

REVISITING FEW-SAMPLE BERT FINE-TUNING

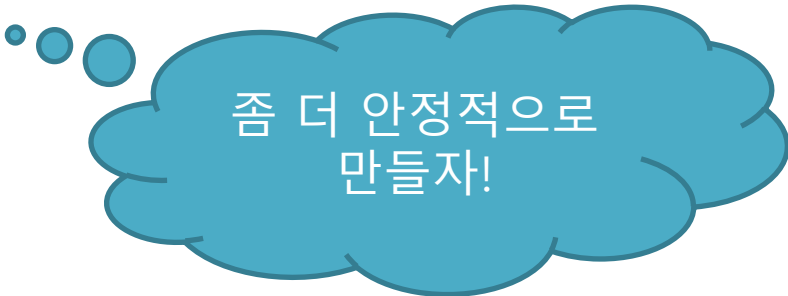
Contents

1. Introduction
2. Background and Related Work
3. Experimental Methodology
4. Optimization Algorithm
5. Initialization
6. Training Iterations
7. Conclusion

1. Introduction

Introduction

- 현재 NLP tasks에서의 S.O.T.A
 - Fine-tuning self-supervised pre-trained model
- 그 중 효율적인 모델인 BERT
- But,
 - Fine-tuning은 여전히 불안정
 - 소규모 데이터셋에서 BERT_{Large} 사용할 때 더 불안정
- 불안정?
 - 여러 랜덤 값에 의해 같은 모델 같은 데이터셋을 사용해도 검증 지표가 제각각
 - 모델 배포 코스트와 시간을 증가 시킴
 - 과학적인 비교 어려움



좀 더 안정적으로
만들자!

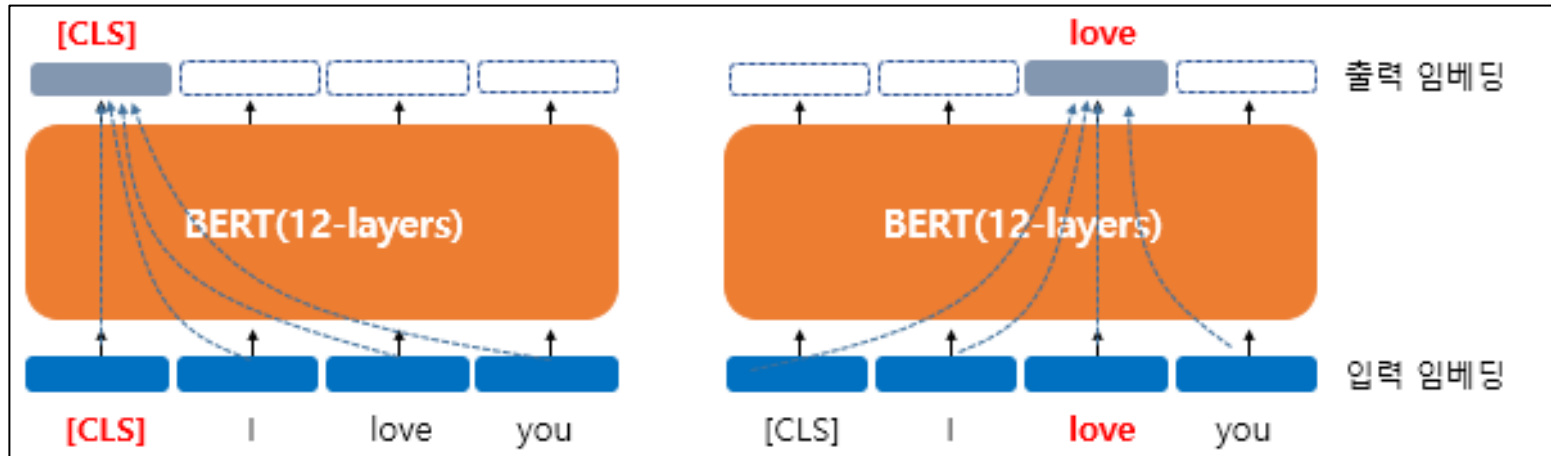
How?

- 최적화 프로세스를 위한 optimizer 알고리즘
- Fine-tuning 모델 초기화 방식
- Fine-tuning 학습의 iteration 횟수 선정

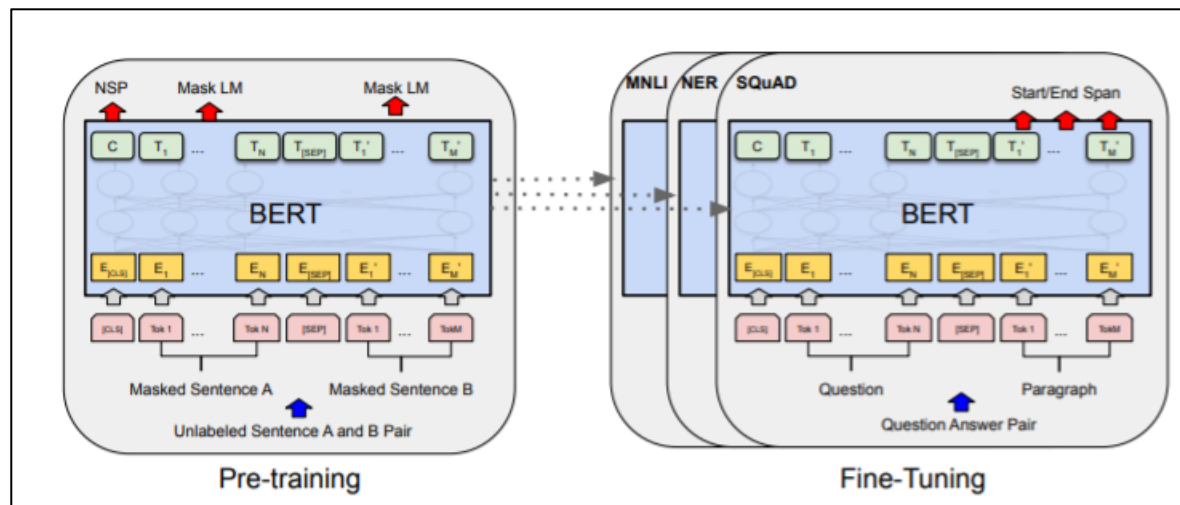
2. Background and Related Work

Background

- BERT



- Fine-tuning



Fine-tuning Instability

- 큰 intermediate task에서 pre-train 모델을 fine-tuning 하면 작은 데이터셋에서 fine-tuning이 안정화 됨 (Phang et al. (2018))
- Pre-train된 가중치에 가깝게 유지하도록 fine-tuning 모델을 제한하여 안정화 시킴 (Lee et al. (2020))
- 학습 과정에서 최종 성능이 낮게 나올 가능성이 있는 랜덤 시드 필터링을 위한 조기 중단 (Dodge et al. (2020))
- BERT의 optimizer인 BERTAdam이 fine-tuning의 불안정성을 초래 (Mosbach et al. (2020))

BERT Representation Transferability

- BERT의 pre-train된 representation은 중간 레이어의 pre-train된 피쳐가 fine-tuning 후 더 많이 변경되는 경우, 이후 레이어의 피쳐보다 더 transferability하거나 새로운 task에 적용할 수 있음 (Peters et al., 2019; Merchant et al., 2020)
- 즉, fine-tuning 후, later(상위) 레이어의 가중치가 더 많이 바뀐다고 함
- 본 논문에서는 pre-train된 가중치가 fine-tuning 프로세스에 어떻게 영향을 미치는지에 대해 중점을 두고 연구

3. Experimental Methodology

Data

- GLUE benchmark
 - Natural language inference (RTE, QNLI, MNLI)
 - Paraphrase detection (MRPC, QQP)
 - Sentiment classification (SST-2)
 - Linguistic acceptability (CoLA)

Task	RTE NLI	MRPC Paraphrase	STS-B Similarity	CoLA Acceptability	SST-2 Sentiment	QNLI NLI	QQP Paraphrase	MNLI NLI
# of training samples	2.5k	3.7k	5.8k	8.6k	61.3k	104k	363k	392k
# of validation samples	139	204	690	521	1k	1k	1k	1k
# of test samples	139	205	690	521	1.8k	5.5k	40k	9.8k
Evaluation metric	Acc.	F1	SCC	MCC	Acc.	Acc.	Acc.	Acc.
Majority baseline (val)	52.9	81.3	0	0	50.0	50.0	50.0	33.3
Majority baseline (test)	52.5	81.2	0	0	49.1	50.5	63.2	31.8

Experimental Setup

- 선행 연구의 하이퍼파라미터와 동일 (Lee et al. (2020))
- 24 layer BERT_{Large}
- 배치 사이즈: 32, 드롭아웃: 0.1, 에폭: 3
- 최대 학습률: 2×10^{-5}
 - 업데이트 이후 10% 동안 선형 학습률 워밍업
 - 이후 선형 감소
- Mixed precision training 사용
 - 실험 속도 향상을 위한 것
 - 통계적으로 유의하지 않아 사용 가능

	CoLA	MRPC	RTE	STS-B
Mixed precision	60.3 ± 1.5	89.2 ± 1.2	71.8 ± 2.1	90.1 ± 0.7
Full precision	59.9 ± 1.5	88.7 ± 1.4	71.4 ± 2.2	90.1 ± 0.7

- 조기 중단 적용 (validation 성능 10번 평가 후)
- 20개의 랜덤 시드 사용

4. Optimization Algorithm: Debiasing Omission in BERTAdam

Debiasing Omission in BERTAdam

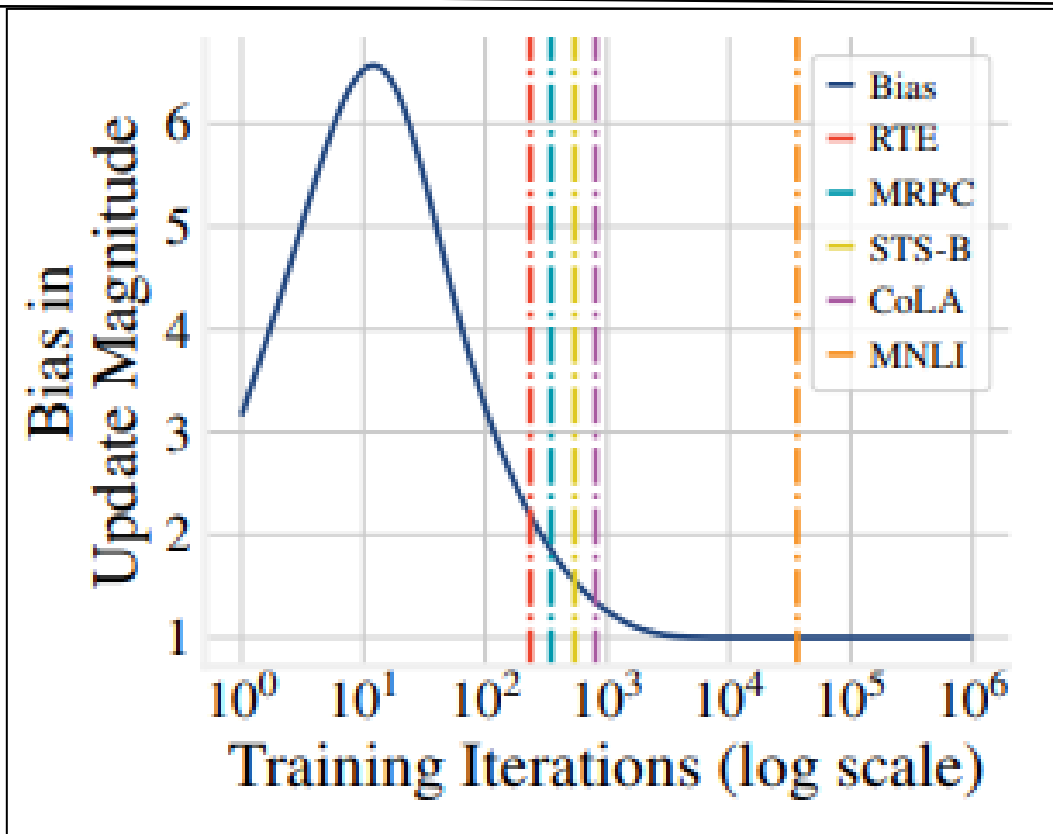
- BERTAdam
 - Adam 에서 편향 보정 생략
 - 본 논문에서는 BERT fine-tuning의 불안정성 요인으로 생각

Algorithm 1: the ADAM pseudocode adapted from Kingma & Ba (2014), and provided for reference. g_t^2 denotes the elementwise square $g_t \odot g_t$. β_1 and β_2 to the power t are denoted as β_1^t β_2^t . All operations on vectors are element-wise. The suggested hyperparameter values according to Kingma & Ba (2014) are: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. BERTADAM (Devlin et al., 2019) omits the bias correction (lines 9–10), and treats m_t and v_t as \hat{m}_t and \hat{v}_t in line 11.

Require: α : learning rate; $\beta_1, \beta_2 \in [0, 1)$: exponential decay rates for the moment estimates; $f(\theta)$: stochastic objective function with parameters θ ; θ_0 : initial parameter vector; $\lambda \in [0, 1)$: decoupled weight decay.

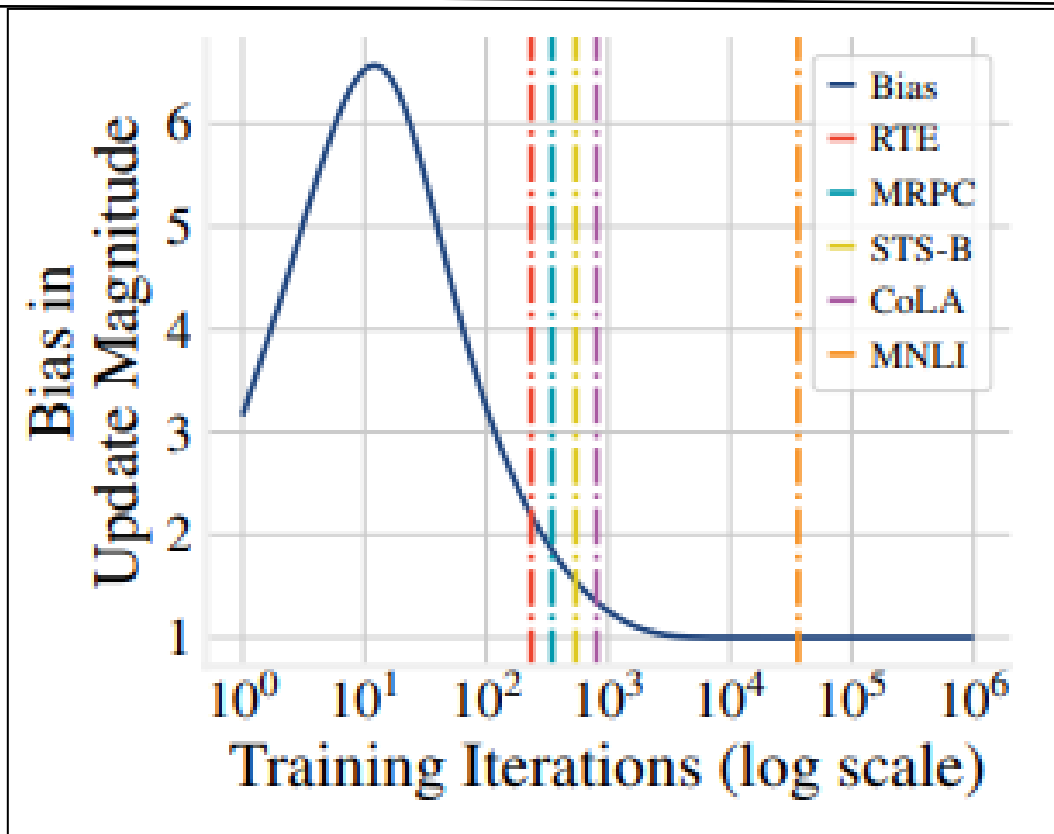
```
1:  $m_0 \leftarrow 0$  (Initialize first moment vector)
2:  $v_0 \leftarrow 0$  (Initialize second moment vector)
3:  $t \leftarrow 0$  (Initialize timestep)
4: while  $\theta_t$  not converged do
5:    $t \leftarrow t + 1$ 
6:    $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$  (Get gradients w.r.t. stochastic objective at timestep  $t$ )
7:    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$  (Update biased first moment estimate)
8:    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  (Update biased second raw moment estimate)
9:    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)
10:   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)
11:   $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  (Update parameters)
12: end while
13: return  $\theta_t$  (Resulting parameters)
```

Debiasing Omission in BERTAdam



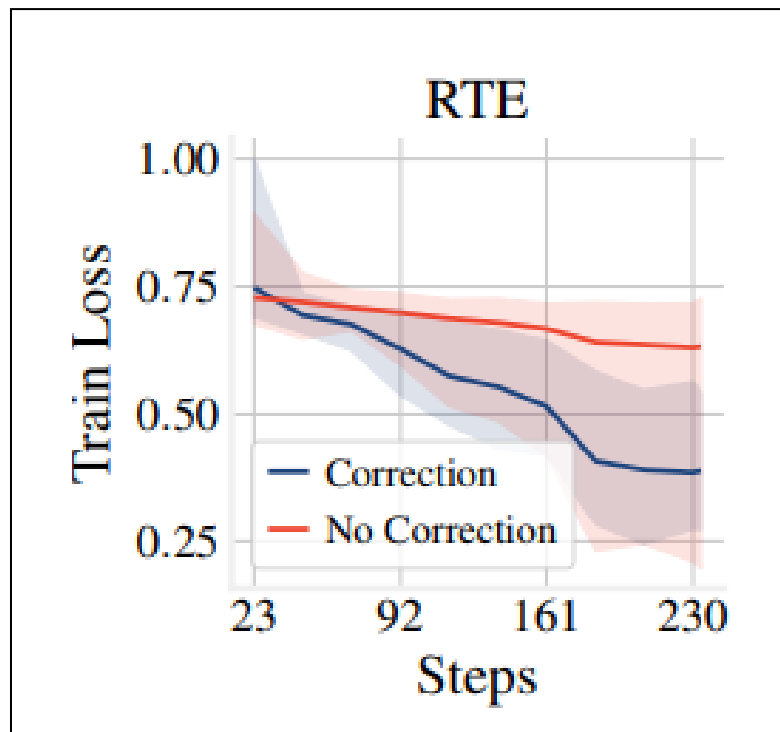
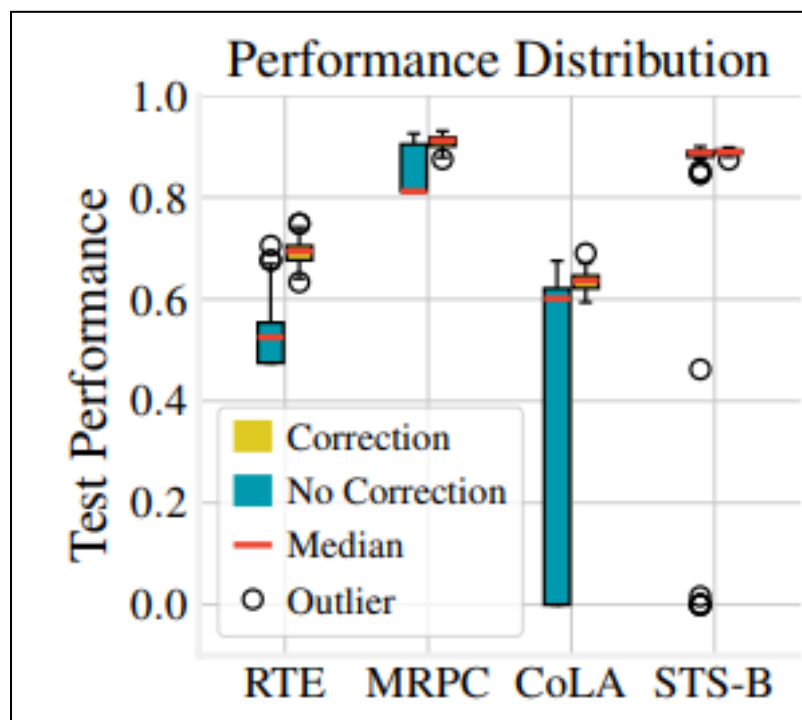
- 학습 iteration에 따른 편향된 추정값과 편향되지 않은 추정값을 사용한 업데이트 간의 비율 ($\frac{\hat{m}_t}{\sqrt{\hat{v}_t}}$)
- 편향이 1로 수렴하면 편향 추정치 무시 가능

Debiasing Omission in BERTAdam



- 작은 데이터셋은 편향 비율이 1보다 높아 학습 어려움
- MNLI는 수렴하는 영역에서 발생 → 상대적 안정적

Debiasing Omission in BERTAdam



- 50개의 랜덤 시드에서 실험

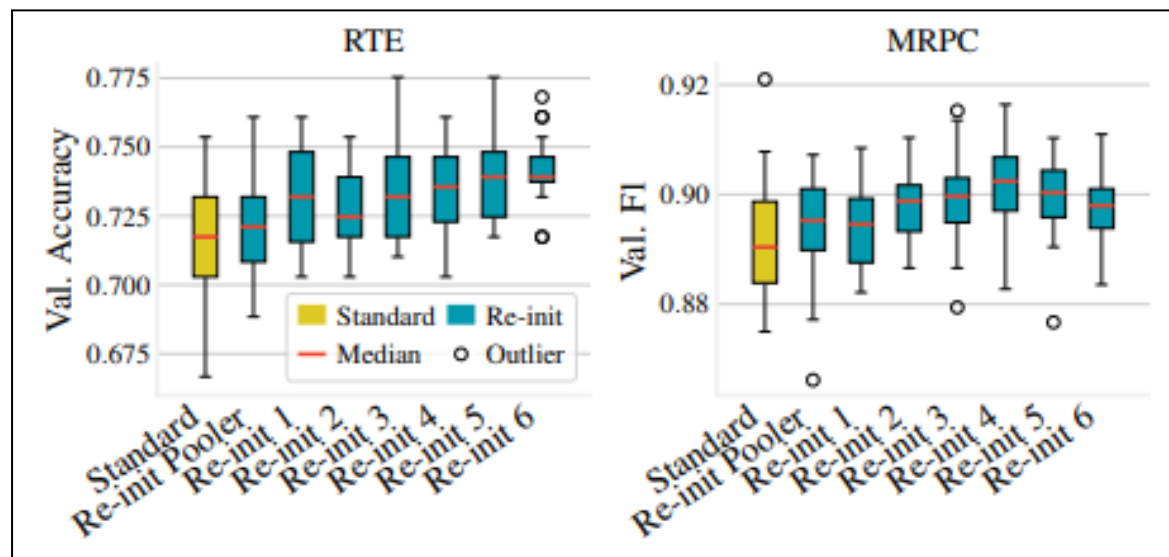
➔ few-sample fine-tuning에서는 debiasing Adam이 안정적

5. Initialization: Re-initializing BERT Pre-trained Layers

Re-initializing BERT Pre-trained Layers

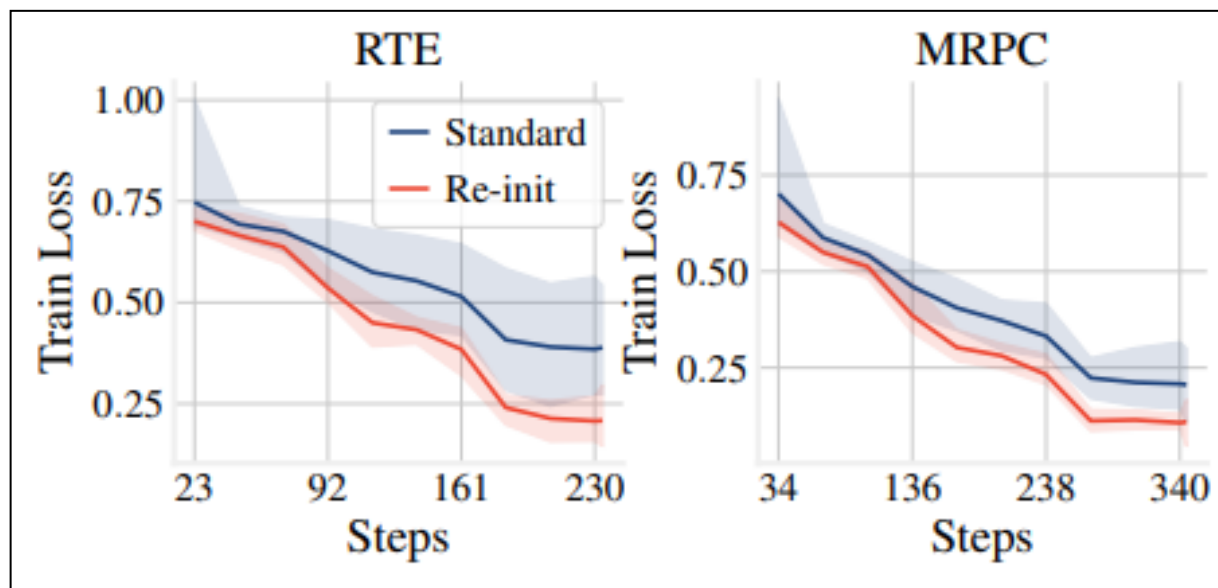
- 일반적으로 BERT fine-tuning은 pre-train된 가중치를 사용해 모든 레이어를 초기화 시킴
- Pre-train된 하위 레이어: 일반적인 피쳐 학습
- Pre-train된 상위 레이어: 좀 더 전문화된 피쳐 학습
- 본 논문에서는 pre-train된 가중치를 모두 사용하는 것과 일부만 사용하는 것으로 나누어 비교 실험
 - BERT에서 사용한 방식이 가장 효과적인 방법이 아님을 증명하기 위함
- Re-init
 - 상위 L개의 레이어를 재초기화함
 - $N(0, 0.02^2)$

Re-initializing BERT Pre-trained Layers



- Re-init이 모델의 성능에 끼친 영향
- Re-init된 레이어 수에 따른 민감도

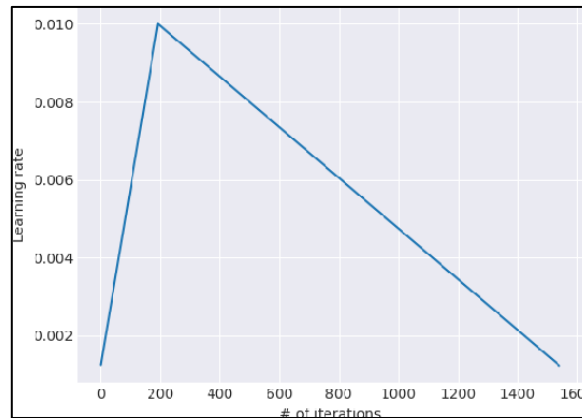
- 수렴과 파라미터 변경에 대한 영향



6. Training Iterations: Fine-tuning BERT for Longer

Fine-tuning BERT for Longer

- 일반적으로 BERT는 기울어진 삼각형 학습률로 fine-tuning을 진행
- GLUE를 fine-tuning 하는데 있어 가장 좋은 3 에폭
- BERT fine-tuning을 좀 더 길게 하면 안정성과 성능 모두 향상



- Setup
 - 8가지 데이터셋들에 대한 fine-tuning iteration을 증가 시켜보며 그 효과를 연구
 - 3 에폭이 96 스텝에 해당하는 1,000개의 다운샘플링된 데이터셋의 경우 iteration 횟수를 {200, 400, 800, 1600, 3200}으로 조정

Fine-tuning BERT for Longer

- 대부분 3 에폭 설정의 학습보다 더 오래 학습했을 때, 성능이 향상
- 1,000개로 다운샘플링된 데이터셋에서 더 두드러짐

Dataset	RTE		MRPC		STS-B		CoLA	
	3 Epochs	Longer	3 Epochs	Longer	3 Epochs	Longer	3 Epochs	Longer
Standard	69.5 ± 2.5	72.3 ± 1.9	90.8 ± 1.3	90.5 ± 1.5	89.0 ± 0.6	89.6 ± 0.3	63.0 ± 1.5	62.4 ± 1.7
Re-init	72.6 ± 1.6	73.1 ± 1.3	91.4 ± 0.8	91.0 ± 0.4	89.4 ± 0.2	89.9 ± 0.1	63.9 ± 1.9	61.9 ± 2.3

Dataset	RTE (1k)		MRPC (1k)		STS-B (1k)		CoLA (1k)	
	3 Epochs	Longer	3 Epochs	Longer	3 Epochs	Longer	3 Epochs	Longer
Standard	62.5 ± 2.8	65.2 ± 2.1	80.5 ± 3.3	83.8 ± 2.1	84.7 ± 1.4	88.0 ± 0.4	45.9 ± 1.6	48.8 ± 1.4
Re-init	65.6 ± 2.0	65.8 ± 1.7	84.6 ± 1.6	86.0 ± 1.2	87.2 ± 0.4	88.4 ± 0.2	47.6 ± 1.8	48.4 ± 2.1

Dataset	SST (1k)		QNLI (1k)		QQP (1k)		MNLI (1k)	
	3 Epochs	Longer	3 Epochs	Longer	3 Epochs	Longer	3 Epochs	Longer
Standard	89.7 ± 1.5	90.9 ± 0.5	78.6 ± 2.0	81.4 ± 0.9	74.0 ± 2.7	77.4 ± 0.8	52.2 ± 4.2	67.5 ± 1.1
Re-init	90.8 ± 0.4	91.2 ± 0.5	81.9 ± 0.5	82.1 ± 0.3	77.2 ± 0.7	77.6 ± 0.6	66.4 ± 0.6	68.8 ± 0.5

7. Conclusion

Conclusion

Few-sample BERT fine-tuning이 안정적이기 위한 3가지

1. Debiasing Omission in BERTAdam

- BERTAdam의 편향 보정 제거가 성능 저하 원인!
- Debaised Adam을 사용하는 것이 더욱 효과적

2. Re-initializing BERT Pre-trained Layers

- Pre-train된 상위 레이어가 학습 딜레이 등 fine-tuning에 나쁜 영향을 줌!
- 이러한 레이어를 다시 초기화하여 성능 향상 및 학습 속도 up

3. Fine-tuning BERT for Longer

- 더 많은 iteration이면 fine-tuning이 안정화!