Deep Learning for Computer Vision

113-1/Fall 2024

https://cool.ntu.edu.tw/courses/41702 (NTU COOL)

http://vllab.ee.ntu.edu.tw/dlcv.html (Public website)

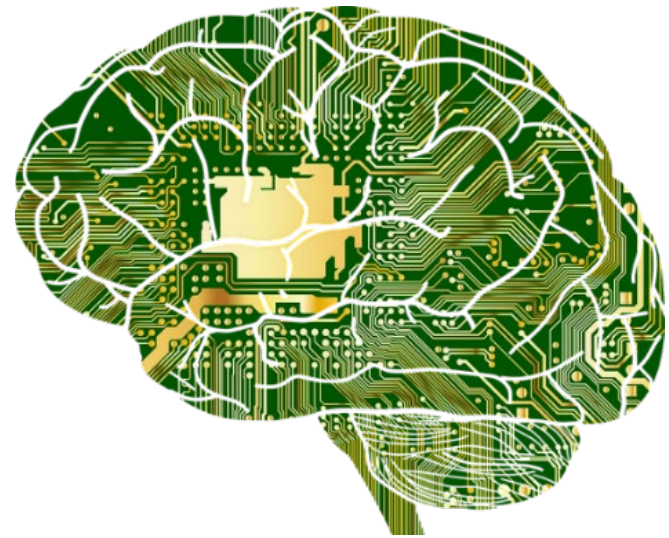
Yu-Chiang Frank Wang 王鈺強, Professor

Dept. Electrical Engineering, National Taiwan University

2024/10/29

What to Be Covered?

- **Learning Vision & Language Models**
  - **Pretraining**
  - **Finetuning,**
    In-Context Learning &
    Retrieval-Augmented Generation
  - **Parameter-Efficient Fine-Tuning**
- ~~**Advanced Topics**~~
  - ~~**Concept Editing**~~
  - ~~**Concept Unlearning**~~
- **Experience Sharing** (30 min.)
  - Grad Study & AI opportunities in France
- **HW #3 is out!**



Pretrain & Finetune LLM/VLM/MLLM

Stage 1

Pre-training by
self-supervised learning
or supervised learning

Stage 2

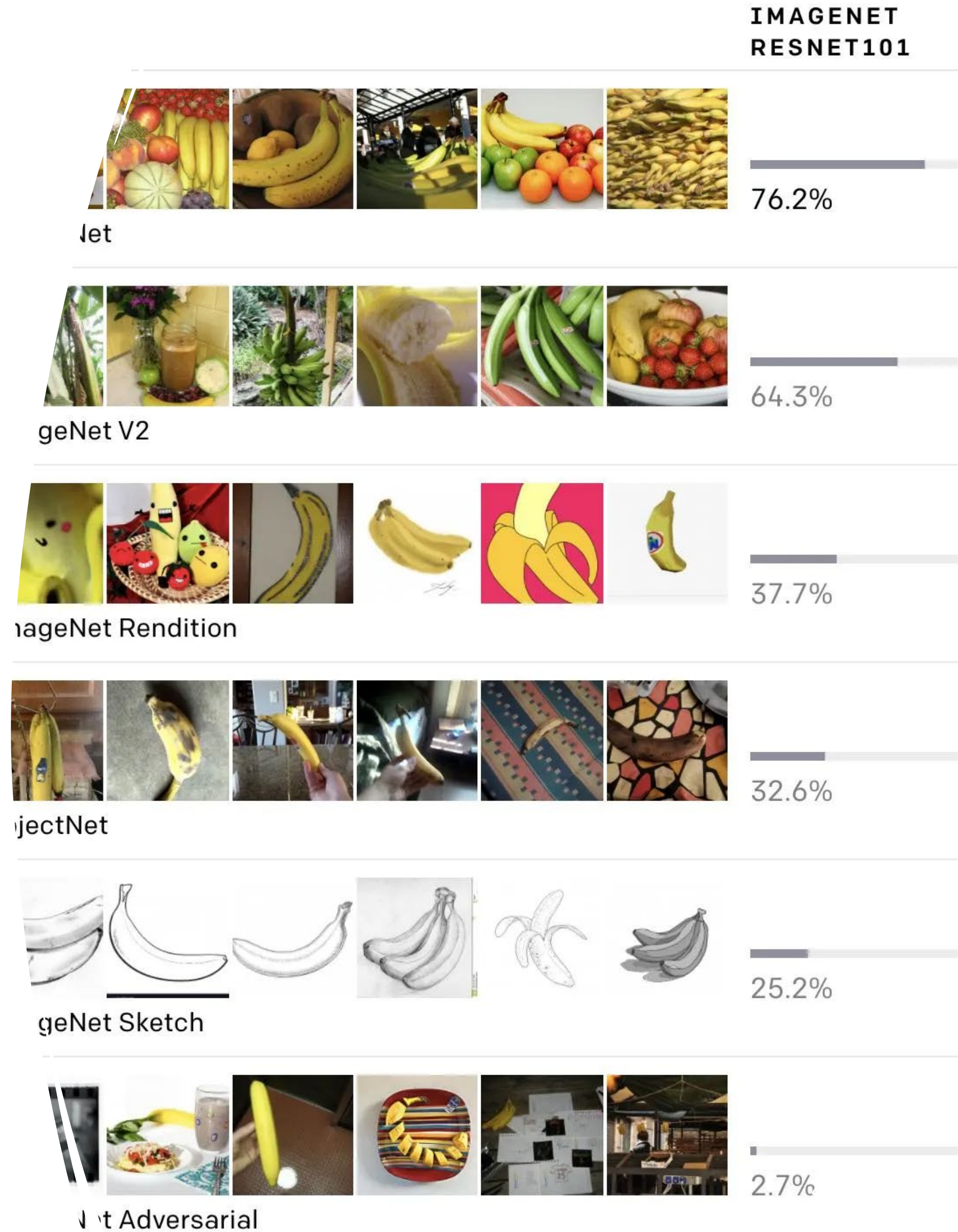
Finetuning
by downstream tasks
in target domains

Stage 3

RLHF - Reinforcement Learning
with Human Feedback
(will not cover)

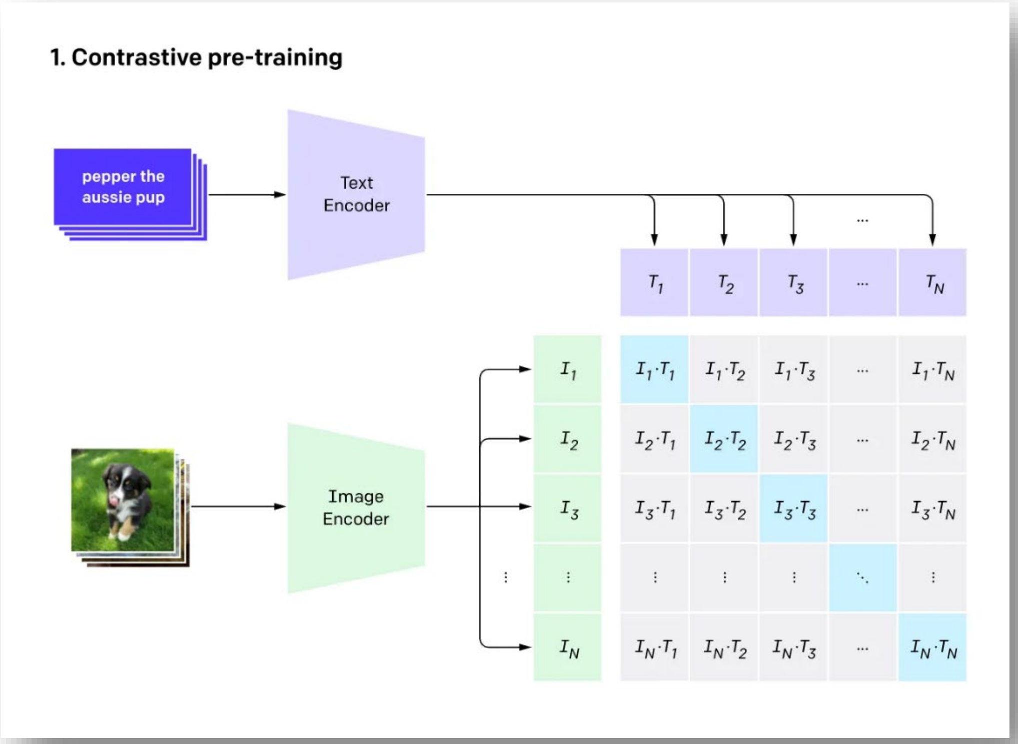
Recap: CLIP - Contrastive Language-Image Pretraining

- OpenAI, *Learning Transferable Visual Models From Natural Language Supervision*, NeurIPS WS 2021 (w/ 9000+ citations)
- Why DL/CNN not good enough?
  - Require annotated data for training image classification
  - Domain gap between closed-world and open-world domain data
  - Lack of ability for zero-shot classification



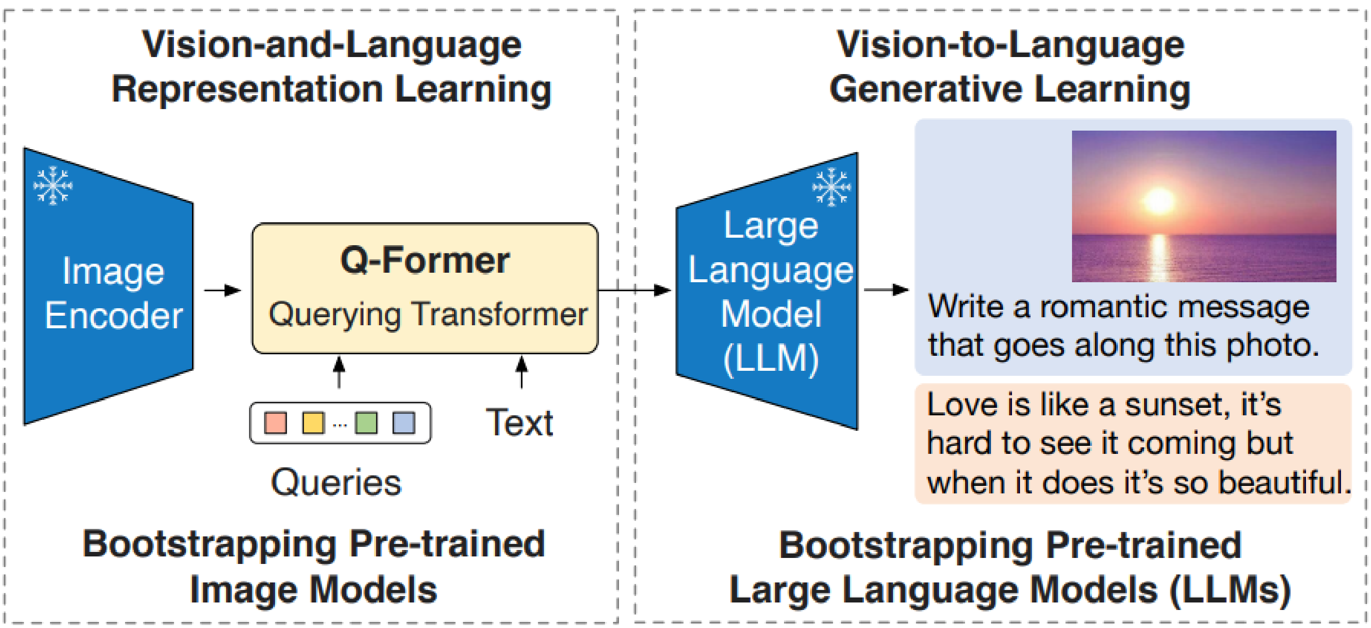
CLIP (cont'd)

- Why DL/CNN not good enough?
  - Require annotated data for training image classification
  - Domain gap between closed-world and open-world domain data
  - Lack of ability for zero-shot classification
- Motivation/Objectives
  - Cross-domain contrastive learning from large-scale image-language data



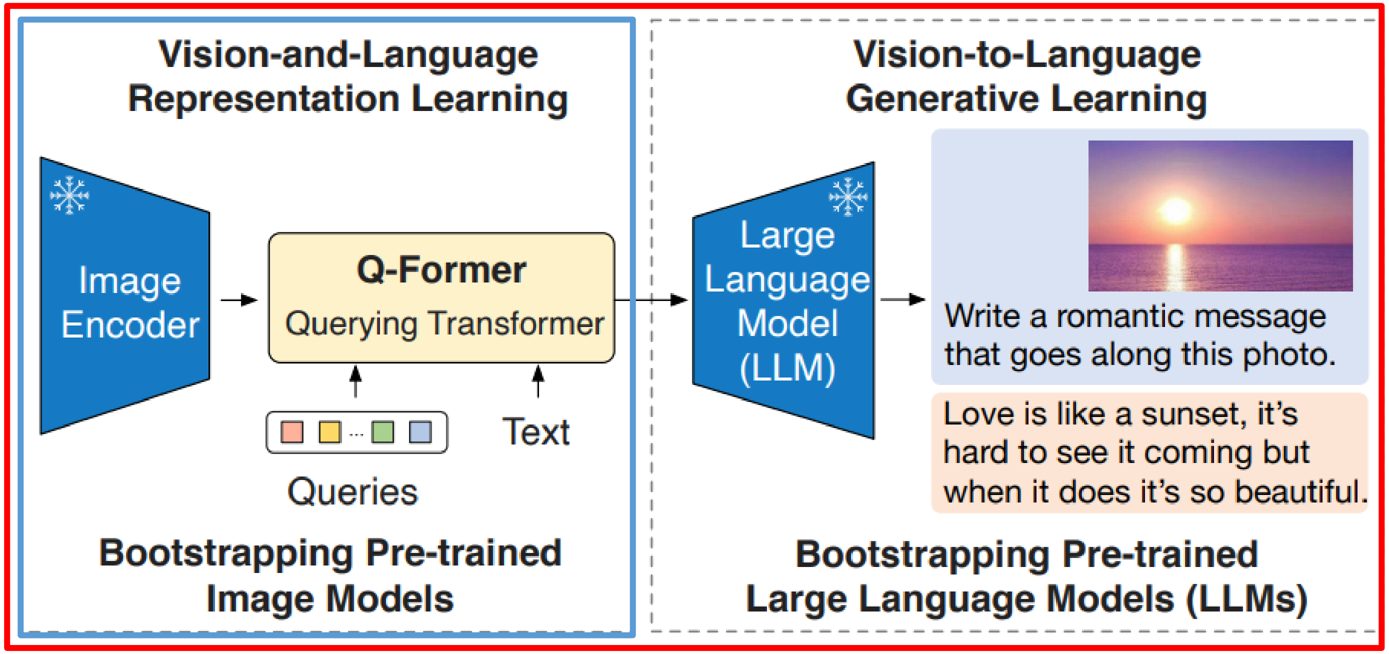
Recap: BLIP-2 (ICML'23)

- **BLIP:**
  Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation, Salesforce Research, NeurIPS 2021
- **Goal:**
  Bridge the modality gap by a lightweight Querying Transformer (Q-Former) with a **frozen pre-trained image encoder** and a **frozen large language model**.



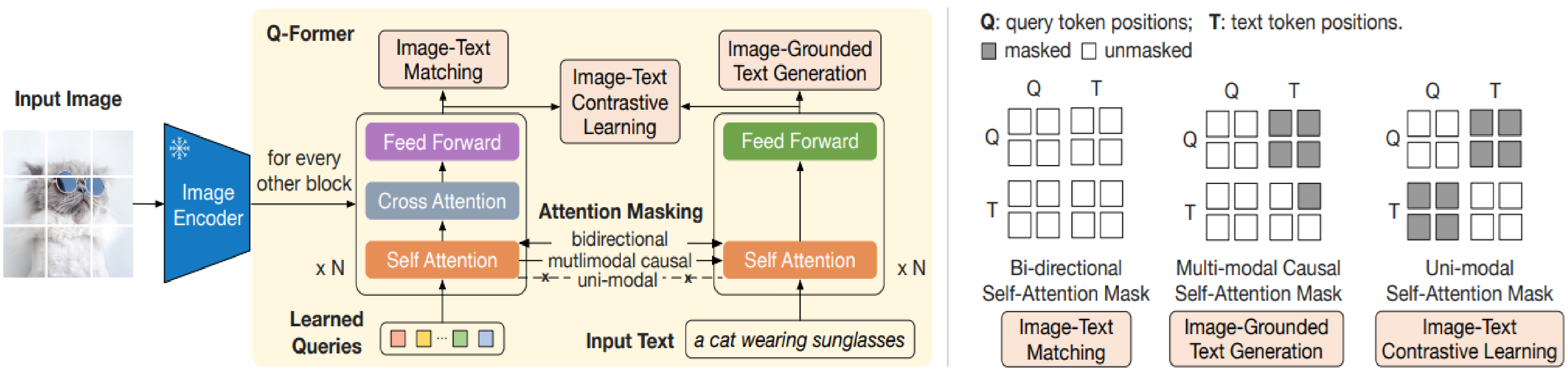
Pre-training of BLIP2

- A **two-stage** pre-training strategy
  - Stage 1: Representation Learning
    - enforce **Q-Former** to learn **visual representation** that is most relevant to the text description
  - Stage 2: Generative Learning
    - make the output representation of **Q-Former** to be understood by **LLM**



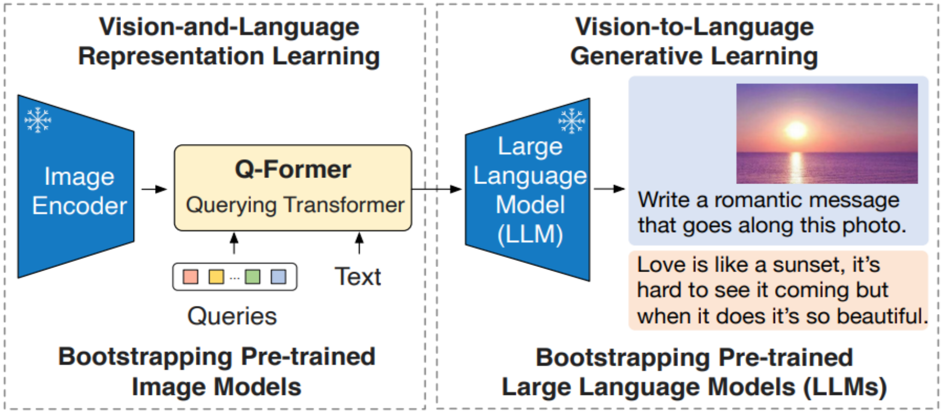
Pre-training Stage 1 - VL Representation Learning

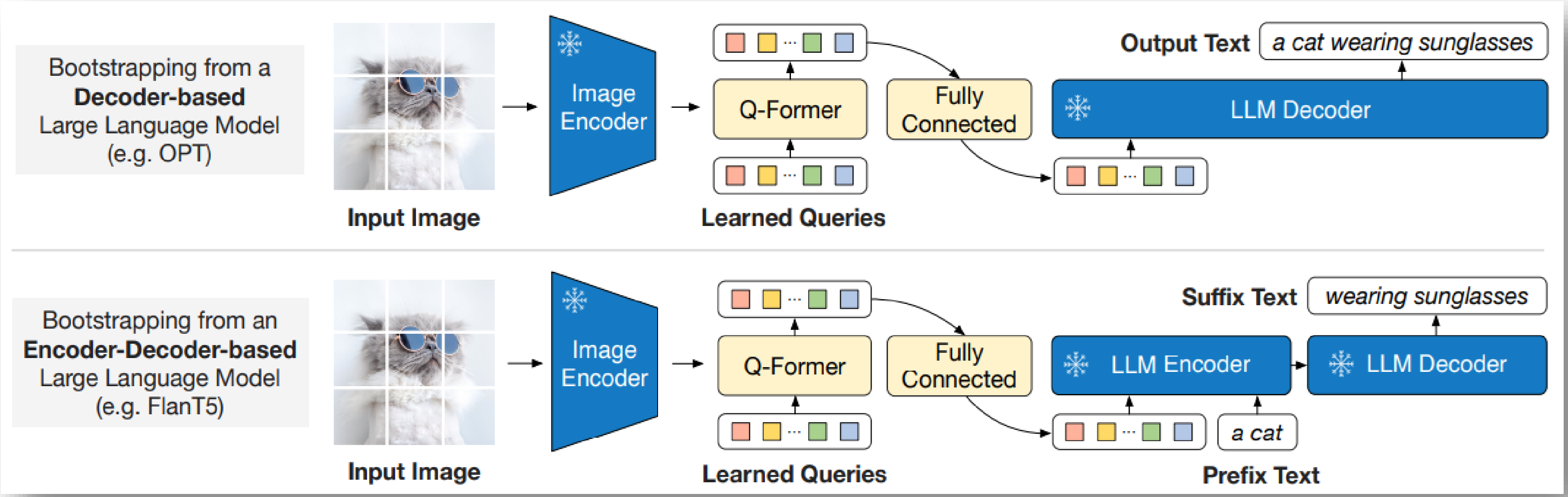
- **Goal:** enforce **Q-Former** to extract visual representation relevant to text
- **Method:** three pre-training tasks
  - **Image-Text Contrastive Learning (ITC)**:
    self-attn in Q/T, followed by max (sim(Q, T)) -> can be viewed as CLIP training
  - **Image-Text Matching (ITM)**:
    for each learnable query -> linear classifier for binary decision
  - **Image-grounded Text Generation (ITG)**:
    self-attn in Q for encoder training; T->Q for image-to-text generation



Pre-training Stage 2 - VL Generative Learning

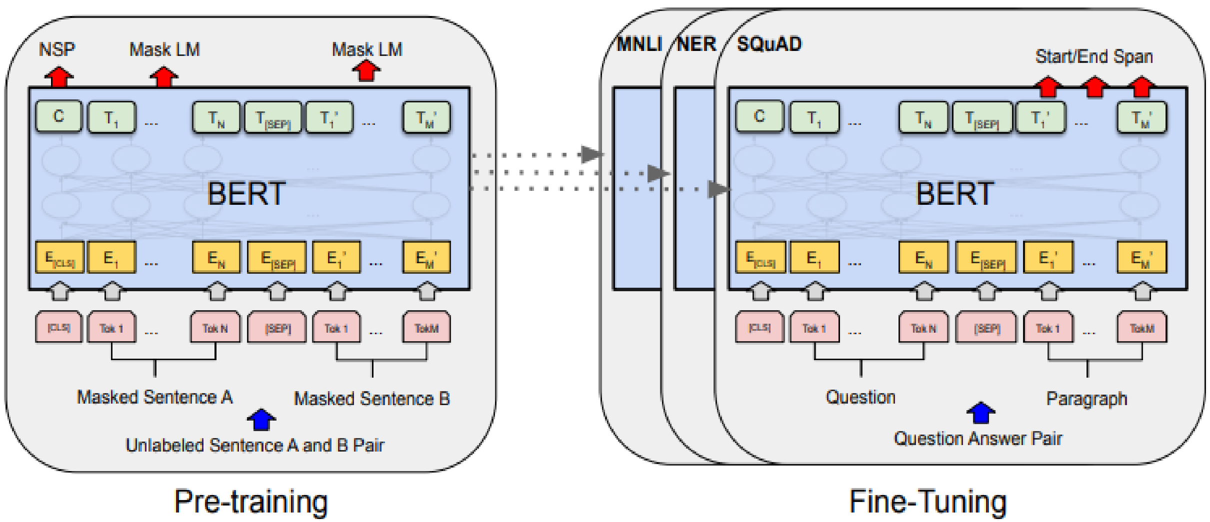
- **Goal:**
  Learning with LLM guidance
  i.e., make the output representation of **Q-Former** to be understood by **LLM**
- **Method:**
  pre-training with
  Image-grounded Text Generation (ITG)





Take BERT as an example

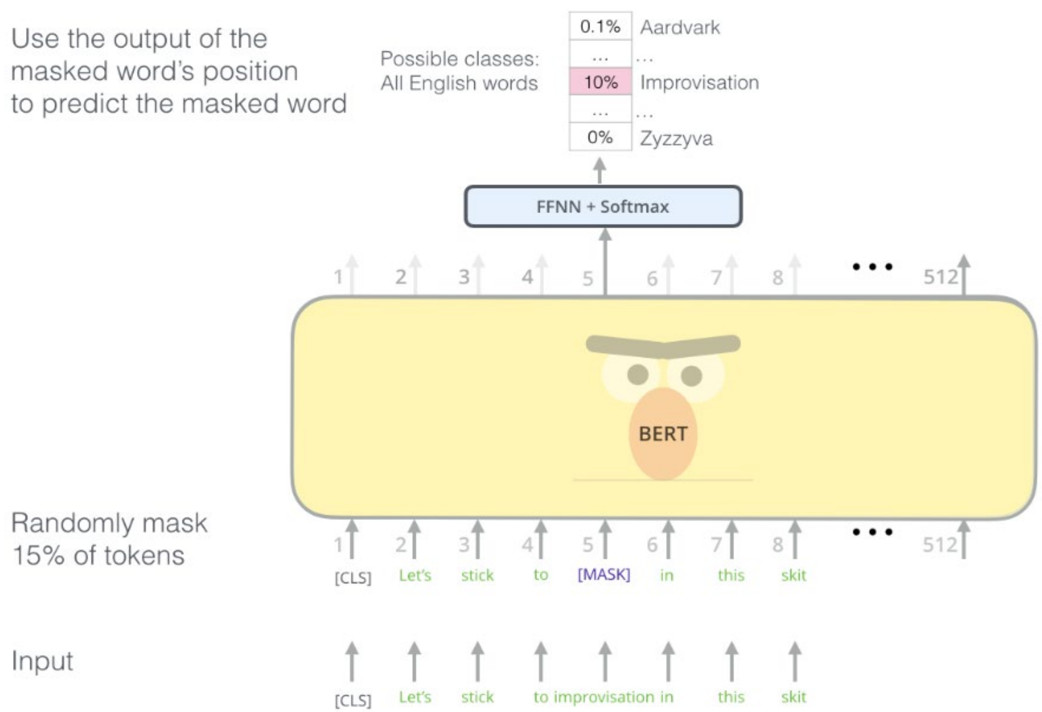
- BERT, short for <u>B</u>idirectional <u>E</u>ncoder <u>R</u>epresentations from <u>T</u>ransformers



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Google, arxiv 2018

Take BERT as an example (cont'd)

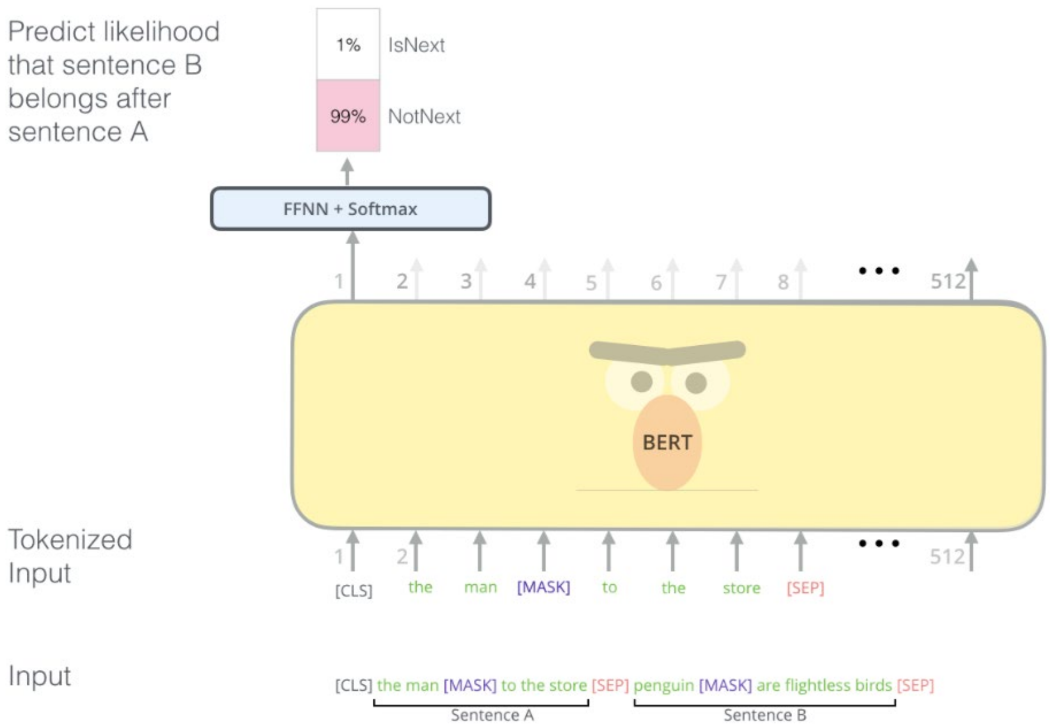
- Pre-training strategy #1:
  Masked Token Prediction (or Masked Language Modeling)
    - Mask out 15% of the input text and predict the masked outputs



BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

Take BERT as an example (cont'd)

- Pre-training strategy #2: Next Sentence Prediction
  - Given two sentences A and B, enforce model to learn their relationship
  - Beneficial to QA-type downstream tasks



The second task BERT is pre-trained on is a two-sentence classification task. The tokenization is oversimplified in this graphic as BERT actually uses WordPieces as tokens rather than words --- so some words are broken down into smaller chunks.

Pretrain & Finetune LLM/VLM/MLLM

Stage 1

Pre-training by
self-supervised learning
or supervised learning

Stage 2

Finetuning
by downstream tasks
in target domains

Stage 3

RLHF - Reinforcement Learning
with Human Feedback
(will not cover details)

Finetuning of BERT

- Plug in task or domain-specific input/output pairs to finetune all model parameters
  - Token-level tasks (e.g., QA)
  - Classification tasks (e.g., sentiment analysis)
- How about V&L models?

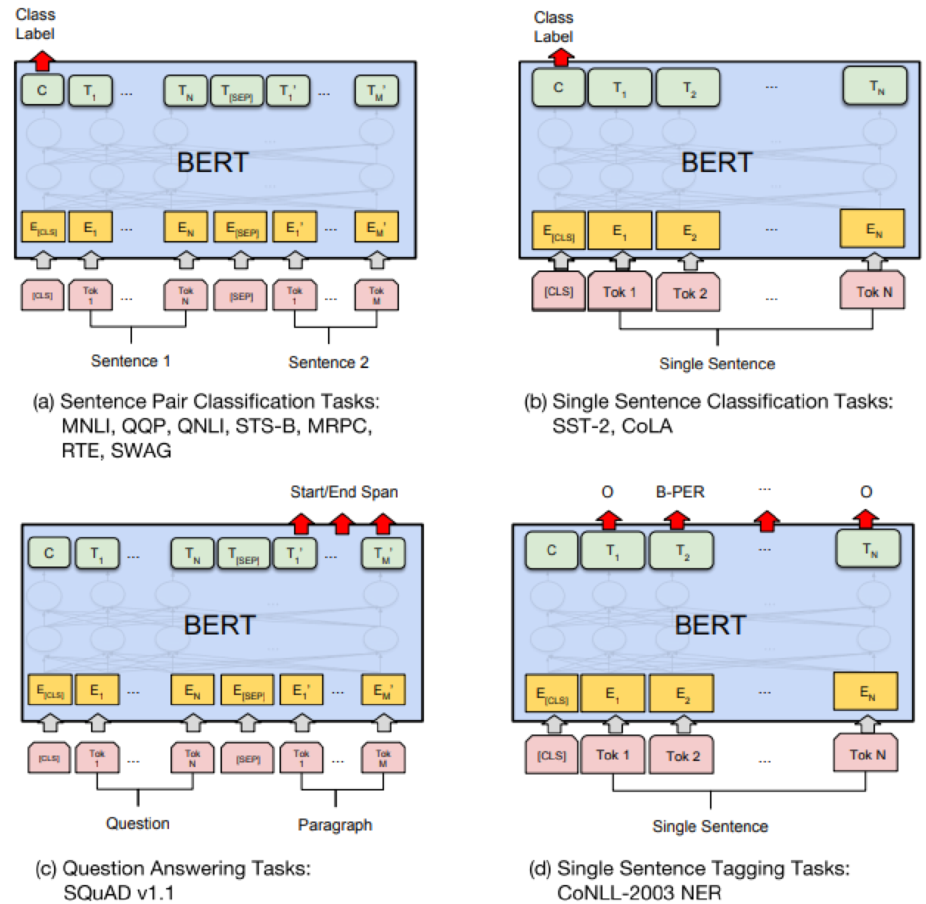
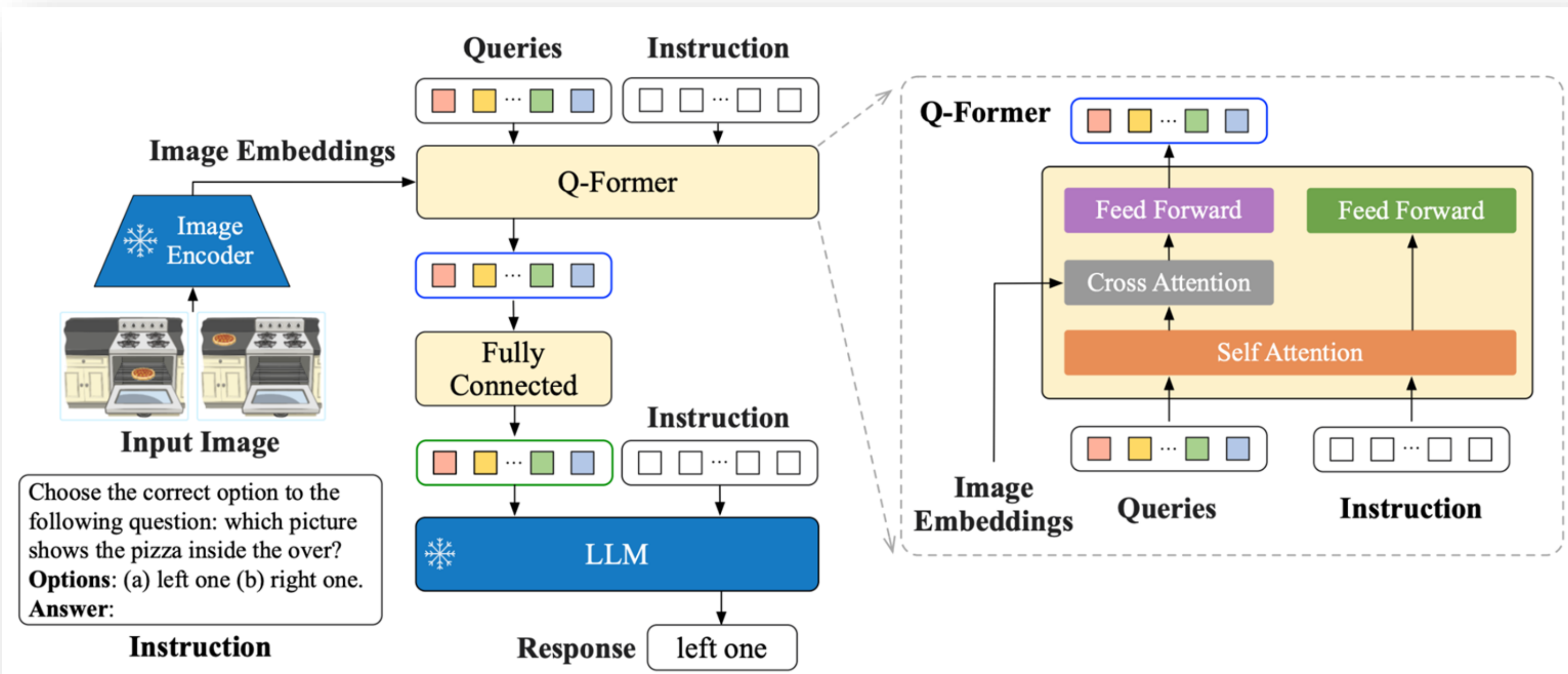


Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

InstructBLIP (NeurIPS'23)

- Enable **BLIP2** to understand instruction following tasks
- Collect **existing** vision-language datasets (**held-in** / **held-out**)

for training for zero-shot



Examples of instruction tuning templates

Task	Instruction Template
Image Captioning	<Image>A short image caption:
	<Image>A short image description:
	<Image>A photo of
	<Image>An image that shows
	<Image>Write a short description for the image.
	<Image>Write a description for the photo.
	<Image>Provide a description of what is presented in the photo.
	<Image>Briefly describe the content of the image.
	<Image>Can you briefly explain what you see in the image?
	<Image>Could you use a few words to describe what you perceive in the photo?
	<Image>Please provide a short depiction of the picture.
	<Image>Using language, provide a short account of the image.
	<Image>Use a few words to illustrate what is happening in the picture.
VQA	<Image>{Question}
	<Imagc>Qucstion: {Qucstion}
	<Image>{Question} A short answer to the question is
	<Image>Q: {Question} A:
	<Image>Question: {Question} Short answer:
	<Image>Given the image, answer the following question with no more than three words. {Question}
	<Image>Based on the image, respond to this question with a short answer: {Question}. Answer:
	<Image>Use the provided image to answer the question: {Question} Provide your answer as short as possible:
	<Image>What is the answer to the following question? "{Question}"
	<Image>The question "{Question}" can be answered using the image. A short answer is
VQG	<Image>Given the image, generate a question whose answer is: {Answer}. Question:
	<Image>Based on the image, provide a question with the answer: {Answer}. Question:
	<Image>Given the visual representation, create a question for which the answer is "{Answer}".
	<Image>From the image provided, craft a question that leads to the reply: {Answer}. Question:
	<Image>Considering the picture, come up with a question where the answer is: {Answer}.
	<Image>Taking the image into account, generate an question that has the answer: {Answer}. Question:

Results on downstream tasks

- Finetune BLIP2 on **held-in** datasets

	ScienceQA IMG	OCR-VQA	OKVQA	A-OKVQA Direct Answer Val	A-OKVQA Direct Answer Test	A-OKVQA Multi-choice Val	A-OKVQA Multi-choice Test
Previous SOTA	LLaVA [25] 89.0	GIT [43] 70.3	PaLM-E(562B) [9] **66.1**	[15] 56.3	[37] 61.6	[15] 73.2	[37] 73.6
BLIP-2 (FlanT5$_{XXL}$)	89.5	72.7	54.7	57.6	53.7	80.2	76.2
InstructBLIP (FlanT5$_{XXL}$)	**90.7**	**73.3**	55.5	57.1	54.8	**81.0**	**76.7**
BLIP-2 (Vicuna-7B)	77.3	69.1	59.3	60.0	58.7	72.1	69.0
InstructBLIP (Vicuna-7B)	79.5	72.8	62.1	**64.0**	**62.1**	75.7	73.4

- Zero-shot vision-language tasks aren't used in instruction tuning (**held-out**)

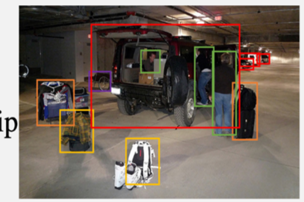
	NoCaps	Flickr 30K	GQA	VSR	IconQA	TextVQA	Visdial	HM	VizWiz	SciQA image	MSVD QA	MSRVTT QA	iVQA
Flamingo-3B [4]	-	60.6	-	-	-	30.1	-	53.7	28.9	-	27.5	11.0	32.7
Flamingo-9B [4]	-	61.5	-	-	-	31.8	-	57.0	28.8	-	30.2	13.7	35.2
Flamingo-80B [4]	-	67.2	-	-	-	35.0	-	46.4	31.6	-	35.6	17.4	40.7
BLIP-2 (FlanT5$_{XL}$) [20]	104.5	76.1	44.0	60.5	45.5	43.1	45.7	53.0	29.8	54.9	33.7	16.2	40.4
BLIP-2 (FlanT5$_{XXL}$) [20]	98.4	73.7	44.6	68.2	45.4	44.1	46.9	52.0	29.4	64.5	34.4	17.4	45.8
BLIP-2 (Vicuna-7B)	107.5	74.9	38.6	50.0	39.7	40.1	44.9	50.6	25.3	53.8	18.3	9.2	27.5
BLIP-2 (Vicuna-13B)	103.9	71.6	41.0	50.9	40.6	42.5	45.1	53.7	19.6	61.0	20.3	10.3	23.5
InstructBLIP (FlanT5$_{XL}$)	119.9	**84.5**	48.4	64.8	50.0	46.6	46.6	56.6	32.7	70.4	43.4	25.0	53.1
InstructBLIP (FlanT5$_{XXL}$)	120.0	83.5	47.9	**65.6**	**51.2**	46.6	**48.5**	54.1	30.9	**70.6**	**44.3**	**25.6**	**53.8**
InstructBLIP (Vicuna-7B)	**123.1**	82.4	49.2	54.3	43.1	50.1	45.2	**59.6**	**34.5**	60.5	41.8	22.1	52.2
InstructBLIP (Vicuna-13B)	121.9	82.8	**49.5**	52.1	44.8	**50.7**	45.4	57.5	33.4	63.1	41.2	24.8	51.0

Visual Instruction Tuning (NeurIPS'23)

- **LLaVA**: **L**arge **L**anguage **a**nd **V**ision **A**ssistant
- Data source: **generated** by GPT-4

Context type 1: Captions

A group of people standing outside of a black vehicle with various luggage.
Luggage surrounds a vehicle in an underground parking area
People try to fit all of their luggage in an SUV.
The sport utility vehicle is parked in the public garage, being packed for a trip
Some people with luggage near a van that is transporting it.



Context type 2: Boxes

person: [0.681, 0.242, 0.774, 0.694], person: [0.63, 0.222, 0.686, 0.516], person: [0.444, 0.233, 0.487, 0.34], backpack: [0.384, 0.696, 0.485, 0.914], backpack: [0.755, 0.413, 0.846, 0.692], suitcase: [0.758, 0.413, 0.845, 0.69], suitcase: [0.1, 0.497, 0.173, 0.579], bicycle: [0.282, 0.363, 0.327, 0.442], car: [0.786, 0.25, 0.848, 0.322], car: [0.783, 0.27, 0.827, 0.335], car: [0.86, 0.254, 0.891, 0.3], car: [0.261, 0.101, 0.787, 0.626]

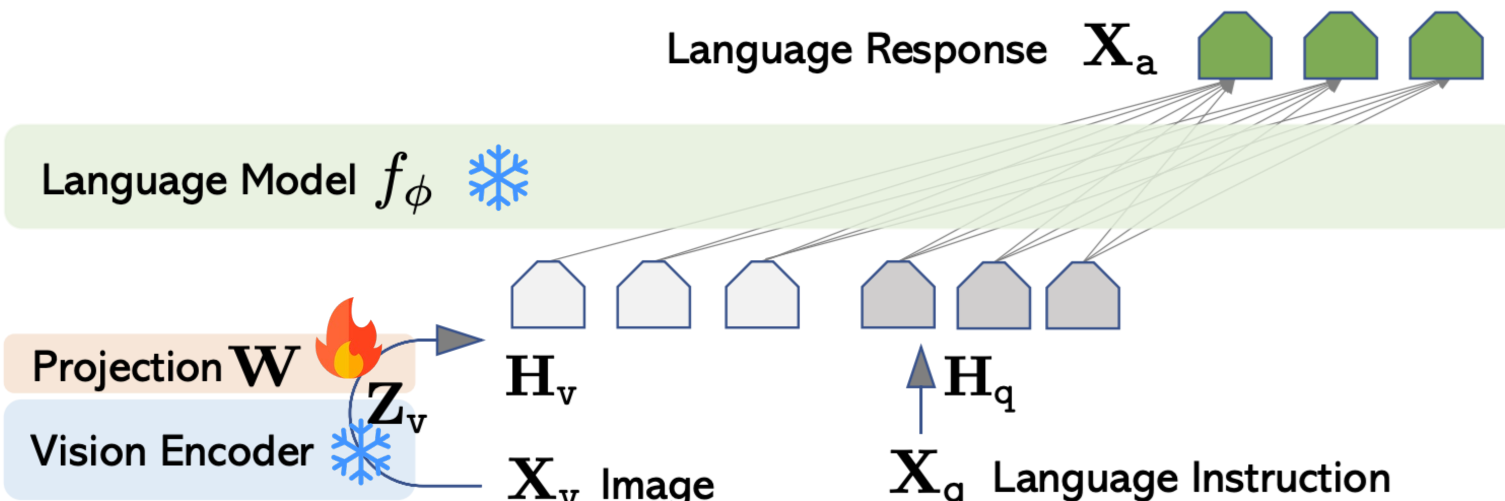
GPT-4

type 1: conversation | **type 2: detailed desciption** | **type 3: complex reasoning**

LLaVA

Stage I: Re-Training for Feature Alignment

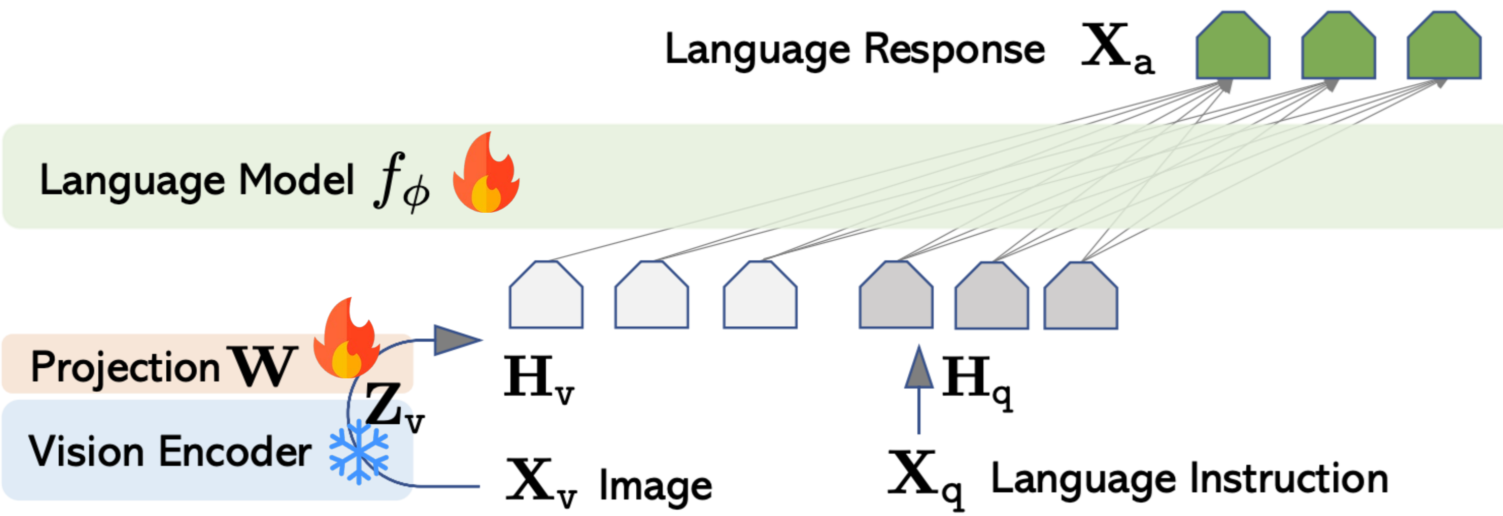
- Finetune projection layer by **image-text pairs** with **instruction tuning**



LLaVA

Stage II: End-to-End Finetuning

- Instruction tuning LLM and projection layers with **vision-language complex reasoning data** from GPT-4



ScienceQA Results

- LLaVA outperforms GPT-4 on ScienceQA dataset

Method	Subject			Context Modality			Grade		Average
	NAT	SOC	LAN	TXT	IMG	NO	G1-6	G7-12	
Representative & SoTA methods with numbers reported in the literature									
Human [30]	90.23	84.97	87.48	89.60	87.50	88.10	91.59	82.42	88.40
GPT-3.5 [30]	74.64	69.74	76.00	74.44	67.28	77.42	76.80	68.89	73.97
GPT-3.5 w/ CoT [30]	75.44	70.87	78.09	74.68	67.43	79.93	78.23	69.68	75.17
LLaMA-Adapter [55]	84.37	88.30	84.36	83.72	80.32	86.90	85.83	84.05	85.19
MM-CoT$_{Base}$ [57]	87.52	77.17	85.82	87.88	82.90	86.83	84.65	85.37	84.91
MM-CoT$_{Large}$ [57]	95.91	82.00	90.82	95.26	88.80	92.89	92.44	90.31	91.68
Results with our own experiment runs									
GPT-4	84.06	73.45	87.36	81.87	70.75	90.73	84.69	79.10	82.69
LLaVA	90.36	95.95	88.00	89.49	88.00	90.66	90.93	90.90	90.92

NAT: Natural Science

SOC: Social Science

LAN: Language Science

TXT: text context

IMG: image context

NO: no context

G1-6: grades 1-6

G7-12: grads 7-12

Pretrain & Finetune LLM/VLM/MLLM

Stage 1

Pre-training by
self-supervised learning
or supervised learning

Stage 2

Finetuning
by downstream tasks
in target domains

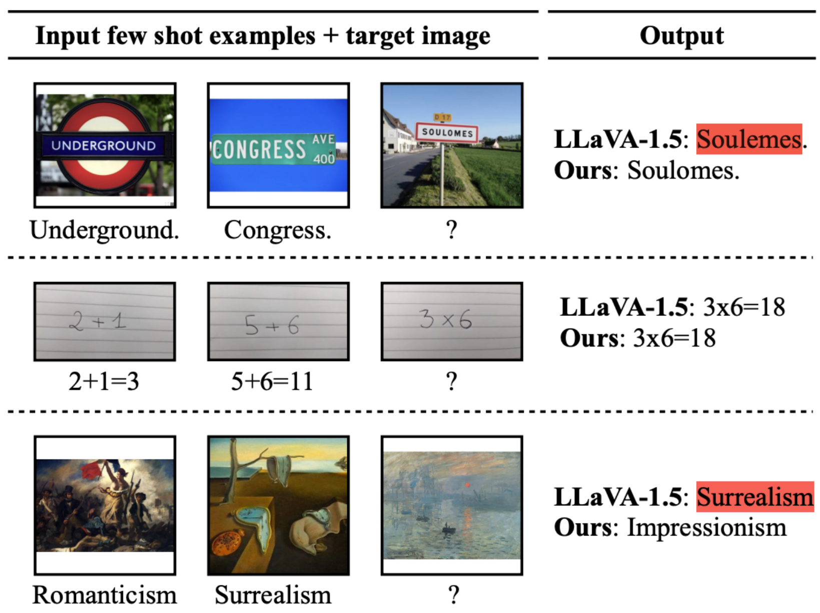
Stage 3

RLHF - Reinforcement Learning
with Human Feedback
(will not cover)

Any concern of the
aforementioned approaches?

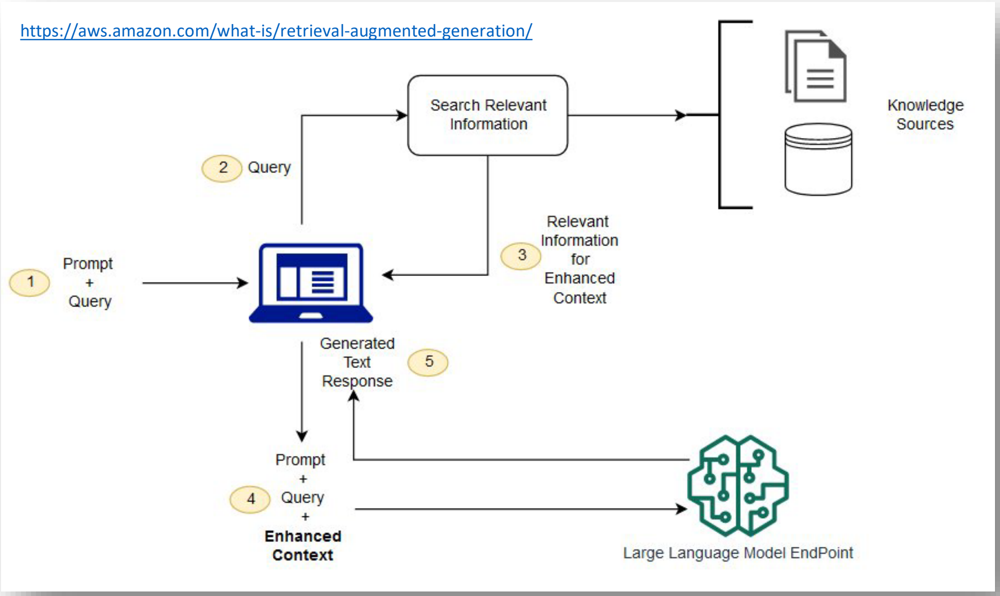
In-Context Learning (ICL)

- Finetuning may not be practical for real-world scenarios
  (e.g., require a large & task-specific dataset)
- Utilize LLM as a few-shot learner -> aka 舉一反三
  (e.g., humans do not require large supervised datasets to learn most tasks.
  A brief directive is typically sufficient…)
- Any limitation?



Retrieval-Augmented Generation (RAG)

- **Ideas:**
  - Combining (traditional) info retrieval + GenAI
  - Can be viewed as open-book exam with cheat sheet
- **Pros:**
  - Access to fresh/untrained information
  - Mitigate hallucination
- **Any limitation?**



Retrieval-Augmented Generation for Knowledge-Intensive NLP Task, Facebook AI Research & UCL, NeurIPS 2020

Pretrain & Finetune LLM/VLM/MLLM

Stage 1

Pre-training by
self-supervised learning
or supervised learning

Stage 2

Finetuning
by downstream tasks
in target domains

Stage 3

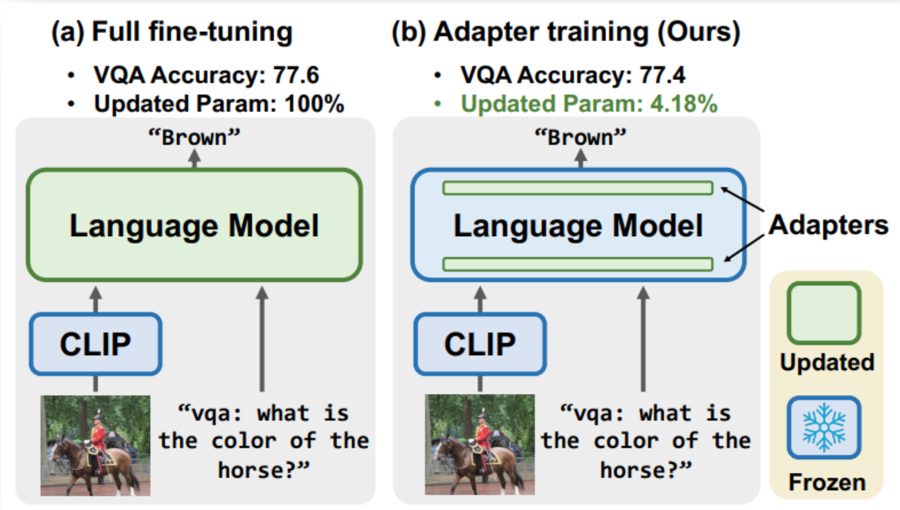
RLHF - Reinforcement Learning
with Human Feedback
(will not cover details)

Can we do FT
in a more efficient way?

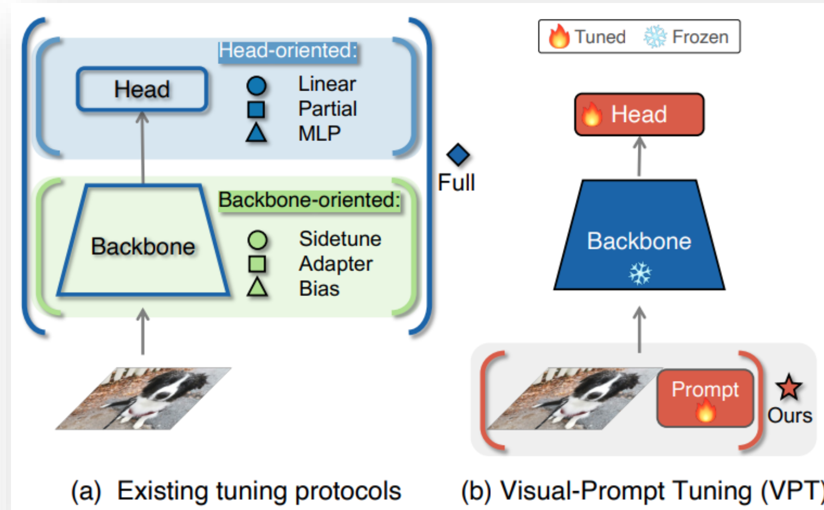
Parameter-Efficient Fine-Tuning (PEFT)

- **Adapter** (帶小抄考試)
    - VL-ADAPTER: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks (CVPR, 2022)
- **Prompt Tuning** (卷哥卷姊提詞器)
    - Visual Prompt Tuning (ECCV, 2022)
- **LoRA** (帶小考答案卷)
    - LoRA: **Lo**w-**R**ank **A**daptation of Large Language Models (ICLR, 2022)
- **DoRA** (台灣研發LoRA進階版)
    - Weight-**D**ecomposed L**o**w-**R**ank **A**daptation (ICML, 2024)

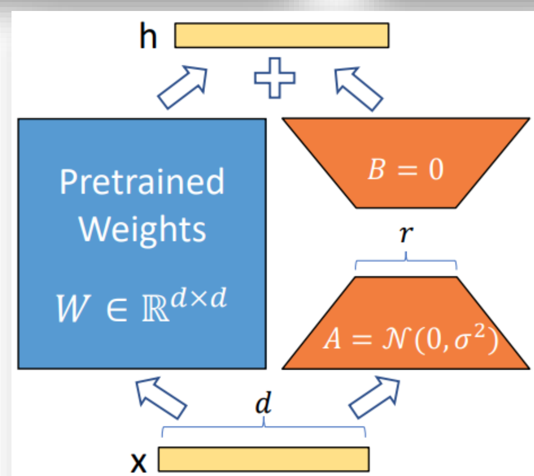
Parameter Efficient Fine Tuning



Adapter



Prompt Tuning

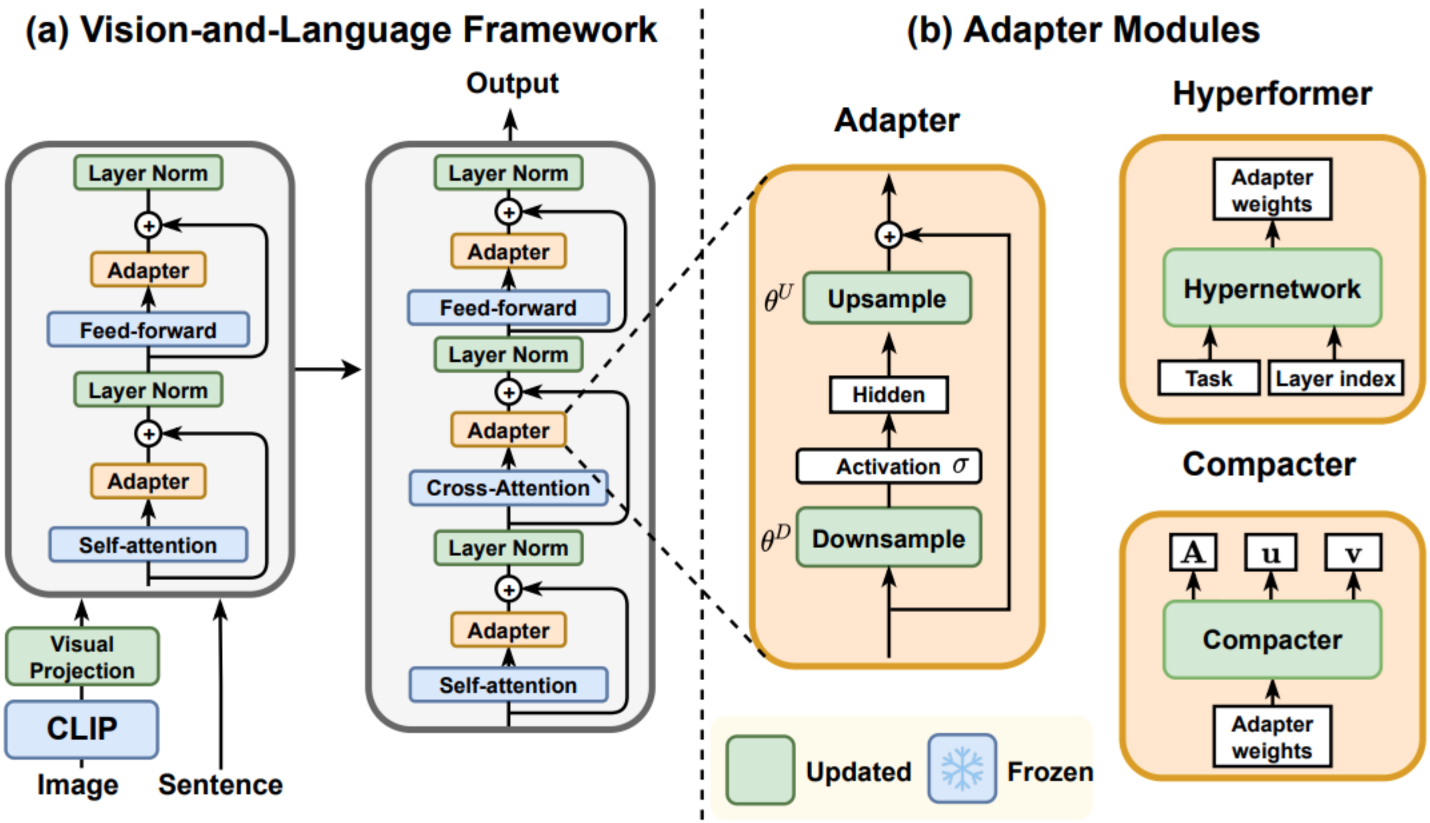


LoRA

Pros & Cons for each?
Will discuss later...

PEFT (1/3): VL-ADAPTER

Parameter-Efficient Transfer Learning for Vision-and-Language Tasks



VL-Adapter: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks, UNC, CVPR 2022

PEFT (1/3): VL-ADAPTER

Parameter-Efficient Transfer Learning for Vision-and-Language Tasks (cont'd)

- Variants for adapter modules
  - Adapter

    $$h = f_{\theta^U}(\sigma(f_{\theta^D}(\boldsymbol{x}))) + \boldsymbol{x}$$

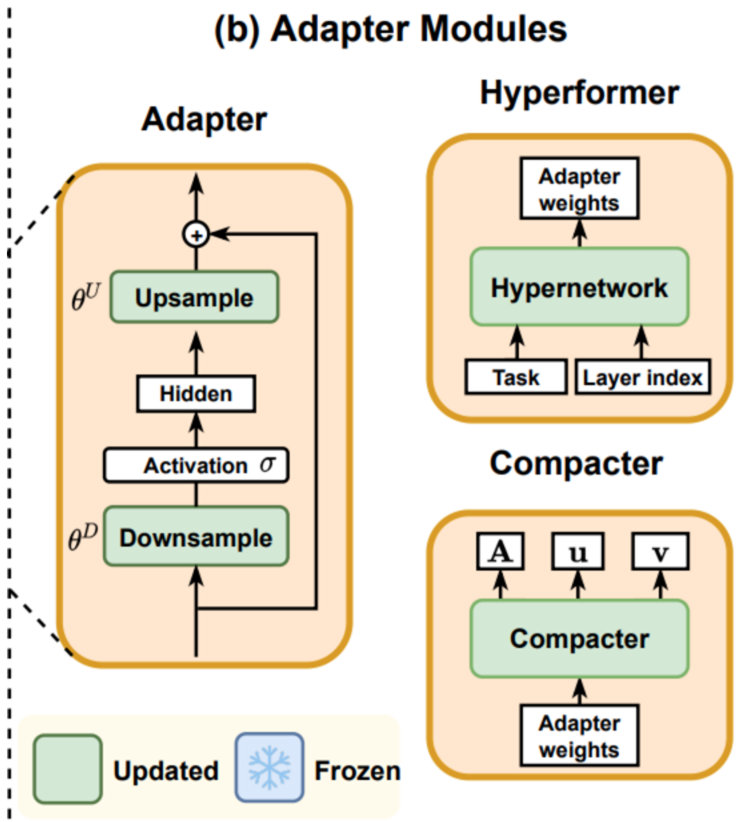
  - Hyperformer

    $$[\theta^D, \theta^U] = f_{\theta^H}(f_{\theta^T}([\boldsymbol{t}_j, \boldsymbol{l}_i]))$$

  - Compacter

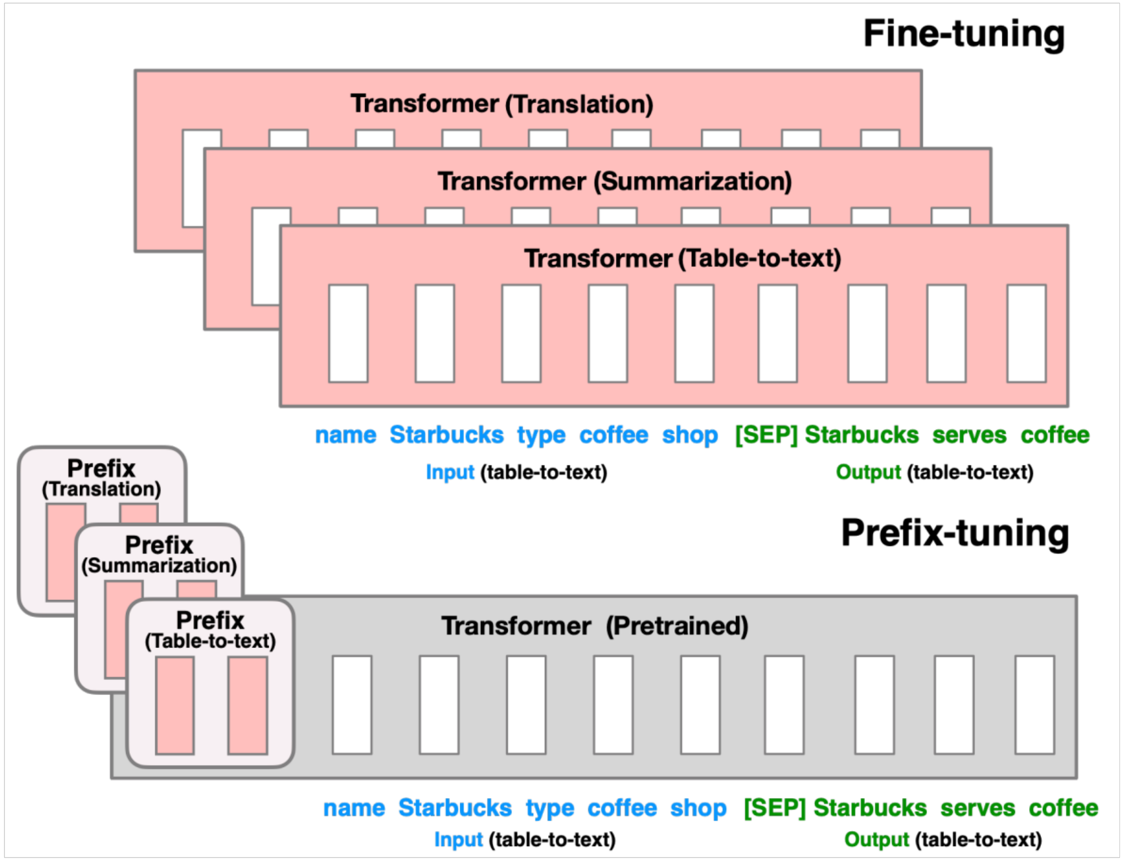
    $$\theta^D = \sum_{i=1}^{k} A_i \otimes B_i = \sum_{i=1}^{k} A_i \otimes (\boldsymbol{u}_i \boldsymbol{v}_i)$$

- Trade-off between performance and efficiency
- Any concern?



PEFT (2/3): Prefix Tuning (Prompt Tuning)

提詞機



Prefix-Tuning: Optimizing Continuous Prompts for Generation, Stanford, ACL 2021

Prompt Tuning

- Shallow:

$$[\mathbf{x}_1, \mathbf{Z}_1, \mathbf{E}_1] = L_1([\mathbf{x}_0, \mathbf{P}, \mathbf{E}_0]) \tag{4}$$

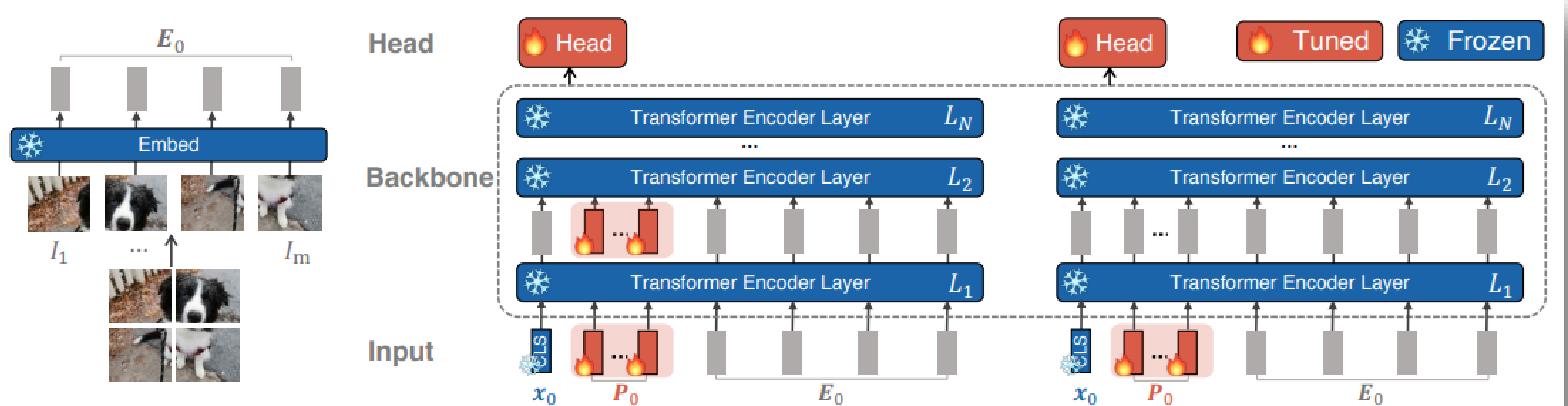
$$[\mathbf{x}_i, \mathbf{Z}_i, \mathbf{E}_i] = L_i([\mathbf{x}_{i-1}, \mathbf{Z}_{i-1}, \mathbf{E}_{i-1}]) \qquad i = 2, 3, \ldots, N \tag{5}$$

$$\mathbf{y} = \texttt{Head}(\mathbf{x}_N) \ , \tag{6}$$

- Deep:

$$[\mathbf{x}_i, \_\_, \mathbf{E}_i] = L_i([\mathbf{x}_{i-1}, \mathbf{P}_{i-1}, \mathbf{E}_{i-1}]) \qquad i = 1, 2, \ldots, N \tag{7}$$

$$\mathbf{y} = \texttt{Head}(\mathbf{x}_N) \ . \tag{8}$$

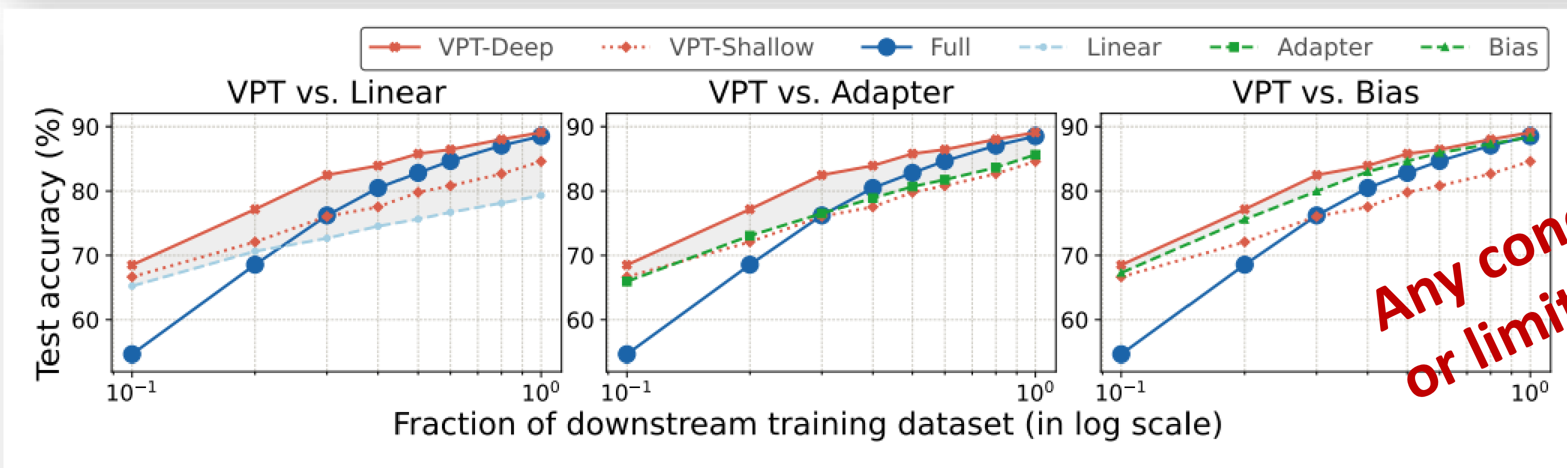


(a) Visual-Prompt Tuning: Deep (b) Visual-Prompt Tuning: Shallow

Visual Prompt Tuning, Meta AI, ECCV 2022

Visual Prompt Tuning

	ViT-B/16 (85.8M)	Total params	Scope Input	Scope Backbone	Extra params	FGVC	VTAB-1k Natural	VTAB-1k Specialized	VTAB-1k Structured
	Total # of tasks					5	7	4	8
(a)	Full	24.02×		✓		88.54	75.88	83.36	47.64
(b)	Linear	1.02×				79.32 (0)	68.93 (1)	77.16 (1)	26.84 (0)
	Partial-1	3.00×				82.63 (0)	69.44 (2)	78.53 (0)	34.17 (0)
	Mlp-3	1.35×			✓	79.80 (0)	67.80 (2)	72.83 (0)	30.62 (0)
(c)	Sidetune	3.69×		✓	✓	78.35 (0)	58.21 (0)	68.12 (0)	23.41 (0)
	Bias	1.05×		✓		88.41 (3)	73.30 (3)	78.25 (0)	44.09 (2)
	Adapter	1.23×		✓	✓	85.66 (2)	70.39 (4)	77.11 (0)	33.43 (0)
(ours)	VPT-shallow	1.04×	✓		✓	84.62 (1)	76.81 (4)	79.66 (0)	46.98 (4)
	VPT-deep	1.18×	✓		✓	**89.11** (4)	**78.48** (6)	**82.43** (2)	**54.98** (8)



PEFT (3/3): LoRA
Low-Rank Adaptation of Large Language Models

- Previous problems
  - Adapter Layers introduce extra inference latency
  - Directly optimizing the prompt may not be easy

- LoRA

$$W_0 \in \mathbb{R}^{d\times k}$$
$$W_0 + \Delta W = W_0 + BA$$
$$B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$$
$$\text{rank } r \ll \min(d, k)$$

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

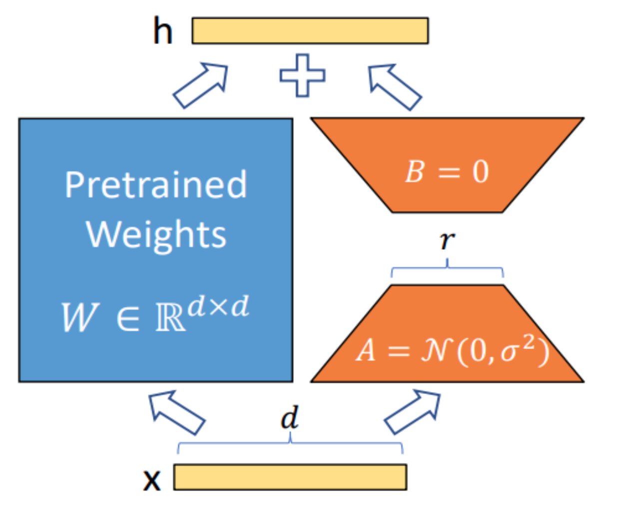


Figure 1: Our reparametrization. We only train A and B.

LoRA: Low-Rank Adaptation of Large Language Models,
Microsoft, ICLR 2022

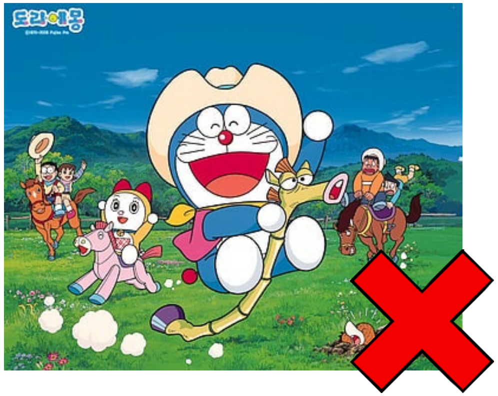
LoRA: Low-Rank Adaptation of LLMs (cont'd)

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$_{\pm.3}$	**95.1**$_{\pm.2}$	89.7$_{\pm.7}$	63.4$_{\pm1.2}$	**93.3**$_{\pm.3}$	90.8$_{\pm.1}$	**86.6**$_{\pm.7}$	**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	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.9**$_{\pm1.2}$	**68.2**$_{\pm1.9}$	**94.9**$_{\pm.3}$	91.6$_{\pm.1}$	**87.4**$_{\pm2.5}$	**92.6**$_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	90.2$_{\pm.3}$	96.1$_{\pm.3}$	90.2$_{\pm.7}$	**68.3**$_{\pm1.0}$	**94.8**$_{\pm.2}$	**91.9**$_{\pm.1}$	83.8$_{\pm2.9}$	92.1$_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	**90.5**$_{\pm.3}$	**96.6**$_{\pm.2}$	89.7$_{\pm1.2}$	67.8$_{\pm2.5}$	**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	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.2**$_{\pm1.0}$	68.2$_{\pm1.9}$	**94.8**$_{\pm.3}$	91.6$_{\pm.2}$	**85.2**$_{\pm1.1}$	**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	**91.9**$_{\pm.2}$	96.9$_{\pm.2}$	**92.6**$_{\pm.6}$	**72.4**$_{\pm1.1}$	**96.0**$_{\pm.1}$	**92.9**$_{\pm.1}$	**94.9**$_{\pm.4}$	**93.0**$_{\pm.2}$	**91.3**

Model & Method	# Trainable Parameters	E2E NLG Challenge				
		BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M ($\text{Adapter}^{\text{L}}$)*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M ($\text{Adapter}^{\text{L}}$)*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M ($\text{Adapter}^{\text{H}}$)	11.09M	67.3$_{\pm.6}$	8.50$_{\pm.07}$	46.0$_{\pm.2}$	70.7$_{\pm.2}$	2.44$_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	**70.4**$_{\pm.1}$	**8.85**$_{\pm.02}$	**46.8**$_{\pm.2}$	**71.8**$_{\pm.1}$	**2.53**$_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L ($\text{Adapter}^{\text{L}}$)	0.88M	69.1$_{\pm.1}$	8.68$_{\pm.03}$	46.3$_{\pm.0}$	71.4$_{\pm.2}$	**2.49**$_{\pm.0}$
GPT-2 L ($\text{Adapter}^{\text{L}}$)	23.00M	68.9$_{\pm.3}$	8.70$_{\pm.04}$	46.1$_{\pm.1}$	71.3$_{\pm.2}$	2.45$_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	**70.4**$_{\pm.1}$	**8.89**$_{\pm.02}$	**46.8**$_{\pm.2}$	**72.0**$_{\pm.2}$	2.47$_{\pm.02}$

DoRA: Weight-Decomposed Low-Rank Adaptation

- Previous problems
  - Adapter layers introduce extra inference latency
  - Directly optimizing the prompt may not be sufficient
  - LoRA still exhibits performs gap (vs. FT)



- LoRA

$$W_0 \in \mathbb{R}^{d \times k}$$

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

$$B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$$

$$\text{rank } r \ll \min(d, k)$$

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

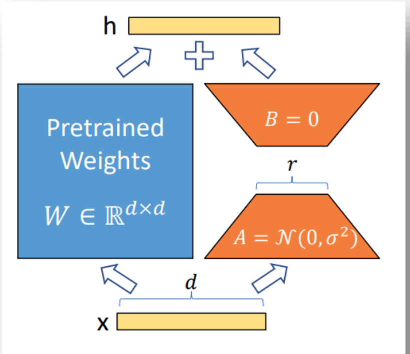


Figure 1: Our reparametrization. We only train A and B.

DoRA: Weight-Decomposed Low-Rank Adaptation
NVIDIA (Taiwan), ICML 2024

DoRA: Weight-Decomposed Low-Rank Adaptation (cont'd)

- **Weight Decomposition Analysis**
  - Decompose weight matrix into magnitude & direction components
  - Investigate ***ΔM***, ***ΔD*** during training

$$W = m\frac{V}{||V||_c} = ||W||_c\frac{W}{||W||_c}$$

$W \in \mathbb{R}^{d\times k}$ - weight matrix

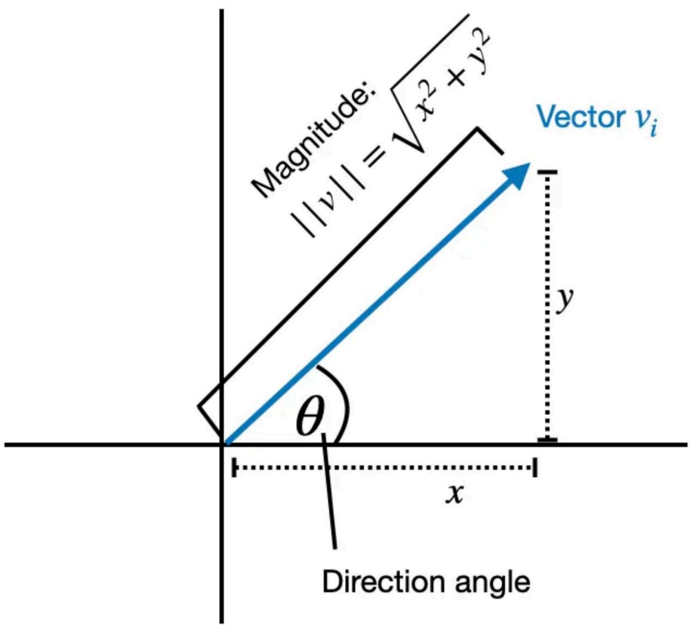
$m \in \mathbb{R}^{1\times k}$ - magnitude vector, Euclidean norm of ***W***

$V \in \mathbb{R}^{d\times k}$ - directional component

Magnitude and direction deviation from the original weights:

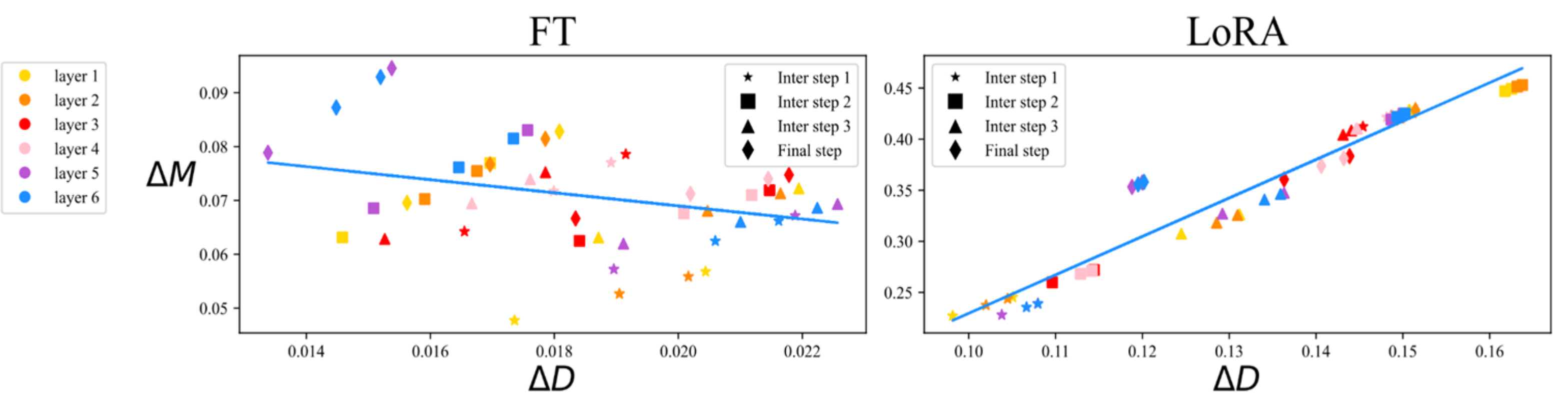
$$\Delta M_{\mathrm{FT}}^{t} = \frac{\sum_{n=1}^{k}|m_{\mathrm{FT}}^{n,t} - m_0^n|}{k}$$

$$\Delta D_{\mathrm{FT}}^{t} = \frac{\sum_{n=1}^{k}(1 - \mathbf{cos}(V_{\mathrm{FT}}^{n,t}, W_0^n))}{k}$$



DoRA: Weight-Decomposed Low-Rank Adaptation (cont'd)

- **Observations** (can come back to this later...)
  - LoRA shows positive slope trends (i.e., ***ΔM/ΔD***) across all training steps
  - FT results in somewhat diverse yet negative slope...
    Probably due to the fact that pre-trained models contains sufficient knowledge and no need to update both **M** and **D** drastically
  - LoRA lacks the above learning capability in carrying out subtle adjustment.



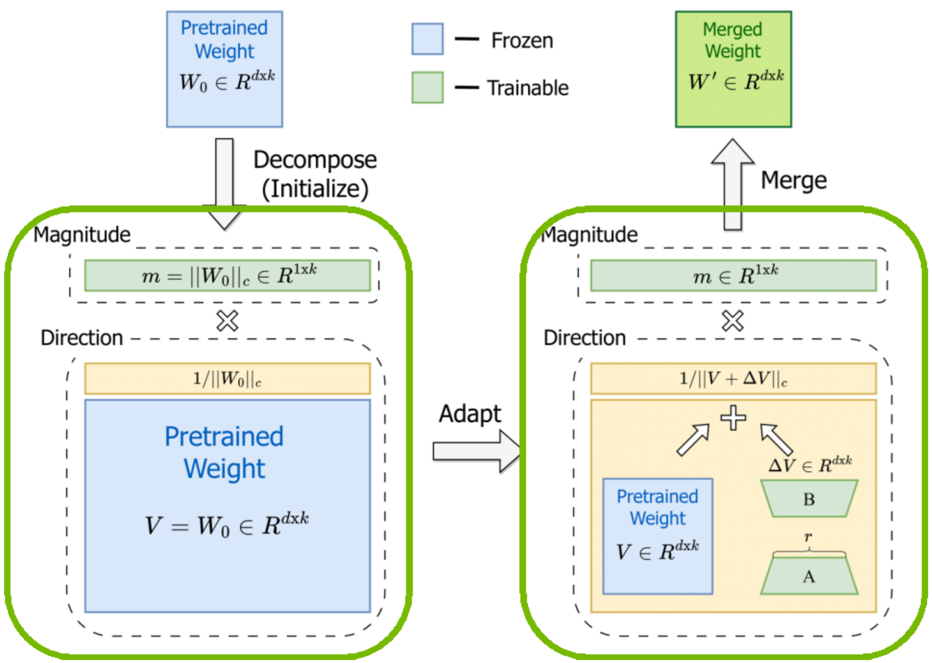
(a) (b)

DoRA: Weight-Decomposed Low-Rank Adaptation (cont'd)

- Ideas
  - Capacity gap between LoRA & FT comes from the complexity of learning of both magnitude and directional adaption
  - Enforce **directional adaptation** via **LoRA**,
    while allowing the **magnitude** component to be **tunable**
  - No additional inference costs (same as LoRA)

Step 1: Decomposition

$$W = m\frac{V}{||V||_c} = ||W||_c\frac{W}{||W||_c}$$



Step 2: Adaptation

LoRA tuning

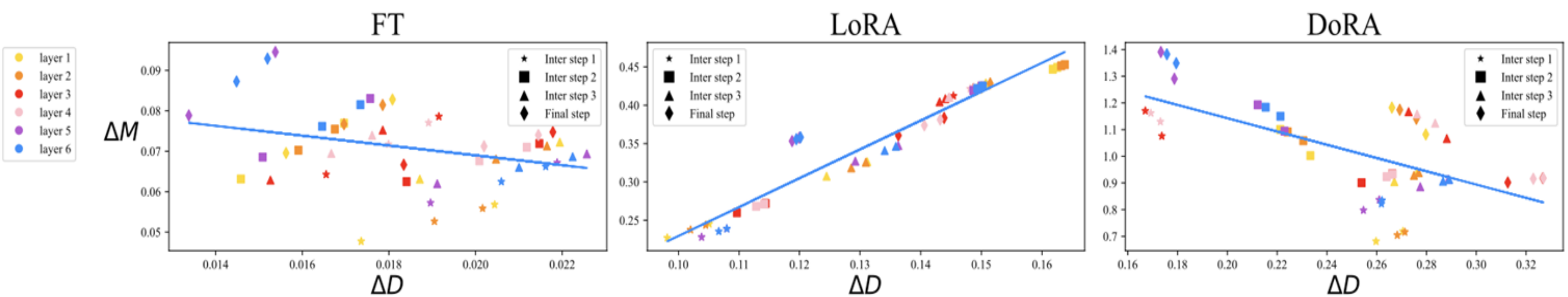
$$W' = \boxed{m}\frac{\boxed{V + \Delta V}}{||V + \Delta V||_c} = m\frac{W_0 + \underline{BA}}{||W_0 + \underline{BA}||_c}$$

Standard tuning

DoRA, NVIDIA (Taiwan), ICML 2024

DoRA: Weight-Decomposed Low-Rank Adaptation (cont'd)

- **Remarks**
  - In contrast to LoRA, DoRA, and FT are characterized by a distinct negative slope
  - DoRA demonstrates the ability to make only substantial directional adjustments with relatively minimal changes in magnitude or the reverse, showing learning patterns closer to FT's (i.e., better learning ability over LoRA).
  - Discussions on Twitter/X: 4K+ likes, 700+ tweets, 500K+ views!



(a) (b) (c)

DoRA, NVIDIA (Taiwan), ICML 2024

DoRA: Weight-Decomposed Low-Rank Adaptation (cont'd)

- Results
  - How to perform fair comparisons?

Model	PEFT Method	# Params (%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
ChatGPT	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	Prefix	0.11	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series	0.99	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel	3.54	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.2
	LoRA	0.83	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	DoRA† (Ours)	0.43	70.0	82.6	79.7	83.2	80.6	80.6	65.4	77.6	**77.5**
	DoRA (Ours)	0.84	69.7	83.4	78.6	87.2	81.0	81.9	66.2	79.2	**78.4**
LLaMA-13B	Prefix	0.03	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Series	0.80	71.8	83	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Parallel	2.89	72.5	84.9	79.8	92.1	84.7	84.2	71.2	82.4	81.4
	LoRA	0.67	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA† (Ours)	0.35	72.5	85.3	79.9	90.1	82.9	82.7	69.7	83.6	**80.8**
	DoRA (Ours)	0.68	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	**81.5**
LLaMA2-7B	LoRA	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA† (Ours)	0.43	72.0	83.1	79.9	89.1	83.0	84.5	71.0	81.2	**80.5**
	DoRA (Ours)	0.84	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	**79.7**
LLaMA3-8B	LoRA	0.70	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA† (Ours)	0.35	74.5	88.8	80.3	95.5	84.7	90.1	79.1	87.2	**85.0**
	DoRA (Ours)	0.71	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	**85.2**

DoRA, NVIDIA (Taiwan), ICML 2024

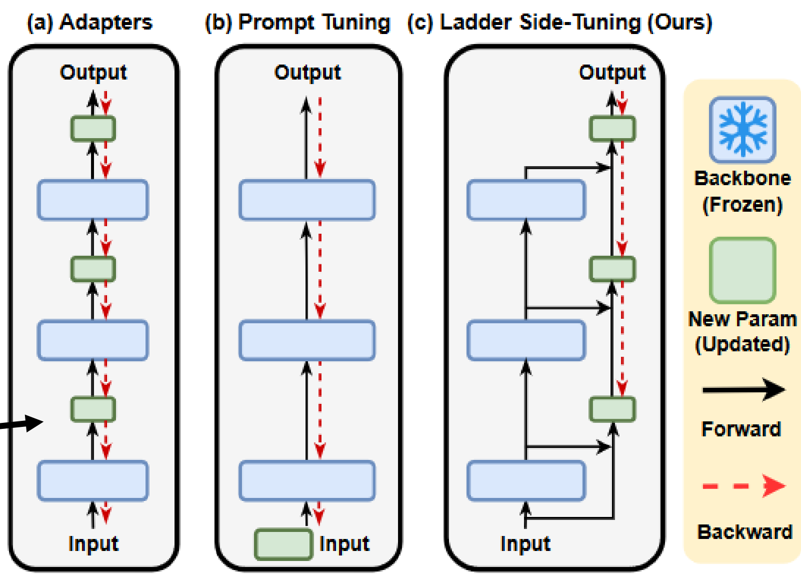
Not Exactly PEFT…
Memory Efficient Adapter (NeurIPS'22)

- To update adapters, all intermediate results must be saved to do backprop.
  ⇒ Standard adapters won't reduce memory much during training

Intermediate gradient to store

Chain rule: $\frac{\partial l_{i+1}}{\partial f_A} = \overbrace{\frac{\partial l_{i+1}}{\partial f_i^1} \frac{\partial f_i^1}{\partial f_i^2} \frac{\partial f_i^2}{\partial f_i^3}} \cdots \frac{\partial f_i^m}{\partial f_A}$

Weights to update

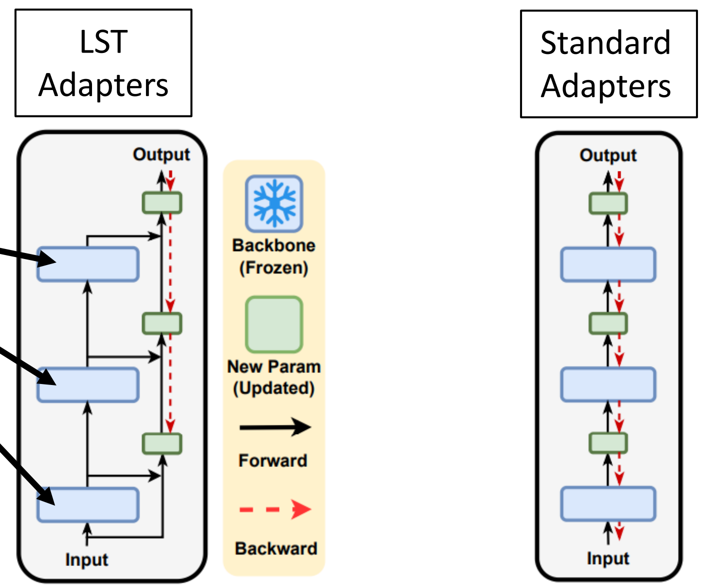


LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning, UNC, NeurIPS'22

Memory Efficient Adapter (cont'd)

- We can save memory by using "ladder" design of adapters
  ⇒ Fewer resources are required to finetune a large pre-trained model

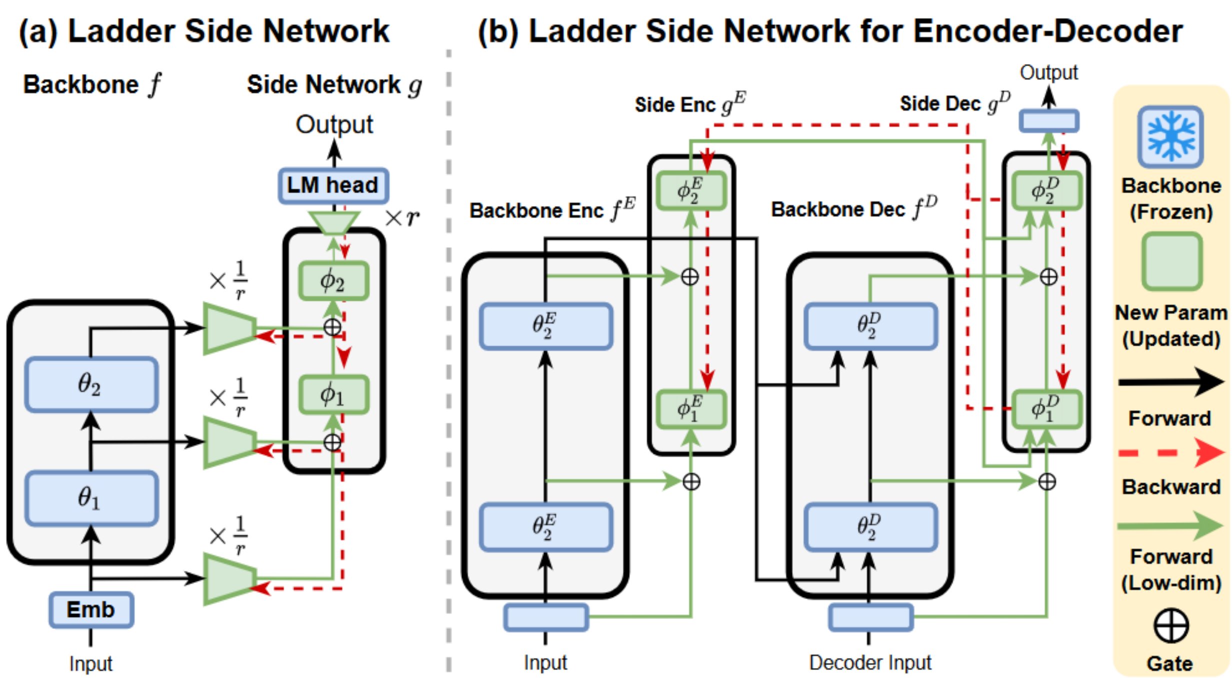
We don't need to save results in backbone to do backprop.



LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning, UNC, NeurIPS'22

Memory Efficient Adapter (cont'd)

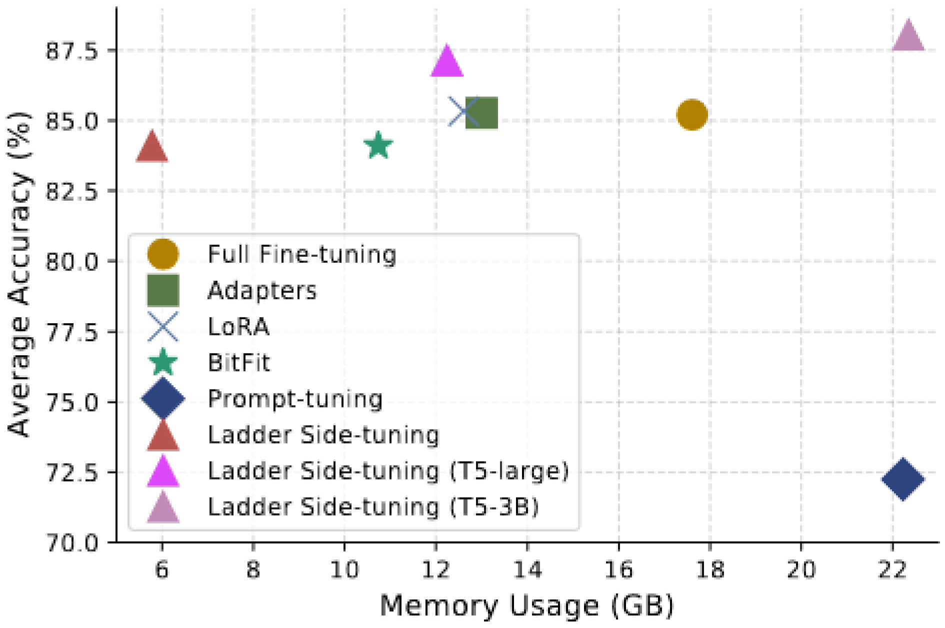
- We can save memory by using "ladder" design of adapters
  ⇒ Fewer resources are required to finetune a large pre-trained model



LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning, UNC, NeurIPS'22

Memory Efficient Adapter (cont'd)

- Results



LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning, UNC, NeurIPS'22

Pretrain & Finetune LLM/VLM/MLLM

Stage 1

Pre-training by
self-supervised learning
or supervised learning

Stage 2

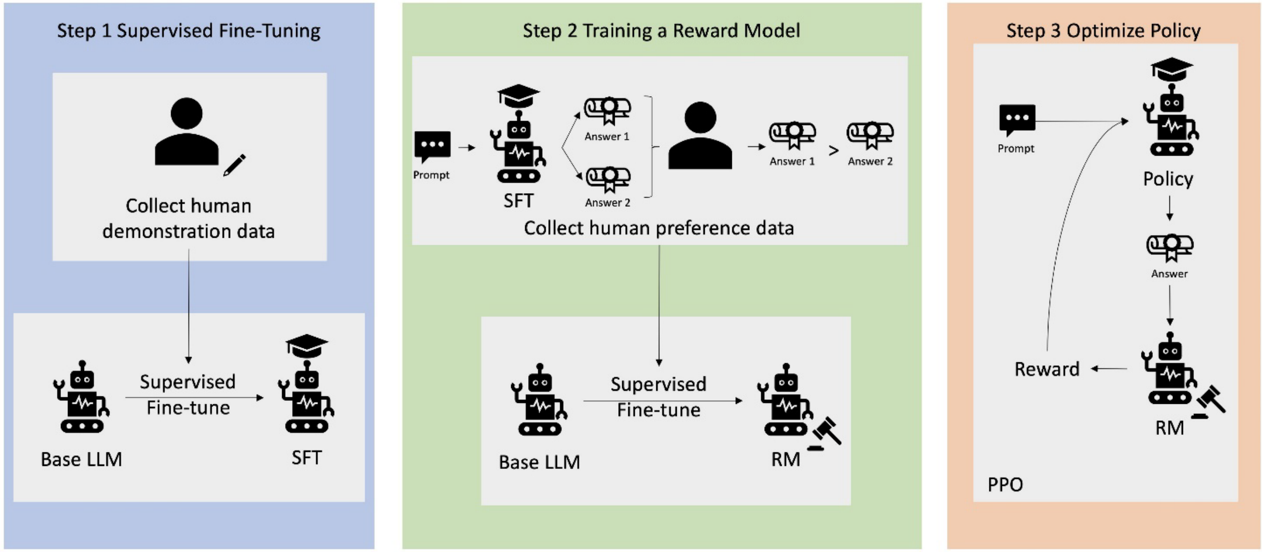
Finetuning
by downstream tasks
in target domains

Stage 3

RLHF - Reinforcement Learning
with Human Feedback
(will not cover details)

RLHF

- **Ideas:**
  - Localizing & finetuning specific network modules/layers are not easy
  - High-level, complex, or subjective information cannot be explicitly modeled
  - Train an "human expert" model to criticize/update the model accordingly.
  - How?
    - Pre-train model -> supervised finetuning -> reward model training -> Policy optimization
  - **Any concern?**



https://aws.amazon.com/tw/what-is/reinforcement-learning-from-human-feedback/

What to Be Covered?

- Learning Vision & Language Models
  - Pretraining
  - Finetuning,
    In-Context Learning &
    Retrieval-Augmented Generation
  - Parameter-Efficient Fine-Tuning
- Advanced Topics
  - Concept Editing
  - Concept Unlearning
- **Experience Sharing** (30 min.)
  - Grad Study & AI opportunities in France
- **HW #3 is out!**