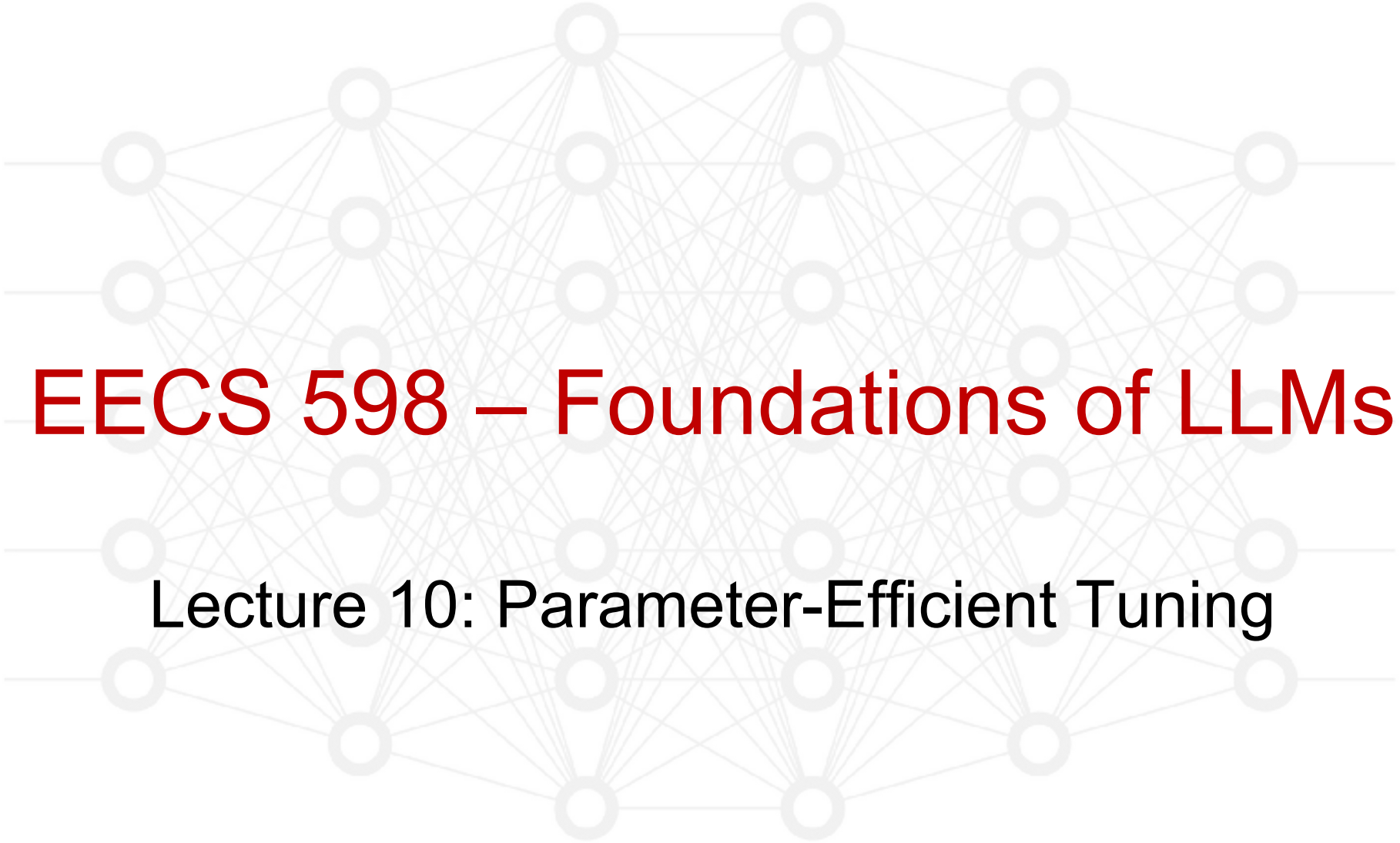


EECS 598 – Foundations of LLMs

Lecture 10: Parameter-Efficient Tuning

Midterm Logistics

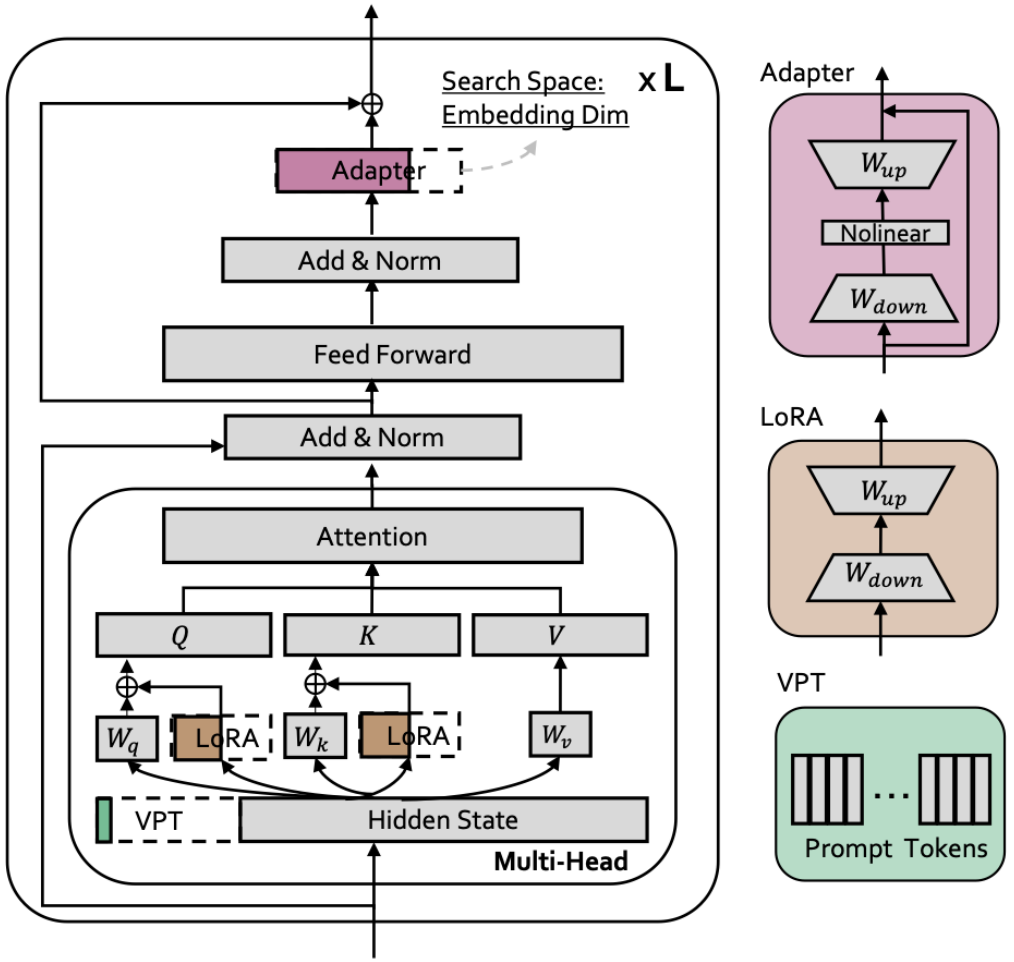
1. Midterm will take place between 12-1:45pm on Wednesday Feb 21.
2. Midterm will be online and be released over Gradescope. It will also be made available under Canvas/Assignments. No need to come to class.
3. You can join the midterm exam over the following link: https://umich.zoom.us/j/91263625624
   - The primary purpose is answering any questions you may have. You don't have to open cameras.
4. Joining midterm via zoom is optional.
5. You have to submit your midterm exam by 1:45pm unless you have a formal reason.
6. A practice midterm with 15 problems is uploaded to Canvas/Assignments.
7. Each problem will have only 1 correct answer. You will not be deduced grades if your answer is incorrect.
8. Midterm is not open-book. Access to slides or the internet resources is not allowed. You can have 2 back-and-forth cheat sheets. It is OK to prepare the cheat sheet on the computer but the font size should be 10 or above.
9. Midterm topics include everything we have covered. This includes Lecture 9 but does not include Associative Recall discussion. In general, if a topic is niche or too complex, it is less likely that we will ask that topic.
10. There will be knowledge questions as well as computational questions. Please see the practice exam for examples.

Midterm Topics (not exclusive)

- Transformer Architecture
- Mathematics of Self-attention
- Copying/Retrieval Problem
- Transformer Training
- Architecture types
  - Encoder-only, Decoder-only…
- Compute Cost of Transformers
- Efficient Attention Approximation
- FlashAttention, Scaling Laws
- Copying/Retrieval Problem
- Positional Encoding
- KV Cache
- Efficient Inference
  - Sliding Window, Attention Sinks
- Scaling Laws & Emergent Abilities
- Recurrent Networks

What is Parameter-Efficient Fine-Tuning?

- PEFT is a strategy in machine learning that involves updating a **small subset** of a model's parameters during the fine-tuning process.



Why PEFT?

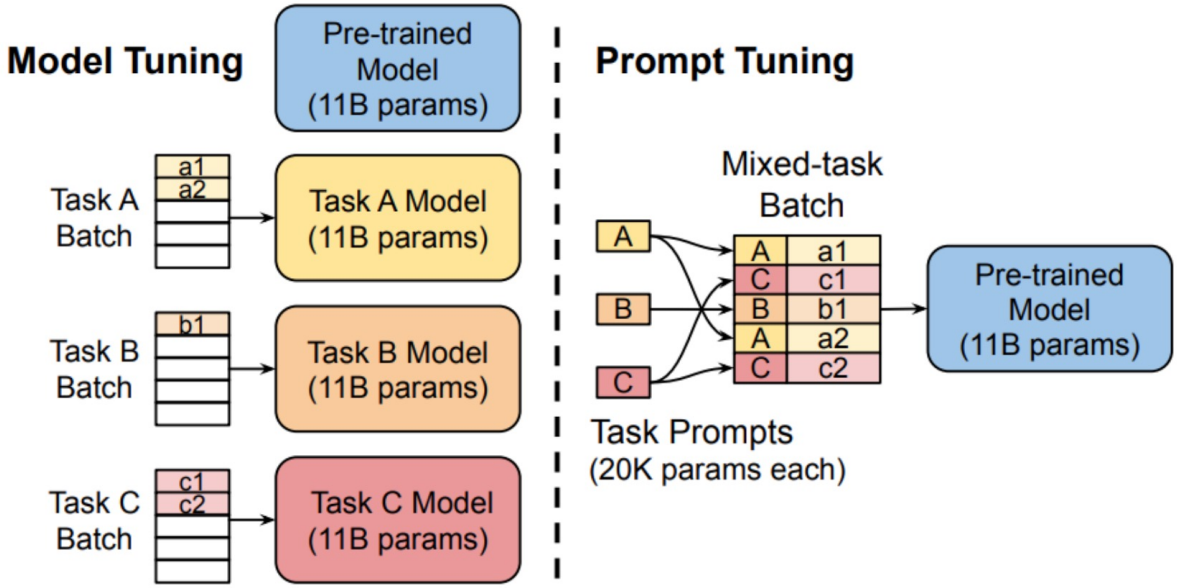
- Fine-tuning
  - Process:
    - Step 1: large-scale pretraining on large amounts of data
    - Step 2: fine-tuning to downstream tasks
  - Bottleneck of full fine-tuning:
    - As models get larger, full fine-tuning becomes infeasible on consumer hardware
    - Storing and deploying fine-tuned models independently for each downstream task becomes very expensive

PEFT approaches are meant to address both problems!

Why PEFT?

– Bottleneck of full fine-tuning:

- As models get larger, full fine-tuning becomes infeasible on consumer hardware
- Storing and deploying fine-tuned models independently for each downstream task becomes very expensive



Why PEFT?

1. **Decreased computational and storage costs:**
   - Up to x1000 less parameters needed. You can store many checkpoints and tasks
   - Reduces memory requirements during fine-tuning (up to x3 for LoRA).
   - Important: PEFT does not really have computation/FLOP benefit during training or inference. You still need same FLOPs for forward/backward passes even if you update few parameters.
2. **Overcoming catastrophic forgetting:** During full fine-tuning of LLMs, catastrophic forgetting can occur where the model forgets the knowledge it learned during pretraining.
3. **Better performance in low-data regimes:** PEFT performs better than full fine-tuning in low-data regimes and generalizes better to out-of-domain scenarios.
4. **Portability:** The trained weights from PEFT approaches are easy to deploy and use for multiple tasks without replacing the entire model.
   Ability to store many checkpoints and many tasks
5. **Performance comparable to full fine-tuning**

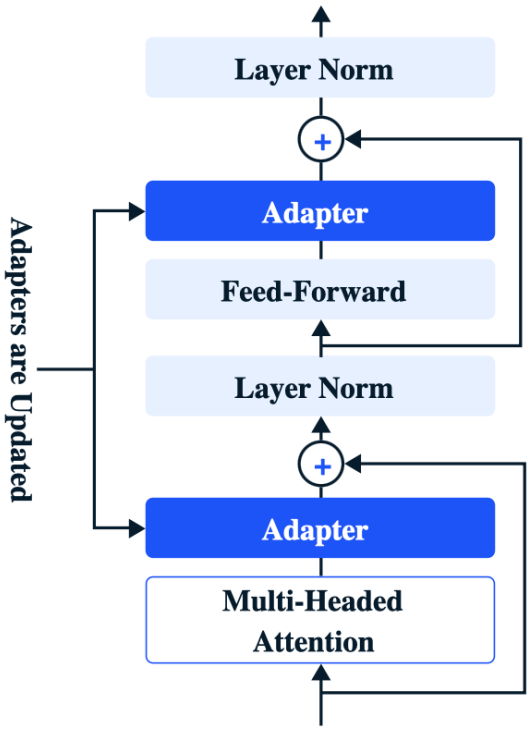
Key Techniques in PEFT

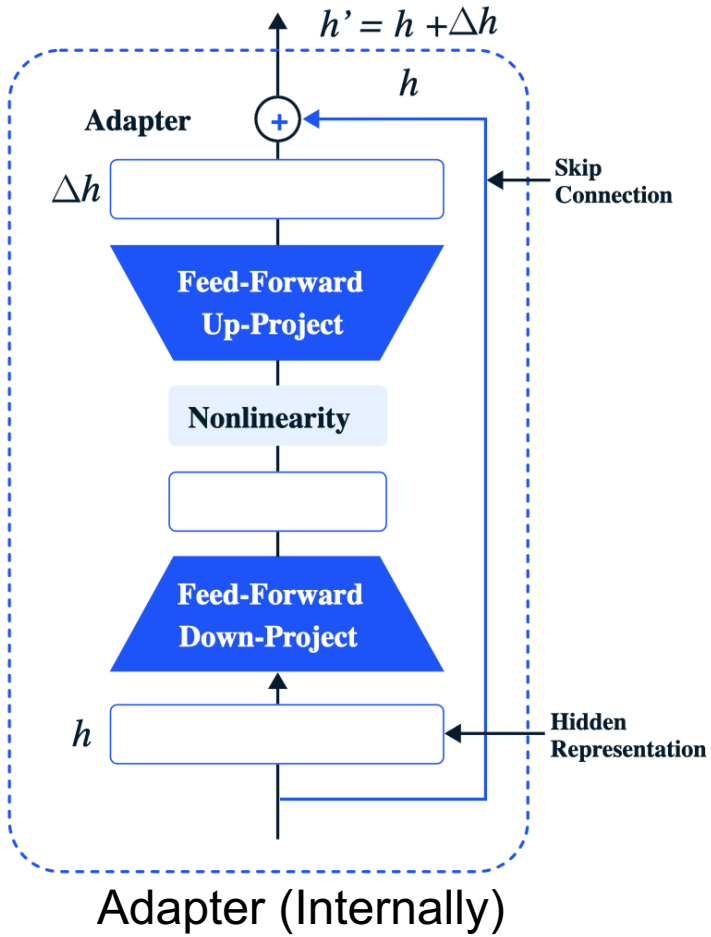
- PEFT is not a one-size-fits-all approach; it encompasses various techniques, each with its unique advantages:
  1. Adapters
  2. Low-Rank Adaptation (LoRA)
  3. Prompt Tuning
  4. BitFit
  5. …

Key Techniques in PEFT

Adapter

- **Left**: Add the adapter module twice to each Transformer layer: after the projection following multi-headed attention and after the two feed-forward layers.
- **Right**: The adapter consists of a bottleneck which contains few parameters relative to the attention and feedforward layers in the original model. The adapter also contains a skip connection.



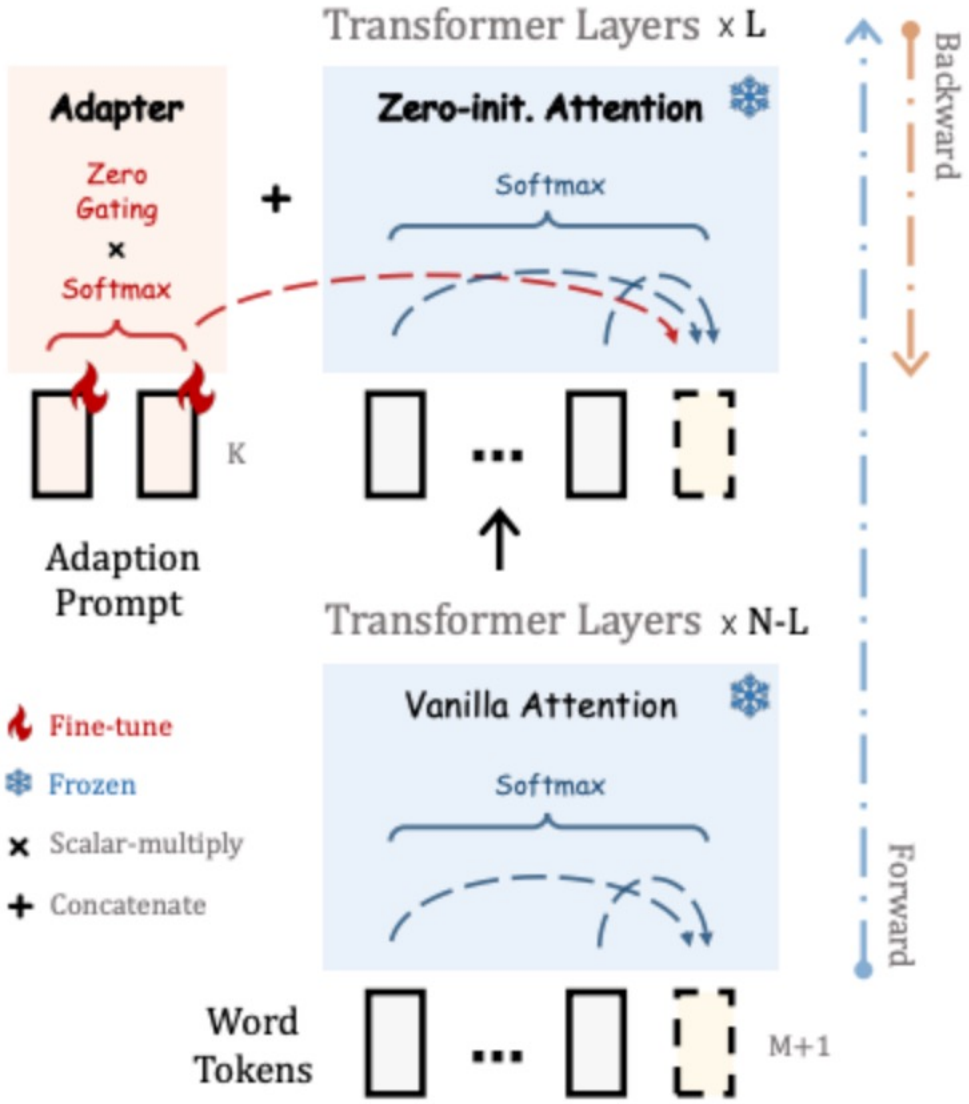


Adapter (Internally)

Key Techniques in PEFT

Llama-Adapter

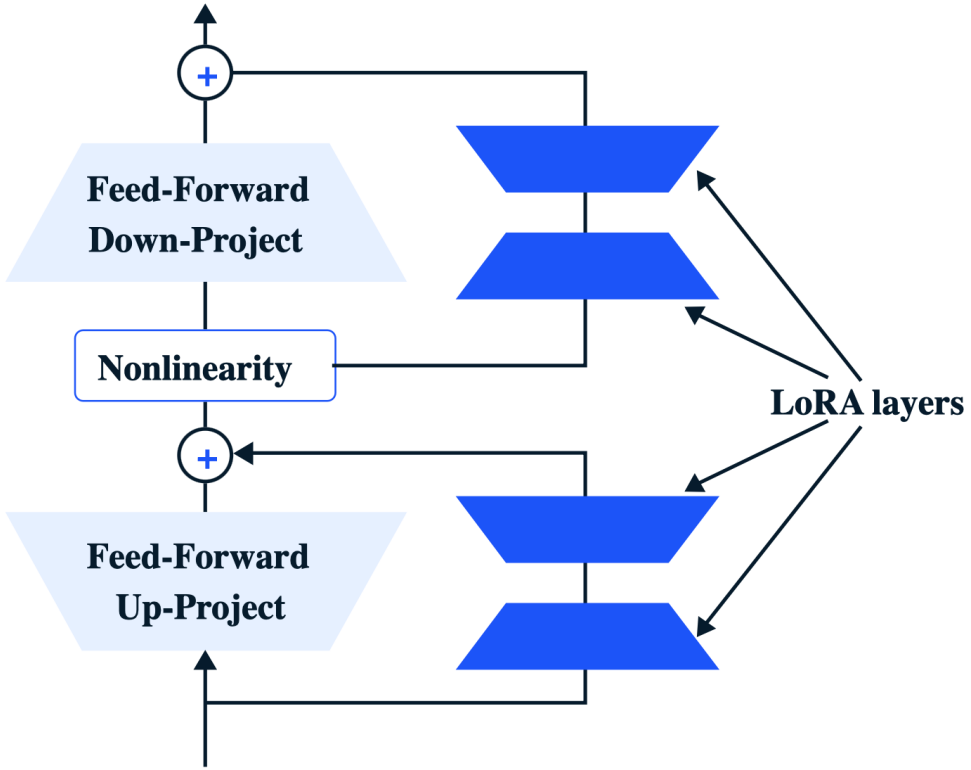
- Llama-Adapter is a method for **adapting Llama into an instruction-following** model.
- A set of of learnable adaption prompts are prefixed to the input instruction tokens. These are inserted into the upper layers of the model because it is better to learn with the higher-level semantics of the pretrained model. The instruction-output tokens prefixed to the input guide the adaption prompt to generate a contextual response.

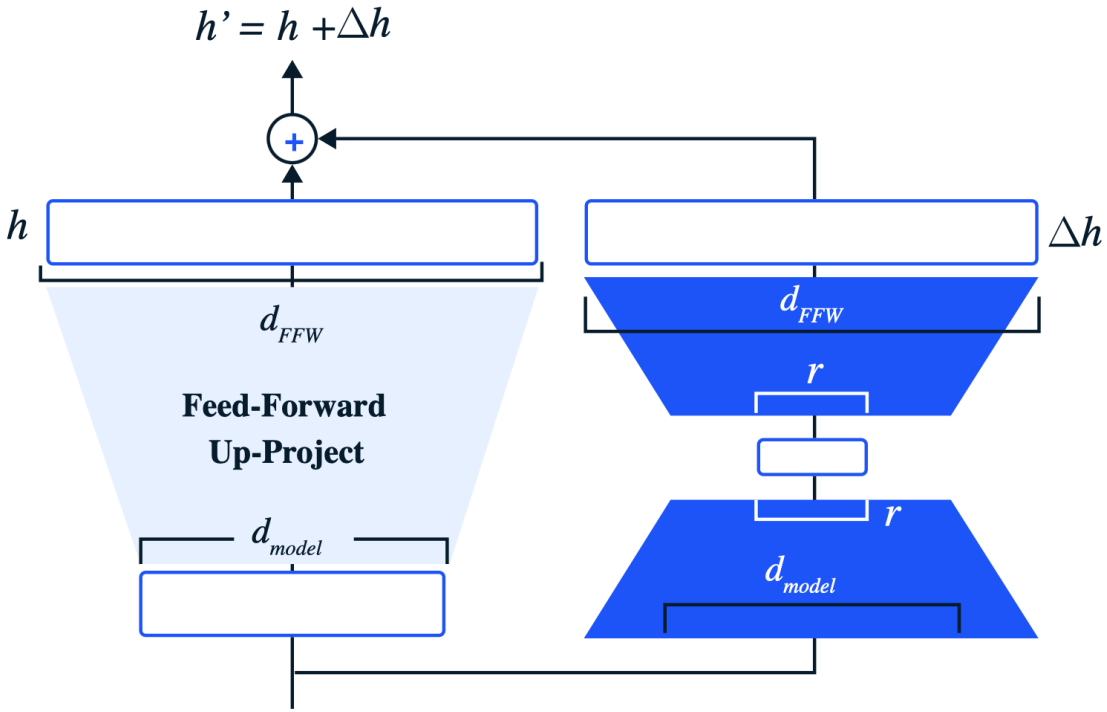


Key Techniques in PEFT

Low-Rank Adaptation (LoRA)

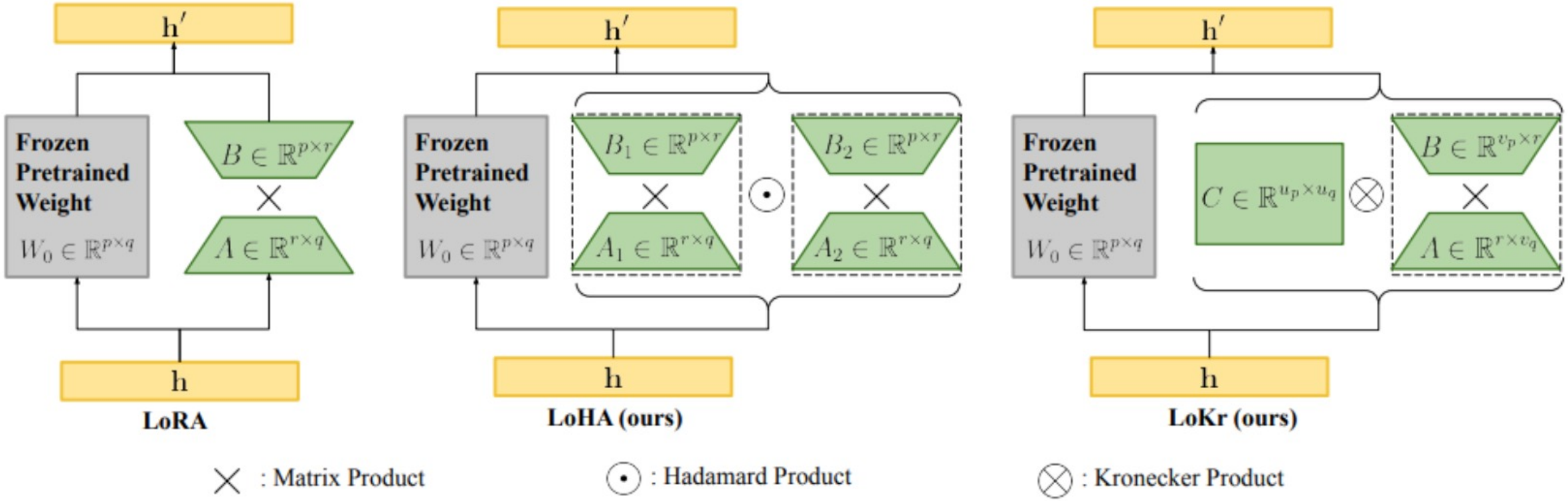
Freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture





Key Techniques in PEFT

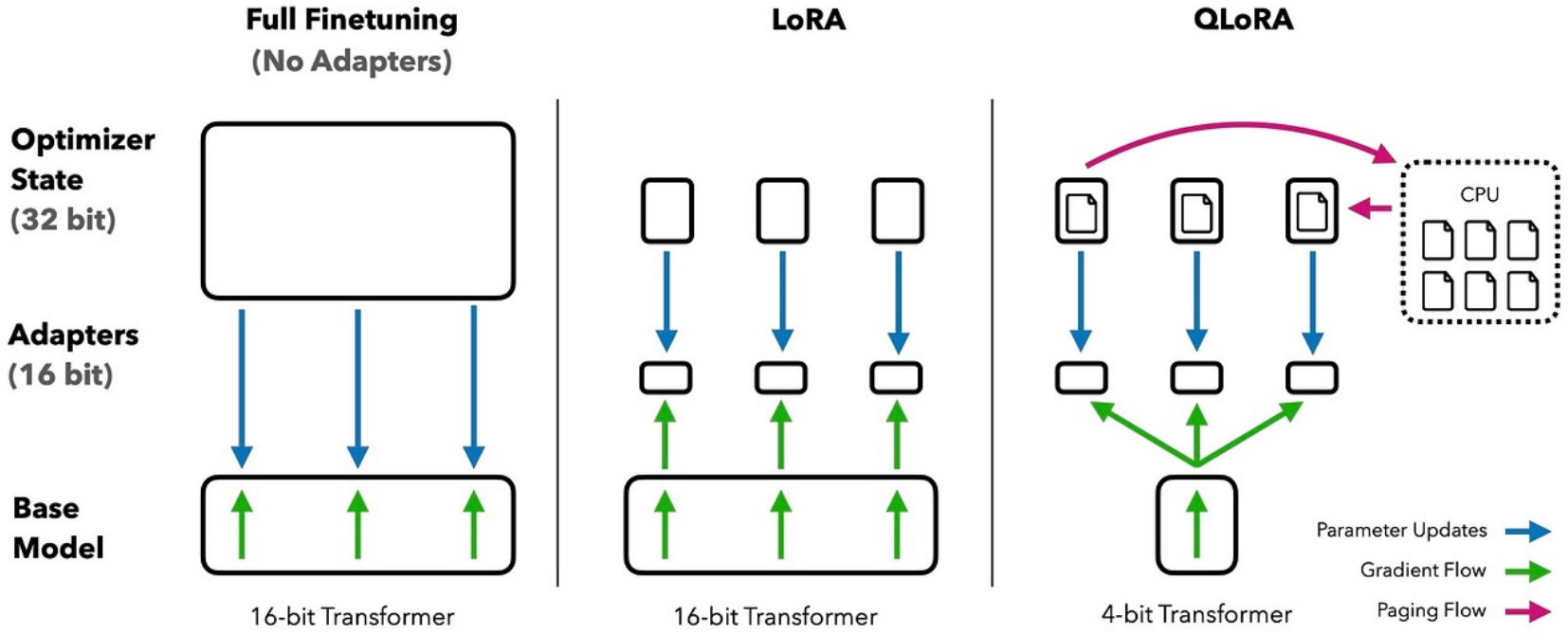
Low-Rank Hadamard Product (LoHa) and Low-Rank Kronecker Product (LoKr)



Key Techniques in PEFT

QLoRA (Quantized Low-Rank Adaptation)

QLoRA is an extension of the PEFT approach for adapting large pre-trained language models like BERT. QLORA backpropagates gradients through a frozen, 4-bit quantized pretrained language model into LoRA parameters.



Key Techniques in PEFT

DoRA: Weight-Decomposed Low-Rank Adaptation

- Weight-Decomposed Low-Rank Adaptation
- Released last week
- **Key observation:** decomposes the pre-trained weight into two components, magnitude and direction

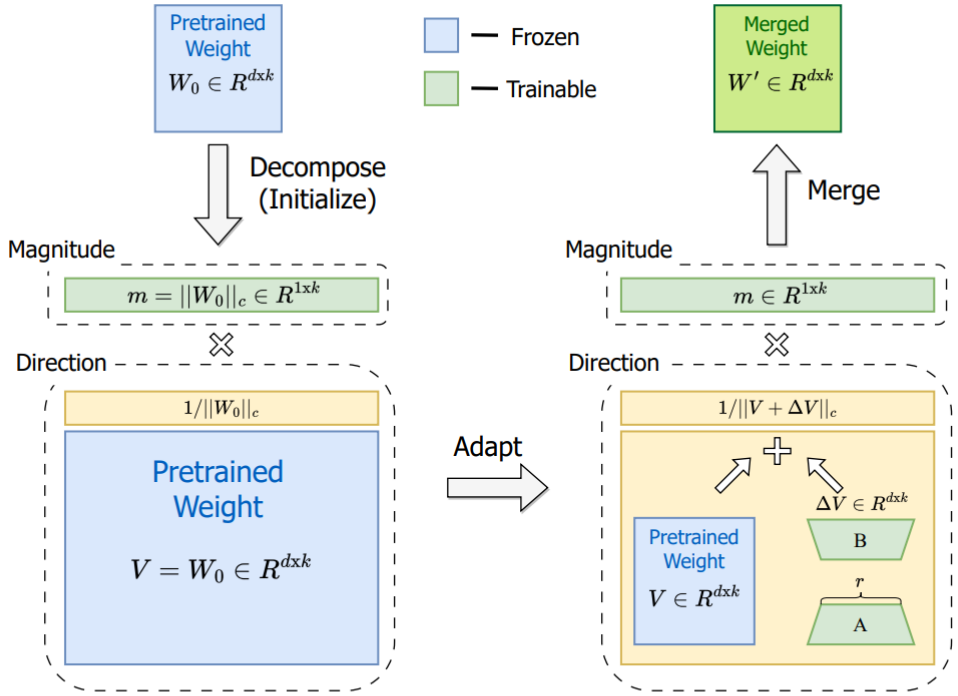


Figure 1. An overview of our proposed DoRA, which decomposes the pre-trained weight into *magnitude* and *direction* components for fine-tuning, especially with LoRA to efficiently update the direction component. Note that $|| \cdot ||_c$ denotes the vector-wise norm of a matrix across each column vector.

DoRA comparison

Table 1. Accuracy comparison of LLaMA 7B/13B with various PEFT methods on eight commonsense reasoning datasets. Results of all the baseline methods are taken from (Hu et al., 2023). DoRA†: the adjusted version of DoRA with the rank halved.

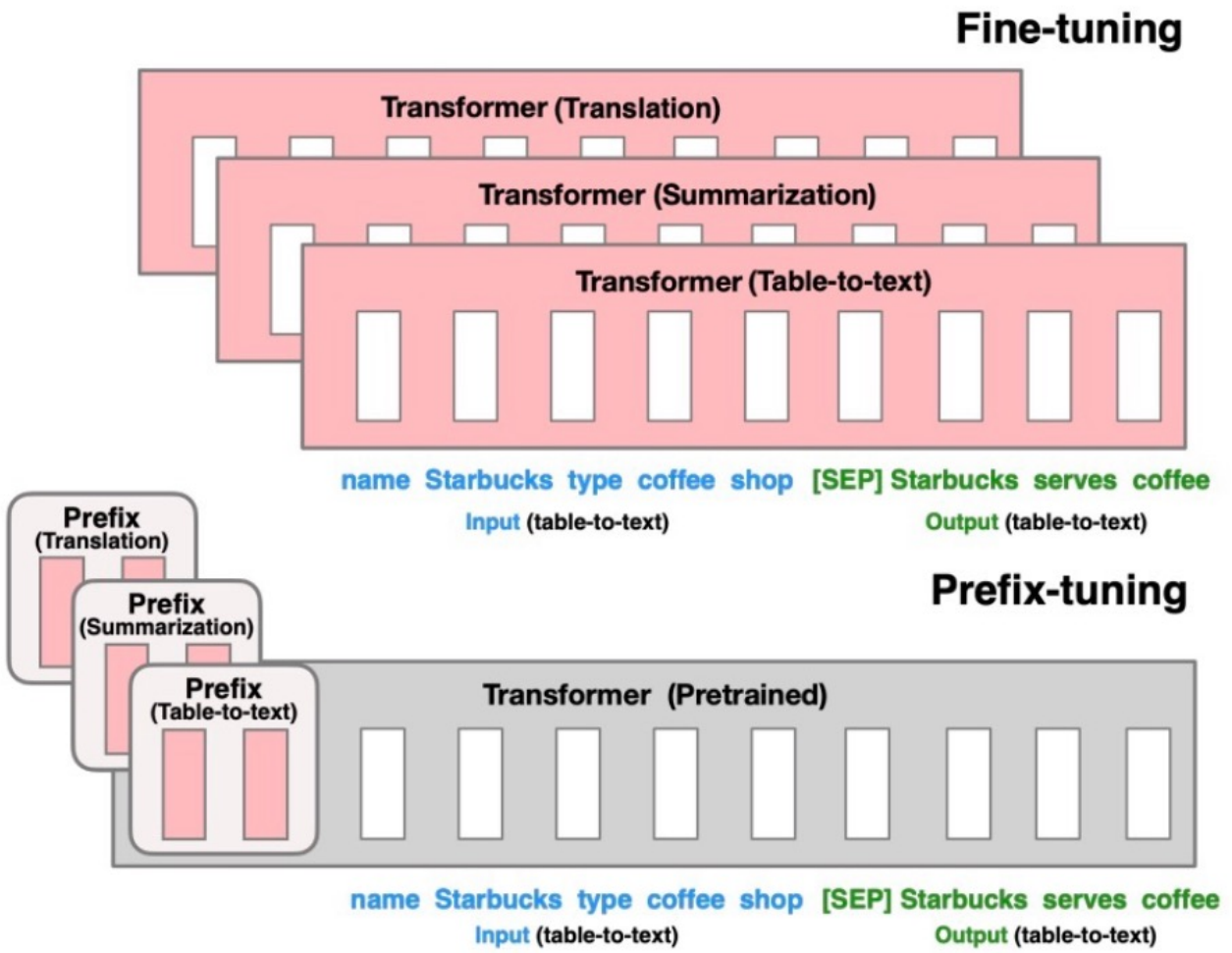
Model	PEFT Method	# Params (%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
ChatGPT	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	Prefix	0.11	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series	0.99	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel	3.54	67.9	76.4	78.8	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	68.5	82.9	79.6	84.8	80.8	81.4	65.8	81.0	**78.1**
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**

Key Techniques in PEFT

Prefix tuning

Prefix-tuning keeps the language model parameters frozen and optimizes a small continuous task-specific vector called the **prefix**.

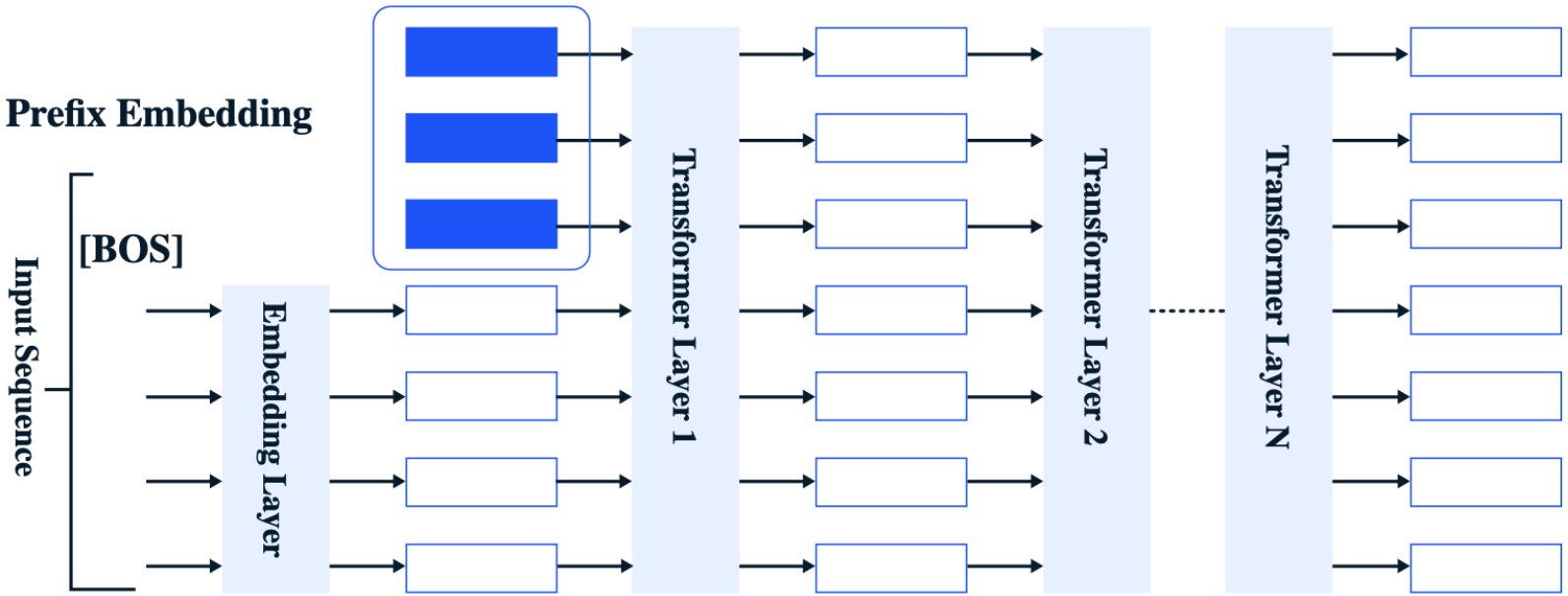
Prefix is a set of learnable tokens



Key Techniques in PEFT

Prompt tuning

Prompt tuning is a simpler variant of prefix tuning. In it, some vectors are prepended at the beginning of a sequence at the input layer.

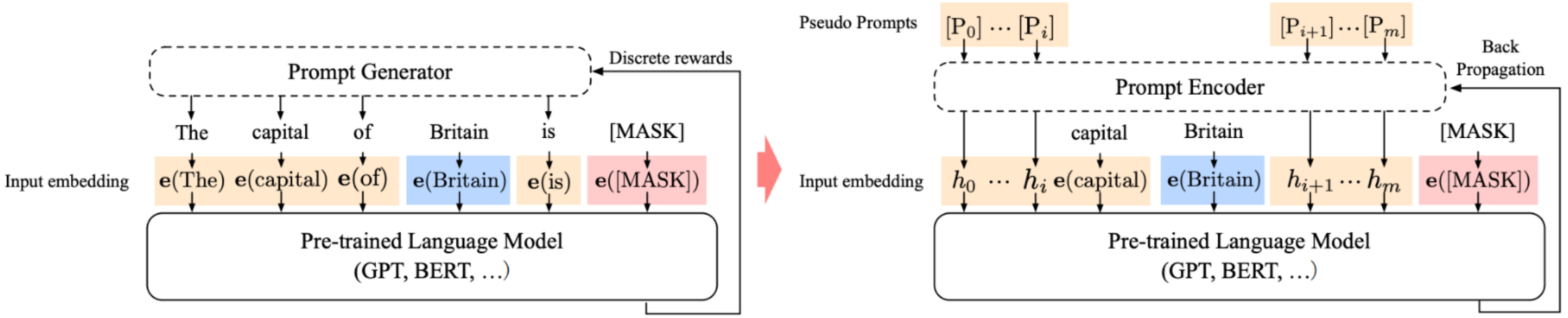


Key Techniques in PEFT

P-tuning

Unlike prefix tuning though:

- the prompt tokens can be inserted anywhere in the input sequence, and it isn't restricted to only the beginning
- the prompt tokens are only added to the input instead of adding them to every layer of the model
- introducing anchor tokens can improve performance because they indicate characteristics of a component in the input sequence



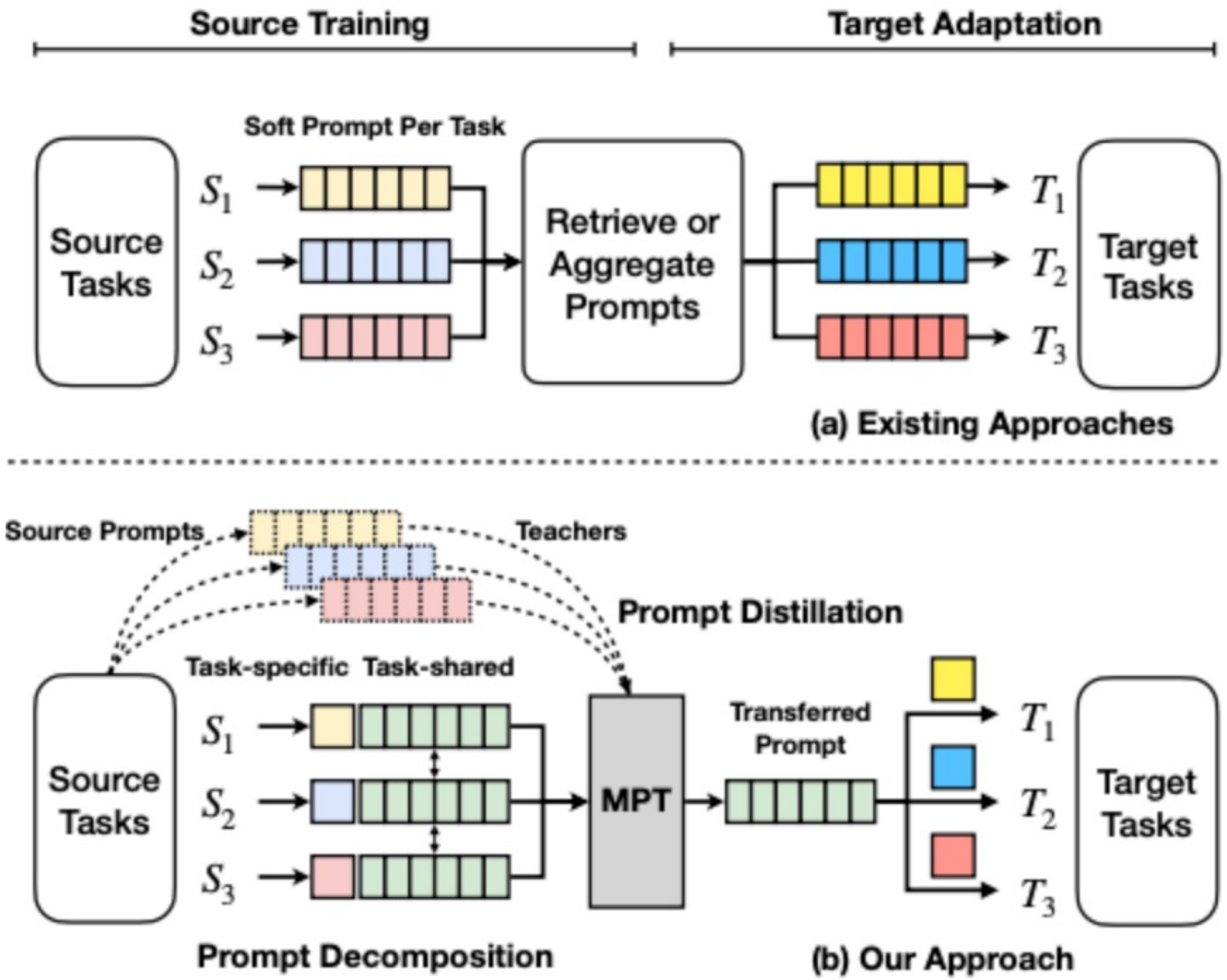
(a) Discrete Prompt Search

(b) P-tuning

Key Techniques in PEFT

Multitask prompt tuning(MPT)

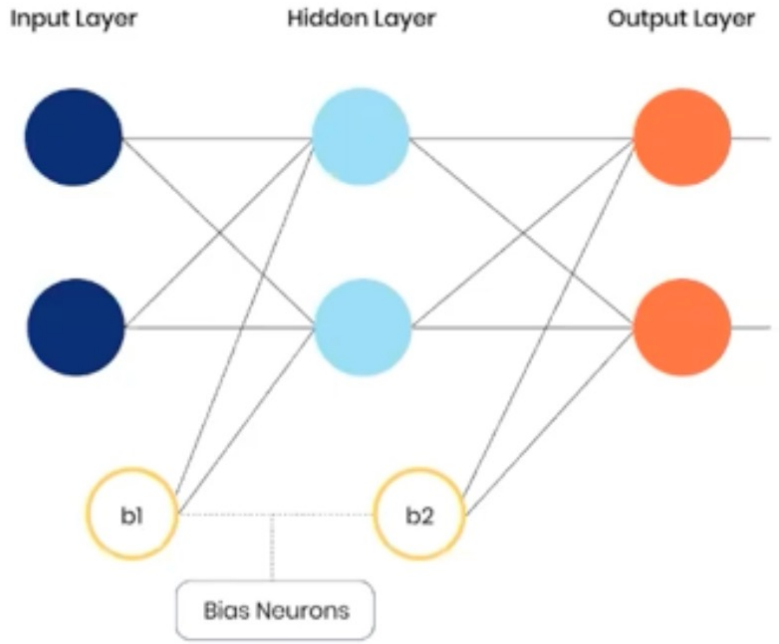
MPT learns a single prompt from data for multiple task types that can be shared for different target tasks.



Key Techniques in PEFT

BitFit

Train only the bias-terms and the task-specific classification layer



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

$$\mathbf{h}_1^\ell = att\left(\mathbf{Q}^{1,\ell}, \mathbf{K}^{1,\ell}, \mathbf{V}^{1,\ell}, .., \mathbf{Q}^{m,\ell}, \mathbf{K}^{m,\ell}, \mathbf{V}^{m,l}\right)$$

and then fed to an MLP with layer-norm (LN):

$$\mathbf{h}_2^\ell = \text{Dropout}\left(\mathbf{W}_{m_1}^\ell \cdot \mathbf{h}_1^\ell + \mathbf{b}_{m_1}^\ell\right) \quad (1)$$
$$\mathbf{h}_3^\ell = \mathbf{g}_{LN_1}^\ell \odot \frac{(\mathbf{h}_2^\ell + \mathbf{x}) - \mu}{\sigma} + \mathbf{b}_{LN_1}^\ell \quad (2)$$
$$\mathbf{h}_4^\ell = \text{GELU}\left(\mathbf{W}_{m_2}^\ell \cdot \mathbf{h}_3^\ell + \mathbf{b}_{m_2}^\ell\right) \quad (3)$$
$$\mathbf{h}_5^\ell = \text{Dropout}\left(\mathbf{W}_{m_3}^\ell \cdot \mathbf{h}_4^\ell + \mathbf{b}_{m_3}^\ell\right) \quad (4)$$
$$\text{out}^\ell = \mathbf{g}_{LN_2}^\ell \odot \frac{(\mathbf{h}_5^\ell + \mathbf{h}_3^\ell) - \mu}{\sigma} + \mathbf{b}_{LN_2}^\ell \quad (5)$$

Recent ideas

VISION TRANSFORMERS NEED REGISTERS

Timothée Darcet[1,2], **Maxime Oquab**[1], **Julien Mairal**[2] **& Piotr Bojanowski**[1]
[1] FAIR, Meta
[2] INRIA
{timdarcet,qas,bojanowski}@meta.com
julien.mairal@inria.fr

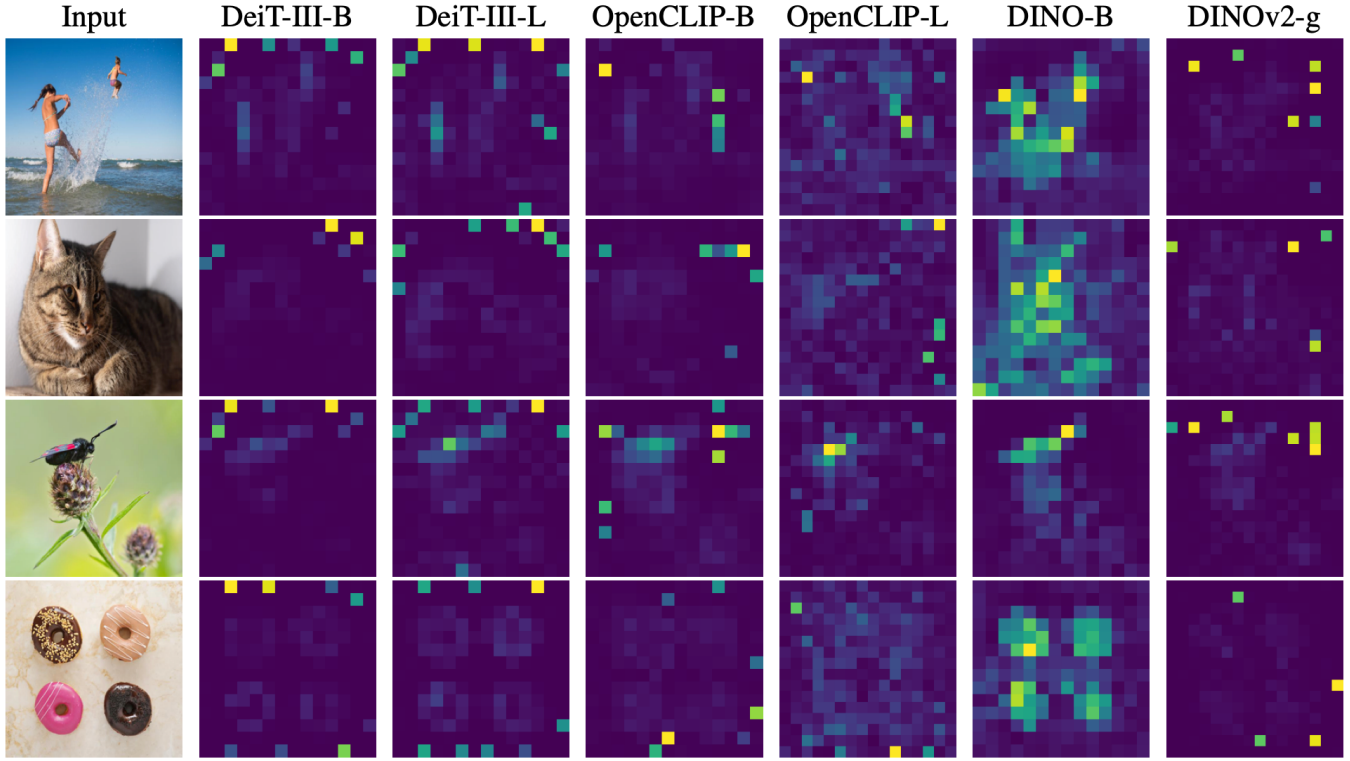
ABSTRACT

Transformers have recently emerged as a powerful tool for learning visual representations. In this paper, we identify and characterize artifacts in feature maps of both supervised and self-supervised ViT networks. The artifacts correspond to high-norm tokens appearing during inference primarily in low-informative background areas of images, that are repurposed for internal computations. We propose a simple yet effective solution based on providing additional tokens to the input sequence of the Vision Transformer to fill that role. We show that this solution fixes that problem entirely for both supervised and self-supervised models, sets a new state of the art for self-supervised visual models on dense visual prediction tasks, enables object discovery methods with larger models, and most importantly leads to smoother feature maps and attention maps for downstream visual processing.

Recent ideas

Motivation

Artifacts observed in the attention maps of modern vision transformers



Recent ideas

Motivation

Artifacts are high-norm outlier tokens.

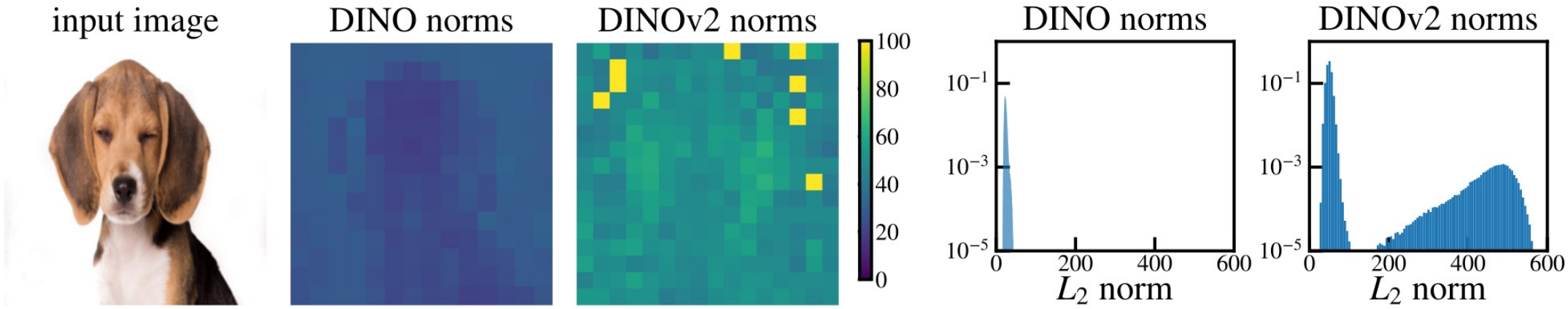


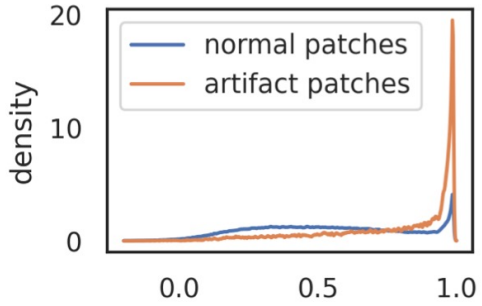
Figure 3: Comparison of local feature norms for DINO ViT-B/16 and DINOv2 ViT-g/14. We observe that DINOv2 has a few outlier patches, whereas DINO does not present these artifacts. For DINOv2, although most patch tokens have a norm between 0 and 100, a small proportion of tokens have a very high norm. We measure the proportion of tokens with norm larger than 150 at 2.37%.

Recent ideas

Motivation

What data does the high-norm feature capture?

- Lacking spatial information



(a) Cosine similarity to neighbors.

	position prediction		reconstruction
	top-1 acc	avg. distance ↓	L2 error ↓
normal	**41.7**	**0.79**	**18.38**
outlier	22.8	5.09	25.23

(b) Linear probing for local information.

Figure 5: **(a)**: Distribution of cosine similarity between input patches and their 4 neighbors. We plot separately artifact patches (norm of the *output token* over 150) and normal patches. **(b)**: Local information probing on normal and outlier patch tokens. We train two models: one for predicting position, and one for reconstructing the input patch. Outlier tokens have much lower scores than the other tokens, suggesting they are storing less local patch information.

Recent ideas

Motivation

What data does the high-norm feature capture?

- Lacking spatial information
- Hold global information

	IN1k	P205	Airc.	CF10	CF100	CUB	Cal101	Cars	DTD	Flow.	Food	Pets	SUN	VOC
[CLS]	**86.0**	**66.4**	**87.3**	**99.4**	**94.5**	**91.3**	96.9	**91.5**	**85.2**	**99.7**	**94.7**	**96.9**	**78.6**	89.1
normal	65.8	53.1	17.1	97.1	81.3	18.6	73.2	10.8	63.1	59.5	74.2	47.8	37.7	70.8
outlier	69.0	55.1	79.1	99.3	93.7	84.9	**97.6**	85.2	84.9	99.6	93.5	94.1	78.5	**89.7**

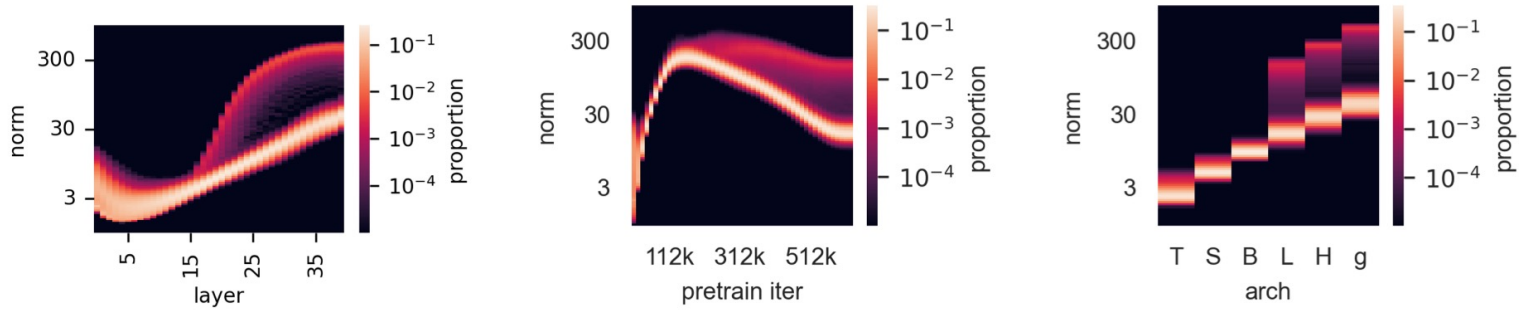
Table 1: Image classification via linear probing on normal and outlier patch tokens. We also report the accuracy of classifiers learnt on the class token. We see that outlier tokens have a much higher accuracy than regular ones, suggesting they are effectively storing global image information.

Recent ideas

Motivation

When does it happen?

- Large and sufficiently trained models



(a) Norms along layers. (b) Norms along iterations. (c) Norms across model size.

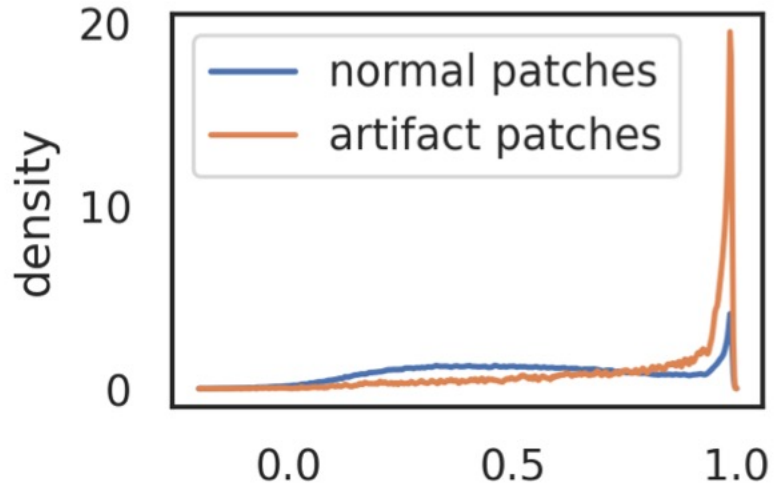
Figure 4: Illustration of several properties of outlier tokens in the 40-layer DINOv2 ViT-g model. **(a)**: Distribution of output token norms along layers. **(b)**: Distribution of norms along training iterations. **(c)**: Distribution of norms for different model sizes. The outliers appear around the middle of the model during training; they appear with models larger than and including ViT-Large.

Recent ideas

Motivation

When does it happen?

- Large and sufficiently trained models
- Learn to recognize redundant tokens



(a) Cosine similarity to neighbors.

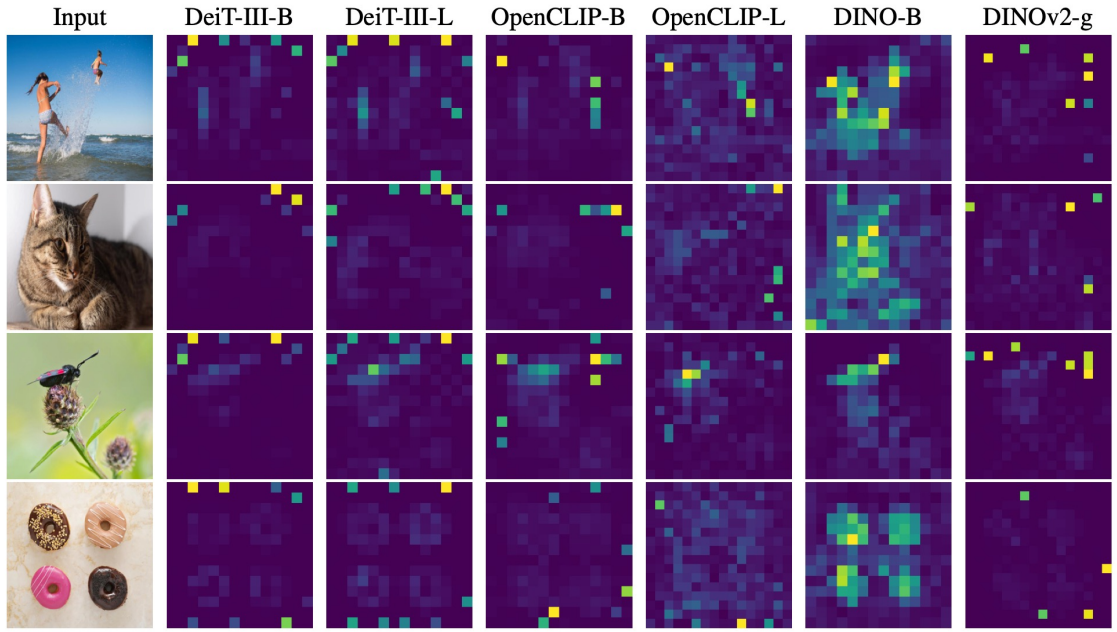
High-norm tokens appear on patches that are very similar to their neighbors. This suggests that these patches contain redundant information and that the model could discard their information without hurting the quality of the image representation.

Recent ideas

Motivation

When does it happen?

- Large and sufficiently trained models
- Learn to recognize redundant tokens

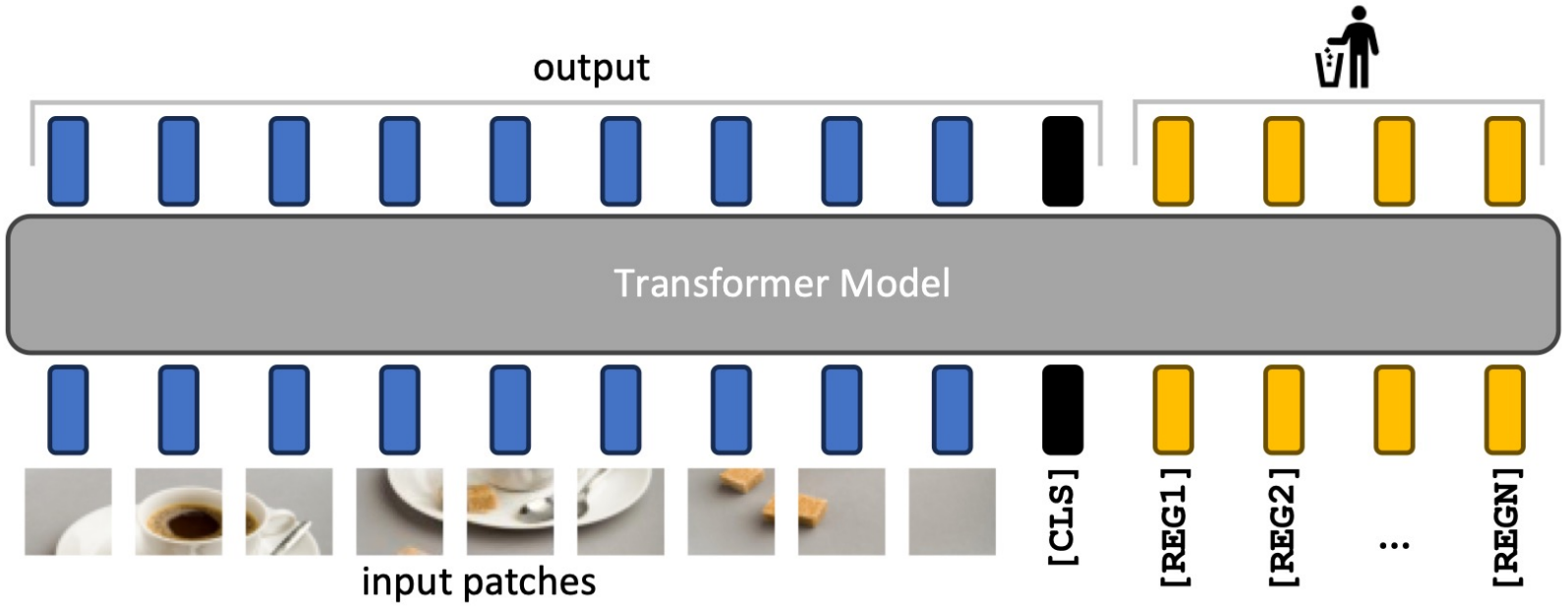


High-norm tokens often appear in uniform, background areas.

Recent ideas

The fix: Registers

We add **N additional learnable input tokens** (depicted in yellow), that the model can use as *registers*. At the output of the model, only the patch tokens and CLS tokens are used, both during training and inference.



Recent ideas

The fix: Registers

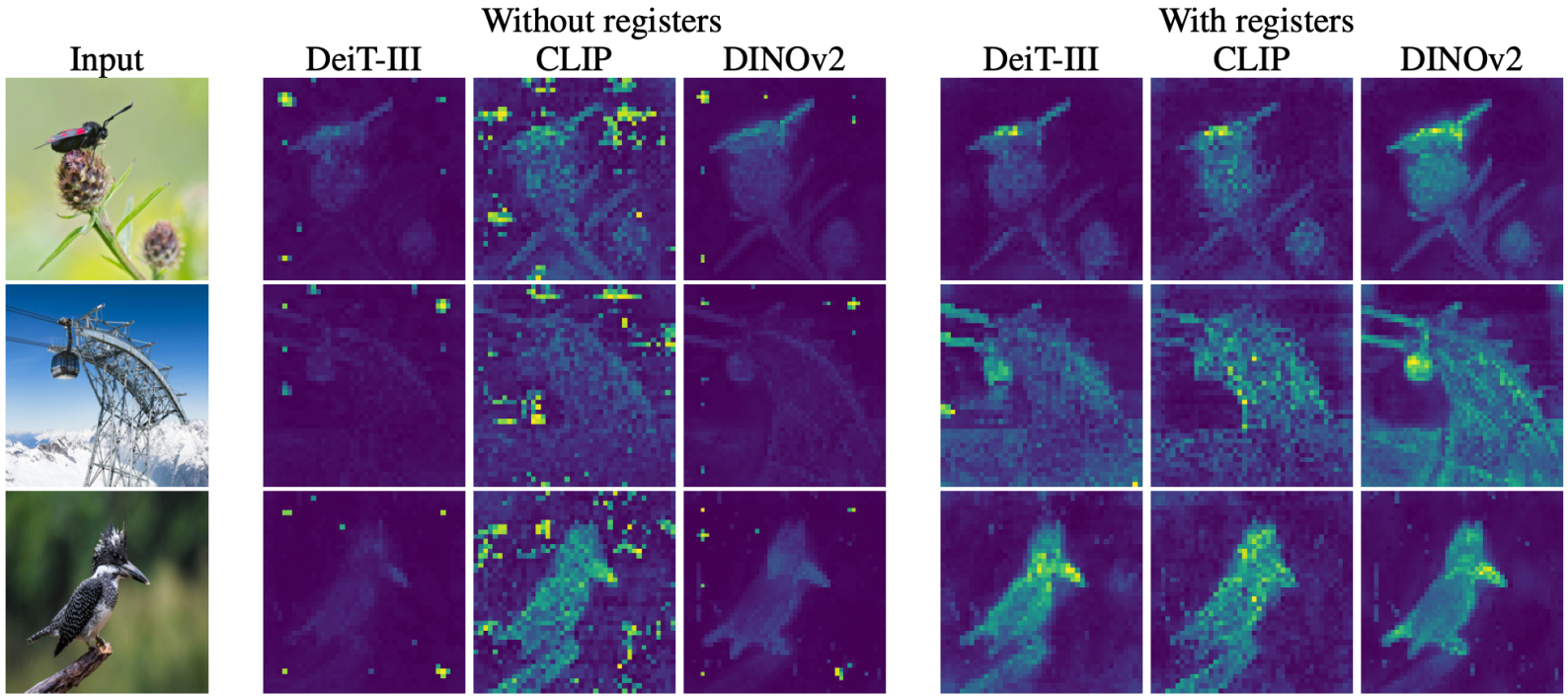


Figure 1: Register tokens enable interpretable attention maps in all vision transformers, similar to the original DINO method (Caron et al., 2021). Attention maps are calculated in high resolution for better visualisation. More qualitative results are available in appendix D.

Recent ideas

The fix: Registers

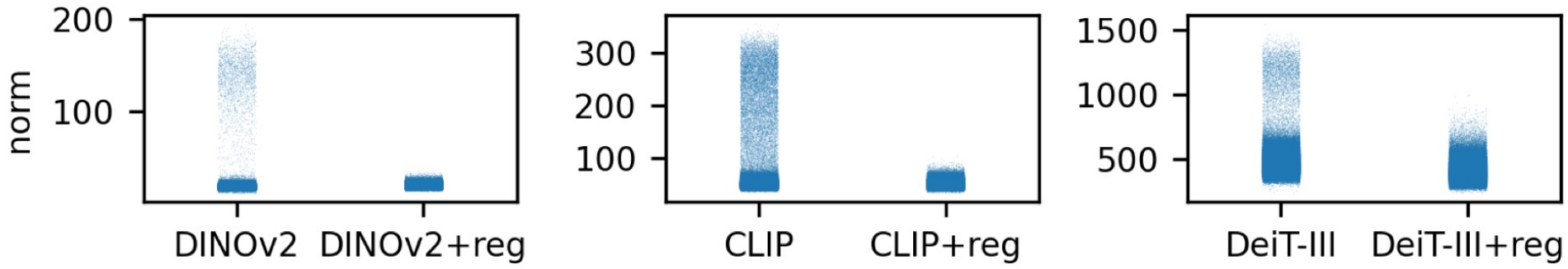


Figure 7: Effect of register tokens on the distribution of output norms on DINOv2, CLIP and DeiT-III. Using register tokens effectively removes the norm outliers that were present previously.

Recent ideas

Results

	ImageNet Top-1	ADE20k mIoU	NYUd rmse ↓
DeiT-III	84.7	38.9	0.511
DeiT-III+reg	84.7	39.1	0.512
OpenCLIP	78.2	26.6	0.702
OpenCLIP+reg	78.1	26.7	0.661
DINOv2	84.3	46.6	0.378
DINOv2+reg	84.8	47.9	0.366

(a) Linear evaluation with frozen features.

	ImageNet Top-1
OpenCLIP	59.9
OpenCLIP+reg	60.1

(b) Zero-shot classification.

Table 2: Evaluation of downstream performance of the models that we trained, with and without registers. We consider linear probing of frozen features for all three models, and zero-shot evaluation for the OpenCLIP model. We see that using register not only does not degrade performance, but even improves it by a slight margin in some cases.