CSC290/420 ML Systems for Efficient AI Introduction

Sreepathi Pai

August 25, 2025

URCS

Outline

ML Systems

Efficiency

Models

Administrivia

Outline

ML Systems

Efficiency

Models

Administrivia

Machine Learning

Machine learning is an artificial intelligence technique where programs "learn" to perform tasks without being explicitly performed to do so. As used today, this mostly involves, and this is highly simplified, learning patterns from large amounts of data.

Deep Learning

- Multi-layered Neural Networks
- Demonstrated late 2000s
    - Made possible due to GPUs
- "Superhuman" performance on many tasks
    - pattern recognition
    - speech recognition
- Also used in computer vision
- Already incorporated into many products

Generative AI

- Sometimes known as "foundation" models
- Demonstrate ability to perform many tasks without having been explicitly trained to do so.
- Biggest successes:
  - Chatbots / Large Language Models
  - Image generation
- Very interesting, but unclear these are useful

ML Programs

Programs that implement machine learning algorithms such as neural networks.

Writing ML Programs

```
device = torch.accelerator.current_accelerator().type if torch.acce
print(f"Using {device} device")

class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)
print(model)
```

Excerpt from the PyTorch tutorial.

ML Program Execution modalities

- Training mode
  - Forward mode and backward mode
  - The "backward mode" program is automatically derived from a ML program
- Inference mode
  - Forward mode only

ML Program Execution

- Most ML programs run on CPUs or GPUs
  - Central Processing Unit
  - Graphics Processing Unit
- Neither of these understands "Python"
  - Python programs must be translated to the "native" language of these machines
  - Assembly language, Native code, Machine code
  - There may be multiple layers of translation: Python to CUDA to GPU machine code

ML Systems

- Any computer system that can run an ML program

Useful Categories

- Shared memory systems
  - useful shorthand: one single memory (usually, one machine)
- Distributed memory systems
  - useful shorthand: at least two machines connected by a wire (network)
  - individual machines can only read each other's memory by transferring data over the wire

Some characteristics of ML programs

- Large amounts of compute
  - defined as arithmetic operations
- Large amounts of memory
  - defined as data size
- Large amounts of communication
  - defined as data transferred between distributed systems

Outline

ML Systems

Efficiency

Models

Administrivia

Defining Efficiency

- Perform useful work with minimum resources

Algorithmic Efficiency

- $O(n)$ vs $O(n^2)$
- Limited use
    - Why?
- Still first thing to check

Resources

- Time
- Energy
- Compute resources: CPUs needed, GPUs needed
- Memory resources: RAM (size)
- Storage resources: hard disk space
- Network resources: bandwidth

Scalability

- How do resource requirements change as work increases?

Outline

ML Systems

Efficiency

Models

Administrivia

Time

Time for execution can be defined as

$$T = \frac{W \times t}{P}$$

where:

- W is "work"
- t is average time per work
- P is parallelism

Example

- Work 1: Make coffee, time 10 minutes
- Work 2: Make omlette, time 5 minutes
- If we had a stove with a single burner, time?
- If we had a stove with two burners, time?
    - Average parallelism?

ML Programs

- ML programs don't make coffee when they run
  - inside a processor
- Like all computer programs, ML programs perform:
  - arithmetic
  - memory (read / write)
  - I/O (input / output) – storage, network, etc.

Things we need

- Types of work a CPU / GPU performs
- Time per work
- Parallelism

Things we need

- Types of work a CPU / GPU performs
  - Read processor manuals
  - available through *performance counters*
- Time per work
  - Sometimes processor manuals, but other resources available
  - Often, need to write test programs
- Parallelism
  - Available through performance counters

Energy

- Simple, initial model:

$$E = \sum_{w \in W} E_w$$

- E_w is the energy for performing work w

Measuring Energy consumption

- External sensors
- Processor-internal sensors
  - e.g. Intel's RAPL
  - coarse-grained (i.e., on whole system basis)

Storage

$$\text{Data Compression Ratio} = \frac{\text{uncompressed data size}}{\text{compressed data size}}$$

- Why is compression possible?
- What is the tradeoff?

Types of Compression

- Lossless compression: recover original data exactly
- Lossy compression: cannot recover original data
    - but decompressed data resembles original data

Outline

ML Systems

Efficiency

Models

Administrivia

Organization of this course

- Part I: Computer Organization
- Part II: Real-World Efficiency
- Part III: Advanced Topics

People

- Instructor: Dr. Sreepathi Pai
  - E-mail: sree@cs.rochester.edu
  - Office: Wegmans Hall 3409
  - Office Hours: Wednesday 14:00 to 15:00 (or by appointment)
- TA:
  - Ethan Chen
  - Office Hours: Wednesday 20:00 to 21:00, Zoom (see Blackboard for link)

Places

- Class: Hylan 203
  - M,W 0900–1015
- Course Website
  - `https://cs.rochester.edu/~sree/courses/csc-290-420/fall-2025/`
- Blackboard
  - Announcements, Discussions
- Gradescope
  - Assignments, Homeworks, Grades, etc.

References

- No textbooks
- Slides, lecture notes, and assigned readings

Grading

- Homeworks: 15%
- Assignments: 60% (5 to 6)
- Mid-term: 10%
- Project: 25%
- Graduate students should expect to read a lot more, and work on harder problems.

There is no fixed grading curve. See course website for grade scale.

See course website for late submissions policy.

Academic Honesty

- Unless explicitly allowed, you may not show your code to other students
- You may discuss, brainstorm, etc. with your fellow students but all submitted work must be your own
- All help received must be acknowledged in writing when submitting your assignments and homeworks
- All external code you use must be clearly marked as such in your submission
    - Use a comment and provide URL if appropriate
- If in doubt, ask the instructor
- It is a violation of course honesty to make your assignments on GitHub (or similar sites) public

All violations of academic honesty will be dealt with strictly as per UR's Academic Honesty Policy.

Code Generation Tools

- Tools like CoPilot, ChatGPT are best not used
  - unfortunately these tools are hard to avoid
- Unless otherwise stated, you are free to use "AI" tools to do your assignments (e.g. ChatGPT, Co-pilot, etc.) provided you follow the rules below. Not following the rules below will be treated as a honesty code violation.
  - Please put a comment at the top in each file that includes AI-generated material (include auto-completions) indicating what system was used, like "# AI: ChatGPT"
  - If this is a system like ChatGPT, include the *full* transcript as a comment at the end of the source file.
  - If this was a system like Co-pilot which and you used prompts, leave the prompts in the source code.

Instructor/TA expectations about generated code

- Note that the TA and I will only help you debug AI-generated code at our discretion.

Course Goals

You will be able to:

- describe how a program executes on a modern CPU and GPU,
- model and reason about performance bottlenecks,
- demonstrate the application of various techniques to improve performance of ML/AI programs