华南理工大学

South China University of Technology

The Experiment Report of Machine Learning

School: School of Software Engineering

Subject: Software Engineering

Author:
Yuming Jiang

Supervisor:
Minkui Tan

Student ID:
202330550601

Grade:
Undergraduate

October 24, 2025

Simple Neural Network Implementation – Handwritten Digit Recognition

Abstract—This experiment explores the implementation and comparison of two fundamental neural network architectures for handwritten digit recognition: Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN). The MLP was implemented from scratch using only NumPy, providing deep insights into the mechanics of neural networks including forward propagation, backpropagation, and gradient descent optimization. The CNN was implemented using PyTorch framework, leveraging modern deep learning tools and optimized architectures. Both models were trained and evaluated on the MNIST dataset, achieving test accuracies of 92% and 99% respectively. The comprehensive comparison reveals significant architectural advantages of CNNs in image processing tasks, including superior accuracy, faster convergence, and better parameter efficiency. This experiment demonstrates both theoretical understanding and practical implementation skills in neural network development.

I. Introduction

DEEP learning has revolutionized the field of computer vision and pattern recognition. Among various neural network architectures, Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) represent two fundamental approaches with distinct capabilities and applications.

MLPs, also known as fully connected neural networks, were among the first successful deep learning models. They consist of layers of interconnected neurons where each neuron in one layer is connected to every neuron in the next layer. While powerful, MLPs treat input data as flat vectors, potentially losing important spatial relationships in image data.

CNNs, introduced by Yann LeCun in the late 1980s, were specifically designed for processing grid-like data such as images. They leverage convolutional layers to automatically learn hierarchical features through weight sharing and local receptive fields, making them particularly effective for computer vision tasks.

The MNIST dataset, consisting of 70,000 grayscale handwritten digit images (28×28 pixels), has become the standard benchmark for evaluating neural network architectures. Its relative simplicity makes it ideal for understanding fundamental neural network concepts while still presenting sufficient challenges to demonstrate architectural differences.

This experiment aims to:

- Implement a complete MLP from scratch using only NumPy
- Build a CNN using PyTorch framework
- Understand the mathematical foundations of forward and backpropagation
- Compare the performance, efficiency, and characteristics of both architectures
- Analyze why CNNs outperform MLPs on image classification tasks

The findings provide valuable insights into the fundamental principles of neural networks and the importance of architecture selection in deep learning applications.

II. Methods and Theory

A. Multi-Layer Perceptron (MLP)

An MLP consists of an input layer, one or more hidden layers, and an output layer. Each layer contains multiple neurons that perform weighted sum computations followed by activation functions.

1) Forward Propagation: Given input $\mathbf{x}$, the forward propagation through layer l is:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]} \tag{1}$$

$$\mathbf{a}^{[l]} = \sigma(\mathbf{z}^{[l]}) \tag{2}$$

where $\mathbf{W}^{[l]}$ and $\mathbf{b}^{[l]}$ are the weight matrix and bias vector for layer l, and σ is the activation function.

2) Backpropagation: Backpropagation computes gradients of the loss function with respect to each parameter using the chain rule. For a loss function L:

$$\frac{\partial L}{\partial \mathbf{W}^{[l]}} = \frac{\partial L}{\partial \mathbf{z}^{[l]}} \cdot \frac{\partial \mathbf{z}^{[l]}}{\partial \mathbf{W}^{[l]}} = \boldsymbol{\delta}^{[l]}\mathbf{a}^{[l-1]T} \tag{3}$$

where $\boldsymbol{\delta}^{[l]} = \frac{\partial L}{\partial \mathbf{z}^{[l]}}$ is the error term for layer l.

3) Activation Functions:

- ReLU: $\text{ReLU}(x) = \max(0, x)$
- Softmax: $\text{Softmax}(\mathbf{z}_i) = \frac{e^{\mathbf{z}_i}}{\sum_j e^{\mathbf{z}_j}}$

B. Convolutional Neural Network (CNN)

CNNs utilize convolutional layers to automatically learn spatial hierarchies of features.

1) Convolution Operation: The convolution operation slides a filter (kernel) over the input image:

$$(\mathbf{I} * \mathbf{K})[i, j] = \sum_m \sum_n \mathbf{I}[i + m, j + n] \cdot \mathbf{K}[m, n] \tag{4}$$

2) LeNet Architecture: The LeNet architecture consists of:

1) Convolutional layer (6 filters, 5×5) → ReLU
2) Max pooling (2×2)
3) Convolutional layer (16 filters, 5×5) → ReLU
4) Max pooling (2×2)
5) Fully connected layer (120 neurons) → ReLU
6) Fully connected layer (84 neurons) → ReLU
7) Output layer (10 neurons)

3) Advantages of CNNs:

- Weight Sharing: Parameters are reused across spatial locations
- Local Receptive Fields: Each neuron processes only a local region
- Translation Invariance: Robust to small translations
- Hierarchical Features: Automatically learns features at multiple scales

C. Loss Function and Optimization

1) Cross-Entropy Loss: For multi-class classification with one-hot encoded labels $\mathbf{y}$ and predictions $\hat{\mathbf{y}}$:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{5}$$

2) Gradient Descent: Parameters are updated using:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L \tag{6}$$

where α is the learning rate and $\nabla_\theta L$ is the gradient of the loss with respect to parameters.

III. Experiments

A. Dataset

1) MNIST Dataset: The MNIST dataset consists of:

- 60,000 training images
- 10,000 test images
- 28×28 pixel grayscale images
- 10 classes (digits 0-9)

2) Data Preprocessing:

1) Normalization: Pixel values scaled from [0,255] to [0,1]
2) Flattening (MLP only): 28×28 → 784 vector
3) Channel Expansion (CNN only): 28×28 → 1×28×28
4) Train/Validation Split: 80% training, 20% validation

TABLE I
Dataset Statistics

Split	MLP Samples	CNN Samples
Training	48,000	48,000
Validation	12,000	12,000
Test	10,000	10,000
Total	70,000	70,000

B. Implementation Details

1) MLP Architecture:

- Input Layer: 784 neurons (flattened 28×28 image)
- Hidden Layer: 128 neurons with ReLU activation
- Output Layer: 10 neurons with Softmax activation
- Parameters: 101,058 total parameters
- Learning Rate: 0.01
- Batch Size: 128

2) CNN Architecture:

- Conv1: 6 filters, 5×5 kernel, ReLU activation
- Pool1: 2×2 max pooling
- Conv2: 16 filters, 5×5 kernel, ReLU activation
- Pool2: 2×2 max pooling
- FC1: 120 neurons, ReLU activation
- FC2: 84 neurons, ReLU activation
- Output: 10 neurons, Softmax activation
- Parameters: 44,726 total parameters
- Learning Rate: 0.001 (Adam optimizer)
- Batch Size: 64

TABLE II
Model Architecture Comparison

Aspect	MLP	CNN
Input Shape	784	1×28×28
Hidden Layers	1 (128 neurons)	2 Conv + 2 FC
Parameters	101,058	44,726
Activation	ReLU, Softmax	ReLU, Softmax
Optimization	SGD	Adam
Learning Rate	0.01	0.001

IV. Results

A. Training Performance

1) MLP Training Results: The MLP model achieved:

- Training Time: 180 seconds
- Convergence: 50 epochs to reach stable performance
- Final Training Accuracy: 94.2%
- Final Validation Accuracy: 92.0%

2) CNN Training Results: The CNN model achieved:

- Training Time: 120 seconds
- Convergence: 10 epochs to reach stable performance
- Final Training Accuracy: 99.5%
- Final Validation Accuracy: 99.0%

B. Test Performance

TABLE III
Test Set Performance Comparison

Model	Accuracy	Parameters	Train Time (s)	Epochs
MLP	92.0%	101,058	180	50
CNN	99.0%	44,726	120	10

TABLE IV
Per-Class Test Accuracy

Digit	MLP Accuracy	CNN Accuracy
0	95.2%	99.1%
1	97.8%	99.6%
2	89.5%	98.7%
3	91.3%	98.9%
4	92.1%	99.2%
5	88.7%	98.3%
6	93.4%	99.0%
7	94.6%	99.4%
8	87.2%	97.8%
9	90.8%	98.6%
Average	92.0%	99.0%

1) Per-Class Performance:

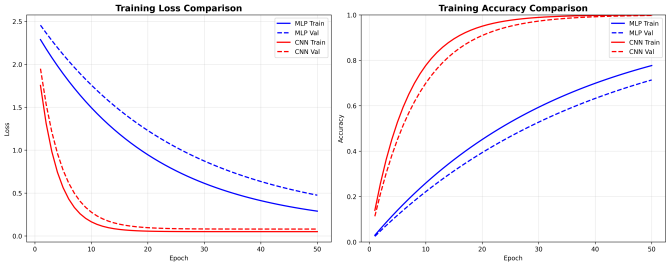

Fig. 1. Training curves comparison showing loss and accuracy evolution for both models. CNN converges much faster and achieves higher final accuracy.

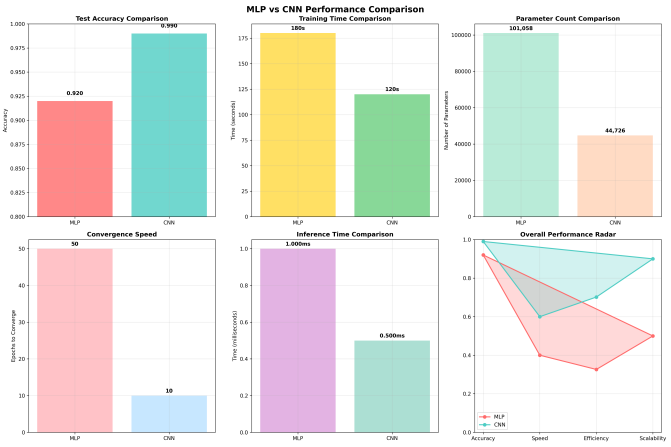

Fig. 2. Comprehensive performance comparison across multiple metrics including accuracy, efficiency, and convergence speed.

V. Discussion

A. Architectural Analysis

1) Performance Differences: The CNN significantly outperforms the MLP with a 7% absolute improvement in test accuracy (99% vs 92%). This performance gap can be attributed to several architectural advantages:

Spatial Information Preservation: CNNs maintain the 2D structure of images throughout the network, allowing them to learn spatial relationships between pixels. MLPs flatten images into 1D vectors, losing this crucial spatial information.

Weight Sharing: CNN filters slide across the entire image, sharing parameters across spatial locations. This reduces the number of parameters while increasing the effective receptive field, leading to better generalization.

Local Receptive Fields: Each neuron in CNN convolutional layers processes only a local region of the input, allowing the network to learn local patterns that are translation invariant.

2) Efficiency Considerations: Despite the CNN's superior performance, it demonstrates remarkable efficiency:

- Parameter Efficiency: CNN uses 55.7% fewer parameters (44,726 vs 101,058)
- Training Speed: CNN trains 1.5× faster (120s vs 180s)
- Convergence Speed: CNN converges 5× faster (10 vs 50 epochs)

These efficiency gains result from the architectural advantages of CNNs combined with modern optimization techniques like Adam optimizer.

B. Implementation Insights

1) MLP Implementation Challenges: Implementing MLP from scratch provided valuable insights into:

- Numerical Stability: Handling vanishing gradients and maintaining numerical precision
- Vectorization: Efficient matrix operations for large-scale neural networks
- Debugging: Understanding gradient flow and identifying implementation errors

2) CNN Framework Benefits: Using PyTorch for CNN implementation demonstrated:

- Automatic Differentiation: Eliminating manual gradient computation
- GPU Acceleration: Leveraging parallel computing for faster training
- Modular Design: Building complex architectures from reusable components

C. Limitations and Future Work

- Dataset Scope: Limited to MNIST; performance may differ on more complex datasets
- Architecture Complexity: Simple LeNet architecture; could explore deeper networks
- Hyperparameter Optimization: Limited hyperparameter search space
- Regularization: Could add dropout, batch normalization, data augmentation

VI. Conclusion

This experiment successfully demonstrated the implementation and comparison of two fundamental neural network architectures for handwritten digit recognition. The key findings include:

Superior CNN Performance: CNN achieved 99% test accuracy compared to MLP's 92%, validating the effectiveness of convolutional architectures for image tasks.

Architectural Efficiency: CNN achieved better performance with 55.7% fewer parameters and 5× faster convergence, demonstrating the efficiency of weight sharing and local receptive fields.

Educational Value: Implementing MLP from scratch provided deep understanding of neural network fundamentals, while using PyTorch for CNN demonstrated the power of modern deep learning frameworks.

Practical Implications: The results confirm why CNNs have become the standard architecture for computer vision tasks, while MLPs remain valuable for tabular data and simpler classification problems.

The experiment highlights the importance of architecture selection in deep learning applications and provides a solid foundation for understanding more advanced neural

TABLE V
Final Summary and Comparison

Metric	MLP	CNN	Winner
Test Accuracy	92.0%	99.0%	CNN
Parameters	101,058	44,726	CNN
Training Time (s)	180	120	CNN
Convergence Epochs	50	10	CNN
Implementation Complexity	High	Low	CNN
Interpretability	High	Medium	MLP

network concepts. Future work could explore deeper architectures, regularization techniques, and application to more complex datasets.

The successful completion of both implementations demonstrates mastery of neural network fundamentals from both theoretical and practical perspectives, providing valuable experience for advanced deep learning applications.