18-661 Introduction to Machine Learning

Pytorch

Arian Raje

Spring 2025

ECE – Carnegie Mellon University

Outline

1. A Small History Lesson
2. What is Pytorch?
3. Pytorch Examples

Disclaimer: this lecture will not appear on your final exam, though some content, in particular PyTorch, may be used on the Homework

Some History About Neural Networks

A Grossly Abreviated History of AI

4 Time Periods of Artificial Intelligence/Neural Networks -

1. Early Stages (1950s-1980s)
    - Turing Test
    - Perceptron
2. AI Winter (1980s-2000s)
    - *Perceptrons: An Introduction to Computational Geometry* by Marvin Minsky
3. Resurgence and GPU Acceleration (2000s-2010s)
    - DARPA
    - IBM Watson
    - CUDA
4. Modern AI (2010s-2020s)
    - Pytorch/Tensorflow
    - AlexNet
    - ChatGPT

A Grossly Abreviated History of AI

4 Time Periods of Artificial Intelligence/Neural Networks -

1. Early Stages (1950s-1980s)
   - Turing Test
   - Perceptron
2. AI Winter (1980s-2000s)
   - *Perceptrons: An Introduction to Computational Geometry* by Marvin Minsky
3. Resurgence and GPU Acceleration (2000s-2010s)
   - DARPA
   - IBM Watson
   - CUDA
4. Modern AI (2010s-2020s)
   - PyTorch / TensorFlow
   - AlexNet
   - ChatGPT

What is Pytorch?

What is PyTorch?

- **Open-Source Machine Learning Library** developed by Facebook's AI Research (FAIR) team
- Built primarily for **Deep Learning** tasks (vision, NLP, recommender systems, etc.)
- Provides:
  - **Automatic differentiation**
  - **GPU acceleration**
- Pythonic, easy to debug, and flexible

Automatic Differentiation (Autodiff)

- **Idea:**
    - Computes exact derivatives by "tracing" operations during the forward pass.
    - Replaces manual/finite-difference methods (less error-prone, more efficient).
- **Key Benefits:**
    - No manual math or derivative coding.
    - Works seamlessly with complex, dynamic architectures.
- **Typical Workflow (e.g., PyTorch):**
    1. Define forward pass (model + loss).
    2. Operations are recorded in a graph.
    3. Call `.backward()` for gradients w.r.t. parameters.

Example w/ Sequential Model

```python
class MySequentialModel(nn.Module):
    def __init__(self):
        super(MySequentialModel, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(10, 5),
            nn.ReLU(),
            nn.Linear(5, 2)
        )

    def forward(self, x):
        # Forward pass: simply feed x through the Sequential layers
        return self.layers(x)
```

MySequentialModel definition

```python
# 3. Perform a forward pass
output = model(x)

# 4. Define a dummy target and loss function
target = torch.randn(3, 2)  # same shape as output
criterion = nn.MSELoss()

# 5. Calculate the loss
loss = criterion(output, target)

# 6. Backward pass: compute gradients w.r.t. all parameters
loss.backward()
```

Forward pass, loss, and backward call

Key Benefit of Autodiff

You don't need to have an **explicit** backward function. Only defining a forward function suffices.

GPU Acceleration

- **Why GPUs?**
  - Optimized for parallel operations (fast matrix math)
  - Huge speedups for neural network training
- **Using GPUs in PyTorch:**
  - `device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`
  - `model.to(device)` and `x.to(device)` before training
  - Automatic differentiation runs on GPU
- **Multi-GPU / Distributed:**
  - `nn.DataParallel` or `DistributedDataParallel`
  - Scale across multiple GPUs/machines

GPU vs. CPU

	CPU	GPU
Core Count	Few (4–32)	Hundreds to Thousands
Parallelism	Limited	Massively Parallel
Memory Bandwidth	Lower	Higher
Primary Use	General-purpose	Graphics / ML Acceleration

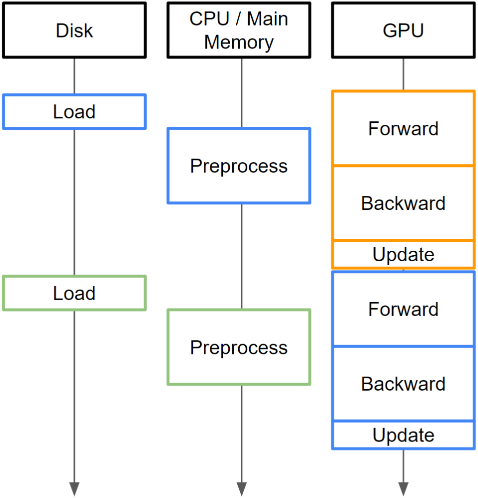
GPU vs. CPU

Question: Why wouldn't you always use GPUs?

The cores in CPUs are capable of performing very different instructions.
GPU cores have to perform the same instruction.

A Typical Deep Learning Pipeline

- The CPU is usually used for data preprocessing only.
- All parameter and gradient computations take place on the GPU
- Data loading and preprocessing should be pipelined to avoid impacting runtime.



Advantages of Pytorch-like Frameworks

2 Key Advantages -

- Automatic differentiation
- GPU acceleration

Previously, you would have to write custom CUDA code and calculate gradients by hand. With these frameworks, all of that is done for you.

Other Related Frameworks

Similar Frameworks -

- Theano $\rightarrow$ TensorFlow
- Torch $\rightarrow$ Pytorch
- JAX

Other Related Frameworks

Framework	Developer	Why Use It?
TensorFlow	Google Brain	Large ecosystem; production-ready (TensorFlow Serving / Lite)
PyTorch	Meta AI (Facebook)	Pythonic and dynamic; strong research and industry adoption
JAX	Google	High-performance autodiff via just-in-time compilation (XLA)

PyTorch Example