Introduction to Large Language Models(LLM)

Outline

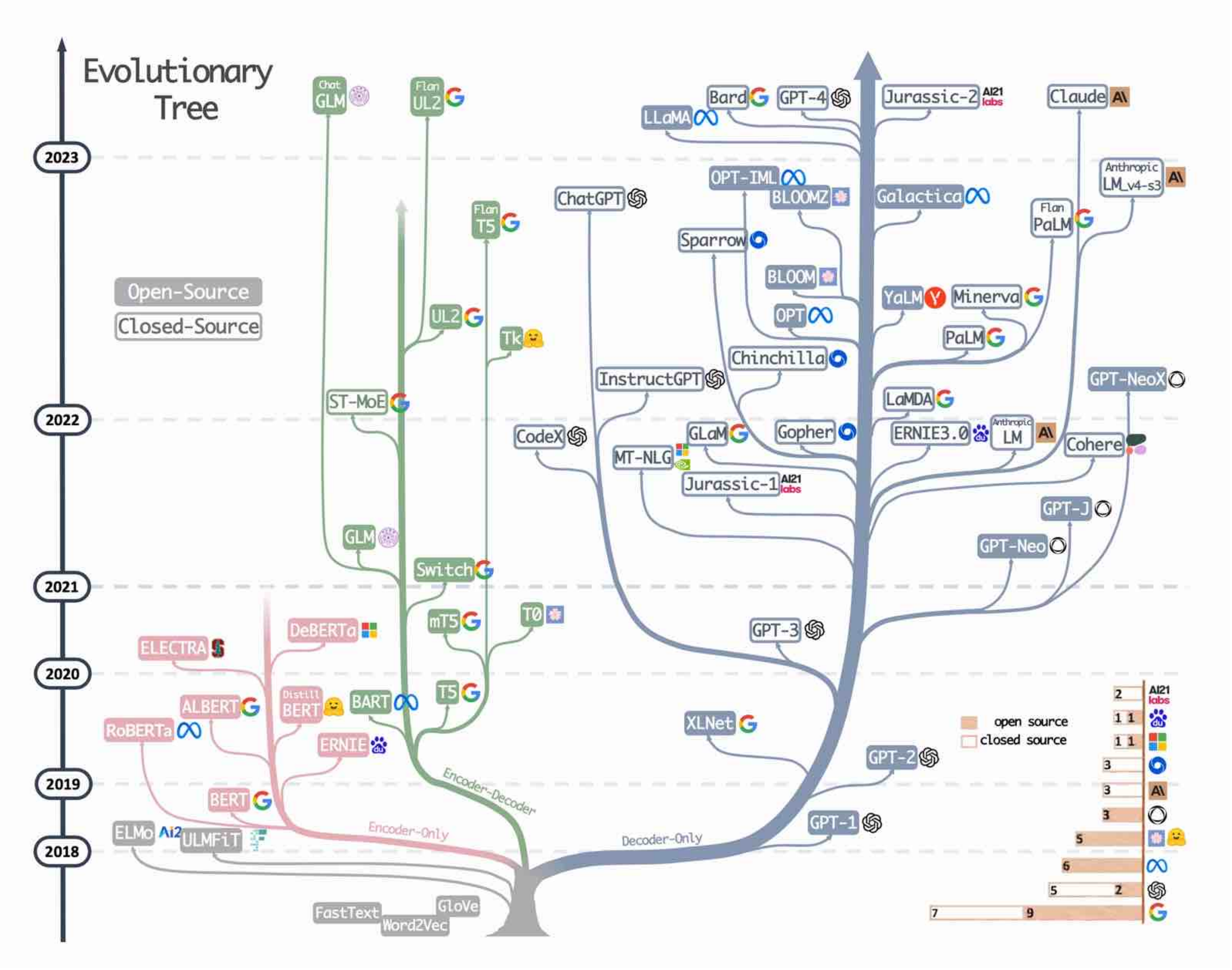
1. Overview from 30,000 feet above
2. Transformer in nutshell
3. Pretraining - Parallel paradigm
4. Finetuning - Parameter efficient finetuning
5. Steering the decoding of LLM - Prompting
6. Augmentation and Plugins

Overview from 30000 feet above

- Paradigm transition in AI
  - From: **training**(specific) -> **prediction**(specific)
  - To: **pretraining**(general) -> **finetuning**(general/specific) -> **in-context prompt**(specific)
- Primary steps of new paradigm
  - Pretraining with self-supervised learning
  - Finetuning on instruction from mutiple domains
  - Application by steering/prompt the decoding process of LLM
- Where are we to AGI?
  - From explanation to prediction
  - From correlation to causality

A LLMs Evolution Tree

- Decoder Only
- Encoder Only
- Encoder-Decoder



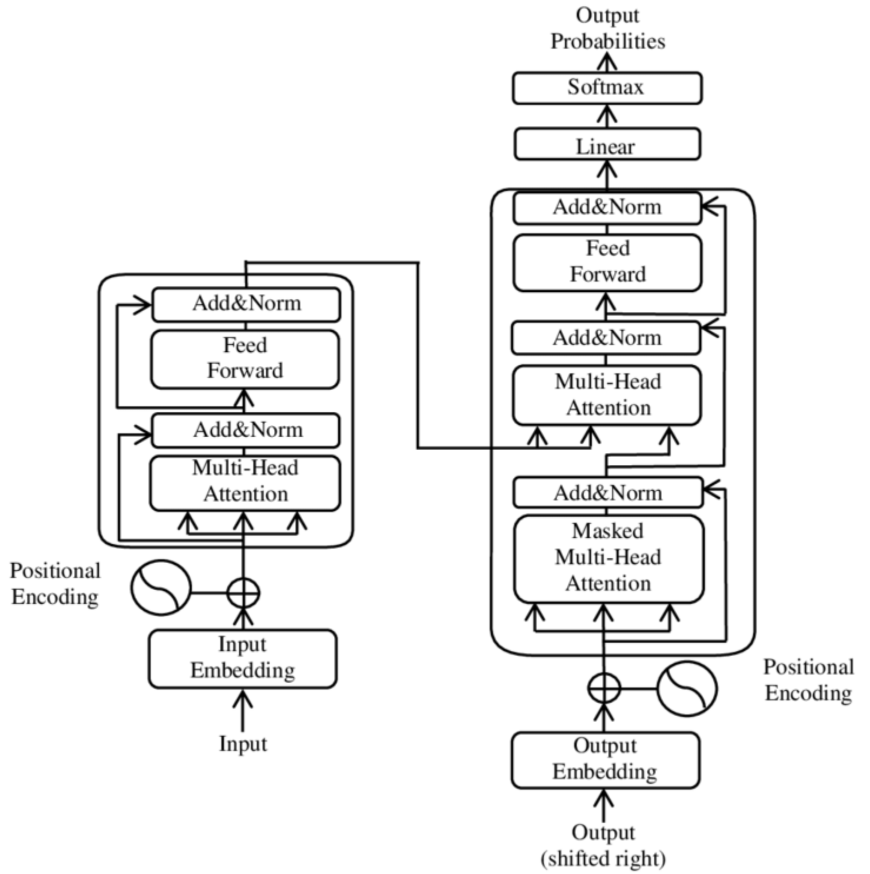
Img Source: Jingfeng Yang et al. 2023

Transformer in nutshell

- Transformer modules
- Aspects of alternative
- Parameter concentration
- Computation concentration

Transformer modules

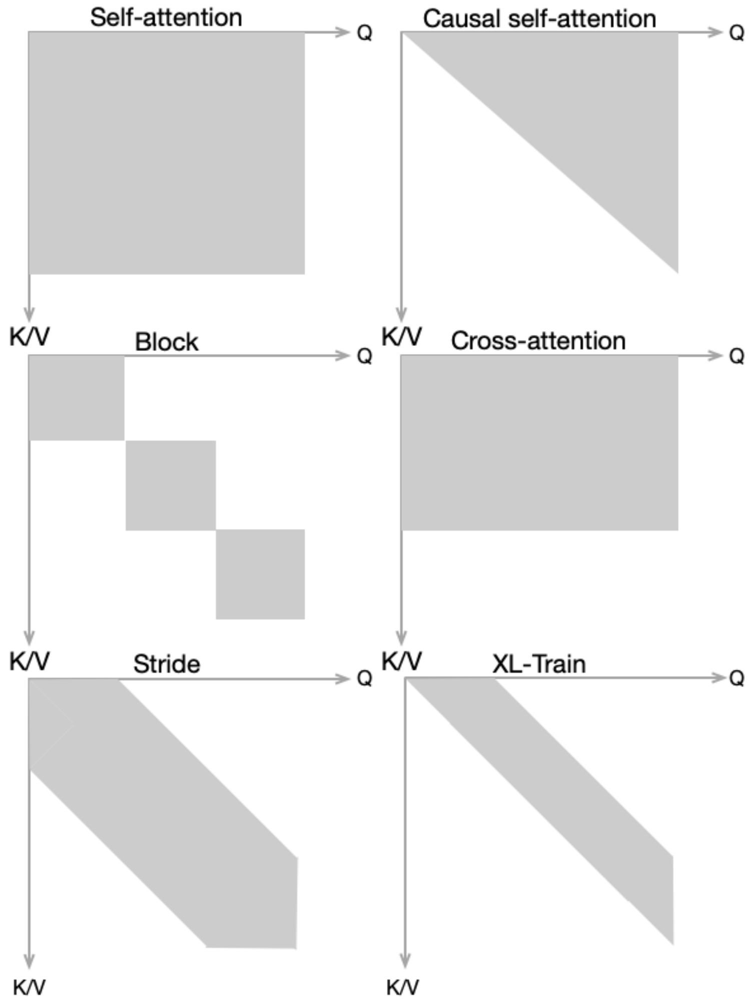
- Token & positional embedding :
  - $E \in R^{V \times d}, P \in R^{T \times d}$
- Multi-head attention
  - Self-attention & Cross-attention
    - Weight matrix: $W_Q^h, W_K^h, W_V^h \in R^{d \times d_h}$
    - Head projection: $W_O \in R^{d \times d}$
- ResNet & LayerNorm
- Feedfoward Network
  - $W_1 \in R^{d \times 4d}, W_2 \in R^{4d \times d}$
- Output head: task related



Source : Wiki:Transformer_(machine_learning_model)

Aspects of alternative

- Efficient Transformer (for long sequence)
    - Coarse sequence resolution:
        - Block/Stride/Clustering/Neural Memory
        - TransformerXL
    - Attention matrix approximation:
        - Linear Transformer
- LLMs specific
    - **Encoder** vs **Decoder** vs **Encoder-Decoder**
    - Pretraining objective
    - Positional encoding
    - Input or output LayerNorm
    - Activation



Parameter concentration

Decoder only Transformer(GPT)

- Parameter size:
  - Embedding: $(V + T) \times d$
  - Attention: $L \times (3 \times d \times d_h \times H + (d_h \times H) \times d) = 4Ld^2$
  - FFN: $L \times ((d \times 4d + 4d) + (4d \times d + d)) \approx 8Ld^2$
- On GPT3-175B:

Total	PE	TE	Attn	FFN
174,597M	25M(0.01%)	617M(0.35%)	57,982M(33.21%)	115,970M(66.42%)

Computation concentration

Decoder only Transformer

Per-token calculation:

- QKV+project: $2 \times L \times (3 \times H \times h_d \times d + (H \times h_d) \times d) = 2 \times 4Ld^2$
- Attention: $2 \times L \times T \times d$
- FFN: $2 \times L \times ((d \times 4d + 4d) + (4d \times d + d)) \approx 2 \times 8Ld^2$

Model training flops utilization(MFU):

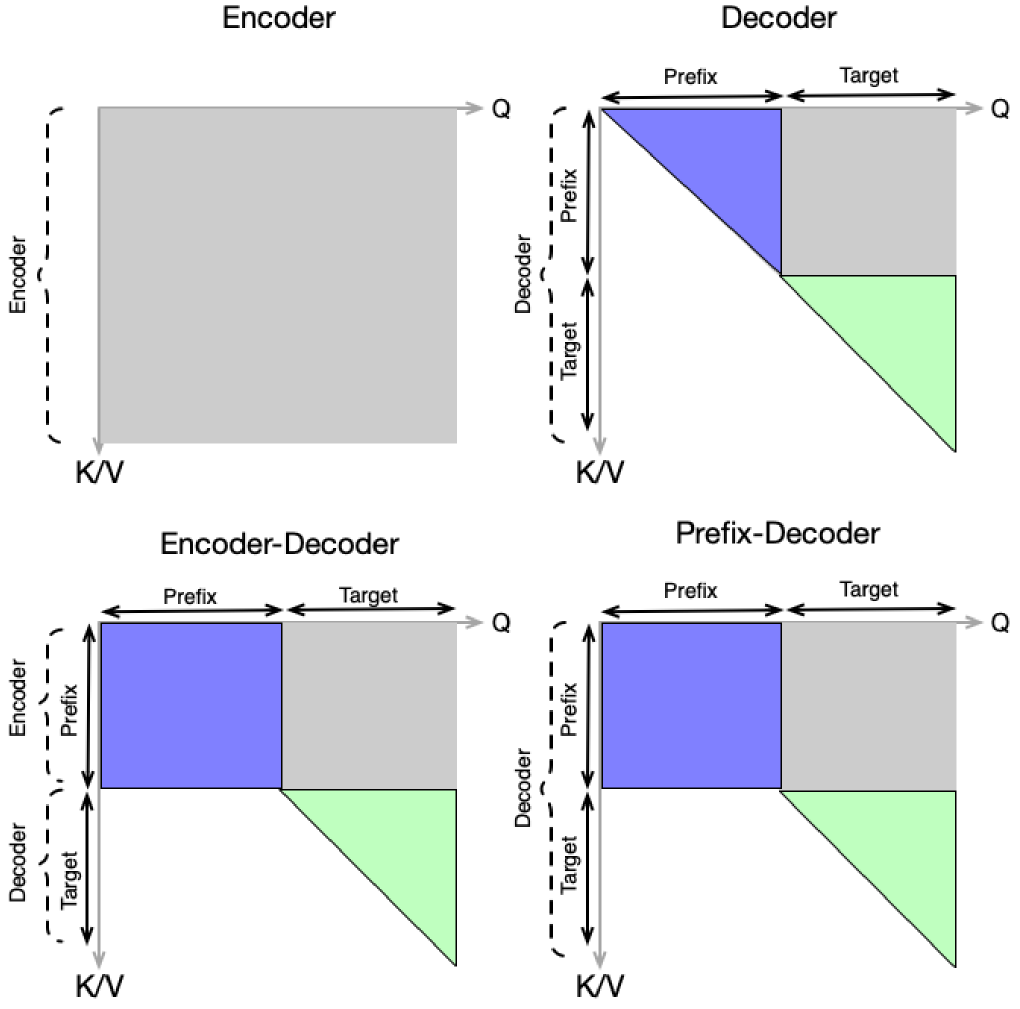
- Forward and backward: $(1 + 2) \times (2N + 2LTd) \approx 6N$, where $N \approx 12Ld^2$
- Theoretical peak throughput: $\frac{P}{6N+6LTd}$
- $\text{MFU} = \frac{\text{Observed throughput}}{\text{Theoretical peak throughtput}}$

Pretraining

- Training objectives
- Text Corpus
- Parallel strategies
- Results & Evaluation

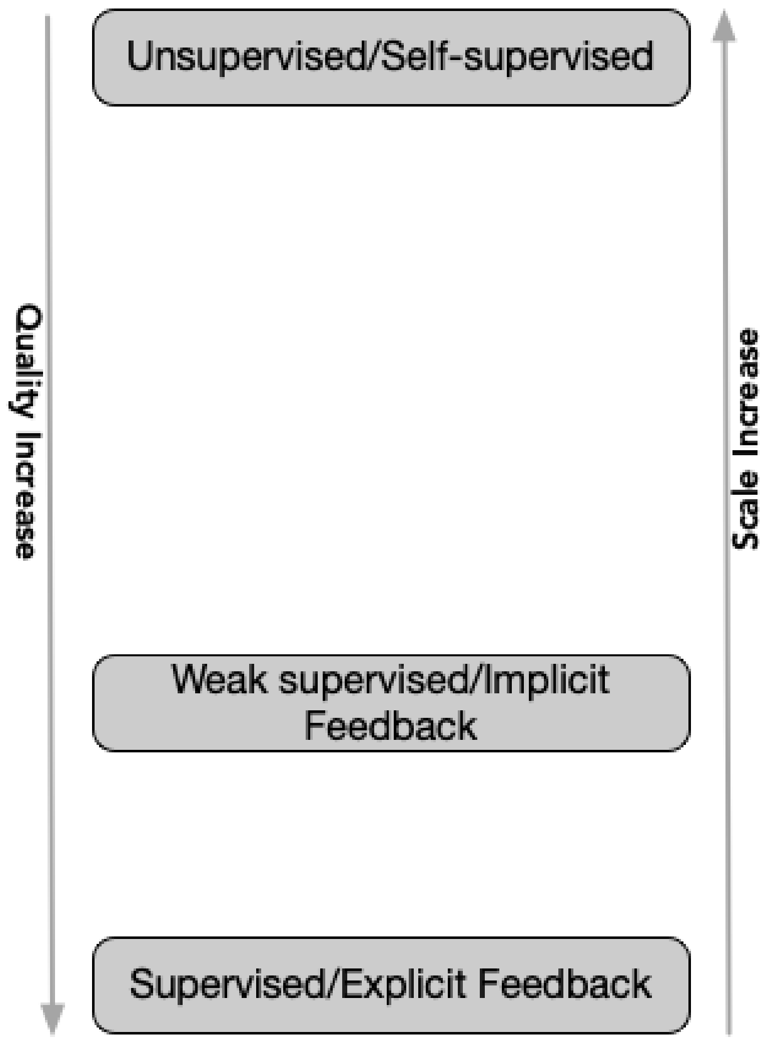
Training objectives

- Masked Language Models
- Auto-regressive Language Models



Text Corpus

- Unsupervised text
  - BookCorpus : 11,000; Gutenberg : 70,000
  - OpenWebText: 8M outlinks of Reddit.com
  - Common Crawl; C4
  - Code: BigQuery Github
- Weak supervised text
  - Reddit TL;DR; PushShift.io Reddit: Posts
  - StackExchange : Question & Answer w/ score
- Supervised text: task related
  - ~16 NLP tasks related datasets (Sentiment/QA/Reasoning, etc.)
  - Human answer to prompt: InstructGPT



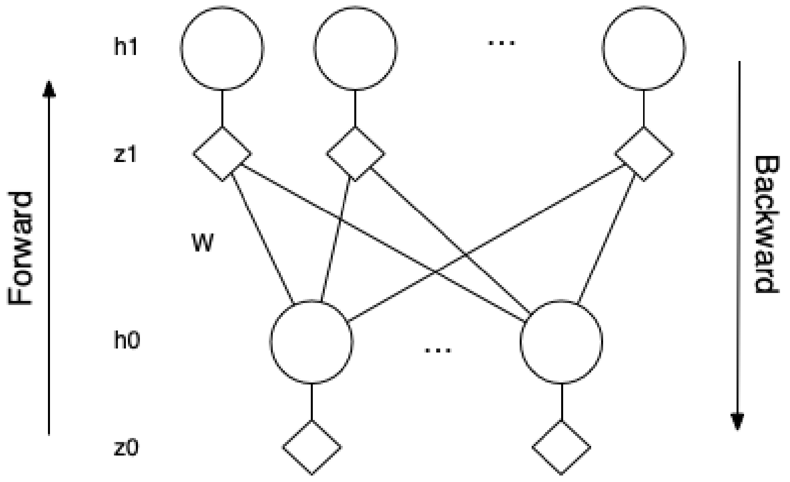
Parallel strategies

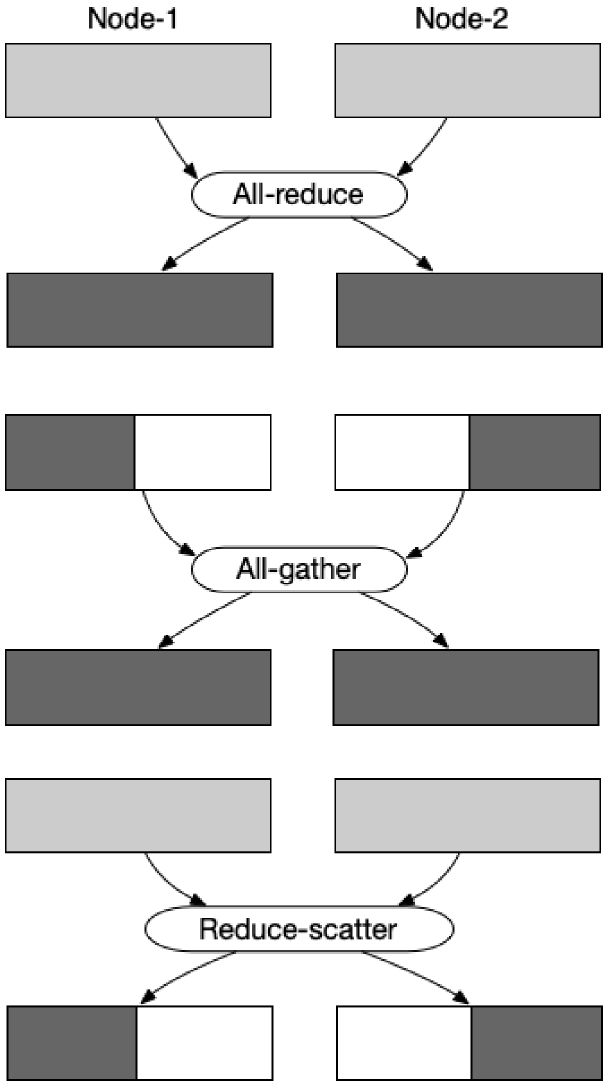
- Why bother?
- Three parallel paradigms
  - Data parallel: ZeRO
  - Model parallel
    - Tensor parallel: Megatron-LM
    - Pipeline parallel: GPipe
- Combined implementations: DeepSpeed/ColossalAI

Why bother?

- Too big to fit in single GPU memory
  - 175B: $\sim (2+2+3\times 4)\times 175 = 2800$GB for mixed-precision training
    - Parameter & gradient(FP16): parameter(W), gradient(g, $\frac{\partial L}{\partial W}$)
    - Optimizer State(FP32): parameter, momentum(m), variance(v)
    - Activation(FP16): $2\times(1+4+1)\times d\times B\times T\times L$
  - A100 Spec:
    - GPU memory: 80GB
    - GPU memory bandwidth: 2039GB/s; NVLink: 600GB/s; PCIe 4.0: 64GB/s
    - TF32: 156TFlops
- Speedup
  - Scales linearly with # of GPU cores?

Basics on Forward & Backward and Parallel Ops





Forward:

$$h_0 = \sigma(z_0)$$
$$z_1 = W^T h_0 + b$$
$$h_1 = \sigma(z_1)$$

Backward:

$$\left(\frac{\partial L}{\partial h_1}, W\right) \rightarrow \frac{\partial L}{\partial h_0}$$
$$\left(\frac{\partial L}{\partial h_1}, h_0\right) \rightarrow \Delta \frac{\partial L}{\partial W}$$
$$m \leftarrow \beta_1 m + (1 - \beta_1) \frac{\partial L}{\partial W}$$
$$v \leftarrow \beta_2 v + (1 - \beta_2) \left(\frac{\partial L}{\partial W}\right)^2$$
$$W \leftarrow W - \frac{\alpha}{\sqrt{\hat{v}+\epsilon}} \hat{m}$$

Data parallel: from DDP to FSDP(ZeRO)

- *Pesudo code* for DDP and FSDP

```
# forward pass :
for layer_i in layers:
  forward pass for layer_i
# backward pass :
for layer_i in layers:
  backward pass for layer_i
  full: all-reduce gradients for layer_i
  full: update momentum & variance
  full: update weights
```

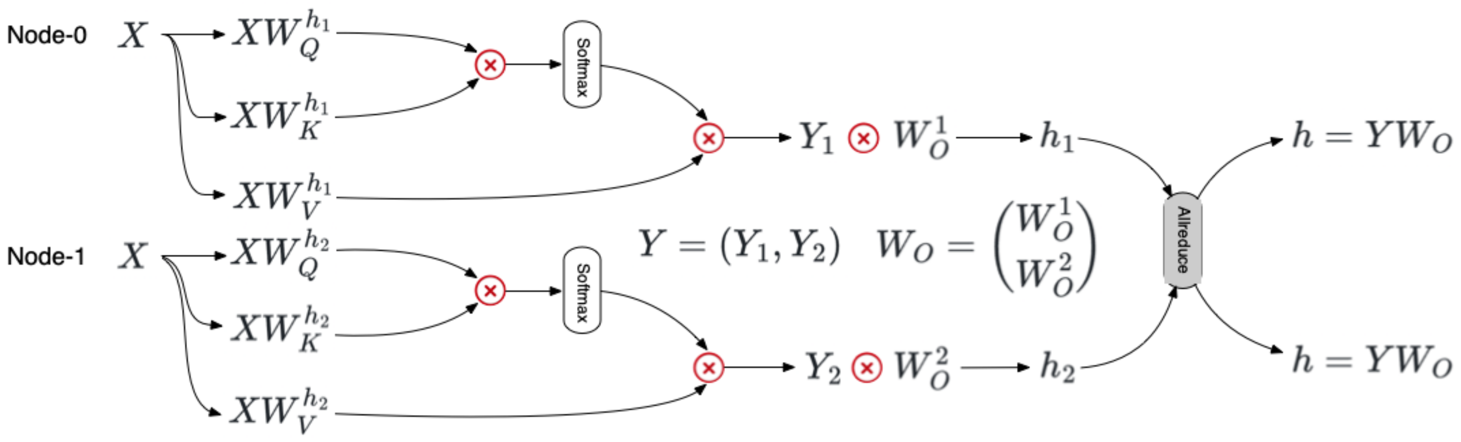
```
# forward pass :
for layer_i in layers:
  all-gather full weights for layer_i
  forward pass for layer_i
  discard full weights for layer_i
# backward pass:
for layer_i in layers:
  all-gather full weights for layer_i
  backward pass for layer_i
  discard full weights for layer_i
  part: reduce-scatter gradients for layer_i
  part: update momentum & variance
  part: update weights
```

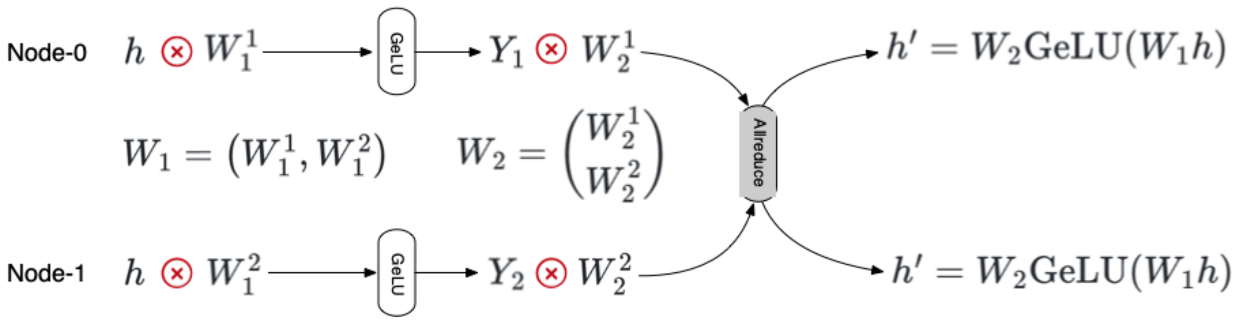
- Advantages of ZeRO
  - Parameter & gradient and optimizer states evenly shard to N nodes
  - Computation and communication overlaps

ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, Microsoft, 2019

Tensor parallel: Megatron-LM

- W_Q, W_K, W_V partition by head(col)
- W_O partition by row
- W_1 partition by col, W_2 by row
- Backward Allreduce for gradient



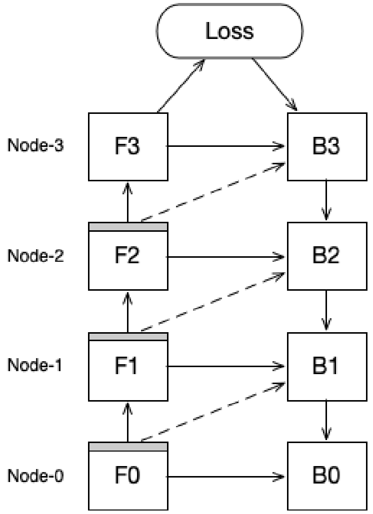


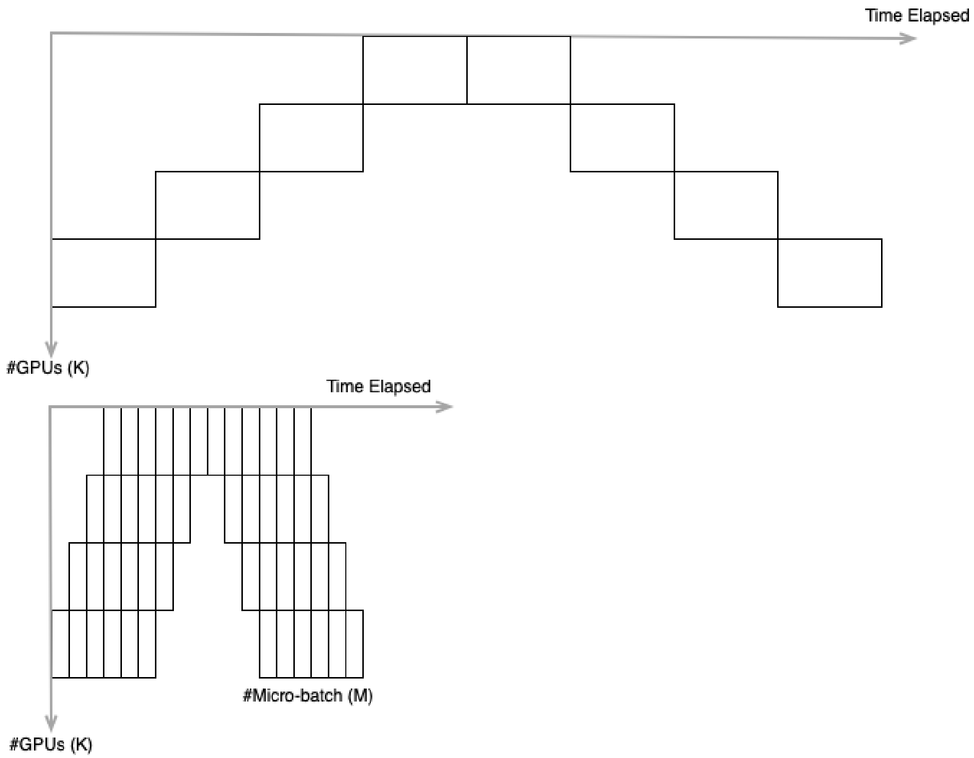
Performance comparison of ZeRO and Megatron-LM

- Experiment hardware:
  - Megatron-LM: 32 DGX-2H servers: 512 V100, 32GB GPUs
  - ZeRO: 25 DGX-2 servers: 400 V100, 32GB GPUs
- TFlops Results:
  - Megatron-LM: 15.1 PFlops
    - 76% scaling efficiency for single GPU 39TFlops(30% of peak Flops)
  - ZeRO: 15 PFlops, with fewer GPU cores

Pipeline parallel: GPipe

- Layer-wise model partition
- Re-materializaion: output activation stored and communicated
- Pipeline reduce bubble ratio from $\frac{K-1}{K}$ to $\frac{K-1}{M+K-1}$





GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism, Google, 2018

Combined implementations: Megatron-DeepSpeed/ColossalAI

Megatron-DeepSpeed

- Data/Model parallel supported(3D)
- ZeRO-Offload
- Sparse attention
- 1-bit Adam and 0/1 Adam
- MoE specific parallelism
- RLHF Demo: deepspeed-chat

ColossalAI

- Data/Model parallel supported(3D)
- 3D Tensor parallelism
- ZeRO-Offload: model data supported
- MoE specific parallelism
- RLHF Demo: ColossalChat

Colossal-AI: A Unified Deep Learning System For Large-Scale Parallel Training, HPC-AI, 2021
Megatron-DeepSpeed

End of Parallel strategies

Result & Evaluations

Finetuning

- Target and issues
- Instruct finetuning
- Finetuning for specific task
- Parameter efficient finetuning

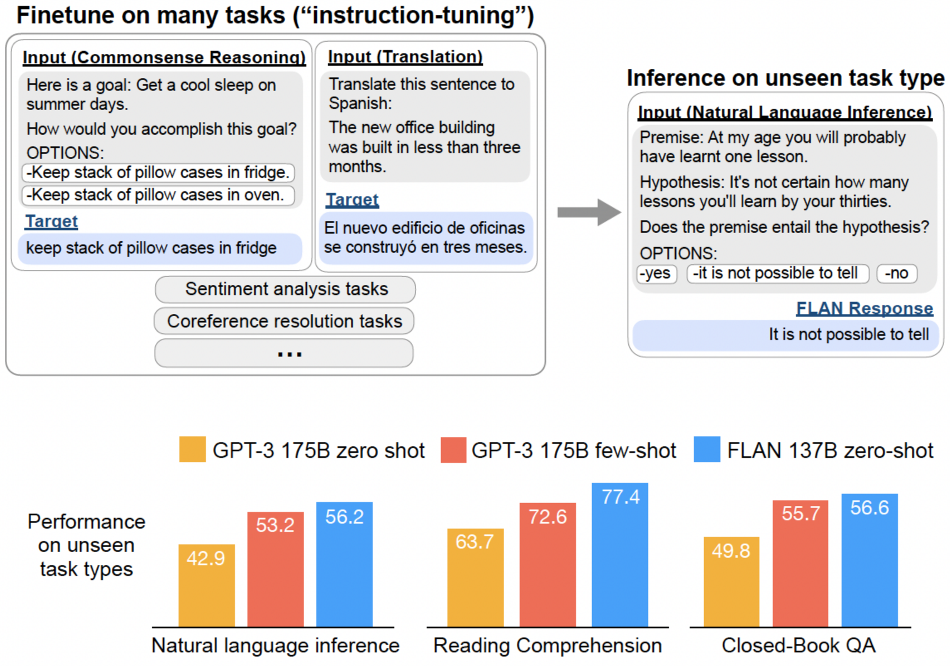
Target and issues

- Target
  - From pattern completion to real world tasks
- Issues
  - Instruction following
  - Hallucination
  - Toxicity and ethics
  - Securities

Instruct Finetuning

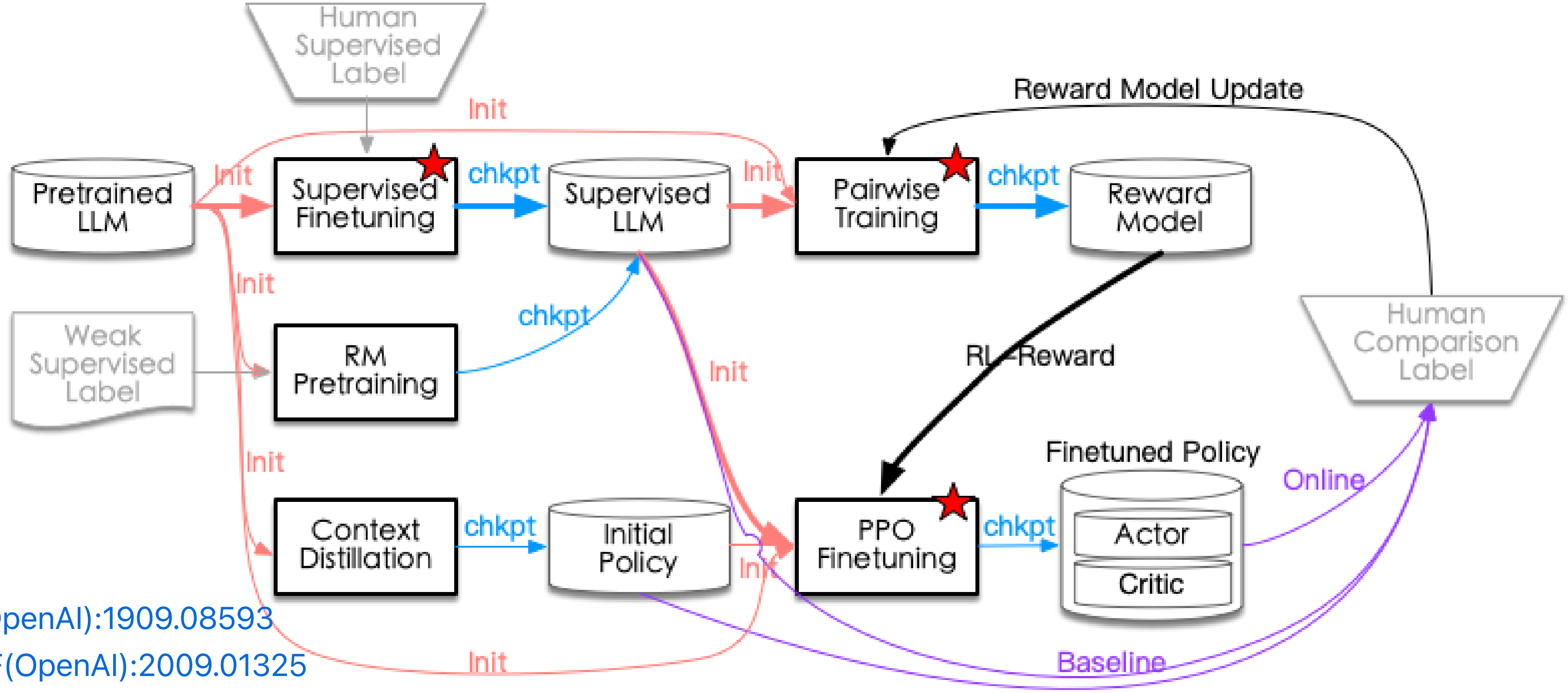
Key to success

- Number of finetuning datasets:
  - scaling from 62 text datasets to 18K
- Model scale: 137B LaMDA-PT
- Natural language instructions



Finetuned Language Models Are Zero-Shot Learners, 2021, Google
Scaling Instruction-Finetuned Language Models, 2022, Google

Finetuning for specific task



Finetune from HF(OpenAI):1909.08593
Summarize from HF(OpenAI):2009.01325
HHHA(Anthropic):2112.00861
InstructGPT(OpenAI):2203.02155
HHA(Anthropic):2204.05862
Sparrow(Deepmind):2209.14375

Problems from Supervised Finetuning(SFT)

- Learning only the task format and the way to response for the format
- Knowledge labeled but not in the LLM leads to more hallucination
- Knowledge in the LLM but labeled as `don't know` leads to withhold information

What we want from finetuning:

> outputs its(LLM's) state of knowledge with the correct amount of hedging and expressing its uncertainty

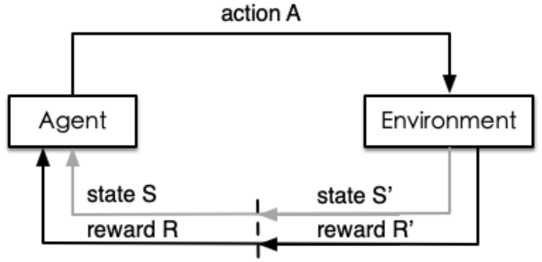
Advance of RL to SFT for truthfullness

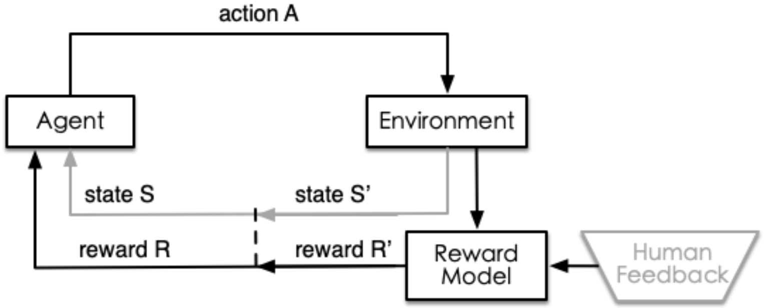
- LLMs know what they know
  - Calibrated probability, uncertainty
- RLHF can leverage the self-awareness
  - Design reward function: `correct answer=1`, `don't know=0` and `wrong answer=-4`
  - RL learn optimal threshold of probability to maximize the reward
- No oracle for the correctness, delegate to Reward Model
  - Reward model: **relative criteria** trained by pairwise loss from human feedback
  - Open problem: true probabilites of everything?
  - Open problem: go beyond things that labelers can easily do
    - Verification is easier than generation

Anthropic: 2207.05221
Talk of John Schulman @ EECS Berkeley, 2023/4/24

More on Reinforcement Learning

- Catalog of algorithms(**PPO belongs**)
  - World model or **model free**
  - Value-based or **policy-based(actor-critic)**
  - MC or **TD bootstrapping**
  - Off-policy or **on-policy**
  - Deterministic or **stochastic** policy
- Design consideration
  - **Sample efficiency**: Off-policy > On-policy
  - **Stability & Convergence**: Deadly Triad issue
  - **Explore & Exploit**: Random at episode beginning
  - **Bias & Variance**: Advantage Function

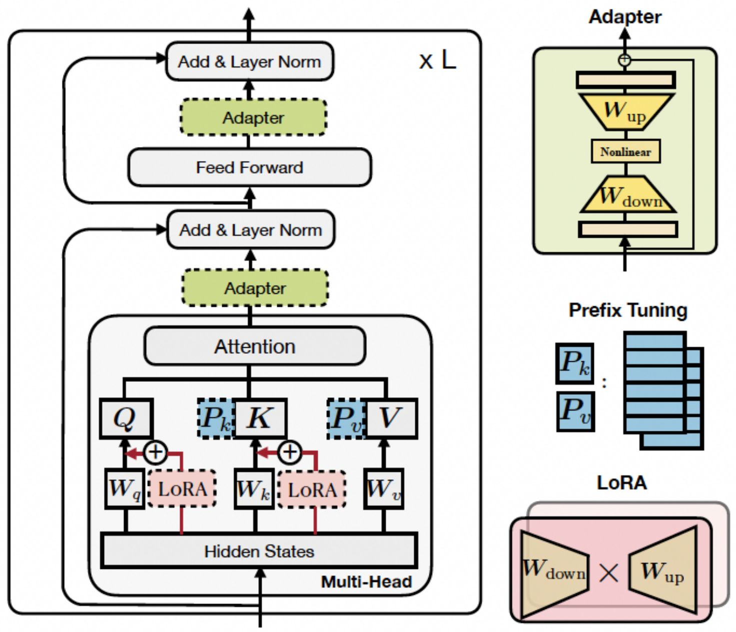


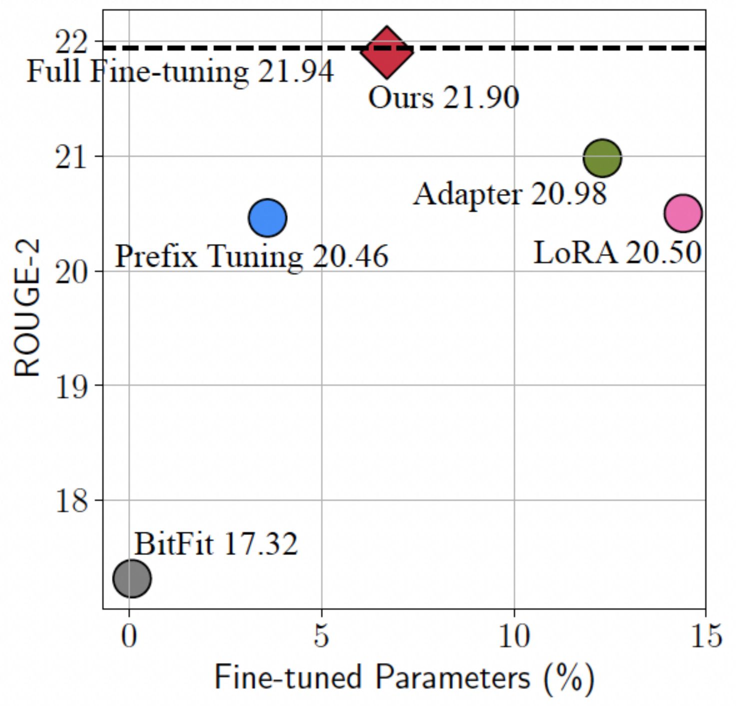


Parameter Efficient Finetuning

- Design aspects
- Primary implementations
  - Adapter
  - Prefix-tuning
  - LoRA

Methods and performance on Summerization task





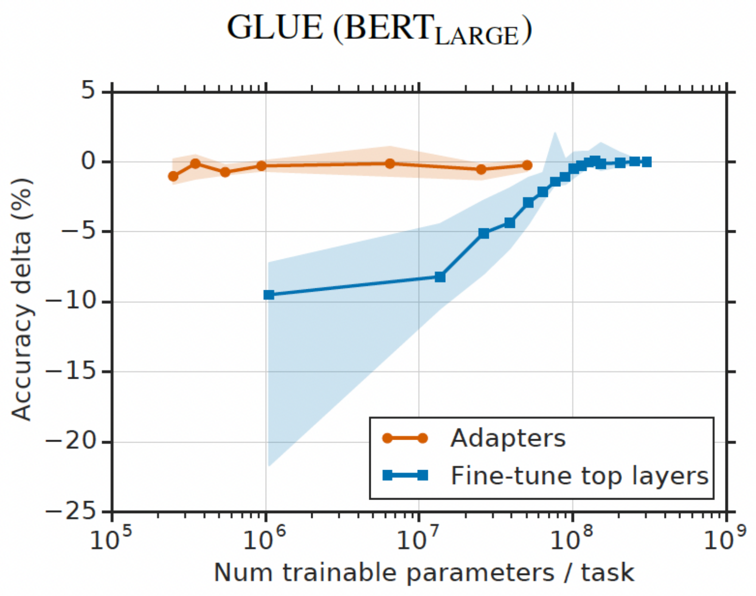
PEFT illustration and performance comparison (Source: Junxian He, et.al)

Design aspects

- **Finetuned Modules**:
  - Attention-key/value matrix: LoRA(Q/V)
  - Attention-head: Prefix-Tuning(K/V)
  - Attention: Adapter
  - After FFN: Adapter
- **Other aspects**:
  - Multi-task consideration
  - Task related head

Adapter

- Implementation & training notes
  - $h \leftarrow h + f(hW_{\text{down}})W_{\text{up}}$
  - Parameter scale: $2 \times L \times (r \times d + r + d), r \ll d$
  - Adapt after FFN sub-layer works too
- Results
  - Finetune BERT for 26 classification Tasks
    - 3.6% parameters for 0.4% GLUE performance gap
  - Ablation: fewer layers adapted -> worser performance



Adapter: Parameter-Efficient Transfer Learning for NLP, Google, 2019

Prefix-Tuning

- Implementation
  - $\text{head} = \text{Attn}(XW_Q, [P_K; XW_K], [P_V; XW_V])$
  - Reparameterization for finetuning stability:
    - $[P_K, P_V] = \text{MLP}_\theta(P^E)$
    - P^E: prefix embedding
- Parameter scale:
  - Vanilla: $|P| \times d \times 2 \times L$
  - Reparameteration: $|P| \times d + d \times H + H \times d \times 2 \times L$

PrefixTuning: Optimizing Continuous Prompts for Generation, Stanford, 2021

More on Prefix-Tuning: Training and scaling

- Training
  - Initialization:
    - Real/high frequency words activation
    - *Task relevant words | Classification labels*
  - LM Head: Next Token/Class Label
- Results & discussion
  - Finetuning $0.1\% \sim 3\%$ parameters, comparable or better performance
  - Optimal prefix length varies: longer for more complex tasks
  - Reparameterization works task-dependently

PrefixTuning: Optimizing Continuous Prompts for Generation, Stanford, 2021
PromptTuning: The Power of Scale for Parameter-Efficient Prompt Tuning, Google, 2021
P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks, Tsinghua, 2021

LoRA

- Implementation & training notes
  - Transformer: $W = W_0 + (BA)^T, A \in R^{r \times d}, B \in R^{d_h \times r}, r \ll \min\{d_h, d\}$
  - W_Q and W_V considered, parameter scale: $2 \times 2 \times d \times r \times L$
  - Modularized: `Embedding`, `Linear`, `MergedLinear`, `Conv2D`
  - Initialization: A kaiming-random, B zeros
  - Weight merged for inference efficiency
- Results
  - For 175B GPT-3 finetuning: 0.01% parameters , on par or better results
  - No additional inference computation and latency
  - Additivity for **finetuning merge** and **incremental update**

LoRA: Low-Rank Adaptation of Large Language Models, Microsoft, 2021

More on LoRA: which part to update & rank settings

	# of Trainable Parameters = 18M						
Weight Type Rank r	W_q 8	W_k 8	W_v 8	W_o 8	W_q, W_k 4	W_q, W_v 4	W_q, W_k, W_v, W_o 2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

LoRA: Low-Rank Adaptation of Large Language Models, Microsoft, 2021

End of Parameter Efficient Finetuning

Steering the decoding process of LLM

- Decoding strategies
- Prompt enginneering

Decoding strategies

- Temperature in decoding: $p(w_i|w_{<i}) = \frac{\exp(o_i/T)}{\sum_j \exp(o_j/T)}$, o_i logits from LLM
- Maximal Likelihood Search
  - Greedy search: $w_i = \arg\max p(w_i|w_{<i})$, eq. to $T = 0$
  - Beam search: $w_i \in \text{TopN}\ p(w_i|w_{i-1}, w_{<i-1})p(w_{i-1}|w_{<i-1})$
- Sampling
  - top-K sampling: $w_i = \text{sample TopK}\ p(w_i|w_{<i})$
  - top-p(Nucleus) sampling: $w_i = \text{sample TopK}_i\ \sum_{w_i<K_i} p(w_i|w_{<i}) \geq p$
  - Repetition penalized sampling
- Guided decoding
  - $score(x_{t+1}, b_t) = score(b_t) + \log p(x_{t+1}) + \sum_i \alpha_i f_i(x_{t+1})$

Prompt engineering

- Single round interactive
  - Instruction/Zero-shot Prompt
  - Few-shot Promp
  - In-context Learning(Prompt)
  - Chain-of-Thought
- Multi rounds interactive
  - MRKL: `[Thought/Action/Action Input/Observation]+`, no demo, access tools
  - Self-ask: `Followup question? [Question/Answer]+ Final Answer`, demo
  - ReACT: `[Thought/Action/Observation]+`, demo, access tools

More on In-context Learning

What matters?

- Examples template(instruct/CoT) & order: yes
- Label space & Input distribution(diversity): yes
- Exact `{Question, Answer}` pair: no/yes

Why it works?

- Interpretation from Topic Model: $P(o|p) = \int_z P(o|z,p)P(z|p)dz$
- Induction head: `[a][b] ... [a] -> [b]`

A Mathematical Framework for Transformer Circuits, 2021, Anthropic
How does in-context learning work?, 2022, Stanford
Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?, 2022, Meta
Larger language models do in-context learning differently, 2023, Google

Augmentation and Plugins

- Augmented Language Models
- Automatic prompting
- Plugins

Augmented Language Models

- Problems in vanilla LLMs
- Two augment aspects
    - Retrieval augmented language models
    - Tool augmented language models

Problems in vailla LLMs

Retrieval augmented language models

Tool augmented language models

End of Augmented Language Models

Automatic prompting

Plugins

- Plugins ecosystem
- Primary implementations
  - Langchain Agent/Tool
  - ChatGPT Plugins
  - Fixie
  - Other paradigm proposal

Plugins ecosystem

- Tool as a service
- From SEO to LMO
- Orchestration by LLM

Langchain Agent/Tool

ChatGPT Plugins

Fixie

Other paradigm proposal

End of Plugins

Thanks & QA?