Carnegie Mellon University

Customizing LLMs via full model finetuning

Large Language Models: Methods and Applications

Daphne Ippolito

How we conceived of LMs+NLP several years ago.

One model per task trained exclusively on task-specific data.

(left) Original transformer paper trained on MT

(right) Earliest LM-based chatbot paper trained on movie substitles

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

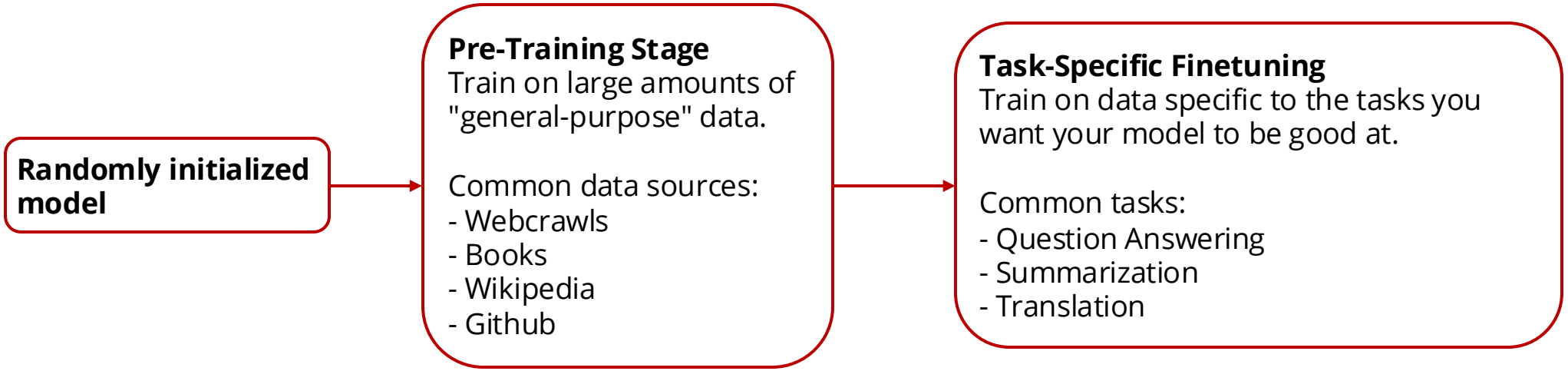
4.2. OpenSubtitles dataset

We also tested our model on the OpenSubtitles dataset (Tiedemann, 2009). This dataset consists of movie conversations in XML format. It contains sentences uttered by characters in movies. We applied a simple processing step removing XML tags and obvious non-conversational text (e.g., hyperlinks) from the dataset. As turn taking is not clearly indicated, we treated consecutive sentences assuming they were uttered by different characters. We trained our model to predict the next sentence given the previous one, and we did this for every sentence (noting that this doubles our dataset size, as each sentence is used both for context and as target). Our training and validation split has 62M sentences (923M tokens) as training examples, and the validation set has 26M sentences (395M tokens). The split is done in such a way that each sentence in a pair of sentences either appear together in the training set or test set but not both. Unlike the previous dataset, the OpenSubtitles is quite large, and rather noisy because consecutive sentences may be uttered by the same character. Given the broad scope of movies, this is an open-domain conversation dataset, contrasting with the technical troubleshooting dataset.

"Attention is all you need." 2018. <https://arxiv.org/pdf/1706.03762> "A Neural Conversational Model" (2015) https://arxiv.org/abs/1506.05869

Paradigm shift around ~2018

One language model pre-trained for generic language understanding then finetuned for each task.

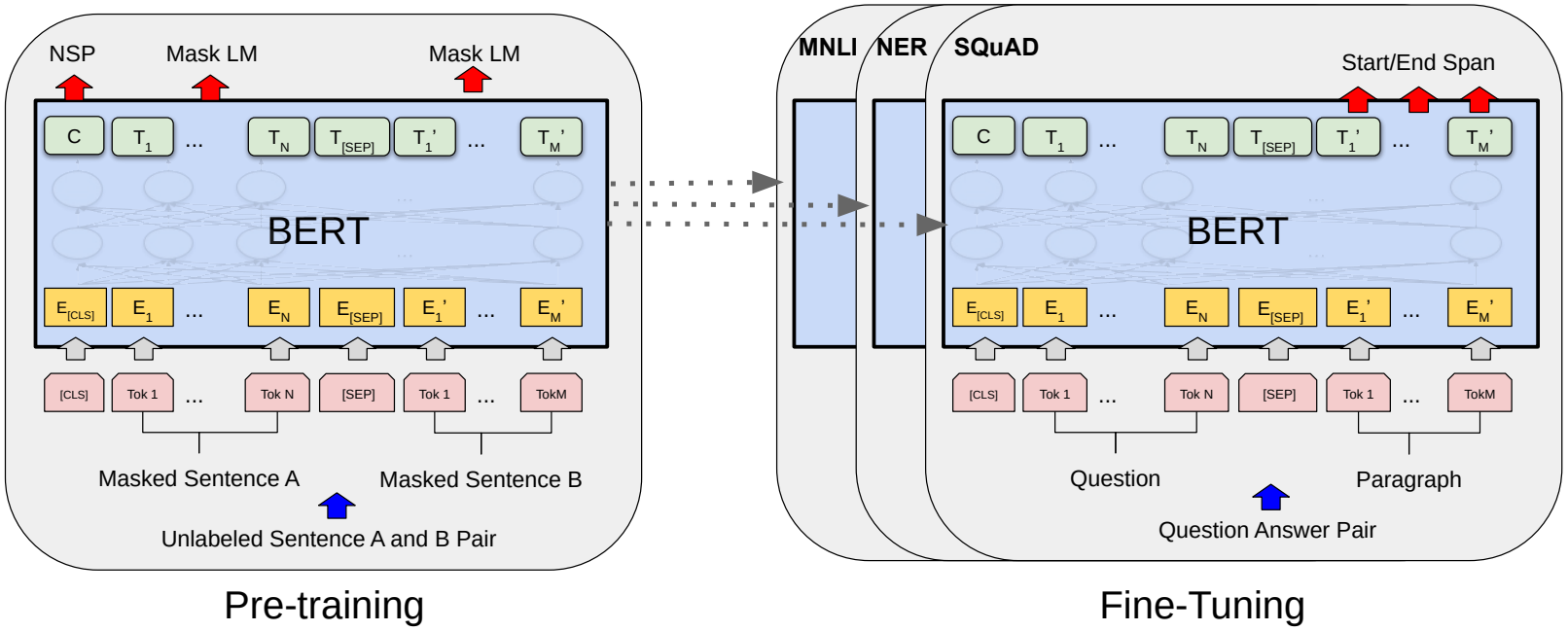


Paradigm shift around ~2018

One language model pre-trained for generic language understanding then finetuned for each task.

BERT:

- Pre-train an encoder-only model with mask infilling and sentence ordering objectives.
- Finetune once per NLP task



Paradigm shift around ~2018

One language model pre-trained for generic language understanding then finetuned for each task.

GPT-1:

- Pre-train a decoder-only LM with a language modelling objective.
- Finetune once per NLP task

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

"Improving Language Understanding by Generative Pre-Training." 2018. https://hayate-lab.com/wp-content/uploads/2023/05/43372bfa750340059ad87ac8e538c53b.pdf

Disadvantages of full model-finetuning

- One model per task is fine for small models, but not for today's big ones.
- Training separately on each task is more expensive than training on all tasks at once.
- Overfitting / catastrophic forgetting on small datasets
- Storage of all those model checkpoints (one per task) is expensive.
- During inference time, it's expensive to have all those models loaded onto GPUs at the same time.

Solutions to Improve Efficiency

- Avoid finetuning entirely
  - In-context learning
- Parameter-efficient finetuning
- Multi-task finetuning → instruction finetuning

Carnegie Mellon University

In-Context Learning

Large Language Models: Methods and Applications

Daphne Ippolito

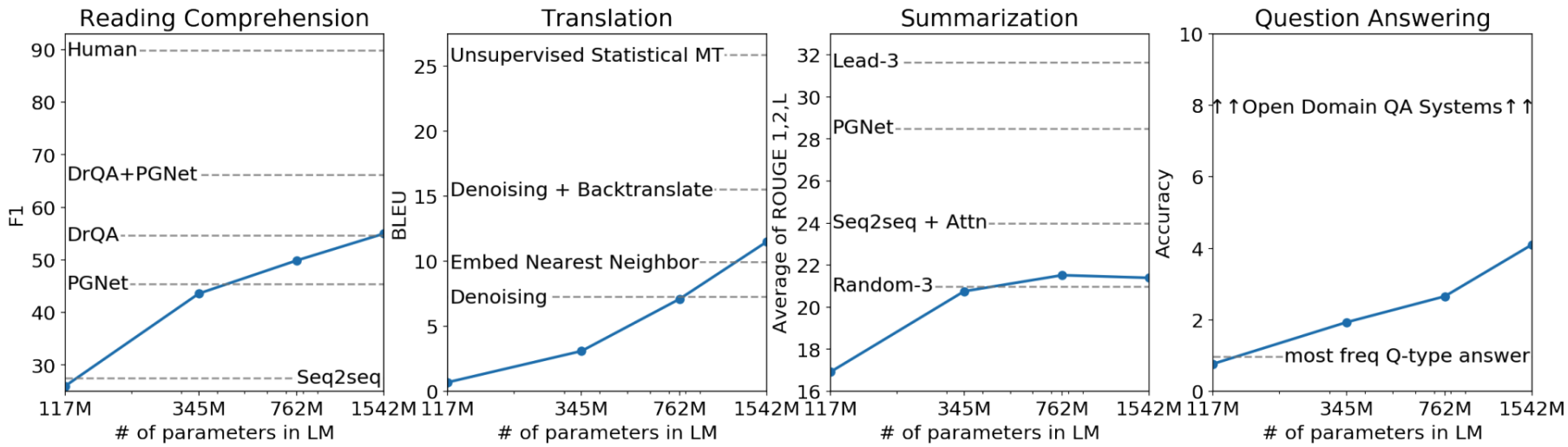
Language Models are Unsupervised Multitask Learners (aka GPT-2 paper)

The authors “demonstrate that language models begin to learn [question answering, machine translation, reading comprehension, and summarization] tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText.”

Language Models are Unsupervised Multitask Learners (aka GPT-2 paper)

The authors “demonstrate that language models begin to learn [question answering, machine translation, reading comprehension, and summarization] tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText.”

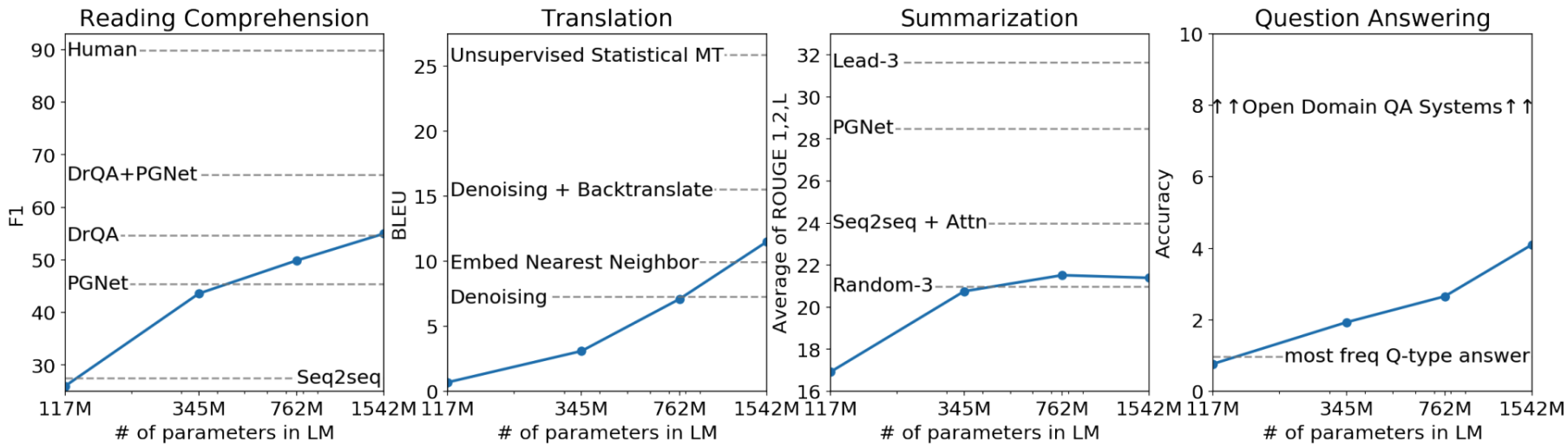
Across a variety of tasks, they saw this behaviour “emerge” with larger model sizes.



Language Models are Unsupervised Multitask Learners (aka GPT-2 paper)

The authors “demonstrate that language models begin to learn [question answering, machine translation, reading comprehension, and summarization] tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText.”

Across a variety of tasks, they saw this behaviour “emerge” with larger model sizes.



In-Context Learning

All the information necessary to get the LM to do the task is included as part of the textual prompt intputed to the LM.

In-Context Learning

All the information necessary to get the LM to do the task is included as part of the textual prompt intputed to the LM.

LLM zero-shot learning: a prompt that contains instructions for the task, but no actual examples of the task being performed.

LLM few-shot learning: a prompt that contains both instructions as well as several examples of the task being performed.

Zero-shot learning for sentiment classification

Prompt:
Review: Let there be no question: Alexions owns the best cheeseburger in the region and they have now for decades. Try a burger on Italian bread. The service is flawlessly friendly, the food is amazing, and the wings? Oh the wings... but it's still about the cheeseburger. The atmosphere is inviting, but you can't eat atmosphere... so go right now. Grab the car keys... you know you're hungry for an amazing cheeseburger, maybe some wings, and a cold beer! Easily, hands down, the best bar and grill in Pittsburgh.

On a 1 to 4 star scale, the reviewer would probably give this restaurant a

Model's Output Probabilities:

P("1" | prompt) = 0.0016
P("2" | prompt) = 0.0032
P("3" | prompt) = 0.012
P("4" | prompt) = 0.78

Template used:
Review: <review>

On a 1 to 4 star scale, the reviewer would probably give this restaurant a <class_label>

This template is tried for all possible class labels.

Output: the number with the highest likelihood.

Zero-shot learning for abstractive summarization

Prompt:

India's moon rover completes its walk. Scientists analyzing data looking for signs of frozen water

BEW DELHI -- India’s moon rover has completed its walk on the lunar surface and been put into sleep mode less than two weeks after its historic landing near the lunar south pole, India’s space mission said.

“The rover completes its assignments. It is now safely parked and set into sleep mode," with daylight on that part of the moon coming to an end, the Indian Space Research Organization said in a statement late Saturday.

The rover's payloads are turned off and the data it collected has been transmitted to the Earth via the lander, the statement said.

[...]

Active since the 1960s, India has launched satellites for itself and other countries, and successfully put one in orbit around Mars in 2014. India is planning its first mission to the International Space Station next year, in collaboration with the United States.

TL;DR;

Model Generation:

India's moon rover has completed its assignments and gone to sleep mode after just two weeks of being on the lunar surface. The rover has started transmitting the data it has collected to Earth, but scientists are unsure if it can survive the extremely cold temperatures on the moon and wake up in order to conduct another set of missions when the sun rises again. India successfully landed the rover and underscored its status as a major tech power and space program.

Template used:

<title>
<article>
TL;DR;

We return the model’s generation from the filled out template.

Few-shot learning for machine translation

Prompt:

The dog chased a squirrel at the park. = 那只狗在公园里追一只松鼠。

I was late for class. = 我上课迟到了。

The hippopotamus ate my homework. =

Model Generation:

河马吃了我的家庭作业。

Template Used:

<example1_en> = <example1_zh>

<example2_en> = <example2_zh>

<query_en> =

There are often many ways to verbalize a task.

Prompt:

The dog chased a squirrel at the park. = 那只狗在公园里追一只松鼠。

I was late for class. = 我上课迟到了。

The hippopotamus ate my homework. =

Prompt withan Alternative Template:

Translate from English to Chinese.

The dog chased a squirrel at the park. = 那只狗在公园里追一只松鼠。

I was late for class. = 我上课迟到了。

The hippopotamus ate my homework. =

Prompt with an Alternative Template:

Translate from English to Chinese.

English: The dog chased a squirrel at the park.
Chinese: 那只狗在公园里追一只松鼠。

English: I was late for class.
Chinese: 我上课迟到了。

English: The hippopotamus ate my homework.
Chinese:

The different templates are called **verbalizers**.

Summary of Terms

- **Emergence:** when quantitative changes in a system result in qualitative changes in behavior.
- **Emergent behaviors:** abilities that larger models have and smaller models don't
- **In-context learning**: when a language model "learns" how to do a task from a textual prompt containing a natural language instruction for the task, several exemplars of the task, or both.
- **Zero-shot learning:** In context learning that does not include any exemplars of the task.
- **Few-shot learning:** In context learning that contains several exemplars of the task.
- **Prompt engineering**: The painstaking process of trying out many different prompts until you find one that works well for your task.
- **Verbalizer:** The template we wrap an example in in order to perform the task.

More on prompt engineering (why it works, cases where it doesn’t) in next lecture.

Carnegie Mellon University

Parameter-Efficient Finetuning

Large Language Models: Methods and Applications

Daphne Ippolito

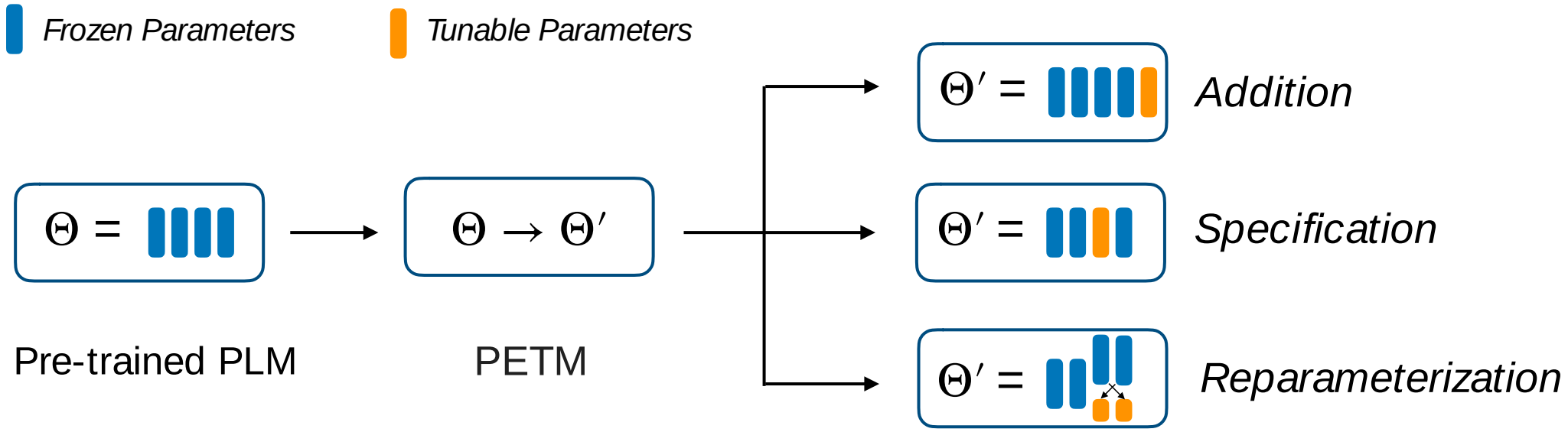
What is parameter efficient tuning?

Rather than finetuning the entire model, we finetune only small amounts of weights.

In this lecture, we'll break PETM techniques into roughly three categories.

Three Categories of PETM

1. **Addition:** What if we introduce additional trainable parameters to the neural network and just train those?
2. **Specification:** What if we pick a small subset of the parameters of the neural network and just tune those?
3. **Reparameterization:** What if we re-parameterize the model into something that is more efficient to train?



What if we introduce additional trainable parameters to the neural network and just train those?

Methods in this category:
prompt tuning, prefix tuning, adapters, and compacters

Intuition for Prompt Tuning

Prompt engineering requires a lot of human decision-making and can be very finicky to get working.

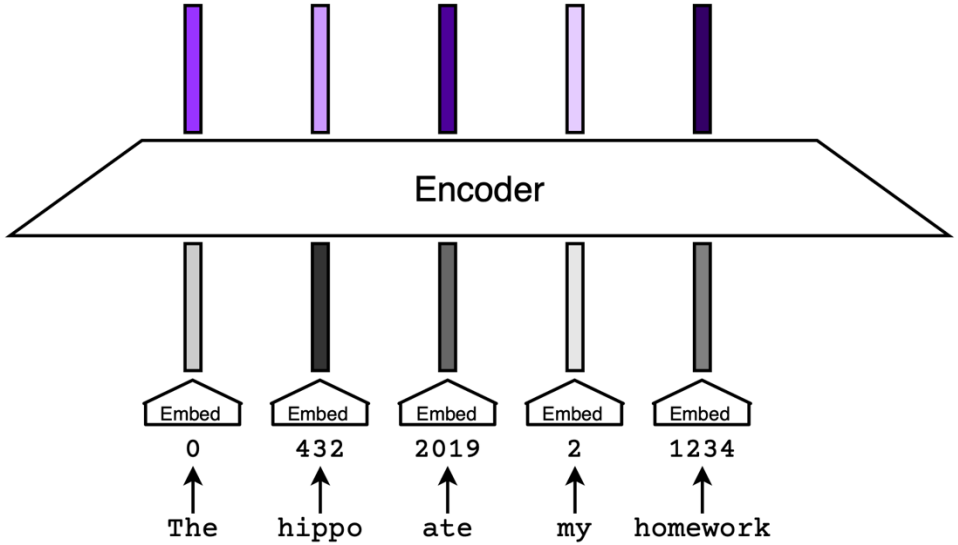
If we have a bunch of examples of the task, why can't we just train a neural network to produce a good prompt for the task?

Prompt Tuning Method

What we want to do: optimize a sequence of tokens that we can prepend to our task query, causing the LLM to do the task in question.

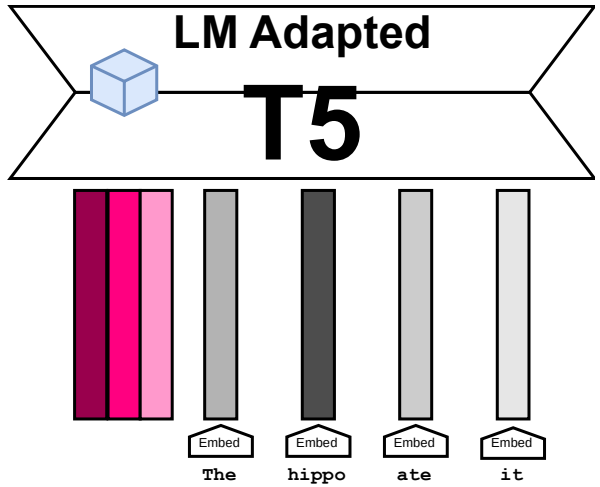
In practice, optimizing over discrete tokens is hard.

What we do instead: Optimize a sequence of *embeddings* we can prepend to our query to the LLM, causing the LLM to do the task.



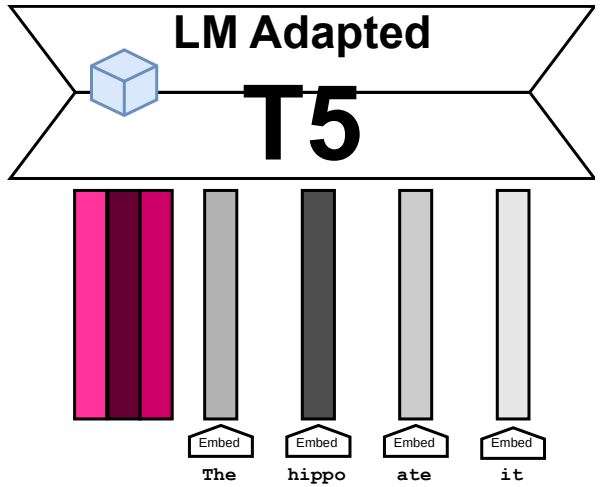
Prompt Tuning Method

1. Finetune T5 to act a bit more like a traditional language model.
   - This only needs to be done once, and empirically makes prompt tuning working better.
   - This is probably because the span-corruption objective T5 was originally trained with isn't amenable to prompting.
2. Freeze the weights of T5.
3. Create a new learnable embedding matrix $\mathbf{P} \in \mathbb{R}^{k \times d}$
   - Set the first *k* input embeddings to be learnable.
   - *k* is a hyperparameter up to the choice of the implementer.
4. Initialize the *k* learnable embeddings. Some options include:
   - Random initialization
   - Initialize to values drawn from the vocabulary embedding matrix
5. Train on your task specific data.



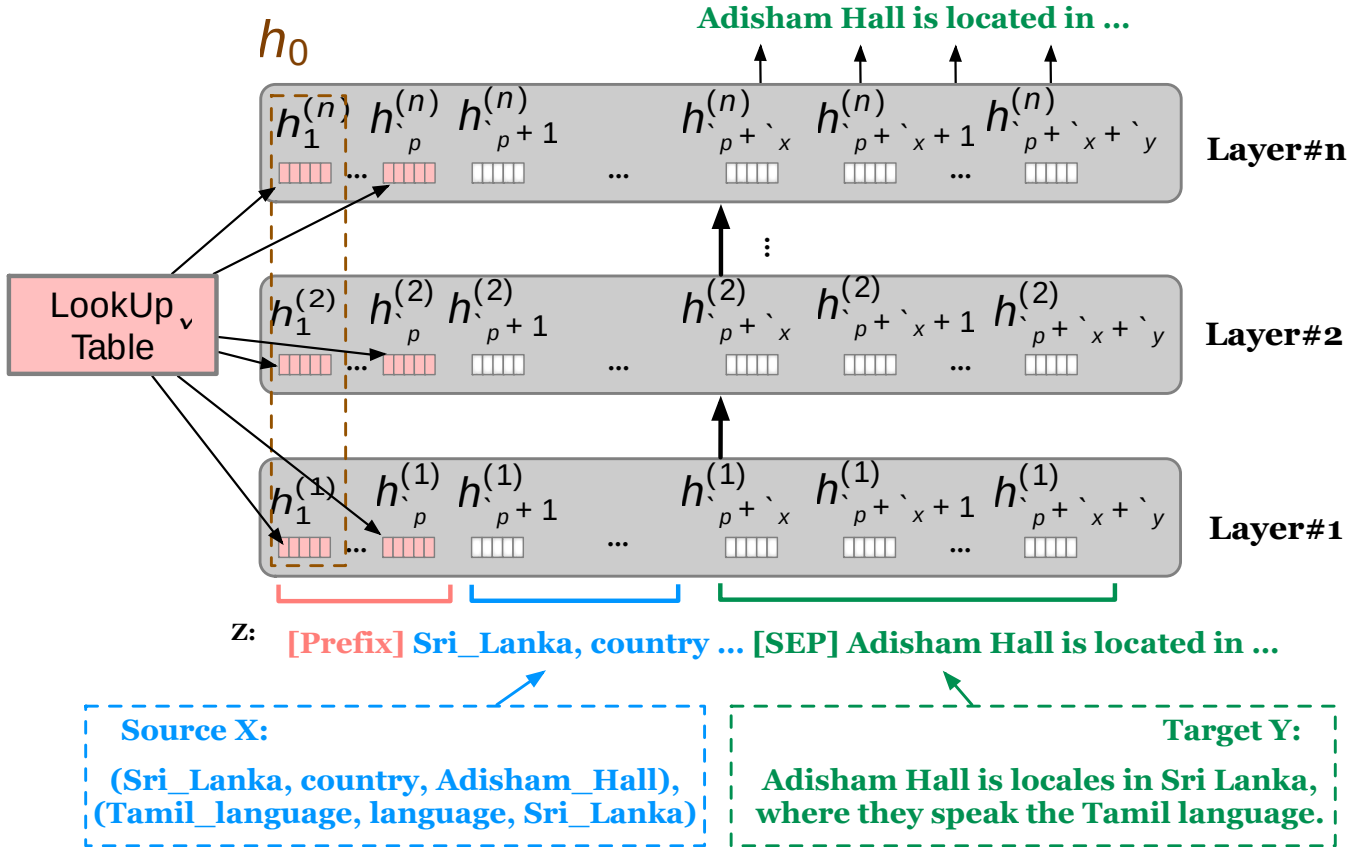
Prompt Tuning Method

1. Finetune T5 to act a bit more like a traditional language model.
   - This only needs to be done once, and empirically makes prompt tuning working better.
   - This is probably because the span-corruption objective T5 was originally trained with isn't amenable to prompting.
2. Freeze the weights of T5. Set the first *k* input embeddings to be learnable.
   - *k* is a hyperparameter up to the choice of the implementer.
3. Initialize the *k* learnable embeddings. Some options include:
   - Random initialization
   - Initialize to values drawn from the vocabulary embedding matrix
4. Train on your task specific data.



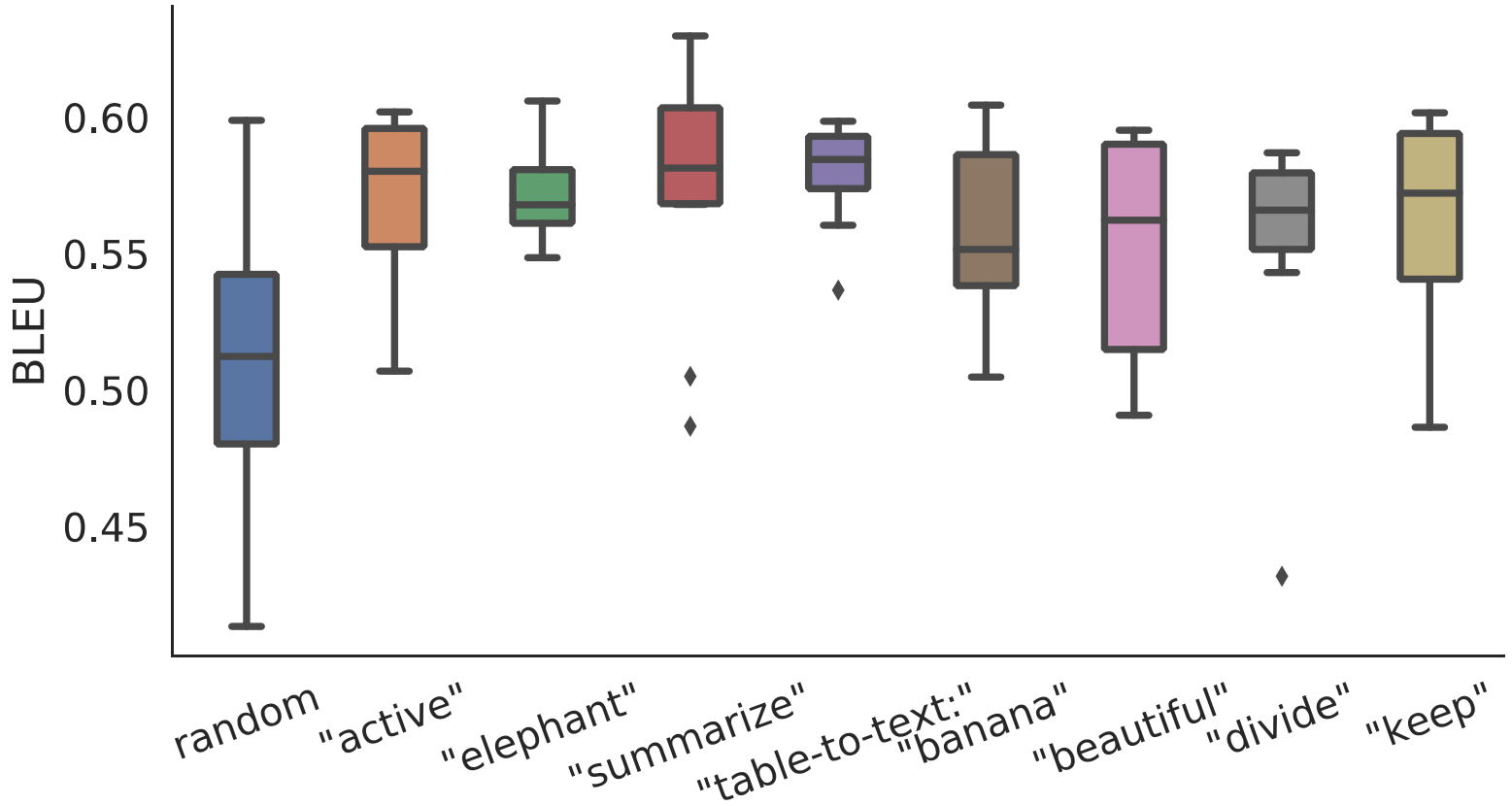
Prefix Tuning

Same idea as prompt tuning, except that the learned prefix is appended not just to the input embeddings, but rather at each layer of the Transformer.



How to initialize the prefix?

Initializing to real embeddings seems to work better than random initialization.



Advantages of Prefix/Prompt Tuning

- The learned embeddings tend to be relatively small, just a few megabytes or less.
    - It is cheap to keep around one set of embeddings per task.
- The pre-trained LLM can be loaded into memory (such as on a server), and at inference time, the appropriate task-specific embeddings can be swapped in.
    - Example use case: User customization

Pitfalls of Prefix and Prompt Tuning

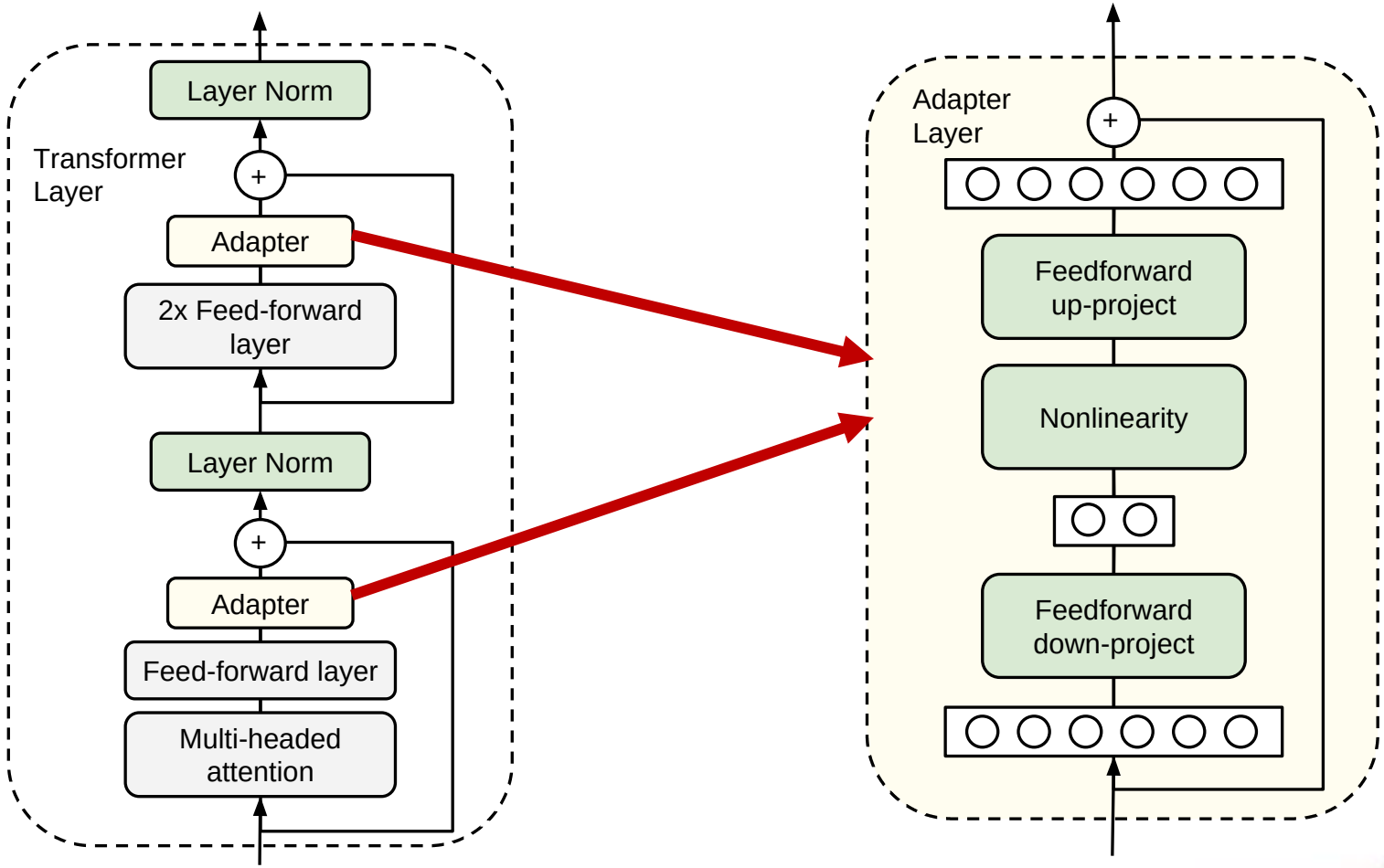
- In practice, these methods tend to converge significantly slower than full parameter fine-tuning.
- Unclear what the best prefix length is for any particular task.
  - Every sequence position you “spend” on the prefix is one less you have for your actual task.
- Learned embeddings are not very interpretable.

Adapters

- Adapters are new NN modules that are added between layers of a pre-trained network.
- The original model weights are fixed; just the adapter modules are tuned.
- The adapters are initialized s.t. the output of the adapter-inserted module resembles the original model.

What are adapters?

- **Adapters** are new modules that are added between layers of a pre-trained network.
- The original model weights are fixed; just the adapter modules are tuned.
- The adapters are initialized s.t. the output of the adapter-inserted module resembles the original model.



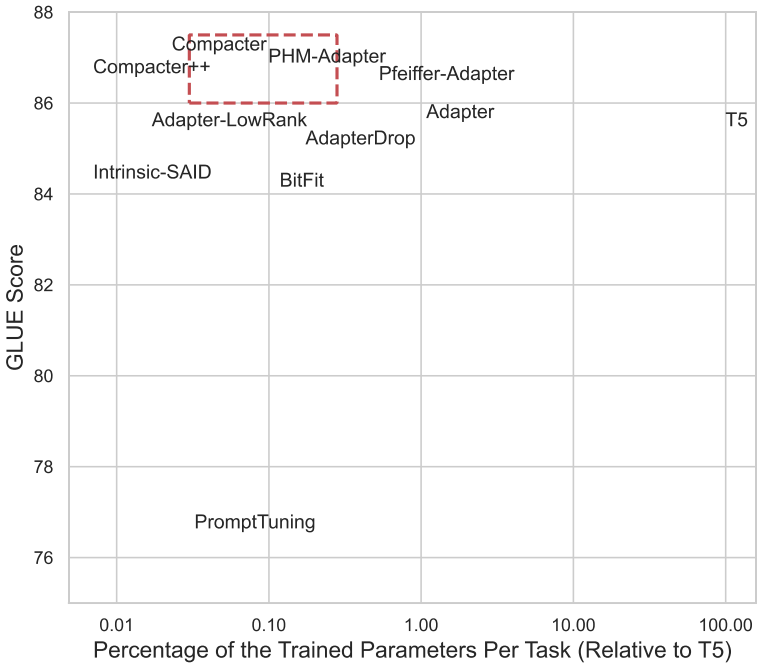
Compacters

- Compacters are an extension of adapters which aim to make the technique more efficient.
- Adapters are standard fully connected layers.
  - Linear project to lower dimension followed by nonlinearity followed by projection back up to original dimension.
  - $y = \boldsymbol{W_2}\,\mathrm{GELU}(\boldsymbol{W_1 x} + \boldsymbol{b_1}) + \boldsymbol{b_2}$
- The compacter replaces the fully connected layer with a more efficient architecture.
  - Combination of hypercomplex multiplication and parameter sharing
  - Each $\boldsymbol{W}$ above is learned as a sum of *n* Kronecker products
  - Users can specify *n*: how many divisions to make of the linear layer as a hyperparamter.
- Compacters reduce the number of parameters in the adapter layer to 1/*n* without harming the performance.

Other Variants

There have been many other extensions to adapters which we won't discuss in this class.

Name & Refs	Method	#Params
SEQUENTIAL ADAPTER Houlsby et al. (2019) COMPACTER Mahabadi et al. (2021a) ADAPTERDROP Rücklé et al. (2021)	$\text{LayerNorm}(\mathbf{X} + \mathbf{H}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{H}(\mathbf{X})))$ $\text{LayerNorm}(\mathbf{X} + \mathbf{F}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{F}(\mathbf{X})))$ $\text{ADT}(\mathbf{X}) = \mathbf{X} + \sigma(\mathbf{X}\mathbf{W}_{d_h \times d_m})\mathbf{W}_{d_m \times d_h}, \ \sigma = \text{activation}$	$L \times 2 \times (2d_h d_m)$ $L \times 2 \times (2(d_h + d_m))$ $(L - n) \times 2 \times (2d_h d_m)$
PARALLEL ADAPTER He et al. (2022)	$\text{LayerNorm}(\mathbf{X} + \mathbf{H}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{X}) + \mathbf{H}(\mathbf{X}))$ $\text{LayerNorm}(\mathbf{X} + \mathbf{F}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{X}) + \mathbf{F}(\mathbf{X}))$ $\text{ADT}(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}_{d_h \times d_m})\mathbf{W}_{d_m \times d_h}, \ \sigma = \text{activation}$	$L \times 2 \times (2d_h d_m)$
ADAPTERBIAS	$\text{LayerNorm}(\mathbf{X} + \mathbf{F}(\mathbf{X})) \rightarrow \text{LayerNorm}(\text{ADT}(\mathbf{X}) + \mathbf{F}(\mathbf{X}))$ $\text{ADT}(X) = \mathbf{X}\mathbf{W}_{d_h \times 1}\mathbf{W}_{1 \times d_h}$	$L \times 2 \times d_h$



Advantages of Adapter-Based Methods

- Faster to converge during training than prompt/prefix tuning.
- Have been shown to be quite effective in multi-task settings.
  - There are methods for training task-specific adapters and then combining the to leverage the cross-task knowledge (see AdapterFusion).
- Tend to be faster to tune than full model finetuning.
- Possibly more robust to adversarial perturbations of the tuning data than full model finetuning.
  - (see robust transfer learning paper)

Pitfalls of Adapter Methods

- Adding in new layers means making inference slower.
- It also makes the model bigger (possibly harder to fit on available GPUs).
    - Number of parameters to add scales with number of layers in the model.

What if we pick a small subset of the parameters of the neural network and just tune those?

Methods in this category:
layer freezing, BitFit, diff pruning

Layer Freezing

- Research has shown that earlier layers of the Transformer tend to capture linguistic phenomena and basic language understand; later layers are where the task-specific learning happens.

- This means we should be able to learn new tasks by freezing the earlier layers and just tuning the later ones.

BitFit: Bias-terms Fine-tuning

- Only tune the bias terms and final classification layer (if doing classification)
- Recall the equations for multi-head attention

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

  - ℓ is the layer index
  - m is the attention head index
  - Only the bias terms (shown in red) are updated.

DiffPruning

- In prior methods we discussed, the choice of what parameters to freeze and what parameters to tune was done manually.
- Why not learn this instead?
- Main idea:
  - For each parameter, finetune a learnable “delta” which gets added to the original parameter value.
  - Use an L0-norm penalty to encourage sparsity in the deltas.
- Advantages:
  - Model learns what parameters are important to update for new task.

What if we re-parameterize the model into something that is more efficient to train?

Methods in this category:
LoRa, QLoRA, $(IA)^3$

Intuition for Re-Parameterizing the Model

Finetuning has a low intrinsic dimension, that is, the minimum number of parameters needed to be modified to reach satisfactory performance is not very large.

This means that we can reparameterize a subset of the original model parameters with low-dimensional proxy parameters, and just optimize the proxy.

What do we mean by **intrinsic dimension**?

- An objective function's intrinsic dimension measures the minimum number of parameters needed to reach a satisfactory solution to the objective.
- Can also be thought of as the lowest dimensional subspace in which one can optimize the original objective function to within a certain level of approximation error.

What do we mean by **intrinsic dimension**?

- Suppose we have model parameters θ^D
  - D is the number of parameters.
- Instead of optimizing θ^D, we could instead optimize a smaller set of parameters θ^d where $d \ll D$.
- This is done through clever factorization:
  - $\theta^D = \theta_0^D + P(\theta^d)$ where $P: \mathbb{R}^d \rightarrow \mathbb{R}^D$
  - *P* is typically a linear projection: $\theta^D = \theta_0^D + \theta^d M$
- If you are interested in this, there are a lot more details in the paper.

LoRA: **L**ow **R**ank **A**daption

- Intuition: It's not just the model weights that are low rank; *updates* to the model weights are also low-rank.
- LoRA freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer.
- Like DiffPruning, we are learning a delta to apply to each weight. In the case of LoRA, this delta has been re-paramaterized to be lower dimension than the original model parameters.
- In practice, LoRA only adapts the attention weights and keeps the rest of the Transformer as-is.

"LoRA: Low-Rank Adaptation of Large Language Models." Hu et al. 2021.

$(IA)^3$: **I**nfused **A**dapter by **I**nhibiting and **A**mplifying **I**nner **A**ctivations

- Intended to be an improvement over LoRA
- Three goals:
  - must add or update as few parameters as possible to avoid incurring storage and memory costs
  - should achieve strong accuracy after training on only a few examples of a new task
  - must allow for mixed-task batches
- Main idea:
  - Rescale inner activations with lower-dimensional learned vectors, which are injected into the attention and feedforward modules
- Main differences from LoRA:
  - LoRA learns low-rank updates to the attention weights
  - $(IA)^3$ learns injectable vectors.

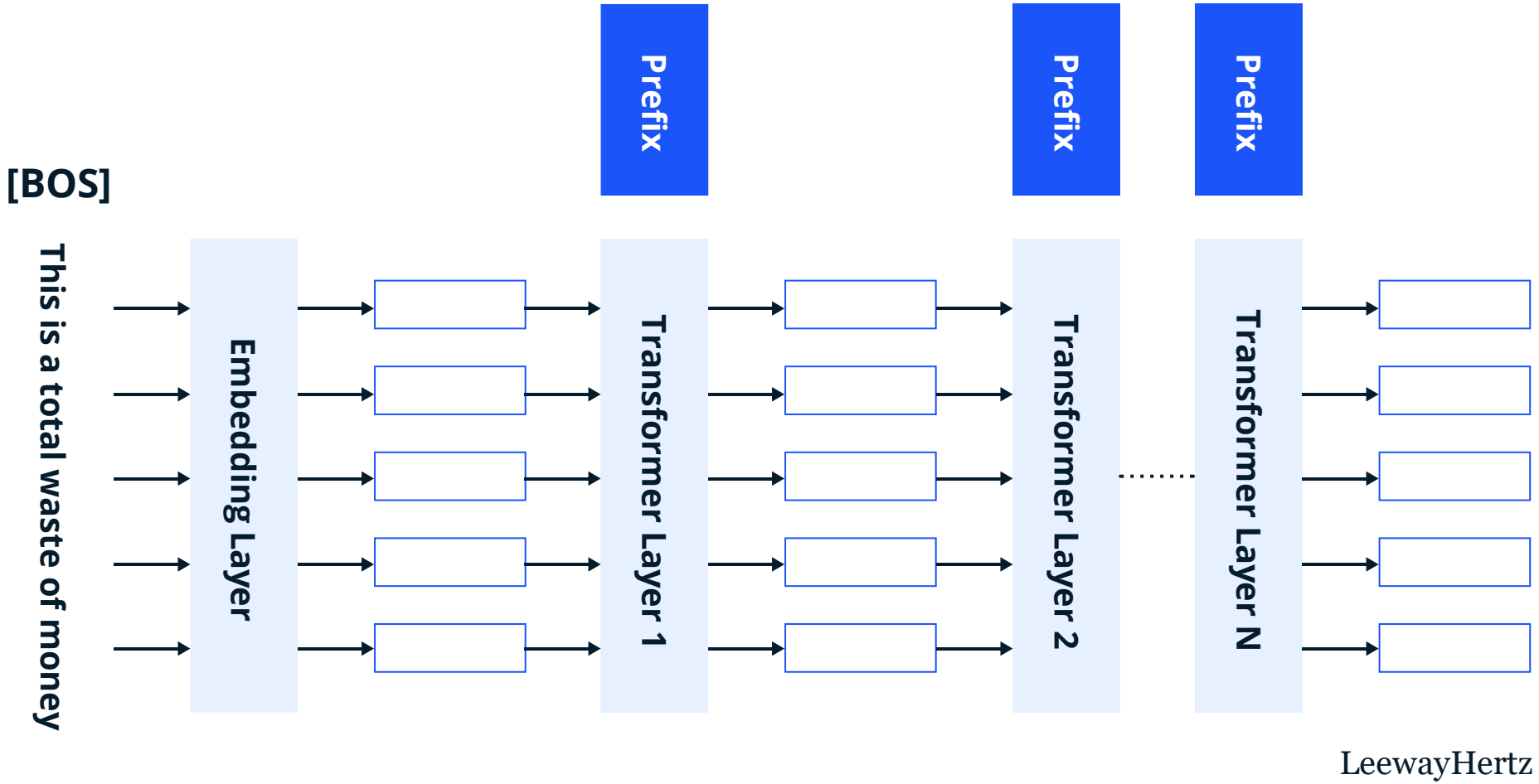
Advantages of Re-Paramaterization Methods

- These methods are faster to tune than standard full model finetuning.
- It is straight-forward to swap between tasks by swapping in and our just the tuned weights.
- Empirically, LoRA has been shown to be very effective on a variety of tasks.
- QLoRA (Quantized LoRA) is even more efficient than LoRA.

"QLoRA: Efficient Finetuning of Quantized LLMs." 2023. https://arxiv.org/abs/2305.14314

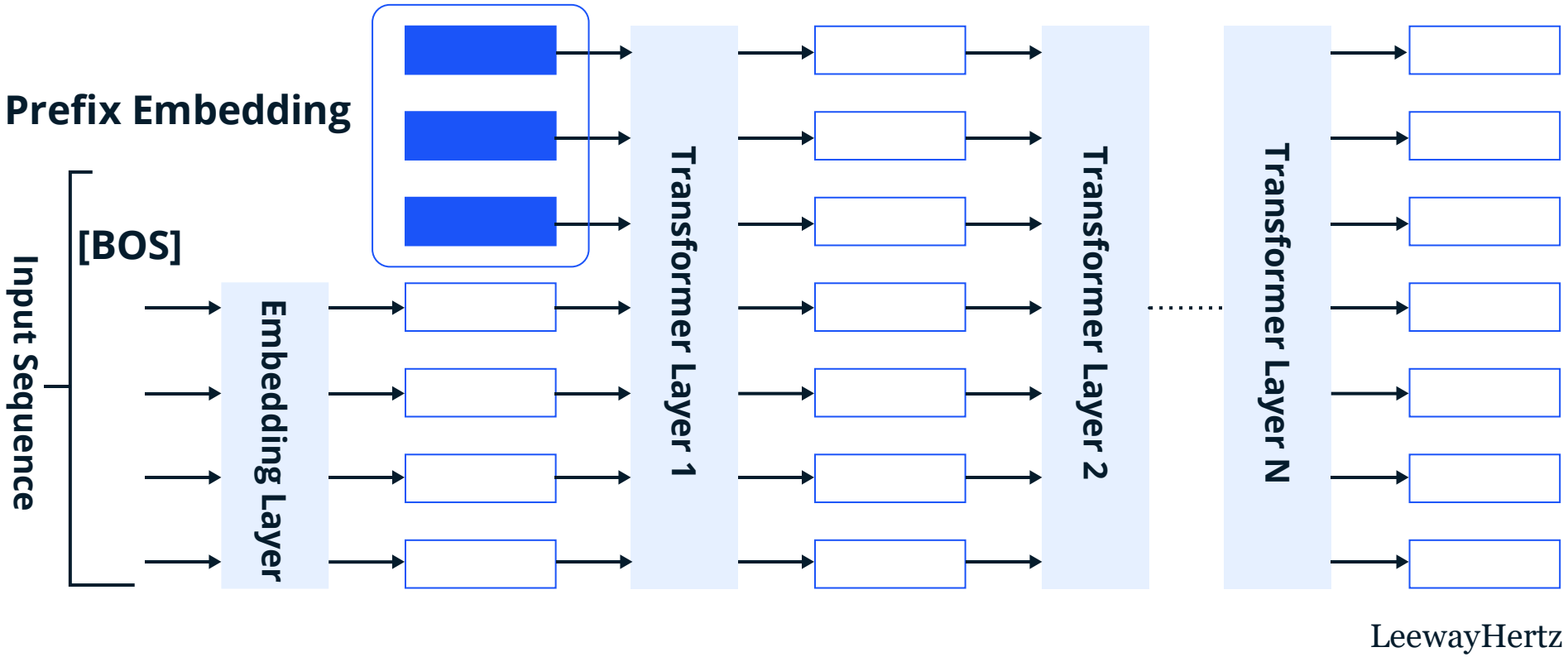
Summary

Prefix Tuning

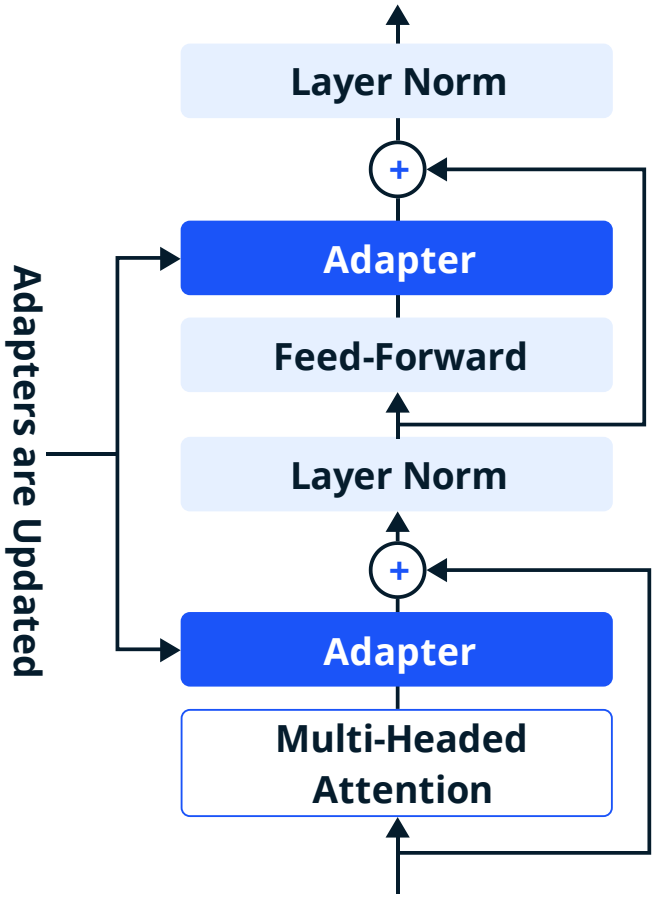


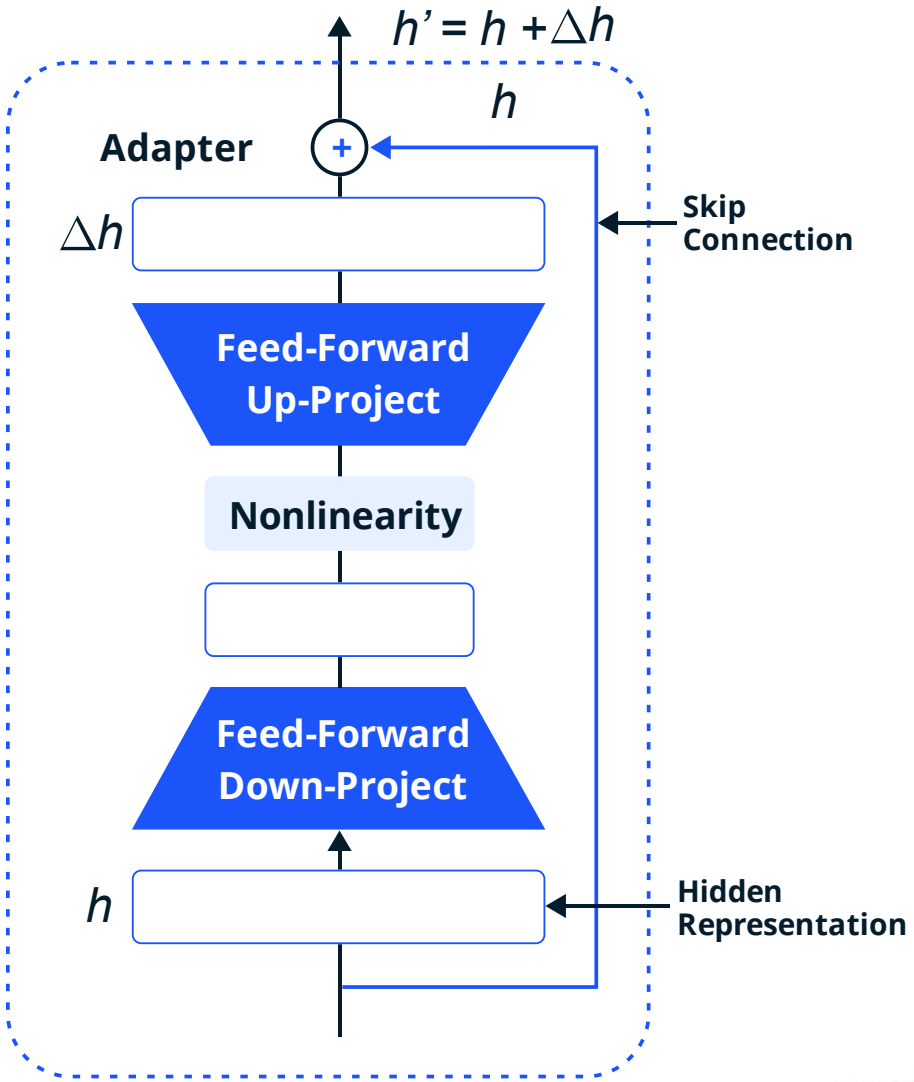
LeewayHertz

Prompt Tuning

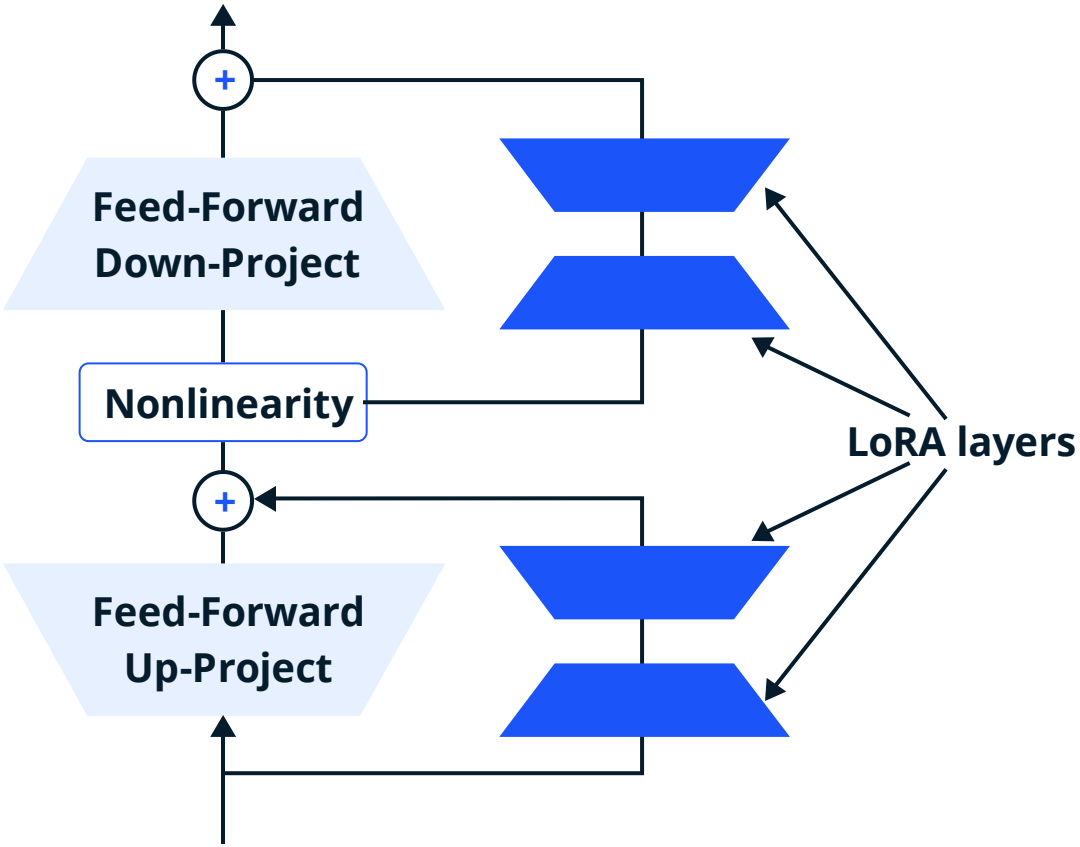


Adapters

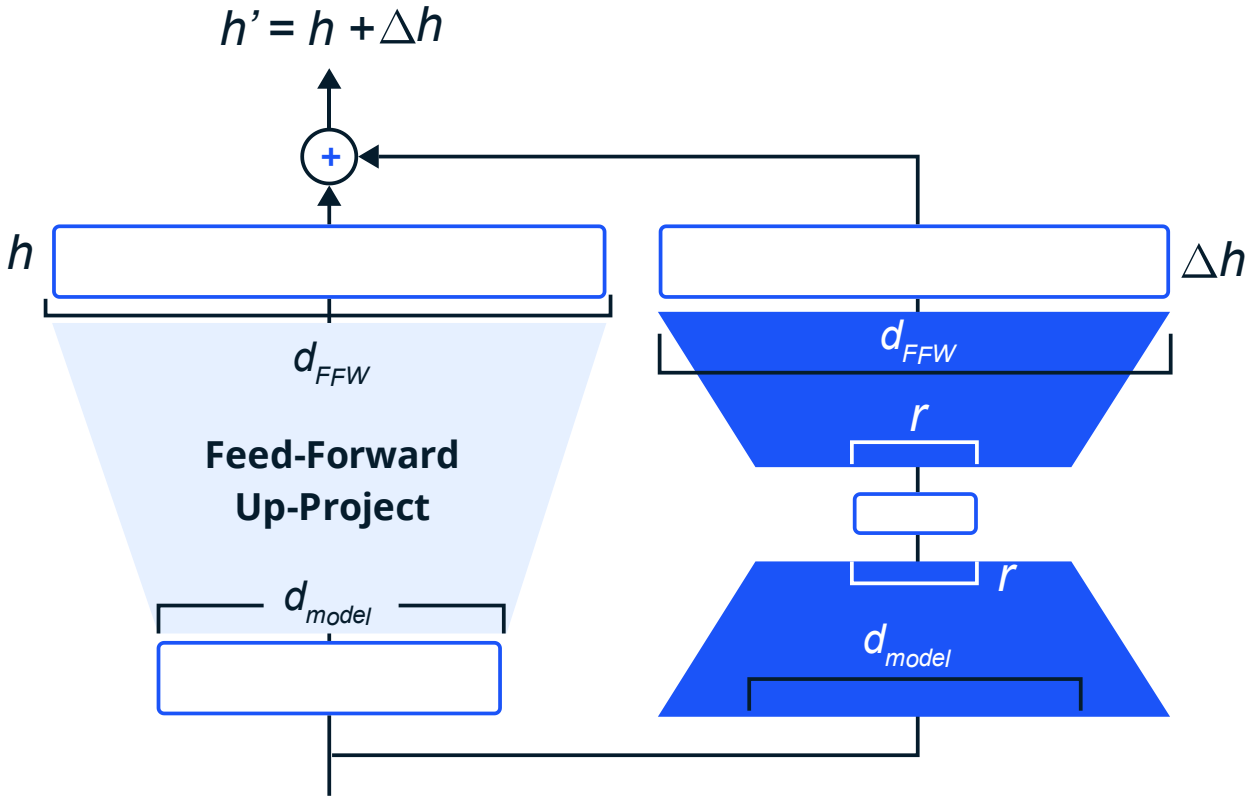

Layer Norm
+
Adapter
Feed-Forward
Layer Norm
+
Adapter
Multi-Headed Attention
Adapters are Updated


h' = h + Δh
h
Adapter
+
Δh
Skip Connection
Feed-Forward Up-Project
Nonlinearity
Feed-Forward Down-Project
h
Hidden Representation

LoRA



LeewayHertz



LeewayHertz

Training Subset of Existing Parameters

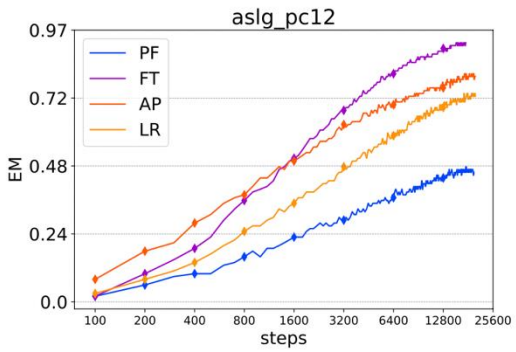
- Manually chose what to tune
  - Just tune the last few layers
  - Just tune the bias terms (BitFit)
- Learn which parameters need to be tuned (DiffPruning)

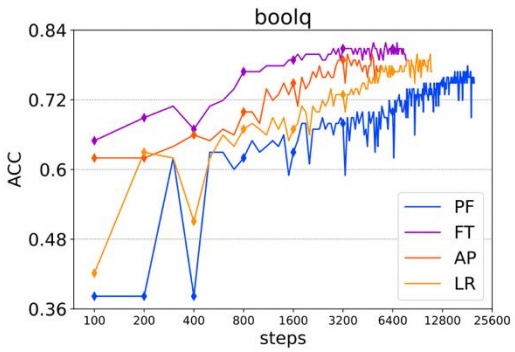
Summary

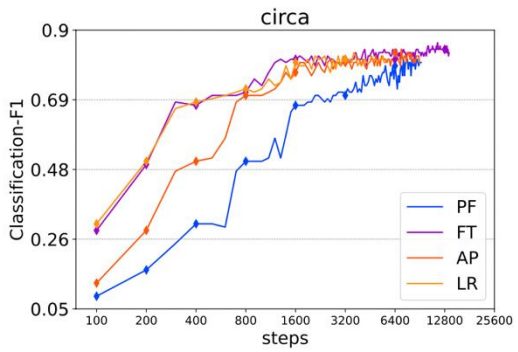
Name & Refs	Method	#Params
SEQUENTIAL ADAPTER Houlsby et al. (2019) COMPACTER Mahabadi et al. (2021a) ADAPTERDROP Rücklé et al. (2021)	$\text{LayerNorm}(\mathbf{X} + \mathbf{H}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{H}(\mathbf{X})))$ $\text{LayerNorm}(\mathbf{X} + \mathbf{F}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{F}(\mathbf{X})))$ $\text{ADT}(\mathbf{X}) = \mathbf{X} + \sigma(\mathbf{X}\mathbf{W}_{d_h \times d_m})\mathbf{W}_{d_m \times d_h}, \ \sigma = \text{activation}$	$L \times 2 \times (2d_h d_m)$ $L \times 2 \times (2(d_h + d_m))$ $(L - n) \times 2 \times (2d_h d_m)$
PARALLEL ADAPTER He et al. (2022)	$\text{LayerNorm}(\mathbf{X} + \mathbf{H}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{X}) + \mathbf{H}(\mathbf{X}))$ $\text{LayerNorm}(\mathbf{X} + \mathbf{F}(\mathbf{X})) \rightarrow \text{LayerNorm}(\mathbf{X} + \text{ADT}(\mathbf{X}) + \mathbf{F}(\mathbf{X}))$ $\text{ADT}(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}_{d_h \times d_m})\mathbf{W}_{d_m \times d_h}, \ \sigma = \text{activation}$	$L \times 2 \times (2d_h d_m)$
ADAPTERBIAS	$\text{LayerNorm}(\mathbf{X} + \mathbf{F}(\mathbf{X})) \rightarrow \text{LayerNorm}(\text{ADT}(\mathbf{X}) + \mathbf{F}(\mathbf{X}))$ $\text{ADT}(X) = \mathbf{X}\mathbf{W}_{d_h \times 1}\mathbf{W}_{1 \times d_h}$	$L \times 2 \times d_h$
PREFIX-TUNING Li & Liang (2021)	$\mathbf{H}_i = \text{ATT}(\mathbf{X}\mathbf{W}_q^{(i)}, [\text{MLP}_k^{(i)}(\mathbf{P}'_k) : \mathbf{X}\mathbf{W}_k^{(i)}], [\text{MLP}_v^{(i)}(\mathbf{P}'_v) : \mathbf{X}\mathbf{W}_v^{(i)}])$ $\text{MLP}^{(i)}(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}_{d_m \times d_m})\mathbf{W}_{d_m \times d_h}^{(i)}$ $P' = \mathbf{W}_{n \times d_m}$	$n \times d_m + d_m^2$ $+ L \times 2 \times d_h d_m$
LORA Hu et al. (2021a)	$\mathbf{H}_i = \text{ATT}(\mathbf{X}\mathbf{W}_q^{(i)}, \text{ADT}_k(\mathbf{X}) + \mathbf{X}\mathbf{W}_k^{(i)}, \text{ADT}_v(\mathbf{X}) + \mathbf{X}\mathbf{W}_v^{(i)})$ $\text{ADT}(\mathbf{X}) = \mathbf{X}\mathbf{W}_{d_h \times d_m}\mathbf{W}_{d_m \times d_h}$	$L \times 2 \times (2d_h d_m)$
BITFIT Zaken et al. (2021)	$f(\mathbf{X}) \rightarrow f(\mathbf{X}) + \mathbf{B}, \ \text{for all function } f$	$L \times (7 \times d_h + d_m)$

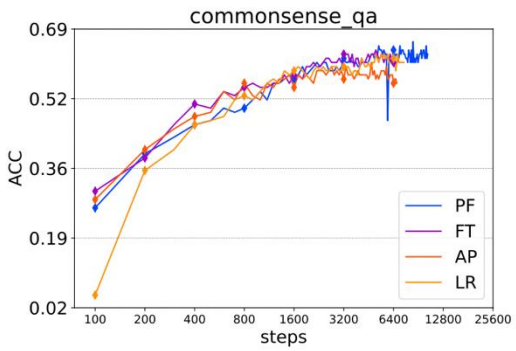
Results

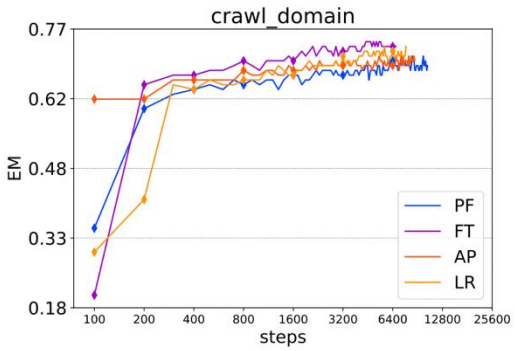
If you have the resources, full fine-tuning tends to work the best.

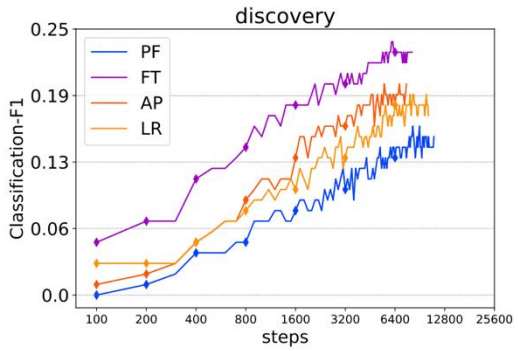


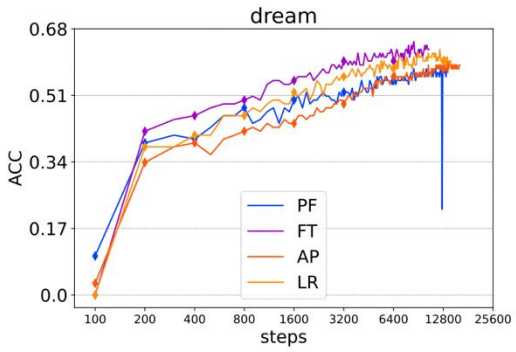


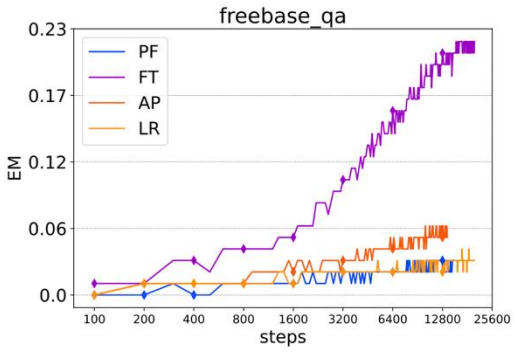


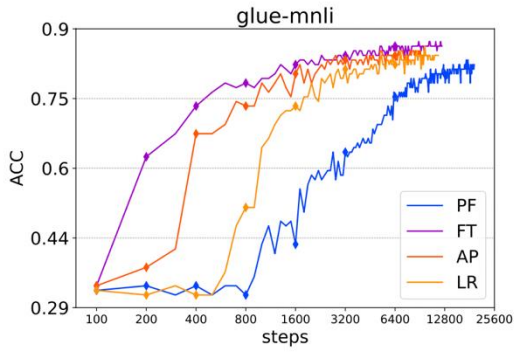










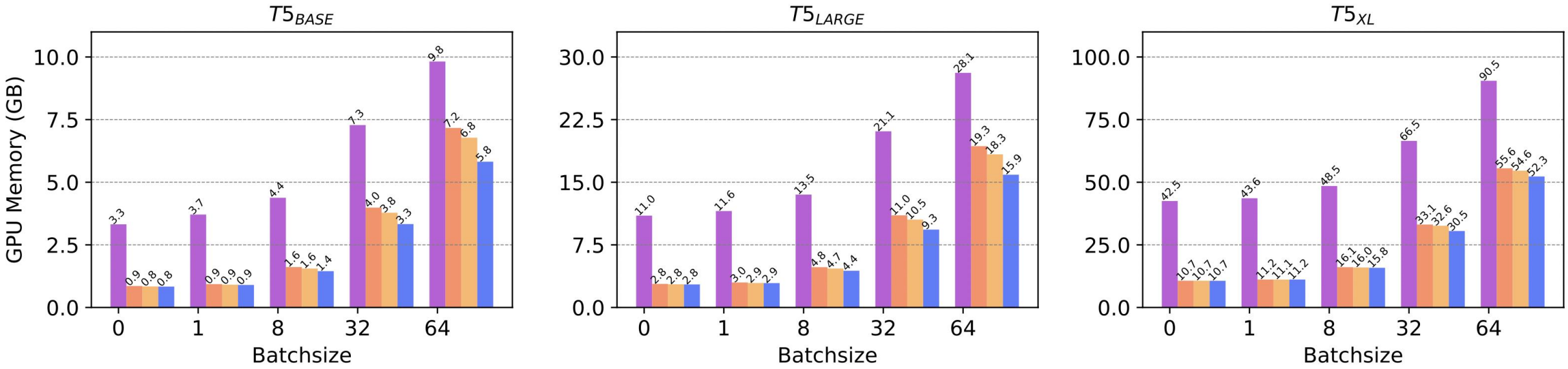


PF: prefix tuning
FT: full fine-tuning
AP: adapter
LR: LoRA

This survey overall found:
Full fine-tuning >
LoRA >
Adapters >
Prefix Tuning >
Prompt Tuning
In terms of performance.

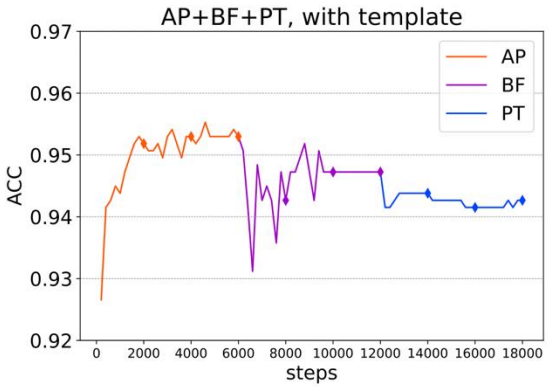
Plots for many more tasks can be found in the paper.

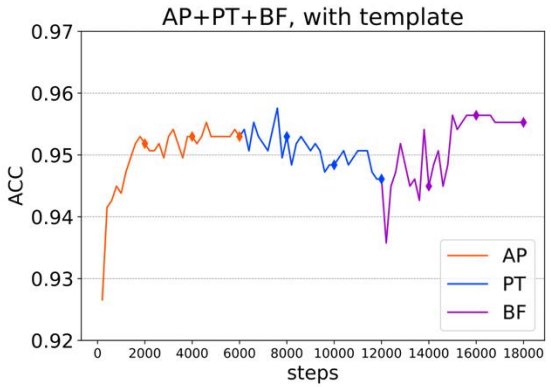
What does memory usage look like?

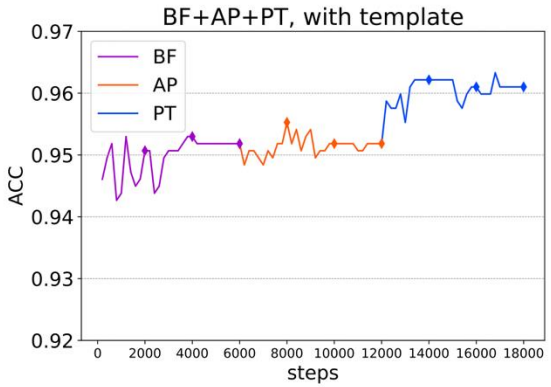


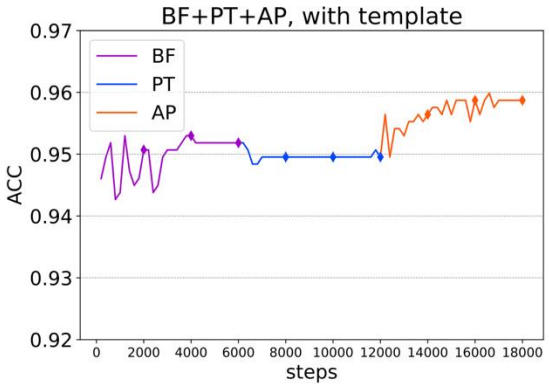
FT: full fine-tuning
AP: adapter
LR: LoRA
BF: BitFit
Prompt tuning and prefix tuning not included because they use the same amount of memory as full fine-tuning

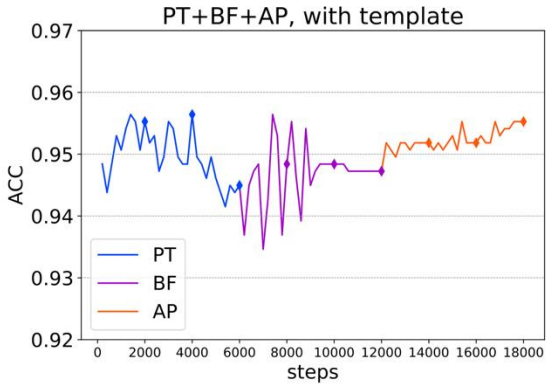
Can the methods be combined?

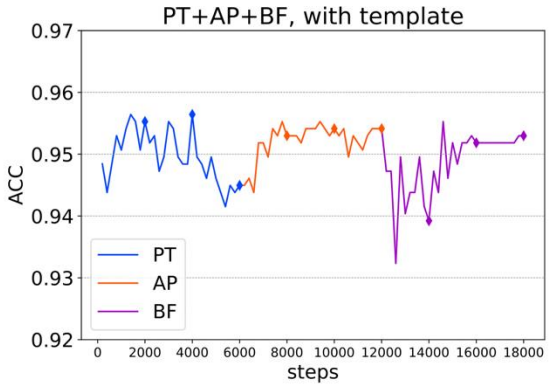












FT: BitFit
AP: adapter
PT: prompt tuning

Results on SST-2 sentiment classification

Param-Efficient Finetuning Options Available to You

How can you use parameter-efficient tuning?

- OpenAI finetuning API
  - It is extremely likely they are using a version of one of the methods described.
  - Unfortunately, we can only rely on speculation.
- HuggingFace PEFT Library
  - LoRA, prefix tuning, prompt tuning, and $(IA)^3$ all implemented.
  - Several different models available to be adapted.

RECAP: Solutions to Improve Efficiency

- ✅ Avoid finetuning entirely
  - In-context learning
- ✅ Parameter-efficient finetuning
- Multi-task finetuning → instruction finetuning