BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models

Elad Ben-Zaken[1] **Shauli Ravfogel**[1,2] **Yoav Goldberg**[1,2]
[1]Computer Science Department, Bar Ilan University
[2]Allen Institute for Artificial Intelligence

張信富　洪慶弦

Outline

- Abstract
- Background
- Existing Approaches
- Method Overview (BitFit)
- Experiments and Result
- Conclusions

Abstract

Abstract

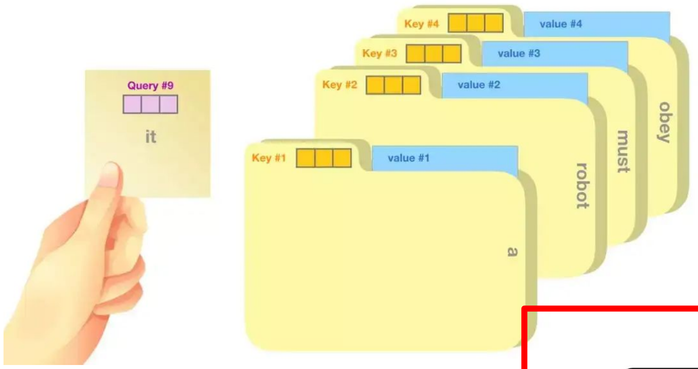
- **BitFit**: Fine-tuning method focusing only on model's bias terms.
- **Key Hypothesis**:
  - Fine-tuning exposes pre-trained knowledge rather than learning new linguistic knowledge.
- **Key Benefits**:
  - Competitive performance on small-to-medium datasets.
  - Efficient deployment with reduced parameter updates.

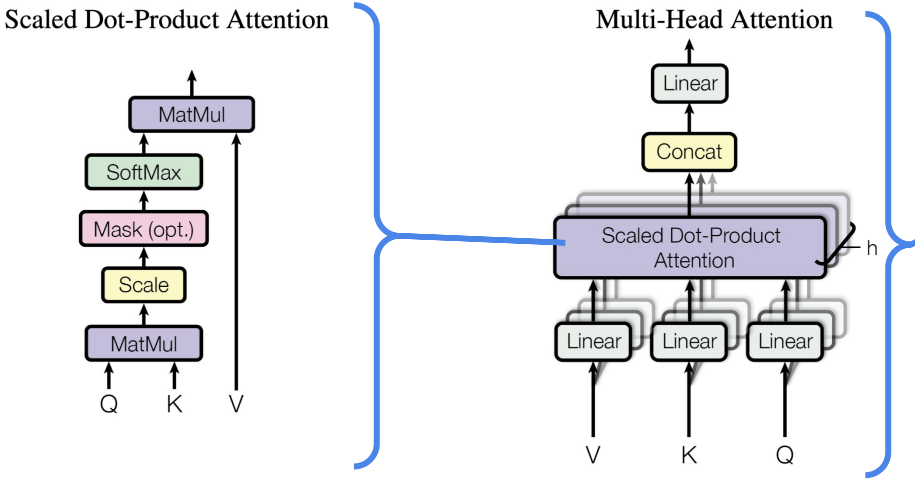
Background

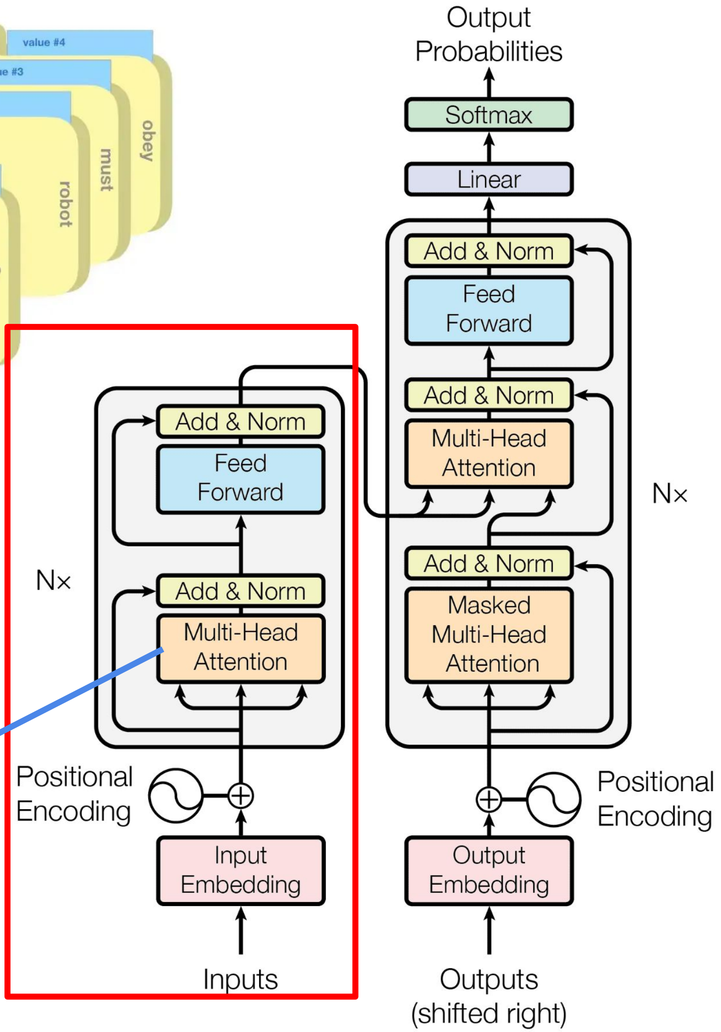
Background

Transformer Models:

- Dominate NLP tasks (e.g., BERT, RoBERTa).
- However, expensive to fine-tune and deploy.







Background

Challenges:

1. Fine-tuning changes all model parameters.
2. Large memory demand for multi-task applications.

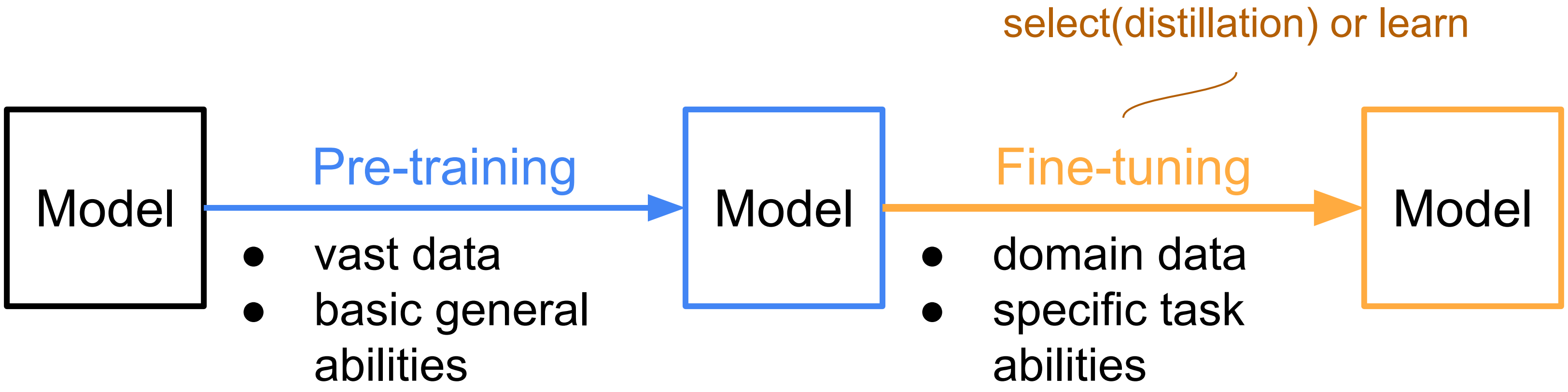
Key Question:

- How much of fine-tuning modifies pre-trained knowledge vs. exposes it?

Background

Key Question:

- How much of fine-tuning
  modifies pre-trained knowledge vs. exposes it?



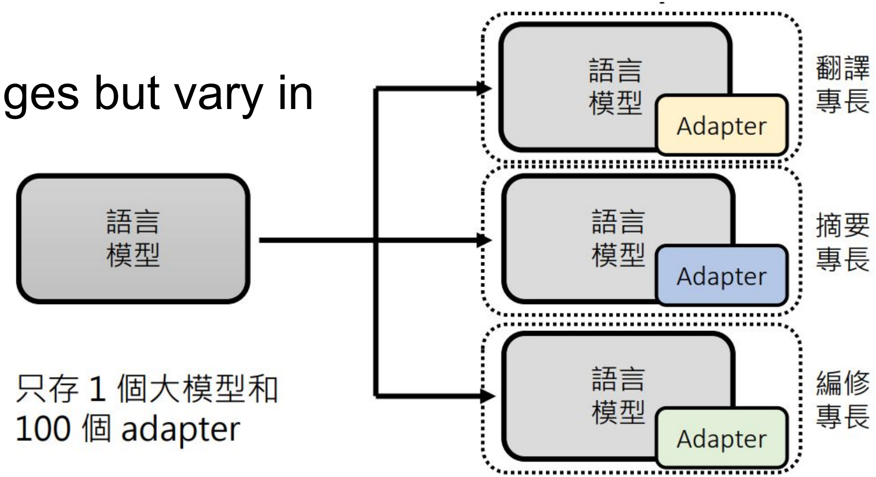
Existing Approaches

Existing Approaches

- **Existing Methods**:
  - **Adapters**: Task-specific modules between layers.
  - **Diff-Pruning**: Sparse difference vectors for tasks.
- **Comparison**:
  - Both methods limit parameter changes but vary in efficiency and accuracy.

$$\boldsymbol{\theta}_\tau = \boldsymbol{\theta} + \boldsymbol{\delta}_\tau$$

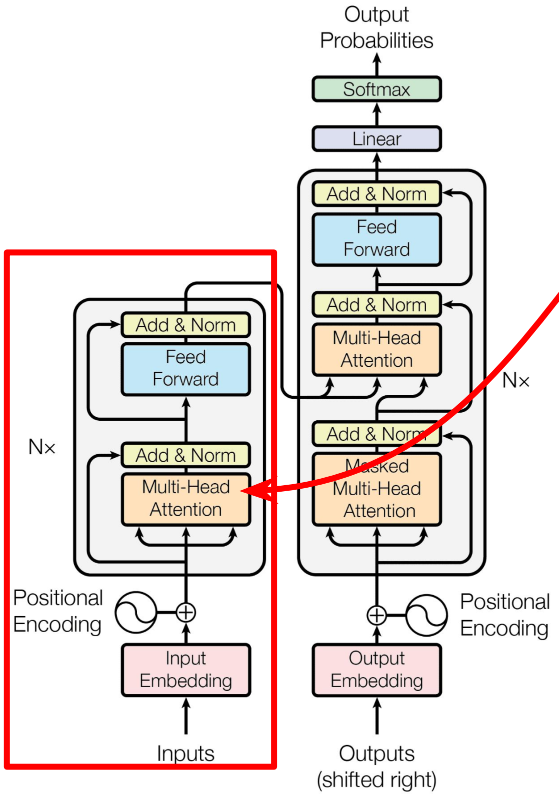
Diff-Pruning



Adapter

Method Overview (BitFit)

Method Overview (BitFit)



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

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

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

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

MLP classifier

$$\mathbf{h}_2^{\ell} = \mathrm{Dropout}\left(\mathbf{W}_{m_1}^{\ell} \cdot \mathbf{h}_1^{\ell} + \mathbf{b}_{m_1}^{\ell}\right)$$

$$\mathbf{h}_3^{\ell} = \mathbf{g}_{LN_1}^{\ell} \odot \frac{(\mathbf{h}_2^{\ell} + \mathbf{x}) - \mu}{\sigma} + \mathbf{b}_{LN_1}^{\ell}$$

$$\mathbf{h}_4^{\ell} = \mathrm{GELU}\left(\mathbf{W}_{m_2}^{\ell} \cdot \mathbf{h}_3^{\ell} + \mathbf{b}_{m_2}^{\ell}\right)$$

$$\mathbf{h}_5^{\ell} = \mathrm{Dropout}\left(\mathbf{W}_{m_3}^{\ell} \cdot \mathbf{h}_4^{\ell} + \mathbf{b}_{m_3}^{\ell}\right)$$

$$\mathrm{out}^{\ell} = \mathbf{g}_{LN_2}^{\ell} \odot \frac{(\mathbf{h}_5^{\ell} + \mathbf{h}_3^{\ell}) - \mu}{\sigma} + \mathbf{b}_{LN_2}^{\ell}$$

Method Overview (BitFit)

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

Key Idea:

- Fine-tune **only bias terms** (b) in transformers.

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

Key Properties:

1. Matches performance of full fine-tuning.
2. Minimal parameter updates (< 0.1%).
3. Task-invariant and hardware-friendly.

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

Experiments and Result

Experiments and Result

GLUE Benchmark (8 tasks) evaluate with Accuracy, F1, Spearman correlation.

	Train size	%Param	Avg.
(V)	Full-FT†	100%	**84.8**
(V)	Full-FT	100%	84.1
(V)	Diff-Prune†	0.5%	**84.6**
(V)	BitFit	0.08%	84.2
(T)	Full-FT‡	100%	81.2
(T)	Full-FT†	100%	**81.8**
(T)	Adapters‡	3.6%	81.1
(T)	Diff-Prune†	0.5%	**81.5**
(T)	BitFit	0.08%	80.9

$\text{BERT}_{\text{Large}}$ Comparison

	Method	%Param	Avg.
BB	Full-FT	100%	82.3
BB	BitFit	0.09%	**82.4**
BL	Full-FT	100%	84.1
BL	BitFit	0.08%	**84.2**
Ro	Full-FT	100%	**85.3**
Ro	BitFit	0.09%	84.6

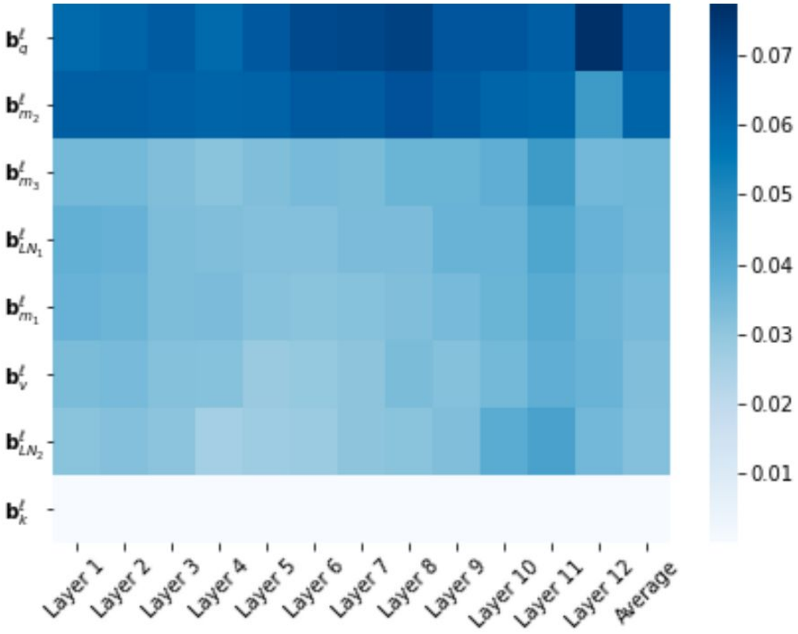
$\text{BERT}_{\text{Base}}$,
$\text{BERT}_{\text{Large}}$,
$\text{RoBERTa}_{\text{Base}}$
compare with others

	% Param	Avg.
Full-FT	100%	82.3
BitFit	0.09%	**82.4**
$\mathbf{b}_{m2}, \mathbf{b}_q$	0.04%	81.1
$\mathbf{b}_{m2}$	0.03%	80.0
$\mathbf{b}_q$	0.01%	76.6
Frozen	0.0%	62.1
rand uniform	0.09%	78.5
rand row/col	0.09%	79.5

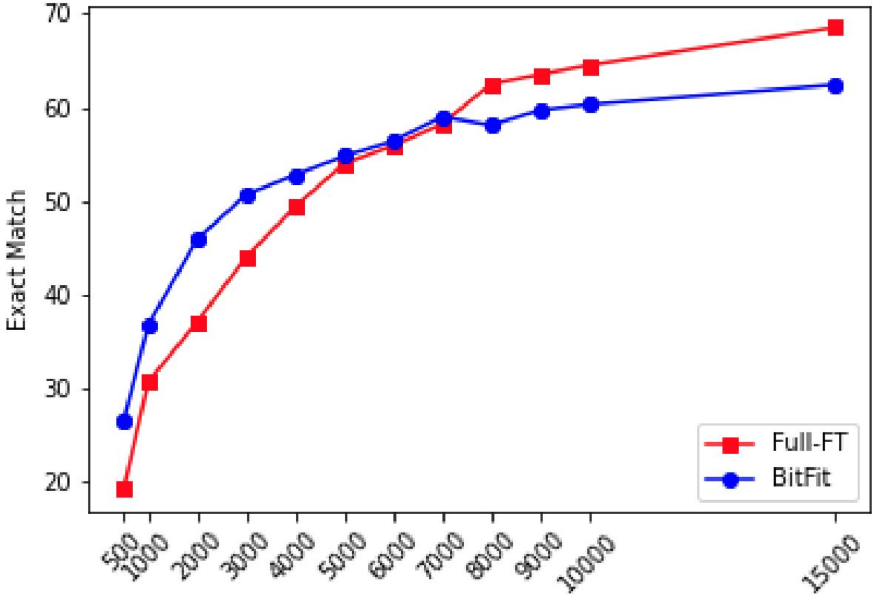
$\text{BERT}_{\text{Base}}$ with different modify rate on RTE

(Recognizing Textual Entailment)

Experiments and Result



$\text{BERT}_{\text{Base}}$ change per bias term and layer on RTE task



$\text{BERT}_{\text{Base}}$ SQuAD score on various size of datasets

Conclusions

Conclusions

BitFit achieves:

- Parameter efficiency.
- Competitive task performance.
- Scalability across tasks and hardware constraints.

Future Work:

- Investigate bias terms' role in transfer learning.
- Extend to non-language domains.

Q & A