WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Handwritten Digit Recognition using Convolutional Neural Networks: A Deep Learning Approach

Raut Vipul Prakash, Anil Kumar Kadam

Department of Engineering, ME AI & DS, AISSMS College of Engineering, Pune, India, rautvipul25@gmail.com
Department of Engineering, ME AI & DS, AISSMS College of Engineering, Pune, India, ajkadam@aissmscoe.com

A B S T R A C T

This paper introduces a deep learning solution for handwritten digit recognition through the use of Convolutional Neural Networks (CNNs). CNNs are highly effective for image classification tasks due to their ability to automatically learn spatial features from input images. The proposed system is implemented using Python and leverages open-source libraries including TensorFlow and Keras. The model is trained and validated using the MNIST dataset, a standard benchmark in image classification research. Performance evaluation demonstrates high accuracy, exceeding 98%, indicating the model's robustness. The study also presents visual analyses such as confusion matrices and prediction graphs, which provide additional insight into the model's behavior and prediction quality.

Keywords: Convolutional Neural Network, Deep Learning, Image Classification, Handwritten Digits, MNIST, TensorFlow, Keras.

1. Introduction

As image data becomes more prevalent, automated recognition systems have gained importance in real-world applications. Handwritten digit classification is a classical problem in computer vision and pattern recognition, serving as a foundation for more complex image tasks. CNNs, inspired by the visual cortex, have emerged as a powerful tool for handling such challenges due to their ability to capture hierarchical patterns in images. This work focuses on implementing a CNN-based system to classify handwritten digits using the MNIST dataset, aiming to demonstrate both technical feasibility and high prediction performance.

2. Literature Review

Early models for digit recognition relied heavily on manually crafted features and classical machine learning algorithms. However, with the emergence of deep learning, researchers have shifted toward end-to-end learning techniques like CNNs. Over the years, CNN architectures have evolved significantly—from LeNet-5 to advanced networks like ResNet and DenseNet. These developments have significantly improved performance on various computer vision benchmarks. Recent works have also emphasized the ease of development and experimentation using modern deep learning frameworks such as TensorFlow and PyTorch, enabling rapid prototyping of models for tasks like digit classification.

3. Methodology

3.1 Dataset Description

The MNIST dataset contains 70,000 grayscale images of handwritten digits ranging from 0 to 9. Each image has a resolution of 28x28 pixels. The dataset is divided into 60,000 training images and 10,000 test images.

3.2 CNN Architecture Design

The extraction phase involves retrieving raw data from various sources. In this study, we used the Deutsche Bank Customer Churn Dataset (2024) stored in a CSV file. The extraction process includes:

The architecture used in this study includes:

- Input Layer: Accepts 28x28x1 input images.

- Conv Layer 1: 32 filters, kernel size 3x3, ReLU activation.
- Pooling Layer 1: MaxPooling with 2x2 filter.
- Conv Layer 2: 64 filters, kernel size 3x3, ReLU activation.
- Pooling Layer 2: MaxPooling with 2x2 filter.
- Flatten Layer: Converts feature maps into a single vector.
- Dense Layer: 128 neurons with ReLU activation.
- Output Layer: 10 neurons with Softmax activation

3.3 Model Training Setup

- Optimizer: Adam
- Loss Function: Categorical Crossentropy
- Batch Size: 64
- Epochs: 10
- Framework: TensorFlow 2.x with Keras API

3.4 Evaluation Metrics

Model performance was assessed using:

- Overall Accuracy
- Confusion Matrix
- Training vs Validation Accuracy and Loss plots
- Misclassified Sample Visualization

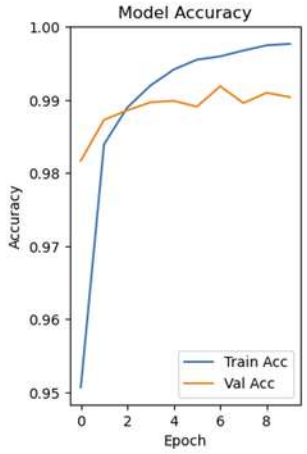
4. Results and Analysis

The final model achieved a classification accuracy of 98.4% on the test set. Visualizations were created to evaluate the model:

- **Confusion Matrix**: Shows correct and incorrect predictions per class.
- **Accuracy and Loss Graphs: Monitor model learning trends over epochs.**
- **Misclassification Examples**: Provide insights into edge cases and limitations.

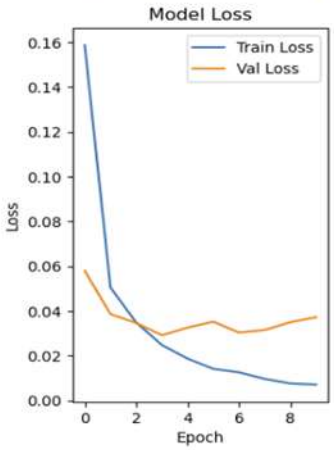
[11]:
```
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

[11]:
```
<matplotlib.legend.Legend at 0x1ed81599220>
```



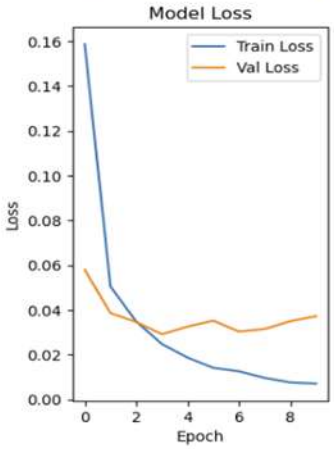
[12]:
```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

[12]:
```
<matplotlib.legend.Legend at 0x1ed819782f0>
```



[12]:
```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

[12]:
```
<matplotlib.legend.Legend at 0x1ed819782f0>
```



[34]:
```
print("Classification Report:")
print(classification_report(y_test, y_pred_classes))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       980
           1       0.99      1.00      1.00      1135
           2       0.99      0.99      0.99      1032
           3       0.98      1.00      0.99      1010
           4       0.98      1.00      0.99       982
           5       0.99      0.98      0.99       892
           6       1.00      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       1.00      0.98      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

[37]:
```
misclassified_indices = np.where(y_pred_classes != y_test)[0]
plt.figure(figsize=(10, 10))
for i, index in enumerate(misclassified_indices[:9]):
    plt.subplot(3, 3, i + 1)
    plt.imshow(x_test[index].reshape(28, 28), cmap='gray')
    plt.title(f"True: {y_test[index]}, Pred: {y_pred_classes[index]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

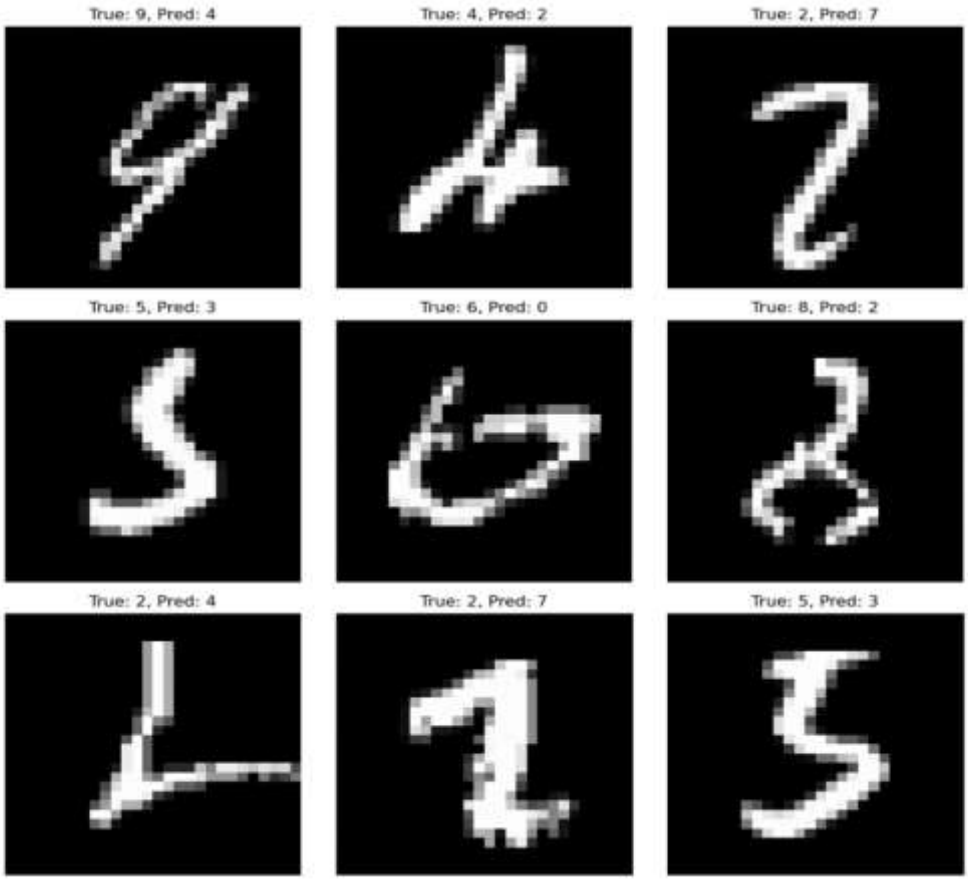


Fig.– Confusion matrix and learning curves

5. Conclusion and Future Directions

This research highlights the effectiveness of CNNs in automating the classification of handwritten digits. By training on the MNIST dataset, the model achieved high accuracy with minimal preprocessing, validating the strength of deep learning models in feature extraction and classification tasks.

Future enhancements may include:

- Experimentation with more complex networks like ResNet or MobileNet
- Deployment of the model through web APIs for real-time use
- Training on diverse handwriting datasets to improve generalization
- Integration with Optical Character Recognition (OCR) systems

Acknowledgements

We are grateful to the faculty and mentors of the AI & DS department at AISSMS College of Engineering for their guidance. We also acknowledge the developers of TensorFlow and Keras for providing accessible tools for deep learning development.

Appendix A. Tools and Libraries used

- Python 3.10
- TensorFlow/Keras – Neural network modeling
- NumPy & Pandas – Data handling.
- Matplotlib & Seaborn – Visualization

Appendix B. Sample Python Code for Data Transformation

```
import tensorflow as tf
```

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
# Load data and preprocess
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
# Build CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
# Compile and train
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10, batch_size=64, validation_data=(x_test, y_test))
```

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] L. Deng, “The MNIST database of handwritten digit images for machine learning research,” IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, 2012.

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[4] F. Chollet, Deep Learning with Python, Manning Publications, 2018.

[5] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” arXiv preprint arXiv:1803.08375, 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, vol. 25, 2012.

[7] TensorFlow Documentation. https://www.tensorflow.org/tutorials/images/cnn

[8] Keras Documentation. https://keras.io/examples/vision/mnist_convnet/

[9] W. McKinney, Python for Data Analysis, O’Reilly Media, 2017.

[10] Hunter, J. D., “Matplotlib: A 2D Graphics Environment,” Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.