模型对比分析：

• **使用 sgd 优化器的最佳训练结果参数**

batch size=128, epochs=20, learning_rate = 0.01, prob = 0.5, optimizer = sgd, use_scheduler

序号\指标	模型	损失函数	准确率	loss	准确率差
2	Alex	cross_entropy	91.69%	0.2617	4.76%
31	alex	mse	79.71%	0.031	0.17%
32	VGG	mse	77.48%	0.0335	0.23%
33	VGG	cross_entropy	91.80%	0.2471	3.73%

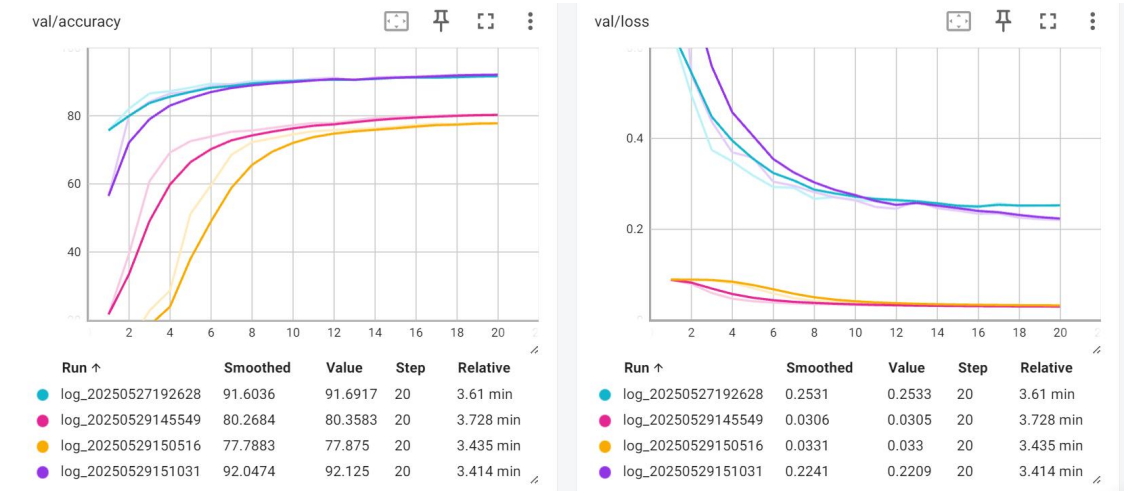


图 14 模型对比 sgd 最优参数

根据图表分析我们可以得到：Alex 与 VGG 模型上的差距依然很小；在损失函数上，mse 的训练结果要远远差于交叉熵损失，训练的收敛速度也极为缓慢，这一结果验证了我们在实验原理部分的分析，也验证了我们对上述反常结果的原因推测 2：mse 不具有持续稳定的良好训练表现，相比之下交叉熵损失更适合对此分类问题进行模型的训练。

4 总结与收获

通过本次实验，我全面了解了卷积神经网络的层次结构与工作原理，掌握了模型调参的重要要素以及参数对模型的重大影响。这对我在课堂上理论知识的学习有着很好的促进作用。与此同时，在模型对比时反常数据的出现，让我了解到很多理论知识的原理并不一定是绝对的，在实验结果与理论值违背时，我们应积极地探索其背后的原因并主动去验证，只有这样的不断探索的过程，我们才能对知识有更全面的了解和掌握。本次实验让我受益匪浅，希望我能够继续在人工智能领域去探索、去成长、去进步。