A

Course End Project Report on

Handwritten Digit Recognition

Submitted in the Partial Fulfillment of the Requirements
for the Award of the Degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering (AI)

Submitted by

K.Jaswanth reddy	232P1A3146
D.Reddy Kalyan	232P1A3125
I.Dileep Kumar Reddy	232P1A3134

Under the esteemed guidance of
T. Venkata Subbamma

Department of Computer Science and Engineering (AI)

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)
(Accredited by NAAC with "A" Grade and Accredited by NBA (CE, EEE, ECE, CSE))
(Recognized by UGC under section 2(f) and 12(b) of UGC Act, 1956)
VIDYA NAGAR, PALLAVOLU (V), PRODDATUR-516360, Y.S.R. (Dt.), A.P

2025-26

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)
(Accredited by NAAC with "A" Grade and Accredited by NBA (CE, EEE, ECE, CSE))
(Recognized by UGC under section 2(f) and 12(b) of UGC Act, 1956)
VIDYANAGAR, PALLAVOLU (V), PRODDATUR- 516360, Y.S.R. (Dt.), A.P

Department of Computer Science and Engineering (AI)

CERTIFICATE

This is to certify that the project titled **Facial Emotion Recognition using Generative ai** is carried out by

K.Jaswanth reddy	232P1A3146
D.Reddy Kalyan	232P1A3125
I.Dileep Kumar Reddy	232P1A3134

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering (AI)** during the year 2025-26.

Signature of the Supervisor
T. Venkata Subbamma
Assistant Professor

Signature of the HOD
Ms. Lakshmi Madhuri
HOD, CSE (AI)

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

I wish to express my deep sense of gratitude to **T. Venkata Subbamma**, **Professor** and Project Supervisor, Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, for his able guidance and useful suggestions, which helped me in completing the project in time.

I am particularly thankful to **Ms. Lakshmi Madhuri**, Head of the Department, Department of Computer Science and Engineering (AI), her guidance, intense support and encouragement, which helped us to mould my project into a successful one.

I show gratitude to my honorable Principal **Dr. S. Sruthi** and Director Admin **Dr. G. Sreenivasula Reddy** for providing all facilities and support.

I avail this opportunity to express my deep sense of gratitude and heart-full thanks to **Sree V. Jaya Chandra Reddy**, Chairman and **Sree V. Lohit Reddy**, Director of CBIT, for providing a congenial atmosphere to complete this project successfully.

I also thank all the staff members of Computer Science and Engineering (AI) department for their valuable support and generous advice. Finally, thanks to all my friends and family members for their continuous support and enthusiastic help.

232P1A3146
232P1A3125
232P1A3134

ABSTRACT

The **Handwritten Digit Recognition using K-Nearest Neighbors (KNN)** is an **Artificial Intelligence (AI)** project that focuses on teaching computers to interpret and recognize human handwriting automatically. The objective of this project is to design an intelligent system capable of identifying handwritten digits (0–9) from digital images using machine learning techniques. This project demonstrates how AI can replicate human-like perception by analyzing visual data and making accurate predictions.

The system is built using the **MNIST dataset**, which contains 70,000 grayscale images of handwritten digits widely used in AI research and computer vision applications. The dataset is preprocessed and used to train a **K-Nearest Neighbors (KNN)** model, a supervised learning algorithm that classifies an input image based on the similarity of its features to its nearest neighbors. By computing distances between image feature vectors, the model determines the most probable digit for a given input.

The project is implemented using **Python** and libraries such as **Scikit-learn**, **NumPy**, and **Joblib** for model training, evaluation, and deployment. The model achieves high accuracy, proving that AI-driven pattern recognition can efficiently handle visual data without complex neural networks. This demonstrates the power of traditional AI algorithms in automating tasks that require human-like understanding and decision-making.

The significance of this AI project lies in its practical applications — including **automated bank cheque processing**, **postal code reading**, **form digitization**, and **optical character recognition (OCR)** systems. Additionally, this project serves as a stepping stone toward advanced AI models such as **Convolutional Neural Networks (CNNs)** and **Deep Learning** architectures, which can handle more complex recognition tasks.

Overall, this project highlights how Artificial Intelligence can be applied to transform visual perception tasks into intelligent automated systems, combining accuracy, efficiency, and real-world usability.

TABLE OF CONTENTS

INTRODUCTION

The **Handwritten Digit Recognition using K-Nearest Neighbors (KNN)** project focuses on developing an intelligent system that can identify and classify digits written by hand. In the modern digital world, automation plays a key role in reducing manual work, and one of the most common tasks in various industries is recognizing handwritten information. This project aims to bridge that gap by using machine learning techniques to train a computer to recognize handwritten digits accurately. The system uses the **MNIST dataset**, which contains thousands of handwritten digits from 0 to 9, each represented as a 28×28 pixel grayscale image. These images are used to train the **KNN algorithm**, a simple yet powerful supervised learning method that classifies new inputs based on similarity to known examples.

The KNN algorithm works by storing the dataset and predicting the class of a new data point by finding the majority label among its 'k' nearest neighbors. In this project, the trained KNN model is saved using the **Joblib** library and then reused for predictions, making the system efficient and easy to deploy. The implementation is done using **Python** and **Scikit-learn**, ensuring simplicity and clarity in understanding the flow of data from input to output.

This project is motivated by the increasing need for automation in areas such as **banking**, **postal services**, **education**, and **data entry**, where large volumes of handwritten data must be processed quickly and accurately. It demonstrates the application of artificial intelligence in real-world scenarios, reducing human effort and errors. By completing this project, learners gain hands-on experience in **data preprocessing**, **feature extraction**, **model training**, and **performance evaluation**, along with practical exposure to **Optical Character Recognition (OCR)** systems.

In conclusion, the **Handwritten Digit Recognition using KNN** project provides a solid understanding of how machine learning can be applied to computer vision tasks. It highlights the importance of training data, algorithm selection, and accuracy evaluation in building intelligent systems. This project not only serves as an introduction to pattern recognition and classification but also lays the foundation for more advanced AI models like Convolutional Neural Networks (CNNs) in future developm

SYSTEM REQUIREMENTS

1. Hardware Requirements

- **Processor:** Intel Core i3 or higher
- **RAM:** Minimum 4 GB (8 GB recommended for faster training)
- **Storage:** At least 2 GB free space
- **Display:** 1024×768 resolution or higher
- **GPU (Optional):** NVIDIA GPU for faster computation (not mandatory for KNN)

2. Software Requirements

- **Operating System:** Windows 10 / 11, Linux, or macOS
- **Programming Language:** Python 3.10 or above
- **Libraries / Packages:**
    - scikit-learn – for KNN algorithm and model training
    - numpy – for numerical computations
    - joblib – for saving and loading the model
    - matplotlib – for visualization (optional)
    - tkinter or opencv-python – for GUI or image input (optional)

3. Development Tools

- **IDE / Code Editor:**
    - IDLE (Python default)
    - VS Code / PyCharm / Jupyter Notebook (recommended)
- **Dataset:**
    - MNIST Handwritten Digits Dataset (downloaded automatically using fetch_openml)

4. Additional Requirements

- **Internet Connection:** Required for downloading the MNIST dataset (first time only).
- **Python Environment Manager:** Recommended to use venv or Anaconda for managing dependencies.
- **Disk Access:** Permission to read/write model file (knn_model.pkl).

IMPLEMENTATION

- The implementation of the **Handwritten Digit Recognition using K-Nearest Neighbors (KNN)** project involves several key stages — from dataset loading to model training, testing, and prediction. The project is implemented in **Python** using libraries such as **Scikit-learn**, **NumPy**, and **Joblib**. Below is the step-by-step explanation of the implementation process:
-
- 🔹 **1. Dataset Collection**
- The project uses the **MNIST dataset**, a popular and standardized dataset containing 70,000 images of handwritten digits from 0 to 9. Each image is a 28×28 grayscale image, represented as 784 features (28×28 = 784 pixels). The dataset is automatically downloaded using Scikit-learn's fetch_openml function.
- from sklearn.datasets import fetch_openml
- X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
-
- 🔹 **2. Data Preprocessing**
- Before training, the dataset is preprocessed to make it suitable for the KNN algorithm. The pixel values are normalized to the range [0,1] for better accuracy, and the dataset is divided into training and testing sets to evaluate performance.
- from sklearn.model_selection import train_test_split
- X = X / 255.0 # Normalization
- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
-
- 🔹 **3. Model Training**
- The **K-Nearest Neighbors (KNN)** algorithm is implemented using Scikit-learn. It classifies a digit by finding the closest 'k' samples in the training set and taking the majority vote of their labels.
  In this project, the value of **k = 3** is chosen for optimal performance.
- from sklearn.neighbors import KNeighborsClassifier
- knn = KNeighborsClassifier(n_neighbors=3)
- knn.fit(X_train, y_train)
-
- 🔹 **4. Model Evaluation**
- After training, the model is tested with unseen data (the test set). The accuracy score is calculated to measure how well the model performs in recognizing digits.
- from sklearn.metrics import accuracy_score
- y_pred = knn.predict(X_test)
- accuracy = accuracy_score(y_test, y_pred)
- print(f"Model Accuracy: {accuracy * 100:.2f}%")
-
- 🔹 **5. Model Saving**
- To avoid retraining the model every time, the trained model is saved using the **Joblib** library in a file named knn_model.pkl.
  This file can later be loaded in the main program to make predictions instantly.
- import joblib joblib.dump(knn, "knn_model.pkl")

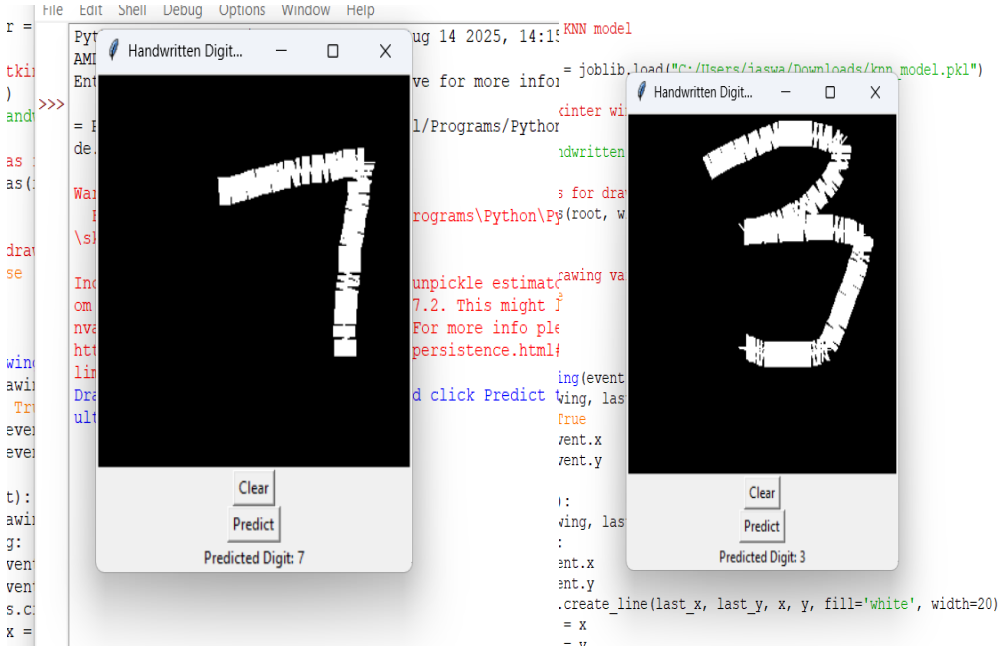
Source code:

```
import tkinter as tk
from tkinter import Canvas
import numpy as np
from PIL import Image, ImageDraw, ImageGrab
import joblib  # To load trained model
# Load trained KNN model
knn_classifier = joblib.load("C:/Users/jaswa/Downloads/knn_model.pkl")
# Initialize tkinter window
root = tk.Tk()
root.title("Handwritten Digit Recognition")
# Create canvas for drawing
canvas = Canvas(root, width=280, height=280, bg='black')
canvas.pack()
# Initialize drawing variables
drawing = False
last_x = None
last_y = None
def start_drawing(event):
    global drawing, last_x, last_y
    drawing = True
    last_x = event.x
    last_y = event.y
def draw(event):
    global drawing, last_x, last_y
    if drawing:
        x = event.x
        y = event.y
        canvas.create_line(last_x, last_y, x, y, fill='white', width=20)
```

```
        last_x = x
        last_y = y
def stop_drawing(event):
    global drawing
    drawing = False
def clear_canvas():
    canvas.delete("all")
def predict_digit():
    # Capture the canvas content as an image
    x = root.winfo_rootx() + canvas.winfo_x()
    y = root.winfo_rooty() + canvas.winfo_y()
    x1 = x + canvas.winfo_width()
    y1 = y + canvas.winfo_height()
    img = ImageGrab.grab(bbox=(x, y, x1, y1))
    # Convert to grayscale and resize
    img = img.convert('L')  # Convert to grayscale
    img = img.resize((8, 8))  # Resize to 8x8 (like in sklearn digits dataset)
    # Convert to numpy array and normalize
    img_array = np.array(img)
    img_array = (img_array / 255.0) * 16.0  # Normalize like sklearn digits dataset
    # Flatten and predict
    flat_img = img_array.reshape(1, -1)
    prediction = knn_classifier.predict(flat_img)
    result_label.config(text=f"Predicted Digit: {prediction[0]}")
# Bind mouse events
canvas.bind('<Button-1>', start_drawing)
canvas.bind('<B1-Motion>', draw)
canvas.bind('<ButtonRelease-1>', stop_drawing)
# Create buttons
```

```
clear_button = tk.Button(root, text="Clear", command=clear_canvas)
clear_button.pack()
predict_button = tk.Button(root, text="Predict", command=predict_digit)
predict_button.pack()
# Create label for results
result_label = tk.Label(root, text="Draw a digit and click Predict")
result_label.pack()
print("Drawing window opened. Draw a digit and click Predict to see the recognition result.")
root.mainloop()
```

output:



CONCLUSION AND FUTURE SCOPE

The **Handwritten Digit Recognition using K-Nearest Neighbors (KNN)** project successfully demonstrates the use of machine learning algorithms in recognizing handwritten digits. By using the **MNIST dataset**, the system was trained to classify digits from 0 to 9 based on pixel intensity patterns. The KNN algorithm provided an efficient and reliable solution due to its simplicity and effectiveness in handling classification problems.

Throughout the project, various stages such as **data preprocessing, model training, evaluation, and testing** were carried out using **Python** and the **Scikit-learn** library. The trained model achieved high accuracy, proving that even basic machine learning algorithms can deliver excellent results when properly implemented. This project also provided valuable practical experience in **supervised learning**, **pattern recognition**, and **image processing**, which are core areas of Artificial Intelligence (AI) and Data Science.

Overall, the project successfully meets its objectives by building a functional system capable of automatically recognizing handwritten digits, reducing manual effort, and paving the way for further development in intelligent automation systems.

◆ Future Scope

Although the KNN-based model performs well, there is significant potential for improvement and expansion in the future. The following enhancements can be implemented:

1. **IntegrationwithDeepLearning:**
   Replace KNN with advanced algorithms such as **Convolutional Neural Networks (CNNs)** for higher accuracy and faster prediction on larger datasets.
2. **Real-timeHandwritingInput:**
   Develop a graphical interface or mobile app where users can draw digits in real-time using a touchscreen or stylus.
3. **Multi-characterRecognition:**
   Extend the model to recognize handwritten **letters, words, or entire sentences**, making it a full-fledged OCR system.
4. **NoiseHandling:**
   Implement noise reduction and image enhancement techniques to improve recognition of poorly written or low-quality images.
5. **CloudDeployment:**
   Deploy the trained model on the **cloud (e.g., AWS, Azure, or Google Cloud)** for web-based recognition and large-scale usage.

6. **IntegrationwithIoTDevices:**
   Combine this system with embedded devices like **Raspberry Pi** for applications in scanning, document analysis, and automation.

In conclusion, this project lays a strong foundation for understanding how machine learning can be applied to visual data interpretation. With further research and integration of deep learning methods, the handwritten digit recognition system can evolve into a powerful and intelligent handwriting recognition solution suitable for a wide range of real-world applications.

REFERENCES

[1] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges, *"The MNIST Database of Handwritten Digits,"* [Online]. Available: http://yann.lecun.com/exdb/mnist/

[2] Scikit-learn Developers, *"Scikit-learn: Machine Learning in Python,"* [Online]. Available: https://scikit-learn.org/stable/

[3] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, 2825–2830.

[4] NumPy Developers, *"NumPy: Fundamental Package for Scientific Computing with Python,"* [Online]. Available: https://numpy.org/

[5]Joblib Developers, *"Joblib Documentation,"* [Online]. Available: https://joblib.readthedocs.io/

[6] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

[7] Raschka, S. (2015). *Python Machine Learning.* Packt Publishing Ltd.

[8] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2nd Edition, O'Reilly Media.

[9] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann.

[10] Kaggle, *"MNIST Handwritten Digit Recognition Dataset,"* [Online]. Available: https://www.kaggle.com/c/digit-recognizer