

2014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： | 沈谋全 |
| 起讫日期： | 2018.03-2018.06 |

2018年6月

基于人工神经网络的在线识别Web应用

摘 要

近年来，随着计算机技术的发展，图像识别的方法越来越多，让计算机拥有强大的识别能力也受到了人们的重视。本文针对人工神经网络的图像分类识别，主要设计并训练了一个用于识别包含大写或小写英文字母图片的模型，调整参数选出最优模型。接着，搭建服务器，整合模型和接口，完成后端配置。然后，开发微信小程序作为前端入口。最终，将应用部署上线

**关键词：**图像识别 人工神经网络 后端开发 前端开发

**The Research of Event-Triggered Control for Linear System**

**Abstract**

In recent years, as the scale of control systems has gradually become larger and more decentralized, the limitations of traditional periodic sampling methods have become increasingly apparent. In other words, when there are unstable fluctuations in the system, the frequent periodic transmission of data will lead to the computational burden and energy loss. In order to reduce the resource consumption and energy consumption, the research on the event trigger mechanism has received increasingly attention. This paper focuses on the stability of the event-triggered system when the trigger threshold is an exponential function for the linear system. It uses the Lyapunov theory and the linear matrix inequality to complete the stability analysis and obtains the design conditions of the controller. Then, consider the non-fragile estimation of the system after adding external disturbances on this basis, and compare the two different disturbances at this point. In addition, in order to avoid the occurrence of Zeno behavior, the lower bound of the event triggering interval is given. Finally, simulation examples show the feasibility and effectiveness of the conclusions.

**Key Words:**Event-Triggered;Exponential Stability;Linear Matrix Inequality;

Non-fragile Estimation

目 录

摘要…………………………………………………………………………………Ⅰ

ABSTRACT…………………………………………………………………………Ⅱ

[第一章 绪论 1](#_Toc515832666)

[1.1 研究背景及研究意义 1](#_Toc515832667)

[1.2 图像识别研究现状 1](#_Toc515832668)

[1.3 本文主要内容 2](#_Toc515832669)

[第二章 预备知识 4](#_Toc515832670)

[2.1 神经网络 4](#_Toc515832671)

[2.1.1 神经元 4](#_Toc515832672)

[2.1.2神经网络模型 5](#_Toc515832673)

[2.1.2神经网络的求解 6](#_Toc515832674)

[2.2 npm包介绍 9](#_Toc515832675)

[第三章 神经网络部分 10](#_Toc515832676)

[3.1 迭代过程 10](#_Toc515832677)

[3.1.1前向传播计算 10](#_Toc515832678)

[3.1.2反向传播计算 11](#_Toc515832679)

[3.2 构建网络 14](#_Toc515832680)

[3.2.1准备图片库 14](#_Toc515832681)

[3.2.2参数选择 15](#_Toc515832682)

[3.2.3训练 16](#_Toc515832683)

[3.2.4测试 19](#_Toc515832684)

[第四章 前后端开发 21](#_Toc515832685)

[4.1 后端部分 21](#_Toc515832686)

[4.1.1接口配置 21](#_Toc515832687)

[4.1.2模型配置 24](#_Toc515832688)

[4.1.2 https配置 24](#_Toc515832689)

[4.2 前端部分 24](#_Toc515832690)

[4.2.1 画板 24](#_Toc515832691)

[4.2.2 接入接口 25](#_Toc515832692)

[第五章 结论 27](#_Toc515832693)

[参考文献 28](#_Toc515832694)

[致谢 31](#_Toc515832695)

# 第一章 绪论

## 1.1 研究背景及研究意义

图像识别，是利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对象的技术。图像识别是人类的基本能力，但近年来，随着计算机技术的迅速发展，让计算机拥有和人类类似的识别能力受到了越来越多的人们的重视，这也是机器人和人工智能技术发展的前提。最简单的图像无非是字母，同时，字母在各个领域都有广泛应用，如：车牌识别，单词甚至英文文章的识别，因此，字母识别是一项具有实际应用的课题。

而随着互联网的迅猛发展，移动智能终端近乎人手一台，智能设备的普及也带动了第三方应用市场的发展，大量应用层出不穷。本文将图像识别与web应用相结合，开发出一款能够在线识别英文字母的应用，任何人通过手机打开就能使用。其中，关于算法的开发几乎都基于python、C、matlab等语言，本文开创性的使用JavaScript作为开发语言，并成功训练出模型，其前后端的开发语言也都是JavaScript，实现了真正意义上的全栈开发，并为之后的应用开发流程提供了指导作用。

## 1.2 图像识别研究现状

图像识别方法随着算法的发展而不断进步，目前，图像识别的主要方法有：统计识别法，结构识别法，模糊集识别法，支持向量机法，模板匹配法，神经网络法。

统计识别的理论基础是策理论，在决策理论基础上建立统一识别模型，对

要分类的图像进行统计分析，统计出图像的各类特征，再找出反映其类别的特征，最后进行分类。其主要技术有统计法、聚类分析法等。但是这种方法不能识别图像的结构关系。

结构识别法即句法识别，是对统计识别法的缺陷的补充。结构识别法用符号表现图像的特征，其结构是层次结构，把复杂图像分解成子图像，子图像再分解成更简单的图像，一直分解下去，直到分解为最简单的子模式，即模式基元。通过对模式基元的识别，进而识别子模式，最终识别复杂图像。

模糊集识别法的理论基础是模糊数学，先对图像进行模糊的判别，这时图像可能属于多个类别，当找到其他特征时在进行更精确的判别。模糊集识别法根据一定的模糊化规则将图像的纹理、形状等特征分别作为模糊变量。虽然单个模糊变量不能准确判断图像，多个变量可以一步一步筛选出精确的类别。这种方法使用不太精确的方式去判别事物，更加符合人类的思维。

支持向量机方法将图像特征向量映射到更高维度的空间，在新空间中建立一个参数阈值。首先判断图像的大类别，在阈值的两边再建立新的阈值，进行分类，对图像不断判别。

模板匹配法是一种比较算法，将标准模板与待识别图像放入分类器中进行相关运算，若两个信号出现了自相关，即出现了主峰值，这表明两个图像匹配，将该值作为分类器的判断规则，即可实现匹配。对要识别的图像做出一个标准模板，作为待识别图像的标准，然后将标准模板与待分类模板进行比较，从而判断属于哪一类。

神经网络法通过神经网络算法进行分类，神经网络通过学习，能够从原始图像的数据中找出图像特征，对图像进行分类。这种方法克服了统计识别法中的复杂性和模型选择的困难，其是一种非线性计算，不需要分清图像中的非线性关系，极大方便了图像的分类过程。

通过对比以上方法，本文使用神经网络法进行图像的识别。

## 1.3 本文主要内容

本文主要设计了一个神经网络，并使用反向传播算法，完成模型的训练。再搭建服务器，接入模型，完成后端服务的配置。最后开发微信小程序，接入后端接口，部署应用。

本文研究内容为以下所示：

第一章为绪论部分。主要介绍本文研究背景现状；简要说明了神经网络的发展。末尾介绍了本论文所做工作及意义。

第二章为本文的预备知识。介绍了本文研究中所必备的一些前置条件和基础前置知识，主要介绍了神经网络的运作原理和反向传播的步骤，为后文证明提供依据。

第三章通过一个例子具体分析了神经网络运行过程中数据的变化过程。

第四章主要完成了神经网络的构建，包括图片库的准备，参数的选择及配置，模型的训练，最后测试模型正确率。

第五章主要完成后端部分的工作，包括图片的处理，接口配置，模型接入，https配置。

第六章主要完成了前端部分的工作，包括画板的实现，接入后端接口。

第七章则是对上述内容作了一个简要概括。

# 第二章 预备知识

## 2.1 神经网络

### 2.1.1 神经元

神经元是神经网络中最为基础的结构，它的输入是向量，线性组合输入的每个维度的值，再和一个阈值进行对比，若大于该值，则输出1，否则就输出-1。感知机与神经元有几点相似之处：第一，多输入，单输出；第二，以一个阈值分隔不同输出。

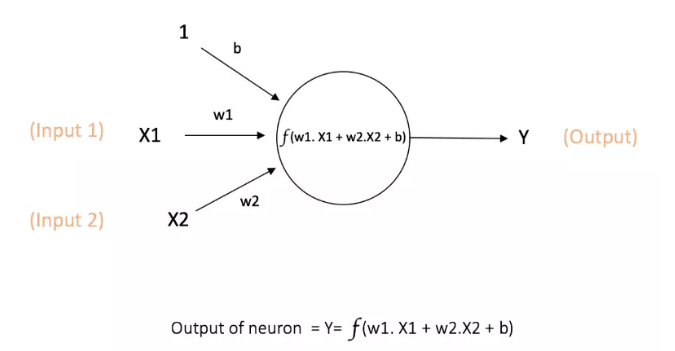


图2.1

神经元的结构里有两个最基本的成分：对输入向量的线性变换和对线性组合的结果进行阈值判断，实质是非线性变换。其实质上是仿射变换加一个非线性变换。而这个非线性变换，在机器学习领域中被叫做激活函数。

图2.1是一个以,和截距1作为输入的感知机，其输出为：

其中函数被称为激活函数。在本文中选用sigmoid函数作为激活函数：

其导数为

### 2.1.2神经网络模型

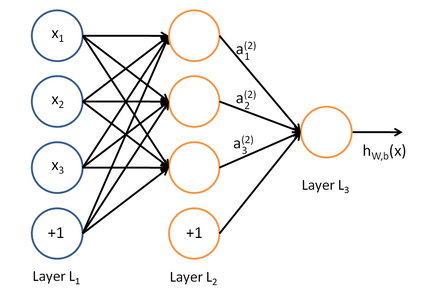


图2.2

许多单一神经元互相联结构成神经网络，一个神经元的输出是另一个神经元的输入，

如图2.2，最左边一层是输入层，最右边一层是输出层，中间左右的节点组成的一层是隐含层，标上“+1”的节点是偏置节点，也就是截距项。由于偏置节点不计入单元数，所以上例中有3个输入单元，3个隐含单元和1个输出单元。

一般用表示网络的层数，本例中，将第层记为，于是是输入层，是输出层。该神经网络有参数，其中是第层第单元与第层第单元之间的连接参数，即权重，是第层第单元的偏置项。偏置单元没有输入且总是输出。用表示第层的节点数，偏置单元不计入内。

用表示第层第单元的输出值，当时，，也就是第个网络输入值。对于给定参数集合，该网络可以按照公式（2,1）来计算输出结果，步骤如下：

以上计算步骤被称为前向传播。

除了图2.2的网络结构，神经网络还可以包含多个隐含层，即第1层是输入层，第层是输出层，中间的每个层与层紧密相连。这种模式下，也可以按照公式（2,1）计算，逐一计算第层的所有激活值，然后计算层，以此类推，直到第层。网络也可以有多个输出单元。由于这种联接没有闭环和回路，所以称为前馈神经网络。

### 2.1.2神经网络的求解

假设有一个样本集，包含个样例，则可以使用批量梯度下降法求解神经网络。对于单个样例，其代价函数为：

上式是一个方差代价函数。给定一个包含个样例的数据集，可以定义整体代价函数为：

公式（2,3）的第一项是均方差项，第二项是权重衰减项，其作用是减小权重的幅度，可防止过拟合。公式（2,2）是针对单个样例计算得到的方差代价函数，公式（2,3）是整体样本的代价函数，包含权重衰减项。

为了求解神经网络，需要针对参数和来求函数的最小值。先将每一个参数和初始化为一个接近零的随机值，之后对目标函数使用如梯度下降法的最优化算法。梯度下降法中每一次迭代都按照以下公式对参数进行更新：

式中是学习速率。其中的关键步骤是计算偏导数，反向传播算法是计算偏导数的一种有效算法。

由公式（2,3）（2,4）（2,5）可得：

由上式可知，只要先求出单个样例的代价函数的偏导数，就可推导出整体代价函数的偏导数。

反向传播算法的思路如下：给定一个样例，先进行前向运算，计算出网络中所有的激活值，然后针对第层的每一个节点，可计算出残差，表明了改节点对最终值的残差产生了多少影响。对于最终的输出节点，可直接算出网络的激活值与实际值之差，将这个差距定义为。而对于隐含节点，将基于节点残差的加权平均值计算，具体步骤如下：

1. 利用前向传导公式（2,1），得到各层的激活值
2. 对于输出层，即第层的每个输出，可根据以下公式计算残差：
3. 对于的各层，第层的第个节点的残差可按照下式计算：

将公式（2,6）中与替换为与，可得：

以上逐次从前向后求导的过程即为反向传导。

1. 计算偏导数

至此，整体代价函数的偏导已可求得，将其带入公式（2,4）（2,5）即可求得每次迭代后更新的权重参数，进而迭代完成。

## 2.2 npm包介绍

npm是Node.js 的包管理器，它是js的开源库，以下是本文中用到的主要包。

1.Synaptic

Synaptic是一个用于node.js和浏览器的神经网络库，它由原生js构建，无需任何依赖，可以构建和训练出任何一阶或二阶网络。

2.opencv4nodejs

Opencv4nodejs提供了全面的Nodejs绑定到OpenCV和OpenCV-contrib模块的API接口，可以使nodejs操作本地opencv，弥补了js在计算机视觉方面功能不足的缺点。

3.express

Express是最流行的基于nodejs的服务端框架，路由功能强大，性能优越，可扩展性强，几乎90%以上的node服务都基于express。

# 第三章 神经网络部分

## 3.1 迭代过程

为进一步解释神经网络运作原理，下面给出一个简单的网络迭代示例。如图3.1所示，该网络具有3个输入，2个隐含和2个输出。本例中使用单个训练集，给定输入分别为，期望输出分别为0.1,0.9，学习率此处给定为。一开始初始化时需要初始化和为随机值，这里给出的随机值分别为:

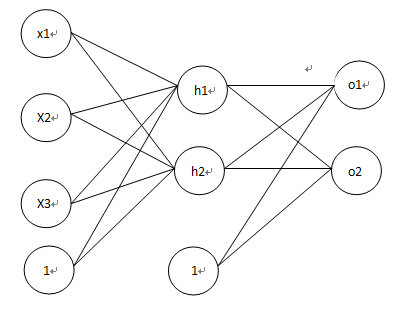


图3.1

### 3.1.1前向传播计算

首先计算隐含层输入

然后使用sigmoid函数进行压缩

接下来计算实际的网络输出，和上方操作类似

下面计算总误差

### 3.1.2反向传播计算

对：

先求

下面求

最后求

结合式（3.1）（3.2）（3.3）可得：

从当前权重中减去该值获得新的权重

重复以上步骤后，获得新的权重如下：

隐含层

对

其中，因为h1影响到了两个输出

下面开始计算

得

同理

因此

继续求得

综上

则权重可更新为

同理

至此，所有权重已更新完毕，一轮迭代完成。重复数次，每次迭代后的偏差都会减小，当其减小至设定值时，则停止迭代，训练完成。

## 3.2 构建网络

### 3.2.1准备图片库

本文需要识别英文字母，所以需要26个英文字母的大小写手写图片数据。

关于神经网络，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，每张图片的长宽都是28像素，这些图片是由不同人手写的0到9的数字。

神经网络的训练离不开大量的数据，经过数十年的发展，已经积累了各式各样的数据集。我在以下网址中找到了英文字母的数据集：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。下载解压后里面是一系列压缩文件，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。

二进制的数据无法直接拿来用，因此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，可以将所有二进制数据转换成26个json文件，每个json对应一个字母。

node --max\_old\_space\_size=2048 emnist --count 100

由于处理的数据量比较大，可能会超过node默认的允许最大内存值，所以需要手动设置最大内存为一个比较大的值，这里设为了2G。--count表示为每个字母最多转化100个图片数据，所以转化后，最多有2600个图片数据。图3.2是生成的json列表。

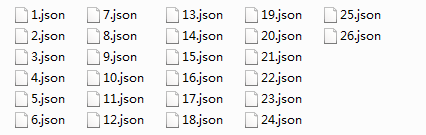


图3.2

这时数据虽然能被读取到，可还是不方便用。考虑到方便性和灵活性，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，只需要引入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。

### 3.2.2参数选择

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，隐含层会减小误差同时也会增加计算量，考虑到本文的问题并不是特别复杂，所以我在这里设置了1层隐含层。

隐含层中的节点数也需要设置，隐含层节点数会直接影响到模型的性能，训练时会出现过拟合一般都是它的原因。对于隐含层节点数的选择，目前有一些公式可以大概估算出来，可是不同公式算出来的结果相差比较大，现在也没有一个精确的方法能计算出来，所以一般是先计算出一个大概的范围，然后根据实际情况，在满足计算精度的情况下，取尽可能少的点数。在本文中，结合一些经验公式和具体情况，隐含层节点取100.

另外，还有一些其他参数需要设置，在一开始，设一个大概的值，在之后悬链的时候，会根据训练的时长和模型的效果等情况进行调整。

学习率rate：0.1

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。

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，有近两千张图片，最大迭代次数设大一点比较好。

最小错误率表示当网络的误差小于该值时，则代表训练完成，就会停止训练，一般设为0.1。

### 3.2.3训练

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。

我是用PuTTY远程连接服务器的，连接后只能以命令行的形式操作。先用FTP工具将资源文件和代码上传到服务器上，再用node运行程序。

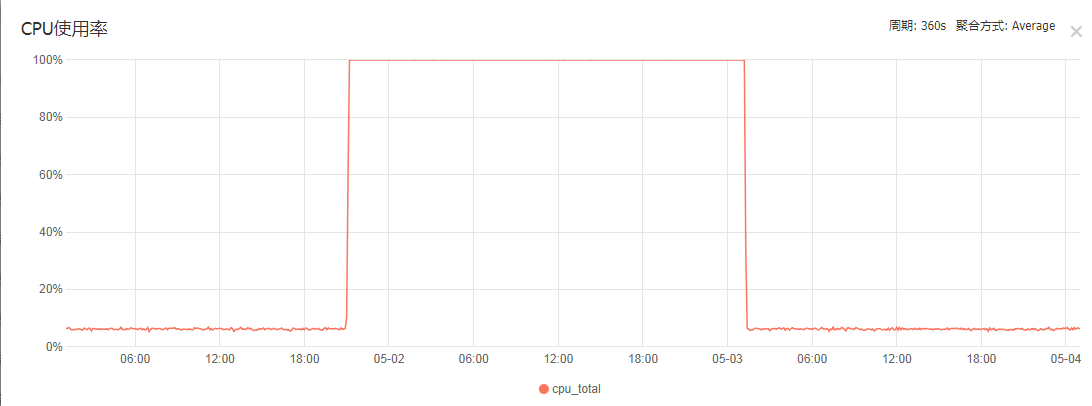
可是由于是远程连接，一旦断开连接，当前跑的程序就会中断，因此为了让程度能在服务器上不间断的跑，需要用到守护进程，这里我选择forever。forever的作用是让node服务能在服务器上长时间的运行，forever有一个自己的进程，每守护一个程序，forever就会在自己的进程下开一个子进程，只要主进程不挂，子进程就不会有事。

Forever还有一些配置项，这里我用了以下配置：

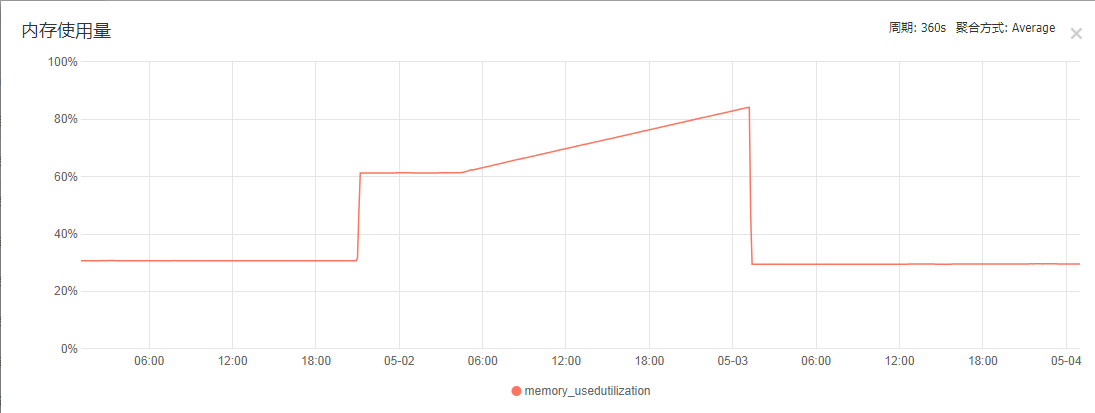
forever start -o train.log -m 1 app-train.js

意思是开始执行app-train.js文件，将输出打印到日志文件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。

下图是一次完整训练过程的监控图表，该云服务器的配置是1核2GB内存，系统是CentOS 7.3。可以看到CPU是一直是满负荷运行，说明训练时消耗的计算量是非常大的。内存占用量一开始不变，之后开始缓慢上升，这是明显的内存泄漏，由于某些原因js的垃圾回收机制并未回收某些资源导致内存占用越来越大。







从图中可以看出是在训练一段时间后内存才开始上升，所以应该是所用库synaptic的缺陷。这个问题有两种应对方法，一是增大内存，只要内存足够大，泄漏的这点内存就可以忽略不计，但这显然是治标不治本；第二种方法可以从源头解决问题，就是修改源码，找出泄漏的源头，一般是引用类型使用完后没有赋值为null。我已将问题反映给作者，期待他的回复。

训练过程中，每经过一次迭代，都会打印一条日志，如：

……

iterations 156 error 0.863792625747795 rate 0.07

iterations 157 error 0.7917011813679228 rate 0.07

iterations 158 error 0.7152523206229313 rate 0.07

iterations 159 error 0.6988286124934737 rate 0.07

iterations 160 error 0.6068983196793305 rate 0.07

iterations 161 error 0.5854829950364042 rate 0.07

……

意思是当前迭代次数，当前错误率，学习率。通过日志，可以看出训练的情况，从而调整参数优化网络。

经过调整了多次参数并分别进行了训练，我总结了一下几条规律：

1. 整体来看，错误率是随着迭代次数增加而减小的，而且错误率越小，下降的越慢。
2. 一次完整的训练一般会有上百次迭代，初始错误率一般在十几到几十之间，初始错误率越大，迭代次数越多，需要的时间也越长。
3. 样本数量与单次迭代所需的时间呈正相关；初始错误率与学习率呈正相关。
4. 训练时间 = 迭代次数 × 单次迭代所需的时间，一次完整的训练一般需要一到两天。

### 3.2.4测试

训练完成后生成的模型是一个js文件，6M左右，里面是大量的网络参数。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。

let data = fs.readFileSync('./nets/net0-07.js', 'utf8')

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

function getMaxIndex(arr) {

let max = 0,

ret = -1;

arr.forEach(function(item, index) {

if (item > max) {

max = item;

ret = index

}

})

return {max: max, i: ret}

}

let num = 0, // 总数

suc = 0， // 成功数

now = new Date();

testSet.forEach(function(item, index) {

let ret = net(item.input);

let res1 = getMaxIndex(ret)

res2 = getMaxIndex(item.output)

num++;

if (res1.i === res2.i) {

suc++;

}

})

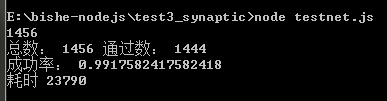
console.log('总数：', num, '通过数：', suc)

console.log('成功率：', suc/num)

console.log('耗时', new Date() - now)

取出测试数据后就可以通过以上代码测试，将输出测试总个数，正确识别数，识别成功率，以及总耗时。、

为了统一，我将测试个数设为1456个。



上图测试的是学习率为0.07，训练个数为2080训练出来的模型，从图中可以看出两点信息：

1. 该模型准确率是相当的高。
2. 单条数据的平均识别时间为16.3毫秒，几乎可以说是实时出结果，速度是相当的快。

评价一个模型的好坏无非从准确率和速度两点出发，因此该模型可以说是非常成功。

下表为其他配置相同，仅学习率不同的情况下训练出来的模型的准确率：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学习率 | 测试总数 | 正确总数 | 测试耗时(ms) | 正确率(%) |
| 0.01 | 1456 | 1411 | 25335 | 96.90934 |
| 0.03 | 1456 | 1449 | 25044 | 99.51923 |
| 0.07 | 1456 | 1444 | 23790 | 99.17582 |

# 第四章 前后端开发

本应用的结构如图4.1所示：



图4.1

## 4.1 后端部分

### 4.1.1接口配置

在写接口之前，需要明确一下需求：前端返回图像的二进制数据，后端返回json格式的识别结果，接口需要遵循RESTful规范。

根据以上条件，可以分析出以下结论。

1. 数据的交互中有二进制数据和json数据，因此不能使用get方式，最好使用post请求。
2. 本文中的接口遵守RESTful规范，因此需要将接口路由统一配置，接口名尽可能的语义化，因此将接口定为/api/imgdata

由于前端传进来的是长宽28像素的二进制图像数据，不同于一般的请求，所以需要使用multiparty这个npm包来获取图像数据。在nodejs中，要使用一个包，必须在一开始用require引入进来，因此需要把multiparty引入后并保存为变量以供之后使用。

const multiparty = require('multiparty')

在接口中，需要实例化一个multiparty对象

const form = new multiparty.Form()

取到请求数据后使用form.parse(res,cb(err,fileds,files))，回调函数的第三个参数files中就存放着二进制数据。通过files.data[0].path能成功取到二进制数据。

由于模型的输入是1x784的一维数组，所以还需要一个二进制转数组的步骤。这里用到了另一个关键的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。

同样，使用前需要先引入：

const cv = require('opencv4nodejs')

通过cv.imreadAsync(filePath,flags,callback(err,img))读取数据，并在回调函数中进行进一步处理。其中，filePath是图片数据，flags是读取方式，为了方便，这里取cv.IMREAD\_GRAYSCALE，表示读取后先进行灰度化处理。

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。

在回调函数中，图像数据是cv中特有的Mat格式，之后所有对它的操作都基于Mat。后端接口不仅要实现需求，还要尽可能的考虑其他异常情况。为了防止前端图像尺寸传错，需要在一开始加一层保险，即对图像重新设置大小为28像素。使用getDataAsArray函数将Mat转化为28x28的矩阵，再将矩阵进行操作拉伸成1x784的矩阵。

到现在为止，仅仅是把数据转化成了符合要求的格式，可是里面的值要进行怎样的处理还不清楚。这里使用反推法，对训练数据执行以上相反的操作，转化为图片显示出来。

下图是原始的g，可以看出原始的数据并不是正的，而是先将字母g进行y轴镜像翻转，再顺时针旋转了90度。

D:\Documents\大四下\毕设文档\论文\imgs\g.png

因此在接口中也要执行相同的操作：

img = img.flip(1);

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

img= img.warpAffine(rotate);

此外，训练数据中的值都在0-1之间，所以还需要进行二值化，可对每个值进行以下操作：

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

整个转化流程如下：

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，每项的值代表对应字母的可能性，越接近1，则越可能是该字母。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。

### 4.1.2模型配置

由于模型是一个可以独立运行的js文件，但在一个js文件里并不能直接执行另一个js文件，光是读取并没有什么用。所以这里需要使用eval语句，先用nodejs的模块以utf8的格式读取模型文件为字符串，再使用eval将字符串解析为语句并执行，具体写法如下：

let netData = fs.readFileSync('./nets/net0-07.js', 'utf8')

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

其中，net就是读取到的模型，只要将其放入接口中，传入数据，就会返回1x26的识别结果。

### 4.1.2 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。

先以utf8格式读取这两个文件，引入https模块，在以https模块启动服务的时候，传入以上两个文件的配置。

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。

## 4.2 前端部分

### 4.2.1 画板

前端部分需要实现的主要功能是用户手写字母，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，它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，接下来一共有三步：

1. 点击、按下事件、

当鼠标点击或手指按下时，保存当前点的坐标，并初始化画图的配置，画笔颜色、粗细，平滑度等。

1. 移动事件

获取移动时点的坐标，与之前保存的坐标连接，笔触移动到新的坐标上，并且用新坐标覆盖原来保存的坐标。每触发一次移动事件就执行一次以上操作。

1. 松开事件

中断轨迹，结束绘制。

除了绘制操作，还需要清空画布的功能。清空画布有几种实现方法，这里使用覆盖法。即画一个与画布等高宽的白色背景的矩形，这样之前画的所有轨迹就都被覆盖掉了。至此，一个简单的画板就实现了。

### 4.2.2 接入接口

在接接口之前，需要获取画布内容。小程序提供的wx.canvasToTempFilePath方法可以将指定大小的画布区域中的内容生成图片，并返回图片路径。本文中需要将整个画布生成28像素宽高的图片。

获取到图片路径后需要调接口上传图片数据，小程序提供了uploadFile方法，以post方式发送请求，接口地址为<https://sadxu.top/api/imgdata>，

wx.uploadFile({

url: 'https://sadxu.top/api/imgdata',

filePath: url,

name: 'data',

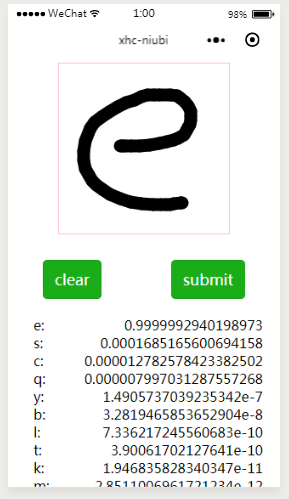
success: function (res) {},

fail: function (err) {}

})

接口的返回分为成功和失败两种情况，失败时在fail中处理，一般给出提示；成功时在success中处理，取到返回体中的数据，是1x26的一维数组，再将这些数据展示到界面上，即每个字母对应的识别率是多少。

小程序截图如下：



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。

# 第五章 结论

本论文是关于使用神经网络进行图片识别应用的开发。内容主要为三个部分：第一部分通过分析具体情况完成神经网络的设计和训练，根据训练情况调整参数，最后获得最优模型。第二部分是将模型嵌入后端，根据实际情况，设计接口。第三部分是完成微信小程序的开发，接入后端接口。一个应用，这三大部分环环相扣，缺一不可。

全栈开发是当今广受讨论的理念，该应用的开发全部基于JavaScript，不光是前后端，还包括算法部分，实现了真正意义上的全栈。本文展现了js近乎全能的优势，也在一定程度上推动了应用开发方式的变革。

将神经网络用在图片识别上，获得了不错的效果，但还有一些缺点，相信在未来，对于图片识别会有更好的算法。

# 参考文献

[1] Gupta, R.A., Chow, M.Y.. Networked control system:overview and research trends [J].IEEE Trans. Ind. Electron.2010,(57): 2527–2535.

[2] Yang, T.. Networked control system: a brief survey [J].IEE.Proc.-Control Theory Appl. 2006,(153) : 403–412.

[3] Walsh, G.C.,Ye, H.,Bushnell, L.G.. Stability analysis of networkedcontrol systems [J].IEEE Trans. Control Syst. Technol.2002,(10) : 438–446.

[4] Raji, R.S.. Smart networks for control [J].IEEE Spectr.1994,(31) : 49–55.

[5] Shi, P.. Filtering on sampled-data systems with parametricuncertainty [J].IEEE Trans. Autom. Control.1998,(43) : 1022-1027.

[6] Fridman, E.. Seuret, A., Richard, J.P., Robust sampled-datastabilization of linear systems: an input delay approach [J].Automatica.2004, (40) : 1441–1446.

[7] Fridman, E.. Arefined input delay approach to sampled-data control [J].Automatica.2010,(46) : 421–427.

[8] Gao, H., Wu, J., Shi, P.. Robust sampled-data H∞ controlwith stochastic sampling [J].Automatica.2009,(45) : 1729–1736.

[9] Arzen, K.E.. A simple event-based PID controller [J].Proc.IFAC World Congr.1999,(18): 423–428.

[10] Dorf R C,Farren M,Phillips C. Adaptive sampling frequency for sampled-datacontrol systems[J].IRE Transactions on Automatic Control,1962,7(1):38-47.

[11] Tabuada, P.. Event-triggered real-time scheduling of stabilizingcontrol tasks [J].IEEE Trans. Autom.2007, Control (52):1680–1685.

[12] Zhang, J., Feng, G.. Event-driven observer-based outputfeedback control for linear systems [J].Automatica.2014,(50): 1852–1859.

[13] Zhang, H., Peng, C., Zhang, J., Zhang, C.. Event-triggeredcontrol in networked control systems: a survey[A].IEEE 27thChinese Control and Decision Conference[C].2015, 3092–3097.

[14] Peng, C., Yang, T.. Event-triggered communication and H∞control co-design for networked control systems [J].Automatica2013,(49): 1326–1332.

[15] Lunze, J., Lehmann, D..Astate-feedback approach to eventbasedcontrol [J].Automatica.2010,(46): 211–215.

[16] Yue, D., Tian, E., Han, Q.. A delay system method fordesigning event-triggered controllers of networked controlsystems [J].IEEE Trans. Autom. Control.2013,(58): 475–481.

[17] Kim, D.S., Lee, Y.S., Kwon, W.H., Park, H.S.: Maximumallowable delay bounds of networked control systems [J].ControlEng. Pract.2003,(11): 1301–1313.

[18] Xiong, J., Lam, J.. Stabilization of linear systems over networkswith bounded packet loss [J].Automatica.2007,(43): 80–87.

[19] Donkers, M.C.F., Heemels, W.P.M.H.. Output-based eventtriggeredcontrol with guaranteed-gain and improved anddecentralized event-triggering [J].IEEE Trans. Autom. Control.2012,(57): 1362–1376.

[20] Yu,H., Antsaklis, P.J.. Event-triggered output feedback controlfor networked control systems using passivity: achievingL2 stability in the presence of communication delays

and signal quantization [J].Automatica.2013,(49):30–38.

[21] Peng, C., Zhang, J.. Event-triggered output-feedback H∞control for networked control systems with time-varyingsampling [J].IET Control Theory Appl.2015,(9):1384–1391.

[22] Li, H., Fu, M..‘A linear matrix inequality approach to robust H∞ filtering’[J].IEEE Trans. Signal Process. 1997, (45): 2338–2350

[23] Kim, J.H.. ‘Delay-dependent robust H∞ filtering for uncertain discrete-time singular systems with interval time-varying delay’[J].Automatica, 2010, (46):

591–597

[24] Du, D., Qi, B., Fei, M., et al.. ‘Multiple event-triggered H2/H∞ filtering for hybrid wired-wireless networked systems with random network-induced

delays’, Inf. [J].Sci.2015, (325): 393–408

[25] Shen, M.. ‘H∞ filtering of continuous Markov jump linear system with partlyknown Markov modes and transition probabilities’[J]. Franklin Inst. 2013,(350): 3384–3399

[26] Yang, G., Guo, X., Che, W., et al.: ‘Linear systems: non-fragile control and filtering’ (Taylor & Francis, Boca Raton, FL, USA, 2013)

[27] Du, H., Lam, J., Sze, K.Y..‘Design of non-fragile H∞ controller for active vehicle suspensions’ [J]. Vib. Control. 2005, (11): 225–243

[28] Chang, X., Yang, G.. ‘Nonfragile H∞ filtering of continuous-time fuzzy systems’[J].IEEE Trans. Signal Process. 2011, (59):1528–1538

[29] Lu, R., Liu, S., Peng, H., Y., et al. ‘Non-fragile filtering for fuzzy stochastic systems over fading channel’.[J].Neurocomputing. 2016, (174): 553–559

[30] Yang, G., Che, W.. ‘Non-fragile H∞ filter design for linear continuous-time systems’. [J].Automatica, 2008, (44): 2849–2856

[31] Ding, D., Li, X., Wang, Y., et al..‘Non-fragile H∞ fuzzy filtering for discretetime non-linear systems’[J]. IET Control Theory Appl. 2013, (7): 848–857

[32] Zhang, D.. Shi, P., Wang, Q., et al.. ‘Distributed non-fragile filtering for T–S fuzzy systems with event-based communications' [J]. Fuzzy Sets Syst. 2017,(306).137–152

[33] Park, P., Ko, J.W., Jeong, C.. Reciprocally convex approachto stability of systems with time-varying delays [J].Automatica.2011,(47):235–238

[34] Wang, Y., Xie, L., de Souza, C.E.. Robust control of a classof uncertain nonlinear systems [J].Syst. Control Lett.1992,(19): 139–149.

[35] de Souza, C.E.. Robust stability and stabilization of uncertaindiscrete-time Markovian jump linear systems [J].IEEETrans. Autom. Control.2006,(51): 836–841

[36] Ding, D., Li, X., Wang, Y., et al.. ‘Non-fragile H∞ fuzzy filtering for discretetimenon-linear systems’[J],IET Control Theory Appl.2013,(7):848–857.

[37] Shu, Z., Lam, J., Xiong, J.. ‘Non-fragile exponential stability assignment ofdiscrete-time linear systems with missing data in actuators’[J].IEEE Trans.Autom. Control.2009,(54),625–630.

[38] Lu, R., Liu, S., Peng, H., Y., et al. ‘Non-fragile filtering for fuzzy stochasticsystems over fading channel’[J].Neurocomputing. 2016, (174):553–559.

[39] 俞立.鲁棒控制-线性矩阵处理方法[M].北京：清华大学出版社，2002.

[40] 刘豹,唐万生. 线代控制理论[M].北京：机械工业出版社，2006.

# 致谢

在此论文完成之际，忠心感谢导师沈谋全副教授对我的教诲和耐心指导。沈老师知识渊博，见解独到，其认真的科学态度，严谨的治学精神，精益求精的研究作风，无一不感染与激励着我，令我钦佩不已。本文从最初课题的选择、开题、研究到论文的完成，沈老师给予了我耐心的指导，整个过程中的一点一滴都离不开沈老师的悉心帮助。除此之外，沈老师对考研等方面的了解和指导也使我受益匪浅，进一步明确了自己的方向。在此，向沈老师表达我的敬意和感谢！同时十分感谢吴银峰同学在论文完成期间的交流与资料共享，帮助我对课题有更深入的了解。

。

