Extreme Compression for Pre-trained Transformers Made Simple and Efficient

Xiaoxia Wu*, **Zhewei Yao***, **Minjia Zhang***, **Conglong Li, Yuxiong He**

Microsoft

{xiaoxiawu, zheweiyao, minjiaz, conglong.li, yuxhe}@microsoft.com

Abstract

Extreme compression, particularly ultra-low bit precision (binary/ternary) quantization, has been proposed to fit large NLP models on resource-constrained devices. However, to preserve the accuracy for such aggressive compression schemes, cutting-edge methods usually introduce complicated compression pipelines, e.g., multi-stage expensive knowledge distillation with extensive hyperparameter tuning. Also, they oftentimes focus less on smaller transformer models that have already been heavily compressed via knowledge distillation and lack a systematic study to show the effectiveness of their methods. In this paper, we perform a very comprehensive systematic study to measure the impact of many key hyperparameters and training strategies from previous works. As a result, we find out that previous baselines for ultra-low bit precision quantization are significantly under-trained. Based on our study, we propose a simple yet effective compression pipeline for extreme compression, named XtrmC. XtrmC demonstrates that (1) we can skip the pre-training knowledge distillation to obtain a 5-layer BERT while achieving better performance than previous state-of-the-art methods, e.g., the 6-layer TinyBERT; (2) extreme quantization plus layer reduction is able to reduce the model size by 50x, resulting in new state-of-the-art results on GLUE tasks.

1 Introduction

Over the past few years, we have witnessed the model size has grown at an unprecedented speed, from a few hundred million parameters (e.g., BERT [10], RoBERTa [28], DeBERTA [15], T5 [37],GPT-2 [36]) to a few hundreds of billions of parameters (e.g., 175B GPT-3 [6], 530B MT-NLT [46]), showing outstanding results on a wide range of language processing tasks. Despite the remarkable performance in accuracy, there have been huge challenges to deploying these models, especially on resource-constrained edge or embedded devices. Many research efforts have been made to compress these huge transformer models including knowledge distillation [20, 54, 53, 63], pruning [57, 43, 8], and low-rank decomposition [30]. Orthogonally, quantization focuses on replacing the floating-point weights of a pre-trained Transformer network with low-precision representation. This makes quantization particularly appealing when compressing models that have already been optimized in terms of network architecture.

Popular quantization methods include post-training quantization [45, 33, 29], quantization-aware training (QAT) [23, 4, 31, 22], and their variations [44, 24, 12]. The former directly quantizes trained model weights from floating-point values to low precision values using a scalar quantizer, which is simple but can induce a significant drop in accuracy. To address this issue, quantization-aware training directly quantizes a model during training by quantizing all the weights during the forward and using a straight-through estimator (STE) [5] to compute the gradients for the quantizers.

*Equal contribution. Code is released as a part of https://github.com/microsoft/DeepSpeed

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

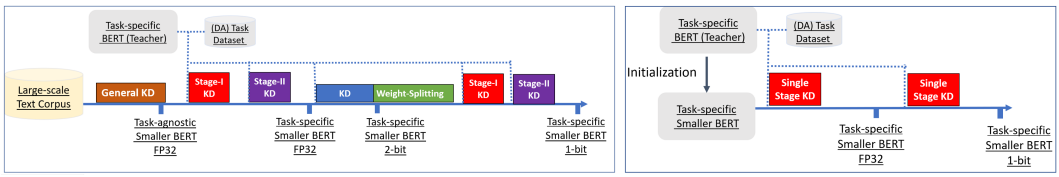

Figure 1: The left figure summarizes how to do 1-bit quantization for a layer-reduced model based on [20, 3]. It involves expensive pretraining on an fp-32 small model, task-specific training on 32-bit and 2-bit models, weight-splitting, and the final 1-bit model training. Along the way, it applies multi-stage knowledge distillation with data augmentation, which needs considerable hyperparameter tuning efforts. The right figure is our proposed method, XtrmC (see details in § 5), a simple while effective pipeline (see Figure 2 for highlighted results). Better read with a computer screen.

Recently, several QAT works further push the limit of BERT quantization to the extreme via ternarized (2-bit) ([61]) and binarized (1-bit) weights ([3]) together with 4-/8-bit quantized activation. These compression methods have been referred to as **extreme quantization** since the limit of weight quantization, in theory, can bring over an order of magnitude compression rates (e.g., 16-32 times). One particular challenge identified in [3] was that it was highly difficult to perform binarization as there exits a sharp performance drop from ternarized to binarized networks. To address this issue, prior work proposed multiple optimizations where one first trains a ternarized DynaBert [17] and then binarizes with weight splitting. In both phases, multi-stage distillation with multiple learning rates tuning and data augmentation [20] are used. Prior works claim these optimizations are essential for closing the accuracy gap from binary quantization.

While the above methodology is promising, several unanswered questions are related to these recent extreme quantization methods. First, as multiple ad-hoc optimizations are applied at different stages, the compression pipeline becomes very complex and expensive, limiting the applicability of extreme quantization in practice. Moreover, a systematical evaluation and comparison of these optimizations are missing, and the underlying question remains open for extreme quantization:

what are the necessities of ad-hoc optimizations to recover the accuracy loss?

Second, prior extreme quantization primarily focused on reducing the precision of the network. Meanwhile, several advancements have also been made in the research direction of knowledge distillation, where large teacher models are used to guide the learning of a small student model. Examples include DistilBERT [42], MiniLM [54, 53], MobileBERT [49], which demonstrate 2-4× model size reduction by reducing the depth or width without much accuracy loss through pre-training distillation with an optional fine-tuning distillation. Notably, TinyBERT [20] proposes to perform deep distillation in both the pre-training and fine-tuning stages and shows that this strategy achieves state-of-the-art results on GLUE tasks. However, most of these were done without quantization, and there are few studies about the interplay of extreme quantization with these heavily distilled models, which poses questions on

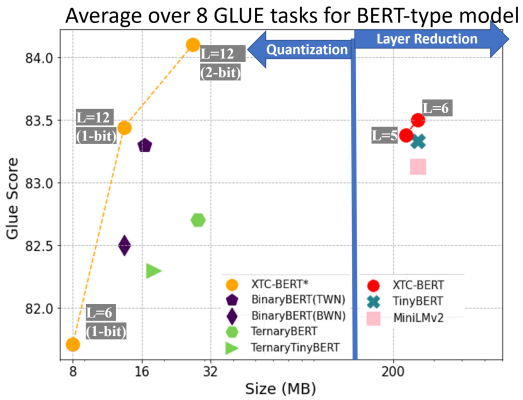

Figure 2: The comparison between XtrmC with other SOTA results.

to what extent, a smaller distilled model benefits from extreme quantization?

Contribution. To investigate the above questions, we make the following contributions:
(1) We present a systematic study of extreme quantization methods by fine-tuning ≥1000 pre-trained Transformer models, which includes a careful evaluation of the effects of hyperparameters and several

methods introduced in extreme quantization.
(2) We find that previous extreme quantization studies overlooked certain design choices, which lead to under-trained binarized networks and unnecessarily complex optimizations. Instead, we derive a celebrating recipe for extreme quantization, which is not only simpler but also allows us to achieve an even larger compression ratio and higher accuracy than existing methods (see Figure 1, right).
(3) We find that extreme quantization can be effectively combined with lightweight layer reduction, which allows us to achieve greater compression rates for pre-trained Transformers with better accuracy than prior methods while enjoying the additional benefits of flexibly adjusting the size of the student model for each use-case individually, without the expensive pre-training distillation.

Our evaluation results, as illustrated in Figure 2, show that our simple yet effective method can:
(1) compress $\text{BERT}_{\text{base}}$ to a 5-layer $\text{BERT}_{\text{base}}$ while achieving better performance than previous state-of-the-art distillation methods, e.g., the 6-layer TinyBERT [20], without incurring the computationally expensive pre-training distillation;
(2) reduce the model size by $50\times$ via performing robust extreme quantization on lightweight layer reduced models while obtaining better accuracy than prior extreme quantization methods, e.g., the 12-layer 1-bit $\text{BERT}_{\text{base}}$, resulting in new state-of-the-art results on GLUE tasks.

2 Related Work

Quantization becomes practically appealing as it not only reduces memory bandwidth consumption but also brings the additional benefit of further accelerating inference on supporting hardware [44, 24, 12]. Most quantization works focus on INT8 quantization [21] or mixed INT8/INT4 quantization [44]. Our work differs from those in that we investigate extreme quantization where the weight values are represented with only 1-bit or 2-bit at most. There are prior works that show the feasibility of using only ternary or even binary weights [61, 3]. Unlike those work, which uses multiple optimizations with complex pipelines, our investigation leads us to introduce a simple yet more efficient method for extreme compression with more excellent compression rates.

On a separate line of research, reducing the number of parameters of deep neural networks models with full precision (no quantization) has been an active research area by applying the powerful knowleadge distillation (KD) [16, 54], where a stronger teacher model guides the learning of another small student model to minimize the discrepancy between the teacher and student outputs. Please see Appendix A for a comprehensive literature review on KD. Instead of proposing a more advanced distillation method, we perform a well-rounded comparative study on the effectiveness of the recently proposed multiple-stage distillation [20].

3 Extreme Compression Procedure Analysis

This section presents several studies that have guided the proposed method introduced in Section 5. All these evaluations are performed with the General Language Understanding Evaluation (GLUE) benchmark [51], which is a collection of datasets for evaluating natural language understanding systems. For the subsequent studies, we report results on the development sets after compressing a pre-trained model (e.g., $\text{BERT}_{\text{base}}$ and TinyBERT) using the corresponding single-task training data.

Previous works [61, 3] on extreme quantization of transformer models state three hypotheses for what can be related to the difficulty of performing extreme quantization:

- Directly training a binarized BERT is complicated due to its irregular loss landscape, so it is better to first to train a ternarized model to initialize the binarized network.
- Specialized distillation that transfers knowledge at different layers (e.g., intermediate layer) and multiple stages is required to improve accuracy.
- Small training data sizes make extreme compression difficult.

3.1 Is staged ternary-binary training necessary to mitigate the sharp performance drop?

Previous works demonstrate that binary networks have more irregular loss surface than ternary models using curvature analysis. However, given that rough loss surface is a prevalent issue when training

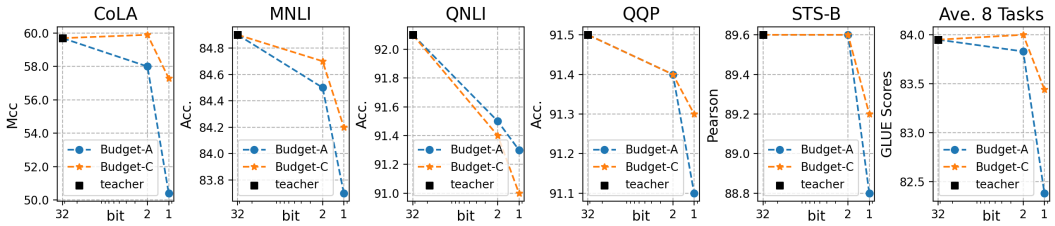

Figure 3: Performance of quantized $\text{BERT}_{\text{base}}$ with different weight bits and 8-bit activation on the GLUE Benchmarks. The results for orange and blue curves respectively represent the costs: (limited) Budget-A and (sufficient) Budget-C. The fp32-teacher scores are shown by black square marker.

neural networks, we question whether all of them have a causal relationship with the difficulty in extreme compression.

We are first interested in the observation from previous work [3] that shows directly training a binarized network leads to a significant accuracy drop (e.g., up to 3.8%) on GLUE in comparison to the accuracy loss from training a ternary network (e.g., up to 0.6%). They suggest that this is because binary networks have a higher training loss and overall more complex loss surface. To overcome this sharp accuracy drop, they propose a staged ternary-binary training strategy in which one needs first to train a ternarized network and use a technique called weight splitting to use the trained ternary weights to initialize the binary model.

It is not surprising that binarization leads to further accuracy drop as prior studies from computer vision also observe similar phenomenon [18, 50]. However, prior works also identified that the difficulty of training binarized networks mostly resides in training with insufficient number of iterations or having too large learning rates, which can lead to frequent sign changes of the weights that make the learning of binarized networks unstable [50]. Therefore, training a binarized network the same way as training a full precision network consistently achieves poor accuracy. Would increasing the training iterations and let the binarized network train longer under smaller learning rates help mitigate the performance drop from binarization?

To investigate this, we perform the following experiment: We remove the TernaryBERT training stage and weight splitting and directly train a binarized network. We use the same quantization schemes as in [3], e.g., a binary quantizer to quantize the weights: $w_i = \alpha \cdot sign(w_i), \alpha = \frac{1}{n}||w||_1$, and a uniform symmetric INT8 quantizer to quantize the activations. We apply the One-Stage quantization-aware KD that will be explained in § 3.2 to train the model. Nevertheless, without introducing the details of KD here, it does not affect our purpose of understanding the phenomena of a sharp performance drop since we only change the training iteration and learning rates while fixing other setups.

We consider three budgets listed in Table 1, which cover the practical scenarios of short, standard, and long training time, where Budget-A and Budget-B take into account the training details that appeared in BinaryBERT [3], TernaryBERT [61]. Budget-C has a considerably larger training budget but smaller than TinyBERT [3]. Meanwhile, we also perform a grid search of peak learning rates {2e-5, 1e-4, 5e-4}. For more training details on iterations and batch size per iteration, please see Table C.1.

Table 1: Training budgets for the GLUE tasks.

Dataset	Data Aug.	Training epochs: Budget-A	Budget-B	Budget-C
QQP/MNLI	✗	3	9	18 or 36
QNLI	✓	1	3	6 or 9
SST-2/STS-B/RTE	✓	1	3	12
CoLA/MRPC	✓	1	3	12 or 18

Table 2: **1-bit** quantization for $\text{BERT}_{\text{base}}$ with various Budget-A, Budget-B and Budget-C.

#	Cost	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Acc Drop
0	Teacher	59.7	84.9/85.6	90.6/86.3	92.1	88.6/91.5	72.2	93.2	90.1/89.6	83.95	-
1	Budget-A	50.4	83.7/84.6	90.0/85.8	91.3	88.0/91.1	72.9	92.8	88.8/88.4	82.38	1.57
2	Budget-B	55.6	84.1/84.4	90.4/86.0	90.8	88.3/91.3	72.6	93.1	88.9/88.5	82.98	0.97
3	Budget-C	57.3	84.2/84.4	90.7/86.5	91.0	88.3/91.3	74.0	93.1	89.2/88.8	83.44	0.51

We present our main results in Table 2 and Figure 3 (see Table C.2 a full detailed results including three learning rates). We observe that although training binarized BERT with more iterations does not fully close the accuracy gap to the uncompressed full-precision model, the sharp accuracy drop from ternarization to binarization (the blue curve v.s. the orange curve in Figure 3) has largely been mitigated, leaving a much smaller accuracy gap to close. For example, when increasing the training time from Budget-A to Budget-C, the performance of CoLA boosts from 50.4 to 57.3, and the average score improves from 82.38 to 83.44 (Table 2). These results indicate that previous studies [3] on binarized BERT were significantly under-trained. It also suggests that the observed sharp accuracy drop from binarization is primarily an optimization problem. We note that in the context of fine-tuning BERT models, Mosbach et al. [32] also observe that fine-tuning BERT can suffer from vanishing gradients, which causes the training converges to a "bad" valley with sub-optimal training loss. As a result, the authors also suggest increasing the number of iterations to train the BERT model to have stable and robust results.

Finding 1. A longer training iterations with learning rate decay is highly preferred for closing the accuracy gap of extreme quantization.

The above claim seems natural. However, we argue that special considerations need to be taken for effectively doing extreme quantization before resorting to more complex solutions.

3.2 The role of multi-stage knowledge distillation

The above analysis shows that the previous binarized BERT models were severely undertrained. In this section, we further investigate the necessity of multi-stage knowledge distillation, which is quite complex because each stage has its own set of hyperparameters. Still, prior works claim multi-stage knowledge distillation to be crucial for improving the accuracy of extreme quantization. Investigating this problem is interesting because it can potentially admit a more straightforward solution with a cheaper cost.

Prior work proposes an interesting knowledge distillation (KD), and here we call it *Two-Stage* KD (2S-KD) [20], which has been applied for extreme quantization [3]. The 2S-KD minimizes the losses of hidden states $\mathcal{L}_{\text{hidden}}$ and attention maps $\mathcal{L}_{\text{att}}$ and losses of prediction logits $\mathcal{L}_{\text{logit}}$ in two separate steps, where different learning rates are used in these two stages, e.g., a $2.5\times$ larger learning rate is used for Stage-I than that for Stage-II. However, how to decide the learning rates and training epochs for these two stages is not well explained and will add additional hyperparameter tuning costs. We note that this strategy is very different from the deep knowledge distillation strategy used in [48, 53, 49], where knowledge distillation is performed with a single stage that minimizes the sum of the losses from prediction logits, hidden states, and attention maps. Despite the promising results in [3], the mechanism for why multi-stage KD improves accuracy is not well understood.

Generally, the knowledge distillation for transformer models can be formulated as minimizing the following objective:

$$\min_{\theta} \mathbb{E}_{x\sim D}[\gamma\mathcal{L}_{\text{logit}}(x;\theta) + \beta(\mathcal{L}_{\text{att}}(x;\theta) + \mathcal{L}_{\text{hidden}}(x;\theta))] \tag{1}$$

where $\mathcal{L}_{\text{logit}}$ denote the loss (e.g., KL divergence or mean square error) between student's and teacher's prediction logits, and $\mathcal{L}_{hidden}$ and $\mathcal{L}_{\text{att}}$) measures the loss of hidden states and attention maps. $\gamma \in \{0,1\}, \beta \in \{0,1\}$ are hyperparameters. See the detailed mathematical definition in § B. To investigate the effectiveness of multi-stage, we compare three configurations (shown in Figure 4 in the appendix):

(1) **1S-KD (One-stage KD):** $(\gamma,\beta) = (1,1)$ for $t \leq \mathcal{T}$;

(2) **2S-KD (Two-stage KD):** $(\gamma,\beta) = (0,1)$ if $t < \mathcal{T}/2$; $(\gamma,\beta) = (1,0)$ if $\mathcal{T}/2 \leq t \leq \mathcal{T}$;

(3) **3S-KD (Three-stage KD):** $(\gamma,\beta) = (0,1)$ if $t < \mathcal{T}/3$; $(\gamma,\beta) = (1,1)$ if $\mathcal{T}/3 \leq t \leq 2\mathcal{T}/3$; $(\gamma,\beta) = (1,0)$ if $2\mathcal{T}/3 \leq t \leq \mathcal{T}$.

where $\mathcal{T}$ presents the total training budget and t represents the training iterations.

Notably, we created the 3S-KD, where we add a transition stage: $\mathcal{L}_{\text{hidden}} + \mathcal{L}_{\text{att}} + \mathcal{L}_{\text{logit}}$ in between the first and second stage of the 2S-KD. The idea is to make the transition of the training objective

smoother in comparison to the 2S-KD. In this case, the learning rate schedules will be correspondingly repeated in three times, and the peak learning rate of Stage-I is $\times 2.5$ higher than that of Stage-II and Stage-III. We apply all three KD methods independently (under a fixed random seed) for binary quantization across different learning rates {2e-5, 1e-4, 5e-4} under the same training budget. We do not tune any other hyperparameters.

Table 3 shows the results under Budget-A (see Table C.2 for Budget-B/C). We note that when the learning rate is fixed to 2e-5 (i.e., Rows 1, 4, and 7), the previously proposed 2S-KD [20] (81.46) does show performance benefits over 1S-KD (80.33). Our newly created 3S-KD (81.57) obtains even better accuracy under the same learning rate. However, multi-stage makes the extreme compression procedure complex and inefficient because it needs to set the peak learning rates differently for different stages. Surprisingly, the simple 1S-KD easily outperforms both 2S-KD and 3S-KD when we slightly increase the search space of the learning rates (e.g., 1e-4 and 5e-4). This means we can largely omit the burden of tuning learning rates and training epochs for each stage in multi-stage by using single-stage but with the same training iteration budget.

Finding 2. Single-stage knowledge distillation with more training budgets and search of learning rates is sufficient to match or even exceed accuracy from multi-stage ones.

Table 3: **1-bit** quantization for $\text{BERT}_{\text{base}}$ with three different KD under the training Budget-A.

#	Stages	learning rate	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. a row	Avg. the best
1	One-Stage	2e-5	44.6	83.1/83.7	88.8/83.1	91.1	87.4/90.7	66.1	**92.8**	87.8/87.5	80.33	**82.38**
2		1e-4	**50.4**	**83.7/84.6**	**90.1/85.3**	**91.3**	**88.0/91.1**	71.5	92.7	**88.8/88.4**	82.16	
3		5e-4	42.3	83.3/84.1	90.0/85.8	89.5	87.8/90.9	**72.9**	92.5	88.1/87.8	81.04	
4	Two-Stage	2e-5	48.2	**83.3/83.8**	89.3/84.6	**90.7**	**87.7/90.9**	**70.4**	**92.5**	**88.7/88.4**	81.46	81.59
5		1e-4	**48.5**	83.3/83.4	**90.0/85.5**	90.4	87.4/90.7	70.4	92.4	88.6/88.2	81.47	
6		5e-4	16.2	74.9/76.4	89.7/84.8	87.8	85.0/89.0	68.2	91.6	86.2/86.5	75.01	
7	Three-Stage	2e-5	**49.3**	**83.1/83.6**	89.5/84.3	**91.0**	**87.6/90.9**	70.8	**92.4**	88.7/88.3	81.57	81.93
8		1e-4	49.0	83.2/83.4	**89.7/84.8**	90.9	87.6/90.7	**73.6**	92.2	**88.8/88.5**	81.84	
9		5e-4	29.5	81.3/81.9	89.1/83.6	88.4	85.0/89.3	66.1	91.9	84.1/83.9	77.34	

3.3 The importance of data augmentation

Prior works augment the task-specific datasets using a text-editing technique by randomly replacing words in a sentence with their synonyms, based on their similarity measure on GloVe embeddings [20]. They then use the augmented dataset for task-specific compression of BERT models, observing improved accuracy for extreme quantization [61, 3]. They hypothesized that data augmentation (DA) is vital for compressing transformer models. However, as we have observed that the binarized networks were largely under-trained and had no clear advantage of multi-stage versus single-stage under a larger training budget, it raises the question of the necessity of DA.

To better understand the importance of DA for extreme compression, we compare the end-task performance of both 1-bit quantized $\text{BERT}_{\text{base}}$ models with and without data augmentation, based on previous findings in this paper that make extreme compression more robust. Table 4 shows results for the comparison of the 1-bit $\text{BERT}_{\text{base}}$ model under two different training budgets. Note that MNLI /QQP does not use DA; we repeat the results. We find that regardless of whether shorter or longer iterations, removing DA leads to an average of 0.66 and 0.77 points drop in average GLUE score. Notably, the accuracy drops more than 1 point on smaller tasks such as CoLA, MPRC, RTE, STS-B. Furthermore, similar performance degradation is observed when removing DA from training an FP32 half-sized BERT model (more details will be introduced in § 4), regardless of whether using 1S-KD or 2S-KD. These results indicate that DA is helpful, especially for extreme compressed models; DA significantly benefits from learning more diverse/small data to mitigate accuracy drops.

Finding 3. Training without DA hurts performance on downstream tasks for various compression tasks, especially on smaller tasks.

We remark that although our conclusion for DA is consistent with the finding in [3], we have performed a far more well-through experiment than [3] where they consider a single budget and 2S-KD only.

Table 4: **The Comparison between the results with and without data augmentation (DA).** Row 1-4 is for $\text{BERT}_{\text{base}}$ under 1-bit quantization using 1S-KD. Row 5-8 is for a layer-reduced $\text{BERT}_{\text{base}}$ (six-layer) under Budget-C without quantization (please see § 4 for more details).

#	Cost or Stages	Data Aug.	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	DIFF
1	Budget-A	✓	50.4	83.7/84.6	90.0/85.8	91.3	88.0/91.1	72.9	92.8	88.8/88.4	82.38	-0.77
2		✗	48.9	83.7/84.6	89.0/83.6	91.0	88.0/91.1	71.5	92.5	87.6/87.5	81.61	
3	Budget-C	✓	57.3	84.2/84.4	90.7/86.5	91.0	88.3/91.3	74.0	93.1	89.2/88.8	83.44	-0.66
4		✗	55.0	84.2/84.4	90.0/85.0	90.7	88.3/91.3	73.3	92.9	88.2/87.9	82.78	
5	One-stage	✓	56.9	84.5/84.9	89.7/84.8	91.7	88.5/91.5	71.8	93.5	89.8/89.4	83.27	-2.18
6		✗	52.2	84.5/84.9	88.4/82.4	91.4	88.5/91.5	63.9	92.9	86.5/86.3	81.09	
7	Two-stage	✓	56.0	84.1/84.2	89.7/84.8	91.5	88.2/91.3	71.8	93.3	89.4/89.1	82.93	-2.03
8		✗	50.2	84.1/84.2	88.7/83.1	90.8	88.2/91.3	64.6	92.9	86.9/86.7	80.90	

4 Interplay of KD, Long Training, Data Augmentation and Layer Reduction

In the previous section, we found strong evidence for statistically significant benefits from 1S-KD compared to 2S-KD and 3S-KD. Moreover, a considerable long training can greatly lift the score to the level same as the full precision teacher model. Although 1-bit $\text{BERT}_{\text{base}}$ already achieves $\times 32$ smaller size than its full-precision counterpart, a separate line of research focuses on changing the model architecture to reduce the model size. Notably, one of the promising methods – reducing model sizes via knowledge distillation – has long been studied and shown good performances [13, 60, 42, 53, 49].

While many works focus on innovating better knowledge distillation (KD) pretraining methods for higher compression ratio and better accuracy, given our observations in Section 3 that extreme quantization with long training and data augmentation can effectively reduce the model size, we are interested in *to what extent can a layer-reduced model benefits from the pretraining with KD?*

To investigate this issue, we perform the following experiments: We prepare five student models, all having 6 layers with the same hidden dimension 768: (1) a pretrained TinyBERT_6 [20][2]; (2) MiniLMv2_6 [53][3]; (3) Top-BERT_6: using the top 6 layers of the fine-tuned $\text{BERT}_{\text{base}}$ model to initialize the student model; (4) Bottom-BERT_6: using the bottom six layers of the fine-tuned $\text{BERT}_{\text{base}}$ model to initialize the student; (5) Skip-BERT_6: using every other layer of the fine-tuned $\text{BERT}_{\text{base}}$ model to initialize the student. In all cases, we fine-tune $\text{BERT}_{\text{base}}$ for each task as the teacher model (see the teachers' performance in Table 5, Row 1). We choose TinyBERT and MiniLM because they are the state-of-the-art for knowledge distillation of BERT models. We choose the other three configurations because prior work [48, 34] also suggested that layer pruning is also a practical approach for task-specific compression. For the initialization of the student model, both TinyBERT_6 and MiniLM_6 are initialized with weights distilled from $\text{BERT}_{\text{base}}$ through pretraining distillation, and the other three students (e.g., top, bottom, skip) are obtained from the fine-tuned teacher model *without incurring any pretraining training cost*. We apply 1S-KD as we also verify

Table 5: Pre-training does not show benefits for layer reduction. Row 3 (rep.*) is a reproduced result by following the training recipe in [20].

#	Model	size	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	Acc. Drop
1	fp32 $\text{BERT}_{\text{base}}$ (teacher)	417.2	59.7	84.9/85.6	90.6/86.3	92.1	88.6/91.5	72.2	93.2	90.1/89.6	83.95	-
Training cost: greater than Budget-C (see [20] or § B)												
2	Pretrained TinyBERT_6 ([20])	255.2 ($\times$1.6)	54.0	84.5/84.5	90.6/86.3	91.1	88.0/91.1	73.4	93.0	90.1/89.6	83.11	-0.84
3	Pretrained TinyBERT_6 (rep.*)	255.2 ($\times$1.6)	56.9	84.4/84.8	90.1/85.5	91.3	88.4/91.4	72.2	93.2	90.3/90.0	83.33	-0.62
Training cost: Budget-C												
4	Pretrained TinyBERT_6	255.2 ($\times$1.6)	54.4	84.6/84.3	90.4/86.3	91.5	88.5/91.5	69.7	93.3	89.2/89.0	82.76	-1.19
5	Pretrained MiniLM_6-v2	255.2 ($\times$1.6)	55.4	84.4/84.5	90.7/86.5	91.4	88.5/91.5	71.8	93.3	89.4/89.0	83.13	-0.82
6	Skip-BERT_6 (ours)	255.2 ($\times$1.6)	56.9	84.6/84.9	90.4/85.8	91.8	88.6/91.6	72.6	93.5	89.8/89.4	**83.50**	-0.45
7	Skip-BERT_5 (ours)	228.2 ($\times$1.8)	57.9	84.3/85.1	90.1/85.5	91.4	88.5/91.5	72.2	93.3	89.2/88.9	**83.38**	-0.57
8	Skip-BERT_4 (ours)	201.2 ($\times$2.1)	53.3	83.2/83.4	90.0/85.3	90.8	88.2/91.3	70.0	93.5	88.8/88.4	82.18	-1.77

that 2S-KD and 3S-KD are under-performed (See Table C.7 in the appendix). We set Budget-C as our training cost because the training budget in reproducing the results for TinyBERT is greatly larger than Budget-C illustrated above. We report their best validation performances in Table 5 across the three learning rate {5e-5, 1e-4, 5e-4}. We apply 1S-KD and budget-C for this experiment. We

[2]The checkpoint of TinyBERT is downloaded from their uploaded huggingface.co.

[3]The checkpoint of miniLMv2 is from their github.

report their best validation performances in Table 5 across three learning rates {5e-5, 1e-4, 5e-4}. For complete statistics with these learning rates, please refer to Table C.5. We make a few observations:

First, perhaps a bit surprisingly, the results in Table 5 show that there are no significant improvements from using the more computationally expensive pre-pretraining distillation in comparison to lightweight layer reduction: Skip-BERT$_6$ (Row 7) achieves the highest average score 83.50. This scheme achieves the highest score on larger and more robust tasks such as MNLI/QQP/QNLI/SST-2.

One noticeable fact in Table 5 is that under the same budget (Budget-C), the accuracy drop of Skip-BERT$_6$ is about 0.74 (0.37) higher than TinyBERT$_6$ in Row 5 (MINILM$_6$-v2 in Row 6). Note that layer reduction without pretraining has also been addressed in [41]. However, their KD is limited to logits without DA, and the performance is not better the pretrained DistillBERT, which shows pretraining was useful (if only logits are used in KD).

Second, with the above encouraging results from a half-size model (Skip-BERT$_6$), we squeeze the depths into five and four layers. The five-/four-layer student is initialized from ℓ-layer of teacher with $\ell \in \{3, 5, 7, 9, 11\}$ or $\ell \in \{3, 6, 9, 12\}$. We apply the same training recipe as Skip-BERT$_6$ and report the results in Row 8 and 9 in Table 5. There is little performance degradation in our five-layer model (83.28) compared to its six-layer counterpart (83.50); Interestingly, CoLA and MNLI-mm even achieve higher accuracy with smaller model sizes. Meanwhile, when we perform even more aggressive compression by reducing the depth to 4, the result is less positive as the average accuracy drop is around 1.3 points.

Third, among three lightweight layer reduction methods, we confirm that a Skip-# student performs better than those using Top-# and Bottom-#. We report the full results of this comparison in the Appendix Table C.6. This observation is consistent with [20]. However, their layerwise distillation method is not the same as ours, e.g, they use the non-adapted pretrained BERT model to initialize the student, whereas we use the fine-tuned BERT weights. We remark that our finding is in contrast to [41] which is perhaps because the KD in [41] only uses logits distillation without layerwise distillation.

Finding 4. Lightweight layer reduction matches or even exceeds expensive pre-training distillation for task-specific compression.

5 Proposed Method for Further Pushing the Limit of Extreme Compression

Based on our studies, we propose a simple yet effective method tailored for extreme compression. Figure 1 (right) or Figure 7 illustrates our proposed method: XtrmC, which consists of 2 steps:

Step I: Lightweight layer reduction. Unlike the common layer reduction method where the layer-reduced model is obtained through computationally expensive pre-training distillation, we select a subset of the fine-tuned teacher weights as a lightweight layer reduction method (e.g., either through simple heuristics as described in Section 4 or search-based algorithm as described in [34]) to initialize the layer-reduced model. When together with the other training strategies identified in this paper, we find that such a lightweight scheme allows for achieving a much larger compression ratio while setting a new state-of-the-art result compared to other existing methods.

Step II: 1-bit quantization by applying 1S-KD with DA and long training. Once we obtain the layer-reduced model, we apply the quantize-aware 1S-KD, proven to be the most effective in § 3.2. To be concrete, we use an ultra-low bit (1-bit/2-bit) quantizer to compress the layer-reduced model weights for a forward pass and then use STE during the backward pass for passing gradients. Meanwhile, we minimize the single-stage deep knowledge distillation objective with data augmentation enabled and longer training Budget-C (such that the training loss is close to zero).

Evaluation Results. The results of our approach are presented in Table 6, which we include multiple layer-reduced models with both 1-bit and 2-bit quantization. We make the following observations:
(1) For the 2-bit + 6L model (Row 1 and Row 2), XtrmC achieves 0.63 points higher accuracy than 2-bit quantized model TernaryBERT [61]. This remarkable improvement consists of two factors: (a) our fp-32 layer-reduced model is better, and (b) the 1-bit model is trained under larger budget and three learning rate searches. To understand how much the factor-(b) benefits, we may refer to the accuracy drop from their fp32 counterparts (last col.) where ours only drops 0.44 points and

Table 6: 1-/2-bit quantization of the layer-reduced model. The last column (Acc. drop) is the accuracy drop from their own fp-32 models. See full details in Table C.8.

#	Method	bit (#-layer)	size (MB)	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	Acc. drop
1	[61]	2 (6L)	16.0 (×26.2)	53.0	83.4/83.8	91.5/88.0	89.9	87.2/90.5	71.8	93.0	86.9/86.5	82.26	-0.76
2	Ours	2 (6L)	16.0 (×26.2)	53.8	83.6/84.2	90.5/86.3	90.6	88.2/91.3	73.6	93.6	89.0/88.7	**82.89**	-0.44
3		2 (5L)	14.2 (×29.3)	53.9	83.3/84.1	90.4/86.0	90.4	88.2/91.2	71.8	93.0	88.4/88.0	**82.46**	-0.72
4		2 (4L)	12.6 (×33.2)	50.3	82.5/83.0	90.0/85.3	89.2	87.8/91.0	69.0	92.8	87.9/87.4	81.22	-0.90
5		1 (6L)	8.0 (×52.3)	52.5	83.4/83.8	90.0/85.3	89.4	87.9/91.1	68.6	93.1	88.4/88.0	81.71	-1.51
6		1 (5L)	7.1 (×58.5)	52.2	82.9/83.2	89.9/85.0	88.5	87.6/90.8	69.3	92.9	87.3/87.0	81.34	-1.84
7		1 (4L)	6.3 (×66.4)	48.3	82.0/82.3	89.9/85.5	87.7	86.9/90.4	63.9	92.4	87.1/86.7	79.96	-2.01

TinyBERT$_6$ drops 0.76. **(2)** When checking between our 2-bit five-layer (Row 3, 82.46) in Table 6 and 2-bit TinyBERT$_6$ (Row 1, 82.26), our quantization is much better, while our model size is 12.5% smaller, setting a new state-of-the-art result for 2-bit quantization with this size. **(3)** Let us take a further look at accuracy drops (last column) for six/five/four layers, which are 0.44 (1.51), 0.72 (1.84), and 0.9 (2.01) respectively. It is clear that smaller models become much more brittle for extreme quantization than BERT$_{\text{base}}$, especially the 1-bit compression as the degradation is $\times 3$ or $\times 4$ higher than the 2-bit quantization.

Besides the reported results above, we have also done a well-thorough investigation on the effectiveness of adding a low-rank (LoRa) full-precision weight matrices to see if it will further push the accuracy for the 1-bit layer-reduced models. However, we found a negative conclusion on using LoRa. Interestingly, if we repeat another training with the 1-bit quantization after the single-stage long training, there is another 0.3 increase in the average GLUE score (see Table C.14). Finally, we also verify that prominent teachers can indeed help to improve the results (see appendix).

6 Conclusions

We carefully design and perform extensive experiments to investigate the contemporary existing extreme quantization methods [20, 3] by fine-tuning pre-trained BERT$_{\text{base}}$ models with various training budgets and learning rate search. Unlike [3], we find that there is no sharp accuracy drop if long training with data augmentation is used and that multi-stage KD and pretraining introduced in [20] is not a must in our setup. Based on the finding, we derive a user-friendly celebrating recipe for extreme quantization (see Figure 1, right), which allows us to achieve a larger compression ratio and higher accuracy. See our summarized results in Table 7.[4]

Discussion and future work. Despite our encouraging results, we note that there is a caveat in our **Step I** when naively applying our method (i.e., initializing the student model as a subset of the teacher) to reduce the width of the model as the weight matrices dimension of teacher do not fit into that of students. We found that the accuracy drop is considerably sharp. Thus, in this domain, perhaps pertaining distillation [20] could be indeed useful, or Neural Architecture Search (AutoTinyBERT [58]) could be another promising direction. We acknowledge that, in order to make a fair comparison with [61, 3], our investigations are based on the classical 1-bit [39] and 2-bit [27] algorithms and without quantization on layernorm [2].

Table 7: Summary of task-specific performance of MNLI and GLUE scores. Also see Figure 2.

#	Model (8-bit activation)	size (MB)	MNLI-m/-mm	GLUE Score
1	BERT$_{\text{base}}$ (Teacher)	417.2 (×1.0)	84.9/85.6	83.95
2	TinyBERT$_6$ (rep*)	255.2 (×1.6)	84.4/84.8	83.33
3	XtrmC-BERT$_6$ (ours)	255.2 (×1.6)	**84.6**/84.9	**83.50**
4	XtrmC-BERT$_5$ (ours)	228.2 (×1.8)	84.3/**85.1**	**83.38**
5	2-bit BERT [61]	26.8 (×16.0)	83.3/83.3	82.73
6	2-bit XtrmC-BERT (ours)	26.8 (×16.0)	**84.6/84.7**	**84.10**
7	1-bit BERT (TWN) [3]	16.5 (×25.3)	84.2/**84.7**	82.49
8	1-bit BERT (BWN) [3]	13.4 (×32.0)	84.2/84.0	82.29
9	1-bit XtrmC-BERT (ours)	13.4 (×32.0)	84.2/84.4	**83.44**
10	2-bit TinyBERT$_6$ [61]	16.0 (×26.2)	83.4/83.8	82.26
11	2-bit XtrmC-BERT$_6$ (ours)	16.0 (×26.2)	**83.6/84.2**	**82.89**
12	2-bit XtrmC-BERT$_5$ (ours)	14.2 (×29.3)	83.3/84.1	**82.46**
13	1-bit XtrmC-BERT$_6$ (ours)	8.0 (×52.3)	83.4/83.8	81.71
14	1-bit XtrmC-BERT$_5$ (ours)	7.1 (×58.5)	82.9/83.2	81.34

Exploring the benefits of long training on an augmented dataset with different 1-/2-bit algorithms (with or without quantization on layer norm) would be of potential interests [24, 35]. Finally, as our experiments focus on BERT$_{\text{base}}$ model, future work can be understanding how our conclusion transfers to decoder models such as GPT-2/3 [36, 6].

[4] We decide not to include many other great works [26, 59, 42, 35] since they do not use data augmentation or their setups are not close.

References

[1] Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 7350–7357. AAAI Press, 2020.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, Online, August 2021. Association for Computational Linguistics.

[4] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 31, 2018.

[5] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[7] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[8] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020.

[9] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220, 2013.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, NAACL-HLT 2019)*, pages 4171–4186, 2019.

[11] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[12] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019.

[13] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020.

[14] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[15] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2020.

[16] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[17] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

[18] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4107–4115, 2016.

[19] Shankar Iyer, Nikhil Dandekar, and Kornl Csernai. First quora dataset release: Question pairs, 2017. *URL https://data. quora. com/First-Quora-Dataset-Release-Question-Pairs*, 2017.

[20] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online, November 2020. Association for Computational Linguistics.

[21] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 8018–8025. AAAI Press, 2020.

[22] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 19–28, 2017.

[23] Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

[24] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR, 2021.

[25] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, 2021.

[26] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[27] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[29] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34, 2021.

[30] Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Quanlu Zhang, Yaming Yang, Yunhai Tong, and Jing Bai. Ladabert: Lightweight adaptation of BERT through hybrid model compression. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3225–3234. International Committee on Computational Linguistics, 2020.

[31] Naveen Mellempudi, Abhisek Kundu, Dheevatsa Mudigere, Dipankar Das, Bharat Kaul, and Pradeep Dubey. Ternary neural networks with fine-grained quantization. *arXiv preprint arXiv:1705.01462*, 2017.

[32] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*, 2021.

[33] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

[34] David Peer, Sebastian Stabinger, Stefan Engl, and Antonio Rodríguez-Sánchez. Greedy-layer pruning: Speeding up transformer models for natural language processing. *Pattern Recognition Letters*, 157:76–82, 2022.

[35] Haotong Qin, Yifu Ding, Mingyuan Zhang, Qinghua YAN, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. BiBERT: Accurate fully binarized BERT. In *International Conference on Learning Representations*, 2022.

[36] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[38] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[39] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

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

[41] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. Poor man's BERT: smaller and faster transformer models. *CoRR*, abs/2004.03844, 2020.

[42] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[43] Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020.

[44] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821, 2020.

[45] Gil Shomron, Freddy Gabbay, Samer Kurzum, and Uri Weiser. Post-training sparsity-aware quantization. *Advances in Neural Information Processing Systems*, 34, 2021.

[46] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv e-prints*, pages arXiv–2201, 2022.

[47] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[48] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4322–4331, 2019.

[49] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, 2020.

[50] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 2625–2631. AAAI Press, 2017.

[51] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations*, 2019.

[52] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *CoRR*, abs/2004.05937, 2020.

[53] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. *CoRR*, abs/2012.15828, 2020.

[54] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[55] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

[56] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[57] Zhewei Yao, Linjian Ma, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. Mlpruning: A multilevel structured pruning framework for transformer-based models. *arXiv preprint arXiv:2105.14636*, 2021.

[58] Yichun Yin, Cheng Chen, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Autotinybert: Automatic hyper-parameter optimization for efficient pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5146–5157, 2021.

[59] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 811–824. IEEE, 2020.

[60] Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in Neural Information Processing Systems*, 33:14011–14023, 2020.

[61] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. Ternary-BERT: Distillation-aware ultra-low bit BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 509–521, Online, November 2020. Association for Computational Linguistics.

[62] Sanqiang Zhao, Raghav Gupta, Yang Song, and Denny Zhou. Extremely small BERT models from mixed-vocabulary training. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 2753–2759, 2021.

[63] Minjia Zhng, Niranjan Uma Naresh, and Yuxiong He. Adversarial data augmentation for task-specific knowledge distillation of pre-trained transformers. *Association for the Advancement of Artificial Intelligence*, 2022.

A Additional Related Work

KD has been extensively applied to computer vision and NLP tasks [52] since its debut. On the NLP side, several variants of KD have been proposed to compress BERT [10], including how to define the knowledge that is supposed to be transferred from the teacher BERT model to the student variations. Examples of such knowledge definitions include output logits (e.g., DistilBERT [42]) and intermediate knowledge such as feature maps [48, 1, 62] and self-attention maps [54, 49] (we refer KD using these additional knowledge as deep knowledge distillation [54]). To mitigate the accuracy gap from reduced student model size, existing work has also explored applying knowledge distillation in the more expensive pre-training stage, which aims to provide a better initialization to the student for adapting to downstream tasks. As an example, MiniLM [54] and MobileBERT [49] advance the state-of-the-art by applying deep knowledge distillation and architecture change to pre-train a student model on the general-domain corpus, which then can be directly fine-tuned on downstream tasks with good accuracy. TinyBERT [20] proposes to perform deep distillation in both the pre-training and fine-tuning stage and shows that these two stages of knowledge distillation are complementary to each other and can be combined to achieve state-of-the-art results on GLUE tasks. Our work differs from these methods in that we show how lightweight layer reduction without expensive pre-training distillation can be effectively combined with extreme quantization to achieve another order of magnitude reduction in model sizes for pre-trained Transformers.

B Additional Details on Methodology, Experimental Setup and Results

B.1 Knowledge Distillation

Knowledge Distillation (KD) [16] has been playing the most significant role in overcoming the performance degradation of model compression as the smaller models (i.e., student models) can absorb the rich knowledge of those uncompressed ones (i.e., teacher models) [40, 25, 43, 14]. In general, KD can be expressed as minimizing the difference ($\mathcal{L}_{kd}$) between outputs of a teacher model T and a student model S. As we are here studying transformer-based model and that the student and teacher models admit the same number (denoted as h) of attention heads, our setup focuses on a particular widely-used paradigm, layerwise KD [20, 61, 3]. That is, *for each layer* of a student model S, the loss in the knowledge transferred from teacher T consist of three parts:

(1) the fully-connected hidden states: $\mathcal{L}_{\text{hidden}} = \mathbf{MSE}(\mathbf{H}^S, \mathbf{H}^T)$, where $\mathbf{H}^S, \mathbf{H}^T \in \mathbb{R}^{l \times d_h}$;

(2) the attention maps: $\mathcal{L}_{\text{att}} = \frac{1}{h}\sum_{i=1}^{h} \mathbf{MSE}(\mathbf{A}_i^S, \mathbf{A}_i^T)$, where $\mathbf{A}_i^S, \mathbf{A}_i^T \in \mathbb{R}^{l \times l}$;

(3) the prediction logits: $\mathcal{L}_{\text{logit}} = \mathbf{CE}(\mathbf{p}^S, \mathbf{p}^T)$ where $\mathbf{p}^S, \mathbf{p}^T \in \mathbb{R}^c$;

where $\mathbf{MSE}$ stands for the mean square error and $\mathbf{CE}$ is a cross entropy loss. Notably, for the first part in the above formulation, $\mathbf{H}^S$ ($\mathbf{H}^T$) corresponds to the output matrix of the student's (teacher's) hidden states; l is the sequence length of the input (in our experiments, it is set to be 64 or 128); d_h is the hidden dimension (768 for BERT$_{base}$). For the second part $\mathbf{A}_i^S$ ($\mathbf{A}_i^T$) is the attention matrix corresponds to the i-th heads (in our setting, $h = 12$). In the final part, the dimension c in logit outputs ($\mathbf{p}^S$ and $\mathbf{p}^T$) is either to be 2 or 3 for GLUE tasks. For more details on how the weight matrices involved in the output matrices, please see [20].

We have defined the three types of KD: 1S-KD, 2S-KD, and 3S-KD in main text. See Figure 4 for a visualization. Here we explain in more details. **One-Stage** KD means we naively minimize the sum of teacher-student differences on hidden-states, attentions and logits. In this setup, a single

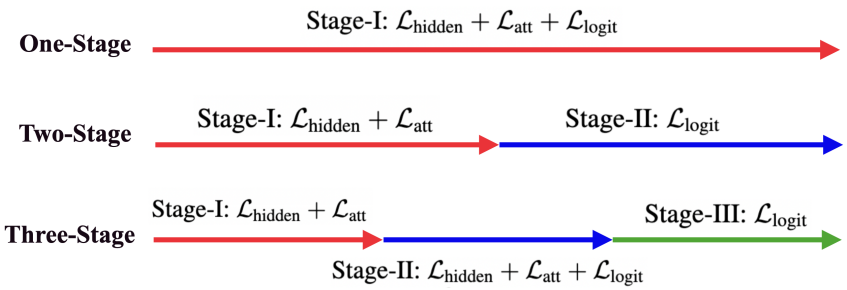

Figure 4: **Three types of knowledge distillation.** 1S-KD KD (top red arrow line) involves all the outputs of hidden-states, attentions and logits from the beginning of the training to the end. 2S-KD KD (middle red and blue arrow line) separates hidden-states and attentions from the logits part. While 3S-KD KD (bottom red, blue and green arrow line) succeed 2S-KD one, it also adds a transition phase in the middle of the training.

one-time learning rate schedule is used; in our case, it is a linear-decay schedule with warm-up steps 10% of the total training time. **Two-Stage** KD first minimizes the losses of hidden-states $\mathcal{L}_{\text{hidden}}$ and attentions $\mathcal{L}_{\text{att}}$, then followed by the loss of logits $\mathcal{L}_{\text{logit}}$. This type of KD is proposed in [20] and has been used in [3] for extreme quantization. During the training, the (linear-decay) learning rate schedule will be repeated from the first stage to the second one; particularly, in [20] they used a $\times 2.5$ peak learning rate for Stage-I than that for Stage-II. Finally, **Three-Stage** KD succeeds the properties of 1S-KD and 2S-KD. That is, after minimizing the losses of hidden-states $\mathcal{L}_{\text{hidden}}$ and attentions $\mathcal{L}_{\text{att}}$, we add a transition phase: $\mathcal{L}_{\text{hidden}} + \mathcal{L}_{\text{att}} + \mathcal{L}_{\text{logit}}$, instead of directly optimizing $\mathcal{L}_{\text{logit}}$. The learning rate schedules will be correspondingly repeated in three times and the peak learning rate of Stage-I is $\times 2.5$ higher than that of Stage-II and Stage-III.

B.2 Experimental Setup

Similar to [20, 3], we fine-tune on GLUE tasks, namely, MRPC [11], STS-B [7], SST-2 [47], QNLI [38], QQP [19], MNLI [56], CoLA [55], RTE [9]). We recorded their validation performances over the training and report the best validation value. The maximum sequence length is set to 64 for CoLA/SST-2, and 128 for the rest sequence pair tasks. See Table C.1 for the size of each (augmented) dataset and training batch size. Each independent experiment is ran on a single A100 GPU with a fixed random seed 42. To understand how much the variance can bring with various random seed, we include a set of experiments by only varying the random seeds 111 and 222. The results are given in Table C.16 and Table C.17. We overall find the standard deviation are within 0.1 on the GLUE score. Note also that we add extra 1 epoch for the warmup training.

The 2-bit algorithm and distillation code are based on the official implementation of [61] [5] and the 1-bit algorithm is based only on the binary code[6] in [3] as explained in the main text. Finally, we comment on the reproduced result shown in Row 3 of Table 5. We apply the 2S-KD where Stage-I distillation requires much longer epochs than Budget-C as ColA requires 50 epochs, MNLI/QQP/QNLI requires 10 epochs, and others 20 epochs. All are trained on augmented datasets; Note that 10-epoch augmented MNLI/QQP is much longer than 18/36-epoch standard ones. Stage-II KD is similar to Budget B but needs to search with three learning rates and two batch sizes.

B.3 Exploration of LoRa

As we see in § 5 (i.e.,Table 6), smaller models is much more brittle on extreme quantization than $\text{BERT}_{\text{base}}$, especially the 1-bit compression. We now investigate whether adding low-rank (LoRa) matrices to the quantize weights would help improve the situation. That is, we introduce two low-rank matrices $U \in \mathbb{R}^{d_{\text{in}} \times r}$ and $V \in \mathbb{R}^{r \times d_{\text{out}}}$ for a quantized weight $\widetilde{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ such that the weight we

[5] https://github.com/huawei-noah/Pretrained-Language-Model/tree/master/TernaryBERT

[6] https://github.com/huawei-noah/Pretrained-Language-Model/blob/master/BinaryBERT/transformer/utils_quant.py#L347

used for input is

$$W = \widetilde{W} + UV \tag{2}$$

Here U and V are computed in full precision. In our experiments for the layer-reduced model (Skip-Bert$_6$), we let $r \in \{1, 8\}$. We use 1S-KD and also include the scenarios for the three budgets and three learning rates {5e-5,1e-4,5e-4}. The full results for 1-bit and 2-bit Skip-Bert$_6$ are presented in Table C.11 and Table C.12. Correspondingly, we summarized the results in Table C.9 and Table C.10 for clarity.

Overall, we see that LoRa can boost performance when the training time is short. For instance, as shown in Table C.9 (Table C.10) under Budget-A, the average score for 1-bit (2-bit) is 80.81 (82.14) without LoRa, compared with 81.09 (82.31) with LoRa. However, the benefit becomes marginal when training is long. Here, under Budget-B, the average score for 1-bit (2-bit) is 81.37 (82.51) without LoRa, compared with 81.59 (82.41) with LoRa.

To further verify the claim of "marginal or none" benefit using LoRa under a sufficient training budget, we include more experiments for five-/four-layer BERT (Skip-BERT$_5$ and Skip-BERT$_4$), and the results are shown in Table C.13. The observation is similar to what we have found for Skip-BERT$_6$.

Interestingly, we find that in Table C.14, instead of adding these full-precision low-rank matrices, there is a considerable improvement by continuing training with another Budget-C iteration based on the checkpoints obtained under Budget-C. Particularly, there is 0.09 in GLUE score's improvement for 1-bit SkipBERT$_5$ (from 81.34 (Row 1) to 81.63 (Row 2)) and 0.28 for 1-bit SkipBERT$_4$. On the other hand, if we trained a 2-bit SkipBERT$_5$/SkipBERT$_4$ first and then continue training them into 1-bit, the final performance (see Row 5 and Row 11) is even worse than a direct 1-bit quantization (see Row 1 and Row 7). In addition to the results above, we also tried to see if continuing training by adding LoRa (rank=1) matrices would help or not. However, the results are negative.

B.4 Exploration of Smaller Teacher

As shown in previous works, such as [53, 25], there is a clear advantage to using larger teacher models. Here shown in Table C.15 under Budget-C, we apply 1-/2-bit quantization on Skip-BERT$_6$ and compare the teachers between BERT$_{\text{base}}$ and the full-precision Skip-BERT$_6$.

We verify that a better teacher can boost the performance, particularly for 1-bit quantization on the smaller tasks such as CoLA, MRPC, RTE, and STS-B. However, when adding LoRa, the advantage of using better teacher diminishes as the gain on average GLUE score decreases from 0.31 (without LoRa) to 0.11 (with LoRa) on 1-/2-bit quantization.

C Figures and Tables

Table C.1: Training details. DA is short for data augmentation.

	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
Dev data	1043	9815/9832	408	5463	40430	277	872	1500
Train (noDA)	8551	392702	3668	104743	363846	2490	67349	5749
Train (DA)	218844	392702	226123	4273773	363846	145865	1133834	327384
DA/noDA	25.6	1	61.6	40.8	1	58.6	16.8	56.9
Train batch-size	16	32	32	32	32	32	32	32

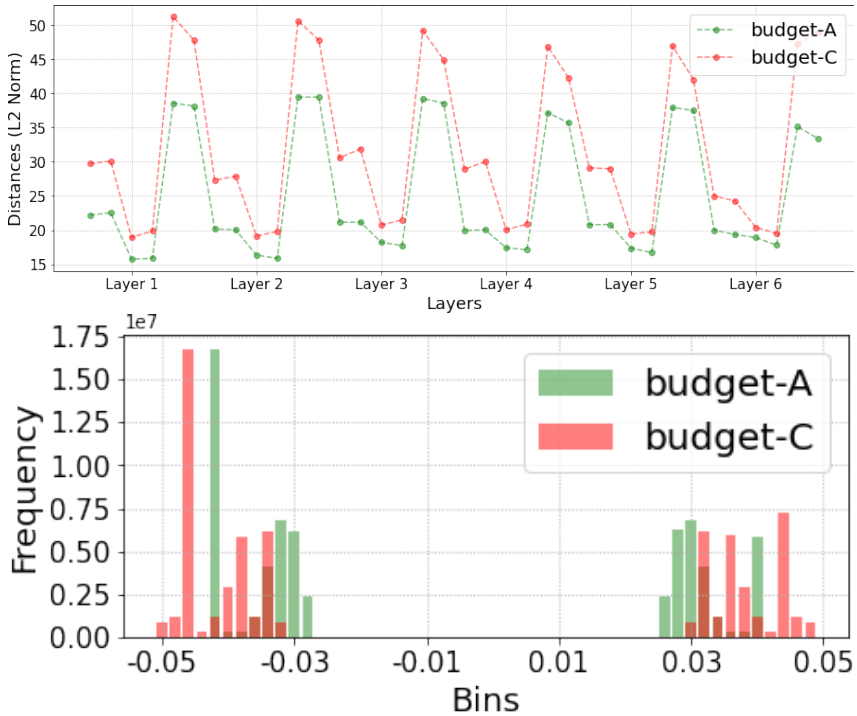

Figure 5: **Top**: The ℓ_2 norm distance between the fp32 SkipBERT$_6$ and its 1-bit counterpart under Budget-A (green) and Budget-C (red) training recipes (quantizing the fine-tuned 6-layer SkipBERT$_6$ on the RTE task with learning rate 2e-5 and 1S-KD). In each layer, we have 6 linear modules (so there are six points in each layer). **Bottom**: Histograms of the binary weight parameters with Budget-A (green) and Budget-C (red) training recipes (models are the same as the top figure). In addition to the 6×6 matrices, the elements of binary weights also include the quantized word embedding matrix.

Table C.2: 1-bit quantization for BERT$_{\text{base}}$ with size 13.0MB ($\times$32.0 smaller than the fp32 version).

Training Cost	Stages	learning rate	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
Budget-A	One-Stage	2e-5	44.6	83.1/83.7	88.8/83.1	91.1	87.4/90.7	66.1	92.8	87.8/87.5	80.33	84.80
		1e-4	50.4	83.7/84.6	90.1/85.3	91.3	88.0/91.1	71.5	92.7	88.8/88.4	82.16	86.12
		5e-4	42.3	83.3/84.1	90.0/85.8	89.5	87.8/90.9	72.9	92.5	88.1/87.8	81.04	85.89
		Best (above)	50.4	83.7/84.6	90.0/85.8	91.3	88.0/91.1	72.9	92.8	88.8/88.4	82.38	86.37
	Two-Stage	2e-5	48.2	83.3/83.8	89.3/84.6	90.7	87.7/90.9	70.4	92.5	88.7/88.4	81.46	85.61
		1e-4	48.5	83.3/83.4	90.0/85.5	90.4	87.4/90.7	70.4	92.4	88.6/88.2	81.47	85.59
		5e-4	16.2	74.9/76.4	89.7/84.8	87.8	85.0/89.0	68.2	91.6	86.2/86.5	75.01	82.36
		Best (above)	48.5	83.3/83.8	90.0/85.5	90.7	87.7/90.9	70.4	92.5	88.7/88.4	81.59	85.72
	Three-Stage	2e-5	49.3	83.1/83.6	89.5/84.3	91.0	87.6/90.9	70.8	92.4	88.7/88.3	81.57	85.60
		1e-4	49.0	83.2/83.4	89.7/84.8	90.9	87.6/90.7	73.6	92.2	88.8/88.5	81.84	85.95
		5e-4	29.5	81.3/81.9	89.1/83.6	88.4	85.0/89.3	66.1	91.9	84.1/83.9	77.34	83.32
		Best (above)	49.3	83.2/83.4	89.7/84.8	91.0	87.6/90.9	73.6	92.4	88.8/88.5	81.93	86.01
Budget-B	One-Stage	2e-5	51.8	84.1/84.4	89.6/84.6	90.8	88.0/91.1	70.8	92.8	88.8/88.4	82.13	85.92
		1e-4	55.6	83.9/84.6	90.4/86.0	90.8	88.3/91.3	72.6	93.1	88.9/88.5	82.98	86.40
		5e-4	45.7	82.9/83.4	90.4/86.0	89.2	87.7/90.8	72.6	92.8	87.8/87.5	81.24	85.69
		Best (above)	55.6	84.1/84.4	90.4/86.0	90.8	88.3/91.3	72.6	93.1	88.9/88.5	82.98	86.40
	Two-Stage	2e-5	52.4	84.0/84.5	89.7/85.0	90.5	88.0/91.1	72.2	92.7	89.2/88.8	82.40	86.15
		1e-4	51.3	83.3/83.6	90.3/85.8	90.2	87.0/90.6	72.9	92.5	88.6/88.3	82.09	85.94
		5e-4	29.7	78.4/79.6	89.2/84.3	88.8	83.2/88.2	69.7	91.3	84.8/85.1	77.20	83.14
		Best (above)	52.4	84.0/84.5	90.3/85.8	90.5	88.0/91.1	72.9	92.7	89.2/88.8	82.57	86.34
	Three-Stage	2e-5	53.0	83.8/84.4	90.1/85.3	90.7	88.0/91.1	72.2	93.2	89.2/88.8	82.54	86.24
		1e-4	51.5	83.4/83.9	90.4/86.0	90.2	87.9/91.1	72.6	92.9	88.7/88.4	82.26	86.10
		5e-4	35.6	80.2/80.9	89.1/84.6	88.5	86.1/89.7	69.3	90.6	86.4/86.2	78.42	83.78
		Best (above)	53.0	83.8/84.4	90.4/86.0	90.7	88.0/91.1	72.6	93.2	89.2/88.8	82.67	86.38
Budget-C	One-Stage	2e-5	57.3	84.1/84.0	90.7/86.5	91.0	88.2/91.2	74.0	93.1	89.2/88.8	83.38	86.64
		1e-4	55.6	84.2/84.4	90.4/86.0	90.4	88.3/91.3	73.6	93.0	88.8/88.4	83.03	86.46
		5e-4	46.1	82.8/82.9	89.9/86.0	89.5	87.7/90.8	71.1	92.4	88.2/88.0	81.09	85.46
		Best (above)	57.3	84.2/84.4	90.7/86.5	91.0	88.3/91.3	74.0	93.1	89.2/88.8	83.44	86.71
	Two-Stage	2e-5	55.9	84.2/83.9	90.6/86.5	90.7	88.3/91.3	73.3	93.1	89.2/88.9	83.12	86.52
		1e-4	52.0	83.2/83.8	90.9/86.5	90.1	88.0/91.0	74.0	93.1	87.8/87.8	82.39	86.19
		5e-4	32.2	79.3/80.3	90.6/86.5	89.6	85.3/89.1	70.0	91.2	87.0/86.8	78.36	84.12
		Best (above)	55.9	84.2/83.9	90.6/86.5	90.7	88.3/91.3	74.0	93.1	89.2/88.9	83.20	86.61
	Three-Stage	2e-5	57.1	84.0/84.4	90.6/86.5	90.9	88.1/91.1	72.6	92.9	89.5/89.2	83.22	86.49
		1e-4	53.8	83.5/84.1	90.8/86.8	90.1	88.3/91.3	72.6	92.8	88.2/87.9	82.58	86.18
		5e-4	31.3	80.0/81.3	89.8/85.3	89.2	86.7/90.0	70.8	91.3	86.4/86.8	78.40	84.29
		Best (above)	57.1	84.0/84.4	90.8/86.8	90.9	88.3/91.3	72.6	92.9	89.5/89.2	83.28	86.55

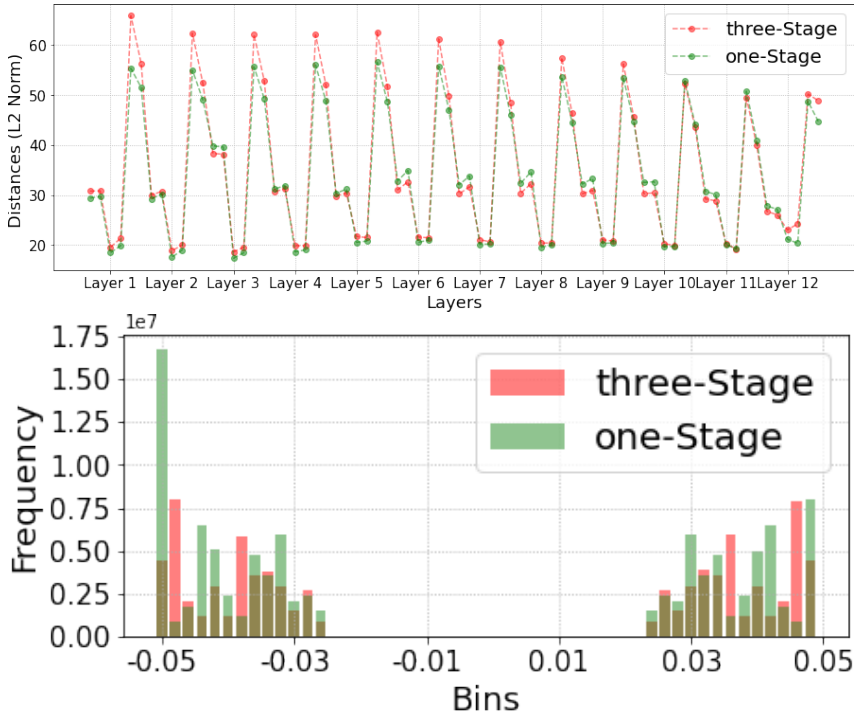

Figure 6: **Top**: The ℓ_2 norm distance between the fp32 $\text{BERT}_{\text{base}}$ and its 1-bit counterpart under 3S-KD (red) and 1S-KD (green) training recipes (quantizing the fine-tuned 12-layer $\text{BERT}_{\text{base}}$ on the RTE task with learning rate 5e-4 and Budget-B). In each layer, we have 6 linear modules (so there are six points in each layer). **Bottom**: Histograms of the binary weight parameters with 3S-KD (red) and 1S-KD (green) training recipes (models are the same as the top figure). In addition to the 12×6 matrices, the elements of binary weights also include the quantized word embedding matrix.

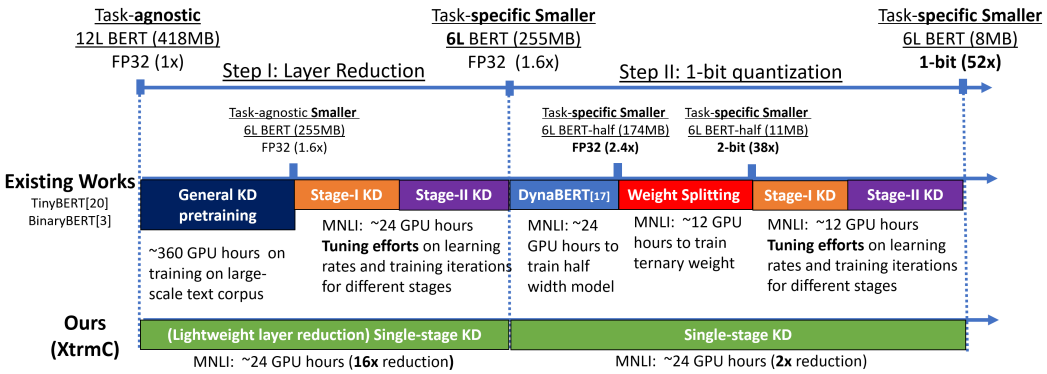

Figure 7: An example on the GPU hours comparing between our methods and the existing works.

Table C.3: 2-bit quantization for $\text{BERT}_{\text{base}}$ with size 26.8MB ($\times$16.0 smaller than the fp32 version).

Training Cost	Stages	learning rate	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
Budget-A	One-Stage	2e-5	58.0	84.2/84.6	89.8/84.8	91.5	88.4/91.4	72.2	93.5	89.4/89.0	83.29	86.45
		1e-4	55.8	84.5/85.0	90.9/87.0	91.5	88.4/91.4	74.0	93.5	89.6/89.2	83.59	87.06
		5e-4	46.1	83.8/84.7	89.6/85.0	90.9	88.0/91.2	71.8	92.8	88.8/88.4	81.68	86.12
		Best (above)	58.0	84.5/85.0	90.9/87.0	91.5	88.4/91.4	74.0	93.5	89.6/89.2	83.83	87.06
	Two-Stage	2e-5	56.6	84.3/85.0	91.0/87.3	91.3	88.3/91.3	71.5	93.5	89.6/89.1	83.38	86.72
		1e-4	52.3	84.2/84.8	90.2/85.5	91.0	87.9/91.1	72.2	92.8	89.4/89.1	82.59	86.38
		5e-4	29.0	78.1/78.8	90.3/85.8	90.3	86.2/89.9	68.2	91.7	87.6/87.5	77.71	83.80
		Best (above)	56.6	84.3/85.0	91.0/87.3	91.3	88.3/91.3	72.2	93.5	89.6/89.1	83.46	86.81
	Three-Stage	2e-5	58.8	84.3/84.7	90.4/86.8	91.4	88.0/91.2	72.6	93.5	89.6/89.1	83.66	86.76
		1e-4	55.2	84.1/84.9	90.6/86.3	91.1	87.9/91.2	74.0	93.1	89.2/88.9	83.23	86.74
		5e-4	33.3	83.1/83.8	88.5/83.6	89.8	86.4/90.0	47.3	91.2	84.4/84.0	76.28	81.65
		Best (above)	58.8	84.3/84.7	90.4/86.8	91.4	88.0/91.2	74.0	93.5	89.6/89.1	83.81	86.94
Budget-B	One-Stage	2e-5	58.2	84.6/84.7	91.3/87.7	91.4	88.4/91.4	73.6	93.3	89.6/89.2	83.83	87.04
		1e-4	56.3	84.3/84.9	90.5/86.3	91.3	88.4/91.4	75.8	93.5	89.5/89.1	83.70	87.12
		5e-4	47.3	84.0/84.3	90.1/86.3	90.5	88.3/91.2	72.6	92.9	88.7/88.2	81.98	86.31
		Best (above)	58.2	84.6/84.7	91.3/87.7	91.4	88.4/91.4	75.8	93.5	89.6/89.2	84.10	87.34
	Two-Stage	2e-5	58.4	84.3/84.9	90.5/86.3	91.4	88.4/91.3	74.0	93.5	89.7/89.4	83.76	86.93
		1e-4	53.6	84.0/84.6	91.1/87.3	91.0	88.0/91.2	73.3	93.5	89.0/88.6	83.06	86.74
		5e-4	34.7	81.5/82.3	90.1/85.5	90.5	85.5/89.2	71.1	91.6	87.6/87.4	79.33	84.91
		Best (above)	58.4	84.3/84.9	91.1/87.3	91.4	88.4/91.3	74.0	93.5	89.7/89.4	83.87	87.05
	Three-Stage	2e-5	59.6	84.5/84.7	90.9/86.5	91.3	88.3/91.3	73.3	93.3	89.7/89.3	83.80	86.82
		1e-4	55.9	84.6/84.8	90.7/86.5	91.1	88.3/91.4	74.4	93.3	89.3/89.0	83.48	86.92
		5e-4	35.3	81.4/81.8	89.9/85.5	90.6	86.9/90.2	72.9	91.6	87.8/87.6	79.68	85.22
		Best (above)	59.6	84.6/84.8	90.9/86.5	91.3	88.3/91.4	74.4	93.3	89.7/89.3	83.96	87.00
Budget-C	One-Stage	2e-5	59.9	84.7/84.4	91.0/86.7	91.4	88.4/91.4	74.0	93.5	89.6/89.2	84.00	87.01
	Two-Stage	2e-5	59.9	84.9/84.7	91.2/87.3	91.2	88.2/91.3	73.3	93.5	89.7/89.4	83.98	86.99
	Three-Stage	2e-5	59.6	84.7/84.3	91.0/87.0	91.2	88.4/91.3	74.4	93.7	89.6/89.2	83.98	87.03

Table C.4: Results without data augmentation. Details of Table 4 for 1-bit quantization on $\text{BERT}_{\text{base}}$ with three learning rates.

Cost	learning rate	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	Avg. w/o CoLA
Budget-A One-Stage	2e-5	47.3	83.1/83.7	87.2/80.1	90.8	87.4/90.7	66.8	92.2	85.5/85.3	80.02	84.11
	1e-4	48.9	83.7/84.6	89.0/83.6	91.0	88.0/91.1	71.5	92.5	87.6/87.5	81.61	85.70
	5e-4	38.1	83.3/84.1	88.4/83.6	90.1	87.8/90.9	69.7	92.0	85.0/85.0	79.64	84.84
	Best (above)	48.9	83.7/84.6	89.0/83.6	91.0	88.0/91.1	71.5	92.5	87.6/87.5	81.61	85.70
Budget-C One-Stage	2e-5	55.0	84.1/84.0	90.0/85.0	90.7	88.2/91.2	73.3	92.9	88.2/87.9	82.71	86.18
	1e-4	52.9	84.2/84.4	89.2/85.0	90.4	88.3/91.3	72.9	92.7	87.7/87.7	82.39	86.08
	5e-4	42.2	82.8/82.9	88.7/83.6	89.8	87.7/90.8	71.1	91.4	87.0/86.8	80.18	84.92
	Best (above)	55.0	84.2/84.4	90.0/85.0	90.7	88.3/91.3	73.3	92.9	88.2/87.9	82.78	86.25

Table C.5: Details of Table 5: Layer reduction with different learning rates (i.e., the 2nd column).

Model (MB)	Learning rate	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
BERT-base	n/a	59.7	84.9/85.6	90.6/86.3	92.1	88.6/91.5	72.2	93.2	90.1/89.6	83.95	86.98
TinyBERT_6	5e-5	53.6	84.3/84.3	90.4/86.3	91.2	88.5/91.5	67.9	93.2	89.2/88.8	82.39	85.99
	1e-4	54.4	84.6/84.3	90.1/85.8	91.5	88.4/91.5	68.6	93.3	89.1/89.0	82.57	86.09
	5e-4	45.1	84.1/84.2	90.1/85.8	91.3	88.5/91.5	69.7	93.2	89.1/88.7	81.56	86.11
MiniLM_6	5e-5	55.1	84.4/84.4	90.4/86.0	91.3	88.3/91.4	70.8	93.0	89.4/89.0	82.87	86.34
	1e-4	55.4	84.2/84.5	89.5/85.3	91.3	88.4/91.5	70.8	93.2	89.4/89.0	82.84	86.28
	5e-4	49.5	84.1/83.9	90.7/86.5	91.4	88.5/91.5	71.8	93.3	89.1/88.7	82.34	86.45
Skip-BERT_6	5e-5	56.9	84.5/84.9	89.7/84.8	91.7	88.5/91.5	71.8	93.5	89.8/89.4	83.27	86.56
	1e-4	56.6	84.6/84.7	90.4/85.8	91.8	88.5/91.6	72.6	93.5	89.7/89.4	83.43	86.79
	5e-4	48.5	84.3/84.2	90.0/85.5	91.5	88.6/91.6	71.5	93.5	89.4/88.9	82.22	86.44
Skip-BERT_5	5e-5	57.9	84.3/85.1	89.8/84.8	91.4	88.3/91.4	72.2	93.3	89.2/88.9	83.29	86.46
	1e-4	56.7	84.2/84.8	90.1/85.5	91.4	88.3/91.4	72.2	93.2	89.2/88.8	83.18	86.49
	5e-4	53.6	84.1/84.4	90.1/85.5	91.4	88.5/91.5	70.4	93.3	88.6/88.2	82.53	86.15
Skip-BERT_4	5e-5	53.3	83.0/83.3	89.9/85.0	90.8	88.1/91.3	70.0	93.2	88.8/88.4	82.08	85.67
	1e-4	52.2	83.2/83.4	90.0/85.3	90.7	88.2/91.3	70.0	93.5	88.6/88.3	82.02	85.75
	5e-4	46.8	82.9/83.0	89.6/85.0	90.4	87.8/91.0	67.9	93.1	88.3/87.8	80.93	85.20

Table C.6: Selection for student networks from their teacher networks (fine-tuned $\text{BERT}_{\text{base}}$). The following results follow the same hyperparamters and the learning rate 5e-5.

Layer Selection	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	Avg. w/o CoLA
BERT-base fp32	59.7	84.9/85.6	90.6/86.3	92.1	88.6/91.5	72.2	93.2	90.1/89.6	83.95	86.98
(1) Skip-6	56.9	84.5/84.9	89.7/84.8	91.7	88.5/91.5	71.8	93.5	89.8/89.4	83.27	86.56
(2) Top-6	52.0	83.9/84.2	90.0/85.3	91.5	88.2/91.3	69.3	93.0	88.2/87.9	82.08	85.84
(3) Bottom-6	51.2	83.2/83.6	89.5/84.6	90.1	87.8/90.9	67.9	92.7	89.5/89.1	81.50	85.29

Table C.7: Compare between different stages KD. All learning rate set to be 5e-5.

#	Model (255.2MB)	#-Stage Same Budget	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	Avg. w/o CoLA
1	TinyBERT_6	One	53.6	84.3/84.3	90.4/86.3	91.2	88.5/91.5	67.9	93.2	89.2/88.8	82.39	85.99
2		Two	55.3	84.2/84.2	89.9/85.3	91.1	88.4/91.4	68.6	92.9	89.2/88.9	82.47	85.86
3		Three	54.2	84.2/84.4	89.9/85.0	91.3	88.5/91.5	70.8	93.1	89.2/88.8	82.63	86.19
4	MiniLM_6-v2	One	55.1	84.4/84.4	90.4/86.0	91.3	88.3/91.4	70.8	93.0	89.4/89.0	82.87	86.34
5		Two	53.8	84.4/84.5	90.7/87.3	91.2	88.3/91.3	67.9	93.1	89.7/89.3	82.58	86.18
6		Three	54.0	84.4/84.6	89.1/85.3	91.3	88.5/91.5	70.4	93.0	89.6/89.2	82.68	86.26
7	Skip-BERT_6	One	56.9	84.5/84.9	89.7/84.8	91.7	88.5/91.5	71.8	93.5	89.8/89.4	83.27	86.56
8		Two	56.0	84.1/84.2	89.7/84.8	91.5	88.2/91.3	71.8	93.3	89.4/89.1	82.93	86.30
9		Three	55.3	84.4/84.6	89.7/84.8	91.7	88.5/91.5	72.2	93.6	89.6/89.2	83.08	86.55

Table C.8: 1-/2-bit quantization of smaller model.

#	Model #-layer	bit	size (MB)	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	Acc. drop
1	TinyBERT_6[61]	32	255.2 (×1.6)	54.1	84.5/84.5	91.0/87.3	91.1	88.0/91.1	71.8	93.0	89.8/89.6	83.02	–
2		2	16.0 (×26.2)	53.0	83.4/83.8	91.5/88.0	89.9	87.2/90.5	71.8	93.0	86.9/86.5	82.26	-0.76
3	XtrmC-BERT_6 SIX	32	255.2 (×1.6)	56.9	84.4/84.8	90.1/85.5	91.3	88.4/91.4	72.2	93.2	90.3/90.0	83.33	–
4		2	16.0 (×26.2)	53.8	83.6/84.2	90.5/86.3	90.6	88.2/91.3	73.6	93.6	89.0/88.7	82.89	-0.44
5		1	8.0 (×52.3)	52.3	83.4/83.8	90.0/85.3	89.4	87.9/91.1	68.6	93.1	88.4/88.0	81.71	-1.51
6	XtrmC-BERT_5 FIVE	32	228.2 (×1.8)	56.7	84.2/84.8	90.1/85.5	91.4	88.3/91.4	72.2	93.2	89.2/88.8	83.18	–
7		2	14.2 (×29.3)	53.9	83.3/84.1	90.4/86.0	90.4	88.2/91.2	71.8	93.0	88.4/88.0	82.46	-0.72
8		1	7.1 (×58.5)	52.2	82.9/83.2	89.9/85.0	88.5	87.6/90.8	69.3	92.9	87.3/87.0	81.34	-1.84
9	XtrmC-BERT_4 FOUR	32	201.2 (×2.1)	52.2	83.2/83.4	90.0/85.3	90.7	88.2/91.3	70.0	93.5	88.6/88.3	82.02	–
10		2	12.6 (×33.2)	50.3	82.5/83.0	90.0/85.3	89.2	87.8/91.0	69.0	92.8	87.9/87.4	81.22	-0.90
11		1	6.3 (×66.4)	48.3	82.0/82.3	89.9/85.5	87.7	86.9/90.4	63.9	92.4	87.1/86.7	79.96	-2.01

Table C.9: **1-bit** quantization for Skip-BERT_6.

#	Cost	LoRa (Y/N)	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
1	Budget-A	N	47.9	82.9/83.2	90.5/86.3	89.7	87.8/90.9	65.7	92.7	88.0/87.6	80.81	84.92
2		Y (DIM=1)	48.1	82.9/83.2	89.4/84.3	89.6	87.6/90.8	65.3	92.4	88.1/87.7	80.52	84.58
3		Y (DIM=8)	49.7	83.4/83.6	89.8/85.0	89.9	87.5/90.8	66.8	92.8	87.8/87.5	81.09	85.01
4	Budget-B	N	50.3	83.2/83.7	90.9/86.8	89.3	87.9/91.0	67.1	92.7	88.2/88.0	81.37	85.25
5		Y (DIM=1)	49.9	83.3/83.8	90.5/86.0	89.3	88.0/91.1	67.5	92.7	88.3/87.9	81.32	85.25
6		Y (DIM=8)	50.9	83.2/83.7	90.3/85.8	89.4	88.1/91.2	69.0	92.8	88.3/88.0	81.59	85.42
7	Budget-C	N	51.9	83.1/83.6	90.5/86.0	89.3	88.2/91.2	69.0	93.1	88.1/87.7	81.70	85.42
8		Y (DIM=1)	52.0	83.2/83.8	89.8/86.0	89.4	88.0/91.1	69.3	93.0	88.2/88.0	81.78	85.50
9		Y (DIM=8)	52.6	83.3/83.8	89.9/85.5	89.2	88.2/91.2	69.3	92.8	88.2/87.9	81.77	85.41

Table C.10: **2-bit** quantization for Skip-BERT_6.

#	Cost	LoRa (Y/N)	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
1	Budget-A	N	51.8	83.8/83.7	90.2/85.8	90.6	88.2/91.3	70.4	93.2	88.7/88.3	82.14	85.94
2		Y (DIM=1)	51.4	83.7/84.2	90.3/85.8	90.5	88.2/91.3	70.8	93.5	88.9/88.5	82.23	86.09
3		Y (DIM=8)	51.6	83.6/84.2	90.6/86.3	90.7	88.2/91.3	71.1	93.1	88.9/88.5	82.31	86.15
4	Budget-B	N	52.7	83.8/84.5	90.0/85.3	90.3	88.4/91.4	72.6	93.0	89.0/88.6	82.51	86.24
5		Y (DIM=1)	53.6	83.6/83.9	90.5/86.0	90.2	88.4/91.4	71.1	93.2	88.9/88.5	82.43	86.04
6		Y (DIM=8)	53.3	83.9/84.4	90.3/85.8	90.2	88.3/91.3	70.8	93.0	89.0/88.7	82.41	86.05

Table C.11: **1-bit** quantization for Skip-BERT$_6$.

Training Cost	LoRa	learning rate	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
Budget-A	N	5e-5	45.6	82.6/83.0	89.3/84.3	89.7	87.5/90.8	63.5	92.4	87.6/87.3	79.94	84.24
		1e-4	47.9	82.9/83.2	89.4/84.6	89.4	87.8/90.9	64.3	92.4	88.0/87.6	80.40	84.46
		5e-4	40.6	81.9/82.6	90.5/86.3	88.4	87.6/90.8	65.7	92.7	86.9/86.7	79.54	84.41
		Best (above)	47.9	82.9/83.2	90.5/86.3	89.7	87.8/90.9	65.7	92.7	88.0/87.6	80.81	84.92
	Y (DIM=1)	5e-5	46.9	82.9/83.2	88.9/83.6	89.6	87.6/90.8	63.2	92.4	87.7/87.4	80.03	84.18
		1e-4	48.1	82.6/83.5	89.4/84.3	89.2	87.6/90.8	65.3	92.3	88.1/87.7	80.47	84.51
		5e-4	39.3	82.1/82.8	89.2/84.3	88.6	87.5/90.7	65.0	92.2	87.0/86.8	79.11	84.09
		Best (above)	48.1	82.9/83.2	89.4/84.3	89.6	87.6/90.8	65.3	92.4	88.1/87.7	80.52	84.58
	Y (DIM=8)	5e-5	49.7	82.8/83.5	88.7/83.1	89.9	87.5/90.8	66.8	92.7	87.7/87.4	80.78	84.66
		1e-4	49.6	83.4/83.6	89.8/85.0	89.4	87.6/90.8	65.7	92.8	87.8/87.5	80.90	84.81
		5e-4	38.4	82.3/83.0	89.4/84.1	88.9	87.6/90.8	66.4	92.0	86.8/86.6	79.19	84.29
		Best (above)	49.7	83.4/83.6	89.8/85.0	89.9	87.5/90.8	66.8	92.8	87.8/87.5	81.09	85.01
Budget-B	N	5e-5	50.3	83.1/83.7	89.2/84.1	89.3	87.9/91.0	66.8	92.7	88.2/88.0	81.02	84.86
		1e-4	49.7	83.2/83.7	90.9/86.8	88.8	87.9/91.0	67.1	92.7	88.1/87.8	81.23	85.18
		5e-4	45.0	81.9/81.9	89.1/84.3	88.1	87.4/90.7	66.1	92.5	85.9/85.7	79.60	83.92
		Best (above)	50.3	83.2/83.7	90.9/86.8	89.3	87.9/91.0	67.1	92.7	88.2/88.0	81.37	85.25
	Y (DIM=1)	5e-5	49.4	83.0/83.6	89.7/85.0	89.3	88.1/91.0	67.5	92.7	88.3/87.9	81.09	85.05
		1e-4	49.9	83.3/83.8	90.5/86.0	88.8	88.0/91.1	67.1	92.4	88.2/87.9	81.18	85.09
		5e-4	41.8	81.8/82.5	89.7/84.8	88.0	87.7/90.8	63.9	92.1	86.0/85.9	79.08	83.74
		Best (above)	49.9	83.3/83.8	90.5/86.0	89.3	88.0/91.1	67.5	92.7	88.3/87.9	81.32	85.25
	Y (DIM=8)	5e-5	50.9	83.2/83.7	90.3/85.8	89.4	87.9/91.0	69.0	92.8	88.3/88.0	81.57	85.40
		1e-4	50.6	83.2/83.4	89.7/84.8	89.1	88.1/91.2	67.5	92.8	88.1/87.9	81.19	85.01
		5e-4	41.7	82.2/82.8	89.2/84.1	88.6	87.7/90.8	65.3	92.1	86.0/85.7	79.29	83.99
		Best (above)	50.9	83.2/83.7	90.3/85.8	89.4	88.1/91.2	69.0	92.8	88.3/88.0	81.59	85.42
Budget-C	N	5e-5	51.9	82.9/83.6	90.2/85.8	89.3	88.2/91.2	69.0	93.1	88.1/87.7	81.66	85.38
		1e-4	50.3	83.1/83.6	90.5/86.0	88.9	87.9/91.0	67.9	92.7	87.6/87.4	81.23	85.10
		5e-4	44.4	81.7/81.9	89.6/84.6	88.4	87.5/90.7	64.6	92.9	87.1/87.0	79.59	83.99
		Best (above)	51.9	83.1/83.6	90.5/86.0	89.3	88.2/91.2	69.0	93.1	88.1/87.7	81.70	85.42
	Y (DIM=1)	5e-5	52.0	83.2/83.8	89.8/86.0	89.4	87.9/91.0	69.3	93.0	88.2/88.0	81.77	85.49
		1e-4	48.9	83.2/83.5	90.4/85.8	89.1	88.0/91.1	68.6	92.5	87.8/87.5	81.17	85.20
		5e-4	42.5	81.8/82.3	89.5/84.6	88.4	87.6/90.7	65.7	92.5	87.1/86.8	79.51	84.14
		Best (above)	52.0	83.2/83.8	89.8/86.0	89.4	88.0/91.1	69.3	93.0	88.2/88.0	81.78	85.50
	Y (DIM=8)	5e-5	52.6	83.3/83.8	89.9/85.5	89.2	88.1/91.1	69.3	92.8	88.2/87.9	81.76	85.40
		1e-4	50.2	83.0/83.8	89.9/85.5	89.0	88.2/91.2	69.0	92.7	87.9/87.6	81.37	85.26
		5e-4	43.0	82.1/82.5	90.2/85.5	88.7	87.5/90.7	67.5	92.4	87.2/86.9	79.96	84.58
		Best (above)	52.6	83.3/83.8	89.9/85.5	89.2	88.2/91.2	69.3	92.8	88.2/87.9	81.77	85.41

Table C.12: **2-bit** quantization for Skip-BERT$_6$ with various learning rates.

Training Cost	LoRa	learning rate	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
Budget-A	N	5e-5	50.8	83.8/83.7	90.0/85.3	90.4	88.2/91.3	70.4	93.2	88.6/88.2	81.94	85.84
		1e-4	51.8	83.6/84.2	90.2/85.8	90.6	88.2/91.3	70.4	92.9	88.7/88.3	82.14	85.94
		5e-4	41.4	83.2/83.5	90.1/85.5	89.8	87.9/91.0	68.2	92.5	87.8/87.4	80.32	85.19
		Best (above)	51.8	83.8/83.7	90.2/85.8	90.6	88.2/91.3	70.4	93.2	88.7/88.3	82.14	85.94
	Y (DIM=1)	5e-5	51.4	83.7/84.2	90.3/85.8	90.5	88.2/91.3	70.8	93.5	88.8/88.4	82.22	86.07
		1e-4	51.2	83.7/84.0	90.3/85.8	90.4	88.1/91.2	70.8	92.9	88.9/88.5	82.10	85.96
		5e-4	43.8	83.6/83.5	89.6/84.8	89.5	87.8/91.1	69.3	92.7	87.8/87.5	80.68	85.29
		Best (above)	51.4	83.7/84.2	90.3/85.8	90.5	88.2/91.3	70.8	93.5	88.9/88.5	82.23	86.09
	Y (DIM=8)	5e-5	51.6	83.6/84.2	90.1/85.3	90.7	88.2/91.3	71.1	93.0	88.9/88.5	82.19	86.01
		1e-4	51.0	83.6/84.1	90.6/86.3	90.3	88.3/91.3	71.1	93.1	88.8/88.5	82.18	86.08
		5e-4	43.8	83.3/83.7	89.3/84.1	89.8	88.1/91.2	69.0	92.2	88.0/87.7	80.57	85.16
		Best (above)	51.6	83.6/84.2	90.6/86.3	90.7	88.2/91.3	71.1	93.1	88.9/88.5	82.31	86.15
Budget-B	N	5e-5	52.5	83.5/84.3	89.7/84.8	90.3	88.3/91.3	71.5	93.0	89.0/88.6	82.24	85.96
		1e-4	52.7	83.8/84.5	90.0/85.3	90.1	88.4/91.4	72.6	93.0	88.9/88.6	82.48	86.20
		5e-4	45.2	83.3/83.4	89.7/85.0	89.9	88.0/91.1	69.0	92.7	88.3/87.9	80.88	85.34
		Best (above)	52.7	83.8/84.5	90.0/85.3	90.3	88.4/91.4	72.6	93.0	89.0/88.6	82.51	86.24
	Y (DIM=1)	5e-5	52.8	83.6/83.9	90.0/85.3	90.2	88.4/91.3	71.1	93.1	88.9/88.5	82.24	85.92
		1e-4	53.6	83.6/83.8	90.5/86.0	89.9	88.4/91.4	71.1	93.2	88.8/88.4	82.38	85.98
		5e-4	46.1	83.1/83.9	89.7/85.0	89.7	88.0/91.1	68.2	92.9	88.1/87.9	80.90	85.25
		Best (above)	53.6	83.6/83.9	90.5/86.0	90.2	88.4/91.4	71.1	93.2	88.9/88.5	82.43	86.04
	Y (DIM=8)	5e-5	53.3	83.9/84.4	89.8/85.0	90.2	88.3/91.3	70.8	93.0	88.9/88.5	82.31	85.94
		1e-4	53.1	83.9/84.0	90.3/85.8	90.1	88.4/91.3	70.8	92.9	89.0/88.7	82.32	85.98
		5e-4	48.7	83.2/83.5	89.7/85.5	90.0	88.3/91.3	70.4	92.8	88.0/87.7	81.49	85.59
		Best (above)	53.3	83.9/84.4	90.3/85.8	90.2	88.3/91.3	70.8	93.0	89.0/88.7	82.41	86.05

Table C.13: LoRa and quantization of five-/four-layer model.The learning rate is $2e-5$

#-layer	#-bit	loRa Y/N	size (MB)	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg. all	Avg. w/o CoLA
Five	32		228.2 (×1.8)	56.7	84.2/84.8	90.1/85.5	91.4	88.3/91.4	72.2	93.2	89.2/88.8	83.18	86.49
	1	N	7.1 (×58.5)	52.2	82.9/83.2	89.9/85.0	88.5	87.6/90.8	69.3	92.9	87.3/87.0	81.34	84.99
		Y	7.3 (×57.4)	51.8	81.8/81.9	90.2/85.5	88.6	87.1/90.3	68.6	92.8	87.5/87.2	80.98	84.62
	2	N	14.2 (×29.3)	53.9	83.3/84.1	90.4/86.0	90.4	88.2/91.2	71.8	93.0	88.4/88.0	82.46	86.02
		Y	14.6 (×28.7)	54.2	83.5/83.9	90.4/86.0	90.0	88.0/91.1	72.9	93.1	88.5/88.1	82.58	86.12
Four	32		201.2 (×2.1)	52.2	83.2/83.4	90.0/85.3	90.7	88.2/91.3	70.0	93.5	88.6/88.3	82.02	85.75
	1	N	6.3 (×66.4)	48.3	82.0/82.3	89.9/85.5	87.7	86.9/90.4	63.9	92.4	87.1/86.7	79.96	83.91
		Y	6.4 (×65.2)	48.5	81.5/80.8	89.8/85.3	87.6	86.9/90.3	64.3	92.8	87.0/86.7	79.79	83.70
	2	N	12.6 (×33.2)	50.3	82.5/83.0	90.0/85.3	89.2	87.8/91.0	69.0	92.8	87.9/87.4	81.22	85.09
		Y	12.8 (×32.6)	49.8	82.8/82.6	90.1/85.5	89.1	87.8/91.0	68.6	92.9	87.9/87.5	81.13	85.05

Table C.14: More quantization of five-/four-layer model.

#	Model	size	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
1	1-bit SkipBERT$_5$	7.1 ($\times$58.5)	52.2	82.9/83.2	89.9/85.0	88.5	87.6/90.8	69.3	92.9	87.3/87.0	81.34	84.99
Continue with above using another one-stage												
2	1-bit	7.1 ($\times$58.5)	52.8	83.0/83.4	89.8/84.8	89.6	87.8/91.0	69.3	93.2	87.6/87.3	81.63	85.24
3	1-bit+LoRa1	7.3 ($\times$57.4)	53.0	83.0/83.5	89.5/84.6	89.6	87.9/91.0	69.3	93.2	87.6/87.3	81.64	85.23
4	2-bit SkipBERT$_5$	14.2 ($\times$29.3)	53.9	83.3/84.1	90.4/86.0	90.4	88.2/91.2	71.8	93.0	88.4/88.0	82.46	86.02
Continue with above using another one-stage												
5	1-bit	7.1 ($\times$58.5)	49.7	82.4/83.0	89.9/85.3	88.2	87.8/90.9	66.8	92.7	86.4/86.1	80.60	84.46
6	1-bit+LoRa1	7.3 ($\times$57.4)	48.6	82.6/83.1	89.9/85.0	88.2	87.7/90.8	66.4	92.9	86.5/86.2	80.46	84.44
7	1-bit SkipBERT$_4$	6.3 ($\times$66.4)	48.3	82.0/82.3	89.9/85.5	87.7	86.9/90.4	63.9	92.4	87.1/86.7	79.96	83.91
Continue with above using another one-stage												
8	1-bit	6.3 ($\times$66.4)	49.8	82.4/82.3	89.5/84.8	88.4	87.6/90.8	63.5	92.9	87.2/86.9	80.23	84.04
9	1-bit+LoRa1	6.4 ($\times$65.2)	49.6	82.4/82.1	89.6/85.0	88.5	87.5/90.8	63.5	93.1	87.2/86.9	80.24	84.08
10	2-bit SkipBERT$_4$	12.6 ($\times$33.2)	50.3	82.5/83.0	90.0/85.3	89.2	87.8/91.0	69.0	92.8	87.9/87.4	81.22	85.09
Continue with above using another one-stage												
11	1-bit	6.3 ($\times$66.4)	46.2	81.8/82.0	89.1/84.6	87.2	87.3/90.6	63.2	92.5	85.7/85.4	79.31	83.45
12	1-bit+LoRa1	6.4 ($\times$65.2)	47.9	81.4/82.0	89.4/85.0	87.1	87.4/90.6	64.3	92.9	85.7/85.4	79.66	83.63

Table C.15: Compare between teachers: a twelve-layer BERT$_{\text{base}}$ vs a six-layer SkipBERT$_6$.

#-bit	LoRa (Size) Y/N (MB)	Teacher	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.	Avg. w/o CoLA
1-bit	N (8.0MB)	BERT$_{\text{base}}$	52.3	83.4/83.8	90.0/85.3	89.4	87.9/91.1	68.6	93.1	88.4/88.0	81.71	85.39
		Skip-BERT$_6$	52.0	83.2/83.5	89.7/85.0	89.3	87.7/91.0	67.5	93.2	87.9/87.6	81.40	85.08
		Difference	+0.3	+0.2/+0.3	+0.3/+0.3	+0.1	+0.2/+0.1	+1.1	-0.1	+0.5/+0.4	+0.31	+0.31
	Y (8.1MB)	BERT$_{\text{base}}$	52.1	83.3/84.0	89.9/85.5	89.5	88.0/91.1	68.2	93.0	88.5/88.1	81.69	85.39
		Skip-BERT$_6$	52.4	83.2/83.5	90.4/85.8	89.1	87.8/91.0	68.2	93.1	87.9/87.6	81.58	85.22
		Difference	-0.3	+0.1/+0.5	-0.5/-0.3	+0.1	+0.2/+0.1	0.0	-0.1	+0.6/+0.5	+0.11	+0.17
2-bit	N (16.0MB)	BERT$_{\text{base}}$	53.8	83.6/84.2	90.5/86.3	90.6	88.2/91.3	73.6	93.6	89.0/88.7	82.89	86.53
		Skip-BERT$_6$	54.5	83.7/83.8	90.5/86.0	90.3	88.1/91.2	71.5	93.3	88.8/88.4	82.57	86.07
		Difference	-0.7	-0.1/+0.4	+0.0/+0.3	+0.3	+0.1/+0.1	+2.1	+0.3	+0.2/+0.3	+0.32	+0.46
	Y (16.2MB)	BERT$_{\text{base}}$	55.2	83.7/84.4	90.3/86.0	90.5	88.2/91.3	71.8	93.2	89.0/88.6	82.79	86.24
		Skip-BERT$_6$	54.3	83.7/83.8	90.4/86.0	90.4	88.0/91.2	72.6	93.2	88.9/88.5	82.68	86.22
		Difference	-0.9	0.0/+0.6	-0.1/+0.0	+0.1	+0.2/+0.1	-0.8	+0.0	+0.1/+0.1	+0.11	+0.02

Table C.16: **1-bit** quantization with various random seeds (learning rate here all set to be 2e-5).

Model Cost	KD #-Stage	Random seed	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.
TinyBERT$_6$ Budget-B	fp32		54.2	84.25/84.38	89.85/85.05	91.3	88.49/91.45	70.8	93.1	89.18/88.79	82.64
	One	42	50.5	82.72/82.95	89.08/84.07	89.3	87.72/90.82	65.0	92.7	87.93/87.64	80.66
		111	49.6	82.75/83.15	89.04/84.07	89.4	87.8/90.97	65.0	92.4	87.95/87.75	80.59
		222	48.4	82.72/83.08	89.11/84.07	89.4	87.61/90.84	64.6	92.7	88.14/87.75	80.43
		Ave. above	49.5	82.73/83.06	89.08/84.07	89.4	87.71/90.88	64.9	92.6	88.01/87.71	80.56
		std above	0.9	0.01/0.08	0.03/0	0.0	0.08/0.07	0.2	0.1	0.09/0.05	0.09
	Two	42	48.8	82.58/82.81	88.74/83.58	89.2	87.75/90.87	65.0	92.4	88.2/87.82	80.37
		111	48.9	82.84/83.31	88.93/83.82	89.0	87.72/90.87	64.3	92.7	88.4/88.06	80.45
		222	48.6	82.72/83.27	89.56/84.8	88.9	87.68/90.89	65.7	92.6	88.2/87.87	80.62
		Ave. above	48.8	82.71/83.13	89.08/84.07	89.0	87.72/90.88	65.0	92.6	88.27/87.92	80.48
		std above	0.1	0.11/0.23	0.35/0.53	0.1	0.03/0.01	0.6	0.1	0.09/0.1	0.10
MiniLM$_6$ Budget-B	fp32		54.0	84.42/84.58	89.13/85.29	91.3	88.53/91.46	70.4	93.0	89.56/89.19	82.67
	Three	42	48.2	83.03/83.24	88.47/83.33	89.2	87.85/90.87	64.3	92.6	88.37/88.06	80.34
		111	47.7	83.2/83.42	88.56/82.84	89.4	87.76/90.9	65.3	92.6	88.48/88.17	80.43
		222	47.2	83.36/83.44	89.04/83.82	89.4	87.88/90.97	64.6	92.6	88.57/88.23	80.43
		Ave. above	47.7	83.2/83.37	88.69/83.33	89.3	87.83/90.91	64.7	92.6	88.47/88.15	80.40
		std above	0.4	0.13/0.09	0.25/0.4	0.1	0.05/0.04	0.5	0.0	0.08/0.07	0.04
SkipBERT$_6$ Budget-B	fp32		56.1	84.37/84.53	90.52/86.03	91.6	88.34/91.42	73.7	93.5	89.3/88.92	83.39
	Three	42	51.1	82.79/83.59	89.95/85.05	89.4	88.01/91.07	67.5	92.9	88.09/87.73	81.28
		111	50.7	82.96/83.2	90.02/85.54	89.4	87.96/91.04	67.9	92.9	88.35/87.99	81.32
		222	51.8	83.02/83.58	90.54/86.27	89.4	87.91/91	67.2	92.6	88.23/87.98	81.43
		Ave. above	51.2	82.92/83.46	90.17/85.62	89.4	87.96/91.04	67.5	92.8	88.22/87.9	81.35
		std above	0.4	0.1/0.18	0.26/0.5	0.0	0.04/0.03	0.3	0.2	0.11/0.12	0.06

Table C.17: **2-bit** quantization with various random seeds (learning rate here all set to be 2e-5).

Model Cost	KD #-Stage	Random seed	CoLA Mcc	MNLI-m/-mm Acc/Acc	MRPC F1/Acc	QNLI Acc	QQP F1/Acc	RTE Acc	SST-2 Acc	STS-B Pear/Spea	Avg.
SkipBERT$_6$ Budget-B	Three	42	53.2	83.67/83.79	91.25/87.25	90.4	88.42/91.4	72.2	93.6	88.77/88.36	82.70
		111	53.3	83.63/83.59	90.91/86.76	90.4	88.32/91.38	72.2	93.5	88.65/88.32	82.60
		222	54.1	83.65/84.27	90.82/86.52	90.6	88.29/91.4	71.5	93.6	88.69/88.38	82.70
		Ave. above	53.6	83.65/83.88	90.99/86.84	90.4	88.34/91.39	72.0	93.5	88.7/88.35	82.66
		std above	0.4	0.02/0.29	0.19/0.3	0.1	0.06/0.01	0.3	0.1	0.05/0.02	0.05