Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT

Sheng Shen*, Zhen Dong*, Jiayu Ye*, Linjian Ma†, Zhewei Yao,
Amir Gholami, Michael W. Mahoney, Kurt Keutzer
University of California at Berkeley
{sheng.s, zhendong, yejiayu, linjian, zheweiy, amirgh, mahoneymw, and keutzer}@berkeley.edu

Abstract. Transformer based architectures have become de-facto models used for a range of Natural Language Processing tasks. In particular, the BERT based models achieved significant accuracy gain for GLUE tasks, CoNLL-03 and SQuAD. However, BERT based models have a prohibitive memory footprint and latency. As a result, deploying BERT based models in resource constrained environments has become a challenging task. In this work, we perform an extensive analysis of fine-tuned BERT models using second order Hessian information, and we use our results to propose a novel method for quantizing BERT models to ultra low precision. In particular, we propose a new group-wise quantization scheme, and we use a Hessian based mix-precision method to compress the model further. We extensively test our proposed method on BERT downstream tasks of SST-2, MNLI, CoNLL-03, and SQuAD. We can achieve comparable performance to baseline with at most 2.3% performance degradation, even with ultra-low precision quantization down to 2 bits, corresponding up to 13× compression of the model parameters, and up to 4× compression of the embedding table as well as activations. Among all tasks, we observed the highest performance loss for BERT fine-tuned on SQuAD. By probing into the Hessian based analysis as well as visualization, we show that this is related to the fact that current training/fine-tuning strategy of BERT does not converge for SQuAD.

1. Introduction. Language model pre-training from large unlabeled data has become the new driving-power for models such as BERT, XLNet, and RoBerta [7, 19, 38]. Built upon Transformer [30], BERT based [7] models significantly improve the state of the art performance when fine-tuned on various Natural Language Processing (NLP) tasks [23, 31]. Recently, many follow-up works push this line of research even further by increasing the model capacity to more than billions of parameters [22]. Though these models achieve cutting-edge results on various NLP tasks, the resulting models have high latency, and prohibitive memory footprint and power consumption for edge inference. This, in turn, has limited the deployment of these models on embedded devices like cellphones or smart assistance, which now require cloud connectivity to function.

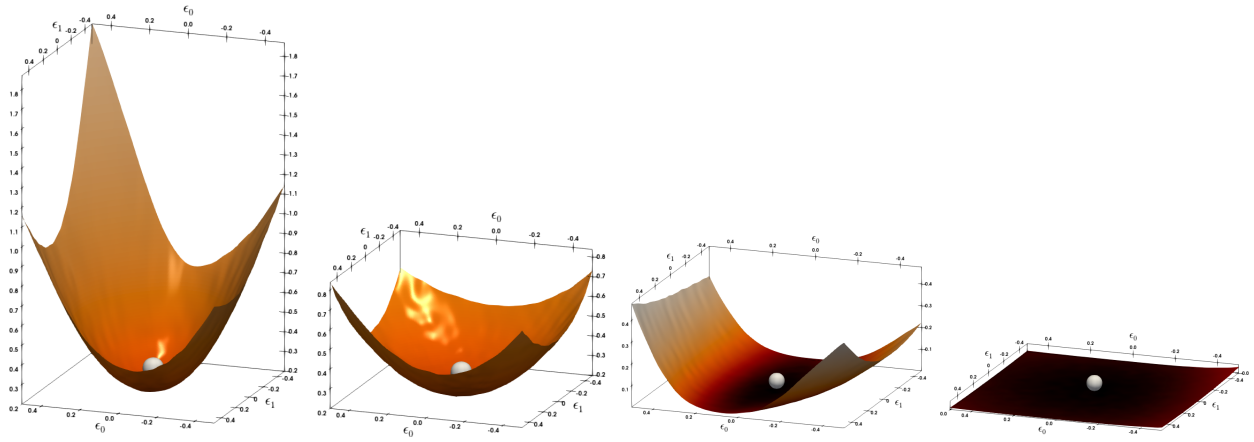
A promising method to address this challenge is quantization, which uses low bit precision for parameter storage and enables low bit hardware operations to speed up inference. The reduced memory footprint and accelerated inference can then enable edge deployment on hardware that supports reduced precision inference such as FPGAs or domain specific accelerators. However, for ultra low-bit setting, e.g., 4 bits, the generalization performance of the quantized model can significantly degrade, and this may not be acceptable for a target application. Historically, in the computer vision area, a large prominent line of work tackles this problem, e.g., different quantization schemes [15, 40], mixed precision quantization [8, 36, 43], etc. However, there is very limited work done on NLP [33, 37], particularly on BERT-based models, which are actually more in need of model compression and acceleration.

In this paper, we focus on ultra low precision quantization of BERT based models, with the goal of minimizing performance degradation while maintaining hardware efficiency. To achieve this, we incorporate a number of novel techniques and propose Q-BERT. The contributions of our work include:

- We apply mixed-precision quantization on BERT, guided by extensive layer-wise analysis of second order information (i.e., Hessian information). We find that BERT exhibits a drastically different Hessian behaviour, as compared with NN models for computer vision [8, 39]. Therefore, we propose a sensitivity measurement based on both mean and variance of the top eigenvalues in order to achieve better mixed-precision quantization, as opposed to [8], which only uses mean value.
- We propose a new quantization scheme, named group-wise quantization, which can alleviate accuracy degradation, without significant increase in hardware complexity. Specifically, in group-wise quantization scheme, we partition each matrix to different groups, each with its unique quantization range and look up table.
- We investigate the bottlenecks in BERT quantization, namely how different factors such as quantization scheme and modules such as embedding, self-attention, and fully-connected layers affect the trade-off between NLP performance and the model compression ratio.

We evaluate Q-BERT in four downstream tasks, including Sentiment Classification, Natural Language Inference, Named Entity Recognition, and Machine Reading Comprehension. Q-BERT achieves 13× compression ratio in weights, 4× smaller activation size, and 4× smaller embedding size, within at most 2.3% accuracy loss. To the best of our knowledge, this is the first work for BERT quantization to ultra low bits with acceptable performance loss.

*Equal Contribution
†Work done while interning at Wave Computing.



(a) MNLI 4th layer (b) MNLI 10th layer (c) CoNLL-03 4th layer (d) CoNLL-03 11th layer

Fig. 1: The loss landscape for different layers in MNLI and CoNNL-03 is illustrated by perturbing the parameters along the first two dominant eigenvectors of the Hessian. The silver sphere shows the point in the parameter space to which the BERT model has converged. Layers that exhibit flatter curvature can be quantized to lower bit precision.

2. Related Work.

Model compression. Model compression is a very active area of research. Efforts in this area could be broadly categorized as follows: (i) new architectures that are compact by design [11, 13]; (ii) automated neural architecture search (NAS) with reward function set as latency or model size [32, 35]; (iii) pruning based methods to reduce model size of existing architectures [16, 18]; (iv) knowledge distillation from a large model to help train a more compact model [1, 10]; (v) hardware and architecture co-design [9]; and (vi) inference quantization [8, 40].

Here we solely focus on quantization [4, 6, 8, 14, 17, 24, 40, 42]. One of the challenges here is that ultra low precision quantization can lead to significant accuracy degradation. Mixed precision quantization [32, 36, 43] and multi-stage quantization [41] have been proposed to solve/alleviate this problem. However, the challenge with mixed-precision quantization is that the search space is exponentially large. For instance, if we have three precision options for a specific layer (2, 4 or 8-bits), then the total search space of each fine-tuned BERT model [7] becomes $3^{12} \approx 5.3\times 10^5$ different precision settings. Recently, [8] proposed a second-order sensitivity based method to address this issue and achieved state-of-the-art results on computer vision tasks. Part of our paper builds upon this prior work and extends the results to include other variations of second order information instead of just the mean value of the Hessian spectrum.

Compressed NLP model. Notable examples for NLP compression work are LSTM and GRU-based models for machine translation and language model [33, 37]. From the recent introduction of Tranformer models, we have observed a significant increase in NLP model size. This is due to the incorporation of very large fully connected layers and attention matrices in Transformers [7, 19, 22, 30, 38]. Model compression is crucial for deploying these models in resource constrained environments. Pilot works addressing this are [3, 21]. From a different angle, [20, 29] have probed the architectural change of self-attention layer to make the Transformer lightweight. There have also been attempts to use distillation to reduce large pre-trained Transformer models such as BERT [7] in [27, 28]. However, significant accuracy loss is observed even for relatively small compression ratio of $4\times$. Here we show that this compression ratio could be increased up to $13\times$, including $4\times$ reduction of embedding layer, with much smaller performance degradation.

3. Methodology.

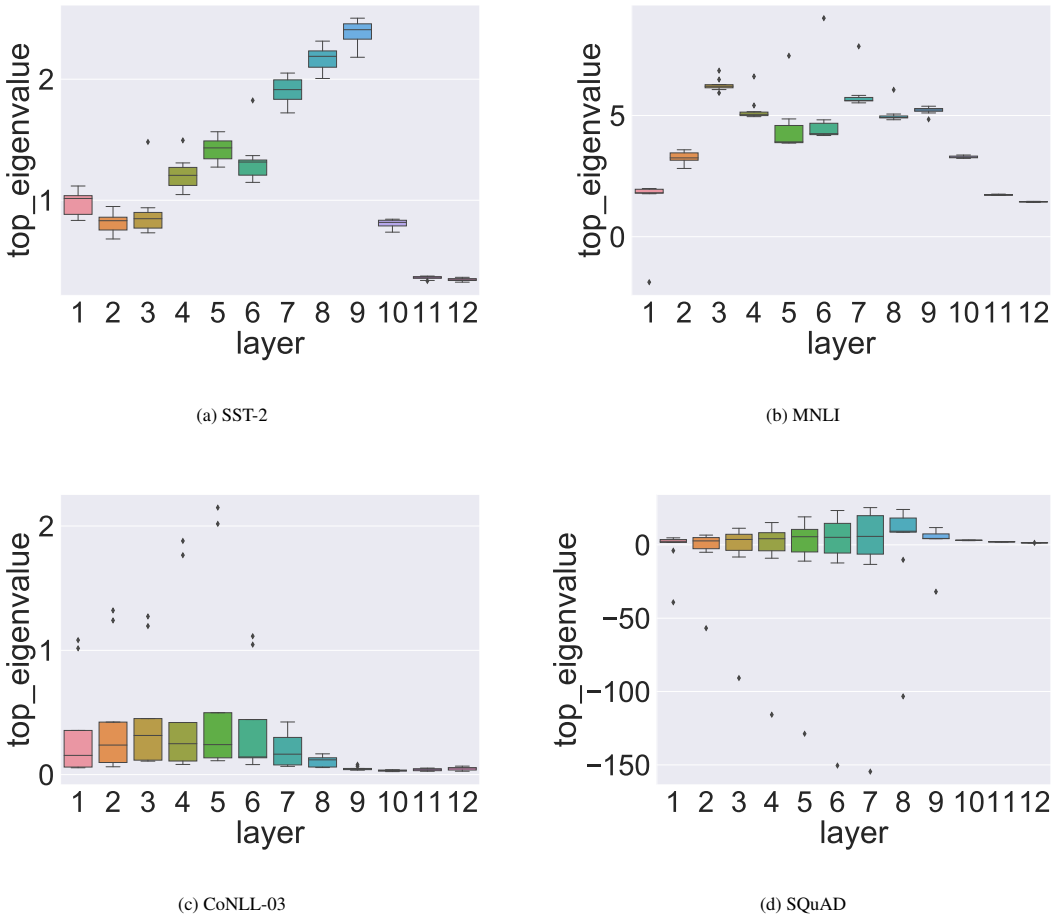
In this section, we introduce our proposed BERT quantization methods, including the mixed precision quantization based on Hessian information, as well as techniques used for the group-wise quantizing scheme.

As in [7], a fine-tuned BERT$_{\text{BASE}}$ model consists of three parts: embedding; Transformer based encoder layers; and output layer. Specifically, assuming $x \in X$ is the input word (sentence) and $y \in Y$ is the corresponding label, we have the loss function L defined as:

$$L(\theta) = \sum_{(x_i, y_i)} \text{CE}(\text{softmax}(W_c(W_n(...W_1(W_e(x_i))))), y_i),$$

where CE is the cross entropy function (or other appropriate loss functions), θ is a combination of W_e, W_1, W_2, ..., W_n and W_c. Here, W_e is the embedding table, W_1, W_2, ..., W_n are the encoder layers, and W_c is the output/classifier layer[1].

[1]Here, we use W_* for both function and its corresponding parameters without confusion.



(a) SST-2 (b) MNLI

(c) CoNLL-03 (d) SQuAD

Fig. 2: From (a) to (d): Top eigenvalue distributions for different encoder layers for SST-2, MNLI, CoNNL-03, SQuAD, respectively. For each task, 10% of the data is used to compute the top eigenvalue, and we perform 10 individual runs to plot the top eigenvalue distribution. It can be seen that layers in the middle have higher mean values, and they also tend to have larger variance than the others. The last three layers have the smallest variance as well as mean values among all layers.

The size of parameters in BERT$_{\text{BASE}}$ model is 91MB for embedding, 325MB for encoder and 0.01MB for output. We do not quantize the output layer due to its negligible size, and focus on quantizing both the embedding and encoder layers. As will be discussed in Sec. 5.1, we find that the embedding layer is more sensitive to quantization than the encoder layers. As a result, we quantize embedding and encoder parameters in different ways. The quantization schemes we used are explained in detail in the following sections.

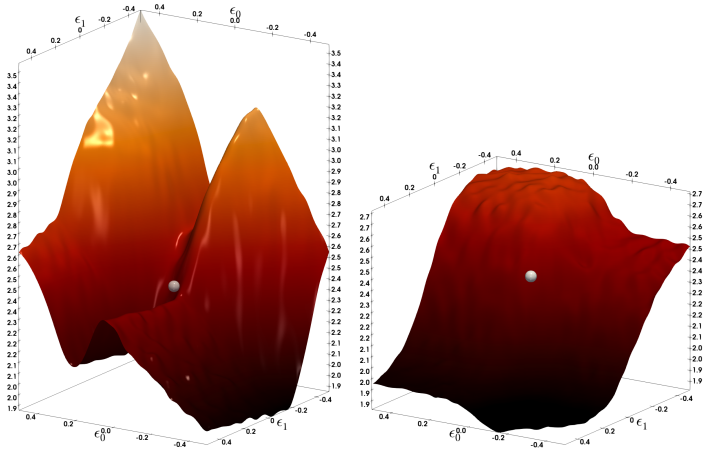
3.1. Quantization process. General NN inference is performed in floating point precision for both weights and activations. Quantization restricts the network weights to a finite set of values defined as follows:

$$Q(z) = q_j, \quad \text{for } z \in (t_j, t_{j+1}],$$

where Q is quantization operator, z is a real valued input tensor (activation or a weight), and $(t_j, t_{j+1}]$ denotes an interval in the real numbers ($j = 0, \ldots, 2^k - 1$). Here k is the quantization precision for a specific layer.

There are multiple choices for quantization function Q. Here we use uniform quantization function, where the range of floating point values in a tensor is equally split [12, 42] and then represented by unsigned integers in $\{0, \ldots, 2^k - 1\}$. It should be noted that a non-uniform quantizer can potentially further increase the accuracy. However, we solely focus on uniform quantization since it allows more efficient and easier hardware implementation. To backpropogate gradients through Q, which is non-differentiable, we use the Straight-through Estimator (STE) [2]. See Appendix A for more details about the forward and backward propagation during the entire quantization process.

3.2. Mixed precision quantization. Different encoder layers are attending to different structures [5], and it is expected that they exhibit different sensitivity. Thus, assigning the same number of bits to all the layers is sub-optimal.



(a) SQuAD 7th layer (b) SQuAD 11th layer

Fig. 3: The loss landscape for different layers in SQuAD is illustrated by perturbing the parameters along the first two dominant eigenvectors of the Hessian. The silver sphere shows the point in the parameter space to which the BERT model has converged. Note that the stopping point of SQuAD has negative eigenvalues for both layers. This could be the reason we observed relatively larger performance drop in SQuAD after quantization; see Tab. 1d.

This scenario is more critical if the targeted model size is very small, which requires ultra low precision such as 4-bits or 2-bits. As a result we explore mixed-precision quantization, where we assign more bits to more sensitive layers in order to retain performance.

In [8], a Hessian AWare Quantization (HAWQ) is developed for mixed-bits assignments. The main idea is that the parameters in NN layers with higher Hessian spectrum (i.e., larger top eigenvalues) are more sensitive to quantization and require higher precision, as compared to layers with small Hessian spectrum (i.e., smaller top eigenvalues). However, there exist 7M parameters for each encoder layer in BERT$_{\text{BASE}}$. Given that the Hessian of each layer is a matrix of size $7M \times 7M$, there is a common misconception that computing second order statistics is infeasible. However, the Hessian spectrum can be computed by a matrix-free power iteration method [39], and this does not require explicit formation of the operator. To illustrate this, we take the first encoder layer as an example. Denoting the gradient of the first encoder layer as g_1, for a random vector v with the same dimension as g_1, we have

$$\frac{\partial g_1^T v}{\partial W_1} = \frac{\partial g_1^T}{\partial W_1} v + g_1^T \frac{\partial v}{\partial W_1} = \frac{\partial g_1^T}{\partial W_1} v = H_1 v, \tag{3.1}$$

where H_1 is Hessian matrix of the first encoder. Here the second equation comes from the fact that v is independent to W_1. The top eigenvalue then can be computed by power iteration, as shown in Alg. 1 in Appendix. We denote λ_i as the top eigenvalue of i-th encoder layer. Using this approach, we show in Fig. 2 the distribution of top Hessian eigenvalue for different layers of BERT$_{\text{BASE}}$. Different layers exhibit different magnitude of eigenvalues even though all layers have exactly same structure and size.

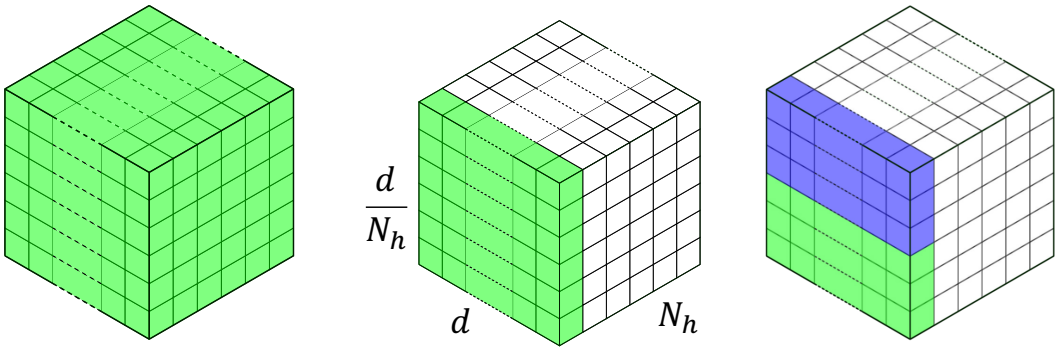
The above Hessian based approach was used in [8], where top eigenvalues are computed and averaged for different training data. More aggressive quantization is performed for layers that have smaller top eigenvalue, which corresponds to flatter loss landscape as in Fig. 1. However, we find that assigning bits based only on the average top eigenvalues is infeasible for many NLP tasks. As shown in Fig. 2, top eigenvalues of Hessian for some layers exhibits very high variance with respect to different portion of the input dataset. As an example, the variance of the 7th layer for SQuAD stays larger than 61.6 while the mean of that layer is around 1.0, even though each data point corresponds to 10% of the entire dataset (which is 9K samples). To address this, we use the following metric instead of just using mean value,

$$\Omega_i \triangleq |\text{mean}(\lambda_i)| + \text{std}(\lambda_i), \tag{3.2}$$

where λ_i is the distribution of the top eigenvalues of H_i, calculated with 10% of training dataset.[2] After Ω_i is computed, we sort them in descending order, and we use it as a metric to relatively determine the quantization precision. We then perform quantization-aware fine-tuning based on the selected precision setting.

An important technical point that we need to emphasize is that our method expects that before performing quantization the trained model has converged to a local minima. That is, the practitioners who trained BERT and performed its

[2]Without confusion, we use λ_i for both single top eigenvalue and its distribution with respect to 10% of the data.



(a) Layer-wise (b) Group-wise (N_h group) (c) Group-wise ($2N_h$ group)

Fig. 4: The overview of Group-wise Quantization Method. We illustrate this with value matrices of a multi-head self attention layer. Here N_h(number of heads) value matrices W_v are concatenated together, which results in a 3-d tensor. The same color denotes the same group with a shared quantization range. As shown in (a), for layer-wise quantization, the entire 3-d tensor will be quantized from a universal quantization range into discrete unsigned integers. A special case of group-wise quantization in (b) is that we treat each dense matrix as a group, and every matrix can have its own quantization range. We show a more general case in (c), where we partition each dense matrix w.r.t. output neuron and bucket every continuous $\frac{d}{2N_h}$ output neurons as a group.

fine-tuning for downstream tasks should have chosen the hyper-parameters and number of iterations such that a local minima has been reached. The necessary optimality conditions are zero gradient, and positive curvature (i.e., positive Hessian eigenvalue). In our analysis, we observed that for the three tasks of MNLI, CoNLL-03, and SST-2 the top Hessian eigenvalue is indeed positive for (see Fig. 1, and Fig. 6 in Appendix). However, we find that the BERT model fine-tuned for SQuAD has actually *not* converged to a local minima, as evident in the Hessian eigenvalues shown in Fig. 2(d), where we observe very large negative eigenvalues. Directly visualizing the loss landscape also shows this very clearly as in Fig. 3. Because of this, our expectation is that performing quantization on SQuAD would lead to higher performance degradation as compared to other tasks, and this is indeed the case as will be discussed next.

3.3. Group-wise Quantization. Assume that the input sequence has n words and each word has a d-dim embedding vector ($d = 768$ for BERT$_{\text{BASE}}$), i.e., $x = (x(1), \ldots, x(n))^T \in \mathbb{R}^{n \times d}$. In Transformer encoder, each self-attention head has 4 dense matrix, i.e., $W_k, W_q, W_v \in \mathbb{R}^{\frac{d}{N_h} \times d}, W_o \in \mathbb{R}^{d \times \frac{d}{N_h}}$, where N_h is the number of attention heads. Here W_k, W_q, W_v and W_o stand for key, query, value and output weight matrix. Each self-attention head computes the weighted sum as

$$\text{Att}(x, x(j)) = W_o \sum_{i=1}^{n} \text{softmax}\left(\frac{x(j)^T W_q^T W_k x(i)}{\sqrt{d}}\right) W_v x(i).$$

Through this reparametrization, the multi-head self-attention (MHSA) will add these features into the final output, that is we will have $\sum_{i=1}^{N_h} \text{Att}_i(x, x(j))$. Directly quantizing each 4 matrices in MHSA as an entirety with the same quantization range can significantly degrade the accuracy, since there are more than 2M parameters in total, which corresponds to $4 \times 12 \times 64 = 3072$ output neurons, and the weights corresponding to each neuron may lie in different range of real numbers. Channel-wise quantization can be used to alleviate this problem in convolutional neural networks, where each convolutional kernel can be treated as a single output channel and have its own quantization range. However, this cannot be directly applied for dense matrices, since each dense matrix itself is a single kernel. Therefore, we propose group-wise quantization for attention-based models. We treat the individual matrix W with respect to each head in one dense matrix of MHSA as a group so there will be 12 groups. Furthermore, in each group, we bucket sequential output neurons together as sub-groups, e.g., each 6 output neurons as one sub-group so there are $12 \times \frac{64}{6} = 128$ sub-group in total (the hidden dimension in each head of BERT$_{\text{BASE}}$ is $\frac{768}{12} = 64$). Each sub-group can have its own quantization range. An illustration is shown in Fig. 4 for W_v, where we concatenate N_h value matrix W_v to be a 3-d tensor. For layer-wise quantization, the entire 3-d tensor will be quantized into the same range of discrete numbers, as shown in Fig. 4a. A special case of group-wise quantization is that we treat each dense matrix as a group, and every matrix can have its own quantization range as shown in Fig. 4b. A more general case in Fig. 4c is that we partition each dense matrix with respect to output neuron, and we bucket every continuous $\frac{d}{2N_h}$ output neurons as a group. The effect of finer group-wise quantization is further investigated in Sec. 4.2.

4. Experiment. In this section, we describe our experiments on evaluating the proposed Q-BERT on four different NLP tasks. Details of the datasets are shown in Appendix B. To the best of our knowledge, there is no published work done on BERT quantization at this point, so we report Direct quantization (DirectQ), i.e., quantization without mixed-precision and group-wise quantization as a baseline.

Table 1: Quantization results for BERT$_{\text{BASE}}$ on Natural Language Understanding tasks. Results are obtained with 128 groups in each layer. We abbreviate quantization bits used for weights as "w-bits", embedding as "e-bits", model size in MB as "Size", and model size without embedding layer in MB as "Size-w/o-e". For simplicity and efficacy, all the models except for Baseline are using 8-bits activation. Furthermore, we compare Q-BERT with direct quantization method ("DirectQ") without using mixed precision or group-wise quantization. Here "MP" refers to mixed-precision quantization.

(a) SST-2

Method	w-bits	e-bits	Acc	Size	Size-w/o-e
Baseline	32	32	93.00	415.4	324.5
Q-BERT	8	8	92.88	103.9	81.2
DirectQ	4	8	85.67	63.4	40.6
Q-BERT	4	8	**92.66**	63.4	40.6
DirectQ	3	8	82.86	53.2	30.5
Q-BERT	3	8	**92.54**	53.2	30.5
Q-BERT$_{\text{MP}}$	2/4 MP	8	**92.55**	53.2	30.5
DirectQ	2	8	80.62	43.1	20.4
Q-BERT	2	8	**84.63**	43.1	20.4
Q-BERT$_{\text{MP}}$	2/3 MP	8	**92.08**	**48.1**	**25.4**

(b) MNLI

Method	w-bits	e-bits	Acc m	Acc mm	Size	Size w/o-e
Baseline	32	32	84.00	84.40	415.4	324.5
Q-BERT	8	8	83.91	83.83	103.9	81.2
DirectQ	4	8	76.69	77.00	63.4	40.6
Q-BERT	4	8	**83.89**	**84.17**	63.4	40.6
DirectQ	3	8	70.27	70.89	53.2	30.5
Q-BERT	3	8	**83.41**	**83.83**	53.2	30.5
Q-BERT$_{\text{MP}}$	2/4 MP	8	**83.51**	**83.55**	53.2	30.5
DirectQ	2	8	53.29	53.32	43.1	20.4
Q-BERT	2	8	**76.56**	**77.02**	43.1	20.4
Q-BERT$_{\text{MP}}$	2/3 MP	8	**81.75**	**82.29**	**46.1**	**23.4**

(c) CoNLL-03

Method	w-bits	e-bits	F_1	Size	Size-w/o-e
Baseline	32	32	95.00	410.9	324.5
Q-BERT	8	8	94.79	102.8	81.2
DirectQ	4	8	89.86	62.2	40.6
Q-BERT	4	8	**94.90**	62.2	40.6
DirectQ	3	8	84.92	52.1	30.5
Q-BERT	3	8	**94.78**	52.1	30.5
Q-BERT$_{\text{MP}}$	2/4 MP	8	**94.55**	52.1	30.5
DirectQ	2	8	54.50	42.0	20.4
Q-BERT	2	8	**91.06**	42.0	20.4
Q-BERT$_{\text{MP}}$	2/3 MP	8	**94.37**	**45.0**	**23.4**

(d) SQuAD

Method	w-bits	e-bits	EM	F_1	Size	Size-w/o-e
Baseline	32	32	81.54	88.69	415.4	324.5
Q-BERT	8	8	81.07	88.47	103.9	81.2
DirectQ	4	8	66.05	77.10	63.4	40.6
Q-BERT	4	8	**80.95**	**88.36**	63.4	40.6
DirectQ	3	8	46.77	59.83	53.2	30.5
Q-BERT	3	8	**79.96**	**87.66**	53.2	30.5
Q-BERT$_{\text{MP}}$	2/4 MP	8	**79.85**	**87.49**	53.2	30.5
DirectQ	2	8	4.77	10.32	43.1	20.4
Q-BERT	2	8	**69.68**	**79.60**	43.1	20.4
Q-BERT$_{\text{MP}}$	2/3 MP	8	**79.25**	**86.95**	**48.1**	**25.4**

4.1. Main Results. We present results of Q-BERT on the development set of the four tasks of SST-2, MNLI, CoNLL-03, and SQuAD, as summarized in Tab. 1. As one can see, Q-BERT performs significantly better compared to the DirectQ method across all four tasks in each bit setting. The gap becomes more obvious for ultra low bit setting. As an example, in 4-bits setting, Direct quantization (DirectQ) of SQuAD results in 11.5% performance degradation as compared to BERT$_{\text{BASE}}$. However, for the same 4-bits setting, Q-BERT only exhibits 0.5% performance degradation. Moreover, under 3-bits setting, the gap between Q-BERT and DirectQ increases even further to 9.68-27.83% for various tasks.

In order to push further the precision setting to lower bits, we investigate the mixed-precision Q-BERT (Q-BERT$_{\text{MP}}$). As can be seen, Q-BERT with uniform 2-bits setting has very poor performance across all four tasks, though the memory is reduced by 20% against 3-bits setting. The reason behind this is the discrepancy that not all the layers have the same sensitivity to quantization as evident from loss landscape visualizations; see Fig. 1 (and Fig. 6 in Appendix). Intuitively, for more sensitive layers, higher bit precision needs to be set, while for layers that are less sensitive, 2-bits setting is already sufficient. To set mixed precision to each encoder layer of BERT$_{\text{BASE}}$, we measure the sensitivity based on Eq. 3.2, which captures both mean and variance of the top eigenvalue of the Hessian shown in Fig. 2. Note that all experiments

in Fig. 2 are based on 10 runs and each run uses 10% of the entire training dataset. We can obverse that for most of the lower encoder layers (layer 1-8), the variance is pretty large compared to the last three layers. We generally observe that the middle part (layer 4-8) has the largest mean(λ_i). Beyond the relatively smaller mean, the last three layers also have much smaller variance, which indicates the insensitivity of these layers. Therefore, higher bits will only be assigned for middle layers according to Eq. 3.2 for Q-BERT 2/3 MP.[3] In this way, with only additional 5MB memory storage, 2/3-bits Q-BERT$_{\text{MP}}$ is able to retain the performance drop within 2.3% for MNLI, SQuAD and 1.1% for SST-2, CoNLL-03, with up to 13$\times$ compression ratio in weights. Note that this is up to 6.8% better than Q-BERT with uniform 2 bits.

One consideration for quantization is that 3-bit quantized execution is typically not supported in hardware. It is however possible to load 3-bit quantized values and cast them to higher bit precision such as 4 or 8 bits in the execution units. This would still have the benefit of reduced memory volume to/from DRAM. It is also possible to avoid using 3 bits and instead use a mixture of 2 and 4 bits as shown in Tab. 1. For example, SST-2 Q-BERT$_{\text{MP}}$ with mixed 2/4-bit precision weights has the same model size as the 3 bit quantization in 53.2MB and achieves similar accuracy. We observe a similar trend for other tasks as well.

One important observation is that we found SQuAD to be harder to quantize as compared to other tasks; see Tab. 1d. For example, 2-bits DirectQ results in more than 10% F_1 score degradation. Even Q-BERT has larger performance drop as compared to other tasks in Tab. 1. We studied this phenomenon further through Hessian analysis. In Fig. 2, among all the tasks, it can be clearly seen that SQuAD not only has much larger eigenvalue variance, but it has very large negative eigenvalues. In fact this shows that the existing BERT model for SQuAD has not reached a local minima. This is further illustrated in the 3-d loss landscape of all four tasks in Fig. 1 and Fig. 3 (and Fig. 6 in Appendix). It can be clearly seen that for the other three tasks, the stopping point is at a quadratic bowl (at least in the first two dominant eigenvalue directions of the Hessian). However, compared to the others, SQuAD has a totally different structure to its loss landscape. As shown in Fig. 3, the stopping points of different layers on SQuAD have negative curvature directions, which means they have not converged to a local minima yet. This could well explain why the quantization of SQuAD results in more accuracy drop. Our initial attempts to address this by changing training hyper-parameters were not successful. We found that the BERT model quickly overfits the training data. However, we emphasize that fixing BERT model training itself is outside the scope of this paper and not possible with academic computational resources.

4.2. Effects of group-wise quantization. We measure the performance gains with different group numbers in Tab. 2. We can observe from the table that performing layer-wise quantization (shown in Fig. 4a) is sub-optimal for all four tasks (the performance drop is around 7% to 11.5%). However, the performance significantly increases as we increase the number of groups. For example, for 12 groups, the performance degradation is less than 2% for all the tasks. Further increasing the group number from 12 to 128 increases the accuracy further by at least 0.3% accuracy. However, increasing the group number further from 128 to 768 can only increase the performance within 0.1%. This shows that the performance gain almost saturates around 128 groups. It is also preferable not to have very large value for the number of group since it increases the number of Look-up Tables (LUTs) necessary for each matrix multiplication. This can adversely affect hardware performance, and based on our results there are diminishing returns in terms of accuracy. In all our experiments, we used 128 groups for both Q-BERT and Q-BERT$_{\text{MP}}$ in Sec. 4.1.

Table 2: Effects of group-wise quantization for Q-BERT on three tasks. The quantization bits were set to be 4 for weights, 8 for embeddings and 8 for activations for all the tasks. From top to down, we increase the number of groups. In order to balance the accuracy and hardware efficiency, we set 128 groups for other experiments.

# Group	SST-2	MNLI-m/mm	CoNLL-03
Baseline	93.00	84.00/84.40	95.00
1	85.67	76.69/77.00	89.86
12	92.31	82.37/82.95	94.42
128	92.66	83.89/84.17	94.90
768 [4]	92.78	84.00/84.20	94.99

5. Discussion. In this Section, we further investigate the quantization effects on different modules, e.g., different embedding layers (e.g., word and position embeddings), and we perform qualitative analysis using attention distribution. This illustrates that Q-BERT better captures the behaviour of the original model as compared to DirectQ in all cases.

[3] Exact detailed bits setting is included in the Appendix C.1

[4] Here we treat each output neuron as a single group.

5.1. Quantization effects on different modules. Here we investigate the quantization effects with respect to different modules of BERT model (multi-head self-attention versus feed-forward network, and different embedding layers, i.e., word embedding versus position embedding).

Generally speaking, we find that embedding layer is more sensitive than weights for quantization. This is illustrated in Tab. 3a, where we use 4-bits layerwise quantization for embedding, which results in an unacceptable performance drop up to 10% for SST-2, MNLI, CoNLL-03 and even more than 20% for SQuAD. This is despite the fact that we used 8/8-bits for weights/activations. On the contrary, encoder layers consume around 79% total parameters ($4\times$ embedding parameter size), while quantizing them to 4-bits in Tab. 1 leads to less performance loss.

Furthermore, we find that position embedding is very sensitive to quantization. For instance, quantizing position embedding to 4 bits results in generally 2% additional performance degradation than quantizing word embedding, even though the position embedding only accounts for less than 5% of the entire embedding. This indicates the importance of positional information in Natural Language Understanding tasks. Given position embedding only accounts for a small portion of model size, we can do mixed-precision quantization for embedding to further push down the model size boundary with a tolerable accuracy drop, as shown in Appendix C.2.

Table 3: Quantization effect to different modules. We abbreviate the quantization bits used for word embedding as "ew-bits", position embedding as "ep-bits", multi-head attention layer as "s-bits" and fully-connected layer as "f-bits". In (a), we set weight and activation bits as 8. In (b), we set embedding and activation bits as 8.

(a) quantization effect on embedding

Method	ew-bits	ep-bits	SST-2	MNLI-m/mm	CoNLL-03	SQuAD
Baseline	32	32	93.00	84.00/84.40	95.00	88.69
Q-BERT	8	8	92.88	83.83/83.91	94.79	88.47
Q-BERT	4	8	91.74	82.91/83.67	94.44	87.55
Q-BERT	8	4	89.11	82.84/82.25	93.86	72.38
Q-BERT	4	4	85.55	78.08/78.96	84.32	61.70

(b) quantization of multi-head attention versus fully-connected layer

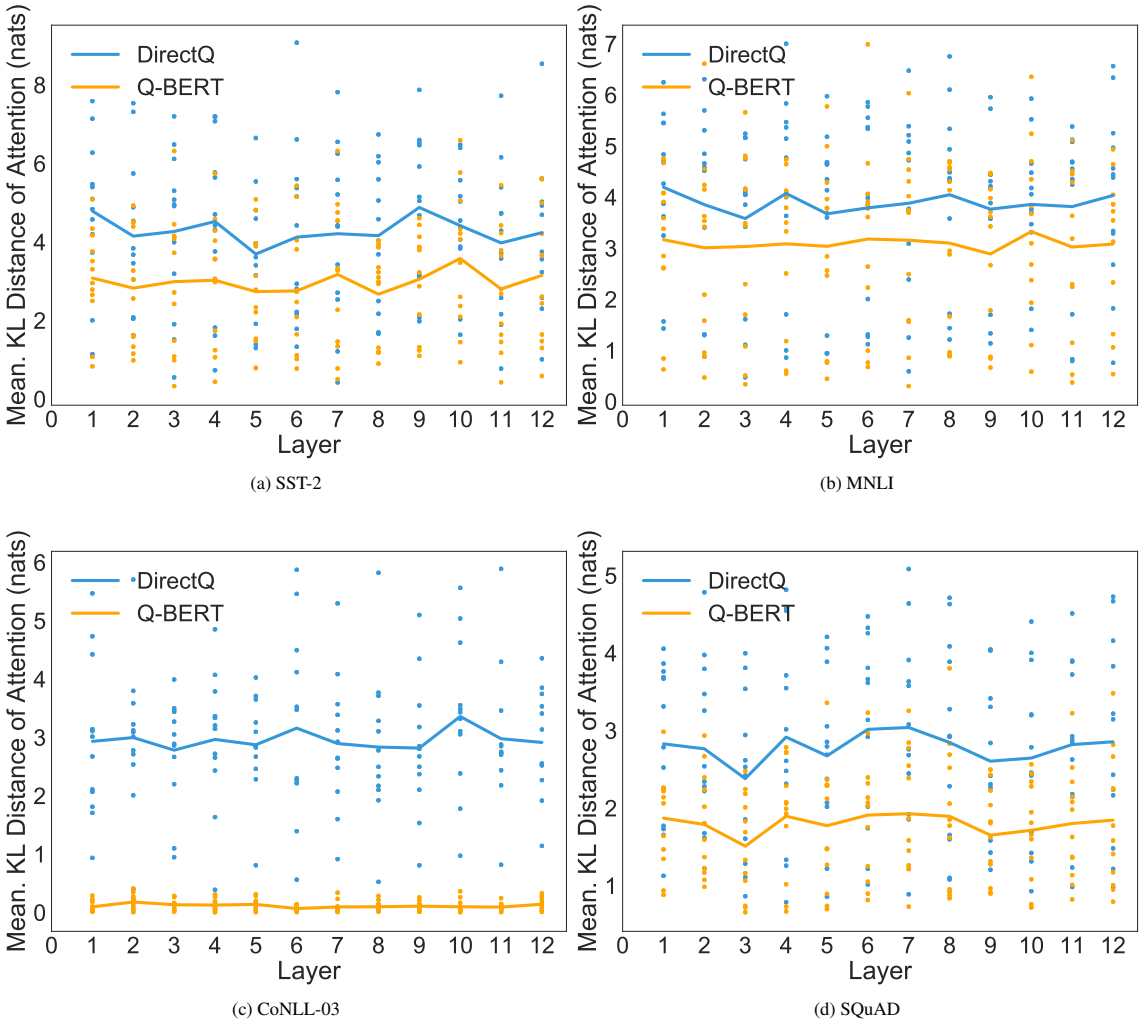
Method	s-bits	f-bits	SST-2	MNLI-m/mm	CoNLL-03	SQuAD
Baseline	32	32	93.00	84.00/84.40	95.00	88.69
Q-BERT$_{\text{MP}}$	1/2$_{\text{MP}}$	2/3$_{\text{MP}}$	89.56	73.66/74.52	91.74	75.81
Q-BERT$_{\text{MP}}$	2/3$_{\text{MP}}$	1/2$_{\text{MP}}$	85.89	70.89/71.17	87.55	68.71
Q-BERT$_{\text{MP}}$	2/3$_{\text{MP}}$	2/3$_{\text{MP}}$	92.08	81.75/82.29	93.91	86.95

To study the quantization effects on self-attention layers and fully-connected networks, we conducted extensive experiments under different bits settings for the encoder layers. The results are shown in Tab. 3b. Specifically, we adopt the Q-BERT$_{\text{MP}}$ setting in Tab. 1, with a mixture of 2 and 3 bits for encoder weights. To test the robustness of the two modules inside each encoder layer, we further reduce one more bit in the corresponding modules and denote the resulting precision setting 1/2$_{\text{MP}}$. From Tab. 3b, we can conclude that generally self-attention layer is more robust to quantization than the fully-connected network, since 1/2$_{\text{MP}}$ self-attention results in about 5% performance drop while 1/2$_{\text{MP}}$ fully-connected will worsen this to 11%.

5.2. Qualitative Analysis. We use attention information to conduct qualitative analysis to analyze the difference between Q-BERT and DirectQ.

To do so, we compute the Kullback–Leibler (KL) divergence between the attention distribution for the same input from the coordinated head of both quantized BERT and full-precision BERT. It should be noted that we compute the average distance out of 10% of the entire training dataset. The smaller KL divergence here means that the output of the multi-head attention of the two models is closer to each other. We illustrate this distance score for each individual head in Fig. 5 for SST-2, MNLI, CoNLL-03 and SQuAD. We compared Q-BERT and DirectQ with 4-bits weights, 8-bits embedding and 8-bits activation. Each scatter point in Fig. 5 denotes the distance w.r.t. one head, and the line chart shows the average results over the 12 heads in one layer. We can clearly see that Q-BERT always incurs a smaller distance to the original baseline model as compared to DirectQ model, for all the different layers.

6. Conclusion. In this work, we perform an extensive analysis of fine-tuned BERT and propose Q-BERT, an effective scheme for quantizing BERT. In order to reduce aggressively the model size by mixed-precision quantization,



(a) SST-2 (b) MNLI

(c) CoNLL-03 (d) SQuAD

Fig. 5: KL divergence over attention distribution between Q-BERT/DirectQ and Baseline. The distance between Q-BERT and Baseline is much smaller than that of DirectQ and Baseline.

we proposed a new layer-wise Hessian based method which captures both the average and the variance of the eigenvalues. Moreover, a new group-wise quantization is proposed to perform fine-grained quantization inside each encoder layer. In four downstream tasks, equipped with the aforementioned methods, Q-BERT achieves $13\times$ compression ratio in weights, $4\times$ smaller activation size, and $4\times$ smaller embedding size, with at most 2.3% accuracy loss. To understand better how different factors will affect the trade-off between performance and the model compression ratio in Q-BERT, we conduct controlled experiments to investigate the effect of different quantization schemes and quantizing different modules in BERT, respectively.

Acknowledgments. We would like to thank Prof. Joseph Gonzalez, Prof. Dan Klein, and Prof. David Patterson for their valuable feedback. This work was supported by a gracious fund from Intel corporation, Berkeley Deep Drive (BDD), and Berkeley AI Research (BAIR) sponsors. We would like to thank the Intel VLAB team for providing us with access to their computing cluster. We also thank gracious support from Google for providing cloud compute. MWM would also like to acknowledge ARO, DARPA, NSF, ONR, and Intel for providing partial support of this work.

References.

[1] J. Ba and R. Caruana. "Do deep nets really need to be deep?" In: *Proceedings of the NIPS*. 2014, pp. 2654–2662.

[2] Y. Bengio, N. Léonard, and A. Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432* (2013).

[3] A. Bhandare, V. Sripathi, D. Karkada, V. Menon, S. Choi, K. Datta, and V. Saletore. "Efficient 8-Bit Quantization of Transformer Neural Machine Language Translation Model". In: *arXiv:1906.00532* (2019).

[4] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan. "Pact: Parameterized clipping activation for quantized neural networks". In: *arXiv:1805.06085* (2018).

[5] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. "What Does BERT Look At? An Analysis of BERT's Attention". In: *arXiv:1906.04341* (2019).

[6] M. Courbariaux, Y. Bengio, and J.-P. David. "Binaryconnect: Training deep neural networks with binary weights during propagations". In: *Proceedings of the NIPS*. 2015, pp. 3123–3131.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the NAACL*. 2019, pp. 4171–4186.

[8] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer. "HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision". In: *arXiv:1905.03696* (2019).

[9] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer. "Squeezenext: Hardware-aware neural network design". In: *Proceedings of the CVPR Workshops*. 2018, pp. 1638–1647.

[10] G. Hinton, O. Vinyals, and J. Dean. "Distilling the knowledge in a neural network". In: *arXiv:1503.02531* (2015).

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv:1704.04861* (2017).

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. "Quantized neural networks: Training neural networks with low precision weights and activations". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6869–6898.

[13] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size". In: *arXiv preprint arXiv:1602.07360* (2016).

[14] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[15] R. Krishnamoorthi. "Quantizing deep convolutional networks for efficient inference: A whitepaper". In: *arXiv:1806.08342* (2018).

[16] Y. LeCun, J. S. Denker, and S. A. Solla. "Optimal brain damage". In: *Proceedings of the NIPS*. 1990, pp. 598–605.

[17] F. Li, B. Zhang, and B. Liu. "Ternary weight networks". In: *arXiv:1605.04711* (2016).

[18] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. "Pruning filters for efficient convnets". In: *arXiv preprint arXiv:1608.08710* (2016).

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv:1907.11692* (2019).

[20] X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, D. Song, and M. Zhou. "A Tensorized Transformer for Language Modeling". In: *arXiv:1906.09777* (2019).

[21] P. Michel, O. Levy, and G. Neubig. "Are Sixteen Heads Really Better than One?" In: *arXiv:1905.10650* (2019).

[22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. "Language Models are Unsupervised Multitask Learners". In: (2019).

[23] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv:1606.05250* (2016).

[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *Proceedings of the ECCV*. Springer. 2016, pp. 525–542.

[25] E. F. Sang and F. De Meulder. "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition". In: *cs/0306050* (2003).

[26] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of EMNLP*. 2013, pp. 1631–1642.

[27] S. Sun, C. Yu, G. Zhe, and L. Jingjing. "Patient Knowledge Distillation for BERT Model Compression". In: *Proceedings of the EMNLP* (2019).

[28] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin. "Distilling Task-Specific Knowledge from BERT into Simple Neural Networks". In: *arXiv:1903.12136* (2019).

[29] Y. Tay, A. Zhang, L. A. Tuan, J. Rao, S. Zhang, S. Wang, J. Fu, and S. C. Hui. "Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks". In: *arXiv:1906.04393* (2019).

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Proceedings of the NIPS*. 2017, pp. 5998–6008.

[31] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. "Glue: A multi-task benchmark and analysis platform for natural language understanding". In: *arXiv:1804.07461* (2018).

[32] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. "HAQ: Hardware-Aware Automated Quantization with Mixed Precision". In: *Proceedings of the CVPR*. 2019, pp. 8612–8620.

[33] P. Wang, X. Xie, L. Deng, G. Li, D. Wang, and Y. Xie. "HitNet: hybrid ternary recurrent neural network". In: *Proceedings of the NIPS*. 2018, pp. 604–614.

[34] A. Williams, N. Nangia, and S. R. Bowman. "A broad-coverage challenge corpus for sentence understanding through inference". In: *arXiv:1704.05426* (2017).

[35] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10734–10742.

[36] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer. "Mixed Precision Quantization of ConvNets via Differentiable Neural Architecture Search". In: *arXiv:1812.00090* (2018).

[37] C. Xu, J. Yao, Z. Lin, W. Ou, Y. Cao, Z. Wang, and H. Zha. "Alternating multi-bit quantization for recurrent neural networks". In: *arXiv:1802.00150* (2018).

[38] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *arXiv:1906.08237* (2019).

[39] Z. Yao, A. Gholami, Q. Lei, K. Keutzer, and M. W. Mahoney. "Hessian-based Analysis of Large Batch Training and Robustness to Adversaries". In: *arXiv:1802.08241* (2018).

[40] D. Zhang, J. Yang, D. Ye, and G. Hua. "LQ-nets: Learned quantization for highly accurate and compact deep neural networks". In: *Proceedings of the ECCV*. 2018, pp. 365–382.

[41] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. "Incremental network quantization: Towards lossless cnns with low-precision weights". In: *arXiv:1702.03044* (2017).

[42] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients". In: *arXiv:1606.06160* (2016).

[43] Y. Zhou, S.-M. Moosavi-Dezfooli, N.-M. Cheung, and P. Frossard. "Adaptive quantization for deep neural network". In: *Proceedings of the AAAI*. 2018.

Algorithm 1: Power Iteration for Eigenvalue Computation

Input: Block Parameter: W_i.
Compute the gradient of W_i by backpropagation, *i.e.*, $g_i = \frac{dL}{dW_i}$.
Draw a random vector v (same dimension as W_i).
Normalize v, $v = \frac{v}{\|v\|_2}$
for $i = 1, 2, \ldots, n$ **do** // Power Iteration
  Compute $gv = g_i^T v$ // Inner product
  Compute Hv by backpropagation, $Hv = \frac{d(gv)}{dW_i}$ // Get Hessian vector product
  Normalize and reset v, $v = \frac{Hv}{\|Hv\|_2}$

Appendix A. Detailed quantization process. In the forward pass, each element in a weight or activation tensor X will be quantized as follows:

$$X' = \text{Clamp}(X, q_0, q_{2^k-1}),$$
$$X^I = \lfloor \frac{X' - q_0}{\Delta} \rceil, \text{ where } \Delta = \frac{q_{2^k-1} - q_0}{2^k - 1},$$
$$Q(X) = \Delta X^I + q_0$$

where $\lfloor \cdot \rceil$ is the round operator, Δ is the distance between adjacent quantized points, X^I is a set of integer indices and so as bias^I, which is omitted for the clarity in following equations. $[q_0, q_{2^k-1}]$ stands for the quantization range of the floating point tensor, and the function Clamp sets all elements smaller than q_0 equal to q_0, and elements larger than q_{2^k-1} to q_{2^k-1}. It should be noted that $[q_0, q_{2^k-1}]$ can be a subinterval of $[min, max]$, in order to get rid of outliers and better represent the majority in a specific tensor. During inference, the expensive floating point arithmetic can be replaced by efficient integer arithmetic for the matrix multiplication with X^I, and then followed by a gathered dequantization operation, which will significantly accelerate the computation process. Since we use the quantization-aware fine-tuning scheme, in the backward pass, the Straight-Though Estimator (STE) [2] is used for computing the gradient for X.

Appendix B. Dataset. We apply Q-BERT on Sentiment Classification, Natural Language Inference, Named Entity Recognition and Machine Reading Comprehension tasks. For Sentiment Classification, we evaluate on Stanford Sentiment Treebank (SST-2) [26]. For Named Entity Recognition, we use CoNLL-2003 English benchmark dataset for NER (CoNLL-03) [25]. For Natural Language Inference, we test on Multi-Genre Natural Language Inference (MNLI) [34]. For Machine Reading Comprehension, we evaluate on the Stanford Question Answering Dataset (SQuAD) [23].

More specifically, SST-2 is a movie review dataset with binary annotations, where the binary label indicates positive and negative reviews. MNLI is a multi-genre NLI task for predicting whether a given premise-hypothesis pair is entailment, contradiction or neural. Its test and development datasets are further divided into in-domain (MNLI-m) and cross-domain (MNLI-mm) splits to evaluate the generality of tested models. CoNLL-03 is a newswire article dataset for predicting the exact span of the annotated four entity types: person, location, organization, and miscellaneous. SQuAD is a task to answer the question by extracting the relevant span from the context, where a paragraph of context and a question is provided for each sample.

Appendix C. Extra results. Here we describe several additional results.

C.1. Ablation Study of Hessian based Mixed Precision Assignment. To demonstrate the effectiveness of our Hessian based Mixed Precision method, we conduct the ablation study here to use the reversed version of 2/3-bit Q-BERT$_{\text{MP}}$ (Q-BERT$_{\text{MP}}$-rev). Specifically, we will assign lower bits to relatively sensitive layers and higher bits vice versa while keeping model size the same. This means the previous layer in 2/3-bit Q-BERT$_{\text{MP}}$ with 2-bit will be assigned 3-bit in Q-BERT$_{\text{MP}}$-rev. [5]

We can obverse that with the same model size, the performance difference between Q-BERT$_{\text{MP}}$-rev and 2-bit Q-BERT is within 2% for MNLI, CoNLL-03, SQuAD and 4% for SST-2, while that of Q-BERT$_{\text{MP}}$ is beyond 5% for MNLI, CoNLL-03, SQuAD and 8% for SST-2. This large discrepancy in the performance illustrates the superiority of leveraging second order Hessian information in mix precision bits assignment.

[5] The bits setting of 2/3-bit Q-BERT$_{\text{MP}}$ and 2/4-bit Q-BERT$_{\text{MP}}$ are included in Tab. 6 and Tab. 7, respectively.

Table 4: Quantization results for reversed Hessian based Mixed Precision setting. We denote the model using reversed bit assignment against Q-BERT$_{\text{MP}}$ as Q-BERT$_{\text{MP}}$-rev.

(a) SST-2

Method	w-bits	e-bits	Acc
Baseline	32	32	93.00
Q-BERT	2	8	84.63
Q-BERT$_{\text{MP}}$-rev	2/3 $_{\text{MP}}$	8	88.42
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**92.08**

(b) MNLI

Method	w-bits	e-bits	Acc-m	Acc-mm
Baseline	32	32	84.00	84.40
Q-BERT	2	8	76.56	77.02
Q-BERT$_{\text{MP}}$-rev	2/3 $_{\text{MP}}$	8	78.91	79.30
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**81.75**	**82.29**

(c) CoNLL-03

Method	w-bits	e-bits	F_1
Baseline	32	32	95.00
Q-BERT	2	8	91.06
Q-BERT$_{\text{MP}}$-rev	2/3 $_{\text{MP}}$	8	92.66
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**93.91**

(d) SQuAD

Method	w-bits	e-bits	EM	F_1
Baseline	32	32	81.54	88.69
Q-BERT	2	8	69.68	79.60
Q-BERT$_{\text{MP}}$-rev	2/3 $_{\text{MP}}$	8	69.71	79.39
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**79.29**	**86.95**

C.2. Mixed Precision Quantization for Embedding. As can be seen from Tab. 1, when 2/3 $_{\text{MP}}$ is used for quantizing the weight parameters, the bottleneck of the model size is bounded by the embedding table size. Also, observed in Tab. 3a, we noticed that word embedding is less sensitive. Therefore, in this section, we further push the embedding table to be 4-bit (word embedding) and 8-bit (position embedding) mixed-precision to reduce the entire model size. Similar to group-wise quantization for weights, in this ultra-low embedding bits setting, we bucket the 768 output neurons in BERT$_{\text{BASE}}$ word and position embedding layer into 128 groups in Tab. 5. We adopt the same setting for weights and activations in Tab. 1, where we employ 128 groups for weights and set 8/8 bits for weight/activation. Note that with around 0.5% performance drop, the embedding table size can be reduced to 11.6MB, which corresponds to around $8\times$ compression ratio in embedding table and $12\times$ compression ratio in total model size.

Table 5: Embedding mixed-precision quantization results for Q-BERT on four tasks. Results are obtained with 128 groups in each encoder layer and **embedding layer**. We abbreviate quantization bits used for weights as "w-bits", embedding as "e-bits", model size in MB as "Size", and model size without embedding layer in MB as "Size-w/o-e". For simplicity and efficacy, all the models except for Baseline are using 8-bits activation. Here "MP" refers to mixed-precision quantization. The mixed-precision embedding here uses 4-bit word embedding and 8-bit position embedding.

(a) SST-2

Method	w-bits	e-bits	Acc	Size	Size-w/o-e
Baseline	32	32	93.00	415.4	324.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	8	**92.55**	53.2	30.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**92.32**	42.0	30.5
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**92.08**	**48.1**	**25.4**
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**91.51**	**36.9**	**25.4**

(b) MNLI

Method	w-bits	e-bits	Acc m	Acc mm	Size	Size w/o-e
Baseline	32	32	84.00	84.40	415.4	324.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	8	**83.51**	**83.55**	53.2	30.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**82.82**	**83.36**	42.0	30.5
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**81.75**	**82.29**	**46.1**	**23.4**
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**81.00**	**81.65**	**34.9**	**23.4**

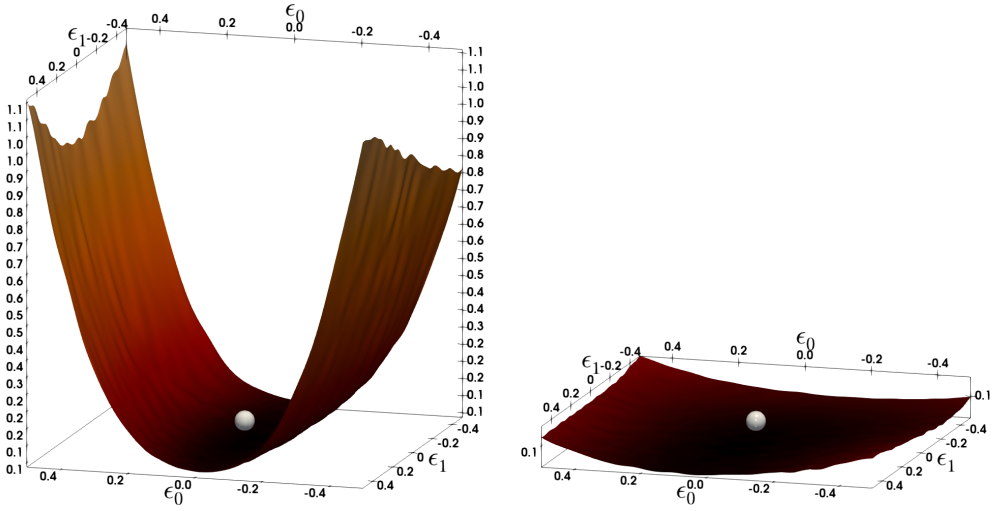
(c) CoNLL-03

Method	w-bits	e-bits	F_1	Size	Size-w/o-e
Baseline	32	32	95.00	410.9	324.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	8	**94.55**	52.1	30.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**94.55**	41.5	30.5
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**94.37**	**45.0**	**23.4**
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**94.45**	**34.4**	**23.4**

(d) SQuAD

Method	w-bits	e-bits	EM	F_1	Size	Size-w/o-e
Baseline	32	32	81.54	88.69	415.4	324.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	8	**79.85**	**87.49**	53.2	30.5
Q-BERT$_{\text{MP}}$	2/4 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**79.53**	**87.14**	42.0	30.5
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	8	**79.25**	**86.95**	**48.1**	**25.4**
Q-BERT$_{\text{MP}}$	2/3 $_{\text{MP}}$	4/8 $_{\text{MP}}$	**78.68**	**86.49**	**36.9**	**25.4**

C.3. Detailed loss landscape for SST-2. We include the detailed loss landscape analysis for the remaining task SST-2 as shown in Fig. 6.



(a) SST-2 3th layer (b) SST-2 10th layer

Fig. 6: The loss landscape for different layers in SST-2 is illustrated by perturbing the parameters along the first two dominant eigenvectors of the Hessian. The silver sphere shows the point in the parameter space that the BERT model has converged to.

Table 6: Bits setting for 2/3-bit Q-BERT$_{MP}$ in all four tasks.

Layer(s)	Layer Type	Parameter Size(M)	Weight bit (SST-2)	Weight bit (MNLI)	Weight bit (CoNLL-03)	Weight bit (SQuAD)
Layer 0	Embedding	23.8	8	8	8	8
Layer 1	Transformer	7.1	2	2	2	2
Layer 2	Transformer	7.1	2	2	3	2
Layer 3	Transformer	7.1	2	2	3	2
Layer 4	Transformer	7.1	3	2	3	3
Layer 5	Transformer	7.1	3	3	3	3
Layer 6	Transformer	7.1	3	3	2	3
Layer 7	Transformer	7.1	3	3	2	3
Layer 8	Transformer	7.1	3	3	2	3
Layer 9	Transformer	7.1	3	2	2	3
Layer 10	Transformer	7.1	2	2	2	2
Layer 11	Transformer	7.1	2	2	2	2
Layer 12	Transformer	7.1	2	2	2	2
Layer 13	FC	0.01	32	32	32	32

Table 7: Bits setting for 2/4-bit Q-BERT$_{MP}$ in all four tasks.

Layer(s)	Layer Type	Parameter Size(M)	Weight bit (SST-2)	Weight bit (MNLI)	Weight bit (CoNLL-03)	Weight bit (SQuAD)
Layer 0	Embedding	23.8	8	8	8	8
Layer 1	Transformer	7.1	2	2	2	2
Layer 2	Transformer	7.1	2	2	2	2
Layer 3	Transformer	7.1	4	2	2	2
Layer 4	Transformer	7.1	4	4	4	4
Layer 5	Transformer	7.1	4	4	4	4
Layer 6	Transformer	7.1	2	4	4	4
Layer 7	Transformer	7.1	4	4	4	4
Layer 8	Transformer	7.1	4	4	4	4
Layer 9	Transformer	7.1	4	2	2	2
Layer 10	Transformer	7.1	2	2	2	2
Layer 11	Transformer	7.1	2	2	2	2
Layer 12	Transformer	7.1	2	2	2	2
Layer 13	FC	0.01	32	32	32	32