Generating LaTex Code for Handwritten Mathematical Equations based on CNN

Mahtab Faraji. [1] and Fatemeh Taghvaei. [2]

[1] *University of Illinois at Chicago, Department of Biomedical Engineering, Illinois, USA*
[2] *University of Illinois at Chicago, Department of Electrical and Computer Engineering, Illinois, USA*

Reception date of the manuscript:
Acceptance date of the manuscript:
Publication date:

Abstract— A complex issue in computer vision is the recognition and processing of handwritten mathematical equations. The problem is made even more challenging by the classification and segmentation of a single character. In this regard, several classifiers and algorithms have been proposed for handwritten recognition. Convolutional neural networks are one of the most effective techniques for handwritten recognition in terms of accuracy. In this project, we aim to implement a system that can take written mathematical equations as input and after recognizing each character which includes digits and mathematical symbols convert it to the corresponding latex code. In this regard, the performance of CNN with LeNet, will be evaluated in this project in order to classify symbols, such as digits, Latin letters, Greek letters, and mathematical operators. Then, each symbol will be transferred to the corresponding Latex letter.

Keywords— LaTeX, Handwritten mathematical equation, Convolutional neural networks

I. Introduction

Handwriting has traditionally been an integral part of human interaction and communication, but digital pens, interactive panels, and smart writing surfaces have replaced it. During the Covid-19 pandemic, schools and universities turned in-person classes into virtual classes. This led to a sudden increase in the usage of handwritten interactive apps on evaluations in online and distance education modes, which in turn resulted in the rise in demand for handwriting recognition.

In handwriting recognition tasks, mathematical expressions and formulas are common and significant components. Thus, converting mathematical formulas into markup language descriptions is a key issue in the hand-written recognition process. [1]. Typically, handwritten mathematical symbols are similar, causing problems with recognition.

Several classifiers and algorithms have been proposed recently for the recognition of handwriting. Some authors have used image processing and machine learning algorithms to segment and classify various symbols in the task of character recognition. However, some authors have begun to focus on deep learning algorithms because of their strength in computer vision applications.

In terms of accuracy, convolutional neural networks (CNNs) [2] are among the best methods for recognizing handwritten documents. CNN consists of two major components: 1) feature extraction, which is performed by the convolution layer, and 2) classification, which is performed by the last layer. Choosing a CNN network for feature extraction is not an act of randomness. Various networks have been proposed for the analysis of different types of data. Currently, these networks are used for feature extraction and are referred to as backbones. [3].

To the best of our knowledge, no publication has evaluated different CNN networks in the handwritten detection task. As a result, we aim to evaluate the performance of different CNN backbones in the classification of various symbols for the handwriting recognition task. We will then convert the recognized symbols into Latex files.

II. Literature Review

There are two types of optical character recognition (OCR): printed character recognition and handwritten character recognition. A printed character recognition system recognizes characters in an image of a newspaper, book, or another paper document. Handwritten recognition refers to the recognition of characters written by humans.

The recognition of handwritten characters can be divided into two types: online and offline. A system that performs offline character recognition takes an image as input and recognizes the characters and converts them into text and words. The process of online character recognition is a dynamic one that is more complex.

As part of the handwritten recognition process using Machine Learning (ML) algorithms, a variety of symbols have been recognized, including Devanagari characters [4] and numbers [5], Hindi numerals, South Indian script [6], and Gurmuckhi script [7].

The general flowchart of the ML-based handwritten methods has been shown in Fig 1 (a).

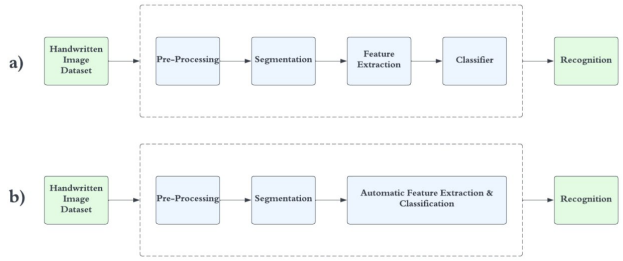

Fig. 1: The general flowchart of the handwritten recognition methods. (a) Machine Learning (b) Deep Learning

As shown in this flowchart, the pre-processing stage is the first step. The most common pre-processing methods include converting an RGB image to a grayscale image, removing noise, and binarizing [8] [9].

In the handwritten recognition process, the segmentation stage is the most critical and challenging. As part of the character recognition process, the image is segmented into lines, words, and characters[10]. When the image is broken, touching, overlapping, or degraded, and there are compound characters, the segmentation process becomes even more challenging.

Line segmentation is the initial step in text-based image segmentation. In this step, the document image is scanned pixel by pixel, left to right, and top to bottom, and the image is then segmented into different regions with each one representing a text line [11] [12].

Generally, line segmentation methods can be divided into top-down and bottom-up approaches. The top-down approach divides the document into text lines, words, and characters, recursively. It works well with documents with reasonably straight text lines since it assumes that the lines are straight. In the bottom-up technique, parts of the image, such as linked components, words, letters, and pixels, are combined to generate corresponding text lines. In this approach, the text line is not assumed to be a straight line as in the clustering process. Therefore, this approach is useful when there are lines with curved or overlapping text in a document. A hybrid technique combines both approaches in a variety of ways [10]. As a result of the bottom-up and top-down approaches, a number of strategies have been developed, and they can generally be classified as projection-based methods, smearing [13], grouping methods [14], Hough-based methods [15], and graph-based methods [16].

Several researchers have implemented word segmentation for a variety of languages. Kumar et al. [17] performed an experiment on ten handwritten Gurumukhi script documents and achieved an accuracy of 99.57 percent. In another experiment conducted by Ryu et al. [18], word segmentation was proposed as a binary quadratic problem, and the applied structured SVM framework was proposed for the handwritten dataset, and the accuracy was 92.82 percent [18].

The next step is to extract the features. ML-based methods require manual extraction of features. Features are measurable properties that are used to classify and recognize data, and later a classifier is used to categorize the data. Table 1 summarizes some of the recent methods for extracting features in the handwritten recognition.

Table 1 The feature extraction methods used in machine learning algorithms [19]

Ref no/year	Techniques used
2021 [20]	Histogram of oriented gradient and Zoning based density%
2019 [21]	Zoning, diagonal, and shadow feature
2019 [22]	Elliptical, Tetragonal and Vertical pixel density histogram
2018 [23]	Shape context

Handwritten recognition concludes with the classification stage. In order to recognize each symbol, the features extracted in the next step will be applied to a classification algorithm, such as SVM, KNN, or Naive Bayes.

Over the past few years, some authors have also explored the application of deep learning algorithms to handwriting recognition [Fig 1 (b)]. These methods use the convolutional layer to extract features automatically after segmentation. Many fields of visual research have utilized convolutional neural networks, including image classification, face recognition, audio retrieval, and ECG analysis. Due to convolution, pooling, and other operations, convolutional neural networks are so successful. As a result of local connection and weight sharing, the convolution operation effectively retains the spatial information of two-dimensional data, whereas the pooling operation is capable of meeting the translation invariance requirements for classification tasks [24].

A CNN is often used in handwritten recognition systems in order to recognize pictures of offline handwritten mathematical expressions. The authors of [25] have developed a Handwritten Character Recognition technique for Tifinagh, a language used in North Africa with the same punctuation as the Latin alphabet. There have been two deep learning approaches proposed, Convolution Neural Networks (CNNs) and Deep Belief Networks (DBNs), as well as the use of visual features. Altair et al. [26] proposed a CNN-based deep learning model for recognizing Arabic handwritten characters. As a first step, the Arabic dataset was created, which was compiled by children aged 7-12. There were 47,434 characters in the dataset, which were written by 591 different authors. Handwritten characters were recognized using a CNN model. A summary of recent papers on handwritten recognition can be found in Table 2.

Table 2 Accuracy results achieved for various scripts using Deep Learning [19]

Ref no/year	Script name	Techniques used	Accuracy
2021 [20]	Gurmukhi	CNN	99%
2021 [27]	Arabic	CNN	97%
2021 [28]	Swedish	CNN	97.12%
2021 [29]	Urdu Manipuri	Resnet 18	86%
2021 [30]	Digits	VGG16	93.2%

III. Dataset

The dataset that we will use in this project is from the Kaggle website `https://www.kaggle.com/datasets/xainano/handwrittenmathsymbols`. Over 100,000 image samples are included in this dataset, with 45x45 pixel sizes. The dataset includes the following symbols.

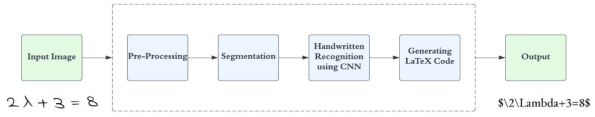

Fig. 2: The block diagram of the proposed method

- Basic Greek alphabet symbols such as alpha, beta, gamma, mu, sigma, phi, and theta
- English alphanumeric symbols
- Math operators, set operators
- Basic pre-defined math functions such as log, lim, cos, sin, tan
- Math symbols like: int, sum, sqrt, delta and more

IV. Discussion

First, we take a photo of a handwritten mathematical equation. Then we preprocess the given image using variety of computer vision algorithms and techniques. Then we perform connected component labeling to extract each symbol from the image. At the same time, we figure out where the symbol is on the given image. We feed the symbols into a convolutional neural network. Then these classifications along with each location goes into a LaTeX generator and that generates the LaTeX code. In this project, we have considered LeNet architecture as the convolutional neural network model. This project has three main parts. First, preprocessing the data and training the data with convolutional neural networks. Second Preprocessing the test image and extracting each symbol and finding the location of each symbol and classifying each symbol with the CNN model. Third, the handwritten mathematical equation should be mapped to the corresponding LaTeX code.

a. Dataset Preprocessing

The dataset that we used is from Kaggle website. It contains 82 different mathematical symbols such as numbers, English and Greek letters, letter-like symbols, operations, arrows, parentheses, and brackets which we have used only 33 mathematical symbols to train the model. The major problem with the dataset is that these images are too thin, that means it is very hard for pre-processing when we pre-process the real image it is very hard to make it thin. So, we did some preprocessing with the dataset itself by binarizing it and filling in small holes and dilating it.

1. Morphological Transforms

Morphological transforms are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of the operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient also comes into play.

Fig. 3: Image after erosion

Fig. 4: Image after dilation

2. Erosion

The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object. The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero). So, the pixels near boundary will be discarded depending upon the size of kernel. The thickness or size of the foreground object decreases or simply white regions decreases in the image. It is useful for removing small white noises and detach two connected objects.

3. Dilation

Dilation is opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So, it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So, we dilate it. Since noise is gone, they will not come back, but our object area increases. It is also useful in joining broken parts of an object.

b. Input Image Preprocessing

1. Converting the image to graye-scale

In image processing tasks, converting an image to gray-scale is often done for several reasons:

Simplification: Converting an image to gray-scale reduces the complexity of the image by reducing the color information from three channels (red, green, and blue) to a single channel (gray). This simplification can make image processing tasks, such as edge detection, thresholding, and segmentation, easier and faster to perform.

Contrast enhancement: In some cases, a gray-scale image can provide better contrast and detail than a color image. This is because color information can sometimes dis-

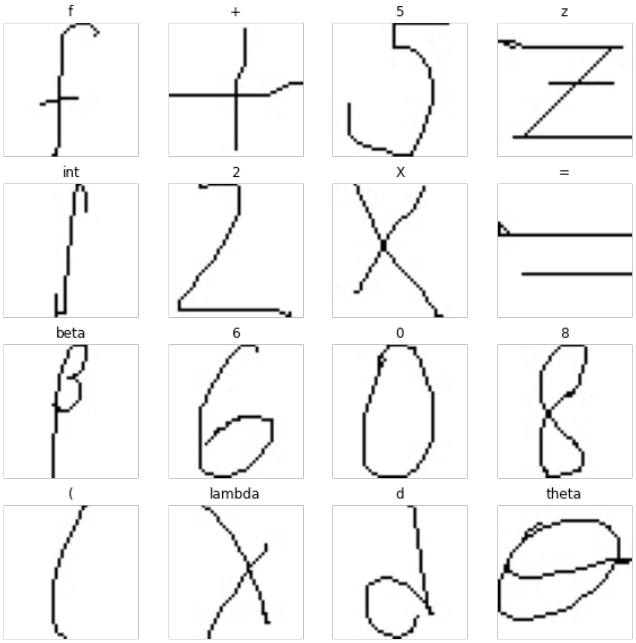

Fig. 5: Raw data

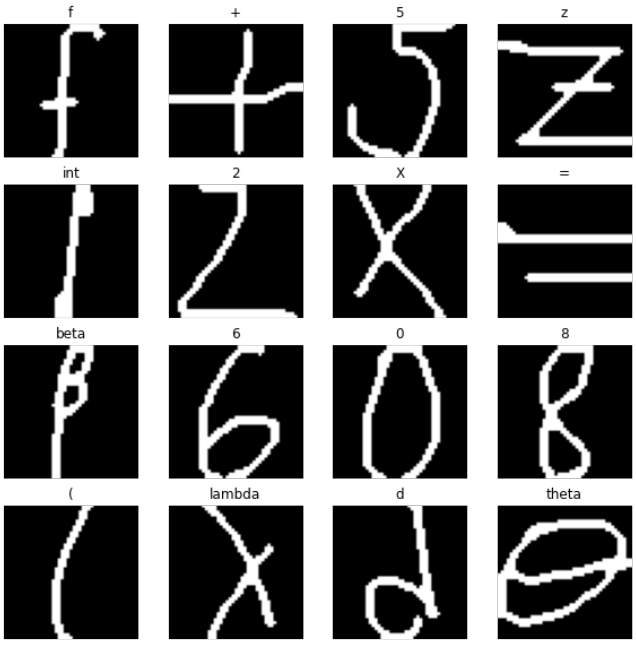

Fig. 6: Data after preprocessing

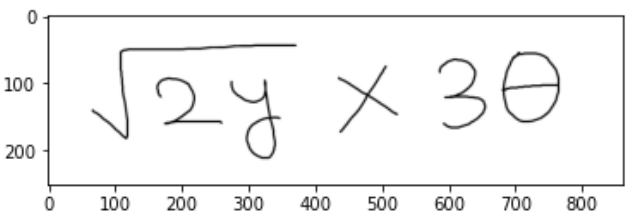

Fig. 7: Test Image

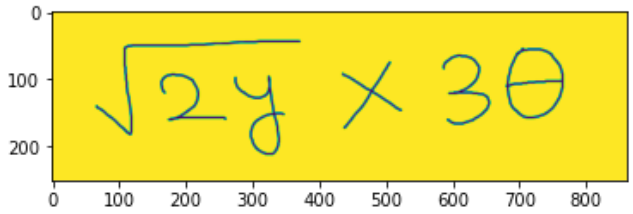

Fig. 8: Gray Scaled Image

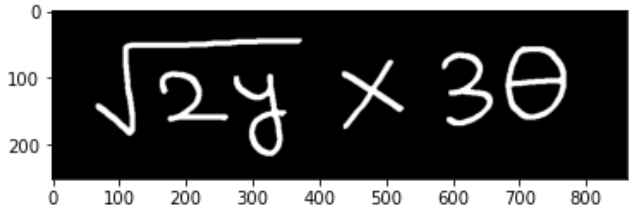

Fig. 9: Image after preprocessing (Noise removing + Binarizing + Dilating)

tract from important features in an image, while gray-scale images emphasize differences in brightness and darkness.

Memory and storage: Storing or processing a gray-scale image requires less memory and storage space than a color image, since it has fewer channels.

Compatibility: Many image processing algorithms are designed to work with gray-scale images, and converting an image to gray-scale can make it compatible with a wider range of image processing software and tools.

2. Removing noise using Gaussian Blur filter

As in any other signals, images also can contain different types of noise, especially because of the source (camera sensor). Image smoothing techniques help in reducing the noise. Image smoothing also called blurring could be done in many ways. Gaussian filters have the properties of having no overshoot to a step function input while minimizing the rise and fall time. In terms of image processing, any sharp edges in images are smoothed while minimizing too much blurring.

3. Removing Salt and Pepper noise using Median Blurring

This is a non-linear filtering technique. As clear from the name, this takes a median of all the pixels under the kernel area and replaces the central element with this median value. This is quite effective in reducing a certain type of noise such as salt and pepper noise with considerably less edge blurring as compared to other linear filters of the same size. Because we are taking a median, the output image will have no new values other than that in the input image.

4. Binarizing the image using Adaptive Threshold

If an image has different lightning conditions in different areas, adaptive thresholding can help. The algorithm determines the threshold for a pixel based on a small region around it. So, we get different threshold for different regions of the same image which gives better results for images with varying illumination

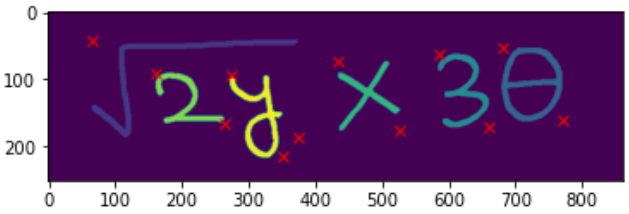

Fig. 10: Image after finding the location of each character

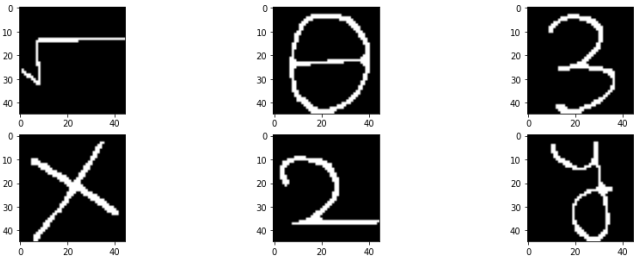

Fig. 11: Extracting each character from the test

5. Connected component labeling

Component labeling is basically extracting a region from the original image, except that we try to find only the components which are "connected" which is determined by the application of the graph theory. We often use connected component analysis in the situations that contours are used; however, connected component labeling can often give us granular filtering of the blobs in a binary image. When using contour analysis, we are often restricted by the hierarchy of the outlines. With connected component analysis, we can more easily segment and analyze these structures.

6. Finding the location of each character

c. Classifying the characters using Convolutional Neural Netwroks

1. LeNet Architecture

LeNet is a multi-layer convolutional neural network, used for image classification. In fact, it was used for text recognition and to classify text as handwritten or machine printed. LeNet is a pre-trained model, that incorporates transfer learning. Transfer learning involves training a model on a number of generalized enough dataset and using it on another model. LeNet is among the first published CNNs to capture wide attention for its performance on computer vision tasks. The model was introduced by Yann LeCun, for the purpose of recognizing handwritten digits in images. LeNet was eventually adapted to recognize digits for processing deposits in ATM machines. At a high level, LeNet consists of two parts. A convolutional encoder consisting of two convolutional layers and a dense block consisting of three fully connected layers. The architecture is summarized in the figure below.

2. Cross-Entropy Loss Function

The Cross-Entropy loss function is a commonly used loss function in machine learning, particularly for multi-class classification problems. It measures the difference between the predicted probabilities of the model and the actual probabilities of the target variable. The goal of training a machine

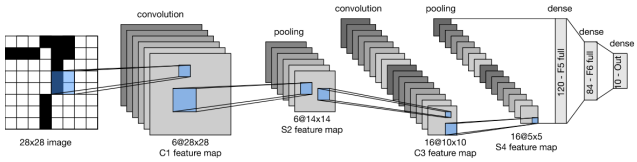

Fig. 12: LeNet Architecture

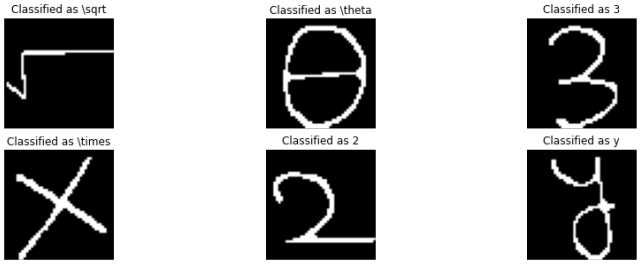

Fig. 13: Classified characters

learning model is to minimize the value of the loss function, which is achieved by adjusting the model's parameters during the training process. In practice, the Cross-Entropy loss function is often combined with an optimization algorithm like Gradient Descent to find the optimal parameters that minimize the loss.

3. Gradient Descent Optimizer

Gradient descent optimizer is a widely used optimization algorithm in machine learning, which is used to minimize the loss function during the training of a model. The basic idea behind gradient descent is to iteratively update the model's parameters in the direction of the negative gradient of the loss function, which is calculated with respect to the parameters. In other words, the gradient descent optimizer computes the gradient of the loss function with respect to the model parameters and then updates the parameters in the direction of the negative gradient to reduce the loss. There are different variants of gradient descent optimizer, such as batch gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent. In batch gradient descent, the optimizer computes the gradient of the loss function with respect to all training samples at once, while in SGD, the optimizer computes the gradient with respect to one sample at a time. Mini-batch gradient descent is a compromise between the two, where the gradient is computed with respect to a small batch of samples at a time.

4. Determining the Subscript and the Superscript

A subscript or superscript is a character (such as a number or letter) that is set slightly below or above the normal line of type, respectively. It is usually smaller than the rest of the text. Subscripts appear at or below the baseline, while superscripts are above. Subscripts and superscripts are most often used in formulas, mathematical expressions but have many other uses as well.

d. Generating the Corresponding LaTex Code

A mathematics typesetting program called LaTex is the standard for most professional mathematics writing. The pro-

gram is based on Donal Knuth's typesetting program Tex. This project will conclude with the generation of LaTex code after the symbols have been recognized.

```
Out[977]: '$ \\sqrt {2 y }\\times 3 \\theta $'
```

Fig. 14: Generated LaTeX code for the handwritten mathematical equation

V. Results

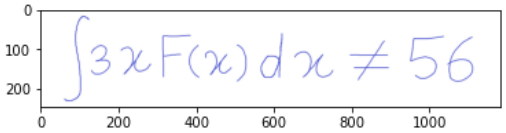

Fig. 15: Test Image

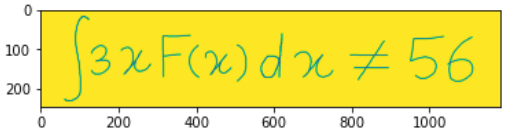

Fig. 16: Gray Scaled Image

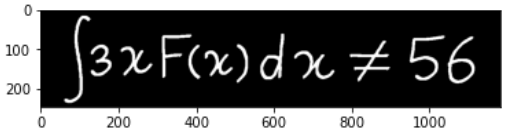

Fig. 17: Image after pre-processing (Noise removing + Binarizing + Dilating)

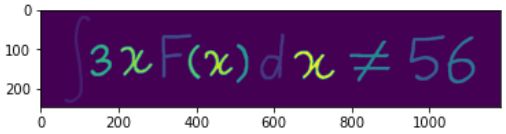

Fig. 18: Image after Connected Component Labeling

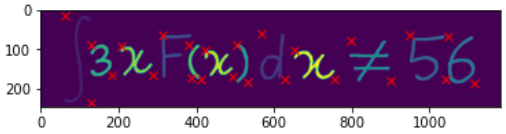

Fig. 19: Finding the location of each character

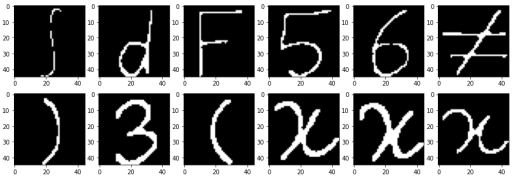

Fig. 20: Extracting each Character from the test image

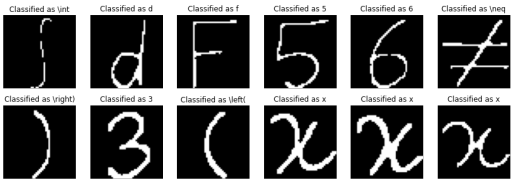

Fig. 21: Classified Characters

```
Out[114]: '$ \\int 3 x f \\left( x \\right) d x \\neq 5 6 $'
```

Fig. 22: Generated LaTeX code for the handwritten mathematical equation

VI. Conclusion

In this project we implemented a program which takes an image of a handwritten mathematical equation as an input. The first part of the project was to prepare the data and preprocess our given image. After receiving the image, we used various image processing and computer vision techniques to prepare our input image and data for the next steps. At the first place, it was needed to prepare the data. For this reason, we used morphological transforms which includes erosion and dilation. The next step was to preprocess the given image. Preprocessing involves converting the image to grayscale, removing noise using Gaussian blur filter and Median filter. Also, binarizing the image using adaptive threshold. Then, we used connected component labeling to extract each character from the image. The second part of this project was to classify our data using convolutional neural network classifier. We considered the LeNet architecture as the CNN architecture which consists of two convolutional layer, two max pooling layer and three fully connected layer. The third part of this project was to generate the corresponding LaTeX code. In the last part, we defined a function which takes the extracted components, their location and at the final stage generates the corresponding LaTeX code for the handwritten mathematical equation.

References

[1] H. V. K. S. S. P. R. P. Aryan Sharma, Ashwitha A Bhandary, "Generating latex code for handwritten mathematical equations using convolutional neural network," *Pattern Recognition Letters*, vol. 09, 2022.

[2] G. Yao, T. Lei, and J. Zhong, "A review of convolutional-neural-network-based action recognition," *Pattern Recognition Letters*, vol. 118, pp. 14–22, 2019.

[3] O. Elharrouss, Y. Akbari, N. Almaadeed, and S. Al-Maadeed, "Backbones-review: Feature extraction networks for deep learning and deep reinforcement learning approaches," *arXiv preprint arXiv:2206.08016*, 2022.

[4] S. K. Shrivastava and S. S. Gharde, "Support vector machine for handwritten devanagari numeral recognition," *International journal of computer applications*, vol. 7, no. 11, pp. 9–14, 2010.

[5] S. Rajashekararadhya and V. P. Ranjan, "Zone-based hybrid feature extraction algorithm for handwritten numeral recognition of four indian scripts," in *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2009, pp. 5145–5150.

[6] O. Rashnodi, H. Sajedi, and M. S. Abadeh, "Using box approach in persian handwritten digits recognition," *International Journal of Computer Applications*, vol. 32, no. 3, pp. 1–8, 2011.

[7] S. M. Mousavi, S. O. Shahdi, and S. A. R. Abu-Bakar, "Car plate segmentation based on morphological and labeling approach," 2011.

[8] M. B. Bora, D. Daimary, K. Amitab, and D. Kandar, "Handwritten character recognition from images using cnn-ecoc," *Procedia*

Computer Science, vol. 167, pp. 2403–2409, 2020, international Conference on Computational Intelligence and Data Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920307596

[9] S. S. Gharde, P. Baviskar, and K. P. Adhiya, "Identification of handwritten simple mathematical equation based on svm and projection histogram," 2013.

[10] S. Joseph and J. George, *A Review of Various Line Segmentation Techniques Used in Handwritten Character Recognition*, 01 2023, pp. 353–365.

[11] R. Santos, G. Clemente, T. Ing Ren, and G. Cavalcanti, "Text line segmentation based on morphology and histogram projection," *Document Analysis and Recognition, International Conference on*, vol. 0, pp. 651–655, 07 2009.

[12] N. Dave, "Segmentation methods for hand written character recognition," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 8, pp. 155–164, 04 2015.

[13] D. Brodic and Z. Milivojevic, "A new approach to water flow algorithm for text line segmentation," *Journal of Universal Computer Science*, vol. 17, pp. 30–47, 01 2011.

[14] M. Feldbach and K. Tönnies, "Line detection and segmentation in historical church registers." 01 2001, pp. 743–747.

[15] S. Saha, S. Basu, M. Nasipuri, and D. Basu, "A hough transform based technique for text segmentation," *journal of computing*, vol. 2, pp. 134–141, 02 2010.

[16] I. Abuhaiba, S. Datta, and M. Holt, "Line extraction and stroke ordering of text pages," 09 1995, pp. 390–393 vol.1.

[17] M. Kumar, R. K. Sharma, and M. K. Jindal, "Segmentation of lines and words in handwritten gurmukhi script documents," in *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia*, ser. IITM '10. New York, NY, USA: Association for Computing Machinery, 2011, p. 25–28. [Online]. Available: https://doi.org/10.1145/1963564.1963568

[18] J. Ryu, H. I. Koo, and N. I. Cho, "Word segmentation method for handwritten documents based on structured learning," *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1161–1165, 2015.

[19] S. Sharma and S. Gupta, "Recognition of various scripts using machine learning and deep learning techniques-a review," in *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE, 2021, pp. 84–89.

[20] S. Sharma, S. Gupta, and N. Kumar, "Holistic approach employing different optimizers for the recognition of district names using cnn model," *Annals of the Romanian Society for Cell Biology*, vol. 25, pp. 3294–3306, 01 2021.

[21] A. Garg, K. Jindal, and A. Singh, "Degraded offline handwritten gurmukhi character recognition: study of various features and classifiers," *International Journal of Information Technology*, vol. 14, pp. 1–9, 11 2019.

[22] S. Bhowmik, S. Malakar, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, "Off-line bangla handwritten word recognition: a holistic approach," *Neural Computing and Applications*, vol. 31, 10 2019.

[23] S. Sahoo, S. K. Nandi, S. Barua, Pallavi, S. Malakar, and R. Sarkar, *Handwritten Bangla City Name Recognition Using Shape-Context Feature*, 01 2018, pp. 451–460.

[24] H. Lin and J. Tan, *Application of Deep Learning in Handwritten Mathematical Expressions Recognition*, 10 2020, pp. 137–147.

[25] L. Sadouk, T. Gadi, and E.-H. Essoufi, "Handwritten tifinagh character recognition using deep learning architectures," 10 2017, pp. 1–11.

[26] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Computing and Applications*, vol. 33, 04 2021.

[27] ——, "Arabic handwriting recognition system using convolutional neural network," *Neural Computing and Applications*, vol. 33, no. 7, pp. 2249–2261, 2021.

[28] H. Kusetogullari, A. Yavariabdi, J. Hall, and N. Lavesson, "Digitnet: A deep handwritten digit detection and recognition method using a new historical handwritten digit dataset," *Big Data Research*, vol. 23, p. 100182, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214579620300502

[29] M. Kashif, "Urdu handwritten text recognition using resnet18," *arXiv preprint arXiv:2103.05105*, 2021.

[30] M. Zhao, A. G. Hochuli, and A. Cheddad, "End-to-end approach for recognition of historical digit strings," in *Document Analysis and Recognition – ICDAR 2021*, J. Lladós, D. Lopresti, and S. Uchida, Eds. Cham: Springer International Publishing, 2021, pp. 595–609.