

人工智能基础

大作业

fashionMNIST 分类

学 号	2017011589
姓 名	吾尔开西
专 业	自动化
日 期	2019.10.12

目录

kaggle 排名.....	2
数据分析与处理.....	3
1、图片分析.....	3
2、数据划分.....	3
数据增强.....	4
模型选择.....	4
1、3 层卷积简单模型	4
2、VGG16 修改.....	4
3、DPN 修改.....	5
4、resnet 18 修改	5
5、resnet 50 修改	6
6、Google Net	6
参数调整.....	6
1、学习率与优化器.....	6
2、数据批量大小 batch size.....	7
3、旋转角度.....	7
4、模型参数.....	7
训练策略.....	8
1、控制随机种子	8
2、预训练	8
3、Bagging 投票	8
总结.....	9
参考资料.....	9

kaggle 排名

用户名: HyperionSoldier


用户名寓意: 科幻小说《海伯利安》中的伯劳是未来人类所创造之终极智能体, 以“海伯利安战士”为名象征着人工智能的强大力量以及使用 AI 解决问题的决心。

private 榜: Score 0.91514 排名第 19

19

▼ 4

HyperionSoldier



0.91514


25

14h

public 榜: Score 0.91466 排名第 15

15

HyperionSoldier




0.91466

25

14h

Your Best Entry [↑](#)

Your submission scored 0.91466, which is an improvement of your previous score of 0.91400. Great job!

 Tweet this!

数据分析与处理

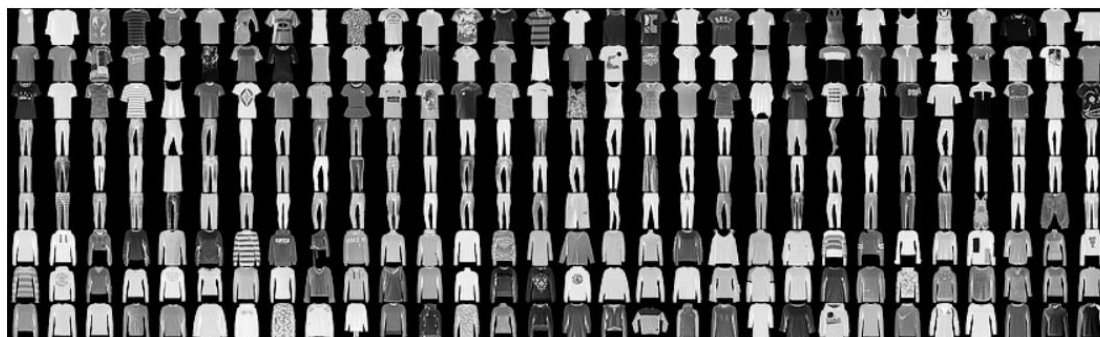
1、图片分析

本次分类任务取的是 fashionMNIST 经过处理后的子集，训练数据集有 3 万张图片，测试数据集有 5 千张图片。

本次任务部分图片如下：



fashionMNIST 原数据集中部分图片如下：



可以看出，与原数据集相比，本次任务对图像进行了随机旋转处理，加入了椒盐噪声，给图片加了随机形状和宽度的白边。

为了去除椒盐噪声，我们可以使用 3×3 大小的中值滤波，上面三排图片的处理结果如下。可以看出椒盐噪声被很好地去除，但空域滤波必然会导致图片信息的丢失，因此是否使用中值滤波是我们训练过程中的一个备选项。



2、数据划分

kaggle 网站上一天只有两次提交机会，为了能在本地更准确地测试模型性能，我们把 3

万张训练数据集分出 10%作为 validation set.

数据增强

由上一部分我们可以看出，衣服图片有被随机旋转，最多大约有 30 度的倾斜角度。图片都是直立的，基本左右对称，但鞋子有左朝向的也有右朝向的。衣物都在图片正中央，但基本都没有占满整个图片。

基于上面所观察到的特征，我们对训练集图片进行如下的数据增强：

- 在图片四周填补 2 个像素的边缘，并从填补后的图片中随机取出 28*28 大小的子图片。
- 对图片进行 0 到 15 度的随机旋转
- 以 5%的概率对图片进行水平翻转
- 将图片像素值归一化到 0~1 的范围
- 使用数据集总体的平均值（0.2926）和方差（0.3343）对像素值进行归一化。

对 validation 集我们不引入随机性，因此只是进行像素值的归一化。

模型选择

1、3 层卷积简单模型

首先使用了有 3 层卷积的简单模型，卷积之间用 batchnorm 层和 relu 层连接，提高模型的泛化能力。

这种简单模型在 validation 集最高能达到 90.9%的准确率，在 kaggle 公榜上能达到 87%的准确率。

2、VGG16 修改

VGG 模型常被用于各种分类任务，曾获得 2014 年 ILSVRC 竞赛的第二名，下面是 VGG 模型的参数配置([Simonyan & Zisserman, 2014](#))。

本次任务我使用了 VGG16 的修改版（表格中的 C），首先将 RGB 3 通道的输入改为 1 通道灰度，再将最后 3 个卷积层和 maxpool 池化层去掉，最后将全连接层的维度改为 256-512-10

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

该模型在 validation set 中准确率能达到 90%以上，在 kaggle 公榜上能达到 89%，

3、DPN 修改

duall path networks 是在 dense net 之后被提出的模型结构，在图像分类任务上有一定的提升，但在本任务中与 VGG16 性能差不多。

4、resnet 18 修改

残差网络是近几年来深度学习中应用广泛的一种网络结构，实验表明，使用残差结构在一定程度上兼顾模型的表达能力（网络深度）和泛化能力。

在使用前几个网络时，我的模型在训练集上的准确率能达 98%以上，但在 validation set 上的准确率最高只有 90%，而在 kaggle 公榜上的准确率又会下降 1 到 2 个百分点，我认为这是模型泛化能力不足带来的问题，于是决定采用 resnet，使用较深的网络同时保持较好的泛化能力。

下面是论文中 resnet 的参数配置(He, Zhang, Ren, & Sun, 2016)。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	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

我对模型进行了一定的修改：

- (1) 将 RGB 3 通道的输入改为 1 通道灰度值的输入。
- (2) 28×28 的图片经过 3 组卷积层后将变成 7×7 的大小，经过 4 组卷积层后将变成 4×4 的大小，需要对应调整最后的 average pool 池化层的尺寸。
- (3) 输出类别数是 10。
- (4) 在调整参数的阶段我尝试了多种模型参数，所以并不是严格的 resnet18，在下一部分会详细说明。

resnet 18 模型结构在 validation set 最高能达到 92% 的准确率，平均水平为 91%，在 kaggle 公榜上能达到 90.2% 的准确率，我最终使用 bagging 投票方法时用的基本也是 resnet 模型。

5、resnet 50 修改

由于 resnet 的特性是能使用较深的网络，所以我还尝试了 resnet 50，但是模型效果与 resnet 18 差不多。这也是合理的，因为本问题图片较小，resnet 18 基本上是足够的。

6、Google Net

在比赛最后阶段进行 bagging 投票时，我为了进一步提高结果的泛化能力，将 Google Net 和 resnet 的多个模型进行 ensemble，得到了不错的结果。

Google Net 使用单个模型进行训练和预测时结果参差不齐，最高可达 92.0%，最低有 90.3% ([Szegedy et al., 2015](#))

参数调整

参数调整在上一部分所述的 5 个模型中都有进行，但由于最终使用的模型主要是 resnet18，所以下面的结论主要基于 resnet 18，且每个结果均是 2 到 3 次实验的平均值。

1、学习率与优化器

优化器尝试了 Adam 优化器和 SGD 优化器。

使用 SGD 优化器时在训练过程中有学习率的调整，会在第 15 个 epoch 和第 30 个 epoch 时分别将学习率降为原来的 10%。优化器的动量参数 momentum=0.9。

下面是使用 SGD 时不同学习率下的一组实验结果：

学习率	Validation 准确率/%
5e-4	90.05
1e-3	90.88
5e-3	90.89
1e-2	90.77

下面是使用 Adam 时不同学习率下的一组实验结果：

学习率	Validation 准确率/%
5e-4	90.52
1e-3	91.56
5e-3	91.10

1e-2	90.88
------	-------

可以看出使用 Adam 时整体准确率高于 SGD，使用 1e-3 的学习率效果最佳。

2、数据批量大小 batch size

在对模型进行训练时，一次需要将 batch size 大小的数据送入进行随机梯度下降，下面是使用不同 batch size 训练时模型的性能：

Batch size	Validation 准确率/%
32	90.7
64	91.53
128	91.44

batch size 为 64 时性能较好。

3、旋转角度

数据增强部分提到，我们对 train 数据集图片数据进行了随机旋转增强，最大旋转角度的变化带来的性能差异较大，如下表所示：

最大旋转角度/°	Validation 准确率/%
8	90.86
15	91.53
20	91.13
30	91.30

视最大旋转角度为 15 度时最优。

4、模型参数

resnet 18 中有多个卷积 layer 块（默认为 4 个）；每个 layer 块中有若干 Basic block 残差残差结构；每个残差结构由两个卷积层组成，卷积层之间由 batch normalization 和 Relu 激活函数所连接，输入通过 down sample 和卷积层的结果相加，再通过一个 Relu 作为残差结构的输出。

用一个列表 layers 表示模型里每个 layer 块中 Basic block 的个数，如 layers=[2,2,2,2] 表示有 4 个 layer 块，layer 块中的 Basic block 个数都为 2。

下面是改变 layers 列表时模型的训练结果：

layers 列表值	Validation 准确率/%
[2, 2, 1, 0]	91.3
[2, 2, 2, 0]	91.53
[2, 2, 2, 1]	91.77
[2, 2, 2, 2]	91.87

上面是在 validation set 上的准确率，实际上，在 kaggle 公榜上各个模型结果有好有坏。我们在最终提交时也是用 bagging 投票的方法将不同模型进行组合，并不是用单一的模型。

另外，我们还尝试改变了 resnet 的全连接层层数，也将训练结果用到了 bagging 中。

训练策略

1、控制随机种子

在训练过程中, 我们认识到随机性对模型结果造成的影响, 比如数据集的划分是随机的, 用不同的划分结果进行训练, 预测的结果可能相差高达 1%。再如训练集数据的 shuffle 也有一定的随机性。

我们对所有有随机性的地方都用一个随机种子进行控制, 一方面方便复现训练结果, 另一方面我们可以通过改变随机种子来得到多个训练结果, 更全面地查看模型的性能, 提高了泛化能力。

2、预训练

模型参数的起始值会对训练结果造成较大的影响, 好的预训练参数能带来一定的性能提升。我们前面提到本次任务数据取的是 fashionMNIST 经过处理后的子集, 因此我们想到能用 fashionMNIST 进行预训练, 下面是 resnet 的不同 layers (见上一部分的 4.) 下使用预训练模型的结果:

layers 列表值	Validation 准确率/%
[2, 2, 2, 0]	91.93
[2, 2, 2, 1]	91.4
[2, 2, 2, 2]	91.67

本次任务的数据图片中加入了椒盐噪声, 为了使子集和原数据集更相似, 我们尝试了对二者都使用中值滤波进行预处理, 但在 kaggle 公榜上的结果一般, 可能是由于中值滤波丢失了原图的一些信息。

我们也将一些预处理模型的预测结果用到了 Bagging 投票中。

3、Bagging 投票

模型 ensemble 是 kaggle 竞赛中提升性能的常用手段, Bagging 投票是其中一种方法。我们在训练模型时并不知道该模型在测试集上的性能, 而我们自己从训练集中分出来的 validation 集由不一定有很好的代表性, 因此训练出来的模型泛化能力往往很差。

为了解决这个问题, Bagging 投票方法指出可以综合多个模型的预测结果, 即对每一条测试图片, 用多个模型预测它的类别, 对这些预测结果进行投票, 将票数最高的作为最终结果。

下面是一种组合方式, 共 18 个模型。其中, 同样参数的模型在训练时使用了不同的随机种子。这个方法让我的准确率大为提升, 在 kaggle 公榜上从单模型的 90.2%提升到 91.466% 的准确率

模型参数	Validation 准确率/%
Resnet, layers[2,2,2,0], 预训练	91.77
Resnet, layers[2,2,2,0], 两层 fc	91.4
Resnet, layers[2,2,1,0]	91.3
Resnet, layers[2,2,2,2]	91.533
Resnet, layers[2,2,2,0]	90.9
Resnet, layers[2,2,2,0]	91.3
Resnet, layers[2,2,2,0]	91.367

Resnet, layers[2,2,2,0]	91.467
Resnet, layers[2,2,2,0]	92.233
Resnet, layers[2,2,2,0]	90.500
Resnet, layers[2,2,2,0]	91.500
Resnet, layers[2,2,2,2], 两层 fc, 数据平均划分	91.87
Resnet, layers[2,2,2,1], 60epoch, 数据平均划分	91.5
Resnet, layers[2,2,2,1], 数据平均划分	92.27
Resnet, layers[2,2,2,2], 数据平均划分	91.6
Resnet, layers[2,2,2,0], batchsize128	91.8
Resnet, layers[2,2,2,0], 使用 SGD, 学习率 1e-2	91.733
Resnet, layers[2,2,2,0]	91.933

总结

本次大作业我体验到了深度学习的强大能力，联想到科幻小说《海伯利安》里人工智能的无所不能，有一种热血沸腾的感觉。深度学习是近几年兴起的领域，这次大作业也让我接触到研究前沿，十分受用。

虽然排名给了我一定的压力，几天不提升性能就会被人超过，但每次改善性能后排名的上涨也让我很有成就感。

为了提升模型性能，我研究了数据集，尝试了多种模型，进行了数据增广，对训练参数以及模型参数进行调整，最后用 bagging 投票的模型组合方法提高泛化能力，基本上是一次完整的数据挖掘任务流程。

虽然这次大作业花了我较长时间，尝试了多种方案，调参也用了很长时间，但体会到了独立完成一个完整项目的乐趣，收获了成就感，不仅提高了我的编程能力，加深了我对深度学习的理解，还提高了我解决问题的能力以及抗压能力，收获满满，也算是付出得到了回报。

参考资料

<https://github.com/zalandoresearch/fashion-mnist>

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Simonyan, K., & Zisserman, A. J. a. p. a. (2014). Very deep convolutional networks for large-scale image recognition.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). *Going deeper with convolutions*. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

