教育部高等学校大学计算机课程教学指导委员会

中国大学生计算机设计大赛

大数据/人工智能挑战类作品文档简要要求

作品编号：61411

作品名称：人工智能战项目四：手写数学推理

作　　者：刘洪

版本编号：

填写日期：2019 年 5 月 16 日

填写说明：

1、本文档适用于大数据挑战赛小类、人工智能挑战赛小类；
2、正文一律用五号宋体，一级标题为二号黑体，其他级别标题如有需要，可根据需要设置；
3、本文档为简要文档，不宜长篇大论简明扼要为上；
4、提交文档时，以 PDF 格式提交本文档；
5、本文档内容是正式参赛内容组成部分，务必真实填写。如不属实，将导致奖项等级降低甚至终止本作品参加比赛。

目　录

第一章 项目背景

数学公式的识别是具有挑战性且有重大现实意义的研究领域。数学公式识别的关键问题是检测和分类数学符号、分析符号布局，并构建意义表示。

第一节 系统概述

本系统主要包括两部分:公式识别和公式计算。

公式识别部分包含如下步骤：（1）首先我们对图像进行预处理，主要包括图像二值化、噪声滤除和图像倾斜矫正；（2）其次我们对图像进行了分割，从数学公式图像中分割出带有属性的公式符号串，我们使用动态比较字符骨架方法反复分割公式字符，主要步骤为：先初始分割公式图像（过滤根号、分式等），然后反复从竖直方向、水平方向用投影法分割公式候选区域，对候选区域进行基于特征提取的模式匹配识别，对低于一定置信度的候选区域拒识别；（3）对于拒识别的候选区域将重复应用上述分割。为了提高识别精度，对于被识别的候选区域，将采用我们训练的 CNN 神经网络进行再识别，若低于一定置信度，则说明该字符可能还存在粘连等现象，此时将会对其进行过分割，对字符粘连和多结构字符进行处理；（4）根据字符属性和公式分析模型对字符进行连接。

在公式计算方面，我们按照大赛要求，将识别出的字符串进行解析，建立树形结构，在此基础上进行化简和计算。

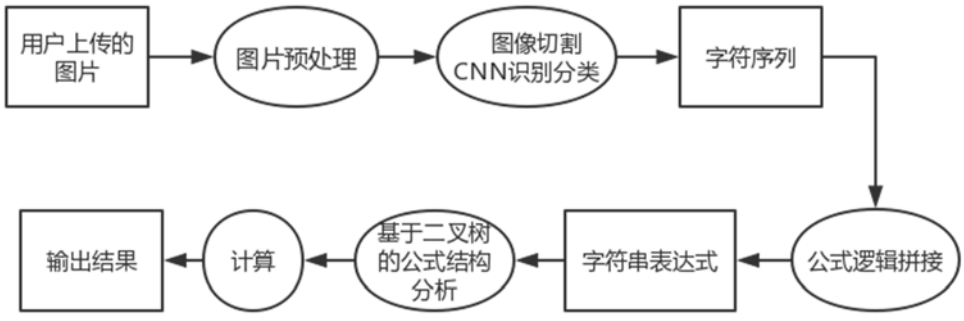


图 1.1 系统概述

第二节 系统比较

手写文字识别是图像处理与模式识别重要的研究领域，手写数学公式符号识别更是受到了学术界和工业界的长期关注。本项目看起来与生活密切相关，似乎有许多较为完善的开源解决方案，但经过我们的仔细检索，商业产品中只有拥有五年研发周期的 PhotoMath 完全符合项目设计要求，而各种开源的手写公式识别代码几乎都止步于简单的四则运算，一定程度上说明了该选题的难度。

虽然没有成体系的开源参考资料，但是很多文献中对手写公式识别的局部都有做深入研究，本系统参考各种文献中的方法，并进行总结，对一些方法进行改进，最终搭建了自己的作品。

表 1.1 系统方法比较

	已有方法	本项目采用方法
字符切割	连通体分析	连通体分析，投影法，归一化动态比较字符骨架的基础上进行重切分
	投影法	
	字符细化，动态比较骨架曲线特征	
字符识别	模式匹配	粗识别：特征提取、模式匹配 根据候选字列表进行重切分、过分割处理 CNN 神经网络再识别分类 再检验：结构分析，纠错
	结构方法	
	统计分析	
	机器学习	
公式拼接	符号关系树：不能处理多行以及复杂的单行表达式	识别待分析公式类型,并根据公式类型实现公式唯一分解；定义了 10 种公式类型，使用产生式规则定义每种类型的分解办法、分解后的子表达式顺序以及子表达式的类型；具体分解时则采用了基于结构的分析方法
	自顶向下和自底向上相结合：上下角标和矩阵处理不完善，不能处理帽子表达式	
	文法分析：处理公式类型有限	
	图改写：文法复杂、计算量大，处理公式类型简单	
输出	Latex； MathML	自定义字符串格式
	Lisp	
错误校正	启发式规则修改错误识别结果	定义公式部分规则，检查字符串
	词法错误：字符识别结果和后处理手段	
	句法错误：产生式规则	
	语法错误：不通用的启发式规则	
字符串解析计算	表达式树形式	赛事规定使用表达式树形式
	堆栈形式	

第三节 改进与创新

1、 动态字符切分——基于相似度判定的递归切分

我们发现因为手写公式可能存在连笔、断笔等现象，并不像打印文件那样规整，并且，有着像根号，分式这种形式的式子，结构复杂，所以无论是连通体分析、投影法上述哪种方法单独使用，都达不到预期的识别效果。

我们对传统的单一切分模式进行改进，将数种切割方法进行组合，使用比较字符骨架的动态识别，进行反复切割，然后利用特征提取、模式匹配的方法进行粗识别，判断是否需要继续切割，如果置信度达到一定阈值，我们将会暂时认为它已经被切分成最小候选区域。

因为特征提取这种非机器学习的局限性，此时不仅字符识别的正确率很低，而且仍会存在许多字符粘连等问题，我们经过反复尝试，最终发现将机器学习和非机器学习相结合的方法，效果会比较好。

所以此时我们将会把初步切分好的候选区域送入 CNN 神经网络进行再识别，如果 CNN 神经网络返回的置信度低于一定阈值，那我们会再次尝试将候选区域进行过分割，以保证不会有漏判小数点这种细小符号，尽可能多获取字符块，然后按照置信度进行两步后处理，或采纳过度切割结果以解决字符粘连问题，或是对多结构字符进行合并，最终获得了比单独应用一种方法更高的切分、识别准确率。

2、 字符识别——传统模式识别方法与卷积神经网络的配合使用

在识别方面，我们刚开始尝试了各种单一方法。

包括用特征提取模式匹配方式、传统的 KNN 监督分类方法、Tesseract-OCR 光学字符识别等，经过我们不断地试验、训练以及精度对比，发现以上非机器学习的识别方法对于字形固定且标准的打印字符有着不错的效果，但是对于灵活多变的手写字符普遍存在“顾此失彼”的问题，即一旦加强某一个字符的训练强度，那么不仅该字符的识别率会提高，而且很多相似的其他字符也会被识别成该字符，从而导致整体的识别精度降低。

而机器学习中我们也尝试了许多模型，例如用 DenseNet 神经网络识别整式、VGG weightnoise 模型、卷积神经网络（CNN）来进行单字符识别，发现若用神经网络来识别整式，则神经网络既要识别单字又要拼接公式，负担太大，识别准确率很低，若只识别切割完好的单个字符，神经网络的识别率是很高。但若切割的不是很完好，神经网络可

能会把一个未完全切割或过度切割字符识别成一些本就书写复杂的特殊符号（如Σ），或组合字符号（如 lim、cos、sin 等），并给其很高的置信度，导致全式的结构都产生混乱，准确度也不是很高。

基于上述问题，我们提出了模式匹配与神经网络共同配合的新系统。一方面在切割时用模式识别进行粗识别，只有高于一定置信度才会传入 CNN 进行再识别，如果 CNN 返回的置信度过低，也会重新再对候选区域进行过分割，查看其是否为粘连字符。这样比较稳定的提高了表达式整体的识别率。

3、 数据集整理——测试结果反馈驱动型的训练数据集调整方案

神经网络的兴起基于数据爆炸，这是一种从数据中发现特征、学习特征的方法，所以一定意义上训练数据集的好坏就决定了整个网络的识别效果。

我们找到的原始数据有 82 个标签，包含超过 370000 张字符图像。但是经过测试发现数据集上的主要问题是有些训练数据特征不明显甚至会对识别造成干扰，有许多字符的书写方式与国人书写方式有着极大差异，为了解决这个问题，我们重新整理了训练数据，花费大量时间，手动去除了一些字形离谱容易造成特征混淆的数据，也去除了一些欧美风格的手写字符使得整体上符合国内的手写风格。

经过第一次整理，整体识别率提高了很多。为了细化数据集优化过程，我们将调整前后的测试结果逐一对比，分析识别错误的原因，将测试图像的字形与对应训练数据集中的字形进行比较，发现其中的相似特征，并提出相应的调整方案，并分工下去调整并重新训练测试。经过反复的调整最终得到现在的 99.64%识别率的神经网络模型。

4、 优化细化函数——采用 Rosenfeld 细化算法进行优化

细化的算法有很多种，比较常用的算法是查表法。

判断一个像素点是否能去掉的主要依据是 8 个相邻点（八连通）的情况，具体判据为：（1）内部点不能删除。（2）孤立点不能删除。（3）直线端点不能删除。（4）如果 P 是边界点，去掉 P 后，如果连通分量不增加，则 P 可删除。但是这样细化出来的图像并不是很平滑，会有很多毛刺，会对识别造成一定的干扰。

于是我们参考 Rosenfeld 细化算法对细化函数进行了优化，Rosenfeld 细化算法描述如下：扫描所有像素，如果像素是北部边界点、南部边界点、东部边界点或西部边界

点，且是 8simple（即：将该点设置为 0 后，不会改变周围 8 个像素的 8 连通性），但不是孤立点和端点，删除该像素。执行完上述步骤后，就完成了一次迭代，我们重复执行上面的迭代过程，直到图像中再也没有可以删除的点后，退出迭代循环。

改变思路，在每行水平扫描的过程中，先判断每一点的左右邻居，如果都是黑点，则该点不做处理，如果某个黑点被删除了，则跳过它的右邻居，处理下一点。对矩形这样做完一遍，水平方向会减少两像素。

然后我们再改垂直方向扫描，方法一样。这样做一次水平扫描和垂直扫描，原图会细化掉水平垂直各两个像素，而且基本保持图像中心收敛。

多次重复上面的步骤，直到图形不在变化为止，我们就得到了新的字符骨架。可以看出，虽然算法的时间复杂度增加了一个级别，但是比查表法的细化函数优化了很多，毛刺减少，趋于平滑。

用新的细化函数处理过后的图像，识别率增长较为明显。

图 1.2 细化函数优化前的图像

图 1.3 细化函数优化后的图像

第四节 测试结果

1、 对于单个字符，我们已经达到了很高的识别率，神经网络的测试集图像 6393 张，测试精度高达 99.64%，测试我们切分好的手写字符图像集，各类符号共 1210 张，识别率达到 96.73%。

表 1.2 系统能识别的 73+8 个字符

序号	类别	字符示例
1	数字	0 1 2 3 4 5 6 7 8 9
2	英文字符	ABCDEFGHIJKLMNPRSTUVWXYZ abcdefghijklmnprstuvwxyz
3	希腊字符	α (alpha) β (beta) Δ (delta) γ (gamma) λ (lambda) μ (mu) Φ (phi) π (pi) σ (sigma) θ (theta) 【δ ε η ρ ϕ ψ ω】
4	数学字符	+ − × ∗ ÷ / = ∞ ∫ √ Σ ≠ > < ⩽ ⩾ 【∮】
5	特殊字符	（ ） [ ] { } ∃(exists)∀(forall)∈(in)→（rightarrow）
	组合字符	sin cos tan log lim

注：【】中的为神经网络中不包含但特征匹配中包含的字符

2、对于整式识别，我们每类式子准备了 10 个左右，邀请了 20 名同学，将每个式子抄写了 1 遍，共得到测试数据图像 1998 张，进行识别测试。

表 1.3 系统能处理的表达式

类型	示例	题目数	样本/正确求解数	识别率
算术表达式	$5+3\times7\div2$	15	298/282	94.63%
分式表达式	$\frac{5}{4}+\frac{8}{17}$	13	260/245	94.23%
根式表达式	$\sqrt[3]{125}$	10	200/189	94.50%
乘方表达式	2^5-4^6	12	240/213	88.75%
三角函数表达式	$\sin(45)+\cos(60)$	10	200/171	85.50%
方程	$x^3+x^2+4x+3=1$	9	180/155	86.11%
方程组	$\begin{matrix}x+y=9\\x-y=0\end{matrix}$	6	120/95	79.16%
积分	$\int_0^1 x\,dx$	7	140/115	82.14%
微分求导	$\frac{d}{dx}x^2+5$	4	80/58	72.50%
极限	$\lim_{x\to0}\frac{\sin x}{x}$	4	80/51	63.75
求和	$\sum_{k=1}^{n}\frac{k}{k^2+1}$	4	80/59	73.75%
矩阵	$\begin{bmatrix}4&5&3\\8&9&6\end{bmatrix}A=\begin{bmatrix}2\\7\end{bmatrix}$	6	120/103	85.83%
总计		100	1998/1736	82.38%

第二章 算法与数据来源

第一节 数据来源

一、传统模式识别训练数据

队员自行手写制作完成，这里由于传统模式识别的“顾此失彼”问题，所以每一个标签并不需要太多的训练数据，这也为我们自行制作手写训练数据成为可能。具体的处理步骤为原始图像——预处理——细化——切割——分类放入对应文件夹。

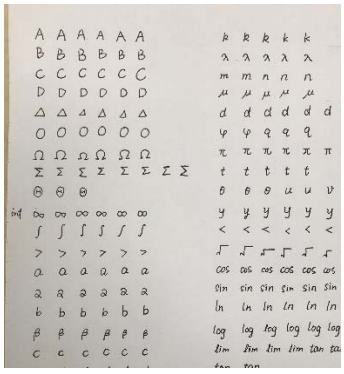


图 2.1 原始图像

图 2.2 预处理后的图像

图 2.3 细化之后的图像

二、神经网络训练数据

1、 来源

训练数据来源为 https://www.kaggle.com/xainano/handwrittenmathsymbols

为开源手写数学符号训练集，取自由国际数学公式识别竞赛组织 (CROHME)提供的标准数学公式符号库。

2、规模

原始数据有 82 个标签，包含超过 370000 张字符图像。

3、 处理过程

第一次训练，我们以 10%的数据作为测试集来检测精度，测试精度高达 98%，但是用来识别我们的手写图像，正确率却不到 10%。我们分析造成这种结果的原因之一可能是因为训练数据特征不明显甚至会对识别造成干扰，比如：

图 2.4 ‘1’：

图 2.5 ‘2’：

图 2.6 ‘7’：

上图可见，训练集中有许多字形离谱容易造成特征混淆的数据，还有一些欧美风格的手写字符，其书写风格与国人书写习惯并不符合，导致我们识别自己的手写字符效果并不佳。

基于上述发现我们对训练集进行了重新整理，我们花费大量时间，手动去除了一些字形离谱容易造成特征混淆的数据，也去除了一些欧美风格的手写字符使得整体上符合国内的手写风格。

经过第一次整理，整体识别率提高了很多。为了细化数据集优化过程，我们将调整前后的测试结果逐一对比，分析识别错误的原因，将测试图像的字形与对应训练数据集中的字形进行比较，发现其中的相似特征，并提出相应的调整方案，并分工下去调整并重新训练测试。这样反复测试调整，直到达到我们预期的识别效果。

最终确定下来的数据集包括 73 个标签，训练图像 287683 张，测试图像 6393 张，共 313255 张图像，各标签的数据集大小分布见图 2.7。

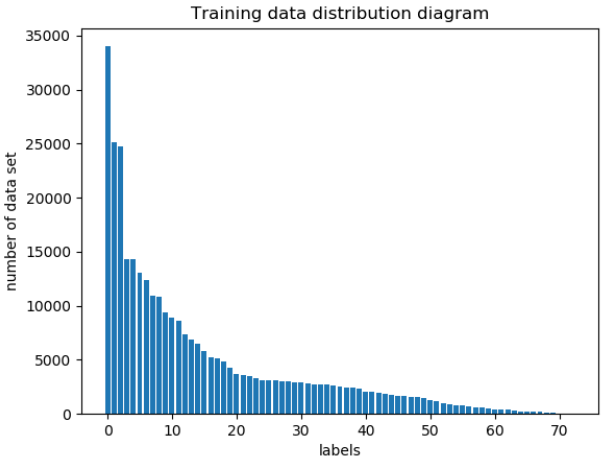


图 2.7 数据集分布

由于不同的数学字符的使用频率不同，因此整体的训练数据分布与使用频率有很强的相关性。数字以及常用的字母（如 x, y）的数量较多，而不太常用的希腊字母数量相对较少。

三、神经网络测试数据

因为目前网络上并没有系统的数学公式图像，所以所有的测试图像均为队员手写制作完成，共 279 例。制作过程类似制作传统模式识别训练数据的过程，但是不用切割出单个字符，只需切割出单个公式即可。

最终测试时，我们共可识别 12 类数学公式，每类式子准备了 10 个左右，共计 100 个公式模板，邀请了 20 名同学，将每个式子抄写了 1 遍，共得到测试数据图像 1998 张。

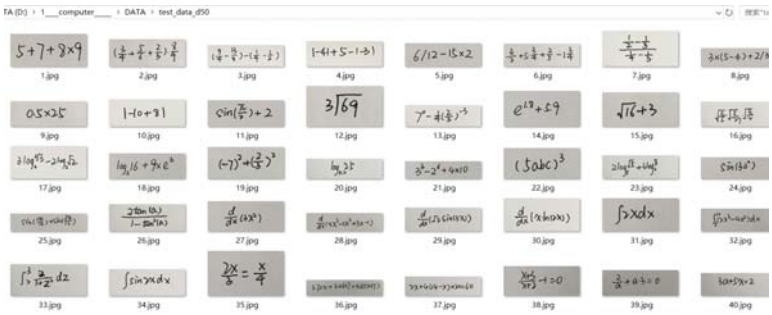


图 2.8 部分测试数据

第二节 算法来源

一、传统模式识别算法

我们采用的是统计模式识别方法，统计模式识别是首先根据待识别对象的所包含的原始数据信息，从中提取出若干能够反映该类对象某方面性质的相应特征参数，并根据识别的实际需要从中选择一些参数的组合作为一个特征向量，根据某种相似性测度，设计一个能够对该向量组表示的模式进行区分的分类器，就可把特征向量相似的对象分为一类。

统计模式识别过程主要由 4 个部分组成，分别有信息获取、预处理、特征提取和选择及构造分类器，构造分类器即对特征向量进行区分的分类器的设计、用设计好的分类器进行最终的分类决策。

具体实现参考 https://github.com/woshiwpa/CoreMathImgProc 开源项目中的训练过程及方法。并加入我们自己制作的训练集进行重新训练。

下图为用我们自己的训练图像提取的矩阵和存入.java 文件的示例

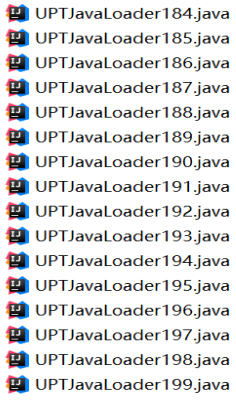


图 2.9 存入文件示例

图 2.10 训练图像提取的矩阵

二、CNN 神经网络模型

1、 模型搭建

我们在第三章系统实现中对该网络进行了详细叙述，这里仅作简单介绍。

原始模型为：https://github.com/anujdutt9/Handwritten-Digit-Recognition-using-Deep-Learning/tree/master/CNN_Keras，我们对其进行了一系列优化调参。

模型的搭建分三个主要阶段：

第一阶段是输入数据。数据作为 N-d 像素阵列提供。

第二阶段为构建卷积网络体系结构，我们使用 Keras 的 Sequential 类来构建网络。该网络由 3 个卷积层、3 个池化层、2 个全连接层组成，每个卷积层和第一个全连接层后面都连接了一个 ReLU 激活函数，最后一层的全连接层通过 softmax 输出结果

第三部分为全连接层，该层由 512 个神经元组成。我们应用 dropout 来减少过拟合，并使用 Softmax 分类器返回 73 个类标签中每个标签的概率列表。

详细参数见第三章系统设计与实现。

2、 模型训练

准备好训练数据并搭建好上述模型后就可以进行模型训练了，训练需要确定以下几个关键部分：

- 批量和迭代次数

  batch=128:由于训练数据集较大，但是特征分布相对比较集中，所以我们选择相对较大的 batch，经过多次比对实验结果最终确定单次学习 128 张图像。

  epoch=20:迭代次数的确定也经过了反复测试，权衡计算量与准确率。

- 损失函数：交叉熵损失函数
- 优化算法：随机梯度下降法 SGD

  我们最开始选择的优化算法为 adam,但是迭代时间较长，并且最终的收敛效果并不是很好，所以我们将优化算法改为比较原始的随机梯度下降法 SGD。

  需要重点指出的是 SGD 中学习率（learn rate）这一参数的确定过程，学习率太小，更新速度慢,迭代周期长；学习率过大，可能跨过最优解。因此，在刚开始训练，距离最优解较远时我们采用稍大的学习率，随着迭代次数增加，在逼近最优解的过程中，逐渐减小学习率，但是为了避免收敛过于缓慢，减少的幅度又不宜过大。经过艰难反复的调参测试，最终确定的算法参数见表 2.1：

Learning rate	Weight decay	momentum	nesterov
0.01	1e-6	0.9	True

表 2.1 优化算法参数表

第三章 系统设计与实现

本系统分为两大部分，一是图像识别部分，二是公式计算部分。

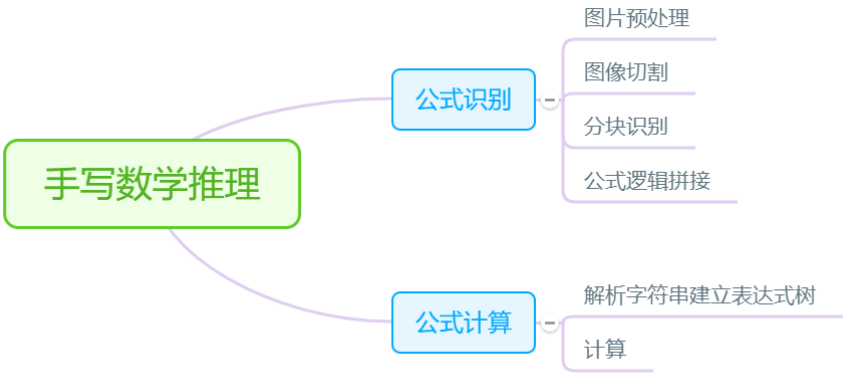


图 3.1 概要设计图

第一节 公式识别

图像识别部分我们首先对图像进行预处理，然后再进行切割和部分识别，最后根据逻辑进行公式拼接。

1. 图像预处理

选题要求识别数学公式照片，这就涉及到图像的预处理，照片会有模糊抖动、噪点等问题。为了更好地识别数学公式，必须对用户输入的图像进行预处理，预处理的结果直接对后续的字符识别效果产生影响。预处理主要包括二值化、降噪、图像倾斜校正和后期同切分一起做的大小和位置的归一化。

1.1 二值化

首先将图像灰度化，取消颜色的影响。

然后将图像二值化。常见的二值化方法为固定阀值和自适应阀值。

固定阀值就是制定一个固定的数值作为分界点，大于这个阀值的像素就设为 255，小于该阀值就设为 0，这种方法效果不一定好。因为即使是非常简单的图像也可能存在物体和背景的灰度变化,这种变化可能由于设备输入、光照不均或其它一些因素造成。

因此,使用变化的阈值(也称为自适应阈值化(adaptivethresholding))进行分割可以产生较好的效果。本系统采用自适应阀值，每次根据图像的灰度情况找合适的阀值。自适应阀

值的方法有很多，本系统采用了一种类似 K 均值的方法，就是先选择一个值作为阀值，统计大于这个阀值的所有像素的灰度平均值和小于这个阀值的所有像素的灰度平均值，再求这两个值的平均值作为新的阀值。重复上面的计算，直到每次更新阀值后，大于该阀值和小于该阀值的像素数目不变为止。

1.2 图像降噪

在二值化图像的基础上处理掉一些干扰因素，因为考虑到该照片为特意拍摄的公式照片，干扰因素并不会很多，而且二值化时已用了自适应阈值，降噪采用了快速的高斯滤波对图像进行降噪处理。

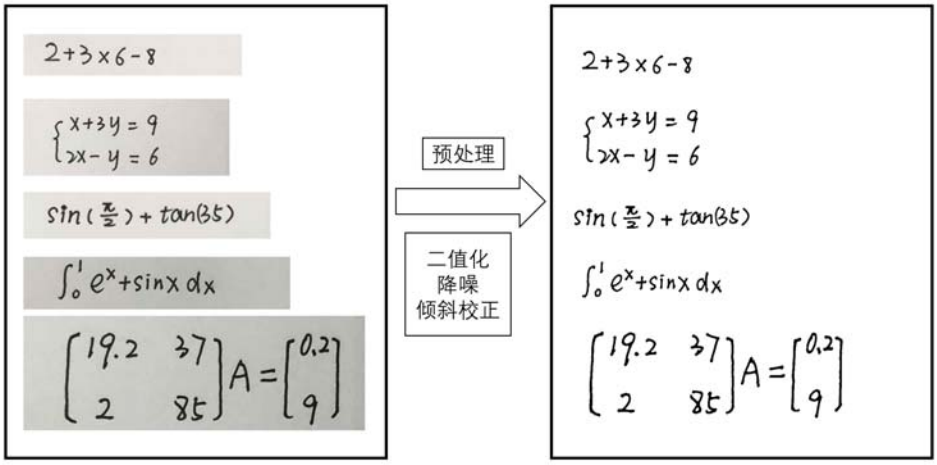


图 3.2 二值化处理结果

1.3 图像倾斜矫正

因为公式在书写拍摄过程中，不可避免的会产生倾斜，我们利用边缘检测，对图像中水平线进行强化处理，然后计算图像的 Radon 变换，获取倾斜角度，根据倾斜角度，对图像进行倾斜校正。

2 图像切割、粗识别

字符分割的目标是将单个符号的图像提取出来,送入识别器进行识别,得到相应符号的标签。字符切割方面：由于数学符号具有二维结构，是非线性的，各个字符间具有多种复杂的空间排布，如上下标、分式、根号等，而且数学公式是由多个字符组合到一起的，具有严密且复杂的逻辑关系,这就使得常用的图像切割运用到数学公式上精度较低。字符识别方面：公式各符号位置和大小均可能不同，而且手写公式可能存在连笔、断笔等现象，所以也对识别造成了很大的困难。

不论是切割还是识别，单独进行都会很大程度上影响识别精度，所以我们对系统进行了

改进，将切割和识别同步进行，动态修正，反复切割识别，一定程度上提高了识别率。

2.1 字符分割概述

我们总结了几种传统的字符切割方法，并进行了组合改进，系统主要采用投影法和动态比较字符骨架反复切分，并结合了连通体分析，以增加切分的正确率。

首先我们将公式定义了 10 种基本公式类型（在下文公式拼接部分详细介绍），第一遍投影时，先将多行表达式、根式表达式、分式表达式、矩阵和特殊符号进行一次筛选。考虑到实际需要，我们先做水平投影，再做垂直投影。

在此基础上，我们进行连通体分析、归一化处理。我们采用的是 Seed Filling（种子填充法）进行连通体分析，像素相邻关系采用的 8-领域图，进一步切割出候选切分字符。由于数学符号具有二维结构，各符号位置和大小均可能不同，为了方便符号识别，对其进行归一化，即将各个符号均缩放为同一大小和同一位置。

切割出候选切分字符后，我们采用动态比较字符骨架方法，先将初步切割好的字符与训练好的字符模型库进行匹配，反复利用投影法进行过度切分，防止有字符连笔无法识别，或者断笔无法识别现象，反复切分时记录其位置等属性信息。

进行反复切割后的字符块一般比较完整，我们再采用自训练的 CNN 神经网络模型进行再次识别确认，如果置信度较高，则采纳结果；如果低于一定置信度，则说明该字符可能还存在粘连等现象，此时将会对其进行过分割，对字符粘连和多结构字符进行处理。

按照逻辑筛选识别好的字符，根据属性信息，进行修改，最终留下识别好的字符。

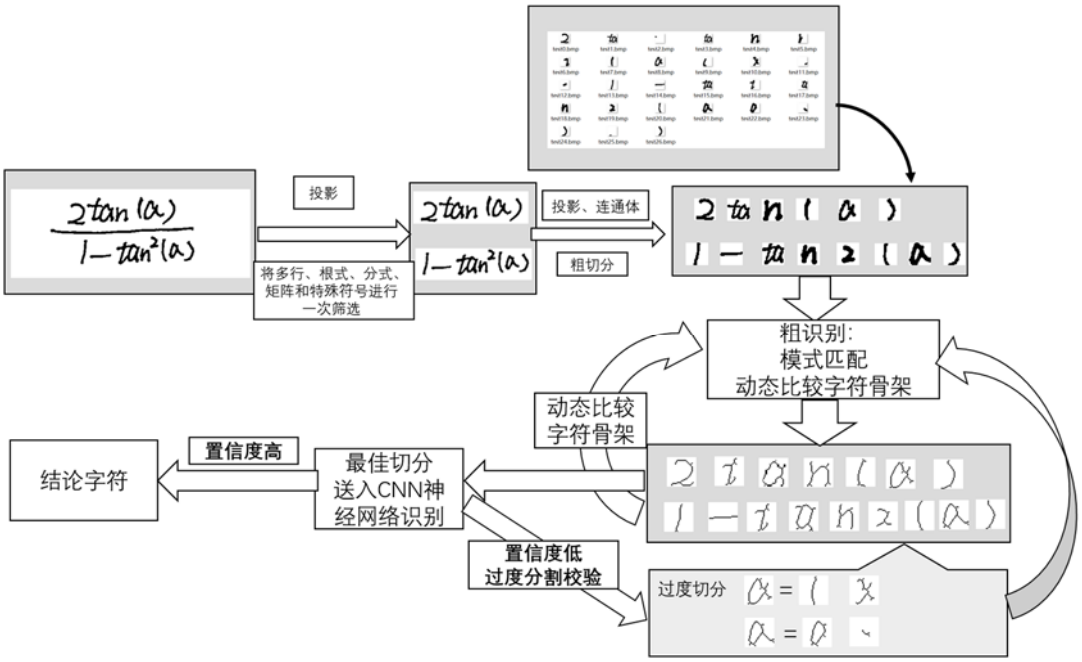


图 3.3 系统识别示意图

2.2 投影分割候选区域

在水平方向分割候选区域中,首先使用动态规划方法搜索可能的分割路径,然后在带有拒识别标签的候选区域上，根据识别模型提供的符号识别可信度，使用动态规划方法合并路径间的图像并形成水平方向的分割结果， 同理再次使用动态规划从竖直方向上多候选分割候选区域。结果如下所示:

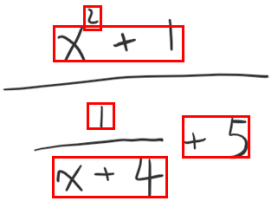


图 3.4 水平方向公式分割结果

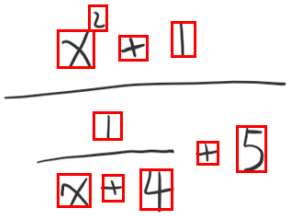


图 3.5 最终公式分割结果

2.3 模式匹配

按上述方法需要识别模型提供符号的可信度。所以我们制作了一个模板库，对字符进行特征提取，在识别时同样对切分好的图像进行特征提取，然后同模板进行匹配，若粗识别匹配不上，则进行动态修改，重新进行上述投影切割步骤进行切分。

模式匹配部分我们采用了统计特征与结构特征相结合,先粗识别再验证的方法。粗识别时利用的是统计特征,候选验证利用的是结构特征。在粗识别阶段,计算待识别符号的特征与所有标准符号的特征之间的距离,选择若干距离最小的符号作为候选识别结果。在验证阶段,使用反映候选字之间差异的特征点序列去匹配待识别符号,如果候选字具有的失配点足够小,则认为这是最佳切割结果,如果是打印体字符,匹配到这里一般可以达到比较高的争取率了,但是手写体字符因为各人书写习惯不同，所以识别率还不是很高，我们将切割好的最佳切分结果再次传入 CNN 进行再识别。

其中我们提取的特征信息有：

笔画特征：我们将二值化后的图像中，所处九宫格所有值之和小于等于 1 的小黑点，视为一个笔画结束。以此为依据，统计字符的笔画数。

字符整体特征:包括宽高比率,即每个字符所在包围盒宽度 w 和高度 h 的比率 w / h ；宽高与面积比率，即每个字符所在包围盒宽度 w 或高度 h 与包围盒面积 a 平方根 a 的比率；宽高与面积比率，即每个字符所在包围盒宽度 w 或高度 h 与包围盒面积 a 平方根 a 的比率。

穿线数：从不同方向、不同位置对已归一化的数学符号图像进行扫描，计算扫描线与笔

画相交的次数，就得到了数学符号在相应方向和相应位置上的穿线数特征。在本项目中，分别从水平９０°和垂直９０°方向上提取３个特征，分别为水平１／４穿线数、水平１／２穿线数、水平３／４穿线数、垂直１／４穿线数、垂直１／２穿线数和垂直 3／４穿线数。

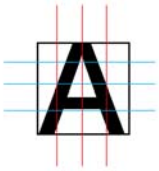


图 3.6 穿线数示例

本程序将字符的笔画信息、宽高比、 质心、端点、穿线数等诸多特征结合起来，得到了一种相对完善的特征模板库。

2.4 过度分割

如果 CNN 返回的可信度很低，我们将对该候选区域进行过度分割，来查看其是否包含一些类似点、减号之类细小不易察觉的符号，或存在严重的连笔现象。在这个模块中,我们采用过分割的方法,即得到尽可能多的字符块。然后进行两步后处理:一步用于解决字符粘连问题,对给定字符集中无法识别的字符块进行再分割,选择投影图的波谷作为分割点,由于这些分割点只有一到两个分割点是真正的分割点,这样就形成多种分割方案,去除假的分割点采用可信度来解决,即对投影图的可能分割都进行识别得到识别结果,每一种识别结果组合对应一种可信度,按可信度高低排序,对相同可信度的分割方案,按分割点数降序排列,选择可信度高的分割方案作为最后的分割结果。另外一步是对多结构字符的合并,所采用的识别字典中的多结构字符有:i、j、=、恒等号、除号、约等号、相似、包含、小于等于等符号,常用的多结构字符、多字符符号示例如图 3.7 所示。如果它们被分割开来,先找到字符中的主要结构,然后在其位置附近找另一结构,再合并字符块,并修改识别结果和边框信息。

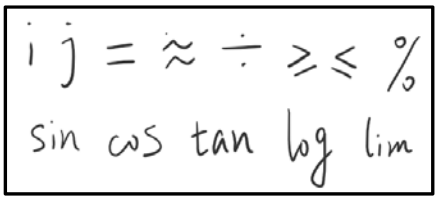


图 3.7 多结构、多字符符号示例

3 CNN 单字符识别

经过上述动态分割步骤，我们已经得到了单个字符的矩阵表达形式，现在需要将单个字符的图像转化为字符，作为后续公式逻辑拼接的原料。我们尝试过各种可能的识别方法：传

统的KNN监督分类方法、Tesseract-OCR光学字符识别等，经过我们不断地试验、训练以及精度对比，发现以上非机器学习的识别方法都存在“顾此失彼”的问题，即一旦加强某一个字符的训练强度，那么不仅该字符的识别率会提高，而且很多相似的其他字符也会被识别成该字符，从而导致整体的识别精度降低，基于上述原因最终我们选择了卷积神经网络（CNN）来进行单字符识别。

3.1 CNN模型简介

这一步我们采用的机器学习方法，本质是寻找一个函数，能将任意传入单字符图像映射到它表示的字符。而我们采用的深度学习方法通过组合大量图像数据的低层特征形成更加抽象的高层表示，最终形成一种深层非线性网络结构，从而实现复杂函数逼近，表征输入图像的分布式表示。

传统的全连接网络在处理图像这种高维数据时，会将图像的每个像素作为输入层，而且需要隐含层和输入层有相同的神经单元，这样不仅使计算量变得难以接受，而且在将图像转为一维向量的过程中会严重破坏图形结构，造成特征丢失。而与此不同的是，卷积神经网络通过对图像进行卷积运算，不仅能压缩图像的大小、去除冗余信息，而且能够提取出图像在空间上的结构化特征，最大程度上保存图像的原始信息。

CNN模型通常由输入层、卷积层、池化层、激活层和全连接层组成，下面将在模型搭建中着重介绍我们的CNN模型对以上各层参数的设置。

3.2 模型搭建

模型的搭建分三个主要阶段：

第一阶段是输入数据。数据作为N-d像素阵列提供。在这一步，我们将32x32像素原始图像转换为灰度图像。

第二阶段为构建卷积网络体系结构，我们使用Keras的Sequential类来构建网络。网络由3个卷积层、3个池化层、2个全连接层组成，每个卷积层和第一个全连接层后面都连接了一个ReLU激活函数，最后一层的全连接层通过softmax输出结果。

a）第一个卷积层：在第一层，我们采用32个卷积滤波器，在32x32大小的所有图像上作为大小为5x5的滑动窗口，并尝试获得具有最大强度值的像素。

b）ReLU功能：我们知道卷积是一种使用反向传播的方法。因此，在卷积层之后使用ReLU函数作为激活函数可以降低梯度消失的可能性并尽可能避免稀疏性。这样我们就不会丢失重要的数据，甚至可以摆脱像素中很多0的冗余数据。

c）Pooling 层：池化层从 ReLU 函数获取数据并对 3D 张量中的步骤进行下采样。简而言之，它汇集了从先前层获得的所有像素，并再次形成较小尺寸的新图像矩阵。这些图像再次输入到第二组图层中，即“CONV => ReLU => POOL”，并且该过程继续进行，直到我们得到一组最小的像素，我们可以依据其中的特征对字符进行分类。

第三部分为全连接层：完全连接层用于将每个先前层连接到下一层。该层由 512 个神经元组成。最后，我们应用 dropout 来减少过拟合，并使用 Softmax 分类器返回 73 个类标签中每个标签的概率列表。选择概率最大的类标签作为网络的最终分类结果并显示在输出中。收到的此输出用于制作模型的混淆矩阵。在此我们可以添加更多层，但添加更多层可能会影响系统的准确性。具体的结构见下图：

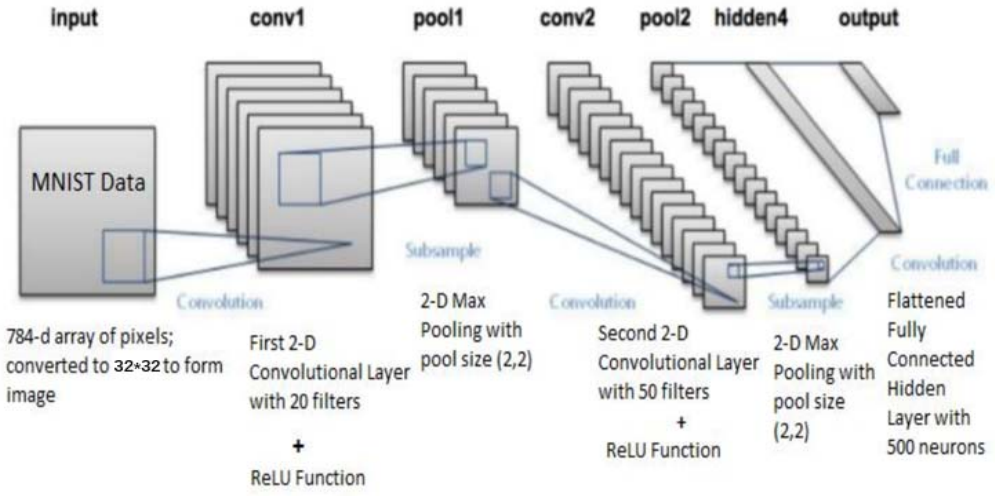


图 3.8 CNN 模型结构

各层参数设置如下表所示：

Layer	Parameter
Convolution	filters=32,kernel_size=(5,5),strides=(1,1)
MaxPooling	pool_size=(2,2),strides=(2,2)
Convolution	filters=64,kernel_size=(5,5),strides=(1,1)
MaxPooling	pool_size=(2,2),strides=(2,2)
Convolution	filters=100,kernel_size=(5,5),strides=(1,1)
MaxPooling	pool_size=(2,2),strides=(2,2)
Dense	units=100,activation=ReLU
Dense	units=73,activayion=softmax

表 3.1 神经网络参数设置

3.3 模型训练

准备好训练数据并搭建好上述模型后就可以进行模型训练了，训练需要确定以下几个关键部分：

- 批量和迭代次数

  batch=128:由于训练数据集较大，但是特征分布相对比较集中，所以我们选择相对较大的 batch，经过多次比对实验结果最终确定单次学习 128 张图像。

  epoch=20:迭代次数的确定也经过了反复测试，权衡计算量与准确率，最终得到的训练数据准确率随迭代次数变化见下图：

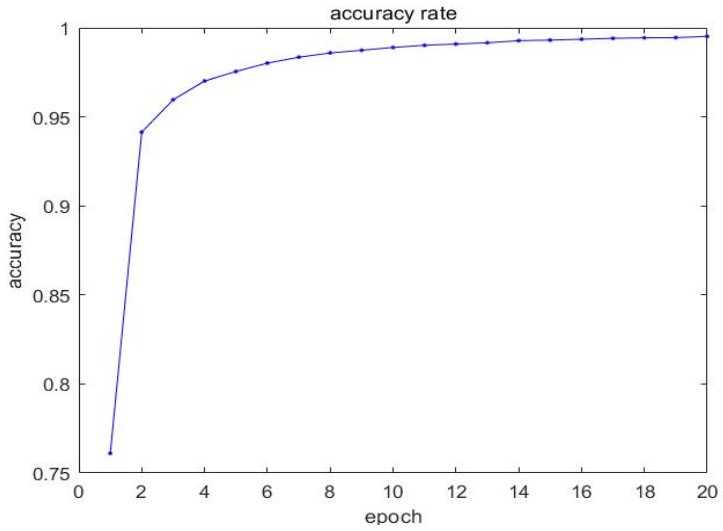


图 3.9 识别率变化曲线

- 损失函数：交叉熵损失函数

  这里我们基于这样的假设：训练数据总是从总体中独立同分布采样，那么就可以利用最小化训练数据的经验误差来降低模型的泛化误差。同时为了减少损失的计算量，最终确定使用交叉熵损失函数。

- 优化算法：随机梯度下降法 SGD

  我们最开始选择的优化算法为 adam 算法,但是迭代时间较长，并且最终的收敛效果并不是很好，所以我们将优化算法改为比较原始的随机梯度下降法 SGD。

  需要重点指出的是 SGD 中学习率（learn rate）这一参数的确定过程，学习率太小，更新速度慢,迭代周期长；学习率过大，可能跨过最优解。因此，在刚开始训练，距离最优解较远时我们采用稍大的学习率，随着迭代次数增加，在逼近最优解的过程中，逐渐减小学习率，但是为了避免收敛过于缓慢，减少的幅度又不宜过大。经过艰难反复的调参测试，最终确定的算法参数见下表：

Learning rate	Weight decay	momentum	nesterov
0.01	1e-6	0.9	True

表 3.2 优化算法参数表

3.4 测试结果

模型训练好后用测试数据中的 6393 张图像进行测试，准确率高达 99.69%，超过训练数据迭代 20 次的准确率 99.52%。

我们针对每次测试结果的差异部分进行尽可能细致的分析，并做出相应的调整数据集方案。具体分析时我们将本次测试结果与上一次的进行比对，着重分析调整方案中涉及到的字符，并根据神经网络的反馈信息再次调整直到达到我们预期的效果，下图为我们的分析员利用 excel 对比不同的测试结果以及所有测试数据和训练数据文件夹：

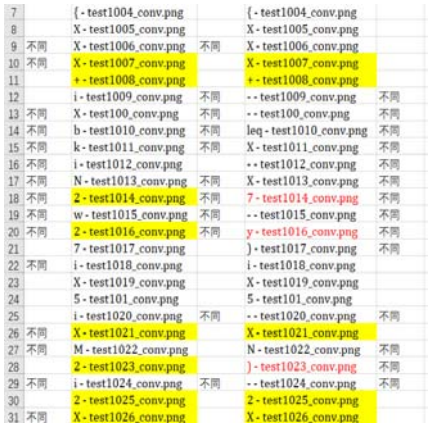


图 3.10 对比测试结果过程示例

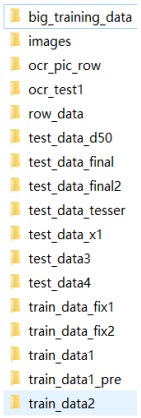


图 3.11 训练数据文件夹

下表列出了我们调整训练集中涉及到的一些典型易混字符：

	‘5’ ‘t’ ‘s’		‘\times’ ‘x’
	‘2’ ‘z’ ‘x’		‘\pi’ ‘x’
	‘8’ ‘z’		‘(’ ‘c’
	‘y’ ‘4’		‘{’ ‘i’ ‘1’

表 3.3 易混淆字符

3 公式逻辑拼接

结构分析部分是手写公式识别中的核心，也是其区别于其他种类的字符识别的标志。只有准确把握了公式中各个字符之间的结构关系，才能得到正确的公式表达式。数学公式的逻辑拼接也是公式识别的一大难点，通过分析切割后小图像的字符属性，按照逻辑，再对识别处理的图像子块进行拼接组合。

例如公式$\int_0^1 \sqrt{x}\,dx$最终输出形如：integrate("sqrt(x)","x","0","1")的字符串。

4.1 字符属性

在切分字符时，我们便提取出如下字符属性。

(1)字符标签 Label:字符标签唯一地标志字符表中的一个数学符号;

(2)空间关系 SR(SpaceRelation):字符之间的空间关系反应数学公式的结构层次,SR 的确定是数学公式结构分析过程中的重要部分。

字符空间位置关系的判定通常有两种方法:基准线法和“井”字形方法。基准线是指公式中的数学符号中心沿水平方向所在的一条直线，一个公式中包括主基准线和嵌套基准线，主基准线就是在字符中心连线所形成的一条类似水平直线;嵌套基准线是主基准线上下方平行的类似水平直线。基准线法对于文档中正规的数学公式能够很好的判断字符的位置关系，但对于手写的数学公式识别效果不佳。“井”字形方法对于大部分字符都能够进行空间位置关系的判断，而对于特殊数学字符，如根号“$\sqrt{\ }$”，括号(“()”，“[]”，“{}”)等，不能满足空间位置关系的判断。本系统结合“井”字形方法并增加了两种位置关系，共计为 10 种位置关系:上，下，左，右，左上，左下，右上，右下，包含，被包含。这 10 种位置关系为两两对称结构，这种对称使得字符的空间位置关系更为精确。

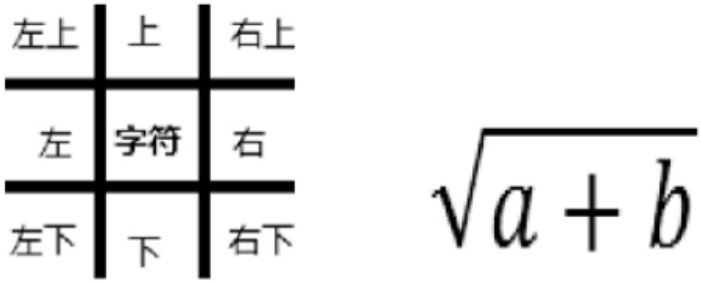


图 3.12 井字形位置关系和包含关系

字符	支配关系
二元操作符/开括号	右水平关系
闭括号	右上关系
根号	包含关系
数字	右上关系
分数线符号	正下/正上关系
可变符号	左上/正上/右上/左下/正下/右下关系
一些函数（如 lim）	右上/左下/正下/右下关系
其余字符	除左水平/右水平的所有关系

表 3.4 字符支配关系

(3)字符级别 Level:用于表示字符的层次,位于表达式中心线上的主字符的级别定义为 1;空间位置上直接附属于主字符的其他字符的级别为 2;依此类推。

(4)分支序号 BranchNumber:用于区分从属于同一个字符与之满足不同空间关系的子字符组。

(5)模块类型标志 MT(ModuleType):用于区分不同类型的子模块,不同模块采用一个大写字母进行区分,如普通型 C(Common)、分数型 F(Fraction)、根式型 R(Radical)、矩阵型 M(Matrix)、帽子型 H(Hat)和重音型 A(Accent)。

(6)块内位置标志 BP(BlockPosition):BP 用来表示字符在特殊模块内部的位置信息。其取值规定为 3 种,分别是 N(Normal)、U(Up)、D(Down)。BP=N,表示字符不在特殊模块内部;BP=U,表示字符位于某特殊模块上部;BP=D,表示字符位于某特殊模块下部。说明:本文所基于的在线手写公式识别课题中没有涉及矩阵结构的表达式处理,所以相关属性的设定并未考虑矩阵情况。然而,可以通过适当调整字符属性数目以及属性取值情况来实现扩充。

(7)下一字符指针 Next:该属性用于在存储树形结构时对字符节点数据按输入顺序进行链接。

4.2 公式分析模型

根据人书写公式的习惯,并参考 LATEX 语言描述公式的方法,定义了 10 种基本公式类型,即多行、分式、根式、定界、矩阵、组、堆叠、角标、普通单行和基元表达式。

多行表达式 （Multiline Expression）	$x+y=3$ $x-y=0$	分式表达式 （Fraction Expression）	$\frac{2}{3}\times\frac{7}{6}$
根式表达式 （Radical Expression）	$\sqrt{x+y-5}$	矩阵表达式 （Matrix Expression）	$\begin{pmatrix}4 & 5\\ 8 & 9\end{pmatrix}$
定界表达式 （Delimiter Expression）	$(2+3)\times 7$	组表达式 （Group Expression）	$\sum_{i=1}^{n}$
普通表达式 （Common Expression）	$48+52-38$	角标表达式 （Script Expression）	x^{y^2}
堆叠表达式 （Stack Expression）	$\lim_{x\to 0}$	基元表达式 （Primary Expression）	x

表 3.5 各类公式分析

（1）复合表达式定位

11 种公式类型中,有的类型的结构非常固定,只要知道其中一个或两个关键符号的位置,那么整个表达式的范围和结构也就确定了。这种类型的表达式称作复合表达式,其中的关键符号规定为 L1 级符号。定义如下四类公式是复合表达式:根式表达式,关键符号为根号;分式表达式,关键符号为分数线;定界表达式和矩阵表达式,关键符号为左右定界符号。图 3.13 是公式中复合表达式的定位结果。

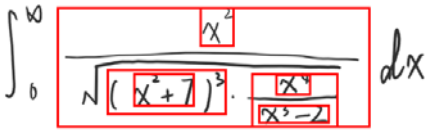


图 3.13 复合表达式定位

（2）多行表达式分解

多行表达式可以通过水平投影分解成多个单行表达式。但是并不是投影得到的每个部分都是一个完整的单行表达式。所以还需要根据符号识别结果、竖直空白间距、上下部分符号位置关系,删除虚假的单行表达式。

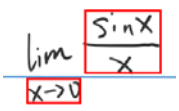


图 3.14 虚假多行表达式

（3）普通单行表达式分解

普通单行表达式分解的原则就是沿着水平方向分解成多个包括骨干符号的单行子表达式。如图 3.15 所示,根据所有符号的中心位置可以估计出公式骨干线的位置。字符中心和骨干线距离接近的符号被称作骨干符号。在分解普通单行表达式时,为每个骨干符号生成一个初始子表达式,然后根据距离把非骨干符号插入最近的子表达式。

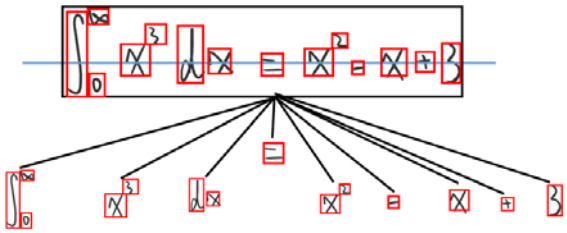


图 3.15 普通单行表达式分析

依照如上关系，最终建立公式分析的分解树，按照树形结构拼接字符，输出表达式。

第二节 公式计算

公式计算环节,我们实现了对各种各样的数学表达式进行计算。表达式的形式十分多样，包括常见的普通四则混合运算、矩阵运算，对方程或方程组求解，以及复杂的求积分和导数的解析解和数值解。

公式计算部分主要分为三个步骤：首先对读入的字符串进行预处理；然后借助二叉树的结构，从字符串中解析出表达式的计算顺序；最后中序遍历表达式树，进行计算。

1. 标记字符串属性

由于公式计算部分实现的功能多且复杂，在解构字符串之前，十分有必要对字符串先做一下预处理，降低字符串的复杂程度。预处理的步骤可以分为以下几个方面：

1）以换行符作为分割标志，如果识别出来多个表达式，先将表达式分离到一个列表中再计算。

2）初步解构表达式

对表达式中的每个字符增加符号属性(nLastPushedCellType)。经过比对，以 nLastPushedCellType 值的不同，对数字字符、运算符、变量或函数名分别做标注。

同时去掉空字符，并忽略字符串中，逗号及其后面的子串。

3） 化简

对于方程，通过移项、合并同类项等方式，将其转变为多项式形式，或者其他更便于求解的形式。对于函数，同样应先将其转变为容易计算的形式，再做结构分析。

2. 基于二叉树的公式结构分析

2.1 算法概述

按照赛事要求，我们采用的是基于分块树的数学公式结构分析方法。

该方法首先根据数学公式内部的不同结构特点，将数学表达式分为若干种不同子模块,如普通模块、分式模块、根式模块、矩阵模块等。除普通模块外,其他模块统称为特殊模块。每个模块包含有若干字符,模块的划分依据为各模块对应的标志符号,如根式模块的标志符号为“根号”。此外,在每一个子模块内部反复进行划分,直至模块不可再分,得到最小子模块。分块处理方式可以根据不同的应用需要,通过增加或删减模块类型对公式识别系统进行扩展或简化。

对于每一个划分的最小子模块,将其解析成树型结构进行表示,其中每一个树节点代表一个字符;再以每一个树型结构作为上一级划分的节点,构成新一级别的树,直至最终完成对表达式的解析。

2.2 字符处理程序描述

- 普通字符的处理

读取字符的信息，包括 Label、质心坐标以及坐标边框信息。

首先要判断是否为首字符，这里首字符是指公式骨架位置上的具有最高级别的主字符。 其属性默认为:Level = 1; Branch Number=0;SR=P;MT=C;BP =N; Next 指向下一个存储单元;

对于非首字符,首先判断其与前一字符的关系: 如果 SR 值为 Parallel，则除 Next 每次指向新的存储单元外，新字符的其它属性值与前一字符相同。如果 SR 值为上标关系或下标关系，表示该字符从属于前一字符，那么 Level 值加 1;上标时 BranchNumber = 1 ，下标时 BranchNumber =2; 此外同样取 MT =C ;BP=N。

如果 SR 值不满足以上 3 种类型，则表示新字符不从属于前一字符，那么继续向前搜索，直到找到满足从属关系的情况，按前述规则对其属性进行赋值。

- 分块字符的处理

在遇到特殊模块标志字符时，即根号、分号等(通过 Label 值判断)，则进入分块处理模式。 所有横坐标位于标识符坐标区域内的其他字符都属于该模块。 对于每一个模块,

首先按照普通符号的处理方式对标志字符属性进行赋值，但 MT 值除外。 模块内部字符的 MT 值与标志字符相同，但在多级特殊模块嵌套情况下遵循局部优先原则，即字符属性根据其所在的最小子模块得到。其余属性与普通字符处理方式一致。

字符处理算法流程如下图示 。

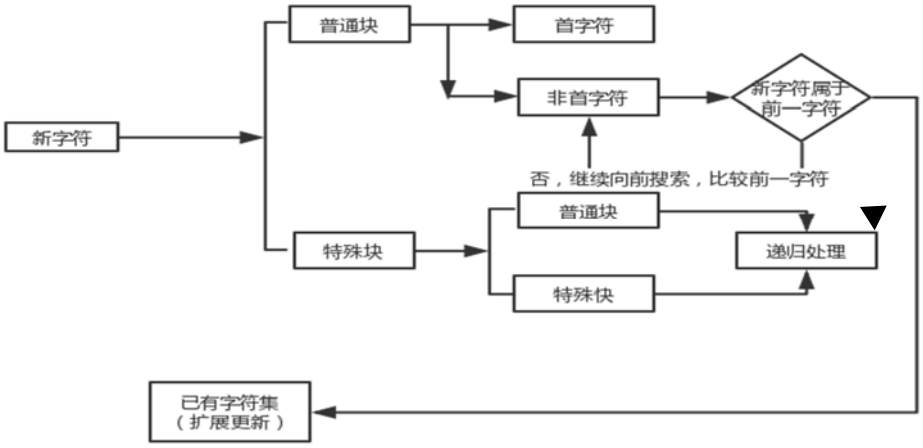


图 3.16 字符处理算法流程

2.3 构建表达式树

采用前面所讨论的结构分析算法对一个具体的数学表达式进行分析，构建二叉树。

由字符串形式的数学表达式，生成其二叉树表示的方法是先建立根结点，根结点的公式元素是字符串中优先级最低的公式元素，由于公式元素串中位于根结点公式元素之前的公式元素在根结点的左子树，位于根结点公式元素之后的公式元素在根结点的右子树，递归建立根结点的左右子树。

例如，“a*sin(pi*x/2)+b*cos(y)”和“diff(x+y^6)”，生成的二叉树如下图所示：

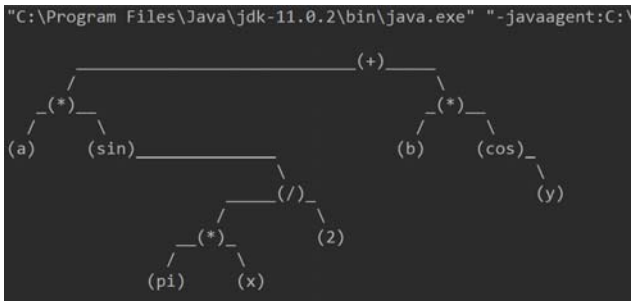


图 3.17 表达式树示例 1

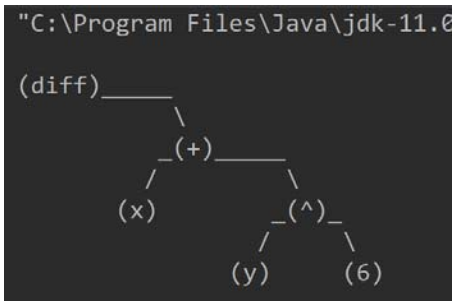


图 3.18 表达式树示例 2

3. 计算

中序遍历表达式树，将左右叶子节点进行父节点运算，动态更新节点的值，直到遍历完整棵树，输出结果。函数求导这一功能，我们还未十分完善，目前调用 Python 的 SymPy 库作为补充，其他计算均可由 Java 程序自行完成。

第四章 系统测试

第一节 测试过程

1. 对于单个字符

我们从总数据集中分出 10%作为神经网络的测试集，共计图像 6393 张，在服务器上进行识别测试。同时我们也用我们自行手写的公式，运行系统，输出切分好的图像测试，各类符号共 1210 张，同样在服务器上进行识别测试。

图 4.1 部分测试字符

2. 对于整式识别

我们每类式子准备了 10 个左右，邀请了 20 名同学，将每个式子抄写 1 遍，共得到测试数据图像 1998 张，在本机进行识别测试。

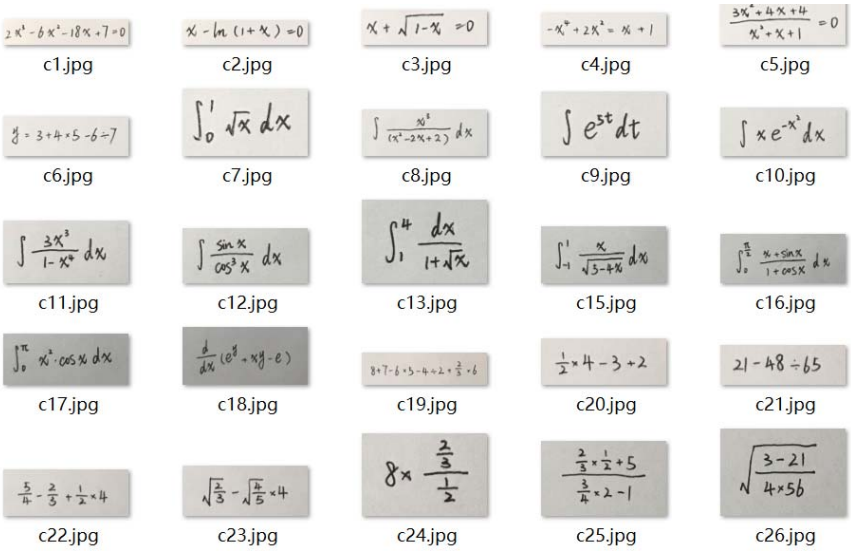


图 4.2 部分测试图像

第二节 测试结果

1. 对于单个字符

2. 我们从总数据集中分出 10%作为测试集，共计图片 6393 张，测试精度高达 99.64%。

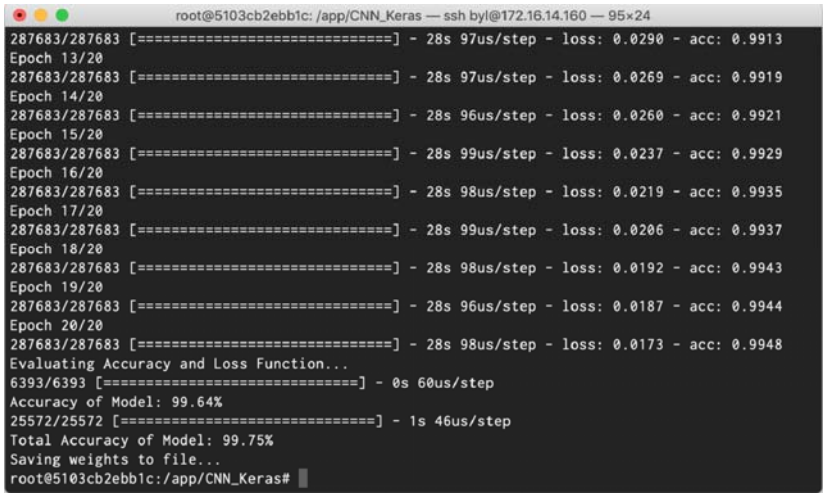


图 4.3 单个符号测试集识别结果

同时我们也用我们系统从整式中切分好的图像测试，各类符号共 1210 张，识别率达到 96.73%。

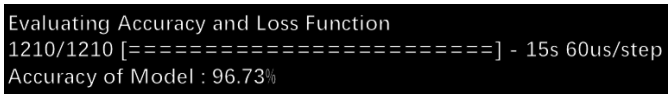


图 4.4 切分出的单个符号识别结果

3. 对于整式识别

类型	示例	题目数	样本/正确求解数	识别率
算术表达式	$5+3\times7\div2$	15	298/282	94.63%
分式表达式	$\frac{5}{4}+\frac{8}{17}$	13	260/245	94.23%
根式表达式	$\sqrt[3]{125}$	10	200/189	94.50%
乘方表达式	2^5-4^6	12	240/213	88.75%
三角函数表达式	$\sin(45)+\cos(60)$	10	200/171	85.50%
方程	$x^3+x^2+4x+3=1$	9	180/155	86.11%
方程组	$\begin{cases} x+y=9 \\ x-y=0 \end{cases}$	6	120/95	79.16%
积分	$\int_0^1 x\,dx$	7	140/115	82.14%
微分求导	$\frac{d}{dx}x^2+5$	4	80/58	72.50%
极限	$\lim_{x\to0}\frac{\sin x}{x}$	4	80/51	63.75
求和	$\sum_{k=1}^{n}\frac{k}{k^2+1}$	4	80/59	73.75%
矩阵	$\begin{bmatrix}4&5&3\\8&9&6\end{bmatrix}A=\begin{bmatrix}2\\7\end{bmatrix}$	6	120/103	85.83%
总计		100	1998/1736	82.38%

表 4.1 系统能处理的表达式

第三节 结果分析

1. 运行时间分析

由于采用了特征识别和 CNN 两种识别方法提高准确度，也增加了识别需要的时间，平均识别时间为 1.3s。

2. 优点

（1）系统能够识别的字符多，题型也比较丰富。

（2）使用卷积神经网络模型和特征识别共同识别字符，精度高，适应性强。

3. 不足

（1）测试的 100 道计算题均为团队成员设计，难度并不是很大。

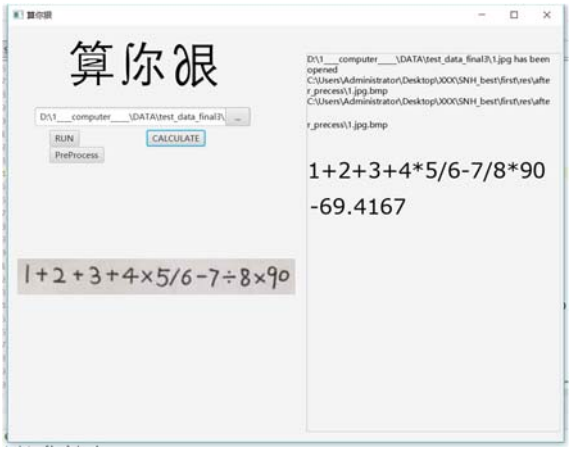
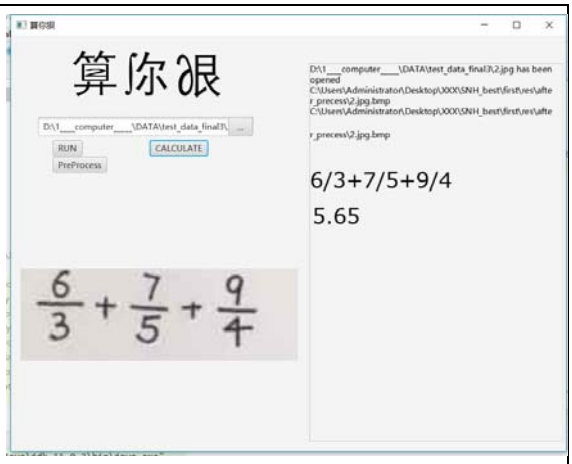
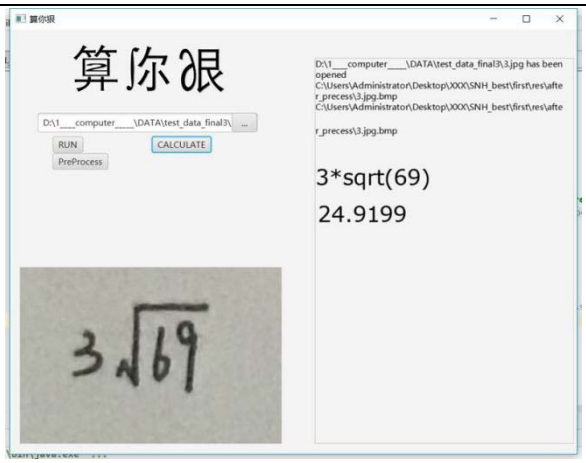
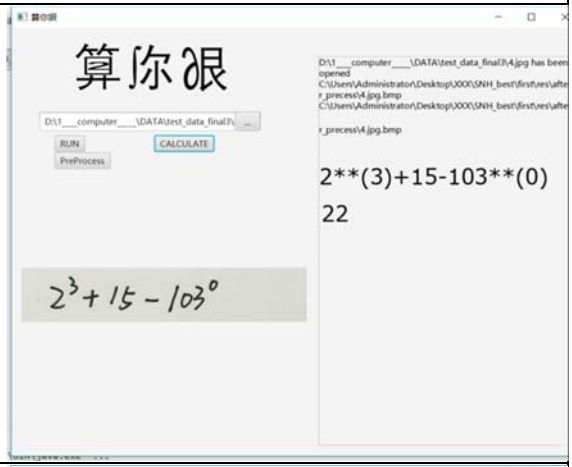
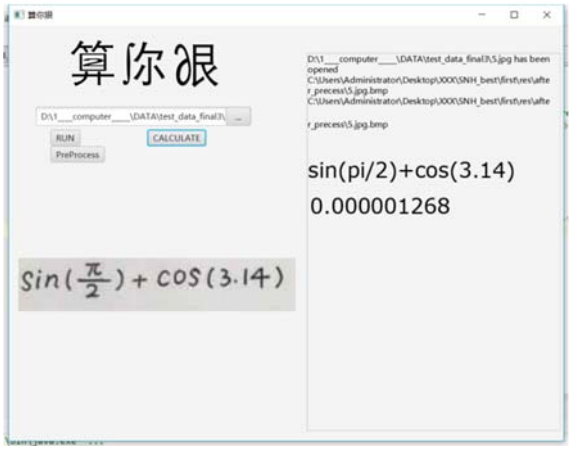
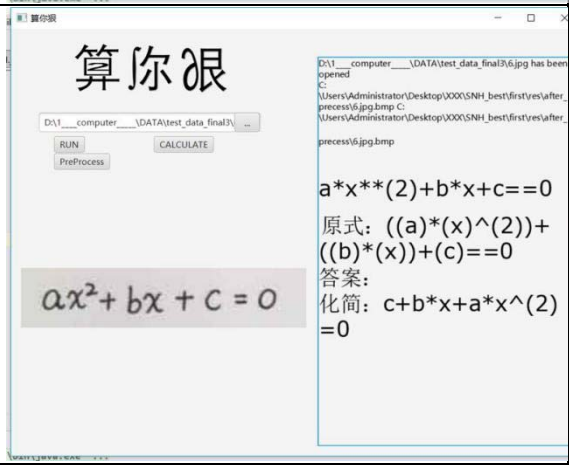
（2）系统识别相对结构复杂的数学计算题精确度比较低。

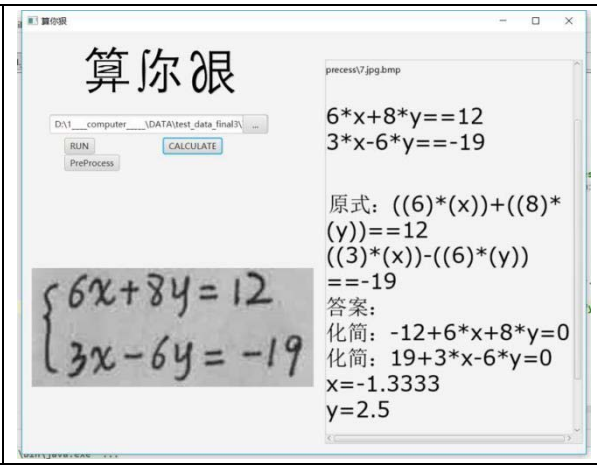
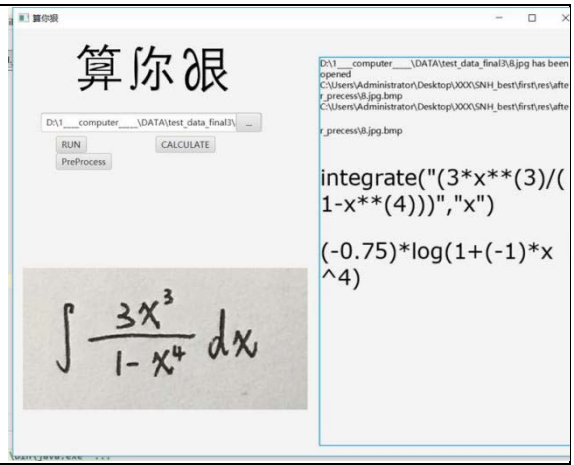
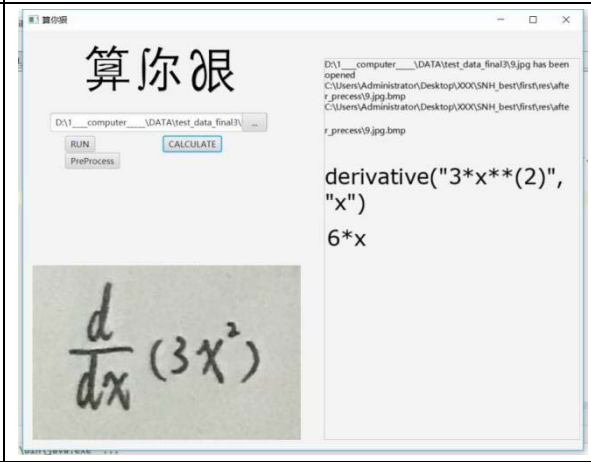
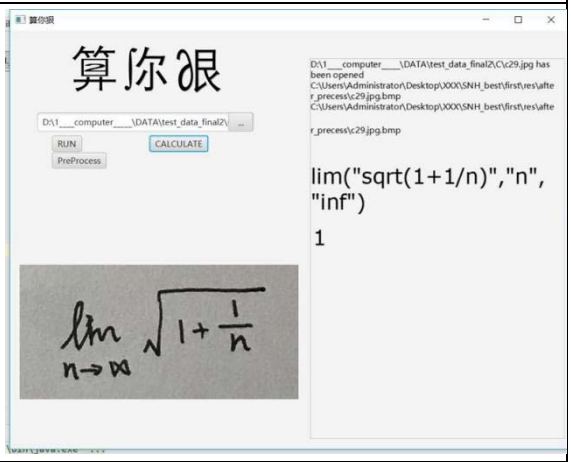
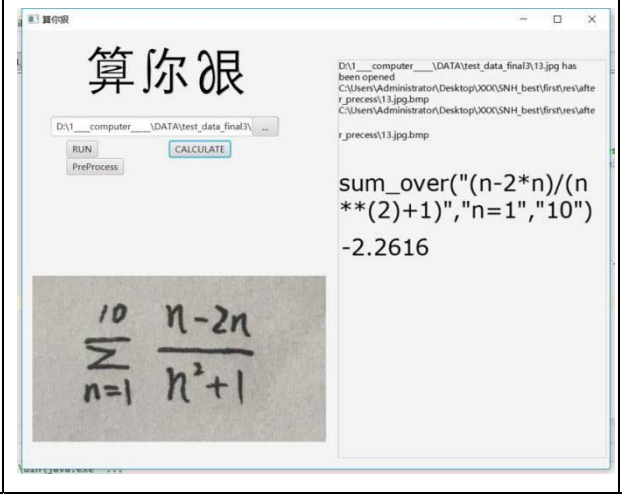
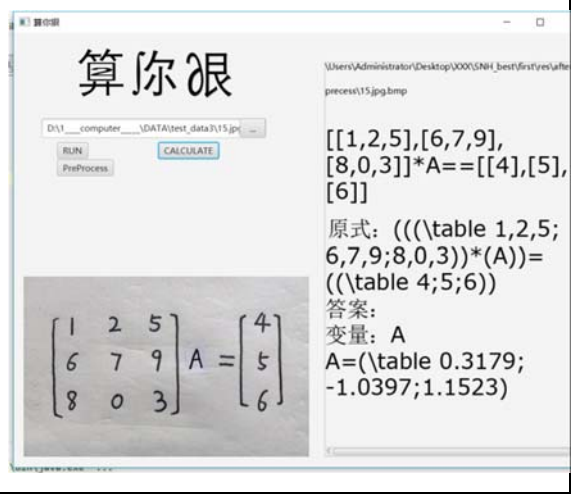
（3）纠错系统逻辑还不够完善，前面环节的错误会导致后面环节的错误。

4. 后续工作展望

不足与展望：目前本模型对简单式子的识别已经能达到很高的精度，不足之处在于对复杂的结构的式子识别率仍旧没能达到我们的期望，在后续的改进中我们希望可以进一步加强对连体字的识别以及对各种单词类（sin，cos 等）数学符号的识别。最终希望我们的模型能在整体上达到更高的精度。另外我们将进一步完善我们的展示界面，增加更多的功能使得我们的成果更加的完善。

第四节 部分测试截图

算术表达式		分式表达式	
根式表达式		乘方表达式	
三角函数表达式		方程	

方程组		积分	
微分		极限	
求和		矩阵运算	

第五章 安装及使用

第一节 安装

Java 环境要求：java9/10/11，javaFX 第三方库。

Python 环境要求：Python3, numpy, PIL, keras, skimage

第二节 使用

1、编译器运行

（1）在编译器中输入图像名称，识别并计算单张图像

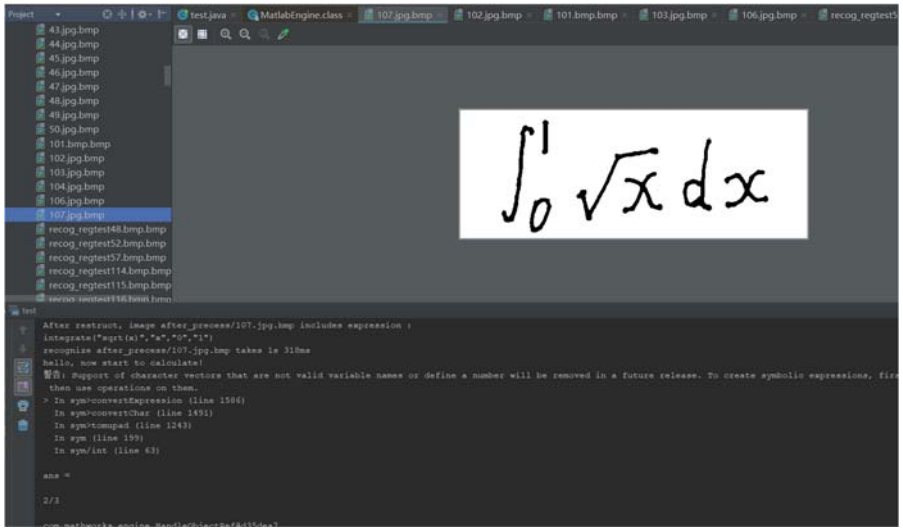


图 5.1 单张图片计算

（2）在编译器中输入文件夹名称，批处理识别多张图像

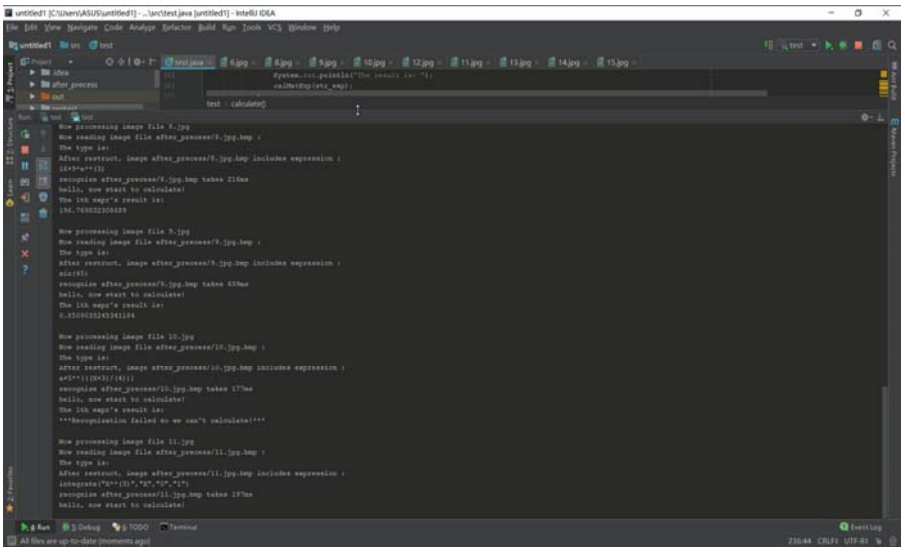


图 5.3 批处理多张图片

2、UI 界面操作

2.1 UI 界面介绍

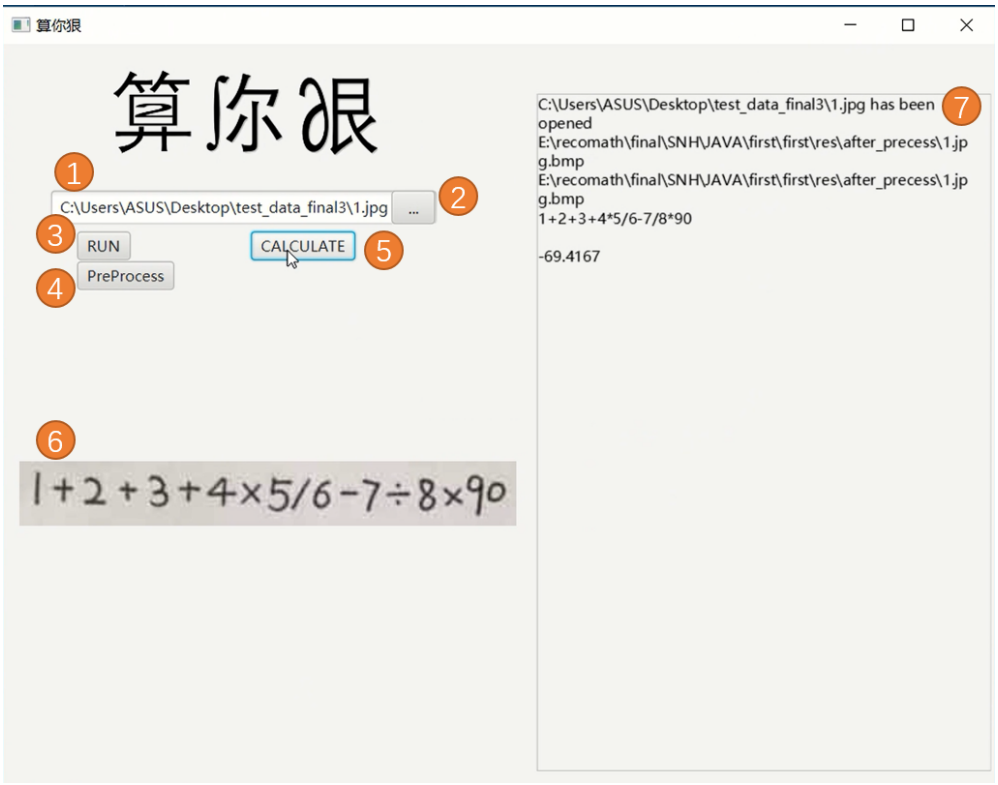


图 5.3 UI 界面

（1）此处文本框可显示选中的图像的路径及图像名

（2）点击此处“…”按钮可选择文件

（3）点击 RUN 按钮可以运行识别程序，识别结果输出在 7 处

（4）点击 PreProcess 按钮可以进行预处理，预处理后的图像输出在 7 处显示的文件夹中

（5）点击 CALCULATE 按钮可以进行计算，计算结果输出在 7 处

（6）此处可显示选中的图像

（7）此处为输出框，显示图像路径，预处理后图像路径，识别结果，计算结果

3.2 操作说明

（1） 首先点击“…”按钮选择文件

（2） 点击 PreProcess 按钮进行预处理

（3） 点击 RUN 按钮运行识别程序

（4） 点击 CALCULATE 按钮进行计算

第六章 项目总结

我们团队由四名大二学生和一名大三学生组成，分别来自计算机科学与技术系、智能科学与技术系、软件工程系，因为我们年纪较小，经验不足，技术知识不如学长们扎实稳健，在项目的推进过程中，我们举步维艰。自 4 月 16 日建队确定选题“手写数学推理”以来，不到一个半月的时间，队员们每天从繁重的课业之中争分夺秒，披星挂月，努力完成项目。网上查资料，课下请教老师学长，东奔西走，为了项目，队员们学新语言、配置环境、了解编译环境，几乎从零入门，飞速成长。

时间紧、任务重、经验少，能做到现在这个成品，全凭队员们的一腔热血，激情奉献。从开始到项目即将结束，一步步走来，我作为其中的参与者，以及全部过程的见证者，对此项目进行一个概括性的总结。

1、项目难点

本选题看起来与生活密切相关，似乎有许多较为完善的解决方案，但经过我们的仔细检索，发现本项目虽然有一些成品软件，但是开源资料极其稀缺，开源资料有的是只能识别电脑上的印刷体、有的需要在平板或者数控板上实时手写，印刷体字形单一，结构整齐，方便识别，实时手写可以获得更多有效信息来识别字体，例如笔画起止，连贯读，书写顺序等，图像也不会有模糊噪点等问题，而本选题要求必须是手写照片，这一下子提升了项目难度。还有一个难点在于，数学公式不同于一般的 OCR 识别，数学公式是二维的，有层次与结构，有包含与被包含关系，例如矩阵的空间关系、分数的上下结构关系、根号、积分的包含关系，平方、角标的所属关系等，这些都对字符分割与拼接提出了极大的挑战。

2、任务分解

我们将项目主要分为两个模块，图像识别与解析计算。

- 图像识别

我们分为两组，一组人员以完成作品为目标，参考网上比较完善的特征提取算法，利用特征提取的方法来识别图中公式。另一组人员以提高精度为目标，研究网上已有的识别数字及字母的机器学习模型，搭建服务器，调试显卡，训练模型，测试数据，来提高识别精度。

- 解析计算

计算方面，我们首先按逻辑解构字符串，并建立表达式树的形式，并按照表达式树特征，

对表达式进行分类，分为例如：普通四则运算、积分、微分、矩阵、方程等类，并尽行细分，例如：积分中的求数值解或求解析解等，在树形基础上进行计算。

3、项目协调

在项目的一个半月时间里，大家都投入了极大的热情，队里大家相处融洽，项目组内部共同探讨解决问题的办法。每天在微信群里报告自己的工作进展，以及遇到的问题，交流自己对后续工作的看法。

参考文献

[1] 胡龙灿,杨帆,樊爱军.手写数学公式的识别研究及在 Android 上的应用[J].计算机应用与软件,2014,31(08):28-31+44.

[2] 李永华,王科俊,上官伟,唐立群.数学公式基线结构分析及识别算法研究[J].计算机工程与应用,2008(16):18-22.

[3] 徐旭明,洪留荣,张建成.一种改进的手写数学公式符号识别算法[J].淮北煤炭师范学院学报(自然科学版),2009,30(02):42-47.

[4] 洪留荣.在线手写数学公式结构分析算法[J].计算机应用,2010,30(09):2545-2548+2552.

[5] 郭育生,黄磊,刘昌平.基于多候选的数学公式识别系统[J].计算机研究与发展,2007(07):1144-1150.

[6] 卢晓卫,林嘉宇.一种基于分块树的手写数学公式结构分析算法[J].计算机工程与科学,2010,32(10):69-72+84.

[7] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [C]. In: International Conference on Machine Learning. Lille: ACM, 2015, PP. 448-456.

[8] Mouchere H, Viard-Gaudin C, Zanibbi R, et al. Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014) [C]. Frontiers in handwriting recognition (ICFHR), 2014 14th international conference on. IEEE, 2014: 791-796.

[9] 方定邦,冯桂,曹海燕,杨恒杰,韩雪,易银城. 基于多特征卷积神经网络手写公式符号识别[J].激光与光电子学进展,2019,56（07）：072001