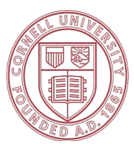
4950/6950 Fall 2026
Special Topics in ECE: AI Hardware

Course Overview

Zhiru Zhang
School of ECE, Cornell University



Cornell University

CSL

Agenda

- Important logistics
- Course motivation
- More course organization

Class Resources

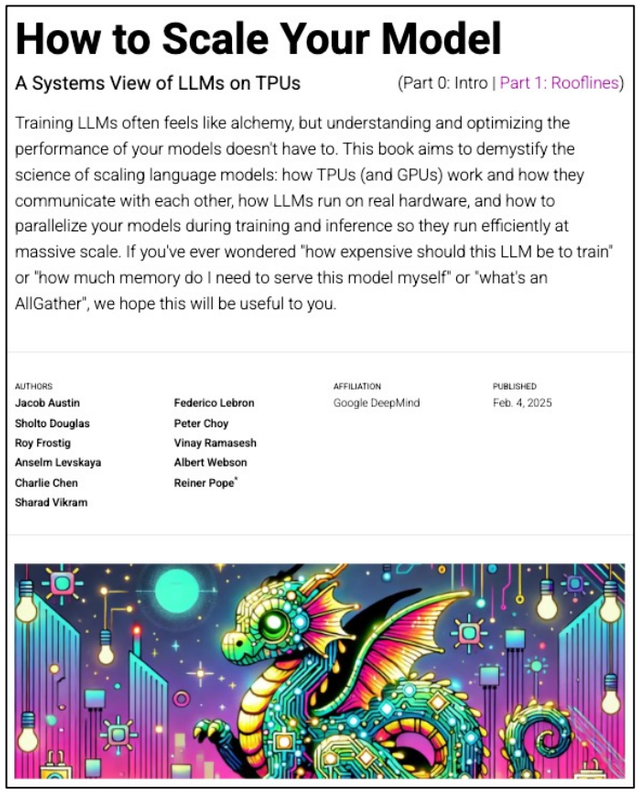
- Course website
  - https://www.csl.cornell.edu/courses/ece4950fa26/
  - Lectures slides, handouts, and other readings

- Ed Discussion
  - Announcements and Q&A

- CMSX: course management system
  - Assignments and grades
  - Electronic submissions required

- Cornell GitHub
  - https://github.coecis.cornell.edu/
  - Lab code distribution

Course Texts



How to Scale Your Model

A Systems View of LLMs on TPUs (Part 0: Intro | Part 1: Rooflines)

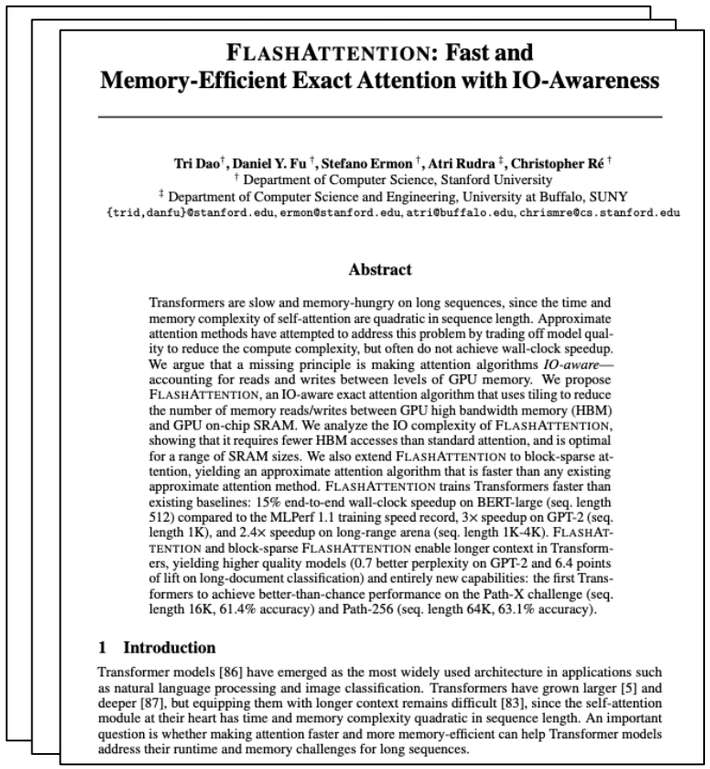
Training LLMs often feels like alchemy, but understanding and optimizing the performance of your models doesn't have to. This book aims to demystify the science of scaling language models: how TPUs (and GPUs) work and how they communicate with each other, how LLMs run on real hardware, and how to parallelize your models during training and inference so they run efficiently at massive scale. If you've ever wondered "how expensive should this LLM be to train" or "how much memory do I need to serve this model myself" or "what's an AllGather", we hope this will be useful to you.

AUTHORS
Jacob Austin, Sholto Douglas, Roy Frostig, Anselm Levskaya, Charlie Chen, Sharad Vikram, Federico Lebron, Peter Choy, Vinay Ramasesh, Albert Webson, Reiner Pope*

AFFILIATION
Google DeepMind

PUBLISHED
Feb. 4, 2025

The Scaling Book
(a living online book)



FLASHATTENTION: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao[†], **Daniel Y. Fu** [†], **Stefano Ermon** [†], **Atri Rudra** [‡], **Christopher Ré** [†]
[†] Department of Computer Science, Stanford University
[‡] Department of Computer Science and Engineering, University at Buffalo, SUNY
{trid,danfu}@stanford.edu, ermon@stanford.edu, atri@buffalo.edu, chrismre@cs.stanford.edu

Abstract

Transformers are slow and memory-hungry on long sequences, since the time and memory complexity of self-attention are quadratic in sequence length. Approximate attention methods have attempted to address this problem by trading off model quality to reduce the compute complexity, but often do not achieve wall-clock speedup. We argue that a missing principle is making attention algorithms *IO-aware*—accounting for reads and writes between levels of GPU memory. We propose FLASHATTENTION, an IO-aware exact attention algorithm that uses tiling to reduce the number of memory reads/writes between GPU high bandwidth memory (HBM) and GPU on-chip SRAM. We analyze the IO complexity of FLASHATTENTION, showing that it requires fewer HBM accesses than standard attention, and is optimal for a range of SRAM sizes. We also extend FLASHATTENTION to block-sparse attention, yielding an approximate attention algorithm that is faster than any existing approximate attention method. FLASHATTENTION trains Transformers faster than existing baselines: 15% end-to-end wall-clock speedup on BERT-large (seq. length 512) compared to the MLPerf 1.1 training speed record, 3× speedup on GPT-2 (seq. length 1K), and 2.4× speedup on long-range arena (seq. length 1K-4K). FLASHATTENTION and block-sparse FLASHATTENTION enable longer context in Transformers, yielding higher quality models (0.7 better perplexity on GPT-2 and 6.4 points of lift on long-document classification) and entirely new capabilities: the first Transformers to achieve better-than-chance performance on the Path-X challenge (seq. length 16K, 61.4% accuracy) and Path-256 (seq. length 64K, 63.1% accuracy).

1 Introduction

Transformer models [86] have emerged as the most widely used architecture in applications such as natural language processing and image classification. Transformers have grown larger [5] and deeper [87], but equipping them with longer context remains difficult [83], since the self-attention module at their heart has time and memory complexity quadratic in sequence length. An important question is whether making attention faster and more memory-efficient can help Transformer models address their runtime and memory challenges for long sequences.

Selected papers & tutorials

Seeking Help After Class

- Ed Discussion
  - Questions on lectures, assignments, projects, etc.
  - Monitored by course staff
- Instructor office hours (online)
  - Thursday 5:00-6:00pm, Zoom link posted on Ed
- Email instructor for personal issues/appointment
- TA OH schedule will be announced soon

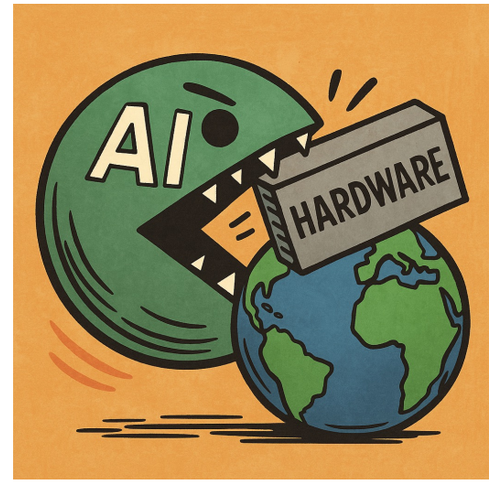
Teaching Assistants

- PhD TA
  - Sunwoo Kim (sk3463)
- MEng TA
  - Stanley Shen (ss3679)
- Undergraduate TAs
  - Linh Anh Nguyen (lan64)
  - Ezra Reiss (er495)
  - Daniel Penas Varela (dp665)
  - Vincent Yi (vsy5)

Grading Scheme

- Class participation (4%)
  - Asking & answering questions during lectures
  - Contributing to discussions on Ed
- Quizzes (6%)
- Homework & Labs (45%)
- Final project (10%)
- Exam (35%)

The Big Picture

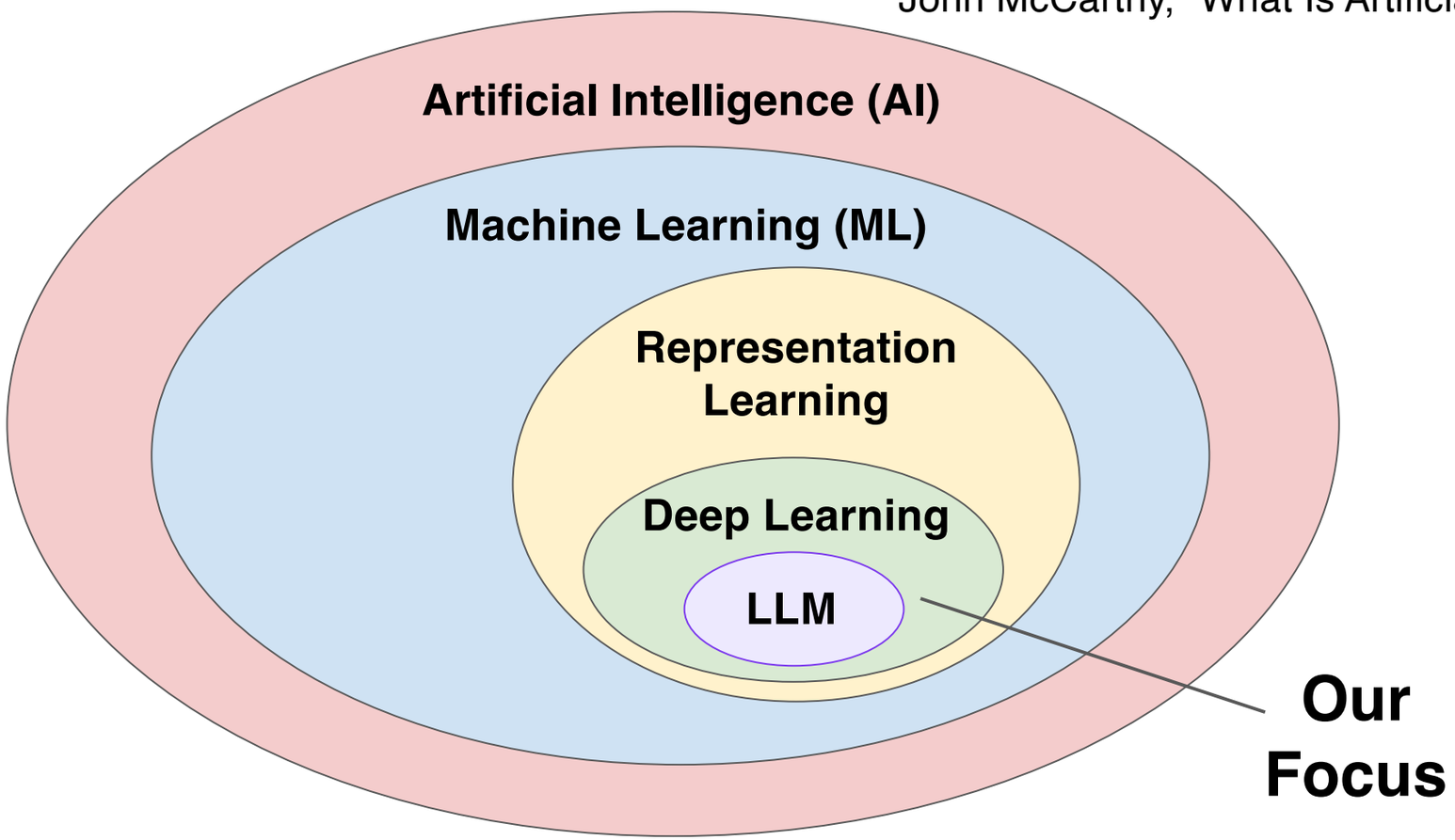


- **AI is "eating the world,"** rapidly transforming every domain

- **Much of AI's progress is constrained by hardware**
  - Modern models demand massive compute, memory capacity & bandwidth, network bandwidth, and energy
  - GPUs and AI accelerators are costly, supply-limited, and challenging to deploy at scale
  - To keep scaling AI, we need both smarter use of today's hardware and better designs for future systems

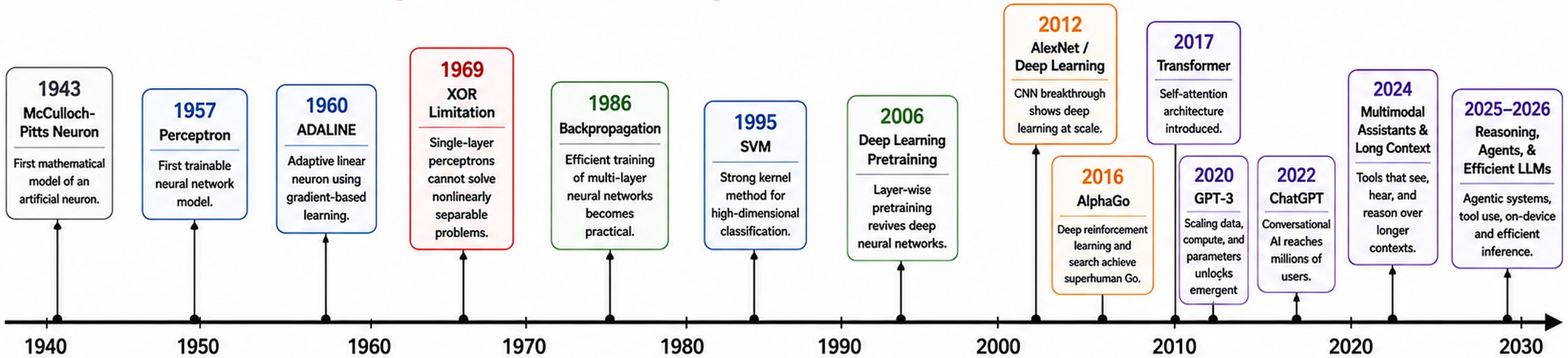
A Taxonomy

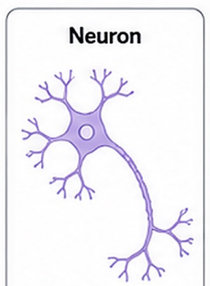
> *AI is the science and engineering of making intelligent machines, especially intelligent computer programs … Intelligence is the computational part of the ability to achieve goals in the world*
>
> John McCarthy, "What Is Artificial Intelligence?", 2007

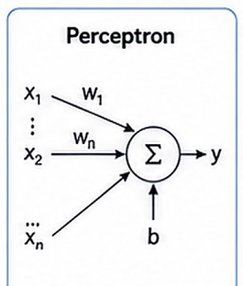


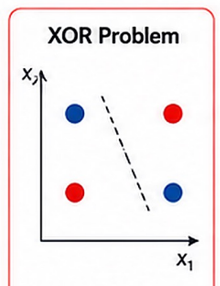
[1] John McCarthy, together with Marvin Minsky, Nathaniel Rochester, and Claude Shannon, first proposed and used the term "Artificial Intelligence" in a proposal for a Dartmouth summer research project.

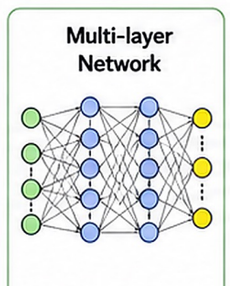
A Timeline of (Mainstream) AI

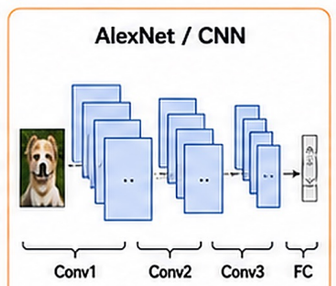


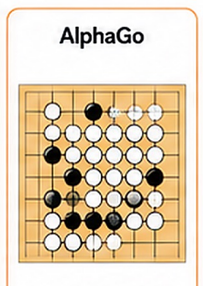


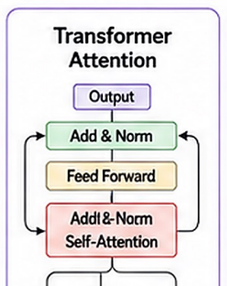


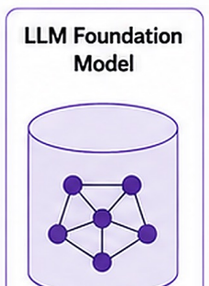


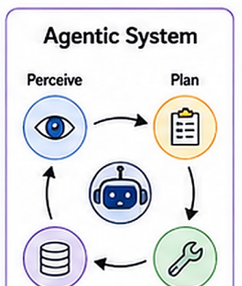






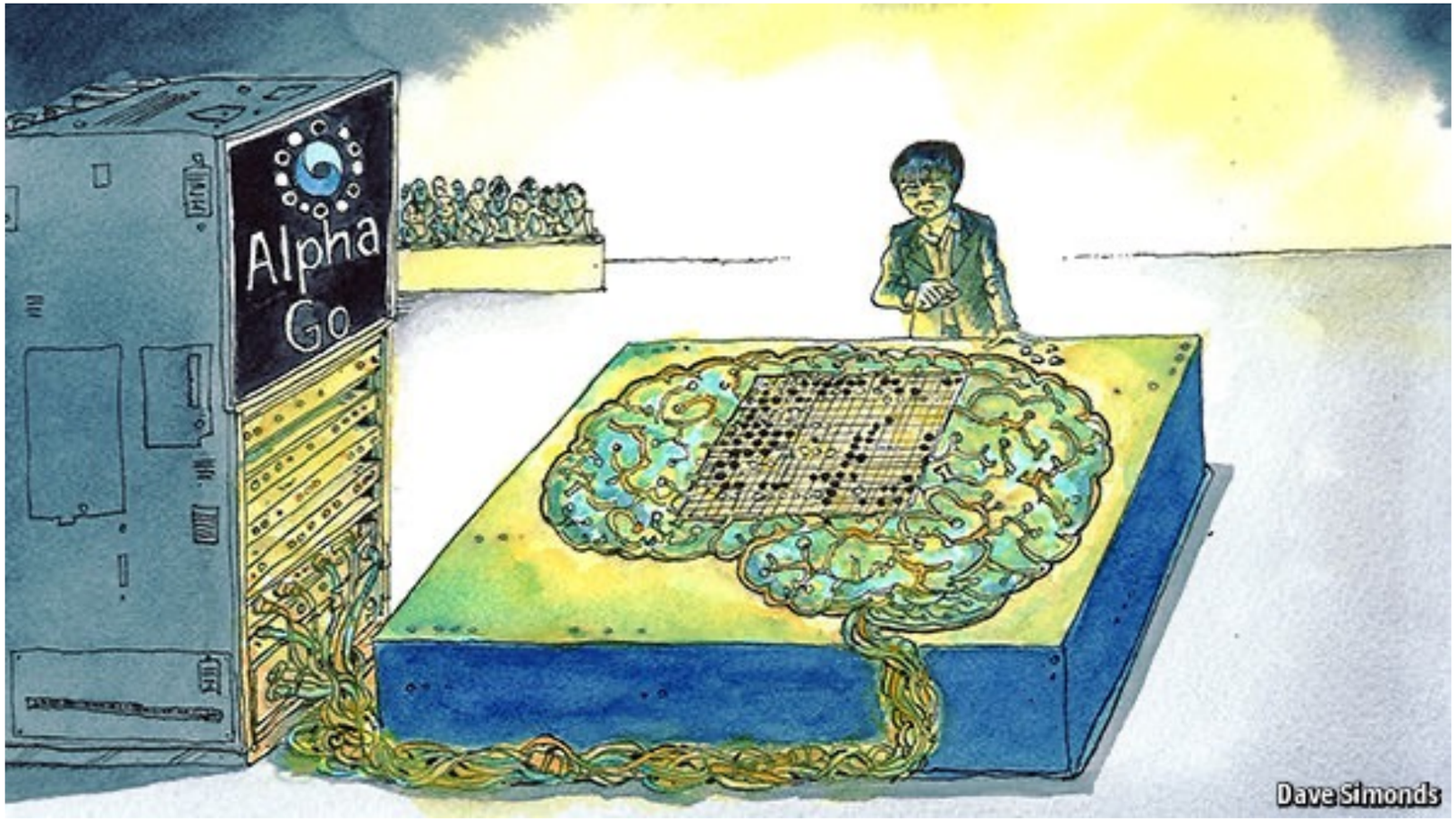






The Game Changing Game AI

(March 2016) Google's AlphaGo won the match against Go top player Lee Sedol



https://economist.com/news/science-and-technology/21694540-win-or-lose-best-five-battle-contest-another-milestone

The Large and Impressive Language Models

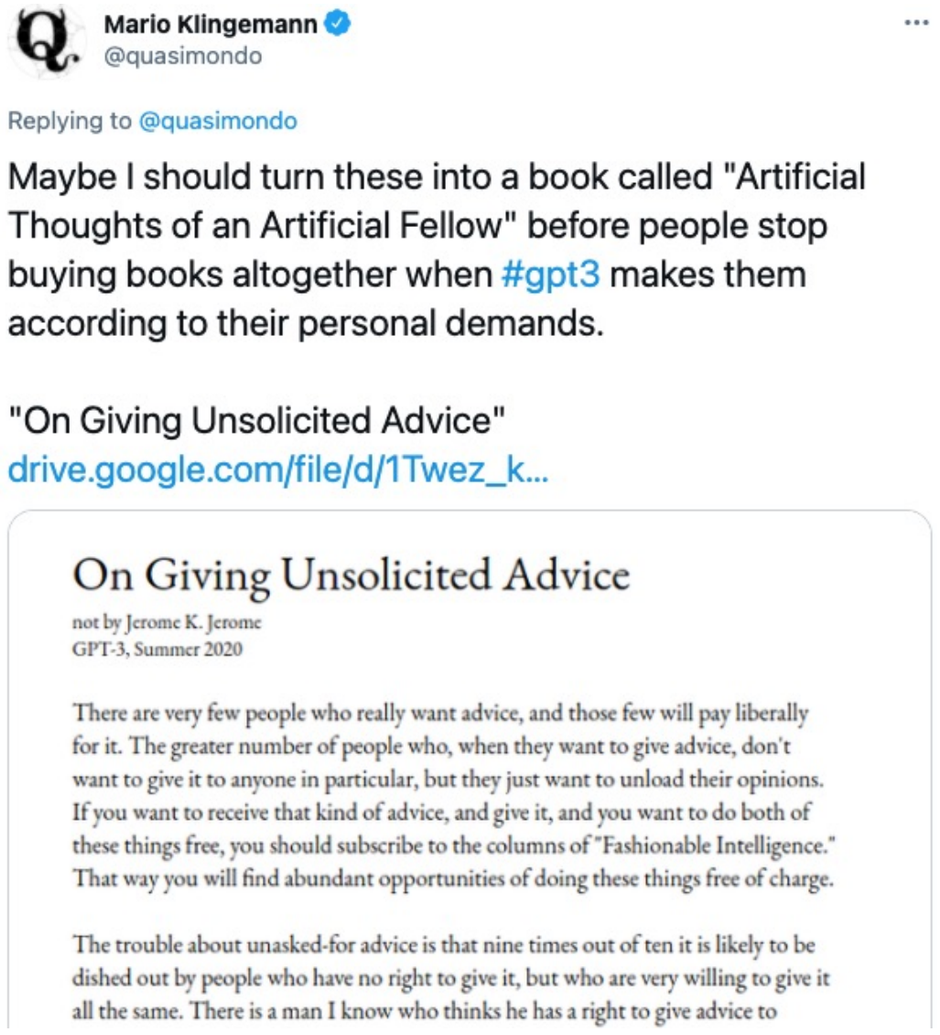
Artificial intelligence / Machine learning

OpenAI's new language generator GPT-3 is shockingly good—and completely mindless

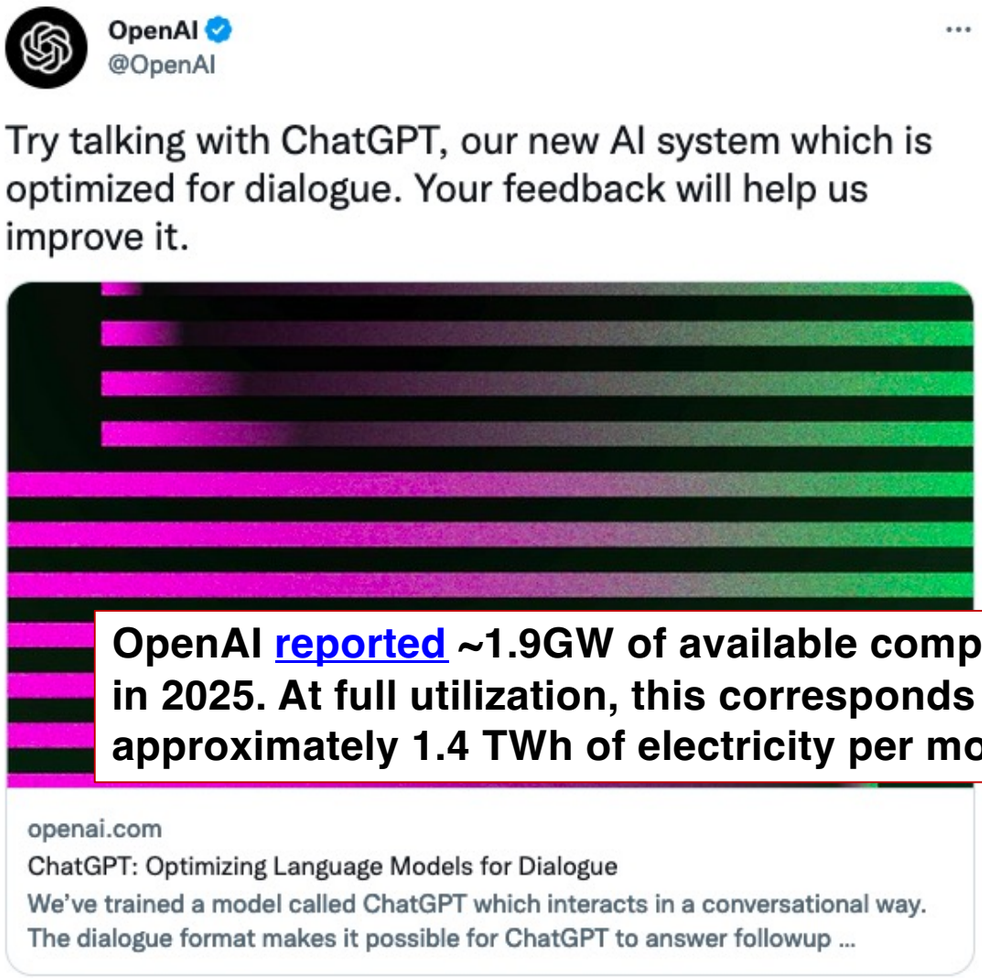
The AI is the largest language model ever created and can generate amazing human-like text on demand but won't bring us closer to true intelligence.

by **Will Douglas Heaven** July 20, 2020

Training GPT-3 is estimated to cost at least **$4.6 million**



The “Intelligent” Chatbot!



OpenAI reported ~1.9GW of available compute capacity in 2025. At full utilization, this corresponds to approximately 1.4 TWh of electricity per month.

NEWS EXPLAINER | 09 December 2022

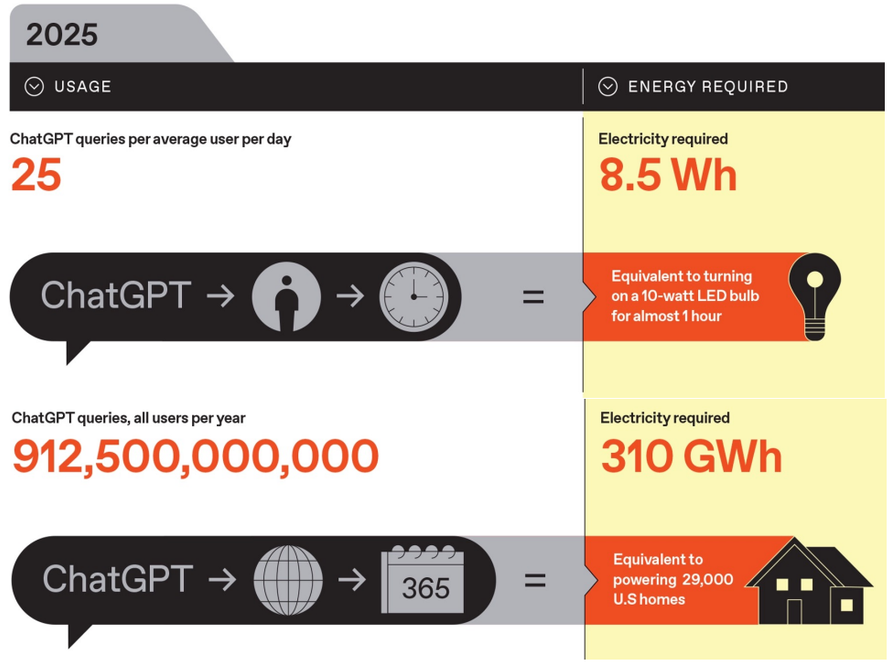
AI bot ChatGPT writes smart essays – should professors worry?

The bot is free for now and can produce uncannily natural, well-referenced writing in response to homework questions.

NEWS | 12 January 2023

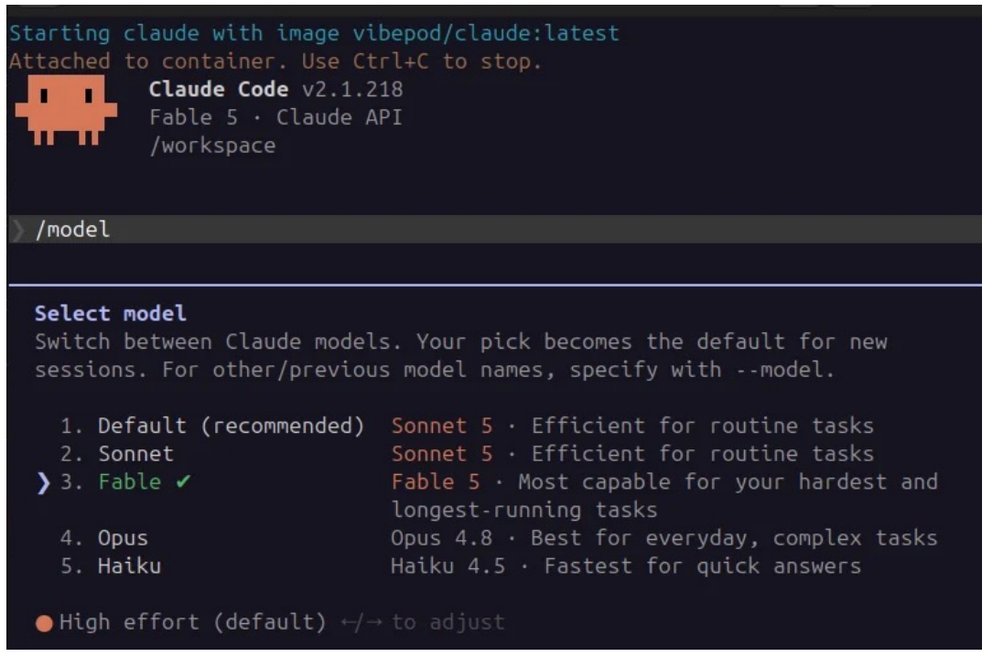
Abstracts written by ChatGPT fool scientists

Researchers cannot always differentiate between AI-generated and original abstracts.



source: spectrum.ieee.org/ai-energy-use

The “Coding” Agent That Does More Than Code



Coding agents turn software development into a closed-loop AI systems workload.

How AI is transforming work at Anthropic (Dec 2025)

1. **Anthropic engineers and researchers use Claude most often for fixing code errors and learning about the codebase.** Debugging and code understanding are the most common uses (Figure 1).
2. **People report increasing Claude usage and productivity gains.** Employees self-report using Claude in 60% of their work and achieving a 50% productivity boost, a 2-3x increase from this time last year. This productivity looks like slightly less time per task category, but considerably more output volume (Figure 2).
3. **27% of Claude-assisted work consists of tasks that wouldn't have been done otherwise**, such as scaling projects, making nice-to-have tools (e.g. interactive data dashboards), and exploratory work that wouldn't be cost-effective if done manually.
4. **Most employees use Claude frequently while reporting they can “fully delegate” 0-20% of their work to it.** Claude is a constant collaborator but using it generally involves active supervision and validation, especially in high-stakes work—versus handing off tasks requiring no verification at all.

Anthropic says its AI models hacked 3 organizations during testing

Nation Jul 31, 2026 1:40 PM EDT

BUSINESS INSIDER Aug 13, 2026

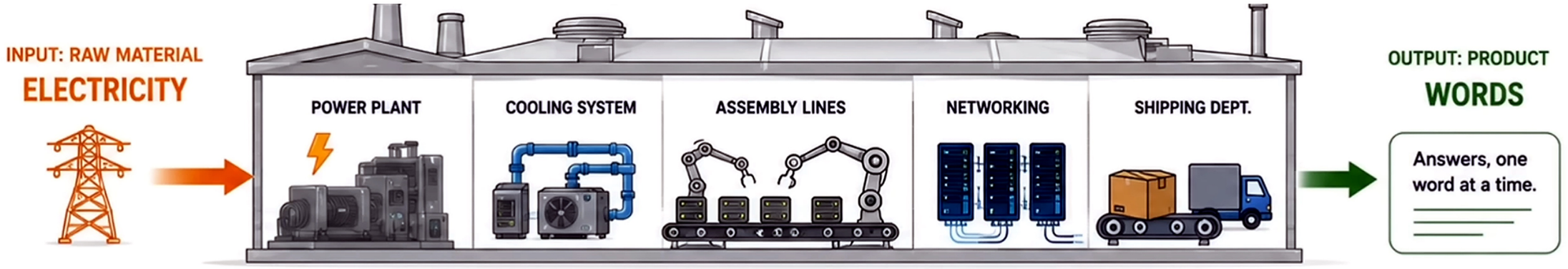
Techies have concerns about Claude's hidden watermark. Anthropic has some answers.

POLITICS & POLICY

The Fed rang the alarm about Anthropic’s Mythos AI model — but had to go months without it

PUBLISHED TUE, JUL 21 2026·6:05 PM EDT | UPDATED WED, JUL 22 2026·8:46 AM EDT

Modern AI Datacenters: An Analogy



AI turns compute into tokens, much like a factory turns electricity into products

THE TOKEN
the product

A token is a small chunk of text, roughly 3/4 of a word. Language models read and produce tokens, then assemble them into fluent answers.

"Modern AI Dacacenters"

1 2 3 4

~= 4 tokens

THE FLOP
the labor

A FLOP is one floating-point operation: one Multiply or one Add. Producing one token often requires hundreds of billions of FLOPs.

x + → 1

TRAINING vs. INFERENCE
building the factory vs. running it

Training

Adjusts >1 trillion parameters using massive datasets until the model learns

Frontier models take months to train on tens of thousands of chips, costing $100M+ per model

Inference

Runs the factory every time a prompt asks for an answer.

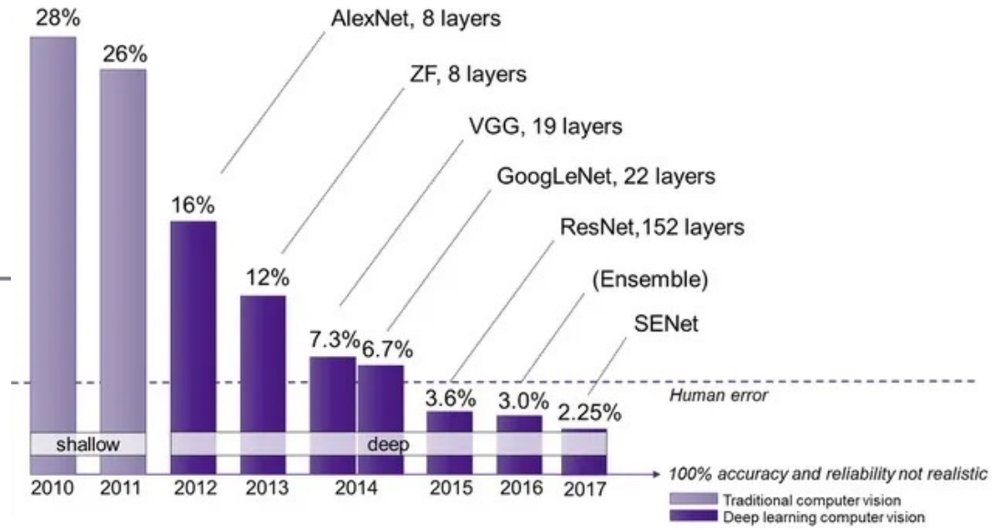
Happens billions of times per day (~2/3 of AI compute in 2026)

OpenAI inference bill is ~$14B in 2026

Slide adapted from “The Entire AI Data Center Explained – From Electricity to ChatGPT” by Leo Cui [YouTube link]

Why Did AI Take So Long to Become Useful?

1. **Computers were too slow**
CPUs not best suited for heavy/parallel arithmetic
2. **Training data was insufficient**
Need a lot of data for high accuracy



DNN Timeline

- 1940s: Neural networks were proposed
- 1960s: Deep neural networks were proposed
- 1989: Neural networks for recognizing hand-written digits (LeNet)
- 1990s: Hardware for shallow neural nets (Intel ETANN)
- 2011: Breakthrough DNN-based speech recognition (Microsoft)
- 2012: DNNs for vision start supplanting hand-crafted approaches (AlexNet)
- 2014+: Rise of DNN accelerator research (Neuflow, DianNao…)

source: V. Sze et al. Efficient Processing of Deep Neural Networks

Enabling breakthroughs

- **Big data**
- Powerful hardware: **GPGPU** ("General-Purpose" GPUs)

Computer Hardware: Then and Now

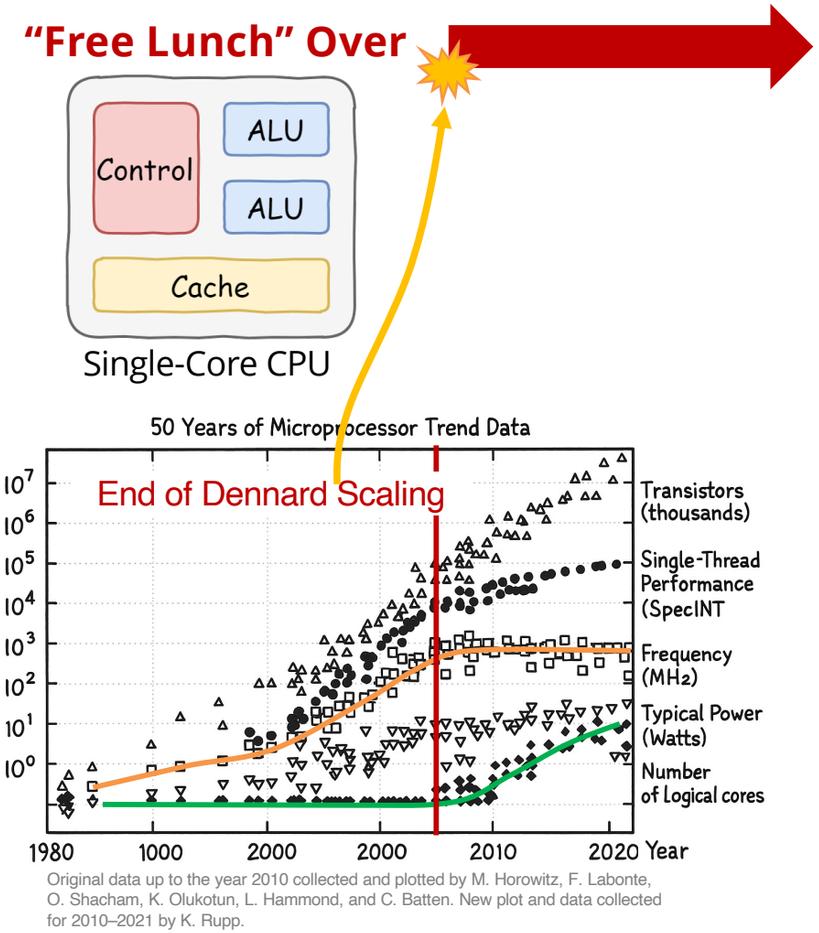
Cray 1 Supercomputer
(1975)
160 MegaFLOPS
115 kW
$7.9M ($40M today)

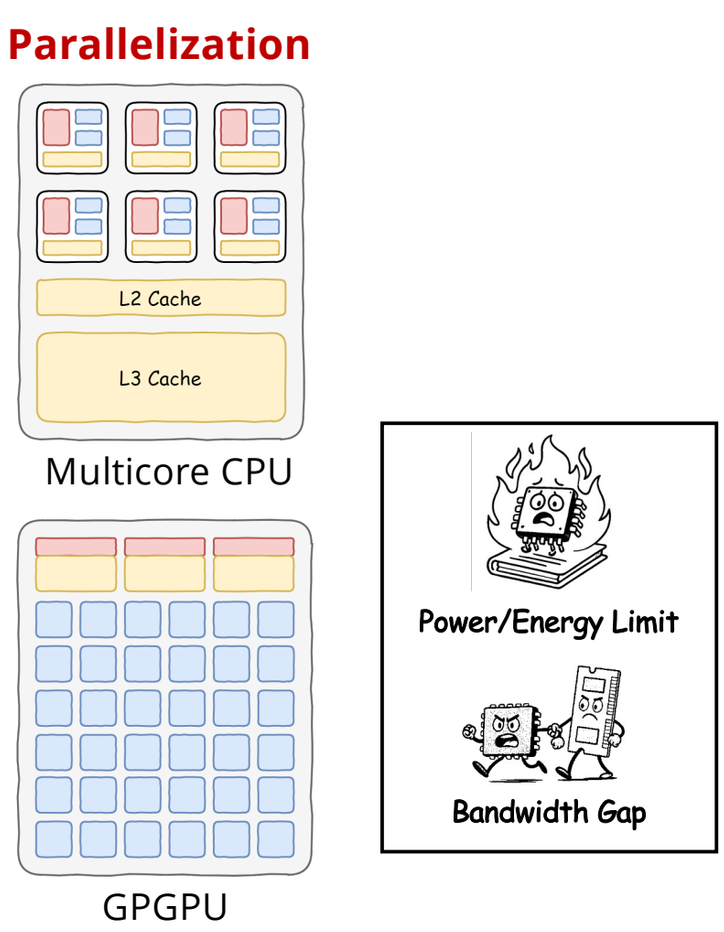
>1,000,000,000x Improvement in FLOPS/W

NVIDIA H100 GPU
(2022)
1.5 PetaFLOPS
700W
$30K

Credit: slide adapted from Jonathan Ragan-Kelley's PLDI 2024 keynote

Evolution of Digital Computing (single device)





Power-Constrained Modern Computers

<<1W/chip ~1W/chip ~15W/chip ~50W/chip ~100W/chip >100W/chip

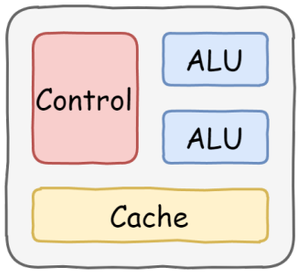
$$Power = \frac{Energy}{Second} = \frac{Energy}{Op} \times \frac{Ops}{Second}$$

fixed budget

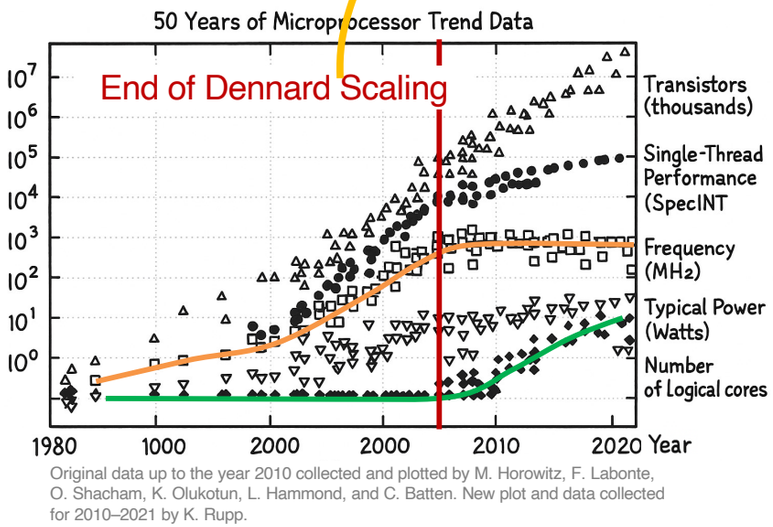
To increase performance (Ops/sec) in a power-constrained regime, energy per operation must decrease—in other words, **energy efficiency** (Ops/Joule) **needs to improve!**

Evolution of Digital Computing (single device, cont'd)

"Free Lunch" Over

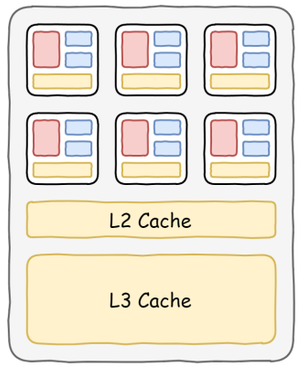


Single-Core CPU

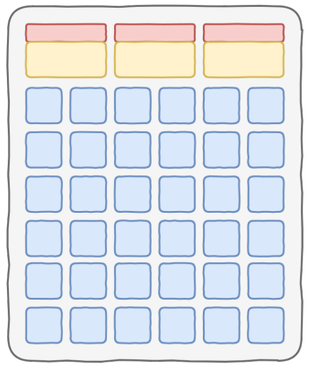


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. New plot and data collected for 2010–2021 by K. Rupp.

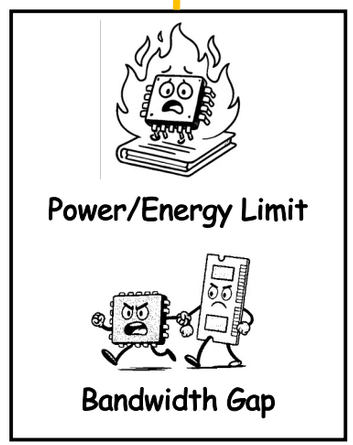
Parallelization



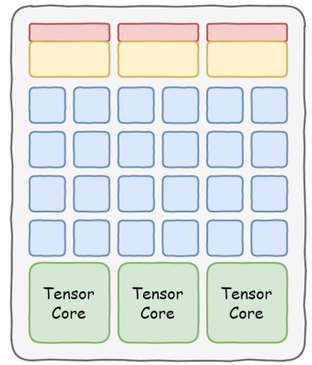
Multicore CPU



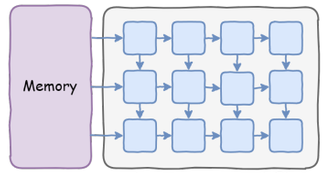
GPGPU



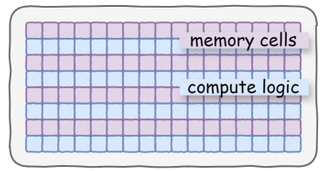
Specialization



GPU + Tensor Cores



Domain-Specific **Accelerators**



Near/In-Memory Processing

What's AI Hardware

- **Accelerators:** specialized hardware designed to perform specific computational tasks (or *workloads*) faster and more efficiently than general-purpose processors

- **AI hardware (in the context of this course)**
  - A class of accelerators designed to efficiently execute modern AI workloads
  - Expected to deliver high performance and energy efficiency for AI inference and/or training

Why AI Hardware Requires a New Course

- **General purpose vs. domain specific**
  - Traditional architecture courses focus on CPUs: flexible, programmable, and designed to run arbitrary workloads
  - AI accelerators are customized for deep learning, e.g., tensor operations with high parallelism & low-precision arithmetic

- **Hardware-software co-design**
  - CPUs use stable ISAs that largely decouple software from hardware implementation
  - AI accelerators evolve quickly across hardware, ISA, compilers, and programming models; achieving high performance requires full-stack co-design

Key Theme: Data Specialization

- The defining characteristic of modern AI accelerators is not just faster arithmetic—it is moving, organizing, and reusing data efficiently

- Why specialize data?
  - **Data movement dominates** latency and energy
  - **AI workloads are highly structured**, exposing predictable compute and data access patterns
  - **AI models tolerate approximation** because training compensates for reduced precision and other structured approximations

- How do we exploit these properties?
  - **Representation:** Specialized numerical formats (BF16, FP8, INT8, ...)
  - **Organization:** Data layout, tiling, sharding, …
  - **Execution:** Collective communication, operator fusion, pipelining, …
  - **Algorithm:** Quantization, sparsity, pruning, …

Course Organization

Refer to syllabus for course organization details

Course Syllabus
ECE 4950/6950 Special Topics in ECE: AI Hardware
Fall 2026, TuThu 2:55pm-4:10pm, Upson 142

---

1. Course Information

Lectures: Tuesday & Thursday 2:55pm-4:10pm, 142 Upson Hall
Website: https://www.csl.cornell.edu/courses/ece4950fa26
CMSX: https://cmsx.cs.cornell.edu
Ed: https://edstem.org/us/courses/100429

Instructor: Zhiru Zhang, zhiruz@cornell.edu
Office Hours: Thursday 5:00-6:00pm, Online

Course Texts:
- Lecture slides/notes posted on the course website
- Austin et al., How to Scale Your Model, Google DeepMind, online, 2025.

Supplementary Materials:
- Required and supplementary readings posted with the course schedule.

2. Course Description and Objectives

Artificial intelligence (AI) is increasingly shaped by the capabilities and limits of the hardware systems used to train, serve, and specialize modern models. As deep neural networks (DNNs) and large language models (LLMs) continue to grow in scale, the design of efficient AI systems requires a working understanding of algorithms, models, arithmetic, memory hierarchy, parallel execution, and accelerator architecture. This course studies the principles and practice of AI hardware across both general-purpose and specialized platforms.

The course offers an introductory exploration of AI hardware design and analysis, connecting first-order performance models with hands-on implementation. Example topics include DNN computation, hardware specialization, roofline performance modeling, systolic arrays, TPUs, GPUs, kernel programming, LLM inference and serving, sparsity, quanti- zation, and specialized LLM accelerators. Through lectures, readings, homework, labs, and a final project, students will develop the skills to reason about AI hardware from workload characteristics through architecture and implementation, and to make informed design choices across performance, efficiency, programmability, and scalability.

A List of Major Topics

- **Background**
  - Neural network basics
  - Hardware specialization
  - Roofline analysis
- **Hardware**
  - Systolic arrays, TPUs, GPUs, NPUs
  - Emerging LLM accelerators
- **Model architecture, mapping, and optimization**
  - LLM: GPT model, inference, serving, training
  - Kernel programming
  - Sparsity, quantization

Preferred Background

- Prerequisites (undergraduate level)
  - Digital logic and basic computer architecture concepts (e.g., multipliers, clock, memories, pipelining)
  - Basic linear algebra

- Helpful background (not required)
  - AI and deep learning fundamentals
  - RTL design for ASIC or FPGA
  - Compiler optimization

Learning Outcomes

- Understand key trade-offs in AI accelerators: performance, efficiency, and flexibility

- Develop a computational view of modern AI models
  - Core compute patterns in DNNs and LLMs
  - Compute bound vs. memory bound
  - Accuracy vs. efficiency trade-offs

- Gain hands-on experience building AI accelerators and mapping LLM workloads to hardware

NOT Our Goals

- Cover the full breadth of AI models (e.g., advanced LLMs, diffusion, world models, GNNs, etc.)
- Teach you to train and fine-tune LLMs or build new agentic workflows
- Make you an expert programmer for GPUs and other AI hardware

The course focuses primarily on single-node inference rather than training or distributed systems

Assignments

- Four problem sets (24%)

- Four lab assignments (21%)
  - Design a mini tensor processing unit (MiniTPU), prototype it on an FPGA, then deploy a small LLM

  - Experiments to be conducted on **ecelinux** servers
    - % ssh -X <netid>@ecelinux-01.ece.cornell.edu
    - Necessary tools will be installed in common directories

Quizzes

- Quizzes (6%)
  - Typically 1–2 pop quiz questions in most lectures
  - **TWO lowest scores will be dropped**

- Selected lectures
  - Some lectures may include additional quiz questions
  - Students are expected to (1) read the assigned paper or book chapter before class (2) answer the additional quiz questions
  - Students will be notified at least one week in advance

Exams

- In-class prelim (15%)
  - Open notes & open book
  - Tentative date: Thursday October 8th

- Final exam (20%)
  - Format/date TBD

Project – 10%

- Hands-on project exploring a topic in AI hardware
  - Teams of 3–4 students, depending on class size
- Project options
  - Program a commercial AI accelerator (AWS Trainium)
  - Extend MiniTPU
- Timeline (~3 weeks)
  - Suggested project topics posted in early November
  - Meetings with course staff to track progress
  - Report and code due before the final exam

Before Next Lecture

- Action items
  - **Read through the course syllabus**
  - **Verify your login on ecelinux**
    - ssh -X <netid>@ecelinux.ece.cornell.edu
  - **Activate your Cornell GitHub account**
    - https://github.coecis.cornell.edu/

Acknowledgements

- The lecture slides contain/adapt materials from
  - ECE 5545 by Prof. Mohamed Abdelfattah (Cornell Tech)
  - Yixiao Du (PhD, Cornell ECE)