

# Introduction to High-Performance Computing Bootcamp

**Project 6: Evaluating Large Language Models for HPC Education**

Murat Keçeli

## Peer Mentors



[Alexis Anderson](#)  
[Tuskegee University](#)



[Oluwaseun Ajayi](#)  
[Argonne National Laboratory/Illinois Institute of Technology](#)



[Samira Begum](#)  
[Swarthmore College](#)

# Project Overview

- The goal of the project is to learn more about HPC and LLMs by exploring and evaluating how LLMs can assist in learning and teaching fundamental HPC concepts.
- Students will use LLMs to generate explanations, tutorial content, and answers to basic HPC-related questions.
- They will then evaluate the accuracy, clarity, and effectiveness of these responses, comparing them against official documentation and expert sources.
- The expected final products are:
  - An evaluation data set (question and answers) for HPC
  - A chatbot customized for HPC

# Project Overview

## ◆ Foundations

- High-Performance Computing (HPC) basics and applications
- Artificial Intelligence (AI) fundamentals
- Large Language Models (LLMs) — capabilities and use cases

## ◆ LLM Ecosystem

- Major LLM providers (e.g., OpenAI, Anthropic, Meta, Google)
- Popular LLM frameworks (e.g., LangChain, HF)
- API access and integration into applications

## ◆ Model Development & Capabilities

- Pretraining, instruction tuning, and alignment techniques
- Multi-modal capabilities (text, images, audio, etc.)
- Prompt engineering and optimization strategies
- Retrieval-Augmented Generation (RAG)
- AI Agents and autonomous workflows

## ◆ Working with LLMs

- Deep research using LLM-assisted methods
- Coding tools for building AI applications
- Evaluation metrics and methods for LLM performance
- Understanding limitations and potential risks

# High-Performance Computing

HPC is the use of powerful computers and parallel processing techniques to solve problems that are too large, too complex, or too time-consuming for regular computers.

- **Computing Power** – HPC systems (clusters or *supercomputers*) combine thousands or even millions of processing cores to work together.
- **Specialized Hardware** – Uses high-speed interconnects, large memory bandwidth, and accelerators (like GPUs) to handle demanding workloads.
- **Parallel Processing** – Tasks are divided into smaller parts and run simultaneously, speeding up computation dramatically.
- **Applications** – Scientific simulations, climate modeling, molecular dynamics, AI training, big data analytics, financial risk modeling, etc.

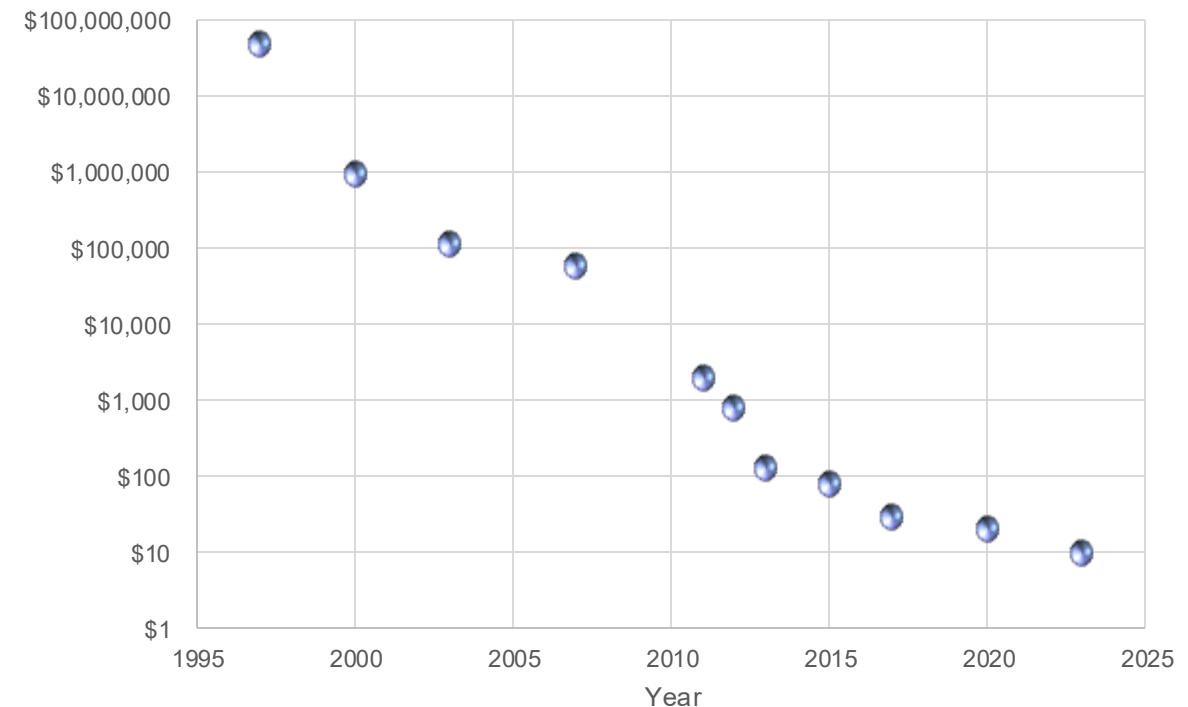
## Analogy:

If a regular computer is like one person solving a puzzle, HPC is like thousands of people each working on a piece and instantly sharing their progress to solve a more complex puzzle in a short time.

**Faster, cheaper, and more abundant computing**



Hardware Cost for TFlop/s



<https://en.wikipedia.org/wiki/FLOPS>


# Exascale Era



# Artificial Intelligence & Machine Learning

 **Artificial Intelligence (AI):** Any technique that enables computers to perform tasks we'd consider intelligent if a human did them.

 **Machine Learning (ML):** Instead of hand-coding rules, we train models from data.

 **Neural Networks:** Flexible function approximators made of layers. Their **parameters** are adjusted by training (e.g., backpropagation) so outputs match patterns in the data.

Hornik, K.; Stinchcombe, M.; White, H. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* **1989**, 2 (5), 359–366  
[https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)

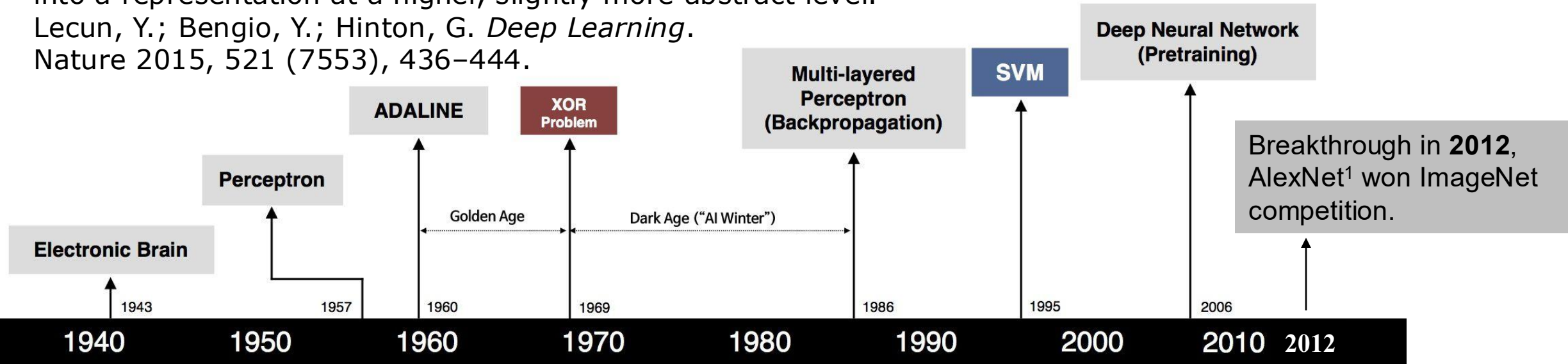
Think of a neural network as a big configurable calculator .

Provide many examples → Tweak its dials (parameters) → it gets better at predicting outputs.

# Big data & compute enabled deep learning

DL methods are “representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level.”

Lecun, Y.; Bengio, Y.; Hinton, G. *Deep Learning*.  
Nature 2015, 521 (7553), 436–444.



S. McCulloch – W. Pitts



F. Rosenblatt



B. Widrow – M. Hoff



M. Minsky – S. Papert



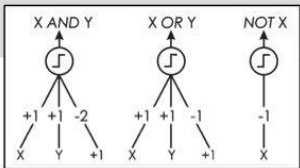
D. Rumelhart – G. Hinton – R. Williams



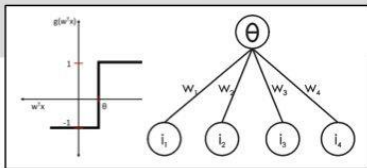
V. Vapnik – C. Cortes



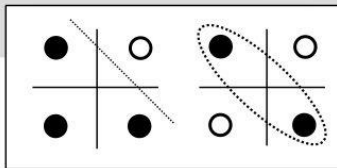
G. Hinton – S. Ruslan



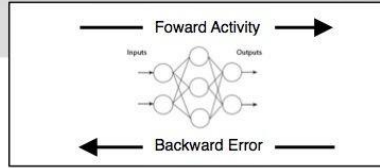
- Adjustable Weights
- Weights are not Learned



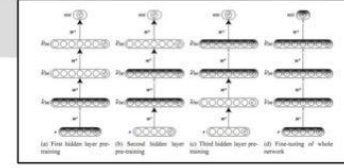
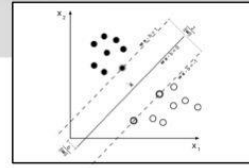
- Learnable Weights and Threshold



- XOR Problem



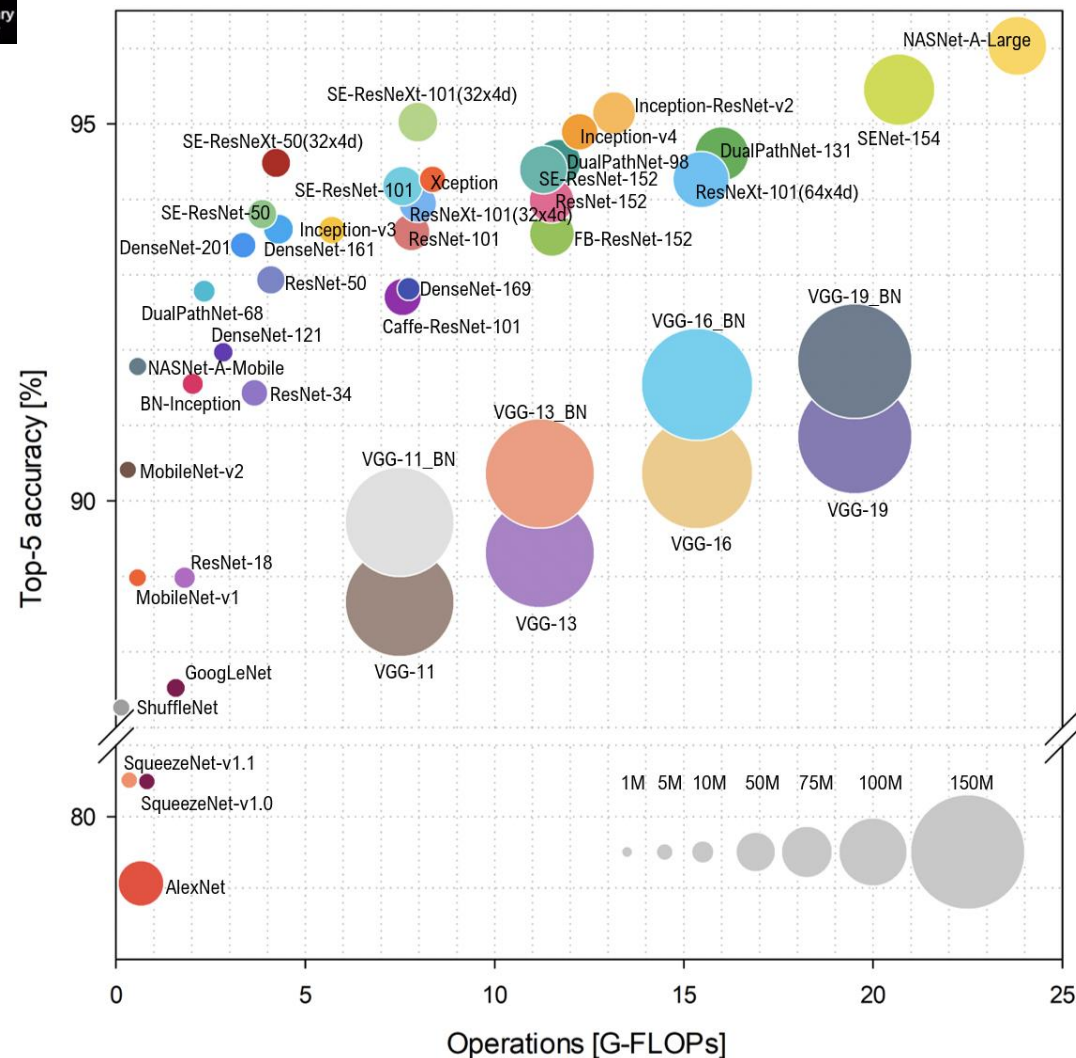
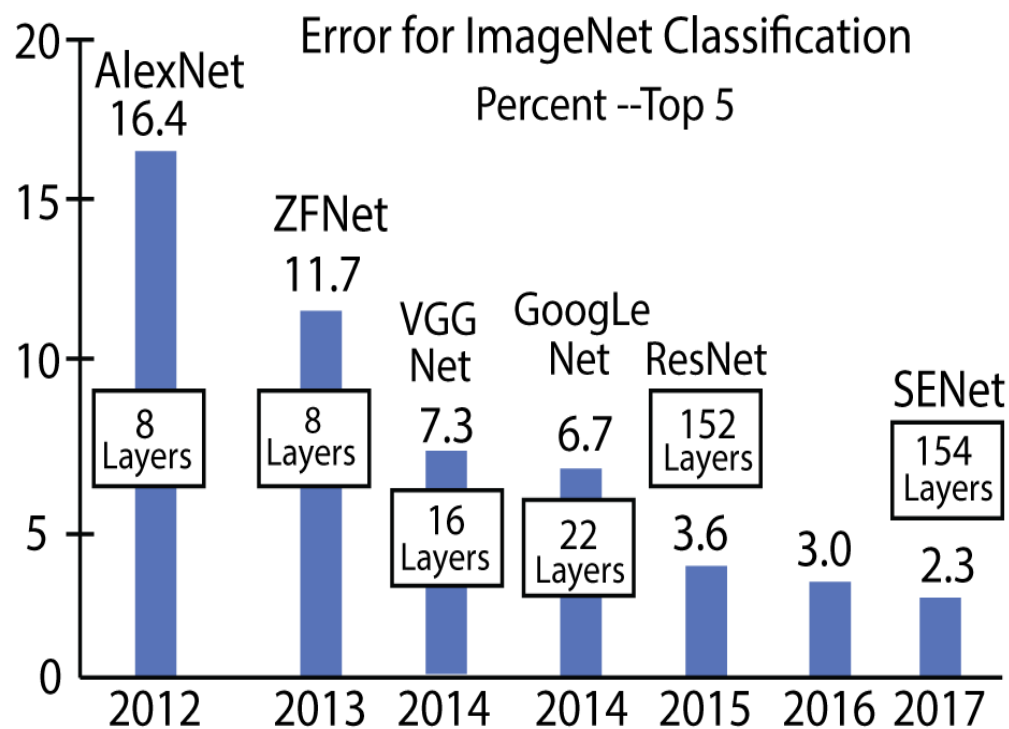
- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting
- Limitations of learning prior knowledge
- Kernel function: Human Intervention



- Hierarchical feature Learning

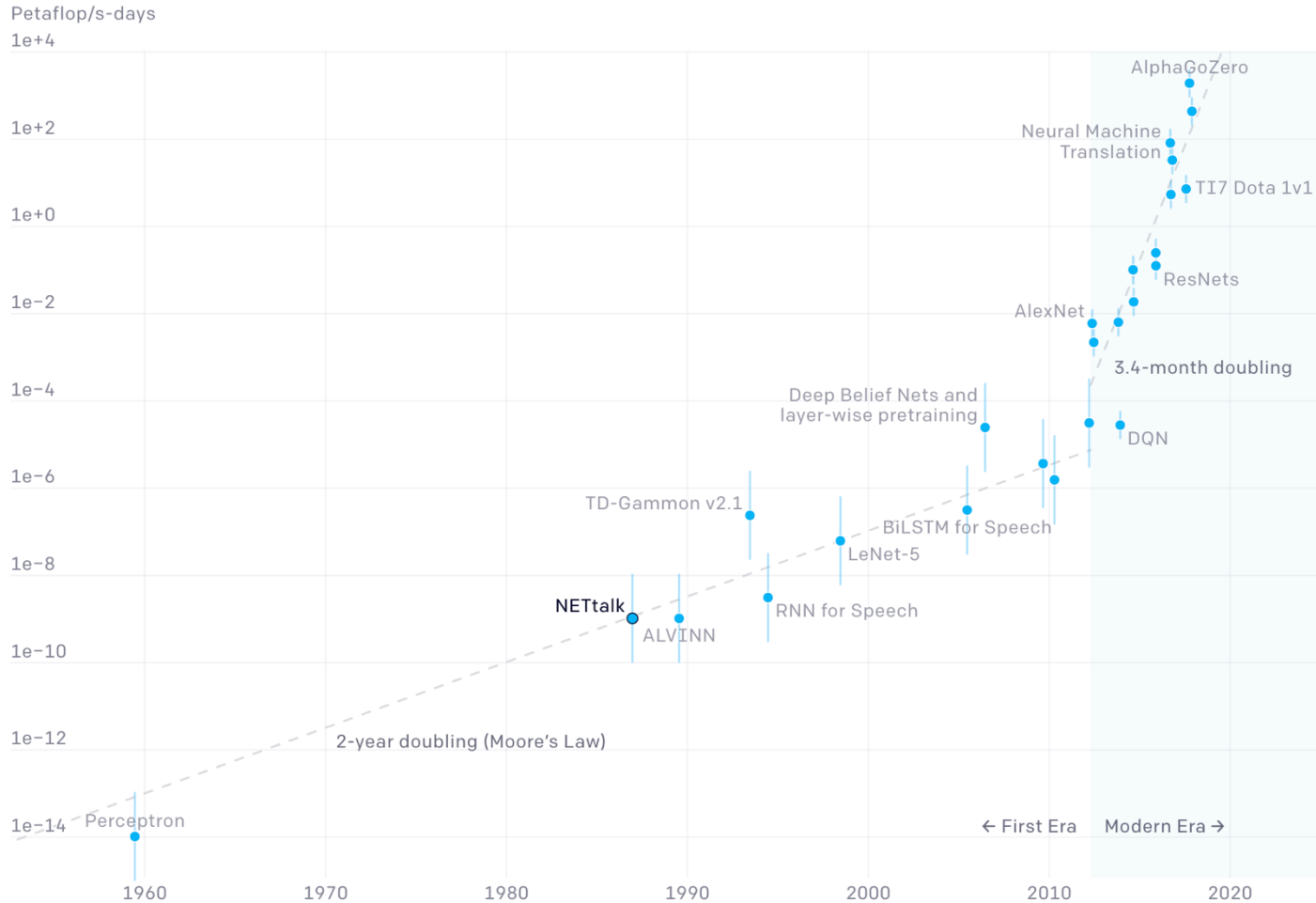


# Scaling Up Deep Learning





# Why do we need HPC?



**AlexNet to AlphaGo Zero:  
300,000 x Compute**

**Why do we need distributed deep learning?**

- Increase in the amount of training data
- Increase in the complexity of the model
- Hyperparameter optimization
- Reduce training and inference time
- Coupling with simulations

# Large Language Models

Large Language Models are **neural networks trained to predict the next token** in a text sequence.

$$P(\text{token}_n | \text{token}_1, \text{token}_2, \dots, \text{token}_{n-1})$$

A **token** is a chunk of text such as a word, part of a word, or even a single character that an LLM uses as a basic unit for processing and predicting language.

They learn patterns from **massive text corpora** and can generate coherent answers, summaries, code, and more.

💡 **Imagine:** A giant, super-smart program trained on billions of books, articles, and all data from internet.  
That's essentially what an **LLM** is!

Enter text:

Einstein is a famous











Einstein is a famous

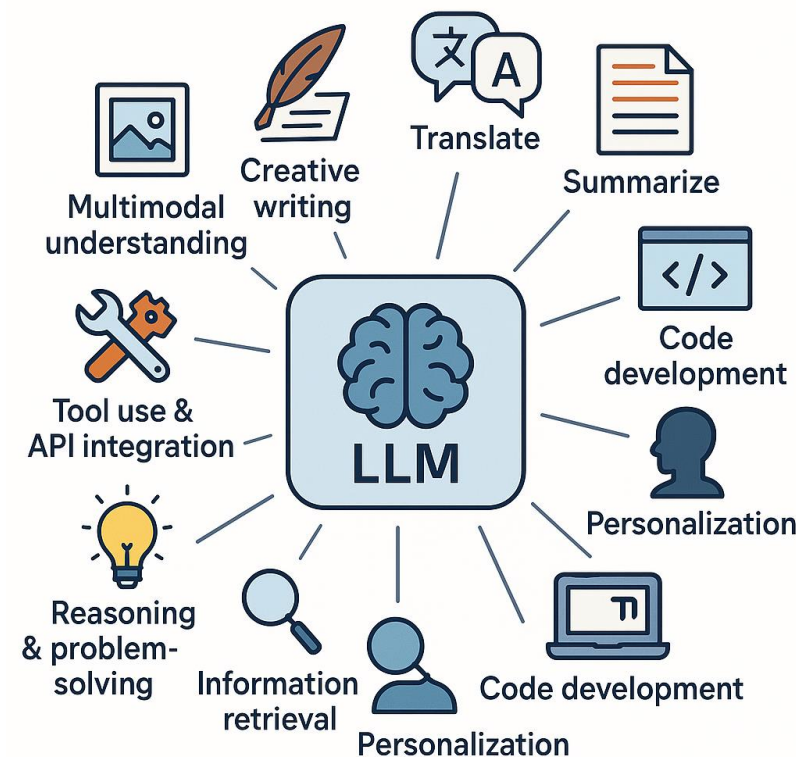
36 11962 318 257 5863

Prediction

#	probs	next token ID	predicted next token
0	7.31%	33013	physicist
1	5.54%	11444	scientist
2	2.02%	3785	figure

# LLM Capabilities

-  **Creative writing:** Poems, stories, songs, scripts, essays, emails.
-  **Translate:** Convert text between languages.
-  **Summarize:** Condense long documents into key points.
-  **Answer questions:** Based on training data, in-context data, or retrieved data.
-  **Code development:** Generate or debug code snippets, functions, or entire programs from natural language.
-  **Multimodal understanding:** Interpret and generate content from text, images, audio, or combinations.
-  **Tool use & API integration:** Plan and execute tasks using external tools or databases.
-  **Information retrieval:** Search and extract relevant facts from large datasets or the web.
-  **Reasoning & problem-solving:** Solve logic, math, and science problems with step-by-step reasoning.
-  **Personalization:** Adapt tone, style, and complexity to the user's needs.



# LLM Basics I

## **Model size:**

Larger models (e.g., GPT 4 with hundreds of billions of parameters) capture more complex patterns and nuances in language, enabling better reasoning and broader knowledge coverage.

Example: A small model might miss subtle sarcasm in a sentence, while a larger one could interpret it correctly.

## ☐ **Context window:**

The model can only “remember” a certain number of tokens at a time; older content falls out of scope.

Example: GPT-3.5 ~4k tokens, GPT-4 ~8k or more.

## **Temperature:**

Controls randomness in output. (OpenAI API allows 0 to 2)

Low values (e.g., 0) make answers more deterministic; high values (e.g., 0.9) encourage creativity.

Example: With temperature 0.2, “Write a sentence about cats” might always yield “Cats are furry animals.” At 0.9, it might produce “In the golden haze, the cat danced among falling leaves.”

## **Logprobs:**

Returns the log probabilities for each output token, useful for analysis or uncertainty estimation.

Example: Can show that the model had 95% confidence in choosing a specific word.

## **Max completion tokens:**

Limits how many tokens the model can generate in the response. Helps control cost and length.

Example: Setting max tokens=100 stops the model after roughly a paragraph.

## **Multimodal capabilities:**

Some models handle text, images, or audio in the same conversation.

Example: Uploading an image of a chart and asking, “Summarize the key trends.”

# LLM Basics II



## **Prompt optimization:**

Carefully crafting inputs to guide the model's behavior and improve reliability.

Example: Instead of “Tell me about HPC,” use “Describe HPC in a lecture format using bullet points.”



## **Chain of thought:**

Encourages the model to “think” step-by-step before answering, improving performance on reasoning tasks.

Example: “Let’s think step-by-step” before solving a math word problem often yields more accurate results.



## **Structured output:**

Requesting outputs in specific formats (JSON, tables, bullet points) for easier parsing and integration into other systems.

Example: Asking “Provide the answer as JSON with keys ‘capital’ and ‘population’” lets code directly consume the output.



## **Iterative refinement:**

Run multiple prompts to refine answers.

*Example:* First ask for an outline, then expand each section in follow-up prompts.



## **Reasoning effort:**

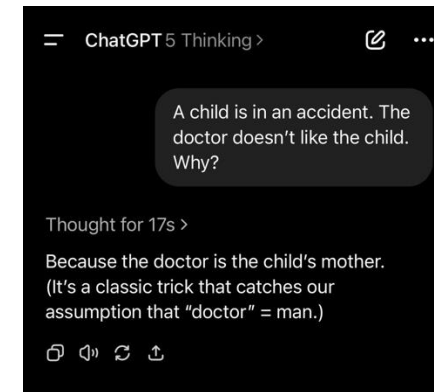
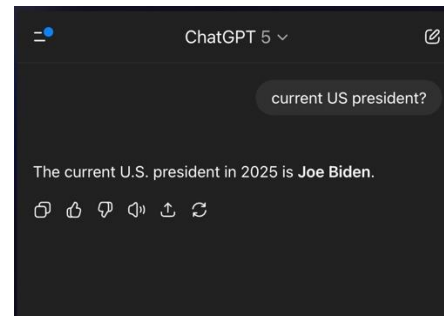
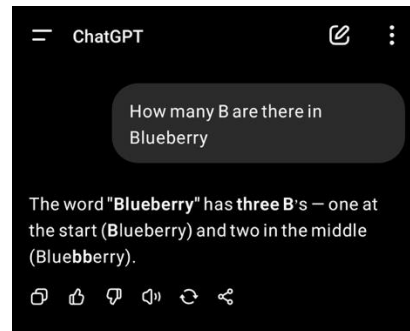
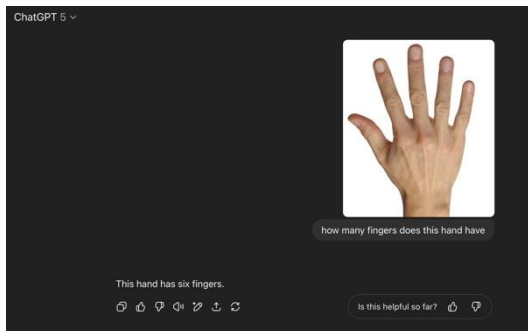
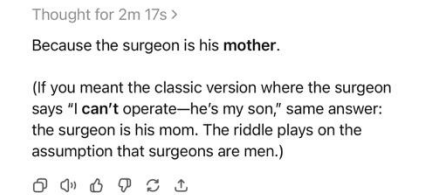
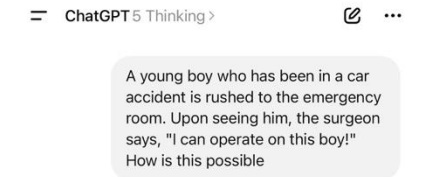
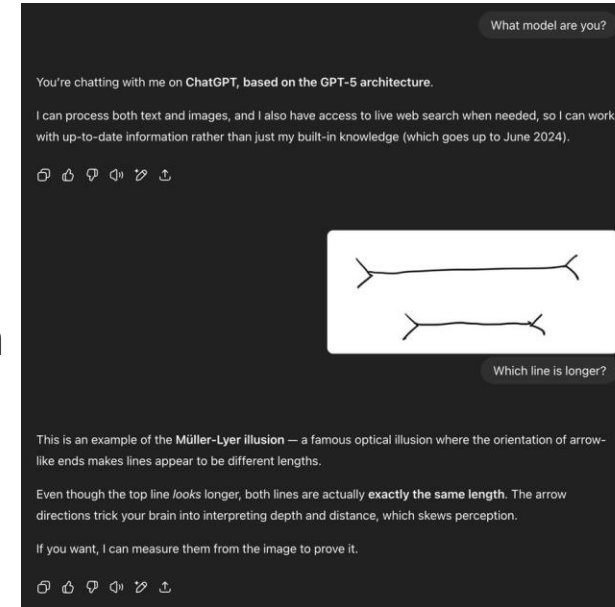
Controls how much effort the model spends on reasoning (minimal, low, medium, high).

Lower settings are faster and cheaper but may miss complex steps; higher settings are more thorough.

*Example:* High reasoning effort might break down a legal case into detailed logical steps, while low effort gives a short answer.

# LLM Limitations

- 🚫 Hallucinations: May produce plausible but false information
- 📅 Outdated knowledge: Limited to training data cut-off
- ⚖️ Biases: May reflect societal or cultural biases
- 📄 Context limits: Struggles with very long conversations
- 🤖 No true understanding: Mimics patterns, not comprehension
- 🎯 Input quality dependency: Prompt clarity impacts results
- 🔍 Opaque reasoning: Hard to explain specific answers
- 💻 Resource demands: Needs significant computing power



As written, there's no puzzle—any qualified surgeon could say, "I can operate on this boy."

If you meant the classic riddle (which usually says, "I can't operate on this boy—he's my son"), the answer is: the surgeon is the boy's **mother** (or more generally, a parent). The riddle plays on the assumption that a surgeon must be a man.

<https://www.realtimetechpocalypse.com/p/gpt-5-is-by-far-the-best-ai-system>  
<https://chatgpt.com/share/6898c767-d648-8012-98d2-e3aaabc41d6d>



# LLM Evaluation

Evaluating large language models is critical to ensure their outputs are accurate, reliable, safe, and aligned with intended use cases.



Accuracy: Measure factual correctness of outputs



Reasoning: Evaluate logical consistency and problem-solving



Coherence: Assess fluency, grammar, and readability



Relevance: Check alignment with prompt intent



Safety: Identify harmful or biased responses



Efficiency: Evaluate latency and computational cost



Robustness: Test performance under varied inputs

Proposed Methodology				
Techniques	MCQ Benchmarks	Open Response Benchmarks	Lab Style Experiments	Field Style Experiments
Main Goal	Testing knowledge breadth, basic reasoning in benchmark setting	Testing knowledge depth, planning, reasoning in benchmark setting	Testing of all capabilities in realistic setting	Capability overview, trend analysis and weakness diagnosis from realistic setting
Problem Type	Predetermined, Fixed Q&As with known solutions	Predetermined, Fixed free-response problems with known solutions	Individual human defined problems with (un)known solutions	Many human defined problems with (un)known solutions
Verification Type	Automatic response verification	Automatic or human response verification	Humans detailed analysis of AI models responses	Scalable automatic summary of human evaluation of AI models responses
Examples	Astro, Climate, AI4S, Existing benchmarks	SciCode, ALDbench	Algorithm research, PDE solving, Checkpoint design	AI models evaluation sessions with 100+ researchers
Cross Cutting Aspects	← Trust and Safety (ChemRisk), Uncertainty Quantification, Scalable Software Infrastructure (STAR) →			

# LLM Benchmarks



Text				4 days ago
Rank (UB)	Model	Score	Votes	
1	gpt-5	1481	3,182	
2	gemini-2.5-pro	1460	26,703	
2	o3-2025-04-16	1450	32,692	
3	chatgpt-4o-latest-20250326	1442	31,219	
4	gpt-4.5-preview-2025-02-27	1438	15,271	
5	grok-4-0709	1429	13,314	
5	qwen3-235b-a22b-instruct-2507	1428	4,831	
6	claude-opus-4-20250514-think...	1420	18,461	
6	kimi-k2-0711-preview	1420	12,400	
7	deepseek-r1-0528	1417	18,662	
View all				

<https://www.vellum.ai/llm-leaderboard>

<https://lmarena.ai/leaderboard>

	Rank	Type	Model	Average	IFEval	BBH	MATH	GPQA	MUSR	MMLU-PRO	CO <sub>2</sub> Cost
1	1	🔹	MaziyarPanahi/calme-3.2-instruct-78b	52.08 %	80.63 %	62.61 %	40.33 %	20.36 %	38.53 %	70.03 %	66.01 kg
2	2	🗨️	MaziyarPanahi/calme-3.1-instruct-78b	51.29 %	81.36 %	62.41 %	39.27 %	19.46 %	36.50 %	68.72 %	64.44 kg
3	3	🗨️	dfurman/CalmeRys-78B-Orpo-v0.1	51.23 %	81.63 %	61.92 %	40.63 %	20.02 %	36.37 %	66.80 %	25.99 kg
4	4	🗨️	MaziyarPanahi/calme-2.4-rys-78b	50.77 %	80.11 %	62.16 %	40.71 %	20.36 %	34.57 %	66.69 %	25.95 kg
5	5	🔹	huihui-ai/Qwen2.5-72B-Instruct-abliterated	48.11 %	85.93 %	60.49 %	60.12 %	19.35 %	12.34 %	50.41 %	76.77 kg
6	6	🗨️	Qwen/Qwen2.5-72B-Instruct	47.98 %	86.38 %	61.87 %	59.82 %	16.67 %	11.74 %	51.40 %	47.65 kg
7	7	🗨️	MaziyarPanahi/calme-2.1-qwen2.5-72b	47.86 %	86.62 %	61.66 %	59.14 %	15.10 %	13.30 %	51.32 %	29.50 kg
8	8	🔹	newsbang/Homer-v1.0-Qwen2.5-72B	47.46 %	76.28 %	62.27 %	49.02 %	22.15 %	17.90 %	57.17 %	29.55 kg
9	9	🗨️	ehristoforu/qwen2.5-test-32b-it	47.37 %	78.89 %	58.28 %	59.74 %	15.21 %	19.13 %	52.95 %	29.54 kg
10	10	🔹	Saxo/Linkbricks-Horizon-AI-Avengers-V1-32B	47.34 %	79.72 %	57.63 %	60.27 %	14.99 %	18.16 %	53.25 %	7.95 kg

[https://huggingface.co/spaces/open-llm-leaderboard/open\\_llm\\_leaderboard#/](https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard#/)

# LLM Training

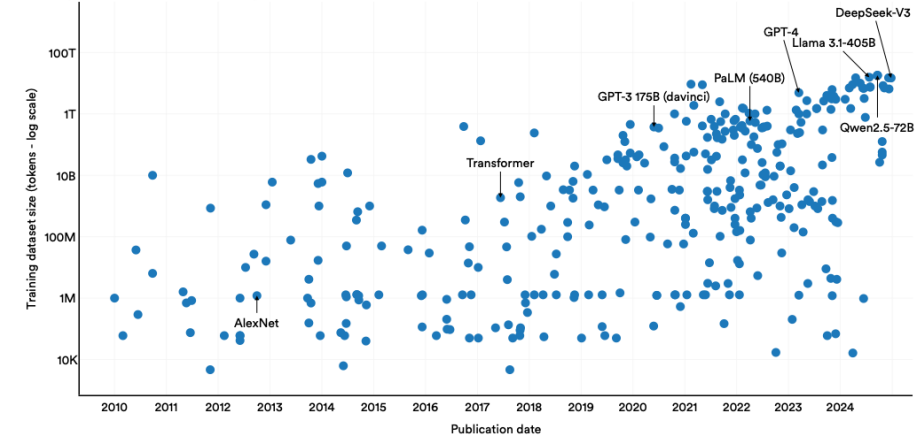
## Training compute of notable models

EPOCH AI



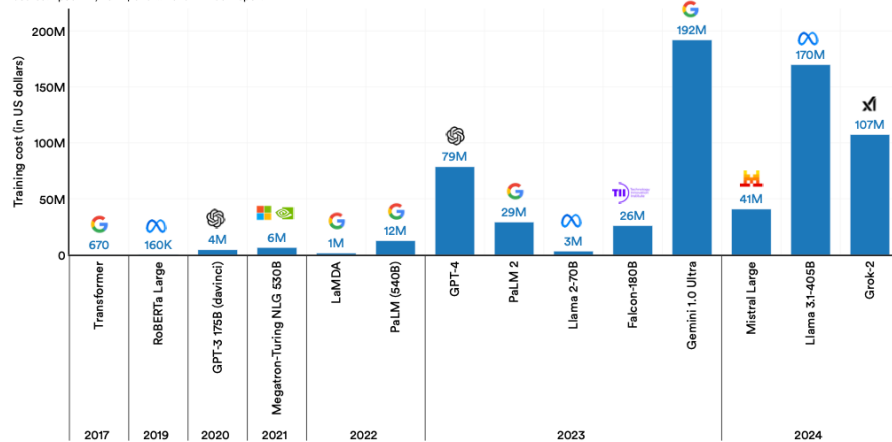
## Training dataset size of notable AI models, 2010–24

Source: Epoch AI, 2025 | Chart: 2025 AI Index report



## Estimated training cost of select AI models, 2019–24

Source: Epoch AI, 2024 | Chart: 2025 AI Index report



## Number of parameters of select notable AI models by sector, 2012–24

Source: Epoch AI, 2025 | Chart: 2025 AI Index report

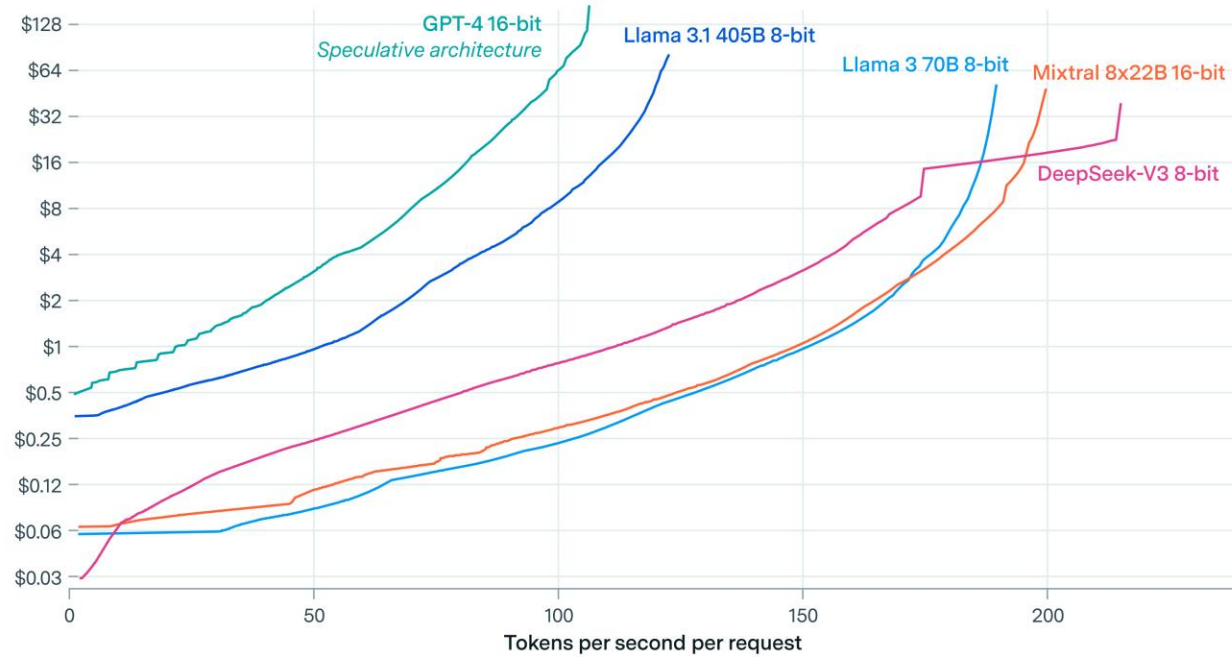


# LLM Inference

## Token economics of various models

EPOCH AI

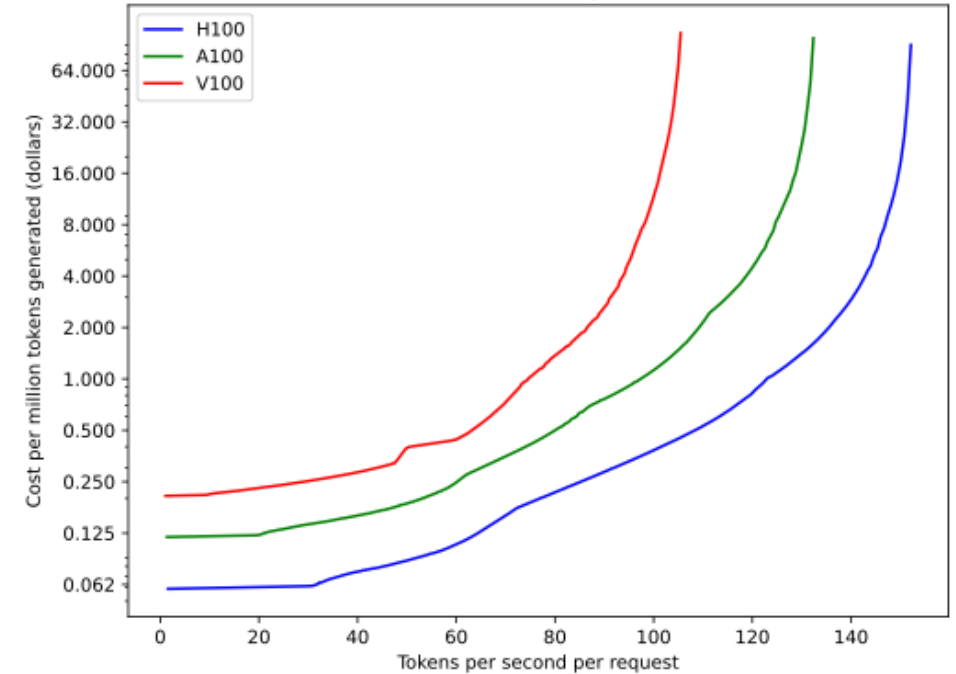
Cost per million tokens generated (USD)



CC-BY

epoch.ai

Token economics of Llama 3 70B (8-bit quantization) on different GPUs



<https://epoch.ai/blog/inference-economics-of-language-models>

Ege Erdil. 'Inference economics of language models'. *ArXiv [cs.LG]*, 2025. arXiv. <https://arxiv.org/abs/2506.04645>.

# Tasks

- Use HF transformers library and create a tokenizer (print token ids for a given text) and next token predictor (print probabilities for top 5 tokens)
- Implement a ChatBot on Google Colab using HF and/or LangChain framework. Bonus: Try using both cloud and local models.
- Create Google Gemini API key
- Use Gemini in your ChatBot.
- Create at least 5 free-form and 5 multiple-choice questions related to HPC concepts you learned today.
- Evaluate your ChatBot based on these questions.