

机器学习纳米学位

毕业项目（算式识别）

杨学冉

2018 年 12 月 04 日

I. 问题的定义

项目概述

项目要求使用深度学习方法识别图像中的数学算式序列。问题涉及到深度学习、计算机视觉等领域。使用深度学习可以解决计算机视觉领域的很多问题，特别是在使用卷积神经网络时。在近些年举办的图像识别大赛中，获得冠军的模型算法均使用了具有 CNN 结构的网络。卷积神经网络旨在通过最少的预处理直接从像素图像识别视觉图案。卷积神经网络的卷积核能够作为图像的特征提取器将图像特征抽象地提取。

本项目提供的数据集是包含算式序列的图像，序列信息的识别不能直接通过多分类的问题的方法解决。对于许多应用场景如语音识别、场景文本识别等，如果在输入到 CNN 网络之前对语音信息序列进行分割，成本和难度都会非常高。并且 CNN 网络无法利用包含在序列中的上下文信息。使用 RNN 网络能够解决这一问题，RNN 网络的输入和输出都可以是序列。并且在 RNN 网络中，序列上下文的信息得到利用，序列中的字符不再孤立存在。本项目需要输入序列，输出也是序列。需要用到 CTC (Connectionist Temporal Classification) 损失函数[1]。

本项目将使用 CRNN 的方案，即 CNN+RNN(LSTM)+CTC loss 的模型结构。

问题陈述

项目提供的数据集是由计算机生成的 100000 张彩色图像。每张图像包含一个算式。字符有“0~9”10 个数字、“+、-、*”3 个运算符、一对括号和一个“=”，共计 16 种字符。不同内容的算式共有 26341 种，相同算式内容的图像样本很少。

图像中的每种字符呈现不同程度的缩放、旋转，字符的字体、笔画粗细也不相同。字符的背景不是纯净的颜色，存在颗粒状的噪声。图片示例输入的图像。算式的长度也不相同，最长的序列有 11 个字符，最短的则有 7 个。

算式识别输入的是包含算式的没有被分割的图像，输出的是图像中的算式序列。因此模型要有识别序列的能力。

评价指标

本项目将使用准确率作为评价指标。具体而言，模型输出的序列与真实算式序列完一致才视为识别正确。任何的字符错误、长度不一致都不算作正确识别。

II. 分析

数据的探索

项目提供的数据集是 100000 张 64*300 像素的像彩色图像，图像大小一致。图 1 是在原始数据集中随机选取的 4 个算式图像样本。算式中可能出现的字符有“0~9”10 个数字、“+、-、*”3 个运算符、一对括号和一个“=”，共计 16 种字符。“+”和“*”的区别在于后者有 6 个瓣。图像中的每种字符呈现不同程度的缩放、旋转，字符的字体、笔画粗细也不相同。字符的背景不是纯净的颜色，存在颗粒状的图像噪声。算式中字符之间的距离很近，并且可以看出算式中的字符序列不是一一对齐的，分割图像中的字符难度较大。因此本项目中会使用到与序列到序列识别的相关算法。



图 1 数据集中随机选取的 4 个图像样本

为了解数据的更多信息，我对图像中算式的长度、图像中的字符在数据集中的分布做可视化展示。数据集中所有算式中每种字符的数量分布如图 3 所示。算式的长度也不一致，其

分布如图 2 所示。

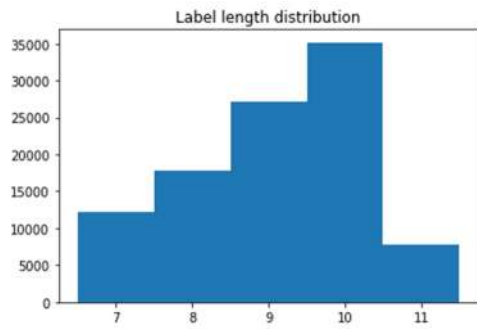


图 2 数据集中算式长度分布

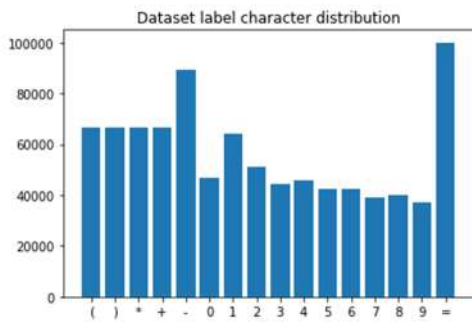


图 3 数据集中算式字符分布

数据集需要分成训练集、验证集和测试集 3 个部分，在本项目中按照 80%、10%、10% 的占比将原始数据集随机分成这 3 部分。每个数据集中字符长度的分布如图所示。可见分割后的数据集的算式长度分布与原始数据集大体一致。

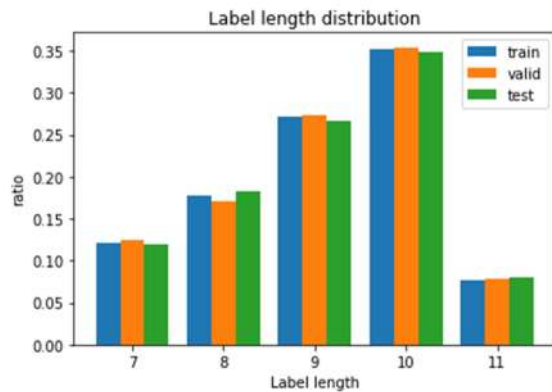


图 4 训练集、验证集、测试集中算式长度的分布

算法和技术

在众多分类算法中，近些年来在图像识别领域应用最广的是 CNN（卷积神经网络）。CNN 是一种特殊的多层神经网络，和普通的神经网络一样使用反向传播算法进行训练。CNN 网络使用被称为过滤器或卷积核的卷积窗口对图像进行卷积操作。每个卷积核处理来自上一层处于其感受野的数据，将图像特征抽象化。CNN 可用于图像和视频识别、语音识别等领域。

Yann LeCun 在 1998 年设计了早期卷积神经网络中最有代表性的网络模型：LeNet[4]，专为手写和机器打印字符识别而设计。目前 LeNet 网络已经更新到最新的 LeNet-5，可以识别具有极端可变性的图案(例如手写字符)，并且具有对扭曲和简单几何变换识别的鲁棒性。目前流行的 CNN 网络结构：ResNet、VGG16、AlexNet 等。这些网络在 ImageNet 等数据集相关的竞赛中取得了很好的成绩。

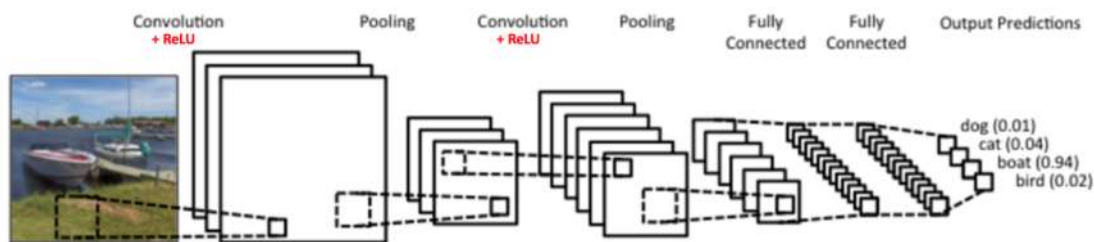


图 5 一个简单的卷积神经网络结构[7]

图 5 是一个简单的卷积神经网络，图中使用了两个卷积层。第一个卷积层输入图像的像素值，卷积层对其进行卷积操作，得到新的像素值。这些新像素值通过一个非线性的激活函数（卷积神经网络中通常使用 ReLU）进行非线性变换，得到第一层的特征值。图中卷积层后有一个池化层，用来降低卷积层的特征维度，同时保留重要信息，减弱过拟合。经过两次卷积-池化后，图像空间维度被大大降低，但特征深度得到增加。这些特征值通过全连接层和 softmax 函数后就可以得到分类结果。

CNN 网络可以识别图像中的字符，对于单个字符或者已经分割好的序列的识别，可以作为普通的二元或多分类问题。这类问题可以使用分类交叉熵作为损失函数进行模型训练。而算式识别项目的输入图像是完整的算式序列，需要输出的也是序列。因此本项目还要处理序列到序列的问题。CNN 网络无法利用图像中元素序列之间的关系，比如一个算式中出现了一个左括号，这个算式后面必定会有一个右括号，或者等号前后的算式数值相同。这些信息对预测算式内容也是有益的[2, 3]。使用 CNN 网络获取了图像的特征后，我使用 RNN 网络来对特征进行处理。

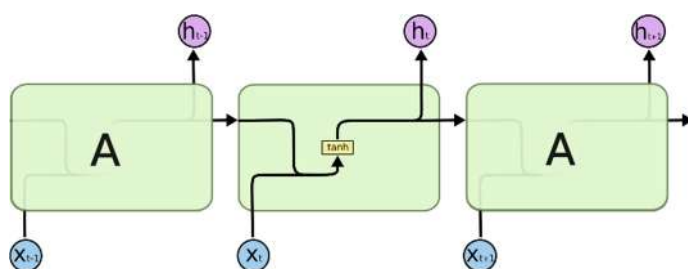


图 6 标准的 RNN 网络结构

RNN (Recurrent Neural Network) 是一类用于处理序列数据的神经网络。RNN 当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。如图 6 中， $t-1$ 时刻的输出不仅会传送到 h_{t-1} 中，还会传送到下一时刻 t 的输入中。

理论上，RNN 能够对任何长度的序列数据进行处理。LSTM (长短期记忆网络) 是 RNN 的

一种，相对于原始的 RNN 网络，LSTM 在隐藏层中添加了门逻辑（输入门、遗忘门、输出门）。在处理较长序列时能够解决信息在隐藏层直间传递时被不断弱化的问题。GRU 网络额也是针对 RNN 网络长期记忆丢失问题的一个变种。本项目我采用了 LSTM 网络。

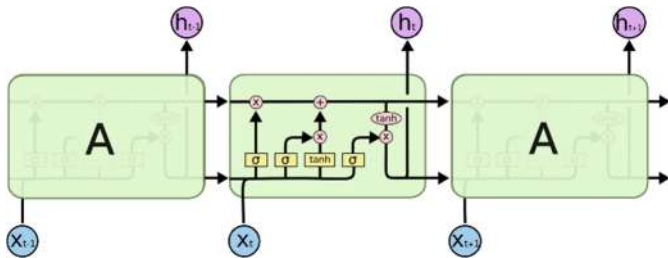


图 7 LSTM 的网络结构

算式识别本质上是图像中字符的分类问题。由于没有分割出单个字符，输入与输出的序列没有对齐，我们无法使用 softmax 函数来获得每个位置的字符概率分布。Connectionist Temporal Classification (CTC) 是一种在没有对齐输入输出字符的情况下通过输入预测输出的算法。在 CTC 中，输入序列 $X = [x_1, x_2, \dots, x_t]$ ，输出序列 $Y = [y_1, y_2, \dots, y_u]$ ，其中输出序列的长度 $u \leq$ 输入序列的长度 t 。CTC 算法会计算每个时间步（信息帧）的预测字符的概率分布 $p_t = (a_t | X)$ ，进而计算出在给定完整的序列 X 时，每种 Y 序列的概率 $p(Y | X)$ 。在本项目中，CTC 的输入是来自 RNN 输出的图像特征序列，数据集中的算式共用 16 种字符。为应对输出为空的特征数据帧和连续重复的字符序列，CTC 会增加一个表示空的类别 ϵ ，即每个时间步的类别集合为 $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, +, -, *, (,), =, \epsilon\}$ 。得到输出的序列后，合并连续重复的字符、去除表示空的 ϵ 即可得到真实的输出序列。如输出序列 “4++3-11 $\epsilon\epsilon$ ===== ϵ 6” 进行上述操作后变成 “4+3-11=6”。

基准指标

Baoguang Shi et al. 在其团队构建的 CRNN 网络中，在无词典状态下，对 ICDAR 2013(IC03)数据集的识别精度为 86.7%，在 Street View Text(SVT)数据集中的识别准确率为 89.4%[5]。

本项目准确率需要高于 99%。

III. 方法

数据预处理

数据集中的图像尺寸一致，因此不需要对图像尺寸做处理。数据集中的图像是彩色图像，考虑到色彩信息对字符识别帮助不大，因此需要对图像进行灰度处理。灰度处理后，原先的 3 个色彩通道变成一个像素值通道，能减少算法的计算量。灰度处理后的图像像素值是位于

[0, 255]区间内的强度值，并将强度值缩放到[0, 1]的区间中，这有助于模型在训练过程中收敛。

数据集中给出的图像算式标注是存储在 csv 文件中的字符串，在进行监督学习时需要
对这种数据进行重新处理，对于多类分类问题一般采用独热编码。 本项目中序列编码要输入到 TensorFlow 中的 CTC 损失函数体中，采用的是另外一种编码，即将算式序列中的每个
字符对应到一个代表字符的整数，如将字符串中的 ‘0’ 编码成序列中的整数 0，算式 ‘1+1=2’
会被编码成[1, 10, 1, 2]。其实这种序列编码在输入 CTC 损失函数中，TensorFlow 还是会将
这种编码转换成独热编码进行运算。

执行过程

本项目采用 Keras 框架，使用 TensorFlow 做后端。搭建了 CNN+LSTM+CTC 的网络结构。

表格 1 模型结构和参数。k、s、p 分别代表卷积核、跨度、填充，n_sample 是输入图像的数量

层操作	参数	输出维度
输入图像	-	n_sample*64*300*1
卷积	k:32@3*3,s:1,p:same	n_sample*64*300*32
最大池化	Window:2*2,p:same	n_sample*32*150*32
卷积	k:64@3*3,s:1,p:same	n_sample*32*150*64
最大池化	Window:2*2,s:2	n_sample*16*75*64
卷积	k:128@3*3,s:1,p:same	n_sample*16*75*128
最大池化	Window:2*2,s:2,p:same	n_sample*8*38*128
卷积	k:256@3*3,s:1,p:same	n_sample*8*38*256
批标准化		
卷积	k:512@3*3,s:1,p:same	n_sample*8*38*512
批标准化		
最大池化	Window:1*2,s:2,p:same	n_sample*4*19*512
Dropout	rate:0.7	
维度置换	(2,1,3)	n_sample*19*4*512
特征到序列	(0,1*2)	n_sample*19*2048
双向 LSTM	units:64	n_sample*19*128
双向 LSTM	units:64	n_sample*19*128
全连接	units:17	n_sample*19*17

表格 1 中显示了模型 CNN+RNN 部分的网络层级、参数配置和维度输出。维度置换层之前是 CNN 部分。CNN 部分采用了 5 个卷积层，每个卷积层的卷积核数量逐步增加，第五层卷积有 512 个卷积核。最后两个卷积层后都添加了批标准化层。维度置换层之后是 LSTM 部分，采用了支持 GPU 计算的 CuDNNLSTM 结构，并配合 Keras 架构中的 Bidirectional 封装器实现双向 LSTM 的结构。维度置换和特征到序列层的作用是将 CNN 获得的图像特征转化成输入

LSTM 的时间-特征序列。

表格 1 中最后一层是全连接层，它输出序列在每个时间步的字符分类序列。这个序列被输入到 CTC 中。CTC 还有其它 3 个输入：图像真实标签、输入到 CTC 中的序列长度和真实标签的序列长度。CTC 的输出是其损失函数值。

整个网络使用 CTC-loss 做损失函数进行统一训练。而模型的序列输出是在 RNN 网络部分最末端的全连接层，全连接层使用 softmax 做激活函数，输出为每个输入图像在每个序列步中所有分类类别的概率分布。

需要对模型输出的序列做解码操作。我在本项目中使用正则表达式将模型输出序列中的多余空值和重复项去除，得到预测的标签字符串。将改字符串与原始标签中的比较即可判断预测是否正确。例如，模型输出的一个序列是：4++3-11===== 6，解码后的算式序列是：4+3-1=6。

模型约有 2220000 个训练参数。

完善

模型构建完成后，初次训练了 20 个周期。训练结果如下图所示。其中测试集上的损失值在前 5 个周期就实现了快速收敛，但在验证集上的损失值在训练集进行到第 7~10 周期时出现明显的震荡，且在后期测试集上的损失有增加的趋势。模型在验证集上的准确率在训练中期出现震荡变化，在后期则出现下降。这说明初期的模型出现了过拟合。

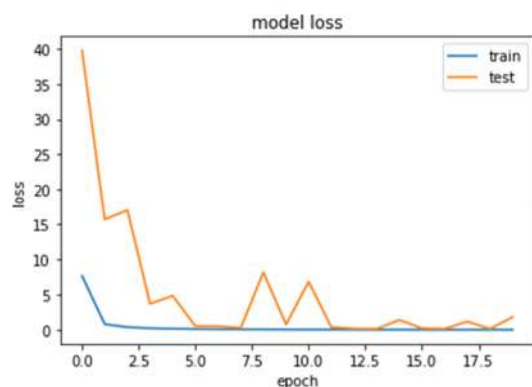


图 8 原始模型损失函数值

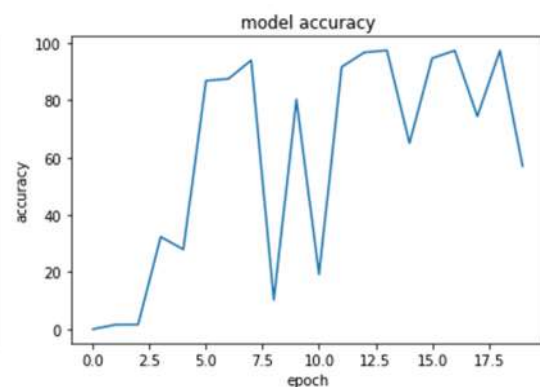


图 9 原始模型准确率

为了应对过拟合，我在 CNN 网络的末尾添加了 Dropout 层，并对丢失率在 0.3、0.5、0.7 三个参数上分别进行了 20 个周期的训练，其他参数不变。结果如表格 1 所示。表格中还比较了随机梯度优化器中不同学习率下的模型表现。

表格 2 不同模型方案 20 个周期的训练结果

方案	验证集准确率	测试集准确率
初始模型	97.32%	68.49%
Dropout (ratio=0.3, SGD lr=0.01)	98.15%	98.09%
Dropout (ratio=0.5, SGD lr=0.01)	97.15%	98.63%
Dropout (ratio=0.7, SGD lr=0.01)	98.13%	98.46%
优化器 (SGD, lr=0.02, ratio=0.7)	98.46%	98.31%
优化器 (SGD, lr=0.05, ratio=0.7)	98.04	98.15%

表格 1 中的参数是在 20 个周期结束后得到的训练结果。可以看出添加了 Dropout 层后模型在测试集上的准确率有很大进步，而在丢失率为 0.5 时得到测试集上最佳的 98.63% 准确率。优化器方面学习率在 0.01 时表现最佳。

最后选取两个在 20 个周期内的表现最好的两组模型进行更长周期的训练。设置总训练周期为 100，并使用早期停止回调函数，使模型在 10 个周期内没有训练提升的情况下终止模型的训练。

表格 3 最终模型的训练结果

方案	验证集准	测试集	早期停止周期
Dropout(ratio=0.5),SGD(lr=0.01)	99.16%	99.04%	79
Dropout(ratio=0.7),SGD(lr=0.01)	98.61%	99.03%	46

两个最终模型在训练至第 79、46 个周期后分别由早期停止回调函数触发停止。

IV. 结果

模型的评价与验证

由表格 3 所示，最终的两个模型在经过足够长时间的训练后在测试集上的准确率均超过 99%。图 11 显示了模型 (Dropout (ratio=0.7), SGD (lr=0.01)) 在训练过程中在训练集和验证集上的损失函数值变化。可以看到模型损失在前 10 个周期就趋于收敛，而最终会收敛于接近零。

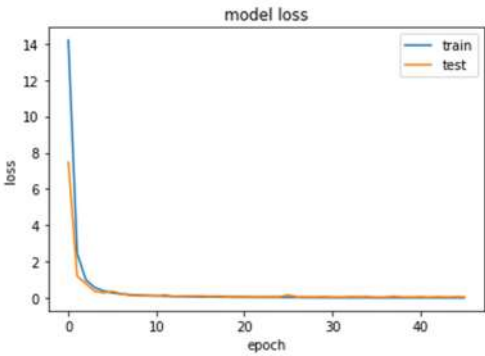


图 10 最终模型训练的验证集 loss

图 11、12 显示的是训练过程中在验证集上的准确率变化。同损失函数一样，准确率也在前 10 个周期就趋于收敛，但在 20 个周期之后模型收敛速度变得很慢。模型训练时每个周期平均耗费约 75 秒，最终保存的模型的大小约为 25MB。

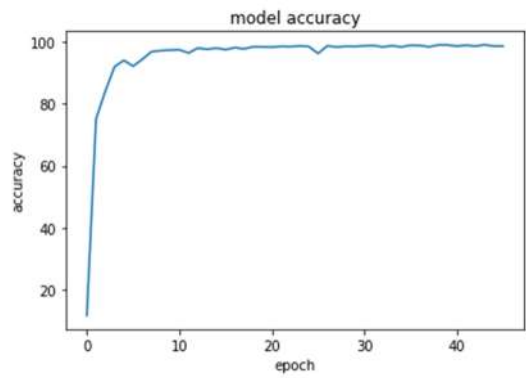


图 11 最终模型在测试集上的准确率
(Dropout ratio=0.7)

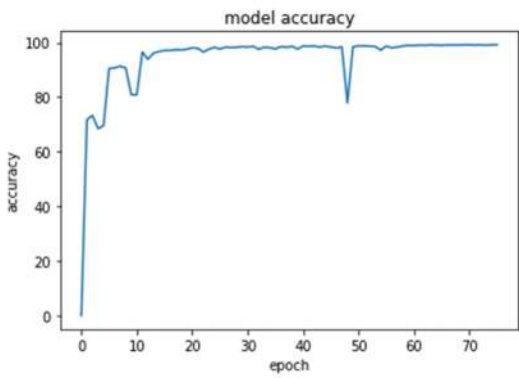


图 12 最终模型在测试集上的准确率
(Dropout ratio=0.5)

V. 项目结论

结果可视化

图 13 是最终模型的结构图。

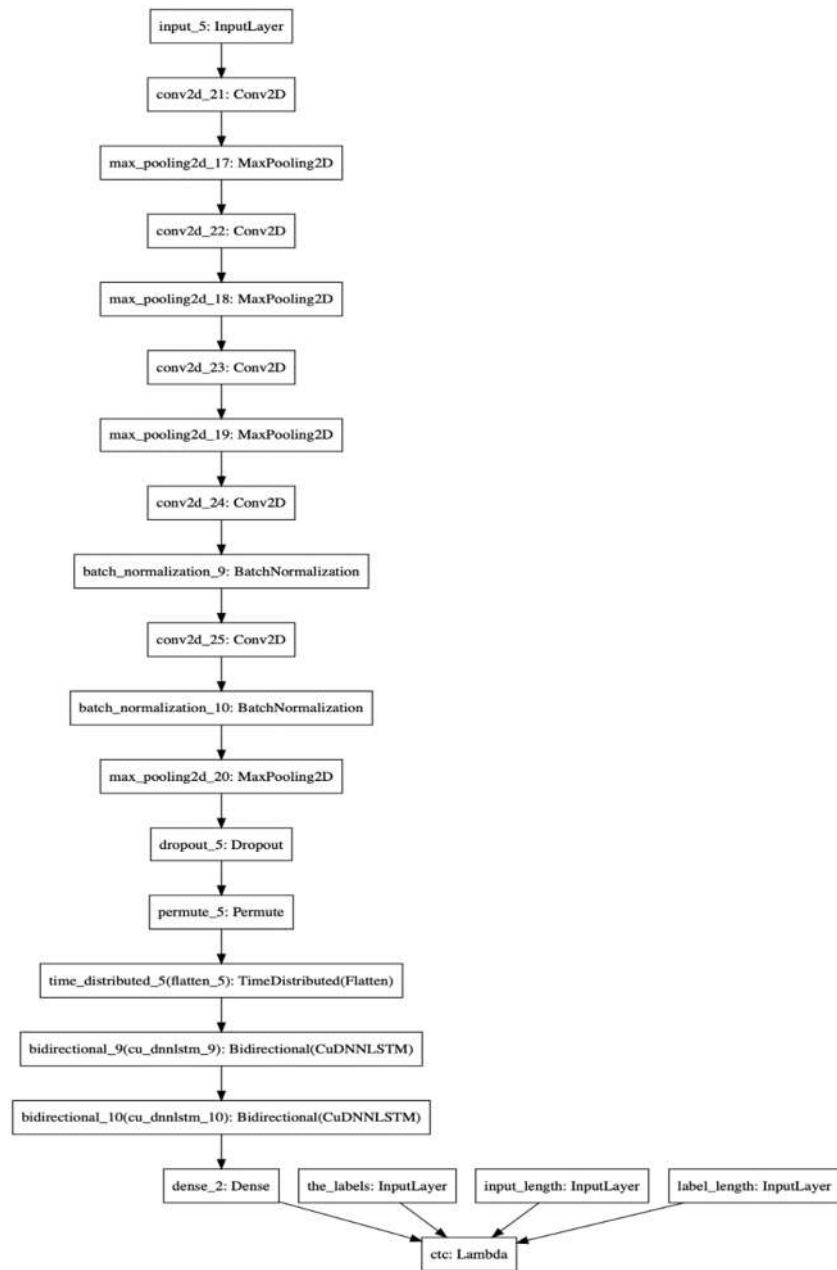



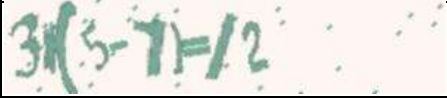


图 13 最终模型结构

这里选取了测试集的几个样本，展示了其序列输出和解码结果。

原始图像	输出序列	解码序列
	4++3-11 ===== 6	4+3-1=6
	8*((00++9)=====7 2	8*(0+9)=72
	6-(99-0)) ===== - 3	6-(9-0)=-3
	3*(5-11) =====1 2	3*(5-1)=1

经过对比发现，在测试集中模型识别错误的算式中，序列长度预测错误的占比 27.84%。

模型中字符识别较多的是 1 和 7 相互识别错误。

对项目的思考

本项目构建了自己的 CRNN 网络，使用 CNN 提取图像特征，图像特征做序列转换后由 RNN 网络利用其上下文信息的能力做序列预测。模型的重点是利用 CTC-loss 作为损失函数进行训练。

在调参过程中，由于网络模型搭建、优化器选择、各种参数设置等众多因素都会影响到模型的性能表现，从中选出最优模型需要仔细记录、对比每个变化对模型指标的影响，这个过程会比较耗时。首先选定一个模型结构，在模型结构基础上调节参数，如果通过调节参数难以达到指标，可以考虑改善模型结构。在本项目中设定 Dropout 层的比例时，首先设置一个较小的比例，训练模型并查看对结果的影响，然后再把参数提高，尝试不同的方案。对于优化器的选择，先使用默认参数测试效果，并在此基础上浮动参数进行训练和比较。我也参考了 Keras 的示例项目上使用的参数。

在训练模型的过程中发现，网络构建好之后，初期选用的参数能够很容易实现损失值的快速收敛，但是训练后期模型提升很慢，准确率收敛在 95%附近。本项目的难点也在于此，如何优化网络和调参，使模型的准确率最终收敛在超过 99%的地方。

需要作出的改进

本项目在模型进行训练时并没有使用数据增强的方法来扩充数据集。因此为进一步提高模型的表现，可以使用数据增强尝试训练模型。除此之外，从最终模型在训练过程中的准确率变化上看，模型并没有出现过拟合的现象。因此在算力和时间充足的情况下，可以尝试增加模型的复杂度。比如增加卷积层的层数、卷积核的个数、RNN 的隐藏层等。

参考资料

- [1] <https://distill.pub/2017/ctc/Sequence>, Modeling With CTC
- [2] <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> Understanding LSTM Networks
- [3] <http://karpathy.github.io/2015/05/21/rnn-effectiveness/> The Unreasonable Effectiveness of Recurrent Neural Networks
- [4] <http://yann.lecun.com/exdb/lenet/index.html> LeNet-5, convolutional neural networks
- [5] Baoguang Shi, Xiang Bai and Cong Yao. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. 2015
- [6] Dan Ciresan, Ueli Meier, Jurgen Schmidhuber. Multi-column Deep Neural Networks for Image Classification.
- [7] <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/> An Intuitive Explanation of Convolutional Neural Networks