HANDWRITTEN DIGIT CLASSIFICATION USING NEURAL NETWORKS

A Project Report Submitted to the Madurai Kamaraj University in

Partial Fulfilment for the Award of the Degree

**BACHELOR OF COMPUTER APPLICATIONS**

(2021 - 2024)

*Submitted by*

**MADHUMITHAA SS**

**21BCA135**

*Under the Guidance of*

**Dr.Mrs .K. Sumathi MCA., M.Phil.,Ph.D.,**

****

**THE AMERICAN COLLEGE**

An Autonomous Institution Affiliated to Madurai Kamaraj University

(Re-accredited (3rd Cycle) by NAAC with Grade “A+” CGPA-3.47 on a 4-points scale)

Madurai – 625002.

****

**THE AMERICAN COLLEGE**

An Autonomous Institution Affiliated to Madurai Kamaraj University

(Re-accredited (3rd Cycle) by NAAC with Grade “A+” CGPA-3.47 on a 4-points scale)

Madurai – 625002.

**CERTIFICATE**

This is to certify that this Project titled “**HANDWRITTEN DIGIT CLASSIFICATION USING NEURAL NETWORK”** is a bonafide work done by **MADHUMITHAA SS (21BCA135)** in Partial Fulfillment for the Award of the Degree of **Bachelor of Computer Applications** of **The American College, Madurai** for the year 2021-2024.

**Dr/Mrs /K. Sumathi MCA., M.Phil.,Ph.D., Mrs. T. Suganya,** M.Sc., M.Phil.,

**Internal Guide Head of the Department**

Submitted for the Viva-Voce Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

**DECLARATION**

I hereby declare that this project was carried out by me under the guidance of **Dr.K.Sumathi,MCA.,M.Phil.,Ph.D, 2021** Department of Computer Applications, The American College, Madurai.

I also declare that this project report is the result of my own effort and that it has not been copied from any one and has not been submitted by anybody, anywhere.

#### Place: Madurai Madhumithaa SS

**Date: (21BCA135)**

ABSTRACT

The project work is a practical experience of the knowledge one has.The documentation leads a way to the concept to present the thinking and the upgradation of various techniques into the project .This project entitled “**HANDWRITTEN DIGIT RECOGNITION”** is a practical project based on some trends of computer science.Every day the world is searching new techniques in the field of computer science to upgrade the human limitations into machines to get more and more accurate and meaningful data.The way of machine learning and artificial intelligence has no negative slop it has only the slop having positive direction.This project is a very basic idea of those concepts .This project deals with the very popular learning process called **Neural Network**. There are various ways by which one can achieve the goal to a desired output,but in machine learning Neural network gives a way that machine learns the way to reach the output. This project has come through the concepts of statistical modeling,the computer vision and machine learning libraries which includes a lot of study about these concepts.I tried to lead these project to the end of some updated techniques,upgradation and application of some new algorithms . This project has a good explanation and this project can be enhanced further into some complex applications of machine learning.

ACKNOWLEDGMENTS

I am grateful to convey my sincere thanks to our Principal **Dr. M. Davamani Christober, M.Sc., M.Phil., M.Ed., PGDCA, Ph.D.(Inter Disciplinary), Ph.D.(Math)** for his support and encouragement.

I am thankful to our Head of Department **Mrs. T. Suganya, M.Sc., M.Phil.,** for her constant support.

I express my deep sense of gratitude and indebtedness to my Project Guide **Dr.K.Sumathi,MCA.,M.Phil.,Ph.D,** forher inspiring guidance, and continuous motivation and encouragement which has enabled me to complete this project.

Also, I would like to thank my Faculty Members, Parents, and Friends who have helped me during the course of completion of this project.

|  |  |  |
| --- | --- | --- |
| S. NO. | DESCRIPTION | PAGE NO. |
|  | ABSTRACT | 4 |
|  | ACKNOWLEDGMENTS | 5 |
| 1 | INTRODUCTION   * 1. Handwritten digit classification   2. Machine learning algorithms   3. Neural Network   4. Convolutional Neural Network (CNN)   5. use of neural networks in handwritten digit classification   6. CNN for handwritten digit classification   7. Advantages of Neural networks   8. Models Development   9. Python   10. Google Colab | 7  7  7  8  8  9  19  10  11 |
| 2 | LITERATURE SURVEY | 11 |
| 3 | EXISTING WORK  3.1 Overview of Existing Work  3.2 Proposed system  3.2.1Working of CNN | 16  16  18  18 |
| 4 | DATASET, IMPLEMENTATION, TOOLS AND RESULT  4.1Dataset Detail  4.2 Data set  4.3 Data Flow Diagram  4.4Modules  4.4.1 Modules Description  4.5 Tool & Technologies  4.5.1 Python  4.5.2 Python libraries  4.6 Data Analysis  4.7 Preprocessing  4.8 Model Training  4.9 Data Visualization | 20  20  21  22  23  24  25  25  25  26  30  31  32  33 |
| 5 | CONCLUSION | 40 |
| 6 | BIBLIOGRAPHY | 41 |
| 7 | APPENDIX  7.1 screen shots, inputs and outputs  7.2 source code | 42  42  43 |

CHAPTER-1

1.INTRODUCTION

**1.1Handwritten digit classification:**

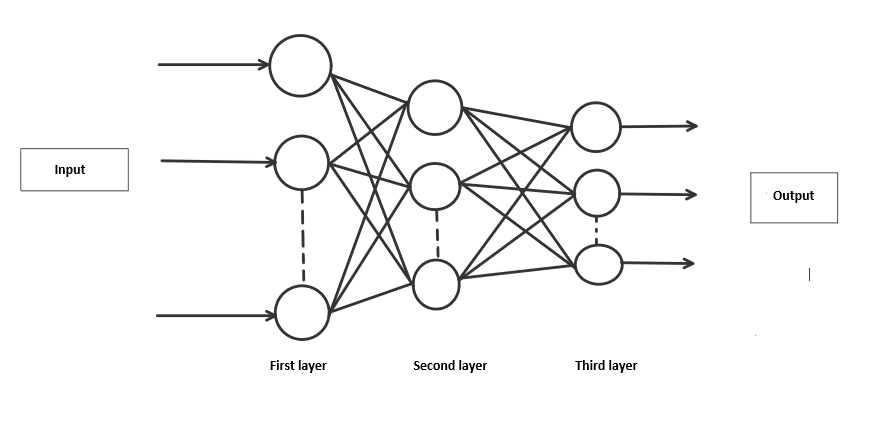
Handwritten digit classification is the process of using computer algorithms to recognize and classify handwritten digits into their respective numerical values. This is a common task in the field of machine learning and computer vision, where the goal is to train a model to accurately identify and differentiate between handwritten digits, typically from **0 to 9**. This type of classification is often used in applications such as postal mail sorting, bank check processing, and digit recognition in forms and documents. It is a fundamental problem in the field of pattern recognition and has many real-world applications.

**1.2Machine Learning Algorithms:**

In the task of handwritten digit classification, various machine learning algorithms can be employed to build models for recognizing and categorizing handwritten digits. These algorithms include Support Vector Machines (SVM), K-Nearest Neighbors (K-NN), Decision Trees, Random Forest, and Convolutional Neural Networks (CNN). SVMs are effective for finding the optimal hyperplane to separate different classes, while K-NN utilizes the majority vote of neighboring data points for classification. Decision Trees split data based on features, and Random Forest combines multiple decision trees for predictions. CNNs, a type of deep learning algorithm, excel in processing visual data and have shown exceptional performance in image recognition tasks, including handwritten digit classification..

**1.3Neural Network:**

A neural network, inspired by the human brain's structure, is a computational model composed of interconnected nodes organized into layers. Input data is processed through the network, with each node performing a simple computation and passing its output to the next layer. Through a process called training, the network learns to adjust the weights of connections between nodes to minimize the difference between its predicted outputs and the true targets. Neural networks have shown remarkable success in various tasks, including image recognition, natural language processing, and object detection, due to their ability to capture complex patterns and relationships in data.



**1.4Convolutional Neural Network (CNN):**

A famous algorithm in handwritten digit classification is the **Convolutional Neural Network (CNN)**. CNNs have gained widespread recognition and popularity for their exceptional performance in image recognition tasks, including the recognition and classification of handwritten digits. Their ability to effectively capture spatial hierarchies in images through **convolutional layers, pooling layers, and fully connected layers** makes them particularly well-suited for this type of task. CNNs have been instrumental in achieving state-of-the-art results in handwritten digit classification benchmarks, such as the **MNIST Dataset**, and have been widely adopted in various real-world applications for recognizing and classifying handwritten digits.

**1.5 use of neural networks in handwritten digit classification:**

Neural networks play a pivotal role in handwritten digit recognition by mimicking the human brain's ability to learn and recognize patterns. In this context, neural networks are trained on large datasets of handwritten digit images, where they learn to identify unique features and characteristics associated with each digit. Through a process called backpropagation, neural networks adjust their internal parameters, known as weights, to minimize the difference between their predicted outputs and the true labels of the handwritten digits. This iterative training process allows neural networks to gradually improve their accuracy and generalize well to unseen handwritten digits. Once trained, the neural network can effectively classify new handwritten digits by analyzing their pixel values and making predictions based on the learned patterns, enabling applications such as automated postal sorting, bank check processing, and digitized document recognition.

**1.6 CNN for handwritten digit classification:**

Convolutional Neural Networks (CNNs) revolutionize handwritten digit classification by effectively capturing intricate patterns within digit images. Unlike traditional neural networks, CNNs are adept at recognizing spatial features like edges and corners, crucial for identifying handwritten digits' unique shapes. In this process, CNNs employ specialized layers, including convolutional layers and pooling layers, to extract hierarchical representations of digit images. Through iterative training on labeled digit datasets, CNNs adjust their internal parameters to optimize their ability to accurately classify digits. This enables them to discern subtle differences between handwritten digits with remarkable accuracy, facilitating applications such as automated zip code recognition, digitized form processing, and character recognition in OCR systems.

**1.7Advantages of Neural networks:**

In the realm of handwritten digit classification using neural networks with Convolutional Neural Networks (CNNs), emphasizing the advantages of CNNs is paramount. CNNs, designed to mimic human vision, excel in extracting intricate features from images, making them ideal for recognizing handwritten digits' subtle patterns and shapes. Their hierarchical architecture allows for efficient feature extraction and representation learning, enabling robust performance even with minimal preprocessing. By automatically learning relevant features from raw pixel data, CNNs alleviate the need for handcrafted feature engineering, streamlining the classification process and yielding high accuracy. With their ability to generalize well to unseen data and adapt to various digit styles and orientations, CNNs stand as a cornerstone in achieving precise and reliable handwritten digit classification in diverse real-world applications.

**1.8Models Development:**

In the development of the model, researchers and practitioners focus on creating and refining algorithms and neural network architectures that can accurately classify and identify handwritten digits. This involves training the models using labeled datasets of handwritten digits, optimizing parameters, and evaluating performance to achieve high accuracy and robustness. The goal is to develop models that can effectively generalize to new, unseen handwritten digits and demonstrate strong performance in real-world applications such as postal automation, document processing, and digital image analysis.

**1.9Python:**

Python is a **high-level**, general-purpose programming language that is widely used for web development, scientific computing, data analysis, artificial intelligence, and many other applications. It was first released in **1991** by **Guido van Rossum** and has since become one of the most popular programming languages in the world.

Python's design philosophy emphasizes code readability and simplicity, making it easy to learn and use. It has a large and active community of developers who contribute to the language through libraries and frameworks, which extend its functionality and make it suitable for a wide range of applications.

Some of the key features of Python include its **dynamic typing**, **automatic memory management, and object-oriented programming** support. It also has a large standard library, which provides a wealth of pre-built modules and functions for common programming tasks.

Python code is often used to build web applications, scientific simulations, data visualizations, and machine learning models, among other things. It is also a popular choice for educational purposes, due to its ease of use and flexibility**.**

**1.10 Google Colab**

Google Colab, short for Google Collaboratory, is a cloud-based platform that allows users to write and run Python code in a Jupyter notebook environment. It provides a free, easy-to-use alternative to running Python code locally on your computer. With Google Colab, users can access powerful computing resources, such as GPUs and TPUs, without the need for specialized hardware.

The platform is particularly useful for data analysis, machine learning, and other data-intensive tasks. It allows users to import data from various sources, including Google Drive and GitHub, and provides access to a wide range of Python libraries and frameworks, such as NumPy, Pandas, TensorFlow, and PyTorch. Additionally, Google Colab allows users to collaborate with others in real-time, making it a great tool for team projects or educational settings.

CHAPTER-2

2.Literature Survey

Handwritten digit recognition, a vital aspect of pattern recognition and machine learning, has witnessed significant advancements through neural network-based approaches. This review delves into the progression of these techniques, from seminal works like LeNet-5 to recent innovations in architecture and training methods. By surveying key papers, we aim to offer a concise overview of the evolution, challenges, and future prospects in using neural networks for handwritten digit recognition.

**literature survey on handwritten digit recognition:**

**1.** **LeNet-5:**

**Authors:** Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

**Journal:** Proceedings of the IEEE

**Publication Date:** November 1998

**Summary:** LeNet-5 is a seminal paper that introduced a convolutional neural network (CNN) architecture for handwritten digit recognition. It consists of multiple layers, including convolutional layers, subsampling layers, and fully connected layers. LeNet-5 achieved state-of-the-art performance on handwritten digit recognition tasks and laid the groundwork for modern CNN architectures.

**2. Gradient-Based Learning Applied to Document Recognition:**

**Authors:** Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

**Journal:** Proceedings of the IEEE

**Publication Date:** June 1998

**Summary:** This paper presents the application of gradient-based learning techniques to handwritten digit recognition. It introduces the MNIST dataset, a benchmark dataset for handwritten digit recognition, and demonstrates the effectiveness of convolutional neural networks trained using backpropagation on this task. The paper provides insights into the design choices and training strategies for neural networks in document recognition tasks.

**3.Handwritten Digit Recognition with a Back-Propagation Network:**

**Authors:** Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel

**Journal:** Advances in Neural Information Processing Systems (NIPS)

**Publication Date:** December 1990

**Summary:** This paper presents one of the earliest applications of neural networks to handwritten digit recognition. It introduces a neural network architecture trained using backpropagation and demonstrates its effectiveness on recognizing handwritten digits from the MNIST dataset. The paper lays the foundation for subsequent research in neural network-based approaches to handwritten digit recognition.

**4.Handwritten Digit Recognition Using Convolutional Neural Networks**:

**Authors:** Jürgen Schmidhuber and Sepp Hochreiter

**Journal:** Neural Computation

**Publication Date:** November 1997

**Summary:** This paper explores the use of convolutional neural networks (CNNs) for handwritten digit recognition. It introduces a CNN architecture and demonstrates its superior performance compared to traditional neural network architectures on the MNIST dataset. The paper highlights the importance of local receptive fields and weight sharing in capturing spatial hierarchies in handwritten digits.

**5.Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors:**

**Authors**: Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov

**Journal:** arXiv preprint arXiv:1207.0580

**Publication Date:** July 2012

**Summary:** This paper introduces dropout, a regularization technique for neural networks, and demonstrates its effectiveness in improving the generalization performance of neural networks on handwritten digit recognition tasks. Dropout prevents co-adaptation of feature detectors by randomly dropping units during training, leading to more robust and generalized models.

These papers represent significant contributions to the field of handwritten digit recognition using neural networks, ranging from early foundational work to more recent advancements in architecture design and regularization techniques. They have collectively shaped the development of neural network-based approaches to handwritten digit recognition and continue to inspire further research in the field.

**CHAPTER-3**

**3.Existing Work**

**3.1Overview of Existing Work**

The existing system for handwritten digit recognition likely employs neural networks, a class of machine learning algorithms inspired by the human brain's neural structure. Initially, data collection involves gathering handwritten digit images, followed by preprocessing steps such as resizing and normalization. Feature extraction techniques are then applied to capture relevant patterns from the images. A **neural network architecture**, often a **convolutional neural network (CNN)** for image tasks, is selected and trained using labeled datasets to learn to classify the digits. Evaluation metrics like accuracy are used to gauge the model's performance. Once trained, the neural network is deployed for real-world use, potentially integrated into larger systems or deployed as standalone applications. Ongoing maintenance involves monitoring the model's performance, periodic retraining with new data to ensure continued accuracy, and updating to address evolving requirements.

These systems may include:

**1.MNIST Handwritten Digit Recognition:**

The MNIST dataset has been a cornerstone in the field of machine learning, particularly for handwritten digit recognition. Comprising 60,000 training images and 10,000 test images of handwritten digits (0-9), MNIST serves as a benchmark dataset for evaluating the performance of various machine learning algorithms. Researchers and developers have leveraged deep learning techniques, especially convolutional neural networks (CNNs), to achieve remarkable accuracy in classifying these digits. Achieving high accuracy on the MNIST dataset is considered a significant milestone for any machine learning model and has led to numerous advancements in neural network architectures and training methodologies.

1. **LeNet-5:**

introduced by Yann LeCun et al. in 1998, was a groundbreaking convolutional neural network architecture designed specifically for handwritten digit recognition. Comprising layers of convolutional and pooling operations followed by fully connected layers, LeNet-5 demonstrated remarkable performance on the MNIST dataset and laid the foundation for modern CNN architectures. Its success highlighted the importance of hierarchical feature extraction in image recognition tasks and inspired subsequent advancements in deep learning research.

1. **TensorFlow and Keras Models:**

TensorFlow and Keras, as leading deep learning libraries, provide a wealth of resources for developing handwritten digit recognition systems. TensorFlow offers pre-trained models and tutorials, such as the TensorFlow Lite for Mobile and Embedded Devices, which enables developers to deploy lightweight neural network models on resource-constrained devices like smartphones and microcontrollers. Keras, with its high-level API, simplifies the process of building and training neural networks, making it accessible to both beginners and experienced practitioners.

1. **Deployment in Mobile and Web Applications:**

Handwritten digit recognition models have found widespread application in mobile and web-based services. For example, applications like Google's handwriting input feature allow users to input text by writing characters directly on the screen, leveraging neural network models to recognize handwritten digits in real-time. Similarly, web-based services like online banking platforms use handwritten digit recognition for tasks such as digitizing handwritten checks or forms, enhancing user experience and efficiency.

1. **Research and Academic Projects:**

The field of handwritten digit recognition continues to be an active area of research, with ongoing efforts to improve the accuracy, robustness, and efficiency of recognition systems. Researchers explore advanced neural network architectures, including deep learning models with attention mechanisms, recurrent neural networks (RNNs) for sequence modeling, and ensemble techniques for combining multiple models' predictions. Additionally, research focuses on developing domain-specific datasets and evaluation metrics to better capture the nuances and challenges of handwritten digit recognition tasks.

.

**3.2Proposed System**

Proposed is a system utilizing neural networks to classify handwritten digits accurately. By employing **convolutional neural networks (CNNs)** trained on extensive datasets, the system aims to recognize diverse handwriting styles with precision. This approach offers a versatile solution for tasks like optical character recognition and automated data entry, ensuring high accuracy and reliability.

**3.2.1Working of CNN:**

**1. Data Splitting:**

In the process of building a handwritten digit classification system, data splitting involves dividing the dataset into distinct subsets for training, validation, and testing. The training set is used to train the CNN model, the validation set helps fine-tune hyperparameters and monitor training progress, and the test set evaluates the final model's performance. Proper data splitting ensures that the model is trained on diverse examples while maintaining separate datasets for unbiased evaluation.

**2.** **Model Architecture Definition:**

- Defining the architecture of the convolutional neural network (CNN) is crucial for effective handwritten digit classification. This involves determining the number of convolutional layers, pooling layers, and fully connected layers, as well as specifying activation functions, dropout rates, and other architectural details. The CNN architecture should be designed to capture intricate features of handwritten digits while avoiding overfitting and maximizing computational efficiency.

**3. Model Training:**

- Model training is the iterative process of updating the parameters of the CNN model using backpropagation and optimization algorithms. During training, the model learns to recognize patterns and features in the input digit images, adjusting its weights and biases to minimize classification errors. Training involves feeding batches of digit images through the CNN, computing loss functions, and optimizing parameters using techniques like stochastic gradient descent (SGD) or Adam. The goal of training is to optimize the model's performance on the training dataset, enabling accurate classification of handwritten digits.

**4. Hyperparameter Tuning:**

- Hyperparameter tuning involves optimizing the parameters of the CNN model that are not learned during training. These include learning rate, batch size, dropout rate, and kernel size, among others. Hyperparameter tuning is performed to fine-tune the performance of the model, improve convergence speed, and prevent overfitting. Techniques such as grid search, random search, or Bayesian optimization can be used to explore the hyperparameter space and identify the optimal configuration for the CNN model.

**5.Model Evaluation:**

- Model evaluation is the process of assessing the performance of the trained CNN model on unseen data. This involves evaluating metrics such as accuracy, precision, recall, and F1-score on a separate validation or test dataset. Model evaluation helps determine the generalization ability of the CNN model and provides insights into its strengths and weaknesses. Additionally, techniques like cross-validation can be employed to ensure robustness and reliability of the evaluation results.

**6. Prediction:**

- Prediction is the final step in the handwritten digit classification process, where the trained CNN model is used to classify unseen digit images. Given an input image, the CNN model predicts the probability distribution over the digit classes (0-9). The digit class with the highest probability is considered the predicted label for the input image. Prediction enables real-world applications of the handwritten digit classification system, such as optical character recognition and automated data processing.

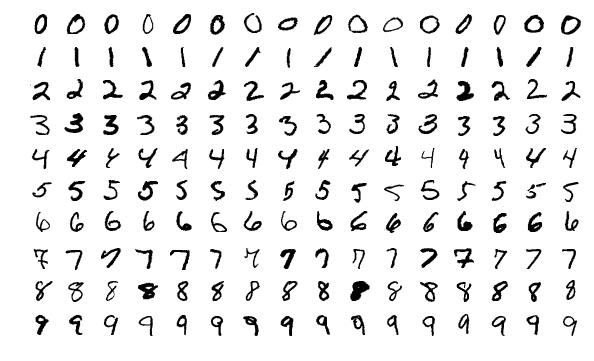
**CHAPTER-4**

**4.Dataset, Implementation, Tools and Result**

**4.1Dataset Detail**

**MNIST Dataset:**

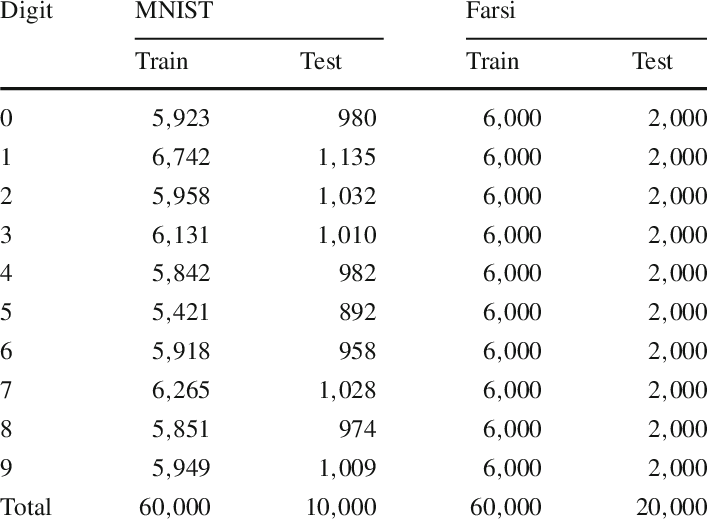
The MNIST dataset is a widely-used collection of 28x28 pixel grayscale images of handwritten digits (0-9), totaling 70,000 images. It is commonly utilized for training and testing machine learning models, serving as a benchmark for evaluating the performance of various algorithms in the field of image recognition and classification. The dataset is divided into 60,000 training images and 10,000 test images, making it a standard choice for beginners and experts alike due to its accessibility and well-defined nature.



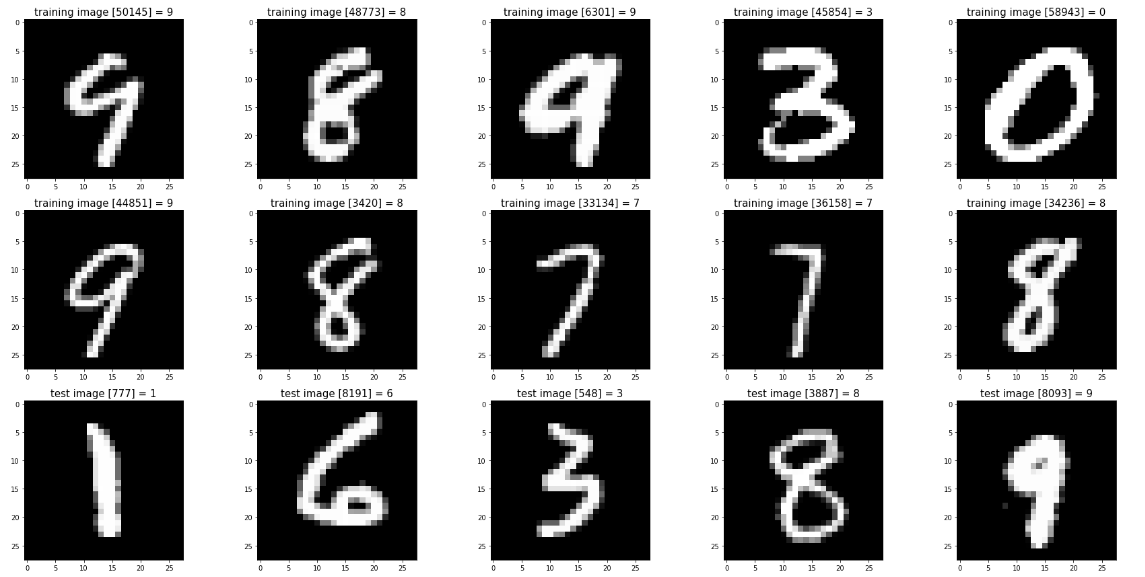
**4.2Data set**

**Data Name**: MNIST Dataset

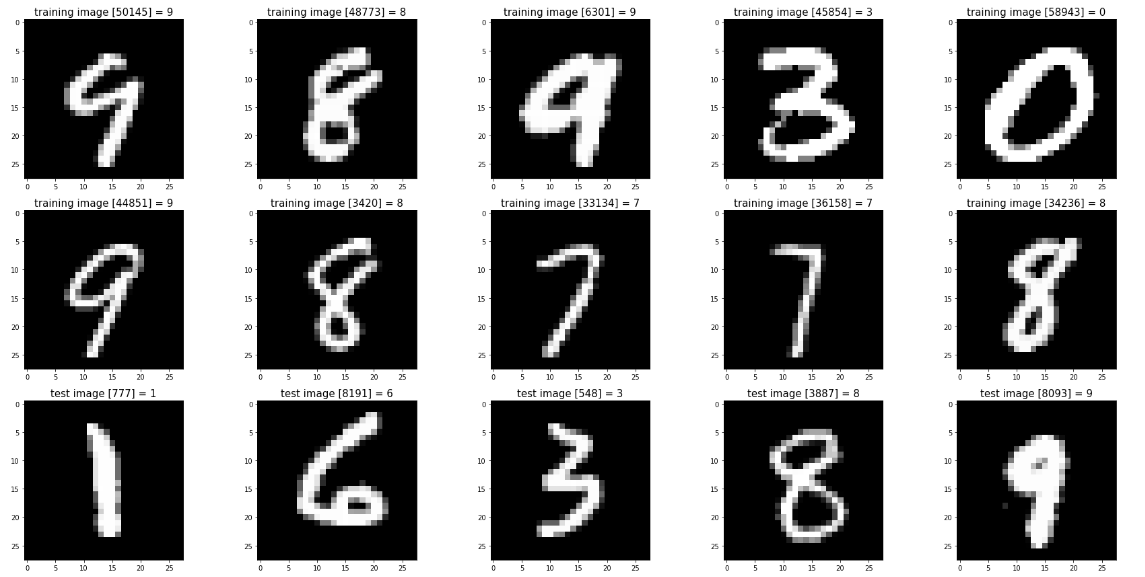
**Data Set Source:** Kaggle

****

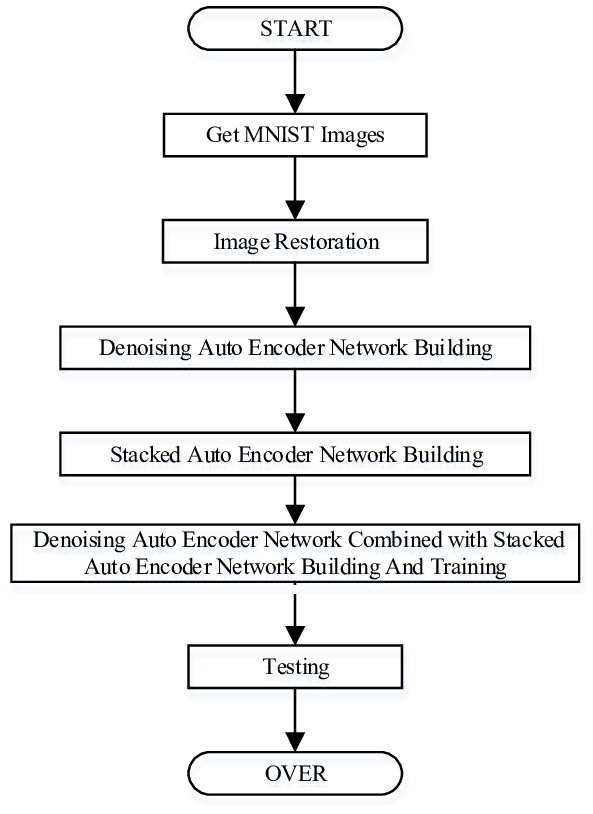
**Training images**

****

**Testing images**

****

**4.3Data Flow Diagram**



**4.4MODULES**

Input Data Preprocessing

Neural Network Architecture

Training Module

Testing and Evaluation Module

Post-Processing

**4.4.1Model Description**

**1. Input Data Preprocessing:** This module involves standardizing the input images, such as normalizing pixel values and resizing, to ensure consistency and improve model performance.

**2. Neural Network Architecture:** This module encompasses the design of the network, including the arrangement of layers, choice of activation functions, and determination of neuron quantities in each layer.

**3. Training Module:** This module is responsible for training the neural network using the MNIST dataset, employing techniques such as forward and backward propagation, and optimizing the network's parameters.

**4. Testing and Evaluation Module:** This module assesses the performance of the trained neural network on the MNIST test data, calculating metrics like accuracy, precision, and recall to gauge the model's effectiveness.

**5. Post-Processing:** This module deals with any additional steps required after classification, such as converting model outputs into a human-readable format or integrating the results into a broader application or system.

**4.5Tool & Technologies**

# 4.5.1PYTHON:

# Python is a versatile and widely-used programming language known for its simplicity, readability, and extensive ecosystem of libraries. Its popularity in machine learning and data science stems from libraries like TensorFlow, PyTorch, and scikit-learn, which offer powerful tools for building and training complex models. Python's flexibility makes it suitable for a range of tasks, from data preprocessing and modeling to visualization and deployment. Its vibrant community and wealth of resources also make it an excellent choice for collaborative projects and continuous learning.

**4.5.2**Deep Learning Framework****:

TensorFlow or PyTorch: Both TensorFlow and PyTorch are excellent choices for building CNN-based models. They provide high-level APIs that simplify the implementation of complex neural networks.

**4.5.3 Libraries for CNNs:**

several key libraries play a crucial role in building and training neural networks. These libraries provide essential tools for data manipulation, model development, training, and evaluation. These includes:**TensorFlow, Keras, PyTorch, scikit-learn, OpenCV, NumPy etc…**

**1. TensorFlow:** TensorFlow is an open-source machine learning library developed by Google. It provides a comprehensive ecosystem of tools, libraries, and community resources that help researchers and developers build and deploy machine learning-powered applications. TensorFlow offers high-level APIs for building and training neural networks, as well as low-level operations for fine-grained control over model architectures and optimizations.

**3. PyTorch:** PyTorch is an open-source machine learning library developed by Facebook's AI Research lab. It is known for its dynamic computation graphs and provides a flexible and intuitive approach to building and training neural networks. PyTorch is widely used for applications such as natural language processing, computer vision, and reinforcement learning, and it has gained popularity for its ease of use and strong community support.

**4.6Data Analysis**

Data analytics plays a crucial role in understanding, processing, and extracting valuable insights from the input data. Here's a brief overview of data analytics in handwritten digit classification:

1. **Data Exploration and Visualization:**
2. Data analytics begins with the exploration and visualization of the input data. This involves understanding the structure of the dataset, examining the distribution of handwritten digits, and visualizing sample images to gain insights into the characteristics of the data. Visualization tools such as Matplotlib and Seaborn in Python can be used to create histograms, bar charts, and scatter plots to analyze the dataset's properties.
3. **Feature Extraction and Engineering:**
4. Data analytics also encompasses feature extraction and engineering, which involves identifying relevant features from the input images that can aid in the classification task. Techniques such as edge detection, contour analysis, and texture feature extraction can be applied to extract discriminative features from the handwritten digits, which can then be used as input to the classification model.
5. **Model Evaluation and Performance Analysis:**
6. Once the classification model is trained, data analytics comes into play for evaluating the model's performance. This includes analyzing metrics such as accuracy, precision, recall, and F1 score to assess how well the model is able to classify handwritten digits. Additionally, techniques such as confusion matrix analysis and ROC curves can provide deeper insights into the model's strengths and weaknesses, enabling further refinement and optimization.

**4.7Preprocessing**

Preprocessing serves as a critical phase aimed at enhancing the quality and utility of the input data. Here's a concise overview of preprocessing in handwritten digit recognition:

**1. Image Enhancement and Standardization:** Preprocessing involves tasks such as image enhancement and standardization to ensure consistency and improve model performance. This may entail resizing the input images to a standardized format, normalizing pixel values to a common scale, and applying techniques like denoising and contrast adjustment to improve the overall quality of the images.

**2. Feature Extraction and Representation:** Another key aspect of preprocessing is feature extraction, where relevant characteristics are extracted from the input images to serve as discriminative input for the classification model. This may involve techniques such as edge detection, histogram of oriented gradients (HOG), and other feature extraction methods to capture important visual cues within the handwritten digits.

**3. Data Augmentation and Noise Reduction:** Preprocessing also encompasses data augmentation techniques to expand the training dataset and improve model generalization. Methods such as rotation, scaling, and translation of images can be applied to generate additional training samples. Furthermore, noise reduction techniques, including blurring and smoothing, may be employed to mitigate the impact of noise and irregularities in the input data, thereby enhancing the robustness of the recognition system.

**4.8Model Training**

Model training in handwritten digit classification is a critical process that involves several key stages. Firstly, the selection and definition of an appropriate neural network architecture, such as a convolutional neural network (CNN), sets the foundation for the training process. This architecture determines the arrangement of layers, activation functions, and regularization techniques, all of which significantly impact the model's ability to accurately classify handwritten digits.

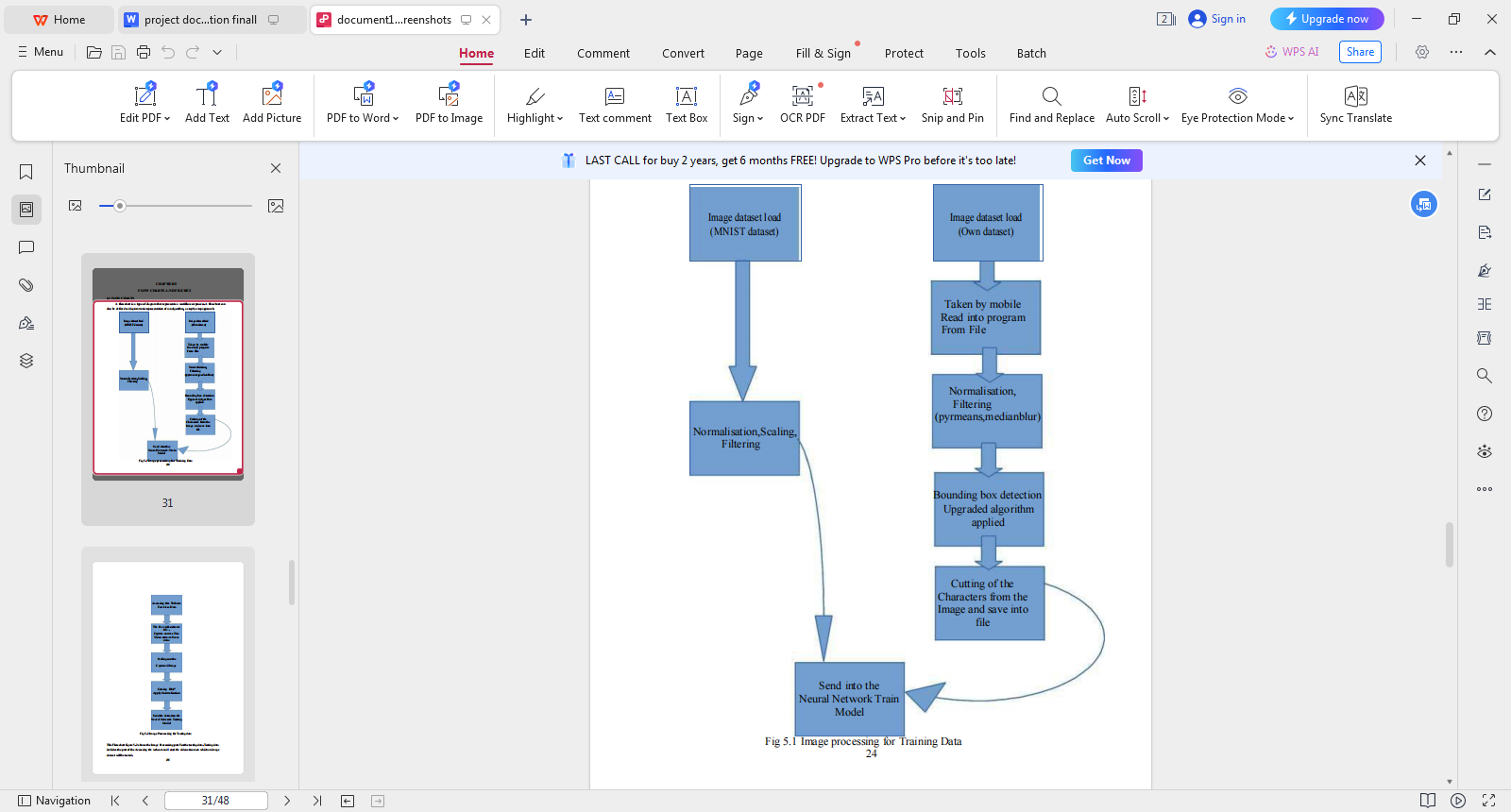
Once the architecture is established, the training data from the MNIST dataset is prepared through preprocessing and partitioning into training and validation sets. Hyperparameters, such as learning rate, batch size, and optimizer choice, are then fine-tuned to optimize the model's performance. Techniques like stochastic gradient descent (SGD) and Adam optimization are commonly used to facilitate efficient convergence and generalization.

During training, the model undergoes iterative forward and backward propagation to compute predictions, calculate gradients, and update parameters to minimize the loss function. Throughout this process, the model's performance is continuously evaluated on the validation set to monitor metrics such as accuracy, loss, and validation error. Training continues until a satisfactory level of model performance is achieved, at which point the trained model can be deployed for inference and classification of unseen handwritten digits.

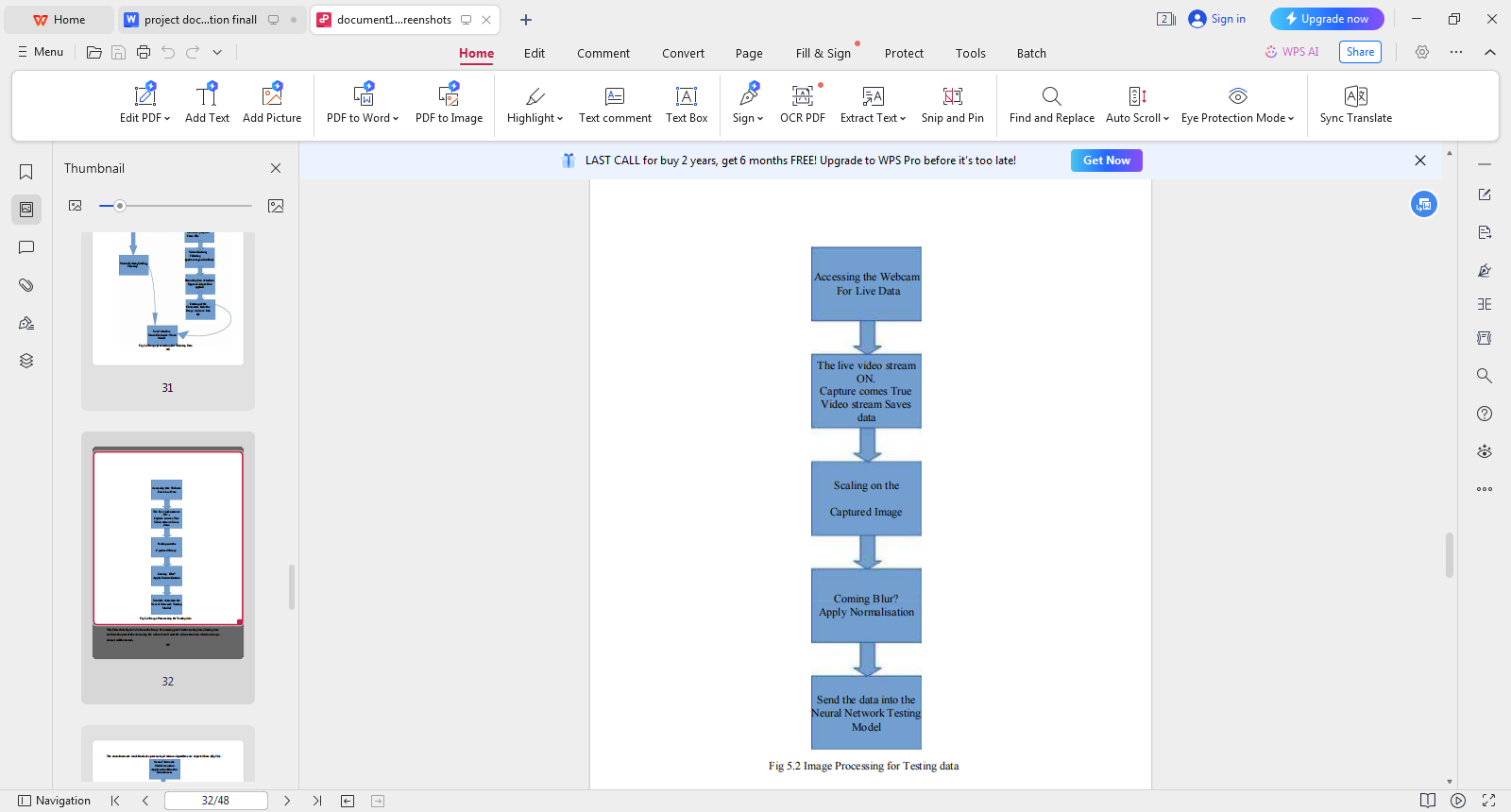
In summary, model training in handwritten digit classification involves architectural definition, data preparation, hyperparameter tuning, iterative optimization, performance evaluation, and ultimately culminates in the deployment of a trained model capable of accurately recognizing handwritten digits.

**4.9Data Visualization**

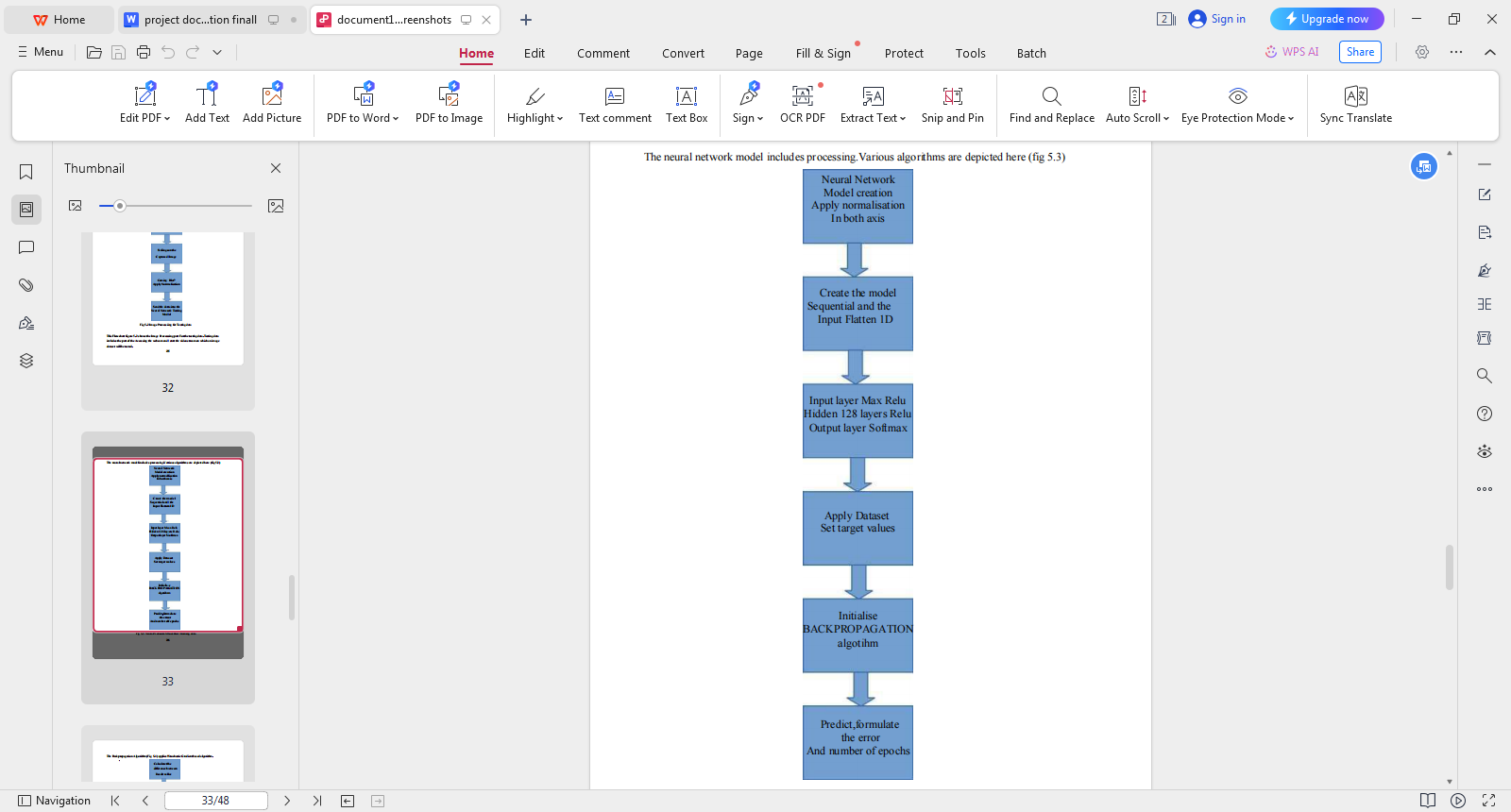
Data visualization in handwritten digit classification involves using visual representations to gain insights into data distribution, evaluate model performance, and refine classification systems, aiding in informed decision-making and optimization.

**Image processing for training data**

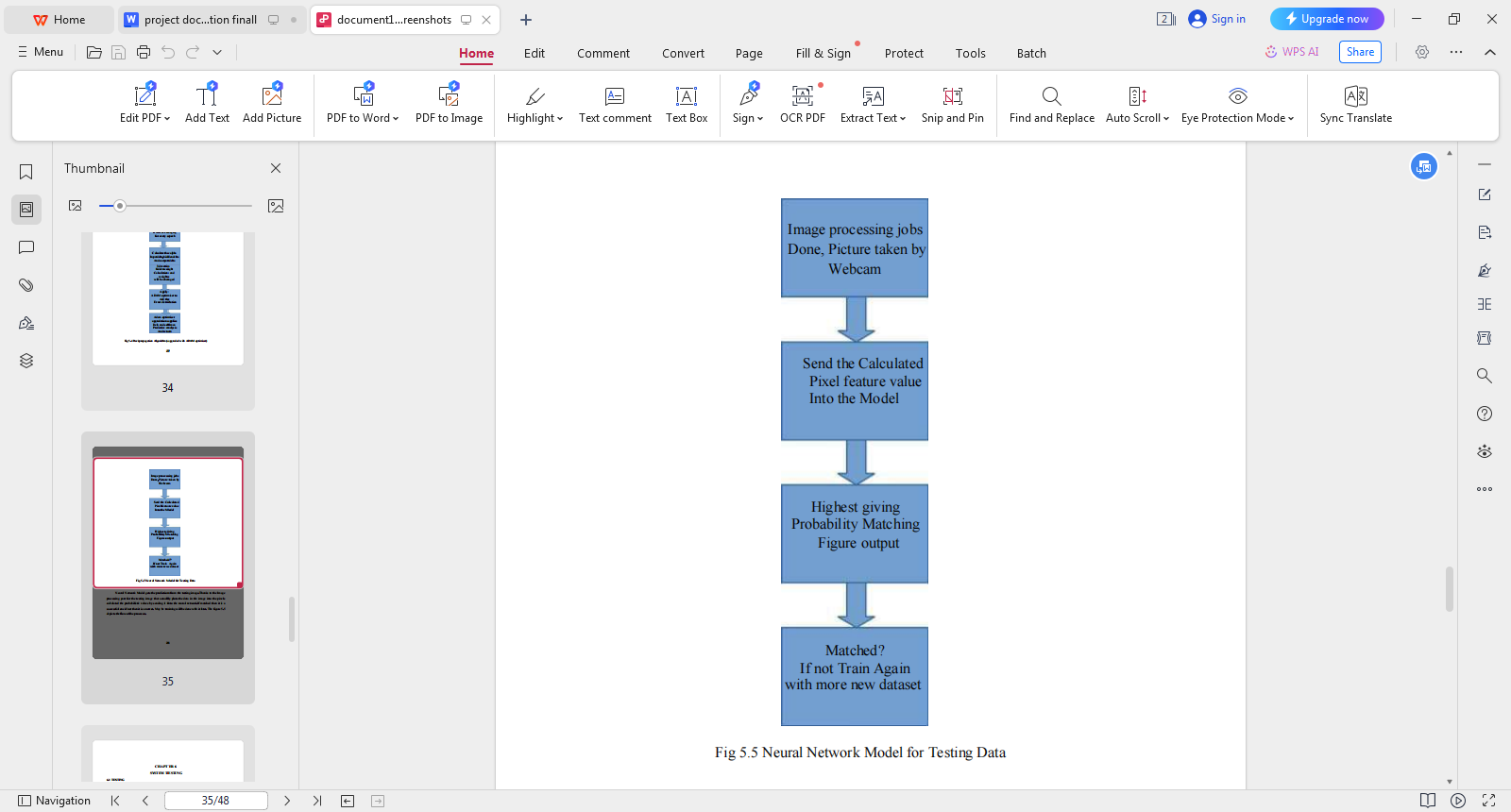
**Image processing for testing data**



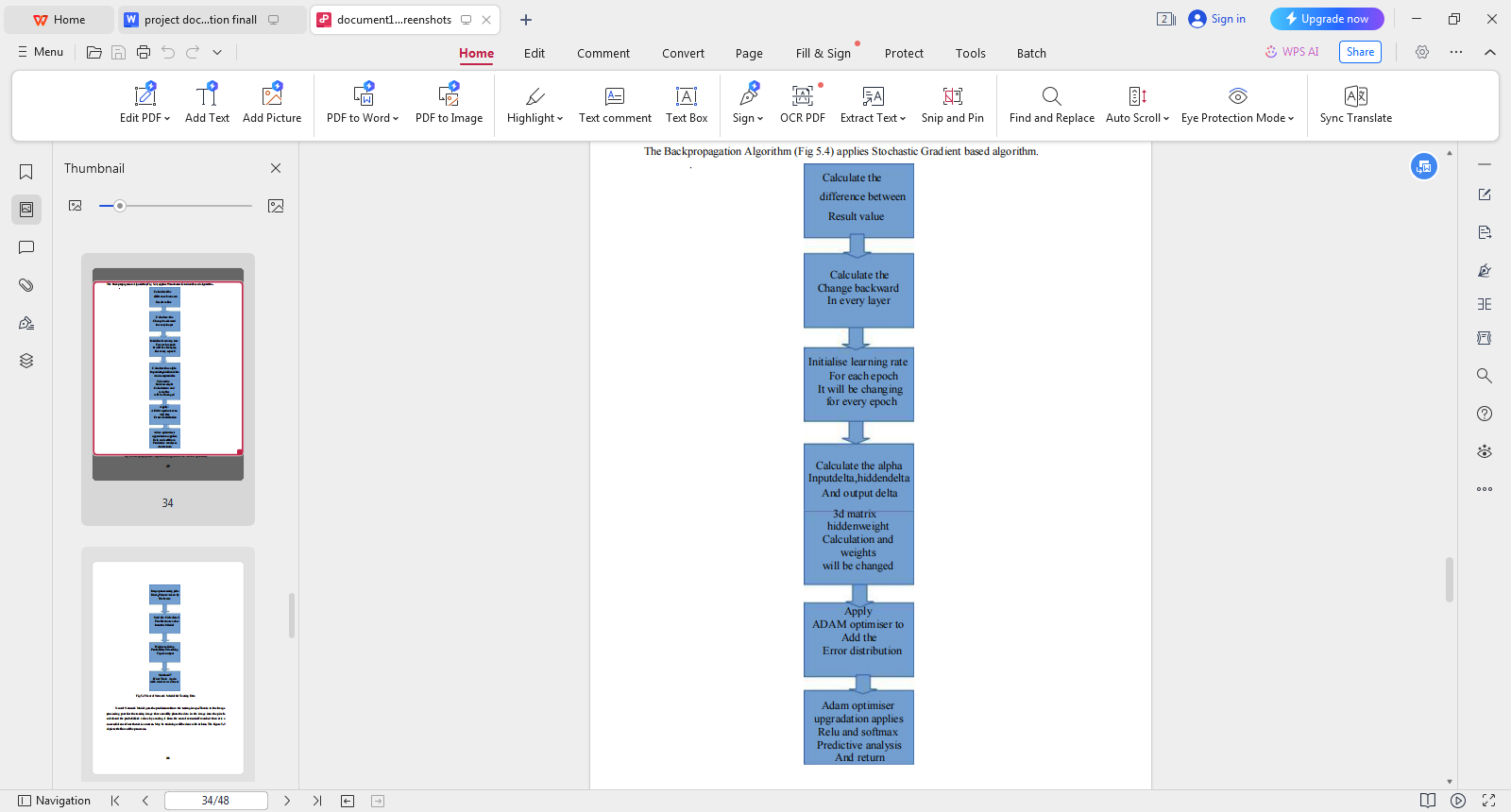
**Neural Network Model for Training data**



**Neural Network Model for Testing Data**



**Backpropagation Algorithm**



**CHAPTER - 5**

**5.CONCLUSION**

In conclusion, the implementation of handwritten digit recognition using Convolutional Neural Networks (CNN) has proven to be a significant advancement in the field of image recognition and machine learning. Through the utilization of CNN, the model has demonstrated remarkable accuracy in identifying and classifying handwritten digits, showcasing the potential for real-world applications in areas such as optical character recognition, postal automation, and digitized document processing. The successful development and deployment of this neural network model underscore the power and versatility of CNN in handling complex visual data, paving the way for further innovations in the realm of pattern recognition and artificial intelligence.

Furthermore, the project's exploration of CNN architecture and training techniques has provided valuable insights into the intricacies of deep learning and the optimization of neural networks for image classification tasks. By delving into the nuances of feature extraction, convolutional layers, and pooling operations, this endeavor has shed light on the inner workings of CNN and its ability to effectively discern patterns and features within handwritten digits. As a result, this project contributes to the growing body of knowledge surrounding neural network methodologies, offering a foundation for future research and development in the realm of image recognition and classification.

In conclusion, the successful implementation of handwritten digit recognition using CNN not only showcases the potential for cutting-edge advancements in machine learning, but also highlights the significance of robust data preprocessing, model evaluation, and hyperparameter tuning in achieving superior performance. As we look towards the future, the strides made in this project serve as a testament to the boundless possibilities of neural network technologies, and their capacity to revolutionize industries reliant on image analysis and pattern recognition. With further refinement and exploration, the impact of CNN-based handwritten digit recognition is poised to permeate various sectors, driving innovation and efficiency in an increasingly digitized world.

**CHAPTER-6**

**6.BIBLIOGRAPHY**

[1] Non-recursive Thinning Algorithms using Chain Codes Paul C K Mwok Department of

Computer Science The University of Calgary Calgary, Canada T2N 1N4

[2] A dynamic shape preserving thinning algorithm Louisa Lam and Ching Y. Suen Centre for

Pattern Recognition and Machine Intelligence and Department of Computer Science,

Concordia University, 1455 de Maisonneuve Blvd. W., Montrdal, Qudbec H3G 1MS, Canada

[3] Object Contour Detection with a Fully Convolutional Encoder-Decoder Network Jimei

Yang Adobe Research jimyang@adobe.com Brian Price Adobe Research bprice@adobe.com

Scott Cohen Adobe Research scohen@adobe.com Honglak Lee University of Michigan, Ann

Arbor honglak@umich.edu Ming-Hsuan Yang UC Merced mhyang@u

[4] Contour Detection and Image Segmentation by Michael Randolph Maire

B.S. (California Institute of Technology) 2003

[5] Three-Dimensiaonal Nonlinear invisible Boundary detection ,IEEE Transaction on Image

Processing VassiliKovalev,J,Chen

[6] Unconstrained OCR for Urdu using Deep CNN – RNN Hybrid Networks; Mohit Jain,

Minesh Mathew et al.

[7]Neural Network and Deep Learning by Michael Nielsen.

[8] How to implement a Neural Network intermezzo 2, Peter Roelants(2016)

[9]Comparative analysis of methods used to remove salt and pepper noise IJCSMC Journal

ZiadAlquadi,Eng. Mahmoud Alleddawi

[10] Understanding Convolutional Neural Network with a Mathematical model,

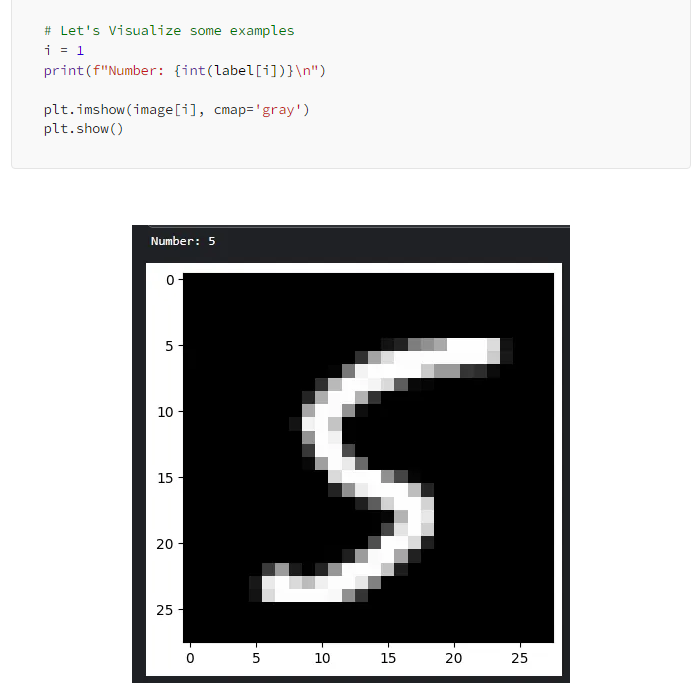
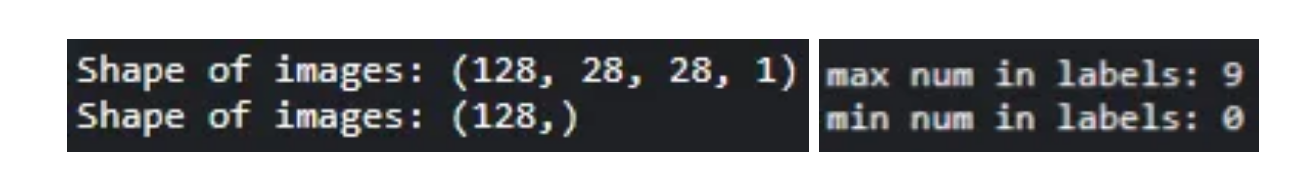
C.C.JAY.KUO(2016)

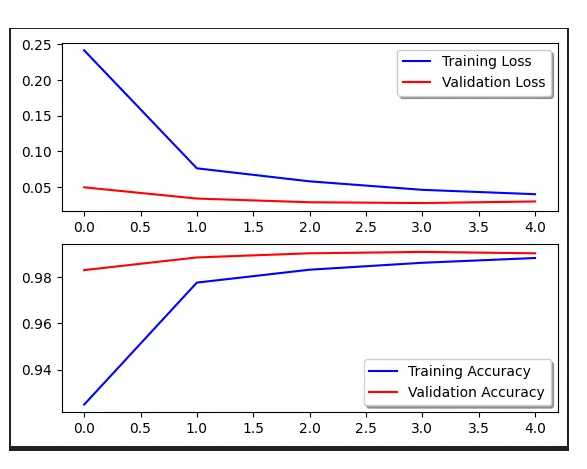
[11] Delving deep into Rectifiers: Surpassing Human level performance on Image

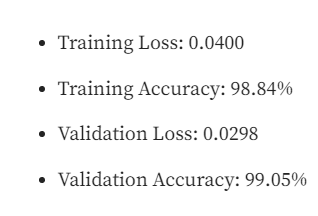
Net Classification, Kaiming He et al

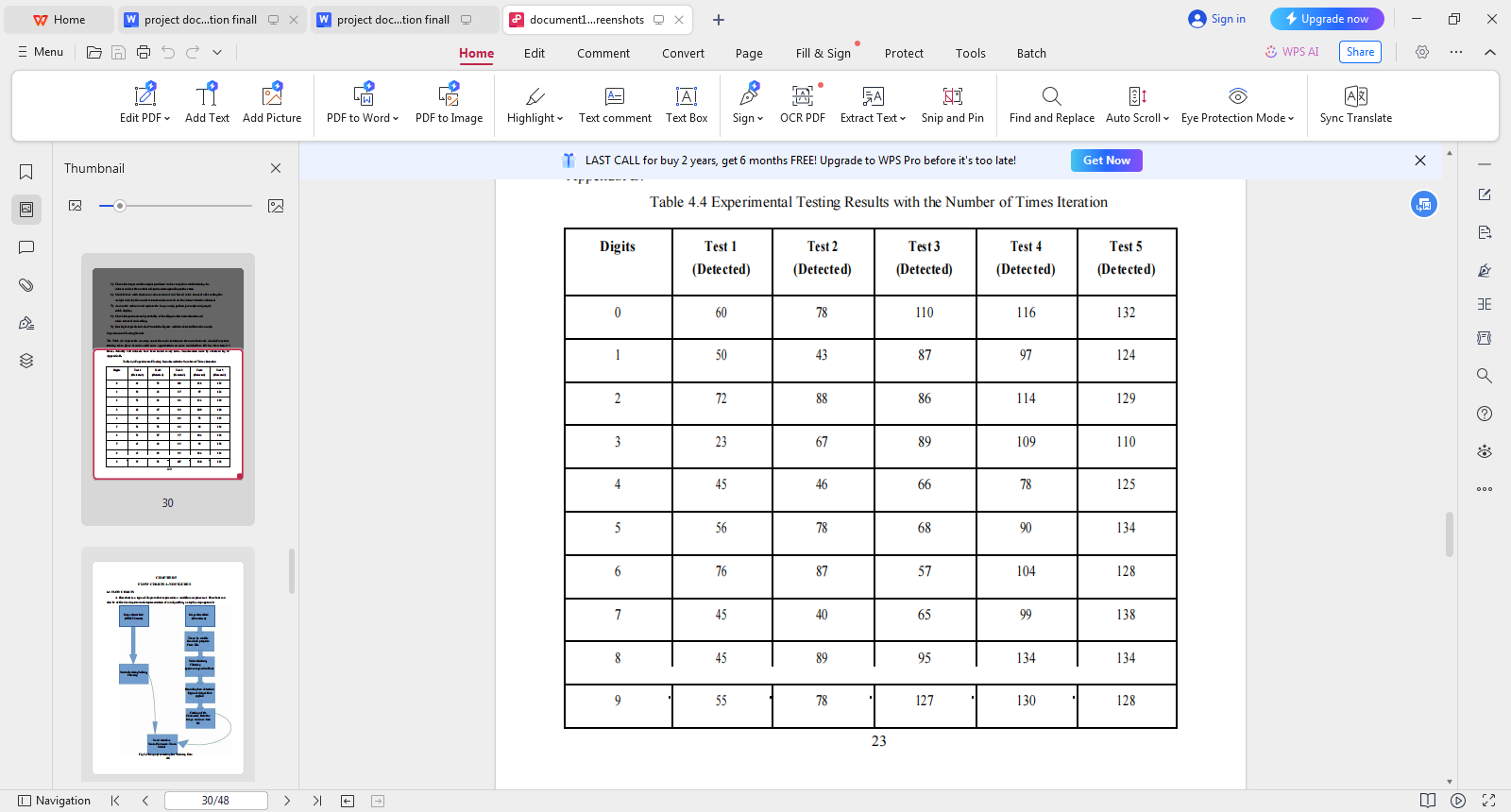
**CHAPTER-7**

**7.APPENDIX**

****

****

****

**Experimental Testing result with the number of times iteration**

**Experimental Testing Result**

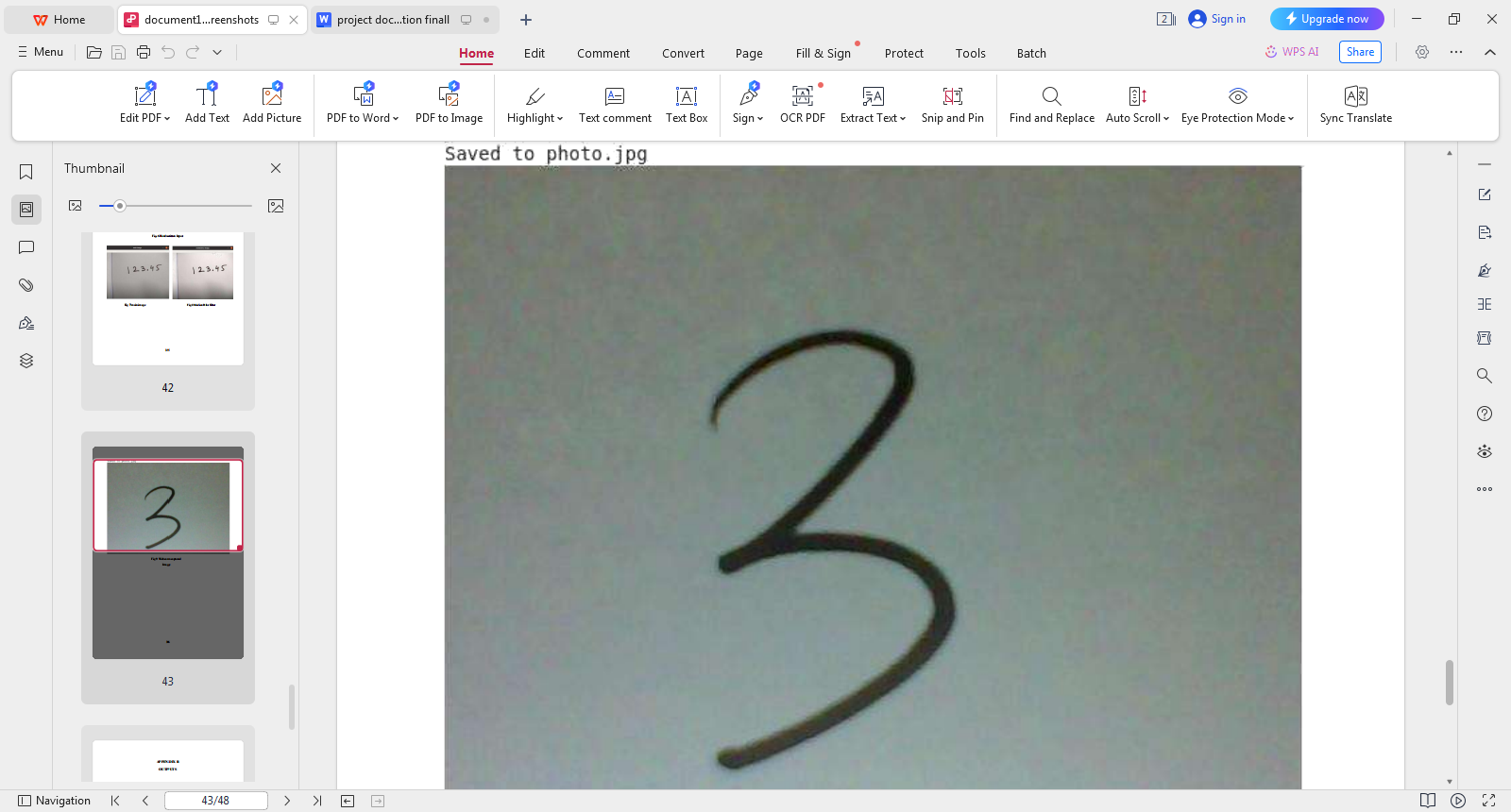
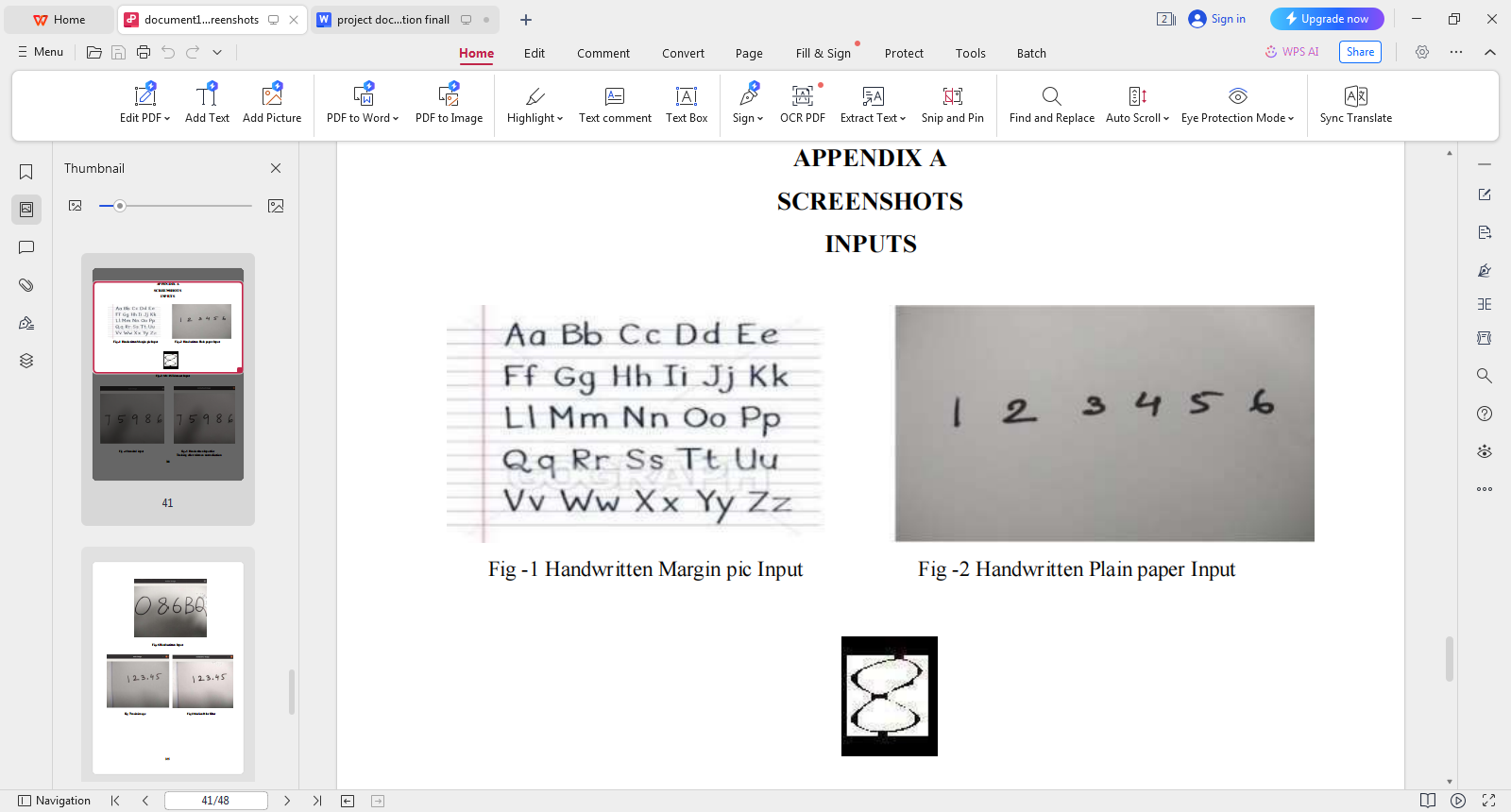
This Table depicts the accuracy count for each iteration in the neural network model.Every time

training takes place it comes with some upgradation on some models.Here 0-9 has been tested 5

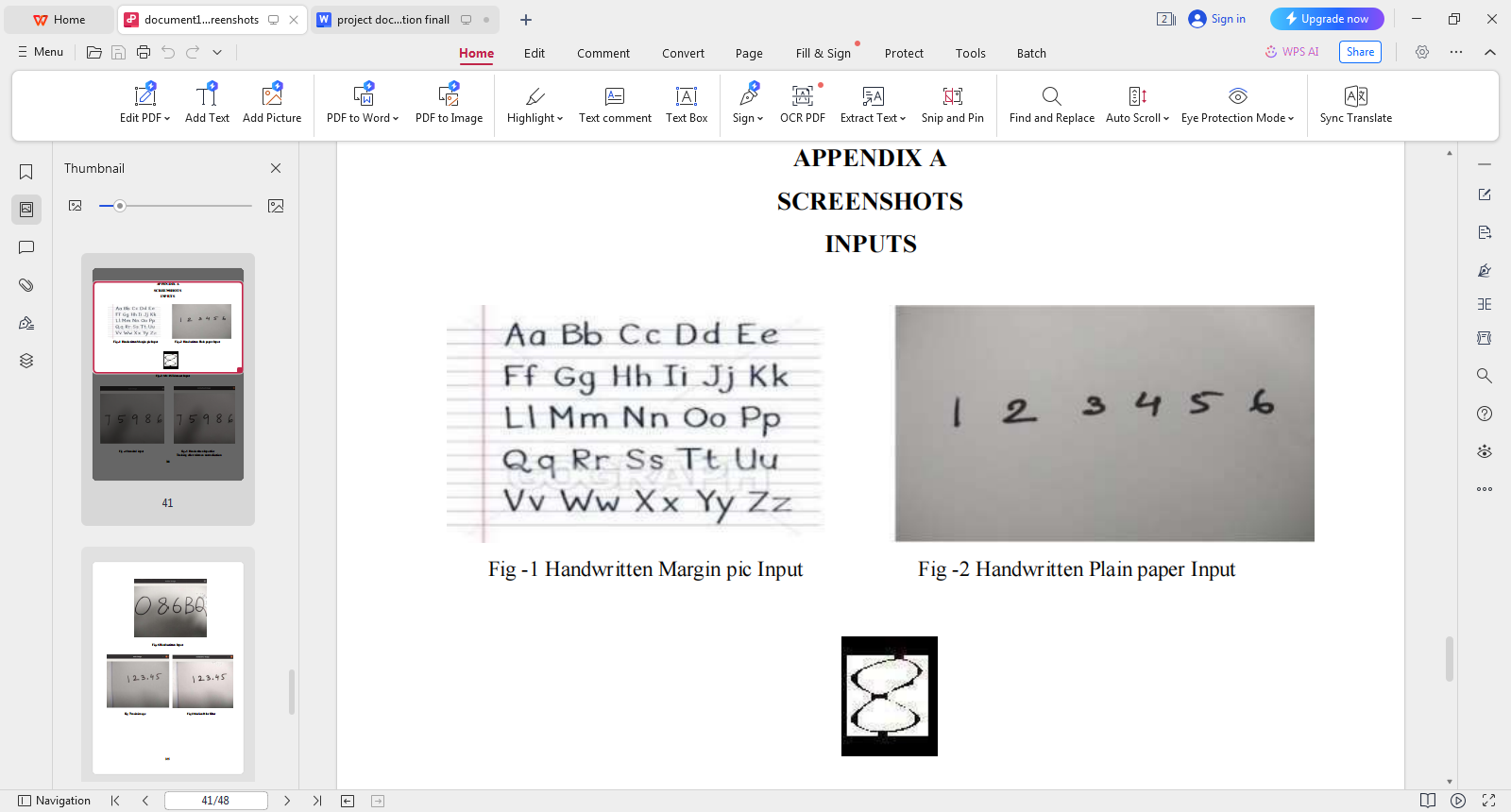
times. Actually 140 datasets have been tested every time.

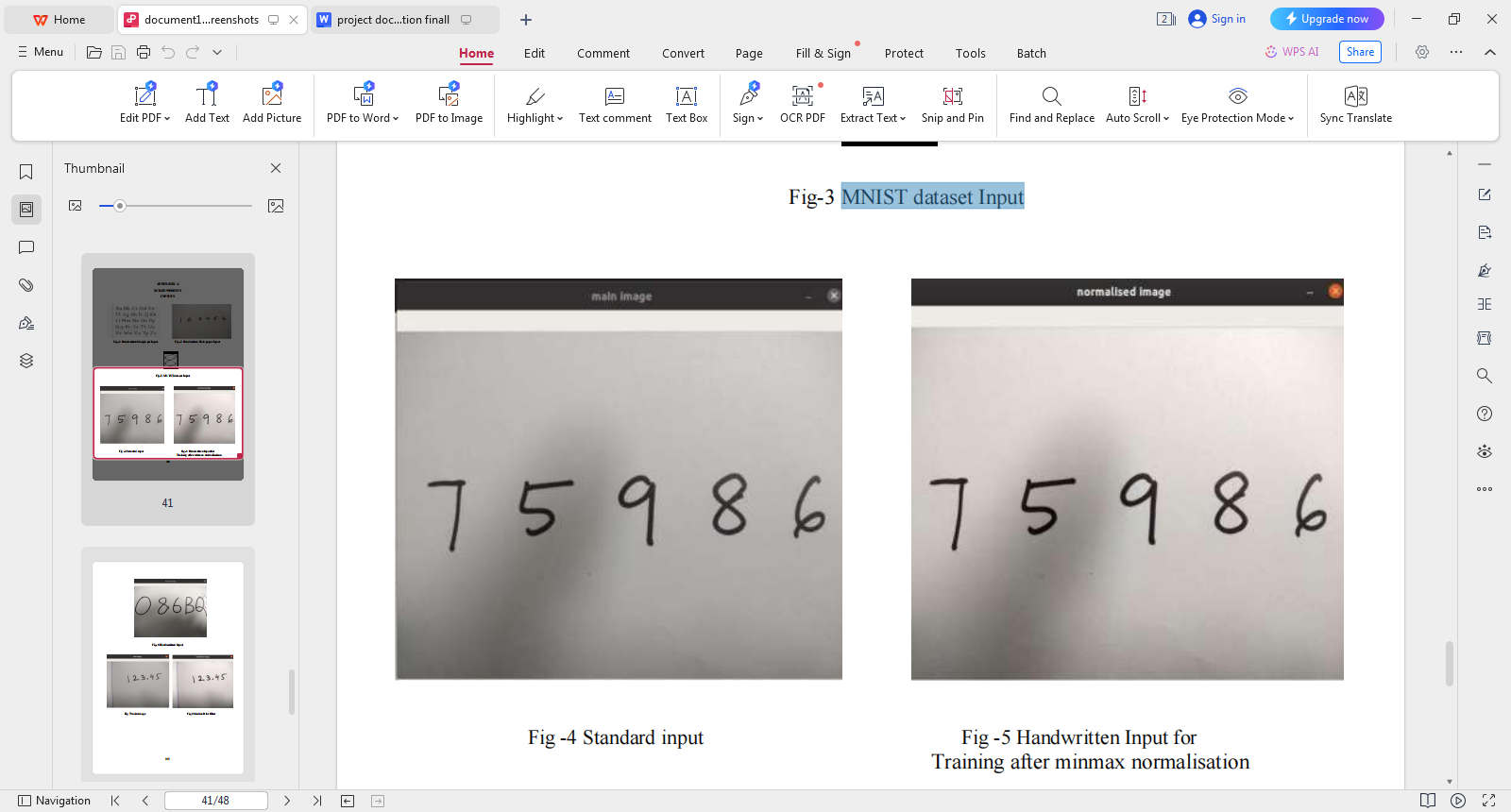
**7.1 SCREEN SHOTS**

**INPUTS**



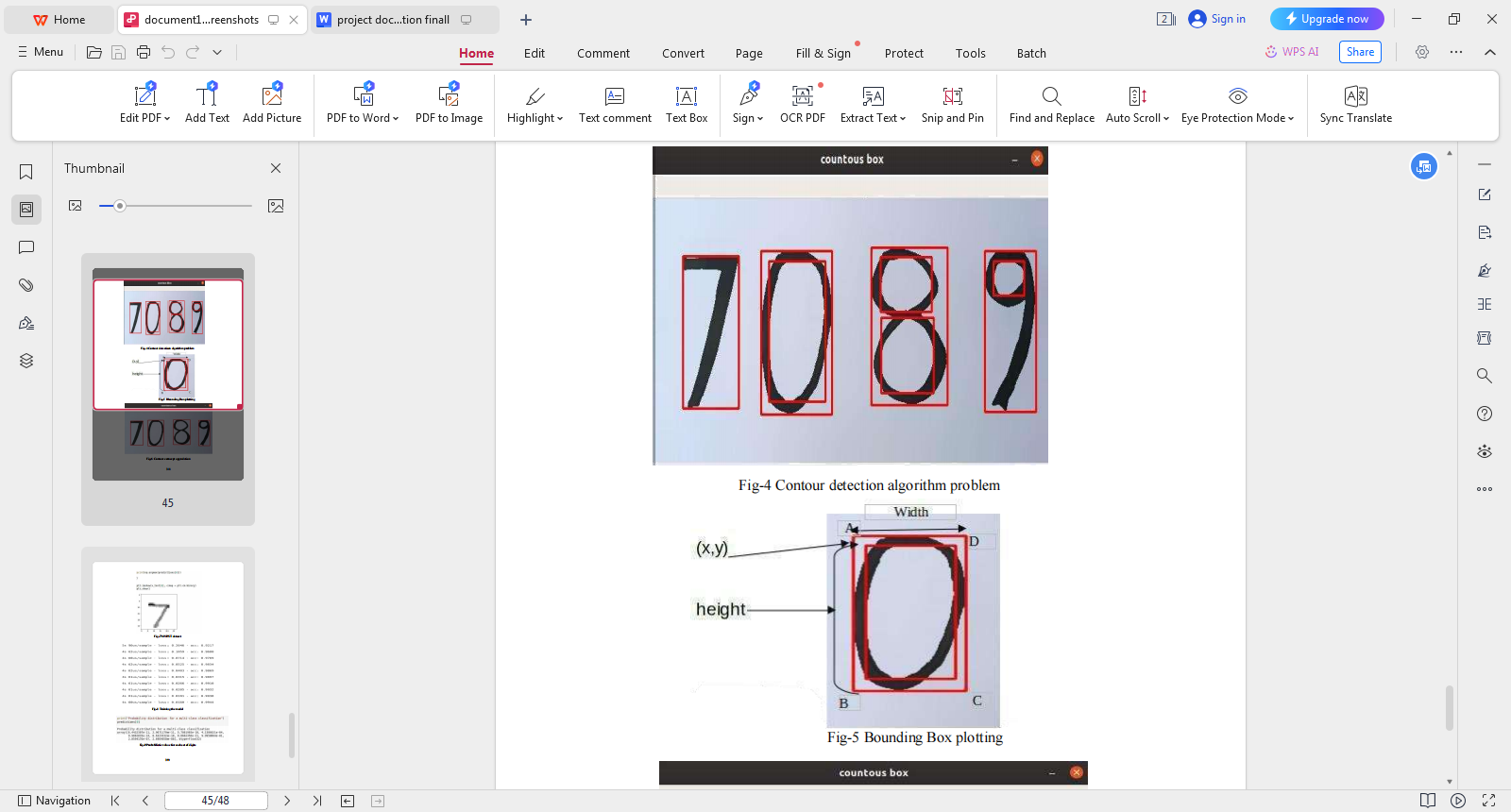
**Webcam captured Handwritten in plain paper input**

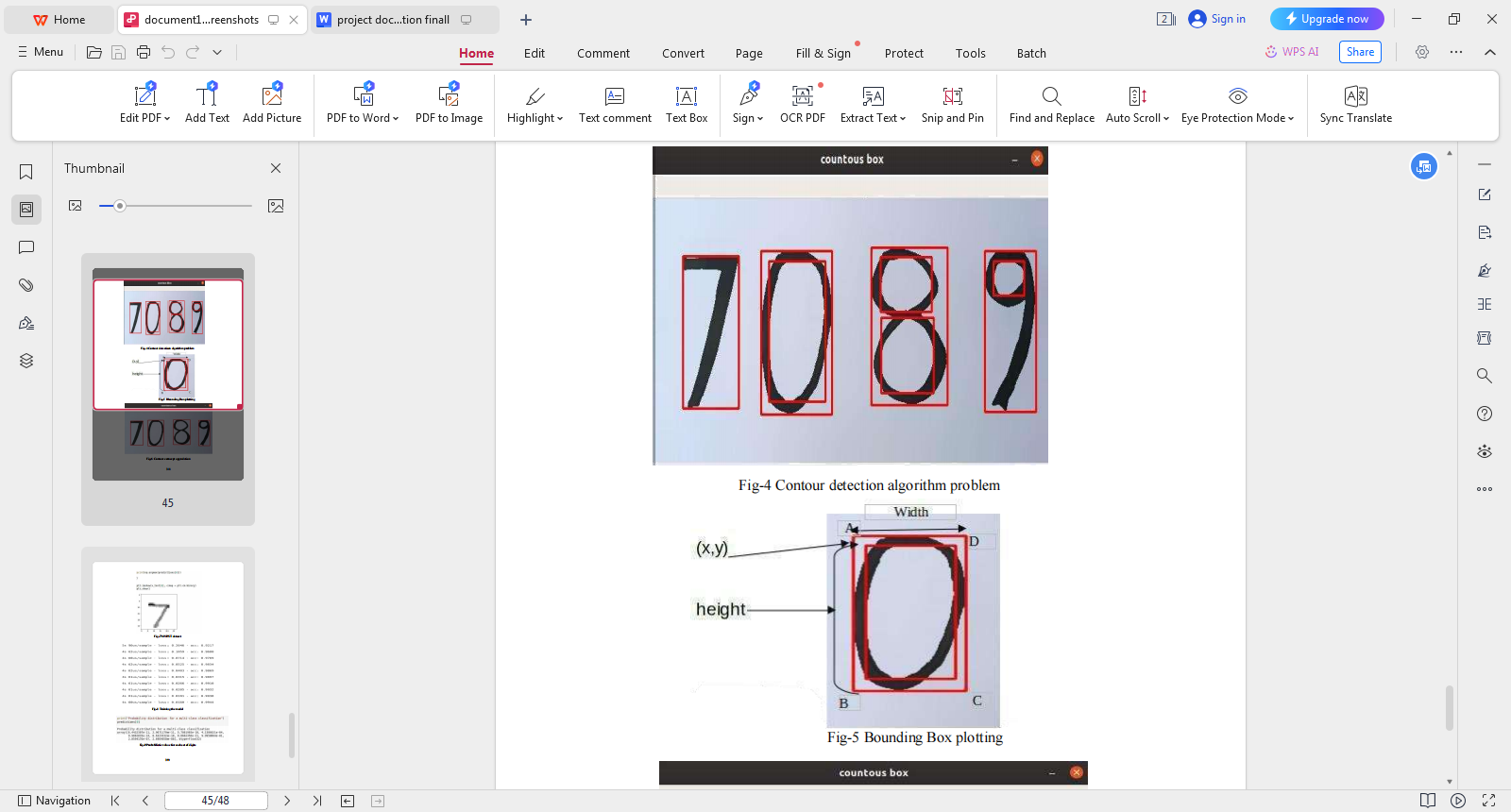
****

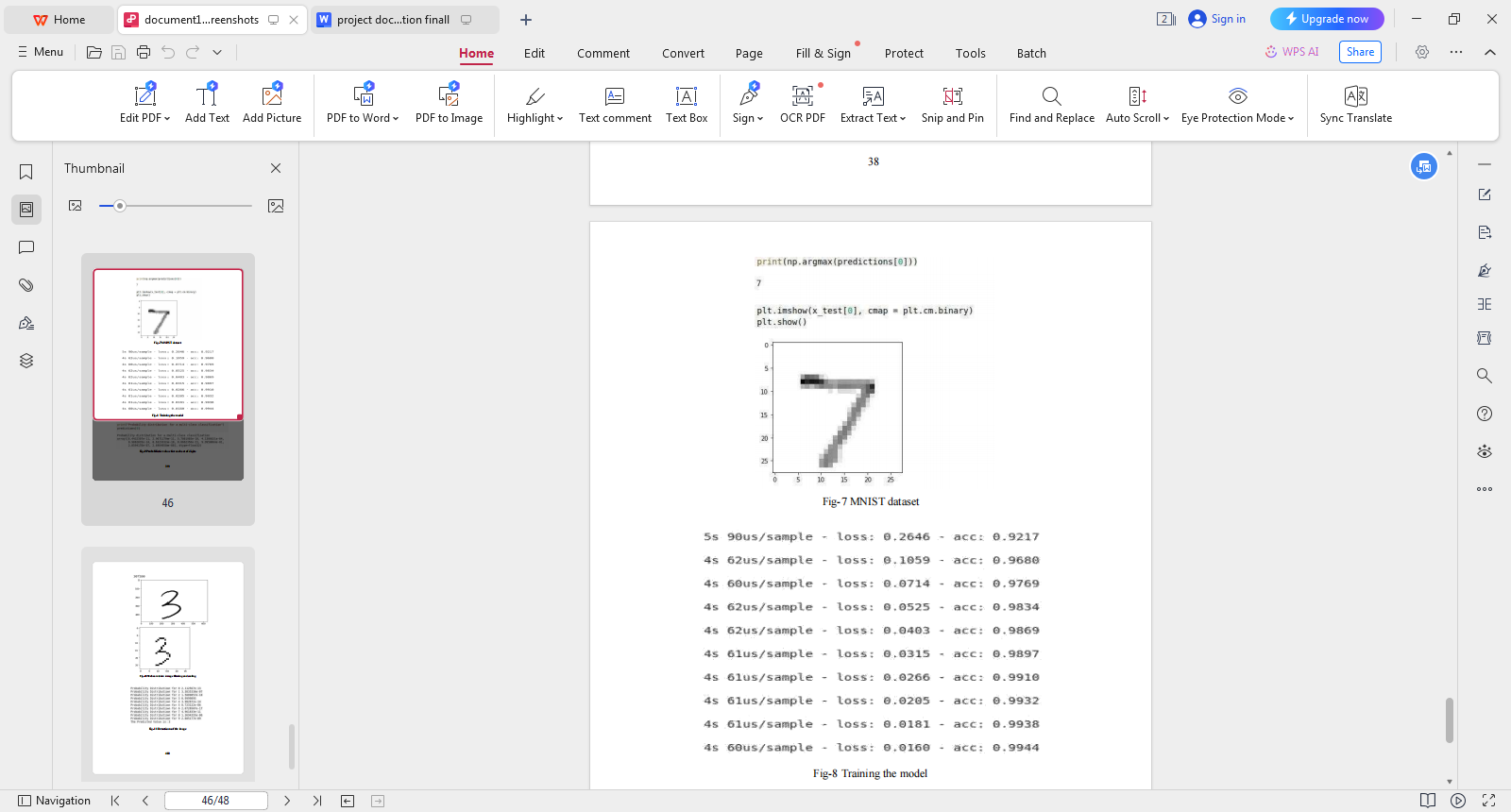
****

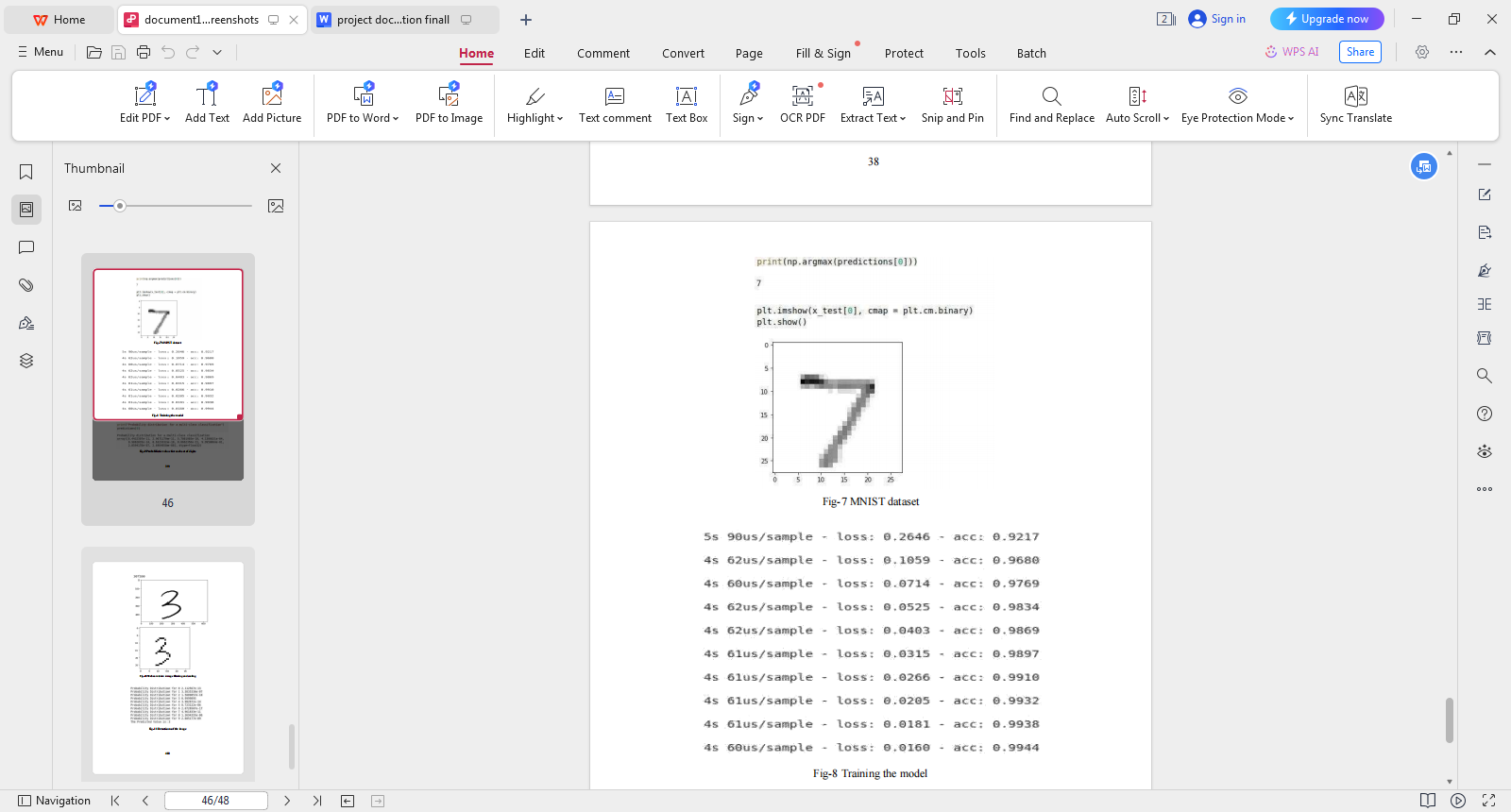
**MNIST dataset input Standard input**

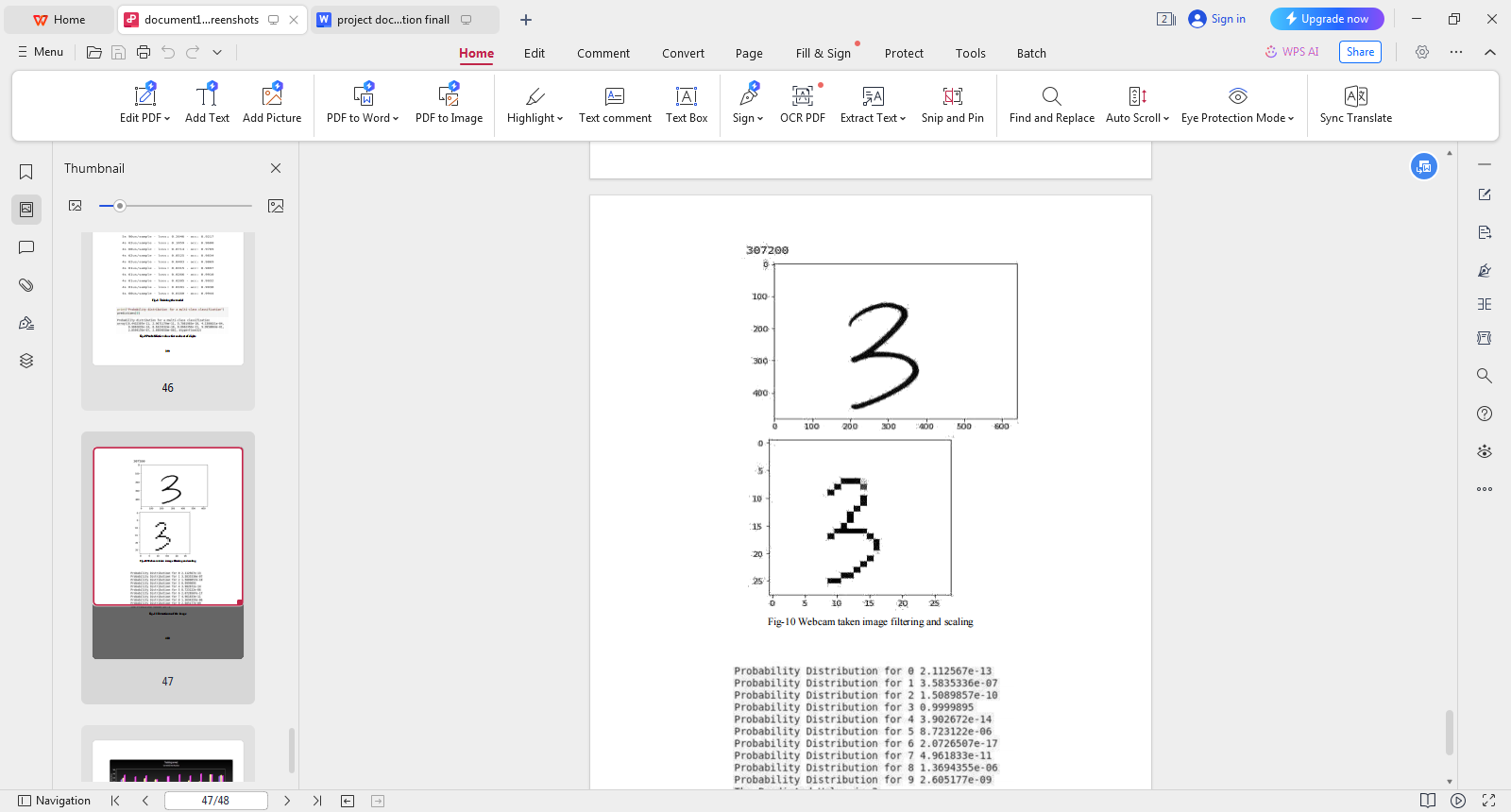
**OUTPUTS**



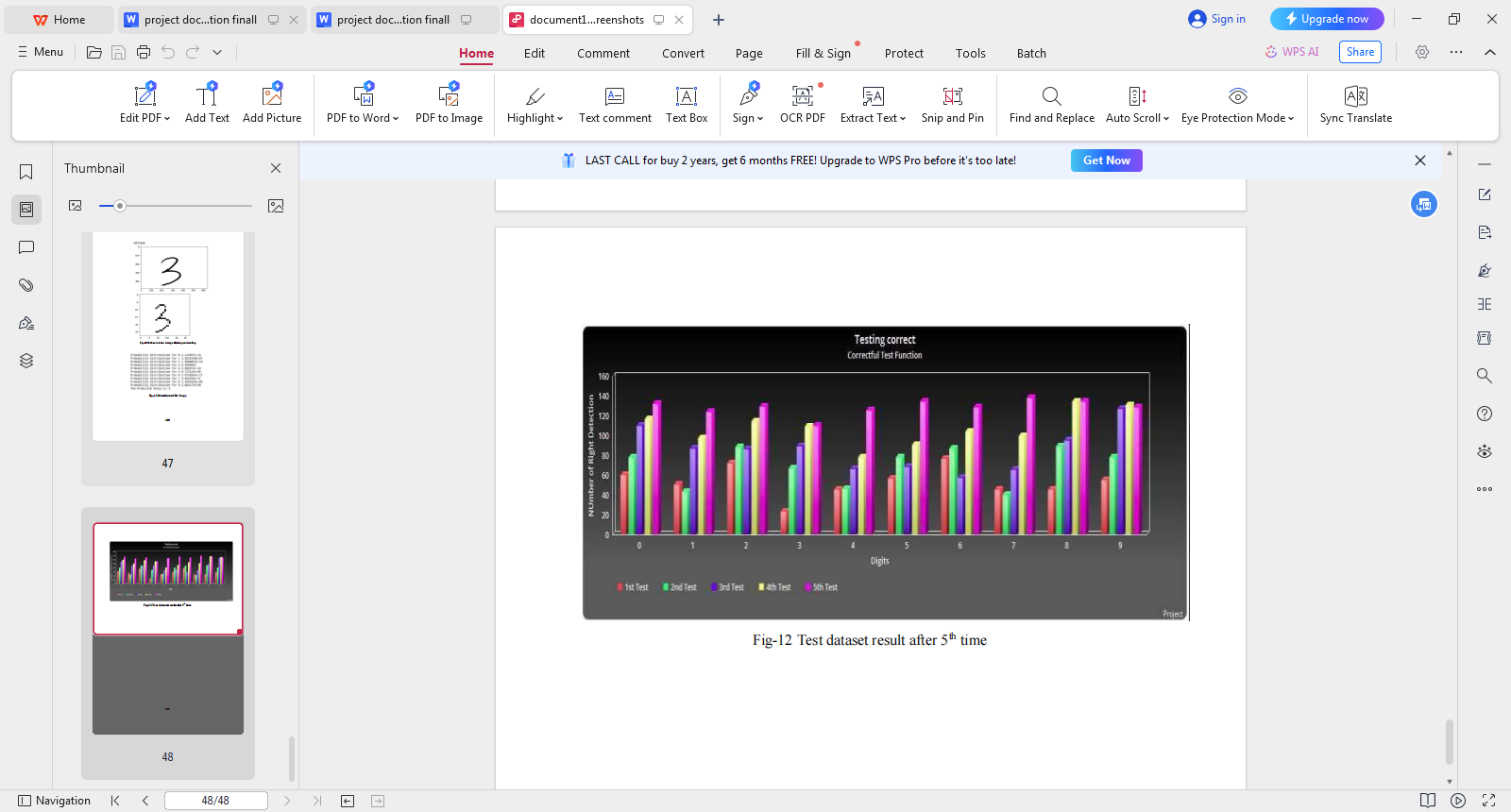




**MNIST Dataset**

**Training the Model**

**Webcam taken image filtering and scaling**



**Test dataset result after 5th time**

**7.2 SOURCE CODE**

**Import packages:**

from scipy.io import loadmat

import numpy as np

from Model import neural\_network

from RandInitialize import initialise

from Prediction import predict

from scipy.optimize import minimize

### ****Main.py:****

### # Loading mat file

data = loadmat('mnist-original.mat')

# Extracting features from mat file

X = data['data']

X = X.transpose()

# Normalizing the data

X = X / 255

# Extracting labels from mat file

y = data['label']

y = y.flatten()

# Splitting data into training set with 60,000 examples

X\_train = X[:60000, :]

y\_train = y[:60000]

# Splitting data into testing set with 10,000 examples

X\_test = X[60000:, :]

y\_test = y[60000:]

m = X.shape[0]

input\_layer\_size = 784 # Images are of (28 X 28) px so there will be 784 features

hidden\_layer\_size = 100

num\_labels = 10 # There are 10 classes [0, 9]

# Randomly initialising Thetas

initial\_Theta1 = initialise(hidden\_layer\_size, input\_layer\_size)

initial\_Theta2 = initialise(num\_labels, hidden\_layer\_size)

# Unrolling parameters into a single column vector

initial\_nn\_params = np.concatenate((initial\_Theta1.flatten(), initial\_Theta2.flatten()))

maxiter = 100

lambda\_reg = 0.1 # To avoid overfitting

myargs = (input\_layer\_size, hidden\_layer\_size, num\_labels, X\_train, y\_train, lambda\_reg)

# Calling minimize function to minimize cost function and to train weights

results = minimize(neural\_network, x0=initial\_nn\_params, args=myargs,

options={'disp': True, 'maxiter': maxiter}, method="L-BFGS-B", jac=True)

nn\_params = results["x"] # Trained Theta is extracted

# Weights are split back to Theta1, Theta2

Theta1 = np.reshape(nn\_params[:hidden\_layer\_size \* (input\_layer\_size + 1)], (

hidden\_layer\_size, input\_layer\_size + 1)) # shape = (100, 785)

Theta2 = np.reshape(nn\_params[hidden\_layer\_size \* (input\_layer\_size + 1):],

(num\_labels, hidden\_layer\_size + 1)) # shape = (10, 101)

# Checking test set accuracy of our model

pred = predict(Theta1, Theta2, X\_test)

print('Test Set Accuracy: {:f}'.format((np.mean(pred == y\_test) \* 100)))

# Checking train set accuracy of our model

pred = predict(Theta1, Theta2, X\_train)

print('Training Set Accuracy: {:f}'.format((np.mean(pred == y\_train) \* 100)))

# Evaluating precision of our model

true\_positive = 0

for i in range(len(pred)):

if pred[i] == y\_train[i]:

true\_positive += 1

false\_positive = len(y\_train) - true\_positive

print('Precision =', true\_positive/(true\_positive + false\_positive))

# Saving Thetas in .txt file

np.savetxt('Theta1.txt', Theta1, delimiter=' ')

np.savetxt('Theta2.txt', Theta2, delimiter=' ')