Parameter-Efficient Transfer Learning for CNNs

Teimuraz Saghinadze

MICM

This Project is funded by the European Union HORIZON EUROPE - WIDERA-2021-ACCESS-03 (Twinning)

programme under the grant agreement No. 101078950

Map of Transformer PEFT methods

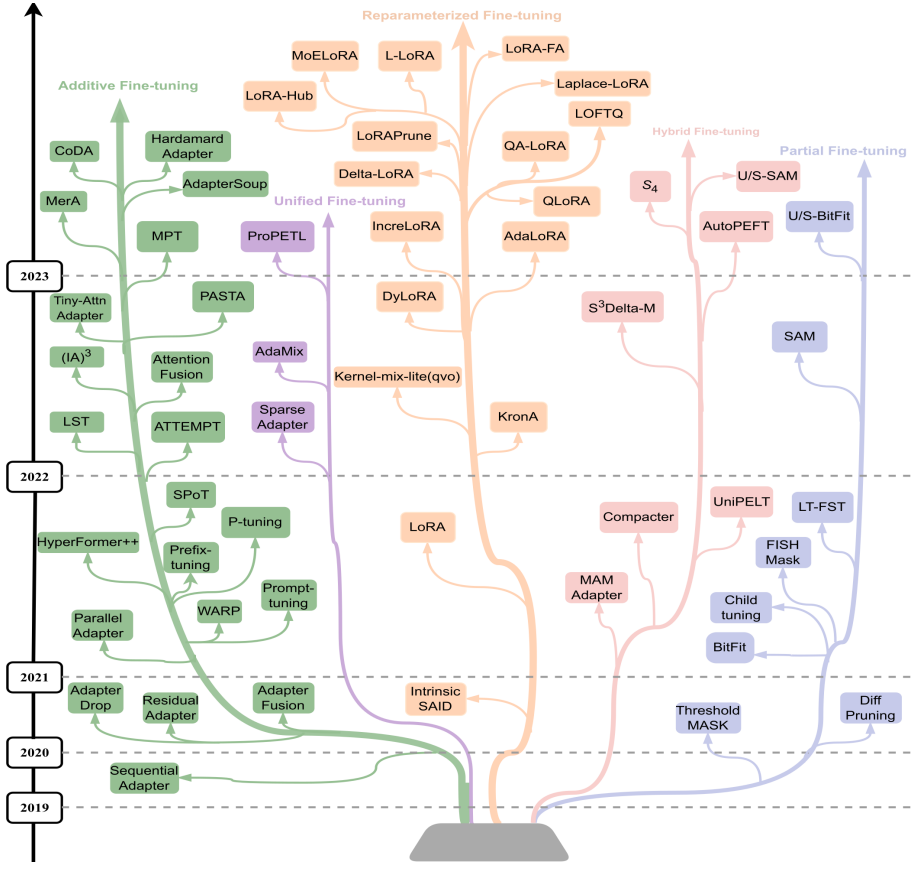
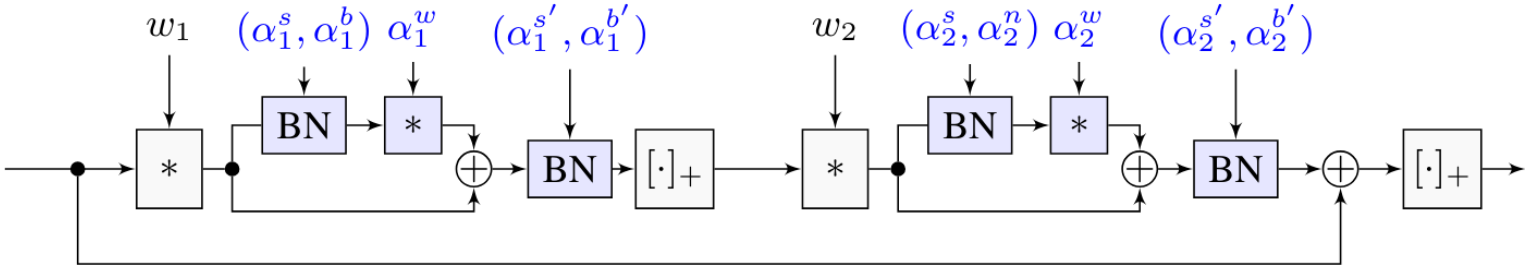


Figure: The evolutionary development of PEFT methods in recent years. [Xu+23]

Properties

- Adequate performance
- Number of elements new parameters should be less than the ones in the original network
- $T_{\text{PEFT}} << T_{\text{FFT}}$

Residual Adapter



- Concept of Adapters/PEFT traces its origin back to CNN-s. [RBV17]
- But it hasn't been investigated to the same degree as PEFT for Transformer architecture.
- But one can capitalize on pre-existing research and produce similar variations of some basic forms of adaptation, with caveats, of course.

Injecting New Parameters Using Separate Layer

Retain original set of parameters as is and inject new ones

- keep original layer $f(z, \Theta)$ intact
- introduce new layer $g(\zeta, \Phi)$ dependent new parameters

Combine those in dimensionally consistent way yields the new value, that'll be used as a substitute

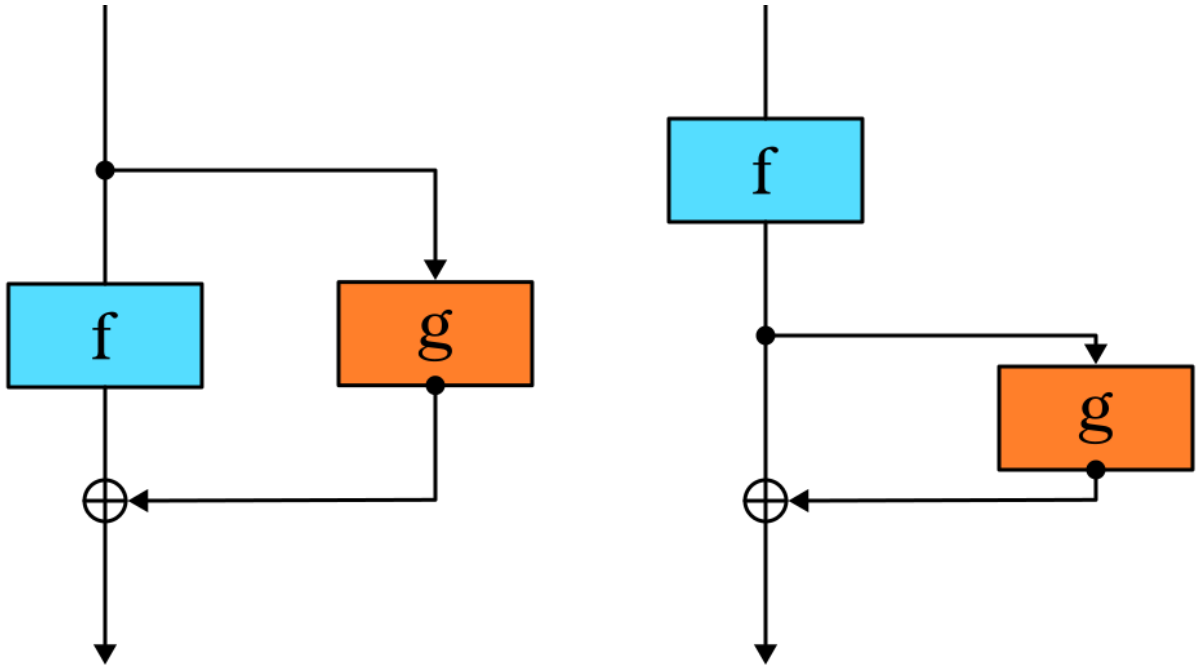
We can use

1. serial: $f(z, \Theta) + g(f(z, \Theta), \Phi)$
2. parallel: $f(z, \Theta) + g(z, \Phi)$

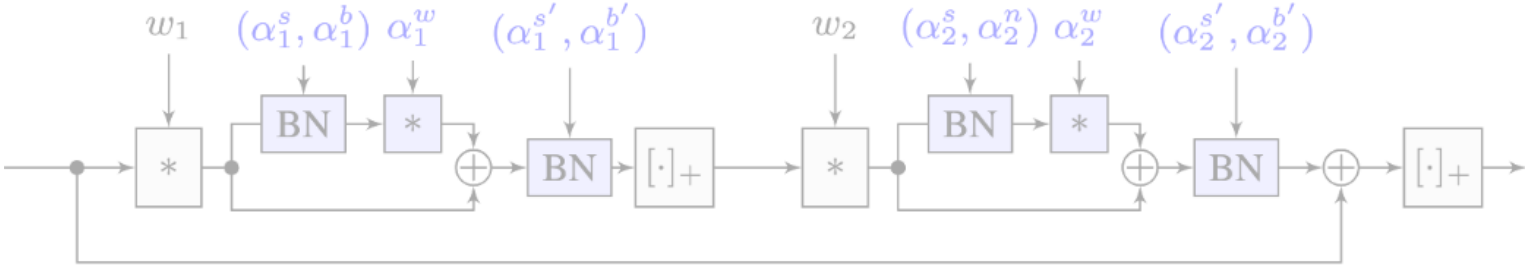
Injecting New Parameters Using Separate Layer

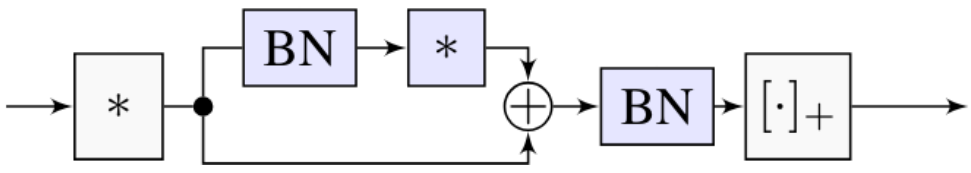
Retain original set of parameters as is and inject new ones

- keep original layer $f(z, \Theta)$ intact
- introduce new layer $g(\zeta, \Phi)$ dependent new parameters



Example of serial arrangement: Residual Adapter





- * - convolution operator
- **BN** - Batch Norm
- $[\cdot]_+$ - ReLU

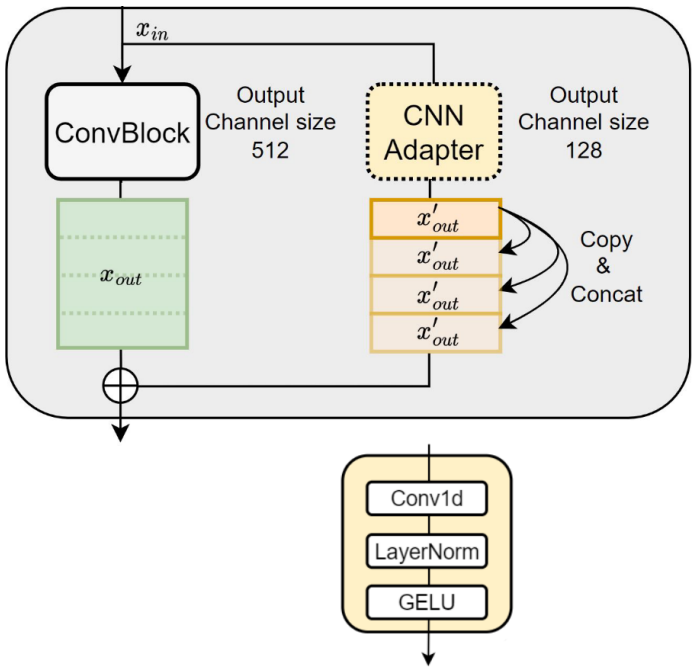
Residual Adapter: Reported Results

Model	#par.	ImNet	Airc.	C100	DPed	DTD	GTSR	Flwr	OGlt	SVHN	UCF	mean	S
# images		1.3m	7k	50k	30k	4k	40k	2k	26k	70k	9k		
Scratch	10×	59.87	57.10	75.73	91.20	37.77	96.55	56.30	88.74	96.63	43.27	70.32	1625
Scratch+	11×	59.67	59.59	76.08	92.45	39.63	96.90	56.66	88.74	96.78	44.17	71.07	1826
Feature extractor	1×	59.67	23.31	63.11	80.33	45.37	68.16	73.69	58.79	43.54	26.80	54.28	544
Finetune	10×	59.87	60.34	82.12	92.82	55.53	97.53	81.41	87.69	96.55	51.20	76.51	2500
LwF [21]	10×	59.87	61.15	82.23	92.34	58.83	97.57	83.05	88.08	96.10	50.04	76.93	2515
BN adapt. [5]	∼ 1×	59.87	43.05	78.62	92.07	51.60	95.82	74.14	84.83	94.10	43.51	71.76	1363
Res. adapt.	2×	59.67	56.68	81.20	93.88	50.85	97.05	66.24	89.62	96.13	47.45	73.88	2118
Res. adapt. decay	2×	59.67	61.87	81.20	93.88	57.13	97.57	81.67	89.62	96.13	50.12	76.89	2621
Res. adapt. finetune all	2×	59.23	63.73	81.31	93.30	57.02	97.47	83.43	89.82	96.17	50.28	77.17	2643
Res. adapt. dom-pred	2.5×	59.18	63.52	81.12	93.29	54.93	97.20	82.29	89.82	95.99	50.10	76.74	2503
Res. adapt. (large)	∼ 12×	67.00	67.69	84.69	94.28	59.41	97.43	84.86	89.92	96.59	52.39	79.43	3131

Table 1: Multiple-domain networks. The figure reports the (top-1) classification accuracy (%) of different models on the decathlon tasks and final decathlon score (S). ImageNet is used to prime the network in every case, except for the networks trained from scratch. The model size is the number of parameters w.r.t. the baseline ResNet. The fully-finetuned model, written blue, is used as a baseline to compute the decathlon score.

Example of parallel arrangement: CHAPTER Adapter

- This adapter was designed for HuBERT to additionally fine tune the embedding layer alongside the rest of transformer blocks [CSL23]
- It hasn't been verified in a pure CNN setting
- But the type of convolution is interesting



CHAPTER Adapter: Reported Results

Method	param(M)	SID	SD	ER
FT	94.7	64.56	**3.58**	68.94
Houlsby_{32}	0.6	87.71	4	65.25
Houlsby_{128}	4.61	89.44	3.91	66.82
CHAPTER	**4.67**	**91.56**	3.84	**70.41**

Table 2: Compared with larger Houlsby adapter. We examine the placement of additional parameters on ER and SD by comparing CHAPTER to Houlsby adapter with bottleneck size 128. For ER, here we present the accuracy of fold 1 testing set.

- speaker identification (SID)
- speaker diarization(SD)
- emotion tasks (ER)

Possible Combinations

- Arrangements:
  1. Serial
  2. Parallel
- Type of 1D convolution
  1. Matching in and out channel count
  2. Copy & concatenate
  3. Group convolution

Possible Combinations

- Arrangements:
  1. Serial
  2. Parallel
- Type of 1D convolution
  1. Matching in and out channel count
  2. Copy & concatenate
  3. Group convolution

Learned scaling parameter λ [He+22]

1. Used
2. Not used

$f(z, \Theta) + g(f(z, \Theta), \Phi, \lambda)$ and $f(z, \Theta) + g(z, \Phi, \lambda)$

Keeping Original Architecture Intact

Alter only the weights $W + \Delta W$ and $b + \Delta b$

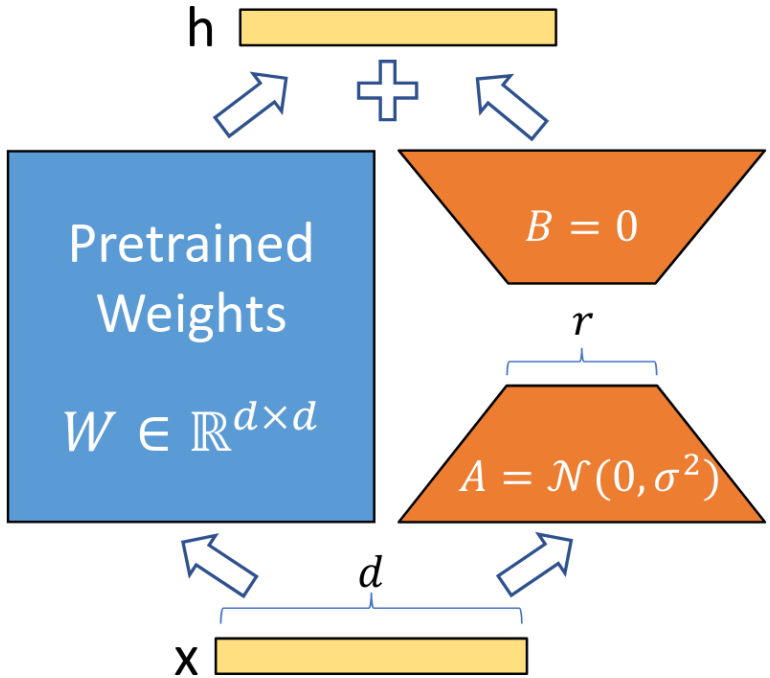
- $W + \Delta W$ variation on LoRA* [Hu+21]
- $b + \Delta b$ bitfit [ZRG22]

* There's no singular-established method of doing this

Quick/Simplified View of LoRA

The idea's quite simple, one needs to create $\Delta W \in \mathbb{R}^{d \times d}$ of rank $r << d$.

- Take $A \in \mathbb{R}^{d \times r}$, $B \in \mathbb{R}^{r \times d} and \lambda \in \mathbb{R}$;
- Take $\Delta W = \lambda AB$.



Issue with Convolution Kernels

- Corresponding object ΔW for convolutions is not necessary a matrix
- $\Delta W \in \mathbb{R}^{c_{out} \times c_{in} \times k_1 \times k_2}$

Issue with Convolution Kernels

- Corresponding object ΔW for convolutions is not necessary a matrix
- $\Delta W \in \mathbb{R}^{c_{out} \times c_{in} \times k_1 \times k_2}$

There are two ways of going about this either:

1. Use tensor decompositions
2. Reframe ΔW as a matrix in $\mathbb{R}^{c_{out} \times (c_{in} \cdot k_1 \cdot k_2)}$

Compressing CNNs

- Event though there's no method for producing low rank adaptation, there are methods to produce low rank approximations to CNN kernels [Pha+20]
- There are 3 methods of doing Thais CPD, TKC and CPD-TKD

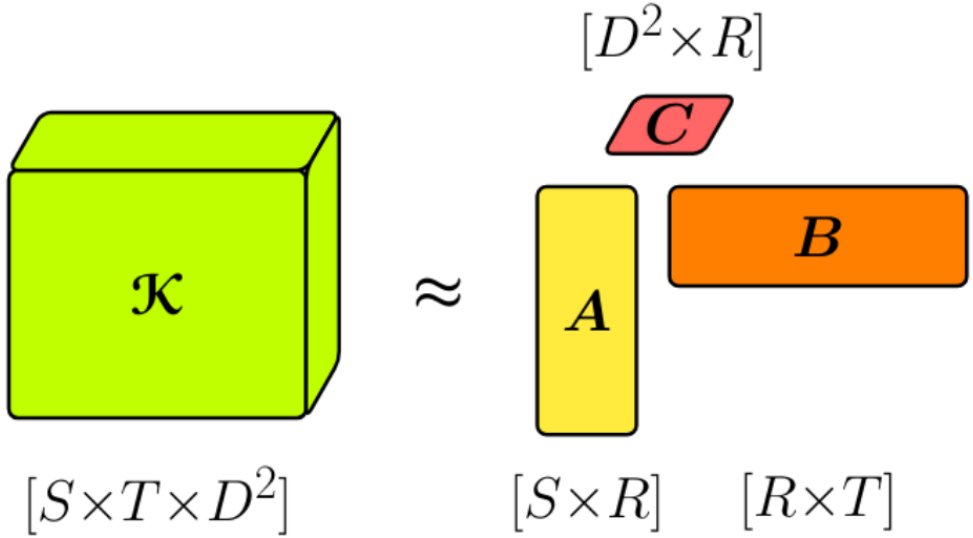
Stable Low-rank Tensor Decomposition for Compression of Convolutional Neural Network - Reported Results

Table 1: Comparison of different model compression methods on ILSVRC-12 validation dataset. The baseline models are taken from Torchvision.

Model	**Method**	**↓ FLOPs**	**Δ top-1**	**Δ top-5**
VGG-16	Asym. [57]	≈ 5.00	-	-1.00
	TKD+VBMF [26]	4.93	-	-0.50
	Our (EPS[1]=0.005)	**5.26**	**-0.92**	**-0.34**
ResNet-18	Channel Gating NN [24]	1.61	-1.62	-1.03
	Discrimination-aware Channel Pruning [58]	1.89	-2.29	-1.38
	FBS [13]	1.98	-2.54	-1.46
	MUSCO [14]	2.42	-0.47	-0.30
	Our (EPS[1]=0.00325)	**3.09**	**-0.69**	**-0.15**
ResNet-50	**Our** (EPS[1]=0.0028)	**2.64**	**-1.47**	**-0.71**

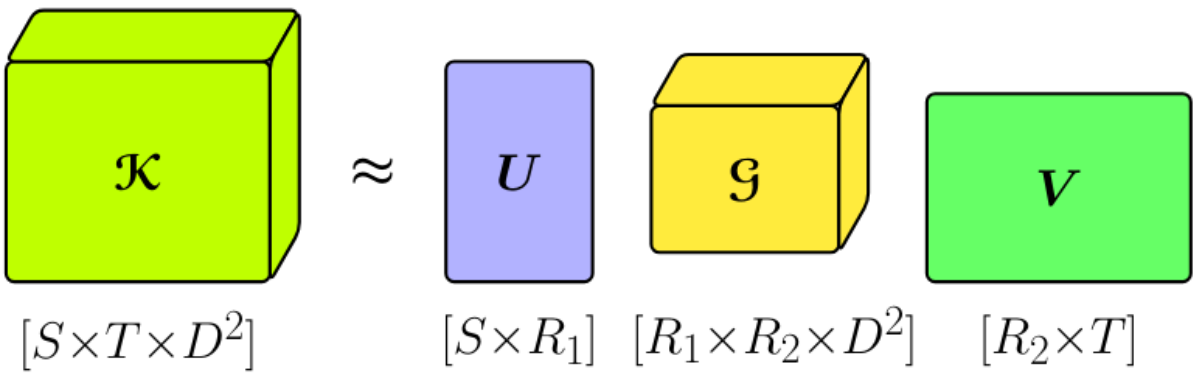
[1] EPS: accuracy drop threshold. Rank of the decomposition is chosen to maintain the drop in accuracy lower than EPS.

Canonical Polyadic Tensor Decomposition (CPD)



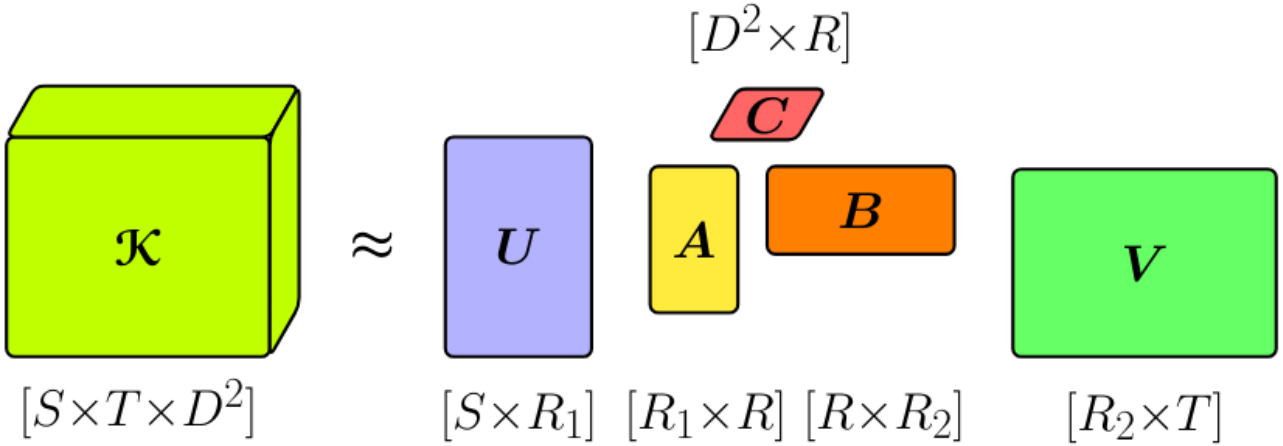
$$\Delta W_{i,j,k} = \sum_{r=1}^{R} A_{i,r} B_{j,r} C_{k,r}$$

Tucker Decomposition (TKD)



$$\Delta W_{i,j,k} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} U_{i,r_1} G_{j,r_1,r_2} V_{k,r_2}$$

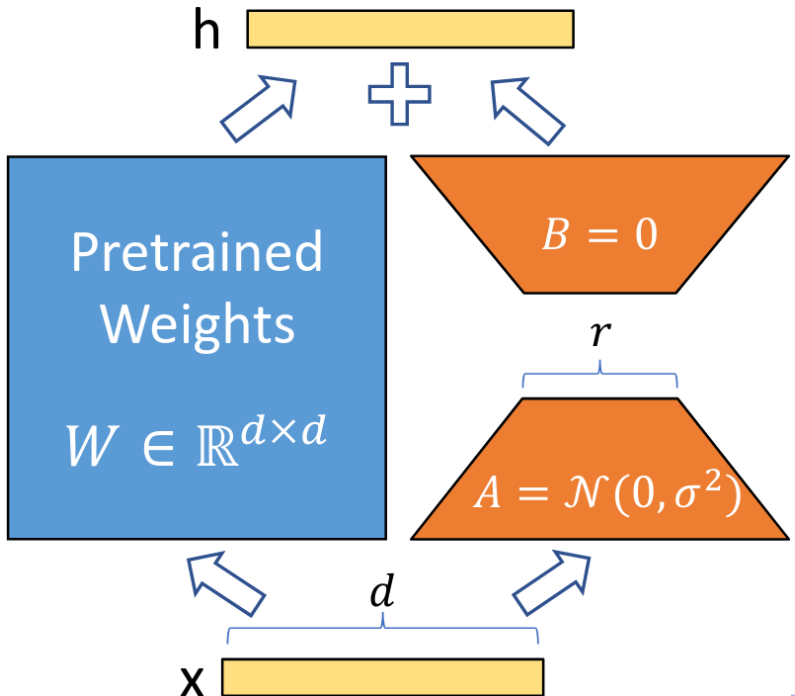
Combined TKD-CPD



$$\Delta W_{i,j,k} = \sum_{r=1}^{R} \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} U_{i,r_1} A_{r_1,r} B_{r_2,r} C_{j,r} V_{k,r_2}$$

Using LoRA directly

- But if we take a closer look at the dimensionality of kernels $(c_{out}, c_{in}, k_1 \cdot k_2)$, we see that kernel dimensions are lumped together
- If we continue this process, we can further reshape it into $(c_{out}, c_{in} \cdot k_1 \cdot k_2)$ the point where LoRA is applicable



Thank You

References I

[RBV17] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. *Learning multiple visual domains with residual adapters*. 2017. arXiv: 1705.08045 [cs.CV]. URL: https://arxiv.org/abs/1705.08045.

[Pha+20] Anh-Huy Phan et al. *Stable Low-rank Tensor Decomposition for Compression of Convolutional Neural Network*. 2020. arXiv: 2008.05441 [cs.CV]. URL: https://arxiv.org/abs/2008.05441.

[Hu+21] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685.

References II

[He+22] Junxian He et al. *Towards a Unified View of Parameter-Efficient Transfer Learning*. 2022. arXiv: 2110.04366 [cs.CL]. URL: https://arxiv.org/abs/2110.04366.

[ZRG22] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. *BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models*. 2022. arXiv: 2106.10199 [cs.LG]. URL: https://arxiv.org/abs/2106.10199.

References III

[CSL23] Zih-Ching Chen, Yu-Shun Sung, and Hung-yi Lee. *CHAPTER: Exploiting Convolutional Neural Network Adapters for Self-supervised Speech Models*. 2023. arXiv: 2212.01282 [eess.AS]. URL: https://arxiv.org/abs/2212.01282.

[Xu+23] Lingling Xu et al. *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*. 2023. arXiv: 2312.12148 [cs.CL]. URL: https://arxiv.org/abs/2312.12148.