MiA robotics

AI TEAM TRAINING'26

TASK 4

Task 4: Implementing a Neural Network

The Scenario:

Welcome to the team. Our rivals, "Team Eigen," have deployed a new trackside camera system, but it's a disaster. Their model is unreliable, constantly misidentifying car numbers from grainy footage a '7' for a '1', a '3' for an '8'. Their flawed data is wrecking their race strategy, and it's our chance to seize the lead.

We need a system that's not just better, but flawless. A system that will give our strategists a crystal-clear, real-time picture of the entire race, allowing them to make the split-second, race-winning decisions we're famous for.

Your Mission:

Your mission is to build our next-generation vision system **from the ground up**. This isn't just about using off-the-shelf parts; it's about understanding the core mechanics to ensure our system is robust, accurate, and unbeatable. The project is divided into several critical phases:

1. **Phase 1: Engineer the Core Engine.** Before we can build a car, we must first engineer its heart. You will start by building a custom automatic differentiation engine (the Value object) that will power our network's ability to learn.
2. **Phase 2: Assemble the Chassis.** With the engine ready, you will construct the neural network's architecture—the Neuron, Layer, and MLP classes that form the framework of our vision model.
3. **Phase 3: The Grand Prix.** This is the main event. You will deploy your custom-built system to tackle the ultimate challenge: correctly identifying handwritten-style digits from trackside footage (represented by the MNIST dataset) and proving its superiority over Team Vortex.
4. **Phase 4: Master the Professional Toolkit.** After proving your mastery by building a system from scratch, your final task is to show you can handle professional-grade tools by re-implementing the solution in **PyTorch**, demonstrating you're ready for any challenge.

Technical Requirements:

You are required to complete all TODO sections in this notebook, which will guide you through:

- Building a Value object with a complete backpropagation mechanism.
- Constructing the Neuron, Layer, and MLP classes from scratch.
- Implementing a training loop to train your custom model on the MNIST dataset.
- Rebuilding and training the model in PyTorch to compare against your from-scratch implementation

Bonus Objectives:

Any further improvements you implement will be considered for bonus recognition. These optional challenges are designed to introduce you to more advanced techniques used in modern deep learning:

- **Cross-Entropy Loss:** A more suitable loss function for classification tasks.
- **ReLU Activation:** A popular and efficient activation function that can help with training stability.
- **Weight Initialization:** Using smarter strategies to set initial parameter values, which can significantly improve training convergence.
- **Dropout:** A regularization technique to prevent model overfitting by forcing the network to learn more robust features.
- **Momentum:** An optimization technique that helps accelerate learning and navigate complex loss landscapes more effectively.