BinaryBERT: Pushing the Limit of BERT Quantization

ACL 2021

Haoli Bai[1], Wei Zhang[2], Lu Hou[2], Lifeng Shang[2],
Jing Jin[3], Xin Jiang[2], Qun Liu[2], Michael Lyu[1], Irwin King[1]

[1]The Chinese University of Hong Kong
[2]Huawei Noah's Ark Lab
[3]Huawei Technologies Co., Ltd.

Background

- **Pre-trained language models**
  - Remarkable performance improvement in various natural language tasks
    - Cost of increasing model size and computation
    - Limits the deployment of these huge pre-trained language models to edge devices

  - Model Compression
    - Knowledge distillation
    - Pruning
    - Low-rank approximation
    - Weight sharing
    - Dynamic networks with adaptive depth and/or width
    - Quantization

Background

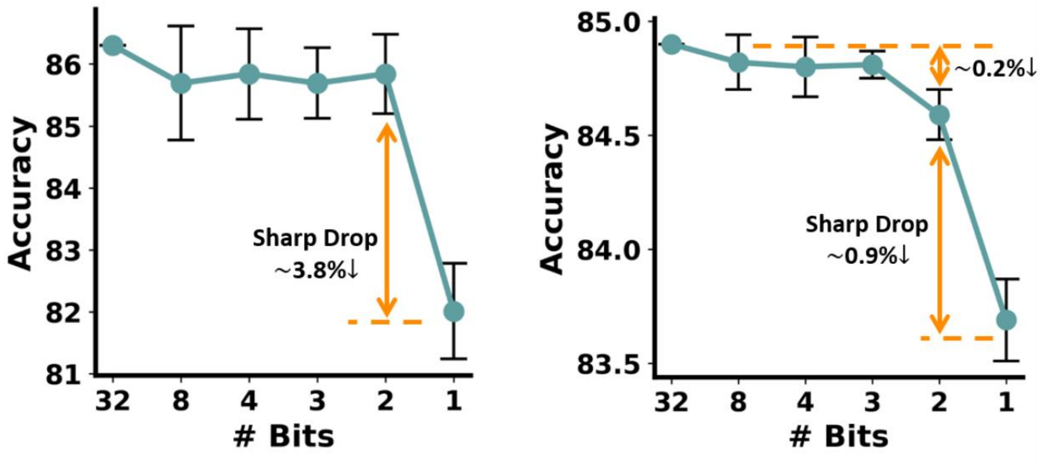
- **Quantization**
  - Replace each 32-bit floating-point parameter with a low-bit fixedpoint representation
  - Existing attempts
    - Even as low as ternary values (2-bit) with minor performance drop
    - None of them achieves the binarization (1-bit)
  - Weight binarization
    - Most 32X reduction in model size
    - Replace most floating-point multiplications with additions
  - Quantizing activation
    - Replace the floating-point addition with int8 and int4 addition
    - Decreasing the energy burden and the area usage on chips (Courbariaux et al., 2015).

M. Courbariaux et al. Binaryconnect: Training deep neural networks with binary weights during propagations. NeurlIPS. 2015.

Introduction

- **Analysis**
  - Performance with varying weight bit-widths and 8-bit activation
    - Sharp performance drop when reducing weight bit-width from 2-bit to 1-bit
  - Loss landscapes of models
    - Full-precision and ternary (2-bit): relatively flat and smooth loss surface
    - Binary (1-bit): rather steep and complex landscape

Performance of quantized BERT



(a) MRPC. (b) MNLI-m.

Background

- **Standard quantization-aware training procedure (zhou et al., 2016)**
  - Latent full-precision weights $\mathbf{w} \in \mathbb{R}^n$
  - Forward propagation
    - Weight $\hat{\mathbf{w}} = \mathcal{Q}(\mathbf{w})$
    - Quantized Function $\mathcal{Q}(\cdot)$
    - Loss $\ell(\hat{\mathbf{w}})$
  - Backward propagation
    - Update latent fullprecision weights $\nabla \ell(\hat{\mathbf{w}})$
    - Straight-through estimator (Courbariaux et al., 2015)

S. Zhou et al. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. Preprint arXiv:1606.06160. 2016.

Background

- **Ternarybert (Zhang et al., 2020)**
  - Quantize the elements in $\mathbf{w}$ to three values $\{\pm\alpha, 0\}$
  - Ternary-weight-network (TWN) (Li et al., 2016)
    - Ternary weight $\hat{w}_i^t = Q(w_i^t) = \begin{cases} \alpha \cdot \mathrm{sign}(w_i^t) & |w_i^t| \geq \Delta \\ 0 & |w_i^t| < \Delta \end{cases}$
    - Sign function $\mathrm{sign}(\cdot)$
    - Threshold parameter $\Delta = \frac{0.7}{n} \|\mathbf{w}^t\|_1$
    - Scaling factor $\alpha = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} |w_i^t|$
    - The number of elements $|\mathcal{I}|$ $\mathcal{I} = \{i \mid \hat{w}_i^t \neq 0\}$

W. Zhang et al. Ternarybert: Distillation-aware ultra-low bit bert. EMNLP. 2020.
F. Li, B. Zhang, and B. Liu. Ternary weight networks. Preprint arXiv:1605.04711. 2016.

Background

- **Binarization (Courbariaux et al., 2015)**
  - Binary-weight-network (BWN) (Hubara et al., 2016)
    - None of them achieves the binarization (1-bit)

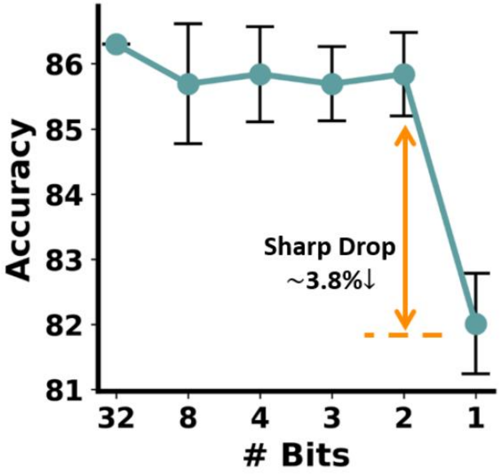
$$\hat{w}_i^b = Q(w_i^b) = \alpha \cdot \mathbf{sign}(w_i^b), \ \alpha = \frac{1}{n}\|\mathbf{w}^b\|_1$$

M. Courbariaux et al. Binaryconnect: Training deep neural networks with binary weights during propagations. NeurlIPS. 2015.
I. Hubara et al. Binarized neural networks. NeurlIPS. 2016.

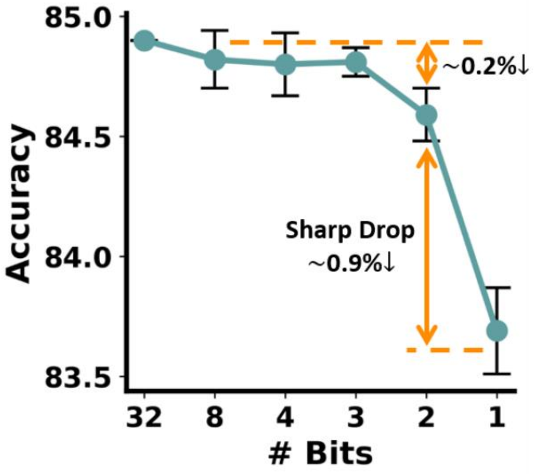
Background

- **Sharp performance drop with weight binarization**
  - Drop mildly from 32-bit to 2-bit, i.e., around 0.6% on MRPC and 0.2% on MNLI-m
  - Drop sharply from the bit-width to one, i.e, 3.8% on MRPC and 0.9% on MNLI-m
  - Weight binarization may severely harm the performance
    - Explain why most current approaches stop at 2-bit weight quantization

Performance of quantized BERT



(a) MRPC.

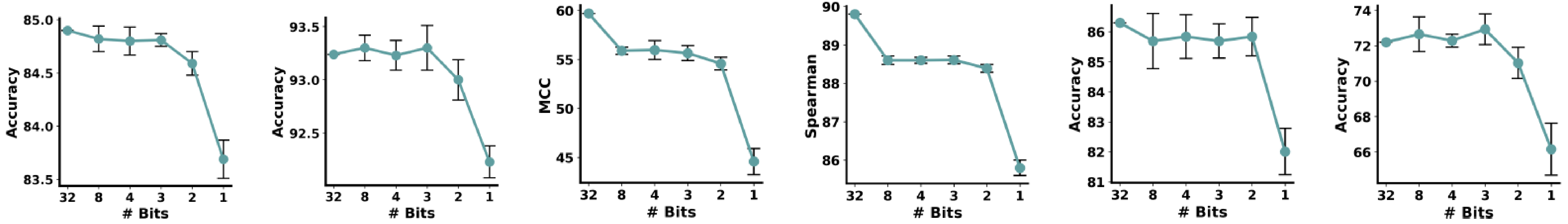


(b) MNLI-m.

Background

- **Sharp performance drop with weight binarization**
  - Drop slowly from full-precision to ternarization
  - Sharp drop by binarization

Performance of quantized BERT with different weight bits and 8-bit activation on the GLUE Benchmarks

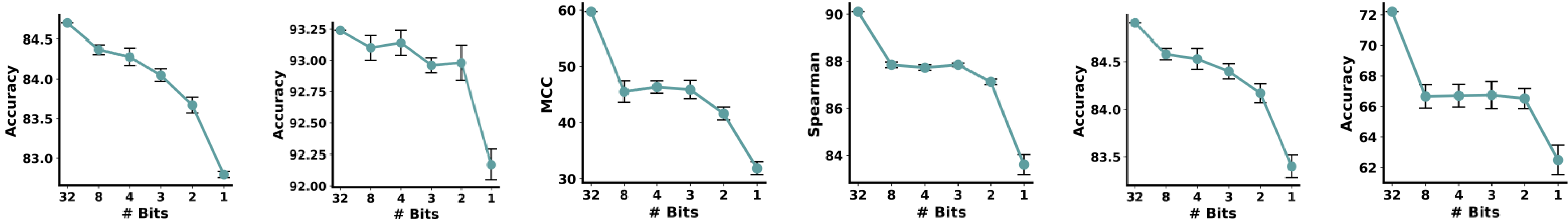


(a) MNLI-m. (b) SST-2. (c) CoLA. (d) STS-B. (e) MRPC. (f) RTE.

Background

- **Sharp performance drop with weight binarization**
  - Drop slowly from full-precision to ternarization
  - Sharp drop by binarization

Performance of quantized BERT with different weight bits and 4-bit activation on the GLUE Benchmarks



(a) MNLI-m. (b) SST-2. (c) CoLA. (d) STS-B. (e) MRPC. (f) RTE.

Background

- **Exploring the Quantized Loss Landscape**
  - $\mathbf{w}_x$, $\mathbf{w}_y$ from value layers of multi-head attention in the first two Transformer layers
  - Perturbations on parameters

$$\tilde{\mathbf{w}}_x = \mathbf{w}_x + x \cdot \mathbf{1}_x$$

$$\tilde{\mathbf{w}}_y = \mathbf{w}_y + y \cdot \mathbf{1}_y$$

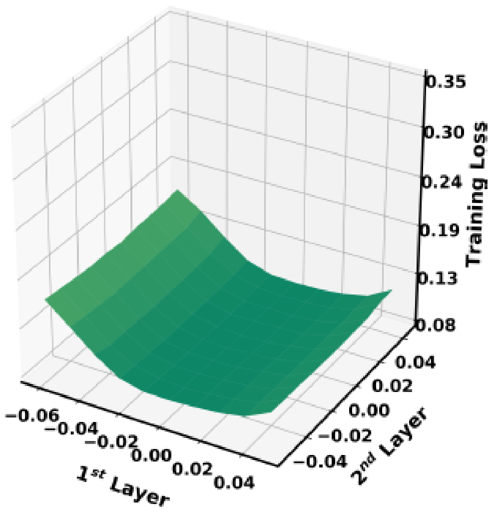
  - Perturbation magnitude $x \in \{\pm 0.2\bar{w}_x, \pm 0.4\bar{w}_x, \ldots, \pm 1.0\bar{w}_x\}$
  - Absolute mean value $\bar{w}_x$ of $\mathbf{W}_x$
  - Vectors with all elements being 1 $\mathbf{1}_x \, \mathbf{1}_y$
  - For each pair of (x, y)
    - Evaluate the corresponding training loss and plot the surface in loss landscape

Y. Nahshan et al. Loss aware post-training quantization. Preprint arXiv:1911.07190. 2019.

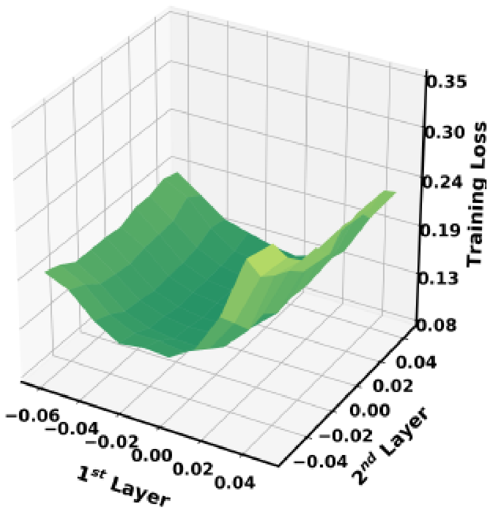
Background

- **Visualization of Loss Landscape**
  - **Full-precision model:** Lowest overall training loss, flat and robust to the perturbation
  - **Ternary model:** Larger perturbations, locally convex and easy to optimize
  - **Binary model:** Steep curvature of loss surface, which attributes to the training difficulty

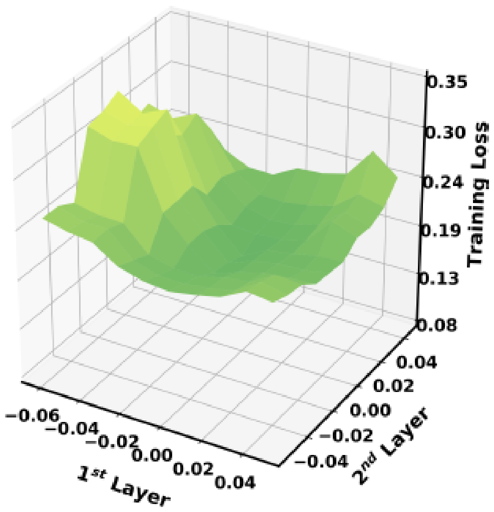
Loss landscapes visualization of BERT on MRPC



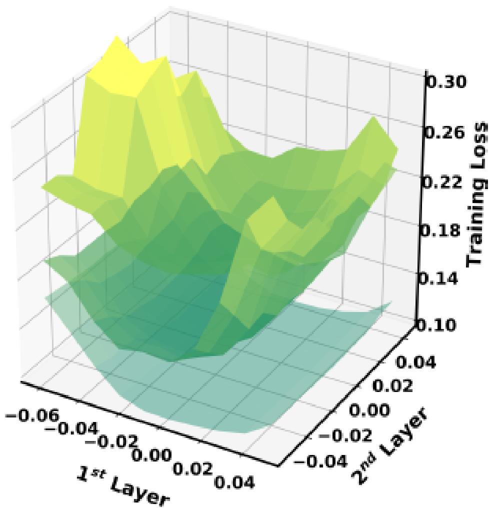
(a) Full-precision Model.



(b) Ternary Model.



(c) Binary Model.



(d) All Together.

Y. Nahshan et al. Loss aware post-training quantization. Preprint arXiv:1911.07190. 2019.

Background

- **Steepness Measurement of Loss Landscape**
  - Start from a local minima $\mathbf{w}$
  - Apply the second order approximation to the curvature
  - Taylor's expansion
    - Loss increase induced by quantizing $\mathbf{w}$ can be approximately upper bounded by

$$\ell(\hat{\mathbf{w}}) - \ell(\mathbf{w}) \approx \boldsymbol{\epsilon}^\top \mathbf{H} \boldsymbol{\epsilon} \leq \lambda_{\max} ||\boldsymbol{\epsilon}||^2$$

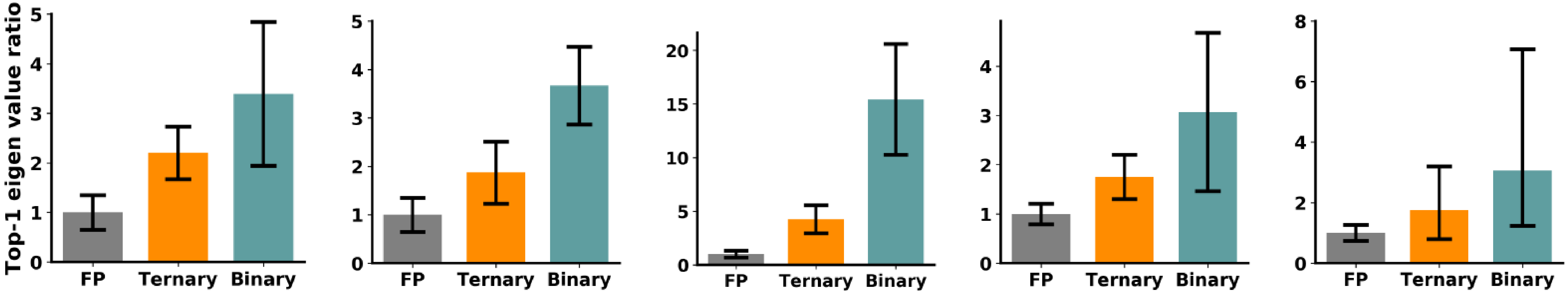
    - Quantization noise $\boldsymbol{\epsilon} = \mathbf{w} - \hat{\mathbf{w}}$
    - Largest eigenvalue of the Hessian $\mathbf{H}$ $\lambda_{\max}$
    - First-order term is skipped due to $\nabla \ell(\mathbf{w}) = 0$
  - Steepness of the loss surface $\lambda_{\max}$

Y. Nahshan et al. Loss aware post-training quantization. Preprint arXiv:1911.07190. 2019.

Background

- **Steepness Measurement of Loss Landscape**
  - Power method to compute $\lambda_{\max}$
  - Top-1 eigen values of MHA-O in binary model are 15X larger than full-precision model
  - Quantization loss increases of full-precision & ternary model are tighter bounded than the binary model

The top-1 eigenvalues of parameters at different Transformer parts of the BERT



(a) MHA-QK. (b) MHA-V. (c) MHA-O. (d) FFN-Mid. (e) FFN-Out.

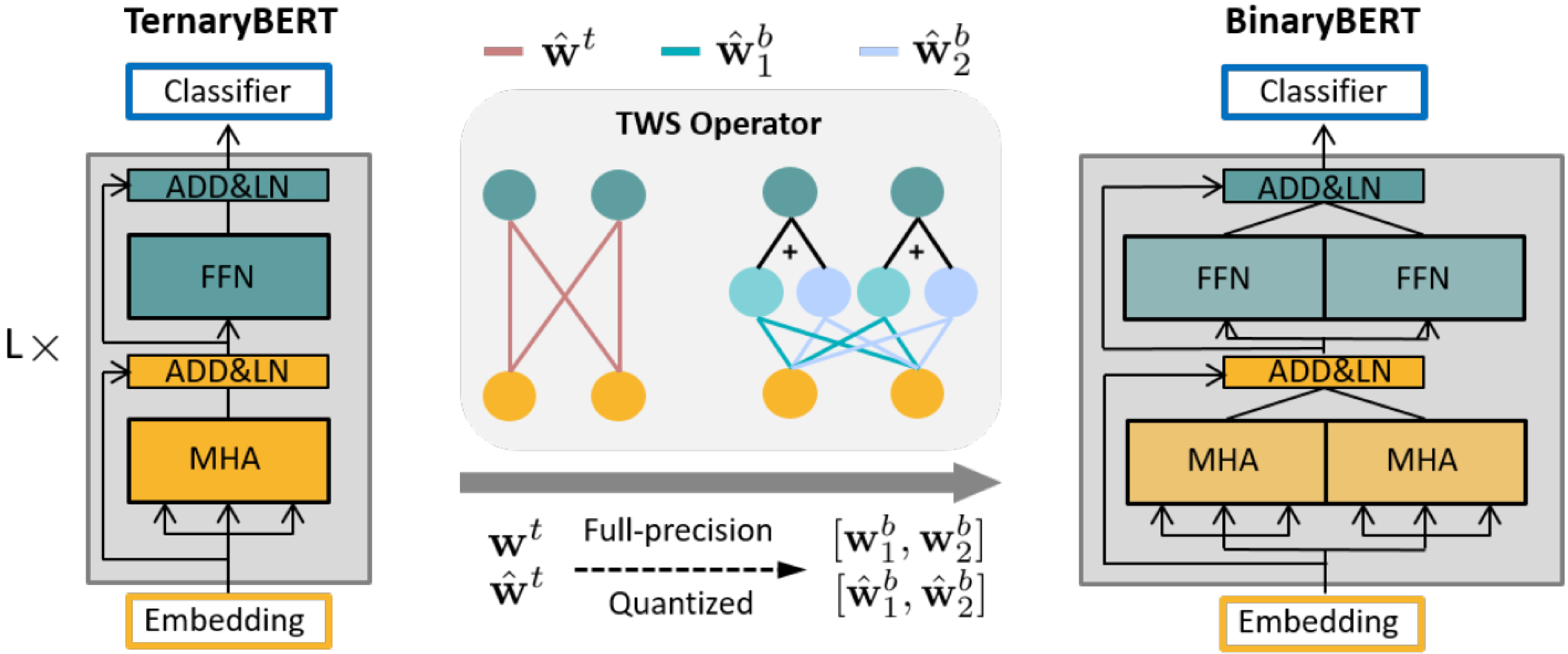
S. Shen et al. Q-bert: Hessian based ultra low precision quantization of bert. AAAI. 2020.

Introduction

- **BinaryBERT**
  - Binarize BERT parameters with quantized activations
  - Ternary weight splitting
    - Takes the ternary model as a proxy
    - Bridge the gap between the binary and full-precision models
  - Adaptive splitting
    - Adaptively perform splitting on the most important ternary modules while leaving the rest as binary
    - Allows flexible sizes of binary models for various edge devices' demands
  - Experimntal result
    - Half-width ternary network is much better than directly-trained binary network
    - Slight performance drop compared to the full-precision BERT-base model (Glue & SQuAD)
    - Adaptive splitting also outperforms other splitting criteria

Introduction

- **Overview**
  - Half-sized ternary BERT -> Ternary weight splitting operator -> Full-sized ternary BERT



Method

- **Ternary Weight Splitting**
  - Exploits the flatness of ternary loss landscape as optimization proxy of binary model
  - Half-sized ternary BERT
    - Split both the latent full-precision & quantized weight
    - Full-precision weight $$\hat{\mathbf{w}}^t = \hat{\mathbf{w}}_1^b + \hat{\mathbf{w}}_2^b$$
    - Quantized weight $$\mathbf{w}^t = \mathbf{w}_1^b + \mathbf{w}_2^b$$

Method

- **Ternary Weight Splitting**
  - Exploits the flatness of ternary loss landscape as optimization proxy of binary model
  - Constrain the latent full-precision weights $\mathbf{w}^t = \mathbf{w}_1^b + \mathbf{w}_2^b$

Cardinality of the set $|\cdot|$

$$w_{1,i}^b = \begin{cases} a \cdot w_i^t & \text{if } \hat{w}_i^t \neq 0 \\ b + w_i^t & \text{if } \hat{w}_i^t = 0, w_i^t > 0 \\ b & \text{otherwise} \end{cases}$$

$$w_{2,i}^b = \begin{cases} (1-a) w_i^t & \text{if } \hat{w}_i^t \neq 0 \\ -b & \text{if } \hat{w}_i^t = 0, w_i^t > 0 \\ -b + w_i^t & \text{otherwise} \end{cases}$$

Equation **(A)**

$$a = \frac{\sum_{i \in \mathcal{I}} |w_i^t| + \sum_{j \in \mathcal{J}} |w_j^t| - \sum_{k \in \mathcal{K}} |w_k^t|}{2 \sum_{i \in \mathcal{I}} |w_i^t|}$$

$$b = \frac{\frac{n}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} |w_i^t| - \sum_{i=1}^{n} |w_i^t|}{2(|\mathcal{J}| + |\mathcal{K}|)}$$

$$\mathcal{I} = \{i \mid \hat{w}_i^t \neq 0\} \quad \mathcal{J} = \{j \mid \hat{w}_j^t = 0 \text{ and } w_j^t > 0\} \quad \mathcal{K} = \{k \mid \hat{w}_k^t = 0 \text{ and } w_k^t < 0\}$$

Method

- **Derivation of Equation (A)**

$$\hat{w}^b_{1,i} = \alpha_1 \text{sign}(w^b_{1,i}) \quad \text{where} \quad \alpha_1 = \frac{1}{n}[\sum_{i\in\mathcal{I}} |aw^t_i| + \sum_{i\in\mathcal{J}} |w^t_j + b| + \sum_{i\in\mathcal{K}} |b|]$$

$$\hat{w}^b_{2,i} = \alpha_2 \text{sign}(w^b_{2,i}) \quad \text{where} \quad \alpha_2 = \frac{1}{n}[\sum_{i\in\mathcal{I}} |(1-a)w^t_i| + \sum_{j\in\mathcal{J}} |-b| + \sum_{k\in\mathcal{K}} |w^t_k - b|]$$

$$\hat{\mathbf{w}}^t = \hat{\mathbf{w}}^b_1 + \hat{\mathbf{w}}^b_2 \quad \text{for those} \quad \hat{w}^t_i = \hat{w}^b_{1,i} + \hat{w}^b_{2,i} = 0$$

$$\frac{1}{n}[\sum_{i\in\mathcal{I}} |aw^t_i| + \sum_{j\in\mathcal{J}} |w^t_j + b| + \sum_{k\in\mathcal{K}} |b|] = \frac{1}{n}[\sum_{i\in\mathcal{I}} |(1-a)w^t_i| + \sum_{j\in\mathcal{J}} |-b| + \sum_{k\in\mathcal{K}} |w^t_k - b|]$$

By assuming $0 < a < 1$ and $b > 0$ $\quad a\sum_{i\in\mathcal{I}} |w^t_i| + \sum_{j\in\mathcal{J}} |w^t_j| = (1-a)\sum_{i\in\mathcal{I}} |w^t_i| + \sum_{k\in\mathcal{K}} |w^t_k|$

$$a = \frac{\sum_{i\in\mathcal{I}} |w^t_i| + \sum_{j\in\mathcal{J}} |w^t_j| - \sum_{k\in\mathcal{K}} |w^t_k|}{2\sum_{i\in\mathcal{I}} |w^t_i|}$$

Method

- **Derivation of Equation (A)**

$$\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}|w_i^t| = \alpha_1 + \alpha_2 \qquad \text{By assuming } 0 < a < 1 \quad \hat{w}_i^t \neq 0 \quad \hat{w}_i^t = \hat{w}_{1,i}^b + \hat{w}_{2,i}^b$$

$$= \frac{1}{n}[\sum_{i\in\mathcal{I}}|aw_i^t| + \sum_{j\in\mathcal{J}}|w_j^t + b| + \sum_{k\in\mathcal{K}}|b|] + \frac{1}{n}[\sum_{i\in\mathcal{I}}|(1-a)w_i^t| + \sum_{j\in\mathcal{J}}|-b| + \sum_{k\in\mathcal{K}}|w_k^t - b|]$$

$$= \frac{1}{n}[\sum_{i\in\mathcal{I}}|w_i^t| + \sum_{j\in\mathcal{J}}|w_j^t| + \sum_{k\in\mathcal{K}}|w_k^t| + 2\sum_{j\in\mathcal{J}}|b| + 2\sum_{k\in\mathcal{K}}|b|]$$

$$= \frac{1}{n}[\sum_{i=1}^{n}|w_i^t| + 2(|\mathcal{J}| + |\mathcal{K}|)\cdot b]$$

$$b = \frac{\frac{n}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}|w_i^t| - \sum_{i=1}^{n}|w_i^t|}{2(|\mathcal{J}| + |\mathcal{K}|)}$$

Method

- **Quantization Details**
  - Layer-wise ternarization (Zhang et al., 2020)
    - One scaling parameter for all elements in the weight matrix
  - Layer-wise ternarization
    - One scaling parameter for each row in the embedding (i.e. word embedding)

  - After splitting, each of the two split matrices has its own scaling factor
  - Aside from weight binarization, Simultaneously quantize activations before all matrix multiplications

  - Following (Zafrir et al., 2019; Zhang et al., 2020)
    - Skip the quantizationn for all layernormalization (LN) layers, skip connections, and bias

  - The last classification layer is also not quantized to avoid a large accuracy drop

W. Zhang et al. Ternarybert: Distillation-aware ultra-low bit bert. EMNLP. 2020.
O. Zafrir et al. Q8bert: Quantized 8bit bert. Preprint arXiv:1910.06188. 2019.

Method

- **Training with Knowledge Distillation**
  - Intermediate-layer distillation from the full-precision teacher
    - Embedding $\ell_{emb} = \mathrm{MSE}(\hat{\mathbf{E}}, \mathbf{E})$
    - Layerwise MHA output $\ell_{mha} = \sum_l \mathrm{MSE}(\hat{\mathbf{M}}_l, \mathbf{M}_l)$
    - FFN output $\ell_{ffn} = \sum_l \mathrm{MSE}(\hat{\mathbf{F}}_l, \mathbf{F}_l)$
    - Objective function $\ell_{int} = \ell_{emb} + \ell_{mha} + \ell_{ffn}$
  - Prediction-layer distillation $\ell_{pred} = \mathrm{SCE}(\hat{\mathbf{y}}, \mathbf{y})$
    - Soft cross-entropy (SCE)

W. Zhang et al. Ternarybert: Distillation-aware ultra-low bit bert. EMNLP. 2020.

Method

- **Adaptive Splitting**
  - Train a mixed-precision model adaptively
    - With sensitive parts being ternary and the rest being binary
  - Split ternary weights into binary ones
  - Enjoy consistent arithmetic precision (1-bit) for all weight matrices
  - Usually easier to deploy than the mixed-precision counterpart

Method

- **Adaptive Splitting**

$\mathbf{u} \in \mathbb{R}_{+}^{Z}$ as the sensitivity vector

Z Total number of splittable weight matrices in all Transformer layers

$\mathbf{c} \in \mathbb{R}_{+}^{Z}$ Cost vector

- Store the additional increase of parameter or FLOPs of each ternary weight matrix against a binary choice

Splitting assignment can be represented as a binary vector $\mathbf{s} \in \{0,1\}^Z$

$s_z = 1$ means to ternarize the z-th weight matrix, and vice versa

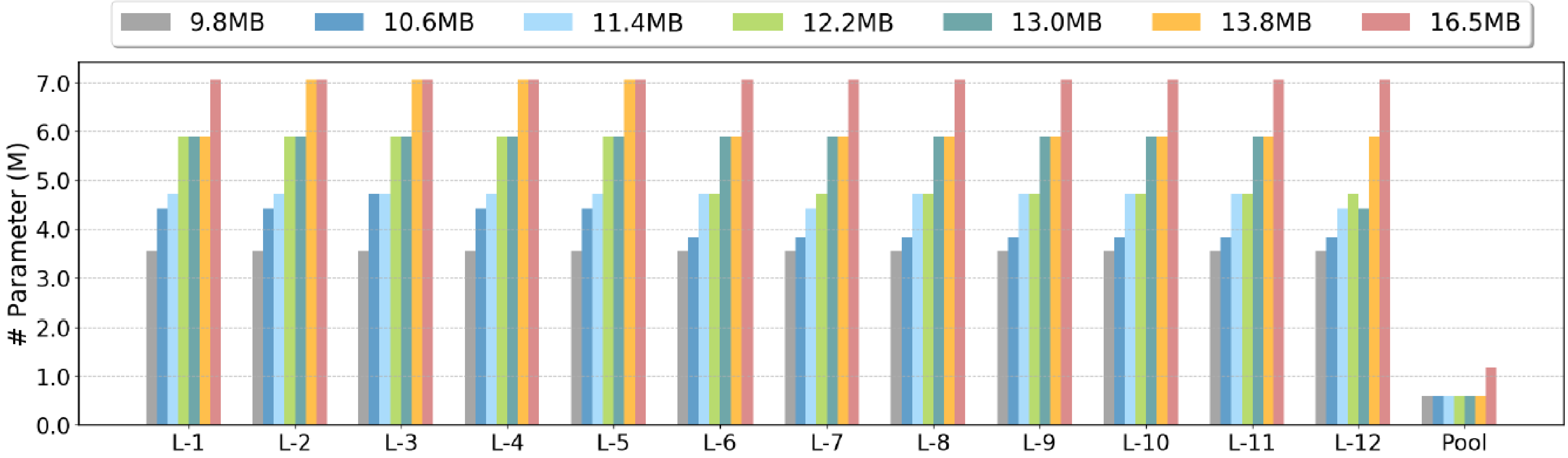
Optimal assignment $\mathbf{s}^*$

$$\boxed{\begin{array}{ll} \max_{\mathbf{s}} & \mathbf{u}^\top \mathbf{s} \\ \text{s.t.} & \mathbf{c}^\top \mathbf{s} \leq \mathcal{C} - \mathcal{C}_0, \ \ \mathbf{s} \in \{0,1\}^Z \end{array}}$$

Baseline efficiency of the half-sized binary network $\mathcal{C}_0$

Method

- **Adaptive Splitting**
  - Architecture visualization for adaptive splitting on MRPC
  - y-axis records the number of parameters split in each layer instead of the storage



Method

- **Adaptive Splitting**
  - Train a mixed-precision model adaptively
    - With sensitive parts being ternary and the rest being binary
  - Split ternary weights into binary ones
  - Enjoy consistent arithmetic precision (1-bit) for all weight matrices
  - Usually easier to deploy than the mixed-precision counterpart

$\mathbf{u} \in \mathbb{R}_{+}^{Z}$ as the sensitivity vector

Z Total number of splittable weight matrices in all Transformer layers

Cost vector $\mathbf{c} \in \mathbb{R}_{+}^{Z}$ res the additional increase of parameter or FLOPs of each ternary weight matrix against a binary choice

Experiment

- **Experimental Setup**
  - Dataset
    - GLUE (Wang et al., 2018): CoLA, STS-B, RTE, MRPC, SST-2, QQP, MNLI-m (matched), MNLI-mm (mismatched)
    - SQuAD (Rajpurkar et al., 2016, 2018)
  - Quantized operations
    - Count the bit-wise operations
    - i,e., the multiplication between an m-bit number and an n-bit number approximately takes mn=64 FLOPs for a CPU with the instruction size of 64 bits

Experiment

- **Experimental Setup**
  - Implementation
    - Backbone: Dynabert (Hou et al., 2020) sub-networks
    - Ternary weight Splitting: Ternary model of width 0:5
    - Adaptive splitting: Split it into a binary model with width 1:0
  - Data augmentation
    - Adopt it with one training epoch in each stage (except for MNLI & QQP)
    - Remove it vanilla training with 6 epochs on these tasks
  - Activation Quantization
    - Further pioneer to study 4-bit activation quantization
    - Uniform quantization can hardly deal with outliers in the activation
    - Then use Learned Step-size Quantization (LSQ) (Esser et al., 2019) to directly learn the quantized values

S. K. Esser et al. Learned step size quantization. ICLR. 2019.

Experiment

- **Results on the GLUE Benchmark**
  - Ternary weight splitting method outperforms BWN
  - 4-bit activation quantization with ternary weight splitting
    - The potential of our approach in extremely low bit quantized models

#	Quant	#Bits (W-E-A)	Size (MB)	FLOPs (G)	DA	MNLI -m/mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
1	-	*full-prec.*	417.6	22.5	-	84.9/85.5	91.4	92.1	93.2	59.7	90.1	86.3	72.2	83.9
2	BWN	1-1-8	13.4	3.1	✗	84.2/84.0	91.1	90.7	92.3	46.7	86.8	82.6	68.6	80.8
3	TWS	1-1-8	16.5	3.1	✗	**84.2/84.7**	**91.2**	**91.5**	**92.6**	**53.4**	**88.6**	**85.5**	**72.2**	**82.7**
4	BWN	1-1-4	13.4	1.5	✗	83.5/83.4	90.9	90.7	**92.3**	34.8	84.9	79.9	**65.3**	78.4
5	TWS	1-1-4	16.5	1.5	✗	**83.9/84.2**	**91.2**	**90.9**	**92.3**	**44.4**	**87.2**	**83.3**	**65.3**	**79.9**
6	BWN	1-1-8	13.4	3.1	✓	84.2/84.0	91.1	91.2	92.7	54.2	88.2	**86.8**	70.0	82.5
7	TWS	1-1-8	16.5	3.1	✓	**84.2/84.7**	**91.2**	**91.6**	**93.2**	**55.5**	**89.2**	86.0	**74.0**	**83.3**
8	BWN	1-1-4	13.4	1.5	✓	83.5/83.4	90.9	91.2	92.5	51.9	87.7	85.5	70.4	81.9
9	TWS	1-1-4	16.5	1.5	✓	**83.9/84.2**	**91.2**	**91.4**	**93.7**	**53.3**	**88.6**	**86.0**	**71.5**	**82.6**

Experiment

• Results on SQuAD Benchmark

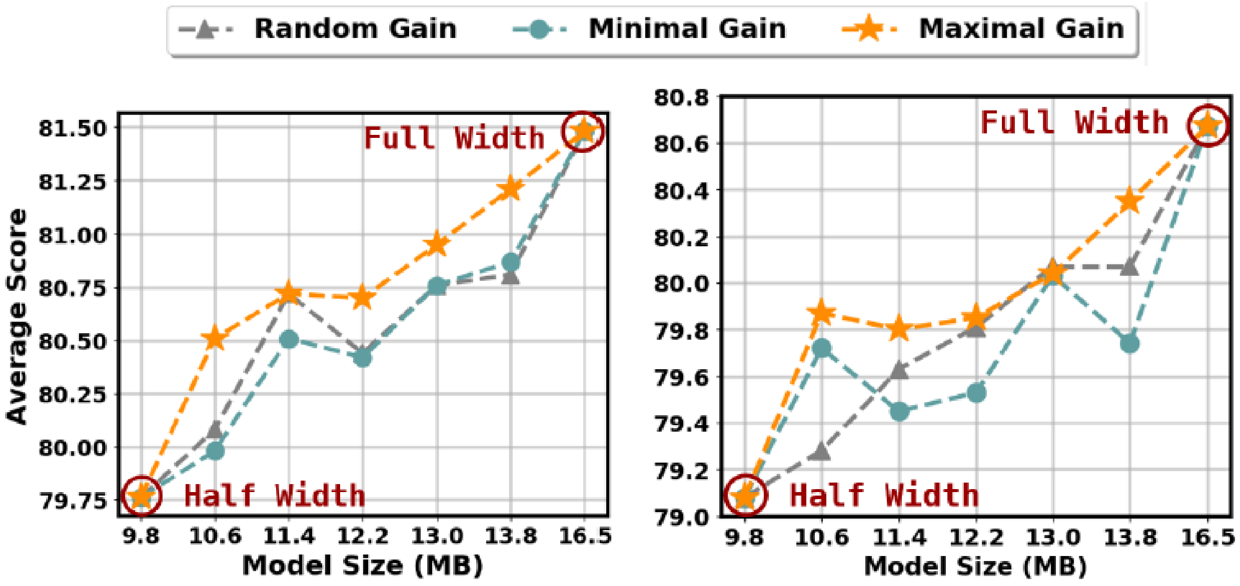
- Ternary weight splitting method outperforms BWN
- Improve the EM score of 4-bit activation by 1.8% and 0.6% on SQuAD v1.1 and v2.0

Quant	**#Bits (W-E-A)**	**Size (MB)**	**FLOPs (G)**	**SQuAD v1.1**	**SQuAD v2.0**
-	*full-prec.*	417.6	22.5	82.6/89.7	75.1/77.5
BWN	1-1-8	13.4	3.1	79.2/86.9	**73.6/76.6**
TWS	1-1-8	16.5	3.1	**80.8/88.3**	73.6/76.5
BWN	1-1-4	13.4	1.5	77.5/85.8	71.9/75.1
TWS	1-1-4	16.5	1.5	**79.3/87.2**	**72.5/75.4**

Experiment

- **Adaptive Splitting**
  - Conversion of mixed ternary and binary precisions for more-fine-grained configurations
  - End-points of 9.8MB and 16.5MB are the half-sized and full-sized BinaryBERT
  - Adaptive splitting generally outperforms baselines under varying model size

Average score over six tasks (QNLI, SST-2, CoLA, STSB, MRPC, RTE)



(a) 8-bit Activation. (b) 4-bit Activation.

Experiment

- **Adaptive Splitting**
  - Conversion of mixed ternary and binary precisions for more-fine-grained configurations
  - End-points of 9.8MB and 16.5MB are the half-sized and full-sized BinaryBERT
  - Adaptive splitting generally outperforms baselines under varying model size

Size (MB)	Strategy	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
10.6	Min.	91.1	93.1	52.8	88.2	85.3	69.3	80.0
	Rand.	90.8	92.7	53.3	88.2	85.5	70.0	80.1
	Max.	91.0	92.7	53.7	88.0	86.5	71.1	80.5
11.4	Min.	91.0	93.0	53.8	88.3	85.5	71.5	80.5
	Rand.	91.0	92.9	54.7	88.4	86.5	70.8	80.7
	Max.	91.0	93.0	54.6	88.4	86.3	71.1	80.7
12.2	Min.	91.1	92.7	53.5	88.5	85.3	71.5	80.4
	Rand.	91.1	92.9	54.1	88.5	86.0	71.8	80.4
	Max.	91.0	92.9	53.8	88.6	86.8	71.1	80.7
13.0	Min.	91.2	92.8	54.8	88.5	85.1	72.2	80.8
	Rand.	91.2	92.9	54.1	88.4	86.0	71.8	80.8
	Max.	91.1	93.1	56.1	88.6	86.1	70.8	81.0
13.8	Min.	91.1	93.0	55.4	88.5	85.8	71.5	80.9
	Rand.	91.5	92.9	54.7	88.5	85.0	72.2	80.8
	Max.	91.4	92.9	55.5	88.7	86.3	72.6	81.2

Size (MB)	Strategy	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
10.6	Min.	90.6	92.6	51.7	87.4	85.3	70.8	79.7
	Rand.	91.1	92.7	51.3	87.6	84.8	68.2	79.3
	Max.	90.9	92.7	53.5	87.5	84.6	70.0	79.9
11.4	Min.	90.9	92.8	50.9	87.6	85.3	69.4	79.5
	Rand.	90.8	92.8	51.7	87.5	84.6	70.4	79.6
	Max.	91.1	92.6	52.1	87.7	85.3	70.0	79.8
12.2	Min.	90.9	92.7	50.8	87.6	84.8	70.4	79.5
	Rand.	91.2	93.0	52.0	87.6	85.1	70.0	79.8
	Max.	90.9	92.9	52.2	87.6	85.1	70.4	79.9
13.0	Min.	91.1	92.8	52.6	87.7	86.3	69.7	80.0
	Rand.	91.3	93.0	52.9	87.8	85.8	69.7	80.1
	Max.	91.3	92.9	53.4	87.8	85.3	69.7	80.1
13.8	Min.	91.1	93.1	51.5	87.9	84.8	70.0	79.7
	Rand.	91.3	92.9	52.3	87.7	85.1	71.1	80.1
	Max.	91.3	92.8	53.6	88.0	85.8	70.8	80.4

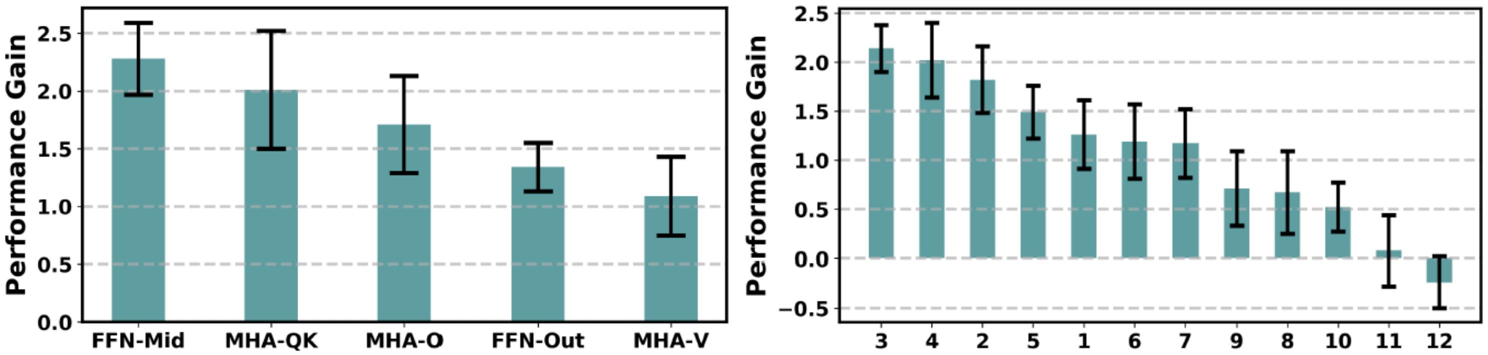
Experiment

- **Discussion**
  - Further Improvement after Splitting
    - Further fine-tuning brings consistent improvement on both 8-bit and 4-bit activation
  - Performance gain of different Transformer parts
    - All numbers are averaged by 10 random runs with standard deviations reported

Further Improvement after Splitting

Quant	#Bits (W-E-A)	SQuAD v1.1	MNLI -m	QNLI	MRPC
$\text{TWN}_{0.5\times}$	2-2-8	80.3/87.9	84.1	91.3	85.7
$\text{TWS}_{1.0\times}$	1-1-8	**80.8/88.3**	**84.2**	**91.6**	**86.0**
$\text{TWN}_{0.5\times}$	2-2-4	78.0/86.4	83.7	90.9	85.5
$\text{TWS}_{1.0\times}$	1-1-4	**79.3/87.2**	**83.9**	**91.4**	**86.0**

Performance gain of different Transformer parts

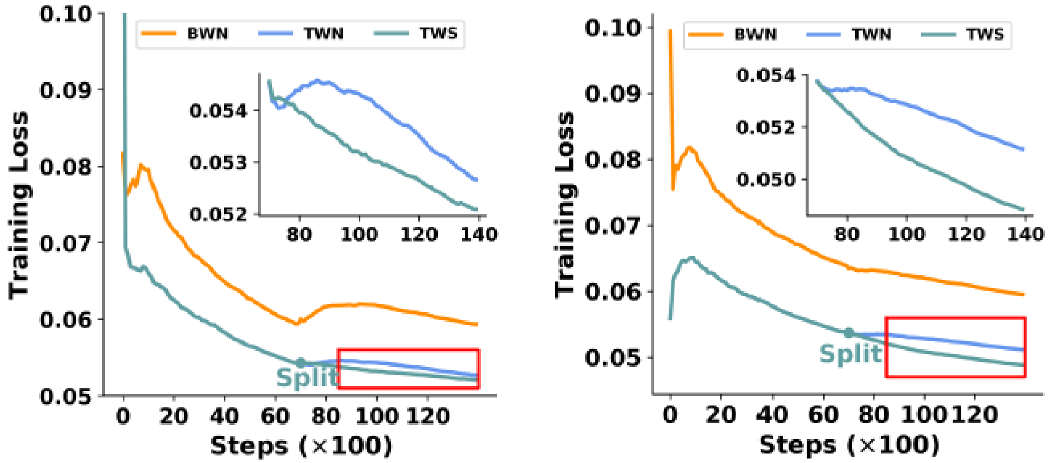


(a) Transformer Parts. (b) Transformer Layers.

Experiment

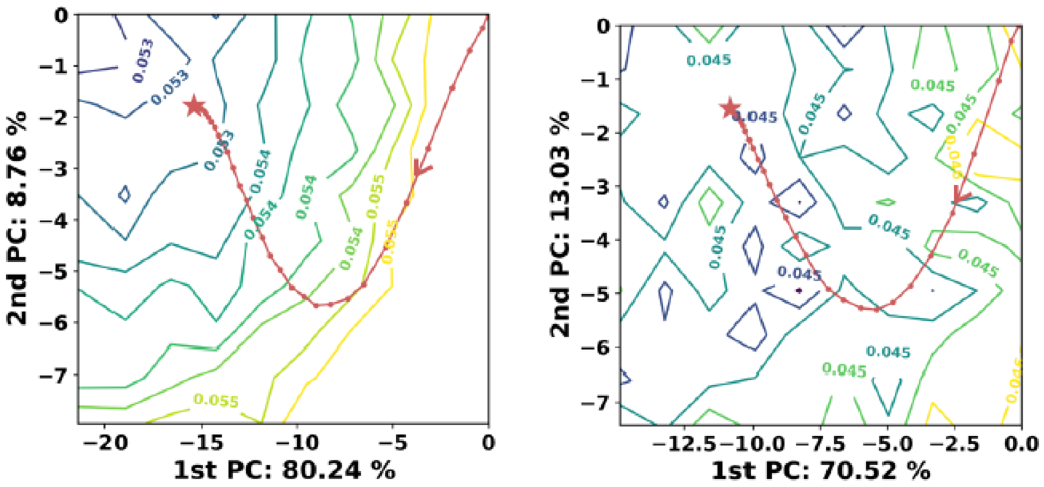
- **Discussion**
  - Training Curves
    - TWS cannot inherit the previous optimizer due to the architecture change
    - Then reset the optimizer and learning rate scheduler of BWN, TWN, TWS
    - TWS attains much lower training loss than BWN, and also surpasses TWN
  - Optimization Trajectory
    - Binary models are optimal solution for 8/4-bit activation quantization on the loss contour

Training curves on MRPC



(a) 8-bit Activation. (b) 4-bit Activation.

Fine-tuning trajectories after splitting



(c) 8-bit Activation. (d) 4-bit Activation.

Conclusion

- **BinaryBERT**
  - Result of the steep and complex loss landscape
    - Directly training a BinaryBERT is hard with a large performance drop

  - Ternary weight splitting improves performance of fine-tuning
  - Adaptive splitting tailor the size of Binary- BERT based on the edge device constraints

  - Our approach significantly outperforms vanilla binary training
  - Achieve state-of-the-art performance on BERT compression