https://hao-ai-lab.github.io/cse234-w25/

CSE 234: Data Systems for Machine Learning
Winter 2025

Staff

Instructor: Hao Zhang

TAs: Abhilash, Daniel, Junda, Ruiyi

@haozhangml @haoailab

haozhang@ucsd.edu

https://hao-ai-lab.github.io/cse234-w25/

CSE 234: ~~Data~~ Systems for ~~Machine Learning~~ LLMs
Winter 2025

Staff

Instructor: Hao Zhang

TAs: Abhilash, Daniel, Junda, Ruiyi

@haozhangml @haoailab

haozhang@ucsd.edu

Instructor

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

- Ph.D. from CMU CS, 2020
  - Projects: Parameter server (week 3), auto-parallelization (week 5)
- Took 4-year leave to work for a "not-so-successful" startup (raised 100M+), 2016-2021
  - Projects: Petuum, MLOps (Previous offering of CSE 234)
- Then postdoc at UC Berkeley working on LLM+systems, 2021 – 2023
  - Projects: vLLM, Vicuna, lmsys.org, Chatbot Arena (Week 8)
- Then co-founded a small startup and acquired by SNOW and started at UCSD

My Lab: https://hao-ai-lab.github.io/

Research Area: Machine Learning + Systems

Recent topics:

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

Some ongoing projects:

lmsys.org
Starred 37.4k

vLLM vllm.ai
Starred 33.2k

FastVideo
Starred 776

Today

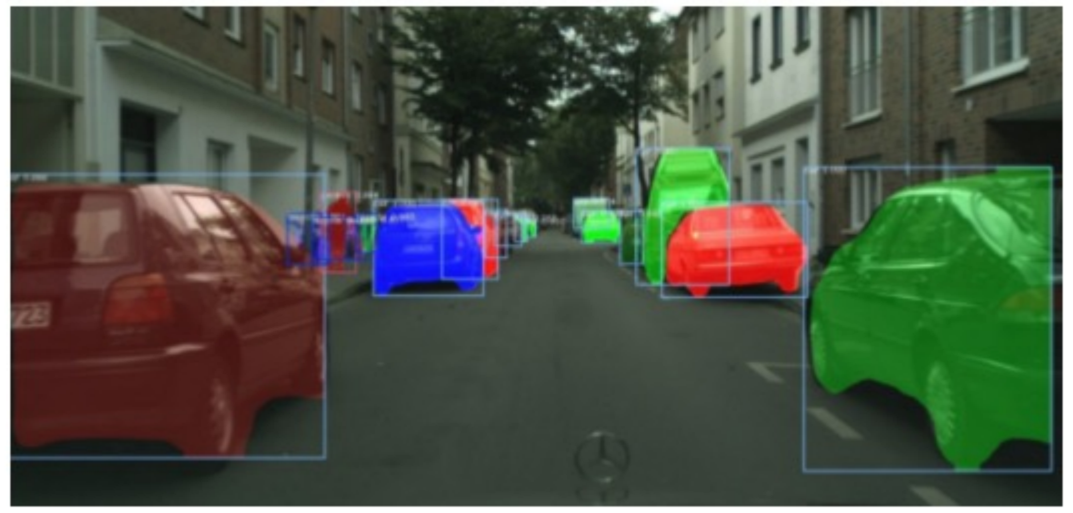
Why study ML Systems

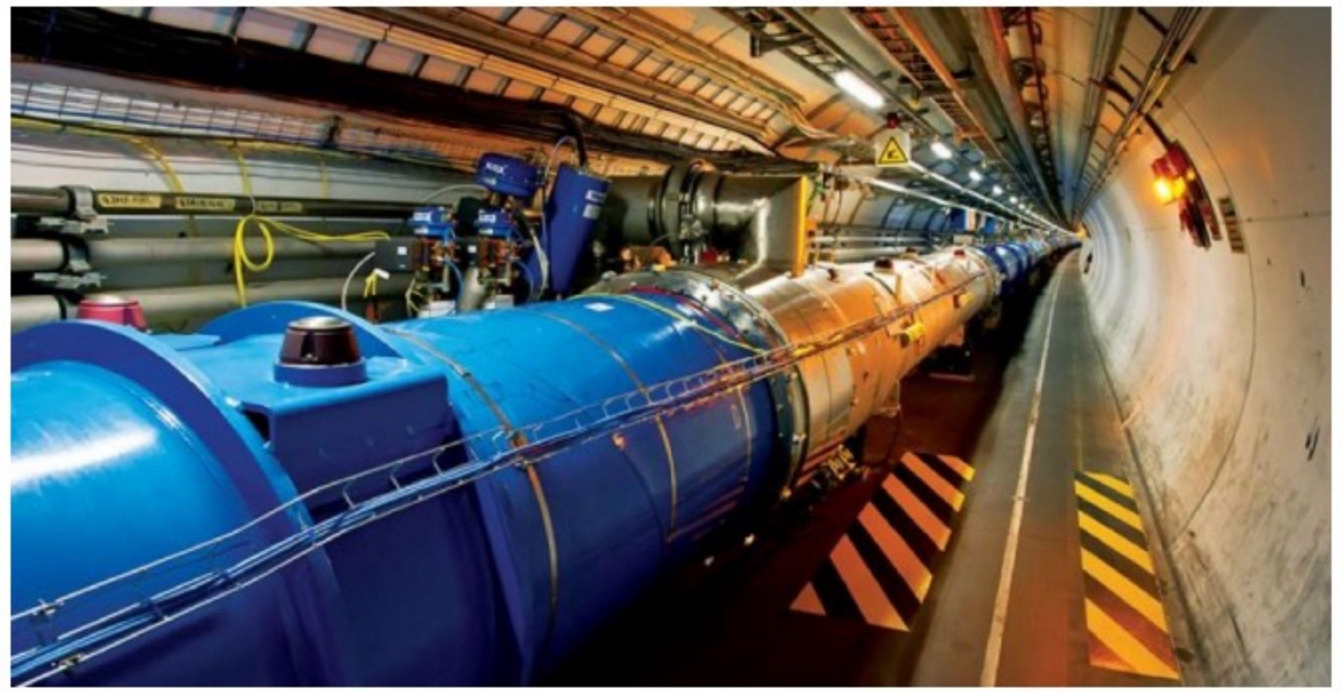
Course overview

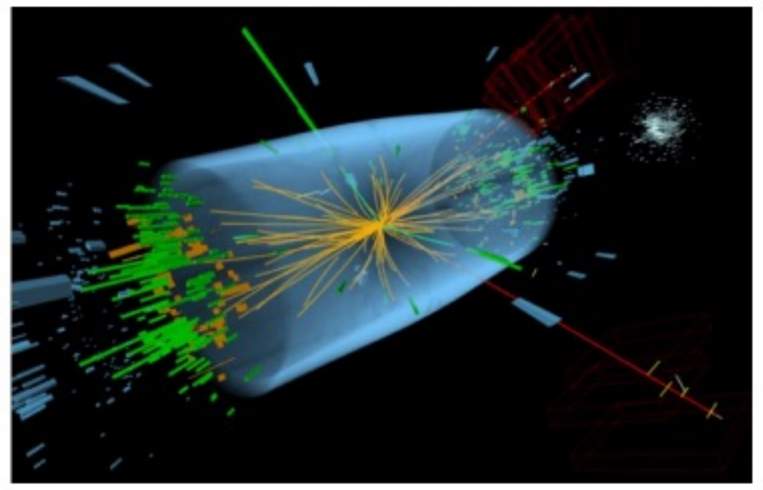
Logistics

Warm up (If time permits)

Success of Machine Learning Today

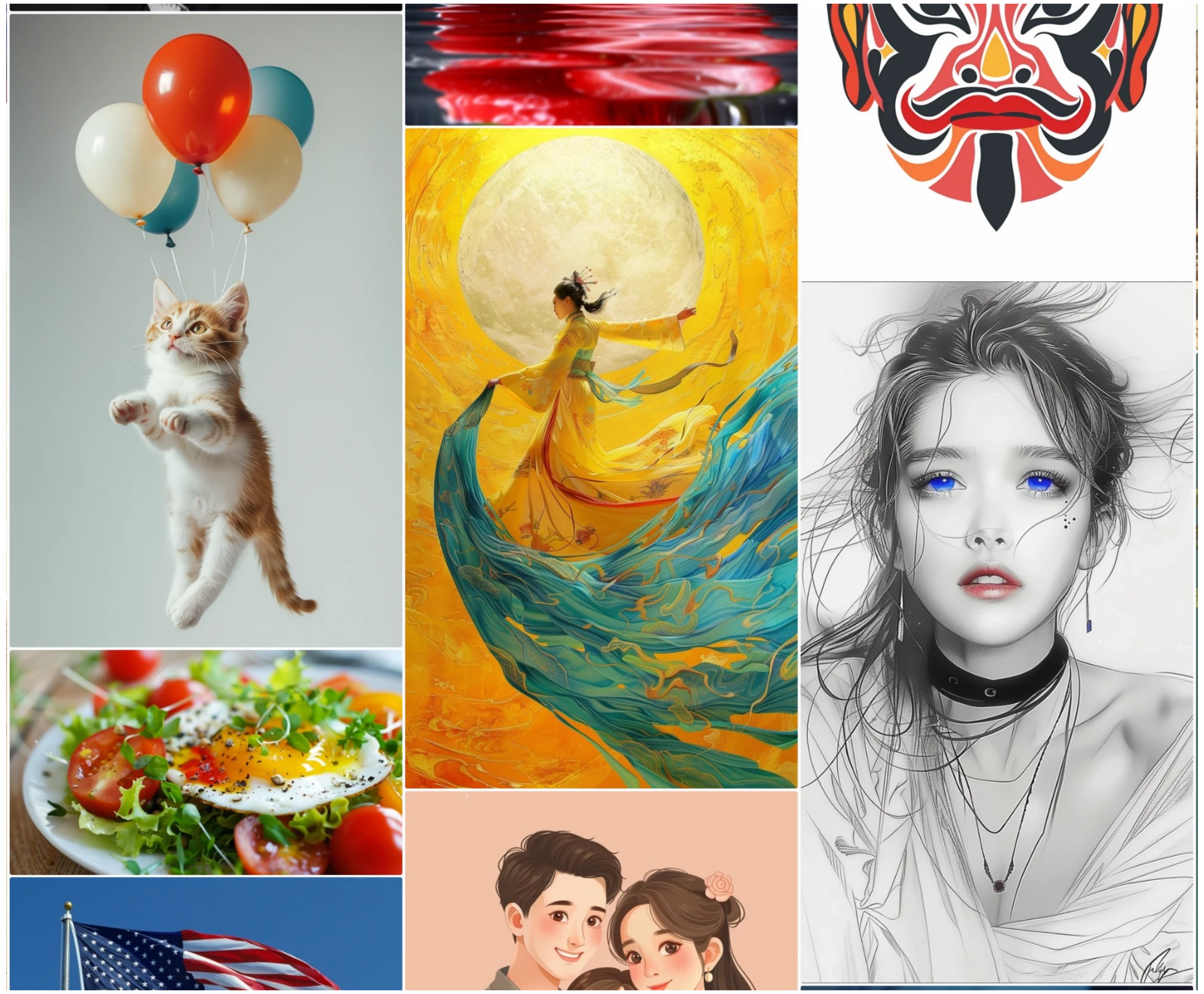


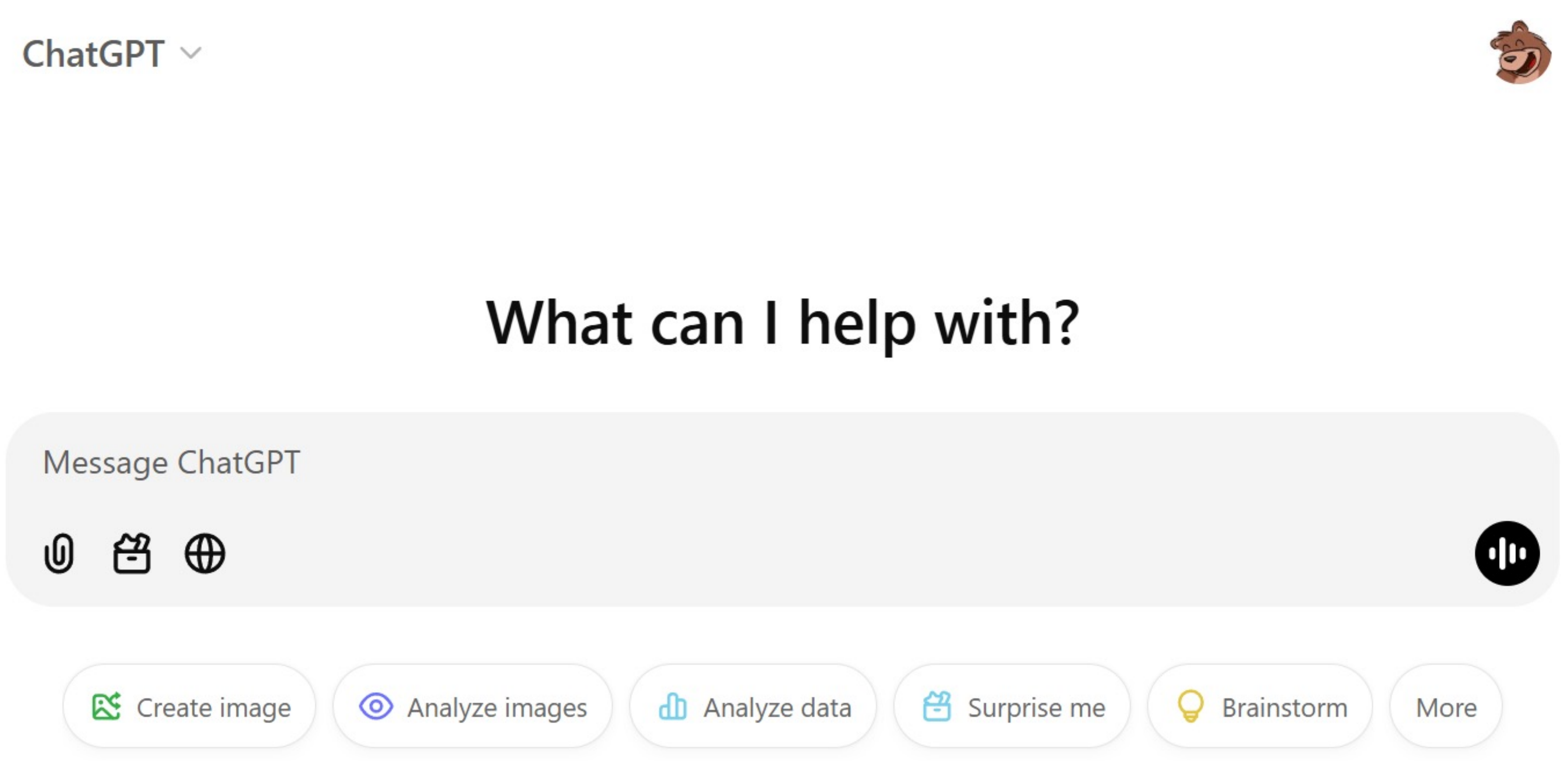


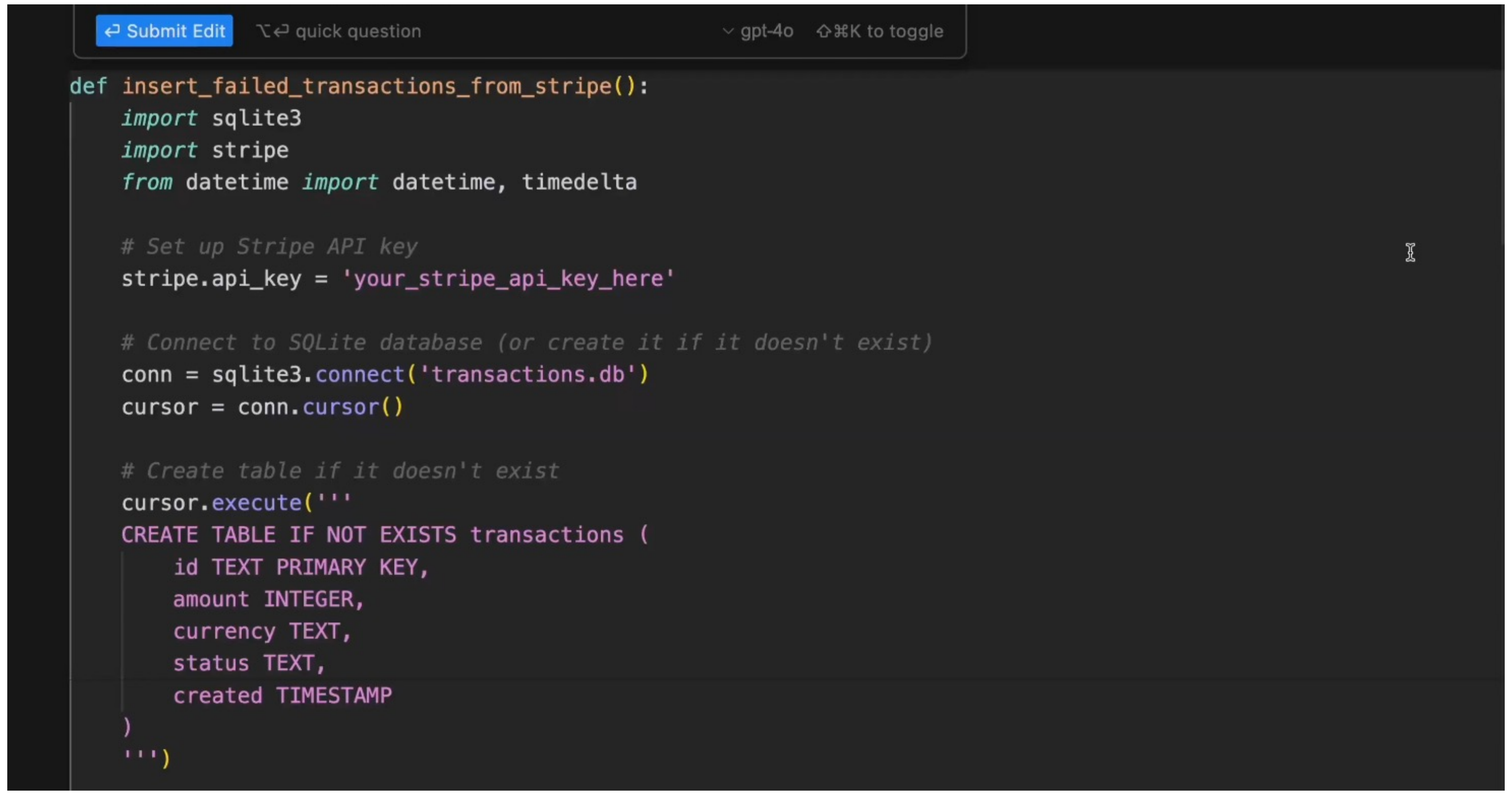


Higgs challenge

Generative AI







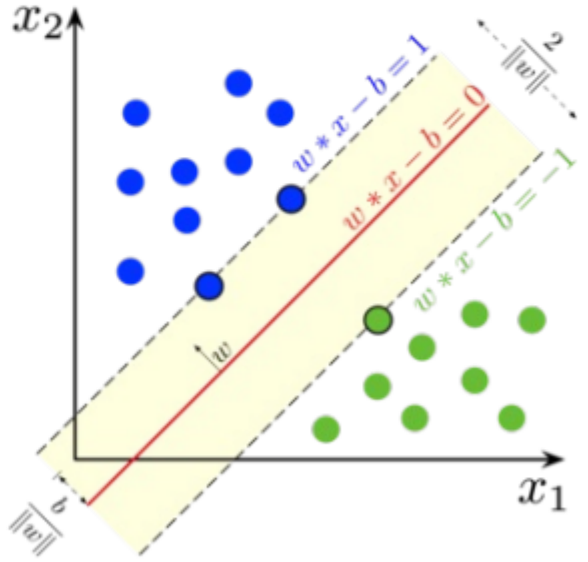
How this happened?

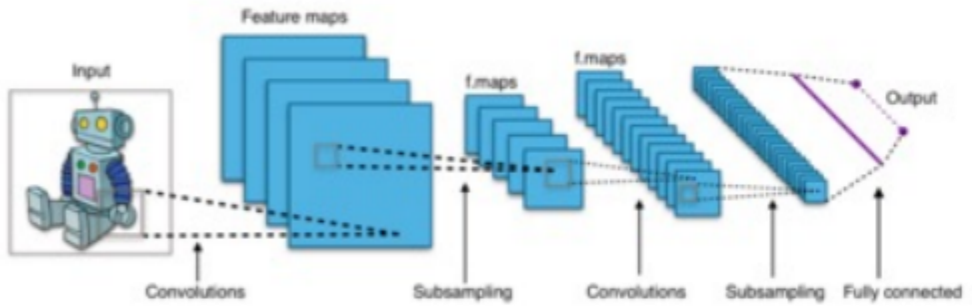
A key ingredient: ML Systems

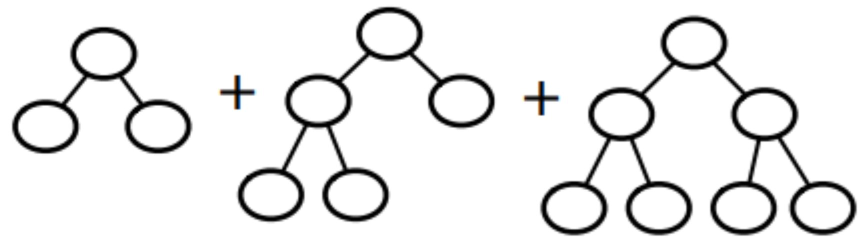
I will Provide 3 Arguments

1. ML system is an essential skill today
2. Developing ML-system way of thinking is essential for solving future problems
3. Reveal later

1958 – 2000: ML Research







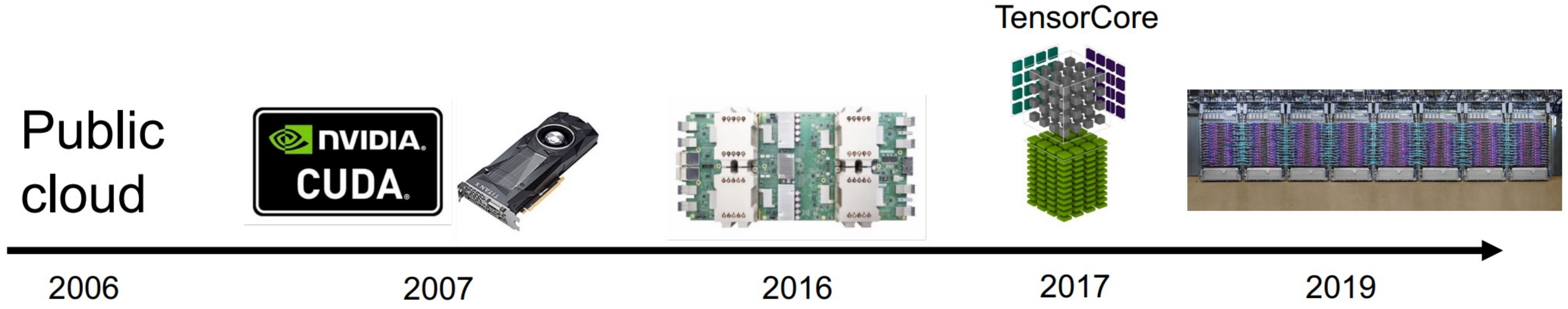
Perceptron Algorithm	Backprop	Support Vector Machine (SVM)	ConvNet	Gradient Boosting Machine (GBM)
1958	1986	1992	1998	1999

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

2000 – 2010: Arrival of Big Data

Data serves as fuel for machine learning models

2006 – Now: Compute and Scaling



Compute scaling

When three things come together and ready?

SVM ConvNet

Backprop GBM

ML Research

1958

WIKIPEDIA

The Free Encyclopedia

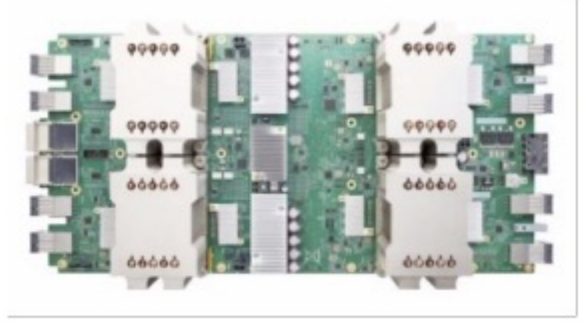
IMAGENET

Data

2000

Public cloud

NVIDIA CUDA



Compute

2007

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.

The bitter lesson by Richard Sutton

"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*."

A real example AlexNet 2012

Year 2012

Methods	**Data**	**Compute**
SGD Dropout ConvNet Initialization	IMAGENET 1M labeled images	Two GTX 580 Six days

W/o ML Systems

Researcher

ResNet ....

Transformer

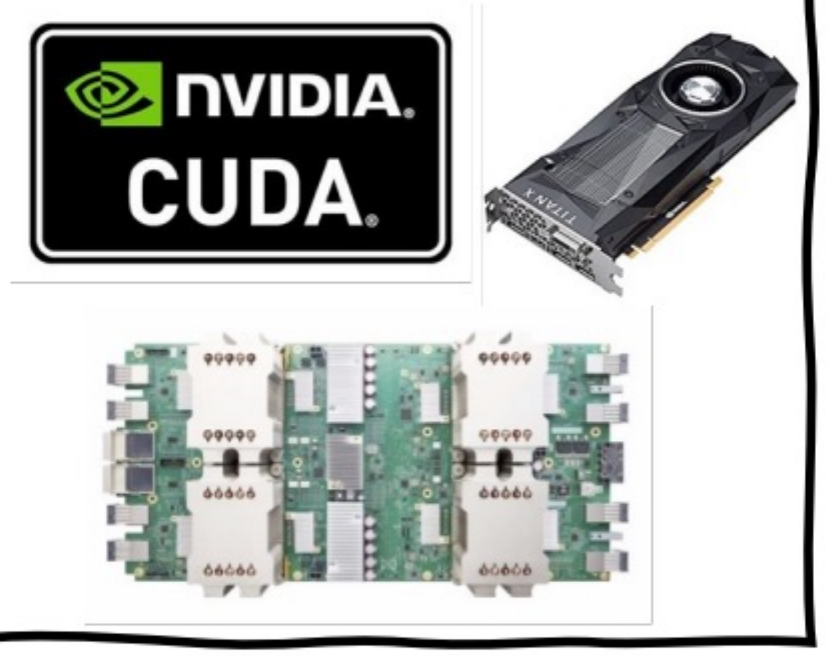
ML Research

44k lines of code

Six months

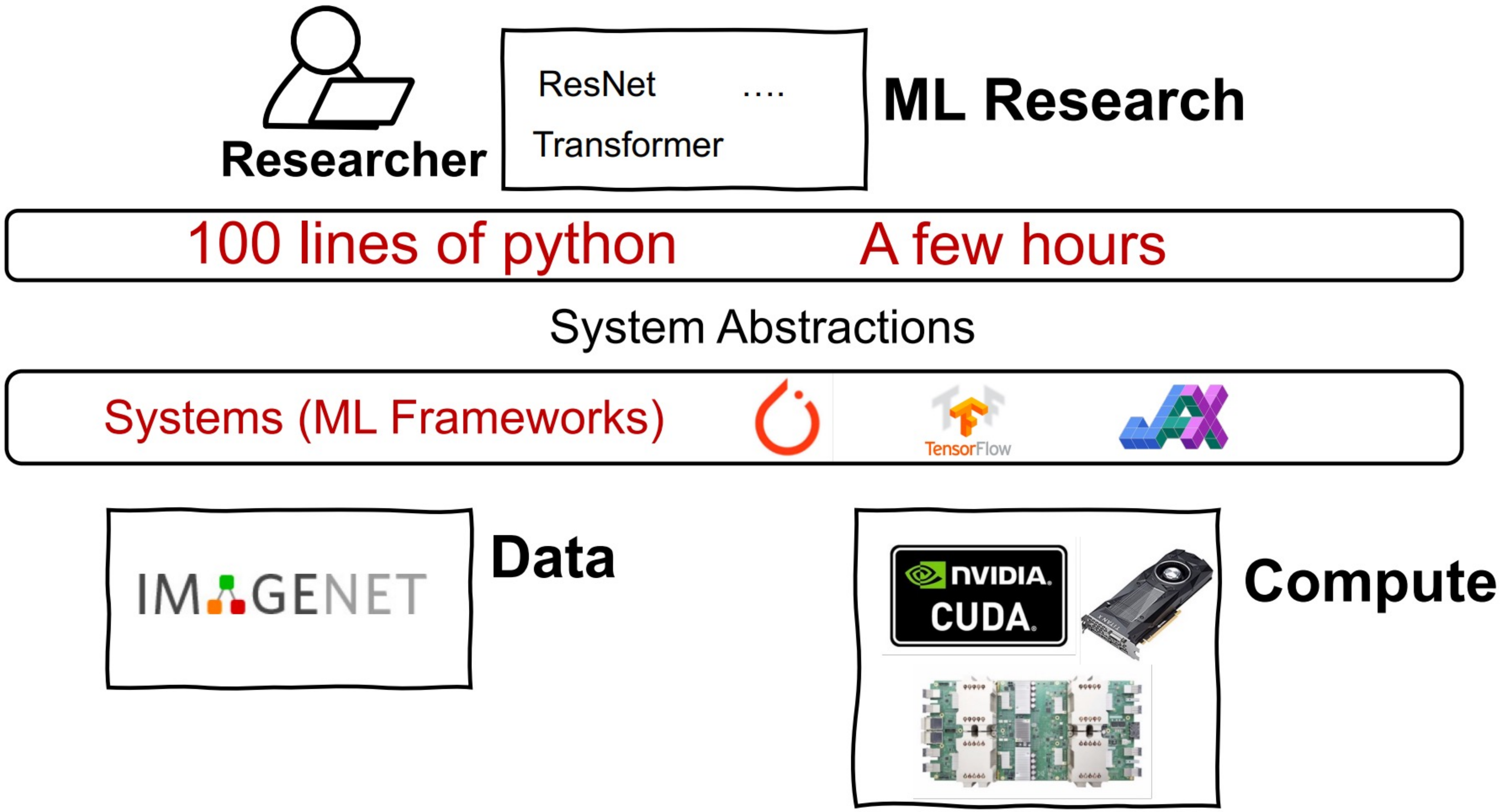
IMAGENET

Data



Compute

W/ ML Systems: accelerating R&D



W/ ML Systems: scale out

Researcher

ResNet ....

Transformer

ML Research

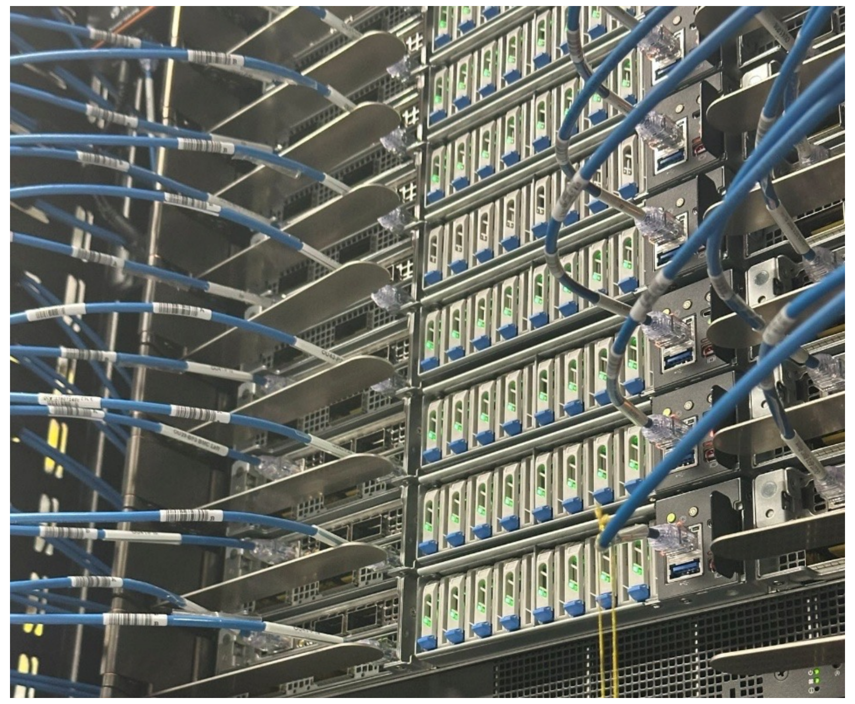
100 lines of python A few hours

System Abstractions

Systems (ML Frameworks)

TensorFlow

Data of the Entire Public Internet



Data center with 100K GPUs

Summary of Argument 1: ML System is an essential skill

- Accelerate ML research
- Bring ML up to scale
- Help deploy ML to everyone
- Enable ML <-> system codesign

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 better learning efficiency, followed by 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 researchers, specialize the implementation to target HW platform to reduce latency

ML-System way of Thinking

Option Step 0: develop a new hardware for the this task

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

1. Collect more data
2. Develop models architectures and algorithms that are able to squeeze the last bit of performance of the available hardware
3. Study data-model scaling law, scale it up using many hardware, subject to compute budget
4. Under compute budget, automatically optimize the HPO

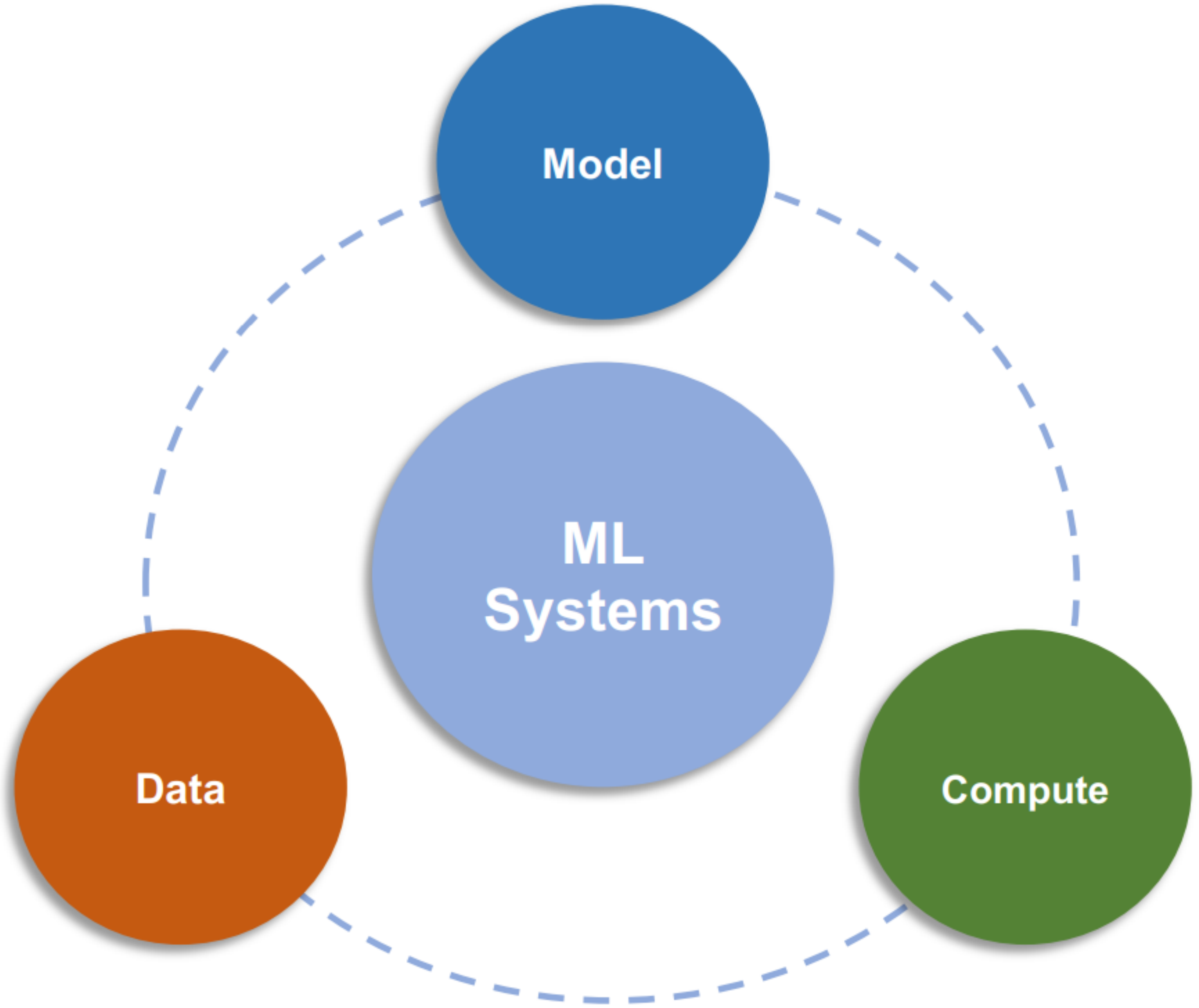
Repeat 1 -> 4

5. Streamline the entire process from development to deployment

Summary of Argument 2: Why Study MLSys

- ML-system way of thinking prepares ourselves to approach emerging problems and work in the area of machine learning engineering.

For PhD students: 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)

MLSys: The New Frontier of Machine Learning Systems

We will reveal Argument 3 soon

Today

Why study ML Systems

Course overview

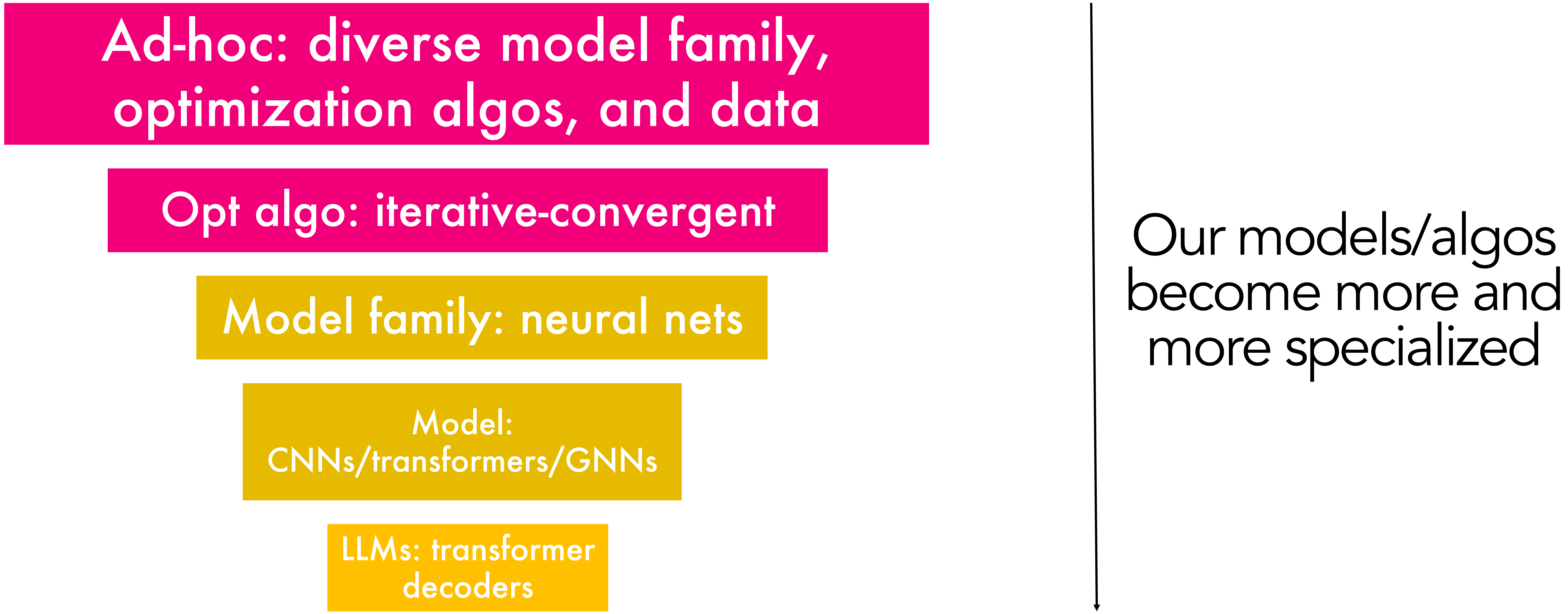
Logistics

Warm up (If time permits)

ML System history

System problems become clearer when problems are more spec

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



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 3 parts

- Basics: models, algorithms, and their compute representations
- Optimization, Parallelization, and Scaling
- LLMs, its related algorithms, optimization, parallelization, scaling, etc.

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: Optimization and Parallelization

- Single device optimization
  - Hardware acceleration
  - Compute/memory optimization
  - Graph optimization / fusion / quantization
  - Compilation
- Distributed ML Basics
  - Data parallelism, model parallelism
  - Inter-op parallelism, intra-op parallelism
  - Automatic parallelization

Part 3: LLMs

Transformers Large language model (LLM) basics

Scaling law: pretraining, test-time compute

LLM training, inference, serving, and their optimizations

LLM+X (X = agents, tool, RAG, Database, etc.)

What is this course not about?

Not an **OS/Distributed system/networking** 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, memory management?
- How networking works: packet delivery, protocols?

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 and calculus
- 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 it scope, a lot of interesting topics

This course will not cover:

- **ML for systems**: learned data base index, learned networking, etc.
- **ML Hardware design:** this is a software course
- **Other topics:** Federated learning, ML energy efficiency, system security in ML

Contrast with similar course offerings in UCSD

Vs. Previous years of CSE 234: Data system for machine learning

- Previous CSE 234 is more data-centric (more data component than ML components)
- This course is ML-centric and system-centric
- This course is more cutting-edge: latest technologies up-to-date

Suggested Textbooks

This is a fast-evolving field hence NO TEXTBOOK

We will read a lot of latest papers.

But there are a lot of online materials:

- Deep learning book by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
- Dive into deep learning: by Mu Li, Alex Smola, Aston Zhang
- 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

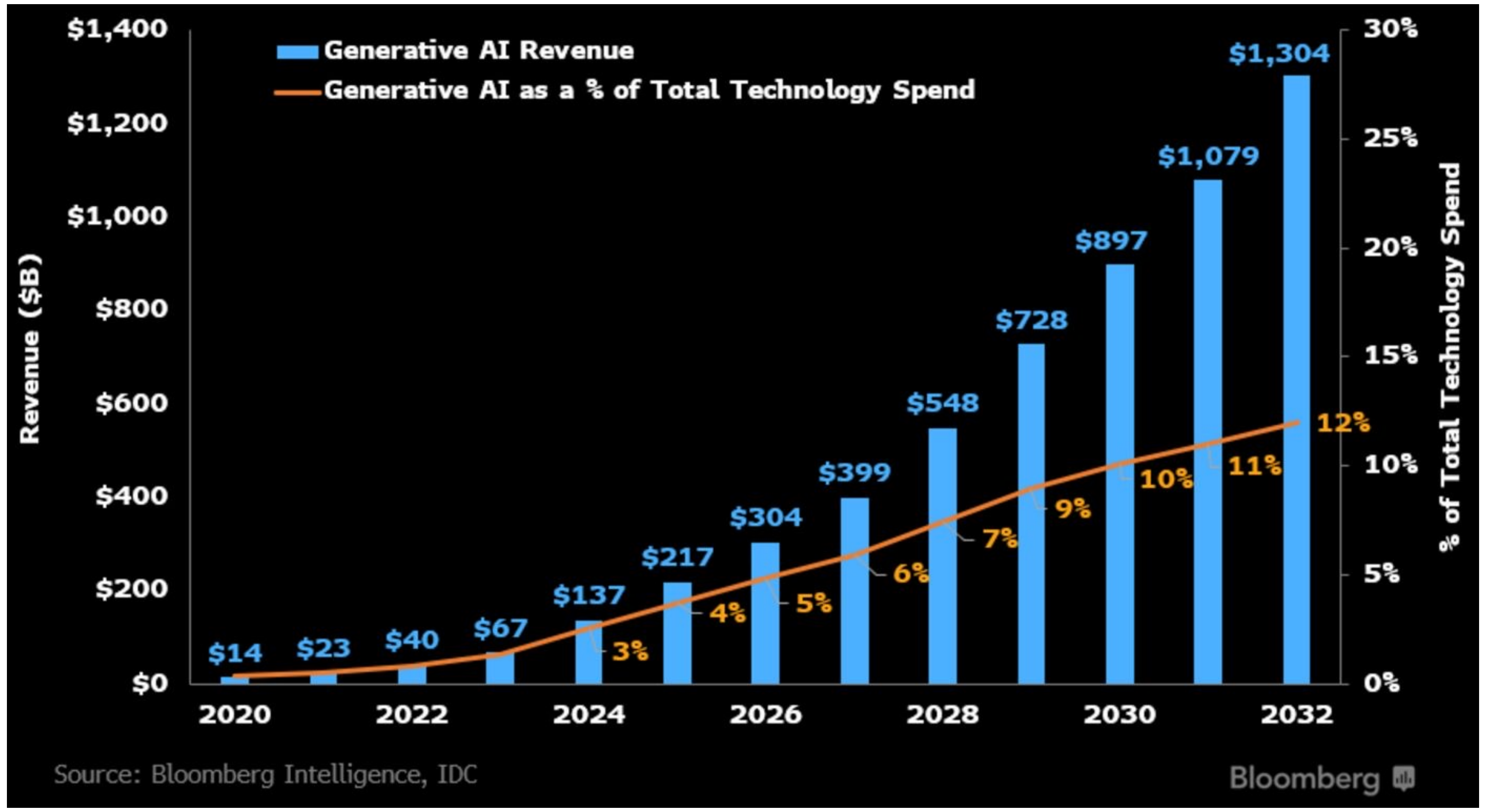
… ground all you learned in the context of LLMs, understand the L of **L**LM, how it is optimized, scaled, trained, served.

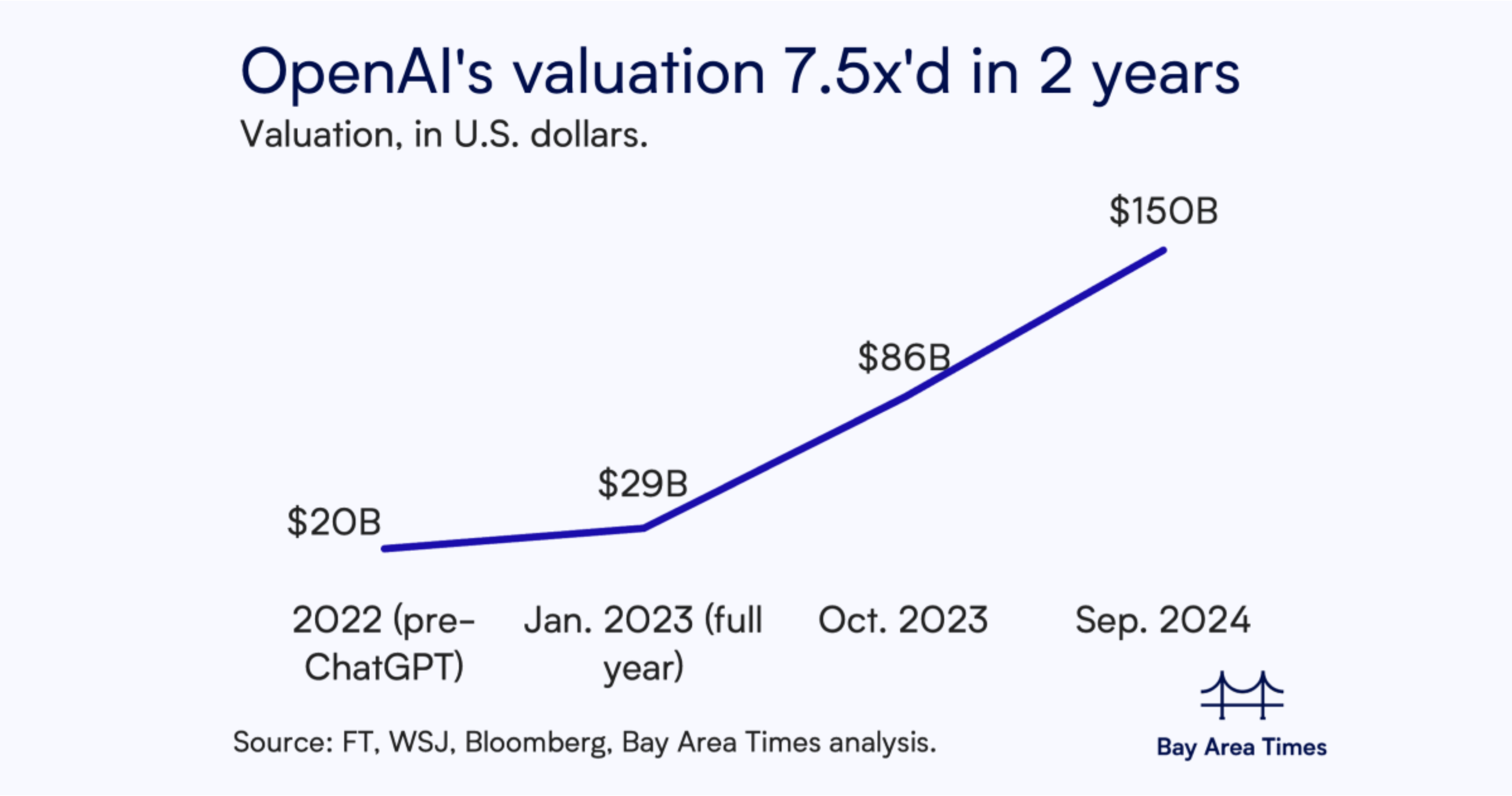
… Have fun

Here comes our Argument 3: Economical Reason

... and a more practical outcome (hopefully):
$$$

Global Picture: AI Industry





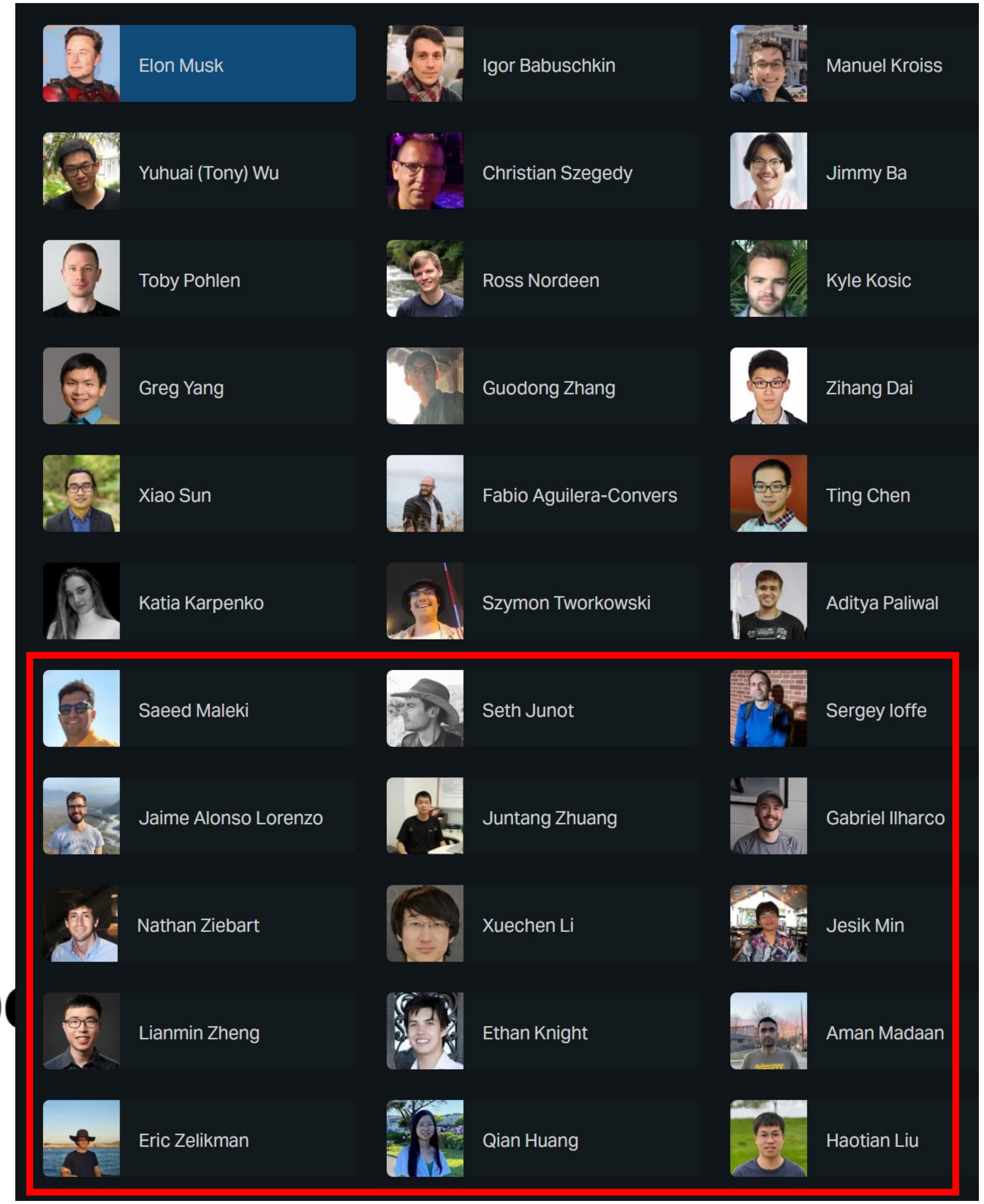
What others are doing

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 Op competitor, for $1.3B



Real Profiles I know

CMU Ph.D. Grad 2020
Area: Systems
First Employer: Google
Package: ~400K

Berkeley Ph.D. Grad 2024
Area: LLM Systems
First Employer: OpenAI
Package: ~1.7M

The Ultimate Outcome

Develop the capability of reasoning and understanding the technological trends and selecting which area to bet your next career on!

Questions?

Course website

CSE 234

- Home
- Syllabus
- Assignments
- Schedule Overview
- Resources
- FAQs
- Staff

Search CSE 234

Office Hours | Canvas | Piazza | GradeScope

CSE 234: Data Systems for Machine Learning

Instructor: Hao Zhang, UC San Diego, Winter 2025

Toggle Dark Mode

Announcements

Week 1 Announcements

Jan 7 · 0 min read

- Welcome to the Winter 2025 offering of CSE 234: Data Systems for Machine Learning!
- We're excited to work with you throughout the quarter, please pay attention to this page for each week's updates.
- In this year offering, we will **focus more on ML systems and LLMs than on database**.
- Check out the tentative schedule.
- The first lecture starts on Jan 7th, 6:30 pm at SOLIS 107.

Week 1

Jan 7: 1 Introduction — Slides • Recording

SURVEY Beginning of Quarter Survey

READINGS **(Due 1/14)**

- **Required:** 1.1 - MLSys : Intro, 1.2 - DNN
- **Optional:** 1.3 - Petuum, 1.4 - Systems Challenges for AI

Jan 9: 2 Basics: Modern DL, computational graph, autodiff, frameworks - 1 — Slides • Recording

https://hao-ai-lab.github.io/cse234-w25/

TAs

Abhilash Shankarampeta (ashankarampeta@ucsd.edu)

MS @ HDSI

Experience: Reasoning and efficient ML

OH: Every Monday 4 – 5 pm @ HDSI 437 and Zoom

Daniel Zhao(djzhao@ucsd.edu)

MS @ CSE

Experience: efficient ML and non-traditional NLP (e.g. music)

OH: Every Wednesday 1 – 2pm at B260A and Zoom

TAs

Junda Chen (juc049@ucsd.edu)

PhD @ CSE

Experience: efficient LLM inference engine and agent systems

OH: Every Tuesday 1 – 2 pm @ B240A and Zoom

Ruiyi Zhang (ruz048@ucsd.edu)

PhD @ ECE

OH: TBD

Components and Grading

- 3 Programming Assignments: **45%** (15% + 15% + 15%)
  - 5 late days in total

Exams

- No Midterm

Final Exam: **37%**

- **Reading summary: 10%, 10 readings, 20 - 40 pages per reading**

No late days.

- Scribe Duties: 8%
- Extra Credit: **6%**

Grading Scheme (grade is the better of the two)

Grade	Absolute Cutoff (>=)	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 (>=)	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, Collective communication
4	graph and memory optimizations
4	Guest lecture: ML compilers
5	Data and model parallelism, auto-parallelization
6	Transformers, LLMs, MoE
6	Guest lecture: LLM pretraining and open science
7	LLM training: flash attention, quantization
8	LLM inference and serving: paged attention, continuous batching, speculative decoding
9	Guest lecture: fast inference
9	Scaling Law, test-time compute, reasoning
10	LLM + X (X = RAG, search, multi-modality, etc.)
10	Guest lecture: LLM, tool use, and agents
10	Final exam reviews
11	Final exam

🌎 https://hao-ai-lab.github.io/cse234-w25/

Lectures

Hao's lecture: high encouraged to attend

In person unless due to travel or weather

Will review some MQAs in the end

Guest lectures: You **must** attend (mostly on Zoom)

About 4, from inventors of key techniques covered in class

TAs will track the attendance

TA recitations: Arrange on demand, using the "discussion" slot

e.g., before-exam recitations

Programming Assignments

- Three newly designed PAs
- Will be based on PyTorch / Huggingface/CUDA/Triton
- Topics

  Implement computational operator/graphs, autodif, and optimize them

  Train it, debug correctness, and benchmark performance

  Scale it up using parallelism

  Serve them and meet SLOs
- 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

Hao's notes: if you really want to learn something practical, PA will be the best source of this course

Reading Summary

Required reading:

The instructor team will select important papers (workload: 20 – 40 pages / week).

One reading summary per week, submit your reading by next week Tuesday midnight.

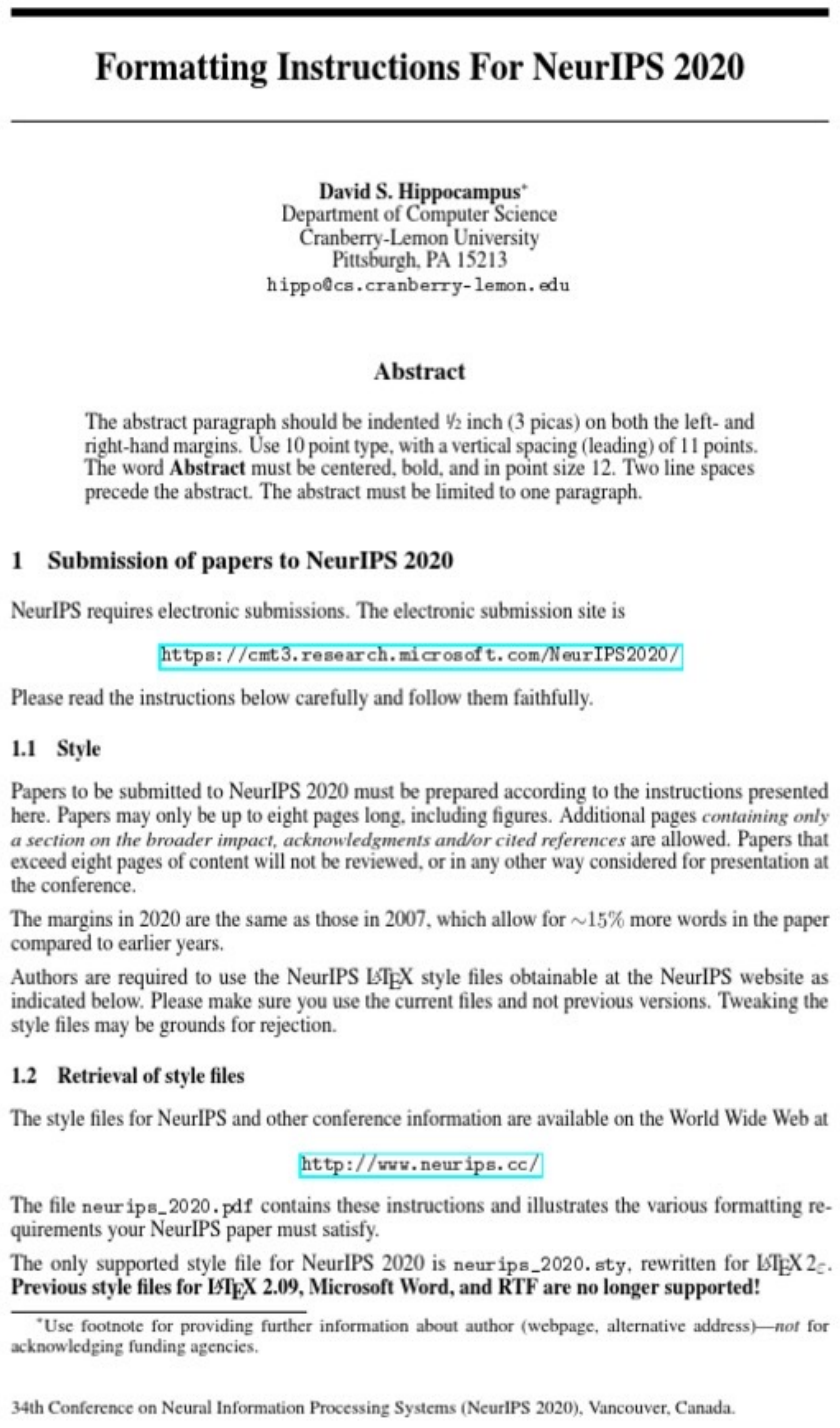
Your reading summary should focus on high-level ideas, and should be >= 3 pages of the Neurips format.

Optional reading:

Other 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 ½ 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 ~15% more words in the paper compared to earlier years.

Authors are required to use the NeurIPS LaTeX 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 LaTeX 2_ε. **Previous style files for LaTeX 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.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Scribe Duties

Sign up your scribe duty here:

https://docs.google.com/spreadsheets/d/18zlX-zmFu5cMR4M-xkWhIQOlnYPZrEXZj-TaO5MlIY0/edit?usp=sharing

You should

- Scribe with as many details as possible
- Collaborate with other scribers
- Submit PRs to course website repo
- Reviewed and maybe iterated with the TA

Exams

- No Mid-term
- In-person Final exam (37%)
- All MCQs
- You can bring as many books/cheat sheets/paper you want
- No phone/laptop/Internet/ChatGPT
- Data: March 18, 2025, 7-10 pm

Exams

Hao’s lectures will feature some MCQs (that may appear in final exams) every week, so make sure to attend lectures or watch recordings.

Karma Points

- Participation: lectures / piazza
- Guest lecture: ask hard questions to challenge our guests ☺
- Completing course surveys and evaluation: it helps me, helps TAs and help yourself

MCQ Example: Who originally developed PyTorch?

OpenAI

Google DeepMind

aws

Meta AI

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!

TODOs after Today's lecture

1. Make sure you are enrolled with Piazza, Canvas, Gradesope
2. Check all contents of course website (Schedule, Syllabus, Exam time)
3. Signup your scribe duty
4. Finish Start-of-quarter survey
5. Start the reading of week 1

Waitlisted Students

- This classroom can accommodate 196 students
- Currently we get 300+ enrollment requests. Students are enrolled using a FCFS scheduler
- CSE will continue to enroll students when previous students dropped
- For most CSE courses: half of enrolled students will drop after week 2
- Fill this form if you are absolutely sure you will take this course: https://tinyurl.com/cse234waitlist

Questions?