

**计算机视觉实验报告**

**实验题目**  学号识别

**学生姓名**  杜述超

**学 号**  2021214745

**专业班级**  21计科-1班

**指导教师**  吴晶晶

**完成日期**  2023.12.20

**合肥工业大学 计算机与信息学院**

# 实验目的

实验内容：

手写数字的识别是机器视觉的入门级项目，是机器视觉的“Hello word”，其在实际场景中有广泛的应用场景。请设计手写数字识别方法识别自 己的学号照片。

具体要求：

➢ 任务输入：学号照片。

➢ 任务输出：学号。

➢ 训练集：MNIST。

➢ 代码语言不限，方法不限，要求提交整个算法源代码，模型结果，算法分析等内容。

➢ 加分项（5分）：使用深度学习方法，代码环境名称以姓名缩写命名（例如吴晶晶的环境名：wjj），实验报告中介绍代码环境配置过程

# 实验原理

1. **MNIST数据集格式：**

train数据：

1，前4个字节（第0～3个字节）是魔数2051（int型，0x00000803, 大端）;

2，再往后4个字节（第4～7个字节）是图像的个数：60000或10000（第1个维度）；

3，再往后4个字节（第8～11个字节）是图像在高度上由多少个像素组成（第2个维度，高28个像素）；

4，再往后4个字节（第12～15个字节）是图像在宽度上由多少个像素组成（第3个维度，宽28个像素）；

3，再往后是一个三维数组，表示10000个或60000个分辨率28x28的灰度图像，一句话来说就是10000x28x28个像素，每个像素的值为0～255（0是背景，为白色；255是黑色）。

label数据：

1，前4个字节（第0～3个字节）是魔数2049（int型，0x00000801, 大端）;

2，再往后4个字节（第4～7个字节）是标签的个数：60000或10000；

3，再往后每1个字节是一个无符号型的数，值为0～9。

对于这里的对齐问题，由于MNIST数据集是用大端对齐的，但是Windows读取后是小端对齐的，所以这里还需要对读取的超过一字节的数据做reverse操作

还有一点，数字图片的格式是要让手写体的重心在图片的中心才能有效识别，这里也需要处理。

1. **ANN模型原理：**

在这个实验中我是采用了OpenCV库自带的ANN模型，所以对这个模型架构原理稍微说明一下：

1. **多层感知器（MLP）**：这是一种前馈神经网络，由多个层组成，包括输入层、隐藏层和输出层。每一层由多个神经元组成，这些神经元在层与层之间通过权重连接。
2. **神经元和激活函数**：每个神经元接收来自前一层神经元的输入（或者在输入层的情况下，接收外部输入数据），然后通过一个激活函数处理这些输入。常用的激活函数包括Sigmoid、ReLU等。
3. **权重和偏置**：每个连接都有一个权重，每个神经元可能有一个偏置。这些权重和偏置是网络在训练过程中学习的参数。
4. **前向传播**：在前向传播过程中，输入数据从输入层经过每个隐藏层，直到输出层，每一层的输出都是下一层的输入。
5. **反向传播和学习**：训练ANN涉及到使用算法（如梯度下降）调整权重和偏置，以最小化预测和实际输出之间的差异。这通常是通过反向传播算法实现的，该算法计算误差并将其传播回网络，以更新参数。
6. **损失函数**：训练过程中使用损失函数（如均方误差或交叉熵损失）来衡量模型的性能。

OpenCV中的ANN模型提供了这些基础功能，并使得用户可以自定义网络的结构，包括层数和每层的神经元数量。此外，它还提供了训练和预测的接口，使用户能够轻松地应用ANN模型到各种计算机视觉和机器学习问题中。

# 实验内容

实验内容主要分为训练数据处理，训练模型，识别数据处理，深度学习环境介绍。

1. **训练数据处理：**

由于train和label的数据处理基本一致，所以这里只说明train数据处理过程。

核心算法思路：

1. **打开文件**：使用**std::ifstream**以二进制模式打开指定路径的文件。
2. **读取文件头信息**：从文件中读取魔数（magic number）、图像数量、每张图像的行数和列数。
3. **字节顺序转换**：由于存储的整数可能是大端字节序，所以使用**reverseInt**函数将它们转换为主机的字节序（通常是小端）。
4. **初始化矩阵**：创建一个**cv::Mat**矩阵，用于存储所有图像。矩阵的行数等于图像数量，列数等于每张图像的像素数（行数×列数）。
5. **读取图像数据**：逐个读取图像数据，每个像素值由一个无符号字符（**uchar**）表示，并将其转换为浮点数，范围归一化到0到1之间。
6. **关闭文件**：读取完所有数据后关闭文件。

核心代码：

cv::Mat read\_images(std::string path)

{

int magic\_number = 0;

int number\_of\_items = 0;

int rows = 0, cols = 0;

// 打开文件

std::ifstream file(path, std::ios::binary);

if (!file.is\_open())

{

std::cout << "read image failed" << std::endl;

system("pause");

return cv::Mat();

}

// 读取文件头信息

file.read((char\*)&magic\_number, sizeof(magic\_number));

file.read((char\*)&number\_of\_items, sizeof(number\_of\_items));

file.read((char\*)&rows, sizeof(rows));

file.read((char\*)&cols, sizeof(cols));

// 字节顺序转换

magic\_number = reverseInt(magic\_number);

number\_of\_items = reverseInt(number\_of\_items);

rows = reverseInt(rows);

cols = reverseInt(cols);

// 输出读取的信息

std::cout << "magic\_number: " << magic\_number << std::endl;

std::cout << "number\_of\_items: " << number\_of\_items << std::endl;

std::cout << "rows: " << rows << std::endl;

std::cout << "cols: " << cols << std::endl;

// 初始化矩阵

cv::Mat images = cv::Mat::zeros(number\_of\_items, rows \* cols, CV\_32FC1);

// 读取图像数据

for (int i = 0; i < number\_of\_items; i++)

{

for (int j = 0; j < rows \* cols; j++)

{

uchar temp = 0;

file.read((char\*)&temp, sizeof(temp));

images.at<float>(i, j) = float(temp) / 255.0; // 归一化到0到1

}

}

// 关闭文件

file.close();

return images;

}

1. **训练模型：**

核心算法思路：

1. **准备训练和测试数据**：读取训练和测试图像及其对应的标签，将标签转换为one-hot编码。
2. **构建并训练ANN模型**：定义ANN模型的层结构和参数，然后用训练数据训练模型。
3. **预测并计算准确率**：使用训练好的模型对测试数据进行预测，并计算准确率。
4. **保存模型**：将训练好的模型保存到指定路径。

核心代码：

void train(std::string model\_path, std::string train\_images\_path, std::string train\_labels\_path, std::string test\_images\_path, std::string test\_labels\_path)

{

// 第一部分：准备训练数据

std::cout << "\*\*\*\*\*start read data\*\*\*\*\*" << std::endl;

// 读取训练和测试数据及标签

cv::Mat train\_labels = read\_labels(train\_labels\_path);

cv::Mat train\_images = read\_images(train\_images\_path);

cv::Mat test\_labels = read\_labels(test\_labels\_path);

cv::Mat test\_images = read\_images(test\_images\_path);

// 将训练标签转换为one-hot编码

cv::Mat train\_labels\_one\_hot = one\_hot(train\_labels, 10);

std::cout << "\*\*\*\*\*read data finished\*\*\*\*\*\n" << std::endl;

// 第二部分：构建ANN模型并训练

std::cout << "\*\*\*\*\*start train\*\*\*\*\*" << std::endl;

// 创建ANN模型

cv::Ptr<cv::ml::ANN\_MLP> ann = cv::ml::ANN\_MLP::create();

// 设置ANN层的大小

cv::Mat layerSize = (cv::Mat\_<int>(1, 3) << 784, 64, 10);

ann->setLayerSizes(layerSize);

// 设置训练方法和激活函数

ann->setTrainMethod(cv::ml::ANN\_MLP::BACKPROP, 0.001, 0.1);

ann->setActivationFunction(cv::ml::ANN\_MLP::SIGMOID\_SYM, 1.0, 1.0);

// 设置终止条件

ann->setTermCriteria(cv::TermCriteria(cv::TermCriteria::MAX\_ITER | cv::TermCriteria::EPS, 10, 0.0001));

// 创建训练数据并训练模型

cv::Ptr<cv::ml::TrainData> train\_data = cv::ml::TrainData::create(train\_images, cv::ml::ROW\_SAMPLE, train\_labels\_one\_hot);

ann->train(train\_data);

std::cout << "\*\*\*\*\*train finished\*\*\*\*\*\n" << std::endl;

// 第三部分：预测并计算准确率

std::cout << "\*\*\*\*\*start predict\*\*\*\*\*" << std::endl;

// 进行预测

cv::Mat pre\_out = cv::Mat::zeros(test\_images.rows, 10, CV\_32FC1);

float ret = ann->predict(test\_images, pre\_out);

// 计算准确率

int equal\_nums = 0;

for (int i = 0; i < pre\_out.rows; i++)

{

cv::Mat temp = pre\_out.rowRange(i, i + 1);

double maxVal = 0;

cv::Point maxPoint;

cv::minMaxLoc(temp, NULL, &maxVal, NULL, &maxPoint);

int max\_index = maxPoint.x;

int test\_index = test\_labels.at<int>(i, 0);

if (max\_index == test\_index)

equal\_nums++;

}

// 输出准确率

float acc = float(equal\_nums) / float(pre\_out.rows);

std::cout << "accuracy on test data set: " << acc \* 100 << "%" << std::endl;

// 保存模型

ann->save(model\_path);

std::cout << "\*\*\*\*\*predict finished\*\*\*\*\*\n" << std::endl;

}

1. **识别数据处理：**

核心算法思路：

1. **读取图片**：以灰度模式读取图片，并检查图片是否成功读取。
2. **图片预处理**：对图片进行反色处理以突出图像中的数字。
3. **二值化**：将图片转换为二值图像，以便于后续的图像处理。
4. **连通区域标记**：使用广度优先搜索（BFS）算法对二值化后的图像进行连通区域标记。
5. **提取边界**：获取每个连通区域的边界。
6. **排序**：根据边界的位置对图像进行排序，确保数字的顺序正确。
7. **分割和调整大小**：根据边界将图像分割成单个数字，并调整这些图像的大小以适配ANN模型的输入。
8. **加载模型和预测**：加载训练好的ANN模型，并对每个调整大小后的图像进行分类。
9. **解析结果**：将分类结果组合成一个数字序列。

核心代码：

std::string pic\_path = "./image/test11.jpg";

// 以灰度模式读取图片

cv::Mat img = cv::imread(pic\_path, cv::IMREAD\_GRAYSCALE);

if (img.empty())

{

std::cout << "read image failed" << std::endl;

system("pause");

return 0;

}

// 对图片进行反色处理

img = 255 - img;

down(img);

// 对图片进行二值化

cv::Mat binary = binaryzation(img, 100);

cv::Mat labels = cv::Mat::zeros(binary.size(), CV\_32SC1);

// 使用BFS算法进行连通区域标记

int labels\_num = bfs(binary, labels);

// 获取每个连通区域的边界

std::vector<std::vector<int>> boundary = get\_boundary(labels, labels\_num);

// 根据边界的位置对图像进行排序

std::vector<int> order = get\_order(labels, labels\_num);

// 分割图像

std::vector<cv::Mat> imgs = cut(img, boundary, order);

// 调整图像大小

std::vector<cv::Mat> resize\_imgs = resize\_(imgs, 28, 28);

// 加载训练好的ANN模型

cv::Ptr<cv::ml::ANN\_MLP> ann = cv::ml::StatModel::load<cv::ml::ANN\_MLP>(model\_path);

long long result = 0;

for (int i = 0; i < resize\_imgs.size(); i++)

{

cv::Mat cur\_img = resize\_imgs[i];

cv::Mat img\_show = cur\_img.clone();

// 转换图像格式并归一化

cur\_img.convertTo(cur\_img, CV\_32F);

cur\_img = cur\_img / 255.0;

cv::Mat pre\_img = cur\_img.reshape(1, 1);

cv::Mat pre\_out;

// 使用ANN模型进行预测

float ret = ann->predict(pre\_img, pre\_out);

double maxVal = 0;

cv::Point maxPoint;

// 获取预测的最大值

cv::minMaxLoc(pre\_out, NULL, &maxVal, NULL, &maxPoint);

int max\_index = maxPoint.x;

// 组合最终的数字序列

result = result \* 10 + max\_index;

std::cout << "max\_index:" << max\_index << "; ";

std::cout << "maxVal:" << maxVal << std::endl;

cv::imshow("img", img\_show);

cv::imwrite("./image/out/" + std::to\_string(i) + ".jpg", img\_show);

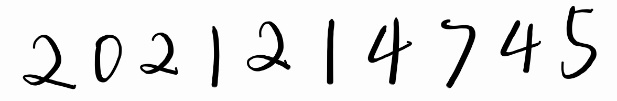
cv::waitKey(0);

}

std::cout << "result: " << result << std::endl;

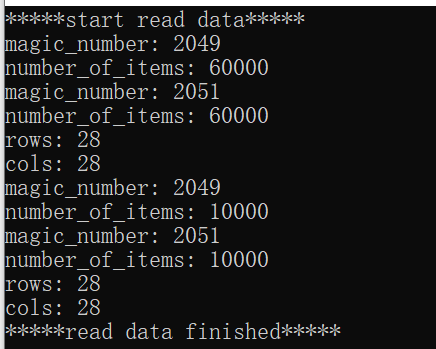
运行结果：

输入图片：



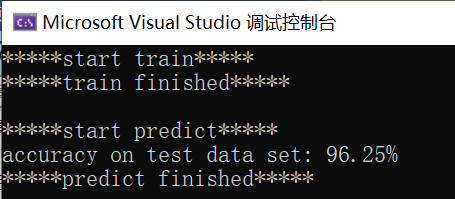
图表 1 输入测试图片

数据处理结果：



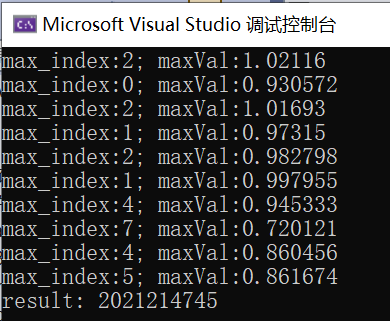
图表 2 数据处理结果

模型训练结果：



图表 3 模型训练结果

测试结果：



图表 4 输出结果

1. **深度学习环境：**

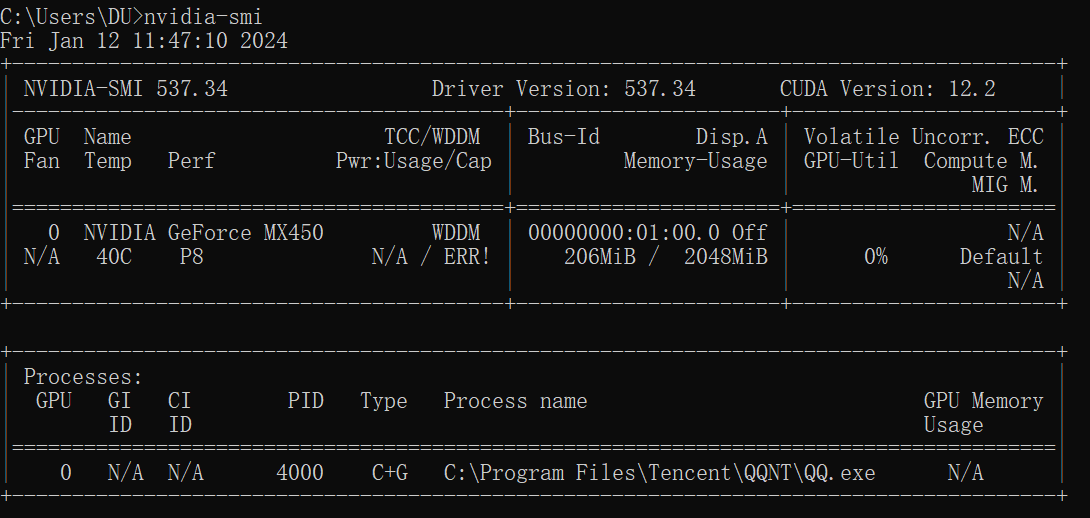
因为我之前做一个项目的时候配置过的，所以这里就不方便给出详细的配置过程，可以给出配置完成的测试结果：

1. 安装Anaconda测试结果：



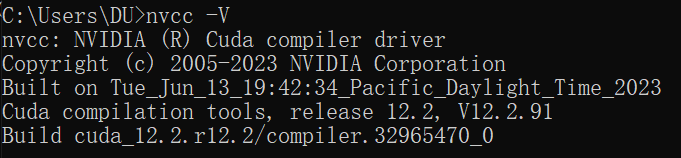
图表 5 Anaconda测试结果

1. 安装NVIDIA显卡驱动结果：



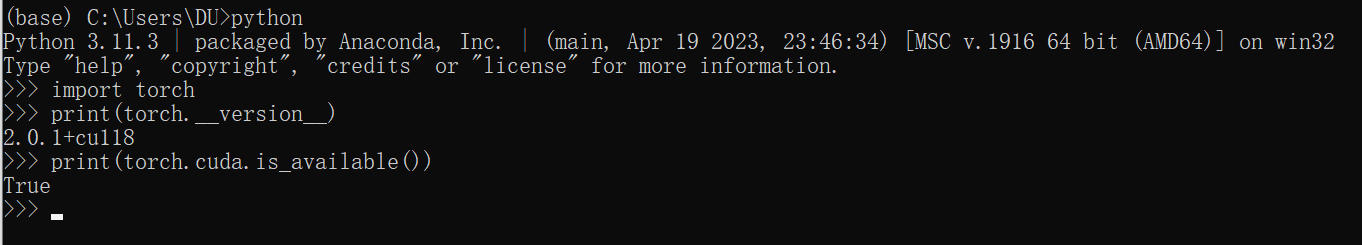
图表 6 NVIDIA显卡驱动测试结果

1. 安装CUDA和cnDNN：



图表 7 CUDA和cnDNN安装结果

1. 安装pytorch



图表 8 pytorch安装结果

# 总结体会

1. 实验目的：手写数字识别是机器视觉领域的基础任务，对于理解图像处理和模式识别算法至关重要。此次实验目的在于通过识别学号照片来掌握这一基本技能。
2. 实验过程：我使用了基于ANN的机器学习方法。首先，我使用了MNIST数据集来训练模型，该数据集包含大量的手写数字样本，非常适合此类任务。在数据预处理阶段，我对学号照片进行了灰度化、归一化和大小调整，以适应模型的输入要求。我构建的网络包含多个隐藏层，并使用ReLU激活函数和softmax输出层。
3. 代码和环境配置：我选择了OpenCV中的ANN框架进行实验，因为它们在机器视觉领域机器学习是有效的，拥有丰富的库和社区支持。
4. 算法分析：使用ANN进行手写数字识别表现出色，准确率较高。不过，模型在处理背景复杂或数字风格显著不同于训练集的图像时，准确度有所下降。
5. 实验结果：模型成功识别了大多数学号照片中的数字。对于识别错误的案例，我发现这些图像通常具有较低的对比度或者数字形式与训练集中的数字差异较大。
6. 实验反思：通过这次实验，我深刻理解了神经网络在图像识别任务中的应用，并学会了如何处理实际图像数据。在未来，我计划探索更先进的网络架构，并尝试在更多样化的数据集上进行训练，以提高模型的泛化能力和鲁棒性。