数学实验综合报告

基于卷积神经网络的英文字母识别

组员姓名：　　舒嗣嘉　张健锋

　　　　　　　郑明晓　张晓航

指导教师：　　弈旭明　教授

二〇一七年六月

摘要

卷积神经网络算法是近年来获得迅速发展的一种机器学习算法，它在计算机视觉和其他人工智能领域上有着其他机器学习算法无法比拟的优势。并且在数字、字母、符号识别领域已经得到了广泛的应用。近年来已经有许多科研工作者提出各种机器学习的方法来识别数字、英文字母、汉字。考虑到识别英文字母的最关键工作就是提取出不同英文字母独特的特征，而卷积神经网络在图片的特征提取上有非常高效并且鲁棒的特性，所以本文提出了一套基于卷积神经网络的手写英文字母识别方法，并利用 EMNIST 手写英文数据库训练和测试我们的模型。

关键词：深度学习；卷积神经网络；特征提取；手写英文字母识别

目录

1　理论基础

1.1　深度神经网络

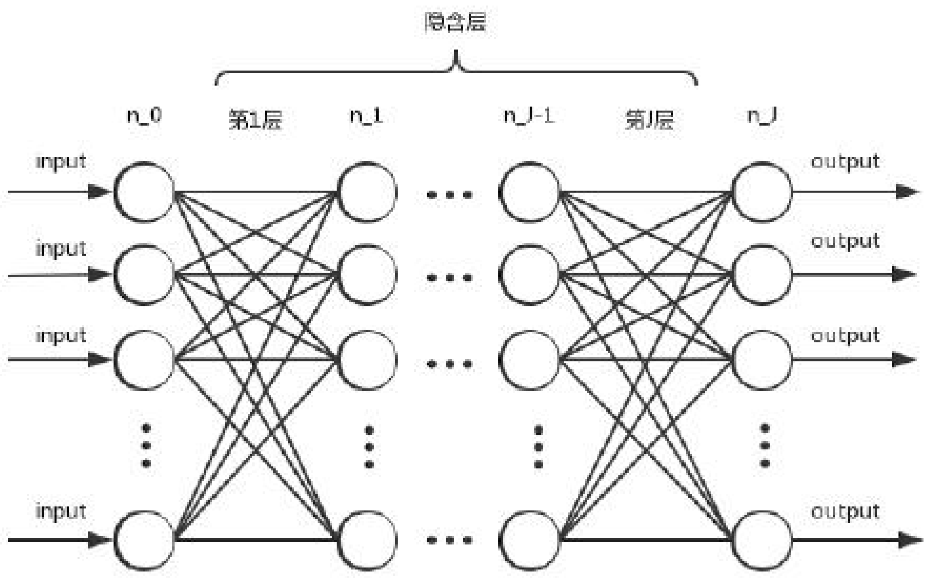


图 1　深度神经网络

深度神经网络即包含多层神经元的神经网络，它最明显的特征就是在输入和输出层之间还包含了许多的隐含层，如图 1 所示，这些层的存在使得网络可以记住更复杂的特征。

在深度神经网络中，输入的信息只能是一些简单的数据（向量、矩阵、张量），但是通过多层神经网络的“加工”后，这些信息就可以转化为复杂抽象的特征。有关多层神经网络是如何对低级信息“加工”成高级特征的，我们留到下一小节卷积神经网络时再做详细介绍。

设神经网络有 L 层，其中任何一层的输入都来自前一层输出与该层权重的积，即

$$\boldsymbol{I}^{l+1}=\boldsymbol{w}^{l+1}\boldsymbol{O}^{l} \qquad (1)$$

其中 $\boldsymbol{I}^{l+1}$ 和 $\boldsymbol{w}^{l+1}$ 分别表示第 i+1 层的输入和权重，$\boldsymbol{O}^{l}$ 表示第 l 层的输出。实际上，为了使系统更具有灵活性，通常在一层的输出与输入之间插入一个激活函数，它可以对网络节点的输出进行适当的修正。于是有

$$O_i^l=h^l(I_i^l),i=1,2,\ldots,N_l \qquad (2)$$

这里 h^l 为第 l 层的激活函数，N_l 为第 l 层的神经节点数。

接下来我们将讨论深度神经网络的参数更新。首先关注输出层，因为在输出层，我们不仅知道输出层经过网络得到的输出，而且还知道该层所期望的输出（即训练样本的标签）。我们令损失函数为

$$E=\frac{1}{2}\sum_{k=1}^{N}\left(\boldsymbol{r}-\boldsymbol{O}^{L}\right)^{2} \qquad (3)$$

其中 N 为训练样本数，$\boldsymbol{r}$ 为训练样本的标签值，$\boldsymbol{O}^{L}$ 为实际的输出值。我们的目的是利用随机梯度下降法来更新神经网络的参数，使得通过多步迭代优化后得到一个最优化的网络，让训练样本通过该网络时的损失函数取最小值。即可用用下面的公式来调整权重

$$\boldsymbol{w}^{L}(k+1)=\boldsymbol{w}^{L}(k)-\alpha^{L}\left[\frac{\partial E\left(\boldsymbol{w}^{L}\right)}{\partial \boldsymbol{w}^{L}}\right]_{\boldsymbol{w}^{L}=\boldsymbol{w}^{L}(k)} \qquad (4)$$

其中 $\boldsymbol{k}$ 代表第 k 次迭代优化。（3）式中损失 $\boldsymbol{E}$ 是输出 $\boldsymbol{O}^{L}$ 的函数，而根据（2）式可知 $\boldsymbol{O}^{L}$ 是 $\boldsymbol{I}^{L}$ 的函数，于是有

$$\frac{\partial E}{\partial \boldsymbol{w}^{L}}=\frac{\partial E}{\partial \boldsymbol{O}^{L}}\frac{\partial \boldsymbol{O}^{L}}{\partial \boldsymbol{I}^{L}}\frac{\partial \boldsymbol{I}^{L}}{\partial \boldsymbol{w}^{L}} \qquad (5)$$

利用随机梯度下降法，分别将（3）、（2）、（1）求导后，代入（5）可得

$$\frac{\partial E}{\partial w_{ij}^{L}}=-\left(r_j-O_j^{L}\right)\left(h_j^{L}(I_j^{L})\right)'O_i^{L-1}=-\delta_j^{L}O_i^{L-1} \quad (6)$$

其中 i 代表上一层的第 i 个节点，j 代表该层的第 j 个结点，且这里我们定义了

$$\delta_j^{L}=\left(r_j-O_j^{L}\right)\left(h_j^{L}(I_j^{L})\right)' \quad (7)$$

将（6）式代入（4）式可得

$$w_{ij}^{L}(k+1)=w_{ij}^{L}(k)+\alpha^{L}\delta_j^{L}(k)\,O_i^{L-1}(k) \quad (8)$$

上式给出了网络最后一层（输出层）权重更新的公式。神经网络的初始权重是任意的，α^{L} 作为学习率是一个可以人为调节的参数，而其他值是已知的，所以我们首先可以知道如何调节网络最后一层的权重。然后利用与上面相同的方法可以得到上一层的权重更新公式

$$w_{ij}^{L-1}(k+1)=w_{ij}^{L-1}(k)+\alpha^{L-1}\delta_j^{L-1}(k)\,O_i^{L-2}(k) \quad (9)$$

但我们可以注意到，上式δ_j^{L-1}中的 r_j 是未知的，因为我们不知道网络的中间层应该得到什么样的输出结果，所以我们不能提供中间层的样本标签。因此我们需要根据已知的或可以在网络中观察到的量来找到重新定义 δ^{L-1} 的方法。

注意到

$$\delta_i^{L-1}=-\frac{\partial E}{\partial O_i^{L-1}}\frac{\partial O_i^{L-1}}{\partial I_i^{L-1}}=-\left(h_i^{L-1}\left(I_i^{L-1}\right)\right)'\frac{\partial E}{\partial O_i^{L-1}} \quad (10)$$

式中 I_i^{L-1}可以根据已知的网络计算得到，h_i^{L-1} 是人为确定的激活函数，所以我们只要确定$\partial E/\partial O_i^{L-1}$就能得到完整的结果，于是

$$\begin{aligned}\frac{\partial E}{\partial O_i^{L-1}} &= \sum_{j=1}^{N_L} \frac{\partial E}{\partial I_j^L} \frac{\partial I_j^L}{\partial O_i^{L-1}} \\ &= -\sum_{j=1}^{N_L} \left(-\frac{\partial E}{\partial I_j^L}\right) \frac{\partial}{\partial O_i^{L-1}} \sum_{i=1}^{N_{L-1}} w_{ji}^L O_i^{L-1} \\ &= -\sum_{j=1}^{N_L} \delta_j^L w_{ji}^L \end{aligned} \quad (11)$$

于是我们可以得到期望δ_i^{L-1}的表达式

$$\delta_i^{L-1} = \left(h_i^{L-1}\left(I_i^{L-1}\right)\right)' \sum_{j=1}^{N_L} \delta_j^L w_{ji}^L \quad (12)$$

最后根据（9）式和（12）式我们可以得到所有层在第 k 次迭代优化时的权重更新公式

$$\begin{cases} w_{ij}^l(k+1) = w_{ij}^l(k) + \alpha^l \delta_j^l(k) O_i^{l-1}(k) \\ \delta_i^l = \left(h_i^l\left(I_i^l\right)\right)' \sum_{j=1}^{N_{l+1}} \delta_j^{l+1} w_{ji}^{l+1} \\ \delta_i^L = \left(r_i - O_i^L\right)\left(h_i^L(I_i^L)\right)' \end{cases} \quad (13)$$

其中$l = 1,2,\ldots,L; i = 1,2,\ldots,N_l; j = 1,2,\ldots,N_{l+1}$，所有层的初始权重可以任给。可以发现，每一次迭代优化时，网络需要从输入层开始逐层计算各层的输入输出，再从输出层反向更新权重，所以这种训练方法也被称为反向传播训练。

深度神经网络在实际运用中已经得到较好的结果，但它对训练的要求十分苛刻，由于网络中每一层中的每一个神经节点都可以与下一层的所有节点相连，这就会使得我们需要耗费大量的训练的时间，而且很容易产生过拟合问题。下面本文将介绍一个深度神经网络的分支——卷积神经网络，它可以良好的解决上述问题。

1.2　卷积神经网络

卷积神经网络实质上还是深度神经网络，上述讨论的原理和方法同样适用于卷积神经网络，不同的是，卷积神经网络对每一层都做出了限制，这使得不同层拥有不同但较为明确的功能。接下来我们将从图像识别的角度对卷积神经网络进行阐述。

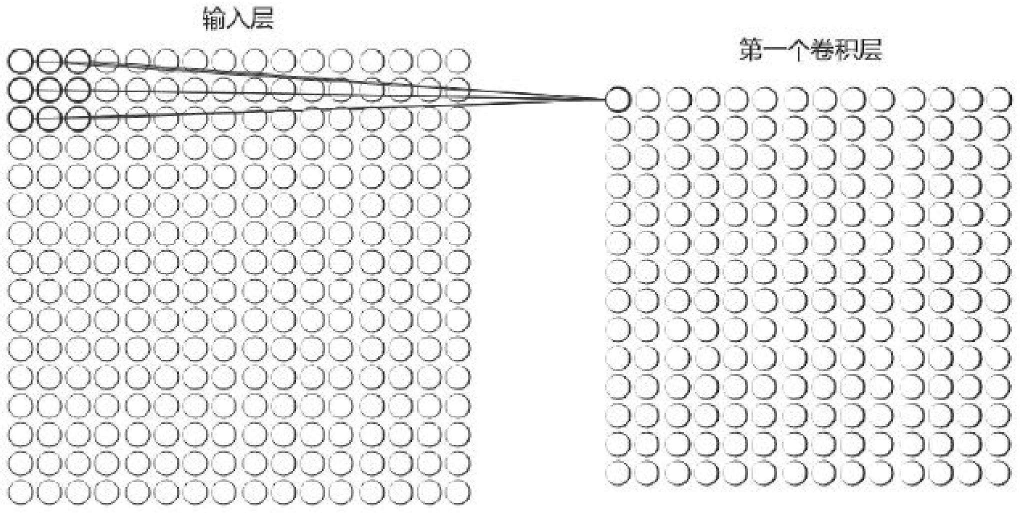
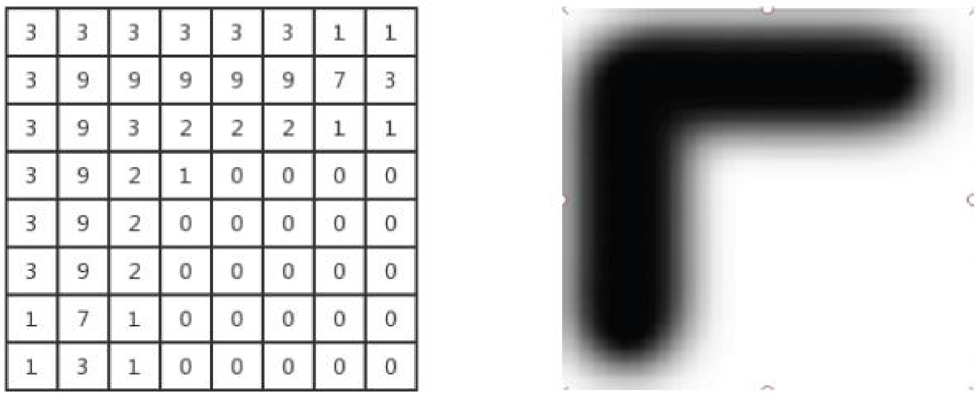


图 2　卷积层

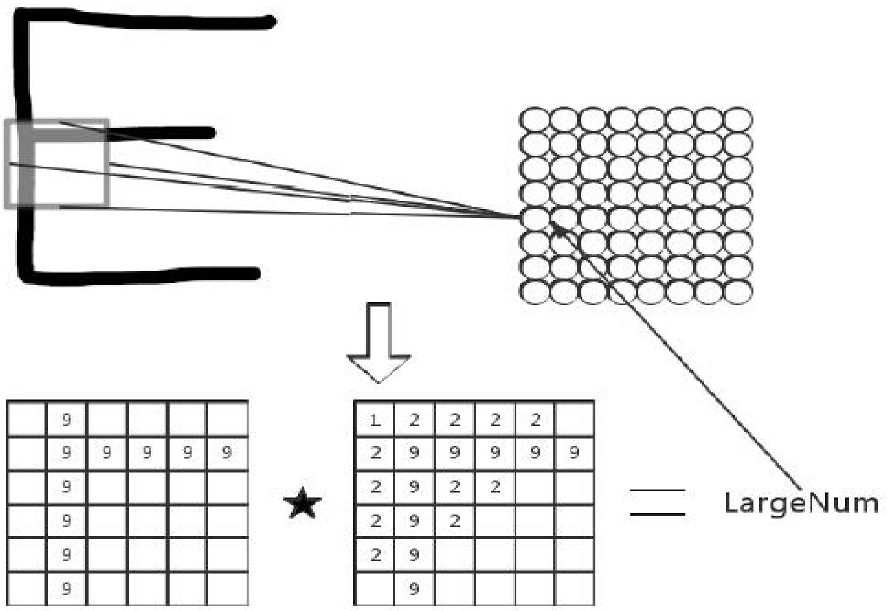
对一幅输入为 16*16 的灰度图像，我们首先使用 256 个神经元作为输入节点，其中每个神经元对应一个像素点的值。如图 2，在第一层中，我们使用 3*3 的卷积核对输入层进行二维卷积得到了 14*14 个输出，其中卷积核就对应了第一层的权重，14*14 个输出对应了第二层神经节点的输入值，我们把这一层称为卷积层。可以看到，卷积层只是对普通的全连接层进行了限制，这个限制使得该层的节点只与上层部分特定的节点相连，且所有节点使用同一个卷积核。



3	3	3	3	3	3	1	1
3	9	9	9	9	9	7	3
3	9	3	2	2	2	1	1
3	9	2	1	0	0	0	0
3	9	2	0	0	0	0	0
3	9	2	0	0	0	0	0
1	7	1	0	0	0	0	0
1	3	1	0	0	0	0	0

图 3　一个特定核的卷积

现在来说明为什么进行这种限制后可以得到更好的图像识别结果。如图 3，假设我们规定图左的卷积核，图右是它的可视化效果。当我们使用这个卷积核对图 4 进行卷积运算时发现，当卷积遇到与其内容相似的区域时可以得到更大的输出，这就使得卷积核能够筛选出具有特定样式的区域，这些区域就可以构成一种特征。在该幅图中，这种特征就可以被描述为“有一个左上方朝内的直角”。



	9				
	9	9	9	9	9
	9				
	9				
	9				
	9				

1	2	2	2	2	
2	9	9	9	9	9
2	9	2	2		
2	9	2			
2	9				
	9				

图 4　对一幅图计算卷积

这也是深度神经网络把原始信息转化为低级特征的第一步，通过构建一个包含我们想寻找的特征的卷积核，我们就能在原始图片中找到含有这些特征的区域并保存它们的位置和相似度（这体现在卷积层中神经节点的位置和值）以供下一层使用。而当需要识别多个特征时，只要增加相应个数的卷积核并增加对应的神经节点即可。

2　模型构建

2.1　LeNet-5

LeNet-5 是一个典型的卷积神经网络，也是本文所采用的神经网络模型。LeNet-5 是由纽约大学计算机系教授 Yann Lecun(1989)所提出的一个高效识别数字的卷积神经网络模型。

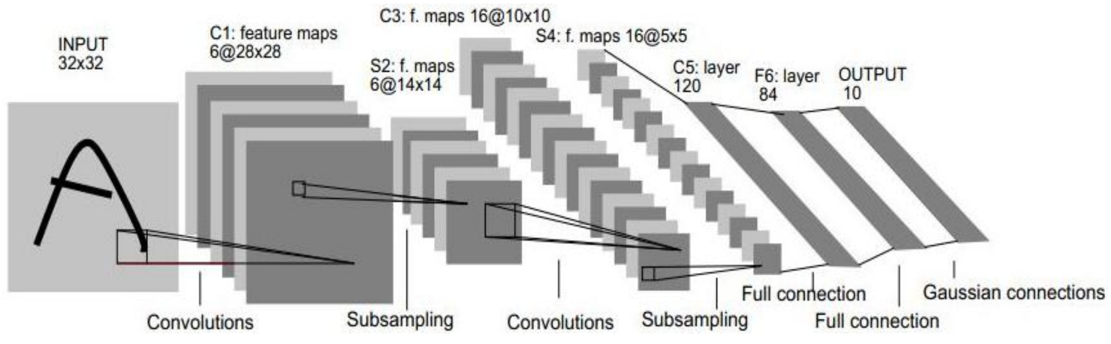


图 5　LeNet-5 网络结构

其中的 Subsampling 层也称为 Pool 层（下采样层或池化层），Pool 层可以起到增强特征、防止过拟合的作用，它的计算过程和卷积计算十分相似，这里不做过多介绍。

2.2　我们的模型

我们在本次综合实验中所提出的模型和 LeNet-5 十分相似，结构示意图如下：

Input->Conv->ReLU->Pool->ReLU->Conv->Pool->FullyConnected->Softmax->Output

图 6　我们的网络结构

其中 Input 层的大小，即每张输入图片的大小，为 28*28；第一个 Conv 层我们用了 12 个卷积核，每个卷积核的大小为 24*24；第一个 Pool 层的有 6 个下采样核，每个的大小为 14*14；同理第二个 Conv 层的大小为 24*10*10，第二个 Pool 层的大小为 12*5*5；最后的全连接层大小为 26。

ReLU 层是一个激活函数——ReLU 函数（$f(x)=max(0,x)$），最后的 Softmax 层是一个分类函数——Softmx 函数，定义如下

$$P(i)=\frac{exp(\theta_i^T x)}{\sum_{k=1}^{K} exp(\theta_k^T x)}$$

这是一个分类函数，其中 K 是总类别数，在这里即 26。θ 即为全连接层的权重，x 则为全连接层的输入。这个函数输出一个 26 维向量，第 i 维的值表示的是这张图片属于第 i 个类别的概率。最终我们将图片分为第 i 类，当 $P_{(i)}$ 最大时。

3　数学实验

3.1　数据来源与预处理

我们的数据来源是 EMNIST，其中包含 26 个英文字母（包含大小写）。我们将每张图片中的字母部分提取出来，调整成 20*18 的大小，再放到一个 28*28 的白板中央。EMNIST 中共有 145600 张图片，我们随机抽取其中的 124800 张图片为训练集，剩余的 20800 张图片为测试集。

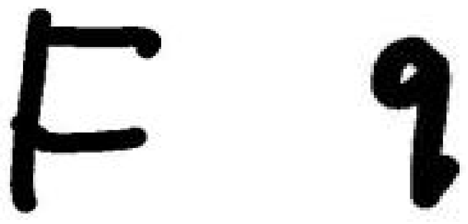


图 6　EMNIST 中的两个样本

3.2　训练网络

我们利用 2.1.2 中提出的卷积神经网络模型来对数据进行训练，设置初始的学习率为 1，迭代次数为 40。训练过程：迭代一次的时间为 500s。损失函数的变化过程如图 7 所示。

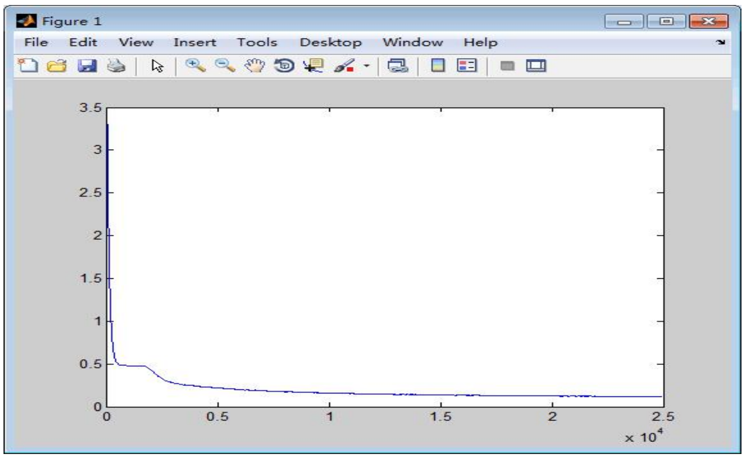


图 7　损失函数

用训练后的卷积神经网络模型来预测测试集的 labels，最终的正确率为 89.7%。不同字母预测的错误率如图 8 所示。

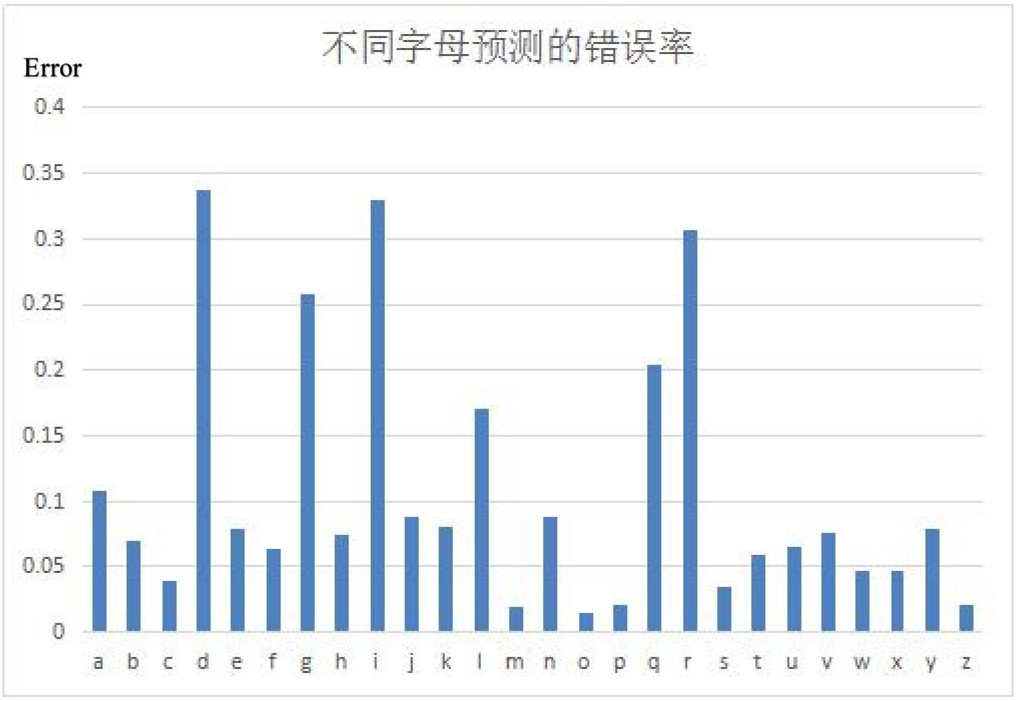


图 8　不同字母预测结果

4　问题分析

4.1　损失下降结果分析

由于我们采用随机梯度下降法来训练，而随机梯度下降法在寻找最优值的过程中，非常容易陷入局部极小点，并且很难跳出局部极小点，因此在迭代进行到一定次数后，损失函数就趋于平稳不再下降了。

4.2　不同字母预测结果分析

如图 8 所示，可以看到字母：d,g,i,l,q,r 的预测错误率较高。经过对比预测图片我们发现，在手写字体中，由于不同人书写习惯的不同，手写体中的 d 和 a 的特征就很相似，g 和 q 书写形式上非常容易弄混，i 与 l 的手写体也很难分辨，而 r 经常会被写成 v 甚至是 n。因此我们发现这些字母预测错误率高的很大一部分原因是不同人的不同书写习惯造成的。

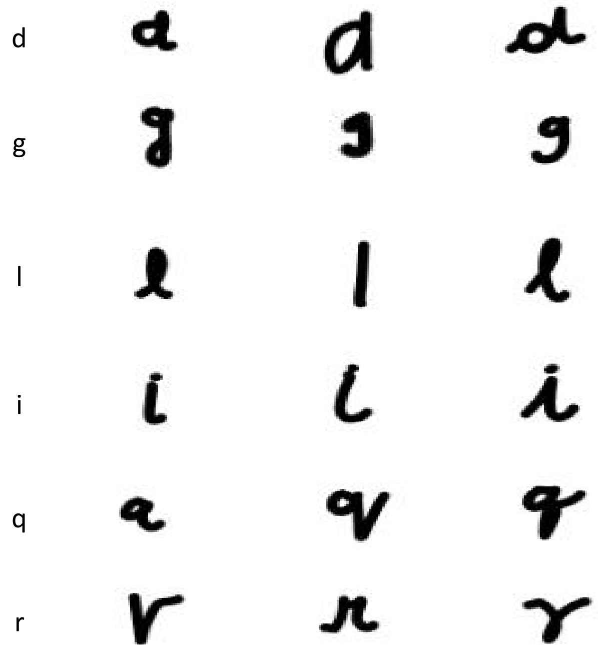


图 9　错误率较高的一些字母

5 代码

代码基于 matlab 的 deeplearningtoolbox 的深度学习框架来完成。

```
%% cnnapplygrads.m
function net = cnnapplygrads(net, opts)
    for l = 2 : numel(net.layers)
        if strcmp(net.layers{l}.type, 'c')
            for j = 1 : numel(net.layers{l}.a)
                for ii = 1 : numel(net.layers{l - 1}.a)
                    net.layers{l}.k{ii}{j} = net.layers{l}.k{ii}{j}
- opts.alpha * net.layers{l}.dk{ii}{j};
                end
                net.layers{l}.b{j} = net.layers{l}.b{j} -
opts.alpha * net.layers{l}.db{j};
            end
        end
    end

    net.ffW = net.ffW - opts.alpha * net.dffW;
    net.ffb = net.ffb - opts.alpha * net.dffb;
end

%% cnnbp.m
function net = cnnbp(net, y)
    n = numel(net.layers);

    %  error
    net.e = net.o - y;
    %  loss function
    net.L = 1/2* sum(net.e(:) .^ 2) / size(net.e, 2);

    %%  backprop deltas
    net.od = net.e .* (net.o .* (1 - net.o));   %  output delta
    net.fvd = (net.ffW' * net.od);              %  feature vector
delta
    if strcmp(net.layers{n}.type, 'c')         %  only conv layers
has sigm function
        net.fvd = net.fvd .* (net.fv .* (1 - net.fv));
    end

    %  reshape feature vector deltas into output map style
    sa = size(net.layers{n}.a{1});
    fvnum = sa(1) * sa(2);
    for j = 1 : numel(net.layers{n}.a)
        net.layers{n}.d{j} = reshape(net.fvd(((j - 1) * fvnum + 1) :
j * fvnum, :), sa(1), sa(2), sa(3));
    end

    for l = (n - 1) : -1 : 1
        if strcmp(net.layers{l}.type, 'c')
            for j = 1 : numel(net.layers{l}.a)
                net.layers{l}.d{j} = net.layers{l}.a{j} .* (1 -
net.layers{l}.a{j}) .* (expand(net.layers{l + 1}.d{j},
[net.layers{l + 1}.scale net.layers{l + 1}.scale 1]) / net.layers{l
+ 1}.scale ^ 2);
            end
        elseif strcmp(net.layers{l}.type, 's')
            for i = 1 : numel(net.layers{l}.a)
                z = zeros(size(net.layers{l}.a{1}));
                for j = 1 : numel(net.layers{l + 1}.a)
                     z = z + convn(net.layers{l + 1}.d{j},
rot180(net.layers{l + 1}.k{i}{j}), 'full');
                end
                net.layers{l}.d{i} = z;
            end
        end
    end

    %%  calc gradients
    for l = 2 : n
        if strcmp(net.layers{l}.type, 'c')
            for j = 1 : numel(net.layers{l}.a)
                for i = 1 : numel(net.layers{l - 1}.a)
                    net.layers{l}.dk{i}{j} =
convn(flipall(net.layers{l - 1}.a{i}), net.layers{l}.d{j},
'valid') / size(net.layers{l}.d{j}, 3);
                end
                net.layers{l}.db{j} = sum(net.layers{l}.d{j}(:)) /
size(net.layers{l}.d{j}, 3);
            end
        end
    end
    net.dffW = net.od * (net.fv)' / size(net.od, 2);
    net.dffb = mean(net.od, 2);

    function X = rot180(X)
        X = flipdim(flipdim(X, 1), 2);
    end
end

%% cnnff.m
function net = cnnff(net, x)
    n = numel(net.layers);
    net.layers{1}.a{1} = x;
    inputmaps = 1;

    for l = 2 : n   %  for each layer
        if strcmp(net.layers{l}.type, 'c')
            %  !!below can probably be handled by insane matrix
operations
            for j = 1 : net.layers{l}.outputmaps   %  for each output
map
                %  create temp output map
                z = zeros(size(net.layers{l - 1}.a{1}) -
[net.layers{l}.kernelsize - 1 net.layers{l}.kernelsize - 1 0]);
                for i = 1 : inputmaps   %  for each input map
                    %  convolve with corresponding kernel and add to
temp output map
                    z = z + convn(net.layers{l - 1}.a{i},
net.layers{l}.k{i}{j}, 'valid');
                end
                %  add bias, pass through nonlinearity
                net.layers{l}.a{j} = sigm(z + net.layers{l}.b{j});
            end
            %  set number of input maps to this layers number of
outputmaps
            inputmaps = net.layers{l}.outputmaps;
        elseif strcmp(net.layers{l}.type, 's')
            %  downsample
            for j = 1 : inputmaps
                z = convn(net.layers{l - 1}.a{j},
ones(net.layers{l}.scale) / (net.layers{l}.scale ^ 2),
'valid');   %  !! replace with variable
                net.layers{l}.a{j} = z(1 : net.layers{l}.scale : end,
1 : net.layers{l}.scale : end, :);
            end
        end
    end

    %  concatenate all end layer feature maps into vector
    net.fv = [];
    for j = 1 : numel(net.layers{n}.a)
        sa = size(net.layers{n}.a{j});
        net.fv = [net.fv; reshape(net.layers{n}.a{j}, sa(1) * sa(2),
sa(3))];
    end
    %  feedforward into output perceptrons
    net.o = sigm(net.ffW * net.fv + repmat(net.ffb, 1, size(net.fv,
2)));

end
%% cnnnumgradcheck.m
function cnnnumgradcheck(net, x, y)
    epsilon = 1e-4;
    er      = 1e-8;
    n = numel(net.layers);
    for j = 1 : numel(net.ffb)
        net_m = net; net_p = net;
        net_p.ffb(j) = net_m.ffb(j) + epsilon;
        net_m.ffb(j) = net_m.ffb(j) - epsilon;
        net_m = cnnff(net_m, x); net_m = cnnbp(net_m, y);
        net_p = cnnff(net_p, x); net_p = cnnbp(net_p, y);
        d = (net_p.L - net_m.L) / (2 * epsilon);
        e = abs(d - net.dffb(j));
        if e > er
            e
            d / net.dffb(j)
            error('numerical gradient checking failed');
        end
    end

    for i = 1 : size(net.ffW, 1)
        for u = 1 : size(net.ffW, 2)
            net_m = net; net_p = net;
            net_p.ffW(i, u) = net_m.ffW(i, u) + epsilon;
            net_m.ffW(i, u) = net_m.ffW(i, u) - epsilon;
            net_m = cnnff(net_m, x); net_m = cnnbp(net_m, y);
            net_p = cnnff(net_p, x); net_p = cnnbp(net_p, y);
            d = (net_p.L - net_m.L) / (2 * epsilon);
            e = abs(d - net.dffW(i, u));
            if e > er
                e
                d / net.ffW(i, u)
                error('numerical gradient checking failed');
            end
        end
    end

    for l = n : -1 : 2
        if strcmp(net.layers{l}.type, 'c')
            for j = 1 : numel(net.layers{l}.a)
                net_m = net; net_p = net;
                net_p.layers{l}.b{j} = net_m.layers{l}.b{j} +
epsilon;
                net_m.layers{l}.b{j} = net_m.layers{l}.b{j} -
epsilon;
                net_m = cnnff(net_m, x); net_m = cnnbp(net_m, y);
                net_p = cnnff(net_p, x); net_p = cnnbp(net_p, y);
                d = (net_p.L - net_m.L) / (2 * epsilon);
                e = abs(d - net.layers{l}.db{j});
                if e > er
                    e
                    d / net.layers{l}.db{j}
                    error('numerical gradient checking failed');
                end
                for i = 1 : numel(net.layers{l - 1}.a)
                    for u = 1 : size(net.layers{l}.k{i}{j}, 1)
                        for v = 1 : size(net.layers{l}.k{i}{j}, 2)
                            net_m = net; net_p = net;
                            net_p.layers{l}.k{i}{j}(u, v) =
net_p.layers{l}.k{i}{j}(u, v) + epsilon;
                            net_m.layers{l}.k{i}{j}(u, v) =
net_m.layers{l}.k{i}{j}(u, v) - epsilon;
                            net_m = cnnff(net_m, x); net_m =
cnnbp(net_m, y);
                            net_p = cnnff(net_p, x); net_p =
cnnbp(net_p, y);
                            d = (net_p.L - net_m.L) / (2 * epsilon);
                            e = abs(d - net.layers{l}.dk{i}{j}(u, v));
                            if e > er
                                error('numerical gradient checking
failed');
                            end
                        end
                    end
                end
            end
        elseif strcmp(net.layers{l}.type, 's')
%           for j = 1 : numel(net.layers{l}.a)
%               net_m = net; net_p = net;
%               net_p.layers{l}.b{j} = net_m.layers{l}.b{j} +
epsilon;
%               net_m.layers{l}.b{j} = net_m.layers{l}.b{j} -
epsilon;
%               net_m = cnnff(net_m, x); net_m = cnnbp(net_m, y);
%               net_p = cnnff(net_p, x); net_p = cnnbp(net_p, y);
%               d = (net_p.L - net_m.L) / (2 * epsilon);
%               e = abs(d - net.layers{l}.db{j});
%               if e > er
%                   error('numerical gradient checking failed');
%               end
%           end
        end
    end
%    keyboard
end

%% cnnsetup.m
function net = cnnsetup(net, x, y)
    inputmaps = 1;
    mapsize = size(squeeze(x(:, :, 1)));

    for l = 1 : numel(net.layers)   %  layer
        if strcmp(net.layers{l}.type, 's')
            mapsize = mapsize / net.layers{l}.scale;
            assert(all(floor(mapsize)==mapsize), ['Layer '
num2str(l) ' size must be integer. Actual: ' num2str(mapsize)]);
            for j = 1 : inputmaps
                net.layers{l}.b{j} = 0;
            end
        end
        if strcmp(net.layers{l}.type, 'c')
            mapsize = mapsize - net.layers{l}.kernelsize + 1;
            fan_out = net.layers{l}.outputmaps *
net.layers{l}.kernelsize ^ 2;
            for j = 1 : net.layers{l}.outputmaps  %  output map
                fan_in = inputmaps * net.layers{l}.kernelsize ^ 2;
                for i = 1 : inputmaps  %  input map
```

```
                    net.layers{l}.k{i}{j} =
(rand(net.layers{l}.kernelsize) - 0.5) * 2 * sqrt(6 / (fan_in +
fan_out));
                end
                net.layers{l}.b{j} = 0;
            end
            inputmaps = net.layers{l}.outputmaps;
        end
    end
    % 'onum' is the number of labels, that's why it is calculated
using size(y, 1). If you have 20 labels so the output of the network
will be 20 neurons.
    % 'fvnum' is the number of output neurons at the last layer,
the layer just before the output layer.
    % 'ffb' is the biases of the output neurons.
    % 'ffW' is the weights between the last layer and the output
neurons. Note that the last layer is fully connected to the output
layer, that's why the size of the weights is (onum * fvnum)
    fvnum = prod(mapsize) * inputmaps;
    onum = size(y, 1);

    net.ffb = zeros(onum, 1);
    net.ffW = (rand(onum, fvnum) - 0.5) * 2 * sqrt(6 / (onum + fvnum));
end
%{
net=cnn;
x=train_x2;
y=train_y2;
%}

%% cnntest.m
function [er, bad] = cnntest(net, x, y)
    %  feedforward
    net = cnnff(net, x);
    [~, h] = max(net.o);
    [~, a] = max(y);
    bad = find(h ~= a);
    er = numel(bad) / size(y, 2);
end
%{
net=cnn;
x=test_x2;
y=test_y2;
net = cnnff(net, x);
[~, h] = max(net.o);
h
%}

%% cnntrain.m
function net = cnntrain(net, x, y, opts)
    m = size(x, 3);
    numbatches = m / opts.batchsize;
    if rem(numbatches, 1) ~= 0
        error('numbatches not integer');
    end
    net.rL = [];
    for i = 1 : opts.numepochs
        disp(['epoch ' num2str(i) '/' num2str(opts.numepochs)]);
        tic;
        kk = randperm(m);
        for l = 1 : numbatches
            batch_x = x(:, :, kk((l - 1) * opts.batchsize + 1 : l
* opts.batchsize));
            batch_y = y(:,    kk((l - 1) * opts.batchsize + 1 : l *
opts.batchsize));

            net = cnnff(net, batch_x);
            net = cnnbp(net, batch_y);
            net = cnnapplygrads(net, opts);
            if isempty(net.rL)
                net.rL(1) = net.L;
            end
            net.rL(end + 1) = 0.99 * net.rL(end) + 0.01 * net.L;
        end
        toc;
    end
end
%{
x=train_x2;
y=train_y2;
net=cnn;
%}

%% siz.m
%9%的错误率
clear all; close all; clc;
addpath('D:MATLAB/toolbox/DeepLearnToolbox-master/data');
addpath('D:MATLAB/toolbox/DeepLearnToolbox-master/util');
```

```
load('E:\character\matlab\emnist-letters.mat')
train_x=dataset.train.images;
train_yt=dataset.train.labels;
test_x=dataset.test.images;
test_yt=dataset.test.labels;
train_y=zeros(26,124800);
test_y=zeros(26,20800);
for i=1:124800
    j=train_yt(i);
    train_y(j,i)=1;
end
for i=1:20800
    j=test_yt(i);
    test_y(j,i)=1;
end
train_x = double(reshape(train_x',28,28,124800))/255;
test_x = double(reshape(test_x',28,28,20800))/255;
train_y = double(train_y);
test_y = double(test_y);
sumx=0;sumy=0;
for i=1:124800
    max=[0,0];min=[28,28];
    for j=1:28
        for k=1:28
            if train_x(j,k,i)>0.5
                if j>max(1)
                    max(1)=j;
                else if j<=min(1)
                        min(1)=j;
                    end
                end
                if k>max(2)
                    max(2)=k;
                else if k<=min(2)
                        min(2)=k;
                    end
                end
            end
        end
    end
    sumx=sumx+max(1)-min(1);
    sumy=sumy+max(2)-min(2);
end
avx=sumx/124800;
```

```
avy=sumy/124800;
%avx=19.2936 平均行数
%avy=17,2399 平均列数

%% predictright.m
function [Yy,right]=predictright(x)
%x 是图片的名字，需要加单引号。使用该函数时请将 cnn.mat 和所需预测的图像放在
相同路径中
load('cnnright2.mat');
net=cnn;
a=imread(x);
if length(size(a))==3
    b=rgb2gray(a);
else
    b=a;
end
b=medfilt2(b,[5,5]);%滤波
siz=size(b);
ax=[0,0];in=siz;
for i=6:siz(1)-5
    for j=6:siz(2)-5
        if b(i,j)<1
            if i>ax(1)
                ax(1)=i;
            end
            if i<in(1)
                in(1)=i;
            end
            if j>ax(2)
                ax(2)=j;
            end
            if j<in(2)
                in(2)=j;
            end
        end
    end
end
c=imresize(b(in(1):ax(1),in(2):ax(2)),[20,18]);
d=255+zeros(28,28,2);
d(5:24,6:23,1)=c;
d=1-d/255;
d(:,:,2)=zeros(28,28);
net = cnnff(net, d);
[~,h] = max(net.o);
```

```
tmp=sort(net.o(:,1),'descend');
right=tmp(1)/sum(tmp(1:3));
if h(1)<=26
    Yy=char(h(1)+64);
else
    Yy=char(h(1)+70);
end
end

%% stat.m
clear all; close all; clc;
load('emnist-letters.mat');
test_x=dataset.test.images;
test_yt=dataset.test.labels;
test_y=zeros(26,20800);
for i=1:20800
    j=test_yt(i);
    test_y(j,i)=1;
end
test_x = double(reshape(test_x',28,28,20800))/255;
test_y = double(test_y);
load('cnnright2.mat');
net = cnnff(cnn, test_x);
[~, h] = max(net.o);
[~, a] = max(test_y);
err=zeros(1,26);
for k=1:20800
    if h(k)~=a(k)
        err(a(k))=err(a(k))+1;
    end
end
err=err/800;

%% tat.m
clear all; close all; clc;
addpath('D:MATLAB/toolbox/DeepLearnToolbox-master/data');
addpath('D:MATLAB/toolbox/DeepLearnToolbox-master/util');
load('E:\character\matlab\emnist-letters.mat')
train_x=dataset.train.images;
train_yt=dataset.train.labels;
test_x=dataset.test.images;
test_yt=dataset.test.labels;
train_y=zeros(26,124800);
test_y=zeros(26,20800);
```

```
for i=1:124800
    j=train_yt(i);
    train_y(j,i)=1;
end
for i=1:20800
    j=test_yt(i);
    test_y(j,i)=1;
end
train_x = double(reshape(train_x',28,28,124800))/255;
test_x = double(reshape(test_x',28,28,20800))/255;
train_y = double(train_y);
test_y = double(test_y);

%% ex1


cnn.layers = {
    struct('type', 'i') %input layer
    struct('type', 'c', 'outputmaps', 12, 'kernelsize',
5) %convolution layer
    struct('type', 's', 'scale', 2) %sub sampling layer
    struct('type', 'c', 'outputmaps', 24, 'kernelsize',
5) %convolution layer
    struct('type', 's', 'scale', 2) %subsampling layer
};

% 这里把 cnn 的设置给 cnnsetup，它会据此构建一个完整的 CNN 网络，并返回
cnn = cnnsetup(cnn, train_x, train_y);

% 学习率
opts.alpha = 1;
% 每次挑出一个 batchsize 的 batch 来训练，也就是每用 batchsize 个样本就调
整一次权值，而不是
% 把所有样本都输入了，计算所有样本的误差了才调整一次权值
opts.batchsize = 100;
% 训练次数，用同样的样本集。训练的时候：
% 1 的时候 11.41% error
% 5 的时候 4.2% error
% 10 的时候 2.73% error
opts.numepochs = 40;

% 然后开始把训练样本给它，开始训练这个 CNN 网络
cnn= cnntrain(cnn, train_x, train_y, opts);
% 然后就用测试样本来测试
```

```
[er, bad] = cnntest(cnn, test_x, test_y);

%plot mean squared error
plot(cnn.rL);
%show test error
disp([num2str(er*100) '% error']);

%% Gui.m
function varargout = Gui(varargin)
% GUI MATLAB code for Gui.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in GUI.M with the given input
arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Gui

% Last Modified by GUIDE v2.5 14-Jun-2017 17:18:40

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Gui_OpeningFcn, ...
                   'gui_OutputFcn',  @Gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Gui is made visible.
function Gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Gui (see VARARGIN)

% Choose default command line output for Gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
addpath(char(strcat(strrep(pwd,'\','/'),'/DeepLearnToolbox-mas
ter/util')));
addpath(char(strcat(strrep(pwd,'\','/'),'/DeepLearnToolbox-mas
ter/CNN')));
% UIWAIT makes Gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
```

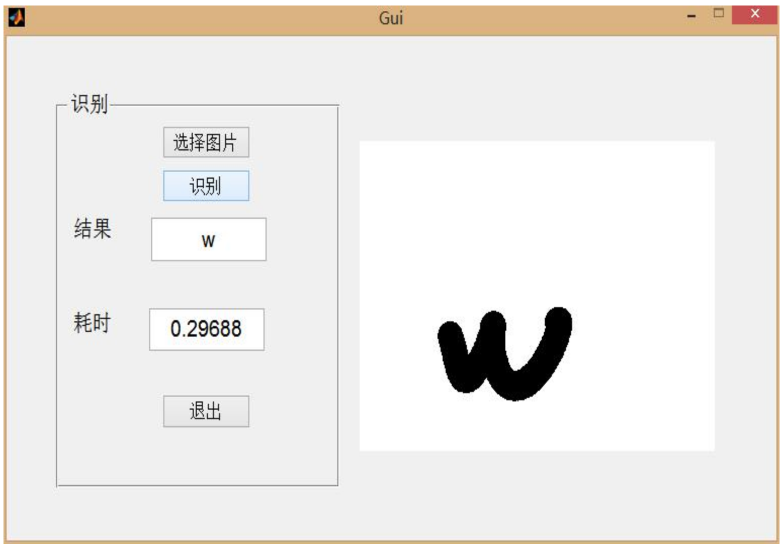
```
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im %设 im 为全局变量
%选择图片路径
[filename,pathname,~] =
uigetfile({'*.jpg;*.tif;*.png;*.gif','All Image Files';...
        '*.*','All Files' },'选择图片');
%合成路径+文件名
str=[pathname filename];
 %读取图片
im=imread(str);
%使用第一个 axes
 %显示图片
imshow(im);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
button=questdlg('是否确认关闭','关闭确认','是','否','是');
if strcmp(button,'是')
    close(gcf)
    delete(hObject);
else
    return;
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im
[result time] =  predict(im);
set(handles.edit5,'string',sprintf(result));
set(handles.edit6,'string',sprintf(num2str(time)));
```

```
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of
edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of
edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of
```

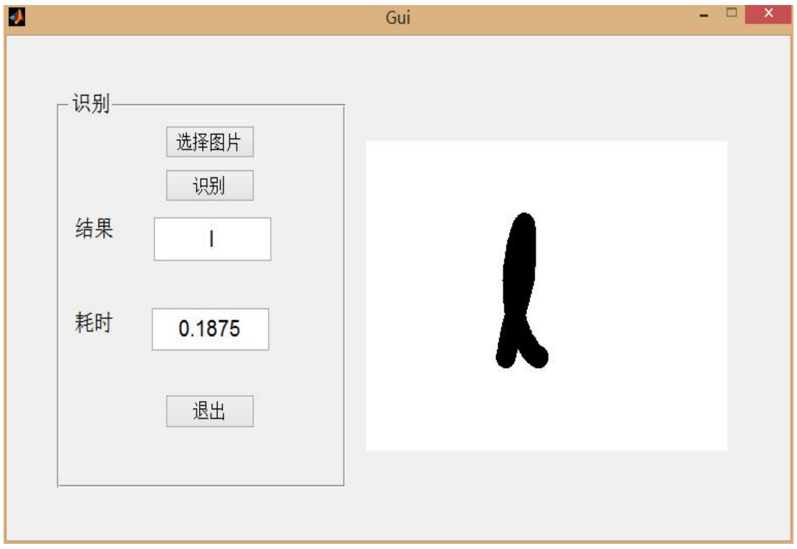
```
edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of
edit4 as a double

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

6 GUI 运行结果



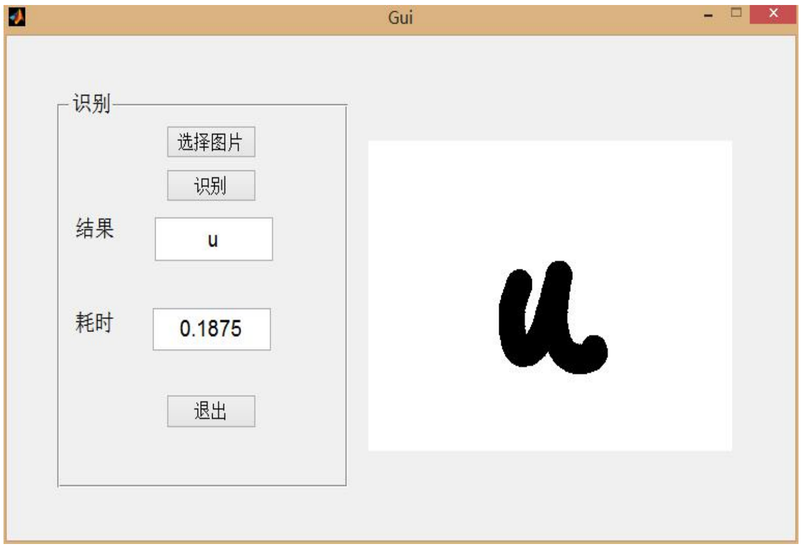
结果 w

耗时 0.29688



结果 l

耗时 0.1875



结果 u

耗时 0.1875

7　心得体会

经过本次实验，我们小组成员科学分工，成功完成，并有如下心得体会：

（1）在字母识别，或者处理其他类似的图片识别、文字识别的问题过程中，要想完成的结果十分理想，需要涉及到机器学习方向，而结果的好坏一方面与选择方法有关，另一方面与训练集的选择有很大关系。我们小组在处理过程中，更换了几次处理算法，并在训练集的选取上耗费了很多时间。在处理这种带有学习、训练性质的问题时，往往需要我们反复选取训练集来达到好的效果；

（2）对于字母识别这一问题，我们也有了一些经验，例如哪些属于高错误率类，哪些字母识别更为方便。或许我们可以再后期在这一方面拓展，选用不同算法处理不同类字母。甚至，我们将字母拓展到汉字领域，实际上现在的人工智能已经可以识别英文达到96%的成功率，而汉字识别在国内也做到了很高的成功率，我们可以在这些方面进一步学习；

（3）我们最后选择了GUI进行展示，这是一种初步的人机交互，我们可以再进行深入，选择读取数据后输出成以整个系统的形式，这样就实现了字母图片识别的应用，并且可以往更大的数据量方向攀登；

（4）运用Matlab做完这一实验后，我们小组对于Matlab的使用

也更加熟悉，更加体会到了它的方便，并且在 GUI 设计这一方面也有了自己的经验。