**黑板手写数字识别与打分系统软件设计**

完成人：张秋月 沈晨栖 尹董艳

**1. 概述**

**1.1. 系统简介**

该系统基于Python的深度学习方法，构建和训练卷积神经网络后，将通过摄像头实时监测，对捕捉到的图片进行处理输入网络，最后输出预测的识别结果和分数。

**1.2. 目标读者**

项目评定人、尝试该方面开发的初学者、使用该系统进行教学的家长或老师。

**1.3. 书写约定**

ROI区域：即感兴趣区域，通过OpenCV的各种算子和函数从被测区域勾勒得到需要进行下一步处理的区域。 张量(tensor)：多维数组，用“阶数”表示张量的维度，分为0阶张量(标量)、1阶张量(向量)、2阶张量(矩阵)等等。

**2. 总体设计**

1. 总体设计 为了对图像内容进行识别，对抓取到的图像需要做一些预处理，去掉多余，增强突出重要区域，以得到合适的、可以输入网络的图像。此为图像预处理模块。 为实现识别功能，我们选择构造一个卷积神经网络，经训练、测试后可实现将预处理后的图片输入进去可以自动得到结果，此为卷积神经网络模块。

**2.1. 图像预处理模块**

从人机交互模块得到的图像需要先进行一些预处理，改变图像的大小和颜色、滤去干扰和噪声等等，得到网络可接收大小的ROI区域，再提供给卷积神经网络模块。

**2.2. 网络模块**

该模块首先需要构建一个残差网络，用FashionMnist数据集将其训练好得到权重，并测试网络功能。而后在系统运行时，把图像预处理模块提供的图像输入进网络输出被测结果和分数。

**3. 详细设计**

对模块，详细描述它所使用的数据结构和算法，可用教材中介绍的过程设计工具，如：程序流程图、PAD图、伪码等表示。\*

**2.1.人机交互模块**

该模块实现与用户的交互，通过摄像头实时抓取被测画面，提供给图像预处理模块，再向用户展现预测的数字和评定分数。

**3.1. 人机交互接口详细设计**

|  |  |  |
| --- | --- | --- |
| 函数名 | cv.VideoCapture | |
| 功能 | 打开摄像头 | |
| 参数说明 | int | 摄像头的设备ID |
| cv.CAP\_DSHOW | 微软设定的传递标志 |
| 返回值说明 | 捕捉对象，即cv2.VideoCapture类的对象 | |
| 使用方法 | 点击运行即开始监测被测区域 | |

|  |  |  |
| --- | --- | --- |
| 函数名 | capture.read | |
| 功能 | 截取一帧的图片 | |
| 参数说明 | 无 | |
| 返回值说明 | ret | 布尔值，标志是否截取到图片 |
| frame | 三维矩阵，即截取到一帧的图片 |
| 使用方法 | capture为cv2.VideoCapture类的对象，该方法用于运行实时循环截取 | |

|  |  |
| --- | --- |
| 函数名 | capture.release |
| 功能 | 析构函数，关闭视频监测 |
| 参数说明 | 无 |
| 返回值说明 | 无 |
| 使用方法 | capture为cv2.VideoCapture类的对象，系统停止运行随即关闭摄像头 |

|  |  |  |
| --- | --- | --- |
| 函数名 | cv.imshow | |
| 功能 | 展示窗口 | |
| 参数说明 | string | 窗口名 |
| img | 窗口中展示的图片 |
| 返回值说明 | 无 | |
| 使用方法 | 展现给用户的窗体 | |

|  |  |  |
| --- | --- | --- |
| 函数名 | cv.putText | |
| 功能 | 打印文字 | |
| 参数说明 | string | 窗口名 |
| string | 打印的文字 |
| Point坐标 | 文本框左下角坐标 |
| font | 字体 |
| int | 尺寸因子决定字体大小 |
| RGB值 | 字体颜色 |
| int | 线条宽度 |
| 返回值说明 | 无 | |
| 使用方法 | 用于输出结果（文字信息） | |

**3.2. 读取解析数据集接口详细设计**

|  |  |
| --- | --- |
| 函数名 | decode\_idx3\_ubyte |
| 功能 | 解析数据集读取图片 |
| 参数说明 | idx3\_ubyte\_file 测试集文件路径 |
| 返回值说明 | images 读取到的图片 |
| 使用方法 | 训练网络前对数据集图片进行解析 |

|  |  |
| --- | --- |
| 函数名 | decode\_idx1\_ubyte |
| 功能 | 解析数据集读取标签 |
| 参数说明 | idx1\_ubyte\_file 测试集标签文件路径 |
| 返回值说明 | lables 读取到的标签 |
| 使用方法 | 训练网络前对数据集标签进行解析 |

**3.3. 图像预处理模块**

3.3. 图像预处理模块

**3.3.1 颜色处理**

1. 灰度化： cv.cvtColor(img, cv.COLOR\_RGB2GRAY)函数将截取到的图像进行灰度化，得到一个单通道的灰度图像img\_gray。
2. 二值化 通过cv.threshold(img\_gray\_resize, 200, 255, cv.THRESH\_BINARY)函数进行二值化，其中200为当前灰度阈值thresh，255为最大灰度阈值maxVal，根据\_cv.THRESH\_BINARY\_参数设定，该图像中的像素点灰度大于当前灰度阈值的，将被设为最大灰度阈值，其余为0。

**3.3.2 滤波去噪**

1. 膨胀： kernel = np.ones((3, 3), np.uint8) img\_open = cv.dilate(img\_bw, kernel, iterations=2) 建立3x3的核结构kernel，对要处理的图像img\_bw进行次数iteration为2的膨胀，即将各点像素值替换为对应邻域上的最大值。
2. 剔除小连通域： num\_labels, labels, stats, centroids = cv.connectedComponentsWithStats(img\_open, connectivity=8, ltype=None) 输入图像是8位单通道图像img\_open，connectivity为8，即选择8连通：上下左右+左上、右上、右下、左下。num\_labels是连通区域的数量；labels是一个与输入图像一样大小的矩形（labels.shape = img\_open.shape），其中每一个连通区域会有一个唯一标识，标识从0开始；stats包含5个参数分别为x,y,h,w,s，分别对应每一个连通区域的外接矩形的起始坐标x、y，外接矩形的wide、height，s是labels对应的连通区域的像素个数；centroids是连通区域的质心。 再由cv.rectangle(img\_open, tuple(sta[0:2]), tuple(sta[0:2] + sta[2:4]), (0, 0, 255), thickness=-1)\_对小连通域用画边框的方式将其填为黑色。
3. 提取ROI 被截取图像经上述处理，粉笔字区域为白色（灰度为255），其余黑板背景为黑色（灰度为0），要识别粉笔数字，需提取出该区域。分别对行列两方向进行向量相加求和，通过遍历判断某行/列的向量和是否大于阈值（即是否存在粉笔字区域），以找到粉笔字区域的近似边缘进行截图，并添加边框，即为ROI区域。

**3.3.3 大小调整**

大小调整共有两次，用cv.resize(img, (m, n))完成，img为待处理图像，m和n为目标大小的长宽。第一次大小调整将摄像头截取的图像设置为600px\*600px便于处理，第二次大小调整将处理后的结果图像设置为网络需要的28px\*28px大小。

**3.4. 卷积神经网络模块详细设计**

本系统采用Resnet残差网络，相较于传统网络，残差网络更易于捕捉模型细微波动，具有更快的收敛速度。残差网络是一种深层次卷积神经网络。对于卷积神经网络而言，加深网络层数可以增强其拟合能力。但随着网络层数的加深，卷积神经网络会变得非常难以训练，当网络层数超过某个值后，网络的识别能力反而会呈现下降趋势。在梯度反向传播的过程中，由于网络层数过深，靠近输出层的网络参数会很快收敛，而靠近输入层的参数则收敛得很慢。为了避免当网络层数过深引起的识别准确率下降的问题，残差网络引入了残差单元，即通过在卷积层之间加入Shortcut结构（跳接），这种结构使网络要训练的目标函数变为了减去输入函数后的残差。设f(x)为原始网络输出，在引入Shortcut结构后实际输出设为 h(x)，有h(x)=f(x)+x，即实际输出为原始输出与原始输入相加，从而将网络对f(x)的拟合转变为对h(x)的拟合。这种结构没有增加新的参数与额外的计算量，同时也解决了网络反向传播过程中梯度弥散的问题。 本系统采用Resnet残差网络，相较于传统网络，残差网络更易于捕捉模型细微波动，具有更快的收敛速度。残差网络是一种深层次卷积神经网络。对于卷积神经网络而言，加深网络层数可以增强其拟合能力。但随着网络层数的加深，卷积神经网络会变得非常难以训练，当网络层数超过某个值后，网络的识别能力反而会呈现下降趋势。在梯度反向传播的过程中，由于网络层数过深，靠近输出层的网络参数会很快收敛，而靠近输入层的参数则收敛得很慢。为了避免当网络层数过深引起的识别准确率下降的问题，残差网络引入了残差单元，即通过在卷积层之间加入Shortcut结构（跳接），这种结构使网络要训练的目标函数变为了减去输入函数后的残差。设f(x)为原始网络输出，在引入Shortcut结构后实际输出设为 h(x)，有h(x)=f(x)+x，即实际输出为原始输出与原始输入相加，从而将网络对f(x)的拟合转变为对h(x)的拟合。这种结构没有增加新的参数与额外的计算量，同时也解决了网络反向传播过程中梯度弥散的问题。

**3.4.1. 构建网络**

本系统构建的残差网络结构如图所示。 本系统构建的残差网络结构如图所示。

1. 卷积层：nn.Conv2d(1, 64, kernel|\_size=7, stride=2, padding=3)将输入图像进行卷积操作，并设置输出通道/特征数为64，卷积核kernel\_size为7x7大小，步长stride为2，3填充（四周补零）。此处输入图像是经由图像预处理后得到的单通道（深度为1），高height和宽width都是28的图像，因而参数中输入通道数为1。在进行卷积运算时，输入矩阵的边缘会比矩阵内部的元素计算次数少，且输出矩阵的大小会在卷积运算中相比较于输入变小。因此，可在输入矩阵的四周补零，称为padding，此处值为3。通俗一些说，就是在原矩阵的周围补三圈零。进行卷积运算时，7x7的卷积核在输入矩阵上移动，进行点积运算。移动步长stride为2，即过滤器每次移动2个单元。由公式(width-kernel\_size+2\*padding)/stride+1可以得出，输出矩阵大小为14x14，通道为64。
2. BatchNorm层归一化：计算输入各个维度的均值和标准差。在训练时，该层计算每次输入的均值与方差，并进行移动平均，默认动量值为0.1；验证时，训练求得的均值/方差将用于标准化验证数据。nn.BatchNorm2d(64)使卷积层得到的通道为64（64个维度，将得到64个均值）的特征矩阵满足均值为0，方差为1的分布规律，增加矩阵内元素落在在{-1,0,1}集合的概率，加快训练网络的收敛性。
3. Relu激活层：激活层给神经元引入非线性因素，增加了网络的非线性，神经网络可以任意逼近任何非线性函数，以便运用到非线性模型中。网络中采用nn.ReLU()激活函数，公式为f(x) = max(0, x)。该函数克服了梯度消失的问题，有效加快训练速度。
4. MaxPool池化层：池化层主要作用是降维、减少参数量、去除冗余信息、对特征进行压缩、简化网络复杂度、减少计算量、减少内存消耗等等。对应代码为nn.MaxPool2d(kernel\_size=3, stride=2, padding=1)，设置过滤器大小为3，步长为2，1填充。高和宽皆为2的输入图片被不重叠的分割成若干个同样大小的小块（kernel\_size）。每个小块内只取最大的数字，再舍弃其他节点后，保持原有的平面结构得出 output。和卷积操作不同，池化作用于图像中不重合的区域。 由公式floor( (width+2\*padding-\*kernerl\_size)/stride+1 )可得，输出为7x7大小的图像。
5. ResNet\_block残差块：网络中设置了三层残差块。如图所示，整个方框表示残差模块。x：残差块的输入，整个模块的输入，即之前层的输出；H(x):残差块的输出，就是这个模块算完的结果，即下面的y通过激活函数的结果；F(x):残差，即训练的目标；W1:第一层的权重；W2:第二层的权重。学习的目的就是找到输出到输入的映射，H(x)=F(x)+x；F(x)=H(x)-x，即就是要不断找到这样的F(x),而F(x)就是权重与输入的运算关系，通过不断迭代最终找到合适的权重，并且这个权重不会受到除了残差模块外其他层的过度影响,进而降低了梯度消失和爆炸问题。图中右侧的曲线即为shortcut（跳接），通过跳接在激活函数前，将上一层（或几层)之前的输出与本层计算的输出相加，将求和的结果输入到激活函数中做为本层的输出。用数学语言描述，假设Residual Block的输入为，则输出y等于:y = F(x,{W})＋x。其中F(x,{Wi}）是我们学习的目标，即输出输入的残差y-x。以上图为例，残差部分是中间有一个Relu激活的双层权重，即:F= W2σ(W1x)，其中σ指代Relu，而W1,W2指代两层权重。
6. GlobalAvgPool(GAP)全局平均池化层：对于输出的每一个通道的特征图的所有像素计算一个平均值，得到一个新的1x1的通道图，即：尺寸为 [ B, C, H, W]的特征图经GAP后，尺寸变为[B, C, 1, 1]。GAP将空间上的信息直接用均值代替，参数会更少，有效抑制过拟合。
7. Flatten层：所用函数为FlattenLayer()，顾名思义就是压平矩阵，将大小为n\*channel\* height\*width的输入数据变成一个简单的向量，其大小为n\* (channel\*height\*width)。其中n为数量，channel为通道数。
8. 线性层/全连接层Linear：nn.Linear(256, 10)对输入的256大小一维张量应用线性变换y = A^(T) \* x + b，输出大小为10的张量，其中A为权重矩阵。
9. 输出层：本系统主要目的是对手写数字分类识别，对分类问题的判断结果实际上是来自于概率，我们需要考虑两个问题：每个输出信号值在0至1之间；所有输出信号的和为1。下图为softmax函数公式。 输出层：本系统主要目的是对手写数字分类识别，对分类问题的判断结果实际上是来自于概率，我们需要考虑两个问题：每个输出信号值在0至1之间；所有输出信号的和为1。下图为softmax函数公式。exp(x)表示ex的指数函数，ak是输出层中第k个输入信号，exp(ak)表示ak的指数函数。分母表示输出层共有n个输出信号(神经元)，并计算所有输出层中的输入信号的指数和。yk是第k个神经元的输出。 softmax函数的输出是0到1，且输出值的总和为1，因此把softmax函数的输出概率解释为“概率”。我们把输出概率最大的神经元所对应的类别作为识别结果，把最大输出概率作为分数，代表被预测数字的可识别性和正确性。

**3.4.2. 训练网络**

hand\_wrtten\_train.py里面训练那块代码实在看不懂了

**3.4.3. 测试网络**

贴一下被测图像和测试结果就行。

**4. 数据设计**

本系统无数据库。训练结束得到的权重文件存放在logs文件夹。

**5. 系统部署**

1. 系统部署

**6. 其它事项**

暂无。