浙江大学

数据挖掘课程报告

报告题目　FashionMNIST 数据集图像分类

姓名与学号　王世龙、吕浩斌

指导教师　马东方

学号　王世龙（22134022）、吕浩斌（22134047）

目录

备注：

课程报告主要是交代了一些课堂展示没办法呈现的细节问题，尽量写得精炼且完整。我们并不打算完整呈现代码细节和运行环境，因为这不是重点，助教感兴趣的话参看以下链接，已经建了 github repository，也做了代码注释和说明。

https://github.com/rookiewangsl/Fashion

贡献排序：王世龙、吕浩斌

一. 任务描述

1 数据集

我们组选择的是 CNN 问题中的 FashionMNIST 图像分类问题，要完成的是一个服装图片种类识别的分类问题。

数据集采用的是 FashionMNIST 数据集，由德国的服装公司 Falando 提供。FashionMNIST 中一共包含 6 万张图片，被分为了 10 类，所以是监督学习。每张图片都是 28*28 的单通道黑白图片。十种类别分别为“T-shirt、Trouser、Pullover、Dress、Coat、Sandal、Shirt、Sneaker、Bag、Ankle Boot”，都是服装穿着类。

我们选取了两种不同的网络结构进行网络训练，运用了各种常见的训练方法来节省训练时间、提高收敛速度和预测准确率，并对这两种网络结构进行了性能上的对比，并将结果可视化呈现，并与 Benchmark 进行了对比。由于课程时间和网络训练时间问题，许多超参数没有太多的进行调参，所以网络性能上还有进步的空间。

2 网络结构

2.1 结构选择

对于图像分类问题，常用CNN网络进行特征提取和数据降维，且分类效果经过实践确实很好，换句话说CNN是专门为了图像的数据特点进行设计的。所以我们采用了两种CNN的网络结构进行尝试：典型的两层简单CNN和残差网络Resnet18。具体性能对比在第四部分。

2.2 CNN

由于FashionMNIST数据集出现较早，用来训练的CNN网络结构其实已经比较成熟，所以我们决定用一个简单的两层的CNN，里面的卷积层参数采用常见的设置。

```
self.cnn1 = nn.Conv2d(
```

in_channels=1, out_channels=32, kernel_size=5, stride=1, padding=2)

self.cnn2 = nn.Conv2d(

in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=2)

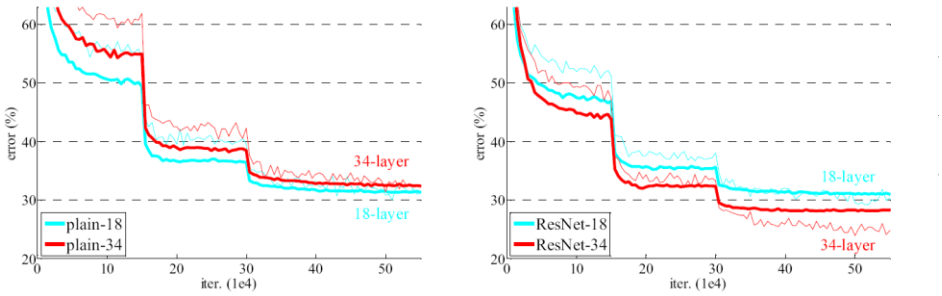
在每个卷积层后级联BN层和最大池化层，将两个卷积块串联后再连接两个全连接层得到最后的分类输出。激活函数使用的是 Relu。

2.3 Resnet18[1]

因为上课讲过残差网络，通过层之间的直接连接解决深度学习网络因为加深导致的性能下降问题。而且Resnet的原论文也是针对图像数据集，对CNN深层网络进行改进的。何凯明的原论文到目前为止已经有了95121次的引用量，惊叹于这篇论文的影响力，我们拜读了原文，对Resnet有了一定的理解，文章中呈现的性能提升非常显著。我们觉得Resnet也可以用于我们的分类问题，于是进行了相应的尝试。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.



图一 Resnet原文给出的结构

我们根据论文中给出的网络结构搭建了对应的 18 层 CNN；同时 pytorch 中提供了 resnet 网络的直接调用接口。由于 Resnet 的输入是三通道的图片，所以我们将原数据集的每一张图片进行了三个通道的重叠作为网络的输入。

二、训练方法与参数设置

1 数据集划分和预处理

没有用老师预先给出的数据集，调用了pytorch中 datasets.FashionMNIST 接口下载数据，自己定义了继承的dataloader类。将数据集打乱后按照4:1的比例划分成训练集和验证集进行交叉验证。在训练之前取出了1万张图片作为最后的测试集。

2 损失函数

采用的是多分类问题常用的交叉熵，能很好地反应分类问题的loss。

$$\mathrm{L}=\frac{1}{\mathrm{N}}\sum_{i}L_i=-\frac{1}{N}\sum_{i}\sum_{c=1}^{M}y_{ic}\log(p_{ic}) \tag{1}$$

3 优化器

优化器用的是SGD with Nesterov Momentum，是一种对动量梯度下降的改进。以下是动量梯度下降（式2）和Nesterov（式3）的梯度更新公式对比。可以看出，Nestrov的区别在于更新中多了这一步梯度更新的项，能够提前看到后面的情况，实践证明这种方法相对传统梯度下降有更快的收敛速度和性能提升。

$$\mathrm{d_i}=\beta d_{i-1}+g(\theta_{i-1}) \tag{2}$$

$$\theta_{\mathrm{i}}=\theta_{i-1}-\alpha d_i$$

$$\mathrm{d_i}=\beta d_{i-1}+g(\theta_{i-1}-\alpha\beta d_{i-1}) \tag{3}$$

$$\theta_{\mathrm{i}}=\theta_{i-1}-\alpha d_i$$

4 学习率调整策略

ReduceOnPlateau，实用的策略。验证集的loss在多个epoch后没有下降的时候，降低学习率，寻找最优点。

三、训练用到的 Tricks

为了抵抗过拟合以及加速收敛，采用了一些常见的训练 trick。

1 数据集预处理

为了增强模型的泛化能力，对图片数据施加了Randomcrop和erasing，随机覆盖和填充。但是经过实践，这样的预处理虽然增强了模型泛化能力，但是略微降低了模型在本数据集中的表现（预测准确率）。

2 权重初始化

根据层的类型做了常见的初始化，卷积层和线性层用 Xvaier，BatchNorm 层用 Normal，加速收敛。另外合适的初始点选择有助于模型收敛到全局最优点，而非局部最优。

3 BN 与 Dropout

在卷积块后添加 BN 层，防止梯度爆炸，同时加速收敛。在最后添加 Dropout 层，随机丢弃，防止模型过拟合。

4 Early Stopping

当 Validloss 数个 epoch 后仍未下降，停止训练，防止过拟合。

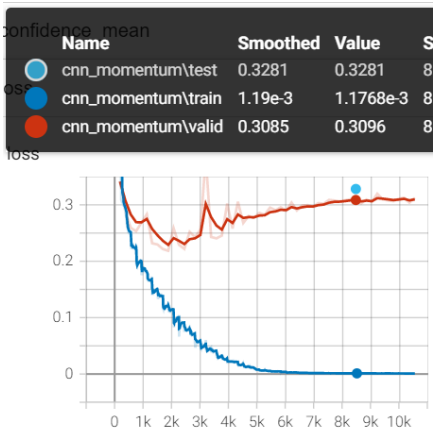
5 迁移学习（预训练）

在训练 Resnet18 时预先载入 ImageNet 上预训练的参数。因为图像数据集有共通之处，采用预训练的参数技能加速收敛，又能有效提高收敛点的性能。

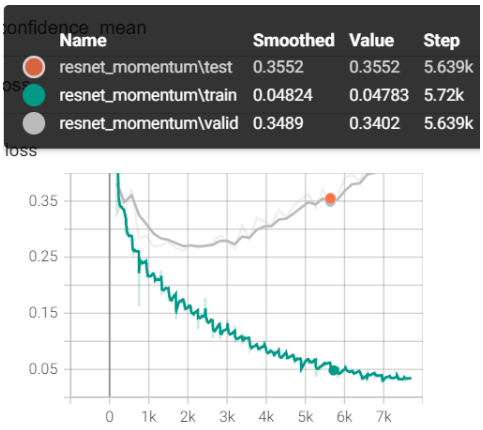
四、实验结果

首先是模型收敛情况，以 0.01 的初始学习率大约训练了 40 个 epoch，得到以下的 loss 曲线。从图中可以看出，两层的 CNN 和 Resnet18 模型已经很好的收敛，Resnet18 即将出现过拟合，早停及时停止了训练。

CNN 的 Testloss 为 0.3281，Resnet 的 Testloss 为 0.3552，CNN 相对 loss 更低。

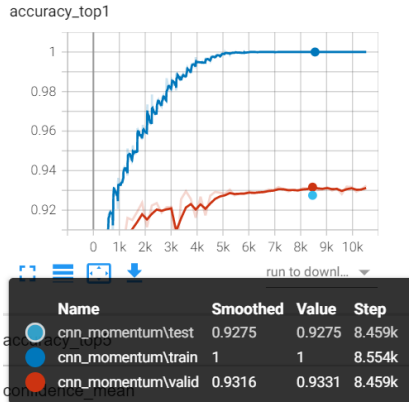


图二 两层CNN的loss曲线

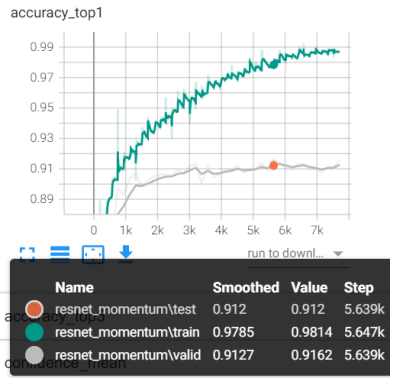


图三 Resnet18的Loss曲线

在预测准确率这一方面，CNN 的准确率达到了 92.75%，而 Resnet 为 91.52%；CNN 比 Resnet18 的性能更好，与直观感觉相反。我们猜想可能的原因是数据集过于简单，18 层的深度网络容易出现过拟合。



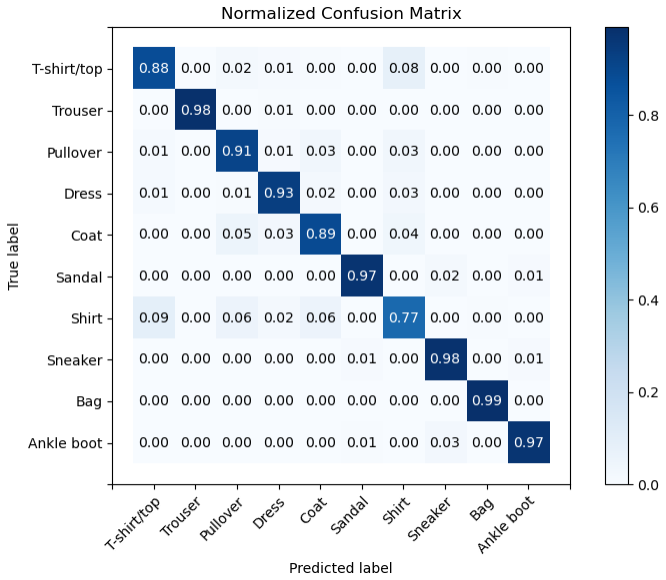
图四 CNN 预测准确率



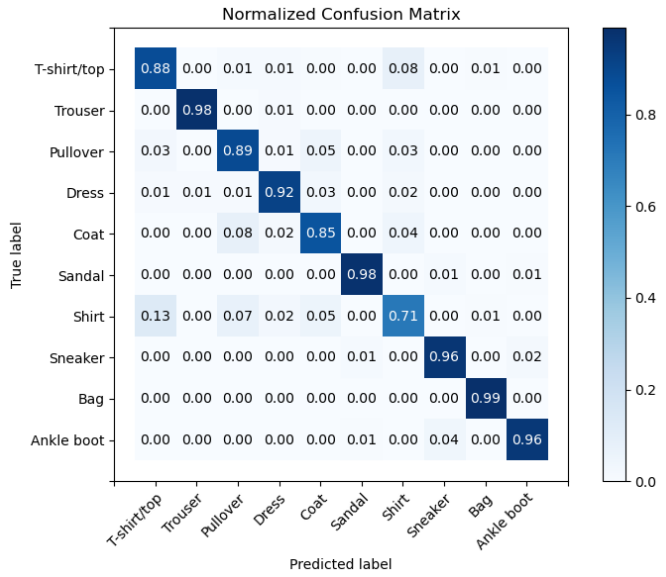
图五 Resnet预测准确率

为了直观看出 10 种类别中类别之间误判的概率，我们绘制了判决模糊矩阵如下。CNN 网络中误判最严重的是 Shirt 类，最容易误判成 T-shirt，判断最准确的是 Bag 类，与人眼直观判断感觉相似。

Resnet 的模糊矩阵得出的结论与 CNN 相似，只是具体的准确率有些微数值上的区别。



图六 CNN 模糊矩阵



图七 Resnet 模糊矩阵

最后我们用了一张测试样例，利用保存的模型参数对输入做了预测。模型输出的类别也是正确的结果。

图八 测试样例图

```
(py38) rookie@zhazijie-MS-7D17:~/fashion$ python infer.py
Loading best model from experiments/cnn_momentum/model_best.pth.tar...
load model success
result: ankle boot
```

图九 Inference 模型输出

五、Benchmark 与参考文献

我们在 github 上找到了 FashionMNIST 数据集的主页，上面给出了各种网络结构预测准确率的 baseline。使用双层 CNN 的预测准确率在 91%~93%之间。

很多论文中的准确率有时候也难以复现，很多 trick 可能作者并没有在文章中展示出来，，所以我们训练的模型准确率 92.75% 属于正常范围，且较为优秀，

README.md

Classifier	Preprocessing	Fashion test accuracy	MNIST test accuracy	Submitter
2 Conv+pooling	None	0.876	-	Kashif Rasul
2 Conv+pooling	None	0.916	-	Tensorflow's doc
2 Conv+pooling+ELU activation (PyTorch)	None	0.903	-	@AbhirajHinge
2 Conv	Normalization, random horizontal flip, random vertical flip, random translation, random rotation.	0.919	0.971	Kyriakos Efthymiadis
2 Conv <100K parameters	None	0.925	0.992	@hardmaru
2 Conv ~113K parameters	Normalization	0.922	0.993	Abel G.
2 Conv+3 FC ~1.8M parameters	Normalization	0.932	0.994	@Xfan1025
2 Conv+3 FC ~500K parameters	Augmentation, batch normalization	0.934	0.994	@cmasch
2 Conv+pooling+BN	None	0.934	-	@khanguyen1207
2 Conv+2 FC	Random Horizontal Flips	0.939	-	@ashmeet13
3 Conv+2 FC	None	0.907	-	@Cenk Bircanoğlu

图十 数据集 Benchmark

参考文献：

[1] HE K, ZHANG X, REN S等. Deep residual learning for image recognition[J]. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, 2016-Decem: 770–778. DOI:10.1109/CVPR.2016.90.