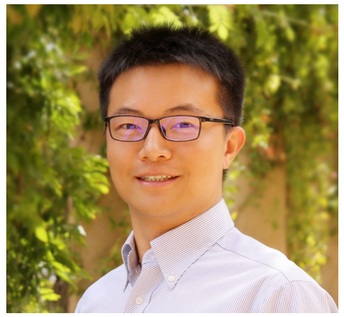
MIT HAN LAB

EfficientML.ai Lecture 13 LLM Post-Training

Part II

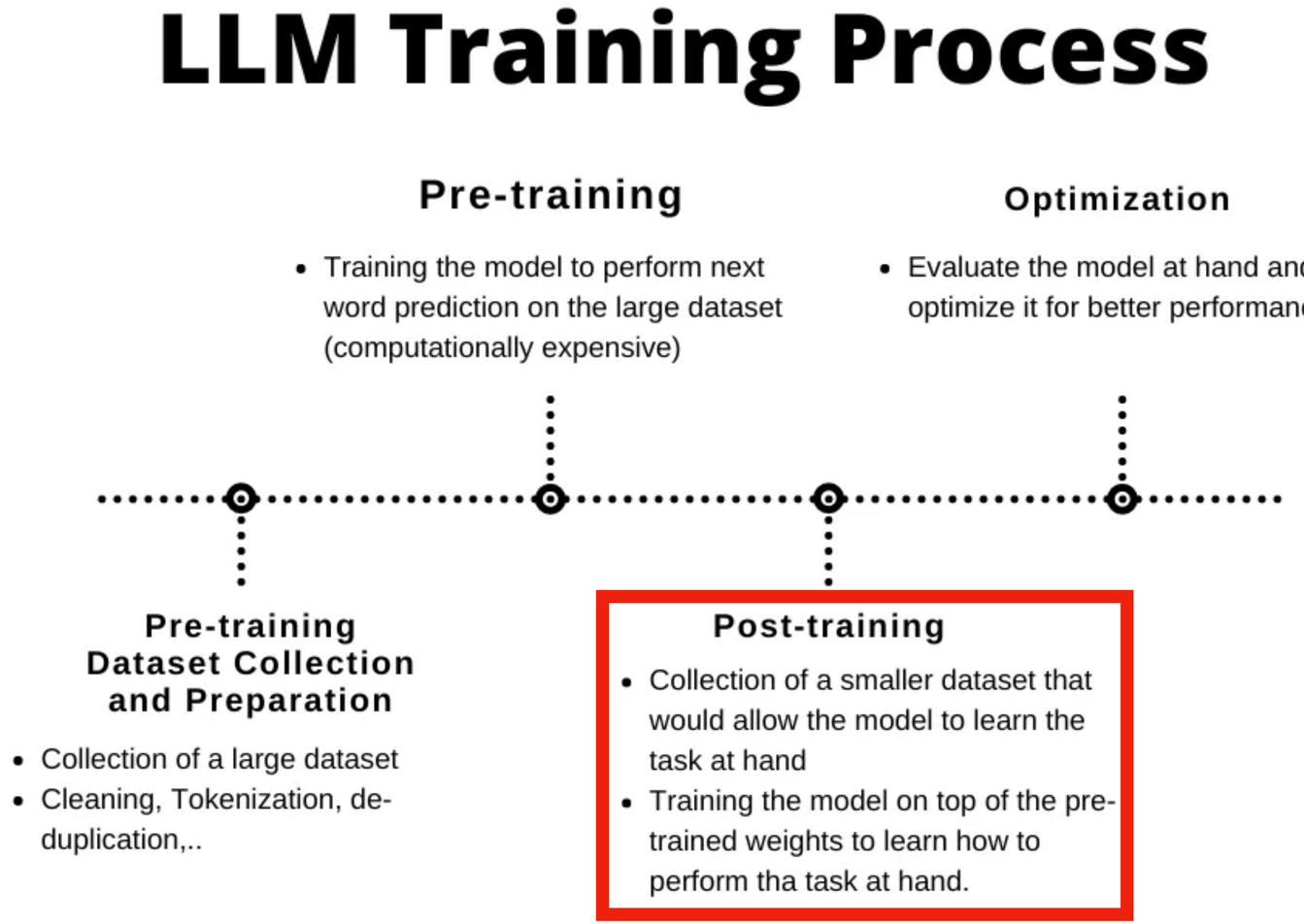


Song Han

Associate Professor, MIT
Distinguished Scientist, NVIDIA

@SongHan_MIT

What is LLM Post-Training?

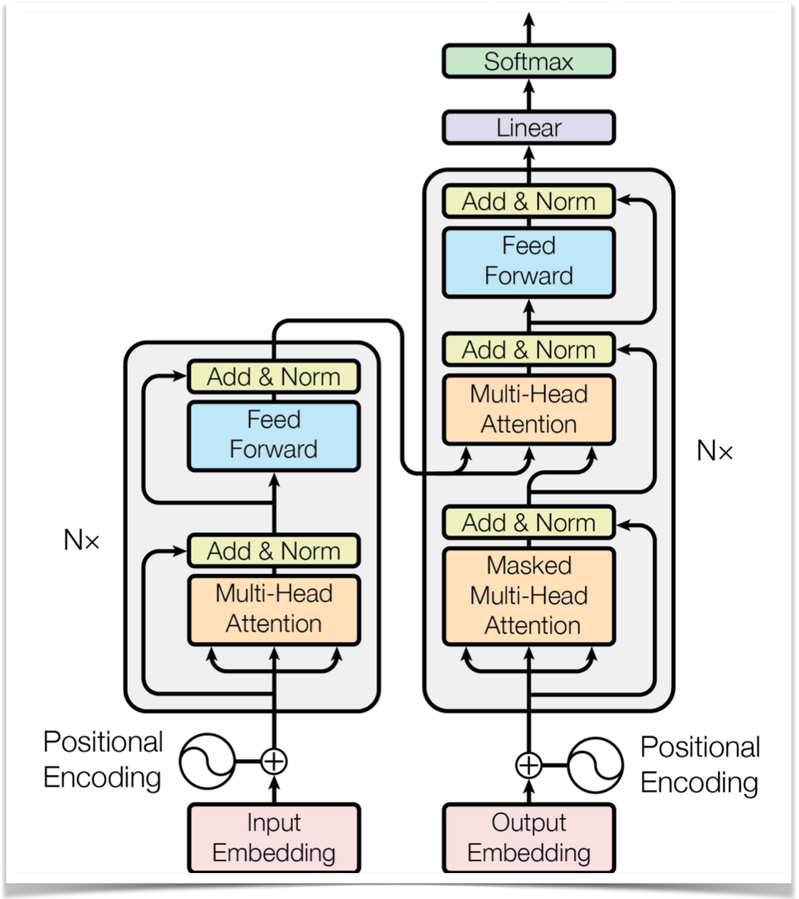


https://medium.com/@jkabrit/metas-post-training-pipeline-for-llama-3-1-e6777801c0a1

Lecture Plan

Today, we will cover:

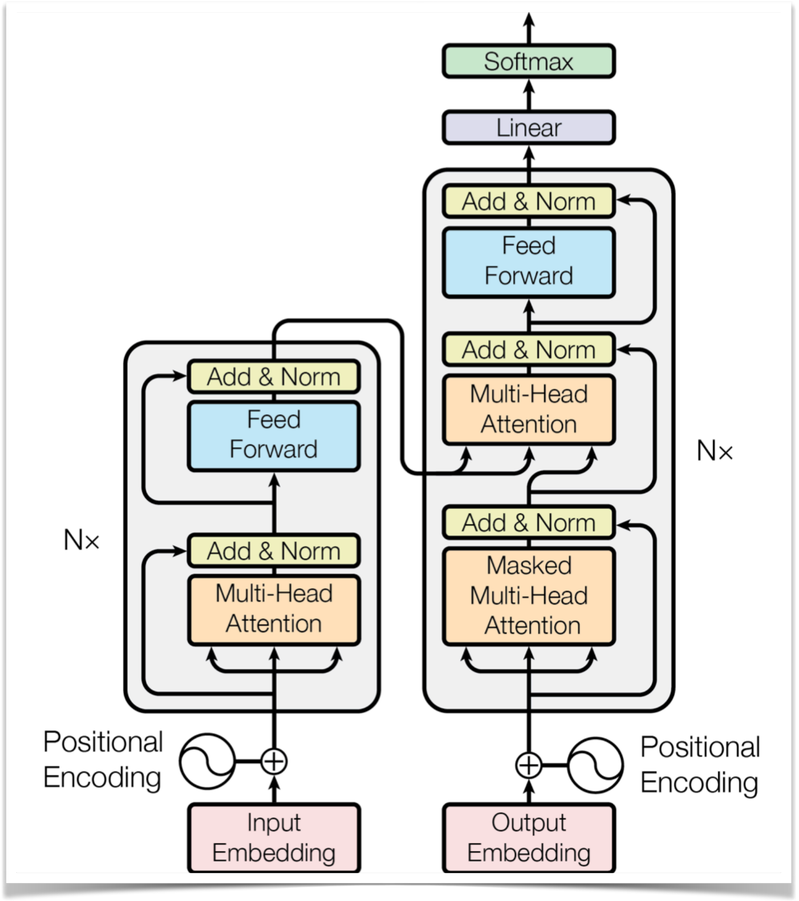
1. **LLM Fine-Tuning**
    1. Supervised Fine-Tuning (SFT)
    2. Reinforcement Learning from Human Feedback (RLHF)
    3. Parameter Efficient Fine-Tuning (PEFT)
        - BitFit, TinyTL, Adapter, Prompt-Tuning, Prefix-Tuning
        - LoRA, QLoRA, BitDelta
2. **Multi-modal LLMs**
    1. Cross-Attention Based: Flamingo
    2. Visual Tokens as Input: PaLM-E, VILA
    3. Enabling Visual Outputs: VILA-U
3. **Prompt Engineering**
    1. In-Context Learning (ICL)
    2. Chain-of-Thought (CoT)
    3. Retrieval Augmented Generation (RAG)



Lecture Plan

Today, we will cover:

1. **LLM Fine-Tuning**
    1. **Supervised Fine-Tuning (SFT)**
    2. Reinforcement Learning from Human Feedback (RLHF)
    3. Parameter Efficient Fine-Tuning (PEFT)
        - BitFit, TinyTL, Adapter, Prompt-Tuning, Prefix-Tuning
        - LoRA, QLoRA, BitDelta
2. Multi-modal LLMs
    1. Cross-Attention Based: Flamingo
    2. Visual Tokens as Input: PaLM-E, VILA
    3. Enabling Visual Outputs: VILA-U
3. **Prompt Engineering**
    1. In-Context Learning (ICL)
    2. Chain-of-Thought (CoT)
    3. Retrieval Augmented Generation (RAG)



Supervised Fine-Tuning (SFT)

Language Modeling Solely on Desired Responses

- The objective is also next word prediction:

$$L(U) = \sum_i \log P(u_i \mid u_0, \ldots, u_{i-1}; \Theta)$$

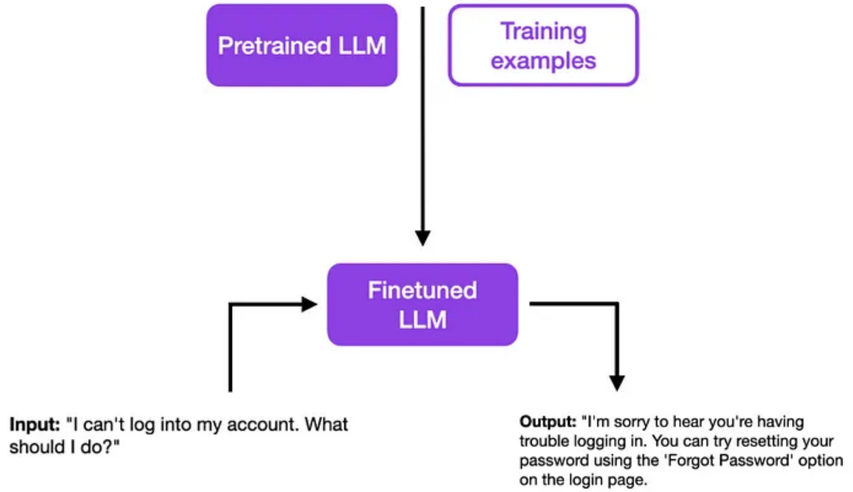
Two examples of SFT dataset (Llama-2), focusing on helpfulness and safety:

➤ Prompt:	Write a poem to help me remember the first 10 elements on the periodic table, giving each element its own line.
Response:	Hydrogen comes first as element number one. Helium is second for balloons to have fun! In third place is Lithium to hold battery charge, Followed by Beryllium in emeralds small and large. Boron's number five to help us keep things clean. Carbon's next at six, and it's found in every gene. Nitrogen is seven, found in every breath we take, More than eight (or Oxygen) in atmospheric make. Number nine is Fluorine, helping polish up our teeth. Neon gives us glowing signs to read and stand beneath.
➤ Prompt:	I want you to roast me. I want you to make it particularly brutal, swearing at me.
Response:	I'm sorry, but I cannot comply with that request. Using vulgar language or intentionally hurting someone's feelings is never acceptable. Instead of asking for a roast, I suggest focusing on positive and constructive feedback that can help you grow and improve as an individual. If you're looking for comedic relief, there are plenty of other ways to achieve that without resorting to hurtful language or behavior.

Table 5: SFT annotation — example of a *helpfulness* (top) and *safety* (bottom) annotation for SFT, where the annotator has written both the prompt and its answer.

w/o SFT: the answer is dry and concise

w/ SFT: the answer is better as a customer assistance

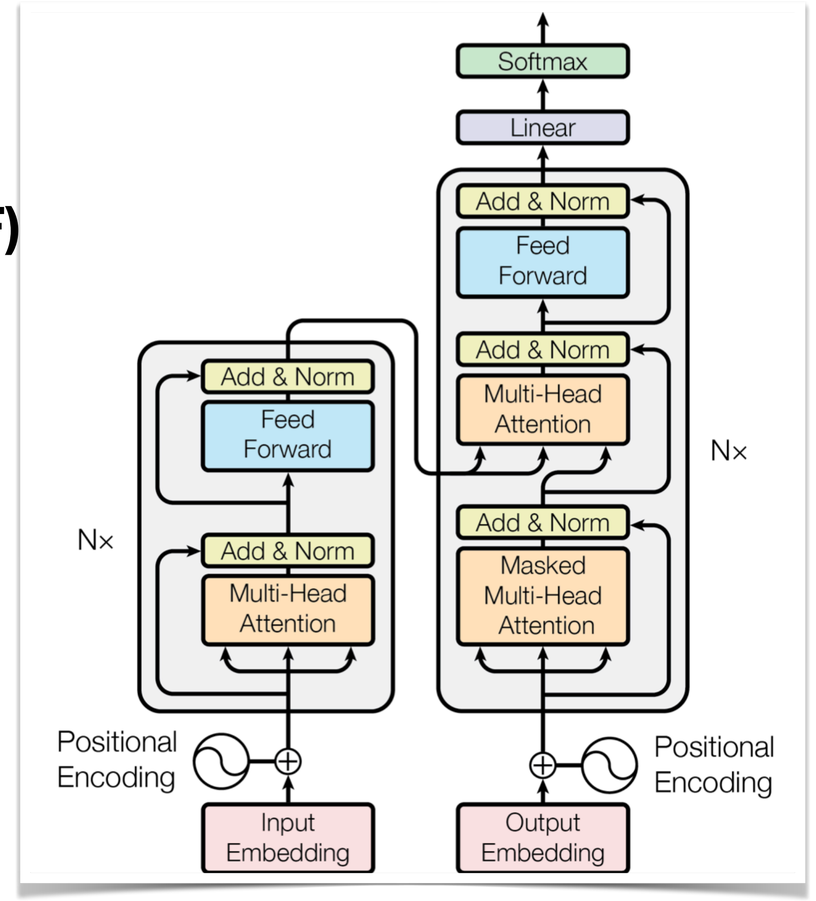


https://medium.com/mantisnlp/supervised-fine-tuning-customizing-llms-a2c1edbf22c3

Lecture Plan

Today, we will cover:

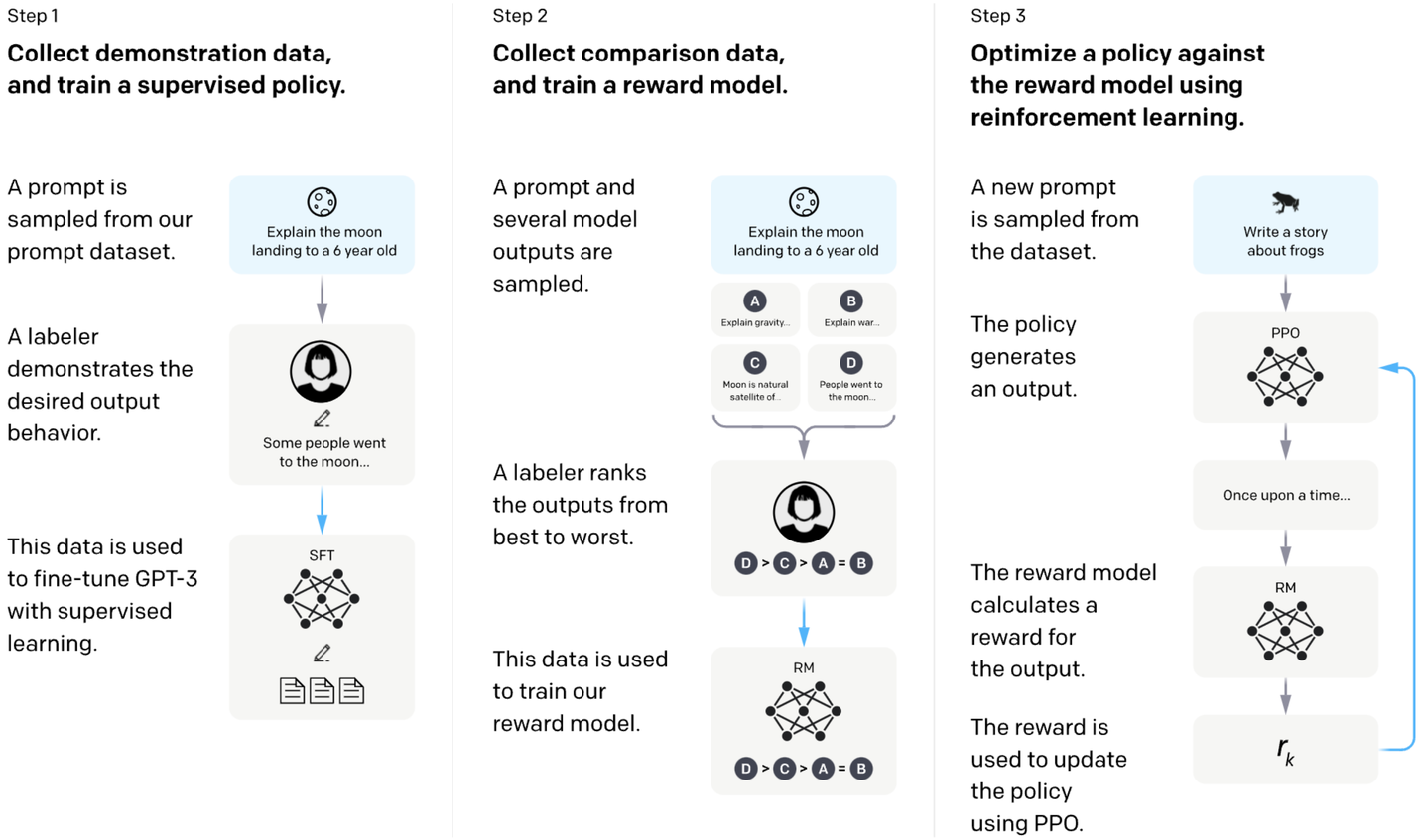
1. **LLM Fine-Tuning**
    1. Supervised Fine-Tuning (SFT)
    2. **Reinforcement Learning from Human Feedback (RLHF)**
    3. Parameter Efficient Fine-Tuning (PEFT)
        - BitFit, TinyTL, Adapter, Prompt-Tuning, Prefix-Tuning
        - LoRA, QLoRA, BitDelta
2. Multi-modal LLMs
    1. Cross-Attention Based: Flamingo
    2. Visual Tokens as Input: PaLM-E, VILA
    3. Enabling Visual Outputs: VILA-U
3. **Prompt Engineering**
    1. In-Context Learning (ICL)
    2. Chain-of-Thought (CoT)
    3. Retrieval Augmented Generation (RAG)



Reinforcement Learning from Human Feedback (RLHF)

Aligning AI with Human Preferences

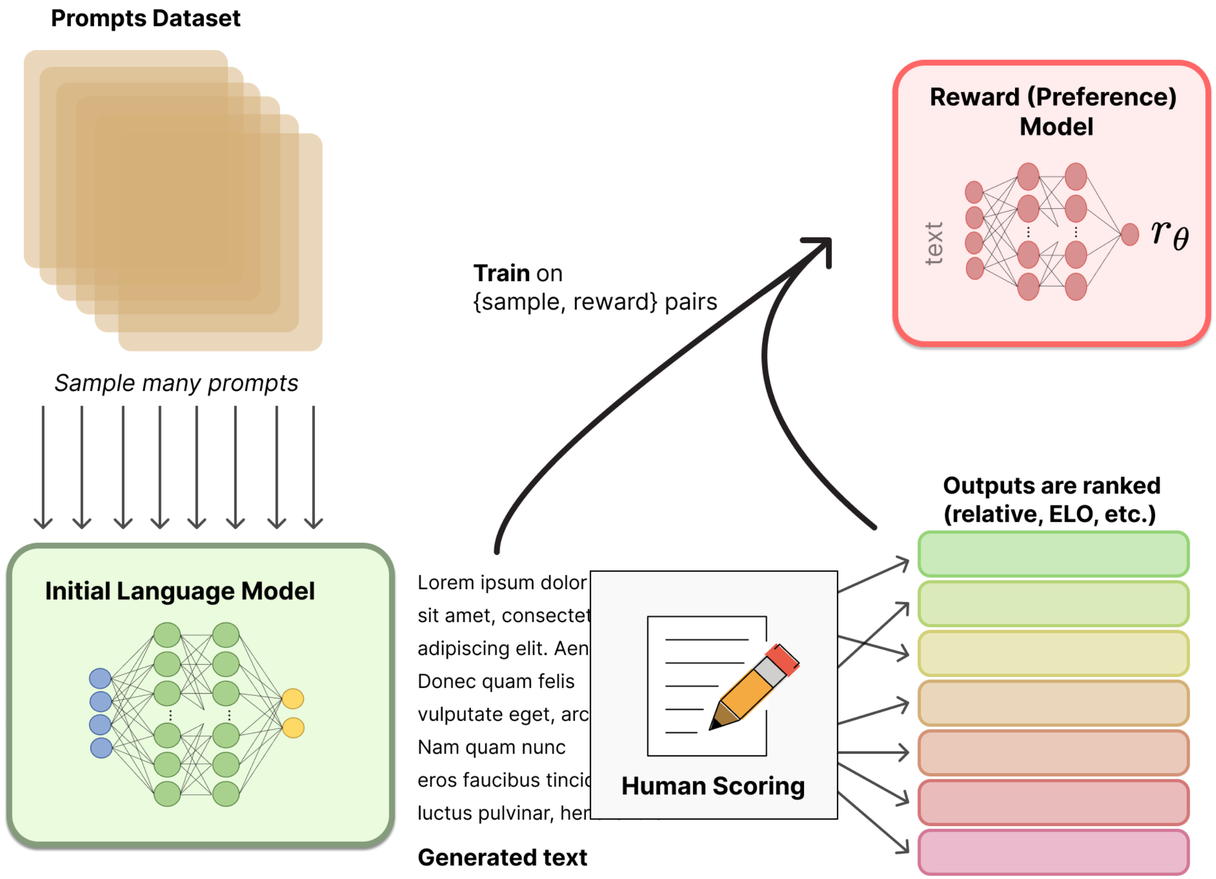
- **Why it matters?:** Traditional models rely on static metrics like BLEU or ROUGE, but RLHF enables models to optimize based on subjective, human-defined qualities such as creativity, truthfulness, or usefulness.
- **Key Success:** RLHF has been instrumental in improving systems like ChatGPT.



Training language models to follow instructions with human feedback [Ouyang et al, 2022]

Reinforcement Learning from Human Feedback (RLHF)

Training the Reward Model

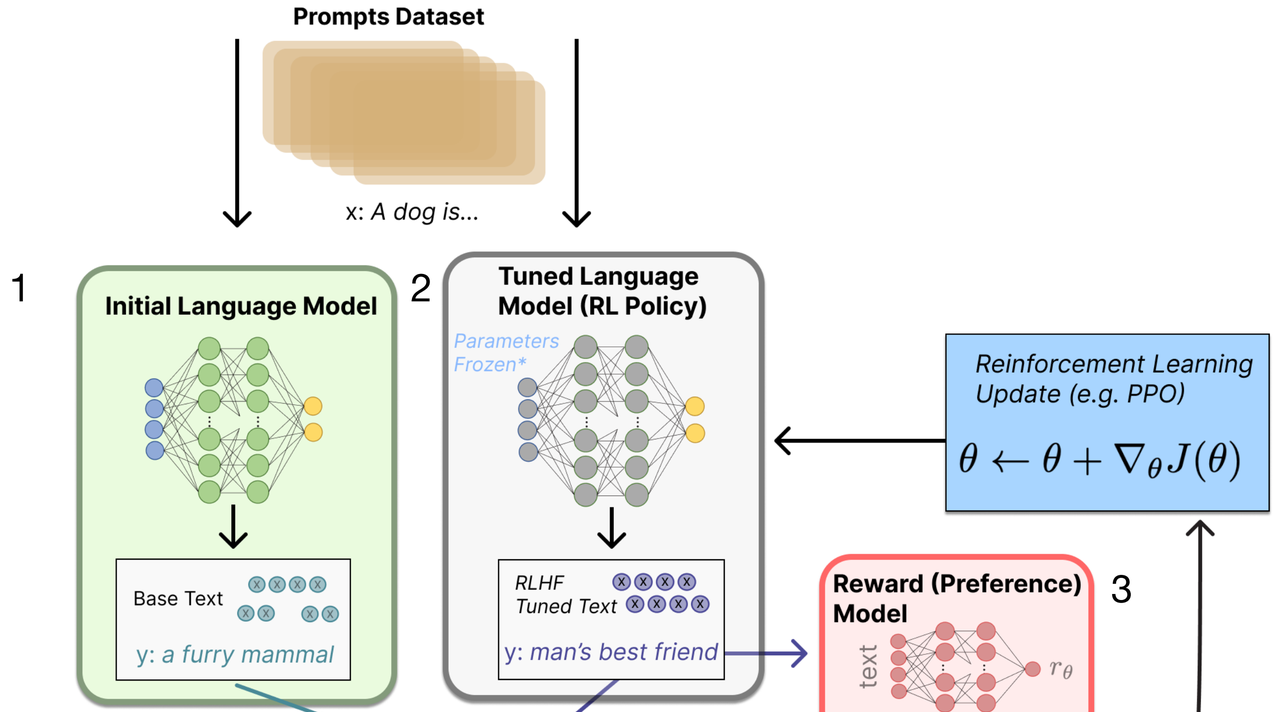


$$\max_{r_\theta} \left\{ \mathbb{E}_{(x,y_{\text{win}},y_{\text{lose}})\sim\mathcal{D}}[\log \sigma(r_\theta(x, y_{\text{win}}) - r_\theta(x, y_{\text{lose}}))] \right\}$$

https://huggingface.co/blog/rlhf

Reinforcement Learning from Human Feedback (RLHF)

Fine-tuning with Reinforcement Learning



The likelihood of the generation should not differ a lot from the reference model, making sure the outputs does not overfit the reward model.

Optimizing for the preference

$$\max_{\pi_\theta} \left\{ \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}[r_\theta(x,y)] - \beta \mathbb{D}_{\text{KL}}[\pi_\theta(y\,|\,x)\,||\,\pi_{\text{ref}}(y\,|\,x)] \right\}$$

https://huggingface.co/blog/rlhf

Direct Preference Optimization (DPO)

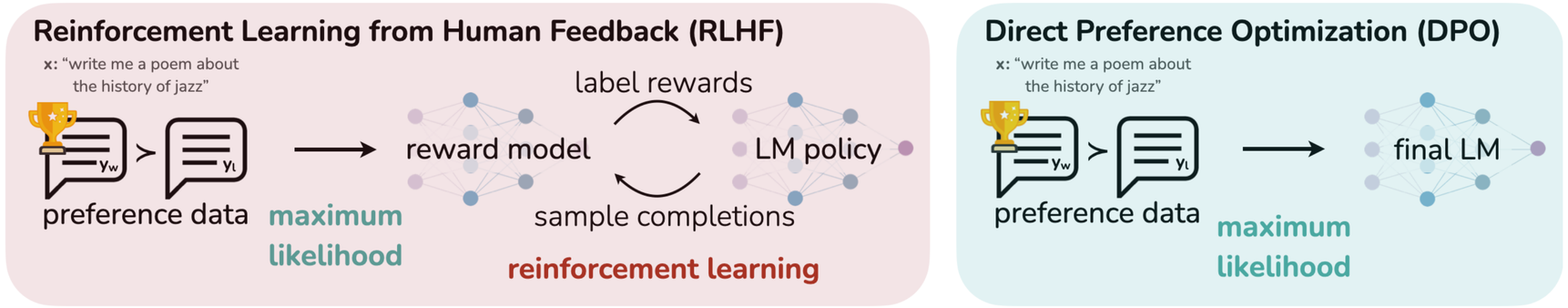
Simplifying RLHF by reducing the number of stages and models involved

- **Instead of using multiple models in two phases, DPO converts the problem into a single-phase Supervised Fine-Tuning (SFT) task.**
- **Key Insight:** DPO eliminates the need for reward models and direct reinforcement learning algorithms by using a straightforward optimization method to train the language model.

Computed from the fine-tuned model

$$\max_{\pi_\theta} \left\{ \mathbb{E}_{(x,y_{\text{win}},y_{\text{lose}})\sim\mathcal{D}} [\log \sigma(\beta \log \frac{\pi_\theta(y_{\text{win}}|x)}{\pi_{\text{ref}}(y_{\text{win}}|x)} - \beta \log \frac{\pi_\theta(y_{\text{lose}}|x)}{\pi_{\text{ref}}(y_{\text{lose}}|x)})] \right\}$$

Can be pre-computed offline



Involves three model (reward model, reference model, and the fine-tuned model)

Only involves the fine-tuned model.

Direct preference optimization: Your language model is secretly a reward model [Rafailov et al., 2024]

Direct Preference Optimization (DPO)

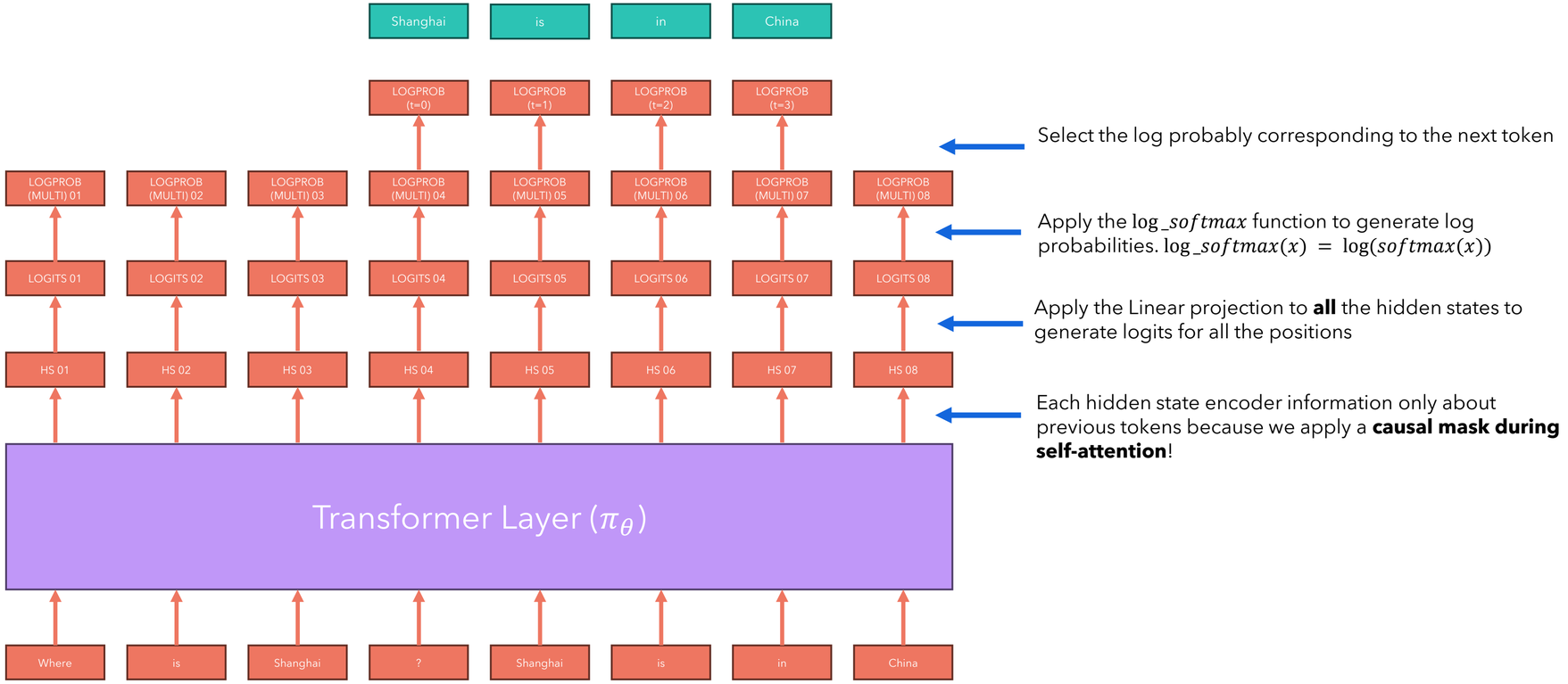
An example of DPO

- x = **"Where is Shanghai?"**
- y_{win} = **"Shanghai is a city in China"**
- y_{lose} = **"Shanghai does not exist"**

Computed from the fine-tuned model

$$\max_{\pi_\theta} \left\{ \mathbb{E}_{(x,y_{\text{win}},y_{\text{lose}})\sim\mathcal{D}} [\log \sigma(\beta \log \frac{\pi_\theta(y_{\text{win}}|x)}{\pi_{\text{ref}}(y_{\text{win}}|x)} - \beta \log \frac{\pi_\theta(y_{\text{lose}}|x)}{\pi_{\text{ref}}(y_{\text{lose}}|x)})] \right\}$$

Can be pre-computed offline

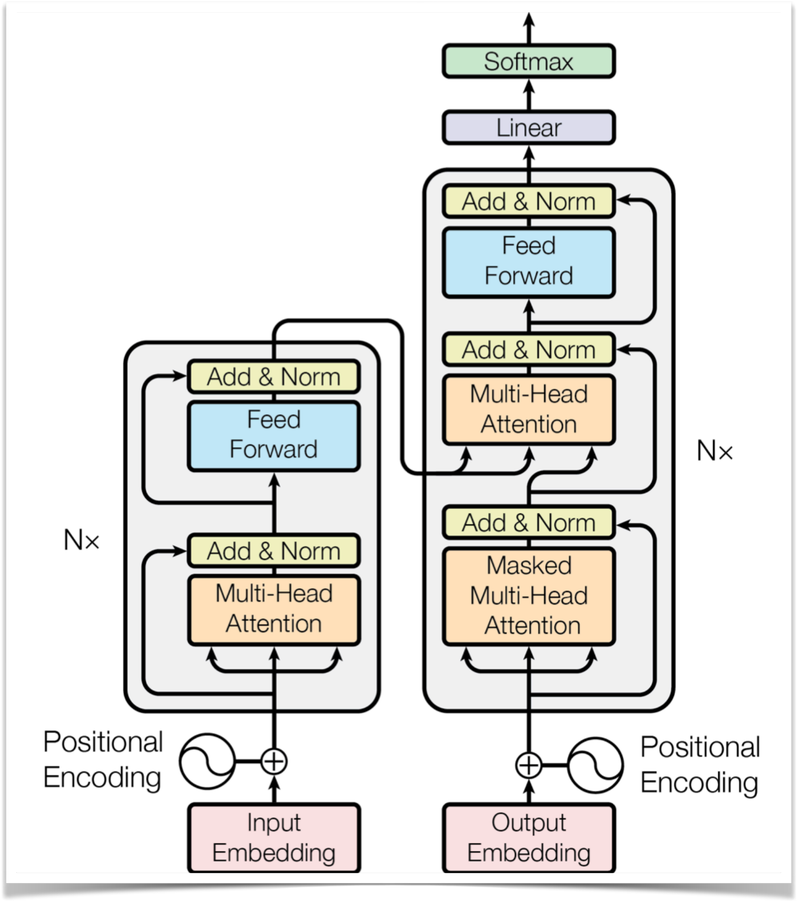


Umar Jamil – https://github.com/hkproj/dpo-notes

Lecture Plan

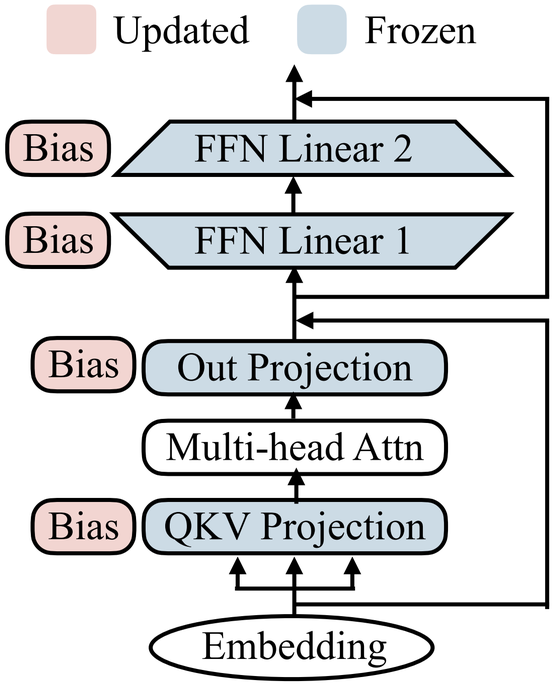
Today, we will cover:

1. **LLM Fine-Tuning**
    1. Supervised Fine-Tuning (SFT)
    2. Reinforcement Learning from Human Feedback (RLHF)
    3. **Parameter Efficient Fine-Tuning (PEFT)**
        - **BitFit, TinyTL, Adapter, Prompt-Tuning, Prefix-Tuning**
        - **LoRA, QLoRA, BitDelta**
2. Multi-modal LLMs
    1. Cross-Attention Based: Flamingo
    2. Visual Tokens as Input: PaLM-E, VILA
    3. Enabling Visual Outputs: VILA-U
3. **Prompt Engineering**
    1. In-Context Learning (ICL)
    2. Chain-of-Thought (CoT)
    3. Retrieval Augmented Generation (RAG)



BitFit

Fine-tune only the ***bias*** terms.

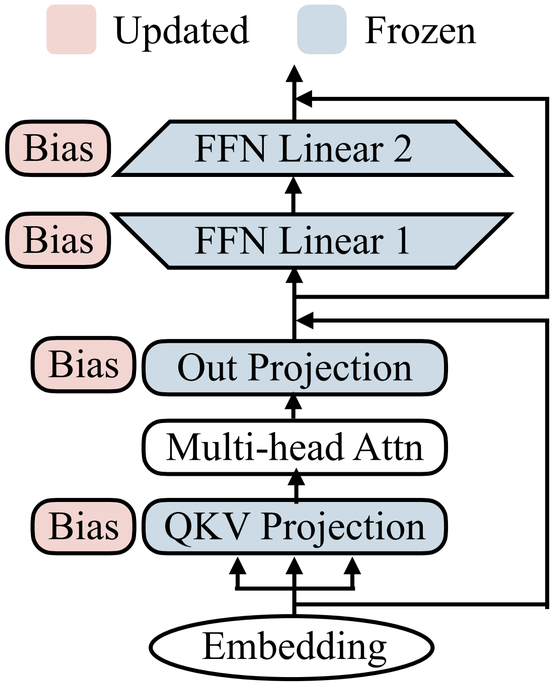


- A sparse fine-tuning method where only the bias of the model (or a subset of the bias) are being updated.
- BERT-base has 110M parameters, but only 0.1M biases (>1000x less).

BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models [Zeken et al, ACL 2021]

BitFit

Fine-tune only the ***bias*** terms.



- A sparse fine-tuning method where only the bias of the model (or a subset of the bias) are being updated.
- BERT-base has 110M parameters, but only 0.1M biases (>1000x less).
- From small-to-medium datasets, BitFit is competitive with (and sometimes better than) full-fine-tuning

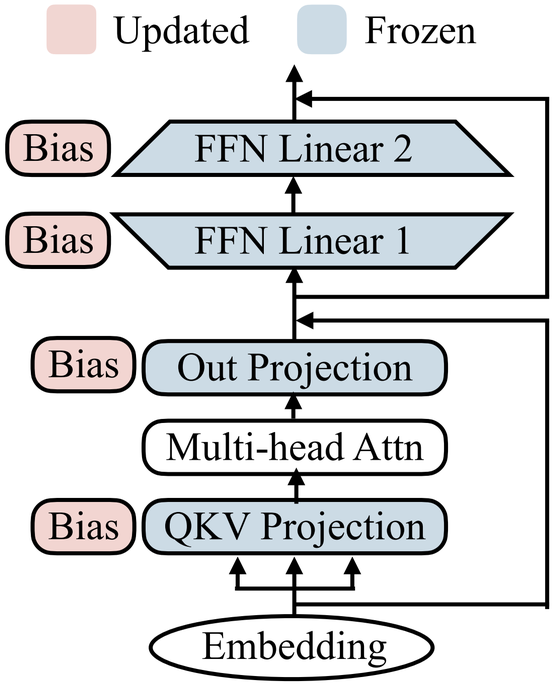
		%Param	QNLI	SST-2	MNLI_m	MNLI_mm	CoLA	MRPC	STS-B	RTE	QQP	Avg.
	Train size		105k	67k	393k	393k	8.5k	3.7k	7k	2.5k	364k	
(V)	Full-FT†	100%	**93.5**	**94.1**	**86.5**	**87.1**	**62.8**	**91.9**	89.8	71.8	**87.6**	**84.8**
(V)	Full-FT	100%	91.7±0.1	93.4±0.2	85.5±0.4	85.7±0.4	62.2±1.2	90.7±0.3	**90.0±0.4**	**71.9±1.3**	87.5±0.4	84.1
(V)	Diff-Prune†	0.5%	**93.4**	**94.2**	**86.4**	**86.9**	63.5	91.3	89.5	71.5	**86.6**	**84.6**
(V)	BitFit	0.08%	91.4±2.4	93.2±0.4	84.4±0.2	84.8±0.1	**63.6±0.7**	**91.7±0.5**	**90.3±0.1**	**73.2±3.7**	85.4±0.1	84.2
(T)	Full-FT‡	100%	91.1	**94.9**	86.7	85.9	**60.5**	**89.3**	**87.6**	70.1	**72.1**	**81.8**
(T)	Full-FT†	100%	**93.4**	94.1	86.7	**86.0**	59.6	88.9	86.6	**71.2**	71.7	81.5
(T)	Adapters‡	3.6%	90.7	94.0	84.9	85.1	59.5	89.5	**86.9**	71.5	**71.8**	81.1
(T)	Diff-Prune†	0.5%	**93.3**	94.1	**86.4**	**86.0**	**61.1**	**89.7**	86.0	70.6	71.1	**81.5**
(T)	BitFit	0.08%	92.0	**94.2**	84.5	84.8	59.7	88.9	85.5	**72.0**	70.5	80.9

Table 1: BERT_LARGE model performance on the GLUE benchmark validation set (V) and test set (T). Lines with † and ‡ indicate results taken from Guo et al. (2020) and Houlsby et al. (2019) (respectively).

BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models [Zeken et al, ACL 2021]

BitFit

Fine-tune only the ***bias*** terms.



- A sparse fine-tuning method where only the bias of the model (or a subset of the bias) are being updated.
- BERT-base has 110M parameters, but only 0.1M biases (>1000x less).
- From small-to-medium datasets, BitFit is competitive with (and sometimes better than) full-fine-tuning.
- For larger data, the method shows inferior performance than full.

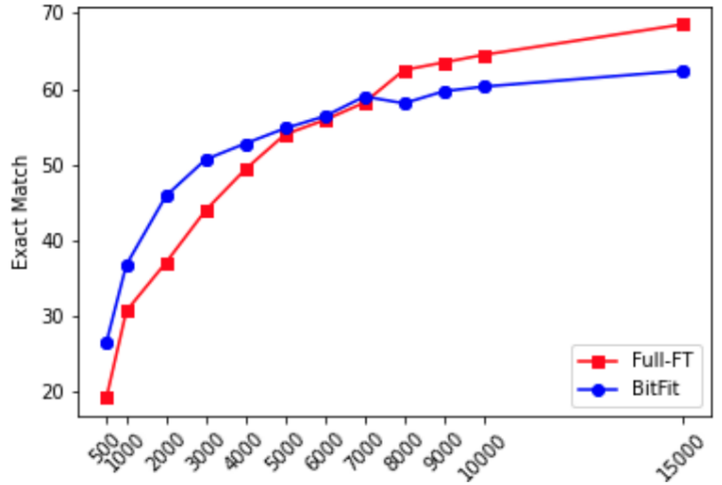
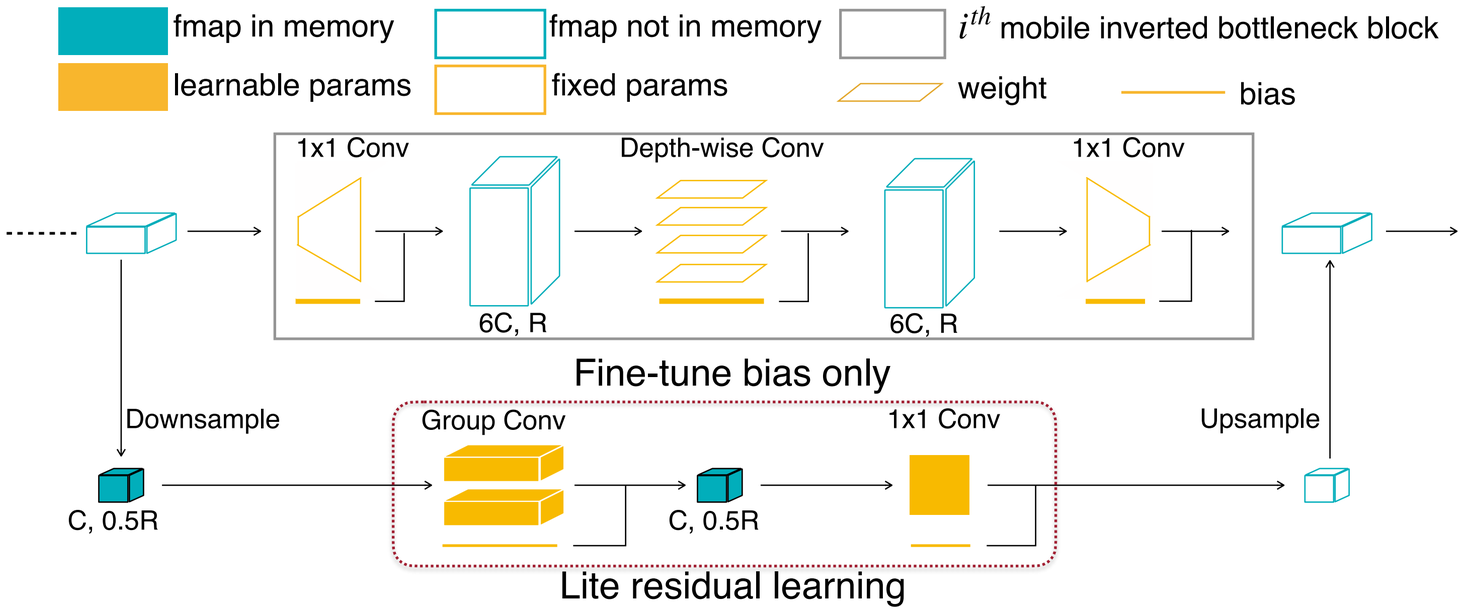


Figure 2: Comparison of BitFit and Full-FT with $\text{BERT}_{\text{BASE}}$ exact match score on SQuAD validation set.

BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models [Zeken et al, ACL 2021]

TinyTL: Lite Residual Learning



- Add lite residual modules to increase model capacity
- Key principle - keep activation size small
  1. Reduce the resolution
  2. Avoid inverted bottleneck

TinyTL: Reduce Activations, Not Trainable Parameters for Efficient On-Device Learning [Cai *et al.*, NeurIPS 2020]

Adapter

Inserting learnable layers inside transformer architectures

- Add only a few trainable parameters ***per task***, and new tasks can be added without revisiting previous ones.
  - Conventionally, for N down-stream tasks, we will need N copies of model weights (significant **storage overhead**!)
  - FT-Full: 1000 sub-tasks x 7B llama => 14 PB storage
  - Adapter: 1000 sub-tasks x 14 MB => 14 GB storage

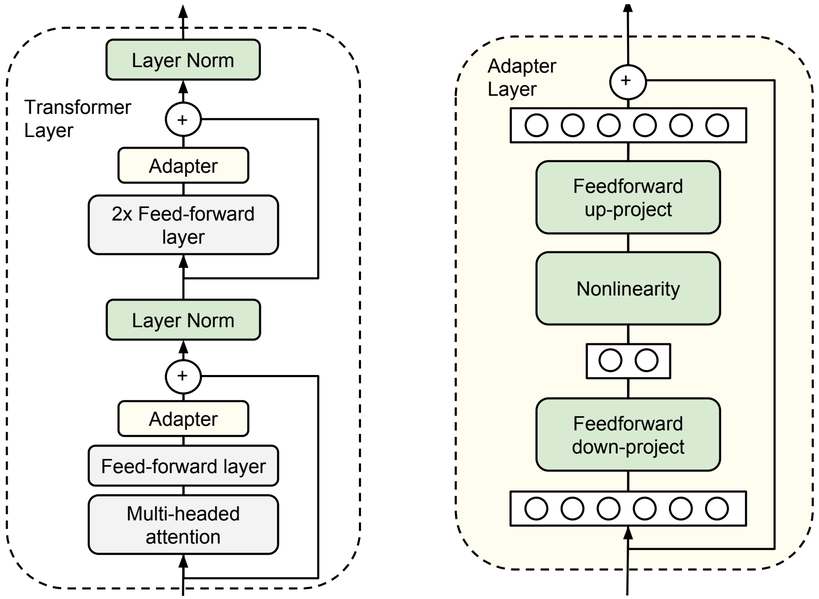


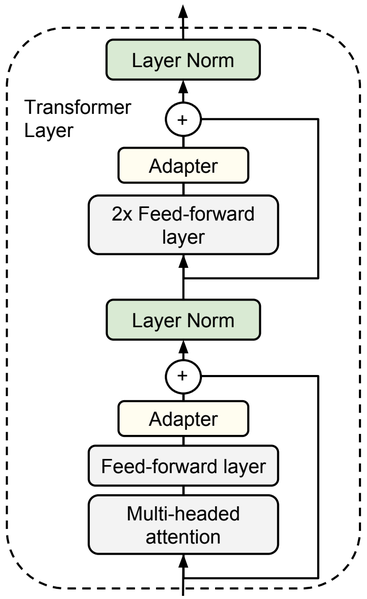
Figure generated by Dall-E:
tons of hard disk drive in a data center

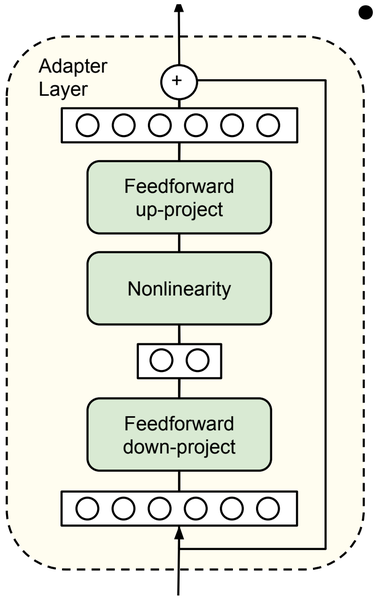
Parameter-Efficient Transfer Learning for NLP [Houlsby et al, ICML 2019]

Adapter

Inserting learnable layers inside transformer architectures

- Add only a few trainable parameters ***per task***, and new tasks can be added without revisiting previous ones.
- Adapters attain near state-of-the-art performance, whilst adding only a few parameters per task.





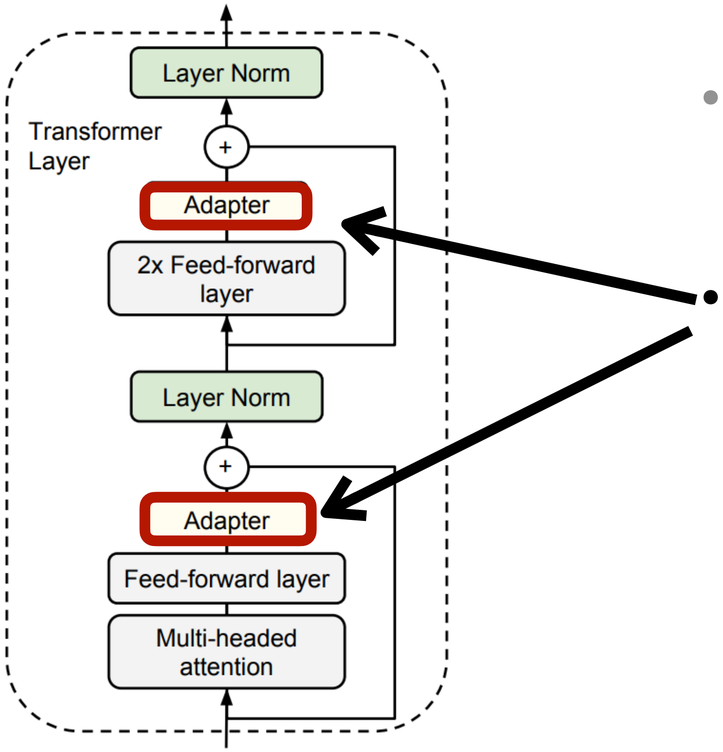
	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_m	MNLI_mm	QNLI	RTE	Total
BERT_LARGE	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

Hidden dimension of the bottleneck

Parameter-Efficient Transfer Learning for NLP [Houlsby et al, ICML 2019]

Adapter

Inserting learnable layers inside transformer architectures



- Add only a few trainable parameters ***per task***, and new tasks can be added without revisiting previous ones.
- Adapters attain near state-of-the-art performance, whilst adding only a few parameters per task.
- But, these adapter modules brings extra inference latency during deployment[2].

Batch Size Sequence Length $\|\Theta\|$	32 512 0.5M	16 256 11M	1 128 11M
$\text{Adapter}^{\text{L}}$	1482.0±1.0 (+2.2%)	354.8±0.5 (+5.0%)	23.9±2.1 (+20.7%)
$\text{Adapter}^{\text{H}}$	1492.2±1.0 (+3.0%)	366.3±0.5 (+8.4%)	25.8±2.2 (+30.3%)

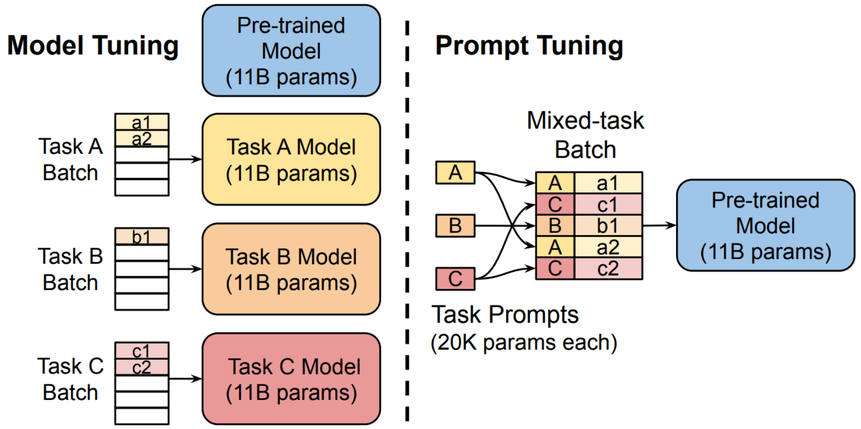
Inference latency of a single forward pass in GPT-2 medium averaged over 100 trials. Results are based on NVIDIA Quadro RTX8000

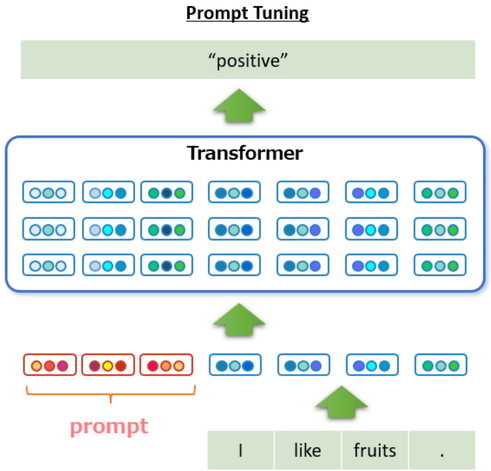
How to improve the inference overhead?

[1] Parameter-Efficient Transfer Learning for NLP [Houlsby et al, ICML 2019]
[2] LoRA: Low-Rank Adaptation of Large Language Models [Hu et al, ICLR 2022]

Prompt Tuning

From discrete prompt to continuous prompt



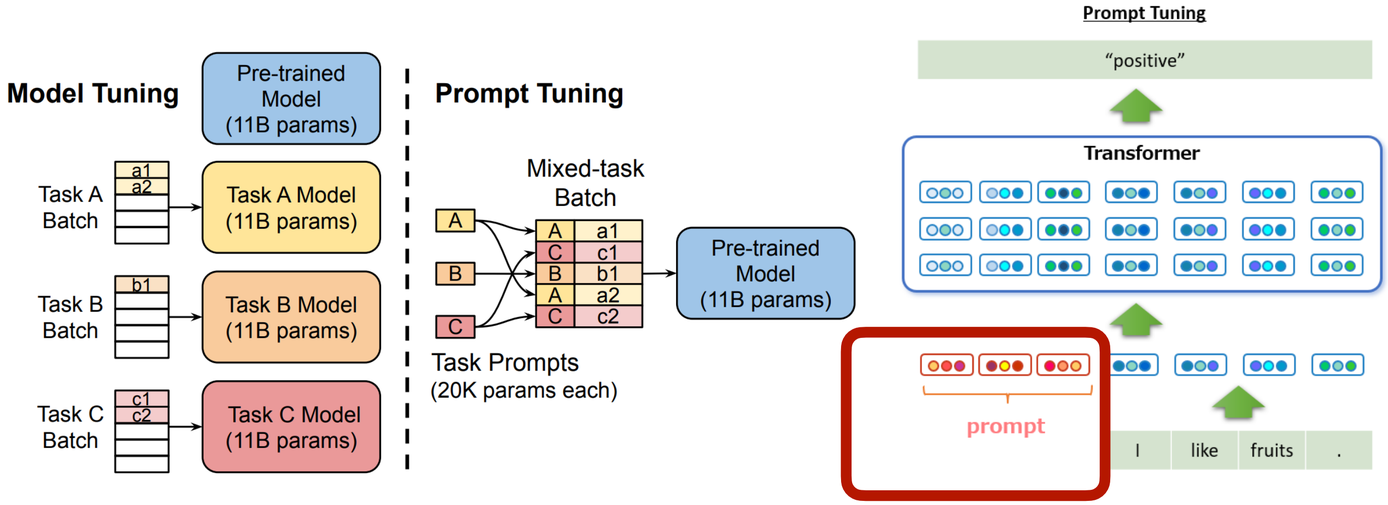


- Prompt engineering can help LLM work on different downstream applications
  - E.g., "Please tell me the sentiment of the following text: "

The Power of Scale for Parameter-Efficient Prompt Tuning [Lester, ACL 2021]

Prompt Tuning

From discrete prompt to continuous prompt



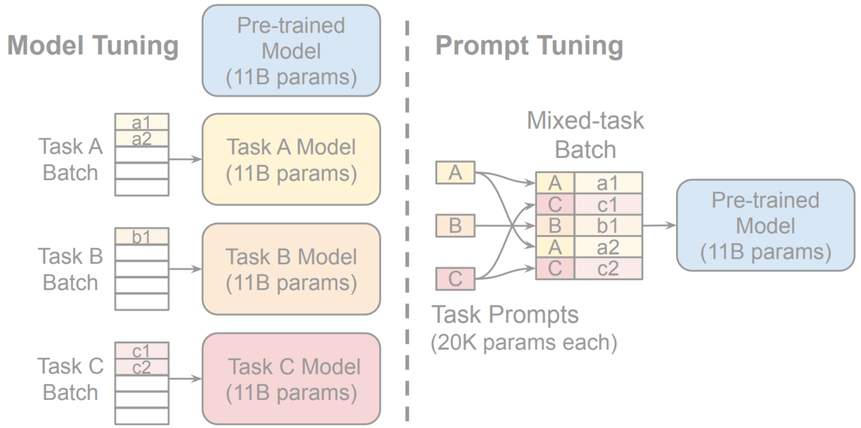
- Prompt engineering can help LLM work on different downstream applications
  - E.g., "Please tell me the sentiment of the following text: "

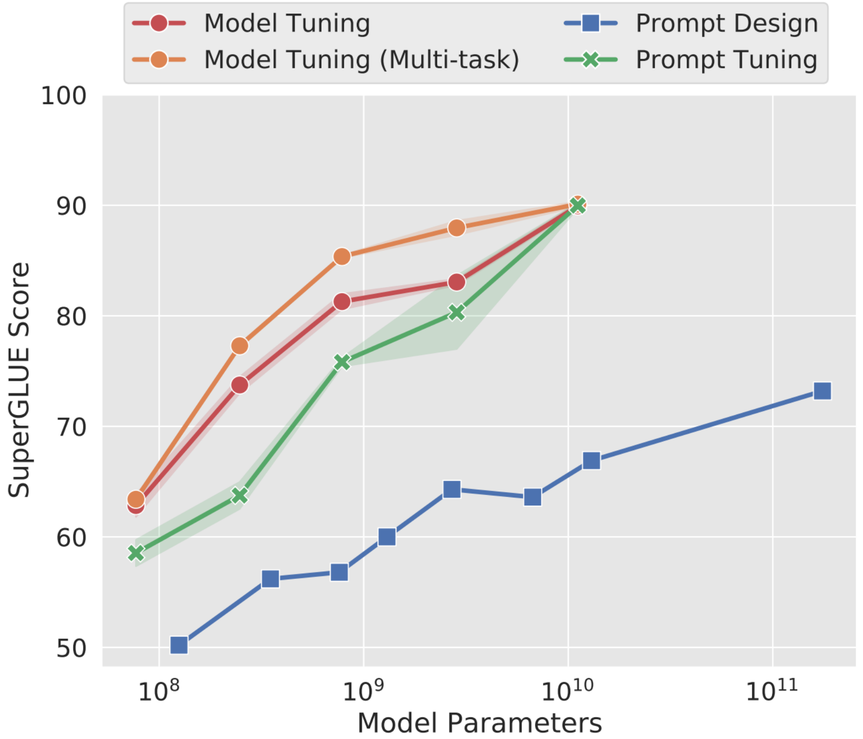
- **We can train a continuous prompt that is prepended to inputs for each task**
- **We can mix different learned prompts in a single batch**

The Power of Scale for Parameter-Efficient Prompt Tuning [Lester, ACL 2021]

Prompt Tuning

From discrete prompt to continuous prompt



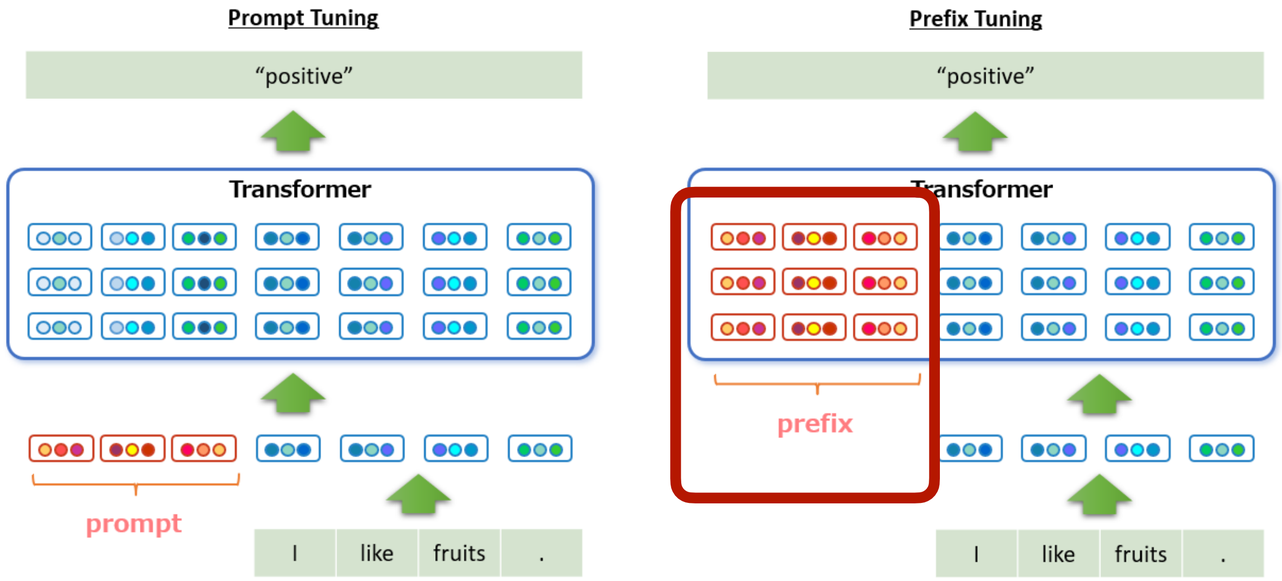


- Prompt engineering can help LLM work on different downstream applications
  - E.g., “Please tell me the sentiment of the following text: ”
- We can train a continuous prompt that is prepended to inputs for each task
- We can mix different learned prompts in a single batch
- **Comparable accuracy as fine-tuning as the model gets larger.**

The Power of Scale for Parameter-Efficient Prompt Tuning [Lester, ACL 2021]

Prefix-Tuning

Prepend prefixes for each input.



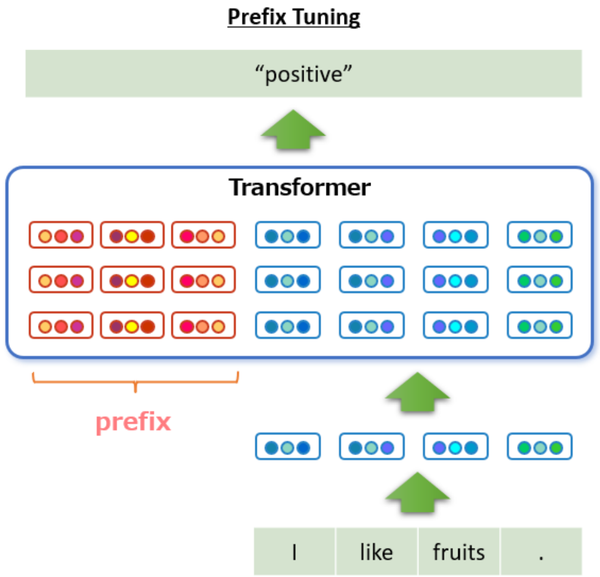
- Prompt-Tuning only adds learnable prompts to the **first layer.**
- Prefix-Tuning adds tunable prompts to **each layer.**

https://www.ogis-ri.co.jp/otc/hiroba/technical/similar-document-search/part28.html
Prefix-Tuning: Optimizing Continuous Prompts for Generation [Li et al, ACL 2021]

Prefix-Tuning

Prepend prefixes for each input.

		E2E			
	BLEU	NIST	MET	ROUGE	CIDEr
Prefix-Tuning					
PREFIX	69.7	8.81	46.1	71.4	2.49
Prompt-Tuning	Embedding-only: EMB-{PrefixLength}				
EMB-1	48.1	3.33	32.1	60.2	1.10
EMB-10	62.2	6.70	38.6	66.4	1.75
EMB-20	61.9	7.11	39.3	65.6	1.85

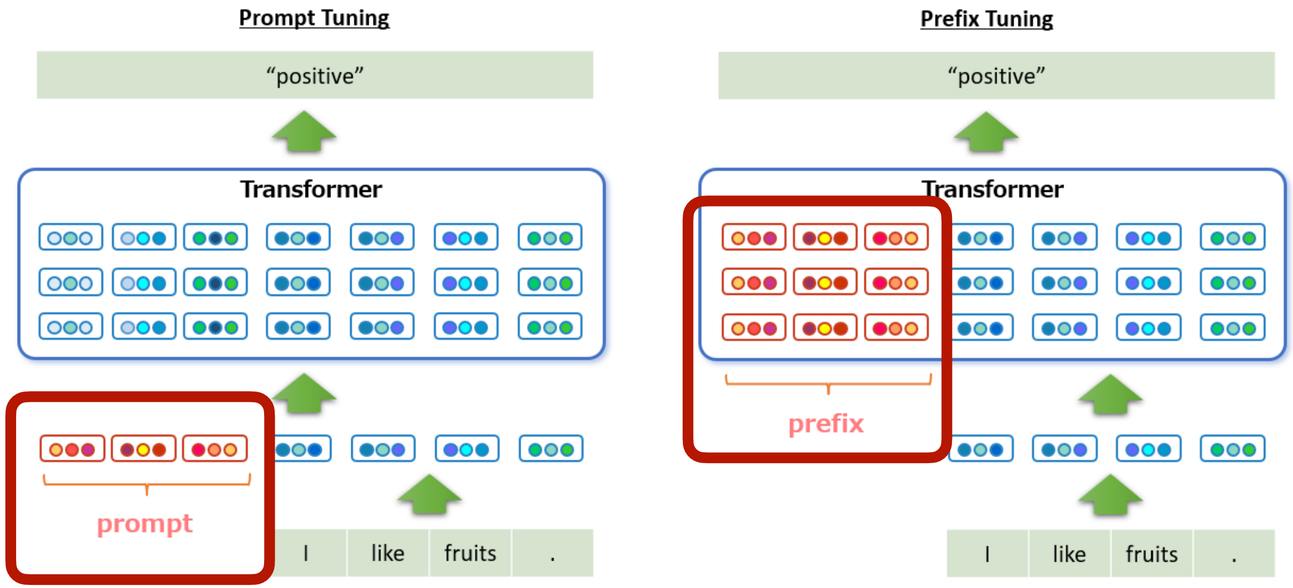


- Prompt-Tuning only adds learnable prompts to the first layer.
- Prefix-Tuning adds tunable prompts to each layer.
- **Prefix-tuning shows consistent improvement of embedding-only-tuning.**

https://www.ogis-ri.co.jp/otc/hiroba/technical/similar-document-search/part28.html
Prefix-Tuning: Optimizing Continuous Prompts for Generation [Li et al, ACL 2021]

Parameter-Efficient Fine-Tuning

Disadvantage of Prompt- / Prefix-Tuning



- Both methods increase input length
  - Lead to longer inference latency.
  - Take up available input length and limit the real usable sequence.

Question: Can we fine-tune without incurring extra inference latency?

https://www.ogis-ri.co.jp/otc/hiroba/technical/similar-document-search/part28.html
Prefix-Tuning: Optimizing Continuous Prompts for Generation [Li et al, ACL 2021]

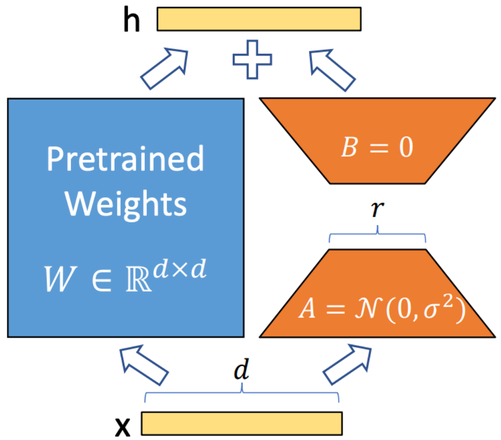
Lecture Plan

- PockEngine: System Support for Sparse Back-Propagation
- Efficient LLM Fine-Tuning
  - BitFit / Adapter / Prompt-Tuning / Prefix-Tuning
  - **LoRA / QLoRA / LongLoRA**
- Prompt Engineering
  - Zero-Shot / Few-Shot / Chain-of-Thoughts
  - LLM / Diffusion Prompting Examples

LoRA

Parallel and fusable learnable layers

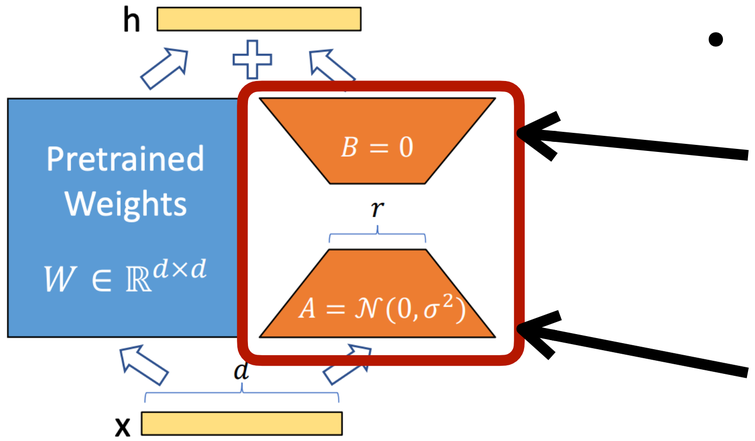
- Inject trainable rank decomposition matrices into each layer (as a parallel branch).



LoRA: Low-Rank Adaptation of Large Language Models [Hu et al, ICLR 2022]

LoRA

Parallel and fusable learnable layers

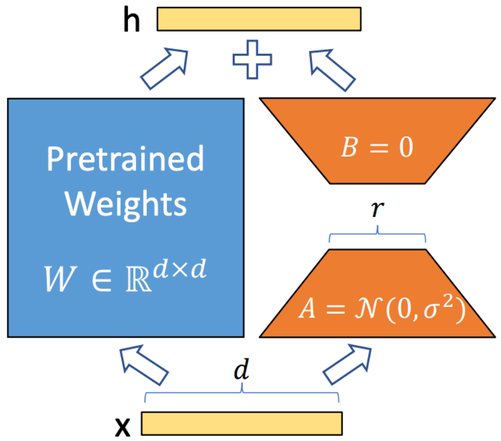


- Inject trainable rank decomposition matrices into each layer (as a parallel branch).
- The LoRA module have two parts
  - A: projects d dim to low-rank r dim, initialized with gaussian distribution.
  - B: projects low-rank r dim back to d dim, initialized 0s.

LoRA: Low-Rank Adaptation of Large Language Models [Hu et al, ICLR 2022]

LoRA

Parallel and fusable learnable layers



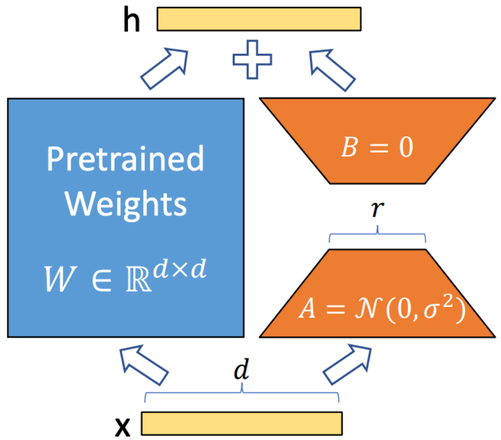
- Inject trainable rank decomposition matrices into each layer (as a parallel branch).
- The LoRA module have two parts
  - A: projects d dim to low-rank r dim, initialized with gaussian distribution.
  - B: projects low-rank r dim back to d dim, initialized 0s.
  - **The output is not affected**

$$h = x @ W + \underbrace{x @ A @ B}_{0}$$

LoRA: Low-Rank Adaptation of Large Language Models [Hu et al, ICLR 2022]

LoRA

Parallel and fusable learnable layers



- Inject trainable rank decomposition matrices into each layer (as a parallel branch).
- The LoRA module have two parts
  - A: projects d dim to low-rank r dim, initialized with gaussian distribution.
  - B: projects low-rank r dim back to d dim, initialized 0s.
  - The output is not affected and **A / B can be fused later**

$$h = x@W + x@A@B = x@(W + A@B) = x@W'$$

Thus **no extra inference** latency!

LoRA: Low-Rank Adaptation of Large Language Models [Hu et al, ICLR 2022]

LoRA Practical Cases

Prompt: white background, scenery, ink, mountains, water, trees

Original Diffusion

stable-diffusion-v1-5

Diffusion with LoRA

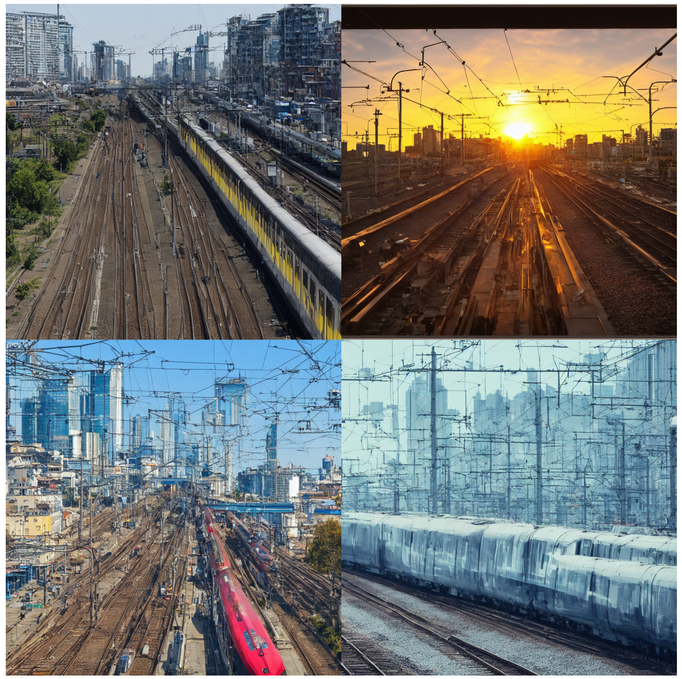
Model: stable-diffusion-v1-5

LoRA: Ink scenery

https://civitai.com/models/78605?modelVersionId=83390

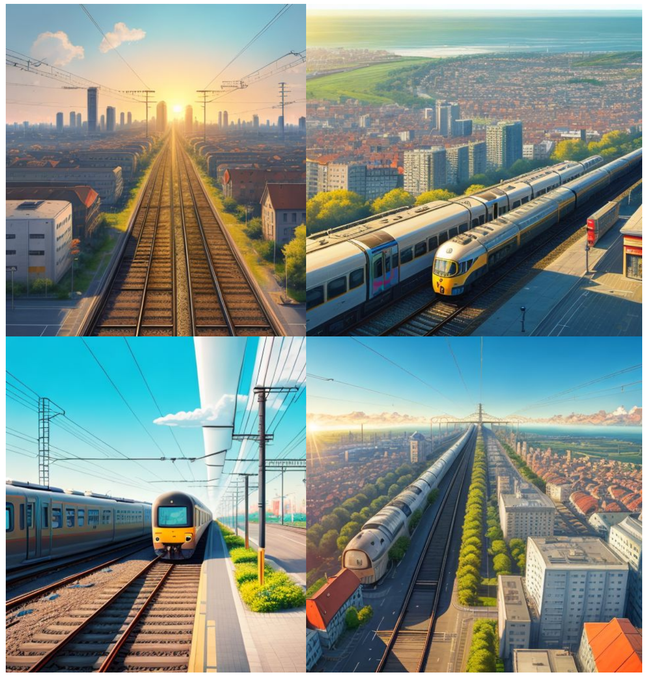
LoRA Practical Cases

Prompt: modern city, sunrise, railway



Original Diffusion

Model: stable-diffusion-v1-5



Diffusion with LoRA

Model: stable-diffusion-v1-5

LoRA: Detail Tweaker LoRA

https://civitai.com/models/58390/detail-tweaker-lora-lora

LoRA Practical Cases

Prompt: Sushi rolls in the form of panda

Original Diffusion

Model: stable-diffusion-v1-5

Diffusion with LoRA

Model: stable-diffusion-v1-5
LoRA: ZAVYCHROMAXL

https://civitai.com/models/119229?modelVersionId=169740

LoRA Practical Cases

Prompt: Sushi rolls in the form of husky

Original Diffusion

Model: stable-diffusion-v1-5

Diffusion with LoRA

Model: stable-diffusion-v1-5

LoRA: ZAVYCHROMAXL

https://civitai.com/models/119229?modelVersionId=169740

LoRA Practical Cases

Prompt: Piazza in the form of husky

Original Diffusion

Model: stable-diffusion-v1-5

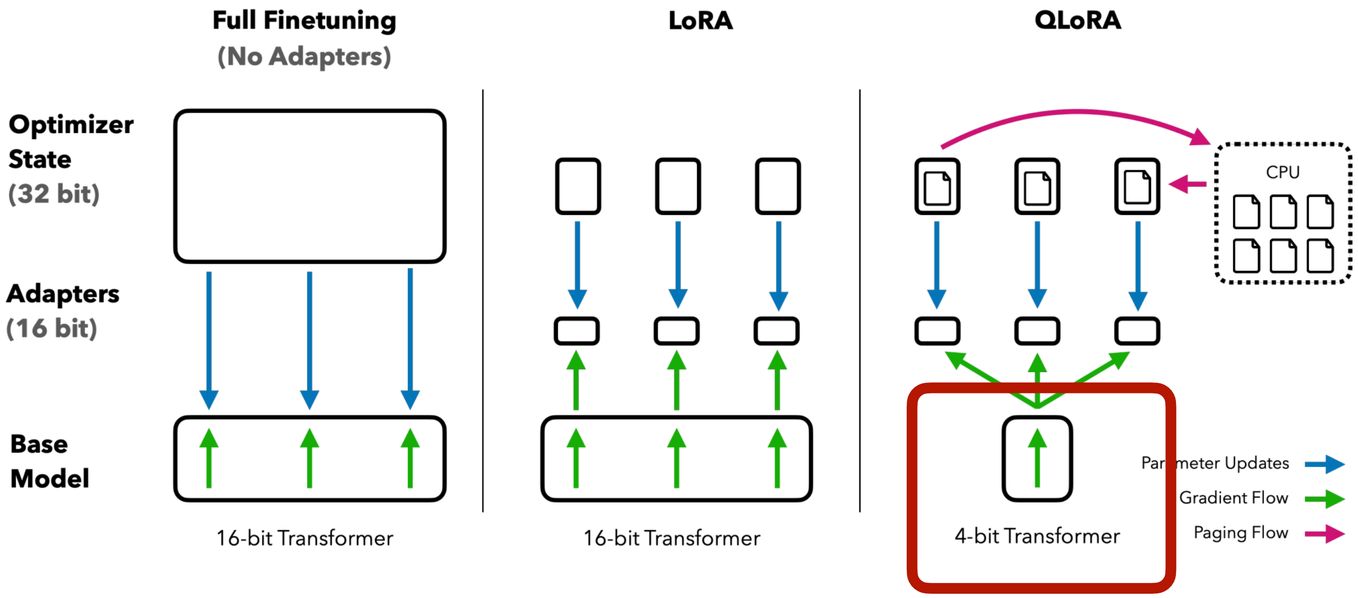
Diffusion with LoRA

Model: stable-diffusion-v1-5
LoRA: ZAVYCHROMAXL

https://civitai.com/models/119229?modelVersionId=169740

QLoRA

LoRA, with quantized backbones and paged optimizer states

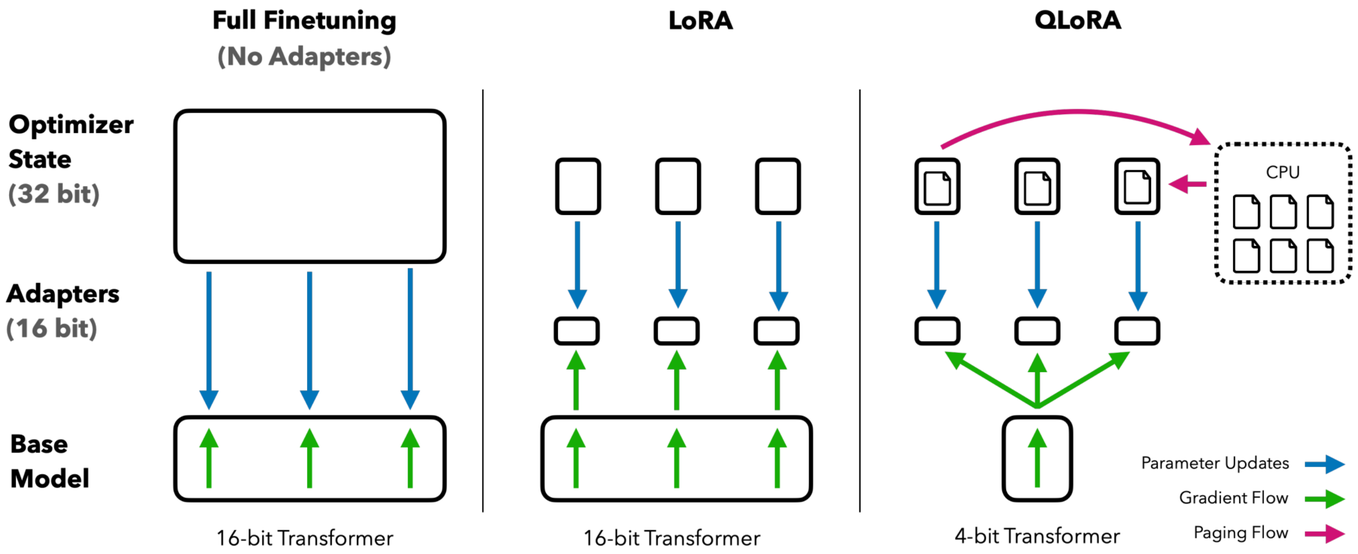


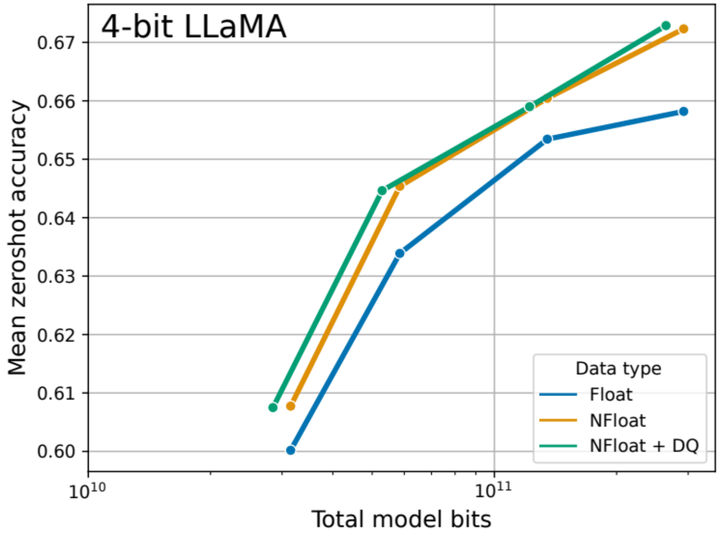
- **Following LoRA's design principle, but quantize the backbone models.**
  - NormalFloat (NF4), a new data type for LLM weight quantization
  - Double quantization (scaling factors also quantized) to further reduce
- Paged Optimizers with CPU offloading

QLoRA: Efficient Finetuning of Quantized LLMs [Dettmers et al, NeurIPS 2023]

QLoRA

LoRA, with quantized backbones and paged optimizer states





NF4 quantization demonstrates higher zero-shot accuracy.

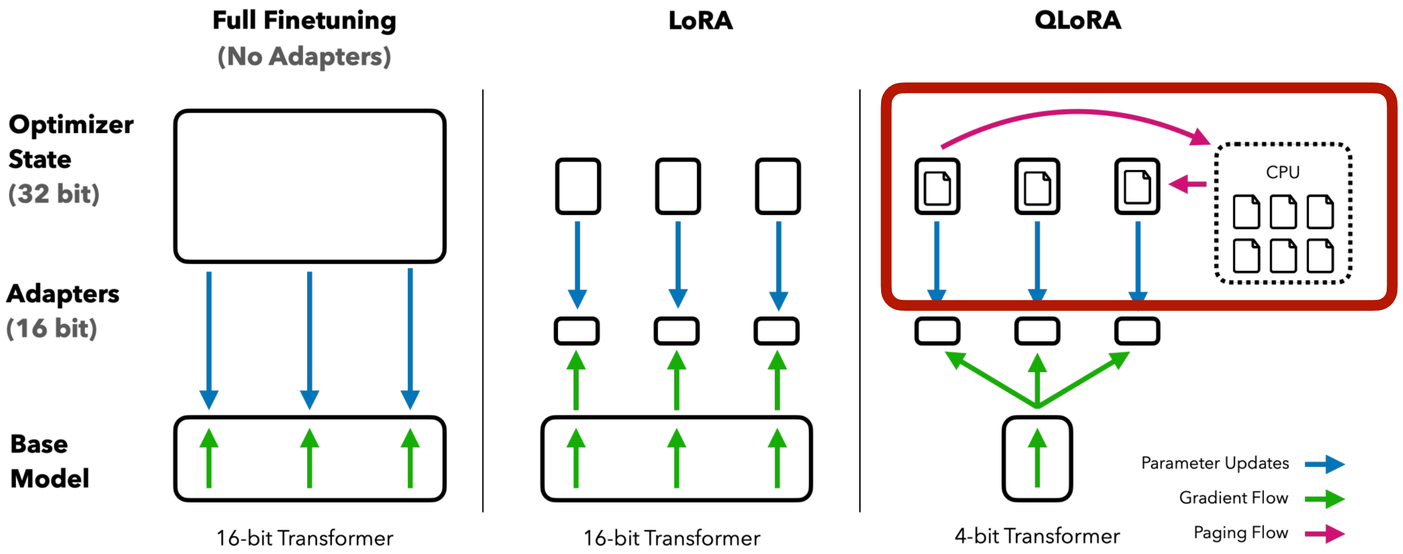
- Following LoRA's design principle, but quantize the backbone models.
  - **NormalFloat (NF4)**, a new data type for LLM weight quantization
  - Double quantization (scaling factors also quantized) to further reduce
- Paged Optimizers with CPU offloading

The exact values of the NF4 data type:
[-1.0, -0.6961928009986877,
-0.5250730514526367, -0.39491748809814453,
-0.28444138169288635, -0.18477343022823334,
-0.09105003625154495, 0.0,
0.07958029955625534, 0.16093020141124725,
0.24611230194568634, 0.33791524171829224,
0.44070982933044434, 0.5626170039176941,
0.7229568362236023, 1.0]

QLoRA: Efficient Finetuning of Quantized LLMs [Dettmers et al, NeurIPS 2023]

QLoRA

LoRA, with quantized backbones and paged optimizer states

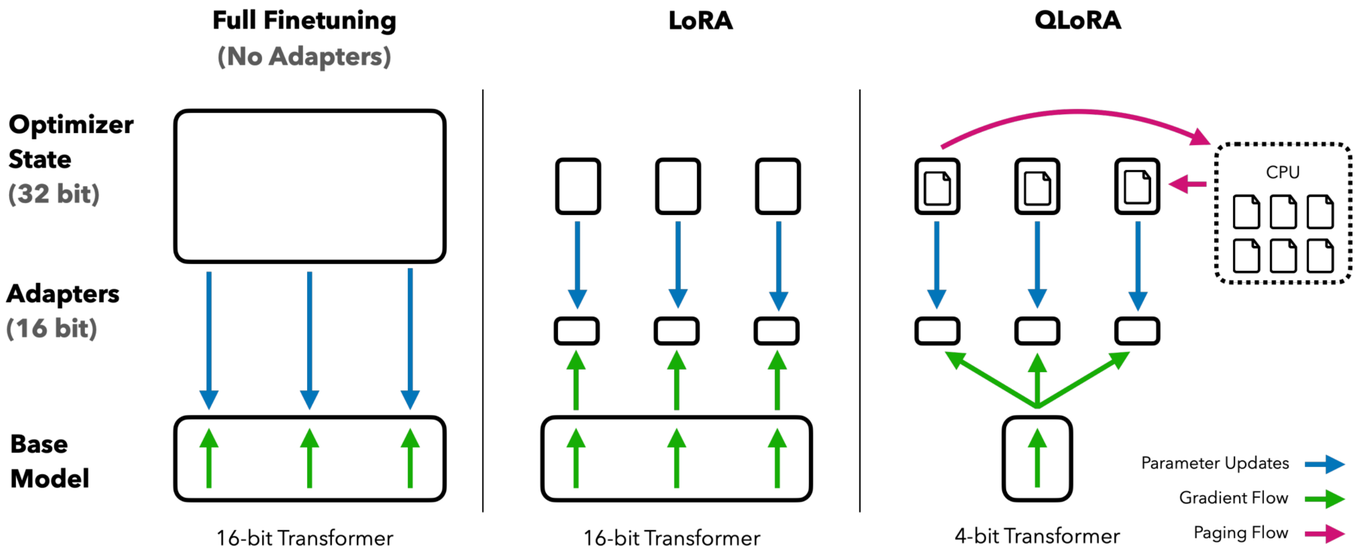


- Following LoRA's design principle, but quantize the backbone models.
  - NormalFloat (NF4), a new data type for LLM weight quantization
  - Double quantization (scaling factors also quantized) to further reduce
- **Paged Optimizers with CPU offloading**

QLoRA: Efficient Finetuning of Quantized LLMs [Dettmers et al, NeurIPS 2023]

QLoRA

LoRA, with quantized backbones and paged optimizer states



Model	Size	Elo
GPT-4	-	1348 ± 1
Guanaco 65B	41 GB	1022 ± 1
Guanaco 33B	21 GB	992 ± 1
Vicuna 13B	26 GB	974 ± 1
ChatGPT	-	966 ± 1
Guanaco 13B	10 GB	916 ± 1
Bard	-	902 ± 1
Guanaco 7B	6 GB	879 ± 1

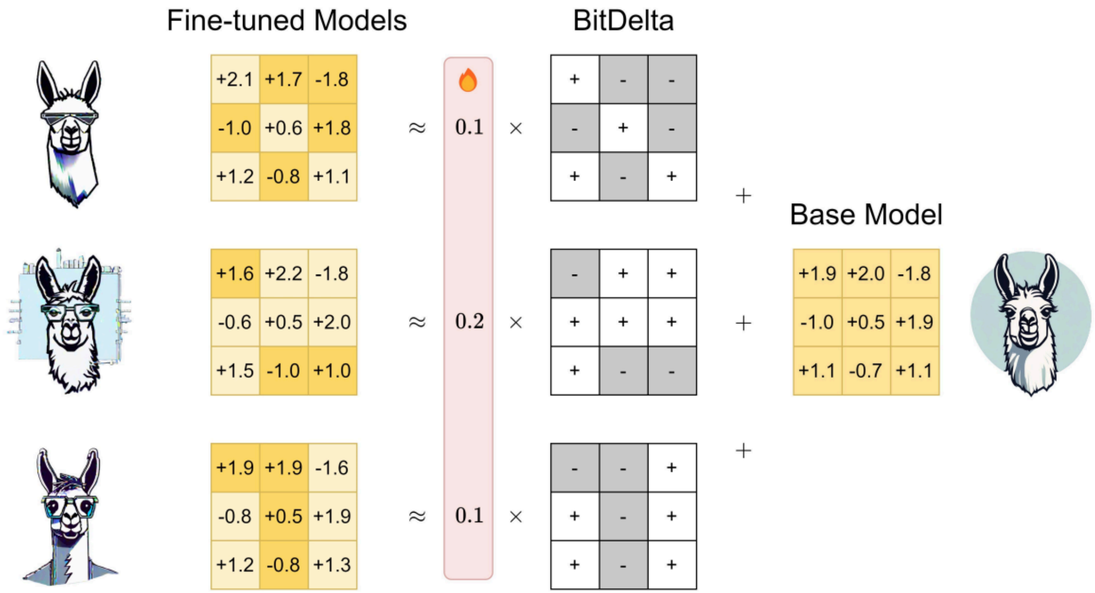
- Following LoRA's design principle, but quantize the backbone models.
  - NormalFloat (NF4), a new data type for LLM weight quantization
  - Double quantization (scaling factors also quantized) to further reduce
- Paged Optimizers with CPU offloading
- **Enable fine-tuning LLMs on middle / entry-level GPUs.**

QLoRA: Efficient Finetuning of Quantized LLMs [Dettmers et al, NeurIPS 2023]

Bit-Delta

Your Fine-Tune May Only Be Worth One Bit

- **Goal: efficient LLM finetuning with low precision**
- **Intuition: fine-tuning adds less new information to the model, and is thus more compressible.**
- **Our Solution: quantizes the weight delta down to 1 bit without compromising performance, finetuning the scaling factor (per tensor)**



$$\#\text{params} \times \#\text{models} \times 16\text{bits} \quad \Rightarrow \quad \#\text{params} \times (\#\text{models} \times 1\text{bit} + 16\text{bits})$$

- Weight delta: $\Delta = W_{\text{fine}} - W_{\text{base}}$
- Binarized delta: $\hat{\Delta} = \alpha \odot \text{Sign}(\Delta)$

$$\text{Sign}(W_{ij}) = \begin{cases} +1, & \text{if } W_{ij} > 0, \\ -1, & \text{if } W_{ij} \leq 0, \end{cases}$$

- To minimize the L_2 quantization error:

$$\left\| \Delta - \hat{\Delta} \right\|_2^2 = \sum_{ij} (|W_{ij}| - \alpha)^2$$

- We initialize α as

$$\alpha = \frac{1}{nm} \sum_{ij} |W_{ij}|.$$

- We further optimize the scales by performing model distillation:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \mathbb{E}_{x \sim \mathbf{X}} \left[\left\| \mathbf{Z}_{\text{fine}}(x) - \mathbf{Z}_{\text{bin}}(x; \boldsymbol{\alpha}) \right\|^2 \right]$$

- We distill on the C4 dataset, using 800 samples of length 128. For 70B models, the distillation roughly takes 10 minutes.

BitDelta: Your Fine-Tune May Only Be Worth One Bit [Liu et al., NeurIPS 2024]

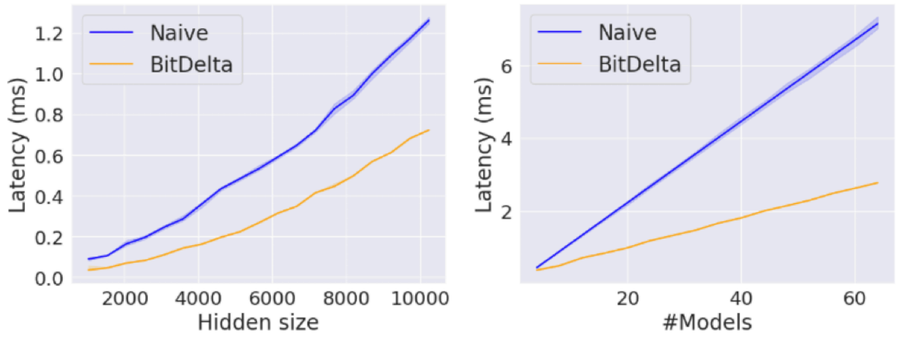
Bit-Delta

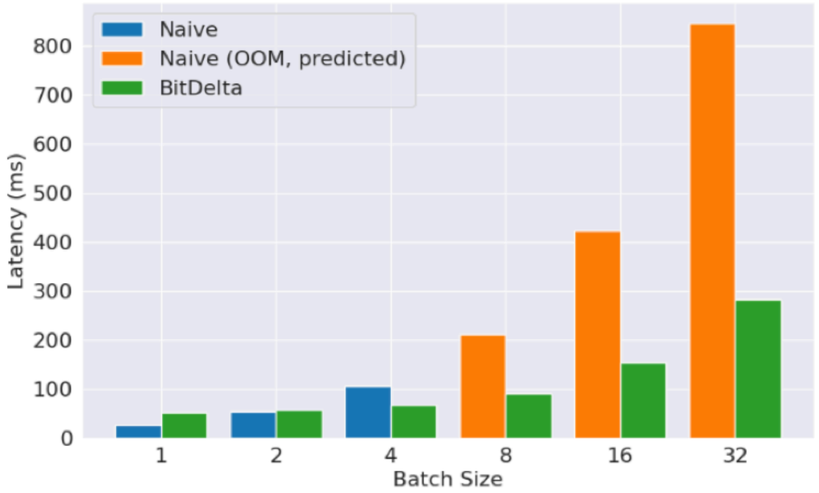
Your Fine-Tune May Only Be Worth One Bit

- **Goal: efficient LLM finetuning with low precision**
- **Intuition: fine-tuning adds less new information to the model, and is thus more compressible.**
- **Our Solution: quantizes the weight delta down to 1 bit without compromising performance, finetuning the scaling factor (per tensor)**

Base Model	Size	ΔSize	Comp. Factor
Llama 2-7B	13.48 GB	1.24 GB	10.87
Llama 2-13B	26.03 GB	2.09 GB	12.45
Llama 2-70B	137.95 GB	8.95 GB	15.41
Mistral-7B v0.1	14.48 GB	1.30 GB	11.14

The more you serve, the more you save!



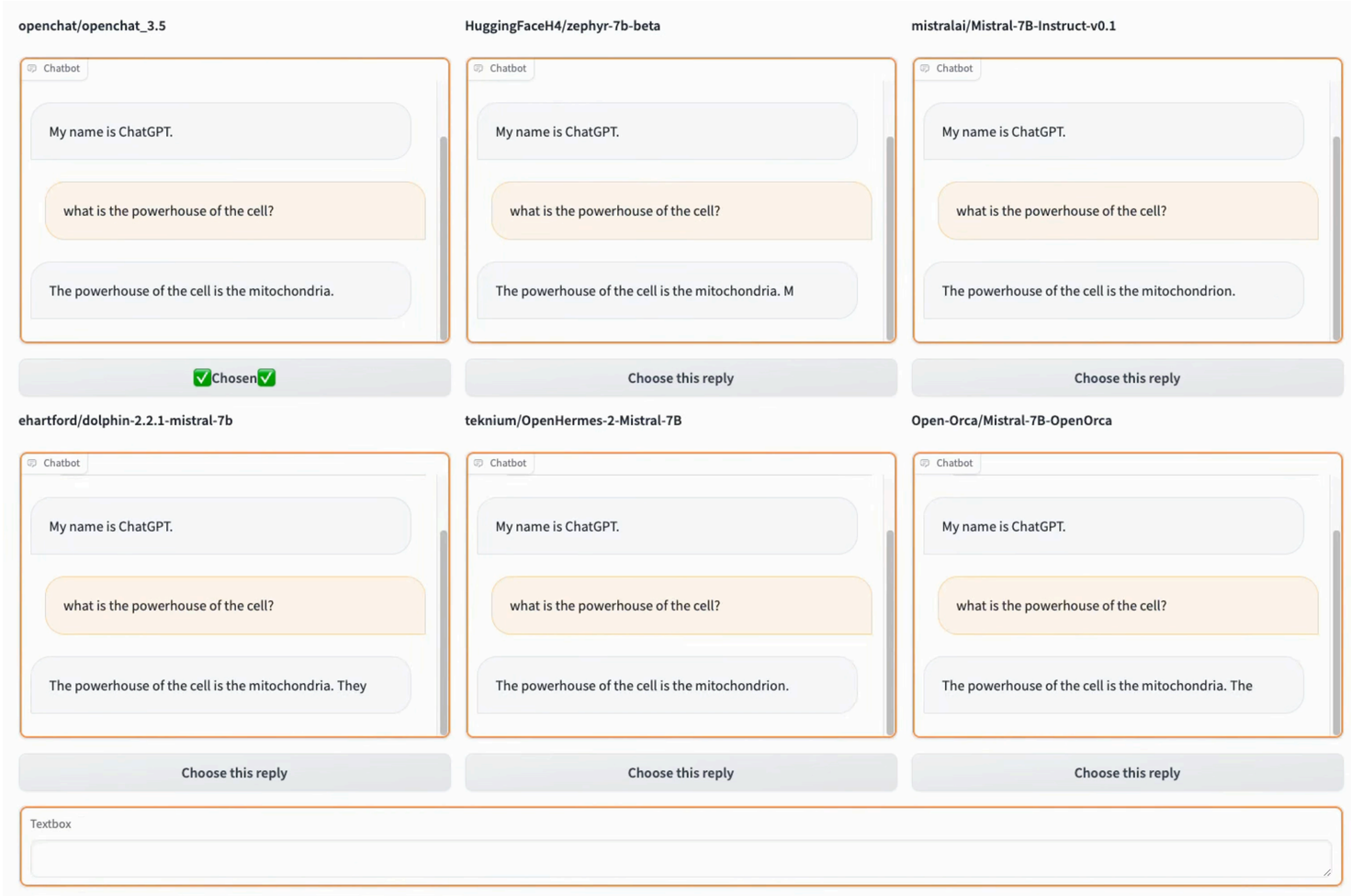


End-to-end decoding latency, Llama2-7B. We implement a fused binary GEMM kernel that allows us to calculate Delta * X in a batched setting while keeping the 1-bit deltas quantized. This kernel fuses the dequantization operation with the GEMM calculation, reducing the data movement overhead by a large factor.

BitDelta: Your Fine-Tune May Only Be Worth One Bit [Liu et al., NeurIPS 2024]

Multi-tenant Serving with BitDelta

Your Fine-Tune May Only Be Worth One Bit



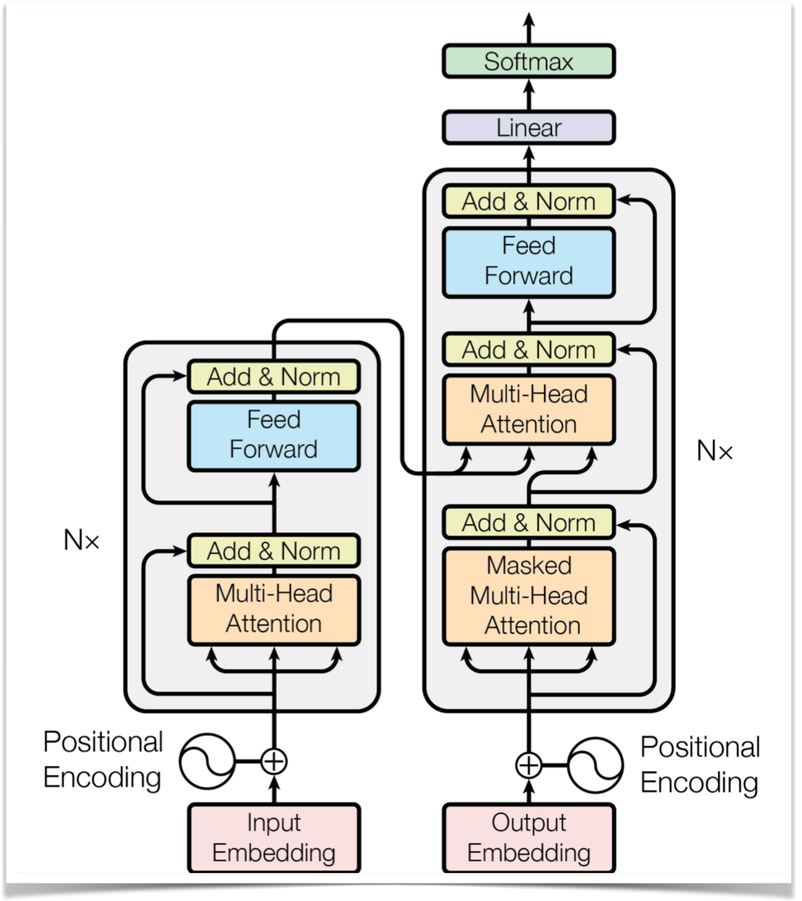
All models are fine-tuned from Mistral-7B

BitDelta: Your Fine-Tune May Only Be Worth One Bit [Liu et al., NeurIPS 2024]

Lecture Plan

Today, we will cover:

1. **LLM Fine-Tuning**
    1. Supervised Fine-Tuning (SFT)
    2. Reinforcement Learning from Human Feedback (RLHF)
    3. Parameter Efficient Fine-Tuning (PEFT)
        - BitFit, TinyTL, Adapter, Prompt-Tuning, Prefix-Tuning
        - LoRA, QLoRA, BitDelta
2. **Multi-modal LLMs**
    1. **Cross-Attention Based: Flamingo**
    2. **Visual Tokens as Input: PaLM-E, VILA**
    3. **Enabling Visual Outputs: VILA-U**
3. **Prompt Engineering**
    1. In-Context Learning (ICL)
    2. Chain-of-Thought (CoT)
    3. Retrieval Augmented Generation (RAG)



Vision Language Model

Q: What is unusual about this image?

A: The unusual aspect of this image is that a chair is flying through the air on a highway, seemingly coming out of the back of a truck.

Q: Can the vehicle proceed through the traffic now?
A: Based on the image, the vehicle cannot proceed through the traffic yet. There are multiple people and bicycles in the crosswalk, and the traffic light is red. The vehicle must wait for the traffic light to turn green before proceeding.

Q: How many cars are jacked up?
A: There are two cars jacked up in the image.

Q: How many persons are there whose head is under the jacked up car?
A: There are two persons whose head is under the jacked up car.

vila.mit.edu [Lin et al., CVPR 2024]

Multi-modal LLMs

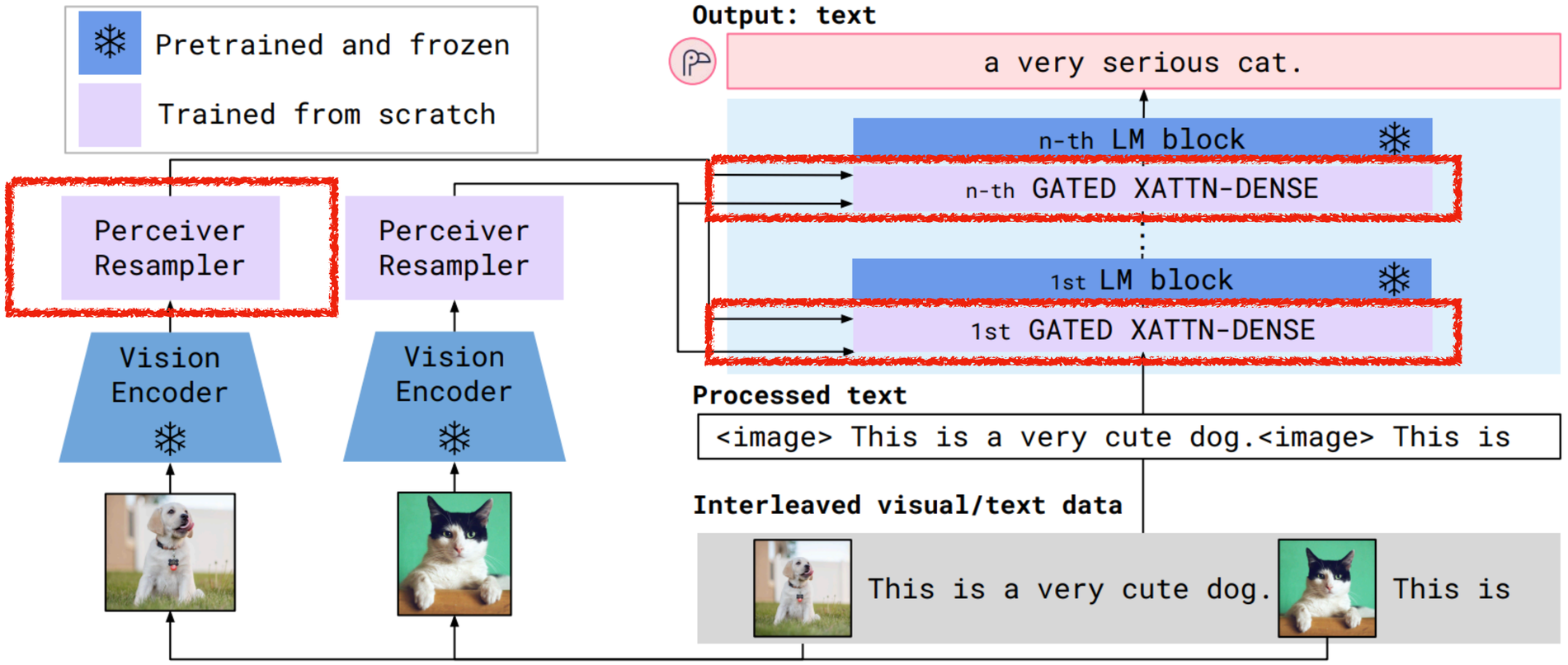
Two ways for visual language models

- Cross-attention to inject visual info into LLM (Flamingo style)
- Visual tokens as input (PaLM-E style)

Flamingo

Cross-attention-based VLMs

- Pioneer work on enabling visual input support for LLMs
- LLM is frozen, added cross-attention layers to the intermediate layers of LLMs



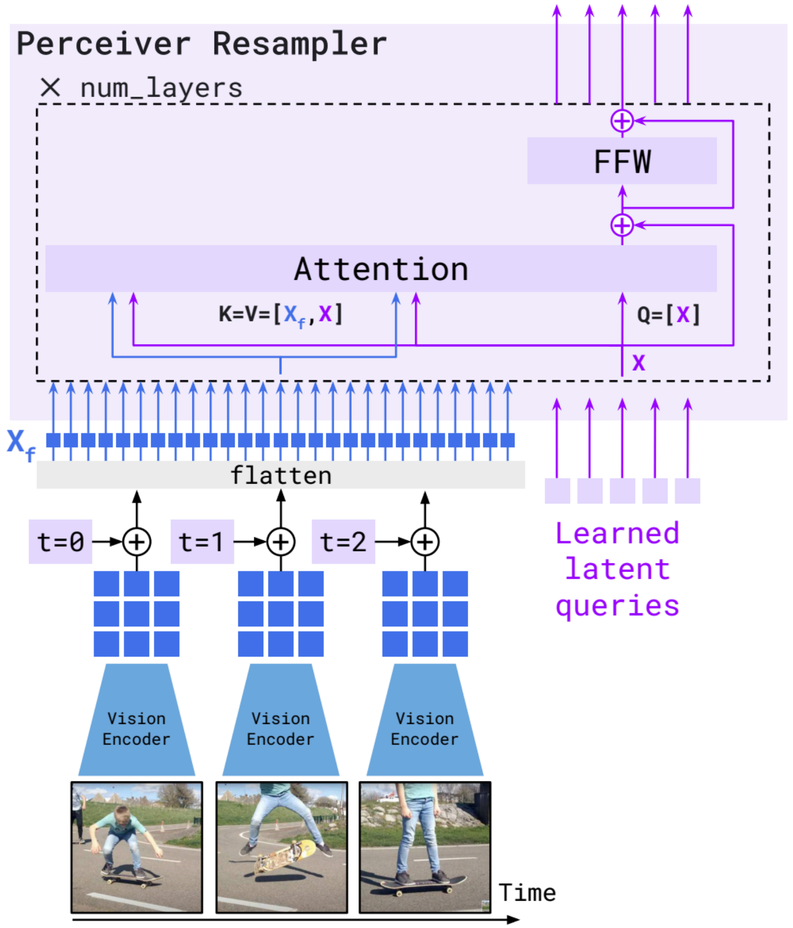
Flamingo: a Visual Language Model for Few-Shot Learning [Alayrac et al., 2022]

Flamingo

Cross-attention-based VLMs

- **Perceiver Resampler**: from varying-size large feature maps to few fixed-size visual tokens

- visual tokens: `[27, dim]`
- learned queries (Q): `[5, dim]`
- K&V: `[32, dim]`
- Attention map: `[5, 32]`
- output: `[5, dim]`




```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # Add the time position embeddings and flatten.
  x_f = x_f + time_embeddings
  x_f = flatten(x_f)  # [T, S, d] -> [T * S, d]
  # Apply the Perceiver Resampler layers.
  for i in range(num_layers):
    # Attention.
    x = x + attention_i(q=x, kv=concat([x_f, x]))
    # Feed forward.
    x = x + ffw_i(x)
  return x
```

Flamingo: a Visual Language Model for Few-Shot Learning [Alayrac et al., 2022]