CSC290/420 ML Systems for Efficient AI Introduction

Sreepathi Pai

August 31, 2026

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 programmed to do so. As used today, this mostly involves, learning and predicting 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
  - Code generation and analysis
  - "agents"
- Very interesting, and perhaps useful

ML Programs

Programs that implement machine learning algorithms such as neural networks.

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
- What exactly does “Large” mean?

ML Systems

Any computer system that can run an ML program

Useful Categories of ML Systems

- 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"

Another way to categorize ML Systems

- General-purpose
  - useful shorthand: can run non-ML programs too
- ML accelerators
  - useful shorthand: can run only some ML tasks (computation, memory, communication) efficiently
  - e.g., a matrix multiply accelerator
- Real systems are often a judicious mix of these two.

Outline

ML Systems

Efficiency

Models

Administrivia

Defining Efficiency

- Perform useful work with minimum resources

Algorithmic Work (and Efficiency)

- $O(n)$ vs $O(n^2)$
- Limited use
  - Why?
- Still first thing to check: is the ML program doing "useless" work?

Resources

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

Scalability

- How do resource requirements change as work increases?

Making breakfast efficient

- Time to make coffee: 8 minutes
- Time to make an omelette: 5 minutes

- What differences would you see in your kitchen vs that of a diner to prepare these two items?

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 8 minutes
- Work 2: Make omelette, 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
  - Theoretical by program design
  - Actual 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)

Memory / 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

- ML Programs
- Compute
- Memory and Storage
- Communication
- Advanced Topics

People

- Instructor: Dr. Sreepathi Pai
  - E-mail: sree@cs.rochester.edu
  - Office: Wegmans Hall 3409
  - Office Hours: Wednesday 15:30 to 16:30 (or by appointment)
- TA:
  - Hannah Davisdon
  - Office Hours: TBD

Places

- Class: Morey 321
  - M,W 1400–1515
- Course Website
  - `https://cs.rochester.edu/~sree/courses/csc-290-420/fall-2026/`
- Blackboard
  - Announcements, Discussions
- Gradescope
  - Assignments, Homeworks, Grades, etc.

References

- No required textbook
- Slides, lecture notes, and assigned readings

Grading

- Homeworks: 10%
- Assignments: 45% (4 to 5)
- Mid-terms: 2, each 10%
- Final 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.

Generative AI Usage

- This course is 100% organic
- No content (slides, assignments, exams, etc.) was produced using Generative AI
- No feedback given by me will use Generative AI to produce it
- I have made no special efforts to "GenAI-proof" my assignments

Course design philosophy

- All material in this course is intended to challenge you intellectually
    - This, like all challenging situations, involves discomfort
- Doing something for the first time involves failure and needs iteration to get it right
- Gaining knowledge, skills, and confidence in oneself is very satisfying

GenAI for Code Generation

- Tools like Claude, CoPilot, ChatGPT are best not used for things you do not know how to do
  - 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.
  - 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 and you used prompts, leave the prompts in the source code.
- Claiming ownership of GenAI generated code will be treated as a honesty code violation.

Using GenAI to Learn

- Asking questions to GenAI or using GenAI as replacement for search can be harmful when you do not know if the answer is correct
- But it can be more convenient than going to office hours!
  - Or waiting for a professor/TA to reply to your email
- GenAI synthesized text is not *an authoritative source.*
  - Even Wikipedia is better!
- Will deploy a safety mechanism – a document in which you can anonymously post GenAI replies about questions, and I can *asynchronously* reply whether it is correct or misleading.

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