

# CS265

## AI Training and Inference Systems

*From First Principles to Self-Design*

<http://daslab.seas.harvard.edu/classes/cs265/>

Prof. Stratos Idreos (stratos@seas.harvard.edu)

*After taking CS265, you will understand the internals of the systems that train, fine-tune, and serve large AI models, be able to reason about the design of any such system from first principles, and have a handle on what it means to do CS systems research.*

### WHAT IS THIS CLASS ABOUT?

The systems that train, fine-tune, and serve large models are the engines of the AI revolution, and their performance comes down to two goals: minimizing data movement across the memory hierarchy and maximizing GPU and CPU utilization. This course teaches the first principles through which system designs and implementations achieve these goals: modularity and abstractions, breaking problems into smaller pieces, parallelism, overlapping computation with data movement, and navigating vast design spaces of tradeoffs. These principles are timeless: the same ideas that shaped decades of systems evolution shape today's AI systems and will shape whatever comes next.

On the training side, topics include GPU kernels, compilation of tensor programs, the parallelism and memory optimizations that fit massive models onto limited hardware, and fine-tuning methods that adapt models by updating only a small fraction of their parameters. On the inference side, topics include KV-cache management, batching, and splitting the serving pipeline across specialized machines. Underneath both sits storage: the data layouts and retrieval pipelines that feed models, where every bit stored and moved bounds the space of possible designs. Agentic applications are the workloads driving these designs, from serving agents efficiently to fine-tuning models for custom agents. We take the principles to their logical conclusion: self-designing systems that automatically synthesize their own architecture for a given workload, hardware, and performance target.

The class is research-oriented, built on recent and classical papers and interactive discussion, with a semester-long systems project integrated with DASlab research.

### What is this class not about?

This is not an introduction to using AI frameworks, prompting models, or training models with PyTorch recipes. It is a systems class about system design: you will learn how AI training and inference systems work at their core and how to design new systems for emerging applications and hardware. By the way, if you know how systems work, you also become better at using them!

### Why take this class?

The systems that train and serve large models are where the AI race is won or lost: a faster training system means more experiments per week, and a faster inference system means more users served at lower cost. The industry is investing hundreds of billions of dollars in AI infrastructure, and the demand for people who understand these systems' internals vastly exceeds supply. At the same time, the field is young and moving fast: what constitutes

good system design is being redefined every year, which makes it the perfect area to do research. CS265 exposes students to the core internals of AI training and inference systems and, through its interactive, research-driven format, is also a great class for anyone who wants to learn how CS research is done.

### **Expected learning outcomes**

- Learn state-of-the-art research and industry trends in AI training and inference systems.
- Understand the tradeoffs in designing and implementing modern AI systems.
- Reason about any AI system through two goals: minimizing data movement and maximizing GPU and CPU utilization.
- Master the first principles that govern all systems and how they apply from classical data systems to LLM training and inference, and to whatever comes next.
- Develop basic research skills: reading, writing, and understanding research papers.
- Deepen systems programming, debugging, and performance profiling skills (C/C++, CUDA, PyTorch internals).

## **CLASS PHILOSOPHY**

CS265 has unlimited office hours, relies on the latest research papers instead of a textbook, and its lectures are based on interaction and discussion instead of just “lecturing”. Many of the problems we discuss are open research problems, and most of all, it is fun! The instructor and the TFs are here to help you every day, at all times, throughout the semester: request as many meetings as you like and as much help as you want. The course is geared toward creative thinking and problem-solving to give you a sense of how computer science research takes place. Many of our students have conducted research projects with DASlab and published papers.

### **How does CS265 compare to CS165?**

If you took and liked CS165, you will like CS265 as well. The same first principles you used to design a data system kernel now apply to the systems that train and serve large models. CS265 is even more interactive and research-oriented: semester projects focus on open research problems with the potential to lead to publication, and every class centers on a single research paper and its detailed understanding. In terms of workload, CS165 is implementation-heavy (5-10K lines of code toward a full system prototype); CS265 is focused on ideas and design, with smaller amounts of code used to explore alternative designs and expose their tradeoffs.

## **CLASS STRUCTURE**

### **Interaction in every class**

The class meets twice a week and is primarily discussion-based: think of each class as an extended brainstorming session about a specific problem. In the first weeks, the instructor gives a series of lectures to establish the principles and background. After that, each class opens with a quick round where every student shares, in about 30 seconds, one question or one idea they brought from the paper, followed by a student presentation of the paper — together these trigger the day's discussion of design trends, tradeoffs, open problems, and opportunities for innovation.

## Office hours and labs

Interaction does not stop at lecture time: we offer daily office hours and labs. Prof. Idreos holds OH weekly starting Week 1, with additional weekend slots periodically. Labs, run by the TFs, are the place for hands-on project help: bring your laptop for persistent bugs, tooling questions (debugging, performance testing), or feedback on your code. Rooms and slots are posted on the class website.

## Recordings and sections

All classes are recorded and available online, so you can catch up on anything you miss and revisit discussions throughout the semester. Sections are pre-recorded videos that go deeper into specific concepts or deliver background needed for upcoming classes and the project; questions about section material are welcome in OH and labs.

## Research sessions

On select days, DASlab PhDs and postdocs present their recent research, connect it to the class material, and discuss open problems and research opportunities with you. Snacks and drinks provided.

## Reading papers

There are no written reviews in CS265. Instead, come to each class having read the paper of the day with the following nine questions in mind — they are the backbone of every discussion and every presentation:

- What is the problem?
- Why is it important?
- Why is it hard?
- Why do existing solutions not work?
- What is the core intuition for the solution?
- Does the paper prove its claims?
- What is the setup of analysis/experiments? Is it sufficient?
- Are there any gaps in the logic/proof?
- Describe at least one possible next step.

In addition, bring one thing to every discussion class: a question the paper left you with, or an idea for improving or extending it. Each class opens with a quick round — 5-10 minutes, about 30 seconds per student — where everyone shares theirs. This is how each brainstorming session begins: your question or idea may well become the discussion of the day.

## Presentations

Each student presents at least one paper during the semester, with 1-2 slides per question above plus detailed slides on the paper's core idea. Avoid bullet lists; follow the formatting of the first lectures: simple slides, minimal text, figures and visual examples, one slide = one message. Slides are reviewed by the instructor at least 24 hours before class and finalized 30 minutes before class. You are welcome to come to OH for help while preparing!

## AI USE

We encourage you to use LLMs — enthusiastically. You are taking a class about the systems behind these models; you should be using them, pushing them, and forming opinions about where they break. But use them the right way around: the outcome of this class is not delivering a task, it is the learning you keep.

In practice: when you face a design problem, a paper, or a bug, first work on it independently — think it through, struggle a bit. That struggle is where the learning happens, and it is exactly what we explore together in class, OH, and labs. Then accelerate: use LLMs to automate the mundane parts, to probe your understanding, to get feedback on your design, and to explore alternatives you had not considered. What you must not do is skip the first step: an LLM-produced deliverable you cannot defend teaches you nothing, and the final evaluation is a 1-1 meeting where you walk us through your design decisions and answer questions about them. The deliverable of this class is the knowledge in your head. Everything else is scaffolding.

## WHO CAN TAKE THIS CLASS?

**IF** you have taken CS165, CS161, or CS261 **GOTO** next section; **ELSE** see below.

Background: prior knowledge of C programming and systems programming, and a solid understanding of computer architecture — particularly the memory hierarchy and caches — is essential (CS50 and especially CS61 or equivalent). Good hacking, algorithm design, and data structures skills are also required. For the systems projects, comfort with Python and PyTorch is expected; the GPU track additionally uses C++ and CUDA.

Graduate students who have taken a mix of systems classes (database, operating, distributed systems) will be fine; we provide enough background to follow. CS265 satisfies the systems requirement for a PhD. Seniors who have taken CS61 but not CS161/CS165 should talk to the instructor. Students without the full background can take the class on the “systems track”, starting with the systems project to build background before moving toward research.

### How can I do great in 265?

Utilize all the resources provided. Show up in class and participate. Come to daily office hours and labs as often as possible. When you are stuck on a design decision or a bug, it is normal to struggle for a while — it is part of the learning process — but after some time, grab your laptop and come by!

### How to prepare

Especially without CS165, spend some time on the foundations before and during the early weeks of the semester. The principles of this class were born in classical data systems, so browse these foundational texts (from the CS165 readings): (1) Architecture of a Database System (Hellerstein, Stonebraker, Hamilton; FnT DB 2007); (2) The Design and Implementation of Modern Column-store Database Systems (Abadi, Boncz, Harizopoulos, Idreos, Madden; FnT DB 2013); (3) Massively Parallel Databases and MapReduce Systems (Babu, Herodotou; FnT DB 2013). Also browse last year's lecture slides (classes 1-12, posted on the website) — they are self-contained and show the style of thinking the class is built on.

Test 0: a standalone data structure design-and-implementation exercise in C that (1) helps you judge your fit for the class and (2) bootstraps your coding skills. Supporting section videos on system coding and profiling are posted on the class website.

## GRADING

- Class discussions: 30%
- Paper presentation: 15%
- Semester project: 45%
- Midway check-in: 10%

## SEMESTER PROJECT

Each student completes one semester project: a systems project or a research project.

**Systems projects** provide hands-on experience building state-of-the-art versions of the techniques we study, with a design component and an implementation component addressing low-level issues such as memory management, hardware-aware processing, parallelism, and performance tradeoffs. This year's projects are: (1) implementing an activation checkpointing algorithm inside PyTorch (profiler, checkpointing policy, modified execution), and (2) building a lightweight LLM inference engine with C++ orchestration and CUDA kernels (embedding, normalization, attention, matrix multiplication), then optimizing it via memory access patterns, tiling, and kernel fusion. Systems projects are individual. Detailed descriptions are on the class website.

**Research projects** aim at open problems, are integrated with active DASlab research, and are meant to lead to publication. They are mostly about thinking, reading, and writing, with proof-of-concept implementations as the end goal in some cases, and are done in groups of three. They are offered, in consultation with the instructor, to students who have taken CS165 or have significant systems background. Topics are announced within the first three weeks; see the latest DASlab publications for active directions. Systems projects can evolve into research projects for students who progress fast.

### Evaluation

There are no finals or midterms. For systems projects, the description specifies the functionality and performance targets; for research projects, we set concrete questions at project setup. At the end of the semester, each student meets 1-1 with the instructor and with the TFs to demonstrate their project and answer design questions. Tip: if you participate frequently in OH and labs, we already know your system and your progress, and the final evaluation becomes a mere formality.

### Collaboration

The systems project is individual: all code, documentation, and reports must be your own, written from scratch. Discussing design and implementation problems with other students is allowed and encouraged — in class, OH, and brainstorming sessions. Research projects are in groups of three; cross-team discussion is encouraged, but each team's deliverable must be clearly its own.

### Deadlines and midway check-in

All projects are due, and graded, at the end of the semester. The project description includes a recommended schedule; it is an ideal plan you may adjust, but note that falling more than a couple of weeks behind usually means not finishing (we will send frequent reminders). Midway check-in (early March, 10%): a design document and one performance experiment demonstrating an early result; a template is provided early in the semester.

## LOGISTICS

### No laptop/phone policy

CS265 is based on interaction, and we bring printed slides for every student so you can follow along and take notes on paper. Research shows laptop note-taking can hurt understanding [Mueller & Oppenheimer, Psychological Science 2014]. If you need a device (accessibility, an expected important call), just let the instructor know and permission will be granted.

### Online discussions

We use Ed (links on the class website; anonymous posting enabled). Basic rules: search before posting; stay on top of pinned staff posts (anything we post is considered “known”); no code or debugging on the forum — bring it to daily labs in person or via Zoom screen share; anything about your personal progress or fit belongs in OH, not the forum.

### Guest lectures

Every semester we arrange guest lectures by leaders in system design from industry and academia, with the opportunity to actively participate in discussion with our guests.

### Textbook

There is no textbook for state-of-the-art AI system design. We use recent and classical research papers and surveys, posted on the course website.

### Feedback

We welcome feedback and ideas about the course at any point — come chat with us during office hours and tell us how you are keeping up and how we can make it easier for you.

### Plagiarism

You are responsible for understanding Harvard and Harvard Extension School policies on academic integrity ([extension.harvard.edu/resources-policies/student-conduct/academic-integrity](https://extension.harvard.edu/resources-policies/student-conduct/academic-integrity)) and how to use sources responsibly. Not knowing the rules, misunderstanding the rules, running out of time, submitting “the wrong draft”, or being overwhelmed are not acceptable excuses. See the Harvard Guide to Using Sources and the free 15-minute tutorials at the Tips to Avoid Plagiarism page.

### Accessibility

Harvard and the Extension School are committed to an accessible academic community. The Disability Services Office offers accommodations and services to students with documented disabilities ([extension.harvard.edu/resources-policies/resources/disability-services-accessibility](https://extension.harvard.edu/resources-policies/resources/disability-services-accessibility)). Do not hesitate to contact Prof. Idreos directly by email with any questions or concerns.

## EXTENSION SCHOOL

CS265 is a highly research-oriented course, structured very differently from other classes. Successfully completing it in practice requires participation in OH and labs — they are where you learn how to think about

the material and design solutions. If you do not have the full background above, budget time for frequent participation and several additional hours of work per week.

Lectures are broadcast live and available on demand within 24 hours; links on Canvas and the class website. Extension students participate live in classes, OH, and labs via Zoom (sessions marked “remote”), including screen sharing for debugging help. Classroom microphones capture the in-class discussions. Extension OH and labs are held remotely on weekends to accommodate students with day jobs; the schedule is posted at the start of the semester.

Grading: because live class participation is not practical for all remote students, there is no class-participation grade; its 30% is redistributed. Extension breakdown: project 60%, presentation 25%, midway check-in 15%. Graduate-credit students additionally deliver one more detailed performance experiment extending their semester project (e.g., an additional design alternative evaluated in depth, or a deeper analysis of a tradeoff in their system), due at the end of the semester (30% of the project grade; guidelines announced early in the semester). A separate Ed forum serves extension students; see the class website. Note: the extension school lists the start date one day after the actual first class, which you can still stream live as usual.