机器学习工程师纳米学位毕业项目
算式识别

王立兴

2019 年 3 月 10 日

目录

1 定义.....4
1.1 项目概述.....4
1.2 问题陈述.....4
1.3 评价指标.....4
2 分析.....5
2.1 数据可视化.....5
2.2 算法和技术.....7
2.2.1 分类算法.....7
2.2.2 神经网络.....8
2.2.3 卷积神经网络.....10
2.2.4 循环神经网络.....11
2.2.5 技术.....12
2.3 基准指标.....12
3 具体方法.....12
3.1 数据预处理.....12
3.2 实现.....14
3.3 改进.....15
4. 结果.....18
4.1 基础训练.....18
4.2 增强训练.....20
4.3 最终效果.....20
4.4 错误分析.....21
5 结论.....22
5.1 模型效果.....22
5.2 总结.....22
5.3 后续改进.....22

图 1　图像识别与评价指标.....4
图 2　前 5 张图的输入空间和算式图片示例.....6
图 3　字符出现频率.....7
图 4　神经网络结构.....8
图 5　使用阶跃函数的感知器.....9
图 6　两级神经元.....9
图 7 卷积层计算.....10
图 8　过滤器效果示意图.....11
图 9　循环神经网络.....12
图 10　图片归一化处理.....13
图 11　各集合字符的分布.....14
图 12　循环神经网络部分的结构.....15
图 13　RNN 3 层.....16
图 14　RNN 6 层.....17
图 15　图像旋转角度.....18
图 16　前 20 次训练 loss 图.....19
图 17　80 次训练 loss 图.....20
图 18　模型实际标签和预测结果.....21

1 定义

1.1 项目概述

项目使用深度学习识别具有加减乘括号算式的图片，输出算式内容，是一个序列识别（recognition sequence）问题。

尽管文本的 OCR（Optical Character Recognition）已经有广泛研究，但任意长度的文本识别仍然是很有挑战的一个问题。此项目由图像识别出字符，虽然不是真实光线环境，但也是有各种颜色和噪点干扰，可以算是基础版的 OCR。作为初次进入此领域的研究者，限定字符范围的算式识别的探索，可为将来实际应用的文本识别奠定基础。

项目采用了卷积循环神经网络(Convolutional Recurrent Neural Network)模型以及 CTC(Connectionist Temporal Classification)来实现项目需求。[1]

1.2 问题陈述

数据集中的标签已经给出。首先可以通过对数据集的探索，发现字符类型有 16 种（不含双引号）“+*/()=0123456789”，最长字符串为 11 个。在此应用背景下简单估算，数据可能的排列已达到 16^11 次方的量级。对每个图像单独做识别处理，是不现实的。所以需要从图像逐步的识别单个字符，再统一处理成连续的算式字符串。

1.3 评价指标

每一个图片由一系列的字符组成，只要其中一个字符错误，那么整个算式就极可能等式两边不成立。例如下图将等式右侧的 7 识别为了 1。

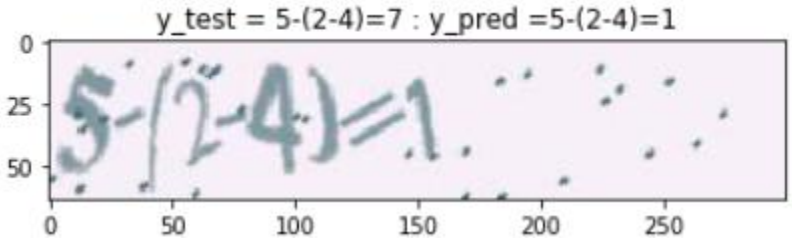


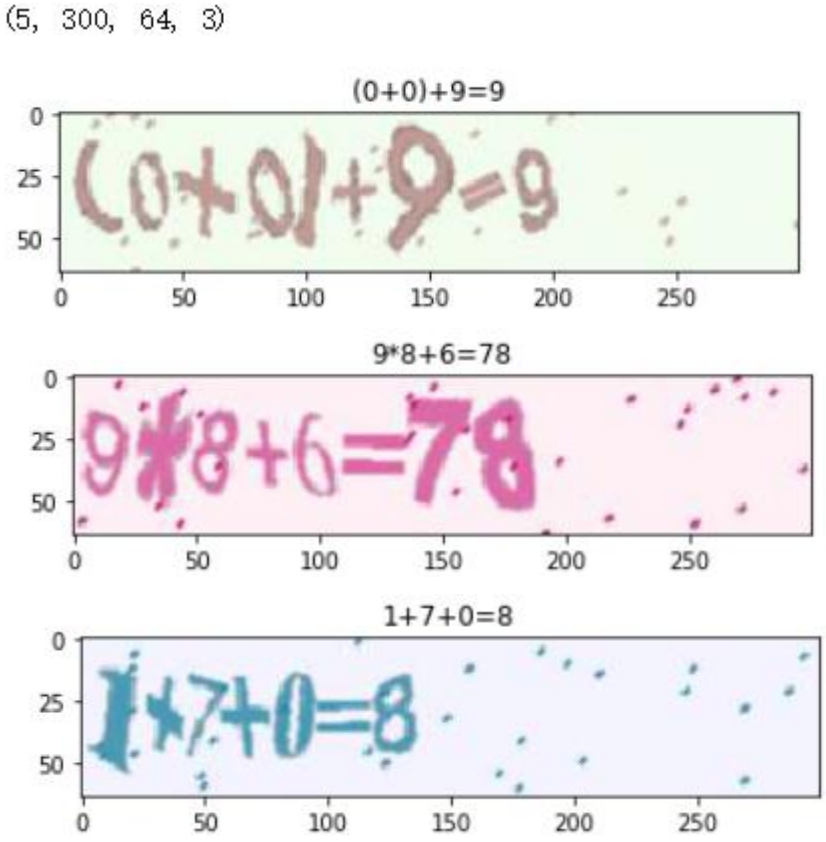
图 1 图像识别与评价指标

所以整个算式应该以是否每个字符都识别对为基准。评价模型的好坏，标准可以用准确率（Accuracy）：样本中每个字符都识别对的样本数目**除以**样本总数。且此处准确率应以未参与训练的测试集样本为准。

2 分析

2.1 数据可视化

数据集由课程提供，可以通过网络下载[2]。该数据集包含了 10 万张图片，每张图片里都有一个算式。每个算式可能包含 +-* 三种运算符，可能包含一对括号，也可能不包含括号。每个字符都可能旋转，所以 + 号可能长得像我们平时手写的 * 号，不过 * 号有六个瓣。每个图片的尺寸都一样，宽 300 像素，高 64 像素，RGB 三色，有杂散的点干扰。举例如下图。



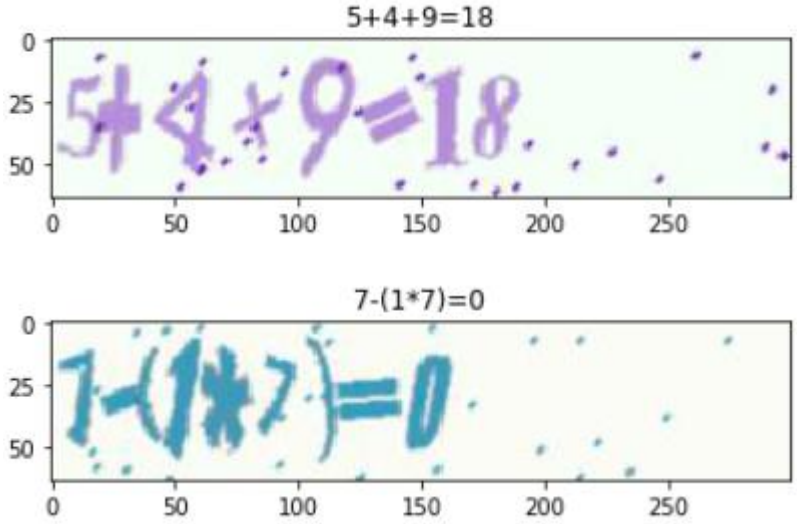
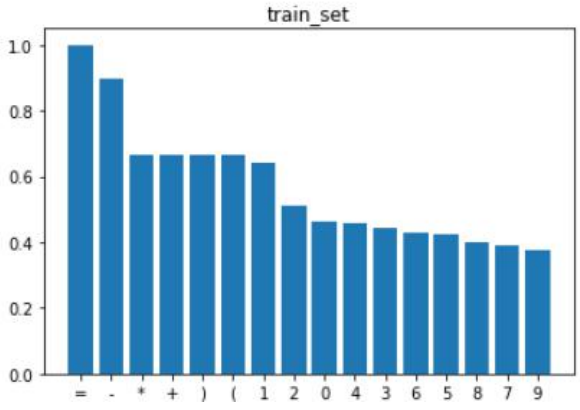


图 2 前 5 张图的输入空间和算式图片示例

下载链接里还包含一个 csv 文件，将图片路径和图片里的算式信息一一对应。

数据集总共包含 10 万条算式图像。为了评估性能将其随机打乱并分为训练集、验证集和测试集，数目分别为 6 万、2 万、2 万。并统计各字符在不同集合里出现的频率。



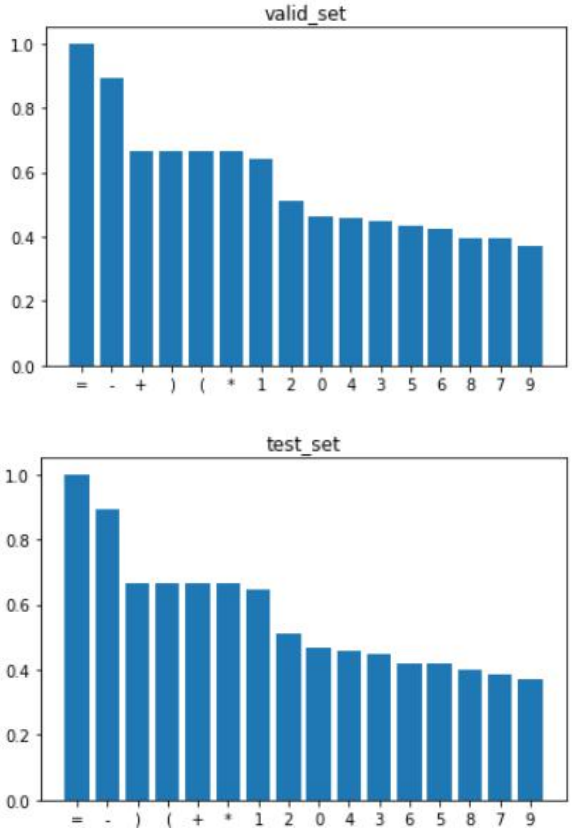


图 3 字符出现频率

可以观察到等号 100%出现，减号出现频率次高。其余小括号加号乘号出现概率持平。仅验证集的 5、6 与训练集和测试集频率相反，但其差别很小，可以忽略。

2.2 算法和技术

2.2.1 分类算法

根据前面的分析，算式的识别包含两步，第一步对局部图像分类，识别单个字符。用于分类的算法则有朴素贝叶斯、决策树、支持向量机、逻辑回归、神经网络等等。

算法的输入是局部图像每一个像素所组成的特征向量，输出是算式字符的类别。输入的局部图片是一个特征空间，而图片的每一个像素是空间的维度。

朴素贝叶斯算法，基于贝叶斯定理与特征条件独立假设的分类方法。对于图片类的像素输入，每个相邻像素之间存在关联而不是独立存在，不适合使用朴素贝叶斯算法。

决策树算法，用所有训练数据根据输入的特征向量和对应标签，基于一定的准则将数据分配到二叉树结构的两支，再不断往下递归分配。而图像的像素点多，如果按照像素点展开，将是庞大的不可实际执行的数量。另外，图像结果与某个固定位置也无关，不能按照某个点分支决策。

逻辑回归算法，是以线性回归为基础，通过非线性映射函数来实现得到结果分类的概率。单个逻辑回归分类器只能做二分类。而对于图片的多类别识别，虽然可以构造多个逻辑回归分类器，但平移旋转图像标签不变，逻辑回归的判别方式，并不适合。

支持向量机算法，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。同逻辑回归一样属于二分类，即使是利用核函数，我们可以把平面投射成曲面，也无法很好的解决图像本身平移旋转标签不变的问题。

2.2.2 神经网络

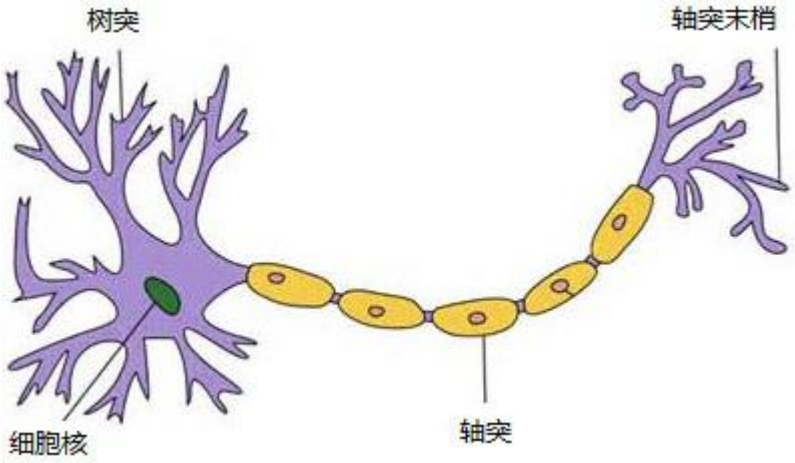


图 4 神经网络结构

19 世纪，意大利解剖学家 C.Golgi 发明神经细胞银染法，在显微镜下首次展示完整形态的神经元：胞体、树突和轴突。稍后，西班牙 S Ramon Y.Cajal 根据这一结果做了系统观察研究，提出神经细胞是彼此独立、又互相连接的神经元学说，并于 1904 年出版《人和脊椎动物神经系统组织学》[3]。

1943 年，神经生理学家 Warren McCulloch 和数学家 Walter Pitts 建立了神经网络和数学模型并发表论文[3]，称为 M-P 模型。他们通过 M-P 模型提出了神经元的形式化数学描述和网络结构方法，证明了单个神经元能执行逻辑功能，从而开创了人工神经网络研究的时代。

神经元模型是一个包含输入，输出与计算功能的模型。输入可以类比为神经元的树突，而输出可以类比为神经元的轴突，计算则可以类比为细胞核。如下图，通

过对输入求加权和，即图中的 W 与 x 点积加上 b * 1，再通过阶跃函数非线性化，输出 Yes/No (即 1/0)

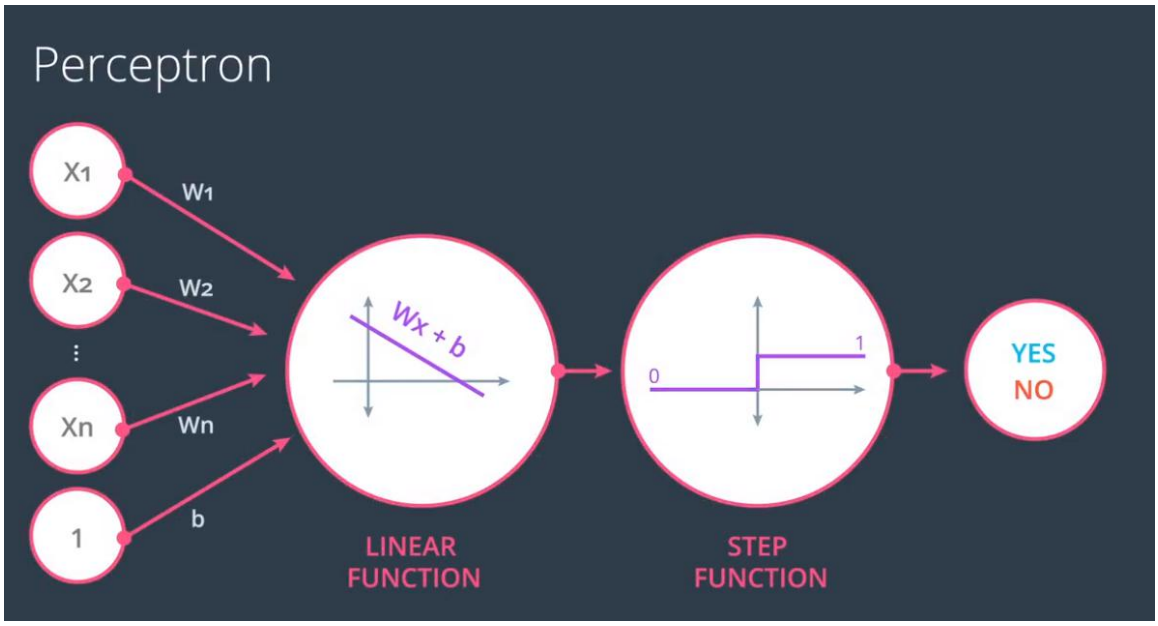


图 5 使用阶跃函数的感知器

而一个神经元处理问题的复杂度是有限的。类似人的神经系统，多个神经元的级联，造就了人类处理复杂事物能力的基础。如下图，使用了两级神经元组成神经网络可以实现更加复杂的处理。神经网络分为输入层、隐藏层、输出层，依次对应下图的左中右列。神经网络层数叫做深度，而隐藏层可以多层级联，多层的神经网络也是深度学习名字的由来。

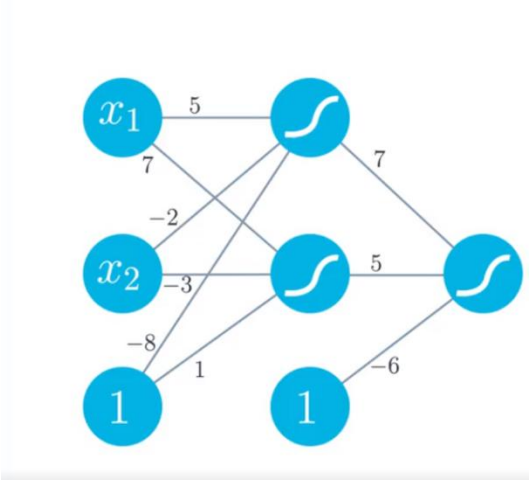


图 6 两级神经元

一开始每个神经元的权重是随机的。对于分类问题，可以通过神经元网络预测结果与真实类别计算出损失函数交叉熵（cross entropy），交叉熵对于每层权重又是可导的，因此可用梯度下降法来更新权重，减少交叉熵。这样对于神经网络庞大的

权重参数提供一条高效调整参数的方法。参数由结果开始的逆向修正，称之为反向传播。

2.2.3 卷积神经网络

20 世纪 60 年代，Hubel 和 Wiesel 在研究猫脑皮层中神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络（Convolutional Neural Networks）。[5]

同时，人眼在识别图像时，往往从局部到全局，局部与局部之间联系往往不太紧密。结合到神经网络，我们发现不需要神经网络中的每个结点都掌握全局的知识，因此从这里可以大量减少需要学习的参数数量。

神经网络，尤其是卷积神经网络，通过核在图像上的移动识别出图像特定的形状。再由全连接层汇总，对各个特征赋予不同权重，最终确定是什么类型。这样即使图像有平移和旋转，也能有效的检测出来。

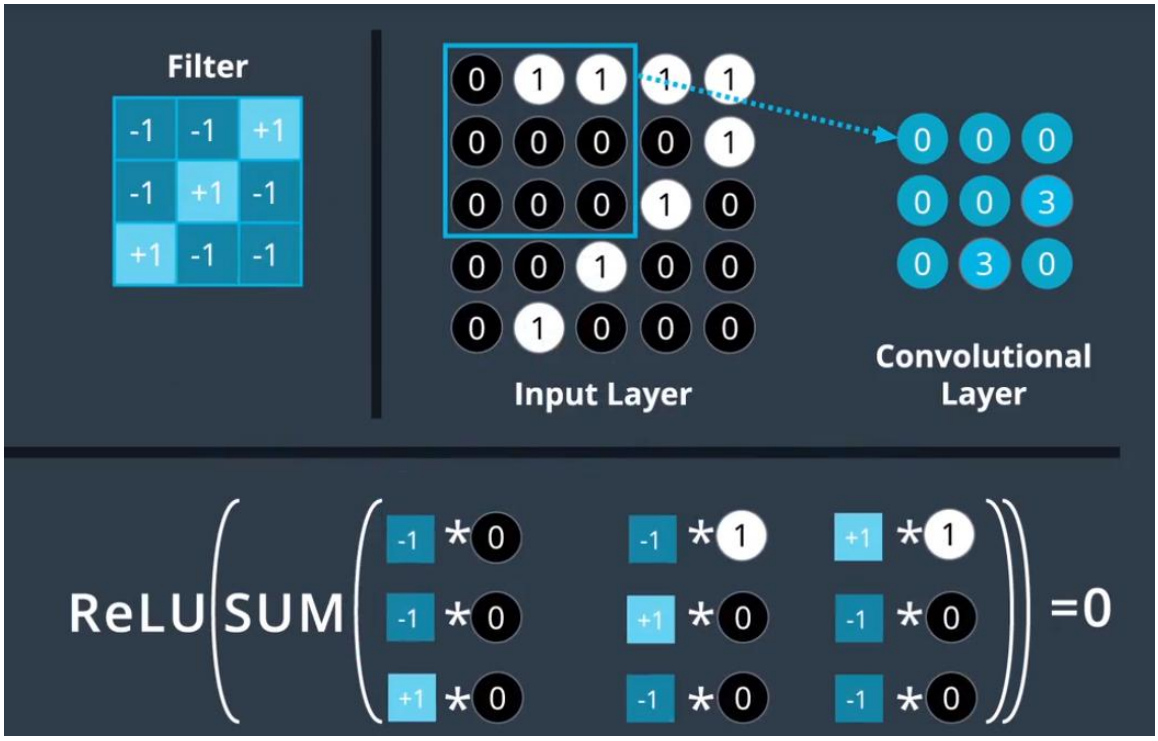


图 7 卷积层计算

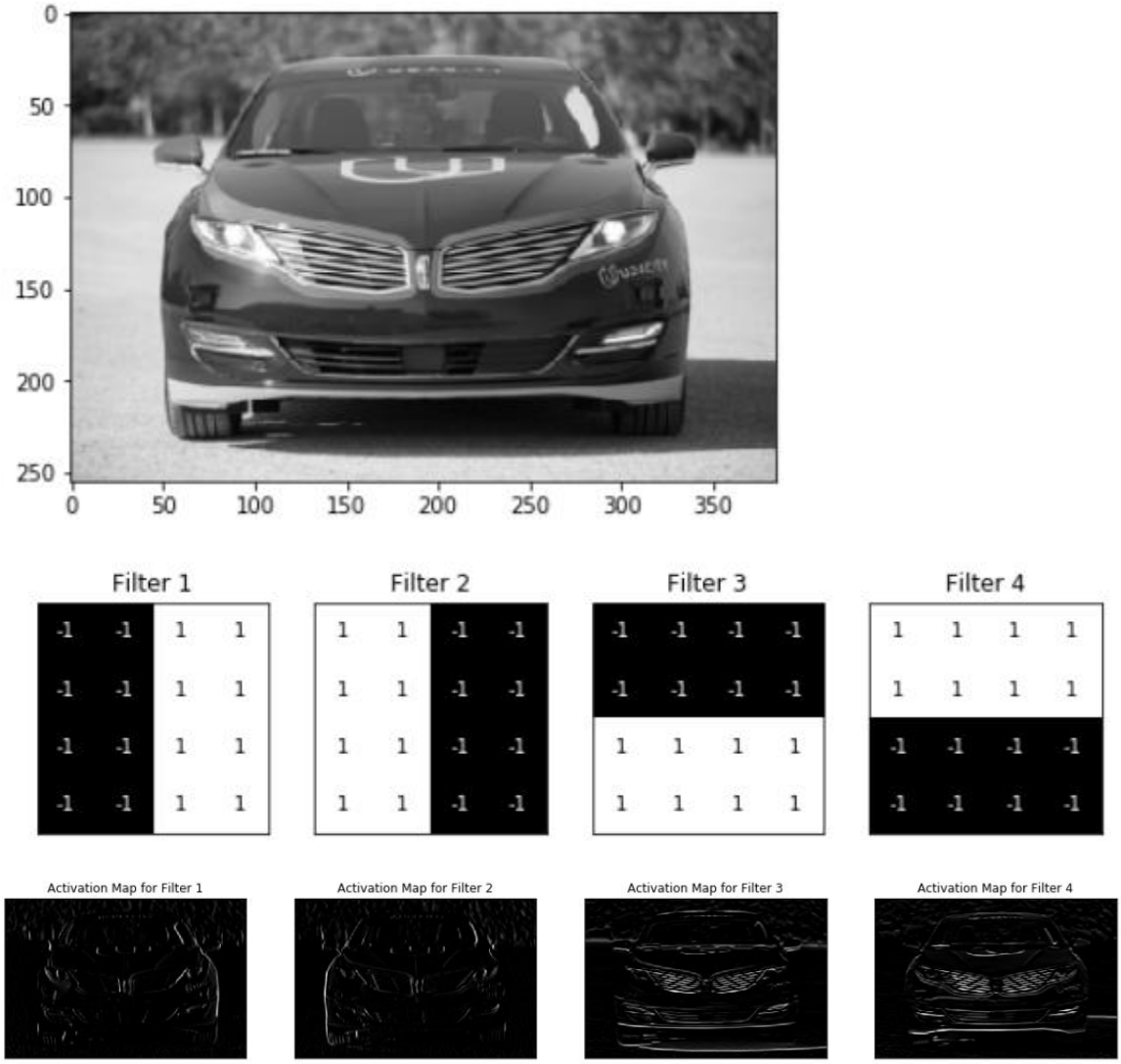


图 8 过滤器效果示意图

2.2.4 循环神经网络

循环神经网络一般用于对序列数据建模。对于一串字符的排列组合很多，不可能对图片单独分类识别，需要对图片切片，以序列的形式逐个输入识别独立的字符。不同与卷积神经网络的是，循环神经网络的隐藏层随着输入的序列也是逐次变化的。参考图示如下[1]。

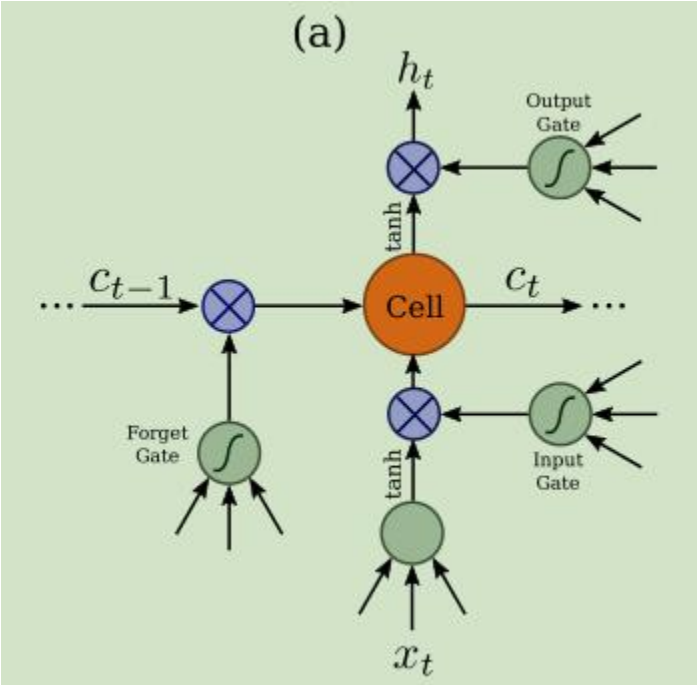


图 9 循环神经网络

2.2.5 技术

项目采用 Google 的开源学习框架 tensorflow，并使用高阶接口 keras 简化配置模型的复杂度。

得益于 keras 的封装，可以更简便的实现模型的搭建配置，以及模型的展示。

2.3 基准指标

项目的基准要求达到 99%的正确率。

3 具体方法

3.1 数据预处理

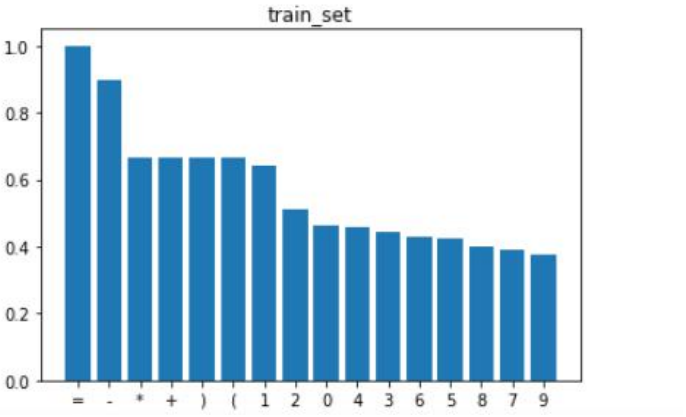
图片包含红绿蓝三色，一色一个通道。每个通道为[0, 255]之间整数，不利于网络的训练和收敛，所以会进行归一化，化为 0 到 1 的浮点数来方便处理。处理代码如下。

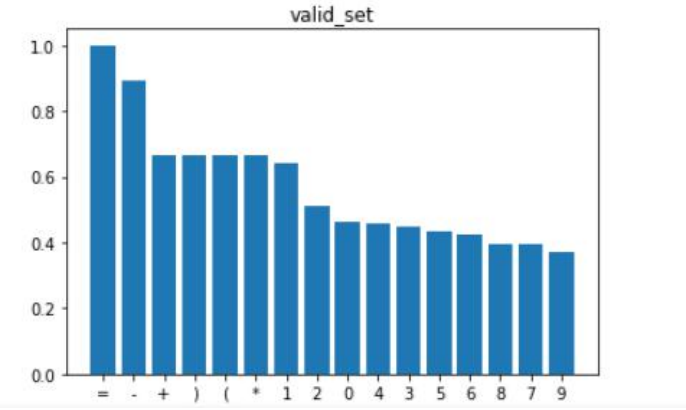
```
img = img.astype(np.float32) / 255
return img
```

图 10 图片归一化处理

图片总数 10 万张。数据打乱顺序，划分 6 万、2 万、2 万张图片为训练集，验证集和测试集。

查看不同集合的字符分布，也是基本一致，不做额外处理。





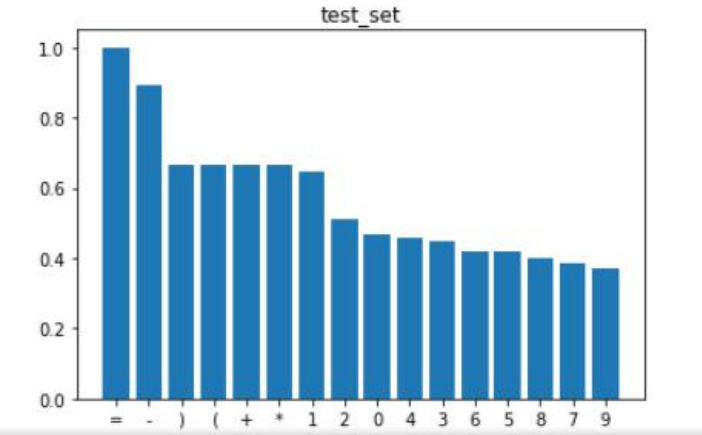


图 11 各集合字符的分布

3.2 实现

通过添加卷积层识别图片中的形状的特性，增加 batch normalization 层，避免调整前层数据引起数据分布变化。增加最大池化层，缩小每层图像大小，抽象出图像特征信息。

通过 CNN 处理后的数据使用了循环神经网络 LSTM(Long Short Term Memory)的变种，循环门单元(Gated Recurrent Unit，GRU)来实现序列的输入处理。最后通过 CTC 输出字符串。

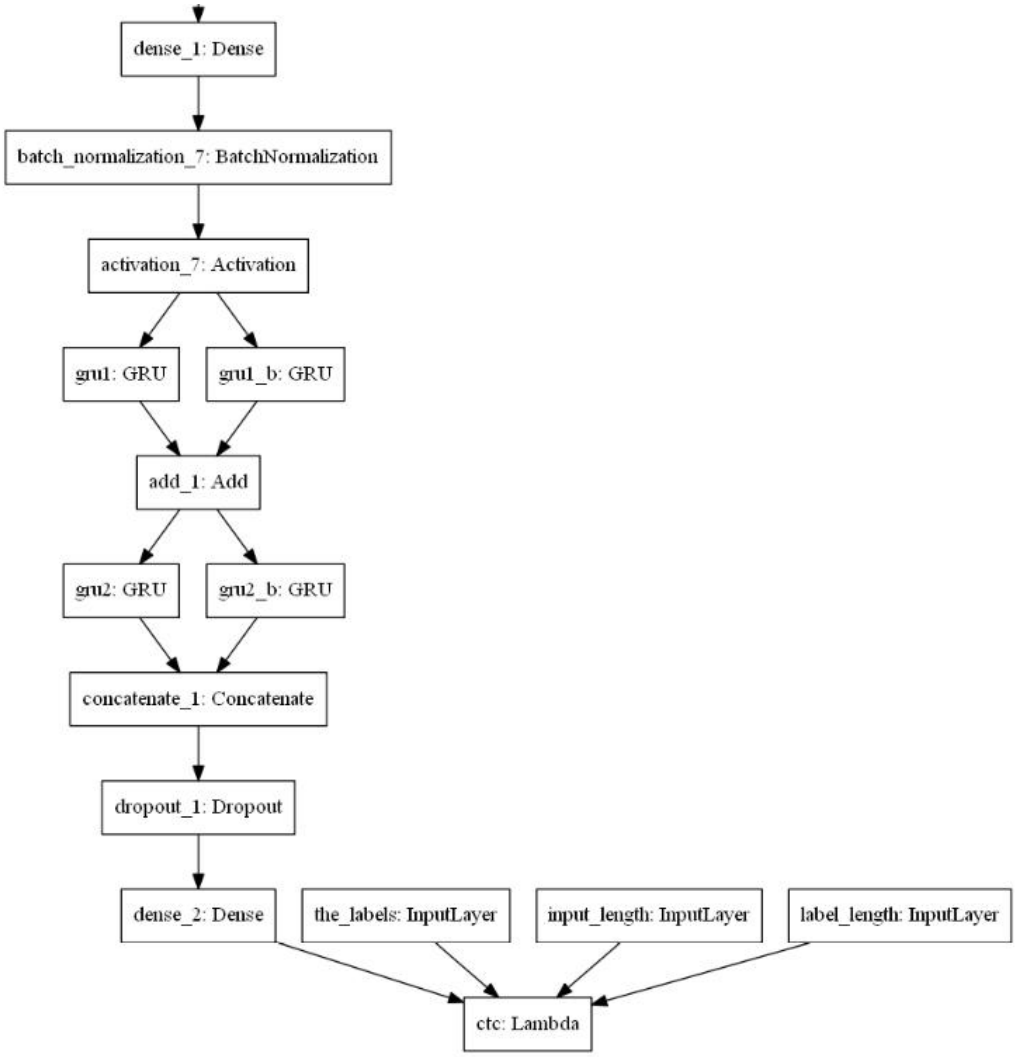


图 12 循环神经网络部分的结构

3.3 改进

模型前面 CNN 部分，不考虑池化和标准化层，最初使用了课程最基本的 3 层结构，loss 无法再小。后面增加到了 6 层了，此问题得到了改善。

Layer (type)	Output Shape	Param #	Connected to
the_input (InputLayer)	(None, 300, 64, 3)	0	
conv2d_25 (Conv2D)	(None, 300, 64, 16)	208	the_input[0][0]
batch_normalization_30 (BatchNo	(None, 300, 64, 16)	64	conv2d_25[0][0]
activation_30 (Activation)	(None, 300, 64, 16)	0	batch_normalization_30[0][0]
max_pooling2d_16 (MaxPooling2D)	(None, 150, 32, 16)	0	activation_30[0][0]
conv2d_26 (Conv2D)	(None, 150, 32, 32)	2080	max_pooling2d_16[0][0]
batch_normalization_31 (BatchNo	(None, 150, 32, 32)	128	conv2d_26[0][0]
activation_31 (Activation)	(None, 150, 32, 32)	0	batch_normalization_31[0][0]
max_pooling2d_17 (MaxPooling2D)	(None, 75, 16, 32)	0	activation_31[0][0]
conv2d_27 (Conv2D)	(None, 75, 16, 64)	8256	max_pooling2d_17[0][0]
batch_normalization_32 (BatchNo	(None, 75, 16, 64)	256	conv2d_27[0][0]
activation_32 (Activation)	(None, 75, 16, 64)	0	batch_normalization_32[0][0]
max_pooling2d_18 (MaxPooling2D)	(None, 37, 8, 64)	0	activation_32[0][0]

图 13 RNN 3 层

the_input (InputLayer)	(None, 300, 64, 3)	0	
conv2d_1 (Conv2D)	(None, 300, 64, 32)	2432	the_input[0][0]
batch_normalization_1 (BatchNor	(None, 300, 64, 32)	128	conv2d_1[0][0]
activation_1 (Activation)	(None, 300, 64, 32)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 300, 64, 32)	25632	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 300, 64, 32)	128	conv2d_2[0][0]
activation_2 (Activation)	(None, 300, 64, 32)	0	batch_normalization_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 150, 32, 32)	0	activation_2[0][0]
conv2d_3 (Conv2D)	(None, 150, 32, 64)	32832	max_pooling2d_1[0][0]
batch_normalization_3 (BatchNor	(None, 150, 32, 64)	256	conv2d_3[0][0]
activation_3 (Activation)	(None, 150, 32, 64)	0	batch_normalization_3[0][0]
conv2d_4 (Conv2D)	(None, 150, 32, 64)	65600	activation_3[0][0]
batch_normalization_4 (BatchNor	(None, 150, 32, 64)	256	conv2d_4[0][0]
activation_4 (Activation)	(None, 150, 32, 64)	0	batch_normalization_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 75, 16, 64)	0	activation_4[0][0]
conv2d_5 (Conv2D)	(None, 75, 16, 128)	73856	max_pooling2d_2[0][0]
batch_normalization_5 (BatchNor	(None, 75, 16, 128)	512	conv2d_5[0][0]
activation_5 (Activation)	(None, 75, 16, 128)	0	batch_normalization_5[0][0]
conv2d_6 (Conv2D)	(None, 75, 16, 128)	147584	activation_5[0][0]
batch_normalization_6 (BatchNor	(None, 75, 16, 128)	512	conv2d_6[0][0]
activation_6 (Activation)	(None, 75, 16, 128)	0	batch_normalization_6[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 37, 8, 128)	0	activation_6[0][0]

图 14 RNN 6 层

在确定 RNN 的模型后，又对 Dropout 参数尝试调整，目标提高准确率和健壮性。最终确定 Dropout 为 0.4。改进过程的表格对比如下。

RNN 模型层数	最大 epoch	验证集最优 epoch	测试集准确率
3	20	20	96.412060%
6	20	12	99.000000%

Dropout 参数	最大 epoch	验证集最优 epoch	测试集准确率
0.4	20	12	99.000000%
0.6	20	17	98.854271%

使用 6 万张训练图片训练了 20 轮。在第 12 次时的验证集 loss 最小，用此时的模型在测试集上测试，结果发现正确率为 99.000000%。为了进一步提升测试集正确率，采用了数据增强，调整图片角度再训练。在总第 70 轮时，验证集 loss 最小，此时在测试集上的正确率达到了 99.155779%，相对预期 99%正确率的目标，还降低 15.5%的错误率。

根据图片宽度减去高度整除 5 的差值乘以 3 除以宽度再加 1 作为范围度数，随机旋转角度。使图片有一定的旋转，但又不至于太大而丢失信息。

```
img = image.random_rotation(img,
                            3 * (w - h//5) / w + 1,
                            row_axis=0,
                            col_axis=1,
                            channel_axis=2,)
```

图 15 图像旋转角度

4. 结果

4.1 基础训练

前 20 轮训练采用原始数据进行训练，使用了 Adam 优化算法（Adam 优化算法可以看作 RMSprop 和 Momentum 的结合。和 RMSprop 对二阶动量使用指数移动平均类似，Adam 中对一阶动量也是用指数移动平均计算）。loss 初期减小很快，

后面几乎不再变化。

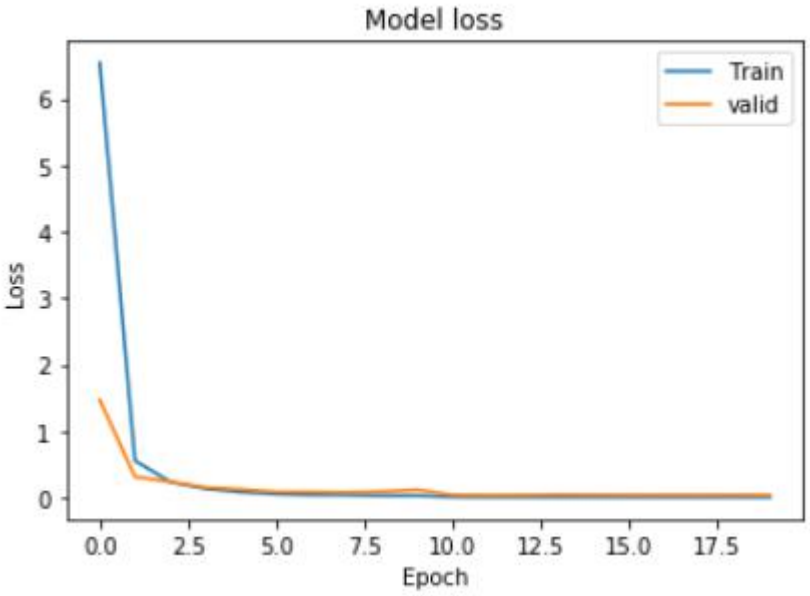


图 16 前 20 次训练 loss 图

对前 20 轮验证集 loss 最小的第 12 次测试集正确率为 99.000000%，第 20 轮测试集正确率为 99.050251%，达到了 99%正确率的设定目标。但是为了进一步提升正确率，增强模型泛化能力，对图片进行了旋转。

4.2 增强训练

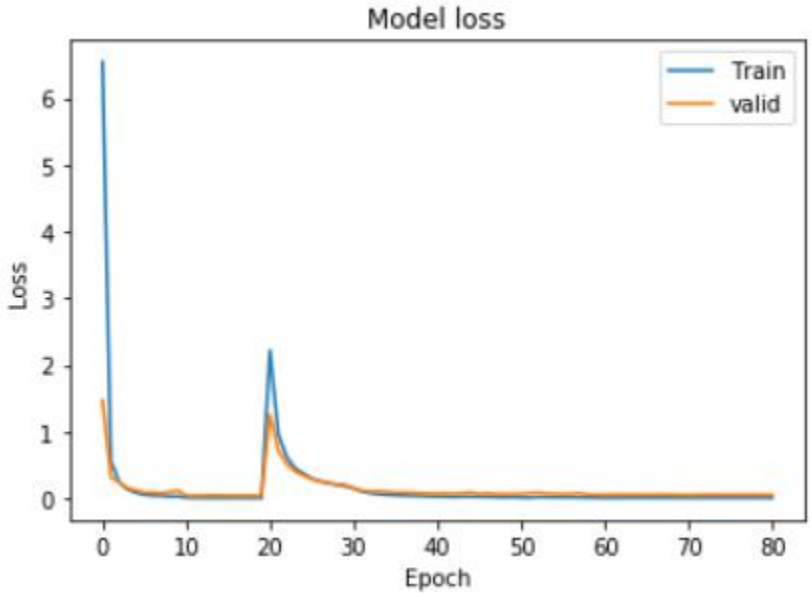
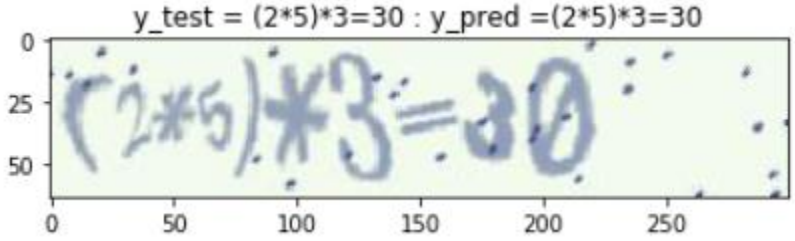


图 17 80 次训练 loss 图

刚开始模型的 loss 增高，说明最初的训练结果偏向过拟合了，对有旋转角度的图片泛化能力不好。通过多次训练以后 loss 不断减小，虽然没有最初未做旋转的情况小，但是通过对后面 60 轮里，验证集 loss 最小的一次（总第 70 轮）以及最后一次训练，对测试集测试正确率均达到了 99.155779% 。 通过对数据旋转处理，增强了模型的泛化能力，相对基准 99%正确率，降低了 15.5%的错误率。

4.3 最终效果

最终在测试集上取得不错的效果。有些 1 或者 7 人眼来看也是容易混淆的，但通过模型仍然能识别到正确的结果。对于人眼难辨别的颜色也是没有问题的。



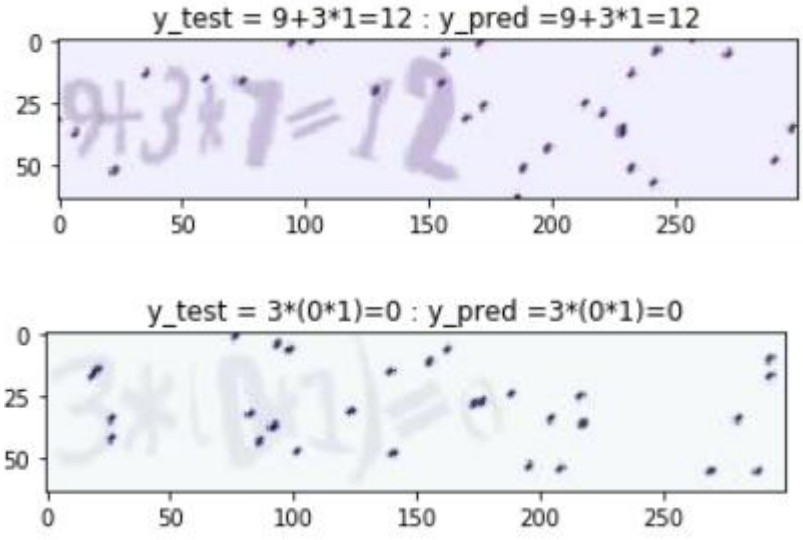
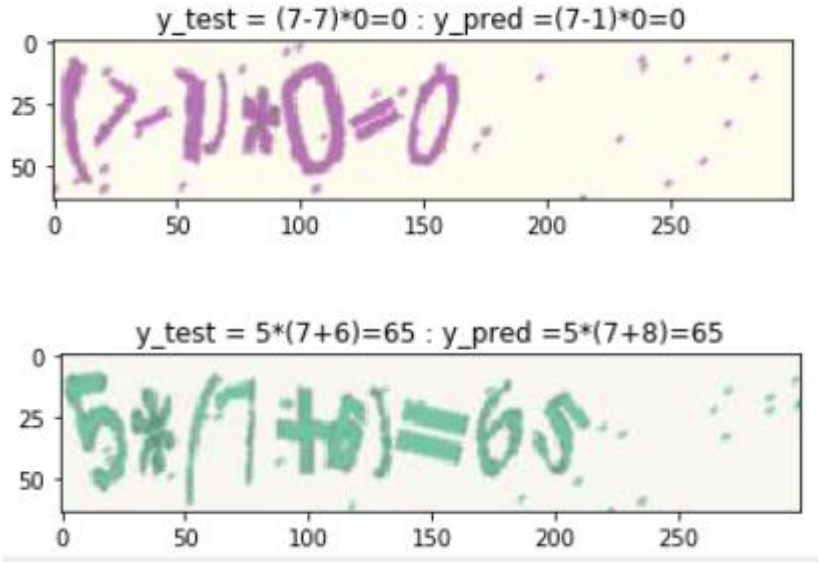
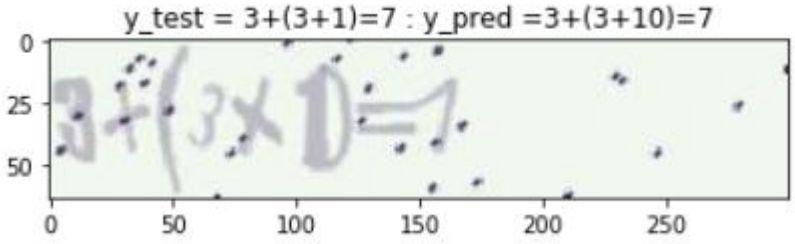


图 18 模型实际标签和预测结果

4.4 错误分析

模型对数字 1 和 7，6 和 8，图像相近容易混淆。另外在 CTC 处理过程中还会出现两个字符图像组合出多一个字符的错误，如将 1 和) 都识别了，并在中间过程产生了 0，此种错误人是不会犯的。





5 结论

5.1 模型效果

模型能够以较高的准确率识别图像，但对于不少容易区分的图像识别还是有个别字符错误，导致模型的正确率未达到极好的程度。

5.2 总结

相对传统 CNN 的图片分类（比如猫狗分类），字符串的识别只要错其中一个字符就是错。本项目通过 CRNN，实现了无需单个字符独立训练，即可用整个字符串来对图片进行训练学习。此次图像都是批量生成的，具有很高的共性，不像现实生活中字符有大有小，有不同形式的干扰因素。加入现实图片对于实际 OCR 的应用会有更高的价值。

5.3 后续改进

引入实际打印的算式图片来增加模型的应用场景。比如小学生作业的检查。

增加模型的层数。但是也有可能带来过拟合，这是建立模型过程需要时刻注意避免的。

实现识别的可泛化后，制作网页版的服务器，可以提供现实的数据增强，比如立刻得到算式的计算结果。

参考文献

[1] Baoguang Shi, Xiang Bai, Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition [J]. *IEEE transactions on pattern analysis and machine intelligence*, 2017, 39(11): 2298-2304

[2] Data source: https://s3.cn-north-1.amazonaws.com.cn/static-documents/nd009/MLND+Capstone/Mathematical_Expression_Recognition_train.zip

[3] Cajal, S. R. English translations: *Histology of the Nervous System of Man and Vertebrates* (trans. N. Swanson and L. W. Swanson), New York: Oxford University Press, 1995

[4] McCulloch, W. S. & W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity [J]. *Bulletin of Mathematical Biology*, 1943, 5(4): 115–133

[5] Hubel, D. H. & T. N. Wiesel. Shape and arrangement of columns in cat's striate cortex [J]. *The Journal of Physiology*, 1963, 165(3): 559–568.2.