上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

统计学习理论与方法

2019-2020 秋

完成人： xx 019xxxxxxxxx

目录

摘要

本次课程设计的任务是处理 Kaggle 上发布的一个图像分类的问题，利用给定的训练集，使用不同的方法完成对发布测试集中图像的分类。在本次的课程设计中，我先使用 SVM、CNN 和 ResNet18 模型对原始数据进行训练，获得初步的得分。接着对同学建立的数据集进行预处理，去除噪声数据，并使用 SVM、CNN 和 ResNet18 模型对扩充后的数据进行训练，将结果与原始数据训练得到的结果进行比较，发现扩充的数据有助于实验结果的提升。然后，再使用数据增强的方法继续扩大数据集，并使用 CNN、ResNet18、ResNet101 和 VGG16 模型对数据增强后的数据集进行训练，比较后发现实验结果获得了较大的提升。最后，再采用一种对实验结果投票的方法，进一步提升了测试集上的得分。在 Kaggle 上任务截止后，我在 public leaderboard 上的得分为 0.91651，排名第 4，在 private leaderboard 上的得分为 0.91977，也是排名第 4。

一、问题描述

利用给定的训练集，使用不同方法完成对发布测试集中每张图像的分类。并利用同学自己建立的数据集，进一步提升实验结果。

二、模型介绍

本次课程设计中，主要使用了 SVM、CNN、ResNet、VGG16 等常用于图像分类任务的模型，下面对这些模型做一些简单的介绍。

2.1、支持向量机(SVM)

支持向量机(Support Vector Machine, SVM)是一类按监督学习的方式对数据进行二元分类的广义线性分类器，随着最大边距决策边界、核方法等理论的研究发展，SVM 衍生出一系列改进和扩展算法，在人像识别、文本分类等模式识别问题中得到广泛应用。

给定训练集样本$D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_m}, y_m)\}, y_i \in \{-1, +1\}$，分类学习最基本的想法就是基于训练集 D 在样本空间中找到一个划分超平面，将不同类别的样本分开。但能将训练样本分开的划分超平面可能有很多，如图 1.1 所示。直观上看，应当选择使分类结果最鲁棒的划分超平面，如图 2.1 中的加粗直线，该划分超平面对训练样本局部扰动的容忍性最好，对未见示例的泛化能力最强。

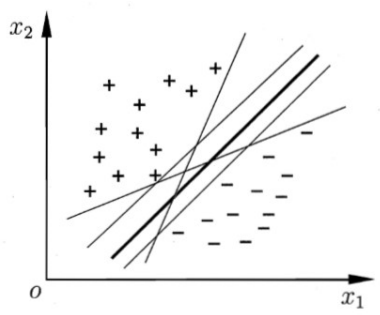


图 2.1 存在多个划分超平面将两类训练样本分开[1]

在样本空间中，划分超平面可通过如下线性方程来描述：

$$\boldsymbol{w^T x} + b = 0 \tag{2.1}$$

将其记为$(\boldsymbol{w}, b)$，样本空间中任一点$\boldsymbol{x}$超平面$(\boldsymbol{w}, b)$的距离可写为

$$r = \frac{|\boldsymbol{w^T x}+b|}{||\boldsymbol{w}||} \tag{2.2}$$

假设超平面$(\boldsymbol{w}, b)$能将训练样本正确分类，即对于$(x_i, y_i) \in D$，若$y_i = +1$，则有$w^T x_i + b > 0$，若$y_i = -1$，则有$w^T x_i + b < 0$，所以有：

$$\begin{cases} w^T x_i + b \geq +1, \ y_i = +1 \\ w^T x_i + b \leq +1, \ y_i = -1 \end{cases} \tag{2.3}$$

如图 2.2 所示，距离超平面最近的几个训练样本点使等号成立，这些点被称为“支持向量”(support vector)。两个异类支持向量到超平面的距离之和为

$$\gamma = \frac{2}{||\boldsymbol{w}||} \tag{2.4}$$

γ就被成为“间隔”(margin)。要找到具有最大间隔的划分超平面，就是要找能满足(1.3)约束的参数$\boldsymbol{w}$和b，使得γ最大，即：

$$\max_{\boldsymbol{w},b} \frac{2}{||\boldsymbol{w}||} \tag{2.5}$$

$$\text{s.t.} \ \ y_i(\boldsymbol{w^T x_i} + b) \geq 1, \text{i} = 1,2, \ldots, \text{m}.$$

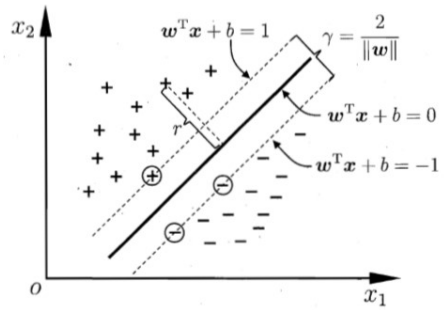


图 2.2 支持向量与间隔[1]

将最大化问题转换为最小化问题，可以得到：

$$\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2 \tag{2.6}$$

$$\text{s.t.} \ \ y_i(\boldsymbol{w^T x_i} + b) \geq 1, \text{i} = 1,2, \ldots, \text{m}.$$

式(1.6)就是支持向量机(Support Vector Machine, SVM)的基本型。

注意到支持向量机是一个有约束问题，可以采用拉格朗日乘子法将其转换为对偶问题(dual problem)，对式(2.6)的每条约束添加拉格朗日乘子$\alpha_i \geq 0$，则该问题的拉格朗日函数可以写为：

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)) \tag{2.7}$$

其中，$\alpha = (\alpha_1; \alpha_2; \ldots; \alpha_m)$.令$L(\boldsymbol{w}, b, \boldsymbol{\alpha})$对$\boldsymbol{w}$和$b$的偏导为 0，可得：

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i} \tag{2.8}$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i \tag{2.9}$$

将式(2.8)代入式(2.7)，即可将$L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 中的$\boldsymbol{w}$和b 消去，再考虑式(2.9)的约束，就得到式(2.6)的对偶问题：

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j} \tag{2.10}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \mathrm{i} = 1,2, \ldots, \mathrm{m}.$$

解出$\boldsymbol{\alpha}$后，求出$\boldsymbol{w}$和b即可得到模型：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i^T x} + b \tag{2.11}$$

在训练样本线性可分的情况下，很容易找到一个划分超平面可以讲训练样本正确分类。然而在现实任务中，原始样本空间内也许不存在一个能正确划分两类样本的超平面，对于这种问题，可以将样本从原始空间映射到一个更高维的特征空间，使得样本在这个特征空间内线性可分，通常使用的是径向基函数核(Radial Basis Function, RBF)，其公式可表示为$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(\frac{-||\boldsymbol{x_i} - \boldsymbol{x_j}||^2}{2\sigma^2}\right)$。

2.2、卷积神经网络(CNN)

卷积神经网络(Convolutional Neural Network) 是由具有可学习的权重和偏置常量的神经元组成。通常包含卷积层、线性整流层、池化层和全连接层。

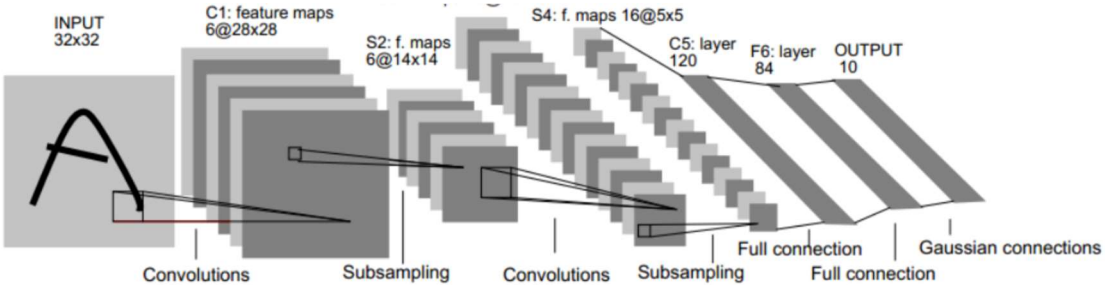


图 2.3 卷积神经网络的一般结构

卷积层(Convolutional layer)由若干卷积单元组成，每个卷积单元的参数都是

通过反向传播算法优化得到的。卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网络能从低级特征中迭代提取更复杂的特征。

线性整流层通常使用 Rectified Linear Units (ReLU)作为激活函数。

池化层(Pooling layer) 是将卷积层之后得到的维度较大的特征，将特征切成几个区域，取其最大值或平均值，得到新的、维度较小的特征。通常使用最大池化或者平均池化的方法进行降维。

全连接层(Fully-Connected layer)就是把所有局部特征结合变成全局特征，用来计算最后每一类的得分。

2.3、深度残差网络(ResNet)

残差网络(Residual Network)是用来解决网络加深带来梯度消失等问题的一种有效途径。随着网络深度增加，网络的准确度应该同步增加，但是网络深度增加的一个问题在于这些增加的层是参数更新的信号，因为梯度是从后向前传播的，增加网络深度后，比较靠前的层梯度会很小。这意味着这些层基本上学习停滞了，也就是梯度消失问题。深度网络的第二个问题在于训练，当网络更深时意味着参数空间更大，优化问题变得更难，因此简单地去增加网络深度反而出现更高的训练误差。

ResNet 提出了一个残差模块可以帮助训练更深的网络，残差模块的结构如图 2.4 所示。深度网络的训练问题称为退化问题，残差单元可以解决退化问题的背后逻辑在于此：想象一个网络 A，其训练误差为 x。现在通过在 A 上面堆积更多的层来构建网络 B，这些新增的层什么也不做，仅仅复制前面 A 的输出。这些新增的层称为 C。这意味着网络 B 应该和 A 的训练误差一样。那么，如果训练网络 B 其训练误差应该不会差于 A。但是实际上却是更差，唯一的原因是让增加的层 C 学习恒等映射并不容易。为了解决这个退化问题，残差模块在输入和输出之间建立了一个直接连接，这样新增的层 C 仅仅需要在原来的输入层基础上学习新的特征，即学习残差，会比较容易。

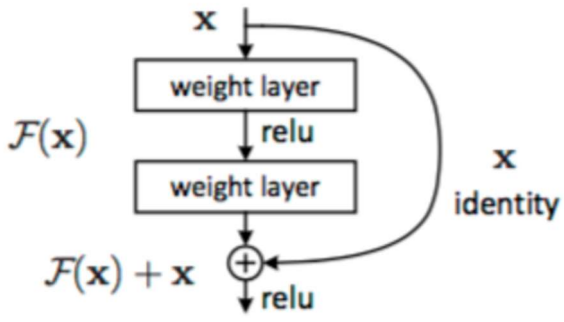


图 2.4 残差单元

2.4、VGG16

VGG16 是由牛津大学 VGG 组提出的，采用连续的几个 3*3 的卷积核代替 AlexNet 中较大的卷积核。对于给定的感受野（与输出有关的输入图片的局部大小），采用堆积的小卷积核是优于采用大的卷积核，因为多层非线性层可以增加网络深度来保证学习更复杂的模式，而且代价还比较小。VGG16 的网络结构如图 2.5 所示

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					

图 2.5 VGG16 的网络结构

三、实验操作及结果分析

3.1、数据集处理

3.1.1、原始数据集

在 Kaggle 发布的数据集中，训练集共包含 15 个图像类别，每个类别中包含 400 张大小为 28*28 的 QuickDraw 图像，所以原始训练集中共包含 6000 张图像。而在发布的测试集中，共包含了未添加标签的 45000 张图像。

首先，为了在训练和测试中完成对图像的归一化，就要先获得数据集的均值和方差，附件中代码 mean_std.py 实现了该功能。最终得到原始训练集的均值为 0.164436，标准差为 0.323920，而测试集的均值为 0.164597，标准差为 0.324155。

然后，因为要观察训练过程中的训练效果，所以需要建立验证集。我对原始

的训练集进行划分，每一个图像类别中随机选取了 30%作为验证集，剩下的 70%作为训练集，附件中代码 datasplit.py 实现了该功能。最终完成了对原始数据集的划分，划分后得到的训练集共包含 4200 张图像，每个类别包含 280 张图像，而验证集共包含 1800 张图像，每个类别包含 120 张图像。

3.1.2、扩充数据集

本次的课程设计中还使用了同学自己建立的数据集，用来提高训练模型的泛化能力。在发布的额外数据集中，每个图像类别都包含了 300 张左右的图像，但是通过观察发现，有些图像并不符合训练集的格式要求，比如有的图像的大小不是 28*28，有的图像是白底黑字，这些图像都是噪声图像，要想性能得到提升，首先要对这个额外数据集进行 data clean 操作。

在 data clean 操作中，我首先将图像格式全部转为 PNG 格式，并将大小不是 28*28 或者 28*28*3 的图像全部剔除。接着用之前 CNN 在原始训练集中训练获得的模型对剩下的图像进行测试，每张图像都获得一个 loss 值，我舍弃了 1oss 最大的 20%的图像，将其视为噪声图像。然后将剩下的图像与原始数据集进行合并，得到扩充后的数据集，附件中代码 dataclean.py 实现了该功能。

同样，需要对扩充后的数据集计算其均值和标准差，使用 mean_std.py 计算得到扩充后的数据集的均值为 0.154208，标准差为 0.308720。然后将扩充后的数据集按照 70%-30%的比例进行划分，得到训练集和验证集，训练集中每个图像类别包含约 400 张图像，验证集中每个图像类别包含 180 张左右的图像。

3.1.1、数据增强后的数据集

由于训练集中一共只包含 6000 张左右的图像，而测试集中就包含了 45000 张图像，训练集过小，使用神经网络进行训练很容易出现过拟合的现象，从在原始数据集和扩充后的数据集上的训练效果可以看出，即便加入了 Dropout 等操作也难免出现过拟合。于是需要进一步扩大训练集的大小。

在本次课程设计中，我对使用 extra data 扩充后的数据集进行了数据增强，对数据集中的每张图片进行平移、旋转、翻转等操作，将数据集的大小扩大了 15 倍。对数据集中每张图片进行如下 15 种变换，可以获得数据增强后的数据集：

①无变化。

②向下移动 3 个像素。

③向右移动 2 个像素。

④向右移动 1 个像素，再向下移动 1 个像素。

⑤顺时针旋转 12°。

⑥逆时针旋转 12°。

⑦左右翻转。

⑧向上平移 1 个像素。
⑨向左平移 1 个像素。
⑩向右平移 1 个像素，再顺时针旋转 8°。
⑪向下平移 1 个像素，再逆时针旋转 8°。
⑫旋转 8°，再左右翻转。
⑬向左平移 1 个像素，再向下平移 1 个像素。
⑭向右平移 1 个像素，再向上平移 1 个像素，再顺时针旋转 6°。
⑮向左平移 1 个像素，再向上平移 1 个像素，再左右翻转。

数据增强之后的数据集变得十分庞大，仍需要计算其均值和标准差，利用 mean_std.py 代码计算得到数据增强后数据集的均值为 0.153998，标准差为 0.308592。同样按照 70%训练集，30%验证集的比例进行划分，划分后训练集每个图像类别大约包含了 6000 张图像，验证集中每个图像类别大约包含了 2700 张图像，即将之前扩充的数据集又扩大了 15 倍。

3.2、模型在原始数据集上的性能分析

对原始数据集的训练，我采用了 SVM、CNN 和 ResNet18 三种模型。

SVM 的参数设置中，采用的是 RBF 核函数，gamma 值设为 0.001，惩罚参数 C 设为 100。对原始数据训练后，再对测试集进行预测，得到预测分数为 0.69681。

CNN 的参数设置中，构建了 3 个卷积层的 CNN 模型，第 1 个卷积层滤波器大小为 3*3，步长为 1，padding 设为 1，滤波器个数为 16，第 2 个卷积层的滤波器个数为 32，其余参数和第一个一样，第 3 个滤波器的滤波器个数是 64，其余参数也和第一个一样。在每个卷积层后都加了一个窗口大小为 2，步长为 2 的最大池化层进行特征降维。并加入了 dropout，设为 0.5，然后通过两个全连接层和 softmax 层得到图像属于每个类别的概率。训练过程中，batch size 大小设为 16，初始的 learning rate 设为 0.0005，每训练 10 个 epoch 进行一次调整，将其降为原来的 0.8 倍。一共训练了 150 个 epoch，训练过程中保存在验证集上表现最好的 epoch 模型，然后用该模型对测试集进行预测，得到预测分数为 0.81355。

ResNet18 参数设置中，采用的是标准的 ResNet18 模型，模型的输出是 15 维概率向量。Dropout 设为 0.5，初始 learning rate 设为 0.0005，同样每训练 10 个 epoch 进行将其降为原来的 0.8 倍。在原始数据集上一共训练了 50 个 epoch，保存验证集表现最好的 epoch，预测得分为 0.82022。

表 3-1 展现了 SVM、CNN、ResNet18 三种模型在原始数据集上训练，在测试集上的预测得分(public、private)，可见 SVM 的分类效果明显不如其他两个模型，而 ResNet18 的表现要稍好于 CNN 模型。虽然 ResNet18 的得分已经由于给出的 baseline 得分(0.79228)，但是从训练过程的输出可看出，训练集的准确率很

快就达到了 99%以上，而在测试集上的表现仅有 82%左右，这是训练集过小而出现了过拟合的现象，于是需要扩大训练集来进一步提高模型的泛化能力。

表 3-1 模型在原始数据集上训练后的预测得分

模型	SVM	CNN	ResNet18
预测得分(public)	0.69681	0.81355	**0.82022**
预测得分(private)	0.68653	0.81428	**0.82047**

3.3、模型在扩充数据集上的性能分析

加入了同学自己建立的数据集后，训练集的数目变大了，并且有更加风格多异的训练数据，可以帮助模型得到更好的泛化性能。对扩充后的数据集训练，我仍采用了 SVM、CNN 和 ResNet18 三种模型。

SVM 的参数设置中，仍然采用的是 RBF 核函数，gamma 值设为 0.001，惩罚参数 C 设为 100。对原始数据训练后，再对测试集进行预测，得到预测分数为 0.69540。

CNN 的参数设置中，仍然使用上述相同的模型，参数设置也相同。训练了 150 个 epoch，保存在验证集上表现最好的 epoch 模型，最后对测试集进行预测的得分是 0.82251。

ResNet18 参数设置中，也和在原始数据集上训练的设置一样。在扩展的数据集上一共训练了 50 个 epoch，保存验证集表现最好的 epoch，预测得分为 0.84096。

表 3-2 展现了 SVM、CNN、ResNet18 三种模型在扩展后的数据集上训练，在测试集上的预测得分(public、private)。从表中可以看出 ResNet18 和 CNN 的预测结果仍然明显由于 SVM，并且 CNN 与 ResNet18 预测效果的差距进一步拉大。和表 3-1 中三种模型在原始数据集上训练得到的预测效果相比，发现 SVM 的预测效果略微下滑，可能是惩罚参数 C 设置的较大导致的。而用 CNN 和 ResNet18 训练得到的预测效果都得到了一定程度的提高，可见经过预处理的扩充数据集起到了积极的作用。但是在训练过程中，仍然出现了训练集的准确率早早达到 99%以上的现象，过拟合仍然存在，说明仍需要继续扩大训练集，增强模型的泛化能力。

表 3-2 模型在扩充数据集上训练后的预测得分

模型	SVM	CNN	ResNet18
预测得分(public)	0.69540	0.82251	**0.84096**
预测得分(private)	0.68546	0.82533	**0.84409**

3.4、模型在数据增强后数据集上的性能分析

为了获得更加庞大的训练集，我采用了数据增强的方法，对加入 extra data 的数据集中的每张图像进行平移、旋转、翻转等一系列操作，将数据集的大小扩大了 15 倍，这样就有足够的训练数据来使模型获得更好的泛化能力。由于之前 SVM 的表现一直不理想，于是对于数据增强后的数据集训练，我采用了 CNN、ResNet18、ResNet101、VGG16 四种模型。

CNN 的参数设置中，仍然使用之前相同的模型，参数设置也相同。训练了 150 个 epoch，保存在验证集上表现最好的 epoch 模型，最后对测试集进行预测的得分是 0.87288。

ResNet18 参数设置中，也和在原始数据集上训练的设置一样。由于数据量的扩大，我在数据增强后的数据集上一共训练了 150 个 epoch，保存验证集表现最好的 epoch，预测得分为 0.90481。、

之后我又使用了 ResNet101 进行训练，batch size 设为 64，其余训练参数与 ResNet18 设置的一样，最后的预测得分为 0.90755。

最后，我还使用了 VGG16 对数据增强后的数据集进行训练，同样使用的是逐渐降低的 learning rate，batch size 设为 64，训练了 150 个 epoch 后得到的预测得分为 0.90081。

表 3-3 展现了 CNN、ResNet18、ResNet101、VGG16 四种模型在数据增强后的数据集上训练，在测试集上的预测得分(public、private)。从表中可以看出 CNN 的预测结果不如其余三个模型，而 ResNet18、ResNet101 和 VGG16 模型的预测得分接近，其中 ResNet101 模型表现最佳。和表 3-2 中模型在扩充后的数据集上训练得到的预测效果相比，发现 CNN 和 ResNet18 训练得到的预测得分都有了显著的提高，可见数据增强对于提高模型的泛化能力起到了很大的作用。其中 ResNet101 的得分已经达到了 0.90755，并且 ResNet18 和 VGG16 都获得相接近的效果，进一步使用数据增强的方法扩大训练集可能只有微乎其微的提高。于是想要进一步提高预测得分只能寻找其他的技巧。

表 3-3 模型在数据增强后的数据集上训练后的预测得分

模型	CNN	ResNet18	ResNet101	VGG16
预测得分(public)	0.87288	0.90481	**0.90755**	0.90081
预测得分(private)	0.87511	0.90803	**0.91425**	0.90088

3.5、投票机制

Voting ensembles 这种投票机制经常用于图像分类中，用来提高模型的分类

效果。从之前的实验中，已经分别获得了 SVM、CNN、ResNet18 在原始数据集与扩充数据集上训练然后进行预测的结果，以及 CNN、ResNet18、ResNet101 与 VGG16 在数据增强后的数据集上训练然后进行预测的结果，共有 10 个已有预测结果。投票机制就是对多个已有的结果进行投票，对于每一张测试图片，将其归类到投票得分最高的那一类中。由于每一种模型在不同训练集上训练后得到的预测结果不同，对多个模型进行投票有可能结果得到提升，也有可能得到下滑，这些预测结果的相关性越弱，就可以和得到越不一样的投票结果。

在本次课程设计中，我分别对在数据增强后的数据集上训练的 4 个模型进行投票，以及对除了 SVM 之外的其余 8 个已有预测结果进行投票，将测试图像归类于投票结果较高的那一类，如果有投票得分相同的情况，则按照 ResNet101 的预测结果进行分类，预测得分如表 3-4 所示。从表中结果可以看出，基于在数据增强后数据集上训练得到的预测结果进行投票，得到的结果有明显的提升，而基于 8 个预测结果的投票，结果却略微下滑。其原因可能是基于 4 个预测结果的投票，这 4 个结果是用不同模型在同一个训练集上训练得到的结果，且这 4 个模型原本的预测性能也相近，而使用的 8 个预测结果彼此之间性能却相差较大，可能会影响原本性能较好模型，导致分类错误。

于是我将基于 4 个预测结果投票后的结果作为最后的预测结果进行提交，在任务截止后，在 private 预测得分上获得了 0.91977 的高分，位列 Leaderboard 的第 4 名。

表 3-4 使用投票机制的预测得分

模型	ResNet101	Vote based 4 models	Vote based 8 models
预测得分(public)	0.90755	**0.91651**	0.90762
预测得分(private)	0.91425	**0.91977**	0.91000

四、结论

在本次的课程设计当中，我先使用 SVM、CNN 和 ResNet18 对原始数据进行了训练，发现训练集过小，出现了过拟合的现象。然后使用预处理后的自建数据对原始数据集进行扩充，并仍使用 SVM、CNN 和 ResNet18 对扩充后的数据进行训练，发现效果得到了提升，但是仍存在过拟合现象。于是采用数据增强的方法进一步扩大数据集，将数据集扩大了 15 倍，并分别使用 CNN、ResNet18、ResNet101 和 VGG16 对数据进行训练，发现预测结果得到了显著提升，其中 ResNet101 的预测结果最好，public 预测得分为 0.90755，private 预测得分为 0.91425。最后，我还使用了投票机制，对数据增强后的 4 个预测结果进行了投

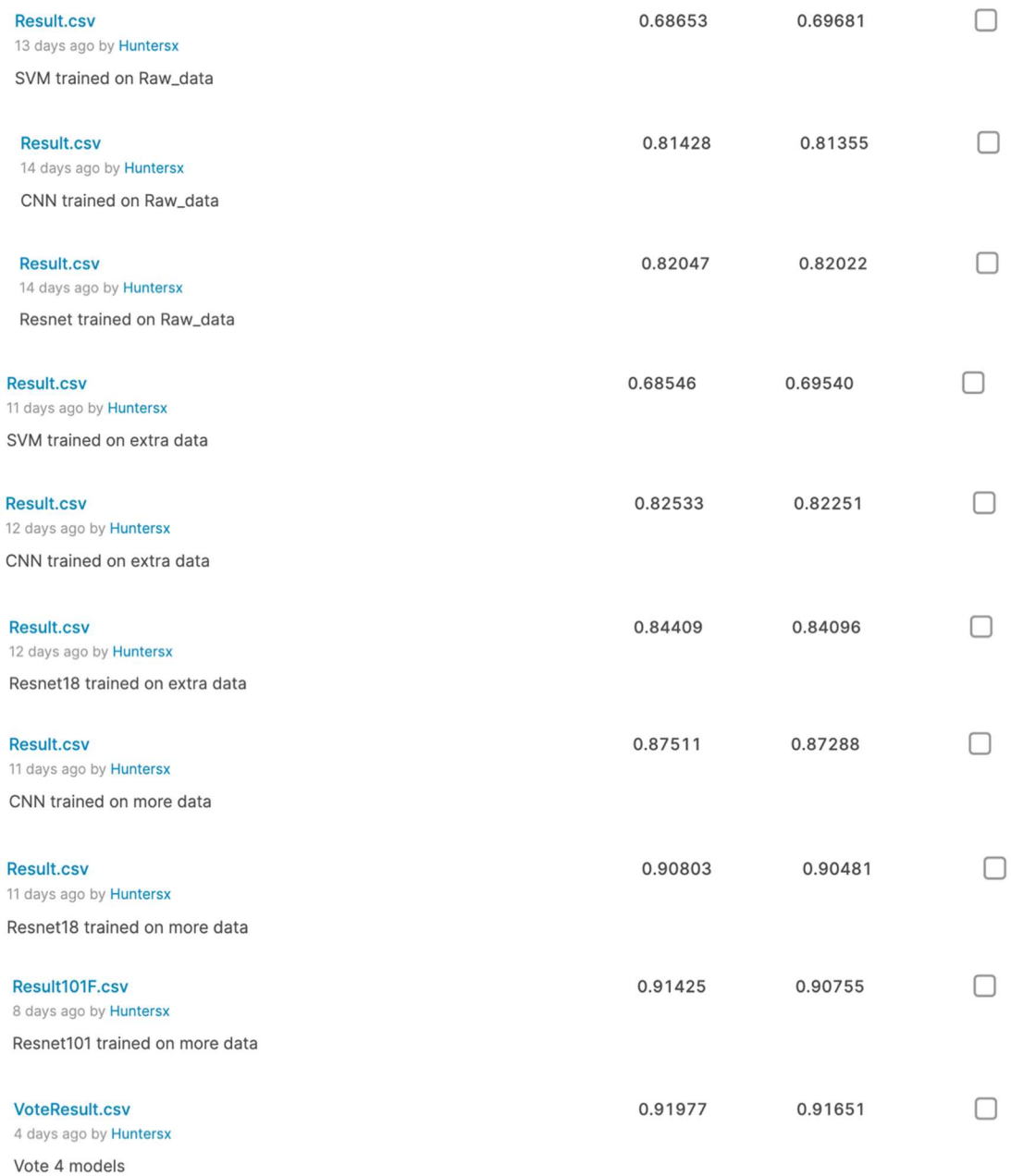
票，进一步提升了预测效果，最终的 public 预测得分为 0.91651，private 的预测得分为 0.91977，在 Leaderboard 上排名第 4。

五、参考文献

[1] 周志华. 机器学习[M]. 北京：清华大学出版社, 2016: 121-132.

六、附录

附提交记录截图：



VoteResult2.csv 3 days ago by Huntersx Vote 8 models	0.91000	0.90762	☐