

# 南京工业大学

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  戴盛              |
| 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，随着移动互联网的发展，图片逐渐成为了传达信息的主要媒介。伴随着图片的增多，人们的检索效率也随着降低，此时，图像识别技术显得尤为重要。本文使用 BP 神经网络方法实现图像识别，首先介绍了神经网络算法的原理和计算步骤，然后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，完成后端开发，再开发微信小程序作为前端入口，完成前端开发，最终，将应用部署上线。该应用的主要开发语言为 JavaScript，使用前后端分离的开发模式，一定程度上展现了该语言的强大和灵活性。结果显示，使用 BP 神经网络进行图像识别具有较高的识别率。

**关键词：**图像识别 人工神经网络 后端开发 前端开发

## **Online Web Application Based on Artificial Neural Network**

### **Abstract**

In recent years, with the development of mobile Internet, pictures have gradually become the main medium for conveying information. With the increase in the number of pictures, people's search efficiency has also decreased, at this time, image recognition technology is particularly important. This paper uses BP neural network method to realize image recognition. Firstly, the principle and calculation steps of the neural network algorithm are introduced. Then a model is designed and trained to identify English alphabetic pictures containing uppercase or lowercase, and the parameters are adjusted to select the optimal model. Then, build the server, integrate the model and interface, complete the back-end development, and then develop the WeChat applet as the front-end portal, complete the front-end development, and finally, deploy the application on-line. The main development language for this application is JavaScript. The separation of development modes before and after the use of the front and back ends demonstrates the power and flexibility of the language to some extent. The results show that using BP neural network for image recognition has a higher recognition rate.

**Key Words:** Image recognition; artificial neural network; back-end development; front-end development

## 目 录

|                     |    |
|---------------------|----|
| 摘要.....             | I  |
| ABSTRACT.....       | II |
| 第一章 绪论 .....        | 1  |
| 1.1 研究背景及研究意义 ..... | 1  |
| 1.2 图像识别研究现状 .....  | 1  |
| 1.3 本文主要内容 .....    | 2  |
| 第二章 实现机理.....       | 4  |
| 2.1 神经网络 .....      | 4  |
| 2.2 NPM 包介绍 .....   | 8  |
| 第三章 构造网络.....       | 10 |
| 3.1 迭代过程 .....      | 10 |
| 3.2 构建网络 .....      | 14 |
| 第四章 系统实现.....       | 21 |
| 4.1 后端部分 .....      | 21 |
| 4.2 前端部分 .....      | 24 |
| 第五章 结论 .....        | 28 |
| 参考文献 .....          | 29 |
| 附录.....             | 31 |
| 致谢.....             | 45 |



# 第一章 绪论

## 1.1 研究背景及研究意义

随着移动互联网的发展，图片信息的数量剧增，图片有着不受语言限制的优势，成为了互联网中信息传达的主要媒介<sup>[1]</sup>。然而伴随着图片的增多，难题也随之出现，当信息由图片记载时，人们无法对图片内容进行检索，对信息的检索效率随之降低。此时，就需要用到图像识别技术<sup>[2]</sup>。

图像识别，是利用计算机对图像进行处理和分析，来识别不同模式的对象的一种技术<sup>[3,4]</sup>。通过图像识别技术，人们不仅可以更快的搜索信息，还能让生活更加智能化。随着识别技术的不断提高，识别的精度和范围都大大提高，越来越多的科技公司通过这类技术开发出了多种多样的智能化产品<sup>[5]</sup>。

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，手写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<sup>[6]</sup>。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<sup>[7]</sup>，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

图像的识别主要是根据一定的规则，把具有某一特征的图像归为一类。其方法随着算法的发展而不断进步，目前，图像识别的方法主要有：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。

（1）统计识别法，其建立统一识别模型的基础是策理论<sup>[8]</sup>，对要分类的图像统计出各个特征，再在其中找出反映其类别的部分，之后进行分类。分类的主要技术有统计法、聚类分析法等。

（2）结构识别法，图像的特征用带有层次结构的符号表现出来，其原理是把图像进行多次分解，每次分解会获得原图像的子图像，直到子图像不可分解为止，即模式基元<sup>[9]</sup>。通过对最后的子图像识别，往上递归识别，最终识别出初始

图像。

(3) 模糊集识别法, 先对进行模糊判别, 判断图像可能属于哪几类, 接着找出其他特征, 进行进一步的判别, 当找到其他特征时在进行更精确的判别次, 持续多次, 直到只剩一个类别为止<sup>[10]</sup>。其特征可能是颜色、形状等, 特征越多, 识别就越准确。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。

(4) 支持向量机法, 该方法的主要思想是将图像的特征向量映射到更高维度<sup>[11]</sup>, 在这个超维度中进行类别的划分。一般来说维度越高, 划分就越具体, 识别也越准确。

(5) 模板匹配法, 将两个图像进行比较运算, 若出现了自相关, 即信号出现了主峰值, 则表明两个图像匹配, 将该值作为分类器的判断规则, 即可实现匹配<sup>[12]</sup>。

(6) 神经网络法, 神经网络通过学习, 能够找到输入与输出之间的非线性映射, 从而实现分类。该方法不需要分清图像中的非线性关系<sup>[13]</sup>, 使得图像的分类过程变得非常方便。

对比以上方法, 由于神经网络法避免了复杂的特征提取, 仅需调整参数就可以获得较高的识别率, 其算法本身也有较高的成熟度, 因此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<sup>[14]</sup>, 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; 简要对比了图像识别的各种方法。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, 为后文的使用提供依据。

第三章首先具体解释了神经网络的迭代过程, 接着完成了网络的构建, 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。



第四章主要是该应用的开发部分，包括后端及前端。后端部分的工作是图片的处理，接口配置，模型接入，https 配置。前端部分的工作是画板的实现和接入后端接口。

第五章则是对上述内容作了一个简要概括。

## 第二章 实现机理

### 2.1 神经网络

#### (1) 神经元

神经元是神经网络中最为基础的结构<sup>[15]</sup>，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。

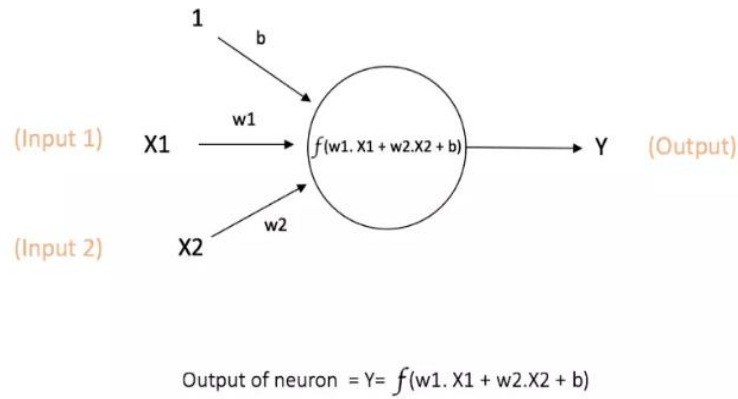


图 2-1

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<sup>[16]</sup>。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  $x_1$ ,  $x_2$  和截距 1 作为输入的感知机，其输出为：

$$Y = h_{W,b}(x) = f(W^T x) = f\left(\sum_{i=1}^2 W_i x_i + b\right) \quad (2,1)$$

其中函数  $f$  被称为激活函数。在本文中的激活函数使用 sigmoid 函数：

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

其导数为

$$f'(z) = 1 - (f(z))^2$$

## （2）神经网络模型

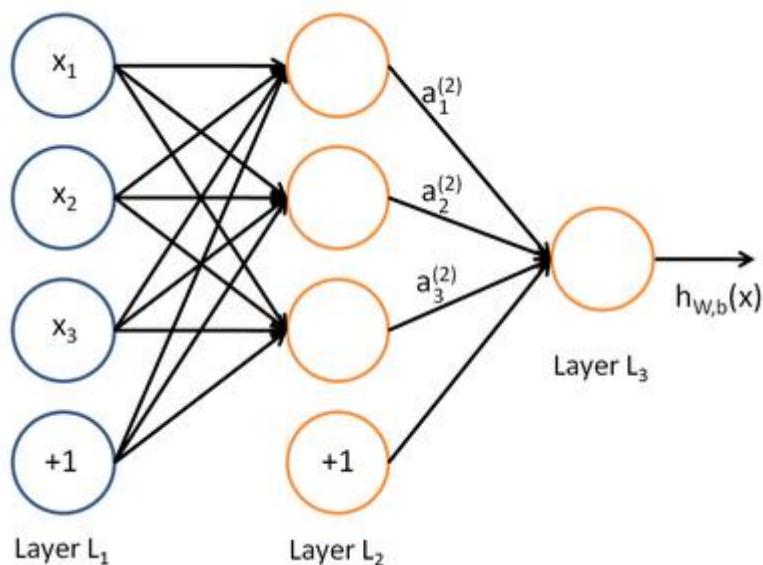


图 2-2

神经网络是由许多单一神经元互相联结构成的，一个神经元的输出作为另一个神经元的输入<sup>[17]</sup>。

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。

网络的层数一般用  $n1$  表示，将第  $l$  层记为  $L_l$ ，于是  $L_1$  是输入层， $L_{n_l}$  是输出层，在本例中  $n1 = 3$ 。该神经网络中有参数  $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ ，其中  $W_{ij}^{(l)}$  是第  $l$  层第  $j$  单元与第  $l + 1$  层第  $i$  单元之间的连接参数，即权重， $b_i^{(l)}$  是第  $l + 1$  层第  $i$  单元的偏置项。偏置单元没有输入且总是输出 +1。用  $S_l$  表示第  $l$  层的节点数，偏置单元不计入节点数内<sup>[18]</sup>。

用  $a_i^{(l)}$  表示第  $l$  层第  $i$  单元的输出值，当  $l = 1$  时，第  $i$  个网络输入值为  $a_i^{(1)} = x_i$ 。对于给定参数集合  $W, b$ ，该网络可以按照公式（2, 1）来计算输出结果，步骤如下：

$$\begin{aligned} a_1^{(1)} &= x_1, a_2^{(1)} = x_2, a_3^{(1)} = x_3 \\ a_1^{(2)} &= f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}) \\ a_2^{(2)} &= f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)}) \end{aligned}$$

$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$

$$Y = h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

以上计算步骤被称为前向传播。

除了图 2-2 的网络结构,还可以包含多个隐含层,即第一层和最后一层不变,中间的每个层之间紧密相连。由于这种联接没有闭环也没有回路,所以被称为前馈神经网络。

### (3) 神经网络的求解

假设有一个样本集 $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ , 包含 $m$ 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 $(x, y)$ , 其代价函数为:

$$J(W, b; x, y) = \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (2,2)$$

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

$$\begin{aligned} J(W, b) &= \left[ \frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^{(l)})^2 \quad (2,3) \\ &= \left[ \frac{1}{m} \sum_{i=1}^m \left( \frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^{(l)})^2 \end{aligned}$$

公式 (2, 3) 的第一项是均方差项, 第二项是权重衰减项。公式 (2, 2) 是针对单个样例计算得到的方差代价函数<sup>[19]</sup>, 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, 需要求函数 $J(W, b)$ 的最小值。先将每个参数 $w_{ij}^{(l)}$ 和 $b_i^{(l)}$ 初始化为一个接近零的随机值, 之后再使用如梯度下降的最优化算法<sup>[20]</sup>。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:

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha \frac{\partial}{\partial w_{ij}^{(l)}} J(W, b) \quad (2,4)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \quad (2,5)$$

式中 $\alpha$ 是学习速率。其中计算偏导数是关键步骤，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：

$$\alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) = \left[ \frac{1}{m} \sum_{i=1}^m \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x^{(i)}, y^{(i)}) \right] + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b) = \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial b_i^{(l)}} J(W, b; x^{(i)}, y^{(i)})$$

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<sup>[21, 22]</sup>：给定一个样例 $(x, y)$ ，先进行前向运算，计算出所有的激活值，接下来计算第一层每个节点的残差，表明了第1层的每一个节点 $i$ 对最终值的残差的影响程度。然后，算出网络的激活值与实际值之差 $\delta_i^{(n_l)}$ ，作为输出节点的残差。而隐含节点的残差比较复杂，需要基于节点残差的加权平均值计算 $\delta_i^{(l)}$ ，具体步骤如下：

1. 利用前向传导公式 (2, 1)，得到2 -  $n_1$ 各层的激活值
2. 对于输出层，即第 $n_l$ 层的每个输出，可根据以下公式计算残差：

$$\begin{aligned} \delta_i^{(n_l)} &= \frac{\partial}{\partial z_i^{(n_l)}} J(W, b; x, y) = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 \\ &= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - a_j^{(n_l)})^2 = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - f(z_j^{(n_l)}))^2 \\ &= -(y_i - f(z_i^{(n_l)})) \cdot f'(z_i^{(n_l)}) = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \end{aligned}$$

3. 对于 $l = n_l - 1, n_l - 2, n_l - 3, \dots, 2$ 的各层，第 $l$ 层的第 $i$ 个节点的残差可按照下式计算：

$$\begin{aligned} \delta_i^{(n_l-1)} &= \frac{\partial}{\partial z_i^{(n_l-1)}} J(W, b; x, y) = \frac{\partial}{\partial z_i^{(n_l-1)}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 \\ &= \frac{\partial}{\partial z_i^{(n_l-1)}} \frac{1}{2} \sum_{j=1}^{S_{n_l}} (y_j - a_j^{(n_l)})^2 = \frac{1}{2} \sum_{j=1}^{S_{n_l}} \frac{\partial}{\partial z_i^{(n_l-1)}} (y_j - a_j^{(n_l)})^2 \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2} \sum_{j=1}^{S_{n_l}} \frac{\vartheta}{\vartheta_{Z_i^{(n_l-1)}}} \left( y_j - f(z_j^{(n_l)}) \right)^2 = \sum_{j=1}^{S_{n_l}} - \left( y_i - f(z_i^{(n_l)}) \right) \cdot \frac{\vartheta}{\vartheta_{Z_i^{(n_l-1)}}} f'(z_j^{(n_l)}) \\
&= \sum_{j=1}^{S_{n_l}} - \left( y_i - f(z_i^{(n_l)}) \right) \cdot \frac{\vartheta_{Z_i^{(n_l)}}}{\vartheta_{Z_i^{(n_l-1)}}} \cdot f'(z_j^{(n_l)}) = \sum_{j=1}^{S_{n_l}} \delta_j^{(n_l)} \cdot \frac{\vartheta_{Z_i^{(n_l)}}}{\vartheta_{Z_i^{(n_l-1)}}} \\
&= \sum_{j=1}^{S_{n_l}} \left( \delta_j^{(n_l)} \cdot \frac{\vartheta}{\vartheta_{Z_i^{(n_l-1)}}} \sum_{k=1}^{S_{n_l-1}} f(z_k^{(n_l-1)}) \cdot W_{jk}^{(n_l-1)} \right) \\
&= \sum_{j=1}^{S_{n_l}} \delta_j^{(n_l)} \cdot W_{ji}^{(n_l-1)} \cdot f'(z_i^{(n_l-1)}) \\
&= \left( \sum_{j=1}^{S_{n_l}} W_{ji}^{(n_l-1)} \delta_j^{(n_l)} \right) f'(z_i^{(n_l-1)}) \tag{2,6}
\end{aligned}$$

将公式 (2,6) 中  $n_l-1$  与  $n_l$  替换为  $l$  与  $l+1$ , 可得:

$$\delta_i^{(l)} = \left( \sum_{j=1}^{S_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \tag{2,7}$$

以上逐次从前向后求导的过程即为反向传导。

#### 4. 计算偏导数

$$\begin{aligned}
\frac{\vartheta}{\vartheta W_{ij}^{(l)}} J(W, b; x, y) &= a_j^{(l)} \delta_i^{(l+1)} \\
\frac{\vartheta}{\vartheta b_i^{(l)}} J(W, b; x, y) &= \delta_i^{(l+1)}
\end{aligned}$$

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  $W, b$ , 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 构造网络

### 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  $x_1 = 0.1, x_2 = 0.2, x_3 = 0.3$ ，期望输出分别为 0.1, 0.9，学习率  $\alpha$  此处给定为 0.4。一开始初始化时需要初始化  $W_{ij}^{(l)}$  和  $b_i^{(l)}$  为随机值，这里给出的随机值分别为：

$$W_{11}^{(1)} = 0.05, W_{12}^{(1)} = 0.1, W_{13}^{(1)} = 0.15, W_{21}^{(1)} = 0.2, W_{22}^{(1)} = 0.25, W_{23}^{(1)} = 0.3$$

$$W_{11}^{(2)} = 0.35, W_{12}^{(2)} = 0.4, W_{21}^{(2)} = 0.45, W_{22}^{(2)} = 0.5,$$

$$b_1 = 0.3, b_2 = 0.6$$

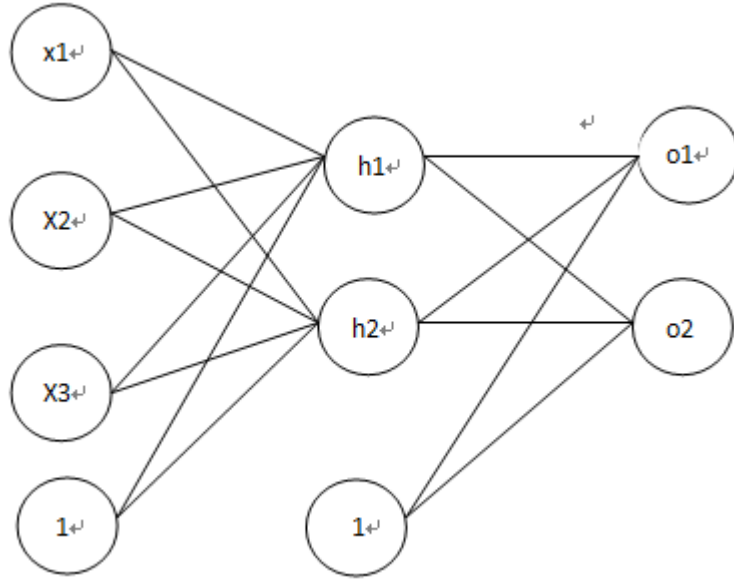


图 3-1

#### (1) 前向传播计算

首先计算隐含层输入

$$h_1 = b_1 + \sum_{i=1}^3 W_{1i}^{(1)} \cdot x_i = 0.3 + 0.05 \cdot 0.1 + 0.1 \cdot 0.2 + 0.15 \cdot 0.3 = 0.37$$

$$h_2 = b_1 + \sum_{i=1}^3 W_{2i}^{(1)} \cdot x_i = 0.3 + 0.2 \cdot 0.1 + 0.25 \cdot 0.2 + 0.3 \cdot 0.3 = 0.46$$

然后使用 sigmoid 函数进行压缩



$$\text{out}_{h1} = \frac{1}{1 + e^{-h_1}} = \frac{1}{1 + e^{-0.37}} = 0.591459$$

$$\text{out}_{h2} = \frac{1}{1 + e^{-h_2}} = \frac{1}{1 + e^{-0.46}} = 0.613014$$

接下来计算实际的网络输出，和上方操作类似

$$o_1 = b_2 + \sum_{i=1}^2 W_{1i}^{(2)} \cdot \text{out}_{hi} = 0.6 + 0.35 * 0.591459 + 0.4 * 0.613014 = 1.052216$$

$$o_2 = b_2 + \sum_{i=1}^2 W_{2i}^{(2)} \cdot \text{out}_{hi} = 0.6 + 0.45 * 0.591459 + 0.5 * 0.613014 = 1.172664$$

$$\text{out}_{o1} = \frac{1}{1 + e^{-o_1}} = \frac{1}{1 + e^{-1.052216}} = 0.741200$$

$$\text{out}_{o2} = \frac{1}{1 + e^{-o_2}} = \frac{1}{1 + e^{-1.172664}} = 0.763626$$

下面计算总误差

$$E_{o1} = \frac{1}{2} (\text{target}_{o1} - \text{out}_{o1})^2 = \frac{1}{2} (0.1 - 0.741200)^2 = 0.205569$$

$$E_{o2} = \frac{1}{2} (\text{target}_{o2} - \text{out}_{o2})^2 = \frac{1}{2} (0.9 - 0.763626)^2 = 0.009299$$

$$E = E_{o1} + E_{o2} = 0.205569 + 0.009299 = 0.214868$$

(2) 反向传播计算

对  $W_{11}^{(2)}$  :

$$\frac{\partial E}{\partial W_{11}^{(2)}} = \frac{\partial E}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial o_1} \cdot \frac{\partial o_1}{\partial W_{11}^{(2)}} \quad (3.1)$$

先求  $\frac{\partial E}{\partial \text{out}_{o1}}$

$$E = \frac{1}{2} (\text{target}_{o1} - \text{out}_{o1})^2 + \frac{1}{2} (\text{target}_{o2} - \text{out}_{o2})^2$$

$$\frac{\partial E}{\partial \text{out}_{o1}} = 2 * \frac{1}{2} (\text{target}_{o1} - \text{out}_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E}{\partial \text{out}_{o1}} = -(\text{target}_{o1} - \text{out}_{o1}) = -(0.1 - 0.741200)$$

$$\frac{\partial E}{\partial \text{out}_{o1}} = 0.6412 \quad (3.2)$$

下面求  $\frac{\partial \text{out}_{o1}}{\partial o_1}$

$$\text{out}_{o1} = \frac{1}{1 + e^{-o_1}}$$

$$\frac{\partial \text{out}_{o1}}{\partial o_1} = \text{out}_{o1}(1 - \text{out}_{o1}) = 0.7412 * (1 - 0.7412) = 0.191823 \quad (3.3)$$

最后求  $\frac{\partial o_1}{\partial W_{11}^{(2)}}$

$$o_1 = b_2 + \sum_{i=1}^2 W_{1i}^{(2)} \cdot x_i$$

$$\frac{\partial o_1}{\partial W_{11}^{(2)}} = \text{out}_{h1} = 0.591459 \quad (3.4)$$

结合式 (3.1) (3.2) (3.3) 可得：

$$\frac{\partial E}{\partial W_{11}^{(2)}} = 0.6412 * 0.191823 * 0.591459 = 0.072748$$

从当前权重中减去该值获得新的权重

$$W_{11}^{(2)+} = W_{11}^{(2)} - \alpha \frac{\partial E}{\partial W_{11}^{(2)}} = 0.35 - 0.4 * 0.072748 = 0.3209008$$

重复以上步骤后，获得新的权重如下：

$$W_{12}^{(2)+} = 0.369840$$

$$W_{21}^{(2)+} = 0.421661$$

$$W_{22}^{(2)+} = 0.470628$$

隐含层

对  $W_{11}^{(1)}$ ：

$$\frac{\partial E}{\partial W_{11}^{(1)}} = \frac{\partial E}{\partial \text{out}_{h1}} \cdot \frac{\partial \text{out}_{h1}}{\partial h_1} \cdot \frac{\partial h_1}{\partial W_{11}^{(1)}} \quad (3.5)$$

其中，因为  $h_1$  影响到了两个输出

$$\frac{\partial E}{\partial \text{out}_{h1}} = \frac{\partial E_{o1}}{\partial \text{out}_{h1}} + \frac{\partial E_{o2}}{\partial \text{out}_{h1}} \quad (3.6)$$

下面开始计算

$$\begin{aligned} \frac{\partial E_{o1}}{\partial \text{out}_{h1}} &= \frac{\partial E_{o1}}{\partial o_1} \cdot \frac{\partial o_1}{\partial \text{out}_{h1}} \\ \frac{\partial E_{o1}}{\partial o_1} &= \frac{\partial E_{o1}}{\partial \text{out}_{o1}} \cdot \frac{\partial \text{out}_{o1}}{\partial o_1} = 0.6412 * 0.191823 = 0.122997 \\ \frac{\partial o_1}{\partial \text{out}_{h1}} &= W_{11}^{(2)} = 0.35 \end{aligned}$$

得

$$\frac{\partial E_{o1}}{\partial out_{h1}} = 0.122997 * 0.35 = 0.043049$$

同理

$$\frac{\partial E_{o2}}{\partial out_{h1}} = 0.053903$$

因此

$$\frac{\partial E}{\partial out_{h1}} = 0.043049 + 0.053903 = 0.096952 \quad (3.7)$$

继续求得

$$\frac{\partial out_{h1}}{\partial h_1} = h_1(1 - h_1) = 0.37(1 - 0.37) = 0.2331$$

$$\frac{\partial h_1}{\partial W_{11}^{(1)}} = x_1 = 0.1$$

综上

$$\frac{\partial E}{\partial W_{11}^{(1)}} = 0.096952 * 0.2331 * 0.05 = 0.0011299$$

则权重可更新为

$$W_{11}^{(1)+} = W_{11}^{(1)} - \alpha \frac{\partial E}{\partial W_{11}^{(1)}} = 0.05 - 0.4 * 0.0011299 = 0.049548$$

同理

$$W_{12}^{(1)+} = 0.099096$$

$$W_{13}^{(1)+} = 0.148644$$

$$W_{21}^{(1)+} = 0.19773864$$

$$W_{21}^{(1)+} = 0.2571733$$

$$W_{23}^{(1)+} = 0.296608$$

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<sup>[23]</sup>，则停止迭代，训练完成。

## 3.2 构建网络

### (1) 准备图片库

本文需要识别英文字母，所以需要准备 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<sup>[24]</sup>，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 mnist --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<sup>[25]</sup>，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。

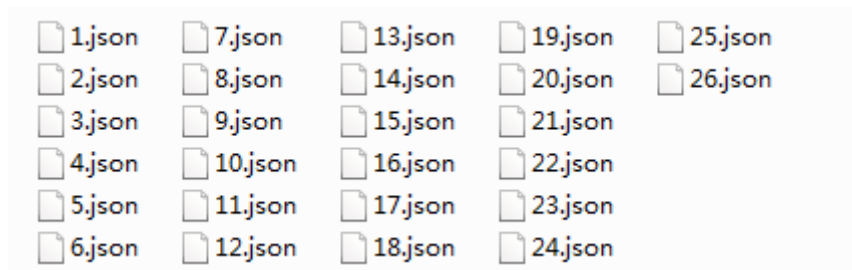


图 3-2

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。

## （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  $28 \times 28 = 784$ ，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<sup>[26]</sup>。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<sup>[27]</sup>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) 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。

图 3-3 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 一直是满负荷运行，说明训练时消耗的计算量是非常大的。

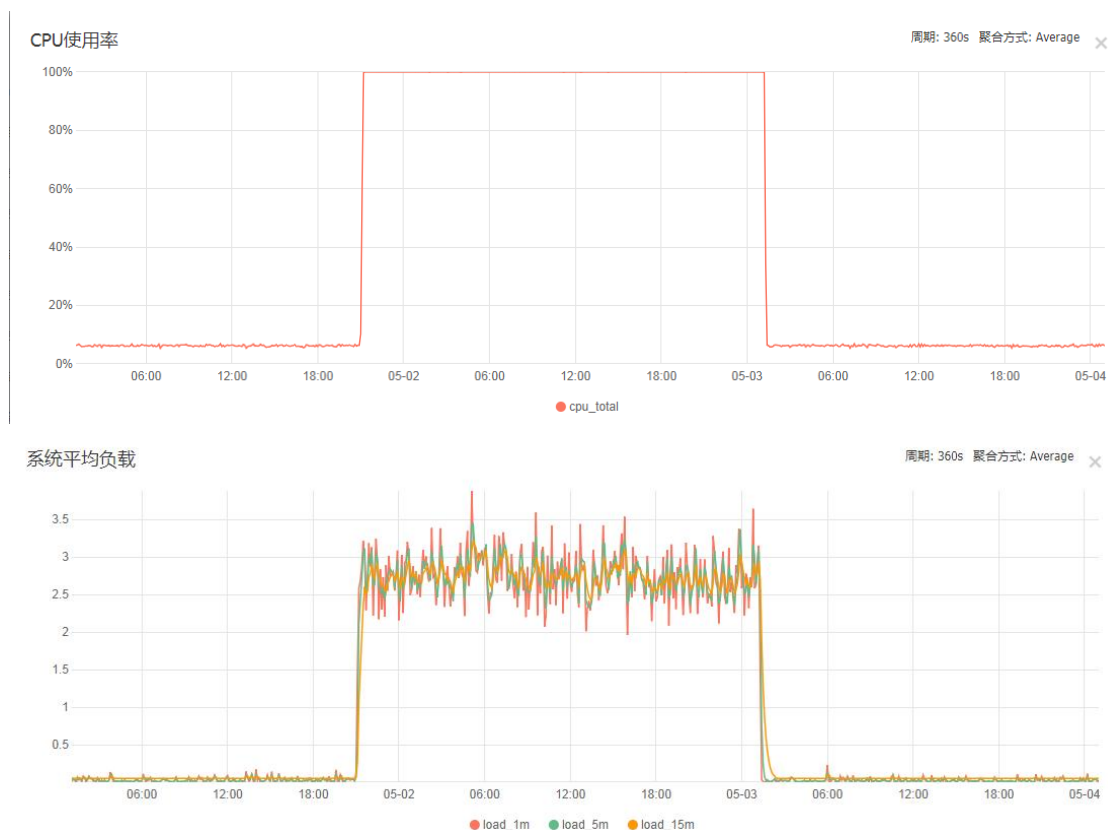


图 3-3

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。

#### (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
```



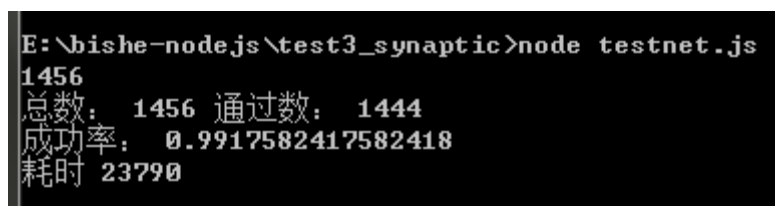
```
now = new Date();
testSet.forEach(function(item, index) {
    let ret = net(item.input);

    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。



The image shows a terminal window with the following text:

```
E:\bishe-nodejs\test3_synaptic>node testnet.js
1456
总数: 1456 通过数: 1444
成功率: 0.9917582417582418
耗时 23790
```

图 3-5

图 3-5 是学习率为 0.07，训练个数为 2080 训练出来的模型的测试结果，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。

表 3-1 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：

表 3-1

| 学习率  | 测试总数 | 正确总数 | 测 试 耗 时<br>(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 系统实现

本应用的结构如图 4-1 所示：



图 4-1

### 4.1 后端部分

#### （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<sup>[28]</sup>，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, fields, 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<sup>[29]</sup>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。

图 4-2 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。



图 4-2

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))
```

整个转化流程如图 4-3:

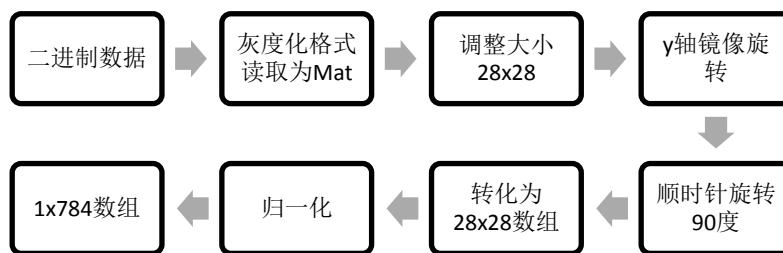


图 4-3

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。

## (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。

## (3) https 配置

配置 HTTPS 需要 https 证书<sup>[30]</sup>，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。

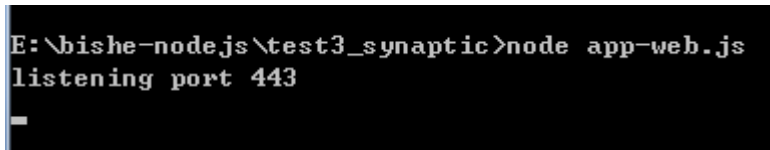
阿里云提供免费 HTTPS 证书，所谓的 https 证书其实就是几个文件。使用 Express 框架配置 https 只需要两个文件：private.key：私钥和 csr.pem：CSR 证书签名。

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。

准备完成后，先在本地启动服务，如图 4-4，说明服务启动完成。



```
E:\bishe-nodejs\test3_synaptic>node app-web.js
listening port 443
```

图 4-4

## 4.2 前端部分

### (1) 画板

前端部分需要实现的主要功能是用用户手写字母，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，并初始化画图配置，画笔颜色、粗细，平滑度等。

#### 2. 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。

#### 3. 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。

### (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。


小程序操作界面如图 4-3 所示:



图 4-3


最上方是画布区域，clear 按钮用来清除画布内容，submit 按钮用来提交图像。下方是数据识别率，表示各个字母的概率，按照由大到小排列，第一个即是最有可能的字母。

图 4-4 是实际使用时的截图，




clear
submit

|    |                       |
|----|-----------------------|
| b: | 0.9942672444290813    |
| d: | 0.6904207694742885    |
| a: | 0.007321862660862805  |
| f: | 0.0011361198258980686 |
| k: | 2.956893816498775e-7  |




clear
submit

|    |                         |
|----|-------------------------|
| b: | 0.9998769487987087      |
| d: | 0.00189995035995235     |
| g: | 0.00023292247117028556  |
| a: | 0.0001400678237260781   |
| h: | 0.000004307474665943853 |
| s: | 1.1607973846307583e-7   |
| l: | 5.405005244494763e-8    |



clear
submit

|    |                       |
|----|-----------------------|
| v: | 0.9995334799278456    |
| z: | 0.9939107838776092    |
| g: | 0.020221236384345927  |
| d: | 0.01028930042715226   |
| u: | 0.0008300590499232527 |



clear
submit

|    |                         |
|----|-------------------------|
| y: | 0.9999919226494695      |
| v: | 0.000007467798872837503 |
| k: | 4.733931766904967e-9    |
| z: | 1.5138829726284222e-9   |
| l: | 1.4286772687104911e-10  |
| i: | 3.516595874465714e-11   |
| x: | 7.369972732002534e-12   |



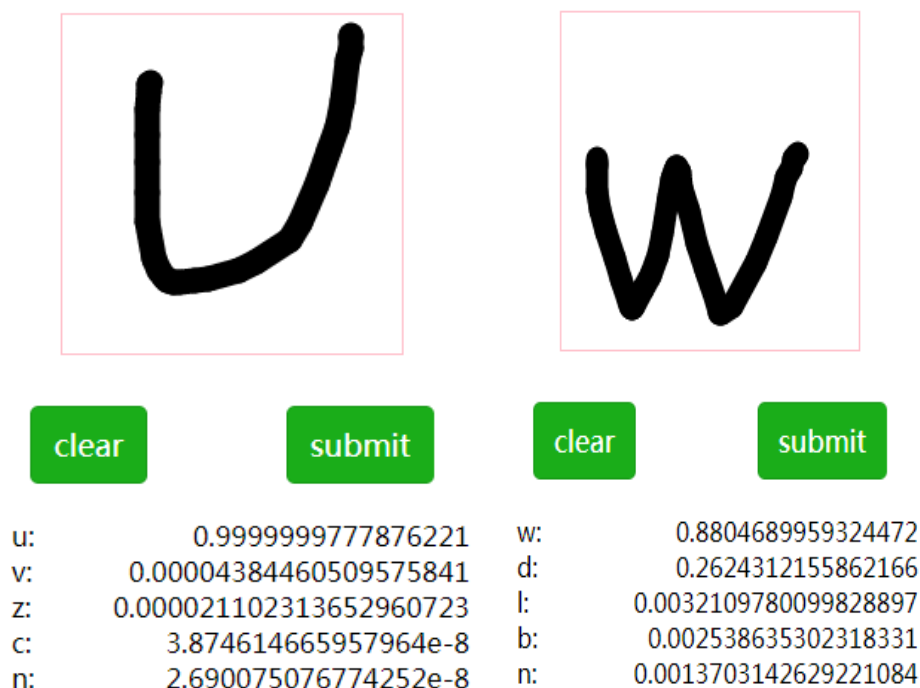


图 4-4

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。第二部分是模型嵌入后端，分析实际情况，设计相应接口。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。

该应用的主要开发语言为 JavaScript，不光是前后端，还包括算法部分，展现了 js 近乎全能的优势。另外，该应用使用了前后端分离的开发模式，也在一定程度上推动了应用开发方式的变革。

在测试模型时，网络的识别率非常高，但在实际应用中，识别率却有所降低，这是因为该网络训练时使用的图片库中的字母是由外国人写的，笔法与中国人相比有所偏差，因此会出现识别不准的情况。不过总体而言，将神经网络应用在图片识别上，获得了不错的效果，但还有一些缺点，比如相关参数调整困难、训练的时间较长等问题，相信在未来，对于图片识别会有更好的算法。

## 参考文献

- [1] Amitabha Mukhopadhyay. Application of visual, statistical and artificial neural network methods in the differentiation of water from the exploited aquifers in Kuwait[J]. Hydrogeology Journal, 2003, 11(3) : 211–215.
- [2] Nadia Nedjah, Ajith Abraham, Luiza M. Mourelle. Hybrid artificial neural network[J]. Neural Computing and Applications, 2007, 16(3) : 51–55.
- [3] Albert J. Dijkstra. Artificial Neural Network or Common Sense Based on Insight?[J]. Journal of the American Oil Chemists' Society, 2011, 88(7) : 14–15.
- [4] Ozge Gundogdu, Erol Egrioglu, Cagdas Hakan Aladag, Ufuk Yolcu. Multiplicative neuron model artificial neural network based on Gaussian activation function[J]. Neural Computing and Applications, 2016, 27(4) : 110–111.
- [5] M. Fellner, A. Delgado, T. Becker. Functional nodes in dynamic neural networks for bioprocess modelling[J]. Bioprocess and Biosystems Engineering, 2003, 25(5) : 3–4.
- [6] Abhishek Pandey, R. Prasad, J. K. Srivastava, S. N. Pandey. Retrieval of soil moisture by artificial neural network using X-band ground based data[J]. Russian Agricultural Sciences, 2012, 38(3) : 38–40.
- [7] Amit Kumar Gupta, Swadesh Kumar Singh, Swathi Reddy, Gokul Hariharan. Prediction of flow stress in dynamic strain aging regime of austenitic stainless steel 316 using artificial neural network[J]. Materials and Design, 2012, 35: 19–21.
- [8] Angad Singh, Avishek Majumder, Arun Goyal. Artificial intelligence based optimization of exocellular glucanase production from *Leuconostoc dextranicum* NRRL B-1146[J]. Bioresource Technology, 2008, 99(17) : 60–65.
- [9] Seniha Elemen, Emriye Perrin Akçakoca Kumbasar, Saadet Yapar. Modeling the adsorption of textile dye on organoclay using an artificial neural network[J]. Dyes and Pigments, 2012, 95(1) : 43–45.
- [10] Zhiye Zhao. Steel columns under fire—a neural network based strength model[J]. Advances in Engineering Software, 2005, 37(2) : 34–35.
- [11] Hikmet Esen, Filiz Ozgen, Mehmet Esen, Abdulkadir Sengur. Artificial neural network and wavelet neural network approaches for modelling of a solar air heater[J]. Expert Systems With Applications, 2009, 36(8) : 77–78.
- [12] H.M. Yao, H.B. Vuthaluru, M.O. Tadé D. Djukanovic. Artificial neural network-based prediction of hydrogen content of coal in power station boilers[J]. Fuel, 2005, 84(12) : 89–90.
- [13] Atika Qazi, H. Fayaz, A. Wadi, Ram Gopal Raj, N.A. Rahim, Waleed Ahmed Khan. The artificial neural network for solar radiation prediction and designing solar systems: a systematic literature review[J]. Journal of Cleaner Production, 2015, 104.
- [14] Gwo-Ching Liao. Hybrid Improved Differential Evolution and Wavelet Neural Network with load forecasting problem of air conditioning[J]. International Journal of Electrical Power and Energy Systems, 2014, 61.
- [15] Yong Soo Kim. Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size[J]. Expert Systems With Applications, 2006, 34(2).
- [16] R.K. Dash, N.K. Barpanda, P.K. Tripathy, C.R. Tripathy. Network reliability optimization problem of interconnection network under node-edge failure model[J]. Applied Soft Computing

Journal,2012,12(8) : 26–28.

[17]E. Won. A hardware implementation of artificial neural networks using field programmable gate arrays[J]. Nuclear Inst. and Methods in Physics Research, A,2007,581(3) : 30–31.

[18]Erdi Tosun,Ahmet Çalık. Failure load prediction of single lap adhesive joints using artificial neural networks[J]. Alexandria Engineering Journal,2016,55(2) : 54–25.

[19]Jun Xi,Yujing Xue,Yinxiang Xu,Yuhong Shen. Artificial neural network modeling and optimization of ultrahigh pressure extraction of green tea polyphenols[J]. Food Chemistry,2013,141(1) : 56–67.

[20]Chunqiao Mi,Jianyu Yang,Shaoming Li,Xiaodong Zhang,Dehai Zhu. Prediction of accumulated temperature in vegetation period using artificial neural network[J]. Mathematical and Computer Modelling,2009,51(11) : 65–67.

[21]Dipti Itchhaporia,Peter B. Snow,Robert J. Almassy,William J. Oetgen. Artificial neural networks: Current status in cardiovascular medicine[J]. Journal of the American College of Cardiology,1996,28(2) : 67–89.

[22]M.M. Rashidi,N. Galanis,F. Nazari,A. Basiri Parsa,L. Shamekhi. Parametric analysis and optimization of regenerative Clausius and organic Rankine cycles with two feedwater heaters using artificial bees colony and artificial neural network[J]. Energy,2011,36(9) : 90–101.

[23]Özgür Çelik,Ahmet Teke,H. Başak Yıldırım. The optimized artificial neural network model with Levenberg–Marquardt algorithm for global solar radiation estimation in Eastern Mediterranean Region of Turkey[J]. Journal of Cleaner Production,2016,116: 45–67.

[24]Hidayet Oğuz,Ismail Sarıtas,Hakan Emre Baydan. Prediction of diesel engine performance using biofuels with artificial neural network[J]. Expert Systems With Applications,2010,37(9).

[25]S. Riad,J. Mania. Rainfall-runoff model using an artificial neural network approach[J]. Mathematical and Computer Modelling,2004,40(7) : 90–98.

[26]Juarez Ortiz,Claudia G.M. Ghefter,Carlos E.S. Silva,Renato M.E. Sabbatini. One-year mortality prognosis in heart failure: A neural network approach based on echocardiographic data[J]. Journal of the American College of Cardiology,1995,26(7) : 45–49.

[27]G. Rajesh,S.K. Bhattacharyya. System identification for nonlinear maneuvering of large tankers using artificial neural network[J]. Applied Ocean Research,2008,30(4) : 21–54.

[28]Ramazan Özçelik,Maria J. Diamantopoulou,John R. Brooks,Harry V. Wiant. Estimating tree bole volume using artificial neural network models for four species in Turkey[J]. Journal of Environmental Management,2009,91(3) : 78–89.

[29]Ian Flood,Bryan T Bewick,Robert J Dinan,Hani A Salim. Modeling blast wave propagation using artificial neural network methods[J]. Advanced Engineering Informatics,2009,23(4).

[30]David Liu,Yudie Yuan,Shufang Liao. Artificial neural network vs. nonlinear regression for gold content estimation in pyrometallurgy[J]. Expert Systems With Applications,2009,36(7).

## 附录

```

1.digitsLoader.js
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;
}

```

```

2.labelsLoader.js
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;
}
```

### 3. rawMaker.js

```
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;
}

```

#### 4.rawWriter.js

```

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();
    }
};

```

#### 5.emnist\_dl.js

```
"use strict"
/* 生成数据 */
// node --max_old_space_size=2048 emnist --count 100

const co = require('co'),
      argv = require('optimist').argv,
      labelsLoader = require('./lib/labelsLoader'),
      digitsLoader = require('./lib/digitsLoader'),
      rawMaker      = require('./lib/rawMaker'),
      rawWriter     = require('./lib/rawWriter');

co(function* () {
  let labels = yield labelsLoader(),
      count = argv.count || labels.length,
      digits = yield digitsLoader();
  console.log('step1')
  console.log(labels.length);
  console.log(digits.length);
  // console.log(digits.length/(28*28));
  let raw  = rawMaker(labels,digits,count);
  rawWriter(raw);

}).catch(function (err) {
  console.error(err.stack);
});

6. app-train.js
const fs = require('fs')

/* 准备数据 */
const emnist = require('./emnist')
const set = emnist.set(2080, 26) // 2080 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: 0.09,
  iterations: 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("./net0-09.js", standalone, function(err) {
  if (err) {
    console.log(err)
  } else {
    // logger.info("The file was saved!")
    console.log("The file was saved!");
  }
});

7. emnist.js
// 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;
```



```
        i < range.length;
        set.push({
            input: range[i++],
            output: output
        })
    };
    return set;
}

MNIST.push(digit);
});

MNIST.set = function (_training, _test) {
    var training = _training / 26 | 0;
    var test = _test / 26 | 0;

    if (training < 1)
        training = 1;
    if (test < 1)
        test = 1;

    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;
}
```

```
8. testnet. js
const fs = require('fs')
const cv = require('opencv4nodejs')
const emnist = require('./emnist')
const set = emnist.set(2, 1456)

const trainingSet = set.training
const testSet = set.test

console.log(testSet.length)

let data = fs.readFileSync('./nets/net.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

let 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)
```

## 8. app-web. js

```
const https = require('https')
const fs = require('fs')
const express = require('express')
const bodyParser = require('body-parser')
const multipart = require('multipart')
const cv = require('opencv4nodejs')
const router = express.Router()
const app = express()

const httpsOptions = {
    key: fs.readFileSync('./https/214400255360454.key', 'utf8'),
    cert: fs.readFileSync('./https/214400255360454.pem', 'utf8')
};

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended:true}))
```

```

let netData = fs.readFileSync('./nets/net0-07.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);

https.createServer(httpsOptions, app).listen(443, function() {
  console.log('listening port 443')
})

9. index.wxml
<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>
</view>

```

```
</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>
</view>
```

10. index.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();
        ctx.setLineWidth(15)
        ctx.setLineCap('round')
    },
    // 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: 'https://sadxu.top/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)
  }
}
})
```

## 致谢

在此论文完成之际,非常感谢导师沈谋全副教授大学四年里对我的教诲和耐心指导。从我大一的时候沈老师就在督促我认真学习,若是没有沈老师的教导,我可能早就荒废这四年了,可以说没有沈老师,就没有现在的我。本文的完成,沈老师也给予了我耐心的指导和帮助。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,作为友好相处了四年的舍友,能交到这样一个朋友,真的很荣幸。