Hittite Journal of Science and Engineering, 2021, 8 (2) 133–140
ISSN NUMBER: 2148–4171
DOI: 10.17350/HJSE19030000223

A Comparative Study of Handwritten Character Recognition by using Image Processing and Neural Network Techniques

Hakan Koyuncu
Altinbas University, Department of Computer Engineering, Istanbul, Turkey

ABSTRACT

This study aims to analyze the effects of noise, image filtering, and edge detection techniques in the preprocessing phase of character recognition by using a large set of character images exported from the MNIST database trained with various sizes of neural networks. Canny and Sobel algorithms are deployed to detect the edges of the images. The Canny algorithm can produce smoother and thinner continuous edges compare to the Sobel algorithm. The structural forms were reshaped using the Skeletonization algorithm. The Laplacian filter was used to increase the sharpness of the images and High pass filtering was used to highlight the fine details in blurred images in the form of image filtering. Gaussian noise or image noise with Gaussian intensity was used in Matlab on MNIST character images with the probability density function P. The effects of noise on character images are displayed during character recognition related to Neural network properties. Neural networks are commonly used to recognize patterns among optical characters. Feedforward neural networks are deployed in this study. A comprehensive analysis of the image processing algorithms is included during character recognition. Improved accuracy is observed with character recognition during the prediction phase of the neural networks. A sample of unknown numeric characters is tested with the application of High pass filtering plus feedforward neural network and 89% average output prediction accuracy was obtained against the average number of hidden layers in the neural network. Other prediction accuracies were also tabulated for the reader's attention.

Article History:
Received: 2021/02/08
Accepted: 2021/05/23
Online: 2021/06/30

***Correspondence to**: Hakan Koyuncu, Altinbas University, Computer Engineering, 34217, Istanbul, TURKEY*
E-Mail: hakan.koyuncu@altinbas.edu.tr

Keywords:
Artificial intelligence (AI); Edge detection; Feature extraction, Gradient; Hidden layer; Image correlation; Image filtering; Noise; Optical character recognition (OCR); Pattern recognition.

INTRODUCTION

Optical character recognition, (OCR), in the field of research where images of typed, handwritten, or printed text get converted into machine-encoded text [1,3]. Optical character data can be obtained from a photo of a document, from a scanned document, or directly from a photograph [4]. The recognition technology is extensively used for data entry such as recording of passport data, invoices, receipts, or any appropriate documentation [5]. Storing printed texts in digital format requires documents to be searched, displayed, edited, processed, and stored in various ways [6]. This may be achieved by using machine processing such as translation, semantic computing, text to speech conversion, and text mining. OCR is an area or research field, [7,8], which focuses on artificial intelligence, (AI), computer vision, and pattern recognition.

Initial versions of OCR required a preparation phase where character images are deployed for training to deal with individual fonts,[9]. A high level of identification accuracy is possible with most fonts due to the new capabilities of advanced systems,[10]. Some systems can produce an approximate format of the original document at the output, [11].

Matrix matching technique compares an image with a stored symbol image on a pixel-by-pixel basis, [12]. This is also identified as "pattern matching", "pattern recognition", or "image correlation". In this case, the input symbol is correctly isolated from the rest of the image and is in a similar font with the symbol stored on the same scale. This method fits well for the typewritten text and struggles as new fonts are introduced. This is the technique, which was implemented directly, [13], with the early photocell-based OCR equipment.

There are two types of OCR algorithm which can produce a list of possible characters. Matrix matching is associated with the comparison of a character picture to a stored character on a pixel-by-pixel basis. This technique depends on the fact that the input character needs to be accurately confined from the rest of the picture.

The technique of extraction of features breaks down symbols into "features" such as lines, closed loops, the direction of the line, and intersections of line,[14]. The extraction features lower the representation dimensionality and render the method of recognition computationally effective. These characteristics are compared to an abstract vector-like representation of a character, which could be reduced to one or more prototypes. Generic feature identification techniques applied to this form of OCR in computer vision are widely used in "intelligent" handwriting recognition, and in most modern OCR applications, [15]. Closest neighborhood classifiers like the k-nearest neighborhood (k-NN) algorithm are used to equate image features with stored character features and pick the closest fit,[16].

METHODOLOGY

Neural Networks

Artificial Neural Networks are inspired by how biological neurons process data. Neural Networks have caused a great amount of enthusiasm in the field of machine learning. It has caused a great amount of success, in areas such as image processing, autonomous driving, speech recognition, and character recognition. In this study, we try to understand the effects of how noise, image filters, and edge detection affect neural networks in the preprocessing phase.

The simplest computing element in a neural network is the neuron that often can be called a unit or node. This node takes information from a particular neuron or external source and determines an output. Every input has a corresponding weight (*w*) associated with it, which is appointed according to its relative influence on other inputs. The node as defined in Fig. 1 has a function, f, for the total weighted inputs.

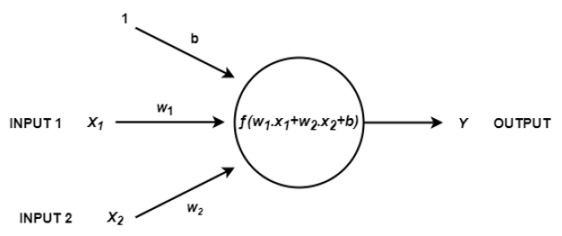

Figure 1. A single neuron.

The single neuron, shown above, receives numerical inputs X_1 and X_2. W_1 and W_2 weights correlate to inputs X_1 and X_2. An additional input 1 is also included with a bias weight b in the network. The primary task of bias, b, is to provide every neuron with a constant value in addition to the normal inputs, [17]. The Neuron output Y is determined as shown in Fig. 1. f is a non-linear function, named the Activation Function. The activation function aims at suggesting non-linearity to the neuron output. It is crucial, as real-world data is non-linear, and learning these non-linear representations would be desirable for the neurons. Each activation function takes an individual numeric value and executes a specific fixed numerical operation on it. There are a few activation functions in the literature. Fig. 2 below displays each of these activation functions:

Sigmoid

This function takes an input value and compresses it in a range of 0 and 1.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

Tanh

This function takes an input value and compresses it to the range between -1 and 1.

$$tanh(x) = 2\sigma(2x) - 1 \tag{2}$$

ReLU

This function is called Rectified Linear Unit, ReLU. To replace negative values with zero, it takes an input value and thresholds it.

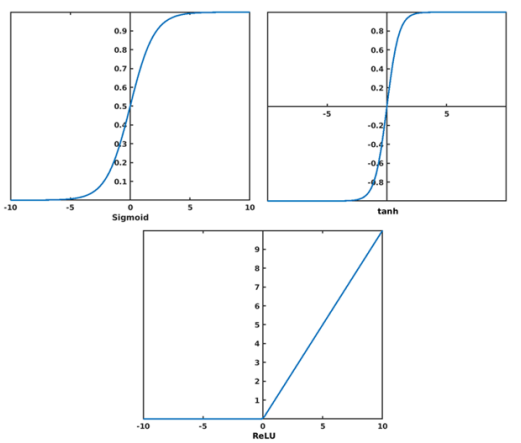

Figure 2. Different activation functions.

Feedforward Neural Network

The term Feedforward is used because the information travels forward in the network. Initially through the input nodes, followed by the hidden nodes, and finally ending at the output nodes. Feedforward networks are generally used for supervised learning where the data being trained is neither sequential nor dependent on time.

A single-layer neural network can be seen in Fig. 3 with S neurons and R number of inputs. Each of the R numbers of inputs is connected to each of the S number of neurons with a weight matrix of S rows. Inputs are p=p_1, p_2, p_3,…,p_R and each enters the network through weight matrix, W=($w_{i,j}$). With b=(bi) being the bias and the output can be written as a =f(W_p+b).

A neural network can have multiple layers. Each layer having its unique weight matrix W, network input vector n, bias vector b, and output vector a. The leftmost layer is called the input layer with the far right being the output layer. The layers that are in between are known as hidden layers. Depending on the task, multilayer networks can perform better compared to single-layered networks. For example, multilayer networks can be used to solve difficult complex problems such as object identification in an image. A single-layered network is only able to learn linearly separable patterns.

Fig. 4 shows a neural network with 2 hidden layers L_2 and L_3, with an output layer L_4 consisting of 2 outputs. For

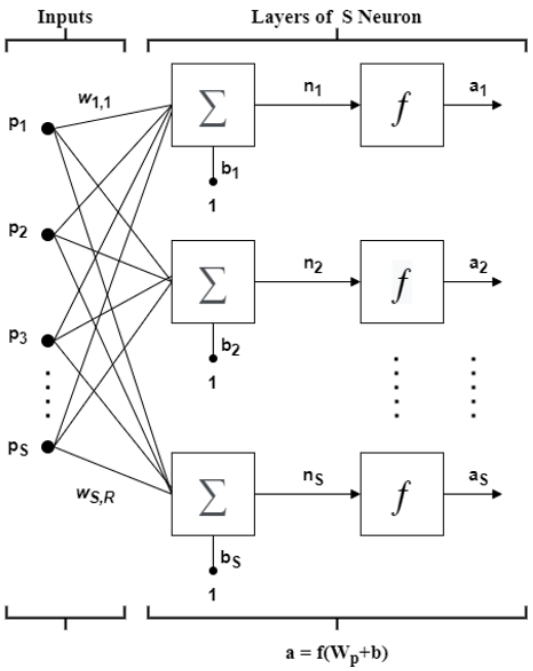

Figure 3. A single layer neural network.

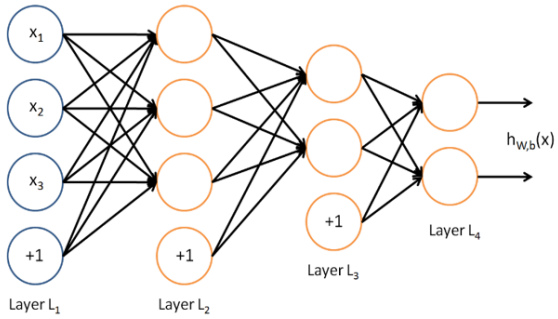

Figure 4. Multi-layer neural networks.

this network to be trained, it needs training samples (x_i, y_i) where $y_i \in R^2 \cdot R^n$ represents the n-dimensional Euclidean input space. This type of network is advantageous in cases where there is more than one output to be predicted. For example, a diagnosis application can use this network setup where vector x contains the patient to input information and output y can show the existence or absence of disease.

Canny Edge Detection Algorithm

Operator Canny is operating in a multi-stage process. Firstly, the Gaussian convolution smoothes the signal. A basic 2-D first derivative operator (like the Roberts Cross operator) is then added to the smooth image to illuminate image regions with the first strong spatial derivatives. In the gradient magnitude image, the edges give rise to ridges. Then the algorithm monitors around the top of these ridges and sets to zero all pixels that are not currently on the top of the ridge to make the output of a thin line, a method known as non-maximal suppression. There is hysteresis controlled by two thresholds in the tracking process: T_1 and T_2, with $T_1 > T_2$. Tracking can start only at a point higher than T_1 on a ridge. Tracking continues from that point in both directions until the ridge's height falls below T_2. Such hysteresis helps insure that rough surfaces do not break into several parts of the edge.

Sobel Edge Detection Algorithm

In image processing techniques Sobel operator is generally used in edge detection. The Sobel operator is convolving the image with a small, integer-valued filter which is applied vertically and horizontally. It is comparatively cost-effective in terms of computation-wise. In computation-wise Sobel operator is relatively cost-effective. Mathematically, the operator uses two 3x3 kernels convolved to the original picture to measure derivative approximations for vertical and horizontal shifts as seen in Fig. 5.

The center pixel of the masks is used for calculating the differences. The 3x3 convolution masks smoothen the image by some amount, hence it is less affected by noise. But it produces thicker edges.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 5. Sobel horizontal and vertical kernels.

Gaussian Noise

Gaussian noise is a statistical noise that is considered to have a probability density function similar to that of the standard distribution, commonly known as Gaussian. In other terms, Gaussian-distributed values are the values the noise will take on. A Gaussian random variable z has the probability density function **P** given by:

$$P_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \quad (3)$$

where z represents the grey level, μ the mean value, and σ the standard deviation. The main source of Gaussian noise in digital images appears during image acquisition. For example, sensor noise caused by poor illumination, high temperature, transmission, and electronic circuit noise. Gaussian noise can be reduced in image processing by using spatial filters to smooth out the image. An undesirable outcome of this smoothing operation may be the blurring of finer details, which correspond to the blocking of high frequencies.

Laplacian Filter

Discrete Laplace operator is often used in image processing for image enhancement purposes. The Laplacian filter highlights the areas of rapid intensity change in grayscale and hence it is used for edge detection. Derivative filters are sensitive to noise, and images are generally smoothed before applying the Laplacian filter. The Laplacian of an image, L (x, y), with I (x, y) pixel intensity values are given as:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (4)$$

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 6. Examples of Laplacian kernels (center with 4 is used in this study).

For two-dimensional signals, the discrete Laplacian approximations to the Laplacian filter can be given as convolution with the following kernels in Fig. 6.

High Pass Sharpening Filter

A high pass filter is used for highlighting fine details or enhancing the details that are blurred inside an image. A high-pass filter does the opposite of a low pass filter. As low-pass filtering smooths out noise, high-pass filtering amplifies the noise to bring out the details. If the original image is not too noisy applying a high pass filter shouldn't be a problem. For example, the Kernel in Fig. 7 shows minus signs for adjacent pixels. Meaning if there is no change in pixel intensity nothing takes place however if one pixel is brighter than its immediate neighbor, it gets boosted.

0	-1/4	0
-1/4	+2	-1/4
0	-1/4	0

Figure 7. High-pass filter kernel.

Skeletonization Algorithm

Representing the structural shape of a 2D region is to reduce it to a graph. This can be obtained by the skeletonization of a region via a thinning algorithm. Thinning algorithms can be used in various fields such as inspection of circuit boards, optical character recognition, etc. It is commonly used to trim the output of the edge detection by reducing all lines to a thickness of a few pixels. The Behavior of the thinning method is obtained through structuring elements as shown in Fig. 8. It makes a total of 8 structuring elements by 90° rotations (4×2).

Consider all pixels on the boundaries of foreground regions (i.e. foreground points that have at least one back-

0	0	0
	1	
1	1	1

	0	0
1	1	0
	1	

Figure 8. Skeletonization kernels.

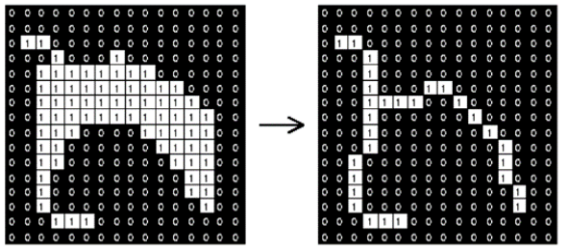

Figure 9. Thinning operation applied.

ground neighbor) in an image. Delete any such point that has more than one foreground neighbor, if doing so does not locally disconnect (i.e. split into two) the region containing that pixel. Iterate until convergence. Fig. 9 shows the result of this thinning operation on a simple binary image.

IMPLEMENTATIONS

Introduction to Neural Network

In this study, various preprocessing techniques have been applied to the MNIST dataset. The aim of applying these techniques is to observe the enhancements and accuracy in character identification. Adding information to the existing original image theoretically should improve the prediction of the neural network. Various methods were applied such as Laplacian transform for sharpening the images, Gaussian noise to observe effects of noise, and edge detection for re-representation of a character. The architectural representation of the neural network is presented in Fig. 10.

Edge Detection (Canny & Sobel) Application

Edge detection is a process used to detect and locate spatial discontinuities within the image. An Edge is identified as the finite area where the image contrast or intensity is largely changed. Edge Detection locates finite areas where there are high contrasts in intensity. Edge detection helps to collect details in an image, such as the shape, position, scale, detail sharpening, and enhancement of artifacts present in the image. Two of the most common

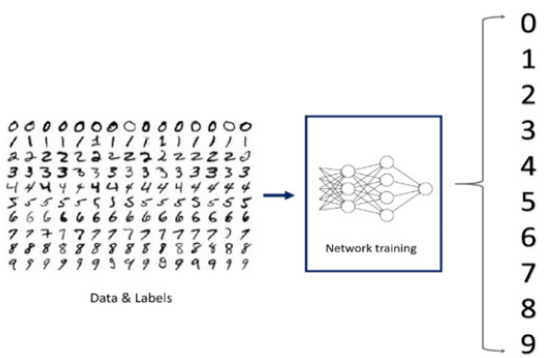

Figure 10. Neural network architecture.

Figure 11. From left a) The original image, b) Sobel edge detection, c) Canny edge detection.

algorithms for edge detection are implemented as shown in Fig. 11.

Sobel and Canny edge detection algorithms are applied to the original 28x28 pixel-sized MNIST images. Since it surrounds the entire character while performing edge detection and Sobel edge detection does not; Canny edge detection is considered for further calculations.

Canny was applied for the entire training MNIST dataset of 5000 number characters images with each has a size of 28x28 pixels. These images were used to train with their corresponding 10-bit labels. It was observed in Fig. 16 that the accuracy of an applied algorithm is measured as the percentage of true prediction with respect to total prediction. Canny edge detection produced an average of 84% accuracy while the Sobel algorithm has produced an 82.6% accuracy in character prediction. as seen in Fig.16 (blue and red color). Hence it was concluded that Canny was better than Sobel in terms of numeric character prediction.

Gaussian Noise Application

Gaussian noise, imnoise (image, 'gaussian', intensity) function in Matlab, with the probability density function P is applied on 5000 number character images of MNIST. During the implementation of Gaussian noise on images; Intensity values of 0.01, 0.1, 0.2, 0.3, 0.4 and 0.5 are deployed. Gaussian noise intensities create sets of varying noise intensity images. Each set of 5000 images were used for training purposes with their corresponding labels. The effects of additional noise in the predictions of feedforward neural networks are recorded. Various levels of Gaussian noise intensities are deployed for the training and prediction of the neural network.

The introduction of noise in character images was employed to show that the noise was an important factor. It degraded the images during testing and reduced the percentage accuracy. This can be predicted without repeating the test furthermore. It can be concluded that the addition of the noise to images would show a deterioration in all the applications. It was observed that as the amount of noise increases in character images, the accuracy of the neural network proportionally decreases. Examples of various Gaussian noise intensity levels can be seen in Fig. 12.

Figure 12. (a) Original İmage (b) Gaussian Noise 0.01 (c) Gaussian Noise 0.1.

The noisy image dataset is used to train the neural network and the resultant percentage accuracy level obtained was an average value of 75.2% as seen in Fig.16 (green color).

Laplacian Filter Application

Laplacian filter is a derivative operator. It emphasizes gray level disjunctions in an image and minimizes areas of slowly varying gray levels. By applying the Laplacian filter, images tend to obtain with grayish edge lines and other discontinuities that are all overlapped on a dark featureless background. Background details can be recovered while keeping the sharpening effect of the Laplacian operation by adding or subtracting the original and Laplacian images depending on the center coefficient. By applying the Laplacian image kernel with center value 4 in Fig. 6 on 5000 number character images of MNIST datasets, the Laplacian of these images is generated.

This can be identified as the enhancement of images. Fig. 13 shows the original image, Laplacian filtered image, and the combined image of an MNIST number character.

The combined image consists of the addition of original and Laplacian filtered images. This combined enhanced image dataset is used to train the neural network which results in average prediction accuracy of 87.4% in Fig. 16 (cyan color).

High Pass Filter Application

A high pass filtering technique is applied to see the sharpening effects on the image. Applying this filter amplifies the noise and brings the image details forward. A high pass filter kernel in Fig. 7 is applied to the character image and these images are convoluted with the kernel to generate a sharper image. 5000 images of number characters from the MNIST image database are high pass filtered. These high pass filtered images are later added

Figure 13. a) Original image, b) Laplacian filtered image, c) Laplacian filter + original image combined.

Figure 14. a) Original image (left), b) original image + High pass filtered image.

to the original image to increase the sharpening effect. These two combined images are fed into the feed-forward neural network for training purposes. A combined image of the high pass filtered image + original image is presented in Fig. 14.

At the end of the training stage, the unknown combined images of characters are tested with a feedforward neural network, and 89% average output prediction accuracy is obtained in Fig.16 (purple color).

Skeletonization Application

Skeletonization algorithm is applied to obtain the reshaping of structural shapes. The implementation of the Skeletonization algorithm yields a skeleton image. An original numeric character image is converted to a skeleton image as shown in Fig. 15. 5000 images of numeric characters are deployed from the MNIST database for skeletonization. The skeleton images of all the 5000-character images are obtained and they are inputted to the neural network for training. Once the training process is completed, at the end of the test stage for unknown characters, the prediction output accuracy is measured as 85.4% in Fig.16 (yellow color).

RESULTS AND DISCUSSIONS

Each application is carried out on 5000 MNIST images. These raw images are converted to resultant images by using different applications such as High Pass Filter, Gaussian Noise, Laplacian Filter, Canny, Sobel, Skeletonization algorithms. These resultant images are deployed to train a feedforward neural network with a different number of hidden layers for pattern recognition purposes in this study. The training of the network for each image type is the first step towards the quantization of prediction accuracy of unknown numerical characters. The number of hidden layers in a feedforward network

Figure 15. a) Original image, b) skeleton image.

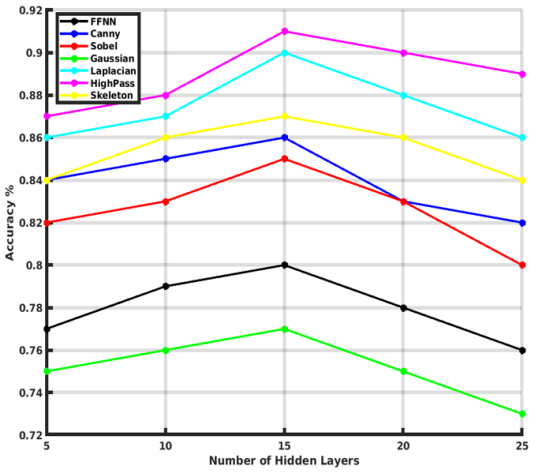

Figure 14. Output prediction accuracy of processed images for different applications versus the number of hidden layers.

where the input image data goes through is an important indicator for the output prediction accuracy of the neural network.

As the number of hidden layers changes, the image data which is going through the network also changes. Each of the affects the output prediction accuracy differently. In this study, 5 to 25 numbers of hidden layers are deployed. Optimum number of hidden layers was found to be 15 to obtain best prediction accuracies. Processed input image data are applied through these hidden layers. Hence, the prediction accuracies of different processed images through hidden layers are obtained by testing the neural network. Prediction accuracies versus the number of hidden layers are presented in Fig. 16.

It can be seen in Fig. 16 that high-pass filtering combined with the original image to increase the sharpening effect performs the highest output prediction accuracy of the neural network. The second-highest output prediction accuracy is obtained with the Laplacian filtering combined with the original image. Both application methods yield better results than the rest of the applications with the feedforward neural network. This is due to the extra enhancement, introduced with High pass and Laplacian filtering by combining the filtered image with the original image.

Edge detection algorithms, Canny and Sobel, both underperform compared to Laplacian and High pass sharpening filters. It is because edge detection algorithms only draw an outer contour of the character images. This causes a vast amount of information loss from the original data.

Canny seems to perform slightly better because of the contour drawing around the entire number character. Sobel, on the other hand, draws the contours with gaps. This small difference in edge detection techniques leads to a slight difference in the accurate prediction of the neural network trained with the Canny algorithm.

The skeletonization algorithm performs better than the edge detection algorithms. This can be possible due to the skeletonization algorithm consists of more pixel information compared to edge detection algorithms. Additionally, white pixels contribute to the learning process of the neural network. Having void regions with no information in images with applied edge detection does not contribute to the training process of the neural network. Hence accuracy prediction decreases.

Lastly, the neural network trained with added Gaussian noise to the character images performs the worst among the other application techniques. Training the neural network with noisy images leads to a worse performance compared to other applications which were predicted from the start. Hence, the prediction output accuracy is the lowest among the other applications.

CONCLUSION

It could be seen in this study that sharpening filters such as high pass and laplacian filters generated image enhancements of their original images before applied to the neural network. These enhanced images were later used to train the neural networks and give high prediction accuracy. In Literature, no study has been conducted on the comparison of different image processing techniques affecting output prediction accuracy of neural networks for character recognition.

A comparative study was proposed to transform the original MNIST images into resultant filtered images at the end of the filtering processes and use these images to train the neural network. The prediction accuracies for these image processing techniques on the character images are also investigated with respect to the number of hidden layers of the neural network.

The novelty lies in identifying the best pre-processing methods for input images to enhance them and in determining the optimum number of hidden layers in the Neural network.

The prediction accuracies are compared with respect to the number of hidden layers of the neural network. The prediction accuracy increases with sharpening filter effects and decreases with the inclusion of the noise. Unknown character images could also be exposed to similar High pass and Laplacian filtering and then applied as test inputs to the neural network to get the best prediction accuracies.

CONFLICT OF INTEREST

Author approve that to the best of their knowledge, there is not any conflict of interest or common interest with an institution/organization or a person that may affect the review process of the paper.

References

1. Yin Y, Zhang W, Hong S, Yang J, Xiong J, Gui, G. Deep learning-aided ocr techniques for chinese uppercase characters in the application of internet of things. IEEE Access 7 (2019) 47043-47049.
2. Yalniz IZ, Manmatha, R. Dependence models for searching text in document images. IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2009) 49-63.
3. Porat S, Carmeli B, Domany T, Drory T, Geva A, Tarem, A. Dynamic masking of application displays using OCR technologies. IBM Journal of Research and Development 53 (2009) 10:1-10:14.
4. Xu Y, Nagy, G. Prototype extraction and adaptive OCR. IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (1999) 1280-1296.
5. Vamvakas G, Gatos B, Stamatopoulos N, Perantonis, SJ. A complete optical character recognition methodology for historical document, in: The 8th IAPR International Workshop on Document Analysis Systems, Nara, pp. 525-532, 2008 .
6. Nikola G, Dragan M, Dejan, G. System For Digital Processing, Storage and internet Publishing of Printed Textual Documents, in: Fifth National Conference With International Participation ETAI'2000, pp. 21-23, 2000.
7. Kim MD, Ueda, J. Dynamics-based motion deblurring improves the performance of optical character recognition during fast scanning of a robotic eye. IEEE/ASME Transactions on Mechatronics. 23 (2018) 491-495.
8. Morns IP, Dlay, SS. Analog design of a new neural network for optical character recognition. IEEE Transactions on Neural Networks 10 (1999) 951-953.
9. Hamad KA, Kaya, M. A detailed analysis of optical character recognition technology. International Journal of Applied Mathematics, Electronics and Computers 4 (2016) 244-249.
10. Garris MD, Dimmick, DL. Form design for high accuracy optical character recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (199) 653-656.
11. Singh H, Sachan, A. A proposed approach for character recognition using document analysis with OCR, in: Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, pp. 190-195, 2018.
12. Kumar, PSJ. Adapted optimal neural network based classifier using optical character recognition engine for Tamil language. International Journal of Foundations of Computer Science 5 (2015) 30-37.
13. Budiwati SD, Haryatno J, Dharma, EM. Japanese character (Kana) pattern recognition application using neural network, in: Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, Bandung, pp. 1-6, 2011.
14. Pasha S, Padma, MC. Recognition of handwritten Kannada characters using hybrid features, in: Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013), Bangalore, pp. 59-65, 2013.
15. Kishna NPT, Francis, S. Intelligent tool for Malayalam cursive handwritten character recognition using artificial neural network and Hidden Markov Model, in: International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, pp. 595-598, 2017.
16. Chaudhari SA, Gulati, RM. An OCR for separation and identification of mixed English — Gujarati digits using kNN classifier, in: International Conference on Intelligent Systems and Signal Processing (ISSP), Gujarat, pp. 190-193, 2013.
17. Ghosn J, Bengio, Y. Bias learning, knowledge sharing. IEEE Transactions on Neural Networks 14 (2003) 748-765.