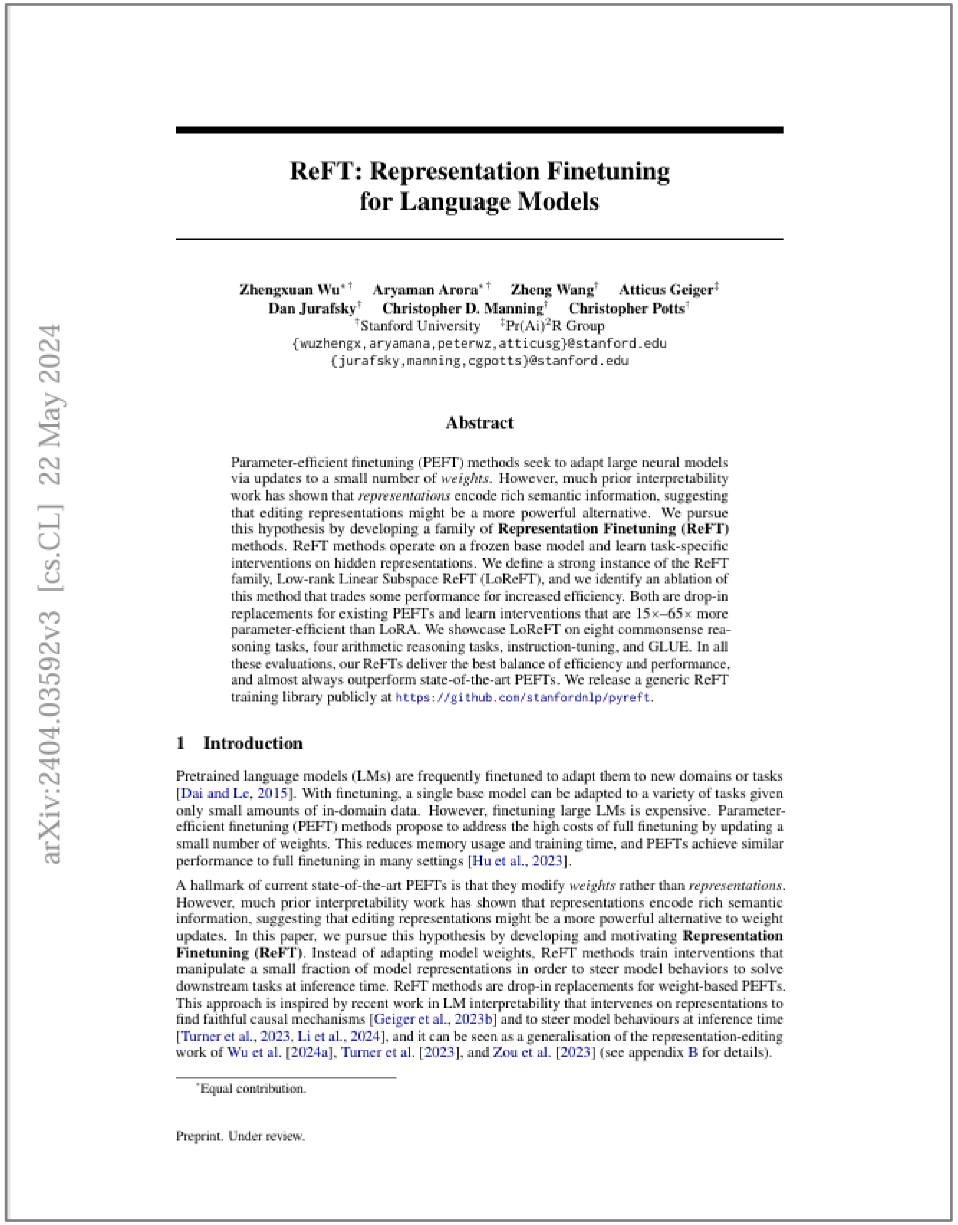
ReFT: Representation Finetuning for Language Models

December 5, 2024

ReFT: Representation Finetuning for Language Models

Zhengxuan Wu et al., Stanford+

https://arxiv.org/abs/2404.03592



arXiv:2404.03592v3 [cs.CL] 22 May 2024

ReFT: Representation Finetuning for Language Models

Zhengxuan Wu[*†] **Aryaman Arora**[*†] **Zheng Wang**[†] **Atticus Geiger**[‡]
Dan Jurafsky[†] **Christopher D. Manning**[†] **Christopher Potts**[†]
[†]Stanford University [‡]Pr(Ai)^{2}R Group
{wuzhengx,aryamana,peterwz,atticusg}@stanford.edu
{jurafsky,manning,cgpotts}@stanford.edu

Abstract

Parameter-efficient finetuning (PEFT) methods seek to adapt large neural models via updates to a small number of *weights*. However, much prior interpretability work has shown that *representations* encode rich semantic information, suggesting that editing representations might be a more powerful alternative. We pursue this hypothesis by developing a family of **Representation Finetuning (ReFT)** methods. ReFT methods operate on a frozen base model and learn task-specific interventions on hidden representations. We define a strong instance of the ReFT family, Low-rank Linear Subspace ReFT (LoReFT), and we identify an ablation of this method that trades some performance for increased efficiency. Both are drop-in replacements for existing PEFTs and learn interventions that are 15×–65× more parameter-efficient than LoRA. We showcase LoReFT on eight commonsense reasoning tasks, four arithmetic reasoning tasks, instruction-tuning, and GLUE. In all these evaluations, our ReFTs deliver the best balance of efficiency and performance, and almost always outperform state-of-the-art PEFTs. We release a generic ReFT training library publicly at https://github.com/stanfordnlp/pyreft.

1 Introduction

Pretrained language models (LMs) are frequently finetuned to adapt them to new domains or tasks [Dai and Le, 2015]. With finetuning, a single base model can be adapted to a variety of tasks given only small amounts of in-domain data. However, finetuning large LMs is expensive. Parameter-efficient finetuning (PEFT) methods propose to address the high costs of full finetuning by updating a small number of weights. This reduces memory usage and training time, and PEFTs achieve similar performance to full finetuning in many settings [Hu et al., 2023].

A hallmark of current state-of-the-art PEFTs is that they modify *weights* rather than *representations*. However, much prior interpretability work has shown that representations encode rich semantic information, suggesting that editing representations might be a more powerful alternative to weight updates. In this paper, we pursue this hypothesis by developing and motivating **Representation Finetuning (ReFT)**. Instead of adapting model weights, ReFT methods train interventions that manipulate a small fraction of model representations in order to steer model behaviors to solve downstream tasks at inference time. ReFT methods are drop-in replacements for weight-based PEFTs. This approach is inspired by recent work in LM interpretability that intervenes on representations to find faithful causal mechanisms [Geiger et al., 2023b] and to steer model behaviours at inference time [Turner et al., 2023, Li et al., 2024], and it can be seen as a generalisation of the representation-editing work of Wu et al. [2024a], Turner et al. [2023], and Zou et al. [2023] (see appendix B for details).

*Equal contribution.

Preprint. Under review.

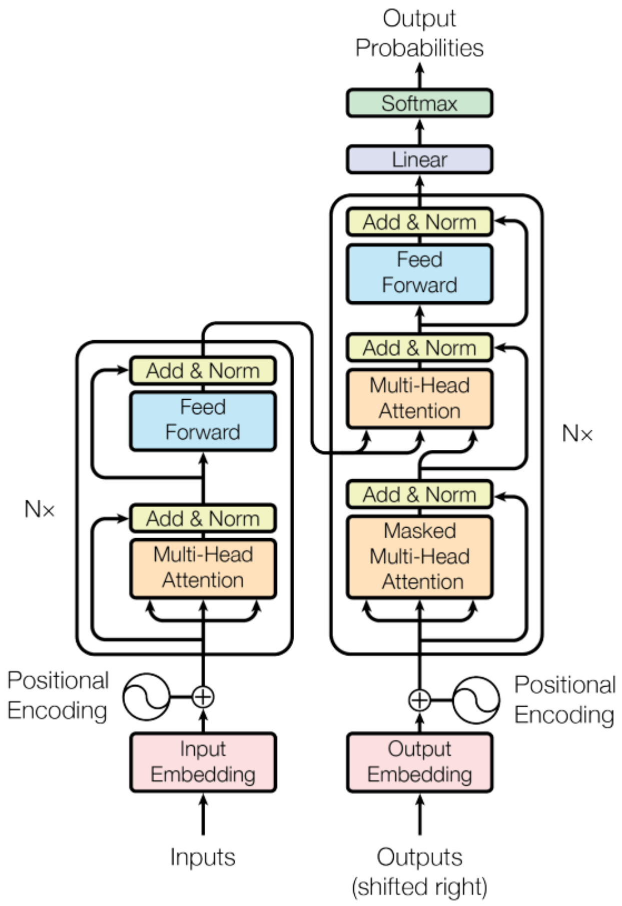
ReFT overview

- Parameter Efficient Fine-Tuning (PEFT) methods, such as LoRA, have gained widespread use as they allow updating of very large LLMs with less memory and compute, making fine-tuning accessible with more modest hardware
- LoRA (and its variants such as DoRA) modify selected weights of a model, such as the query weights of a transformer
  - Memory reduction comes from using low rank versions of the weight matrix
- ReFT directly modifies the activations, not the weights
- The LoReFT instance of ReFT also uses low rank to reduce memory
- Authors report better performance with fewer parameters than PEFT

Related work (§2)

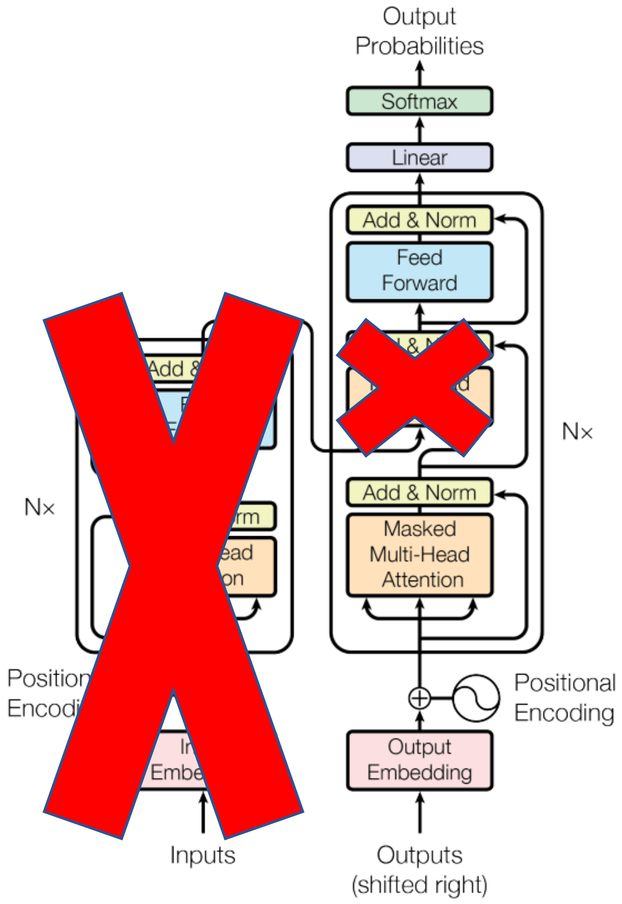
- Parameter Efficient Fine-Tuning (PEFT)
  - Adapter-based methods
    - Add new trainable components either in series or in parallel
    - Modified architecture imposes additional burden at inference time
  - LoRA
    - Low rank matrices in parallel
    - Weight updates can be merged into existing model, so no additional inference
  - Prompt-based methods
    - Add soft tokens to input. No weight changes.
- Representation editing – activation directions & modifying activations
- Interventional interpretability – causal experiments with activations

Original Transformer

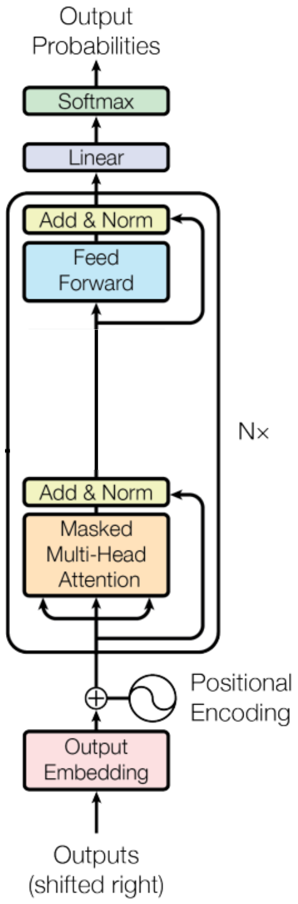


"Attention Is All You Need," Vaswani et al., https://arxiv.org/abs/1706.03762

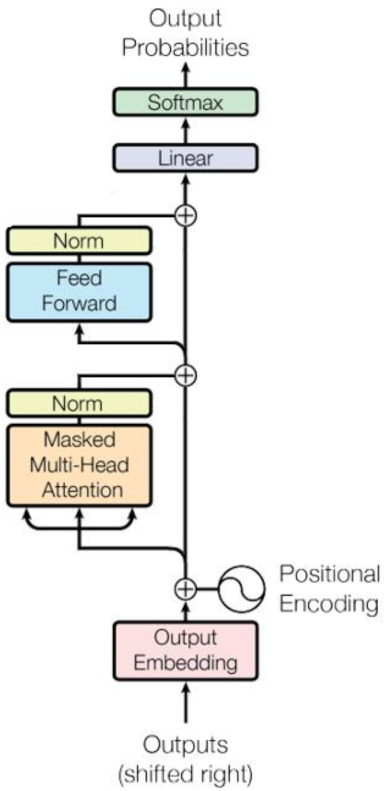
Decoder-only Transformer – 1



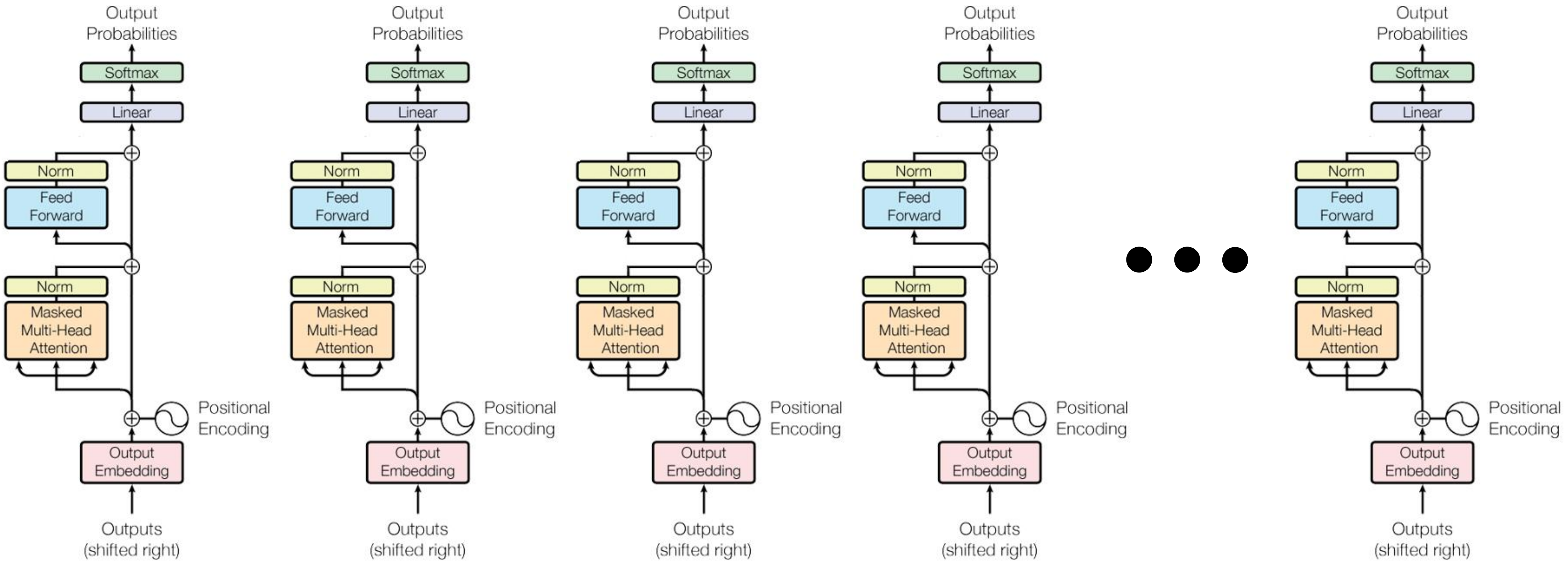
Decoder-only Transformer – 2



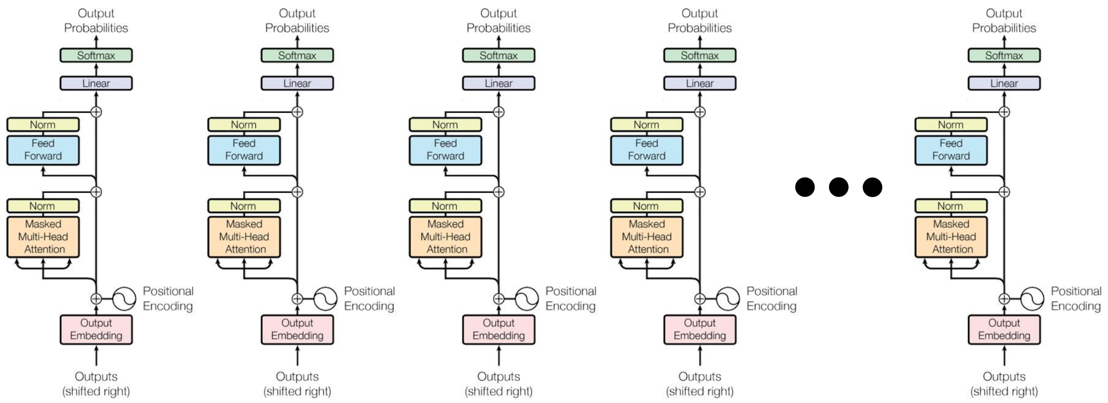
Decoder-only Transformer, Straight Through



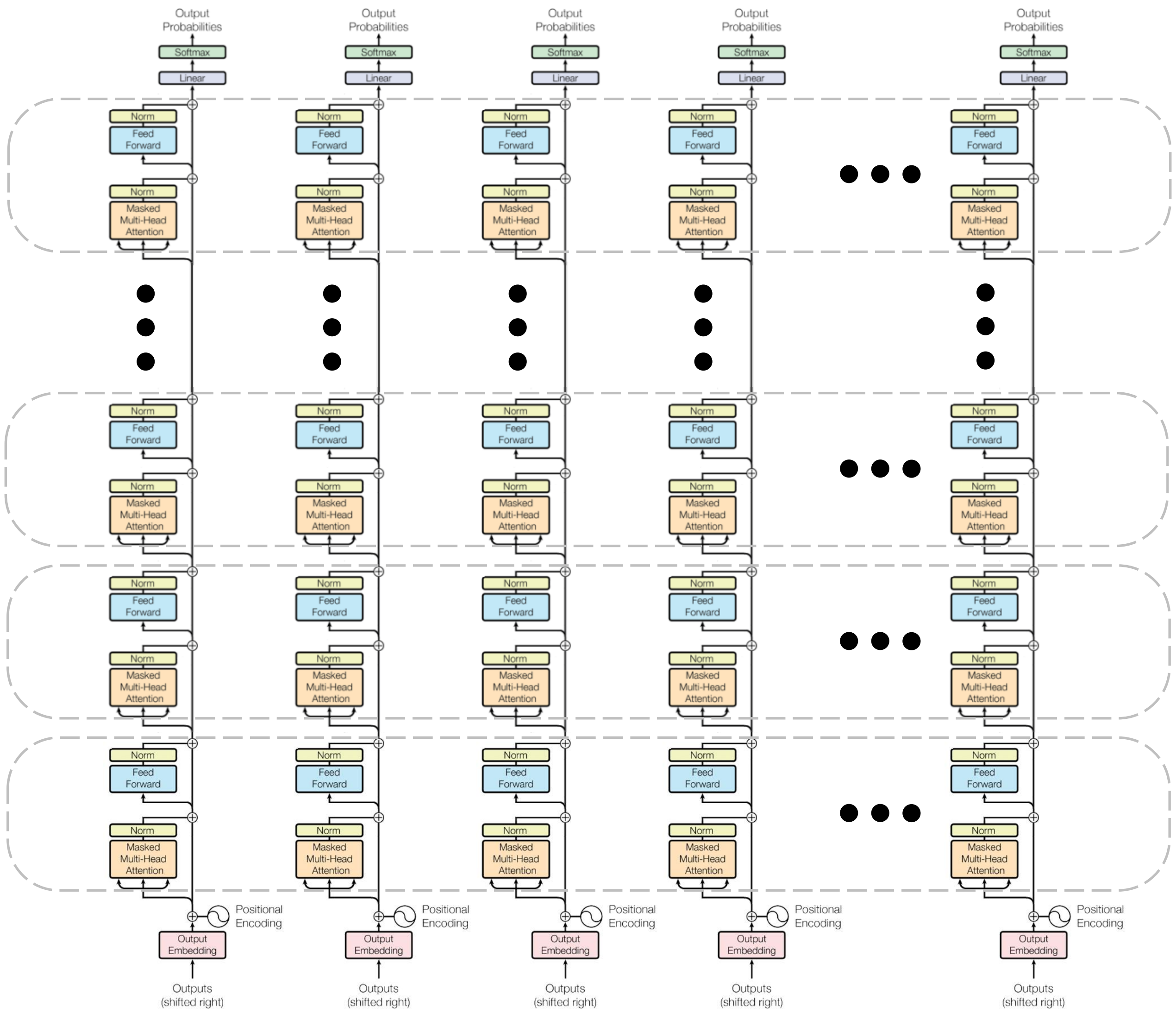
Decoder, Multiple Tokens



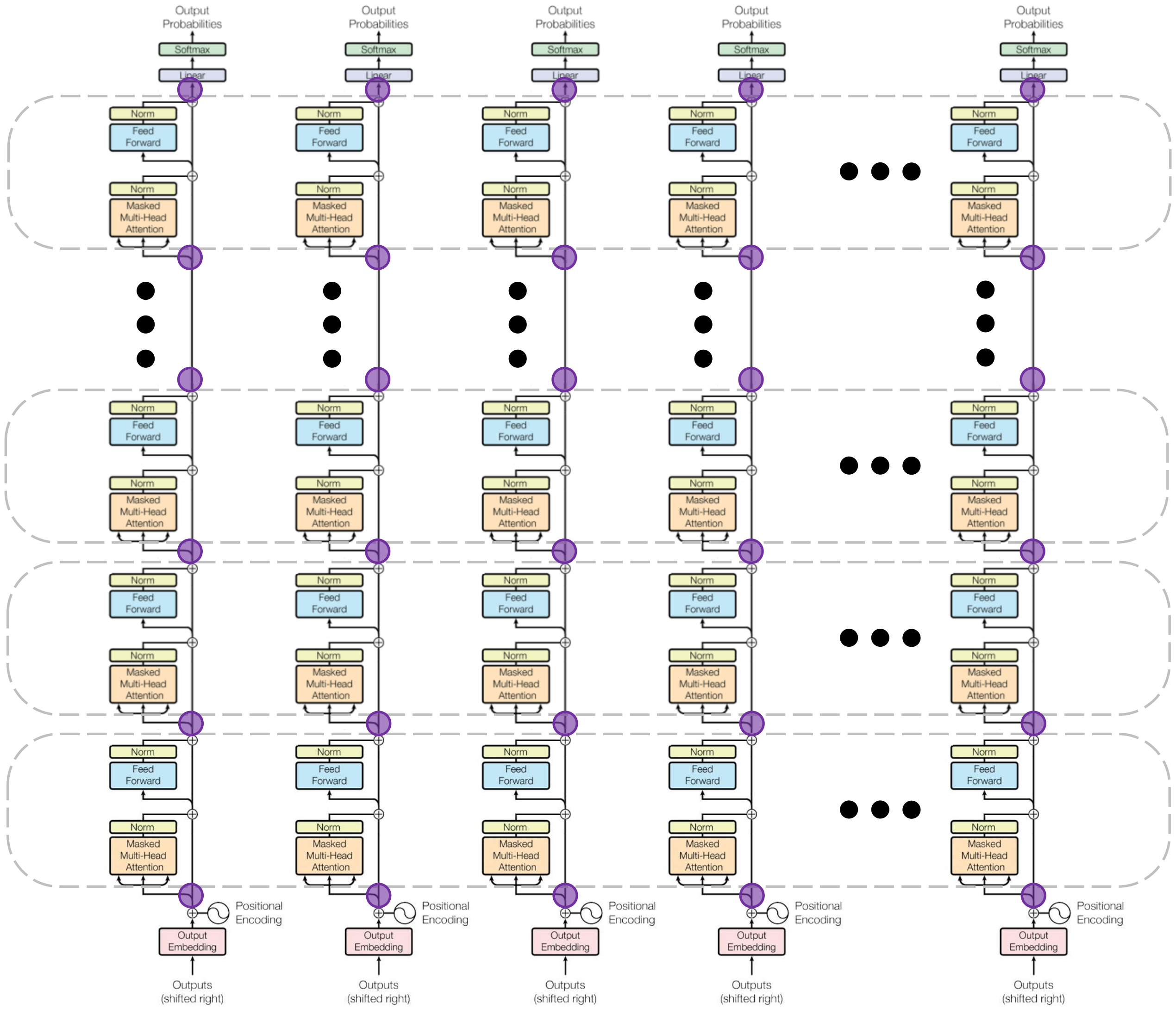
Decoder, Multiple Tokens, Small



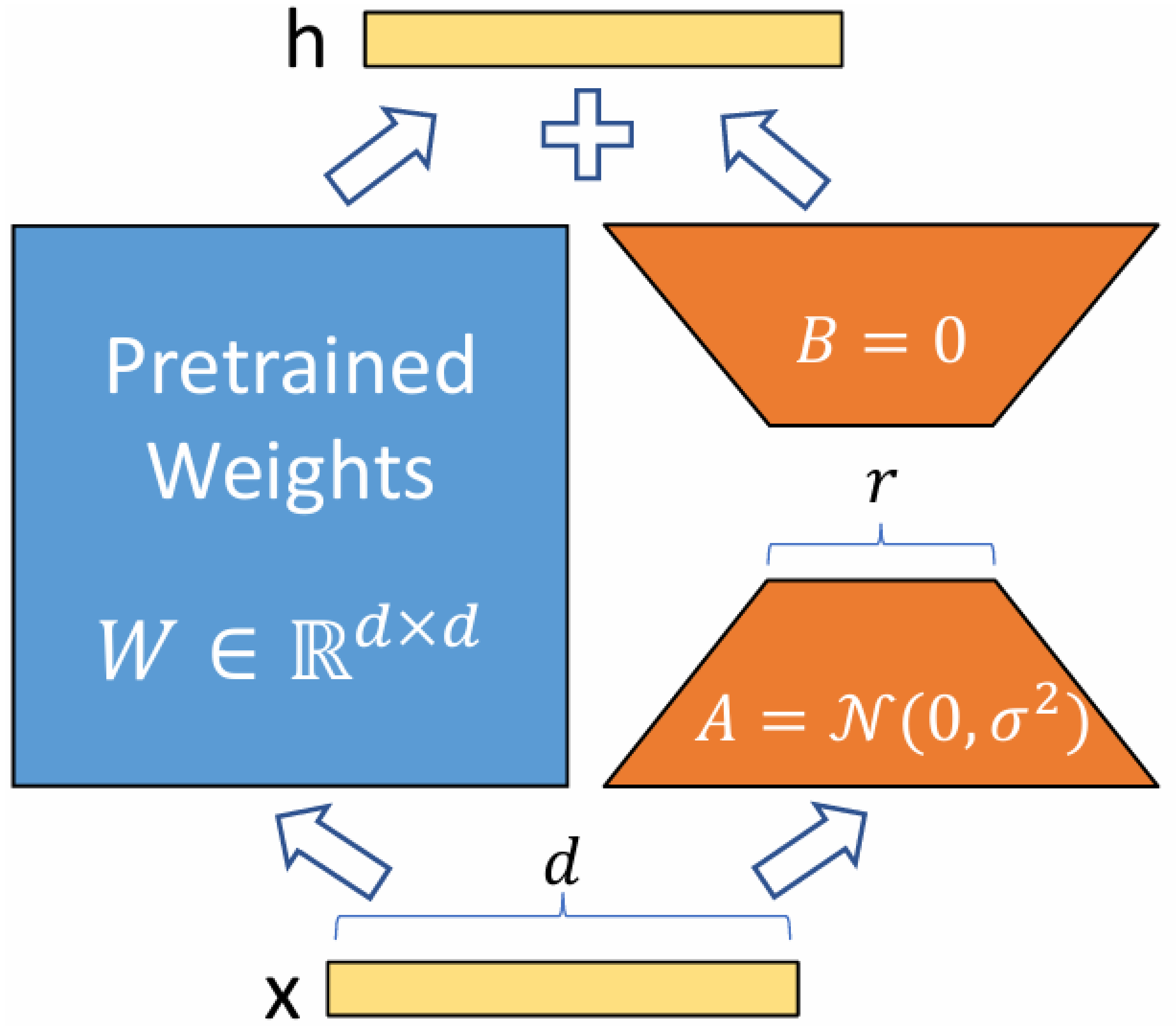
Decoder, Adding Multiple Layers



Decoder, Adding Hidden States



LoRA



h
Pretrained
Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

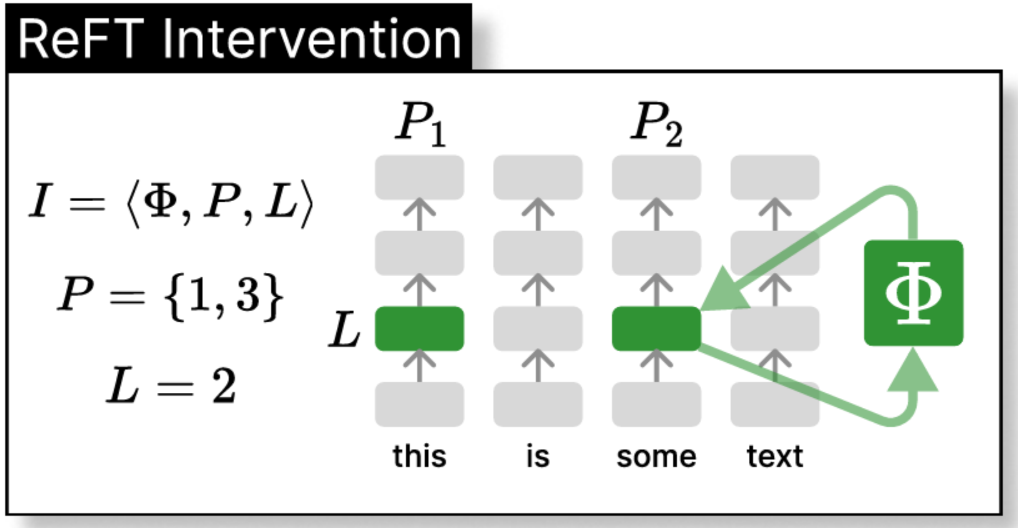
ReFT motivation (§3.1) [1]

- Interchange interventions
  - Use two different input strings, source *s* and base *b*
  - Run regular inference for source input *s*
    - Or sometimes a set of inputs
  - Save the activations from input(s) *s*
    - Or sometimes an average or difference
  - For input *b*, process inference normally for most of the time
  - Only at specific layer(s) and token position(s), overwrite the activations with the saved activations from *s*
    - Or sometimes add to the activations a value based on saved average or difference
  - If behavior changes from that of string *b* to that of string *s*, then you’ve proved the behavior is localized to at most the modified layers and tokens

ReFT motivation (§3.1) [2]

- Distributed Interchange Intervention (DII)
  - Use a low rank projection matrix **R** of shape *r* x *d*, where *r* << *d* is a lower dimensional subspace of the model dimension
  - Note that if **R** has orthonormal rows, then $\mathbf{R}^\top$ reverses the transformation

$$\mathrm{DII}(\mathbf{b}, \mathbf{s}, \mathbf{R}) = \mathbf{b} + \mathbf{R}^\top(\mathbf{R}\mathbf{s} - \mathbf{R}\mathbf{b})$$



Two low-rank ReFT instantiations (§3.2)

- With DII, we have a source we are using to modify inference
- With ReFT, we have outputs we want to fine-tune our model to learn
- LoReFT
  - We learn our projection matrix **R** as well as low-rank weights ***W*** and biases *b*
  - If we replace **R***s* from the DII equation with learned ***W****h* + *b*, then we get:

$$\Phi_{\text{LoReFT}}(\mathbf{h}) = \mathbf{h} + \mathbf{R}^{\top}\left(\mathbf{W}\mathbf{h} + \mathbf{b} - \mathbf{R}\mathbf{h}\right)$$

- DiReFT
  - Less expensive by removing the orthogonal projection constraint

$$\Phi_{\text{DiReFT}}(\mathbf{h}) = \mathbf{h} + \mathbf{W}_2^{\top}\left(\mathbf{W}_1\mathbf{h} + \mathbf{b}\right)$$

The ReFT family of methods (§3.3)

- A single intervention involves:
  - Pick the intervention function Φ that you want to learn
  - Pick the set of token position(s) P where you want to apply the intervention
  - Pick the set of layer(s) l where you want to apply the intervention
  - The intervention is the tuple $< \Phi, P, l >$
- Your ReFT method is a set of interventions
  - Additional constraint that if multiple interventions, they can't operate on the same token position and layer; they must be disjoint
- Note that ReFT interventions are applied to the activations on the residual stream
  - They always add components to the architecture and inference cost
  - Even if DiReFT looks like the same configuration as LoRA, it is in a different location

Experiments and Results (§4)

- Compared these two ReFTs to existing PEFTs
- Both RoBERTa and LLaMA, from 125M to 13B
- 4 NLP domains with over 20 benchmark datasets
- For their LoReFT and DiReFT, need to decide:
  - How many p of the first tokens to intervene on
  - How many of the last few tokens to intervene on
  - Which layers to intervene on
  - Whether or not all token positions in the same layer will share the same parameter (tied/untied)

Commonsense reasoning (§4.2)

Model	PEFT	Params (%)	Accuracy (↑)								
			BoolQ	PIQA	SIQA	HellaS.	WinoG.	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	PrefT*	0.039%	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Adapter^{S}*	1.953%	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Adapter^{P}*	3.542%	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.3
	LoRA*	0.826%	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	DoRA (half)*	0.427%	**70.0**	82.6	79.7	83.2	80.6	80.6	65.4	77.6	77.5
	DoRA*	0.838%	68.5	82.9	79.6	84.8	80.8	81.4	65.8	**81.0**	78.1
	DiReFT (ours)	0.031%	69.5	83.0	79.0	92.5	80.5	82.2	68.0	77.5	79.0
	LoReFT (ours)	0.031%	69.3	**84.4**	**80.3**	**93.1**	**84.2**	**83.2**	**68.2**	78.9	**80.2**
LLaMA-13B	PrefT*	0.031%	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Adapter^{S}*	1.586%	71.8	83.0	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Adapter^{P}*	2.894%	**72.5**	84.9	79.8	92.1	84.7	84.2	71.2	82.4	81.5
	LoRA*	0.670%	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA (half)*	0.347%	**72.5**	85.3	79.9	90.1	82.9	82.7	69.7	83.6	80.8
	DoRA*	0.681%	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	81.5
	DiReFT (ours)	0.025%	71.3	86.1	80.8	94.6	83.6	85.5	72.9	82.7	82.2
	LoReFT (ours)	0.025%	72.1	**86.3**	**81.8**	**95.1**	**87.2**	**86.2**	**73.7**	**84.2**	**83.3**
Llama-2 7B	LoRA*	0.826%	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA (half)*	0.427%	**72.0**	83.1	79.9	89.1	83.0	84.5	71.0	81.2	80.5
	DoRA*	0.838%	71.8	83.7	76.0	89.1	82.6	83.7	68.2	**82.4**	79.7
	DiReFT (ours)	0.031%	70.8	83.6	80.2	93.6	82.1	84.8	70.4	81.5	80.9
	LoReFT (ours)	0.031%	71.1	**83.8**	**80.8**	**94.3**	**84.5**	**85.6**	**72.2**	82.3	**81.8**
Llama-3 8B	LoRA*	0.700%	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA (half)*	0.361%	74.5	88.8	80.3	95.5	84.7	90.1	79.1	87.2	85.0
	DoRA*	0.710%	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	DiReFT (ours)	0.026%	73.4	88.7	81.0	95.6	85.5	91.8	**81.8**	85.4	85.4
	LoReFT (ours)	0.026%	**75.1**	**90.2**	**82.0**	**96.3**	**87.4**	**92.4**	81.6	**87.5**	**86.6**

Arithmetic reasoning (§4.3)

- ReFTs did worse than PEFTs, possibly because of longer sequences or because of CoT reasoning

Model	PEFT	Params (%)	Accuracy (↑)				
			AQuA	GSM8K	MAWPS	SVAMP	Avg.
LLaMA-7B	PrefT*	0.039%	14.2	24.4	63.4	38.1	35.0
	Adapter^{S}*	1.953%	15.0	33.3	77.7	**52.3**	44.6
	Adapter^{P}*	3.542%	18.1	35.3	**82.4**	49.6	46.4
	LoRA*	0.826%	18.9	**37.5**	79.0	52.1	**46.9**
	DiReFT (ours)	0.031%	21.3	24.1	74.5	42.7	40.6
	LoReFT (ours)	0.031%	**21.4**	26.0	76.2	46.8	42.6
LLaMA-13B	PrefT*	0.031%	15.7	31.1	66.8	41.4	38.8
	Adapter^{S}*	1.586%	22.0	44.0	78.6	50.8	48.9
	Adapter^{P}*	2.894%	20.5	43.3	81.1	**55.7**	50.2
	LoRA*	0.670%	18.5	**47.5**	**83.6**	54.6	**51.1**
	DiReFT (ours)	0.025%	20.5	35.8	80.8	54.8	48.0
	LoReFT (ours)	0.025%	**23.6**	38.1	82.4	54.2	49.6

Instruction-following (§4.4)

- ReFTs excelled, so the also tried with fewer parameters and with less data, where LoReFT still did better than RED

Model & PEFT	Params (%)	Win-rate (↑)
GPT-3.5 Turbo 1106†	—	86.30
Llama-2 Chat 13B†	—	81.10
Llama-2 Chat 7B†	—	71.40
Llama-2 7B & FT*	100%	80.93
Llama-2 7B & LoRA*	0.1245%	81.48
Llama-2 7B & RED*	0.0039%	81.69
Llama-2 7B & **DiReFT (ours)**	0.0039%	84.85
Llama-2 7B & **LoReFT (ours)**	0.0039%	**85.60**
Llama-2 7B & **LoReFT (ours, *half*)**	0.0019%	84.12
Llama-2 7B & **LoReFT (ours, *1K*)**‡	0.0039%	81.91

Natural language understanding (§4.5)

- LoReFT was competitive, but DiReFT wasn't, possibly because of the smaller model size

Model	PEFT	Params (%)	Accuracy (↑)								
			MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
base	FT	100%	87.3	94.4	87.9	62.4	92.5	91.7	78.3	90.6	85.6
	Adapter*	0.318%	87.0	93.3	88.4	60.9	92.5	**90.5**	76.5	**90.5**	**85.0**
	LoRA*	0.239%	86.6	93.9	88.7	59.7	**92.6**	90.4	75.3	90.3	84.7
	$\text{Adapter}^{\text{FNN}}$*	0.239%	**87.1**	93.0	88.8	58.5	92.0	90.2	77.7	90.4	84.7
	BitFit*	0.080%	84.7	**94.0**	88.0	54.0	91.0	87.3	69.8	89.5	82.3
	RED*	0.016%	83.9	93.9	**89.2**	**61.0**	90.7	87.2	78.0	90.4	84.3
	DiReFT (ours)	0.015%	82.5	92.6	88.3	58.6	91.3	86.4	76.4	89.3	83.2
	LoReFT (ours)	0.015%	83.1	93.4	**89.2**	60.4	91.2	87.4	**79.0**	90.0	84.2
large	FT	100%	88.8	96.0	91.7	68.2	93.8	91.5	85.8	92.6	88.6
	Adapter*	0.254%	90.1	95.2	90.5	65.4	94.6	**91.4**	85.3	91.5	88.0
	LoRA*	0.225%	90.2	96.0	89.8	65.5	**94.7**	90.7	86.3	**91.7**	88.1
	$\text{Adapter}^{\text{FNN}}$*	0.225%	**90.3**	96.1	**90.5**	64.4	94.3	91.3	84.8	90.2	87.7
	RED*	0.014%	89.5	96.0	90.3	**68.1**	93.5	88.8	86.2	91.3	88.0
	DiReFT (ours)	0.014%	88.7	95.4	88.5	66.7	93.9	88.1	86.9	91.2	87.4
	LoReFT (ours)	0.014%	89.2	**96.2**	90.1	68.0	94.1	88.5	**87.5**	91.6	**88.2**

ReFT conclusion

- ReFT performs fine-tuning by modifying the activations on the residual stream, rather than modifying the model weights
- Introduce LoReFT and DiReFT, which work with low rank projections of the activation space to reduce the number of trainable parameters
- Paper reports mostly better benchmark performance than PEFT methods, while requiring significantly fewer parameters
  - Didn’t perform well for arithmetic reasoning
- One big convenience of LoRA is that after fine-tuning, updated weights can be used with the original model architecture
  - The ReFT paper doesn’t highlight the fact that these techniques require a modified architecture, which makes deployment a little more complicated

References

- LoRA: Low-Rank Adaptation of Large Language Models
  Edward J. Hu, et al. (2021)
  https://arxiv.org/abs/2106.09685