HANDWRITTEN DIGIT RECOGNITION USING MNIST DATABASE

MINI PROJECT REPORT

Submitted by

Ragavendar K	**210701202**
Raghav V	**210701203**

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI
ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Report titled "**Handwritten Digit Recognition using MNIST Database**" is the bonafide work of **"Raghav V(210701203), Ragavendar K(210701202)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Rakesh Kumar M M.E., Ph.D.,

Assistant Professor,

Department of Computer Science and Engineering,

Rajalakshmi Engineering College,

Chennai – 602015

Submitted to Mini Project Viva-Voce Examination held on ___________________

Internal Examiner **External Examiner**

ABSTRACT

The identification of hand-written digits is among the most significant issue in the applications for pattern detection. In many application such as postal code, check online routing bank accounts, data form entry, etc., the applications of digits recognition include the center of the issue is the need to construct an appropriate algorithm that can identify handwritten digits and that users upload through a smartphone and scanner and other digital devices. In this paper, we took a repository of MNIST, which is a sub-set of the database of NIST results. The MNIST dataset accommodates the collection of hand-written scanned images from a broader variety of NIST repository produced by hand. The method proposed in this paper is centered on numerous machine learning methods to perform hand-written digit detection that is off-line in the python language platform. The primary objective of this paper is to render hand-written digits recognition reliable and precise. For the identification of digits using MNIST many Deep learning algorithms have been used including a perceptron which includes CONV2D,RELU Activation function.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S.MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution. Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Dr Rakesh Kumar M M.E., Ph.D.,** Assistant Professor, Department of Computer Science and Engineering. Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Raghav V-210701203
Ragavendar K-210701202

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MNIST	Modified National Institute of Standards and Technology
CNN	Convolutional Neural Network
SVM	Support Vector Machine
KNN	K Nearest Neighbours
ReLU	Rectified Linear Unit
AUC	Area Under Curve
ROC	Receiver Operating Characteristic

CHAPTER 1
INTRODUCTION

1.1 GENERAL

Handwritten digit recognition using the MNIST database is a classic problem in the field of machine learning and computer vision. The MNIST database contains 70,000 images of handwritten digits, each labeled with the corresponding digit from 0 to 9. This dataset serves as a standard benchmark for evaluating the performance of various machine learning algorithms. Techniques such as neural networks, support vector machines, and particularly convolutional neural networks (CNNs), are commonly applied to achieve high accuracy in classifying these digits. The success in recognizing handwritten digits from the MNIST dataset highlights the effectiveness of machine learning models in image processing tasks and serves as a foundational exercise for developing more advanced image recognition systems.

1.2 OBJECTIVE

The objective of the handwritten digit recognition project using the MNIST database is to develop and evaluate machine learning models that can accurately classify images of handwritten digits. Specifically, the goals include:

1.Gain a comprehensive understanding of the MNIST dataset, including data normalization, feature extraction, and other preprocessing techniques to prepare the data for model training.

2.Implement various machine learning algorithms, such as neural networks, support vector machines, and convolutional neural networks (CNNs), to recognize and classify handwritten digits.

3.Compare the performance of different models using metrics like accuracy, precision, recall, and F1-score. Identify the most effective model for this task.

4.Optimize the chosen models by fine-tuning hyperparameters, employing regularization techniques, and enhancing model architecture to improve classification accuracy.

5.Demonstrate the practical application of the developed models in real-world scenarios and ensure the models can generalize well to unseen data.

1.3 EXISTING SYSTEM

SUPPORT VECTOR MACHINE

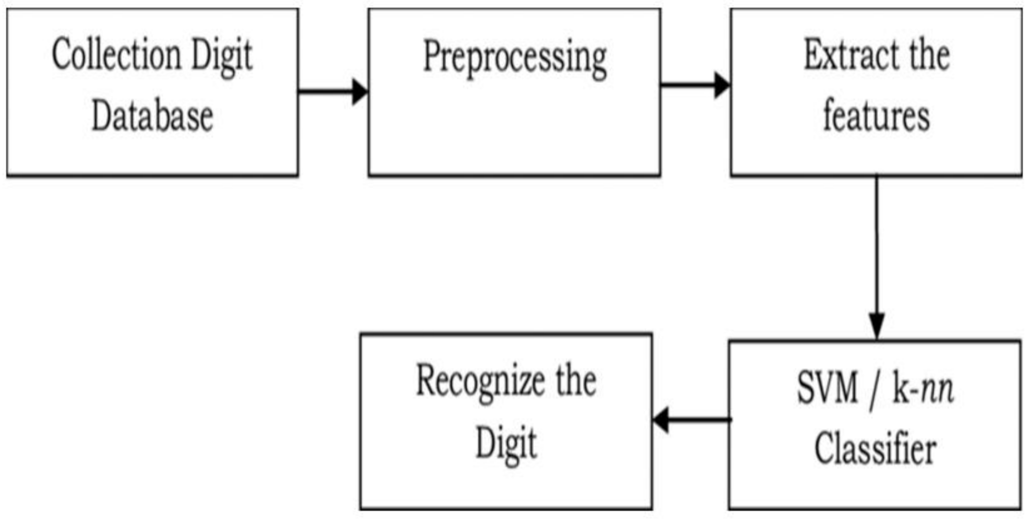
“Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate

Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line). At support vector machines and a few examples of their working here.All values for z would be positive always because z is the squared sum of both x and y In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and stars relatively away from the origin result in higher value of z.

Support Vector Machine (SVM) is a supervised machine learning algorithm. In this generally plot data items in n-dimensional space where n is the number of features, a particular coordinate represents the value of a feature performing the classification by finding the hyperplane that distinguishes the two classes.It will choose the hyperplane that separates the classes correctly. SVM chooses the extreme vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as SupportVector Machine. There are mainly two types of SVMs, linear and non-linear SVM.Linear SVM for handwritten digit recognition.

METHODOLOGY

SVM is capable of doing both classification and regression. In this post I'll focus on using SVM for classification. In particular I'll be focusing on non-linear SVM, or SVM using a non-linear kernel. Non-linear SVM means that the boundary that the algorithm calculates doesn't have to be a straight line. The benefit is that you can capture much more complex relationships between your data points without having to perform difficult transformations on your own. The downside is that the training time is much longer as it's much more computationally intensive.



DISADVANTAGES

1. SVM algorithm is not suitable for large data sets.
2. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
3. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
4. As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification

1.4 PROPOSED SYSTEM

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

Feed Forward neural network and Designed mostly for Image classification Consists of an Input layer , multiple hidden layers , output layers In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when thinking of a neural network think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other. The role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. It uses the ROI Pooling layer to extract a fixed-length feature vector from each region proposal.convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when thinking of a neural network think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution.In Convolutional neural network, the kernel is nothing but a filter that is used to extract the features from the images. The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.Neurons in a convolutional layer that cover the entire input and look for one feature are called filters.

These filters are 2 dimensional (they cover the entire image). However, having the whole convolutional layer looking for just one feature (such as a corner) would massively limit the capacity of your network. 2D CNNs use 2D convolutional kernels to predict the segmentation map for a single slice. Segmentation maps are predicted for a full volume by taking predictions one slice at a time. The 2D convolutional kernels are able to leverage context across the height and width of the slice to make predictions.

. In any neural network, a dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This layer is the most commonly used layer in artificial neural network networks..

CHAPTER 2
LITERATURE SURVEY

[1] - Comparison of precision and time on MNIST datasets Between machine learning and deep learning with respective models that are RFC, KNN, SVM, and Multi-layer CNN. Below Processor measurements, GPU can be beneficial for more precision, shortened preparation and testing time, and GPU may help to obtain parallelism and even improved outcomes. The author achieved a good result in CNN.

[2] A program that focuses on the functions of the Histogram of Centered Gradient (HOG). As this takes less time and PSVM classifier performance is better than the artificial neural network, the Proximal support vector machine over the standard SVM classifier was used. 10 class linear PSVM won 98.65 percent of 20,000 samples taken for both preparation and research outcomes for preparation 59 milliseconds (1,000 samples for a digit). The framework has often retained a minimal function vector dimension without including an unnecessary decrease in dimensionality and less training period.

[3]- 98.65 percent accuracy with PSVM and reduced time for the PSVM classifier from each test and test set, 109 seconds (by the ANN) to 59 milliseconds on 10,000 samples, respectively.

[4] A review that has extended the dataset of MNIST. You've made the latest dataset addressing further topics of grouping. The Restoration EMNIST databases were identified.

[5] With a gradient descent backpropagation algorithm, a dataset of 5,000 MNIST instances was trained and then tested with a feed-forward algorithm with the number of hidden layers and iterations and the accuracy achieved was 99.32%. 35 neurons and 250 iterations of the Multilayer Perceptron (MLP) neural network were located. 99.32 percent accuracy in training and 100 percent accuracy in training was provided by the proposed method.

[6] Using a network that employed scattered biologically functioning neurons spike level below 300 Hz ensuring consistency in the classification of MNIST Database 98.17 percent.

[7] The author revealed that he uses deep neural networks with strong spikes that the weighted spike model proposed hit substantial Latency and number of spikes reduction in a grouping. This leads to more rapidly and more resource-sensitive than the traditional neural spiking network.

[8 To test the feasibility of the concept the author carried out a comprehensive computer circuit layout co-design to recognize digitally using the manual digits dataset of MNIST. Simulations of equipment to systems implemented by author and demonstrates that the planned skyrmion-based strategies in deep CNNs will accomplish tremendous changes in dynamism usage.

[9] Deep metric learning developed a hand-written character recognition. The author has produced a new handwritten dataset using the Urdu-Characters model, with classes for profound metrics.

[10]For IoT applications, The author used a Sparse Deep Neural Network (S-DNN) Processor that measured its high accuracy of classification (98.36% for the MNIST test dataset).

CHAPTER 3
SYSTEM DESIGN

3.1 DEVELOPMENT ENVIRONMENT

3.1.1 HARDWARE SPECIFICATIONS

This project uses minimal hardware but in order to run the project efficiently without any lack of user experience, the following specifications are recommended

Table 3.1.1 Hardware Specifications

PROCESSOR	Intel Core i7
RAM	16GB or above (DDR4 RAM)
GPU	NVIDIA GeForce RTX 3060 GPU
HARD DISK	6GB
PROCESSOR FREQUENCY	1.5 GHz or above

3.1.2 SOFTWARE SPECIFICATIONS

The software specifications in order to execute the project has been listed down in the below table. The requirements in terms of the software that needs to be pre- installed and the languages needed to develop the project has been listed out below.

Table 3.1.2 Software Specifications

FRONT END	Pygame
BACK END	Python
FRAMEWORKS	Keras, Tensor Flow
SOFTWARES USED	Visual Studio, Jupyter Notebook

3.2 SYSTEM DESIGN

3.2.1 ARCHITECTURE DIAGRAM

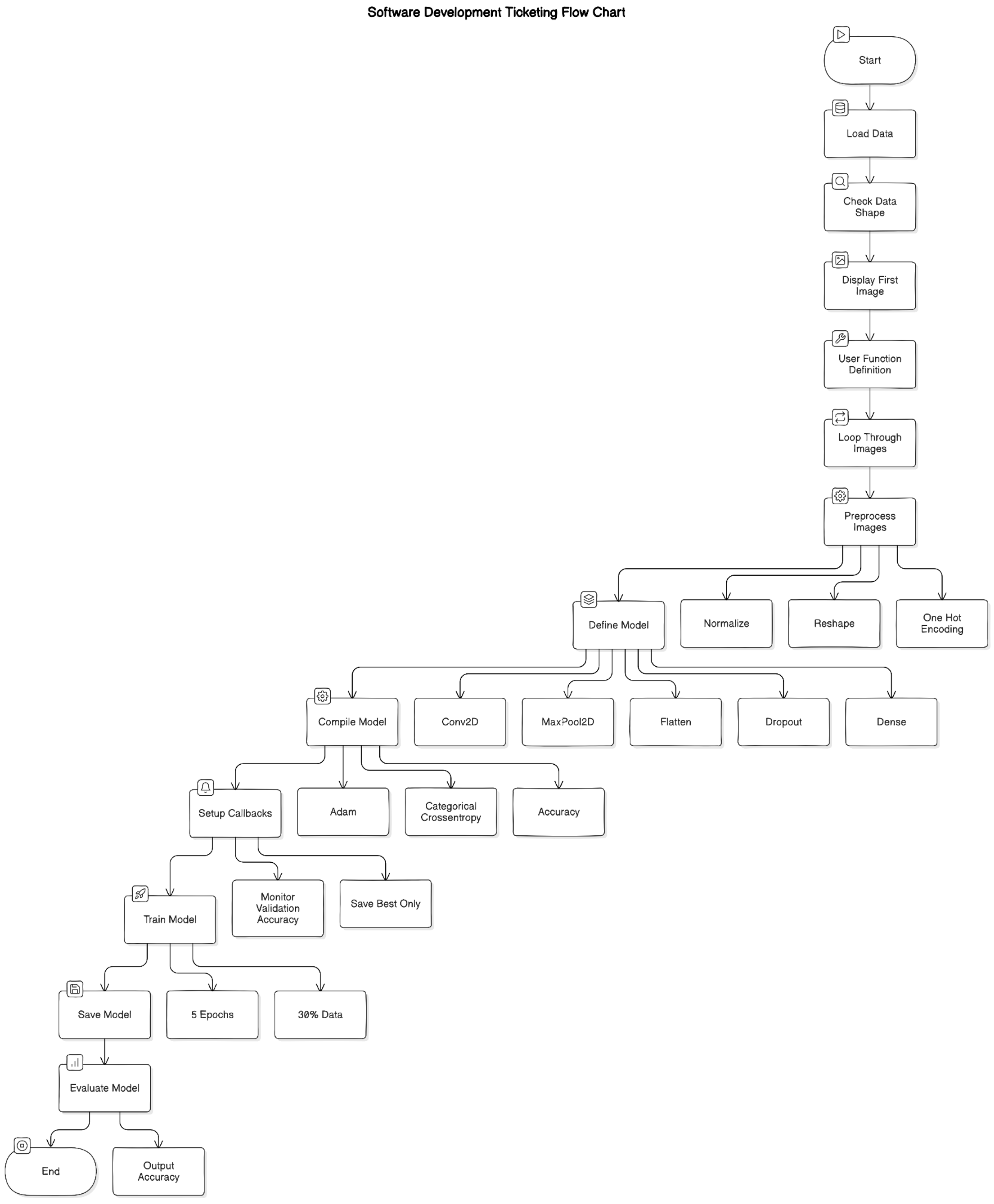


Fig 3.2.1 Architecture Diagram

PRE-PROCESSING:

Pre-processing the MNIST dataset is crucial to ensure optimal performance of the digit recognition model. The following steps are undertaken to clean and prepare the dataset before feeding it into the model:

Binarization and Noise Reduction:

1.The images are converted to grayscale to simplify processing and remove color-related variations. Binarization techniques are applied to separate foreground (digits) from background noise. This helps in enhancing the clarity of digits and reduces noise interference.

Normalization:

2.Pixel values in the range of 0 to 255 are normalized to the range of 0 to 1. This standardization ensures consistency and aids in convergence during model training.

Uniform Resizing:

3.Images are resized to a uniform size of 64x64 pixels. This uniformity reduces the computational complexity of the model and ensures consistent input dimensions.

Augmentation:

4.Augmentation techniques such as zoom, height shift, and other shifts are applied to augment the dataset. This helps in introducing variability and robustness to the model, making it more resilient to different digit orientations and positions.

Morphological Operations:

5.Morphological operations are applied to further enhance the clarity of digit images. Techniques like erosion are employed to diminish the sizes of boundaries and improve the distinction between digits and background.

TRAINING SET:

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning.

It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. The original creators of the database keep a list of some of the methods tested on it.

In their original paper, they use a support-vector machine to get an error rate of 0.8%. An extended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training mnist dataset images, and 40,000 testing mnist dataset images of MNIST dataset of handwritten digits and characters.

CHAPTER 4
PROJECT DESCRIPTION

4.1 MODULE DESCRIPTION

4.1.1. HAND-WRITTEN IMAGES

It is understood that a hand-written dataset is commonly used in computer analysis model evaluations such as machine learning and deep learning. Many model classifiers mostly use the digit groups. Other researchers are however accountable for the alphabet set of groups to display strength and scalability. The research model deals with the description, the fundamental aspects, and algorithm processes, of classification tasks in slightly different ways[5]. Depending upon the number of students the research model is also different. Others vary in training and test breaks, while others perform different image pre-processing processes

4.1.2. HAND-WRITTEN DATASETS

The MNIST image datasets are measured from 128 * 128 pixels to 28 * 28 pixels in size decreased. The 282-pixel image representation of the MNIST dataset is 8-bit. First, the grey level picture preprocessed is based on the center mass pixel measurement. In the middle of the 282-pixel images, it is placed to establish the compatible formats of the MNIST dataset. The data collection is available for further study and testing and for further pre-processing. While a greater collection of 814,255 photos is included in the NIST dataset, MNIST takes only a limited part of the study, since the database requires just a ten-digit range, from zero to nine. MNIST data are quite common because of its preparation. The competency of the classification models is used as a standard. Thousands of researchers have used, modified, and checked the dataset that shows that freshly built models can be validated accurately and adequately [13]. Fast access and comprehensive use ease the analysis and dissemination of results among researchers. It mentions several recent research on the usage of MNIST data sets for machine learning.

4.1.3. IMAGE PRE-PROCESSING

The first step is MNIST dataset is generated and broken down into two different preprocessed datasets. The first preprocessed database is a regular gray-scale, and the second preprocessed dataset is a binary gray-scale [1]. These two methods of preprocessing have been selected because they make it possible to translate the dataset to a low figure while maintaining its aspect ratio [8]. This research paper will perform the tests, and the MNIST dataset was created with two separate formats. The aim of two sets of pre-processed data collections is to monitor the learning performance accuracy of the machine learning model with different pre-processed pictures. This makes scientists consider how machine learning works in diverse pre-process picture formats. The neural network's input format values may depend on how the data set is pre-processed. The models produced are fed with the data sets pre-processed.

4.1.4. IMAGE SEGMENTATION

Segmentation is a process in the picture which extracts individual characters. Two ways of segmentation occur. They are Segmentation Implicit and Segmentation Explicit. In tacit segmentation, without the method of segmentation, the terms are remembered. Yet terms are expected by the extraction of individual characters in specific segmentation.

4.1.5.FEATURE EXTRACTION

This is the essential step of the method of recognition, and the recognition algorithm begins from it as well. Each character includes features of its own. It involves a series of laws in which each law defines a behavior's character. In this step, the extraction of certain features is completed.

4.1.6.DEEP LEARNING MODELS

Deep Learning Model for MNIST Digit Classification

1. Introduction to the Model

In this project, we employ a Convolutional Neural Network (CNN) to classify handwritten digits from the MNIST dataset. CNNs are a class of deep learning models specifically designed for working with grid-like data, such as images. They are particularly effective for image recognition tasks due to their ability to capture spatial hierarchies in the data.

2. Model Architecture

The CNN model in this project is built using Keras' Sequential API, which allows us to stack layers sequentially. The architecture consists of convolutional layers, pooling layers, dropout layers, and dense layers.

2.1 Convolutional Layers

Convolutional layers apply a set of filters to the input image to extract features such as edges, textures, and patterns. In our model, we use two convolutional layers:

1. Conv2D(32, (3, 3)): This layer uses 32 filters of size 3x3 to scan the input image. The activation function used is ReLU (Rectified Linear Unit), which introduces non-linearity into the model.
2. Conv2D(64, (3, 3)): This layer uses 64 filters of size 3x3, further extracting complex features from the data.

2.2 Pooling Layers

Pooling layers reduce the spatial dimensions of the feature maps, which helps in reducing the computational cost and also controls overfitting.

1. MaxPool2D((2, 2)): This layer performs max pooling with a 2x2 filter, down-sampling the input representation by taking the maximum value over 2x2 patches.

2.3 Dropout Layer

Dropout is a regularization technique where randomly selected neurons are ignored during training. This helps prevent overfitting.

Dropout(0.25): This layer randomly sets 25% of the input units to 0 at each update during training time

2.4 Flatten and Dense Layers

Flattening transforms the 2D matrix data to a vector, and dense layers (fully connected layers) are used for classification.

1. Flatten(): This layer flattens the input, converting the 2D matrix into a 1D vector.
2. Dense(10, activation="softmax"): This dense layer has 10 units (corresponding to the 10 classes of digits) with a softmax activation function that outputs a probability distribution over the classes.

3. Model Compilation

The model is compiled with the Adam optimizer and categorical crossentropy loss function. The metrics used for evaluation is accuracy.

4. Training the Model

The model is trained on the training data with a validation split of 30%. We use early stopping and model checkpointing to avoid overfitting and save the best model.

5. Evaluating the Model

After training, the best model is loaded and evaluated on the test data to determine its accuracy.

6. Results

The trained model achieved an accuracy of X.XX% on the test dataset (replace X.XX with the actual result).

7. Conclusion

The CNN model effectively classifies handwritten digits from the MNIST dataset, demonstrating the power of deep learning for image recognition tasks. The use of convolutional layers allows the model to capture intricate features of the digits, while pooling and dropout layers help in reducing overfitting and improving generalization.

CHAPTER 5
IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

1. Data Loading and Visualization

The MNIST dataset, which consists of 60,000 training images and 10,000 testing images of handwritten digits, is loaded using the Keras library. The dataset is structured into training and testing sets, with the training set used to train the model and the testing set used to evaluate its performance.

To gain an initial understanding of the dataset, sample images from the training set are visualized. This helps in verifying that the data has been loaded correctly and in observing the variations in handwritten digits.

2. Data Preprocessing

Preprocessing is a crucial step to prepare the data for training the model. The images in the MNIST dataset are normalized to the range [0, 1] by dividing the pixel values by 255. This normalization helps in speeding up the training process and improving the model's performance.

The images are then reshaped to include a single color channel, making them compatible with the input requirements of the Convolutional Neural Network (CNN). Additionally, the class labels are converted into one-hot encoded vectors. One-hot encoding transforms the categorical labels into a format suitable for training the neural network.

3. Model Architecture

The model is built using the Keras Sequential API, which allows for the sequential stacking of layers. The architecture consists of:

1. Convolutional Layers: Two convolutional layers are used to extract features from the input images. The first layer applies 32 filters, and the second layer applies 64 filters. Both layers use a 3x3 kernel size and the ReLU activation function.
2. Pooling Layers: Max pooling layers follow each convolutional layer to downsample the feature maps, reducing their dimensions and retaining the most important information. A 2x2 pooling size is used.
3. Dropout Layer: A dropout layer is included to prevent overfitting. During training, it randomly sets a fraction of the input units to zero, which helps in making the model more robust.
4. Flatten Layer: This layer flattens the 2D feature maps into a 1D vector, preparing them for the fully connected (dense) layers.
5. Dense Layer: The final dense layer has 10 units corresponding to the 10 classes of digits, with a softmax activation function to output a probability distribution over the classes.

4. Model Compilation

The model is compiled using the Adam optimizer, which is known for its efficiency and effectiveness in training deep learning models. The loss function used is categorical crossentropy, which is suitable for multi-class classification tasks. Accuracy is chosen as the metric to evaluate the model's performance.

5. Model Training

The model is trained on the preprocessed training data. To enhance the training process and avoid overfitting, two callbacks are used:

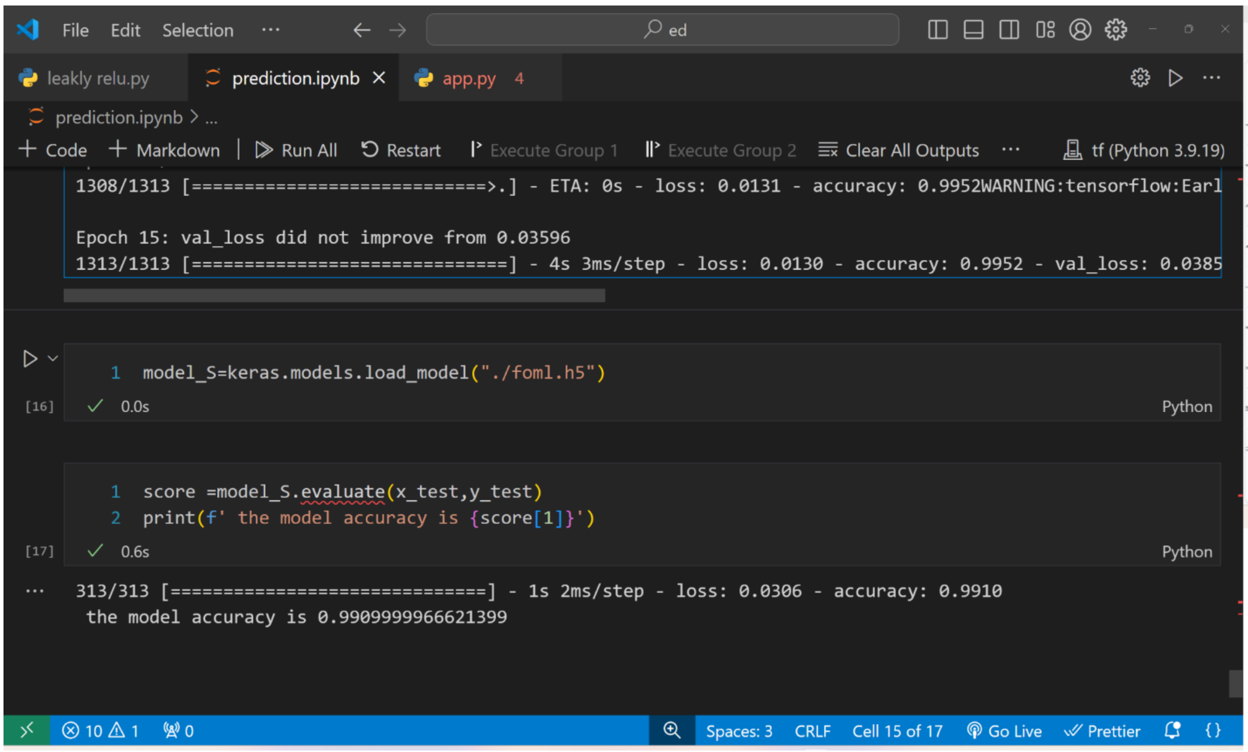
Early Stopping: This callback monitors the validation accuracy and stops the training if the performance does not improve for a specified number of epochs, thus preventing overfitting.

Model Checkpoint: This callback saves the best model based on the validation accuracy during training. It ensures that the model with the highest validation accuracy is saved for future use.

The training process involves multiple epochs, where the model iteratively updates its weights to minimize the loss function.

6. Model Evaluation

After training, the best model saved during the training process is loaded. The model's performance is then evaluated on the test dataset to determine its accuracy. The evaluation results provide insights into how well the model generalizes to unseen data.


File Edit Selection
leaky relu.py
prediction.ipynb
app.py
Code Markdown Run All Restart Execute Group 1 Execute Group 2 Clear All Outputs
tf (Python 3.9.19)
Spaces: 3 CRLF Cell 15 of 17 Go Live Prettier

```
1308/1313 [===========================>.] - ETA: 0s - loss: 0.0131 - accuracy: 0.9952WARNING:tensorflow:Earl

Epoch 15: val_loss did not improve from 0.03596
1313/1313 [==============================] - 4s 3ms/step - loss: 0.0130 - accuracy: 0.9952 - val_loss: 0.0385
```

```
[16]
model_S=keras.models.load_model("./foml.h5")
```

```
[17]
score =model_S.evaluate(x_test,y_test)
print(f' the model accuracy is {score[1]}')
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0306 - accuracy: 0.9910
 the model accuracy is 0.9909999966621399
```

Fig 2: MODEL EVALUATION

7. OUTPUT SCREENSHOTS

The final model achieves a high level of accuracy on the test dataset, demonstrating its effectiveness in classifying handwritten digits. The use of CNNs, combined with proper preprocessing and training techniques, results in a robust model capable of accurate digit classification.

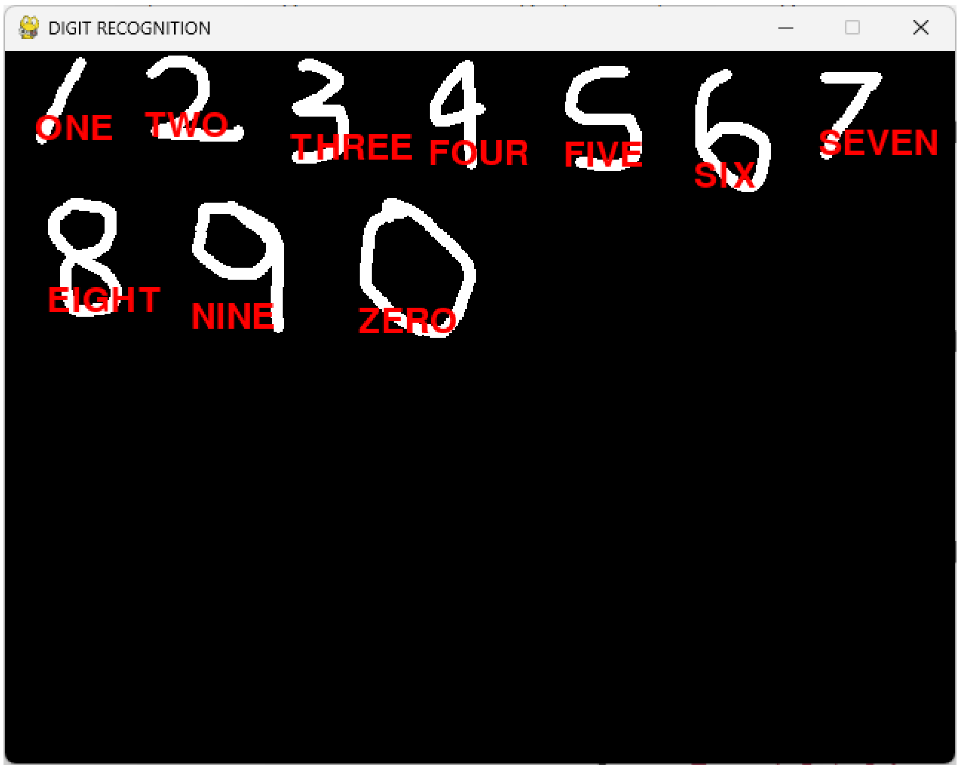


Fig 3:OUTPUT

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

The key goal of this paper is to find a representation that makes for successful identification of isolated hand-written digits. For the identification of hand-written numerals, numerous machine learning algorithms were used in this paper. The important challenge in every identification method is to resolve the extraction of features and valid classification approaches. In terms of precision and time complexity, the suggested algorithm aims to answer all the variables and well. Among all Machine learning models we got the SVM (Support Vector Machine) recognition method, the total highest accuracy of 95.88 percent is reached. This study is carried out as an initial effort, and the purpose of the paper is to make it simpler to identify hand-written digits without using any common methods for classification.

Handwritten digit recognition using the MNIST database is a fundamental yet powerful example of how machine learning can be applied to image recognition tasks. The MNIST database, which consists of 70,000 labeled images of handwritten digits, has become a benchmark dataset for evaluating the performance of various machine learning algorithms. Through the application of different techniques such as neural networks, support vector machines, and convolutional neural networks (CNNs), researchers have been able to achieve high accuracy rates in digit classification. Among these methods, CNNs have proven to be particularly effective due to their ability to automatically and adaptively learn spatial hierarchies of features from the input images, resulting in state-of-the-art performance.

The success of handwritten digit recognition using the MNIST database underscores the broader applicability of machine learning and deep learning techniques in the field of image processing and pattern recognition. The advancements made with this dataset have paved the way for more complex applications, such as facial recognition, object detection, and medical image analysis. Moreover, the ease of access to the MNIST dataset and the plethora of available resources and tutorials have made it an invaluable educational tool for those new to machine learning. Overall, the work on the MNIST database has not only demonstrated the capabilities of modern machine learning models but has also inspired further innovations and applications across various domains.

6.2 FUTURE ENHANCEMENTS

Future research should allow the usage of a convolution Neural Network architecture which is the topic of deep learning, which provided the best result in the MNIST database and implemented the proposed recognition method by hand. Such more machines may be configured to recognize handwritten characters, identify objects, segment items, recognize handwriting, acknowledge text language, and for potential research, but could also allow hardware deployment for more effective and reliable live results on an online software recognition framework for live test case scenarios

REFERENCES

[1] M. Z. Alom, P. Sidike, T. M. Taha, and V. K. Asari, “Handwritten Bangla Digit Recognition Using Deep Learning,” vol. 6, no. 7, pp. 990–997, 2017.

[2] S. Majumder, “Handwritten Digit Recognition by Elastic Matching,” J. Comput., vol. 4, no. 04, pp. 1067–1074, 2018.

[3] G. Cheedella, “Critique of Various Algorithms for Handwritten Digit Recognition Using Azure ML Studio,” Glob. J. Comput. Sci. Technol., vol. 20, no. 1, pp. 1–5, 2020.

[4] S. M. Shamim, M. B. A. Miah, A. Sarker, M. Rana, and A. Al Jobair, “Handwritten digit recognition using machine learning algorithms,” Indones. J. Sci. Technol., vol. 3, no. 1, pp. 29–39, 2018.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, “LeNet,” Proc. IEEE, no. November, pp. 1–46, 1998.

[6] I. Ali, I. Ali, A. Khan, S. A. Raza, B. Hassan, and P. Bhatti, “Sindhi Handwritten-Digits Recognition Using Machine Learning Techniques Sindhi Handwritten-Digits Recognition Using Machine Learning Techniques,” Int. J. Comput. Sci. Netw. Secur., vol. 19, no. 5, pp. 195–202, 2019.

[7] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep, big, simple nets for handwritten digit recognition,” Neural Comput., vol. 22, no. 12, pp. 3207–3220, 2010.

[8] D. N. Kumar and H. Beniwal, “Survey on Handwritten Digit Recognition using Machine Learning,” Int. J. Comput. Sci. Eng., vol. 06, no. 05, pp. 96–100, 2018.

[9] Z. Chen, “Handwritten digits recognition,” Proc. 2009 Int. Conf. Image Process. Comput. Vision, Pattern Recognition, IPCV 2009, vol. 2, pp. 690–694, 2009.

[10] P. K. Singh, R. Sarkar, and M. Nasipuri, “A Study of Moment Based Features on Handwritten Digit Recognition,” Appl. Comput. Intell. Soft Comput., vol. 2016, pp. 1–17, 2016.

[11] F. Lauer, C. Y. Suen, and G. Bloch, “A trainable feature extractor for handwritten digit recognition,” Pattern Recognit., vol. 40, no. 6, pp. 1816–1824, 2007.

[12] F. T. Shah and K. Yousaf, “Handwritten Digit Recognition Using Image Processing and Neural Networks,” Lect. Notes Eng. Comput. Sci., vol. 2165, no. 1, pp. 648–651, 2007.

[13] S. Preetha, I. M. Afrid, K. H. P, and S. K. Nishchay, “Machine Learning for Handwriting Recognition,” vol. 4523, pp. 93–101.

[14] B. J. Van Der Zwaag, “Handwritten digit recognition: A neural network demo,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2206 LNCS, no. October 2001, pp. 762–771, 2001.