

 <https://hao-ai-lab.github.io/dsc291-s24/>

DSC 291 D00: ML Systems Spring 2024



Hao Zhang

 [@haozhangml](https://twitter.com/haozhangml)  [@haoailab](https://twitter.com/haoailab)

 haozhang@ucsd.edu



Instructor

Hao Zhang (<https://cseweb.ucsd.edu/~haozhang/>)

Now: Asst. Prof @ HDSI, Affiliated with CSE, UCSD

- Ph.D. from CMU CS, 2020

- Project: Parameter servers (Week 3), automatic parallelization (Week)

- Took 4-year leave to work for a startup (raised 100M+), 2016-2021

- Project: Petuum, productional ML (CSE 234)

- Then postdoc at UC Berkeley working on LLM+systems, 2021 – 2023

- Project: Alpa, vLLM, Vicuna, lmsys.org

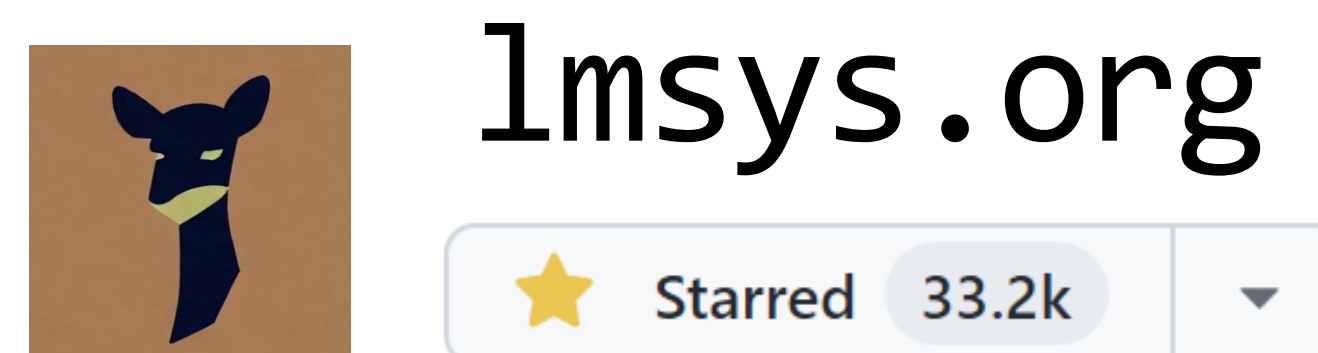
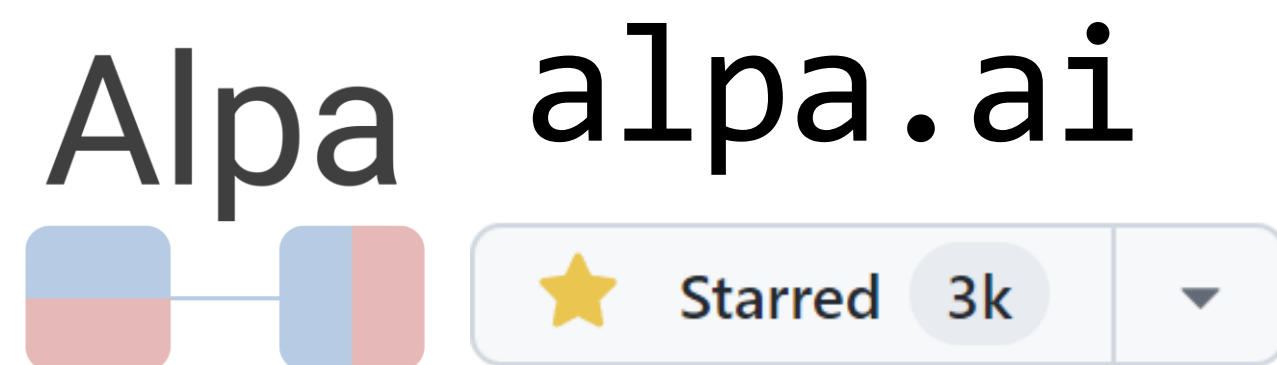
My Lab: Hao AI Lab

Research Area: Machine Learning + Systems

Recent topics:

- Fast LLM Inference and Serving (Week 8)
- Large-scale distributed ML, Model parallelism, etc. (week 6)
- Open source LLMs, data curation, evaluation (week 7)

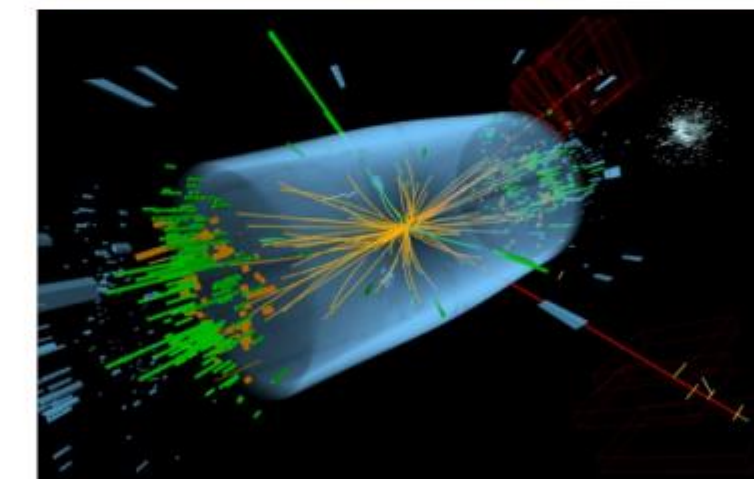
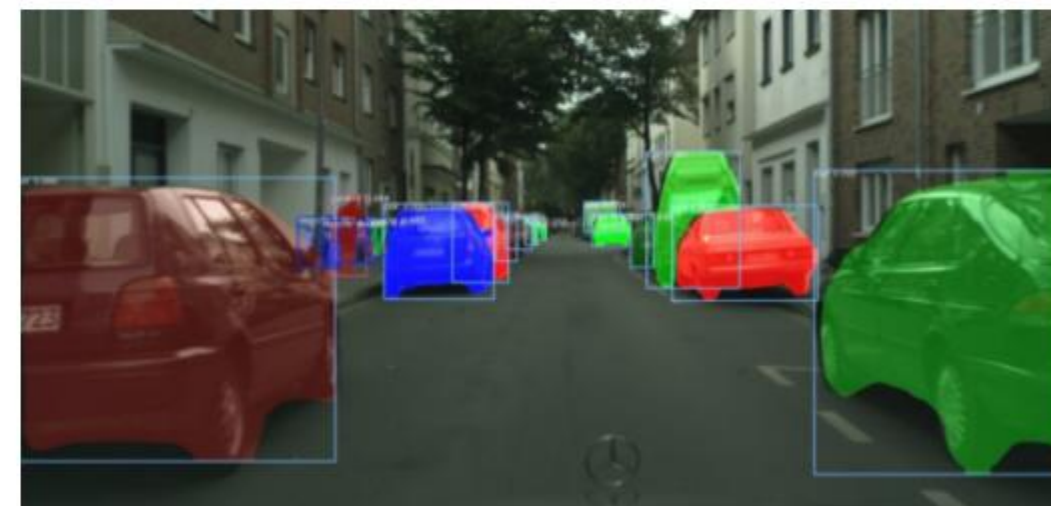
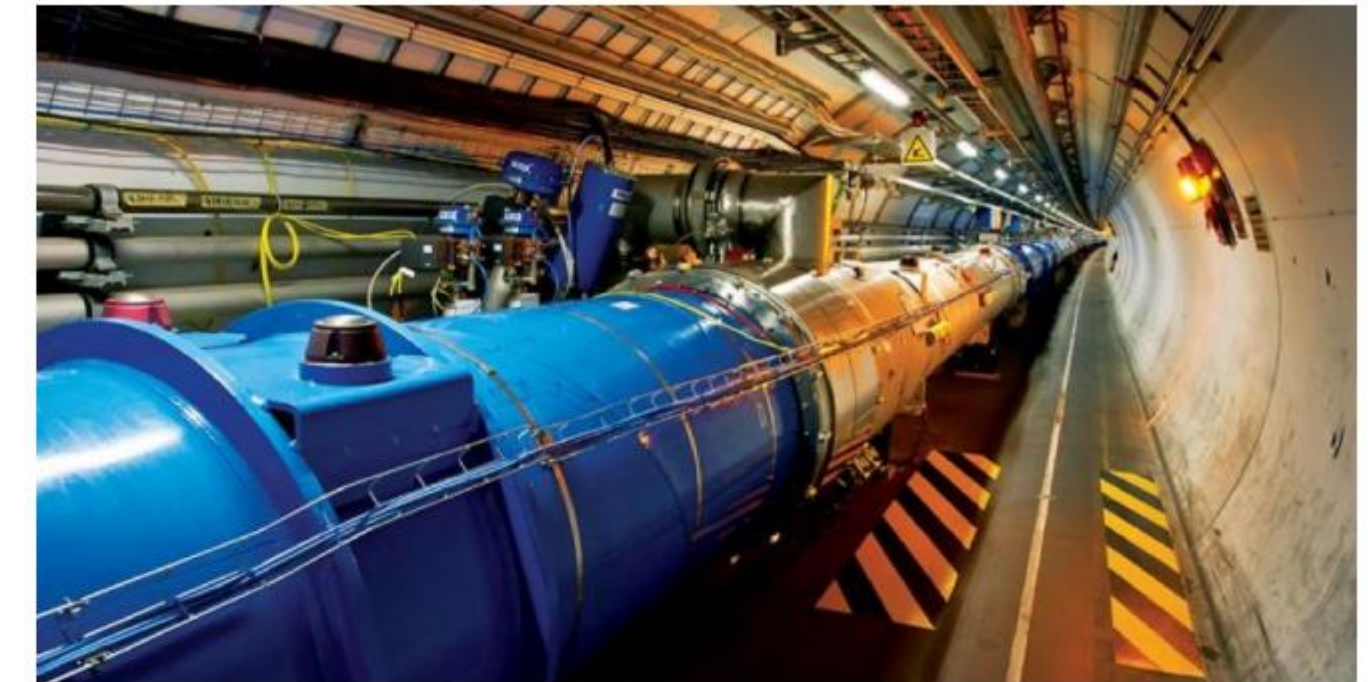
Some ongoing projects:



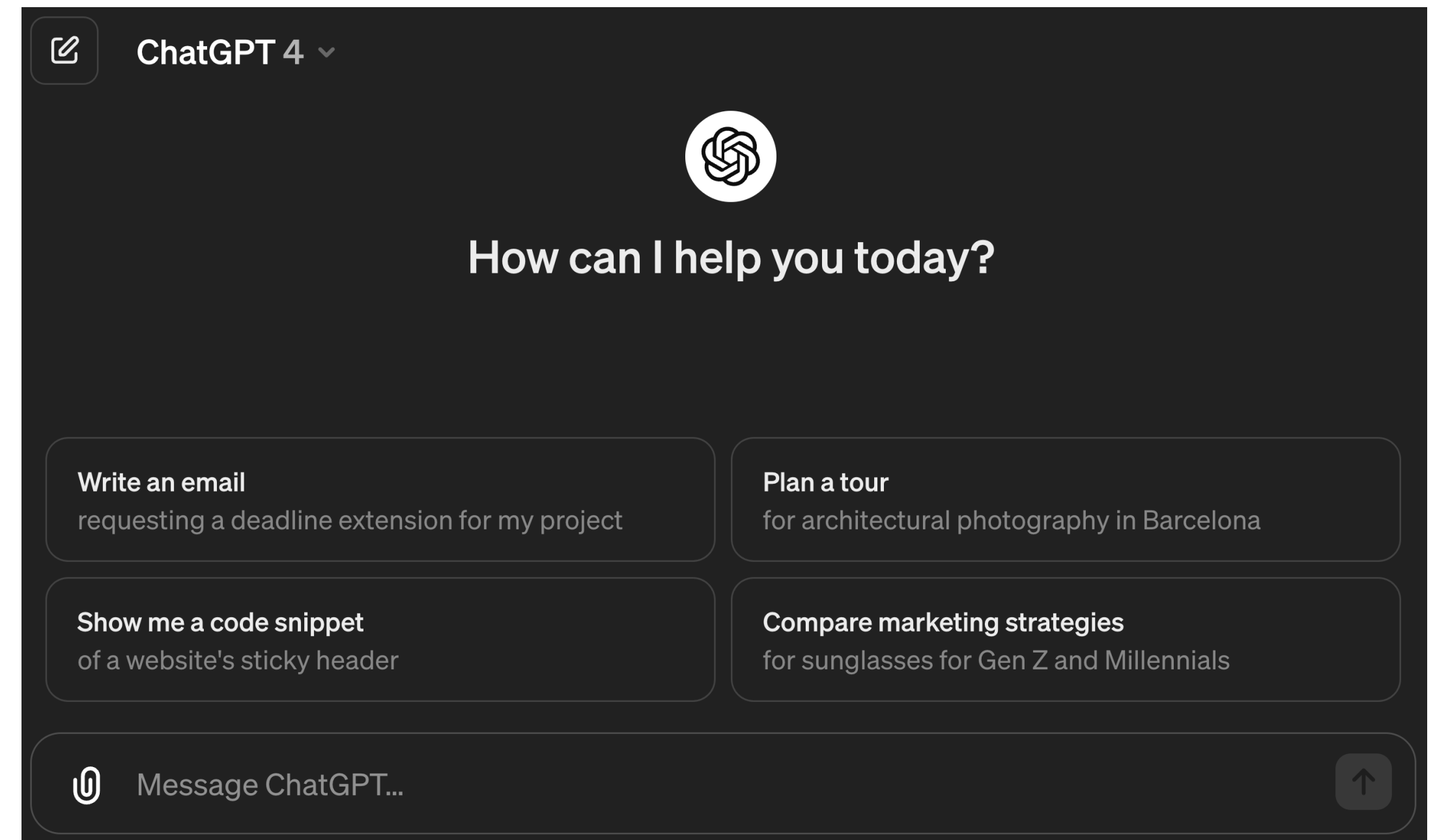
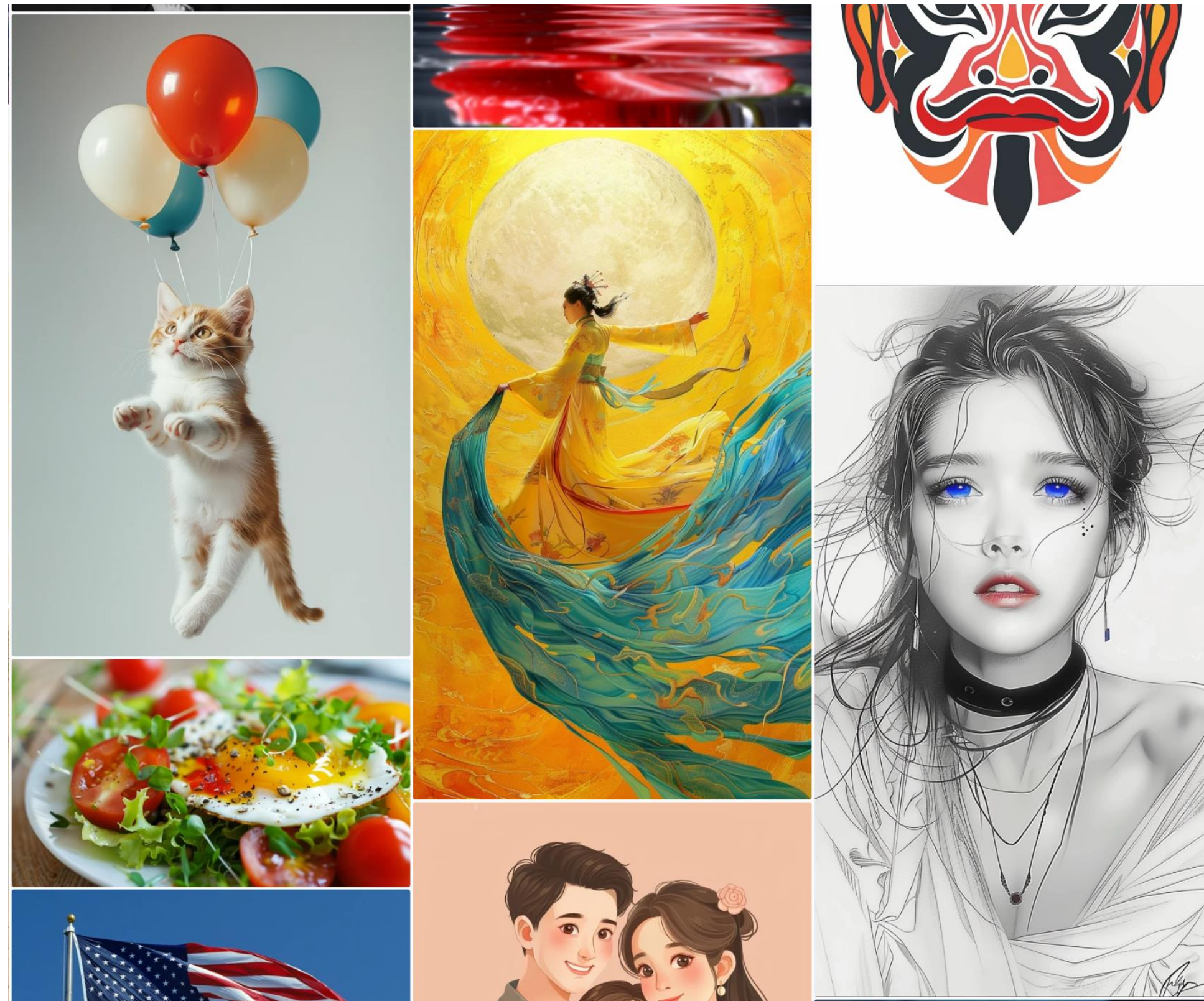
Today

- **Why study ML Systems**
- Course overview
- Logistics
- Warm up (If time permits)

Success of Machine Learning Today



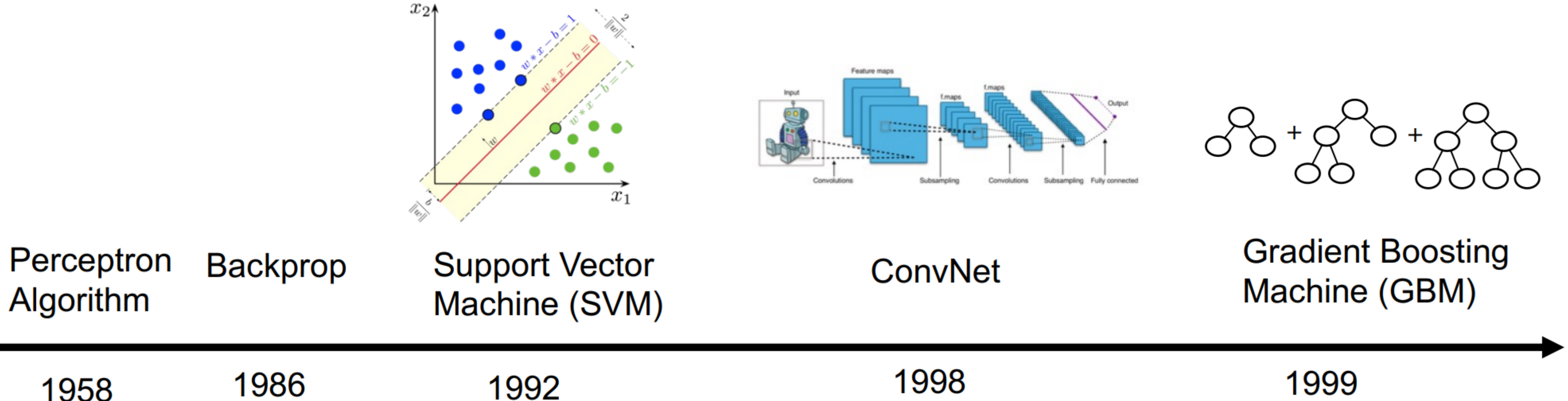
Generative AI



How this happened?

A key ingredient: ML Systems

1958 – 2000: ML Research



Many algorithms we use today are **created before 2000**

2000 – 2010: Arrival of Big Data



2001



2004

MTurk

2005



2009

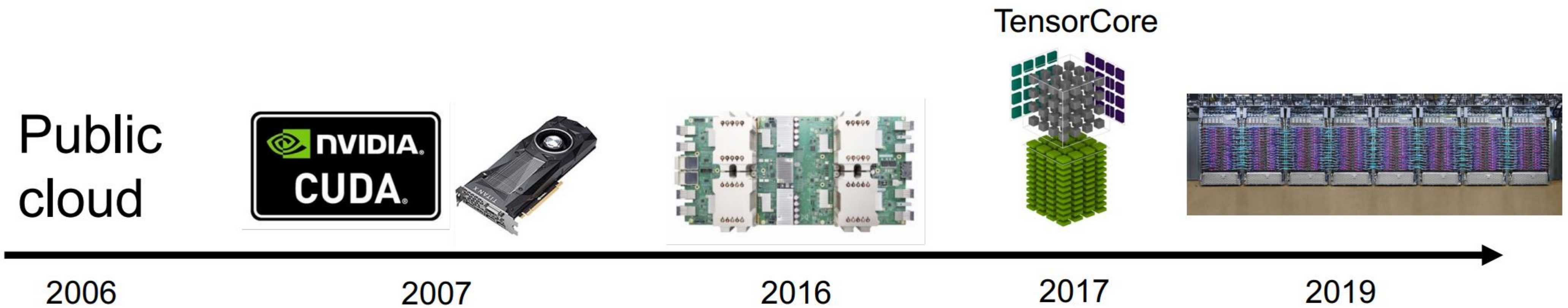


2010



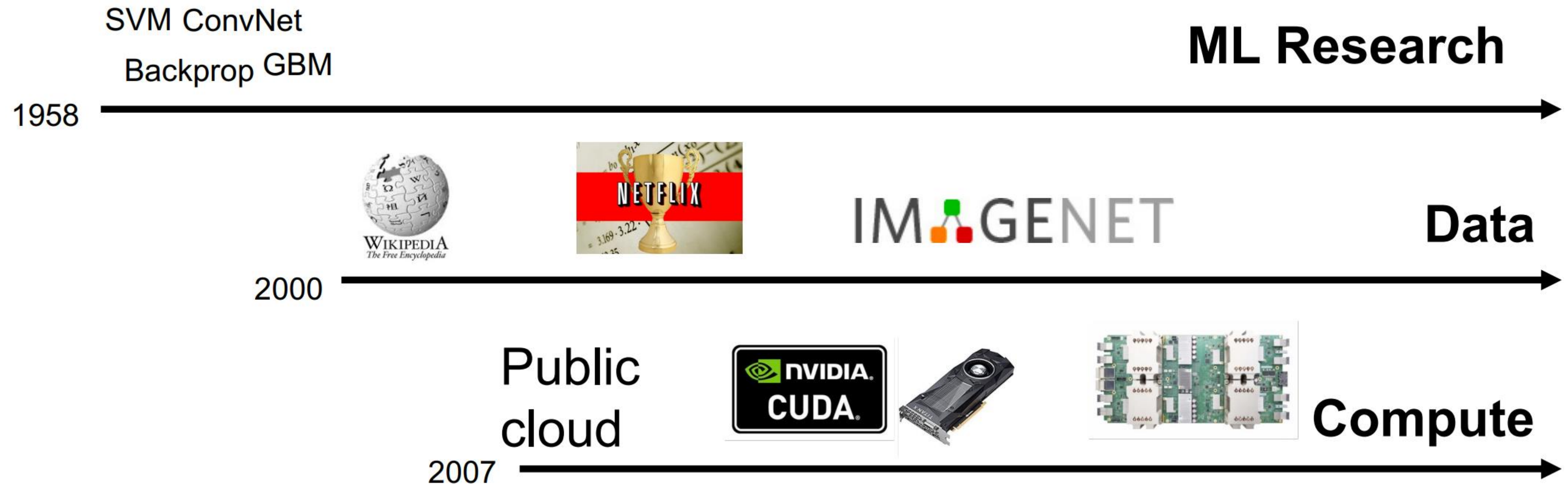
Data serves as fuel for machine learning models

2006 – Now: Compute and Scaling



Compute scaling

When three things come together and ready?



The bitter lesson by Richard Sutton

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess. When a simpler, search-based approach with special hardware and software proved vastly more effective, these human-knowledge-based chess researchers were not good losers. They said that ``brute force" search may have won this time, but it was not a general strategy, and anyway it was not how people played chess. These researchers wanted methods based on human input to win and were disappointed when they did not.

A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning by self play, and learning in general, is like search in that it enables massive computation to be brought to bear. Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research. In computer Go, as in computer chess, researchers' initial effort was directed towards utilizing human understanding (so that less search was needed) and only much later was much greater success had by embracing search and learning.

In speech recognition, there was an early competition, sponsored by DARPA, in the 1970s. Entrants included a host of special methods that took advantage of human knowledge---knowledge of words, of phonemes, of the human vocal tract, etc. On the other side were newer methods that were more statistical in nature and did much more computation, based on hidden Markov models (HMMs). Again, the statistical methods won out over the human-knowledge-based methods. This led to a major change in all of natural language processing, gradually over decades, where statistics and computation came to dominate the field. The recent rise of deep learning in speech recognition is the most recent step in this consistent direction. Deep learning methods rely even less on human knowledge, and use even more computation, together with learning on huge training sets, to produce dramatically better speech recognition systems. As in the games, researchers always tried to make systems that worked the way the researchers thought their own minds worked---they tried to put that knowledge in their systems---but it proved ultimately counterproductive, and a colossal waste of researcher's time, when, through Moore's law, massive computation became available and a means was found to put it to good use.

In computer vision, there has been a similar pattern. Early methods conceived of vision as searching for edges, or generalized cylinders, or in terms of SIFT features. But today all this is discarded. Modern deep-learning neural networks use only the notions of convolution and certain kinds of invariances, and perform much better.

This is a big lesson. As a field, we still have not thoroughly learned it, as we are continuing to make the same kind of mistakes. To see this, and to effectively resist it, we have to understand the appeal of these mistakes. We have to learn the bitter lesson that building in how we think we think does not work in the long run. The bitter lesson is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning. The eventual success is tinged with bitterness, and often incompletely digested, because it is success over a favored, human-centric approach.

One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are *search* and *learning*.

The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; instead we should build in only the meta-methods that can find and capture this arbitrary complexity. Essential to these methods is that they can find good approximations, but the search for them should be by our methods, not by us. We want AI agents that can discover like we can, not which contain what we have discovered. Building in our discoveries only makes it harder to see how the discovering process can be done.

A real example AlexNet 2012

Year 2012

Methods

SGD
Dropout
ConvNet
Initialization

Data

IM  GENET

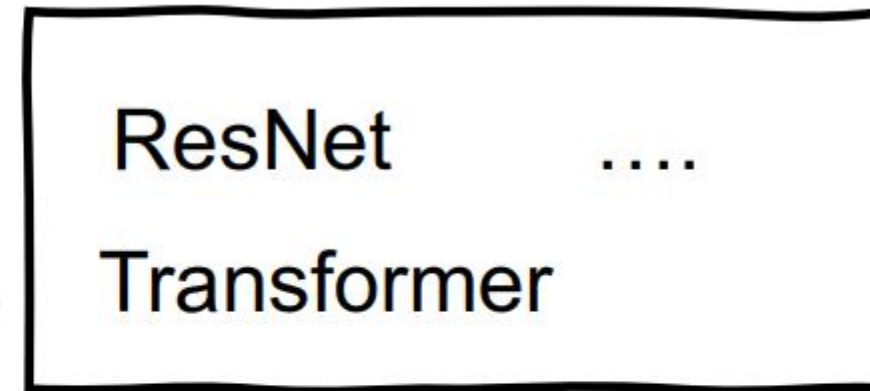
1M labeled
images

Compute

Two GTX 580

Six days

W/o ML Systems



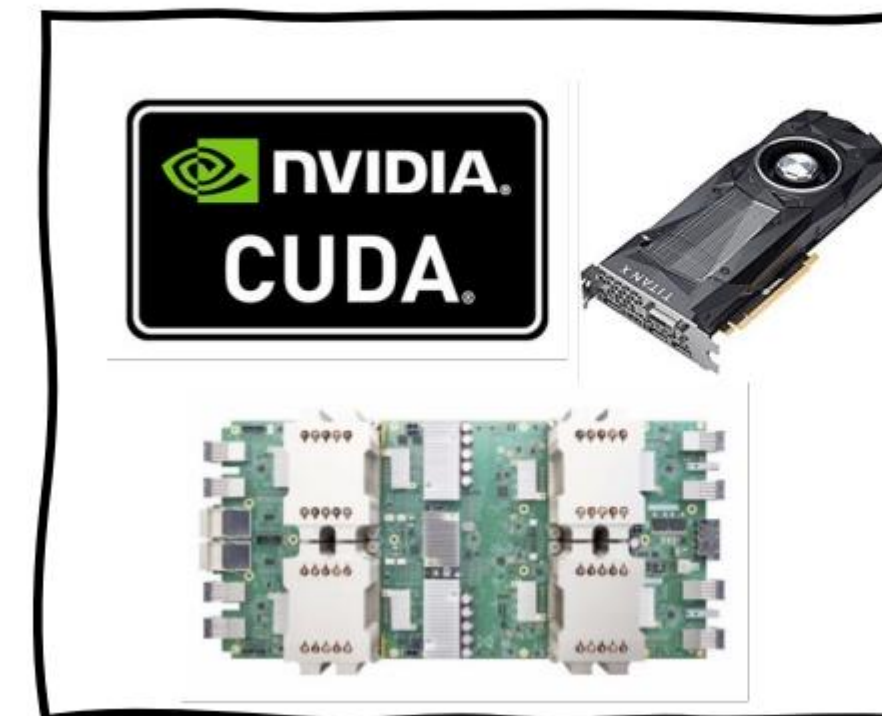
ML Research

44k lines of code

Six months



Data



Compute

W/ ML Systems



Researcher

ResNet
Transformer

ML Research

100 lines of python

A few hours

System Abstractions

Systems (ML Frameworks)



IMAGENET

Data

NVIDIA.
CUDA.



Compute



ML Systems

- Accelerate ML research
- Bring ML up to scale
- Help deploy ML to everyone
- ML \leftrightarrow system codesign
- In summary: ML System is becoming an essential skill

Example problem



Need to improve self-driving car's pedestrian detection to be **X-percent accurate**, at **Y-ms latency budget**

Traditional Way of ML Thinking



Need to improve self-driving car's pedestrian detection to be **X-percent accurate**, at **Y-ms latency budget**

Design a better model with smaller amount of compute via hyperparameter tuning, pruning, distillation

Traditional Way of System Thinking



Need to improve self-driving car's pedestrian detection to be **X-percent accurate**, at **Y-ms latency budget**

Take the best model by ML researcher,
specialize the inference system to reduce
latency

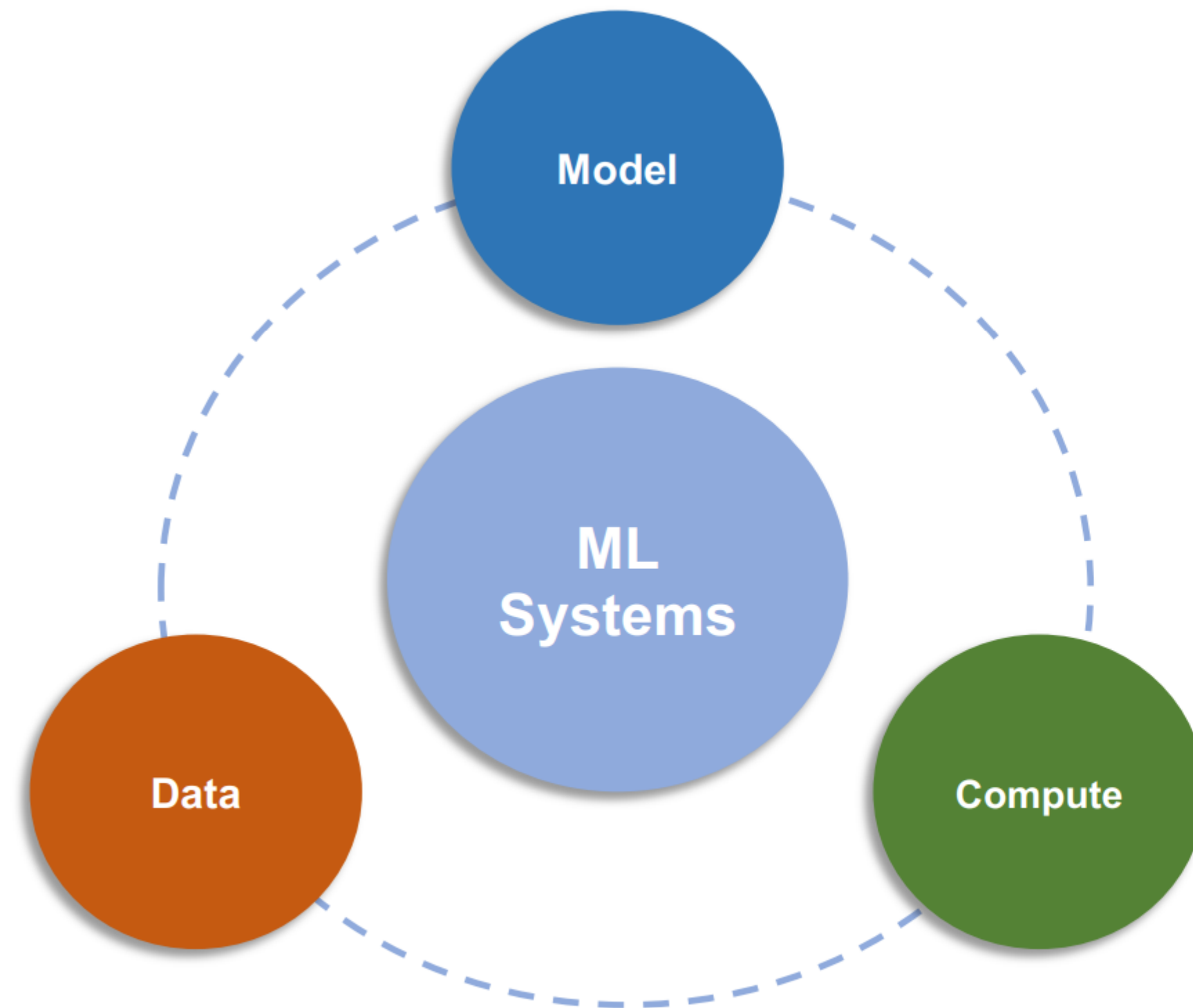
ML-System way of Thinking



Need to improve self-driving car's pedestrian detection to be **X-percent accurate**, at **Y-ms latency budget**

- Collect more data
- Incorporate specialized compute hardware
- Develop models components that optimizes for the specific hardware
- Study data-model scaling law
- Scale it up using many hardware, subject to compute budget
- Automate the hyperparameter search processes , subject to compute budget
- Streamline the entire process from development to deployment

MLSys as an Emerging Research Field



AI Systems Workshop at NeurIPS

MLSys tracks at Systems/DB conferences

Conference on Machine Learning and Systems ([MLSys.org](https://mlsys.org))

MLSys: The New Frontier of Machine Learning Systems

Summary: Why Study MLSys

- **Reason #1** To push the frontier of modern AI applications, we need to have a holistic approach to the problem, understand and make use of existing systems more efficiently.
- **Reason #2** Prepare ourselves to build machine learning systems and work in the area of machine learning engineering.
- **Reason #3** Have fun building our own ML systems!

Today

- Why study *ML Systems*
- **Course overview**
- Logistics
- Warm up (If time permits)

ML System history

- ML Systems evolve as more and more ML components (models/optimization algorithms) are unified

Ad-hoc: diverse model family,
optimization algos, and data

Opt algo: iterative-convergent

Model family: neural nets

Model:
CNNs/transformers/GNNs

LLMs: transformer
decoders

Our models/algos
become more and
more specialized

ML System Scale

Our scale increases -- we double down more resources on a specialized model

Ad-hoc: diverse model family, optimization algos, and data

Opt algo: iterative-convergent

Model family: neural nets

Model:
CNNs/transformers/GNNs

LLMs: transformer decoders

Single-core CPU

Many CPUs and multi-threads

GPUs, accelerators, LPU

8 GPUs in one Box: nvidia DGX

Massively distributed GPUs: 10K GPUs



Hence the course is organized into four parts

- ML System basics
- Single device optimization
- Scaling up: distributed ML, ML parallelization
- LLM + systems

What is this course about? Part 1 Basics

- Machine learning system basic
 - Deep learning
 - Computational graph
 - Autodiff
- ML frameworks: TF/PyTorch, Imperative vs. declarative
- GPUs and CUDA
- Collective Communication

Part 2: Single-device optimization

- Single device optimization
 - Compute optimization
 - Hardware acceleration
 - Graph optimization / fusion
 - Memory optimization
 - Compilation

Part 3: Scaling Up

- Distributed ML Basics
- Machine learning parallelism
 - Data parallelism, model parallelism
 - Inter-op parallelism, intra-op parallelism
- Automatic parallelization

Part 4: Putting in the Context of LLMs

- Transformers Large language model (LLM) basics
- Scaling law
- LLM training, inference, and serving
- LLM-specific optimization
 - Flash attention
 - Paged attention
 - Speculative decoding
 - MoEs

What is this course not about?

- Not an **OS/Distributed system/networking basic** course
- We will assume you know how OS/Distributed system/networking works
- Things to check:
 - How processors works: instruction, ALUs?
 - How computer/OS works: memory hierarchy, scheduling,
 - How networking works

What is this course not about?

- Not a **Linear algebra/deep learning/NLP/optimization** course
- We will assume you have knowledge on ML/DL/optimizations
- Things to check:
 - Basic linear algebra
 - How neural networks work? Training and inference
 - CNNs/RNNs/transformers/graph neural networks?
 - How gradient-based optimization works?
 - Gradient descent/SGD/Adam/L2 norm etc.
 - How these models enable applications?
 - CNN -> image classification, LLM -> language modeling

What is this course not about?

- Not a **Linear algebra/deep learning/NLP/optimization** course
- We will assume you have knowledge on ML/DL/optimizations
- Things to check:
 - Basic linear algebra
 - How neural networks work? Training and inference
 - CNNs/RNNs/transformers/graph neural networks?
 - How gradient-based optimization works?
 - Gradient descent/SGD/Adam/L2 norm etc.
 - How these models enable applications?
 - CNN -> image classification, LLM -> language modeling

What this course does not cover

- MLSys is expanding its scope, a lot of interesting topics
- This course will not cover:
 - Machine learning for systems: learned data base index, learned networking, etc.
 - ML Hardware design
 - this is a software course
 - Scaling down: Quantization, compression, TinyML
 - This course focuses more on system and scaling up
 - Federated learning, ML energy efficiency, system security in ML

Contrast with similar course offerings in UCSD

- Vs DSC 204A: scalable data analytics
 - DSC 204A focuses more on data systems in general and many basics
 - This course is more advanced and focused: ML systems
- Vs. CSE 234: Data system for machine learning
 - CSE 234 is more data-centric (more data component than ML components)
 - CSE 234 talks about more on systems put in production
 - This course is ML-centric
 - This course is more cutting-edge: latest technologies up-to-date

Suggested Textbooks

- This is a fast-evolving field hence NO TEXTBOOK
- But there are a lot of online materials:
 - Deep learning book by Ian Goodfellow, Yoshua Bengio, and Aaron Courville: more principles
 - Dive into deep learning: by Mu Li, Alex Smola, Aston Zhang
 - More coding components
 - ML compilation: by Tianqi Chen etc.
 - Designing Machine Learning Systems, by Chip Huyen.
 - TensorFlow/PyTorch/DeepSpeed documentations

Learning outcomes of this course

By the end of this course, you will ...

... understand the basic functioning of modern DL libraries, including concepts like automatic differentiation, compute operators, etc.

... understand hardware acceleration/CUDA/GPUs, and can program/debug a little accelerator programs

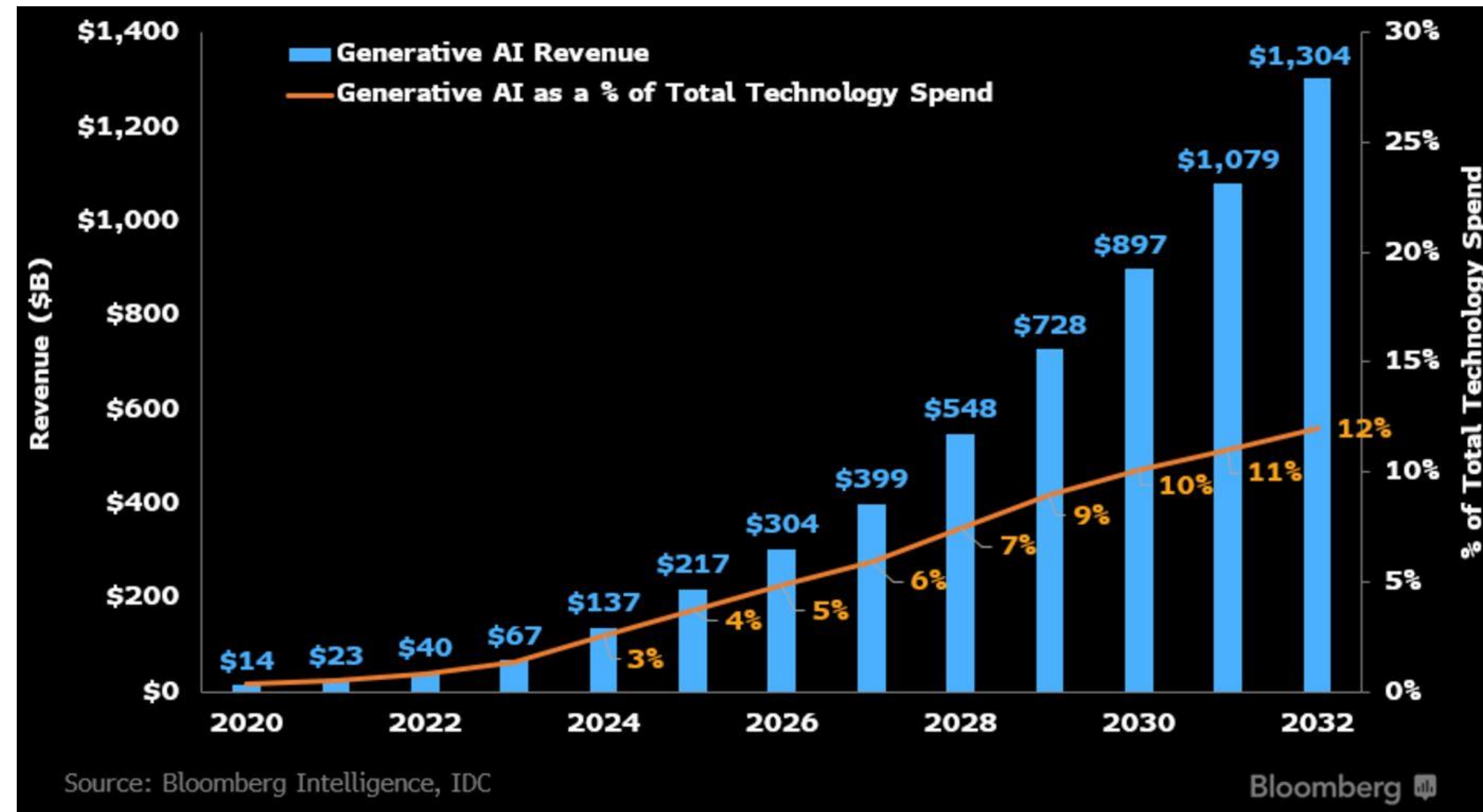
... understand scaling-up, why and how? All sorts of machine learning parallelization techniques, and latest research in the area


... ground all you learned in the context of LLMs, understand the L of LLM, how it is optimized, scaled, trained, served.

... Have fun

... and a more practical outcome:
\$\$\$

Global Picture: AI Industry



 **NVIDIA Corp**
NASDAQ: NVDA

[Overview](#) [Videos](#) [Compare](#) [Financials](#)

Market Summary > NVIDIA Corp

894.52 USD

[+ Follow](#)

+893.70 (108,987.80%) ↑ all time

Closed: Apr 2, 5:39 PM EDT • Disclaimer

After hours 892.67 -1.85 (0.21%)

1D | 5D | 1M | 6M | YTD | 1Y | 5Y | Max



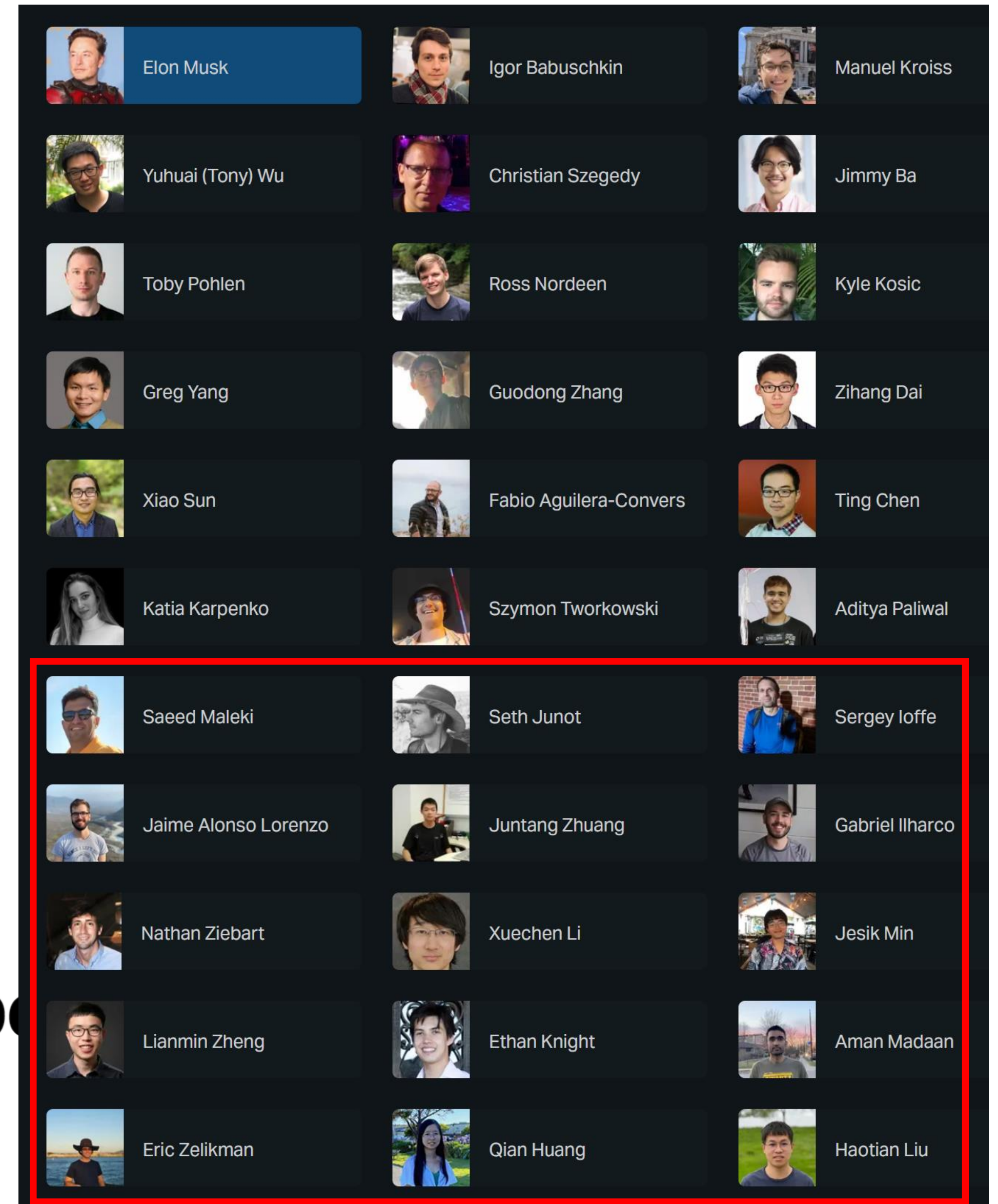
What others are doing

Sergey Brin, Mark Zuckerberg have personally recruited AI staffers as talent war heats up

While ChatGPT-maker OpenAI pays its prized recruits a reported compensation package ranging from \$5 million to \$10 million mostly in the form of stock, Zuckerberg's shop is offering a relatively measly \$1 million to \$2 million annual wage, according to The Information.

AI

Databricks picks up MosaicML, an OpenAI competitor, for \$1.3B



Questions?

Course website

DSC 291

Home
Syllabus
Assignments
Schedule Overview
Resources
FAQs
Staff

DSC 291: Machine Learning Systems

Instructor: Hao Zhang, UC San Diego, Spring 2024

[Toggle Dark Mode](#)

Announcements

Week 1 Announcements

Apr 1 · 0 min read

- Welcome to the Spring 2024 offering of DSC 291: Machine Learning Systems!
- We're excited to work with you throughout the quarter!
- Check out the [tentative schedule](#).
- This is a new course offered for the first time at UCSD, hence we might adjust the schedule and content depending on your learning progress.
- The first lecture starts on April 2nd, 5 pm at MANDE B-150.

Week 1

Apr 2:	1 Introduction <i>Reading: N/A</i> SURVEY Beginning of Quarter Survey
Apr 4:	2 Basics: Deep learning, computational graph, autodiff, ML frameworks <i>Reading: N/A</i>



<https://hao-ai-lab.github.io/dsc204a-w24/>

TAs

- Anze Xie (a1xie@ucsd.edu)
 - MS @ CSE
 - Experience: long-context LLMs, longchat, developed LightSeq
 - OH: Every Friday 3 – 4pm on Zoom
- Will Lin (w5lin@ucsd.edu)
 - PhD @ CSE
 - Experience: FPGA, Disaggregation
 - OH: Every Tuesday 3 – 4pm at CSE 3144

Components and Grading

- 3 Programming Assignments: **42%** (15% + 15% + 12%)
 - 5 late days.
- Exams
 - No Midterm
 - Final Exam (date/location TBD): **26%**
- Reading summary: **8%**, 8 readings, 10 – 20 pages per reading
 - No late days.
- Final presentation: **15%**
- Peer-instructed in-class Quiz: **9%**, 8 - 9 quizzes
- Extra Credit: **6%**

Grading Scheme (grade is the better of the two)

Grade	Absolute Cutoff (\geq)	Relative Bin (Use strictest)
A+	95	Highest 5%
A	90	Next 10% (5-15)
A-	85	Next 15% (15-30)
B+	80	Next 15% (30-45)
B	75	Next 15% (45-60)
B-	70	Next 15% (60-75)
C+	65	Next 5% (75-80)
C	60	Next 5% (80-85)
C-	55	Next 5% (85-90)
D	50	Next 5% (90-95)
F	< 50	Lowest 5%

Grading Scheme (grade is the better of the two)

Grade	Absolute Cutoff (\geq)	Relative Bin (Use strictest)
A+	95	Highest 5%
A	90	Next 10% (5-15)
A-	85	Next 15% (15-30)
B+	80	Next 15% (30-45)
B	75	Next 15% (45-60)
B-	70	Next 15% (60-75)
C+	65	Next 5% (75-80)
C	60	Next 5% (80-85)
C-	55	Next 5% (85-90)
D	50	Next 5% (90-95)
F	< 50	Lowest 5%

Example, 82 and 33%,

Rel: B-; Abs: B+;

Final: B+

The structure of the course (Tentative)

Week	Topic
1-2	Basics: Deep learning, computational graph, autodiff, ML frameworks
3	GPUs, CUDA, Communication
4	ML Compilation, graph optimizations
4	Guest lecture: TBD
5	Communication and memory optimization, distributed ML, data parallelism
6	Model parallelism, auto-parallelization
7	Transformers, LLMs, scaling law
8	LLM training, inference and serving, attention optimizations
9	Guest lecture: TBD
9	Student presentations
10	Student presentations
10	Final exam reviews
11	Final exam, date TBD



<https://hao-ai-lab.github.io/dsc291-s24/>

Lectures

- Hao's lecture: high encouraged to attend
 - In person unless due to travel or weather
 - Will have peer-instruction quiz to promote attendance
- Guest lectures: You must attend (mostly on Zoom)
 - About 3, from inventors of key techniques covered in class
 - TAs will track the attendance
- Student presentations (final two weeks)
 - You must attend to either give presentations or grade your peers presentations

Programming Assignments

- Three newly designed PAs
- Will be based on PyTorch / Huggingface
- Topics
 - Implement computational operator/graphs
 - Optimize them
 - Scale it up using parallelism
 - Debug correctness and benchmark performance
- We will use Google Colab's free GPUs
 - TA will publish a guide
- We are in touch with the university IT to see if we can get more GPUs

Expectations on the PAs

- Expectations on the PAs:
 - Individual projects; see webpage on academic integrity
- Be prepared: plan to spend large amount of time on it
 - Esp. if you do not have a lot of experience in programming at low-level
- TAs will explain and demo the tools; handle all Q&A
- You are expected to put in the effort to learn the details of the tools' APIs using their documentation on your own!

Reading Summary

- Required reading:
 - The instructor team will select 8 most important papers (mostly < 20 pages).
 - One paper per week, submit your reading by next week Tuesday midnight.
 - Your reading summary should focus on high-level ideas, and should be ≥ 2 pages of the Neurips format.
- Optional reading:
 - Important papers, cutting-edge topics
 - Encourage to read if you want to learn more
 - Helpful for PAs

Formatting Instructions For NeurIPS 2020

David S. Hippocampus*
Department of Computer Science
Cranberry-Lemon University
Pittsburgh, PA 15213
hippo@cs.cranberry-lemon.edu

Abstract

The abstract paragraph should be indented $\frac{1}{2}$ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Submission of papers to NeurIPS 2020

NeurIPS requires electronic submissions. The electronic submission site is

<https://cmt3.research.microsoft.com/NeurIPS2020/>

Please read the instructions below carefully and follow them faithfully.

1.1 Style

Papers to be submitted to NeurIPS 2020 must be prepared according to the instructions presented here. Papers may only be up to eight pages long, including figures. Additional pages *containing only a section on the broader impact, acknowledgments and/or cited references* are allowed. Papers that exceed eight pages of content will not be reviewed, or in any other way considered for presentation at the conference.

The margins in 2020 are the same as those in 2007, which allow for $\sim 15\%$ more words in the paper compared to earlier years.

Authors are required to use the NeurIPS L^AT_EX style files obtainable at the NeurIPS website as indicated below. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

1.2 Retrieval of style files

The style files for NeurIPS and other conference information are available on the World Wide Web at

<http://www.neurips.cc/>

The file `neurips_2020.pdf` contains these instructions and illustrates the various formatting requirements your NeurIPS paper must satisfy.

The only supported style file for NeurIPS 2020 is `neurips_2020.sty`, rewritten for L^AT_EX 2_ε. **Previous style files for L^AT_EX 2.09, Microsoft Word, and RTF are no longer supported!**

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Exams

- No Mid-term
- In-person Final exam (26%)
- All MCQs
- You can bring as many books/cheat sheets/paper you want
- No phone/laptop/Internet/ChatGPT
- Data/location TBD

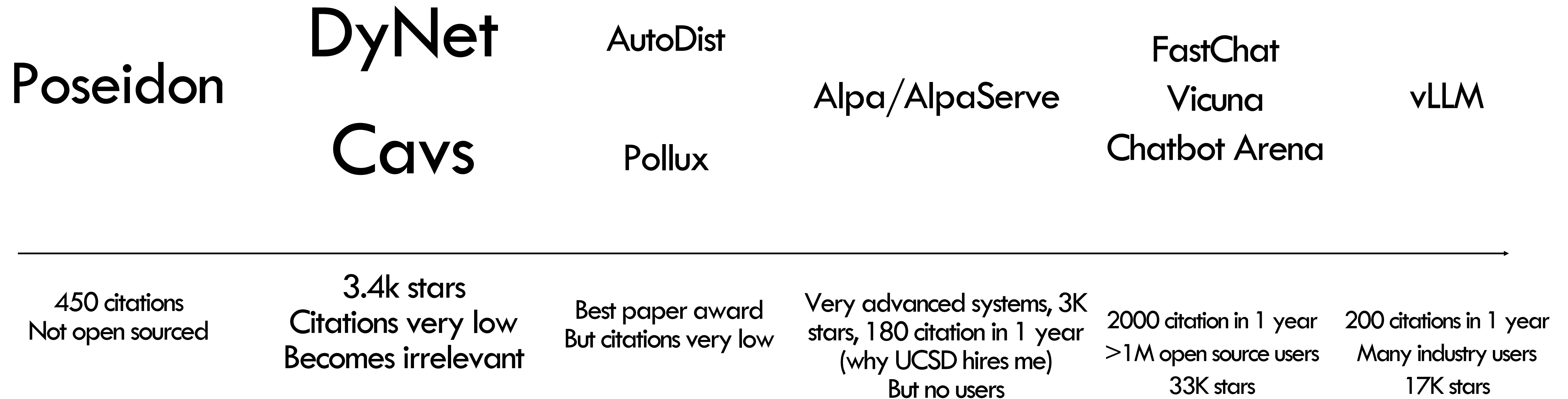
Final presentation

- One primary goal of this class:
 - Learn about ML systems development by
 - Reading several seminal papers
 - Trying their corresponded artifacts (with many stars in Github)
 - Judging how they do things right (or wrong)
 - Share the knowledge with your classmate with a nice presentation
 - Predict the future trends in this area

Final Presentation

- 15%
- Sign-up sheet in website: 4 – 6 per team
- Please spend a significant amount of time on studying your chosen project and making this presentation nice and clear.
- Graded by instruction team (50%) and your classmates (50%)
 - Instructor: based on format, correctness, depth, clarity, insights
 - Peers: make sure your classmates feel they indeed learn something after listening to your presentation
- Happening in the end of the quarter

Instructor Story



Final Presentation: some principles

- What is the problem?
 - Simple, powerful system that users themselves can easily evolve
- What is the project's main idea?
 - Minimalist design, unified abstractions (avoid 2nd system syndrome)
- Play and Evaluate its artifact:
 - Simple interface, easy to install/use?
 - Academic impact vs. practical impact?
- Why did it succeed (or failed)?
 - Uniqueness
 - Addictive to use
 - Open-source
 - High level language made it easy to port to other architectures
 - Become relevant/irrelevant as technology changes?
- TAs will release more rubrics about how to prepare/grade the final presentations

Peer-instruction Quiz (Will)

Goal: make sure you do not forget!

Frequency: once per week since week 2.

Formats:

- MCQ (some of them will appear in final exams)
- We are using iClicker App for attendance!
- Download iClicker App on your phone (free)
 - <https://www.iclicker.com/students/apps-and-remotes/apps>
- Login using your @ucsd.edu email address
- Find DSC 291 ML Systems
- Join class
- Download and setup by next class!
-



Respecting TAs' time

- Use piazza first, seeking helps from your peers
- Students answering questions on Piazza will be rewarded
- Office hours are for getting ideas on how to debug or better approach your homework.
- Write a description! Try to narrow down your problem area as much as possible.
- If you don't have a description, TA can reject your questions.
- Respect TA's working hours.
 - Respond in 24 hours.
 - Members may send msgs at night or on weekends, but only expect to receive a reply on weekday.

General Dos and Do NOTs

- Do:
 - Follow all announcements on Piazza
 - Try to join the lectures/discussions live
 - Participate in discussions in class / on Piazza
 - Raise your hand before speaking
 - View/review podcast videos asynchronously by yourself
 - To contact me/TAs, use piazza first; if you really need to email, use “DSC 204:” as subject prefix

General Dos and Do NOTs

- Do NOT:
 - Harass, intimidate, or intentionally talk over others
 - **Violate academic integrity** on the PAs, exams, or other components; I (and the school) am very strict on this matter!

Questions?