Machine Learning Systems and Hardware

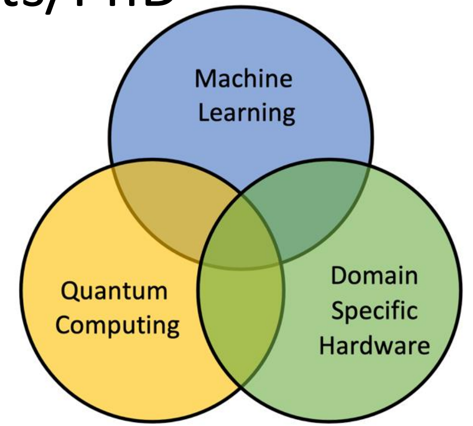
L0: Introduction

Hongxiang Fan

IMPERIAL

About Me (Hongxiang Fan, Assistant Professor)

- Experience
  - Research Scientist, Samsung AI Cambridge
  - Affiliated Research Fellow, CS, University of Cambridge
- My Research:
  - Intersection among **Hardware** / **Machine Learning** / **Quantum Computing**
  - System / Hardware Support for Emerging ML (LLM, Neural Rendering)
  - Quantum Compilation / Simulation / Error Correction
  - ML for Hardware / Software Engineering
- Research Opportunities: UROP/Intern/ISO/Individual Projects/PhD
- Other Lectures in Doc:
  - ❖ Compilers (together with Paul and Jamie)



CATALOG

Why ML Sys & HW: Ubiquity of ML

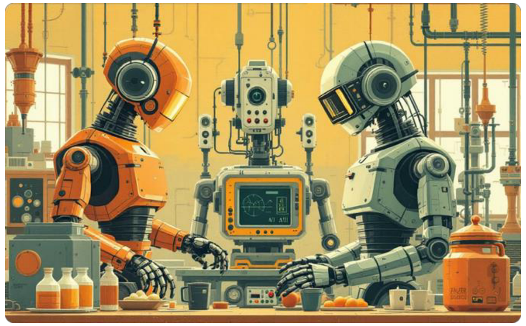
- Machine Learning (ML) is becoming pervasive
- Influencing our daily lives across various applications

Health Care

Science

Content Creation

Education


Manufacturing

Finance

Why ML Sys & HW: Ubiquity of ML

- Generative AI
  - Text generation and understanding such as ChatGPT, Gemini
  - Applications: code generation, question answering, dialogue systems
- Beyond text: multi-modal AI
  - Audio (Whisper from OpenAI)
  - Image/Video (Nano Banana and Veo from Google)

ChatGPT

Whisper

NANO BANANA

Google Veo 3

Sora 2

Gemini

Why ML Sys & HW: What Drives the AI Wave?

- Three key enablers:

Algorithm

?

Deep Neural Networks were invented in 1960s

Data

GPUs

Memory Storage

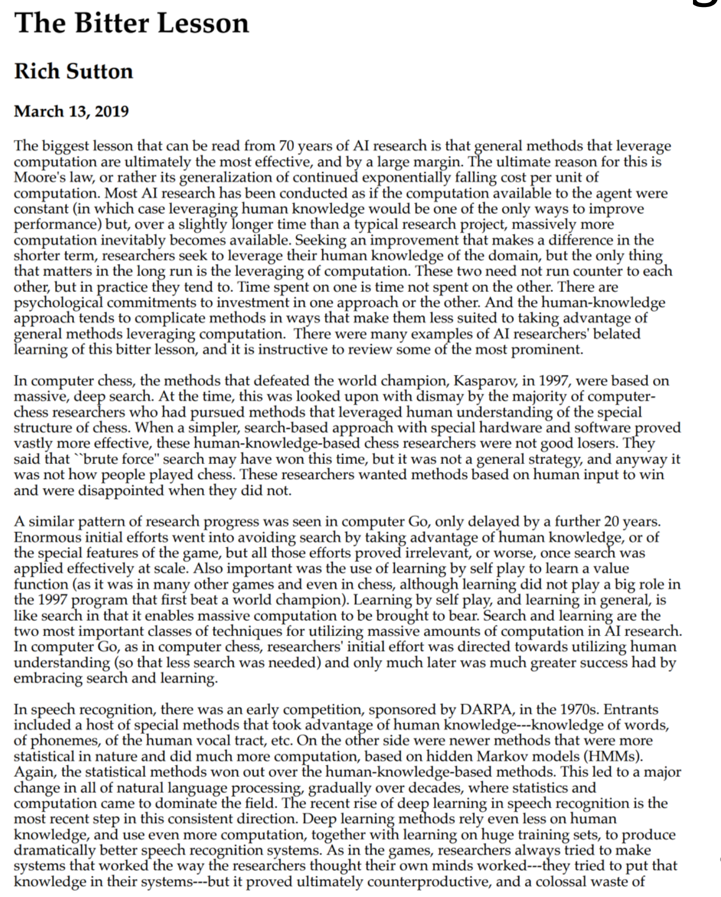
Compute Hardware

ML Systems and Hardware?

Why ML Sys & HW: The Bitter Lesson

- Richard Sutton: Turing Award in 2024
- One of the founders of modern computational Reinforcement Learning

- “General methods that **leverage computation** are ultimately the most **effective**”
- “**computation**”: hardware
- “**leverage**”: system infrastructure



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.

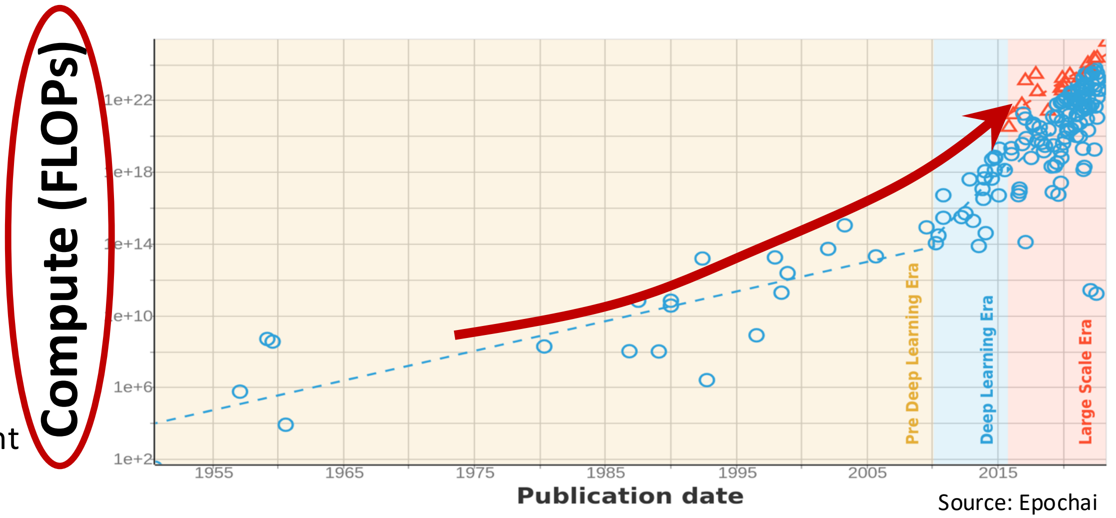
In speech recognition, there was an early competition, sponsored by DARPA, in the 1970s. Entrants included a host of special methods that took advantage of human knowledge---knowledge of words, of phonemes, of the human vocal tract, etc. On the other side were newer methods that were more statistical 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

Source: http://www.incompleteideas.net/IncIdeas/BitterLesson.html

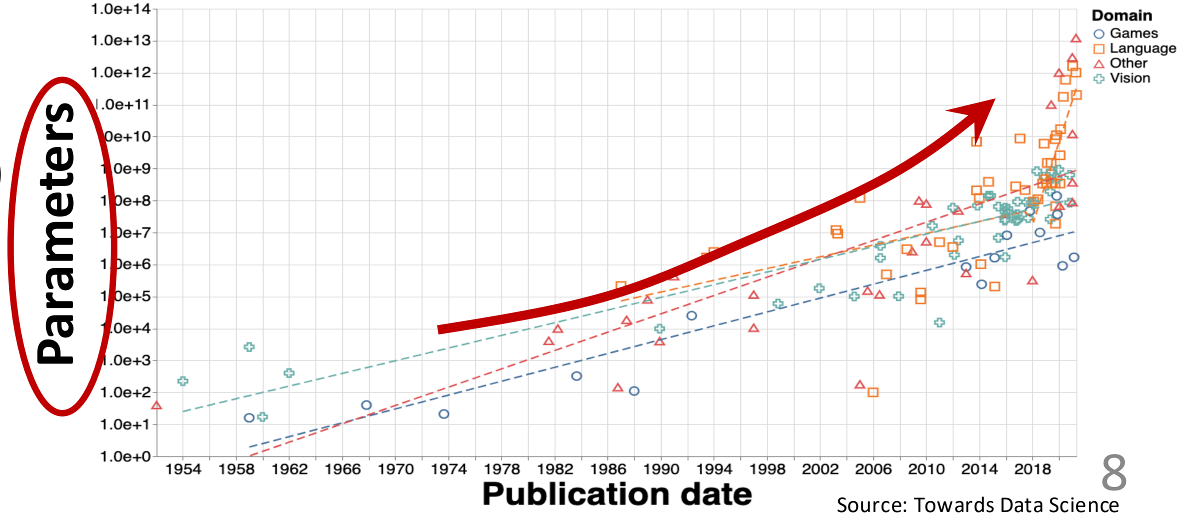
Why ML Sys & HW

- Increasing computational complexity
- Computation:
  - Linear layers
  - Convolutional layers
  - Attention layers
- Parameters:
  - Weight matrices
  - Key-value cache (language models)
  - Multi-model (Agentic AI/Reasoning)

FLOPs: Floating Point Operations



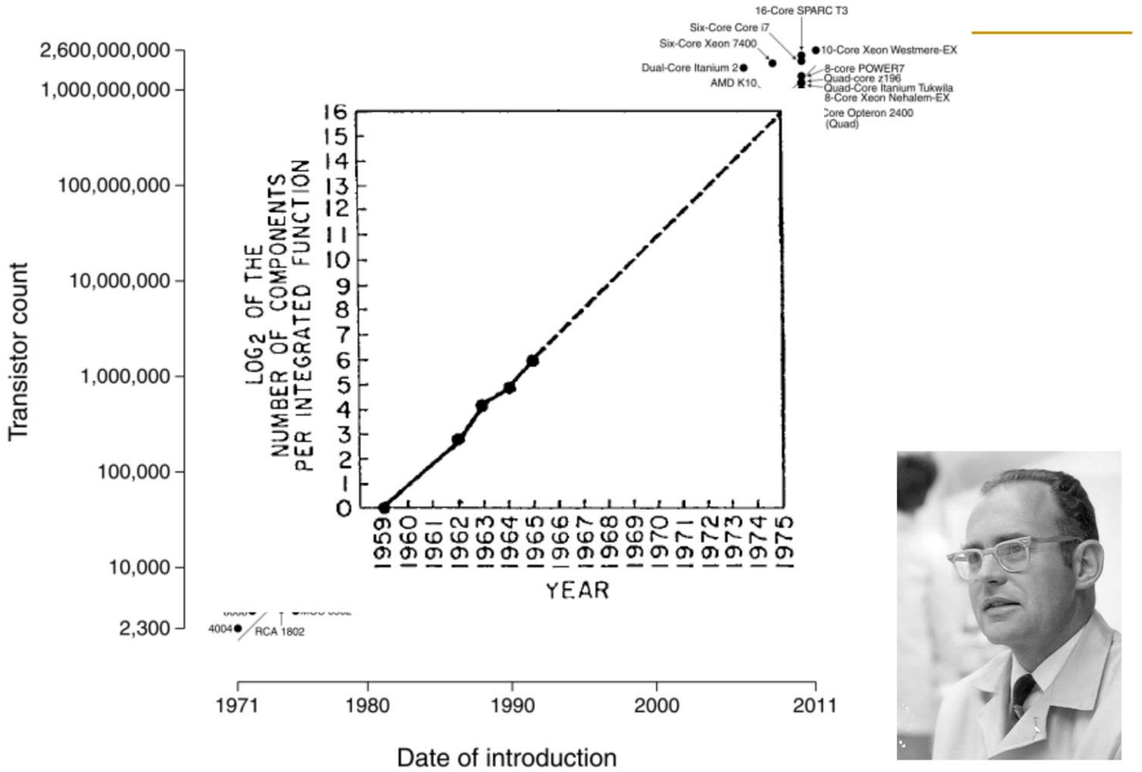
Source: Epochai



Source: Towards Data Science

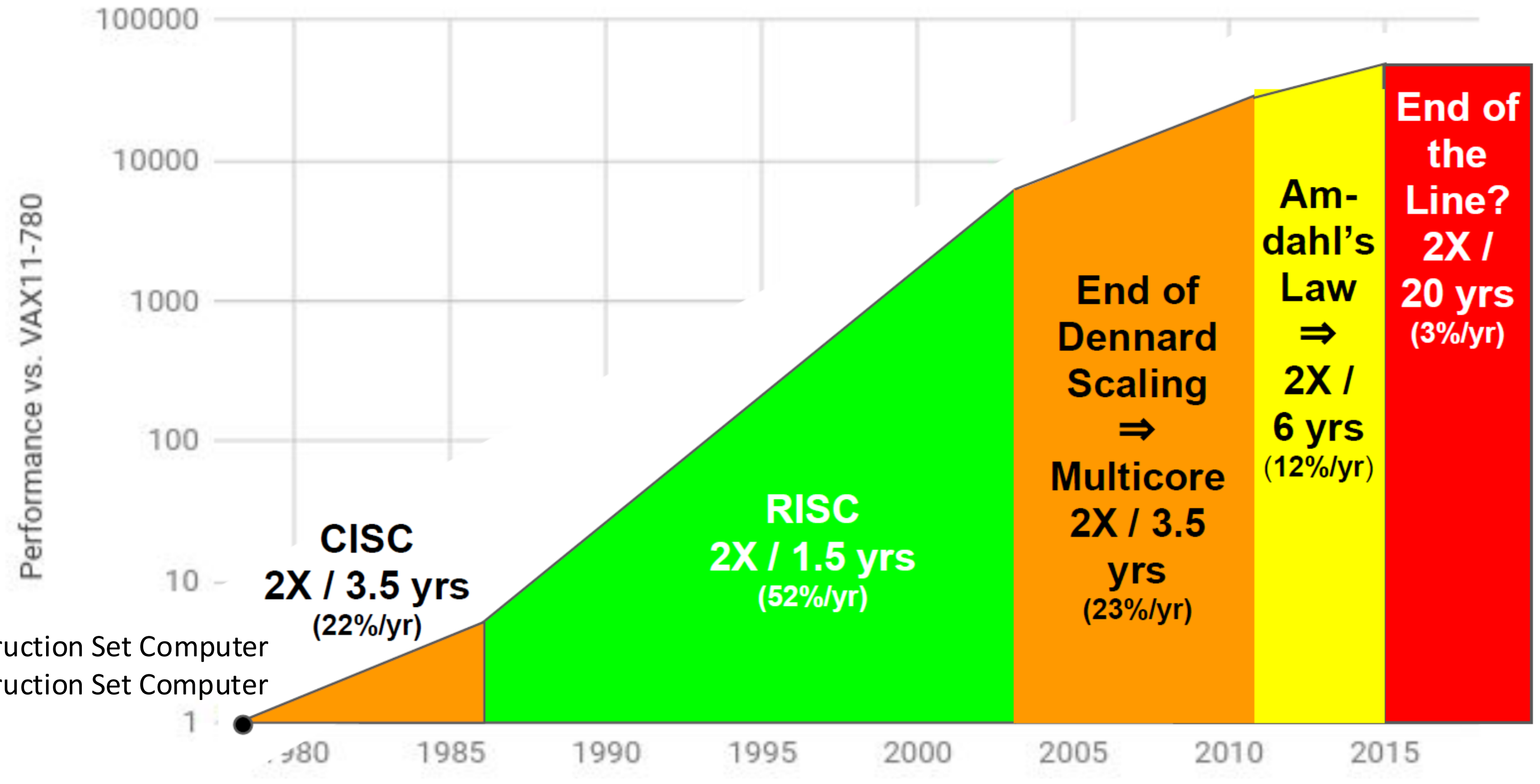
Slowing Down of Moore’s Law

- Number of transistors on an integrated circuit doubles every two years
- For decades, performance improvements came from Moore’s Law scaling
- This trend is slowing, creating a need for specialized processors



Slowing Down of Moore's Law

- CPU performance improvement

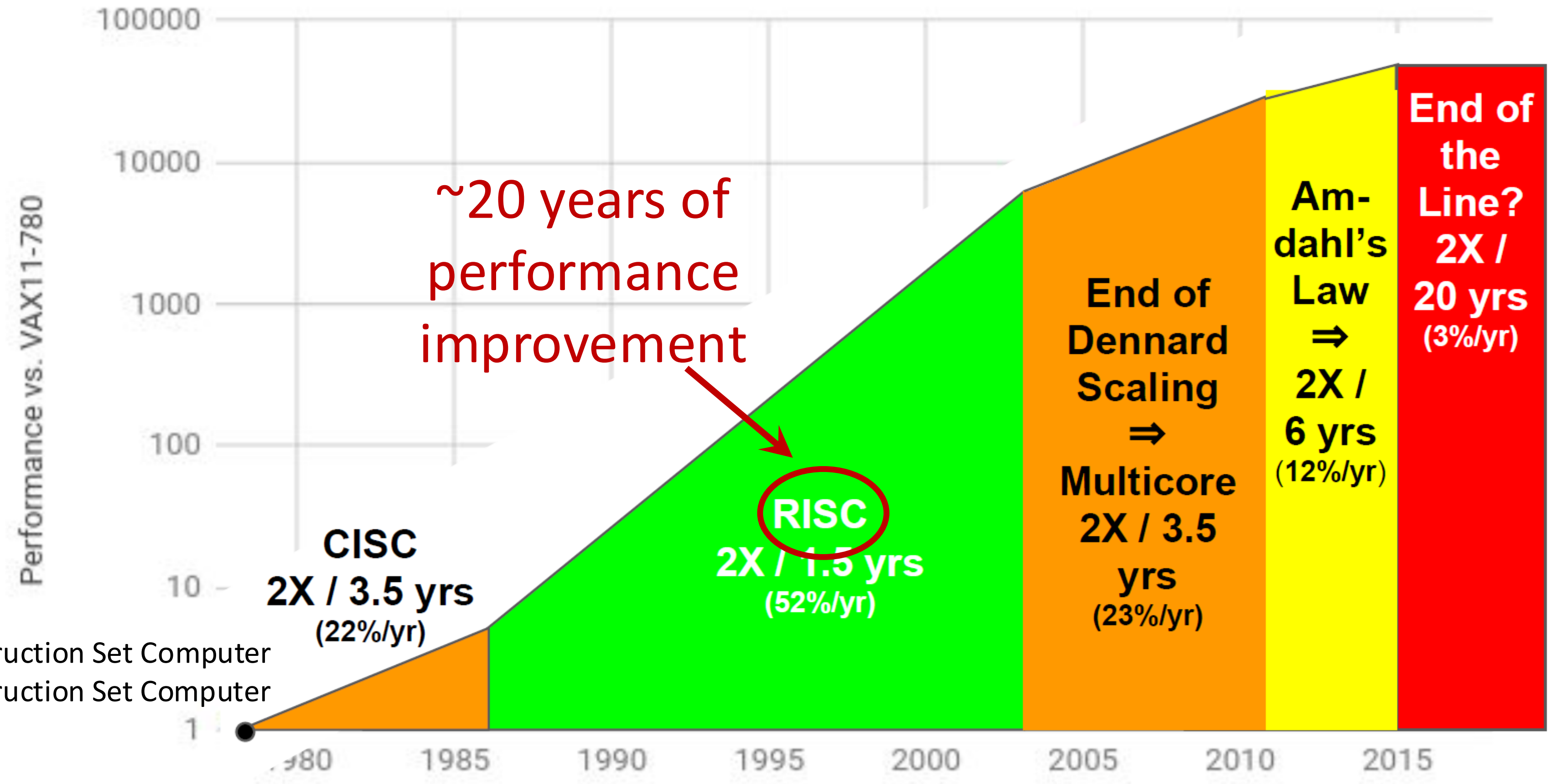


CISC: Complex Instruction Set Computer
RISC: Reduced Instruction Set Computer

Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Slowing Down of Moore's Law

- CPU performance improvement



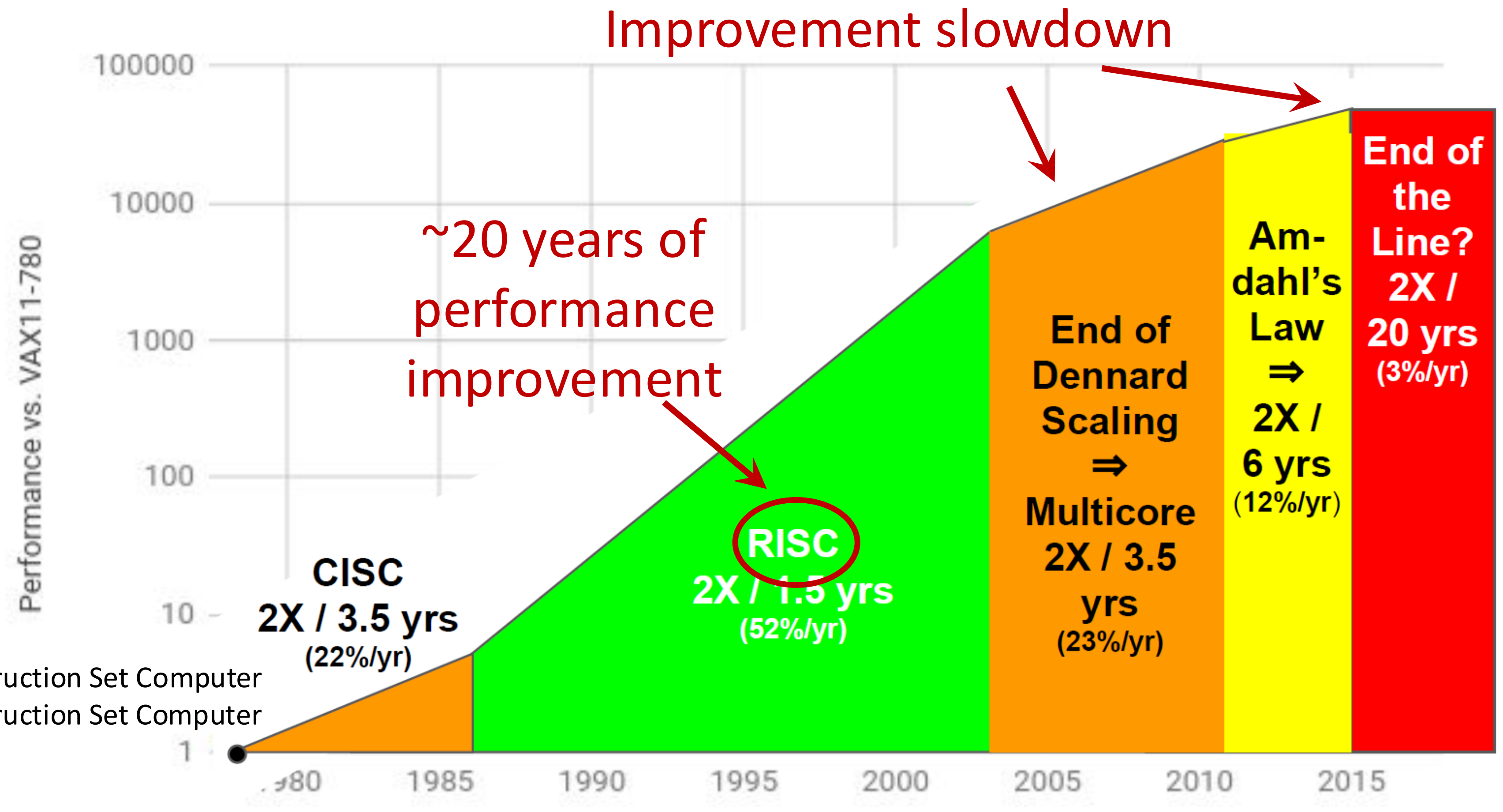
CISC: Complex Instruction Set Computer

RISC: Reduced Instruction Set Computer

Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Slowing Down of Moore's Law

- CPU performance improvement



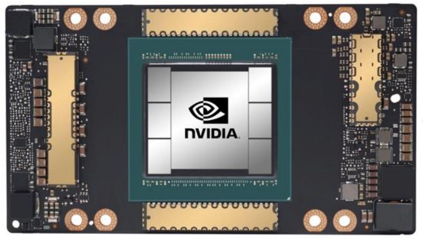
CISC: Complex Instruction Set Computer
RISC: Reduced Instruction Set Computer

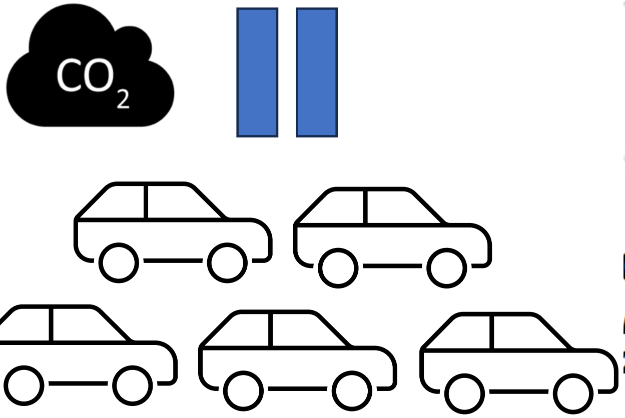
Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Slowing Down of Moore’s Law

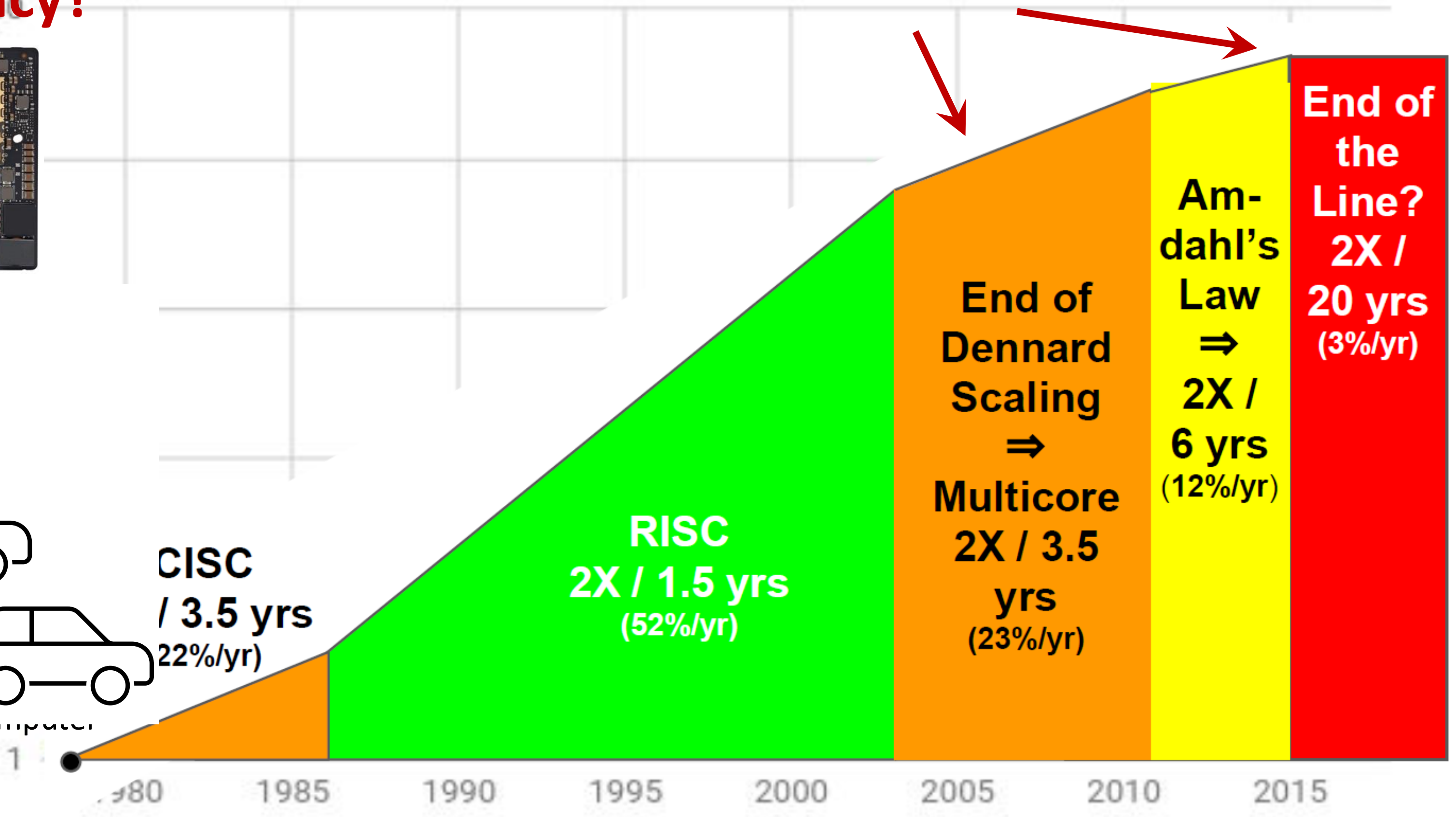
- How about GPU?

Energy efficiency?





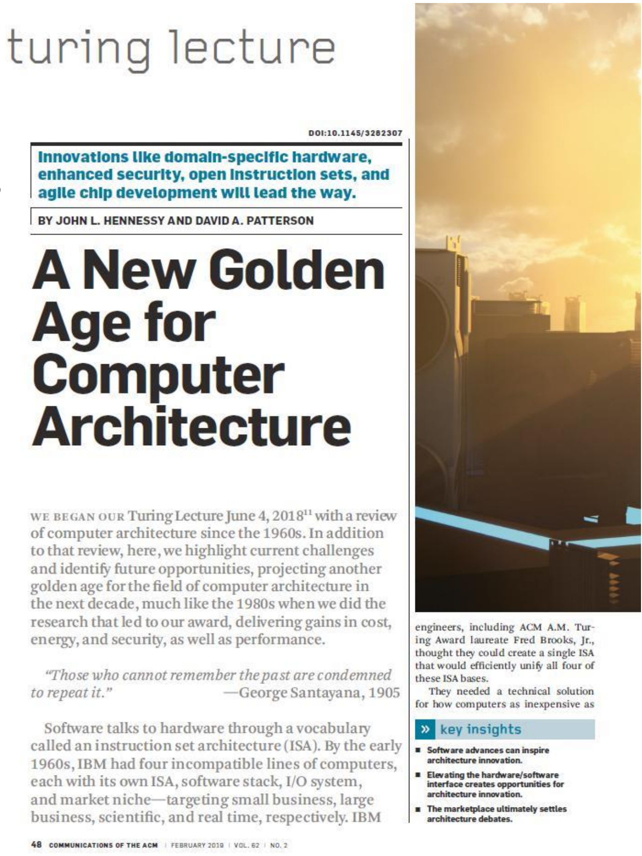
Improvement slowdown



Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Golden Age of Computer Architecture

- John L. Hennessy, David Patterson, 2017's Turing Award:
  - "A New Golden Age for Computer Architecture", Communications of the ACM, 2019
  - More performance gain from advanced architecture design, **not** just technology scaling
- Reconfigurable Accelerators
  - FPGA (Field-Programmable Gate Array)
  - CGRA (Coarse-Grained Reconfigurable Array)
- DSA (Domain Specific Architecture)
  - TPU from Google, NPU from Samsung
  - Processing in/near Memory
  - Processing in/near Sensors
  - Processing in/near Networks



turing lecture

DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

WE BEGAN OUR Turing Lecture June 4, 2018[11] with a review of computer architecture since the 1960s. In addition to that review, here, we highlight current challenges and identify future opportunities, projecting another golden age for the field of computer architecture in the next decade, much like the 1980s when we did the research that led to our award, delivering gains in cost, energy, and security, as well as performance.

"Those who cannot remember the past are condemned to repeat it." —George Santayana, 1905

Software talks to hardware through a vocabulary called an instruction set architecture (ISA). By the early 1960s, IBM had four incompatible lines of computers, each with its own ISA, software stack, I/O system, and market niche—targeting small business, large business, scientific, and real time, respectively. IBM engineers, including ACM A.M. Turing Award laureate Fred Brooks, Jr., thought they could create a single ISA that would efficiently unify all four of these ISA bases.

They needed a technical solution for how computers as inexpensive as

» key insights

- Software advances can inspire architecture innovation.
- Elevating the hardware/software interface creates opportunities for architecture innovation.
- The marketplace ultimately settles architecture debates.

48 COMMUNICATIONS OF THE ACM | FEBRUARY 2019 | VOL. 62 | NO. 2

Source: Communications of the ACM

Why ML Sys & HW

- Systems and Hardware are key enablers for ML/AI

➢ Pursuing the “most effective” approach (Richard Sutton)

- AI Scaling meets end of Moore’s Law

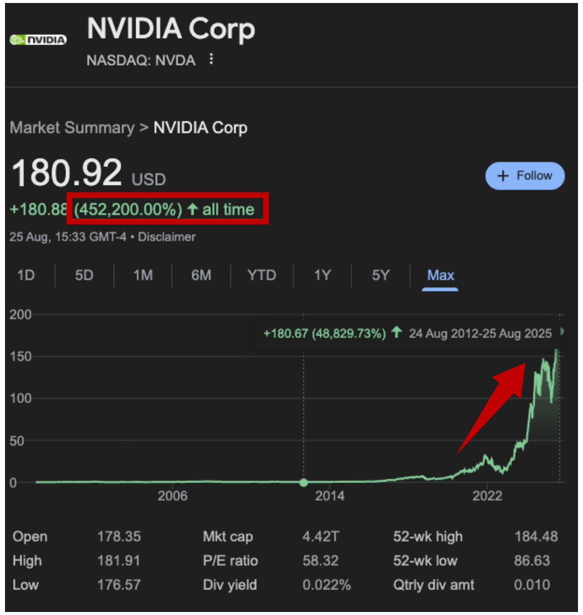
➢ The *Golden Age of Computer Architecture*

➢ Rise of Domain-specific Hardware

- Many additional reasons why ML Systems & Hardware is essential

Why ML Sys & HW

- Global AI industry
  - Massive investments in AI infrastructure worldwide
- Career Opportunities
  - Microsoft, NVIDIA, Google: expanding AI infrastructure in the UK
  - AI-driven automation reshaping entry-level software engineering roles
  - Widespread adoption in finance and trading companies


NVIDIA Corp
NASDAQ: NVDA

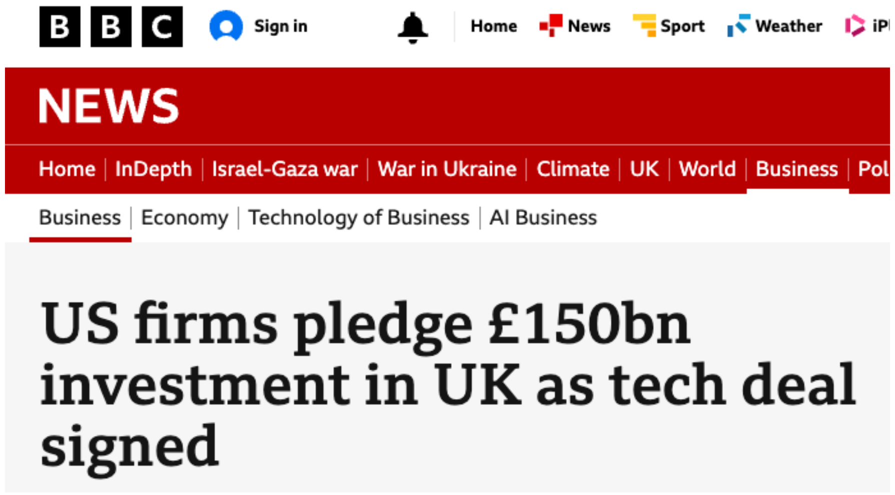
Market Summary > NVIDIA Corp

180.92 USD

+180.88 (452,200.00%) ↑ all time

25 Aug, 15:33 GMT-4 • Disclaimer

Open	178.35	Mkt cap	4.42T	52-wk high	184.48
High	181.91	P/E ratio	58.32	52-wk low	86.63
Low	176.57	Div yield	0.022%	Qtrly div amt	0.010

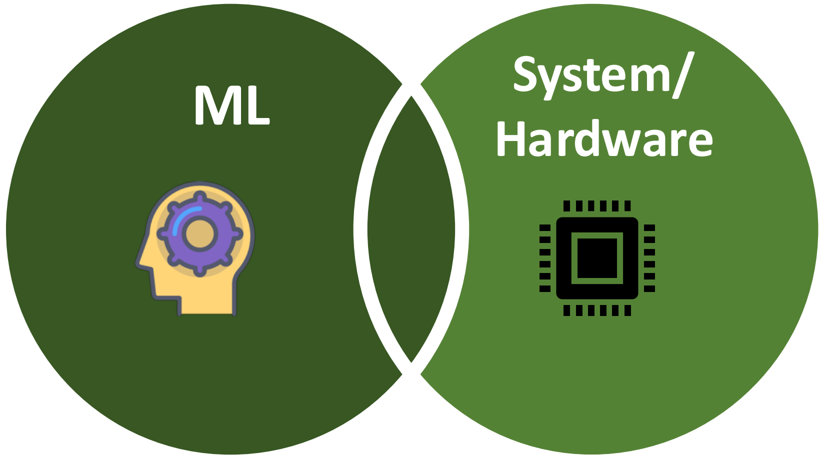


US firms pledge £150bn investment in UK as tech deal signed

Source: BBC

Why ML Sys & HW

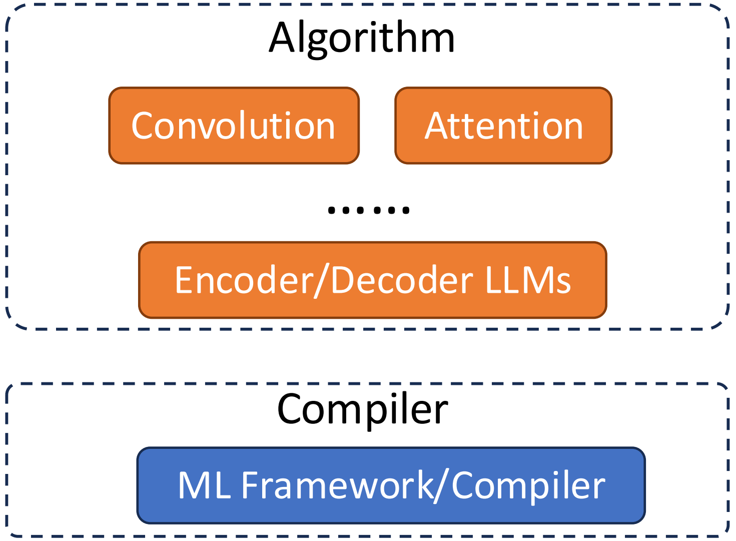
- Unique Position of This Module
  - Bridges ML algorithms and hardware systems
  - Compared with pure hardware modules:
    - More related to ML
    - Latest research of systems and hardware in AI
  - Compared with pure algorithm modules:
    - Stronger focus on practical systems and fundamentals
    - Hardware-aware Efficient ML algorithm



CATALOG

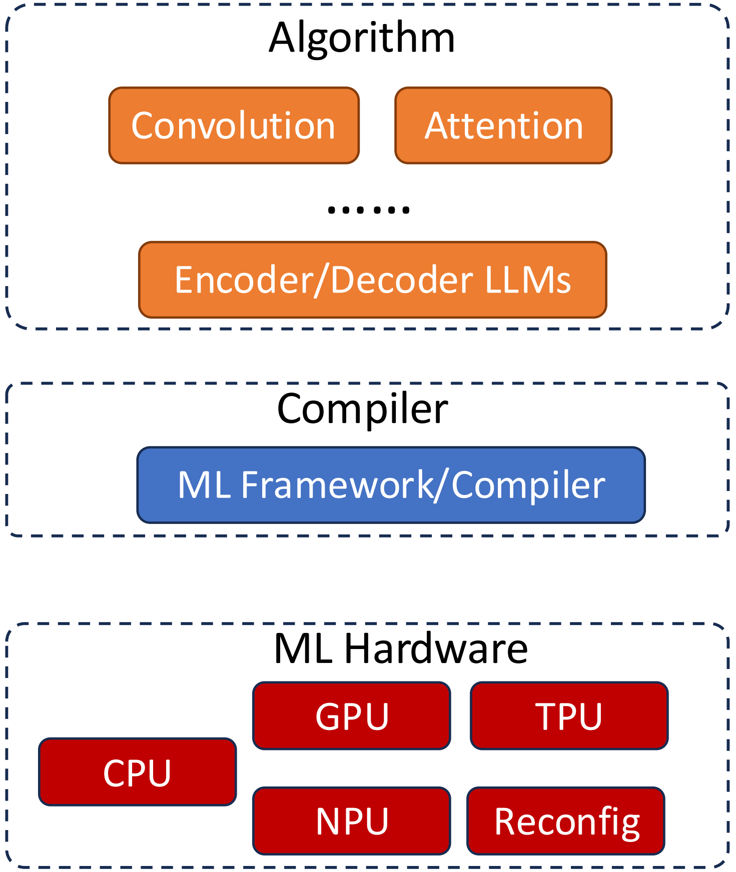
Course Overview: Syllabus

1. Introduction & Module Overview
   - Mainly focus on deep learning
2. Workload Analysis (Beyond Algorithm):
   - Basic computation/operations
     - Linear/Convolutional layers and their efficient variants
     - Attention layers (auto-regressive LLM, diffusion LLM)
   - Computational patterns and complexity analysis
   - Performance analysis: roofline model
     - Analysis of memory-bound and compute-bound
3. ML Compiler and Mapping
   - Front-end: imperative and declarative
   - Dataflow construction: forward and backward
   - Intermediate representation
   - Graph/Tensor optimization
   - Kernel Execution



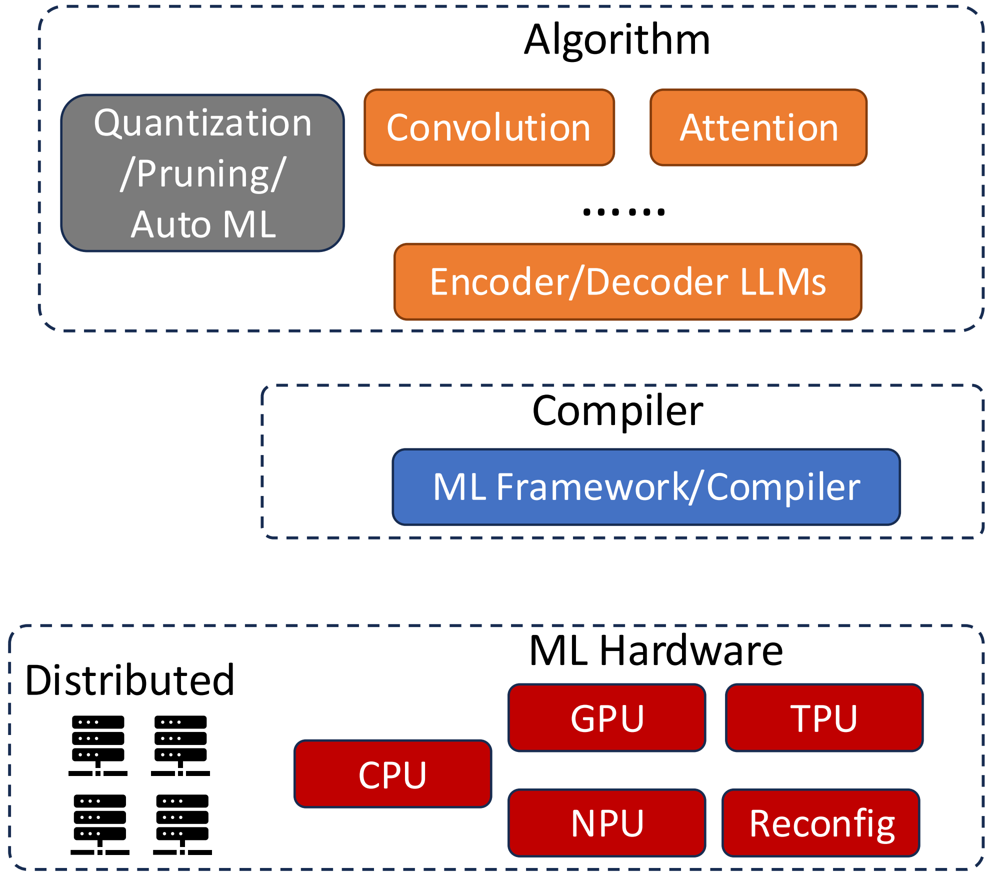
Course Overview: Syllabus

4. Graphics Processing Unit (GPU)
   - GPU hardware architecture
   - GPU programming (CUDA)
     - Memory coalescing
     - Shared memory caching
   - Custom kernels (Triton)
5. AI Hardware
   - Systolic array / Domain-specific accelerator
     - TPU architecture and dataflow variations
   - Reconfigurable acceleration
     - FPGA-based design
     - Coarse-Grain Reconfigurable Architectures (CGRA)



Course Overview: Syllabus

6. Efficient ML
   - Quantization: linear/logarithmic, static/dynamic
   - Pruning: structured/unstructured
   - AutoML: neural architecture search
7. Distributed ML
   - Parallelism strategies
     - tensor, spatial, pipeline, data parallelism
   - Decentralized ML
     - Federated learning
     - Model merging
   - Large-scale LLM systems
     - Disaggregated acceleration
     - Scaling Law
     - Test-time scaling



Tutorial & Guest Lecture

- Seven tutorial sessions (1–2 hours each)
- Question-based session (~1 hour)
  - Work through example problems
  - 20–30 minutes: student answering
  - 20–30 minutes: explanation and discussion
- Paper reading session (~1 hour, flexible)
  - Engage with the latest research
  - 20–30 minutes: individual reading
  - 20–30 minutes: discussion (led by lecturer)
- Guest industry lecture
  - Insights into current industrial practices and trends

Prerequisites

- Basic Mathematics
  - linear algebra and calculus
- Deep Learning
  - Fundamentals of training and inference
- Computer Architecture
  - Basic concepts in instruction pipelines and memory systems
- Programming Skills
  - Python and/or C++

Coursework

- GPU Programming Project
  - Performance profiling
  - Performance analysis
  - Performance optimization
- What to submit
  - Code implementation
  - Short written report
- Release date: 6th of November (same day of the first lab)
- Deadline: 4th of December

Course Overview: Tutorials & Lab Sessions

- Timetable/Schedule (October)

Date	Time	Content
6 Oct	11:00-12:00	Introduction
	12:00-13:00	Operator and Analysis-1
9 Oct	11:00-12:00	Operator and Analysis-2
	12:00-13:00	Tutorial-1 (Q&A)
13 Oct	11:00-12:00	Compilation and Mapping
	12:00-13:00	
16 Oct	11:00-12:00	Tutorial-2 (Q&A)
	12:00-13:00	Tutorial-2 (Paper Reading)
20 Oct	11:00-12:00	GPU
	12:00-13:00	
23 Oct	11:00-12:00	Tutorial-3 (Q&A)
	12:00-13:00	Tutorial-3 (Paper Reading)

Date	Time	Content
27 Oct	11:00-12:00	Custom Kernel
	12:00-13:00	Industrial Lecture
30 Oct	11:00-12:00	AI Hardware
	12:00-13:00	Tutorial-4 (Q&A)

- Room 145
  - Lecture
  - Tutorial (Q&A)
  - Tutorial (Paper Reading)
- Room 219
  - Lab Session

Course Overview: Tutorials & Lab Sessions

- Timetable/Schedule (November)

Date	Time	Content
3 Nov	11:00-12:00	Efficient ML-1
	12:00-13:00	
6 Nov	11:00-12:00	Tutorial-4 (Paper Reading)
	12:00-13:00	Lab Session
10 Nov	11:00-12:00	Efficient ML-2
	12:00-13:00	Tutorial-5 (Q&A)
13 Nov	11:00-12:00	Tutorial-5 (Paper Reading)
	12:00-13:00	Lab Session
17 Nov	11:00-12:00	Tutorial-6 (Q&A)
	12:00-13:00	Tutorial-6 (Paper Reading)
20 Nov	11:00-12:00	Distributed ML System-1
	12:00-13:00	Lab Session

Date	Time	Content
24 Nov	11:00-12:00	Distributed ML System-2
	12:00-13:00	Tutorial-7 (Q&A)
27 Nov	11:00-12:00	Tutorial-7 (Paper Reading)
	12:00-13:00	Lab Session

- Room 145
  - Lecture
  - Tutorial (Q&A)
  - Tutorial (Paper Reading)
- Room 219
  - Lab Session

Course Overview: Learning Outcomes

1. Understand the computation of deep neural networks and examine workload characteristics using analytical models.

2. Analyse the architectural features and design trade-offs of modern ML hardware, including GPUs, TPUs, and emerging AI accelerators.

3. Implement and evaluate efficient kernel design and ML techniques.

4. Understand the principles of algorithm–hardware co-design and AutoML techniques for architecture search and optimization.

5. Analyse large-scale deep learning workloads and evaluate distributed ML systems, including parallelism strategies and decentralized learning.

CATALOG

Logistics: Assessment Strategy and Resources

- Assessment Strategy
  - Exam (80%): Exam Questions are based on taught lectures.
  - Coursework/Assignment (20%): Coding and report
- Supplementary Reading
  - Deep Learning (Ian Goodfellow, Yoshua Bengio and Aaron Courville)
  - Introduction to Machine Learning Systems (Prof. Vijay Janapa Reddi, Harvard University)
  - Dive into Deep Learning
- Online Courses
  - Deep Learning Systems: CMU
  - Data Systems for Machine Learning (LLM-focused): UCSD
  - TinyML and Efficient Deep Learning Computing (EfficientML): MIT

Logistics: Resources

- Module Materials (Scientia)
  - Tutorials and slides uploaded one week in advance
  - Coursework released one day before the lab session
- Discussion (Ed System)
  - Online Q&A and discussion forum
  - Lecturer and GTAs will check from time to time
- Lecture Recordings (Panopto)
  - Automatically recorded teaching sessions

Graduate Teaching Assistants

- GTA1: Zhiwen Mo
  - Research Interests: GPU Microarchitecture, Performance Modeling and Kernel Optimization
- GTA2: Bakhtiar Zadeh
  - Research Interests: Efficient ML Inference and Novel Architectures
- GTA3: Qianzhou (Terry) Wang
  - Research Interests: Hardware Verification and Formal Methods
- GTA4: Guoyu Li
  - Research Interests: Domain Specific Accelerator Design / AI Software-Hardware Co-Optimization
- GTA5: Jinnan Guo
  - Research Interests: ML System and Federate Learning

Q&A

Any Questions?