2023 4th International Conference on Intelligent Technologies (CONIT)
Karnataka, India. Jun 21-23, 2024

CNN Model for Handwritten Digit Recognition with Improved Accuracy and Performance Using MNIST Dataset

Prince Raj[1,2], Ayita Dutta[1,2], Premanshu Sharma[1,2], Md. Hammad Zaya[1,2], Mouli Das[1,3], Tanmoy Ghosh[1,3], Pushpita Roy[1,3]
[1]Narula Institute of Technology, Agarpara, India
[2]{princerajlegal, ayitadutta2002, 09anshu850, hammadzaya}@gmail.com,
[3]{mouli.das, tanmoy.ghosh, pushpita.roy}@nit.ac.in

Abstract— **Handwritten digit identification is an area that requires substantial attention. It is necessary to take into account the differences in handwritten digits' size, thickness, position, and orientation when identifying them. With the advent of machine learning techniques, handwritten digit recognition is no longer as challenging as it once was. Handwritten numbers of various kinds that may be found in postal codes, addresses, historical archives, and medical records are all handled by the system. General performance across different handwriting styles is ensured by using machine learning algorithms, especially neural networks trained on big datasets. For improved performance and accuracy in the context of handwritten digit recognition, a Convolutional Neural Network (CNN) model is put forth and assessed in this study. With its sophisticated convolutional layers and optimal settings, our CNN architecture accurately captures some of the complex properties found in handwritten digits using the widely used MNIST dataset. The work not only advances computer vision research but also sheds light on how CNN architectures could be optimized for practical uses that call for accurate digit recognition. It controls the data and intricacies of diverse writing styles. Its versatility accounts for its exceptional digit recognition abilities in any given circumstance.**

Keywords— CNN, Handwritten Digit Recognition, CNN Architecture

I. Introduction

The process of recognizing something or someone involves classifying or identifying them according to prior information or experiences. Likewise, one of the main uses of computers is handwritten digit recognition, which is a particularly challenging task that is necessary for application development. That is precisely how the machine prepares and interprets numbers using a digit recognition framework. Computers that are capable of reading handwritten digits from a range of sources—such as messages, bank checks, papers, and photos—can recognize license plates on cars, process bank checks, enter numbers in any format, and more. This is known as handwritten digit recognition. High recognition accuracy, low computing complexity, and stable recognition system performance are required in all these domains, which are associated with big datasets. Handwritten character recognition has been the subject of numerous investigations employing various deep learning models. Through the use of examples, Deep Learning is a machine learning technique that teaches computers how to execute tasks that are simple for humans. In a similar vein, human effort can be decreased in several domains, including perception, learning, and identification, by utilizing the handwritten digit recognition system [22]. Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) are the two main Deep Learning techniques.

Since the 1980s, character recognition technology has existed. Handwritten digit recognition is a crucial challenge with many applications, including online digit recognition on PC tablets, reading zip codes on envelopes, processing bank checks, manually filled-out forms (like tax forms), and more. We encounter numerous obstacles and intricacies in our endeavor to tackle this issue. The dimensions, thickness, orientation, and placement of handwritten numbers in relation to the margins are not always consistent. Updating the pattern characterization method to identify handwritten digits from the MINIST dataset of handwritten digit images [11], was the primary objective (0-9).

In this section, we will concentrate on Convolutional Neural Networks, which are a particular type of deep neural network that find extensive use in signal processing, computer vision, natural language processing, image classification, face recognition, and object recognition. Without human oversight or interaction, it can automatically identify the salient characteristics of an item (which might be a handwritten character, a face, a picture, etc. , greatly increasing the efficiency of CNNs. In the fields of pattern recognition and computer vision, handwritten digitizer recognition with convolutional neural networks is expanding quickly. Even in this day and age, where digital data is employed mostly, handwritten numbers are still the primary means of communication for everything from private messages to postal codes on envelopes. They perform better in image-based recognition tasks because CNN designs let them to recognize patterns and minute details present in characters.

Conventional or outdated techniques for reading handwritten numbers frequently have issues with handling different writing styles, sizes, and orientations. CNNs were created as a result of these deep learning methods. When used for image-based tasks or activities, these have shown to be highly effective. With convolutional layers, CNNs are highly adept at comprehending hierarchical information. This aids in their ability to identify intricate patterns and spatial relationships in pictures. CNNs have been a huge success when it comes to handwritten digit recognition, surpassing traditional algorithms in this regard. They are renowned for offering exceptional efficiency and precision as well. Our goal is to contribute to the development of dependable and efficient systems for handwritten digit recognition, both in academic discourse and in real-world applications. By experimentation, analysis, and innovation, we want to address the ever-changing demands of our society.

II. Background & Related Work

In the field of handwritten digit recognition research, the MNIST dataset [23], a widely used reference, has been crucial. The MNIST is made up of an enormous set of grayscales 28 × 28-pixel pictures of handwritten numbers 0 through 9. A fantastic and widely used tool for CNN model evaluation and training is MNIST. Researchers have been continuously refining CNN architectures in order to expand them. The researchers have tried a variety of layers, filters, and pooling techniques in an effort to get satisfactory results. Convolutional neural networks are used on many different data sets outside of the MNIST dataset for handwritten digit recognition in the real world. CNNs are therefore not restricted to a particular collection of data. These CNNs' great adaptability makes them ideal for managing a wide range of writing styles, which advances the system's ability to correctly understand a variety of handwritten inputs. Exciting developments for the recognition of handwritten numbers are on the horizon as researchers continue to develop new concepts and enhance models built on convolutional neural networks. These enhancements can significantly affect a number of areas, including data entry procedures becoming more automated and document comprehension.

Using the MNIST dataset and a convolutional neural network (CNN), the work [2], focuses on handwritten Digit recognition. The MNIST database (Modified National Institute of Standards and Technology database) handwritten digit dataset is used in this paper's methodology. Ten thousand instances are for testing and sixty thousand for training. One of the more challenging and important machine learning tasks is handwritten digit recognition (HDR). Researchers have been using it extensively for many years as a machine learning algorithm theory experiment. In this work, the methodology comprises Artificial neural networks with convolutional layers are deep. It is useful for categorizing pictures. Multilayer perceptron networks (MLPs), which were initially employed in 1980, are essentially the basis for convolutional neural networks (CNNs) [3], and it's computing is inspired by the workings of the human brain.

In the study [18], researchers discuss how variances in people's writing styles and image artifacts—such as noise, blurry, and intensity disparities in photos—make it more difficult and complex to identify characters from images. In order to address the aforementioned characteristics, quantum convolutional neural networks (QCNN) have been used. In this manner, greater accuracy can be attained. The primary reason for using a quantum convolutional neural network (QCNN) over a classical CNN is that the former can handle more complex operations more quickly than the latter. They employed the MNIST dataset, and their suggested model's efficacy resulted in an average accuracy of 91.08%. The first step in their method is gathering data to train the dataset (the MNIST dataset is utilized), after which the data is preprocessed for the modeling phase and the model is constructed using the QCNN algorithm. At this point, predictions can be made and the outcomes can be seen and understood. They add that accurate and trustworthy results can only be obtained with high-quality data. Their model's output demonstrates the superiority of QCNN over traditional CNN. The accuracy at various points has been compared. At point 1, CNN's accuracy ranges from 0% to 25%, while QCNN's accuracy ranges from 75% to 100%. CNN's accuracy ranges from 50% to 75% at point 4, and it stays at 75% at point 10. The accuracy for QCNN stays between 75% and 100% at points 4 and 10. Additionally, the data demonstrate that QCNN experiences less loss than CNN. While QCNN scored a greater accuracy of 91.08%, CNN only managed an 84.68% accuracy. The authors ended by stating that by employing QCNN, which offers both improved accuracy and faster execution times, they have overcome the shortcomings of traditional CNN, such as over-fitting and fading gradients. Additionally, their approach has demonstrated robustness to a wide range of handwriting styles and picture abnormalities.

Since CNNs have been studied, used, and have well-established algorithms, they are a better option for many applications than QCNNs, whose algorithms are still in the early stages of development and may be difficult to apply. Furthermore, compared to traditional computing hardware, quantum hardware is more constrained and difficult. Errors are more common with quantum computers.

A. Different Character recognition using different datasets.

1) Arabic Character Recognition

In a research paper, researchers collected the Arabic Handwritten Characters Dataset [20] which has 16800 elements. The database contains 13440 elements (480 images in each cluster) for the training set and 3360 elements (up to 120 images in each cluster) for the test set. There are many challenges to the recognition of Arabic characters. In the research work [5], They may have different writing forms that complicate the recognition process. The presence of a dot in Arabic Characters is a major challenge to recognize them. For example, in the approach [4], the characters ت [ta] and ث [sa] have different sounds and pronunciations and the difference between them is one extra dot. After testing various models like LeNet, AlexNet, VGG19, ResNet, and GoogLeNet, GoogLeNet stood out with high accuracy across several classes. So, they conclude for Arabic character recognition GoogLeNet is the most efficient for achieving accurate and consistent results.

2) Bangla Character Recognition

The CMATERdb dataset [6], the BanglaLekha-Isolated datasets, and the researcher's own datasets are compared in a research report. The handwritten Bengali characters in BanglaLekha-Isolated number around 10,000. 15,000 images of Bengali characters may be found in the CMATERdb collection, and 2600 handwritten characters can be found in the datasets the researcher developed. A 93.07% recognition accuracy rate is obtained once the researcher's dataset is run through the DCNN model. The CMATERdb dataset [7], yielded the best accuracy when the deep CNN model was applied, as compared to other datasets. Their primary goal is to manipulate and contrast their own dataset with those of others. Researchers who have trained their BDnet deep CNN model with the highly accurate ISI Bengali handwritten numbers dataset use it. A dataset developed by the researcher is displayed in Figure 1(a), whereas datasets isolated from BanglaLekha are displayed in Figure 1(b), and the CMATERdb dataset is displayed in Figure 1(c).

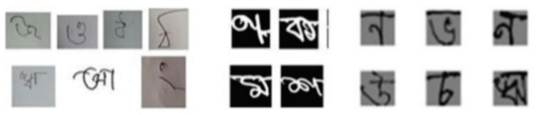


Fig. 1. (a)Researcher's created dataset, (b) BanglaLekha-Isolated datasets, (c) CMATERdb dataset. [7]

B. *Enhancing CNN for Handwritten Digit Recognition.*

The network uses biologically inspired architecture for character classification and feature extraction to improve character recognition. The numerical techniques employed are highly suited to parallel array computers. In this study [8], the three-part method for digit recognition that is independent of the writer is discussed. Initially, character pictures are used for Gabor Component optimization. Classification networks are trained second, using backpropagation learning. Finally, first-order Bayesian statistics are applied to the network's activation strength to separate substitutional errors. The images that the Gabor filter function can display are trained using the model that is thus produced. Gabor functions generate images with high quality. For applications, it leverages a limited set of fundamental functionalities. Using parallel processing during the feature extraction process, the integrated network in Fig. 2 is intended for both feature detection and classification.

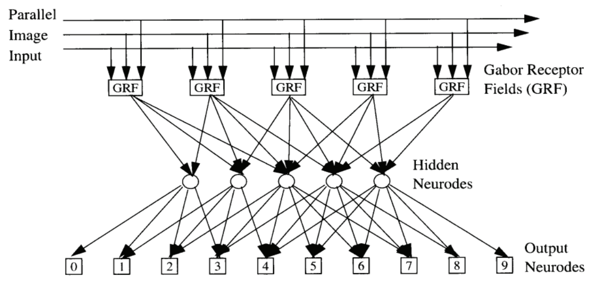


Fig. 2. Gabor filter in CNN [8]

The paper's authors [9] go over the issues that arise with feature extraction and digit recognition. An extraction of features based on LeNet5. CNN architecture is used in black box schemes to solve issues. SVM handles the classification problem in order to improve LeNet5's capacity for generalization. The MINIST dataset was employed in this study [9], and the findings indicated that the architecture created can outperform SVMs and LeNet5. While neural networks concentrate on lowering empirical risk, the Support Vector Machine operates on the notion of risk minimization, which lowers overall risk. The seven levels of the LeNet-5 CNN architecture are depicted in Fig 3. It has three convolutional layers, two subsampling layers, and two fully linked layers composition.

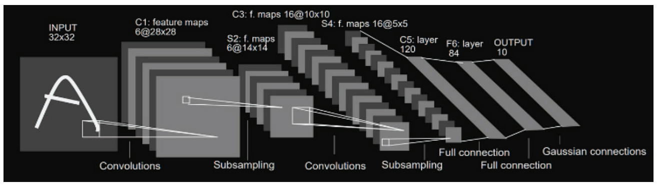


Fig. 3. LeNet 5 CNN Architecture [9]

Using a convolutional neural network based on the ShuffleNet architecture, this study [10] provides an accurate framework for handwritten digit recognition. Thus, the system is designed to categorize, validate, train, and recognize handwritten digit picture datasets into many classes for 10 digits and English characters using the potent ShuffleNet algorithms. Experiment findings indicate that the recognition model has a high 99.50% efficiency. This study provides information on the extremely effective recognition model. The transfer learning approach for learning new features and leveraging the capabilities of Nvidia GPU for parallel computation is described in this research for the pre-trained ShuffleNet convolutional neural network (CNN). For handwritten digit recognition in particular, the development of optical character recognition in deep learning is highly significant. Despite a wealth of prior research, this work [11] contends that an efficient model is necessary for the investigation of self-built data. The paper claims that the Extreme Learning Machine method is combined with the Convolutional Neural Network model. The CNN-ELM model, which is used in deeplearning4j (DL4J), outperforms traditional CNN models in terms of accuracy and computing efficiency.

A significant aspect of the study [12], is its focus on accuracy. They therefore defend the CNN model's application by pointing out that it excels at identifying patterns and features in image analysis. The model is broken down into four phases, according to the paper: segmentation, classification, feature extraction, and pre-processing. The TensorFlow library is used to generate the CNN model for recognition. They started by converting the numerical string to a single digit. In [12], The OpenCV library was utilized to segment the string. The OpenCV BGR2GRAY function was used to transform the colored image into a greyscale version. The digits were divided and then given to the CNN model. An MNIST dataset that had already been processed was utilized to train the model. To meet the needs of the neural network, every image in this dataset was transformed into a 4D tensor. Using the Conv2D command, a convolutional layer was additionally constructed. The images' local patterns and characteristics are captured by the convolutional layer. A 3D patch from the feature map is captured by the convolutional layer's filter sliding across the height, breadth, and depth. Up to the filter's ultimate placement, this is done. Pooling is the following stage. The amount of information gleaned from the photos is diminished. From the collection of nearby pixels, it extracts information, retaining only the most crucial details and eliminating the rest. The pooling process was carried out independently on each channel, using either maximum pooling or average pooling as the pooling rules. Adam optimizer was used to train the model with sparse_categoricalcrossentropy loss. Training pictures and models were used to train the model over five epochs. With a final training accuracy of 99.36% and a testing accuracy of 99.15%, CNN met these requirements. Character recognition using an appropriate database that can be expanded to the HCR (Handwritten Character Recognition) domain is part of their project's other scope.

C. Comparative Analysis of Handwritten Digit Recognition using Various Classifiers.

Handwritten digit recognition has vast practical applications and financial implications. A recognition rate with 100% accuracy is unbelievable for a Handwritten digit recognition system. In this article [14], the researcher proposed CNN based on the Keras model to increase the convolutional layer with pooling and dropout and tuned the model using the filter. The proposed CNN model got 99.06% training accuracy and a testing accuracy of 98.80% with epoch 10.

The results of the experiment in the paper [15], reveals that the proposed CNN is more effective compared to other techniques. Many studies on Arabic handwritten digit recognition have already been carried out. A two-stage classifier was also designed to get higher recognition rates.

Among the traditional classifiers, it was the SVM that showed the best results, whereas CNN was the one to achieve the best results among the investigated techniques. After a hundred experiments it was found that SVM was comparatively better than the k-NN (>10%) and the best accuracy among the traditional classifiers based on all investigated features was 99.3% accuracy obtained by the SVM, and the CNN achieves the best overall accuracy of 99.45%.

In this article [16], the researcher proposed a novel approach for offline handwritten digit recognition classifying the digits into four regions, i.e. upper, lower, left, and right. Images were identified by the curves in these four regions. This method was successfully used for a bunch of databases like NIST and MNIST. On the MNIST database, the best recognition rate of 99.65% has been obtained by using a 6-layer Neural Network. It also showed that the CNN-features + SVM combination generated the highest accuracy. Both CNN and SVM have already achieved superb performances in digit recognition, In the article [16], researchers focused numerous research on their fusion to bring out their best qualities. The experimental results show that the classifier fusion can achieve a classification accuracy of ≥ 98%.

In the article [17], provides a comparative analysis of different classifiers for handwritten digit recognition, which goes as SVM, Random Forest ANN, and K-Means Algorithm. SVM offers robust and effective performance and is based on statistical learning techniques. With 372.29 seconds of training time and 97.77 seconds of test time, they are reaching 90% accuracy. The trained decision tree algorithms that make up Random Forest. After 63.05 seconds of training and 1.10 seconds of testing, they are hitting 97% accuracy. An artificial neural network (ANN) can be used to solve problems involving people's decision-making processes, such handwritten digit recognition. With 178.39 seconds of training time and 1.01 seconds of test time, they are reaching 97% accuracy. K-Means performed really well in the tests and had very few clustering errors. 98% accuracy can be attained in 8.35 training seconds and 0.56 test seconds.

III. Problem Statement & Objectives

Problem: Handwritten digit recognition is a very crucial task, and it has applications in various fields, such as digitizing historical documents, automated postal sorting, and improving accessibility for individuals having visual impairments. However, using existing approaches, reaching high accuracy and robust performance is still a concern when dealing with different writing styles and complex patterns in handwritten digit recognition.

Objectives: To improve the performance of existing methods and to achieve greater accuracy in handwritten digit recognition: Develop and implement a Convolutional Neural Network (CNN) architecture to effectively recognize complex patterns in handwritten digits.

To improve the overall performance metrics: Continuously investigate and implement new techniques and strategies to optimize training parameters, data preprocessing techniques, and model hyperparameters to achieve desired precision, recall, and overall efficiency in digit recognition.

IV. Methodology

The process diagram for our approach, which begins with user-provided photos, is depicted in Figure 4. facilitates training with the MNIST dataset. The model evaluates its accuracy and looks at how well it recognizes digits by utilizing a Convolutional Neural Network (CNN) for classification. Performance in digit identification from user-provided photos using the MNIST dataset is guaranteed by this methodical approach.

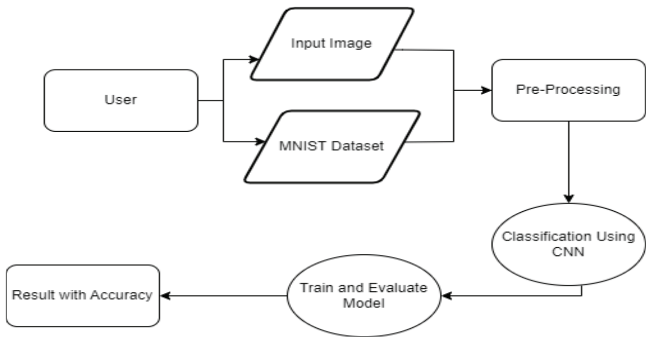


Fig. 4. Flowchart

CNN is a crucial instrument for image classification. MNIST-like datasets yield good performance from convolutional neural networks. It works for feature extraction through some levels and is more effective at capturing spatial relationships. But it isn't always relevant, thus we have to pick the model based on the available information, complexity, and resources.

A. Dataset

The MNIST handwritten digit database is the dataset we use in this paper for handwritten digit recognition. Matching learning is the primary application of the MNIST dataset. Each of these 28 × 28 pixel makes about 70,000 grayscale in total. There are ten distinct classes in all, which correspond to the digits 0 through 9. After that, the dataset is divided into 10,000 for the test set and 60,000 for the training set. Using the label, the training dataset teaches the model how each digit should look. Next, by inserting the image into the model, the test dataset determines if the model can accurately predict the image that it has never seen. The grayscale images in Figure 5 are from the MNIST collection, which has 28 × 28-pixel measurements. From 0 to 9 are the numbers that are included in it.

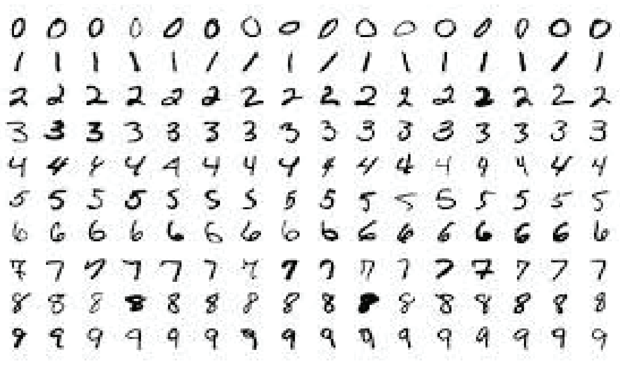


Fig. 5. MNIST Dataset [23]

B. CNN

Convolutional neural networks integrate deep learning with artificial neural networks. It has been used for many years for image recognition jobs. For instance, the topic of our work was handwritten digit recognition. CNNs are the first effective deep learning method that use multilayer hierarchical structure networks.

Convolution is the initial layer, which is used to extract features from the images. By employing a lower pixel filter, it reduces the image's size without sacrificing the relationship between pixels. A method utilized in deep learning and image processing is the convolution operation, as seen in Figure 6. To make a filtered output, multiplying the corresponding elements of an input signal after filtering it, and then adding them up is the process. This technique is frequently applied to extract characteristics.

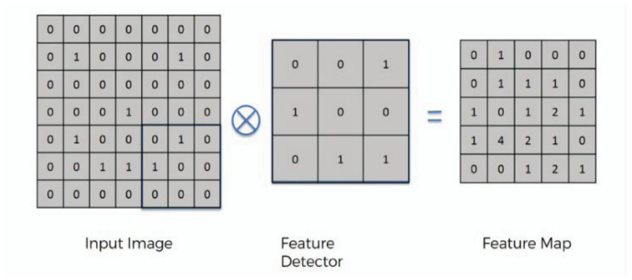


Fig. 6. Convolution Operation [1]

Max pooling is utilized in order to decrease the spatial dimensions of the input feature maps. The process is called down sampling. A speedier training and inference process is achieved by downsampling the feature maps, which lowers the computational cost of the network. Max pooling is a type of downsampling, as seen in Figure 7, in which the input data is picked to yield the maximum value within a given window. For applications like image recognition and convolutional neural networks, this helps minimize the size of the data while maintaining features.

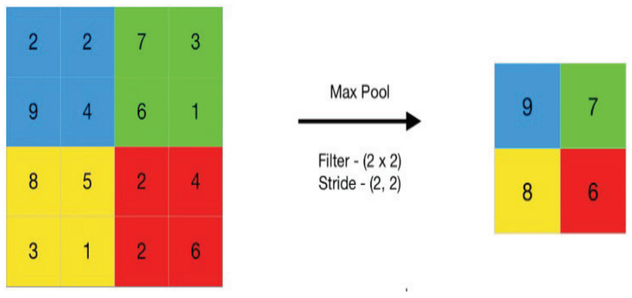


Fig. 7. Max Pooling Operation [1]

Layers that are fully connected establish a dense network of connections by linking each neuron in one layer to every other layer's neuron. At the end, a fully connected layer is built for picture classification, and time and space complexity can be decreased by using convolutional and pooling layers. Neurons in one layer are connected to all other neurons in the other layer by neural networks, as illustrated by Fully Connected Layers in Figure 8. In doing so, data from all layers is combined to allow the network to discover patterns and relationships.

The TensorFlow Python package is used to create this CNN model. The colored image has to be first converted to greyscale using the OpenCV library's BGR2GRAY function. The image was displayed on the screen using the Malplotlib library. The image was passed to the CNN model after preparation. After that, each image was normalized to a pixel value between 0 and 1 by importing the pre-processed MNIST dataset. Then, using the Conv2D command, the convolutional

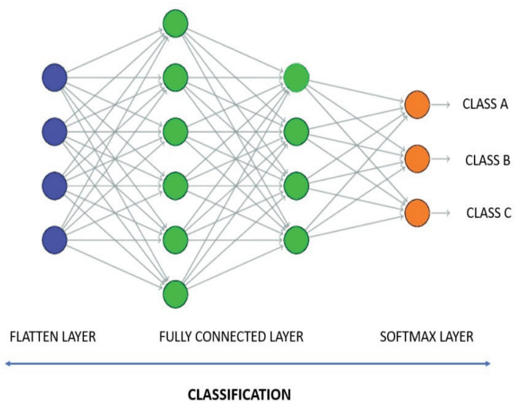


Fig. 8. Fully Connected Layers [1]

layer was added. The input data's local patterns and features are captured by the convolutional layer. Details are captured by the convolutional layer using a filter that moves across the image's height, breadth, and depth. The process persists until the filter is finally positioned. That's followed by applying the MaxPooling filter. The information's size is decreased by pooling, which functions as a filter. The algorithm collects data from the set of adjacent pixels, retaining only the most crucial elements and eliminating the rest. With a stride of two, pooling is done using a 2 x 2 window. This process is done on each channel independently. Upon applying maximum pooling of 2 x 2 to an output of 26 x 26 x 1, 13 x 13 x 1 is the consequent output. Thus, half of the information is decreased, as can be seen. After convolutional and pooling layer output is converted into a vector to be passed on to fully connected layers, it is passed through a flattened layer. A one-dimensional array is created by this layer using the multi-dimensional output from the preceding pooling layers. It stacks all the values to accomplish this. A three-dimensional array with height, breadth, and channels, for instance, would be converted to a one-dimensional array by the flattened layer. These dimensions multiplied by one will be its length. Layers with full connectivity employ the ReLU activation mechanism. With a SoftMax activation for categorization, its 128 neurons are located in the last layer. Weights and biases associated with these layers are learned during training, allowing the model to be optimized for the particular job at hand. As seen in Figure 9, Using layers to extract features and pooling layers to downsample, followed by connected layers to capture high-level information, is our suggested CNN architecture for digit recognition on the MNIST dataset. Accuracy and performance are intended to be optimized by this design..

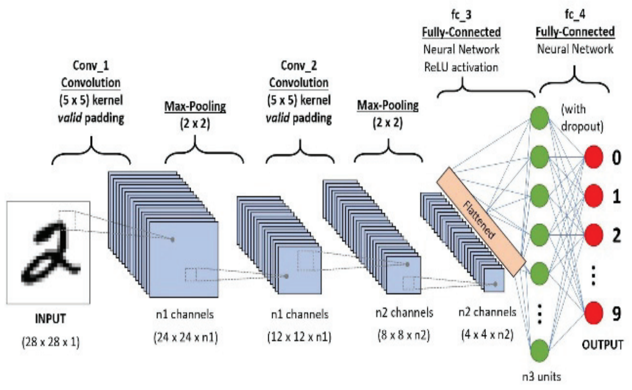

Conv_1 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
Conv_2 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
fc_3 Fully-Connected Neural Network ReLU activation
fc_4 Fully-Connected Neural Network
(with dropout)
Flattened
INPUT (28 x 28 x 1)
n1 channels (24 x 24 x n1)
n1 channels (12 x 12 x n1)
n2 channels (8 x 8 x n2)
n2 channels (4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

Fig. 9. The Architecture of our Proposed CNN [21]

Ease of use is the intended motivation for TensorFlow. For performing intricate mathematical operations, this low-level tool is utilized. The majority of the time, they attempt to pick up knowledge from experts by working around them, building different learning structures along the way, and ultimately turning everything into programming that is functional. Thus, it can be conceptualized as a programming framework in which computations are represented as graphs, nodes are math operations, and edges are tensors that relate the operations. Reduce the number of lines of code by using Python's many built-in functions Because of its simplicity, it is a useful coding language for software engineers. One language that allows you to work quickly is Python. It can be applied to efficient coding. Just like the Conda package and virtual environment director, Anaconda Navigator, Anaconda comes with more than 1,400 packages, eliminating the need to figure out how to add each library freely. The Anaconda appropriation has a graphical user interface (GUI) called Anaconda Navigator, which allows users to manage condo packages, conditions, and channels and launch applications without the need for command line instructions.

V. Implementation & Results

The MNIST dataset is being chosen since it is widely used in handwritten digitization recognition; therefore, before conducting any more study, we need assess the accuracy and performance of our approach. To begin, use the Keras programming interface to locate and select the dataset. The MNIST dataset's images all have levels and 28 by 28 values. Prior to preprocessing utilizing the Keras programming interface, the user-provided input and MNIST dataset [23] are entered. We train and assess our models using CNN classification, and we do this task quickly and effectively by utilizing the Python framework. Next, the accuracy and outcome are obtained.

For quick reduction, the initial stage involves reducing the size of the photos to 28 × 28-pixel. Step 2 is called Convolution 1 and is where we ensure that there is no padding at all. The end product is compressed into 24 by 24 pixels while maintaining its original dimensions. Max pooling is the method used to compress this image into 12 × 12-pixel. Using convolution and max pooling, the resulting image is then shrunk to 4 by 4 pixels. Ultimately, in order to use CNN, the Rectified Linear Activation Function (ReLu) is enabled. For each pixel in the ReLu activation process, a value of 0 or 1 is assigned based on the previous filter technique. These values are flattened prior to transmission to the MNIST data collection. It yields the desired result after being compared to its data collection.

A. Loading the Dataset

For the purpose of developing, training, and deploying machine learning models, a library called the Keras API was created. For loading the dataset, it offers a simple technique. This makes deep learning model development and testing simple. Downloading and using an MNIST dataset in the software is relatively easy to perform with the Keras API.

B. Easy CNN for MNIST

MNIST may now be used to build a convolutional neural network, which will be used to train the model. This section implements a basic CNN that demonstrates all the features of a standard CNN implementation, such as pooling layers, dense and dropout layers, and convolutional layers. Importing the necessary classes and functions is the first step. The loading of MNIST is the next step. It is reshaped in accordance with the CNN model's training requirements. It will seem as though you are receiving three input images for each color image because there are three pixels in the RGB for the blue, green, and red components. Photographs are greyscale with pixel values set to 1. Prior to that, the values of the pixels are normalized to fall between 0 and 1. Convolutional2D is the structure that the first hidden layer uses to function as a convolutional layer. This layer has 32 5 x 5 feature maps and a corrected activation function implemented. This is the layer of input where raw data is received for processing. Additionally, a pooling layer known as MaxPooling2D is defined. It is specified with a 2 x 2 pool size. The Dropout layer is the next layer that regularization is achieved with. By deleting neurons, it lessens overfitting. Subsequently, the flattened layer is employed to convert the two-dimensional output derived from convolutional and pooling layers into a single-dimension vector, which is then sent to fully connected layers. The completely connected layers come next. It comprises 128 neurons. The rectifier activation function is utilized. Ten neurons total, each of which represents one of the ten classes, make up the output layer. It adds non-linearity by incorporating a SoftMax activation function. Predicting the results for every class is helpful.

Algorithm 1: CNN Digit Classifier

```
/*  Import necessary libraries  */
        import_library("deep_learning_library")
        import_library("plotting_library")
        import_library("numerical_library")
/*  Load the dataset  */
        train_data, test_data = load_dataset ()
/*  Explore and preprocess the data.  */
        preprocess_data(train_data)
        preprocess_data(test_data)
/*  Build a neural network model.  */
        model = create_neural_network ()
/*  Add convolutional layers.  */
        add_convolutional_layers(model)
/*  Add activation functions for non-linearity.  */
        add_activation_functions(model)
/*  Add max-pooling layers.  */
        add_maxpooling_layers(model)
/*  Flatten the output and add dense layers.  */
        flatten_and_add_dense_layers(model)
/*  Compile the model.  */
        compile_neural_network(model)
/*  Train the model.  */
        train_neural_network (model, train_data)
/*  Evaluate the model on testing data.  */
```

```
        test_loss, test_accuracy=**evaluate_model**
        (model, test_data)
        **print** ("Test Loss:", test_loss, "Test Accuracy:",
        test_accuracy)
/*  Make predictions on a new input.  */
        new_input=
        **process_new_input**('path_to_new_input')
    **preprocess_new_input**(new_input)
/*  Use the trained model to predict.  */
        prediction=**make_prediction**(model,
        new_input)
    **print** ("Predicted Output:", prediction)
```

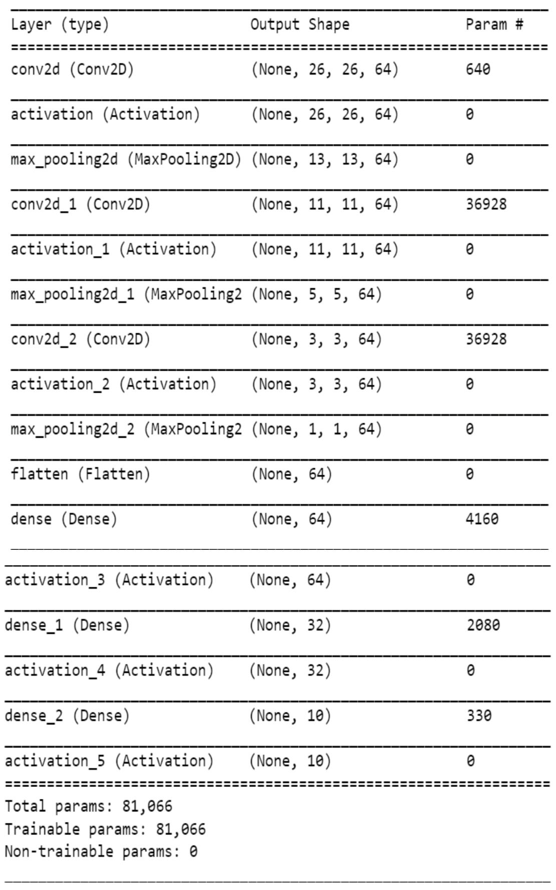
Algorithm 1 provides a basis for image recognition applications and offers insights into the handwritten digit categorization task. It can use the OpenCV library to load and preprocess external images in order to generate predictions about them. The method it develops is based on CNNs and digit categorization. Recognizing photographs of handwritten numbers 0 through 9 is its primary goal. The input photographs are used to teach this architecture about hierarchical characteristics.

In the first step, the import of the necessary libraries has occurred. For deep learning, these libraries are required. They are helpful for mathematical calculations and data visualization. Deep learning libraries like TensorFlow or PyTorch are employed in an actual implementation. The graphing library Matplotlib is utilized. NumPy is the library used for numerical operations. Step two involves loading the dataset into the RAM. Using TensorFlow's Keras API, the MNIST (Modified National Institute of Standards and Technology) dataset is loaded. Here, the third phase involves preparing the training and testing data as well as exploratory data analysis. Verifying the data's form is a step in the exploratory data analysis process. It entails comprehending the properties of the data and displaying it visually. Normalization is one of the preprocessing stages. It transforms data into a format that is appropriate for the neural network and handles missing values. The image's pixel values are normalized within the interval of 0 to 1.

In the fourth stage, a neural network model instance is created. It provides details about the neural network's architecture. From this point on, the CNN (Convolutional Neural Network) is constructed. The number of layers is specified. The kind of layers (dense, convolutional, etc.) and their arrangements are also described. Neural network construction is done using TensorFlow's Keras Sequential API. Convolutional layers are frequently employed in assignments involving images in the fifth phase. The neural network model gains convolutional layers in this step. These layers would be added using the code included in the add_convolutional_layers() function. Parameters like the number of filters, kernel size, and activation functions are specified. Convolutional layers pick up specific local patterns and characteristics in the incoming data during this process. Here, in the sixth stage, the neural network gains non-linearity from the activation functions. It can pick up intricate patterns. The neural network would get an activation function through the use of the add_activation_functions() method. An activation function known as the "ReLU" is utilized to construct convolutional neural networks. Rectified Linear Unit is what it stands for. It facilitates the introduction of non-linearity and enhances computational effectiveness.

In the seventh step, Max-Pooling layers are added. Max-pooling layers aid in lowering the data's spatial dimensions. It records the main characteristics. To downsample the output from the convolutional layers, the add_maxpooling_layers() function would add max-pooling layers. It takes the supplied image and extracts its features. Dense and flattened layers are added in eight stages. The multidimensional output from earlier levels is flattened to create a one-dimensional array. Layers that are completely connected are called dense layers. The output would be flattened and dense layers would be added to the model using the flatten_and_add_dense_layers () function. In order to pass the output to fully connected layers, flatten layers take the output from convolutional and pooling layers and turn it into a vector. Configuring the model's learning process is the ninth step in the process of compilation. Namely, the optimizer needs to be specified. The loss function and metrics are also specified. The code to assemble the model would be in this phase. Using sparse categorical cross entropy as the loss function and the Adam optimizer, the model is assembled.

Providing the neural network with training data is the eleventh phase in the model's training process. Parameterizing the model in accordance with the error also falls under this. Training would be handled by the train_neural_network() method. The number of epochs and other training parameters are specified in order to achieve this. The trained model is assessed on test data that hasn't been seen yet in the eleventh



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 64)        640
_________________________________________________________________
activation (Activation)      (None, 26, 26, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
activation_1 (Activation)    (None, 11, 11, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
activation_2 (Activation)    (None, 3, 3, 64)          0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
activation_4 (Activation)    (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                330
_________________________________________________________________
activation_5 (Activation)    (None, 10)                0
=================================================================
Total params: 81,066
Trainable params: 81,066
Non-trainable params: 0
_________________________________________________________________
```

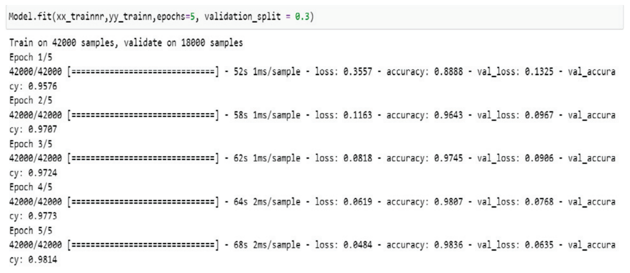
Fig. 10. Model Summary of CNN to be applied

stage. The evaluate_model () function would contain code for calculating test accuracy and loss. Preparing a new input for forecasting represents the eleventh step. This stage would take care of preprocessing and reading a fresh input. Make assumptions about a fresh picture. OpenCV is used to read an image, convert it to grayscale, resize it to the necessary dimensions, normalize the pixel values, and then reshape it to fit the input geometry of the model.

At last, predictions on the fresh input can be made using the trained model. Code for running the fresh input through the trained model and obtaining the expected output is contained in the make_prediction() method.

Figure 10 shows the model summary that will be applied to the images of the MNIST dataset and defines the output shapes and parameters after applying the convolutional neural network strategies.

Figure 11 shows the training of the model for 5 epochs per iteration with a validation split of 30%. The model is trained on 42000 samples and validated on 18000 samples.



```
Model.fit(xx_trainnr,yy_trainn,epochs=5, validation_split = 0.3)

Train on 42000 samples, validate on 18000 samples
Epoch 1/5
42000/42000 [==============================] - 52s 1ms/sample - loss: 0.3557 - accuracy: 0.8888 - val_loss: 0.1325 - val_accura
cy: 0.9576
Epoch 2/5
42000/42000 [==============================] - 58s 1ms/sample - loss: 0.1163 - accuracy: 0.9643 - val_loss: 0.0967 - val_accura
cy: 0.9707
Epoch 3/5
42000/42000 [==============================] - 62s 1ms/sample - loss: 0.0818 - accuracy: 0.9745 - val_loss: 0.0906 - val_accura
cy: 0.9724
Epoch 4/5
42000/42000 [==============================] - 64s 2ms/sample - loss: 0.0619 - accuracy: 0.9807 - val_loss: 0.0768 - val_accura
cy: 0.9773
Epoch 5/5
42000/42000 [==============================] - 68s 2ms/sample - loss: 0.0484 - accuracy: 0.9836 - val_loss: 0.0635 - val_accura
cy: 0.9814
```

Fig. 11. CNN Model Training

Figure 12 demonstrates the code in which the image of digit '0' is passed to the model to predict the output.

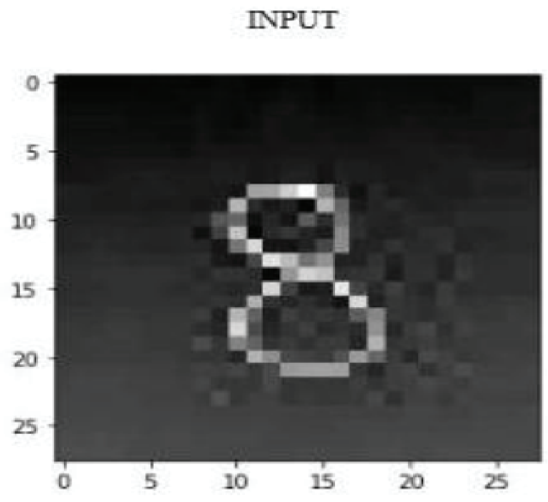
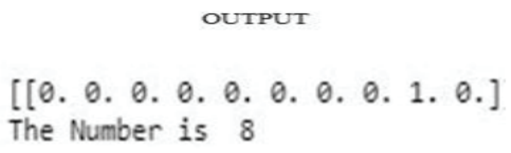


Fig. 12. Input image of digit 8.

Figure 13 shows the predicted output by the trained model. The image of the digit is picked from the mentioned location. The colored image is converted to greyscale by the BGR2GRAY function. The image is resized to a 28 × 28-pixel square using interlinear interpolation. Next, the pixel values are normalized from 0 to 1 and finally passed to the mode to predict the output.



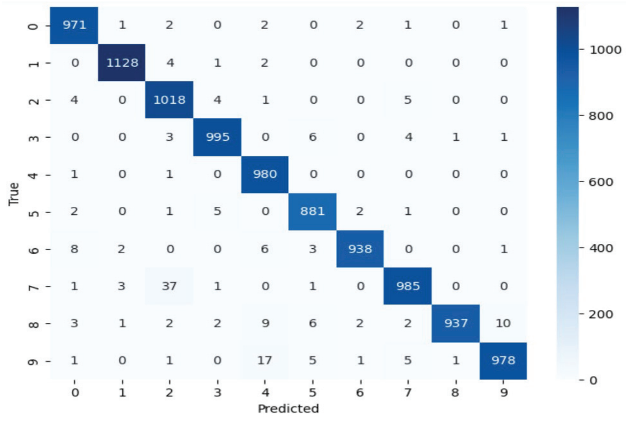
OUTPUT

```
[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
The Number is  8
```

Fig. 13. Predicting the output.

Using the CNN classifier the digit is compared and recognized. After the completion of all the processes, the result is displayed on the screen.

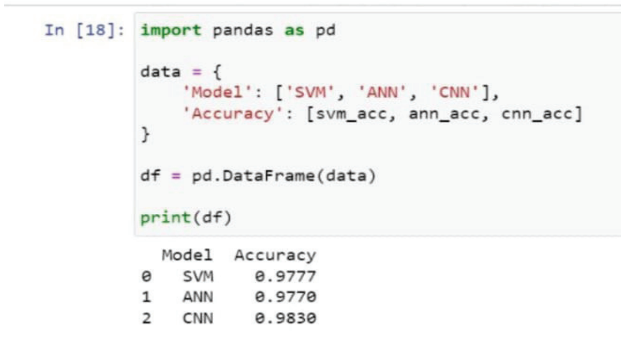
In Figure 14 the diagonal elements in the confusion matrix show the correct predictions. The values in the white cells show the wrong predictions. It clearly shows the correct predictions are higher than incorrect ones.



	0	1	2	3	4	5	6	7	8	9
0	971	1	2	0	2	0	2	1	0	1
1	0	1128	4	1	2	0	0	0	0	0
2	4	0	1018	4	1	0	0	5	0	0
3	0	0	3	995	0	6	0	4	1	1
4	1	0	1	0	980	0	0	0	0	0
5	2	0	1	5	0	881	2	1	0	0
6	8	2	0	0	6	3	938	0	0	1
7	1	3	37	1	0	1	0	985	0	0
8	3	1	2	2	9	6	2	2	937	10
9	1	0	1	0	17	5	1	5	1	978

Fig. 14. Confusion Matrix

There are many Machine learning algorithms. In this project ANN, SVM and CNN classifiers are used. They have been trained and tested. These are trained and tested on same MNIST dataset. Figure 15 shows the comparison of classifiers. These deep learning techniques are used to achieve a high amount of accuracy. In contradiction to other research methods, this one focuses on higher accuracy. This method checks which classifier achieves accuracy more than 98%. In this research, Keras is used as backend and TensorFlow. This CNN model is capable of producing accuracy of 98% while ANN could create accuracy of 97.70% and SVM could produce 97.77 % accuracy. It could be said that CNN produced the best results with highest accuracy.



```
In [18]: import pandas as pd

         data = {
             'Model': ['SVM', 'ANN', 'CNN'],
             'Accuracy': [svm_acc, ann_acc, cnn_acc]
         }

         df = pd.DataFrame(data)

         print(df)

           Model  Accuracy
         0   SVM    0.9777
         1   ANN    0.9770
         2   CNN    0.9830
```

Fig. 15. Comparison of classifiers.

VI. Conclusion

From a researcher's perspective, image recognition is the area that is currently developing and expanding. According to the model, task performance, accuracy, and mistake rate may vary, it is discovered after investigation. Consequently, the model chosen becomes crucial. Therefore, the creation of handwritten digitization recognition was the main focus of this research, which employed deep learning techniques, a popular machine learning approach. Therefore, for photo classification, training a convolutional neural network is preferable. The model is trained and tested to get the highest accuracy achievable using the CNN technique and the MNIST dataset. Maximizing accuracy was the main goal. 98.36% accuracy may be attained by a CNN model using the Python TensorFlow Machine Learning Library and the Anaconda graphical user interface. The project's completion produced a desired outcome. The proposed model can be enhanced in the future to recognize handwritten characters in a variety of languages, including Bengali, Hindi, French, and English.

ACKNOWLEDGEMENT

I would like to thank Ms. Mouli Das (Assistant professor, Dept of CSE, Narula Institute of Technology, Agarpara) for her valuable suggestions and comments that helped improve this work, this support is greatly appreciated.

REFERENCES

[1] Hossain, M.A. and Ali, M.M., 2019. Recognition of handwritten digits using convolutional neural network (CNN). Global Journal of Computer Science and Technology, 19(2), pp.27-33.

[2] Alwzwazy, H.A., Albehadili, H.M., Alwan, Y.S. and Islam, N.E., 2016. Handwritten digit recognition using convolutional neural networks. International Journal of Innovative Research in Computer and Communication Engineering, 4(2), pp.1101-1106.

[3] Ahlawat, S., Choudhary, A., Nayyar, A., Singh, S. and Yoon, B., 2020. Improved handwritten digit recognition using convolutional neural networks (CNN). Sensors, 20(12), p.3344.

[4] Pomazan, V., Tvoroshenko, I. and Gorokhovatskyi, V., 2023. Handwritten character recognition models based on convolutional neural networks.

[5] El-Sawy, A., Loey, M. and El-Bakry, H., 2017. Arabic handwritten character recognition using convolutional neural network. WSEAS Transactions on Computer Research, 5(1), pp.11-19.

[6] Chakraborty, P., Islam, A., Abu Yousuf, M., Agarwal, R. and Choudhury, T., 2022. Bangla handwritten character recognition using convolutional neural network. In Machine Intelligence and Data Science Applications: Proceedings of MIDAS 2021 (pp. 721-731). Singapore: Springer Nature Singapore.

[7] Sufian, A., Ghosh, A., Naskar, A., Sultana, F., Sil, J. and Rahman, M.H., 2022. Bdnet: Bengali handwritten numeral digit recognition based on densely connected convolutional neural networks. Journal of King Saud University-Computer and Information Sciences, 34(6), pp.2610-2620.

[8] Garris, M.D., Wilkinson, R. and Wilson, C.L., 1991, July. Methods for enhancing neural network handwritten character recognition. In International Joint Conference on Neural Networks. (Vol. 1, pp. 695-700).

[9] Lauer, F., Suen, C.Y. and Bloch, G., 2007. A trainable feature extractor for handwritten digit recognition. Pattern Recognition, 40(6), pp.1816-1824.

[10] Al-Haija, Q.A., 2022. Leveraging ShuffleNet transfer learning to enhance handwritten character recognition. Gene Expression Patterns, 45, p.119263.

[11] Ali, S., Li, J., Pei, Y., Aslam, M.S., Shaukat, Z. and Azeem, M., 2020. An effective and improved CNN-ELM classifier for handwritten digit recognition and classification. Symmetry, 12(10), p.1742.

[12] Vinjit, B.M., Bhojak, M.K., Kumar, S. and Nikam, G., 2021. Implementation of handwritten digit recognizer using CNN. In Workshop on Advances in Computational Intelligence at ISIC.

[13] Niu, X.X. and Suen, C.Y., 2012. A novel hybrid CNN–SVM classifier for recognizing handwritten digits. Pattern Recognition, 45(4), pp.1318-1325.

[14] Agrawal, A.K., Shrivas, A.K. and kumar Awasthi, V., 2021, May. A Robust model for handwritten digit recognition using machine and deep learning technique. In 2021 2nd International Conference for Emerging Technology (INCET) (pp. 1-4). IEEE.

[15] Akhtar, M.S., Qureshi, H.A. and Alquhayz, H., 2019. High-quality wavelets features extraction for handwritten arabic numerals recognition. Int. J. Adv. Sci. Eng. Inf. Technol, 9, pp.700-710.

[16] Bishnoi, D.K. and Lakhwani, K., 2012. Advanced approaches of handwritten digit recognition using hybrid algorithm. International Journal of communication and computer Technologies, 1(1), pp.45-50.

[17] KARAKAYA, R. and KAZAN, S., 2021. Handwritten digit recognition using machine learning. Sakarya university journal of science, 25(1), pp.65-71.

[18] Daniel, R., Prasad, B., Pasam, P.K., Sudarsa, D., Sudhakar, A. and Rajanna, B.V., 2024. Handwritten digit recognition using quantum convolution neural network. Int J Artif Intell, 13(1), pp.533-541.

[19] Shopon, M., Mohammed, N. and Abedin, M.A., 2016, December. Bangla handwritten digit recognition using autoencoder and deep convolutional neural network. In 2016 International Workshop on Computational Intelligence (IWCI) (pp. 64-68). IEEE.

[20] Balaha, H.M., Ali, H.A., Saraya, M. and Badawy, M., 2021. A new Arabic handwritten character recognition deep learning system (AHCR-DLS). Neural Computing and Applications, 33, pp.6325-6367.

[21] Ratan, P. (2023) What is the Convolutional Neural Network Architecture?, Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/ (Accessed: 08 January 2024).

[22] Beohar, D. and Rasool, A., 2021, March. Handwritten digit recognition of MNIST dataset using deep learning state-of-the-art artificial neural network (ANN) and convolutional neural network (CNN). In 2021 International conference on emerging smart computing and informatics (ESCI) (pp. 542-548). IEEE.

[23] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998. The MNIST Database of Handwritten Digits. Available at: http://yann.lecun.com/exdb/mnist/ [Accessed: 18 January 2024.