机器学习工程师纳米学位毕业项目
算式识别(挑战项目)

丘海华

2019 年 5 月 28 日

目录

1 定义 1
1.1 项目概述 1
1.2 问题陈述 1
1.3 解决方案 2
1.4 评价指标 3
2 分析 3
2.1 数据可视化 3
2.2 算法和技术 5
2.2.1 神经网络 5
2.2.2 卷积神经网络 5
2.2.3 长短期记忆网络 7
2.2.4 CTC 损失函数 8
2.2.5 项目技术 9
3 具体方法 10
3.1 数据预处理 10
3.1.1 数据集读取 10
3.1.2 TFRecord 文件转换 10
3.1.3 算式字符串的稀疏矩阵转换 11
3.2 实现 11
3.2.1 卷积层 13
3.2.2 循环层 14
3.2.3 转录层 15
3.3 改进 15
3.3.1 正则化及随机舍弃 15
3.3.2 早期停止 16
3.3.3 学习率指数衰减 16
3.3.4 模型结构微调 16
4 结果 17
4.1 模型评价与验证 17
4.2 结果分析 19
5 结论 20
5.1 总结 20
5.2 后续改进 21
参考文献 21

1 定义

1.1 项目概述

本项目来源于优达学城的机器学习纳米学位的毕业项目—算式识别(挑战项目)。本项目要求使用深度学习的技术实现对各种图片的四则运算算式进行识别输出算式的内容，并且训练得到的深度模型可以部署在 app 或者服务器上，提供光学字符识别(OCR)算式的服务。

算式识别的任务是典型 OCR 应用的一种，可以通过微调模型来实现 OCR 其他应用，比如：身份证号码 OCR 识别、汉字 OCR 识别等。因此，该项目的实现模型及其变种具有很大的应用领域。

1.2 问题陈述

项目使用的数据集来自纳米项目说明中提供的算式数据集。此数据集包含 10 万张图片，每张图里面都有一个算式。

算式训练集如图 1 所示，具有以下特征：(1)可能包含+、-、*三种运算符；(2)可能包含一对括号，也可能不包含括号；(3)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。(4)算式字符串也就是标签具有任意的长度。(5)算式的图片可能出现若干干扰像素点。

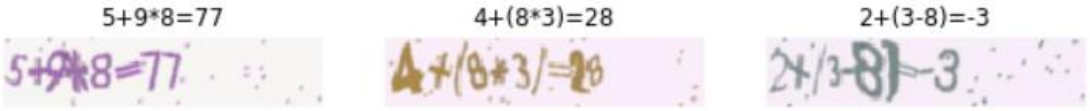

图 1　算式图片及字符串标签

项目的任务是识别图像中的算式，输入元素是图像的所有像素，输出元素是由算式字符构成的字符串。因此，从监督学习来看，算式字符串就是样本的标签，而不同于一般的单个字符的标签。

1.3 解决方案

我们可以把算式识别步骤大概分为两步：(1)识别图像中每个字符；(2)把每个字符按识别位置顺序连接起来构成算式字符串。

一种直观的方法是先圈定图像中每个字符的 ROI 区域，然后把 ROI 区域的图像分割出来传递给卷积神经网络进行字符的分类，最后把分类后的字符按识别位置顺序连接起来构成算式字符串[1,2]。该方法的效果直接受到 ROI 区域划分的质量影响，并且 ROI 区域的划分增加了模型训练的复杂度[1,2]。

另一种更通用的方法是结合卷积神经网络、长短期记忆网络和 CTC 损失函数构成端到端可训练的神经网络[3]。图片经由卷积神经网络提取长条形状的特征向量，并经由长短期

记忆网络输出具有允许冗余空格、字符项的字符串，最后被 CTC 损失处理去掉这些荣誉项，生成最终的字符串。

本项目采用了第二种方法，该方法的好处在于采用端到端的模型结构，简化了训练流程，并适用于不等长的序列图片。

1.4 评价指标

对于算式识别任务来说，只有图片的整个算式字符串都被识别正确了才被认为识别正确，而只有部分算式字符识别正确则算错误。

这里把识别图片的准确率(Accuracy)作为该项目的评价指标，准确率定义为正确识别的图片数除以用于识别的图片总数。

2 分析

2.1 数据可视化

算式数据集按照 8:1:1 的比例随机分割成训练集、验证集和测试集，并保存成 TFRecord 格式的文件。TFRecord 文件格式是 TensorFlow 官网推荐的适合用于内存读取大型数

据集的二进制格式。图 2 是训练集、验证集和测试集中具有的算式字符的统计分布。由于每个样本的算式字符串都有“=”字符，因此，训练集数量为 80000，验证集和测试集的数据为 10000。并且大体可以看出，训练集、验证集和测试集字符的统计分布几乎相同。但由于算式字符串的排列性和多样性，无法判断训练集、验证集和测试集的数据分布是否一致。

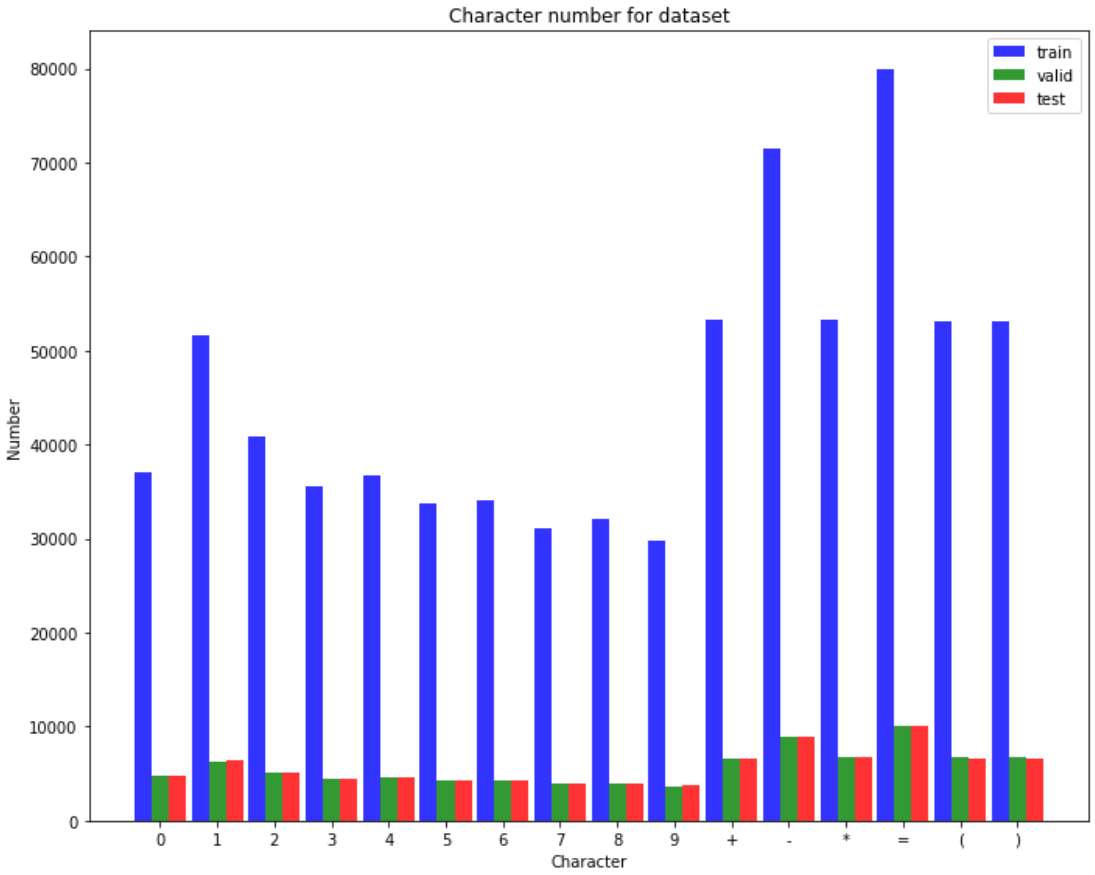

图 2　训练集、验证集和测试集的字符统计分布

2.2 算法和技术

2.2.1 神经网络

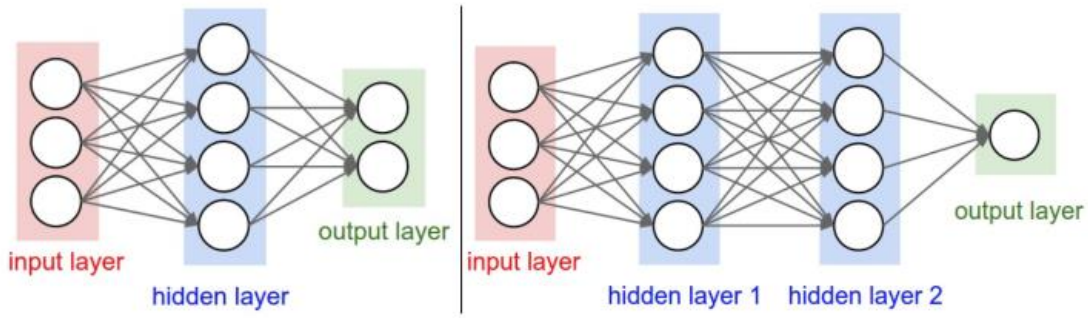

图 3　2 层(左)和 3 层(右)神经网络结构图

神经网络模型通过权重更新学习得到了一种输入数据与类别或者数值之间的映射。神经网络层级结构如图 3 所示，可以分为输入层，输出层和隐藏层三类网络层。

输入层(Input layer)：众多神经元(Neuron)接受大量非线性输入信息。输入的信息称为输入向量。

输出层(Output layer)：信息在神经元链接中传输、分析、权衡，形成输出结果。输出的信息称为输出向量。

隐藏层(Hidden layer)：简称“隐层”，是输入层和输出层之间众多神经元和链接组成的各个层面。如果有多个隐藏层，则意味着多个激活函数。

2.2.2 卷积神经网络

对图像(不同的数据窗口数据)和滤波矩阵(一组固定的权

重：因为每个神经元的多个权重固定，所以又可以看做一个恒定的滤波器 filter)做内积(逐个元素相乘再求和)的操作就是所谓的卷积操作，也是卷积神经网络的名字来源，简称 CNN。

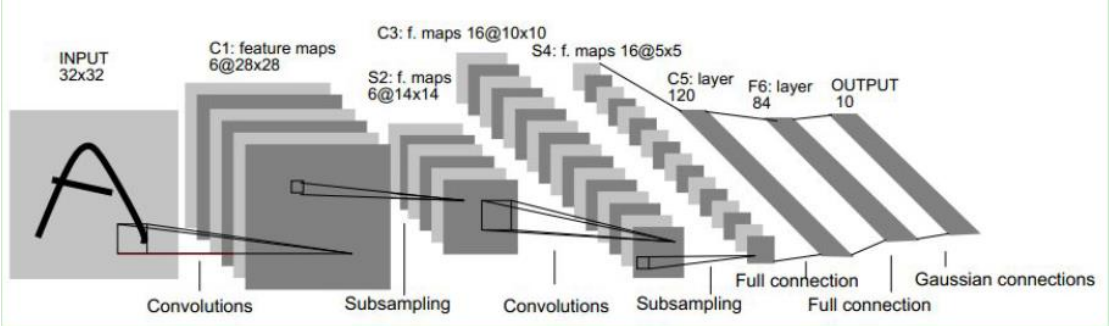

图 4　LeNet-5 结构的卷积神经网络

图 4 展示了 LeNet-5 结构的典型卷积神经网络，它主要有卷积层、池化层和全连接层构成。

卷积层：因为通过卷积运算我们可以提取出图像的特征，通过卷积运算可以使得原始信号的某些特征增强，并且降低噪声。

池化层：因为对图像进行下采样，可以减少数据处理量同时保留有用信息，采样可以混淆特征的具体位置，因为某个特征找出来之后，它的位置已经不重要了，我们只需要这个特征和其他特征的相对位置，可以应对形变和扭曲带来的同类物体的变化。

全连接层：采用 softmax 全连接，得到的激活值即卷积神经网络提取到的图片特征。

2.2.3 长短期记忆网络

长短期记忆网络即为 LSTM，是一种循环神经网络(RNN)，可以学习长期依赖问题。RNN 都具有一种重复神经网络模块的链式的形式。在标准的 RNN 中，这个重复的模块只有一个非常简单的结构，例如一个 tanh 层。

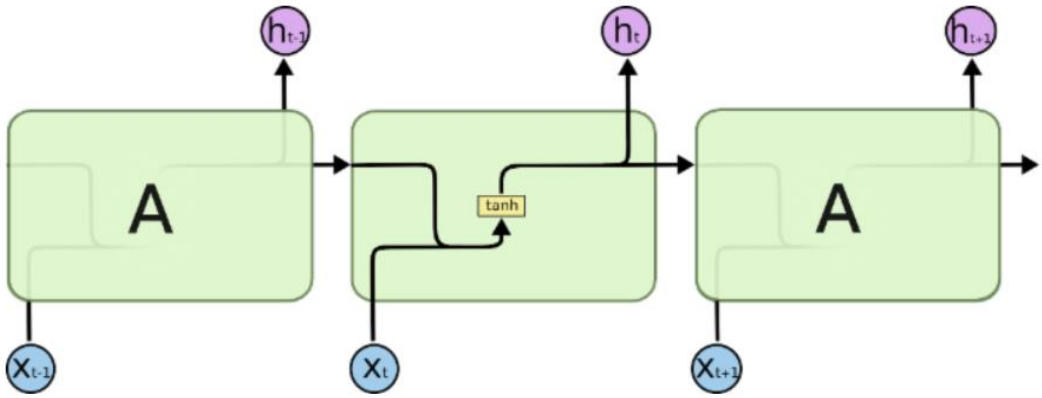

图 5　RNN 神经网络结构

如图 5 为标准的 RNN 神经网络结构，LSTM 则与此不同，其网络结构如图 6：

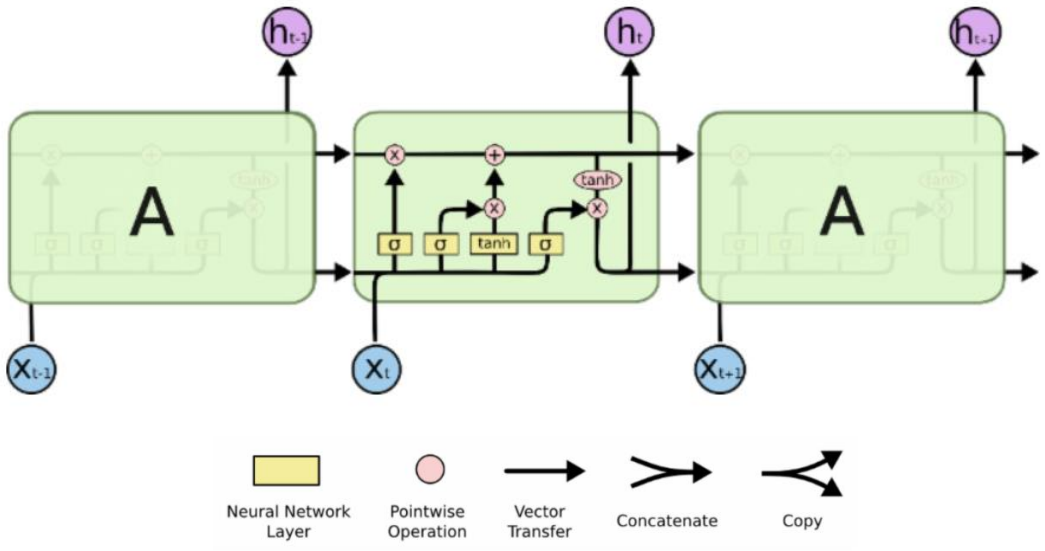

图 6　LSTM 神经网络结构

LSTM 通过精心设计的称作为“门”的结构来去除或者增加信息到细胞状态的能力。门是一种让信息选择式通过的方法。他们包含一个 sigmoid 神经网络层和一个 pointwise 乘法操作。LSTM 拥有三个门，来保护和控制细胞状态。

2.2.4 CTC 损失函数

按照每帧预测 $y=y_1,...,y_T$ 对标签序列 l 定义 CTC 概率，并忽略 l 中每个标签所在的位置。因此，当我们使用这种概率的负对数似然作为训练网络的 CTC 损失函数时[4]，我们只需要图像及其相应的标签序列，避免了标注单个字符位置的劳动。

CTC 条件概率的公式简要描述如下：输入是序列 $y=y_1,...,y_T$，其中 T 是序列长度。这里，每个 $y_t \in R^{|L'|}$ 是在集合 $L'=L^{\cup}$ 上的概率分布，其中 L 包含了任务中的所有标签(例如:所有英文字符)，以及由-表示的“空白”标签。序列到序列的映射函数 B 定义在序列 $\pi \in L'^{T}$ 上，其中 T 是长度。B 将 π 映射到 l 上，首先删除重复的标签，然后删除 blank。例如，B 将“–hh-e-l-ll-oo–”(-表示 blank)映射到“hello”。然后，条件概率被定义为由 B 映射到 l 上的所有 π 的概率之和:

$$p(\mathbf{l}|\mathbf{y}) = \sum_{\pi:\mathcal{B}(\pi)=\mathbf{l}} p(\pi|\mathbf{y})$$

使用 CTC 损失函数的训练过程，本质上是通过求梯度

p(l|y)调整 LSTM 的参数 w，使得输入样本给定时，p(l|y) 取得最大。

2.2.5 项目技术

项目使用了 Google 开源的 TensorFlow 框架构建卷积循环神经网络(CRNN)[3]来识别算式图片，所用的架构具有以下独有特性：

(1)与大多数现有的组件需要单独训练和协调的算法相比，它是端对端训练的。

(2)它自然地处理任意长度的序列，不涉及字符分割或水平尺度归一化。

(3)它产生了一个有效而小得多的模型，这对于现实世界的应用场景更为实用。

本项目采用了 Google 免费开源的 colab 研究工具来训练卷积循环神经网络。卷积网络的构建使用了 TensorFlow 低水平的 API 接口 tf.nn 以命名空间为单位进行层级的模块化构建。长短期记忆网络采用了 rnn.stack_bidirectional_dynamic_rnn 接口堆叠 2 层前后向 LSTM 网络，结合了上下文的记忆，具有更好的输出预测效果。最后，CTC 损失函数通过 TensorFlow 官方提供的 tf.nn.ctc_loss 来提供训练损失函数，而用

tf.nn.ctc_beam_search_decoder 来解码推理的结果。

3 具体方法

3.1 数据预处理

3.1.1 数据集读取

本项目把 10 万张 64*300 分辨率的算式图片转换成了 32*150 分辨率的图片，以匹配论文[3]提供的模型输入维度，并且像素的数据类型转换成 np.uint8。使用 sorted 方法对读取的图片名称按照数字顺序排序，以便于与对应数组位置的算式标签正确一一对应。

3.1.2 TFRecord 文件转换

项目的开始使用了自定义数据集类来读取算式图片和标签进行训练和推断，但存在内存溢出的问题。因此，采用了 tf.python_io.TFRecordWriter 把算式图片和标签保存为 TFRecord 文件，并用 tf.data.TFRecordDataset 类来读取保存的 TFRecord 文件进行训练和推断。

3.1.3 算式字符串的稀疏矩阵转换

由于 tf.nn.ctc_loss 方法形参 labels 要求是稀疏矩阵，因此使用了 util.py 文件中的 sparse_tuple_from 方法对算式标签进行了转换。其中，indices 表示二维 int64 的矩阵，代表非 0 的坐标点。values 是二维 tensor，代表 indices 位置的数据值。shape 代表稀疏矩阵的维度大小。

当然，在进行测试的时候，还得通过 util.py 文件中的 decode_sparse_tensor 方法解码稀疏矩阵来进行算式标签的比对。

3.2 实现

序号	层类型	结构参数
0	Input	32*150*3
1	Convolution	Maps:64;k:3*3;s:1;p:1
2	MaxPooling	Window:2*2;s:2
3	Convolution	Maps:128;k:3*3;s:1;p:1
4	MaxPooling	Window:2*2;s:2
5	Convolution★	Maps:256;k:3*3;s:1;p:1;dropout:0.5
6	Convolution★	Maps:256;k:3*3;s:1;p:1;dropout:0.5

7	Convolution★	Maps:256;k:3*3;s:1;p:1;dropout:0.8
8	MaxPooling	Window:2*1;s:2*1
9	Convolution	Maps:512;k:3*3;s:1;p:1
10	BatchNorm	
11	Convolution	Maps:512;k:3*3;s:1;p:1
12	BatchNorm	
13	MaxPooling	Window:2*1;s:2*1
14	Convolution★	Maps:512;k:2*2;s:3*1;p:0;dropout:0.5
15	Map-to-Sequence	
16	Bidirectional-LSTM	units:256;forget_bias=1
17	Bidirectional-LSTM★	units:256;forget_bias=1;dropout:0.4
18	Output	batchsize*maxtimesteps*numclasses
19	Transcription	

表 1　本项目所用的模型结构,带★号的层是与论文[3]结构有差异的地方

本项目采用了论文[3]提到的 CRNN 模型结构，如表 1 所示，由三部分组成，包括卷积层(0-15)，循环层(16-18)和转录层(19)。

3.2.1 卷积层

在 CRNN 模型中，通过采用标准 CNN 模型(去除全连接层)中的卷积层和最大池化层来构造卷积层的组件。这样的组件用于从输入图像中提取序列特征表示。在进入网络之前，所有的图像需要缩放到相同的高度。然后从卷积层组件产生的特征图中提取特征向量序列，这些特征向量序列作为循环层的输入。

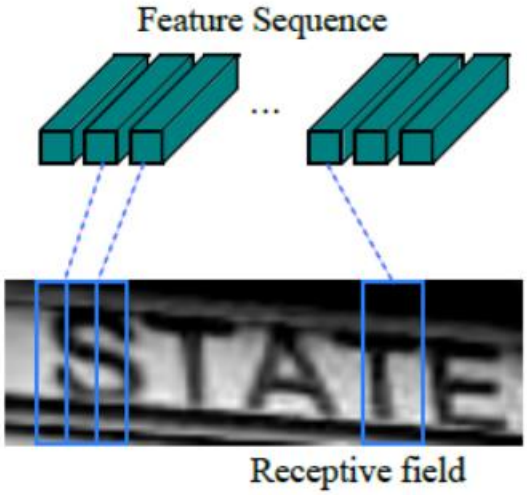

图 7　感受野

具体地，特征序列的每一个特征向量在特征图上按列从左到右生成。这意味着第 i 个特征向量是所有特征图第 i 列的连接。在我们的设置中每列的宽度固定为单个像素。特征图的每列对应于原始图像的一个矩形区域(称为感受野)，并且这些矩形区域与特征图上从左到右的相应列具有相同的顺序，如图 7 所示。提取的特征序列中的每一个向量关联输入图像的一个感受野，可认为是该区域的特征向量。

3.2.2 循环层

深度双向循环神经网络是建立在卷积层的尾部，作为循环层。循环层的优点是三重的。首先，RNN 具有很强的捕获序列内上下文信息的能力。其次，RNN 可以将误差差值反向传播到其输入，即卷积层，从而允许我们在统一的网络中共同训练循环层和卷积层。第三，RNN 能够从头到尾对任意长度的序列进行操作。

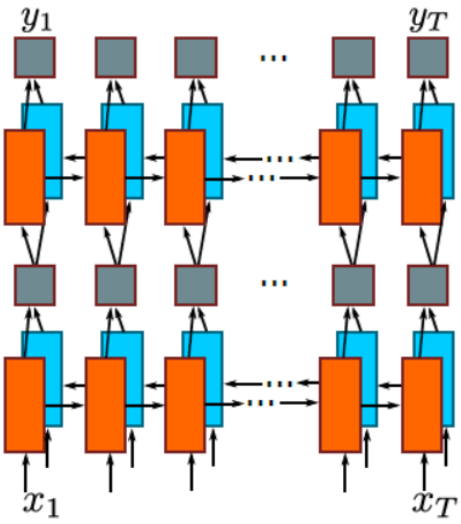

图 8　深度双向 LSTM 结构

然而，传统的 RNN 单元有梯度消失的问题，这限制了其可以存储的上下文范围，并给训练过程增加了负担。长短时记忆(LSTM)是一种专门设计用于解决这个问题的 RNN 单元。但是，LSTM 是定向的，它只使用过去的上下文。然而，在基于图像的序列中，两个方向的上下文是相互有用且互补的。因此，项目将两个 LSTM，一个向前和一个向后组合到一个双向 LSTM 中。此外，由于深层结构允许比浅层抽象更高层

次的抽象，项目堆叠了多个双向 LSTM。最后形成了图 8 所示的深度双向 LSTM 结构。

为了使循环层上的梯度能够反向传播到卷积层中，项目创建了一个称为“Map-to-Sequence”的自定义网络层，作为卷积层和循环层之间的桥梁。

3.2.3 转录层

本项目的转录层使用了 CTC 损失函数将 RNN 所做的每帧预测转换成标签序列，具体的原理可以参见 2.2.4 节。

3.3 改进

3.3.1 正则化及随机舍弃

正则化和随机丢弃(dropout)能够有效降低模型的过拟合程度。原始的模型是基于论文[3]的 CRNN 结构，此时在验证集的准确率只能到达 92%左右。因此，在模型层(5,7,14,17)采用了 dropout 方法使得模型在验证集的准确率能到达 95%左右。

3.3.2 早期停止

本项目采用早期停止的方法来降低模型的过拟合层度。每隔 1 个训练集的 epoch 测试验证集的准确率和每隔 10 个训练集的 epoch 测试训练集的准确率，来判断模型训练的早期停止时间，降低模型训练时间和过拟合程度。

3.3.3 学习率指数衰减

在训练模型时，采用了 tensorflow 提供 tf.train.exponential_decay()方法，该方法对学习率采用了指数衰减函数，具体的计算方法如下公式。

$$decayed\_learning\_rate = learning\_rate * decay\_rate^{\frac{global\_step}{decay\_steps}}$$

该方法的优点在于：1. 训练伊始可以使用较大学习率，以快速得到比较优的解。2. 后期通过逐步衰减后的学习率进行迭代训练，以使模型在训练后期更加稳定。

在采用了以上 3 种方法训练模型以后，模型在测试集的准确率可以达到 98.5%。

3.3.4 模型结构微调

由于项目说明对本项目的要求是准确率要大于 99%[5]，此

时还没有达到要求。在对数据增强和模型结构微调方法的实现难度进行评估以后，采用了模型结构微调方法来提高准确率。通过对原始模型增加了模型层(6)以后，模型在验证集的准确率可以达到 99%以上。采用这个方法的思路是因为在通过前面的方法后大幅度降低了过拟合程度，可以达到的验证集只有 98.5%，说明相对于 99%的目标来说，还是存在轻微的欠拟合问题。因此通过增加一层卷积层来提高模型复杂度来降低欠拟合程度，并进行 dropout 方法降低过拟合程度，使得模型在验证集的准确率可以达到 99%以上。

4 结果

4.1 模型评价与验证

训练模型时采用了 Adam 梯度下降算法，训练样本是 80000 张图片和对应标签，分为 800 个 batch，每个 batch 具有 100 个训练样本。训练的过程中使用了早期停止，最终模型经过 XXX 次权重更新，在验证集上的准确率超过 99%。从图 9 所示训练过程的损失函数曲线来看，整体趋势是：随着训练迭代次数的增加，损失函数值从 77.55 逐渐降低，并最终趋向于 0。

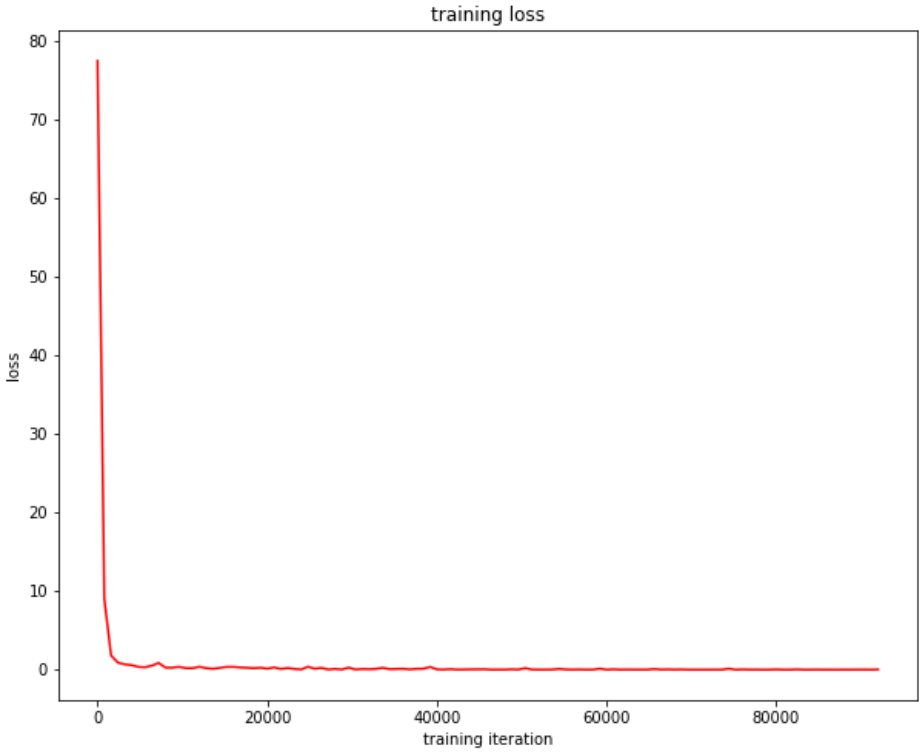

图 9　模型在训练时的损失函数曲线

另一方面，从图 10 所示的模型训练的准确率变化曲线来看，模型对训练集和验证集的识别准确率从刚开始的 0，不断上升直到趋于 100%。并且，相对对验证集的识别准确率，模型对训练集的识别准确率以更少的迭代次数达到项目要求的 99%。这说明了模型训练过程中存在不明显的过拟合现象，但最终对验证集的识别精度达到了项目要求。

注意：由于每隔 1 个 epoch 次数，保存一次验证集的准确率；而由于训练集具有 80000 个样本，所以为了降低训练时间，采取了每隔 10 个 epoch 次数才保存一次训练集的准确率的措施。因此，图像的 0 到 8000 迭代次数之间的曲线，

无法反映训练集准确率的变化情况，这里是采用了线性插值的方法来填补 0 到 8000 迭代次数之间的准确率值。而超过 8000 迭代次数以后，训练集的准确率趋于饱和，则可以反映出训练集准确率的变化情况。

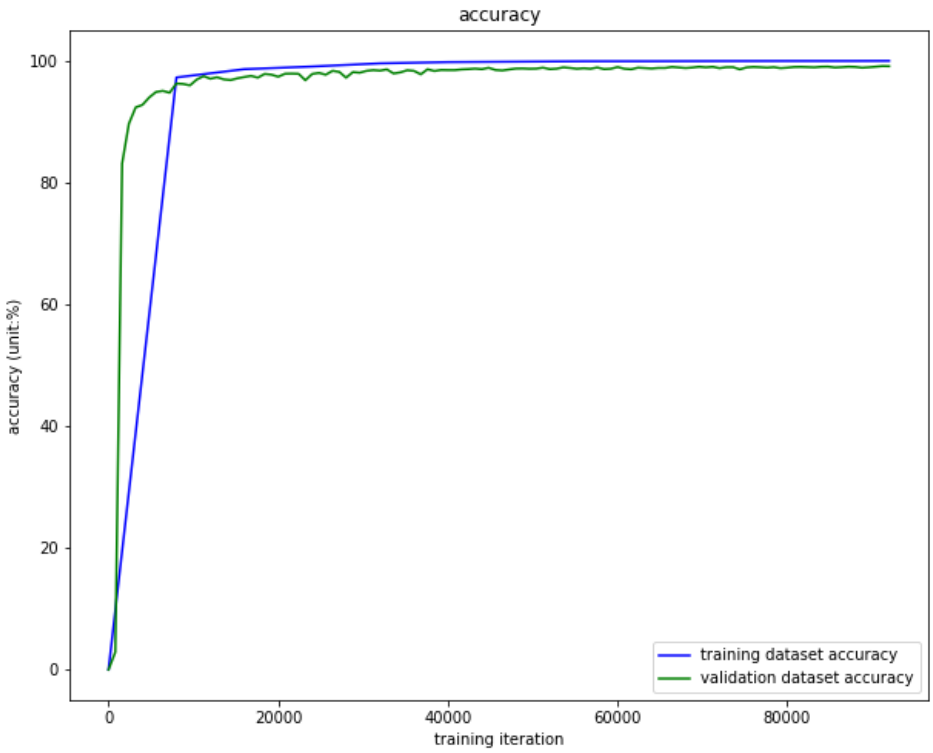

图 10　模型在训练时的准确率曲线

4.2 结果分析

为了保证模型的泛化性能和简化训练过程，在代码中设定只保存对测试集识别准确度超过 99%的模型。从逻辑上看，它具有比只对验证集识别准确度超过 99%的模型更好的泛化

性能。

从图 9 和图 10 的训练曲线看，曲线的整体趋势都符合人们的基本认知：即随着训练次数的增加，模型的损失函数值会越来越小，而对训练集和验证集的准确率都会越来越大。正常情况下，模型的训练对训练集准确率提升更加容易。并通过之前章节讨论的降低过拟合的措施，有效的提高了模型的泛化性能。

5 结论

5.1 总结

过去识别序列的图片需要通过把图片分割成一个个待分类的字符图片，增加了操作的复杂性。本项目采用了端到端可训练的神经网络模型，不仅使得模型的训练变得简单，而且适用于不同长度和类型的序列。通过正则化、随机舍弃、早期停止、学习率指数衰减和模型结构微调等方法，训练出来的模型在训练集、验证集和测试集的准确率都超过了 99%，符合项目的要求。

5.2 后续改进

1. 可以使用数据增强的方法来进一步增强模型的泛化性能，提高模型对未知数据的识别准确度。
2. 可以进一步完成项目推荐的要求：使用 Flask 框架把训练好的模型部署在服务器上供他人使用。
3. 可以用 keras API 来重构模型结构，能够简化模型构建的代码和优化训练的效果。

参考文献

[1] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[2] Wang, Tao, et al. "End-to-end text recognition with convolutional neural networks." Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). IEEE, 2012.

[3] Shi, Baoguang, Xiang Bai, and Cong Yao. "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition." IEEE transactions on pattern analysis and machine intelligence 39.11 (2016): 2298-2304.

[4] Graves, Alex, et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.

[5] https://github.com/udacity/cn-machine-learning/tree/master/mathematical_expression_recognition