ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

임근태, 윤주환

July, 22, 2024

Abstract

ABSTRACT

Increasing model size when pretraining natural language representations often results in improved performance on downstream tasks. However, at some point further model increases become harder due to GPU/TPU memory limitations and longer training times. To address these problems, we present two parameter-reduction techniques to lower memory consumption and increase the training speed of BERT (Devlin et al., 2019). Comprehensive empirical evidence shows that our proposed methods lead to models that scale much better compared to the original BERT. We also use a self-supervised loss that focuses on modeling inter-sentence coherence, and show it consistently helps downstream tasks with multi-sentence inputs. As a result, our best model establishes new state-of-the-art results on the GLUE, RACE, and SQuAD benchmarks while having fewer parameters compared to BERT-large. The code and the pretrained models are available at https://github.com/google-research/ALBERT.

배경 (Background)

모델 크기를 증가시키면 자연어 표현을 사전 학습할 때
다운스트림 작업 과정에서 성능이 향상되는 경우가 종종 있음

그러나, **GPU/TPU 메모리 제한 & 학습 시간 증가로 인해**
일정 시점 이후에는 모델 크기를 더 늘리는 것에 대한 한계 존재.

Abstract

ABSTRACT

Increasing model size when pretraining natural language representations often results in improved performance on downstream tasks. However, at some point further model increases become harder due to GPU/TPU memory limitations and longer training times. To address these problems, we present two parameter-reduction techniques to lower memory consumption and increase the training speed of BERT (Devlin et al., 2019). Comprehensive empirical evidence shows that our proposed methods lead to models that scale much better compared to the original BERT. We also use a self-supervised loss that focuses on modeling inter-sentence coherence, and show it consistently helps downstream tasks with multi-sentence inputs. As a result, our best model establishes new state-of-the-art results on the GLUE, RACE, and SQuAD benchmarks while having fewer parameters compared to BERT-large. The code and the pretrained models are available at `https://github.com/google-research/ALBERT`.

방법 (Methods)

이러한 문제를 해결하기 위해, 본 논문에서는 **BERT(Devlin et al., 2019)의 메모리 소비를 줄이고 학습 속도를 높이기 위한 두 가지 매개변수 축소 기법을 제안하고,**

문장 간 일관성을 모델링하는 데 중점을 둔 **자가 지도 손실(self-supervised loss)을 사용하여 다중 문장 입력이 있는 다운스트림 작업에서 일관되게 도움이 된다는 것을 보여주었음**

Abstract

ABSTRACT

Increasing model size when pretraining natural language representations often results in improved performance on downstream tasks. However, at some point further model increases become harder due to GPU/TPU memory limitations and longer training times. To address these problems, we present two parameter-reduction techniques to lower memory consumption and increase the training speed of BERT (Devlin et al., 2019). Comprehensive empirical evidence shows that our proposed methods lead to models that scale much better compared to the original BERT. We also use a self-supervised loss that focuses on modeling inter-sentence coherence, and show it consistently helps downstream tasks with multi-sentence inputs. As a result, our best model establishes new state-of-the-art results on the GLUE, RACE, and SQuAD benchmarks while having fewer parameters compared to BERT-large. The code and the pretrained models are available at `https://github.com/google-research/ALBERT`.

결과 (Results)

그 결과, **본 논문에서 제안한 방법들이 기존 BERT model에 비해 훨씬 더 잘 확장되는 모델로 이어진다는 것을 검증**하였으며,

결과적으로, **BERT-large보다 적은 매개변수로** GLUE, RACE 및 SQuAD 벤치마크에서 새로운 SOTA 결과를 수립함

Introduction

➢ 기존 연구의 한계점 : Pre-training 시 모델의 크기가 커지면 성능이 향상되지만, 메모리 제한 & 연산 시간 증가

1) Memory Limitation

Model		Parameters
BERT	base	108M
	large	334M

System	Seq Length	Max Batch Size
BERT-Base	64	64
...	128	32
...	256	16
...	320	14
...	384	12
...	512	6
BERT-Large	64	12
...	128	6
...	256	2
...	320	1
...	384	0
...	512	0

12GB GPU 기준, OOM이 발생하지 않기 위한 Input Sequence Length에 따른 Maximum Batch Size

2) Training Time

- BERT Base
  - L = 12, H = 768, A = 12, Total Parameters = 110M
  - 16 TPU chips 활용 -> 4일
  - 16 GPU V100 활용 -> 5일 이상
- BERT Large
  - L = 24, H = 1024, A = 16, Total Parameters = 340M
  - 64 TPU chips 활용 -> 4일
  - 64 GPU V100 활용 -> 8일 이상

→ GPT3 or BERT 등의 모델은 수억 개에 달하는 파라미터를 사용하기 때문에 OOM(Out-Of Memory) 문제를 자주 직면함.

→ 분산 학습을 통해 해결하려고 해도, 분산된 장비 간의 통신과정에서 학습속도가 저하되기 때문에, 문제가 있음.

Introduction

➢ ALBERT의 문제 접근

1) Memory Limitation

Model		Parameters
BERT	base	108M
	large	334M

System	Seq Length	Max Batch Size
BERT-Base	64	64
...	128	32
...	256	16
...	320	14
...	384	12
...	512	6
BERT-Large	64	12
...	128	6
...	256	2
...	320	1
...	384	0
...	512	0

12GB GPU 기준, OOM이 발생하지 않기 위한 Input Sequence Length에 따른 Maximum Batch Size

2) Training Time

- BERT Base
  - L = 12, H = 768, A = 12, Total Parameters = 110M
  - 16 TPU chips 활용 -> 4일
  - 16 GPU V100 활용 -> 5일 이상
- BERT Large
  - L = 24, H = 1024, A = 16, Total Parameters = 340M
  - 64 TPU chips 활용 -> 4일
  - 64 GPU V100 활용 -> 8일 이상

✓ Factorized embedding parameterization

✓ Cross-Layer Parameter Sharing

✓ Sentence Order Prediction(SOP)

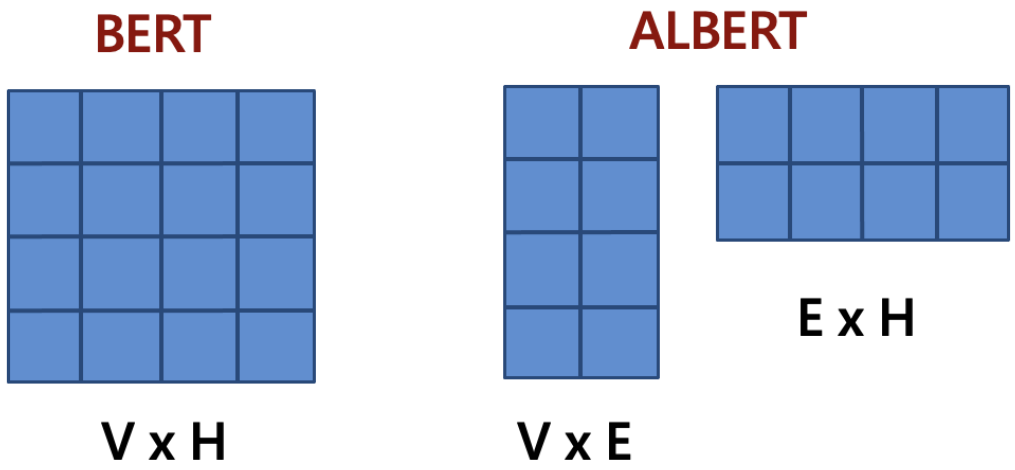
✓ 전체 Parameter 수 감소

✓ Parameter 효율 개선

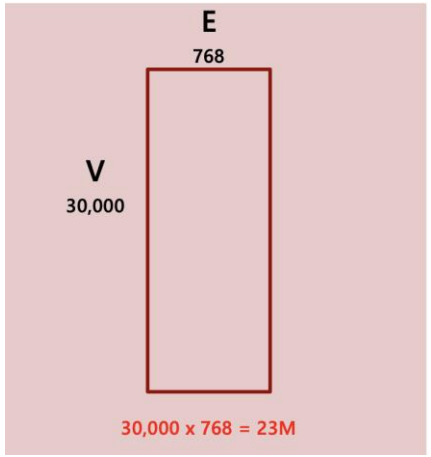
✓ 학습 안정화

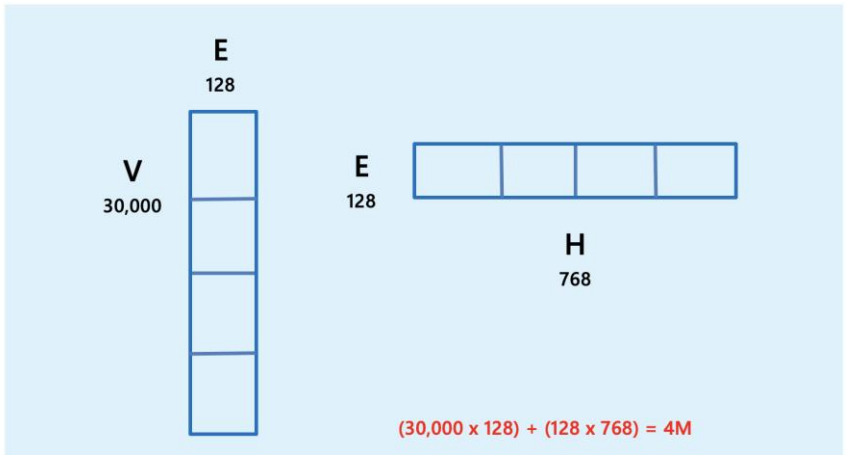
The Elements of ALBERT

➢ Factorized embedding parameterization



- V : Vocabulary size, 모델의 어휘력
- H : Hidden state dimension, 모델 내부의 벡터
- E : Word embedding dimension, 단어의 초기 표현

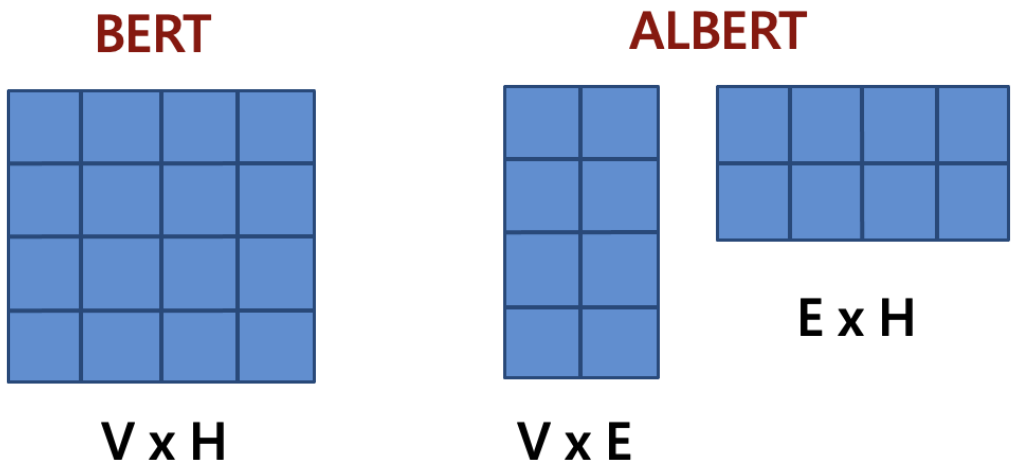




https://velog.io/@9e0na/%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%B0-NLP-ALBERT-A-Lite-BERT-for-Self-supervised-Learning-of-Language-Representations-2020-Summary

The Elements of ALBERT

➢ Factorized embedding parameterization



- V : Vocabulary size, 모델의 어휘력
- H : Hidden state dimension, 모델 내부의 벡터
- E : Word embedding dimension, 단어의 초기 표현

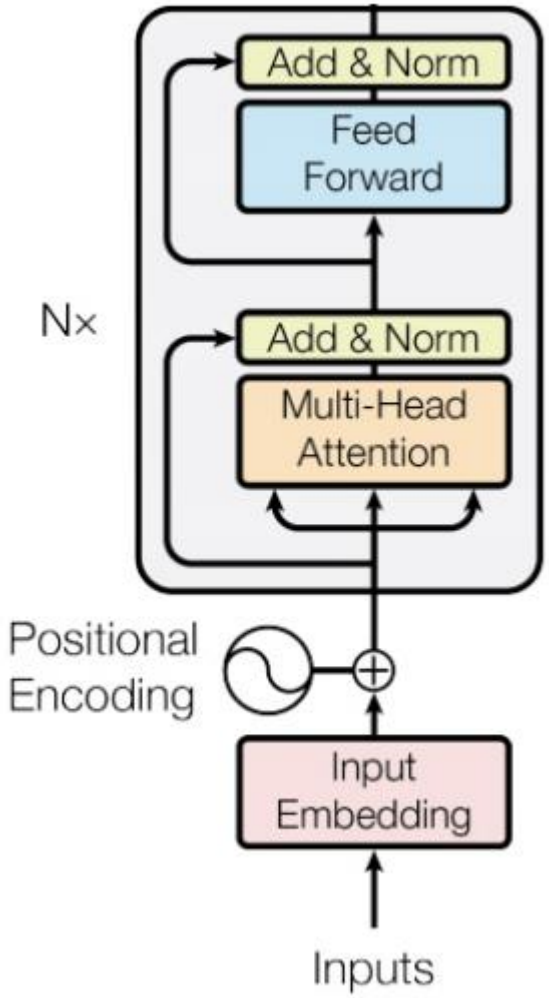
Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared (BERT)	64 (1/12)	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128 (1/6)	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256 (1/3)	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared (ALBERT)	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

Table 3: The effect of vocabulary embedding size on the performance of ALBERT-base.

https://velog.io/@9e0na/%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%B0-NLP-ALBERT-A-Lite-BERT-for-Self-supervised-Learning-of-Language-Representations-2020-Summary

The Elements of ALBERT

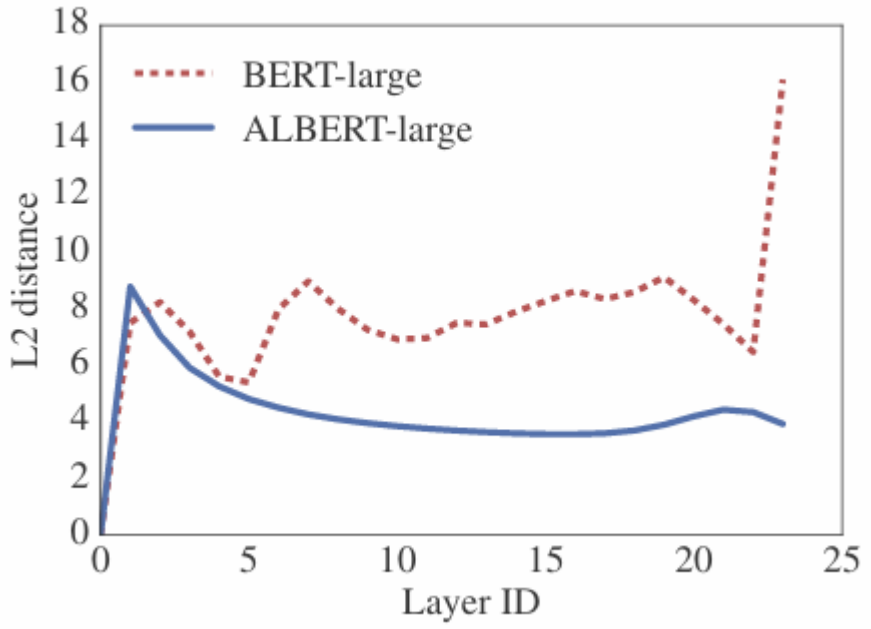
- **Cross-Layer Parameter Sharing**

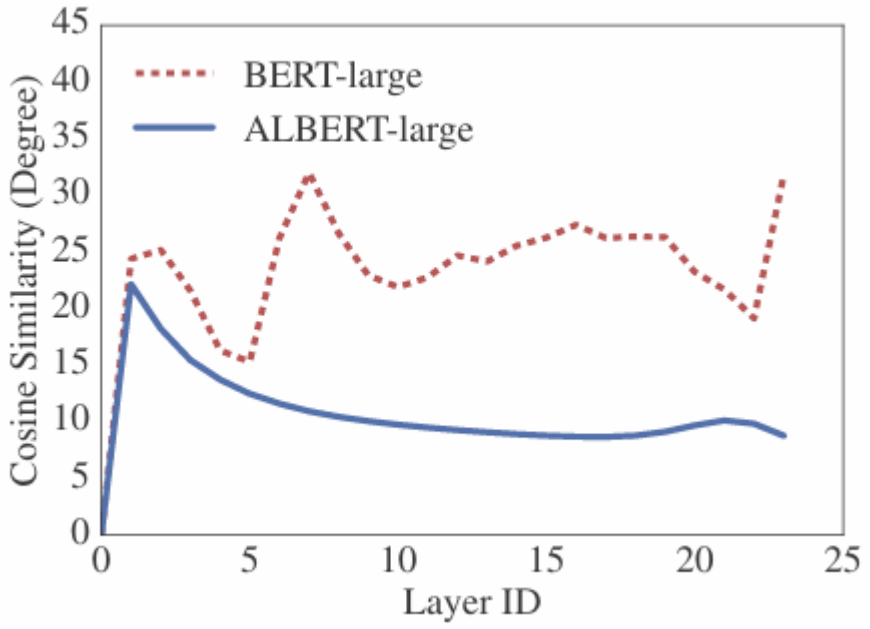


- **Attention Parameter Sharing**
- **Feed-Forward Parameter Sharing**
- **ALBERT : All Parameter Sharing**

The Elements of ALBERT

- **Cross-Layer Parameter Sharing**





- **BERT와 ALBERT에서 각 layer의 input과 output embedding의 L2 distance와 cosine similarity를 비교**
- **ALBERT의 경우 Layer ID에 따른 그래프 변화가 크지 않음**
- **여러 Layer들이 균등하게 영향을 미치기 때문에 안정적인 학습 가능**

The Elements of ALBERT

➢ Cross-Layer Parameter Sharing

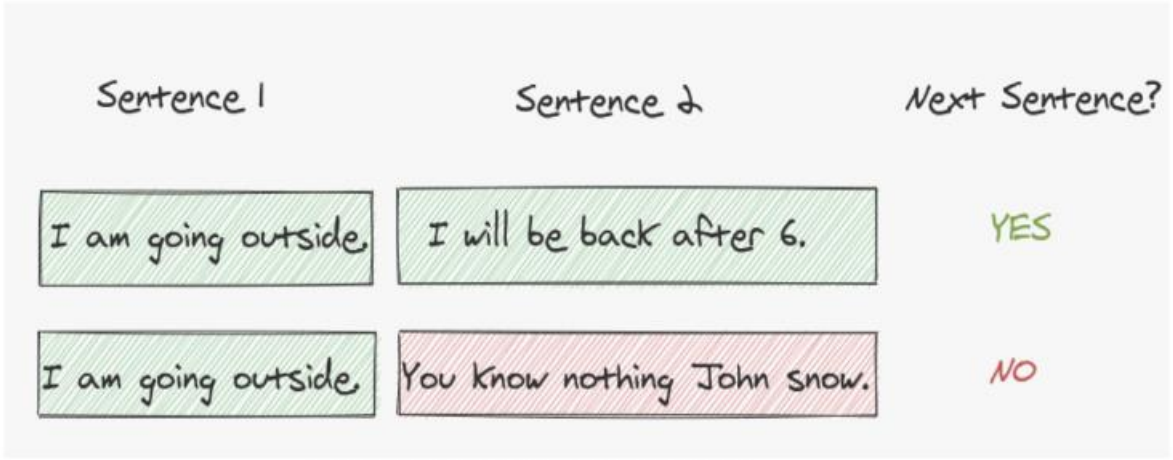
	Model	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared (BERT)	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared (BERT)	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

Table 4: The effect of cross-layer parameter-sharing strategies, ALBERT-base configuration.

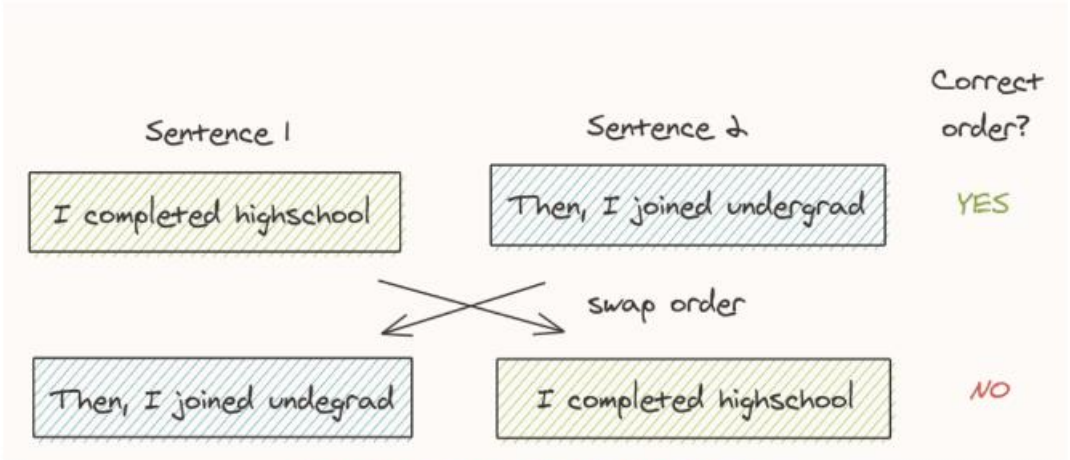
- **shared-attention의 경우 큰 성능 저하가 발생하지 않음**
- **shared-FFN에서 성능 저하가 크게 발생**
- **Embedding size가 E=128인 경우가 성능 저하가 비교적 덜함**

The Elements of ALBERT

- **Sentence Order Prediction(SOP)**



Next Sentence Prediction(NSP)



Sentence Order Prediction(SOP)

SP tasks	Intrinsic Tasks MLM	NSP	SOP	Downstream Tasks SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
(XLNet, RoBERTa) None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
(BERT) NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
(ALBERT) SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

Table 5: The effect of sentence-prediction loss, NSP vs. SOP, on intrinsic and downstream tasks.

Experimental Results

- **Comparison between BERT and ALBERT**

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Table 1: The configurations of the main BERT and ALBERT models analyzed in this paper.

Experimental Results

➢ Comparison between BERT and ALBERT

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

Table 2: Dev set results for models pretrained over BOOKCORPUS and Wikipedia for 125k steps. Here and everywhere else, the Avg column is computed by averaging the scores of the downstream tasks to its left (the two numbers of F1 and EM for each SQuAD are first averaged).

- **ALBERT-xxlarge는 BERT-large의 70%의 매개변수만 사용하고도 더 좋은 성능을 달성**
- **ALBERT-large는 BERT-large보다 데이터 처리 속도가 1.7배 빠르며 ALBERT-xxlarge는 3배 더 느림**
- **동일한 레이어 수와 hidden 크기에서 ALBERT 모델의 크기가 훨씬 작으며, 학습 속도가 훨씬 빠름**

Experimental Results

➢ Factorized Embedding Parameterization

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared (BERT)	64 (1/12)	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128 (1/6)	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256 (1/3)	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared (ALBERT)	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

Table 3: The effect of vocabulary embedding size on the performance of ALBERT-base.

- **BERT style에서는 Embedding size가 클수록 모델 성능이 증가하지만, 상승폭은 미미함**
- **ALBERT-style에서는 E=128에서 가장 좋은 결과를 보여줌.**

Experimental Results

➢ Cross-Layer Parameter Sharing

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared (BERT)	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared (BERT)	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

Table 4: The effect of cross-layer parameter-sharing strategies, ALBERT-base configuration.

- all-shared와 shared-FFN에서 성능이 제일 떨어짐
- 그러나 parameter가 감소되는 폭에 비해 성능이 많이 떨어지지 않기에 all-shared 사용

Experimental Results

- **Sentence Order Prediction(SOP)**

		Intrinsic Tasks			Downstream Tasks					
	SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
(XLNet, RoBERTa)	None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
(BERT)	NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
(ALBERT)	SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

Table 5: The effect of sentence-prediction loss, NSP vs. SOP, on intrinsic and downstream tasks.

- **NSP : SOP 해결 X → Modeling Only Topic Shift**
- **SOP : NSP 해결 O, Multi-Sentence Encoding Task에 대해 성능 개선을 보임**

Experimental Results

- **Train with same amount of time**

Models	Steps	Time	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
BERT-large	400k	34h	93.5/87.4	86.9/84.3	87.8	94.6	77.3	87.2
ALBERT-xxlarge	125k	32h	**94.0/88.1**	**88.3/85.3**	87.8	**95.4**	**82.5**	**88.7**

Table 6: The effect of controlling for training time, BERT-large vs ALBERT-xxlarge configurations.

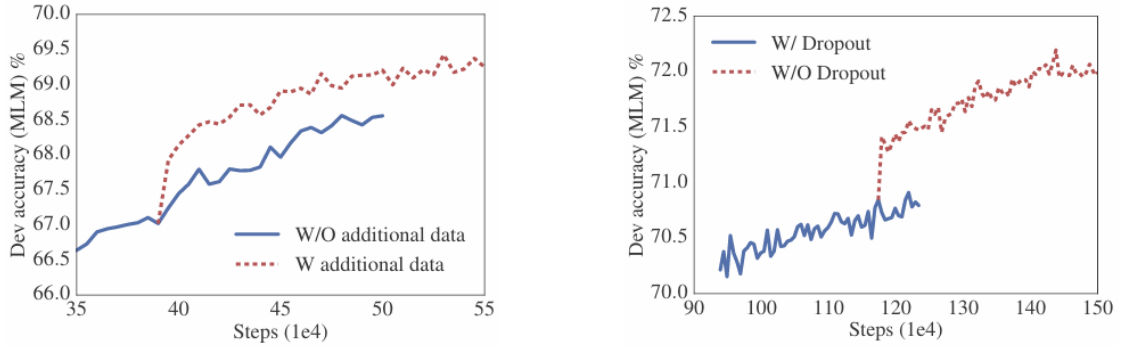
- **동일한 시간동안 학습시켰을 때 BERT-large와 ALBERT-xxlarge의 성능을 비교**
- **BERT-large : 400k training steps & 34h**
- **ALBERT-xxlarge : 125k training step & 32h**

Experimental Results

➢ Additional training data and dropout effects

	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
No additional data	**89.3/82.3**	**80.0/77.1**	81.6	90.3	64.0	80.1
With additional data	88.8/81.7	79.1/76.3	**82.4**	**92.8**	**66.0**	**80.8**

Table 7: The effect of additional training data using the ALBERT-base configuration.



(a) Adding data (b) Removing dropout

Figure 2: The effects of adding data and removing dropout during training.

	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
With dropout	94.7/89.2	89.6/86.9	90.0	96.3	85.7	90.4
Without dropout	**94.8/89.5**	**89.9/87.2**	**90.4**	**96.5**	**86.1**	**90.7**

Table 8: The effect of removing dropout, measured for an ALBERT-xxlarge configuration.

➢ **Additional Training Data**

- ✓ **Data를 추가시켰을 때 성능이 더 좋음**
- ✓ **Wikipedia 기반의 SQuAD는 성능 하락**

➢ **Dropout Effects**

- ✓ **ALBERT를 1M steps 이상 학습시켰음에도 Overfitting X**
- ✓ **Model의 Capacity를 늘리기 위해 Dropout 제거 후 학습**
- ✓ **Dropout을 제거했을 때 성능이 향상**

Conclusion

➢ Current State-Of-The-Art on NLU Tasks

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

Table 9: State-of-the-art results on the GLUE benchmark. For single-task single-model results, we report ALBERT at 1M steps (comparable to RoBERTa) and at 1.5M steps. The ALBERT ensemble uses models trained with 1M, 1.5M, and other numbers of steps.

Models	SQuAD1.1 dev	SQuAD2.0 dev	SQuAD2.0 test	RACE test (Middle/High)
Single model (from leaderboard as of Sept. 23, 2019)				
BERT-large	90.9/84.1	81.8/79.0	89.1/86.3	72.0 (76.6/70.1)
XLNet	94.5/89.0	88.8/86.1	89.1/86.3	81.8 (85.5/80.2)
RoBERTa	94.6/88.9	89.4/86.5	89.8/86.8	83.2 (86.5/81.3)
UPM	-	-	89.9/87.2	-
XLNet + SG-Net Verifier++	-	-	90.1/87.2	-
ALBERT (1M)	94.8/89.2	89.9/87.2	-	86.0 (88.2/85.1)
ALBERT (1.5M)	**94.8/89.3**	**90.2/87.4**	**90.9/88.1**	**86.5 (89.0/85.5)**
Ensembles (from leaderboard as of Sept. 23, 2019)				
BERT-large	92.2/86.2	-	-	-
XLNet + SG-Net Verifier	-	-	90.7/88.2	-
UPM	-	-	90.7/88.2	
XLNet + DAAF + Verifier	-	-	90.9/88.6	-
DCMN+	-	-	-	84.1 (88.5/82.3)
ALBERT	**95.5/90.1**	**91.4/88.9**	**92.2/89.7**	**89.4 (91.2/88.6)**

Table 10: State-of-the-art results on the SQuAD and RACE benchmarks.

- ALBERT는 기존 NLP Task에서 모두 SOTA와 동일하거나 더 높은 성능 달성
- 기존 Model 대비 훨씬 적은 Training Time과 Parameter로 좋은 성능 달성
- 모델을 경량화 함으로써, 모델 크기 증가를 막는 Memory Limitation을 극복