LLMs in Practice: Adapting Foundation Models

د. رياض سنبل

LLMs in Practice: Three Ways to Adapt

We can improve LLMs in three places: before, inside, or after the model.

1. PRE (Before the LLM)

RAG (Retrieval-Augmented Generation)

Give the model the right information before it answers.

Good for:

- ✓ Up-to-date info
- ✓ Private data
- ✓ Lower cost
- ✓ No training

Challenges:

- Retrieval quality
- Context limit
- Latency

2. IN (Inside the LLM)

Fine-Tuning (and PEFT)

Adapt the model by updating it with new data.

Good for:

- ✓ Domain specialization
- ✓ Better behavior
- ✓ Follow custom style

Challenges:

- Needs training data
- High compute cost
- Risk of forgetting

Methods (Examples):

Full Fine-Tuning • LoRA • QLoRA • Adapters • Prompt Tuning

3. POST (After the LLM)

Fact Checking / Verification

Check and improve the answer after it is generated.

Good for:

- ✓ More accurate outputs
- ✓ Fewer hallucinations
- ✓ Higher trust

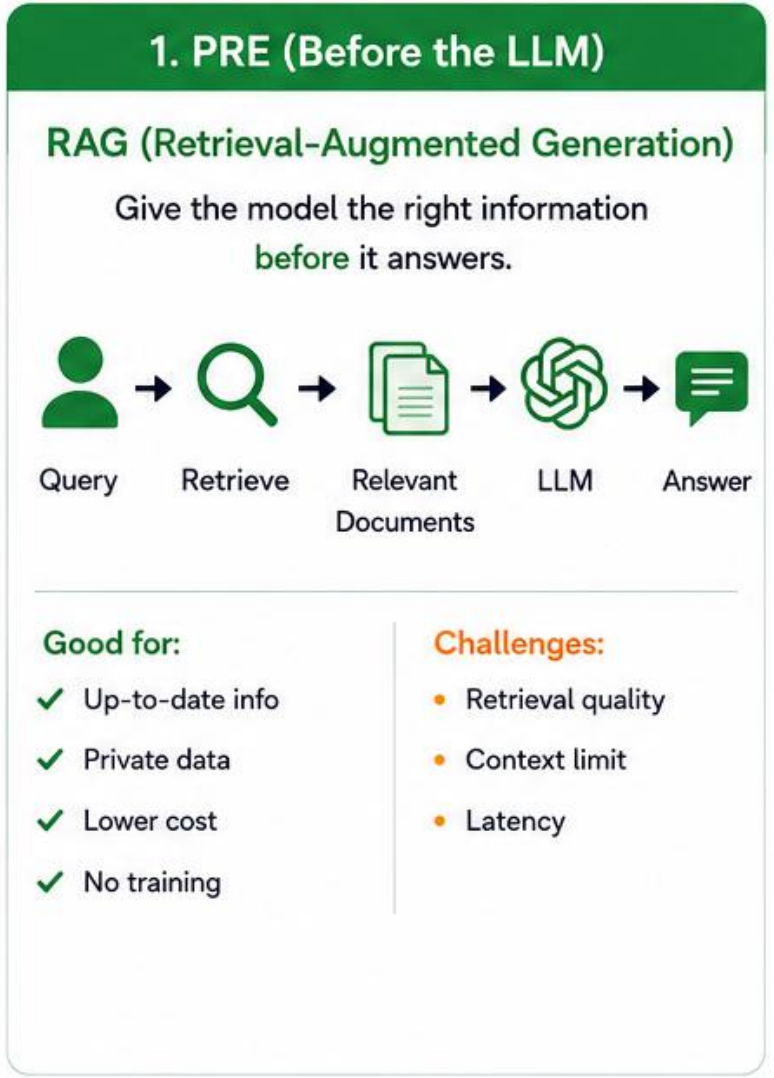
Challenges:

- Extra latency
- Need good verifier
- Not always perfect

Ways to Verify:

Rules • Retrieval • LLM-as-a-Judge
Self-Check • Multi-Agent

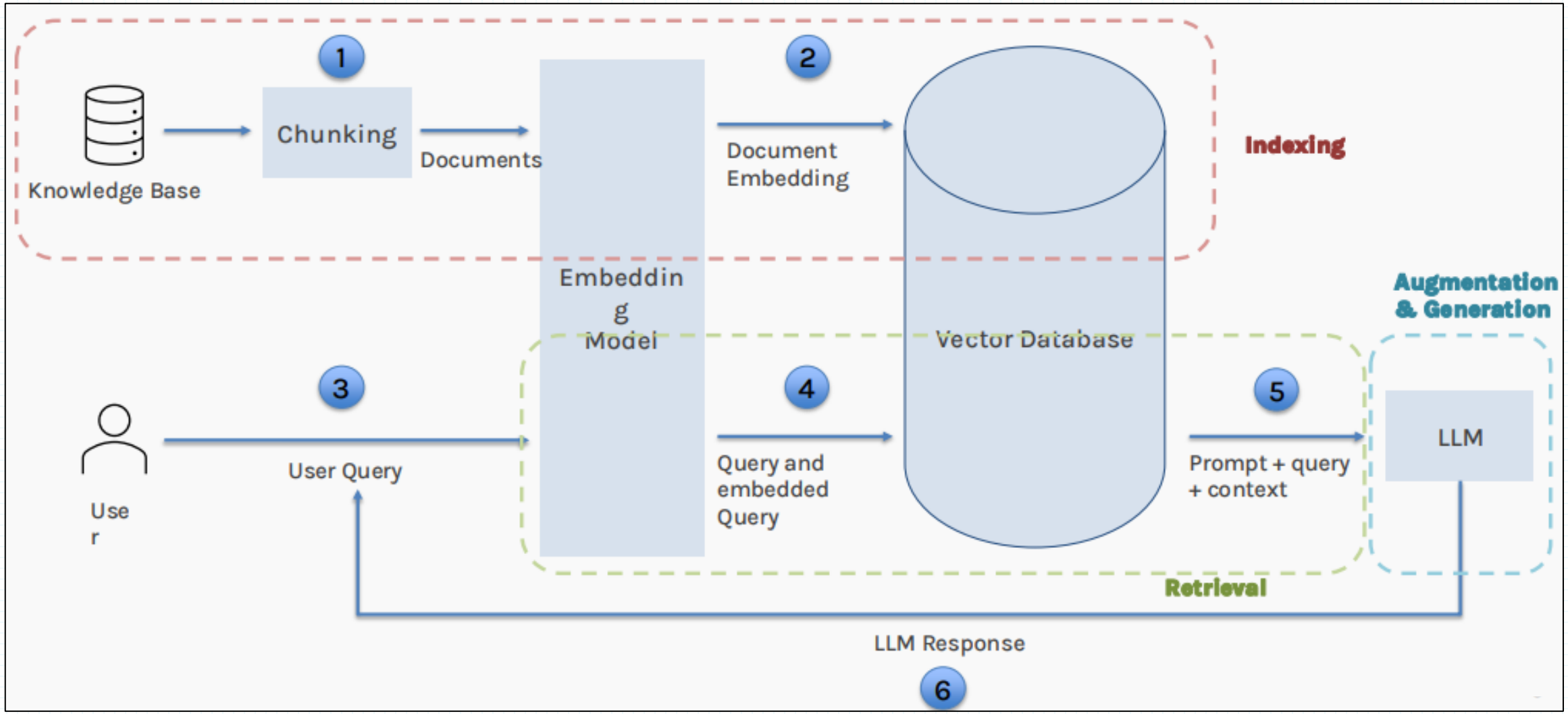
LLM in practice: Retrieval-Augmented-Generation (RAG)



Retrieval-Augmented-Generation (RAG)

- RAG stands for Retrieval-Augmented-Generation.
- It is technique that improves the performance of a LLM, especially for tasks that require accurate and detailed information.
- **Benefits of RAG:**
  - **Increased Accuracy:** By incorporating external knowledge.
  - **Contextual Relevance:** RAG allows LLMs to tailor responses to specific contexts and user needs.
  - **Up-to-Date Information:** RAG can access and utilize the latest information from external sources.
  - **Customization:** RAG enables organizations to integrate their specific knowledge bases and data into LLM applications.
  - **Improved Reliability:** Users can verify the sources of information used by RAG models, enhancing trust in their responses.

RAG


1
Chunking
Knowledge Base
Documents
2
Document Embedding
Indexing
Embeddin g Model
Vector Database
Augmentation & Generation
3
User Query
Use r
4
Query and embedded Query
5
Prompt + query + context
LLM
Retrieval
LLM Response
6

Pre-Retrieval Optimization (Improve the chunking process)

Let's take another example

Advancements in Transfer Learning for NLP

Abstract:
"Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks."

Methodology:
"We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency."

Results:
"The results indicate a 20% increase in accuracy for domain-specific tasks using our fine-tuning and domain adaptation techniques. We observed substantial performance gains compared to baseline models."

Now, if we do character splitting for chunks (chunk size=200), we get:

Chunk 1:
Recent techniques in transfer learning for NLP Abstract: Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and **dom**

Chunk 2:
ain adaptation methods. Our experiments demonstrate significant improvements in performance across various **NLP tasks. Methodology:** We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different do

....

Pre-Retrieval Optimization (Improve the chunking process)

Let's take another example

Advancements in Transfer Learning for NLP

Abstract:
"Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks."

Methodology:
"We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency."

Results:
"The results indicate a 20% increase in accuracy for domain-specific tasks using our fine-tuning and domain adaptation techniques. We observed substantial performance gains compared to baseline models."

Now, if we do character splitting for chunks (chunk size=200), we get:

Chunk 1:
Recent techniques in transfer learning for NLP Abstract: Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and **dom**

Chunk 2:
ain adaptation methods. Our experiments demonstrate significant improvements in performance across various **NLP tasks. Methodology:** We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different do

....

Pre-Retrieval Optimization (Improve the chunking process)

Better Approach!
Semantic chunking

Let's take another example

Advancements in Transfer Learning for NLP

Abstract:
"Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks."

Methodology:
"We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency."

Results:
"The results indicate a 20% increase in accuracy for domain-specific tasks using our fine-tuning and domain adaptation techniques. We observed substantial performance gains compared to baseline models."

The optimal way to do it would be:

Chunk 1:
Advancements in Transfer Learning for NLP

Chunk 2:
Abstract:
Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks.

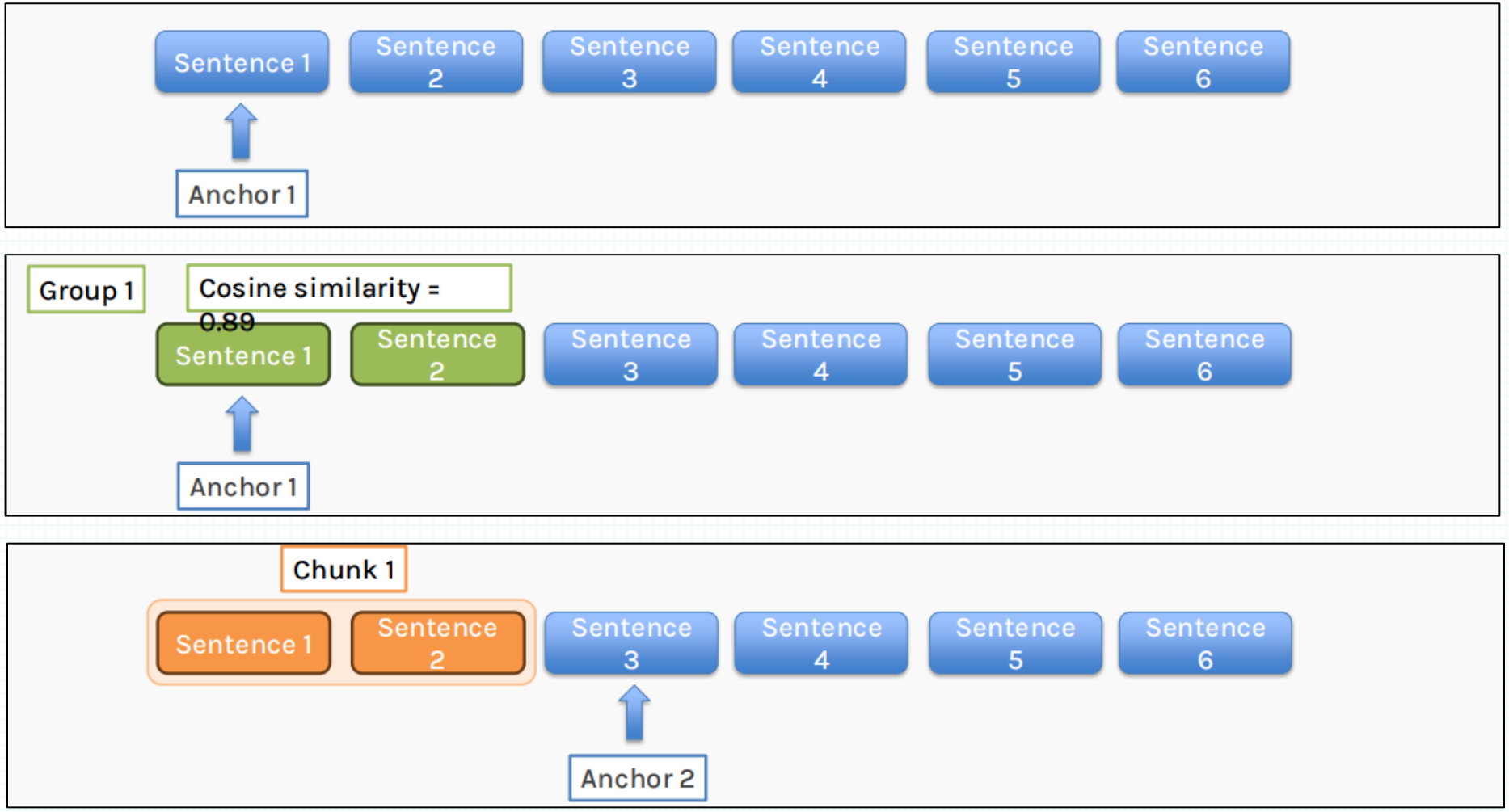
Chunk 3:
Methodology:
We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher

Pre-Retrieval Optimization (Improve the chunking process)

- **Semantic Chunking – Steps**
  - Splitting: We split the document to sentences using separators(.,?,!).
  - Grouping: Select anchor sentences and choose how many sentences to consider at either side of the anchor (window size).
  - Similarity Check: Calculate the distance between the group of sentences (e.g.: cosine similarity).
  - Chunking: Chunk together the similar sentences.

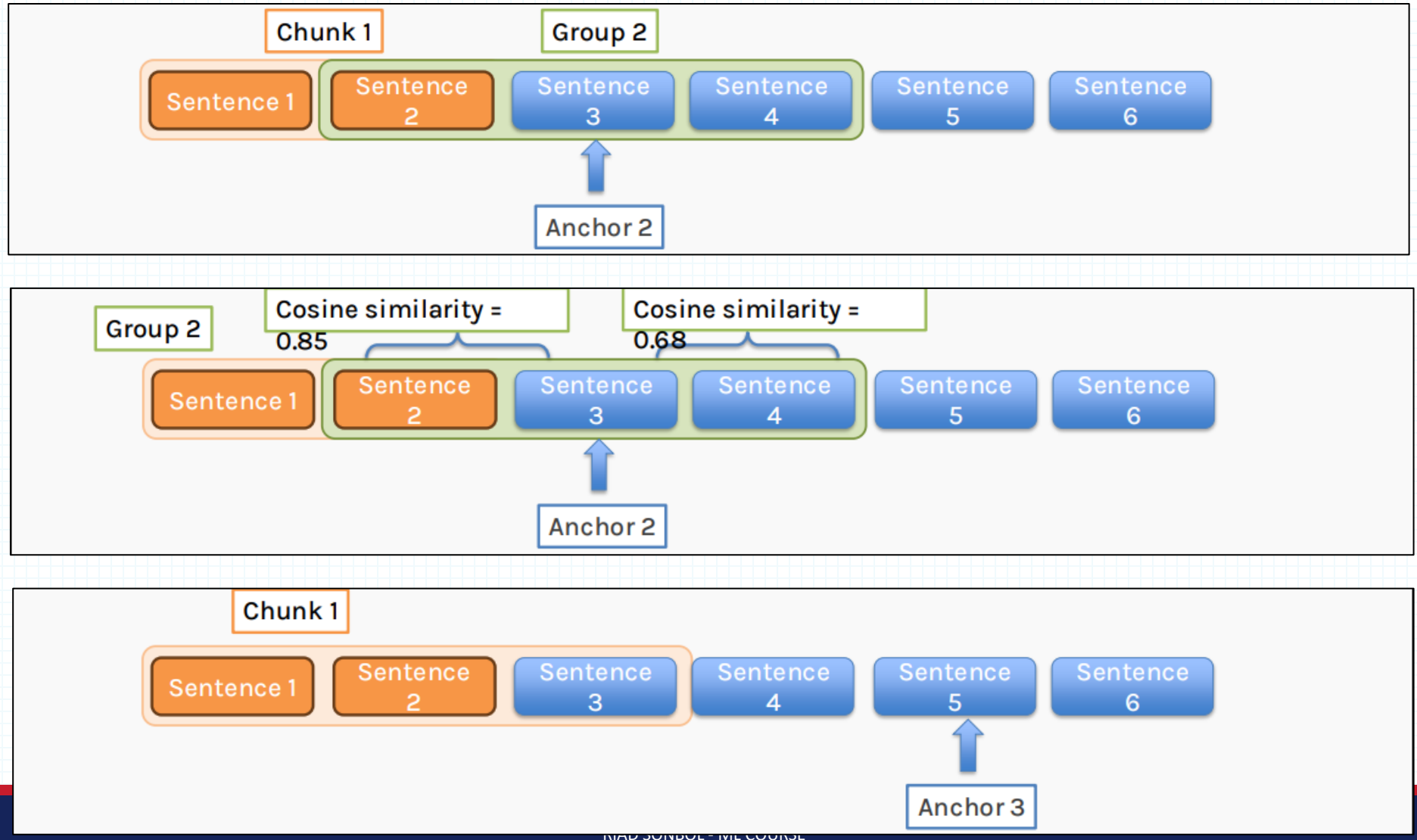
Pre-Retrieval Optimization (Improve the chunking process)

Semantic Chunking

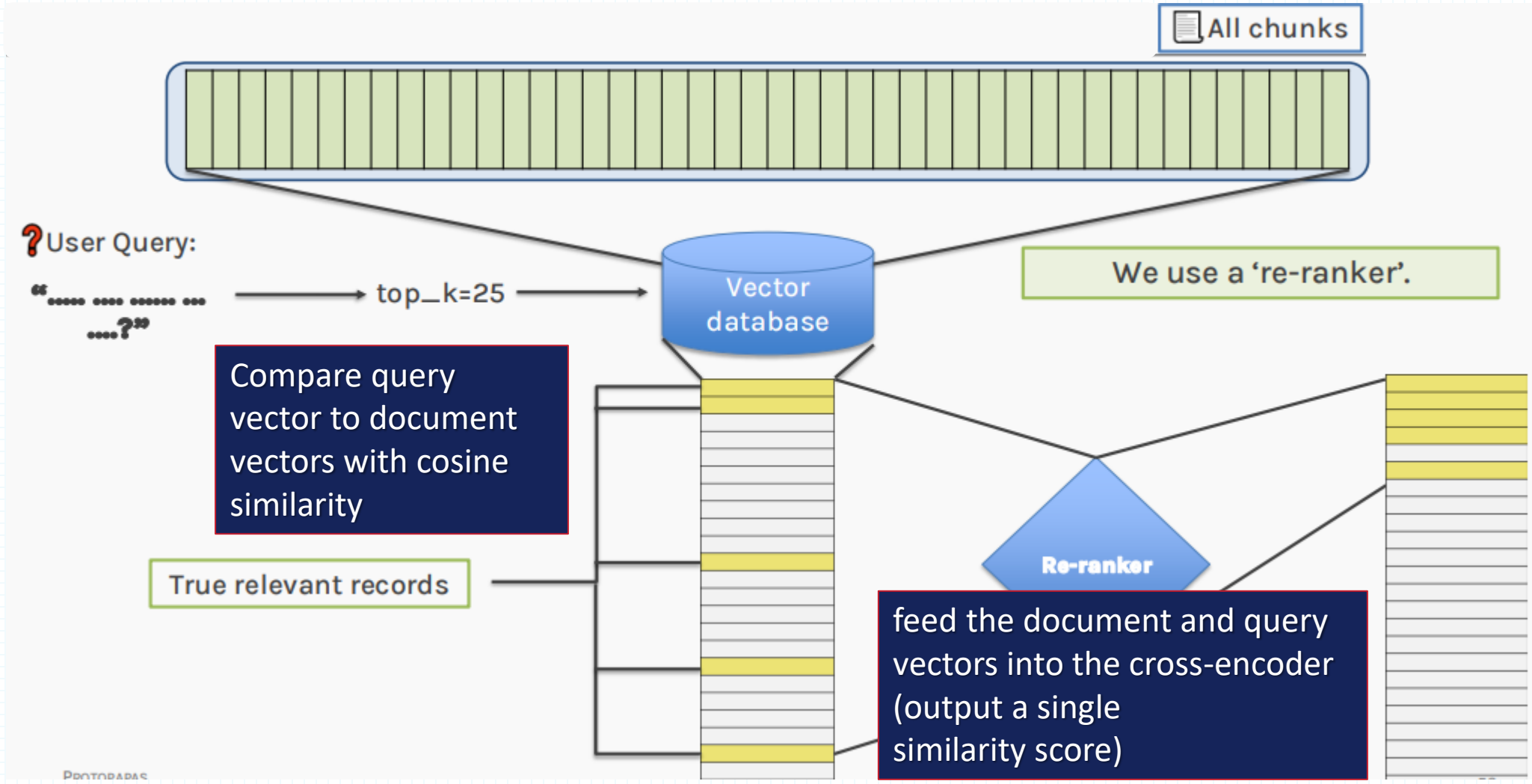


Pre-Retrieval Optimization (Improve the chunking process)

Semantic Chunking



Post-Retrieval – Re-ranking



Post-Retrieval – Re-ranking

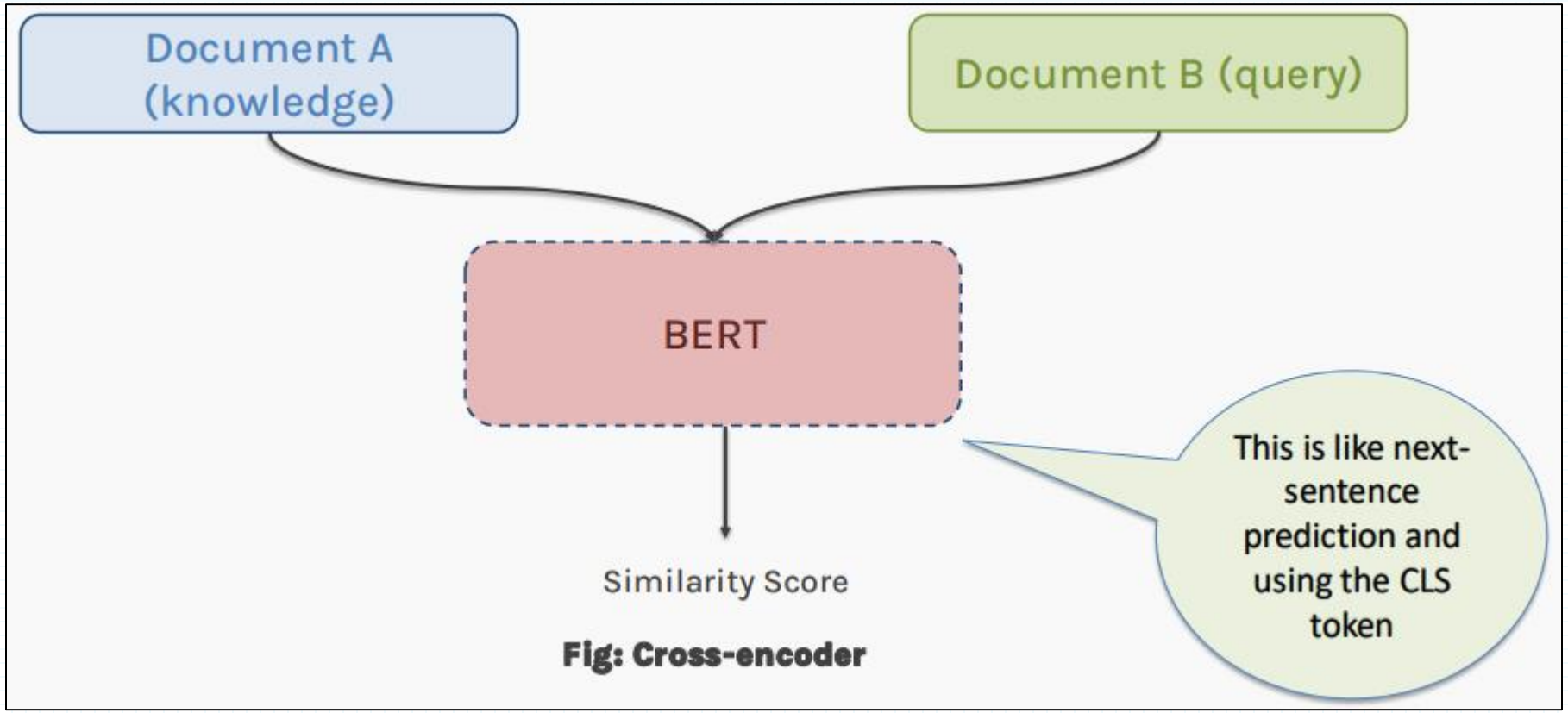
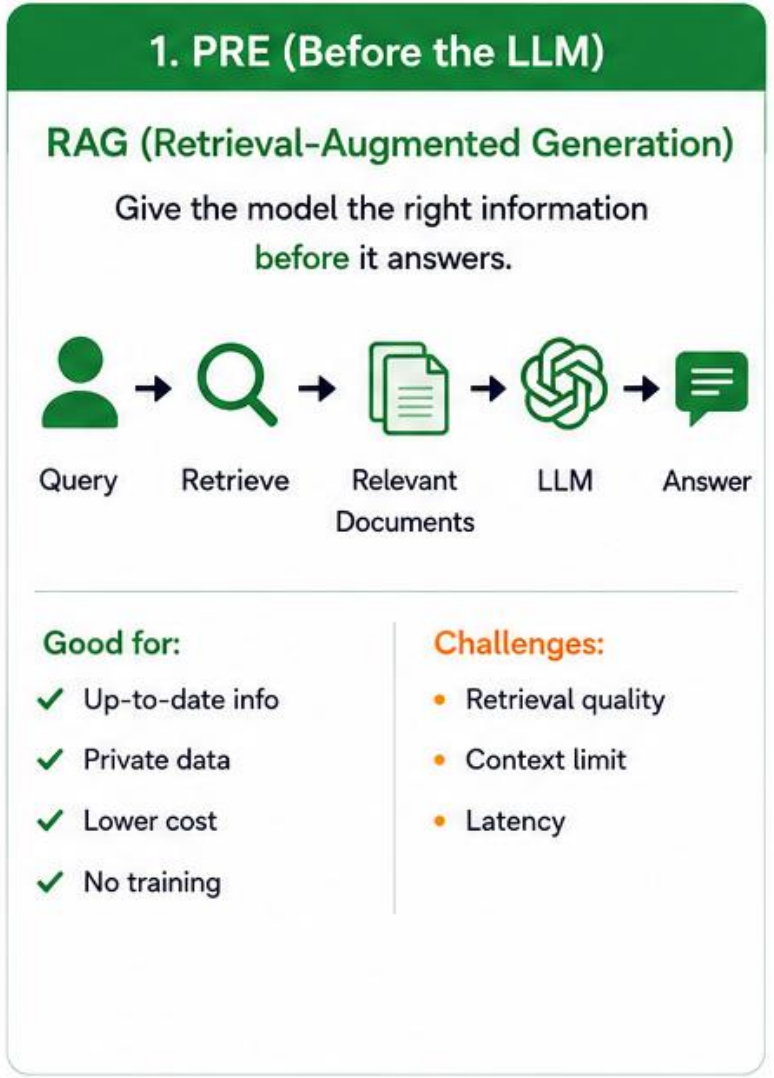


Fig: Cross-encoder

LLM in practice: Prompting



Zero/few-shot prompting

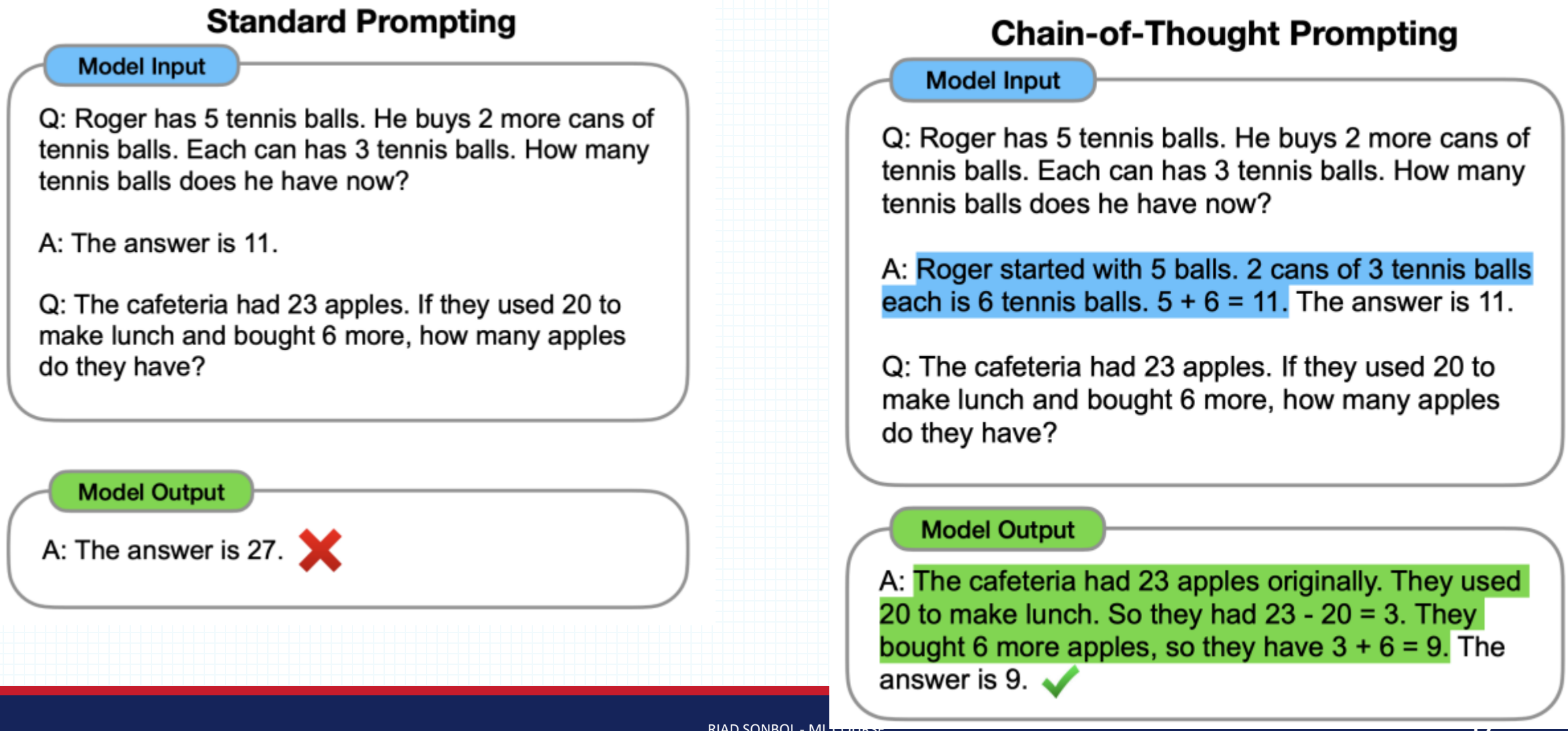
Limits of prompting for harder tasks?

- Some tasks seem too hard for even large LMs to learn through prompting alone. Especially tasks involving richer, multi-step reasoning.

```
19583 + 29534 = 49117
98394 + 49384 = 147778
29382 + 12347 = 41729
93847 + 39299 = ?
```

Solution: Chain-of-thought prompting

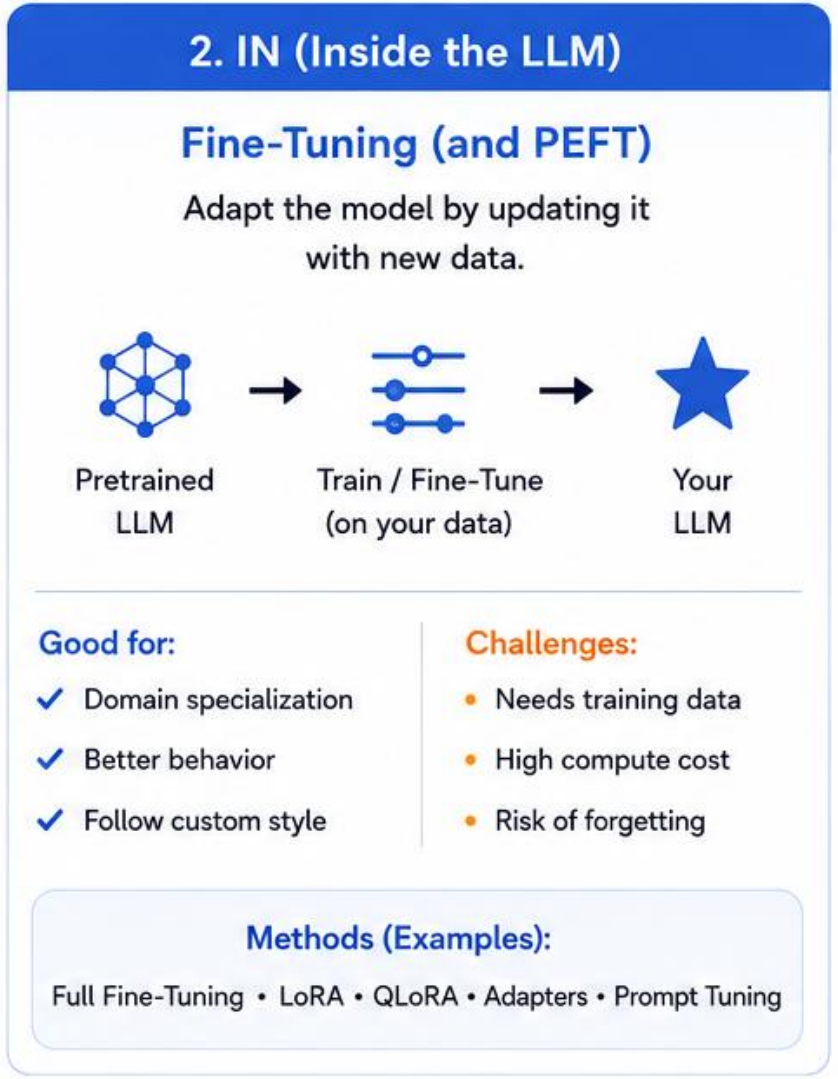
Chain-of-thought prompting



Downside of prompt–based learning

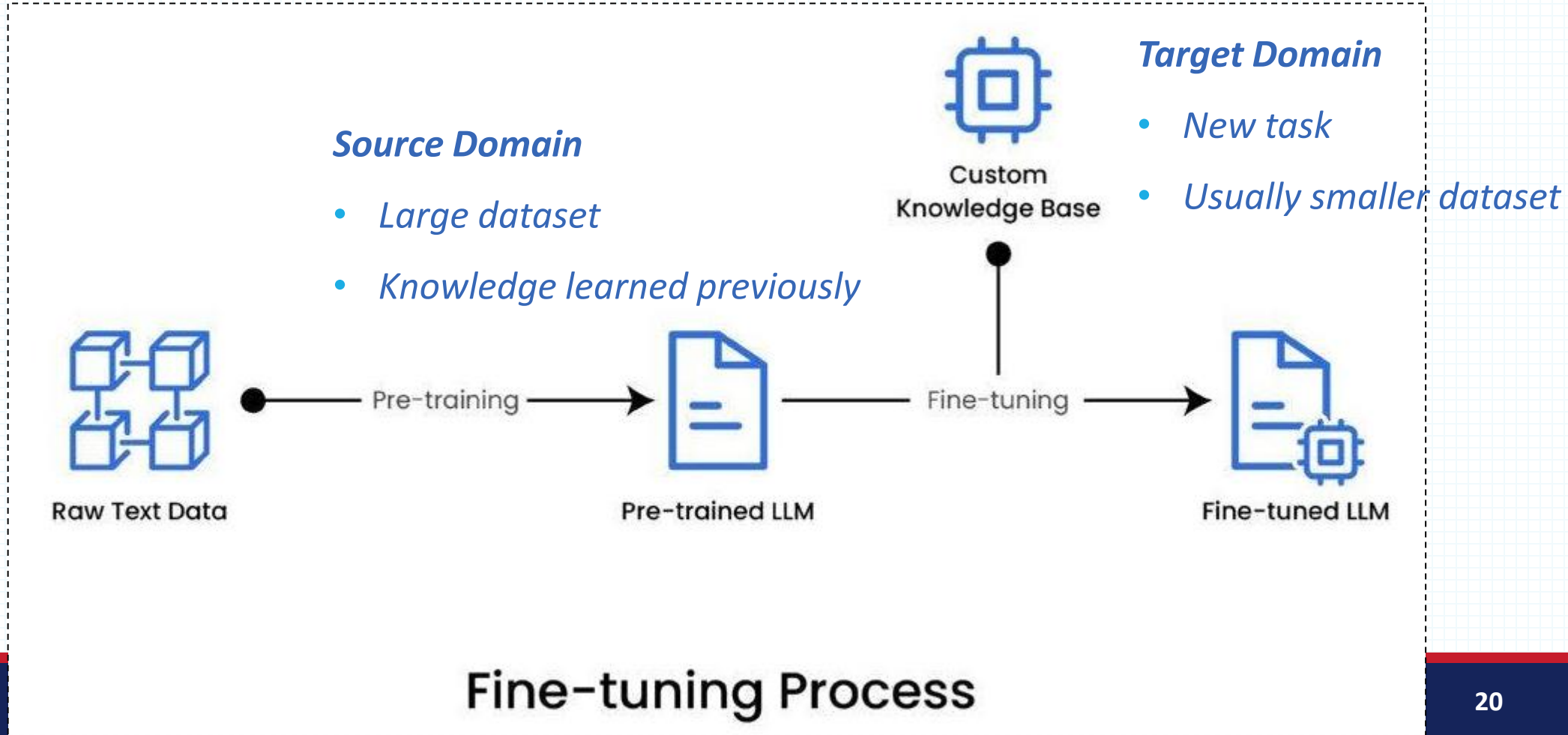
- Inefficiency: The prompt needs to be processed every time the model makes a prediction.
- Poor performance: Prompting generally performs worse than fine-tuning
- Sensitivity to the wording of the prompt
- Lack of clarity regarding what the model learns from the prompt.

LLM in practice: Fine-Tuning



Transfer Learning

- Transfer Learning is the process of leveraging knowledge learned from one task/domain to improve performance on another related task/domain.



What is LLM fine-tuning?

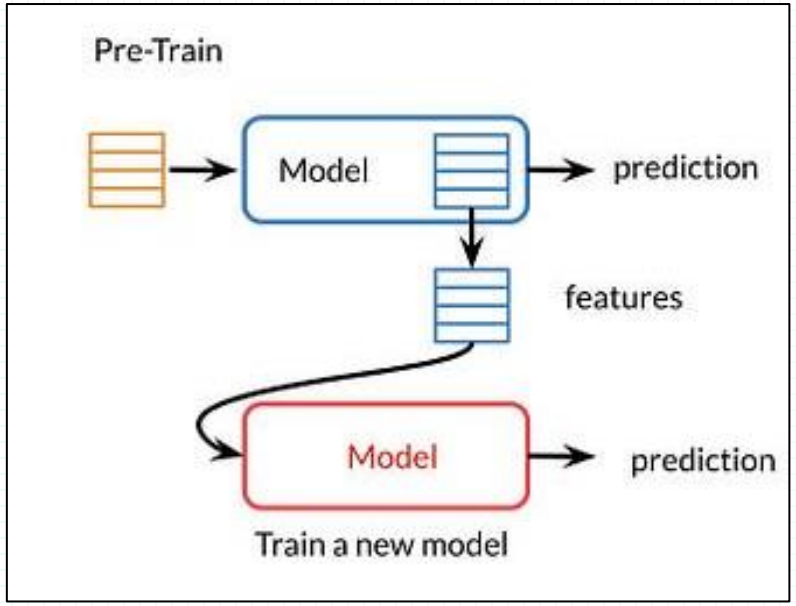
- Fine-tuning is the process of adjusting the parameters of a pre-trained large language model to a **specific task or domain**.
- Although pre-trained language models like GPT possess **vast language knowledge**, **they lack specialization** in specific areas.
- By exposing the model to task-specific examples during fine-tuning, the model can acquire a **deeper understanding of the nuances of the domain**

Why is LLM fine-tuning important?

- **Customization**: Every domain or task has its own unique language patterns, terminologies, and contextual nuances.
- **Data compliance**: In many industries, such as healthcare, finance, and law, strict regulations govern the use and handling of sensitive information. Organizations can ensure their model adheres to data compliance standards by fine-tuning the LLM on proprietary or regulated data.
- **Limited labeled data:** Fine-tuning allows organizations to leverage pre-existing labeled data more effectively by adapting a pre-trained LLM to the available labeled dataset, maximizing its utility and performance.

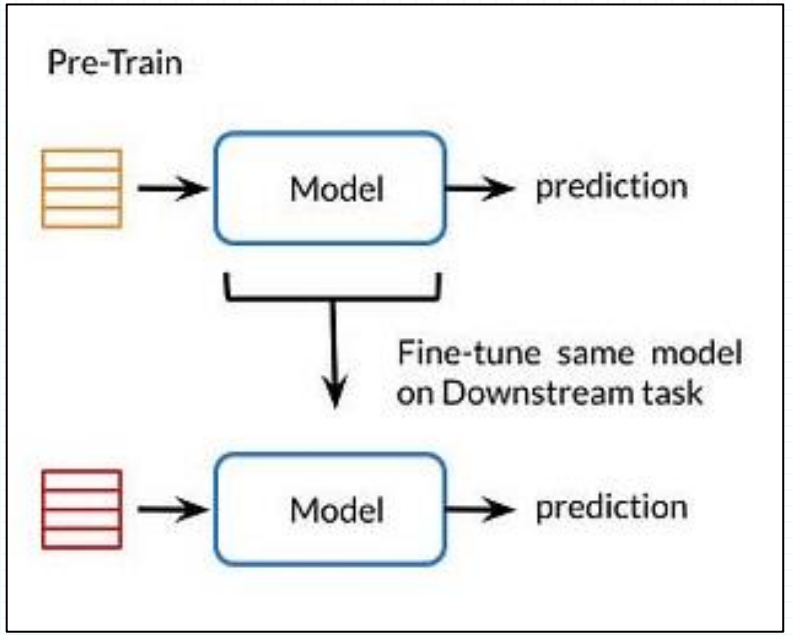
Types of LLM fine-tuning

- **Feature extraction:**
  - It involves freezing the weights of a pre-trained model's layers and using it as a feature extractor
  - Used when a smaller dataset is available, and the target domain is closely aligned with the original domain of the pre-trained model.
  - Faster training, requires less computational resources, and can be more effective when the new dataset is small.



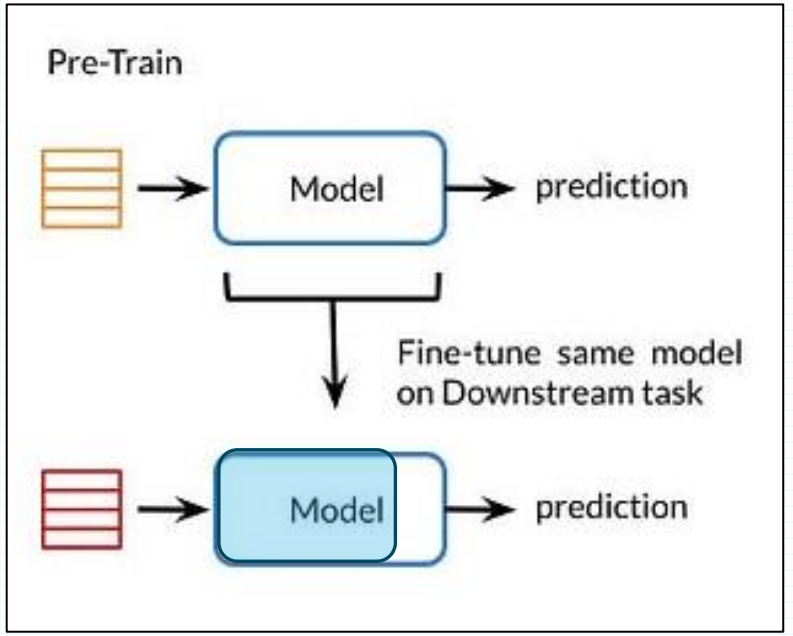
Types of LLM fine-tuning

- **Full fine-tuning**
  - involves unfreezing some or all of the pre-trained layers and retraining them along with new layers.
  - Used when a larger dataset is available and more flexibility is needed to adapt the pre-trained model to the specific task
  - Allowing the model to learn new features that are specific to the target task
  - Can achieve higher accuracy, better adaptability to the new task, and allows the model to learn more task-specific features while preserving the general knowledge from the original dataset.



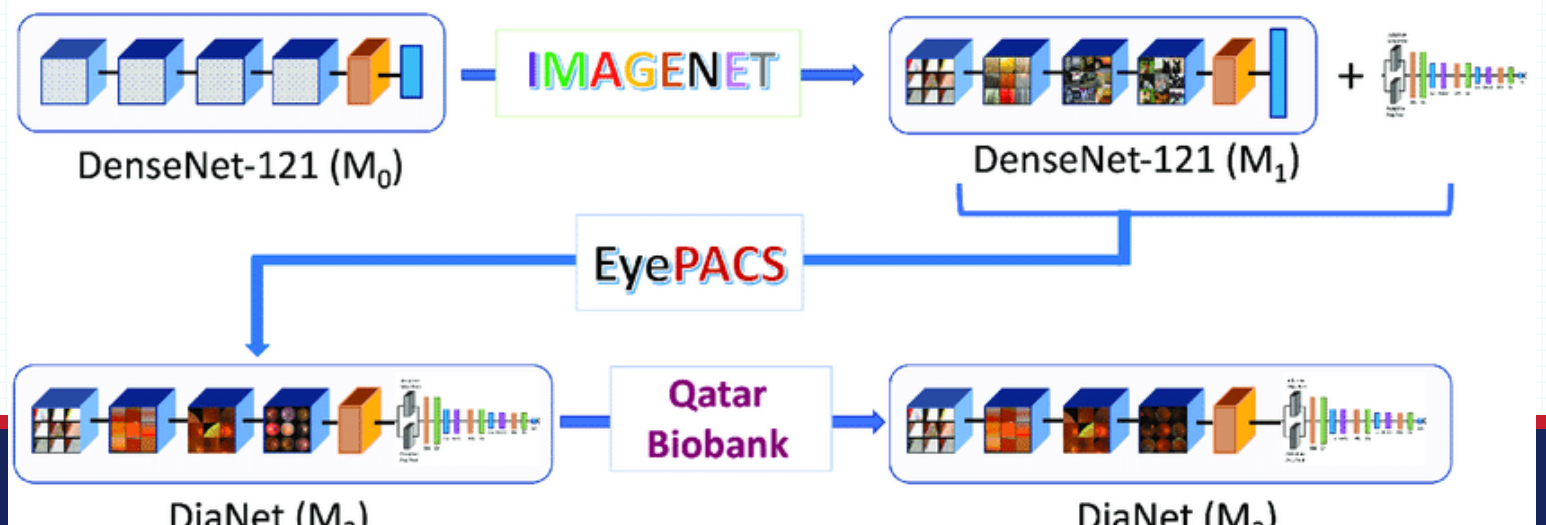
Types of LLM fine-tuning

- **partial fine-tuning**
  - Partial Fine-Tuning is a transfer learning approach where only a subset of the pretrained model's parameters is updated, while the remaining parameters stay frozen.



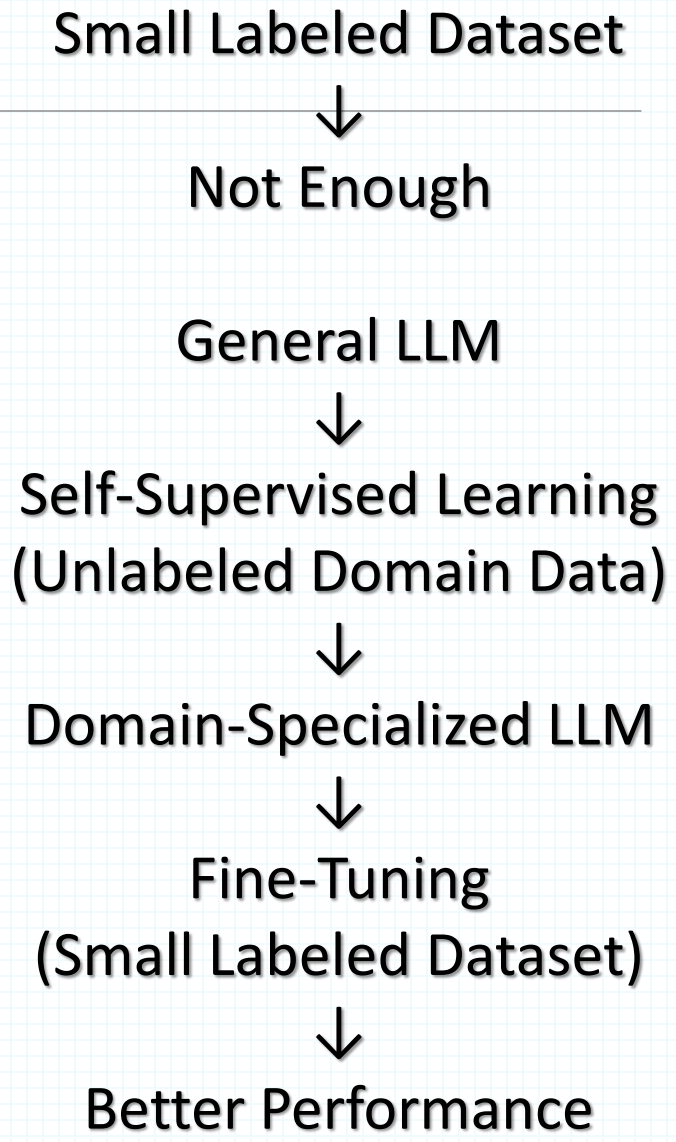
Types of LLM fine-tuning

- **Multi-stage fine-tuning**
  - A technique where a model is trained in multiple sequential stages, each stage building upon the previous one.
  - The process might involve:
    - Pre-training
    - Initial Fine-Tuning: smaller dataset relevant to a specific domain e.g. medical text.
    - Domain-Specific Fine-Tuning: further fine-tuned on a dataset specifically related to a narrow sub-domain, such as clinical trial reports.



Self-Supervised Learning as a Bridge to Effective Fine-Tuning

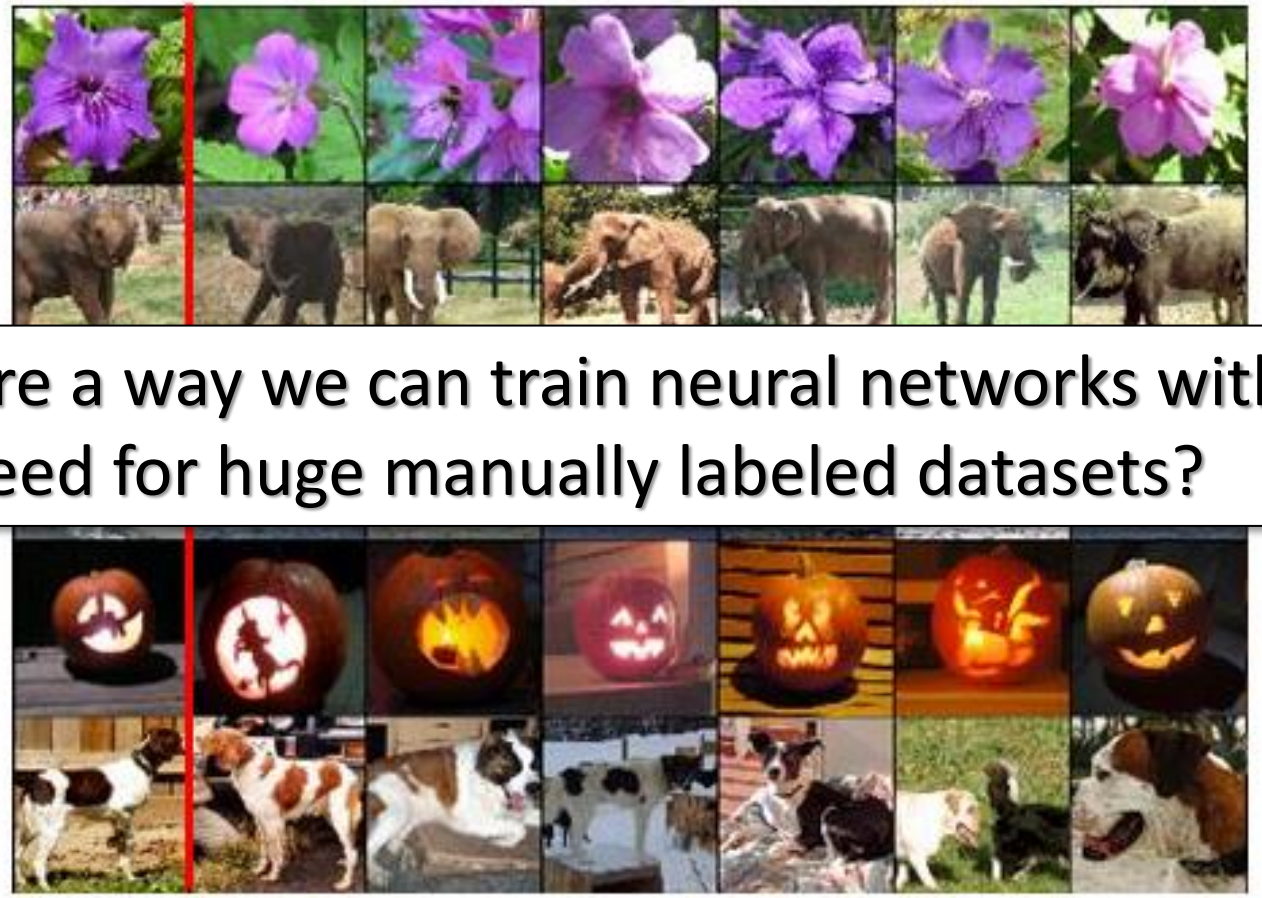
- Self-Supervised Learning serves as a bridge between a general foundation model and effective fine-tuning, particularly when labeled data is scarce.
- When labeled data is scarce, self-supervised learning can be used before fine-tuning to adapt a foundation model to the target domain using large amounts of unlabeled data, leading to better downstream performance.



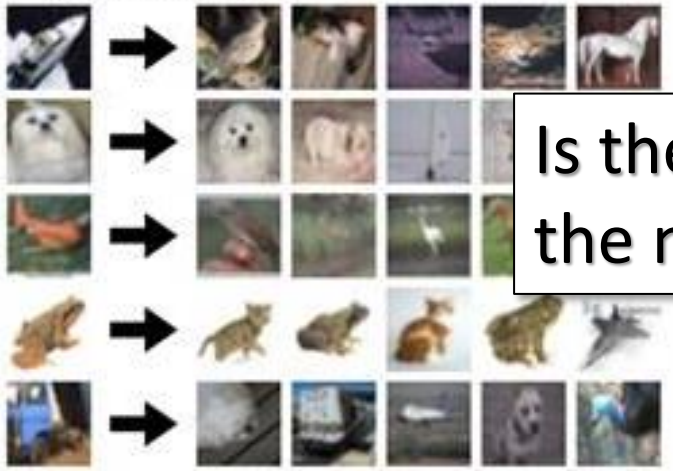
Self-Supervised Learning as a Bridge to Effective Fine-Tuning

Test image

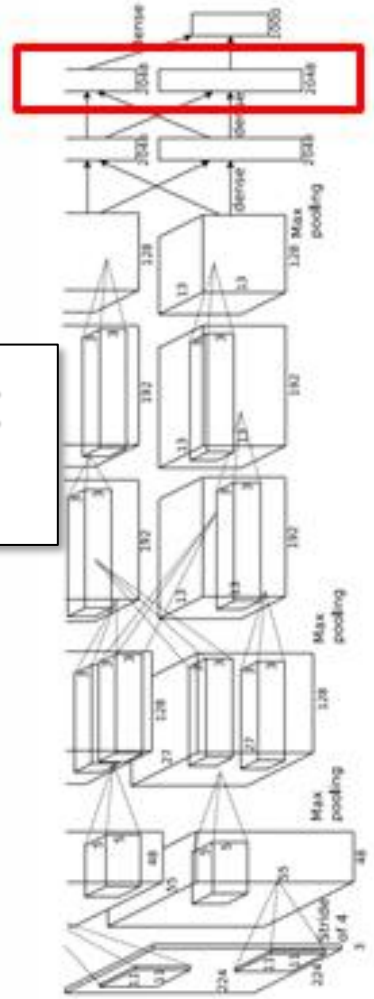
L2 Nearest neighbors in feature space



Recall: Nearest neighbors in pixel space



4096-dim vector



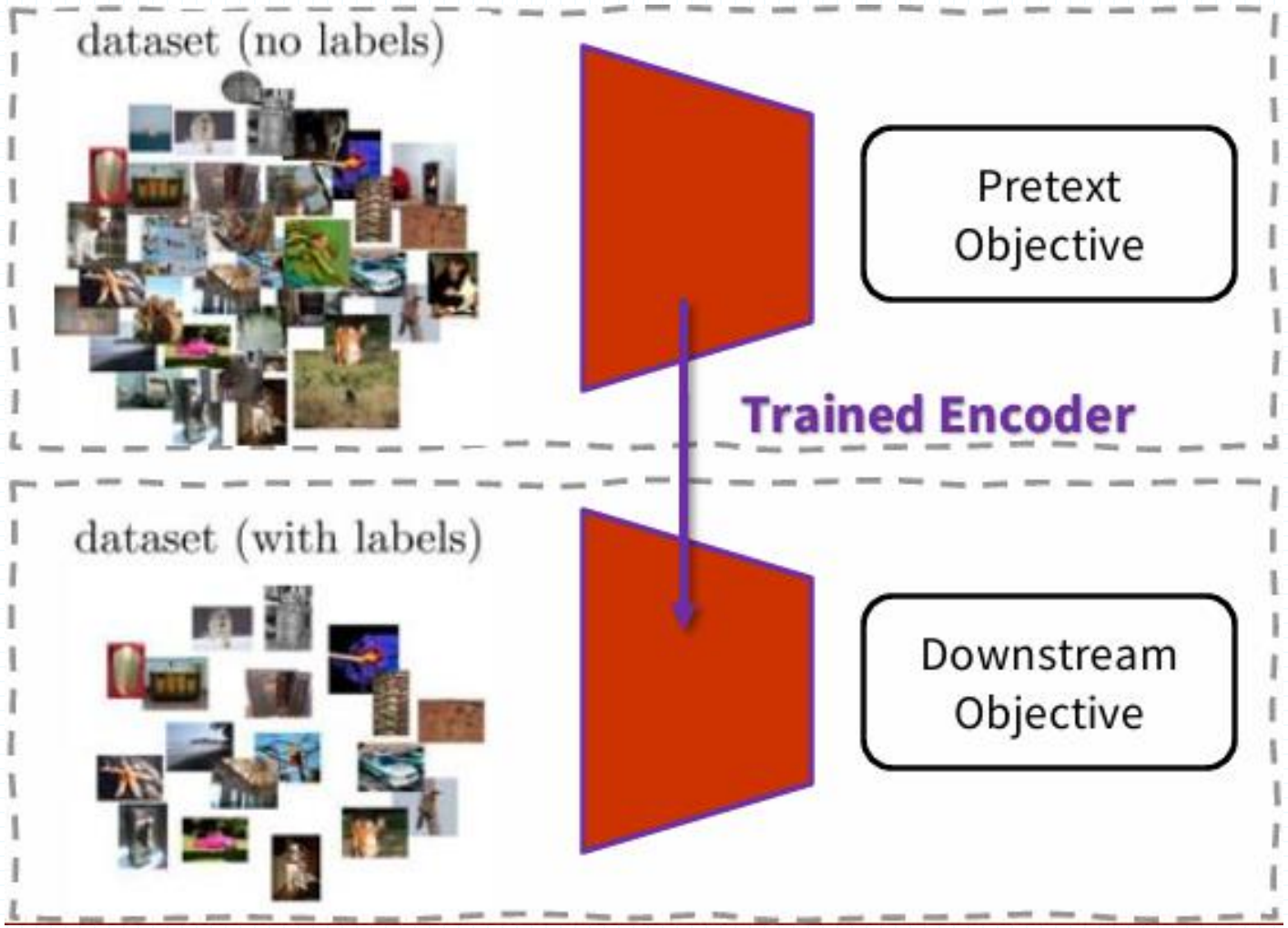
Is there a way we can train neural networks without the need for huge manually labeled datasets?

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

Self-Supervised Learning as a Bridge to Effective Fine-Tuning

Pretext Task

- Define a task based on the data itself
- No manual annotation- Could be considered an unsupervised task;



Self-supervised pretext tasks

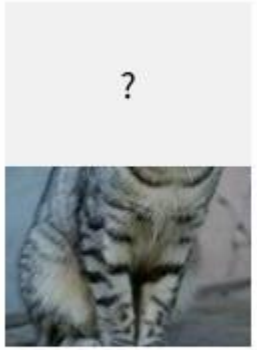


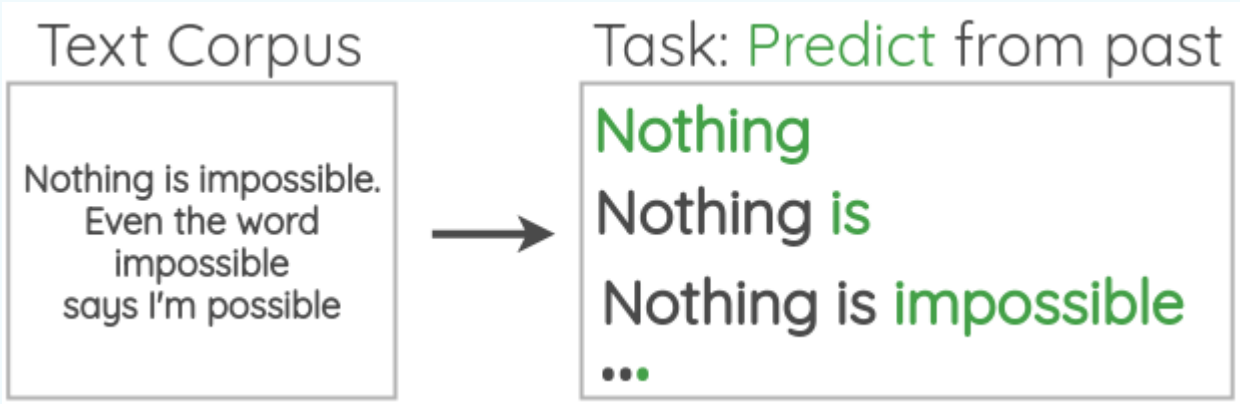
image completion

rotation prediction

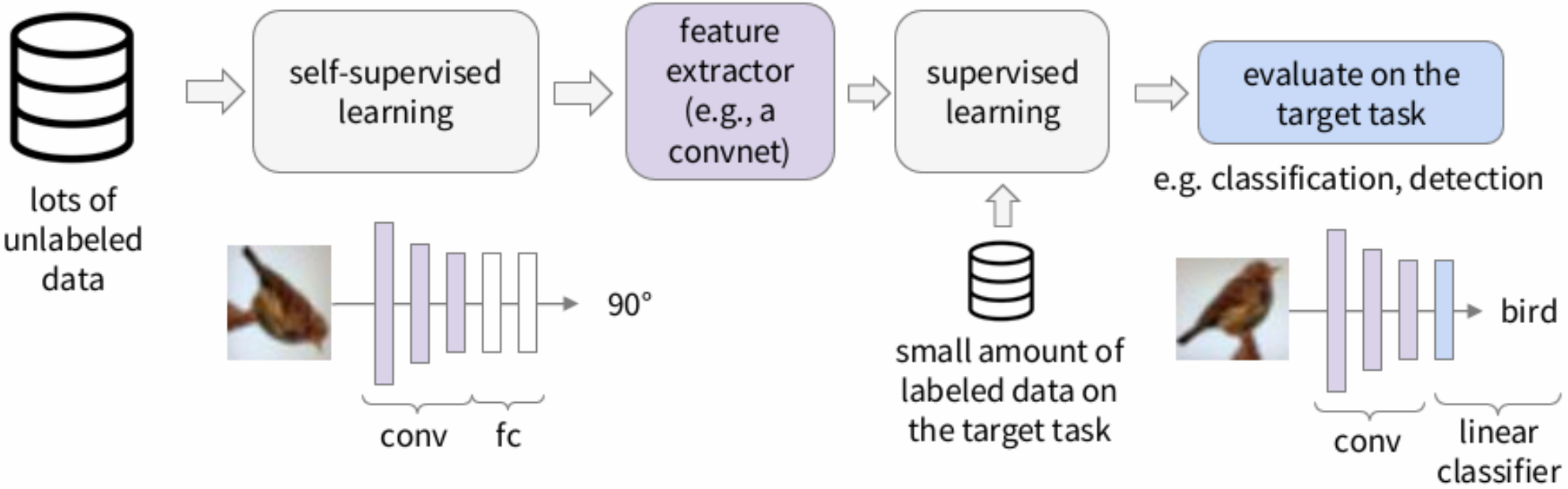
"jigsaw puzzle"

colorization

Masked Autoencoder



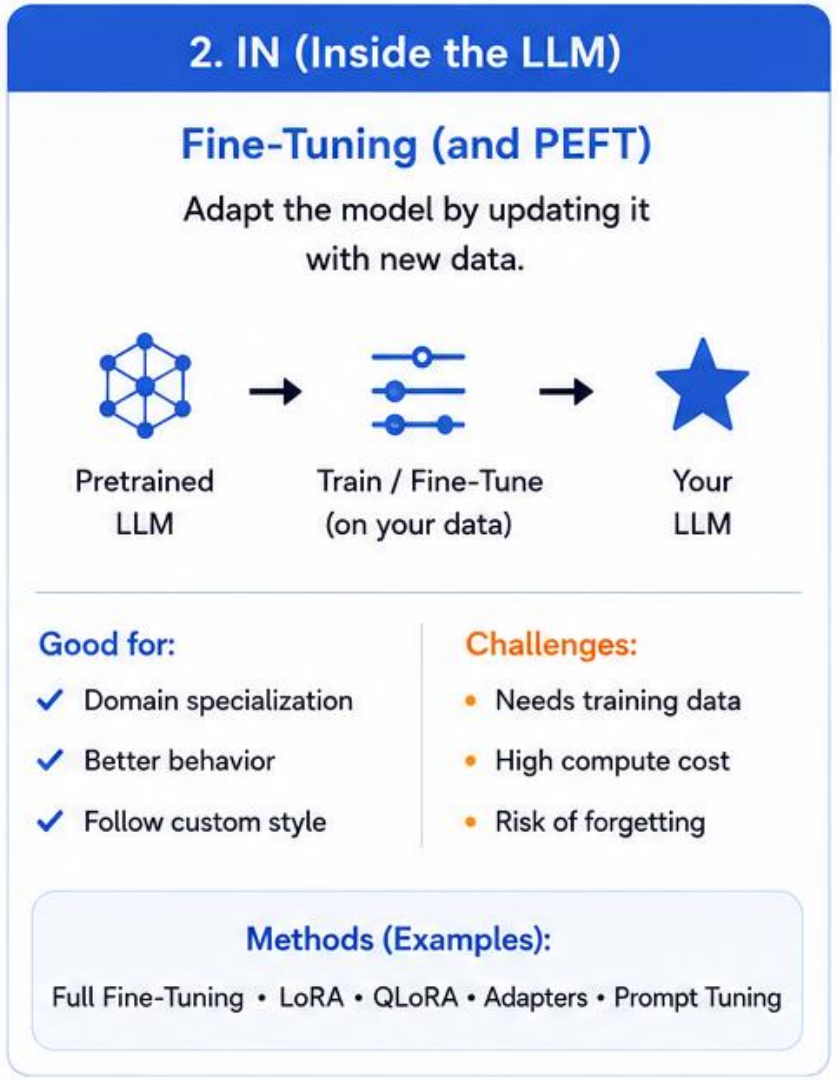
Self-supervised learning method



1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

2. Attach a shallow network on the feature extractor; train the shallow network on the target task with small amount of labeled data

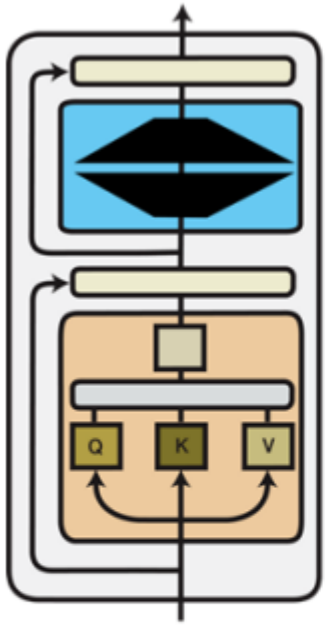
LLM in practice: parameter efficient fine-tuning (PEFT)



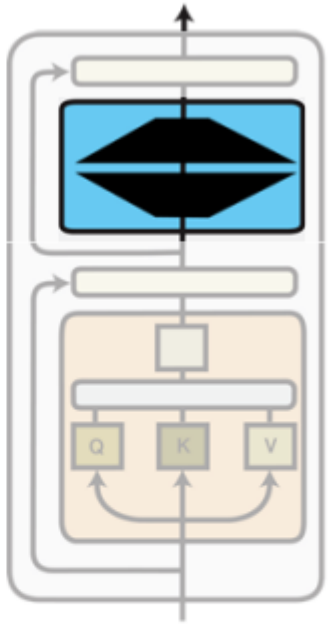
Parameter efficient fine-tuning (PEFT)

- Why fine-tuning only some parameters?
- Fine-tuning all parameters is impractical with large models
- State-of-the-art models are massively overparameterized

→ Parameter-efficient finetuning matches performance of full fine-tuning



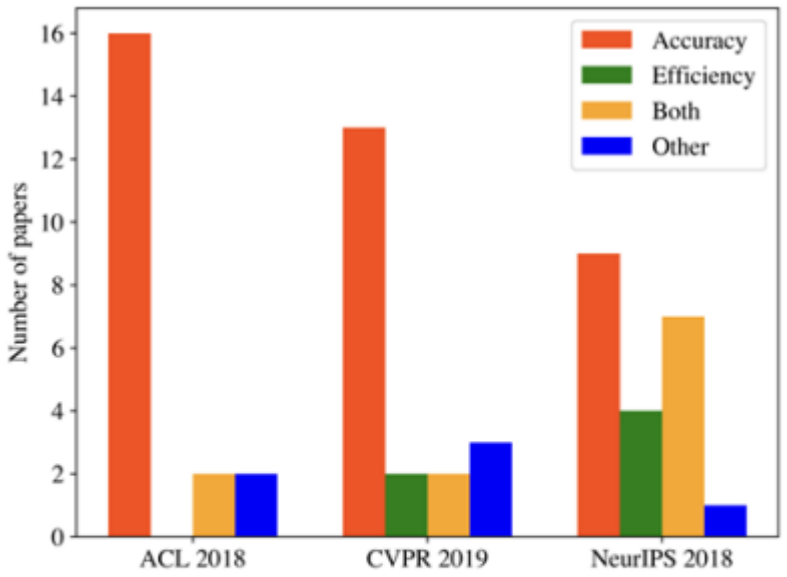
Full Fine-tuning
Update **all model parameters**



Parameter-efficient Fine-tuning
Update a **small subset** of model parameters

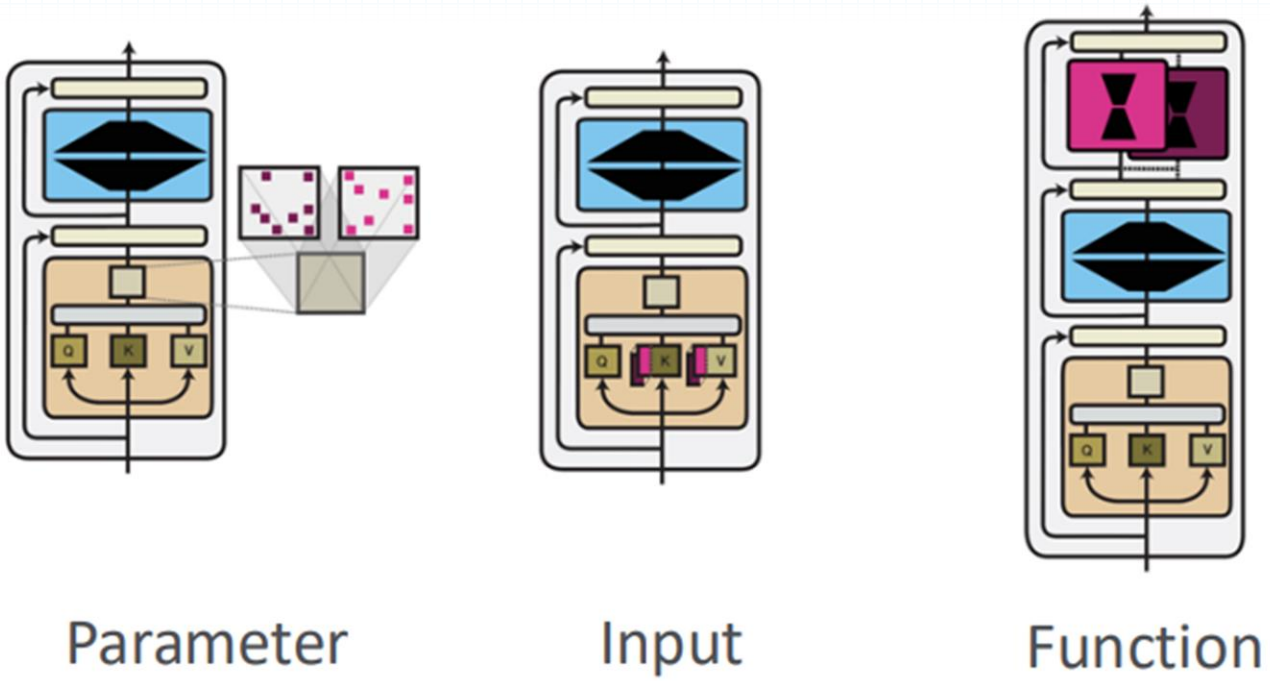
Why do we need efficient adaptation?

- Emphasis on accuracy over efficiency in current AI paradigm
- Hidden environmental costs of training (and fine tuning) LLMs
- As costs of training go up, AI development becomes concentrated in well-funded organizations, especially in industry

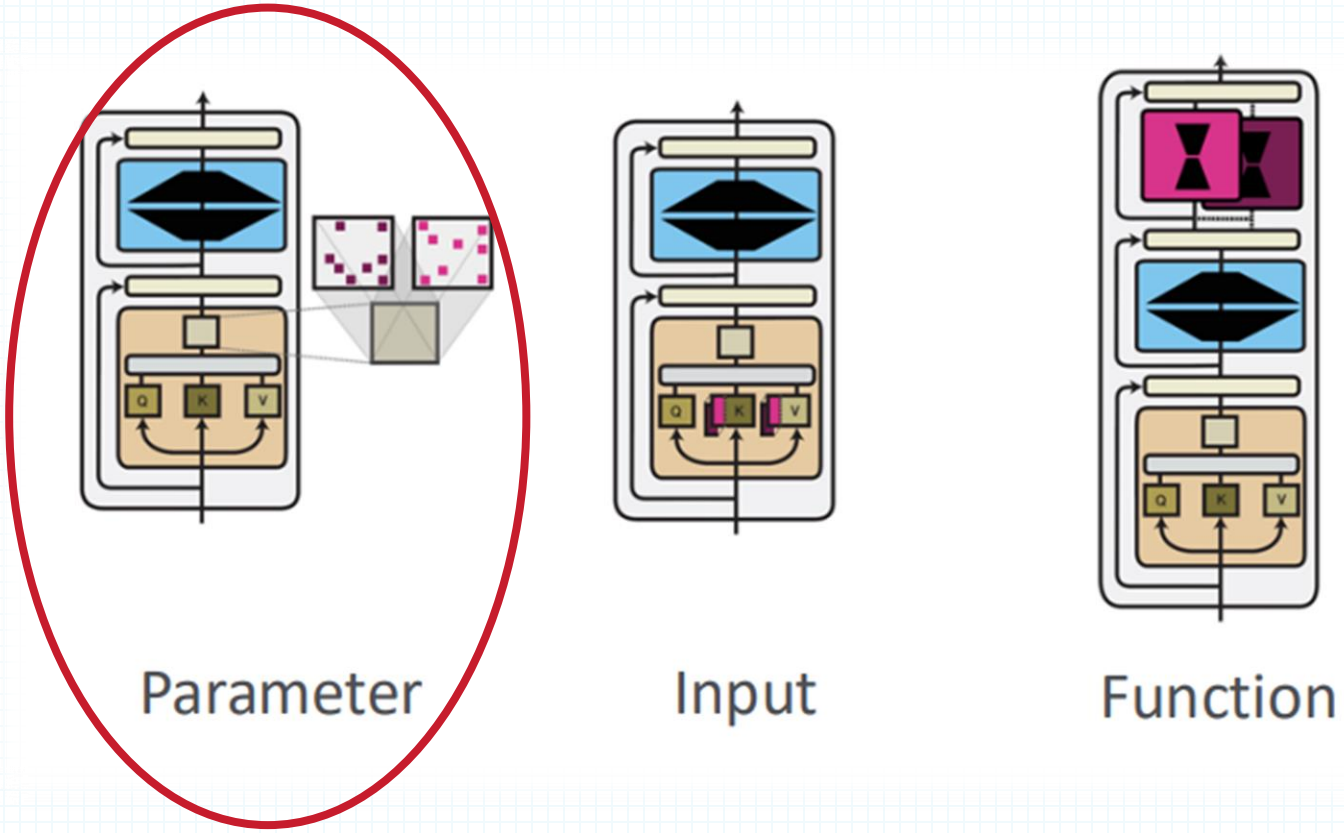


AI papers tend to target accuracy rather than efficiency. The figure shows the proportion of papers that target accuracy, efficiency, both or other from a sample of 60 papers from top AI conferences (Green AI)

Different perspectives to think about PEFT

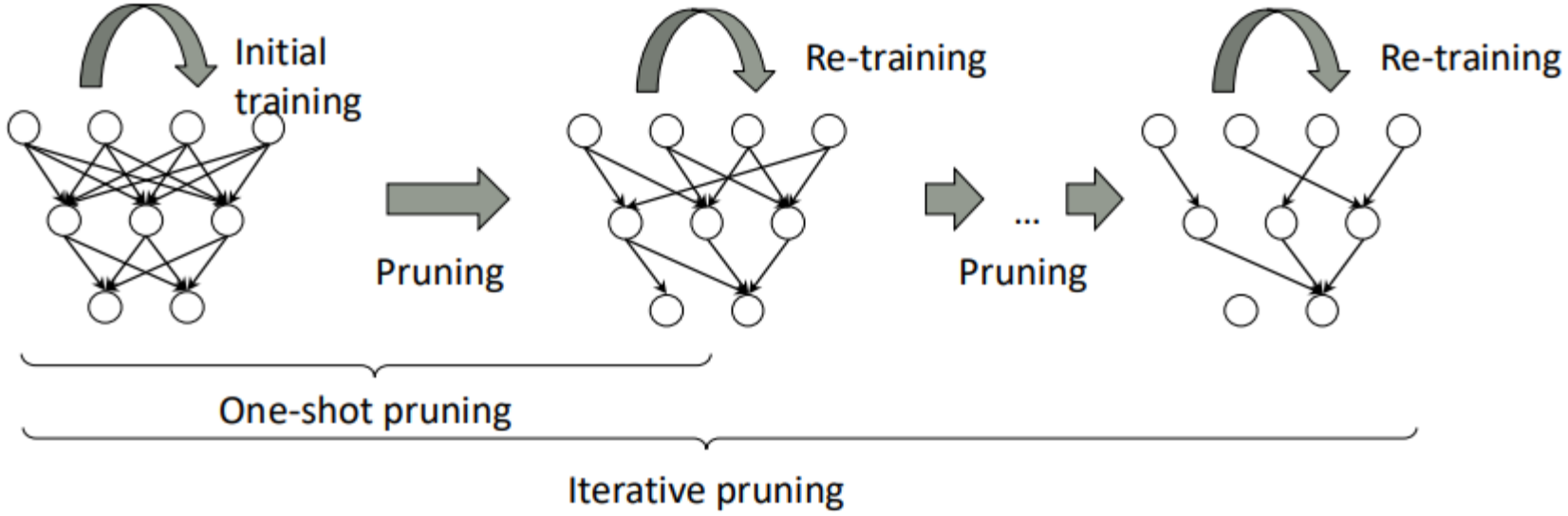


A Parameter Perspective of Adaptation



A Parameter Perspective of Adaptation

A Parameter Perspective of Adaptation - Sparse subnetworks (pruning)

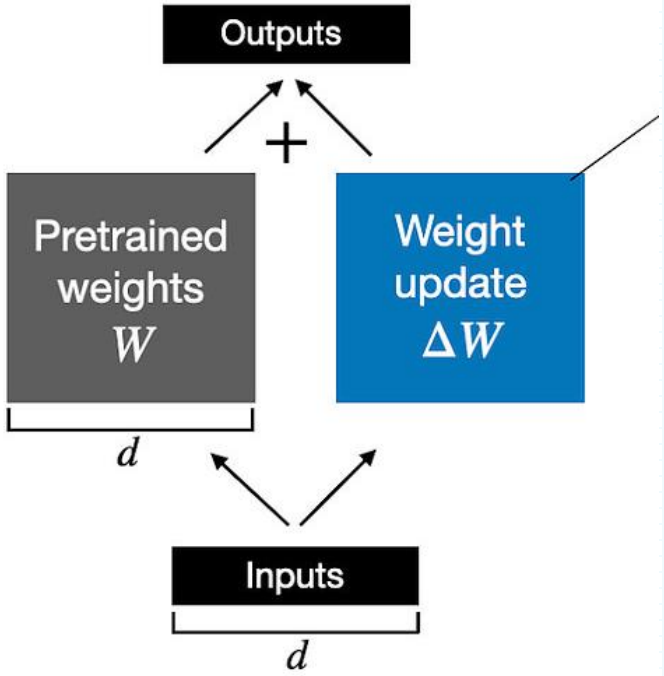


Sparse subnetworks (pruning) adapt a pretrained model by identifying and updating only a small subset of the most important parameters while keeping the majority of weights fixed. This reduces memory and computational requirements while preserving much of the model's original performance, making adaptation more efficient.

A Parameter Perspective of Adaptation

A Parameter Perspective of Adaptation - Low-rank Composition

Weight update in regular finetuning



Weight used in inference:

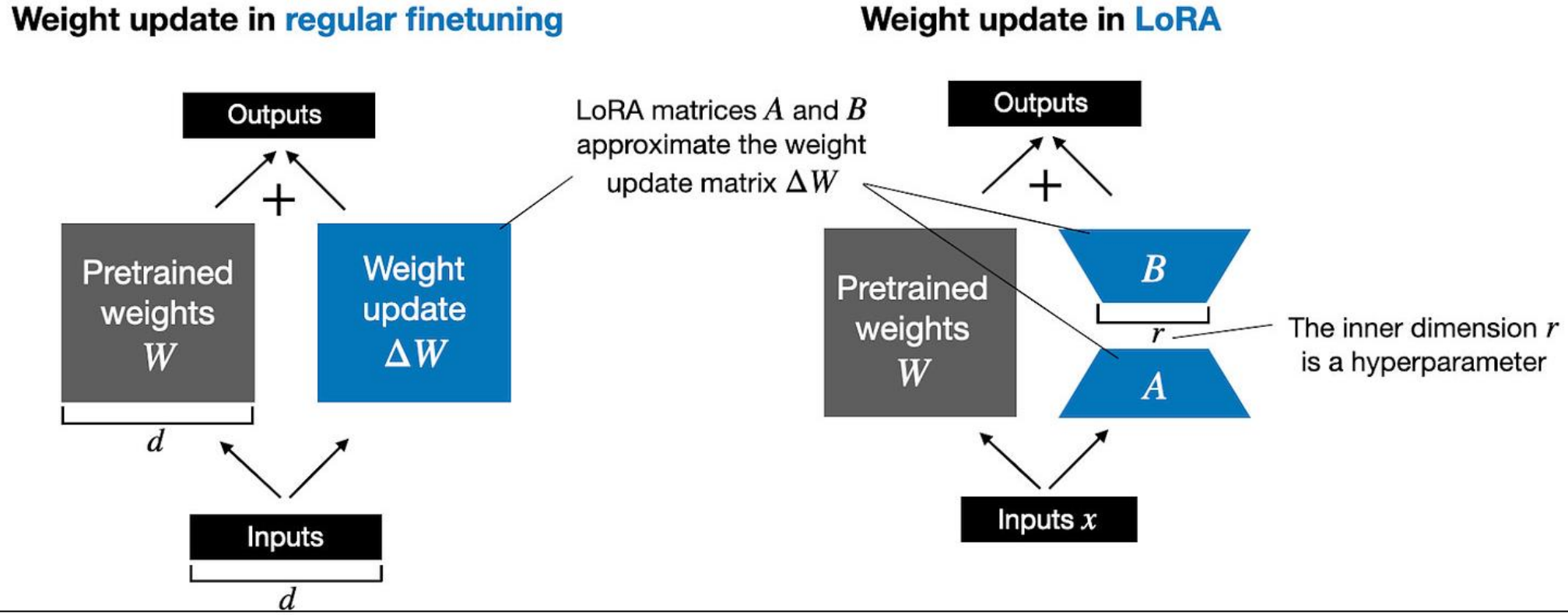
$$W' = W_0 + \Delta W = W_0 + BA$$

Pretrained (frozen) — Low-rank update — Two small trainable matrices

- W_0 : pretrained weight (frozen)
- $A \in \mathbb{R}^{r \times d_{in}}$, $B \in \mathbb{R}^{d_{out} \times r}$: trainable
- $r \ll \min(d_{in}, d_{out})$: rank (typically 4–64)
- $\Delta W = BA$ has rank at most r (low-rank update)

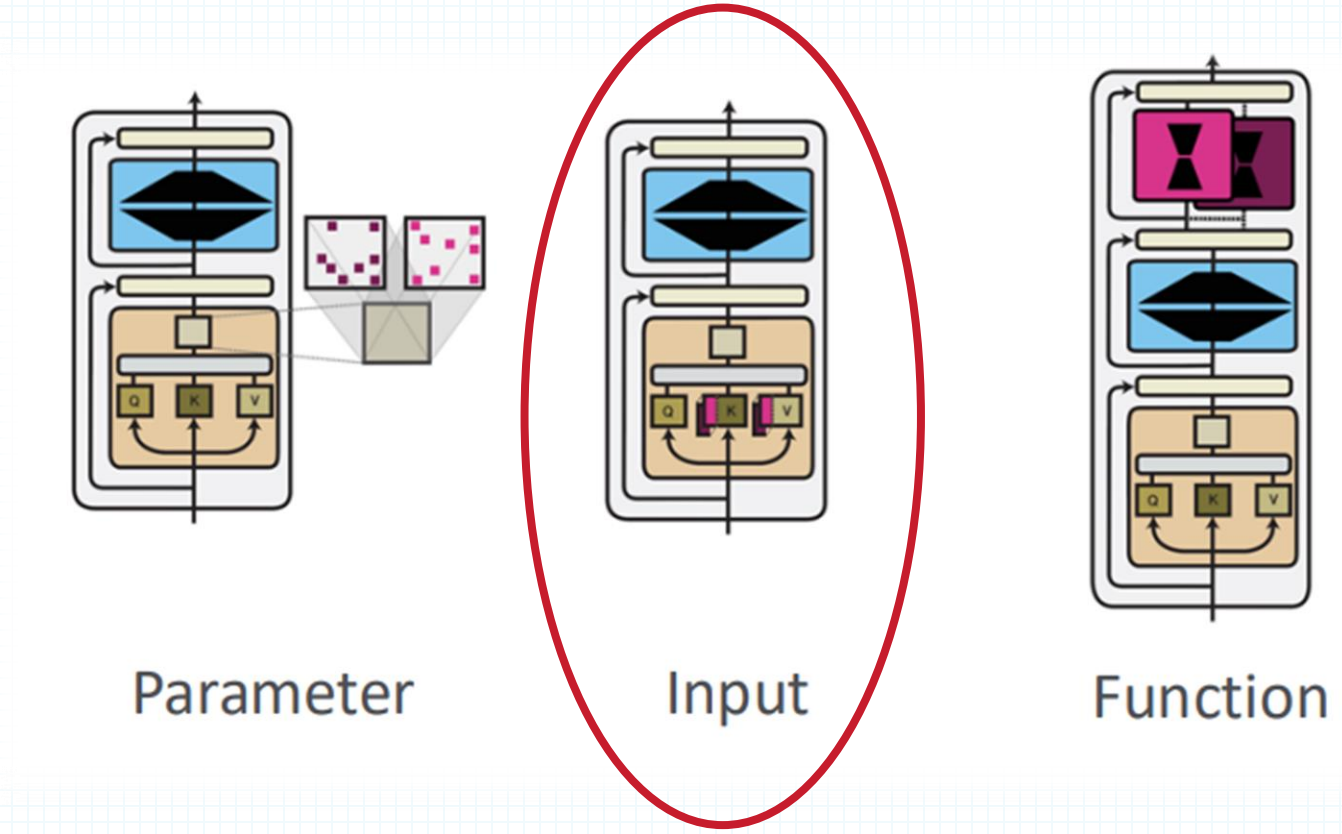
A Parameter Perspective of Adaptation

A Parameter Perspective of Adaptation - Low-rank Composition



Low-rank composition adapts a pretrained model by learning a small set of low-rank matrices that approximate changes to the original weight matrices, instead of updating all parameters. This significantly reduces the number of trainable parameters and memory requirements while maintaining strong performance, as used in methods such as **LoRA (Low-Rank Adaptation)**.

An input perspective of adaptation



An input perspective of adaptation

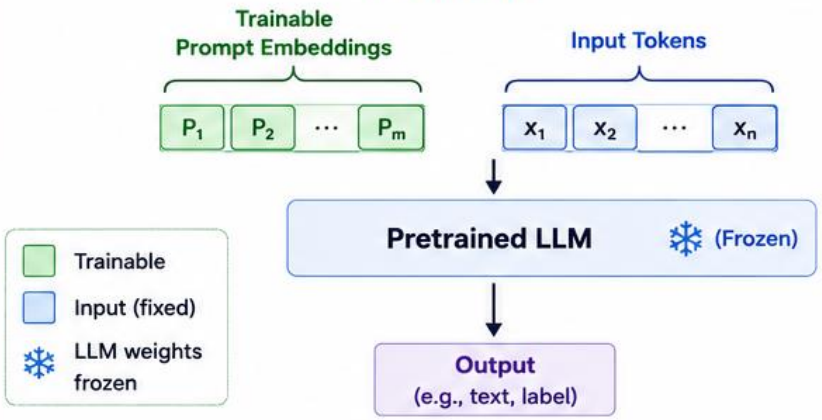
Prefix-Tuning and Prompt-Tuning

- **Prompt-Tuning:** You give the student instructions **before** the exam starts.
- **Prefix-Tuning:** You can whisper guidance **during every step** of solving the exam.
- Prompt-Tuning learns trainable prompts at the input layer, whereas Prefix-Tuning learns trainable prefixes that influence every Transformer layer throughout the model.
- Prompt-Tuning prepends trainable embeddings to the input tokens, whereas Prefix-Tuning prepends trainable Key/Value vectors to the attention mechanism of every Transformer layer.a

PROMPT-TUNING

Learn a small set of virtual prompt tokens added only at the input layer.

How it works



Key Points

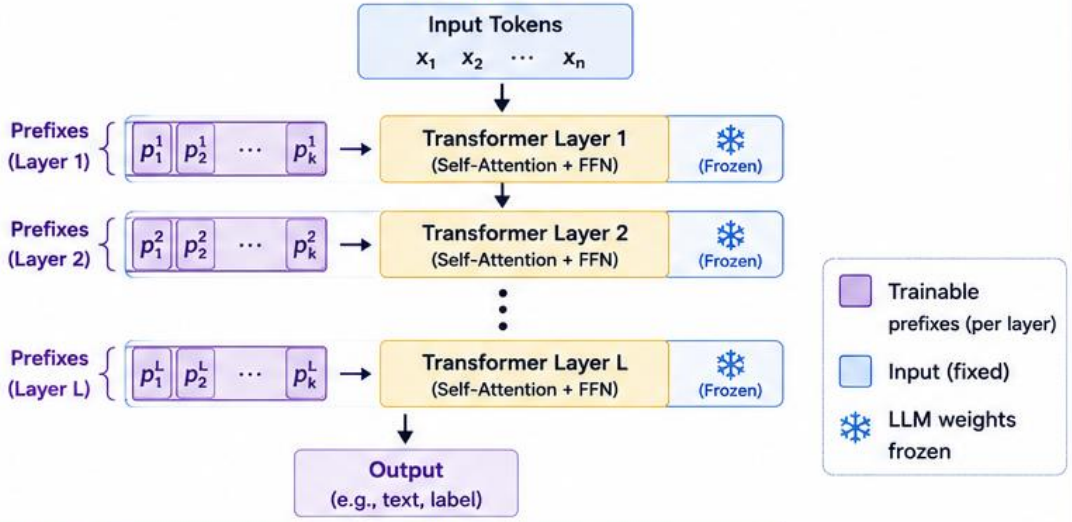
- Only the prompt embeddings are trained.
- Added once at the input.
- Shallow influence (affects the model only through the input).
- Very simple and extremely parameter efficient.
- May be less powerful on complex tasks.

VS.

PREFIX-TUNING

Learn trainable prefix vectors injected into every Transformer layer (into Key and Value of attention).

How it works



Key Points

- Trainable prefixes are added to every Transformer layer.
- Deep influence (affects the model throughout all layers).
- Usually achieves better performance than prompt-tuning.
- More parameters and slightly more complex.

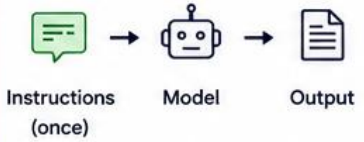
QUICK COMPARISON

Aspect	Prompt-Tuning	Prefix-Tuning
Where added?	Input layer only	Every Transformer layer (in attention)
Trainable parameters	Very few	More (but still small)
Model weights	Frozen	Frozen
Influence on model	Shallow	Deeper
Performance	Lower (on complex tasks)	Usually higher
Complexity	Simple	More complex
Best for	Small datasets, quick adaptation	Stronger adaptation, complex tasks

INTUITION

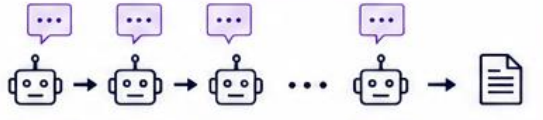
Prompt-Tuning

You give instructions before the model starts.



Prefix-Tuning

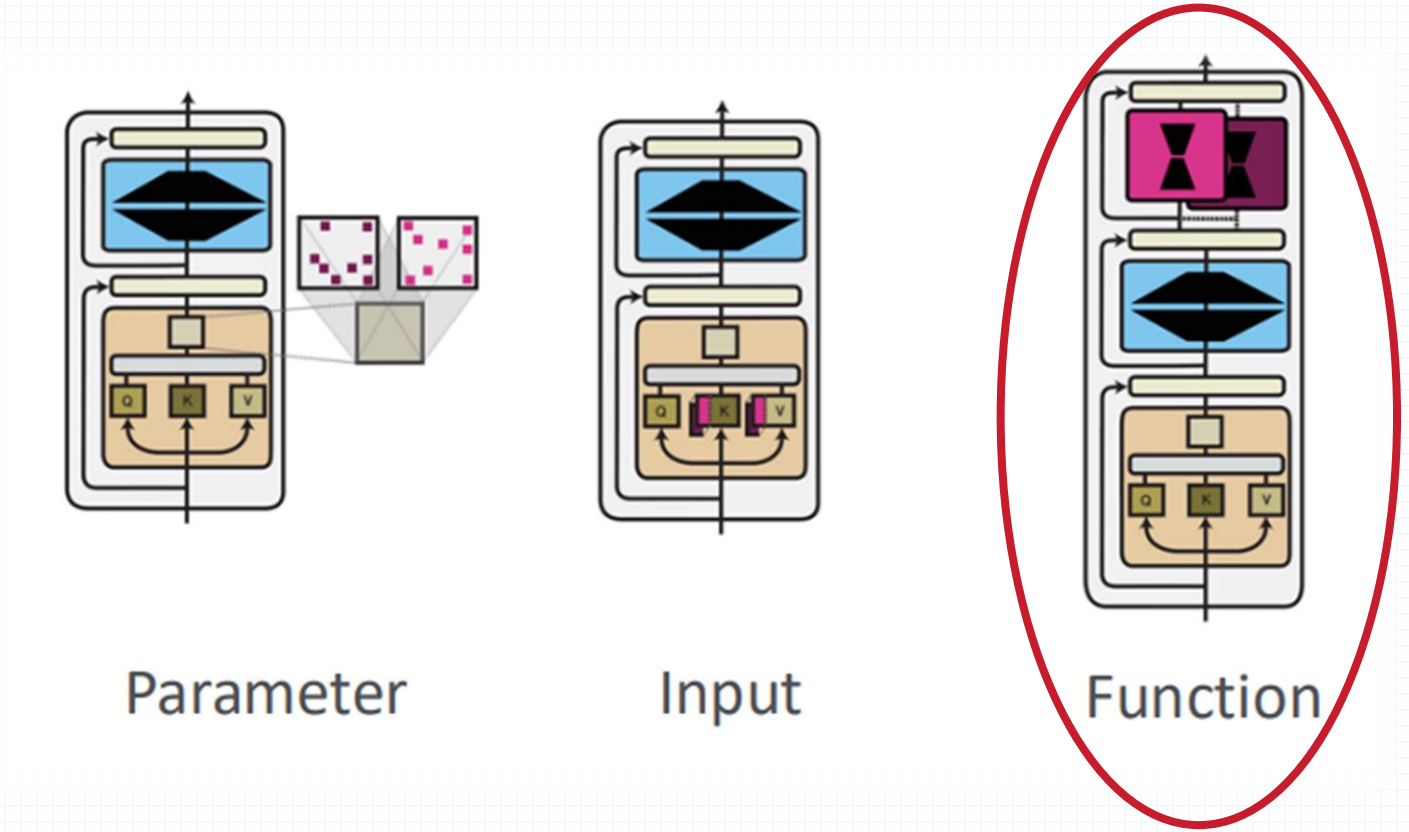
You provide guidance at every step (every layer).



(guidance at every layer)

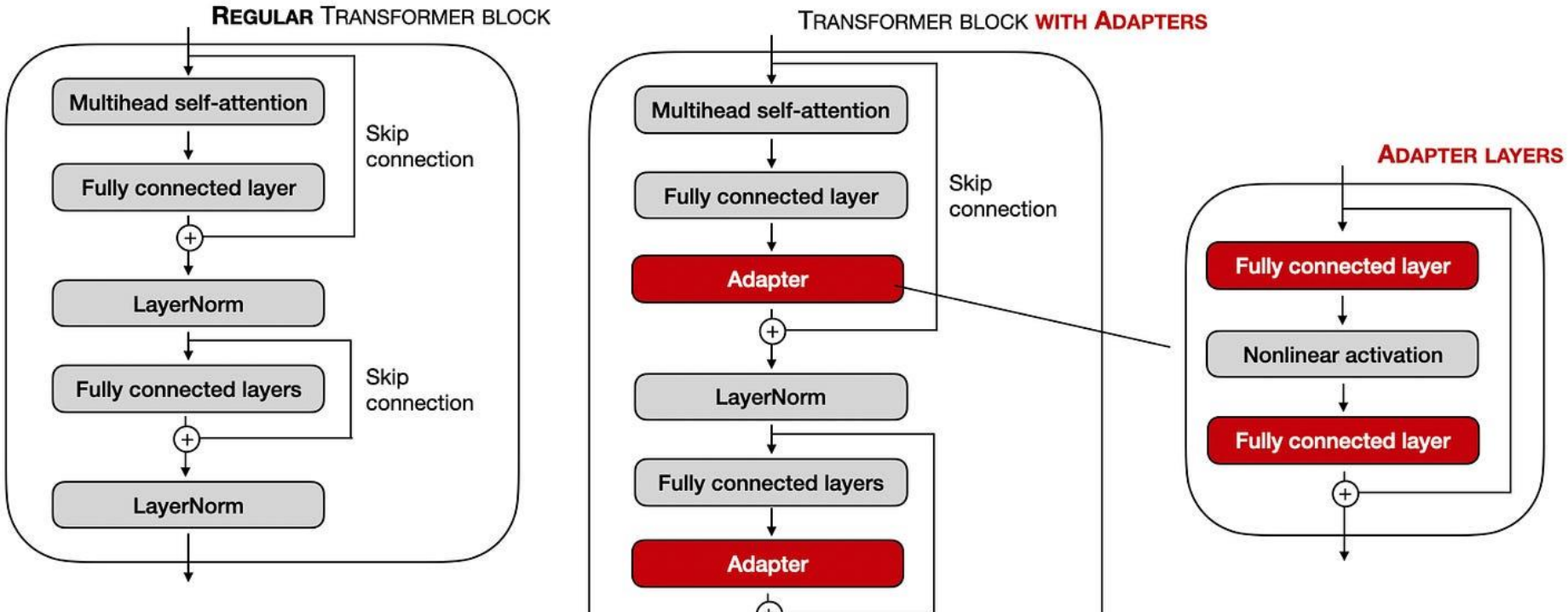
★ **Takeaway:** Prompt-Tuning is simple and lightweight but shallow; Prefix-Tuning is deeper and more powerful.

A functional perspective of adaptation

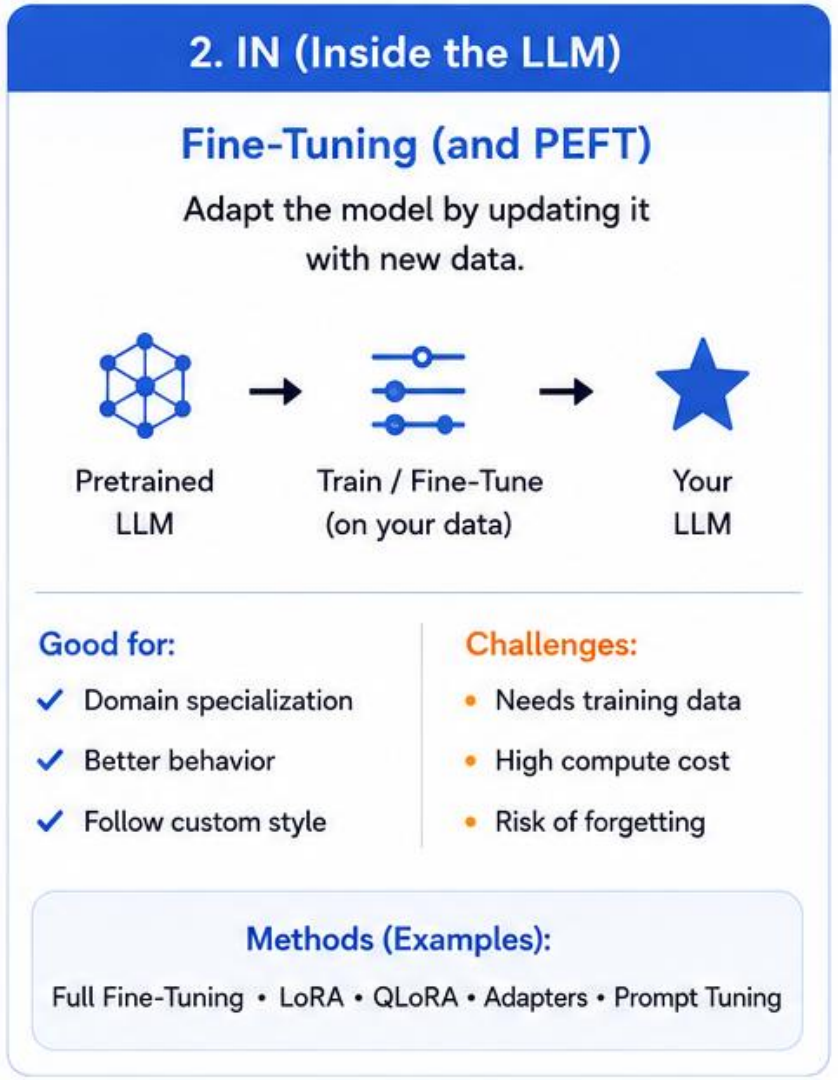


A functional perspective of adaptation

- **Adapter fine-tuning** is a parameter-efficient technique that allows for efficient adaptation of large language models (LLMs) to specific tasks by training small, task-specific "adapter" modules instead of updating all model parameters.
- Adapters: Adapters are small neural network layers (typically a pair of fully connected layers) inserted into the pre-trained model.



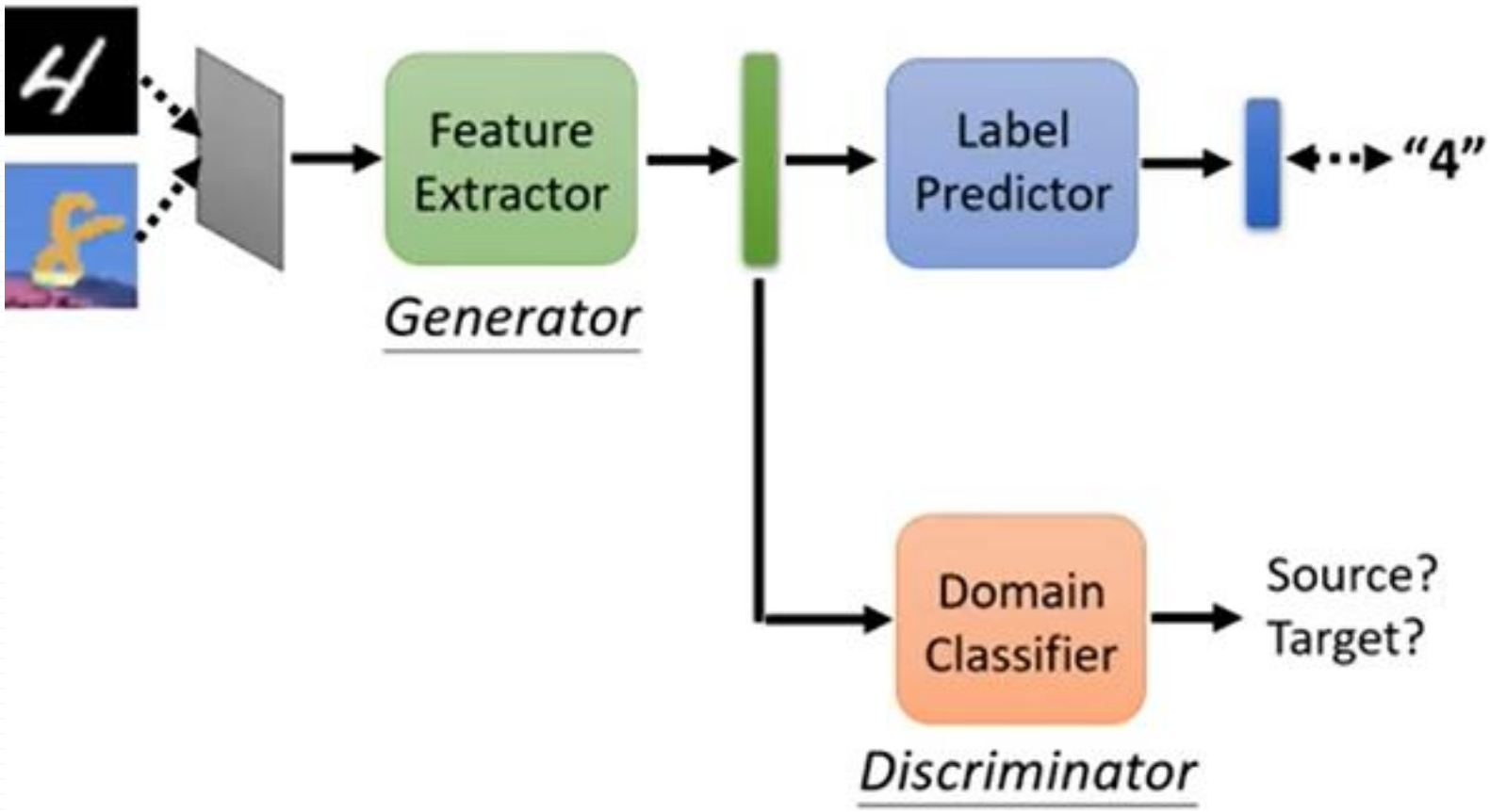
LLM in practice: Domain Adaptation



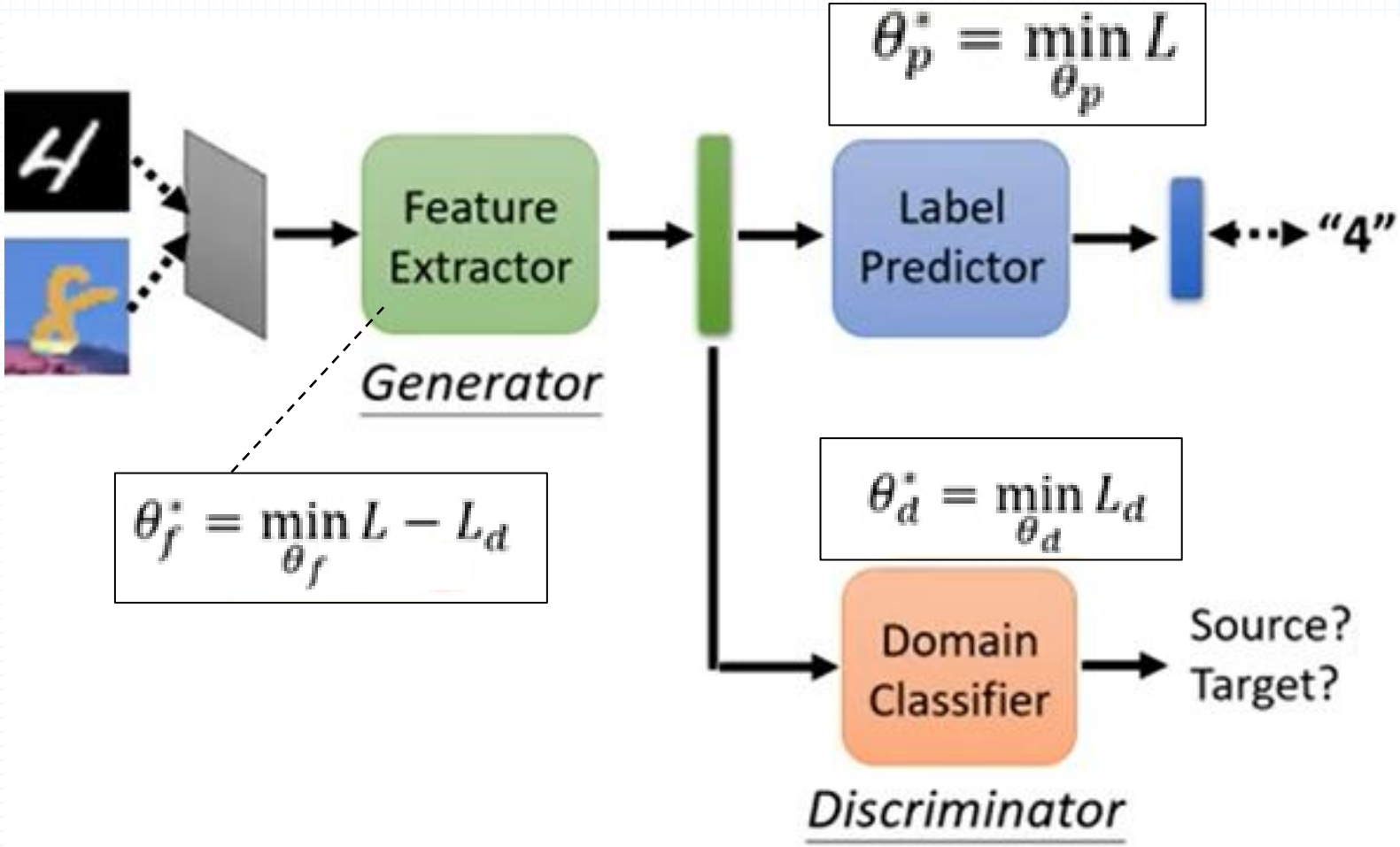
Domain Adaptation

- Domain adaptation is a field associated with machine learning and transfer learning.
- It addresses the challenge of training a model on one data distribution and applying it to a related but different data distribution.

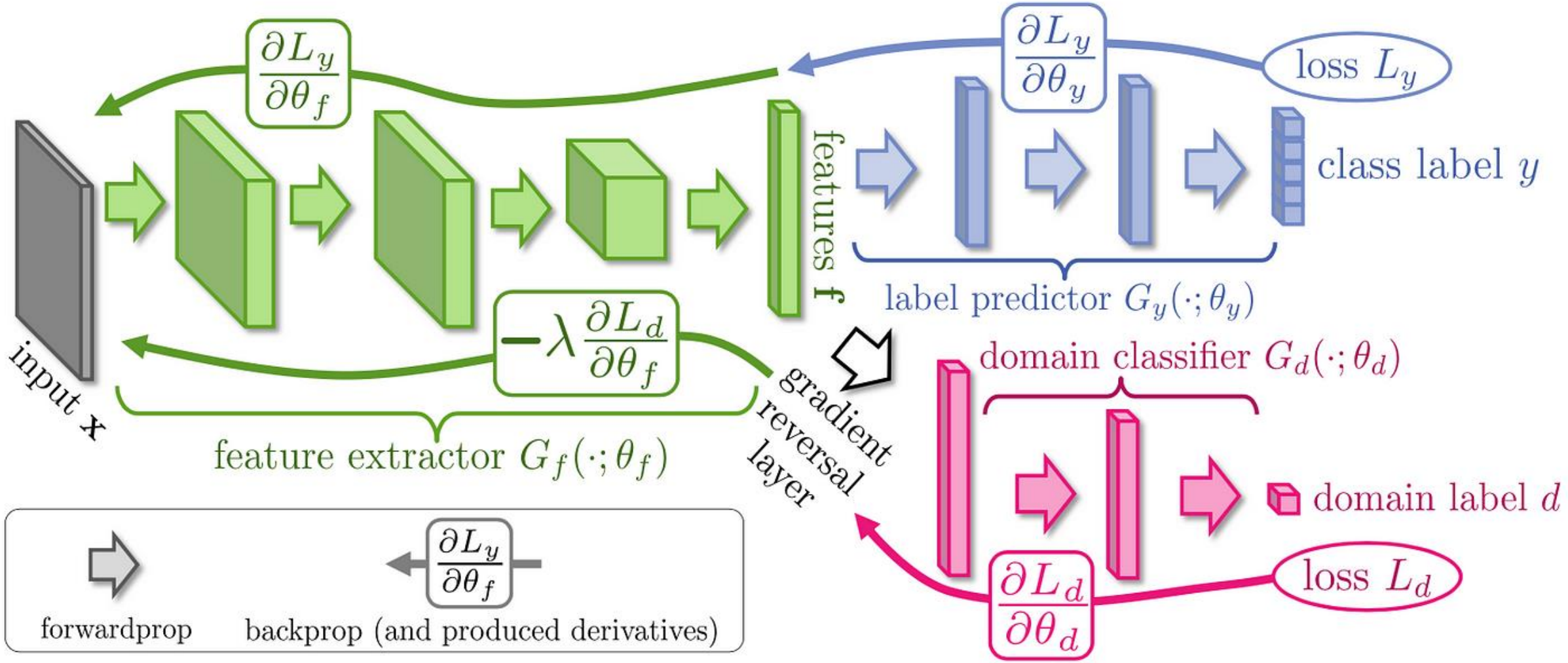
Domain Adaptation (domain adversarial training)



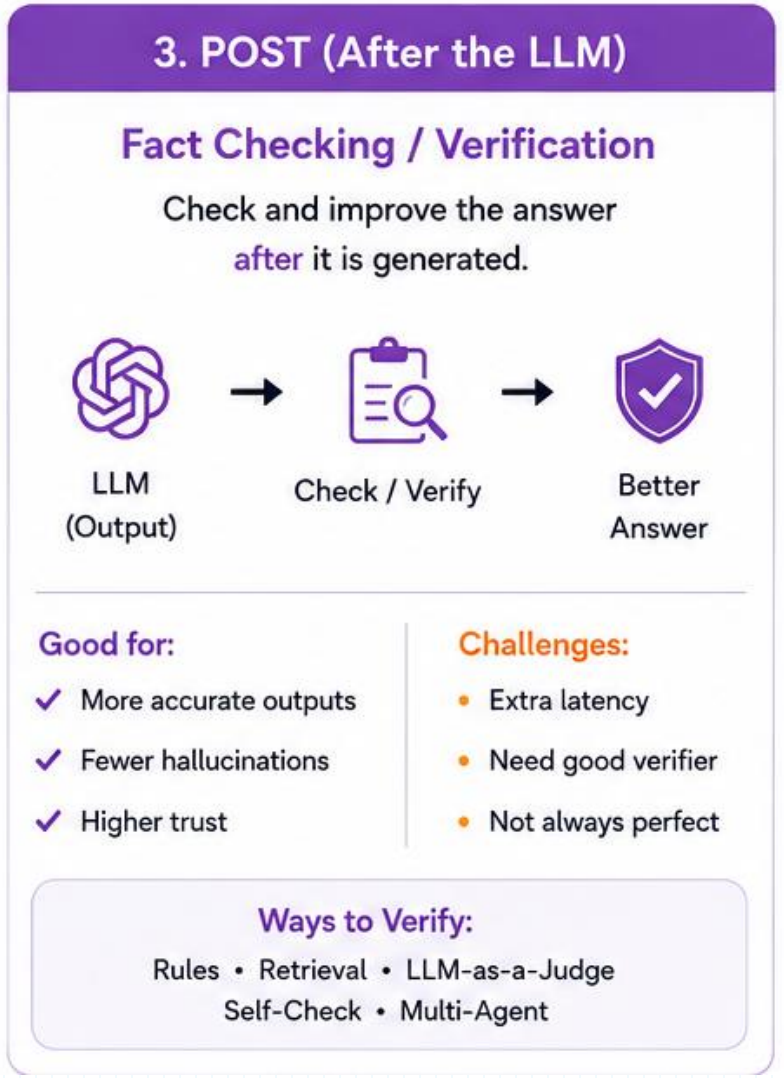
Domain Adaptation (domain adversarial training)



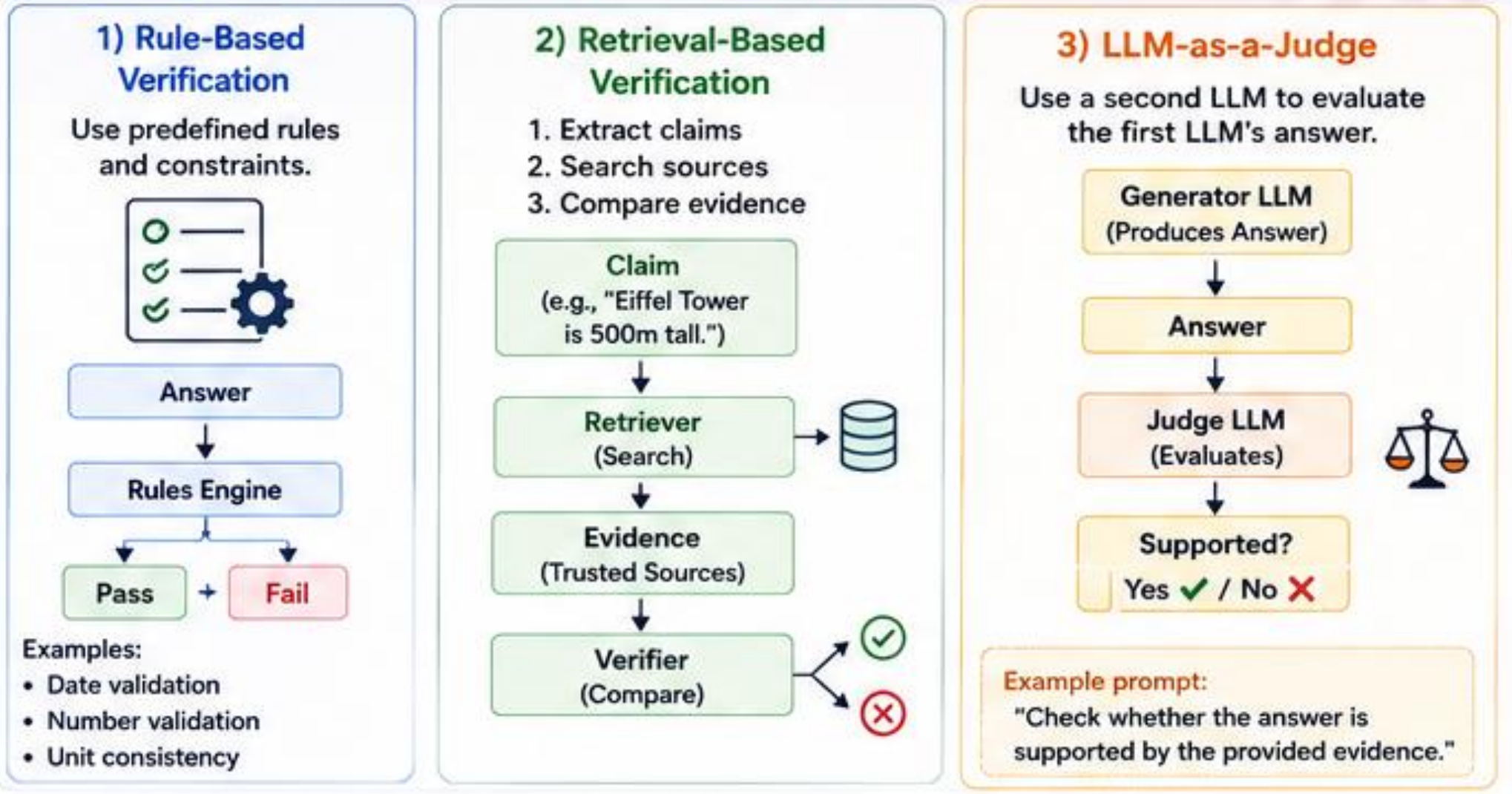
Domain Adaptation (domain adversarial training)



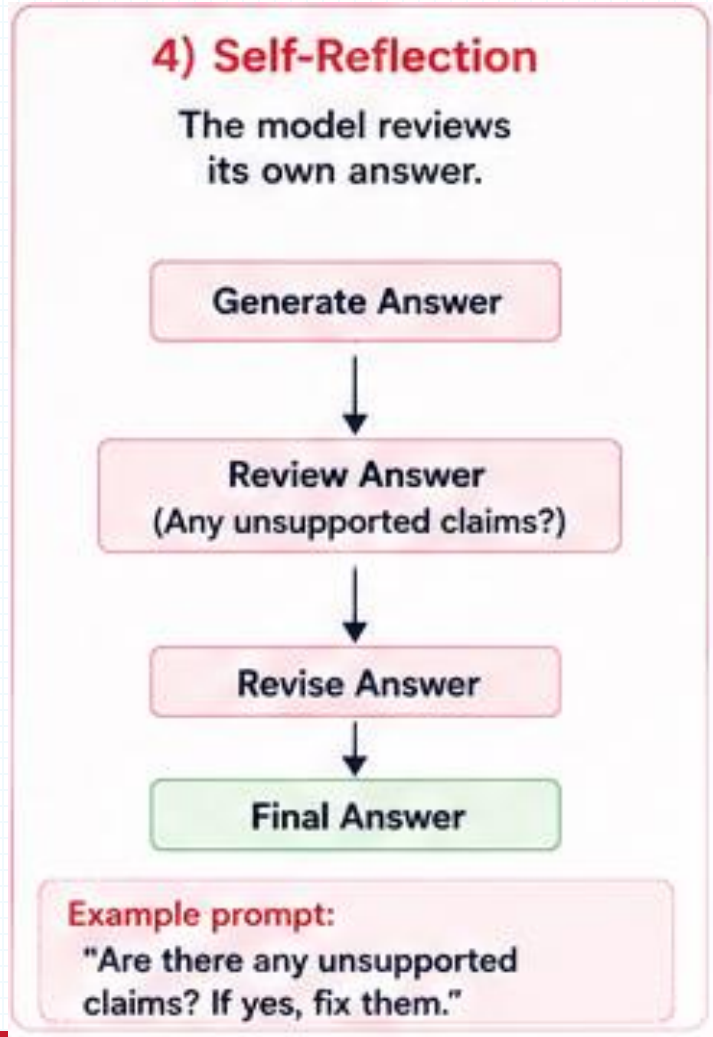
LLM in practice: Fact Checking

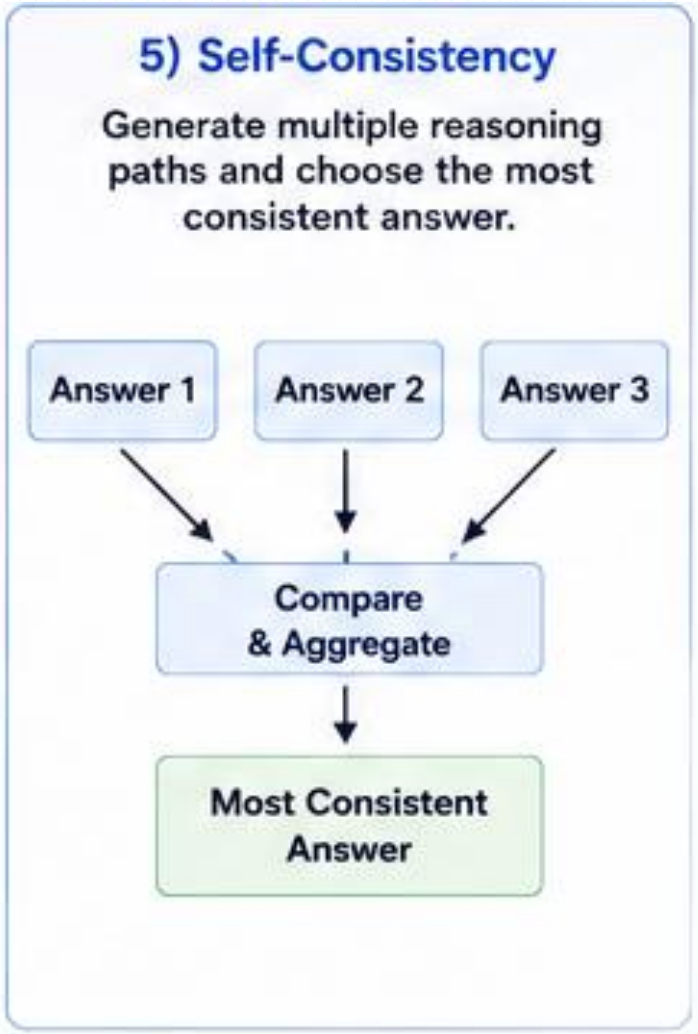


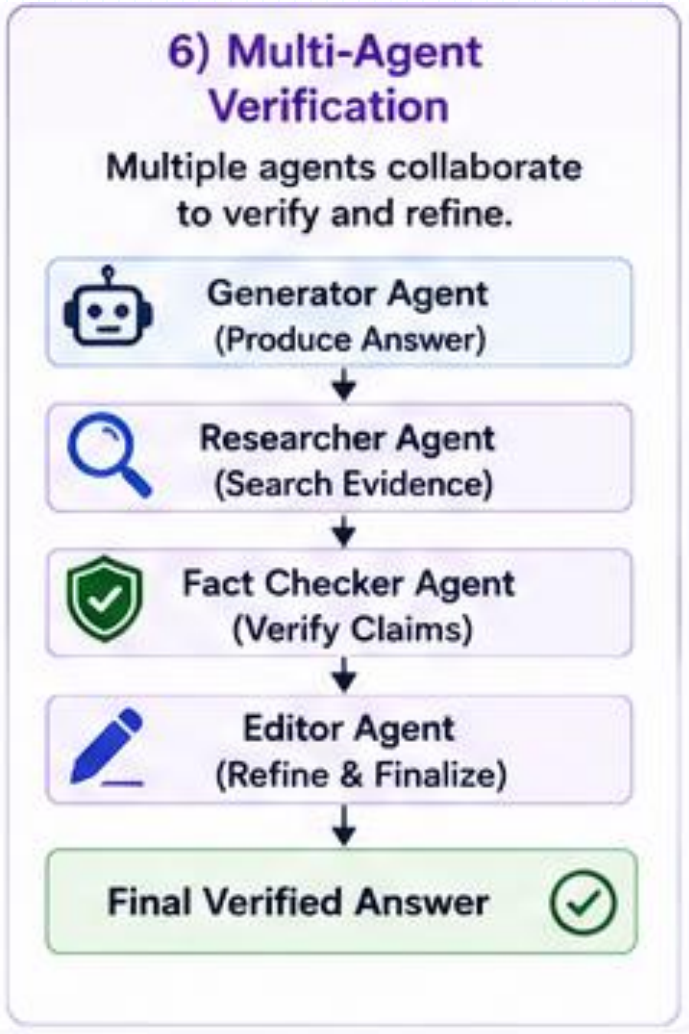
Fact Checking Methods



Fact Checking Methods







LLMs in Practice: Three Ways to Adapt

We can improve LLMs in three places: before, inside, or after the model.

1. PRE (Before the LLM)

RAG (Retrieval-Augmented Generation)

Give the model the right information before it answers.

Good for:

- ✓ Up-to-date info
- ✓ Private data
- ✓ Lower cost
- ✓ No training

Challenges:

- Retrieval quality
- Context limit
- Latency

2. IN (Inside the LLM)

Fine-Tuning (and PEFT)

Adapt the model by updating it with new data.

Good for:

- ✓ Domain specialization
- ✓ Better behavior
- ✓ Follow custom style

Challenges:

- Needs training data
- High compute cost
- Risk of forgetting

Methods (Examples):

Full Fine-Tuning • LoRA • QLoRA • Adapters • Prompt Tuning

3. POST (After the LLM)

Fact Checking / Verification

Check and improve the answer after it is generated.

Good for:

- ✓ More accurate outputs
- ✓ Fewer hallucinations
- ✓ Higher trust

Challenges:

- Extra latency
- Need good verifier
- Not always perfect

Ways to Verify:

Rules • Retrieval • LLM-as-a-Judge
Self-Check • Multi-Agent