Handwritten Digit Recognition Using C++ and OpenCV

Junheng Ying

December 17, 2023

Abstract

OpenCV has a wide range of applications and is useful for a variety of machine learning and artificial intelligence. At the same time, everyone's handwriting is different, and it is difficult for machines to recognize different handwriting in daily life. Therefore, it is very important to write a program that can accurately recognize people's handwritten characters, and handwriting recognition by computers is also of great help to people's life. Based on the use of C++ and OpenCV libraries, this project has written a program for human handwritten digit recognition.

Introduction

Handwritten digit recognition program is a technology based on C++ and OpenCV that uses a support vector machine (SVM) algorithm to recognize handwritten digits. The application prospects of this program are very wide, for example, it can be used in postal code recognition, bank check recognition, automatic form filling12.

In terms of postal code recognition, handwritten digit recognition programs can help the post office automatically identify the postal code on the mail, thereby improving the efficiency of mail sorting. In terms of bank check recognition, the handwritten digit recognition program can help the bank automatically identify the amount on the check, thereby improving the efficiency of the bank's processing of the check. In terms of automatic form filling, handwritten digit recognition program can help users automatically identify the numbers filled in, thereby improving the

efficiency of form filling.

In addition, the handwritten digit recognition program can also be applied to the census, report card entry, industry annual inspection, financial statement entry and other large-scale data statistical fields, need to input a large number of data, previously completely rely on manual input, consume a lot of manpower and material resources. If the use of handwritten character recognition technology will greatly improve the efficiency of this aspect of work.

In short, handwritten digit recognition program has a wide range of application prospects, can help people improve work efficiency, reduce the waste of manpower and material resources, but also can provide support for the core and key automation system.

Program implementation

Handwritten digit recognition program is a technology based on C++ and OpenCV that uses a support vector machine (SVM) algorithm to recognize handwritten digits.

The implementation process of the program includes the following steps:

Environmental preparation:

Start by configuring the C++ and OpenCV environments in vscode. You can refer to https://www.bing.com/search?q=vscode%E9%85%8D%E7%BD%AEOpenCV. After the successful establishment of the program implementation.

Data preparation:

According to the provided data model, train-images.idx3-ubyte, train-labels.idx1-ubyte, train the program.

Program implementation:

1. Read data: the images in the training data are converted into binary images. The

readerimage.cpp and readerable.cpp in the program process each data and finally return it to the training model for training. In these two programs, there are two functions, readlables and readimages.

2. The standard SVM model is used to train the processed data, and the trained model is stored in SVM_DATA.xml, and the subsequent number recognition can be processed by the trained model to reduce the program time. In the SMV.cpp program, the main function is included, which is used for machine training and reading the already processed image.

3. Load the trained SVM model. Read handwritten digital pictures to be recognized. The recognition of handwritten digital image is preprocessed, including binarization, denoising, character segmentation and so on. The pre-processed handwritten digital images were input into SVM model for classification. The identification result is displayed. The finding.cpp program includes the output program and main function, which are used to read the trained model and identify the digital images to be verified. If there are insufficient images, "No images" will be reported. For each image, the name of the image and the predicted number will be output.

SVM Introduction:

Support Vector Machine (SVM) is a commonly used supervised learning algorithm, widely used in classification and regression problems. SVM implements classification by constructing a hyperplane in the feature space. The basic idea is to find an optimal hyperplane that can separate samples of different classes. In binary classification problems, the goal of SVM is to find a hyperplane [1] that maximizes the distance between classes (Margin).

The principle of SVM can be described in the following steps:

1. Data preparation: Given a training dataset on a feature space, where each sample has a

category tag.

2. Find the optimal hyperplane: Find a hyperplane in the feature space, so that the hyperplane can separate different categories of samples, and maximize the distance from the hyperplane to the nearest sample point.

3. Solve the optimization problem: the problem of finding the optimal hyperplane is transformed into a problem of solving convex quadratic programming, and the optimal hyperplane is obtained by solving the problem.

In SVM, an optimal hyperplane is a hyperplane that separates samples of different classes and maximizes the distance between the hyperplane and the nearest sample point. This distance is called the Margin. In two-dimensional space, the optimal hyperplane is a straight line, while in higher-dimensional space, the optimal hyperplane is a hyperplane.

The SVM solution process can be divided into two steps: training and prediction. In the training phase, the SVM uses the training data set to find the optimal hyperplane. In the prediction phase, SVM uses the optimal hyperplane to classify new samples.

Conclusion

In machine learning research, model accuracy is crucial but must be balanced against computational costs to remain practical. It is essential to balance accuracy with computational efficiency. In industry, efficiency and cost-effectiveness govern kernel choice. The polynomial kernel may be preferred for its faster training and lower costs, even if it sacrifices some accuracy.