December 28, 2021

Handwritten Digit Recognition using Deep Learning Algorithms and MNIST Dataset

Yousef Elbayoumi

Department of Big Data Analytics and Management, Bahçeşehir University, Turkey

Corresponding author: Yousef Elbayoumi (e-mail: yousefxelbayomi@gmail.com).

This paper represents my individual work in the topic of handwritten digit recognition using deep learning algorithms and MNIST dataset.

ABSTRACT Humans' reliance on technology has never been higher, to the point that deep learning and machine learning algorithms can conduct everything from object detection in images to adding sound to silent movies. Similarly, handwritten text recognition is a major area of research and development with a plethora of possibilities. Handwriting recognition (HWR), also known as Handwriting Text Recognition (HTR), is a computer's capacity to recognize and understand comprehensible handwritten input from sources such as paper documents, pictures, touch displays, and other devices [1]. Apparently, I will use Support Vector Machines (SVM), Multi-Layer Perceptron (MLP), and Convolution Neural Network (CNN) models to accomplish handwritten digit recognition using MNIST datasets in this paper. For training and testing the proposed model's technique, the MNIST dataset of handwritten digits is employed. The MNIST dataset is made up of a wide range of deformed handwritten digit pictures. My major goal is to compare the accuracy of the above models with their execution times in order to find the optimum digit recognition model.

INDEX TERMS Deep Learning, Machine Learning, MNIST Dataset, Handwritten Digit Recognition, Support Vector Machine (SVM), Multi-Layered Perceptron (MLP), Convolution Neural Network (CNN).

I. INTRODUCTION

Handwritten digit recognition refers to a computer's capacity to recognize human handwritten digits from a variety of sources, such as photographs, papers, and touch displays, and classify them into ten specified categories (0-9). In the realm of deep learning, this has been a topic of inexhaustible investigation. There are numerous applications for digit recognition, including number plate identification, postal mail sorting, and bank check processing [2]. Because handwritten digit recognition is not an optical character recognition, I encounter numerous obstacles due to the various styles of writing used by different people. This study compares and contrasts various machine learning and deep learning methods for handwritten digit recognition. I utilized a Support Vector Machine, a Multilayer Perceptron, and a Convolutional Neural Network to accomplish this. The algorithms are compared based on their accuracy, mistakes, and testing-training time, which are supported by plots and charts created with matplotlib for visualization. Any model's correctness is critical because more accurate models produce better decisions. Low-accuracy models are unsuitable for real-world applications. For example, great accuracy is important in an automated bank cheque processing system that recognizes the amount and date on the check. If the system reads a digit wrongly, it can cause significant damage, which is not desirable. As a result, in these real-world applications, a high-accuracy algorithm is necessary. As a result, I'm offering a comparison of several algorithms based on their accuracy, so that the most accurate method with the fewest chances of errors can be used in a variety of handwritten digit recognition applications. For handwritten digit recognition, this paper provides a reasonable grasp of machine learning and deep learning techniques such as SVM, CNN, and MLP. It also tells you which algorithm is the most effective at doing digit recognition. In the following sections of this paper, I will describe similar work in this topic, as well as the approach and implementation of all three algorithms, in order to have a better understanding of them. The conclusion and outcome are then presented, which are supported by the work I've done in this paper. Furthermore, it will show you some possible future improvements that can be made in this subject. The citations and references utilized are listed in the latter portion of this paper.

II. BACKGROUND

With the humanization of machines, a significant amount of research and development activity has sparked interest in deep learning, machine learning, and artificial intelligence. Machines are becoming increasingly smart throughout time, and they have made our lives more secure and manageable, from fundamental math to retina recognition. Similarly, handwritten text recognition is an important application of deep learning and machine learning that aids in the detection of forgeries, and a large amount of research has already been done that includes a comprehensive study and implementation of various popular algorithms to compare the different models of CNN with the fundamental machine learning algorithms on various criteria such as performance rate, execution time, complexity, and so on to evaluate each algorithm. [3] concluded that the Multilayer Perceptron classifier produced the most accurate results with the lowest error rate, followed by Support Vector Machine, Random Forest Algorithm, Bayes Net, and Random Tree, respectively, while [4] compared SVM, CNN, KNN, and RFC and found that CNN (which took the most time to execute) had the highest accuracy of 98.72 percent and RFC had the lowest accuracy. [5] conducted a detailed study-comparison of SVM, KNN, and MLP models to classify handwritten text, concluding that KNN and SVM correctly predict all classes of dataset with 99.26 percent accuracy, but the process becomes a little more complicated with MLP when it has trouble classifying number 9, and it is suggested that CNN with Keras be used to improve the classification.

All we need is a lot of data and knowledge to train a massive neural network to accomplish what we want, so convolution may be regarded as "looking at functions around it to produce an accurate prediction of its outcome." Using MNIST datasets, [6], [7] used a convolution neural network for handwritten digit recognition. [6] employed a 7-layered CNN model with 5 hidden layers, as well as gradient descent and back prorogation models, to detect and compare accuracy across epochs, resulting in a maximum accuracy of 99.2%. In [7], they examine the many components of CNN, their progression from LeNet-5 to SENet, and comparisons between alternative models such as AlexNet, DenseNet, and ResNet.

III. METHODOLOGY

The comparison of the algorithms (Support vector machines, Multi-layered perceptron, and Convolutional neural network) is based on a characteristic chart of each algorithm on common grounds such as dataset, number of epochs, algorithm complexity, accuracy of each algorithm used to execute the program, and algorithm runtime under ideal conditions.

A. DATASET

The MNIST dataset, a well-known handwritten digit benchmark dataset, was utilized to train and test the suggested classification system in this study. Handwritten images of the ten digits make up the dataset (0 to 9). The photos have been normalized and are available in unsigned byte format. The MNIST dataset is a subset of a bigger dataset accessible from the National Institute of Standards and Technology (NIST), and it contains a total of 70000 data elements, with 60000 examples for training and 10000 examples for testing. A 20x20 pixel matrix was used to represent the photos in the NSIT database. However, there were some gray levels in the photographs. In the MNIST dataset, this problem was handled by normalizing all of the 2020 handwritten images into 2828 images. The previous image's center of mass is determined, and the normalized image is then positioned inside the 2828 image in such a way that the center of mass is at the center. In the MNIST collection, several sample handwritten digits are shown in Fig. 1. It's also worth noting that the MNIST dataset is completely noise-free.

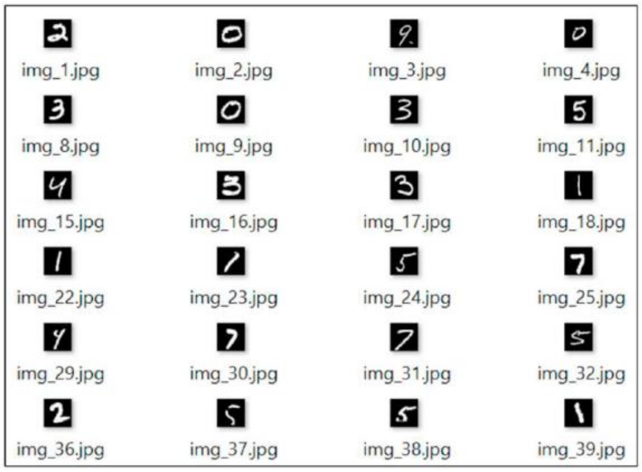

Figure 1. The MNIST Dataset.

Pre-processing is a preliminary phase that involves all of the actions required to convert the input data into a format suitable for feature extraction. It entails obtaining character input, noise reduction, image centering, and scaling. Because feature extraction can be misled if the input photos are not correctly pre-processed, this is a critical step. The MNIST handwritten digits images are already size normalized and centered in this study.

B. SUPPORT VECTOR MACHINE (SVM)

The Support Vector Machine (SVM) is a machine learning algorithm that is supervised. In this case, I plot data items in n-dimensional space, where n is the number of features, and a specific position indicates the value of a feature. I then classify the data by locating the hyperplane that separates the two classes. It will select the hyperplane that best separates the classes. SVM selects the extreme vectors that aid in the construction of the hyperplane. Support vectors represent the extreme examples, which is why the technique is called Support Vector Machine. Linear and non-linear SVMs are the two primary types of SVMs. I employed Linear SVM for handwritten digit recognition in this paper [8].

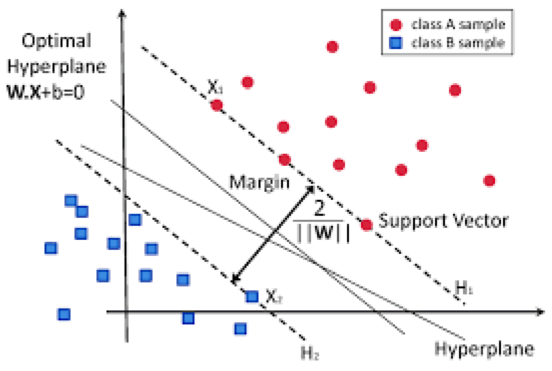

Figure 2. This picture depicts the operating mechanism of SVM Classification with supporting vectors and hyperplanes.

C. MULTILAYERED PERCEPTRON (MLP)

A feedforward artificial neural network called a multilayer perceptron (MLP) is a type of feedforward artificial neural network (ANN). It has three layers: an input layer, a concealed layer, and a final output layer. Each layer is made up of numerous nodes, which are also known as neurons, and each node is connected to the nodes of the following layer. There are three layers in a basic MLP, however the number of hidden layers can be increased to any number depending on the problem, with no limit on the number of nodes. The input and output layers' nodes are determined by the number of attributes and apparent classes in the dataset, respectively. Due to the unpredictable nature of the model, the number of hidden layers or nodes in the hidden layer is impossible to determine and must be chosen empirically. For processing, each hidden layer of the model might have a different activation function. It employs backpropagation, a supervised learning approach, for learning objectives. The node connections in the MLP are made up of weights that are modified to synchronize with each other during the model's training process [9].

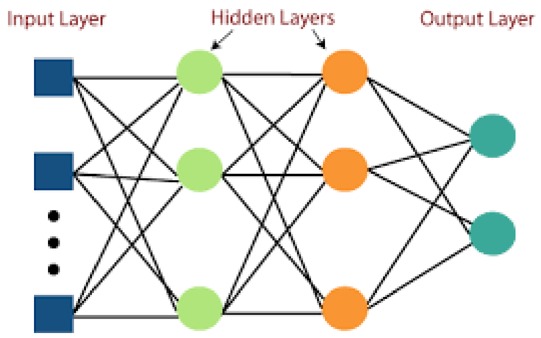

Figure 3. This diagram depicts the Multilayer Perceptron's basic design, as well as the network's variable specifications.

D. CONVOLUTIONAL NEURAL NETWORK (CNN)

CNN is a deep learning technique for image recognition and classification that is frequently used. It's a type of deep neural network that requires the least amount of pre-processing. It inputs the image in discrete chunks rather than a single pixel at a time, allowing the network to recognize ambiguous patterns (edges) more effectively in the image. Convolutional layers, Pooling layers (Max and Average pooling), Fully connected layers (FC), and normalizing layers are among the hidden layers found in CNN [10]. To extract features from the input image, CNN employs a filter (kernel), which is an array of weights. To introduce some non-linearity, CNN uses distinct activation functions at each layer [11]. As we walk deeper into the CNN, we see that the height and width of the screen are shrinking while the number of channels grows. Finally, the result is predicted using the created column matrix [12].

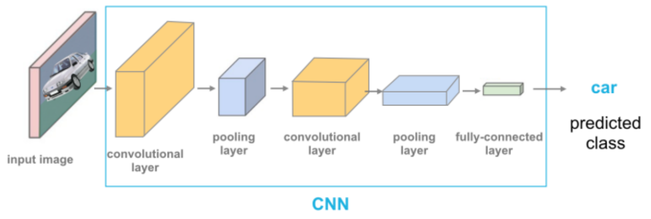

Figure 4. In the style of a flow chart, this diagram depicts the architectural architecture of CNN layers.

E. VISUALIZATIONS

I compared several level deep and machine learning algorithms (i.e., SVM, ANN-MLP, CNN) on the basis of execution time, complexity, accuracy rate, number of epochs, and number of hidden layers using the MNIST dataset (i.e., handwritten digit dataset) (in the case of deep learning algorithms). I used bar graphs and tabular format charts using the matplotlib module to show the information gained from the deep analysis of algorithms, which offers us the most precise visuals of the algorithms' step-by-step advancements in detecting the digit. The graphs are placed at key points in the programs to provide visual representations of each component and to support the outcome.

IV. IMPLEMENTATION

I utilized three separate classifiers to compare the algorithms based on working accuracy, execution time, complexity, and the number of epochs (in deep learning methods):

- Machine Classifier using Support Vectors
- Multilayer Perceptron Classifier (ANN)
- Classifier using a Convolutional Neural Network

I've gone over the implementation of each algorithm in great depth below in order to establish a flow for this analysis that is both fluent and accurate.

I. PRE-PROCESSING

Pre-processing is a stage in machine and deep learning that focuses on removing undesired contaminants and redundancy from the input data. I reshaped all of the photos in the dataset into 2-dimensional images to simplify and break down the input data (28,28,1). The photos' pixel values range from 0 to 255, therefore I normalized them by converting the dataset to 'float32' and then dividing by 255.0, resulting in input features that range from 0.0 to 1.0. Then, using one-hot encoding, I turned the y values into zeros and ones, making each number categorical. For example, an output value of 4 will be changed into an array of zero and one, i.e. [0,0,0,0,1,0,0,0,0,0]

II. SVM - IMPLEMENTATION

Both dense (numpy.ndarray and numpy.asarray) and sparse (any scipy.sparse) sample vectors are supported as input by the SVM in scikit-learn [15]. SVC, NuSVC, and LinearSVC are scikit-learn classes that can conduct multi-class classification on a dataset. LinearSVC was utilized in this paper to classify MNIST datasets that used a Linear kernel has built [16].

NumPy, matplotlib, pandas, Sklearn, and seaborn were among the scikit-learn packages utilized in the implementation. I'll start by downloading the MNIST datasets, then load and read those CSV files with pandas.

Following that, selected data were plotted as well as converted into a matrix, which was followed by normalization and scaling of features. Finally, I developed a linear SVM model and a confusion matrix for assessing the model's accuracy [13].

III. MLP - IMPLEMENTATION

Handwritten digits recognition is implemented using the Keras module to generate an MLP model of Sequential class and add respective hidden layers with distinct activation functions to take an image of 28x28 pixel size as input. We added a Dense layer with varying specifications and drop out layers after developing a sequential model, as seen in the image below. The block diagram is provided for your convenience. You can train a neural network in Keras by following these steps once you have the training and test data.

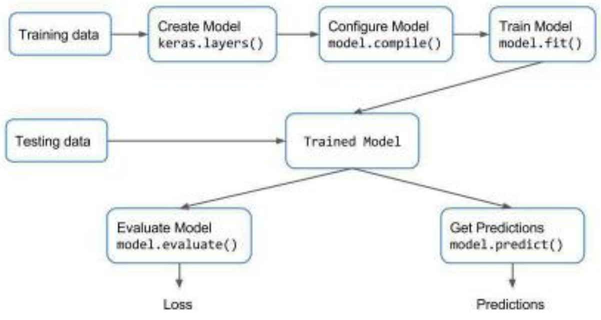

Figure 5. Keras module was used to create a sequential block diagram of a multi-layers perceptron model.

We employed a neural network with four hidden layers and a ten-unit output layer (i.e., total number of labels). In the concealed layers, the number of units is preserved at 512. The 784-dimensional array translated from the 2828 image serves as the network's input. The network was built using the Sequential model. We may simply stack layers in the Sequential model by adding the desired layer one by one. We picked the Dense layer, also known as a completely linked layer, because we're creating a feedforward network with all neurons from one layer coupled to neurons from the preceding one. We also added the Relu activation function, which is essential to introduce non-linearity to the model, in addition to the Dense layer. The network will be able to learn non-linear decision limits as a result of this. Because this is a multiclass classification problem, the final layer is a SoftMax layer [17].

IV. CNN - IMPLEMENTATION

Keras is used to implement handwritten digit recognition using a Convolutional Neural Network [14]. It is a free and open-source neural network library for creating and implementing deep learning models. We used a Sequential class from Keras, which allowed us to build the model layer by layer. The input image's dimensions are 28 (height), 28 (width), and 1 (height) (Number of channels). Next, we developed a model with a Conv layer as the initial layer [18]. This layer employs a matrix to extract features from the input data by convolving it over its height and width. A Filter or Kernel is the name given to this matrix. Weights are the values in the filter matrix. With a stride of 1, we utilized 32 filters in each of the dimensions (3,3). The number of pixels that shift is determined by the stride. The dimension of activation maps is determined by the formula: ((N + 2P - F)/S) + 1, where N is the dimension of the input image, P is padding, F is filter dimension, and S is stride. The output image's Depth (number of channels) is equal to the number of filters employed in this layer. We employed Relu [19] as an activation function to increase the non-linearity. After that, we utilize another convolutional layer with 64 filters of the same dimensions (3,3), a stride of 1, and the Relu function.

Following these layers, the pooling layer [20] is utilized, which minimizes the image's dimensionality and network computation. We've used MAX-pooling, which preserves just the pool's maximum value. In this layer, the network's depth remains unchanged. With a pool-size of (2,2) and a stride of 2, every four pixels will become a single pixel. Dropout layer [21] is used to avoid overfitting in the model by dropping some neurons that are picked at random, allowing the model to be simplified. The probability of a node being dropped out has been set to 0.25 percent. Following that, Flatten Layer [12] is employed, which includes flattening the 2-dimensional matrix, i.e. obtaining a column matrix (vector) from it. The completely linked layer [12] will receive this column vector. There are 128 neurons in this layer, with a dropout frequency of 0.5 or 50%. The output of the Relu activation function is fed into the model's output layer, which is the last layer. The SoftMax function [22] is used to conduct the classification in this layer, which comprises ten neurons that represent classes (numbers 0 to 9). The probability distribution for all ten classes is returned by this function. The output is the class with the highest probability.

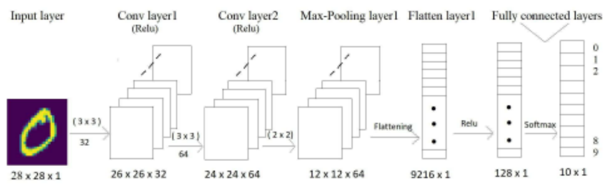

Figure 6. Convolutional Neural Network Architecture in Detail with Appropriate Layer Specifications

V. RESULT

After building all three algorithms, SVM, MLP, and CNN, I compared their accuracies and execution times using experimental graphs to gain a better understanding. All of the models listed above have had their Training and Testing Accuracy taken into account. After testing all of the models, I discovered that SVM has the most accuracy on training data, whereas CNN has the highest accuracy on testing data. I also compared the execution times to acquire a better understanding of how the algorithms work. In general, an algorithm's execution time is proportional to the number of operations it has completed. So, to acquire the best result, I trained our deep learning model for 30 epochs and SVM models according to norms. The SVM took the shortest amount of time to execute, whereas CNN took the longest.

Table 1. Analysis of different models in comparison

MODEL	TRAINING RATE	TESTING RATE	EXECUTION TIM
SVM	99.98%	94.005%	1:35 min
MLP	99.92%	98.85%	2:32 min
CNN	99.53%	99.31%	44:02 min

Here is some of the visualization results

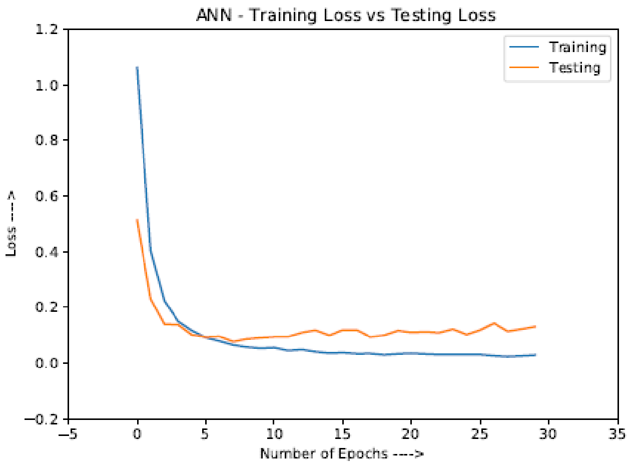

Figure 7. Loss rate v/s Number of epochs graph depicting the transition of training loss as the number of epochs increases in a Multilayer Perceptron.

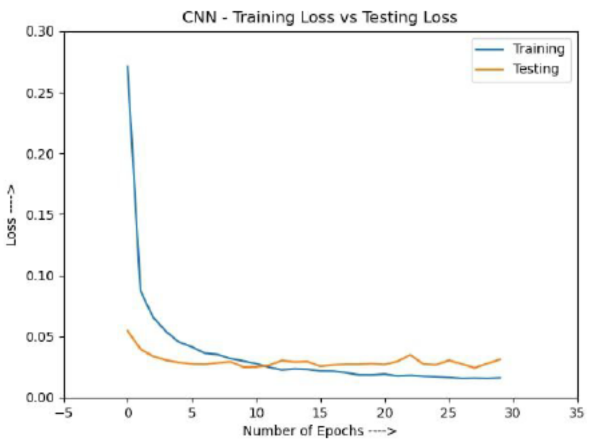

Figure 8. Graph depicting the transition of CNN training loss as the number of epochs increases.

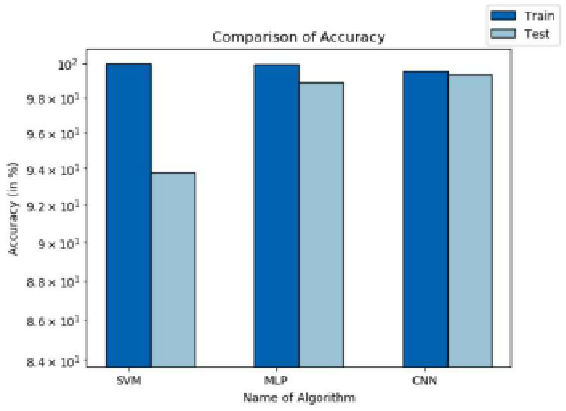

Figure 9. SVM (Train: 99.98 percent, Test: 93.77 percent), MLP (Train: 99.92 percent, Test: 98.85 percent), and CNN (Train: 99.53 percent, Test: 99.31 percent) are shown in a bar graph.

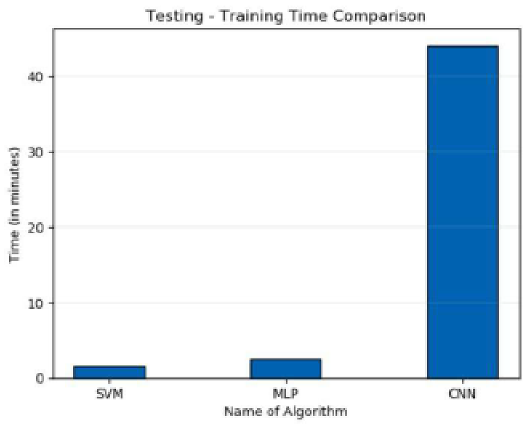

Figure 10. SVM, MLP, and CNN execution times are compared in a bar graph (SVM: 1.58 mins, MLP: 2.53 mins, CNN: 44.05 mins).

VI. CONCLUSION

Using MNIST datasets, we constructed three models for handwritten digit recognition based on deep and machine learning algorithms in this study. We compared them based on their attributes to determine which one was the most accurate. Support vector machines are one of the most basic classifiers, which is why they're faster than other algorithms and, in this case, deliver the highest training accuracy rate. However, due to their simplicity, they can't categorize complicated and ambiguous images as accurately as MLP and CNN algorithms can. For handwritten digit recognition, we discovered that CNN provided the best accurate results. As a result, we may infer that CNN is the best choice for any form of prediction task involving picture data. Following that, we concluded that increasing the number of epochs without changing the configuration of the algorithm is pointless due to the limitation of a particular model, and we observed that after a certain number of epochs, the model begins overfitting the dataset and gives us biased predictions.

VII. FUTURE DIRECTIONS

After The future development of applications based on deep and machine learning algorithms is virtually limitless. In the future, we can work on a denser or hybrid algorithm that uses more manifold data than the existing collection of algorithms to solve a variety of problems.

In the future, the application of these algorithms will range from the general public to high-level authorities; as a result of the above differentiation and future development, we will be able to achieve high-level functioning applications that can be used in classified or government agencies as well as for the general public; and we will be able to use these algorithms in hospitals for detailed medical diagnosis, treatment, and monitoring of patients.

By incorporating these algorithms into the day-to-day and high-level applications, we may contribute to create an environment of safety, awareness, and comfort (i.e. Corporate level or Government level).

ACKNOWLEDGEMENT

I would like to thank my university, Bahçeşehir University for their great instructors who helped me to reach the point where I can make research like this one.
Also, I would like to thank my teacher Dr. Öğr. Üyesi Atınç YILMAZ for his continuous support and guidance.

DECLARATION OF COMPETING INTEREST

I declare that I have no known competing financial interests or personal ties that could have influenced the research reported in this paper.

REFERENCES

[1] “Handwriting recognition – From Wikipedia”: https://en.wikipedia.org/wiki/Handwriting_recognition

[2] “What can a digit recognizer be used for? – From Quora”: https://www.quora.com/What-can-a-digit-recognizer-be-used-for

[3] ”Handwritten Digit Recognition using Machine Learning Algorithms”,
S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair.

[4] ”Handwritten Digit Recognition Using Deep Learning”, Anuj Dutt and Aashi Dutt.

[5] ”Handwritten recognition using SVM, KNN, and Neural networks”,
Norhidayu binti Abdul Hamid, Nilam Nur Binti Amir Sharif.

[6] ”Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers”, Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique.

[7] ”Advancements in Image Classification using Convolutional Neural Network” by Farhana Sultana, Abu Sufian & Paramartha Dutta.

[8] “Support Vector Machine Algorithm – From JavaTpoint”: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[9] “Crash Course on Multi-Layer Perceptron Neural Networks – From Machine Learning Mastery”: https://machinelearningmastery.com/neural-networks-crash-course/

[10] “An Introduction to Convolutional Neural Networks – From ResearchGate”: https://www.researchgate.net/publication/285164623_An_Introduction_to_Convolutional_Neural_Networks

[11] “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning”,
Chigozie Enyinna Nwankpa, Winifred Ijomah,
Anthony Gachagan, and Stephen Marshall:
https://arxiv.org/pdf/1811.03378.pdf

[12] “Basic Overview of Convolutional Neural Network (CNN) by Udeme Udofia – From Medium”: https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17

[13] “SVM digit recognition using MNIST dataset by rishikakushwah16 – From GitHub”: https://github.com/rishikakushwah16/SVM_digit_recognition_using_MNIST_dataset

[14] “Handwritten Digit Recognition by ritikdixit1 – From GitHub”: https://github.com/ritikdixit1/Handwritten-Digit-Recognition

[15] “Support Vector Machines – From scikit-learn.org”: https://scikit-learn.org/stable/modules/svm.html

[16] “LIBSVM – From Wikipedia”: https://en.wikipedia.org/wiki/LIBSVM

[17] “Image Classification using Feedforward Neural Network in Keras – From LearnOpenCV”: https://learnopencv.com/image-classification-using-feedforward-neural-network-in-keras/

[18] “How Do Convolutional Layers Work in Deep Learning Neural Networks? – From Machine Learning Mastery”: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[19] “A Gentle Introduction to the Rectified Linear Unit (ReLU) – From Machine Learning Mastery”: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[20] “A Gentle Introduction to Pooling Layers for Convolutional Neural Networks – From Machine Learning Mastery”: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

[21] “Dropout in (Deep) Machine learning by Amar Budhiraja – From Medium”: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[22] “Understand the Softmax Function in Minutes by Uniqtech – From Medium”: https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d