机器学习工程师纳米学位毕业项目
基于深度学习的街景数字识别

梁瀚明

2016 年 11 月 21 日

目　录

1 定义

1.1 项目概述

项目将训练一个从自然图像中识别数字序列的模型，模型可以应用到手机APP中，实现实时显示出数字序列。

Google和百度等地图制造商通过街景小车采集了许多街景图像。识别街景图片中的门牌号有助于高精度地定位建筑的位置。识别图像中的的数字序列也是光学字符识别(Optical character recognition)中的一个重要问题。尽管文本OCR已经被广泛研究，任意长度的多数字识别仍然是一个很有挑战的问题。Goodfellow et al. (2014)

项目使用的模型是卷积神经网络(Convolutional Neural Network)。卷积神经网络早在1997年便被用来解决字符识别问题，然而它最近地广泛应用源于2012年ImageNet图片分类竞赛中，CNN以最高的分类准确率夺得头筹。Dumoulin and Visin (2016)

项目选择的数据集是街景房屋编号数据集(Street View House Numbers Dataset, SVHN)。SVHN是从真实世界街景采集的数据集，包含10个类别，对应于10个数字。数据分为两种格式。第一种格式，每一个数字被切分出来单独作为一张图片。第二种格式，每一张图片对应一个完整地房屋编号，数据集提供了其中每一个数字的位置。识别单个数字要比识别多个数字组成的数字序列要简单，因此项目将分为两个部分，先从简单的问题，识别一个数字开始，然后再来解决识别数字序列的问题。Netzer et al. (2010)

1.2 问题陈述

项目选择的数据集是从真实世界采集的街景数据。一个挑战在于真实世界数字的颜色、字体、分辨率和朝向多种多样，同时图像采集过程中的阴影、光照条件，由于运动和失焦带来的模糊都会增加识别的难度。

问题分为两个部分。第一个部分是识别图像中的单个数字，第二个部分是识别图像中的数字序列。

识别图像中的单个数字是一个多类的分类问题。输入是由像素构成图像，目标将图像分为0至9十个数字中的一类。

数字序列的识别目标是识别任意多个数字组成的序列，$s = s_1, s_2, \cdots, s_n$。

街景数字识别的一个特殊性在于，数字序列的长度有一个上界。非常少的街牌号会长于5个数字，于是可以假设数字序列的长度最多只有N，其中$N = 5$。模型做出这个假设，应当能够确定什么时候假设不成立，而拒绝返回结果。这样极少数长度大于N的街牌号就不会错误地添加到数据库中。

1.3 评价指标

对于单个数字的识别，评价指标是分类的准确率(Accuracy)，即正确分类的图片数

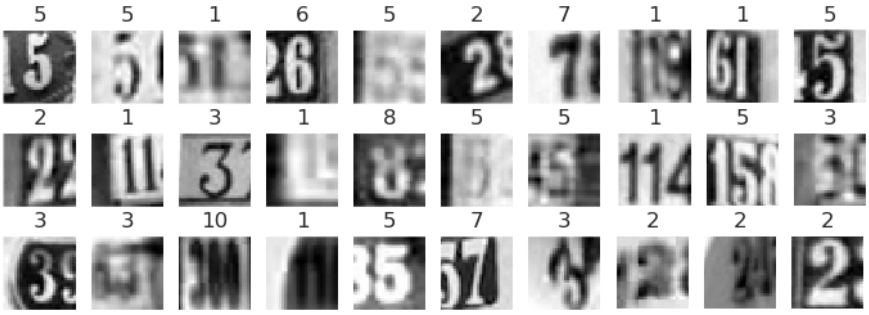

图 1: SVHN数据集的第一种格式。数据集包含32x32像素的图像以及对应的标签。标签为1-10，其中10代表数字0，标签的数字位于图片中央。

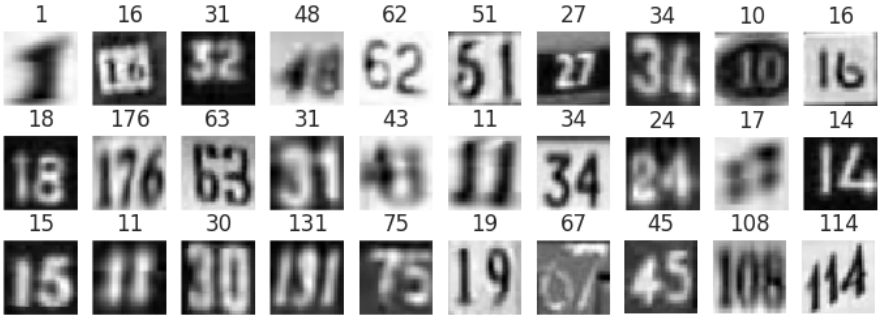

图 2: SVHN数据集的第二种格式。数据集图像大小不一，包含数字所在位置以及标签。预处理的将图片剪裁到恰好完整包含全部数字，并缩放为40x40像素，结果如图所示。图片上侧为数字序列的标签。

除以图片总数。

对于数字序列的识别，数字序列完全被识别正确才被认为识别正确，部分数字识别正确不算正确。这是因为识别街景图片中的数字，目的在于正确地确定建筑的位置，其中有一个数字错误，识别出的编号便没有任何价值。

2 分析

2.1 数据可视化

SVHN数据集分为三个子集，训练数据集、测试数据集以及额外数据集。额外数据集包含大量容易的样本，训练数据集相对更小，包含更难的样本。

SVHN数据集包含两种格式。第一种格式图像被切分为一个个数字，标签所指的数字位于图片的中央，如图1所示。图像中的数字可能多于1个。图像中0的标签为10。观察图片可以看到，数字的字体、大小和亮度差异很大，同时数字的朝向也略有不同。

第二种格式包含大小不一的真实场景图像，除了数字以外还包含其它背景。在预处理的过程中，会将图片剪裁到恰好包含完整数字序列，同时将图片缩放到40x40像素。

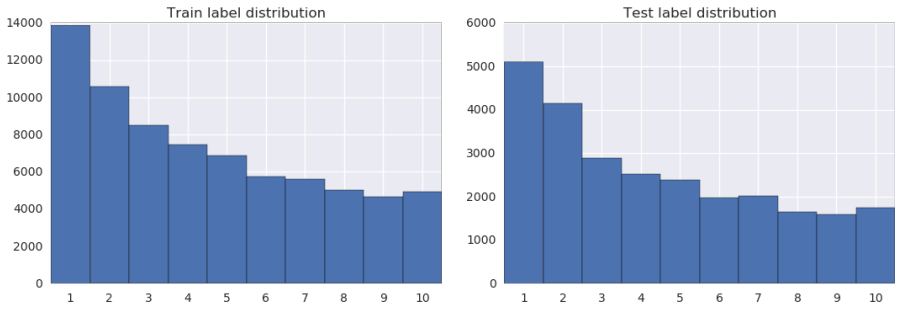

图 3: 训练数据集和测试数据集数字的分布。数字并非均匀分布，数字1出现的次数几乎是数字0出现次数的3倍，但是测试集与训练集的分布几乎保持相同。

观察图片可以看到，部分图片经过缩放后，变得十分模糊，比如图2中标签为17的图片，即使是人眼也难以分辨。

训练数据集和测试数据集数字的分布如图3所示，数字的分布并非均匀分布。数据集并没有说明数字的分布是否符合真实世界的分布，但是由于测试数据集数字的分布与训练数据集的分布相同，并不需要更改数据的分布。

2.2 算法和技术

2.2.1 分类算法

根据前面的分析可知，数字的识别本质上是一个分类的问题。用于分类的算法有很多，比如感知机、逻辑回归、支持向量机、决策树和人工神经网络等。这些算法统称为机器学习算法。算法的输入是由图片每一个像素所组成的特征向量，输出是数字的类别。输入图片是一个特征空间，像素的个数是空间的维度。

感知机和逻辑回归算法的目标是直接在特征空间内求解多个超平面使得错误分类的样本尽可能少。然而对于图片数据，不同类别的数字并不能够通过线性的分类平面分割开来。

支持向量机同样是在特征空间内求解分类超平面，但是希望分类超平面离位于分类边界附近样本距离尽可能远，使得分类超平面具有一定稳健性。但是这仍然不能解决数据本身线性不可分的问题。支持向量机支持核技巧。核技巧隐式地把数据映射到高维，然后在高维求解分类超平面，这等价于在原有空间求解一个分类的曲面。支持向量机加上核技巧能够解决许多非线性问题。然而这种映射是固定的，不能根据数据的特性灵活地改变。

决策树将原有特征空间分割成为多个区域，尽可能地让每一个区域的样本属于同一类。然而随着维数的增加，所需要分割的区域也会随之呈指数的增加，这一现象称为维数灾难(curse of dimentionality)。一般输入图片数据有1000个像素，即1000维。如果每1维分割成2个区域的话，所需空间的个数就会比宇宙的原子数还要多。因此，决策树对于高维的数据效果不好。

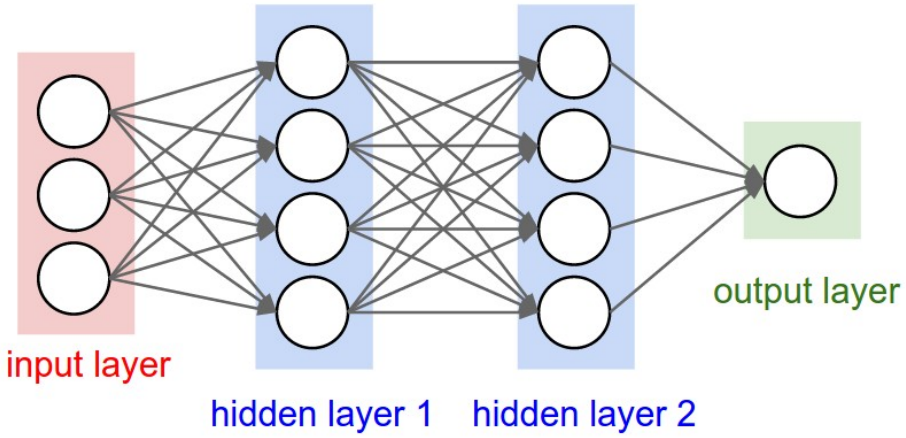

图 4: 前向神经网络的结构，包含输入层、多层中隐藏层和输出层。

2.2.2 神经网络

人工神经网络类似于大脑中的神经网络。大脑中的神经网络由一个个的神经元组成，神经元从多个树突中获取输入信号，然后通过轴突输出；人工神经网络同样由一个个基本单元组成，单元也被称为神经元。每一个单元从其它单元获取输入的值，通过某种计算得到输出，作为下一个神经元的输入。

每一个神经元是一个简单的非线性函数。输入首先加权求和，在数学上这是一个输入与权重的点积操作。然后通过Sigmoid函数或者整流线性单元(ReLU)得到输出。加权求和时的权重是神经元的参数。神经网络中的每一个神经元具有相同的结构，不同的是输入的权重。

为了逼近图像到数字的映射关系，神经网络通过后向传播算法更新权重。对于分类的问题，可以使用神经网络预测的结果和图像的真实类别计算出损失函数交叉熵(cross entropy)。交叉熵对于每一个权重是可导的，因此可以通过梯度下降法更新权重的值。神经网络可以有数百万的权重，后向传播算法则提供了一个高效计算误差关于权重梯度的算法。

神经网络根据神经元连接方式的不同分为两种，前向神经网络(feed-forward neural network)和递归神经网络(reccurent neural network)。项目中使用的卷积神经网络具体来说时一种前向神经网络。前向神经网络的神经元组成一种分层的结构，前一层的神经元只与后一层的神经元相连，每一层之中的神经元没有连接。神经网络的层数又叫做深度，多层的神经网络就是深度学习名字的来源。

前向神经网络可以分为输入层、多层隐藏层和输出层，如图4所示。第一层隐藏层是输入数据的非线性映射，进而后一隐藏层也是前一层的非线性映射。这种映射不同于支持向量机，它是由数据决定的。模型通过权重更新学习得到了一种易于分类的映射。

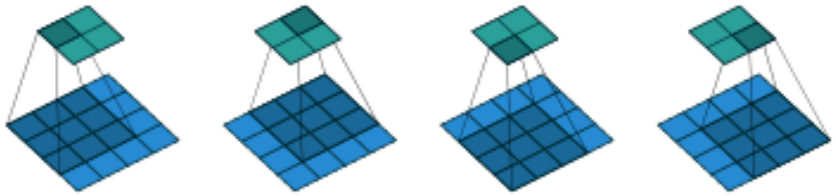

图 5: 卷积神经网络的核心组成部分——卷积。4x4的输入图像，经过3x3的卷积操作得到了2x2的特征图。Dumoulin and Visin (2016)

2.2.3 卷积神经网络

Hubel和Wiesel于19世纪50年代和60年代发现，猫和猴子位于视觉皮质的神经元只对一小块视觉区域响应。一个神经元响应的视觉区域称为感受野(Receptive field)。相邻的神经元具有相似的感受野，同时相邻神经元的感受野相互重叠。他们的研究启发了卷积神经网络(Convolutional Neural Network)。Wikipedia (2016)

普通的前向神经网络，后一层每一个神经元的输入是前一层所有神经元的输出。而卷积神经网络后一层每一个神经元的输入只与前一层局部的输出有关。这一局部区域也被称为感受野，通常大小为3x3px或者5x5px。因此卷积神经网络的前几层只响应于局部的输入，后续的隐藏层的输出逐步变得具有全局性。这等价于约束普通神经网络的一些权重必须为零。

卷积神经网络约束同一层的神经元具有相同的权重。这意味着每一层提取的是相同的特征，这一组相同的特征组成特征图(feature map)。这样不论特征出现的位置，这一层神经元都能够检测到，这称为平移不变性(translation invariance)。一般一层神经元具有多组权重，即同时检测多组特征。

局部连接与权重共享大大减少了模型的参数个数，这意味着使用神经网络进行预测时的内存占用也会减少。内存占用的减少使得计算机能够训练层数更多、每一层神经元数目更多的神经网络。

在数学上，这种局部连接同时权重共享的点积操作叫做卷积(convolution)，这也是卷积神经网络名字的来源。在图像处理领域，这组权重称为图像滤波器或者卷积核(kernel)。在卷积神经网络中，这种隐藏层称为卷积层。卷积操作如图5所示，下侧蓝色图像代表卷积层的输入，上侧绿色图像代表卷积层的输出。图中的阴影代表一个卷积操作输入输出的对应关系，图中感受野的大小为3x3。相邻两个感受野之间的距离称为跳跃距离(hop size)，一般为1。图中4x4的输入图像，经过3x3的卷积操作得到了2x2的特征图。

2.2.4 技术

项目中将会使用Google开源的深度学习框架TensorFlow Abadi et al. (2015)。TensorFlow支持Python和C++的接口，既可以用于学术研究，也可以用于生产环境。许多Google的

内部服务，就使用了TensorFlow，比如Gmail、语音识别等。

TensorFlow会将神经网络的计算流程表示成为一个计算图。计算图的计算会调用底层的用C++编写的函数库，因此同样的运算会比直接在Python中计算效率更高。

有了计算图，TensorFlow还能够自动计算出误差函数关于每一个权重的梯度。这是因为神经网络由多层简单的数学运算组合而成，这些数学运算通常是固定的，通过链式法则就能够计算出梯度。

同时TensorFlow还能够利用多核CPU和GPU的优势，Google甚至为TensorFlow设计了专用芯片TPU(Tensor Processing Unit)，速度要比GPU还快。

此外TensorFlow还拥有图形化调试工具，能够展示在训练过程中，神经网络权重的变化趋势以及误差的变化趋势。

2.3 基准指标

对于单个数字的识别，Sermanet et al. (2012)使用4层的卷积神经网络，多级的特征实现了94.85%的测试集分类正确率。Goodfellow et al. (2014)使用包含11层隐藏层的卷积神经网络实现了97.84%的测试集分类正确率。其实验同时显示神经网络的层数越多，分类器的表现越好，但是层数的增多使得训练花费的时间大大增加。训练这个神经网络，使用10台高性能计算机，花费6天时间才完成。

对于数字序列的识别，Goodfellow et al. (2014)实现了96.03%的测试集正确率。同时实验显示使用阈值只对部分数据进行预测，能在覆盖95.64%的数据上，实现98%的测试集正确率，已经能够与人的表现相媲美。

考虑到神经网络的训练需要高性能的计算机，同时需要耗费很长的时间，而目前的实验条件难以达到论文中的条件。根据客观条件，项目设定的目标为识别准确率在世界最佳模型10%以内。也就是说，单个数字的识别需要达到准确率87.84%以上，数字序列识别准确率需要达到86.03%以上。

3 具体方法

3.1 数据预处理

3.1.1 单个数字识别

考虑到颜色对于数字的识别所起的作用不大，图像首先会从RGB真彩色转换成灰度图。然后将图片进行归一化，即让图片的均值为0，方差为1。

训练集经过随机打乱以后，一部分会被用于交叉验证，即在训练的过程中衡量模型分类性的性能。

对于单个数字的识别，全部的训练集(72457个样本)和100000个额外集的样本混合成新的数据集，从中选择5%的样本作为交叉验证集。最终训练集的大小为164594，交叉验证集的大小为8663，测试集的大小为26032。

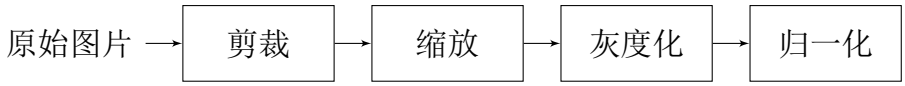

图 6: 图片的预处理步骤。

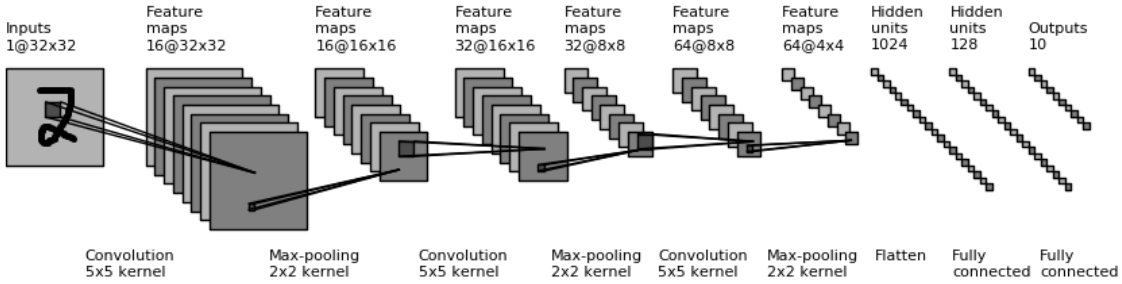

图 7: 识别单个数字卷积神经网络结构图。神经网络包含三层卷积层，两层全连接层，每一层卷积层之后都通过最大值池化。每一层的上侧记录的是特征图(feature map)的个数以及特征图的大小，两层连接处的下侧记录着所经过的操作。

3.1.2 数字序列识别

数字序列的识别的预处理步骤如图6所示。由于数据集给出的是包含每一个数字的矩形边框，需要计算出恰好完整包含数字序列的矩形边框。然后利用计算出的矩形边框将图片剪裁到仅包含数字序列，同时缩放到40x40像素。最后需要对图片灰度化和亮度值归一化。

对于多个数字的识别，每一张图片中，数字序列的长度不同，通常长度在1到5之间。因此需要将图像的标签转换成同一长度，不存在的数字用标签10表示。

全部训练集中的33402张图片以及额外集中的202353张图片将组成新的训练集，从中选择5%的样本作为交叉验证集。最终训练集的大小为223967，交叉验证集的大小为11788，测试集的大小为13068。

3.2 实现

3.2.1 单个数字识别

识别单个数字的卷积神经网络结构包含3层卷积层，1层全连接层和1层输出层，结构图如图7所示。所有的连接均为前向，从一层到相邻的下一层。3层卷积层分别包含[16, 32, 64]张特征图。全连接层包含128个单元。每一层卷积层的输出通过最大值池化，池化窗的大小为2x2，步长为2。训练时，全连接层的输出以0.5的概率随机将值赋为0。所有卷积核的大小为5x5。

使用小批量(Mini-batch)梯度下降计算误差关于权重的梯度，每一批的大小为64个样本。

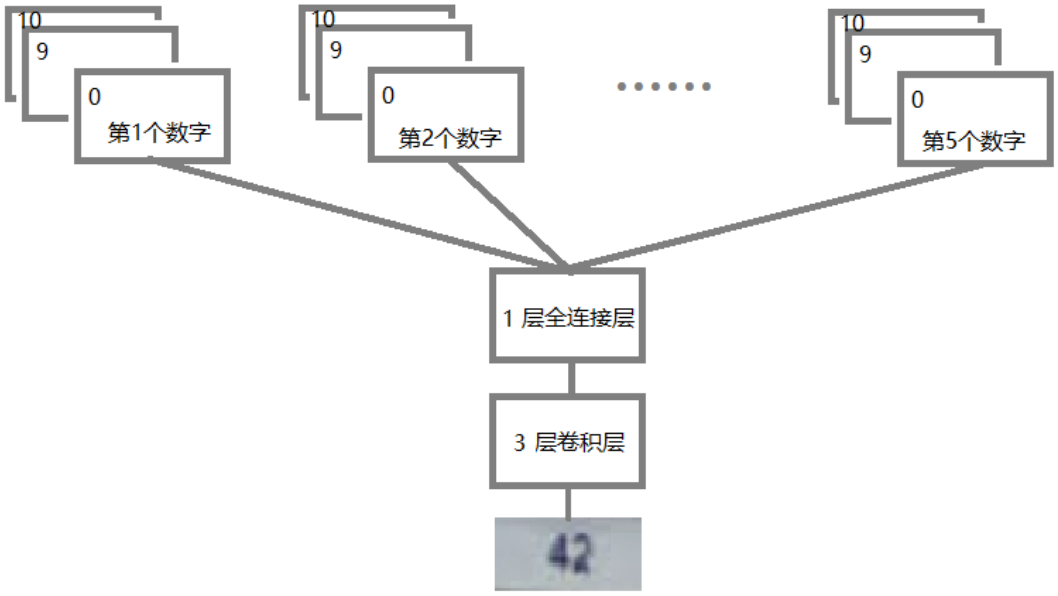

图 8: 识别数字序列卷积神经网络结构图。输入图片通过多层卷积层，然后通过多层全连接层，最后独立地预测5个数字的结果。对于每一个数字包含11个类别，0-9代表数字的类别，10代表数字不存在。

3.2.2 数字序列识别

对于数字序列识别的问题，图片中数字的个数不一，最少有1个数字，最多有5个数字。每一个数字再不同图片中出现的位置也有很大的差异，如果图片中只有一个数字，那么第一个数字一定再图片的中央，如果有4个数字，第一个数字就被挤到了图片的左侧。

传统的方法是首先预测每一个数字所在的位置，将图片剪裁成一个个数字，然后再预测数字的类别。Goodfellow et al. (2014)提出了一种不需要将图片剪裁成一个个数字的方法，但仍然需要预测包含整个数字序列矩形框的位置。神经网络会独立的预测6个输出，其中5个对应于5个数字的类别，第6个输出对应于数字序列的长度。数字序列的长度是一个抽象的概念，论文展示了神经网络能够正确地预测这一输出。论文提出的这一方法在数字数列中的成功启发了本项目中使用的方法。

项目中用于识别数字序列的神经网络，拥有5个输出，对应于5个数字。为了解决不同图片中数字序列长度不同的问题，每一个输出包含11个类别，类别0-9对应于数字的类型，类别10代表该数字不存在。这意味着无形中神经网络需要学会一个约束关系，预测的数字序列如果一个数字不存在，那么其后的数字也必须不存在。

数字序列识别神经网络的结构与单个数字识别神经网络的结构类似，不同之处在于全连接层与5个输出层相连，对应于5个预测的数字。

识别数字序列的卷积神经网络结构包含3层卷积层，1层全连接层和一层输出层，结构图如图8所示。所有的连接均为前向，从一层到相邻的下一层。3层卷积层分别包

方案	小批量准确率	交叉验证集准确率	测试集准确率
初始	100.0%	69.69%	68.43%
正则化($\lambda = 10^{-3}$)	100.0%	71.93%	70.02%
正则化($\lambda = 5 \times 10^{-3}$)	100.0%	74.45%	72.67%
正则化($\lambda = 10^{-2}$)	100.0%	74.81%	73.89%
正则化($\lambda = 3 \times 10^{-2}$)	89.06%	68.82%	68.41%
随机舍弃(p=0.7)	82.81%	75.04%	74.54%
随机舍弃(p=0.6)	81.25%	72.77%	75.10%
随机舍弃(p=0.5)	73.44%	74.09%	74.51%
额外数据集(170000)	85.94%	80.06%	77.95%
额外数据集(202353)	90.62%	83.24%	80.19%
早期停止	100.0%	70.50%	68.54%
学习率下降(0.95)	98.44%	69.12%	68.94%
学习率下降(0.90)	84.38%	69.12%	68.27%
增加图片尺寸(40px)	100.0%	68.58%	66.15%

表 1: 不同改进对于分类准确率的影响。每一种情况只考虑一种改进。初始方案在测试集上的分类准确率为68.43%。改进最大的方案为使用全部的额外数据集，测试集分类准确率为80.19%。其次是随机舍弃，测试集分类准确率为75.10%

含[24, 48, 80]张特征图。全连接层包含256个单元。每一层卷积层的输出通过最大值池化，池化窗的大小为2x2，步长为2。训练时，全连接层的输出以0.6的概率随机将值赋为0。所有卷积核的大小为5x5。

3.3 改进

由于单个数字的识别和数字序列识别的改进过程类似，因此下面主要讨论数字序列识别的改进过程。

没有经过任何改进时，神经网络展明显地现出过拟合的特征。模型的训练集准确率高达100%，而在测试集上的准确率仅为68.43%。从图9中学习曲线上也可以看出同样的问题。在训练后期，训练集准确率仍然保持增长，交叉验证集准确率却不再增加。训练集准确率与交叉验证集准确率的差距逐步增大。

表1中考虑率了常见的改进方案，并计算了仅采取一种改进时，训练集分类准确率、交叉验证集分类准确率和测试集分类准确率。参数正则化、随机舍弃和使用额外数据集均是为了避免过拟合。学习率下降是从优化的角度出发，为了寻找更精确的极小值点。增加图片的尺寸则是我通过观察模型预测的结果，对模型可能存在问题的一个猜想。

以测试集准确率为指标，使用额外数据集改进最大，其次是随机舍弃，再次是对参

图 9: 没有经过任何改进时，数字序列识别的学习曲线。在训练后期，训练集准确率保持增长，交叉验证集准确率却保持不变。训练集准确率与交叉验证集准确率的差距逐步增大。

数正则化。早期停止、学习率下降并没有明显的改进。增加图片的尺寸反而恶化率测试集准确率。

同一种改进方案，同时尝试了多种不同的参数，可以作为模型参数选择的依据。只使用部分额外数据集，测试集分类准确率为77.95%，使用全部额外数据集，分类准确率提高到了80.19%。因此在下一步分析增量改进对于模型的影响时，使用全部额外数据集。同样的原因，选择正则化的参数为10^{-2}，随机舍弃的概率为0.6。

表2展示了增量改进对分类准确率的影响。依次采取5种改进方案，分别是正则化、随机取舍、使用额外数据集、早期停止和学习率下降。下一种改进是在上一种改进的基础上作出的。可以看到不同的避免过拟合的方法是互补的，同时采取正则化和随机取舍，测试集准确率超越了两种单独改进。采取使用额外数据集的改进时，选择了3种不同的学习率。结果显示，当学习率为0.04时，测试集准确率最高。因此采取下一种改进时，固定学习率为0.04。早期停止在单步改进时没有体现明显的作用，但在采取了过拟合的措施以后，测试集准确率有了明显的提升。采用学习率下降，交叉验证集和测试集准确率均没有提升。因此最终的模型不采用学习率下降。

下面是所采取改进的详细说明。

3.3.1 正则化及随机舍弃

没有任何改进时，模型明显地展现出过拟合的特征。数字序列识别的学习曲线如图9所示，在训练后期，训练集准确率保持增长，交叉验证集准确率却保持不变。训练

方案	小批量准确率	交叉验证集准确率	测试集准确率
初始	100.0%	69.69%	68.43%
正则化($\lambda = 10^{-2}$)	100.0%	74.81%	73.89%
随机取舍(p=0.6)	85.94%	76.54%	78.49%
额外数据集($\alpha = 0.02$)	79.69%	73.79%	75.34%
额外数据集($\alpha = 0.04$)	85.94%	83.94%	82.08%
额外数据集($\alpha = 0.08$)	87.50%	71.39%	70.05%
早期停止	90.62%	90.41%	87.40%
学习率下降	92.16%	89.85%	87.11%

表 2: 增量改进对分类准确率的影响。依次采取5种改进方案，分别是正则化、随机取舍、使用额外数据集、早期停止和学习率下降。下一种改进是在上一种改进的基础上作出的。可以看到不同的避免过拟合的方法是互补的，同时采取正则化和随机取舍，测试集准确率超越了两种单独改进。早期停止在单步改进时没有体现明显的作用，但在采取了过拟合的措施以后，测试集准确率有了明显的提升。然而学习率的下降，仍然没有体现明显的改进。

集准确率与交叉验证集准确率的差距逐步增大。最终训练集分类准确率高达100%，交叉验证集分类准确率却只有69.96%。

神经网络参数众多，使得其能够表示复杂的输入输出非线性关系，也使得其易于过拟合于训练集中的噪声。一种常见的问题是，模型将图片中非数字的背景用于分类。测试集图片的背景与训练图片的背景并没有相同的规律，因此模型在测试集准确率上会显著低于训练集准确率。

正则化(regularization)在原始损失函数上加上模型参数的二范数。正则化的目的在于限制模型参数的模值。模型模值越大，模型越容易过拟合。因此限制参数的大小能够避免过拟合。在单步改进中，使用正则化，测试集分类准确率由68.43%提升到73.86%。

随机舍弃(random dropout)将全连接层的输出以一定的概率随机将输出赋值为0。使用随机舍弃等价于训练了很多个不同的模型，同时将这些模型组合了起来，共同确定最后输出的结果。在单步改进种，使用正则化，测试集分类准确率由68.43%提升到75.10%。

同时采取正则化和随机取舍，测试集准确率为78.49%，超越了两种单独改进。这说明这两种避免过拟合的方法时互补的。

3.3.2 使用额外数据集

SVHN数据集包含三个部分，训练集、测试集和额外集。根据数据提供者给出的说明，额外集比训练集识别数字的难度更稍低，然而并没有说明测试集图片的难度是否和训练集相同。对于数字序列识别，训练集包含33402张图片，额外集包含202353张图片。额外集的大小时训练集的6倍。

从表1和表2种可以看到，不论时单步改进还是增量改进，使用额外数据集均能显著提升测试集准确率。在单步改进时，只使用部分额外数据集，测试集准确率从68.43%提升到77.95%，使用全部额外数据集，测试集准确率提升到80.19%。在增量改进时，测试集准确率由78.49%提升到了82.08%。这说明，在可行的情况下，采集更多的数据是一种很好地提升模型性能的方法。

3.3.3 早期停止

训练神经网络的一个问题在于何时停止训练，使用早期停止(early stopping)可以解决这个问题。最初训练经过固定的10000步就结束，这时交叉验证集上的分类准确率已经没有显著提升，难以确定是否应该继续训练。增加训练步数到100000，一方面增加训练步数会显著增加训练时间，另一方面训增加练步数会增加模型过拟合的可能性。立即停止训练，可能会错过改进模型在测试集分类效果的机会。

早期停止每400步计算一次训练集和交叉验证集上的分类准确率。如果当前模型在交叉验证集上的分类准确率为最佳模型，则记录当前步数，并保存模型参数至文件。如果4000步内交叉验证集上的分类准确率没有提升，则停止训练。

对于数字序列的识别，最终模型经过60000次权重更新，交叉验证集上分类准确率在55600步时最佳。从表2中可以看到，在增量改进时，没有采用早期停止测试集准确率为82.08%，采用早期停止后，测试集准确率提升到87.40%。

4 结果

4.1 模型评价与验证

4.1.1 单个数字识别

在训练单个数字的识别模型中，使用的是Momentum方法，每一步从训练集中随机选取64张图片。训练的过程中使用了早期停止，最终模型经过26800次权重更新，交叉验证集上分类准确率在22400步时最佳。训练过程中训练集核交叉验证集的分类准确率如图10所示。交叉验证集准确率在前4000步内急剧上升，在后20000步内仍有小幅上升。在一个小区间内，训练集分类准确率的平均水平与交叉验证集的分类准确率相当，说明模型没有明显的过拟合。

对于单个数字的识别，在测试集上计算得到混淆矩阵(Confusion Matrix)如图11所示。混淆矩阵能够显示模型所犯的不同种类的错误。每一行左侧标签代表真实样本的类别，每一列下侧标签代表神经网络预测的类别，比如第2行第3列代表真实为2预测成为3，包含44个样本。混淆矩阵对角线上的单元代表正确预测的情况，可以看到绝大多数样本模型都能够成功识别出数字。预测错误最多的两种情况是，将5预测为3和将8预测为6，均有76个样本预测错误。

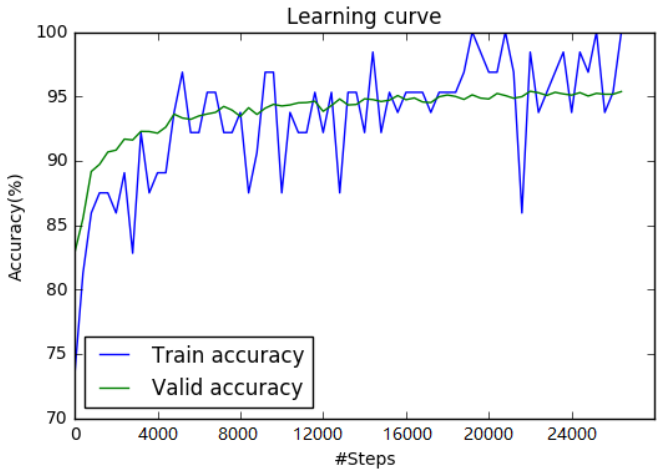

图 10: 单个数字识别在训练时的学习曲线。训练经过26800步。最终在交叉验证集上的分类准确率为97.9%，在测试集上的分类准确率为93.67%

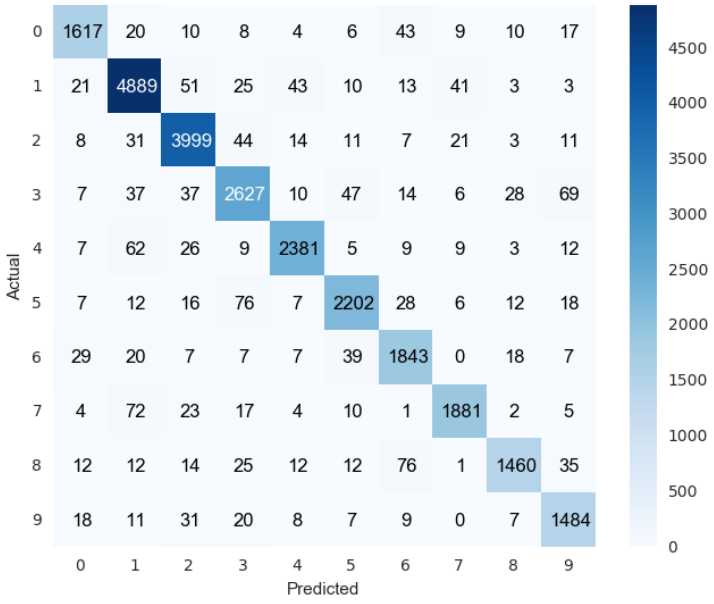

图 11: 单个数字识别在测试集上的混淆矩阵。每一行代表真实样本的类别，每一列代表神经网络预测的类别。预测错误最多的两种情况是，将5预测为3和将8预测为6，均有76个样本预测错误。

图 12: 数字序列识别在训练时的学习曲线。训练经过60000步。最终在交叉验证集上的分类准确率为90.41%，在测试集上的分类准确率为87.40%

Sermanet et al. (2012)使用4层的卷积神经网络，多级的特征实现了94.85%的测试集分类准确率。本项目使用的是4层的卷积神经网络，只有相邻两层存在连接，在测试集上的分类准确率为93.67%，相比世界最佳模型的识别准确率97.84%，差距位于10%以内。

4.1.2 数字序列识别

在训练数字序列的识别模型中，使用的是Adam梯度下降方法，每一步从训练集中随机选取64张图片。训练的过程中使用了早期停止，最终模型经过60000次权重更新，交叉验证集上分类准确率在55600步时最佳。对比单个数字模型训练的步数，训练数字序列模型的步数是单个数字模型步数的2倍。从一个方面反映出，识别数字序列比识别单个数字要更难。训练过程中训练集和交叉验证集的分类准确率如图12所示。交叉验证集准确率在前20000步内急剧上升，在后30000步内仍有小幅上升。在一个小区间内，训练集分类准确率的平均水平与交叉验证集的分类准确率相当，说明模型没有明显的过拟合。

Goodfellow et al. (2014)使用11层隐藏层的卷积神经网络，实现了96.03%的测试集分类准确率。本项目使用了3层隐藏层，在测试集上的分类准确率为87.40%。相比世界最佳模型的识别率96.03%，差距位于10%以内。

4.2 结果分析

4.2.1 单个数字识别

在训练完成得到最优的模型之后，我从测试集选择了30张错误分类的图片和30张正确分类的图片，将预测的结果绘制成图表，如图13和14所示，图片上方是神经网络预测的数字。

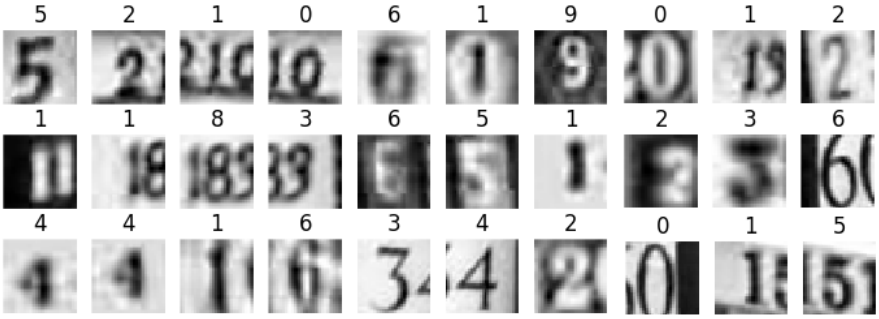

图 13: 对于数字序列识别，部分测试集中识别正确的样本。图片上方是神经网络预测的数字。分析结果可以看出，神经网络能够正确识别背景有显著差异的图片，一些图片中数字为黑色，另一些恰好相反为白色。即便一些数字周围有其它数字干扰，还是能够识别位于图片正中央的数字。

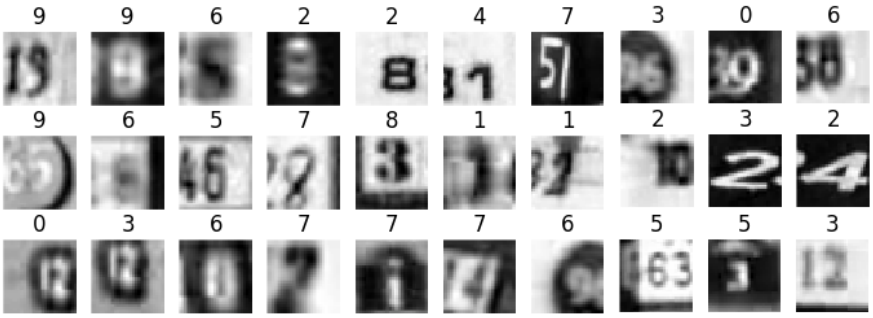

图 14: 对于单个数字的识别，部分测试集中识别错误的样本。图片上方是神经网络预测的类别。分析结果可以看出，一些图片错误分类源于图片过于模糊，而另一些错误人绝对不会犯错，说明神经网络仍有可以改进的地方。

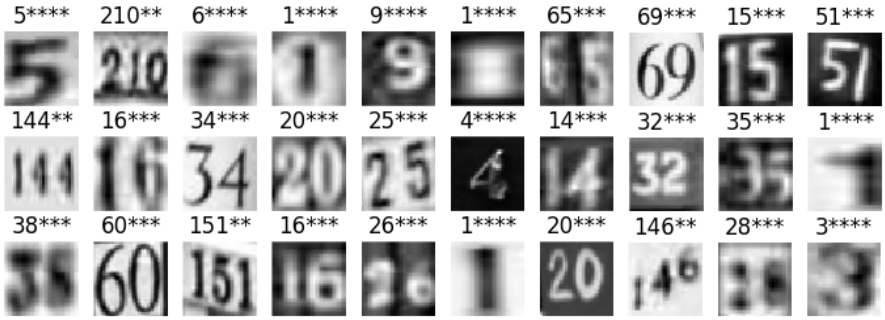

图 15: 对于数字序列识别，部分测试集中识别正确的样本。图片上方是神经网络给出的数字序列，星号表示数字不存在。

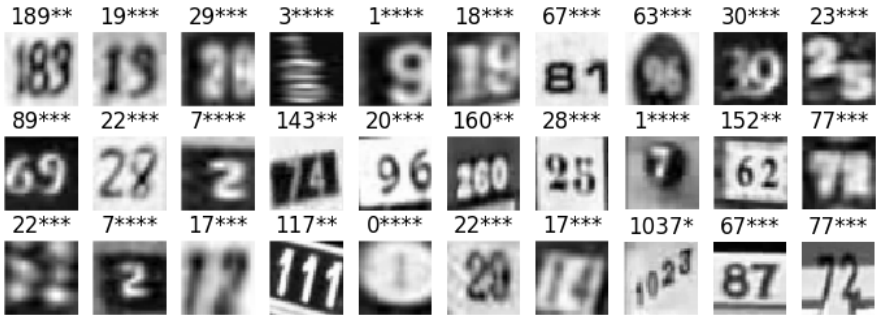

图 16: 对于数字序列识别，部分测试集中识别错误的样本。图片上方是神经网络给出的数字序列，星号表示数字不存在。

分析正确分类的图片可以看出，神经网络具有一定的灵活性。模型能够正确识别不同背景和不同颜色的数字，比如图13中第1行第1个数字的颜色是黑色，第2行第1个数字是白色。模型能够识别不同的字体，比如同样是数字6，第2行的5个数字和第2行的最后一个数字字体就有显著不同。

分析错误分类的图片可以发现，错误分类的原因可能有多种。一些图片标签可能存在错误，比如图片14中第1行第1个数字，位于图片中央的数字明显是9，模型正确识别了数字。一些图片分类错误可能是由于图片过于模糊，比如第2行的第2个数字，在人眼看来既像8，又像6。剩余最多的错误是模型没能正确识别，比如第1行第5个数字明显是8，模型却预测成为了2。

4.2.2 数字序列识别

同样对于数字序列的识别，我在训练完成得到最优的模型之后，从测试集选择了30张错误分类的图片和30张正确分类的图片，将模型识别的结果绘制成图表，如图15和16所示。

正确识别的图片中，数字序列最少有1个，最多有3个，模型均能正确识别出数字序列。这说明不将图片分割成为一个个独立的数字，直接利用神经网络识别数字序列的方

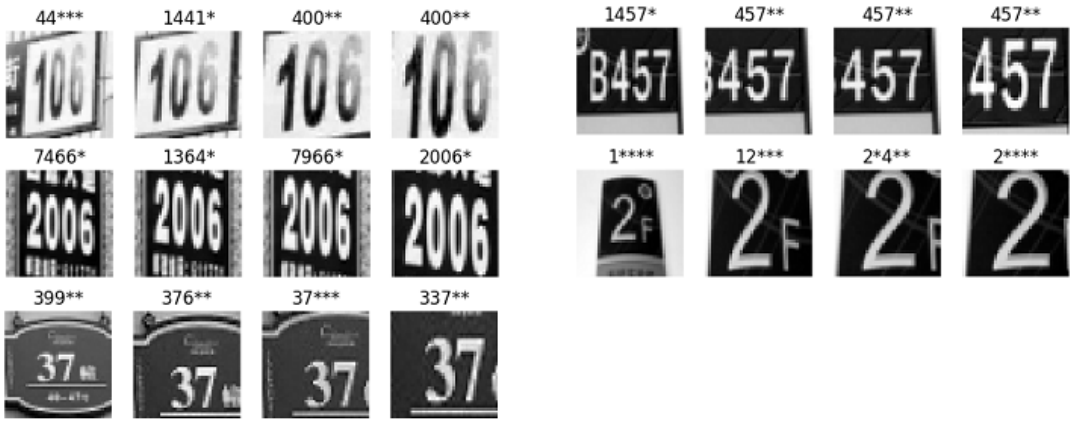

图 17: 对于数字序列识别，对于新采集图片的预测结果。预测结果位于图片上侧。总共有5组图片，每组图片4张，对应于稍有不同的剪裁。每一组图片从左至右，剪裁的区域更加针对于图片中的数字。

案是可行的。由于数字序列的长度不相同，第1个数字的位置可能位于图片的左侧，中央或者右侧，说明神经网络具有一定的平移不变性。从图15中第2行第1张图片和最后一张图片可以看到，神经网络的一个输出并非固定地对应于图片中的一个区域，第1张图片的第1个数字对应于图片的左侧，而最后一张图片的第1个数字对应于图片的右侧。

对于错误识别的图片，错误的类型和单个数字错误的类型也很类似。部分图片标签存在错误，比如图16中第1行的第1张图和第2张图，在我看来明显是189和19。部分图片过于模糊，比如第1行的第4张图片和第3行的第1张图片，即便是人也难以识别图片中的数字。部分错误来自模型把背景中的线条当作了数字，比如第2行的第9张图片模型明显把左侧边框识别成为了1。部分数字序列不在一条水平线上，似乎模型不能很好地处理这种情况，第1行的最后1张图片和第2行的第7张图片都是这种情况。更多的错误还是模型没能正确的识别，比如第2行的第1张图片和第3行的第1张图片，数字都非常清晰。

5 结论

5.1 应用于新采集的图片

在训练完成模型之后，我决定将模型应用于生活中的图片。我使用手机拍摄了几张图片，同时在搜索引擎上下载了几张图片，所有这些图片都是在中国拍摄的。

模型预测的结果如图17所示。总共右5组图片，每组图片4张，对应于稍有不同的剪裁。每一组图片从左至右，剪裁的区域更加针对于图片中的数字。

除了第1组图片，其它4组图片在某种剪裁下，模型能够正确识别数字序列。图片中数字的数量少则只有1个，多则包含4个，数字的大小、数字出现的位置也有很大的差异，模型能够准确预测说明其有一定的灵活性。

从不同的剪裁可以看出，模型受周围文字的干扰很大。第2组图片和第4组图片的一种剪裁方式包含了一个完整的英文字母，模型都把字母预测成为了一个数字而不是忽视它。当图片中不再包含字母的时候，模型便能成功地识别数字序列。当数字周围出现汉字的时候，也发生了类似的事情。第5组图片中模型把汉字预测成为了一个数字。第3组图片中，模型虽然没有把汉字预测成为数字，但汉字于预测的结果产生了影响。出现这种情况原因很可能在于街景数据集中，训练数据除了数字以外并不包含其它符号。在其它国家房屋的编号大多只有一个数字，然而在中国，街景中的数字周围往往会出现一些汉字或者是字母。有两种方案可以解决这个问题。第一种依赖于一个准确的数字定位算法，能够将数字从周围的文字中剪裁出来。第二种需要重新采集中国的街景数据，重新训练一个模型。

除了额外文字的干扰以外，模型还会受到图片中数字亮度变化的影响。第1组图片的后4张图中，并不包含任何其它文字，数字序列也非常清晰，然而模型都不能成功识别数字序列。出现问题的原因可能在于，图片中数字1的亮度和数字6的亮度存在明显的差异，即便在一个数字中，不同部分的亮度也存在差异。要想解决这个问题，可能需要继续调整模型的参数，提升在训练集上的识别准确率。

模型还会作出一些愚蠢的预测。第4组的第3张图片，模型预测第1个数字为2，第2个数字不存在，然后预测第4个数字为4。这个预测结果完全不符合约束条件，在训练数据中如果一个数字不存在，则其后的数字都不存在。第5组的第4张图片，模型预测第1个数字和第2个数字都是3，然而真实情况是只存在一个3。如果将图片分割成一个个数字，这两种错误都不会发生。项目中使用的方法不需要将图片分割成一个个数字，既是它的优点，从这两个错误可以看到，这也是它的不足。也许继续调整模型的参数能够解决这类问题。

5.2 总结

过去识别数字序列，首先需要将图片分割成为一个个数字，本项目实现了一种无需分割，直接识别数字序列的方法。这种方法简化了数字序列识别的流程，但是增加了模型的复杂性。这意味着需要更多的层数，更多的训练数据以及更精细的参数调整。实验显示，无需分割的模型也能够达到与人类相媲美的分类准确率。

项目测试了不同的改进方案对与模型识别准确率的影响。在单步改进的实验中，使用额外数据集、随机舍弃、正则化在测试集准确率上有明显改进。早期停止在单步改进时没有体现明显的提升，但在采取了避免过拟合的措施以后，测试集准确率有了明显的提升。学习率下降在单步改进和增量改进中，在测试集分类准确率上均没有体现出明显的提升。最终模型采取了4项改进，分别时正则化、随机舍弃、使用额外数据集及早期停止。

项目成功达成了设定的目标，即模型识别准确率位于世界最佳模型的10%以内。最终，单个数字识别的准确率为93.67%，相比世界最佳模型的识别准确率97.84%，差距位于10%以内；数字序列识别的准确率为87.40%，相比世界最佳模型的识别率96.03%，

差距位于10%以内。

识别数字序列的神经网络模型也充分反映了其灵活性。由于没有将图片分割成为一个个数字，数字序列的长度不同，最少只有1个数字，最多有5个数字。同时数字在图片中的位置也有很大差异，数字序列的第1个数字可以出现在图片的左侧，也可以出现在图片的右侧。

为了解决数字序列长度不同的问题，Goodfellow et al. (2014)除了预测5个数字以外还单独预测数字的个数。本项目中使用了一个稍微不同的方法，仍然是预测5个数字的类别，但是除了0-9十个数字以外，还增加了一个类别表示该数字不存在。这种方法在计算误差时要比单独预测数字的个数简单。

5.3 后续改进

在数字序列的识别中，Goodfellow et al. (2014)使用11层隐藏层的卷积神经网络，在测试集上的分类准确率为96.03%，同时实验显示神经网络的层数越多分类器的表现越好。本项目中只使用了3层隐藏层，在测试集上的分类准确率为85.32%。一个后续的改进是通过增加神经网络的层数来提高测试集分类准确率。然而，随着层数的增加，模型更加复杂，训练所需的时间也会显著增加。同时模型更复杂意味着模型的参数增加，模型更容易过拟合于训练数据，更加依赖于各种参数的设置，需要更多的训练数据，需要更长时间的调试。

SVHN数据集的训练集和测试集都给出了包含数字的矩形边框，因此不需要对数字进行定位。然而如果要将模型应用到生活中采集的图片中，定位数字是必不可少的。从5.1节应用于新采集新采集的图片一节，可以看到数字的定位对最终模型识别的结果有至关重要的影响。定位的矩形方框包含了数字以外的符号，会影响到识别的数字。定位的矩形边框过大，经过缩放后会导致数字过小而难以识别。一个后续改进就是使用神经网络定位数字的位置。数字的位置可以通过4个参数确定，分别是包含数字矩形方框的中心坐标、宽度和高度，因此这是一个回归问题。

SVHN数据集包含多个国家的街牌号图片，但是并不包含中国街牌号的图片。从5.1节应用于新采集新采集的图片一节，可以看到中国街牌号周围往往会出现汉字和字母。如果在定位的过程中，这些非数字的符号没能被剪裁掉，会影响模型识别的结果。同时在中国街牌号有相对统一的格式，标准的街牌号包含3行，第1行是街道的名称，第2行是街道号，第3行是邮政编码。因此如果想要将识别街景数字的模型应用于中国，需要重新采集中国的街牌数据。

深度学习的一个特点是，虽然训练深度学习模型需要很大的计算量，一旦模型训练完成后，利用模型的参数可以在很短的时间内计算出分类的结果。项目所使用的深度学习框架TensorFlow支持将模型参数保存到文件中，然后重新加载已经训练好的模型参数。同时TensorFlow也支持安卓平台。一个后续改进是制作一个APP，在手机平台上识别街景数字。手机具有摄像头，能够很方便地获取图片数据，并展示识别的结果。同时手机APP具有易于使用的特性，借助手机APP能够将先进的技术应用于人们生活中。

参考文献

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *Computer Science*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2010). Reading digits in natural images with unsupervised feature learning. *Nips Workshop on Deep Learning & Unsupervised Feature Learning*.

Sermanet, P., Chintala, S., and Lecun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. pages 3288–3291.

Wikipedia (2016). Convolutional neural network — wikipedia, the free encyclopedia. [Online; accessed 1-November-2016].