Adapter Modules for NLP

Chiyu Zhang

@ UBC DL-NLP Reading Group

1. Parameter-Efficient Transfer Learning for NLP

   N. Houlsby, A. Giurgiu, S. Jastrzębski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly

2. Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks

   Mahabadi, R. K., Ruder, S., Dehghani, M., & Henderson, J.

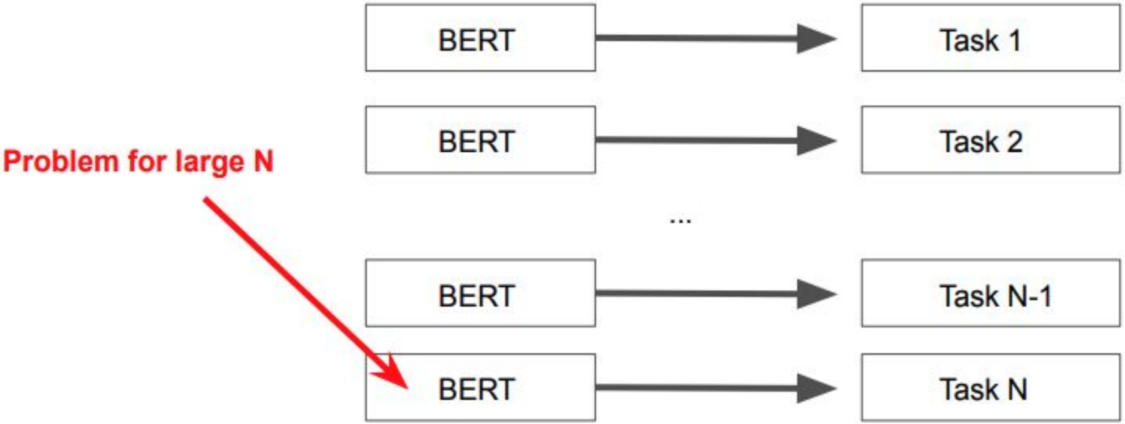
Copied some slides from the presentation of Houlsby et al. at ICML20: https://icml.cc/media/Slides/icml/2019/104(13-11-00)-13-11-20-4884-parameter-effic.pdf

Background

Transfer Learning for NLP

Ingredients:

- A large pretrained model (BERT)
- Fine-tuning

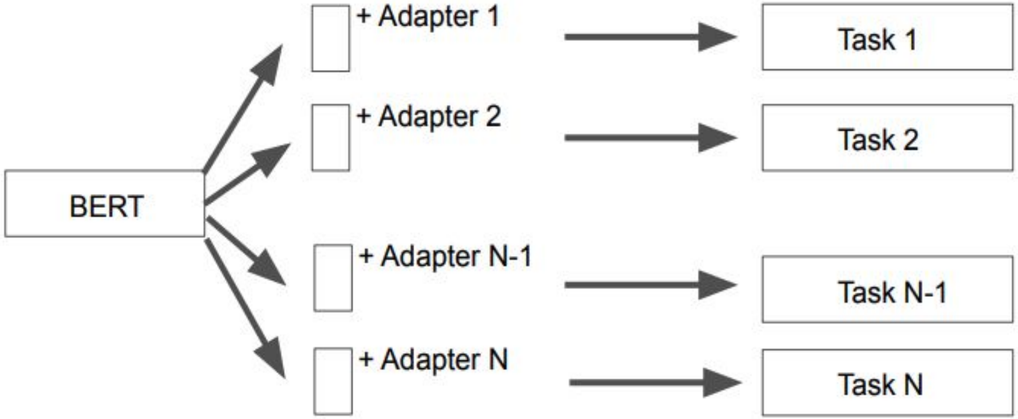


Background

Parameter-efficient Transfer Learning

Ingredients:

- A large pretrained model (BERT)
- Fine-tuning



Houlsby et al. Adapter Module

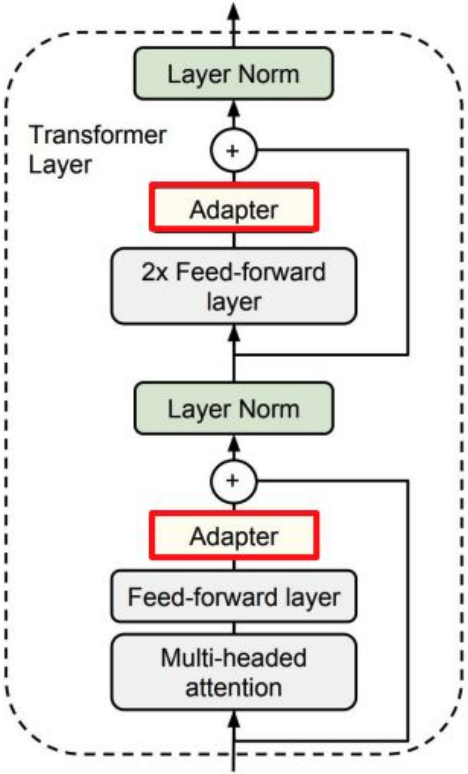
BERT + Adapters

- **Solution**: Train tiny adapter modules at each layer

(i) Good performance,

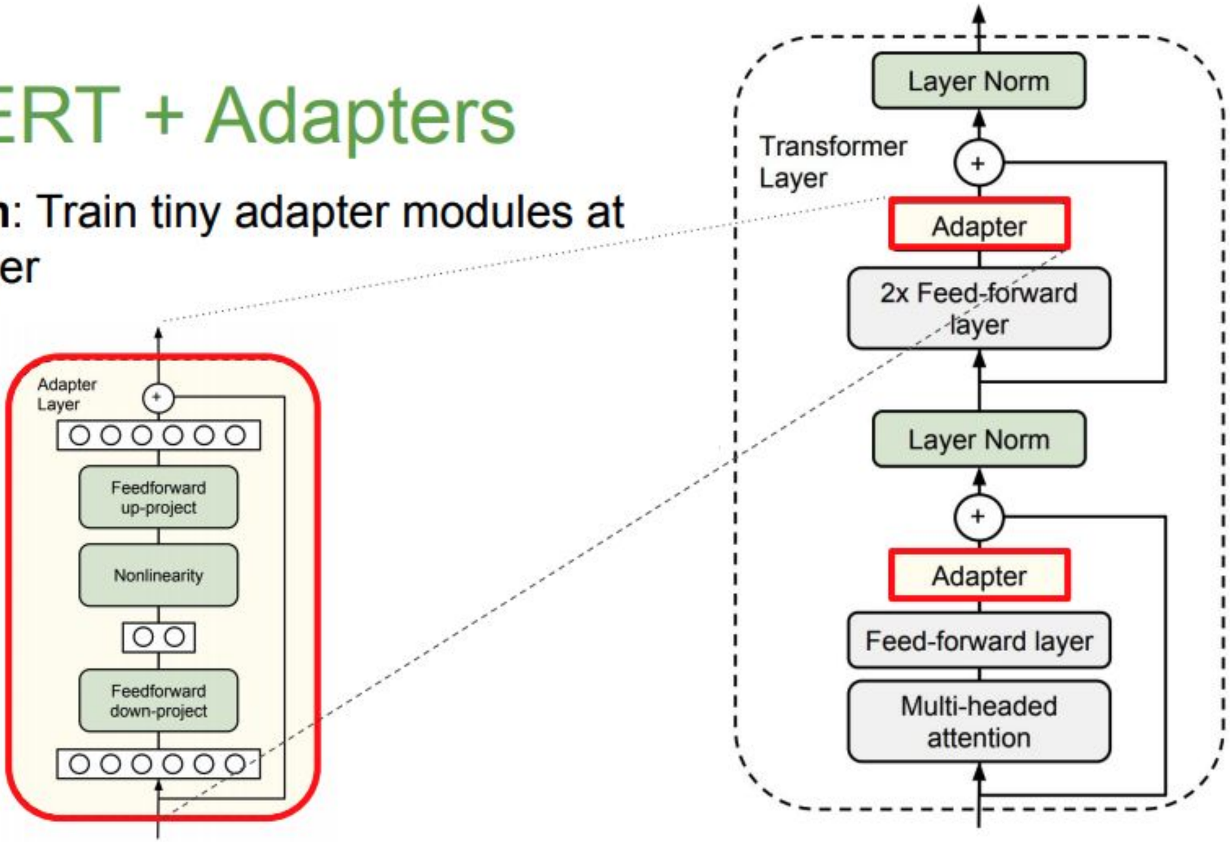
(ii) Not require simultaneous access to all datasets,

(iii) A small number of additional parameters per task.



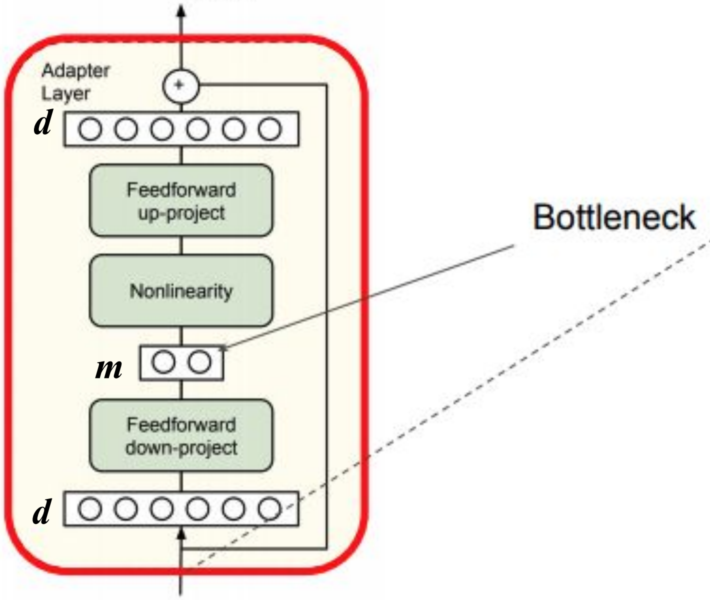
BERT + Adapters

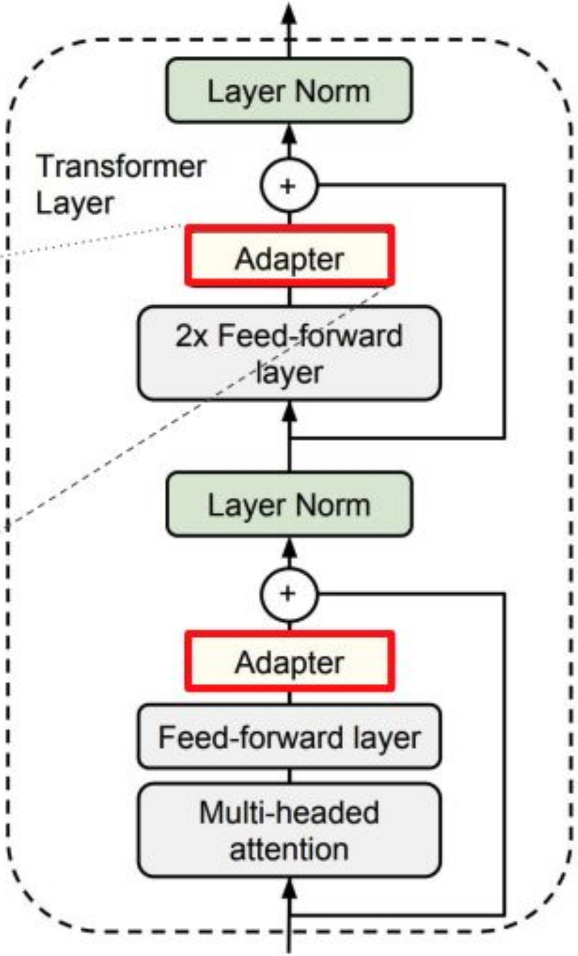
- **Solution**: Train tiny adapter modules at each layer



BERT + Adapters

- **Solution**: Train tiny adapter modules at each layer

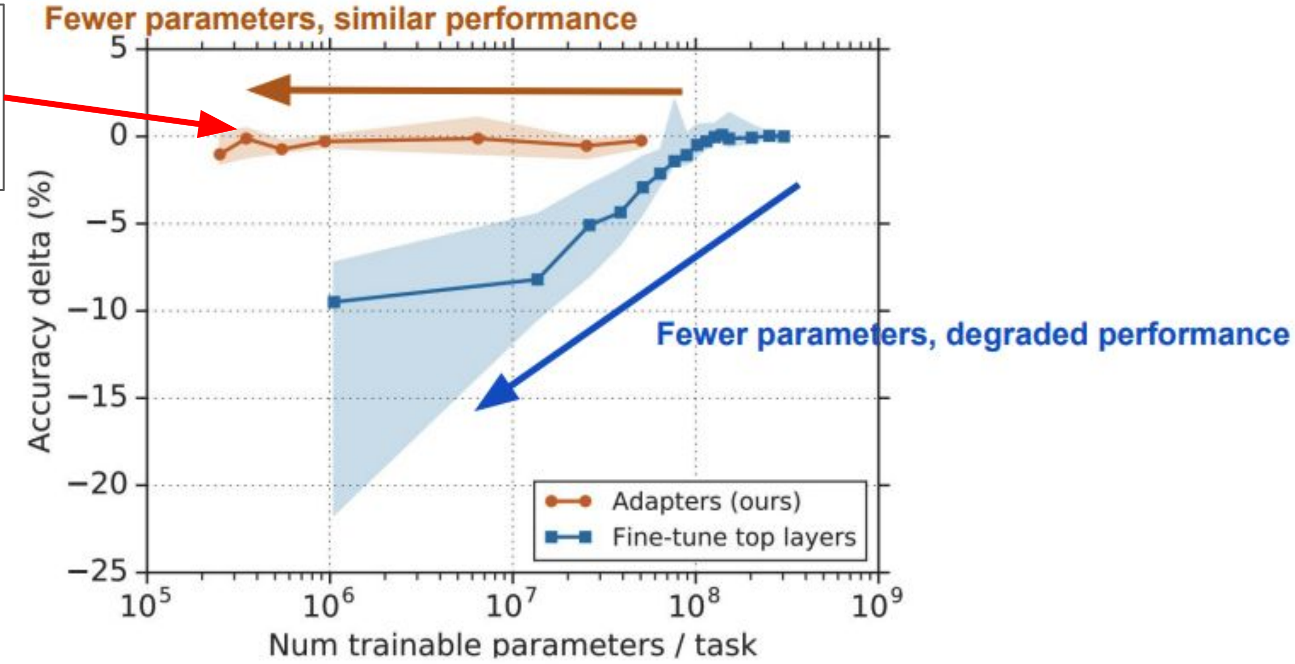




The total # of parameters added per layer: ***$2md+d+m$, $m << d$***

Results on GLUE Benchmark

0.4% accuracy drop for 96.4% reduction in the # of parameters/task



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

Table 1. Results on GLUE test sets scored using the GLUE evaluation server. MRPC and QQP are evaluated using F1 score. STS-B is evaluated using Spearman's correlation coefficient. CoLA is evaluated using Matthew's Correlation. The other tasks are evaluated using accuracy. Adapter tuning achieves comparable overall score (80.0) to full fine-tuning (80.4) using $1.3\times$ parameters in total, compared to $9\times$. Fixing the adapter size to 64 leads to a slightly decreased overall score of 79.6 and slightly smaller model.

Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks

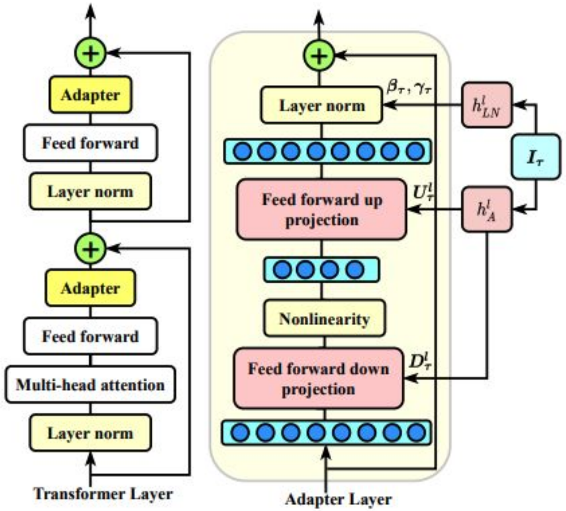
Mahabadi, R. K., Ruder, S., Dehghani, M., & Henderson, J.

Motivation

1. Share information across multiple adapters
2. Positive transfer to low-resource and related tasks.

Propose a parameter-efficient method for multi-task fine-tuning based on hypernetworks and adapter layers.

HYPERFORMER, HYPERFORMER++



Problem formulation

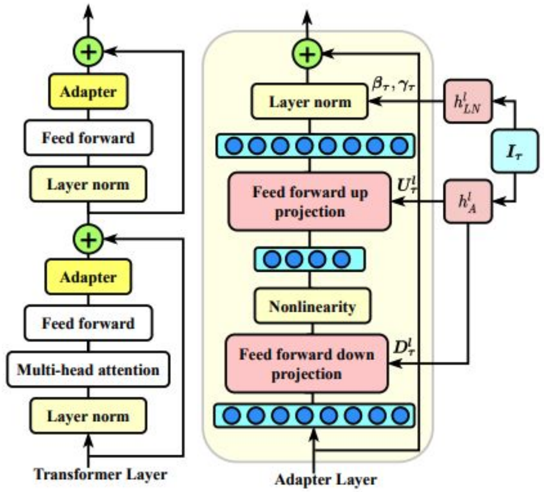
T tasks: $\mathcal{D}_\tau = \{(\boldsymbol{x}_{\boldsymbol{\tau}}^{\boldsymbol{i}}, y_\tau^i)\}_{i=1}^{N_\tau}$

A large-scale pretrained language model: $f_{\boldsymbol{\theta}}(.)$

Standard multi-task fine-tuning:

$$\mathcal{L}(\boldsymbol{\theta}, \{\mathcal{D}_\tau\}_{\tau=1}^{T}) = \sum_{\tau=1}^{T} \sum_{(\boldsymbol{x}_{\boldsymbol{\tau}}^{\boldsymbol{i}}, y_\tau^i) \in \mathcal{D}_\tau} w_\tau l\Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\boldsymbol{\tau}}^{\boldsymbol{i}}), y_\tau^i\Big),$$

Problem formulation



Proposed model:

a. A parametric task embedding: $\{\boldsymbol{I_\tau}\}_{\tau=1}^{T}$
b. Feed task embeddings to ***hypernetworks*** parameterized by $\boldsymbol{\nu}$ that generate the task-specific adapter layers
c. Insert *adapter modules* within the layers of a pretrained model.

$$\mathcal{L}(\boldsymbol{\nu},\{\boldsymbol{I_\tau}\}_{i=1}^{T},\{\mathcal{D}_\tau\}_{\tau=1}^{T})=$$

$$\sum_{\tau=1}^{T}\sum_{(\boldsymbol{x_\tau^i},y_\tau^i)\in\mathcal{D}_\tau} w_\tau l\left(\mathcal{X}_{\boldsymbol{\nu}}(\boldsymbol{x_\tau^i},\boldsymbol{\theta},\boldsymbol{I_\tau}),y_\tau^i\right),$$

HYPERFORMER

1. Pre-trained LM: T5 (an encoder-decoder Transformer)
2. Task conditional adapter layers;
3. Task conditional layer normalizations;
4. Hypernetworks that generate task-specific parameters

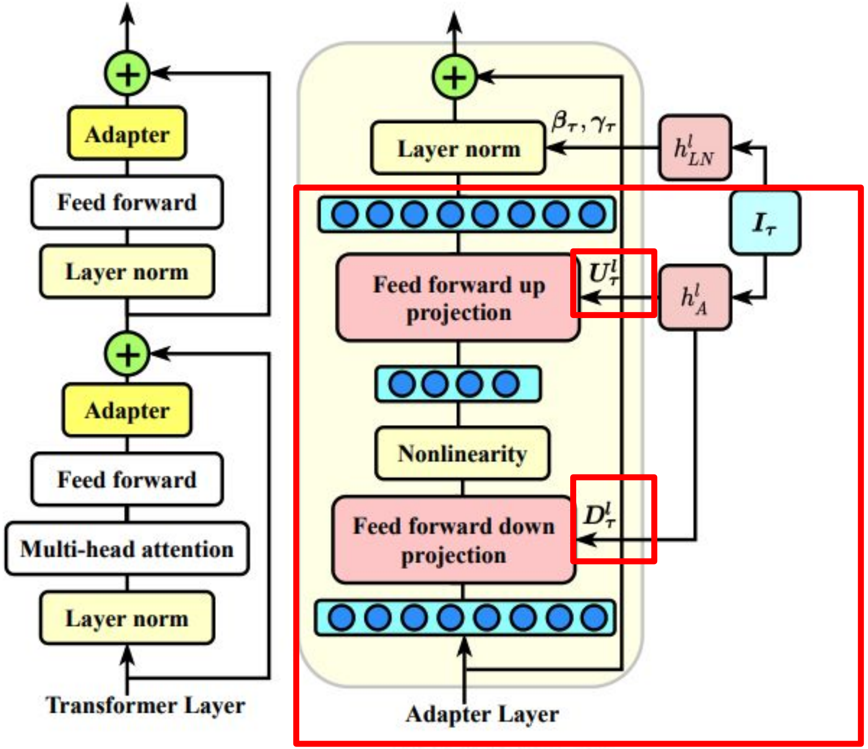
HYPERFORMER

- **Task conditional adapter layers**

Conditional adapter modules, in which we generate the adapters weights based on input task embeddings using shared hypernetworks.

h is the input dimension, and ***d*** is the bottleneck dimension for the adapter layer.

$$A^l_\tau(\boldsymbol{x}) = LN^l_\tau\Big(\boldsymbol{U}^l_\tau(\mathrm{GeLU}(\boldsymbol{D}^l_\tau(\boldsymbol{x})))\Big) + \boldsymbol{x},$$



Adapter weights $(\boldsymbol{U}^l_\tau,\ \boldsymbol{D}^l_\tau)$ through a hypernetwork

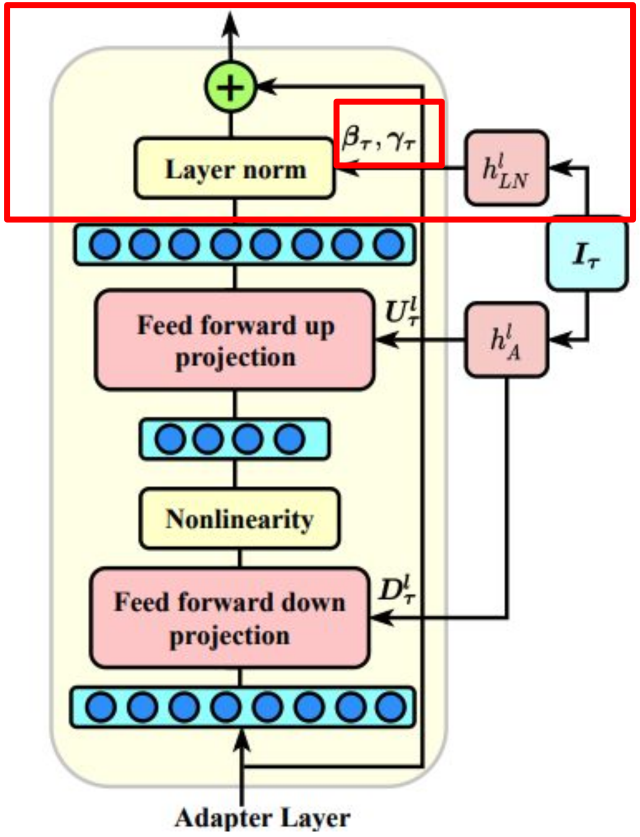
HYPERFORMER

- **Task conditional layer normalizations**

$$LN^l_\tau(\boldsymbol{x}^i_\tau) = \gamma^l_\tau \odot \frac{\boldsymbol{x}^i_\tau - \boldsymbol{\mu}_\tau}{\boldsymbol{\sigma}_\tau} + \boldsymbol{\beta}^l_\tau,$$

γ^l_τ and $\boldsymbol{\beta}^l_\tau$ are learnable parameters with the same dimension as $\boldsymbol{x}^i_\tau$. Values of $\boldsymbol{\mu}_\tau$ and $\boldsymbol{\sigma}_\tau$ show the mean and standard deviation of training data for the τ-th task.

γ^l_τ, β^l_τ via a hypernetwork as a function of task embeddings



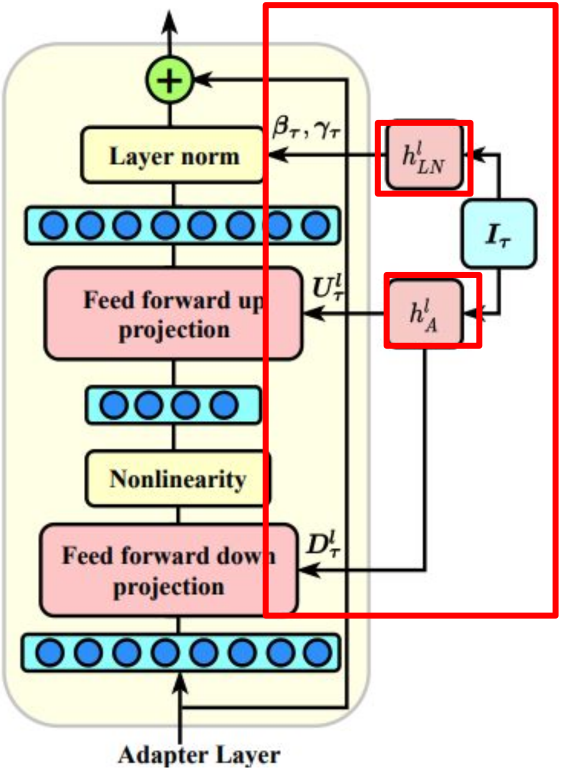
HYPERFORMER

- **Task Conditioned Hypernetworks**

Share information across adapter modules

Generate the parameters of task conditional adapter layers and layer normalization using hypernetworks.

a. Learned task embedding $\boldsymbol{I}_{\tau} = h_I(\boldsymbol{z}_{\tau}),$
b. Removing task prefixes in T5, use task embedding.
c. Task conditioned hypernetworks: **simple linear layers**



$$(\gamma^l_\tau, \beta^l_\tau) := h^l_{LN}(\boldsymbol{I}_\tau) = \left(\boldsymbol{W}^{\gamma^l}, \boldsymbol{W}^{\beta^l}\right)\boldsymbol{I}_\tau, \qquad \boldsymbol{W}^{\gamma^l} \in \mathbb{R}^{h \times t} \text{ and } \boldsymbol{W}^{\beta^l} \in \mathbb{R}^{h \times t}$$

$$(\boldsymbol{U}^l_\tau, \boldsymbol{D}^l_\tau) := h^l_A(\boldsymbol{I}_\tau) = \left(\boldsymbol{W}^{U^l}, \boldsymbol{W}^{D^l}\right)\boldsymbol{I}_\tau, \quad \boldsymbol{W}^{U^l} \in \mathbb{R}^{(d \times h) \times t} \text{ and } \boldsymbol{W}^{D^l} \in \mathbb{R}^{(h \times d) \times t}$$

HYPERFORMER++

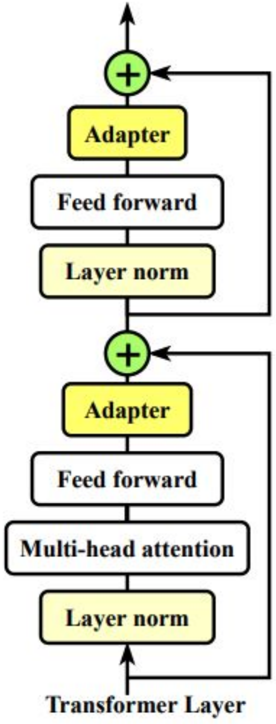
Share hypernetworks across transformer layers

a. Layer id embeddings $\mathcal{I} = \{\boldsymbol{l_i}\}_{i=1}^{L}$

b. Adapter position embeddings $\mathcal{P} = \{\boldsymbol{p_j}\}_{j=1}^{2}$

c. Learned task embedding:

$$\boldsymbol{I_\tau} = h'_I(\boldsymbol{z_\tau}, \boldsymbol{l_i}, \boldsymbol{p_j}),$$



The hypernetwork is able to produce distinct weights for **each task**, **adapter position**, and **layer of a transformer**.

Experiments

Datasets:

GLUE:

multiple tasks of paraphrase detection (MRPC, QQP), sentiment classification (SST-2), natural language inference (MNLI, RTE, QNLI), semantic textual similarity benchmark (STS-B), and linguistic acceptability (CoLA).

Results

Model	#Total params	#Trained params / per task	CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Avg
					Single-Task Training						
T5$_{\text{SMALL}}$	8.0×	100%	**46.81**	**90.47**	**86.21/90.67**	**91.02/87.96**	**89.11/88.70**	**82.09**	**90.21**	**59.42**	**82.06**
Adapters$_{\text{SMALL}}$ ❣	1+8×0.01	**0.74%**	40.12	89.44	85.22/89.29	90.04/86.68	83.93/83.62	81.58	89.11	55.80	79.53
T5$_{\text{BASE}}$	8.0×	100%	54.85	92.19	**88.18/91.61**	**91.46/88.61**	**89.55/89.41**	**86.49**	91.60	67.39	84.67
Adapters$_{\text{BASE}}$ ❣	1+8×0.01	**0.87%**	**59.49**	**93.46**	88.18/91.55	90.94/88.01	87.44/87.18	86.38	**92.26**	**68.84**	**84.88**
					Multi-Task Training						
T5$_{\text{SMALL}}$ ♠	1.0×	12.5%	50.67	**91.39**	84.73/88.89	**89.53/86.31**	**88.70/88.27**	81.04	89.67	59.42	81.69
Adapters†$_{\text{SMALL}}$	1.05×	0.68%	39.87	90.01	88.67/91.81	88.51/84.77	88.15/87.89	79.95	89.60	60.14	80.85
HyperFormer$_{\text{SMALL}}$	1.45×	5.80%	47.64	**91.39**	**90.15/92.96**	88.68/85.08	87.49/86.96	**81.24**	**90.39**	65.22	82.47
HyperFormer++$_{\text{SMALL}}$	1.04×	**0.50%**	**53.96**	90.59	84.24/88.81	88.44/84.46	87.73/87.26	80.69	**90.39**	**71.01**	**82.51**
T5$_{\text{BASE}}$ ♠	1.0×	12.5%	54.88	92.54	90.15/93.01	**91.13/88.07**	88.84/88.53	85.66	92.04	75.36	85.47
Adapters†$_{\text{BASE}}$	1.07×	0.82%	61.53	93.00	90.15/92.91	90.47/87.26	89.86/89.44	86.09	**93.17**	70.29	85.83
HyperFormer$_{\text{BASE}}$	1.54×	6.86%	61.32	93.80	**90.64/93.33**	90.13/87.18	89.55/89.03	**86.33**	92.79	**78.26**	**86.58**
HyperFormer++$_{\text{BASE}}$	1.02×	**0.29%**	**63.73**	**94.03**	89.66/92.63	90.28/87.20	**90.00/89.66**	85.74	93.02	75.36	86.48

Few-shot Learning

Dataset	# Samples	T5$_{BASE}$	Adapters†$_{BASE}$	HyperFormer++$_{BASE}$
Natural Language Inference				
SciTail	4	79.60$_{\pm 3.3}$	79.54$_{\pm 2.8}$	**82.00**$_{\pm 4.9}$
	16	80.03$_{\pm 2.3}$	83.25$_{\pm 1.7}$	**86.55**$_{\pm 1.4}$
	32	81.97$_{\pm 1.3}$	85.06$_{\pm 1.1}$	**85.85**$_{\pm 1.4}$
	100	84.04$_{\pm 0.7}$	88.22$_{\pm 1.3}$	**88.52**$_{\pm 0.7}$
	500	88.07$_{\pm 0.7}$	91.27$_{\pm 0.8}$	**91.44**$_{\pm 0.6}$
	1000	88.77$_{\pm 1.0}$	91.75$_{\pm 0.8}$	**92.34**$_{\pm 0.5}$
	2000	91.01$_{\pm 1.0}$	92.72$_{\pm 0.5}$	**93.40**$_{\pm 0.2}$
CB	4	57.78$_{\pm 10.9}$	51.11$_{\pm 9.2}$	**60.74**$_{\pm 16.66}$
	16	**77.04**$_{\pm 7.2}$	74.81$_{\pm 5.4}$	76.29$_{\pm 4.45}$
	32	80.0$_{\pm 7.6}$	74.81$_{\pm 5.9}$	**81.48**$_{\pm 6.2}$
	100	85.93$_{\pm 5.4}$	80.74$_{\pm 7.6}$	**87.41**$_{\pm 2.96}$
	250	85.19$_{\pm 4.7}$	86.67$_{\pm 5.0}$	**89.63**$_{\pm 4.32}$

Sentiment Analysis				
IMDB	4	77.23$_{\pm 3.0}$	81.55$_{\pm 1.9}$	**81.77**$_{\pm 1.8}$
	16	82.74$_{\pm 1.7}$	82.54 $_{\pm 1.0}$	**84.06**$_{\pm 0.7}$
	32	83.42$_{\pm 1.0}$	83.39 $_{\pm 0.8}$	**84.64**$_{\pm 0.4}$
	100	84.58$_{\pm 0.6}$	83.35 $_{\pm 0.8}$	**84.74**$_{\pm 0.4}$
	500	84.99$_{\pm 0.3}$	85.37$_{\pm 0.5}$	**86.00**$_{\pm 0.2}$
	1000	85.50$_{\pm 0.1}$	86.27$_{\pm 0.4}$	**86.37** $_{\pm 0.4}$
	2000	86.01$_{\pm 0.2}$	86.57$_{\pm 0.2}$	**86.60**$_{\pm 0.1}$
Yelp polarity	4	76.85$_{\pm 14.3}$	81.37$_{\pm 13.1}$	**90.25**$_{\pm 1.0}$
	16	87.84$_{\pm 1.5}$	**91.08**$_{\pm 0.2}$	90.36$_{\pm 1.2}$
	32	89.22$_{\pm 0.7}$	91.09$_{\pm 0.5}$	**91.15**$_{\pm 0.5}$
	100	90.19$_{\pm 0.7}$	90.15$_{\pm 0.7}$	**91.06**$_{\pm 0.6}$
	500	90.92$_{\pm 0.2}$	91.52$_{\pm 0.2}$	**92.09**$_{\pm 0.4}$
	1000	91.32$_{\pm 0.2}$	92.26$_{\pm 0.6}$	**92.50**$_{\pm 0.2}$
	2000	91.68$_{\pm 0.1}$	92.36$_{\pm 0.4}$	**92.70**$_{\pm 0.1}$

Low-resource Learning

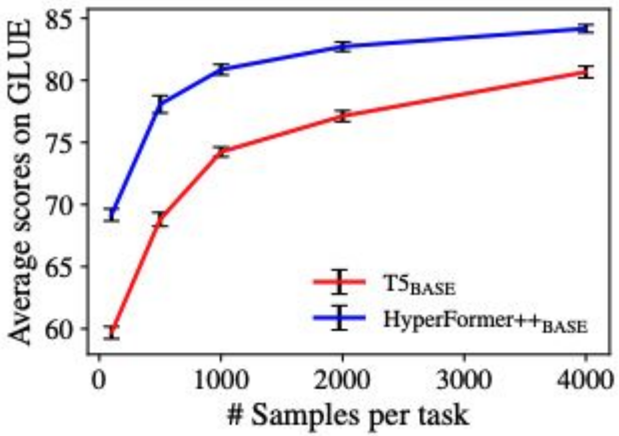


Figure 2: Results on GLUE for the various number of training samples per task (100,500,1000,2000,4000). We show mean and standard deviation across 5 seeds.

Ablation Studies

Model variant	GLUE
HYPERFORMER$_{\text{SMALL}}$	82.47
– Adapter blocks	68.37
– Conditional layer norm	79.83
– Task projector	81.56
– T5 Layer norm	81.29
– Conditional layer norm, T5 Layer norm	78.92

Table 4: Impact when removing different components of our framework. We report the average results on GLUE.

Visualization

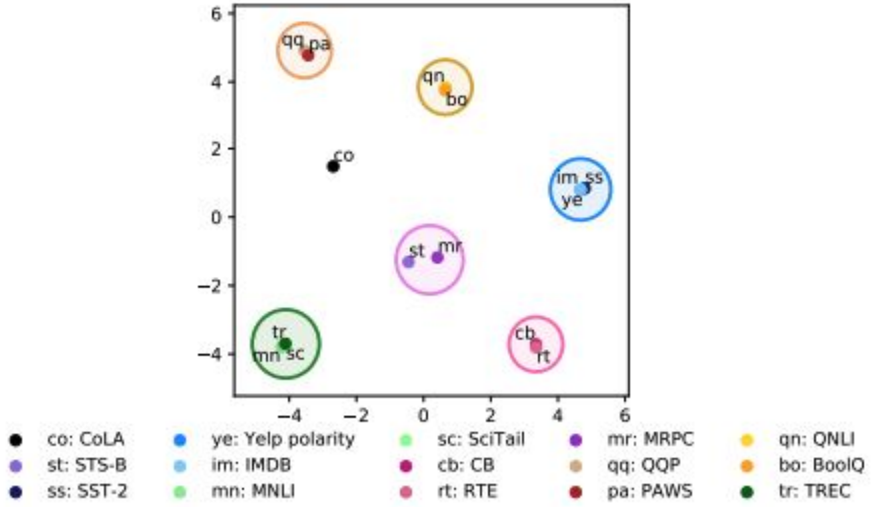


Figure 3: Visualization of learned task embeddings by HYPERFORMER++$_{\text{BASE}}$.