Parameter-efficient Finetuning Methods (PEFT)

FOUNDATION MODELS COURSE

UNIVERSITY OF AMSTERDAM

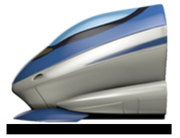
2024

LECTURER: YUKI M. ASANO

Introducing

Meta Llama 3

- released yesterday!
- 8B, 70B, 400B-multimodal (to be released!)

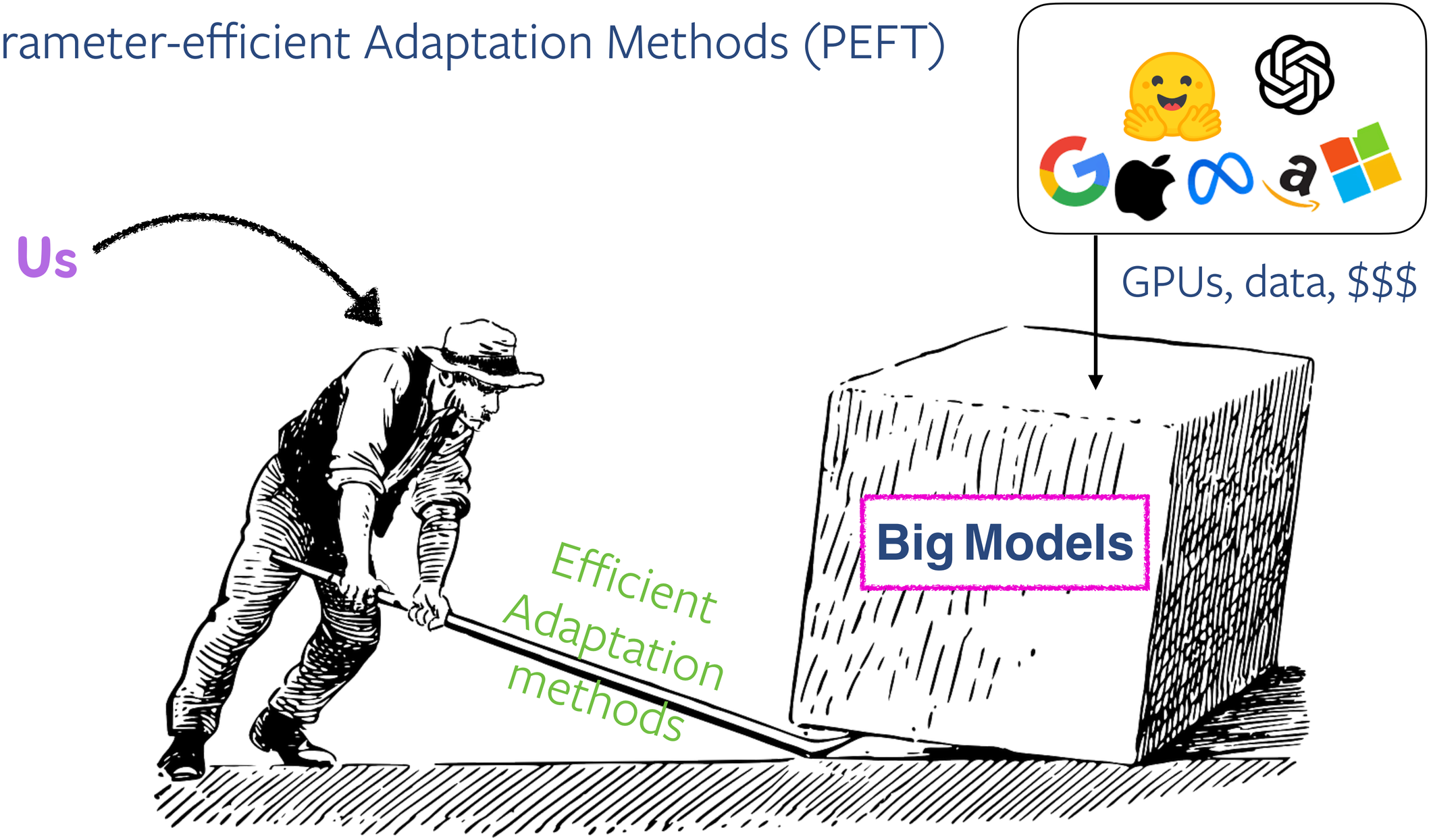


Base pretrained models

Category	Benchmark	Llama 3 8B	Llama2 7B	Llama2 13B	Llama 3 70B	Llama2 70B
General	MMLU (5-shot)	66.6	45.7	53.8	79.5	69.7
	AGIEval English (3-5 shot)	45.9	28.8	38.7	63.0	54.8
	CommonSenseQA (7-shot)	72.6	57.6	67.6	83.8	78.7
	Winogrande (5-shot)	76.1	73.3	75.4	83.1	81.8
	BIG-Bench Hard (3-shot, CoT)	61.1	38.1	47.0	81.3	65.7
	ARC-Challenge (25-shot)	78.6	53.7	67.6	93.0	85.3
Knowledge reasoning	TriviaQA-Wiki (5-shot)	78.5	72.1	79.6	89.7	87.5
Reading comprehension	SQuAD (1-shot)	76.4	72.2	72.1	85.6	82.6
	QuAC (1-shot, F1)	44.4	39.6	44.9	51.1	49.4
	BoolQ (0-shot)	75.7	65.5	66.9	79.0	73.1
	DROP (3-shot, F1)	58.4	37.9	49.8	79.7	70.2

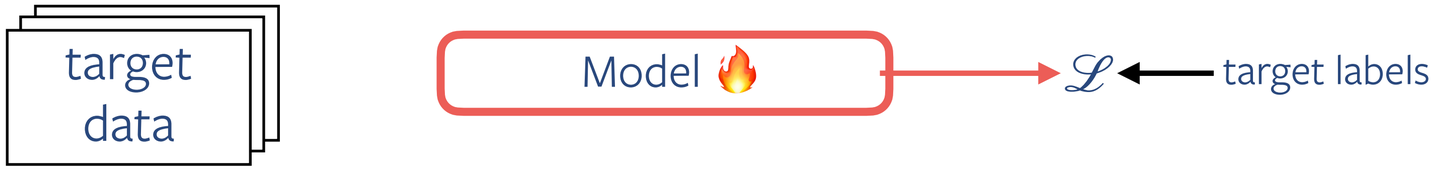
https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

Parameter-efficient Adaptation Methods (PEFT)

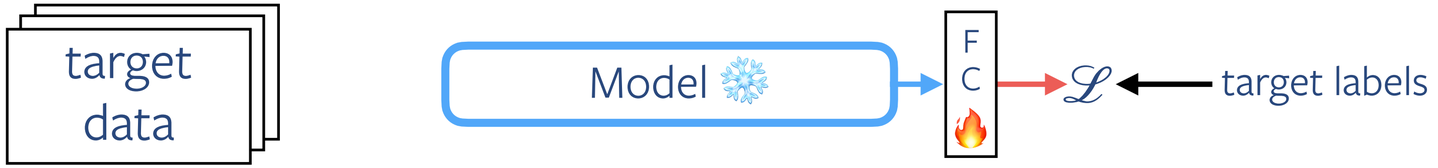


Main ways of adapting models (1/2)

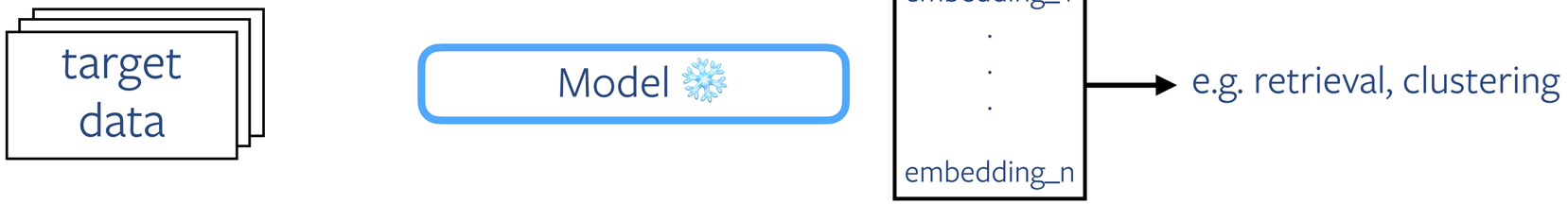
Full-finetuning



Limited-finetuning (e.g. linear probing)

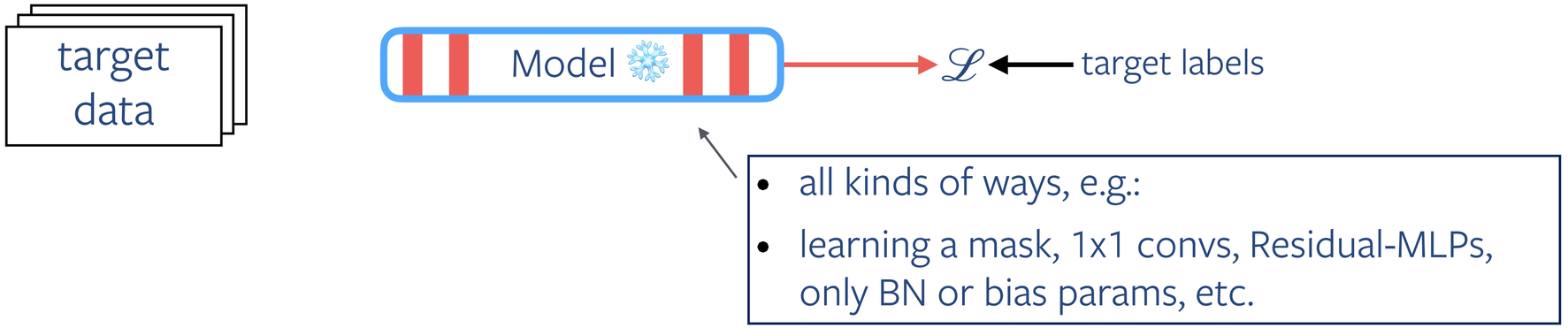


No-finetuning (e.g. used for retrieving similar instances)

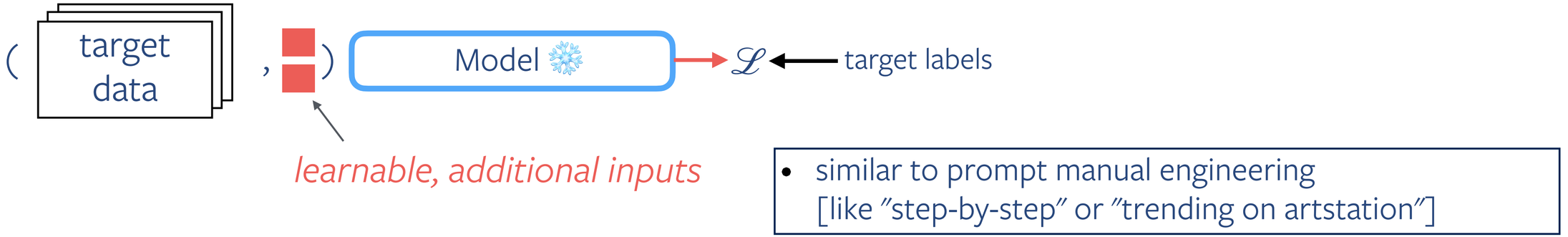


Main ways of adapting models (2/2)

Adapters

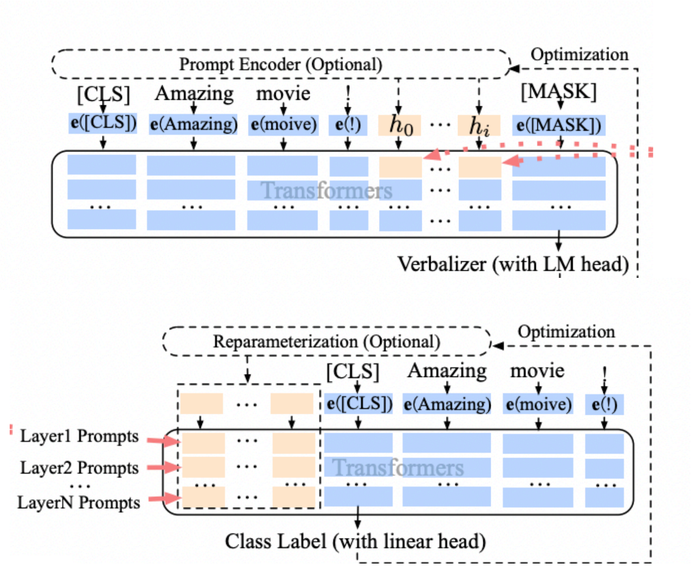


Prompt/prefix learning



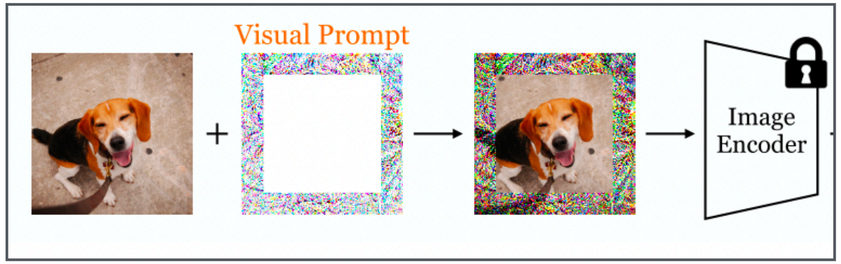
Rebuffi et al. Learning multiple visual domains with residual adapters. NeurIPS 2017, Houlsby et al. Parameter-Efficient Transfer Learning for NLP. ICML 2019
Elsayed et al. Adversarial reprogramming of neural networks. ICLR 2019
Li et al. Prefix-tuning: Optimizing continuous prompts for generation. ACL 2021

Prompt learning: per task

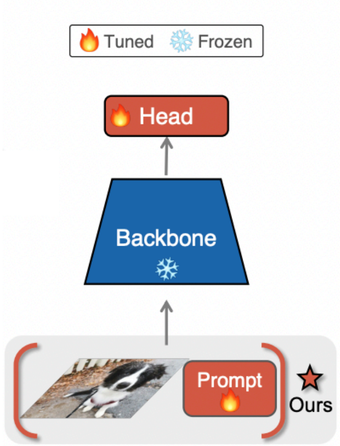


Li et al. Prefix-tuning: Optimizing continuous prompts for generation. ACL 2021

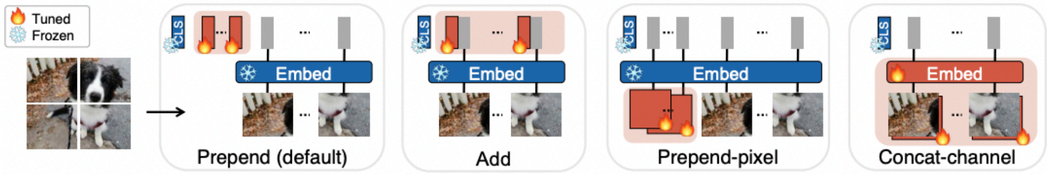
- prefixes are just learnable vectors
- 🧐: are reparameterised as an MLP that gets a fixed input ("more stable")
- Extend this: "deep prompt tuning"
- but increases memory (bc. of attn)



- Works also for CNNs
- Strictly input-only



- Actually also trains linear layer on top
- 👇 explore various ways of prompting inputs for visual inputs



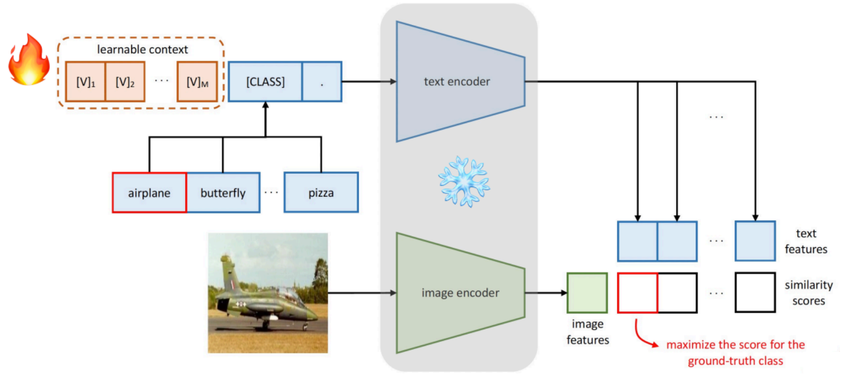
Li et al. Prefix-tuning: Optimizing continuous prompts for generation. ACL 2021
Liu et al. P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks. ACL 2022
Bhang et al. Exploring Visual Prompts for Adapting Large-Scale Models. 2022
Zhou et al. Learning to Prompt for Vision-Language Models.IJCV 2021l, Zhou et al. Conditional Prompt Learning for Vision-Language Models. CVPR 2022
Jia et al. Visual Prompt Tuning.. ECCV 2022

Visual prompting: covered in last lecture

Zhou et al. IJCV 2022

CoOp: Context Optimization

Key idea: **model context words** with learnable vectors, keep other params frozen

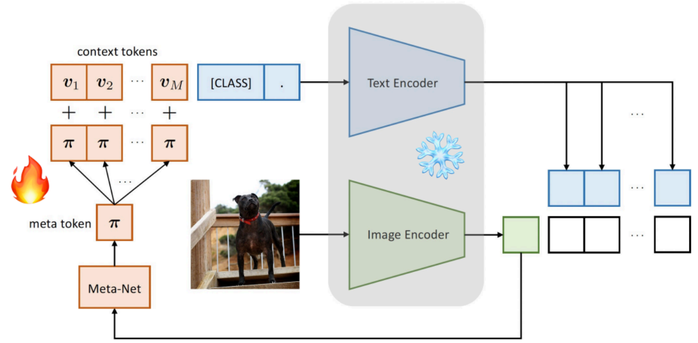


- Learnable, vector-version of "this is an [image/ photograph/illustration] of..."

Zhou et al. CVPR 2022

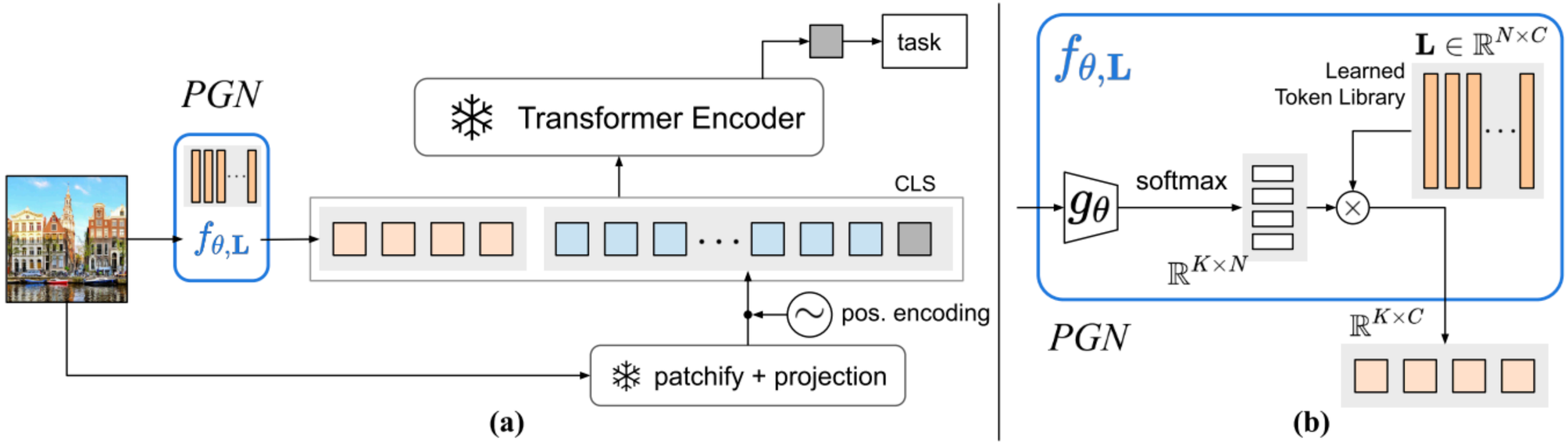
CoCoOp: Conditional Context Optimization

Key idea: **condition the prompt** on each input image rather than fixed once learned.



- Condition this additionally on image

Prompt learning per datum, *in input-space only*



- Learn a input-to-prompt mini-network
- Generate prompts from a set of learnable prompts
- Prompts (learned in space after first conv1), can be made to be input-only (convs are linear operation!)

PGN learns what's missing in CLIP

	PGN backbone (alone)	CLIP	CLIP with PGN
CIFAR-100	63.7	63.1	79.3

More robust compared to linear probing (LP)

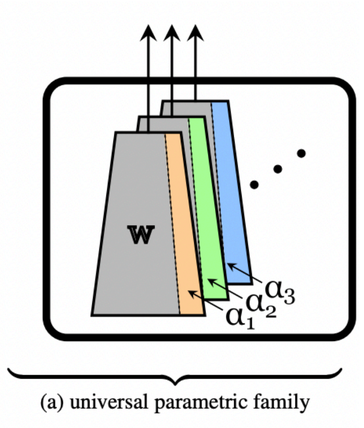
Method	ImageNet	A	R	V2	Sketch
PGN	66.0	22.8	62.5	56.7	36.5
LP	67.0	10.6	38.1	1.0	36.1

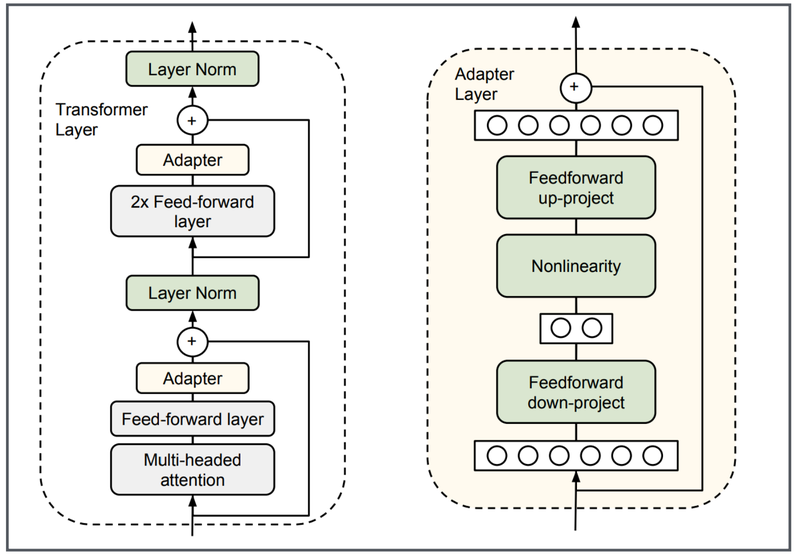
Adapters: any modification "in the middle" of NNs

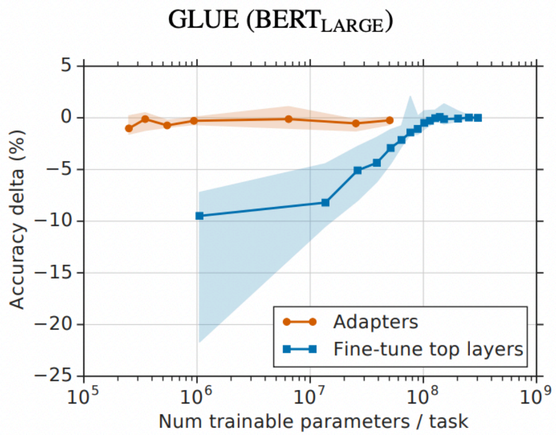
Simplest form: residual adapters

$$g(x;\alpha) = x + \alpha * x.$$

limit α to e.g. 1x1 conv



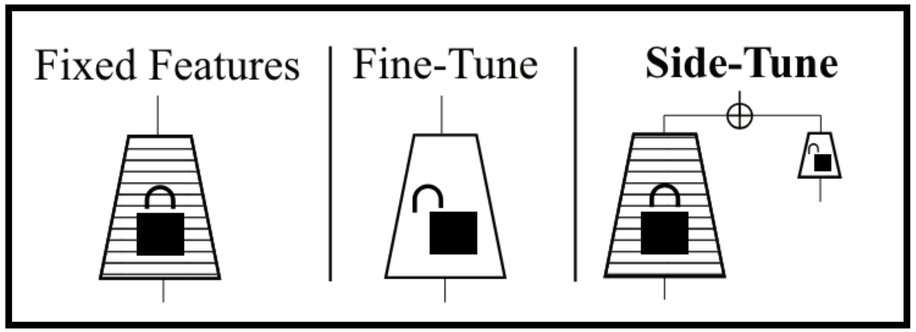




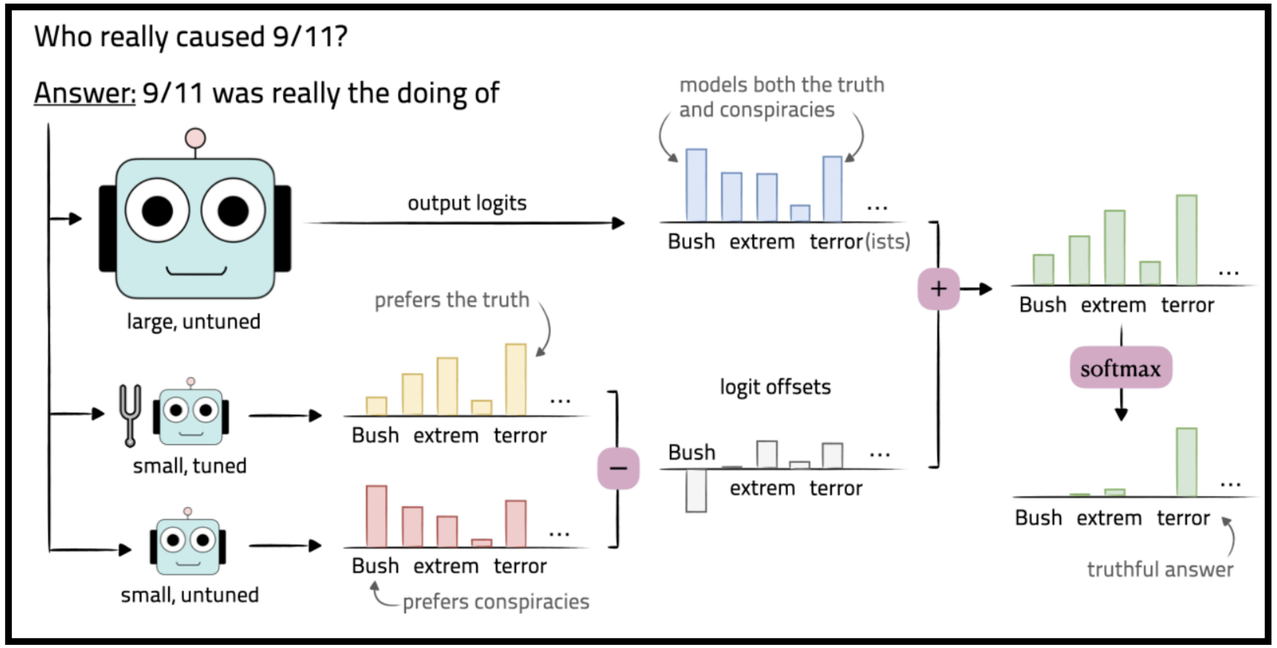
- (-) makes computation graph more complex; adds inference time
- (+) doesn't require much memory to store
- (+) very expressive/performant and fast to learn

Rebuffi et al. Learning multiple visual domains with residual adapters. NeurIPS 2017, Houlsby et al. Parameter-Efficient Transfer Learning for NLP. ICML 2019

Side-tuning, DExperts, Tuning Language Models by Proxy



- Learn new network that learns "the missing parts"
- Keep original model frozen



- Simply combine outputs in logit-space of output tokens
- strangely, simply arithmetic (+ and -) works very well!

Side-Tuning: A Baseline for Network Adaptation via Additive Side Networks. Zhang et al. ECCV 2020
DEXPERTS: Decoding-Time Controlled Text Generation with Experts and Anti-Experts. Liu et al. ACL 2021
Tuning Language Models by Proxy. Liu et al. 2024

Fine-tuning only the bias terms / Norm layers

BitFit

$$\mathbf{Q}^{m,\ell}(\mathbf{x}) = \mathbf{W}_q^{m,\ell}\mathbf{x} + \mathbf{b}_q^{m,\ell}$$
$$\mathbf{K}^{m,\ell}(\mathbf{x}) = \mathbf{W}_k^{m,\ell}\mathbf{x} + \mathbf{b}_k^{m,\ell}$$
$$\mathbf{V}^{m,\ell}(\mathbf{x}) = \mathbf{W}_v^{m,\ell}\mathbf{x} + \mathbf{b}_v^{m,\ell}$$

frozen, trainable

- Learn only the bias terms
- Learning vectors instead of matrices -> efficient

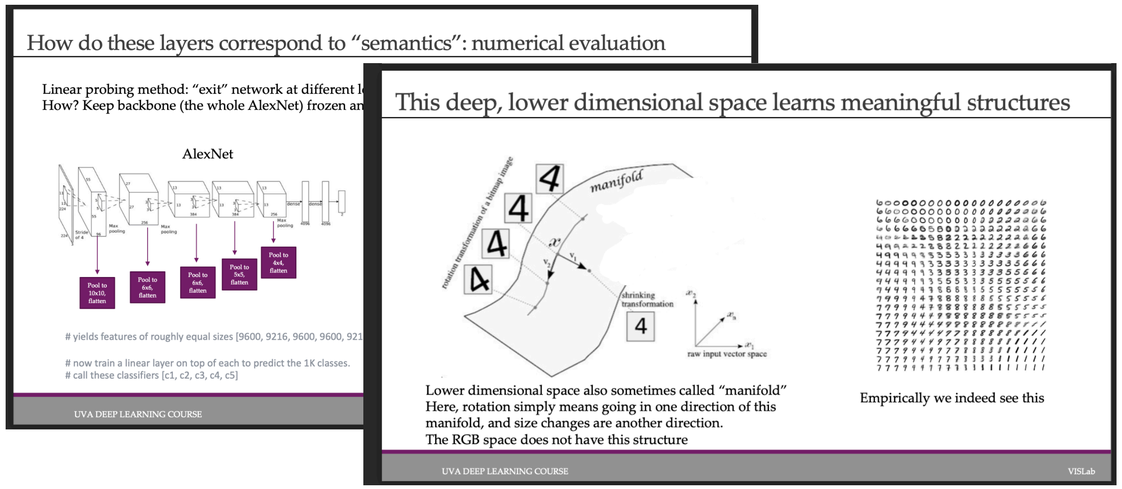
Feature normalization transforms such as Batch and Layer-Normalization have become indispensable ingredients of state-of-the-art deep neural networks. Recent studies on fine-tuning large pretrained models indicate that just tuning the parameters of these affine transforms can achieve high accuracy for downstream tasks. These findings open the questions about the expressive power of tuning the normalization layers of frozen networks. In this work, we take the first step towards this question and show that for random ReLU networks, fine-tuning only its normalization layers can reconstruct any target network that is $O(\sqrt{\text{width}})$ times smaller. We show that this holds even for randomly sparsified networks, under sufficient overparameterization, in agreement with prior empirical work.

- Learn only the LayerNorm / BatchNorm parameters
- Show that it is quite expressive in theory (and practice)

BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. Ben-Zaken et al. ACL 2022
The Expressive Power of Tuning Only the Normalization Layers. Giannou et al. PMLR 2023

LoRA: adapting matrix multiplies in efficiently / "a generalisation of full-finetuning"

$$\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$$



Remember from DL1:

- real data ~ lies on lower dimensional manifold,
- DNNs map from RGB space gradually to more semantic space.

Normal fully connected layer:

$$h = W_0 x$$

LoRA adapted:

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

BA is low-rank matrix.

"Low-rank"
--> think of it as outer-product of few vectors

- (-) not as expressive as adapters
- (+) linear op, so after training can be fused with original weights --> same speed

Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. ICLR 2022
Aghajanyan et al. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. ACL 2021

In LoRA: Why are two matrices A & B learned, even though they are only used as A*B?

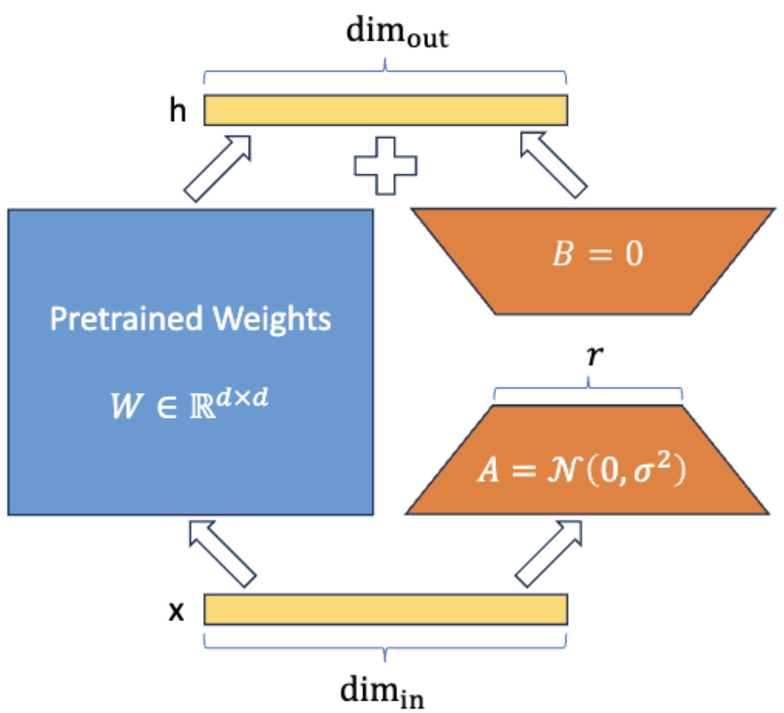
1) Learning two easy linear transformations is easier than one complicated one
2) Enforcing low-rank ness via singular value decomposition is expensive
3) By allowing A and B to be square-sized we obtain more optimised matrix multiplies
4) Actually, rather than two linear layers, a single bigger one would also do it.

VeRA: Vector-based Random Matrix Adaptation

DAWID J. KOPICZKO, TIJMEN BLANKEVOORT, YUKI M. ASANO

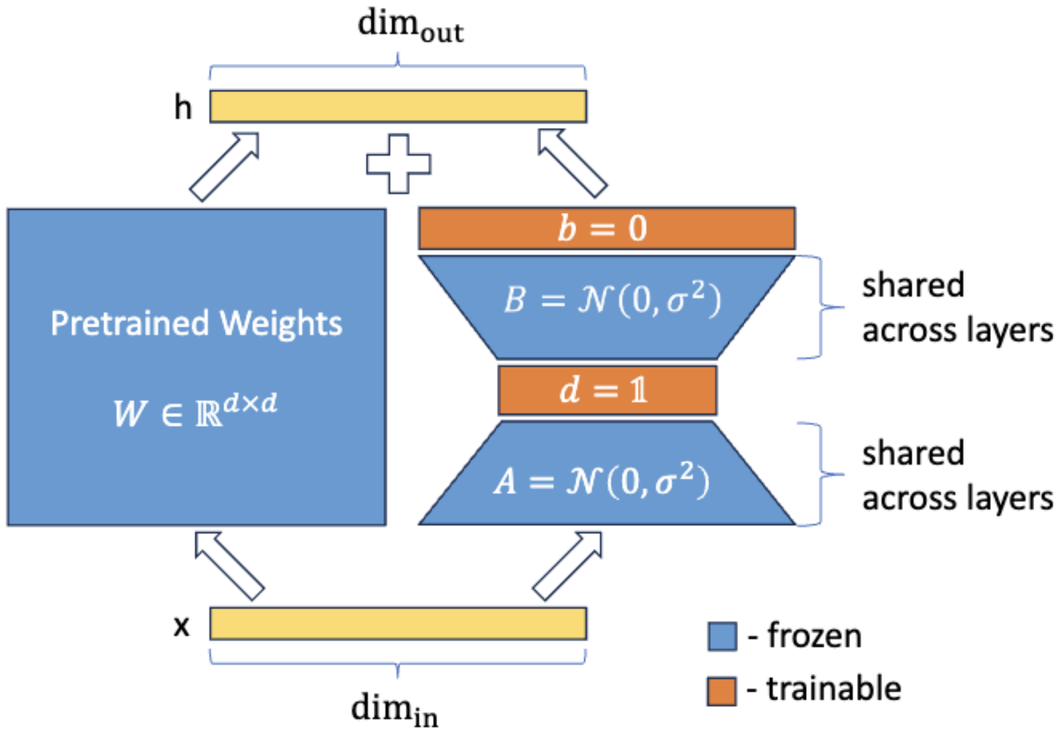
ICLR'24

We make LoRA more efficient



Low-Rank Adaptation (LoRA)

W_new = W_old + AB,
where A,B are low-rank
learned per-layer



Vector-based Random Matrix Adaptation (VeRA)

W_new = W_old + AdBb,
where A,B are random & frozen, same across layers;
d,b are learned vectors

Results on GLUE with RoBERTa

	Method	# Trainable Parameters	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
BASE	FT	125M	94.8	90.2	63.6	92.8	78.7	91.2	85.2
	BitFit	0.1M	93.7	**92.7**	62.0	91.8	81.5	90.8	85.4
	Adpt^{D}	0.3M	$94.2_{\pm 0.1}$	$88.5_{\pm 1.1}$	$60.8_{\pm 0.4}$	$93.1_{\pm 0.1}$	$71.5_{\pm 2.7}$	$89.7_{\pm 0.3}$	83.0
	Adpt^{D}	0.9M	$94.7_{\pm 0.3}$	$88.4_{\pm 0.1}$	$62.6_{\pm 0.9}$	$93.0_{\pm 0.2}$	$75.9_{\pm 2.2}$	$90.3_{\pm 0.1}$	84.2
	LoRA	0.3M	$\mathbf{95.1}_{\pm 0.2}$	$89.7_{\pm 0.7}$	$63.4_{\pm 1.2}$	$\mathbf{93.3}_{\pm 0.3}$	$\mathbf{86.6}_{\pm 0.7}$	$\mathbf{91.5}_{\pm 0.2}$	**86.6**
	VeRA	**0.043M**	$94.6_{\pm 0.1}$	$89.5_{\pm 0.5}$	$\mathbf{65.6}_{\pm 0.8}$	$91.8_{\pm 0.2}$	$78.7_{\pm 0.7}$	$90.7_{\pm 0.2}$	85.2
LARGE	Adpt^{P}	3M	$96.1_{\pm 0.3}$	$90.2_{\pm 0.7}$	$\mathbf{68.3}_{\pm 1.0}$	$\mathbf{94.8}_{\pm 0.2}$	$83.8_{\pm 2.9}$	$92.1_{\pm 0.7}$	87.6
	Adpt^{P}	0.8M	$\mathbf{96.6}_{\pm 0.2}$	$89.7_{\pm 1.2}$	$67.8_{\pm 2.5}$	$\mathbf{94.8}_{\pm 0.3}$	$80.1_{\pm 2.9}$	$91.9_{\pm 0.4}$	86.8
	Adpt^{H}	6M	$96.2_{\pm 0.3}$	$88.7_{\pm 2.9}$	$66.5_{\pm 4.4}$	$94.7_{\pm 0.2}$	$83.4_{\pm 1.1}$	$91.0_{\pm 1.7}$	86.8
	Adpt^{H}	0.8M	$96.3_{\pm 0.5}$	$87.7_{\pm 1.7}$	$66.3_{\pm 2.0}$	$94.7_{\pm 0.2}$	$72.9_{\pm 2.9}$	$91.5_{\pm 0.5}$	84.9
	LoRA-FA	3.7M	96.0	90.0	68.0	94.4	86.1	92.0	87.7
	LoRA	0.8M	$96.2_{\pm 0.5}$	$90.2_{\pm 1.0}$	$68.2_{\pm 1.9}$	$\mathbf{94.8}_{\pm 0.3}$	$85.2_{\pm 1.1}$	$\mathbf{92.3}_{\pm 0.5}$	**87.8**
	VeRA	**0.061M**	$96.1_{\pm 0.1}$	$\mathbf{90.9}_{\pm 0.7}$	$68.0_{\pm 0.8}$	$94.4_{\pm 0.2}$	$\mathbf{85.9}_{\pm 0.7}$	$91.7_{\pm 0.8}$	**87.8**

Results on E2E benchmark with GPT2

	Method	# Trainable Parameters	BLEU	NIST	METEOR	ROUGE-L	CIDEr
MEDIUM	FT[1]	354.92M	68.2	8.62	46.2	71.0	2.47
	Adpt^{L1}	0.37M	66.3	8.41	45.0	69.8	2.40
	Adpt^{L1}	11.09M	68.9	8.71	46.1	71.3	2.47
	Adpt^{H1}	11.09M	67.3	8.50	46.0	70.7	2.44
	DyLoRA[2]	0.39M	69.2	8.75	46.3	70.8	2.46
	AdaLoRA[3]	0.38M	68.2	8.58	44.1	70.7	2.35
	LoRA	0.35M	68.9	8.69	46.4	71.3	**2.51**
	VeRA	**0.098M**	**70.1**	**8.81**	**46.6**	**71.5**	2.50
LARGE	FT[1]	774.03M	68.5	8.78	46.0	69.9	2.45
	Adpt^{L1}	0.88M	69.1	8.68	46.3	71.4	2.49
	Adpt^{L1}	23.00M	68.9	8.70	46.1	71.3	2.45
	LoRA	0.77M	70.1	8.80	46.7	**71.9**	2.52
	VeRA	**0.17M**	**70.3**	**8.85**	**46.9**	71.6	**2.54**

Instruction tuning: better than LoRA with 100x less parameters

Model	Method	# Parameters	Score
Llama 13B	-	-	2.61
LLAMA 7B	LoRA	159.9M	5.03
	VeRA	1.6M	4.77
LLAMA 13B	LoRA	250.3M	5.31
	VeRA	2.4M	5.22
LLAMA2 7B	LoRA	159.9M	5.19
	VeRA	1.6M	5.08
LLAMA2 13B	LoRA	250.3M	5.77
	VeRA	2.4M	5.93

Works also on Image Classification with pretrained ViT

	Method	# Trainable Parameters	CIFAR100	Food101	Flowers102	RESISC45
ViT-B	Head	-	77.7	86.1	98.4	67.2
	Full	85.8M	**86.5**	**90.8**	98.9	**78.9**
	LoRA	294.9K	85.9	89.9	98.8	77.7
	VeRA	**24.6K**	84.8	89.0	**99.0**	77.0
ViT-L	Head	-	79.4	76.5	98.9	67.8
	Full	303.3M	86.8	78.7	98.8	**79.0**
	LoRA	786.4K	87.0	**79.5**	99.1	78.3
	VeRA	**61.4K**	**87.5**	79.2	**99.2**	78.6

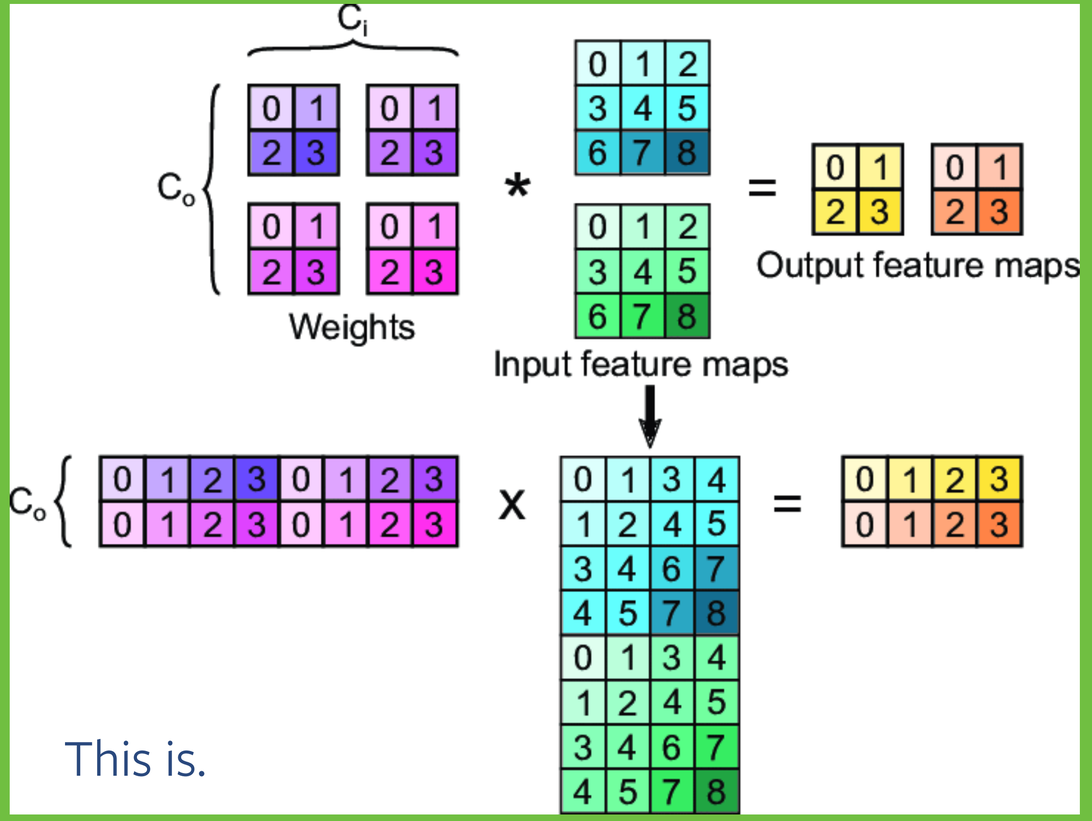
now on HuggingFace!

https://github.com/huggingface/peft

Think: how would you apply LoRA on a convolutional network like a ResNet/U-Net?

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix}$$

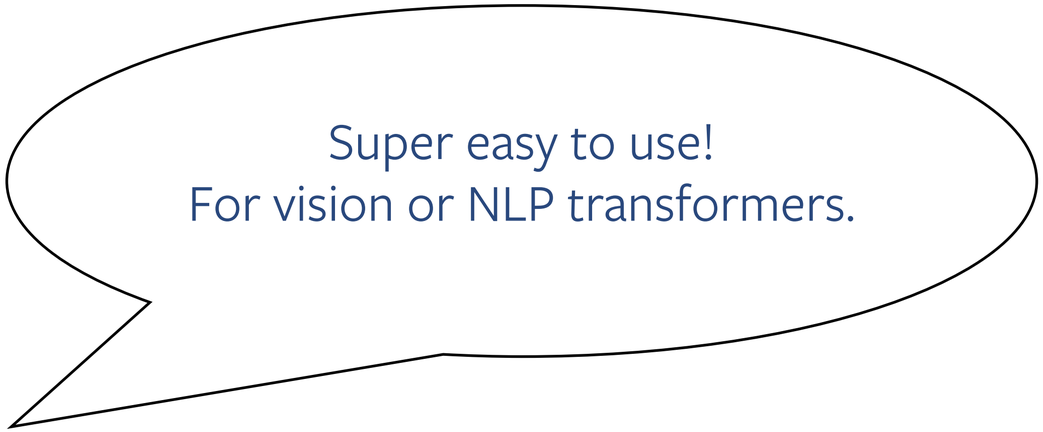
This is not the right way to think about it.



🤗 PEFT: https://github.com/huggingface/peft

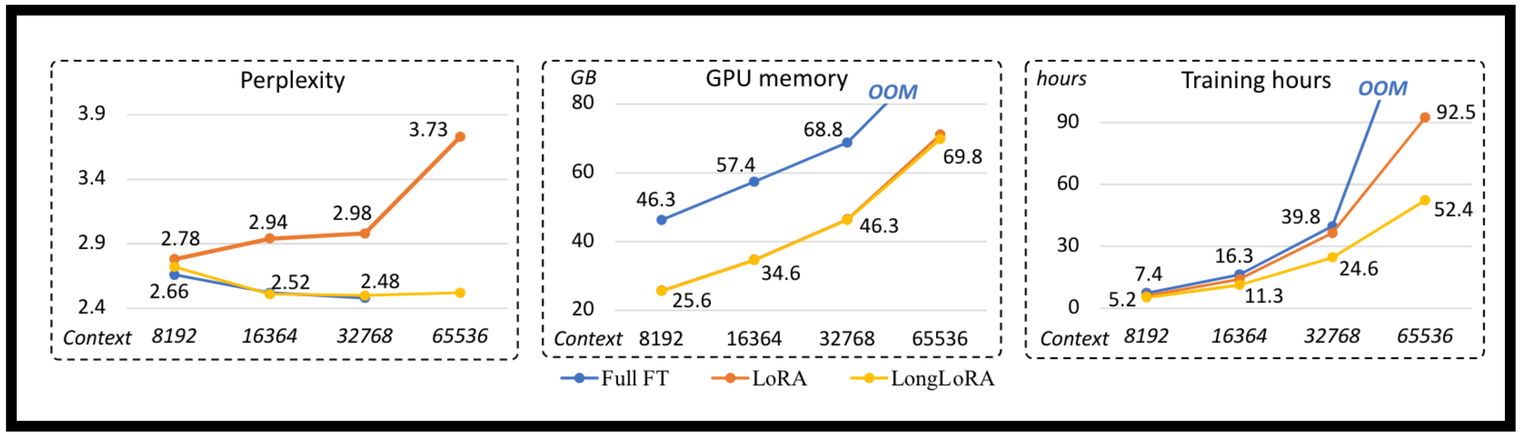
In [2]:
```
batch_size = 256
model_name_or_path = "roberta-base"
task = "mrpc"
peft_type = PeftType.VERA
device = "cuda"
num_epochs = 30
max_length = 128
```

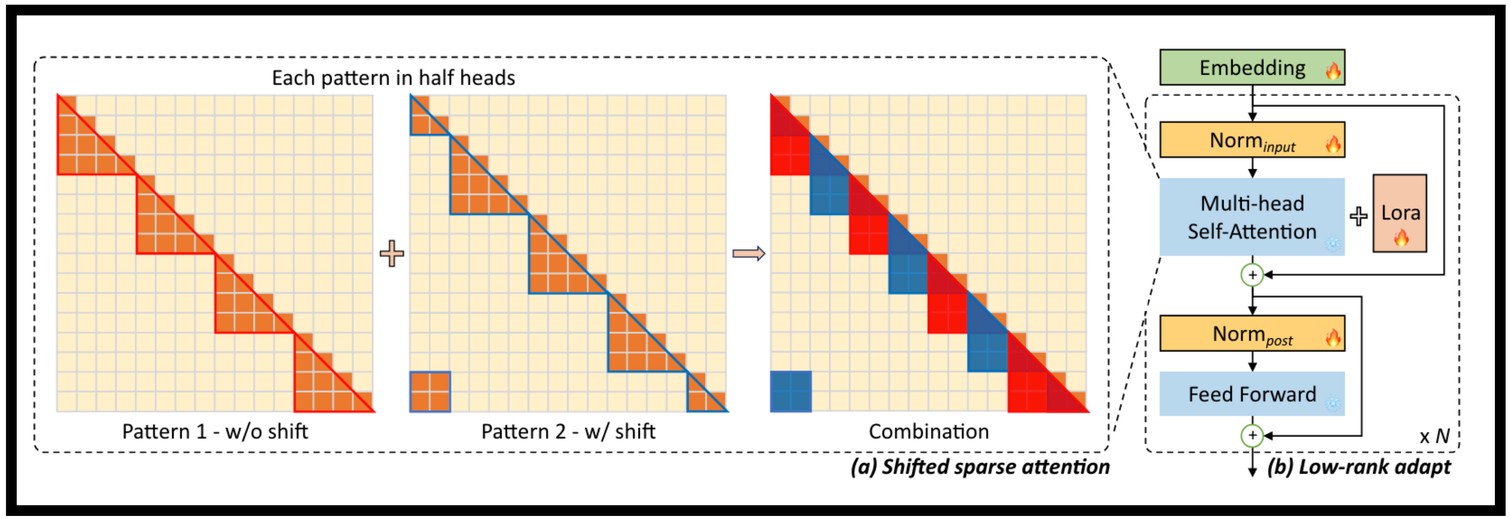
In [3]:
```
peft_config = VeraConfig(
    task_type="SEQ_CLS",
    inference_mode=False,
    r=512,
    projection_prng_key=0xABC,
    d_initial=0.1,
    target_modules=["query", "value"],
    save_projection=True
)
head_lr = 1e-2
vera_lr = 2e-2
```



In [5]:
```
model = AutoModelForSequenceClassification.from_pretrained(model_name_or_path, return_dict=True, max_length=None)
model = get_peft_model(model, peft_config)
model.print_trainable_parameters()
model
```

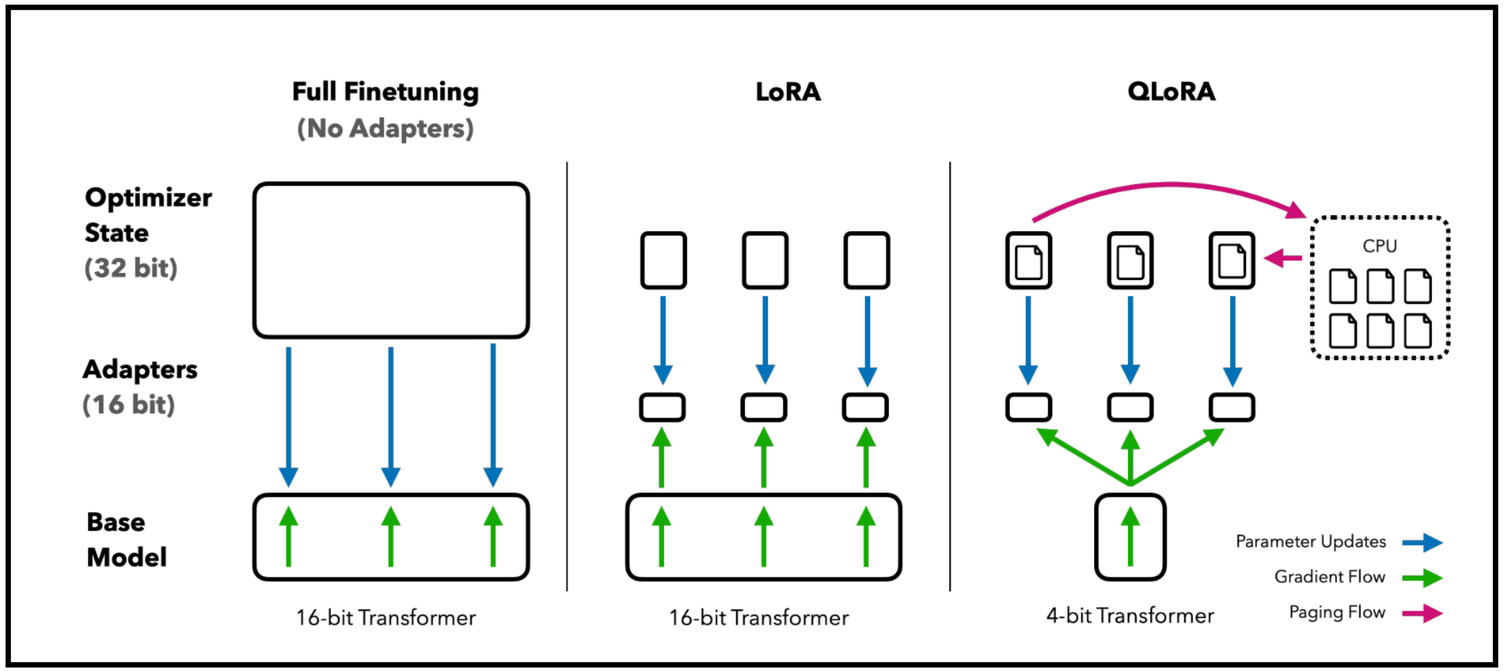
LongLoRA: turn LLM into long-context LLM cheaply





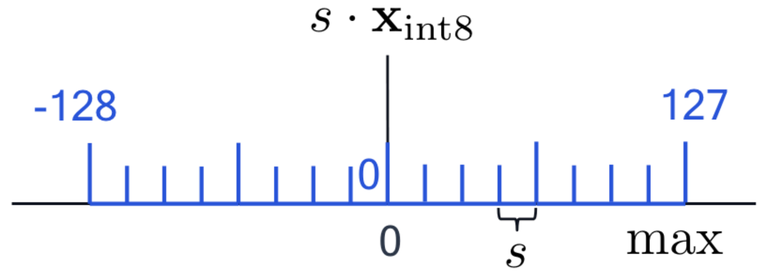
- full-finetuning works best, but requires too much GPU memory
- LoRA alone isn't enough for high performance
- LongLoRA: LoRA + limited finetuning + smart attention patterns

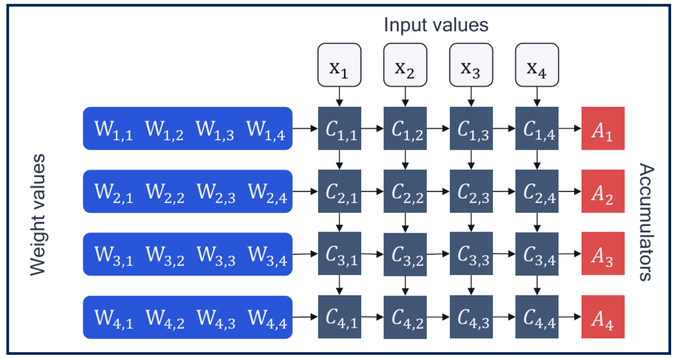
QLoRA



- Better 4-bit datatype
- Double quantisation: quantise the quantisation constants

Quick intro: quantisation





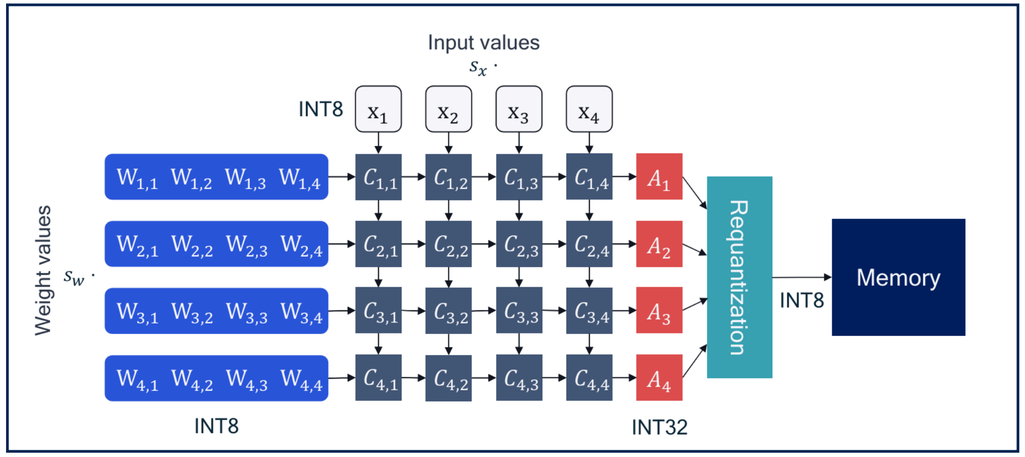
Quantisation:

$$\mathbf{x}_{\text{int}} = \text{clamp}\left(\left\lfloor \frac{\mathbf{x}}{s} \right\rceil + z; 0, 2^b - 1\right)$$

s: scale, b: bitwidth,
z: zero-point

Dequantisation:

$$\mathbf{x} \approx \widehat{\mathbf{x}} = s\left(\mathbf{x}_{\text{int}} - z\right)$$



Quantisation grid limits
(below & above values are clipped)

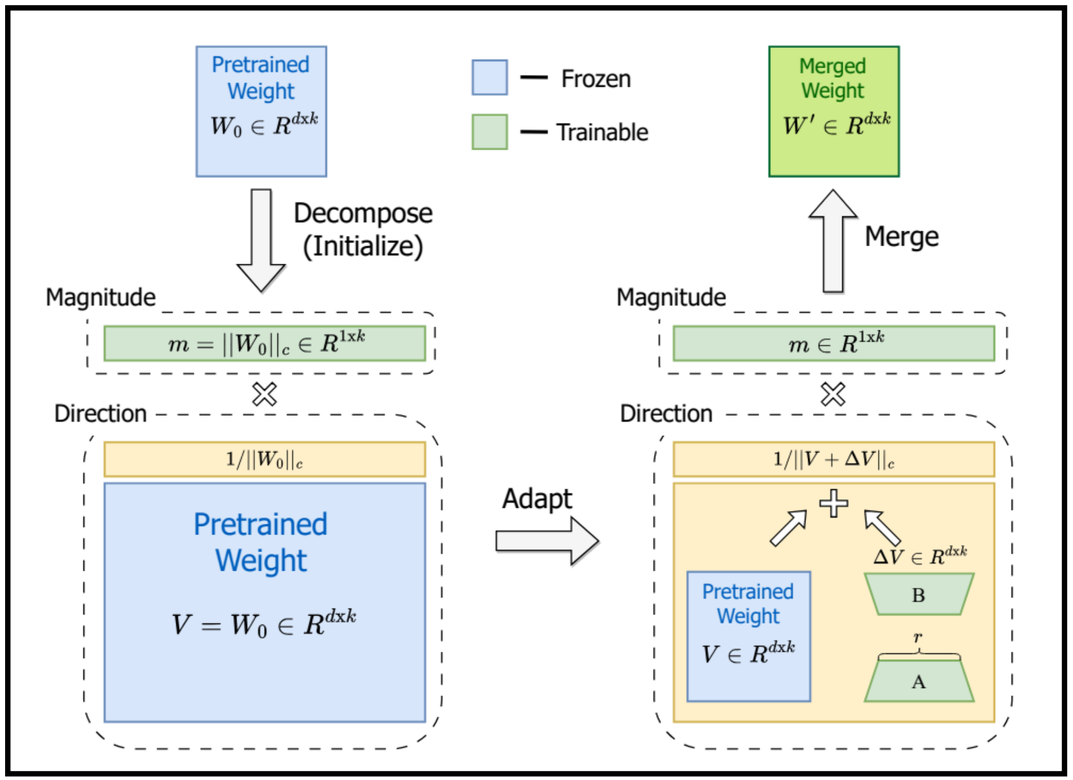
$$q_{\text{min}} = -sz \text{ and } q_{\text{max}} = s(2^b - 1 - z).$$

larger q-range --> lower **clipping error**,
but higher **rounding error** [-0.5s, 0.5s]

Imagine you have an addition operation in your network. How many ″quantise″ operations would this require?

1) 1
2) 2
3) 3
4) 4

DoRA: Weight-Decomposed Low-Rank Adaptation



- Adapt the direction, not the magnitude
- See also weight-norm (2016)

Table 5. Average scores on MT-Bench assigned by GPT-4 to the answers generated by fine-tuned LLaMA-7B/LLaMA2-7B.

Model	PEFT Method	# Params (%)	Score
LLaMA-7B	LoRA	2.31	5.1
	DoRA (Ours)	2.33	**5.5**
	VeRA	0.02	4.3
	DVoRA (Ours)	0.04	**5.0**
LLaMA2-7B	LoRA	2.31	5.7
	DoRA (Ours)	2.33	**6.0**
	VeRA	0.02	5.5
	DVoRA (Ours)	0.04	**6.0**

- Combinable with VeRA

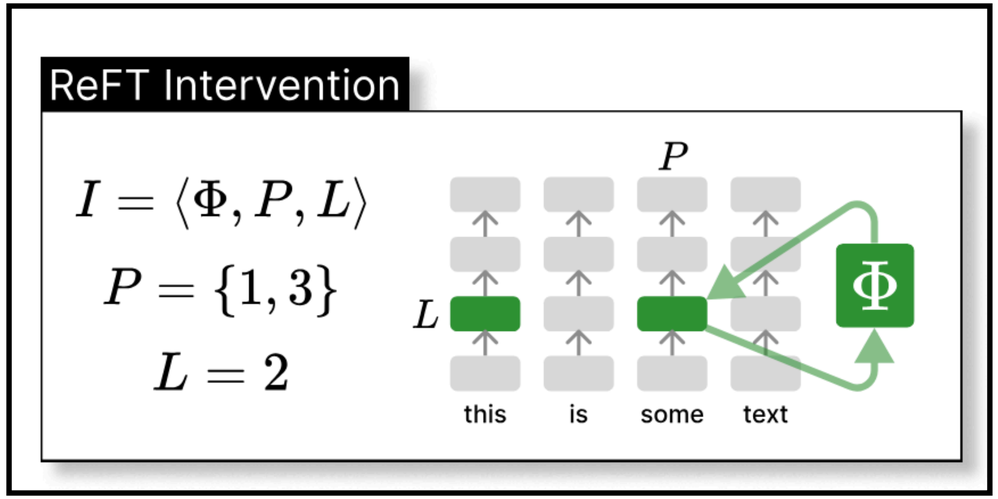
DoRA: Weight-Decomposed Low-Rank Adaptation. Liu et al. 2024
Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. Salimans & Kingma. NeurIPS 2016

ReFT: Representation Finetuning for Language Models

- The intervention function $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with learned parameters ϕ.
- A set of input positions $P \subseteq \{1, \ldots, n\}$ that the intervention is applied to.
- The layer $L \in \{1, \ldots, m\}$ at which the intervention is applied.

We implement the intervention I as the following operation that overwrites some representations $\mathbf{h}$:

$$\mathbf{h}^{(L)} \leftarrow \left(\Phi\left(\mathbf{h}_p^{(L)}\right) \text{ if } p \in P \text{ else } \mathbf{h}_p^{(L)}\right)_{p \in 1,\ldots,n} \quad (6)$$



- Adapt *representations* instead of weights. 🤔
- Larger search-space
- Adaptation depends on position in sentence! 😅

Take 1-2 minutes to write down what you've learned so far in this lecture, also include what you find hard to understand.

Next, turn to your neighbor and share notes for 2-3min.

LongLoRA: turn LLM into long-context LLM cheaply

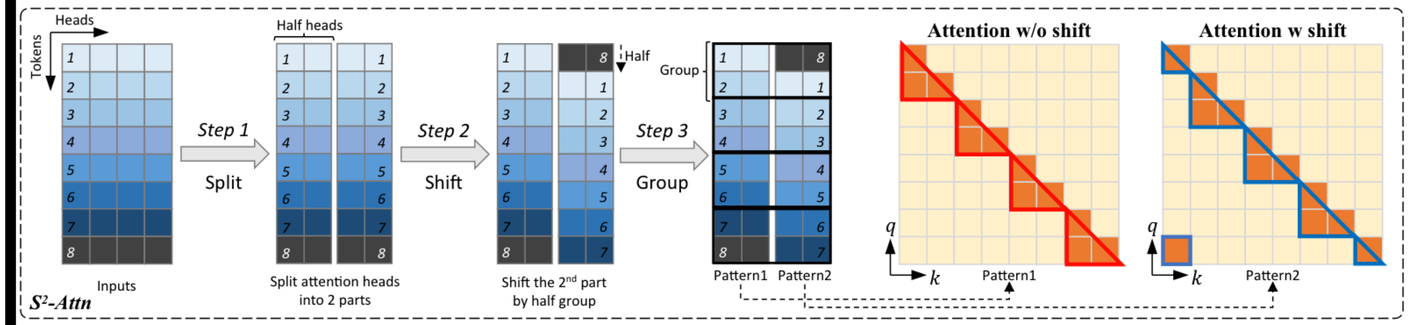


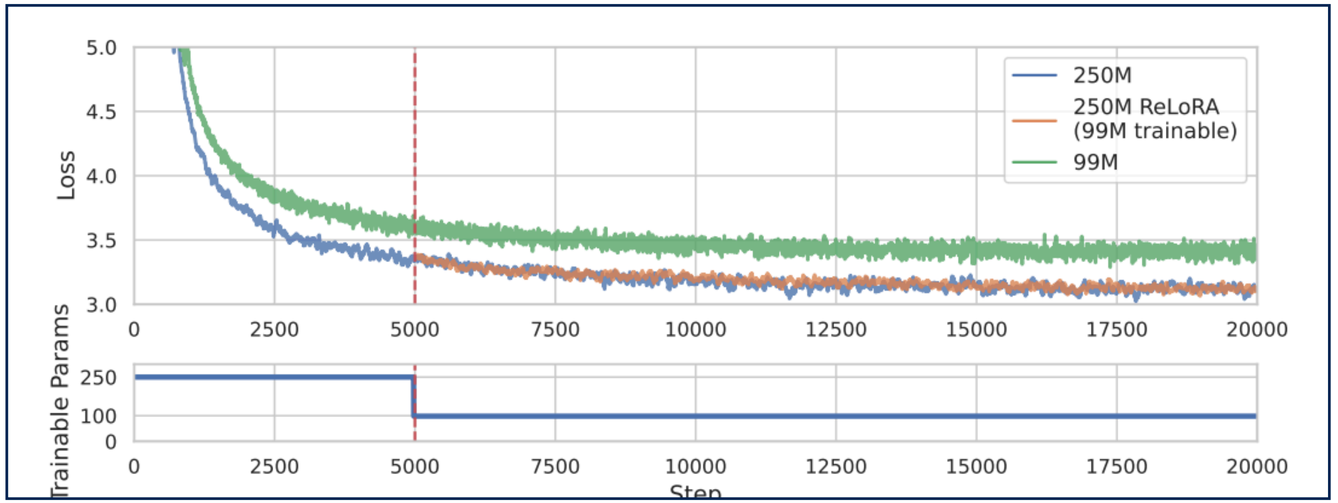
Figure 3: **Illustration of S^2-Attn**. It involves three steps. First, it splits features along the head dimension into two chunks. Second, tokens in one of the chunks are shifted by half of the group size. Third, we split tokens into groups and reshape them into batch dimensions. Attention only computes in each group in ours while the information flows between groups via shifting. Potential information leakage might be introduced by shifting, while this is easy to prevent via a small modification on the attention mask. We ablate this in the variant 2 in Section B.3 in the appendix.

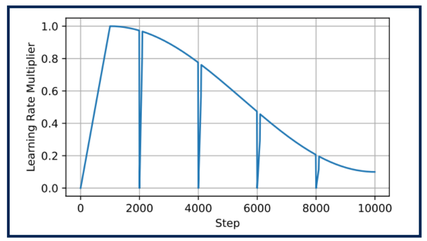
- In addition finetune norm-layers and embedding layer
- By allowing attention within groups, model can look at long context

```
# B: batch size; S: sequence length or number of tokens; G: group size;
# H: number of attention heads; D: dimension of each attention head

# qkv in shape (B, N, 3, H, D), projected queries, keys, and values
# Key line 1: split qkv on H into 2 chunks, and shift G/2 on N
qkv = cat((qkv.chunk(2, 3)[0], qkv.chunk(2, 3)[1].roll(-G/2, 1)), 3).view(B*N/G,G,3,H,D)
```

PEFT ideas for pretraining





- first train normally
- then only with LoRA, absorbing the AB matrices occasionally and relearning new ones

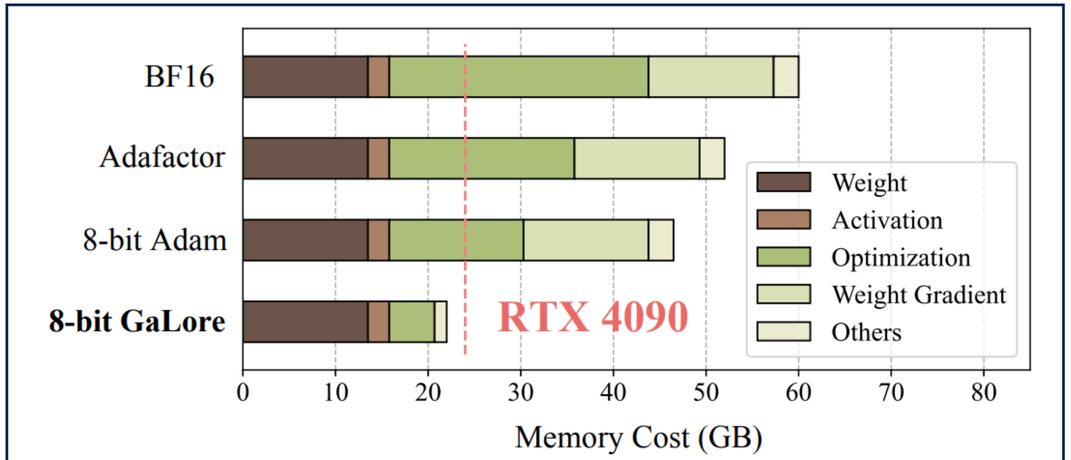


Figure 1: Memory consumption of pre-training a LLaMA 7B model with a token batch size of 256 on a single device, without activation checkpointing and memory offloading. Details refer to Section 5.5.

- LoRA not ideal for full-finetuning
- Here: full-finetuning, but project the gradients to low-rank space (--> reduce memory of Adam statistics etc.)

GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. Zhao et al. 2024
ReLoRA: High-Rank Training Through Low-Rank Updates. Lialin et al. 2023

LoRA family

- LoRA: https://arxiv.org/abs/2106.09685
  - the OG of parameter-efficient finetuning
- VeRA: https://arxiv.org/abs/2310.11454
  - 10x more parameter-efficient than LoRA
- QLoRA https://arxiv.org/abs/2305.14314
  - LoRA on a quantised LLM + tricks
- LoRA-FA: https://arxiv.org/abs/2308.03303
  - 2x more parameter-efficient than LoRA
- OFT: https://arxiv.org/abs/2306.07967
  - LoRA but with orthogonal matrices
- BOFT: https://arxiv.org/abs/2311.06243
  - Upgrade on OFT,
- LoKr: https://arxiv.org/abs/2103.10385
  - Combines two LoRAs via Kronecker product
- LoHa: https://arxiv.org/abs/2108.06098
  - Hadamard product of two LoRA updates
- NOLA: https://arxiv.org/abs/2310.02556
  - Uses learnable Kronecker products of random matrices
- DyLoRA: https://arxiv.org/abs/2210.07558
  - trains LoRA with any ranks and then picks one
- KronA: https://arxiv.org/abs/2212.10650
  - adaptation based on Kronecker products
- Delta-LoRA: https://arxiv.org/abs/2309.02411
  - Incremental updates to the original fully connected layer
- AdaLoRA: https://arxiv.org/abs/2303.10512
  - adaptively allocates the rank of LoRA during training
- LoftQ: https://arxiv.org/abs/2310.08659
  - Initialise LoRA to minimise quantisation error of LLM

Let's check out 1-2 papers?

Can you come up with your own PEFT idea?

Get together with your neighbor and discuss / brainstorm!

NeurIPS deadline is in 4 weeks ;) !

PINs: Positional Insert unlocks object localisation abilities in VLMs

MICHAEL DORKENWALD, NIMROD BARAZANI, CEES G. M. SNOEK*, YUKI M. ASANO*

CVPR'24

Vision-Language Models are great at many things, but not localisation.

Prompt 1: Provide a bounding box around the cat

Prompt 2: Localise the cat in the image

Dorkenwald, Barazani, Snoek, Asano. PINs: Positional Insert unlocks object localisation abilities in VLMs, CVPR'24.

Our solution: *unlock* localisation abilities in frozen VLMs

VLMs are bad at localising and cannot handle the bbox detection task

But (somewhat noisy) localisation does emerge in some VLMs

Try to **unlock** the forgotten localisation abilities in **frozen VLMs**

Dorkenwald, Barazani, Snoek, Asano. PINs: Positional Insert unlocks object localisation abilities in VLMs, CVPR'24.

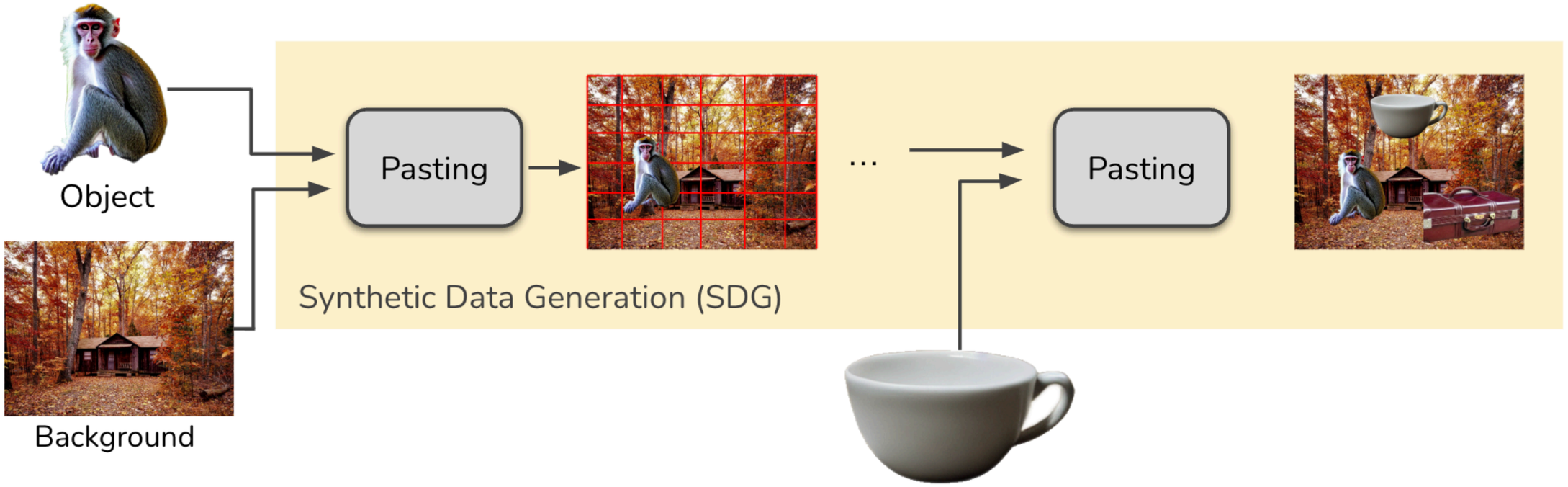
Our approach

frozen VLM, e.g. Flamingo

Positional Insert (PIN) module

Synthetic, unlabeled data

The data



- Because we paste the object, we know it's location
- By pasting multiple objects, we avoid the model focusing on artifacts

Zhao et al. X-Paste: Revisiting Scalable Copy-Paste for Instance Segmentation using CLIP and StableDiffusion. ICML 2023

Example generated data

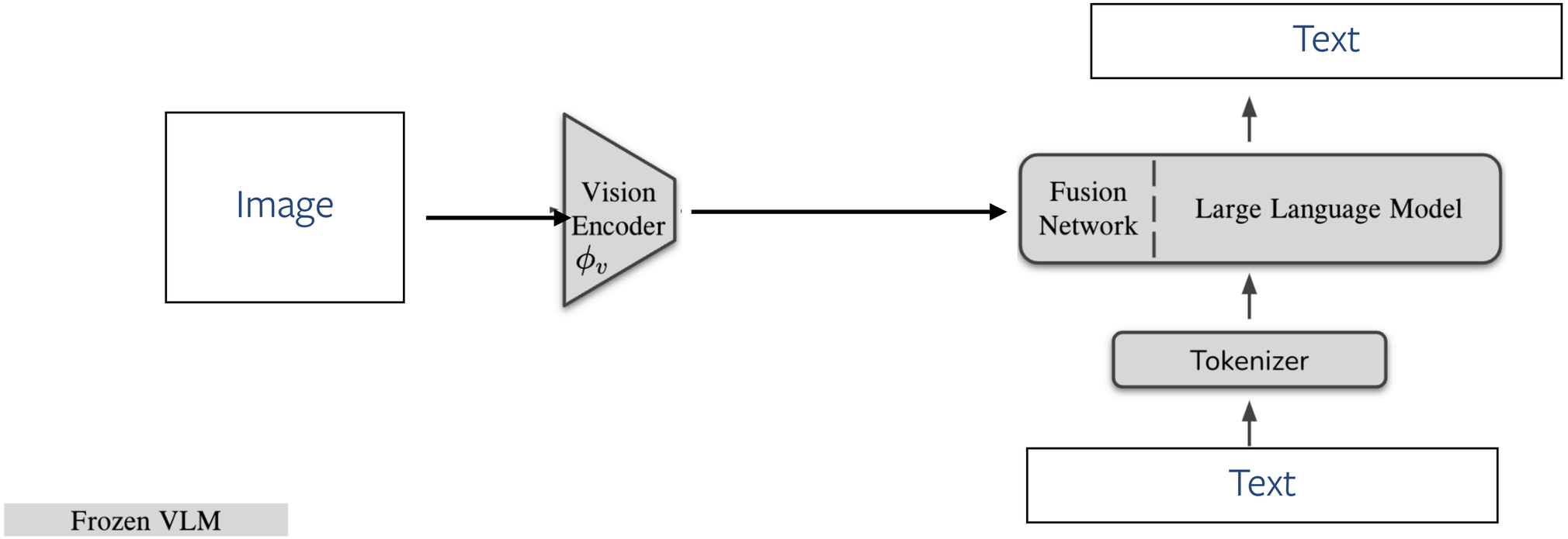




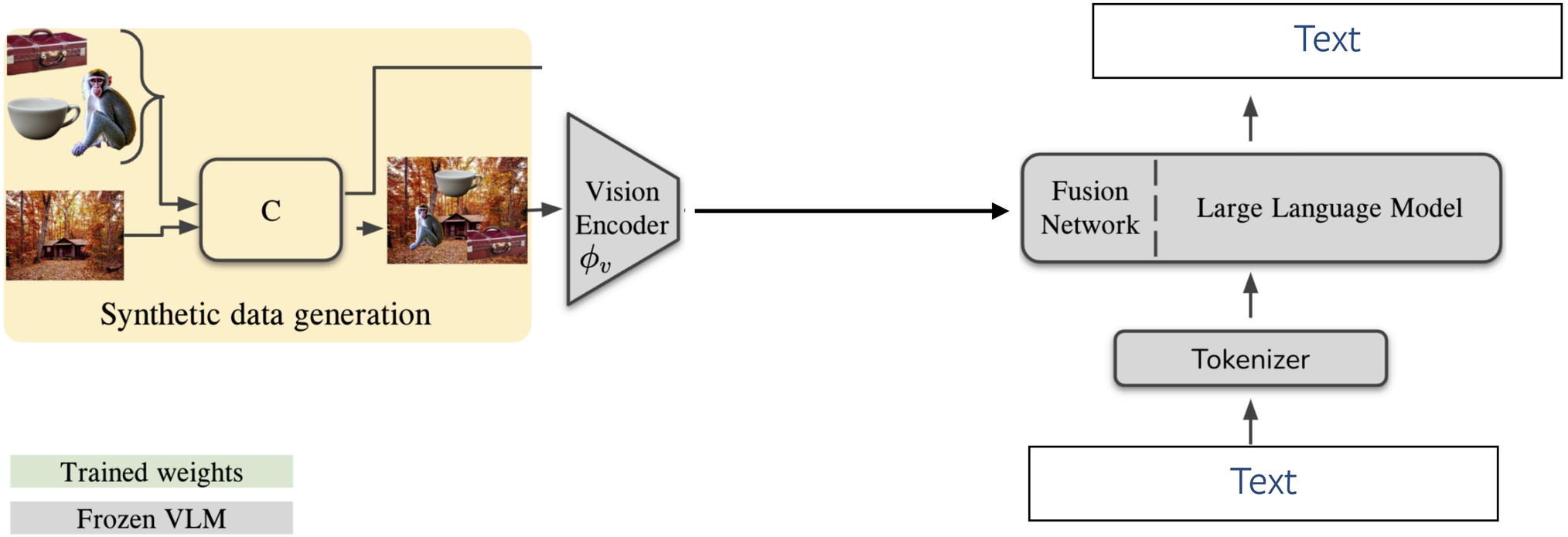
- Note: non-realism is not an issue, as we keep the vision encoder completely frozen
- We only paste objects from categories non-overlapping with our test data
- This means we're in the zero-shot transfer case

Dorkenwald, Barazani, Snoek, Asano. PINs: Positional Insert unlocks object localisation abilities in VLMs, CVPR'24.

Default Flamingo

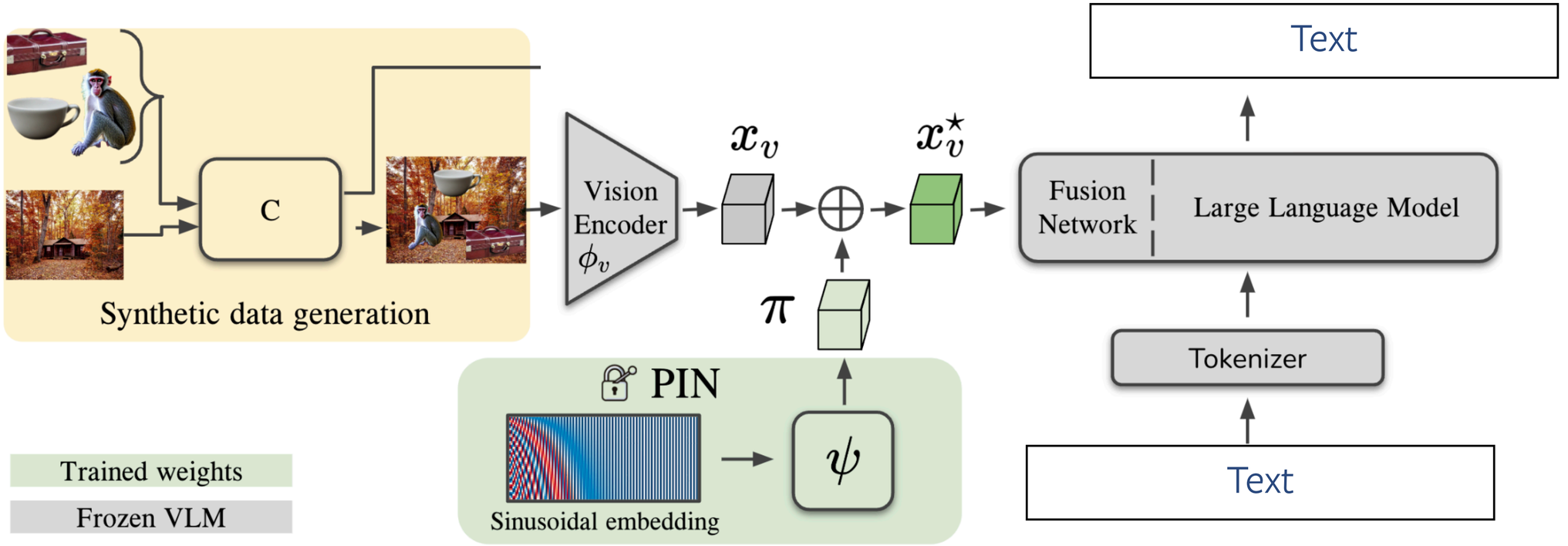


Our method 1: feed the frozen vision encoder synthetic data



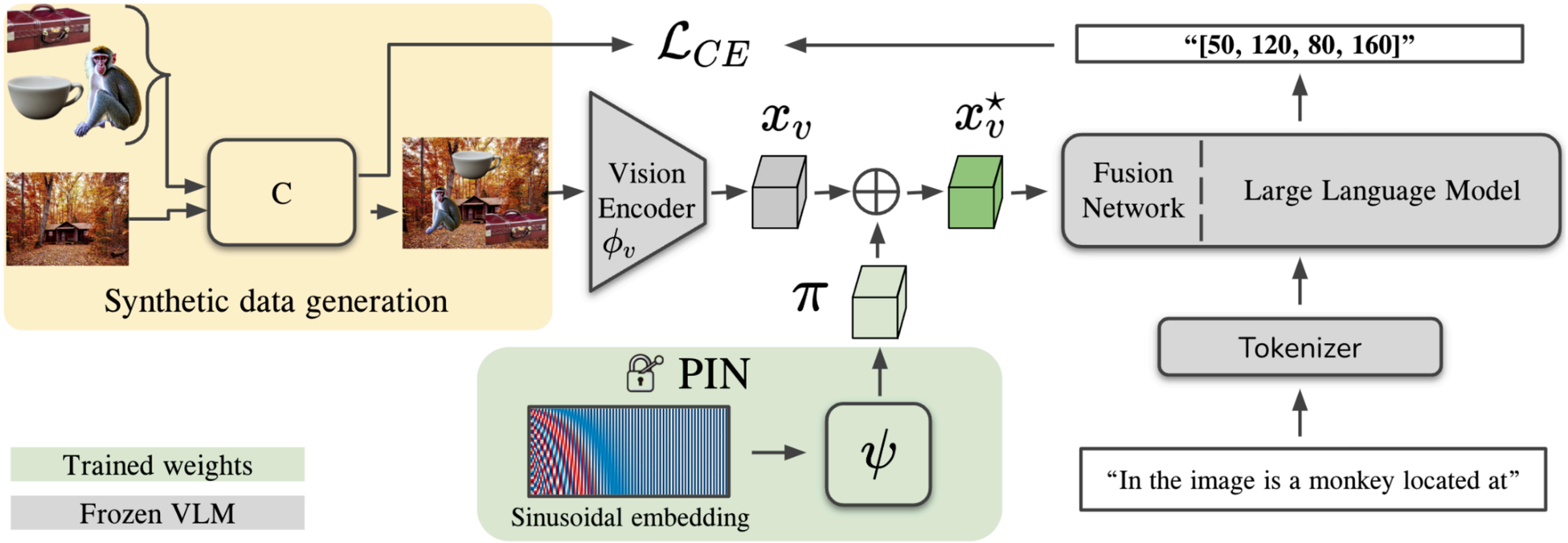
Dorkenwald, Barazani, Snoek, Asano. PINs: Positional Insert unlocks object localisation abilities in VLMs, CVPR'24.

Our method 2: provide VLM spatial learning capacity



Dorkenwald, Barazani, Snoek, Asano. PINs: Positional Insert unlocks object localisation abilities in VLMs, CVPR'24.

Our method 3: train using pasted obj locations via next-word prediction



Dorkenwald, Barazani, Snoek, Asano. PINs: Positional Insert unlocks object localisation abilities in VLMs, CVPR'24.

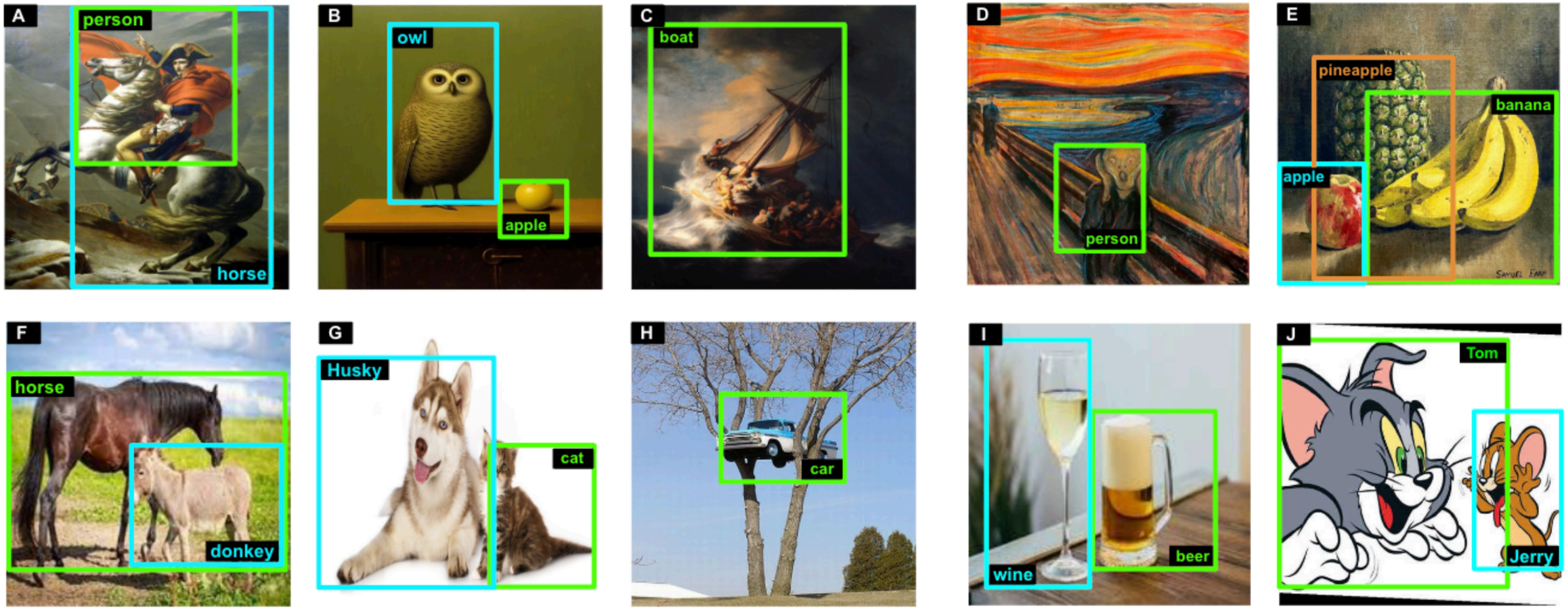
What is the PIN? It's a PEFT method for Vision-Language Models.

```
pos_encoding = get_sinusoid_encoding_table(n_patches=196, d_hid=64)

MLP = nn.Sequential(
    nn.Linear(64, 512),
    nn.SiLU(),
    nn.LayerNorm(512),
    nn.Linear(512, 768),
    nn.SiLU(),
    nn.LayerNorm(768),
    nn.Linear(768, 1024),
)

PIN = MLP(pos_encoding)
```

Results (all on categories not in our pasting-objects)

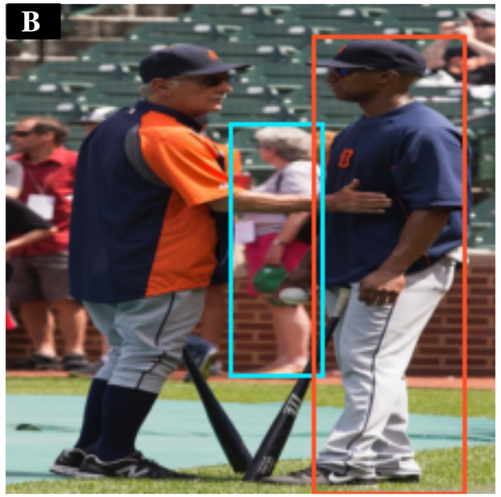


We beat common PEFT methods

	Method	PVOC$_{\leq 3 \text{ Objects}}$ mIoU	PVOC$_{\leq 3 \text{ Objects}}$ mIoU$_M$	PVOC$_{\leq 3 \text{ Objects}}$ mIoU$_L$	COCO$_{\leq 3 \text{ Objects}}$ mIoU	COCO$_{\leq 3 \text{ Objects}}$ mIoU$_M$	COCO$_{\leq 3 \text{ Objects}}$ mIoU$_L$	LVIS$_{\leq 3 \text{ Objects}}$ mIoU	LVIS$_{\leq 3 \text{ Objects}}$ mIoU$_M$	LVIS$_{\leq 3 \text{ Objects}}$ mIoU$_L$
OpenFlamingo [5]	*Baselines*									
	raw	0	0	0	0	0	0	0	0	0
	random	0.22±0.04	0.10±0.02	0.33±0.06	0.12±0.04	0.07±0.02	0.22±0.08	0.07±0.03	0.06±0.02	0.18±0.09
	2 context	0.19±0.11	0.08±0.05	0.30±0.18	0.10±0.08	0.06±0.04	0.18±0.16	0.04±0.06	0.03±0.04	0.10±0.15
	5 context	0.19±0.09	0.07±0.04	0.31±0.15	0.10±0.08	0.06±0.04	0.20±0.16	0.06±0.05	0.04±0.03	0.17±0.13
	10 context	0.20±0.11	0.06±0.03	0.32±0.18	0.09±0.07	0.05±0.04	0.17±0.14	0.05±0.05	0.03±0.03	0.15±0.14
	PEFT									
	CoOp on LLM	0.28	0.11	0.43	0.22	0.10	0.39	0.13	0.07	0.40
	VPT on F	0.34	0.16	0.51	0.26	0.15	0.47	0.19	0.14	0.48
	VPT on ϕ_V	0.42	0.21	0.61	0.33	0.22	0.57	0.23	0.19	0.56
	LoRA on ϕ_V	0.44	0.26	**0.62**	0.33	0.23	0.58	0.23	0.19	0.55
	PIN (ours)	**0.45**	**0.27**	**0.62**	**0.35**	**0.26**	**0.59**	**0.26**	**0.24**	**0.61**
BLIP-2 [32]	*PEFT*									
	VPT on F	0.33	0.12	0.51	0.27	0.12	0.50	0.18	0.11	0.47
	VPT on ϕ_V	0.32	0.12	0.50	0.26	0.11	0.48	0.17	0.10	0.46
	PIN (ours)	**0.44**	**0.24**	**0.63**	**0.34**	**0.22**	**0.60**	**0.26**	**0.23**	**0.60**

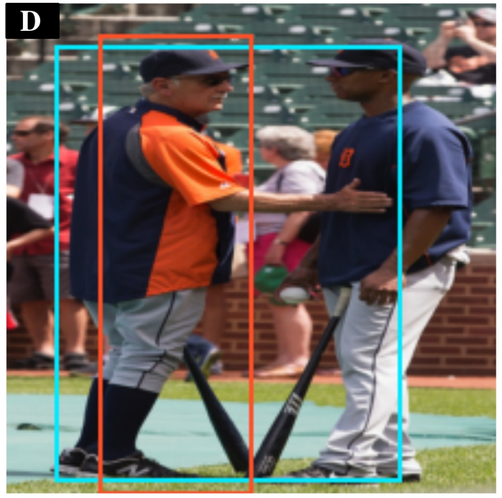
With slight modification, can work on RefCOCO.

"Left black shirt"

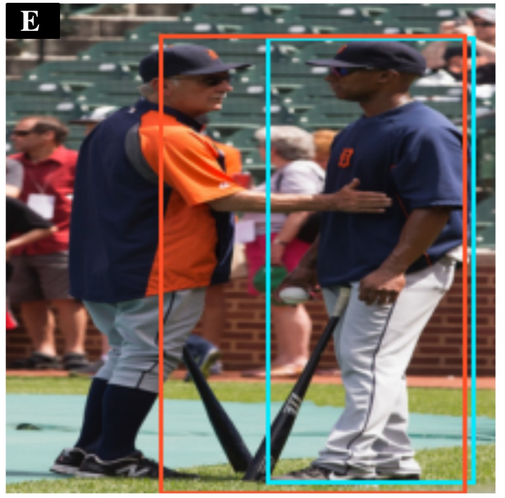


"Old lady in between the players"

"A guy in red on left"



"Guy in orange"



"Right player"

"Top left apron strings"

"Pizza squares left"

"Pizza right front piece in middle"

"A man black"

"A right person"

Predictions

Ground Truth

Some advice for your research projects

- Google slides for progress
  - Progress: reproduction, better numbers, fixing bug..
- Strive for simplicity
  - no more than 2 losses, no excessive hparams
- Share and discuss idea widely (and wildly)
- Do the (smart) unexpected thing
- Be crazy about running lots of experiments and logging them
- Start writing early
- It's still just 1 course: keep your sanity and (phys & mental) health
- You learn most from failure