Parameter-Efficient Transfer Learning for NLP

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzębski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, Sylvain Gelly

April 25, 2019

Google, Jagiellonian University

Table of contents

Introduction

Problem

Problem: Consider the online setting, where tasks arrive in a stream. The goal is to build a system that performs well on all of them, but without training an entire new model for every new task.

Transfer Learning in NLP

The two most common transfer learning techniques in NLP are feature-based transfer and fine-tuning.

- **Feature-based:** Involves pre-training real-valued embeddings vectors that may be at the word, sentence, or paragraph level. The embeddings are then fed to custom downstream models.
- **Fine-tuning**: Involves copying the weights from a pre-trained network and tuning them on the downstream task. Recent work shows that fine-tuning often enjoys better performance than feature-based transfer

Both feature-based transfer and fine-tuning require a new set of weights for each task.

Adapters

Adapter tuning for NLP

Consider a function (neural network) with parameters $\boldsymbol{w}$: $\phi_{\boldsymbol{w}}(\boldsymbol{x})$.

- Feature-based transfer composes $\phi_{\boldsymbol{w}}$ with a new function, $\chi_{\boldsymbol{v}}$, to yield $\chi_{\boldsymbol{v}}(\phi_{\boldsymbol{w}}(\boldsymbol{x}))$. Only the new, task-specific, parameters, $\boldsymbol{v}$, are then trained.
- Fine-tuning involves adjusting the original parameters, $\boldsymbol{w}$, for each new task, limiting compactness.

Adapter tuning for NLP

Adapters are new modules **added between layers** of a pre-trained network.

For adapter tuning, a new function, $\psi_{w,v}(x)$, is defined, where parameters w are copied over from pre-training.

The initial parameters v_0 are set such that the new function resembles the original: $\psi_{w,v_0}(x) \approx \phi_w(x)$.

During training, only v are tuned. For deep networks, defining $\psi_{w,v}$ typically involves adding new layers to the original network, ϕ_w.

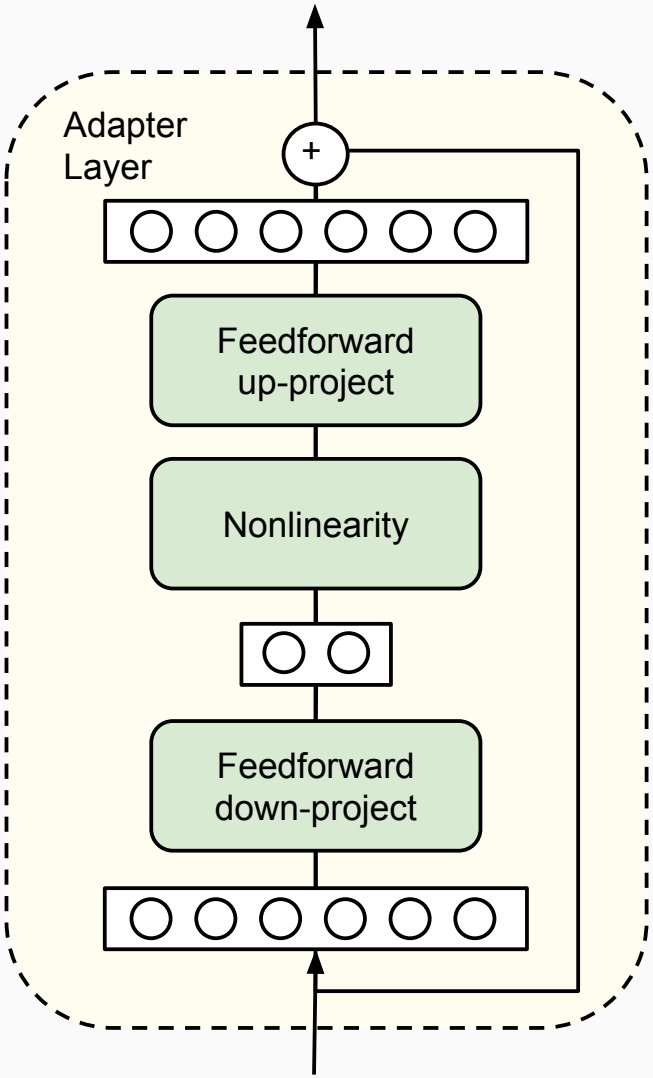
If one chooses $|v| \ll |w|$, the resulting model requires $\sim |w|$ parameters for many tasks. Since w is fixed, the model can be extended to new tasks without affecting previous ones.

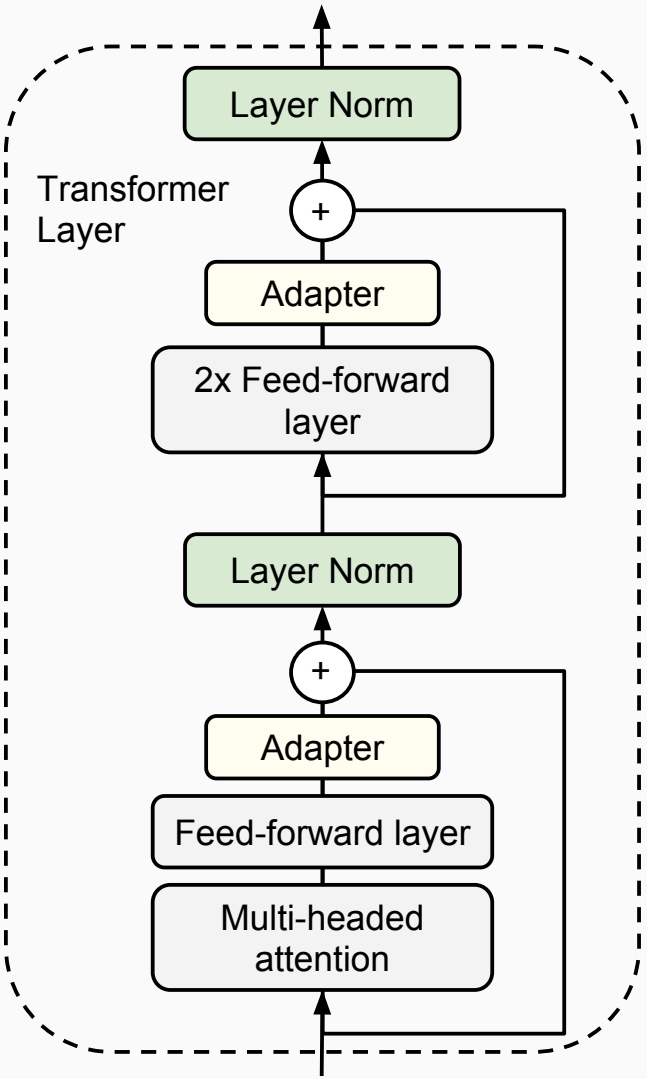
Adapter tuning for NLP

Adapter modules have two main features: a small number of parameters, and

- A *small* number of parameters: The adapter modules need to be small compared to the layers of the original network, meaning that the total model size grows relatively slowly when more tasks are added.
- A near-identity initialization: A near-identity initialization is required for stable training of the adapted model, by doig this the original network is unaffected when training starts. During training the adapters may be activated to change the distribution of activations throughout the network, but the adapter modules may also be ignored if not required. In fact, they realized that if the initialization deviates too far from the identity function, the model may fail to train.

Instantiation for Transformer Networks





Instantiation for Transformer Networks

To limit the number of parameters, they propose a **bottleneck architecture**. The adapters first project the original d-dimensional features into a smaller dimension, m, apply a nonlinearity, then project back to d dimensions. The total number of parameters added per layer, including biases, is $2md + d + m$. By setting $m \ll d$, they **limit the number of parameters added per task** (around $0.5 - 8\%$ of the parameters of the original model).

The bottleneck dimension, m, provides a simple means to **trade-off performance with parameter efficiency**. **The adapter module has a skip-connection internally**, so if the parameters of the projection layers are initialized to near-zero, the module is initialized to an approximate identity function.

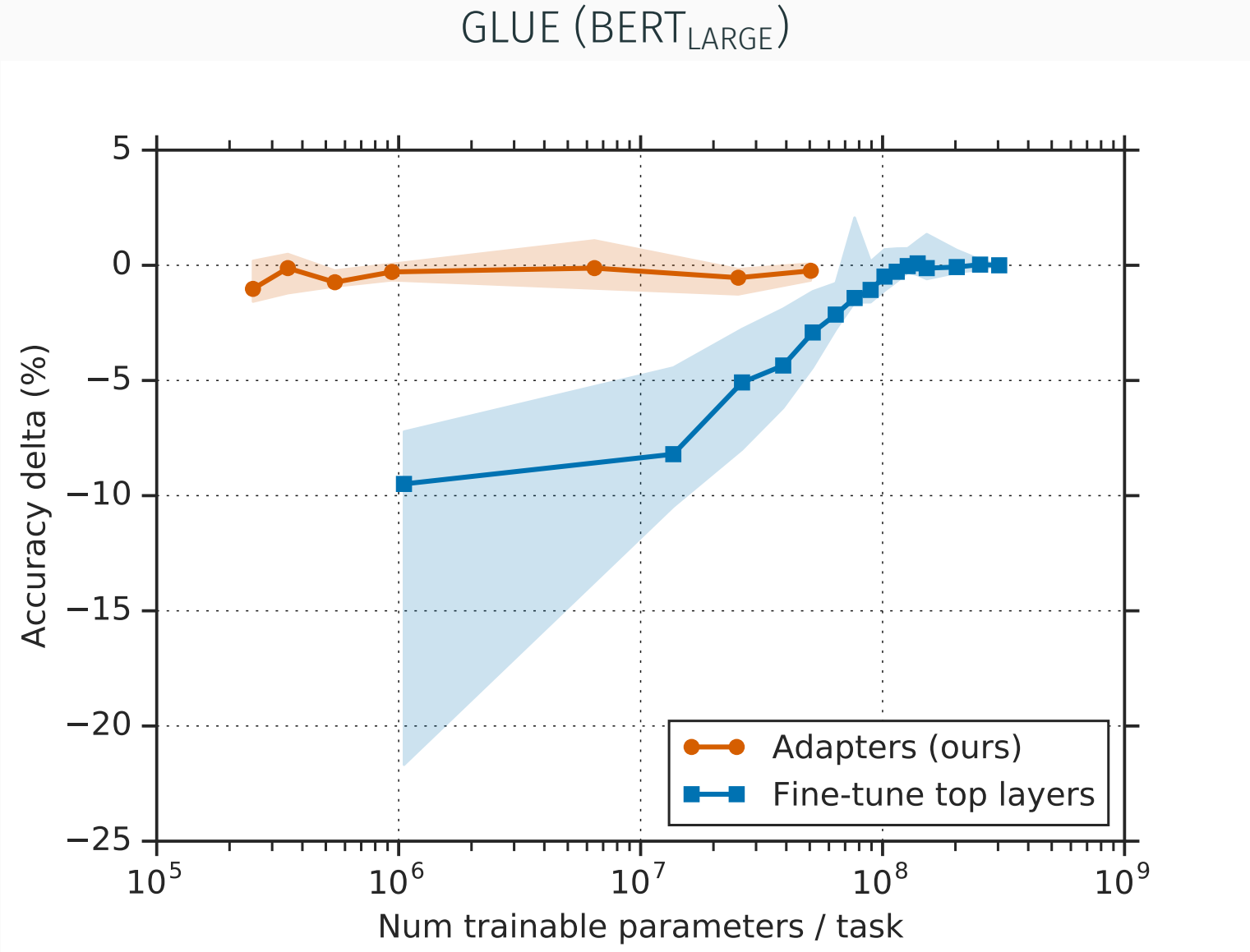
Alongside the layers in the adapter module, they also train new layer normalization parameters per task.

Experiments

GLUE benchmark

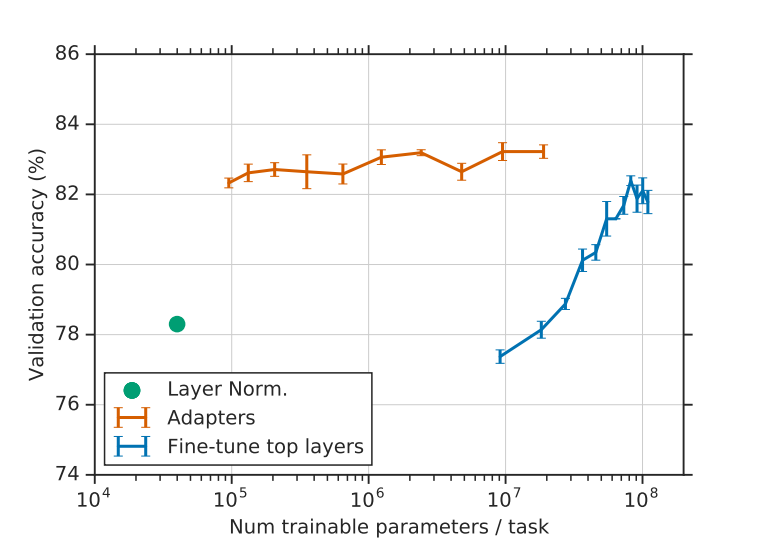
	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_{m}	MNLI_{mm}	QNLI	RTE	Total
$\text{BERT}_{\text{LARGE}}$	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6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

GLUE benchmark

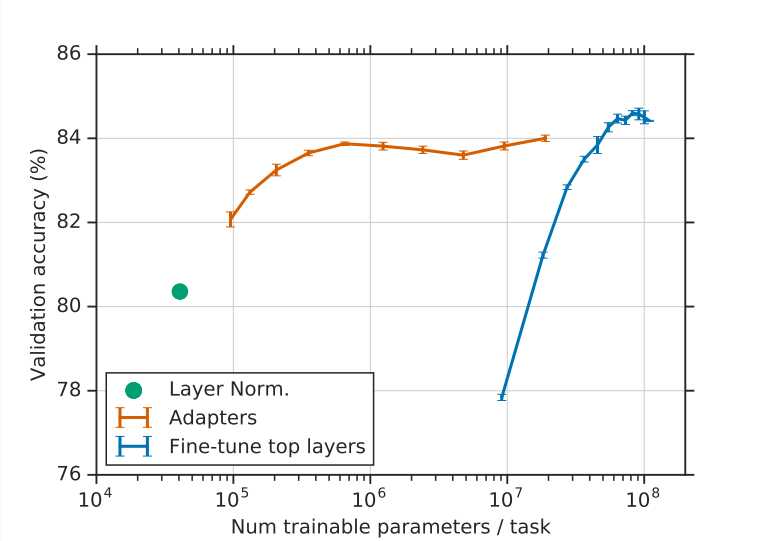


GLUE benchmark

CoLA (BERT$_{BASE}$)



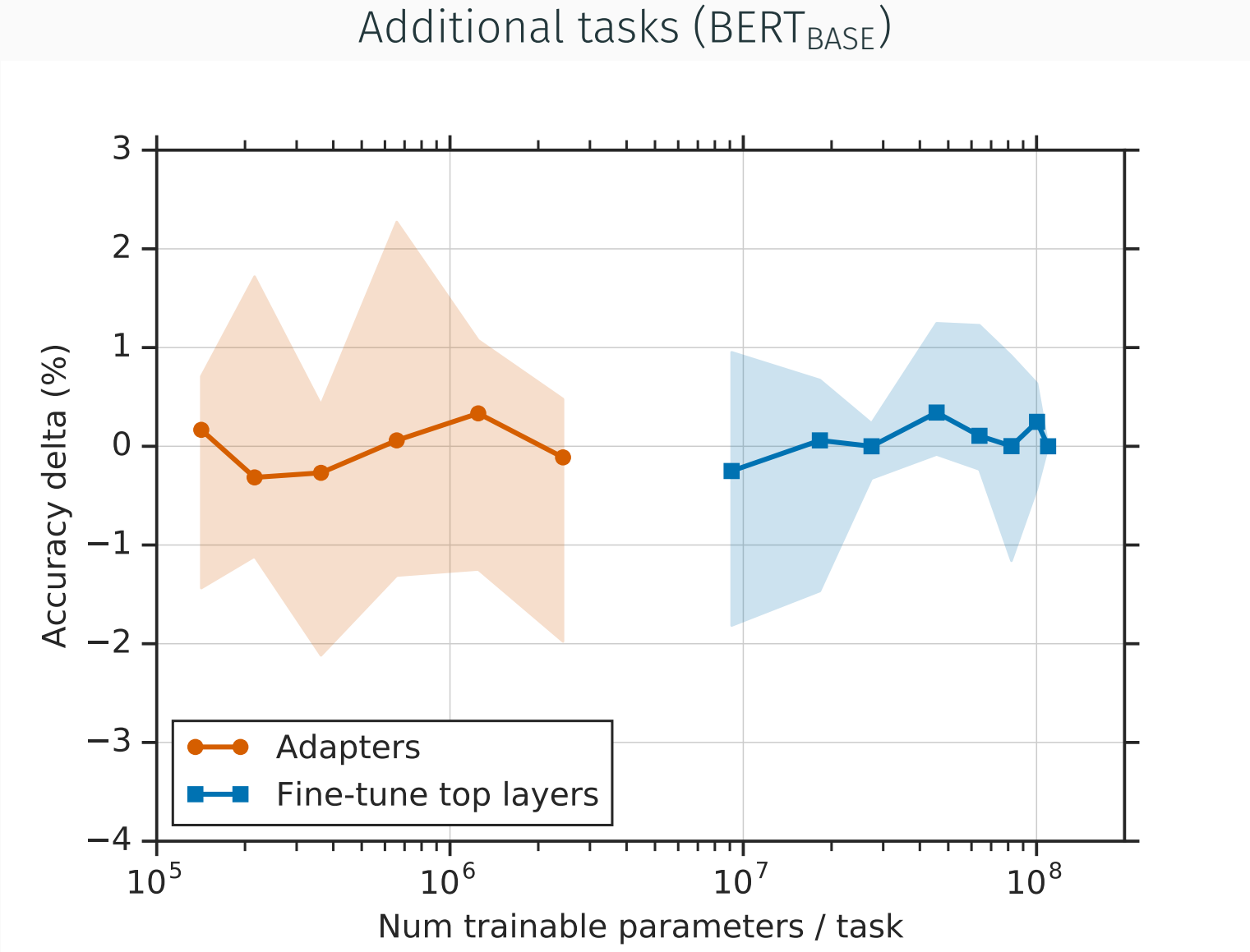
MNLI$_m$ (BERT$_{BASE}$)



Additional Tasks

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	80.8	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%

Additional Tasks



Thank you!