机器学习纳米学位

毕业项目

Xuefeng Sun 优达学城
2018年7月22日

1. 问题的定义

1.1 项目概述

项目将从一个图片中正确识别算式表达式，并将其提取转化为字符串。算式验证码是非常常见的一种验证码，同时也是属于 OCR (Optical Character Recongnition)领域的一个问题。人工在识别图片时准确率低，并且需要耗费时间。引入机器学习，让机器自动的学习和识别能够减少人力成本，并且准确率高于人工识别。OCR 图像识别应用非常的广泛，比如文字提取，车牌号识别，身份信息提取等。
本项目将以算式验证码识别作为研究课题，所用的数据集由 Udacity 提供，共包含了 10 万张图片。这些图片将会作为模型的输入，训练模型使其能够正确的识别出图片中的算术式。

1.2 问题陈述

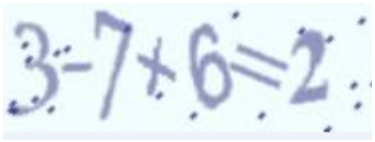
本项目将原始图片作为输入，输出为图片中的算式表达式。如图:

图片由数学表达式和噪点组成，每一个字符都有可能旋转或者和其相邻的字符粘连在一起。目前比较流行的识别的方法有分割法识别和深度学习识别。分割法，比较传统就是就是将每一个字符“扣”下来，再利用KNN/SVM 等这些机器学习算法进行单个字符的识别。例如：

深度学习识别是利用了卷积神经网络 CNN (Convolutional Neural Network) 提取图片特征和循环神经网络 RNN (Recurrent Neural Network) 处理图片中的序列自动的提取图片特征并且识别图片中的序列。通过这样的一个模型自动的识别和分析图片，让机器能够自我学习如何识别。_

1.3 评价指标

本项目将选择准确度作为评判标准，只有当所有算式字符都正确，才判为该识别图片被正确识别，反之识别错误。例如：



，只有输出为 “3-7+6=2” 是才能记为正确。

2. 分析

2.1 数据的探索

项目所用的数据集来源于：https://s3.cn-north-1.amazonaws.com.cn/static-documents/nd009/MLND+Capstone/Mathematical_Expression_Recognition_train.zip 整个数据集有10万张彩色图片和一张对应于每张图片的标签列表。每一张图片都包含一个算式。标签列表中与之文件名相对的则是该图片对应的算式表达式。例如：

标签列表

filename	label
train/0.jpg	(0+0)+9=9
train/1.jpg	9*8+6=78

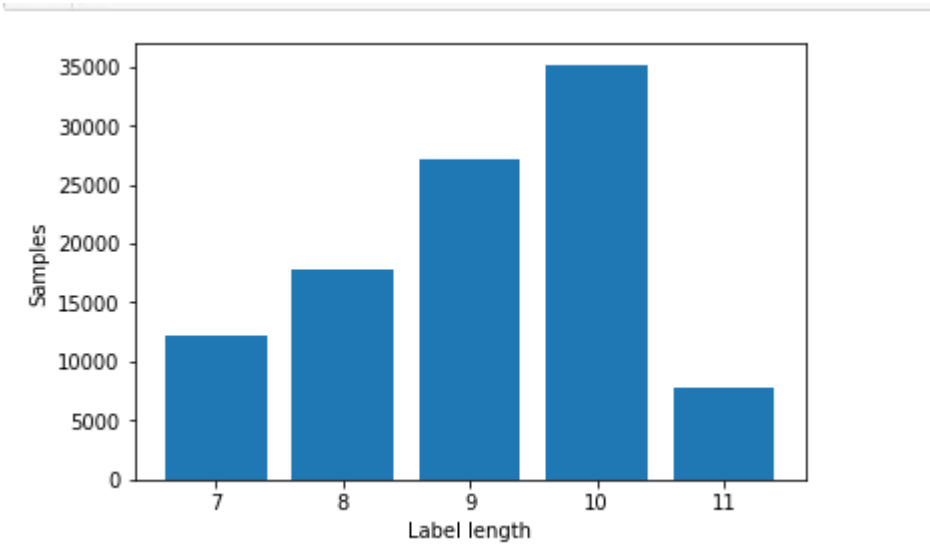
图片

0.jpg

算式中的字符为数字0-9，括号()，以及运算符*+- 和 =。每张图片的大小为 300 x 64 x 3, 算式序列长短不一，在 7-11 之间.
在该项目中将整个数据集（10万张图片）划分为训练集，用于训练模型参数；验证集，对模型进行验证；测试集，用于最终的模型测试所用数据。

2.2 探索性可视化

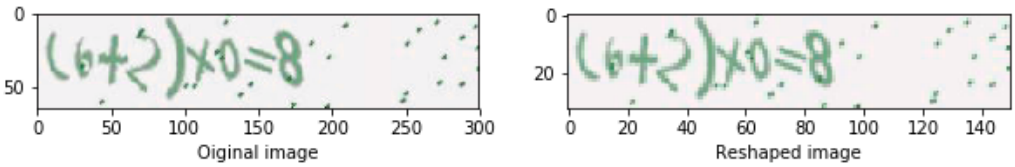
1. 分析样本



观察数据可以确定序列长度的范围为7-11。

2. 缩放图片

```
The original image size:  (64, 300, 3)
Reshape image size to:  (32, 150, 3)
Out[92]: <matplotlib.image.AxesImage at 0x7fcfd4e0d550>
```

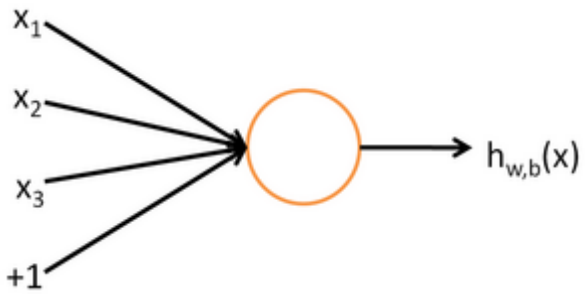


原始图片的大小是 64 x 300 x 3， 缩放后图片为 32 x 150 x 3。可以看到，经过缩放后的图片没有变模糊。缩放之后能够加快训练速度。

2.3 算法和技术

2.3.1 神经网络

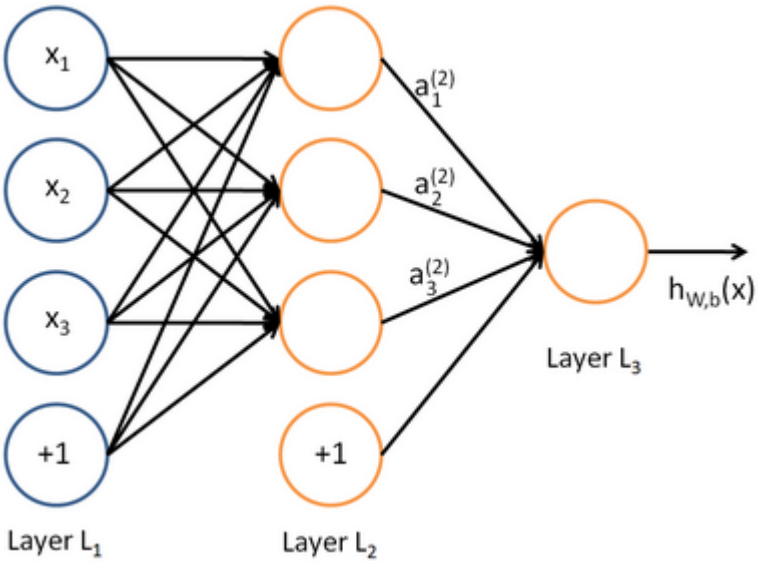
神经网络（Neural Networks）的概念在很早之前就被提出来，类比于生物神经网络，这里的神经网络也是由神经元组成，但和生物领域的神经网络有着本质的区别。下面将简单的介绍下这种技术，详情请参阅参考文献1。 一个单一的神经元从数学的角度来看，是将一个由一个非线性方程组成。输入值和权重的点积，再经过激活函数（一般为非线性方程，例如：sigmoid，relu，softmax...）得到输出。如下图：



这个 “神经元” 是一个以 $x1, x2, x3$ 及截距 $+1$为输入值的运算单元，其输出为 $h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$，其中函数 $f : \Re \mapsto \Re$ 就被称为“激活函数”。本例以sigmoid充当的便是这个角色。

$$f(z) = \frac{1}{1 + \exp(-z)}.$$

这个单一“神经元”的输入－输出映射关系其本质就是一个逻辑回归（logistic regression）。而所谓的神经网络就是将许多个这样的神经元连接起来组成一个网络。



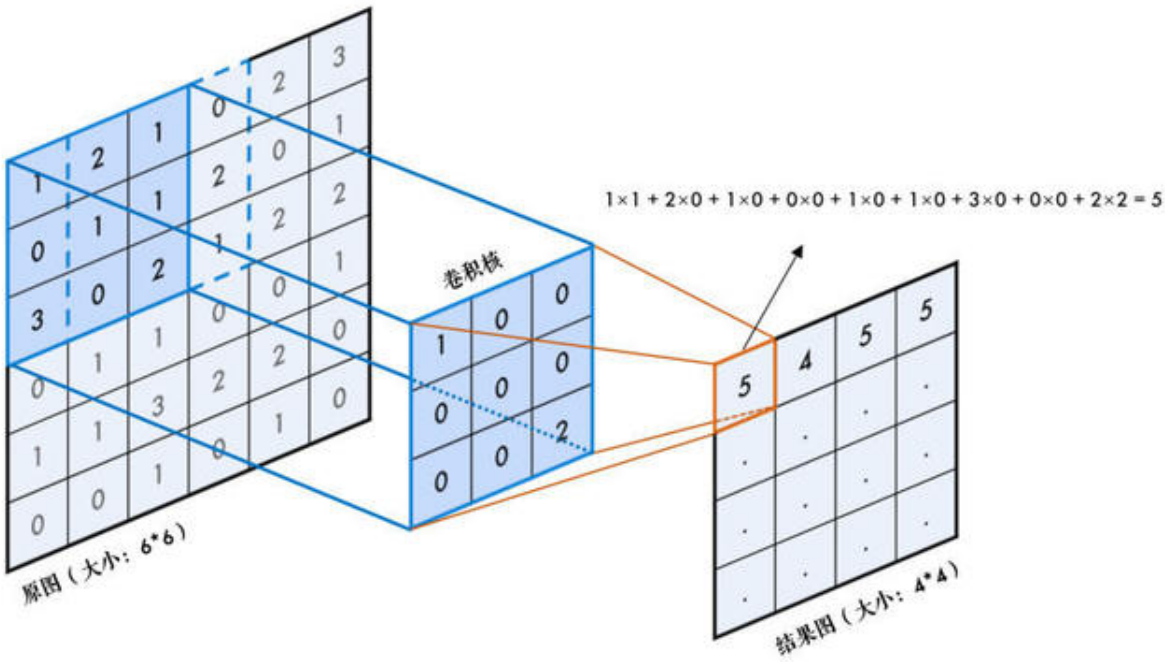
其对应的公式如下：

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

其中L1 是输入层，L2是隐藏层，L3 是输出层。神经网络中的隐藏层可以被扩展到2,3,4... 每个神经元的模型基本一样，不同之处在于其权重 w 和 b。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。之所以称之为后向传播，是因为在计算提的时候有最后一层向前利用导数的链式法则一层层计算参数。

2.3.2 卷积神经网络 CNN (Convolutional Neural Network)

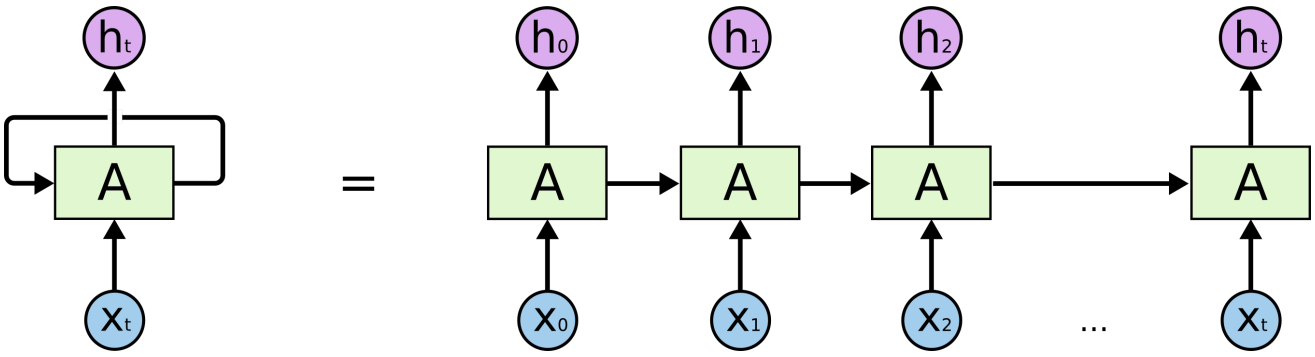
CNN 是神经网络的一种，已经被应用到了语音识别和图像识别等多个领域。之所有有如此的广泛的应用是因为它具有两个 “法宝” 感受野和参数共享。 所谓的感受野就是用来表示网络内部的不同位置的神经元对原图像的感受范围的大小。神经元感受野的值越大表示其能接触到的原始图像范围就越大，也意味着他可能蕴含更为全局、语义层次更高的特征；而值越小则表示其所包含的特征越趋向于局部和细节。因此感受野的值可以大致用来判断每一层的抽象层次。换句话说就是使用一个一定大小的 “滤波器” 对图片特征的提取，而这个“滤波器”也被称为卷积核（Kernel）。



如图，卷积核以一定的步长进行滑动，并与原图像素相乘得到输出。
可以看出不同的卷积核的值得到的输出是不一样的，那么提取的特征也就是不同的。而卷积核的值就是权重，一张图片被同一个卷积核过滤，所以权重也是一样的，这也就是参数共享。这意味着每一层提取的是相同的特征，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。
卷积神经网络利用卷积核既从不同的角度提取了图片的特征，挖掘了图片潜在的特征，又减少了神经网络的参数，提高了效率。

2.3.3 循环神经网络 RNN (Recurrent Neural Network)

RNN 主要解决的是序列问题，比如，音频，文本，视频等。这类数据的样本间存在顺序关系，每个样本和它之前的样本存在关联。比如说，在文本中，一个词和它前面的词是有关联的。RNN将状态在自身网络中循环传递，因此可以接受更广泛的时间序列结构输入。



RNN训练的过程比较困难，主要是因为隐藏层参数w在传递过程中需要多次相乘，这也就引来了梯度消失和梯度爆炸。GRU((Gated Recurrent Unit)) / LSTM (Long Short-Term Memory networks) 这两种技术采用记忆门和遗忘门来选择记住那些信息和忘记那些信息从而避免梯度消失和梯度爆炸。另外，双向的 RNN (bi-RNN) 模型不仅能够从前往后传播信息，也能够从后往前传播信息。使得在某一个时间点既能够了解 “过去的知识”，也能够了解 “未来的知识”。

2.3.4 CTC (Connectionist temporal classification)

CTC 是用来计算模型损失值的。在经过 RNN 之后，模型会输出一个概率序列（一般是label长度的两倍以上），这个输出会作为 CTC 算法的输入与标签对齐并计算 loss，在 Keras 中长度较短的序列，CTC 会用 -1 来进行文本对齐。CTC 的神奇之处在于不需要知道具体位置的情况下能够让模型自动收敛。

2.4 基准模型

项目要求测试集准确率为99%

3. 方法

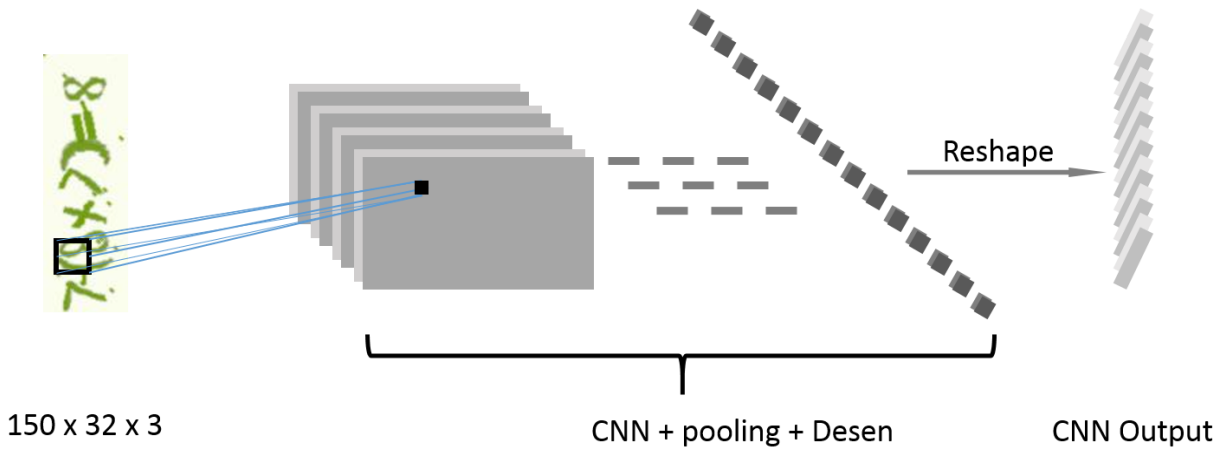
3.1 数据预处理

1. 数据集划分。有10万张图片作为项目的数据集。在开始训练中之前应该先预留出测试集和验证集。本次将10万张图片按照 (8:1:1) 的比例进行划分。其中训练集8万张，验证集合测试集各占1万张。
2. 定义生成器分批加载图片。考虑到同时加载8万张图片到内存比较消耗内存，所以采取生成器的方式在各个数据集中随机选取小批量图片进行训练。
3. 缩放图片，加快训练时间。原图片的大小是 300 x 64 x 3，将图片缩小到 150 x 32 x 3。图片没有明显的失真，并且能够加快训练时间。

3.2 执行过程

整个模型分为三个阶段特征提取，序列训练和计算loss。

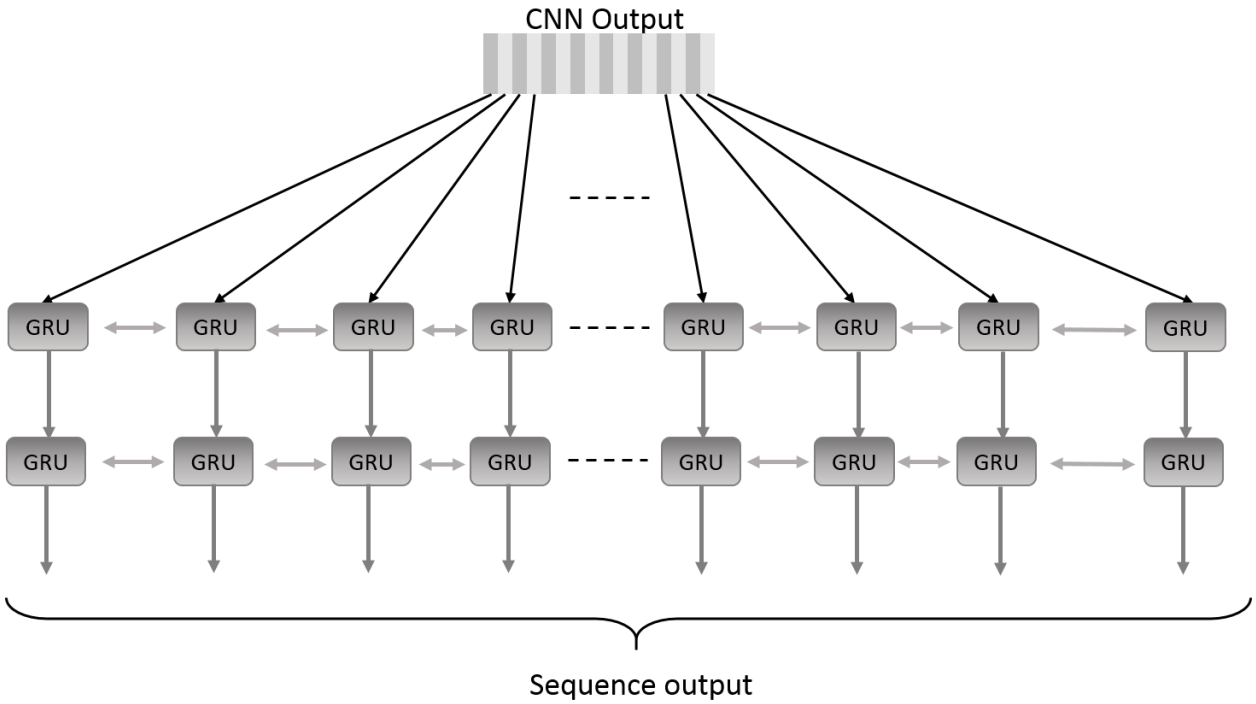
1. 图片特征提取
   在经过预处理之后，原始大小为（300,64,3）长度图片被缩放为（150,32,3）作为输入，在经过卷积层之后，输出提取的特征。如图：



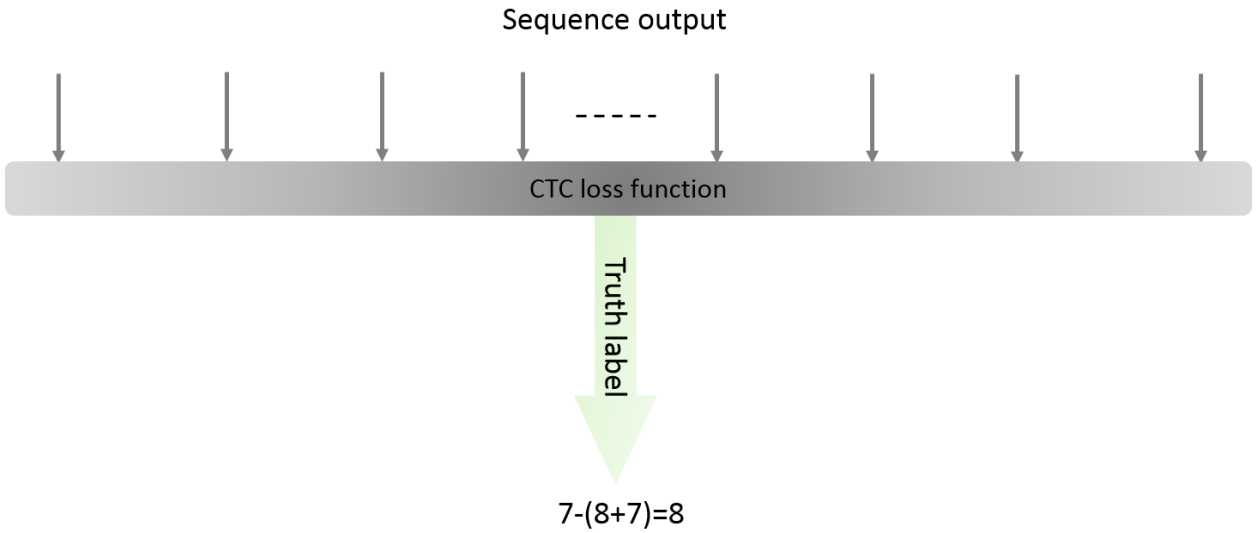
初始化CNN模型结构：

Type	Configuration
Input	image size: (150, 32, 3)
Convolution	kernels 64, size:(3, 3), s:1, p:"valid"
MaxPooling	size: (2, 2), s:2
BatchNormaliztion	
Convolution	kernels 128, size:(3, 3), s:1, p:"valid"
Convolution	kernels 128, size:(3, 3), s:1, p:"valid"
Convolution	kernels 128, size:(3, 3), s:1, p:"valid"
MaxPooling	size: (2, 2), s:2
BatchNormaliztion	
Convolution	kernels 256, size:(3, 3), s:1, p:"same"
Convolution	kernels 256, size:(3, 3), s:1, p:"same"
Convolution	kernels 256, size:(3, 3), s:1, p:"same"
Convolution	kernels 256, size:(3, 3), s:1, p:"same"
MaxPooling	size: (1, 2), s:1
BatchNormaliztion	
Convolution	kernels 512, size:(3, 3), s:1, p:"same"
Convolution	kernels 512, size:(2, 2), s:1, p:"same"
Convolution	kernels 512, size:(2, 2), s:1, p:"same"
MaxPooling	size: (1, 2), s:1
BatchNormaliztion	
Reshape	
Dense	128
Dropout	0.2
Bidirectional(GRU)	rnn_size:128
Bidirectional(GRU)	rnn_size:128
Dropout	0.3
Dense	n_class:(16 + 1)
ctc	
optimizer	adadelta

2. 将 CNN 提取好的特征送入双向 GRU



3. CTC 计算loss



CTC loss 已经被集成在了keras，ctc_batch_cost(y_true, y_pred, input_length, label_length)

- y_true：形如(samples，max_tring_length)的张量，包含标签的真值
- y_pred：形如(samples，time_steps，num_categories)的张量，包含预测值或输出的softmax值
- input_length：形如(samples，1)的张量，包含y_pred中每个batch的序列长
- label_length：形如(samples，1)的张量，包含y_true中每个batch的序列长

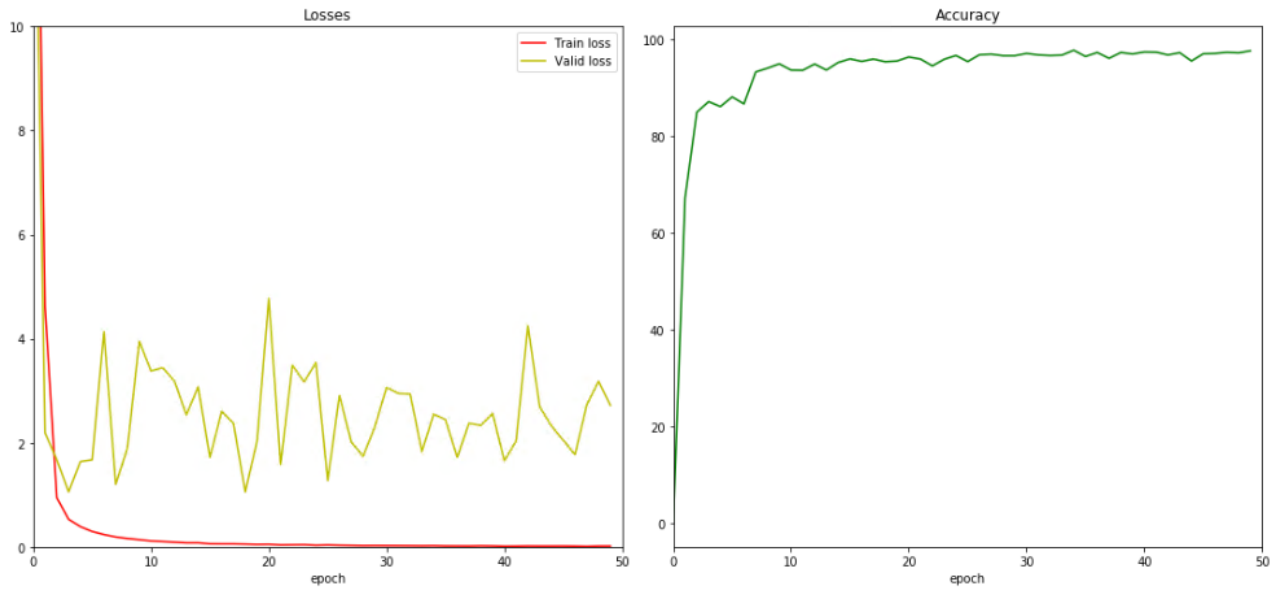
处理CTC loss 中 label_length 的时候花费了一点时间。起初从 ctc_decode 得知其长度不足最大长度的序列会用 -1 进行填充的。所以在处理 label_length 的时候也就用 -1 来填充，这样方便和y_pred 比较计算精确度。可是训练的时候提示label不能是负数。查找了资料才发现填充值需要用 n_class - 1 补足对齐。

3.5 完善

3.5.1 方案1

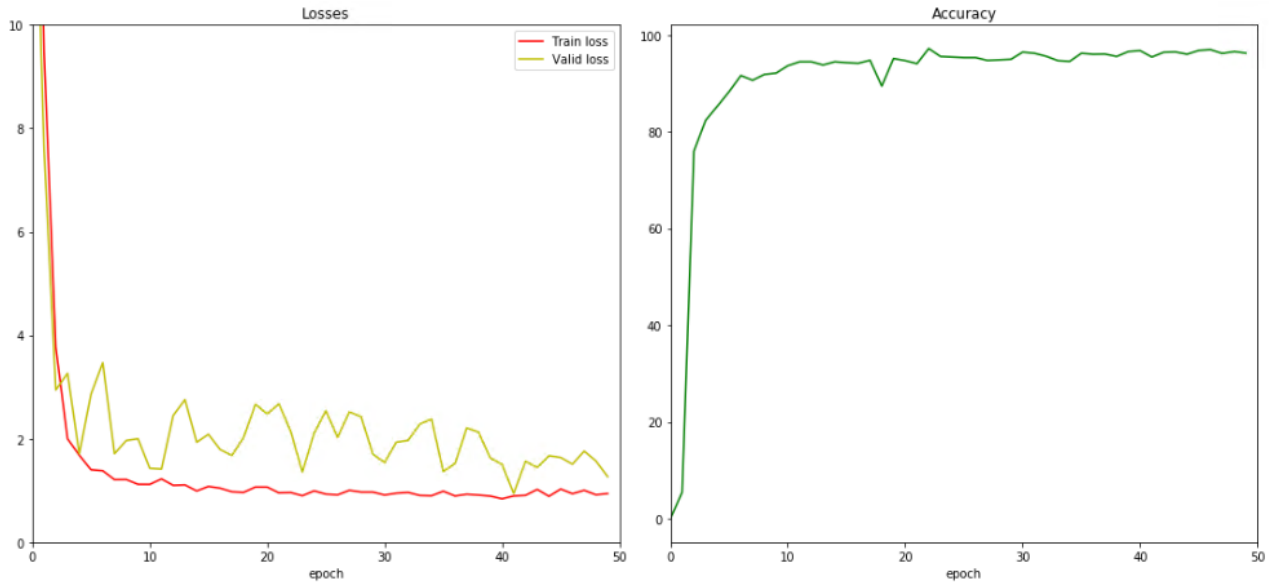
调试	验证集准确度	测试集准确度	训练时间
初始模型	97.62%	97.22%	3.26h
l2正则化（l2_rate=1e-5）	95.78%	95.74%	3.37h
l2正则化（l2_rate=1e-3）	93.40%	95.34%	3.48h
l2正则化（l2_rate=1e-8）	96.25%	96.21%	3.71h
Adam + ReduceLROnPlateau	89.67%	90.23%	3.70h

1. 训练3.2中初始模型结构，其得到的分数如表中第一行所示。在验证集和测试集有97%左右的准确度。loss 曲线如下图所示：



loss 曲线和val_loss 相差较大，有一些过拟合。

2. 表中第三行到第四行，加入了l2 正则化尝试防止过拟合。得到曲线如下图：



明显的改善了过拟合现象，但是新的问题是准确率却降低了。

3. 尝试着改变 optimizer 为 Adam, 并且使用 ReduceLROnPlateau 动态的改变learning rate，但从表最后一行可以看出，并没有改善，准确度反而将低了。

3.5.2 方案2

经过第一次尝试，感觉模型上有点问题。参考 “参考文档5”对模型进行了重构。

Type	Configuration
Input	image size: (150, 32, 3)
Convolution	kernels 64, size:(1, 1)
BatchNormalization	
Convolution	kernels 128, size:(3, 3), s:1, p:"valid"
Convolution	kernels 128, size:(3, 3), s:1, p:"valid"
Convolution	kernels 128, size:(3, 3), s:1, p:"valid"
BatchNormalization	
MaxPooling	size: (2, 2), s:2
Convolution	kernels 256, size:(3, 3), s:1, p:"valid"
Convolution	kernels 256, size:(3, 3), s:1, p:"valid"
Convolution	kernels 256, size:(3, 3), s:1, p:"same"
Convolution	kernels 256, size:(3, 3), s:1, p:"same"
BatchNormalization	
MaxPooling	size: (1, 2), s:2
Convolution	kernels 512, size:(3, 3), s:1, p:"same"
Convolution	kernels 512, size:(3, 3), s:1, p:"same"
Convolution	kernels 512, size:(2, 2), s:1, p:"same"
BatchNormalization	
MaxPooling	size: (1, 2), s:1
Reshape	
Dense	256
Dropout	0.2
Dense	128
Dropout	0.2
Bidirectional(GRU)	rnn_size:128
Bidirectional(GRU)	rnn_size:128
Dropout	0.3
Dense	n_class:(16 + 1)
ctc	
optimizer	Adam

调试	验证集准确度	测试集准确度	训练时间
初始模型	99.18%	99.15%	7.08h
缩放图片 (150, 32, 3) --> (170, 32, 3)	99.22%	99.38%	7.48h

1. 初始化模型取得了较好的成绩 99%，第一层使用 1x1 kernel 目的是加深图片channel 然后再使用BN归一化。同时在 CNN 的最后加了一层 Dense 和 Dropout。
2. 表中第二行对图片宽度进行了扩大，可见准确率有所提高。有尝试着在图片大小不变的情况下，更改前几层卷积层padding 参数为valid， 但这样显存总是溢出。所以最终决定将图片拓宽。

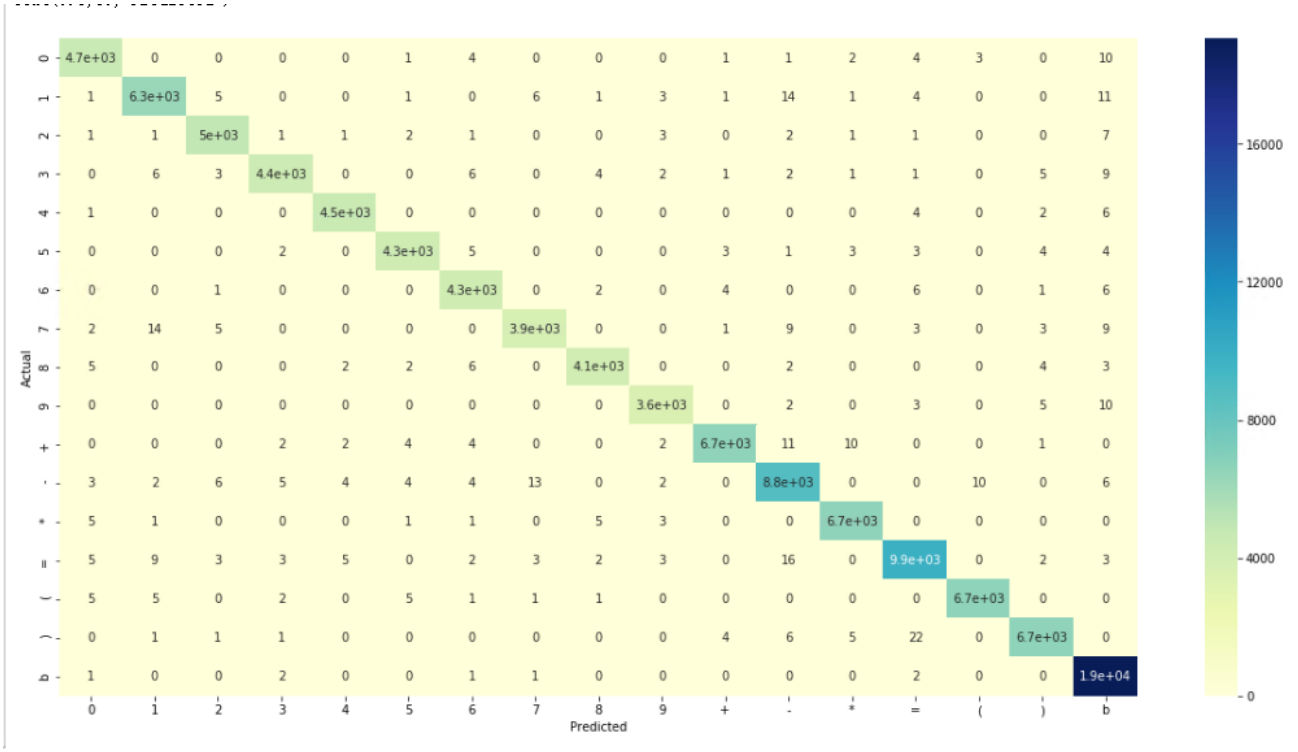
4. 结果

4.1 模型的评价与验证

如在 3.5 中描述。在经过两次调整模型之后，第二次模型在验证集合测试集上的精确度趋于稳定并且一直在99%以上。其中测试集有1万张且独立于其它两类数据，不会参与到训练之中。

最终选择方案2中最后一次调试作为了最终模型。对此模型重复训练了4次，都得到了相似的结果，该模型能够稳定重现。

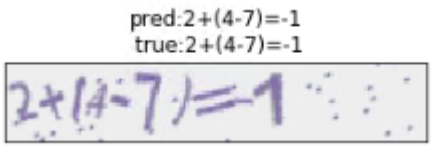
在测试集随机挑选了10000 张图片，并绘制混淆矩阵如下：



从混淆矩阵中看出")" 被 22 次错误识别为 “=”， 同样 “1” 和 “7”， “-” 和 “1”， 以及 “-” 和 “=”都是很容易被错误识别的。

4.2 合理性分析

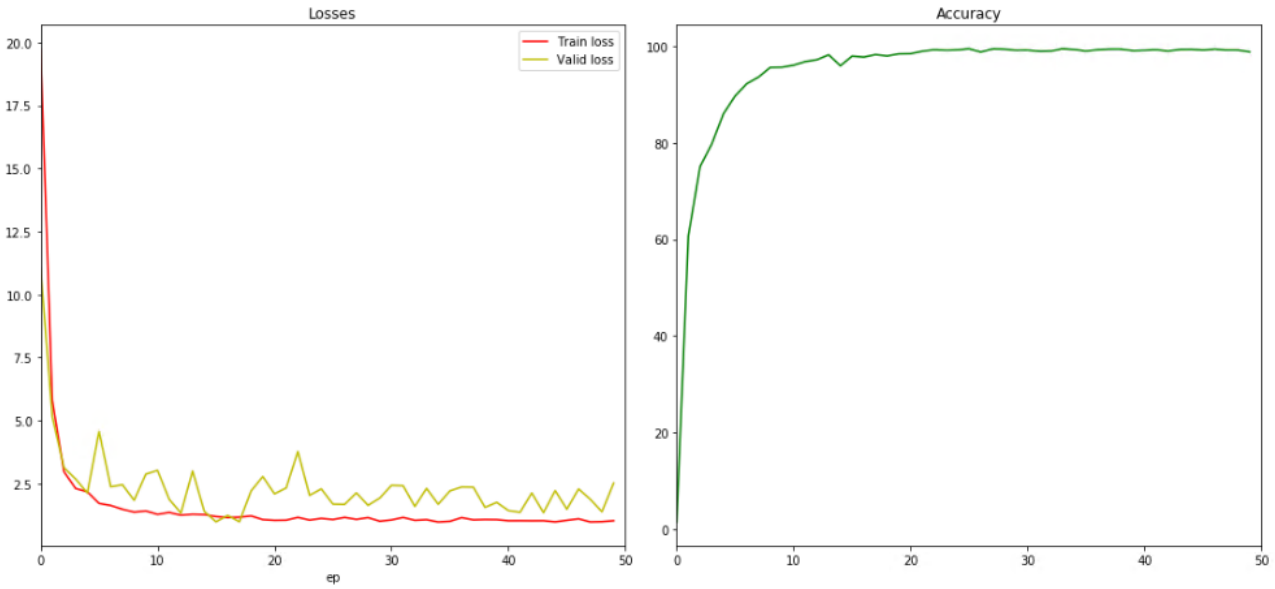
基准模型设定是 99% 的准确率，本模型准确率都在99以上。能够准确的识别图片中的算式表达式，即使像下图这种粘连的图片也能够准确的识别。



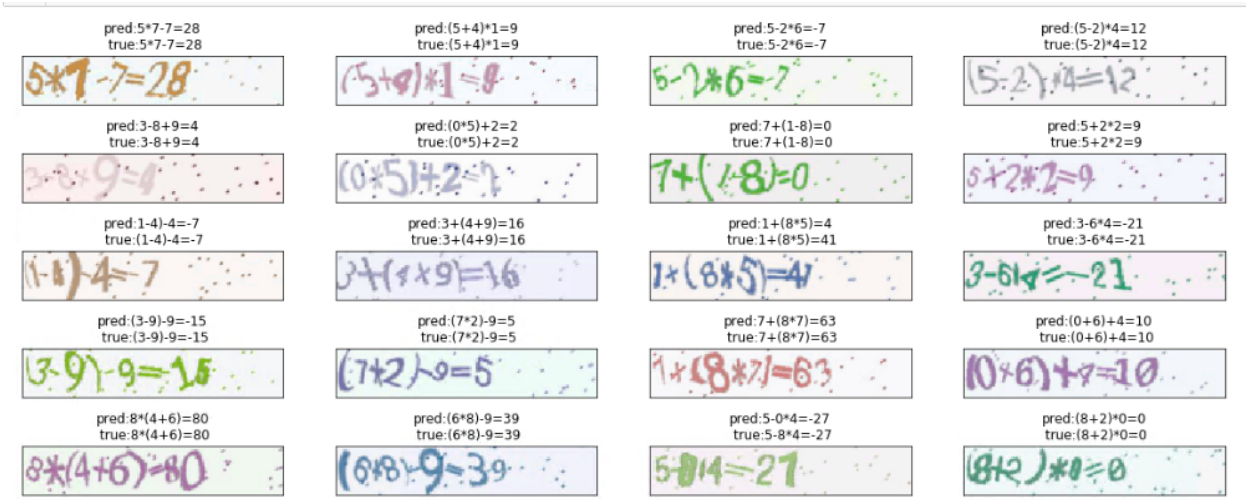
5. 项目结论

5.1 结果可视化

1. 精确度和损失曲线



2. 20张图片结果图：



5.2 对项目的思考

项目从着手开始做到结束总共持续了一个多月，从刚开始了解 CRNN 到最终完成模型并达到基准要求。学到了很多，也真是的体会到了一个模型从无到有的建立过程。其中遇到了困难也收获了不少的乐趣和知识。最开始对CTC input_length 理解不够透彻，碰了很多次壁。在参数调优的过程中，由于每次训练的时间比较长，然后要根据这次的结果来进行下一步的调优，比较耗费时间。 最初模型的准确率只有97%左右，并且按照自己的理解进行调优，模型的准确率不升反降。无奈之下，只能将模型重新构建，尝试新一轮的调试。再经过了不断的尝试之后，新的模型取得了比较满意的结果。 同时，之前学习 CNN 的时候。都是按照给定项目的固定流程一步步构建，当自己真正去做的时候才对 CNN 模型有了进一步的认识。更奇特的是，CNN，RNN 和 CTC 组合构建的模型并不需要分割字符，也不需要严格意义上的固定长度。这在处理序列识别的时候特别的方便和高效。

5.3 需要作出的改进

- 生成算式验证码生成器, 扩充数据集。
- 尝试现有CNN（VGG-19/xception/inception）模型。

6. 参考文档

1. http://deeplearning.stanford.edu/wiki/index.php/%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C
2. https://blog.csdn.net/stdcoutzyx/article/details/41596663
3. https://zhuanlan.zhihu.com/p/28492837
4. https://feisky.xyz/machine-learning/rnn/
5. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition