Journal of Computer Engineering & Information Technology

Research Article A SCITECHNOL JOURNAL

Handwritten Digit Recognition

Shubhangi* and Ravi Shankar Pandey

Department of Computer Science and Engineering, Birla Institute of Technology Mesra, Jharkhand, India

***Corresponding author:** Shubhangi, Department of Computer Science and Engineering, Birla Institute of Technology Mesra, Jharkhand, India, Tel: 08102428844; E-mail: shubhangisingh453@gmail.com

Received date: 25 November, 2022, Manuscript No. JCEIT-23-81425;
Editor assigned date: 30 November, 2022, PreQC No. JCEIT-23-81425 (PQ);
Reviewed date: 14 December, 2022, QC No. JCEIT-23-81425;
Revised date: 02 January, 2023, Manuscript No. JCEIT-23-81425 (R);
Published date: 30 January, 2023, DOI: 10.4172/2324-9307.1000254

Abstract

Handwritten Digit Recognition (HDR) is the process of converting images of handwritten digit into digital format. A lot of money is wasted on converting the information that is in paper to digital format. This problem can be solved by using HDR. The heart of our project lies within the ability to develop an efficient algorithm that can recognize the handwritten digits which are scanned and sent as input by the user. The goal of this paper is to observe the variation of different algorithms that can classify the handwritten digits using different hidden layers, various numbers of epochs and to make a comparison based on the accuracy. This experiment is performed using the Modified National Institute of Standards and Technology (MNIST) dataset.

Keywords: Handwritten digit recognition; Algorithm; Machine learning; Language model; Convolutional neural network

Introduction

Developers are using different machine learning and deep learning techniques to make machines more intelligent. In deep learning, Convolutional Neural Networking (CNN) is being used in many fields like object detection, face recognition, spam detection, image classification. Handwritten digit recognition has not only professional and commercial applications, but also has practical application in our daily life and can be of great help to the visually impaired [1]. It also helps us to solve complex problems easily thus making our lives easier. Many algorithms have been developed for hand written digit recognition. But due to infinite variation in writing styles they are still not up to mark. Poor contrast, image text vagueness, disrupted text stroke, unwanted objects, deformation, disoriented patterns and also interclass and intraclass similarity also cause misclassification in handwritten numeral recognition system.

The goal of this project was to recognize handwritten digit and give the best output. A set of training data will be used for this purpose. The objective is to design and implement digit recognition in java that will detect handwritten digits drawn on screen similar to the training data. The problem of digit recognition has been studied extensively. A wide spectrum of techniques will be used including, template matching, neural networks, maximal rejection classification and model-based detection [2].

Materials and Methods

Early scanners

The first driving force behind handwritten text classification was for digit classification for postal mail. Jacob Rabinows early postal readers incorporated scanning equipment and hardwired logic to recognize mono-spaced fonts. By making a sophisticated scanner which allowed for more variations in how the text was written as well as encoding the information onto a bar code that was printed directly on the letter [3].

To the digital age

The first prominent piece of OCR software was invented by Ray Kurzweil in 1974 as the software allowed for recognition for any font. This software used a more developed use of the matrix method (pattern matching). Essentially, this would compare bitmaps of the template character with the bitmaps of the read character and would compare them to determine which character it most closely matched with. The downside was this software was sensitive to variations in sizing and the distinctions between each individual's ways of writing. To improve on the templating, OCR software began using feature extraction rather than templating. For each character, software would look for features like projection histograms, zoning, and geometric moments.

Future works

Firstly, to have more compelling and robust training, we could apply additional preprocessing techniques such as jittering. We could also divide each pixel by its corresponding standard deviation to normalize the data. Next, given time and budget constraints, we were limited to 20 training examples for each given word in order to efficiently evaluate and revise our model. Another method of improving our character segmentation model would be to move beyond a greedy search for the most likely solution.

We would approach this by considering a more exhaustive but still efficient decoding algorithm such as beam search. We can use a character/word-based language-based model to add a penalty/benefit score to each of the possible final beam search candidate paths, along with their combined individual softmax probabilities, representing the probability of the sequence of characters/words. If the language model indicates perhaps the most likely candidate word according to the softmax layer and beam search is very unlikely given the context so far as opposed to some other likely candidate words, then our model can correct itself accordingly [4].

Proposed method

The aim of proposed method is to develop a system of improved facilities. The proposed method can overcome few limitations of the existing system. The system provides proper accuracy and reduces manual work.

- Accuracy
- High Speed

- Increased Performance
- Reduced Error

The proposed model contains four stages to classify and detect the digits:

Pre-processing: Pre-processing is a part of HDR. If there are some rules like a box for each digit then, it will be much easier to detect the boundaries. The fundamental motivation behind pre-processing is to take off noise filtering, smoothing, and standardization. Binarization converts a Greyscale image into a binary image.

Feature extraction: Different type of algorithms used for feature extraction has different types of error rate. The errors made by each separate algorithm does not overlap, so combining all these methods lead to a perfect recognition rate and also helps to reject the ambiguous digits recognition and improve the recognition rate of misclassified digits that can be recognized by humans.

Classification and recognition: In the classification and recognition step, the extracted feature vectors are given as single input values to each classifier. CNN Convolution layer and the subsampling layer can have various different layers. The down sampling layer is also known as pooling layer. The image is divided into small segments of small areas, and a value is calculated for each area. Then the calculated values are rearranged in sequence to form a new image [5].

Results and Discussion

Training and testing

Finally, to evaluate a model, the test dataset is used. Training is less complex because each module is designed to handle a specific sub problem. It is expected that each module can tackle the specific problem more efficiently and accurately because each module is trained independently which is easy to add and delete modules.

Three different layer on which convolution neural network are performed: There are three types of layers in a convolutional neural network: Convolutional layer, pooling layer, and fully connected layer. Each of these layers has different parameters that can be optimized and performs different tasks on the input data. Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels. Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network. Fully connected layers/dense layer are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP (Figure 1) [6].

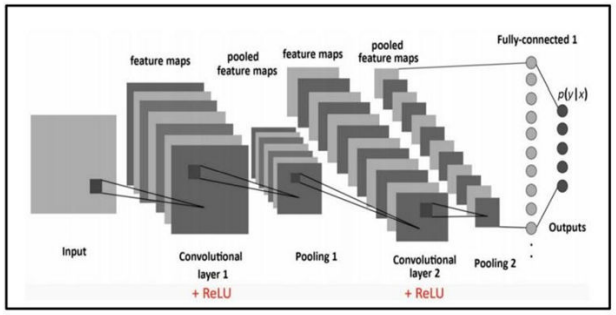

Figure 1: An example CNN with two convolutional layers, two pooling layers, and a fully connected layer which decides the final classification of the image into one of several categories.

Algorithm

Different processes performed for generating final model by applying Convolution neural network process are:

Loaded dataset: We know that the images are all pre-aligned (e.g. each image only contains a hand-drawn digit), that the images all have the same square size of 28 × 28 pixels, and that the images are gray scale. Therefore, I have loaded the images and reshape the data arrays to have a single color channel. I have used load dataset function which implements these behaviors and can be used to load the dataset.

Prepared pixel data: We know that the pixel values for each image in the dataset are unsigned integers in the range between black and white, or 0 and 255. But we know that some scaling will be required. A good starting point is to normalize the pixel values of grayscale images, e.g. rescale them to the range (0,1). This involves first converting the data type from unsigned integers to floats, then dividing the pixel values by the maximum value. I have converted integer to float and then normalized it to range 0-1.

Created tensor flow model: The model has two main aspects: The feature extraction front end comprised of convolutional and pooling layers, and the classifier backend that has made a prediction. For the convolutional front-end, I started with a single convolutional layer with a small filter size (3,3) and a modest number of filters followed by a max pooling layer. The filter maps are then be flattened to provide features to the classifier. Given that the problem is a multi-class classification task, I knew that I will require an output layer with 10 nodes in order to predict the probability distribution of an image belonging to each of the 10 classes. I also used softmax activation function. Between the feature extractor and the output layer, I added dense layer to interpret the features, in this case with 100 nodes. All layers have used the ReLU activation function and the He weight initialization scheme. I have used a conservative configuration for the stochastic gradient descent optimizer with a learning rate of 0.01 and a momentum of 0.9. The sparse categorical cross-entropy loss function was optimized, suitable for multi-class classification, and monitored the classification accuracy metric, which was appropriate given we had the same number of examples in each of the 10 classes.

Evaluate model: The model was then evaluated using Keras evaluate function. Each test set was the 20% of the training dataset, or about 12,000 examples, close to the size of the actual test set for this problem. The training dataset was shuffled prior to being split, and the sample shuffling was performed each time, so that any model I evaluated was having the same train and test datasets in each fold,

providing an end to end comparison between models. The model was evaluated using five-fold cross-validation. The value of k=5 was chosen to provide a baseline for both repeated evaluation and to not be so large as to require a long running time. I have trained the baseline model for a modest 10 training epochs with a default batch size of 32 examples. The test set for each fold was used to evaluate the model both during each epoch of the training run, so that we can later create learning curves, and at the end of the run, so that we estimated the performance of the model. As such, we kept track of the resulting history from each run, as well as the classification accuracy of the fold. A summary of the model performance was calculated. The model was having an estimated skill of about 98.6%, which is reasonable [7].

Visualized training process: Training process was visualized by plotting graph of accuracy *vs.* epoch number to see the difference between trained model and validation model.

Predicted result: At last I saved evaluated model, I used my saved model to make a prediction on new images. The model assumes that new images are grayscale, that they have been aligned so that one image contains one centered handwritten digit, and that the size of the image is square with the size 28 × 28 pixels (Figures 2 and 3).

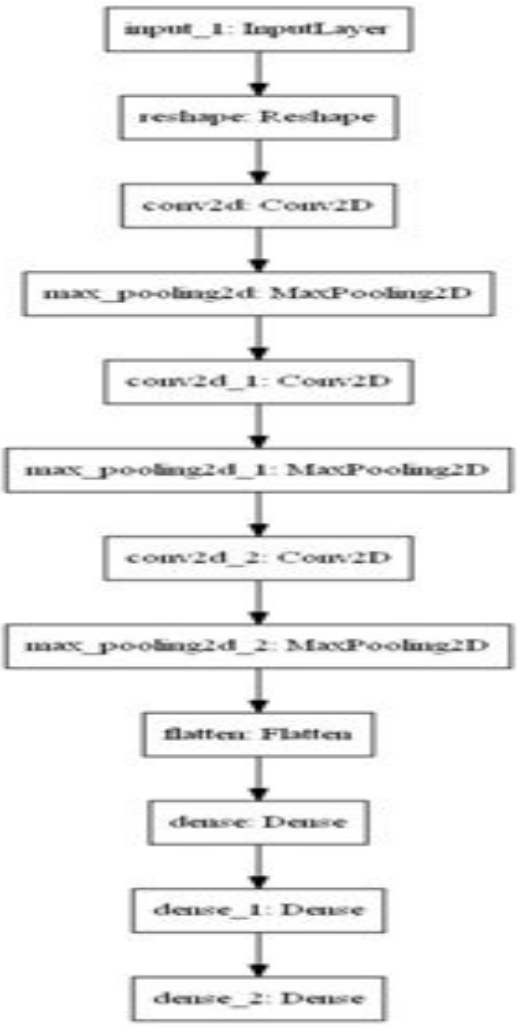

Figure 2: CNN model.

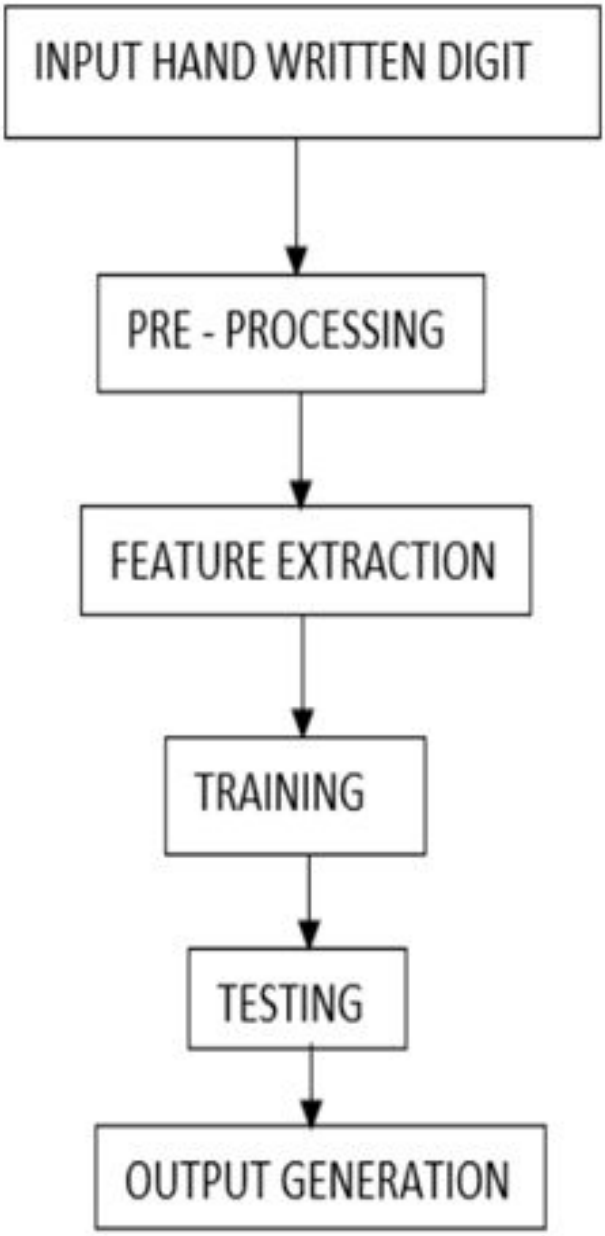

Figure 3: Architecture.

Feasibility study

There is a nice intersection between machine learning and front-end development, but we often find overwhelming to gain knowledge in both areas and get them to play together. My intention is to solve that with my project where I will dive into the basics of creating an ML model and how can I apply it to a mobile app built in Flutter.

The project work is a practical experience of the knowledge one has. This project entitled "hand written digit recognition application" will be practical project based on some trends of computer science. Every day the world is searching new techniques in the field of computer science to upgrade the human limitations into machines to get more and more accurate and meaningful data. The way of machine learning and artificial intelligence has no negative slop it has only the slop having positive direction. This project is a very basic idea of those concepts. This project deals with the very popular learning process called Neural Network. There are various ways by which one can achieve the goal to a desired output, but in machine learning Neural network gives a way that machine learns the way to reach the output. This project has come through the concepts of statistical modeling, the computer vision and machine learning libraries which includes a lot of study about these concepts. This project will be good explanation (Figures 4 and 5) [8,9].

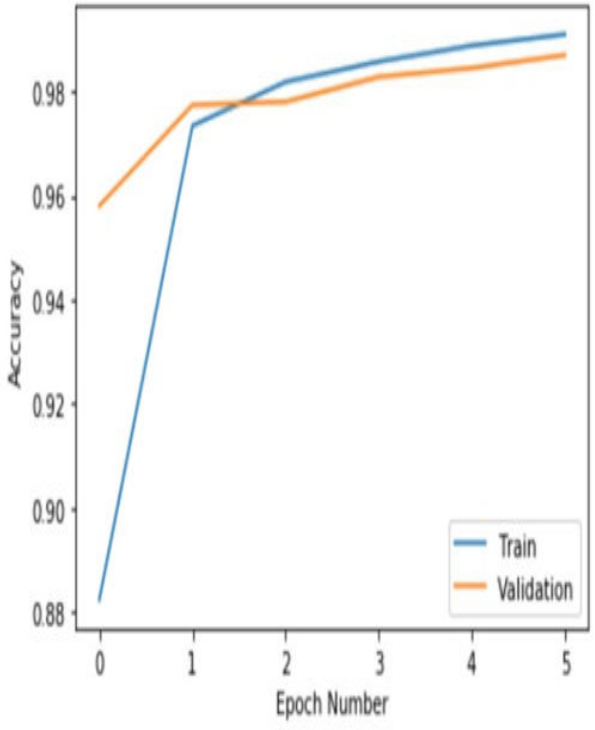

Figure 4: Accuracy learning curves for the baseline model during validation.

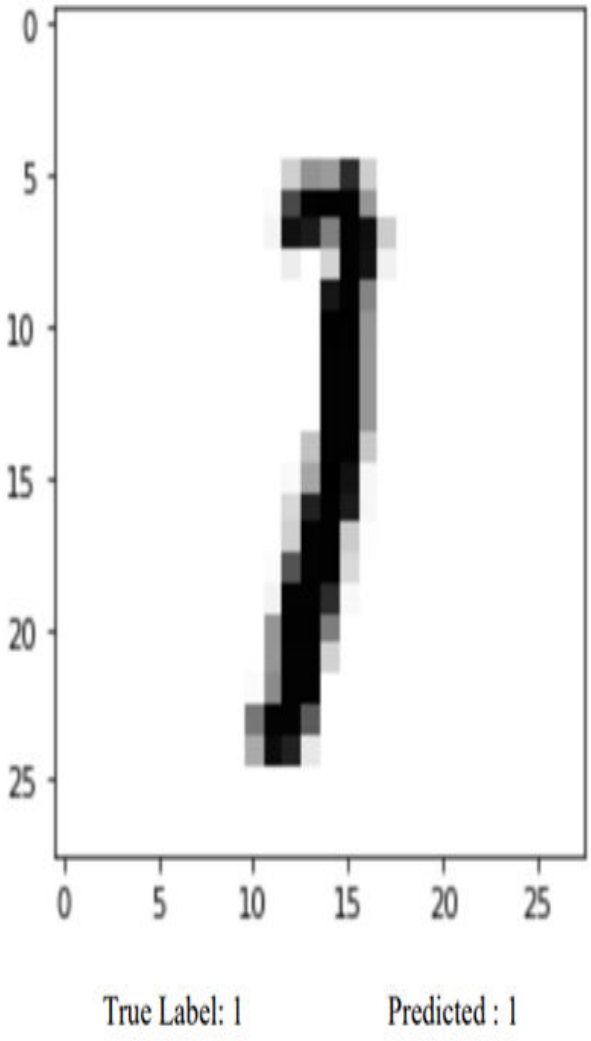

Figure 5: Prediction model.

Conclusion

Convolutional Neural Network gets trained from the real-time data and makes the model very simple by reducing the number of variables and gives relevant accuracy. In our project, we used CNN with some libraries like Keras, Matplotlib, CV2, Tensor flow to get the maximum accuracy. A comparison on different Machine Learning algorithms like random forest classifier, convolutional neural network, linear regression, K-nearest neighbors, and support vector machine is done, in which the accuracy for CNN is 99.63%.

- Taking huge datasets
- Adopting many suitable algorithms
- Hyper-parameter tuning
- Compile the model with a greater number of epochs

Acknowledgment

I would like to take this opportunity to thank the people who have made the implementation of this project possible. Completion of this Project without any guidance would not be possible which was led by our professor and teaching staff. I express our sincere gratitude to our beloved guide Dr. Ravi Shankar Pandey Sir, Assistant professor, who provided valuable guidance, suggestions and hand in hand cooperation throughout the completion of this project. I also wish to extend our sincere gratitude towards the teaching and non-teaching staff of the Department of Computer Science and Engineering for their technical support.

References

1. LeCun Y, Boser B, Denker J, Henderson D, Howard R (1989) Handwritten digit recognition with a back-propagation network. Adv Neural Informat Proces Syst 2: 396-404.
2. Han X, Li Y (2015) The application of convolution neural networks in handwritten numeral recognition. Int J Database Theor Appl 8: 367-376.
3. Krevat E, Cuzzillo E (2006) Improving off-line handwritten character recognition with hidden markov models. Transact Pattern Analys Machine Learn 33.
4. Tschopp F, Martel JN, Turaga SC, Cook M, Funke J (2016) Efficient convolutional neural networks for pixelwise classification on heterogeneous hardware systems. IEEE 13th International Symposium on Biomedical Imaging, pp. 1225-1228.
5. Ciresan DC, Meier U, Gambardella LM, Schmidhuber J (2010) Deep, big, simple neural nets for handwritten digit recognition. Neural Comput 22: 3207-3220.
6. Deng L (2012) The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process Magazine 29: 141-142.
7. Le Cun Y, Jackel LD, Boser B, Denker JS, Graf HP, et al. (1989) Handwritten digit recognition: Applications of neural network chips and automatic learning. IEEE Communicat Magazine 27: 41-46.
8. Sudhakar R, Rao PV (2019) Video super resolution using non-linear regression and deep learning. Imaging Sci J 67: 305-318.
9. Uchida S, Ide S, Iwana BK, Zhu A (2016) A further step to perfect accuracy by training CNN with larger data. In 2016 15th International Conference on Frontiers in Handwriting Recognition, pp. 405-410.