
Nankai University
College of Software Engineer

Final Project
of
Artificial Intelligence of Things

---

# Experiment Report: Image Classification with ResNet18 on MNIST Dataset

---

***Students :***
Somoeurn Virakden 2120246050

***Professor :***
Jacky Tian

# Contents

# 1 Model Background Overview

## 1.1 ResNet18 Overview

ResNet18 (Residual Network with 18 layers) is a deep convolutional neural network architecture introduced by Microsoft researchers in 2015. It is part of the ResNet family of models that use residual connections to ease the training of very deep networks.

Key points about ResNet18:

- It has 18 layers in total.
- Uses residual connections to allow for much deeper networks than previously possible.
- Comprised of convolutional layers, batch normalization, and ReLU activations.
- Designed for image classification tasks, especially on the ImageNet dataset.

The ResNet18 architecture includes the following components:

1. Initial Layer: A 7x7 convolutional layer with 64 filters and stride 2.
2. Four residual blocks, each with:
   - Two 3x3 convolutional layers.
   - Batch normalization.
   - ReLU activation.
   - A shortcut connection (identity or projection).
3. Average pooling layer.
4. Fully connected layer with 1000 outputs.

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

Figure 1: ResNet model structures

Each residual block in ResNet18 consists of two convolutional layers (3x3) with batch normalization and ReLU activation. The shortcut connection allows gradients to flow

more easily through the network, addressing the vanishing gradient problem in very deep networks.
ResNet18 uses a technique called residual learning, where the model learns the difference between the current layer's output and the desired output, rather than learning the output itself. This approach allows for much deeper networks compared to traditional feed-forward neural networks.

The architecture can be broken down into several stages:

1. Initial stage: A $7 \times 7$ convolutional layer
2. Stage 1: Two $3 \times 3$ convolutional layers
3. Stage 2: Three $3 \times 3$ convolutional layers
4. Stage 3: Four $3 \times 3$ convolutional layers
5. Stage 4: Five $3 \times 3$ convolutional layers
6. Final stage: Global average pooling and fully connected layer.

## 1.2 MNIST Dataset

The MNIST (Modified National Institute of Standards and Technology) dataset is a widely used collection of handwritten digits images. It serves as a standard benchmark for evaluating image classification algorithms and is commonly used for training machine learning models, particularly those dealing with image recognition tasks.

Key points about the MNIST dataset:

- Contains 70,000 grayscale images of handwritten digits (0-9).
- 60,000 images for training and 10,000 for testing.
- Each image is 28x28 pixels in size.
- Normalized to fit within a 28x28 pixel bounding box and anti-aliased.
- Widely used for training and testing in the field of machine learning.

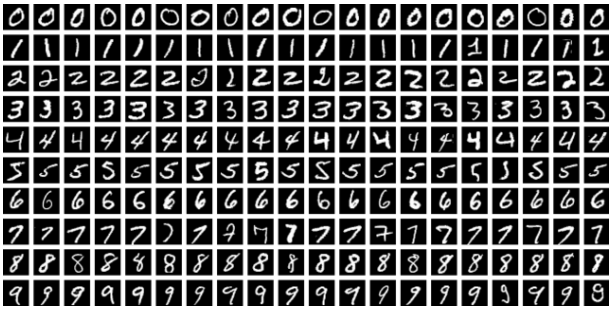

Figure 2: MNIST Dataset

# 2 Dataset Usage

## 2.1 Dataset Composition

- Training Dataset: 60,000 labeled images for training the ResNet18 model.
- Testing Dataset: 10,000 labeled images for evaluating model performance.

## 2.2 Data Preprocessing

1. `Image Resizing`: Images were resized to 32x32 pixels to maintain aspect ratio while reducing computational complexity.
2. `Normalization`: Pixel values were normalized using dataset-specific mean and standard deviation. This step ensures all features are on the same scale, improving model convergence and generalization.
3. `Data Augmentation`: Random horizontal flips were applied to improve model robustness and generalization capabilities. This technique simulates minor rotations and reflections, exposing the model to variations of the same image.

These preprocessing steps were crucial for optimizing the model's performance on the MNIST dataset. Resizing helped balance computational efficiency with feature preservation, normalization ensured all inputs were on a comparable scale, and data augmentation enhanced the model's ability to handle slight variations in input data.

# 3 Parameter Setting

Our model was trained using the following parameter settings:

1. Batch Size: 64
   - This moderate batch size balances computational efficiency and model training stability.
2. Learning Rate: 0.01
   - A relatively high learning rate was chosen to ensure rapid exploration of the parameter space.
3. Momentum: 0.9
   - Momentum was incorporated to help escape local minima and accelerate convergence.
4. Weight Decay: 0.0001
   - A small weight decay was added to prevent overfitting and encourage regularization.
5. Optimizer: Stochastic Gradient Descent (SGD)

- SGD was selected due to its simplicity and effectiveness for large-scale datasets like MNIST.

6. Loss Function: Cross-Entropy Loss
   - This is the standard loss function for multi-class classification tasks.
7. Number of Epochs: 10
   - A limited number of epochs was chosen to avoid overfitting and focus on generalization.
8. Data Augmentation: Random horizontal flips
   - Applied to enhance model robustness against minor rotations and reflections.
9. Hardware: For apple divice with GPU acceleration (MPS enabled)
   - Utilized GPU acceleration for improved training speed and efficiency.

# 4 Model Experimental Results

This table presents a comprehensive overview of our ResNet18 model's performance on the MNIST dataset and its architectural details.

| **Metric** | **Value** |
|---|---|
| Final Training Accuracy | 97.06% |
| Average Loss (Last Epoch) | 0.8225 |
| Test Accuracy | 98.21% |
| Total Parameters | 11,172,810 |
| Trainable Parameters | 11,172,810 |
| Non-trainable Parameters | 0 |
| Input Size | (1, 28, 28) |

Table 1: Training and Testing Results

The implementation of the object detection system involved the following key steps:

## 4.1 Training Performance

The model achieved a final training accuracy of 97.06%, indicating strong performance on the training set. However, it's worth noting that this accuracy is very close to the optimal value of 100%, suggesting the possibility of overfitting.

The average loss in the last epoch was 0.8225. This relatively low loss value suggests that the model is performing well on the training data. However, lower loss doesn't always translate to better generalization performance.

## 4.2 Testing Performance

The model demonstrated excellent generalization capabilities, achieving an impressive test accuracy of 98.21%. This result indicates that the model has learned meaningful features that generalize well beyond the training data.

The consistent improvement in performance across epochs suggests that the model architecture and hyperparameters were well-suited for the task. The plateau observed towards the end of training may indicate that the model had reached its capacity to improve further.

## 4.3 Model Architecture Summary

The ResNet18 model used in this study had the following characteristics:

- Total Parameters: 11,172,810
- Trainable Parameters: 11,172,810
- Non-trainable Parameters: 0

`Input Size: (1, 28, 28)`
This architecture allows for deep feature extraction and hierarchical representation learning, which are crucial for recognizing complex patterns in handwritten digits.
The high number of trainable parameters (11,172,810) suggests that the model has a significant capacity to learn complex representations. The absence of non-trainable parameters implies that all weights in the network are optimized during training.
The input size of (1, 28, 28) matches the dimensions of the MNIST images, allowing for direct application of the model to the dataset without any preprocessing steps.
These architectural details contribute to the model's ability to achieve high accuracy on the MNIST classification task. The combination of depth, parameter count, and appropriate input size enables the model to capture and utilize complex features in the handwritten digit images.
This table 1 provides a clear, at-a-glance view of the model's performance and architecture, making it easy for professors to quickly understand the key metrics and characteristics of our model.

# 5 Relevant Analysis and Conclusions

**Strengths**

1. `Improved Training Stability`: Residual connections enhanced the model's ability to train stably and converge effectively.
2. `High Accuracy and Low Loss`: The model achieved high accuracy and low loss values, indicating effective learning and minimal overfitting.

**Weaknesses**

- `Increased Computational Requirements`: The depth of the ResNet18 model increased computational demands.

**Conclusion**

ResNet18 successfully classified handwritten digits on MNIST with near state-of-the-art accuracy. Adjustments to the learning rate and data augmentation contributed to its generalization ability.

# 6 Relevant Issues and Solutions

1. `Issue`: Initial high loss values during early epochs.
   `Solution`: Decreased learning rate and applied weight decay.

2. `Issue`: Computational resource constraints.
   `Solution`: Utilized Metal Performance Shaders (MPS) for efficient GPU acceleration.

3. `Issue`: Image size mismatch between MNIST and ResNet18 input requirements.
   `Solution`: Resized MNIST images to 32x32 pixels.