EMNLP2022

논문 세미나

문현석

Natural Language Processing
& Artificial Intelligence

Parameter-Efficient Tuning
(PETuning)

1. **An Empirical Study on the Transferability of Transformer Modules in Parameter-Efficient Fine-Tuning**
https://aclanthology.org/2022.emnlp-main.726.pdf

2. **Revisiting Parameter-Efficient Tuning: Are We Really There Yet?**
https://aclanthology.org/2022.emnlp-main.168.pdf

3. **Parameter-Efficient Tuning Makes a Good Classification Head**
https://aclanthology.org/2022.emnlp-main.514.pdf

Introduction

Main Motivation

Lottery Ticket Hypothesis

: We can find a good sub-network

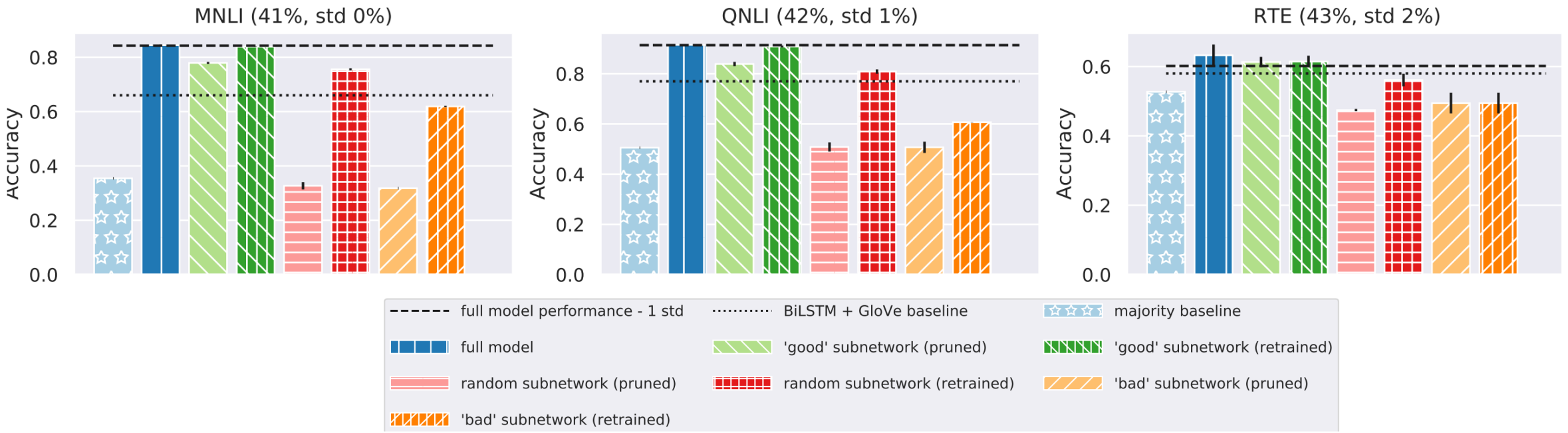
which can compete with the full fine-tuning setting

➡ Winning Ticket !!

Introduction

Previous Work

An Empirical Study on the Transferability of Transformer Modules in Parameter-Efficient Fine-Tuning



PLM parameter 중 일부만을 fine-tunining해도

PLM 전부를 fine-tuning했을 때 얻을 수 있는 성능을 낼 수 있다

단, 학습하는 parameter를 ”잘”선택했을 경우

https://aclanthology.org/2020.emnlp-main.259.pdf

Introduction

Methods

An Empirical Study on the Transferability of Transformer Modules in Parameter-Efficient Fine-Tuning

Previous works

- **Adapter**

  : Freeze PLM, then training only adapter

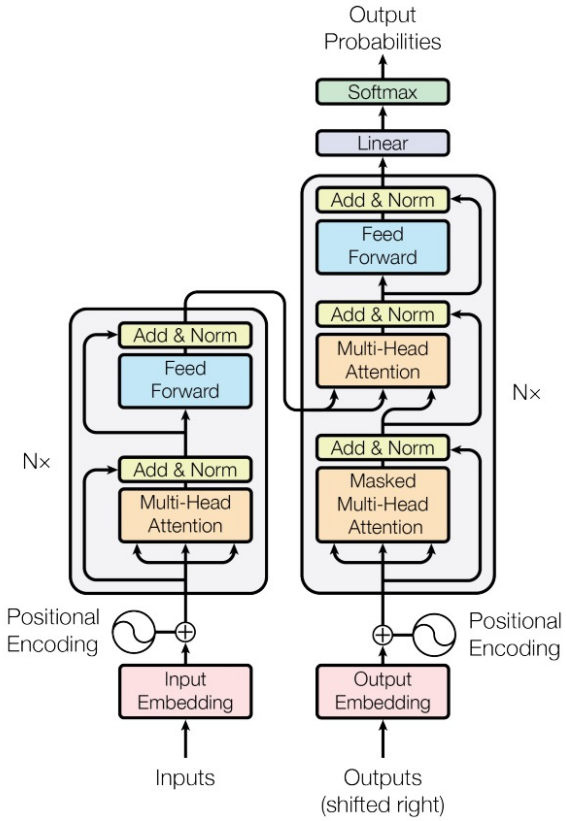
- **Cross-Attention**

  : Training only cross-attention modules in a PLM

- **BitFit**

  : Training only bias-terms in a PLM

This works



Module-wise Fine-tuning

- FFN
- Multi-head
- LayerNorm

Experiments

Main Results

검증 목적

- we check if the winning ticket is a property that can be associated only with some specific modules in the transformer block.

Model	%Param.	CoLA	SST-2	MRPC	STS-B	QQP	MNLI_m	MNLI_{mm}	QNLI	RTE	*Avg.*
Full-FT	100.0%	54.7±0.7	**93.1±0.1**	90.9±1.0	89.0±0.5	**87.3±0.1**	**83.4±0.2**	**83.7±0.4**	**91.5±0.1**	71.7±1.2	82.8
Feed-Forward	51.76%	56.2±1.5	92.7±0.1	91.1±0.5	88.9±0.6	87.1±0.1	81.9±0.3	82.7±0.2	90.8±0.0	71.1±0.8	82.5
Multi-Head	25.89%	**59.0±1.3**	92.9±0.3	**91.2±0.3**	**89.1±0.5**	86.7±0.2	83.2±0.2	83.6±0.3	91.2±0.1	72.0±0.2	**83.2**
LayerNorms	0.03%	52.8±2.0	92.2±0.4	91.0±0.2	88.4±0.1	81.5±0.1	79.3±0.2	80.7±0.3	89.4±0.2	70.2±0.4	80.6
LayerNorms_A	0.02%	50.8±0.5	91.8±0.2	90.6±0.5	88.3±0.2	80.9±0.1	76.3±0.2	75.9±0.3	88.8±0.1	70.5±0.9	79.3
LayerNorms_F	0.02%	52.9±1.9	91.7±0.2	91.0±0.2	88.4±0.1	81.0±0.1	77.0±0.1	77.2±0.3	88.3±0.0	69.2±0.7	79.6
BitFit	0.12%	53.6±1.9	89.6±0.2	90.5±0.2	88.2±0.0	81.8±0.3	†81.4±0.2	†82.2±0.2	†90.2±0.2	**72.8±0.2**	81.1
Random	0.03%	35.5±3.0	86.4±0.3	85.1±0.5	84.3±0.0	74.8±0.2	64.5±0.2	66.1±0.3	76.4±0.0	60.8±0.5	70.4
Frozen	0.00%	35.2±1.6	81.5±0.3	80.7±0.1	68.3±0.1	60.1±0.2	44.0±0.3	45.1±0.2	68.8±0.1	58.6±0.5	60.25

Table 1: The performance of BERT on the GLUE benchmark with different fine-tuning strategies. We report Matthew's correlation for CoLA, F1 score for MRPC and QQP, Spearman's correlation for STS-B, and accuracy for the rest. **LayerNorms_A** (**LayerNorms_F**) stands for the scenario in which only the LayerNorms of **A**ttention (**F**eedforward) modules are set to be trainable. The best and the second-best results are highlighted for each task. † Results from Ben Zaken et al. (2022).

Experiments

Main Results

검증 결과

- All individual modules possess ”winning ticket property” to some extent

Model	%Param.	CoLA	SST-2	MRPC	STS-B	QQP	MNLI_m	MNLI_{mm}	QNLI	RTE	*Avg.*
Full-FT	100.0%	54.7±0.7	**93.1±0.1**	90.9±1.0	89.0±0.5	**87.3±0.1**	**83.4±0.2**	**83.7±0.4**	**91.5±0.1**	71.7±1.2	82.8
Feed-Forward	51.76%	56.2±1.5	92.7±0.1	91.1±0.5	88.9±0.6	87.1±0.1	81.9±0.3	82.7±0.2	90.8±0.0	71.1±0.8	82.5
Multi-Head	25.89%	**59.0±1.3**	92.9±0.3	**91.2±0.3**	**89.1±0.5**	86.7±0.2	83.2±0.2	83.6±0.3	91.2±0.1	72.0±0.2	**83.2**
LayerNorms	0.03%	52.8±2.0	92.2±0.4	91.0±0.2	88.4±0.1	81.5±0.1	79.3±0.2	80.7±0.3	89.4±0.2	70.2±0.4	80.6
LayerNorms_A	0.02%	50.8±0.5	91.8±0.2	90.6±0.5	88.3±0.2	80.9±0.1	76.3±0.2	75.9±0.3	88.8±0.1	70.5±0.9	79.3
LayerNorms_F	0.02%	52.9±1.9	91.7±0.2	91.0±0.2	88.4±0.1	81.0±0.1	77.0±0.1	77.2±0.3	88.3±0.0	69.2±0.7	79.6
BitFit	0.12%	53.6±1.9	89.6±0.2	90.5±0.2	88.2±0.0	81.8±0.3	†81.4±0.2	†82.2±0.2	†90.2±0.2	**72.8±0.2**	81.1
Random	0.03%	35.5±3.0	86.4±0.3	85.1±0.5	84.3±0.0	74.8±0.2	64.5±0.2	66.1±0.3	76.4±0.0	60.8±0.5	70.4
Frozen	0.00%	35.2±1.6	81.5±0.3	80.7±0.1	68.3±0.1	60.1±0.2	44.0±0.3	45.1±0.2	68.8±0.1	58.6±0.5	60.25

Table 1: The performance of BERT on the GLUE benchmark with different fine-tuning strategies. We report Matthew's correlation for CoLA, F1 score for MRPC and QQP, Spearman's correlation for STS-B, and accuracy for the rest. **LayerNorms_A** (**LayerNorms_F**) stands for the scenario in which only the LayerNorms of **A**ttention (**F**eedforward) modules are set to be trainable. The best and the second-best results are highlighted for each task. † Results from Ben Zaken et al. (2022).

Experiments

Main Results

검증 결과

- Layer Normalization can transfer knowledge (37k among 110M)

Model	%Param.	CoLA	SST-2	MRPC	STS-B	QQP	MNLI_m	MNLI_{mm}	QNLI	RTE	*Avg.*
Full-FT	100.0%	54.7±0.7	**93.1±0.1**	90.9±1.0	89.0±0.5	**87.3±0.1**	**83.4±0.2**	**83.7±0.4**	**91.5±0.1**	71.7±1.2	82.8
Feed-Forward	51.76%	56.2±1.5	92.7±0.1	91.1±0.5	88.9±0.6	87.1±0.1	81.9±0.3	82.7±0.2	90.8±0.0	71.1±0.8	82.5
Multi-Head	25.89%	**59.0±1.3**	92.9±0.3	**91.2±0.3**	**89.1±0.5**	86.7±0.2	83.2±0.2	83.6±0.3	91.2±0.1	72.0±0.2	**83.2**
LayerNorms	0.03%	52.8±2.0	92.2±0.4	91.0±0.2	88.4±0.1	81.5±0.1	79.3±0.2	80.7±0.3	89.4±0.2	70.2±0.4	80.6
LayerNorms_A	0.02%	50.8±0.5	91.8±0.2	90.6±0.5	88.3±0.2	80.9±0.1	76.3±0.2	75.9±0.3	88.8±0.1	70.5±0.9	79.3
LayerNorms_F	0.02%	52.9±1.9	91.7±0.2	91.0±0.2	88.4±0.1	81.0±0.1	77.0±0.1	77.2±0.3	88.3±0.0	69.2±0.7	79.6
BitFit	0.12%	53.6±1.9	89.6±0.2	90.5±0.2	88.2±0.0	81.8±0.3	†81.4±0.2	†82.2±0.2	†90.2±0.2	**72.8±0.2**	81.1
Random	0.03%	35.5±3.0	86.4±0.3	85.1±0.5	84.3±0.0	74.8±0.2	64.5±0.2	66.1±0.3	76.4±0.0	60.8±0.5	70.4
Frozen	0.00%	35.2±1.6	81.5±0.3	80.7±0.1	68.3±0.1	60.1±0.2	44.0±0.3	45.1±0.2	68.8±0.1	58.6±0.5	60.25

Table 1: The performance of BERT on the GLUE benchmark with different fine-tuning strategies. We report Matthew's correlation for CoLA, F1 score for MRPC and QQP, Spearman's correlation for STS-B, and accuracy for the rest. **LayerNorms_A** (**LayerNorms_F**) stands for the scenario in which only the LayerNorms of **A**ttention (**F**eedforward) modules are set to be trainable. The best and the second-best results are highlighted for each task. † Results from Ben Zaken et al. (2022).

Experiments

Analysis

분석 결과

- different layers do not contribute equally to the ultimate performance in transfer learning
- Layer normalization in middle layer can transfer knowledge (5k among 110M)

Task	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	Full-FT
CoLA	44.0±1.5	43.3±2.8	46.8±2.4	47.6±1.7	46.1±1.9	47.0±2.1	47.0±3.4	**48.1±0.9**	47.1±2.5	45.7±1.5	42.3±1.9	36.9±0.2	**54.7±0.7**
MRPC	84.7±0.5	85.9±0.8	84.2±0.3	85.9±0.6	**89.0±1.1**	88.8±0.4	87.5±0.7	86.1±0.2	86.6±0.2	86.4±0.7	85.2±0.5	82.6±0.1	**90.9±1.0**
STS-B	85.1±0.3	85.7±0.1	86.1±0.1	86.1±0.2	86.7±0.3	87.1±0.1	86.9±0.1	**87.2±0.1**	86.7±0.1	86.6±0.1	86.5±0.1	83.5±0.2	**89.0±0.5**
RTE	61.5±1.7	65.3±0.6	63.9±1.5	64.1±0.6	65.2±0.6	64.1±1.5	67.3±0.3	**67.5±1.0**	63.5±0.8	65.8±0.3	67.0±0.2	60.5±1.4	**71.7±1.2**

Table 3: The performance of layer-wise fine-tuning of LayerNorms on the selected downstream tasks for BERT. The LayerNorms in the middle layers tend to have the highest transferability.

Conclusion

- Our results pave the way for better parameter efficient fine-tuning of large language models
  - without the need for costly algorithms to determine the optimum sub-network
  - Without introducing additional parameters for knowledge transfer

Introduction

Main Motivation

Can Parameter-Efficient Tuning replace full fine-tuning ?

1. In current evaluation strategies, dev set is used for both early stopping and reporting results

2. PETuning processed are inherently unstable due to various randomness

Key Findings

1. No PETuning consistently outperform finetuning
: PETuning may be more suitable for low-resource tasks

2. PETuning show instability across different random seeds
: Randomness comes from both weight initialisation and training data order

3. Prompt-tuning lags far behind finetuning
: prompt-tuning is unstable and cannot robustly and consistently re-produce its reported performance

4. Reducing the size of trainable parameters is likely to yield better stability
: not necessary to yield better or poorer performance

5. The stability is proportional to the scale of training data
: We further highlight the most crucial factor behind is the number of training iterations

Analysis

The Broken Protocol

GLUE / superGLUE : Test sets are not released

→ Model performance is reported based on the dev set

→ Convenient approximation, but data leakage problem - single set is used for both validation and test

→ Reported results might come from overly-optimistic checkpoint

	Evaluation loss		*Accuracy*	
	RTE_{1-2}	RTE_{2-2}	RTE_{1-2}	RTE_{2-2}
FT	**78.89**$_{\pm 1.36}$	**78.89**$_{\pm 1.36}$	79.28$_{\pm 1.9}$	**79.62**$_{\pm 2.22}$
Adapter	75.1$_{\pm 1.60}$	**76.3**$_{\pm 4.26}$	76.55$_{\pm 3.57}$	**78.42**$_{\pm 3.7}$
PT	57.55$_{\pm 2.71}$	**66.19**$_{\pm 8.51}$	57.84$_{\pm 4.85}$	**67.19**$_{\pm 11.37}$
LoRA	75.22$_{\pm 2.77}$	**75.94**$_{\pm 3.39}$	75.11$_{\pm 3.3}$	**77.7**$_{\pm 4.57}$
BitFit	70.79$_{\pm 10.38}$	**71.3**$_{\pm 10.19}$	66.76$_{\pm 12.98}$	**68.2**$_{\pm 13.72}$

Table 1: Mean and standard deviation results with different dev/test splits for RTE task across 20 runs. *Evaluation loss* and *accuracy* are the stopping metrics. Bold denotes the highest mean value for corresponding method with specific stopping metric.

Split Dev set into two part : Dev1 / Dev2

RTE_{1-2} : validation - Dev1 / test - Dev2

RTE_{2-2} : validation - Dev2 / test - Dev2

Analysis

Resource-wise

Revisiting Parameter-Efficient Tuning: Are We Really There Yet?

Dataset↓, Model→	FT	Adapter	PT	LoRA	BitFit
			Low-Resource		
CB	$70.00_{\pm 13.32}$	$77.49_{\pm 13.20}$	$46.55^{\downarrow}_{\pm 5.74}$	$\mathbf{82.05}^{\uparrow}_{\pm 9.62}$	$81.12^{\uparrow}_{\pm 8.94}$
COPA	$54.70_{\pm 3.36}$	$65.90^{\uparrow}_{\pm 5.42}$	$55.35_{\pm 5.07}$	$\mathbf{66.4}^{\uparrow}_{\pm 9.05}$	$56.65_{\pm 3.72}$
WSC	$\mathbf{63.46}_{\pm 0.0}$	$\mathbf{63.46}_{\pm 0.0}$	$58.7^{\downarrow}_{\pm 4.69}$	$\mathbf{63.46}_{\pm 0.0}$	$\mathbf{63.46}_{\pm 0.0}$
Avg. (Low)	$62.72_{\pm 4.58}$	$68.95^{\uparrow}_{\pm 4.83}$	$53.53^{\downarrow}_{\pm 3.22}$	$\mathbf{70.64}^{\uparrow}_{\pm 4.32}$	$67.08^{\uparrow}_{\pm 3.57}$
			Medium-Resource		
RTE	$73.77_{\pm 3.17}$	$\mathbf{73.88}_{\pm 1.88}$	$57.36^{\downarrow}_{\pm 8.01}$	$69.69^{\downarrow}_{\pm 7.89}$	$70.67_{\pm 10.77}$
MRPC	$90.54_{\pm 1.05}$	$\mathbf{91.06}_{\pm 0.63}$	$89.35^{\downarrow}_{\pm 1.31}$	$91.03_{\pm 0.95}$	$\mathbf{91.06}_{\pm 0.71}$
WiC	$65.47_{\pm 2.04}$	$65.12_{\pm 1.88}$	$62.12^{\downarrow}_{\pm 1.32}$	$61.29^{\downarrow}_{\pm 6.7}$	$\mathbf{66.0}_{\pm 1.41}$
STS-B	$90.42_{\pm 0.26}$	$90.23^{\downarrow}_{\pm 0.1}$	$89.64_{\pm 0.39}$	$\mathbf{90.47}_{\pm 0.11}$	$90.44_{\pm 0.15}$
BoolQ	$\mathbf{78.75}_{\pm 0.72}$	$76.93_{\pm 0.92}$	$75.44_{\pm 0.47}$	$76.92_{\pm 1.33}$	$76.9_{\pm 0.84}$
Avg. (Medium)	$\mathbf{79.79}_{\pm 0.99}$	$79.44_{\pm 0.74}$	$74.78^{\downarrow}_{\pm 1.62}$	$77.88^{\downarrow}_{\pm 2.02}$	$79.01_{\pm 2.22}$
			High-Resource		
SST-2	$\mathbf{94.15}_{\pm 0.0}$	$93.34^{\downarrow}_{\pm 0.31}$	$\mathbf{94.15}_{\pm 0.0}$	$\mathbf{94.15}_{\pm 0.0}$	$93.92_{\pm 0.07}$
QNLI	$\mathbf{92.40}_{\pm 0.12}$	$92.31_{\pm 0.09}$	$92.31_{\pm 0.27}$	$91.00_{\pm 0.69}$	$91.60_{\pm 1.01}$
QQP	$\mathbf{91.38}_{\pm 0.06}$	$90.28_{\pm 0.0}$	$88.90^{\downarrow}_{\pm 0.32}$	$90.45^{\downarrow}_{\pm 0.17}$	$89.28^{\downarrow}_{\pm 0.0}$
MNLI	$\mathbf{87.42}_{\pm 0.20}$	$86.88^{\downarrow}_{\pm 0.17}$	$86.30^{\downarrow}_{\pm 0.08}$	$86.96_{\pm 0.24}$	$85.50^{\downarrow}_{\pm 0.32}$
Avg. (High)	$\mathbf{91.34}_{\pm 0.09}$	$90.70^{\downarrow}_{\pm 0.12}$	$90.42^{\downarrow}_{\pm 0.14}$	$90.64^{\downarrow}_{\pm 0.21}$	$90.08^{\downarrow}_{\pm 0.29}$
Avg. (All)	79.37	**80.57**	74.68	80.32	79.72

Table 2: Mean and standard deviation results for each of the 12 tasks across finetuning (FT) and four PETuning methods. We report the F1 score for CB and MRPC, Pearson correlation for STS-B, and accuracy for other tasks (matched accuracy for MNLI). Higher is better for all metrics. One-tailed t-test is used for the comparison between PETuning and finetuning. One PETuning method outperforms (↑) or falls behind (↓) finetuning when accepting the corresponding alternative hypothesis, where p-value < 0.05 (meaning the difference is significant).

	Low	Medium	High
Adapter	↗	→	↘
PT	↘	↘	↘
LoRA	↗	↘	↘
BitFit	↗	→	↘

Table 3: Performance comparison between PETuning and finetuning on low-, medium-, and high-resource settings, respectively. Arrows indicate whether corresponding PETuning method significantly outperforms finetuning (↗), falls behind (↘), or their results across multiple runs without significant differences (→).

Low : < 1k data points

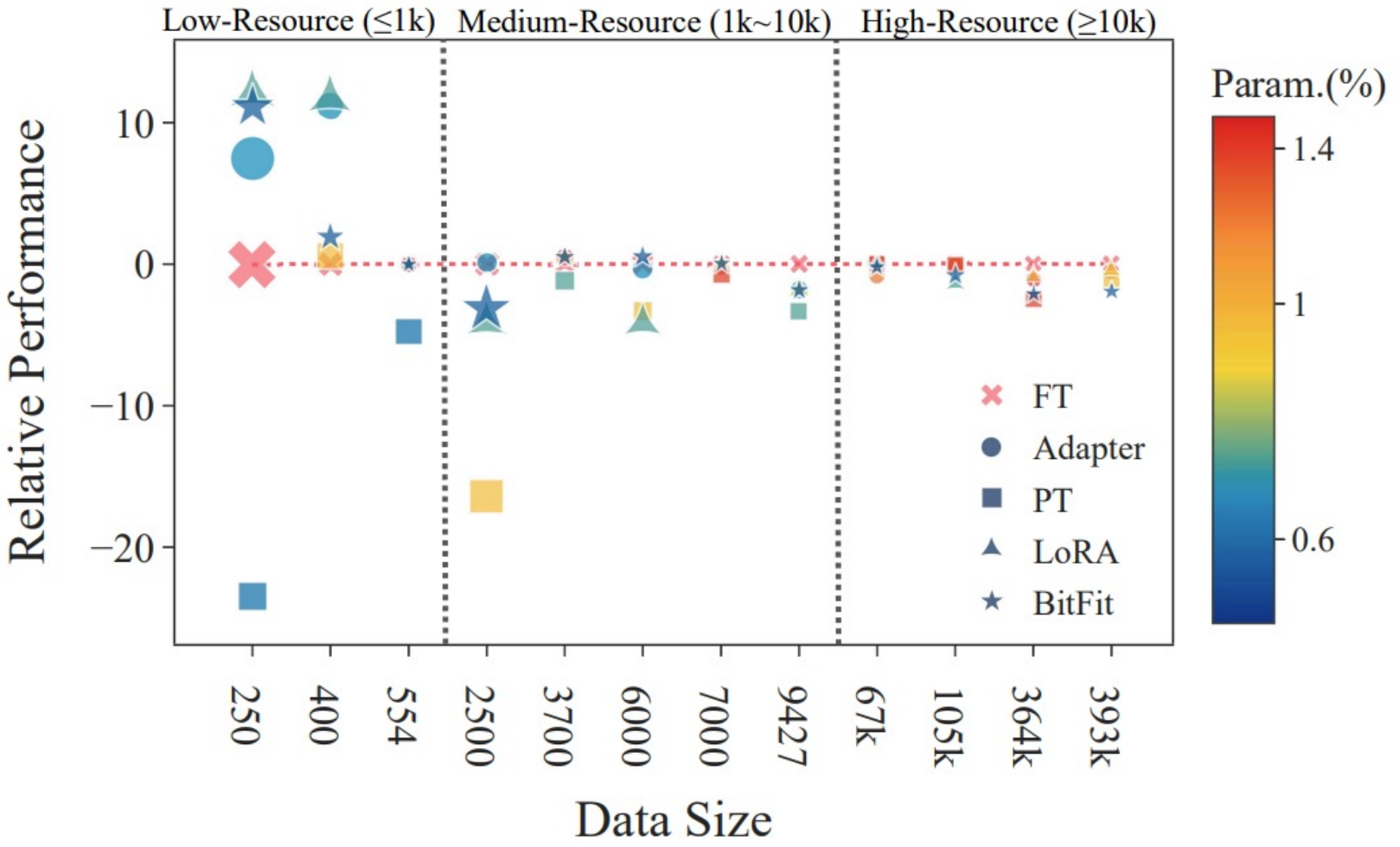
Medium : < 10k data points

High : > 10k data points

Split training data (9:1)
→ Regard training / dev set

Analysis

Resource-wise



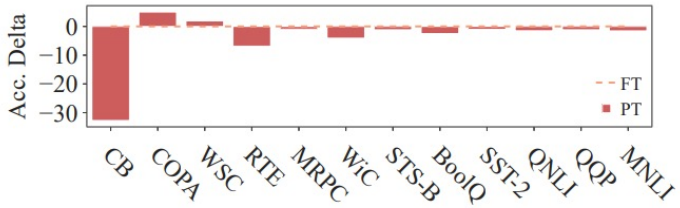


Figure 3: Relative performance differences of prefix tuning (PT) over full finetuning (FT) on the upper bounds of multi-run results. PT achieves close upper bounds compared with FT on most of the 12 tasks.

Analysis

Analysis of Stability

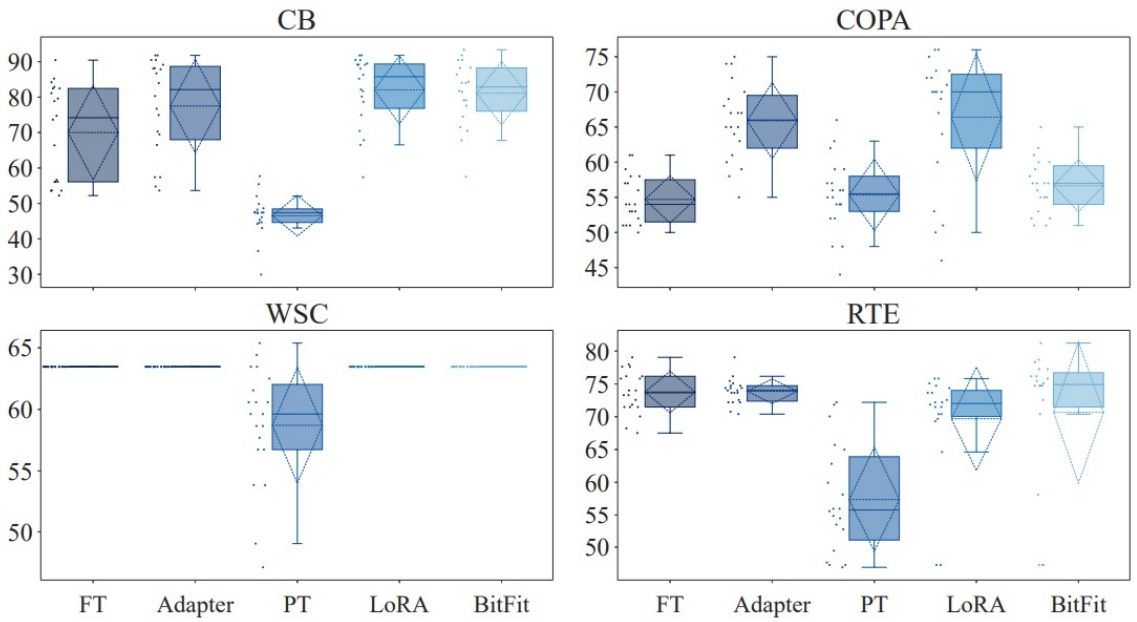


Figure 4: The experimental results over 20 different random seeds across CB, COPA, WSC, and RTE datasets, where finetuning and PETuning methods show large instability. The dashed rhombuses denote the mean (horizontal dashed line) and standard deviation (vertical distance).

	WI	DO	Global
FT	**55.40**$_{\pm 4.55}$	55.35$_{\pm \mathbf{3.32}}$	54.70$_{\pm 3.36}$
Adapter	**67.15**$_{\pm \mathbf{5.40}}$	66.35$_{\pm 7.36}$	65.90$_{\pm 5.42}$
PT	55.00$_{\pm 5.13}$	54.75$_{\pm \mathbf{4.97}}$	**55.35**$_{\pm 5.07}$
LoRA	63.60$_{\pm \mathbf{7.93}}$	64.60$_{\pm 8.56}$	**66.40**$_{\pm 9.05}$
BitFit	**58.40**$_{\pm \mathbf{2.29}}$	56.00$_{\pm 4.00}$	56.65$_{\pm 3.72}$

Table 4: Performance over 20 runs on COPA task, controlled by global random seeds, weight initialization (WI) random seeds, and data order (DO) random seeds, respectively. (Visualised in Figure 12 in the Appendix.)

Instability caused by

- Weight Initialization
- Data Order

Analysis

Analysis of Stability

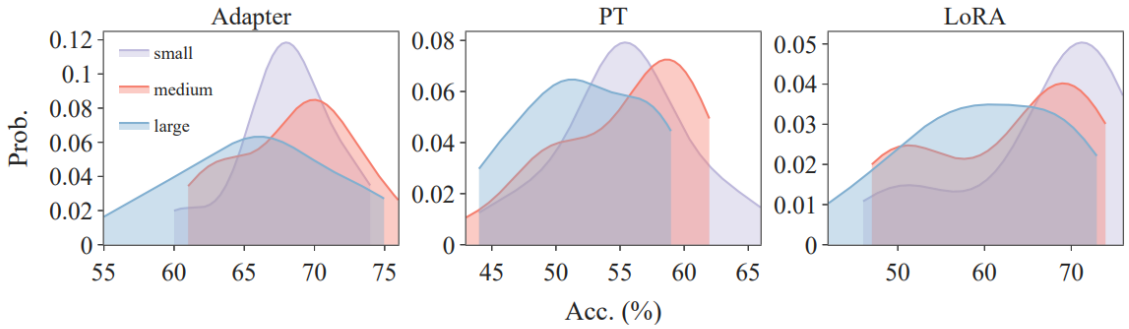
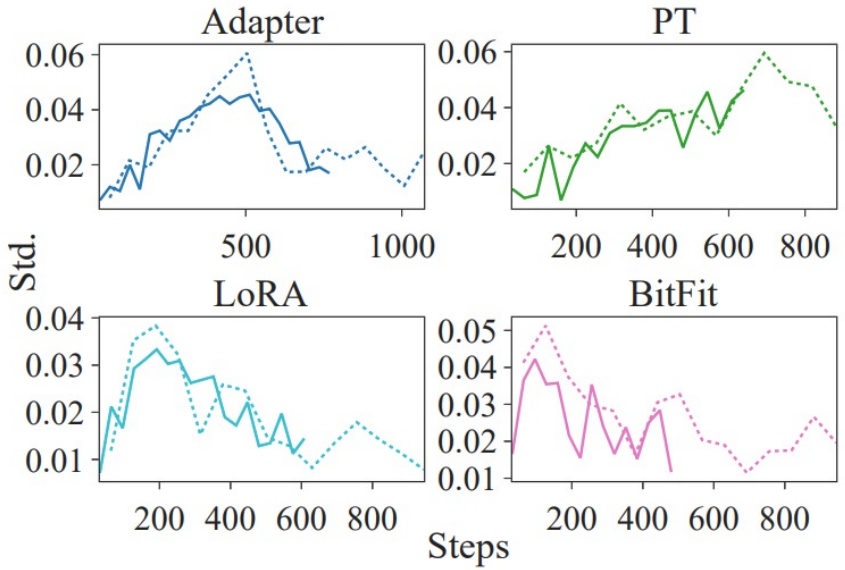


Figure 5: Performance probability density curves of Adapter, prefix tuning (PT), and LoRA over small, medium, and large parameter scales on COPA task across 20 runs. (See the numerical results and analyses in Appx. §C.3.)

Fewer trainable parameters,

More stable performance

(그래프가 가로로 넓을수록 unstable)



The true underlying variable
that leads to the discrepancy of instability across different training data sizes

is essentially the number of training iterations/steps.

실선: 동일 task에 대해 1k training data sampling
점선: 동일 task에 대해 2k training data sampling

Conclusion

- Finetuning still cannot be fully replaced by PETuning so far,

- There are many key challenges for PETuning in terms of both performance and instability

(However, PETuning is a fast-moving field and our conclusions do not necessarily generalise to all existing and upcoming models.)

Introduction

Main Motivation

How to utilize PETuning even more effectively ?

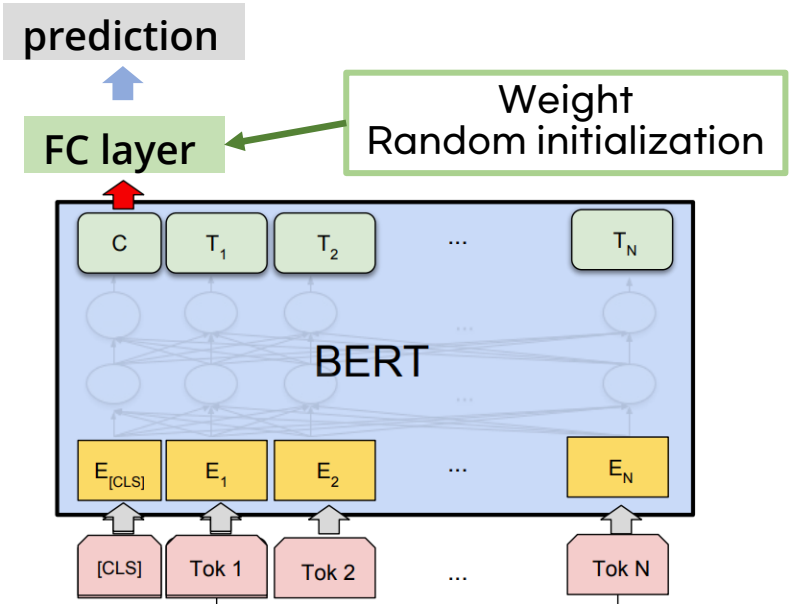
PETuning then Fine-tuning
(EH-FT)

Introduction

Background

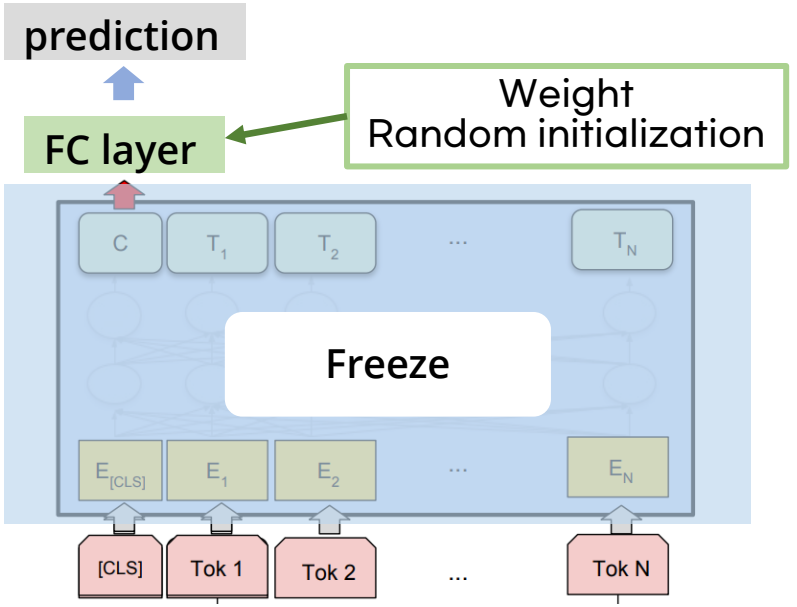
FINE-TUNING CAN DISTORT PRETRAINED FEATURES

AND UNDERPERFORM OUT-OF-DISTRIBUTION **(ICLR2022)**



Conventional

Fine-tuning



Linear

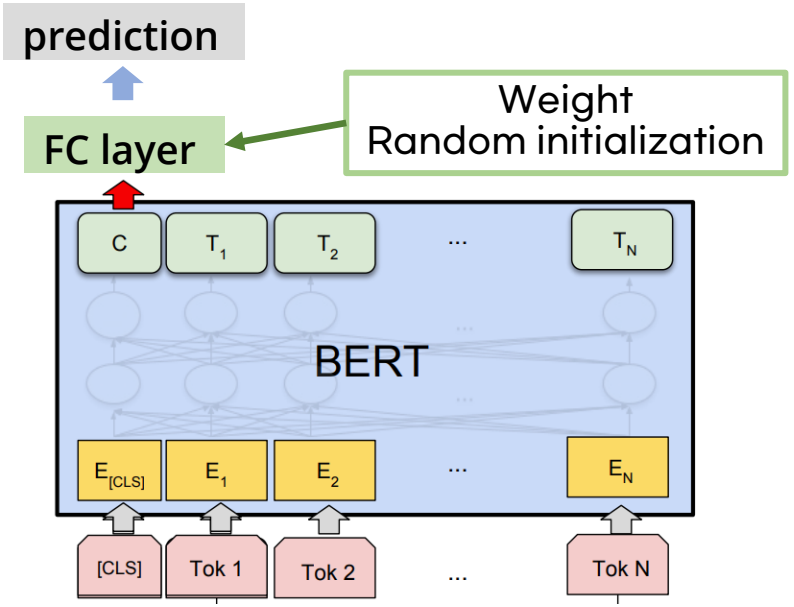
Probing

Introduction

Background

FINE-TUNING CAN DISTORT PRETRAINED FEATURES

AND UNDERPERFORM OUT-OF-DISTRIBUTION **(ICLR2022)**



Conventional

Fine-tuning

Random Initialization

of the parameters in FC layer (head)

PLM의 weight를 매우 크게 변화시킴

→주어진 task에 대해서는 높은 적합도를 보일 수는 있음 (in-distribution)

→단, out-of-distribution에 대해서는 낮은 적합도 (distort pretrained feature)

Introduction

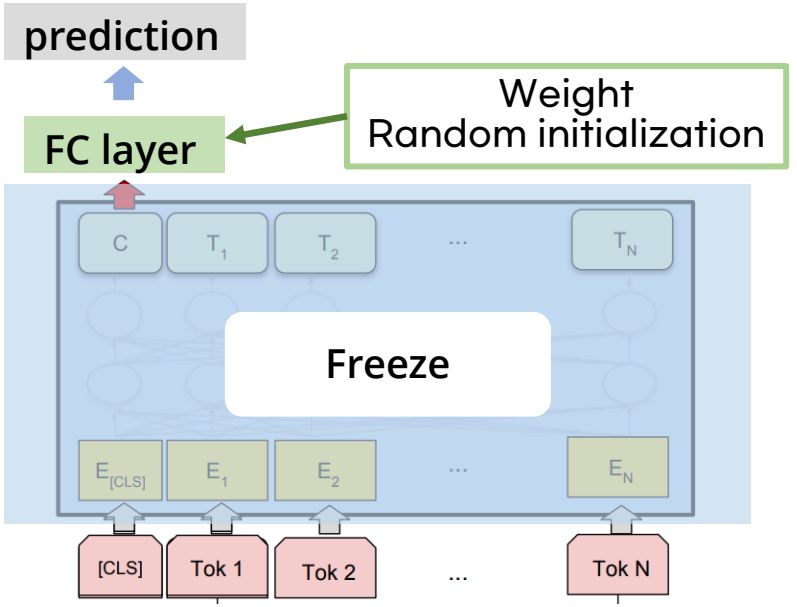
Background

Revisiting Parameter-Efficient Tuning: Are We Really There Yet?

FINE-TUNING CAN DISTORT PRETRAINED FEATURES
AND UNDERPERFORM OUT-OF-DISTRIBUTION **(ICLR2022)**

In-distribution (ID) 에 대해서는
전체 모델을 학습할 때 보다 낮은 성능을 보이나,

기존 PLM이 보전되기 때문에
Out-of-distribution (OOD) 데이터에서는
준수한 성능을 냄



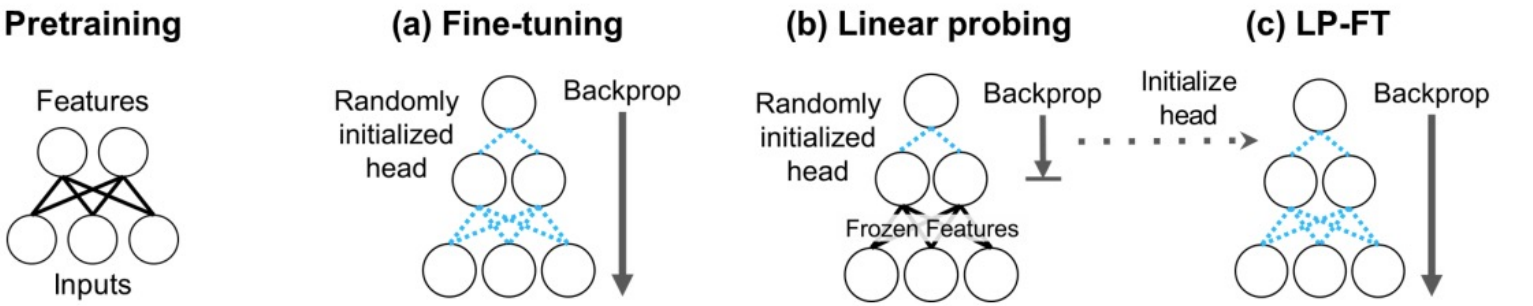
Linear
Probing

Introduction

Background

FINE-TUNING CAN DISTORT PRETRAINED FEATURES

AND UNDERPERFORM OUT-OF-DISTRIBUTION **(ICLR2022)**



	(a) Fine-tuning	(b) Linear probing	(c) LP-FT
ID test	85.1%	82.9%	**85.7%**
OOD test	59.3%	66.2%	**68.9%**

Average accuracies (10 distribution shifts)

Linear probing 이후 전체 모델 fine-tuning

→ Relieving distortion of pretrained feature during fine-tuning

High ID performance without deteriorating OOD performance

Introduction

Background

FINE-TUNING CAN DISTORT PRETRAINED FEATURES
AND UNDERPERFORM OUT-OF-DISTRIBUTION **(ICLR2022)**

왜 head-first fine-tuning (LP-FT)가 효과적인가?

→ random initialized head를 통해 fine-tuning을 진행하는 경우

In-distribution과 관련한 feature는 크게 변하나, Out-of-distribution과 관련한 feature는 거의 변하지 않기 때문에,

Linear probing으로 good classification head를 만든다면?

→ 해당 head는 ID와 OOD를 모두 잘 다루게 될 것

Linear probing으로 학습된 head를 활용해서 fine-tuning을 진행한다면?

→ ID와 OOD의 feature에 대해서 모두 고르게 학습할 수 있을 것이다

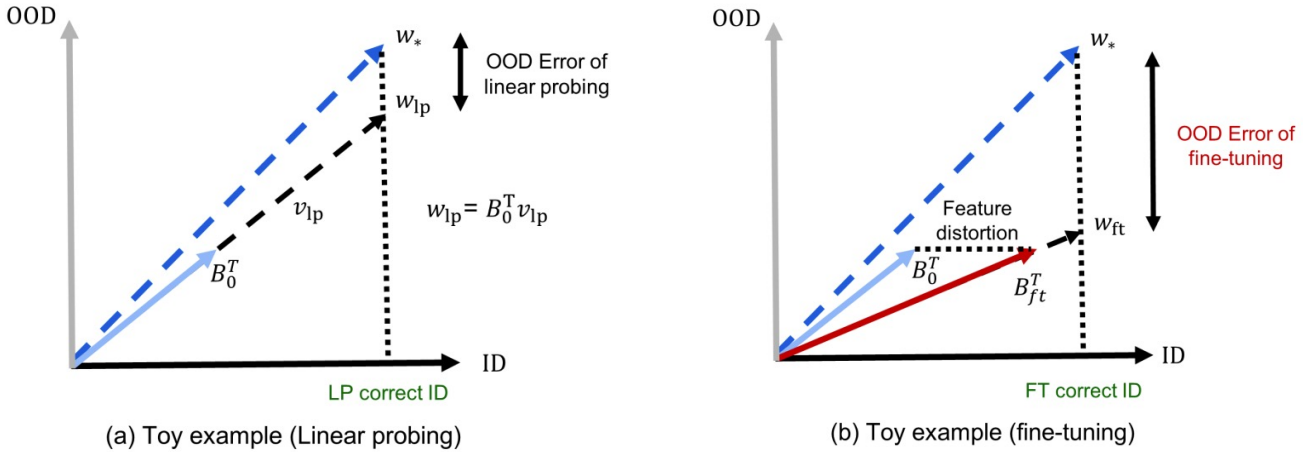
Introduction

Background

Revisiting Parameter-Efficient Tuning: Are We Really There Yet?

FINE-TUNING CAN DISTORT PRETRAINED FEATURES

AND UNDERPERFORM OUT-OF-DISTRIBUTION **(ICLR2022)**



(a) Toy example (Linear probing)

(b) Toy example (fine-tuning)

전체 모델을 fine-tuning할 경우,
Linear probing을 하는 경우에 비해서
OOD에 더 많이 취약

Feature Extraction Model

1. Linear Model 인 경우 ($f_{v,B}(x) = \bar{v}^\top Bx$)
   : 이론적으로 증명
2. Image Model 인 경우 (CLIP)
   : 실험적으로 밝힘

This Work

Does this kind of head-first finetuning technique also work in NLP?

Head-only tuning in LP-FT cannot obtain a good enough classification head

if the pretrained task is greatly different from the downstream task

이전 논문에서 사용한 CLIP: image-caption matching contrastive learning

Pretrained Model in NLP: Masked language modeling

Method

Efficient-Head Fine-Tuning (EH-FT)

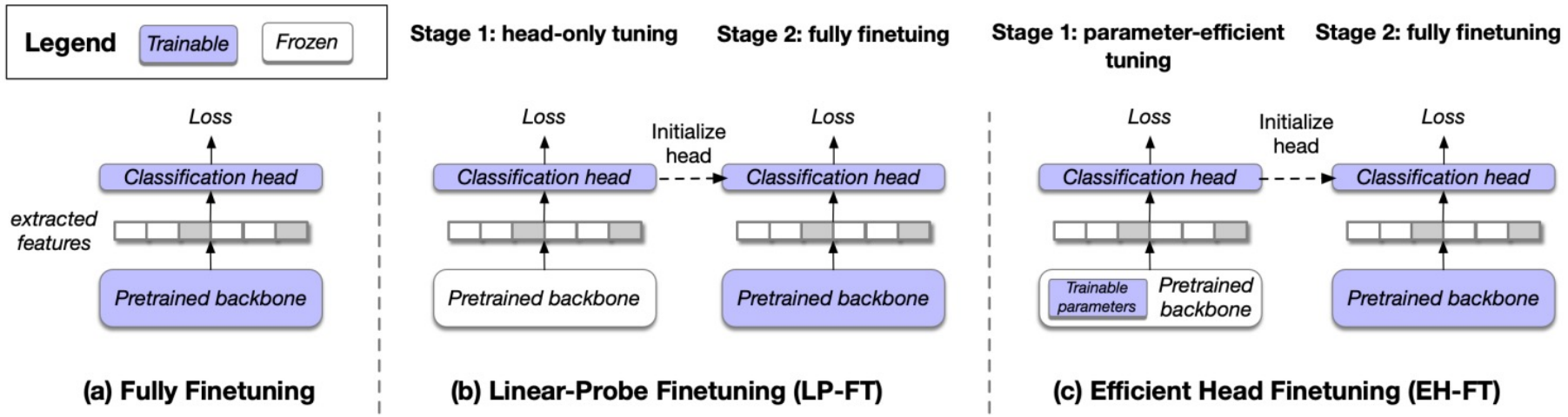


Figure 2: Illustration for different finetuning strategies. (a) Fully finetuning directly optimize all the parameters. (b) LP-FT first only trains the head then the whole model. (c) Our EH-FT first trains the head and a very small part (1%∼ 4%) of parameters by a parameter-efficient tuning algorithm at the stage 1, and then restart a fully finetuning with the trained head from the stage 1.

Working Principles

- The initialization of classification head matters
- LP-FT neglects the difference between the pretraining and downstream task
- Efficient head ensures a nearby near-optimal point for fine-tuning the backbone weights

Experiments

Main Results

Methods	RTE	BoolQ	COPA	CB	WIC	MRPC	QNLI	COLA	STS-B	Avg
Finetuning (Liu et al., 2019)	86.60‡	86.90	94.00	98.20‡	75.60	90.90‡	94.70	68.00	92.40‡	87.48
Finetuning (reproduce)	87.52	86.32	93.75	94.64	73.90	90.81	94.75	68.72	92.27	86.96
Linear Probing	61.10	64.31	80.25	79.47	67.97	75.45	69.95	33.17	60.25	65.77
Prefix-Tuning	77.00	83.90	87.75	**100**	65.05	89.56	94.55	57.89	89.92	82.85
LoRA	88.50	86.29	94.75	**100**	73.04	90.43	94.37	67.86	92.17	87.49
BitFit	87.05	86.13	95.50	99.11	72.01	90.32	94.68	57.89	92.05	87.24
Top-K Tuning	86.55	85.34	93.00	98.66	73.71	90.94	94.12	65.78	91.47	86.62
Mixout	85.56	86.06	95.00	98.66	74.45	90.87	94.18	65.45	91.62	86.87
Child-Tuning$_D$	88.18	86.65	94.5	92.86	74.07	91.36	94.44	68.52	**92.51**	87.00
LP-FT	86.14	86.38	94.00	93.50	74.73	**91.47**	94.78	67.45	92.20	86.74
EH-FT$_{\text{LoRA}}$	**88.68**	86.69	94.5	99.12	74.65	91.00	94.73	69.00	92.24	**87.85**
EH-FT$_{\text{PT}}$	87.22	86.9	**95.00**	**100**	73.63	90.56	**94.89**	**69.10**	92.31	87.73
EH-FT$_{\text{BitFit}}$	88.10	**86.97**	94.75	99.12	**75.20**	91.00	94.61	68.78	92.25	**87.86**

Table 1: Results with RoBERTa-Large. All scores are the mean result of 4 random seeds. Results with ‡ finetuned starting from the MNLI model, which is expected a better performance than single-task finetuning. It is recommended to compare the reproduced finetuing result with LP-FT and EH-FT, because they share the same code, only with different head initialization.

Experiments

Main Results

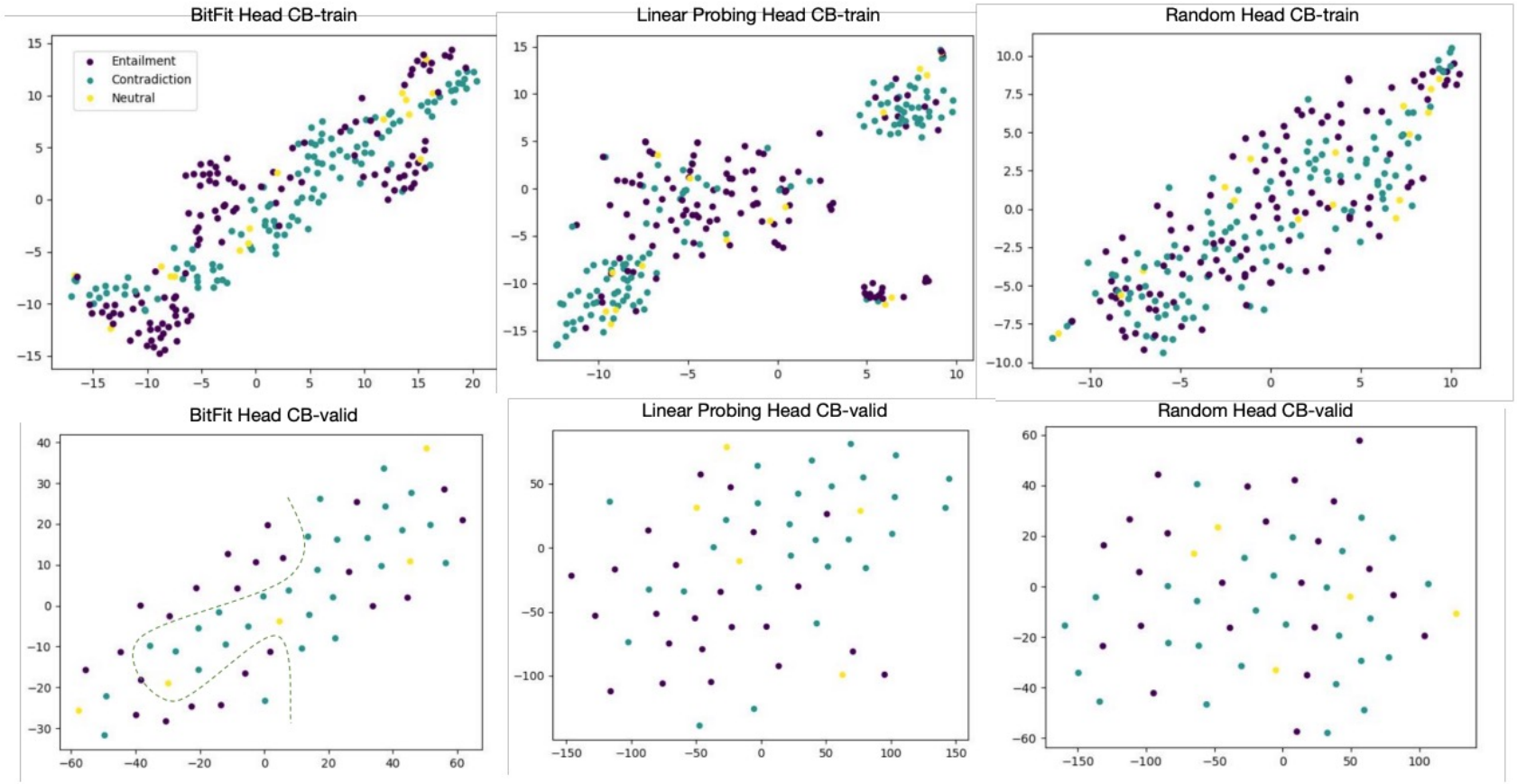


Figure 4: Features distribution with different head initialization strategies. We reduce the dimension of the features output by classification using T-SNE. The upper plots show the distribution of samples in CB training set, and the lower plots show the samples in development set. Different labels are distinguished by point color. **Parameter-efficient tuning can make a good clustering which is not highly concentrated.**

Conclusion

- EH-FT can make a good initialization of head which can guide model to a local minimum close to the pretrained point.
  - alleviating the catastrophic forgetting
  - alleviating the overfitting of large-scale pretrained models.

- EH-FT can be applied to any pretrained model, as long as it only needs a classification head for downstream tasks.

1. Big model의 사용자 관점에서, parameter efficient tuning에 대해 경시할 수는 없을 것

2. 과제하면서, 또한 개인 연구 하면서 폭넓게 적용 가능한 전략들

3. 많은 연구가 이루어져왔지만, 아직 많은 연구가 필요한 분야

감사합니다

Q&A