期中作业：微调与目标检测

何益涵　20307110032
吕文韬

摘要： 本项目的仓库地址为：
https://github.com/HeDesertFox/Neural-Networks-and-Deep-Learning-Homework-Group-Tas
git。
详见其中的文件夹 group_task2。其中包含微调任务 task1_finetuning
和目标检测任务 task2_object_detection。
训练好的模型权重详见：

第一章　微调在 ImageNet 上预训练的卷积神经网络实现鸟类识别

第 1 节　任务描述

1. 修改现有的 CNN 架构（如 AlexNet，ResNet-18）用于鸟类识别，通过将其输出层大小设置为 200 以适应数据集中的类别数量，其余层使用在 ImageNet 上预训练得到的网络参数进行初始化；
2. 在 [CUB-200-2011] 数据集上从零开始训练新的输出层，并对其余参数使用较小的学习率进行微调；
3. 观察不同的超参数，如训练步数、学习率，及其不同组合带来的影响，并尽可能提升模型性能；
4. 与仅使用 CUB-200-2011 数据集从随机初始化的网络参数开始训练得到的结果进行对比，观察预训练带来的提升。

第 2 节　项目架构

此任务的所有文件在 `task1_finetuning` 文件夹中，其中包含以下文件：

1. `data_loading_preprocessing.py` 文件：包含数据下载函数与预处理函数。
2. `model.py` 文件：包含网络构造函数。
3. `training_fine_tuning.py` 文件：包含训练和超参数调优函数。
4. `main_notebook.ipynb` 文件：包含微调任务的全部流程，如数据加载、调参和训练可视化。

第 3 节　实验设置

3.1. 数据增广

本实验根据 CUB-200-2011 中的 `train_test_split.txt` 文件对数据集进行划分，形成训练集和验证集。在数据预处理中，对训练集的数据进行了图像增强，其中包括了

随机水平翻转和随机旋转。

3.2. 模型选择

此项目的模型构建函数可以生成两种 CNN 架构：Alexnet 和 ResNet-18，并且可以选择采用 Imagenet 预训练的参数初始化或随机初始化。生成的模型的最后一层被替换为输出维数为 200 的全连接层以适应本任务的要求。

为了实现更高的模型性能，本实验采用 ResNet-18 架构，使用者可以在模型构造函数中输入参数`"alexnet"` 将模型改为 Alexnet。

3.3. 优化器设置

本实验的优化器均选择 SGD 优化器，在所有实验中都采用 0.9 的动量以加速收敛，采用 1e-3 的权重衰退缓解过拟合。

在后续的调参和训练过程中，预训练模型的学习率在最后一层都正常设置，其余层的学习率都除 10。

第 4 节　调参结果

本实验调节两个参数以尽可能提高模型性能：学习率 (`lr`) 和训练轮数 (`epoch`)。

总体来说，增加训练轮数不会减少验证集上的准确率。这是因为 ResNet-18 的表现力足够强，一般可以在训练集上将精度提升到接近 100%，此时梯度几乎为零，因此最终验证集精度不会因为训练过久而下降。

经过若干次调参，预训练模型的最后一次调参的参数列表定为 `lr`: {0.5e-3, 1e-3, 2e-3}, `epoch`: 20。最优参数为 `lr` = 2e-3，`epoch` = 20。

实验表明，在这三种学习率之下，精度都比较接近，略高于 70%，其中 `lr` = 2e-3 时的表现略好。

```
Tuning hyperparameters for pretrained model...
Training with lr=0.005, num_epochs=20
Final validation accuracy: 0.7164
New best accuracy: 0.7164 with lr=0.005 and num_epochs=20
Training with lr=0.01, num_epochs=20
Final validation accuracy: 0.7245
New best accuracy: 0.7245 with lr=0.01 and num_epochs=20
Training with lr=0.02, num_epochs=20
Final validation accuracy: 0.7370
New best accuracy: 0.7370 with lr=0.02 and num_epochs=20
Best Params: lr=0.02, num_epochs=20, Accuracy=0.7370
Best hyperparameters for pretrained model: {'lr': 0.02, 'num_epochs': 20,
```

图 1.1: 最后一次调参结果

随机初始化模型也可以做调参。本实验中随机初始化模型采用和预训练模型一样的参数，保证比较的公平性。

第 5 节　实验结果

5.1.　数据分析

以下图片中，橙线都是预训练模型的数据曲线，蓝线都是随机初始化模型的数据曲线。完整的数据文件在文件夹 `run` 中。

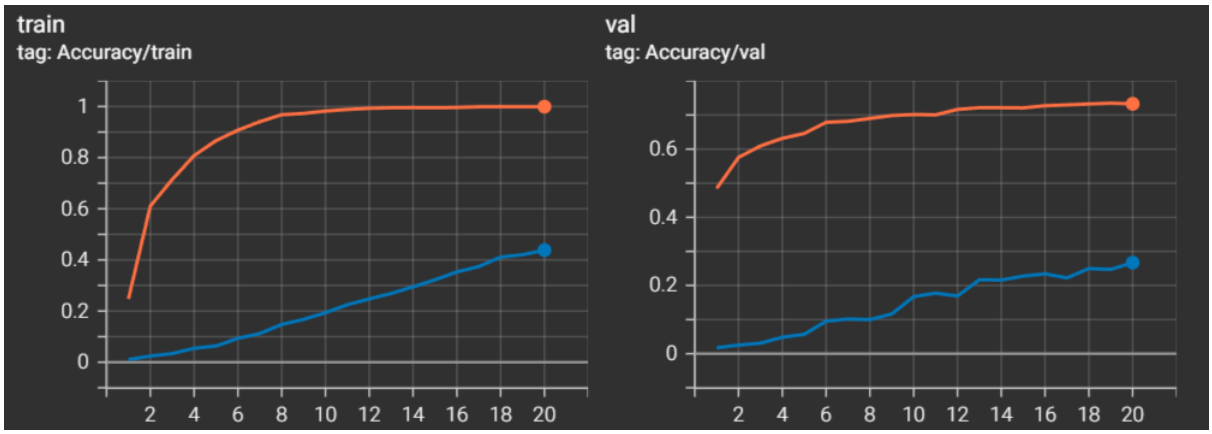

图 1.2: 准确率对比

预训练模型的训练准确率迅速上升，并且在初期就已经接近 1，这表明模型能够很好地学习训练数据。准确率接近 1 通常表明模型对训练数据的拟合非常好。随机初始化模型的训练准确率相比之下增长较慢，最终也没有接近 1，这可能意味着模型学习较慢，或者是模型容量不足以完全学习数据。

预训练模型的验证准确率也很快上升并保持在较高水平，这说明预训练模型在未见数据上也具有很好的泛化能力。验证准确率的高稳定性同时也表明模型没有出现显著的过拟合。随机初始化模型的验证准确率虽有所提高，但整体上显著低于预训练模型，这表明其泛化能力较弱。验证准确率的较低水平也表明该模型可能受限于其初始参数设置，未能充分利用训练数据。

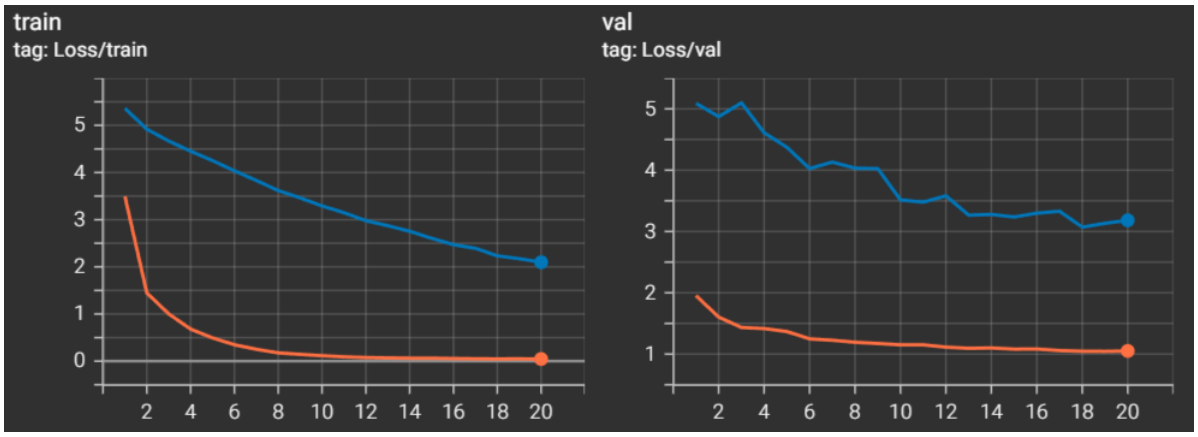

图 1.3: 损失函数对比

预训练模型的训练损失迅速下降并趋于平稳，这与高训练准确率一致，表明模型有效地减少了误差。随机初始化模型的训练损失下降较慢，结束时仍高于预训练模型的损失，这进一步证实了其学习效率较低。

预训练模型的验证损失下降并保持较低，与高的验证准确率相符，说明模型在未见数据上的表现良好。验证损失的低水平和稳定性表明没有过拟合。随机初始化模型的验证损失相比之下较高并稍有波动，可能指示模型在未见数据上的性能不够稳定，这可能是由于模型参数初始化不佳或者模型结构不够优化。

5.2. 结论

预训练模型明显优于随机初始化模型，无论是在训练过程还是在验证过程中。这表明利用预训练的权重可以显著提升模型的学习效率和泛化能力。

随机初始化模型可能需要更多的训练周期、更复杂的网络结构或者更多的调优来提高其性能。

基于以上分析，使用预训练模型在类似的任务中是一个有效的策略，特别是当可用的训练数据量不足以从头训练一个复杂模型时。对于随机初始化的模型，可能需要探索额外的技术，如更深的网络、正则化策略或更精细的超参数调整，以提高其性能。

第二章　在 VOC 数据集上训练并测试目标检测模型 Faster R-CNN 和 YOLO V3

第 1 节　任务描述

1. 学习使用现成的目标检测框架——如 mmdetection 或 detectron2——在 VOC 数据集上训练并测试目标检测模型 Faster R-CNN 和 YOLO V3；

2. 挑选 4 张测试集中的图像，通过可视化对比训练好的 Faster R-CNN 第一阶段产生的 proposal box 和最终的预测结果。

3. 搜集三张不在 VOC 数据集内包含有 VOC 中类别物体的图像，分别可视化并比较两个在 VOC 数据集上训练好的模型在这三张图片上的检测结果（展示 bounding box、类别标签和得分）