Real-time Recognition of Handwritten Numbers: A LeNet-5 Based Approach and Its Improvements

Zhenhao Liu

University College of London, London, WC1E 6BT, United Kingdom

robliu0122@gmail.com

Abstract. The main idea of this project is to use edge detection to construct the shapes of each number and segregate each contour into separate images in order to isolate the numbers. Then feeding the images through a neuro network to perform recognition. Finally, the information is printed onto a video showing both the paper and boxes, boxing the individual numbers. The results achieved are fairly accurate, however needs improvements on the accuracy of the number seven. The goal of this project is to enable the real-time recognition of various handwritten digits presented on a physical paper medium. This method places no restrictions on the colors of the paper or pen, as long as the digits remain legible. The dataset employed originates from the MNIST collection, and the model architecture used closely mirrors that of LeNet-5. The project is categorized into three main sections: model development, image parameter assessment, and implementation for real-time video analysis.

Keywords: Neuro network, LeNet-5, Convolutional network, Motion detection, Handwritten number recognition, Mnist.

1. Introduction

In the digital era, document analysis technologies are becoming increasingly crucial as they serve as a bridge connecting the physical and digital worlds. Among these, Handwritten Mathematical Expression Recognition (HMER) stands as a pivotal task whose significance cannot be overstated. Handwritten mathematical equations, serving as essential conduits for the dissemination and advancement of human knowledge, permeate numerous fields such as education, scientific inquiry, and the publishing industry. As online education continues to proliferate, along with the development of intelligent evaluation systems and digital repositories, the necessity for automated recognition of handwritten mathematical expressions has become increasingly critical.. Therefore, as a foundational project, the recognition of numbers is also important as it provides preparation and insight into the field of study. The majority of recognition projects focusing on handwritten numbers have the limitations of allowing only one number within the page and white paper with a black pen as an assumed condition. This project challenges to break these limitations and apply the recognition system to a real time video captured via a camera connected to the computer. Outputting a window showing the real time video of the paper with the numbers written on it being boxed in separate boxes, and the numbers indicated in text on the top section of the video. Although the initial objectives outlined earlier, the initiative can be seamlessly developed into practical applications such as vehicle license plates or other identification tag recognitions.

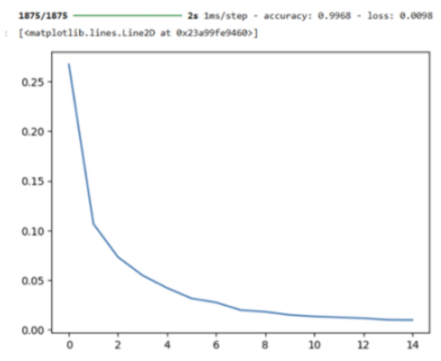
2. Model construction and training

A neural network is a computational framework that is derived from the operational principles of the human brain. [1]. It consists of layers of interconnected nodes that process data through weighted connections. Each neuron receives input, processes it through an activation function, and passes the result to the next layer. The network learns by adjusting these weights based on the errors it makes during training. When constructing models with large inputs such as images, the amount of computational power needed is often un-obtainable or inefficient for the purpose of use. The resolution of this issue can be approached through two primary considerations: Firstly, one can minimize the input data by downscaling the image to a reduced resolution and converting the color scheme from RGB to grayscale. Secondly, rather than employing fully connected layers, it is recommended to utilize convolutional networks to derive feature maps prior to the incorporation of any Dense layers. This project started by taking the first aspect of reducing input data, and then turning to the use of a convolution network with the aim of improving accuracy.

2.1. First model

This segment employs TensorFlow to establish a fully connected network (FCN) adept at recognizing individual digits within distinct images. [2]. The model consists of four layers: A flattened layer to convert the two-dimensional image into a one-dimensional vector, allowing further processing; two dense layers with 128 units and activation of relu to extract patterns; a single dense layer with 10 units and activation of SoftMax, matching the 10 integers. The input shape was set to a 28 by 28 grey image in order to compensate for the issue of having too many pixels to process, reducing the amount of weights needed to be trained. This is for decreasing the processing time and probability of over-fitting.

After compiling the model together with an optimizer of 'adam', loss of sparse categorical cross entropy, and metrics of accuracy. The model was trained using parts of the data set from mnist [3]. By viewing the loss graph, an epoch of 15 was chosen, leading to an accuracy of 99% and loss of 0.0098 during the last epoch.



```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9968 - loss: 0.0098
[<matplotlib.lines.Line2D at 0x23a99fe9460>]
```

Figure 1. Loss against epochs graph for model

Evaluations on test sets from mnist are also done and the accuracy maintained around 98%. However, when tests were done on handwritten numbers by me on the 'paint' windows application, the accuracy of recognition on the numbers 4 and 6 were low.

This problem was consistent with the final product of this project, showing possible overfitting. After attempts of improving the performance by changing hyper-parameters, the structure of the FCN, and the image pre-processing, the effects were little. Consequently, convolutional networks were introduced to enhance performance. [4].

2.2. Second model

Convolution neuro networks (CNN), as previously mentioned, possess the capability to decrease the computational power required while enhancing precision. Central to CNNs are the kernels, often referred to as filters. A filter is a compact, learnable matrix of weights that traverses the input data and executes a convolution operation. The purpose of each filter is to detect specific patterns in the input data. For example, filters in the first convolutional layer might detect edges, while filters in deeper layers might detect complex patterns like textures, shapes, or entire objects. A good example is indicated in Figure2. The table on the left indicates an image with a brighter color on the left than on the right. This image is convoluted by the filter in the middle, giving the outcome on the right indicating the position of the vertical edge.

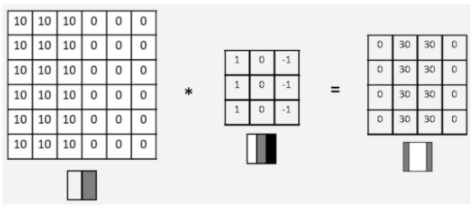


Figure 2. Convolution process using filters [5]

The usage of filters allows spatial features to be determined, capturing relationships between nearby pixels, which are important in understanding an image. Local connectivity and weight sharing make CNNs highly efficient. Instead of learning a separate weight for every pixel in the input, CNNs use the same filter (or kernel) across different regions of the input, reducing the number of parameters and the possibility of over-fitting. Further on, due to the nature of convolutional operations, CNNs are inherently translational invariant. A filter applied to one part of an image will detect the same feature in any other part of the image, enabling the network to detect objects regardless of their position. Making it well-suited for tasks like image classification.

The CNN used in this project is modified from LeNet-5 [6]. The model consists of 7 layers, not counting the input, with a combination of convolutional, pooling (subsampling), and fully connected layers. The input layer is adjusted to 28 by 28 as the image being input from the video is going to be in this shape, matching the shape of images in the mnist data set. The kernel size of most convolutional layers was reduced to match the reduction in size of the input image. Using the mnist data set, the accuracy of 98.9% was gained.

3. Image parameter testing

Image pre-processing and parameter tuning are crucial steps in optimizing machine learning models, especially in tasks like image recognition and computer vision. Pre-processing techniques, such as perspective transformation to a bird's-eye view, Gaussian blur, erosion, and dilation, are used to enhance the quality of input data, reduce noise, and highlight important features of the model. This section tests the pre-processing parameters. Aiming to find the contour of each number and construct them in separate images. This results in a set of images that have the correct format and can be fed through the model. The methodology employed is influenced by a lane detection initiative as well as a project focused on measuring object dimensions. [7, 8].

The physical set up is shown in Figure3(a), where a camera is pointed to a stack of paper and the relative positions of these objects are fixed. By manually labeling the vertices of the paper, the region of interest can be focused and warped to a bird’s eye perspective. Reducing any parallax deformations. The azure points depicted in Figure 3(b) indicate the vertices of the paper, and by employing functions from OpenCV , one can derive the transformation matrix. [9]. Then by plotting the image captured, the transformation matrix, and the target shape of the output image into ‘cv2.warpPerspective’, Figure3(c) was returned. Providing a more uniform perspective, making it easier for the model to recognize patterns.

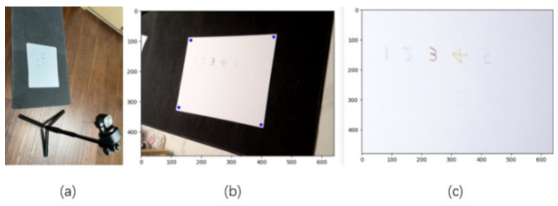


Figure 3. (a) Project setup, (b) Image directly captured, (c) Image after perspective transfer

After rotating and mirroring Figure3(c) into the correct orientation, edge detections are performed on this image. Gaussian Blurring was utilized to reduce noise, and the Canny edge detection algorithm was implemented to emphasize locations where there are significant changes in the image's brightness. [10]. Canny performs non-maximum suppression, which means it identifies and retains only the most prominent edges, eliminating spurious responses caused by noise or local fluctuations. This ensures that the edges are thin and precise. Morphological operations, including dilations and erosions, are used to further reduce noise while connecting disconnected lines. The contours are then recognized and labeled using 'cv2.findContours'. Re-organizing the contour labels from left to right, then drawing them onto the image of the paper.

Utilizing the 'cv2.minAreaRect' function, boxes surrounding the numbers are found [9] . Contours occupying an area of less than 25 pixels were ignored to reduce noise. By using the vertices of these boxes, new square boxes centering each number are constructed. Using the longest side and diagonal as the side length of the square. Basing the midpoints between most high and most low vertices as the vertical centrals, and the midpoint between most left and right as horizontal centrals. A two percent length of the sides is added to each side of the image in order to further centralize the number. All of the square boxes constructed will be up right in stead of following the orientation of the minimal area rectangles. This is because numbers like '4' are often boxed in a 45-degree angle even if the number was written up right. This results in a limitation in the performance of this project, which is, the numbers must be written in designed orientation. Otherwise it is likely that the numbers become miss interpretated.

Using the coordinates of square boxes for each number and the information of the contours, multiple new images centering each individual number can be created. For each number a black based canvas is constructed, and the number is printed on using filled contours. Due to the thinness of the lines of the handwritten numbers, the contours are often falsely connected. Resulting in parts of the number being filled while others are not. To compensate for this issue, a strong dilation followed by multiple erosion processes was applied. The idea is if the contours are miss connected, the line must be thin. There for the dilation will fuse the two parallel contours together, and then the erosion will return the number to its original size as indicated in Figure 4.

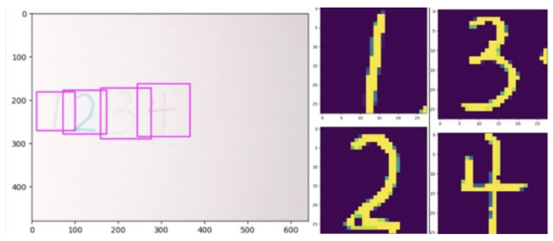


Figure 4. Boxed numbers and the reconstructed isolated numbers.

The isolated numbers are then rescaled to 28 by 28 pixels, and the resulting images are indicated on the right-hand side of Figure4. Each image will then be processed by the model, performing number recognition. Aiming to produce a real time recognition system, the previous processes were applied to a live video.

4. Application to video

The identification of handwritten numerals in images represents a significant breakthrough, facilitating applications such as automated data input, form analysis, and the digitization of notes. However, recognizing handwritten digits in live video streams has far greater practical applications. For example, in production line monitoring, instant feedback systems that perform recognition of digits can enable immediate action and decision-making, improving efficiency and accuracy. This real-time recognition capability not only increases automation, but also creates new possibilities for scaling solutions in industries such as banking, and transportation where speed, accuracy, and responsiveness are critical.

By applying the previous sections together on a video, a real-time recognition system can be constructed. However, running every frame through the model requires a large amount of computational power, which may lead to noticeable lags of the output video. To resolve this issue, a motion detection system is applied to the video. Performing a recognition process once, only under the following conditions: significant fluctuations are detected in the region of interest, and the fluctuations has stopped for at least 4 seconds. This dramatically reduces the number of frames that will be processed. The resulting product is shown below in Figure5(a).

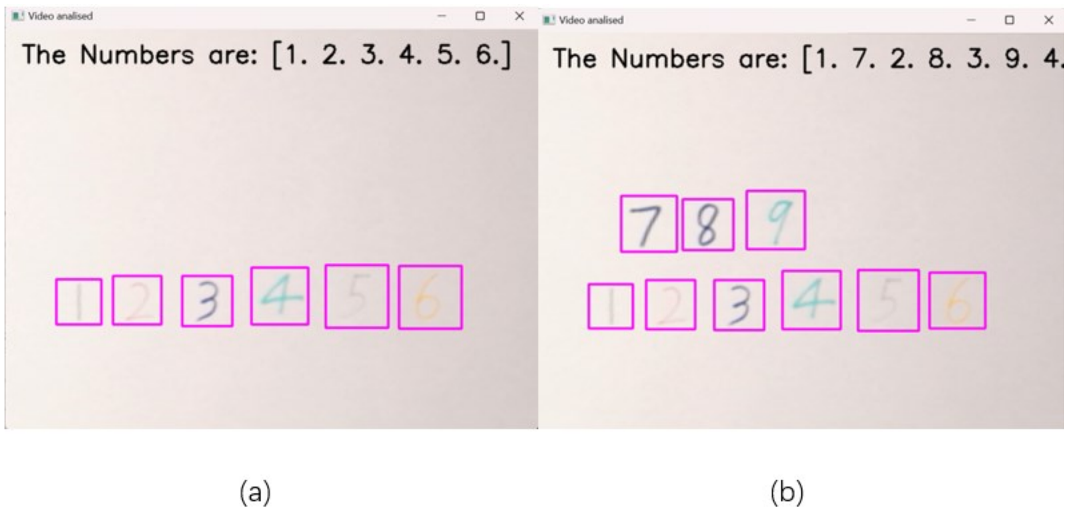

Video analised
The Numbers are: [1. 2. 3. 4. 5. 6.]
Video analised
The Numbers are: [1. 7. 2. 8. 3. 9. 4.
(a)
(b)

Figure 5. Image of video output

The video presents the annotated image featuring the boxed numerical identifiers along with forecasts regarding their identities when motion is absent in the designated area of interest.. When there is motion, the real time video of the paper will be shown until the motion is stops for 4 seconds. The predictions for the numbers are listed in the order of how the handwritten numbers are ordered from left to right. Tests for numbers in different vertical coordinates are done and shown in Figure5(b).

5. Conclusion

The overall precision is satisfactory; however, it is more prone to failure under two specific conditions.. First, when numbers are positioned too close horizontally but remain tall vertically, overlaps may occur during the segmentation process, making it difficult to distinguish individual digits. Second, the digit '7' can sometimes be misinterpreted as '1' if the horizontal bar at the top is too short. This may be caused by the erosion and dilation process, fusing the bar on the top of ‘7’ to the vertical line. Resulting in a look that is more similar to ‘1’ than intended.

Other improvements are also preserved. One of which is the maximum number of numbers that can be displayed is 7 as indicated in Figure5(b). This is due to the size of the window and text. It could be easily fixed by adding a correlation between the amount of numbers and font size.

This project could be brought further by adding another layer of edge detection, identifying the position of the paper instead of restricting the position of it to a fixed location. More applications could be released with this modification, such as license plate reading or inventory tracking. Eliminating the need for manual data entry and saving time.

In conclusion, these proposed enhancements not only promise to elevate the current project's capabilities but also pave the way for novel and impactful applications in the field of automated data collection and processing. The future of this technology is promising, with vast opportunities for exploration and growth.

References

[1] Ng, A. (n.d.). Neural networks and deep learning [Course]. Coursera. https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning
[2] TensorFlow. (n.d.). TensorFlow: An end-to-end open-source machine learning platform. TensorFlow. https://www.tensorflow.org/
[3] LeCun, Y., Cortes, C., & Burges, C. (n.d.). The MNIST database of handwritten digits.
[4] González, R. C. (2008). Digital image processing (3rd ed.). Pearson.
[5] Xu, M. F. (n.d.). Neural networks part 3: Convolutional neural networks (CNN). Michael F. Xu.
[6] Ang, L., Zhou, Y., Shi, J., & Yang, M. (2022). Convolutional neural network-based object detection for remote sensing images. Journal of Sensors, 2022, Article 1636203.
[7] Wang, X. (n.d.). SEU_LaneDetect [Source code]. GitHub. https://github.com/xlwang123/SEU_LaneDetect
[8] Rosebrock, A. (2016, March 28). Measuring size of objects in an image with OpenCV. PyImageSearch.
[9] OpenCV. (n.d.). OpenCV documentation (Version 4.x). OpenCV. https://docs.opencv.org/4.x/index.html
[10] Canny, J. (1986). A computational approach to edge detection. Pattern Recognition, 19(6), 679-698. https://doi.org/10.1016/S0031-3203(00)00023-6