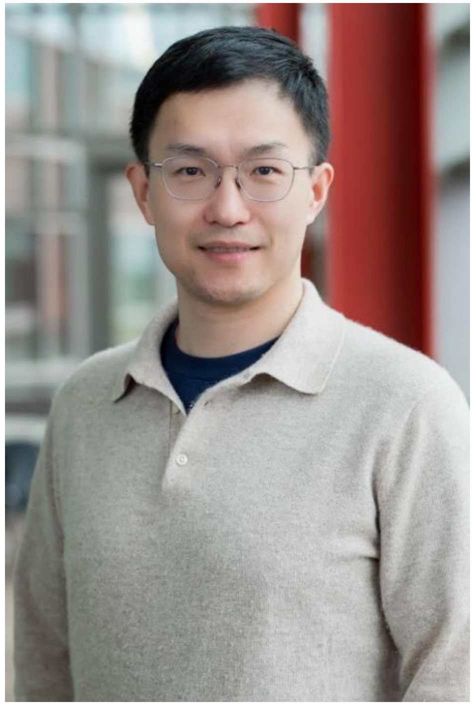
CS 498: Machine Learning System
Spring 2025

Minjia Zhang

The Grainger College of Engineering

About the Instructor



Minjia Zhang

- Now: Assistant Professor at UIUC
  - Affiliated with ECE and NCSA, UIUC
  - https://siebelschool.illinois.edu/about/people/faculty/minjiaz
- Principal researcher at Microsoft (2016-2023)
  - Project: DeepSpeed, Megatron-DeepSpeed

My Lab: Supercomputing System and AI Lab (SSAIL)

Research area: Systems + Machine Learning

Recent topics:

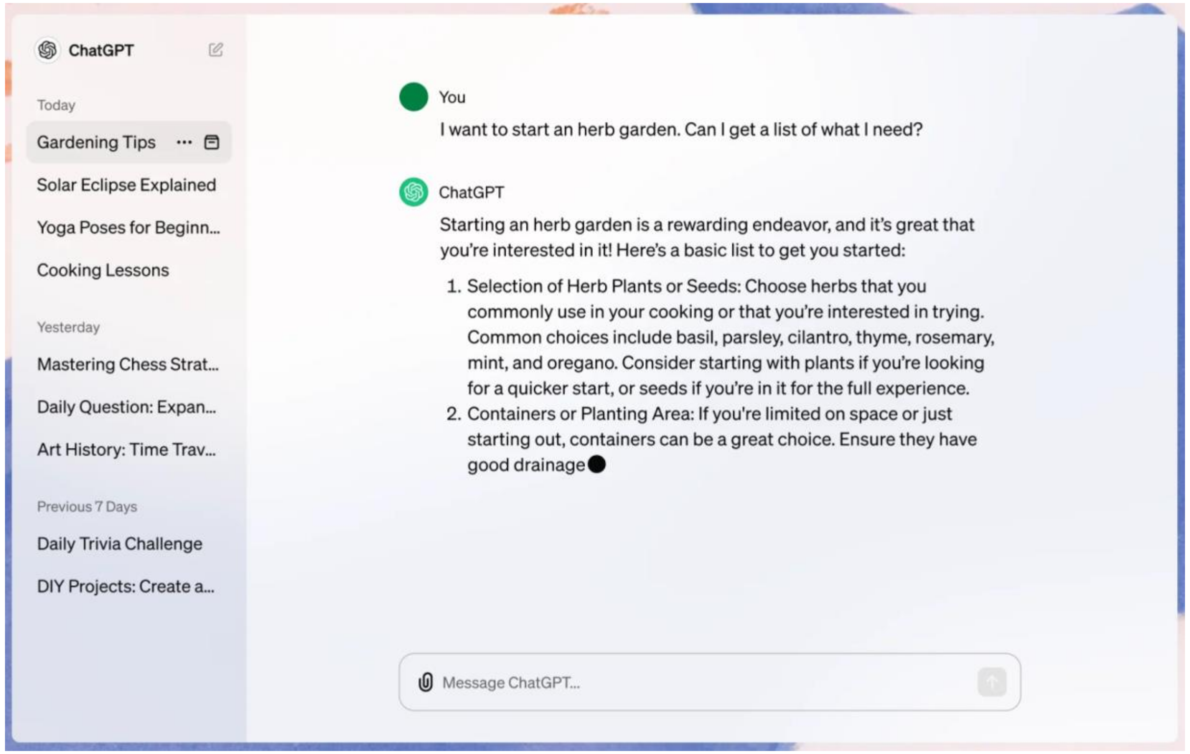
- Efficient machine learning systems (training/inference on parallel/distributed/heterogeneous hardware)
- Effective efficiency algorithms (model compression, data efficiency, parameter-efficient tuning, etc.)
- Large-scale DL/AI applications (RAG, Image/Video Generation, VLM, DLRM, Vector DB, etc)

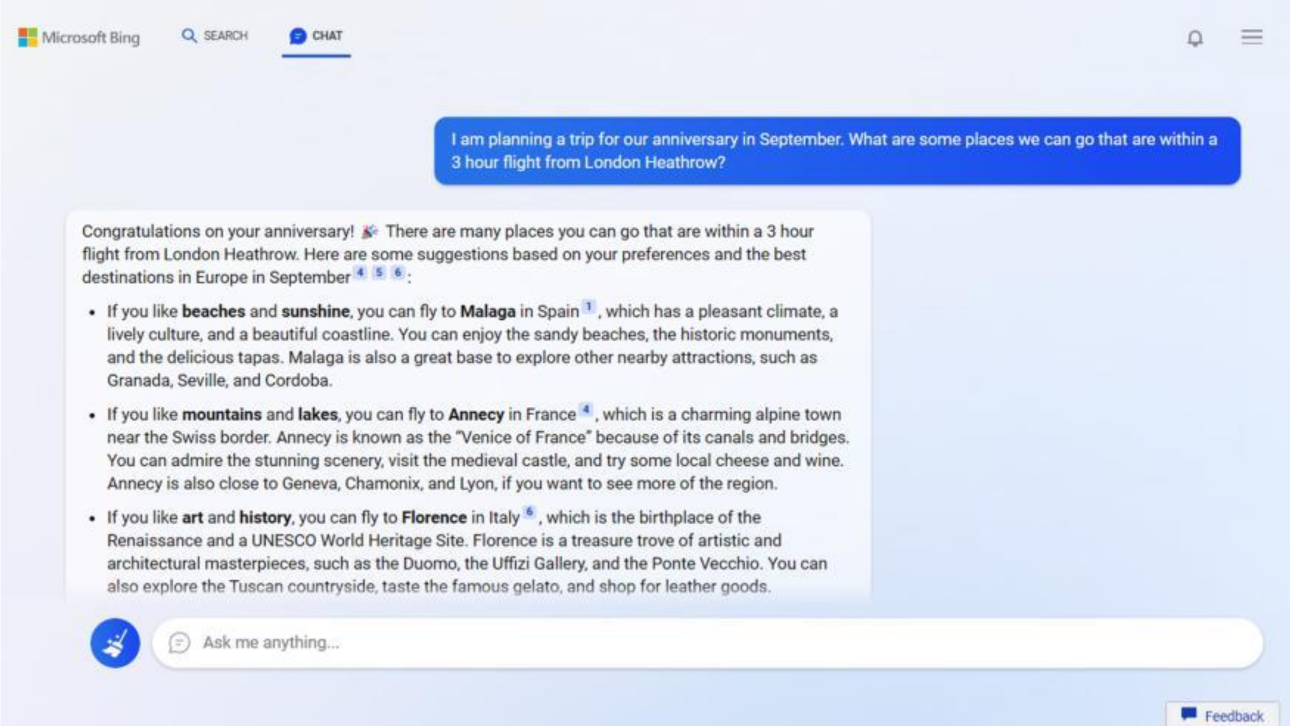
Today

- **Why study ML Systems?**
- Course overview
- Logistics

The Large Language Model Revolution

ChatGPT/Search





[ChatGPT: Optimizing Language Models for Dialogue]

Continuation and Generation

```python
"""
Python 3
Get the current value of a Bitcoin in US dollars using the bitcoincharts api
"""

import requests
import json

def get_bitcoin_price():
    url = 'http://api.bitcoincharts.com/v1/weighted_prices.json'
    response = requests.get(url)
    data = json.loads(response.text)
    return data['USD']['7d']

if __name__ == '__main__':
    print(get_bitcoin_price())
```

[Suggest code and entire function in your editor – Github/OpenAI Codex]

Image Generation from Text

TEXT DESCRIPTION

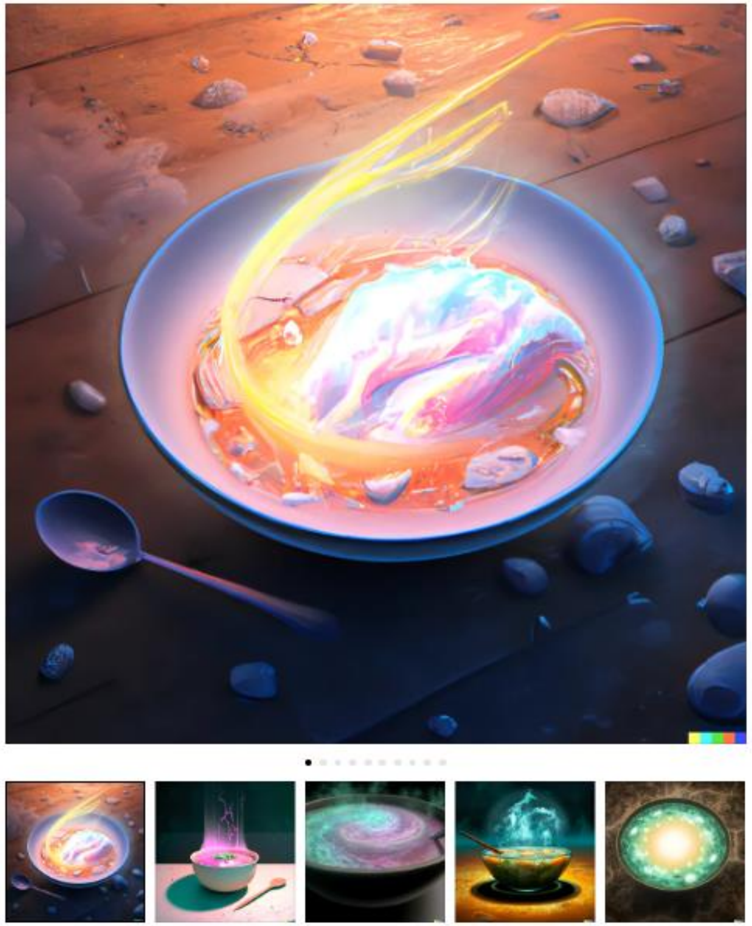
An astronaut Teddy bears A bowl of soup

that is a portal to another dimension that looks like a monster as a planet in the universe

as digital art in the style of Basquiat drawn on a cave wall

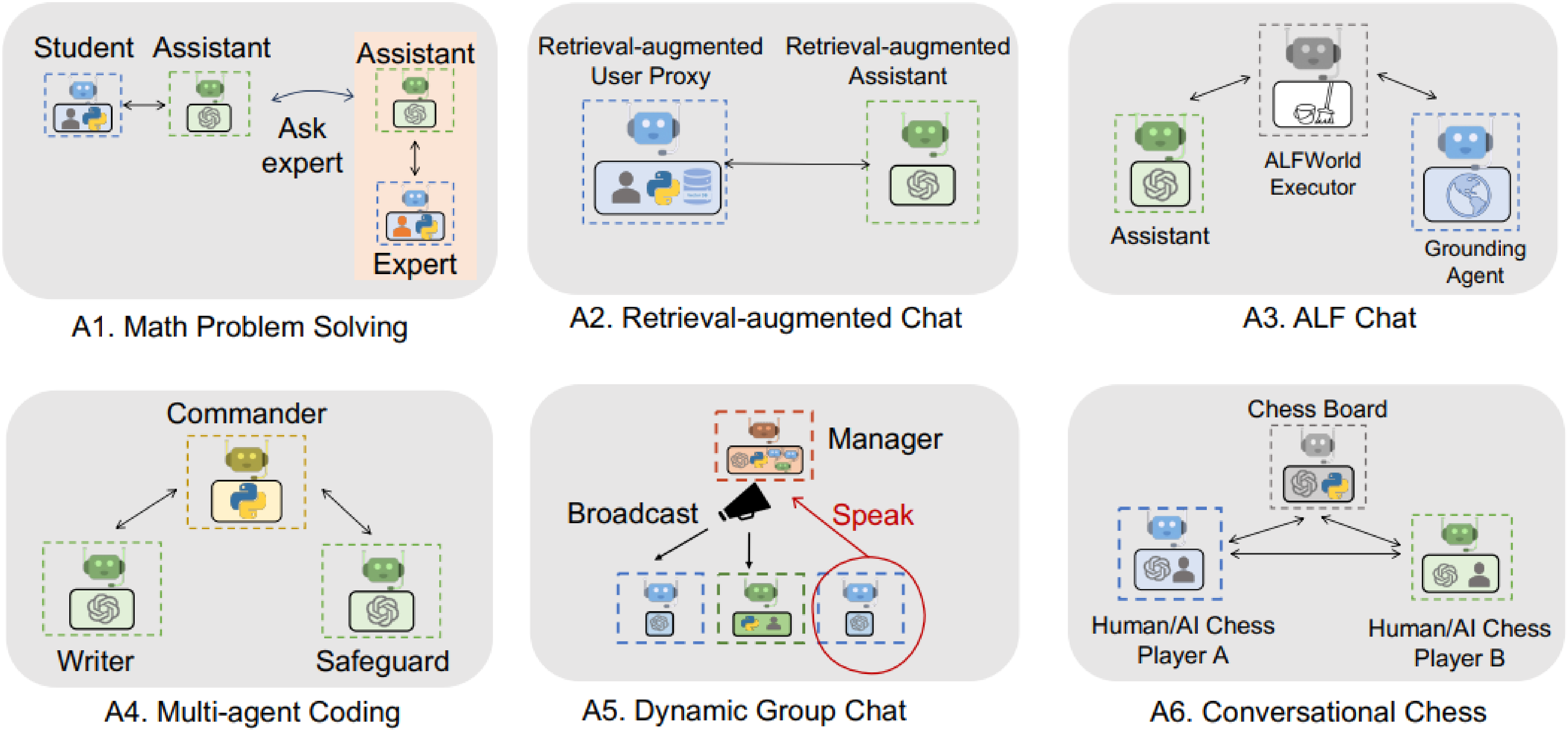
→

DALL·E 2



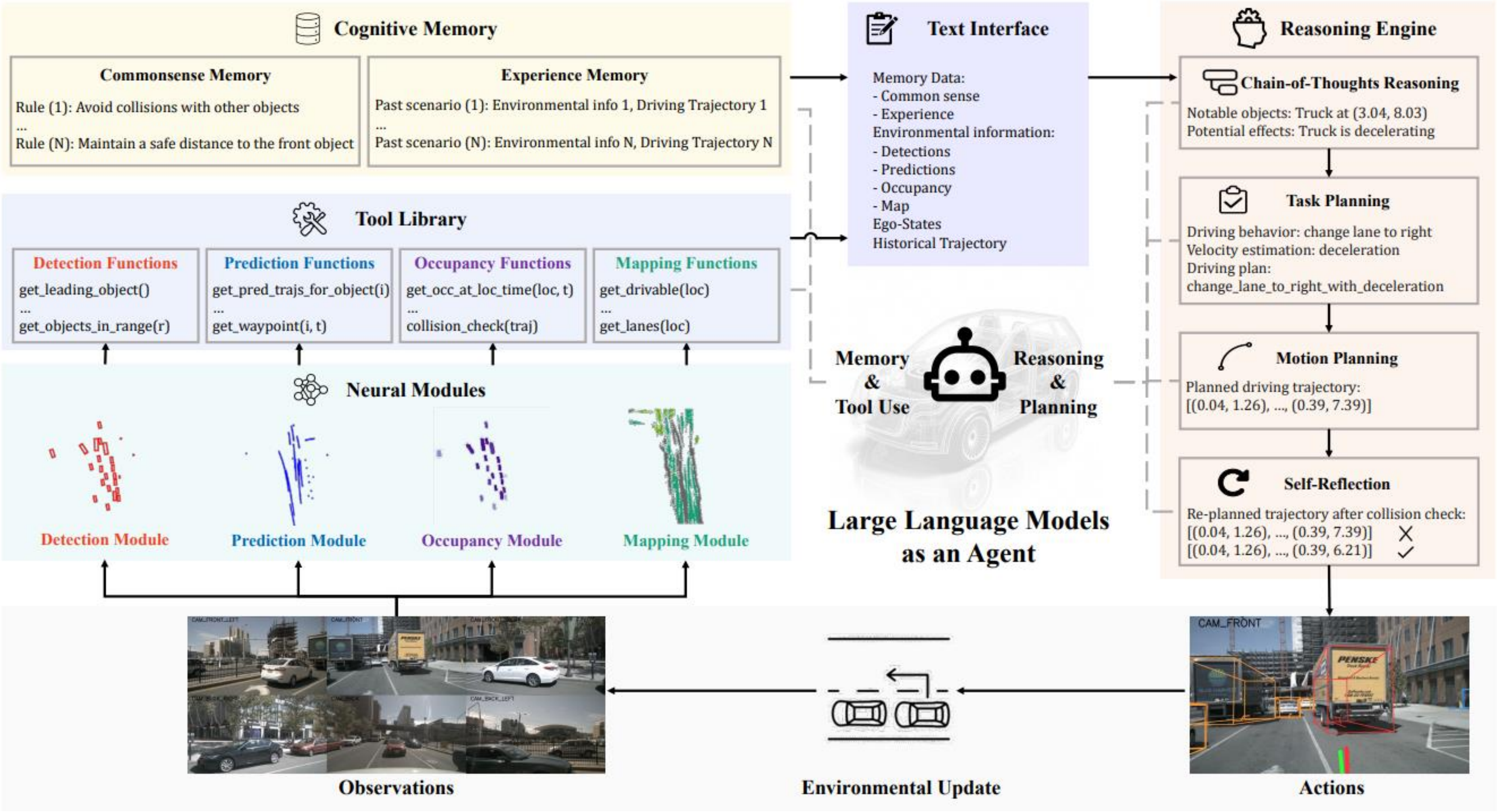
[DALL·E: Creating Images from Text - OpenAI]

Agentic AI



[AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation]

Autonomous Driving



[A Language Agent for Autonomous Driving](#)

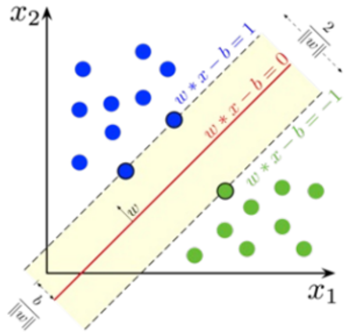
Robotics

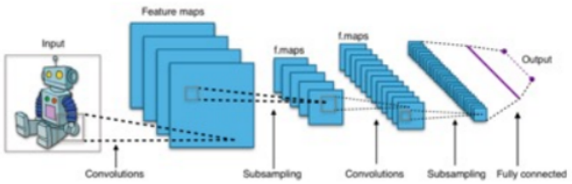
[Using Generative AI to Enable Robots to Reason and Act]

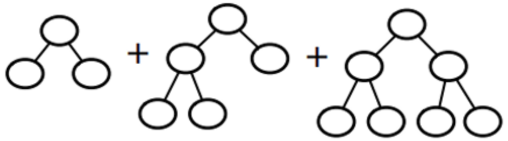
How does this Happen?

A key ingredient: ML Systems

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**

Source: https://mlsyscourse.org/slides/01-course-introduction.pdf

2000 – 2010: Arrival of Big Data

WIKIPEDIA
The Free Encyclopedia

flickr

MTurk

kaggle

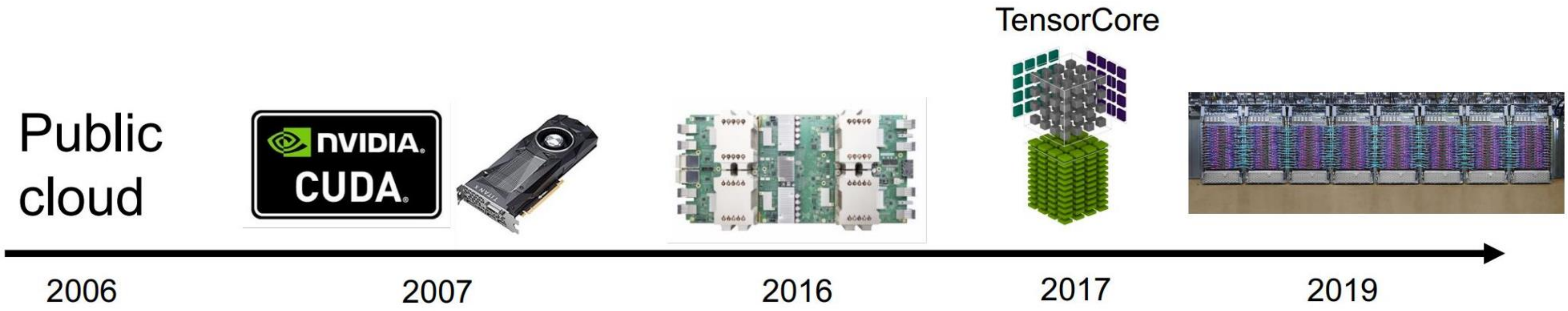
IMAGENET

2001 2004 2005 2009 2010

Data serves as fuel for machine learning models

Source: https://mlsyscourse.org/slides/01-course-introduction.pdf

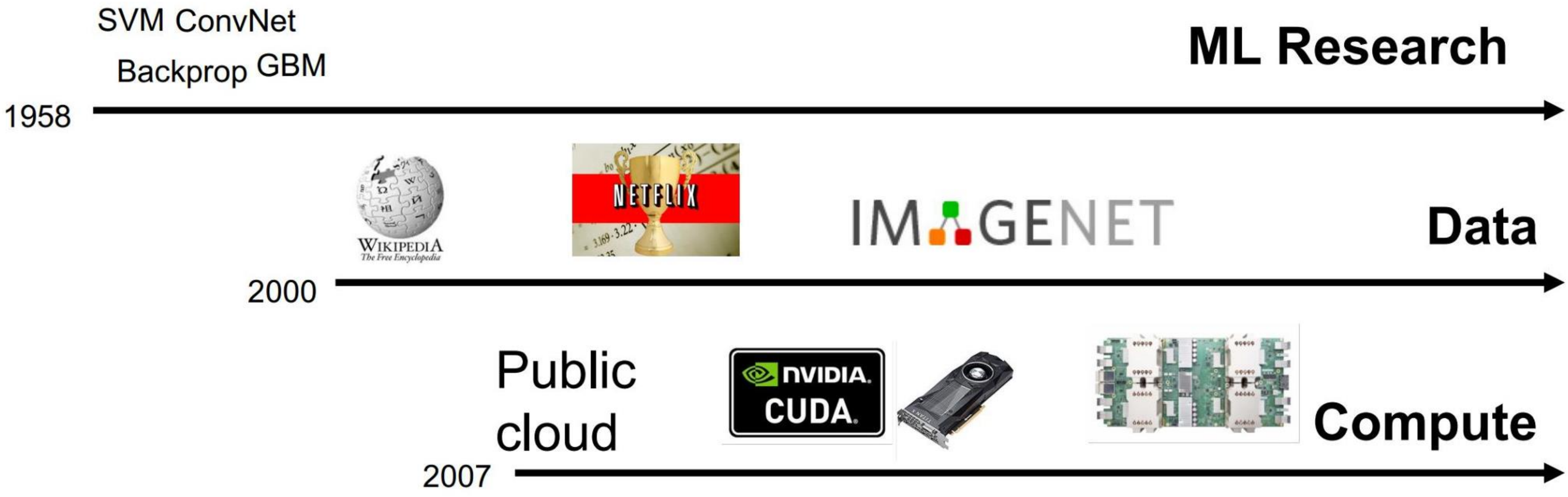
2006 – Now: Compute and Scaling



Compute scaling

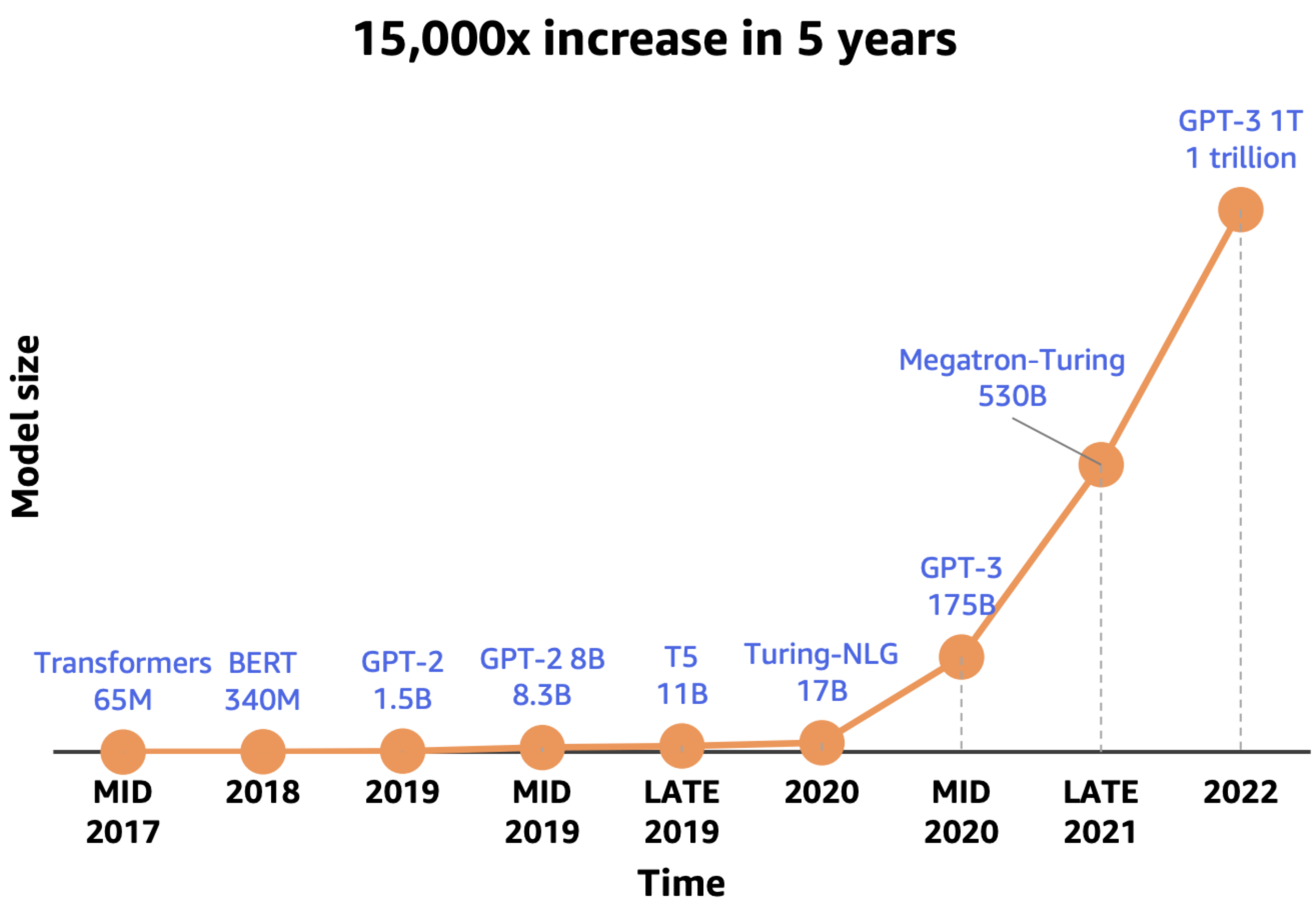
Source: https://mlsyscourse.org/slides/01-course-introduction.pdf

When three things come together and ready



Source: https://mlsyscourse.org/slides/01-course-introduction.pdf

Evolution of DL Models



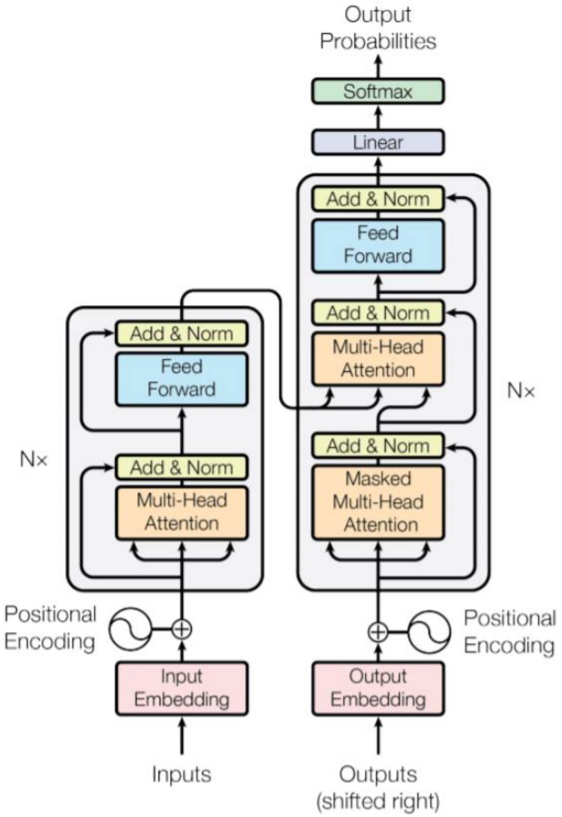
Larger models → better accuracy

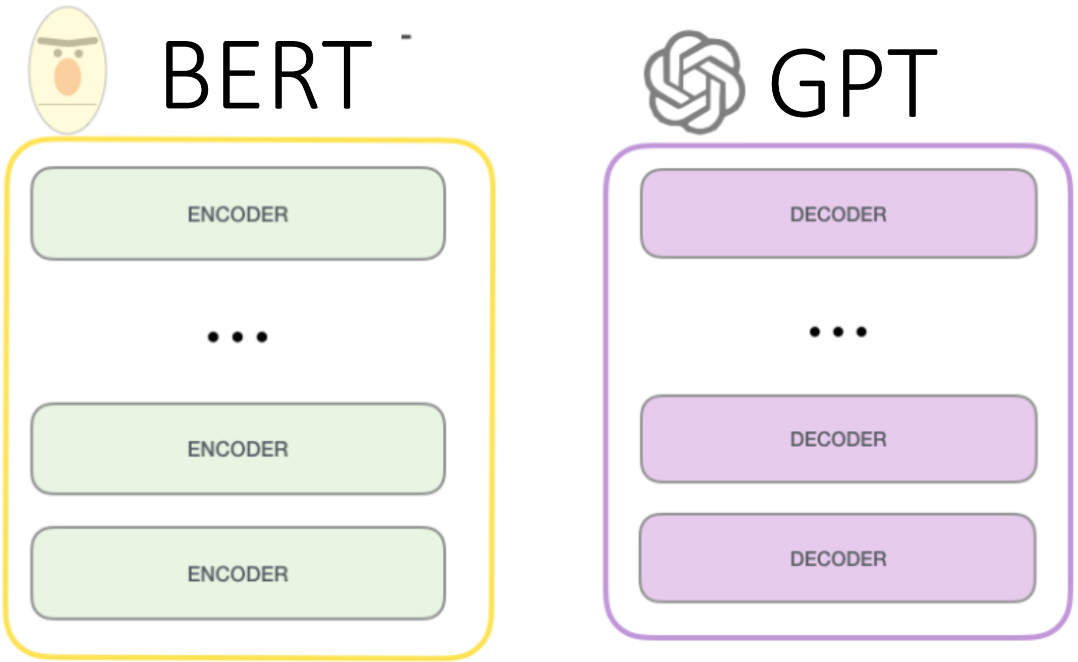
Model size is still growing

Not reached the accuracy limit yet

More compute-efficient to train larger models than smaller ones to same accuracy

Transformers for Language Modeling





Attention Is All You Need, NeurIPS 2017

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, ACL 2019

Language Models are Few-Shot Learners, NeurIPS 2020

LLMs are Impressively Scaling!

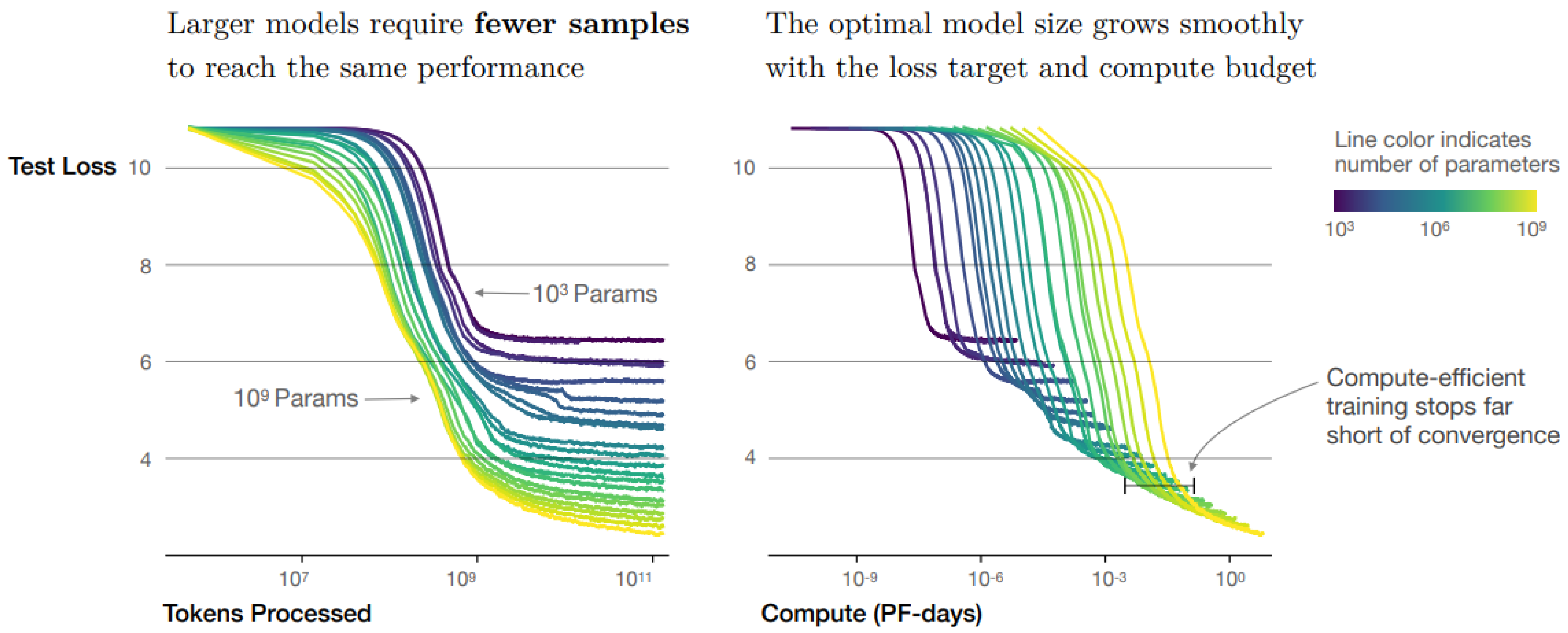


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Scaling Laws for Neural Language Models, OpenAI, 2020

ML System Challenges

- Too slow to train high-quality models on massive data
  - More hardware ≠ higher throughput, bigger model
  - Higher throughput ≠ better accuracy, faster convergence, lower cost
  - Better techniques ≠ handy to use
- Slow and expensive to deploy the models

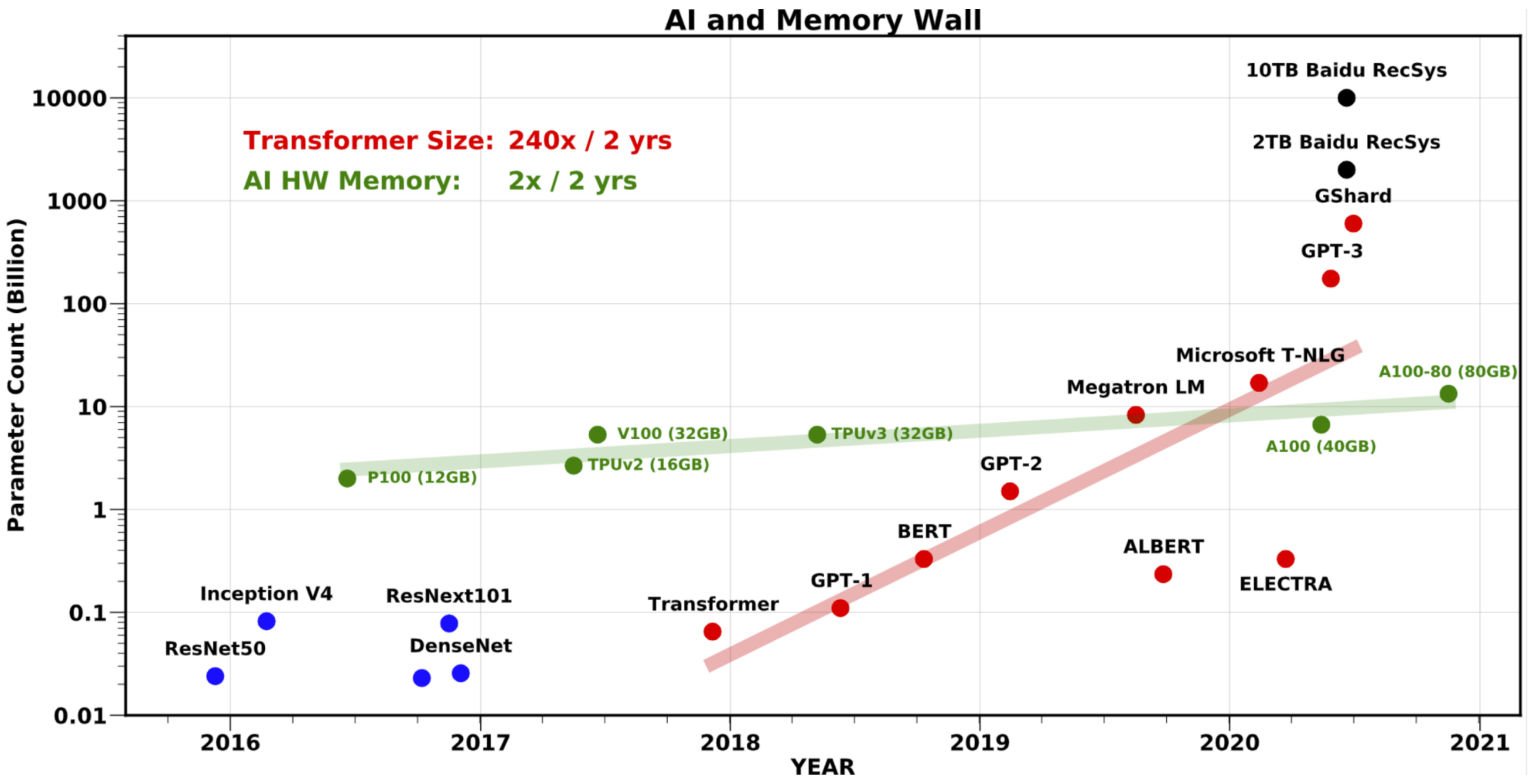
ML System Desired Capabilities

Efficiency: Efficient use of hardware for high scalability and throughput

Effectiveness: High accuracy and fast convergence, lowering cost

Easy to use: Improve development productivity of model scientists

DNN Training Hits the Memory Wall



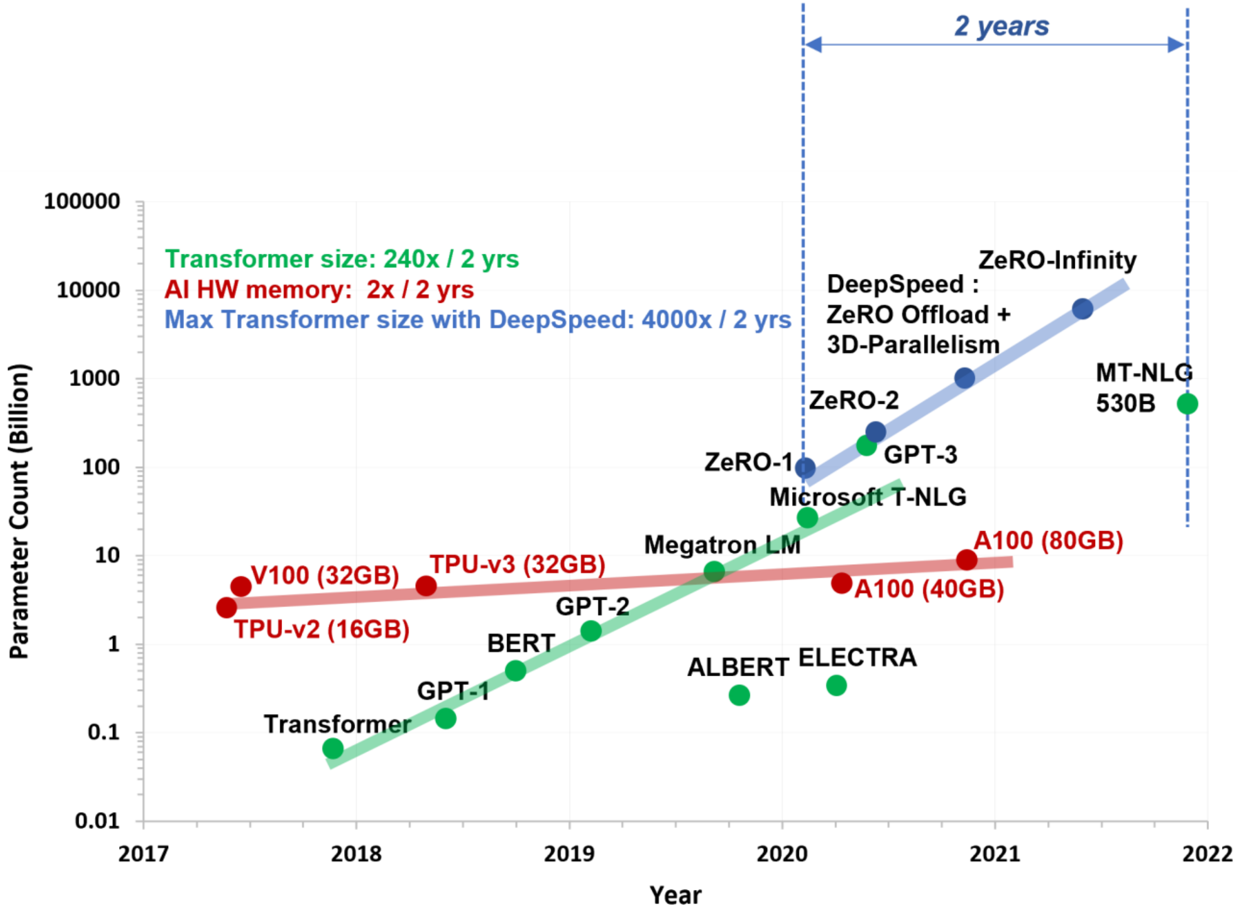
*AI and Memory Wall. (This blogpost has been written in… | by Amir Gholami | riselab | Medium

ML System Research Reshape the Large Model Training Landscape

deepspeed

DeepSpeed Powered Massive Models:
- Z-code MoE **10B**
- Microsoft-Turing NLG **17B**
- GPT Neo-X **20B**
- Jurrasic-1 **178B**
- Big Science **200B** (ongoing)
- Megatron-Turing NLG **530B**

Key training technologies:
- ZeRO redundancy optimizer
- 3D parallelism
- Optimized CUDA/ROCm/CPU kernels
- Optimized communication libraries
- Mixed precision training
- Communication efficient Adam
- Sparse Attention
- Mixture of quantization
- Curriculum learning
- …



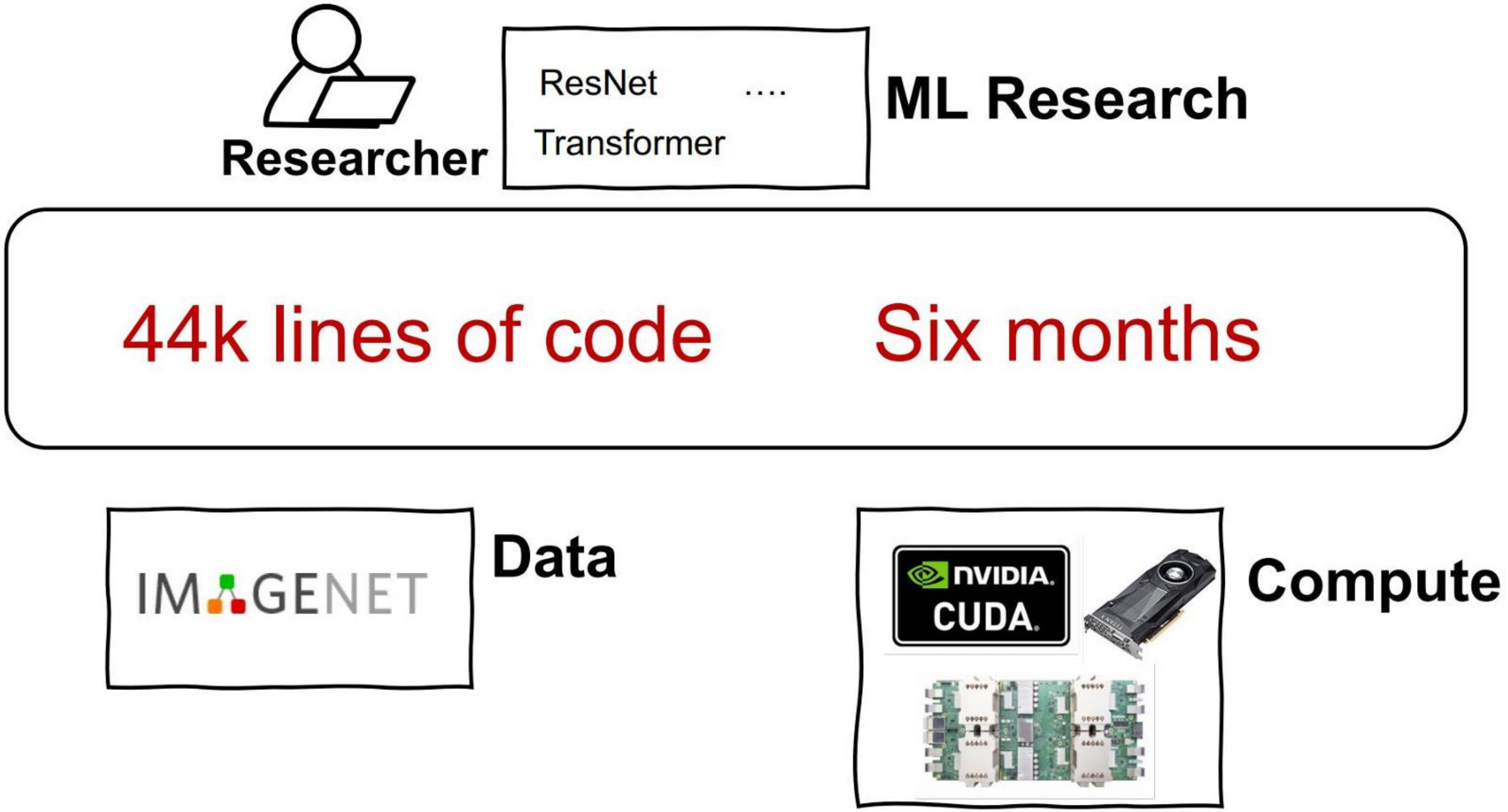
Another Example: AlexNet 2012

Year 2012

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

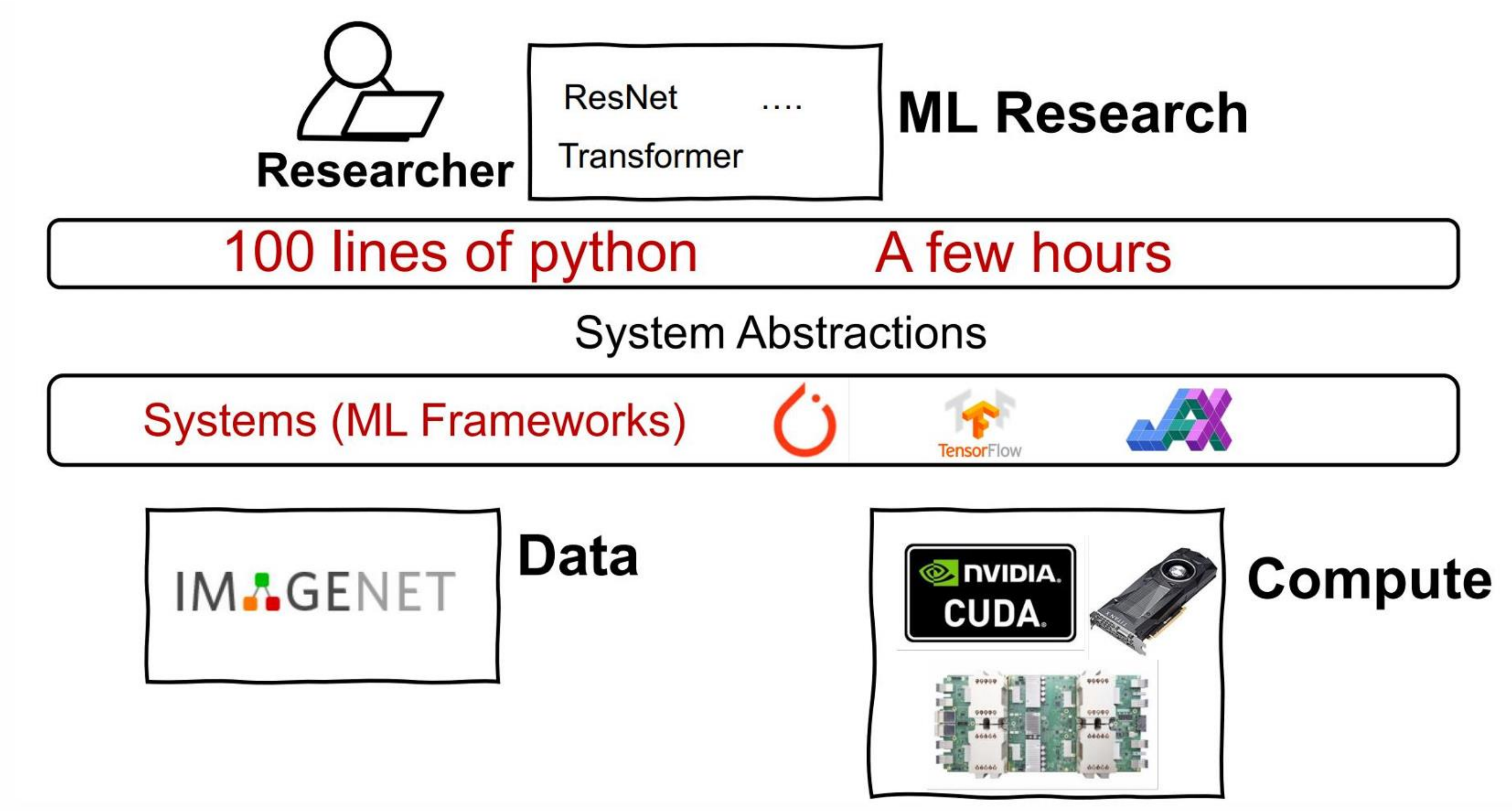
Source: https://mlsyscourse.org/slides/01-course-introduction.pdf

Without ML Systems



Source: https://mlsyscourse.org/slides/01-course-introduction.pdf

With ML Systems



Source: https://mlsyscourse.org/slides/01-course-introduction.pdf

ML Systems

- Enable new system capabilities to break the memory wall
- Accelerate ML research
- Reduce deployment cost
- Democratize AI to everyone
- ML ←→ System codesign
- …

- In summary: ML System is becoming an essential skill

Today

- Why study ML Systems?
- **Course overview**
- Logistics

ML System Stack

ML Applications	Computer Vision, Speech Recognition, Language Translation, Autonomous Driving, Recommender Systems, Fraud Detection, Advertising
ML Data Sets	ImageNet, COCO, VOC, KITTI, WMT
ML Models	ResNet, GoogLeNet, SqueezeNet, MobileNet, SSD, GNMT
ML Frameworks	Tensor-Flow, PyTorch, Caffe, MXNet, CNTK, Theano
Graph Formats	ONNX, NNEF
Graph Compilers	TVM, nGraph, Glow, XLA
Optimized Libraries	CUDA, MKL DNN, OpenBLAS, CuBLAS, Eigen
Operating Systems	Linux, Android, Windows, BSD/OS-X, RTOS
Hardware	GPU, CPU, TPU, NPU, DSP, FPGA, Accelerators

ML System Stack

ML Applications	Computer Vision	Speech Recognition	Language Translation	Autonomous Driving	Recommender Systems	Fraud Detection	Advertising
ML Data Sets	ImageNet	COCO	VOC	KITTI	WMT		
ML Models	ResNet	GoogLeNet	SqueezeNet	MobileNet	SSD	GNMT	
ML Frameworks	Tensor-Flow	PyTorch	Caffe	MXNet	CNTK	Theano	
Graph Formats	ONNX	NNEF					
Graph Compilers	TVM	nGraph	Glow	XLA			
Optimized Libraries	CUDA	MKL DNN	OpenBLAS	CuBLAS	Eigen		
Operating Systems	Linux	Android	Windows	BSD/OS-X	RTOS		
Hardware	GPU	CPU	TPU	NPU	DSP	FPGA	Accelerators

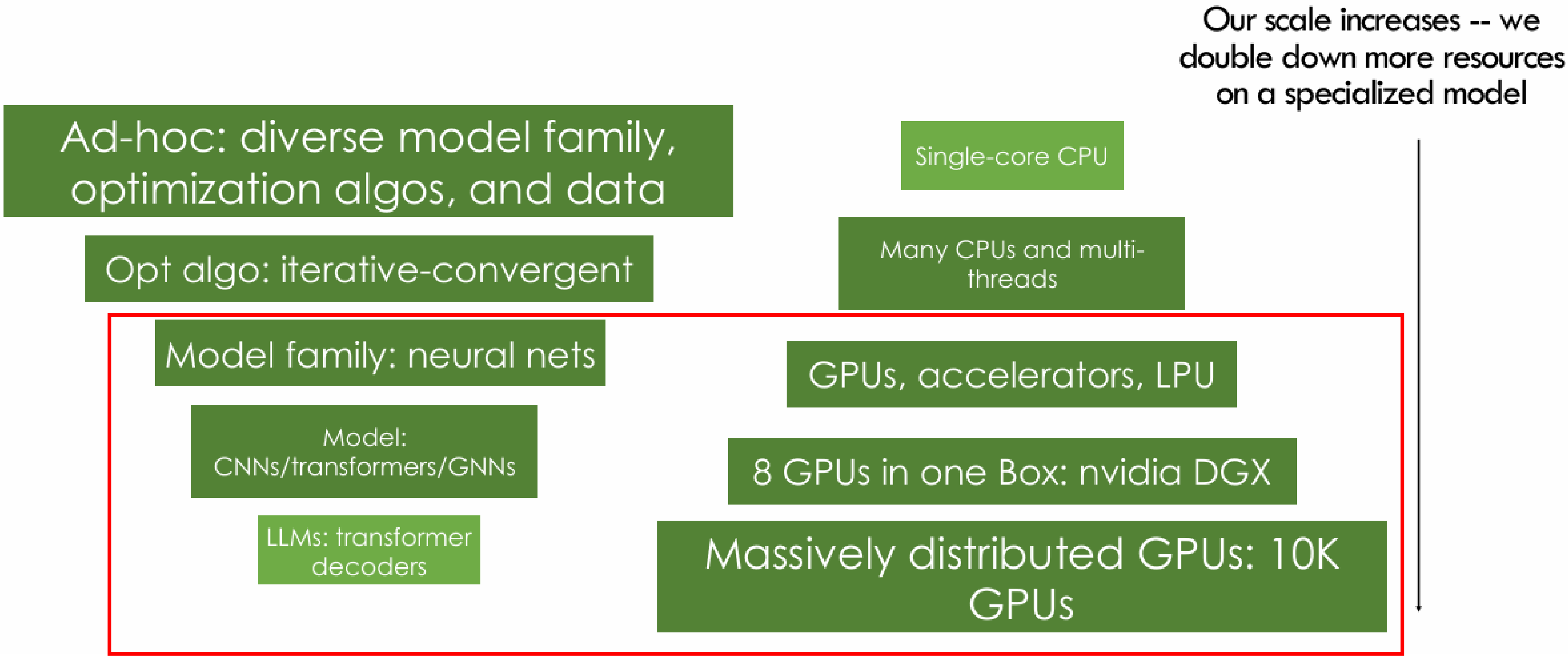
ML System at Scale

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

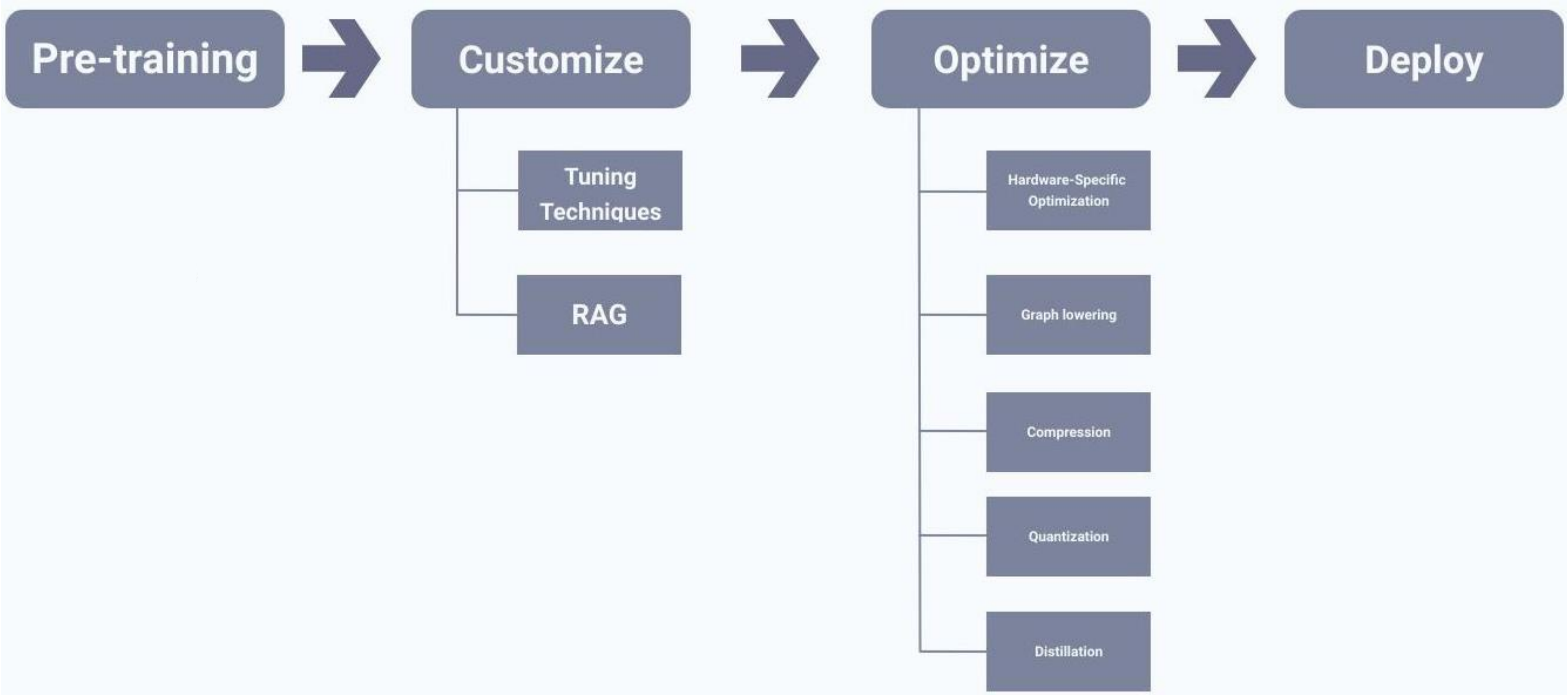
- Ad-hoc: diverse model family, optimization algos, and data
- Opt algo: iterative-convergent
- Model family: neural nets
- Model: CNNs/transformers/GNNs
- LLMs: transformer decoders

- Single-core CPU
- Many CPUs and multi-threads
- GPUs, accelerators, LPU
- 8 GPUs in one Box: nvidia DGX
- Massively distributed GPUs: 10K GPUs

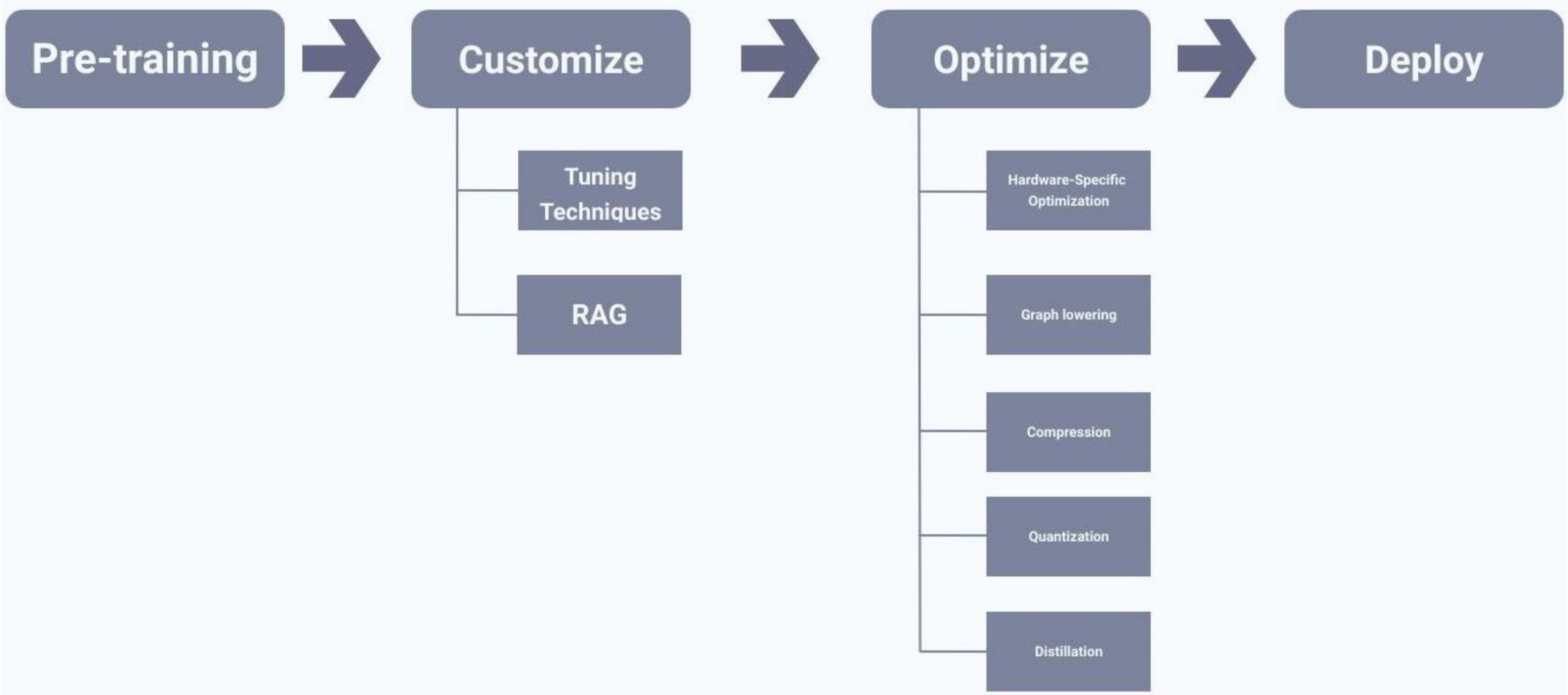
ML System at Scale



ML Workflow


Pre-training
Customize
Optimize
Deploy
Tuning Techniques
RAG
Hardware-Specific Optimization
Graph lowering
Compression
Quantization
Distillation

ML Workflow



ML system challenges in all stages!

Course organization

- Distributed ML, ML parallelization
- Efficient model adaption
- ML inference optimizations
- Compression algorithms
- AI applications

Machine learning system basic

- Deep Learning workloads
- Computation graph
- ML frameworks

Distributed training strategies to break the memory wall

- Data parallelism
- Tensor parallelism
- Pipeline parallelism
- Sequence parallelism
- Gradient checkpointing
- Auto parallelism

Ultra-fast inference optimizations

- CUDA kernels
- Kernel fusion
- Flash attention
- Paged attention

Compression algorithms to make model smaller, faster, and cheeper

- Quantization
- Sparsification
- Low rank decomposition
- Distillation

Part 5: LLM-specific optimizations

- Mixture-of-Experts
- Speculative decoding
- LoRA
- RAG
- Agents
- …

Learning outcomes of this course

By the end of this course, you will

- Understand the basic functioning of modern DL libraries, including concepts like compute operators, automatic differentiation, etc.
- Under the full pipeline of modern ML systems, starting from pre-training all the way to deployment…
- Understand scaling-up, why and how? All sorts of machine learning parallelization techniques, and latest research in the area …
- Understand hardware acceleration/CUDA/GPUs, and can program/debug a little accelerator programs …
- Ground all you learning in the context of LLMs, understand the L of **L**LM, how it is optimized, scaled, trained, served…
- Have fun!

Questions?

Course Website

CS 498 Machine Learning System, Spring 2025

Basic Information

Instructor: Minjia Zhang
Schedule: Tuesdays and Thursdays 2-3:15pm CST
Location: 1214 Siebel Center for Computer Science
Office hours: TBD
Instructor Email: minjiaz AT illinois.edu
TA: Ahan Gupta
TA Email: ag82@illinois.edu
LMS: Canvas
Recommended Prerequisites: CS 425 - Distributed Systems CS 484 - Parallel Programming, CS 533 - Parallel Computer Architecture, CS 446 - Machine Learning

Course Description

Welcome to the Spring 2025 offering of CS 498: Machine Learning System!

This is a new undergraduate course offered for the first time at UIUC. Therefore, we might adjust the schedule and content depending on your learning progress.

The goal of this course is to provide students with an in-depth understanding of various elements of modern machine learning systems, ranging from the performance characteristics of ML m will also conduct case studies on modern large language model training and serving and cover the design rationale behind state-of-the-art machine learning frameworks.

Tentative Course Schedule

Course Policy

Grading

The course assignments consist of (i) attendance and class participation, (ii) lab assignments, (iii) reading summary, (iv) final project presentation, and (v) completing an open-ended research

Grading Breakdown	
Attendance and class participation	20%
Lab assignments	20% (2 lab assignments, 10% each)
Reading summary	20% (10 readings, 2% each)
Final research project presentation	15%
Project report	25% (5% + 5% + 15%)

https://minjiazhang.github.io/courses/cs-mlsys-498-2025spring.html

TA

- Ahan Gupta (ag82@illinois.edu)
  - PhD@CS
  - Experience: LLM system optimizations
  - OH: TBD

Components and Grading

- Attendance and class participation 20%
- Lab assignments 20% (2 lab assignments, 10% each)
- Reading summary 20% (10 readings, 2% each)
- Final research project presentation 15%
- Project report 25% (5% + 5% + 15%)

Grading Scheme (grade is the better of the two)

Grade	Absolte Cutoff (>=)	Relative Bin
A+	95	Highest 5%
A	90	Next 10%
A-	85	Next 15%
B+	80	Next 15%
B	75	Next 15%
B-	70	Next 15%
C+	65	Next 5%
C+	60	Next 5%
C-	55	Next 5%
D	50	Next 5%
F	<50	Lowest 5%

Grading Scheme (grade is the better of the two)

Grade	Absolte Cutoff (>=)	Relative Bin
A+	95	Highest 5%
A	90	Next 10%
A-	85	Next 15%
B+	80	Next 15%
B	75	Next 15%
B-	70	Next 15%
C+	65	Next 5%
C+	60	Next 5%
C-	55	Next 5%
D	50	Next 5%
F	<50	Lowest 5%

Example, 82 and 33%,
Abs: B+; Rel: B-;
Final: B+

Structure of the Course (Tentative)

Week	Part 1: Basics	
1	Course intro	DL Workloads
2	DL frameworks	AutoDiff
	Part 2: Distributed ML	
3	Overview of training	Communication collectives
4	Data parallelism, tensor parallelism	Pipeline parallelism
5	Zero-style data parallelism	Heterogeneous GPU-CPU
6	3D parallelism	Auto parallelism
7	Mixed precision training	Communication compression
	Part 3: Inference optimizations	
8	CUDA basics	
9	FlashAttention	PagedAttention
10	Continusou batching	Efficient scaling of transformer inference
11	TVM and DL compiler	
	Part 4: Compression	
11	Quantization 1	Quantization 2
12	Sparsification 1	Sparsification 2
13	Distillation	Low-rank decomposition
14	KV cache compression 1	KV cache compression 2
	Part 5: Misc	
15	MoE 1	MoE2
16	Vector db	RAG

Lab Assignments

- Two lab assignments (TBD)
  - Likely will use NCSA clusters for GPUs
  - The instructor needs to figure out some details

- Topics
  - Inference optimizations
  - Compress ML models

Reading Summary

- Required reading:
  - The instruction will select 10 highly relevant papers in MLSys (mostly under 12 pages).
  - One paper per week (starting from Jan 27), submit your reading by the end of day of each Friday.
  - The reading summary should be done independently and include the following content:
    - The problem the paper is trying to tackle.
    - What's the impact of the work, e.g., why is it an important problem to solve?
    - The main proposed idea(s).
    - A summary of your understanding of different components of the proposed technique, e.g., the purpose of critical design choices.
    - Your perceived strengths and weaknesses of the work, e.g., novelty, significance of improvements, quality of the evaluation, easy-to-use.
    - Is there room for improvement? If so, what idea do you have for improving the techniques?
  - The reading summary length should be around 4-5 paragraphs.

Reading Summary

Grading criteria, each summary has 12 points in total. For each review item above, you get:

- 2: The summary item demonstrates a clear understanding of the paper.
- 1: The summary item misses the point of the paper.
- 0: The summary item is missing.

Reading List

Paper Reviews

The instruction will select 10 highly relevant papers in MLSys (mostly under 12 pages). One paper per week (starting from Jan 27), submit your :

Reading List

The reading summary should be done independently and include the following contents:

- The problem the paper is trying to tackle.
- What's the impact of the work, e.g., why is it an important problem to solve?
- The main proposed idea(s).
- A summary of your understanding of different components of the proposed technique, e.g., the purpose of critical design choices.
- Your perceived strengths and weaknesses of the work, e.g., novelty, significance of improvements, quality of the evaluation, easy-to-use.
- Is there room for improvement? If so, what idea do you have for improving the techniques?

ML System Reading List

(3D Parallelism) Efficient large-scale language model training on GPU clusters using megatron-LM
SC 2021

(ZeRO-style Data Parallelism) ZeRO: Memory Optimizations Toward Training Trillion Parameter Models
SC 2020

Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning
OSDI 2022

FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness
NeurIPS 2022

Orca: A Distributed Serving System for Transformer-Based Generative Models
OSDI 2022

(vLLM) Efficient Memory Management for Large Language Model Serving with PagedAttention
SOSP 2023

GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers
ICLR 2023

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models
ICML 2023

Fast Inference from Transformers via Speculative Decoding
ICML 2023

Mamba: Linear-Time Sequence Modeling with Selective State Spaces
Arxiv 2024

https://minjiazhang.github.io/courses/publication-list-v3.html

Course Project

- The course also includes proposing and completing a course project. The project can involve, but is not limited to, any of the following tasks:
  - Benchmark and analyze important DL workloads to understand their performance gap and identify important angles to optimize their performance.
  - Apply and evaluate how existing solutions work in the context of emerging AI/DL workloads.
  - Design and implement new algorithms that are both theoretically and practically efficient.
  - Design and implement system optimizations, e.g., parallelism, cache-locality, IO-efficiency, to improve the compute/memory/communication efficiency of AI/DL workloads.
  - Offer customized optimization for critical DL workloads where latency is extremely tight.
  - Build library/tool/framework to improve the efficiency of a class of problems.
  - Integrate important optimizations into existing frameworks (e.g., DeepSpeed), providing fast and agile inference.
  - Combine system optimization with modeling optimizations.
  - Combine and leverage hardware resources (e.g., GPU/CPU, on-device memory/DRAM/NVMe/SSD) in a principled way.
  - ...
- The project will be done in groups of 2-3 people, which consists of a proposal, mid-term report, final presentation, and final report. The tentative timeline for the project is as follows.

Final Presentation

- 15%
- Please spend a significant amount of time on working your 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 semester

Final Report

- Final report: The final report will be in the style of a research paper describing your project. The recommended length is about **6-8 pages** long (excluding references) and a potential division can be: An abstract, which summarizes the project (0.25 pages).
  - An introduction, which describes and motivates the problem and summarizes the main results of the work (0.5 pages).
  - A brief discussion of related work (0.25 pages).
  - A brief overview of preliminary and background knowledge needed to understand the paper (0.25 pages).
  - Analysis and characterization to show the existence and severity of the problem (1 page).
  - Main design and implementation (1 pages).
  - Evaluation methodology and experiment results (1 page).
  - Concluding remarks, which can include a discussion of open questions or directions for future work (0.25 pages).

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.

Late Submissions

- All assignments are due on the respective due date. Only on-time assignments will be accepted.

Questions?