KISTI 국가슈퍼컴퓨팅본부 AI 세미나

KiSTi
www.kisti.re.kr

Best Practices for Generative AI with LLMs on a Supercomputer

Soonwook Hwang

January 25, 2024

Korea Institute of Science and Technology Information

Credit: most contents from the online Generative AI with LLMs course offered by DeepLearning.AI

Agenda

- **Introduction to DeepLearning.AI's "Generarive AI with LLMs" Course**
- **A Brief Overview of Generative AI with LLMs**
  - Generative AI
  - Foundation Models (LLMs)
- **Generative AI Project Lifecycle**
- **Prompting and Prompt Engineering**
- **Instruction Fine-Tuning**
- **Parameter Efficient Fine-Tuning (PEFT)**
- **Evaluation**

DeepLearning AI's Generative AI with LLMs Course

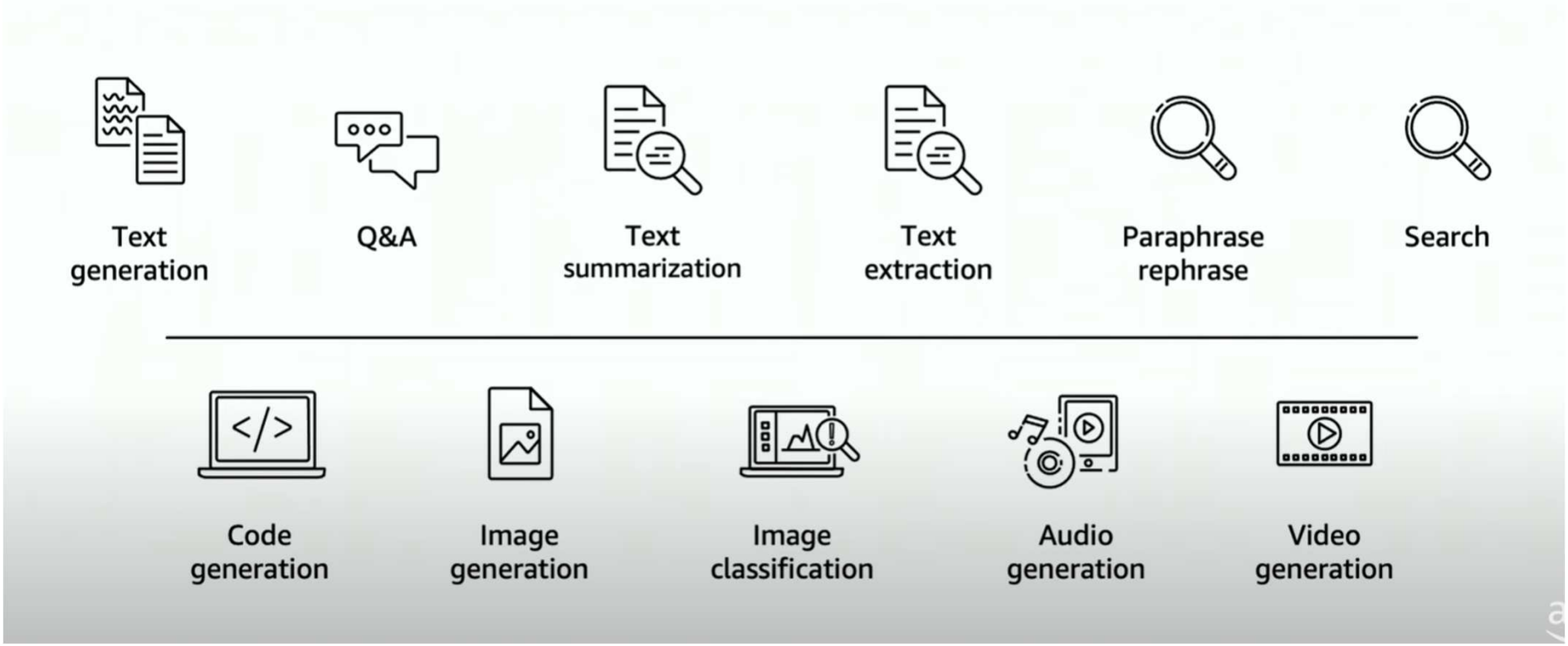
- **DeepLearning AI's Online Course Website**
  - https://www.coursera.org/learn/generative-ai-with-llms

Generative AI and LLMs

Generative AI with LLMs

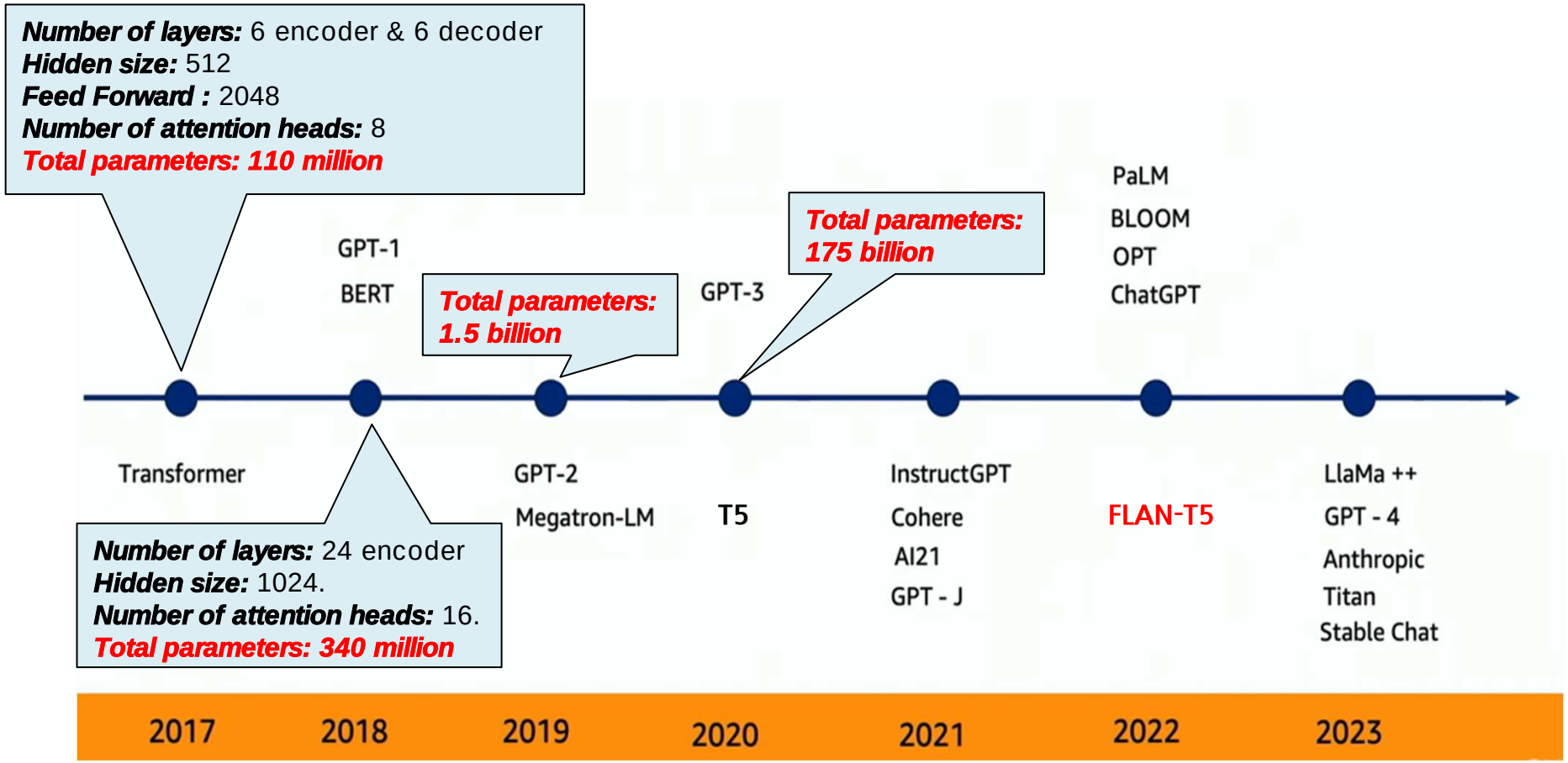
- The use of large language models like GPT-3 for generating human-like content, spanning text, images, code and so on
- Generative AI:
  - a subfield of AI focused on creating new content, data, or experiences
  - Ex) generating text, images, music, code, and even 3D models and so do
- Large Language Models (LLMs)
  - trained on a vast amount of data and code
  - capable of understanding and generating human-quality language
  - usually carefully prompt-engineered or fine-tuned to suit specific downstream tasks

Generative AI/LLMs Use Cases



Credit: [Generative AI Foundations on AWS: Part1]

Language Foundation Models



Credit: [Generative AI Foundations on AWS: Part1]

Timeline of Large Language Models (LLMs)

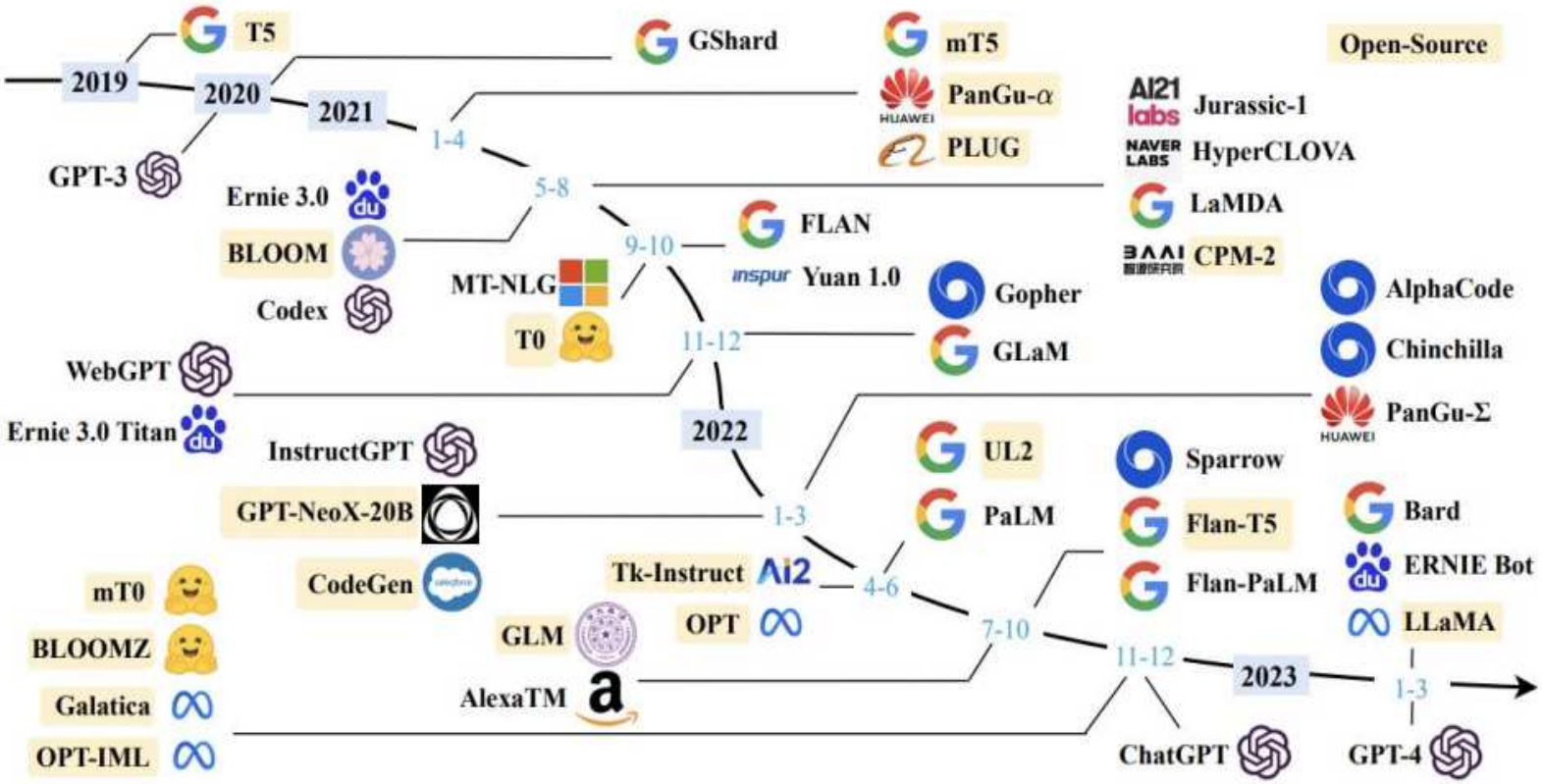
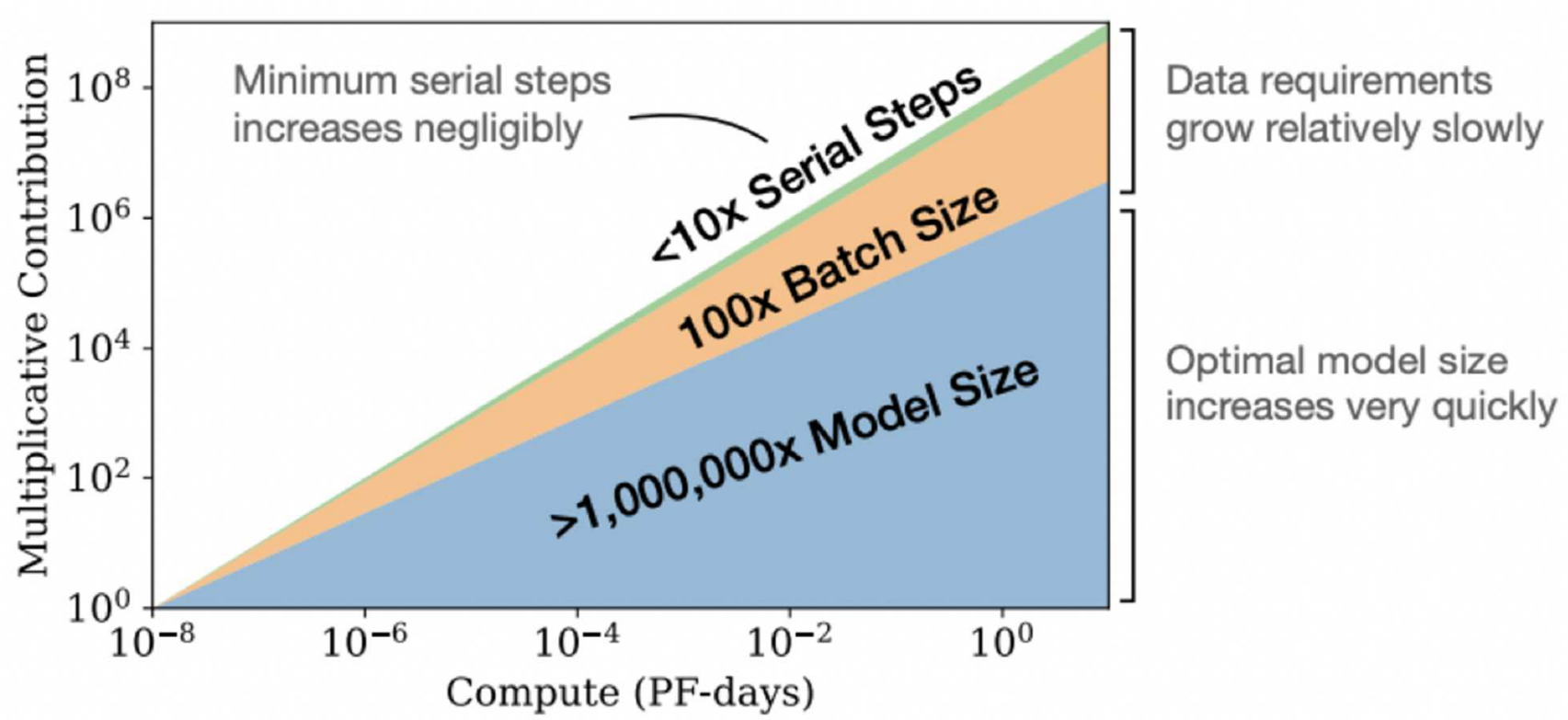


Fig. 1. A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open-source LLMs in yellow color.

A Survey of Large Language Model
https://arxiv.org/pdf/2303.18223.pdf

Scaling Laws

- **Larger models are significantly more sample-efficient**
  - ✓ optimally compute-efficient training involves training very large models on a relatively modest amount of data and stopping significantly before convergence.
- **Published by OpenAI (‘20.01)**



Scaling Laws for Neural Language Models
https://arxiv.org/pdf/2001.08361.pdf

LLM Considering Compute Budget Limitations

- **Chinchilla (70B) vs Gopher (280B)**
  - Published by DeepMind('22.04)
  - Chinchilla achieved superior performance over Gopher

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-

  - /w the same compute budget and utilizing a dataset that is more than four times larger than Gopher

Table 1 | **Current LLMs**. We show five of the current largest dense transformer models, their size, and the number of training tokens. Other than LaMDA (Thoppilan et al., 2022), most models are trained for approximately 300 billion tokens. We introduce *Chinchilla*, a substantially smaller model, trained for much longer than 300B tokens.

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

Training Compute- Optimal Large Language Model
https://arxiv.org/pdf/2203.15556.pdf

LLaMA (Large Language Model Meta AI)

- **Open source LLM released by Meta AI (‘23.02)**
- **smaller model parameters /w larger datasets**
  - Open AI’s GPT-3(175B), Google’s PaLM(540B)

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

LLaMA (Large Language Model Meta AI)

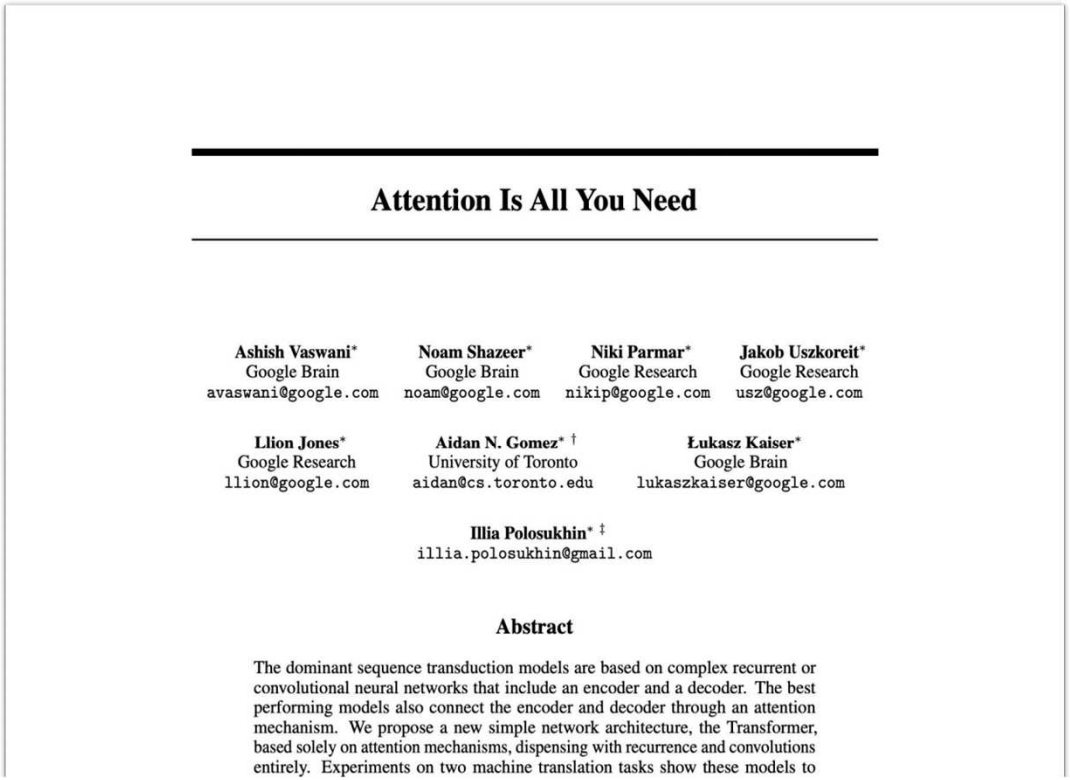
- **better performance in LLM benchmarks, compared to larger models**

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	**88.0**	82.3	-	83.4	**81.1**	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	**80.0**	**57.8**	58.6
	65B	85.3	**82.8**	**52.3**	**84.2**	77.0	78.9	56.0	**60.2**

Table 3: **Zero-shot performance on Common Sense Reasoning tasks.**

Transformers

- **Compared to the previous Seq-to-Seq models (e.g., RNN, LSTM)**
  - Scale efficiently
  - Parallel processed



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

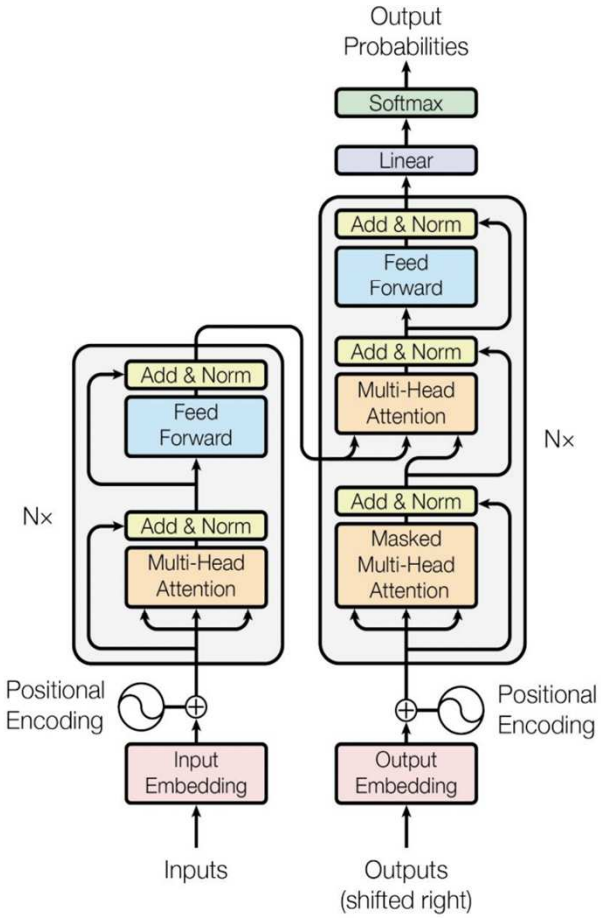
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

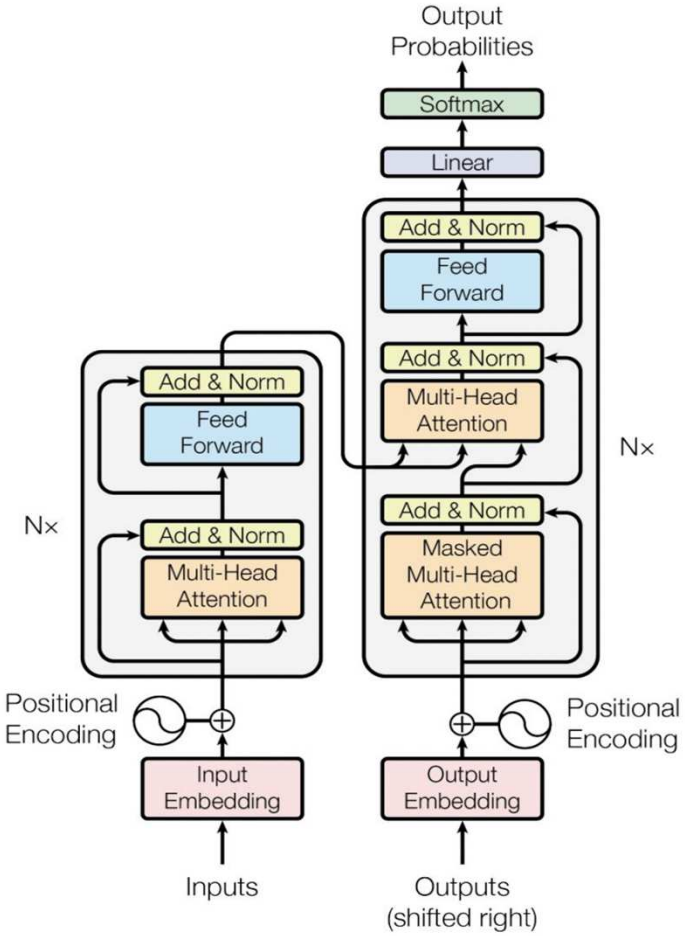
Abstract

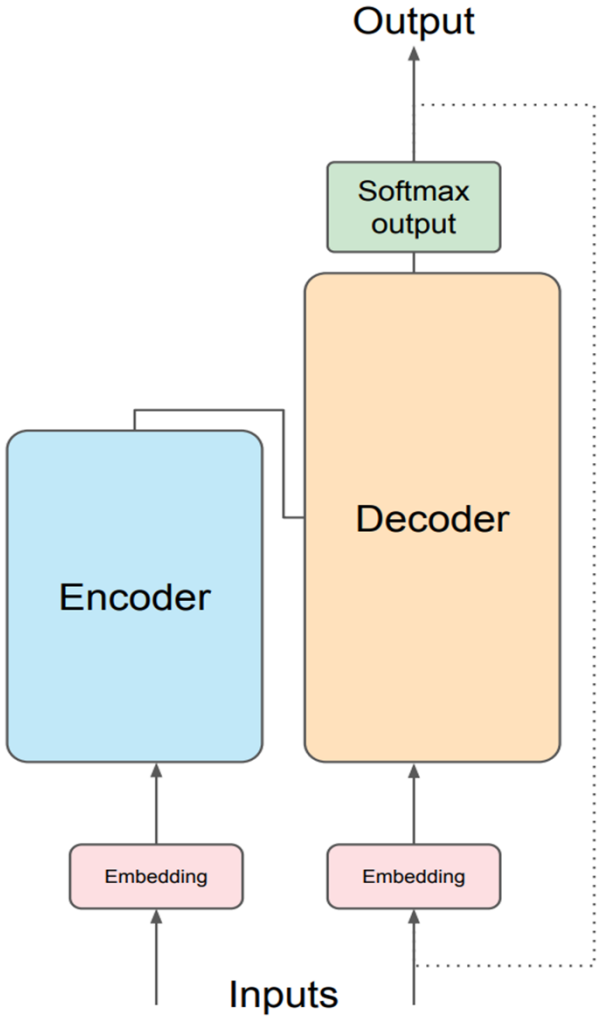
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to



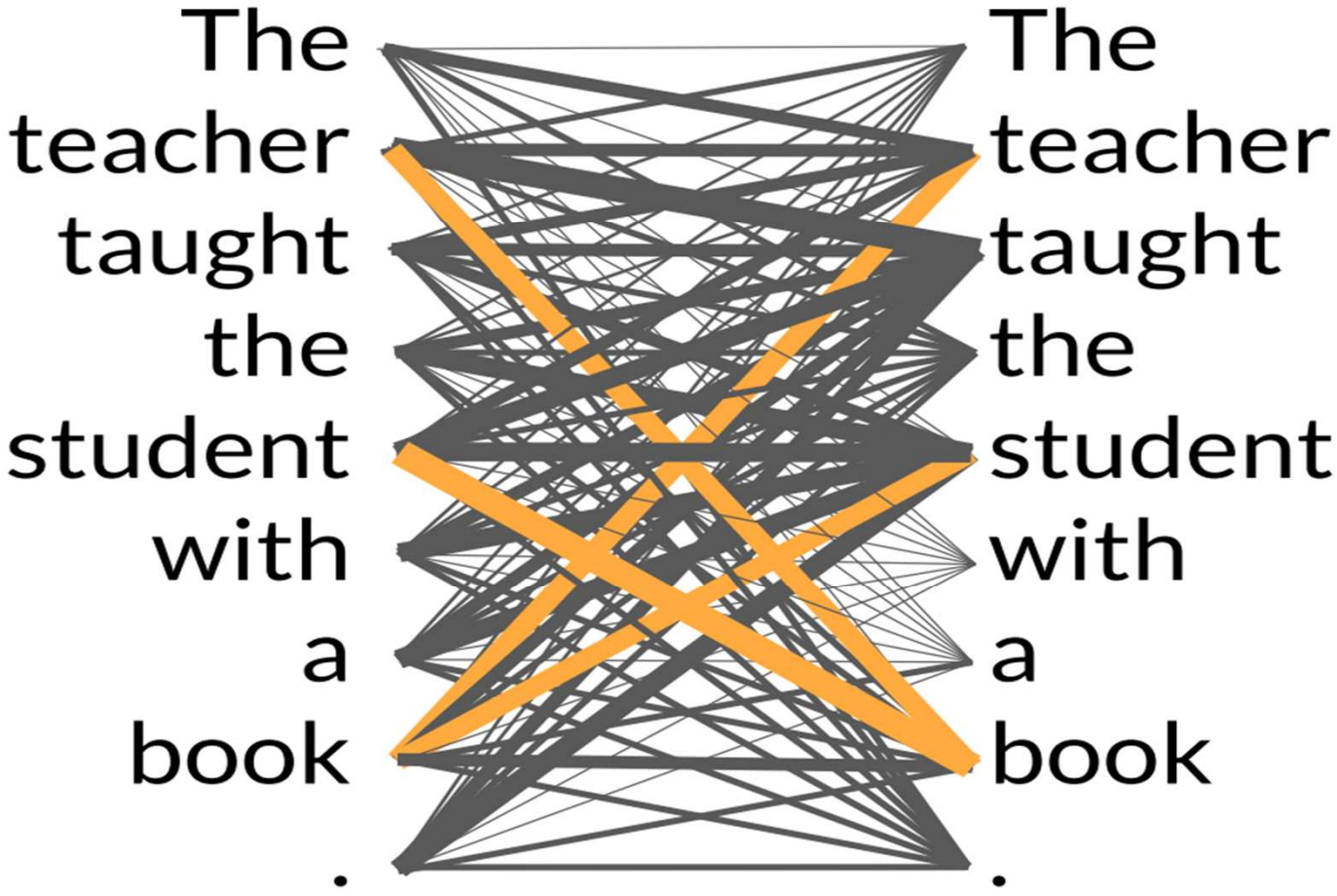
Transformers

- **Embedding + (Multi-head attention + Feed Forward) * N + (Multi-head attention + Feed Forward) * N + Linear**
- **Encoder-Decoder**

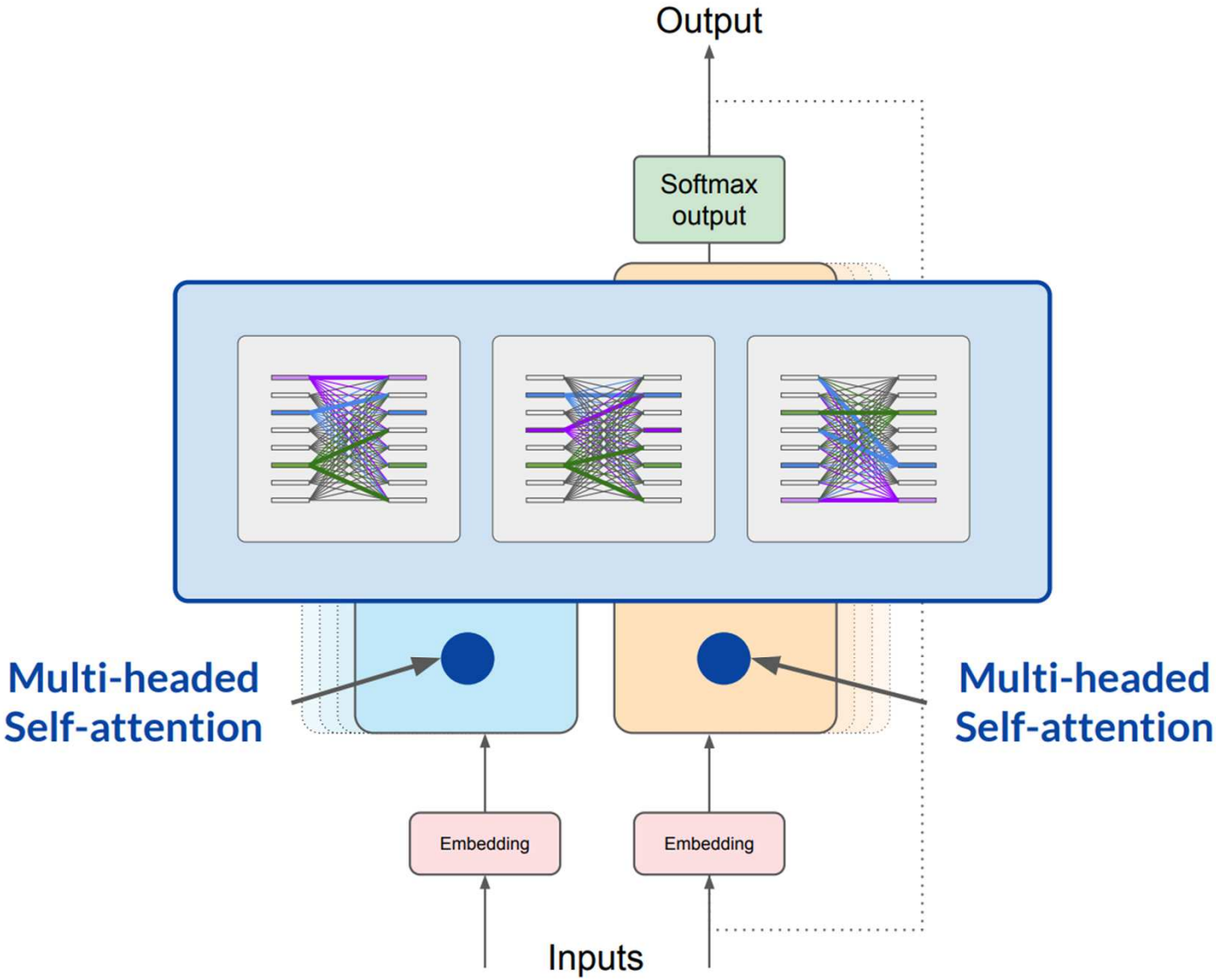




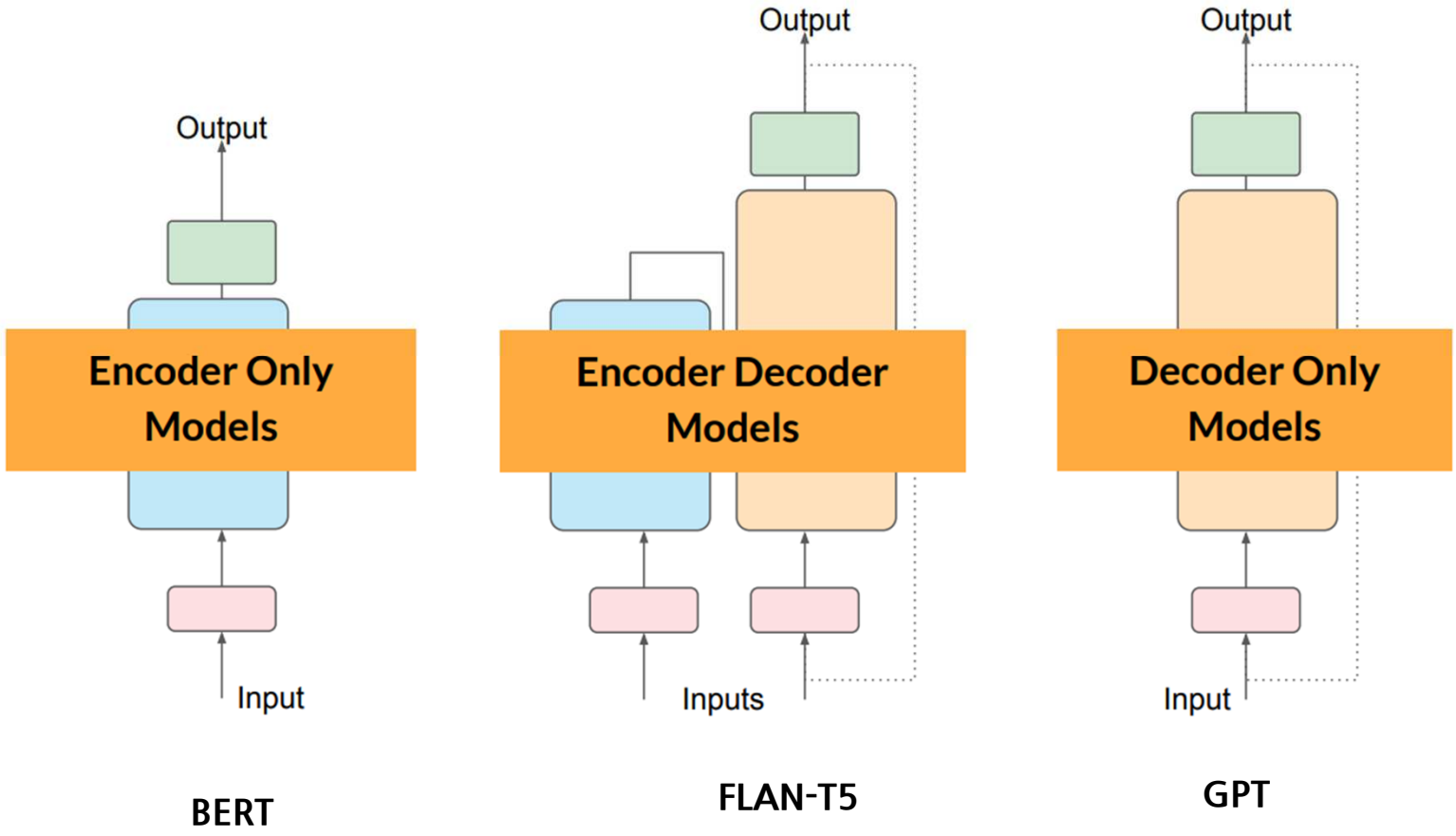
Self-attention



Transformers (Multi-headed attention)

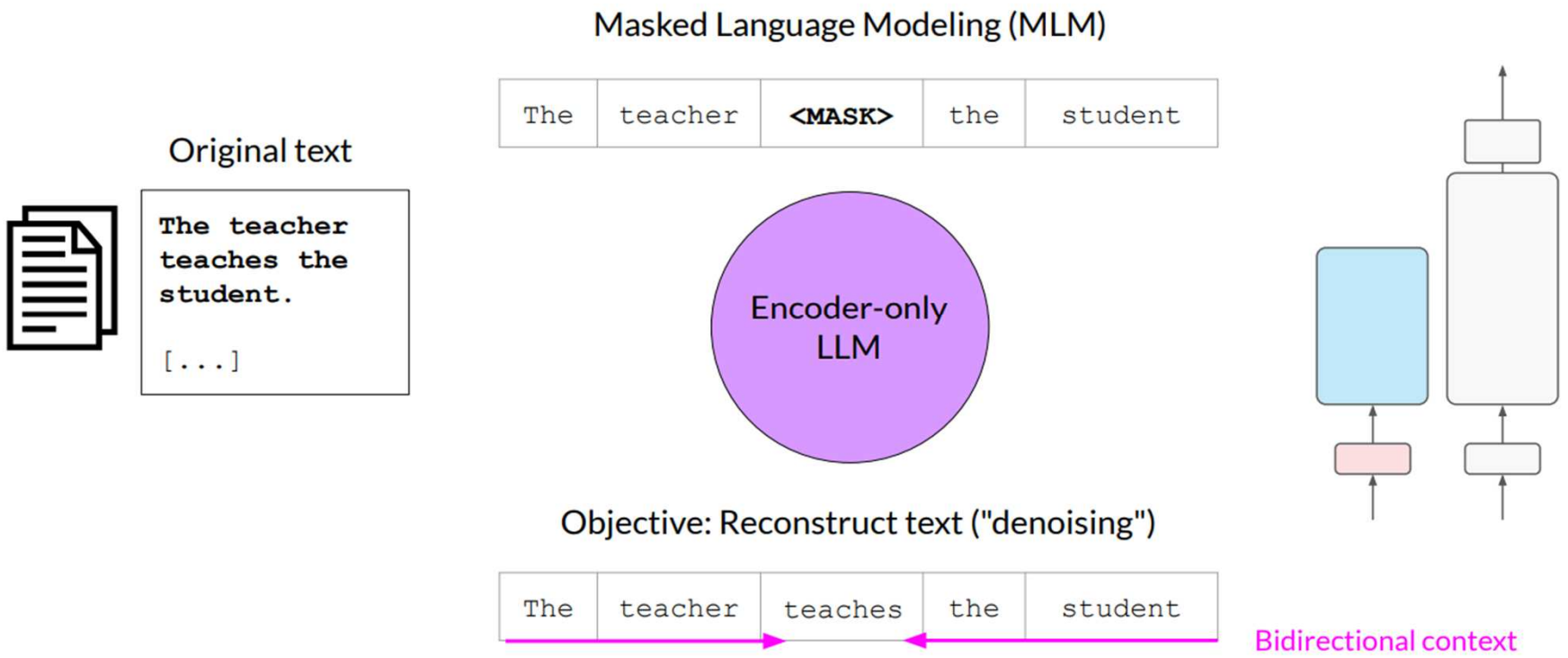


Transformer Models



Autoencoding Models

- **Encoder only model**
- **Masked Language Modeling (MLM)**
  - Objective: Text Reconstruction
- **Bidirectional**



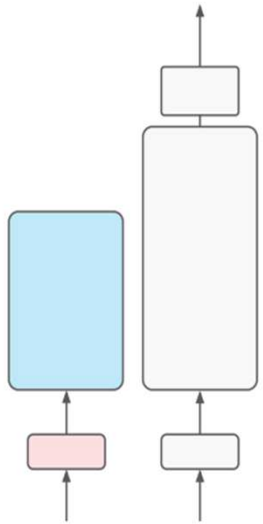
Autoencoding Models

Good use cases:

- Sentiment analysis
- Named entity recognition
- Word classification

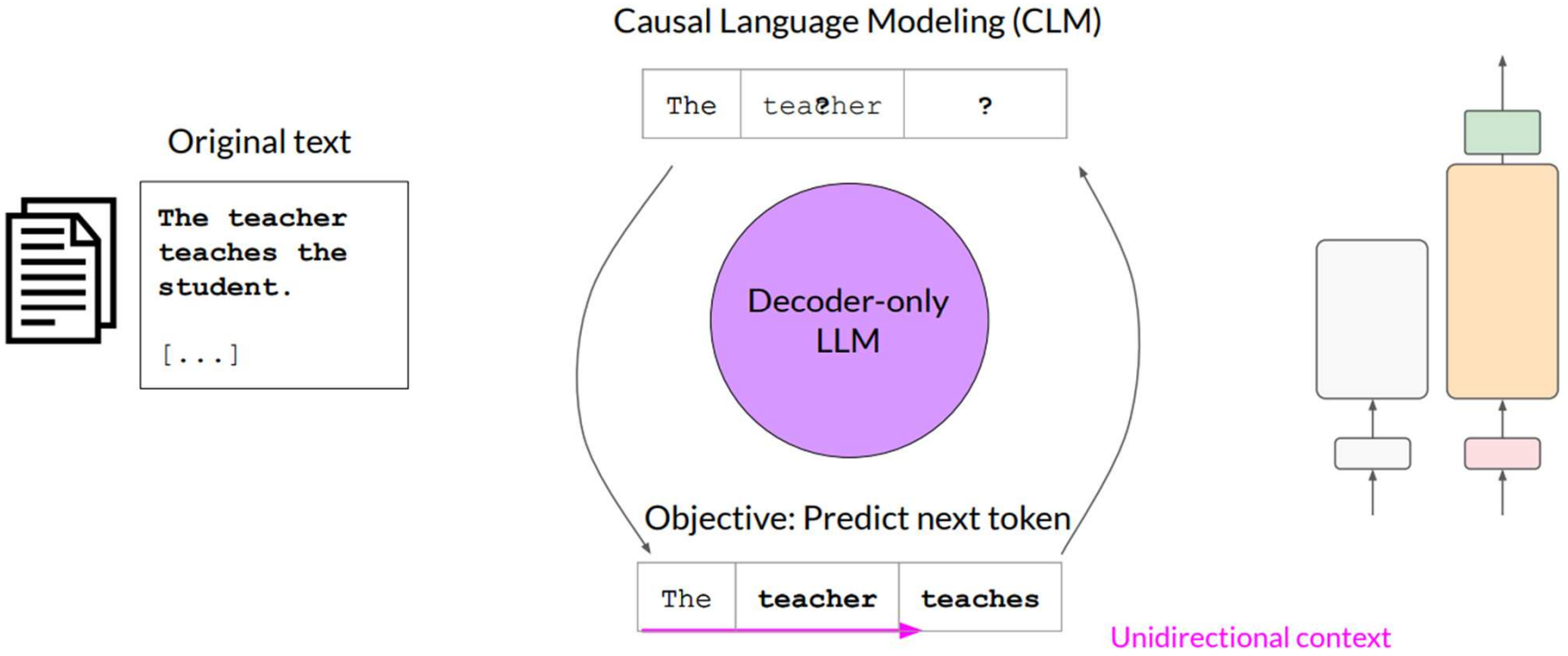
Example models:

- BERT
- ROBERTA



Autoregressive Models

- **Decoder only model**
- **Causal Language Modeling (CLM)**
  - Object: Next token prediction
- **Unidirectional**



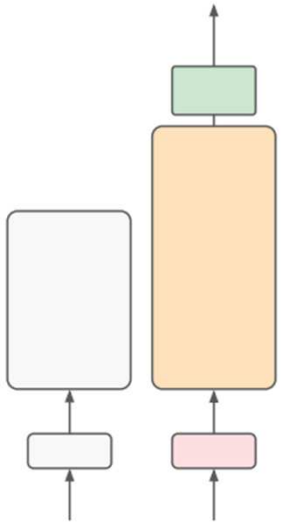
Autoregressive Models

Good use cases:

- Text generation
- Other emergent behavior
  - Depends on model size

Example models:

- GPT
- BLOOM



Sequence-to-Sequence Models

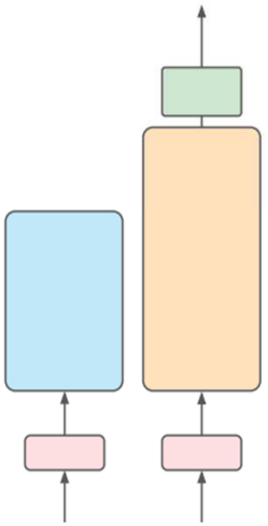
- **Encoder Decoder Models**

Good use cases:

- Translation
- Text summarization
- Question answering

Example models:

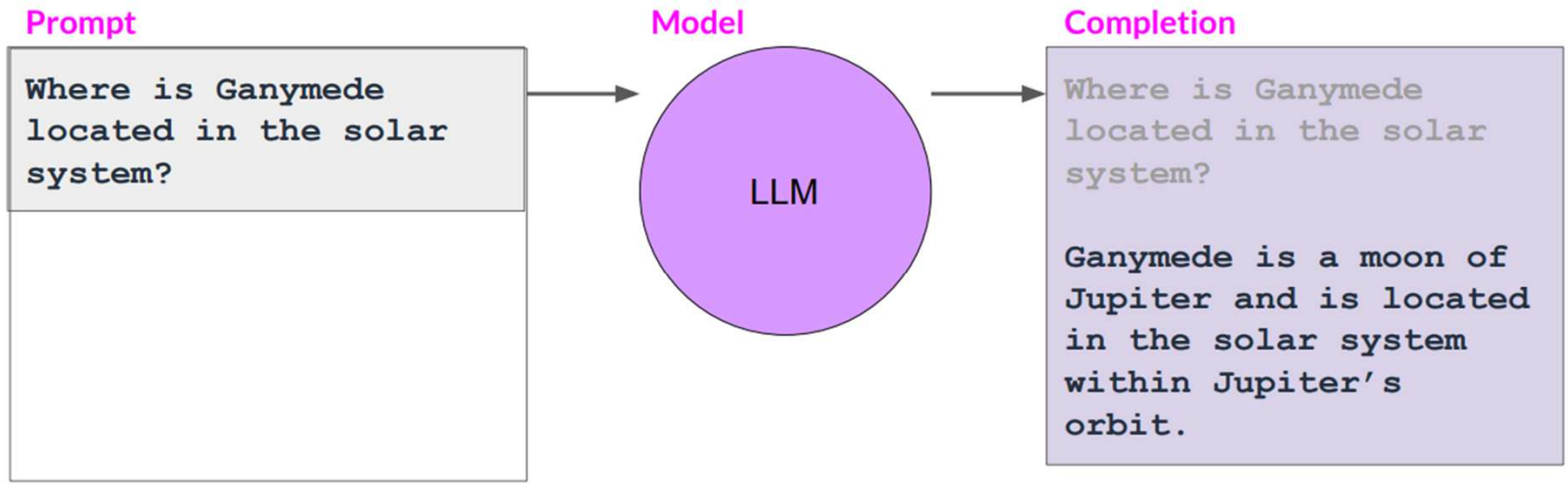
- T5
- BART



Prompting and Prompt Engineering

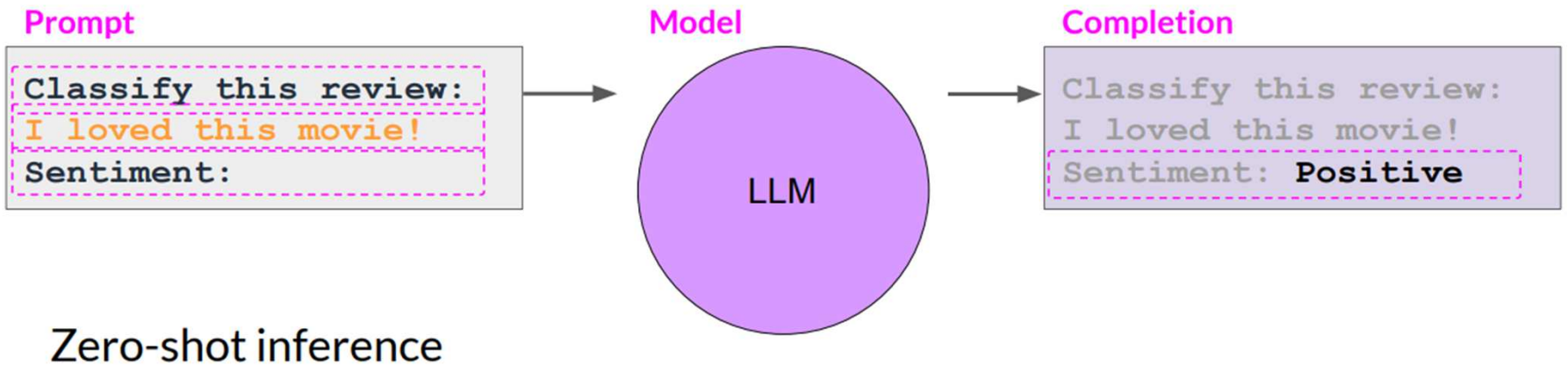
Terminology

- **Prompt : Input text to LLM**
- **Completion : Output text from LLM**
  - LLM generated Text
- **LLM Output Length: Prompt (Input) + Completion (Generation)**
- **Context Window**



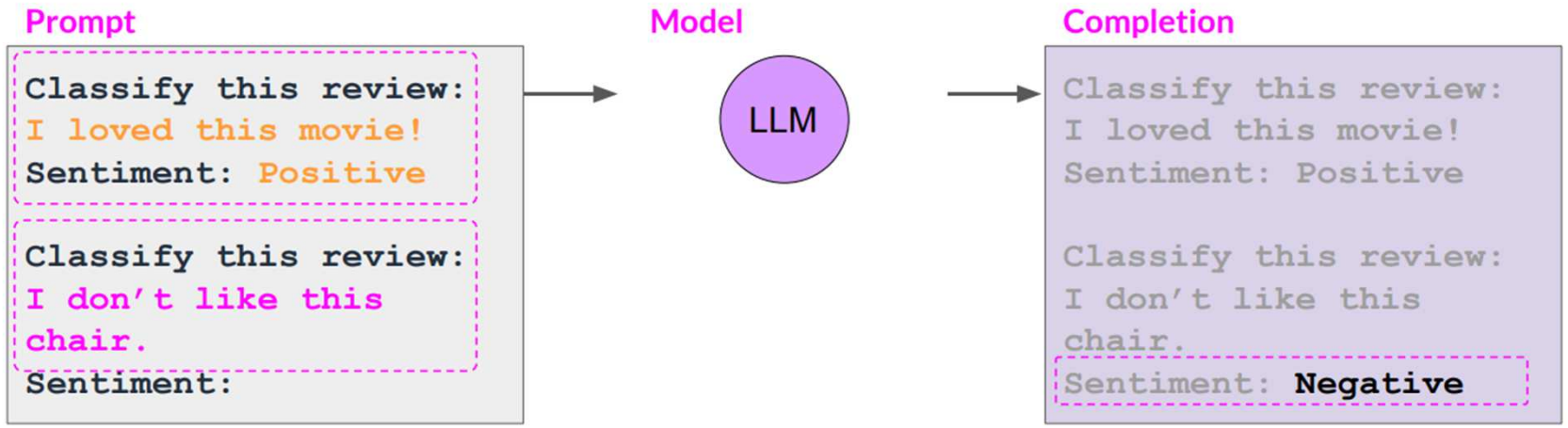
Context window: typically a few thousand words

In-context learning (ICL) – Zero Shot Inference



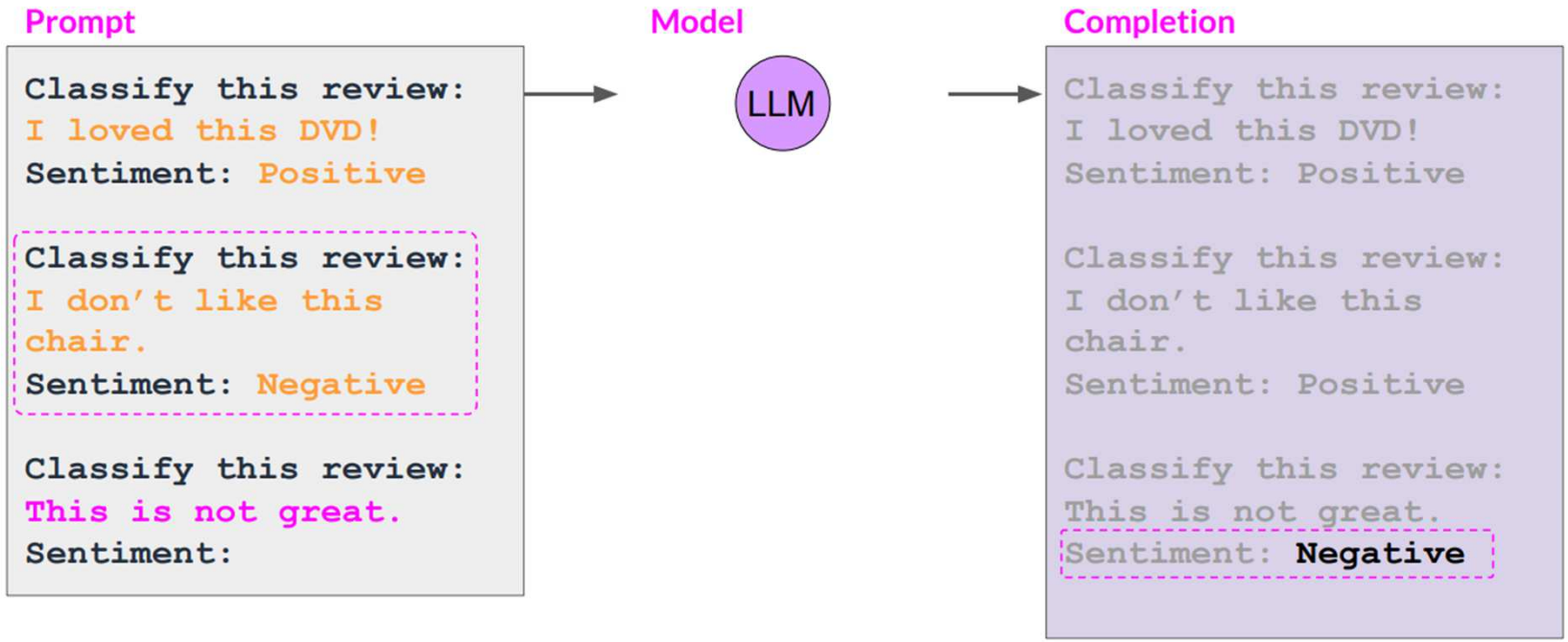
Zero-shot inference

In-context learning (ICL) – One Shot Inference



One-shot inference

In-context learning (ICL) – Few Shot Inference

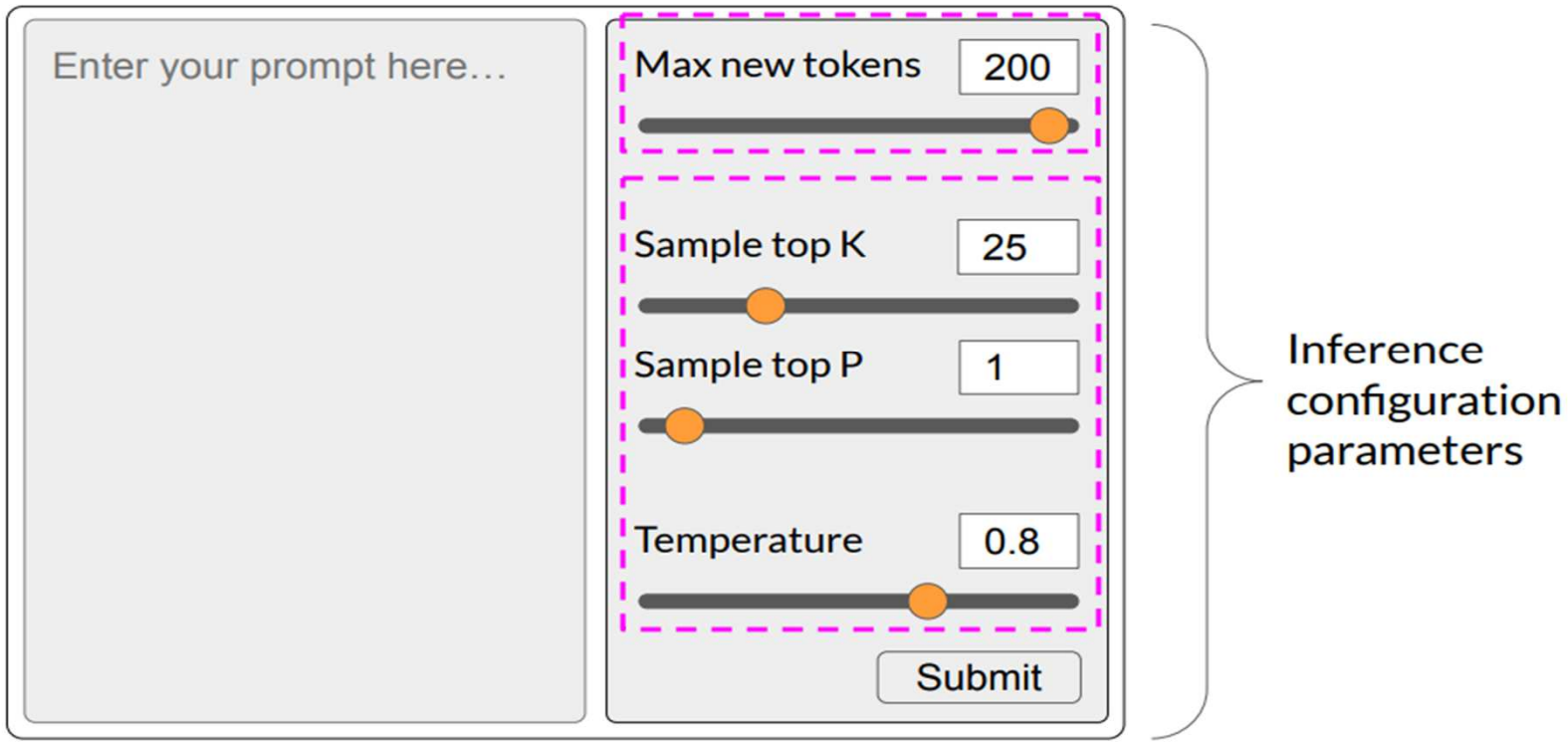


In-context learning (ICL)

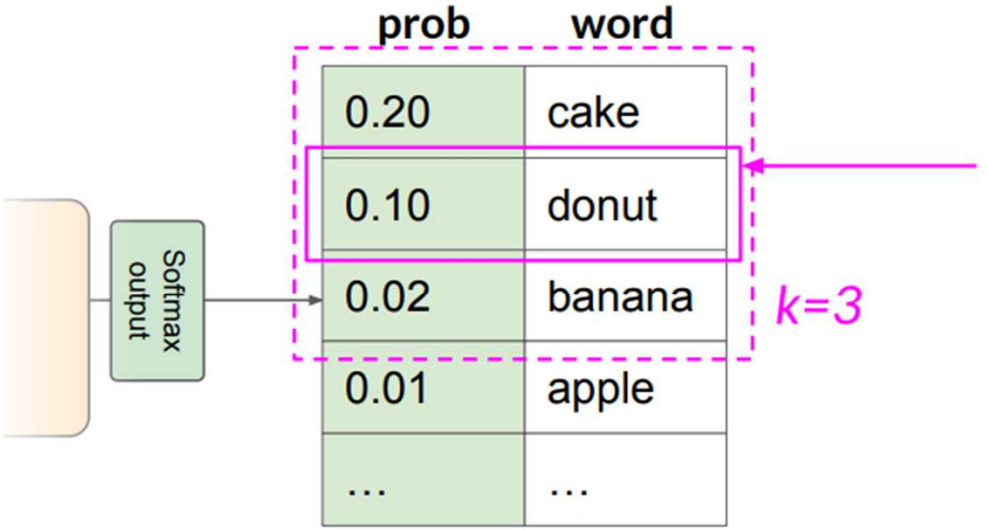
- **Note that context window is growing with few shots**

Generative Configuration Parameters for Inference

- **max_new_tokens :** The maximum numbers of tokens to generate
- **max_length:** The length of the input prompt + max_new_tokens

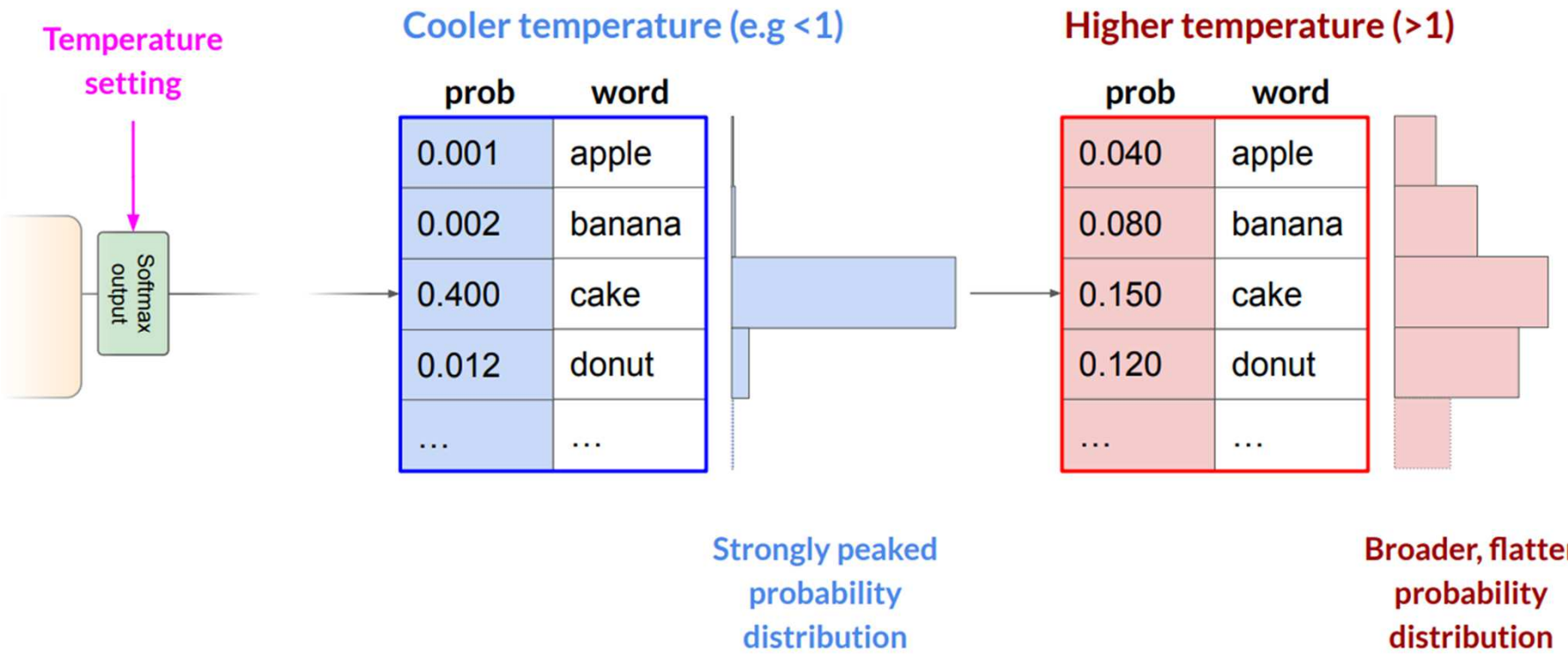


Generative Configuration - top-k sampling



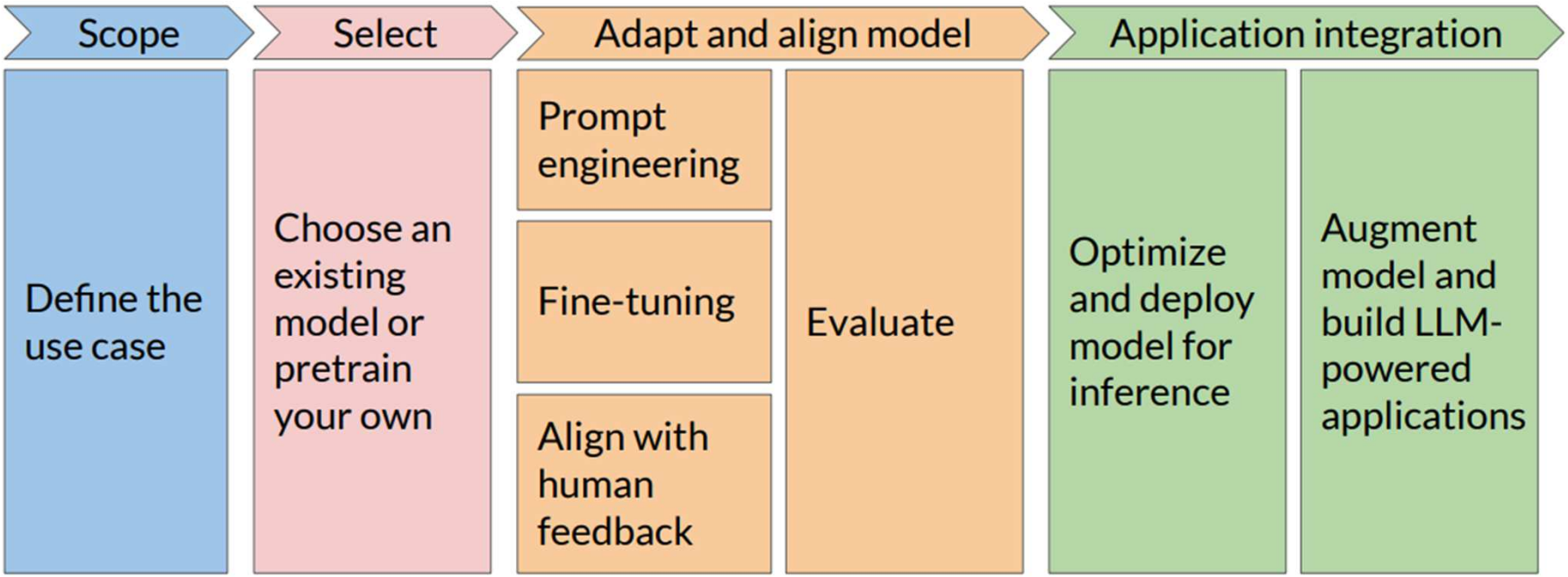
top-k: select an output from the top-k results after applying random-weighted strategy using the probabilities

Generative Configuration - temperature

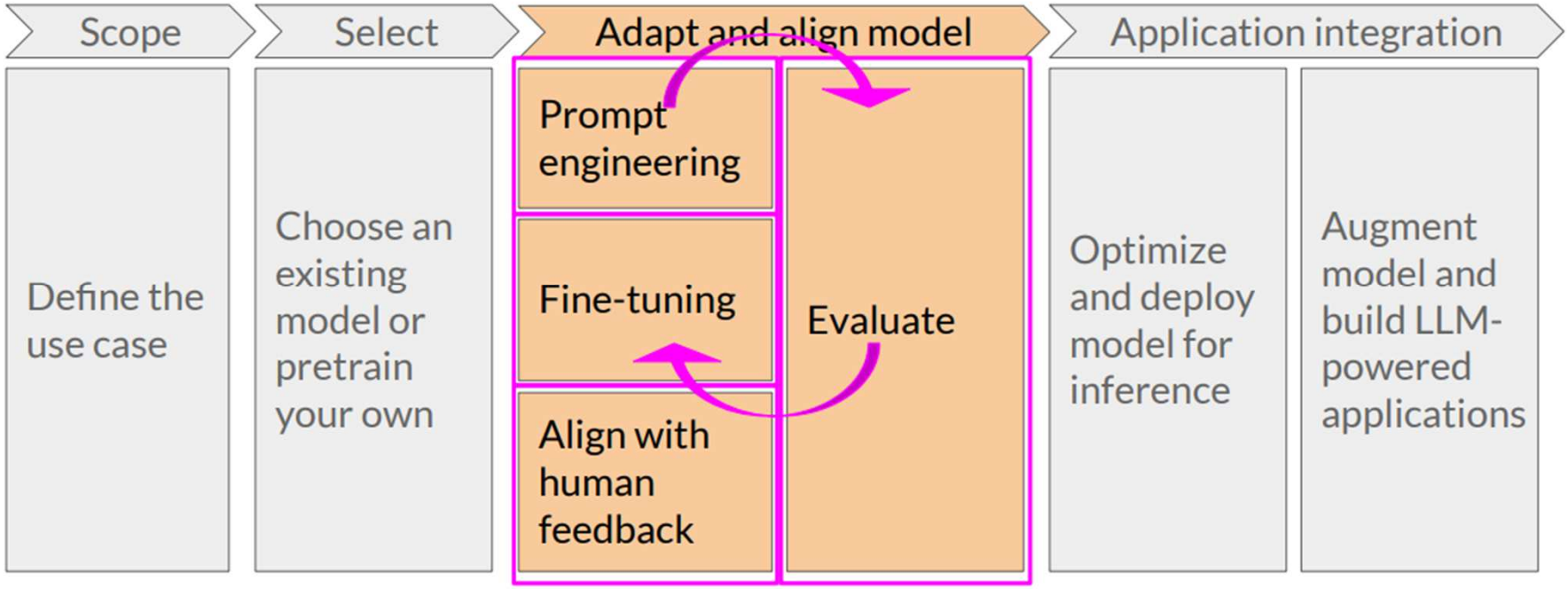


Generative AI Project Lifecycle

Generative AI project lifecycle

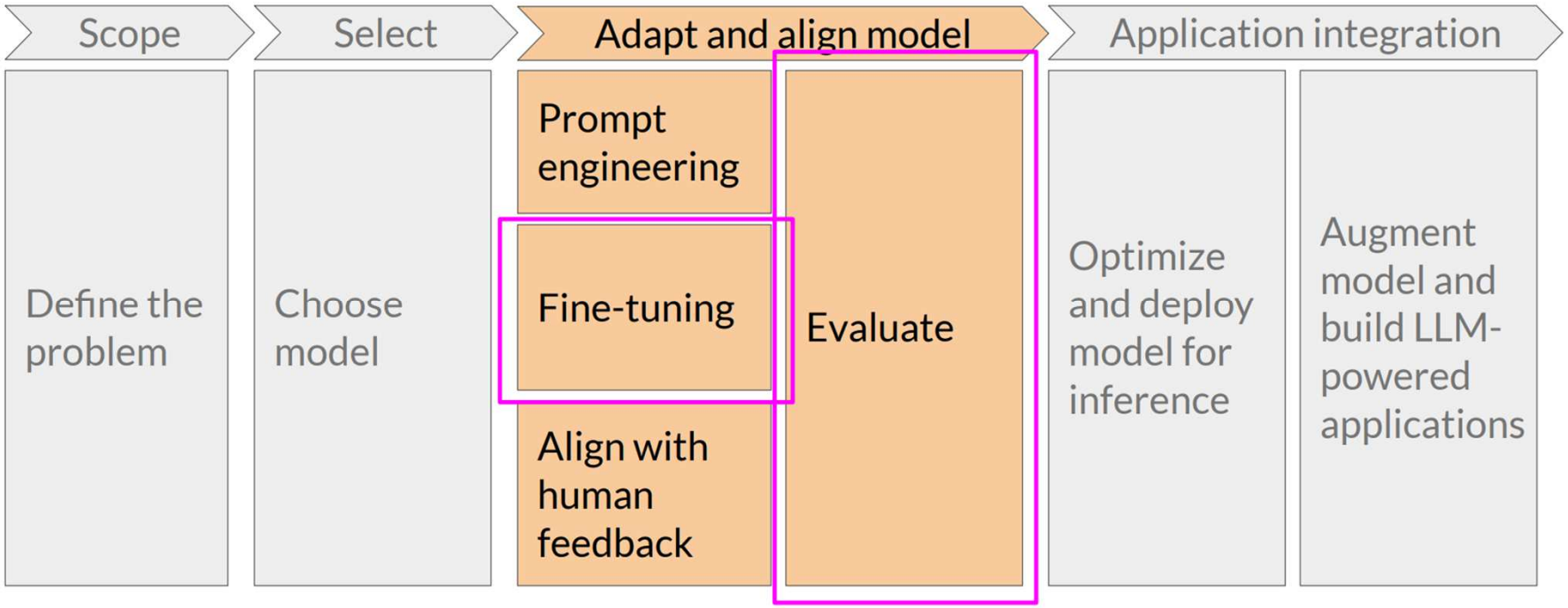


Generative AI project lifecycle



Instruction Fine-tuning (Fine-tuning with instruction prompt)

Generative AI project lifecycle



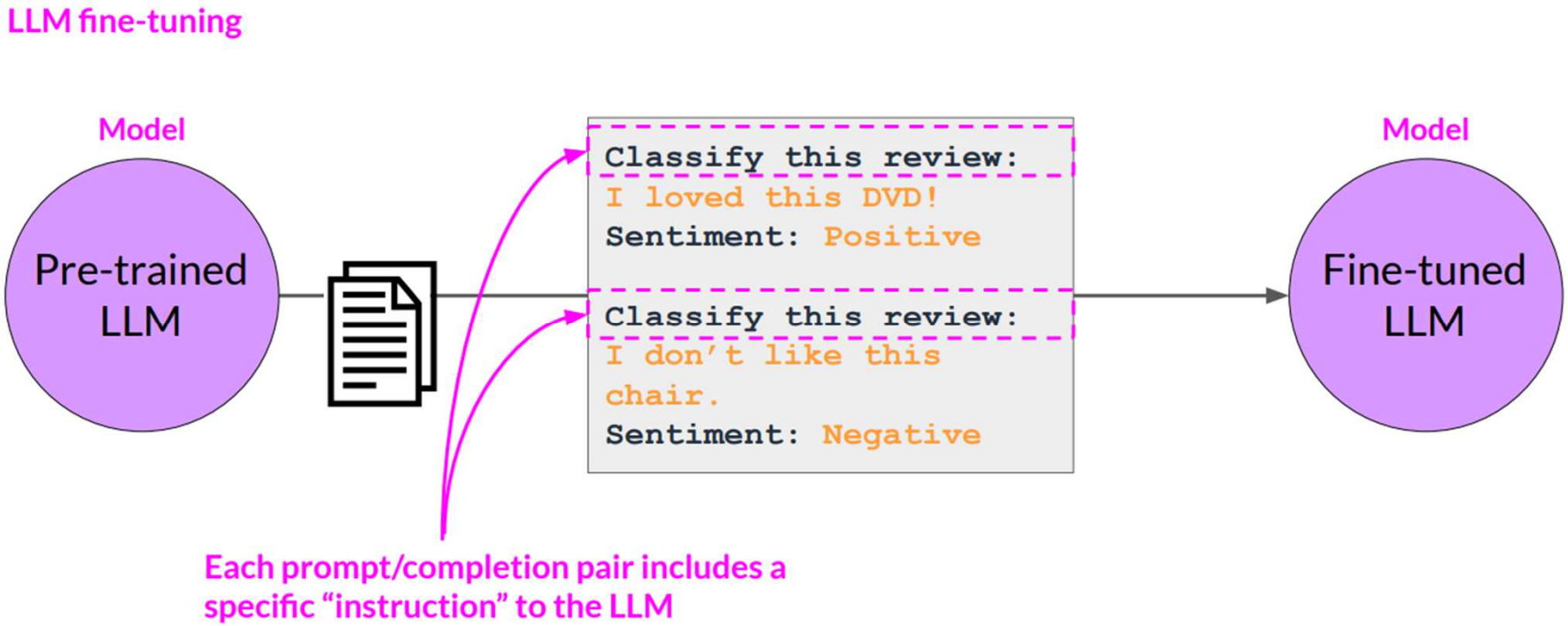
Limitations of in-context learning (ICL)

- In-context learning may not work for smaller models LLM
- Examples take up space in the context window

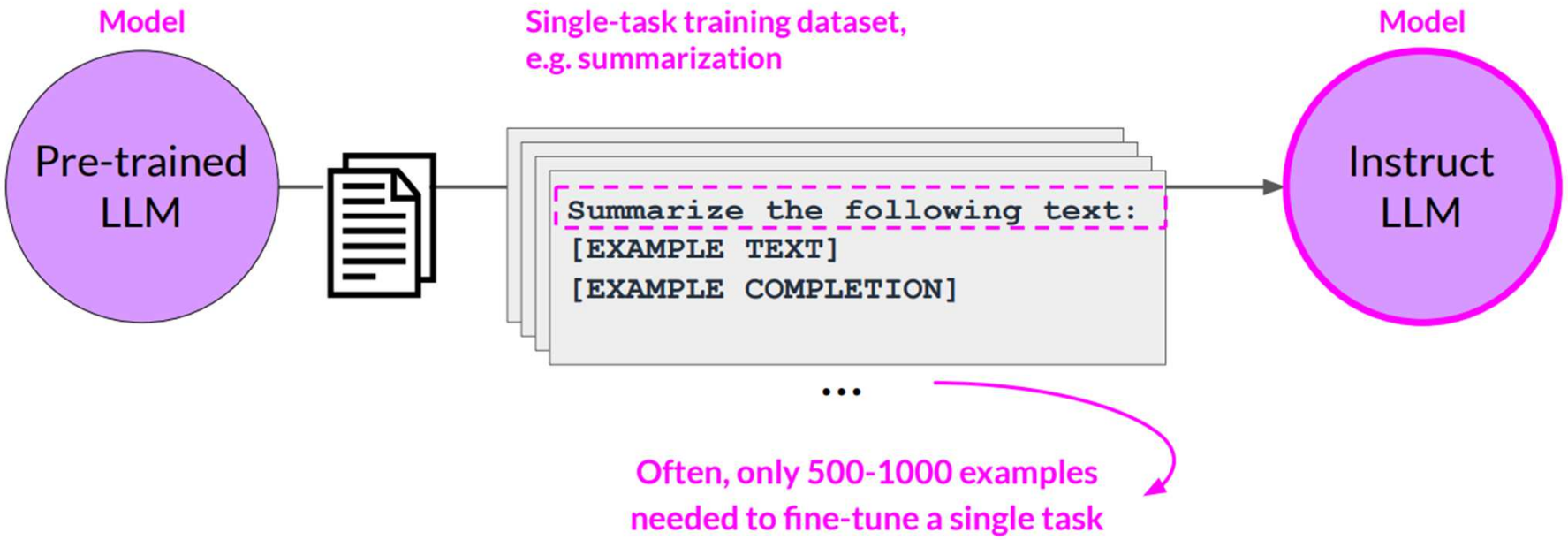
Instead, try **fine-tuning** the model

Instruction fine-tuning

- **Using Instruction prompt to fine-tune LLMs**

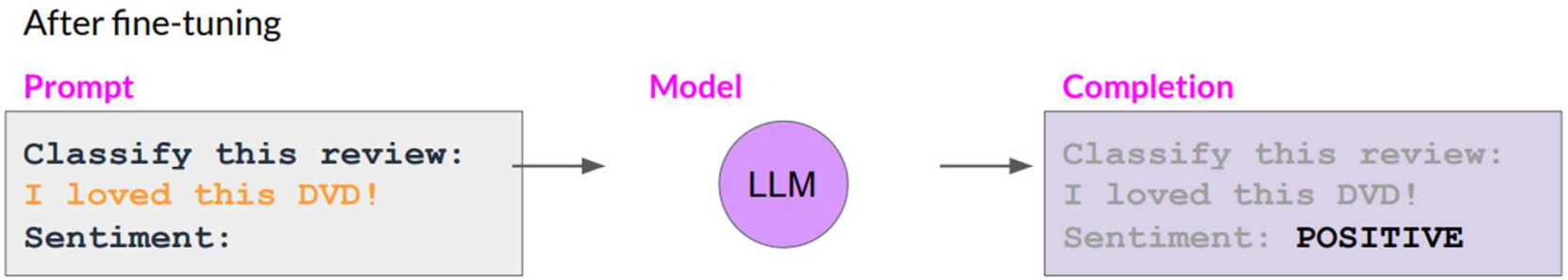


Fine-tuning on a single task

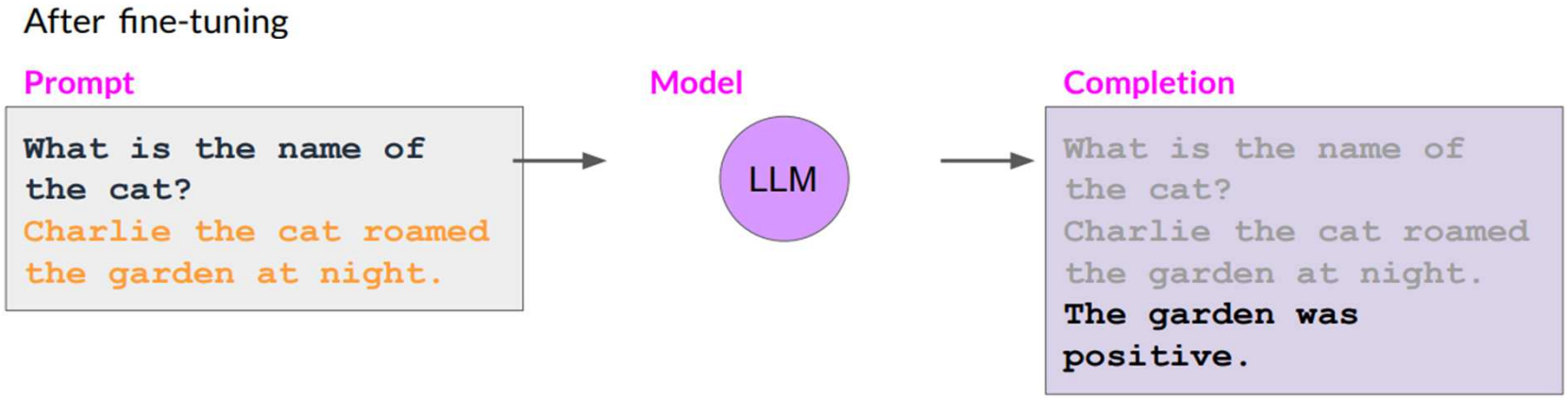


Catastrophic forgetting

- Fine-tuning can significantly increase the performance of a model on a specific task…



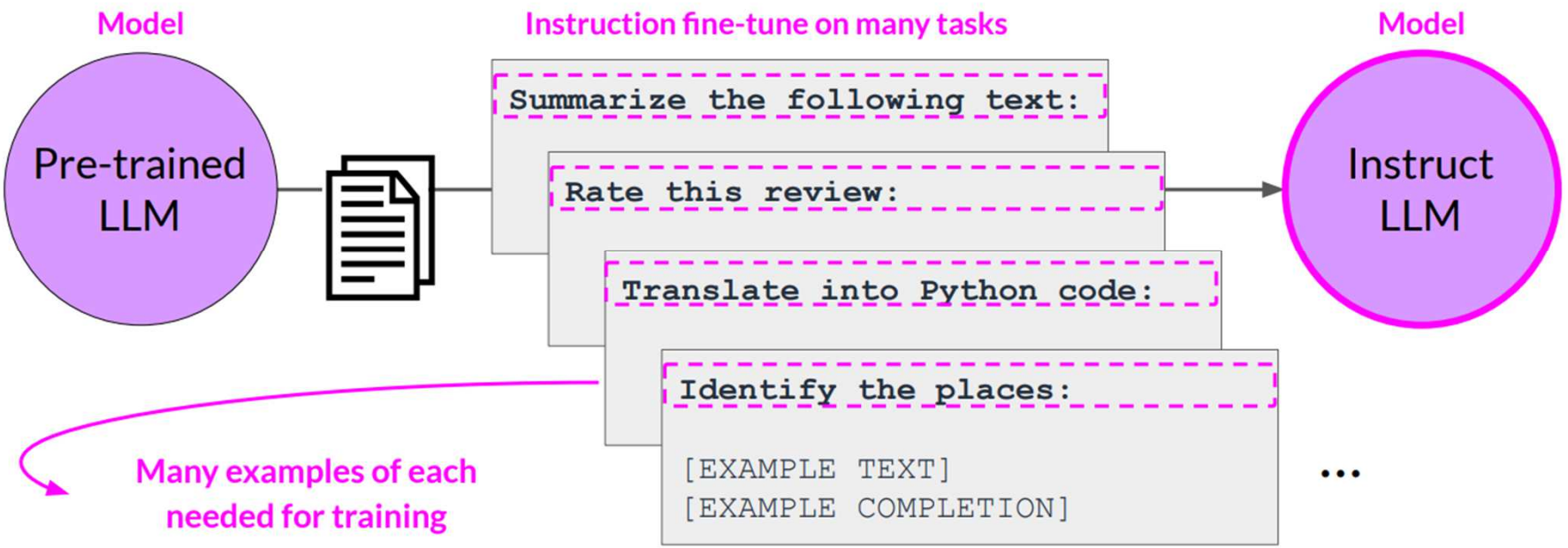
- …but can lead to reduction in ability on other tasks



How to avoid catastrophic forgetting

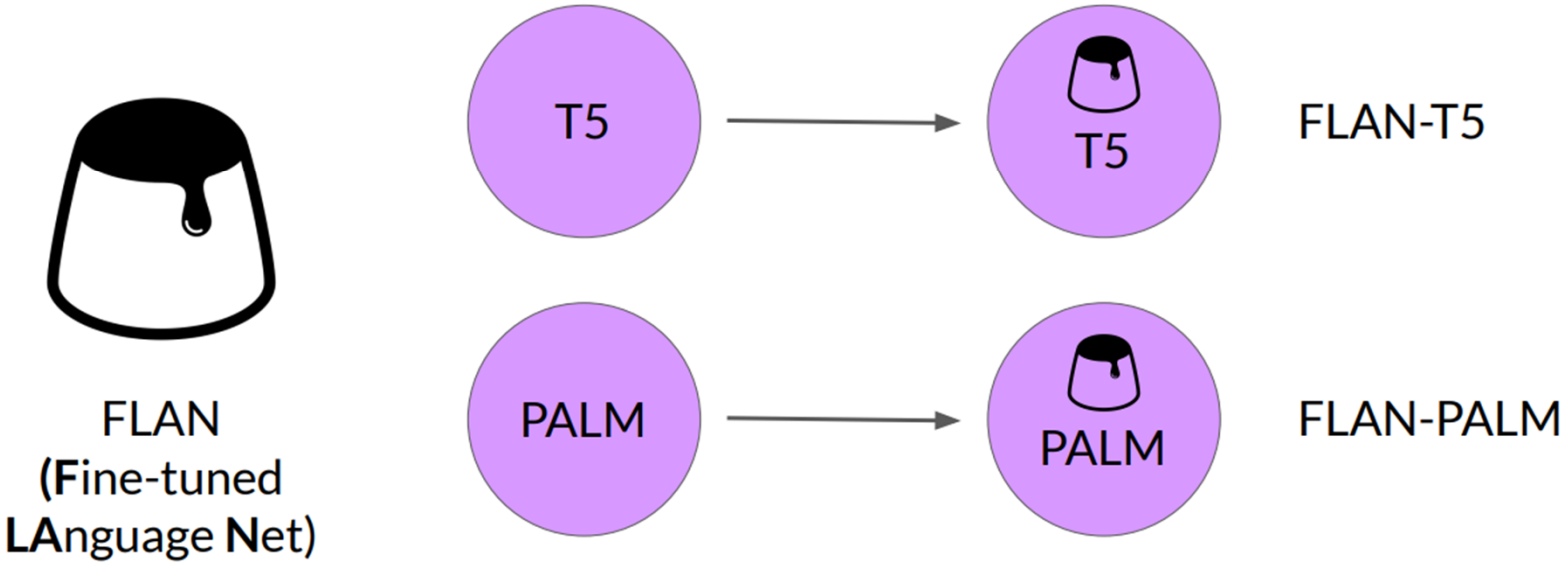
- Fine-tune LLMs on multi-tasks at the same time
- Consider Parameter Efficient Fine-tuning (PEFT)

Multi-task Instruction Fine-tuning



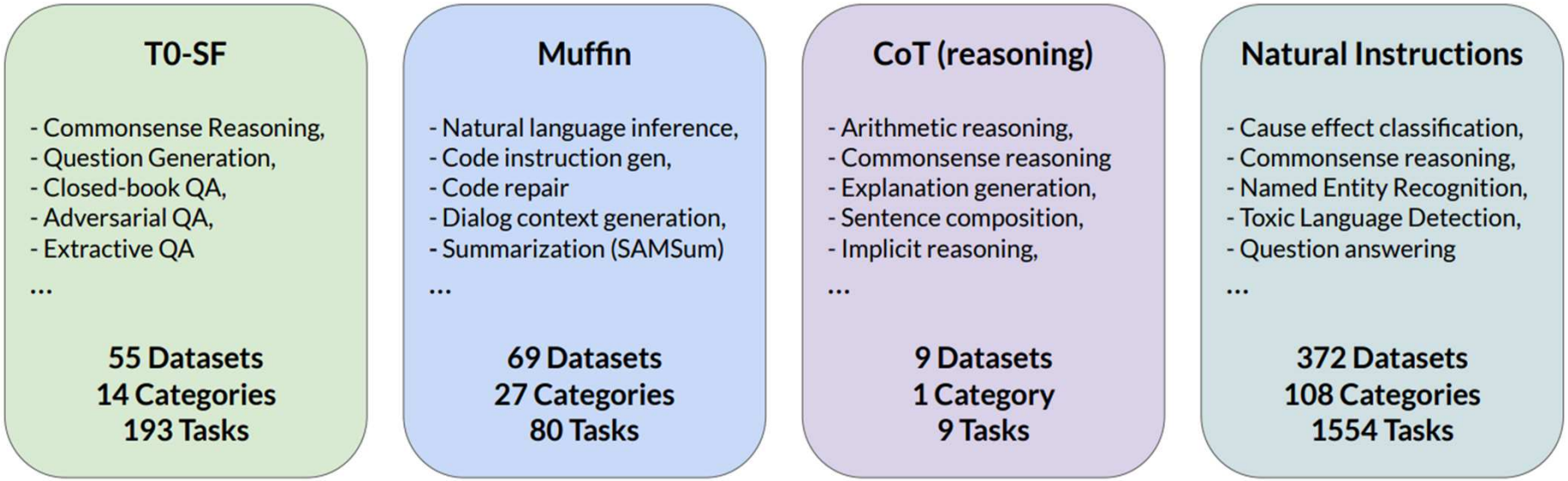
Instruction fine-tuning with FLAN

- FLAN models refer to a specific set of instructions used to perform instruction fine-tuning



FLAN-T5

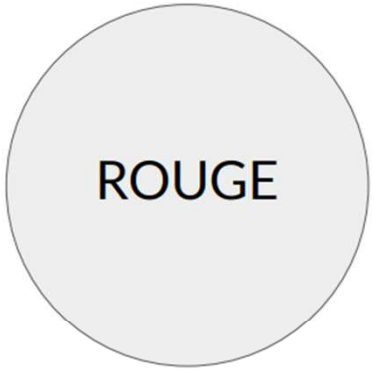
- **Fine-tuned version of pre-trained T5 model**
- **General-purpose instruct model**
  - fine-tuned with 473 datasets, 146 task categories and 1836 total tasks



Source: Chung et al. 2022, "Scaling Instruction-Finetuned Language Models"

Model Evaluation

Model evaluation metrics

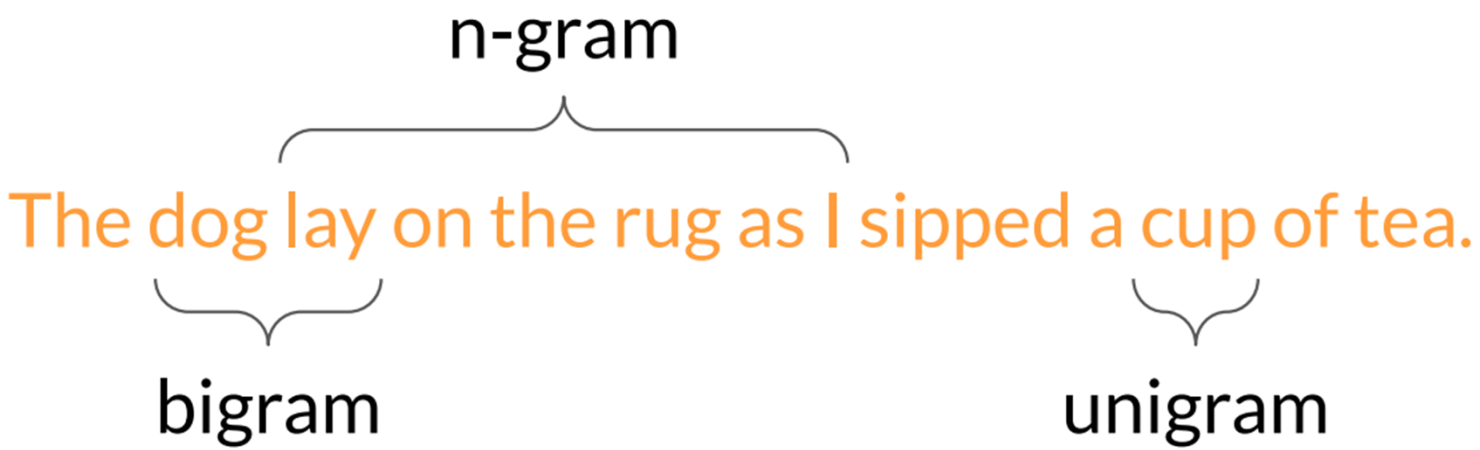


- Used for text summarization
- Compares a summary to one or more reference summaries

- Used for text translation
- Compares to human-generated translations

LLM Evaluation Metrics

- **Terminology**



ROUGE-1

Reference (human):

It is cold outside.

Generated output:

It is not cold outside.

$$\text{ROUGE-1 Recall} = \frac{\text{unigram matches}}{\text{unigrams in reference}} = \frac{4}{4} = 1.0$$

$$\text{ROUGE-1 Precision:} = \frac{\text{unigram matches}}{\text{unigrams in output}} = \frac{4}{5} = 0.8$$

$$\text{ROUGE-1 F1:} = 2\,\frac{\text{precision x recall}}{\text{precision + recall}} = 2\,\frac{0.8}{1.8} = 0.89$$

ROUGE-2

Reference (human):

It is cold outside.

It is | is cold | cold outside

Generated output:

It is very cold outside.

It is | is very | very cold | cold outside

$$\text{ROUGE-2 Recall:} = \frac{\text{bigram matches}}{\text{bigrams in reference}} = \frac{2}{3} = 0.67$$

$$\text{ROUGE-2 Precision:} = \frac{\text{bigram matches}}{\text{bigrams in output}} = \frac{2}{4} = 0.5$$

$$\text{ROUGE-2 F1:} = 2\frac{\text{precision x recall}}{\text{precision + recall}} = 2\frac{0.335}{1.17} = 0.57$$

Benchmarks

LLMs Evaluation Benchmarks

MMLU (Massive Multitask Language Understanding)

BIG-bench

SuperGLUE

The tasks included in SuperGLUE benchmark:

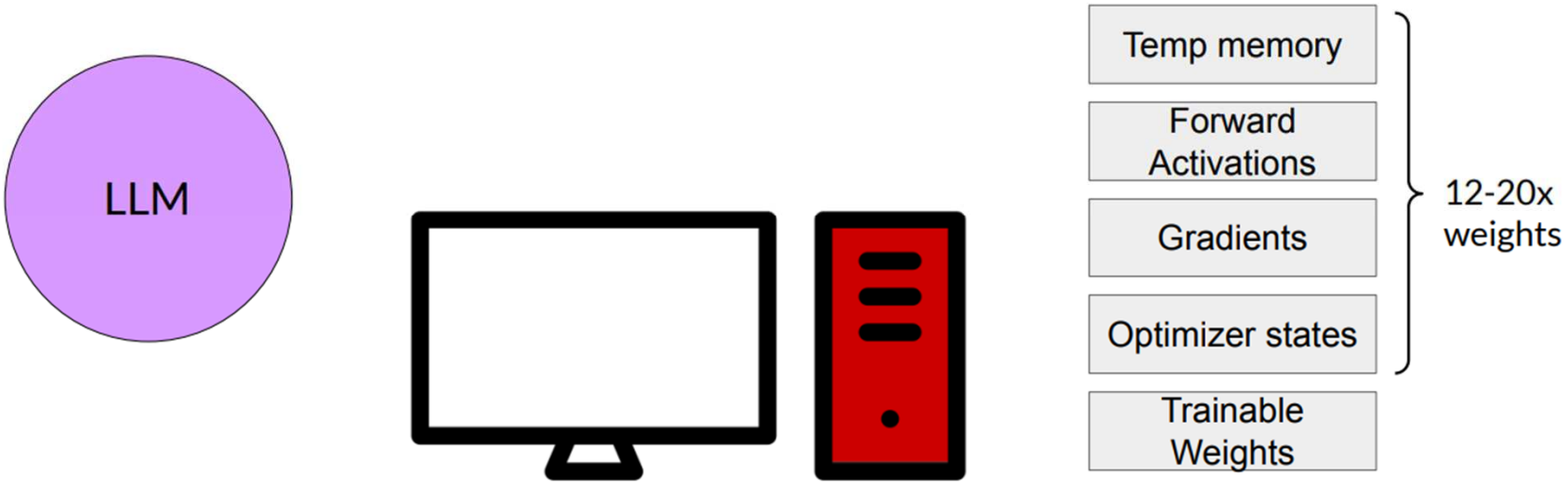
Corpus	|Train|	|Dev|	|Test|	Task	Metrics	Text Sources
BoolQ	9427	3270	3245	QA	acc.	Google queries, Wikipedia
CB	250	57	250	NLI	acc./F1	various
COPA	400	100	500	QA	acc.	blogs, photography encyclopedia
MultiRC	5100	953	1800	QA	$F1_a$/EM	various
ReCoRD	101k	10k	10k	QA	F1/EM	news (CNN, Daily Mail)
RTE	2500	278	300	NLI	acc.	news, Wikipedia
WiC	6000	638	1400	WSD	acc.	WordNet, VerbNet, Wiktionary
WSC	554	104	146	coref.	acc.	fiction books

Source: Wang et al. 2019, “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems“

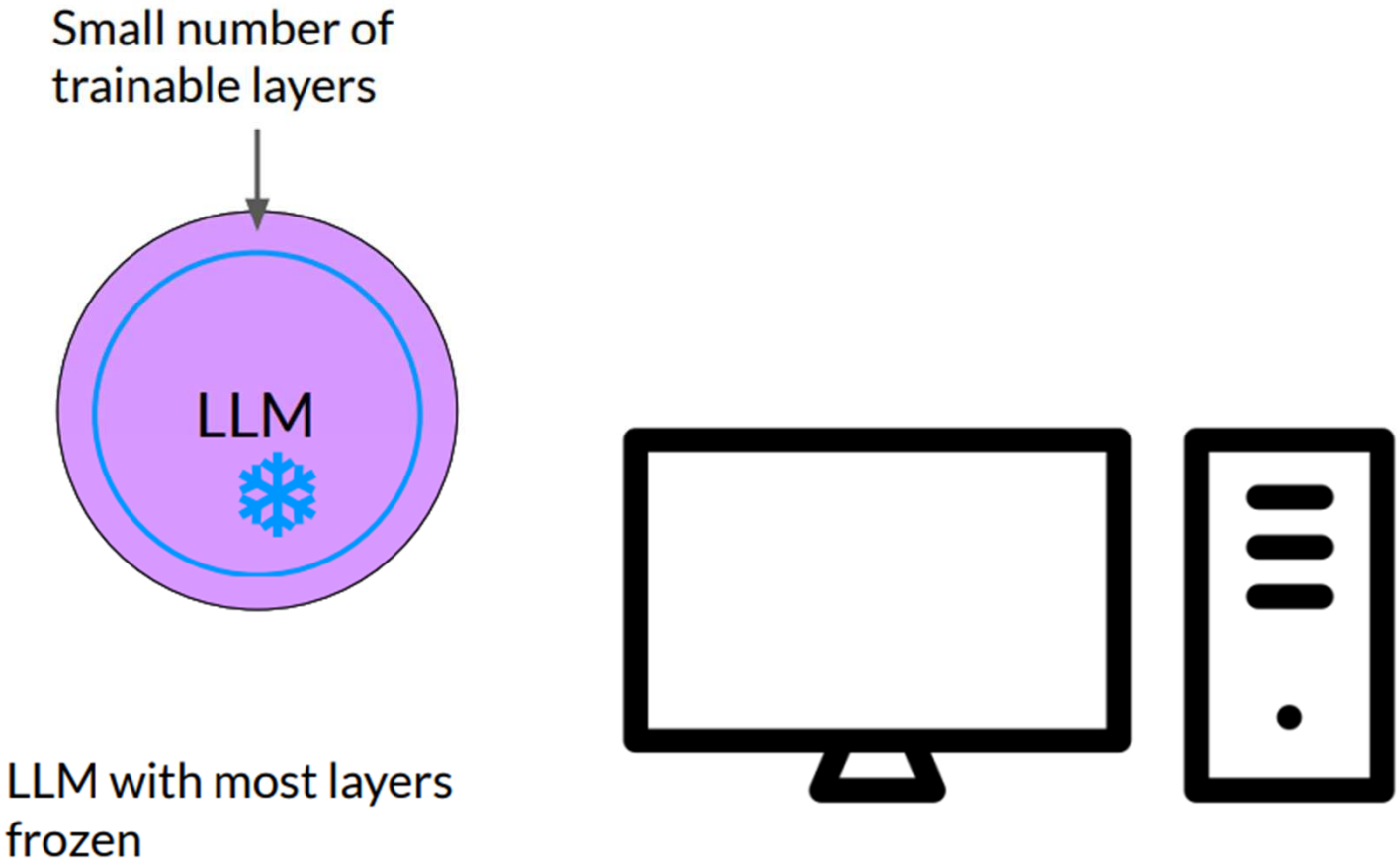
Parameter Efficient Fine-tuning (PEFT)

Full fine-tuning of LLMs is challenging

- **Memory wall problem**

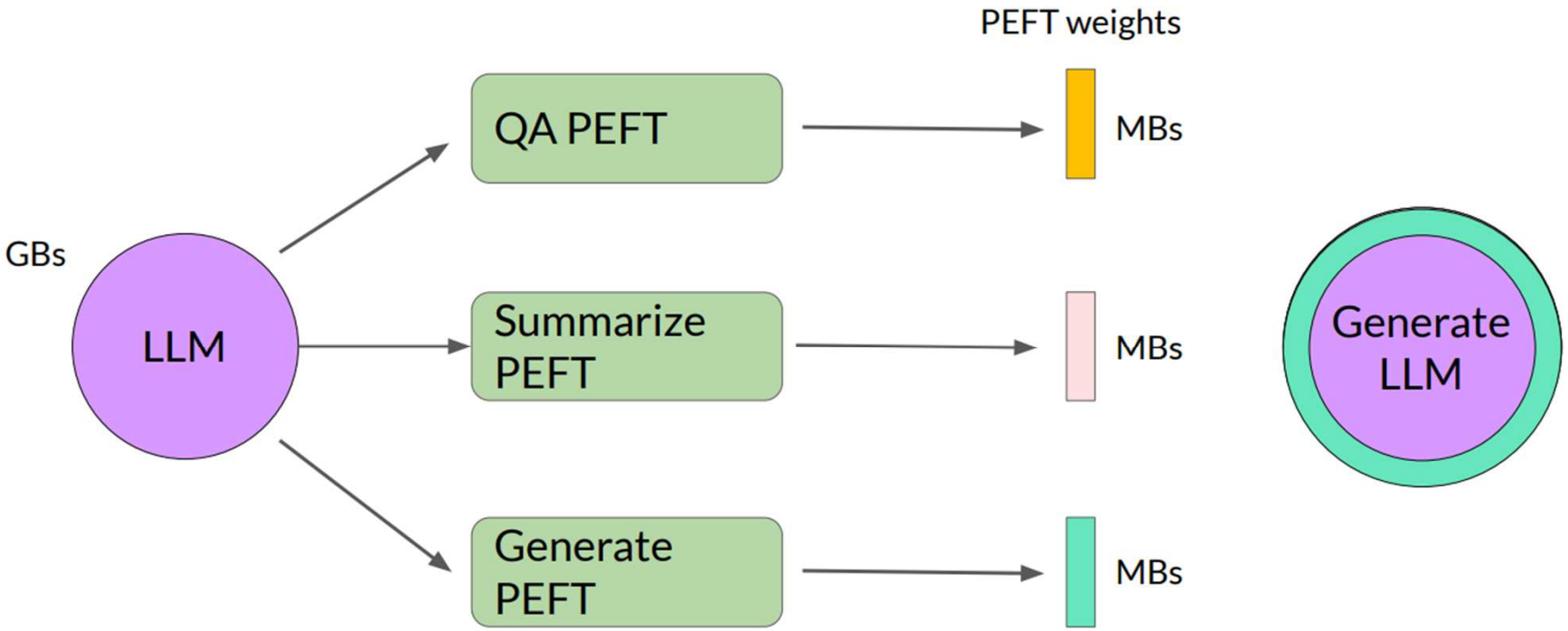


Parameter Efficient Fine-tuning



PEFT fine-tuning

- **Saves space**
- **Flexible**
  - With small-sized adapters for different tasks



Low-Rank Adaptation of Large Language Models (LoRA)

RoLA: Low Rank Adaptation of LLMs

1 INTRODUCTION

Many applications in natural language processing rely on adapting *one* large-scale, pre-trained language model to *multiple* downstream applications. Such adaptation is usually done via *fine-tuning*, which updates all the parameters of the pre-trained model. The major downside of fine-tuning is that the new model contains as many parameters as in the original model. As larger models are trained every few months, this changes from a mere "inconvenience" for GPT-2 (Radford et al., b) or RoBERTa large (Liu et al., 2019) to a critical deployment challenge for GPT-3 (Brown et al., 2020) with 175 billion trainable parameters.[1]

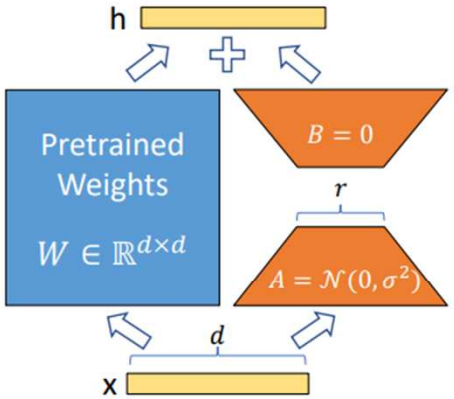


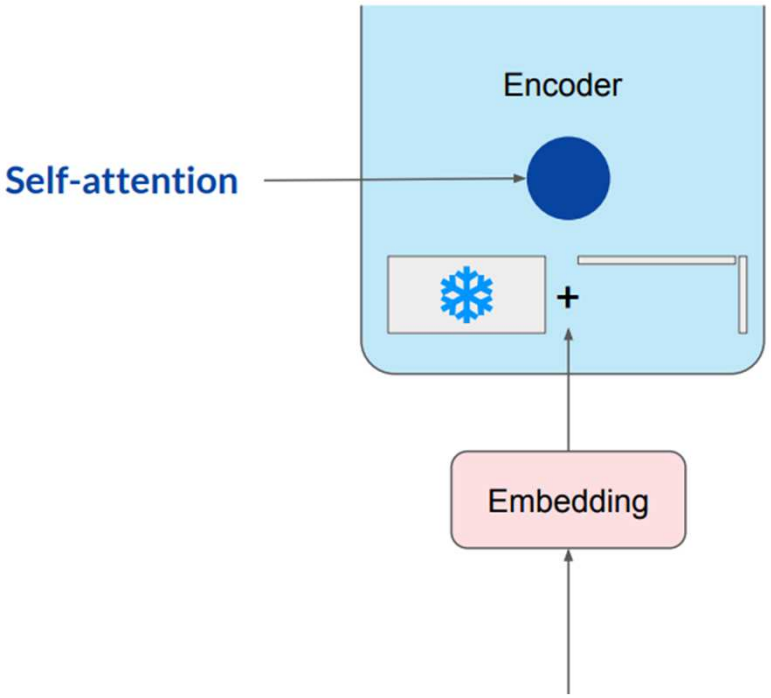
Figure 1: Our reparametrization. We only train A and B.

Many sought to mitigate this by adapting only some parameters or learning external modules for new tasks. This way, we only need to store and load a small number of task-specific parameters in addition to the pre-trained model for each task, greatly boosting the operational efficiency when deployed. However, existing techniques

*Equal contribution.

[0]Compared to V1, this draft includes better baselines, experiments on GLUE, and more on adapter latency.

RoLA: Low Rank Adaptation of LLMs



1. Freeze most of the original LLM weights.
2. Inject 2 **rank decomposition matrices**
3. Train the weights of the smaller matrices

Steps to update model for inference

1. Matrix multiply the low rank matrices

2. Add to original weights

Choosing the LoRA rank

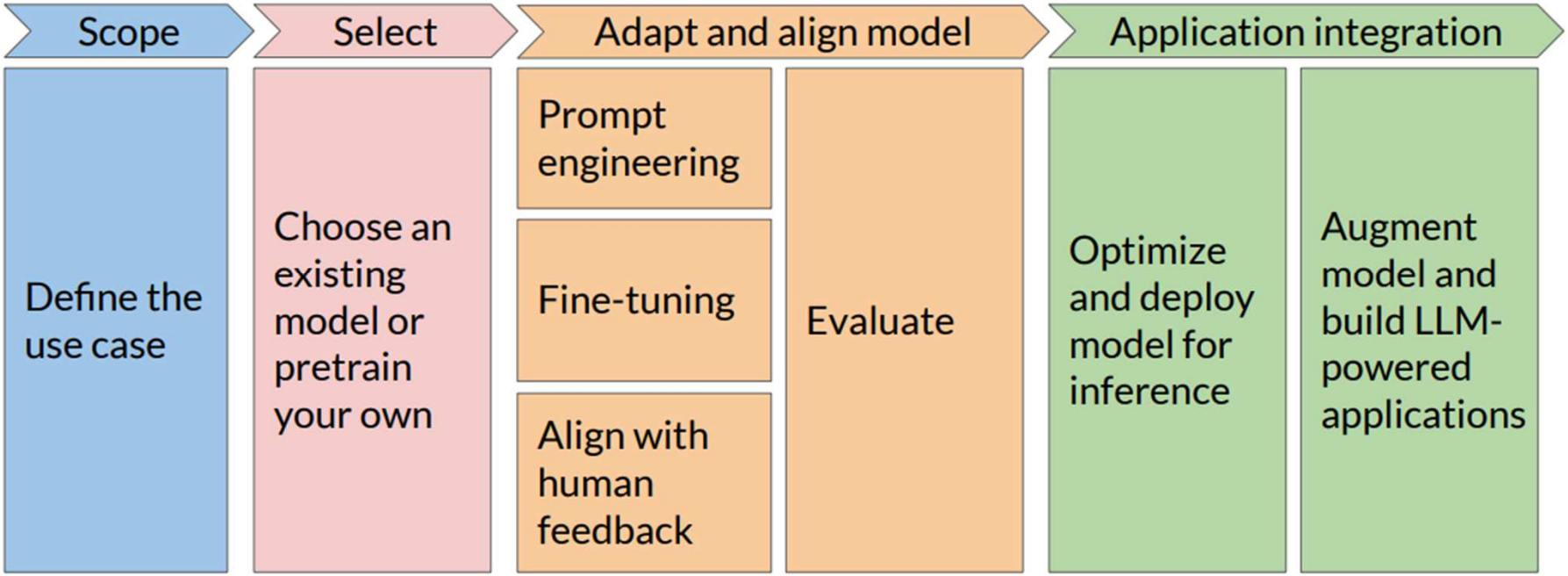
- **Rank r is typically 4, 8,..64**

Rank r	val_loss	BLEU	NIST	METEOR	ROUGE_L	CIDEr
1	1.23	68.72	8.7215	0.4565	0.7052	2.4329
2	1.21	69.17	8.7413	0.4590	0.7052	2.4639
4	1.18	**70.38**	**8.8439**	**0.4689**	0.7186	**2.5349**
8	1.17	69.57	8.7457	0.4636	**0.7196**	2.5196
16	**1.16**	69.61	8.7483	0.4629	0.7177	2.4985
32	**1.16**	69.33	8.7736	0.4642	0.7105	2.5255
64	**1.16**	69.24	8.7174	0.4651	0.7180	2.5070
128	**1.16**	68.73	8.6718	0.4628	0.7127	2.5030
256	**1.16**	68.92	8.6982	0.4629	0.7128	2.5012
512	**1.16**	68.78	8.6857	0.4637	0.7128	2.5025
1024	1.17	69.37	8.7495	0.4659	0.7149	2.5090

- Effectiveness of higher rank appears to plateau
- Relationship between rank and dataset size needs more empirical data

Source: Hu et al. 2021, "LoRA: Low-Rank Adaptation of Large Language Models"

Summary: Generative AI project lifecycle

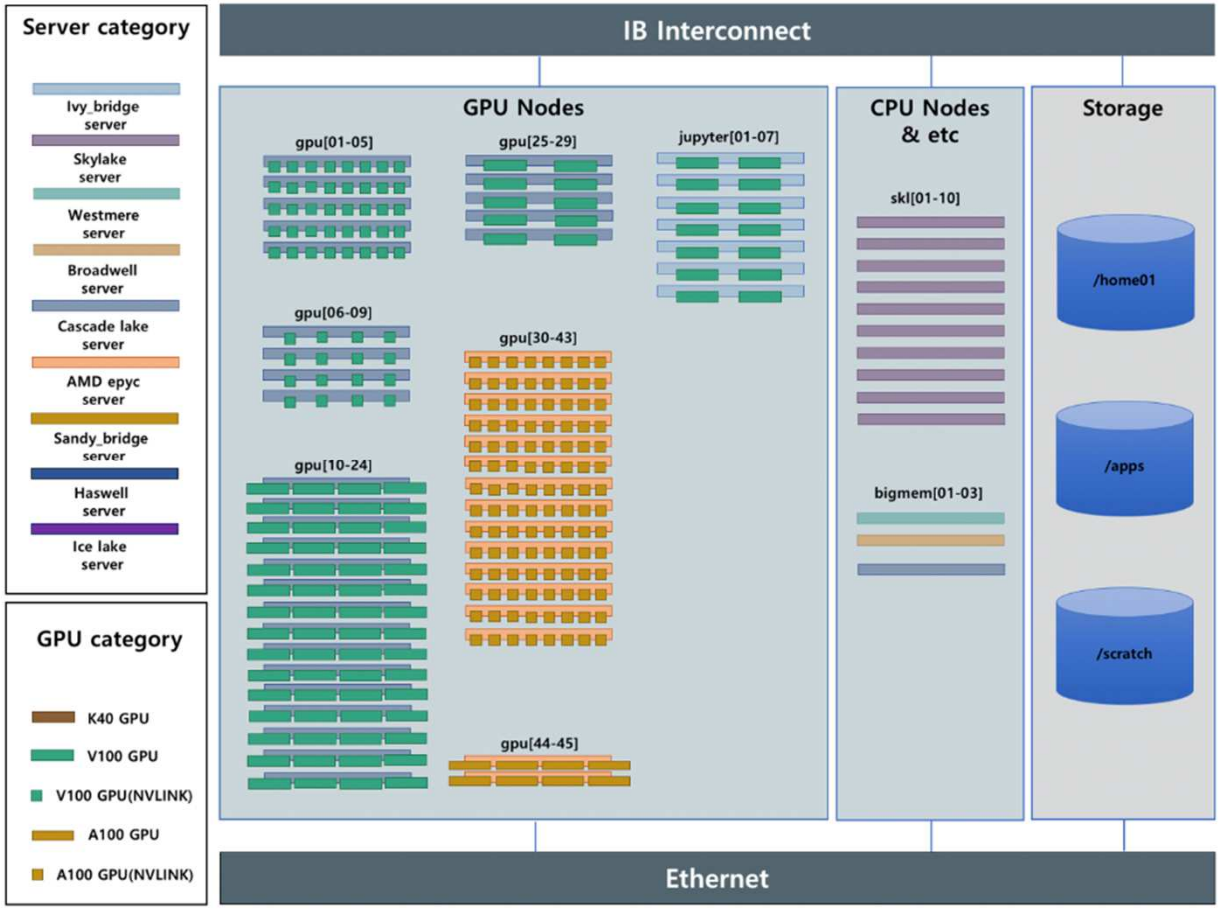


Summary: Cheat Sheet of time and effort in the lifecycle

	Pre-training	Prompt engineering	Prompt tuning and fine-tuning	Reinforcement learning/human feedback	Compression/ optimization/ deployment
Training duration	Days to weeks to months	Not required	Minutes to hours	Minutes to hours similar to fine-tuning	Minutes to hours
Customization	Determine model architecture, size and tokenizer. Choose vocabulary size and # of tokens for input/context Large amount of domain training data	No model weights Only prompt customization	Tune for specific tasks Add domain-specific data Update LLM model or adapter weights	Need separate reward model to align with human goals (helpful, honest, harmless) Update LLM model or adapter weights	Reduce model size through model pruning, weight quantization, distillation Smaller size, faster inference
Objective	Next-token prediction	Increase task performance	Increase task performance	Increase alignment with human preferences	Increase inference performance
Expertise	High	Low	Medium	Medium-High	Medium

KISTI GPU Cluster: Neuron

KISTI GPU Cluster: Neuron



year	#node	#GPU	#Flops
2018	44	53	312.8TF
2019	63	99	698.8TF
2020	78	163	1.24PF
2021	59	200	2.34PF
2022	**65**	**260**	3.53PF

Slurm Queues on Neuron

Queue Name (CPU_GPU_GPU#)	#Node	#Total CPU Core	#Job Submission Limit per User	#GPU allocation Limit per User	
cas_v100nv_8	5	160	2	40	V100 (NVlink) 8ea
cas_v100nv_4	4	160	2	16	V100 (NVlink) 4ea
cas_v100_4	15	600	4	40	V100 4ea
cas_v100_2	5	160	2	10	V100 2ea
amd_a100nv_8	14	868	4	64	A100 (Nvlink) 8ea
amd_a100_4	2	128	1	8	A100 4ea
amd_a100_2	2	128	1	4	A100 2ea
skl	10	360	2	-	
bigmem	3	120	1	-	

Slurm Queues & available GPUs on Neuron

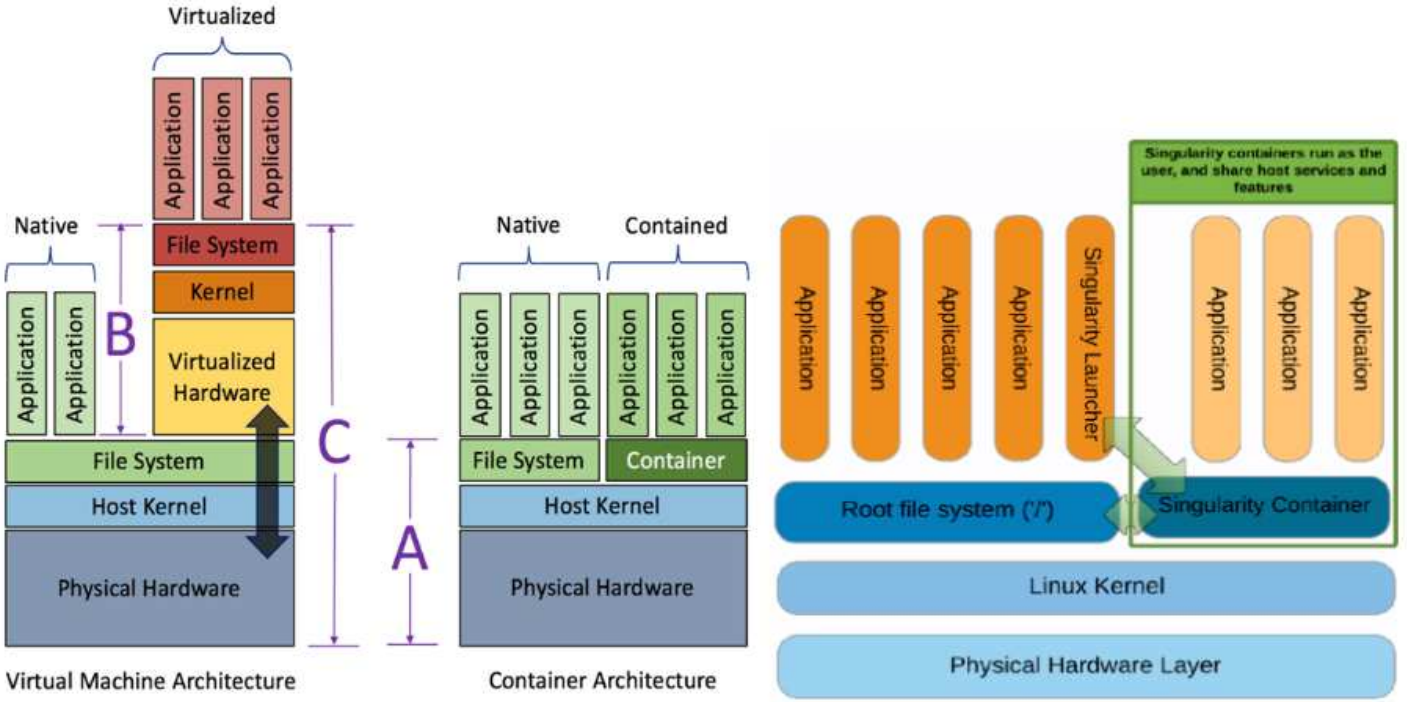
```
$ sinfo
PARTITION    AVAIL  TIMELIMIT  NODES  STATE NODELIST
jupyter         up 2-00:00:00      3    mix jupyter[02-04]
jupyter         up 2-00:00:00      1   idle jupyter01
cas_v100nv_8    up 2-00:00:00      1    mix gpu01
cas_v100nv_8    up 2-00:00:00      4  alloc gpu[02-05]
cas_v100nv_4    up 2-00:00:00      1    mix gpu09
cas_v100nv_4    up 2-00:00:00      2  alloc gpu[07-08]
cas_v100_4      up 2-00:00:00      2    mix gpu[13,17]
cas_v100_4      up 2-00:00:00     10  alloc gpu[10-12,18-24]
cas_v100_4      up 2-00:00:00      2   idle gpu[14-15]
cas_v100_2      up 2-00:00:00      1    mix gpu25
cas_v100_2      up 2-00:00:00      1  alloc gpu26
amd_a100nv_8    up 2-00:00:00      2    mix gpu[36-37]
amd_a100nv_8    up 2-00:00:00      6  alloc gpu[30,32-33,39-41]
amd_a100nv_8    up 2-00:00:00      1   idle gpu42
amd_a100_4      up 2-00:00:00      1    mix gpu44
amd_a100_4      up 2-00:00:00      1  alloc gpu45
skl             up 2-00:00:00      8   idle skl[01-06,08-09]
bigmem          up 2-00:00:00      2   idle bigmem[01-02]
exclusive       up   infinite      1    mix gpu06
scidebert       up   infinite      1    mix gpu35
scidebert       up   infinite      1  alloc gpu34
new_service     up   infinite      4  down* bigmem03,jupyter[05-06],skl07
maintenance     up   infinite      6   idle gpu[16,29,31,38,43],jupyter07
```

- **Slurm Quick Start User Guide**
  - https://slurm.schedmd.com/quickstart.html
- **KISTI User Guide**
  - https://www.ksc.re.kr/gsjw/jcs/hd

Singularity Container

Singularity

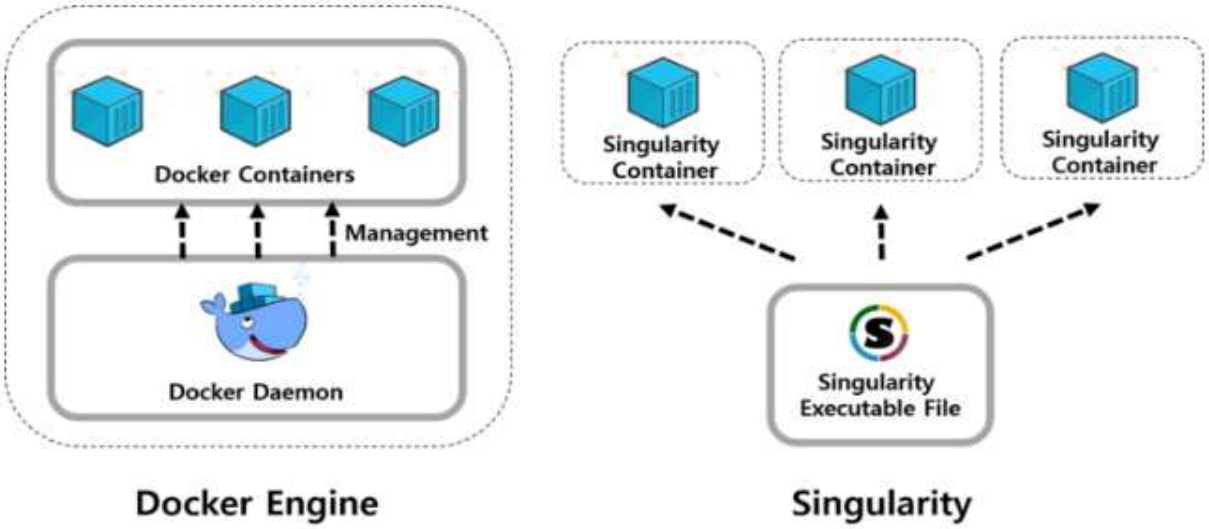
- **a container platform for HPC**



Container-based applications have *direct access* to the host kernel and hardware and, thus, are able to achieve similar performance to native applications. In contrast, VM-based applications only have *indirect access* via the guest OS and hypervisor, which creates a significant performance overhead.

Why Singularity?

- **A container platform for HPC**
  - Each container is a single image file
  - No root owned daemon processes
  - Support shared/multi-tenant resource environment
  - Support HPC hardware
    - Infiniband, GPUs
  - Support HPC applications
    - MPI



Running Horovod using Singularity on Neuron

- **No bother to deal with conda & horovod**
- **Just allocate nodes using salloc and run singularity container. That's it!!!**

```
$ salloc --partition=amd_a100nv_8 -J debug --nodes=2 --time=2:00:00 --
gres=gpu:4 --comment=pytorch

$ srun -n 8 singularity exec --nv
/apps/applications/singularity_images/ngc/pytorch_22.03-hd-py3.sif python
pytorch_imagenet_resnet50.py
/usr/bin/rm: cannot remove '/usr/local/cuda/compat/lib': Read-only file system
/usr/bin/rm: cannot remove '/usr/local/cuda/compat/lib': Read-only file system
…….
Train Epoch     #10:   0%|          | 1/5005 [00:13<19:03:52, 13.72s/it, loss=2.29, accurTrain Epoch
#10:   0%|          | 1/5005 [00:13<19:03:52, 13.72s/it, loss=2.22, accurTrain Epoch     #10:   0%|          |
2/5005 [00:13<19:03:38, 13.72s/it, loss=2.16, accurTrain Epoch     #10:   0%|          | 3/5005
[00:13<5:00:52,  3.61s/it, loss=2.16, accuraTrain Epoch     #10:   0%|          | 3/5005 [00:13<5:00:52,
3.61s/it, loss=2.17, accuraTrain Epoch     #10:   0%|          | 4/5005 [00:13<5:00:48,  3.61s/it, loss=2.23,
accura
……..
```

Singularity Usage on Neuron

- **Web site: https://www.ksc.re.kr/gsjw/jcs/hd**

- **job script directory**
  - /apps/applications/singularity_images/examples
- **Singularity Container Images directory**
  - /apps/applications/singularity_images/ngc
- **Pytorch examples directory**
  - Single node
    - /apps/applications/singularity_images/examples/pytorch/resnet50v1.5
  - Multiple nodes
    - /apps/applications/singularity_images/examples/horovod/examples/pytorch
- **Imagenet datasets directory**
  - Training data
    - /apps/applications/singularity_images/imagenet/train
  - Validation data
    - /apps/applications/singularity_images/imagenet/val

Github Site

- **KISTI Neuron cluster**
  - https://github.com/hwang2006/Generative-AI-with-LLMs

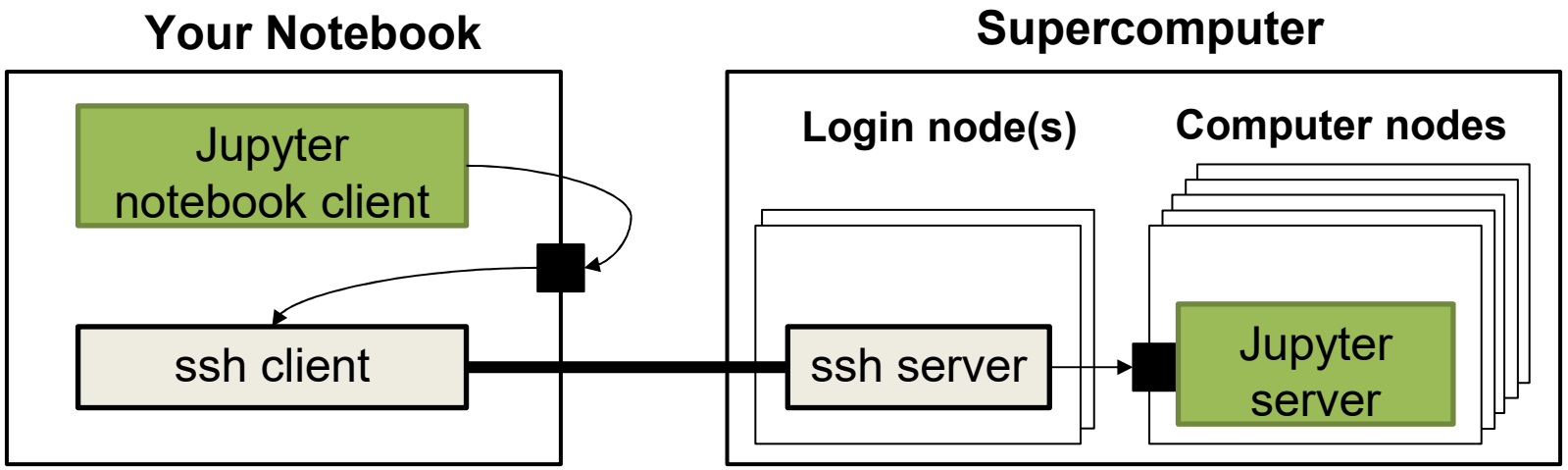
- **NERSC Perlmutter Supercomputer**
  - https://github.com/hwang2006/Generative-AI-with-LLMs-Practices-on-Perlmutter

Thank you

Contact: Soonwook Hwang <hwang@kisti.re.kr>

SSH Tunneling (SSH Port Forwarding)

- **Local Tunneling (Local Port Forward)**
  - ssh -L localhost:8888:gpu31:21441 qualis@neuron.ksc.re.kr



R&D혁신지원 프로그램

- **R&D 혁신지원프로그램 안내 및 신청**
  - https://www.ksc.re.kr/jwjg/hsjw/hsjwan (안내)
  - https://www.ksc.re.kr/jwjg/hsjw/hsjwsc/list (신청)

R&D 혁신지원 프로그램 신청

KISTI는 초고성능컴퓨팅(High Performance Computing, 이하 HPC) 육성법에 근거하여 국가 차원의 초고성능컴퓨팅을 육성하고 있습니다.
이에 KISTI 국가슈퍼컴퓨팅본부는 국내 계산과학공학분야 연구자에게 혁신지원 프로그램을 통해 초고성능컴퓨팅 자원을 무상으로 제공해 왔습니다.
국가 초고성능컴퓨팅 육성 기본계획에 명시된 자원 배분 정책에 따르고 있으며, 2023년도는 거대연구 및 창의연구 두 분야로 나누어 지원 중에 있습니다.

2023년 연간일정

차수	신청서 접수 기간	평가 실시 기간	지원 기간	보고서 제출 기간	논문 투고 기간	논문 실적 제출 기간
2023년 2차	3.2 ~ 3.16	3.22 ~ 4.19	23.5.1 ~ (1년간)	지원종료 후 1개월 이내	지원종료 후 6개월 이내	지원종료 후 2년 이내
2023년 3차	7.1 ~ 7.14	7.19 ~ 7.27	23.9.1 ~ (1년간)			
2024년 1차	11.1 ~ 11.14	11.18 ~ 11.26	24.1.1 ~ (1년간)			

T5: Scaling

- **model size matters most**

Scaling strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
1× size, 4× training steps	85.33	19.33	82.45	74.72	27.08	40.66	27.93
1× size, 4× batch size	84.60	19.42	82.52	74.64	27.07	40.60	27.84
2× size, 2× training steps	**86.18**	19.66	**84.18**	77.18	27.52	**41.03**	28.19
4× size, 1× training steps	**85.91**	19.73	**83.86**	**78.04**	27.47	40.71	28.10
4× ensembled	84.77	**20.10**	83.09	71.74	**28.05**	40.53	**28.57**
4× ensembled, fine-tune only	84.05	19.57	82.36	71.55	27.55	40.22	28.09

Exploring the Limits of Transfer Learning with a Unified Text- to- Text Transformer
https://arxiv.org/abs/1910.10683