

课程设计作业

（2024-2025学年 第一学期）

题 目：基于LeNet-5网络的手写数字识别

**专业： 网络工程**

**学生姓名：**

**班级学号：**

**指导教师：**

**指导单位： 物联网学院**

**日期： 2024年 12月 1日**

**基于LeNet-5网络的手写数字识别**

1. **课题内容**

使用MATLAB基于LeNet5网络进行手写数字识别，先构造出神经网络架构，再进行训练得到模型。使用时构造图形用户界面，用户上传写有数字的图片，程序自动识别并输出识别结果。

**二、设计原理介绍**

手写数字识别是计算机视觉领域中的经典任务，广泛应用于邮政编码识别、银行支票识别等场景。该任务的目标是开发一种能够识别手写数字的系统，其中输入为用户手写的数字图像，输出为数字的分类结果。

LeNet-5是Yann LeCun等人于1998年提出的一种卷积神经网络（Convolutional Neural Networks, CNN）架构，是最早的深度学习网络之一，尤其适用于图像识别任务。该网络架构包括以下几个关键组成部分：

1. 卷积层：通过卷积操作提取图像的局部特征。其核心计算公式为：



其中，是第个特征图在位置的输出，是输入图像在卷积核覆盖区域内的像素值，是第个卷积核的权重，是偏置项，是激活函数。

1. 池化层：通过最大池化或平均池化操作降低特征的维度，减少计算量，同时保留重要特征。最大池化的公式为：



其中，是池化窗口的左上角位置，是池化窗口的大小。

1. 全连接层：将高维的特征图转化为一维向量并进行分类，其输出为：



其中，为权重，为输入特征，为偏置，为激活函数。

1. 激活函数：LeNet-5使用了Sigmoid激活函数，形式为如下：



该函数用于引入非线性，增强模型的表达能力。

1. Softmax输出层：在最后的输出层，使用Softmax函数对分类结果进行概率化处理，其公式为：



其中，是类别对应的激活值，是输入属于类别的概率。

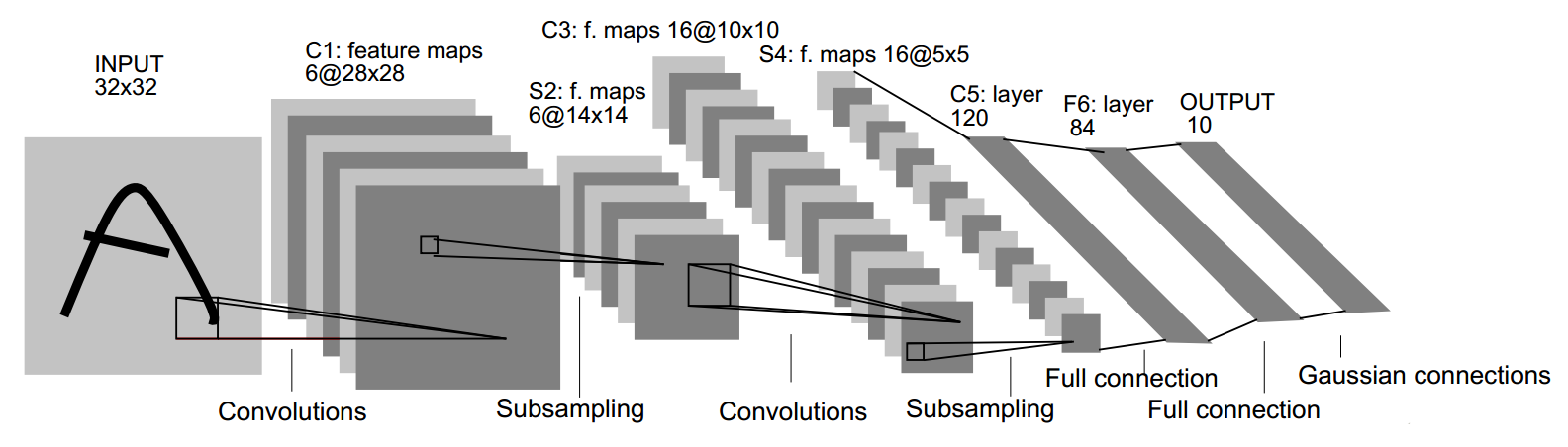


图1 LeNet-5的网络模型示意图

根据 LeNet-5 网络的结构，我将手写数字识别任务的输入图像（28x28像素的手写数字图像）通过卷积层和池化层进行特征提取，再通过全连接层进行分类。训练过程中，我们使用手写数字数据集（MNIST）来训练网络模型。

我使用了MATLAB中的Deep Network Designer APP来设计神经网络，然后使用MATLAB自带的trainNetwork函数进行训练，训练结束后使用MNIST提供的验证集进行验证，得到满意的模型结果后保存模型，用于后续的推理。

为了让用户能够方便地使用手写数字识别系统，我们设计了一个图形用户界面（GUI）。在这个界面上，用户可以上传图片进行识别，并且支持个位数，多位数以及彩色数字识别。用户上传的图片首先会经过预处理进行拆分裁剪，图像加强等步骤，然后会被转换为与训练数据集相同的尺寸（28x28像素的灰度图像），输入到LeNet-5模型进行前向传播，每个拆分的数字最终输出一个0-9的数字类别。将所有结果合并后，在GUI界面中实时显示。

**三、概要设计**

本系统基于LeNet-5卷积神经网络（CNN）进行手写数字识别，涉及数据预处理、神经网络设计、训练过程和推理过程。

**1. 数据集处理**

数据预处理的主要目标是将MNIST数据集中的图像和标签转换为适合训练的格式。MNIST数据集包含大量的手写数字图像以及相应的标签，图像是28x28像素的灰度图。



图2 MNIST数据集示例（部分）

图像处理方式如下：

1. 从原始数据文件中读取图像数据。
2. 将图像数据按行列进行重排列，确保数据格式适应神经网络的输入要求。

③ 图像的像素值进行归一化处理，缩放到[0, 1]区间，这样能够提高训练的稳定性和收敛速度。

标签处理方式如下：

1. 从标签数据文件中读取每张图像的对应标签。

② 将标签转换为分类标签(0~9)，并转换为categorical数据类型，以便进行分类任务。

**2. 神经网络设计**

本系统采用LeNet-5作为卷积神经网络架构，LeNet-5是经典的深度学习模型，适合用于手写数字的分类任务。其网络结构如下：

图示

描述已自动生成

图3 LeNet-5的网络结构图

输入层：接收28x28像素的单通道灰度图像，大小为[28, 28, 1]。

卷积层：第一个卷积层使用6个5x5的卷积核，提取输入图像的低级特征。第二个卷积层使用16个5x5的卷积核，进一步提取高级特征。

池化层：每个卷积层后面跟随一个2x2的最大池化层，用于下采样图像，减少数据量，降低计算复杂度，并防止过拟合。

全连接层：使用三个全连接层，分别有120，84和10个神经元，提取最终的特征。

输出层：使用Softma层，将网络的输出转换为每个数字类别的概率，分类输出结果。

在MATLAB中，可以通过deepNetworkDesigner APP来快速搭建神经网络，只需拖拽神经网络组件并进行连接即可，连接完成后程序会自动生成代码。

图示

描述已自动生成

图4 LeNet-5神经网络的MATLAB设计图

表1 网络模型的具体参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **网络层** | **卷积核尺寸** | **步长** | **填充** | **输出大小** |
| 输入层 |  |  |  | 32 \* 32 \* 1 |
| 卷积层1 | 5 | 1 | 0 | 28 \* 28 \* 6 |
| 最大池化层1 | 2 | 2 | 0 | 14 \* 14 \* 6 |
| 卷积层2 | 5 | 1 | 0 | 10 \* 10 \* 6 |
| 最大池化层2 | 2 | 2 | 0 | 5 \* 5 \* 16 |
| 全连接层1 |  |  |  | 1 \* 1 \* 120 |
| 全连接层2 |  |  |  | 1 \* 1 \* 84 |
| 全连接层3 |  |  |  | 1 \* 1 \* 10 |
| Softmax层 |  |  |  | 1 \* 1 \* 10 |
| 分类层 |  |  |  | 1 \* 1 \* 10 |

**3. 训练过程**

训练模型的目标是优化神经网络中的参数，使得网络能够准确地识别手写数字。

学习率调度：采用分段学习率调度策略，每经过指定的训练周期（5次epochs），学习率降低为原来的0.2倍。这有助于在训练后期更精细地调整权重，防止过拟合。

训练参数：

①最大训练轮次（MaxEpochs）：设置为20轮，意味着模型最多进行20轮数据迭代，若模型在此期间收敛则提前停止。

②小批量大小（MiniBatchSize）：设定为128，每次训练时从训练集中随机选择128张图像进行训练，以提高训练效率。

③损失函数：系统采用交叉熵损失函数，它在多分类问题中表现优秀，能够有效地衡量预测结果与真实标签之间的差异。

**4. 推理方法**

在模型训练完成后，系统将进行手写数字的推理，具体步骤如下：

用户通过图形用户界面（GUI）上传图片，图片可以是灰度图或RGB彩色图，也可以包含单个数字或多个数字，甚至是一组或多组数字。上传的图像随后将经历一系列预处理操作，最后将推理结果展示给用户。在这一过程中，系统不仅支持多种图片格式，还能智能识别并处理不同复杂度的数字输入。

当用户上传图片后，系统首先通过图像分析算法自动判断出图像中包含的数字数量。接着，系统将这些数字一一切割成单独的小图像，每个小图像仅包含一个数字。每个切割出的图像都会经过以下预处理步骤：

1. 灰度转换：首先，系统将每张图像转换为灰度图，以统一图像的色彩深度，避免RGB通道差异对模型推理造成影响。

2. 尺寸调整：将每个切割后的数字图像调整为固定尺寸（28x28像素），以匹配训练数据的输入格式，确保与神经网络的输入要求一致。

3. 二值化处理：接下来，系统将图像进行二值化处理，将背景设置为黑色，数字部分设置为白色。这一过程确保了图像的格式与网络训练时的格式完全一致，从而提高推理的准确性。

完成图像预处理后，系统将每个处理过的单独数字图像输入到已经训练好的神经网络中进行推理。神经网络将根据输入图像的特征输出对应的数字。对于包含多个数字的输入图像，系统将依次对每个小图像进行推理，并将所有的推理结果拼接在一起，形成完整的数字串。最终，系统将在GUI界面上显示识别出的数字结果，清晰地告知用户系统所识别出的数字。

**四、详细设计**

**1. 数据集预处理**

**（1）主程序部分：**

文件路径的定义：将数据集路径与文件名拼接，分别指定训练图片、训练标签、测试图片和测试标签的文件路径。这部分代码的目的是为后续的数据加载提供文件路径。

**（2）数据处理：**

调用processMNISTimages和processMNISTlabels函数，对训练和测试数据分别进行预处理，得到标准化图像数据和分类标签。

**processMNISTimages函数：**

功能：从 MNIST 数据集中读取图片数据，进行解码、重塑和归一化，最终返回一个标准化的四维数组。

工作原理：打开文件，验证是否可以正常读取。读取文件头的 magic number 并验证文件类型是否为图片数据。读取图像数量、行数和列数，解析出每个图像的分辨率。读取图像的像素值，并调整维度顺序，保证符合通道数的要求。对图像数据进行归一化处理，将像素值缩放到 [0, 1]。添加单通道维度，便于后续处理。

**processMNISTlabels函数：**

功能：从 MNIST 数据集中读取标签数据，将其转换为分类变量格式。

工作原理：打开文件，验证是否可以正常读取。读取文件头的magic number并验证文件类型是否为标签数据。读取标签数量。读取所有标签值，并将其转换为分类变量（categorical），以便于后续建模或分析使用。

% 数据路径设定

datapath = "./Mnist/"; % 指定MNIST数据集的存储路径

% 文件名拼接

filenameImagesTrain = strcat(datapath, "train-images-idx3-ubyte"); % 训练图片文件路径

filenameLabelsTrain = strcat(datapath, "train-labels-idx1-ubyte"); % 训练标签文件路径

filenameImagesTest = strcat(datapath, "t10k-images-idx3-ubyte"); % 测试图片文件路径

filenameLabelsTest = strcat(datapath, "t10k-labels-idx1-ubyte"); % 测试标签文件路径

% 调用预处理函数处理训练和测试数据

XTrain = processMNISTimages(filenameImagesTrain); % 处理训练图片，返回归一化后的图像数据

YTrain = processMNISTlabels(filenameLabelsTrain); % 处理训练标签，返回分类变量格式的标签数据

XTest = processMNISTimages(filenameImagesTest); % 处理测试图片，返回归一化后的图像数据

YTest = processMNISTlabels(filenameLabelsTest); % 处理测试标签，返回分类变量格式的标签数据

% 函数：处理MNIST图片

function X = processMNISTimages(filename)

[fileID,errmsg] = fopen(filename,'r','b'); % 打开文件，采用二进制模式读取；如果失败，返回错误信息

if fileID < 0

error(errmsg); % 如果文件无法打开，抛出错误

end

magicNum = fread(fileID,1,'int32',0,'b'); % 读取magic number，标志文件类型

if magicNum == 2051 % 验证magic number是否为图片文件的标志值

fprintf('\nRead MNIST image data...\n') % 输出提示信息

end

numImages = fread(fileID,1,'int32',0,'b'); % 读取图像数量

fprintf('Number of images in the dataset: %6d ...\n',numImages); % 输出图像数量

numRows = fread(fileID,1,'int32',0,'b'); % 读取图像的行数（高度）

numCols = fread(fileID,1,'int32',0,'b'); % 读取图像的列数（宽度）

X = fread(fileID,inf,'unsigned char'); % 读取剩余的所有图像像素数据

X = reshape(X,numCols,numRows,numImages); % 将一维数据重塑为 [宽度, 高度, 图像数量] 的三维数组

X = permute(X,[2 1 3]); % 调整维度顺序为 [高度, 宽度, 图像数量]

X = X./255; % 将像素值归一化到 [0, 1] 区间

X = reshape(X, [28,28,1,size(X,3)]); % 添加单通道维度，最终维度为 [28, 28, 1, 图像数量]

fclose(fileID); % 关闭文件

end

% 函数：处理MNIST标签

function Y = processMNISTlabels(filename)

[fileID,errmsg] = fopen(filename,'r','b'); % 打开文件，采用二进制模式读取；如果失败，返回错误信息

if fileID < 0

error(errmsg); % 如果文件无法打开，抛出错误

end

magicNum = fread(fileID,1,'int32',0,'b'); % 读取magic number，标志文件类型

if magicNum == 2049 % 验证magic number是否为标签文件的标志值

fprintf('\nRead MNIST label data...\n') % 输出提示信息

end

numItems = fread(fileID,1,'int32',0,'b'); % 读取标签数量

fprintf('Number of labels in the dataset: %6d ...\n',numItems); % 输出标签数量

Y = fread(fileID,inf,'unsigned char'); % 读取所有标签

Y = categorical(Y); % 将标签数据转换为分类变量格式

fclose(fileID); % 关闭文件

end

**2. 搭建并训练神经网络**

在MATLAB中，可以通过使用Deep Network Designer应用程序快速搭建神经网络。只需简单地拖拽各个神经网络组件并将其连接起来，系统会自动生成相应的代码。双击每个组件，进入设置界面，修改其详细属性和参数（参数设计见表1）。修改完成后，使用内置的分析工具对网络进行验证，确保网络配置正确无误后，即可导出生成的神经网络代码。

layers = [

imageInputLayer([28 28 1],"Name","imageinput")

convolution2dLayer([5 5],6,"Name","conv\_1","Padding","same")

tanhLayer("Name","tanh\_1")

maxPooling2dLayer([2 2],"Name","maxpool\_1","Padding","same","Stride",[2 2])

convolution2dLayer([5 5],16,"Name","conv\_2","Padding","same")

tanhLayer("Name","tanh\_2")

maxPooling2dLayer([2 2],"Name","maxpool\_2","Padding","same","Stride",[2 2])

fullyConnectedLayer(120,"Name","fc\_1")

fullyConnectedLayer(84,"Name","fc\_2")

fullyConnectedLayer(10,"Name","fc\_3")

softmaxLayer("Name","softmax")

classificationLayer("Name","classoutput")

];

导出后的网络代码如上。至此，LeNet-5网络的设计已完成，接下来就是修改训练参数，然后进行训练了。相关训练参数和训练代码如下：

options = trainingOptions('sgdm', ... %优化器

'LearnRateSchedule','piecewise', ... %学习率

'LearnRateDropFactor',0.2, ...%学习率下降比率

'LearnRateDropPeriod',5, ...%每个多少epoch进行学习率下降

'MaxEpochs',20, ... %最大学习epoch个数

'MiniBatchSize',128, ... %每次学习样本数

'ExecutionEnvironment','cpu',...%使用的训练设备

'Plots','training-progress' %画出整个训练过程

);

trainNet = trainNetwork(XTrain, YTrain, layers, options);

% 训练网络，XTrain训练的图片，YTrain训练的标签，layers要训练的网络，options训练时的参数

save Minist\_LeNet5 trainNet %训练完后保存模型

yTest = classify(trainNet, XTest); %计算模型的准确率

accuracy = sum(yTest == YTest) / numel(YTest);

fprintf('Accuracy on test dataset: %.4f\n', accuracy); %打印准确率

依次运行上面所有训练代码后，程序跳出训练窗口，可以实时的观察训练的准确率曲线和损失值曲线。

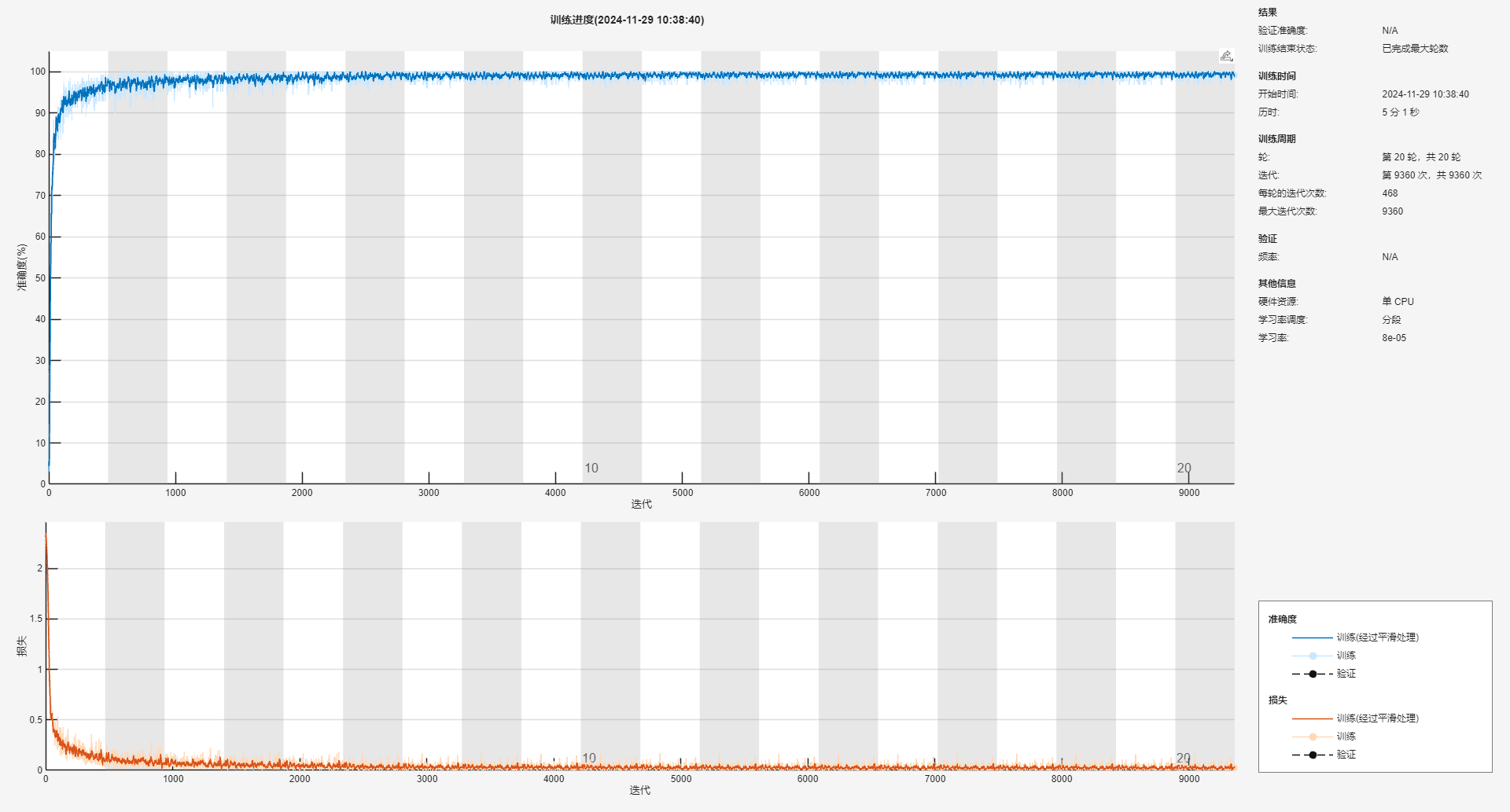


图5 训练结果图

**3. GUI推理设计**

我使用了MATLAB自带的GUIDE简单设计GUI界面。界面主要有三个组件，一个按钮用于点击后选择输入图片；一个图片展示区用于显示输入的图片；一个文本框用于展示推理结果（如下图6）。设计完成后，只需要使用槽函数将UI事件和处理函数连接起来，即可运行程序。

图形用户界面, 应用程序

描述已自动生成

图6 GUI界面布局

由于我训练的模型只能识别单个数字，所以当遇到多位数时，首先需要切割图像转化为单个数字，然后再进行推理。切割图像的关键函数说明如下：

**(1) 灰度图像转换 (rgb2gray)**

灰度图像转换是图像预处理的基础步骤，目的是将输入的彩色图像（通常包含RGB三个通道）转化为只有亮度信息的单通道图像。这个步骤简化了后续处理流程，因为灰度图像不再包含色彩信息，处理起来计算量更小，且边缘检测和区域分割等操作在灰度图像中更加高效和易于操作。转换过程中，RGB图像的每个像素通过加权平均法计算出其灰度值，公式如下：



其中分别是图像中某个像素的红、绿、蓝色通道的值。灰度图像的优势在于，接下来的边缘检测和区域分割等步骤不再受到颜色通道变化的影响，能够更好地聚焦于图像的形状和结构特征。

**(2) 边缘检测 (edge)**

边缘检测是图像处理中非常重要的一步，用于识别图像中物体的轮廓或边界。Canny 边缘检测算法是一种经典且效果优异的边缘检测方法，它通过以下几个步骤实现边缘检测：

1. 高斯平滑：首先对图像进行高斯滤波，减少噪声的干扰。

2. 梯度计算：通过计算图像中每个像素的梯度（即强度变化）来识别边缘。梯度大小反映了图像中亮度变化的速率，边缘通常对应着较大的梯度。

3. 非极大值抑制：沿着梯度方向抑制非边缘点，保留局部最大值，从而精确地确定边缘的细节。

4. 双阈值和边缘连接：设置两个阈值，将像素点分为强边缘、弱边缘和非边缘，并根据边缘的连通性进一步确认边缘位置。

Canny算法通过动态调整阈值，使得边缘检测灵敏度更适应不同图像的复杂度。检测出的边缘轮廓为后续的区域分割提供了基础，确保了算法能够准确识别出图像中的数字区域。

**(4) 区域生长**

区域生长是一种常用于图像分割的算法，它通过从图像中的一个初始点（称为种子点）出发，逐步扩展邻域，直到找到完整的连通区域。在手写数字识别中，区域生长主要用于分割图像中的各个数字。其基本步骤如下：

1. 选择种子点：在图像中找到一个合适的种子点，通常是图像中区域的中心部分。种子点选择的策略是寻找那些可能属于数字的区域，通常是那些与背景有明显区别的像素。

2. 扩展邻域：从种子点开始，检查其周围的像素（即邻域像素），如果邻域中的像素与种子点相似（通常基于像素的颜色、亮度或灰度值相似度），则将这些像素纳入当前区域。

3. 区域判断：区域生长会继续扩展邻域，直到当前区域不再扩展为止。每一次扩展都需要检查邻域像素与已知区域的相似度，并判断是否符合扩展条件。

4. 移除已处理区域：在处理完某一区域后，标记为已处理，确保算法不会重复处理同一部分区域。

区域生长算法可以有效地从图像中提取出连通的数字区域，确保每个数字区域被准确识别和切割。

**(5) 裁剪区域**

一旦图像中的各个数字区域被识别并提取出来，系统将根据每个区域的像素坐标从原始灰度图像中裁剪出对应的子图像。这一操作的步骤如下：

1. 提取区域坐标：在区域生长或边缘检测的基础上，算法会得到每个数字的边界坐标。

2. 裁剪数字区域：使用这些坐标从原始灰度图像中裁剪出每个数字所在的区域。为了确保数字的完整性，通常会留出一定的边距。

3. 扩展边界：为了避免裁剪时可能发生的边界切割问题，通常会对每个裁剪出的区域加上10像素的额外边界。这一扩展操作有助于保留数字的完整形状，并避免部分数字的细节被截断。

4. 保存和处理：所有裁剪后的子图像将存储在一个数组中，方便后续的操作（例如调整尺寸、二值化处理等）。这些子图像将作为网络输入的基本单元，供后续的推理过程使用。

**(6) 数字分组**

在一些情况下，用户上传的图像包含多个相邻的数字，这些数字可能由于间隔过小而被误判为一个数字。因此，分组操作非常重要，主要用于将这些相邻的数字分开。该操作步骤如下：

1. 图像扩大：为避免数字之间的间隙过小，首先将每个切割出来的数字图像四周加上一个人工设定的阈值进行图像扩展，阈值默认设定为100像素。

2. 相邻数字合并：对于经过扩展后的图像，如果相邻的数字图像发生了重叠，系统会认为这两个数字属于同一组，反之则视为不同组。

3. 归类数字组：对于识别出的每一组相邻数字，系统会为其分配一个唯一的编号，并将其归类到相同的组中。通过对数字区域进行分组处理，系统能够准确地识别并分隔多位数字。

4. 保存数字分组信息：每个分组的数字图像将被存储为一个单元数组，并且会记录该分组的标识信息，方便后续的操作。这样，系统可以对每一组数字进行单独处理并推理出最终结果。

通过这些精确的图像处理和分割算法，系统能够有效地识别和处理多位数的手写数字，提高了整体推理的准确性和鲁棒性。

function [I\_out, group] = grow(I, group\_threshold)

I1 = rgb2gray(I); % 转化为灰度图像

I2 = edge(I1, 'canny', graythresh(I) \* 0.7); % 边缘检测，使用Canny算法，阈值是灰度图像的全局阈值的0.7倍

I2 = imdilate(I2, ones(3, 3)); % 进行膨胀操作，扩展边缘

I2 = imfill(I2, 'holes'); % 填充孔洞

I2 = imerode(I2, ones(3, 3)); % 腐蚀操作，去除小的噪点

[x, y] = find(I2 == 1); % 找到二值化后的边缘部分（值为1的部分）

X = round(mean(x)); % 计算这些点的纵向位置平均值

I3 = I2(X, :); % 取出这一部分，即在原图像中选取了某一横行

wz = find(I3 == 1); % 找到该行中边缘部分的位置

j = 0; % 计数器，初始化为0

group = []; % 保存每个切片所属的分组信息

current\_group = 1; % 当前的分组编号，从1开始

while ~isempty(wz) % 当wz不为空时，进行迭代

j = j + 1; % 计数器+1

seed = [X, wz(1)]; % 选择当前横行中的第一个边缘点作为种子点

new = false(size(I2)); % 创建一个和I2同样大小的逻辑零数组，用来保存当前的二值图像

new(seed(1), seed(2)) = true; % 将种子点位置设为1，表示开始生长

while true % 开始生长过程

for i = 1:size(seed, 1) % 对每个种子点进行遍历

new(seed(i, 1)-1:seed(i, 1)+1, seed(i, 2)-1:seed(i, 2)+1) = I2(seed(i, 1)-1:seed(i, 1)+1, seed(i, 2)-1:seed(i, 2)+1); % 将种子点周围区域扩展到新区域

end

[x1, y1] = find(new == 1); % 找到新的种子点的位置

seed1 = [x1, y1]; % 更新种子点集合

if length(seed) == length(seed1) % 如果本次和上次的种子点数目相同，说明生长已经完成

break; % 退出生长循环

end

seed = seed1; % 更新种子点

end

% 找到当前生长区域的x和y坐标

pd = find(new(X, :) == 1);

wz = wz(length(pd) + 1:end); % 去除已经生长完成的部分，更新wz

[x, y] = find(new == 1); % 获取当前生长区域的所有坐标

% 动态调整边界值

min\_x = max(min(x) - 10, 1); % 最小横坐标，加入了偏移量，防止边界过于紧凑

max\_x = min(max(x) + 10, size(I1, 1)); % 最大横坐标

min\_y = max(min(y) - 10, 1); % 最小纵坐标

max\_y = min(max(y) + 10, size(I1, 2)); % 最大纵坐标

out = I1(min\_x:max\_x, min\_y:max\_y); % 截取出该区域的灰度图像

% 判断当前生长区域与上一切片是否属于同一组

if j > 1

prev\_min\_y = group(j-1).bounding\_box(3); % 获取前一个切片的右边界（即纵坐标的最小值）

if min\_y - prev\_min\_y <= group\_threshold % 如果当前切片的左边界与前一个切片的右边界之间的差值小于阈值，则认为它们属于同一组

group(j).group\_id = group(j-1).group\_id; % 将当前切片分配给前一组

else

current\_group = current\_group + 1; % 否则为新的一组

group(j).group\_id = current\_group;

end

else

group(j).group\_id = current\_group; % 第一个切片初始化为第一组

end

% 保存当前切片的信息，包括分组编号和边界框

group(j).bounding\_box = [min\_x, max\_x, min\_y, max\_y];

group(j).image = out;

% 存入输出数组

I\_out(j) = mat2cell(out, size(out, 1), size(out, 2)); % 将当前切片的图像存储在cell数组中

end

if length(I\_out) == 1 % 如果输出数组只有一个切片

I\_out = mat2cell(I1, size(I1, 1), size(I1, 2)); % 返回原始图像的cell数组形式

group(1).group\_id = 1; % 单个图像的分组编号为1

end

end

用户上传图片以后，程序的关键处理函数如下。pushbutton1\_Callback是MATLAB GUI 的回调函数，当按下一个按钮时触发。其主要功能是：

1. 通过GUI的文件选择器选择一幅图像。
2. 使用上面grow函数处理并分割图像中的对象。
3. 对分割出的每个对象使用预训练的LeNet-5神经网络进行分类。
4. 根据group数组判断每个分割出来的对象属于哪个组，输出时在不同组之间添加空格用以区分。
5. 将识别结果显示在GUI的文本框中。

function pushbutton1\_Callback(hObject, eventdata, handles)

[Fnameh, Pnameh] = uigetfile({'\*.\*'; '\*.jpg'; '\*.png'}); % 弹出文件选择对话框，选择图像文件

filename = [Pnameh, Fnameh]; % 合成文件的完整路径

I = imread(filename); % 读取选中的图像文件

imshow(I); % 显示读取的图像

% 设置图像分割时的阈值，用于判断分割结果

group\_threshold = 120; % 可调的阈值，控制分组的分割效果

[I\_out, group] = grow(I, group\_threshold); % 调用grow函数进行图像分割，返回切片结果和分组信息

% 加载已经训练好的LeNet-5模型

load("Minist\_LeNet5.mat", 'trainNet'); % 加载训练好的神经网络模型，包含手写数字识别网络

% 初始化一个数组，存储每个切片的识别结果

L = length(I\_out); % 获取切片的数量

num = zeros(1, L); % 初始化一个零数组，用来存储每个切片的识别结果

for i = 1:L % 对每个切片进行遍历

I1 = I\_out{i};% 获取当前的切片图像

figure, imshow(I1); % 可视化当前的切片图像

% 对图像进行预处理，准备进行分类

img = preprocessImage(I1); % 调用图像预处理函数，将切片处理为网络可以接受的格式

% 使用训练好的深度学习模型对图像进行分类

result = classify(trainNet, img); % 使用训练好的网络进行预测分类

num(i) = double(result) - 1; % 将分类结果转换为数字标签（假设分类输出是字符类标签 '0'-'9'）

end

% 按照组进行拼接数字，并在每组之间插入空格

result = ""; % 初始化结果字符串

for g = 1:max([group.group\_id]) % 遍历所有的分组

group\_digits = num([group.group\_id] == g); % 获取当前组的所有识别数字

result = result + join(string(group\_digits), '') + " "; % 将当前组的数字拼接在一起，并加上空格

end

% 显示最终结果到界面上的文本框中

set(handles.edit1, 'string', strtrim(result)); % 更新GUI中的文本框，显示识别结果

end

**五、仿真结果分析**

运行程序，打开GUI界面，根据提示上传输入的图像。上传的图片无特殊要求，可以是手机拍照的图片，也可以是屏幕截图的图片，还可以是彩色数字，不连续的间断数字，甚至扎堆的数字图片，均可以识别。但需要尽量保证手写的数字和背景图颜色差异较大，否则难以切割图片。

**1. 单个数字检测**

上传写有单个数字的图片，可以是彩色图片也可以是截图的黑白图片。



图7 单个数字检测结果

**2. 多位数检测**

上传写有多个数字的图片，程序先进行图像分割，然后进行识别。

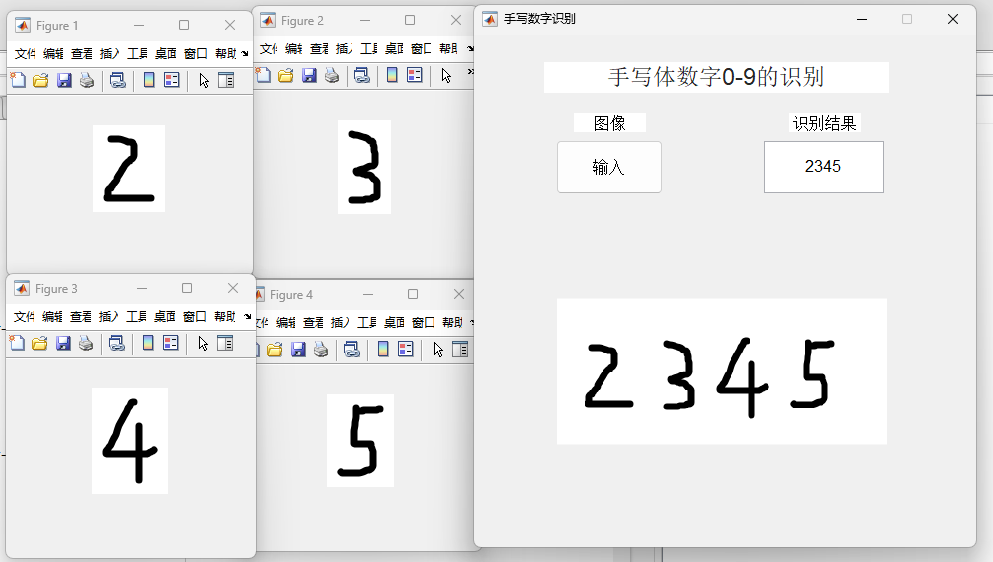


图8 截图多位数检测

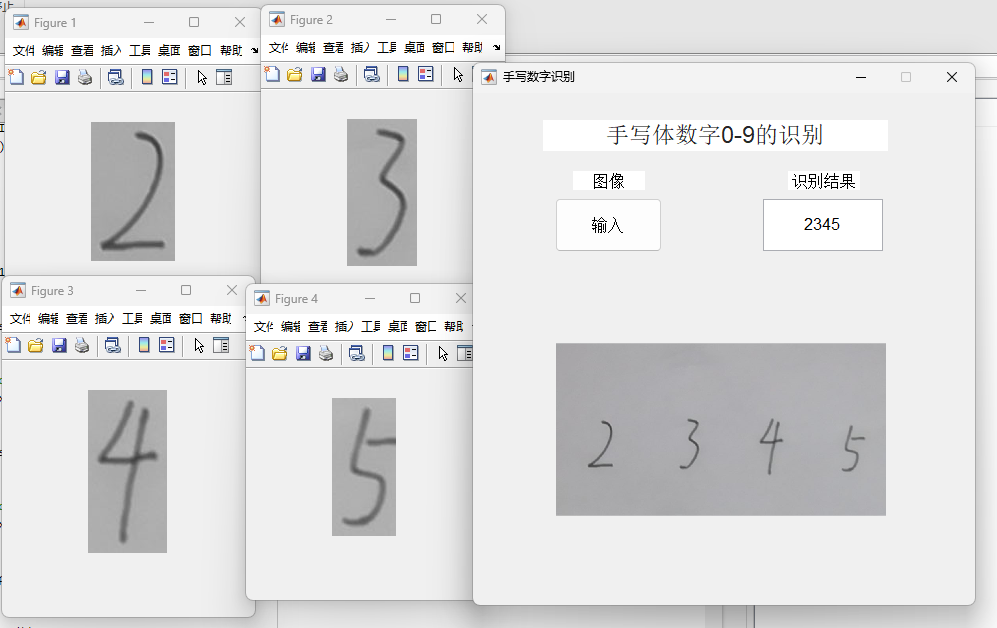


图9 拍照多位数检测（分组阈值调大）

对于间距过大的情况，由于边缘检测和生长算法，也可以很好的排除空余部分，实现准确精准的预测。并且会归类到不同的分组中，不同分组的数字之间通过空格隔开。

图形用户界面, 应用程序

描述已自动生成

图10 数字间距过大的多位数检测

对于分组扎堆的数字，可以根据手动设置的阈值，判断出不同的分组，分组和分组之间使用空格隔开。

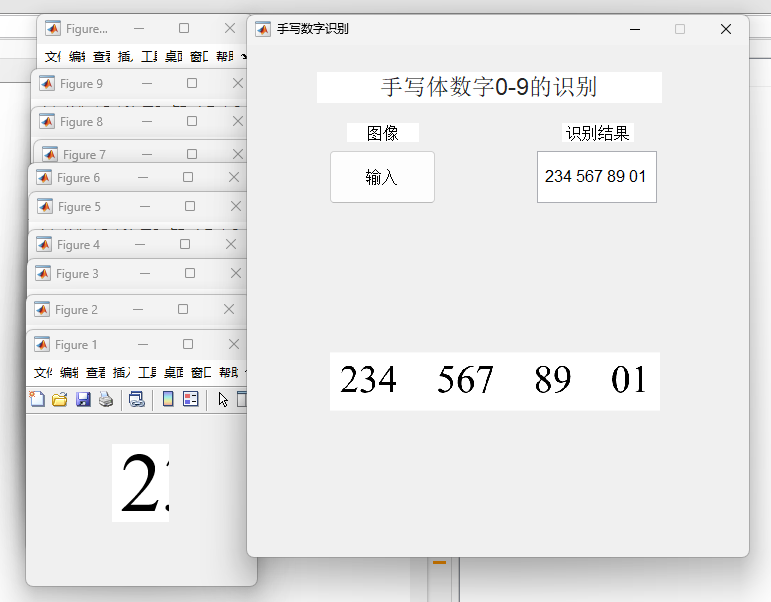


图11 Word打印的扎堆数字识别

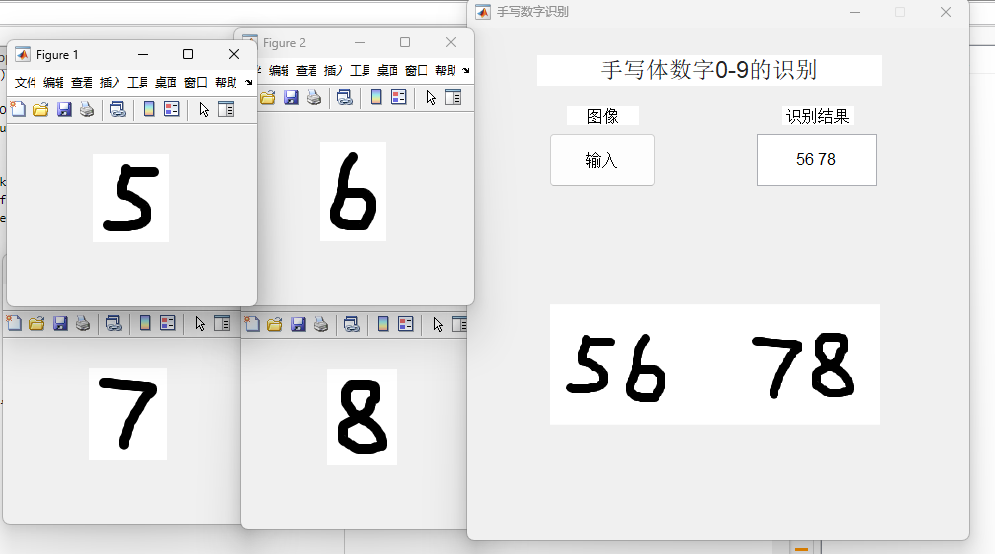


图12 手写的扎堆数字识别

**3. 彩色数字识别**

对于彩色数字，在预处理时，先将其转化为灰度图，然后进行颜色增强，再进行判断，所以也可以很好的判断出来。

图形用户界面, 应用程序

描述已自动生成

图13 简单彩色数字识别

图形用户界面, 应用程序, Word

描述已自动生成

图14 较为复杂的彩色数字识别（左侧小图仅展示4个）

对于背景有颜色的图片，只要数字颜色和背景颜色差值较大，转成灰度后也能轻松被识别准确：



图15 背景较浅的彩色数字识别（左侧小图仅展示4个）

对于背景颜色较深的彩色数字识别，需要一定的色差才能准确识别，以下是一个成功识别的案例展示：

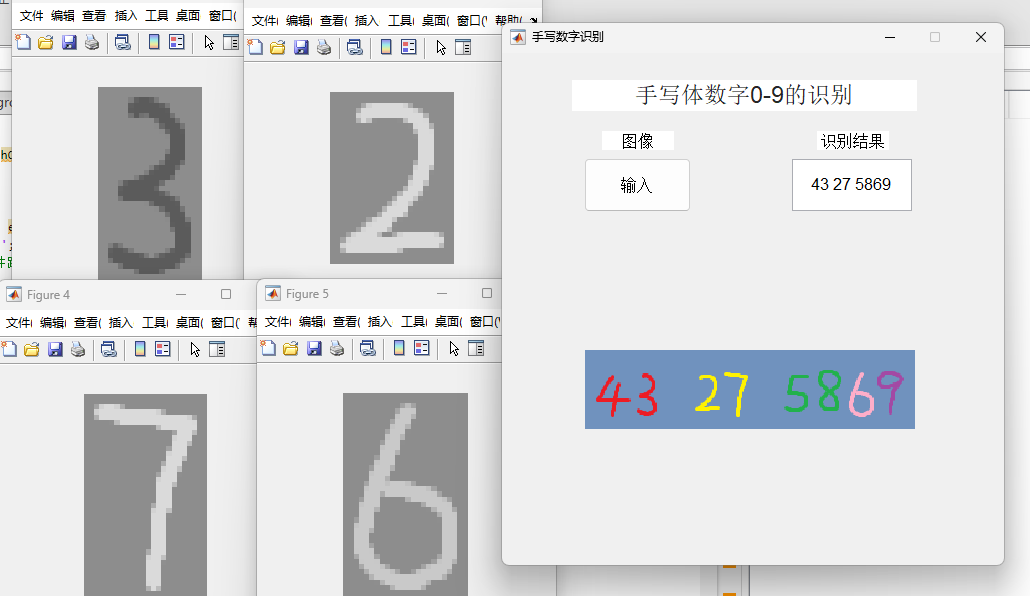


图16 背景较深的彩色数字识别（左侧小图仅展示4个）

**六、仿真过程中的问题**

在仿真过程中，模块设计和调试的过程中主要面临以下问题，这些问题通过我的多次尝试和优化逐步得到了解决：

**问题 1：一开始只能单数字识别，后来改进为多数字识别**

① 问题描述：

初期设计时，系统只能识别图像中单个数字，无法处理多数字的场景。这种局限性使得系统无法满足实际应用需求，比如处理手写数字表单或连续数字的图像。

② 问题解决：

通过引入图像分割模块（grow函数）对输入图像进行切割，将包含多个数字的图像分割为多个子图，每个子图包含一个单独的数字。这样，每个子图可以独立传递给神经网络进行分类，从而实现了多数字识别。

③ 关键代码：

I\_out = grow(I); % 使用区域生长算法对图像进行分割

L = length(I\_out); % 获取分割后的子图数量

num = zeros(1, L); % 存储每个子图的识别结果

for i = 1:L

I1 = cell2mat(I\_out(i)); % 获取单个子图

img = rgb2gray(imresize(I1, [28, 28])); % 转为灰度图，调整尺寸

img = imbinarize(img, 0.5); % 二值化处理

img = reshape(imcomplement(img), [28, 28, 1]); % 调整输入格式

result = classify(trainNet, img); % 使用训练好的网络分类

num(i) = double(result); % 将分类结果保存

end

④ 优化结果：

改进后，系统能够从一张包含多个数字的图像中提取出单个数字，并逐个分类，大大提高了识别的适用性和灵活性。

**问题 2：多数字划分切割图像算法改进**

① 问题描述：

在实现多数字识别的过程中，图像分割的算法曾经采用投影法，通过统计图像像素的水平和垂直分布来判断切割位置。然而，投影法在处理某些复杂场景时（如数字间隔不均、背景噪声等）表现出较低的准确性，导致分割出的子图不准确，从而影响后续分类的精度。

② 问题解决：

为了提高分割的准确性，投影法被替换为基于区域生长的分割算法（grow 函数）。区域生长算法通过检测连通区域来提取每个数字，从而更适应不同数字间隔和形态的变化，避免了投影法对于像素分布均匀性的强依赖。

**投影法代码示例（早期实现）：**

% 投影法分割

proj = sum(I, 1); % 计算垂直方向上的投影

threshold = mean(proj) \* 0.5; % 设定阈值

segments = proj > threshold; % 根据投影切割图像

**改进后区域生长算法关键代码：**

% 区域生长算法分割

seed = [X, wz(1)]; % 初始种子点

new = false(size(I2)); % 初始化逻辑矩阵

new(seed(1), seed(2)) = true; % 设置种子点

while true

for i = 1:size(seed, 1)

% 扩展种子点，覆盖邻域3×3范围

new(seed(i, 1)-1:seed(i, 1)+1, seed(i, 2)-1:seed(i, 2)+1) = ...

I2(seed(i, 1)-1:seed(i, 1)+1, seed(i, 2)-1:seed(i, 2)+1);

end

% 更新种子点

[x1, y1] = find(new == 1);

seed1 = [x1, y1];

if length(seed) == length(seed1) % 若生长稳定，退出循环

break;

end

seed = seed1;

end

③ 优化结果：

改用区域生长算法后，分割精度显著提升，尤其是在处理数字间隔较大或背景复杂的图像时，分割结果更加稳定。识别流程变得更加鲁棒，能够应对更广泛的实际场景。

**其他算法改进设想**

在现有基础上，还可以进一步优化以下几个方面：

1. 增强图像预处理：

引入噪声去除算法，如中值滤波（I = medfilt2(I, [3, 3])）或形态学操作，以减少背景干扰。对分割后的子图进行几何校正，统一数字方向和位置，进一步提高神经网络分类的准确性。

② 自适应分割策略：

结合图像内容动态调整分割参数，如区域生长的种子点选择或阈值设定，使算法更具灵活性。

③ 集成学习算法：

在现有分类器（LeNet-5）的基础上引入集成学习模型，通过多个神经网络投票，提高分类的稳定性和准确性。

results = [classify(net1, img), classify(net2, img), classify(net3, img)];

final\_result = mode(results); % 投票法得出最终分类结果

**七、课程设计总结**

本次课程设计的目标是利用MATLAB实现手写数字识别。从程序的初步实现到功能的逐步完善，我经历了算法设计、代码实现与调试优化的完整流程，收获了许多关于MATLAB应用的实践经验，也提升了分析和解决问题的能力。

在本次设计中，我深刻体会到MATLAB在图像处理与数据分析中的强大功能。从灰度化、二值化到边缘检测与连通域分析，MATLAB内置的函数简化了算法的实现，也让我能够快速完成不同算法的比较与优化。例如，在图像分割环节，我初步尝试了投影法，并通过MATLAB的调试工具观察到其不足之处；随后转为使用区域生长法，借助连通域分析功能，大幅提高了分割的准确率。

在识别部分，我学习并应用了深度学习相关的功能模块，将神经网络模型与图像预处理结合，完成了多数字图像的自动识别。数据预处理阶段的一些细节处理（如调整图像尺寸、二值化和背景调整）让我更深入理解数据对模型性能的影响，也积累了关于MATLAB应用于分类任务的宝贵经验。

调试是程序设计过程中不可或缺的一部分，通过此次设计，我的调试能力得到了显著提升。MATLAB的可视化工具为调试提供了很大帮助，例如利用imshow函数直观展示每一步处理后的图像结果，快速发现算法问题。区域生长法的调试过程中，我通过逐步调试逐步分析每轮种子点生长情况，从而优化了生长规则，提高了算法性能。

此外，我还在调试过程中发现了代码效率问题。例如，分割与识别模块中存在多次重复计算，经过优化，消除了冗余代码，大幅提升了程序运行速度。这让我认识到调试不仅是发现和修复错误的过程，更是优化程序的重要手段。

通过本次课程设计，我掌握了MATLAB在图像处理与数据分析中的许多实用功能，对图像分割与识别技术也有了更加全面的了解。从最初的单数字识别到最终的多数字识别，从算法的初步实现到逐步优化，我经历了一次完整的实验与仿真流程，这让我在程序设计与调试能力上都得到了极大的提升。这次设计的经验将为我今后解决实际问题和进一步学习相关技术奠定坚实基础。