Large Language Models

Parameter Efficient Fine Tuning I

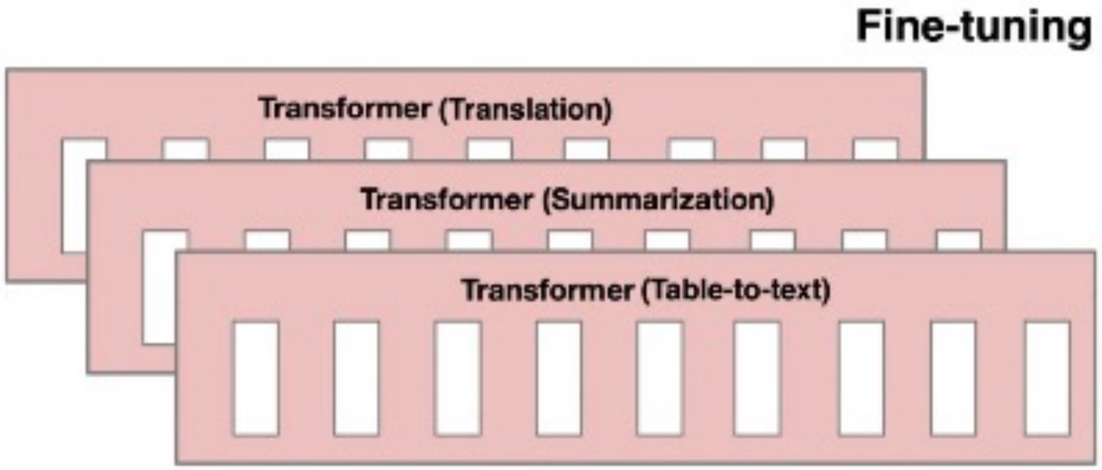
Mohammad Hossein Rohban

Fall 2023

Courtesy: Most of the slides are adopted from the course COS 597G and the paper "Parameter-Efficient Transfer Learning for NLP" by Houlsby et al 2019, and AdapterHub and AdapterFusion by J. Pfeiffer et. al

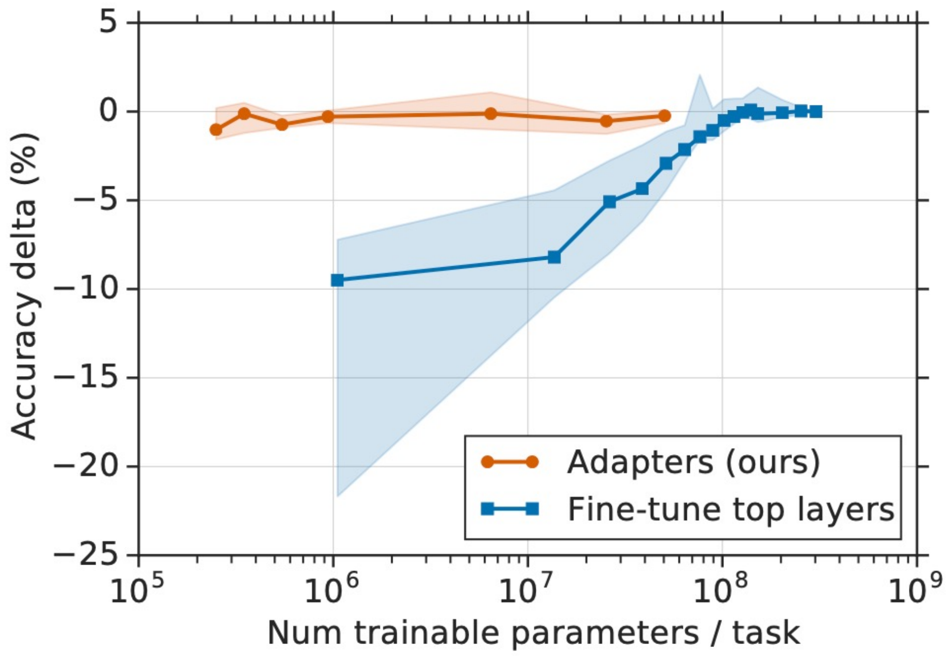
Motivation

- Large enough training set makes full fine tuning (on all weights) really good.
- But this needs enormous separate models to be stored.
  - For each task
  - For each user …
- Fine tuning (FT) on a subset of the parameters is the way to go: parameter-efficiency.



Let's discuss

- Which subset of parameter should be selected for FT?
- Last few layers?
- Turns out to be inefficient.
- A lot of weights needed to reach full FT accuracy.
- Only some layers (variable FT)?



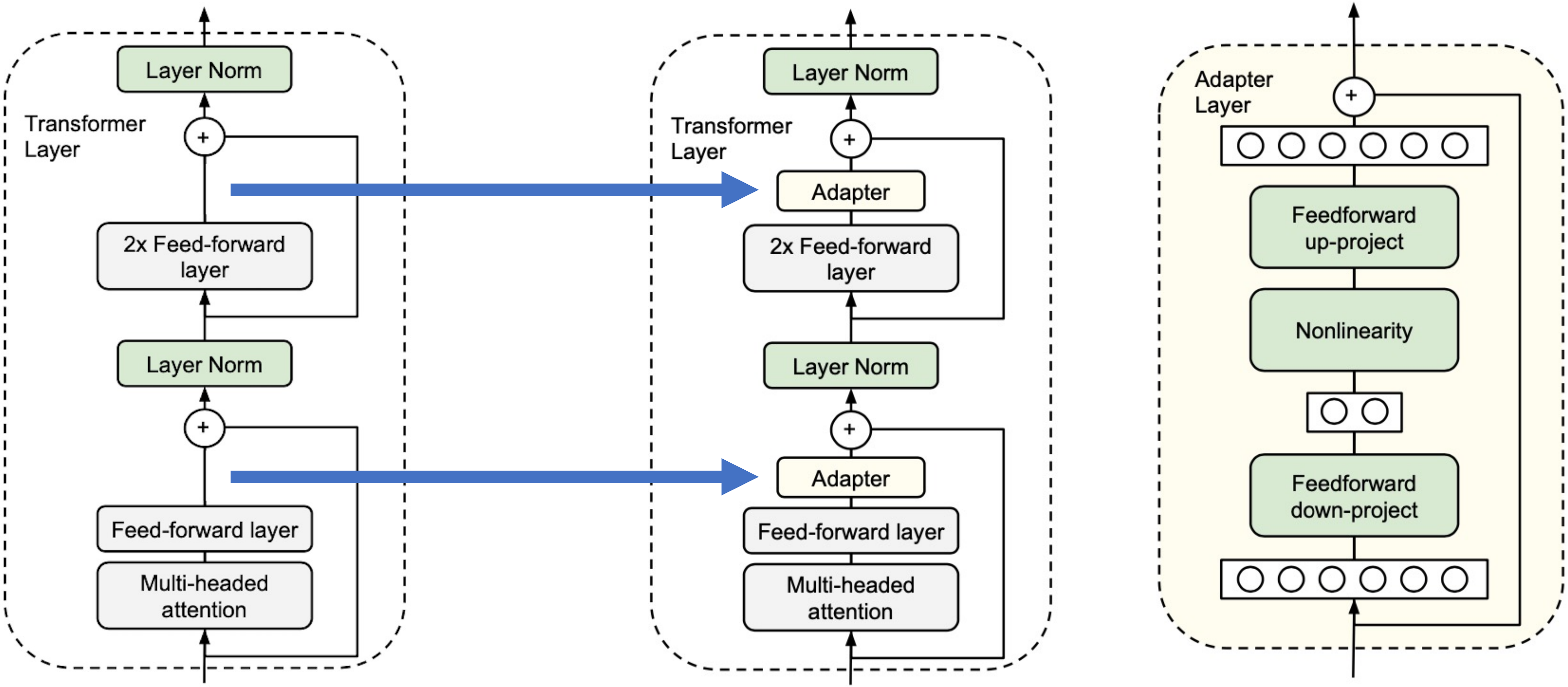
Adapters comes in handy!

- Introduced in ICML 2019.

Parameter-Efficient Transfer Learning for NLP

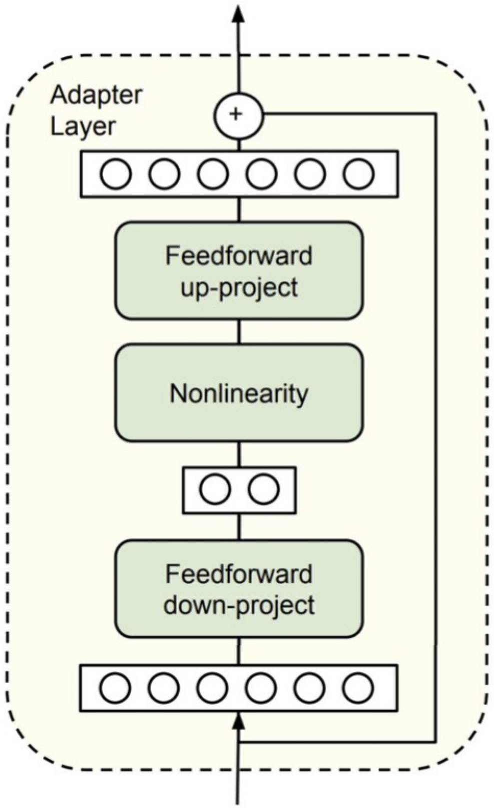
Neil Houlsby [1] **Andrei Giurgiu** [1*] **Stanisław Jastrzębski** [2*] **Bruna Morrone** [1] **Quentin de Laroussilhe** [1]
Andrea Gesmundo [1] **Mona Attariyan** [1] **Sylvain Gelly** [1]

Adapters


Layer Norm
Transformer Layer
+
2x Feed-forward layer
Layer Norm
+
Feed-forward layer
Multi-headed attention
Layer Norm
Transformer Layer
+
Adapter
2x Feed-forward layer
Layer Norm
+
Adapter
Feed-forward layer
Multi-headed attention
Adapter Layer
+
Feedforward up-project
Nonlinearity
Feedforward down-project

Adapters (cont.)

- Bottleneck architecture.
- Inserted in both sublayers; right before the skip connection.
- The adapter has a skip connection itself. Why?
- New layer normalization parameters per task as well.
- All other weights are frozen.
- Near identity initialization of the adapter. How? Why?



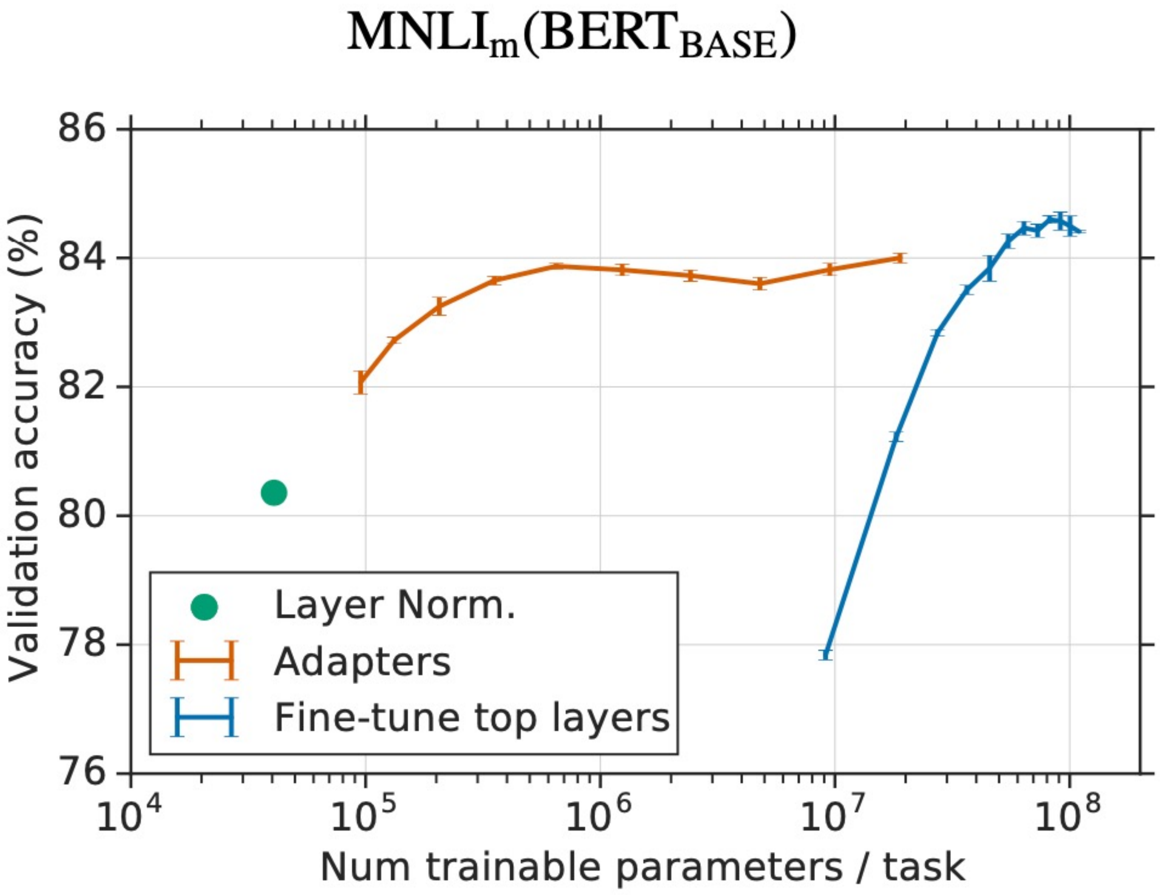
Results

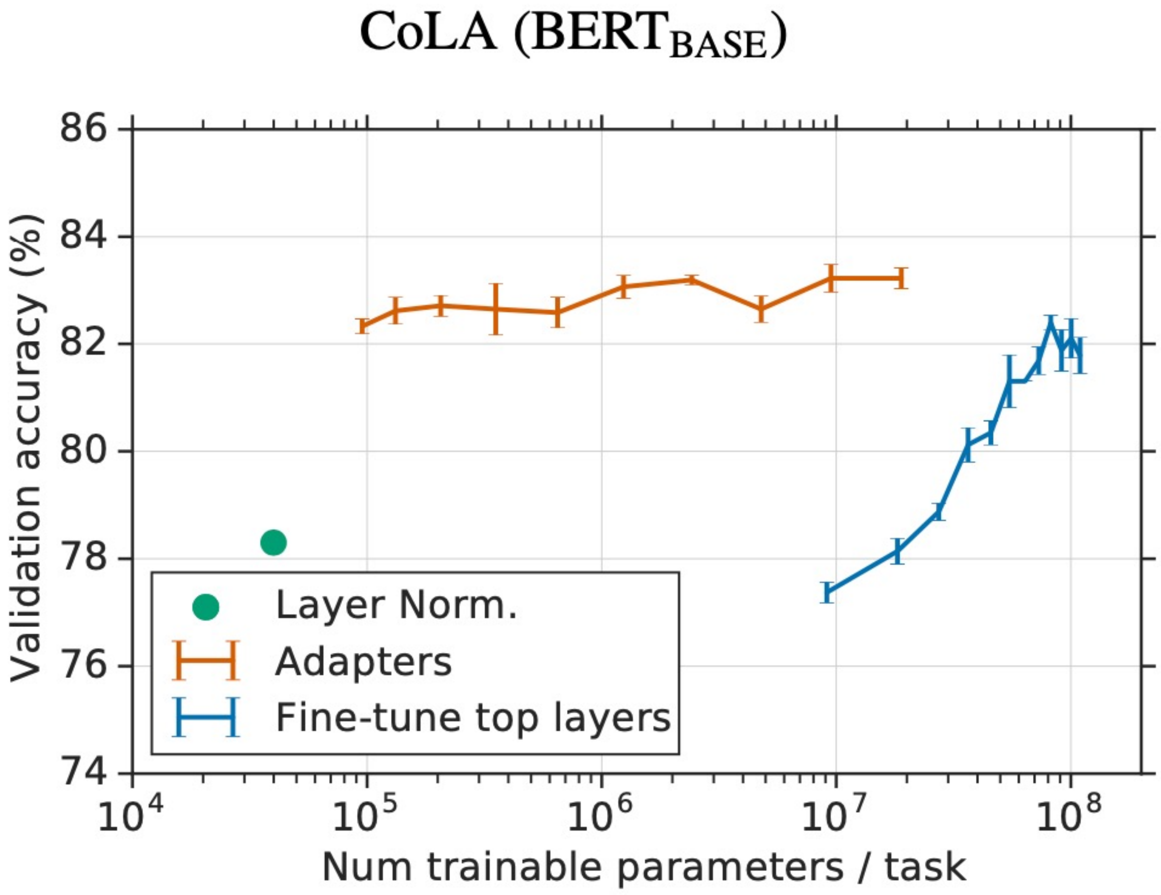
	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_{m}	MNLI_{mm}	QNLI	RTE	Total
$\text{BERT}_{\text{LARGE}}$	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

Results (cont.)

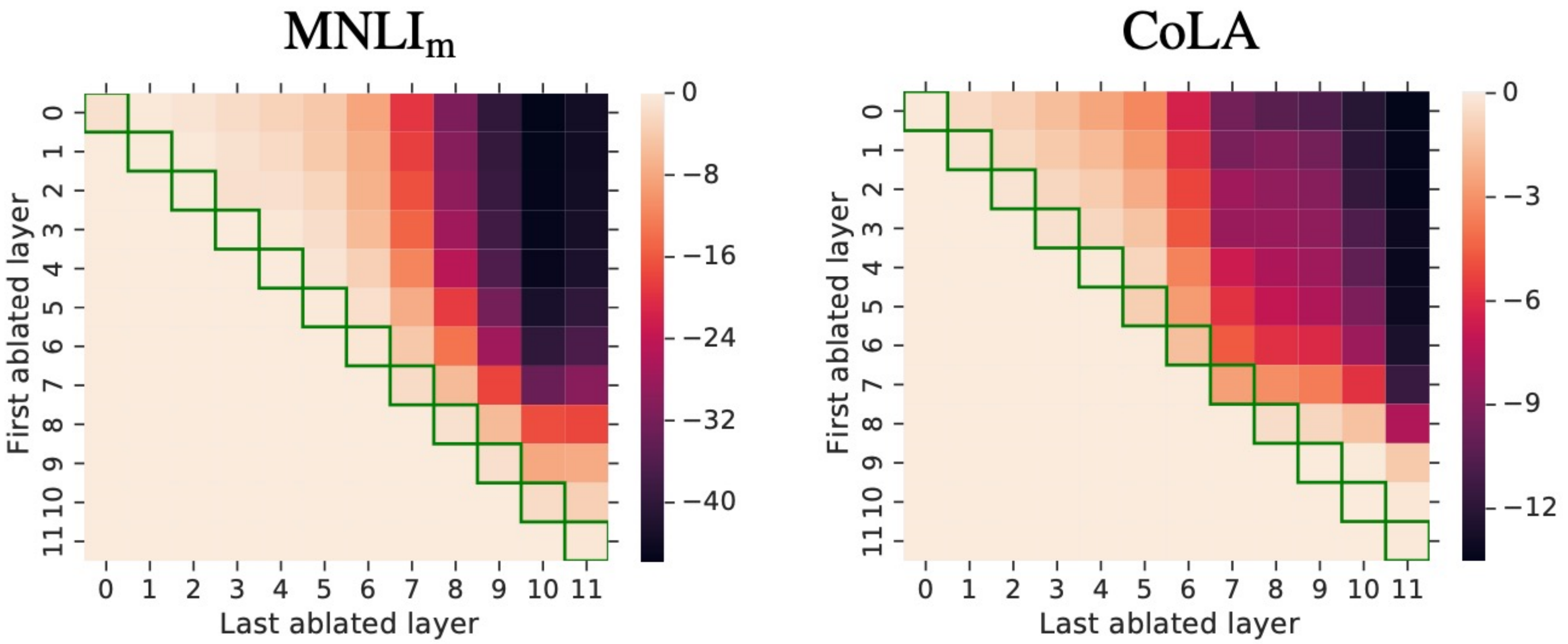
Dataset	No BERT baseline	$\text{BERT}_{\text{BASE}}$ Fine-tune	$\text{BERT}_{\text{BASE}}$ Variable FT	$\text{BERT}_{\text{BASE}}$ Adapters
20 newsgroups	91.1	92.8 ± 0.1	92.8 ± 0.1	91.7 ± 0.2
Crowdflower airline	84.5	83.6 ± 0.3	84.0 ± 0.1	84.5 ± 0.2
Crowdflower corporate messaging	91.9	92.5 ± 0.5	92.4 ± 0.6	92.9 ± 0.3
Crowdflower disasters	84.9	85.3 ± 0.4	85.3 ± 0.4	84.1 ± 0.2
Crowdflower economic news relevance	81.1	82.1 ± 0.0	78.9 ± 2.8	82.5 ± 0.3
Crowdflower emotion	36.3	38.4 ± 0.1	37.6 ± 0.2	38.7 ± 0.1
Crowdflower global warming	82.7	84.2 ± 0.4	81.9 ± 0.2	82.7 ± 0.3
Crowdflower political audience	81.0	80.9 ± 0.3	80.7 ± 0.8	79.0 ± 0.5
Crowdflower political bias	76.8	75.2 ± 0.9	76.5 ± 0.4	75.9 ± 0.3
Crowdflower political message	43.8	38.9 ± 0.6	44.9 ± 0.6	44.1 ± 0.2
Crowdflower primary emotions	33.5	36.9 ± 1.6	38.2 ± 1.0	33.9 ± 1.4
Crowdflower progressive opinion	70.6	71.6 ± 0.5	75.9 ± 1.3	71.7 ± 1.1
Crowdflower progressive stance	54.3	63.8 ± 1.0	61.5 ± 1.3	60.6 ± 1.4
Crowdflower US economic performance	75.6	75.3 ± 0.1	76.5 ± 0.4	77.3 ± 0.1
Customer complaint database	54.5	55.9 ± 0.1	56.4 ± 0.1	55.4 ± 0.1
News aggregator dataset	95.2	96.3 ± 0.0	96.5 ± 0.0	96.2 ± 0.0
SMS spam collection	98.5	99.3 ± 0.2	99.3 ± 0.2	95.1 ± 2.2
Average	72.7	73.7	74.0	73.3
Total number of params	—	$17\times$	$9.9\times$	$1.19\times$
Trained params/task	—	100%	52.9%	1.14%

Results (cont.)

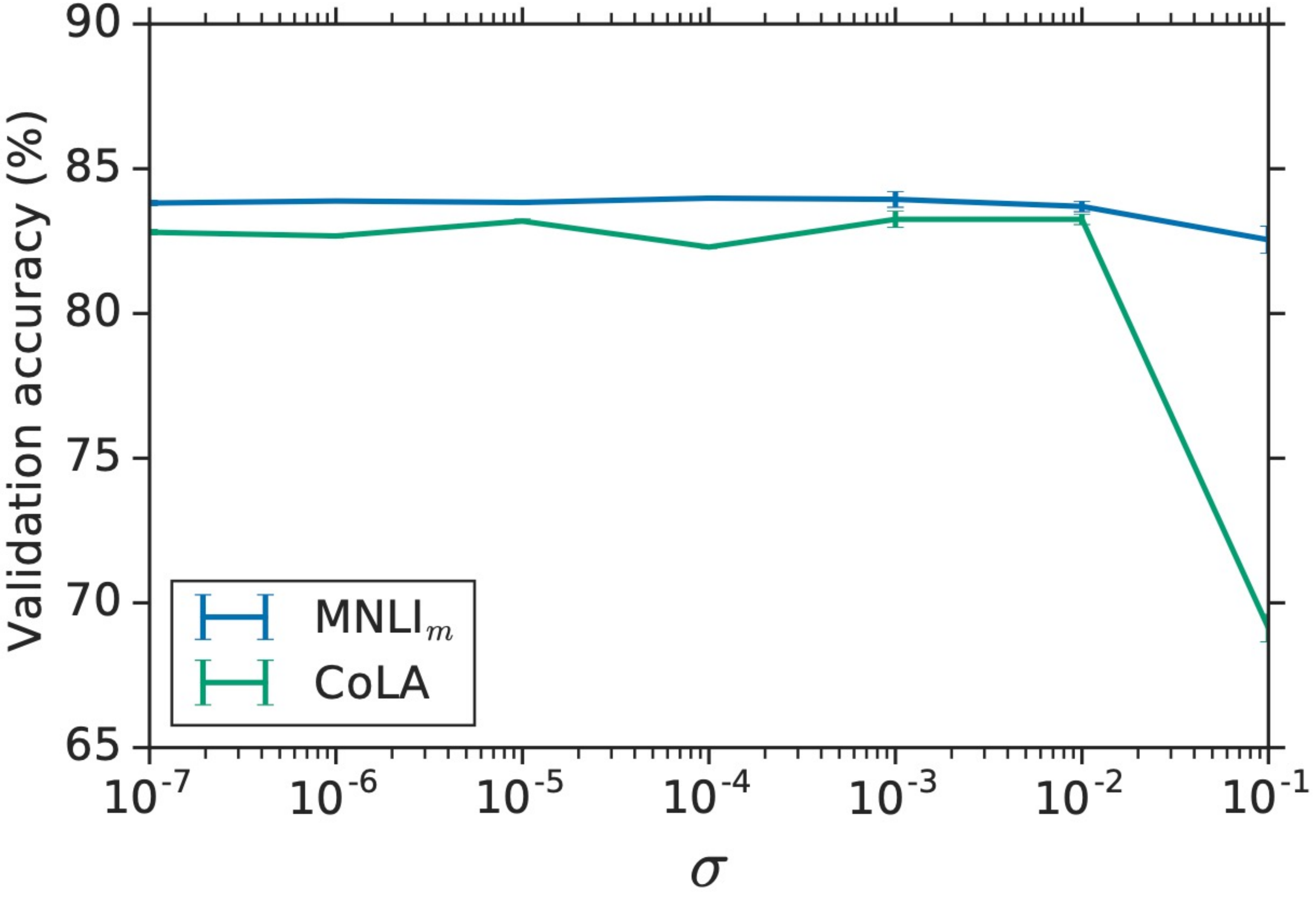




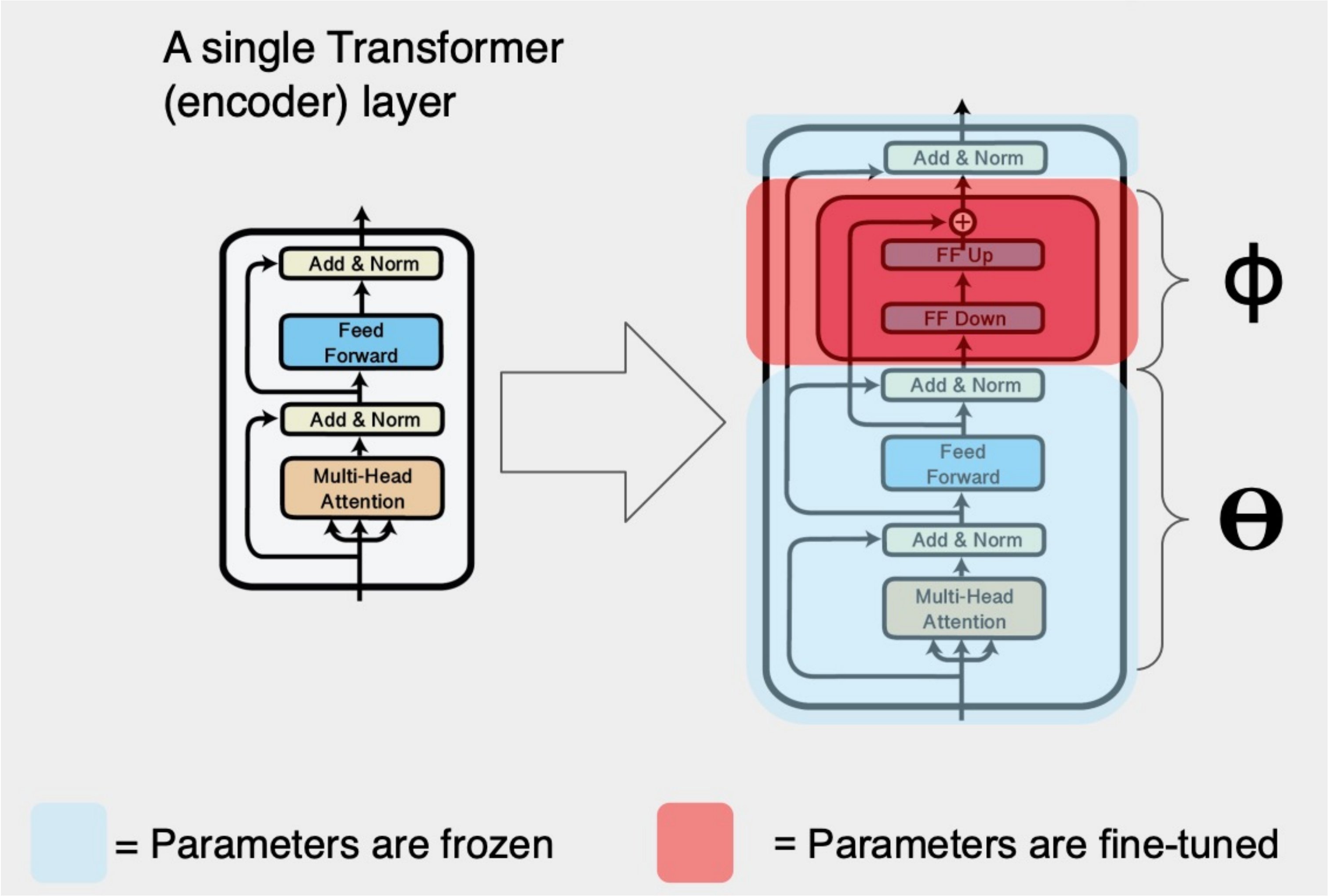
Most impactful layers?



Weight initialization impact?



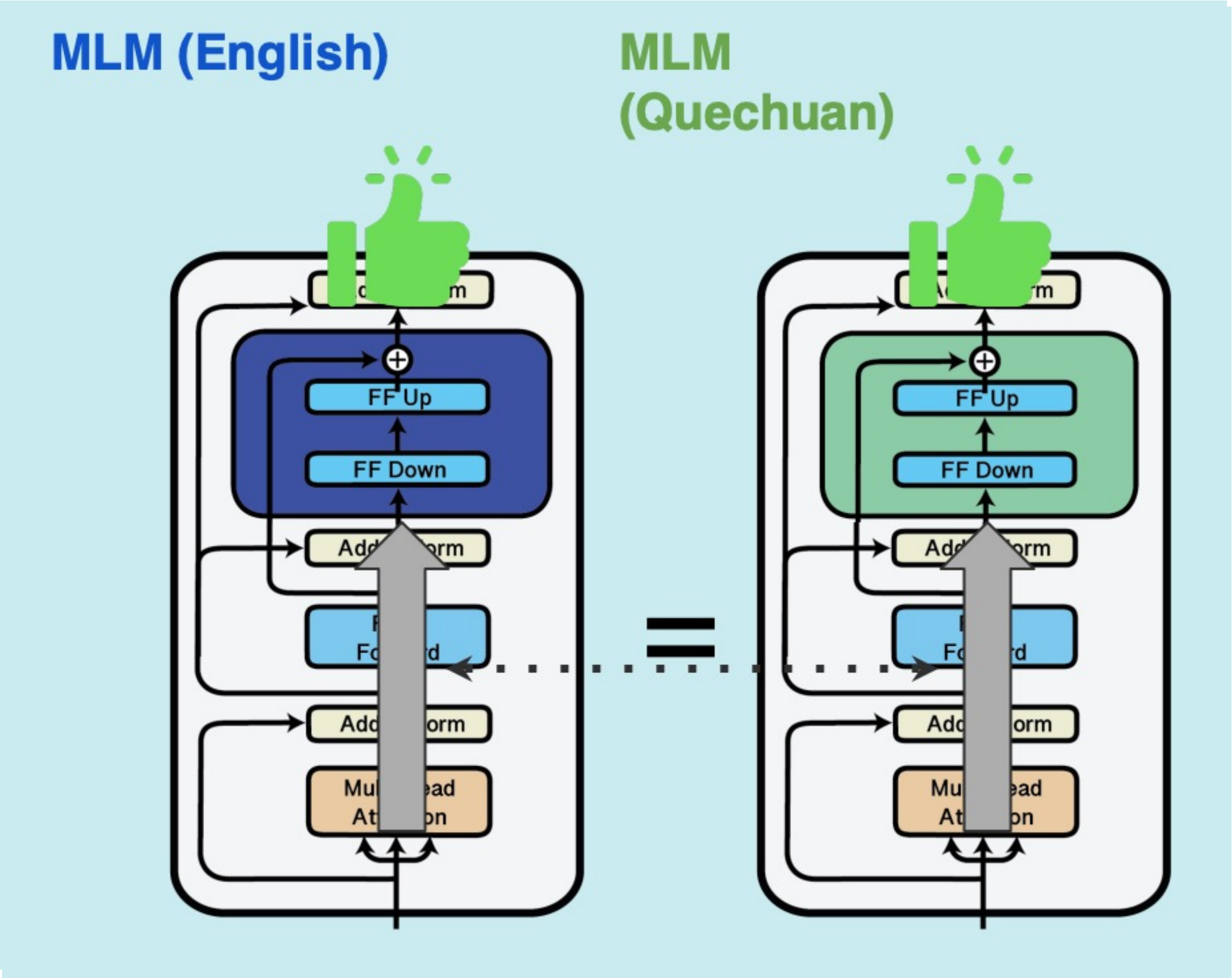
Other versions of adapters (Pfeiffer et al.)

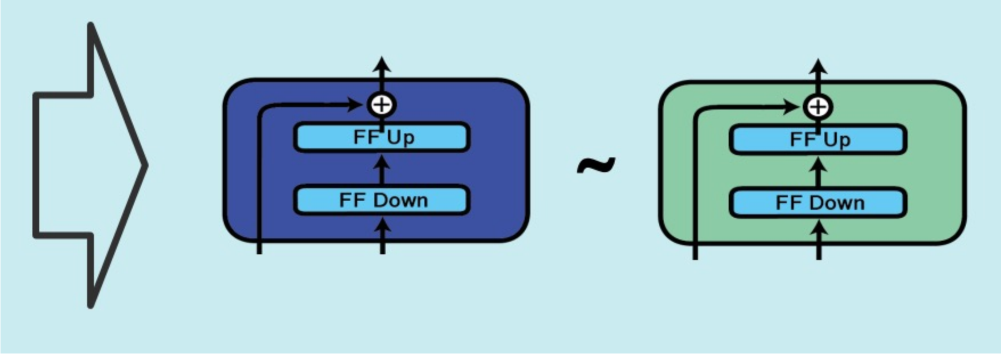


Modularity of Representation

- Surrounding parameters of an Adapter are fixed.
- What are the implications?
- At each layer the Adapter is forced to learn an output representation that is compatible with the subsequent transformer layers.

Modularity of Representation (cont.)





AdapterHub

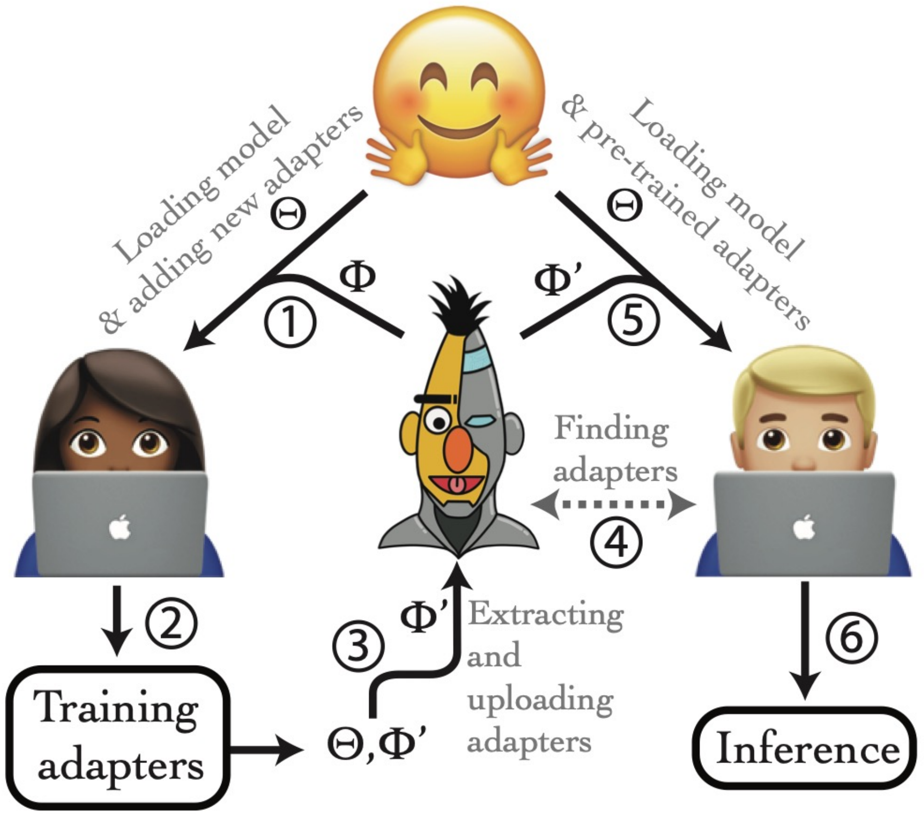


Figure 1: The AdapterHub Process graph. Adapters Φ are introduced into a pre-trained transformer Θ (step ①) and are trained (②). They can then be extracted and open-sourced (③) and visualized (④). Pre-trained adapters are downloaded on-the-fly (⑤) and stitched into a model that is used for inference (⑥).

```python
1 from transformers import AutoModelForSequenceClassification, AdapterType
2 model = AutoModelForSequenceClassification.from_pretrained("roberta-base")
3 model.add_adapter("sst-2", AdapterType.text_task, config="pfeiffer")
4 model.train_adapter(["sst-2"])
5 # Train model ...
6 model.save_adapter("adapters/text-task/sst-2/", "sst-2")
7 # Push link to zip file to AdapterHub ...
```

Figure 2: ① Adding new adapter weights Φ to pre-trained RoBERTa-Base weights Θ (line 3), and freezing Θ (line 4). ③ Extracting and storing the trained adapter weights Φ' (line 6).

```python
1 from transformers import AutoModelForSequenceClassification, AdapterType
2 model = AutoModelForSequenceClassification.from_pretrained("roberta-base")
3 model.load_adapter("sst-2", config="pfeiffer")
```

Figure 4: ⑤ After the correct adapter has been identified by the user on the explore page of AdapterHub.ml, they can load and stitch the pre-trained adapter weights Φ' into the transformer Θ (line 3).

Is it all about reducing # of parameters?

- We also seek transfer of knowledge across the tasks!
- Want the model to work on low-resources languages.
- Can Adapters help mitigate these challenges?

AdapterFusion:
Non-Destructive Task Composition for Transfer Learning

Jonas Pfeiffer[1], Aishwarya Kamath[2], Andreas Rücklé[1],
Kyunghyun Cho[2,3], Iryna Gurevych[1]
[1]Ubiquitous Knowledge Processing Lab (UKP Lab), Technical University of Darmstadt
[2]New York University [3]CIFAR Associate Fellow
pfeiffer@ukp.tu-darmstadt.de

Knowledge sharing across task

- Sequential Learning of tasks
  - Catastrophic Forgetting
- Multi-task Learning setup
  - Need to have access to all tasks at once
  - Adding a new task would be a pain in the neck
  - Overfit to low-resource tasks
  - Underfit to high-resource tasks

Problem Definition

- We are given

$$D_0 := \text{Large corpus of unlabelled text}$$
$$L_0 := \text{Masked language modelling loss}$$
$$\Theta_0 \leftarrow \underset{\Theta}{\text{argmin}}\ L_0(D_0; \Theta)$$

- And also

$$C = \{(D_1, L_1), \ldots, (D_N, L_N)\}$$

- The aim is to leverage C to improve single-task solving of $C_m = (D_m, L_m)$ with m being in {1, …, N}

AdapterFusion Method

- Step 1: Train an Adapter for each task separately (single-task adapters)

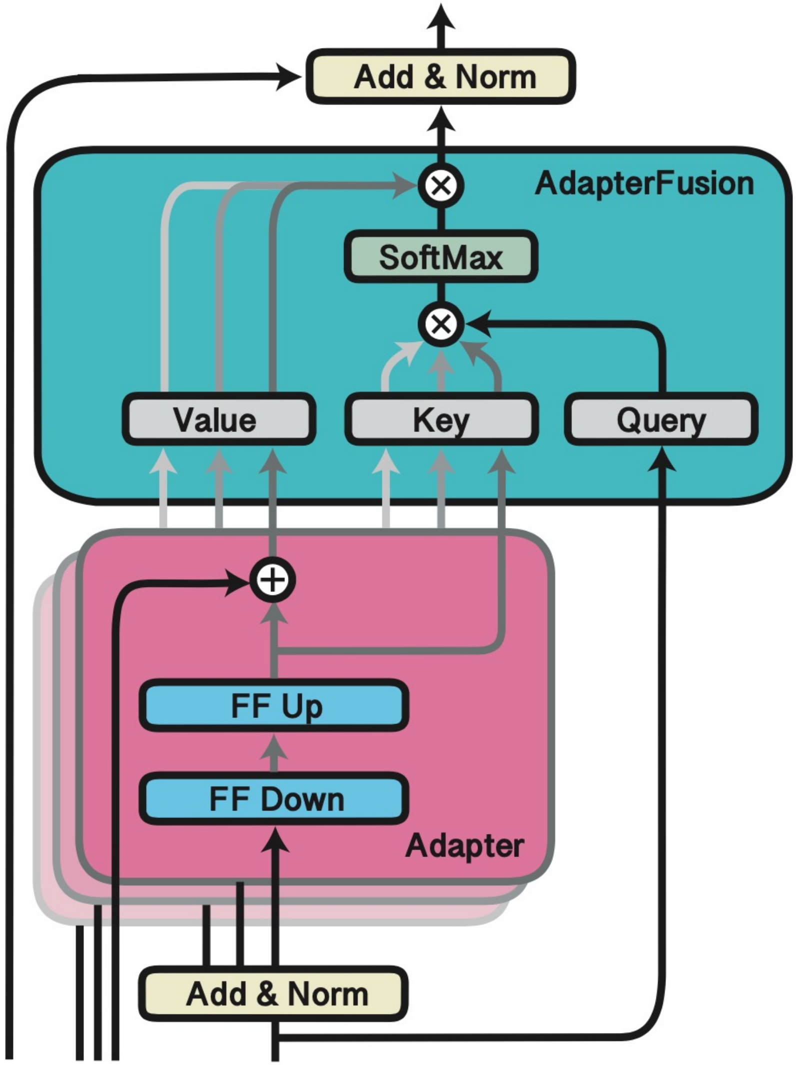
$$\Phi_n \leftarrow \underset{\Phi}{\operatorname{argmin}}\ L_n(D_n; \Theta_0, \Phi)$$

- Step 2: Fix both the parameters Θ (base transformer) and $\Phi_1, \ldots, \Phi_N$ (task adapters) introduce parameters Ψ_m to combine task adapters for the m-th task.

$$\Psi_m \leftarrow \underset{\Psi}{\operatorname{argmin}}\ L_m(D_m; \Theta, \Phi_1, \ldots, \Phi_N, \Psi)$$

AdapterFusion Method (cont.)

- Ψ_m = Key, Value and Query matrices at layer l, i.e. K_l, V_l and Q_l.
- At each layer, the output of the feed-forward sub-layer is taken as the query vector.
- The output of each adapter $z_{l,t}$ is used as input to both the value and key transformations.



AdapterFusion Method (cont.)

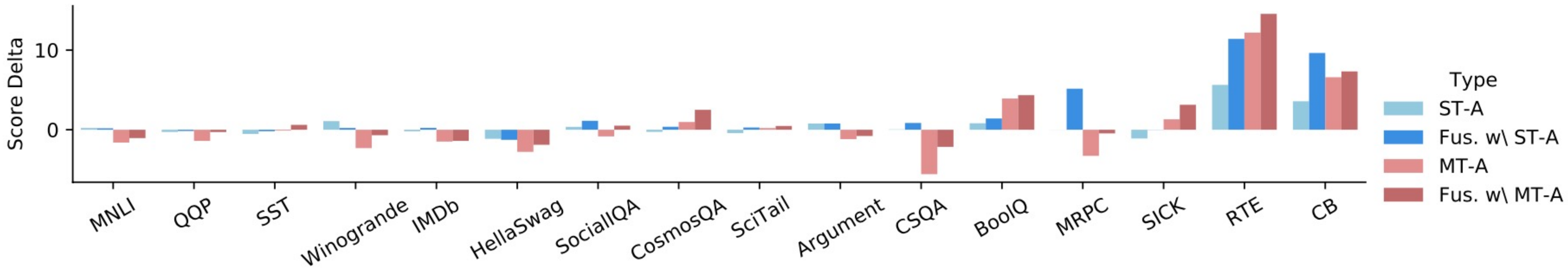
$$
\begin{aligned}
\mathbf{s}_{l,t} &= \mathrm{softmax}(\mathbf{h}_{l,t}^{\top}\mathbf{Q}_l \otimes \mathbf{z}_{l,t,n}^{\top}\mathbf{K}_l), n \in \{1, ..., N\} \\
\mathbf{z}'_{l,t,n} &= \mathbf{z}_{l,t,n}^{\top}\mathbf{V}_l, n \in \{1, ..., N\} \\
\mathbf{Z}'_{l,t} &= [\mathbf{z}'_{l,t,0}, ..., \mathbf{z}'_{l,t,N}] \\
\mathbf{o}_{l,t} &= \mathbf{s}_{l,t}^{\top}\mathbf{Z}'_{l,t}
\end{aligned}
$$

Where $\otimes$ represents the dot product and $[\cdot, \cdot]$ indicates the concatenation of vectors.

Results

Dataset	**Head**	**Full**	**ST-A**	**MT-A**	**F. w/ ST-A**	**F. w/ MT-A**	**ST-A$^{\text{Houlsby}}$**
MNLI	54.59	84.10	**84.32**	82.49 ±0.49	84.28	83.05	84.13
QQP	76.79	**90.87**	90.59	89.47 ±0.60	90.71	90.58	90.63
SST	85.17 ±0.45	92.39 ±0.22	91.85 ±0.41	92.27 ±0.71	92.20 ±0.18	**93.00** ±0.20	92.75 ±0.37
WGrande	51.92 ±0.35	60.01 ±0.08	**61.09** ±0.11	57.70 ±1.40	60.23 ±0.31	59.32 ±0.30	59.32 ±1.33
IMDB	85.05 ±0.22	**94.05** ±0.21	93.85 ±0.07	92.56 ±0.54	93.82 ±0.39	92.66 ±0.32	93.96 ±0.22
HSwag	34.17 ±0.27	**39.25** ±0.76	38.11 ±0.14	36.47 ±0.98	37.98 ±0.01	37.36 ±0.10	38.65 ±0.25
SocIQA	50.33 ±2.50	62.05 ±0.04	62.41 ±0.11	61.21 ±0.89	**63.16** ±0.24	62.56 ±0.10	62.73 ±0.53
CosQA	50.06 ±0.51	60.28 ±0.40	60.01 ±0.02	61.25 ±0.90	60.65 ±0.55	**62.78** ±0.07	61.37 ±0.35
SciTail	85.30 ±2.44	94.32 ±0.11	93.90 ±0.16	94.53 ±0.43	94.04 ±0.23	**94.79** ±0.17	94.07 ±0.39
Argument	70.61 ±0.59	76.87 ±0.32	**77.65** ±0.34	75.70 ±0.60	**77.65** ±0.21	76.08 ±0.27	77.44 ±0.62
CSQA	41.09 ±0.27	58.88 ±0.40	58.91 ±0.57	53.30 ±2.19	59.73 ±0.54	56.73 ±0.14	**60.05** ±0.36
BoolQ	63.07 ±1.27	74.84 ±0.24	75.66 ±1.25	78.76 ±0.76	76.25 ±0.19	**79.18** ±0.45	76.02 ±1.13
MRPC	71.91 ±0.13	85.14 ±0.45	85.16 ±0.52	81.86 ±0.99	**90.29** ±0.84	84.68 ±0.32	86.66 ±0.81
SICK	76.30 ±0.71	87.30 ±0.42	86.20 ±0.00	88.61 ±1.06	87.28 ±0.99	**90.43** ±0.30	86.12 ±0.54
RTE	61.37 ±1.17	65.41 ±0.90	71.04 ±1.62	77.61 ±3.21	76.82 ±1.68	**79.96** ±0.76	69.67 ±1.96
CB	68.93 ±4.82	82.49 ±2.33	86.07 ±3.87	89.09 ±1.15	**92.14** ±0.97	89.81 ±0.99	87.50 ±4.72
Mean	64.17	75.51	76.05	75.80	**77.33**	77.06	76.32

Results (cont.)



Results (cont.)

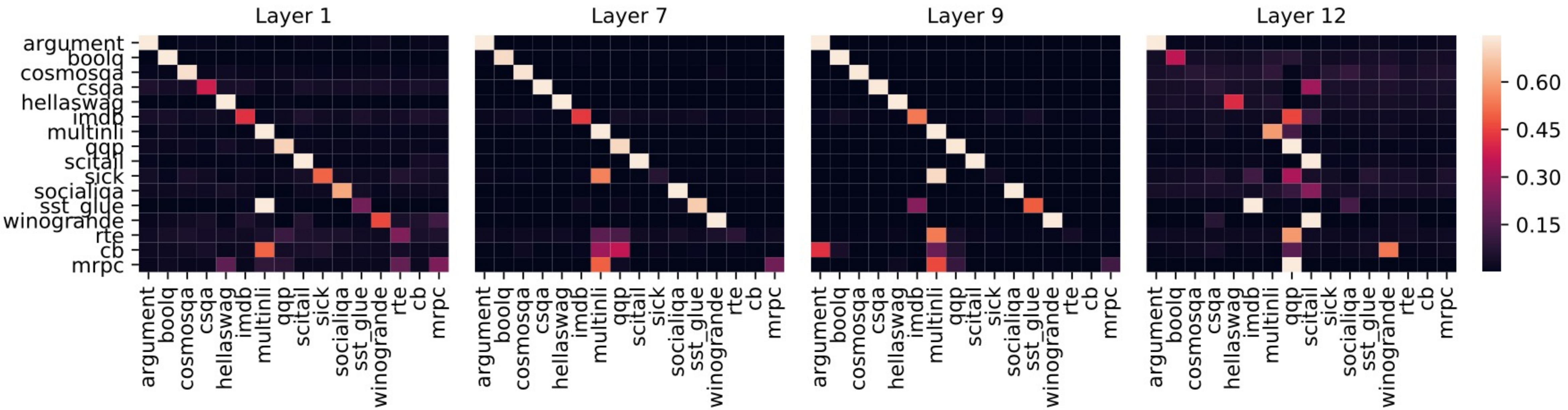


Figure 4: AdapterFusion activations of pretrained **ST-Adapters**. Rows indicate the target task m, columns indicate adapters n. We assume that the softmax activation for $\Phi_{n,l}$ is high if the information of adapter n is useful for task m. For our analysis, we calculate the softmax activation for each adapter $\Phi_{n,l}$, where $n \in \{1, \ldots, N\}$, and average over all activations within the same layer l calculated over all instances in the development set.

Surprise: Solve NER for a low-resource Lang.

- Given a general corpus of a low-resource language: Quechuan.
- No annotated dataset of NER is available in this language.
- Do a quick research to find a solution for this problem.