ALBERT

BERT 的问题

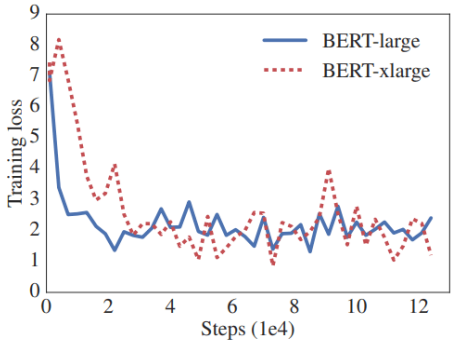
BERT 发布后，在排行榜上产生了许多 NLP 任务的最新成果。但是，模型非常大，导致了一些问题。"ALBERT" 论文将这些问题分为两类：

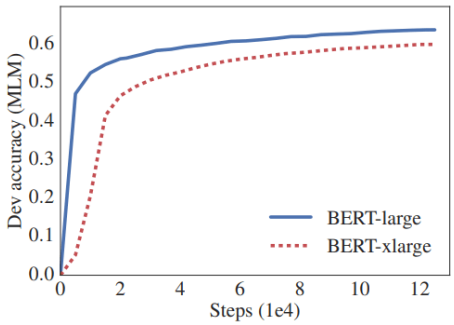
- **内存限制**

> BERT-large 是一个复杂的模型，它有 24 个隐藏层，在前馈网络和多头注意力机制中有很多节点，总共有 3.4 亿个参数，如果想要从零开始训练，需要花费大量的计算资源

- **模型退化**

> 最近在 NLP 领域的研究趋势是使用越来越大的模型，以获得更好的性能。ALBERT 的研究表明，无脑堆叠模型参数可能导致效果降低. "BERT-xlarge"表现都不如 BERT-large





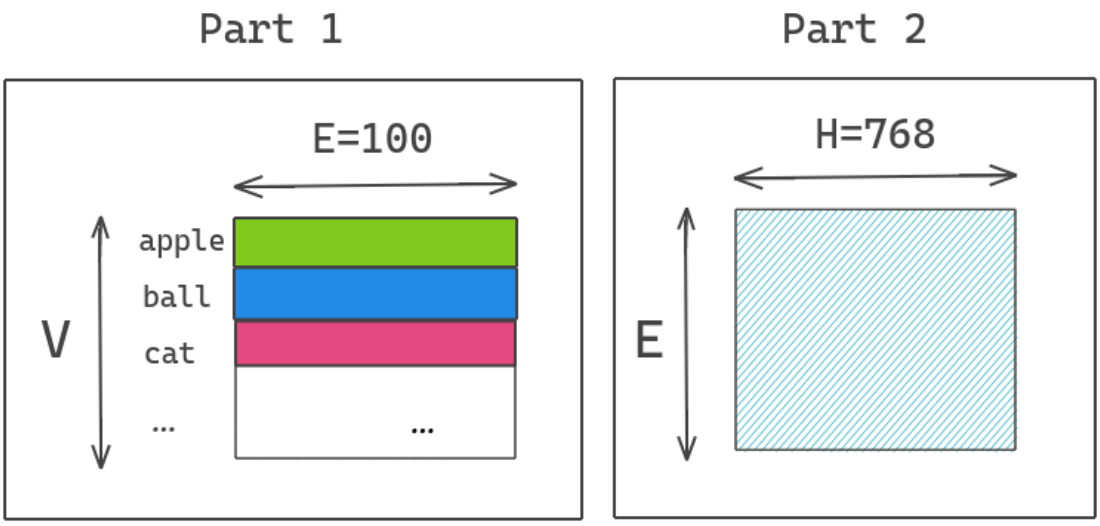
ALBERT

概述

> ALBERT 利用了参数共享、矩阵分解等技术大大减少了模型参数，用 SOP（Sentence Order Prediction） Loss 取代 NSP（Next Sentence Prediction） Loss 提升了下游任务的表现。但是 ALBERT 的层数并未减少，因此**推理时间（Inference Time）还是没有得到改进**。不过参数减少的确使得训练变快，同时 ALBERT 可以扩展到比 BERT 更大的模型（ALBERT-xxlarge），因此能得到更好的表现
>
> ALBERT 的结构和 BERT 基本一样，采用了 Transformer 以及 GELU 激活函数。具体的创新部分有三个：
>
> 1. embedding 层参数因式分解
> 2. 跨层参数共享
> 3. 将 NSP 任务改为 SOP 任务
>
> 前两个改进的主要作用是减少参数。第三个改进其实算不上什么创新了，因为之前已经有很多工作发现 BERT 中 NSP 任务并没有什么积极的影响

Factorized Embedding Parameterization

> 原始的 BERT 模型以及各种依据 Transformer 的预训连语言模型都有一个共同特点，即E = H ，其中E指的是 Embedding Dimension，H指的是 Hidden Dimension。这就会导致一个问题，当提升 Hidden Dimension 时，Embedding Dimension 也需要提升，最终会导致参数量呈平方级的增加。所以 ALBERT 的作者将E和H **进行解绑**，具体的操作就是**在 Embedding 后面加入一个矩阵进行维度变换**。E的维度是不变的，如果H增大了，我们只需要在 E 后面进行一个升维操作即可



所以，ALBERT 不直接将原本的 one-hot 向量映射到 hidden space size of H，而是分解成两个矩阵，原本参数数量为 $V * H$，V 表示的是 Vocab Size。分解成两步则减少为 $V * E + E * H$，当 H 的值很大时，这样的做法能够大幅降低参数数量

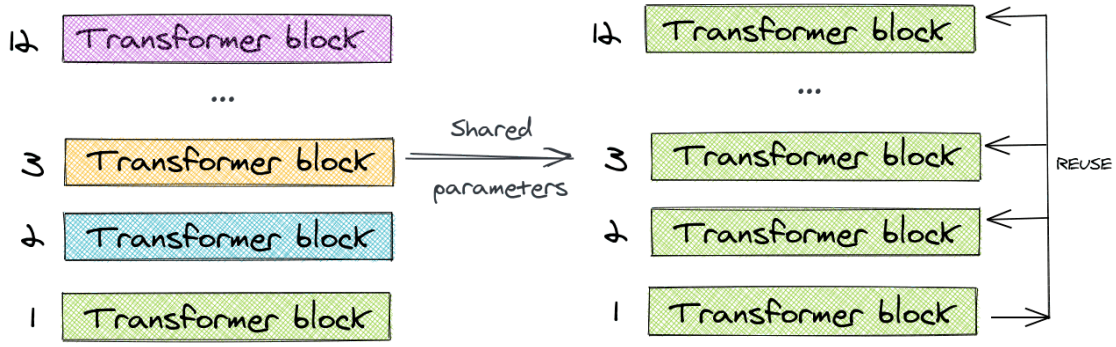
> $V * H = 30000 * 768 = 23,040,000$
>
> $V * E + E * H = 30000 * 256 + 256 * 768 = 7,876,608$
>
> 举个例子，当 V 为 30000，H 为 768，E 为 256 时，参数量从 2300 万降低到 780 万

通过因式分解 Embedding 的实验可以看出，对于参数不共享的版本，随着 E 的增大，效果是不断提升的。但是参数共享的版本似乎不是这样，E 最大并不是效果最好。同时也能发现参数共享对于效果可能带来 1-2 个点的下降

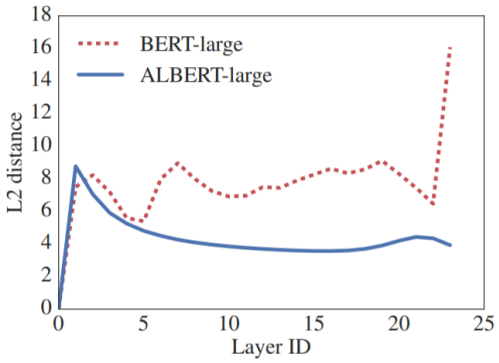
Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	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

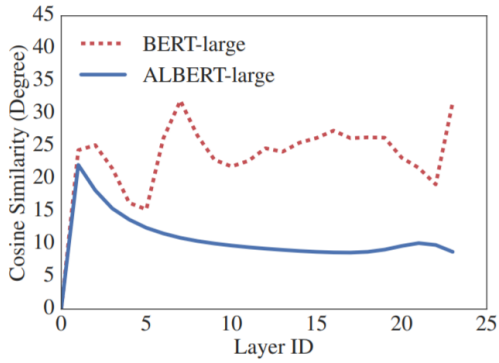
Cross-Layer Parameter Sharing

传统 Transformer 的每一层参数都是独立的，包括各层的 self-attention、全连接。这样就导致层数增加时，参数量也会明显上升。之前有工作试过单独将 self-attention 或者全连接层进行共享，都取得了一些效果。ALBERT 作者尝试将所有层的参数进行共享，相当于只学习第一层的参数，并在剩下的所有层中重用该层的参数，而不是每个层都学习不同的参数



同时作者通过实验还发现了，使用参数共享可以有效的提升模型稳定性，实验结果如下图





BERT-base 和 ALBERT 使用相同的层数以及 768 个隐藏单元，结果 BERT-base 共有 1.1 亿个参数，而 ALBERT 只有 3100 万个参数。通过实验发现，feed-forward 层的参数共享会对精度产生比较大的影响；共享注意力参数的影响是最小的

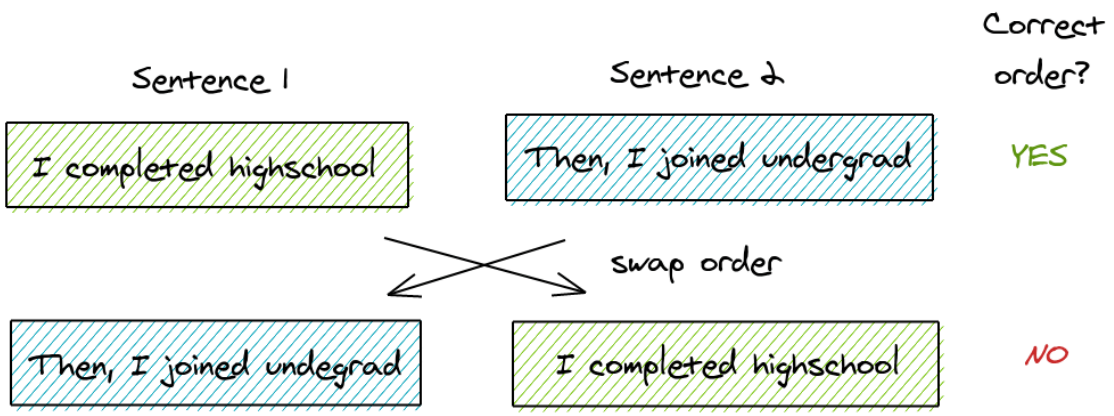
Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
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

Sentence-Order Prediciton (SOP)

BERT 引入了一个叫做**下一个句子预测**的二分类问题。这是专门为提高使用句子对，如 "自然语言推理" 的下游任务的性能而创建的。但是像 RoBERTa 和 XLNet 这样的论文已经阐明了 NSP 的无效性，并且发现它对下游任务的影响是不可靠的

因此，ALBERT 提出了另一个任务 —— **句子顺序预测**。关键思想是：

- 从同一个文档中取两个连续的句子作为一个正样本
- 交换这两个句子的顺序，并使用它作为一个负样本



可以得出的结论是：

1. 在相同的训练时间下，ALBERT 得到的效果确实比 BERT 好
2. 在相同的 Inference 时间下，ALBERT base 和 large 的效果都没有 BERT 好，而且差了 2-3 个点，作者在最后也提到了会继续寻找提高速度的方法（Sparse attention 和 Block attention）