TernaryBERT: Distillation-aware Ultra-low Bit BERT

Wei Zhang[*], **Lu Hou**[*], **Yichun Yin**[*], **Lifeng Shang, Xiao Chen, Xin Jiang, Qun Liu**
Huawei Noah's Ark Lab
{zhangwei379, houlu3, yinyichun, shang.lifeng, chen.xiao2, jiang.xin, qun.liu}@huawei.com

Abstract

Transformer-based pre-training models like BERT have achieved remarkable performance in many natural language processing tasks. However, these models are both computation and memory expensive, hindering their deployment to resource-constrained devices. In this work, we propose TernaryBERT, which ternarizes the weights in a fine-tuned BERT model. Specifically, we use both approximation-based and loss-aware ternarization methods and empirically investigate the ternarization granularity of different parts of BERT. Moreover, to reduce the accuracy degradation caused by the lower capacity of low bits, we leverage the knowledge distillation technique (Jiao et al., 2019) in the training process. Experiments on the GLUE benchmark and SQuAD show that our proposed TernaryBERT outperforms the other BERT quantization methods, and even achieves comparable performance as the full-precision model while being 14.9x smaller.

1 Introduction

Transformer-based models have shown great power in various natural language processing (NLP) tasks. Pre-trained with gigabytes of unsupervised data, these models usually have hundreds of millions of parameters. For instance, the BERT-base model has 109M parameters, with the model size of 400+MB if represented in 32-bit floating-point format, which is both computation and memory expensive during inference. This poses great challenges for these models to run on resource-constrained devices like cellphones. To alleviate this problem, various methods are proposed to compress these models, like using low-rank approximation (Ma et al., 2019; Lan et al., 2020), weight-sharing (Dehghani et al., 2019; Lan et al., 2020), knowledge distillation (Sanh et al., 2019; Sun et al., 2019; Jiao et al., 2019), pruning (Michel et al., 2019; Voita et al., 2019; Fan et al., 2019), adaptive depth and/or width (Liu et al., 2020; Hou et al., 2020), and quantization (Zafrir et al., 2019; Shen et al., 2020; Fan et al., 2020).

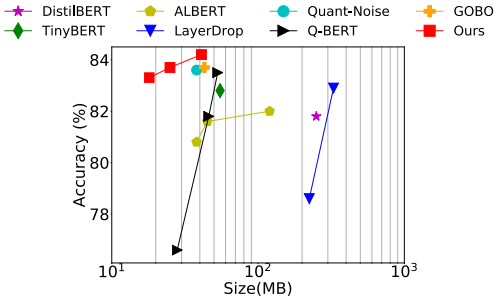


Figure 1: Model Size vs. MNLI-m Accuracy. Our proposed method (red squares) outperforms other BERT compression methods. Details are in Section 4.4.

Compared with other compression methods, quantization compresses a neural network by using lower bits for weight values without changing the model architecture, and is particularly useful for carefully-designed network architectures like Transformers. In addition to weight quantization, further quantizing activations can speed up inference with target hardware by turning floating-point operations into integer or bit operations. In (Prato et al., 2019; Zafrir et al., 2019), 8-bit quantization is successfully applied to Transformer-based models with comparable performance as the full-precision baseline. However, quantizing these models to ultra low bits (e.g., 1 or 2 bits) can be much more challenging due to significant reduction in model capacity. To avoid severe accuracy drop, more complex quantization methods, like mixed-precision quantization (Shen et al., 2020; Zadeh and Moshovos, 2020) and product quantization (PQ) (Fan et al., 2020), are used. However, mixed-precision quantization is unfriendly to some hardwares, and PQ requires extra clustering operations.

[*] Authors contribute equally.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 509–521,
November 16–20, 2020.

Besides quantization, knowledge distillation (Hinton et al., 2015) which transfers knowledge learned in the prediction layer of a cumbersome teacher model to a smaller student model, is also widely used to compress BERT (Sanh et al., 2019; Sun et al., 2019; Jiao et al., 2019; Wang et al., 2020). Instead of directly being used to compress BERT, the distillation loss can also be used in combination with other compression methods (McCarley, 2019; Mao et al., 2020; Hou et al., 2020), to fully leverage the knowledge of teacher model.

In this work, we propose TernaryBERT, whose weights are restricted to $\{-1, 0, +1\}$. Instead of directly using knowledge distillation to compress a model, we use it to improve the performance of ternarized student model with the same size as the teacher model. In this way, we wish to transfer the knowledge from the highly-accurate teacher model to the ternarized student model with smaller capacity, and to fully explore the compactness by combining quantization and distillation. We investigate the ternarization granularity of different parts of the BERT model, and apply various distillation losses to improve the performance of TernaryBERT. Figure 1 summarizes the accuracy versus model size on MNLI, where our proposed method outperforms other BERT compression methods. More empirical results on the GLUE benchmark and SQuAD show that our proposed TernaryBERT outperforms other quantization methods, and even achieves comparable performance as the full-precision baseline, while being much smaller.

2 Related Work

2.1 Knowledge Distillation

Knowledge distillation is first proposed in (Hinton et al., 2015) to transfer knowledge in the logits from a large teacher model to a more compact student model without sacrificing too much performance. It has achieved remarkable performance in NLP (Kim and Rush, 2016; Jiao et al., 2019) recently. Besides the logits (Hinton et al., 2015), knowledge from the intermediate representations (Romero et al., 2014; Jiao et al., 2019) and attentions (Jiao et al., 2019; Wang et al., 2020) are also used to guide the training of a smaller BERT.

Instead of directly being used for compression, knowledge distillation can also be used in combination with other compression methods like pruning (McCarley, 2019; Mao et al., 2020), low-rank approximation (Mao et al., 2020) and dynamic networks (Hou et al., 2020), to fully leverage the knowledge of the teacher BERT model. Although combining quantization and distillation has been explored in convolutional neural networks (CNNs) (Polino et al., 2018; Stock et al., 2020; Kim et al., 2019), using knowledge distillation to train quantized BERT has not been studied. Compared with CNNs which simply perform convolution in each layer, the BERT model is more complicated with each Transformer layer containing both a Multi-Head Attention mechanism and a position-wise Feed-forward Network. Thus the knowledge that can be distilled in a BERT model is also much richer (Jiao et al., 2019; Wang et al., 2020).

2.2 Quantization

Quantization has been extensively studied for CNNs. Popular ultra-low bit weight quantization methods for CNNs can be divided into two categories: approximation-based and loss-aware based. Approximation-based quantization (Rastegari et al., 2016; Li et al., 2016) aims at keeping the quantized weights close to the full-precision weights, while loss-aware based quantization (Hou et al., 2017; Hou and Kwok, 2018; Leng et al., 2018) directly optimizes for the quantized weights that minimize the training loss.

On Transformer-based models, 8-bit fixed-point quantization is successfully applied in fully-quantized Transformer (Prato et al., 2019) and Q8BERT (Zafrir et al., 2019). The use of lower bits is also investigated in (Shen et al., 2020; Fan et al., 2020; Zadeh and Moshovos, 2020). Specifically, In Q-BERT (Shen et al., 2020) and GOBO (Zadeh and Moshovos, 2020), mixed-precision with 3 or more bits are used to avoid severe accuracy drop. However, mixed-precision quantization can be unfriendly to some hardwares. Fan et al. (2020) propose Quant-Noise which quantizes a subset of weights in each iteration to allow unbiased gradients to flow through the network. Despite the high compression rate achieved, the quantization noise rate needs to be tuned for good performance.

In this work, we extend both approximation-based and loss-aware ternarization methods to different granularities for different parts of the BERT model, i.e., word embedding and weights in Transformer layers. To avoid accuracy drop due to the reduced capacity caused by ternarization, various distillation losses are used to guide the training of the ternary model.

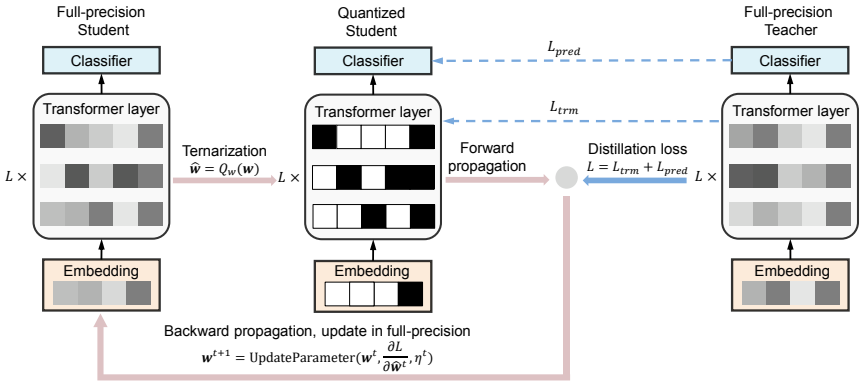


Figure 2: Depiction of the proposed distillation-aware ternarization of BERT model.

3 Approach

In this section, we elaborate on the method of using knowledge distillation to train TernaryBERT, the weights of which take values in $\{-1, 0, +1\}$.

Let the full-precision weight in the BERT model be $\mathbf{w}$, where $\mathbf{w} = \text{vec}(\mathbf{W})$ returns a vector by stacking all the columns of weight matrix $\mathbf{W}$. The corresponding ternarized weight is denoted as $\hat{\mathbf{w}} = Q_w(\mathbf{w})$ where Q_w is the weight ternarization function. The whole framework, which we call Distillation-aware ternarization, is shown in Figure 2. Specifically, at the t-th training iteration, we first ternarize the weights $\mathbf{w}^t$ in the student BERT model to $\hat{\mathbf{w}}^t$. Then we do the forward pass with the ternarized model. After that, the gradient of the distillation loss w.r.t. the quantized weights $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{w}}^t}$ is computed. As is shown in (Courbariaux et al., 2016; Hou and Kwok, 2018), it is important to keep the full-precision weight during training. Hence, we use the full-precision weight for parameter update: $\mathbf{w}^{t+1} = \text{UpdateParameter}(\mathbf{w}^t, \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{w}}^t}, \eta^t)$, where η^t is the learning rate at the t-th iteration.

In the following, we will first introduce what and how to quantize in Section 3.1. Then in Section 3.2, we introduce the distillation loss used to improve the performance of the ternarized model.

3.1 Quantization

The BERT model (Devlin et al., 2019) is built with Transformer layers (Vaswani et al., 2017). A standard Transformer layer includes two main sub-layers: Multi-Head Attention (MHA) module and Feed-Forward Network (FFN).

For the l-th Transformer layer, suppose the input to it is $\mathbf{H}_l \in \mathbb{R}^{n \times d}$ where n and d are the sequence length and hidden state size, respectively. Suppose there are N_H attention heads in each layer, and head h is parameterized by $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V \in \mathbb{R}^{d \times d_h}$ where $d_h = \frac{d}{N_H}$. After computing the attention scores by dot product of queries and keys

$$\mathbf{A}_h = \mathbf{Q}\mathbf{K}^\top = \mathbf{H}_l \mathbf{W}_h^Q \mathbf{W}_h^{K\top} \mathbf{H}_l^\top, \quad (1)$$

the softmax function is applied on the normalized scores to get the output as $\text{head}_h = \text{Softmax}(\frac{1}{\sqrt{d}}\mathbf{A}_h)\mathbf{H}_l\mathbf{W}_h^V$. Denote $\mathbf{W}^* = [\mathbf{W}_1^*, \cdots, \mathbf{W}_{N_H}^*]$ where $*$ can be Q, K, V. The output of the multi-head attention is:

$$\begin{aligned} &\text{MHA}_{\mathbf{W}^Q,\mathbf{W}^K,\mathbf{W}^V,\mathbf{W}^O}(\mathbf{H}_l) \\ = \ &\text{Concat}(\text{head}_1, \cdots, \text{head}_{N_H})\mathbf{W}^O. \end{aligned} \quad (2)$$

The FFN layer composes two linear layers parameterized by $\mathbf{W}^1 \in \mathbb{R}^{d \times d_{ff}}, \mathbf{b}^1 \in \mathbb{R}^{d_{ff}}$ and $\mathbf{W}^2 \in \mathbb{R}^{d_{ff} \times d}, \mathbf{b}^2 \in \mathbb{R}^d$ respectively, where d_{ff} is the number of neurons in the intermediate layer of FFN. Denote the input to FFN as $\mathbf{X}_l \in \mathbb{R}^{n \times d}$, the output is then computed as:

$$\text{FFN}(\mathbf{X}_l) = \text{GeLU}(\mathbf{X}_l\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2. \quad (3)$$

Combining (2) and (3), the forward propagation for the l-th Transformer layer can be written as

$$\begin{aligned} \mathbf{X}_l &= \text{LN}(\mathbf{H}_l + \text{MHA}(\mathbf{H}_l)) \\ \mathbf{H}_{l+1} &= \text{LN}(\mathbf{X}_l + \text{FFN}(\mathbf{X}_l)), \end{aligned}$$

where LN is the layer normalization. The input to the first transformer layer

$$\mathbf{H}_1 = \text{EMB}_{\mathbf{W}^E,\mathbf{W}^S,\mathbf{W}^P}(\mathbf{z}) \quad (4)$$

is the combination of the token embedding, segment embedding and position embedding. Here $\mathbf{z}$

is the input sequence, and $\mathbf{W}^E, \mathbf{W}^S, \mathbf{W}^P$ are the learnable word embedding, segment embedding and position embedding, respectively.

For weight quantization, following (Shen et al., 2020; Zafrir et al., 2019), we quantize the weights $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O, \mathbf{W}^1, \mathbf{W}^2$ in (2) and (3) from all Transformer layers, as well as the word embedding $\mathbf{W}^E$ in (4). Besides these weights, we also quantize the inputs of all linear layers and matrix multiplication operations in the forward propagation. We do not quantize $\mathbf{W}^S, \mathbf{W}^P$, and the bias in linear layers because the parameters involved are negligible. Following (Zafrir et al., 2019), we also do not quantize the softmax operation, layer normalization and the last task-specific layer because the parameters contained in these operations are negligible and quantizing them can bring significant accuracy degradation.

Weight Ternarization. In the following, we discuss the choice of the weight ternarization function Q_w in Figure 2.

Weight ternarization is pioneered in ternaryconnect (Lin et al., 2016) where the ternarized values can take $\{-1, 0, 1\}$ represented by 2 bits. By ternarization, most of the floating-point multiplications in the forward pass are turned into floating-point additions, which greatly reduces computation and memory. Later, by adding a scaling parameter, better results are obtained in (Li et al., 2016). Thus in this work, to ternarize the weights of BERT, we use both approximation-based ternarization method TWN (Li et al., 2016) and loss-aware ternarization LAT (Hou and Kwok, 2018), where the ternary weight $\hat{\mathbf{w}}$ can be represented by the multiplication of a scaling parameter $\alpha > 0$ and a ternary vector $\mathbf{b} \in \{-1, 0, +1\}^n$ as $\hat{\mathbf{w}} = \alpha\mathbf{b}$. Here n is the number of elements in $\hat{\mathbf{w}}$.

In the t-th training iteration, TWN ternarizes the weights by minimizing the distance between the full-precision weight $\mathbf{w}^t$ and ternarized weight $\hat{\mathbf{w}}^t = \alpha^t\mathbf{b}^t$ with following optimization problem (Li et al., 2016)

$$\begin{aligned} \min_{\alpha^t, \mathbf{b}^t} \quad & \|\mathbf{w}^t - \alpha^t\mathbf{b}^t\|_2^2 \\ \text{s.t.} \quad & \alpha^t > 0, \mathbf{b}^t \in \{-1, 0, 1\}^n. \end{aligned} \tag{5}$$

Let $\mathbf{I}_\Delta(\mathbf{x})$ be a thresholding function that $[\mathbf{I}_\Delta(\mathbf{x})]_i = 1$ if $x_i > \Delta$, -1 if $x_i < -\Delta$, and 0 otherwise, where Δ is a positive threshold. Let $\odot$ be element-wise multiplication, the optimal solution of (5) satisfies (Hou and Kwok, 2018):

$\mathbf{b}^t = \mathbf{I}_{\Delta^t}(\mathbf{w}^t)$ and $\alpha^t = \frac{\|\mathbf{b}^t \odot \mathbf{w}^t\|_1}{\|\mathbf{b}^t\|_1}$, where

$$\Delta^t = \arg\max_{\Delta > 0} \frac{1}{\|\mathbf{I}_\Delta(\mathbf{w}^t)\|_1} \left(\sum_{i:|[\mathbf{w}^t]_i| > \Delta} |[\mathbf{w}^t]_i| \right)^2.$$

The exact solution of Δ^t requires an expensive sorting operation (Hou et al., 2017). Thus in (Li et al., 2016), TWN approximates the threshold with $\Delta^t = \frac{0.7\|\mathbf{w}^t\|_1}{n}$.

Unlike TWN, LAT directly searches for the ternary weights that minimize the training loss $\mathcal{L}$. The ternary weights are obtained by solving the optimization problem:

$$\begin{aligned} \min_{\alpha, \mathbf{b}} \quad & \mathcal{L}(\alpha\mathbf{b}) \\ \text{s.t.} \quad & \alpha > 0, \mathbf{b} \in \{-1, 0, 1\}^n. \end{aligned} \tag{6}$$

For a vector $\mathbf{x}$, let $\sqrt{\mathbf{x}}$ be the element-wise square root, $\text{Diag}(\mathbf{x})$ returns a diagonal matrix with $\mathbf{x}$ on the diagonal, and $\|\mathbf{x}\|_Q^2 = \mathbf{x}^\top Q\mathbf{x}$. Problem (6) can be reformulated as solving the following subproblem at the t-th iteration (Hou and Kwok, 2018)

$$\begin{aligned} \min_{\alpha^t, \mathbf{b}^t} \quad & \|\mathbf{w}^t - \alpha^t\mathbf{b}^t\|^2_{\text{Diag}(\sqrt{\mathbf{v}^t})} \\ \text{s.t.} \quad & \alpha^t > 0, \mathbf{b}^t \in \{-1, 0, 1\}^n, \end{aligned} \tag{7}$$

where $\mathbf{v}^t$ is a diagonal approximation of the Hessian of $\mathcal{L}$ readily available as the second moment of gradient in adaptive learning rate optimizers like Adam (Kingma and Ba, 2015). Empirically, we use the second moment in BertAdam[1], which is a variant of Adam by fixing the weight decay (Loshchilov and Hutter, 2019) and removing the bias compensation (Kingma and Ba, 2015). For (7), both an expensive exact solution based on sorting operation, and an efficient approximate solution based on alternative optimization are provided in (Hou and Kwok, 2018). In this paper, we use the more efficient approximate solution.

In the original paper of TWN and LAT, one scaling parameter is used for each convolutional or fully-connected layer. In this work, we extend them to the following two granularities: (i) **layer-wise ternarization** which uses one scaling parameter for all elements in each weight matrix; and (ii) **row-wise ternarization** which uses one scaling parameter for each row in a weight matrix. With more scaling parameters, row-wise ternarization has finer granularity and smaller quantization error.

[1] https://github.com/huggingface/transformers/blob/v0.6.2/pytorch_pretrained_bert/optimization.py

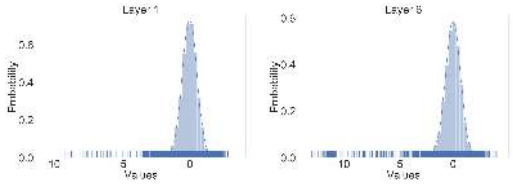


Figure 3: Distribution of the 1st and 6th Transformer layer's hidden representation of the full-precision BERT trained on SQuAD v1.1.

Activation Quantization. To make the most expensive matrix multiplication operation faster, following (Shen et al., 2020; Zafrir et al., 2019), we also quantize the activations (i.e., inputs of all linear layers and matrix multiplication) to 8 bits. There are two kinds of commonly used 8-bit quantization methods: symmetric and min-max 8-bit quantization. The quantized values of the symmetric 8-bit quantization distribute symmetrically in both sides of 0, while those of min-max 8-bit quantization distribute uniformly in a range determined by the minimum and maximum values.

We find that the distribution of hidden representations of the Transformer layers in BERT is skewed towards the negative values (Figure 3). This bias is more obvious for early layers (Appendix A). Thus we use min-max 8-bit quantization for activations as it gives finer resolution for non-symmetric distributions. Empirically, we also find that min-max 8-bit quantization outperforms symmetric quantization (Details are in Section 4.3).

Specifically, for one element x in the activation $\mathbf{x}$, denote $x_{max} = \max(\mathbf{x})$ and $x_{min} = \min(\mathbf{x})$, the min-max 8-bit quantization function is

$$Q_a(x) = \text{round}((x - x_{min})/s) \times s + x_{min},$$

where $s = (x_{max} - x_{min})/255$, is the scaling parameter. We use the straight-through estimator in (Courbariaux et al., 2016) to back propagate the gradients through the quantized activations.

3.2 Distillation-aware Ternarization

The quantized BERT uses low bits to represent the model parameters and activations. Therefore it results in relatively low capacity and worse performance compared with the full-precision counterpart. To alleviate this problem, we incorporate the technique of knowledge distillation to improve performance of the quantized BERT. In this teacher-student knowledge distillation framework, the quantized BERT acts as the student model, and learns to recover the behaviours of the full-precision teacher model over the Transformer layers and prediction layer.

Specifically, inspired by Jiao et al. (2019), the distillation objective for the Transformer layers $\mathcal{L}_{trm}$ consists of two parts. The first part is the distillation loss which distills knowledge in the embedding layer and the outputs of all Transformer layers of the full-precision teacher model to the quantized student model, by the mean squared error (MSE) loss: $\sum_{l=1}^{L+1} \text{MSE}(\mathbf{H}_l^S, \mathbf{H}_l^T)$. The second part is the distillation loss that distills knowledge from the teacher model's attention scores from all heads $\mathbf{A}_l^T$ in each Transformer layer to the student model's attention scores $\mathbf{A}_l^S$ as $\sum_{l=1}^{L} \text{MSE}(\mathbf{A}_l^S, \mathbf{A}_l^T)$. Thus the distillation for the Transformer layers $\mathcal{L}_{trm}$ is formulated as:

$$\mathcal{L}_{trm} = \sum_{l=1}^{L+1} \text{MSE}(\mathbf{H}_l^S, \mathbf{H}_l^T) + \sum_{l=1}^{L} \text{MSE}(\mathbf{A}_l^S, \mathbf{A}_l^T).$$

Besides the Transformer layers, we also distill knowledge in the prediction layer which makes the student model's logits $\mathbf{P}^S$ learn to fit $\mathbf{P}^T$ from the teacher model by the soft cross-entropy (SCE) loss:

$$\mathcal{L}_{pred} = \text{SCE}(\mathbf{P}^S, \mathbf{P}^T).$$

The overall objective of knowledge distillation in the training process of TernaryBERT is thus

$$\mathcal{L} = \mathcal{L}_{trm} + \mathcal{L}_{pred}. \tag{8}$$

We use the full-precision BERT fine-tuned on the downstream task to initialize our quantized model, and the data augmentation method in (Jiao et al., 2019) to boost the performance. The whole procedure, which will be called Distillation-aware ternarization, is shown in Algorithm 1.

4 Experiments

In this section, we evaluate the efficacy of the proposed TernaryBERT on both the GLUE benchmark (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016, 2018). The experimental code is modified from the huggingface transformer library.[2] We use both TWN and LAT to ternarize the weights. We use layer-wise ternarization for weights in Transformer layers while row-wise ternarization

[2]Given the superior performance of Huawei Ascend AI Processor and MindSpore computing framework, we are going to open source the code based on MindSpore (https://www.mindspore.cn/en) soon.

Table 1: Development set results of quantized BERT and TinyBERT on the GLUE benchmark. We abbreviate the number of bits for weights of Transformer layers, word embedding and activations as "W-E-A (#bits)".

		W-E-A (#bits)	Size (MB)	MNLI-m/mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE
	BERT	32-32-32	418 (×1)	84.5/84.9	87.5/90.9	92.0	93.1	58.1	89.8/89.4	90.6/86.5	71.1
	TinyBERT	32-32-32	258 (×1.6)	84.5/84.5	88.0/91.1	91.1	93.0	54.1	89.8/89.6	91.0/87.3	71.8
2-bit	Q-BERT	2-8-8	43 (×9.7)	76.6/77.0	-	-	84.6	-	-	-	-
	Q2BERT	2-8-8	43 (×9.7)	47.2/47.3	67.0/75.9	61.3	80.6	0	4.4/4.7	81.2/68.4	52.7
	TernaryBERT$_{\text{TWN}}$ (ours)	2-2-8	28 (×14.9)	83.3/83.3	86.7/90.1	91.1	92.8	55.7	87.9/87.7	91.2/87.5	72.9
	TernaryBERT$_{\text{LAT}}$ (ours)	2-2-8	28 (×14.9)	83.5/83.4	86.6/90.1	91.5	92.5	54.3	87.9/87.6	91.1/87.0	72.2
	TernaryTinyBERT$_{\text{TWN}}$ (ours)	2-2-8	18 (×23.2)	83.4/83.8	87.2/90.5	89.9	93.0	53.0	86.9/86.5	91.5/88.0	71.8
8-bit	Q-BERT	8-8-8	106 (×3.9)	83.9/83.8	-	-	92.9	-	-	-	-
	Q8BERT	8-8-8	106 (×3.9)	-/-	88.0/-	90.6	92.2	58.5	89.0/-	89.6/-	68.8
	8-bit BERT (ours)	8-8-8	106 (×3.9)	84.2/84.7	87.1/90.5	91.8	93.7	60.6	89.7/89.3	90.8/87.3	71.8
	8-bit TinyBERT (ours)	8-8-8	65 (×6.4)	84.4/84.6	87.9/91.0	91.0	93.3	54.7	90.0/89.4	91.2/87.5	72.2

Table 2: Test set results of the proposed quantized BERT and TinyBERT on the GLUE benchmark.

	W-E-A (#bits)	Size (MB)	MNLI (-m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	score
BERT	32-32-32	418 (×1)	84.3/83.4	71.8/89.6	90.5	93.4	52.0	86.7/85.2	87.6/82.6	69.7	78.2
TernaryBERT$_{\text{TWN}}$	2-2-32	28 (×14.9)	83.1/82.5	71.0/88.6	90.2	93.4	50.1	84.7/83.1	86.9/81.7	68.9	77.3
TernaryBERT$_{\text{TWN}}$	2-2-8	28 (×14.9)	83.0/82.2	70.4/88.4	90.0	92.9	47.8	84.3/82.7	87.5/82.6	68.4	76.9
TernaryTinyBERT$_{\text{TWN}}$	2-2-8	18 (×23.2)	83.8/82.7	71.0/88.8	89.2	92.8	48.1	81.9/80.3	86.9/82.2	68.6	76.6
8-bit BERT	8-8-8	106 (×3.9)	84.2/83.5	71.6/89.3	90.5	93.1	51.6	86.3/85.0	87.3/83.1	68.9	77.9
8-bit TinyBERT	8-8-8	65 (×6.4)	84.2/83.2	71.5/89.0	90.4	93.0	50.7	84.8/83.4	87.4/82.8	69.7	77.7

Algorithm 1 Distillation-aware ternarization.

initialize: A fixed teacher model and a trainable student model using a fine-tuned BERT model.
input: (Augmented) training data set.
output: TernaryBERT $\hat{\mathbf{w}}$.

1: **for** $t = 1, ..., T_{train}$ **do**
2: Get next mini-batch of data;
3: Ternarize $\mathbf{w}^t$ in student model to $\hat{\mathbf{w}}^t$;
4: Compute distillation loss $\mathcal{L}$ in (8);
5: Backward propagation of the student model and compute the gradients $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{w}}^t}$;
6: $\mathbf{w}^{t+1} = \text{UpdateParameter}(\mathbf{w}^t, \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{w}}^t}, \eta^t)$;
7: $\eta^{t+1} = \text{UpdateLearningRate}(\eta^t, t)$.
8: **end for**

for the word embedding, because empirically finer granularity to word embedding improves performance (Details are in Section 4.3).

We compare our proposed method with Q-BERT (Shen et al., 2020) and Q8BERT (Zafrir et al., 2019) using their reported results. We also compare with a weight-ternarized BERT baseline Q2BERT by modifying the min-max 8-bit quantization to min-max ternarization using the released code of Q8BERT.[3] For more direct comparison, we also evaluate the proposed method under the same 8-bit quantization settings as Q-BERT and Q8BERT. When the weights are quantized to 8-bit, we use layer-wise scaling for both the weights in Transformer layers and the word embedding as 8-bit quantization already has high resolution.

4.1 GLUE benchmark

Setup. The GLUE benchmark is a collection of diverse natural language understanding tasks, including textual entailment (RTE), natural language inference (MNLI, QNLI), similarity and paraphrase (MRPC, QQP, STS-B), sentiment analysis (SST-2) and linguistic acceptability (CoLA). For MNLI, we experiment on both the matched (MNLI-m) and mismatched (MNLI-mm) sections. The performance metrics are Matthews correlation for CoLA, F1/accuracy for MRPC and QQP, Spearman correlation for STS-B, and accuracy for the other tasks.

The batch size is 16 for CoLA and 32 for the other tasks. The learning rate starts from 2×10^{-5} and decays linearly to 0 during 1 epoch if trained with the augmented data while 3 epochs if trained with the original data. The maximum sequence length is 64 for single-sentence tasks CoLA and SST-2, and 128 for the rest sentence-pair tasks. The dropout rate for hidden representations and the attention probabilities is 0.1. Since data augmentation does not improve the performance of STS-B, MNLI, and QQP, it is not used on these three tasks.

Results on BERT and TinyBERT. Table 1 shows the development set results on the GLUE

[3] https://github.com/NervanaSystems/nlp-architect.git

benchmark. From Table 1, we find that: 1) For ultra-low 2-bit weight, there is a big gap between the Q-BERT (or Q2BERT) and full-precision BERT due to the dramatic reduction in model capacity. TernaryBERT significantly outperforms Q-BERT and Q2BERT, even with fewer number of bits for word embedding. Meanwhile, TerneryBERT achieves comparable performance with the full-precision baseline with $14.9\times$ smaller size. 2) When the number of bits for weight increases to 8, the performance of all quantized models is greatly improved and is even comparable as the full-precision baseline, which indicates that the setting '8-8-8' is not challenging for BERT. Our proposed method outperforms Q-BERT on both MNLI and SST-2 and outperforms Q8BERT in 7 out of 8 tasks. 3) TWN and LAT achieve similar results on all tasks, showing that both ternarization methods are competitive.

In Table 1, we also apply our proposed quantization method on a 6-layer TinyBERT (Jiao et al., 2019) with hidden size of 768, which is trained using distillation. As can be seen, the quantized 8-bit TinyBERT and TernaryTinyBERT achieve comparable performance as the full-precision baseline.

Test set results are summarized in Table 2. The proposed TernaryBERT or TernaryTinyBERT achieves comparable scores as the full-precision baseline. Specially, the TernaryTinyBERT has only 1.6 point accuracy drop while being 23.2x smaller.

4.2 SQuAD

Setup. SQuAD v1.1 is a machine reading comprehension task. Given a question-passage pair, the task is to extract the answer span from the passage. SQuAD v2.0 is an updated version where the question might be unanswerable. The performance metrics are EM (exact match) and F1.

The learning rate decays from 2×10^{-5} linearly to 0 during 3 epochs. The batch size is 16, and the maximum sequence length is 384. The dropout rate for the hidden representations and attention probabilities is 0.1. Since $\mathcal{L}_{trm}$ is several magnitudes larger than $\mathcal{L}_{pred}$ in this task, we separate the distillation-aware quantization into two stages, i.e., first using $\mathcal{L}_{trm}$ as the objective and then $\mathcal{L}$ in (8).

Results. Table 3 shows the results on SQuAD v1.1 and v2.0. TernaryBERT significantly outperforms Q-BERT and Q2BERT, and is even comparable as the full-precision baseline. For this task, LAT performs slightly better than TWN.

Table 3: Development set results on SQuAD.

	W/E/A (#bits)	Size (MB)	SQuAD v1.1	SQuAD v2.0
BERT	32-32-32	418	81.5/88.7	74.5/77.7
Q-BERT	2-8-8	43	69.7/79.6	-
Q2BERT	2-8-8	43	-	50.1/50.1
TernaryBERT$_{\text{TWN}}$	2-2-8	28	79.9/87.4	73.1/76.4
TernaryBERT$_{\text{LAT}}$	2-2-8	28	80.1/87.5	73.3/76.6

4.3 Ablation Study

In this section, we perform ablation study on quantization, knowledge distillation, initialization, and data augmentation.

Weight Ternarization Granularity. We evaluate the effects of different granularities (i.e., row-wise and layer-wise ternarization in Section 3.1) of TWN on the word embedding and weights in Transformer layers. The results are summarized in Table 4. There is a gain of using row-wise ternarization over layer-wise ternarization for word embedding. We speculate this is because word embedding requires finer granularity as each word contains different semantic information. For weights in the Transformer layers, layer-wise ternarization performs slightly better than row-wise quantization. We speculate this is due to high redundancy in the weight matrices, and using one scaling parameter per matrix already recovers most of the representation power of Transformer layers. Appendix E shows that the attention maps of TernaryBERT (with layer-wise ternarization for weights in Transformer layers) resemble the full-precision BERT. Thus empirically, we use row-wise ternarization for word embedding and layer-wise ternarization for weights in the Transformer layers.

Table 4: Development set results of TernaryBERT$_{\text{TWN}}$ with different ternarization granularities on weights in Transformer layers and word embedding.

Embedding	Weights	MNLI-m	MNLI-mm
layer-wise	layer-wise	83.0	83.0
layer-wise	row-wise	82.9	82.9
row-wise	layer-wise	**83.3**	**83.3**
row-wise	row-wise	83.2	82.9

Activation Quantization. For activations, we experiment on both symmetric and min-max 8-bit quantization with SQuAD v1.1 in Table 5. The weights are ternarized using TWN. As can be seen, the performance of min-max quantization outperforms the symmetric quantization. As discussed in Section 3.1, this may because of the non-symmetric distributions of the hidden representation.

Table 5: Comparison of symmetric 8-bit and min-max 8-bit activation quantization methods on SQuAD v1.1.

W(#bit)	E(#bit)	A(#bit)	EM	F1
2	2	8 (sym)	79.0	86.9
2	2	8 (min-max)	79.9	87.4

Knowledge Distillation. In Table 6, we investigate the effect of distillation loss over Transformer layers (abbreviated as "Trm") and final output logits (abbreviated as "logits") in the training of TernaryBERT$_{\text{TWN}}$. As can be seen, without distillation over the Transformer layers, the performance drops by 3% or more on CoLA and RTE, and also slightly on MNLI. The accuracy of all tasks further decreases if distillation logits is also not used. In particular, the accuracy for CoLA, RTE and SQuAD v1.1 drops by over 5% by removing the distillation compared to the counterpart.

Table 6: Effects of knowledge distillation on the Transformer layers and logits on TernaryBERT$_{\text{TWN}}$. "-Trm-logits" means we use cross-entropy loss w.r.t. the ground-truth labels as the training objective.

	MNLI-m/mm	CoLA	RTE	SQuADv1.1
TernaryBERT	83.3/83.3	55.7	72.9	79.9/87.4
-Trm	82.9/83.3	52.7	69.0	76.6/84.9
-Trm-logits	80.8/81.1	45.4	56.3	74.3/83.2

Initialization and Data Augmentation. Table 7 demonstrates the effect of initialization from a fine-tuned BERT otherwise a pre-trained BERT, and the use of data augmentation in training TernaryBERT. As can be seen, both factors contribute positively to the performance and the improvements are more obvious on CoLA and RTE.

Table 7: Effects of data augmentation and initialization.

	CoLA	MRPC	RTE
TernaryBERT	55.7	91.2/87.5	72.9
-Data augmentation	50.7	91.0/87.5	68.2
-Initialization	46.0	91.0/87.2	66.4

4.4 Comparison with Other Methods

In Figure 1 and Table 8, we compare the proposed TernaryBERT with (i) Other Quantization Methods: including mixed-precision Q-BERT (Shen et al., 2020), post-training quantization GOBO (Zadeh and Moshovos, 2020), as well as Quant-Noise which uses product quantization (Fan et al., 2020); and (ii) Other Compression Methods: including weight-sharing method ALBERT (Lan et al., 2019), pruning method LayerDrop (Fan et al., 2019), distillation methods DistilBERT and TinyBERT (Sanh et al., 2019; Jiao et al., 2019). The result of DistilBERT is taken from (Jiao et al., 2019). The results for the other methods are taken from their original paper. To compare with the other mixed-precision methods which use 3-bit weights, we also extend the proposed method to allow 3 bits (the corresponding model abbreviated as 3-bit BERT, and 3-bit TinyBERT) by replacing LAT with 3-bit Loss-aware Quantization (LAQ) (Hou and Kwok, 2018). The red markers in Figure 1 are our results with settings 1) 2-2-8 TernaryTinyBERT, 2) 3-3-8 3-bit TinyBERT and 3) 3-3-8 3-bit BERT.

Table 8: Comparison between the proposed method and other compression methods on MNLI-m. Note that Quant-Noise uses Product Quantization (PQ) and does not have specific number of bits for each value.

Method	W-E-A (#bits)	Size (MB)	Accuracy (%)
DistilBERT	32-32-32	250	81.6
TinyBERT-4L	32-32-32	55	82.8
ALBERT-E64	32-32-32	38	80.8
ALBERT-E128	32-32-32	45	81.6
ALBERT-E256	32-32-32	62	81.5
ALBERT-E768	32-32-32	120	82.0
LayerDrop-6L	32-32-32	328	82.9
LayerDrop-3L	32-32-32	224	78.6
Quant-Noise	PQ	38	83.6
Q-BERT	2/4-8-8	53	83.5
Q-BERT	2/3-8-8	46	81.8
Q-BERT	2-8-8	28	76.6
GOBO	3-4-32	43	83.7
GOBO	2-2-32	28	71.0
3-bit BERT (ours)	3-3-8	41	84.2
3-bit TinyBERT (ours)	3-3-8	25	83.7
TernaryBERT (ours)	2-2-8	28	83.5
TernaryTinyBERT (ours)	2-2-8	18	83.4

Other Quantization Methods. In mixed precision Q-BERT, weights in Transformer layers with steeper curvature are quantized to 3-bit, otherwise 2-bit, while word embedding is quantized to 8-bit. From Table 8, our proposed method achieves better performance than mixed-precision Q-BERT on MNLI, using only 2 bits for both the word embedding and the weights in the Transformer layers. Similar observations are also made on SST-2 and SQuAD v1.1 (Appendix B).

In GOBO, activations are not quantized. From Table 8, even with quantized activations, our proposed TernaryBERT outperforms GOBO with 2-bit weights and is even comparable to GOBO with 3/4 bit mixed-precision weights.

Other Compression Methods. From Table 8, compared to other popular BERT compression methods other than quantization, the proposed method achieves similar or better performance, while being much smaller.

5 Conclusion

In this paper, we proposed to use approximation-based and loss-aware ternarization to ternarize the weights in the BERT model, with different granularities for word embedding and weights in the Transformer layers. Distillation is also used to reduce the accuracy drop caused by lower capacity due to quantization. Empirical experiments show that the proposed TernaryBERT outperforms state-of-the-art BERT quantization methods and even performs comparably as the full-precision BERT.

References

M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. In *Advances in Neural Information Processing Systems*.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

A. Fan, E. Grave, and A. Joulin. 2019. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin. 2020. Training with quantization noise for extreme model compression. Preprint arXiv:2004.07320.

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

L. Hou and J. T. Kwok. 2018. Loss-aware weight quantization of deep networks. In *International Conference on Learning Representations*.

L. Hou, Yao Q., and J. T. Kwok. 2017. Loss-aware binarization of deep networks. In *International Conference on Learning Representations*.

L. Hou, L. Shang, X. Jiang, and Q. Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. Preprint arXiv:2004.04037.

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

J. Kim, Y. Bhalgat, J. Lee, C. Patel, and N. Kwak. 2019. Qkd: Quantization-aware knowledge distillation. Preprint arXiv:1911.12491.

Y. Kim and A. M. Rush. 2016. Sequence-level knowledge distillation. In *Conference on Empirical Methods in Natural Language Processing*.

D. P. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *AAAI Conference on Artificial Intelligence*.

F. Li, B. Zhang, and B. Liu. 2016. Ternary weight networks. Preprint arXiv:1605.04711.

Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio. 2016. Neural networks with few multiplications. In *International Conference on Learning Representations*.

W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. In *Annual Conference of the Association for Computational Linguistics*.

I. Loshchilov and F. Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, D. Song, and M. Zhou. 2019. A tensorized transformer for language modeling. In *Advances in Neural Information Processing Systems*.

Y. Mao, Y. Wang, C. Wu, C. Zhang, Y. Wang, Y. Yang, Q. Zhang, Y. Tong, and J. Bai. 2020. Ladabert: Lightweight adaptation of bert through hybrid model compression. Preprint arXiv:2004.04124.

J. S. McCarley. 2019. Pruning a bert-based question answering model. Preprint arXiv:1910.06360.

P. Michel, O. Levy, and G. Neubig. 2019. Are sixteen heads really better than one? Preprint arXiv:1905.10650.

A. Polino, R. Pascanu, and D. Alistarh. 2018. Model compression via distillation and quantization. In *International Conference on Learning Representations*.

G. Prato, E. Charlaix, and M. Rezagholizadeh. 2019. Fully quantized transformer for improved translation. Preprint arXiv:1910.10485.

P. Rajpurkar, R. Jia, and P. Liang. 2018. Know what you don't know: Unanswerable questions for squad. Preprint arXiv:1806.03822.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. Preprint arXiv:1606.05250.

M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542.

A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. 2014. Fitnets: Hints for thin deep nets. Preprint arXiv:1412.6550.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. Preprint arXiv:1910.01108.

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

P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou. 2020. And the bit goes down: Revisiting the quantization of neural networks. In *International Conference on Learning Representations*.

S. Sun, Y. Cheng, Z. Gan, and J. Liu. 2019. Patient knowledge distillation for bert model compression. In *Conference on Empirical Methods in Natural Language Processing*, pages 4314–4323.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Annual Conference of the Association for Computational Linguistics*.

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

W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Preprint arXiv:2002.10957.

A. H. Zadeh and A. Moshovos. 2020. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. Preprint arXiv:2005.03842.

O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. 2019. Q8bert: Quantized 8bit bert. Preprint arXiv:1910.06188.

APPENDIX

A Distributions of Hidden Representations on SQuAD v1.1

Figure 4 shows the distribution of hidden representations from the embedding layer and all Transformer layers on SQuAD v1.1. As can be seen, the hidden representations of early layers (e.g. embedding and transformer layers 1-8) are biased towards negative values while those of the rest layers are not.

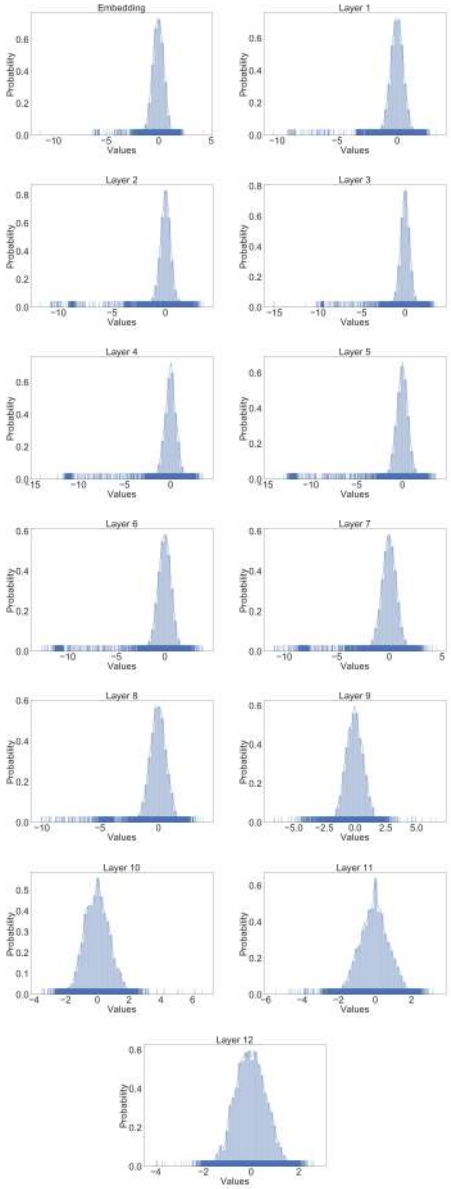


Figure 4: Distribution of Transformer layer's hidden representation of a full-precision BERT trained on SQuAD v1.1.

B More Comparison between TernaryBERT and Q-BERT

We compare with reported results of Q-BERT on SST-2 and SQuAD v1.1 in Table 9. Similar to the observations for MNLI in Section 4.4, our proposed method achieves better performance than mixed-precision Q-BERT on SST-2 and SQuAD v1.1.

Table 9: Comparison between TernaryBERT and mixed-precision Q-BERT.

	W-E-A (#bits)	Size (MB)	SST-2	SQuAD v1.1
BERT	32-32-32	418	93.1	81.5/88.7
Q-BERT	2/3-8-8	46	92.1	79.3/87.0
TernaryBERT$_{\text{TWN}}$	2-2-8	28	**92.8**	**79.9/87.4**

C Training Curve on MNLI

Figure 5 shows the training loss and validation accuracy of TernaryBERT and 8-bit BERT on MNLI-m. As can be seen, 8-bit BERT has smaller loss and higher accuracy than TernaryBERT. There is no significant difference between the learning curve of TernaryBERT using TWN and LAT.

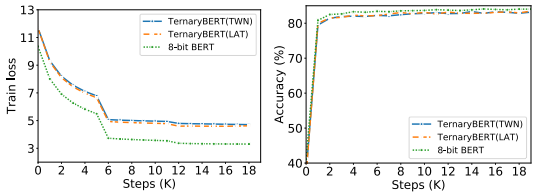


Figure 5: Learning curve of TernaryBERT and 8-bit BERT on MNLI-m.

D 3-bit BERT and TinyBERT

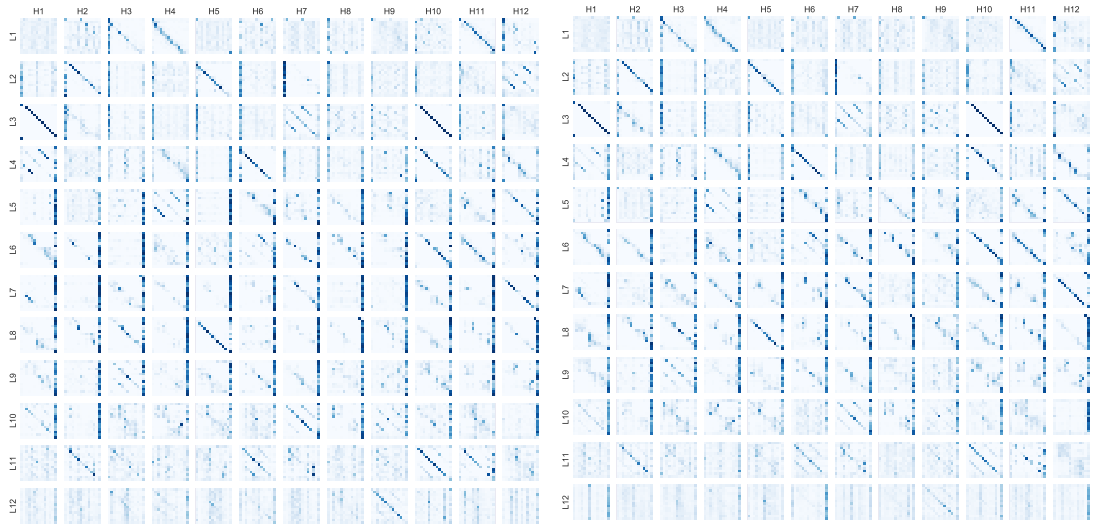
In Table 10, we extend the proposed method to allow 3 bits by replacing LAT with 3-bit Loss-aware Quantization (LAQ). Compared with TernaryBERT$_{\text{LAT}}$, 3-bit BERT performs lightly better on 7 out of 8 GLUE tasks, and the accuracy gap with the full-precision baseline is also smaller.

E Attention Pattern of BERT and TernaryBERT

In Figures 6-9, we compare the attention patterns of the fine-tuned full-precision BERT-base model and the ternarized TernaryBERT$_{\text{TWN}}$ on CoLA and SST-2. CoLA is a task which predicts the grammatical acceptability of a given sentence, and SST-2 is a task of classifying the polarity of movie reviews. As can be seen, the attention patterns of TernaryBERT resemble those in the full-precision BERT.

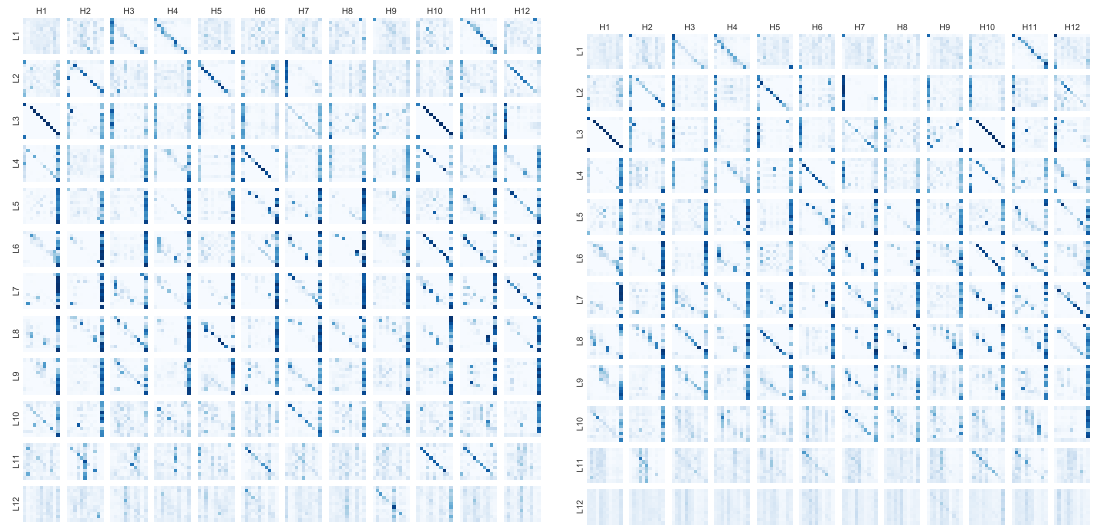
Table 10: Development set results of 3-bit quantized BERT and TinyBERT on GLUE benchmark.

	W-E-A (#bits)	Size (MB)	MNLI-m/mm	QQP	QNLI	SST-2	CoLA	MRPC	STS-B	RTE
TernaryBERT$_{\text{LAT}}$	2-2-8	28 ($\times$14.9)	83.5/83.4	86.6/90.1	91.5	92.5	54.3	91.1/87.0	87.9/87.6	72.2
3-bit BERT	3-3-8	41 ($\times$10.2)	**84.2/84.7**	86.9/90.4	**92.0**	92.8	**54.4**	**91.3/87.5**	**88.6/88.3**	70.8
3-bit TinyBERT	3-3-8	25 ($\times$16.7)	83.7/84.0	**87.2/90.5**	90.7	**93.0**	53.4	91.2/87.3	86.1/85.9	**72.6**



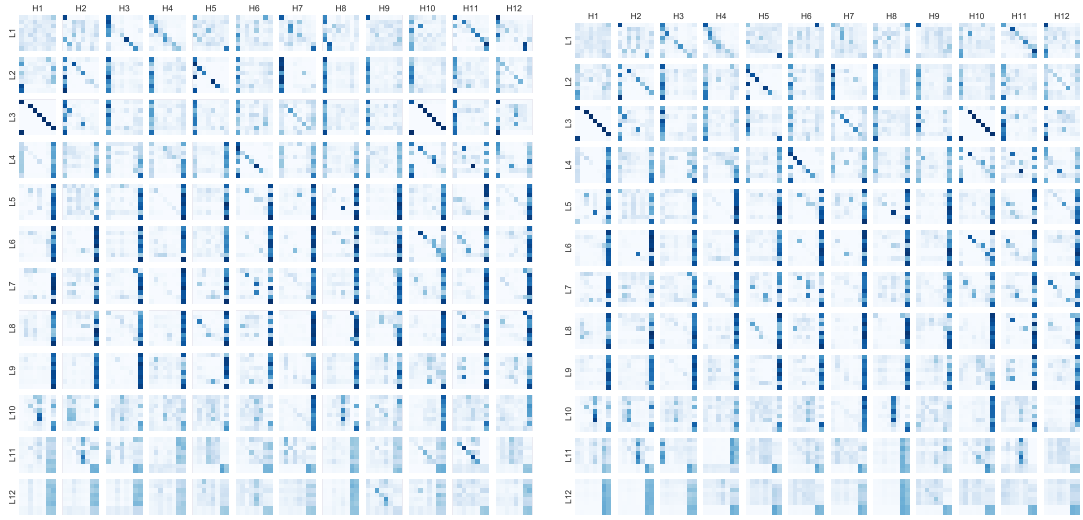
(a) Full-precision BERT. (b) TernaryBERT.

Figure 6: Attention patterns of full-precision and ternary BERT trained on CoLA. The input sentence is "The more pictures of him that appear in the news, the more embarrassed John becomes."



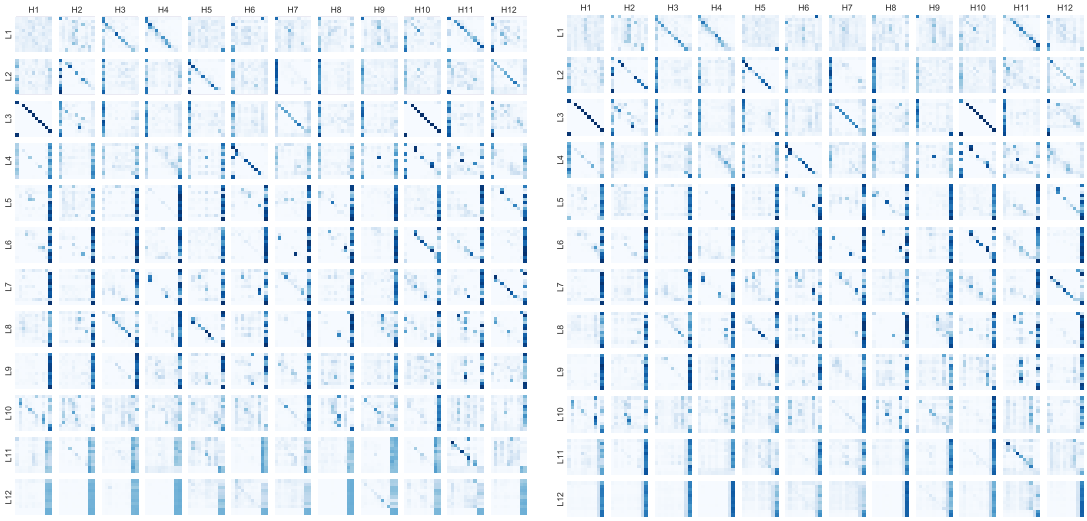
(a) Full-precision BERT. (b) TernaryBERT.

Figure 7: Attention patterns of full-precision and ternary BERT trained on CoLA. The input sentence is "Who does John visit Sally because he likes?"



(a) Full-precision BERT. (b) TernaryBERT.

Figure 8: Attention patterns of full-precision and ternary BERT trained on SST-2. The input sentence is "this movie is maddening."



(a) Full-precision BERT. (b) TernaryBERT.

Figure 9: Attention patterns of full-precision and ternary BERT trained on SST-2. The input sentence is "old-form moviemaking at its best."