From Finetuning to Applications

Hamid Bekamiri

Aalborg University Business School
Innovation, Knowledge, and Economic Development (IKE)
AI Denmark (AIDK)
hamidb@business.aau.dk
Feb 02, 2026

Today's lecture overview

1. Recap of Lecture 1
2. Fine-tuning: API vs. Native PyTorch
3. The History of GPT Models
4. Optimizing GPTs
5. The LLM Ecosystem
6. Quantization
7. Applications of LLMs

About AI-LAB
How to access
- Who can get access?
- 1. Fill out the application form
- 2. Processing of your request
- 3. Follow our guides

System overview
Troubleshooting
Terms and Conditions
Fair Usage

GUIDES
Index
Video Onboarding Guide
Getting started
Additional Guides
Workshop

How to access

Who can get access?

Access is available to all students and teachers at Aalborg University.

1. Fill out the application form

Fill out this application form.

2. Processing of your request

You will recieve an email when your request is approved. Expect a delay up to 30 minutes before you can log in to AI-LAB.

3. Follow our guides

After getting access, please follow our guides

List of Group

Fill info in this Excel file:

https://docs.google.com/spreadsheets/d/1TIyo2RfctdyVmMipJZAxZ0tl53pFC4PmjejzAnk97DY/edit?usp=sharing

Recap: Lecture 1

Let’s refresh what we learned in the previous lecture!

BERT and SBERT

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

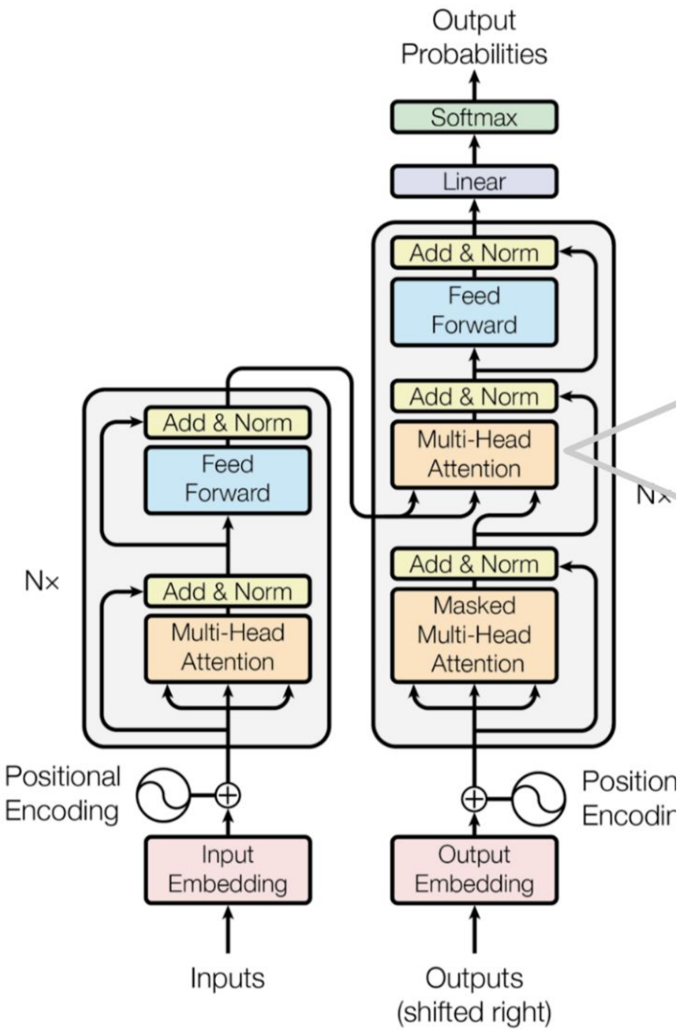
Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

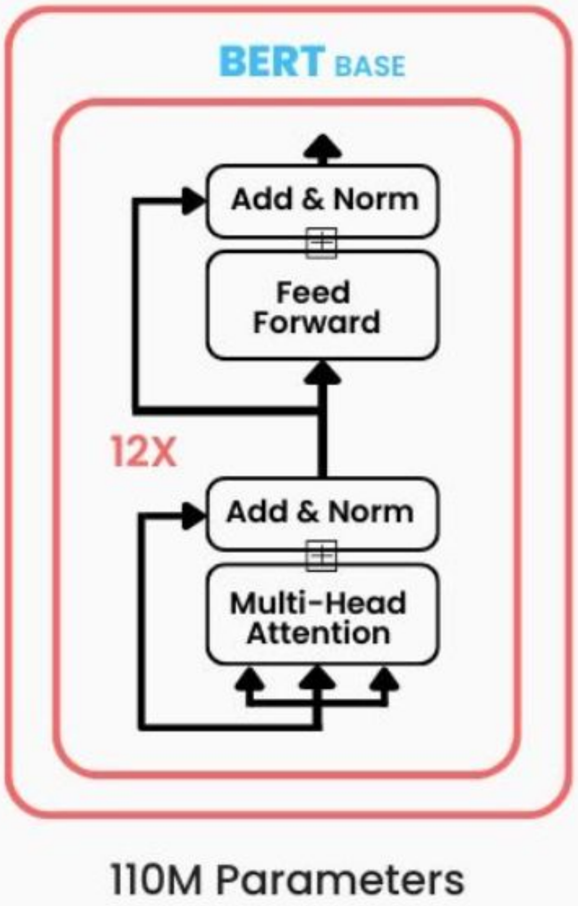
Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.





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

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

1 Introduction

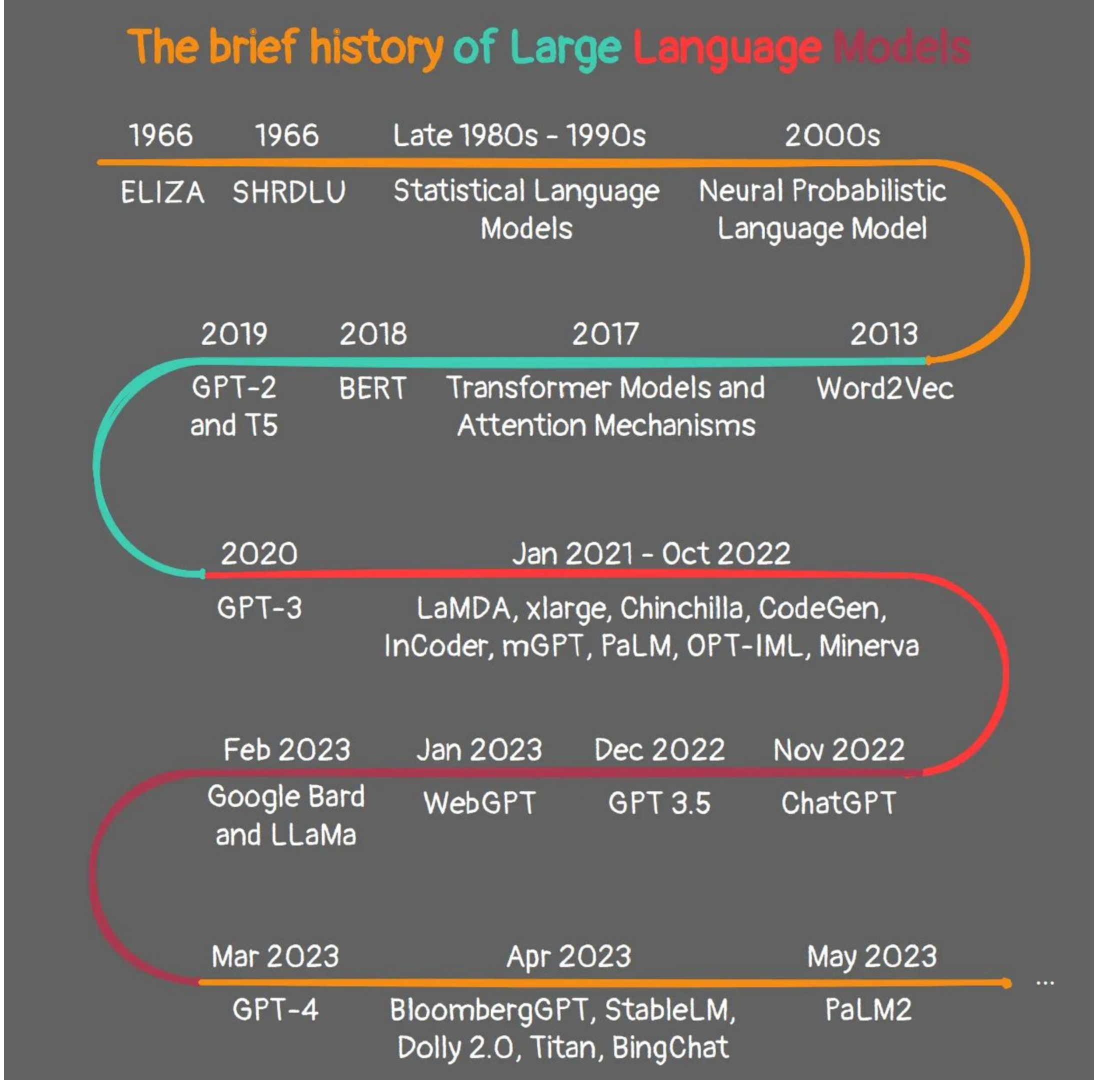
Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.

LLMs in Harry Potter: Tom Riddle introduces himself to Harry Potter


The brief history of Large Language Models
1966
ELIZA
1966
SHRDLU
Late 1980s - 1990s
Statistical Language Models
2000s
Neural Probabilistic Language Model
2013
Word2Vec
2017
Transformer Models and Attention Mechanisms
2018
BERT
2019
GPT-2 and T5
2020
GPT-3
Jan 2021 - Oct 2022
LaMDA, xlarge, Chinchilla, CodeGen, InCoder, mGPT, PaLM, OPT-IML, Minerva
Nov 2022
ChatGPT
Dec 2022
GPT 3.5
Jan 2023
WebGPT
Feb 2023
Google Bard and LLaMa
Mar 2023
GPT-4
Apr 2023
BloombergGPT, StableLM, Dolly 2.0, Titan, BingChat
May 2023
PaLM2
...

Optimizing LLMs

These techniques are essential for efficiently adapting large, pre-trained models like GPT or BERT to specialized tasks or domains, optimizing resource usage and reducing training time.

1. **Prompt Engineering (In-Context Learning)**:
   - **Definition**: Crafting input prompts to guide a Large Language Model (LLM) for desired outputs.
   - **Application**: Uses natural language prompts to "program" the LLM, leveraging its contextual understanding.
   - **Model Change**: No alteration to the model's parameters; relies on the model's existing knowledge and interpretive abilities.
2. **Prompt Tuning**:
   - **Difference from Prompt Engineering**: Involves appending a trainable tensor (prompt tokens) to the LLM's input embeddings.
   - **Process**: Fine-tunes this tensor for a specific task and dataset, keeping other model parameters unchanged.
   - **Example**: Adapting a general LLM for specific tasks like sentiment classification by adjusting prompt tokens.
3. **Parameter-Efficient Fine-Tuning (PEFT)**:
   - **Overview**: A set of techniques to enhance model performance on specific tasks or datasets by tuning a small subset of parameters.
   - **Objective**: Targeted improvements without the need for full model retraining.
   - **Relation to Prompt Tuning**: Prompt tuning is a subset of PEFT, focusing on fine-tuning specific parts of the model for task/domain adaptation.

Prompt Engineering: In context Learning

Google Research March 9, 2023

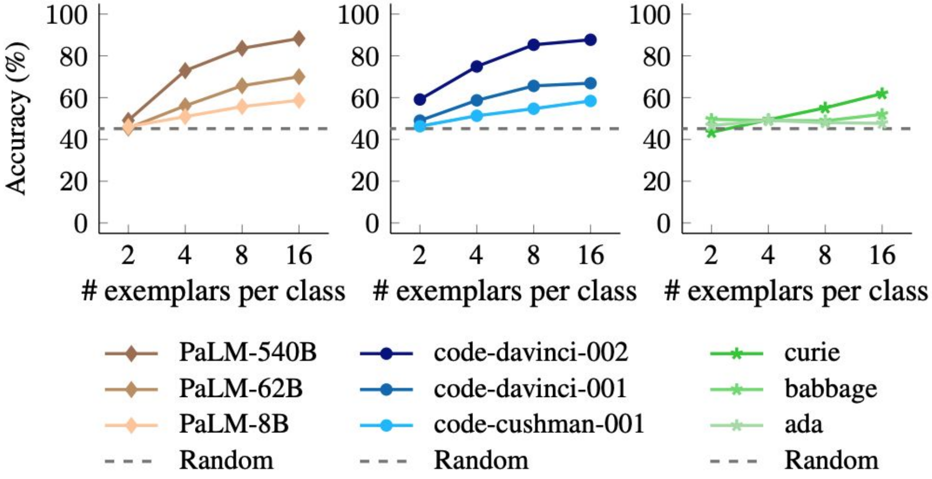
LARGER LANGUAGE MODELS DO IN-CONTEXT LEARNING DIFFERENTLY

Jerry Wei[1,2,*] Jason Wei[1] Yi Tay[1] Dustin Tran[1] Albert Webson[1,3,*]
Yifeng Lu[1] Xinyun Chen[1] Hanxiao Liu[1] Da Huang[1] Denny Zhou[1]
Tengyu Ma[1,2,†]

[1] Google Research, Brain Team [2] Stanford University [3] Brown University

ABSTRACT

We study how in-context learning (ICL) in language models is affected by semantic priors versus input–label mappings. We investigate two setups—ICL with flipped labels and ICL with semantically-unrelated labels—across various model families (GPT-3, InstructGPT, Codex, PaLM, and Flan-PaLM). First, experiments on ICL with flipped labels show that overriding semantic priors is an emergent ability of model scale. While small language models ignore flipped labels presented in-context and thus rely primarily on semantic priors from pretraining, large models can override semantic priors when presented with in-context exemplars that contradict priors, despite the stronger semantic priors that larger models may hold. We next study *semantically-unrelated label ICL* (SUL-ICL), in which labels are semantically unrelated to their inputs (e.g., foo/bar instead of negative/positive), thereby forcing language models to learn the input–label mappings shown in in-context exemplars in order to perform the task. The ability to do SUL-ICL also emerges primarily with scale, and large-enough language models can even perform linear classification in a SUL-ICL setting. Finally, we evaluate instruction-tuned models and find that instruction tuning strengthens both the use of semantic priors and the capacity to learn input–label mappings, but more of the former.



Prompt Engineering Techniques

1. **N-shot Prompting**
2. **Chain-of-Thought (CoT) Prompting**
3. **Generated Knowledge Prompting**
4. **Self-Consistency**
5. **Retrieval Augmented Generation (RAG)**
6. **ReAct Prompting**

Zero shot and few shot prompting method

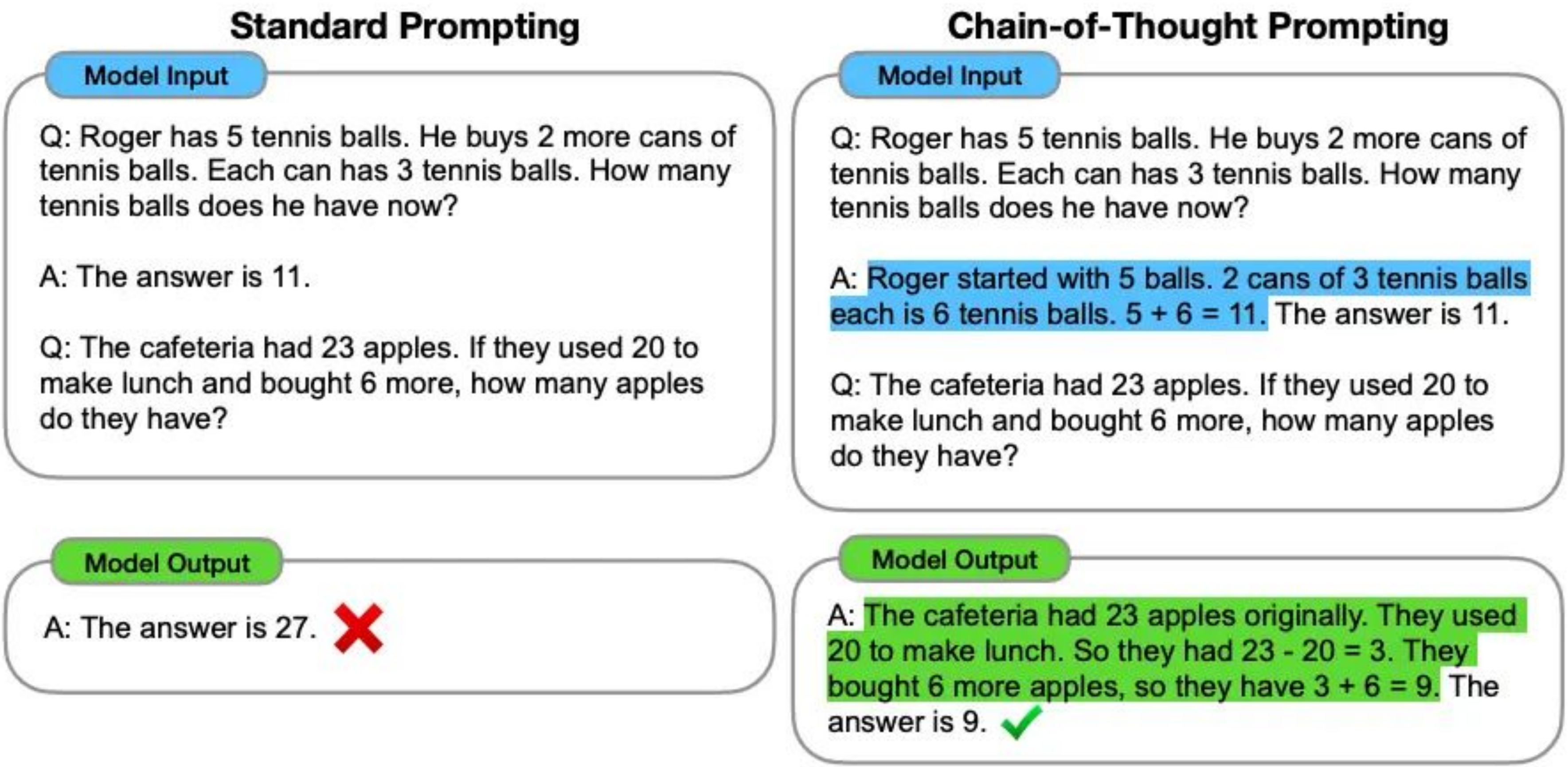
This slide explains the two types of N-shot prompting, named zero-shot and few-shot prompting. The purpose of this slide is to represent the prompt input and model's response for both types with the help of a few input examples.

Zero-shot prompting

Overview	o Refers to a scenario when predictions are produced without any explicit or supplemental examples o Excels at– • Classification tasks like Sentiment analysis or spam detection • Text transformation tasks like translation or summarization, and basic text synthesis • Add text here
Prompt input	"What is the sentiment of the following sentence: 'I had a tiring day at the work'?"
Model's response	"The sentiment of the sentence is negative."

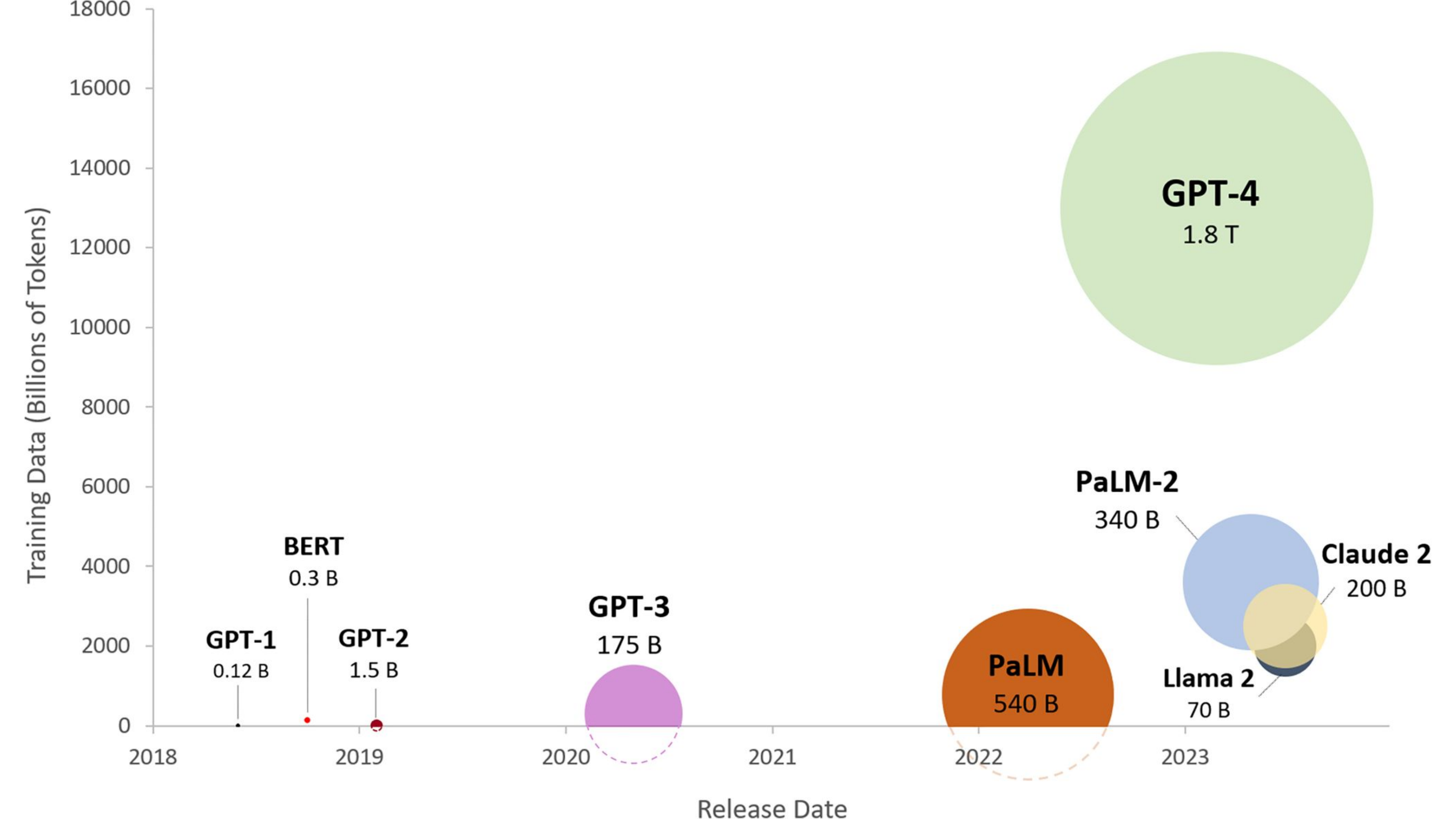
Few-shot prompting

Overview	o Uses a small number of instances, often between two and five, to direct the model's output o Intended to point the model in the direction of higher performance when dealing with more context-specific issues o Allow the model to – • More effectively tune its replies • Improve the precision of its predictions by providing a snapshot of the intended outcome
Prompt input	Write a rhymed couplet about moonlight. Example 1: 'Under moonlight's gentle hue, Stars twinkle as the night imbues.' Example 2: 'Moonlit night, serene and still, Whispers secrets, nature's thrill.' Now, write a rhymed couplet about sunshine.
Model's response	'Sunshine dances, bright and clear, Chasing shadows, spreading cheer.'


Standard Prompting
Model Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The answer is 27.
Chain-of-Thought Prompting
Model Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9.

LLMs Ecosystems

Tool	Category	Best For	Type
LangChain	Orchestration	Agents, tools, RAG, observability	Open-source
Flowise	App Builder / Orchestration (Visual)	Low-code drag-and-drop LLM apps (chatbots, RAG flows), rapid prototyping	Open-source
CrewAI	Agent Orchestration (Multi-agent)	Role-based multi-agent workflows, task delegation, coordinated tool-using agents	Open-source
Hugging Face	Model Hub	Open models, fine-tuning, hosting	Platform
vLLM / SGLang	Serving	High-throughput / Structured generation	Open-source
Ollama / llama.cpp	Local Run	Local inference & model management	Open-source
bitsandbytes	Quantization (4/8-bit)	Fit models into less VRAM; decent speed/quality tradeoffs	Open-source
Pydantic	Validation / Schemas	Type-safe data validation; enforce structured outputs and tool I/O	Open-source
LlamaIndex	Data / RAG	Ingestion, indexing, retrieval	Open-source
Haystack	RAG Pipelines	Production pipelines, Doc QA	Open-source
Semantic Kernel	Orchestration	Enterprise workflows (C#/Python)	Open-source



18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Training Data (Billions of Tokens)
2018
2019
2020
2021
2022
2023
Release Date
GPT-1
0.12 B
BERT
0.3 B
GPT-2
1.5 B
GPT-3
175 B
PaLM
540 B
PaLM-2
340 B
Claude 2
200 B
Llama 2
70 B
GPT-4
1.8 T

Storage capacity for each model?

0	0	0	0	0	0	0	0	Lowest value = 0
1	1	1	1	1	1	1	1	Highest value = 255

256 possible combinations of 8 bits

$$2\text{x}2\text{x}2\text{x}2\text{x}2\text{x}2\text{x}2\text{x}2 = 2^8 = 256$$

Let's explore Llama 2:

$$\textbf{64-bits} = \frac{64}{8} \times 70\text{B} \approx \textbf{560}\ \text{GB}$$

$$\textbf{32-bits} = \frac{32}{8} \times 70\text{B} \approx \textbf{280}\ \text{GB}$$

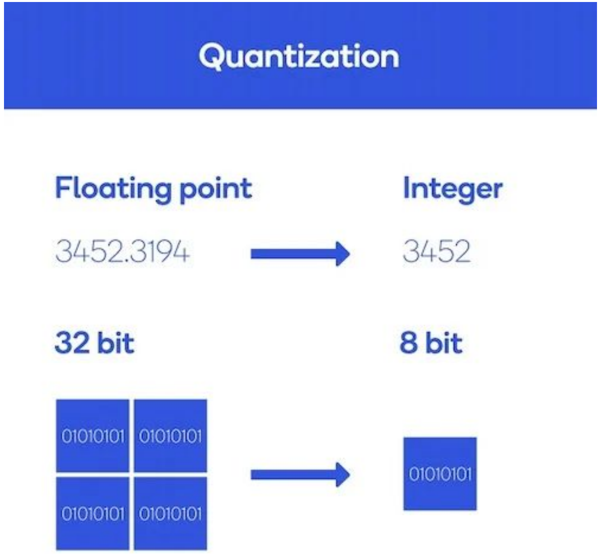
$$\textbf{16-bits} = \frac{16}{8} \times 70\text{B} \approx \textbf{140}\ \text{GB}$$

Binary vs. Powers of 2

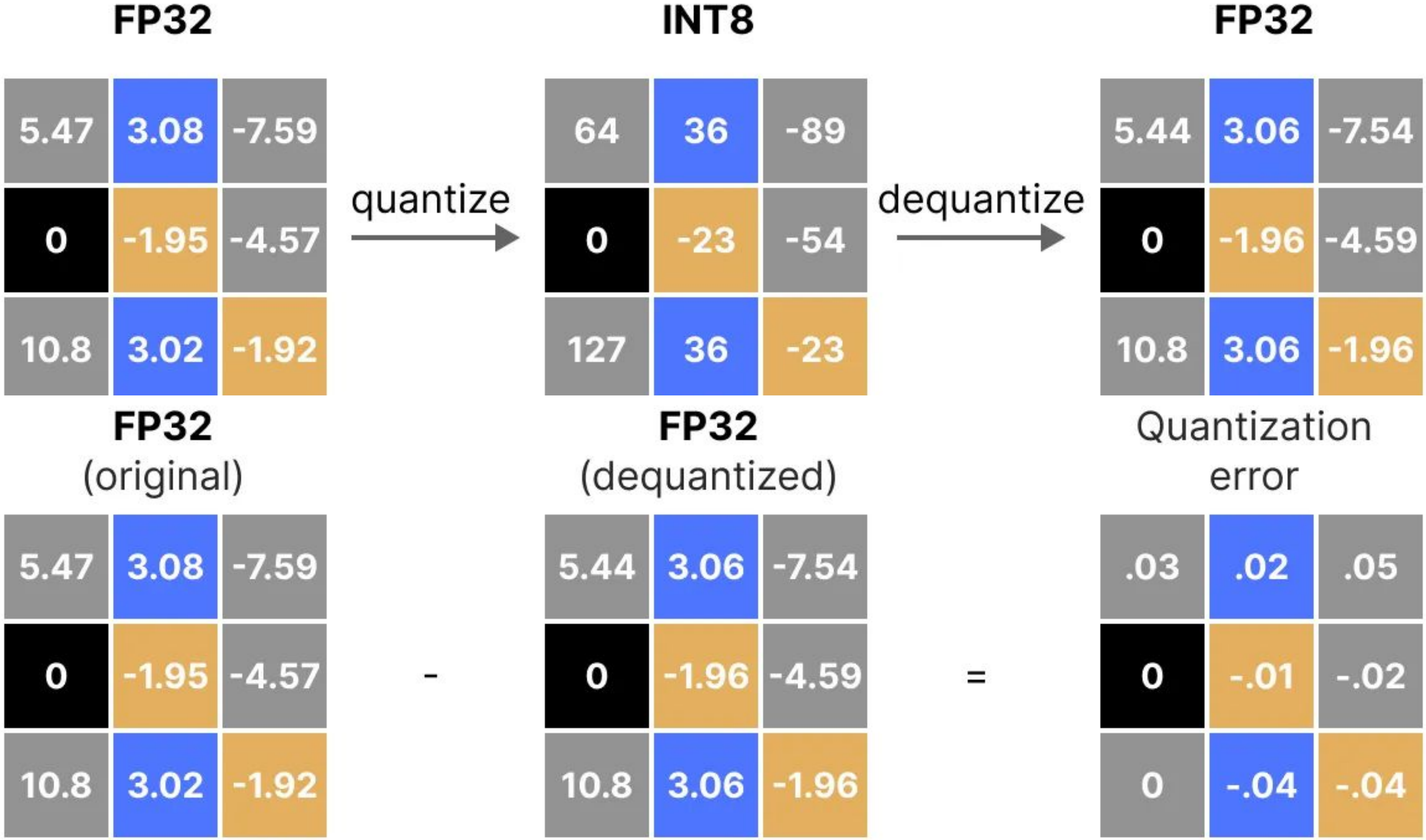
10000000	2^0 = 1
01000000	2^1 = 2
00100000	2^2 = 2x2 = 4
00010000	2^3 = 2x2x2 = 8
00001000	2^4 = 2x2x2x2 = 16
00000100	2^5 = 2x2x2x2x2 = 32
00000010	2^6 = 2x2x2x2x2x2 = 64
00000001	2^7 = 2x2x2x2x2x2x2 = 128
	2^8 = 2x2x2x2x2x2x2x2 = 256

FP32 (Single-Precision Floating-Point)

We can reduce the storage size of LLMs by a factor of four using quantization when moving from 32-bit to 8-bit floating point.



Original Image **“Quantized” Image**


FP32
5.47
3.08
-7.59
0
-1.95
-4.57
10.8
3.02
-1.92
quantize
INT8
64
36
-89
0
-23
-54
127
36
-23
dequantize
FP32
5.44
3.06
-7.54
0
-1.96
-4.59
10.8
3.06
-1.96
FP32
(original)
5.47
3.08
-7.59
0
-1.95
-4.57
10.8
3.02
-1.92
-
FP32
(dequantized)
5.44
3.06
-7.54
0
-1.96
-4.59
10.8
3.06
-1.96
=
Quantization
error
.03
.02
.05
0
-.01
-.02
0
-.04
-.04

Running an open-source LLM: 3 common approaches

Same model “weights” — different runtimes & ergonomics

A 🤗 Transformers

- Python loads the model inside your process
- You call model.generate() directly
- Best for: experiments, debugging, custom logic
- Serving/API: you build it yourself

B vLLM server

- Runs as a separate inference server
- OpenAI-compatible HTTP API (easy to plug in)
- Optimized for throughput: continuous batching + PagedAttention
- Best for: many calls, agents, multi-user serving

C Ollama app

- Runs a local model “app” + local API
- Often uses GGUF (quantized) model files
- Import models via Modelfile (FROM …gguf)
- Best for: simplest local setup

Rule of thumb: Transformers = maximum control • vLLM = fastest serving • Ollama = simplest local running

Do vLLM & Ollama use the local GPU?

Yes — they use the GPU of the machine where the model process is running

Where the model runs = where the GPU is used

Local computer

- vLLM/Ollama run on your machine
- They use your machine's GPU (if supported)
- Verify: nvidia-smi (NVIDIA) / activity monitor

Google Colab (default)

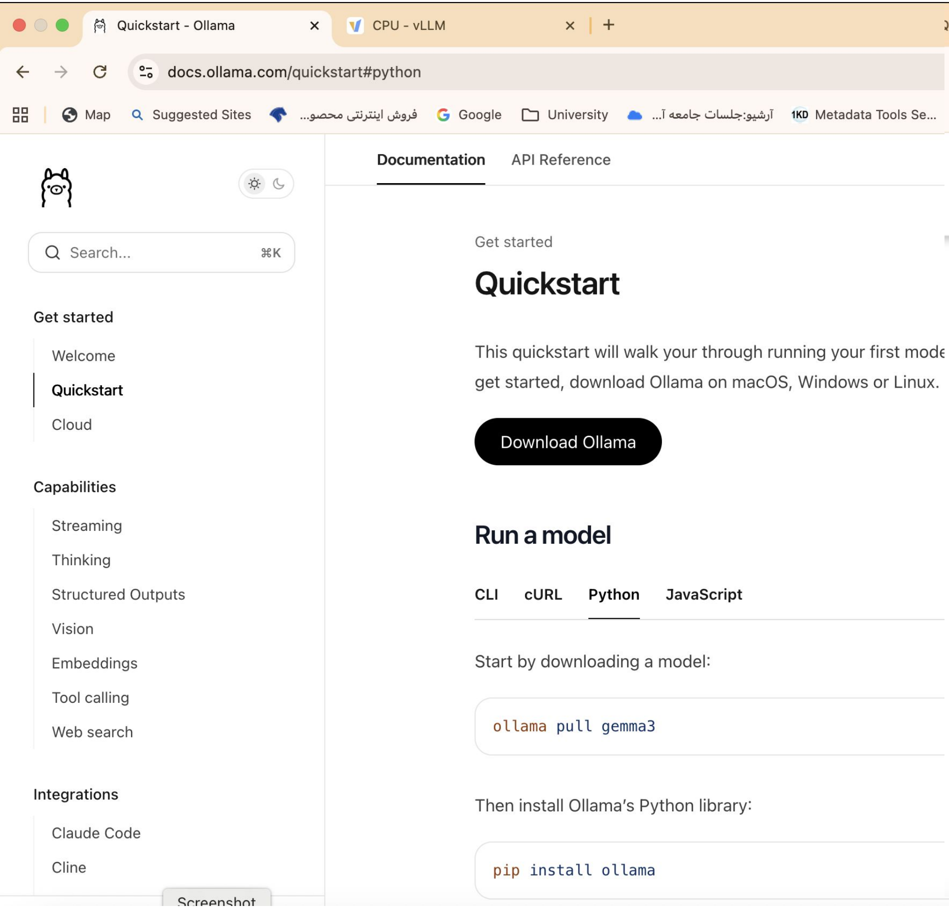
- Your code runs on a Google VM
- GPU is the VM's GPU (not your laptop GPU)
- Exception: "Local runtime" uses your hardware

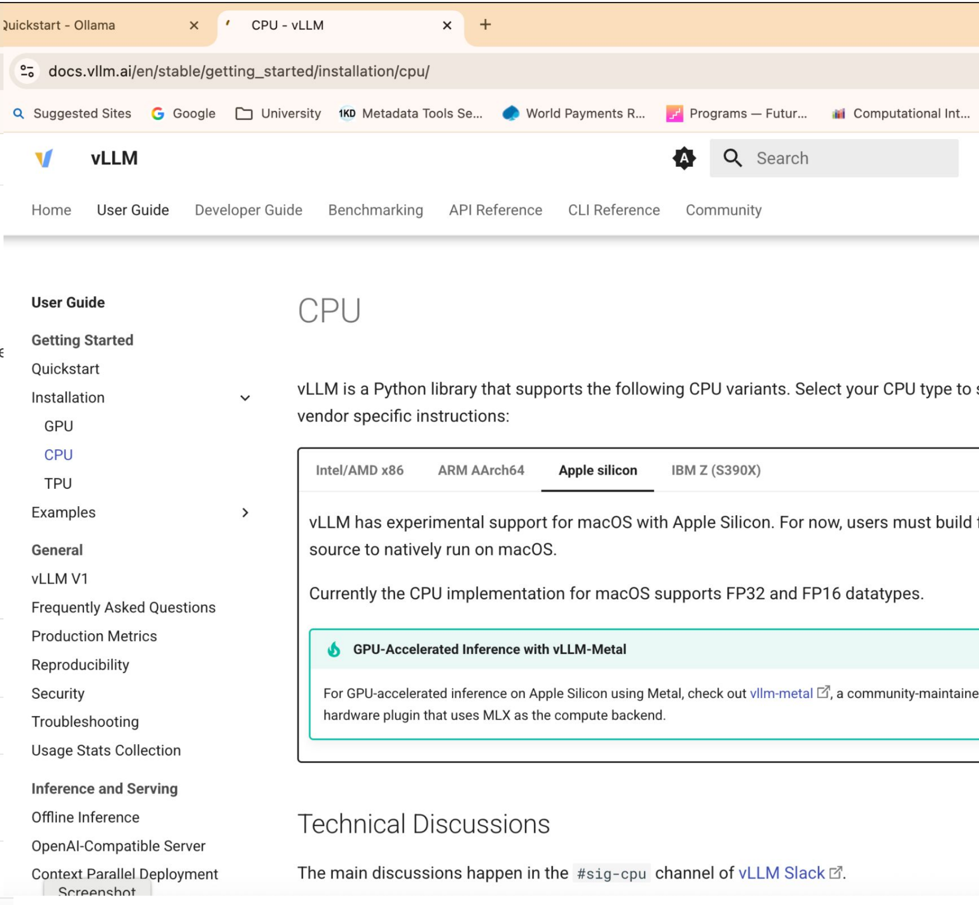
AAU / cluster

- Jobs run on allocated compute nodes
- GPU is the node's GPU
- Verify: nvidia-smi inside the job/session

Quick check: if the model is "local" but you're in Colab, "local" means the Colab VM unless you connect a Local runtime.

Do vLLM & Ollama use the local GPU?





AI -LAB

About AI-LAB
How to access
System overview
Troubleshooting
Terms and Conditions
Fair Usage

GUIDES
Index
Video Onboarding Guide
Getting started
1. Prerequisites
2. Login
Understanding AI-LAB Access
Basic SSH Connection
Setting Up SSH Shortcuts (Recommended)
Troubleshooting
3. File Handling
4. Running Jobs
5. Getting Containers

Logging into AI-LAB

This guide will help you connect to AI-LAB using SSH (Secure Shell). SSH is a secure way to access remote computers over a network.

Understanding AI-LAB Access

AI-LAB has two front-end nodes that act as entry points:

- **ailab-fe01.srv.aau.dk**
- **ailab-fe02.srv.aau.dk**

You can connect to either node - they provide the same functionality.

Basic SSH Connection

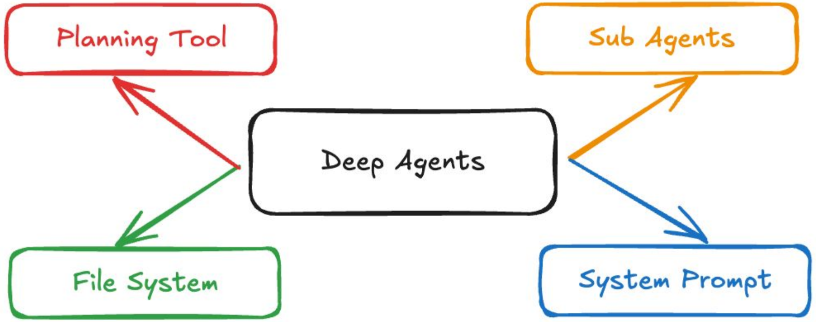
Step 1: Open Your Terminal

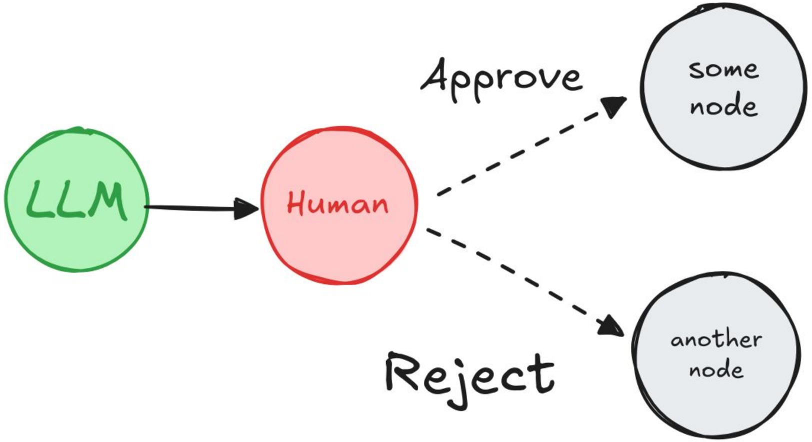
Windows macOS **Linux**

- Open your preferred terminal application

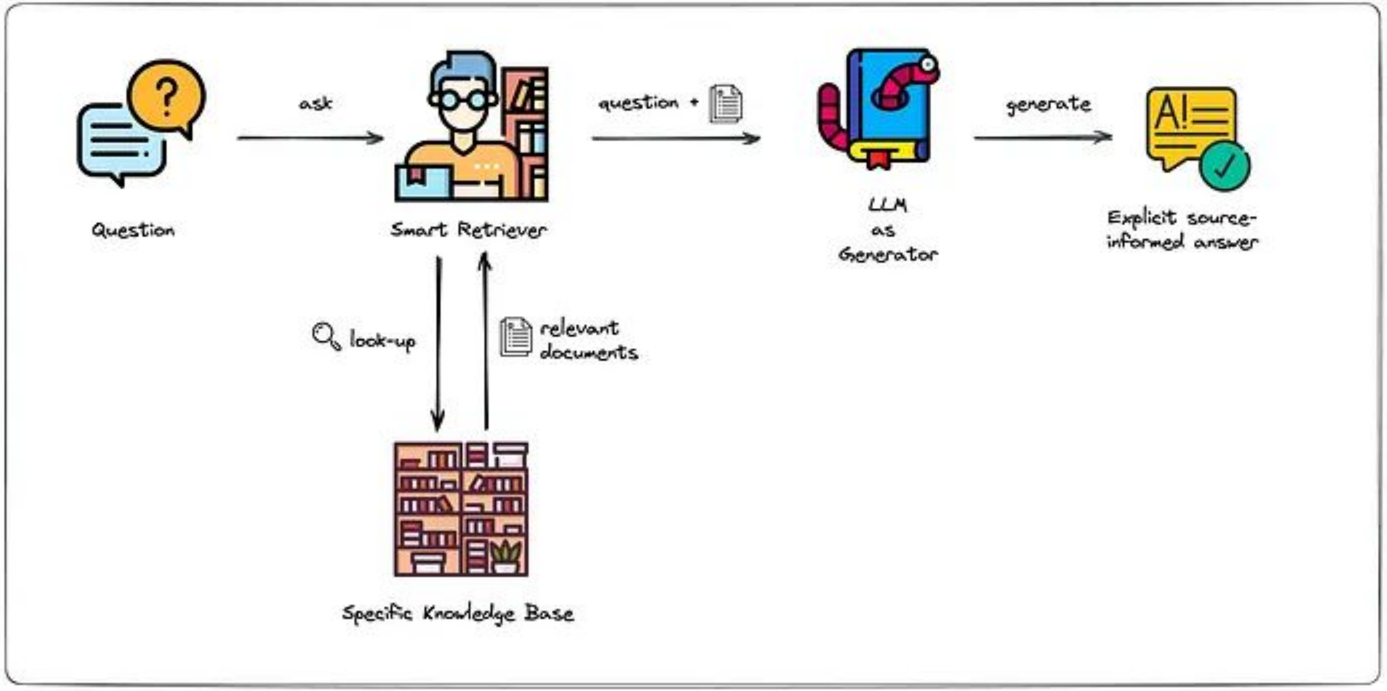
LangChain

- DeepAgent
- Human In The Loop

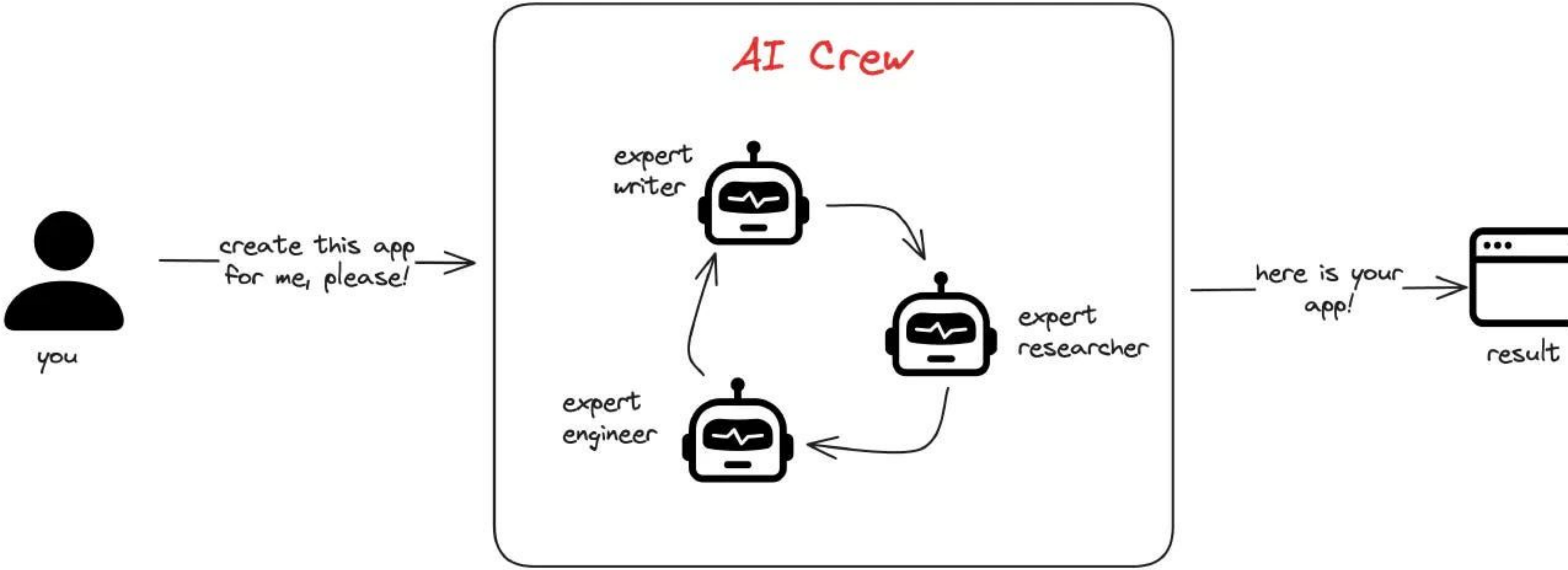




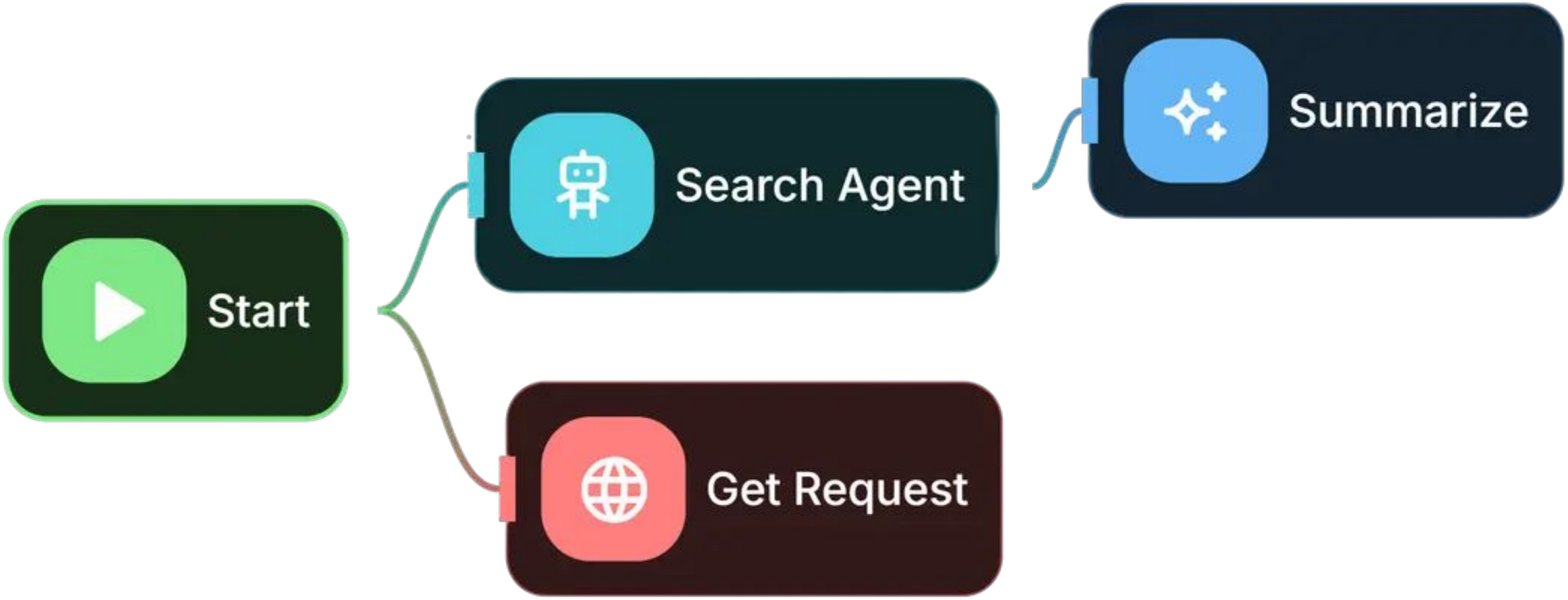
RAG + ChromaDB



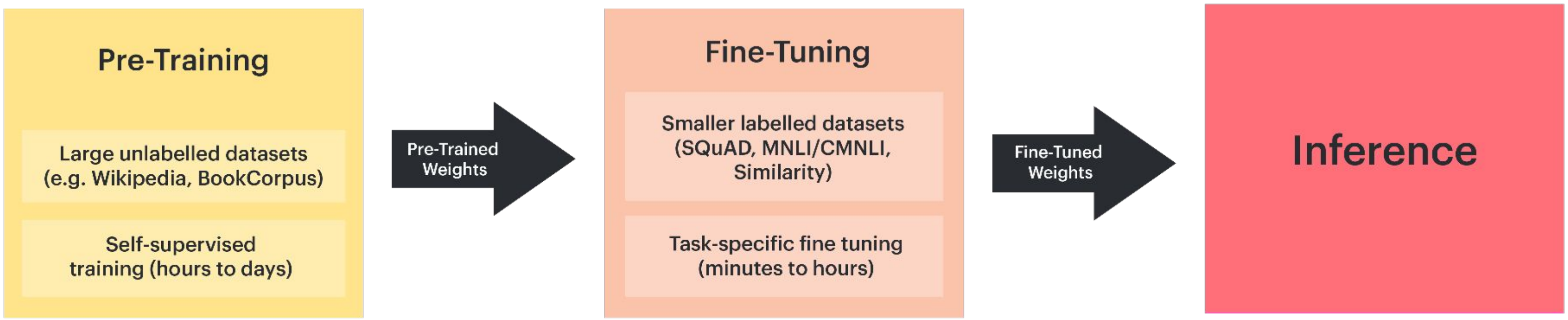
CrewAI


AI Crew
expert writer
expert researcher
expert engineer
create this app for me, please!
here is your app!
you
result

Flowise



Comparison	BERT October 11, 2018	RoBERTa July 26, 2019	DistilBERT October 2, 2019	ALBERT September 26, 2019
Parameters	**Base:** 110M **Large:** 340M	**Base:** 125 **Large:** 355	**Base:** 66	**Base:** 12M **Large:** 18M
Layers / Hidden Dimensions / Self-Attention Heads	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16	**Base:** 6 / 768 / 12	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16
Training Time	**Base:** 8 x V100 x 12d **Large:** 280 x V100 x 1d	1024 x V100 x 1 day (4-5x more than BERT)	**Base:** 8 x V100 x 3.5d (4 times less than BERT)	[not given] **Large:** 1.7x faster
Performance	Outperforming SOTA in Oct 2018	88.5 on GLUE	97% of BERT-base's performance on GLUE	89.4 on GLUE
Pre-Training Data	BooksCorpus + English Wikipedia = 16 GB	BERT + CCNews + OpenWebText + Stories = 160 GB	BooksCorpus + English Wikipedia = 16 GB	BooksCorpus + English Wikipedia = 16 GB
Method	Bidirectional Transformer, MLM & NSP	BERT without NSP, Using Dynamic Masking	BERT Distillation	BERT with reduced parameters & SOP (not NSP)


Pre-Training
Large unlabelled datasets (e.g. Wikipedia, BookCorpus)
Self-supervised training (hours to days)
Pre-Trained Weights
Fine-Tuning
Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
Task-specific fine tuning (minutes to hours)
Fine-Tuned Weights
Inference