MNIST Handwritten Digits Classification Using CNN

Assistant Professor Mr. Balaji C[*1], Mr. Chandhiran S[*2]

[*1]Assistant Professor, Department of Commerce with Business Analytics, Dr. N.G.P. Arts and Science College, Coimbatore, Tamil Nadu, India.

[*2]Student, Department of Commerce with Business Analytics, Dr. N.G.P. Arts and Science College, Coimbatore, Tamil Nadu, India.

Abstract

A machine learning model was developed for handwritten digit recognition using Convolutional Neural Networks (CNNs). CNNs effectively classify images by learning patterns, outperforming traditional neural networks. A dataset of 1,000 custom images was created, with 80% for training and 20% for testing. Images were preprocessed (cleaning, scaling, reshaping) to optimize performance. A CNN model, built with TensorFlow and Keras, achieved 92.5% accuracy in recognizing digits (0–9). Key libraries include NumPy, Pandas, TensorFlow, and Keras. The study highlights CNNs' advantages in accuracy and efficiency, making them ideal for OCR, banking, and automated document processing.

Keywords: Machine Learning, Handwritten Digit Recognition, Convolutional Neural Networks (CNNs), Image Classification, Optical Character Recognition (OCR), Automated Document Processing.

INTRODUCTION

Handwriting recognition dates back to Crane's 1964 patented pen device, evolving through the SRI pen in the 1970s and commercial systems in the 1980s. Today, it plays a crucial role in automating information processing, converting handwritten text into machine-readable formats. Applications include license plate recognition, postal sorting, document digitization, and banking systems.

Advancements in deep learning and Convolutional Neural Networks (CNNs) have significantly improved recognition accuracy. CNNs excel in image classification, object detection, facial recognition, and speech processing. Their high efficiency and accuracy make them ideal for handwritten digit recognition, benefiting academic, financial, and business sectors by automating processes and enhancing digital transformation.

Objectives:

The primary goal of this research is to develop a model capable of accurately recognizing and classifying handwritten digits from images using Convolutional Neural Networks (CNNs). Although the focus is on digit recognition, the approach can be extended to recognize letters and even unique handwriting patterns of individuals.

1. Understanding and applying CNNs.
2. Developing a robust model.
3. Performance evaluation.
4. Enhancing real-world applications.

STATEMENT OF PROBLEM

Handwriting recognition has been a significant area of research for nearly four decades. This study focuses on analyzing the effectiveness of classification techniques, particularly Convolutional Neural Networks (CNNs), in recognizing and predicting handwritten digits from large datasets. The field of machine learning, especially deep learning approaches like CNNs and Artificial Neural Networks (ANNs), has greatly improved the accuracy and efficiency of recognition systems.

Several methodologies have been developed for handwritten digit recognition, broadly classified into four categories. Knowledge-based methods use predefined rules and heuristics for character recognition. Feature-based methods extract distinct characteristics such as shape, stroke direction, and curvature from handwritten digits. Template-based methods compare the input with stored templates to find the closest match. Appearance-based methods rely on statistical and deep learning models to recognize patterns in digit images.

Errors in handwritten digit recognition can have serious consequences. Misreading digits on bank cheques can cause financial discrepancies, while errors in postal sorting or automated form processing may lead to delays and miscommunication. Ensuring high accuracy in digit recognition is crucial to prevent such issues and enhance the reliability of automated systems.

METHODOLOGY

SVM for Classification

Support Vector Machine (SVM) is a powerful algorithm capable of performing both classification and regression tasks. However, it is most commonly used for classification problems due to its ability to create clear decision boundaries between different classes.

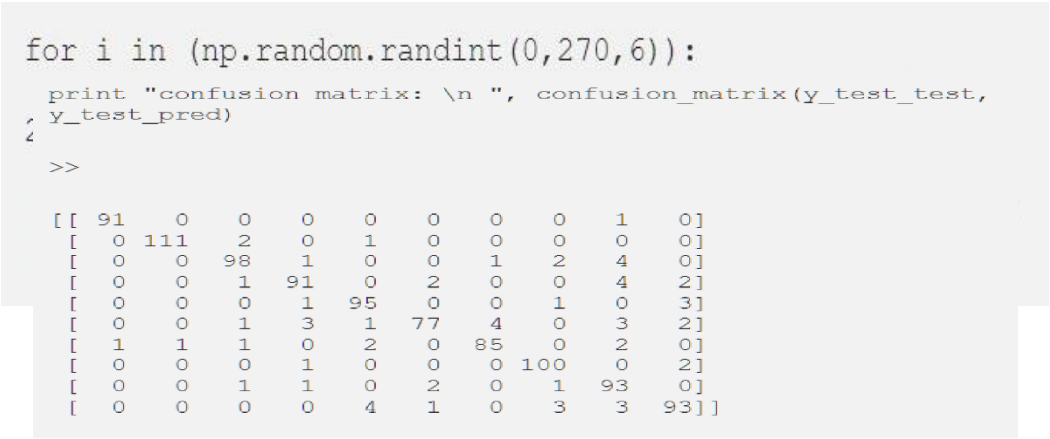
Non-Linear SVM

- In real-world applications, data is often not linearly separable, meaning a simple straight-line boundary is insufficient to classify data points accurately.

- Non-linear SVM addresses this by using kernel functions to map input data into a higher-dimensional space where it becomes linearly separable.
- This allows the model to capture more complex patterns and relationships between the data without requiring manual feature transformation.
- The downside of non-linear SVMs is their computational cost, as they require significantly more processing time compared to linear SVMs.

Despite its higher complexity, non-linear SVM is widely used in image recognition, text classification, and bioinformatics, where data relationships are intricate and difficult to separate using simple linear methods.

Fig. 1 - SVM Methodology Code



```
for i in (np.random.randint(0,270,6)):
  print "confusion matrix: \n ", confusion_matrix(y_test_test,
, y_test_pred)
<
  >>

  [[ 91   0   0   0   0   0   0   0   1   0]
   [  0 111   2   0   1   0   0   0   0   0]
   [  0   0  98   1   0   0   1   2   4   0]
   [  0   0   1  91   0   2   0   0   4   2]
   [  0   0   0   1  95   0   0   1   0   3]
   [  0   0   1   3   1  77   4   0   3   2]
   [  1   1   1   0   2   0  85   0   2   0]
   [  0   0   0   1   0   0   0 100   0   2]
   [  0   0   1   1   0   2   0   1  93   0]
   [  0   0   0   0   4   1   0   3   3  93]]
```

Fig 1. shows Support Vector Machine Methodology code using random, randint matplotlib.

Fig. 2 - Confusion Matrix

Fig 2. illustrates the Confusion Matrix for the Support Vector Machine (SVM). The Confusion Matrix is a tool used to assess the effectiveness of a classification model. It provides a comparison between the actual target values and the predicted values generated by the machine learning model. This comparison helps in evaluating the model's accuracy and identifying misclassifications

RESULTS AND DISCUSSION

CAPTURED IMAGES AND INPUT

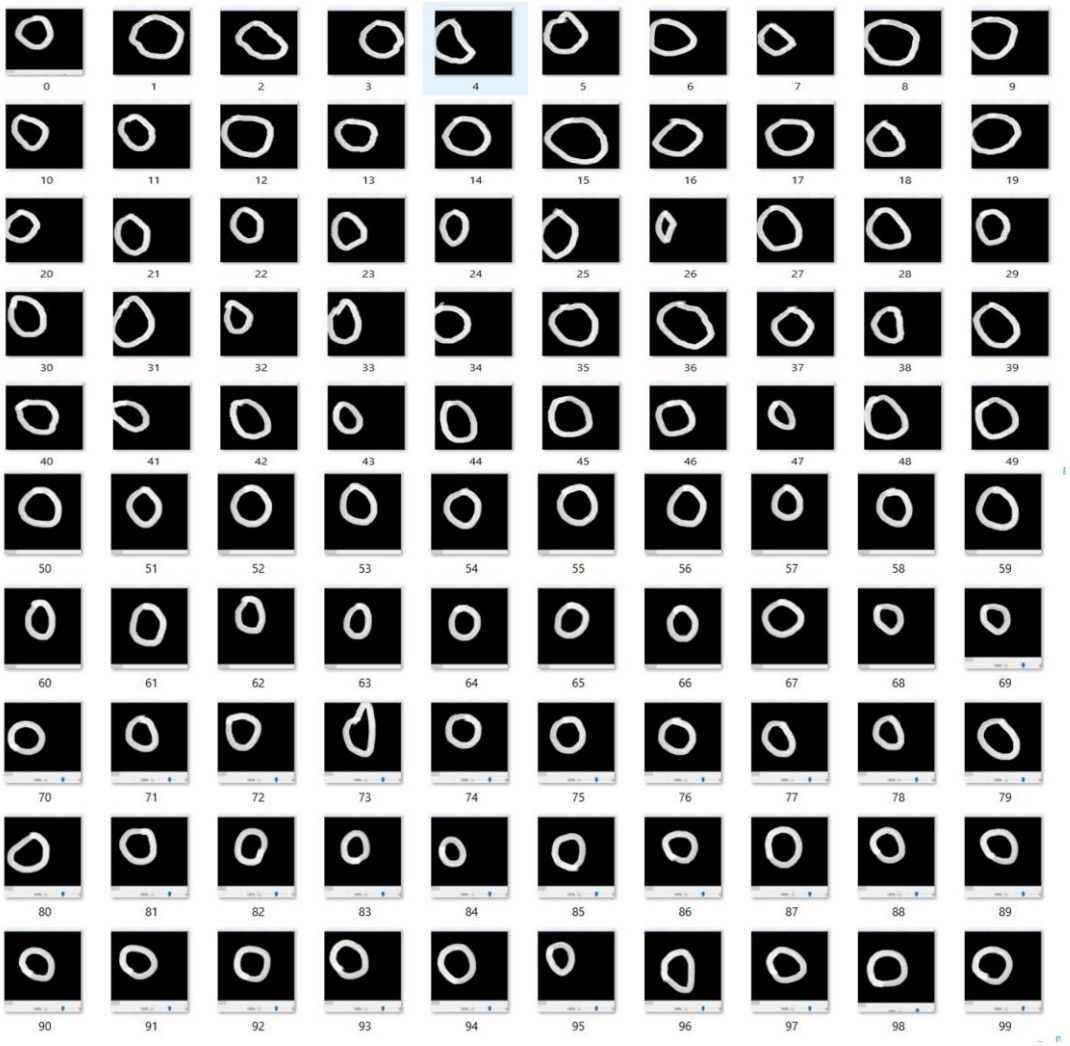


Fig. 3 - Captured Images

Fig 3. represents captured images obtained using the pyscreenshot package. These images are taken as input for further processing and analysis, as shown in the above figure.

PREDICTION OF DIGITS

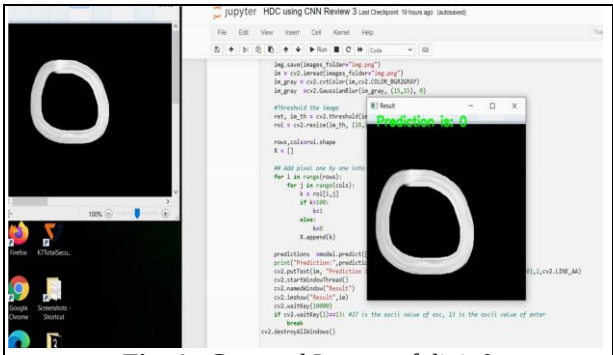


Fig. 4 - Captured Images of digit 0

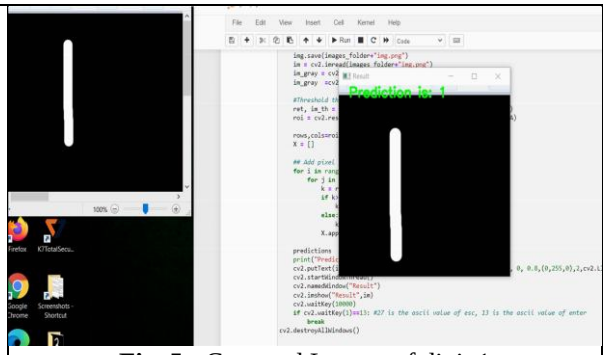


Fig. 5 - Captured Images of digit 1

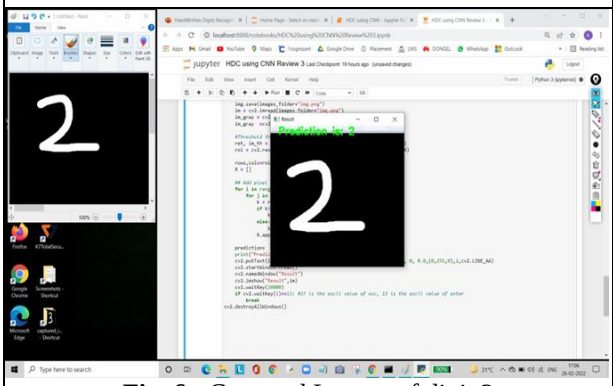


Fig. 6 - Captured Images of digit 2

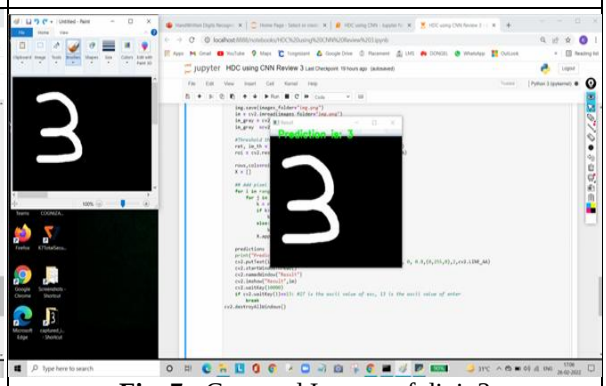


Fig. 7 - Captured Images of digit 3

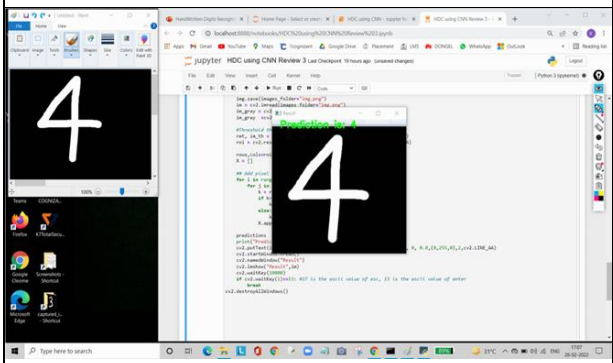


Fig. 8 - Captured Images of digit 4

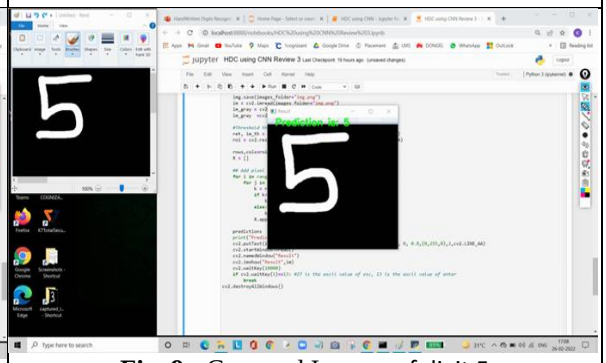


Fig. 9 - Captured Images of digit 5

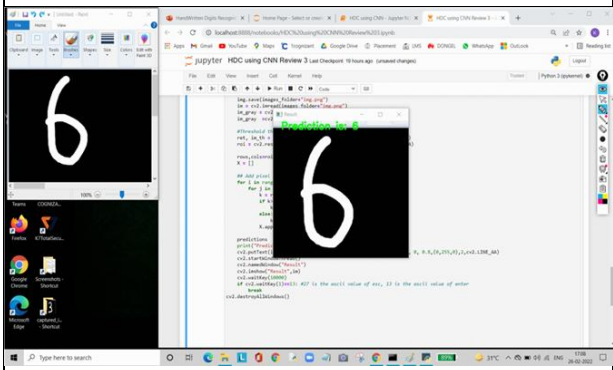


Fig. 10 - Captured Images of digit 6

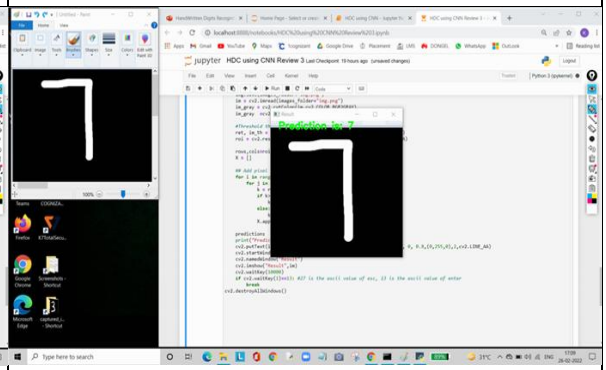


Fig. 11 - Captured Images of digit 7

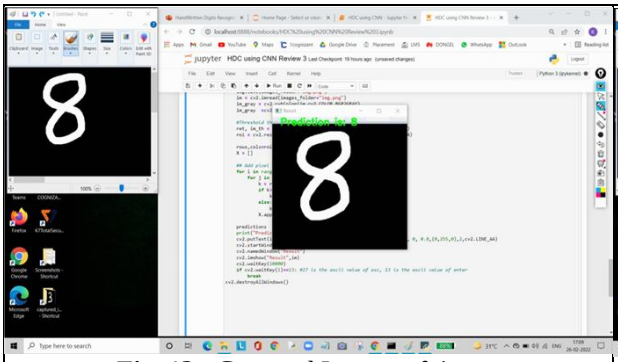


Fig. 12 - Captured Images of digit 8

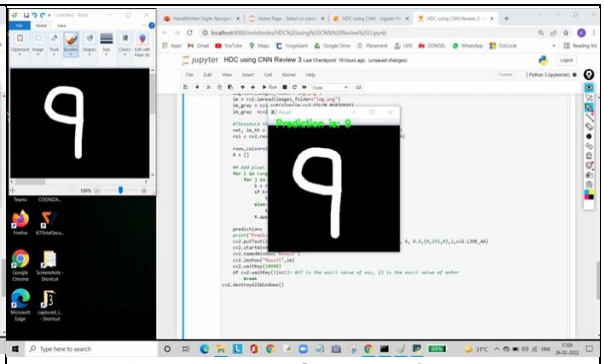


Fig. 13 - Captured Images of digit 9

ACCURACY

```
In [7]: from sklearn import metrics
        prediction=classifier.predict(test_x)
        print("Accuracy= ",metrics.accuracy_score(prediction, test_y)*100,"%")

        Accuracy=  92.5 %
```

Fig. 14 - Accuracy

Fig 14. displays the 92.5% accuracy achieved in Handwritten Digits Classification using a Convolutional Neural Network.

APPENDICES

```
import numpy as np

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to_categorical

# Load MNIST dataset

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Normalize pixel values to range 0-1

train_images = train_images.astype('float32') / 255.0

test_images = test_images.astype('float32') / 255.0

# Reshape images to include channel dimension (28x28x1)
```

```
train_images = np.expand_dims(train_images, axis=-1)
test_images = np.expand_dims(test_images, axis=-1)
# One-hot encode the labels
train_labels = to_categorical(train_labels, num_classes=10)
test_labels = to_categorical(test_labels, num_classes=10)
#Model Architecture Code
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
# Build CNN model
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(10, activation='softmax')
])
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# Model summary
model.summary()
```

Model Training Code

The following snippet shows how to train the model:

```
# Train the model
history = model.fit(
```

```
    train_images, train_labels,
    validation_split=0.2,
    epochs=20,
    batch_size=64
)
```

CONCLUSION & FUTURE SCOPE

Convolutional Neural Networks (CNNs) play a crucial role in handwritten digit classification. A dataset of 99 images was used, with 80 for training and 19 for testing, after preprocessing steps like cleaning, scaling, and reshaping. A CNN model, developed using TensorFlow, achieved 92.5% accuracy in recognizing digits (0–9).

The study demonstrated the efficiency of CNNs in feature extraction, pattern learning, and dimensionality reduction. The model's performance was evaluated based on different convolutional, pooling, and fully connected layers, ensuring high accuracy.

Future improvements will focus on hyperparameter tuning, data augmentation (rotation, scaling, flipping), and handling rotated digits (e.g., distinguishing between 6 and 9). Expanding functionality to colored images, multi-digit recognition, and character recognition will enhance real-world applications in banking, postal services, and document processing.

REFERENCES

Research Papers

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems (NeurIPS), 25, 1097-1105.
3. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Online Sources

1. TensorFlow Documentation. Building and training deep learning models.
   - Available at: https://www.tensorflow.org/
2. MNIST Dataset Overview. A standard dataset for machine learning.
   - Available at: http://yann.lecun.com/exdb/mnist/
3. PyTorch Documentation. Deep learning framework for AI research and development.
   - Available at: https://pytorch.org/
4. Wikipedia Contributors. Convolutional neural networks. Wikipedia.
   - Available at: https://en.wikipedia.org/wiki/Convolutional_neural_network
5. Chollet, F. Keras Documentation. Building deep learning models using Keras.
   - Available at: https://keras.io/

Additional References

Abadi, M., Barham, P., Chen, J., et al. (2016). TensorFlow: A system for large-scale machine learning. OSDI, 16, 265-283.