Lecture 1: Course Overview. Why Scale Machine Learning?

CS4787/5777 — Principles of Large-Scale Machine Learning Systems

Term	Fall 2026	**Instructor**	Christopher De Sa
Course website	`cs.cornell.edu/courses/cs4787/`	**E-mail**	`cmd353@cornell.edu`
Schedule	MW 7:30pm - 8:45pm	**Office hours**	W 2:00pm – 3:00pm or by appointment

. . .

Why scale machine learning?

Two subquestions:

- Why machine learning? Why scalability?

The standard machine learning pipeline. Scaling up to big data presents challenges **at every stage of the pipeline**!

- Exploring data in real-time
- Selecting models and tuning hyperparameters over huge search spaces
- Training on massive datasets can take months
- Inference and deployment when latency, throughput, and memory use matter

We use techniques from three broad areas: optimization, statistics, and systems.

Why optimization?

- The core task of learning is finding a model that performs well on some metric — that's optimization
- By representing learning as an optimization problem that a computer can handle automatically, we can learn even when there are too many parameters for humans to reason about

Principle #1: Write your learning task as an optimization problem, and solve it via fast gradient-based algorithms that update the model iteratively with easy-to-compute steps using numerical linear algebra.

- Recent evolution that goes outside this principle: in-context learning.
- Examples of this principle from your previous machine learning classes?

Why statistics?

- Machine learning is statistical: statistics is in some sense the right way to handle data
- Need to deal with a lot of uncertainty
  - Especially when we scale up to dataset sizes where humans can't reason about the uncertainty present in their system manually

Principle #2: Make it easier to process a large dataset by (1) processing a small random subsample instead and/or (2) imposing a directed/acyclic structure.

- Examples of this principle from your previous machine learning classes?

Why parallel systems? Why computer architecture?

- The free lunch is long over — Moore's law is coming to an end
- We can no longer expect our performance and scalability to increase by just waiting two years for our CPUs to get two times faster
- To scale up, we need to leverage additional compute in the form of parallel and distributed systems
- At the same time, ML computations are particularly amenable to specialized hardware, such as GPUs

Principle #3: Use algorithms that fit your hardware, and use hardware that fits your algorithms.

- Examples of this principle from your previous machine learning classes?

What will we cover in this course? CS4787/5777 will explore the principles behind scalable ML.

- Fast, scalable learning with stochastic gradient descent (SGD)
- Deep learning frameworks and automatic differentiation.
- Modern deep learning models at the largest scales. Transformers. Large language models.
- Model selection and hyperparameter optimization. Scaling laws.
- Parallel and distributed training.
- Quantization, model compression, and other methods for fast inference.

. . .

Course Details and Policies

Grading for CS4787.

- Problem sets (15%)
- Programming assignments (35%)
- Prelim exam (20%)
- Final exam (30%)

Grading for CS5777.

- Problem sets (10%)
- Programming assignments (30%)
- Paper reading (10%)
- Prelim exam (20%)
- Final exam (30%)

Materials. The course is based on books, papers, and other texts in machine learning, scalable optimization, and systems. Texts will be provided ahead of time on the website on a per-lecture basis. You aren't expected to necessarily read the texts, but they will provide useful background for the material we are discussing.

Logistics.

- **Problem sets** biweekly, usually in groups of up to four (but may vary per-assignment). **Please do find a homework group of four.** Submitted on Gradescope, where they can be viewed.
- **Programming assignments (labs)** also biweekly, also usually in groups of up to four. Project report component. Submitted on Gradescope. The programming assignments are (mostly) designed to run *on Google Colab*. You can also run some of them locally.
- **Late work policy.** Each assignment has a two-day "free" grace period after the written assignment deadline in which you can turn in the assignment without penalty. Beyond this grace period, late problem sets will not be accepted, except for good reason and on a case-by-case basis, as we will have released solutions. Beyond the grace period, late programming assignments may be accepted for partial credit as specified in the assignment description.
- **Regrade policy.** Regrades for problem sets and the prelim exam may be submitted on Gradescope up to seven days after the grades are released. Regrades for programming assignments may be submitted on CMS up to seven days after the grades are released. When regrading an assignment, we reserve the right to look at the whole assignment, not just the part that you are asking for a regrade of.
- **Mastery checks.** In addition to the problem sets and programming assignments themselves, your knowledge of their content will be checked via short oral "mastery checks" with the TAs. Details of how these will be conducted are still being finalized and will be announced later in the semester.
- **Additional resources.** The course has an Ed Discussions page, where you can ask questions about the course and discuss with your peers. There is a link to this page on the course website and on the course Canvas.