人工智能导论第二次大作业
“猫狗”识别

于海粟
无 56
2015010901

January 28, 2018

1 算法设计

1.1 概述

本次作业要求通过给出的 20，000 个猫狗图片分析出包含猫狗图像的特征区别并设计出一套算法辨别任意图片的猫狗特征。考虑到数据集的数据量比较大，基本符合深度学习 [1] 的要求，而且目前基于卷积神经网络的图像分类技术已经十分成熟，调研若干文献之后选择 Resnet 网络作为本次作业的分类器。本次作业基于 pytorch 环境完成，pytorch 的 torchvision 库给出了 Resnet 的模型结构并提供 pretrained 的模型参数。再结合数据集进行训练和检验，在一定 epochs 内选择正确率最高的模型输出。

1.2 Resnet

Resnet 的根本动机在于“退化”问题，即当模型的层次加深时错误率反而提高。虽然模型加深意味着学习能力的增强，但 SGD 优化也会随着模型复杂度的提升而变得更加困难，导致了模型达不到好的学习效果，所以 He et al. [2] 提出了 Residual 结构，即增加一个 identity mapping（恒等映射），将原始所需要学的函数 H(x) 转换成 F(x)+x，而作者认为这两种表达的效果相同，但是优化的难度却并不相同，作者假设 F(x) 的优化会比 H(x) 简单的多。这一想法也是源于图像处理中的残差向量编码，通过一个 reformulation，将一个问题分解成多个尺度直接的残差问题，能够很好的起到优化训练的效果。这个 Residual block 通过 shortcut connection 实现，通过 shortcut 将这个 block 的输入和输出进行一个 element-wise 的加叠，这个简单的加法并不会给网络增加额外的参数和计算量，同时却可以大大增加模型的训练速度、提高训练效果，并且当模型的层数加深时，这个简单的结构能够很好的解决退化问题。

1.3 Resnet 与特征提取

卷积神经网络（CNN）是深度学习中十分常见的图像特征提取方法 [3]。卷积神经网络的第一个卷积层用来检测如边、角等低阶特征，第二个卷积层组合低阶特征并检测三角形、四边形等稍微复杂的特征，以此类推检测越来越复杂的形状构成的图片。本实验中的猫狗识别也是采用这种方式，通过 Resnet 解决卷积深度对 SGD 优化的影响并通过增加深度来提取尽量复杂的图像特征，最终通过全连接层整合图像特征输出最终的分类结果。

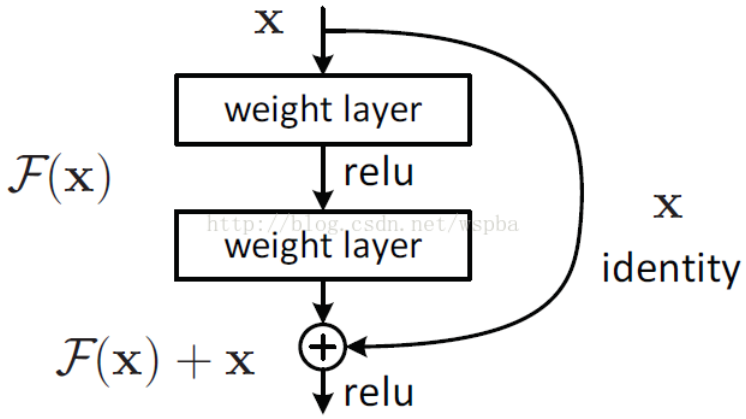

Figure 1: Residual Structure

1.4 算法实现

算法基于 pytorch 实现，主要包括如下部分：

1. 数据准备：采用 pytorch 提供的 ImageFolder 类，按照要求将数据集 80% 用作训练集，20% 用作测试集，按照 *data/train(val)/cat(dog)/cat(dog).xxxx.jpg* 结构分配数据文件；
2. 数据预处理：采用 pytorch 提供的配合 PIL 使用的库函数进行数据预处理，包括尺寸调整、随机翻转、转为 Tensor 并正则化；
3. 训练与检验：通过若干 epoch 进行训练和检验，每完成一个 epoch 的训练就紧随一个 epoch 的检验，统计平均损失和正确率，但检验结果在训练阶段只用于选择最佳模型而不用于反向传播优化模型。

2 评估

2.1 训练过程

训练过程中对于每个 epoch 都有 Training Phase 和 Validation Phase，都会针对各自的数据集进行正确率和损失函数的检验并将检验结果写入日志文件中；评估过程读取日志文件并寻找记录的损失函数值（Loss）和匹配准确度（Accuracy）并绘制成折线图如 Figure 2 所示；

图中可以看到在 200，250 个 epoch 处

2.2 交叉验证

采用 10 倍交叉验证方法 (10-fold cross-validation)。首先将猫狗数据集分别随机分成十份再组合，使用九份数据进行训练并使用剩余的一份进行测试，记录正确率结果如 Table 1。

存在跳变，这是因为受实验条件限制我在实验过程中将 epoch 上限设置为 200 并通过加载最新参数迁移学习，导致正确率存在跳变。

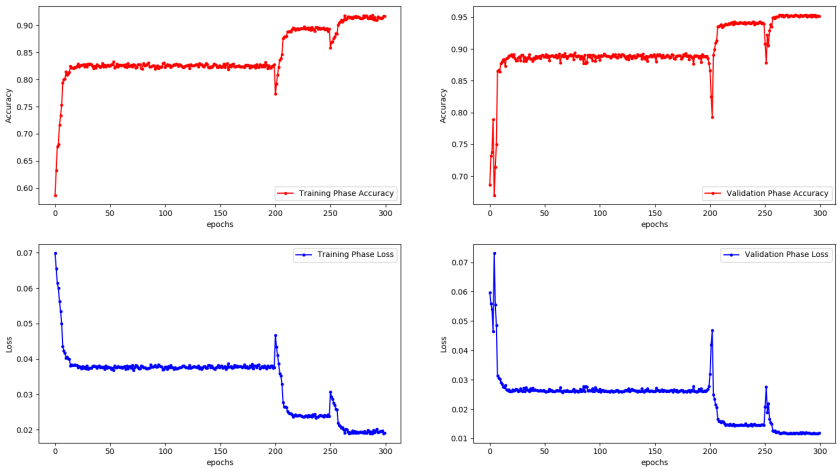

Figure 2: Accuracy and Loss in training and validation phase

3 算法优化

最便捷的一种算法优化方法就是使用 pytorch 提供的 pretrained 模型，可以将正确率直接提高到 99% 左右而没有 pretrained 的模型由于数据集的局限性和一些其他原因导致正确率无法达到这么高。其他优化算法例如使用更加复杂的网络模型（Resnet34 等，即使用更复杂的特征提取算法）或者采用更丰富的数据预处理措施和一些 offline、online 的数据增强处理等方法。

4 总结

本次实验主要通过对猫狗识别这一具体问题的解决，基本熟练掌握了深度学习和卷积神经网络的应用，对 pytorch 深度学习框架运用更加熟练，了解了 Resnet 的基本原理及其在图像分类领域的广泛应用，代码能力也得到了一定的增强。本次实验的不足之处在于受计算能力的限制，在短时间内无法快速地验证算法的正确性和鲁棒性，导致很多算法优化的思路都没有时间尝试，交叉验证过程为了节省时间也使用了训练好的 best_model。最终的识别正确率可以达到 95% 以上，还算完成任务。

References

[1] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. *Deep Learning*. Nature, 7553(521): 436-444, 2015.

[2] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C]// Computer Vision and Pattern Recognition. IEEE, 2016:770-778.

[3] Matthew D. Zeiler, Rob Fergus. Visualizing and Understanding Convolutional Networks[J]. 2013, 8689:818-833.

测试集编号	训练集正确率	测试集正确率
0	0.9141	0.9100
1	0.9111	0.9145
2	0.9116	0.9230
3	0.9135	0.9105
4	0.9124	0.9195
5	0.9122	0.9120
6	0.9127	0.9165
7	0.9151	0.9025
8	0.9132	0.9030
9	0.9160	0.9080
Average	0.9131	0.9120

Table 1: 十折交叉验证实验结果

类别	训练集正确率	测试集正确率
pre-trained = False	0.9535	0.9122
pre-trained = True		
0.9518	0.9880	

Table 2: Resnet18 训练最终结果

A 文件清单

- logs: 日志文件夹
  - cross_val.log:10 折交叉验证实验日志文件
  - resnet.log: 使用 resnet 训练和验证的日志文件
  - resnet_pretrained.log: 使用 pytorch 提供的预训练模型参数的 resnet18 网络训练和验证的日志文件
- src: 代码源文件夹
  - cross_val_plot.py: 交叉验证结果绘制文件夹
  - cross_val.py: 交叉验证实验
  - extract_feature.py: 特征提取实验
  - logger.py: 日志文件配置初始化
  - resplot.py:resnet18 训练验证结果绘制
  - train2.py:resnet18 训练验证实验
  - visualize.py:resnet18 网络结构可视化
- trained_models: 训练好的模型文件夹
  - best_model.pth:resnet18 训练验证过程的最佳模型
  - resnet18-5c106cde.pth:pytorch 提供的预训练 resnet18 模型

B Resnet18 结构

Resnet18 结构全貌见 Figure 3。

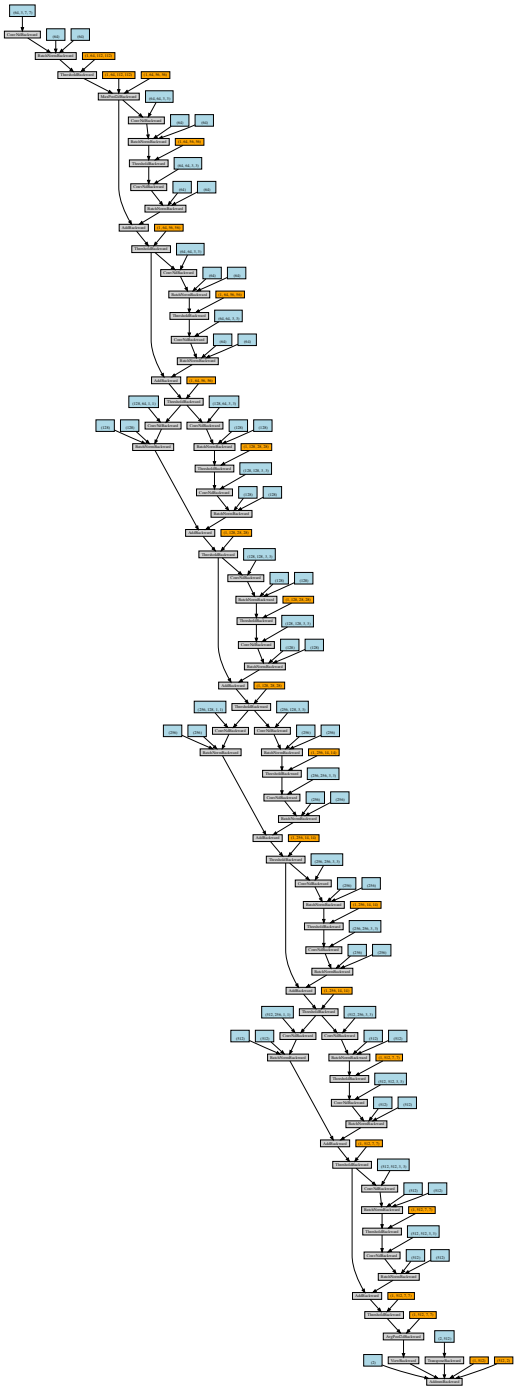

Figure 3: Resnet18 Architecture