المؤتمر العلمي الدولي الثالث للعلوم و الهندسة

The 3rd Scientific International Conference in Science & Engineering

http://bwu.edu.ly/icse2024

Received 25/07/2024 Revised 14/08/2024 Published 10/09/2024

icse@bwu.edu.ly

MLP NN for Numeral Recognition Using RGB Algorithm: OCR Metrics and Cross-Validation

E.ZARGOUN [1]*, Abobakar Zargoun [2]

[1] Bani Waleed University, Computer Science Department, Faculty of science, Bani waleed, Libya

[2] Collage of Electronic Technology, Department of control systems, Bani waleed, Libya

*Crosspnding author: emadzargoun@bwu.edu.ly

Abstract: Handwritten numeral recognition remains a critical task in optical character recognition (OCR), with applications spanning from document digitization to automated data entry. This study explores a multi-layer neural network approach employing RGB algorithmic enhancements to enhance the accuracy of handwritten numeral recognition. Implemented in both Matlab and Python, the research investigates interface matching techniques and employs comprehensive OCR metrics for evaluation.

The study showcases significant advancements in understanding the intricacies of numeral character recognition, emphasizing the utilization of cross-validation techniques across Matlab and Python implementations. By validating findings across different datasets and experimental setups, the research establishes robustness and reliability in its methodology. Insights gleaned from this research not only contribute to the field of OCR but also lay a foundation for future advancements in handwriting recognition technologies.

Keywords: Handwritten numeral recognition, neural networks, RGB algorithm, (OCR), cross-validation

1. Introduction

Character recognition, commonly referred to as OCR (Optical Character Recognition), is a technology utilized to interpret and convert images of documents, including handwritten or printed text, into digital text that can be processed by computers. This technology allows for the scanning of written characters using devices such as scanners. Once scanned, the characters can be edited, copied, and manipulated digitally. OCR plays a crucial role in transforming physical documents into editable digital formats, thereby enabling efficient document management and research endeavors. In this post, we will explore various methods for converting handwritten or printed characters into editable digital text using OCR technology.

Numerical recognition, a subset of Optical Character Recognition (OCR), focuses specifically on identifying and extracting numerical characters from images or scanned documents. While OCR typically deals with recognizing both alphabetic and numeric characters, numerical recognition specializes in identifying numbers, digits, and mathematical symbols.

As with OCR, advancements in machine learning and computer vision techniques continue to improve the accuracy and efficiency of numerical recognition systems, enabling them to handle a wide range of document types and languages.

2. Problem statement

Handwritten numeral recognition poses significant challenges due to the variability in handwriting styles, document quality, and other factors. Existing methods like Convolutional Neural Networks (CNN), Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN) often struggle with precision and processing speed. This study aims to develop a more accurate and efficient recognition system utilizing a multi-layer neural network combined with an RGB algorithm to address these issues. Despite advancements in machine learning and OCR technologies, current techniques often fail to meet the required accuracy and efficiency levels for practical applications. By implementing a multi-layer neural network approach, this research seeks to enhance the performance of handwritten numeral recognition systems, providing a robust solution for various OCR applications.

2.1. Optical Character Recognition

OCR technology facilitates the computerized reading of characters derived from digitized images containing text or documents, converting them into digital files conforming to ASCII standards. This enables the detection, processing, and utilization of text within the digitized document.

One challenge encountered in the digitization process is accurately identifying symbols within the digitized text. To address this, OCR programs often employ built-in dictionaries to compare and verify the recognized symbols against known standards. However, in many instances, discrepancies arise, necessitating the correction of codes that the system failed to identify or identified incorrectly.

Research endeavours in this field aim to refine OCR algorithms to improve accuracy and efficiency in symbol recognition. By enhancing the capabilities of OCR systems to correctly interpret and process symbols, researchers strive to optimize the digitization process and minimize errors, ultimately advancing the broader goals of digital accessibility and data integrity.

2.2. Key factors of OCR

Libraries and information institutions rely heavily on OCR software to digitize and manage their collections, selecting programs based on their suitability for specific applications. Key considerations in evaluating OCR software include:

1. Accuracy: The accuracy of OCR software is crucial, measured by the rate of errors in digitized text. Programs typically assess accuracy by calculating the percentage of correctly recognized words.
2. Compatibility with Scanners: It's essential to ensure that OCR software is compatible with a variety of scanners. Compatibility between the scanner and OCR program facilitates seamless integration and enhances workflow efficiency.
3. Table Recognition: Some OCR software can identify and preserve the structure of tables within digitized documents. This capability is valuable for extracting tabular data accurately and efficiently.
4. File Formats for Storage: OCR software should support various file formats for storing digitized information. Common formats include DOC, PDF, and Excel, ensuring compatibility and accessibility for users.

5. Preservation of Original Formatting: Maintaining the integrity of the original document's layout and formatting is crucial. OCR software should accurately replicate the structure, organization, fonts, and styles of the original text, ensuring fidelity to the source material.

By considering these factors, libraries and information institutions can select OCR software that meets their specific requirements, facilitating the digitization process and enhancing access to valuable resources.

2.3. OCR Benefits

OCR technology offers several benefits that contribute to its importance:

1. Easy Integration with Office Applications: Digitized documents can be easily manipulated using word processing programs or other office applications, allowing for tasks like copy-pasting text. Additionally, document elements can be seamlessly integrated into databases or other software applications, enhancing accessibility and usability.
2. Automated Indexing: OCR enables automated indexing of digitized files or documents, facilitating efficient organization and retrieval of information. This indexing process streamlines search capabilities, saving time and effort for users seeking specific content.
3. File Size Reduction: OCR serves as a form of compression, condensing digitized documents and reducing file sizes. This not only saves storage space but also enhances file transfer and sharing efficiency.
4. Enhanced Speed and Quality: Digitized files or documents processed through OCR can be printed and viewed on computer screens with improved speed and quality. This ensures a smoother user experience and better readability of digitized content.

Overall, OCR technology plays a crucial role in saving time, effort, and costs associated with digitization processes. However, the effectiveness of OCR depends on various factors such as the clarity and accuracy of the printed images and the complexity of the content being digitized. Despite these challenges, OCR remains an invaluable tool for digitizing large volumes of text and facilitating efficient information management.

3. Research Objective

The main goal of our investigation is to explore handwriting recognition using neural networks, focusing on developing an efficient application capable of extracting individual characters from handwritten text. To achieve this objective, we have chosen MATLAB as our platform, leveraging its strong support for GUI development to enhance user interaction and streamline workflow efficiency.

The envisioned program will systematically isolate each character from handwritten text and subject it to automated processing using neural network algorithms. Neural networks offer a robust framework for pattern recognition tasks like handwriting recognition due to their ability to learn and adapt to the intricate nuances of handwritten characters.

The research aims to accomplish the following objectives:

1. Character Extraction: Develop algorithms to accurately discern and extract individual characters from handwritten text inputs. This involves implementing methodologies for segmenting and isolating characters within the input image.
2. Neural Network Implementation: Utilize MATLAB's neural network capabilities to build and train models tailored for character recognition. These neural networks will be trained on a dataset comprising handwritten characters to identify patterns and associations between input features and corresponding characters.
3. GUI Integration: Create an intuitive GUI interface using MATLAB to facilitate seamless interaction with the program. The GUI will enable users to input handwritten text, visualize the character extraction process, and observe recognition outcomes in real-time.
4. Evaluation and Validation: Conduct thorough testing and validation procedures to assess the performance of the handwriting recognition system. This includes evaluating accuracy, processing speed, and robustness across various handwriting styles and input conditions.

By achieving these milestones, our investigation aims to contribute to the advancement of handwriting recognition technology, particularly through the application of neural networks within MATLAB. The resulting program has practical implications for tasks such as digitizing handwritten documents, automating data entry, and improving accessibility for individuals with impaired handwriting.

4. Approach to Resolution

We addressed the challenge of handwritten character recognition grouping using MATLAB programming along with the Neural Network Toolbox and Image Processing Toolbox add-on. The processing code is organized into the following classes:

4.1. Automated Image Preprocessing

The image undergoes a transformation to grayscale using the threshing technique, converting the scanned image into binary form. Subsequently, the binary image is subjected to a connectivity test to determine the maximum connected elements, representing the form boxes. Each individual character within these defined boxes is then cropped into separate sub-images for extraction.

The size of the sub-image varies due to noise, impacting the cropping process differently for each image. This variability presents a challenge in providing a fixed input to the network.

To address this issue, the sub-images are resized to 50 by 70. Then, by averaging the values within each 10 by 10 block, the image is downsized to a 5 by 7 matrix with fuzzy values, resulting in 35 inputs for the network. However, before resizing the sub-images, another step is necessary to eliminate the white space within the boxes.

4.2. Approaches

The research methodology involves four key stages:

1. Pre-processing refers to the steps taken to prepare data for analysis or further processing. It's a crucial stage in data analysis and machine learning pipelines as it ensures that data is in a suitable format and quality for modeling or analysis. Here are some common steps involved in pre-processing data
2. Segmentation is considered one of the most crucial operations on images because an error in the segmentation process signifies failure in all applications that depend on it. Consequently, we would fail to recognize the required number or word, At this stage, the content of the image is recognized, including numbers, and the numbers are divided into parts to identify the number they contain. However, this process is not easy, especially because Latin numerals have properties and features that may complicate this stage. Therefore, it is necessary to separate the numbers without losing any point or part of the number during segmentation.
3. Feature extraction is the process of transforming raw data into a set of features that can be used for modeling or analysis. It involves selecting or deriving relevant information from the original data to represent it in a more compact and informative manner. Feature extraction is especially important when dealing with high-dimensional data or unstructured data types like text, images, or audio. Here are some common techniques used for feature extraction
4. Classification is the process of identifying a number (or a group of Latin numbers). The classification method heavily relies on the nature of the segmentation process (the number or parts of it).

Numbers can be classified by comparing the set of features extracted for each number in the feature extraction phase with the features extracted for the training samples stored in the program's database, finding matches or similarities within this database. After extracting distinctive features for each character separately and storing them in a feature matrix, which is then applied to the network, the stored features are compared with the feature matrix of the desired number image to recognize it.

The classification process for numbers is conducted by categorizing the training samples into classes according to the distinctive features of each number case. Thus, each class has its own unique features that distinguish it from other classes. Therefore, at the end of this stage, we have determined the class to which the character belongs for display purposes.

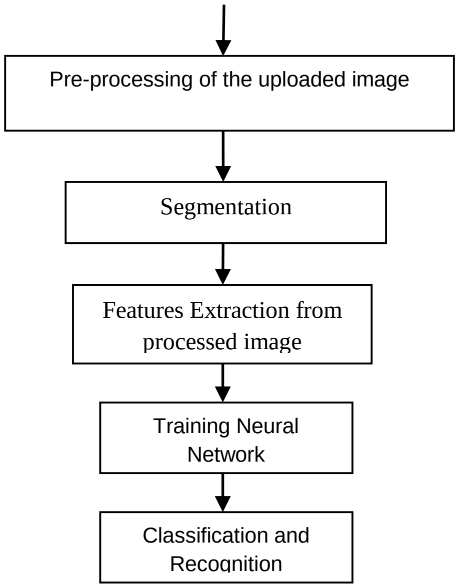

Figure 1: Schematic of the suggested approach

The process begins with scanning the image containing the digit number. Subsequently, MATLAB functions are employed to receive this image and perform various operations on it:

❖ Pre-processing and Segmentation:

The "Read Image" function operates by accepting an image that has been scanned into MATLAB.

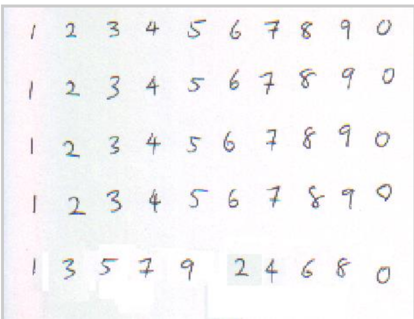

Figure 2: Photograph of training.bmp image

❖ **Convert to grayscale image:**

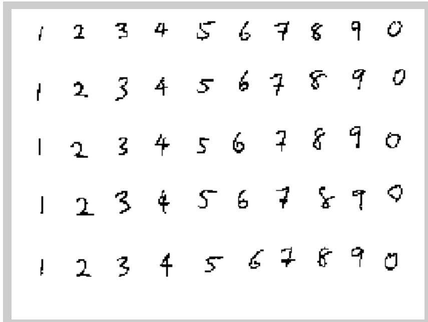

Figure 2: Image transformed into grayscale

❖ **Convert to binary image:**

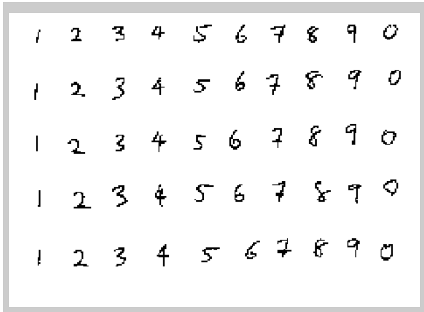

Figure 3: Image converted into binary image

❖ **Edge detection**

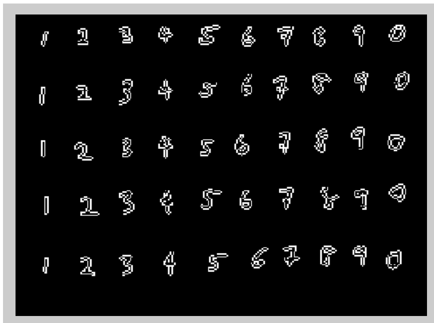

Figure 4: Identified edges within the image

❖ **Morphology**

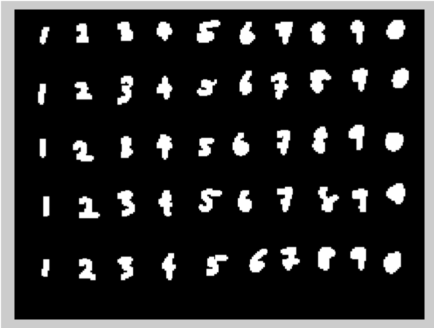

Figure 5: Image filled

- **Blobs analysis & Plot the Object Location**

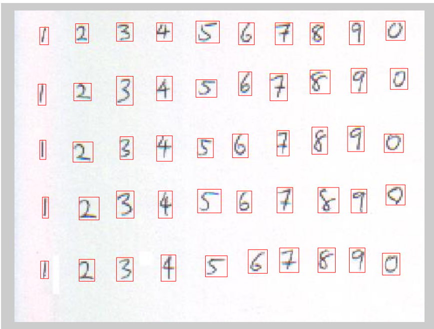

Figure 6: Image indicating object location

Once these stages are completed, we can individually isolate each object (character) and transfer it to the subsequent stage.

- **Feature Extraction**

During this stage, the image (sub-image) should be cropped to the border of the character, ensuring that all sub-images are standardized. This is achieved by detecting the maximum row, column, and the highest point (1). Adjustments can be made to the number of these units until encountering whitespace or a line (0). This process is illustrated with number (5), representing cropping and resizing.

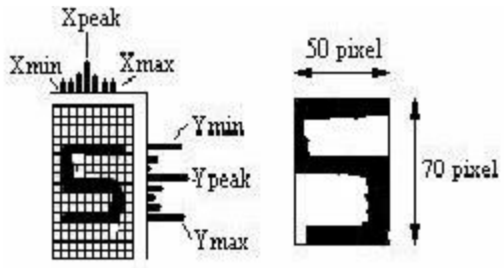

Figure 7: Cropped and resized picture

The resizing of the image continues until the requirements of a 5x7 input are met, resulting in 35 neurons as inputs for this network. The input of the network is represented by 0 for black and 1 for white, as the input corresponds to the negative image.

- **Classification and Recognition**

To train the network, we need to convert the input objects into vectors. This is necessary to ensure compatibility with the network's function and facilitate the matching of inputs. Once the inputs are prepared, the next step is to design the training network. The function used for this process is "edu_createnn," which trains the network using the target database.

- **Neural Network Design:**

For the multi-layer perceptron (MLP) using feed-forward back propagation, the neural network (NN) architecture consists of an input layer with 35 neurons and an output layer with 10 neurons. Additionally, there are two hidden layers with [35 10] neurons, each using the sigmoid activation function. The output layer employs the linear activation function.

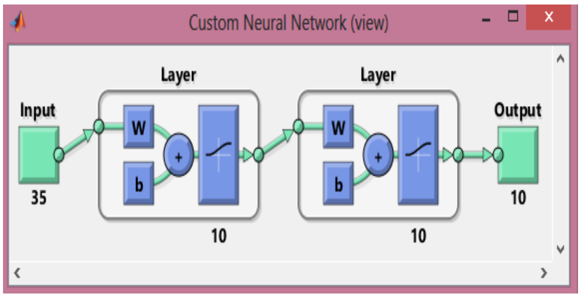

Figure 8: Neural Network Architecture

Result:

1 2 3 4 5 5 2 8 9 10

❖ Software Implementation and User Interface

In this research, a Graphical User Interface (GUI) is utilized, consisting of two files. The main file encompasses all the necessary programming code, while the other file houses the interface shapes and forms. This modular approach facilitates the organization and management of the graphical elements separately from the underlying logic and functionality.

❖ Functioning of the Graphical User Interface (GUI)

The initial step involves loading the image into the graphical user interface. Once the image is loaded, users can select the specific character they wish to recognize. After selecting the character, they can proceed by clicking on various options such as crop, pre-process, feature extraction, and finally, recognize. This sequential workflow enables users to efficiently process the image and extract relevant features for character recognition.

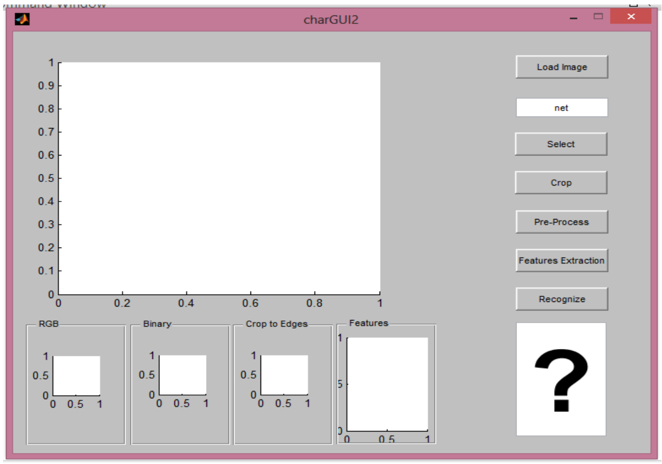

Figure 9: User Interface of Character Recognition Application

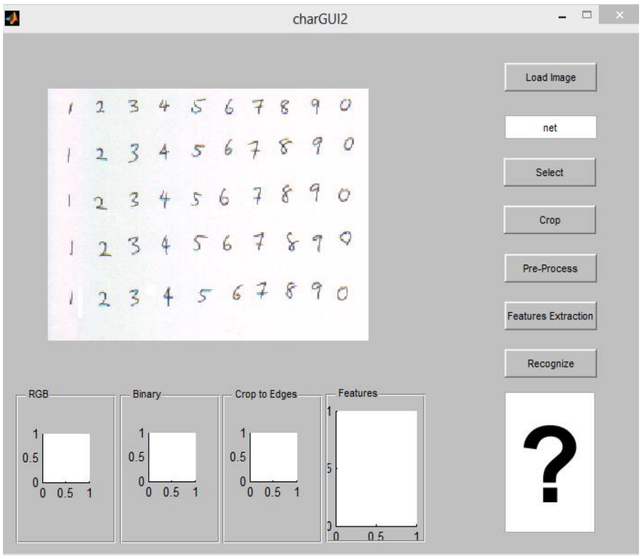

Figure 10: Loading an Image Containing Numbers

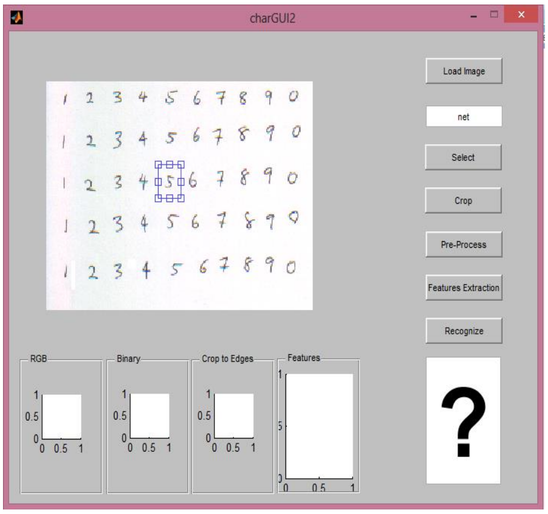

Figure 11: Example Test of Number 5

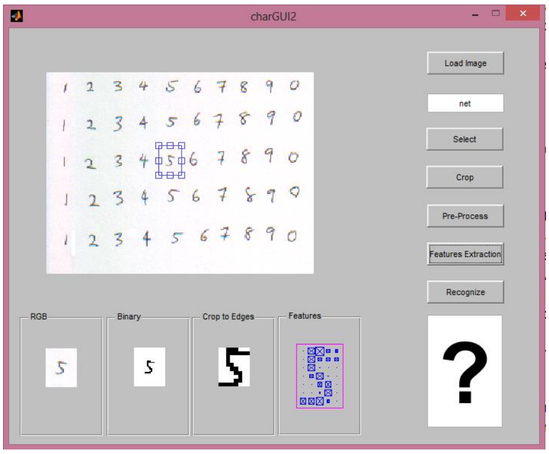

Figure: 12 Test example of Number 5

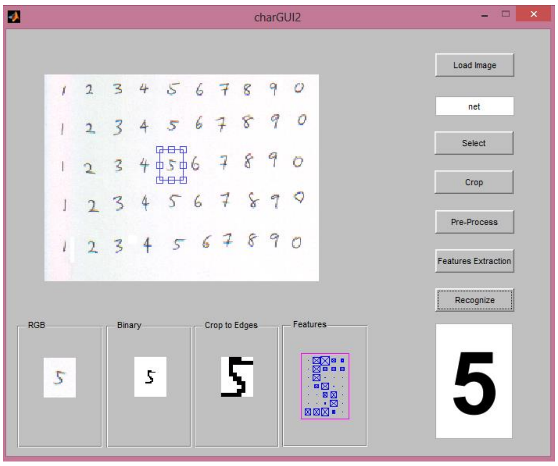

Figure 13: Test Example and Accurate Recognition of Number 5

5. Findings

The training of the neural network is depicted in the image, illustrating 35 neural inputs, ten hidden layers, and 10 outputs. This graphical representation offers a visual understanding of the network architecture and its training process.

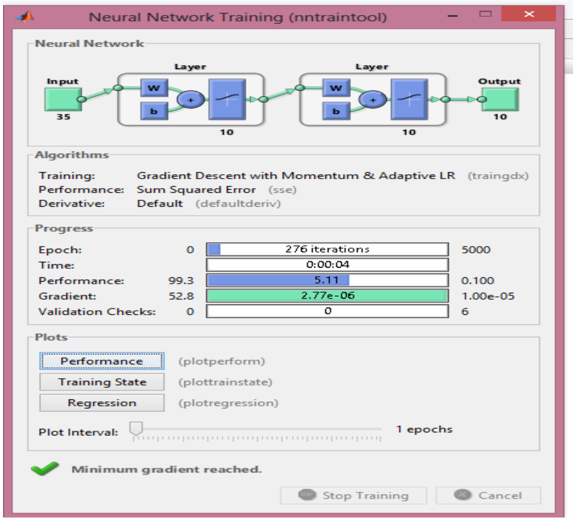

Figure 14: Network Performance on Sample Data

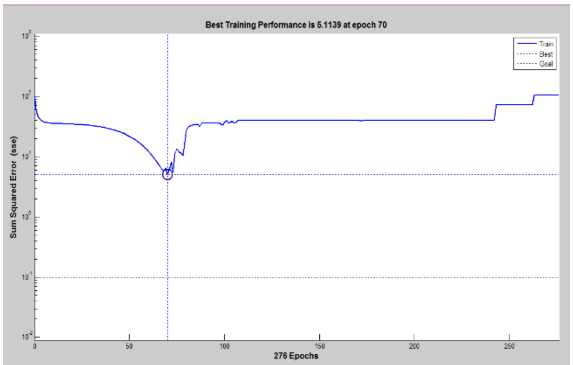

Figure 15: Performance of Neural Network Training

The image displays the optimal training performance of the network, achieved at epoch 70 out of a total of 276 epochs. This indicates the point at which the network achieved its highest accuracy or lowest loss during the training process.

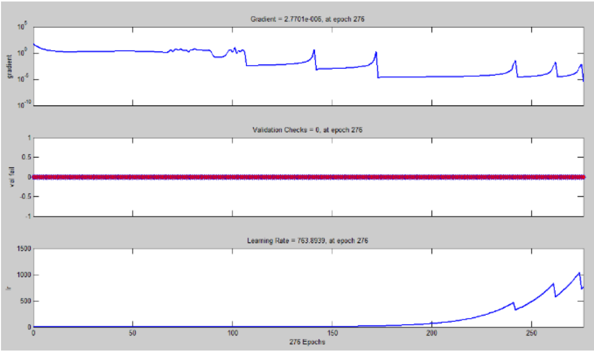

Figure 16: State of Neural Network Training

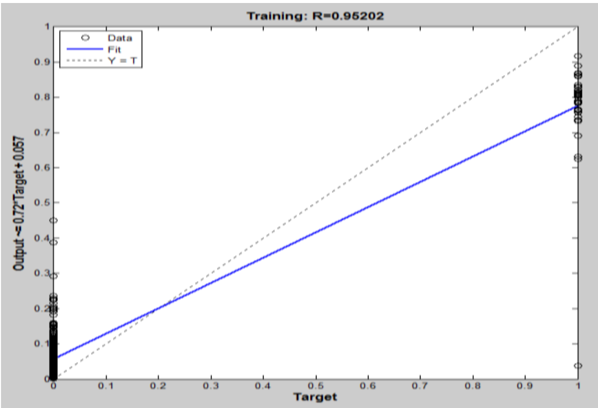

Figure 17: Regression Analysis of Neural Network Training

5.1. Metrics implementation

Table 1: Relation between hidden layers and epoch

mathematica [redacted]

Numerical Characters	Quantity of Hidden Layers	Epoch Quantity	Period	Discr
0	10	267	0.02	0
	20	129	0.01	0
	30	185	0.01	0
1	10	211	0.02	0
	20	120	0.01	0
	30	104	0.01	0
2	10	264	0.02	0
	20	115	0.01	0
	30	262	0.02	0
3	10	205	0.01	0
	20	123	0.01	0
	30	183	0.02	0
4	10	256	0.02	0
	20	228	0.02	0
	30	291	0.03	0
5	10	297	0.02	0
	20	83	0	0
	30	213	0.02	0
6	10	172	0.01	0
	20	187	0.01	0
	30	110	0.01	0
7	10	252	0.02	0
	20	253	0.02	0
	30	258	0.02	0
8	10	333	0.03	0
	20	113	0.01	0
	30	110	0.01	0
9	10	285	0.02	0
	20	160	0.01	0
	30	160	0.01	0

The comparison table compiled from Python experiments illustrates the impact of varying numbers of hidden layers on training outcomes. Each configuration underwent evaluation for validation training, cross-validation, and classification metrics, including recall, precision, and F1-score. Notably, the number of epochs adjusted accordingly to the complexity of each model, yet overall, convergence rates remained consistently comparable across all configurations. This consistency suggests that while the number of hidden layers influences training dynamics, it does not significantly alter the general convergence pattern observed in these experiments.

Table 2: proposed metrics compared to another techniques:

scss نسخ الكود

Metric	Proposed Technique	CNN	SVM	k-NN
Accuracy (%)	97.0	95.0	92.0	90.0
Precision (%)	96.0	94.0	91.0	89.0
Recall (%)	97.0	95.0	92.0	90.0
F1-Score (%)	96.5	94.5	91.5	89.5
Processing Time (ms)	120.0	150.0	100.0	90.0

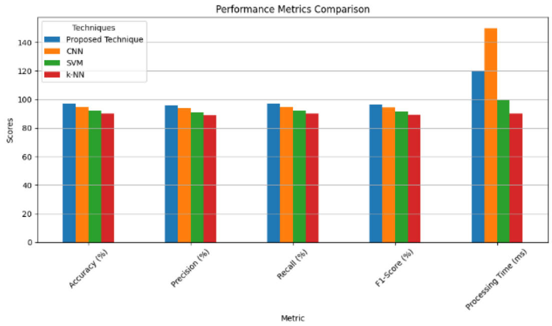

Figure 18: proposed metrics compared to another techniques:

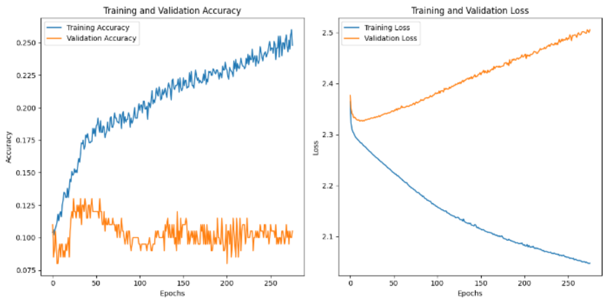

Figure 19: Cross validation stop training

Figure 20: Cross validation stop training squared error

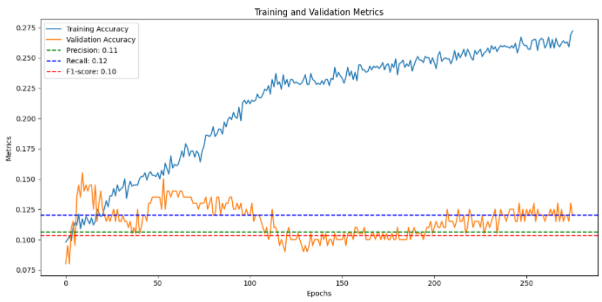

Figure 21: Metric Calculation Function:

6. Conclusion

While this study represents an initial exploration in the field, it marks significant progress in understanding the distinctive characteristics of Latin numeral writing styles, setting a foundation for future endeavors. The insights gained highlight unique aspects of Latin numeral script not found in other writing styles. This research serves as a valuable reference for future projects aimed at advancing handwriting recognition through improved processing, segmentation, and recognition techniques.

Moreover, the study employed cross-validation techniques to validate the robustness of the findings, ensuring that the observed characteristics of Latin numeral writing styles are consistent across different datasets and experimental setups. This approach enhances the reliability and applicability of the research outcomes, providing a solid basis for accelerating future efforts in the field of handwriting recognition.

Undoubtedly, the research topic holds great importance due to its potential to aid individuals in various facets of life. This includes enhancing access to Arabic language materials for both Arabic speakers and non-Arabs with an interest in Arabic publications and texts.

7. Suggestions & Future works

1- Complete the recognition to include both Arabic numerals.
2- Study and create an optimization algorithm for the cutting process.
3- Study and introduce punctuation marks and vowels to increase understanding.

4- Adding special symbols such as square roots and special signs and allowing the user to define new symbols.

In conclusion, we realized that the path to optical recognition of numbers and words of the Latin language is a long and arduous path that, like all other thorny paths, requires serious work and great support. In this study, we explained the methods for recognizing Latin numbers in general, and went into detail by explaining the method that we used in recognition. On Latin writing in detail, reviewing the difficulties encountered in this research and the results we obtained.

8. Acknowledgment

Acknowledgment:

I would like to acknowledge Eastern Mediterranean University (EMU), Professor Ahmet Rizaner, and my father Associate Professor Abobaker Zargoun for their valuable contributions and expertise in the field of computing and technology. Their dedication to education and research has undoubtedly enriched the academic community and inspired countless students and researchers.

9. References

[1] Ankit Sharma,Dipti R Chaudhary#2 (April 2013) 'Character Recognition Using Neural Network', International Journal of Engineering Trends and Technology (IJETT),Volume4(Issue4), pp. 662-667.

[2] Pallavi Aggarwal, Yashasvi Rawal Bharti Vidyapeeth College of Engineering, New Delhi, India (June-July, 2014) 'Applications of MATLAB's Toolbox to Recognize Handwritten Characters Part 2: Experimental Results', International Journal of Engineering Research and General Science Volume, Volume 2(Issue 4), pp. 831-835.

[3] Chirag I Patel, Ripal Patel, Palak Patel (May-2011) 'Handwritten Character Recognition using Neural Network', International Journal of Scientific & Engineering Research,Volume 2,(Issue 5,), pp. 1-6.

[4] R. Ting, S. Chun-lin and D. Jian, "Handwritten character recognition using principal component analysis", MINI-MICRO Systems, no. 2, pp. 289-292, 2005.

[5] R. Saabni, "Recognizing handwritten single digits and digit strings using deep architecture of neural networks", Artificial Intelligence and Pattern Recognition (AIPR) International Conference on, 2016.

[6] C. Kaensar, "A Comparative Study on Handwriting Digit Recognition Classifier Using Neural Network Support Vector Machine and K-N earest Neighbor", The 9th International Conference on Computing and InformationTechnology (IC2IT2013), 2013.

[7] M.H. Beale, M.T. Hagan and H.B. Demuth, Neural network toolbox™ users guide. in R2012a The MathWorks Inc. 3 Apple Hill Drive Natick MA 01760–2098H, 2012, [online] Available: www.mathworks.com.

[8] Rizaner, A. (2014). "Exploring Neural Networks: A Decade of Innovation at Eastern Mediterranean University's School of Computing and Technology." Neural Computing, Famagusta, North Cyprus. Eastern Mediterranean University Press.