Multi-Task Learning on MNIST: Laboratory Report

Digit Classification + Parity + Threshold Prediction

Course: [Insert Course Name]
Student: [Insert Your Name & Student ID]
Date: November 25, 2025

Table of Contents

---

1. Introduction

The purpose of this laboratory task is to extend the classical MNIST digit-classification problem into a multi-task learning setup.

The provided code from the lecture performs three simultaneous predictions:

- Digit classification (0–9)
- Parity classification (even = 0, odd = 1)
- Threshold classification (<5 = 0, ≥5 = 1)

Goals of this lab:

- Run the lecturer's baseline implementation
- Modify and improve the model/training process
- Compare performance before and after modifications

- Conclude whether improvements helped and explain why

MNIST is a relatively simple dataset; therefore, the task also demonstrates the limitations of improving an already high-performing model.

---

2. Baseline Results

The baseline network is a 3-layer CNN with three separate output heads:

- fc2_digit → 10 classes
- fc2_parity → 2 classes
- fc2_threshold → 2 classes

Trained for **10 epochs** using Adam optimizer and StepLR scheduler.

Baseline Test Results

Metric	**Value**
Test Loss	0.0518
Digit Accuracy	**99.24%**
Parity Accuracy	**99.56%**
Threshold Accuracy	**99.46%**

These results show that the model already achieves near-ceiling accuracy on MNIST.

[IMAGE PLACEHOLDER 1 – Baseline Training & Validation Curves]
Four subplots (clearly labeled, 10 epochs):

- Top-left: Total Loss (train/val)
- Top-right: Digit Accuracy
- Bottom-left: Parity Accuracy
- Bottom-right: Threshold Accuracy

---

3. Modifications Attempted

#	Modification	Purpose	Code Comment Example
3.1	Increase epochs (10 → 20)	Allow deeper convergence	num_epochs = 20 # increased for deeper convergence
3.2	Add Batch Normalization after conv layers	Stabilize and accelerate training	self.bn1 = nn.BatchNorm2d(32) etc.
3.3	Add Dropout (p=0.3) after dense layers	Prevent overfitting	self.dropout = nn.Dropout(0.3)
3.4	Replace Flatten → Global Average Pooling	Reduce parameters, improve generalization	self.gap = nn.AdaptiveAvgPool2d(1)
3.5	Custom multi-task loss weights	Prioritize the main digit task	loss = loss_digit * 2 + loss_parity + loss_threshold

4. Observations

4.1 Increasing Epochs to 20

- Loss started oscillating instead of decreasing further
- Accuracy remained virtually unchanged
- MNIST is too small → longer training introduces noise rather than gains

4.2 BatchNorm + Dropout + Global Average Pooling

- Training curves became smoother and more stable
- Best final test accuracies (combined run):
    - Digit: ~99.23%

    - Parity: ~99.50%
    - Threshold: ~99.46%
- Essentially identical to baseline

4.3 Multi-Task Loss Weighting

- 20 epochs: slight degradation
- 10 epochs: nearly identical to baseline
- Parity & threshold tasks are too easy → weighting has negligible impact

4.4 Summary

The dataset is simple and the baseline model is already at saturation. Architectural improvements add stability but cannot push accuracy significantly higher on MNIST.

[IMAGE PLACEHOLDER 2 – Modified Model Training & Validation Curves]
Same four-subplot layout as baseline – best combined run (BatchNorm + Dropout + GAP + 20 epochs)

---

5. Conclusion

In this laboratory assignment, a multi-task CNN was trained to simultaneously predict digit class, parity, and threshold on the MNIST dataset.

The baseline already achieved **≈99.2–99.5%** accuracy across all tasks. The attempted improvements (Batch Normalization, Dropout, Global Average Pooling, extended training, and loss weighting) resulted in **minimal to no accuracy gains**.

Key Takeaways

- The lecturer’s architecture is more than sufficient for MNIST
- Regularization techniques improved training stability but not final performance
- Parity and threshold tasks are trivial and do not require deeper models

- Further gains would need data augmentation, label noise, or harder datasets
- The experiment perfectly illustrates **diminishing returns** on a saturated benchmark

The lab successfully demonstrates both the mechanics of multi-task learning and the practical limits of optimization on simple datasets such as MNIST.

End of Report