ShareBERT: Embeddings Are Capable of Learning Hidden Layers

Jia Cheng Hu, Roberto Cavicchioli, Giulia Berardinelli, Alessandro Capotondi

University of Modena and Reggio Emilia
via G.Campi 213/b
41125, Modena, Italy
jiachenghu@unimore.it, roberto.cavicchioli@unimore.it,
giulia.berardinelli@unimore.it, alessandro.capotondi@unimore.it

Abstract

The deployment of Pre-trained Language Models in memory-limited devices is hindered by their massive number of parameters, which motivated the interest in developing smaller architectures. Established works in the model compression literature showcased that small models often present a noticeable performance degradation and need to be paired with transfer learning methods, such as Knowledge Distillation. In this work, we propose a parameter-sharing method that consists of sharing parameters between embeddings and the hidden layers, enabling the design of near-zero parameter encoders. To demonstrate its effectiveness, we present an architecture design called ShareBERT, which can preserve up to 95.5% of BERT Base performances, using only 5M parameters (21.9× fewer parameters) without the help of Knowledge Distillation. We demonstrate empirically that our proposal does not negatively affect the model learning capabilities and that it is even beneficial for representation learning. Code will be available at https://github.com/jchenghu/sharebert.

Introduction

Pre-trained language models (PLMs) (Devlin et al. 2018; Brown et al. 2020) have achieved outstanding results on many natural language tasks and established the practice of training a large model on a massive amount of data and fine-tuning on smaller downstream tasks. However, the fact that these models are often made of at least a hundred million parameters poses an obstacle to their deployment in memory-limited devices (Sandler et al. 2018; Lin et al. 2018; Roth et al. 2020; Sarkar et al. 2023) and introduces inference bottlenecks in off-chip operations, mobile-devices (Tabani et al. 2021) and multi-node communications. To combat these problems, many model compression methods were developed and, to the best of our knowledge, the best language model among tiny ones is ALBERT Small (Lan et al. 2019), which reproduces 97.4% of BERT Base (Devlin et al. 2018) performances using only 12M parameters. This is possible thanks to sharing one BERT encoder across all layers. However, even the cost of one single encoding layer can be a limiting factor in some applications. For instance, several edge devices (Ojo et al. 2018; Merenda, Porcaro, and Iero 2020) cannot afford to store more than 10M parameters in mixed precision. One possible solution consists of reducing the hidden size drastically, but established works (Jiao et al. 2019; Sanh et al. 2019) showcased that they do not achieve a satisfying level of performance unless knowledge transfer methods, such as Knowledge Distillation (KD) (Hinton, Vinyals, and Dean 2015), are adopted. Whereas, regarding the first factor, there is a lack of research literature. Input embeddings are often overlooked in model compression techniques even though they can account for a significant portion of the total model size, especially in the lighter variants. For instance, they represent 21% of base BERT (Devlin et al. 2018) and 30% of GPT-3 small (Brown et al. 2020).

In this work, we propose a novel parameter-sharing strategy based on the embeddings, that enables the design of near-zero parameter encoders, pushing the design of tiny pre-trained language models to a new extent. Experimental results show that we achieve 95.5% of BERT Base performances using only 5M parameters (21.9× fewer parameters) and, most importantly, without the help of any transfer learning techniques. The method is orthogonal to quantization and KD, which can be applied to optimize the results further. Additionally, analytic results showcase that our methods do not negatively impact the learning of hidden layers and can pave the way for new developments in transfer learning research. The main contributions of this paper are the following: *(i)* we propose two novel parameter-sharing methods, called *embeddings parameter-sharing* and *virtual embeddings parameter-sharing*; *(ii)* we present the ShareBERT architecture as an instance of the application of the methods on BERT; *(iii)* we discuss the impact of the proposed methods on *representation learning* using Word Analogy and Word Similarity benchmarks, and analyze the learning of hidden layers using the *edge probing* technique introduced in (Tenney, Das, and Pavlick 2019); *(iv)* to showcase the robustness of our approach, we present its impact on two additional applicative domains like Neural Machine Translation and Image Captioning.

Related Works

Model Compression. Pre-trained Language Model compression is a very active research area. One trend consists of applying Knowledge Distillation (Hinton, Vinyals, and Dean 2015), which follows the principle of transferring the knowledge from one model (teacher) into another one (stu-

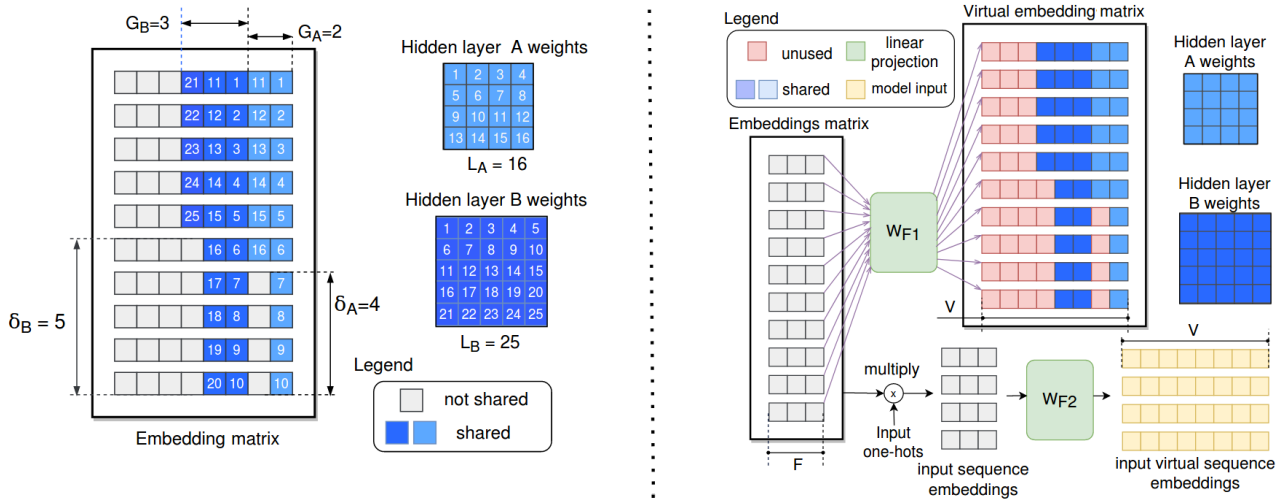

Figure 1: Left) Example of our proposed parameter-sharing given an embedding matrix of shape 10 × eight and two layers A and B, made of L_A=16 and L_B=25 parameters respectively. G_A denotes the number of shared embedding dimensions dedicated for layer A, and δ_A describes the number of parameters not used by the same layer. Right) Virtual embedding matrix mechanism illustrated. F represents the embedding vector size. V is the (virtual) hidden size of the model. W_{F1} and W_{F2} denote linear projections.

dent) of a smaller size. The student model can be characterized by a smaller hidden size, fewer number of layers, and reduced Feed-forward size (Jiao et al. 2019; Sanh et al. 2019; Sun et al. 2019, 2020; Vinoda and Yadav 2022; Xu et al. 2020). Pruning methods (Han, Mao, and Dally 2015; He, Zhang, and Sun 2017) exploit the presence of redundancy in deep neural networks (Denil et al. 2013; Frankle and Carbin 2018) and reduce the number of weights in dense layers (McCarley, Chakravarti, and Sil 2019; Sajjad et al. 2020; Fan, Grave, and Joulin 2019). Quantization tackles the problem of model compression by reducing the number of bits required to store model weights (Zafrir et al. 2019), (Shen et al. 2020), (Bai et al. 2020; Zhang et al. 2020). Other model compression approaches include: replacing the Attention module with lighter formulations (Lee-Thorp et al. 2021), applying NAS (Chen et al. 2020), leveraging low-rank structures on the pre-trained model (Ren et al. 2022) and fine-tuning models (Hu et al. 2021). These methodologies typically focus on the compression of the encoder layers. In contrast, our proposed approach enables the design of near-zero parameter encoders by embeddings specialization. Still, most of these works, like quantization and pruning, are orthogonal to ours and can be combined to further reduce the model size or the inference time.

Parameter Sharing. Our work focuses on the parameter-sharing approach, which refers to using the same weights in multiple layers or instances. Recurrent neural networks (Hochreiter and Schmidhuber 1997; Cho et al. 2014; Luong, Socher, and Manning 2013) early proved the effectiveness of the method and shared the parameters along the sequence length, whereas, more recently, supported by the results of (Bai, Kolter, and Koltun 2019), the works of (Lan et al. 2019; Reid, Marrese-Taylor, and Matsuo 2021) extended the concept over the neural network's depth. (Dehghani et al. 2018; Hao et al. 2019) integrated the inductive bias of recurrent networks in self-attentive networks (Vaswani et al. 2017). (Suzuki and Nagata 2016) performed parameter-sharing among token representations to reduce the embedding matrix size. (Sachan and Neubig 2018) proposed partially shared decoders for multi-lingual translation. To the best of our knowledge, our work is the only one that involves embedding matrix parameters in both word representations and hidden layers. In this way, we push the parameter-sharing approach to an unprecedented extent by encompassing approximately the entire network in the sharing.

Method

Embeddings Parameter-Sharing (EPS)

Embeddings Parameter-Sharing (EPS) mainly consists of leveraging a portion of the embedding matrix columns for the learning of hidden layers. We break down the method into multiple steps and apply it on BERT (Devlin et al. 2018) to provide both an application example and the base concept of ShareBERT.

(i) Target Selection. We first identify the components to be constructed from the embedding matrix. In the case of BERT, the most memory-expensive components are the Feed-Forward (FF) and Self-Attention (SA) layers:

$$\begin{aligned} FF(X) &= \sigma(XW_{HI} + b_{HI})W_{IH} + b_{IH} \\ SA(X) &= Softmax(QK^{\intercal})VW_O + b_O \\ Q &= XW_Q + b_Q \\ K &= XW_K + b_K \\ V &= XW_V + b_V \end{aligned} \tag{1}$$

where σ refers to the activation function, W_Q, W_K, W_V,

$W_O \in \mathbb{R}^{H\times H}$, $b_Q, b_K, b_V, b_O, b_{IH} \in \mathbb{R}^H$, $W_{HI}, W_{IH}^\intercal \in \mathbb{R}^{H\times I}$ and $b_{HI} \in \mathbb{R}^I$ are learned parameters. H and I represent the hidden and intermediate sizes, respectively. Because of their negligible cost, biases are excluded from the discussion and are implemented in the standard way. Similarly to (Lan et al. 2019), we plan to use one single encoder layer shared with all N layers. Thus, we need to construct only one instance of SA and FF.

(ii) Shared Columns Partitioning. The cost of FF and SA, denoted as L_{FF} and L_{SA} respectively, can be broke down as follows:

$$\begin{aligned} L_{FF} &= L_{HI} + L_{IH} \\ L_{SA} &= L_Q + L_K + L_V + L_O \\ L_{HI} &= L_{IH} = HI \\ L_Q &= L_K = L_V = L_O = H^2 \end{aligned} \tag{2}$$

where L_{HI} and L_{IH} represent the number of parameters required by W_{HI} and W_{IH} in Equation 1. Whereas L_Q, L_K, L_V and L_O refers to the number of parameters required by W_Q, W_K, W_V and W_O. For each element to be constructed, denoted as $*$, we select G_* columns of the embedding matrix according to its cost L_*. Assuming that the embedding matrix is of size $E \times H$, where E is the number of tokens, we create six separate groups of columns of size G_{IH}, G_{HI}, G_Q, G_K, G_V and G_O:

$$\begin{aligned} G_{IH} &= G_{HI} = \left\lceil \frac{L_{HI}}{E} \right\rceil \\ G_Q &= G_K = G_V = G_O = \left\lceil \frac{L_Q}{E} \right\rceil \end{aligned} \tag{3}$$

Each separate group of columns will create a separate linear transformation out of embedding parameters and they will be trained for both word representation and hidden layer learning. Since different columns are reserved for each group, the following condition must be satisfied:

$$G_{IH} + G_{HI} + G_Q + G_K + G_V + G_O \leq H \tag{4}$$

(iii) Hidden Layers Construction. Once the embedding matrix columns to be shared are defined, we concretely build each projection layer by selecting each parameter in a top-to-bottom order and populating the transformation weights in a right-to-left fashion, as illustrated in Figure 1-Left. The ceiling operator in Equation 3 ensures enough columns are selected for each layer. As a result, however, it takes an upper-bound to the actual number of required parameters:

$$\begin{aligned} &(G_H + G_{HI} + G_Q + G_K \\ &+ G_V + G_O)E \geq L_{FF} + L_{SA} \end{aligned} \tag{5}$$

The Inequality 5 can be reformulated as:

$$\begin{aligned} &(L_E + \delta_{HI}) + (L_{IH} + \delta_{HI}) + (L_Q + \delta_Q) + (L_K \\ &+ \delta_K) + (L_V + \delta_V) + (L_O + \delta_O) \geq L_{FF} + L_{SA} \end{aligned} \tag{6}$$

Where $0 \leq \delta_{IH}, \delta_{IH}, \delta_Q, \delta_K, \delta_V, \delta_O < E$ represent the number of unused parameters in a shared column and are simply discarded.

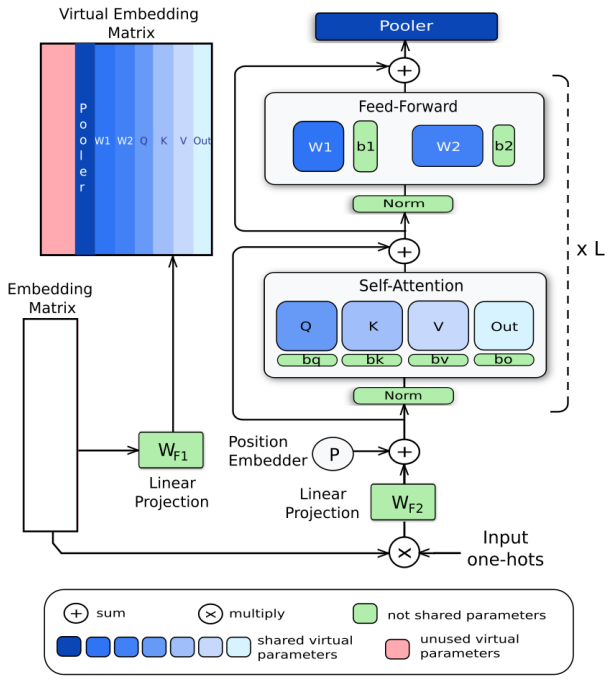

Figure 2: ShareBERT parameters allocation. The Pooler is involved only in the evaluation phase. L refers to the number of layers.

Virtual Embeddings Parameter-Sharing (VEPS)

A practical obstacle of the EPS method is described in Equation 4. The approach cannot be adopted if the required shared dimension exceeds the word embedding size H. To overcome this limitation, we propose the *virtual* embeddings parameter-sharing (VEPS), which consists of applying a linear projection over the entire embedding matrix to increase its dimensionality. We call the result *the virtual embedding matrix* and we construct the dense layers from it according to EPS. VEPS not only mitigates the problem of an insufficient number of dimensions but also enables the adoption of embedding factorization as described in (Lan et al. 2019). Therefore, we deploy one projection to the embeddings to create the virtual embedding matrix and another one to conform the dimensionality of the input vectors to the encoders. In this regard, although input word embeddings and hidden layers parameters are not properly the same parameters in contrast to the EPS, we refer to them as shared regardless, because the two sets of parameters share the representation space from which they originate. The mechanism is illustrated in Figure 1-Right.

ShareBERT

ShareBERT is an architecture design that is similar to BERT but differs in the fact that it leverages VEPS to construct one single Self-Attention and Feed-Forward layer, to be shared across all layers. Since encoders are generated from the embeddings, they require near-zero additional trainable parameters, and embeddings parameters closely account for the total number of learned parameters. For simplicity, biases and

Model	# params	Compression rate	Benchmark	Relative perf.
BERT Base	109.5M	1.0×	-	100.0%
TinyBERT$_4$ w/o KD (Jiao et al. 2019)	14.5M	7.5×	GLUE test	88.8%
TinyBERT$_4$ (Jiao et al. 2019)	14.5M	7.5×	GLUE test	96.8%
DistilBERT (Sanh et al. 2019)	66M	1.6×	♣	96.8%
BERT$_3$-PKD (Sun et al. 2019)	45M	2.4×	GLUE test	91.6%
AlBERT Base (Lan et al. 2019)	12M	9.0×	†	97.3%
MobileBERT$_{\text{TINY}}$ (Sun et al. 2020)	15M	7.2×	GLUE test	96.8%
Bert-of-Theseus (Xu et al. 2020)	66M	1.6×	♢	98.4%
Drop-6 (Sajjad et al. 2020)	66M	1.6×	GLUE dev	97.0%
FNet (Lee-Thorp et al. 2021)	83M	1.3×	GLUE dev	92.0%
BERT-IV-36-128 (Ren et al. 2022)	25M	4.3×	GLUE test	97.7%
oBERT$_{3,90}$ (Kurtic et al. 2022)	27M	4.0×	♡	93.1%
HomoBERT-tiny (Liang et al. 2023)	14.1M	7.7×	GLUE dev	93.3%
LAD (Lin, Chen, and Kao 2023)	66.5M	1.6×	♢	99.0%
ShareBERT Small	5.0M	21.9×	GLUE dev	95.5%
ShareBERT Base	13.8M	7.9×	GLUE dev	97.4%
ShareBERT Large	27.0M	4.0×	GLUE dev	100.0%

GLUE test/dev:mnli+qqp+qnli+mrpc+sst2+cola+stsb+rte. †: SQUAD+RACE+mnli+sst2.
♣:mnli+qqp+qnli+mrpc+sst2+cola+stsb+rte+wnli. ♡: mnli+qqp+qnli+mrpc+sst2+cola+stsb.
♢: sst2+mrpc+qqp+mnli+qnli+rte.

Table 1: Compression rate and relative performance comparison of State-of-the-Art model compression methods against the BERT Base. "# params" reports the number of learned parameters of the entire model.

normalization layers are implemented in the standard way. We propose three variants of ShareBERT named "Small", "Base", and "Large". The "Small" variant is made of N=12 encoders, virtual hidden size V=2048, Feed-Forward size I=4096, an embeddings matrix of E=30528 token vectors of size F=128 and a virtual embedding matrix of size $E \times V$. "Base" is made of the following configurations $\{N$=12, V=2048, I=4096, F=384$\}$, whereas "Large" is characterized by $\{N$=6, V=2048, I=4096, F=768$\}$. During inference time, a linear projection, called Pooler, is allocated on top of the output corresponding to the "[CLS]" token. Figure 2 illustrates the complete parameter allocation.

Experimental Results

In this Section, we present the performances of ShareBERT models compared to BERT and other model compression methods in the literature. Then, offer an ablation study and explain the reason behind the choice of hyperparameters. Moreover, we show the effectiveness of our approach in other applications. Since our method consists of leveraging embeddings in the learning of hidden layers, we also investigate its impact on both representation and hidden layers learning.

For ShareBERT variants and BERT baselines, we perform MLM training on the 2022 English Wikipedia and BookCorpus (Zhu et al. 2015) of date 2020, we use the same sub-word tokenization of BERT (Devlin et al. 2018) in the uncased instance. All models are trained for 23000 steps, batch size of 4000 and fine-tuned on GLUE tasks. FP16 mixed precision is adopted. During the analysis of representation learning, no additional training is required. The training details of probing models follow the configuration of (Tenney, Das, and Pavlick 2019). When EPS is applied in other applicative fields, models are trained with the same setup of (Vaswani et al. 2017) and (Hu, Cavicchioli, and Capotondi 2022).

Comparison With Other Model Compression Methods

Since established works adopt different training strategies and perform the evaluation on different benchmarks or subsets of GLUE (Wang et al. 2018) tasks, in Table 1, we note the relative accuracy of each method compared to the BERT Base as reported in their respective work and experiment configuration. ShareBERT models achieve State-of-the-Art compression and competitive performances compared to existing methods. Compared to TinyBERT$_4$ and AlBERT, which appear to be the best among small PLMs, ShareBert Base achieves comparable results with AlBERT Base. However, our architecture can be designed to compress BERT even further thanks to near-zero parameter encoders. In particular, ShareBERT Small achieves 95.5% of the BERT Base performances using 1/22 of BERT total parameter cost. When compared to TinyBERT$_4$, it performs 1.3% worse but outperforms the latter architecture by a margin of 6.7% in the absence of KD, using only 1/3 of the parameters. Finally, we observe that ShareBERT Large achieves the same performance as BERT using 1/4 of its parameters without the help of KD, suggesting better scalability compared to other approaches.

ShareBERT Variants Study

In this Section, we analyze the impact of EPS and VEPS on the performances, in terms of compression and accuracy on the GLUE (Wang et al. 2018) development set, and motivate the hyper-parameter choices of ShareBERT.

Table 2 shows that applying EPS to BERT Base (BERT + EPS) leads to a performance drop of 4.4% accuracy. This is expected since sharing a portion of parameters for two different tasks introduces noise in both representation and hidden layer learning. This drawback, however, can be com-

Model	# params (γ)	w/o embeds (γ)	mnli	qqp	qnli	sst2	cola	stsb	mrpc	rte	avg
BERT											
$\{H$=768,L=12,I=3072$\}$ (Base)	109.5M (1.0×)	85.6M (1.0×)	81.1	88.7	89.1	90.4	80.2	83.6	81.9	56.6	81.4
$\{H$=128,L=12,I=512$\}$	6.3M (17.3×)	2.4M (35.6×)	71.8	80.5	83.2	83.8	69.9	65.8	81.5	54.5	73.8
$\{H$=256,L=12,I=1024$\}$	17.4 (6.3×)	9.5M (9.0×)	77.1	84.1	84.2	87.2	74.4	78.6	81.5	53.4	77.5
BERT + EPS											
$\{H$=768,L=12,I=1024$\}$	24.5M (4.4×)	1.0M (85.6×)	71.5	80.2	77.1	84.8	75.7	36.5	76.0	54.5	69.5
$\{H$=768,L=12,I=3072$\}$	24.5M (4.4×)	1.0M (85.6×)	74.5	83.3	83.7	85.5	70.5	82.5	77.0	59.2	77.0
$\{H$=768,L=12,I=6144$\}$	24.5M (4.4×)	1.0M (85.6×)	75.9	84.0	85.0	84.9	73.1	74.5	81.2	53.7	76.5
BERT + EPS + EF											
$\{F$=1152,H=2048,L=12,I=4096$\}$	43.0M (2.5×)	7.8M (10.9×)	79.0	86.0	87.2	88.7	79.8	83.4	87.0	56.3	80.9
BERT + VPES + EF (ShareBERT)											
$\{F$=24,V=512,L=10,I=512$\}$	0.9M (121.6×)	0.2M(428.0×)	61.3	72.6	79.0	78.5	69.1	40.3	80.7	44.6	65.7
$\{F$=64,V=768,L=12,I=768$\}$	2.2M (49.7×)	0.3M (285.3×)	71.4	81.5	81.5	82.5	69.4	68.5	81.8	53.0	73.7
$\{F$=128,V=2048,L=12,I=4096$\}$ (Small)	5.0M (21.9×)	1.0M (85.6×)	75.4	83.8	84.4	86.2	70.4	81.0	83.5	57.7	77.8
$\{F$=384,V=2048,L=12,I=4096$\}$ (Base)	13.8M (7.9×)	2.1M (40.7×)	77.5	85.1	85.0	87.9	75.6	82.3	84.7	56.6	79.3
$\{F$=768,V=2048,L=6,I=4096$\}$ (Large)	27.0M (4.0×)	3.5M (24.4×)	79.3	86.0	86.7	89.6	77.4	83.9	86.7	61.7	81.4

H=Hidden size. L=Number of layers. I=Feed forward size. EPS=Embeddings parameter-sharing. VEPS=Virtual EPS.
V=Model hidden size in case of VEPS. EF=Embedding factorization. F=Embeddings size in case of EF.

Table 2: Performance and the number of parameters comparison between ShareBERT variants and BERT evaluated on the GLUE dev set. Accuracy is reported for *wnli*, *qnli*, *sst2*, *cola* and *rte*. F1 is used for *qqp* and *mrpc*. Spearman's Rank Correlation is reported for *stsb*. "Avg"= average score. "# params" = the number of learned parameters of the entire model. "w/o embeds" = the parameter cost without considering embeddings. "γ" = compression rate compared to BERT Base.

pensated by increasing the model size, which, to some extent, can be done without the expense of additional parameters. For instance, increasing the Feed-Forward size from I=1024 to I=6144 increases the number of shared columns in the embedding matrix but does not change the memory footprint. Unfortunately, increasing the Feed-Forward size over the popular configuration 3072 appears to have a limited impact. In contrast, a more effective strategy involves increasing the network hidden size (Devlin et al. 2018). For instance, in "BERT + EPS + EF" we take about the right amount of required columns (embedding size F=1152) to construct a network of hidden size H=2048 and observe comparable results to BERT Base using half the number of parameters. To push the compression reduction even further, in ShareBERT Small, we reduce the factorization dimension to 128 but preserve the network hidden size using the VEPS method (BERT + VPES + EF). This results in a drop of 3.1% average GLEU score, but it achieves 95.5% of BERT performance using 21.9× fewer learned parameters. Increasing the parameter F increases scores, in this way ShareBERT Base and Large achieve 97.4% and 100% of BERT Base accuracy respectively. For BERT architecture to produce similar performances to ShareBERT, it requires up to 4× more parameters, outlining better scalability of our approach.

EPS in Other Applicative Domains

To investigate the robustness of our method across multiple applications we apply EPS in two additional tasks: English-Vietnamese Neural Machine Translation (En-Vi NMT) and Image Captioning (IC)[1]. In NMT, the first baseline is the base Transformer (Vaswani et al. 2017), whereas for IC we implement the same architecture but halve the number of layers for IC. We implement another baseline called "Simple-Share" in which a single Self-Attention block is shared throughout the encoder's depth, two are shared across all decoder's layers, and a single Feed-Forward is shared across all layers. Our proposal, denoted as "Ours", replicates "Simple-Share" but adopts the EPS method for all linear transformations. From Table 3, we observe that, in the case of NMT, our proposal achieves 95.0% of the Transformer's performance using less than 1/3 of the number of parameters. On the IC task, our model achieves roughly the same results while being 3.48× smaller. Additionally, our

Machine Translation						
Model (#params.)	**BLEU**					
$\text{Baseline}_{\text{MT}}$ (63M)	31.13					
$\text{Simple-Share}_{\text{MT}}$ (26M)	29.85					
Ours_{MT} (18M)	29.58					
Image Captioning						
Model (#params.†)	**B1**	**B4**	**R**	**M**	**S**	**C**
$\text{Baseline}_{\text{IC}}$ (38M)	81.8	39.6	59.2	29.2	23.0	132.8
$\text{Simple-Share}_{\text{IC}}$ (15M)	80.9	38.6	58.6	28.8	22.7	130.8
Ours_{IC} (11M)	81.6	39.2	59.0	29.0	22.9	132.6

†: Backbone is excluded from the calculation.

Table 3: EPS in Neural Machine Translation and Image Captioning. B1=BLEU1, B4=BLEU4 (Post 2018), R=Rouge (Lin 2004), M=Meteor (Banerjee and Lavie 2005), S=Spice (Anderson et al. 2016), C=CIDEr-D (Vedantam, Lawrence Zitnick, and Parikh 2015)

[1]IC models are trained using the standard SCST (Rennie et al. 2017) with SacreEOS (Hu, Cavicchioli, and Capotondi 2023) signature: `STANDARD_wInit` + `Cider-D[n4,s6.0]` + `average[nspi5]` + `1.0.2`. NMT models are trained according to (Vaswani et al. 2017) and evaluated using SacreBLEU (Post 2018) with signature: `BLEU` + `case.mixed` + `lang.en-vi` + `numrefs.1` + `smooth.exp` + `tok.13a` + `version.2.0`

Model	Shared dims (%)	RW	WS353	SCWS	WS Avg	WA	Perf.
BERT (Baseline)							
{H=768, L=12, I=512}	0 (0%)	0.365	0.519	0.537	0.473	9.7%	80.5
{H=768, L=12, I=1024}	0 (0%)	0.365	0.519	0.551	0.478	9.9%	80.9
{H=768, L=12, I=3072} (Base)	0 (0%)	0.367	0.529	0.559	0.485	9.9%	81.4
BERT + EPS							
{H=768, L=12, I=1024}	130 (16.9%)	0.404	0.534	0.646	0.528	3.9%	69.5
{H=768, L=12, I=3072}	233 (30.3%)	0.127	0.304	0.362	0.264	3.3%	77.0
{H=768, L=12, I=6144}	388 (50.5%)	0.186	0.461	0.414	0.353	4.2%	76.5
{BERT + EPS + EF}							
F=1152, H=2048, L=12, I=4096	1100 (95.4%)	0.385	0.585	0.550	0.506	1.9%	80.9
BERT + VPES + EF (ShareBERT)							
{F=128, V=2048, L=12, I=4096} (Small)	128 (100%)	0.491	0.679	0.614	0.594	10.7%	77.8
{F=384, V=2048, L=12, I=4096} (Base)	384 (100%)	0.525	0.653	0.629	0.602	12.6%	79.3
{F=768, V=2048, L=6, I=4096} (Large)	768 (100%)	0.565	0.698	0.652	0.638	15.2%	81.4

H=Hidden size. L=Number of layers. I=Feed forward size. EPS=Embeddings parameter-sharing. VEPS=Virtual EPS.
V=Model hidden size in case of VEPS. EF=Embedding factorization. F=Embeddings size in case of EF.

Table 4: Impact of embeddings parameter sharing on embeddings learning. Scores are measured in Spearman's Rank Correlation coefficient for Word Similarity tasks (RW, WS353, SCWS) and the percentage accuracy for the Word Analogy (denoted as WA) task. "WS Avg" reports the average of Word Similarity scores. "Perf." = average score on GLUE dev set. "Shared dims" = the number of embedding matrix columns involved in the generation of hidden layer weights. In the case of "BERT + VPES + EF", we evaluate the virtual sequence embeddings of size V, and "Shared dims" refers to the embeddings of size F.

approach outperforms the "Simple-Share" variant on the IC task by a margin of 1.8 CIDEr (Vedantam, Lawrence Zitnick, and Parikh 2015) but performs slightly worse by a margin of 0.27 BLEU (Post 2018) in the case of NMT. These results suggest that embeddings can be leveraged to compress models across multiple applications while preserving a good portion of performances and outlining the generalizability of our method.

Embedding Learning Analysis

In this Section, we study whether our proposed parameter-sharing method poses an obstacle to word representation learning. Following the work of (Pennington, Socher, and Manning 2014), embeddings are evaluated on the Word Analogy (WA) task (Mikolov et al. 2013) and Word Similarity (WS) benchmarks: WordSim-353 (Finkelstein et al. 2001), RW (Luong, Socher, and Manning 2013) and SCWS (Huang et al. 2012) against BERT Base. In Table 4, it can be observed that vanilla embeddings parameter-sharing methods (BERT + EPS and BERT + EPS + EF) lead to a clear drop in WA accuracy and mixed results in WS tasks, which can not be attributed to the increase of Feed-Forward size. We believe the occasional superior Spearman's Rank Correlation coefficient (in brief *correlation*) observed in the WS tasks is accidental since deploying the same parameters for two semantically different goals translates into introducing noise in both tasks. Surprisingly, in the case of virtual EPS (BERT + VEPS + EF), regardless of the GLUE score, we observe a positive trend with a maximum improvement of 0.153 WS average correlation and 5.3% accuracy in WA. A possible explanation for this phenomenon is that involving embeddings in the learning of hidden layers gives them access to higher-level information about how tokens interact with each other, improving the semantics captured in the representations. This also suggests that decoupling the virtual embedding matrix from the input sequence embeddings using two linear projections is enough to overcome the vanilla EPS's downside mentioned above. For these reasons, while EPS showcased mixed results we believe VEPS to be beneficial for representation learning.

Edge Probing Analysis

In this Section, we study the impact of our methods on the quality of hidden layers learning. To this end, we adopt the *Edge Probing* method described in (Tenney, Das, and Pavlick 2019) and compare the results of ShareBERT against our BERT implementation. We assume a series of small probing models on top of our models and train them to perform different linguistic tasks. We prepare two setups, denoted as l=0 and l=12. In the first, probing models have access only to the embeddings, whereas, in the second, they also have access to all layers' output. We report the F1 score in each task, and in the case of l=12, we note the "centre of gravity", which estimates how deep the probing model needs to "look" to find the required information to solve the task.

On the left of Figure 3, we observe that our baseline BERT Base confirms the pattern observed in (Tenney, Das, and Pavlick 2019): syntactic information such as POS, SRL, and Consts are found earlier in the network, while high-level semantic information required by Ent, Coref and Rel tasks can be found at deeper layers. By computing the Spearman's Rank Correlation between ShareBERT variants and BERT centers of gravity, we get 0.77, 0.77, 0.94, 0.88, and 0.88 for architectures (b), (c), (d), (e), and (f) respectively (following the naming in Figure 3). Such values suggest a reasonably strong agreement on how the models operate internally. F1 scores reported on the right of Figure 3 support the hypothesis formulated in the previous Section: involving embeddings in higher-order interactions leads to semantically richer representations. This can be observed because when

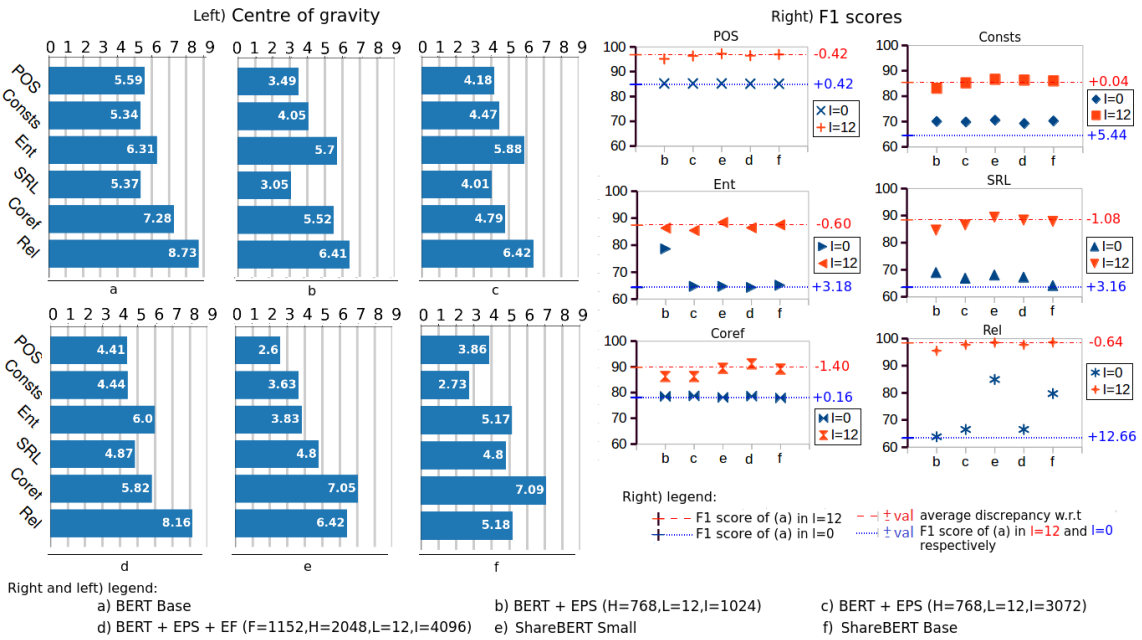

Figure 3: On the right) F1 scores of probing models in various linguistic tasks (Tenney, Das, and Pavlick 2019). l=0 and l=12 refer to the case in which models have access only to the embeddings and all 12 layers' output respectively. On the left) Centre-of-Gravity of probing models in the l=12 case. Linguistic tasks (Tenney, Das, and Pavlick 2019): POS=Part-of-Speech. Consts=Constituents. Ent=Entities. SRL=Semantic Role Labeling. Coref=Coreferences. Rel=Relationships.

EPS and VEPS are applied, probing models with access only to the embeddings (l=0) generally achieve higher or comparable F1 scores against BERT across all tasks. Only a slight reduction is observed in the case of l=12. From these results, we conclude that our proposed parameter-sharing methods have a limited impact on the learning of hidden layers and preserve the overall quality of BERT's inner workings.

Limitations and Discussion

In contrast to other model compression methods, EPS and VEPS reduce the number of parameters but are not designed to decrease the computational cost. On the contrary, ShareBERT increases the hidden size of the network to compensate for the inherent performance degradation caused by a significant reduction of parameters. Assuming batch size one and sequence length 128, ShareBERT Base and Small are 4.5× slower than BERT (126 GFLOPS against 28 GFLOPS), whereas ShareBERT Large is only 2.6× slower (74 GFLOPS) because the number of layers is halved.

In this work, we focused exclusively on achieving competitive accuracies with a limited number of parameters. For instance, we adopted 12 encoder layers for ShareBERT Small and Base in conformity with the choice of BERT Base (Devlin et al. 2018) for analytical purposes (edge probing comparison). This may not be the optimal choice (Ba and Caruana 2014) and given that all ShareBERT encoders are identical except for biases and normalization layers, the network can be interpreted as applying roughly the same function f to the input over and over for each layer $y_i = f(x_{i-1}), i \in \{1, \ldots, 12\}$. Instead, one can learn directly an approximating function $\hat{f}$ such that $y_{12} = \hat{f}(x_0)$ using the method of KD (Hinton, Vinyals, and Dean 2015).

Conclusions

The number of learned parameters in large pre-trained language models is prohibitively expensive for memory-limited devices. We propose a parameter-sharing approach that leverages the embeddings for the learning of hidden layers. We present the ShareBERT architecture, which achieves competitive results using only 5M parameters, 21.9× fewer w.r.t BERT Base. Comprehensive experiments showcased that our proposed parameter-sharing methods do not compromise the quality of hidden layers. Moreover, their usage can be beneficial also for representation learning and they can be applied across multiple applicative domains.

Future research will explore additional compression techniques, incorporating well-established methods such as pruning (Kurtic et al. 2022), quantization (Shen et al. 2020), and knowledge distillation (Hinton, Vinyals, and Dean 2015). These methods are independent of our current proposal and can be synergistically integrated to substantially enhance both inference speed and performance (Jiao et al. 2019), particularly in the context of knowledge distillation.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 programme dAIEDGE (G.A. No 101120726).

References

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2016. Spice: Semantic propositional image caption evaluation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, 382–398. Springer.

Ba, J.; and Caruana, R. 2014. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27.

Bai, H.; Zhang, W.; Hou, L.; Shang, L.; Jin, J.; Jiang, X.; Liu, Q.; Lyu, M.; and King, I. 2020. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2019. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32.

Banerjee, S.; and Lavie, A. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 65–72.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, D.; Li, Y.; Qiu, M.; Wang, Z.; Li, B.; Ding, B.; Deng, H.; Huang, J.; Lin, W.; and Zhou, J. 2020. Adabert: Task-adaptive bert compression with differentiable neural architecture search. *arXiv preprint arXiv:2001.04246*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, Ł. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Denil, M.; Shakibi, B.; Dinh, L.; Ranzato, M.; and De Freitas, N. 2013. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fan, A.; Grave, E.; and Joulin, A. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Finkelstein, L.; Gabrilovich, E.; Matias, Y.; Rivlin, E.; Solan, Z.; Wolfman, G.; and Ruppin, E. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, 406–414.

Frankle, J.; and Carbin, M. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Hao, J.; Wang, X.; Yang, B.; Wang, L.; Zhang, J.; and Tu, Z. 2019. Modeling recurrence for transformer. *arXiv preprint arXiv:1904.03092*.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, 1389–1397.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hu, J. C.; Cavicchioli, R.; and Capotondi, A. 2022. ExpansionNet v2: Block Static Expansion in fast end to end training for Image Captioning. *arXiv preprint arXiv:2208.06551*.

Hu, J. C.; Cavicchioli, R.; and Capotondi, A. 2023. A Request for Clarity over the End of Sequence Token in the Self-Critical Sequence Training. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 14233 LNCS: 39 – 50.

Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th annual meeting of the association for computational linguistics (Volume 1: Long papers)*, 873–882.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Kurtic, E.; Campos, D.; Nguyen, T.; Frantar, E.; Kurtz, M.; Fineran, B.; Goin, M.; and Alistarh, D. 2022. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259*.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Lee-Thorp, J.; Ainslie, J.; Eckstein, I.; and Ontanon, S. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.

Liang, C.; Jiang, H.; Li, Z.; Tang, X.; Yin, B.; and Zhao, T. 2023. Homodistil: Homotopic task-agnostic distillation of pre-trained transformers. *arXiv preprint arXiv:2302.09632*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Lin, S.; Liu, N.; Nazemi, M.; Li, H.; Ding, C.; Wang, Y.; and Pedram, M. 2018. FFT-based deep learning deployment in embedded systems. In *2018 Design, Automation*

& Test in Europe Conference & Exhibition (DATE), 1045–1050. IEEE.

Lin, Y.-J.; Chen, K.-Y.; and Kao, H.-Y. 2023. LAD: Layer-Wise Adaptive Distillation for BERT Model Compression. *Sensors*, 23(3).

Luong, T.; Socher, R.; and Manning, C. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 104–113. Sofia, Bulgaria: Association for Computational Linguistics.

McCarley, J.; Chakravarti, R.; and Sil, A. 2019. Structured pruning of a BERT-based question answering model. *arXiv preprint arXiv:1910.06360*.

Merenda, M.; Porcaro, C.; and Iero, D. 2020. Edge machine learning for ai-enabled iot devices: A review. *Sensors*, 20(9): 2533.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ojo, M. O.; Giordano, S.; Procissi, G.; and Seitanidis, I. N. 2018. A review of low-end, middle-end, and high-end IoT devices. *IEEE Access*, 6: 70528–70554.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Post, M. 2018. A call for clarity in reporting BLEU scores. *arXiv preprint arXiv:1804.08771*.

Reid, M.; Marrese-Taylor, E.; and Matsuo, Y. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. *arXiv preprint arXiv:2101.00234*.

Ren, Y.; Wang, B.; Shang, L.; Jiang, X.; and Liu, Q. 2022. Exploring extreme parameter compression for pre-trained language models. *arXiv preprint arXiv:2205.10036*.

Rennie, S. J.; Marcheret, E.; Mroueh, Y.; Ross, J.; and Goel, V. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7008–7024.

Roth, W.; Schindler, G.; Zöhrer, M.; Pfeifenberger, L.; Peharz, R.; Tschiatschek, S.; Fröning, H.; Pernkopf, F.; and Ghahramani, Z. 2020. Resource-efficient neural networks for embedded systems. *arXiv preprint arXiv:2001.03048*.

Sachan, D. S.; and Neubig, G. 2018. Parameter sharing methods for multilingual self-attentional translation models. *arXiv preprint arXiv:1809.00252*.

Sajjad, H.; Dalvi, F.; Durrani, N.; and Nakov, P. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sarkar, S.; Babar, M. F.; Hassan, M. M.; Hasan, M.; and Santu, S. K. K. 2023. Exploring Challenges of Deploying BERT-based NLP Models in Resource-Constrained Embedded Devices. *arXiv preprint arXiv:2304.11520*.

Shen, S.; Dong, Z.; Ye, J.; Ma, L.; Yao, Z.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 8815–8821.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; and Zhou, D. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Suzuki, J.; and Nagata, M. 2016. Learning Compact Neural Word Embeddings by Parameter Space Sharing. In *IJCAI*, 2046–2052.

Tabani, H.; Balasubramaniam, A.; Marzban, S.; Arani, E.; and Zonooz, B. 2021. Improving the efficiency of transformers for resource-constrained devices. In *2021 24th Euromicro Conference on Digital System Design (DSD)*, 449–456. IEEE.

Tenney, I.; Das, D.; and Pavlick, E. 2019. BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vedantam, R.; Lawrence Zitnick, C.; and Parikh, D. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4566–4575.

Vinoda, D.; and Yadav, P. K. 2022. SDBERT: SparseDistilBERT, a faster and smaller BERT model. *arXiv preprint arXiv:2208.10246*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Xu, C.; Zhou, W.; Ge, T.; Wei, F.; and Zhou, M. 2020. Bert-of-theseus: Compressing bert by progressive module replacing. *arXiv preprint arXiv:2002.02925*.

Zafrir, O.; Boudoukh, G.; Izsak, P.; and Wasserblat, M. 2019. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, 36–39. IEEE.

Zhang, W.; Hou, L.; Yin, Y.; Shang, L.; Chen, X.; Jiang, X.; and Liu, Q. 2020. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*.

Zhu, Y.; Kiros, R.; Zemel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, 19–27.