

届毕业设计（论文）

题 目：基于人工神经网络的在线识别Web应用

专 业： 自动化

班 级： 自1402

姓 名： 徐浩聪

指导老师：

起讫日期：

年 月

目 录

摘要…………………………………………………………………………………I

ABSTRACT………………………………………………………………………II

第一章 应用架构………………………………………………………………1

1.1应用结构及思路……………………………………………………………2

1.1.1应用组成部分………………………………………………………2

1.1.2为什么要用javascript……………………………………………2

1.1.3人工神经网络基础知识……………………………………………3

1.2 SVM的识别………………………………………………………………6

第二章 开发流程……………………………………………………………9

2.1 模型部分………………………………………………………………9

2.1.1 准备图片库……………………………………………………12

2.1.2 构建网络………………………………………………………12

2.1.3 训练网络………………………………………………………12

2.1.4 测试网络………………………………………………………12

2.2 后端部分………………………………………………………………9

2.2.1 搭建服务……………………………………………………10

2.2.2 对接模型……………………………………………………12

2.2.3 写接口………………………………………………………12

2.2.4 HTTPS配置……………………………………………………12

2.3前端部分………………………………………………………………14

2.3.1 交互页面……………………………………………………17

2.3.2 微信小程序…………………………………………………17

结语………………………………………………………………………………104

参考文献…………………………………………………………………………106

致谢………………………………………………………………………………107

摘 要

本文首先介绍了该应用的大体架构及各个部分的作用，并介绍了在该应用中所用到的神经网络算法。然后具体讲述了模型、后端、前端这三大部分的具体实现步骤及注意事项。

**关键词**：人工神经网络 JavaScript nodejs synaptic emnist 前后端 微信小程序

第一章 应用架构

* 1. **应用结构及思路**

1.1.1 应用组成部分

该应用可分为三大部分：模型部分，前端部分和后端部分。

模型部分主要是准备图片库，构建网络，训练、测试，最终生成识别模型。

前端部分主要是完成前端页面的制作及接口对接数据展示，这里准备采用微信小程序的形式。

后端部分主要是搭建服务，对接模型，写接口，连通前端和模型。

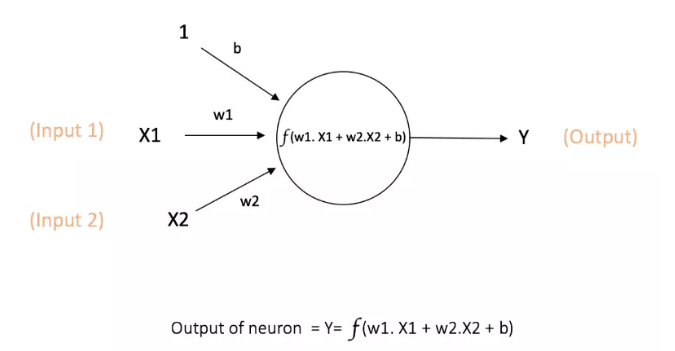
整个应用的流程是这样的：用户打开微信小程序，画出某个字符，点击识别按钮，会将字符的数据发送给服务器，服务器接收到数据，进行一系列处理转化成模型可接收的数据形式并传给模型，模型输出识别结果，服务器将获取到的结果返回给前端，最后前端将结果展现出来，完成识别。

1.1.1 为什么要用javascript

本应用主要用到的语言为javascript。对于javascript，大多数人总是对它有所偏见，认为这是它只是个依附于浏览器的没多大用处的语言。我作为一个js的忠实粉丝，将一一列出js的过人之处。速度快。在V8引擎下，js已经快过了python和ruby，虽然和java、C++相比还是慢了不少，不过作为一个解释型语言，那也是没办法的事。Js也可以做矩阵操作。随着js的不断发展，各种库陆陆续续出现，比如矩阵操作方面，就有math.js。也可以做服务端开发，比如nodejs。

1.1.1 人工神经网络基础知识

人工神经网络（ANN）是一种由人脑启发式的计算模型。它由神经元组成，与人脑类似，神经网络的基本组成单元是神经元。首先数据传入网络，输入先赋给神经元，给每个输入端加一个权重，权重的大小与输入的重要程度有关。得出所有输入的加权和，再给激活函数计算，最后得出输出。

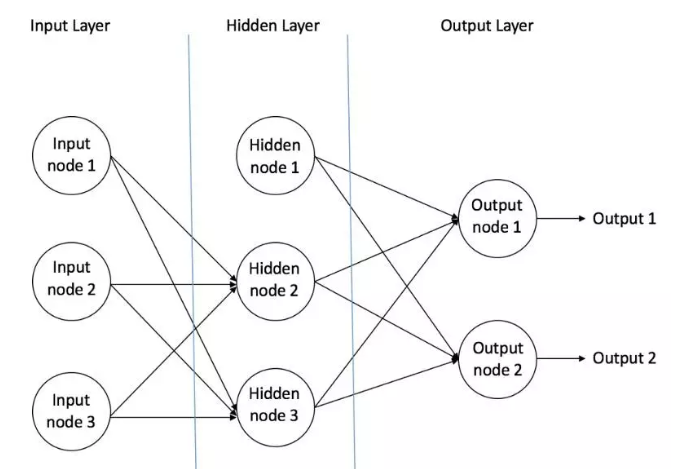


上述网络包含输入X1和X2，与这些输入相关联的权重 w1和w2。还有另一个输入1 与权重b（偏差）。神经元的输出Y如上图所示。

由于大多数现实世界的数据都是非线性的，而线性模型只能解决线性问题，所以线性模型并不能适用于大部分现实问题。所以需要使用非线性模型。这里的激活函数f是非线性的，作用是引入非线性因素。

前馈神经网络

前馈神经网络是最简单的，也是第一个人工神经网络。



一个前馈神经网络由以下三种类型的节点组成：

输入节点：主要作用是将传入网络的信息传递给下一层，称为“输入层”。在输入节点中都不含有计算过程，只起到传递作用，将信息传递给隐藏节点，若没有隐含层，则直接传给输出层。

隐含节点：隐含节点处于网络的内部，外部无法直接访问，所以对我们为你来说，隐含层是不可见的。它的主要作用是执行计算，网络中的大量计算都在隐含层中，计算完成后再将信息从输入节点传到输出节点。隐含节点的集合叫做“隐含层”，隐含节点越多，网络越复杂，计算量也越大。虽然前馈网络只能有一个输入层和输出层，但可以具有多个隐藏层。

输出节点：输出节点构成的集合称为“输出层”，主要进行和隐含层类似的计算，并最后将信息输出到外部。

在前馈网络中，数据的流动方向只有一个，从输入节点，到隐含节点，最后到输出节点，网络中没有循环。

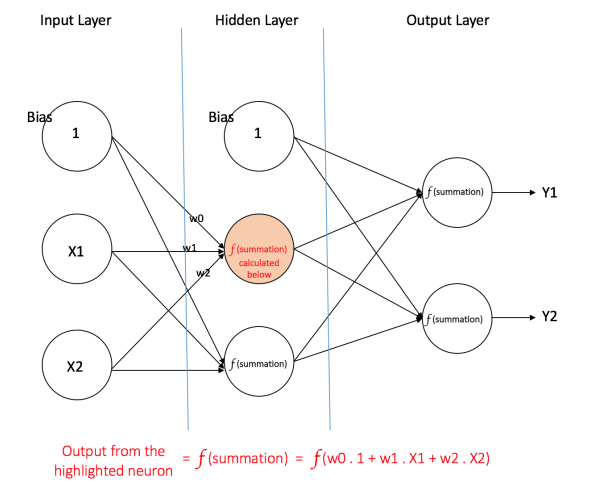
前馈网络有两个类别：

单层感知器：是最简单的前馈网络，由于没有隐藏层，效果不好，一般不用。

多层感知器：有一个或多个隐藏层，计算量与单层相比大得多，效果也好得多。在实际应用中，问题大多比较复杂，所以大多使用多层感知机。

多层感知器 MLP

多层感知器具有一个输入层，一个输出层和一个或多个隐含层。



上图的多层感知机有一个隐含层，其中输入层包含三个节点，首先在输入层中不执行计算，1，X1和X2分别作为输入层的输出送到隐含层中。

隐含层有三个节点，偏置节点的值是1。来自输入层（1，X1，X2）的输出以及与对应的权重决定了隐含层中其他两个节点的输出。其中F是激活函数。经过激活函数计算后被送到输出层中。

输出层：有两个节点，它们从隐含层获取输入并执行相似的计算，计算的结果Y1、Y2作为多层感知器的输出。

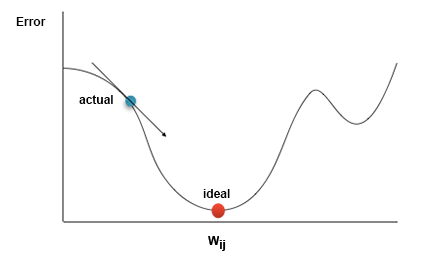
反向传播算法

缩写为BackProp，是多层感知器的一种学习过程。它被叫做有监督的学习，类似于“从错误中学习”，在出现错误时会对其进行更正。

ANN由不同层次的节点组成; 输入层，隐含层和输出层。相邻层的节点之间的连接具有与其相关的权重。训练的目标是给各个节点分配正确的权重。给定一个输入量，这些权重将决定多大的输出量。

BackProp算法：

最初所有的权重是随机分配的。在给网络一个输入之后，它会产生一个输出，下一步是教网络该输入应该是什么样的正确输出（理想输出）。网络将采用这个理想的输出，并开始调整权重以在下一次产生更准确的输出，从输出层开始并向后返回，直到到达输入层。因此，下一次我们向网络展示相同的输入时，它将使输出更接近我们训练输出的理想输出。这个过程将重复很多次迭代，直到认为理想输出和网络输出之间的误差够小，即到达设定的最小误差。

该算法使用梯度下降计算来调整权重大小。假设网络中输出的某个权重和错误之间的关系如下图：

梯度下降是在函数曲线中找到一个下降路径，从高处逐步下降，直到到达空间的最低点的过程。算法从起点开始，一步一步向下，每一步会求出当前点的斜率，从而得知该位置的坡度。然后，会选择目前向下的斜率最大的一个方向，再继续向下。重复以上步骤，梯度下降算法就会找到一个极值点，得到目标函数的极小值。

超参数是指那些需要我们在训练过程中结合模型的实际情况与现实问题，进行手动调整的参数。梯度下降算法的超参数主要有：学习速率a，每次下降使用的样本数量 x，学习速率的衰减率d。

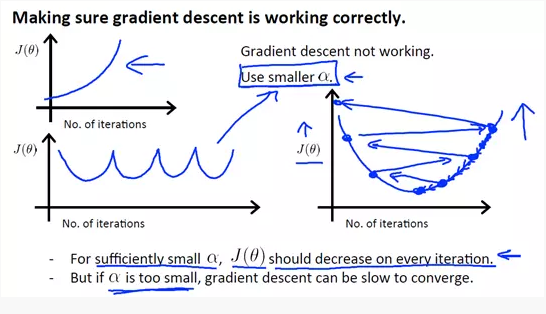
学习速率a对于梯度下降算法来说是一个非常重要的参数。它代表着算法每一次向下走的幅度。它的大小决定了算法收敛的速度：a过大时，算法下降速度过快，可能略过终点，而导致在终点处来回徘徊，反而收敛较慢；a过小时，算法下降速度较慢，导致最终收敛时间较长。

根据样本选择的数量，一般分为三类梯度下降算法：

批量梯度下降：所有的数据对下降的梯度都会有影响，就是一般的梯度下降。

随机梯度下降：这种梯度下降是通过每次随机取一组数据作为样本来进行的，这样速度很快，但问题是样本数量小，下降方向不准确，很容易只取到极小值而不是最小值。

小批量梯度下降：取一部分数据来调整梯度下降算法，它的优点是结合了批量梯度下降和随机梯度下降，既通过一定量的数据保证了梯度下降的可靠性，也减小了每回合训练的数据量，加速训练速度。

随着算法逐渐下降至终点，最容易出现的情况是由于每步的步幅过大，导致点在终点周围来回摆动，不能收敛。但如果将学习速率a调小，那么在前期就会面临下降速度过慢训练时间过长等问题。

成功训练出模型后，为了测试模型是否训练成功，还需要通过测试数据来得出模型的准确率。一般来说，数据集可分为训练集/测试集(7:3)，训练集和测试集的数据最好不要有重复，否则会影响准确率。

交叉验证 Cross Validation

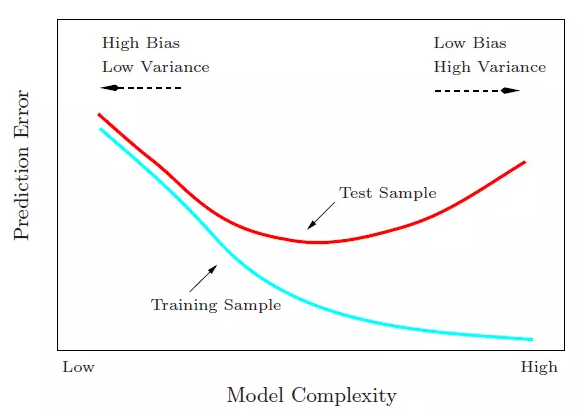
当数据量不够的时候，我们会需要通过交叉验证的方法在保证训练数据量足够的同时来实现合理的验证。在《统计学习方法》中，提到了三种常见的方法：

简单交叉认证：直接将数据集分为7:3，来进行验证。

S折交叉验证：将数据随机切分为S个子集，这些集合大小相同且互不相交，取1个集合做测试，剩下的S-1个子集做训练，

留一测试：在第二种方法中，将n个数据切为n个集合，即上一种方法的极限情况。

过度拟合（overfitting）： 基于数据的训练，容易使得模型仅在本数据集中产生良好的表现，而在其它的数据集中表现极差，这被称为过拟合现象。



可以看到，对于一般的机器学习模型来说，模型复杂度越大，测试集与训练集的误差也会越大，所以我们需要找到一个平衡点，使得模型的误差较小的同时，两个数据集的相对误差也较少。可以参考奥卡姆剃刀原理：我们需要的是一个能够很好解释已知数据同时又是最简单的模型。

1.1.1 SVM的识别

在做这个识别字符前，我想到了两种方法来识别，一种是使用HOG和SVM来识别，另一种是使用ANN识别，这两种方法各有优缺点。

我最先做了SVM版的识别，实现了一个简单的光学字符识别（OCR）算法，将训练SVM根据其计算的面向方向梯度（HOG）特征向量来识别手写的小写字母。

1. 初始化HOGDescriptor。HOG特征描述符是主要用于目标检测的通用描述符，最初被使用于行人检测。HOG使用渐变直方图，梯度由不同角度的边缘检测确定。对于我们的每个字母图像，我们将计算HOG特征矢量，并且我们将向SVM输入这些特征矢量以标示每个字母的特征。
2. 初始化SVM,支持向量机是机器学习为我们提供的各种分类器之一，例如决策树，回归，贝叶斯分类器或神经网络。
3. 训练SVM。需要确保字母在图像的正中间，因为左上角中字母的HOG描述符与右下角角中相同字母的描述符不同。这是因为梯度位于对应特征向量的不同单元格中。

SVM实质上是一种分类器，最简单的情况是在一个平面上，有一个圆圈和一个三角，svm的作用是找到一条直线能把这两个分隔开来。但实际问题远远比这个复杂，通常一条直线无法做出分割，甚至多条也不行。这就需要把平面映射为超平面，即3D化，这样分割后在平面上的投影就会是一条曲线，能完成复杂对象的分类。

在试过SVM后，觉得不是特别好，虽然只需要少量样本就能实现不错的识别效果，但还是人工神经网络对我更有吸引力。

第二章 开发流程

下面将分别讲解这三大部分的实现过程：

**2.1模型部分**

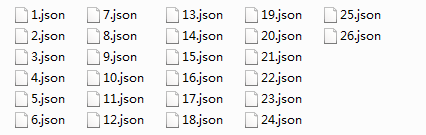
2.1.1准备图片库

最著名的图片库是mnist，它由6万多张训练图片和1万多张测试图片组成，每张图片都是长28像素宽28像素，这些图片是由不同人手写的0到9的数字。可是我要做的不是识别数字，而是英文字母。我找到了一个和mnist相比不那么有名的数据集：emnist。

https://www.nist.gov/itl/iad/image-group/emnist-dataset

和mnist一样，emnist里图片的长宽也都是28像素。emnist里是一系列压缩文件，这里需要emnist-letters-train-images-idx3-ubyte.gz和emnist-letters-train-labels-idx1-ubyte.gz，分别是二进制的图片数据和标记图片对应字母的映射。

// digitLoader.js labelsLoader.js rawMaker.js rawWriter.js emnist\_dl.js



二进制的数据不能直接拿来用，需要将原始数据处理成合适的格式，这里我做了一个转换器，将二进制数据转换成26个可被js直接读取的json文件，每个文件分别对应一个字母。

这时数据虽然能被读取到，可是不能直接使用，我又做了一个工具，将每个图片处理成1X784的一维数组，只要输入需要的图片数量，就会返回一个由这些一维数组构成的二维数组和相应的结果。

至此图片库已经准备好了，只要引入emnist.js，调用emnist.set(n1, n2)，就会返回n1条训练数据和n2条测试数据。

* + 1. 参数选择

神经网络的程序一般是用MATLAB、Python或是C、R等语言实现，但js也有相关的npm包，如Brain.js，ConvNetJS，synaptic，Deeplearnjs等。

这里用到了一个叫Synaptic的npm包，它是用javascript写的神经网络库，可以构建和训练出一般的神经网络。

首先，必须确定该网络需要多少个输入和输出神经元。由于每个图像的大小为28x28px，因此网络必须输入的像素数为28 x 28 = 784，这就是为什么我要将图片数据处理成1x784的数组的原因。而识别结果应为26个英文字母中的一个，因此输出神经元的数量为26。此外，网络应该至少有一个隐藏层，我在这里设置了1层隐含层，包含100个神经元。

另外，还有一些其他参数需要设置

学习率rate：0.07

最大迭代次数iterations：1000

最小错误率error：0.1

成本函数cost：Trainer.cost.CROSS\_ENTROPY

参数走到最优值的速度由学习率决定，如果训练时长时间不能收敛，可能是遇到了局部最优的情况，这时的学习率可能过小，可以适当调大。如果训练时函数无法收敛或是发散，那么可能是学习率过大，导致超过了最优值，此时应该调小学习率，使函数收敛。

设置一个合适的学习率，需要进行不断尝试。在一开始的时候，可以稍微设大一点，这样权重会改变的快一些，在迭代到一定的次数之后再减小学习率，看看变化。在经过几次尝试，将学习率设为0.07时，效果最好。

随着迭代次数的增加，模型在训练集上的偏差会越来越小，但也有可能在达到最小偏差后不降反升。由于本次训练数据比较大，有近两千多张图片，最大迭代次数设大一点比较好。

2.1.3 训练网络

参数配置好后就可以开始训练了，由于训练需要几天时间，训练过程中CPU一直是100%，对电脑是个不小的损耗，为了避免过热死机等因素破坏训练过程，我把代码传到了云服务器上，在阿里云ECS上跑，由于是云服务器，稳定性较高，而且安全可靠，可以远程监控，当CPU占用量由100%降到5%以内时，就代表训练完成了。

为了让程度能在服务器上跑，这里需要用到守护进程，我选择了forever。forever的作用是让node服务能在服务器上长时间的运行，forever有一个自己的进程，每守护一个程序，forever就会在自己的进程下开一个子进程，只要主进程不挂，子进程就不会有事。

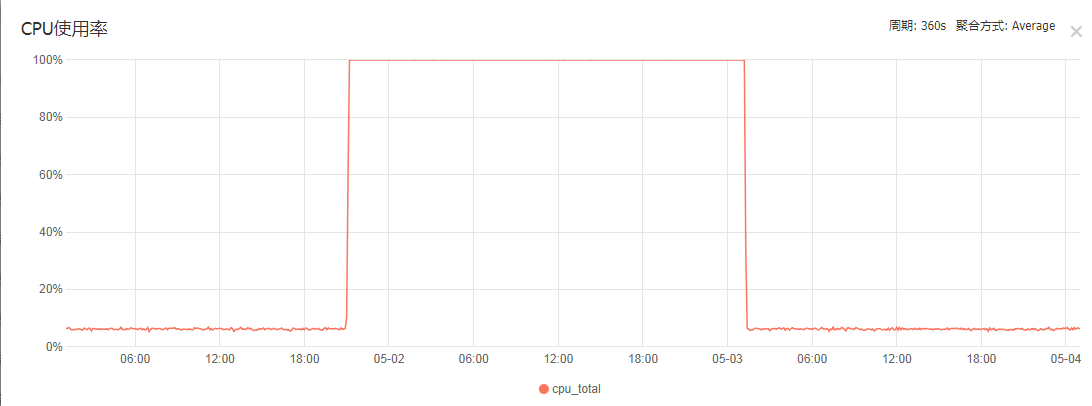
Forever还有一些配置项，这里我用了以下配置：

forever start -o train.log -m 1 app-train.js

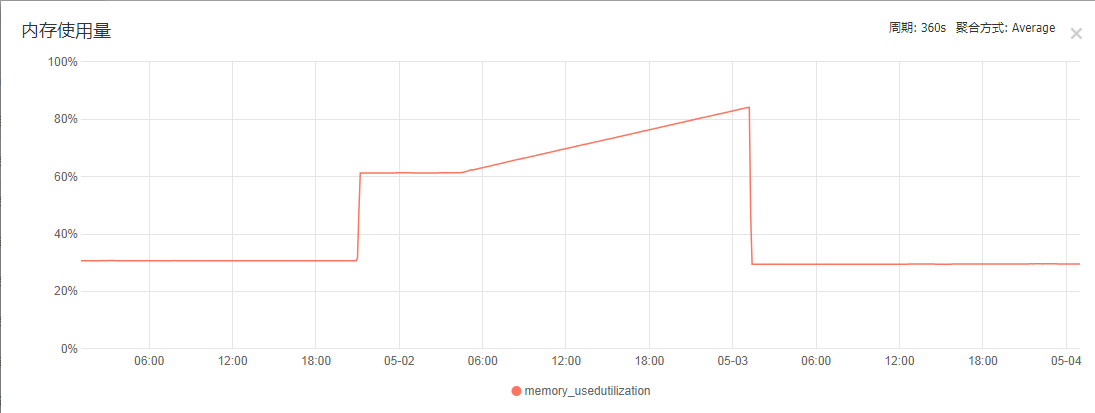
意思是开始执行app-train.js文件，输出日志到train.log中，最大执行次数为1次。

通过安装主机监控插件，云监控可以采集服务器监控数据，展示在控制台上。

在训练过程中，可以随时监控服务器资源使用情况：







* + 1. 测试网络

训练完成的模型也是一个js文件，6M左右，里面是大量的参数。由于不知道训练是否成功，所以训练完成后要做的第一件事就是测试模型的质量。这里要用到上文提到的emnist工具，传入需要测试的图片数量，返回相应的数据，将这些数据传入模型，获取识别结果，再将该结果与原数据的正确结果比对，从而得出识别率。识别率越大，则模型训练的就越好。

**2.2后端部分**

2.2.1搭建服务

后端使用nodejs，nodejs是运行在服务器上的JavaScript，语法和js基本相同，它是一个js的运行环境，基于V8 引擎，v8引擎是数一数二的js引擎。另外nodejs的IO模型天生有异步的特点，特别适用于高并发的情况。npm是Node.js 的包管理器，它是一个js的开源库，每天都有成千上万的人将自己的开源项目打包，分享到npm上。nodejs之所以能快速的发展，除了天生异步的语言特点外，还有一个重要的原因就是npm。

为了搭建web服务，这里使用了Express，一个基于nodejs的开发框架，

2.2.2对接模型

由于模型是一个js文件，光是读取并没有什么用处，还需要执行里面的代码，这里需要使用eval语句，返回值是一个函数，将其存起来，使用的时候传入1x784的数组，会返回1x26的识别结果。

2.2.3写接口

接口是连接前端和模型的桥梁，数据的流通方向大致可描述成：前端-接口-模型-接口-前端。

由于前端传进来的是长宽28像素的二进制图像数据，不同于一般请求，所以需要使用multiparty这个npm包来获取图像数据，而把二进制图像数据转化成数组则需要另一个关键的npm包opencv4nodejs，这个包实际上是opencv的js版，安装它之前需要先安装opencv和python，它提供了一系列api，通过调用它的api来调用本地的opencv进行图像处理操作。

为了防止前端图像尺寸传错，首先需要对图像resize为28x28。

然后进行图像的预处理，前端返回的图像是256色的位图的二进制数据，由于256色比较复杂，也没有必要，所以需要先进性灰度化处理。普通彩图的RGB三位的值不同，所以会显示出不同的颜色，而灰度化后像素的RGB三个值是相等的，图像就没有了颜色，整体上显灰色，只有亮度上的不同，所以叫做灰度化。灰度值的范围是0到255。值越大，图像越白；灰度值越小，图像越黑。

注意，这里有一个大坑。将emnist的数据转化为28x28，再画出来后，会发现图不是正的，而是先沿y轴做了一个镜像翻转，再顺时针旋转了90度，所以之后对输入数据也需要进行同样的操作。否则出来的结果千奇百怪。

现在的图像经过灰度化之后还是28x28的二维数组，值是0-255的数字，0代表黑色，255代表白色。在转为一维数组的同时，还需要对里面的灰度值进行计算，即进行二值化。因为原始数据是0-1，0代表白色，1代表黑色。因此需要将0-255转化为1-0，不光大小要归一，值也要反过来。这时需要进行阈值的选择，以阈值为分界线，两边的值为0或1。

最后得到的一维数组就可以传入模型，模型识别非常快，几毫秒就能返回结果，结果是一个1x26的数组，代表26个英文字母，每项的值代表可能性，越接近1，则越可能是对应的字母。一般来说，如果识别成功的话，26个值里只有1项无限趋近于1，而其他25个值无限趋近于0.

* + 1. HTTPS配置

配置HTTPS需要https证书，而在申请证书之前，需要先申请域名，并做好备案，域名指向和解析到服务器。

阿里云提供免费HTPPS证书，所谓的https证书其实就是几个文件。使用Express框架配置https只需要两个文件：private.key: 私钥和csr.pem: CSR证书签名。

由于https的端口是443，所以还需要在服务器上开放443端口，另外本应用的后端只开放443端口。

**2.3前端部分**

2.3.1交互界面

前端部分需要实现的主要功能是用户能手写字母，并将图像发送给后端。这里使用了html中的canvas标签，它有很多功能，主要是它是用来绘制客户端的矢量图形。它只是一块画布，但是有很多api给js使用，所以可以通过脚本绘制图像到画布上。

canvas标签最初在 Safari浏览器中由Apple引入。

Canvas标签仅仅是一个图形的容器，具体的绘制需要js来实现。这里我需要实现一个画板的功能。

首先获取canvas上下文，之后的所有绘图操作都要通过它来实现。然后绑定按下事件，当鼠标点击或手指按下时，获取点击点的坐标。再绑定移动事件，当鼠标或手指按住移动时，会频繁触发该事件，将此时的点和上一个点连起来，就绘制出了轨迹。最后绑定抬起事件，当鼠标或手指松开时，轨迹中断，结束绘制。这样，就可以绘制图形了。

另外，绘制完成后需要向后端发送数据，为了尽可能的减少数据量，也为了方便后端处理，在发送前需要将图像调整大小为28x28像素。

关于前端，我既做了网页的形式，也做了小程序的形式。两者各有优缺点，网页在手机上和电脑上都可以使用，写起来比较自由，就是比较复杂，而且在手机上体验不是特别好。而小程序只能在微信里使用，但做起来比较简单，用户体验接近原生app，缺点自由度不高，只能调用官方提供的接口，无法引入外部包，而且需要审核通过后才能使用。另外为了网络的安全性，微信强制规定所有小程序的后端接口都需要https加密。

2.3.2 微信小程序

微信小程序是微信的一个功能，它实际上是只能在微信里打开的网页。小程序的初衷是让网页具有和原生APP相同的体验。

小程序框架一共有三大块，WXML，WXSS和JavaScript。其中WXML和WXSS属于视图层，js属于逻辑层。框架实现了视图与逻辑这两层的数据传输，可以在开发时省去繁杂的视图与数据之间的数据互通，更方便的聚焦于数据与逻辑上。这种功能实现的核心是响应式的数据绑定，即数据变化后，绑定该数据的视图监听到了数据的变化，从而发生相应的改变。

作为一个前端开发者，刚接触小程序时，会有一种非常熟悉的感觉，因为它的响应式的数据绑定和现在流行的前端框架React.js、Vue.js等实现的效果几乎一模一样，所以上手非常快。

小程序使用的技术和常规的前端开发类似，但由于运行环境的不同，所以又有一些区别。

JavaScript: 小程序的js运行环境是微信App的上下文，一般js的运行环境是浏览器Browser或是Nodejs。由于微信中没有dom的概念，所以不可以操作传统浏览器中的DOM元素，也不能通过 Node.js 相关接口访问操作系统 API。因为微信本质上还是类似于浏览器的，所以，虽然开发中用到的技术和一般前端开发中用到的差不多，但实际上并不是html5，因为html5是国际标准，而小程序只是由html演变而来的框架。

WXML: 作为微信小程序的结构层，是应用的结构框架，为了配合整个框架，并不是使用的HTML，而是用的类似于HTML语法的描述语言。

WXSS: 是小程序的样式层。WeiXin Style Sheets是一套样式语言，和css之于html一样，wxss 用来描述WXML的组件应该怎么展示。

微信小程序最大的好处是它只在微信中运行，不需要做其他设备的适配，也不会出现在这台设备中显示正常，在另外一台设备中无法运行的情况。小程序虽然是一个闭源，且在封闭形态下的前端开发技术，但由于有微信这一个巨大的后盾，现在正在越来越流行。

https是具有安全性的ssl加密传输协议，不同于http的80端口，一般使用443端口，是由SSL+HTTP协议建立的可进行加密传输、身份认证的网络协议，相比http，https传输都是密文，更加安全，但由于需要进行加密解密，会比http慢一点，不过和安全性相比，这点速度的牺牲是完全有必要的。

**结语**

经测试，JavaScript训练出来的模型可以识别26个字母及其大小写，并且有着非常好的识别率，并可在小程序体验版“神经网络1字母识别”中使用。

参考文献

（1）[1] justadudewhohacks.Opencv4nodejs文档[OL]. https://github.com/justadudewhohacks/opencv4nodejs，日期．

（2）[2] Cazala.synapptic.synaptic文档[OL]<https://github.com/cazala/synaptic>,日期

（3）[1]UjjwalKarn.A Quick Introduction to Neural Networks [OL]. https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/，日期．

（4）[1] Mark Feng. 浏览器使用synaptic.js训练简单的神经网络推荐系统[OL]. https://github.com/markselby9/ml-in-browser，日期．

（5）[1] 微信公众平台. 小程序开发文档[OL]. https://developers.weixin.qq.com/miniprogram/dev/index.html?t=2018518，日期．

（6）[1] 作者. 浏览器使用synaptic.js训练简单的神经网络推荐系统[OL].http://…，日期．

附录：

1.Emnist：

// MNIST digits

var MNIST = [];

// 图片尺寸 28 x 28

var size = 28;

// 原始数据

var raw = [

require('./data/1.json').data,

require('./data/2.json').data,

require('./data/3.json').data,

require('./data/4.json').data,

require('./data/5.json').data,

require('./data/6.json').data,

require('./data/7.json').data,

require('./data/8.json').data,

require('./data/9.json').data,

require('./data/10.json').data,

require('./data/11.json').data,

require('./data/12.json').data,

require('./data/13.json').data,

require('./data/14.json').data,

require('./data/15.json').data,

require('./data/16.json').data,

require('./data/17.json').data,

require('./data/18.json').data,

require('./data/19.json').data,

require('./data/20.json').data,

require('./data/21.json').data,

require('./data/22.json').data,

require('./data/23.json').data,

require('./data/24.json').data,

require('./data/25.json').data,

require('./data/26.json').data

];

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 ,22, 23, 24, 25, 26].forEach(function (id) {

// mnist digit

var digit = {

id: id - 1

};

// raw data

digit.raw = raw[digit.id];

// 数量取28的倍数

digit.length = digit.raw.length / (size \* size) | 0;

digit.get = function (\_which) {

var which = \_which;

if ('undefined' == typeof which || which > digit.length || which < 0) {

which = Math.random() \* digit.length | 0;

}

var sample = [];

for (

var length = size \* size,

start = which \* length,

i = 0;

i < length;

sample.push(digit.raw[start + i++])

);

return sample;

}

digit.range = function (start, end) {

if (start < 0)

start = 0;

if (end >= digit.length)

end = digit.length - 1;

if (start > end) {

var tmp = start;

start = end;

end = tmp;

}

var range = [];

for (

var i = start;

i <= end;

range.push(digit.get(i++))

);

return range;

}

digit.set = function (start, end) {

var set = [];

var output = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0, 0 ];

output[digit.id] = 1;

var range = digit.range(start, end);

for (

var i = 0;

i < range.length;

set.push({

input: range[i++],

output: output

})

);

return set;

}

MNIST.push(digit);

});

MNIST.set = function (\_training, \_test) {

var training = \_training / 10 | 0;

var test = \_test / 10 | 0;

if (training < 1)

training = 1;

if (test < 1)

test = 1;

if (training + test + 1 > MNIST.\_\_MINLENGTH) {

console.warn('There are not enough samples to make a training set of ' + training + ' elements and a test set of ' + test + ' elements.');

if (training > test) {

test = MNIST.\_\_MINLENGTH \* (test / training);

training = MNIST.\_\_MINLENGTH - training;

}

else {

training = MNIST.\_\_MINLENGTH \* (training / test);

test = MNIST.\_\_MINLENGTH - test;

}

}

var trainingSet = [];

var testSet = [];

for (var i = 0; i < 26; i++) {

trainingSet = trainingSet.concat(MNIST[i].set(0, training - 1));

testSet = testSet.concat(MNIST[i].set(training, training + test - 1));

}

return {

training: shuffle(trainingSet),

test: shuffle(testSet)

}

}

// 打乱数据

function shuffle(v) {

for (var j, x, i = v.length; i; j = parseInt(Math.random() \* i), x = v[--i], v[i] = v[j], v[j] = x);

return v;

};

/\*\* 导出 \*\*/

// CommonJS & AMD

if (typeof define !== 'undefined' && define.amd) {

define([], function () { return MNIST });

}

// Node.js

if (typeof module !== 'undefined' && module.exports) {

module.exports = MNIST;

}

// Browser

if (typeof window == 'object') {

(function () {

var old = window['mnist'];

MNIST.ninja = function () {

window['mnist'] = old;

return MNIST;

};

})();

window['mnist'] = MNIST;

}

2.app-train

const fs = require('fs')

/\* 准备数据 \*/

const emnist = require('./emnist')

const set = emnist.set(2002, 26) // 2002 训练 78 测试

const trainingSet = set.training

const testSet = set.test

/\* 构建网络 \*/

const synaptic = require('synaptic')

const Layer = synaptic.Layer

const Network = synaptic.Network

const Trainer = synaptic.Trainer

const inputLayer = new Layer(784)

const hiddenLayer = new Layer(150)

const outputLayer = new Layer(26)

inputLayer.project(hiddenLayer)

hiddenLayer.project(outputLayer)

const myNetwork = new Network({

input: inputLayer,

hidden: [hiddenLayer],

output: outputLayer

})

console.log('start train...')

const trainer = new Trainer(myNetwork)

trainer.train(trainingSet, {

rate: 0.1, // 学习率

interations: 1000, // 迭代次数

error: 0.1, // 最小错误

shuffle: true, // 随机排序

log: 1, //

cost: Trainer.cost.CROSS\_ENTROPY

})

console.log('finish train...')

// 保存训练好的网络

let standalone = myNetwork.standalone();

fs.writeFile("./net3.js", standalone, function(err) {

if (err) {

console.log(err)

} else {

// logger.info("The file was saved!")

console.log("The file was saved!");

}

});

1. app-train

const http = require('http')

const fs = require('fs')

const express = require('express')

const bodyParse = require('body-parser')

const multiparty = require('multiparty')

const cv = require('opencv4nodejs')

const router = express.Router()

const app = express()

app.use(bodyParse.json());

app.use(bodyParse.urlencoded({extended:true}))

let netData = fs.readFileSync('./nets/net0-03.js', 'utf8')

const net = eval('(' + netData + ')');

router.post('/imgdata', (req, res) => {

const form = new multiparty.Form()

form.parse(req, function(err, fields, files) {

if (err) {console.log(err)}

cv.imreadAsync(files.data[0].path,cv.IMREAD\_GRAYSCALE,(err, img) => {

img = img.flip(1); // 镜像旋转 0 x 1 y -1 x+y

let rotate = cv.getRotationMatrix2D(new cv.Point(13.5,13.5), 90) // 逆90

img= img.warpAffine(rotate)

let arr = img.getDataAsArray(); // mat -> arr

// 二维 -> 一维

arr = arr.join(',').split(',')

for (let i = 0; i < arr.length; i++) {

arr[i] = Number(Math.abs((Number(arr[i]-255))/255).toFixed(3));

}

// 识别

let ret = net(arr)

res.json({'err':0,'msg':ret})

})

})

})

app.use('/api', router);

http.createServer(app).listen(8081, function() {

console.log('listening port 8081')

})

1. digitsLoader

const Q = require('q')

module.exports = function (labelFileName = './data/emnist-letters-train-images-idx3-ubyte') {

const fs = require('fs'),

deferred = Q.defer();

digits = [];

// fs,readStream

const stream = new fs.ReadStream(labelFileName);

let ver = 0, digitCount = 0, x = 0, y = 0, start = 0;

stream.on('readable', function () {

let buf = stream.read();

if (buf) {

if (ver != 2051) {

ver = buf.readInt32BE(0);

console.log(`DB digits Version: ${ver}`);

digitCount = buf.readInt32BE(4);

console.log(`Total digits: ${digitCount}`);

x = buf.readInt32BE(8);

y = buf.readInt32BE(12);

console.log(`x x y: ${x} x ${y}`);

start = 16;

}

for (let i = start; i< buf.length; i++) {

digits.push(buf.readUInt8(i));

}

start = 0;

}

});

stream.on('end', function () {

deferred.resolve(digits);

});

return deferred.promise;

}

1. labelsLoader

const Q = require('q')

module.exports = function (labelFileName = './data/emnist-letters-train-labels-idx1-ubyte') {

const fs = require('fs'),

deferred = Q.defer();

labels = [];

// fs,readStream

const stream = new fs.ReadStream(labelFileName);

let ver = 0, labelCount = 0, start = 0;

stream.on('readable', function () {

let buf = stream.read();

if (buf) {

if (ver != 2049) {

ver = buf.readInt32BE(0);

//console.log(`DB Labels Version: ${ver}`);

labelCount = buf.readInt32BE(4);

//console.log(`Total labels: ${labelCount}`);

start = 8;

}

for (let i = start; i< buf.length; i++) {

labels.push(buf.readUInt8(i));

}

start = 0;

}

});

stream.on('end', function () {

//console.log(labels);

deferred.resolve(labels);

//console.log('finish');

});

return deferred.promise;

}

1. rawMaker

module.exports = function (labels,digits,count) {

let raw = [];

const imageSize = 28\*28,

normalize = function (num) {

if (num != 0) {

return Math.round(1000/(255/num))/1000;

} else {

return 0;

}

};

count = count || labels.length;

for (let i in labels) {

if (i >= count) {

break;

}

let start = i\*imageSize;

if (! Array.isArray(raw[labels[i]])) {

raw[labels[i]] = [];

}

let range = digits.slice(start,start+imageSize).map(normalize);

raw[labels[i]].push(...range);

if (i % 1000 == 0) {

console.log(`Pass ${i} items...`);

}

}

console.log(`Finish processing ${count} items...`);

return raw;

}

1. rawWriter

const Q = require('q')

module.exports = function (raw, digitsDir = './digits') {

const fs = require('fs');

for (let i in raw) {

console.log(`Start make "${i}.json with ${raw[i].length/(28\*28)} images"`);

let wstream = fs.createWriteStream(`${digitsDir}/${i}.json`);

wstream.write('{ "data": [' + raw[i].join(',') +']}');

wstream.end();

}

};

1. 前端核心代码

<view class="container">

<canvas class="canvas" style="width: 200px; height: 200px;"

disable-scroll="true" canvas-id="drawMap"

binderror="canvasErrorCb" bindtouchstart="drawStart" bindtouchmove='drawMove' bindtouchend="drawEnd" bindtouchcancel="drawCancle"></canvas>

<!-- 按钮 -->

<view class="menu-wrapper">

<button type="primary" size='default' bindtap="clearCtx">clear</button>

<button type="primary" size="default" bindtap="getAndUpload">submit</button>

</view>

<!-- 结果 -->

<view class="result-wrapper" wx:if="{{resArr.length == 26}}">

<view class="result-item" wx:for="{{resArr}}">

<text>{{item[0]}}:</text>

<text>{{item[1]}}</text>

</view>

</view>

</view>

//logs.js

Page({

data: {

ctx: null,

// drawFlag: false,

xy: [],

//

wordList: ['a','b','c','d','e','f','g','h','i','j','k','l','m','n',

'o','p','q','r','s','t','u','v','w','x','y','z'],

resArr: []

},

canvasErrorCb: function (e) {

console.log(e.detail.errMag)

},

onLoad: function () {

let ctx = wx.createCanvasContext("drawMap");

ctx.setLineWidth(15)

ctx.setLineCap('round')

ctx.setFillStyle('white')

ctx.fillRect(0, 0, 200, 200)

ctx.draw()

this.setData({

ctx: ctx

})

},

drawStart: function (e) {

this.data.xy = [e.touches[0].x, e.touches[0].y]

},

drawMove: function (e) {

let x1, y1;

x1 = e.touches[0].x;

y1 = e.touches[0].y;

let ctx = this.data.ctx,

xy = this.data.xy;

ctx.moveTo(xy[0], xy[1])

ctx.lineTo(x1, y1)

ctx.stroke()

ctx.draw(true)

this.data.xy = [x1, y1]

},

drawEnd: function (e) {

},

drawCancle: function (e) {

console.log('err')

console.log(e)

},

// 清空画布

clearCtx: function() {

let ctx = this.data.ctx;

this.setData({

resArr: []

})

ctx.setFillStyle('white')

ctx.fillRect(0, 0, 200, 200)

ctx.draw();

},

// 获取并上传

getAndUpload: function () {

let imgUrl = '',

that = this;

wx.canvasToTempFilePath({

destWidth: 28,

destHeight: 28,

canvasId: 'drawMap',

success: function (res) {

console.log(res.tempFilePath)

that.postImg(res.tempFilePath)

},

fail: function (err) {

console.log(err)

}

})

},

postImg: function (url) {

const that = this;

wx.uploadFile({

url: 'http://192.168.199.219:8081/api/imgdata',

filePath: url,

name: 'data',

success: function (res) {

console.log(res.data)

let data = JSON.parse(res.data);

if (data.err == 0) {

that.showRes(data.msg)

} else {

console.log('err in success')

}

},

fail: function (err) {

console.log(err)

}

})

},

// 结果处理

showRes(data) {

const that = this;

if (data.length === 26) {

let arr = [];

data.forEach(function(item, index) {

arr.push([that.data.wordList[index], item])

})

arr.sort(function(a1,a2) {

return a2[1] - a1[1]

})

this.setData({

resArr: arr

})

console.log(arr)

}

}

})