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

Sewon Min[1], **Danqi Chen**[2], **Luke Zettlemoyer**[1,3], **Hannaneh Hajishirzi**[1,4]
[1]University of Washington, Seattle, WA [2]Princeton University, Princeton, NJ
[3]Facebook AI Research, Seattle, WA [4]Allen Institute for AI, Seattle, WA
{sewon,lsz,hannaneh}@cs.washington.edu danqic@cs.princeton.edu

KISTI-UST
JUYEON YU

Contents

1. Introduction

2. Related Work

3. Model

4. Experiments

5. Results

6. Conclusion

01 Introduction

Outline

- Increasing model size in pre-training results in improved performance
- Probelms
  - GPU/TPU memory limitations
  - Longer training times
  - Model Degradation
- Solve: two parameter reduction techniques
  - Lower memory consumption and
  - Increase the training speed of BERT
- A Lite BERT (ALBERT) architecture that has significantly fewer parameters than a traditional BERT architecture.
- Much better performance compared to the original BERT

02 Related Work

- Scaling up representation learning
  - Shift from pre-training word embeddings, or contextualized to full-network pre-training followed by task-specific fine-tuning
  - Larger model size improves proves performance (they stop at a hidden size of 1024)
  - Gradient checkpointing to reduce the memory requirement (Chen et al., 2016)
  - Model parallelization to train a giant model (Raffel et al., 2019)
- Cross-layer parameter sharing
  - Explored with the Transformer architecture
    - focused on training for standard encoder-decoder tasks rather than the pretraining/finetuning setting
  - Networks with cross-layer parameter sharing get better performance (Dehghani et al., 2018)
  - Deep Equilibrium Model for transformer networks (Bai et al., 2019)

These solutions address the memory limitation problem,

but not the communication overhead

03 Model

Model Architecture Choices

- The backbone of the ALBERT architecture is similar to BERT
  - Transformer encoder
  - GELU

03 Model

Model Architecture Choices

- Two parameter reduction techniques
  - Factorized embedding parameterization
    - Decomposing the large vocabulary embedding matrix into two small matrices
    - Separate the size of the hidden layers from the sizer of vocabulary embedding

    -> grow the hidden size without significantly increasing the parameter size of the vocabulary embeddings
  - Cross-layer parameter sharing
    - Prevents the parameter from growing with the depth of the network

    -> Both techniques significantly reduce the number of parameters
- Self-supervised loss for sentence-order prediction (SOP)
  - Predicting the ordering of two consecutive segments
  - Focuses on inter-sentence coherence

03 Model

Factorized embedding parameterization

- WordPiece embeddings are meant to learn context-independent representations

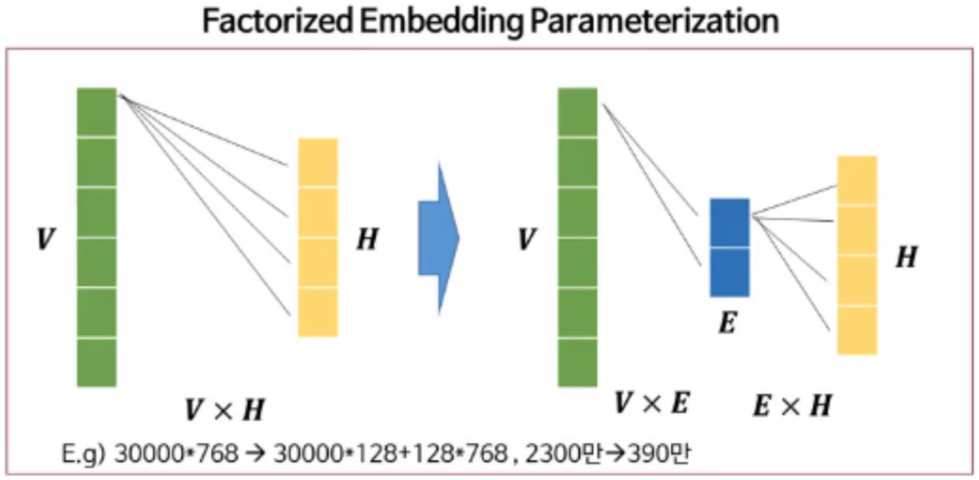
whereas hidden-layer embeddings are meant to learn context-dependent representations.

  - Untying the WordPiece embedding size E from the hidden layer size H
  - H >> E

- Project one-hot vectors directly into the hidden space of size H?
- Project one-hot vectors -> a lower dim. embedding space of size E

-> project it to the hidden space

$$O(V \times H) \text{ vs. } O(V \times E + E \times H)$$



* Notation

E : the vocabulary embedding size (Context Independent Representation)

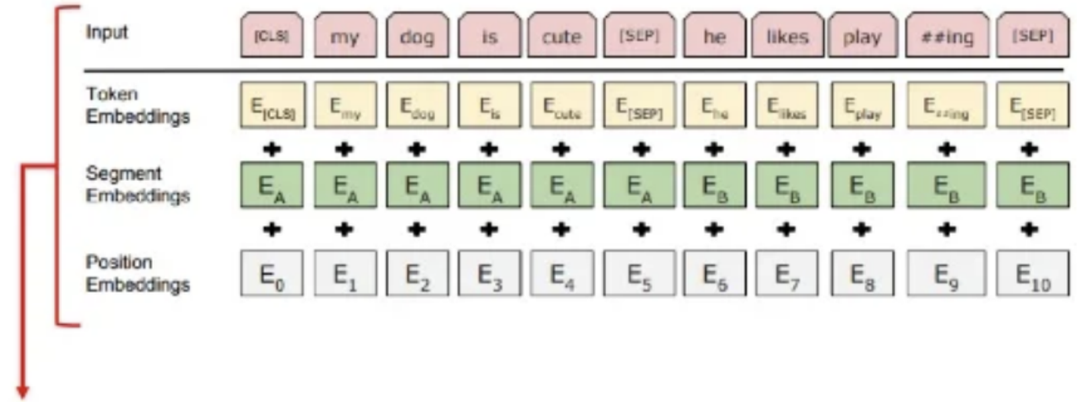
V : the vocabulary size

H : the hidden size (Contextualized Representation)

03 Model

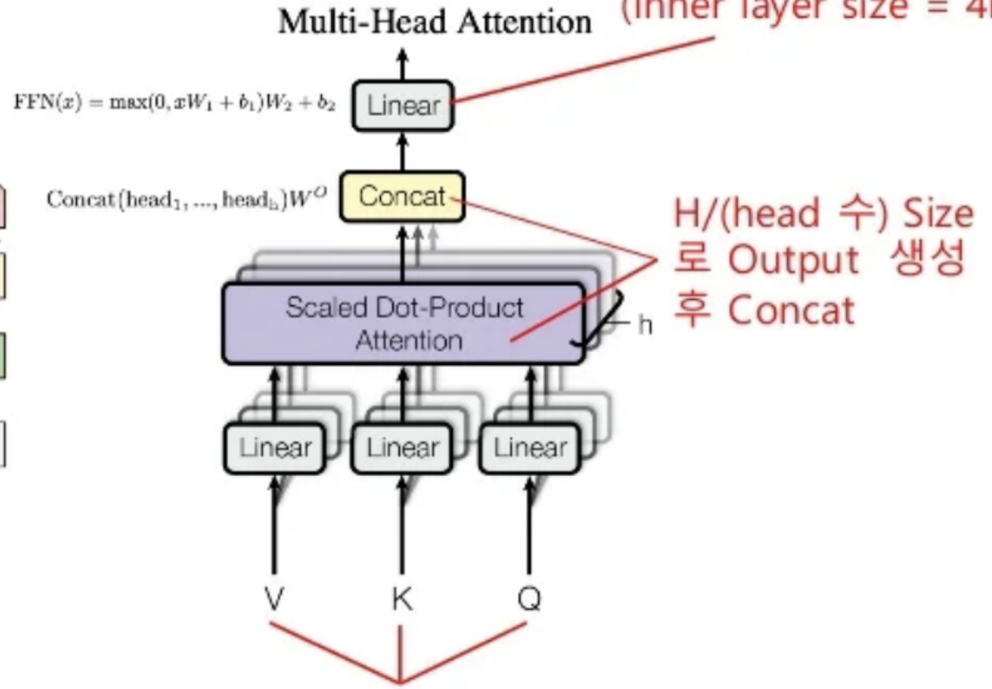
Factorized embedding parameterization

- BERT / Transformer Embedding Size Review



각 Embedding Size가
H와 같음

Input/output Size = H
(Inner layer size = 4H)



H/(head 수) Size
로 Output 생성
후 Concat

03 Model

Cross-layer parameter sharing

- Previous researches
  - Only sharing feed-forward network (FFN) parameters across laters, or
  - Only sharing attention parameters
- ALBERT share all parameters across layers
- Transitions from layer to layer are much smoother for ALBERT than for BERT

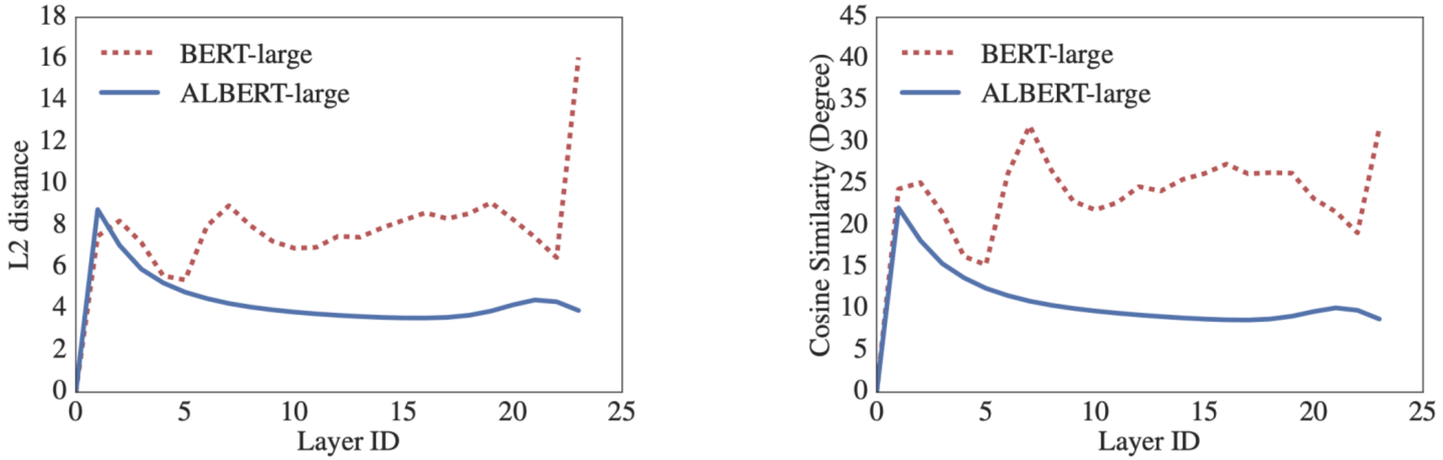
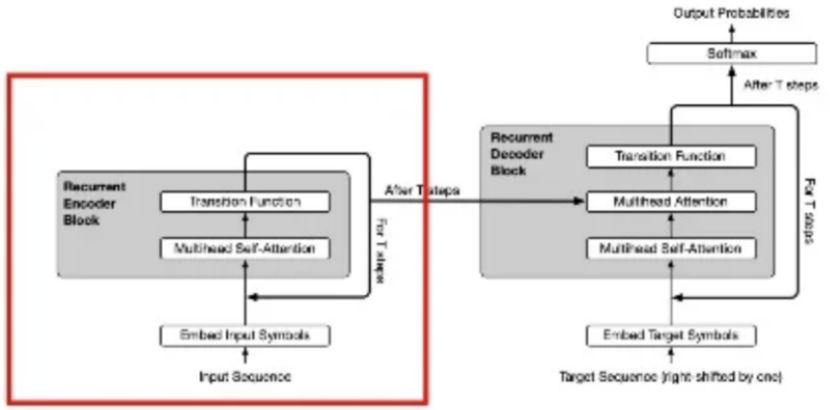


Figure 1: The L2 distances and cosine similarity (in terms of degree) of the input and output embedding of each layer for BERT-large and ALBERT-large.

03 Model

Cross-layer parameter sharing

- **Similar Idea**
  - Deep Equilibrium Models
    - https://arxiv.org/abs/1909.01377
  - Universal Transformer
    - https://arxiv.org/abs/1807.03819
    - [Intuition] RNN이 Sequence Data를 활용하여 Hidden State를 Iterative Update 하는 것처럼 Recurrent Layer를 통해 Output을 Iterative Update(Recursive Transformation)
    - RNNs' inductive bias towards learning iterative
    - Machine Translation , bAbI (NLU Task) 등에서 Transformer 보다 높은 성능



03 Model

Inter-sentence coherence loss

- BERT use the masked language modeling(MLM) loss and next-sentence prediction(NSP)
  - NSP is a binary classification loss for predicting whether two segments appear consecutively in the original text.
- However, subsequent studies found NSP's impact unreliable and decided to eliminate it
- ALBERT use a sentence-order prediction (SOP) loss
  - SOP avoids topic prediction and instead focuses on modeling inter-sentence coherence
  - Positive examples: two consecutive segments from the same document
  - Negative examples: the same two consecutive segments but with their order swapped

04 Experiments

Setup

- Book Corpus, Wikipedia - 16GB (BERT)
- Batch Size = 4096
- Train all models for 125,000 Steps
- TPU V3
  - The number of TPUs used for training ranged from 64 to 512
- Dataset
  - GLUE 9 tasks:

    The General Language Understanding Evaluation
  - SQUAD 1.1, 2.0:

    The Stanford Question Answering Dataset
  - RACE:

    the ReAding Cimprehension from Examinations

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.

04 Experiments

Overall comparison between BERT and ALBERT

- ALBERT-xxlarge achieves significant improvements over BERT-large

(around 70% of BERT-large's parameters)

- The speed of data throughput at training time under the same training configuration

(same number of TPUs)

  - Because of less communication and fewer computations
  - ALBERT-large is about 1.7 times faster
  - ALBERT-xxlarge is about 3 times slower because of the larger structure

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).

04 Experiments

Factorized embedding parameterization

- The effect of changing the vocabulary embedding size E using an ALBERT-base
- The speed of data throughput at training time under the same training configuration

(same number of TPUs)

  - Because of less communication and fewer computations
  - ALBERT-large is about 1.7 times faster
  - ALBERT-xxlarge is about 3 times slower because of the larger structure
- Under the non-shared condition (BERT-style), larger embedding sizes give better performance, but not by much.
  - Under the all-shared condition (ALBERT-style), an embedding of size 128 appears to be the best.
  - -> embedding size E = 128

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	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	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.

04 Experiments

Cross-layer parameter sharing

- The not-shared strategy (BERT-style)
- Only the attention parameters are shared (but not the FNN ones)
- Only the FFN parameters are shared (but not the attention ones)

- The performance drop appears to come from sharing the FFN-layer parameters

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	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	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.

04 Experiments

Cross-layer parameter sharing

- The transitions from layer to layer are much smoother for ALBERT than for BERT
- Weight-sharing has an effect on stabilizing network parameters.
- Although there is a drop for both metrics compared to BERT, they nevertheless do not converge to 0

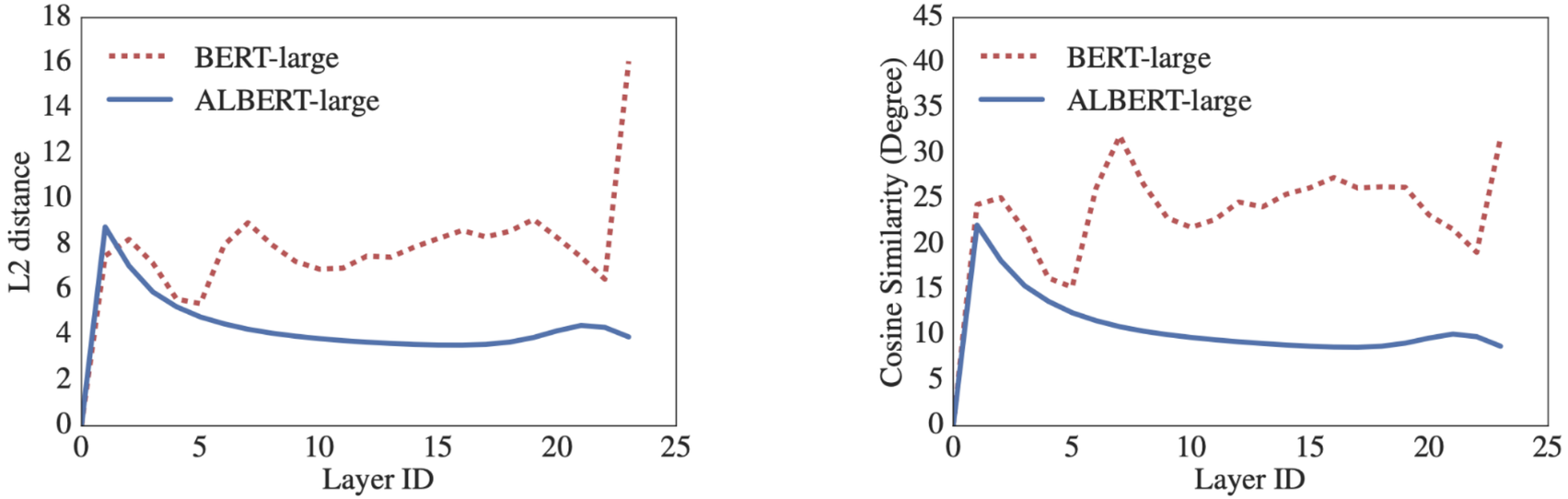


Figure 1: The L2 distances and cosine similarity (in terms of degree) of the input and output embedding of each layer for BERT-large and ALBERT-large.

04 Experiments

Inter-sentence coherence loss

- None (XLNet- and RoBERTa-style)
- NSP (BERT-style)
- SOP (ALBERT-style), using an ALBERTbase configuration.

	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
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.

04 Experiments

Parameter size: BERT vs. 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.

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).

Number of layers	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
1	18M	31.1/22.9	50.1/50.1	66.4	80.8	40.1	52.9
3	18M	79.8/69.7	64.4/61.7	77.7	86.7	54.0	71.2
6	18M	86.4/78.4	73.8/71.1	81.2	88.9	60.9	77.2
12	18M	89.8/83.3	80.7/77.9	83.3	91.7	66.7	81.5
24	18M	**90.3/83.3**	**81.8/79.0**	83.3	91.5	**68.7**	**82.1**
48	18M	90.0/83.1	**81.8/78.9**	**83.4**	**91.9**	66.9	81.8

Table 11: The effect of increasing the number of layers for an ALBERT-large configuration.

Hidden size	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
1024	18M	79.8/69.7	64.4/61.7	77.7	86.7	54.0	71.2
2048	60M	83.3/74.1	69.1/66.6	79.7	88.6	58.2	74.6
4096	225M	**85.0/76.4**	**71.0/68.1**	**80.3**	**90.4**	**60.4**	**76.3**
6144	499M	84.7/75.8	67.8/65.4	78.1	89.1	56.0	74.0

Table 12: The effect of increasing the hidden-layer size for an ALBERT-large 3-layer configuration.

04 Experiments

What if we train for the 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**

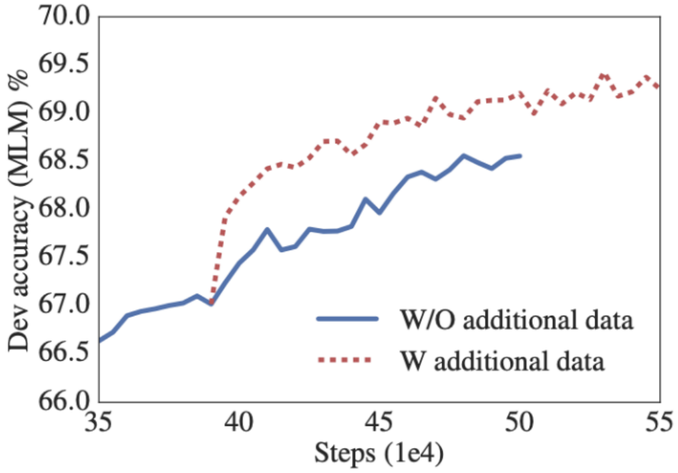
Table 6: The effect of controlling for training time, BERT-large vs ALBERT-xxlarge configurations.

04 Experiments

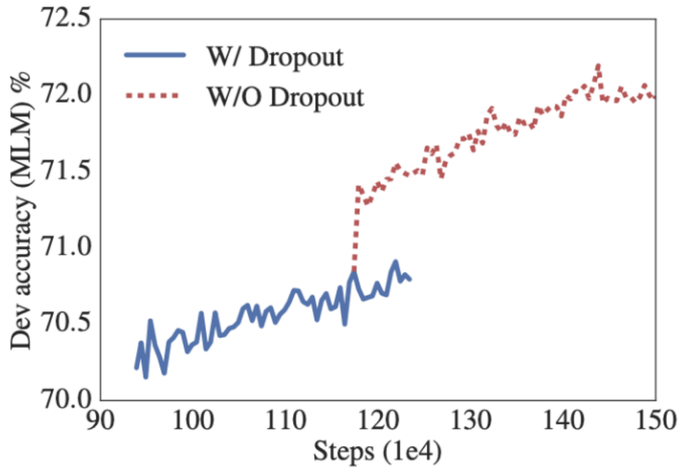
Additional training data and dropout effects

	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.

05 Conclusion

ALBERT Innovations

1. Parameter Sharing

2. Embedding Factorization

3. Sentence Order Prediction (SOP)

4. N-gram Masking

5. LAMB Optimizer

6. SentencePiece Model

References

1. https://www.youtube.com/watch?v=f2__p05aY2I&ab_channel=DabinMin

2. https://y-rok.github.io/nlp/2019/10/23/albert.html

3. https://www.youtube.com/watch?v=pLCvIceRrgI&ab_channel=%E2%80%8D%EA%B0%95%EC%8A%B9%EC%8B%9D%28%EA%B5%90%EC%9B%90-%EC%9D%B8%EA%B3%B5%EC%A7%80%EB%8A%A5%EC%A0%84%EA%B3%B5%29

4. https://www.youtube.com/watch?v=f2__p05aY2I&ab_channel=DabinMin

5. https://www.youtube.com/watch?v=vsGN8WqwvKg&ab_channel=ChrisMcCormickAI

THANK YOU

Q & A