Turkish handwriting recognition system using multi-layer perceptron

Melih Kuncan[1], Enes Vardar[2], Kaplan Kaplan[3], H. Metin Ertunç[4]
[1]Department of Electrical and Electronics Engineering, Siirt University, Siirt, Turkey
[2, 3, 4]Department of Mechatronics Engineering, Kocaeli University, Kocaeli, Turkey
[1]Corresponding author
E-mail: [1]*melihkuncan@siirt.edu.tr*, [2]*enes_vrdr@hotmail.com*, [3]*kaplan.kaplan@kocaeli.edu.tr*, [4]*hmertunc@kocaeli.edu.tr*

Received 30 May 2020; received in revised form 31 August 2020; accepted 4 October 2020
DOI https://doi.org/10.21595/jmai.2020.21502

Abstract. Recently, handwriting recognition has found many application areas along with technological advances. Handwriting recognition systems can greatly simplify human life by reading tax returns, forwarding mail, reading bank checks, and so on. On the other hand, these systems can reduce the need for human interaction. Therefore, academic and commercial studies of handwriting characters have recently become an important research topic in pattern recognition. In this study, Turkish handwritten letter recognition system from A to Z was developed in C++ environment by using Artificial Neural Networks (ANNs). After the feature data were extracted, handwriting images were presented to the network, the training process of ANN was completed, and different handwriting images were classified with trained ANN. In this study, MLP (Multi-Layered Perceptron: MLP) type ANN and back-propagation learning algorithm were used. The ANN used has 35 inputs and 23 outputs. In the hidden layer, ANNs with different numbers of artificial neural cells (neurons) were evaluated and the most appropriate neural number ANN was selected. As a result, ANN with 24 neurons was selected in the hidden layer and handwriting images was classified with an accuracy rate of 94.90 %.

Keywords: artificial neural networks, handwriting recognition, image processing, back propagation algorithm, feature extraction, multi-layer perceptron.

1. Introduction

In general, artificial neural networks are parallel information processing systems developed inspired by the nerve cells of the human brain. This information is given to artificial neural networks after they are trained with examples of the related events. Thus, after the completion of the training process, these models can produce solutions to events that have never been encountered until then. ANN neurons are connected to each other with specified weight values and form artificial neural networks. By connecting artificial neural networks with weighted values, they gain skills such as the relationship between memory and data. Thus, it produces solutions for problems that require features of human nature, such as thinking, observing, and decision making [1].

Today, artificial neural networks are used in voice recognition [2, 3], license plate recognition [4], fingerprint recognition [5], fault recognition [6, 7], motion recognition [8], gender recognition [9] and many classification applications. Another research area that is one of the important recognition systems is handwriting recognition systems [10, 11]. Many handwriting recognition systems have been developed in the literature using artificial neural networks. One of these studies is the system that recognizes 33 handwritten characters of the Malayalam alphabet made by Alex M. and Das S. [12]. After Alex M. and Das S. divided the handwritten images into the regions they determined, they extracted the curvature and size features of those regions. Then, using these features, they developed a system that recognizes the handwriting with an accuracy rate of 89.2 % with the ANN. Asthana S et al. developed a single system that categorizes hand numbers from 0 to 9 written in Tamil, Telugu, Urdu, Devnagri and English alphabets used in India [13]. They used

5 different handwriting samples for each language while training the ANN. They used the back propagation algorithm while training the ANN model. A study similar to the work of Asthana et al. is a handwritten number recognition system developed by A. L. Mansoori [14]. In this system, he extracted 400 feature data by scanning 20×20 handwritten numbers. In training, he also used the back propagation algorithm. In this system, differently, he used an artificial neural network with a number of inputs 400, a single hidden layer and a 250-neuron numbers in the hidden layer. They trained ANNs with 5000 different handwritten number samples, and he realized a classification with an accuracy rate of 99.32 %. Yao and Cheng performed a Chinese handwriting character recognition system [15], Jayech et al. studied Arabic handwriting character recognition in their work [16], Marti and Bunke used English sentence database for handwriting recognition in their work [17], Yang developed handwriting character recognition for Russian capital letter [18], Espana et al. studied Spanish handwriting character recognition [19], Erdem and Uzun employed ANNs to develop Turkish handwriting character recognition system [20]. Carbune et al. proposed a new online handwriting recognition system based on deep learning methods, and they noted that the system's recognition accuracy compared to other studies improved by 20-40 % depending on the language when using smaller and faster models [21]. As seen from Alex M. and Das S. [12], Asthana S. [13] and A. L. Mansoori [14] and other studies, artificial neural network method has been used in many different languages for handwriting recognition operations. There are many parameters that affect the performance of the ANN method such as the hidden layer, the number of neurons in the hidden layer, the number of training samples, and the number of features.

In this study, a system has been developed for classifying Turkish handwritten images using ANNs. Since the image of the handwriting was created in a computer environment, the noise in the image was minimal and contributed significantly to the success of the study. Because noise is a parameter that negatively affects the success of work in many areas, especially image processing applications. For each handwriting targeted to be classified, 34 samples written in different ways were evaluated and their characteristics were extracted. With this feature data and back propagation algorithm, ANNs were trained to recognize different types of handwriting. The ANNs input layer is composed of three layers, a hidden layer and an output layer. Since 35 feature data were extracted in the feature extraction process, an ANNs with 23 outputs was created since it was aimed to classify 35 inputs and 23 letters. The number of neurons in the hidden layer is determined as a result of testing the hidden layer with different neuron numbers, and the most appropriate number of neurons for the system has been selected. As a result of this study, 23 Turkish handwriting character is recognized with 94.90 % accuracy.

2. Handwriting recognition system

This study consists of three main parts: image processing, feature extraction and construction of ANNs. The image of the handwriting in the image processing section is made ready for using in the training of the ANNs model for classification. In the feature extraction section, the extraction procedures of the handwritten character sample that will be used in training or classification of ANNs are performed.

2.1. Image processing

Many of the images require image processing algorithms before applying any recognition technique. The image processing algorithm is crucial for increasing accuracy in extracting important features for image recognition. The image processing steps in this study are explained with the steps as given below.

2.1.1. Transforming an Image to a Binary Image

In this section, images are converted from RGB space to gray space using C++ and OpenCV

software and the image is transformed to grayscale by calculating histogram density automatically with applied threshold value. n the binary image, all pixel values above the threshold value are 1 (white), and all pixel values below the threshold value are 0 (black). Pixel values in the digitized image are from 0 to 255 [22]. With this method, a picture can be divided into two parts, 1 and 0, with the threshold value. In order to perform the division process more successfully, the threshold value should be selected automatically by the system. For this reason, the Otsu Method was used with a library in OpenCV in this study. The Otsu method is a threshold determination method that can be applied for images with a gray level [23]. When this method is used, it is assumed that the image consists of two color classes as background and foreground. Then, for all threshold values, in-class variance values to which two color classes belong are calculated. The threshold value that enables this value to be the smallest is considered to be the optimum threshold value.

2.1.2. Finding multiple characters in an image

There can be more than one character in the image converted into a binary image with the threshold method. As a first step, these characters need to be separated from the image and evaluated as a single image for each character. The first step is to find boundary lines with the "findContours" function using the C ++ OpenCV library [23]. The boundary lines found are stored in an array with two memories. The first memory of the array contains the object, and the second memory contains the points of the boundary lines in the object. With a simple for loop, object boundary points are taken from the array in sequence, and with the OpenCV "rectangle" function, the smallest rectangles surrounding these boundary lines are found as in Fig. 1 [23]. Then, this rectangle found is cropped sequentially from the left of the image, and each cropped rectangle is evaluated as an image to be classified.

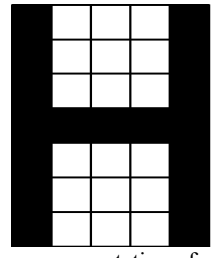
Fig. 1. Image separated into character groups

2.1.3. Image standardization

The final stage in image processing is the standardization of the image. The reason why this process is important is that the character data of differently sized characters can be compared and reduced to the same size in order to be recognized. Thanks to this algorithm, the image to be classified in different sizes is standardized in pixel dimensions (5×7).

2.2. Feature extraction

The feature extraction approach used in this study is to represent the character as a matrix, which is one of the classical methods used in many character recognition systems [13, 14, 17]. The matrix consisting of a 5×7 binary number of the letter 'H' is shown in Fig. 2.



1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1

Fig. 2. Binary representation of a smooth H character with a 5×7 matrix

It is possible to encode the character as a binary number so that each pixel corresponding to

black pixels is "1" on the picture and other pixels are "0". With this matrix, a 35-bit binary number written as a single vector is obtained. Using this number as input data, training and testing of ANNs are performed. For this reason, the input number of the ANNs used in the study is chosen as 35.

2.3. Artificial neural networks (ANNs)

Artificial neural networks can be expressed as parallel and distributed information processing elements, which are inspired by the human brain, are connected to each other with the help of links with weight values. Each neurons consists of process elements with its own memory. ANNs can be described as computer programs that simulate biological neural networks. ANNs are capable of self-learning [1-5, 24-26]. These networks have the ability to learn, memorize and create relationships between the information. Although there are many types of ANNs in the literature, some of them are more common to use than others. The most widely used ANNs type is known as MLP (Multi-Layered Perceptron: MLP) [12-14, 24, 25]. Due to its widespread use and successful performance, in this study, we used a MLP model. The ANN used in this study, as shown in Figure 3, consists of 3 layers, the input layer, a hidden layer and the output layer. Neurons in the input layer are responsible for sending the input information to the hidden layer, and the neurons in the output layer are responsible for processing the information from the hidden layer as the output of the model [1, 26].

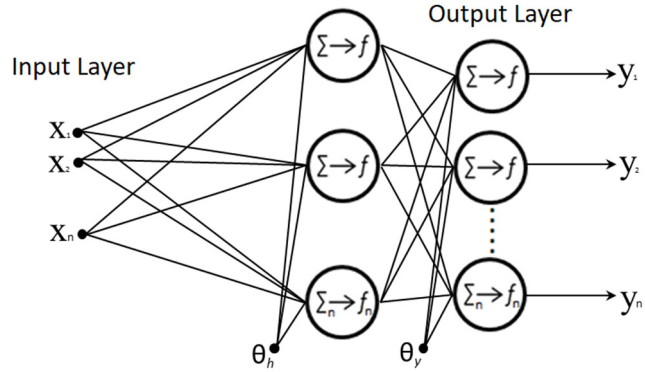


Fig. 3. A multi-layered ANN

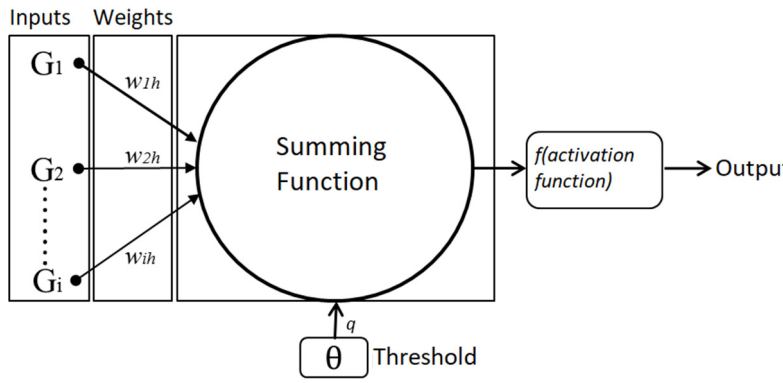


Fig. 4. A typical ANN neuron

The basic unit in ANN is the neuron shown in Fig. 4, which contains the process elements and the nodes between them. The inputs indicated by G_1, G_2,…, G_i are the inputs of neurons. The weight values of the inputs and the weight value of the threshold unit are indicated by w, q

respectively. Weight values indicate the effectiveness of input data in ANN. If these weight values are too large, that node is important or strong ties, and small means that it is weakly tied or less important [1]. The summing function shown in Eq. (1) calculates the total weight values of the inputs to the neuron. On the other hand, the activation function calculates the output value of the neuron using the exact information found with the summing process values [1, 26, 27]:

$$Net = \sum_{h=1}^{n_s} \theta_q + G_i w_i. \quad (1)$$

The activation functions have different representations as linear function, step function, sine function, threshold value function, sigmoid function and hyperbolic tangent function [28]. Generally, Sigmoid Function is preferred in many recognition systems [12, 27]. Sigmoid function is a limited and nonlinear function. The output values of the sigmoid function vary between 0 and 1. The equation of the sigmoid function is shown as in Eq. (2) [28]:

$$f(Net) = \frac{1}{1 - e^{(-Net)}}. \quad (2)$$

The ANN created in this study has 35 inputs and 23 outputs and a single hidden layer. The number of neurons in this hidden layer affects the accuracy of the model and its operating speed. Therefore, in order to select the appropriate number of neurons in the hidden layer, ANNs with different numbers of artificial neural cells were evaluated and the results were compared.

2.3.1. Back propagation learning rule

The learning ability of ANN depends on the weight values in the nodes to be adjusted to an appropriate value with the learning algorithm selected. Looking at the literature, the back propagation algorithm is quite common and has been used in many systems [12-14, 29]. In this learning rule, data is given to the input of ANNs and an output set is created. The resulting output set is compared with the desired output set and error values are found. The weight values are adjusted according to the desired output set by keeping this error value equal to the specified error criterion or by keeping a specified iteration number. In this study, it is determined that 26000 iteration numbers are suitable as a result of various trials. The algorithm is applied up to the specified number of iterations and weights are updated and the total error in the output values generated by the desired output is reduced.

In Table 1, when the property data of character images are applied to ANN, the desired output value is given. In the study, each node was assigned random weight values, this weight was adjusted with the back propagation algorithm, and the output values in Table 1 were targeted. The back propagation algorithm consists of seven steps as follows [1, 28, 30], where i – number of ANN inputs $i = 1, 2,\ldots, 35$; k – number of ANN outputs $k = 1, 2,\ldots, 23$; u_k – the desired value at output k; n_s – number of neurons in hidden layer; v_h – output value of hidden layer in h neuron $h = 1, 2,\ldots, n_s$; q_h – weight values between the neurons in the hidden layer and the threshold values; q_k – weight values between neurons in the output layer and threshold values; w_{ih} – weight values between input layer and hidden layer; w_{kh} – weight values between the output layer and the hidden layer.

Step 1: Learning coefficient (n) is determined. Since the ANN learning coefficient performs well for values between 0.1 and 0.3, the learning coefficient of 0.25 was chosen in the study [30].

Step 2: All weights are randomly assigned small values between 0 (zero) and 1 (one).

Step 3: The study contains the output value for 34 samples of each handwritten letter.

Step 4: Calculate the error term (δ_k) for each k output unit as below:

$$\delta_k = y_k(1 - y_k)(u_k - y). \quad (3)$$

Step 5: As seen in Eq. (4), the error term (δ_h) is calculated for each neuron in the interlayer:

$$\delta_h = v_h(1 - v_h)\sum_{h=1}^{n_s}\sum_{k=1}^{23} w_{kh}\delta_k. \quad (4)$$

Step 6: Weight values are updated for each network connection:

$$w_{ih} = w_{ih} + n\delta_h x_i, \quad (5)$$
$$w_{ik} = w_{ik} + n\delta_k v_k, \quad (6)$$
$$q_h = q_h + n\delta_h, \quad (7)$$
$$q_k = q_k + n\delta_k. \quad (8)$$

Step 7: Iteration count is checked. If the number of iterations is equal to 26000, the training is completed. If not, it is returned to Step 3 and these cycles are repeated until the number of iterations meets the 26000 condition.

Table 1. Desired output values for the characters to be classified

Output	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	T	U	V	Y	Z
u_1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u_{10}	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
u_{11}	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
u_{12}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
u_{13}	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
u_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
u_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
u_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
u_{17}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
u_{18}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
u_{19}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
u_{20}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
u_{21}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
u_{22}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
u_{23}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

2.3.2. Determining the number of neurons in the hidden layer

One of the most important steps in this study is to determine the number of neurons in the ANN hidden layer. Because the determining factor in an ANN performance is the number of neurons in the hidden layer. Although this number is generally between the number of entrances and the number of exits, there is no strict rule in determining the number of neurons. It is known that the increase in the number of neurons in general increases the ANN memory and increases the training time due to the processing load [1, 28]. However, this can vary from system to system. Therefore, in this study, the number of neurons in the hidden layer was determined by trial and error method considering the performance of the algorithm.

In this study, ANNs with different numbers of neurons in the hidden layer of 5, 10, 15, 20, 24,

27, 33, 37 were tested. ANNs with more than 37 neurons are not taken into consideration because the character classification process gives inadequate results. The evaluation was made by creating an MLP type ANN in C ++ environment with 26000 iteration conditions and 0.25 learning coefficient, and trained with back propagation learning algorithm. As training data, images containing 782 handwriting characters for 23 letters, 34 examples from each letter were used. These training data were given from letter A to letter Z, 34 samples for each iteration ANN and weights were updated. In Table 2, after ANN training process with different neuron numbers performed in C++ environment, classification performance is given for 34 training sets. Performance was determined by finding the ratio of the number of samples that could not be classified correctly for each training set to the number of samples in the training set (number of training samples in one set = 23). The error rates for ANN with different neuron numbers in the hidden layer are given in Table 2. As seen in Table 2, ANN with 5 neuron numbers in the hidden layer has the biggest error rate compared to ANNs with other neuron numbers.

Table 2. Error rates due to the number of neurons in the hidden layer

	$n_s = 5$	$n_s = 10$	$n_s = 15$	$n_s = 20$	$n_s = 24$	$n_s = 27$	$n_s = 30$	$n_s = 33$	$n_s = 37$
Set 1	21 %	0 %	3 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 2	21 %	0 %	3 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 3	12 %	3 %	3 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 4	24 %	3 %	3 %	6 %	0 %	0 %	0 %	3 %	0 %
Set 5	15 %	6 %	3 %	3 %	3 %	0 %	3 %	3 %	0 %
Set 6	12 %	3 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 7	24 %	3 %	6 %	0 %	0 %	3 %	0 %	0 %	0 %
Set 8	24 %	6 %	3 %	3 %	3 %	3 %	0 %	6 %	3 %
Set 9	24 %	6 %	0 %	0 %	3 %	0 %	0 %	0 %	0 %
Set 10	21 %	3 %	3 %	0 %	3 %	0 %	0 %	0 %	0 %
Set 11	15 %	3 %	6 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 12	18 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 13	18 %	3 %	0 %	0 %	3 %	0 %	0 %	0 %	0 %
Set 14	27 %	6 %	6 %	3 %	3 %	3 %	3 %	3 %	0 %
Set 15	39 %	3 %	6 %	9 %	3 %	6 %	3 %	3 %	3 %
Set 16	27 %	3 %	0 %	6 %	3 %	0 %	0 %	0 %	0 %
Set 17	24 %	0 %	3 %	3 %	0 %	0 %	0 %	0 %	0 %
Set 18	21 %	3 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 19	15 %	3 %	3 %	6 %	0 %	3 %	0 %	0 %	0 %
Set 20	21 %	0 %	0 %	3 %	0 %	0 %	0 %	0 %	0 %
Set 21	30 %	3 %	6 %	3 %	0 %	0 %	0 %	0 %	0 %
Set 22	18 %	6 %	6 %	3 %	6 %	0 %	6 %	6 %	3 %
Set 23	21 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 24	12 %	0 %	6 %	6 %	9 %	0 %	0 %	0 %	0 %
Set 25	12 %	0 %	0 %	0 %	3 %	0 %	0 %	0 %	0 %
Set 26	12 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 27	21 %	9 %	9 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 28	18 %	3 %	6 %	3 %	3 %	0 %	0 %	0 %	0 %
Set 29	21 %	3 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 30	12 %	0 %	3 %	0 %	0 %	0 %	0 %	0 %	0 %
Set 31	30 %	9 %	3 %	3 %	9 %	6 %	3 %	0 %	0 %
Set 32	36 %	12	6 %	6 %	12	6 %	0 %	6 %	0 %
Set 33	18 %	9 %	6 %	0 %	3 %	0 %	3 %	3 %	0 %
Set 34	30 %	21 %	21 %	12 %	6 %	0 %	0 %	3 %	0 %

When the number of neurons is 10, the error rate decreases for each set sample. However, this error rate may not decrease regularly for each training set. For example, ANN with Set 4 neuron number 15 can be classified with 3 % error, while ANN with 20 neuron number can be classified

with 6 % error rate. At the same time, if the performances of each training set are compared in general, ANN with 20 neurons performs better than ANN with 15 neurons. In Fig. 5, the total error rate is shown by finding the ratio of the number of samples that cannot be classified in all training sets to the total number of samples (all samples number = 782). As shown in Fig. 5, ANN with the number of 5 neurons has the highest error rate. This ANN was not able to classify 234 handwritten images from 782 training data with a 30 % error rate. ANN with the lowest error rate is ANN with 37 neurons. This artificial neural network could not classify 3 handwritten images from 782 training data with an error rate of 0.38 %.

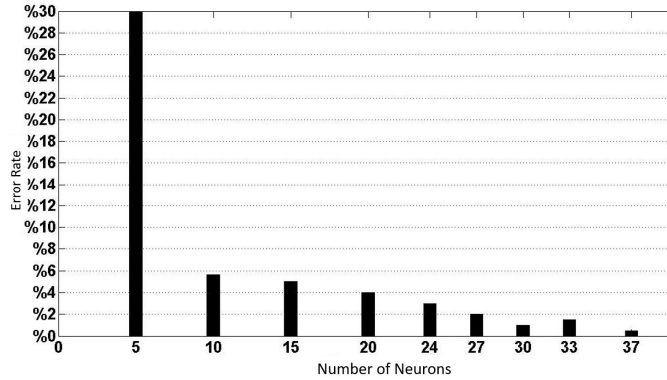


Fig. 5. Total error rates due to the number of neurons in the hidden layer

Therefore, it can be seen as ANN with 37 neuron number in the network hidden layer, which gives the best results in Figure 5. However, if a very low error tolerance is chosen, the ANN gains the ability to memorize more than the ability to learn. A network that has the ability to memorize cannot classify different input data when there is a slight change in training data. For this reason, in order to prevent ANN from memorizing, by choosing an ANN with 24 neurons in the hidden layer, it gains learning ability rather than the memorization ability of the network and it can classify handwriting samples similar to the training data.

3. Performance criteria

In this study, confusion matrices are constructed for determining the performance metrics of the proposed model. Metrics of accuracy, precision and f-measure were used in this study [31]. They can be expressed as follows:

Accuracy = (TP+TN)/(TP+TN+FP+FN).
Fault rate = (FP+FN)/(TP+TN+FP+FN).
Precision = TN/(TN+FP).
Recall = TP/(TP+FN).
F-Measure = (2xRecall*Precision)/(Recall+Precision).

4. Experimental Results

In this study, the test phase of the model was carried out first with the training data set. 70 % for ANN with 5 neurons in the hidden layer, 94.37 % for ANN with 10 neurons, 95 % for ANN with 15 neurons, 96 % for ANN with 20 neurons, 97 % for ANN with 24 neurons, 27 % for 27 neurons Overall accuracy rates of 98 % for ANN, 99 % for ANN with 30 neurons, 98.5 % for ANN with 33 neurons, and 99.5 % for ANN with 37 neurons were obtained. As the number of neurons in the hidden layer increases, ANN should have a certain fault tolerance. Therefore, ANN with 24 neurons hidden layer was chosen. Then, the model was tested with a different data set. The confusion matrix of the network created with test data is given in Table 3. The matrix

measures how accurately the network classifies. In the matrix, rows represent the actual labels of the samples in the test set, and the columns represent the predicted labels of the model. For example, with the test data given according to Table 3, the model seems to misclassify the letter A as the letters B, H and O.

Table 3. Confusion matrix for handwriting recognition model

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	T	U	V	Y	Z
A	20	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
B	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	21	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
D	0	1	0	21	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
E	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	22	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
H	0	2	0	0	0	0	0	19	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
I	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	1
K	0	1	0	0	0	0	1	0	0	0	19	0	1	0	0	0	1	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	1	0	0	21	0	0	0	1	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	22	0	0	0	0	0	0	0
R	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0
S	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	1
U	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	22	0	0	0
V	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	22	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23

As can be seen in Table 4, letters are classified with a high f1 score of 100 % for E, I, L, Y letters. In the remaining letters, high f1 score rates were obtained, and the artificial neural network created with 24 hidden neurons achieved a 94.90 % overall success rate. The model obtained the lowest performance rates in the letters H and K.

In addition, Set20 handwriting was evaluated as an example in the test of ANN with 24 neurons in the hidden layer of which the training phase was finished. Fig. 6 shows the handwritten image given to the ANN. In Fig. 7, C++ image of this handwritten program is given. Handwriting is correctly classified as shown in Fig. 7.

Fig. 6. Set 20 C++ screen image

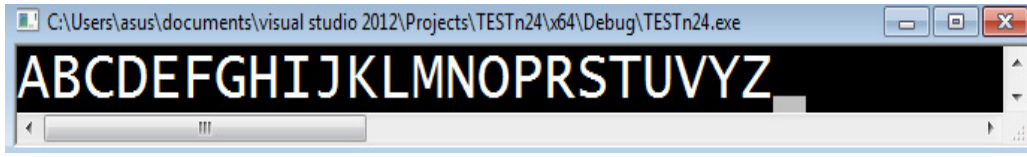


Fig. 7. Classification in Set 20 C++ environment

5. Conclusions

It has been seen in literature studies that there are both academic and many application studies on handwriting recognition. The fact that there are many patent studies on the subject of handwriting recognition and the continuation of academic studies support the development of new applications in this field. In addition, it was observed in the literature review that handwriting studies in different languages are also available. In this study, Turkish handwritten letter recognition system from A to Z was developed by using image processing, feature extraction and ANN algorithms in C++ and OpenCV environments. The handwriting images were presented to the network after the feature data was extracted and the training process of ANN was completed and the handwriting images were tried to be classified.

In this study, MLP (Multi-Layered Perceptron: MLP) type neural network and back-propagation learning algorithm were used. Different handwritten character images, image processing algorithms, feature extraction are examined. The accuracy of the system is increased by examining more than one sample for each letter. As a result of this examination, it was determined that the same data of each letter does not change, each letter does not have the same number of variable data numbers and that each letter does not contain the same variable data. Therefore, the letters are divided into character groups and the algorithm is created using the variable data of each letter group. The Turkish handwriting characters were successfully classified by developed model. For the handwriting characters of the ANN, it has been trained with 782 samples and ANNs with different numbers of neurons have been created and tested in the hidden layer, and the ANN to be used in classification has been determined. As a result, 24 neuron numbered ANN was selected in the hidden layer and the handwriting was classified with an accuracy rate of 94.90 %.

In order to increase the model performance rate, it is targeted for future studies to create a more comprehensive network by creating training datasets with more examples. In addition, it is planned to train the model for more difficult tasks by working on handwriting. In order to increase model success, it is planned to use models such as Support Vector Machines (SVMs), Random Forest, which have been proven its effectiveness on classification in the literature.

References

[1] **Elmas Ç.** Yapay Zekâ Uygulamaları. Seçkin Yayıncılık, İstanbul, 2016.

[2] **Kamble B. C.** Speech recognition using artificial neural network–a review. International Journal of Computing, Communications and Instrumentation Engineering, Vol. 3, Issue 1, 2016, p. 61-64.

[3] **Çubukçu A., Kuncan M., Kaplan K., Ertunc H. M.** Development of a voice-controlled home automation using Zigbee module. 23nd Signal Processing and Communications Applications Conference (SIU), 2015, p. 1801-1804.

[4] **Bhushan B., Singh S., Singla R.** License plate recognition system using neural networks and multithresholding technique. International Journal of Computer Applications, Vol. 84, Issue 5, 2013.

[5] **Abd Allah M. M.** Artificial neural networks based fingerprint authentication with clusters algorithm. Informatica, Vol. 29, Issue 3, 2005, p. 303-307.

[6] **Kaplan K., Bayram S., Kuncan M., Ertunç H. M.** Feature extraction of ball bearings in time-space and estimation of fault size with method of ANN. Proceedings of the 16th Mechatronika, 2014.

[7] **Bayram S., Kaplan, Kuncan M., Ertunç H. M.** Bilyeli Rulmanlarda Zaman Uzayında İstatistiksel Öznitelik Çıkarımı ve Yapay Sinir Ağları Metodu ile Hata Boyutunun Kestirimi. Otomatik Kontrol Ulusal Toplantısı, 2013.

[8] **Kuncan F., Kaya Y., Kuncan M.** A novel approach for activity recognition with down-sampling 1D local binary pattern. Advances in Electrical and Computer Engineering, Vol. 19, Issue 1, 2019, p. 35-44.

[9] **Kuncan F., Kaya Y., Kuncan M.** New approaches based on local binary patterns for gender identification from sensor signals. Gazi University Journal of Engineering and Architecture, Vol. 34, Issue 4, 2019, p. 2173-2185.

[10] **Taşdemir E. F. B., Yeşilyurt A. B. Y.** Large vocabulary recognition for online Turkish handwriting with sublexical units. Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 26, Issue 5, 2018, p. 2218-2233.

[11] **Vural E., Erdogan H., Oflazer K., Yanikoglu B.** An online handwriting recognition system for Turkish. Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference, 2004, p. 607-610.

[12] **Alex M., Das S.** An approach towards Malayalam handwriting recognition using dissimilar classifiers. Procedia Technology, Vol. 25, 2016, p. 224-231.

[13] **Asthana S., Haneef F., Bhujade R.** Handwritten multiscript numeral recognition using artificial. Neural Networks (IJSCE), Vol. 1, Issue 1, 2011.

[14] **Saeed Al Mansoori** Intelligent handwritten digit recognition using artificial neural network int. Journal of Engineering Research and Applications, Vol. 5, Issue 5, 2015, p. 46-51.

[15] **Yao C., Cheng G.** Approximative Bayes optimality linear discriminant analysis for Chinese handwriting character recognition. Elsevier Neurocomputing, Vol. 207, 2016, p. 346-353.

[16] **Jayech K.** Synchronous Multi Stream Hidden Markov Model for offline Arabic handwriting recognition without explicit segmentation. Neurocomputing, Vol. 214, 2016, p. 958-971.

[17] **Marti U. V., Bunke H.** The IAM-database: An English sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition, Vol. 5, Issue 1, 2002, p. 39-46.

[18] **Yang Y.** Application of artificial immune system in handwritten Russian uppercase character recognition. International Conference on Computer Science and Service System (CSSS), 2011, p. 238-241.

[19] **Espana S., Castro M. J., Hidalgo J. L.** The SPARTACUS-Database: a Spanish Sentence Database for Offline Handwriting Recognition. 4th International Conference on Language Resources and Evaluation, Lisbon, Portugal, 2004, p. 227-230.

[20] **Erdem A., Uzun E.** Turkish Times New Roman, Arial, and Handwriting Characters Recognition by Neural Network. Gazi University Journal of Engineering, Vol. 20, Issue 1, 2005, p. 13-19.

[21] **Carbune V., Gonnet P., Deselaers T., Rowley H. A., Daryin A., Calvo M., Gervais P.** Fast multi-language LSTM-based online handwriting recognition. International Journal on Document Analysis and Recognition (IJDAR), Vol. 23, 2020, p. 89-102.

[22] **Şengül Ö., Öztürk S., Kuncan M.** Color based object separation in conveyor belt using PLC. Avrupa Bilim ve Teknoloji Dergisi, Vol. 18, 2020, p. 401-412.

[23] **Kuyumcu B.** OpenCV Görüntü İşleme ve Yapay Öğrenme. LEVEL, İstanbul, 2015.

[24] **Yao C., Cheng G.** Approximative Bayes optimality linear discriminant analysis for Chinese handwriting character recognition. Neurocomputing, Vol. 207, 2016, p. 346-353.

[25] **Jayech K., Mahjoub M. A., Amara N. E. B.** Synchronous multi-stream hidden Markov model for offline Arabic handwriting recognition without explicit segmentation. Neurocomputing, Vol. 214, 2016, p. 958-971.

[26] **Öztemel E.** Yapay Sinir Ağları. Papatya Yayıncılık Eğitim, 3. Basım, İstanbul, 2012.

[27] **Singh D., Khehra B. S.** Digit recognition system using back propagation neural network. International Journal of Computer Science and Communication, Vol. 2, Issue 1, 2011, p. 197-205.

[28] **Debes K., Koenig A., Gross H.** Transfer Functions in Artificial Neural Networks A Simulation Based Tutorial Department of Neuroinformatics and Cognitive Robotics, Technical University Ilmenau, Germany.

[29] **Lecun Y., Boser B., Henderson D., Hubbard W., Howed R., Jackel D.** Backpropagation Applied to Handwritten Zip Code Recognition. AT& T Bell Laboratories Holmdel, NJ 07733 USA.

[30] **Cilimkovic M.** Neural Networks and Back Propagation Algorithm Institute of Technology Blanchardstown Road North Dublin 15, Ireland.

[31] **Kaya Y., Kuncan M., Kaplan K., Minaz M. R., Ertunç H. M.** A new feature extraction approach based on one dimensional gray level co-occurrence matrices for bearing fault classification. Journal of Experimental and Theoretical Artificial Intelligence, 2020, https://doi.org/10.1080/0952813X.2020.1735530.

Melih Kuncan received the B.S., M.S., and Ph.D. degrees in mechatronics engineering from Kocaeli University, Kocaeli, Turkey, in 2010, 2013, and 2017. He is currently an Assistant Professor with the Department of Electrical and Electronics Engineering, Siirt University. His areas of research include control systems, instrumentation, and measurement, signal processing, photovoltaics.

Enes Vardar received the B.S. degrees in mechatronics engineering from Kocaeli University, Kocaeli, Turkey, in 2017. His research areas are control systems, signal processing, and artificial intelligence.

Kaplan Kaplan received the B.S. and M.S. degrees in mechatronics engineering from Kocaeli University, Kocaeli, Turkey, in 2012 and 2015, respectively, where he is currently pursuing the Ph.D. degree. His research areas are control systems, signal processing, and artificial intelligence.

H. Metin Ertunc received the B.S. degree from Electrical and Electronics Engineering Department at Hacettepe University in 1991 in Ankara, Turkey. He obtained his M.S. degree from Systems and Control Engineering Department and Ph.D. degree from Electrical Engineering and Computer Science Department at Case Western Reserve University, Cleveland, OH, USA, in 1995 and 1999, respectively. Currently, Dr. Ertunc is a Professor at Mechatronics Engineering Department at Kocaeli University. His research interests include automatic control and monitoring systems, fault detection based on signal processing techniques, artificial neural networks.