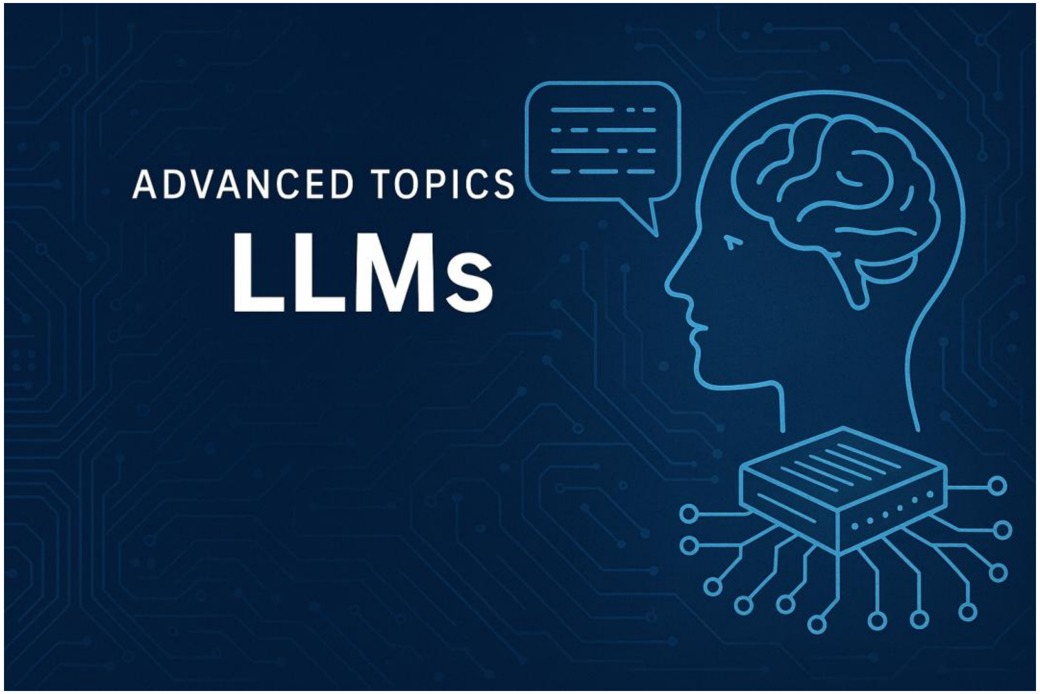
University of Ljubljana, Faculty of Computer and Information Science

Other useful topics in LLMs



Prof Dr Marko Robnik-Šikonja

Natural Language Processing, Edition 2025

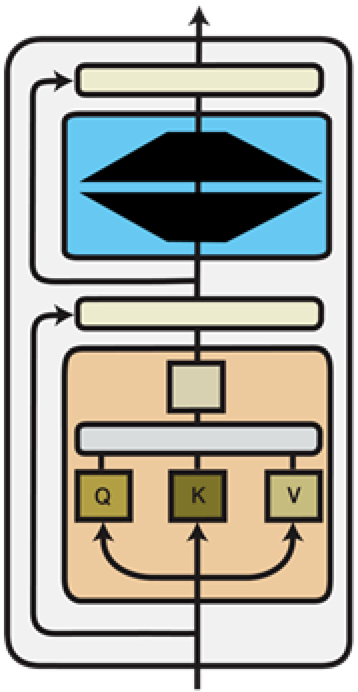
Contents

- parameter efficient fine-tuning (PEFT)
- agent architectures

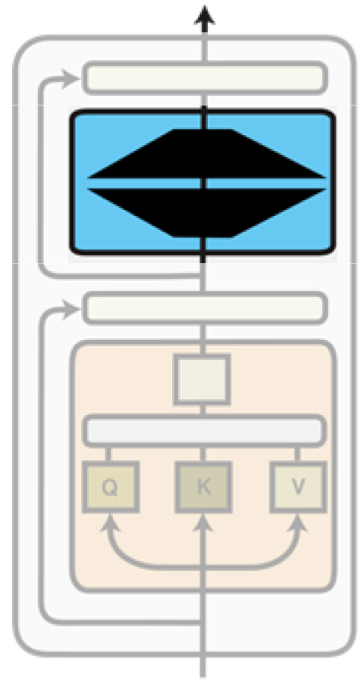
Camile Lendering, Manfred González, and Joaquín Figueira: Efficient fine-tuning techniques for Slovenian language models. *Proceedings of Language technologies & digital humanities conference, 2024*.

- Some slides and examples adapted from Yang , Ruder, Pfeiffer, & Vulić
- check out: https://www.modulardeeplearning.com/

Parameter efficient fine-tuning (PEFT)



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



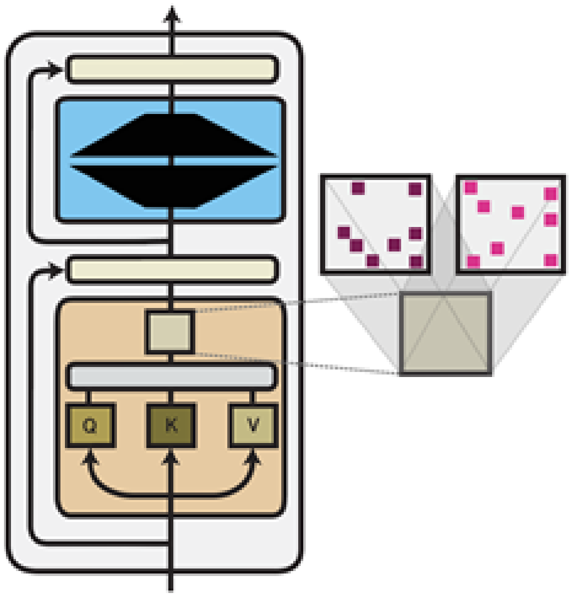
Parameter-efficient Fine-tuning
Update a **small subset** of model parameters

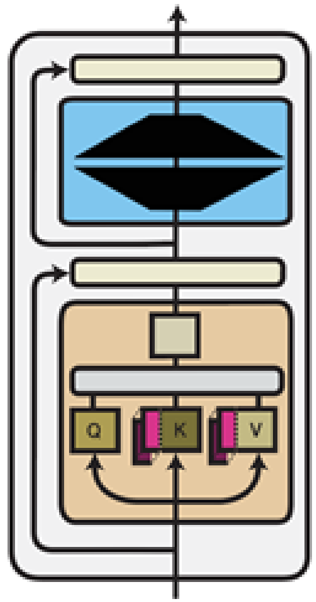
Why PEFT?

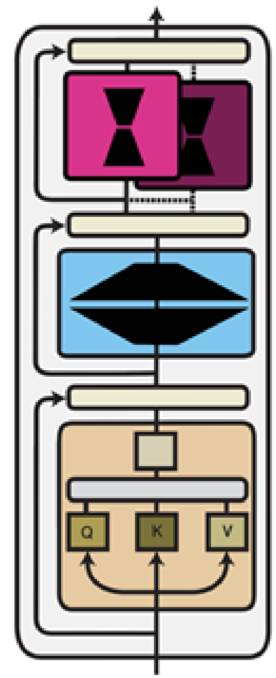
- Why fine-tuning *only some* parameters?
- Fine-tuning all parameters is impractical with large models. Why?
- State-of-the-art models are massively over-parameterized
- → Parameter-efficient fine-tuning (almost) matches performance of full fine-tuning

- 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 large companies

Opportunities for PEFT







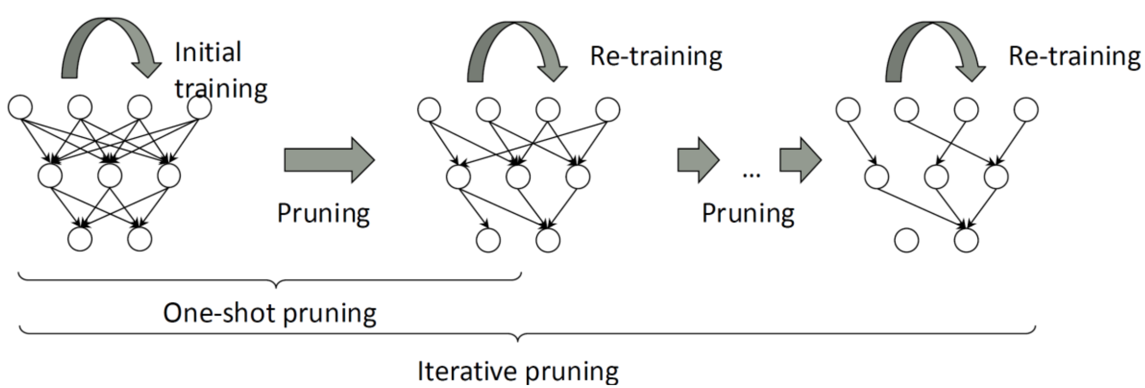
Parameter

Input

Functions

Parameters: Sparse subnetworks

- A common inductive bias on the module parameters is **sparsity**
- Most common sparsity method: **pruning**
- Pruning can be seen as applying a binary mask $\mathbf{b} \in \{0, 1\}^{\theta}$ that selectively keeps or removes each connection in a model and produces a subnetwork.
- Most common pruning criterion: **weight magnitude**
- Sparsity ratios: from 40% (SQuAD) to 90% (QQP and WNLI)
- During pruning, a fraction of the lowest-magnitude weights are removed
- The non-pruned weights are re-trained
- Pruning for multiple iterations is more common



The full fine-tuning

- Assume we have a pre-trained autoregressive language model $P_{\phi}(y|x)$
- E.g., GPT based on Transformer
- Adapt this pretrained model to downstream tasks (e.g., summarization, NL2SQL, reading comprehension)
- Training dataset of context-target pairs $\{(x_i, y_i)\}$ $i=1,\dots,N$
- During full fine-tuning, we update the parameters of the model ϕ_o to $\phi_o + \Delta\phi$ by following the gradient to maximize the conditional language modeling objective

$$\max_{\phi} \sum_{(x,y)} \sum_{t=1}^{|y|} \log(P_{\phi}(y_t|x, y_{<t}))$$

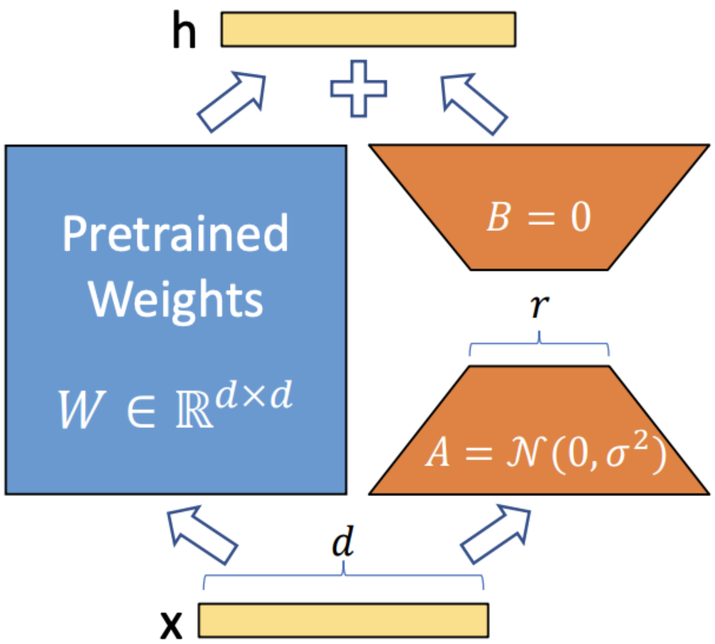
LoRA: low rank adaptation

- Full fine-tuning: For each downstream task, we learn a different set of parameters $\Delta\phi$
- $|\Delta\phi| = \phi_o$
- GPT-3 has a $|\phi_o|$ of 175 billion
- Expensive and challenging for storing and deploying many independent instances
- Can we do better?
- **Key idea**: encode the task-specific parameter increment $\Delta\phi = \Delta\phi(\Theta)$ by **a smaller-sized set of parameters $\boldsymbol{\Theta}$**, $\Theta \ll |\phi_o|$
- The task of finding $\Delta\phi$ becomes optimizing over Θ

$$\max_{\Theta} \sum_{(x,y)} \sum_{t=1}^{|y|} \log(P_{\phi_o+\Delta\phi(\Theta)}(y_t|x, y_{<t}))$$

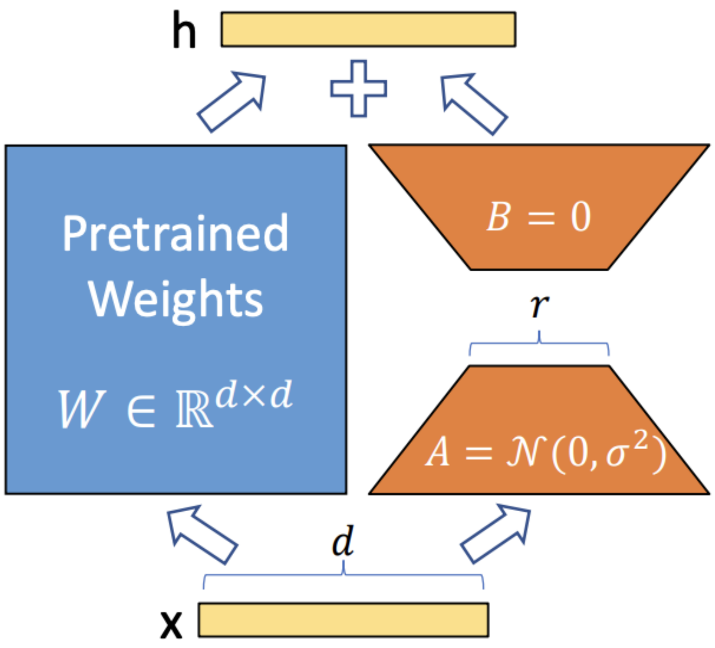
Low-rank-parameterized update matrices

- Updates to the weights have a low "intrinsic rank" during adaptation
- $W_0 \in \mathbb{R}^{d \times k}$: a pretrained weight matrix
- Constrain its update with a low-rank decomposition: $W_0 + \Delta W = W_0 + \alpha BA$ where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, $r \ll \min(d, k)$
- α is the tradeoff between pre-trained "knowledge" and task-specific "knowledge"
- Only A and B contain **trainable** parameters



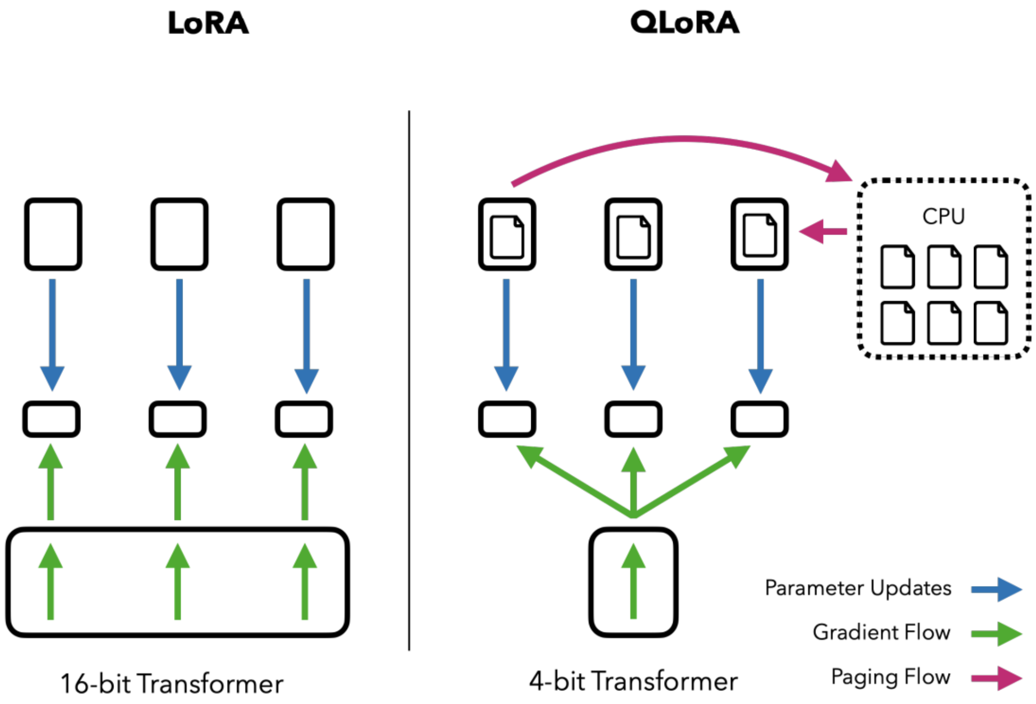
LoRA details

- As one increase the number of trainable parameters, training LoRA converges to training the original model
- **No additional inference latency:** when switching to a different task, recover W_0 by subtracting BA and adding a different $B'A'$
- Often LoRA is applied to the weight matrices in the self-attention module

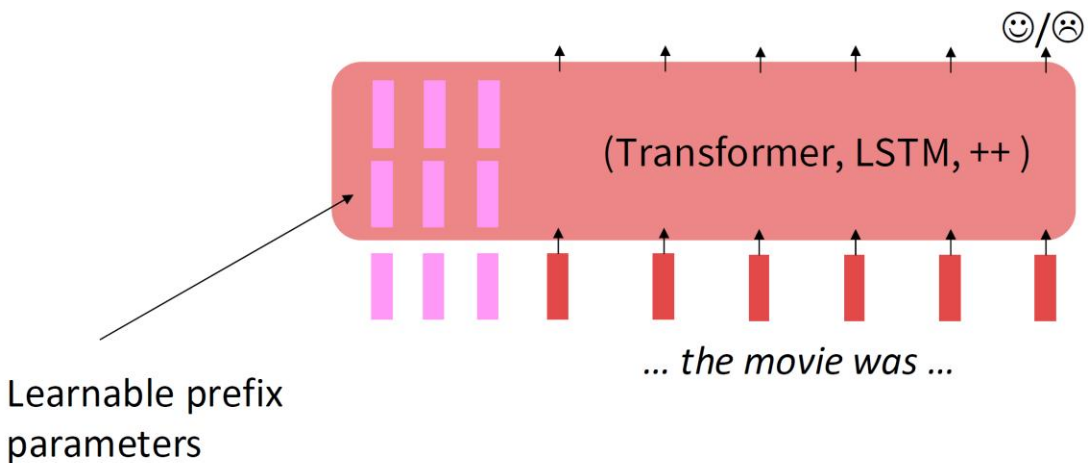


From LoRA to QLoRA

- QLORA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizer to handle memory
- •4-bit NormalFloat (NF4)
- A new data type that is information theoretically optimal for normally distributed weights

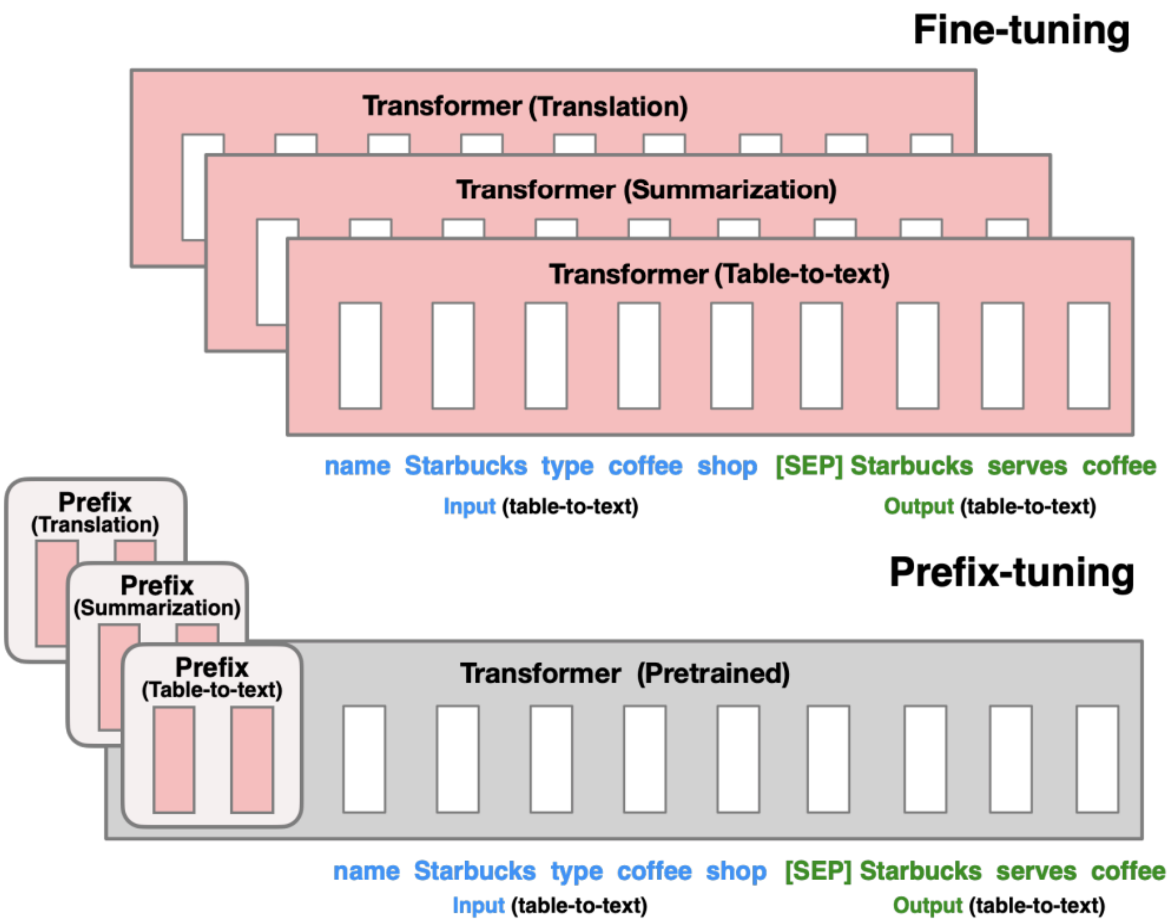


An input perspective of adaptation



Prefix-Tuning

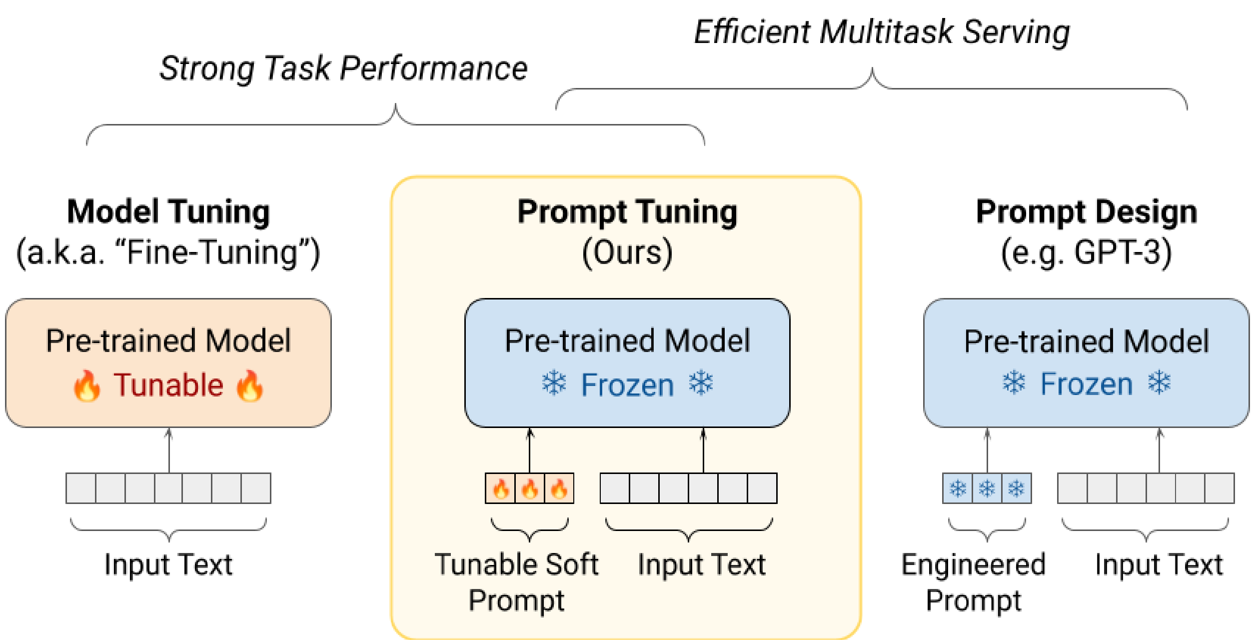
- Prefix-Tuning adds a **prefix** of parameters and **freezes all pretrained parameters.**
- •The prefix is a sequence of continuous task-specific vector and is processed by the model just like real words would be, i.e., **“virtual tokens”.**
- •**Advantage:** each element of a batch at inference could run a different tuned model.



Prompt-Tuning

- Learning "soft prompts" to condition frozen LMs to perform downstream tasks
- Prepend virtual tokens to input, and learn embeddings of these special tokens only

- Standard model tuning achieves strong performances but requires scoring separate copies of model for each end task
- •Prompt tuning matches the quality of model tuning as size increases



A functional perspective of adaptation

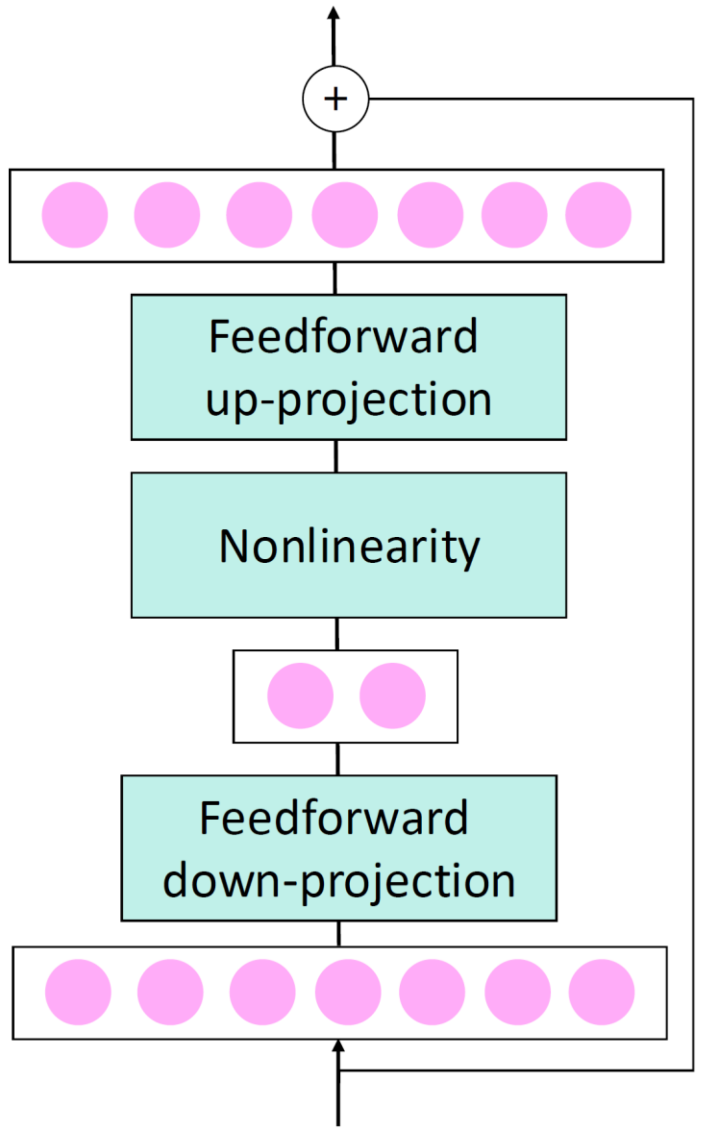
- Function composition augments a model's functions with new task-specific functions:

$$f_i'(\boldsymbol{x}) = f_{\theta_i}(\boldsymbol{x}) \odot f_{\phi_i}(\boldsymbol{x})$$

- Most commonly used in multi-task learning, where modules of different tasks are composed

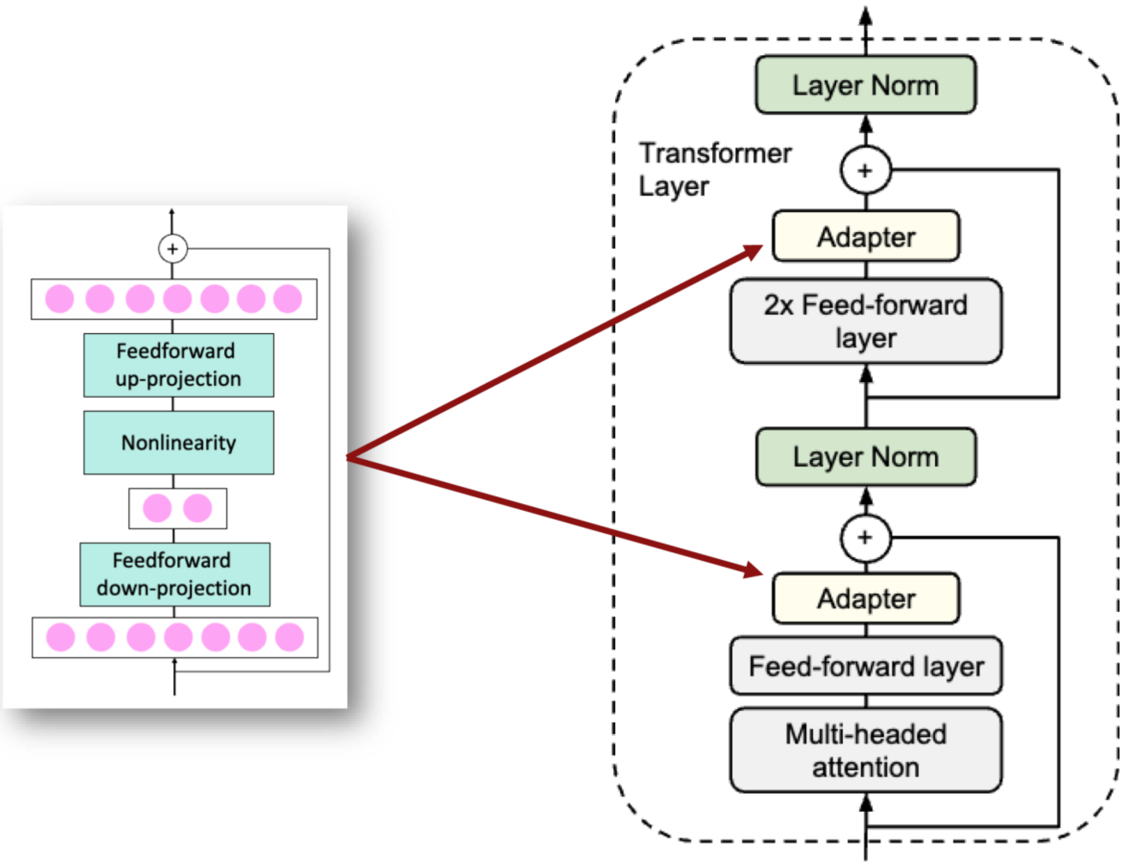
Adapters

- Insert a new function f_ϕ between layers of a pre-trained model to adapt to a downstream task
- known as “adapters”
- An **adapter** in a Transformer layer consists of:
- A feed-forward down-projection $W^D \in R^{k \times d}$
- A feed-forward up-projection $W^U \in R^{d \times k}$
- $f_\phi(\boldsymbol{x}) = W^U(\sigma(W^D \boldsymbol{x}))$



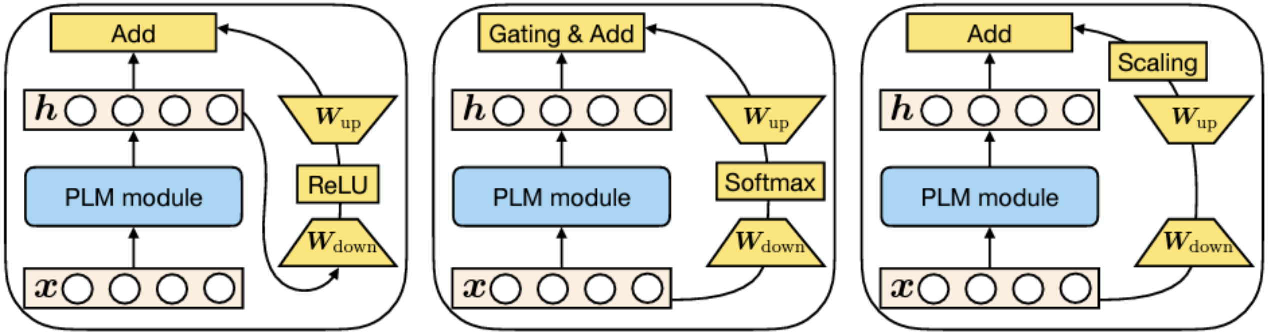
Adapter placement

- The adapter is usually placed after the multi-head attention and/or after the feed-forward layer
- Most approaches have used this bottleneck design with linear layers
- Adapter-based tuning attains a similar performance to full fine-tuning with two orders of magnitude fewer trained
- parameters



Unifying View

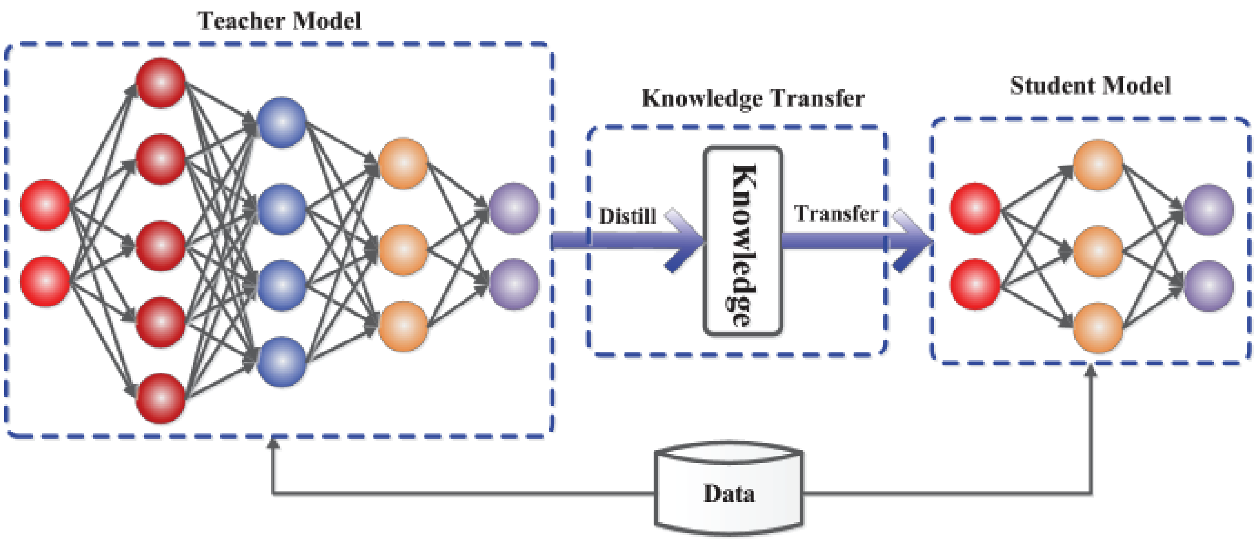
- • He et al. [2022] (Towards a unified view of parameter-efficient transfer learning) show that LoRA, prefix tuning, and adapters can be expressed with a similar functional form
- All methods can be expressed as modifying a model's hidden representation $\boldsymbol{h}$



(a) Adapter (b) Prefix Tuning (c) LoRA

- Sparsity, structure, low-rank approximations, rescaling, and other properties can also be applied and combined in many settings
- Prompt tuning underperforms the other methods due to limited capacity
- Adapter achieves better performance but add more parameters

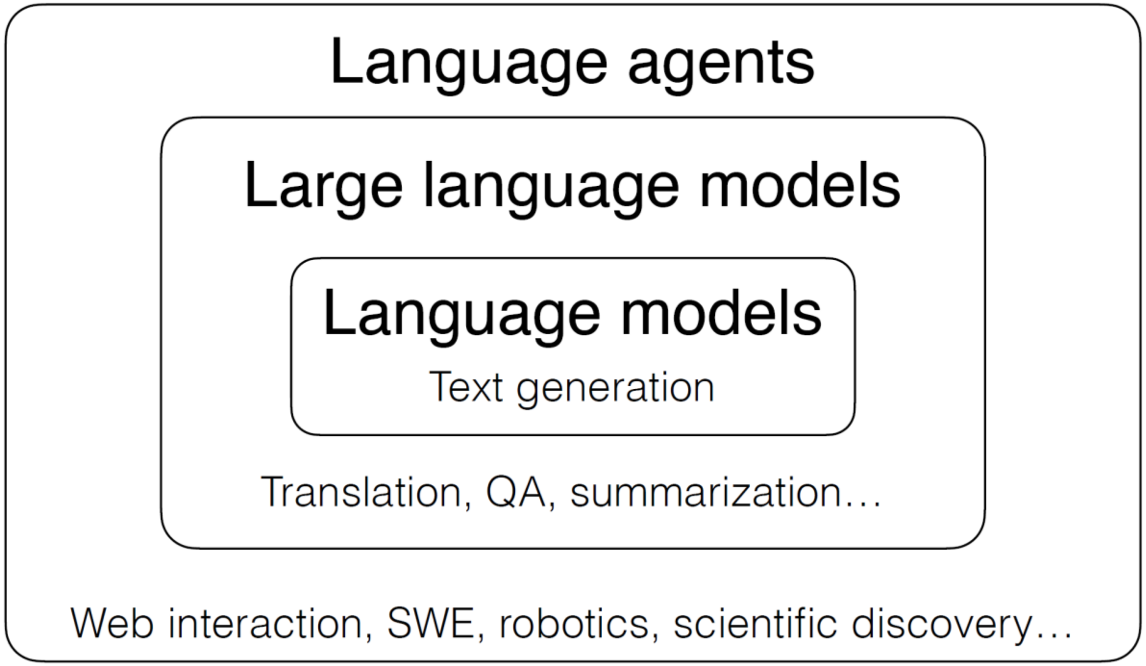
Knowledge distillation to obtain smaller models



- The generic teacher-student framework for knowledge distillation

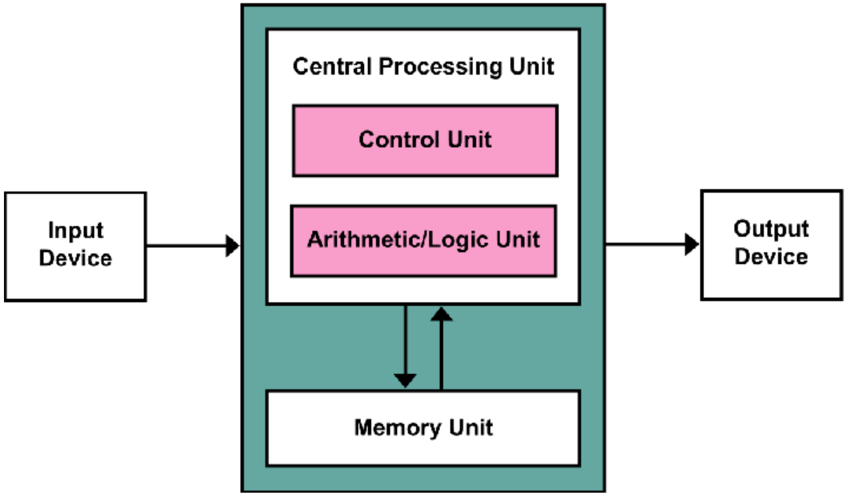
Language agents

- Example
  Sumers, Yao, Narasimhan, Griffiths. CoALA: Cognitive Architectures for Language Agents TMLR 2024



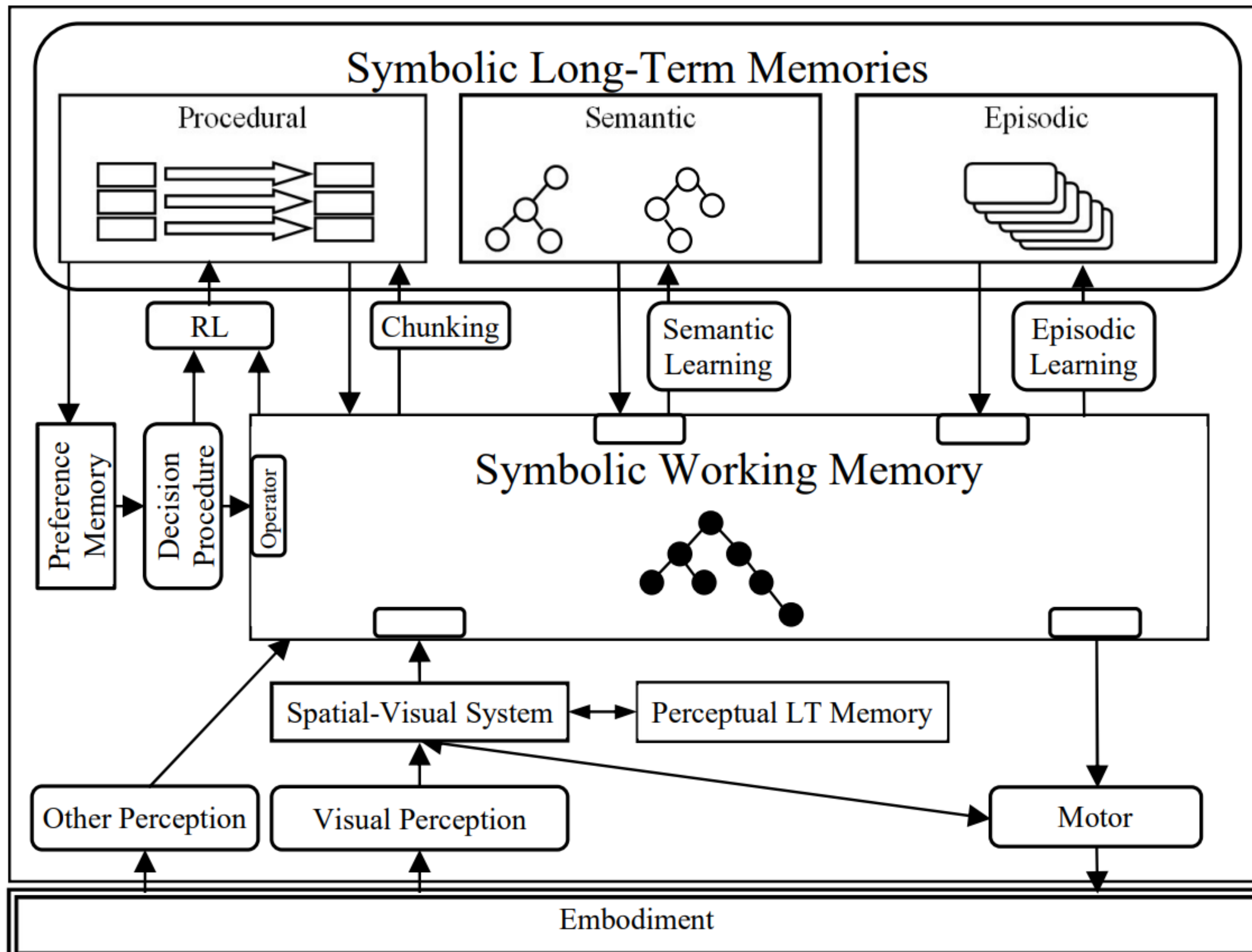
Architecture

- How do we make sense of various LLM systems (digital circuits)?
- Where should the field be going?
- von Neuman architecture

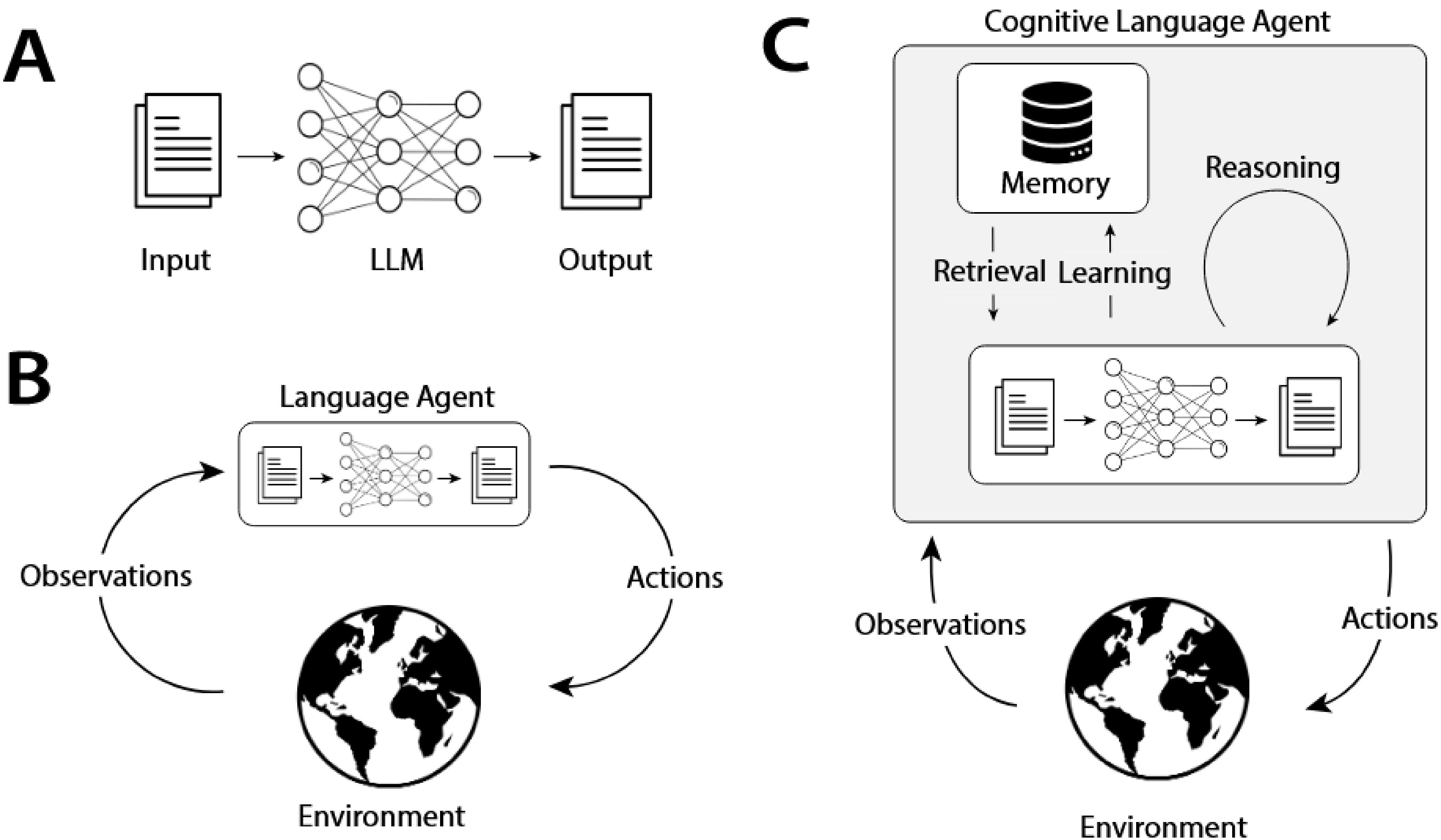


Cognitive architectures

- **Cognitive architectures**: frameworks to modularize and build complex symbolic AI agents, using cognitive inspirations
- E.g. Soar architecture
- or ACT-R



From LLMs to CLAs



Cognitive Architectures for Language Agents (CoALA)

- Memory: short and long term
- Action space: internal and external
  1. Reasoning (update short-term memory)
  2. Retrieval (read long-term memory)
  3. Learning (write long-term memory)
  4. Grounding (update external world)
- Decision making: choose an action