机器学习纳米学位

算式识别 金阳 Udacity

2018年12月16日

I. 问题的定义

简介

图像中的序列识别一直是计算机视觉里的一个热点。该项目是识别一张图片中的算式，就是图片序列识别的子集。该项目通过一个端到端的深度神经网络实现序列识别，在测试集上的正确率超过99%。

项目概述

使用深度学习识别一张图片中的算式。

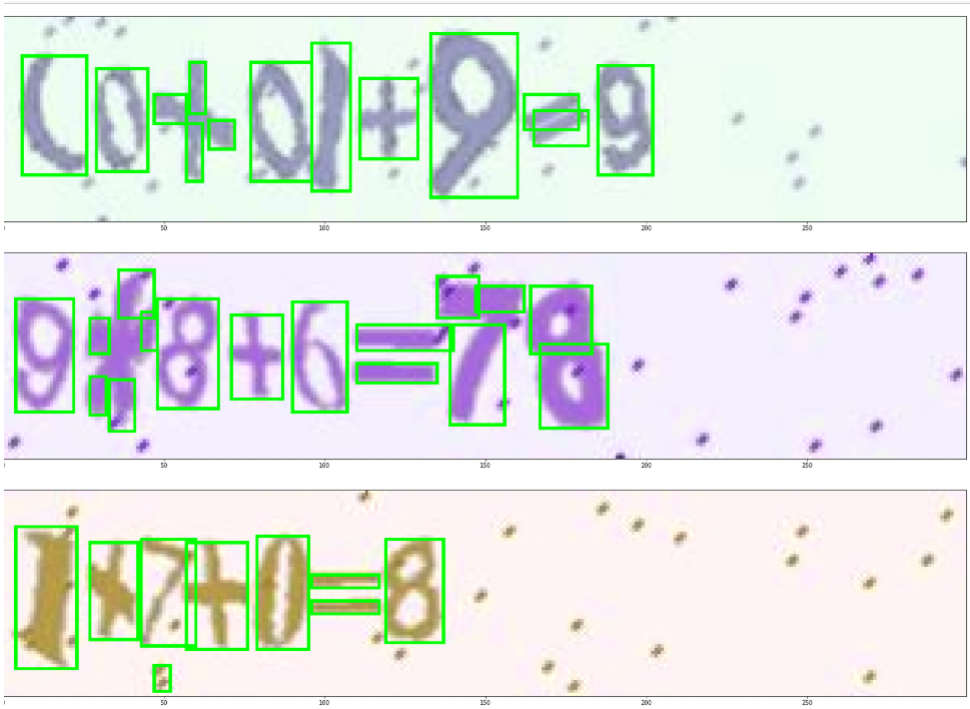
卷积神经网络（Convolutional Neural Network, CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。

卷积神经网络由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池化层（pooling layer）。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习结构相比，卷积神经网络在图像和语音识别方面能够给出更好的结果。这一模型也可以使用反向传播算法进行训练。相比较其他深度、前馈神经网络，卷积神经网络需要考量的参数更少，使之成为一种颇具吸引力的深度学习结构。

但是该项目是识别图像中的序列，图像中有不定数目的对象需要识别，所以不能直接套用卷积神经网络模型。

- plan A

我开始决定先识别出字符的位置，切分出单个字符然后使用卷积神经网路模型处理。由于数据集是没有字符位置标注的，所以不能使用深度模型来做识别，我选择了使用传统机器学习中的轮廓检测来做，但是效果不理想，不能很准确的识别出字符位置。所以这个方案不成立。



- plan B （最终使用的方案）

长短期记忆（LSTM）是一种时间递归神经网络（RNN），论文首次发表于1997年。由于独特的设计结构，LSTM适合于处理和预测时间序列中间隔和延迟非常长的重要事件。

LSTM的表现通常比时间递归神经网络及隐马尔科夫模型（HMM）更好，比如用在不分段连续手写识别上。本项目的算式识别，算是不分段连续手写识别的一种。所以在RNN部分使用LSTM。

综上，所以我决定结合CNN和RNN，使用一个端到端的模型来完成项目。

这是一个图片识别问题，所以需要用到卷积神经网络（CNN），并且需要对图片数据做一些预处理 。

算式图片中出现的长度是不定长的，属于序列识别，需要用到递归神经网络（RNN）中的LSTM得到计算结果。

我决定使用卷积神经网络提取出特征之后，输入到递归神经网络中，识别出其中的算式。

评价指标

算式图片正确率=识别正确的算式数量/算式的总数

当算式图片识别的每一个字符都正确时，该算式为识别正确。

II. 分析

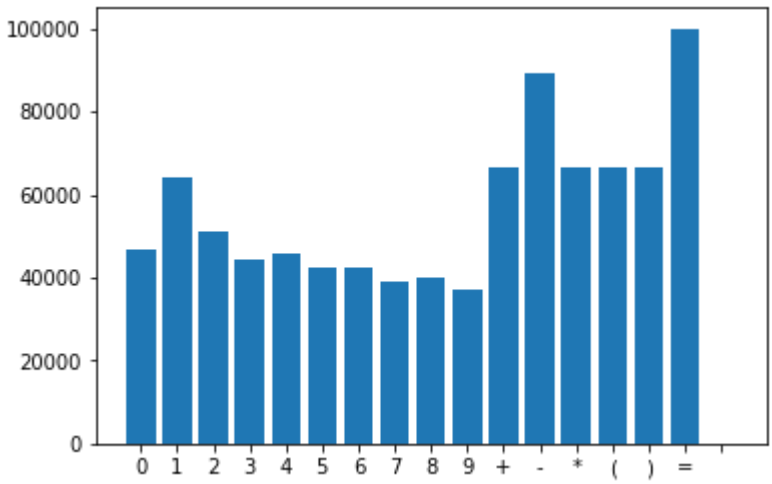
数据的探索

- 此数据集包含10万张图片，每张图里面都有一个算式。
  - 每个算式可能包含 `+-*` 三种运算符， 可能包含一对括号，可能包含0-9中的几个数字，以及每个算式包含一个等号。所以一共出现的字符总数是16种。
  - 每个字符都可能旋转。
  - 图片大小统一是300*64。
  - 图片字体是各种颜色的，背景也是各种颜色的，但是背景都是浅色（接近白色）
  - 图片中有一些噪点。

探索性可视化

统计标签中各个符号的数量，画柱形图。

发现等号最多，每个算式都有，运算符号和括号平均比数字类型的符号多。



算法和技术

- CNN模型的选择：我没有选用一些主流强大的模型如Resnet，Xception等，而是自己搭建一个简单的CNN模型。原因主要有两点：
  - 由于需要识别的图片较小（300*64），如果使用Resnet模型，需要图片的长宽像素不小于 224，需要在原始图片边上填充大量像素，不大合适。
  - 而且要识别的内容比较简单，使用那些模型有些大材小用，使用简单的CNN模型就可以胜任工作。
- CNN的设计：
  - 卷积层的hidden unit逐渐增加，最开始输入的图片形状是（64，300，1），最后输出形状是（1, 60, 1100），把这三维理解成长宽高的话，我认为把一个扁平形状的数据通过卷积处理成长条状的数据，是比较理想的。
  - 每两层卷积层，插入批标准化层，加快运算速度，减小过拟合。
  - 每两层卷积层，插入最大池化层，减小卷积面积，减小过拟合。
  - 使用relu作为每层卷积层的激活层。
- 优化器：选择了Adam，Adam结合了AdaGrad和RMSProp的优点，是目前综合性能表现最好的优化器。
- 损失函数：categorical_crossentropy（分类交叉熵），一般用于多分类问题，像本项目中字符总数是16种，属于多分类问题。

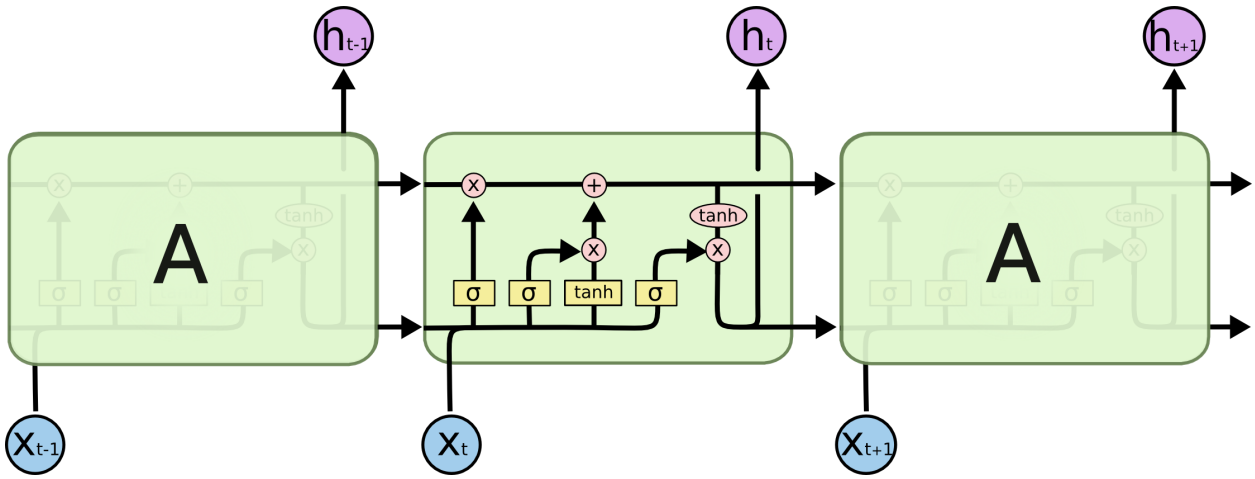
$$loss = -\sum i = 1ny^i 1logyi1 + y^i 2logyi2 + \cdots + y^i mlogyim$$

- 因为输出的序列长短是不一样的，我使用空来代替空缺的位置。

这里的补空不需要用到CTC loss，因为只有算式的尾部需要补充空缺。

比如第一张图片的label是“(0+0)+9=9”，一共是九个字符，而我们的设定的输出是11个字符，所以在这个label后补空，直到它变成11位，“(0+0)+9=9空空”。

- 长短期记忆（LSTM）是一种时间递归神经网络（RNN），它通过输入门，输出门，遗忘门来控制流程，和传统的RNN模型相比，解决了梯度消失的问题。



$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

其中x是输入，h是输出，C是lstm的单元状态。

f,i,o分别代表遗忘门，输入门，输出门。激活函数是sigmoid函数，所以门的值域是(0,1)。当值为1的时候表 示“完全保留”，0 表示“完全舍弃”，输出值会介于二者之间。

~C_t代表新的单元状态信息，由激活函数是tanh，所以值域是(-1,1)。

前向传播时:

通过X_t（输入），h_t-1（之前的单元状态），使用每个门对应的W和b计算出，f_t（遗忘门），i_t（输入门），o_t（输出门），~C_t（新单元状态信息） 的值。

f_t（遗忘门）与C_t-1（上个单元状态）相乘，遗忘门控制保留多少上个单元状态。

i_t（输入门）与~C_t（新单元状态信息）相乘，输入门控制保留多少新单元状态信息。

两者相加的和就作为新的单元状态，单元状态经过激活函数tanh后，与o_t（输出门）相乘，获取h_t(输出)，输出门控制输出多少单元状态。

技术

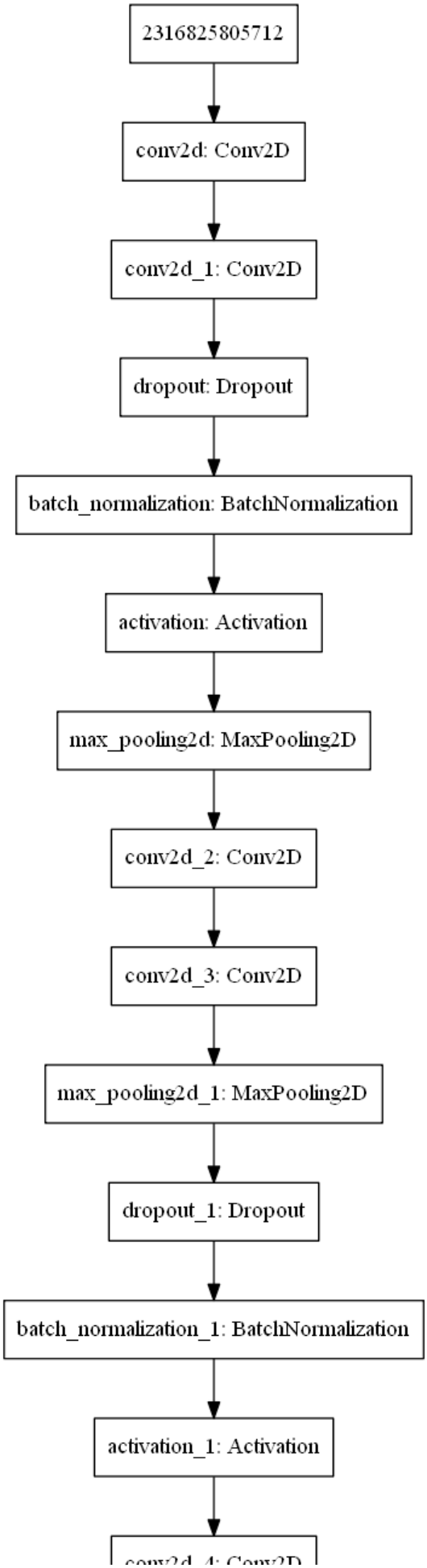
项目中主要使用keras框架。使用keras框架能快速搭建模型，利用GPU计算优势，快速迭代模型。

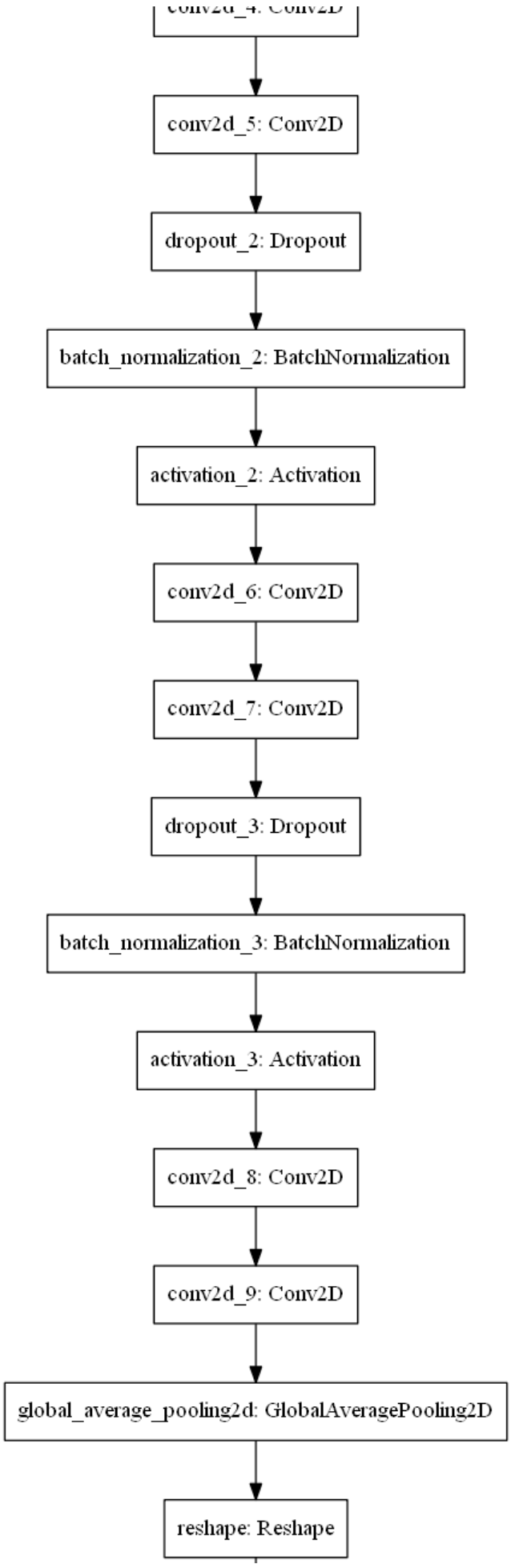
基准模型

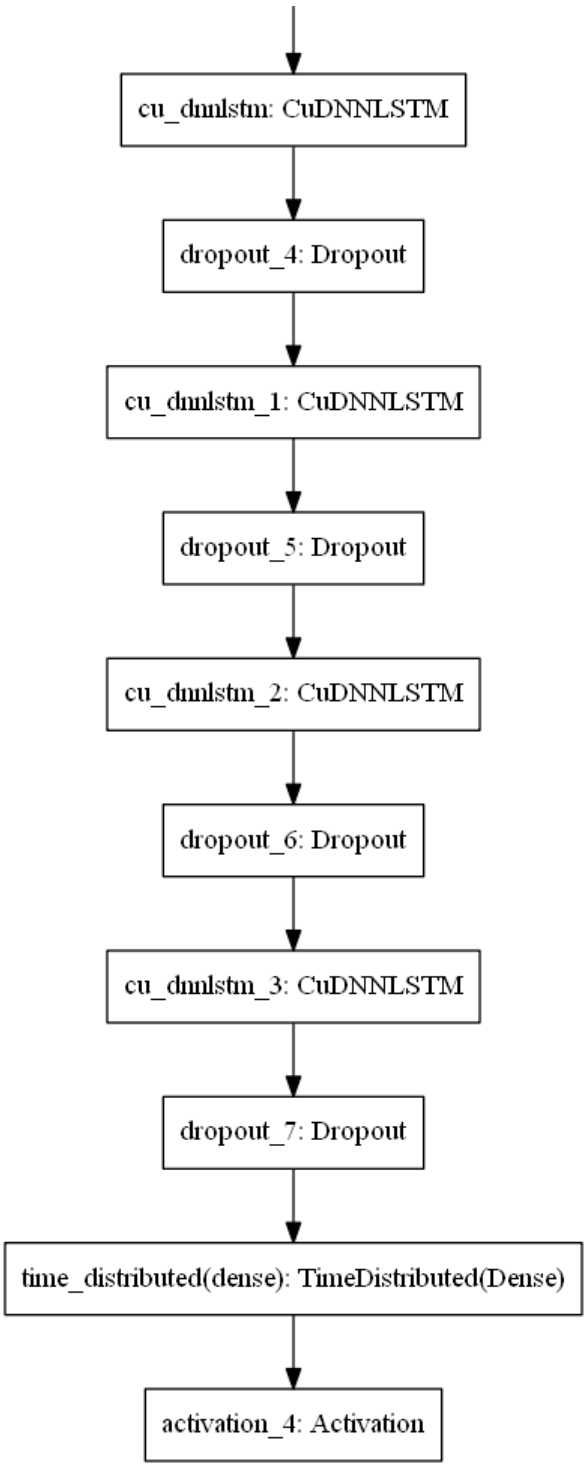
一个类似的项目，识别现实生活中音符照片的序列识别论文中，模型在测试集上的正确率有84%.参考了它的模型，我设计了基准模型如下：

其中：

- Convolution layer：kernel size：3*3，strides:1，padding:0
- MaxPooling layer：window size：2*2，strides:0
- 所有的relu激活函数被省略：


2316825805712
conv2d: Conv2D
conv2d_1: Conv2D
dropout: Dropout
batch_normalization: BatchNormalization
activation: Activation
max_pooling2d: MaxPooling2D
conv2d_2: Conv2D
conv2d_3: Conv2D
max_pooling2d_1: MaxPooling2D
dropout_1: Dropout
batch_normalization_1: BatchNormalization
activation_1: Activation


conv2d_5: Conv2D
dropout_2: Dropout
batch_normalization_2: BatchNormalization
activation_2: Activation
conv2d_6: Conv2D
conv2d_7: Conv2D
dropout_3: Dropout
batch_normalization_3: BatchNormalization
activation_3: Activation
conv2d_8: Conv2D
conv2d_9: Conv2D
global_average_pooling2d: GlobalAveragePooling2D
reshape: Reshape



使用该基准模型在数据上训练了20个批次，在验证集上准确率达到86%，目标准确率定位99%。

III. 方法

数据预处理

字符识别中颜色影响不大，首先把图像从RGB图转化成灰度图，可以减少图像层数为1层，大幅度减低计算量。图像的形状都是统一的，不需要处理。然后对图片进行归一化，使数值落在0到1之间，方便后续计算。

由于数据中最大长度为11，可能出现的字符为17种：'0','1','2','3','4','5','6','7','8','9','+','-','','(',')','=',''。*所以把标记（label）内容处理成11*17的one-hot形式，方便后续计算。

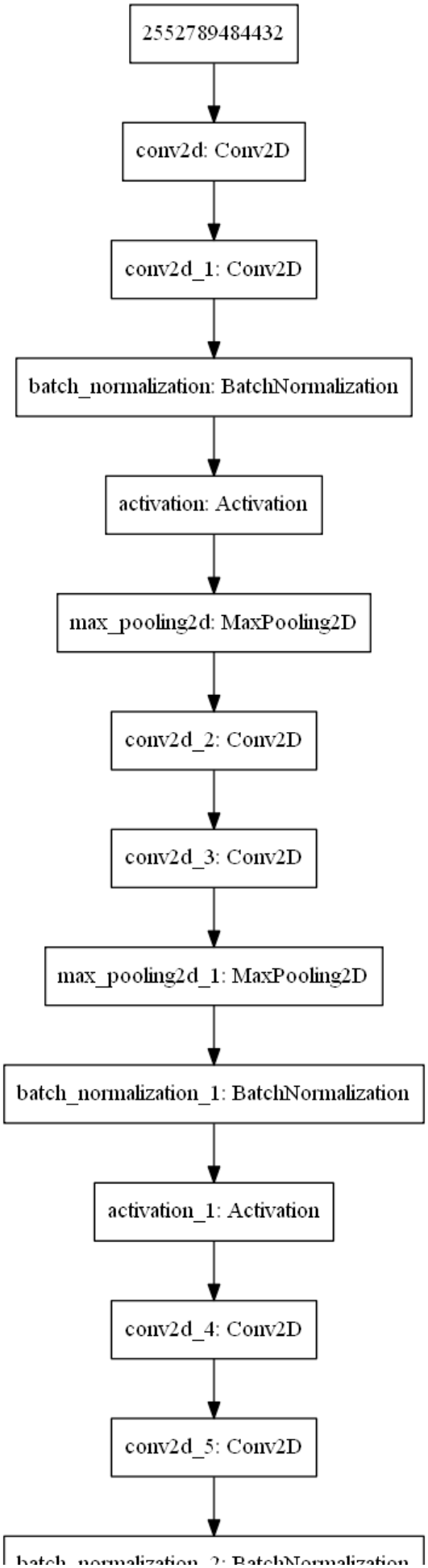
然后把图像和标记内容划分为训练集，验证集和测试集。总共图片为10W张，划分为训练集85500张，验证集9500张，测试集5000张。

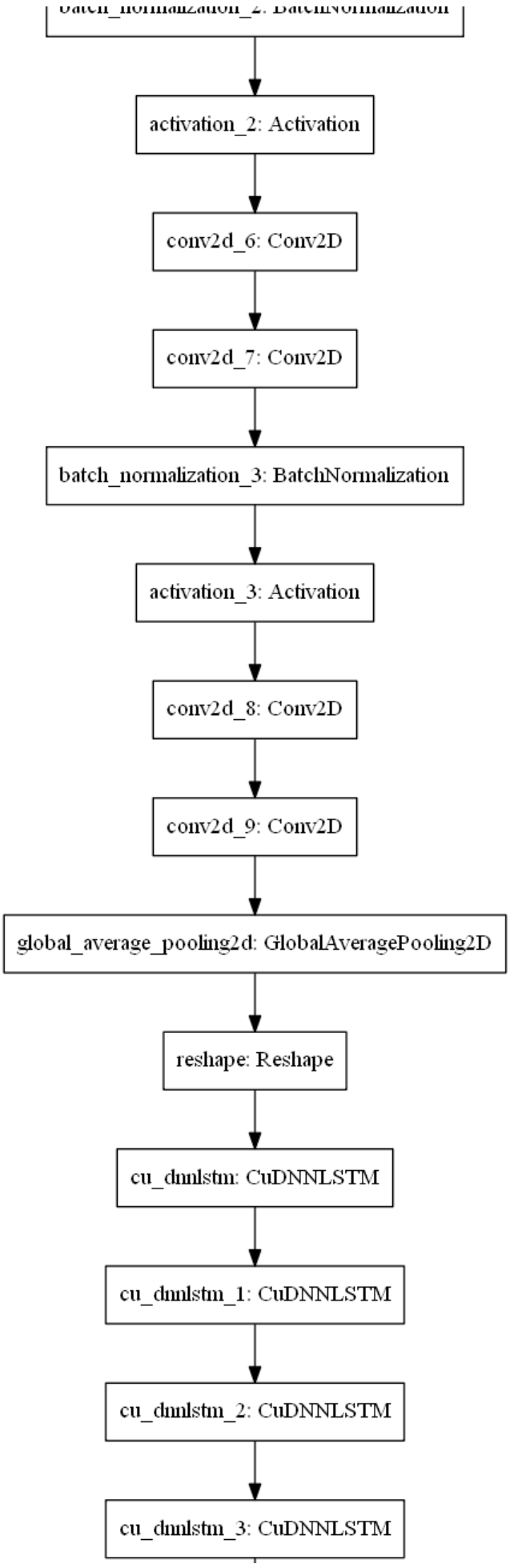
执行过程

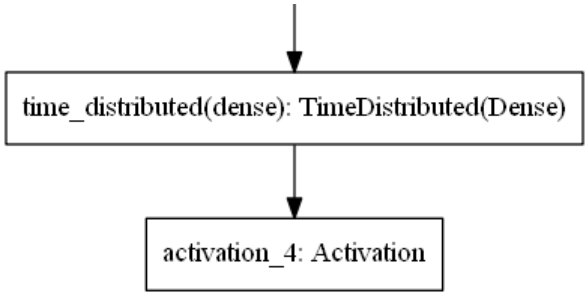
损失函数：categorical_crossentropy（交叉熵）

优化器：Adam

由于去除了Dropout，最终模型如下图：

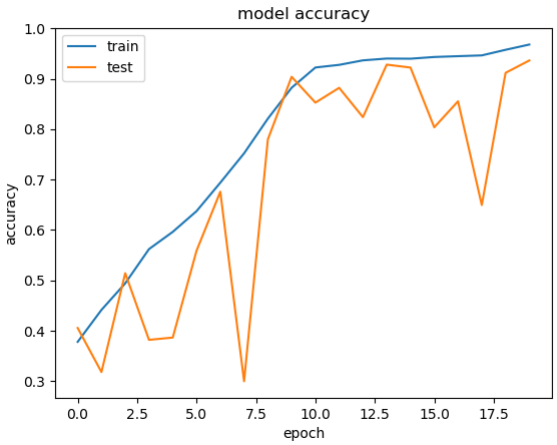

2552789484432
conv2d: Conv2D
conv2d_1: Conv2D
batch_normalization: BatchNormalization
activation: Activation
max_pooling2d: MaxPooling2D
conv2d_2: Conv2D
conv2d_3: Conv2D
max_pooling2d_1: MaxPooling2D
batch_normalization_1: BatchNormalization
activation_1: Activation
conv2d_4: Conv2D
conv2d_5: Conv2D

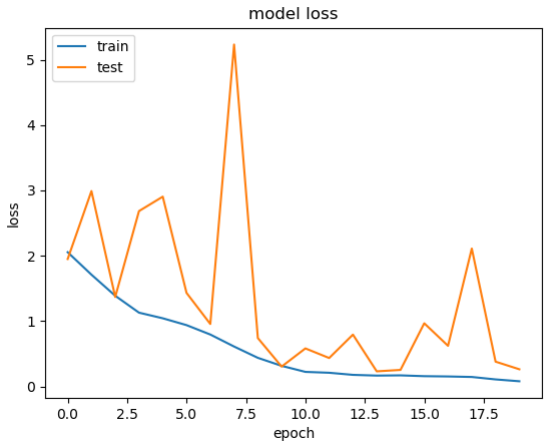

activation_2: Activation
conv2d_6: Conv2D
conv2d_7: Conv2D
batch_normalization_3: BatchNormalization
activation_3: Activation
conv2d_8: Conv2D
conv2d_9: Conv2D
global_average_pooling2d: GlobalAveragePooling2D
reshape: Reshape
cu_dnnlstm: CuDNNLSTM
cu_dnnlstm_1: CuDNNLSTM
cu_dnnlstm_2: CuDNNLSTM
cu_dnnlstm_3: CuDNNLSTM



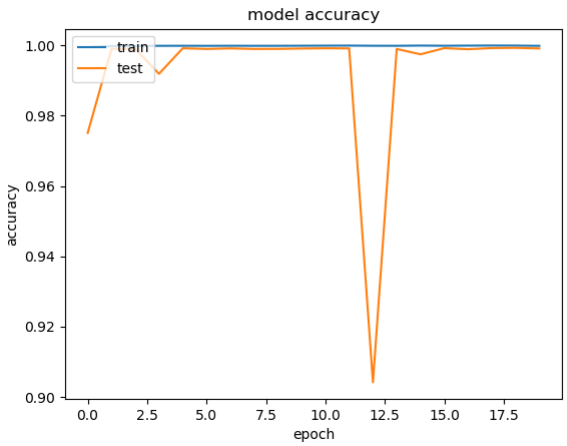
- 执行过程，把数据输入模型，学习率为0.001，批大小为128张图片，执行20个批次，选择保存在验证集在误差（loss）上最小的模型，观察随批次误差和正确率的情况。

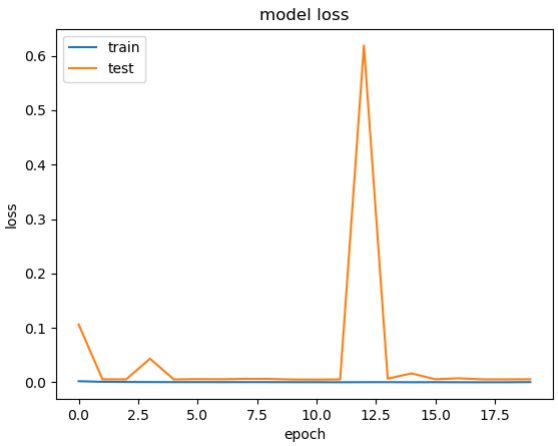
观察到误差基本稳定下降，正确率上升知道平缓。验证集上的表现与测试集表现相对贴合。



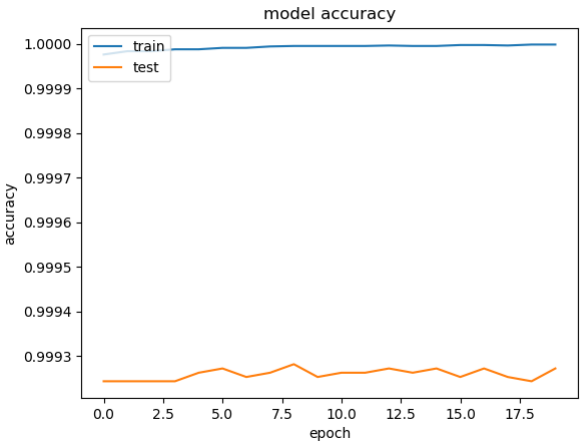


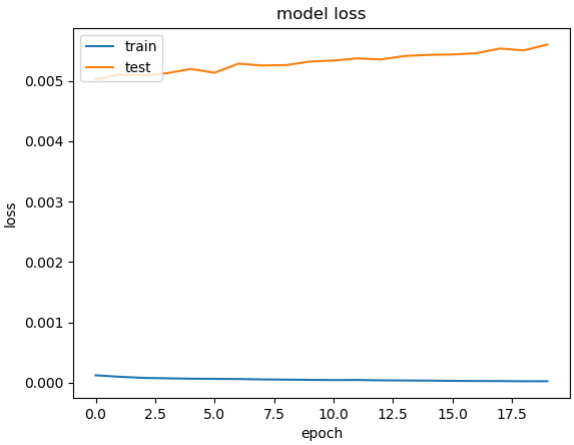
观察到正确率和误差曲线逐渐平缓，将学习率改成0.0001，再训练了20个批次，在验证集上正确率（字符）达到99.93%，达到预期目标。





然后再调小学习率为0.0001，训练了20个批次，观察到虽然训练集的loss曲线还在往下降，但是验证集的loss曲线开始往上升，有过拟合的趋势。而验证集的accuary曲线也在不断波动，决定结束训练。





执行过程中遇到的问题是：

- 内存溢出，由于一开始是把所有的图片读入内存中，然后进行训练的，16G的电脑内存并在读入7W张图片后，出现内存溢出问题。

  现在改使用fit_generator方法，一开始只读取图片的地址，在训练需要时读取才读取对应的图片数据。

完善

因为一个算式里只要有一个字符没识别对，这个算式就算错误，所以字符识别错误率会比算式识别率低接近10倍，所以我们的目标是在验证集上字符的正确率超过99.9%，这样最后在测试集上算式识别率就能达到目标设定的99%了。

开始使用比例为0.2的Dropout，训练40个批次，但是验证集误差下不来。

后来去除了Dropout，发现误差下来了。发现dropout是需要考虑到具体问题的，需要的时候才加。

Dropout:	**0.2**	**无**
训练集误差:	0.0003	0.00002
训练集正确率（字符）：	0.9999	0.9999
验证集误差:	0.0203	0.00432
验证集正确率（字符）：	0.9981	0.9992
验证集正确率（算式）：	0.985	0.9924

在验证集上得到正确率最高的模型后，我使用测试集来拟合该模型在实际使用中的正确率。

类别	
测试集误差	0.0034
测试集正确率（字符）：	0.9993
测试集正确率（算式）：	0.9936

IV. 结果

模型的评价与验证

模型由CNN和RNN两个部分组成：

CNN部分：

有5个模块组成，前四个模块的结构类似，以第一个为例，组成为：两个卷积核大小为3*3的卷积层，一个标准正则化层，一个relu激活层。然后每个模块卷积层的卷积核的数量。

最后一个模块在两个卷积层后，通过全局平均池化层，把输出变成一维（1600），然后再把输出的形状调整成（11*100）后，为输入RNN部分做准备。

RNN部分：

每个模块为一个LSTM层，如此4个模块后，加上一个全连接层，一个*Softmax激活层，最终输出的形状是（11 * 17）

我先后尝试两次训练，最终在测试集上都到超过99.5的正确率，我觉得模型的鲁棒性不错，比较稳定。

合理性分析

测试集正确率（算式）= 0.9936

使用最终模型在测试集上的正确率达到了设定的目标。

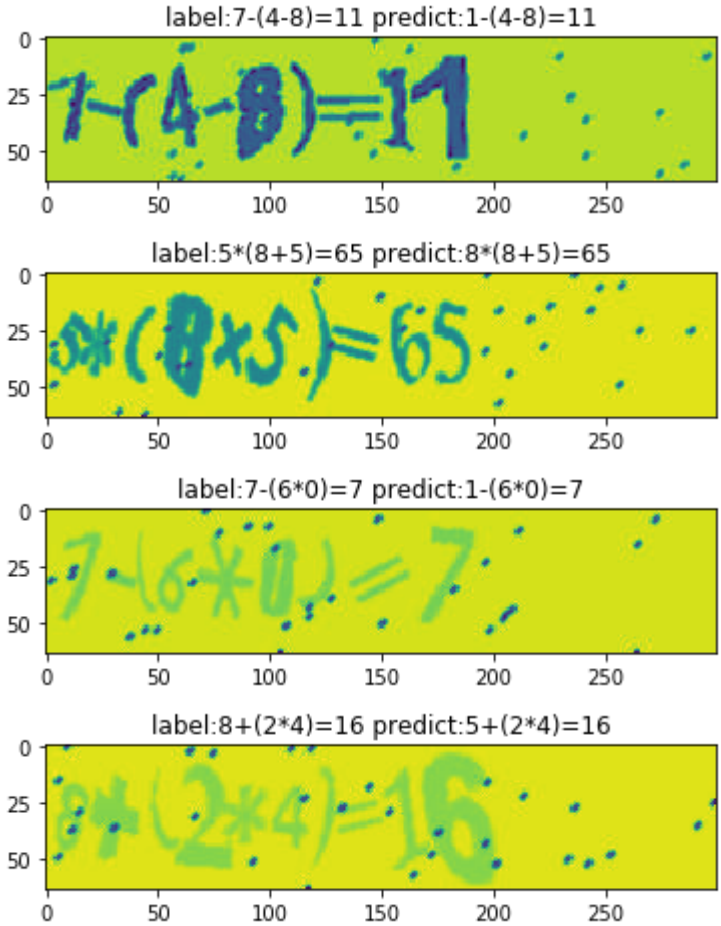
V. 项目结论

结果可视化

查看显示错误的图片，错误主要发生在比较相似的数字之间的辨认上，如7和1，5和8上。

像+-* 三种运算符， 括号，等号由于数量较多，且没有相似的符号，所以没有出现识别错误。

这些识别字符还是比较容易被人正确识别的，所以该算法还有提升的空间。



对项目的思考

结合CNN和RNN的端到端模型效果很好，数据图片中的噪点以及符号一定程度的旋转，不需要手动的处理，也不会影响到结果，模型能从这些干扰中找到正确的结果，所以深度学习还是很强大的，在计算机视觉项目中，可以减少很多人工对图片的预处理。

比较困难的地方是收到机器内存的限制，无法使用全部图片，所以机器学习内存很重要。

我觉得这个模型在正确率和易用性上符合我的期待。对于其他通用场景的问题，我觉得在对处理车牌识别，或者幼儿算术题识别，街道文字识别等领域有一定的交界相通的地方。

需要作出的改进

- 开始设计的模型效果就挺好，我觉得可以尝试的是，使用更简单，层数更少的模型，这样能在保持正确率的情况下，加快计算的速度，节约计算资源。
- 可以用上图片增强，生成一些不同角度，颜色，缩放的图片，来获得更多的图片数据用于训练模型，增强模型泛化性能。

引用

- S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.
- A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis

and Machine Intelligence, vol. 31, no. 5, 2009.
- Baoguang Shi, Xiang Bai，Cong Yao（2015）An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition
- Sebastian Ruder（2017）An overview of gradient descent optimization algorithms*
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/