Parameter Efficient Learning for Transformers

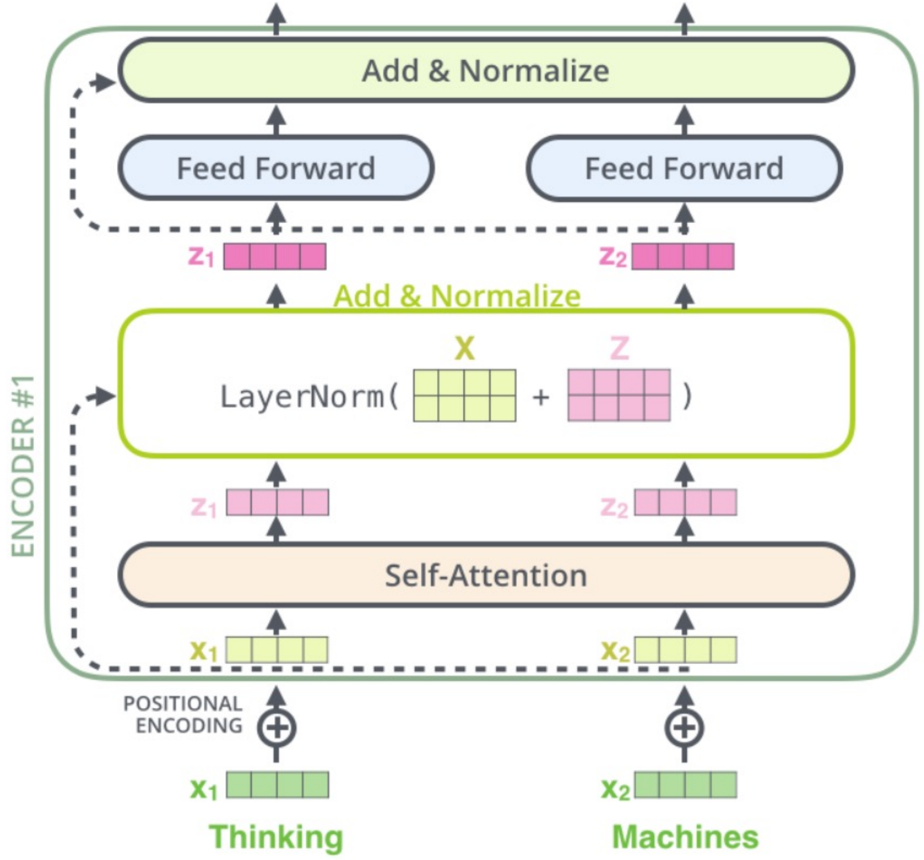
22.5.6

Outline

- Review: Transformer Basics

- Parameter-Efficient Learning for Transformers
  - Intrinsic Dimensionality of Transformers
  - Parameter-Efficient Tuning Methods
  - Theoretical Perspectives

Transformer Block



$$\mathrm{FFN}(\boldsymbol{x}) = \mathrm{ReLU}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2$$

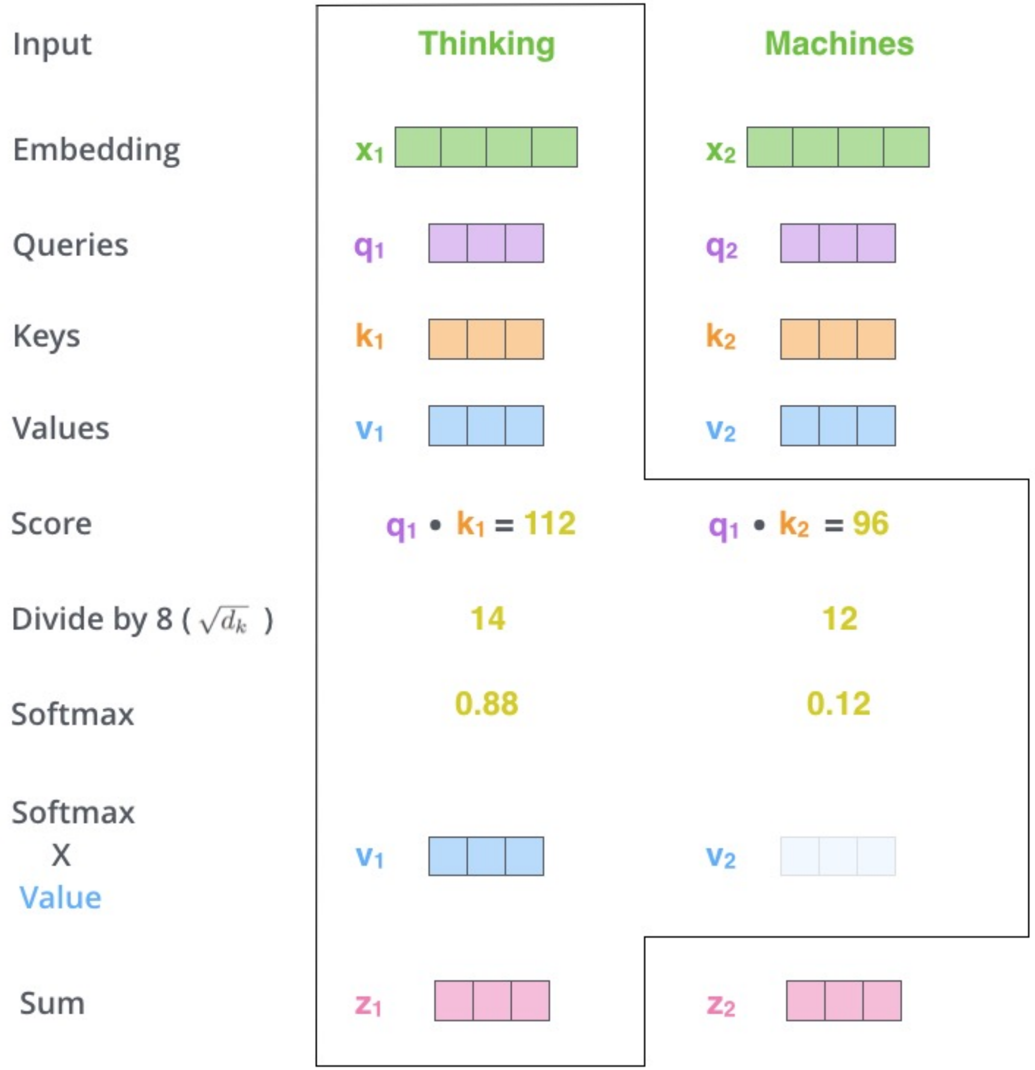
$$\mathrm{MHA}(\boldsymbol{C}, \boldsymbol{x}) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_{\mathrm{h}})\boldsymbol{W}_o$$

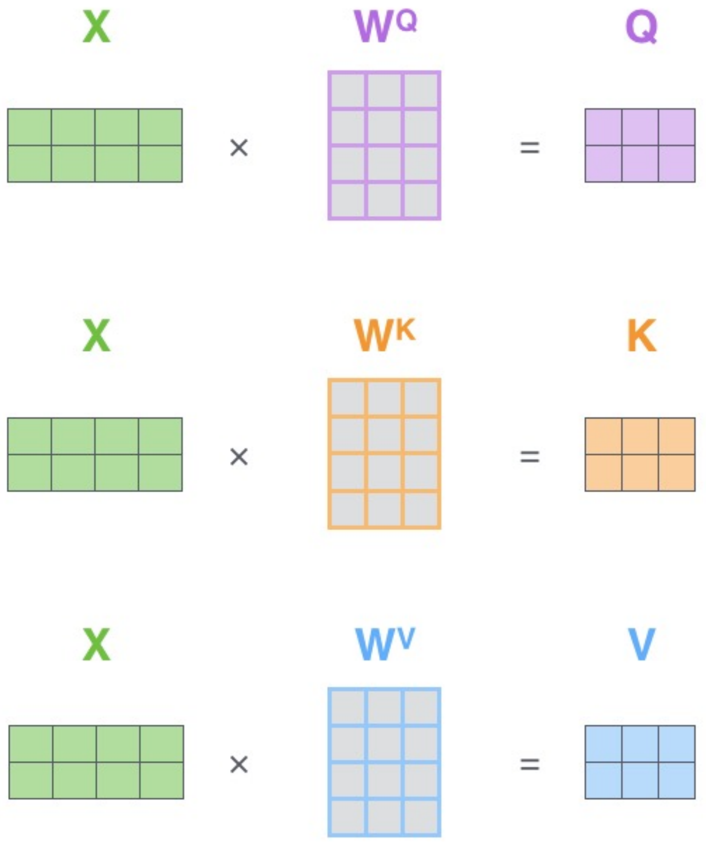
$$\mathrm{head}_{\mathrm{i}} = \mathrm{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)})$$

http://jalammar.github.io/illustrated-transformer/

Self Attention

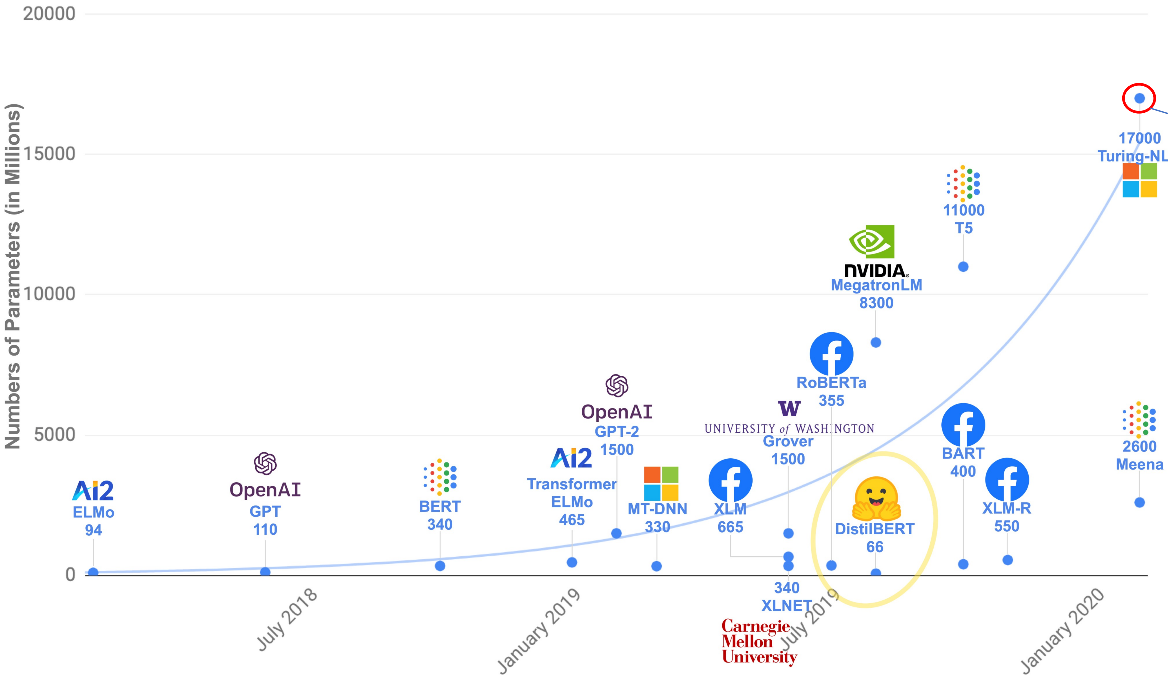
$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}$$

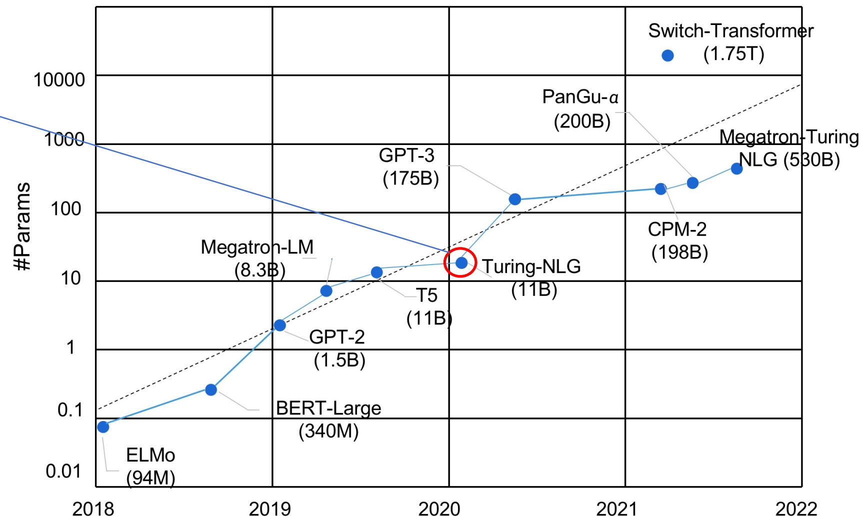




http://jalammar.github.io/illustrated-transformer/

Transformer are big models





Fine-Tuning as Predominant Paradigm

	Rank	Name	Model	URL	Score
+	1	Liam Fedus	ST-MoE-32B		91.2
	2	Microsoft Alexander v-team	Turing NLR v5		90.9
	3	ERNIE Team - Baidu	ERNIE 3.0		90.6
	4	Yi Tay	PaLM 540B		90.4
+	5	Zirui Wang	T5 + UDG, Single Model (Google Brain)		90.4
+	6	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4		90.3
	7	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8
+	8	T5 Team - Google	T5		89.3
	9	SPoT Team - Google	Frozen T5 1.1 + SPoT		89.2
+	10	Huawei Noah's Ark Lab	NEZHA-Plus		86.7

SuperGLUE Leaderboard (22.05)

Drawbacks of Full Fine Tuning

- Parameter Inefficiency:
  - An entire new model is required for every downstream task.
  - Hard to storing different instances for different tasks as the model scales.
- Resource-intensive deployment and computation:

```
heguande@jungpu34 ~/codes/sota_lm
% python run_lm_large.py --bs 32
```

```
RuntimeError: CUDA out of memory. Tried to allocate 148.00 MiB
```

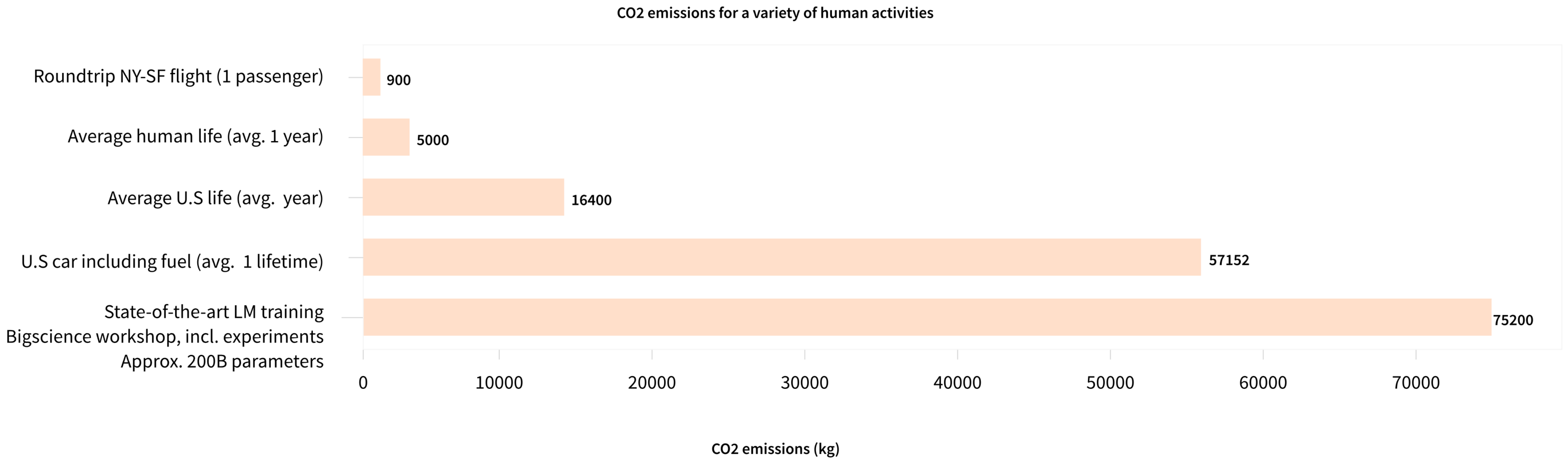
which has resulted in scarce usage of large models in research

Table 1: The usage of models of different sizes in research published in NLP conferences, the statistic is based on 1000 randomly selected papers. Large PLMs are defined as PLMs with over 1 billion parameters.

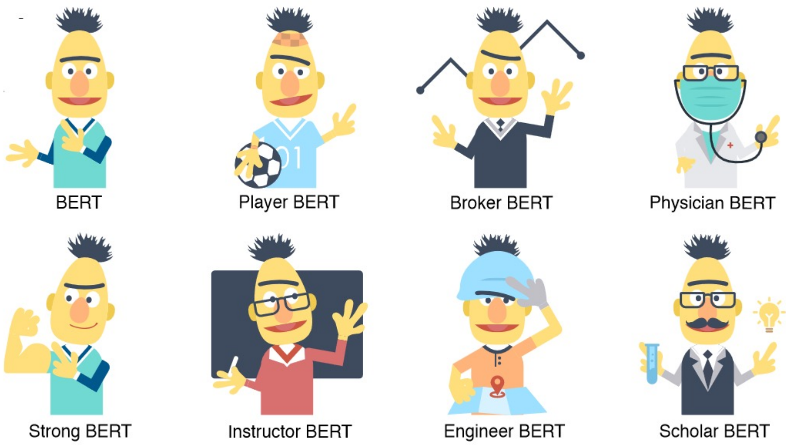
Venue	No PLMs	Small PLMs	Large PLMs	Per. of Large PLMs
ACL 2021	41	151	8	4.0%
EMNLP 2021	46	150	4	2.0%
NAACL 2021	37	158	5	2.5%
ACL 2020	107	92	1	0.5%
EMNLP 2020	62	137	1	0.5%

Drawbacks of Full Fine Tuning

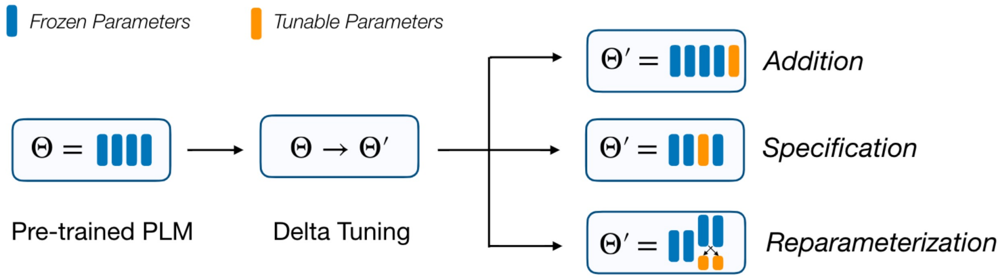
- Not Environmental Friendly



Parameter Efficient Tuning



- Only updates a small number of parameters.
- Achieves comparable results to full FT.
- Several implementation ways:
  - **Addition-based** methods introduce extra trainable neural modules or parameters that do not exist in the original model;
  - **Specification-based** methods specify certain parameters in the original model or process become trainable, while others frozen;
  - **Reparameterization-based** methods reparameterize existing parameters to a parameter-efficient form by transformation.



Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models (2022)

Intrinsic Dimensionality

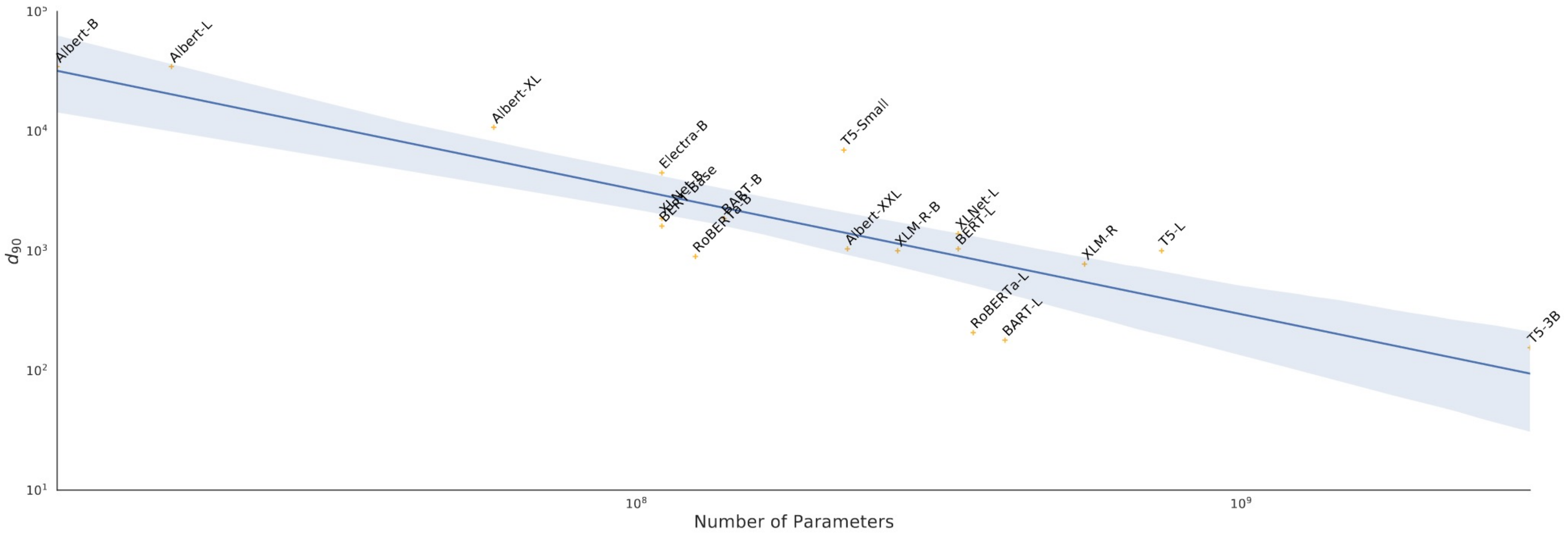
- An objective function's intrinsic dimension:
  - Measures the minimum number of parameters needed to reach satisfactory solutions to the objective.
  - Represents the lowest dimensional subspace in which one can optimize the original objective function to within a certain level of approximation error.

- Structure Aware Intrinsic Dimension: $\theta_i^D = \theta_{0,i}^D + \lambda_i P(\theta^{d-m})_i$

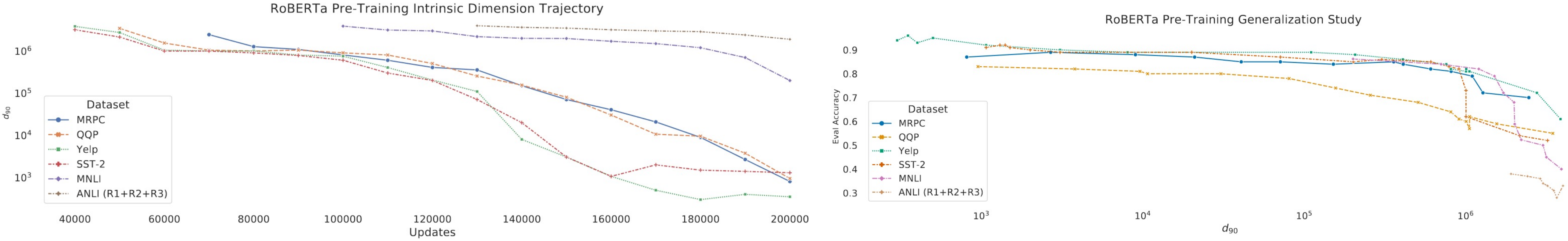
- A *satisfactory solution* is defined as being 90% of the full training metric (d_{90}).

Intrinsic Dimensionality of Transformers

- Larger models tend to have a smaller intrinsic dimension.
- Pre-training implicitly optimizes the ***description length*** over the average of NLP tasks.
- Within the same window of number of parameters, pre-training methodology becomes essential. (e.g. RoBERTa beats BERT)



Intrinsic dimension for a large set of pre-trained models



Intrinsic dimension, pre-training, and generalization

Generalization Bounds through Intrinsic Dimension

Definition 1. (γ, S) *compressible using helper string* s

Suppose $G_{\mathcal{A},s} = \{g_{\theta,s} | \theta \in \mathcal{A}\}$ *is a class of classifiers indexed by trainable parameters A and fixed strings s. A classifier* f *is* (γ, S)*-compressible with respect to* $G_{\mathcal{A}}$ *using helper string s if there exists* $\theta \in \mathcal{A}$ *such that for any* $x \in S$*, we have for all y*

$$|f(x)[y] - g_{\theta,s}(x)[y]| \leq \gamma \tag{6}$$

Remark 1. *If we parameterize* $f(x;\theta)$ *via the intrinsic dimension approach as defined in Equation 1, then* f *is compressible losslessly using a helper string consisting of the random seed used to generate the static random projection weights and the initial pre-trained representation* θ_0^D*. Therefore we say* f *parameterized by either DID or SAID is* $(0, S)$ *compressible.*

Theorem 1. *Let* f *be a function which is parameterized by* θ^D *as described in Equation 1 with a total of* d *trainable intrinsic parameters on a dataset with* m *samples. Then with a high probability, we can state the following asymptotic generalization bound*

$$\mathcal{L}_0(f) \leq \hat{\mathcal{L}}_0(f) + \mathcal{O}\left(\sqrt{\frac{d}{m}}\right) \tag{5}$$

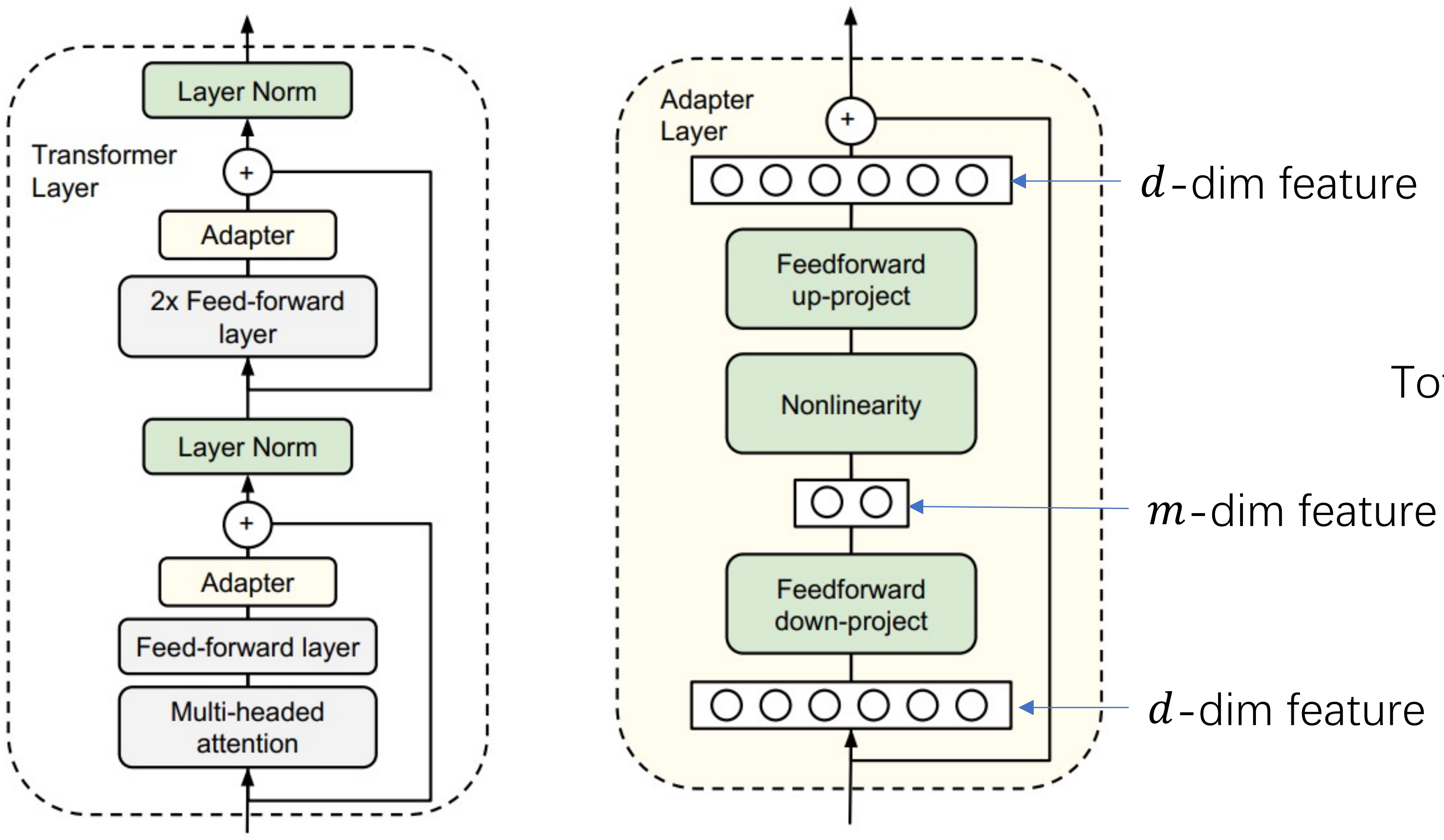
Stronger generalization bounds for deep nets via a compression approach ICML'18

Delta Tuning (Pamameter Efficient Tuning)

- Addition-based Methods:
  - Adapter and its variants
  - Prefix Tuning

- Specification-based Methods:
  - BitFit
- Reparameterization-based Methods:
  - LoRA

Adapter Module with Transformer



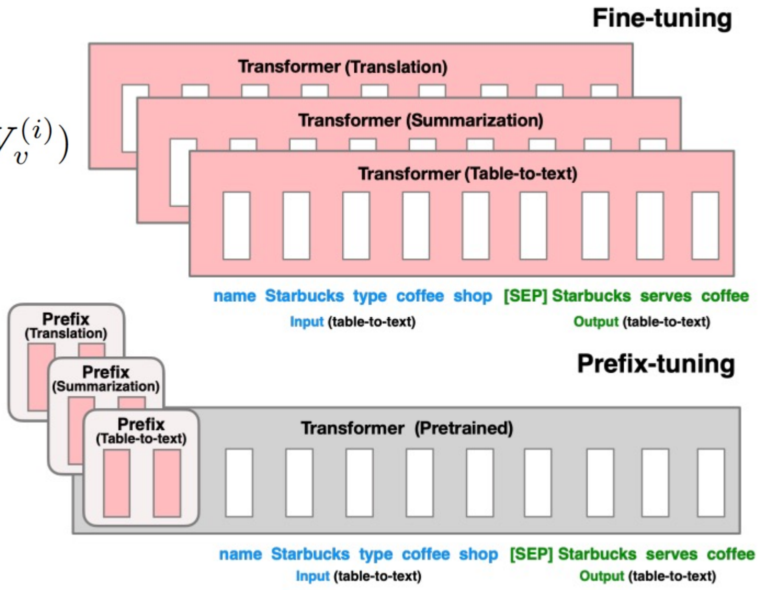
Total: $2md + d + m$

- For each task, the adapter, the layer normalization parameters, and the final task specific layer are trained.

Parameter-Efficient Transfer Learning for NLP ICML'19

Prefix Tuning

$$\text{head}_\text{i} = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)})$$



- Intuition: Prompting or in-context learning
  - GPT-3 can be deployed **without task-specific tuning** by prepending a natural language task instruction and a few examples to the task input.
  - However, optimization over the discrete instructions is challenging.

$$\text{head}_i = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \text{concat}(\boldsymbol{P}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}), \text{concat}(\boldsymbol{P}_v^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)}))$$

- Prefix tuning prepends several tunable prefix vectors to keys and values of the multi-head attention **at every layer.**

- For optimization stability, the prefix embedding matrix is reparameterized by a MLP with a smaller matrix.

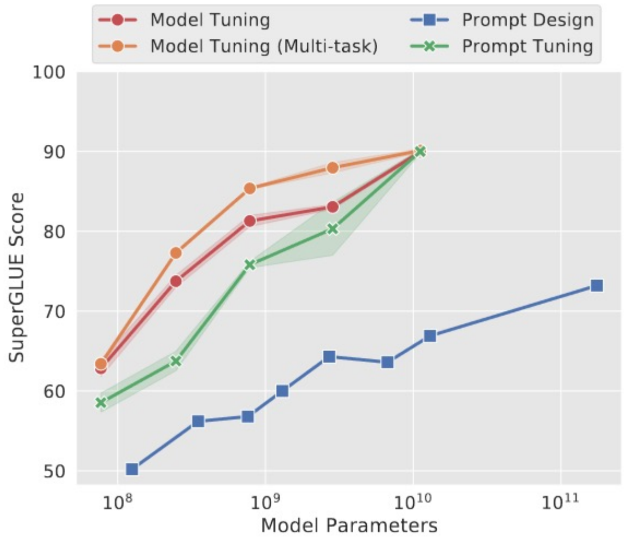
Prefix-Tuning: Optimizing Continuous Prompts for Generation ACL'21

Soft Prompt Tuning

- Simplifying prefix-tuning by only prepending to the input word embeddings **in the first layer.**

- Yields comparable performance on SuperGLUE when the model scales to T5-XXL with 11B parameters.

- Exhibits sensitivity to the length and initialization point.



BitFit: Bias-terms Fine-tuning

- Freezing all the parameters $W^{(\cdot)}$ and $g^{(\cdot)}$ and fine-tuning only the additive bias terms $g^{(\cdot)}$.
- Hypothesis: fine-tuning is mainly about **exposing knowledge** induced by language-modeling training, rather than learning new task-specific linguistic knowledge.

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

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

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

Low-Rank Adaption of Large Language Models

- Over-parameterized models reside on a low intrinsic dimension

- Existing solutions are not good enough:
  - Adapter introduces inference latency.
  - Prefix/Prompt tuning is hard to optimize.
    and will reduce usable seq length.

Batch Size Sequence Length $\|\Theta\|$	32 512 0.5M	16 256 11M	1 128 11M
Fine-Tune/LoRA	1449.4±0.8	338.0±0.6	19.8±2.7
Adapter^{L}	1482.0±1.0 (+2.2%)	354.8±0.5 (+5.0%)	23.9±2.1 (+20.7%)
Adapter^{H}	1492.2±1.0 (+3.0%)	366.3±0.5 (+8.4%)	25.8±2.2 (+30.3%)

- LoRA: Injecting trainable rank decomposition matrices into each layer of the Transformer architecture, while freeze the pre-trained weights.

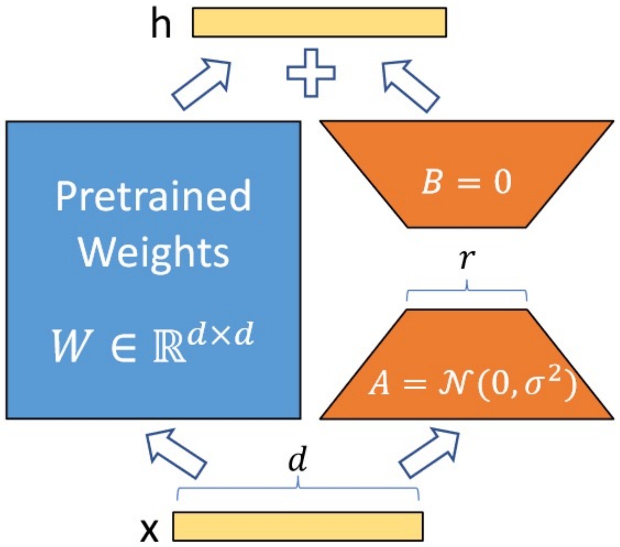
LoRA

- For pre-trained matrix $W_0 \in \mathbb{R}^{d \times k}$, constrain its update by representing the latter with low-rank decomposition:

$$W_0 + \Delta W = W_0 + BA$$

where $\mathrm{B} \in \mathbb{R}^{d \times r}$, $\mathrm{A} \in \mathbb{R}^{r \times k}$, rank $r \ll \min(d, k)$

- During fine-tuning, W_0 is frozen, only apply LoRA on attention weights.



Unified View of Parameter-Efficient Tuning

- A variety of parameter-efficient tuning method that only fine-tune a small number of extra parameters can attain strong performance compared with full fine tuning.

- The critical ingredients for success and connections among various methods are poorly understood.

Unified Formula

- Adapters:

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + f(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$$

- Prefix Tuning:

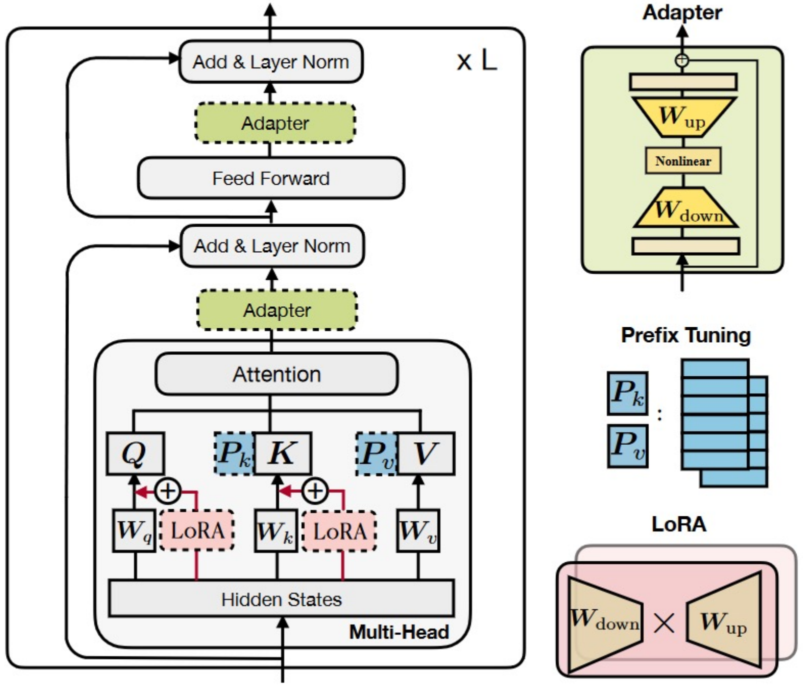
$$\text{head}_i = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \text{concat}(\boldsymbol{P}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}), \text{concat}(\boldsymbol{P}_v^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)}))$$

which can be reformed as:

$$\boldsymbol{h} \leftarrow (1-\lambda(\boldsymbol{x}))\boldsymbol{h} + \lambda(\boldsymbol{x})f(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2$$

- LoRA:

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \boldsymbol{x}\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$$

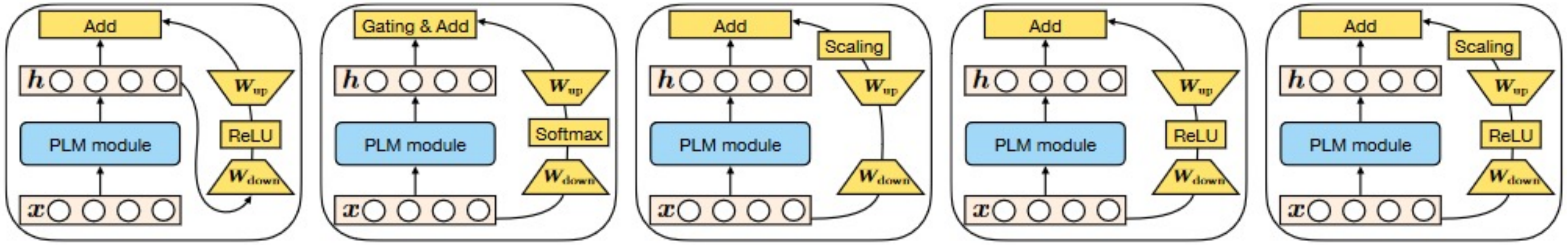


$$\text{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}$$

$$\text{MHA}(\boldsymbol{C}, \boldsymbol{x}) = \text{Concat}(\text{head}_1, \cdots, \text{head}_\text{h})\boldsymbol{W}_o, \ \text{head}_\text{i} = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)}),$$

Design Factors

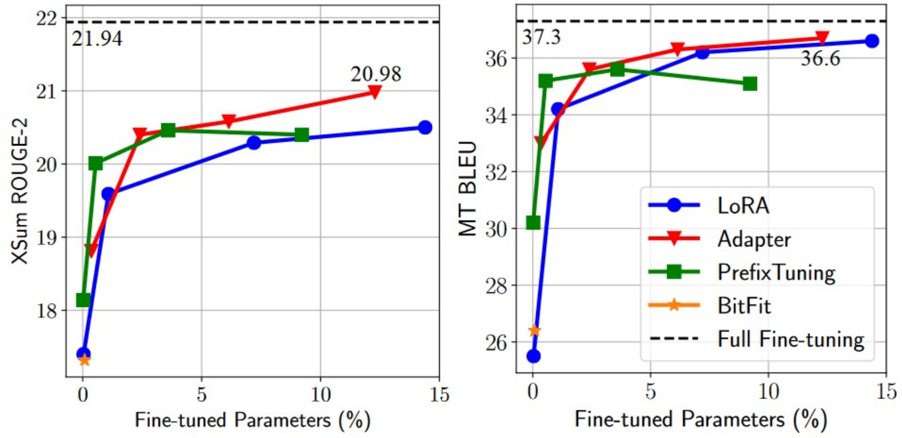
Method	Δh functional form	insertion form	modified representation	composition function
Existing Methods				
Prefix Tuning	$\text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v$	parallel	head attn	$\boldsymbol{h} \leftarrow (1-\lambda)\boldsymbol{h} + \lambda\Delta\boldsymbol{h}$
Adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	sequential	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
LoRA	$\boldsymbol{x}\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$	parallel	attn key/val	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$
Proposed Variants				
Parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Muti-head parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	head attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Scaled parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$



(a) Adapter (b) Prefix Tuning (c) LoRA (d) Parallel Adapter (e) Scaled PA

Results of Existing Methods

Method (# params)	MNLI	SST2
Full-FT (100%)	$87.6_{\pm.4}$	$94.6_{\pm.4}$
Bitfit (0.1 %)	84.7	93.7
Prefix (0.5%)	$86.3_{\pm.4}$	$94.0_{\pm.1}$
LoRA (0.5%)	$87.2_{\pm.4}$	$94.2_{\pm.2}$
Adapter (0.5%)	$87.2_{\pm.2}$	$94.2_{\pm.1}$

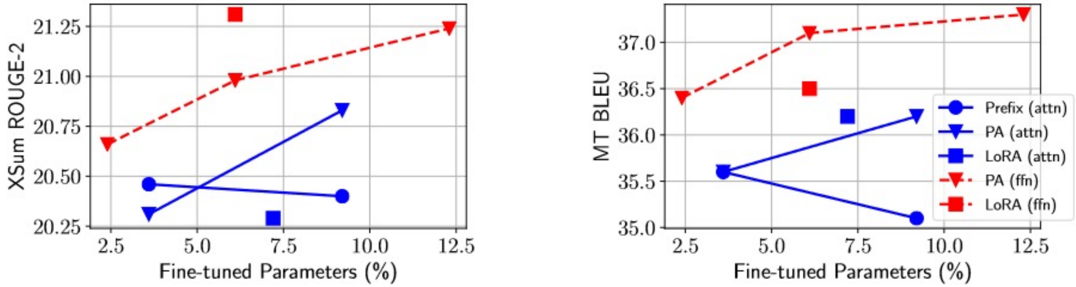


- Existing methods could match Full-FT performance easily on classification tasks.
- Obvious gap presents on generation tasks.

Factor comparation

Method	# params	XSum (R-1/2/L)	MT (BLEU)
Prefix, l=200	3.6%	43.40/20.46/35.51	35.6
SA (attn), r=200	3.6%	42.01/19.30/34.40	35.3
SA (ffn), r=200	2.4%	43.21/19.98/35.08	35.6
PA (attn), r=200	3.6%	43.58/20.31/35.34	35.6
PA (ffn), r=200	2.4%	**43.93/20.66/35.63**	**36.4**

Parallel v.s. Sequential



ffn v.s. attention

Table 4: Results on en-ro dataset.

Method	# params	MT (BLEU)
PA (attn), r=200	3.6%	35.6
Prefix, l=200	3.6%	35.6
MH PA (attn), r=200	3.6%	35.8
Prefix, l=30	0.1%	35.2
-gating, l=30	0.1%	34.9
PA (ffn), r=30	0.1%	33.0
PA (attn), r=30	0.1%	33.7
MH PA (attn), r=30	0.1%	**35.3**

Low parameter budget

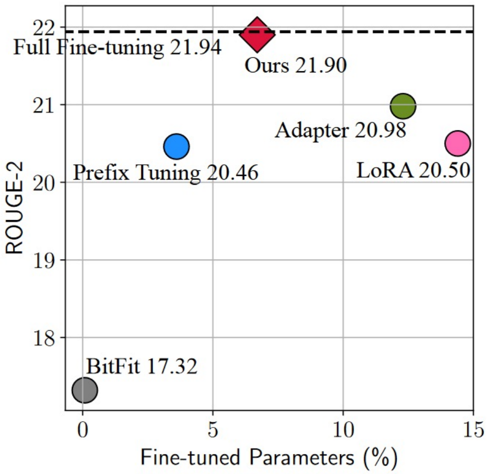
- Parallel design beats sequential ones in all cases.
- FFN modification utilize the added parameters more effectively.
- Modifying head attention achieves best performance on low parameter budget

Composition Function

Method (# params)	XSum (R-1/2/LSum)
LoRA (6.1%), s=4	44.59/21.31/36.25
LoRA (6.1%), s=1	44.17/20.83/35.74
PA (6.1%)	44.35/20.98/35.98
Scaled PA (6.1%), s=4	**44.85/21.54/36.58**
Scaled PA (6.1%), trainable s	44.56/21.31/36.29

- The value of s could have a significant effect on the results.
- Scaling composition is better than the vanilla additive one.

Results



Method	# params	XSum (R-1/2/L)	MT (BLEU)
Full fine-tuning†	100%	45.14/22.27/37.25	37.7
Full fine-tuning (our run)	100%	44.81/21.94/36.83	37.3
Bitfit (Ben Zaken et al., 2021)	0.1%	40.64/17.32/32.19	26.4
Prompt tuning (Lester et al., 2021)	0.1%	38.91/15.98/30.83	21.0
Prefix tuning (Li & Liang, 2021), l=200	3.6%	43.40/20.46/35.51	35.6
Pfeiffer adapter (Pfeiffer et al., 2021), r=600	7.2%	44.03/20.89/35.89$_{\pm.13/.10/.08}$	36.9$_{\pm.1}$
LoRA (ffn), r=102	7.2%	44.53/21.29/36.28$_{\pm.14/.07/.10}$	36.8$_{\pm.3}$
Parallel adapter (PA, ffn), r=1024	12.3%	44.71/21.41/36.41$_{\pm.16/.17/.16}$	37.2$_{\pm.1}$
PA (attn, r=30) + PA (ffn, r=512)	6.7%	44.29/21.06/36.12$_{\pm.31/.19/.18}$	37.2$_{\pm.1}$
Prefix tuning (attn, l=30) + LoRA (ffn, r=102)	6.7%	44.84/21.71/36.77$_{\pm.07/.05/.03}$	37.0$_{\pm.1}$
MAM Adapter (our variant, l=30, r=512)	6.7%	**45.06/21.90/36.87**$_{\pm.08/.01/.04}$	**37.5**$_{\pm.1}$

Generation tasks

Method (# params)	MNLI	SST2
Full-FT (100%)	87.6$_{\pm.4}$	94.6$_{\pm.4}$
Bitfit (0.1 %)	84.7	93.7
Prefix (0.5%)	86.3$_{\pm.4}$	94.0$_{\pm.1}$
LoRA (0.5%)	87.2$_{\pm.4}$	94.2$_{\pm.2}$
Adapter (0.5%)	87.2$_{\pm.2}$	94.2$_{\pm.1}$
MAM Adapter (0.5%)	**87.4**$_{\pm.3}$	94.2$_{\pm.3}$

Classification tasks

- MAM Adapter: Prefix Tuning with small bottleneck dim + scaled parallel adapter

Optimization Perspective

- Objective function of the original LM: $\mathcal{F}(\theta)$
- New objective after inducing delta parameters: $\tilde{\mathcal{F}}(\theta, \delta)$
- The starting point is (θ_0, δ_0) and usually we have $\tilde{\mathcal{F}}(\theta, \delta_0) = \mathcal{F}(\theta)$
- Let $\theta^+ = \arg\min_\theta \tilde{\mathcal{F}}(\theta, \delta_0)$ and $\delta^+ = \arg\min_\delta \tilde{\mathcal{F}}(\theta_0, \delta)$

- We are only interested in the gap between $\tilde{\mathcal{F}}(\theta, \delta_0) = \mathcal{F}(\theta)$ (full FT) and $\tilde{\mathcal{F}}(\theta_0, \delta)$ (Parameter-Efficient Tuning).

Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models (2022)

Optimization Perspective

- Low-dimensional representation in solution space:
  - Assume we can embed the original parameters θ to a low dimensional space, i.e. $\theta = \psi(\delta) + \epsilon$, where ϵ is the error term depending on θ_0, θ^+.
  - Then, we have $\tilde{\mathcal{F}}(\theta, \delta_0) = \mathcal{F}(\theta), \quad \tilde{\mathcal{F}}(\theta_0, \delta) = \mathcal{F}(\psi(\delta))$.
  - Let $\delta^+ = \arg\min_\delta \mathcal{F}(\psi(\delta))$, and $\theta^+ = \psi(\delta') + \epsilon'$. Suppose that $\mathcal{F}$ and $\mathcal{F} \circ \psi$ are Lipschitz continuous, we have following bound of the approximation error of delta tuning to the full-parameter FT:

$$\begin{aligned}|\mathcal{F}(\theta^+) - \mathcal{F}(\psi(\delta^+))| &\le |\mathcal{F}(\theta^+) - \mathcal{F}(\psi(\delta'))| + |\mathcal{F}(\psi(\delta')) - \mathcal{F}(\psi(\delta^+))| \\ &\le L_1\|\epsilon'\|_2 + L_2\|\delta' - \delta^+\|_2 \le L_1\|\epsilon'\|_2 + L_2(\|\delta'\|_2 + \|\delta^+\|_2).\end{aligned}$$

- Low dimensional representation in functional space:

$$|\mathcal{F}(\theta) - \hat{\mathcal{F}}(\delta)| < \epsilon,$$

Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models (2022)

Optimal Control Perspective

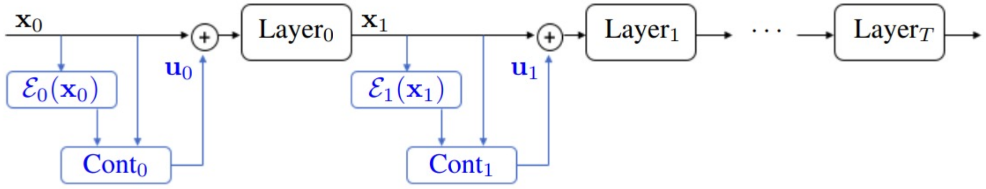
- Deep learning can be interpreted as a optimal control problem (Li et al. ,2017).
- Delta tuning can be viewed as seeking the optimal control of PLMs for specific downstream tasks:

$$\min_{\{\delta^{(0)},\ldots,\delta^{(L-1)}\}} \mathbb{E}_{(x,y)\sim\mathcal{D}_{tr}} \left[S\left(h_o^{(L)}, y\right) + \sum_{j=0}^{L-1} R\left(\delta^{(j)}\right) \right]$$

$$h_o^{(j+1)} = h_o^{(j)} + \mathcal{G}_\theta^{(j)}\left(h_o^{(j)}, \delta^{(j)}\right),\ h_o^{(0)} = z_o = [\mathrm{ANS}],\ 0 \le j \le L-1$$

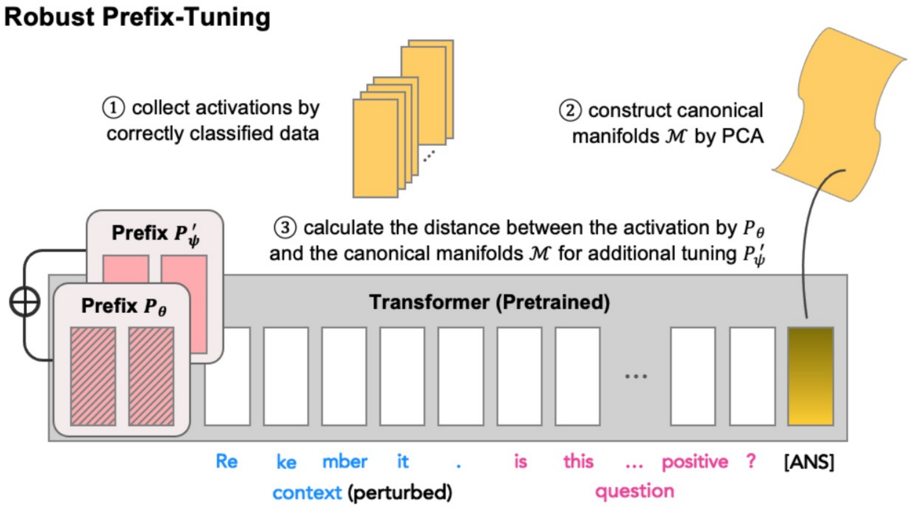
Example: Robust Prefix Tuning

- A instance of seeking the **close-loop control** for robust downstream tasks.



Close-Loop Control

- Pipeline:
  - Collect layer-wise LM activations of correctly classified training examples.
  - Project the activation matrix onto a low-level manifold via PCA.
  - Tuning a additional prefix using the distance between test examples' activation and the manifold.



- Improves robustness over several strong baselines against different textual attacks.

Towards Robust Neural Networks via Close-loop Control ICLR'21
On Robust Prefix-Tuning for Text Classification ICLR'22

Discussion

- Parameter-efficient methods do provide ways to be able to effectively utilize and adapt big transformer-based models.

- The optimal design factors and scale for specific tasks?

- Relation between the pre-trained model
  - Help to understand how pre-trained models work.
  - Potential for correcting model bias.

THANKS

References

- [1] Ding, Ning et al. "Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models." (2022).
- [2] Aghajanyan, Armen et al. "Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning." (2020).
- [3] Arora, Sanjeev et al. "Stronger generalization bounds for deep nets via a compression approach." (2018).
- [4] Houlsby, Neil et al. "Parameter-Efficient Transfer Learning for NLP." (2019).
- [5] Li, Xiang Lisa et al. "Prefix-Tuning: Optimizing Continuous Prompts for Generation." (2021).
- [6] Lester, Brian et al. "The Power of Scale for Parameter-Efficient Prompt Tuning." (2021).
- [7] Zaken, Elad Ben et al. "BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models." (2021).
- [8] Hu, Edward J. et al. "LoRA: Low-Rank Adaptation of Large Language Models." (2021).
- [9] He, Junxian et al. "Towards a Unified View of Parameter-Efficient Transfer Learning." (2021).

References

- [10] Chen, Zhuotong et al. "Towards Robust Neural Networks via Close-loop Control." (2021).
- [11] Yang, Zonghan et al. "On Robust Prefix-Tuning for Text Classification." (2022).