Weight Squeezing: Reparameterization for Knowledge Transfer and Model Compression

Anonymous ACL submission

Abstract

In this work, we present a novel approach to simultaneous knowledge transfer and model compression called **Weight Squeezing**. With this method, we perform knowledge transfer from a teacher model **by learning the mapping from its weights to smaller student model weights**.

We applied Weight Squeezing to a pre-trained text classification model based on a BERT-Medium model. We compared our method to various other knowledge transfer and model compression methods using the GLUE multitask benchmark. We observed that our approach produces better results while being significantly faster than other methods for training student models.

We also proposed a variant of Weight Squeezing called Gated Weight Squeezing, in which we combined fine-tuning a small BERT model and learning mapping from larger BERT weights. We showed that, in most cases, fine-tuning a BERT model with Gated Weight Squeezing can outperform plain fine-tuning.

1 Introduction

Fine-tuning pre-trained models became a de-facto standard technique in natural language processing (NLP). Devlin et al. (2019) introduced BERT (Bidirectional Encoder Representations from Transformers), a language representation Transformer model (Vaswani et al., 2017) trained to predict masked tokens in texts from unlabeled data.

While BERT is capable of learning rich representations of text, using it for solving simple downstream tasks can be excessive. This is especially important when running downstream models on edge devices, such as mobile phones. A common approach in such cases is model compression (Ganesh et al., 2020).

In this work, we present a novel approach to simultaneous transfer learning and model compression called **Weight Squeezing** where we learn the mapping from a teacher model's weights to a student model's weights.

We applied Weight Squeezing to a pre-trained teacher text classification model to obtain a smaller student model. We compared our method with common model compression approaches, including variations of Knowledge Distillation (Ba and Caruana, 2014; Hinton et al., 2015; Romero et al., 2014) without any reparameterizations and low-rank matrix factorization methods. Our experiments show that in most cases, Weight Squeezing achieves better performance than other methods.

We also proposed a method called Gated Weight Squeezing to improve fine-tuning BERT models. This method combines fine-tuning with mapping larger BERT-Base weights. We showed that Gated Weight Squeezing produces more accurate results than plain fine-tuning.

2 Related Work

Approaches to compress BERT include pruning (Voita et al., 2019; McCarley et al., 2019), parameter sharing (Lan et al., 2019), and knowledge distillation. Task-agnostic KD involves reducing the size of BERT itself (Sanh et al., 2019; Sun et al., 2020), while task-specific KD can be seen as fine-tuning BERT on a downstream task first, and then applying compression techniques to train a smaller model Turc et al. (2019).

Some of these methods can be combined to achieve better results (Mao et al., 2020). Low-rank matrix factorization approaches (e.g., SVD) also focus on reducing the size of model parameters.

3 Weight Squeezing

We now introduce a method to perform knowledge transfer and model compression by **learning the mapping between teacher and student weights**.

		W/O REPARAM.			WS			SVD		
CPU Time ↓		(×1, ×1)			(×1, ×1)			(×5.3, ×4.9)		
GPU Time ↓		(×1, ×1)			(×1, ×1)			(×2.2, ×2.2)		
		MLE	KD	KD-EO	MLE	KD	KD-EO	MLE	KD	KD-EO
SST2	32	82.7	82.1	83.1	82.8	83.7	**83.8**	82.0	81.3	82.0
	16	82.1	82.3	82.9	**84.1**	82.9	82.9	82.0	81.8	81.7
MNLI-mm	32	65.0	64.9	70.5	**71.5**	64.4	68.2	68.1	61.1	70.9
	16	57.0	59.4	59.2	57.3	57.3	**64.3**	56.3	55.3	60.8
COLA	32	17.3	18.1	17.4	17.0	19.3	**20.7**	18.3	17.7	18.0
	16	16.1	17.0	15.6	15.0	16.6	16.5	**16.8**	16.7	16.0
STSB	32	21.0	20.9	20.8	27.4	16.5	28.1	27.6	**28.8**	28.0
	16	20.4	21.1	21.2	17.9	19.9	25.1	**29.3**	27.3	27.5
MRPC	32	77.6	77.3	77.9	**79.0**	78.5	77.5	77.7	77.8	78.2
	16	**78.8**	78.2	78.0	78.1	78.5	78.2	76.5	76.7	77.5
QQP	32	76.6	77.7	77.0	76.2	76.6	76.1	**79.1**	76.6	78.4
	16	74.9	**75.7**	73.2	73.9	75.3	73.5	72.6	73.1	74.0
QNLI	32	61.3	61.6	63.4	74.3	66.7	**79.9**	64.5	64.9	76.8
	16	61.7	62.1	61.9	64.6	64.6	**68.6**	62.8	62.7	62.8
RTE	32	59.2	59.6	59.2	60.3	59.6	59.6	60.7	**61.0**	60.3
	16	58.5	58.8	59.2	57.4	60.3	**61.0**	58.5	58.5	58.1

Table 1: Accuracy on GLUE tasks for the Model Compression experiment (see Section 4). We report inference time results (lower is better) for each of the reparameterization methods. We refer to d as the model hidden size. We report inference time for CPU and GPU in "(d=16 time, d=32 time)" format. WS outperformed training without reparameterization.

Suppose we have a pre-trained teacher Transformer model with a large hidden state. For some linear layer l, we have a weight matrix Θ_l^t with the shape $n_l \times m_l$ (we will omit the l subscript later for simplicity).

We explore a case where the weights of a pre-trained teacher model are too big to run and store the model on an edge device. For this reason, we may want to train a student model with a smaller number of parameters. Let us say that we want the student model to make the weight matrix Θ^s at the same layer l to have the shape equal to $a \times b$, where $a \ll n$ and $b \ll m$.

In this approach, instead of training student model weights Θ^s from scratch, we reparameterize them as a trainable linear mapping from teacher model weights Θ^t. Doing so allows us to transfer knowledge stored in the teacher weights to the student weights.

$$\Theta^s = \mathcal{L}\Theta^t\mathcal{R} \qquad (1)$$

where $\mathcal{L}$ and $\mathcal{R}$ are randomly initialized trainable parameters of the mapping with shapes equal to $a \times n$ and $m \times b$ respectively .

At the same time, mapping of teacher biases and word embeddings is performed as a single linear mapping as follows:

$$\Theta^s_{single} = \Theta^t\mathcal{R} \qquad (2)$$

where biases are matrices of size $1 \times b$ and word embeddings have size $V \times b$, and V is the total number of words in the vocabulary.

After reparameterization of the student model weights using Equations 1 and 2, we train mapping weights $\mathcal{L}$ and $\mathcal{R}$ using plain negative log-likelihood (Weight Squeezing) or KD loss (Weight Squeezing combined with KD). When the mapping weights are trained, we compute student weights and then use them to make predictions dropping $\mathcal{L}$ and $\mathcal{R}$ matrices.

4 Model Compression with Weight Squeezing

In this work, we focus on applying Weight Squeezing for task-specific model compression. For this purpose, we fine-tuned the BERT-Medium model (41M parameters) on a particular dataset to obtain the pre-trained teacher model. We then applied Weight Squeezing to reparameterize weights of the significantly smaller target model (1M and 0.5M parameters, 40 and 80 times smaller than the teacher model, respectively.).

We trained all models on GLUE datasets (Wang et al., 2018). For each dataset, we trained a teacher model by fine-tuning the pre-trained BERT-Medium model.

We consider the following methods for reparameterization of student models: without weight reparametrization, Weight Squeezing, and SVD. Each of the baselines above could be trained with ambiguous methods. We used the following approaches: MLE, Knowledge Distillation (KD), Knowledge Distillation on Encoder Outputs (distillation on hidden states of teacher model).

Since we focused on making models smaller in terms of the overall number of parameters, we trained student models in two configurations of small hidden sizes equal to 16 and 32. For all models, we used the number of heads equal to 4, and a fixed number of Transformer layers equal to 8 for teacher models.

SA Heads:	4		8
Hidden size:	16	32	512
Plain BERT	0.52M	1.1M	41.3M
WS	4.18M	8.4M	-
SVD	0.53M	1.1M	-

Table 2: The number of trainable parameters for each model. Note that once the WS model is trained, we no longer have to store mapping matrices $\mathcal{L}$ and $\mathcal{R}$. Therefore, student models trained with WS will have their number of parameters equal to Plain BERT during inference.

For Weight Squeezing, we used fine-tuned teacher models as the source of the mapping for weight reparameterization. This way, we reparameterized all linear layers in the model as in Equation 1 and embedding vectors as in Equation 2. We optimized the loss with respect to the mapping parameters used to reparameterize the student model weights and the rest of the student model parameters that were not reparameterized (e.g., the layer normalization weights).

Note that in Low-Rank approach, we did not directly train the student model with the specified hidden state size as in Non Low-Rank Factorization methods (Non-LRMF). Instead, we injected a botteneck in the middle of each layer, which allowed us to reduce the total number of parameters in the model. For this reason, we evaluated the number of parameters for Non-LRMF models for hidden sizes equal to 16 and 32 and then found appropriate factorization ranks to make SVD models have an approximately similar number of parameters. This way, we compared Non-LRMF models of hidden size equal to 16 and 32 with SVD models with r equal to 2 and 7 (See Table 2).

4.1 Training Details

Hyperparameters for each model were tuned using Bayesian hyperparameter search. We performed a search for about 8-15 GPU days for each model with NVIDIA A100 GPU. We maximized the appropriate metric on each GLUE dataset dev split to find each method's best training configuration and report the best performing method's results.

We used Adam (Kingma and Ba, 2015) optimizer to train all models with linear warmup and linear decay of the learning rate. We also applied dropout to the attention matrix and to the averaged hidden state before the last linear layer, which produced logits of predictions.

5 Fine-Tuning with Gated Weight Squeezing

We also propose **Gated Weight Squeezing**, where we fine-tuned BERT-Base on specific task to obtain a large teacher network and then reparameterized the weights of a student model (in our experiments, we used BERT$_6$ with 66M parameters) as follows:

$$\Theta^s = (1 - \sigma(s)) \odot \mathcal{L}\Theta^t\mathcal{R} + \sigma(s) \odot \Theta^b \quad (3)$$

Θ^t are the weights of the teacher model (the ones after fine-tuning a BERT-Base model), Θ^b are the weights of BERT$_6$ which we want to fine-tune, s is a scalar value, σ is the sigmoid function, and $\odot$ is an elementwise multiplication. We used $\mathcal{L}$, $\mathcal{R}$, Θ^b, and s as trainable parameters. Embeddings were also reparameterized in a gated way according to Equation 2.

We compared plain fine-tuning and fine-tuning via Gated Weight Squeezing of the 66M BERT$_6$ model (6 layers, $d = 768$, 12 heads) on GLUE tasks in this experiment. We also compared results obtained with other models trained by plain fine-tuning of compressed BERTs (Sanh et al., 2019; Jiao et al., 2019; Wang et al., 2020; Lan et al., 2019).

5.1 Training Details

We first trained a teacher model by fine-tuning BERT-Base (109M parameters). Then we used this fine-tuned model and the BERT$_6$ model for the

Model	Params	SST2	CoLA	QNLI	RTE	MNLI-m
BERT_6*	66M	92.1	57.7	89.6	72.2	81.9
Gated WS*	66M	92.7	**60.8**	90.0	72.2	82.3
DistilBERT	66M	92.7	51.3	89.2	59.9	82.2
TinyBERT	66M	**93.0**	54.0	**91.1**	**73.4**	**84.5**
MiniLM	66M	92.0	49.2	91.0	71.5	84.0

Table 3: Accuracy on the GLUE tasks for fine-tuning with the Gated Weight Squeezing experiment (See Section 5 for more details). "*" denotes experiments conducted by us. Results for DistilBERT, TinyBERT, and MiniLM were obtained from the TinyBERT paper (Jiao et al., 2019), while results for ALBERT models were taken from Lan et al. (2019). BERT-Medium results refer to teacher models used in Model Compression experiments (see Section 4 and Table 1).

reparameterization of student model weights proposed in the Equation 3. This resulted in a student model with 66M parameters, which we compared with baseline fine-tuning. The reparameterized student model was trained with MLE loss.

Since BERT-Base and BERT_6 have a different number of hidden layers (12 and 6 respectively), we used the first 6 layers of BERT-Base to perform weight mapping.

We followed the same training strategy as in Section 4.1. We added a new hyperparameter for training the Gated WS model for the initial gate s value, which we used to select from a range equal to $[1; 4]$. In addition, we optimized parameter s and Θ^b separately from $\mathcal{L}$ and $\mathcal{R}$ with different learning rates.

6 Results

See Table 1 for the list of best model accuracy and relative speed measurements. We observed that WS generally produced better results than models trained without weight reparameterization. SVD were competitive to WS for some datasets due to the fact SVD operates with substantially bigger hidden states to evaluate attention. However, SVD models were significantly slower in inference than Non-LRMF (up to **2-5** times slower for SVD), which makes WS the more appropriate choice when speed is of essence.

6.1 Fine-Tuning

Fine-tuning of BERT_6 with Gated Weight Squeezing produced better results than plain fine-tuning without involving knowledge from the BERT-Base model. We also outperformed DistilBERT and MiniLM on most tasks. Although TinyBERT outperformed Gated WS on most tasks, which is the groundwork for the future.

7 Conclusion & Future Work

We introduced Weight Squeezing and Gated Weight Squeezing, a novel approach to knowledge transfer and model compression. Our work shows that it can compress pre-trained text classification models and create competitive lightweight and fast models.

We conducted experiments with GLUE and demonstrated that Weight Squeezing produced better results than models trained without weight reparameterization. In addition, our results show that Weight Squeezing could be a competitive alternative to Low-Rank Factorizing Methods in terms of accuracy, while being substantially faster.

Furthermore, we demonstrated that Gated Weight Squeezing could be used to fine-tune pretrained models, improving the resulting accuracy compared to plain fine-tuning.

While our current work focused on transferring knowledge to task-specific models, an important area of research is the application of WS to task-agnostic compression in order to create more applicable BERT models trained for MLM tasks.

Another important direction of research is experimenting with the initialization of mappings. We are interested in applying this method in domains beyond NLP to compress other types of layers (convolutional, etc.). In addition, reducing memory footprint during WS training is crucial to making the training more effective.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*, page 2654–2662.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing large-scale transformer-based models: A case study on bert. In *arXiv preprint arXiv:2002.11985*.

Geoffrey Hinton, Oriol Vinyals, , and Jeff Dean. 2015. Distilling the knowledge in a neural network. In *arXiv preprint arXiv:1503.02531*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. In *arXiv preprint arXiv:1909.10351*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Yaming Yang, Quanlu Zhang, Yunhai Tong, and Jing Bai. 2020. Ladabert: Lightweight adaptation of bert through hybrid model compression. In *arXiv preprint arXiv:2004.04124*.

J.S. McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structured pruning of a bert-based question answering model. In *arXiv preprint arXiv:1910.06360*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. In *arXiv preprint arXiv:2004.02984*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. In *arXiv preprint arXiv:1908.08962*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*.