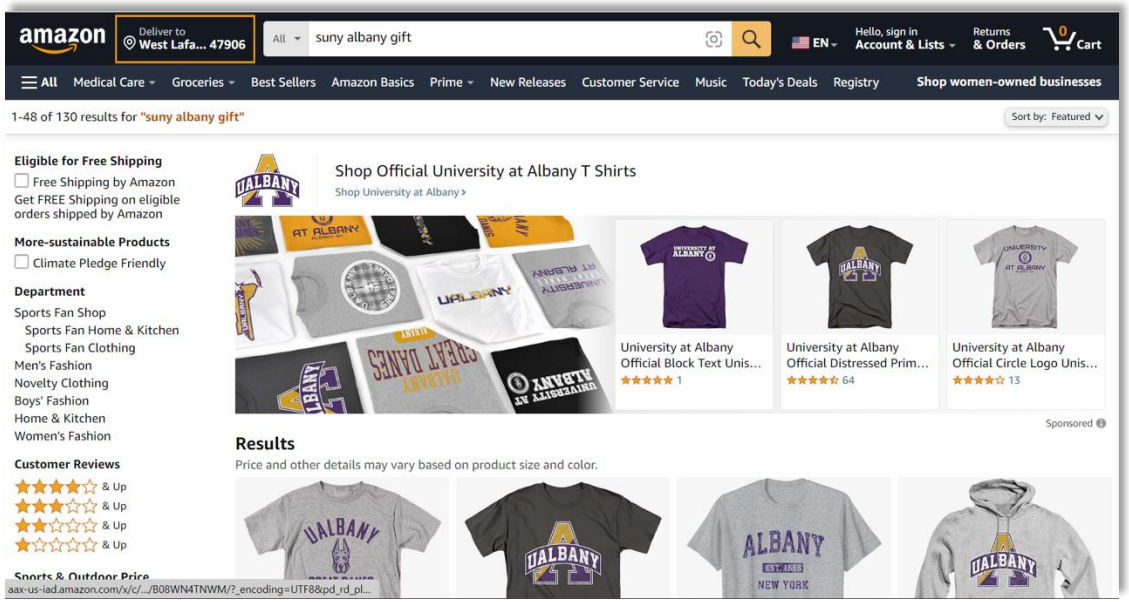
Pre-trained Language Model with Limited Resources

Haoyu Wang[♤], Yaqing Wang[♡], Tianci Liu[◊], Jing Gao[◊]

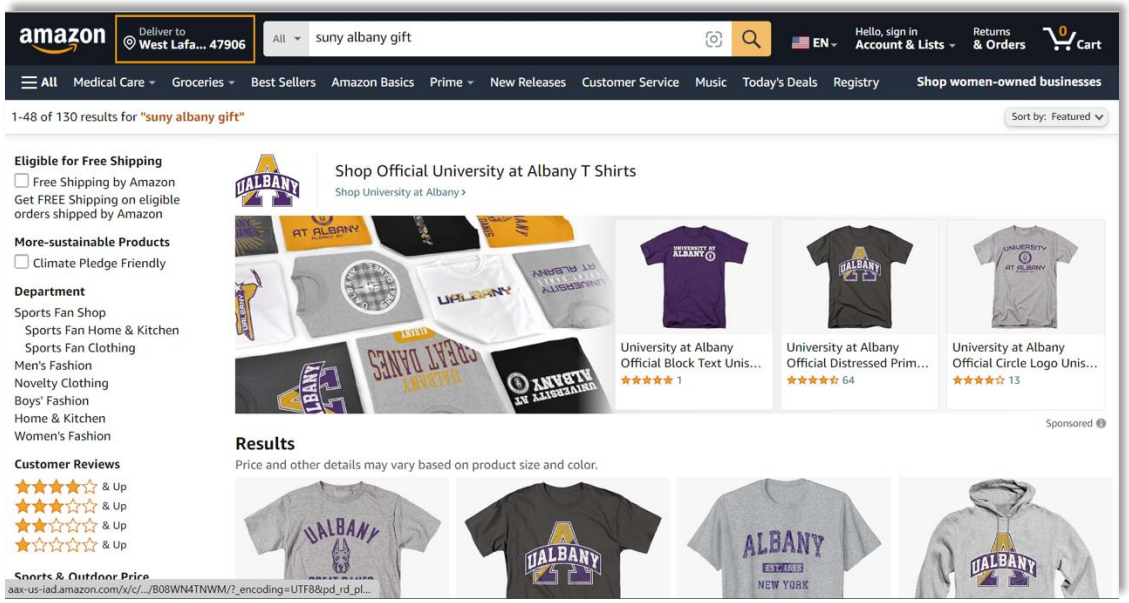
[♤]University at Albany, [♡]Google Deepmind, [◊]Purdue University

Pre-trained Language Model Systems

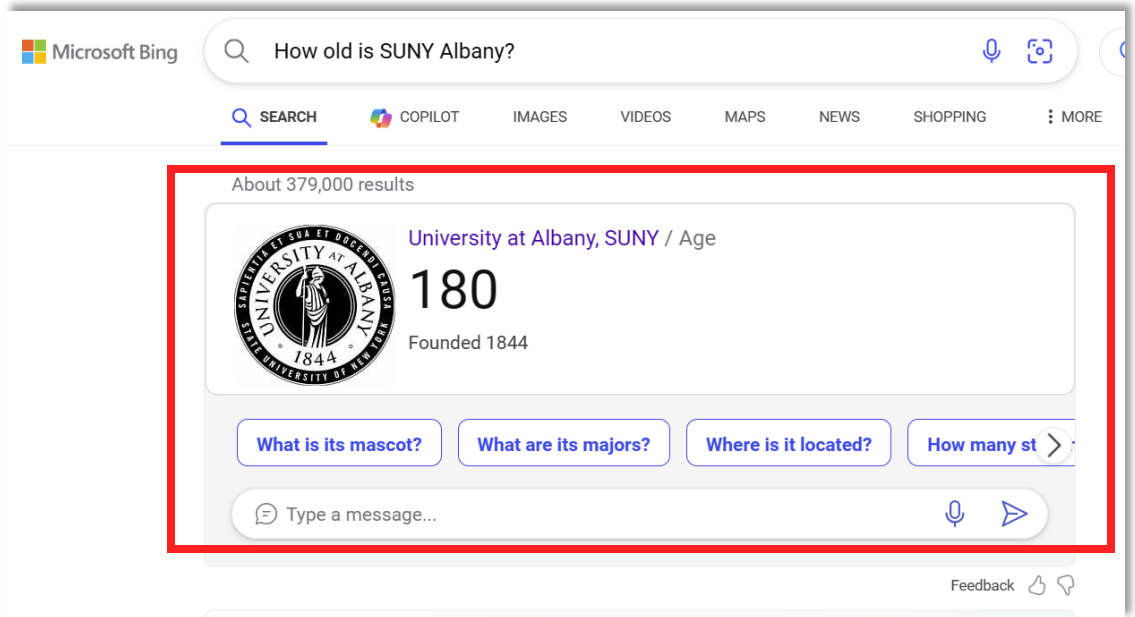


Amazon

Pre-trained Language Model Systems

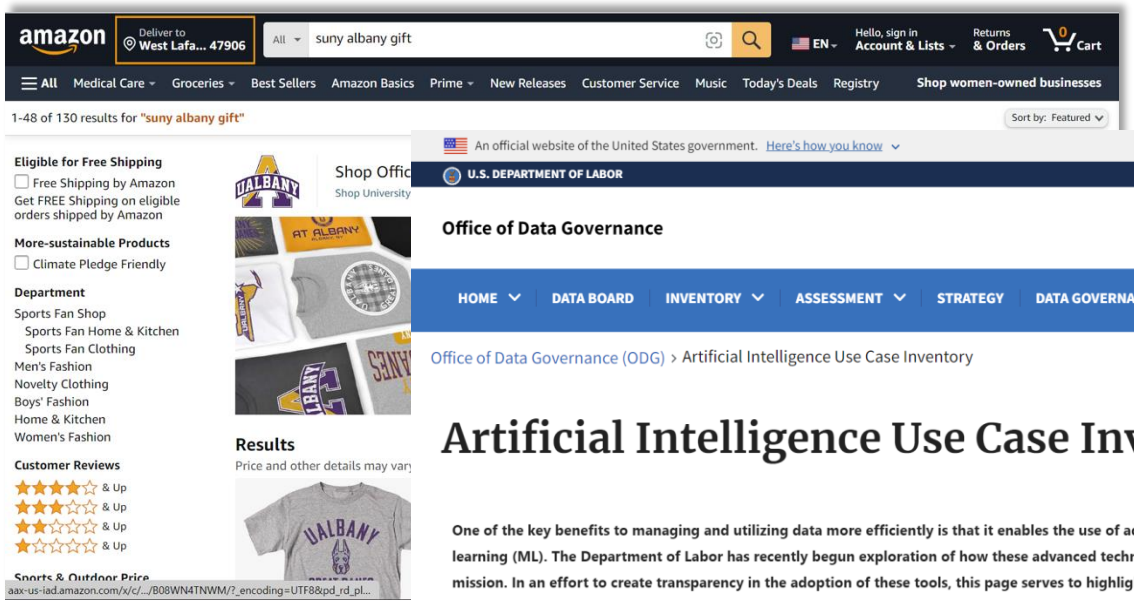


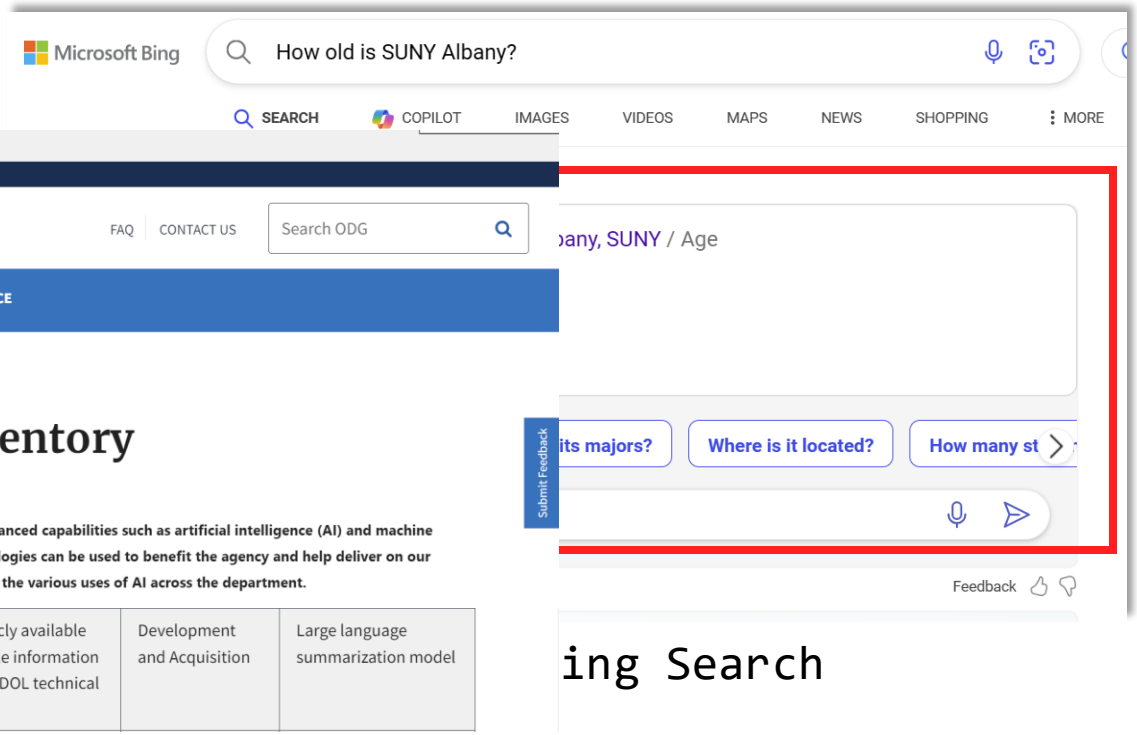
Amazon



Bing Search

Pre-trained Language Model Systems

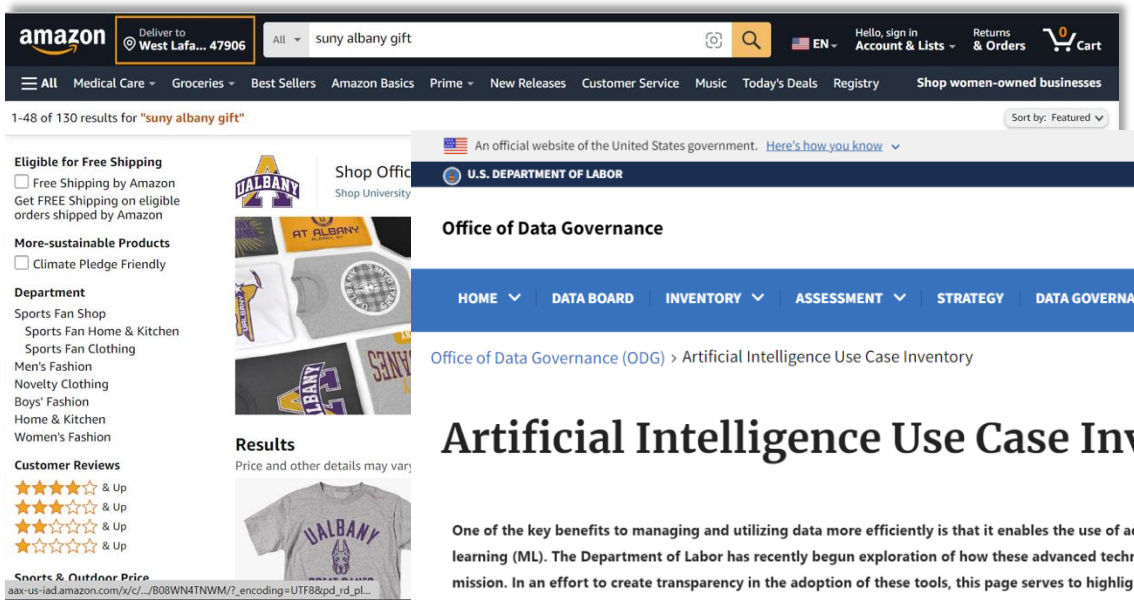


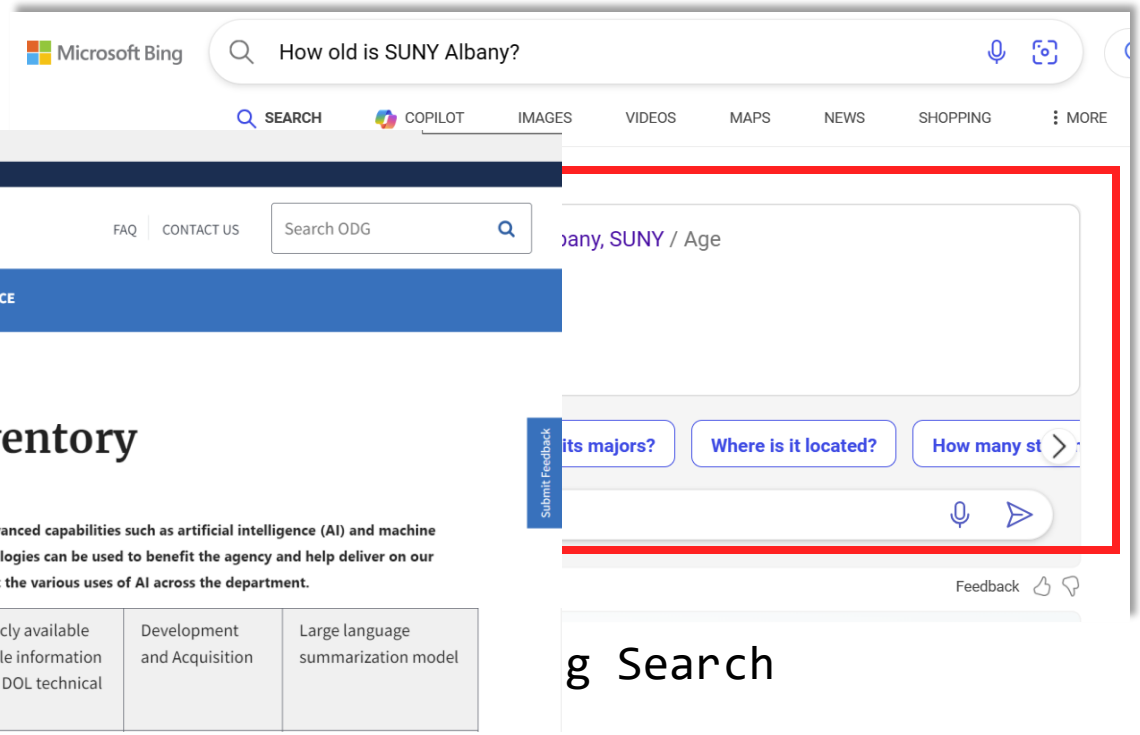


U.S. Department of Labor

ing Search

Pre-trained Language Model Systems





g Search

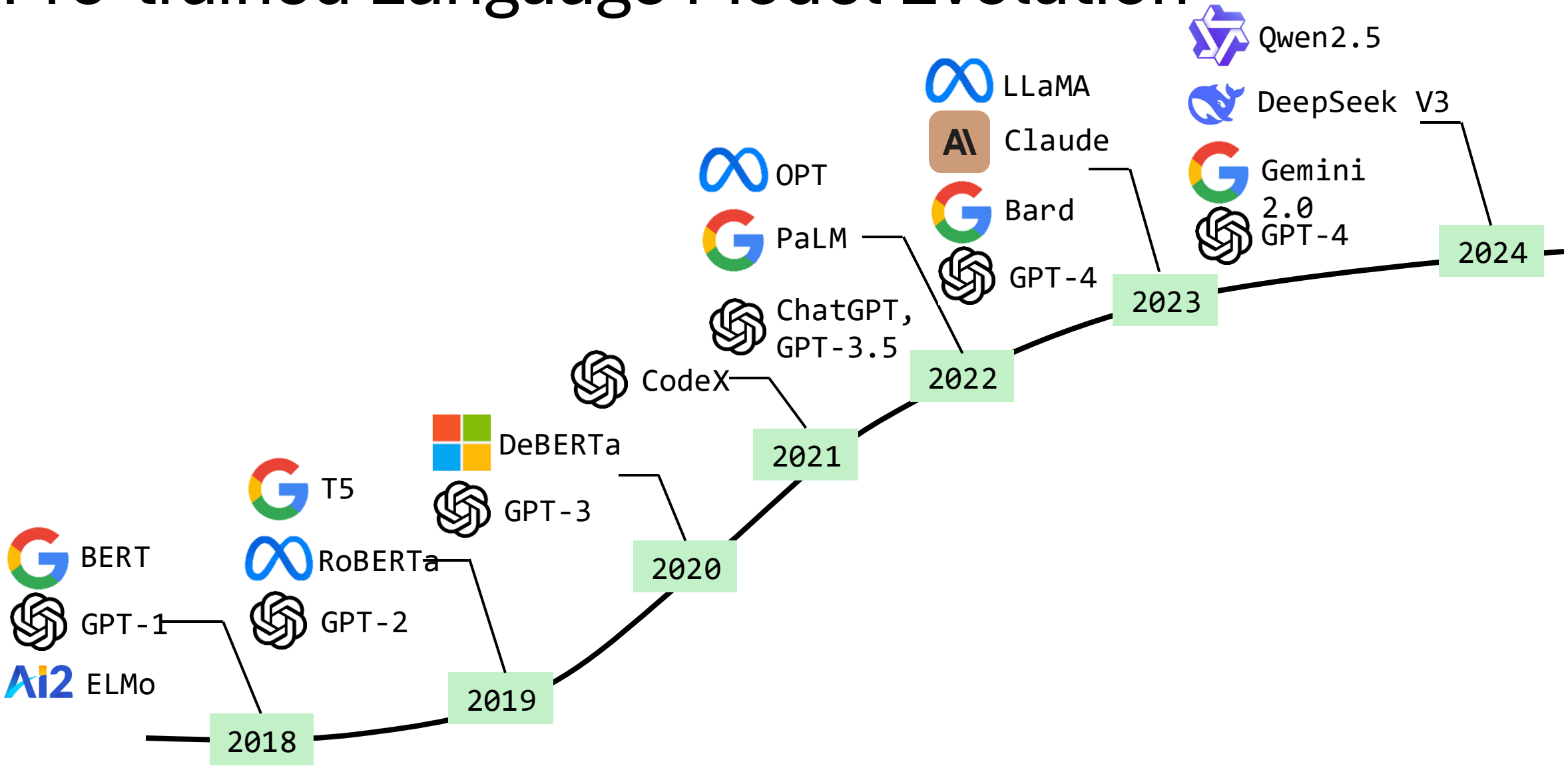
U.S. Department of Labor

Pre-trained Language Models have achieved remarkable performance in various real-world applications.

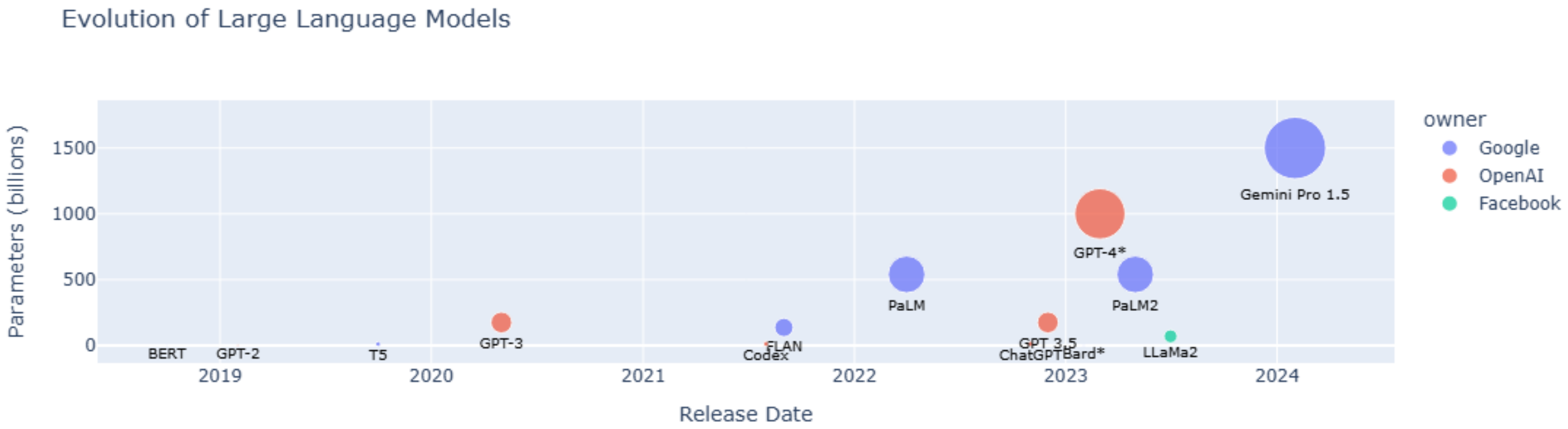
Many other use cases:

- Content generation
  - Writing outlines, descriptions, stories, summaries, scripts, emails.
- Natural language understanding
  - Grammar corrections, translations, sentiment analysis.
- Question answering and chatbots
  - Respond to queries and have a conversation.
- Analysis and optimization
  - Extract insights from text and workflow optimization.
- Software development
  - Code completion, debugging and explanation.

Pre-trained Language Model Evolution

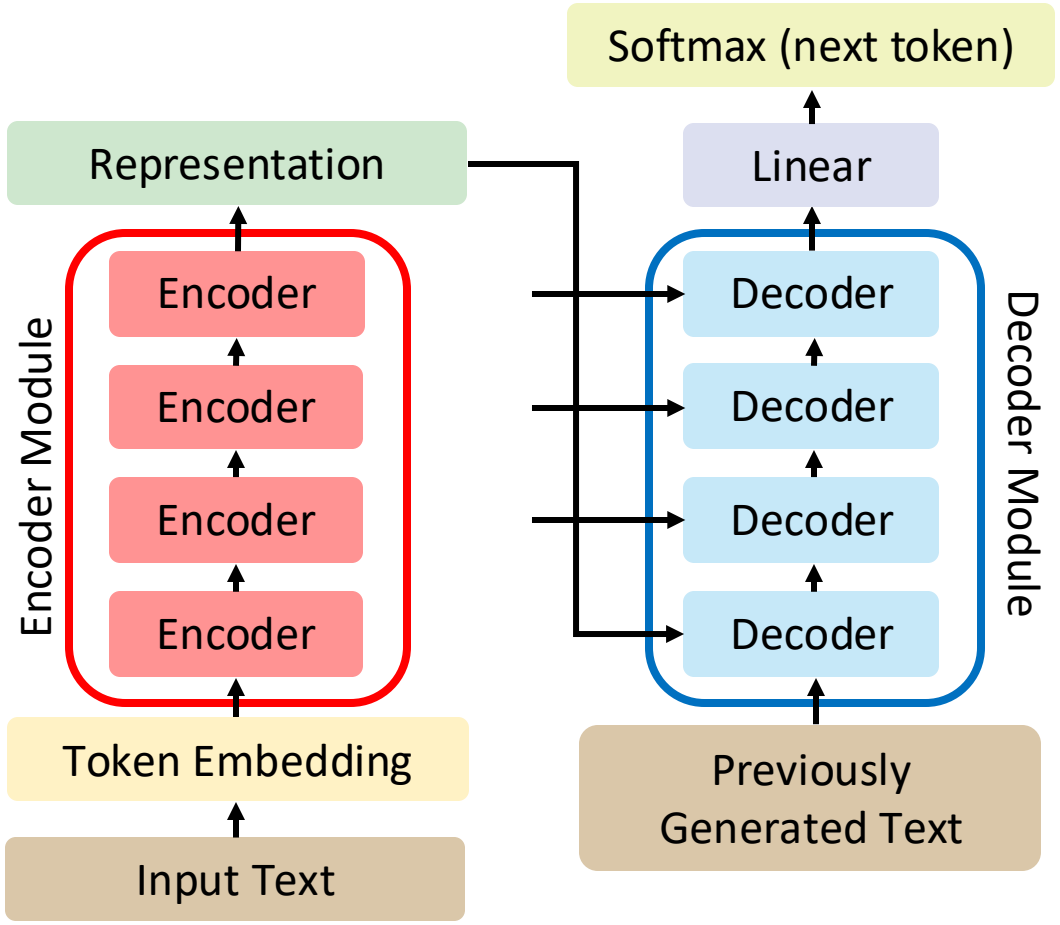


Pre-trained Language Model Evolution



Background of Pre-trained Language Models

- Architecture



Transformer architecture

Encoder Module

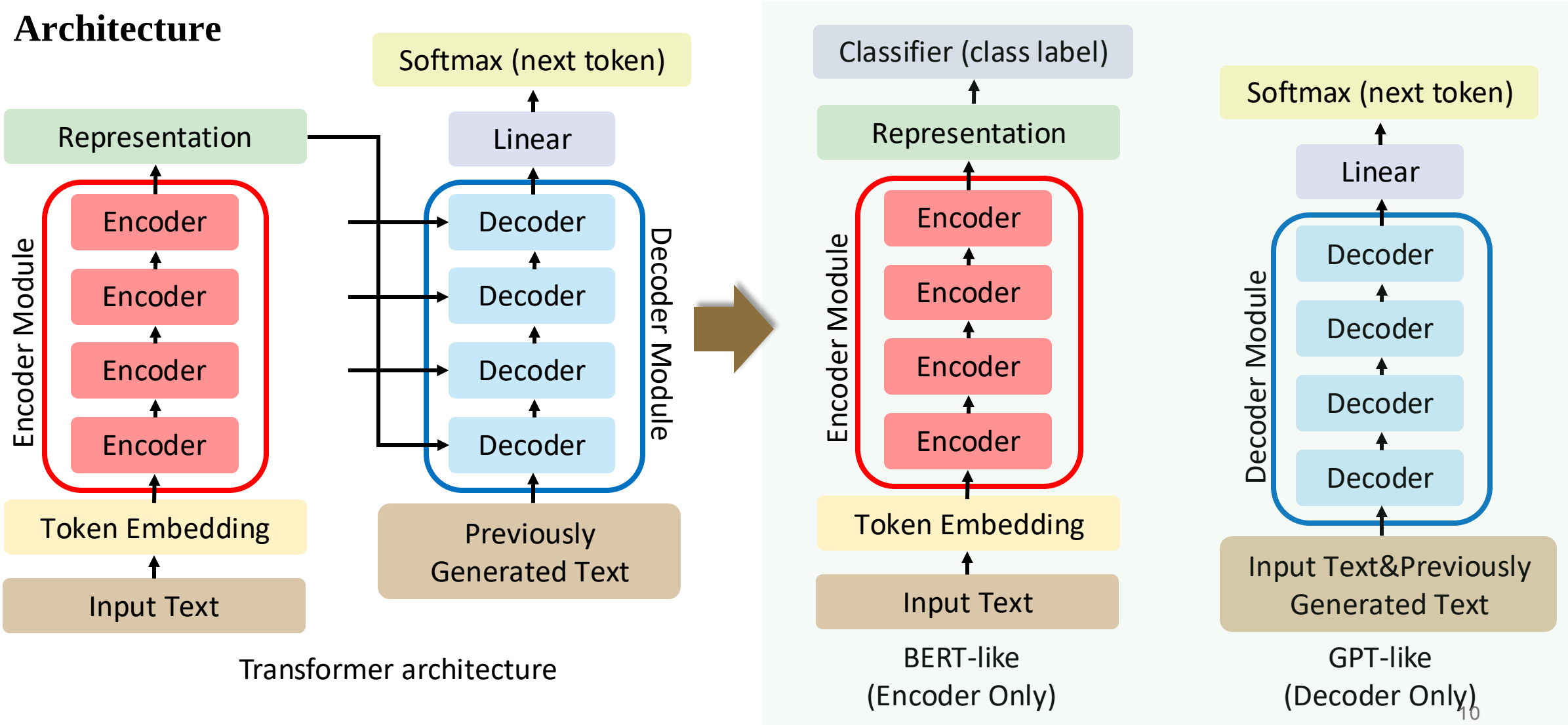
- **Input**: a sequence of text
- **Output**: representations corresponding to the tokens of the input text

Decoder Module

- **Input**: the output of the Encoder, the decoder module output at the previous time step
- **Output**: next token probability (autoregressive)

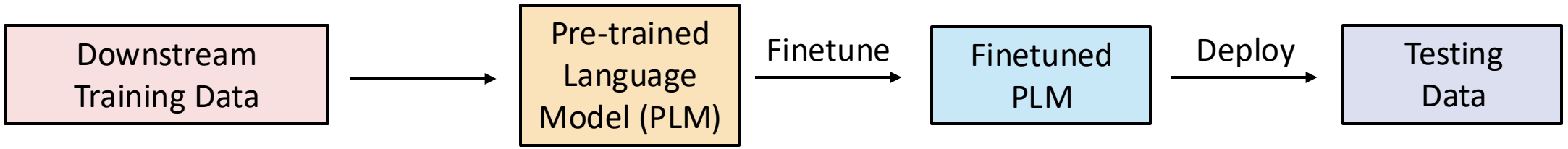
Credits: *Attention Is All You Need*

Background of Pre-trained Language Models

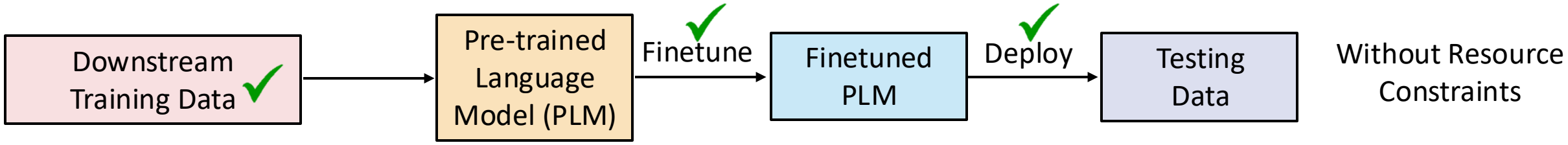


Background of Pre-trained Language Models

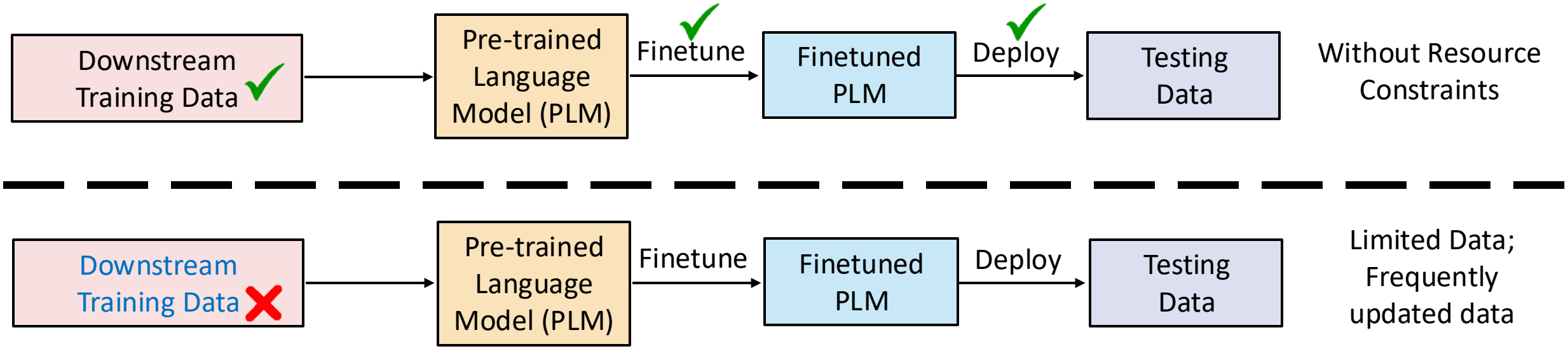
- Training paradigm



Challenges From Resource Constraints

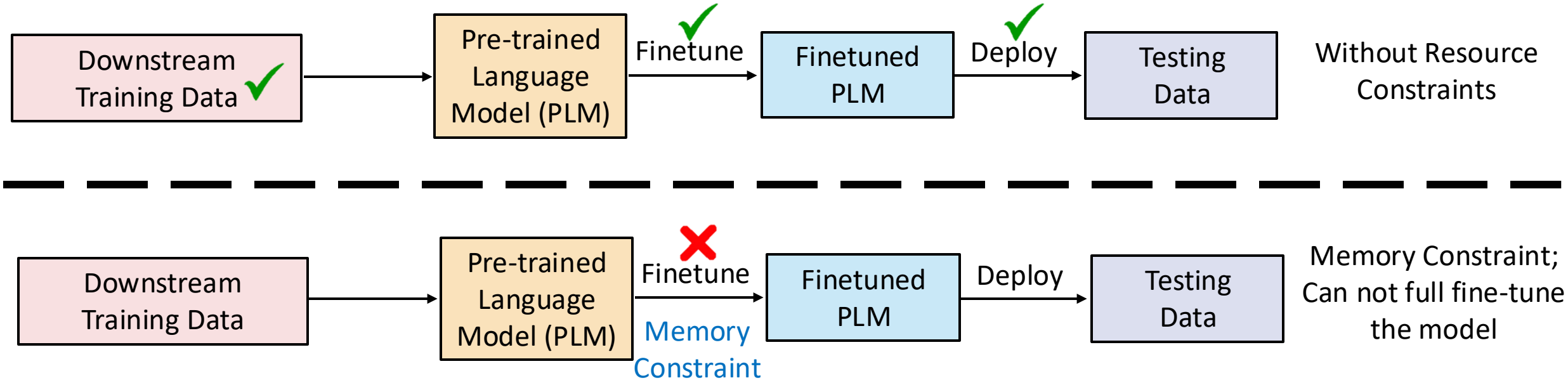


Challenges From Resource Constraints

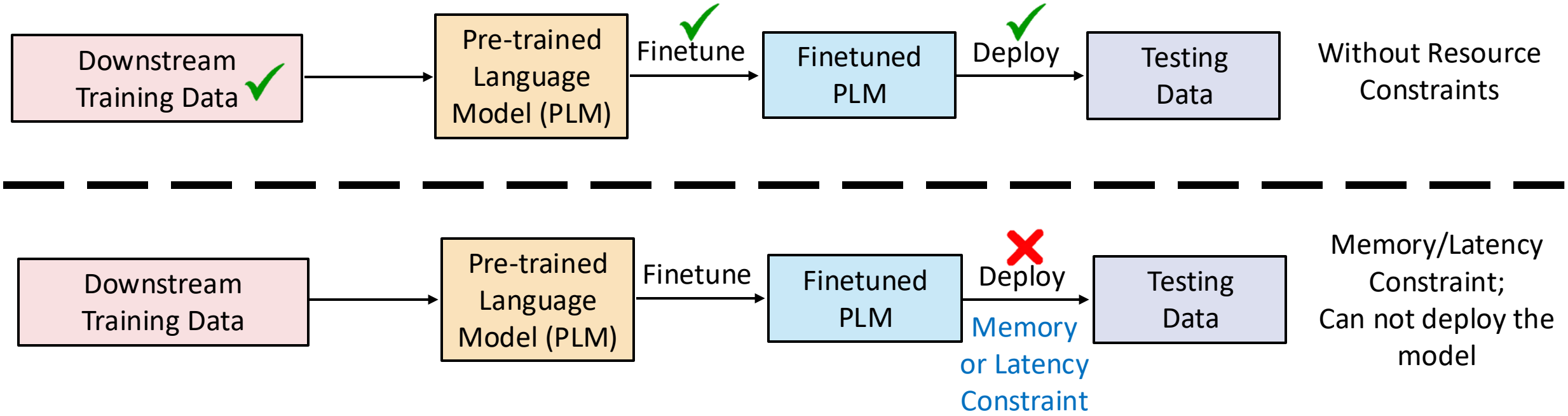


User privacy data;
Data that are expensive to collect;
Data that requires expert annotation

Challenges From Resource Constraints



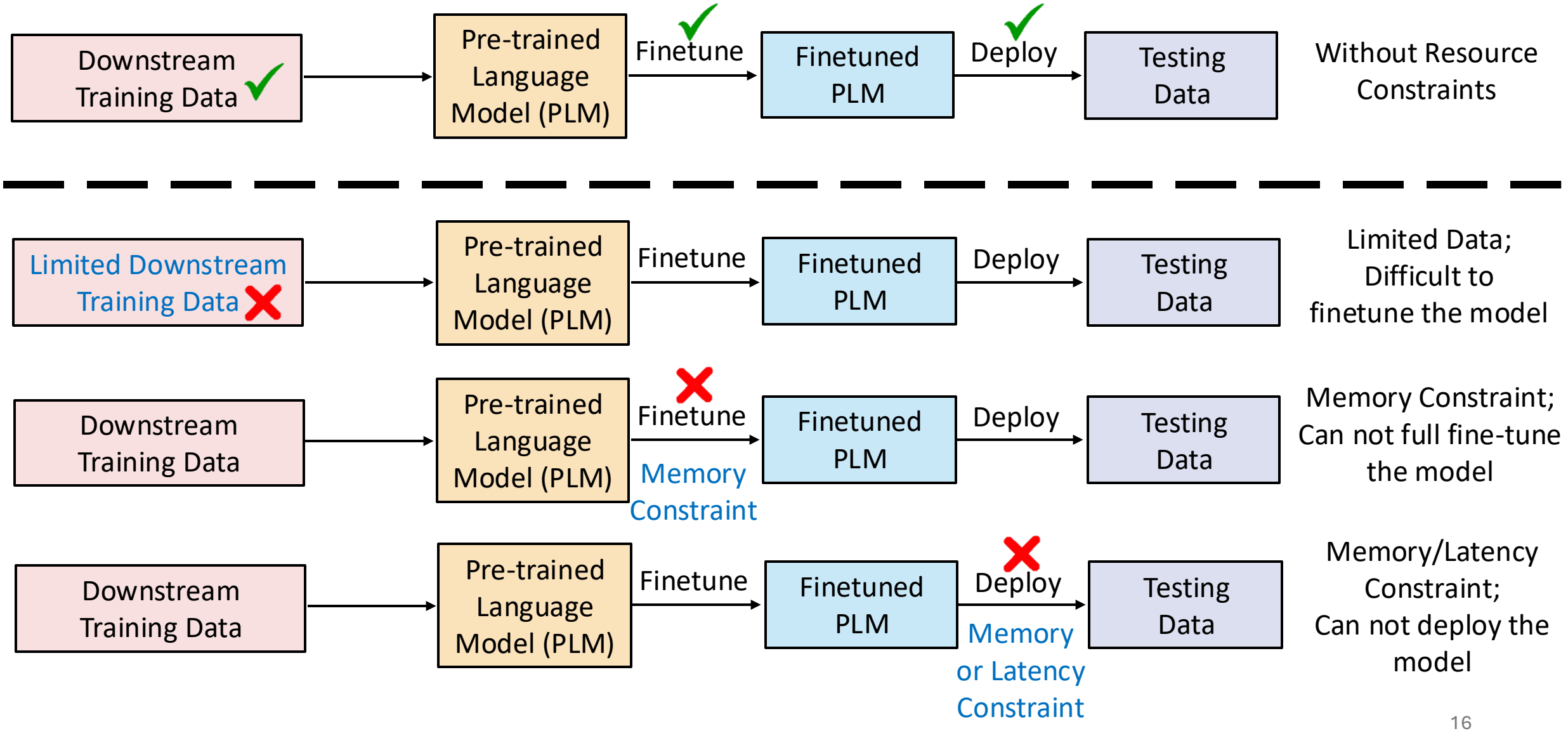
Challenges From Resource Constraints



Edge Devices

Real-time Systems

Challenges From Resource Constraints



Outline

- Model Efficiency
  - Parameter-efficient Fine-tuning
  - Model Compression
  - Inference Acceleration
- Data Efficiency
  - Zero/few-shot Learning
  - Knowledge Editing
  - Retrieval-augmented Generation

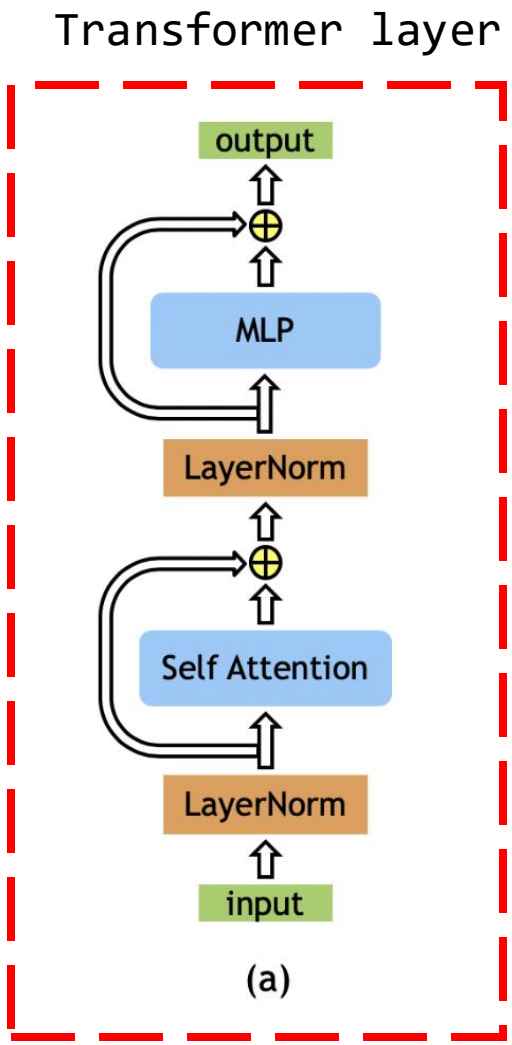
Outline

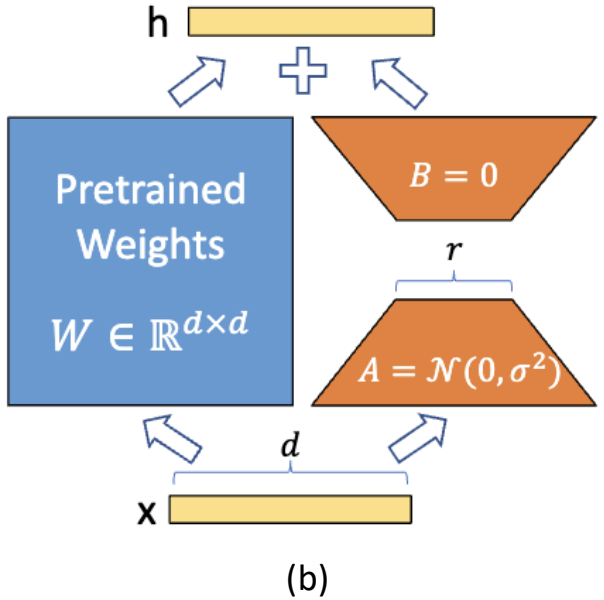
- Model Efficiency
  - Parameter-efficient Fine-tuning
  - Model Compression
  - Inference Acceleration
- Data Efficiency
  - Zero/few-shot Learning
  - Knowledge Editing
  - Retrieval-augmented Generation

Parameter-Efficient Fine-tuning

- Adapt pre-trained LLMs to downstream tasks without full fine-tuning.
  - Freeze pre-trained parameters.
  - Inject learnable **lightweight** modules only.

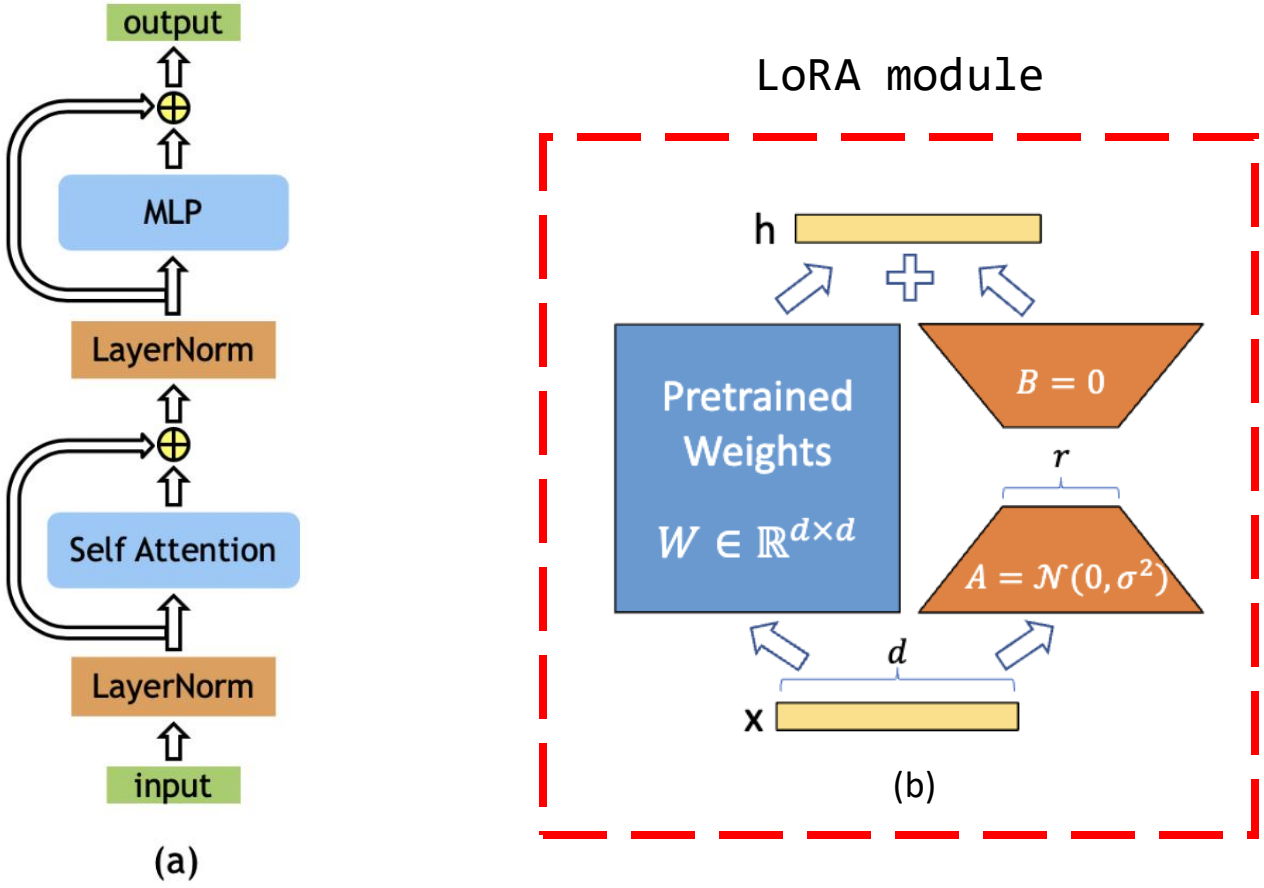
LoRA





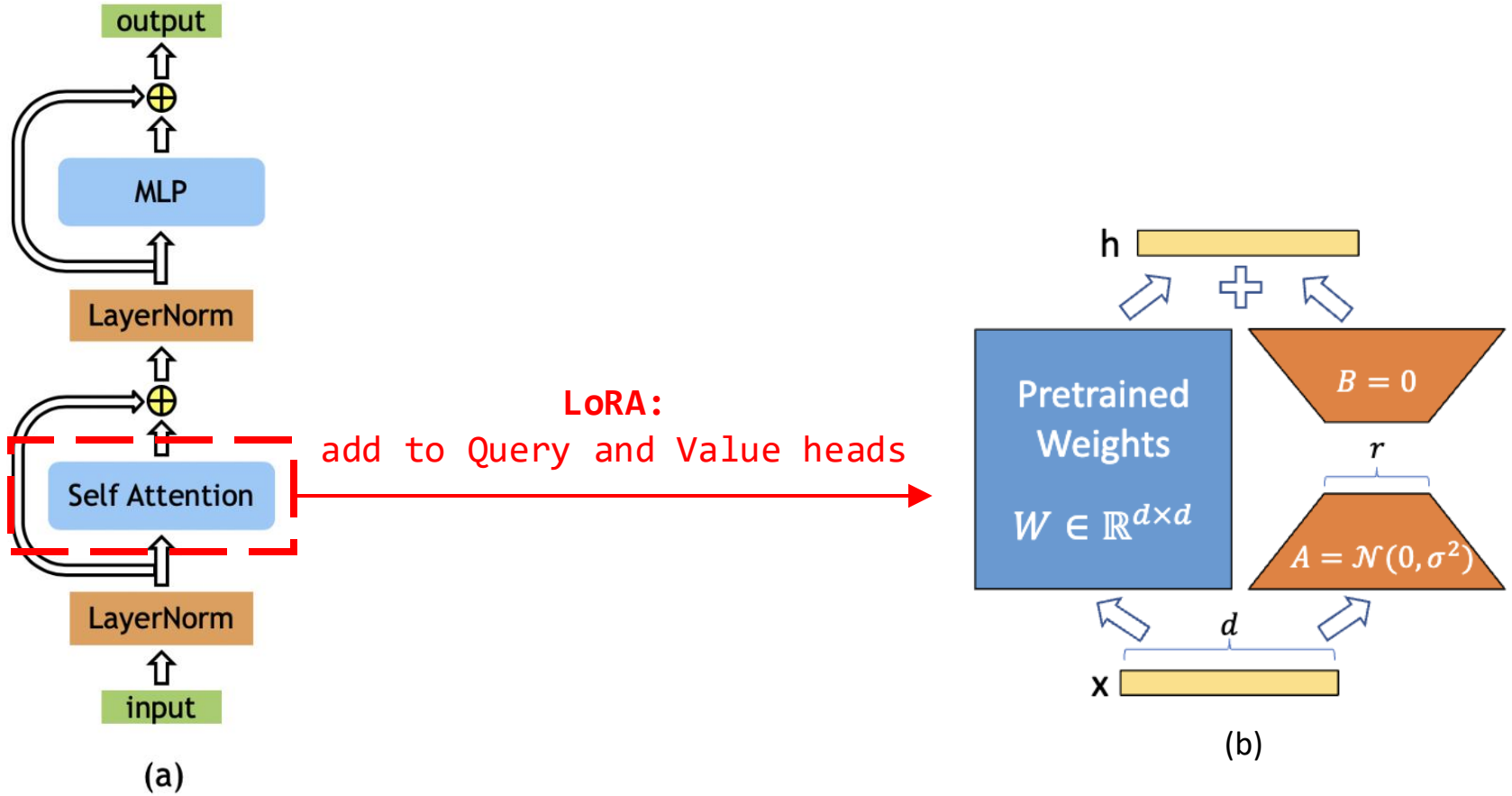
Hu et al. "*LoRA: Low-Rank Adaptation of Large Language Models.*" International Conference on Learning Representations (ICLR), 2022.

LoRA

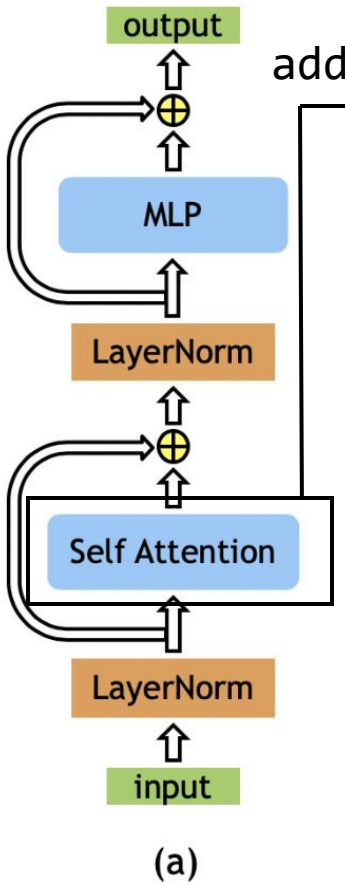


Hu, Edward J., et al. "*LoRA: Low-Rank Adaptation of Large Language Models.*" International Conference on Learning Representations (ICLR), 2022.

LoRA

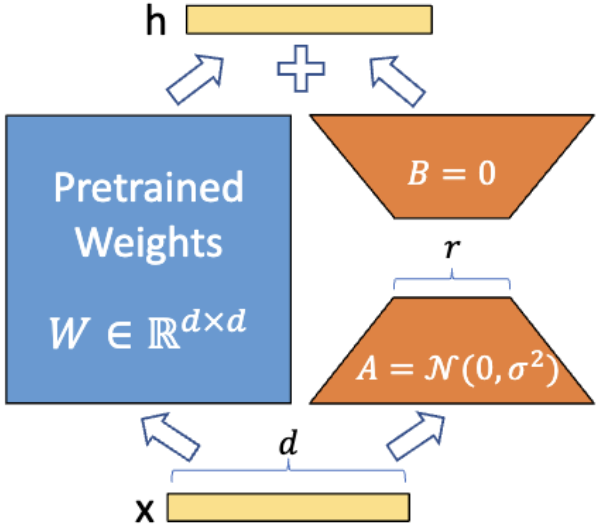

output
MLP
LayerNorm
Self Attention
LayerNorm
input
(a)
LoRA:
add to Query and Value heads
h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x
(b)

LoRA



(a)

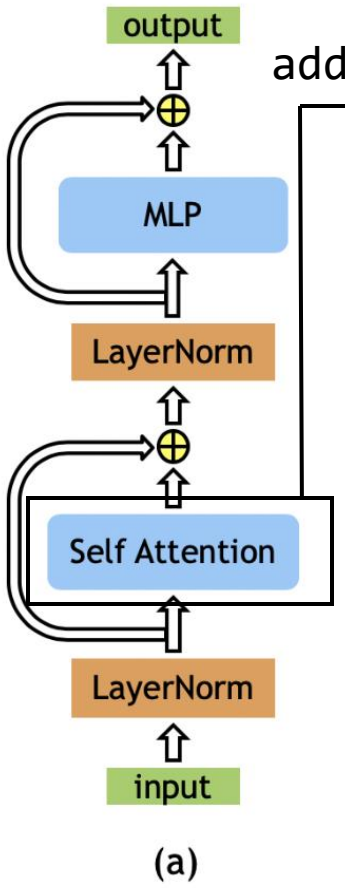
LoRA:
add to Query and Value heads



(b)

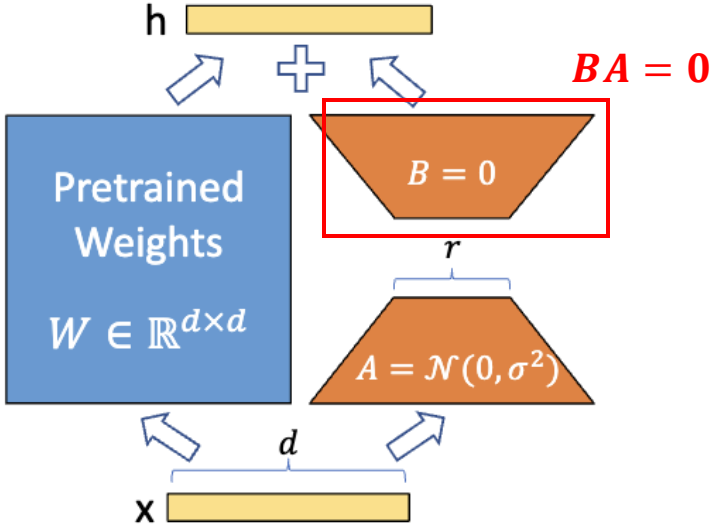
- $\boldsymbol{W = W^0 + BA}, \boldsymbol{W^0} \in \mathbb{R}^{d_1 \times d_2}, \boldsymbol{B} \in \mathbb{R}^{d_1 \times r}, \boldsymbol{A} \in \mathbb{R}^{r \times d_2}$
- $\min_{\boldsymbol{A},\boldsymbol{B}} \mathcal{L}(\mathcal{D}; \boldsymbol{W^0} + \boldsymbol{BA})$ with $\boldsymbol{W^0}$ fixed.
- Efficiency: $r \ll \{d_1, d_2\} \Rightarrow$ less than 0.5% params.
- LoRA saves **backward** computation.

LoRA



(a)

LoRA:
add to Query and Value heads



(b)

- $\boldsymbol{W = W^0 + BA}, \boldsymbol{W^0} \in \mathbb{R}^{d_1 \times d_2}, \boldsymbol{B} \in \mathbb{R}^{d_1 \times r}, \boldsymbol{A} \in \mathbb{R}^{r \times d_2}$
- $\min_{\boldsymbol{A},\boldsymbol{B}} \mathcal{L}(\mathcal{D}; \boldsymbol{W^0} + \boldsymbol{BA})$ with $\boldsymbol{W^0}$ fixed.
- Efficiency: $r \ll \{d_1, d_2\} \Rightarrow$ less than 0.5% params.
- LoRA saves **backward** computation.

LoRA

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1}_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm.3}$	$90.8_{\pm.1}$	$\mathbf{86.6}_{\pm.7}$	$\mathbf{91.5}_{\pm.2}$	**87.2**
$\text{RoB}_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\text{RoB}_{\text{large}}$ (LoRA)	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.9}_{\pm1.2}$	$\mathbf{68.2}_{\pm1.9}$	$\mathbf{94.9}_{\pm.3}$	$91.6_{\pm.1}$	$\mathbf{87.4}_{\pm2.5}$	$\mathbf{92.6}_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	$90.2_{\pm.3}$	$96.1_{\pm.3}$	$90.2_{\pm.7}$	$\mathbf{68.3}_{\pm1.0}$	$\mathbf{94.8}_{\pm.2}$	$\mathbf{91.9}_{\pm.1}$	$83.8_{\pm2.9}$	$92.1_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	$\mathbf{90.5}_{\pm.3}$	$\mathbf{96.6}_{\pm.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	$\mathbf{94.8}_{\pm.3}$	$91.7_{\pm.2}$	$80.1_{\pm2.9}$	$91.9_{\pm.4}$	87.9
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	6.0M	$89.9_{\pm.5}$	$96.2_{\pm.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm.2}$	$92.1_{\pm.1}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	87.8
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
$\text{RoB}_{\text{large}}$ (LoRA)†	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.2}_{\pm1.0}$	$68.2_{\pm1.9}$	$\mathbf{94.8}_{\pm.3}$	$91.6_{\pm.2}$	$\mathbf{85.2}_{\pm1.1}$	$\mathbf{92.3}_{\pm.5}$	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	$\mathbf{91.9}_{\pm.2}$	$96.9_{\pm.2}$	$\mathbf{92.6}_{\pm.6}$	$\mathbf{72.4}_{\pm1.1}$	$\mathbf{96.0}_{\pm.1}$	$\mathbf{92.9}_{\pm.1}$	$\mathbf{94.9}_{\pm.4}$	$\mathbf{93.0}_{\pm.2}$	**91.3**

LoRA

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1}_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm.3}$	$90.8_{\pm.1}$	$\mathbf{86.6}_{\pm.7}$	$\mathbf{91.5}_{\pm.2}$	**87.2**
$\text{RoB}_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\text{RoB}_{\text{large}}$ (LoRA)	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.9}_{\pm1.2}$	$\mathbf{68.2}_{\pm1.9}$	$\mathbf{94.9}_{\pm.3}$	$91.6_{\pm.1}$	$\mathbf{87.4}_{\pm2.5}$	$\mathbf{92.6}_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	$90.2_{\pm.3}$	$96.1_{\pm.3}$	$90.2_{\pm.7}$	$\mathbf{68.3}_{\pm1.0}$	$\mathbf{94.8}_{\pm.2}$	$\mathbf{91.9}_{\pm.1}$	$83.8_{\pm2.9}$	$92.1_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	$\mathbf{90.5}_{\pm.3}$	$\mathbf{96.6}_{\pm.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	$\mathbf{94.8}_{\pm.3}$	$91.7_{\pm.2}$	$80.1_{\pm2.9}$	$91.9_{\pm.4}$	87.9
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	6.0M	$89.9_{\pm.5}$	$96.2_{\pm.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm.2}$	$92.1_{\pm.1}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	87.8
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
$\text{RoB}_{\text{large}}$ (LoRA)†	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.2}_{\pm1.0}$	$68.2_{\pm1.9}$	$\mathbf{94.8}_{\pm.3}$	$91.6_{\pm.2}$	$\mathbf{85.2}_{\pm1.1}$	$\mathbf{92.3}_{\pm.5}$	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	$\mathbf{91.9}_{\pm.2}$	$96.9_{\pm.2}$	$\mathbf{92.6}_{\pm.6}$	$\mathbf{72.4}_{\pm1.1}$	$\mathbf{96.0}_{\pm.1}$	$\mathbf{92.9}_{\pm.1}$	$\mathbf{94.9}_{\pm.4}$	$\mathbf{93.0}_{\pm.2}$	**91.3**

On GPT-3 (175B)

Num. of Nvidia V100 32GB: 96 ➔ 24

Model checkpoint Size: 1 TB ➔ 200 MB

AdaLoRA

LoRA:

- $\boldsymbol{W = W^0 + BA}$, $\boldsymbol{A}$, $\boldsymbol{B}$ have rank r
  - larger r makes better performance but more computationally expensive.
  - is a fixed hyper-param, ignoring layer importance.

- Suboptimal efficiency.

AdaLoRA:

- $\boldsymbol{W = W^0 + \Delta}$, adjust the rank of $\boldsymbol{\Delta}$ to control the budget.

Zhang et al. "*AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning*." International Conference on Learning Representations (ICLR), 2023.

AdaLoRA

AdaLoRA:

- $\boldsymbol{W = W^0 + \Delta}$

By SVD: $\boldsymbol{\Delta = P\Lambda Q}$

- Diagonal $\boldsymbol{\Lambda} \in \mathbb{R}^{r \times r} \Rightarrow$ singular values $\{\lambda_i\}_{1 \le i \le r}$.
- $\boldsymbol{P} \in \mathbb{R}^{d_1 \times r}, \boldsymbol{Q} \in \mathbb{R}^{r \times d_2} \Rightarrow$ left and right singular vectors.
- Regularize to enforce the orthogonality:

$$R(\boldsymbol{P}, \boldsymbol{Q}) = \| \boldsymbol{P}^\top \boldsymbol{P} - \boldsymbol{I} \|_F^2 + \| \boldsymbol{Q}\boldsymbol{Q}^\top - \boldsymbol{I} \|_F^2$$

AdaLoRA

AdaLoRA:

- $\boldsymbol{W = W^0 + \Delta}$

By SVD: $\boldsymbol{\Delta = P\Lambda Q}$

Iteratively prune to adjust the rank

- Diagonal $\boldsymbol{\Lambda} \in \mathbb{R}^{r \times r} \Rightarrow$ singular values $\{\lambda_i\}_{1 \le i \le r}$.
- $\boldsymbol{P} \in \mathbb{R}^{d_1 \times r}, \boldsymbol{Q} \in \mathbb{R}^{r \times d_2} \Rightarrow$ left and right singular vectors.
- Regularize to enforce the orthogonality:

$$R(\boldsymbol{P}, \boldsymbol{Q}) = \| \boldsymbol{P}^\top \boldsymbol{P} - \boldsymbol{I} \|_F^2 + \| \boldsymbol{Q}\boldsymbol{Q}^\top - \boldsymbol{I} \|_F^2$$

AdaLoRA

Rank Allocation: Assume $k \in [n]$ LoRA modules are learned together. Given budget $b^{(t)}$ at step t,

$$\widetilde{\Lambda}_k^{(t)} = \Lambda_k^{(t)} - \eta \nabla_{\Lambda_k} \mathcal{L}, \quad \Lambda_k^{(t+1)} = \mathcal{T}(\widetilde{\Lambda}_k^{(t)}, S_k^{(t)})$$

With

$$\mathcal{T}(\widetilde{\Lambda}_k^{(t)}, S_k^{(t)})_{ii} = \begin{cases} \widetilde{\Lambda}_{k,ii}^{(t)} & S_{k,i}^{(t)} \text{ is in the top-} b^{(t)}, \\ 0 & \text{otherwise,} \end{cases}$$

AdaLoRA

Rank Allocation: Assume $k \in [n]$ LoRA modules are learned together. Given budget $b^{(t)}$ at step t,

$$\widetilde{\Lambda}_k^{(t)} = \Lambda_k^{(t)} - \eta \nabla_{\Lambda_k} \mathcal{L}, \quad \Lambda_k^{(t+1)} = \mathcal{T}(\widetilde{\Lambda}_k^{(t)}, S_k^{(t)})$$

With

$$\mathcal{T}(\widetilde{\Lambda}_k^{(t)}, S_k^{(t)})_{ii} = \begin{cases} \widetilde{\Lambda}_{k,ii}^{(t)} & S_{k,i}^{(t)} \text{ is in the top-} b^{(t)}, \\ 0 & \text{otherwise,} \end{cases}$$

Importance of the k-th LoRA's i-th rank at step t.

AdaLoRA

Rank Allocation:

$$\widetilde{\Lambda}_k^{(t)} = \Lambda_k^{(t)} - \eta \nabla_{\Lambda_k} \mathcal{L}, \quad \Lambda_k^{(t+1)} = \mathcal{T}(\widetilde{\Lambda}_k^{(t)}, S_k^{(t)})$$

With

$$\mathcal{T}(\widetilde{\Lambda}_k^{(t)}, S_k^{(t)})_{ii} = \begin{cases} \widetilde{\Lambda}_{k,ii}^{(t)} & S_{k,i}^{(t)} \text{ is in the top-} b^{(t)}, \\ 0 & \text{otherwise,} \end{cases}$$

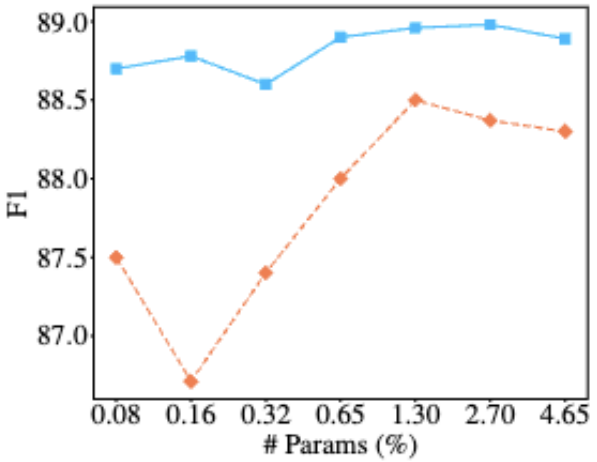
Sensitivity-based Importance Score:

$$S_{k,i} = s(\lambda_{k,i}) + \frac{1}{d_1} \sum_{j=1}^{d_1} s(P_{k,ji}) + \frac{1}{d_2} \sum_{j=1}^{d_2} s(Q_{k,ij}),$$

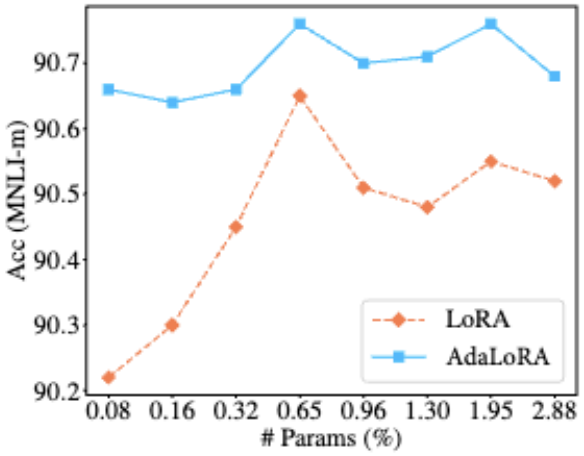
Sensitive score of single entry adjusted by uncertainty.

AdaLoRA

Method	# Params	MNLI m/mm	SST-2 Acc	CoLA Mcc	QQP Acc/F1	QNLI Acc	RTE Acc	MRPC Acc	STS-B Corr	All Ave.
Full FT	184M	89.90/90.12	95.63	69.19	**92.40/89.80**	94.03	83.75	89.46	91.60	88.09
BitFit	0.1M	89.37/89.91	94.84	66.96	88.41/84.95	92.24	78.70	87.75	91.35	86.02
HAdapter	1.22M	90.13/90.17	95.53	68.64	91.91/89.27	94.11	84.48	89.95	91.48	88.12
PAdapter	1.18M	90.33/90.39	95.61	68.77	92.04/89.40	94.29	85.20	89.46	91.54	88.24
LoRA$_{r=8}$	1.33M	90.65/90.69	94.95	69.82	91.99/89.38	93.87	85.20	89.95	91.60	88.34
AdaLoRA	1.27M	**90.76/90.79**	**96.10**	**71.45**	**92.23/89.74**	**94.55**	**88.09**	**90.69**	**91.84**	**89.31**
HAdapter	0.61M	90.12/90.23	95.30	67.87	91.65/88.95	93.76	85.56	89.22	91.30	87.93
PAdapter	0.60M	90.15/90.28	95.53	69.48	91.62/88.86	93.98	84.12	89.22	91.52	88.04
HAdapter	0.31M	90.10/90.02	95.41	67.65	91.54/88.81	93.52	83.39	89.25	91.31	87.60
PAdapter	0.30M	89.89/90.06	94.72	69.06	91.40/88.62	93.87	84.48	89.71	91.38	87.90
LoRA$_{r=2}$	0.33M	90.30/90.38	94.95	68.71	91.61/88.91	94.03	85.56	89.71	**91.68**	88.15
AdaLoRA	0.32M	**90.66/90.70**	**95.80**	**70.04**	**91.78/89.16**	**94.49**	**87.36**	**90.44**	91.63	**88.86**



(b) SQuADv2.0



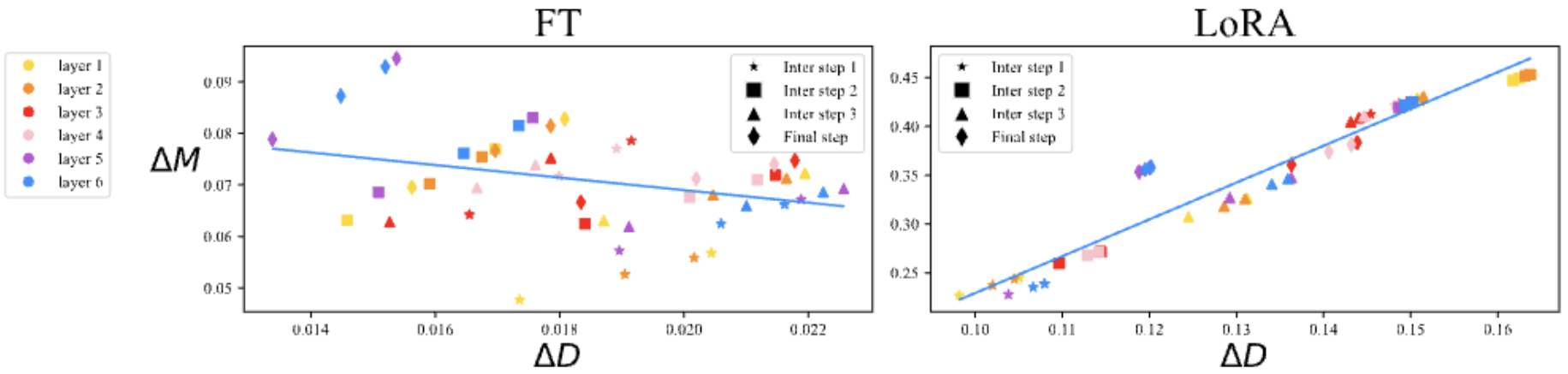
(a) MNLI

DoRA

LoRA:

- $\boldsymbol{W = W^0 + BA}$
  - Performance gap between LoRA and Full Fine-tuning (FFT).

Direction (D) and Magnitude (M) are of different importance in FFT that **LoRA fails to capture**.



Liu, Shih-Yang, et al. "*DoRA: Weight-Decomposed Low-Rank Adaptation*." International Conference on Machine Learning (ICML), 2024.

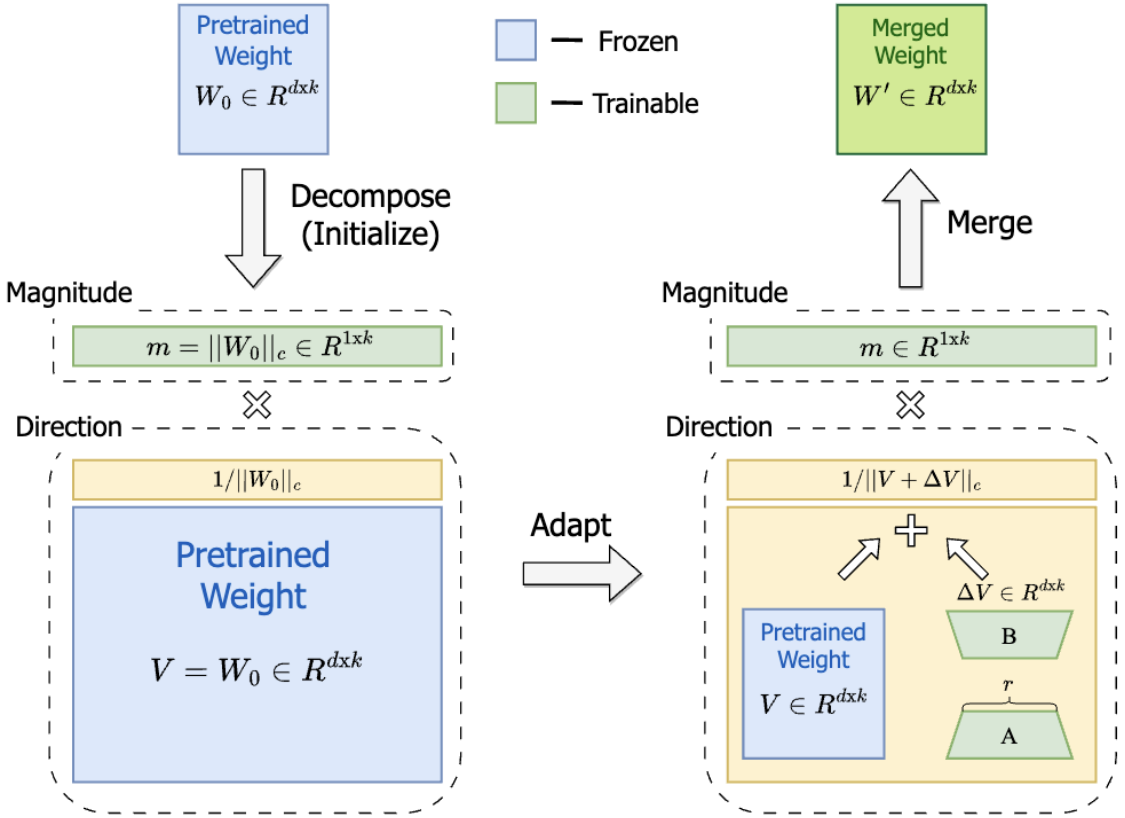
DoRA

DoRA: Weight-Decomposition

- LoRA ⇒ direction update
- + new vector ⇒ magnitude update

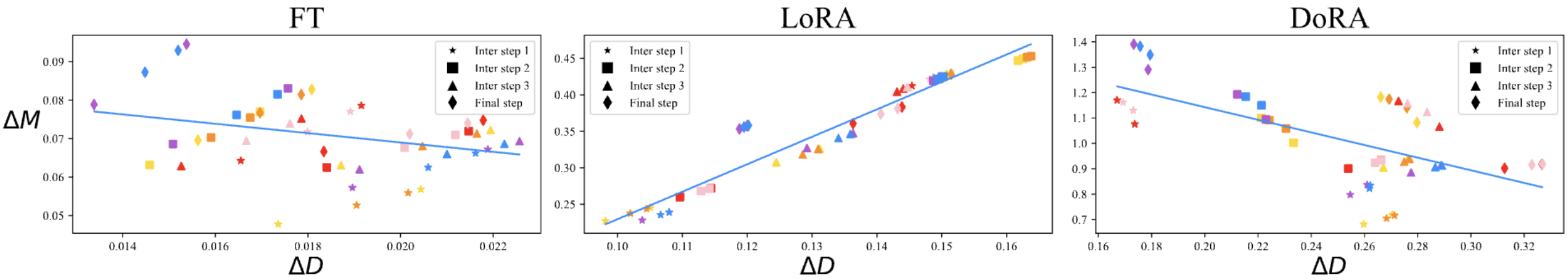
~ 0.01% overhead

$$W' = \underline{m}\frac{V + \Delta V}{||V + \Delta V||_c} = \underline{m}\frac{W_0 + \underline{BA}}{||W_0 + \underline{BA}||_c}$$

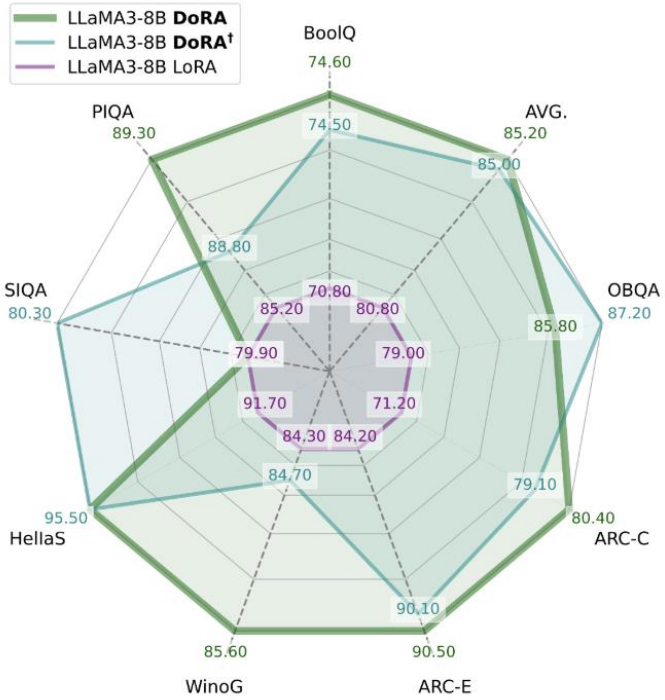


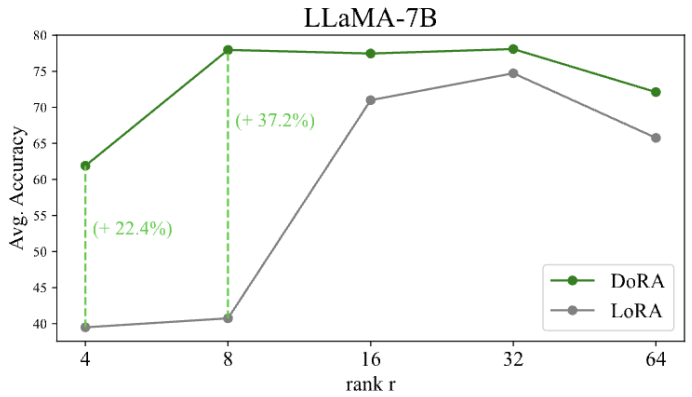
DoRA

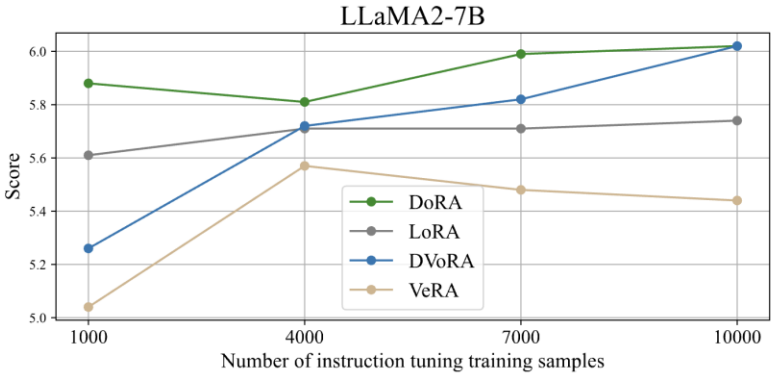
DoRA captures the magnitude and direction updates pattern in FFT.



DoRA


LLaMA3-8B DoRA
LLaMA3-8B DoRA†
LLaMA3-8B LoRA
BoolQ
74.60
PIQA
89.30
AVG.
85.20
SIQA
80.30
OBQA
87.20
HellaS
95.50
ARC-C
80.40
WinoG
85.60
ARC-E
90.50
74.50
85.00
88.80
85.80
70.80
85.20
80.80
79.90
79.00
91.70
71.20
84.30
84.20
84.70
79.10
90.10

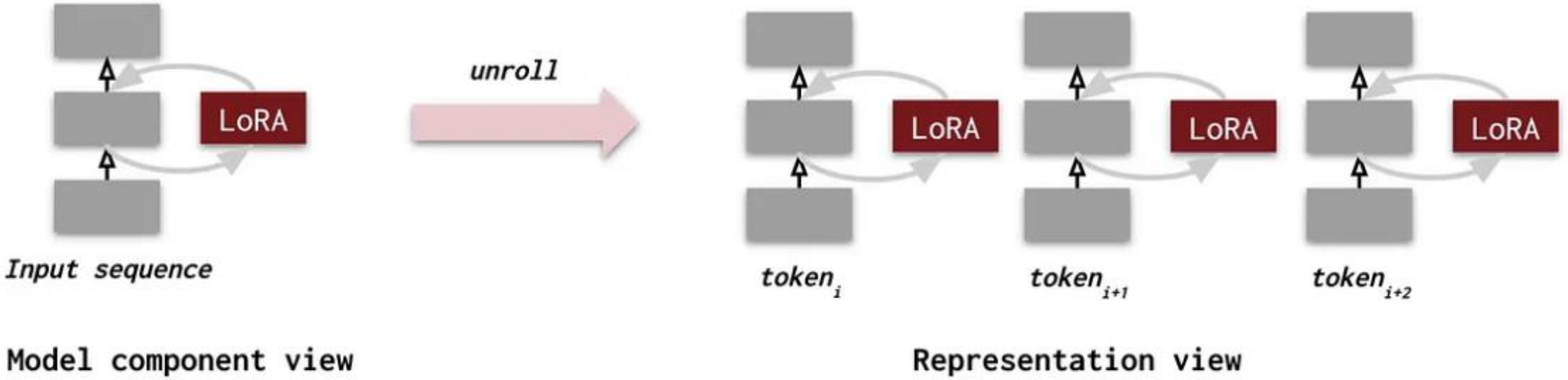

LLaMA-7B
Avg. Accuracy
rank r
(+ 22.4%)
(+ 37.2%)
DoRA
LoRA


LLaMA2-7B
Score
Number of instruction tuning training samples
DoRA
LoRA
DVoRA
VeRA

ReFT

LoRA:

- Update model weights directly.
- Change representations at **all** timesteps.

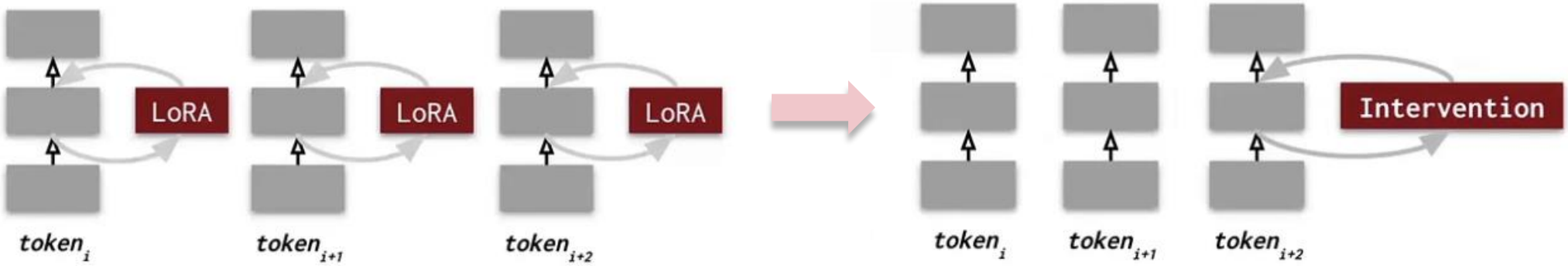


Wu et al. "*ReFT: Representation Finetuning for Language Models*." Advances in Neural Information Processing Systems (NeurIPS), 2024.

ReFT

ReFT:

- ~~Update model weights~~. **Learn a set of interventions.**
- Change representations ~~across all timesteps~~. **At a few timesteps.**



(a) LoRA (b) ReFT

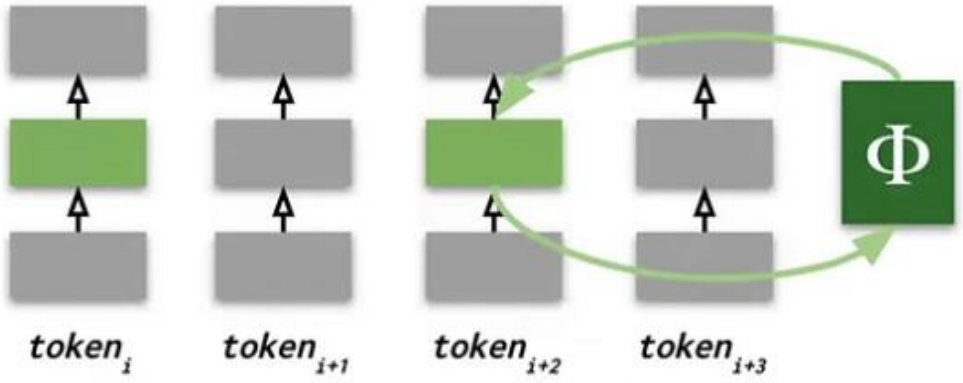
Credits: *ReFT: Representation Finetuning for Language Models*.

ReFT

ReFT:

- Select **layer** l and **timestep** i to apply the intervention.
- **Intervention** is a low-rank and linear update:

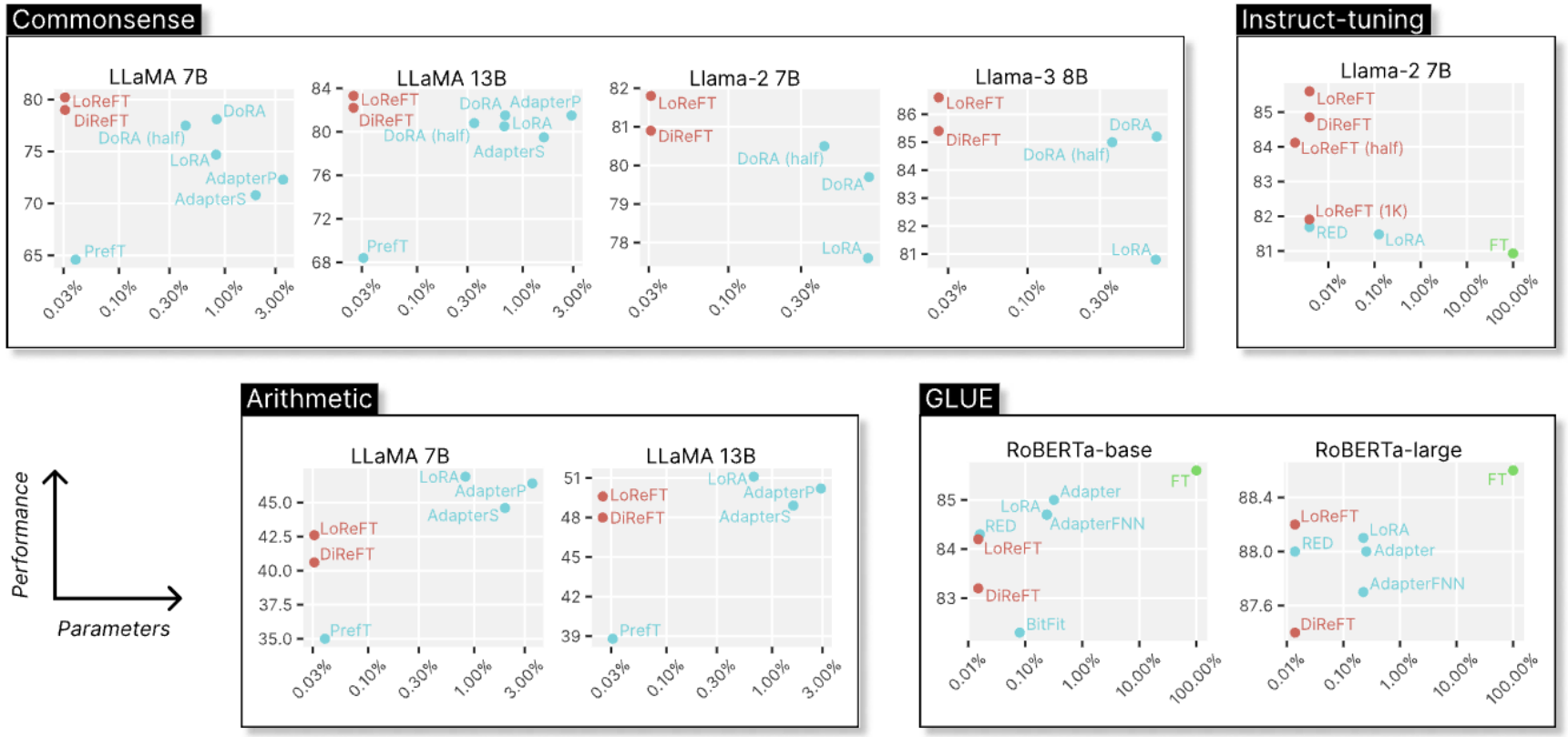
$$\Phi_l(\boldsymbol{h}_i^{(l)}; \phi_l) = \boldsymbol{h}_i^{(l)} + \mathbf{R}_l^\top(\mathbf{A}_l \boldsymbol{h}_i^{(l)} + \boldsymbol{b}_l - \mathbf{R}_l \boldsymbol{h}_i^{(l)}),$$



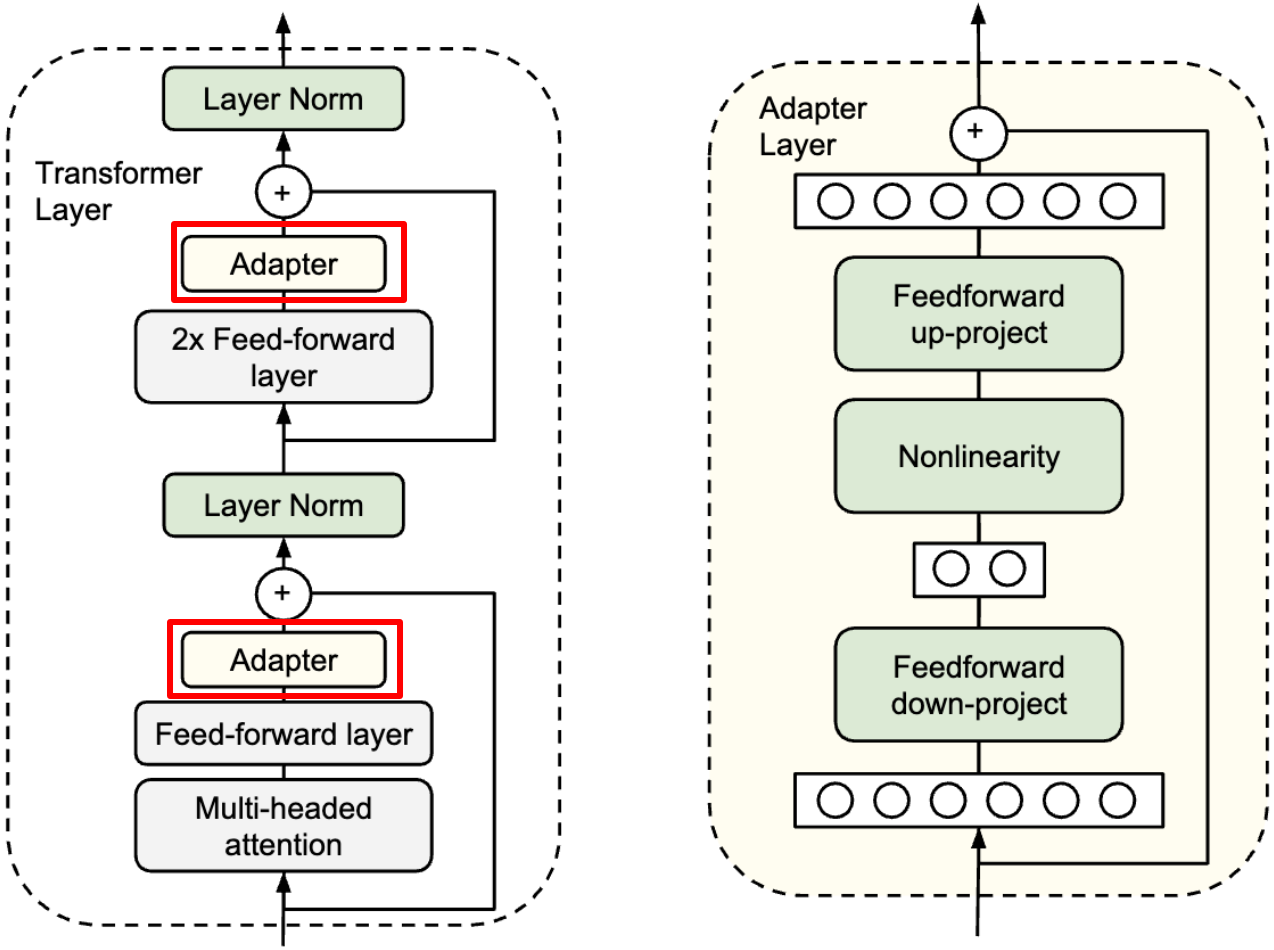
Credits: *ReFT: Representation Finetuning for Language Models*.

ReFT

Efficiency: 0.8% (LoRA) ⇒ 0.03% (ReFT)

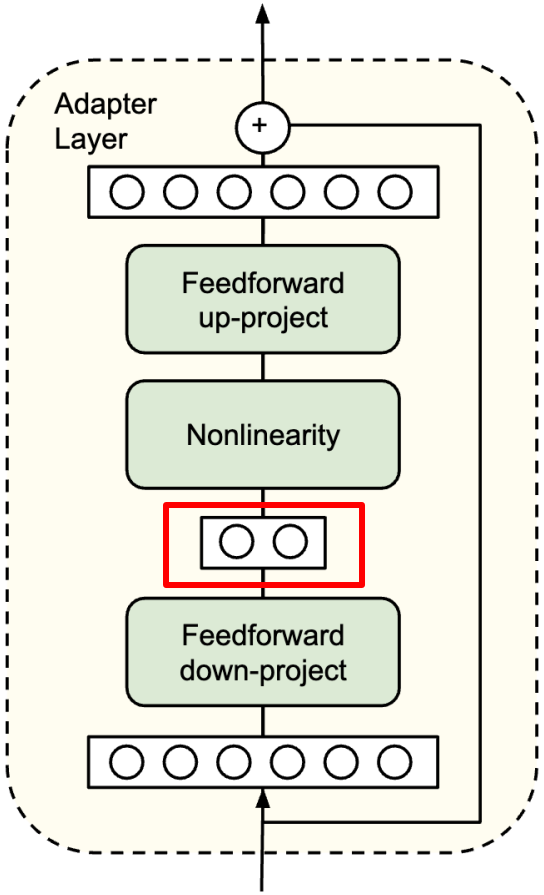


Adapter



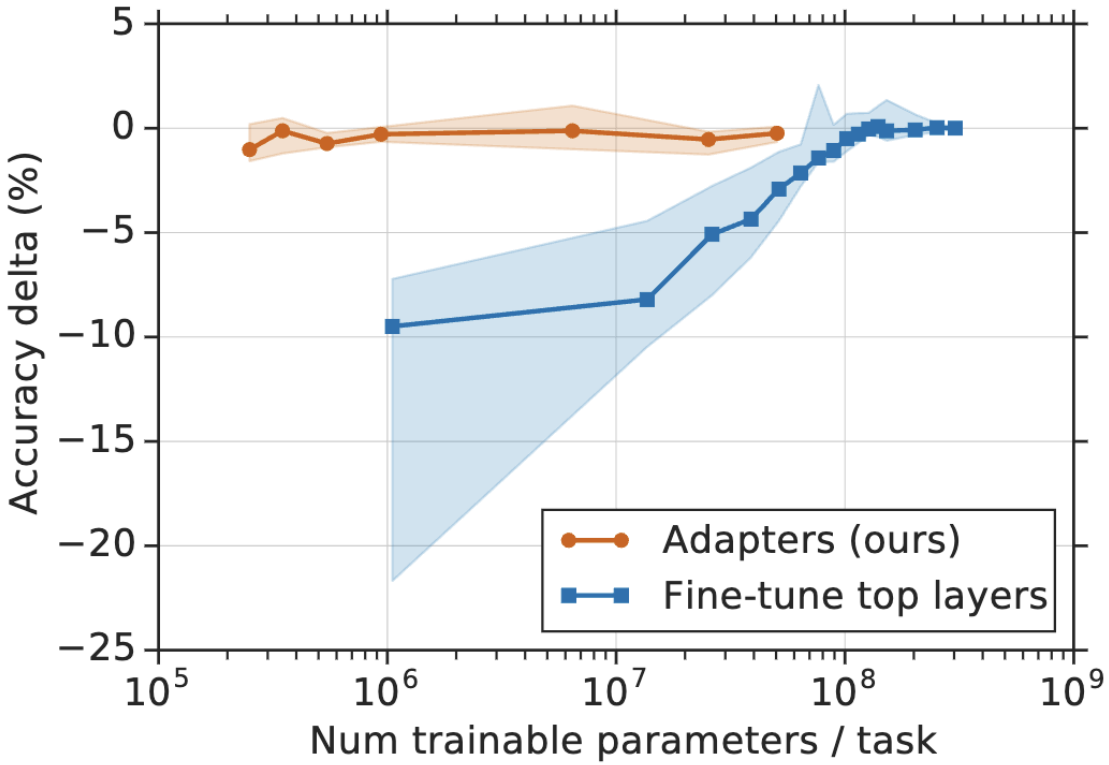
Houlsby et al. "*Parameter-efficient transfer learning for NLP.*" International conference on Machine Learning (ICML), 2019.

Adapter



Bottleneck to save # params

Adapter

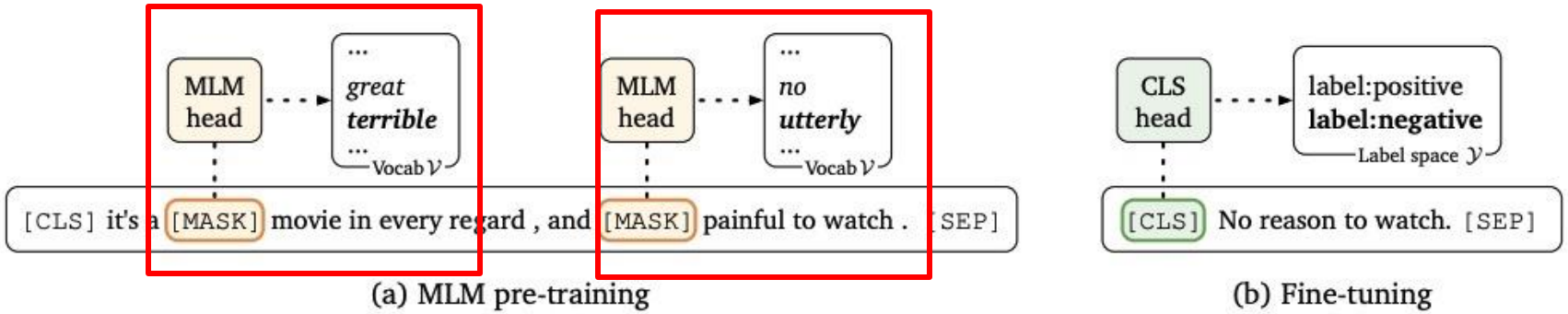


0.4% accuracy drop for 96.4% reduction in the #of parameters/task

LiST

Encoder-based Model:

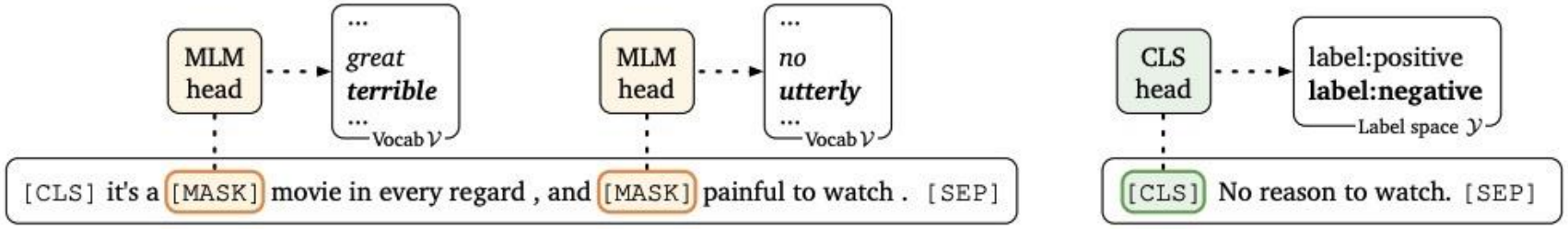
- Pre-trained to reconstruct randomly masked words given others.
- Fine-tuned to conduct downstream tasks using [CLS].



Wang et al. "*LiST: Lite Prompted Self-training Makes Parameter-efficient Few-shot Learners.*" Findings of NAACL, 2022.

LiST

Encoder-based Model:

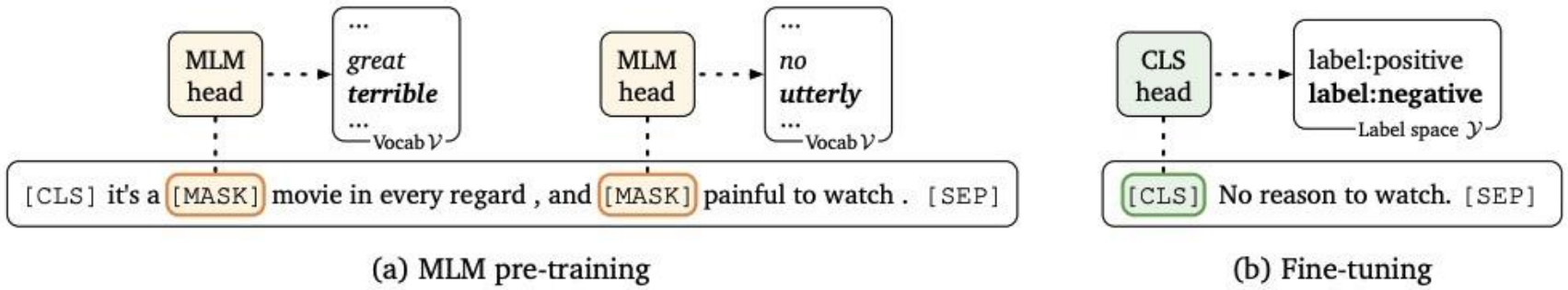


Pre-training and Fine-tuning objective gap

- **CLS token is used differently**

LiST

Encoder-based Model:



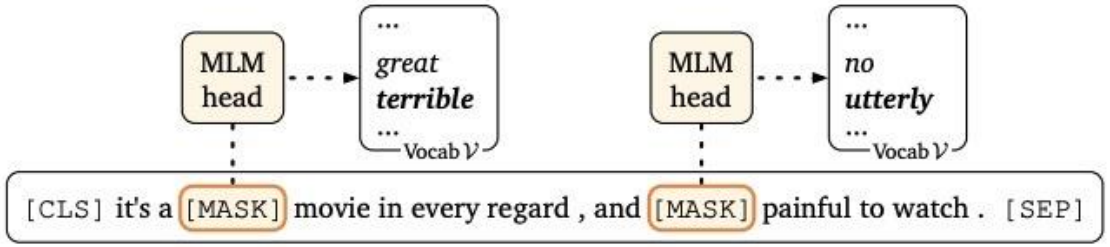
Use pre-training objective

[CLS] No reason to watch. It was [MASK]. [SEP]

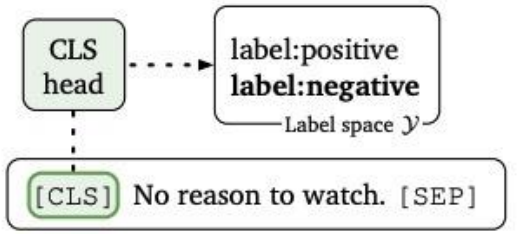
Original Input Prompt

(c) Prompt-based Fine-tuning

LiST



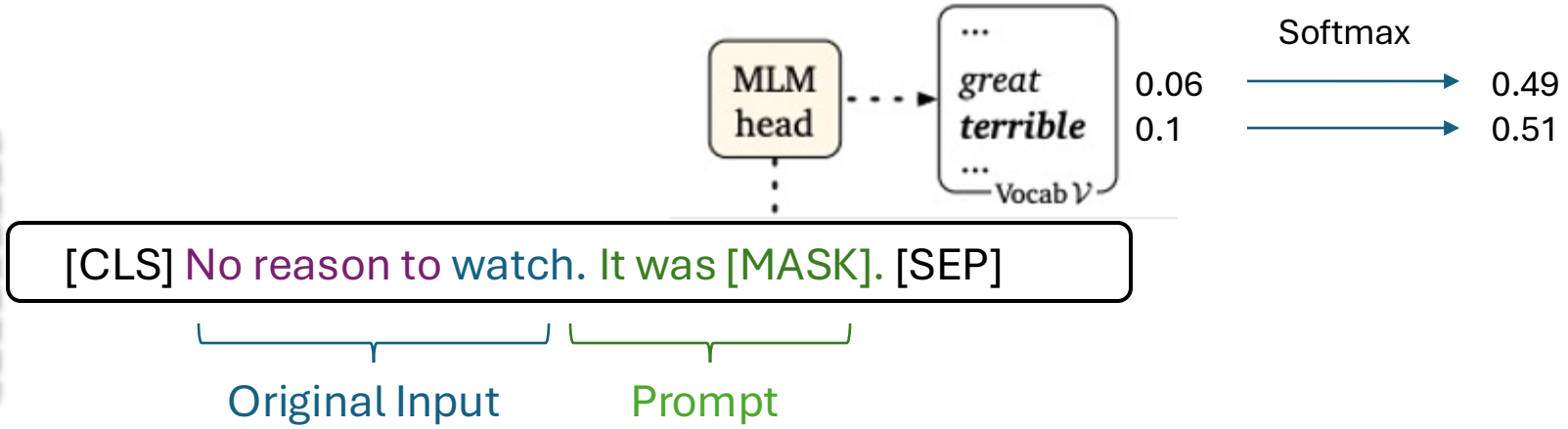
(a) MLM pre-training



(b) Fine-tuning

Sentiment Analysis with pre-defined label words

- Use *great* and *terrible* in Vocab V to indicate *pos* and *neg* classes



(c) Prompt-based Fine-tuning

LiST

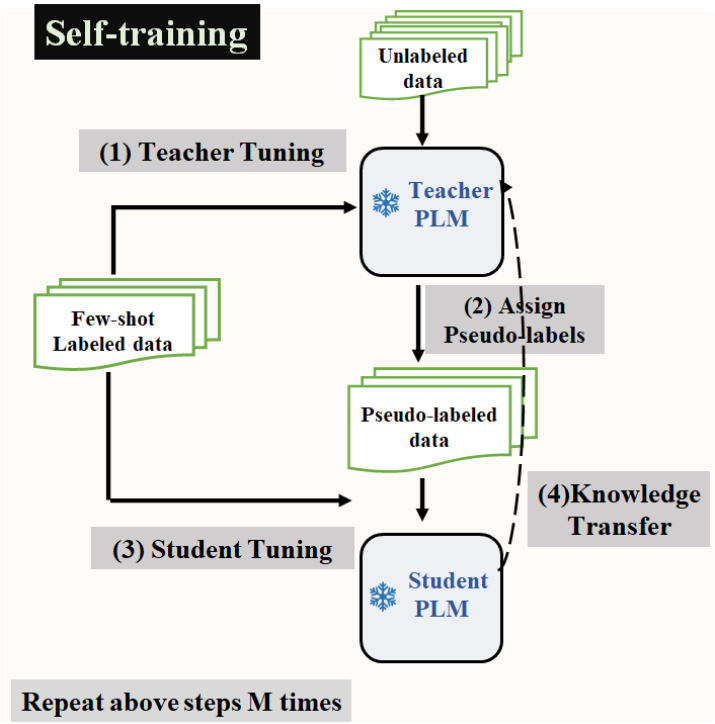
Standard Benchmarks:

- Large-scale labeled data
- No unlabeled data

Self-training

Real-world Applications:

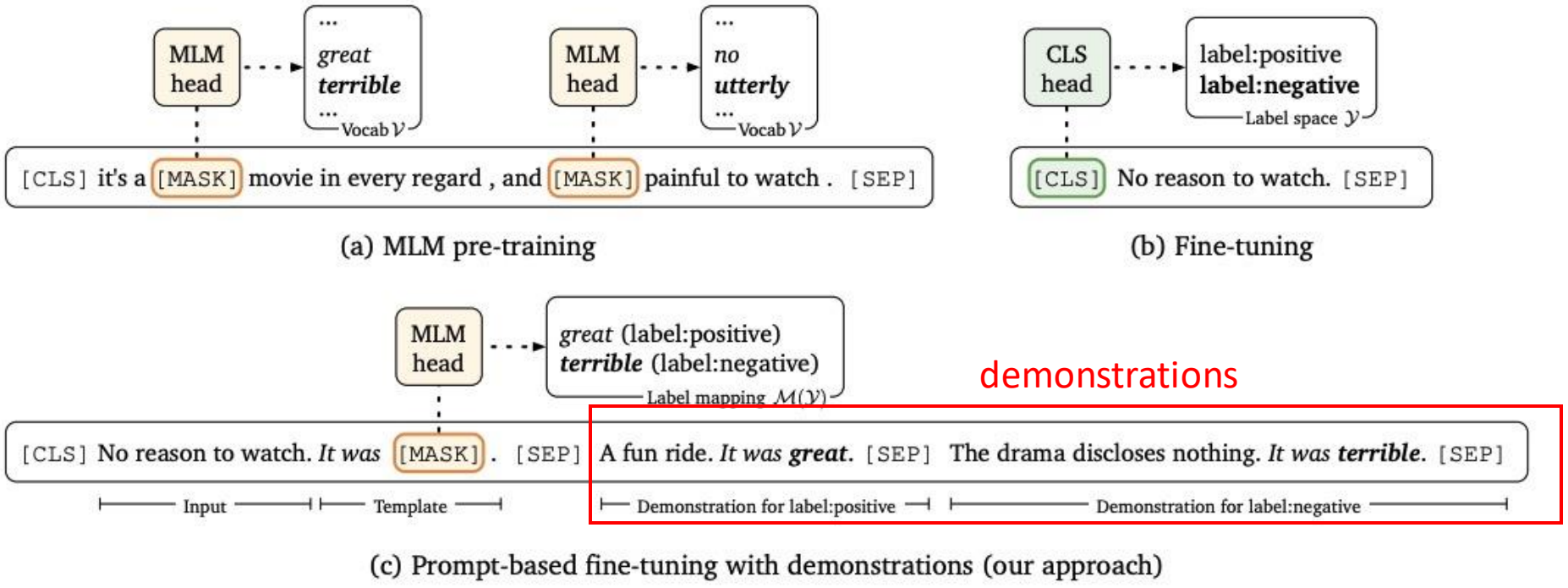
- Limited labeled data
- Many unlabeled data



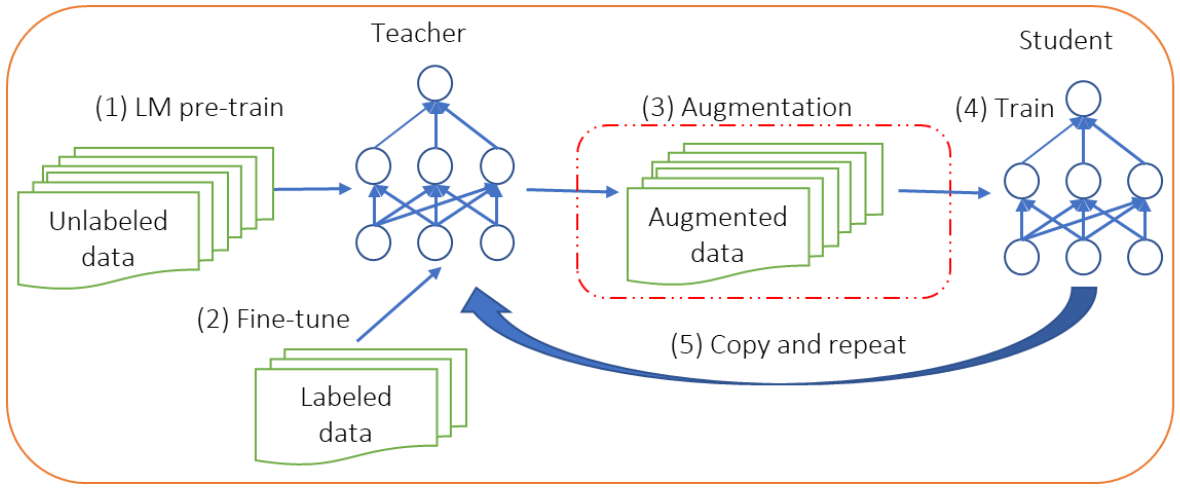
LiST:

- Prompt-based tuning
- Self-training
- PEFT: Adapter

LiST


MLM head
... great terrible ... Vocab $\mathcal{V}$
MLM head
... no utterly ... Vocab $\mathcal{V}$
[CLS] it's a [MASK] movie in every regard , and [MASK] painful to watch . [SEP]
(a) MLM pre-training
CLS head
label:positive label:negative Label space $\mathcal{Y}$
[CLS] No reason to watch. [SEP]
(b) Fine-tuning
MLM head
great (label:positive) terrible (label:negative) Label mapping $\mathcal{M}(\mathcal{Y})$
demonstrations
[CLS] No reason to watch. It was [MASK] . [SEP] A fun ride. It was great. [SEP] The drama discloses nothing. It was terrible. [SEP]
Input
Template
Demonstration for label:positive
Demonstration for label:negative
(c) Prompt-based fine-tuning with demonstrations (our approach)

LiST



Standard Self-training

$$\hat{\psi}_{\text{stu}}^{(t)} = \hat{\psi}_{\text{stu}}^{(t-1)} - \alpha\nabla\Big(\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\hat{y}_i^{(t)}, \text{enc}(x_i^u; \Theta_{\text{PLM}}, \hat{\psi}_{\text{stu}}^{(t-1)})\Big)$$

Pseudo-labels

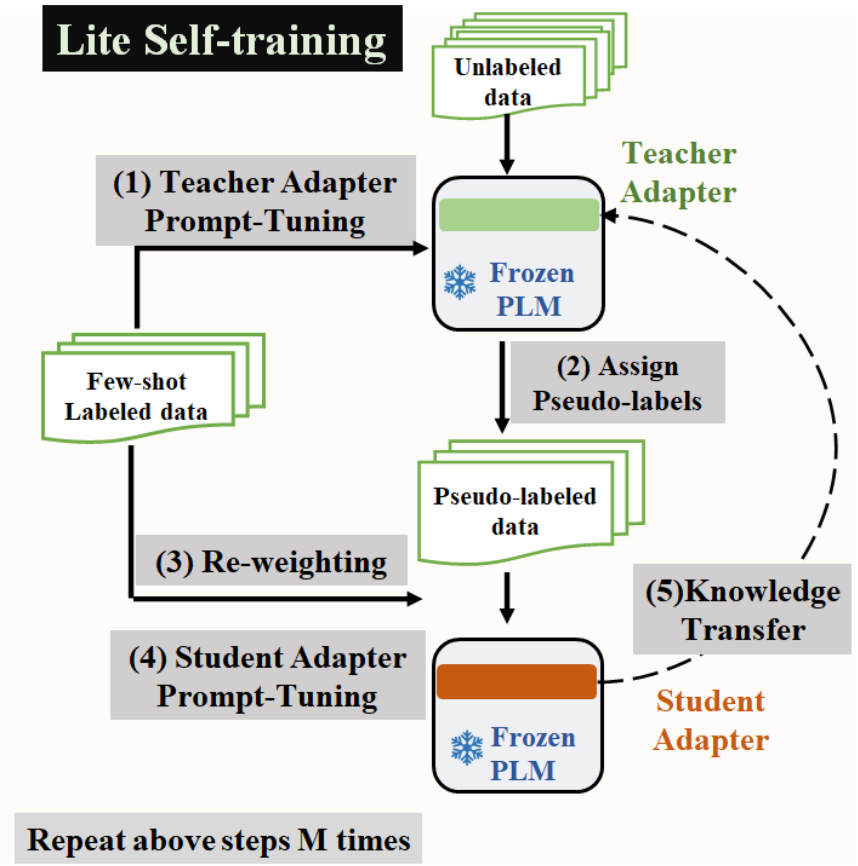
Self-training with Re-weighting

$$\hat{\psi}_{\text{stu}}^{(t)}(\epsilon) = \hat{\psi}_{\text{stu}}^{(t-1)} - \alpha\nabla\Big(\frac{1}{N}\sum_{i=1}^{N}[\epsilon_i^{(t)} \cdot \mathcal{L}(\hat{y}_i^{(t)}, \text{enc}(\tilde{x}_i^u; \Theta_{\text{PLM}}, \hat{\phi}_{\text{stu}}^{(t-1)}))]\Big)$$

Weight for Pseudo-label:

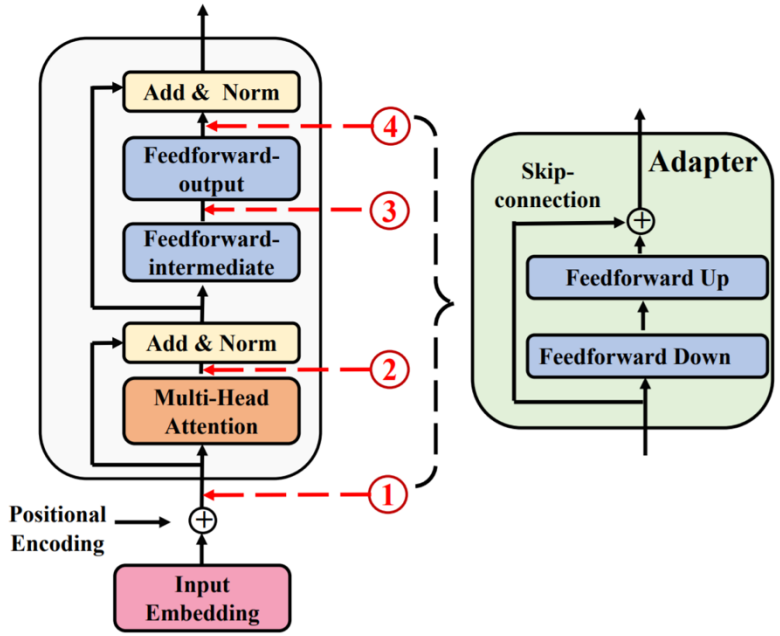
- Meta-learning (2rd-ord gradient).
- Validate w/ few-shot training samples.

LiST



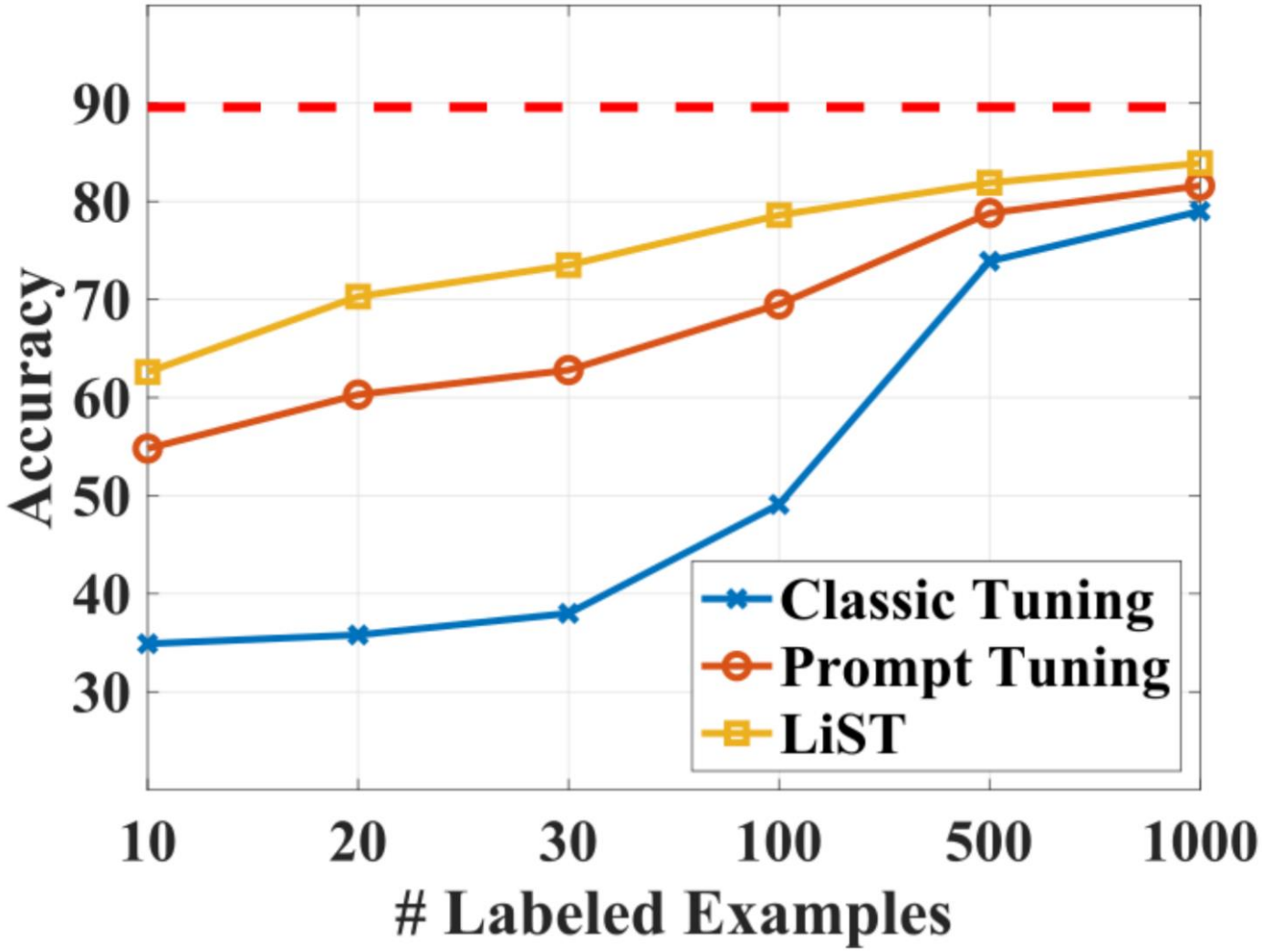
Adapter

- Frozen PLM encoder.
- Teacher and student learn separate adapter.



LiST

- **“Prompt Tuning”:** prompt-based tuning



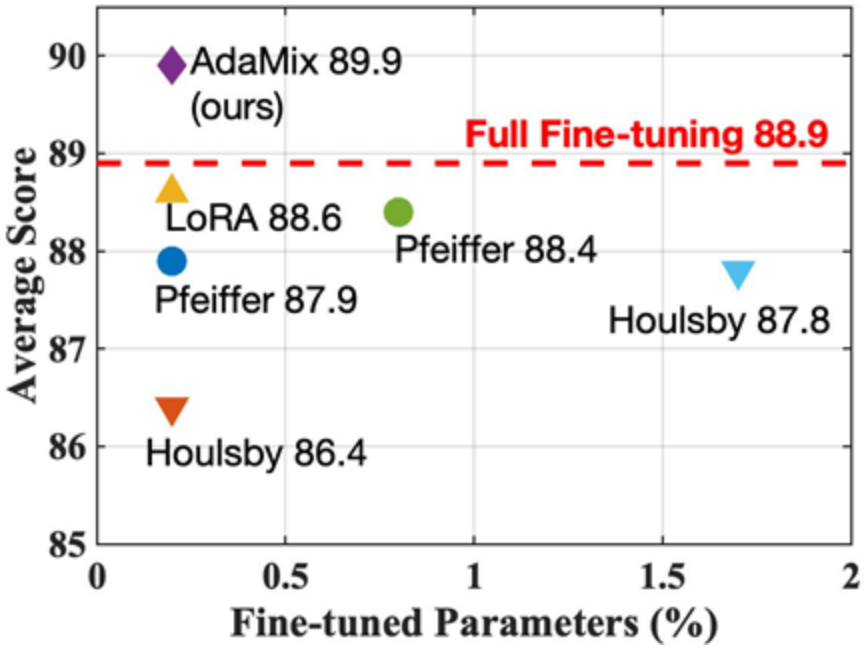
AdaMix

Performance gap between PEFT and FFT

- (Specialized: DoRA [ICML'24])

AdaMix:

- Mixture of Expert + Weight ensemble.

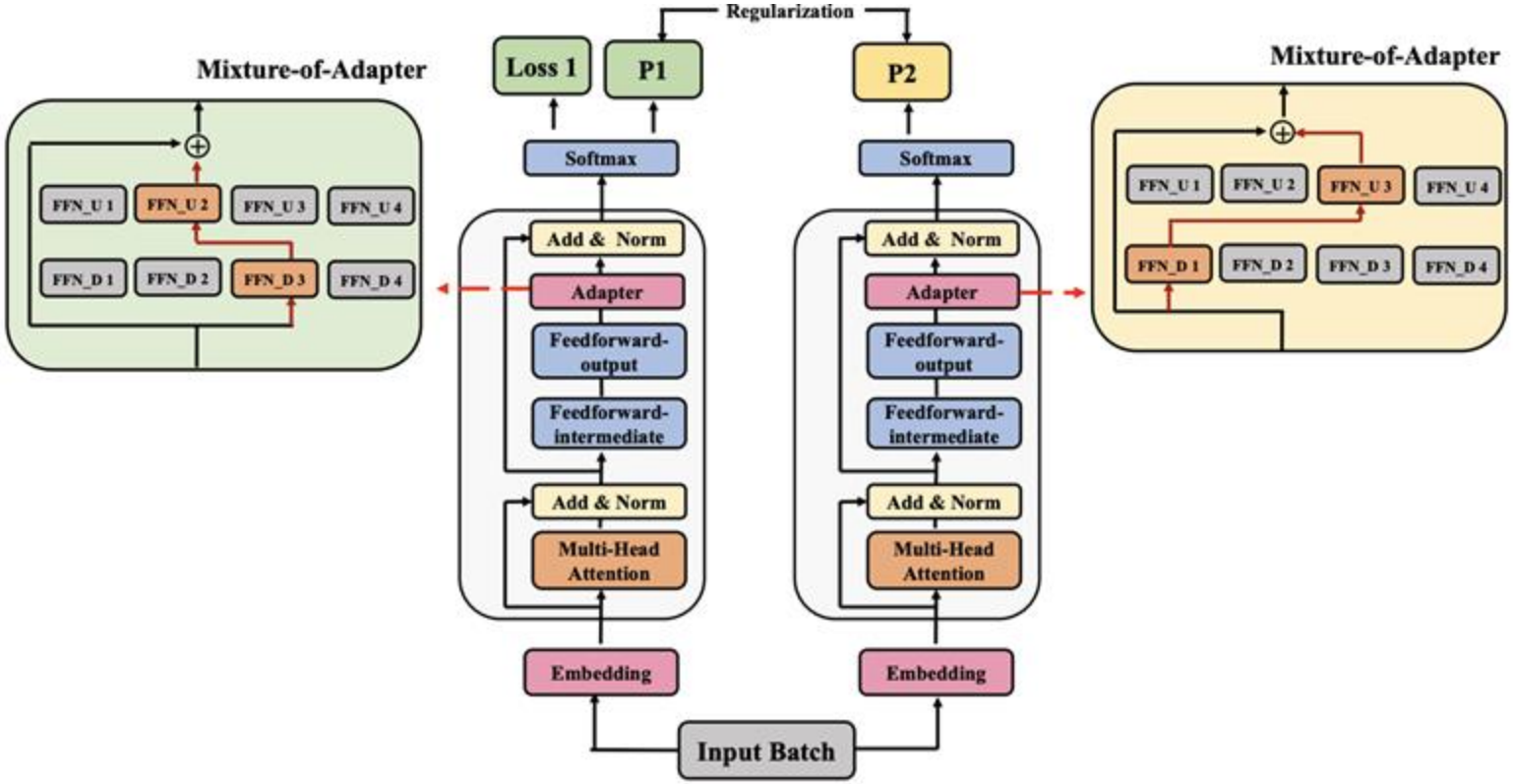


(a) Average performance on GLUE development

Wang et al. "*AdaMix: Mixture-of-Adaptations for Parameter-efficient Model Tuning*." Empirical Methods in Natural Language Processing (EMNLP), 2022.

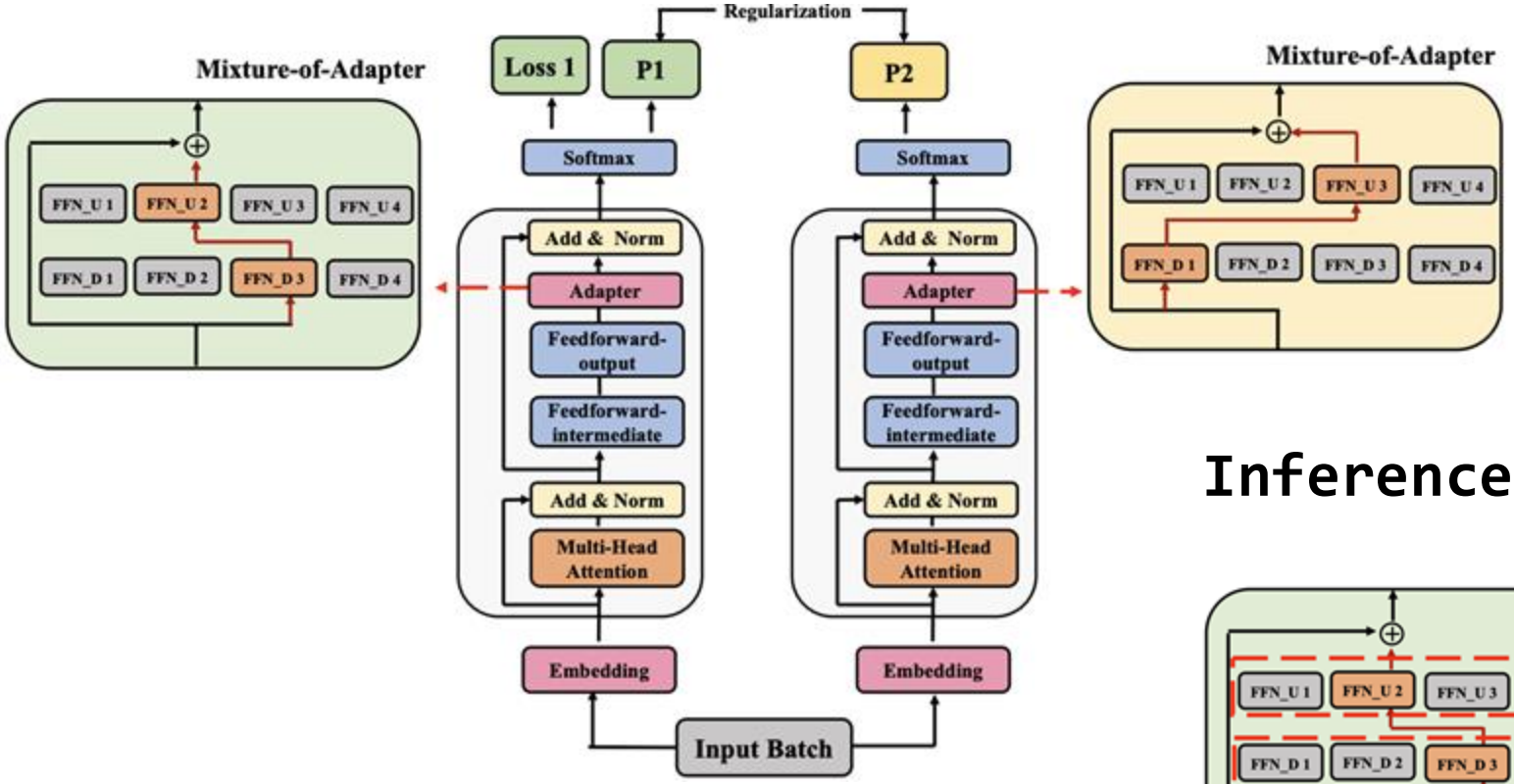
AdaMix

Training

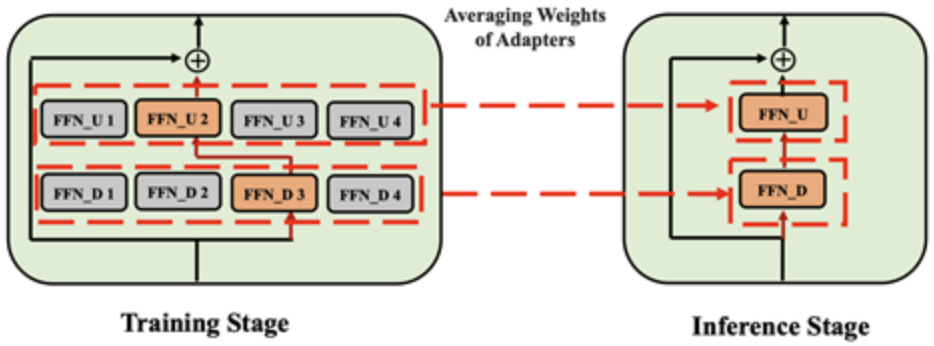


AdaMix

Training



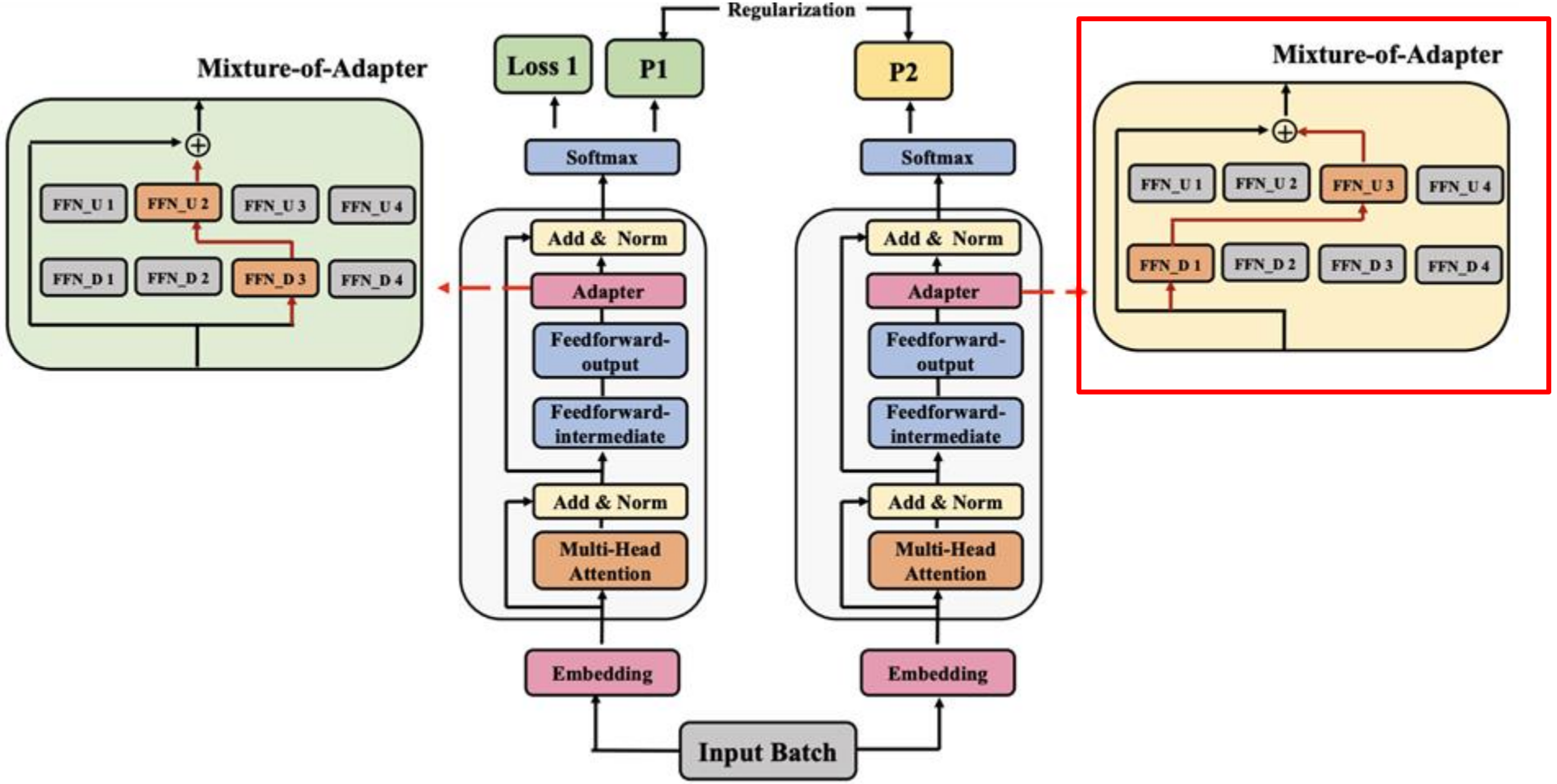
Inference



AdaMix

Adapter ⇒ **Mixture**-of-Adapter:

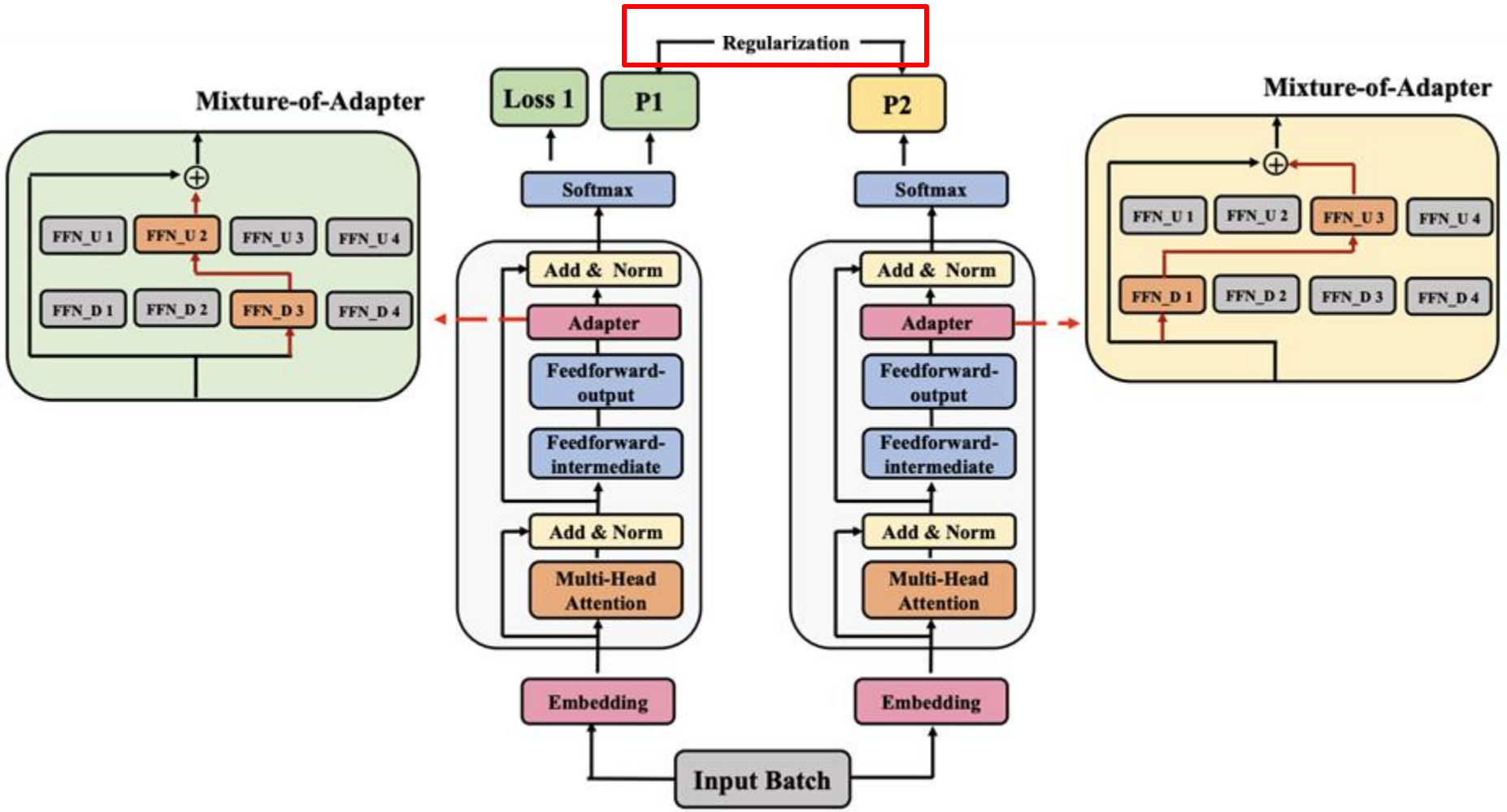
- Copies of Adapters
- Random Routing



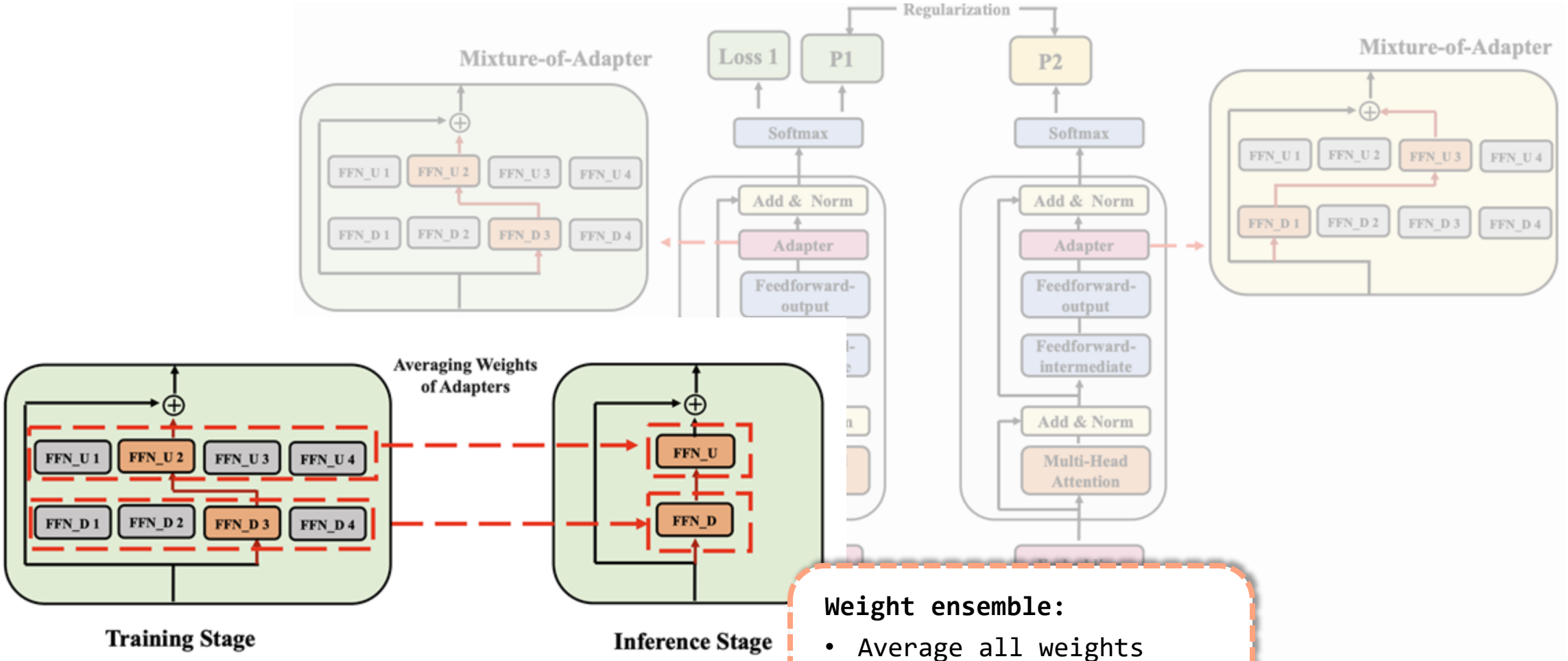
AdaMix

Consistency loss between logits from two forwards

- Connect different copies of params to **avoid collapse**



AdaMix



Weight ensemble:

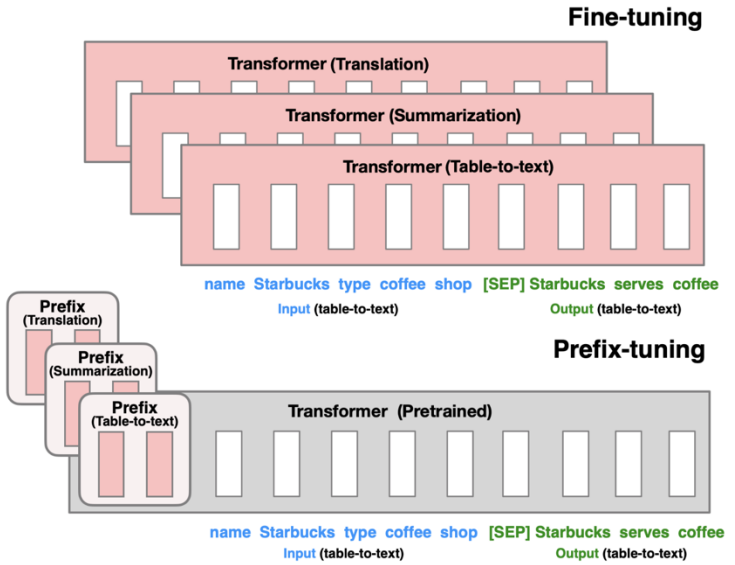
- Average all weights
- Improve the performance
- Reduce the storage cost

AdaMix

Model	#Param.	MNLI Acc	QNLI Acc	SST2 Acc	QQP Acc	MRPC Acc	CoLA Mcc	RTE Acc	STS-B Pearson	**Avg.**
Full Fine-tuning†	355.0M	90.2	94.7	96.4	92.2	90.9	68.0	86.6	**92.4**	88.9
Pfeiffer Adapter†	3.0M	90.2	94.8	96.1	91.9	90.2	68.3	83.8	92.1	88.4
Pfeiffer Adapter†	0.8M	90.5	94.8	96.6	91.7	89.7	67.8	80.1	91.9	87.9
Houlsby Adapter†	6.0M	89.9	94.7	96.2	92.1	88.7	66.5	83.4	91.0	87.8
Houlsby Adapter†	0.8M	90.3	94.7	96.3	91.5	87.7	66.3	72.9	91.5	86.4
LoRA†	0.8M	90.6	94.8	96.2	91.6	90.2	68.2	85.2	92.3	88.6
AdaMix Adapter	0.8M	**90.9**	**95.4**	**97.1**	**92.3**	**91.9**	**70.2**	**89.2**	**92.4**	**89.9**

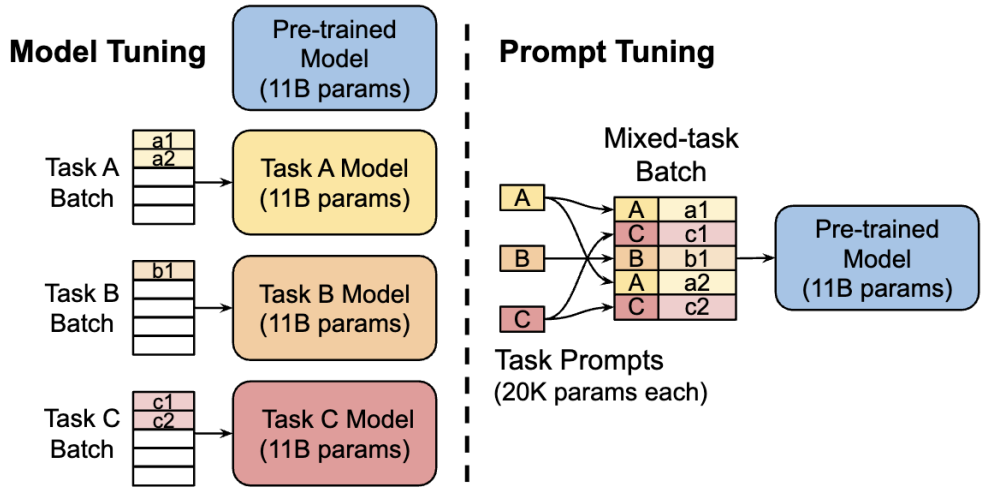
There are more PEFT methods to mix.

More Architectures...



Each layer: $h_i = \begin{cases} P_\theta[i,:], & \text{if } i \in \mathsf{P}_{\text{idx}}, \\ \text{LM}_\phi(z_i, h_{<i}), & \text{otherwise.} \end{cases}$ (3)

(a) Prefix-tuning



Only input layer: $[P_e; X_e] \in \mathbb{R}^{(p+n)\times e}$

(b) Prompt-tuning

Prepend learnable representations to the actual input

[1] Li and Liang. "*Prefix-tuning: Optimizing continuous prompts for generation.*" (ACL), 2021.
[2] Lester et al. "*The power of scale for parameter-efficient prompt tuning.*" Empirical Methods in Natural Language Processing (EMNLP), 2022.

Outline

- Model Efficiency
  - Parameter-efficient Fine-tuning
  - Model Compression
  - Inference Acceleration
- Data Efficiency
  - Zero/few-shot Learning
  - Knowledge Editing
  - Retrieval-augmented Generation

Model Compression

- PEFT methods allow fine-tuning pre-trained LLMs for specific downstream applications with limited resources.

- However, when applications themselves have limited computational resources, they may not afford to run fine-tuned LLMs.

- Therefore, **model compression** is needed.

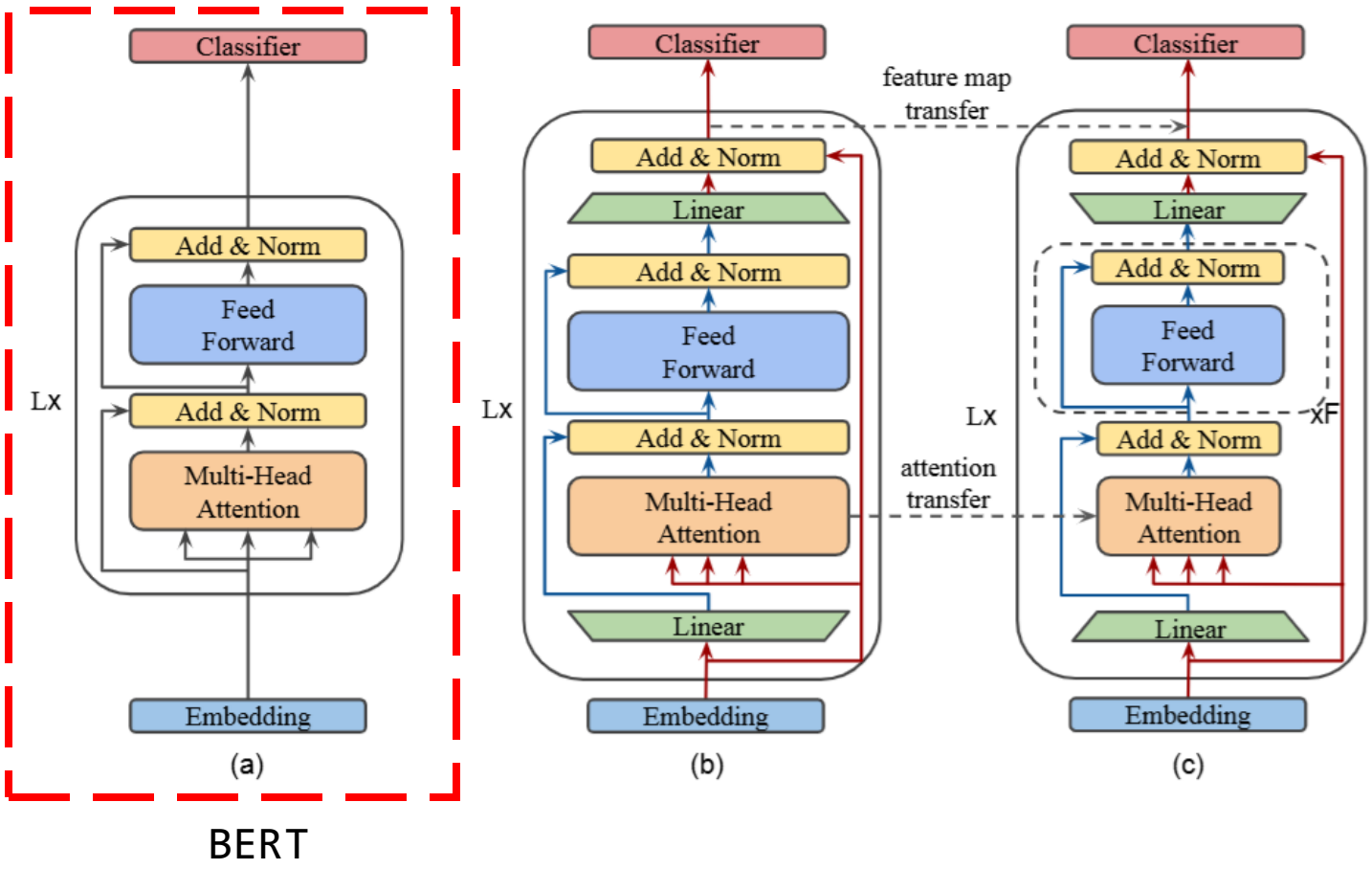
MobileBERT

- The compressed model to have strong performance while being small.
  - As deep as BERT but much thinner (bottleneck).

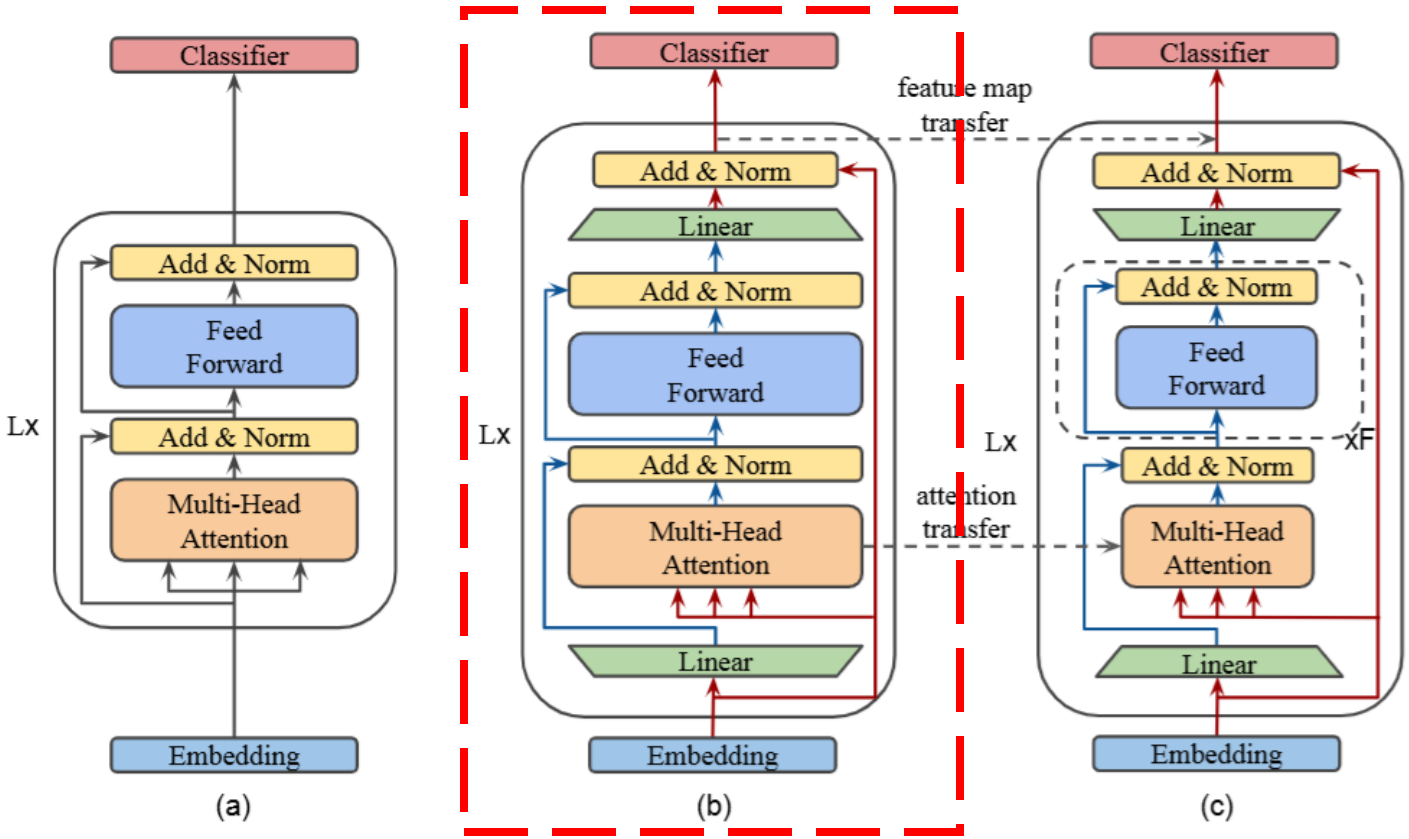
- It is hard to train a deep & thin network from the scratch.
  - Design a special teacher network for knowledge transfer (inverted-bottleneck).

Sun et al. "*MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices*." Proceedings of ACL 2020.

MobileBERT

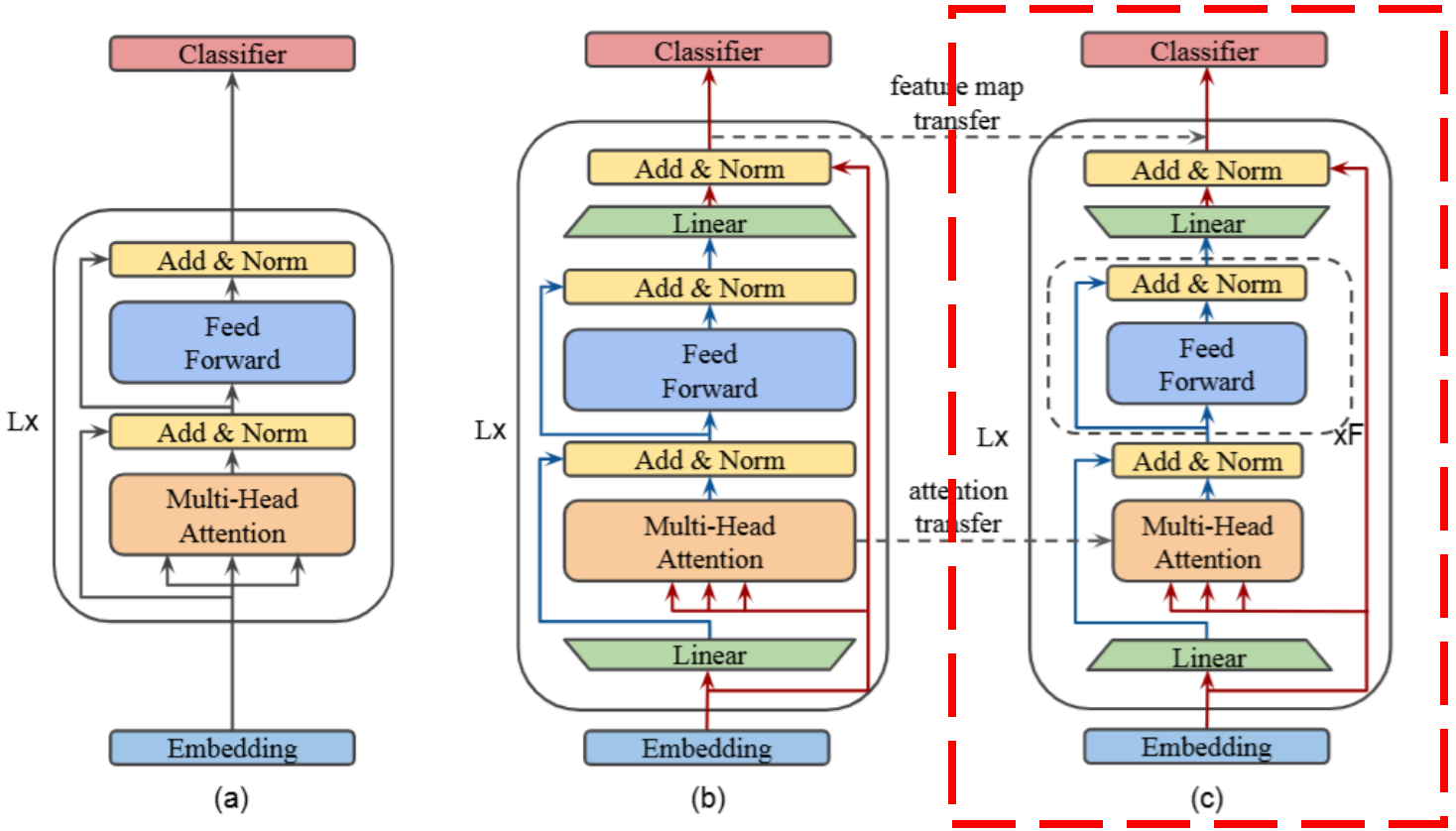

Classifier
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Lx
Embedding
(a)
BERT
Classifier
Add & Norm
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Linear
Lx
Embedding
(b)
feature map transfer
attention transfer
Classifier
Add & Norm
Linear
Add & Norm
Feed Forward
xF
Add & Norm
Multi-Head Attention
Linear
Lx
Embedding
(c)

MobileBERT



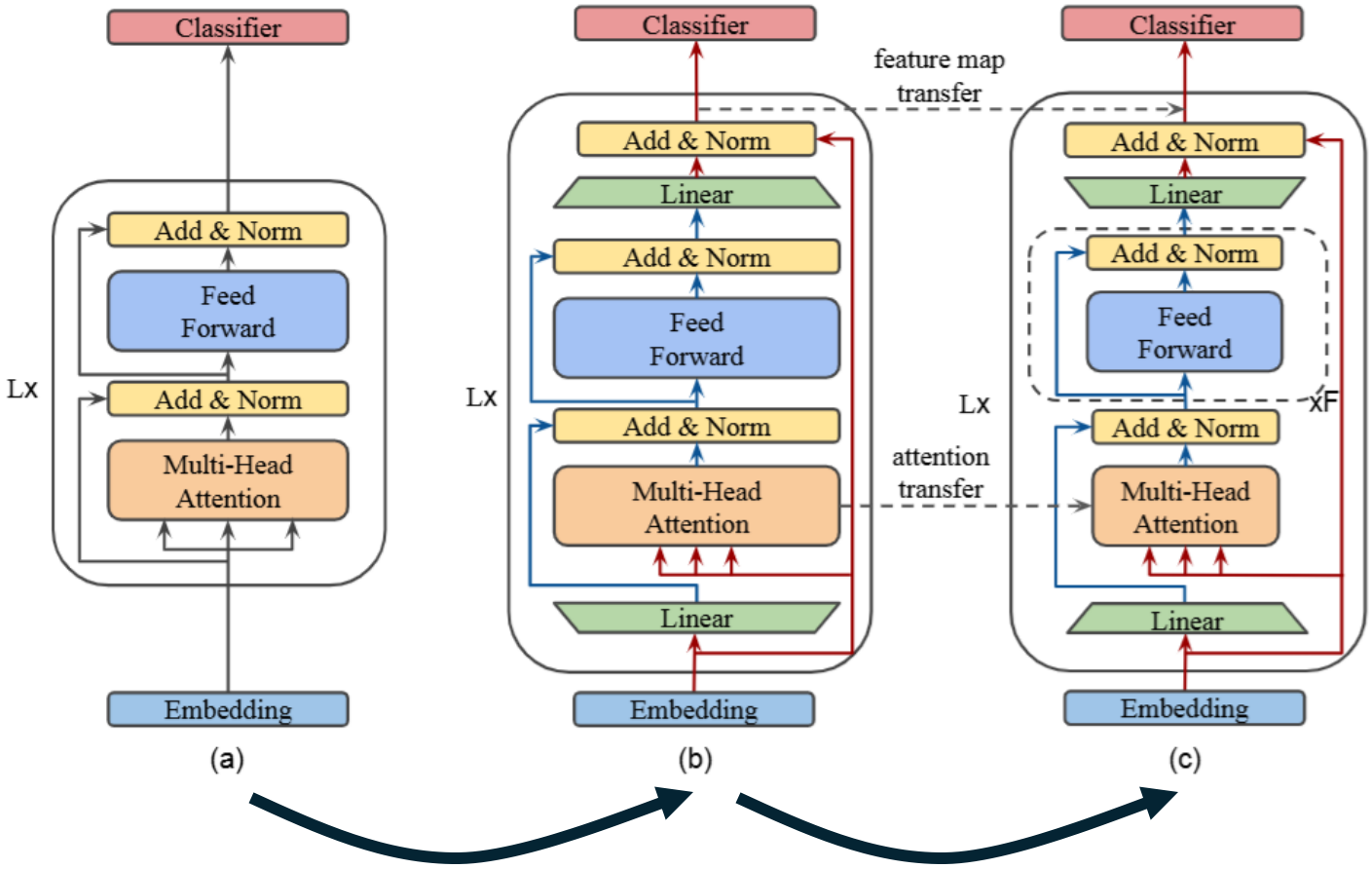
IB-BERT
(Teacher Model)

MobileBERT



MobileBERT
(Student Model)

MobileBERT



First Step: training teacher model

Second Step: training student model

MobileBERT

- IB-BERT is a little lighter than BERT while matching the hidden space dimension of MobileBERT.

			$\text{BERT}_{\text{LARGE}}$	$\text{BERT}_{\text{BASE}}$	$\text{IB-BERT}_{\text{LARGE}}$	MobileBERT	$\text{MobileBERT}_{\text{TINY}}$
embedding		$h_{\text{embedding}}$	1024	768	128		
			no-op	no-op	3-convolution		
		h_{inter}	1024	768	512		
body	Linear	h_{input} h_{output}	$\left[\begin{pmatrix}1024\\16\\1024\end{pmatrix}\begin{pmatrix}1024\\4096\\1024\end{pmatrix}\right]\times 24$	$\left[\begin{pmatrix}768\\12\\768\end{pmatrix}\begin{pmatrix}768\\3072\\768\end{pmatrix}\right]\times 12$	$\left[\begin{pmatrix}512\\1024\end{pmatrix}\begin{pmatrix}512\\4\\1024\end{pmatrix}\begin{pmatrix}1024\\4096\\1024\end{pmatrix}\begin{pmatrix}1024\\512\end{pmatrix}\right]\times 24$	$\left[\begin{pmatrix}512\\128\end{pmatrix}\begin{pmatrix}512\\4\\128\end{pmatrix}\begin{pmatrix}128\\512\\128\end{pmatrix}\times 4\begin{pmatrix}128\\512\end{pmatrix}\right]\times 24$	$\left[\begin{pmatrix}512\\128\end{pmatrix}\begin{pmatrix}128\\4\\128\end{pmatrix}\begin{pmatrix}128\\512\\128\end{pmatrix}\times 2\begin{pmatrix}128\\512\end{pmatrix}\right]\times 24$
	MHA	h_{input} #Head h_{output}					
	FFN	h_{input} h_{FFN} h_{output}					
	Linear	h_{input} h_{output}					
#Params			334M	109M	293M	25.3M	15.1M

MobileBERT

- IB-BERT is a little lighter than BERT while matching the hidden space dimension of MobileBERT.

			$\text{BERT}_{\text{LARGE}}$	$\text{BERT}_{\text{BASE}}$	$\text{IB-BERT}_{\text{LARGE}}$	MobileBERT	$\text{MobileBERT}_{\text{TINY}}$
embedding		$h_{\text{embedding}}$	1024	768		128	
			no-op	no-op		3-convolution	
		h_{inter}	1024	768		512	
body	Linear	h_{input} h_{output}	$\begin{bmatrix} 1024 \\ 1024 \end{bmatrix} \times 24$	$\begin{bmatrix} 768 \\ 768 \end{bmatrix} \times 12$	$\begin{bmatrix} 512 \\ 1024 \end{bmatrix} \times 24$	$\begin{bmatrix} 512 \\ 128 \end{bmatrix} \times 24$	$\begin{bmatrix} 512 \\ 128 \end{bmatrix} \times 24$
	MHA	h_{input} #Head h_{output}			$\begin{bmatrix} 512 \\ 4 \\ 1024 \end{bmatrix}$	$\begin{bmatrix} 512 \\ 4 \\ 128 \end{bmatrix}$	$\begin{bmatrix} 128 \\ 4 \\ 128 \end{bmatrix}$
	FFN	h_{input} h_{FFN} h_{output}			$\begin{bmatrix} 1024 \\ 4096 \\ 1024 \end{bmatrix}$	$\begin{bmatrix} 128 \\ 512 \\ 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 128 \\ 512 \\ 128 \end{bmatrix} \times 2$
	Linear	h_{input} h_{output}			$\begin{bmatrix} 1024 \\ 512 \end{bmatrix}$	$\begin{bmatrix} 128 \\ 512 \end{bmatrix}$	$\begin{bmatrix} 128 \\ 512 \end{bmatrix}$
#Params			334M	109M	293M	25.3M	15.1M

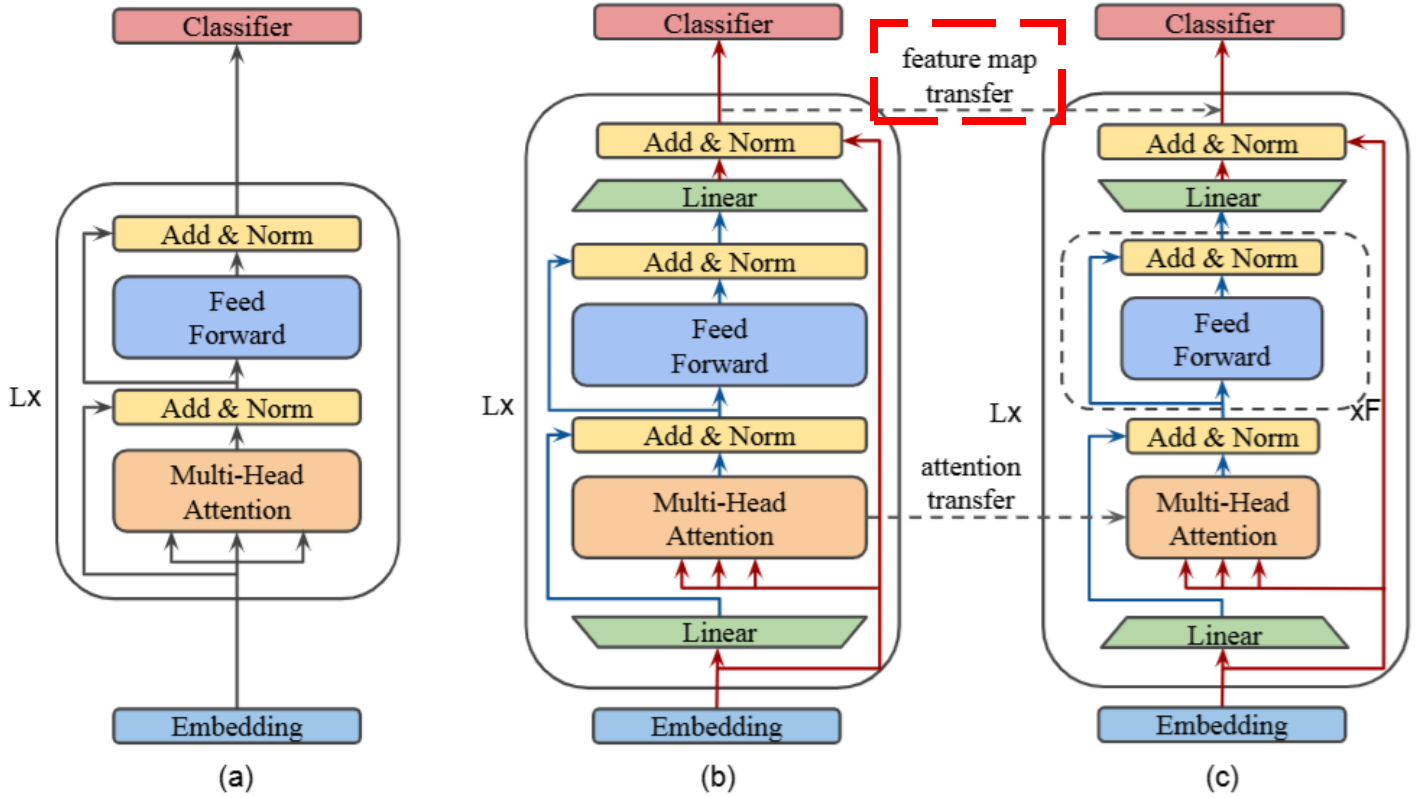
MobileBERT

- IB-BERT is a little lighter than BERT while matching the hidden space dimension of MobileBERT.

			$\text{BERT}_{\text{LARGE}}$	$\text{BERT}_{\text{BASE}}$	$\text{IB-BERT}_{\text{LARGE}}$	MobileBERT	$\text{MobileBERT}_{\text{TINY}}$
embedding		$h_{\text{embedding}}$	1024	768	128		
			no-op	no-op	3-convolution		
		h_{inter}	1024	768	512		
body	Linear	h_{input} h_{output}	$\begin{bmatrix} 1024 \\ 16 \\ 1024 \\ 1024 \\ 4096 \\ 1024 \end{bmatrix} \times 24$	$\begin{bmatrix} 768 \\ 12 \\ 768 \\ 768 \\ 3072 \\ 768 \end{bmatrix} \times 12$	$\begin{bmatrix} 512 \\ 1024 \\ 512 \\ 4 \\ 1024 \\ 1024 \\ 4096 \\ 1024 \\ 1024 \\ 512 \end{bmatrix} \times 24$	$\begin{bmatrix} 512 \\ 128 \\ 512 \\ 4 \\ 128 \\ 128 \\ 512 \\ 128 \times 4 \\ 128 \\ 512 \end{bmatrix} \times 24$	$\begin{bmatrix} 512 \\ 128 \\ 128 \\ 4 \\ 128 \\ 128 \\ 512 \\ 128 \times 2 \\ 128 \\ 512 \end{bmatrix} \times 24$
	MHA	h_{input} #Head h_{output}					
	FFN	h_{input} h_{FFN} h_{output}					
	Linear	h_{input} h_{output}					
#Params			334M	109M	293M	25.3M	15.1M

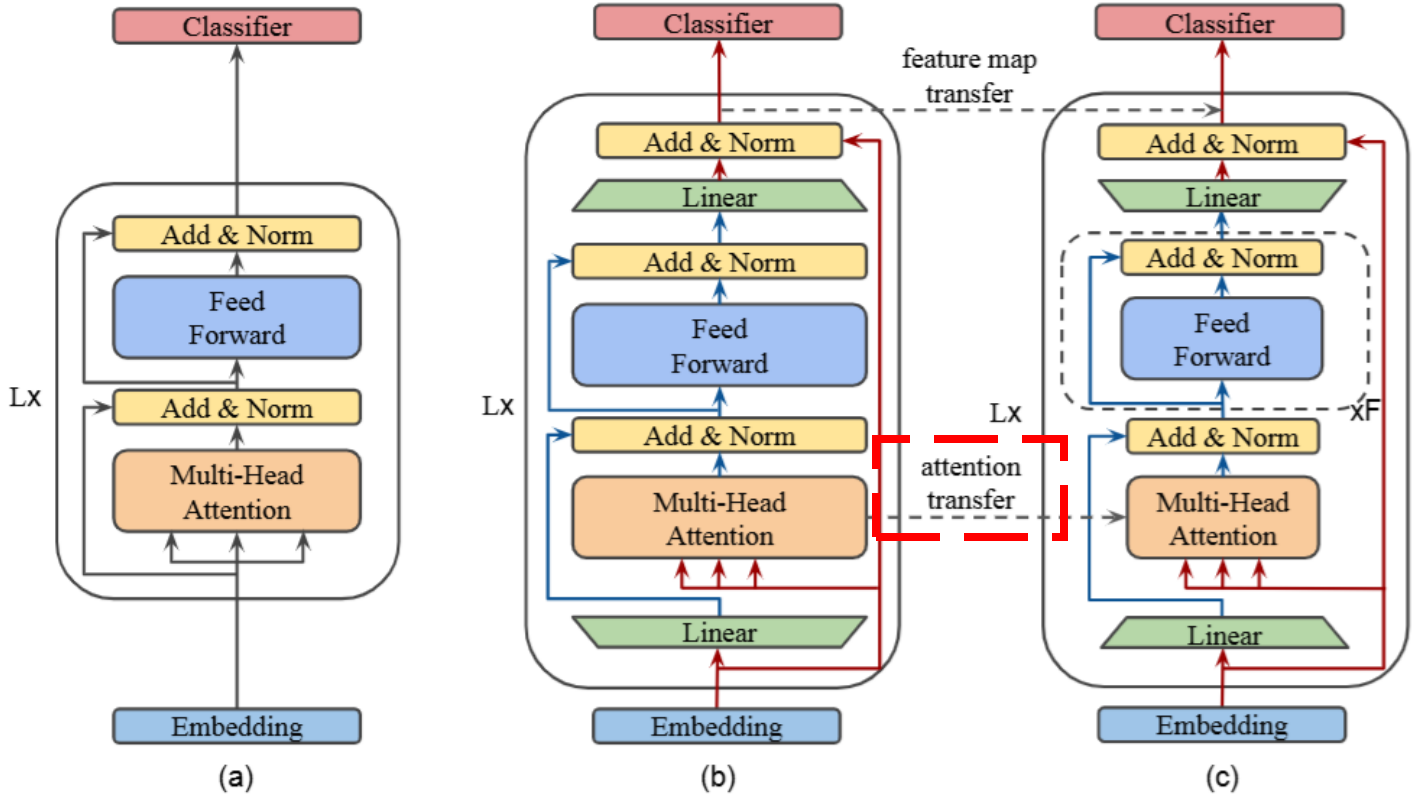
MobileBERT

- MSE loss: $\mathcal{L}_{FMT}^{\ell} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N}\left(H_{t,\ell,n}^{tr} - H_{t,\ell,n}^{st}\right)^2$



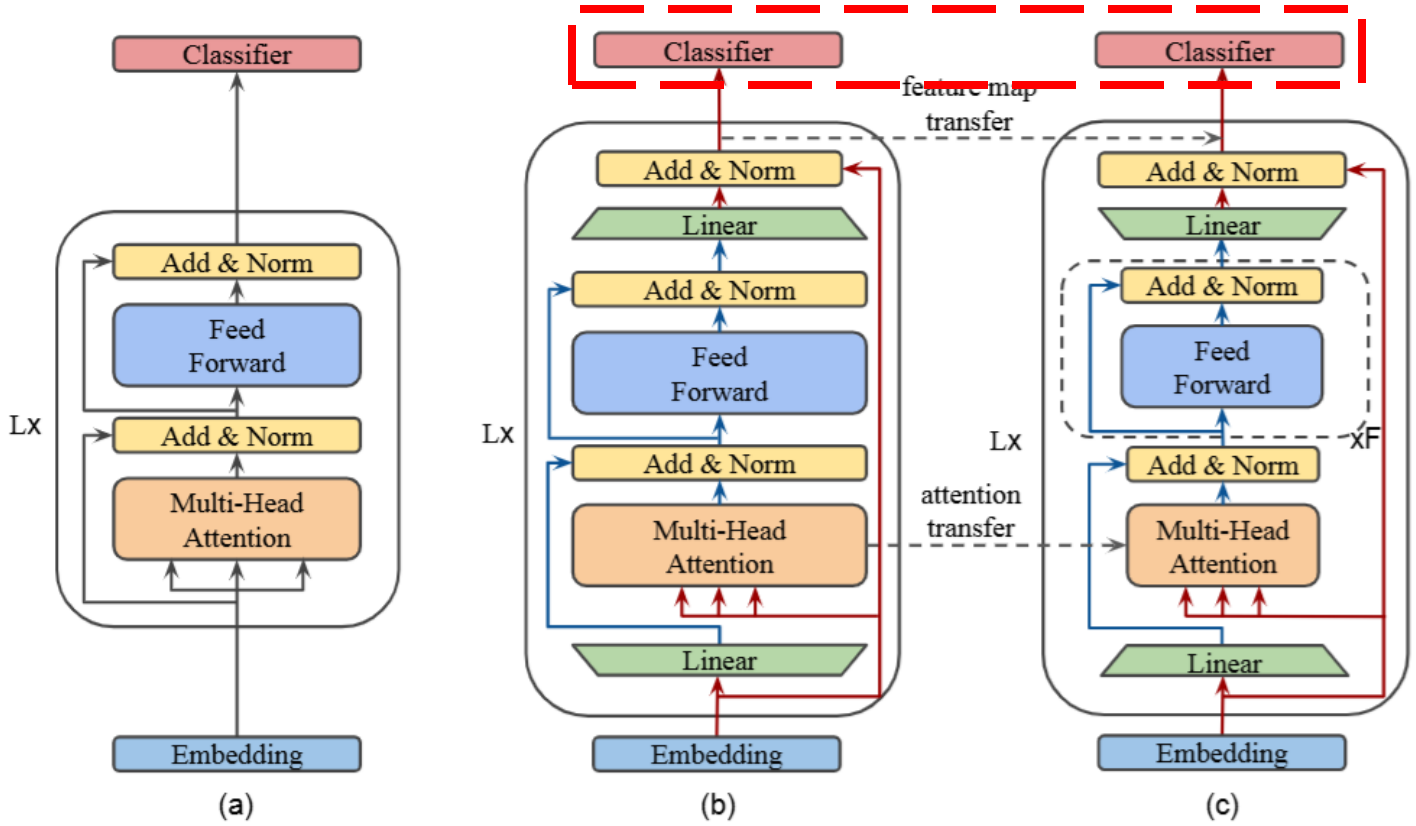
MobileBERT

- KL penalty: $\mathcal{L}_{AT}^{\ell} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N} D_{KL}(a_{t,\ell,n}^{tr} || a_{t,\ell,n}^{st})$



MobileBERT

- $\mathcal{L}_{PD} = \alpha\mathcal{L}_{MLM} + (1-\alpha)\mathcal{L}_{KD} + \mathcal{L}_{NSP}$

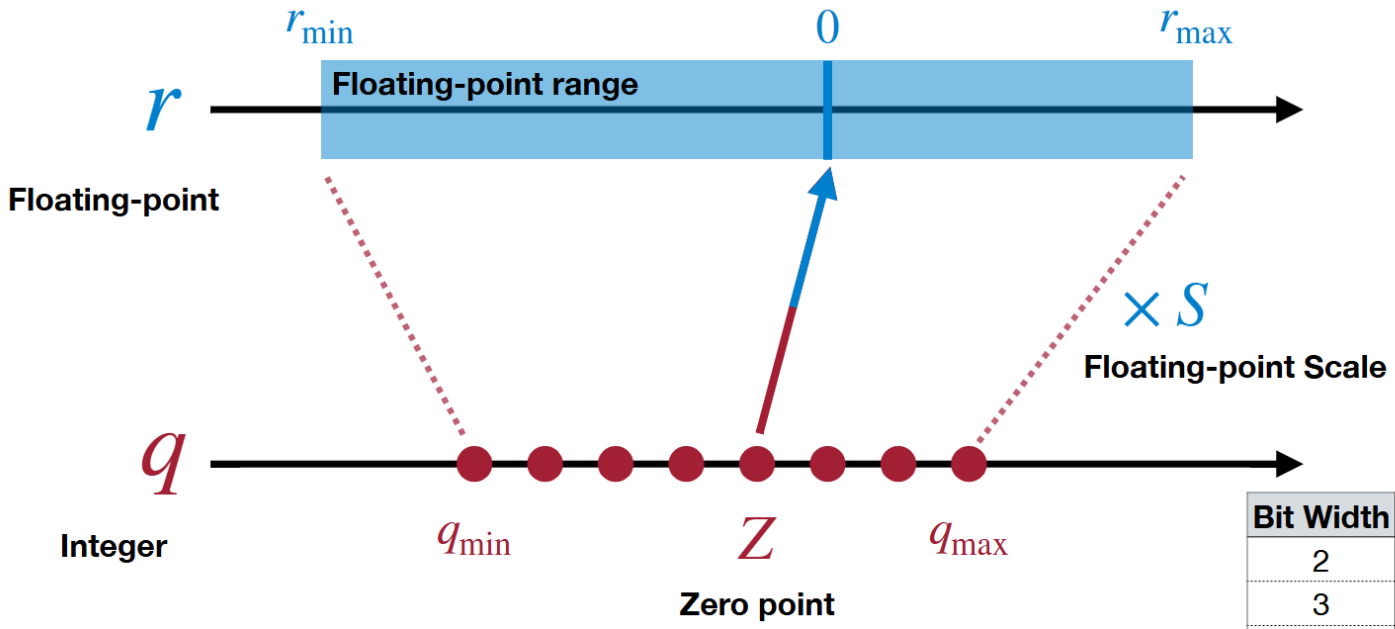


MobileBERT

	#Params	#FLOPS	Latency	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 5.7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	GLUE
ELMo-BiLSTM-Attn	-	-	-	33.6	90.4	84.4	72.3	63.1	74.1/74.5	79.8	58.9	70.0
OpenAI GPT	109M	-	-	47.2	93.1	87.7	84.8	70.1	80.7/80.6	87.2	69.1	76.9
$\text{BERT}_{\text{BASE}}$	109M	22.5B	342 ms	**52.1**	**93.5**	**88.9**	**85.8**	71.2	**84.6/83.4**	90.5	66.4	78.3
$\text{BERT}_{\text{BASE}}$-6L-PKD*	66.5M	11.3B	-	-	92.0	85.0	-	70.7	81.5/81.0	89.0	65.5	-
$\text{BERT}_{\text{BASE}}$-4L-PKD†*	52.2M	7.6B	-	24.8	89.4	82.6	79.8	70.2	79.9/79.3	85.1	62.3	-
$\text{BERT}_{\text{BASE}}$-3L-PKD*	45.3M	5.7B	-	-	87.5	80.7	-	68.1	76.7/76.3	84.7	58.2	-
$\text{DistilBERT}_{\text{BASE}}$-6L†	62.2M	11.3B	-	-	92.0	85.0		70.7	81.5/81.0	89.0	65.5	-
$\text{DistilBERT}_{\text{BASE}}$-4L†	52.2M	7.6B	-	32.8	91.4	82.4	76.1	68.5	78.9/78.0	85.2	54.1	-
TinyBERT*	14.5M	1.2B	-	43.3	92.6	86.4	79.9	**71.3**	82.5/81.8	87.7	62.9	75.4
$\text{MobileBERT}_{\text{TINY}}$	15.1M	3.1B	40 ms	46.7	91.7	87.9	80.1	68.9	81.5/81.6	89.5	65.1	75.8
MobileBERT	25.3M	5.7B	62 ms	50.5	92.8	88.8	84.4	70.2	83.3/82.6	90.6	66.2	77.7
MobileBERT w/o OPT	25.3M	5.7B	192 ms	51.1	92.6	88.8	84.8	70.5	84.3/**83.4**	**91.6**	**70.4**	**78.5**

Linear Quantization

An affine mapping of integers to real numbers $r = S(q - Z)$

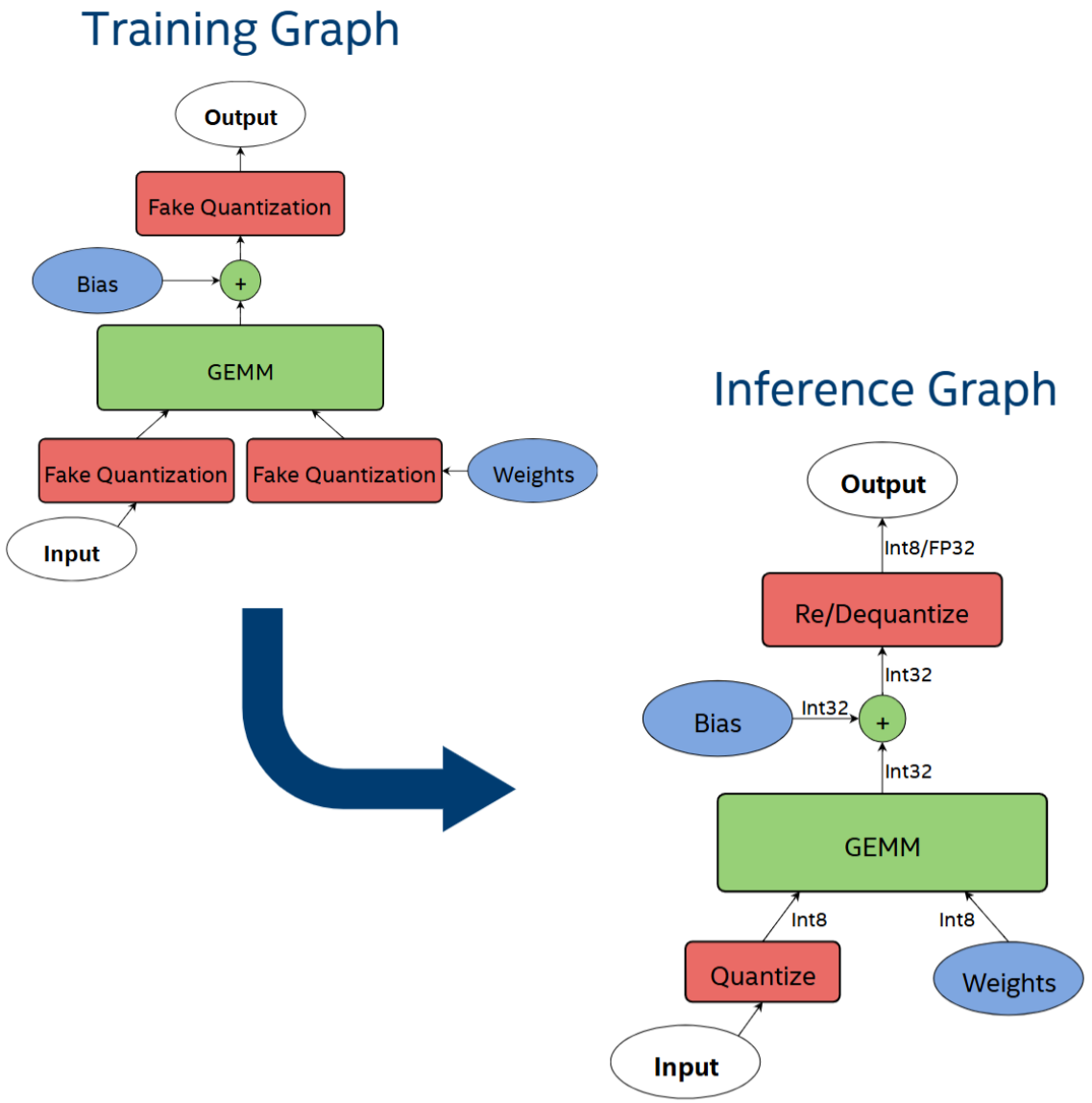


Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

Credits: *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models.*

Q8BERT

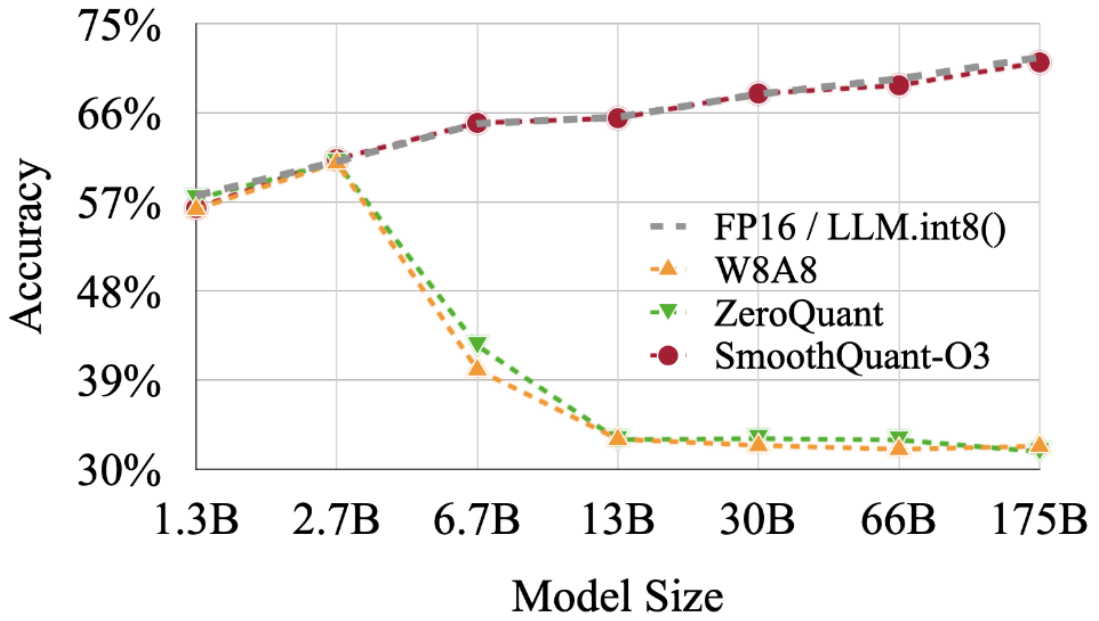
- **Train** Neural Networks (NN) to be **quantized-aware** at the inference stage.
- Fake quantization is used to introduce the quantization error training.
- Apply Fake Quantization on all General Matrix Multiply (GEMM) & Word/Position Embedding layers.
- Sensitive operations are kept in FP32 (Softmax, LN, GELU).



Zafrir et al. "*Q8bert: Quantized 8bit bert*." Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS). IEEE, 2019.

Quantization Difficulty of **LLMs**

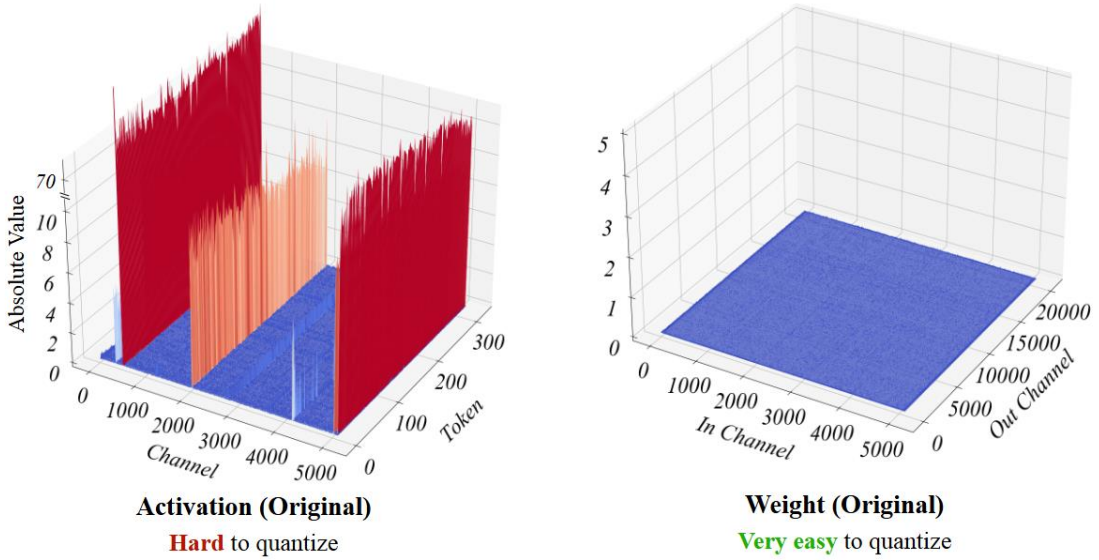
- Systematic outliers emerge in activations when scaling up to **LLMs beyond 6.7B.**
- Naive quantization methods will destroy the accuracy.



Credits: *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models.*

Quantization Difficulty of LLMs

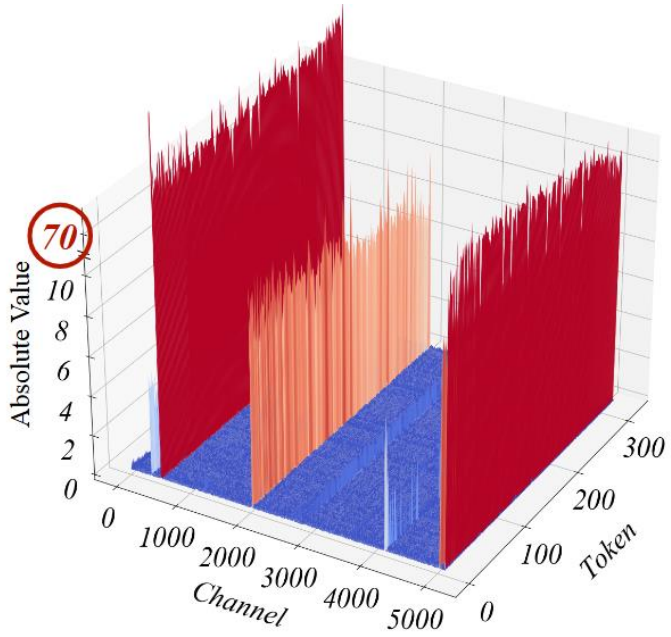
- **Activations are harder to quantize than weights**
  - Previous work has shown quantizing the weights of LLMs with INT8 or even INT4 doesn’t degrade accuracy.

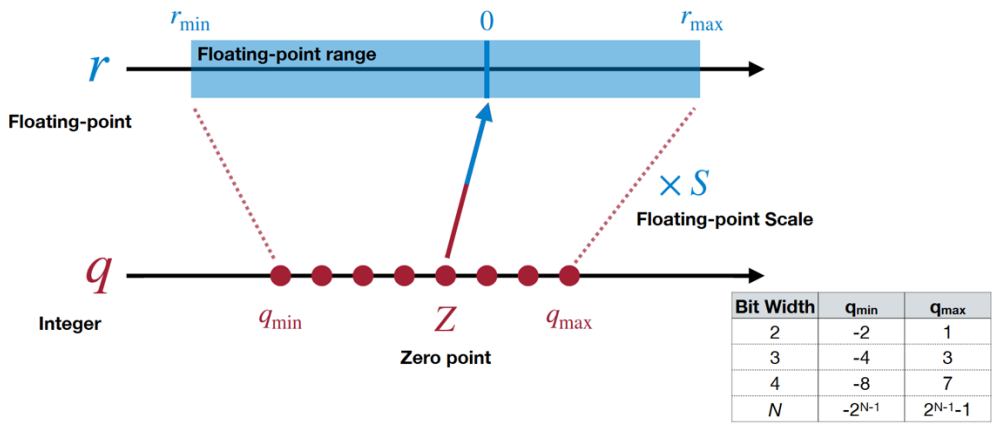


Credits: *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models.*

Quantization Difficulty of LLMs

- Outliers make activation quantization difficult
  - The scale of outliers is **~100x larger** than most of other activations.
  - If we use INT8 quantization, normal activations will be **zero**.



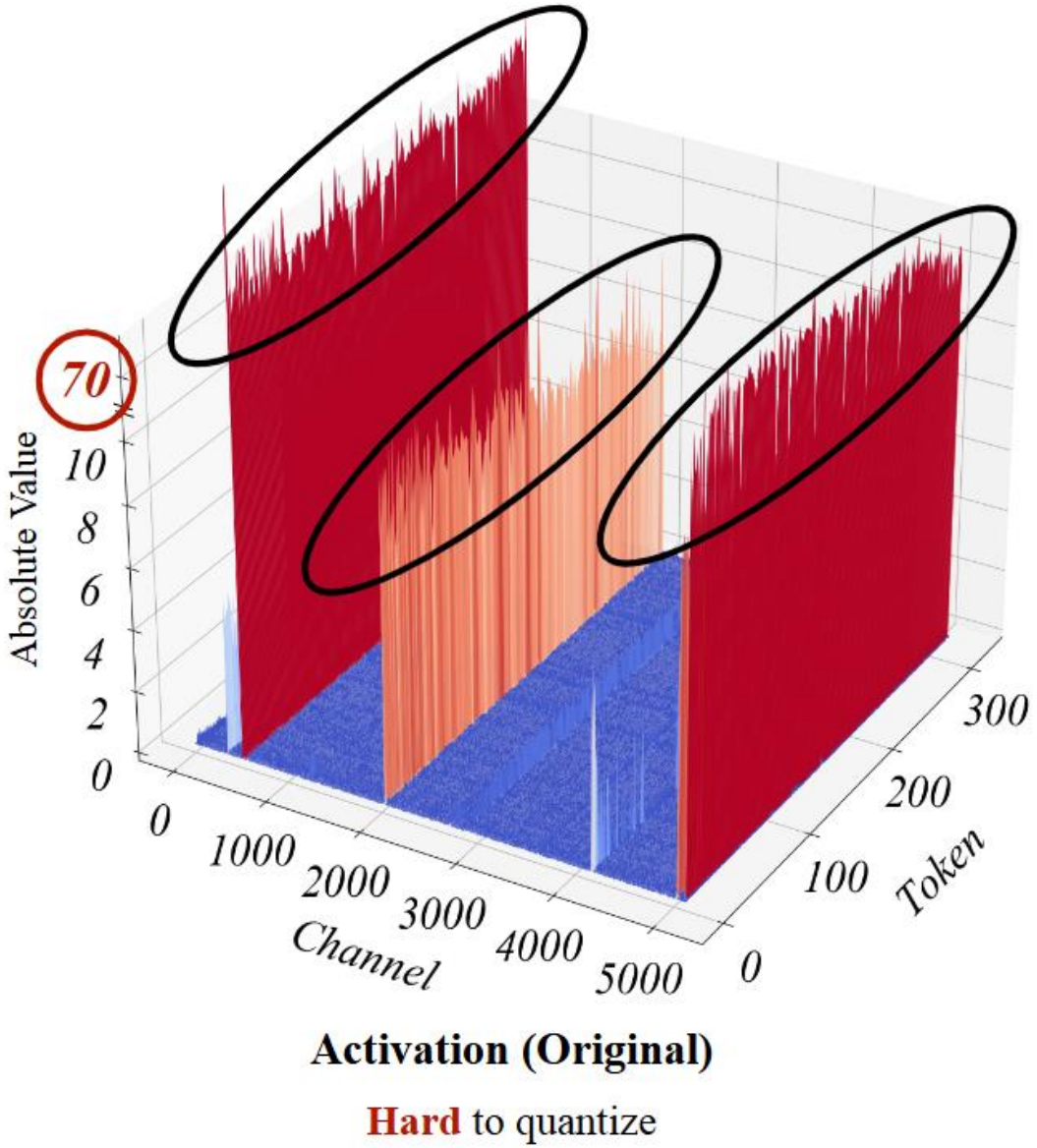


Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

Credits: *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models.*

Quantization Difficulty of LLMs

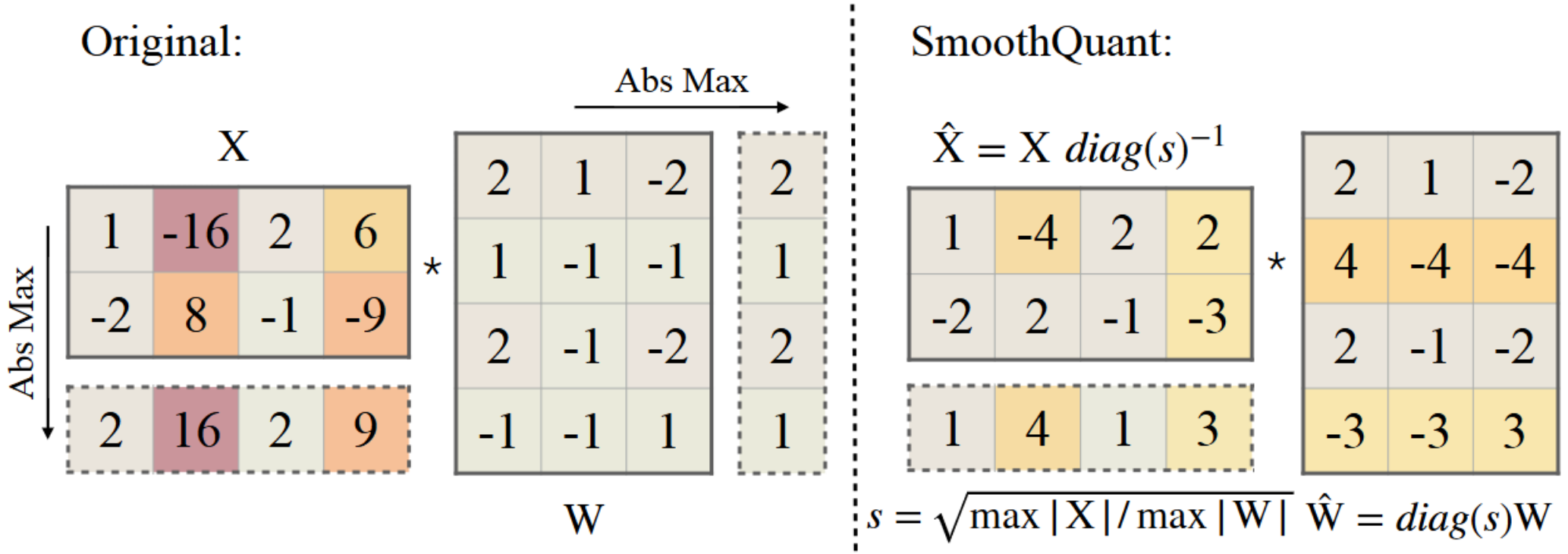
- Outliers persist in fixed channels
  - Fixed channels have outliers, and the outlier channels are persistently large.



Credits: *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models.*

SmoothQuant

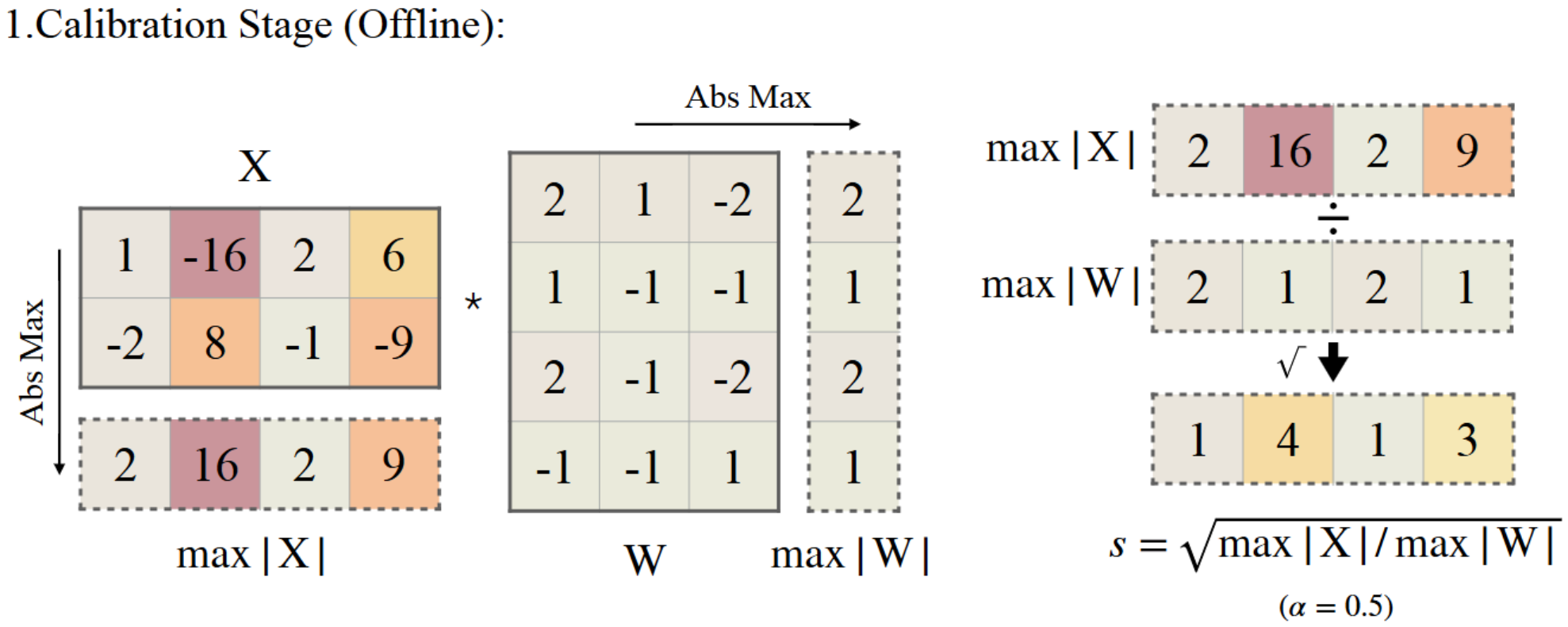
- Activation Smoothing
  - $\boldsymbol{s}_j = \frac{\max(|\boldsymbol{X}_j|)^{\alpha}}{\max(|\boldsymbol{W}_j|)^{1-\alpha}}, j = 1,2,\ldots,C_i$
  - $\boldsymbol{Y} = (\boldsymbol{X}diag(\boldsymbol{s})^{-1}) \cdot (diag(\boldsymbol{s})\boldsymbol{W}) = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{W}}$



Xiao et al. "*Smoothquant: Accurate and efficient post-training quantization for large language models*." International Conference on Machine Learning (ICML), 2023.

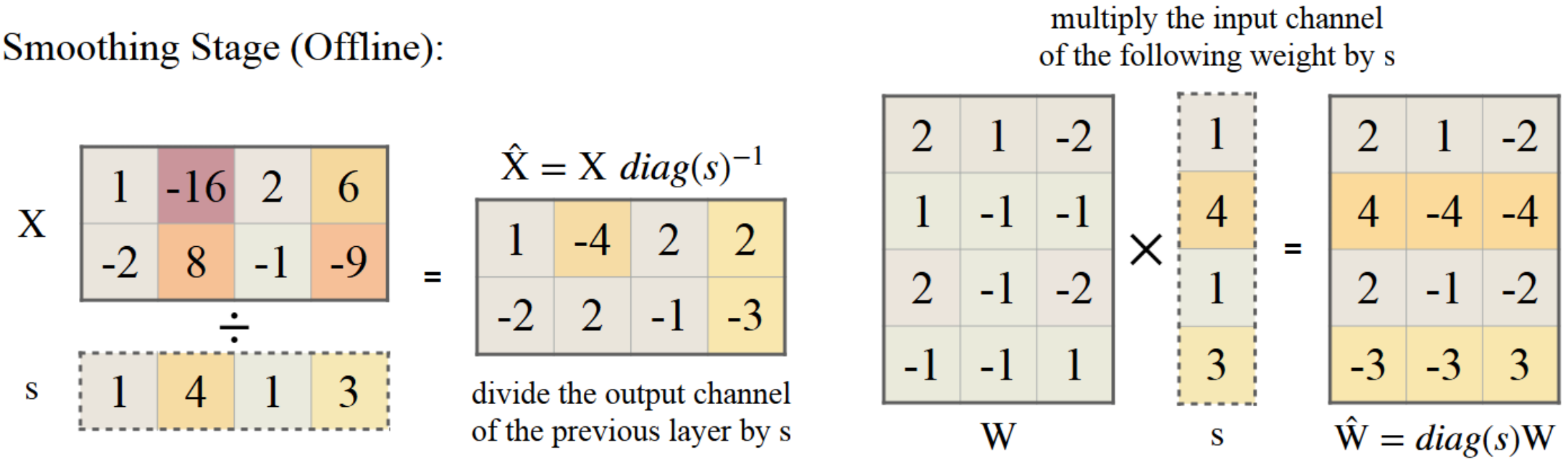
SmoothQuant

- Activation Smoothing
  - Calibration Stage (Offline)
  - $\boldsymbol{s}_j = \frac{\max(|\boldsymbol{X}_j|)^{\alpha}}{\max(|\boldsymbol{W}_j|)^{1-\alpha}}, j = 1,2,\dots,C_i$
  - $\boldsymbol{Y} = (\boldsymbol{X}diag(\boldsymbol{s})^{-1}) \cdot (diag(\boldsymbol{s})\boldsymbol{W}) = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{W}}$



SmoothQuant

- Activation Smoothing
  - Smoothing Stage (Offline)
  - $\boldsymbol{s}_j = \frac{\max(|\boldsymbol{X}_j|)^{\alpha}}{\max(|\boldsymbol{W}_j|)^{1-\alpha}}, j = 1,2,\ldots,C_i$
  - $\boldsymbol{Y} = (\boldsymbol{X}diag(\boldsymbol{s})^{-1}) \cdot (diag(\boldsymbol{s})\boldsymbol{W}) = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{W}}$



SmoothQuant

- Activation Smoothing
  - Inference (deployed model)
  - $\boldsymbol{s}_j = \frac{\max(|\boldsymbol{X}_j|)^{\alpha}}{\max(|\boldsymbol{W}_j|)^{1-\alpha}}, j = 1,2,\ldots,C_i$
  - $\boldsymbol{Y} = (\boldsymbol{X}diag(\boldsymbol{s})^{-1}) \cdot (diag(\boldsymbol{s})\boldsymbol{W}) = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{W}}$

3. Inference (deployed model):

$\hat{X}$

1	-4	2	2
-2	2	-1	-3

At runtime, the activations are smooth and easy to quantize

$*$

2	1	-2
4	-4	-4
2	-1	-2
-3	-3	3

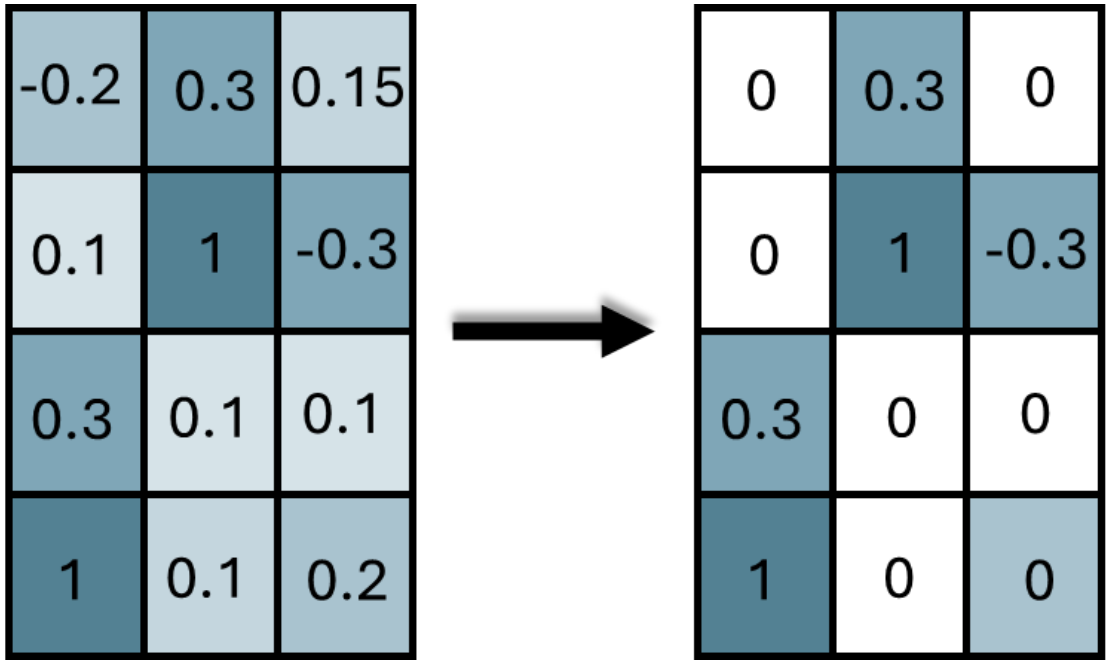
$\hat{W}$

SmoothQuant

OPT-175B	LAMBADA	HellaSwag	PIQA	WinoGrande	OpenBookQA	RTE	COPA	**Average↑**	**WikiText↓**
FP16	74.7%	59.3%	79.7%	72.6%	34.0%	59.9%	88.0%	66.9%	10.99
W8A8	0.0%	25.6%	53.4%	50.3%	14.0%	49.5%	56.0%	35.5%	93080
ZeroQuant	0.0%*	26.0%	51.7%	49.3%	17.8%	50.9%	55.0%	35.8%	84648
`LLM.int8()`	74.7%	59.2%	79.7%	72.1%	34.2%	60.3%	87.0%	66.7%	11.10
Outlier Suppression	0.00%	25.8%	52.5%	48.6%	16.6%	53.4%	55.0%	36.0%	96151
SmoothQuant-O1	74.7%	59.2%	79.7%	71.2%	33.4%	58.1%	89.0%	66.5%	11.11
SmoothQuant-O2	75.0%	59.0%	79.2%	71.2%	33.0%	59.6%	88.0%	66.4%	11.14
SmoothQuant-O3	74.6%	58.9%	79.7%	71.2%	33.4%	59.9%	90.0%	66.8%	11.17

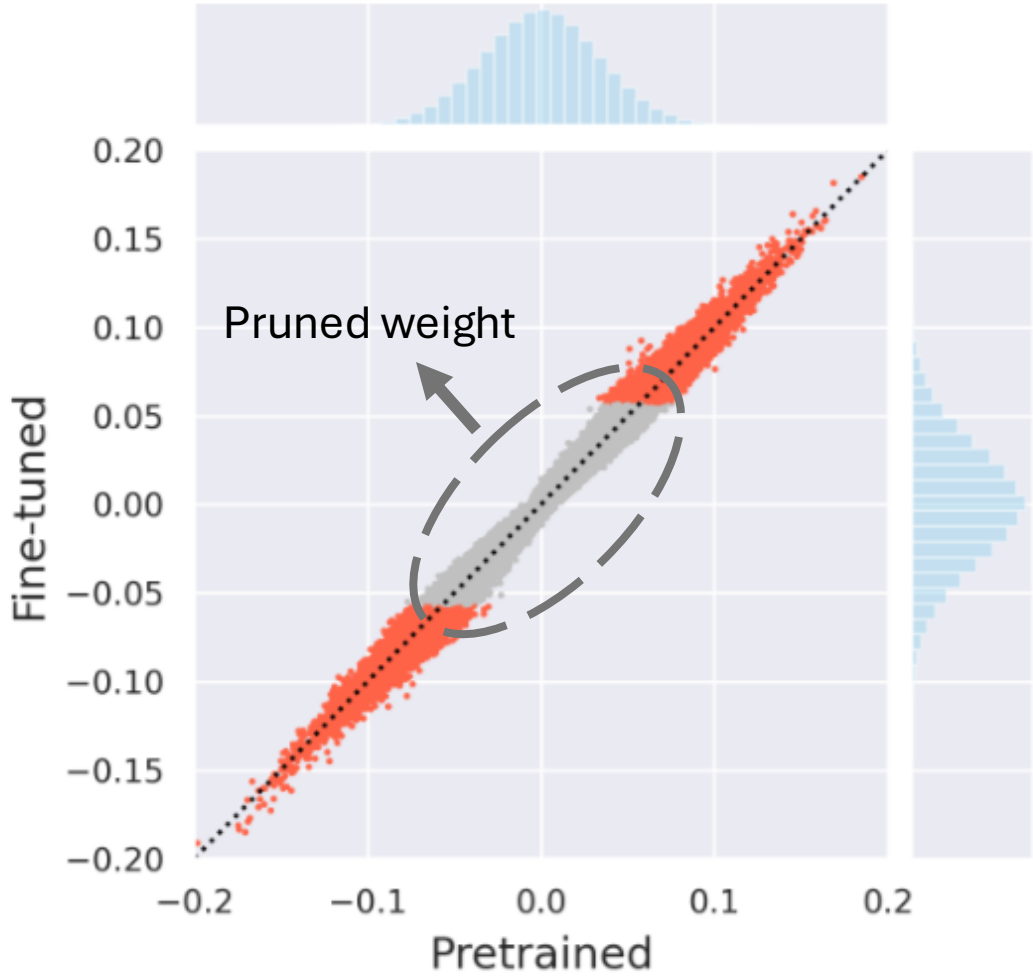
Model Pruning

- Score-based pruning
  - Select the $v\%$ highest values in S (important score)
  - $\mathrm{Top}_v(S)_{i,j} = \begin{cases} 1, S_{i,j} \text{ in top } v\% \\ 0, o.w. \end{cases}$
- Magnitude pruning
  - Use $|W_{i,j}|$ as the score.



Challenge of Magnitude Pruning

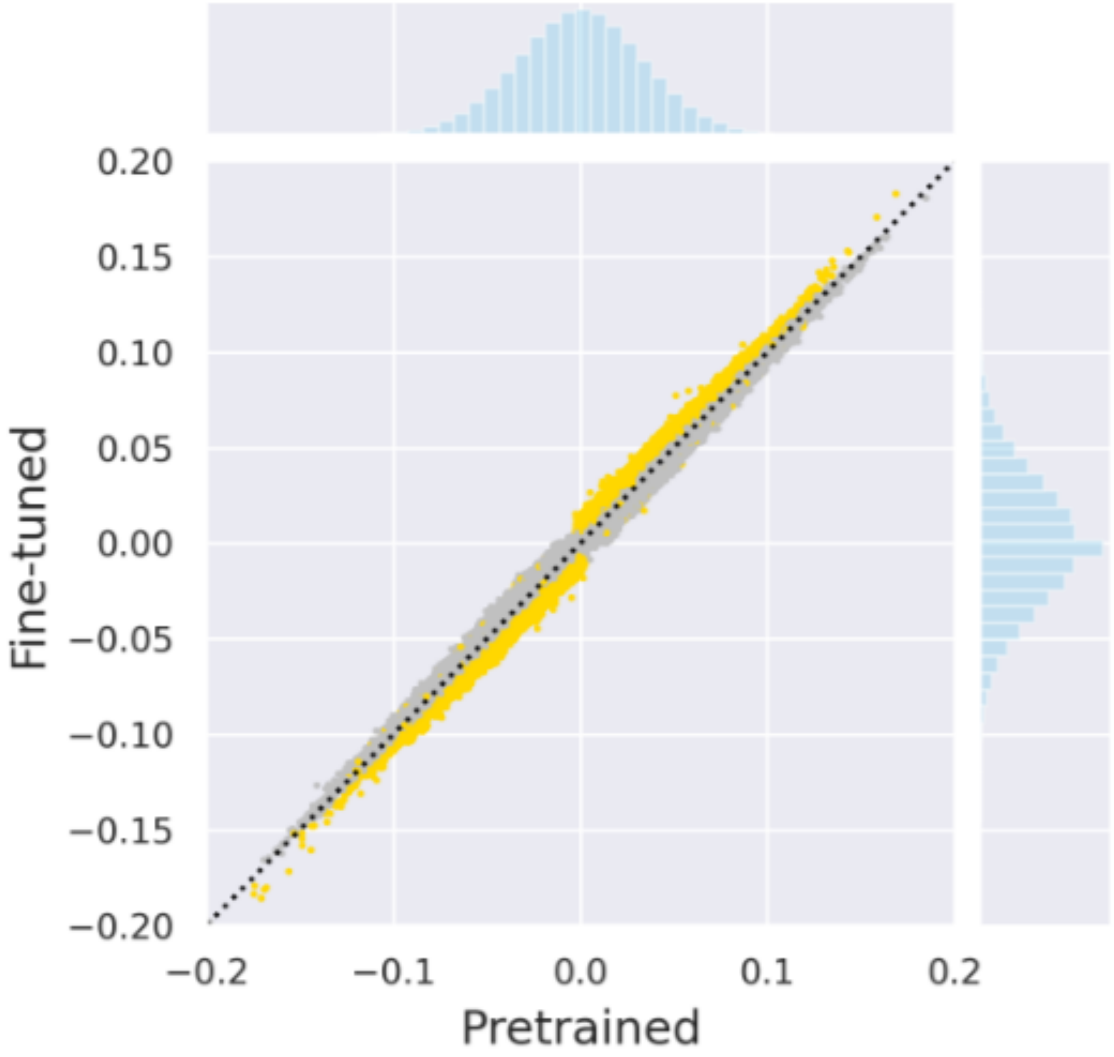
- The weights stay close to their pre-trained values which limits the adaptivity of magnitude pruning



Credits: *Movement Pruning: Adaptive Sparsity by Fine-Tuning.*

Movement Pruning

- Learn both the weights $\boldsymbol{W}$ and their importance scores $\boldsymbol{S}$
  - $a = (\boldsymbol{W} \odot \boldsymbol{M})x$, where $\boldsymbol{M} = \mathrm{Top}_v(\boldsymbol{S})$.
  - $\frac{\partial \mathcal{L}}{\partial \boldsymbol{S}_{i,j}} = \frac{\partial \mathcal{L}}{\partial a_i} \frac{\partial a_i}{\partial \boldsymbol{S}_{i,j}} = \frac{\partial \mathcal{L}}{\partial a_i} \boldsymbol{W}_{i,j} x_j$.
- Movement pruning selects weights that are moving away from 0.



Sanh et al. "*Movement pruning: Adaptive sparsity by fine-tuning.*" Advances in neural information processing systems, 2020

Movement Pruning

	BERT base fine-tuned	Remaining Weights (%)	MaP	L_0 Regu	MvP	soft MvP
SQuAD - Dev EM/F1	80.4/88.1	10%	67.7/78.5	69.9/80.0	**71.9/81.7**	71.3/81.5
		3%	40.1/54.5	61.2/73.3	65.2/76.3	**69.5/79.9**
MNLI - Dev acc/MM acc	84.5/84.9	10%	77.8/79.0	77.9/78.5	79.3/79.5	**80.7/81.1**
		3%	68.9/69.8	75.1/75.4	76.1/76.7	**79.0/79.6**
QQP - Dev acc/F1	91.4/88.4	10%	78.8/75.1	87.5/81.9	89.1/85.5	**90.5/87.1**
		3%	72.1/58.4	86.5/81.0	85.6/81.0	**89.3/85.6**

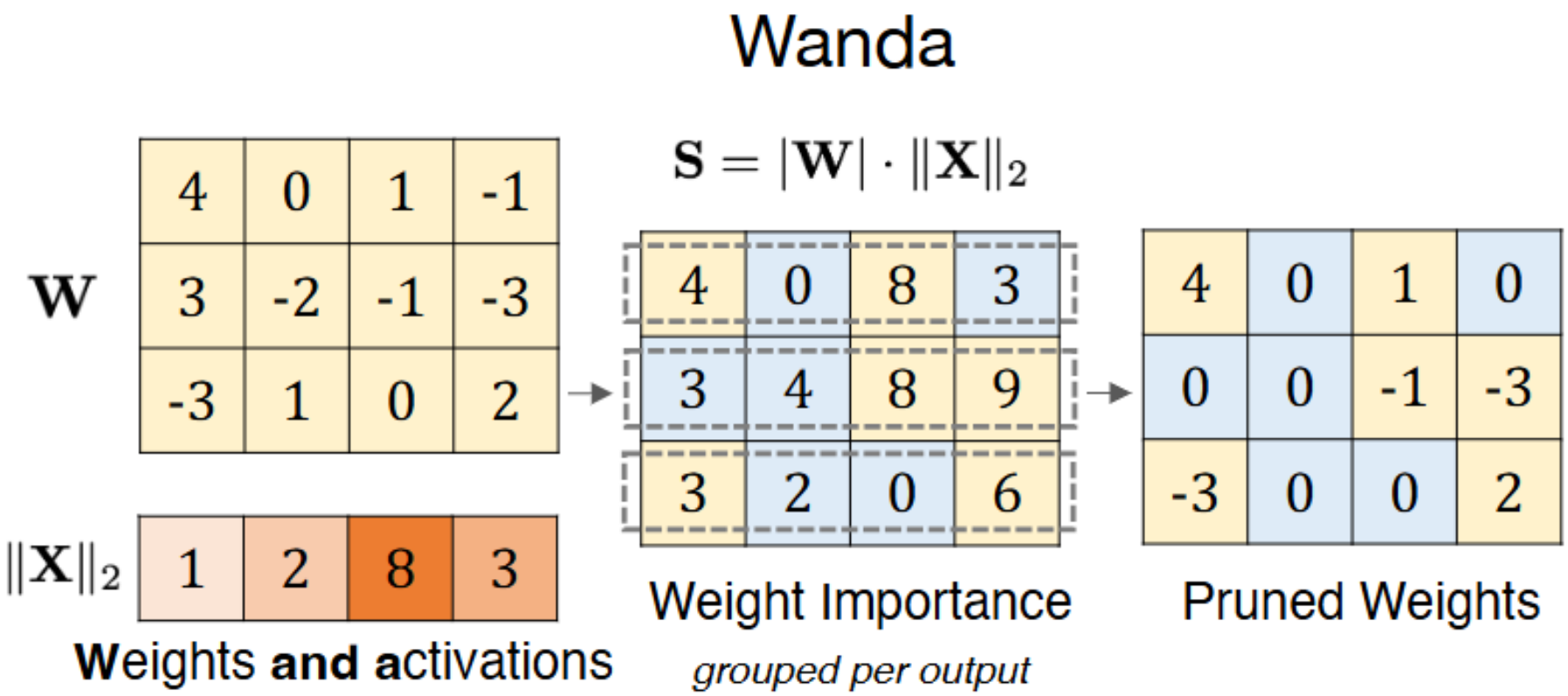
Performance at high sparsity levels.

	BERT base fine-tuned	Remaining Weights (%)	MaP	L_0 Regu	MvP	soft MvP
SQuAD - Dev EM/F1	80.4/88.1	10%	70.2/80.1	72.4/81.9	75.6/84.3	**76.6/84.9**
		3%	45.5/59.6	64.3/75.8	67.5/78.0	**72.7/82.3**
MNLI - Dev acc/MM acc	84.5/84.9	10%	78.3/79.3	78.7/79.7	80.1/80.4	**81.2/81.8**
		3%	69.4/70.6	76.0/76.2	76.5/77.4	**79.5/80.1**
QQP - Dev acc/F1	91.4/88.4	10%	79.8/65.0	88.1/82.8	89.7/86.2	**90.2/86.8**
		3%	72.4/57.8	87.0/81.9	86.1/81.5	**89.1/85.5**

Distillation-augmented performances for selected high sparsity levels.

Wanda

- Motivation:
  - $y = w_1 x_1 + w_2 x_2$
  - $|w_1| \leq |w_2|$, but $|x_1| \gg |x_2|$
- Pruning Metric:
  - $S_{ij} = |W_{ij}| \cdot \|X_j\|_2$



Sun et al. "*A Simple and Effective Pruning Approach for Large Language Models*." International Conference on Learning Representations. (ICLR), 2024.

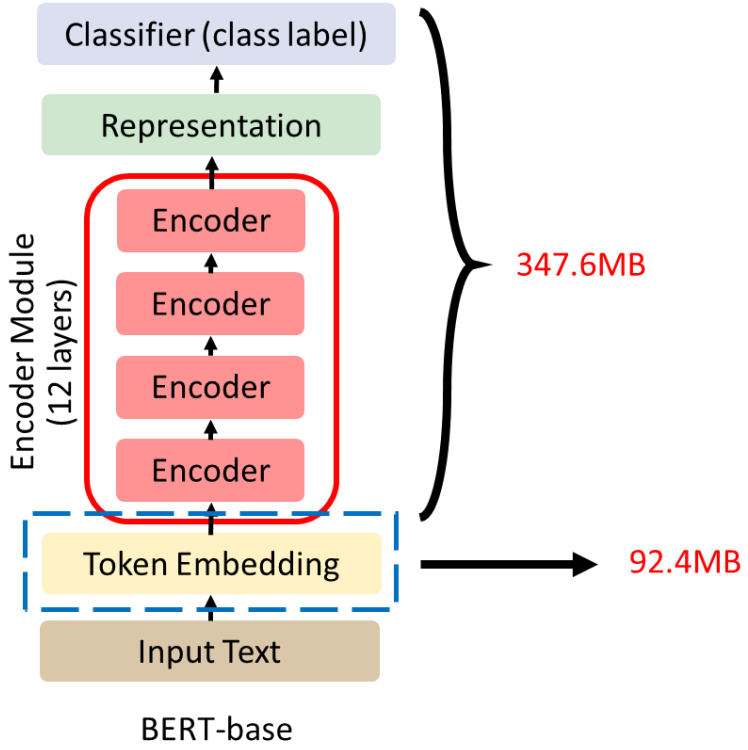
Wanda

Method	Weight Update	Sparsity	LLaMA 7B	LLaMA 13B	LLaMA 30B	LLaMA 65B	LLaMA-2 7B	LLaMA-2 13B	LLaMA-2 70B
Dense	-	0%	59.99	62.59	65.38	66.97	59.71	63.03	67.08
Magnitude	✗	50%	46.94	47.61	53.83	62.74	51.14	52.85	60.93
SparseGPT	✓	50%	**54.94**	58.61	63.09	66.30	**56.24**	60.72	**67.28**
Wanda	✗	50%	54.21	**59.33**	**63.60**	**66.67**	**56.24**	**60.83**	67.03
Magnitude	✗	4:8	46.03	50.53	53.53	62.17	50.64	52.81	60.28
SparseGPT	✓	4:8	**52.80**	55.99	60.79	64.87	**53.80**	**59.15**	65.84
Wanda	✗	4:8	52.76	**56.09**	**61.00**	**64.97**	52.49	58.75	**66.06**
Magnitude	✗	2:4	44.73	48.00	53.16	61.28	45.58	49.89	59.95
SparseGPT	✓	2:4	**50.60**	**53.22**	58.91	62.57	**50.94**	54.86	63.89
Wanda	✗	2:4	48.53	52.30	**59.21**	**62.84**	48.75	**55.03**	**64.14**

Mean zero-shot accuracies on 7 zero-shot tasks
of pruned LLaMA and LLaMA-2 models.

Token Embedding Compression

- Token embedding matrix often occupies a significant proportion of the whole model.
- Existing compression methods usually do not focus on the token embedding matrix specifically.
- Approximation error of token embedding will accumulate in later layers.

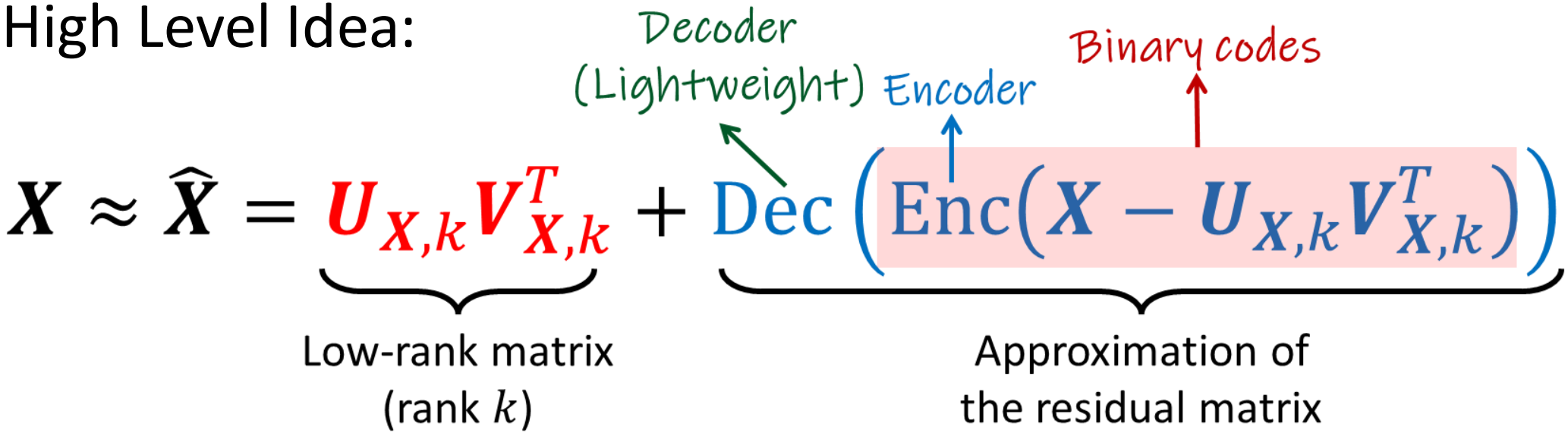


	Accuracy	Compression Ratio	Task-agnostic	Model-agnostic
SVD	Low	Low	√	√
Quantization [Zafrir O et al., 2019]	High	Low	√	√
DistilBERT [Sanh V et al., 2019]	High	×	√	√
MobileBERT (SVD+KD) [Sun Z et al., 2019]	High	Medium	√	×
BinaryBERT (Quantization+KD) [Bai H et al., 2019]	Medium	High	×	√

LightToken

- Problem Formulation
  - Token embedding matrix $\boldsymbol{X} \in \mathbb{R}^{d \times N}$.
  - Compressed token embedding matrix $\widehat{\boldsymbol{X}}$.

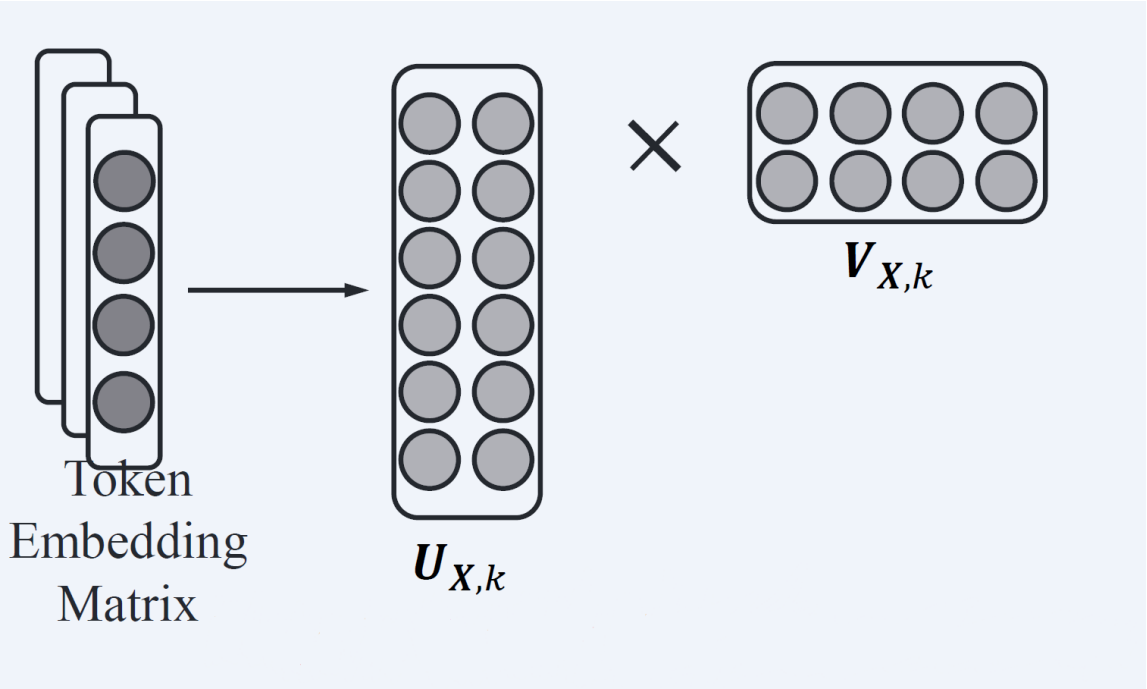
High Level Idea:



Wang et al. "*LightToken: A Task and Model-agnostic Lightweight Token Embedding Framework for Pre-trained Language Models*." KDD. 2023.

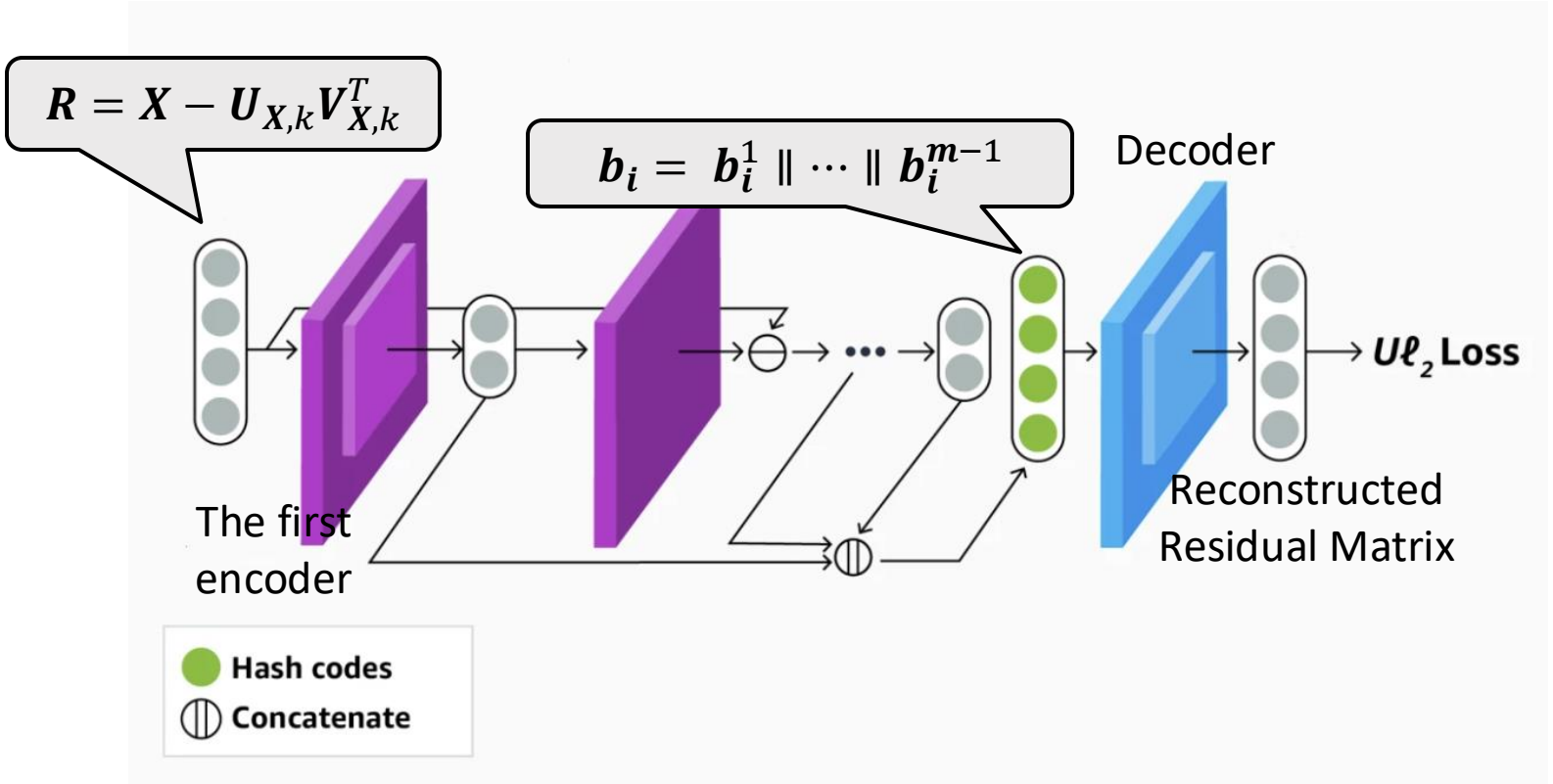
LightToken

- Stage 1: Rank-k SVD Approximation
  - ***Goal: Extract coherent components***
  - $\boldsymbol{X = USV^T \approx U_{X,k} V_{X,k}^T}$

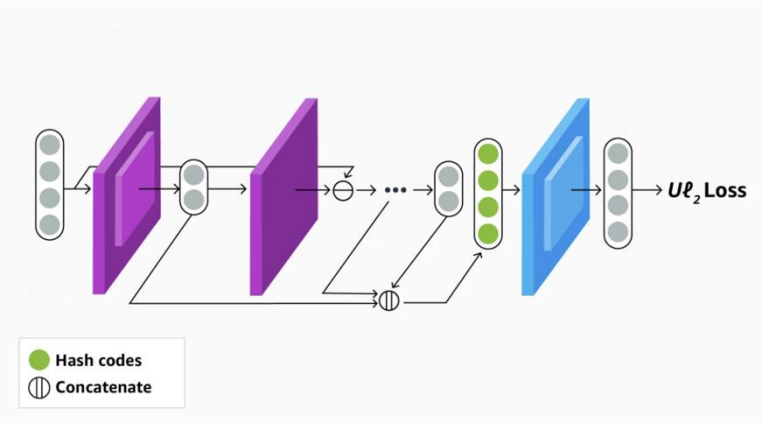


LightToken

- Stage 2: Residual Binary Autoencoder
  - ***Goal: Approximate the residual***



LightToken



- Stage 2: Residual Binary Autoencoder
  - Reconstructed Token Embedding

$$\widehat{\boldsymbol{X}} = \widehat{\boldsymbol{R}} + \boldsymbol{U}_{X,k}\boldsymbol{V}_{X,k}^{T}$$

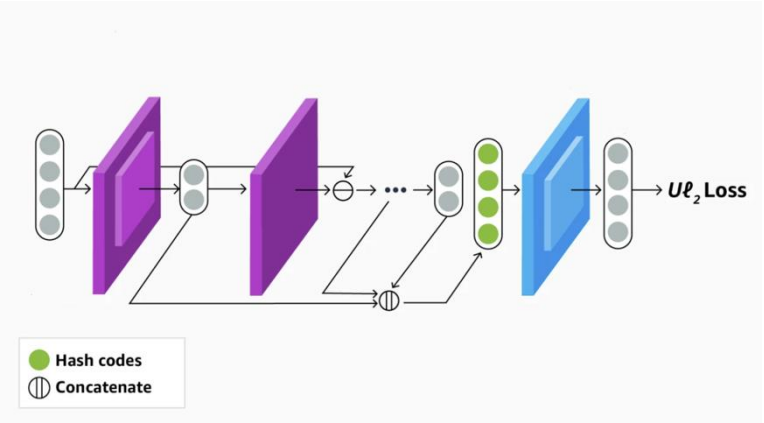
  - Euclidean distance

$$\|\boldsymbol{X}_i - \widehat{\boldsymbol{X}}_i\|_2^2 = \underbrace{\|\boldsymbol{X}_i - \widehat{\boldsymbol{X}}_i^{\parallel}\|_2^2}_{\text{Error in parallel direction}} + \underbrace{\|\boldsymbol{X}_i - \widehat{\boldsymbol{X}}_i^{\perp}\|_2^2}_{\text{Error in orthogonal direction}}$$

$$\|\widehat{\boldsymbol{X}}_i\|_2^2 \left(1 - \left(\frac{\langle \boldsymbol{X}_i, \widehat{\boldsymbol{X}}_i \rangle}{\|\boldsymbol{X}_i\| \|\widehat{\boldsymbol{X}}_i\|}\right)^2\right)$$

  - Limitation: Provide too much freedom to learn the autoencoder parameters and may confuse/mislead the approximation learning

LightToken



- Stage 2: Residual Binary Autoencoder
  - Upper Bound of Error in Orthogonal Direction

$$\|\widehat{\boldsymbol{X}}_i\|_2^2\left(1-\left(\frac{\langle \boldsymbol{X}_i,\widehat{\boldsymbol{X}}_i\rangle}{\|\boldsymbol{X}_i\|\|\widehat{\boldsymbol{X}}_i\|}\right)^2\right)\leq 2\|\widehat{\boldsymbol{X}}_i\|_2^2\left(1-\frac{\langle \boldsymbol{X}_i,\widehat{\boldsymbol{X}}_i\rangle}{\|\boldsymbol{X}_i\|\|\widehat{\boldsymbol{X}}_i\|}\right)$$

  - The $U\ell_2$ loss:

$$\sum_i \underbrace{\|\boldsymbol{X}_i-\widehat{\boldsymbol{X}}_i\|_2^2}_{\text{Euclidean distance}}+\underbrace{\|\widehat{\boldsymbol{X}}_i\|_2^2\left(1-\frac{\langle \boldsymbol{X}_i,\widehat{\boldsymbol{X}}_i\rangle}{\|\boldsymbol{X}_i\|\|\widehat{\boldsymbol{X}}_i\|}\right)^2}_{\text{Weighted cosine similarity loss}}$$

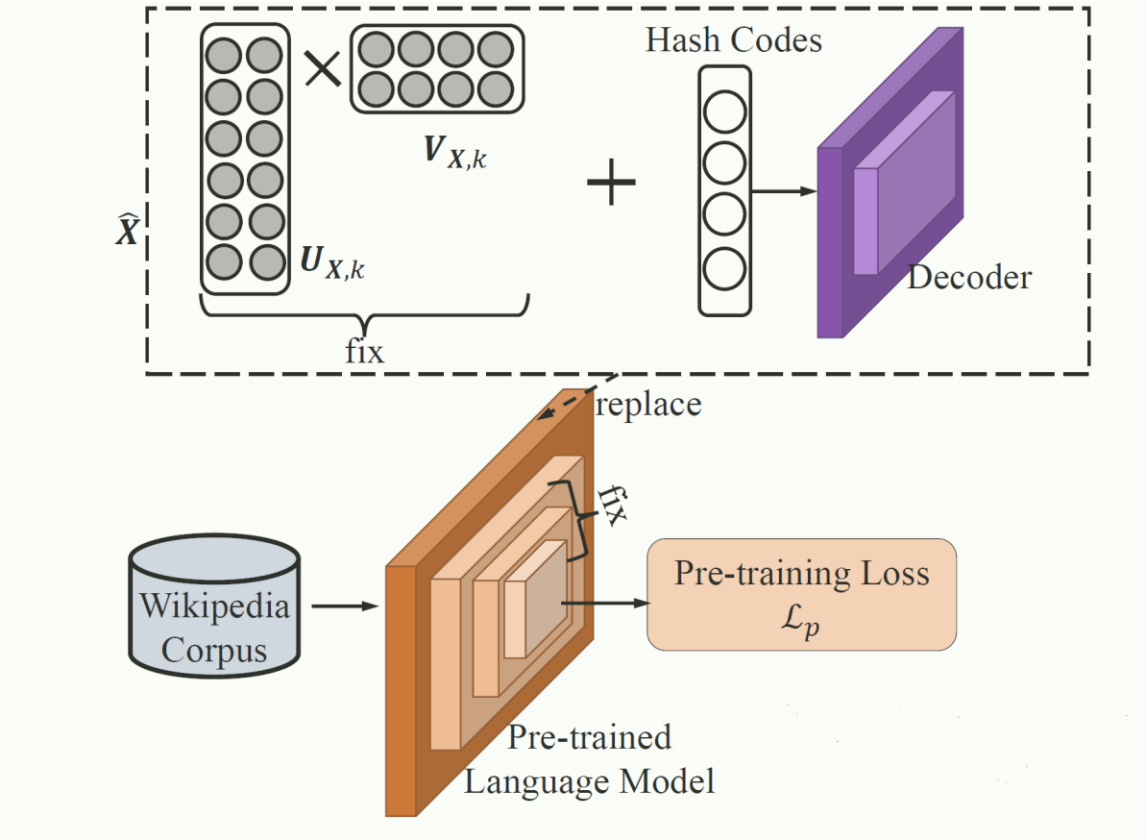
LightToken

- Stage 3: Fine-tuning Decoder with Pre-training Objective
  - Fine-tune the learned decoder on the subset of Wikipedia corpus
  - $\min_{\boldsymbol{\Theta}_D} \mathcal{L}_p\left(\mathfrak{X}; \boldsymbol{B}, \boldsymbol{\Theta}_D, \boldsymbol{\Theta}_{PLM}\right)$

Decoder parameters

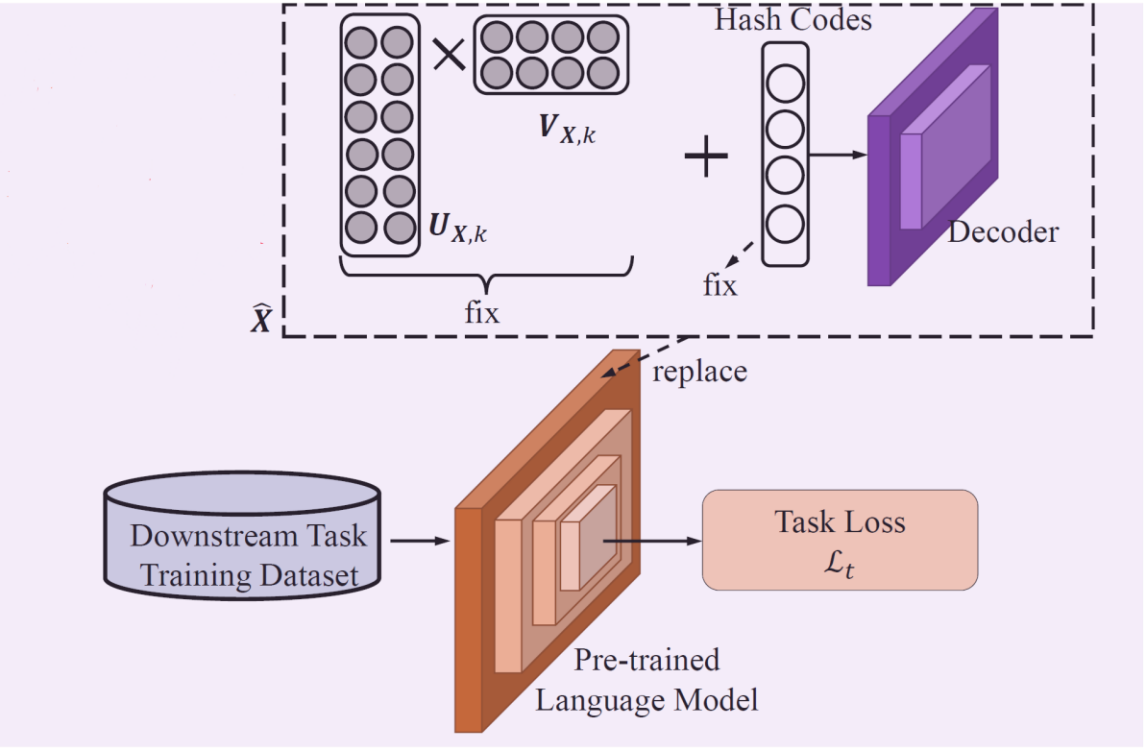
Subset of Wiki corpus

Parameters of the PLM except token embedding

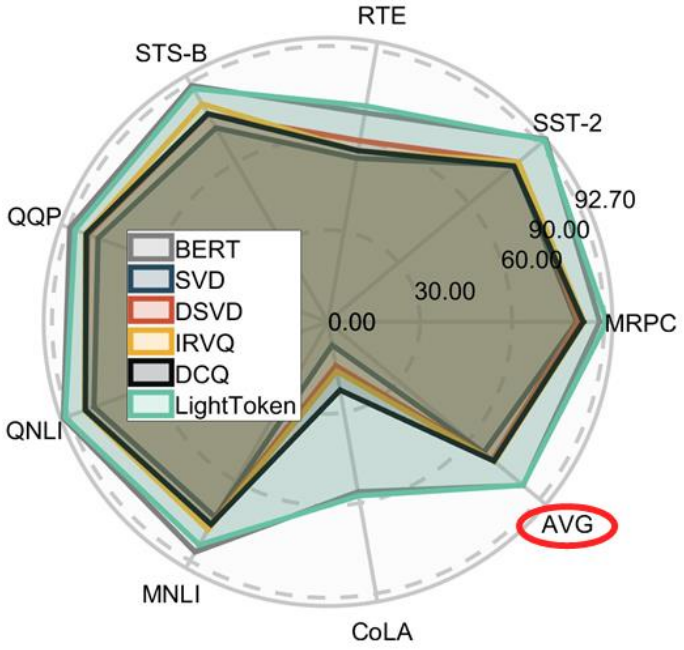


LightToken

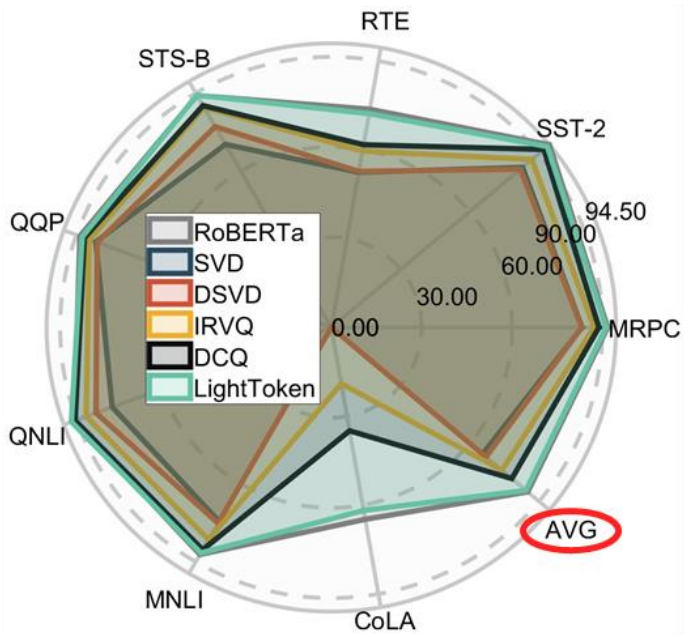
- Stage 4: Fine-tuning PLM for Downstream Tasks
  - Fine-tune the model with task-specific loss functions
  - $\min_{\Theta_{PLM}} \mathcal{L}_p\left(\mathfrak{X}; \boldsymbol{B}, \boldsymbol{\Theta}_D, \boldsymbol{\Theta}_{PLM}\right)$



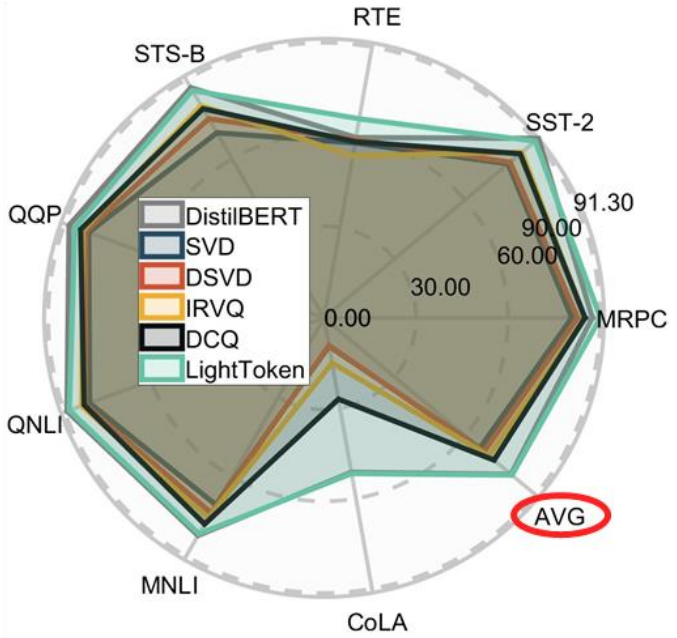
LightToken



BERT



RoBERTa



DistilBERT

Outline

- Model Efficiency
  - Parameter-efficient Fine-tuning
  - Model Compression
  - Inference Acceleration
- Data Efficiency
  - Zero/few-shot Learning
  - Knowledge Editing
  - Retrieval-augmented Generation

Inference Acceleration

- **What is Inference Acceleration?**
  - Optimizing the speed and efficiency of the inference process without significant accuracy drop.
- **Why is it Important?**
  - Growing demand for deploying complex models.
  - Limitations of hardware capabilities.
  - Need for real-time responses in various applications.
- **Conditional Computation offers a flexible alternative:**
  - Allows **dynamic resource allocation per input**, reducing unnecessary computation.
  - Maintains strong accuracy while improving efficiency.

Conditional Computation

Core Idea

- **Traditional inference uses all model parameters for every input.**
- **Conditional computation activates only necessary parts of the model per input.**
  - Saves computation by dynamically adjusting model execution.
  - Enables trade-offs between accuracy and efficiency.
  - Can be applied at different levels: **layer-wise, token-wise, or expert selection**.
- **Three Major Approaches:**
  - **Mixture of Experts (MoE)** – Activate only a subset of specialized experts for each input.
  - **Early Exit** – Allow models to terminate inference early if confidence is high.
  - **Cascade Models** – Process simpler inputs with smaller models.

Mixture of Experts (MoE) – Sparse Computation

Key Idea: Instead of using all parameters, activate only a subset of **specialized "expert" networks** per input.

How it Works:
- A **router network** selects a few experts from a pool.
- Each input is processed by a small fraction of the total model.
- Allows scaling **without linearly increasing compute cost**.

Advantages:
- Enables extremely large models **without proportional inference cost**.
- Improves efficiency while maintaining accuracy.

Challenges:
- **Routing overhead**: Selecting the right experts introduces latency.
- **Load balancing**: Some experts may become over-utilized, leading to inefficiency.

Example Models:
- Sparsely-Gated Mixture-of-Experts layer, Switch Transformer

Sparsely-Gated Mixture-of-Experts layer (MoE)

- **Sparsely-Gated MoE Layer**: Uses a gating network to select a sparse subset of experts.
- **Top-k Routing**: Only the **k most relevant experts** are activated per input.
- **Conditional Computation**: Each input token is routed to a small fraction of the model, reducing computational cost.
- **Gradient Updates**: Experts receive gradients only when they are selected, leading to uneven training.
- **Load Balancing Loss**: Encourages more uniform expert utilization to avoid a few experts being overused.

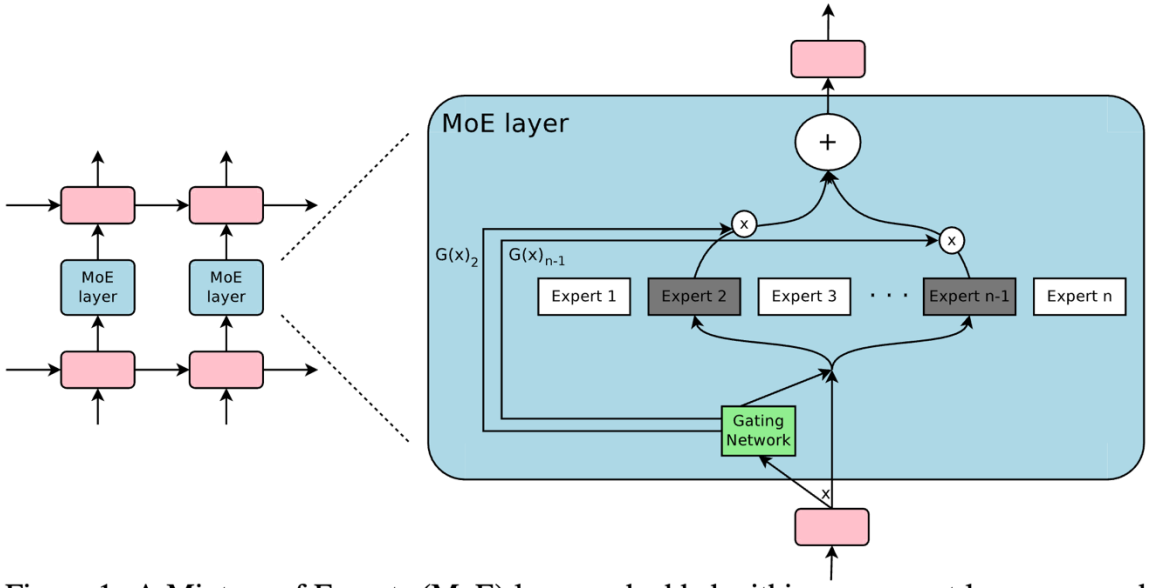


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

Credit: *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*.

Sparsely-Gated Mixture-of-Experts layer (MoE)

Contributions/Impacts

- **First scalable MoE model** in deep learning, allowing large models with minimal increase in computation.
- **Efficient Training**: Sparse activation significantly reduces FLOPs compared to dense models.
- **Introduced Expert Balancing**: Ensures all experts contribute meaningfully to learning.
- **Proved Feasibility of Sparse MoE**: Demonstrated that routing small subsets of the network maintains accuracy while reducing cost.

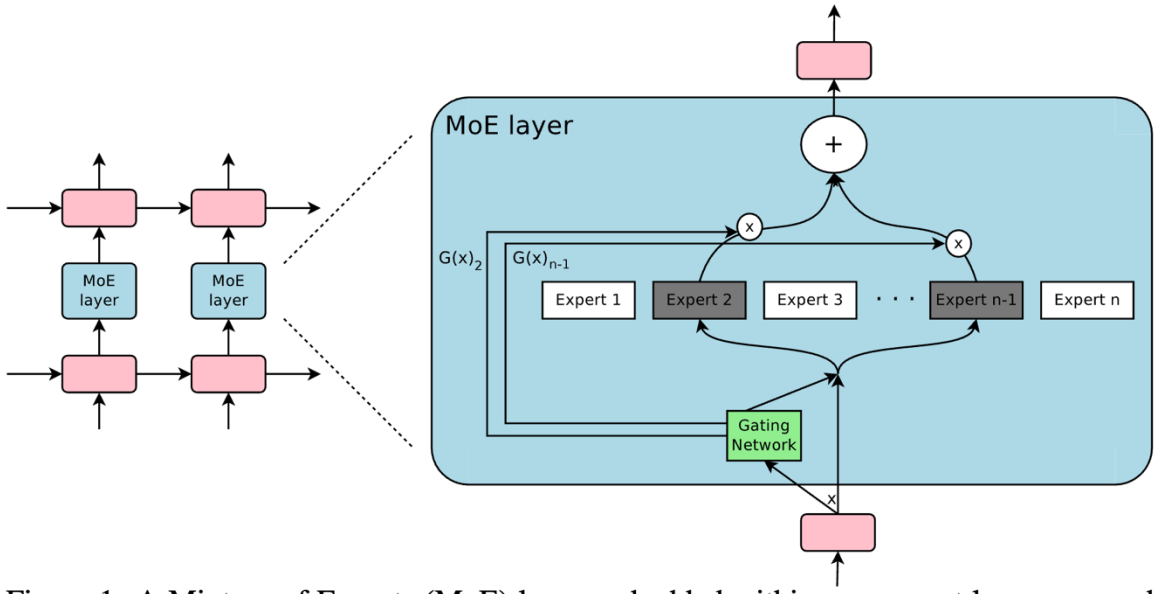


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

Credit: *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*.

Sparsely-Gated Mixture-of-Experts layer (MoE)

- **Language Modeling Performance**: MoE outperforms dense baselines while using only 10% of the computation complexity.
- **Scaling Benefits**: Larger models continue to improve without additional FLOPs, proving MoE's effectiveness at scaling.

Table 1: Summary of high-capacity MoE-augmented models with varying computational budgets, vs. best previously published results (Jozefowicz et al., 2016). Details in Appendix C.

	Test Perplexity 10 epochs	Test Perplexity 100 epochs	#Parameters excluding embedding and softmax layers	ops/timestep	Training Time 10 epochs	TFLOPS /GPU
Best Published Results	34.7	30.6	151 million	151 million	59 hours, 32 k40s	1.09
Low-Budget MoE Model	34.1		4303 million	8.9 million	15 hours, 16 k40s	0.74
Medium-Budget MoE Model	31.3		4313 million	33.8 million	17 hours, 32 k40s	1.22
High-Budget MoE Model	**28.0**		4371 million	142.7 million	47 hours, 32 k40s	**1.56**

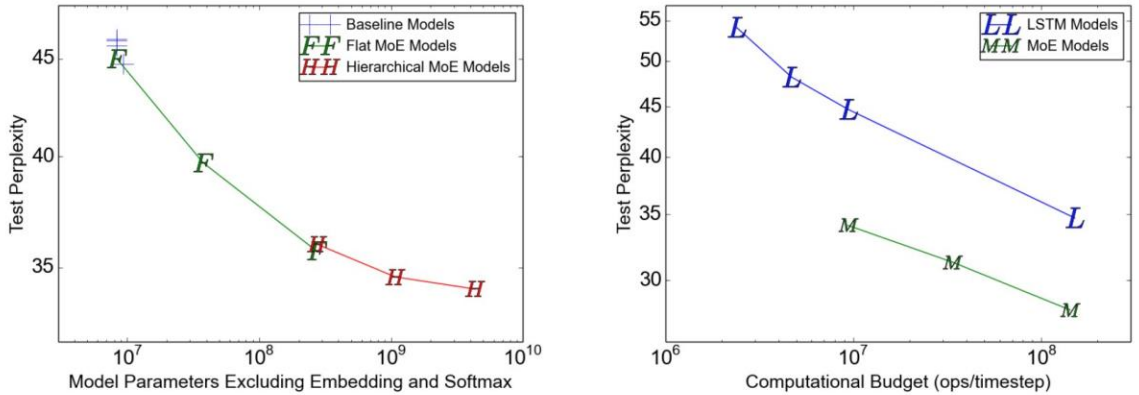
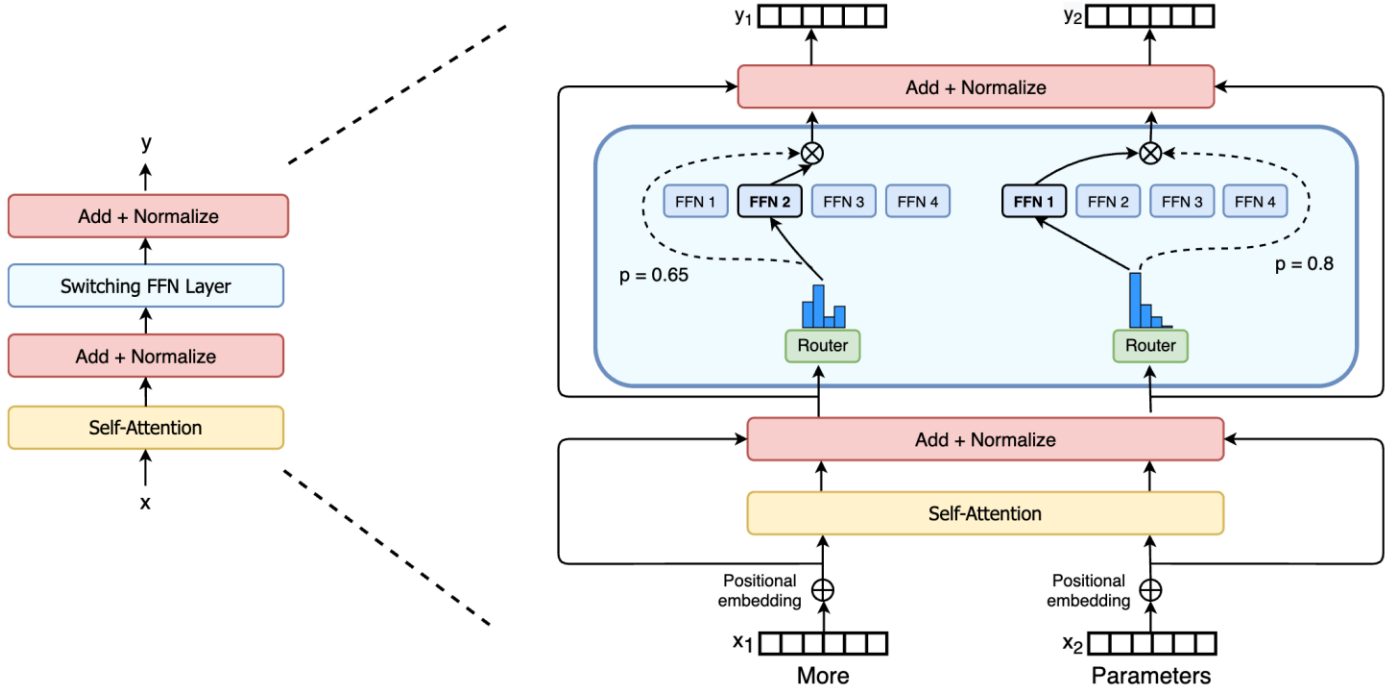


Figure 2: Model comparison on 1-Billion-Word Language-Modeling Benchmark. On the left, we plot test perplexity as a function of model capacity for models with similar computational budgets of approximately 8-million-ops-per-timestep. On the right, we plot test perplexity as a function of computational budget. The top line represents the LSTM models from (Jozefowicz et al., 2016). The bottom line represents 4-billion parameter MoE models with different computational budgets.

Credit: *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*.

Switch Transformer

- **Single Expert Routing**: Unlike previous MoEs, Switch Transformer activates **only one expert per token**, reducing communication overhead.
- **Improved Load Balancing Loss**: Ensures all experts receive roughly equal workloads.
- **Pretrained on Large Datasets**: Used **Colossal Clean Crawled Corpus (C4)** and other large datasets.
- **Gradient-Based Expert Selection**: Dynamically selects experts based on learned routing signals.



Credit: *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*.

Switch Transformer

- **Simplified MoE with lower communication overhead,** making large-scale deployment practical.
- **Showed trillion-parameter models are feasible,** proving that larger models continue to improve performance.
- **Improved Routing Mechanisms:** Reduced the computational waste of prior MoE implementations.
- **Outperformed Dense Transformers:** Achieved better accuracy while using only **30% of the computation** compared to T5.

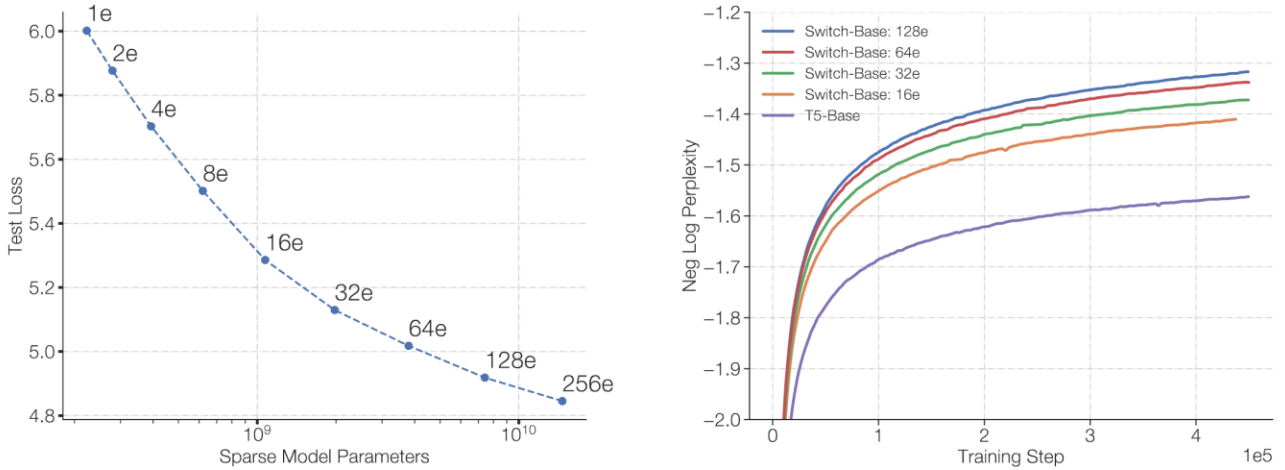


Figure 1: Scaling and sample efficiency of Switch Transformers. Left Plot: Scaling properties for increasingly sparse (more experts) Switch Transformers. Right Plot: Negative log perplexity comparing Switch Transformers to T5 (Raffel et al., 2019) models using the same compute budget.

Credit: *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*.

Switch Transformer

- **Scaling Experiments**: Showed **linear scaling laws** in NLP tasks with increasing model size.
- **Efficiency vs. Accuracy**: Demonstrated that Switch Transformer uses less training time to achieve similar or even better performance.

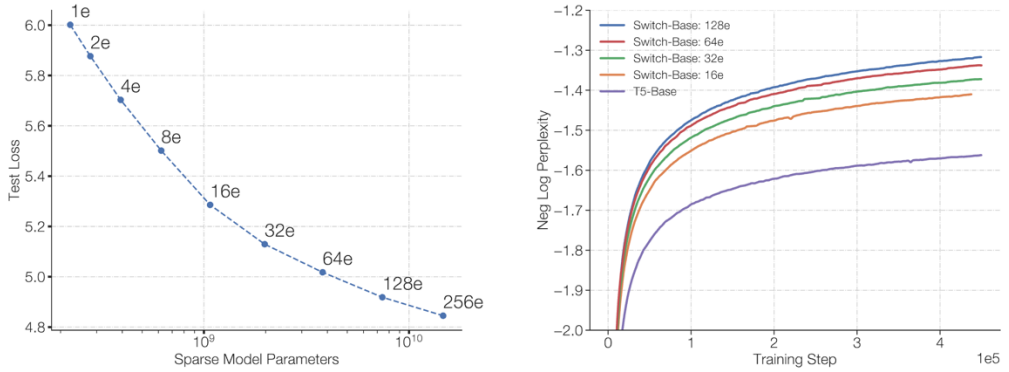


Figure 1: Scaling and sample efficiency of Switch Transformers. Left Plot: Scaling properties for increasingly sparse (more experts) Switch Transformers. Right Plot: Negative log perplexity comparing Switch Transformers to T5 (Raffel et al., 2019) models using the same compute budget.

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860
MoE-Base	1.25	-1.559	80.7	790
Switch-Base	1.25	-1.553	65.0	910
MoE-Base	1.0	-1.572	80.1	860
Switch-Base	1.0	-1.561	**62.8**	1000
Switch-Base+	1.0	**-1.534**	67.6	780

Credit: *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*.

Early Exit – Adaptive Depth Computation

Key Idea: Allow models to **terminate inference early** if they achieve high confidence before reaching the final layers.

How it Works:

- Each layer has an **exit condition** that determines whether computation should continue.
- Simple inputs exit early, while harder cases pass through more layers.

Advantages:

- Reduces computation on straightforward queries.
- Can improve latency without modifying model architecture.

Challenges:

- **Reliable confidence estimation**: Hard to determine when to exit without degrading performance.
- **Layer-wise training complexity**: Requires well-calibrated thresholds.

Example Applications:

- **BERT with early exit** for classification tasks.
- **Token-level early exit** in autoregressive decoding.

DeeBERT

- This approach integrates **internal classifiers** at each layer of the BERT model, enabling inputs to exit the model early if a certain confidence threshold is met.

- Experiments demonstrate that DeeBERT can reduce inference time by approximately 40% with minimal impact on model quality.

	SST-2		MRPC		QNLI		RTE		QQP		MNLI-(m/mm)	
	Acc	Time	F_1	Time	Acc	Time	Acc	Time	F_1	Time	Acc	Time
BERT-base												
Baseline	93.6	36.72s	88.2	34.77s	91.0	111.44s	69.9	61.26s	71.4	145min	83.9/83.0	202.84s
DistilBERT	−1.4	−40%	−1.1	−40%	−2.6	−40%	−9.4	−40%	−1.1	−40%	−4.5	−40%
DeeBERT	−0.2	−21%	−0.3	−14%	−0.1	−15%	−0.4	−9%	−0.0	−24%	−0.0/−0.1	−14%
	−0.6	−40%	−1.3	−31%	−0.7	−29%	−0.6	−11%	−0.1	−39%	−0.8/−0.7	−25%
	−2.1	−47%	−3.0	−44%	−3.1	−44%	−3.2	−33%	−2.0	−49%	−3.9/−3.8	−37%
RoBERTa-base												
Baseline	94.3	36.73s	90.4	35.24s	92.4	112.96s	67.5	60.14s	71.8	152min	87.0/86.3	198.52s
LayerDrop	−1.8	−50%	-	-	-	-	-	-	-	-	−4.1	−50%
DeeBERT	+0.1	−26%	+0.1	−25%	−0.1	−25%	−0.6	−32%	+0.1	−32%	−0.0/−0.0	−19%
	−0.0	−33%	+0.2	−28%	−0.5	−30%	−0.4	−33%	−0.0	−39%	−0.1/−0.3	−23%
	−1.8	−44%	−1.1	−38%	−2.5	−39%	−1.1	−35%	−0.6	−44%	−3.9/−4.1	−29%

Table 1: Comparison between baseline (original BERT/RoBERTa), DeeBERT, and other acceleration methods. LayerDrop only reports results on SST-2 and MNLI. Time savings of DistilBERT and LayerDrop are estimated by reported model size reduction.

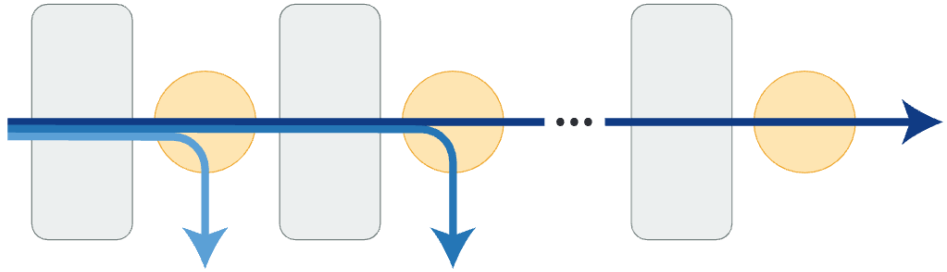
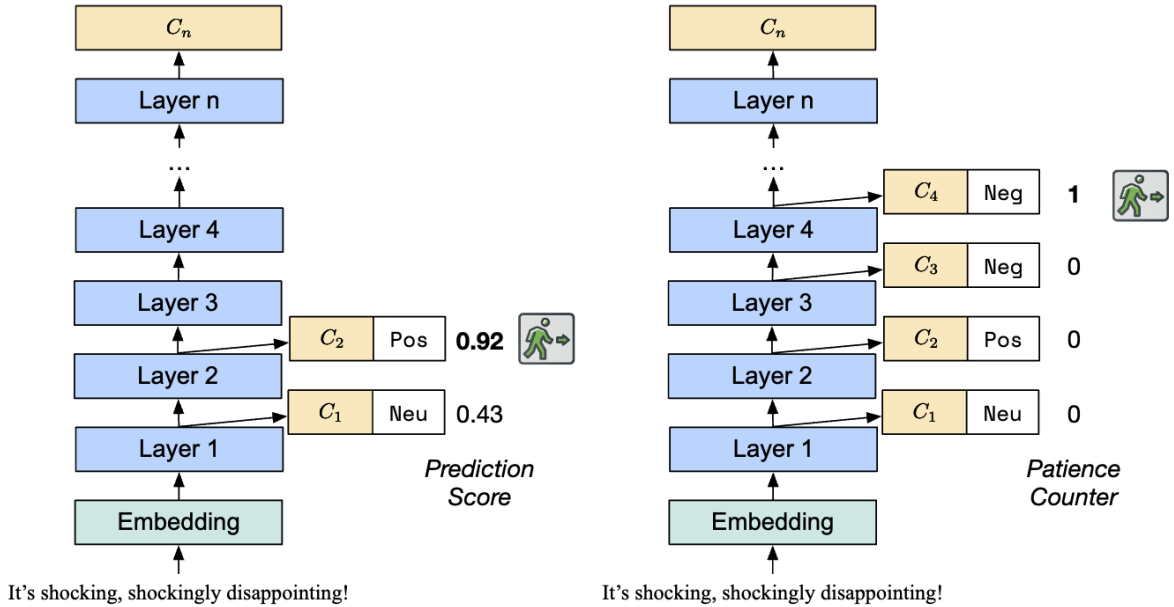


Figure 1: DeeBERT model overview. Grey blocks are transformer layers, orange circles are classification layers (off-ramps), and blue arrows represent inference samples exiting at different layers.

Xin et al. "*DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference*", ACL 2020.

PABEE

PABEE introduces a mechanism where an input exits the model once consecutive internal classifiers produce consistent predictions.



(a) Shallow-Deep Net [5] (b) Patience-based Early Exit (PABEE)

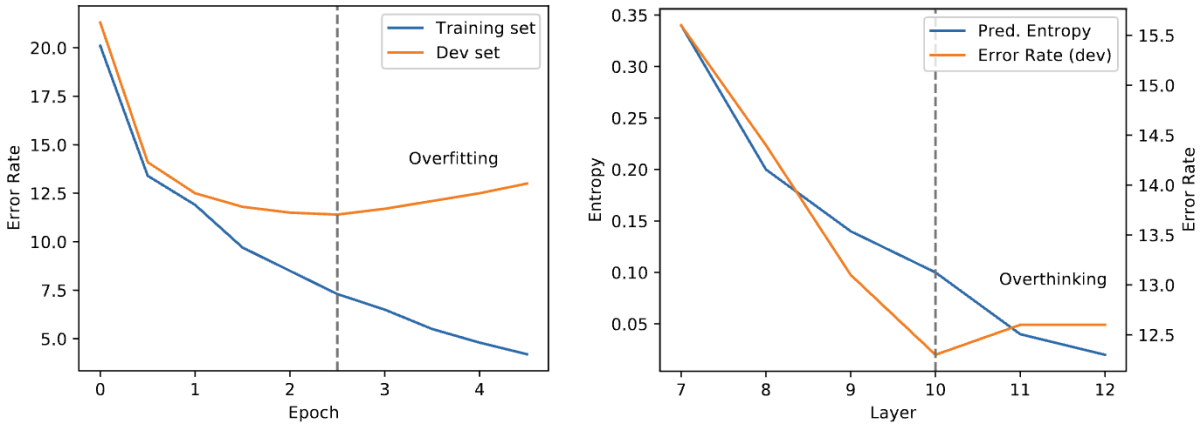
Figure 1: Comparison between Shallow-Deep Net, a prediction score based early exit (threshold is set to 0.9), and our Patience-based Early Exit (patience $t = 1$). A classifier is denoted by C_i, and n is the number of layers in a model. In this figure, Shallow-Deep incorrectly exits based on the prediction score while PABEE considers multiple classifiers and exits with a correct prediction.

Zhou et al. "*BERT Loses Patience: Fast and Robust Inference with Early Exit*," NeurIPS 2020.

PABEE

Insights:

- Deep models sometimes "overthink" by using later layers that may introduce noise or over-complicate the decision, leading to misclassifications. By stopping earlier, PABEE prevents this degradation in performance and can even improve accuracy.



(a) Overfitting in training

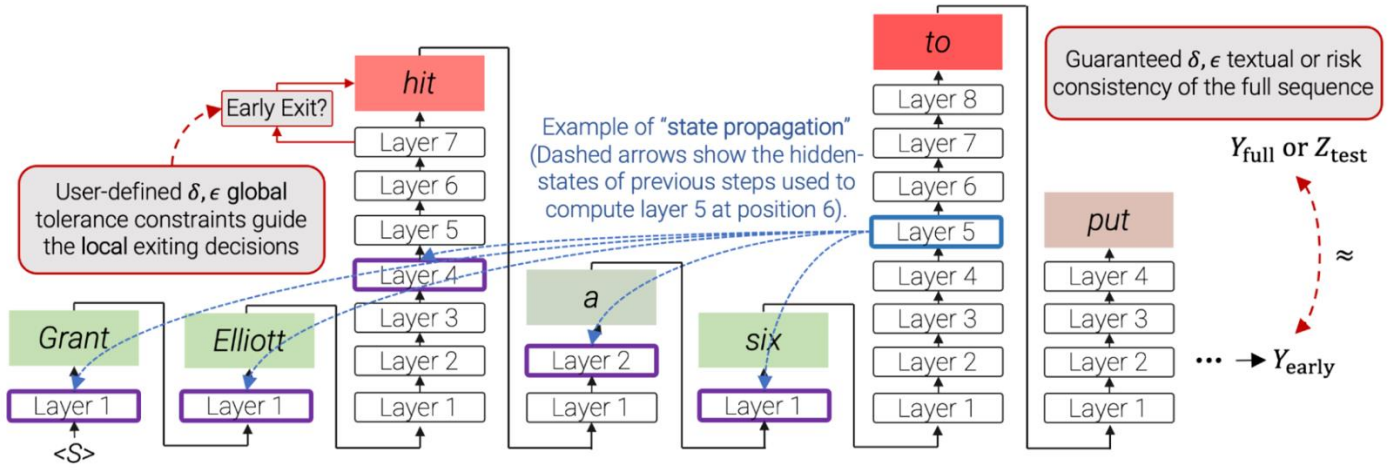
(b) Overthinking in inference

Figure 2: Analogy between overfitting in training and overthinking in inference. **(a)** In training, the error rate keeps going down on the training set but goes up later on the development set. **(b)** We insert a classifier after every layer. Similarly, the predicted entropy keeps dropping when more layers are added to inference but the error rate goes up after 10 layers. The results are obtained with ALBERT-base on MRPC.

Zhou et al. "*BERT Loses Patience: Fast and Robust Inference with Early Exit*", NeurIPS 2020.

CALM

- CALM proposes a token-level early exit strategy for **autoregressive** language models.
- By estimating the confidence of predictions at each step, the model decides whether to proceed with further computations or output the current token



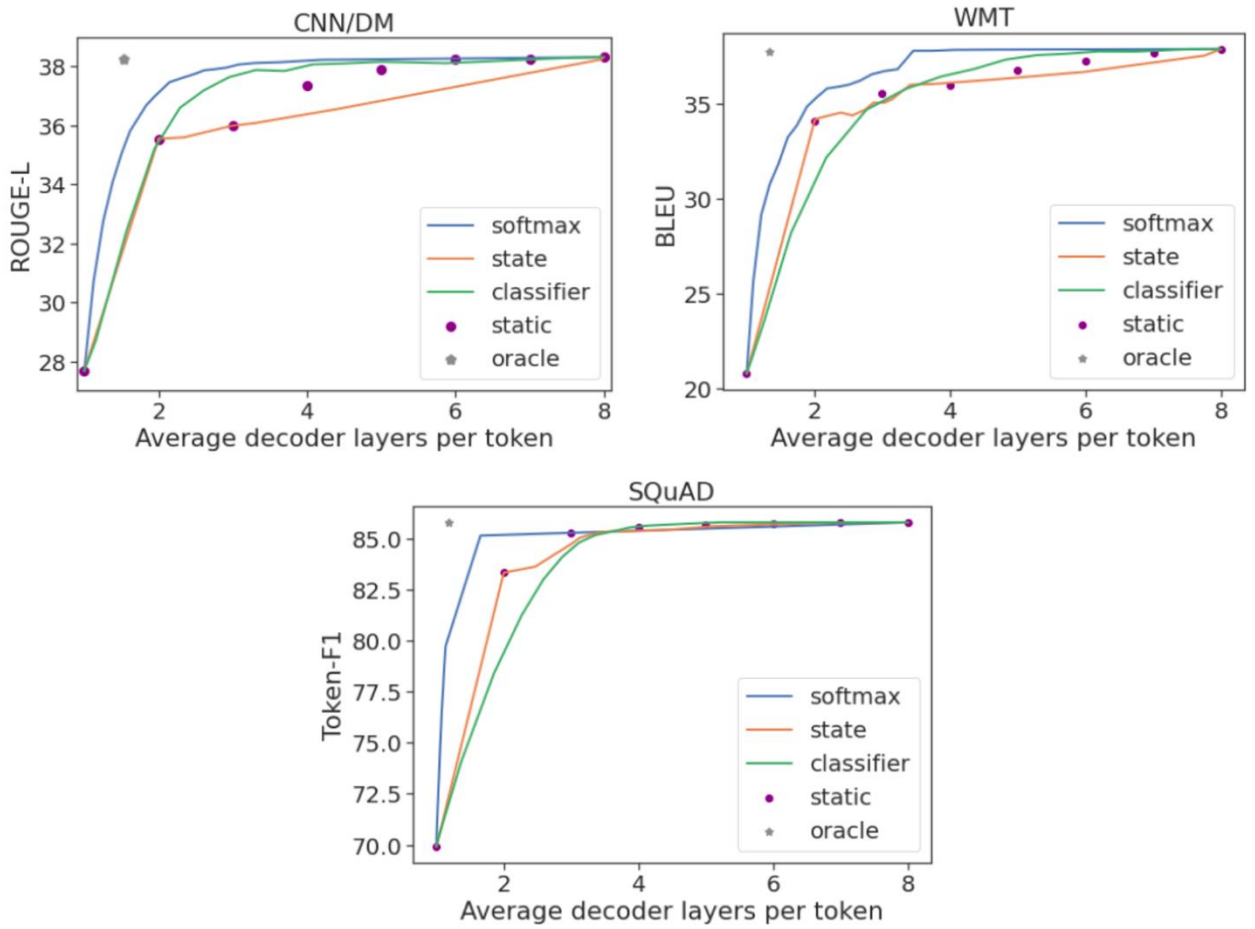
In CALM, local per-timestep confidence thresholds for early exiting decisions are derived, via LTT calibration, from user-defined consistency constraints over the full output text. Red boxes indicate that CALM used most of the decoder's layers for that specific prediction. Green boxes indicate that CALM saved time by using only a few Transformer layers. Full sentence shown in the last example of this post.

Schuster et al., "*Confident Adaptive Language Modeling*". NeurIPS 2022

CALM

Insights:

- CALM distinguishes itself by introducing dynamic, confidence-calibrated early exiting specifically tailored for models with decoder.
- Unlike generic early exit methods that rely on fixed thresholds, CALM adjusts exit criteria on-the-fly during sequential generation, accounting for uncertainty and fluency as the sequence unfolds.
- This enables more fine-grained control over when to stop decoding, optimizing both inference speed and output quality without requiring significant architectural changes.



HadSkip

HadSkip introduces an adaptive layer-skipping method to improve inference efficiency in pre-trained language models. Instead of sequentially processing all layers, it uses learnable gating mechanisms to skip layers dynamically based on a predefined budget.

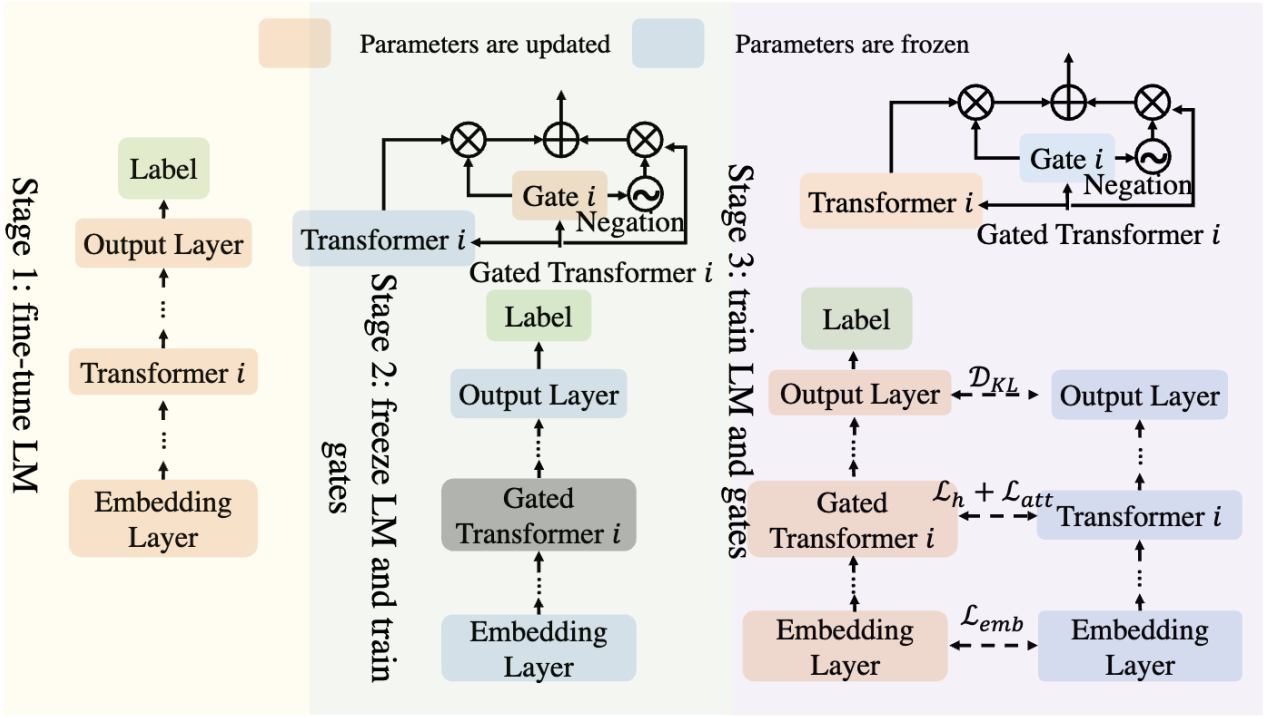


Figure 1: The framework of HadSkip.

Wang et al. "*HadSkip: Homotopic and Adaptive Layer Skipping of Pre-trained Language Models for Efficient Inference*", Findings of EMNLP 2023.

HadSkip

Insights:

- HadSkip's innovation lies in its learned, "hard-skip" mechanism that conditionally bypasses entire blocks of layers rather than terminating inference early.
- Unlike simple threshold methods, HadSkip uses a lightweight decision module trained jointly with the model to assess whether the current activations are sufficiently informative.

	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	AVG
BERT*	59.4	84.3	91.3	88.5	90.4	69.3	92.5	82.2
Budgeted-Exit*	0.0	70.0	75.8	77.4	81.8	54.7	81.0	63.0
BranchyNet*	0.0	63.8	75.7	74.2	71.6	54.7	79.9	60.0
Shallow-Deep*	0.0	64.1	75.6	74.3	71.4	54.7	79.5	59.9
BERxiT*	0.0	63.5	75.6	73.3	68.2	55.3	79.5	59.3
PABEE*	0.0	63.9	75.8	73.6	68.6	55.8	79.9	59.7
PCEE-BERT*	9.8	73.4	78.8	80.4	79.6	58.4	83.6	66.3
HadSkip-BERT	35.6	78.1	84.2	84.1	88.5	54.9	88.5	73.4
Budgeted-Exit*	0.0	79.6	84.7	85.3	89.3	68.1	88.6	70.8
BranchyNet*	0.0	78.3	83.0	87.1	89.3	67.4	88.3	70.5
Shallow-Deep*	0.0	78.2	82.8	87.2	89.6	67.2	88.4	70.5
BERxiT*	12.3	78.4	82.9	87.0	89.1	67.3	88.3	72.2
PABEE*	0.0	78.9	83.1	87.2	89.6	67.7	88.7	70.7
PCEE-BERT*	23.2	80.1	84.8	87.1	90.8	69.4	90.4	75.1
HadSkip-BERT	50.8	82.4	85.9	88.7	90.5	62.5	90.0	78.7
Budgeted-Exit*	51.9	83.0	87.0	88.4	90.3	69.0	91.2	80.1
BranchyNet*	52.1	83.0	85.8	89.3	90.1	68.0	91.2	79.9
Shallow-Deep*	52.3	82.9	85.7	89.3	90.1	67.8	91.2	79.9
BERxiT*	52.2	83.2	86.2	89.6	90.1	68.1	91.4	80.1
PABEE*	52.4	83.4	86.1	89.8	90.4	68.3	91.7	80.3
PCEE-BERT*	52.8	83.4	86.8	90.5	91.2	69.7	91.8	80.9
HadSkip-BERT	58.3	83.8	88.4	90.7	90.8	66.8	92.7	81.6

Table 1: Performance comparison on GLUE dataset. Blue, green, and orange shadow parts represent methods with 3, 6, and 9-layer budgets respectively. "AVG" notes the average performance of the 7 tasks. Results with "*" are taken from (Zhang et al., 2022).

HadSkip

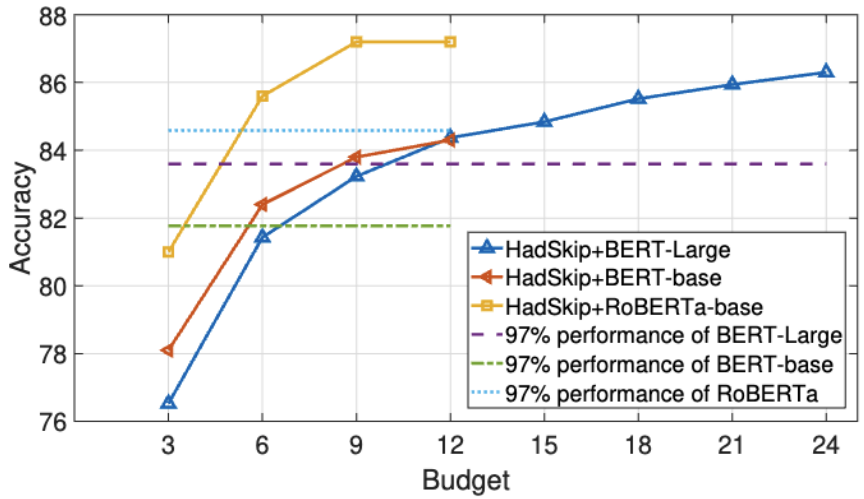


Figure 6: Accuracy with different budgets.

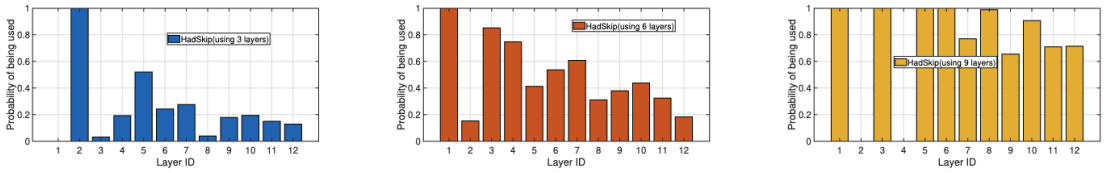


Figure 2: Visualization of the probability of each layer being used on MNLI dataset.

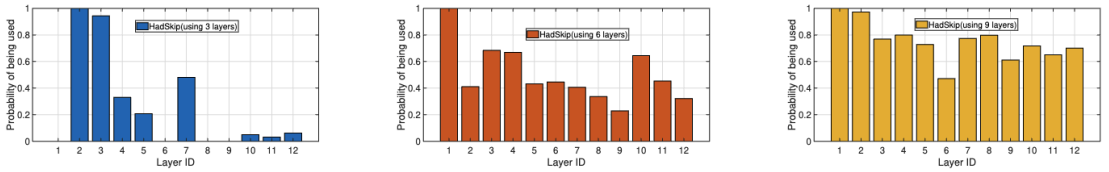


Figure 3: Visualization of the probability of each layer being used on QNLI dataset.

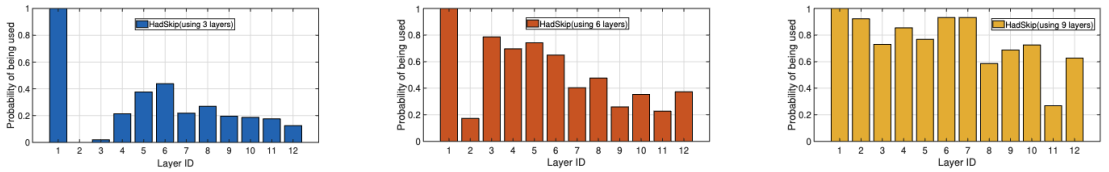


Figure 4: Visualization of the probability of each layer being used on SST2 dataset.

Observation show that with a lower budget, the model selectively utilizes specific layers more frequently, while at higher budgets, layer usage becomes more evenly distributed.

Cascade Models – Adaptive Complexity

Key Idea: Use a sequence of models or modules that progressively increase in complexity.

- Simple cases **exit early** using a small model.
- More complex cases **pass through additional processing layers**.

Advantages:

- Saves computation on easy tasks.
- Ensures high accuracy for complex inputs.

Challenges:

- Designing optimal routing mechanisms.
- Avoiding cascading errors in progressive inference.

Speculative Decoding

What is Speculative Decoding?

- A technique to accelerate LLM inference by **predicting multiple tokens at once**, reducing latency.
- Uses a **draft model** to generate multiple candidate tokens before verifying them with a **larger model**.

Speculative Decoding

What is Speculative Decoding?

- A technique to accelerate LLM inference by **predicting multiple tokens at once**, reducing latency.
- Uses a **draft model** to generate multiple candidate tokens before verifying them with a **larger model**.

Why is it Useful?

- Standard autoregressive decoding is **sequential and slow**.
- **Parallel token generation** improves efficiency without sacrificing output quality.

Speculative Decoding

What is Speculative Decoding?

- A technique to accelerate LLM inference by **predicting multiple tokens at once**, reducing latency.
- Uses a **draft model** to generate multiple candidate tokens before verifying them with a **larger model**.

Why is it Useful?

- Standard autoregressive decoding is **sequential and slow**.
- **Parallel token generation** improves efficiency without sacrificing output quality.

How It Works?

- **Draft Model** proposes a batch of candidate tokens.
- **Final Model** verifies and accepts or rejects them.
- **Accepted tokens** advance immediately, while rejected ones are recomputed.

Speculative Decoding

Key Contributions

- **Inference Speedup:** Achieves up to **2-3× faster inference** for large-scale transformers.
- **Maintains Output Quality:** Preserves accuracy while reducing latency.
- **Generalizable:** Works across different transformer architectures and model sizes.

Insights

- **Better Draft Models = Higher Efficiency**
  - If the draft model closely matches the final model, fewer tokens need to be recomputed.
- **Trade-off Between Speed & Quality**
  - A larger batch size increases speed but may introduce more rejected tokens.
- **Compatible with Hardware Acceleration**
  - Can be combined with quantization, TPUs, and GPU optimizations.

Table 2. Empirical results for speeding up inference from a T5-XXL 11B model.

TASK	M_q	TEMP	γ	α	SPEED
ENDE	T5-SMALL ★	0	7	0.75	**3.4X**
ENDE	T5-BASE	0	7	0.8	2.8X
ENDE	T5-LARGE	0	7	0.82	1.7X
ENDE	T5-SMALL ★	1	7	0.62	**2.6X**
ENDE	T5-BASE	1	5	0.68	2.4X
ENDE	T5-LARGE	1	3	0.71	1.4X
CNNDM	T5-SMALL ★	0	5	0.65	**3.1X**
CNNDM	T5-BASE	0	5	0.73	3.0X
CNNDM	T5-LARGE	0	3	0.74	2.2X
CNNDM	T5-SMALL ★	1	5	0.53	**2.3X**
CNNDM	T5-BASE	1	3	0.55	2.2X
CNNDM	T5-LARGE	1	3	0.56	1.7X

Leviathan et al., "*Fast Inference from Transformers via Speculative Decoding*". ICML 2023

Outline

- Model Efficiency
  - Parameter-efficient Fine-tuning
  - Model Compression
  - Inference Acceleration
- Data Efficiency
  - Zero/few-shot Learning
  - Knowledge Editing
  - Retrieval-augmented Generation

Background & Motivation

- **Context**
  - Large-scale Pre-trained Language Models (PLMs) have led to major breakthroughs in NLP.
  - Labeled data can be expensive or scarce in many domains. How do we **achieve strong performance with limited labeled data**?
- **Few-shot & Zero-shot Learning**
  - **Few-shot**: Use only a handful of labeled examples to fine-tune or adapt a model.
  - **Zero-shot**: The model is applied to a new task **without any** labeled examples.
- **Why Synthetic Data?**
  - Generating or augmenting training samples automatically.
  - Reduce annotation cost, improve model robustness, and expand coverage of data.

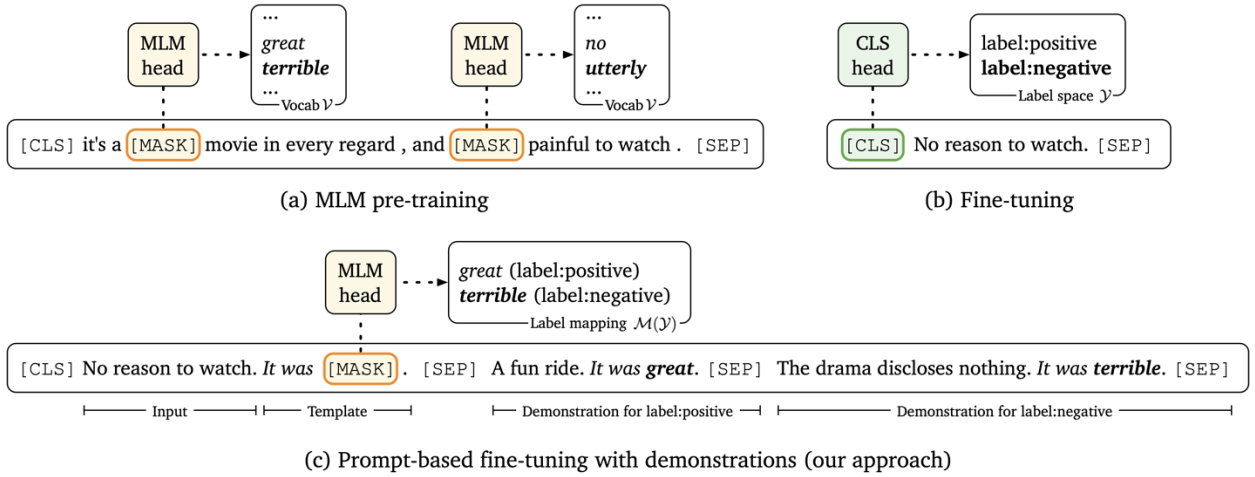
Encoder-Only Era: Fine-tuning

- **Full Fine-tuning** (typical in BERT-era)
  - Standard practice: take the entire BERT model and fine-tune on a downstream task.
  - **Challenge**: Requires a sufficient amount of labeled data to avoid overfitting.
- **Why Few-shot/Zero-shot was Hard**
  - Encoder-only models (e.g., BERT) lack autoregressive generation, making open-ended task adaptation difficult.
  - Their masked language modeling (MLM) objective limits dynamic response generation.
- **Initial Solutions**
  - Some attempts involved leveraging **semantic representations** (CLS embeddings, etc.) or minimal modifications to the model.
  - Often still needed a decent amount of labeled data to fine-tune effectively.

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019

LM-BFF: An Example of Few-shot Progress (Encoder-Only)

- **Key Ideas**:
  - Convert downstream tasks to a "masked language modeling" style via **prompt templates**.



- **Takeaway**: Showed that even encoder-based models like BERT can be adapted for few-shot learning with prompt techniques, bridging the gap to more generative-style methods.

Credit: *Making Pre-trained Language Models Better Few-shot Learners*.

Transition to Decoder Models: T5, GPT-3 & Emerging Zero-shot Capabilities

- **Filling the Gaps**
  - Decoder (generative) models, such as **GPT** or **T5**, are naturally suited for text generation.
  - This flexibility allows tasks to be framed as “text-to-text” problems.
- **Key Milestones**
  - **T5 [1]** (2019): Proposed a unified “Text-to-Text” framework for various NLP tasks.
  - **GPT-3 [2]** (2020): Demonstrated **In-Context Learning**, using only prompts and a few examples without weight updates.
- **Few-shot & Zero-shot**
  - T5 and GPT-3 showed impressive generalization, especially GPT-3’s ability to handle unseen tasks with just a prompt and few examples.

[1] Raffel et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. JMLR. 2019
[2] Brown et al. Language Models are Few-Shot Learners. NeurIPS. 2020

T5

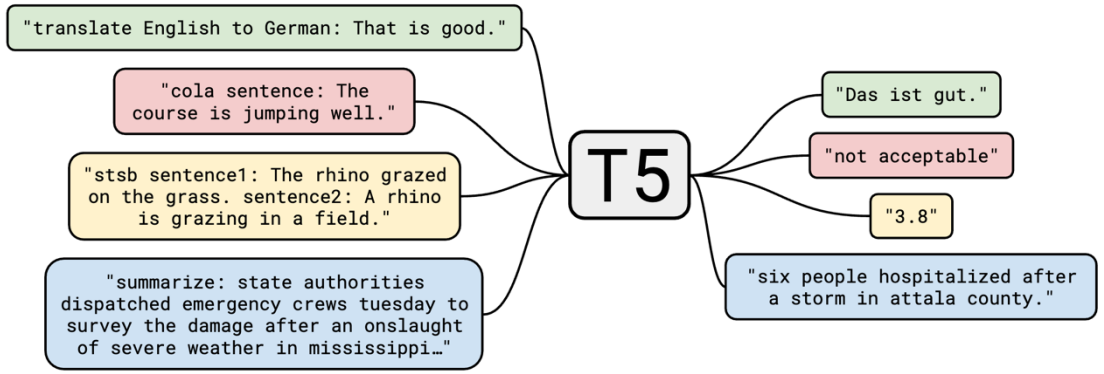
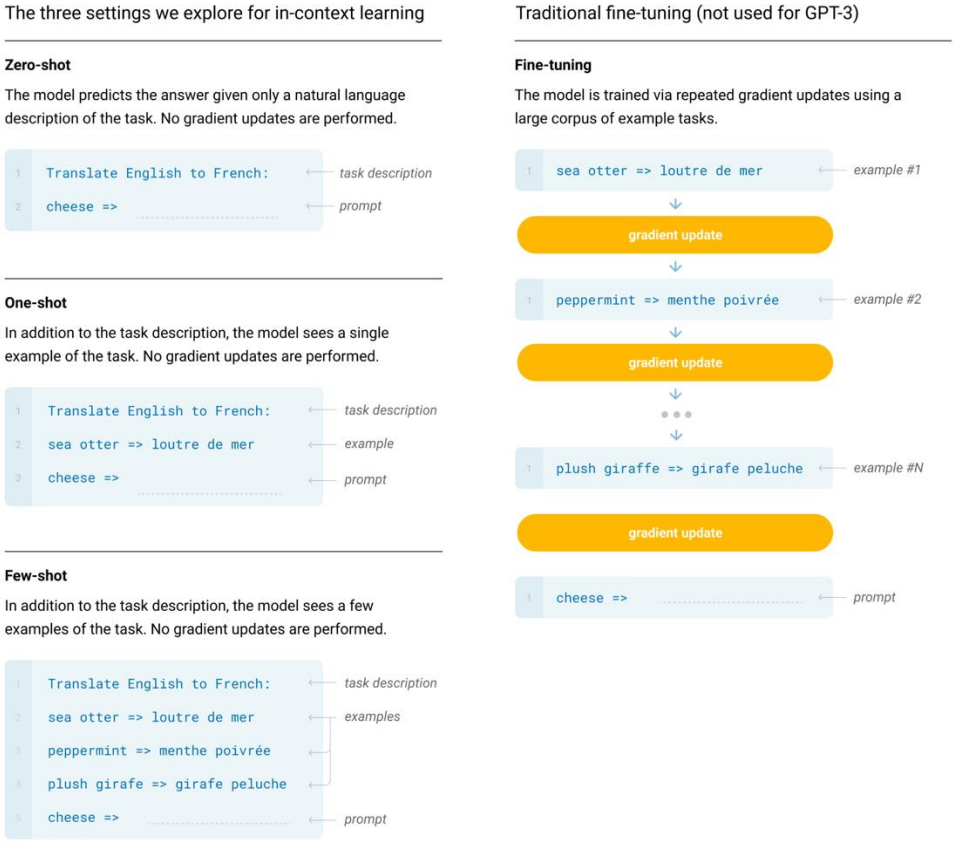


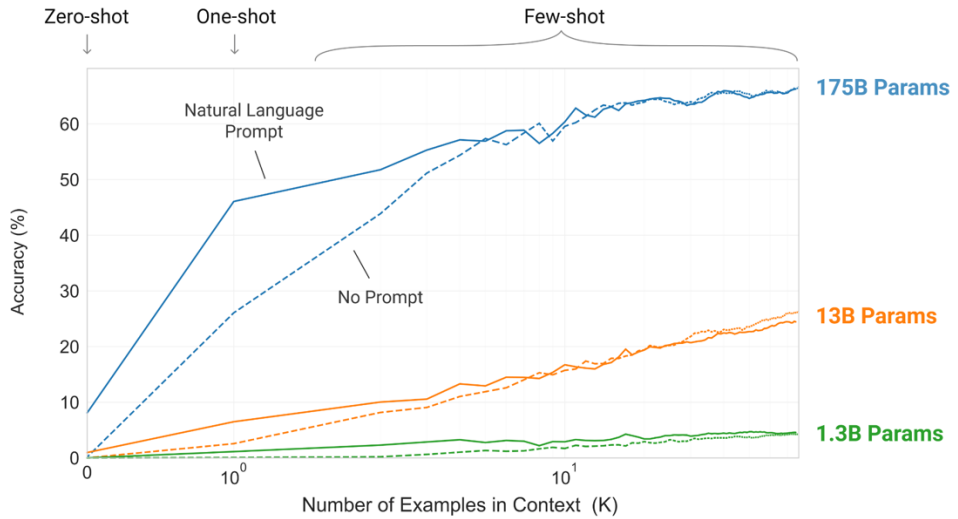
Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

- **Key Milestone:** Unified NLP tasks by framing them as text-to-text problems
- **Impact:** Set new performance standards on diverse NLP benchmarks by simplifying task formulation

Credit: *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.*

GPT3





- **Key Milestone:** Pushed the boundaries of scaling with 175 billion parameters

- **Impact:** Demonstrated strong zero-shot and few-shot learning abilities, reshaping expectations for language models

Credit: *Language Models are Few-Shot Learners*.

Instruction tuning

- **Definition**
  - Train or fine-tune language models to explicitly **follow natural language instructions** for a variety of tasks.
- **Motivation**
  - Enhances **zero-shot** and **few-shot** performance: The model learns how to interpret instructions without per-task fine-tuning.
- **Key Points**
  - Often uses many tasks, each with multiple prompt templates.
  - Can be combined with **synthetic** or **human-labeled** instruction data.
  - Leads to more **generic** and **adaptive** models.

T0 – Early Multitask Prompted Model

- **Method**
  - T0 is based on **T5**, but trained on many tasks with **multiple prompt templates** each.
- **Zero-Shot Generalization**
  - T0 can handle new tasks with just the prompt, no additional labels.
- **Significance**
  - Demonstrated the power of *prompt-based multi-task training* to achieve good zero-shot results.
  - Laid the groundwork for subsequent *instruction tuning* expansions.

Sanh et al., *Multitask Prompted Training Enables Zero-Shot Task Generalization*. ICLR 2022

T0

- **Key Milestone:** Effectiveness of fine-tuning on a large variety of prompts for zero-shot task generalization
- **Impact:** Advanced instruction-following capabilities, proving effective across unseen tasks without specific fine-tuning

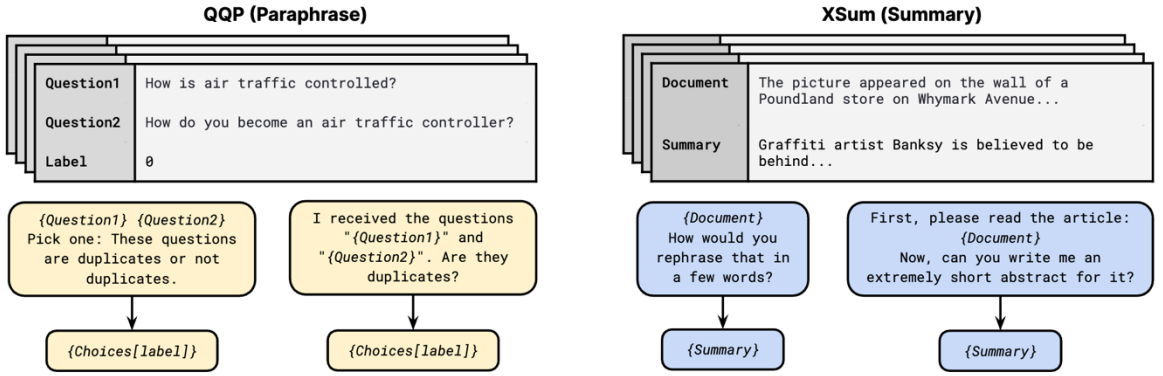


Figure 3: Prompt templates from the P3 prompt collection. Each dataset has multiple prompt templates consisting of an input and a target template. These use the fields of the raw data examples as well as template metadata, e.g., the left paraphrasing identification prompts use *Choices*, a template-level list variable `['Not duplicates', 'Duplicates']`. These templates are materialized to produce the prompted instance shown in Figure 1. The complete set of prompt templates used in T0 is given in Appendix G.

Credit: *Multitask Prompted Training Enables Zero-Shot Task Generalization*.

FLAN – Expanding Instruction Tuning at Scale

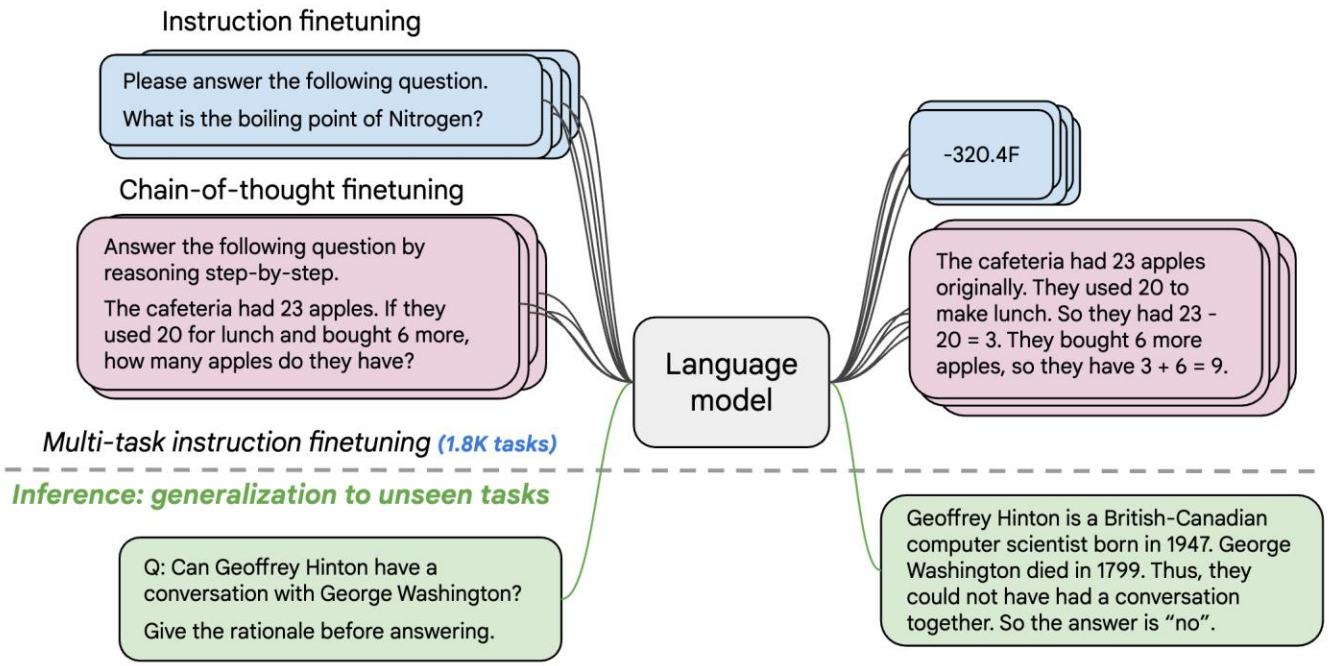
- **Core Idea**
  - Further extends training with a **much wider range** of tasks & advanced prompting strategies (e.g., chain-of-thought in newer versions).
- **Contributions**
  - Larger & more diverse task coverage.
  - Refining prompt engineering to improve zero/few-shot performance.
- **Outcome**
  - FLAN models show strong cross-task generalization, confirming instruction tuning's efficacy.

Wei et al. Finetuned Language Models Are Zero-Shot Learners, ICLR 2022

FLAN

Key Milestones:

- **Unified Prompting:** Introduced instruction tuning by unifying prompts across diverse tasks
- **Further Scale Multi-Tasking:** Demonstrated that scaling multi-task capabilities enhances model performance by training on a wide array of tasks simultaneously.
- **Chain-of-Thought (CoT):** Incorporated CoT prompting to improve the model's multi-step reasoning, leading to more coherent and effective problem-solving.



Credit: *Finetuned Language Models Are Zero-Shot Learners.*

InstructGPT – Instruction Tuning + Human Feedback

- **Method**
  - **Supervised Fine-tuning** on instruction–response pairs.
  - **RLHF** (Reinforcement Learning from Human Feedback): Collect preference data, train a reward model, and refine GPT to optimize for user alignment.
- **Why It Matters**
  - Prioritizes *helpfulness*, *harmlessness*, *human alignment*.
  - Directly led to systems like **ChatGPT**.
- **Data Efficiency Angle**
  - Key insight: A relatively smaller amount of **human feedback** data can significantly improve the *quality* and *usability* of large language models.

Ouyang et al., Training language models to follow instructions with human feedback, NeurIPS. 2022

InstructGPT

Key Milestone:

- **Instruction Tuning:** Introduced a novel approach using reinforcement learning from human feedback (RLHF) to align model outputs with user instructions.

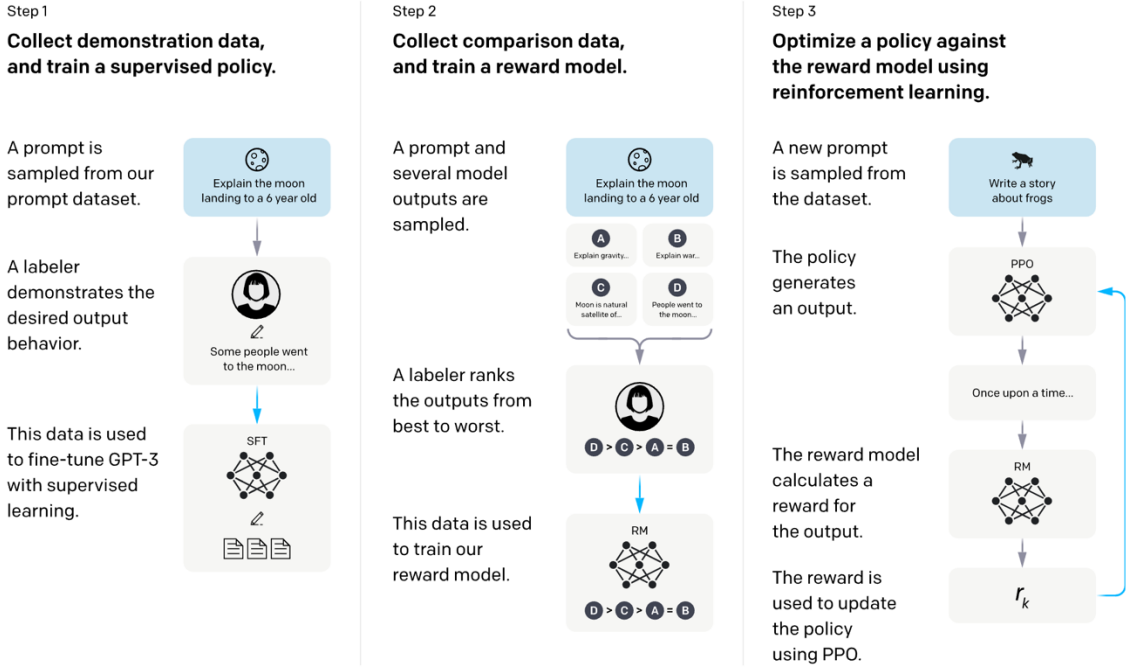


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

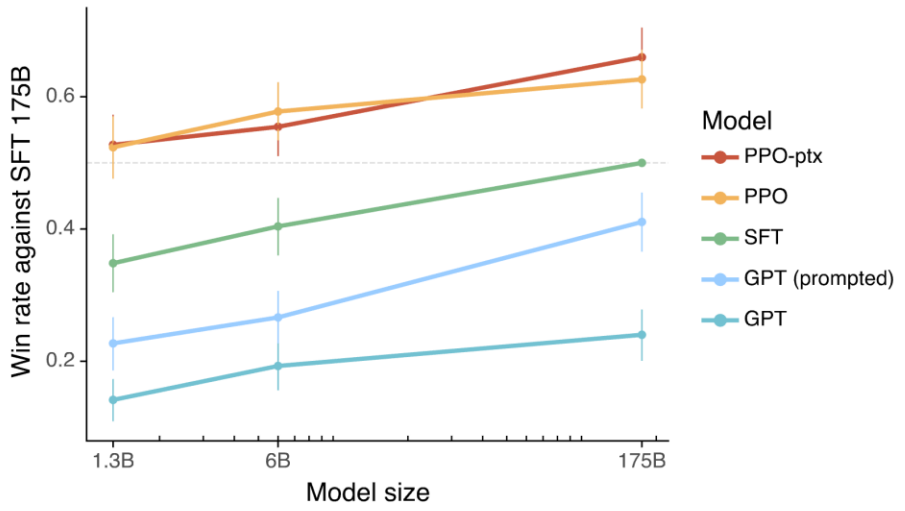


Image from: Ouyang et al., Training language models to follow instructions with human feedback, NeurIPS. 2022

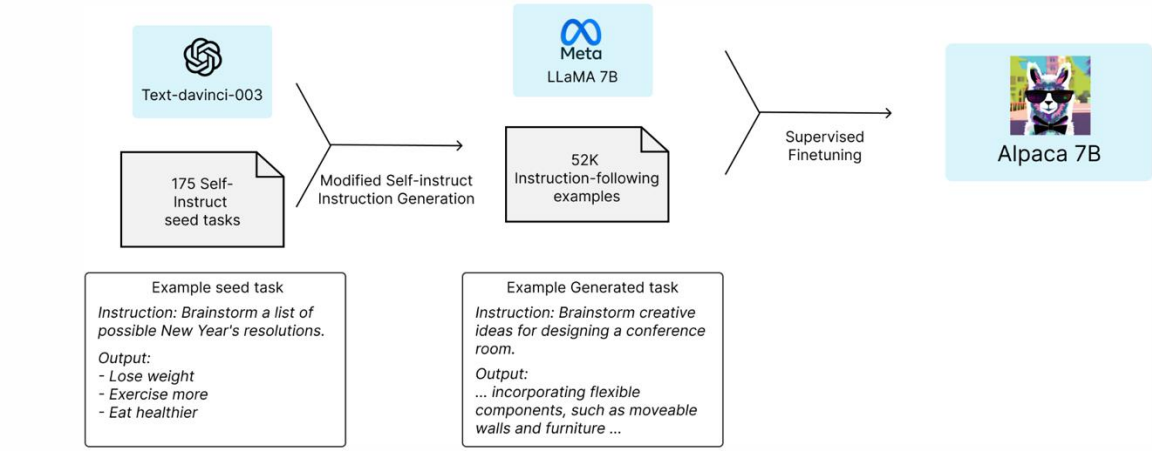
Synthetic Data

- Motivation
  - **Limited Data**: Real-world labeled data is often expensive or limited.
  - **Privacy Concerns**: Need to avoid using sensitive user data directly.
  - **Scalability & Diversity**: Synthetic data can provide large, varied training sets at lower cost.

- High-Level Ideas
  - **LLMs as Data Generators**: Using large models (GPT, etc.) to create question–answer pairs, reasoning steps, or domain-specific samples.
  - **Controlled or Semi-automatic Pipelines**: Combining LLM outputs with filtering, validation, or iterative refinement to ensure quality.

Alpaca – Self-Instruct Applied

- **Process**
  - Use GPT-3.5 to generate ~52k instruction–response pairs automatically.
  - Fine-tune a **7B LLaMA** model on this synthetic set.
- **Results**
  - A surprisingly *ChatGPT-like* model with minimal direct human annotation.
  - Demonstrates the **low-cost** route to building an instruction-following model.
- **Significance**
  - **Proof of concept**: Synthetic data from strong models can rapidly boost smaller models' performance on instruction tasks.



Credit: *"Alpaca" by Taori et al. Stanford*.

Orca— Chain of Thought Matters

- **Main Contribution**
  - Introduces **complex explanation traces** (chain-of-thought, step-by-step reasoning) from a *teacher* model (GPT-4 or other strong LLM).
  - A **student** LLM then fine-tunes on these CoT-augmented synthetic examples, achieving stronger reasoning performance.

- **Why It's Important**
  - **Richer Synthetic Signals**: Instead of just input–output pairs, each example includes *how* the teacher arrived at its conclusion.
  - **Boosted Reasoning Ability**: Students learn *reasoning patterns*, not just final answers—often outperforming models trained on plain Q–A data.
  - **Better Data Utilization**: A moderate amount of CoT-labeled data can sometimes surpass large amounts of non-explanatory data.

Mukherjee et al., Orca: Progressive Learning from Complex Explanation Traces of GPT-4. ArXiv: 2306.02707, 2023.

LIMA-Less Is More for Alignment

- **Core Idea**
  - Fine-tune a 65B LLaMA model on just **1,000 carefully curated** instruction–response pairs.
  - Achieves performance *comparable* to models trained on tens or hundreds of thousands of synthetic instructions.
- **Key Takeaways**
  - **High-Quality Beats Large Quantity**: A small but *expertly chosen* dataset can often yield near state-of-the-art alignment.
  - **Focus on Diversity & Difficulty**: Curating diverse queries ensures coverage, while carefully validated responses maintain correctness.
  - **Implication**: Massive data hunts aren't always necessary; methodical selection and curation can provide *most* of the benefit.

Zhou et al. LIMA: Less Is More for Alignment. NeurIPS. 2023

Synthetic Data – Expanding Training Data

- Key Challenges
  - **Quality Control**: Avoid hallucinations, incorrect labels, or biases in generated data.
  - **Balancing Quantity vs. Quality**: Large-scale synthetic sets can still overfit or degrade performance if noisy.
  - **Evaluation Contamination**: Synthetic data may contain rephrasing of test examples, complicating fair benchmarking.

- This raises an important question:
  Does synthetic data help models generalize, or does it reinforce memorization?

From Synthetic Data to Post-Training – SFT vs. RL

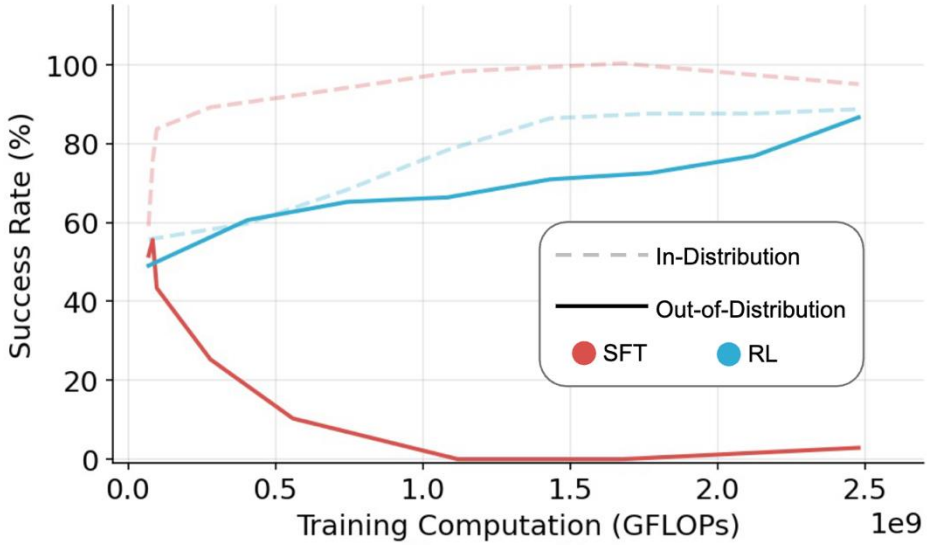
- **Synthetic data provides large-scale training resources**, but post-training methods determine how models use this data effectively.
- **Key Question:**
  - Does supervised fine-tuning (SFT) make models **overfit to training data**?
  - Can reinforcement learning (RL) **improve generalization** to unseen tasks?
- **Transitioning to Post-Training:**
  - **SFT (Supervised Fine-Tuning):** Ensures instruction adherence but may lead to memorization.
  - **RL (Reinforcement Learning):** Encourages flexibility but is harder to optimize.

- The recent study **"SFT Memorizes, RL Generalizes"** provides insights into these trade-offs.

Chu et al., SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. arXiv:2501.17161. 2025

SFT Memorizes, RL Generalizes

- **Core Question:**
  - **What are the strengths and weaknesses of SFT vs. RL** in foundation model post-training?
- **Key Findings:**
  - **SFT (Supervised Fine-Tuning)**
    - Memorizes seen data well.
    - Struggles with out-of-distribution (OOD) generalization.
    - Effective for tasks with **fixed structures**.
  - **RL (Reinforcement Learning)**
    - Adapts to unseen variations.
    - Improves **OOD generalization**.
    - Harder to train due to reward optimization complexity.



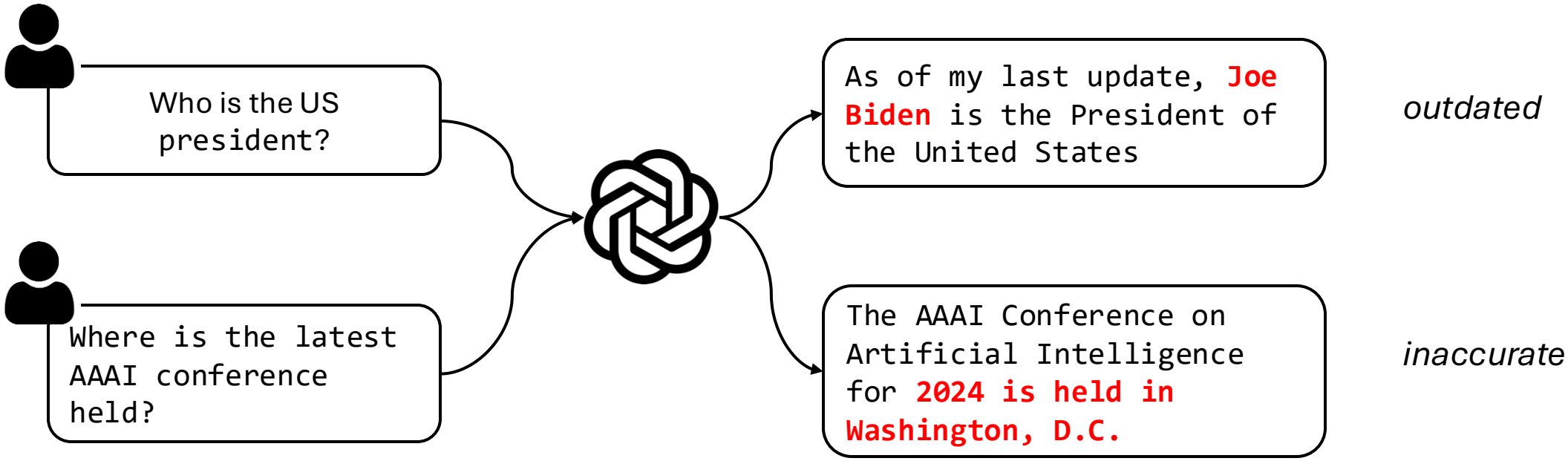
- **Conclusion:**
  - **SFT is good for stability, RL is better for adaptability.**
  - Combining **SFT + RL** leads to the best trade-off between memorization and generalization.

Image from: Chu et al., SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. arXiv:2501.17161. 2025

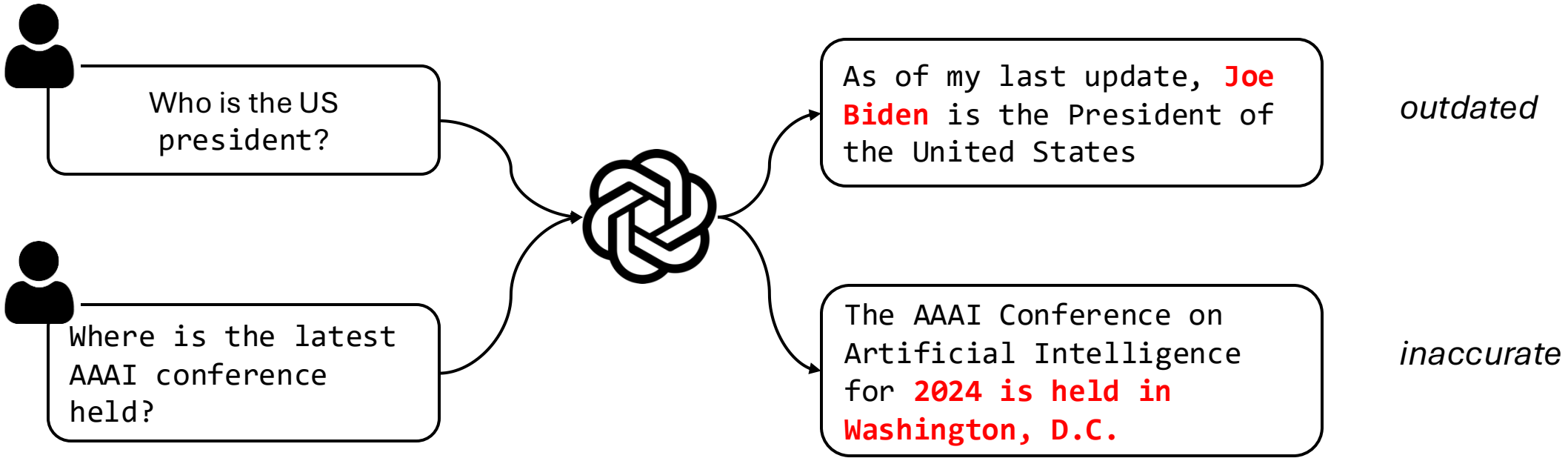
Outline

- Model Efficiency
  - Parameter-efficient Fine-tuning
  - Model Compression
  - Inference Acceleration
- Data Efficiency
  - Zero/few-shot Learning
  - Knowledge Editing
  - Retrieval-augmented Generation

Why Editing?


Who is the US president?
Where is the latest AAAI conference held?
As of my last update, Joe Biden is the President of the United States
The AAAI Conference on Artificial Intelligence for 2024 is held in Washington, D.C.
outdated
inaccurate

Why Editing?



- LLMs learn *outdated* and *inaccurate* knowledge from their training data.

Ways to update a LLM's knowledge

Fine-tuning

- Expensive
- Hurt others

Editing

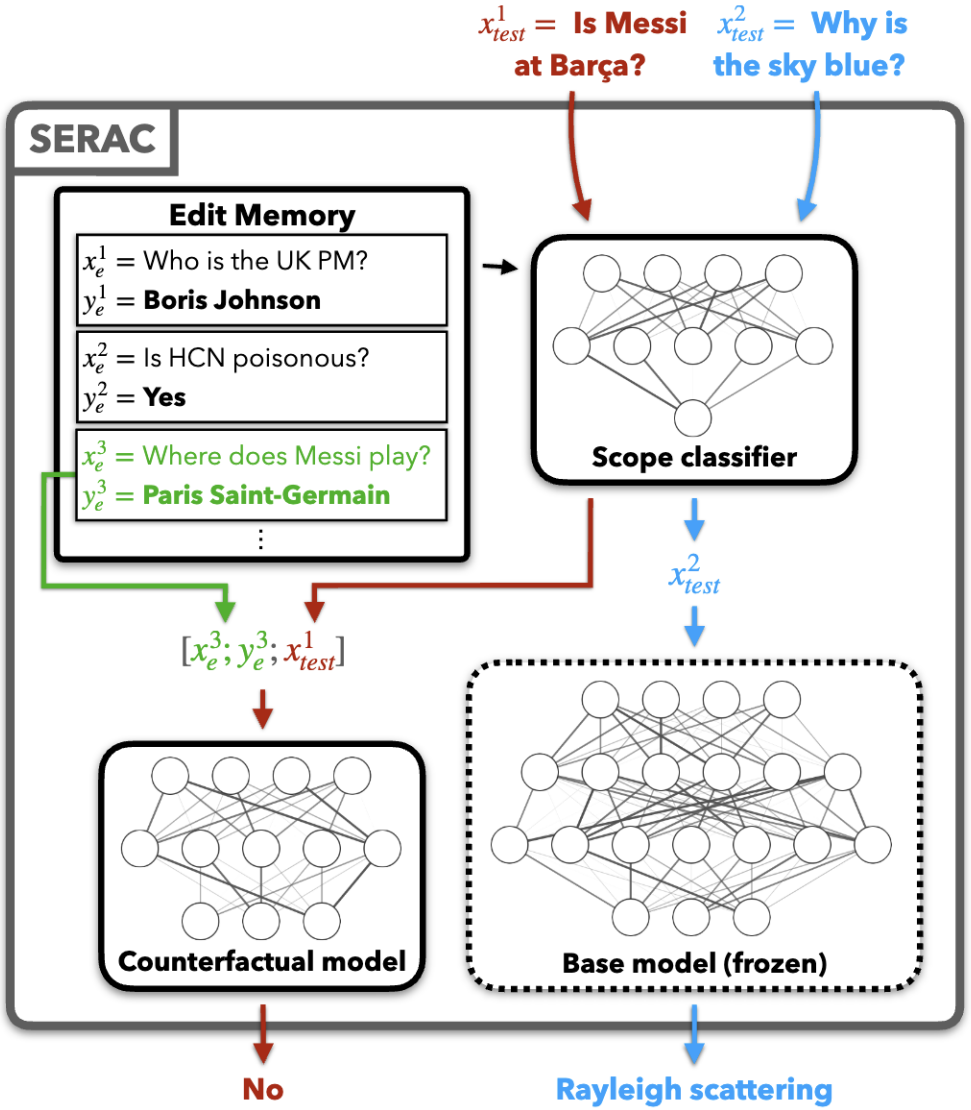
Good Editing

- **Reliability:** *Who is the US president?*
- **Generality:** *Who currently holds the office of President of the United States?*
- **Portability:** *Who is the first lady of the US?*
- **Locality:** *Who is the prime minister of the UK?*

SERAC

Keep model parameters and learn **new**:

- **Counterfactual model:** handle *new knowledge*.
- **Scope classifier:** *which model to use*.



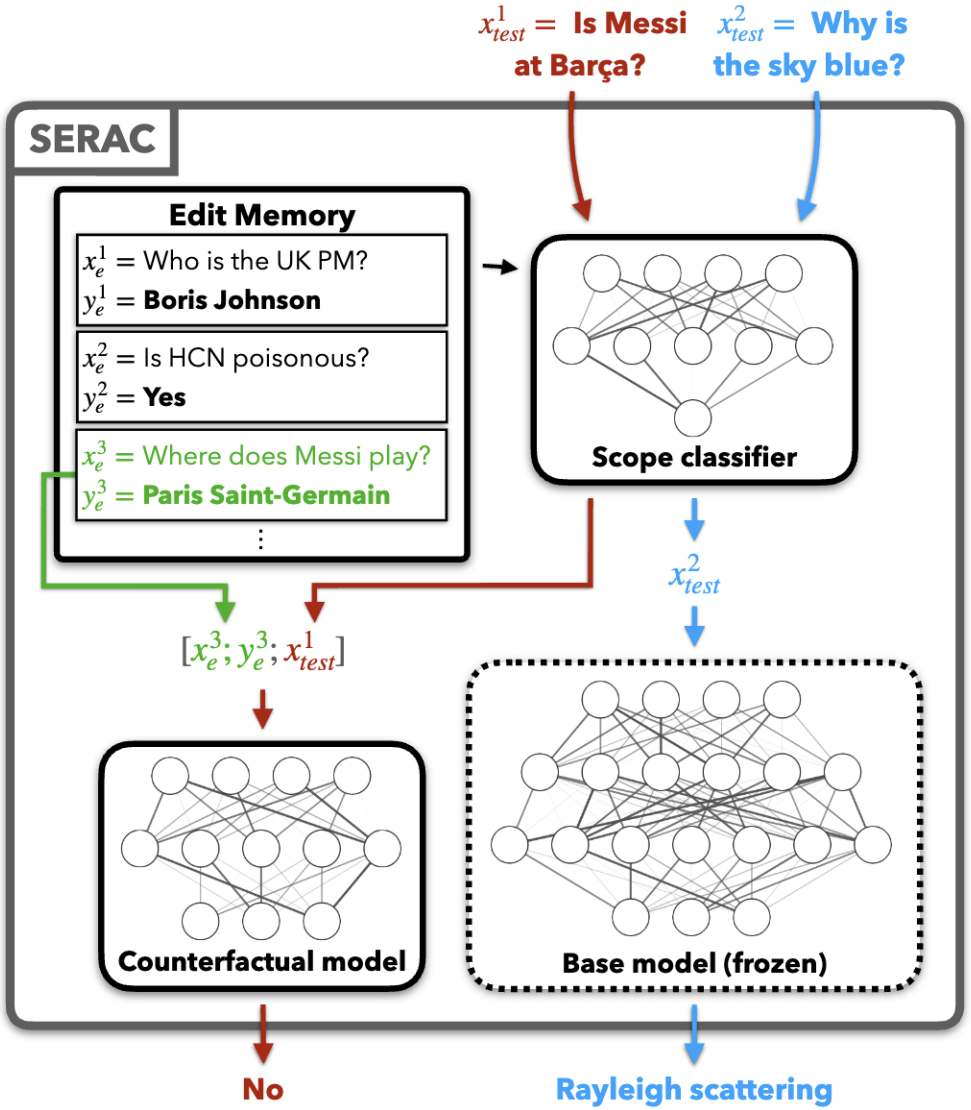
Mitchel et al. "*Memory-Based Model Editing at Scale.*" ICML. 2022.

SERAC

Keep model parameters and learn **new**:

- **Counterfactual model:** handle *new knowledge*.
- **Scope classifier:** *which model to use.*

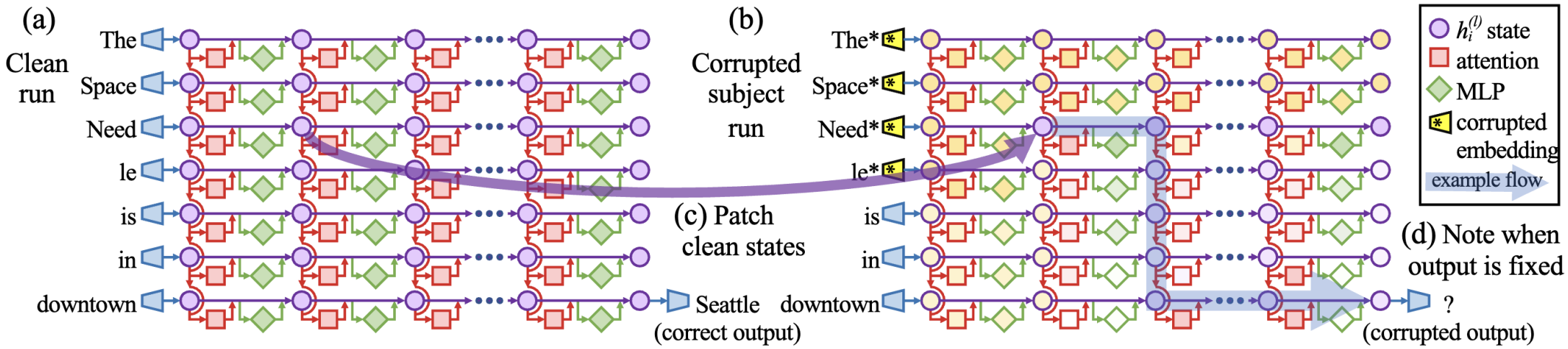
  Less efficient



Mitchel et al. "*Memory-Based Model Editing at Scale.*" ICML. 2022.

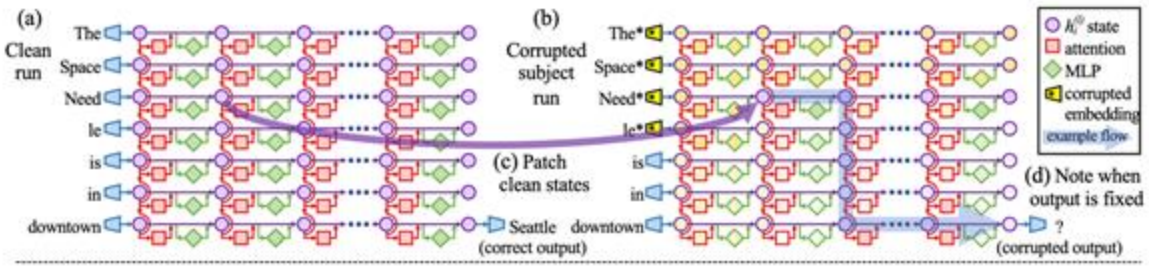
ROME

- **Locate-and-Edit paradigm:**
- Locate: *find a few relevant parameters.*
- Edit: *update these parameters only.*



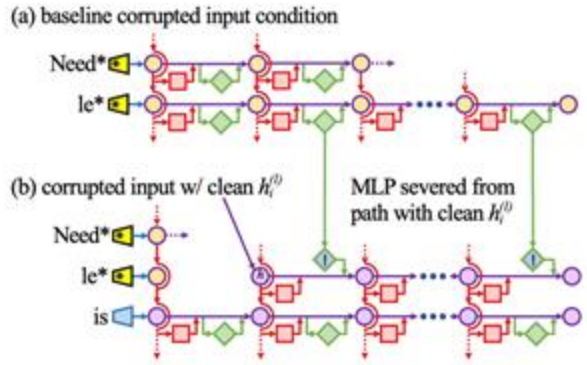
Meng et al. "*Locating and Editing Factual Associations in GPT*." NeurIPS, 2022.

ROME

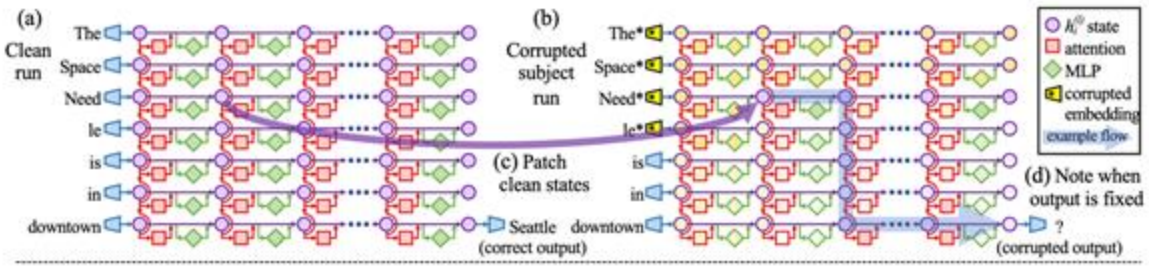


Locate: *Find a layer that affects the final prediction most.*

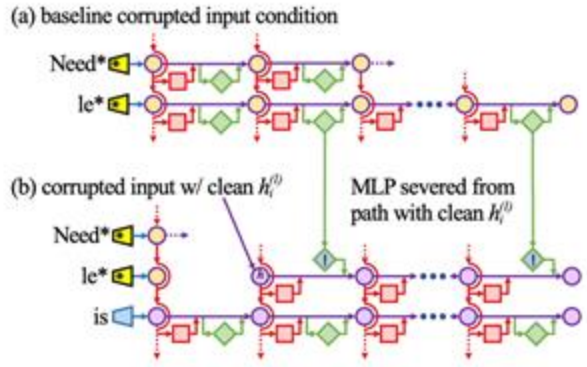
- *Check how the layer can casually lead to the correct knowledge.*



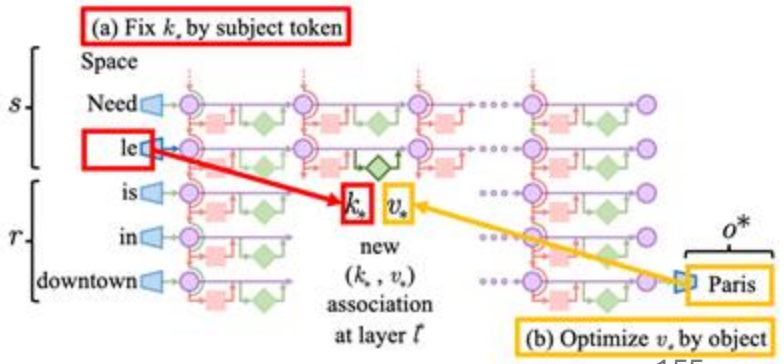
ROME



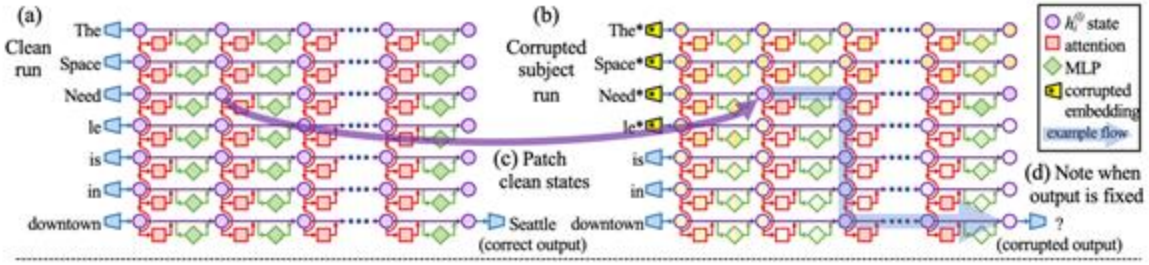
Locate: *Find a layer that affects the final prediction most.*



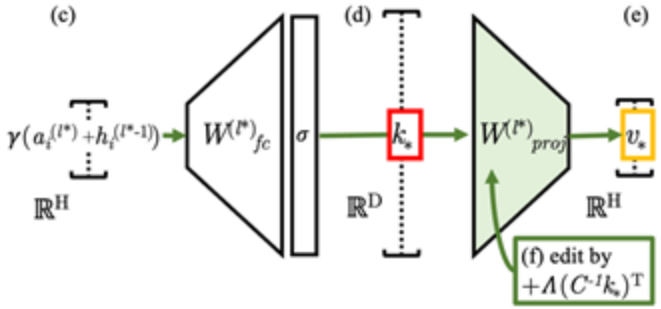
Edit:
*Add a **new association** to the located layer to output new knowledge.*



ROME



Locate: *Find a layer that affects the final prediction most.*

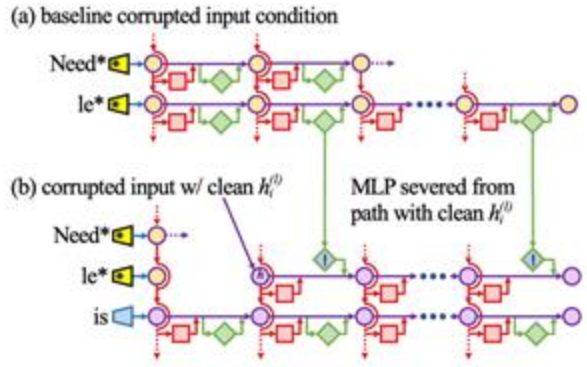


Update: *closed-form.*

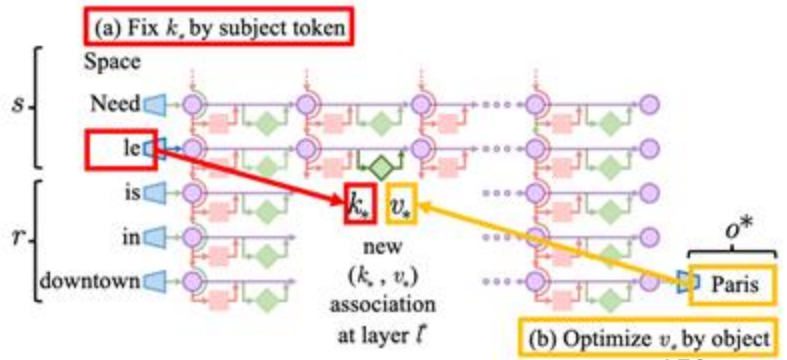
maintain pretrained edit new

minimize $\|\hat{W}K - V\|$ such that $\hat{W}k_* = v_*$

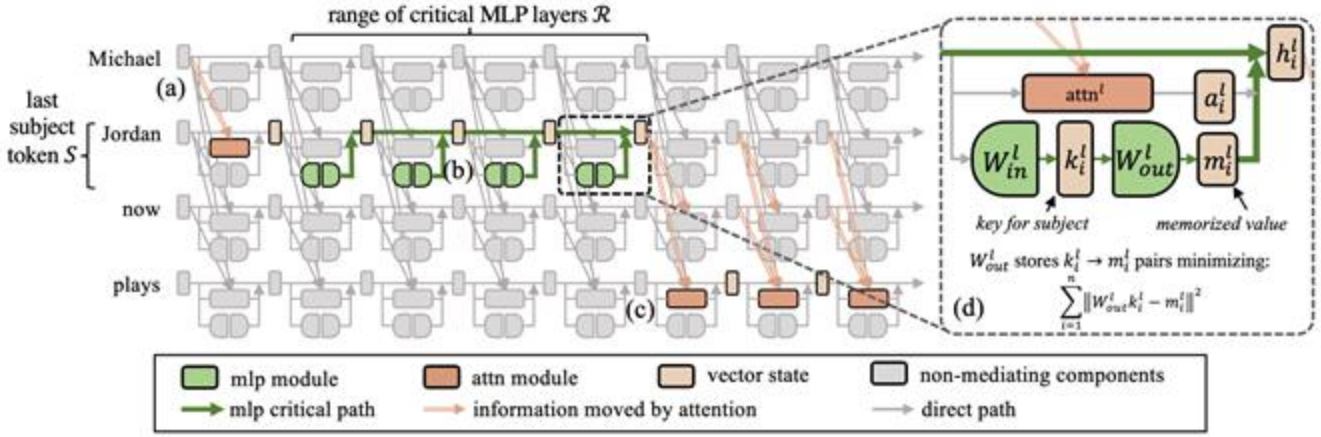
$$\hat{W} = W + \Lambda(C^{-1}k_*)^T.$$



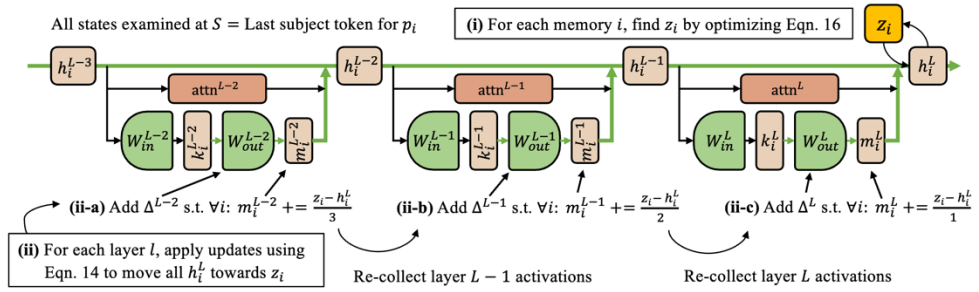
Edit:
*Add a **new association** to the located layer to output new knowledge.*



MEMIT



Locate: *Find several layers*

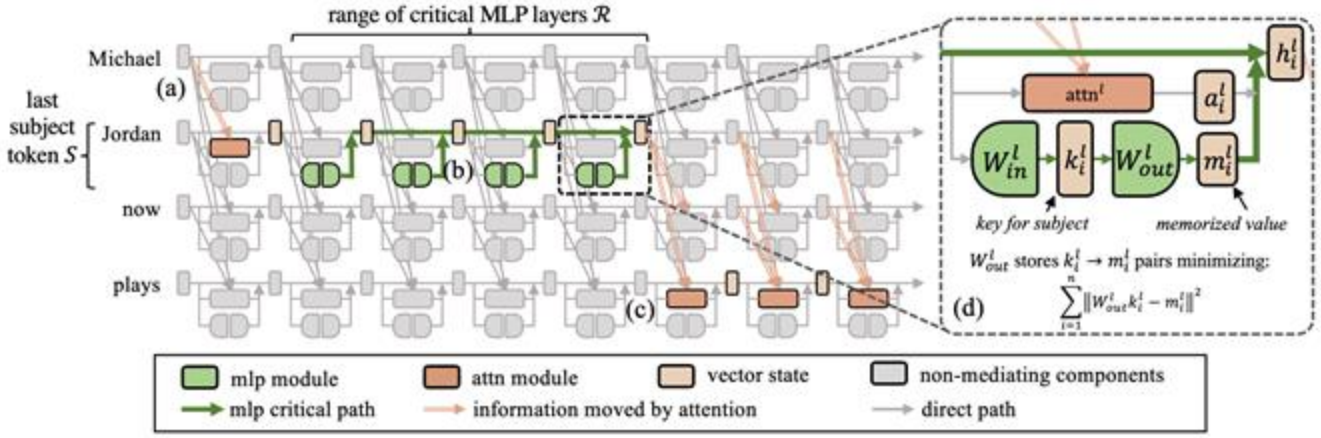


Multiple Knowledge

Meng et al. "*Mass-Editing Memory in a Transformer*." ICLR. 2023.

MEMIT

Multiple Knowledge

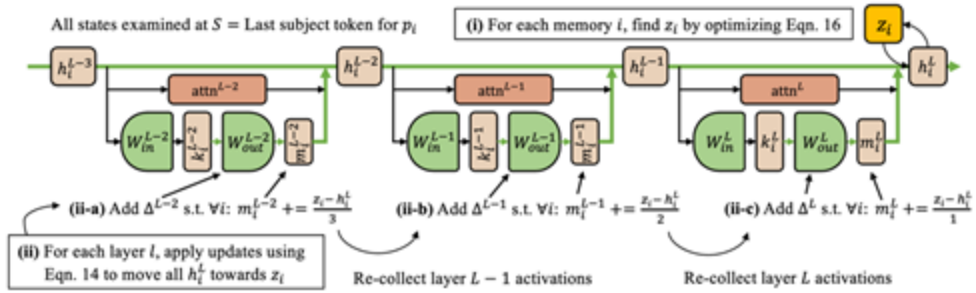


Locate: *Find several layers*

Update: *from shallower to deeper layers:*

$$W_1 \triangleq \underset{\hat{W}}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \left\| \hat{W} k_i - m_i \right\|^2 + \sum_{i=n+1}^{n+u} \left\| \hat{W} k_i - m_i \right\|^2 \right).$$

maintain pretrained edit new



Edit:
Each knowledge needs an association.

RoseLoRA

LoRA:

- Strong PEFT for general-purposed tasks.
- Make **dense** update, i.e., product of two dense low-rank matrices.
- Hard to **selectively** modify specific knowledge.

Wang et al. "*Roselora: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning.*" EMNLP 2024.

RoseLoRA

LoRA:

- Strong PEFT for general-purposed tasks.
- Make **dense** update, i.e., product of two dense low-rank matrices.
- Hard to **selectively** modify specific knowledge.

Solution: ***sparse*** *update*.

LoRA update

$$\min_{\boldsymbol{A},\boldsymbol{B}} \quad \mathcal{L}(\mathcal{D}; \boldsymbol{W}^o + \boldsymbol{BA})$$

$$\text{s.t.} \quad \frac{\|\boldsymbol{BA}\|_0}{d_1 d_2} \leq \tau,$$

Sparsity

RoseLoRA

LoRA:

- Strong PEFT for general-purposed tasks.
- Make **dense** update, i.e., product of two dense low-rank matrices.
- Hard to **selectively** modify specific knowledge.

Proposition. *The sparsity of* $\boldsymbol{BA}$ *is greater or equal to* $max\{0,1+\sum_{i=1}^{r}(s(\boldsymbol{A}_{i*})+s(\boldsymbol{B}_{*i})-s(\boldsymbol{A}_{i*})s(\boldsymbol{B}_{*i}))-r\}$.

Solution: ***sparse*** *update*.

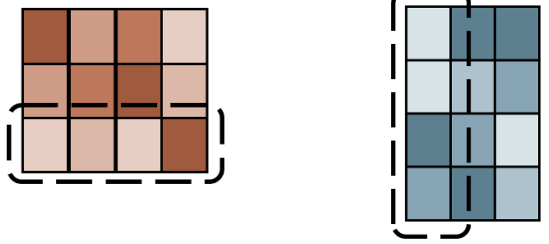
LoRA update

$$\min_{A,B} \mathcal{L}(\mathcal{D}; \boldsymbol{W}^o + \boldsymbol{BA})$$
$$\text{s.t.} \quad \frac{\|\boldsymbol{BA}\|_0}{d_1 d_2} \leq \tau,$$

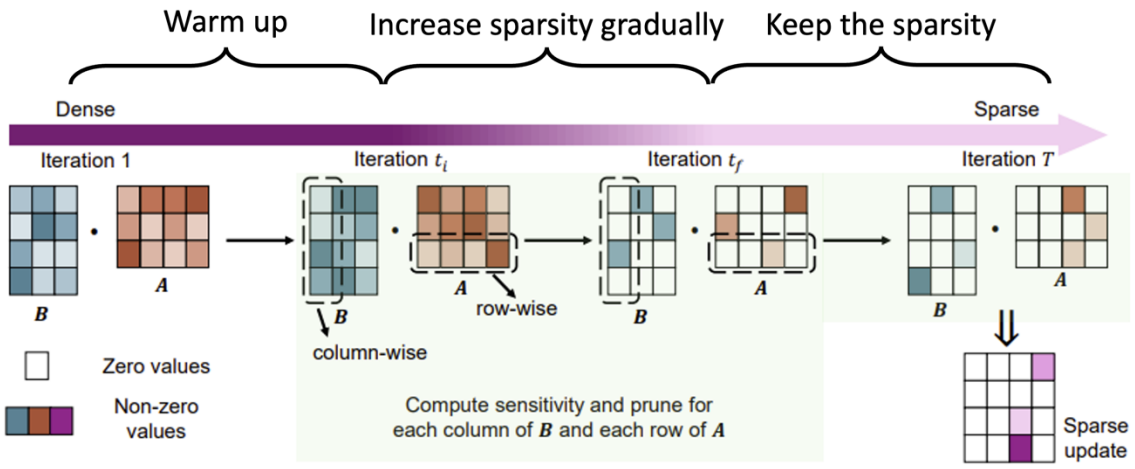
Sparsity

Challenge:
Product parameterization

$$\min_{A,B} \mathcal{L}(\mathcal{D}; \boldsymbol{W^0} + \boldsymbol{BA})$$
$$\text{s.t.} \quad \frac{\|\boldsymbol{A}_{i*}\|_0}{d_2} \leq \tau, \frac{\|\boldsymbol{B}_{*i}\|_0}{d_1} \leq \tau, i = 1, \ldots, r.$$



RoseLoRA



Dataset	Metric	FT-L$^{\heartsuit}$	AdaLoRA$^{\heartsuit}$	ROME$^{\heartsuit}$	MEMIT$^{\heartsuit}$	RoseLoRA
$\text{WikiData}_{\text{recent}}$	Edit Succ.(↑)	71.2	65.6	85.1	85.3	**98.4**
	Locality(↑)	63.7	55.8	66.2	64.8	**83.4**
	Portability(↑)	48.7	47.2	37.5	37.9	**54.3**
	Fluency(↑)	549	538	574	567	**585**
	AVG(↑)	59.6	55.6	61.5	61.2	**73.7**
$\text{WikiData}_{\text{counterfact}}$	Edit Succ.(↑)	51.1	72.1	83.2	83.4	**99.4**
	Locality(↑)	62.5	66.8	65.4	63.7	**90.9**
	Portability(↑)	39.1	55.2	38.7	40.1	**57.2**
	Fluency(↑)	545	554	579	569	**592**
	AVG(↑)	51.8	62.4	61.3	61.0	**76.7**
ZsRE	Edit Succ.(↑)	51.1	72.1	83.2	83.4	**100**
	Locality(↑)	62.5	66.8	65.4	63.7	**92.5**
	Portability(↑)	39.1	**55.2**	38.7	40.1	50.9
	Fluency(↑)	545	554	**579**	569	574
	AVG(↑)	54.6	62.1	58.2	54.0	**75.2**
WikiBio	Edit Succ.(↑)	66.3	97.0	95.1	94.3	**99.5**
	Locality(↑)	60.1	57.9	47.0	51.6	**92.5**
	Fluency(↑)	604	616	617	617	**620**
	AVG(↑)	62.3	72.2	67.9	69.2	**84.6**

GRACE

Previous methods are good at editing pre-specified number of knowledge.

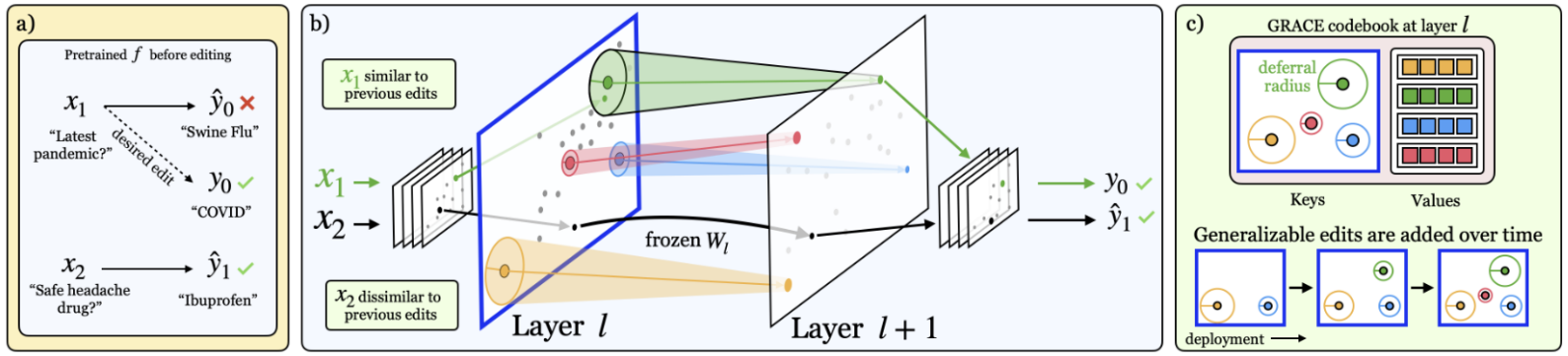
But how to edit new knowledge that emerge continually?

Hartvigsen et al. "*Aging with grace: Lifelong model editing with discrete key-value adaptors.*" NeurIPS, 2023.

GRACE

But how to edit new knowledge that emerge continually?

Use dynamic parameters.

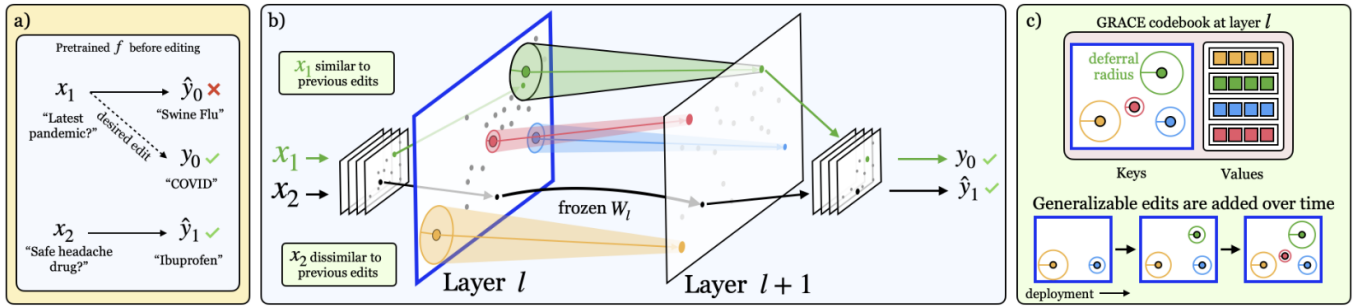


GRACE

Algorithm 1: Update Codebook at layer l.

Input: $\mathcal{C} = \{(\mathbb{K}_i, \mathbb{V}_i, \epsilon_i)\}_{i=0}^{C-1}$, codebook
Input: $f(\cdot)$, model
Input: y_t, desired label
Input: x_t, edit input for which $f(x_t) \neq y_t$
Input: ϵ_{init}, initial ϵ
Input: $\mathrm{d}(\cdot)$, distance function
Output: $\mathcal{C}$, updated codebook
$C = \|\mathcal{C}\|$
$\hat{y}, h^{l-1} = f^L(x_t), f^{l-1}(x_t)$
$\mathrm{d}_{\text{min}}, i = \min_i(d(h^{l-1}, \mathbb{K}_i))$
If $\mathrm{d}_{\text{min}} > \epsilon_i + \epsilon_{\text{init}}$ or $C = 0$:
  # h^{l-1} far from existing entries or empty $\mathcal{C}$
  $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
  $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \epsilon_{\text{init}})$ # *Add entry*
Else:
  # h^{l-1} near existing entries
  If $f^L(k_i) = y$:
    # *Same label → Expand*
    $\mathcal{C}_i := (k_i, v_i, \epsilon_i + \epsilon_{\text{init}})$
  Else:
    # *Different label → Split*
    $\mathcal{C}_i = (k_i, v_i, \mathrm{d}_{\text{min}}/2)$ # *Update entry i*
    $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
    $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \mathrm{d}_{\text{min}}/2)$ # *Add entry*
return: $\mathcal{C}$

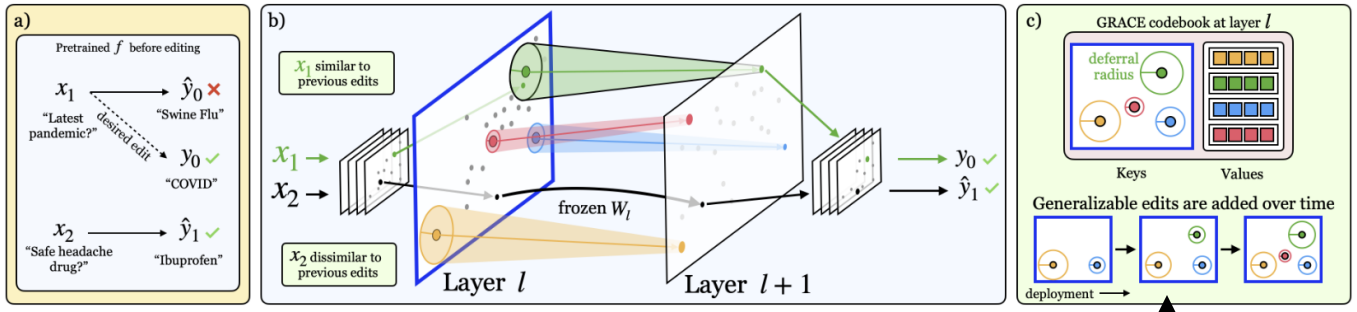
Use dynamic parameters.



GRACE

Algorithm 1: Update Codebook at layer l.

Input: $\mathcal{C} = \{(\mathbb{K}_i, \mathbb{V}_i, \epsilon_i)\}_{i=0}^{C-1}$, codebook
Input: $f(\cdot)$, model
Input: y_t, desired label
Input: x_t, edit input for which $f(x_t) \neq y_t$
Input: ϵ_{init}, initial ϵ
Input: $\mathrm{d}(\cdot)$, distance function
Output: $\mathcal{C}$, updated codebook
$C = \|\mathcal{C}\|$
$\hat{y}, h^{l-1} = f^L(x_t), f^{l-1}(x_t)$
$\mathrm{d}_{\text{min}}, i = \min_i(d(h^{l-1}, \mathbb{K}_i))$
If $\mathrm{d}_{\text{min}} > \epsilon_i + \epsilon_{\text{init}}$ or $C = 0$:
  # h^{l-1} far from existing entries or empty $\mathcal{C}$
  $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
  $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \epsilon_{\text{init}})$ # *Add entry*
Else:
  # h^{l-1} near existing entries
  If $f^L(k_i) = y$:
    # *Same label → Expand*
    $\mathcal{C}_i := (k_i, v_i, \epsilon_i + \epsilon_{\text{init}})$
  Else:
    # *Different label → Split*
    $\mathcal{C}_i = (k_i, v_i, \mathrm{d}_{\text{min}}/2)$ # *Update entry i*
    $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
    $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \mathrm{d}_{\text{min}}/2)$ # *Add entry*
return: $\mathcal{C}$



Continual Editing: *gradually grow number of parameters to learn new knowledge*

GRACE

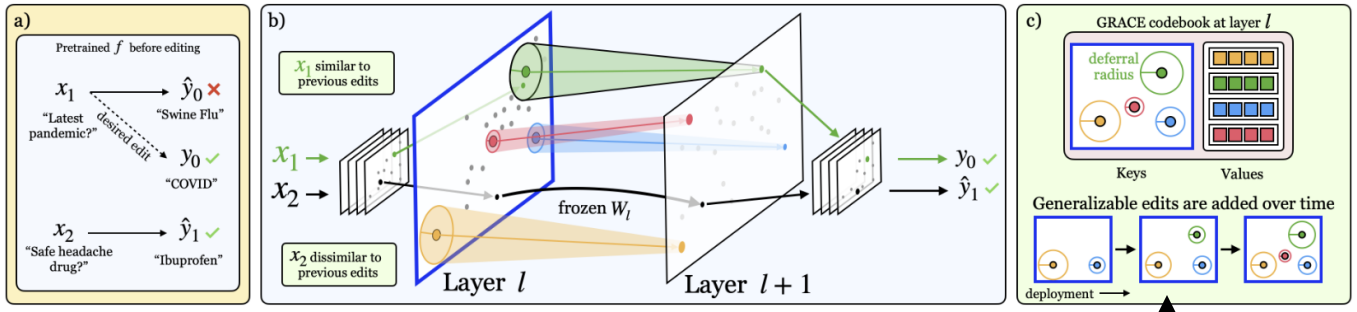
Algorithm 1: Update Codebook at layer l.

Input: $\mathcal{C} = \{(\mathbb{K}_i, \mathbb{V}_i, \epsilon_i)\}_{i=0}^{C-1}$, codebook
Input: $f(\cdot)$, model
Input: y_t, desired label
Input: x_t, edit input for which $f(x_t) \neq y_t$
Input: ϵ_{init}, initial ϵ
Input: $\mathrm{d}(\cdot)$, distance function
Output: $\mathcal{C}$, updated codebook

$C = \|\mathcal{C}\|$
$\hat{y}, h^{l-1} = f^L(x_t), f^{l-1}(x_t)$
$\mathrm{d}_{\text{min}}, i = \min_i(d(h^{l-1}, \mathbb{K}_i))$
If $\mathrm{d}_{\text{min}} > \epsilon_i + \epsilon_{\text{init}}$ or $C = 0$:
  # h^{l-1} far from existing entries or empty $\mathcal{C}$
  $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
  $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \epsilon_{\text{init}})$ # *Add entry*
Else:
  # h^{l-1} near existing entries
  If $f^L(k_i) = y$:
    # *Same label → Expand*
    $\mathcal{C}_i := (k_i, v_i, \epsilon_i + \epsilon_{\text{init}})$
  Else:
    # *Different label → Split*
    $\mathcal{C}_i = (k_i, v_i, \mathrm{d}_{\text{min}}/2)$ # *Update entry i*
    $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
    $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \mathrm{d}_{\text{min}}/2)$ # *Add entry*
return: $\mathcal{C}$

$$h^l = \begin{cases} \text{GRACE}(h^{l-1}) & \text{if } \min_i(d(h^{l-1}, \mathbb{K}_i)) < \epsilon_{i_*}, \text{ where } i_* = \text{argmin}_i(d(h^{l-1}), \mathbb{K}_i), \\ f^l(h^{l-1}) & \text{otherwise,} \end{cases}$$

Inference: *avoid incorrect edits*



Continual Editing: *gradually grow number of parameters to learn new knowledge*

GRACE

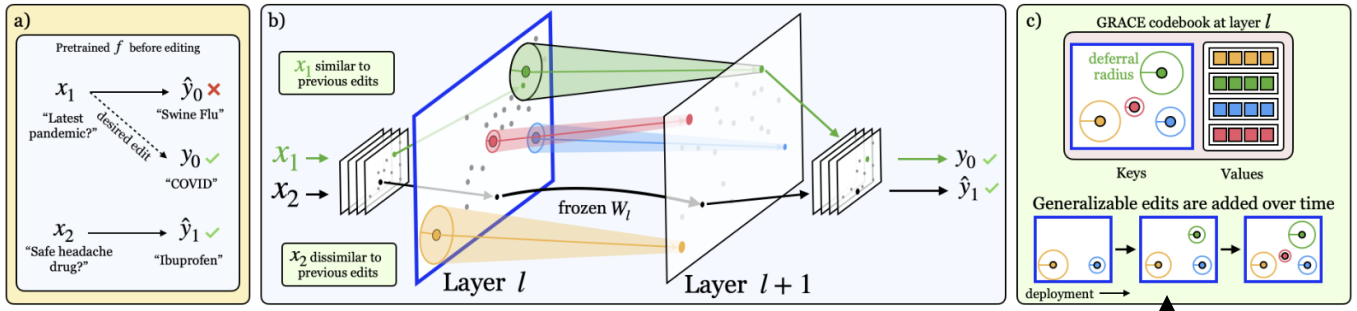
Algorithm 1: Update Codebook at layer l.

Input: $\mathcal{C} = \{(\mathbb{K}_i, \mathbb{V}_i, \epsilon_i)\}_{i=0}^{C-1}$, codebook
Input: $f(\cdot)$, model
Input: y_t, desired label
Input: x_t, edit input for which $f(x_t) \neq y_t$
Input: ϵ_{init}, initial ϵ
Input: $\mathrm{d}(\cdot)$, distance function
Output: $\mathcal{C}$, updated codebook

$C = \|\mathcal{C}\|$
$\hat{y}, h^{l-1} = f^L(x_t), f^{l-1}(x_t)$
$\mathrm{d}_{\text{min}}, i = \min_i(d(h^{l-1}, \mathbb{K}_i))$
If $\mathrm{d}_{\text{min}} > \epsilon_i + \epsilon_{\text{init}}$ or $C = 0$:
  # h^{l-1} far from existing entries or empty $\mathcal{C}$
  $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
  $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \epsilon_{\text{init}})$ # *Add entry*
Else:
  # h^{l-1} near existing entries
  If $f^L(k_i) = y$:
    # *Same label → Expand*
    $\mathcal{C}_i := (k_i, v_i, \epsilon_i + \epsilon_{\text{init}})$
  Else:
    # *Different label → Split*
    $\mathcal{C}_i = (k_i, v_i, \mathrm{d}_{\text{min}}/2)$ # *Update entry i*
    $v_{\text{new}} =$ finetune on $P_f(y|v_{\text{init}})$
    $\mathcal{C}_C = (h^{l-1}, v_{\text{new}}, \mathrm{d}_{\text{min}}/2)$ # *Add entry*
return: $\mathcal{C}$

$$h^l = \begin{cases} \text{GRACE}(h^{l-1}) & \text{if } \min_i(d(h^{l-1}, \mathbb{K}_i)) < \epsilon_{i_*}, \text{ where } i_* = \text{argmin}_i(d(h^{l-1}), \mathbb{K}_i), \\ f^l(h^{l-1}) & \text{otherwise,} \end{cases}$$

Inference: *avoid incorrect edits*



Continual Editing: *gradually grow number of parameters to learn new knowledge*

Different codebooks:
- **LoRA:** MELO [AAAI 2024]
- **FFN:** WISE [NeurIPS 2024]
- …

BaFT

Parameter-based edits alter all inputs in the same way.

How to get more fine-grained updates?

ReFT [NeurIPS 2024]: alter a few representations in some low-rank subspace

$$\Phi_l(\boldsymbol{h}_i^{(l)}; \phi_l) = \boldsymbol{h}_i^{(l)} + \mathbf{R}_l^\top(\mathbf{A}_l \boldsymbol{h}_i^{(l)} + \boldsymbol{b}_l - \mathbf{R}_l \boldsymbol{h}_i^{(l)}),$$

Liu et al. "*Unlocking Efficient, Scalable, and Continual Knowledge Editing with Basis-Level Representation Fine-Tuning.*" ICLR 2025.

BaFT

ReFT: alter pre-specified representations in a low-rank subspace

$$\Phi_l(\boldsymbol{h}_i^{(l)}; \phi_l) = \boldsymbol{h}_i^{(l)} + \mathbf{R}_l^\top(\mathbf{A}_l \boldsymbol{h}_i^{(l)} + \boldsymbol{b}_l - \mathbf{R}_l \boldsymbol{h}_i^{(l)}),$$ linear

Impossible *edit – generality - locality triangle*
(paraphrase)

Theorem 2.3. *When fine-tuning an LM, ReFT learns to update the old representation* $\boldsymbol{h}_0$ *to targeted* $\boldsymbol{t} = \Phi(\boldsymbol{h}_0)$. *If ReFT maintains good generality such that* $\forall\, \boldsymbol{h} \in B(\boldsymbol{h}_0, \varepsilon(\boldsymbol{h}_0))$,

$$\|\Phi(\boldsymbol{h}) - \Phi(\boldsymbol{h}_0)\| = \|\Phi(\boldsymbol{h}) - \boldsymbol{t}\| < \varepsilon(\boldsymbol{t}),$$

where $\|\cdot\|$ *denote the* ℓ_2 *norm. Then for any irrelevant input* $\boldsymbol{h}_{ir}$ *with a small stable-ball radius*

$$\varepsilon(\boldsymbol{h}_{ir}) < \frac{\|\boldsymbol{t} - \boldsymbol{h}_0\| - (\varepsilon(\boldsymbol{t}) + \varepsilon(\boldsymbol{h}_0))}{\varepsilon(\boldsymbol{t}) + 2\varepsilon(\boldsymbol{h}_0)} \varepsilon(\boldsymbol{h}_0),$$

and is not too far from $\boldsymbol{h}_0$ *such that*

$$\|\boldsymbol{h}_{ir} - \boldsymbol{h}_0\| = \varepsilon(\boldsymbol{h}_{ir}) + \varepsilon(\boldsymbol{h}_0),$$

ReFT will output $\Phi(\boldsymbol{h}_{ir}) \notin B(\boldsymbol{h}_{ir}, \varepsilon(\boldsymbol{h}_{ir}))$ *and break its locality guarantee.*

BaFT

ReFT: alter pre-specified representations in a low-rank subspace

$$\Phi_l(\boldsymbol{h}_i^{(l)}; \phi_l) = \boldsymbol{h}_i^{(l)} + \mathbf{R}_l^\top(\mathbf{A}_l\boldsymbol{h}_i^{(l)} + \boldsymbol{b}_l - \mathbf{R}_l\boldsymbol{h}_i^{(l)}),$$ linear

Impossible *edit – generality - locality triangle*
(paraphrase)

Theorem 2.3. *When fine-tuning an LM, ReFT learns to update the old representation* $\boldsymbol{h}_0$ *to targeted* $\boldsymbol{t} = \Phi(\boldsymbol{h}_0)$*. If ReFT maintains good generality such that* $\forall\, \boldsymbol{h} \in B(\boldsymbol{h}_0, \varepsilon(\boldsymbol{h}_0))$*,*

$$\|\Phi(\boldsymbol{h}) - \Phi(\boldsymbol{h}_0)\| = \|\Phi(\boldsymbol{h}) - \boldsymbol{t}\| < \varepsilon(\boldsymbol{t}),$$

where $\|\cdot\|$ *denote the* ℓ_2 *norm. Then for any irrelevant input* $\boldsymbol{h}_{ir}$ *with a small stable-ball radius*

$$\varepsilon(\boldsymbol{h}_{ir}) < \frac{\|\boldsymbol{t} - \boldsymbol{h}_0\| - (\varepsilon(\boldsymbol{t}) + \varepsilon(\boldsymbol{h}_0))}{\varepsilon(\boldsymbol{t}) + 2\varepsilon(\boldsymbol{h}_0)}\varepsilon(\boldsymbol{h}_0),$$

and is not too far from $\boldsymbol{h}_0$ *such that*

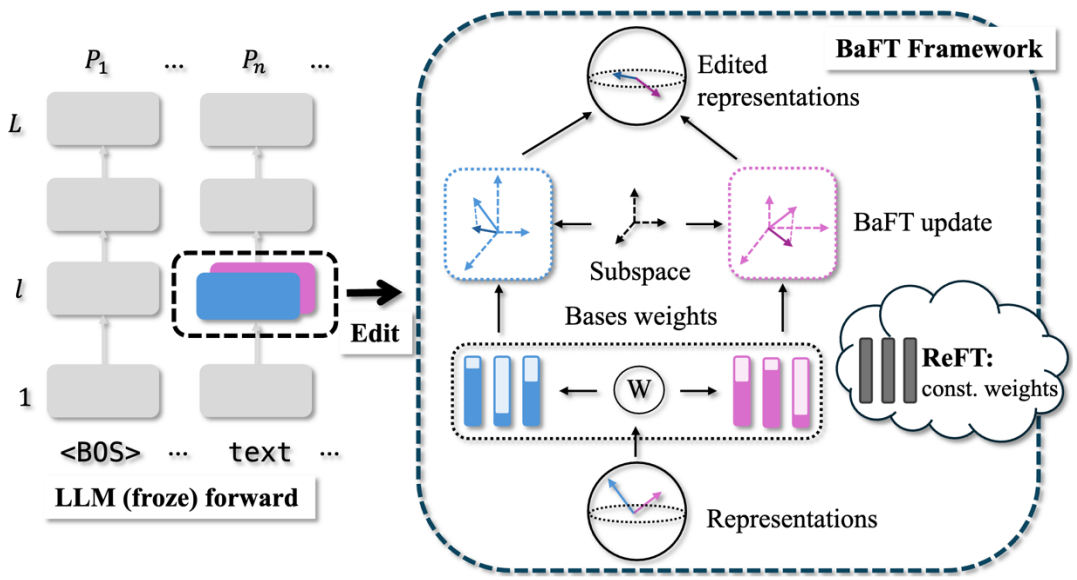
$$\|\boldsymbol{h}_{ir} - \boldsymbol{h}_0\| = \varepsilon(\boldsymbol{h}_{ir}) + \varepsilon(\boldsymbol{h}_0),$$

ReFT will output $\Phi(\boldsymbol{h}_{ir}) \notin B(\boldsymbol{h}_{ir}, \varepsilon(\boldsymbol{h}_{ir}))$ *and break its locality guarantee.*

BaFT: *use the subspace adaptively.*

$$\Phi(\boldsymbol{h}) = \boldsymbol{h} + \mathbf{R}^\top \mathbf{W}(\boldsymbol{h})\,(\mathbf{A}\boldsymbol{h} + \boldsymbol{b} - \mathbf{R}\boldsymbol{h})\,.$$ nonlinear

BaFT



ReFT: alter pre-specified representations in a low-rank subspace

$$\Phi_l(\boldsymbol{h}_i^{(l)}; \phi_l) = \boldsymbol{h}_i^{(l)} + \mathbf{R}_l^\top(\mathbf{A}_l \boldsymbol{h}_i^{(l)} + \boldsymbol{b}_l - \mathbf{R}_l \boldsymbol{h}_i^{(l)}),$$ linear

***Impossible** edit – generality - locality triangle*
(paraphrase)

Theorem 2.3. *When fine-tuning an LM, ReFT learns to update the old representation* $\boldsymbol{h}_0$ *to targeted* $\boldsymbol{t} = \Phi(\boldsymbol{h}_0)$*. If ReFT maintains good generality such that* $\forall\, \boldsymbol{h} \in B(\boldsymbol{h}_0, \varepsilon(\boldsymbol{h}_0))$*,*

$$\|\Phi(\boldsymbol{h}) - \Phi(\boldsymbol{h}_0)\| = \|\Phi(\boldsymbol{h}) - \boldsymbol{t}\| < \varepsilon(\boldsymbol{t}),$$

where $\|\cdot\|$ *denote the* ℓ_2 *norm. Then for any irrelevant input* $\boldsymbol{h}_{ir}$ *with a small stable-ball radius*

$$\varepsilon(\boldsymbol{h}_{ir}) < \frac{\|\boldsymbol{t} - \boldsymbol{h}_0\| - (\varepsilon(\boldsymbol{t}) + \varepsilon(\boldsymbol{h}_0))}{\varepsilon(\boldsymbol{t}) + 2\varepsilon(\boldsymbol{h}_0)} \varepsilon(\boldsymbol{h}_0),$$

and is not too far from $\boldsymbol{h}_0$ *such that*

$$\|\boldsymbol{h}_{ir} - \boldsymbol{h}_0\| = \varepsilon(\boldsymbol{h}_{ir}) + \varepsilon(\boldsymbol{h}_0),$$

ReFT will output $\Phi(\boldsymbol{h}_{ir}) \notin B(\boldsymbol{h}_{ir}, \varepsilon(\boldsymbol{h}_{ir}))$ *and break its locality guarantee.*

BaFT: *use the subspace adaptively.*

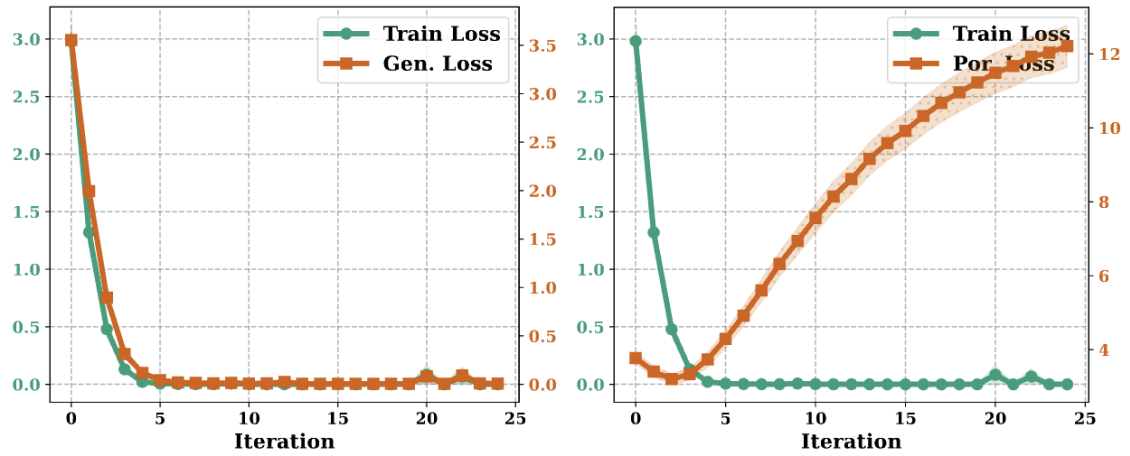
$$\Phi(\boldsymbol{h}) = \boldsymbol{h} + \mathbf{R}^\top \mathbf{W}(\boldsymbol{h})\,(\mathbf{A}\boldsymbol{h} + \boldsymbol{b} - \mathbf{R}\boldsymbol{h})\,.$$ nonlinear

BaFT

	LLaMA 2-7b(-chat)		LLaMA 3-8b-Instruct		Gemma 1.1-7b-Instruct	
	# Params.	Time (sec./edit)	# Params.	Time (sec./edit)	# Params.	Time (sec./edit)
AdaLoRA	6,292,224	26.24	5,112,576	28.71	4,817,568	44.24
FT-L	45,088,768	9.73	58,720,256	10.84	75,497,472	11.95
ROME	/	27.27	/	25.01	/	52.07
MEMIT	/	20.01	/	25.35	/	/
GRACE	/	34.38	/	87.08	/	43.45
$\text{WISE}_{\text{light}}$	5,636,096	58.00	7,340,032	65.77	9,437,184	20.20
ReFT	393,264	10.99	393,264	9.33	294,960	7.79
BaFT (Ours)	606,256	13.46	606,256	12.69	454,704	10.13

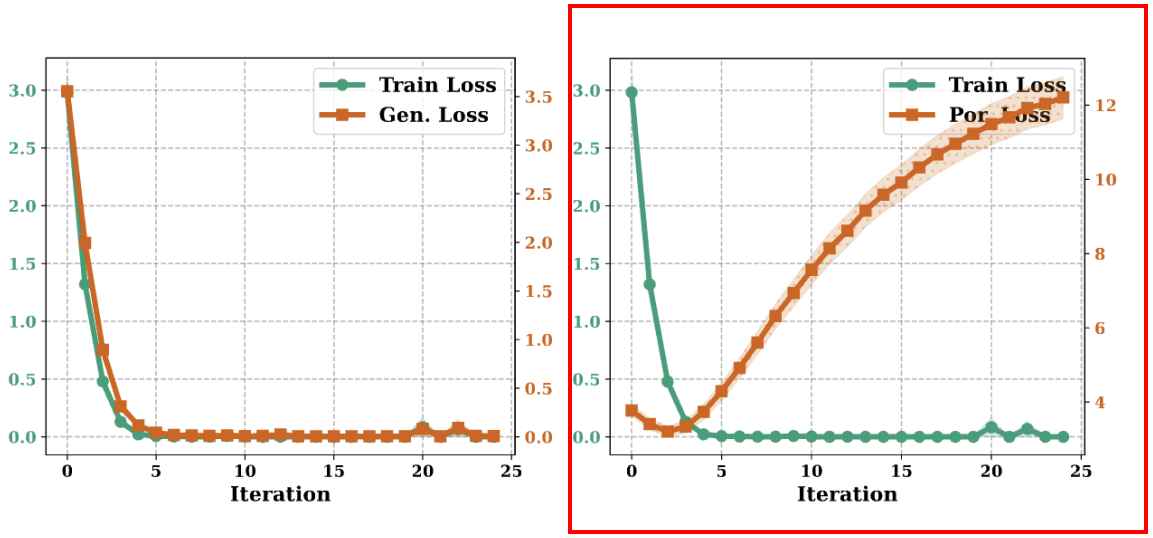
OVERTONE

Edited model tends to memorize rather than learn.



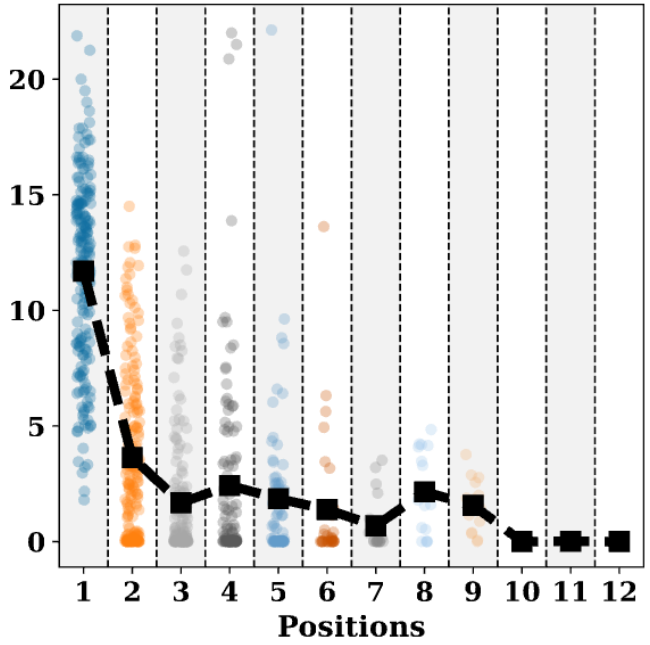
Liu et al. "*Mitigating Heterogeneous Token Overfitting in LLM Knowledge Editing*." 2025.

OVERTONE [Arxiv 2025]

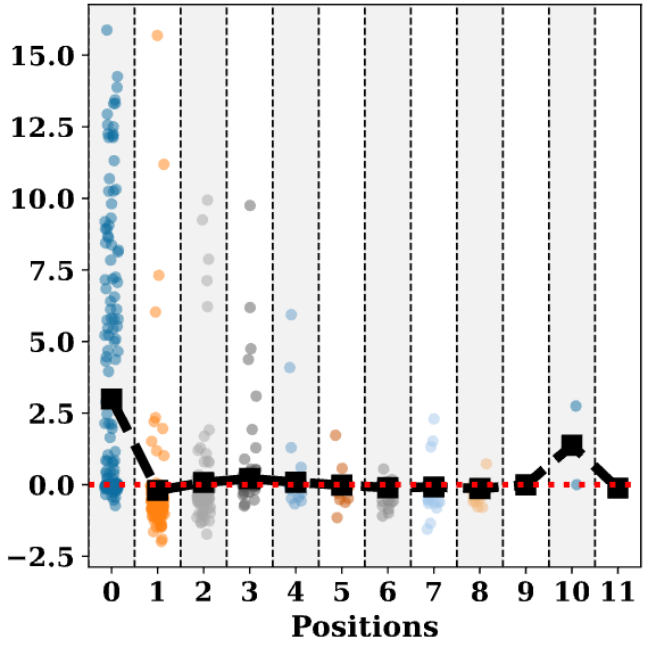


- Training: The US president is [X].
- Portability: The US first lady is?

OVERTONE [Arxiv 2025]



(a) Initial loss.



(b) Underfitting degree (UD).

$$\ell_{\mathrm{CE}}(\theta) \triangleq \sum_{i=1}^{m} \mathrm{CE}[\delta_{y_i}(y) \| \pi_\theta(y \mid \boldsymbol{x} \oplus \boldsymbol{y}_{<i})] \qquad (1)$$

$$= -\sum_{i=1}^{m} \log \pi_\theta(y_i \mid \boldsymbol{c}_i)$$

Token wise-smoothing

$$\ell_{\mathrm{OVERTONE}}(\theta) \triangleq \sum_{i=1}^{m} \max(\mathrm{D}_{\mathrm{KL}}[\pi_{\mathrm{tar}}(y \mid \boldsymbol{c}_i) \| \pi_\theta(y \mid \boldsymbol{c}_i)], \epsilon),$$

OVERTONE [Arxiv 2025]

keep “correct” token unchanged

$$\pi_{\text{tar}}^{(i)} \triangleq \begin{cases} \pi_{\text{tar}}^{\text{can}} \triangleq \lambda\delta_{y_i} + (1-\lambda)\pi_{\text{flt}}^{(i)} & \text{if } y_i = \text{argmax}_y\, \pi_{\text{tar}}^{\text{can}}, \\ \delta_{y_i} & \text{otherwise,} \end{cases} \tag{3}$$

Algorithm 1 OVERTONE Training Paradigm

Input: Editing data $(\boldsymbol{x}, \boldsymbol{y} = [y_1, \ldots, y_m])$, LM parameters θ_0, mixing hyper-parameter λ, early-stopping threshold ϵ, filtering threshold n, total training steps T.
Initialize: $\theta = \theta_0$.
for $t = 1, \ldots, T$ **do**
  # Inner loop is parallelized in practice, unroll for better readability.
  for $i = 1, \ldots, m$ **do**
    Set context $\boldsymbol{c}_i = \boldsymbol{x} \oplus \boldsymbol{y}_{<i}$.
    Compute logits from the LM as $\boldsymbol{s}^{(i)} = f_\theta(\boldsymbol{c}_i) \in \mathbb{R}^{|\mathcal{V}|}$. Take softmax and get $\pi_\theta^{(i)}$.
    Top $n\sigma$-filter (Tang et al., 2024): Compute $s_{\max}^{(i)} = \max_k \boldsymbol{s}^{(i)}$, $\sigma = \text{std}(\boldsymbol{s}^{(i)})$. Define filtered logit $\tilde{s}_k^{(i)} = -\infty$ if $s_k^{(i)} \leq s_{\max}^{(i)} - n\sigma$ else $\tilde{s}_k^{(i)} = s_k^{(i)}$.
    Take softmax on filtered $\tilde{\boldsymbol{s}}$ and get filtered $\pi_{\text{flt}}^{(i)}$.
    Compute target $\pi_{\text{tar}}^{(i)}$ based on Eq (3).
    Compute loss

$$\ell_{\text{OVERTONE}}^{(i)} = \max(\text{D}_{\text{KL}}[\pi_{\text{tar}}^{(i)} \| \pi_\theta^{(i)}], \epsilon).$$

  end for
  Compute sample loss

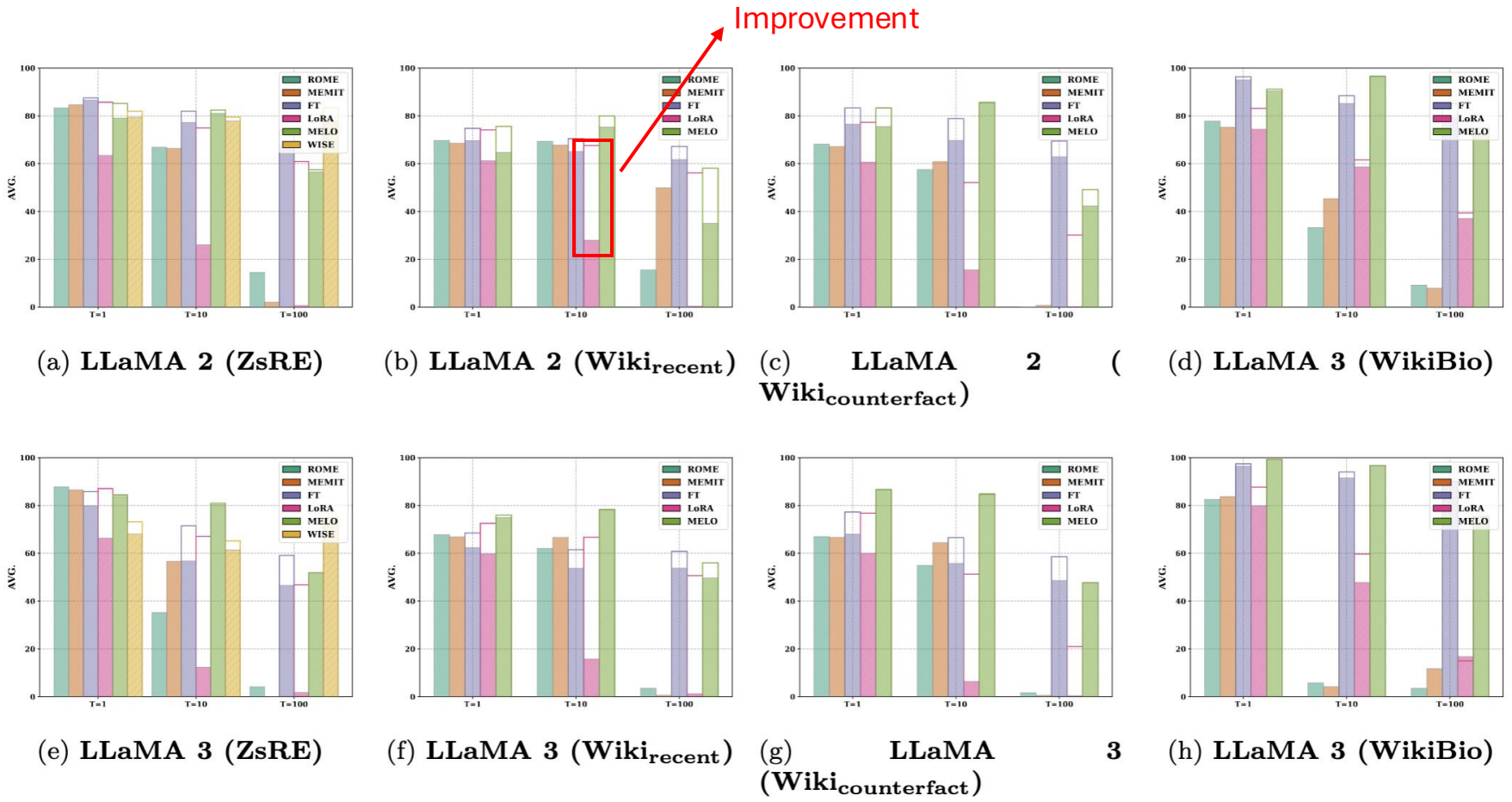
$$\ell_{\text{OVERTONE}}(\theta) = \sum_{i=1}^{m} \ell_{\text{OVERTONE}}^{(i)}.$$

  Update with learning rate α

$$\theta \leftarrow \theta - \alpha \nabla_\theta \ell_{\text{OVERTONE}}(\theta)$$

end for
output Edited parameter θ.

OVERTONE [Arxiv 2025]



(a) **LLaMA 2 (ZsRE)** (b) **LLaMA 2 ($\text{Wiki}_{\text{recent}}$)** (c) **LLaMA 2 ($\text{Wiki}_{\text{counterfact}}$)** (d) **LLaMA 3 (WikiBio)**

(e) **LLaMA 3 (ZsRE)** (f) **LLaMA 3 ($\text{Wiki}_{\text{recent}}$)** (g) **LLaMA 3 ($\text{Wiki}_{\text{counterfact}}$)** (h) **LLaMA 3 (WikiBio)**

Figure 3: Continual Editing performance under different sequence length T. Solid and transparent bars show performance with and without OVERTONE. Unfilled area marks the performance gap. ROME and MEMIT didn't use OVERTONE.

Outline

- Model Efficiency
  - Parameter-efficient Fine-tuning
  - Model Compression
  - Inference Acceleration
- Data Efficiency
  - Zero/few-shot Learning
  - Knowledge Editing
  - Retrieval-augmented Generation

Failure Cases of Pre-trained Large Language Models

List 5 most cited papers authored by **Geoffrey Hinton**

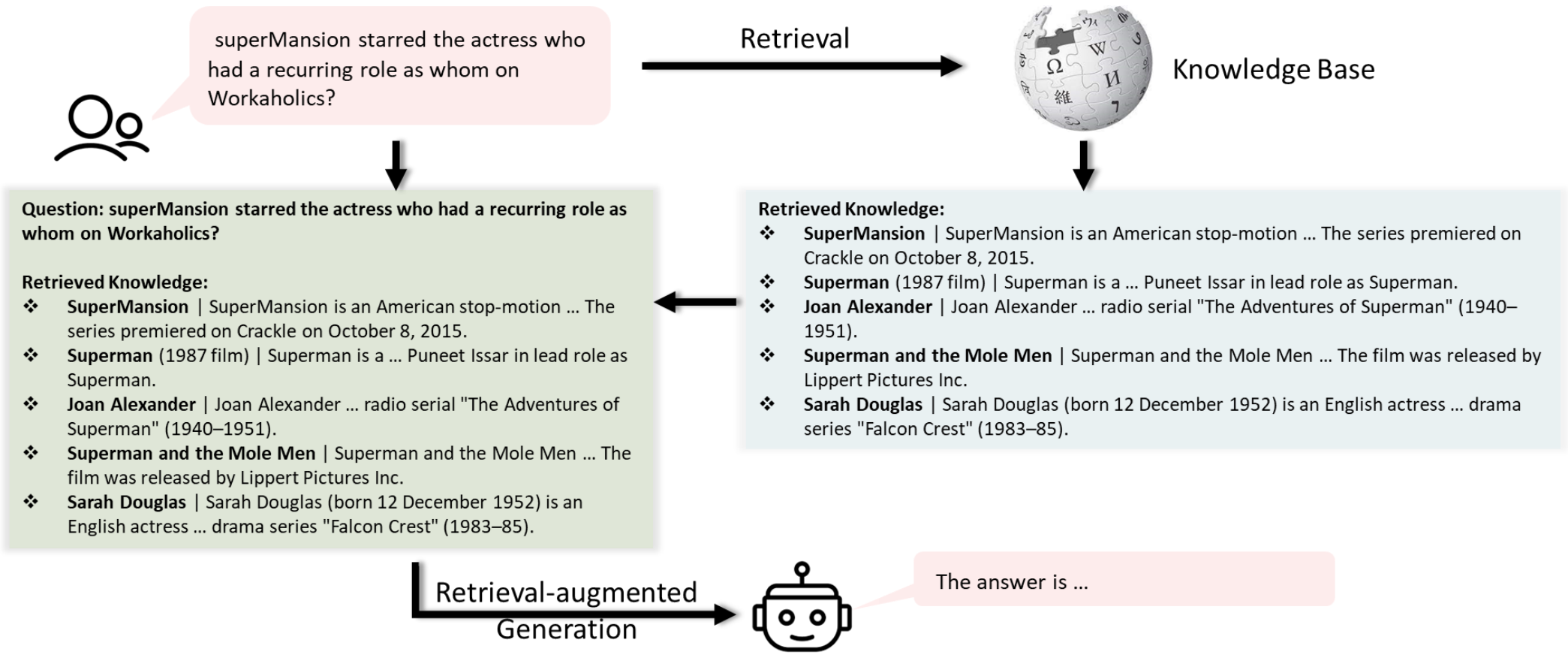
1."**Learning Representations by Back-Propagating Errors**" … ✓
2."**A Fast Learning Algorithm for Deep Belief Nets**" … ✗
3."**Reducing the Dimensionality of Data with Neural Networks**" … ✗
4."**Dropout: A Simple Way to Prevent Neural Networks from Overfitting**"… ✓
5."**Rectified Linear Units Improve Restricted Boltzmann Machines**" … ✗

which team is Luka Dončić in?

Luka Dončić is a professional basketball player for the Dallas Mavericks in the NBA.

- LLMs can't memorize all (long-tail) knowledge in their parameters [Akari A et al., 2023].
- LLMs' knowledge is easily outdated and hard to update frequently [Akari A et al., 2023].

Retrieve-then-generate

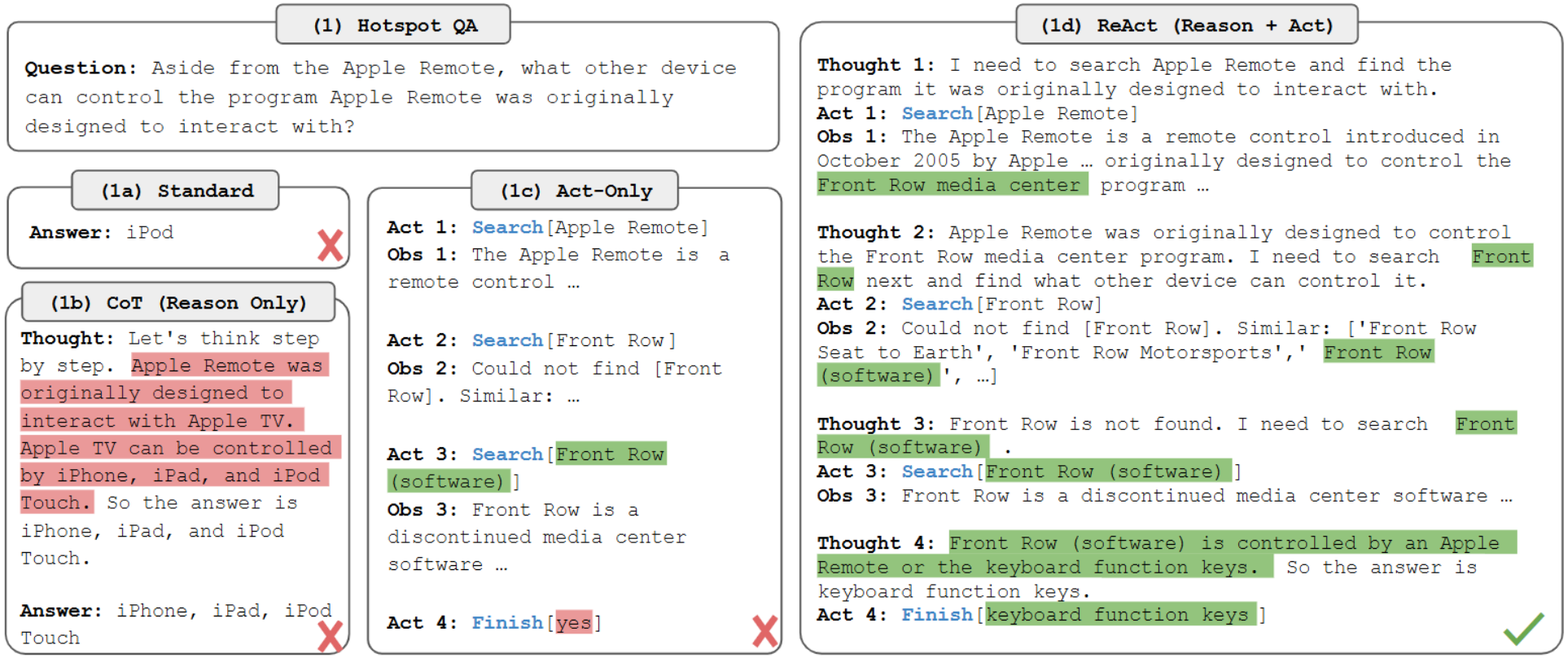


Challenge

- Complex queries make it difficult to retrieve documents accurately.
- Irrelevant information will mislead the LLM.

ReAct

- Ask the LLM to propose the question needing to solve (Thought), the action (Search/Finish), and intermediate answer (Obs).
- The LLM will decompose the input query into several sub-questions and take actions dynamically.



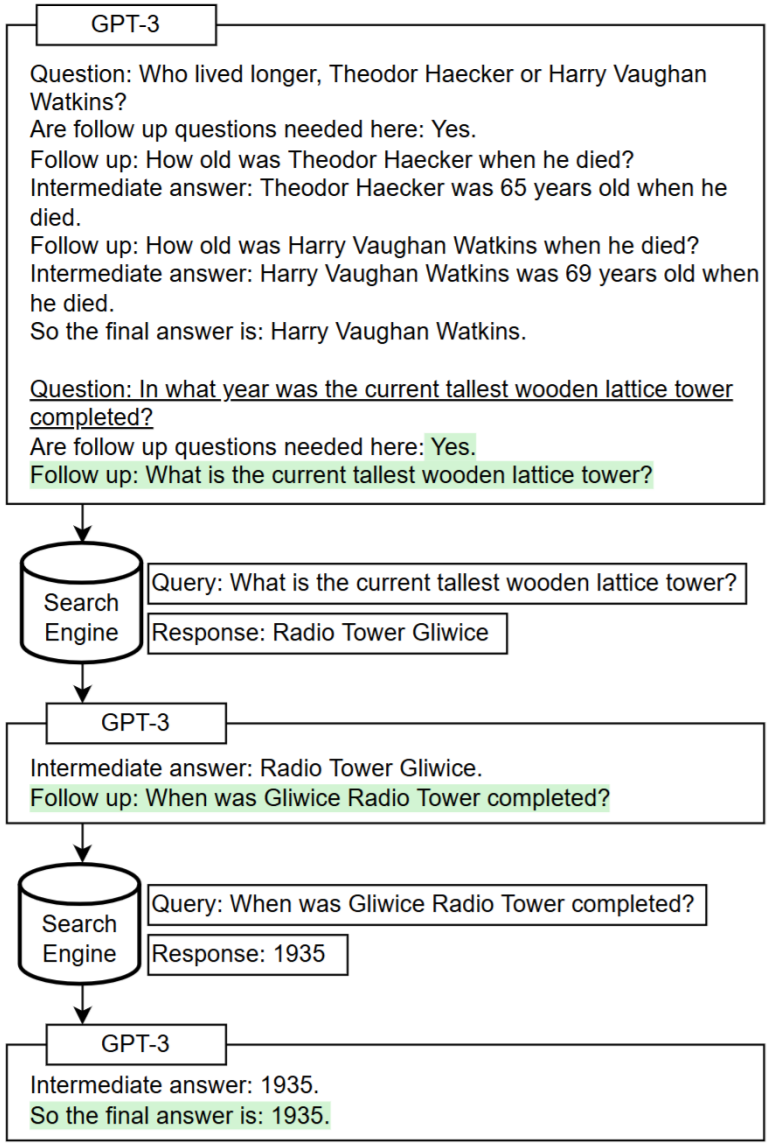
Yao, Shunyu, et al. "React: Synergizing reasoning and acting in language models." *International Conference on Learning Representations (ICLR)*. 2023.

ReAct

Prompt Method[a]	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	**64.6**
ReAct → CoT-SC	**35.1**	62.0
Supervised SoTA[b]	67.5	89.5

Self-ask

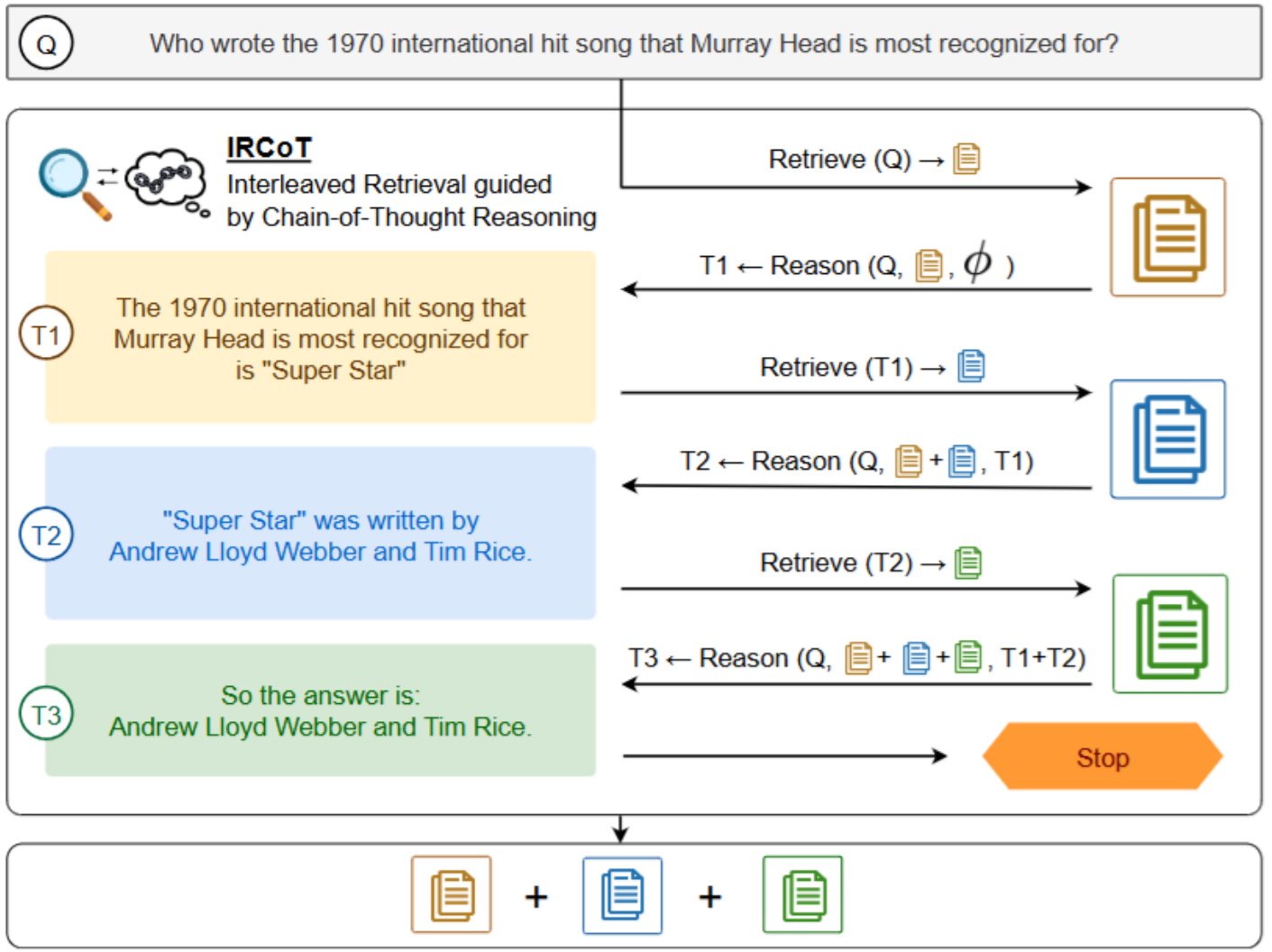
- Share the similar spirit with ReAct.
- Decompose original question into follow-up questions and combine retrieval.



Press, Ofir, et al. "Measuring and Narrowing the Compositionality Gap in Language Models." Findings of the Association for Computational Linguistics: EMNLP 2023.

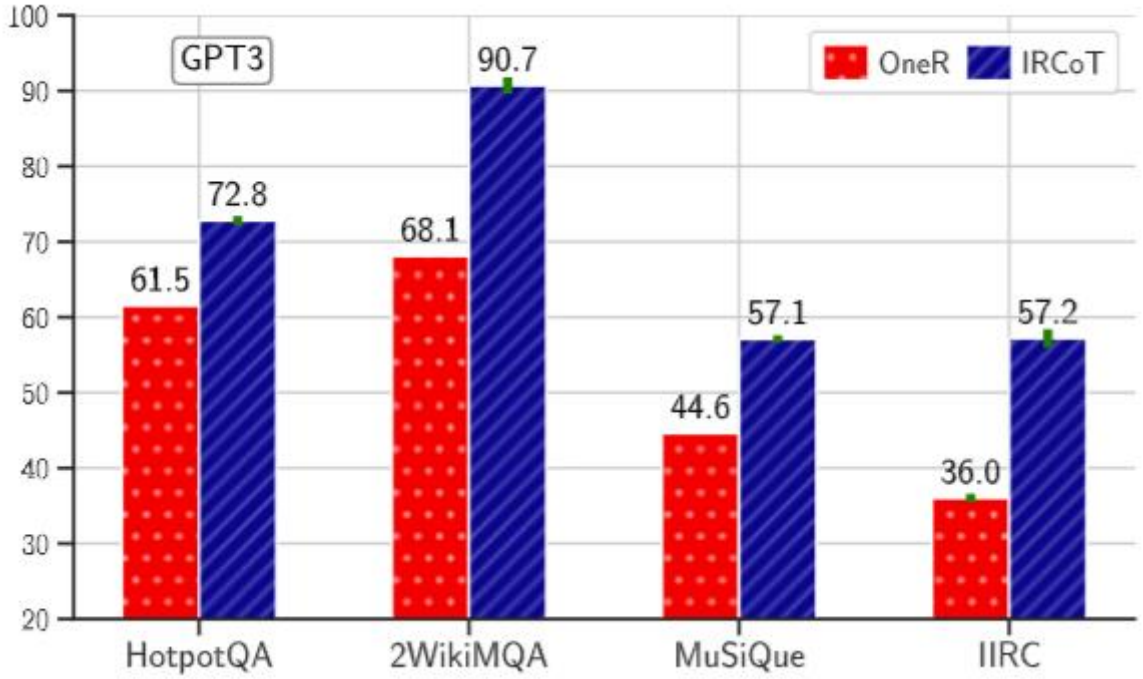
IRCoT

- What to retrieve depends on what has already been derived, which in turn may depend on what was previously retrieved

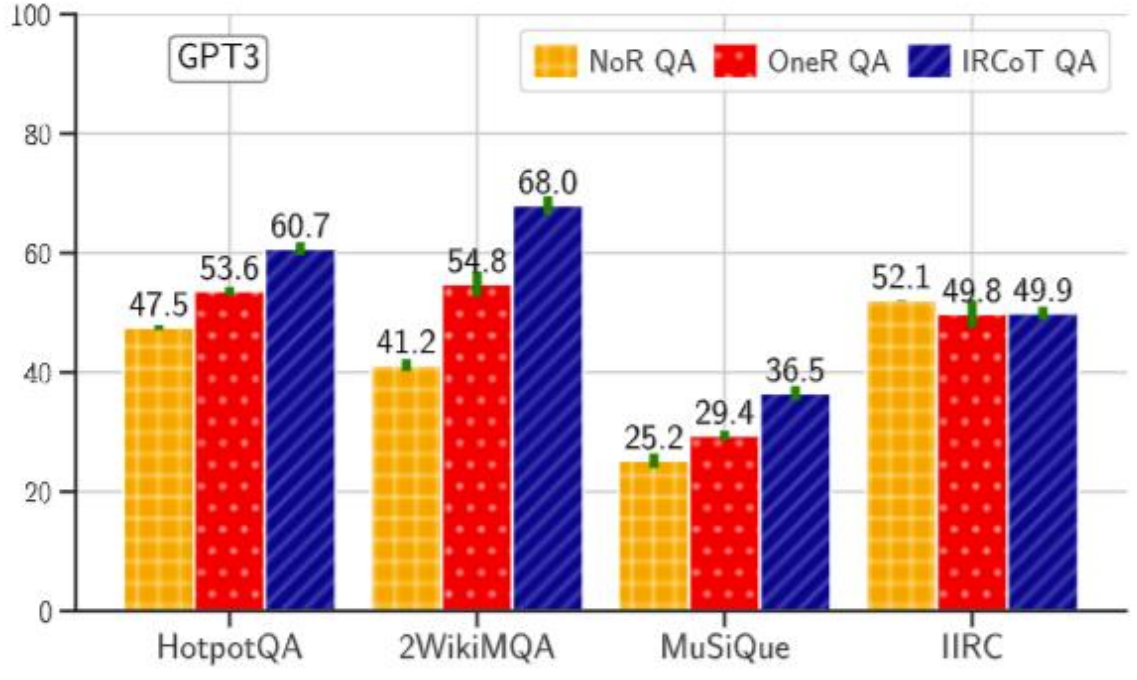


Trivedi, Harsh, et al. "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions." Proceedings of ACL. 2023.

IRCoT



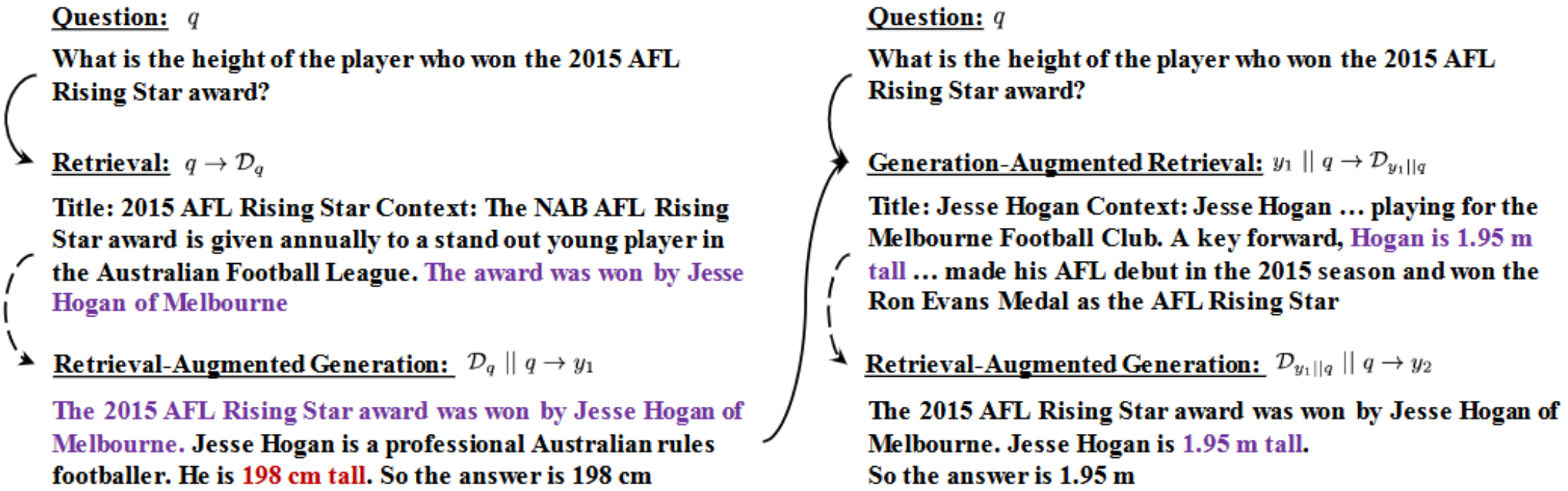
Retrieval recall



Answer F1

ITER-RETGEN

- Iterate retrieval and generation
- leverage the model output from the previous iteration as a specific context to help retrieve more relevant knowledge



Shao, Zhihong, et al. "Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy." Findings of EMNLP 2023.

ITER-RETGEN

Method	HotPotQA			2WikiMultiHopQA			MuSiQue			Bamboogle			Feverous		StrategyQA	
	EM	F1	Acc†	EM	F1	Acc†	EM	F1	Acc†	EM	F1	Acc†	Acc	Acc†	Acc	Acc†
Without Retrieval																
Direct	21.9	36.8	44.8	21.3	29.2	33.9	7.0	18.7	15.8	11.2	24.4	28.0	60.1	60.1	66.5	66.7
CoT	30.0	44.1	50.0	30.0	39.6	44.0	19.4	30.9	28.6	**43.2**	**51.1**	60.0	59.8	59.8	71.0	71.0
With Retrieval																
Direct	31.6	44.7	53.3	27.3	35.4	43.6	13.9	28.2	26.5	17.6	31.8	43.2	69.8	69.8	65.6	65.6
ReAct	24.9	44.7	61.1	28.0	38.5	45.9	23.4	37.0	37.9	21.8	31.0	40.3	66.4	66.4	66.9	66.9
Self-Ask	36.8	55.2	64.8	**37.3**	**48.8**	55.9	**27.6**	41.5	**42.9**	31.5	41.2	54.8	70.7	70.7	70.2	70.2
DSP	43.8	55.0	60.8	-	-	-	-	-	-	-	-	-	-	-	-	-
ITER-RETGEN 1	39.2	53.9	65.5	33.7	45.2	55.4	24.2	38.6	38.1	36.8	47.7	57.6	67.0	67.0	72.0	72.0
ITER-RETGEN 2	44.1	58.6	71.2	34.9	47.0	58.1	26.4	41.1	41.0	38.4	48.7	59.2	68.8	68.8	73.0	73.0
ITER-RETGEN 3	45.2	59.9	71.4	34.8	47.8	58.3	25.7	41.4	40.8	37.6	47.0	59.2	69.0	69.0	72.3	72.3
ITER-RETGEN 4	45.8	**61.1**	**73.4**	36.0	47.4	58.5	26.7	41.8	40.8	38.4	49.6	60.0	**71.5**	**71.5**	73.8	73.8
ITER-RETGEN 5	45.2	60.3	72.8	35.5	47.5	58.8	25.7	40.7	39.6	39.2	49.7	**60.8**	70.3	70.3	73.2	73.2
ITER-RETGEN 6	**45.9**	61.0	73.3	35.5	48.1	**59.4**	25.9	40.5	39.8	40.0	50.0	59.2	70.9	70.9	72.4	72.4
ITER-RETGEN 7	45.1	60.4	72.9	35.5	47.4	58.4	26.1	**42.0**	41.0	40.0	50.7	**60.8**	70.5	70.5	**74.1**	**74.1**

QA Performance

Iteration	1	2	3	4	5	6	7
HotPotQA	49.5	66.1	65.7	66.5	66.7	66.7	67.1
2WikiMultiHopQA	29.0	45.2	46.2	46.7	45.8	45.8	46.5
MuSiQue	18.6	32.3	32.3	33.7	32.7	33.5	32.9
Bamboogle	20.8	36.0	36.8	36.0	35.2	36.0	36.0

Retrieval Recall

Challenge

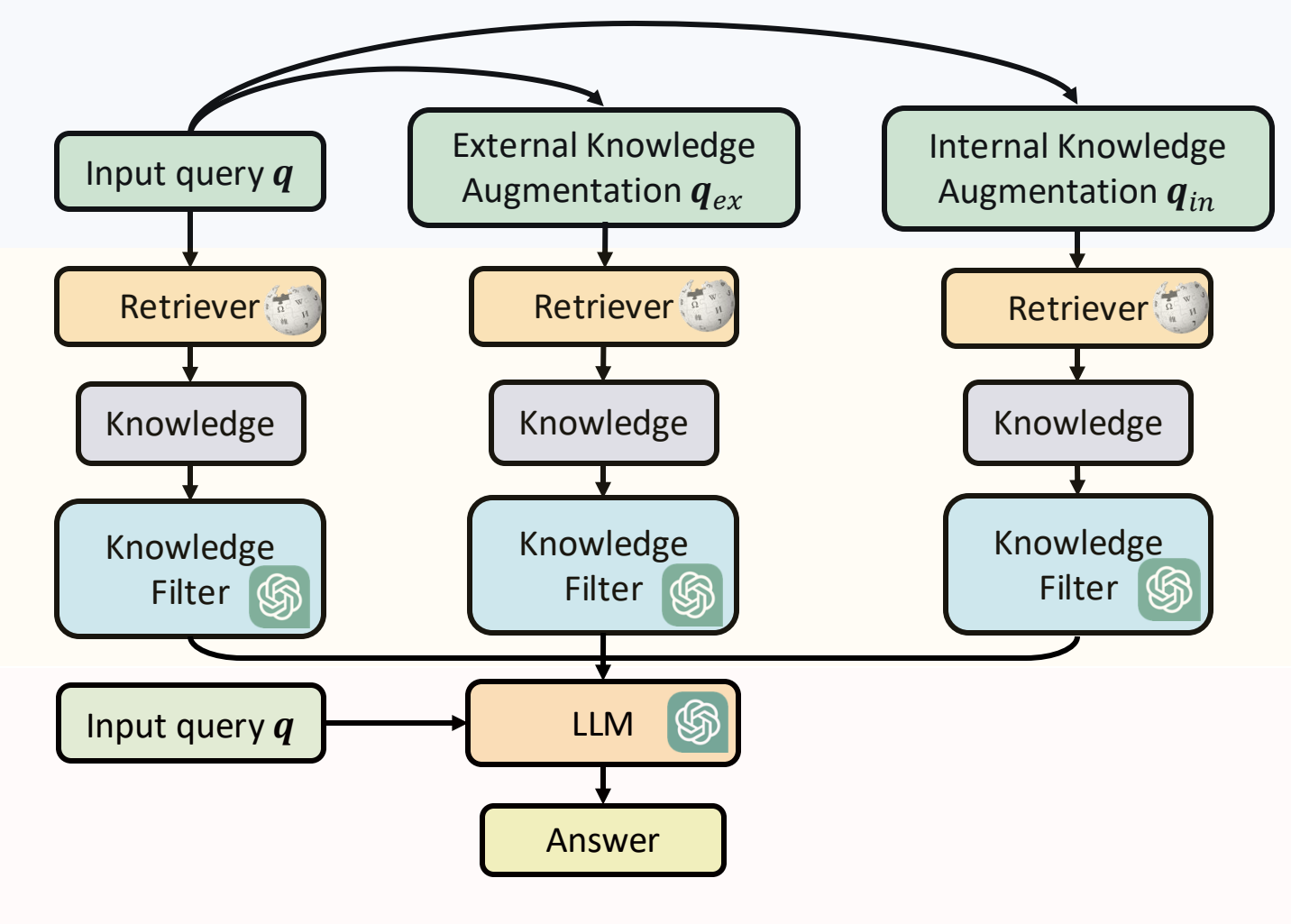
- Complex queries make it difficult to retrieve documents accurately.
- Irrelevant information will mislead the LLM

RetRobust

- In-context learning method:
  - Leverage an extra natural language inference (NLI) model
  - NLI model assesses whether retrieved context is irrelevant
  - If irrelevant, the model will generate without using the retrieved information
- Fine-tuning method:
  - Leverage an LLM to generate training data that mixes both relevant and irrelevant contexts
  - Standard Supervised Fine-tuning on the generated data

Yoran, Ori, et al. "Making Retrieval-Augmented Language Models Robust to Irrelevant Context." The Twelfth ICLR.

BlendFilter



Query Blending

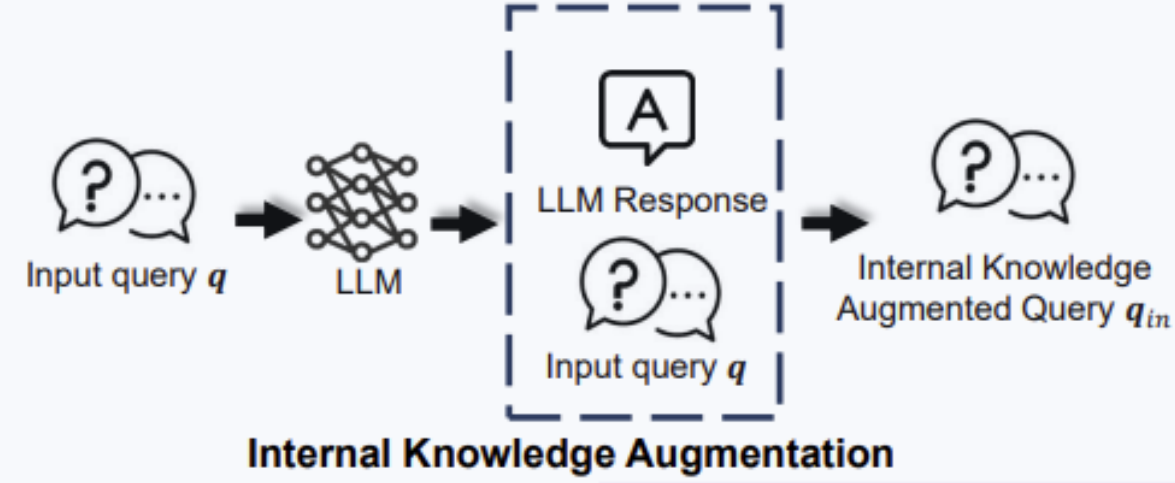
Knowledge Filtering

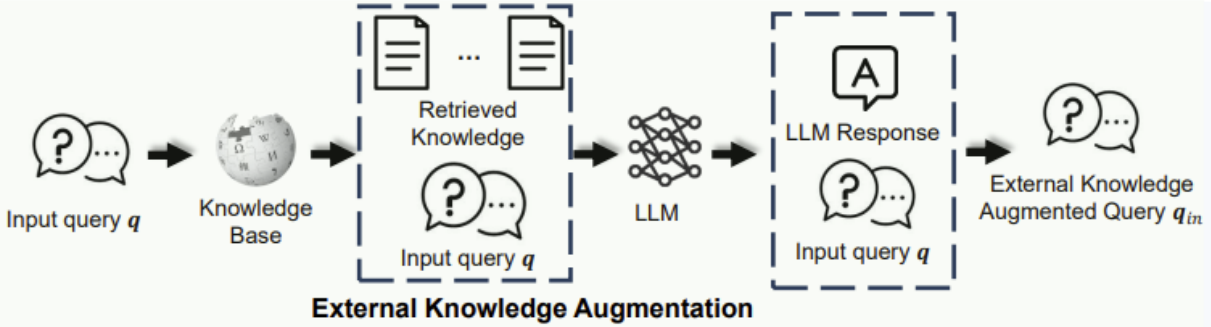
Answer Generation

Wang, Haoyu, et al. "BlendFilter: Advancing Retrieval-Augmented Large Language Models via Query Generation Blending and Knowledge Filtering." Proceedings of EMNLP 2024.

BlendFilter

Query Blending



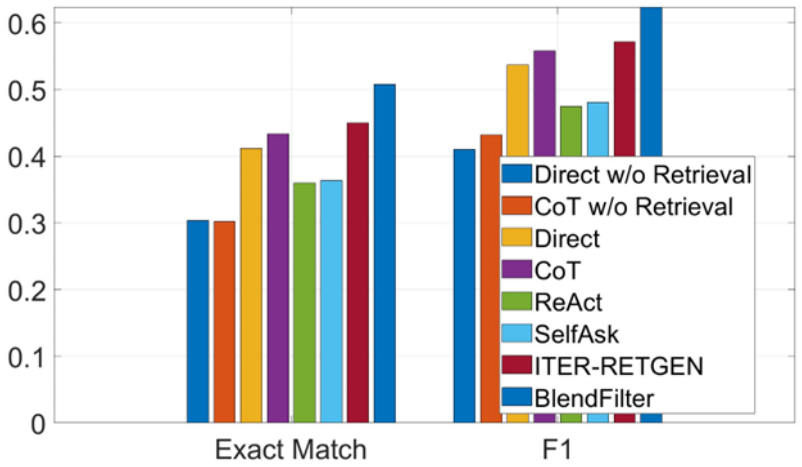


BlendFilter

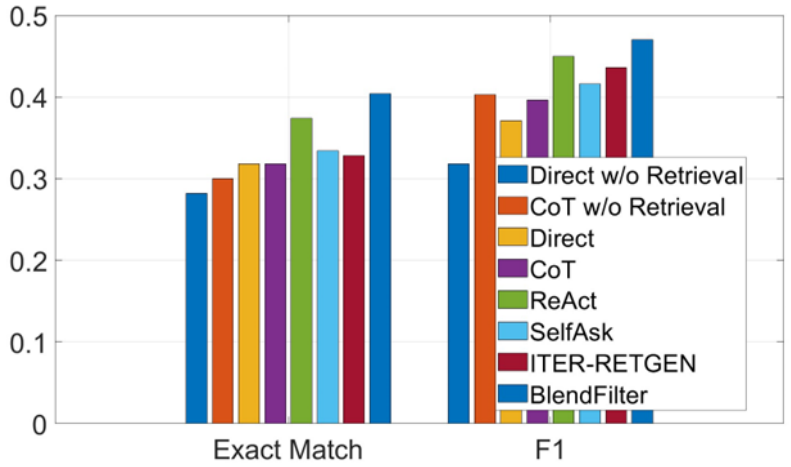
Knowledge Filtering

```
Forget your knowledge about {topic}. Please only consider the
knowledge below.
knowledge 0 : {Retrieved_knowledge0}
knowledge 1 : {Retrieved_knowledge1}
knowledge 2 : {Retrieved_knowledge2}
knowledge 3 : {Retrieved_knowledge3}
knowledge 4 : {Retrieved_knowledge4}
Please check the relevance between {question} and knowledges
0-4 one by one, remove the irrelevant ones and show me the
relevant ones. There may be multiple relevant ones. Please
take a deep breath and do it step by step.
```

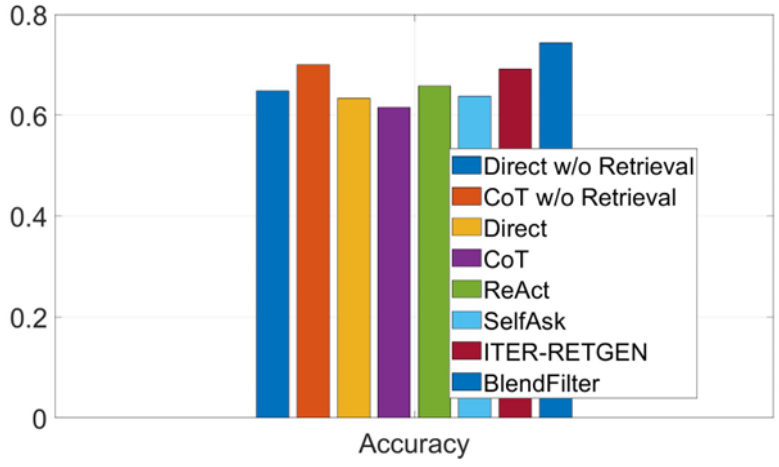
BlendFilter



HotPotQA



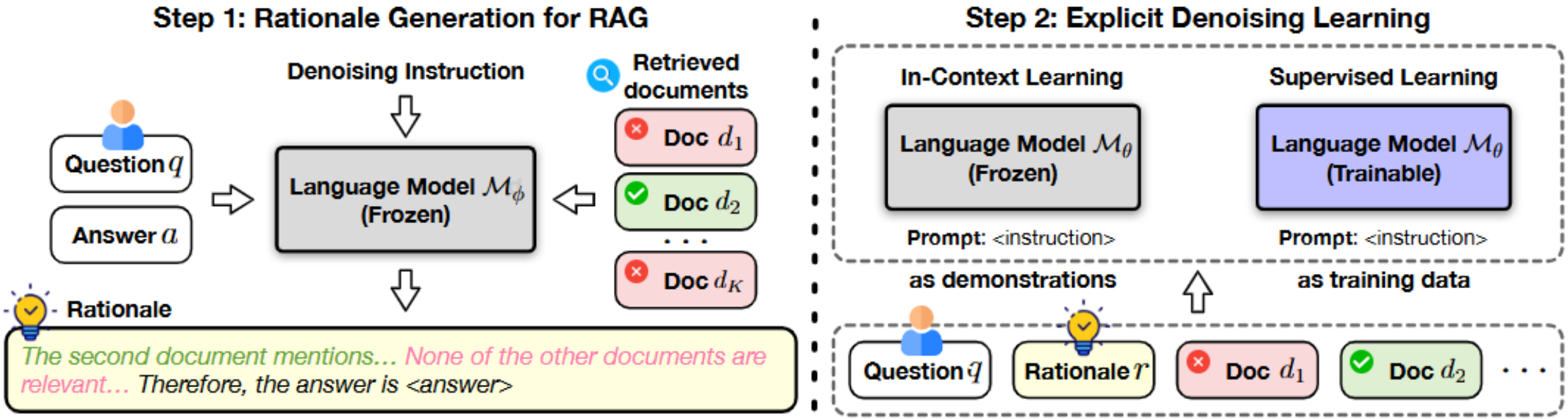
2WikiMultihopQA



StrategyQA

InstructRAG

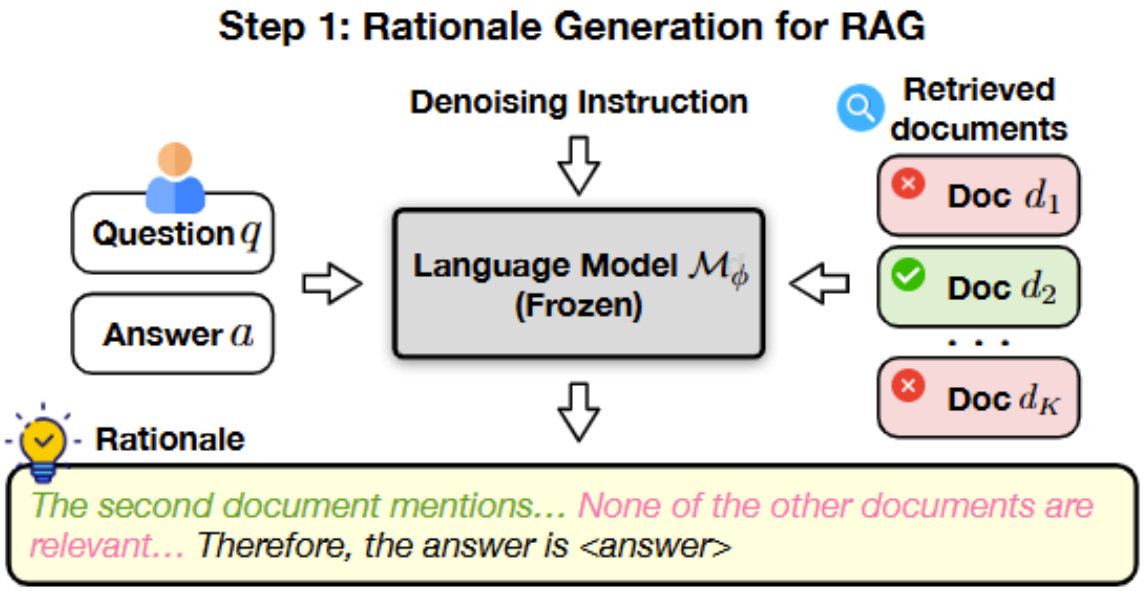
- Enable the LLM to generate rationale for the retrieved documents
- Conduct SFT on the generated rationale



Wei, Zhepei, Wei-Lin Chen, and Yu Meng. "InstructRAG: Instructing Retrieval-Augmented Generation via Self-Synthesized Rationales." ICLR 2025

InstructRAG

Rationale Generation via Instruction-Following



Rationale Generation

Input: Read the following documents relevant to the given question: $\{q_i\}$

Document [1] (Title: $\cdots$): {contents of d_i^1}
$\cdots$
Please identify documents that are useful to answer the given question: $\{q_i\}$, and explain how the contents lead to the answer: $\{a_i\}$

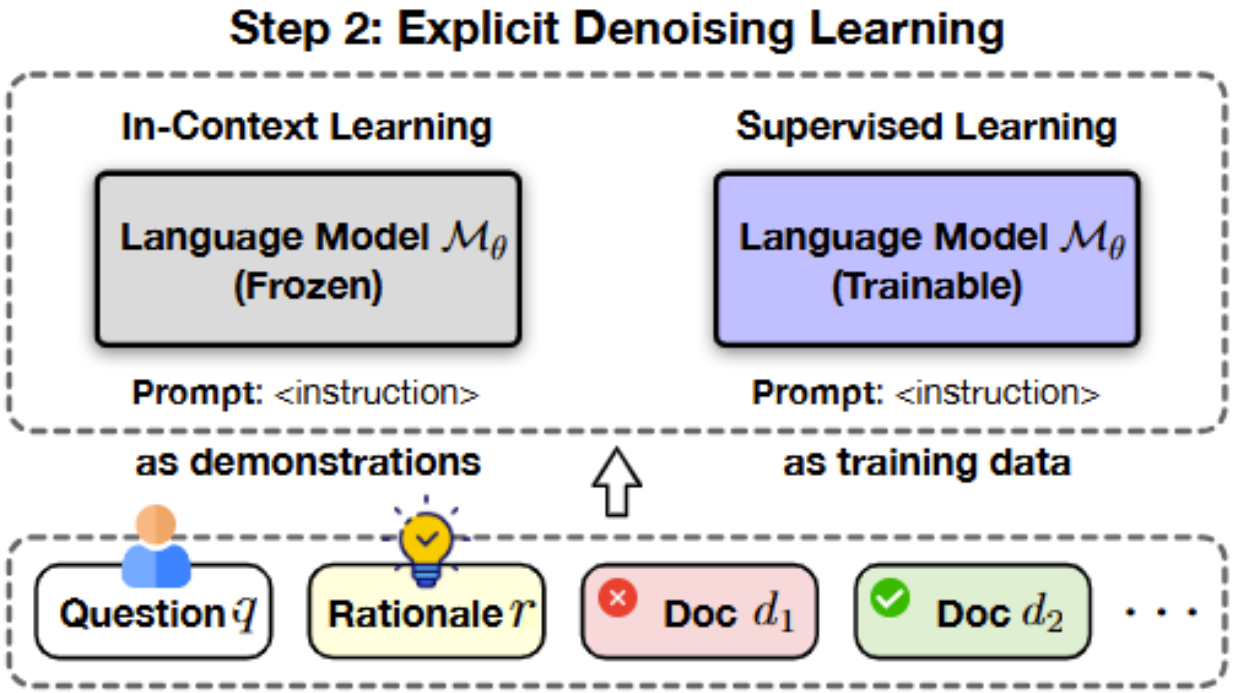
{task-specific instruction}

Output: {rationale r_i}

InstructRAG

- Learning Denoising Rationales in RAG

$$\max_{\theta} \mathbb{E}_{q,r\sim\mathcal{T}^+} \log p_\theta\,(r|q,D)$$



InstructRAG

Method	PopQA (acc)	TriviaQA (acc)	NQ (acc)	MultiHopQA (acc)	ASQA (em)	ASQA (pre)	ASQA (rec)
Baselines w/o Retrieval							
Vanilla Zero-shot Prompting							
ChatGPT*	29.3	74.3	–	–	35.3	–	–
Llama-3-Instruct$_{8B}$	22.8	69.4	46.6	45.6	30.6	–	–
Llama-3-Instruct$_{70B}$	28.9	80.6	57.9	57.5	39.1	–	–
RAG w/o Training							
In-Context RALM [70]							
ChatGPT*	50.8	65.7	–	–	40.7	65.1	76.6
Llama-3-Instruct$_{8B}$	62.3	71.4	56.8	43.4	40.0	62.1	66.4
Llama-3-Instruct$_{70B}$	63.8	76.3	60.2	51.2	43.1	62.9	67.6
Few-Shot Demo. w/ Instruction							
Llama-3-Instruct$_{8B}$	63.1	74.2	60.1	45.3	42.6	55.0	64.4
Llama-3-Instruct$_{70B}$	63.9	79.1	62.9	53.9	45.4	49.3	57.1
INSTRUCTRAG-ICL							
Llama-3-Instruct$_{8B}$	64.2	76.8	62.1	50.4	44.7	**70.9**	**74.1**
Llama-3-Instruct$_{70B}$	**65.5**	**81.2**	**66.5**	**57.3**	**47.8**	69.1	71.2
RAG w/ Training							
Vanilla Supervised Fine-tuning							
Llama-3-Instruct$_{8B}$	61.0	73.9	56.6	56.1	43.8	–	–
Self-RAG [3]							
Llama-2$_{7B}$	55.8	68.9	42.4	35.9	30.0	66.9	67.8
Llama-2$_{13B}$	56.3	70.4	46.4	36.0	31.4	**70.3**	**71.3**
Llama-3-Instruct$_{8B}$	55.8	71.4	42.8	32.9	36.9	69.7	69.7
RetRobust [105]							
Llama-2$_{13B}$	–	–	39.6	51.5	–	–	–
Llama-3-Instruct$_{8B}$	56.5	71.5	54.2	54.7	40.5	–	–
INSTRUCTRAG-FT							
Llama-3-Instruct$_{8B}$	**66.2**	**78.5**	**65.7**	**57.2**	**47.6**	65.7	70.5

ROSERAG

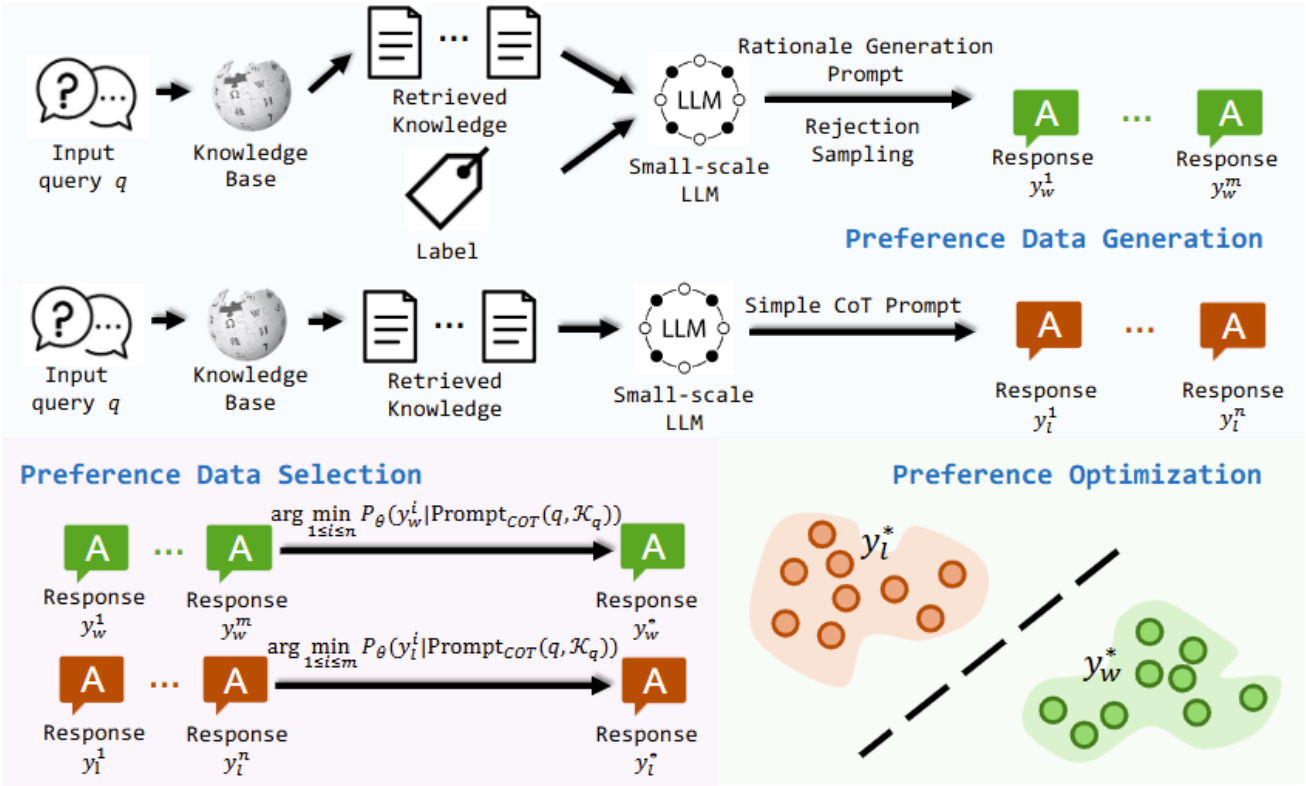
Motivation

- Small-scale LLMs (SLMs) have limited reasoning and generalization capacity.
- SFT tends to mimic the behavior present in the training data, making it highly sensitive to noise and prone to overfitting.

Liu, Tianci, et al. "RoseRAG: Robust Retrieval-augmented Generation with Small-scale LLMs via Margin-aware Preference Optimization." arXiv preprint arXiv:2502.10993 (2025).

RoseRAG

- Three stages:
  - Preference Data Generation
  - Preference Data Selection
  - Preference Optimization



RoseRAG

Preference Data Generation

- Preferred response y_w
  - Ask the SLM to generate reasoning process given retrieved knowledge, the question, and the ground-truth answer
  - Rejection sampling
- Non-preferred response y_l
  - Generate from vanilla CoT response

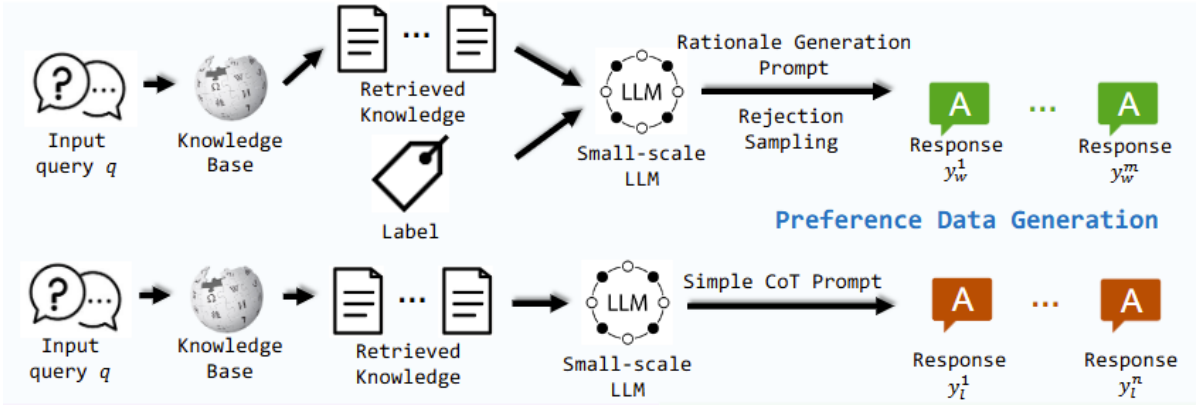
Rationale Generation

System Prompt: You are a useful assistant. I will provide one question, several pieces of knowledge (which may be related or unrelated to the question), and the answer to the question. Please explain your reasoning process in a single paragraph consisting of no more than four sentences. If the provided knowledge is insufficient, you may make an informed guess, but do not respond with "Unknown".

User Prompt: Knowledge: $\mathcal{K}_q$
Question: q
Answer: a^*

Assistant Prompt: Let's think step by step.

Output: {rationale r}



RoseRAG

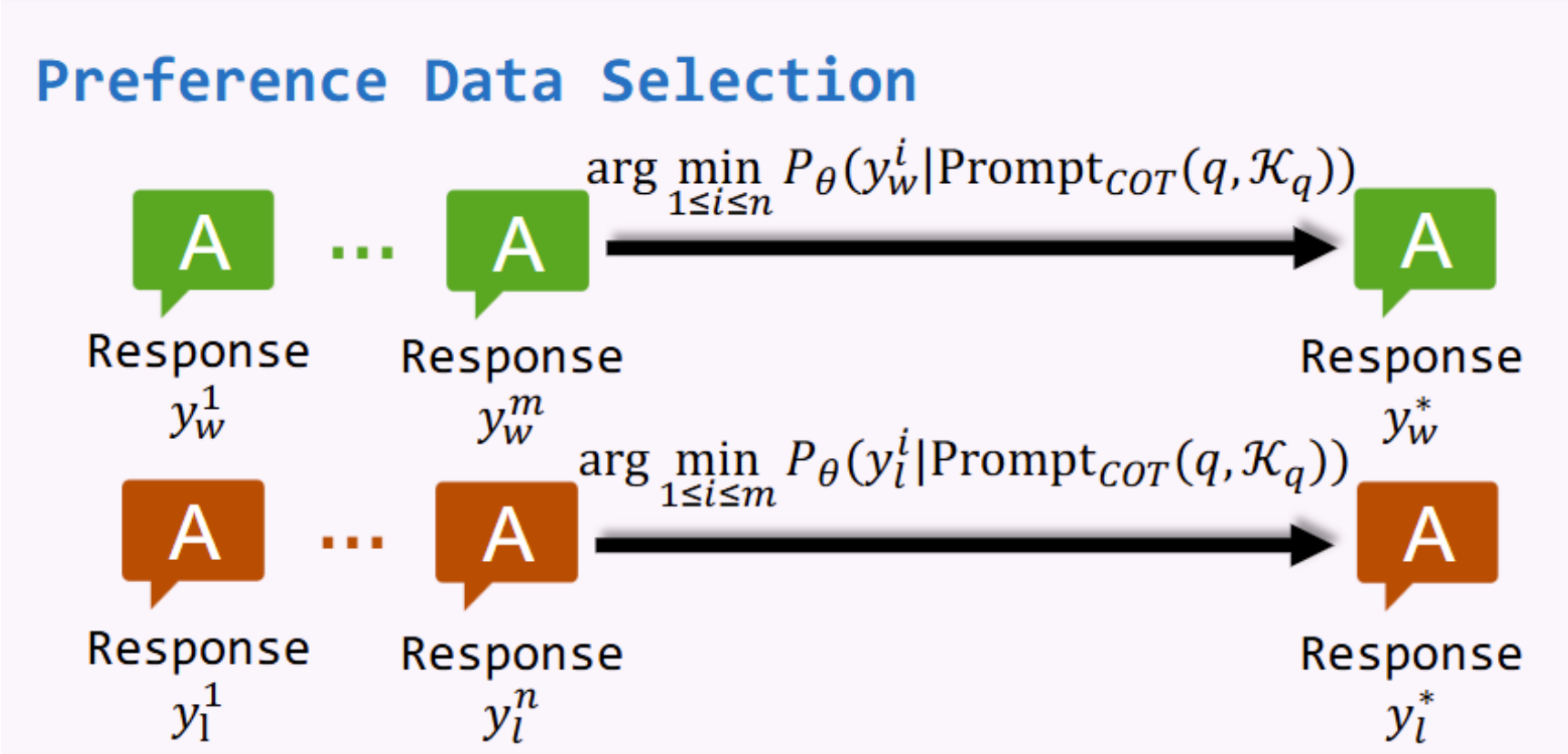
Preference Data Selection

- Least confident preferred data

$$y_w^* = \arg\min_{1 \le i \le n} P_\theta\left(y_w^i \middle| \text{Prompt}_{\text{CoT}}(q, \mathcal{K}_q)\right)$$

- Most confident preferred data

$$y_l^* = \arg\min_{1 \le i \le m} P_\theta\left(y_l^i \middle| \text{Prompt}_{\text{CoT}}(q, \mathcal{K}_q)\right)$$

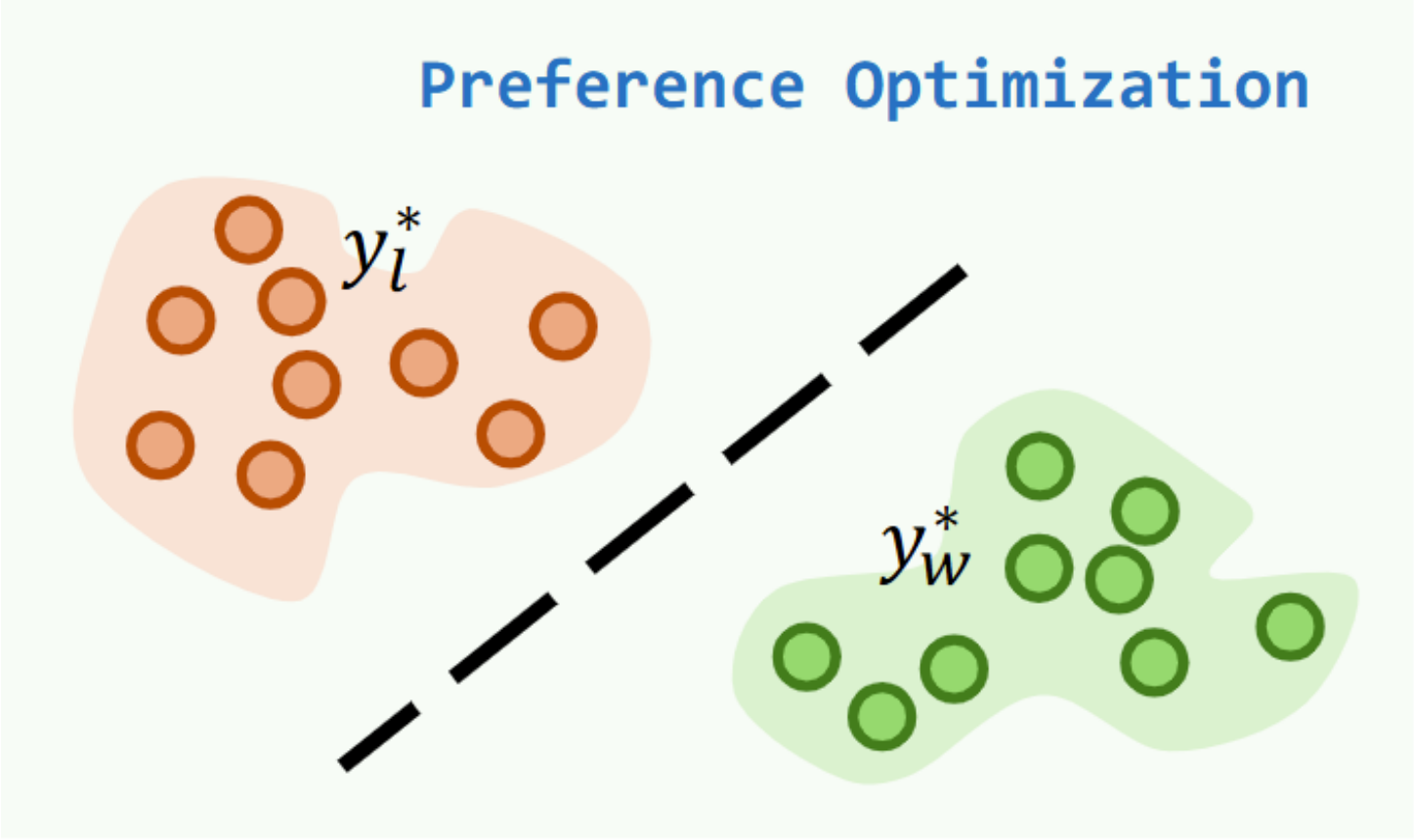


RoseRAG

Preference Optimization

- Use preference optimization (e.g., ORPO, PPO) to enable the SLM to align and distinguish the preference data

$$\min_{\theta} \mathbb{E}_{(x,y_w^*,y_l^*)} \ell_{ORPO}(x, y_w^*, y_l^*)$$



RoseRAG

Method	HotPotQA		2WikiMultiHopQA		StrategyQA	
	EM	F1	EM	F1	Acc	F1
CoT	13.0	20.9	18.8	23.5	55.9	17.9
vanilla RAG	27.2	38.4	6.2	11.4	55.5	23.9
ReAct	14.9	25.9	9.6	22.2	55.0	48.8
SelfAsk	20.4	32.8	19.2	25.3	51.1	42.9
BlendFilter	26.4	37.5	19.4	24.2	<u>59.4</u>	<u>45.0</u>
InstructRAG	<u>31.2</u>	<u>39.6</u>	<u>23.6</u>	<u>26.9</u>	53.7	39.1
RetRobust	16.8	24.6	13.0	19.9	51.5	37.3
ASTUTE	21.4	27.8	19.0	23.4	55.0	12.0
ICL+RAG	28.6	39.3	21.8	26.4	56.3	27.5
RoseRAG	**34.8**	**44.8**	**31.6**	**35.0**	**59.8**	**52.1**

		Positive		
		w/o Selection	Minimal	Maximal
Negative	w/o Selection	30.6/39.9	33.2/43.1	27.6/37.1
	Minimal	29.0/39.2	33.2/42.2	26.6/35.3
	Maximal	33.0/43.6	**34.8/44.8**	28.8/38.4

Future Directions

- Model compression and pruning
  - Pruning, quantization, knowledge distillation, low-rank adaptation.
- Efficient training techniques
  - Alternative optimization that reduces time complexity, sparse and adaptive training, federated learning.
- Data efficiency
  - Reduce the need for large-scale data by transfer learning, synthetic data generation, self-supervised/weakly supervised learning.

Future directions: How to enable LLM to think efficiently

- **Problem**
  - LLMs often rely on long, multi-step reasoning (Chain-of-Thought), which can be time-consuming and computationally expensive.
- **Key questions**
  - When to Think Long vs. Short?
  - How can the model decide between a brief inference path or a more detailed reasoning chain?

Future directions: Data & model efficiency in AI agents

- **Problem**
  - Long-horizon, complex tasks make it challenging to rely solely on Supervised Fine-Tuning.
  - High computational overhead from LLMs when there exist many sequential steps.
- **Key questions**
  - How to Combine RL + Simulation?
  - Can simulation-based RL reduce real-world data needs and improve adaptability?

Future directions: AI on edge devices

Problem

- Edge devices have strict constraints on computation, memory, and energy, making it difficult to deploy large AI models.
- Real-time or near-real-time inference demands low latency and high efficiency.

Key questions

- How to handle on-device learning and ensure real-time response?
- Can we adapt or update models locally without relying on the cloud, and how do we maintain privacy?
- Can we ensure a holistic optimization of various resource constraints (memory, latency, energy consumption, etc.)?

Future directions: Hardware–software co-design

- **Problem**
  - Traditional hardware is not always optimized for the unique demands of large-scale AI.
  - Mismatches between model architectures and hardware capabilities lead to inefficiencies and wasted energy.
- **Key questions**
  - How to match AI algorithms to hardware?
  - What co-optimization strategies can we use?