手写数字识别实验报告

本实验使用 PyTorch 实现了使用卷积神经网络对猫狗分类数据集进行分类的任务。

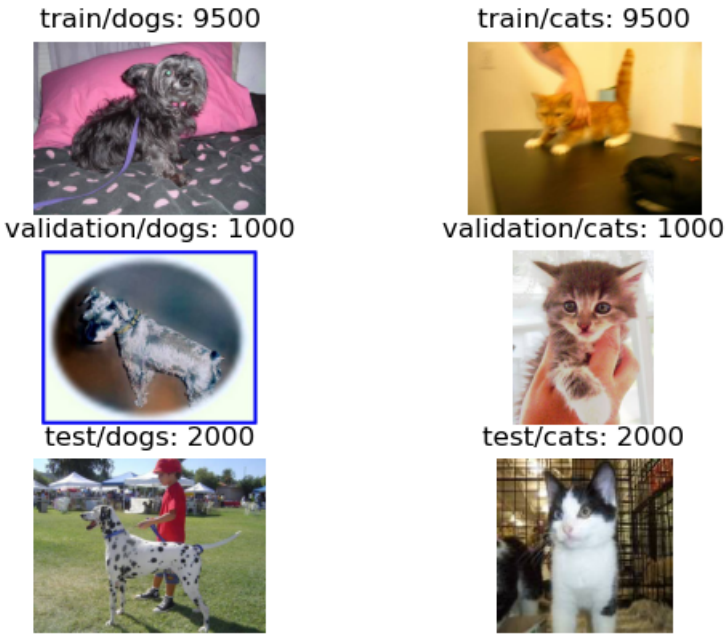
数据集

Dogs vs. Cats 数据集是一个由 Microsoft Research 创建的用于图像分类任务的数据集，图片来源于 Flickr 网站，其中包含 12500 张狗的图像和 12500 张猫的图像，大小从 32x32 像素到 500x500 像素不等。

本实验将数据集划分为三部分，其中训练集猫狗照片各 9500 张，验证集各 1000 张，测试集各 2000 张。目录结构如下：

```
├── data
|   ├── train
|   |   ├── cats
|   |   ├── dogs
|   ├── validation
|   |   ├── cats
|   |   ├── dogs
|   ├── test
|   |   ├── cats
|   |   ├── dogs
```

预览如下：



预处理：

- Resize 到 224×224 像素
- 标准化

实验环境

软件环境：Python 3.8.13，PyTorch 1.13.1。

硬件环境：Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz，Tesla P100-PCIE-16GB

实验设置

卷积神经网络结构：

Layer (type)	Output Shape	Parameters	Kernels	Kernel Size	Stride
Conv2d-1	[-1, 16, 111, 111]	448	16	3	2
BatchNorm2d-2	[-1, 16, 111, 111]	32			
ReLU-3	[-1, 16, 111, 111]	0			
MaxPool2d-4	[-1, 16, 55, 55]	0		2	0
Dropout-5	[-1, 16, 55, 55]	0			
Conv2d-6	[-1, 32, 27, 27]	4,640	32	3	2
BatchNorm2d-7	[-1, 32, 27, 27]	64			
ReLU-8	[-1, 32, 27, 27]	0			
MaxPool2d-9	[-1, 32, 13, 13]	0		2	0
Dropout-10	[-1, 32, 13, 13]	0			
Conv2d-11	[-1, 64, 6, 6]	18,496	64	3	2
BatchNorm2d-12	[-1, 64, 6, 6]	128			
ReLU-13	[-1, 64, 6, 6]	0			
MaxPool2d-14	[-1, 64, 3, 3]	0		2	0
Dropout-15	[-1, 64, 3, 3]	0			
Linear-16	[-1, 32]	18,464			
ReLU-17	[-1, 32]	0			
Dropout-18	[-1, 32]	0			
Linear-19	[-1, 2]	66			

超参数设置：

- 初始学习率：10^{-3}。
- 优化算法：Adam
- 批量大小：128
- 迭代次数：50

- 激活函数：ReLU
- 损失函数：交叉熵损失函数
- Dropout rate：0、0.1、0.2

代码说明

文件 `split_dataset.py` 用于划分数据集：

- `get_image_name_list`：获取原始数据集的图片路径
- `split_dog_cat`：将原始数据集的图片路径中狗和猫的图片分开
- `copy_file`：批量复制图片
- `test`：检查划分结果，并生成预览
- `main`：函数入口

文件 `main.py` 用于训练和测试模型。其中核心的函数和类：

- `MyDataset`：继承 PyTorch 提供的 Dataset 类，对图片进行读取和预处理
- `CNN`：实现了一个卷积神经网络
- `train_epoch`：使用训练集对模型参数进行一轮完整更新，并计算准确率和样本平均损失
- `train`：通过调用 `train_epoch` 函数，对模型参数进行 `EPOCH` 轮更新，并保存测试集表现最好的模型参数，绘制训练中的损失曲线
- `evaluate`：对数据集进行预测，计算准确率和平均损失
- `get_loader`：加载 MNIST 数据集
- `main`：函数入口

此外还实现了一些辅助功能：

- `args`：命令行参数
- `log`：日志记录
- `show_error_images`：展示预测错误的图片

models 文件夹中包含了三个训练完成的模型权重。

运行代码

安装依赖：

```
pip install torch torchvision torchsummary matplotlib tqdm shutil
```

划分数据集：

```
python split_dataset.py
```

训练模型：（参数均为可选）

```
python main.py --batch_size 128 --epoch 50 --learning_rate 0.001 --dropout 0.1
```

测试模型：（`model_path` 为模型参数文件路径）

```
python main.py --test true --model_path models/cnn_drop_0.pth
```

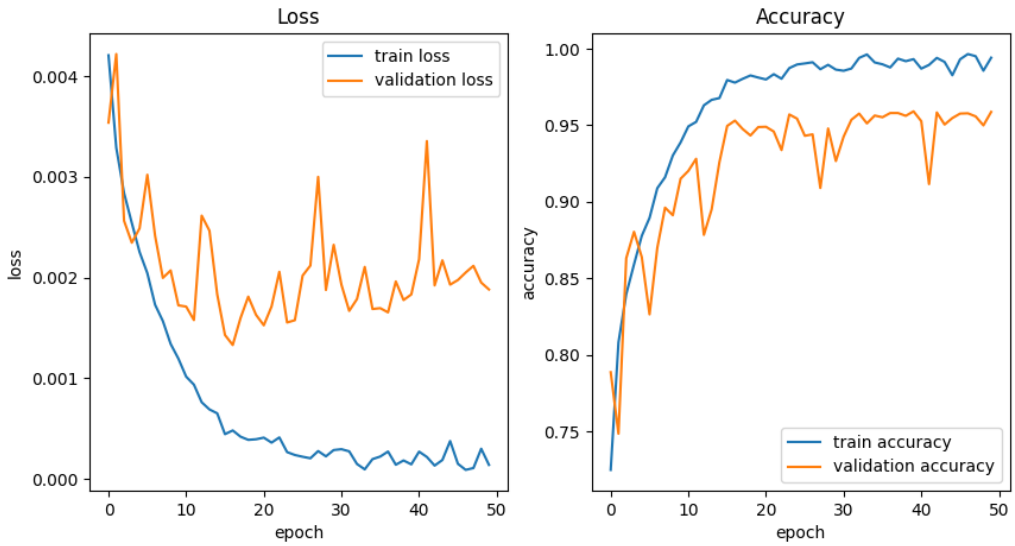
实验结果和分析

分别令 dropout rate 为 0、0.1、0.2，进行完整 50 个 epoch 的训练，选取验证集损失最小的模型作为最终模型，结果如下：

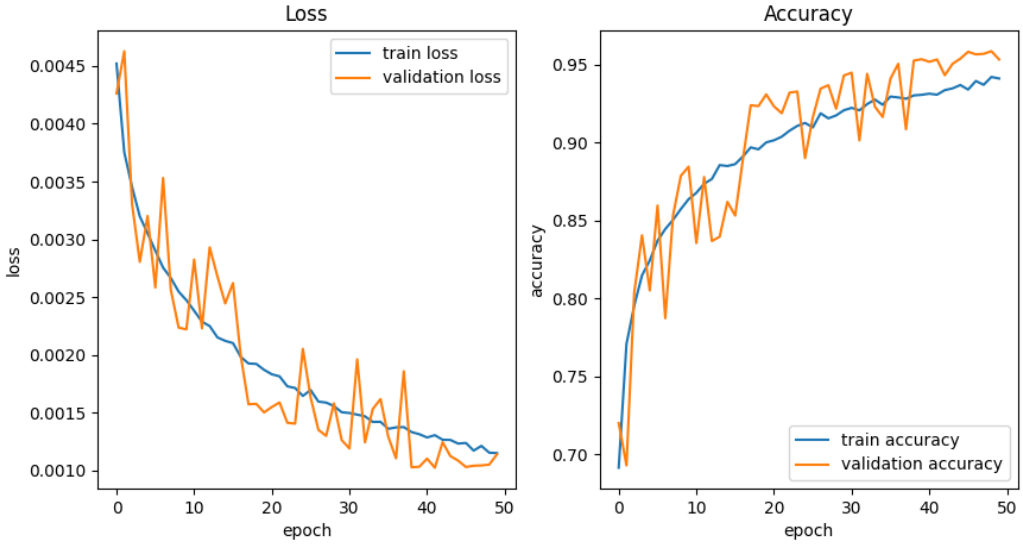
Dropout	Epoch	训练集损失	训练集准确率	验证集损失	验证集准确率	测试集损失	测试集准确率
0	17	0.000482	0.9778	0.001330	0.953	0.001290	0.9585
0.1	42	0.001306	0.9306	0.001025	0.9532	0.000985	0.9579
0.2	45	0.002268	0.8788	0.001591	0.9250	0.001597	0.9254

训练中损失曲线和准确率曲线如下：

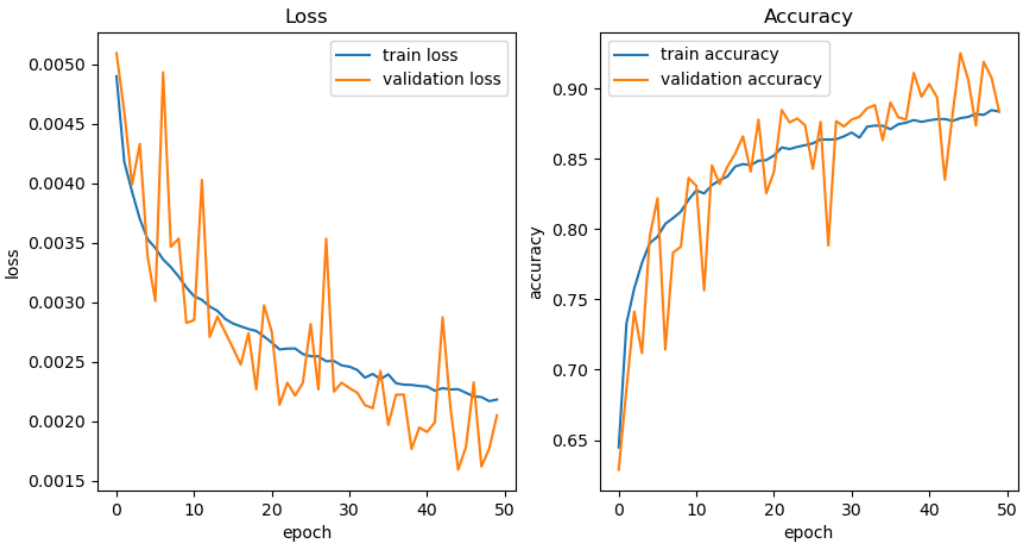
dropout rate = 0:



dropout rate = 0.1:



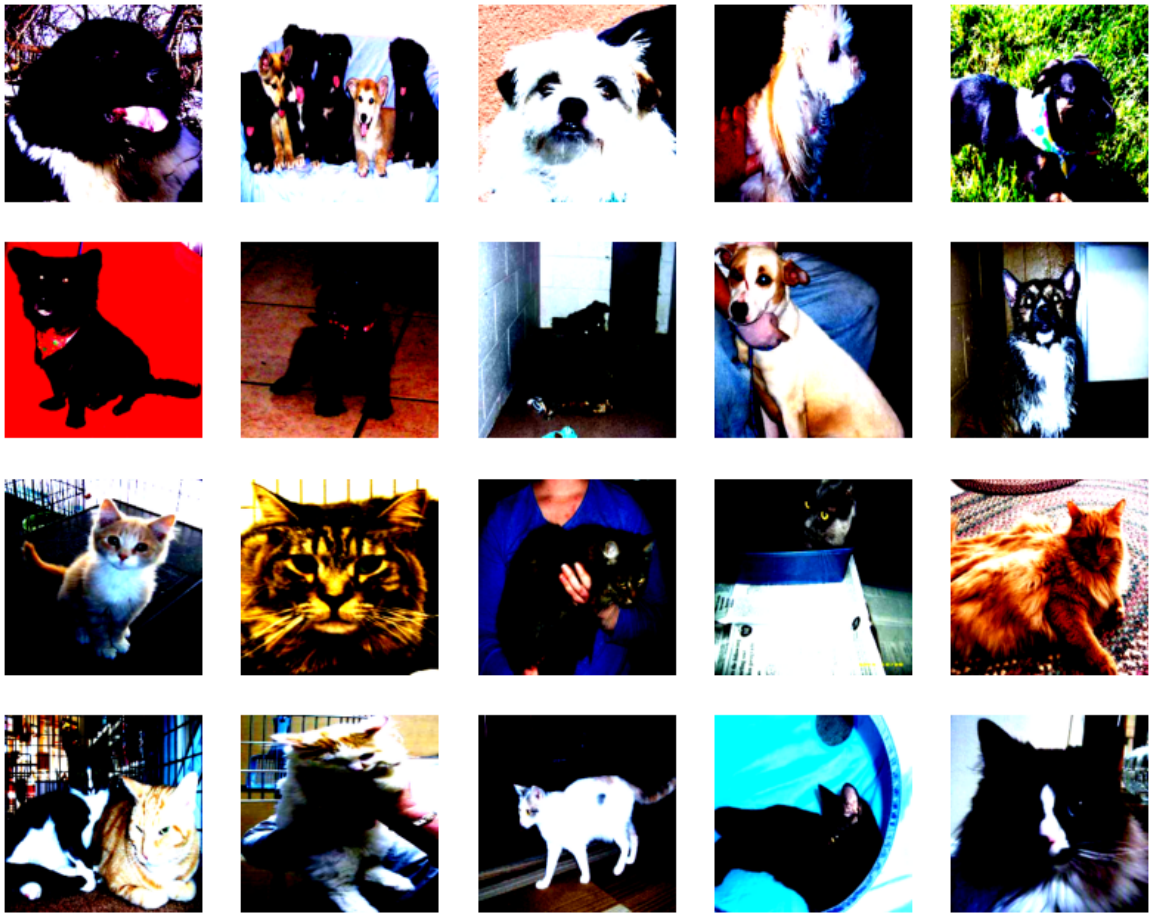
dropout rate = 0.2:



分析：

- Dropout rate 越高，模型收敛越慢。Batch Normalization 和 Dropout 都有对模型进行正则化的作用，因此本任务中 Dropout 并没有体现出效果提升的作用（在 50 个 epoch 以内）。
- Dropout rate 为 0 时，模型很快就收敛（17 个 epoch），后续进入过拟合状态。
- 验证集的规模太小，导致验证损失曲线不够平滑，震荡比较严重。

以效果最好的 dropout rate = 0 的模型为例，测试集错分了 160 张狗的图片，137 张猫的图片，其中的 20 张图片如下，前两行为狗的图片，后两行为猫的图片：（经过了预处理）



问题

由于本实验运算量较大，模型在 GPU 上进行训练，训练完成后保存了模型参数。然而在 CPU 机器上（Python 3.8.16，PyTorch 2.0.0）加载该模型参数进行测试时，发现测试结果有所差异：

	测试集损失	**测试集准确率**
GPU 平台	0.001290	0.9585
CPU 平台	0.001740	0.9470

猜测可能的原因是不同硬件平台浮点数精度不同，以及 Python 和 PyTorch 的版本不同。