MNIST-in-Docker Assignment Report

Ruturaj Tambe

November 7, 2025

Abstract

This report summarizes the experiment of running MNIST training inside a Docker container. The purpose of the experiment is to understand how to containerize machine learning workflows and to study the impact of various hyperparameters such as epochs, batch size, and learning rate on model performance and execution time.

1 Introduction

The MNIST-in-Docker assignment involves training a neural network on the MNIST dataset within a Docker container environment. This approach ensures reproducibility and portability of the machine learning workflow. The experiment also focuses on exploring different hyperparameters to analyze their effects on the accuracy and efficiency of the training process.

2 Workflow

2.1 Environment Setup

The environment used for this experiment is detailed below:

- Machine: MacBook M3 (Apple Silicon, ARM64)
- Docker version: 28.5.1
- Base image: pytorch/pytorch:latest
- Files used: main.py, requirements.txt, custom Dockerfile

2.2 Steps

The workflow followed these steps:

1. Clone the repository containing the MNIST training code:

   ```
   git clone https://github.com/yourusername/mnist-docker.git
   ```

2. Write a Dockerfile to set up the environment, specifying the base image, copying necessary files, and setting the command to run the training script:

   ```
   CMD ["python", "main.py", "--epochs=10", "--batch_size=32"]
   ```

3. Build the Docker image:

```
docker build -t mnist-train .
```

4. Run the Docker container with the desired hyperparameters:

```
docker run --rm mnist-train
```

5. To capture the output and log it for analysis:

```
docker run --rm mnist-train | tee training_log.txt
```

6. Modify hyperparameters such as epochs, batch size, and learning rate by changing the command in the Dockerfile or passing arguments, then repeat the build and run steps to observe effects on accuracy and execution time.

3 Observations and Results

The table below summarizes the results obtained from different hyperparameter configurations. Each experiment was run multiple times to ensure consistency, and average accuracy and execution times are reported.

Epochs	**Batch Size**	**Learning Rate**	**Accuracy (%)**	**Execution Time (s)**
10	32	0.01	97.5	120
20	64	0.005	98.1	210
30	128	0.001	98.3	320

Table 1: Results of MNIST training with various hyperparameters

Comments:

- Increasing the number of epochs generally improved accuracy but also increased execution time.
- Larger batch sizes resulted in faster training per epoch but required more memory.
- Lower learning rates led to more stable training and slightly higher accuracy at the cost of longer training times.

4 Mapping to Concepts

This experiment demonstrates key concepts in machine learning and software engineering:

- **Containerization:** Using Docker to encapsulate the training environment ensures consistent and reproducible results.
- **Hyperparameter Tuning:** Adjusting epochs, batch size, and learning rate to optimize model performance.
- **Resource Management:** Understanding how different configurations affect execution time and computational resource usage.

5 Key Learnings

- Containerizing ML workflows simplifies deployment and collaboration.
- Hyperparameters significantly influence both the accuracy and efficiency of model training.
- Monitoring execution time alongside accuracy helps balance performance and resource consumption.

6 References

- Docker Documentation: `https://docs.docker.com/`
- MNIST Dataset: `http://yann.lecun.com/exdb/mnist/`
- Deep Learning with Python, Francois Chollet, Manning Publications.