

Dynamic Programming Assisted Quantization Approaches for Compressing Normal and Robust DNN Models

Dingcheng Yang¹, Wenjian Yu^{1*}, Haoyuan Mu¹, Gary Yao²

¹Dept. Computer Science & Tech., BNRist, Tsinghua Univ., Beijing, China.

²Case Western Reserve University, USA.

Email: ydc19@mails.tsinghua.edu.cn, yu-wj@tsinghua.edu.cn, muhy17@mails.tsinghua.edu.cn, gxy76@case.edu

ABSTRACT

In this work, we present effective quantization approaches for compressing the deep neural networks (DNNs). A key ingredient is a novel dynamic programming (DP) based algorithm to obtain the optimal solution of scalar K-means clustering. Based on the approaches with regularization and quantization function, two weight quantization approaches called DPR and DPQ for compressing normal DNNs are proposed respectively. Experiments show that they produce models with higher inference accuracy than recently proposed counterparts while achieving same or larger compression. They are also extended for compressing robust DNNs, and the relevant experiments show 16X compression of the robust ResNet-18 model with less than 3% accuracy drop on both natural and adversarial examples.

KEYWORDS

Dynamic Programming, Neural Network Compression, Quantization, Robust Model, Weight Sharing.

ACM Reference Format:

Dingcheng Yang¹, Wenjian Yu^{1*}, Haoyuan Mu¹, Gary Yao², ¹Dept. Computer Science & Tech., BNRist, Tsinghua Univ., Beijing, China., ²Case Western Reserve University, USA., Email: ydc19@mails.tsinghua.edu.cn, yu-wj@tsinghua.edu.cn, muhy17@mails.tsinghua.edu.cn, gxy76@case.edu, . 2021. Dynamic Programming Assisted Quantization Approaches for Compressing Normal and Robust DNN Models. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21), January 18–21, 2021, Tokyo, Japan*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3394885.3431538>

1 INTRODUCTION

Deep neural networks (DNNs) have been demonstrated to be successful on many tasks. However, the size of DNN model has continuously increased while it achieves better performance. As a result, the storage space of DNN becomes a major concern if we deploy it on resource-constrained devices, especially in the edge-computing and AI-of-things applications

In recent years, there are a lot of work on compressing DNN models. The proposed techniques consist of pruning [12, 17], knowledge distillation [14], quantization [2–4, 8, 11, 28], low-rank approximation [5, 7, 21, 25], etc. Among them, quantization based methods represent the network weights with very low precision, thus yielding highly compact DNN models compared to their floating-point counterparts. Weight sharing [2, 3, 8, 11] is a kind of quantization method, which applies clustering on the weights, so as to achieve compression by only recording cluster centers and weight assignment indexes. Other quantization methods can be regarded as variants of scalar weight sharing, which restrict the weights to floating-point numbers satisfying certain constraints [4, 28]. The parameter space with them is a subspace of the parameter space for the DNN applying the weight sharing with same bit length. Therefore, the weight sharing approach could provide better performance of compression, while the other quantization schemes may be more friendly to inference acceleration.

DNNs are vulnerable to adversarial examples, which can be crafted by adding visually imperceptible perturbations on images. Several approaches for training robust DNN models were recently proposed [9, 29], to defense the adversarial examples. However, there is few work devoted to the compression of robust DNN model, and most of them only employ the pruning and/or simple quantization technique [10, 15].

In various quantization approaches, the K-means clustering problem is often involved. It is always solved with the Lloyd's algorithm [20] in existing work, resulting in a solution which is non-optimal, and sensitive to the initial guess as revealed by experiments in [11]. In this work, we focus on the compression of DNN models. We consider the scalar K-means clustering without using the Lloyd's algorithm, and explore better weight quantization approaches for the compression of normal and robust DNNs. One of our key contributions is a dynamic programming (DP) based algorithm producing the optimal solution of scalar clustering problem. It has $O(N^2K)$ time complexity, where N and K are the numbers of scalars and clusters respectively. The algorithm is collaborated with the weight quantization approaches to improve the compression of normal and robust DNN models.

The major contributions of this work are as follows.

- A dynamic programming (DP) based algorithm is proposed to obtain the optimal solution of the K-means clustering problem with scalar data.
- A DP assisted approach with regularization (called DPR) and a DP assisted approach with quantization function (called DPQ) are proposed to compress normal DNN models. DPR trains a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-7999-1/21/01...\$15.00

<https://doi.org/10.1145/3394885.3431538>

clustering-friendly network and then compresses it with weight clustering. DPQ trains the network with a formulation including quantization function and employs the DP based algorithm to obtain better clustering. They are then extended to DPR⁺ and DPQ⁺ approaches respectively, through collaborating with the state-of-the-art TRADES [29], for compressing robust DNNs.

- Experiments on normal DNNs have shown the advantages of DPR and DPQ over other compression approaches like Deep K-Means [24] and LQ-Net [28]. For GoogLeNet on ImageNet dataset, DPQ results in a model which is 2.5X smaller and with 1% higher inference accuracy than that produced by Deep K-means. For ResNet-18, the models obtained with DPQ show 0.5% higher accuracy than those by LQ-Net with same compression ratio. Besides, up to 77X compression of Wide ResNet is achieved (with < 3% accuracy drop) by a combinatorial scheme including DPR, the pruning and Huffman coding techniques.
- Experiments on robust DNNs have validated the effectiveness of the proposed DPR⁺ and DPQ⁺ approaches. With 2-bit quantization, DPR⁺ produces a compressed robust ResNet-18 model which exhibits less than 3% accuracy drop on both natural and adversarial examples.

2 BACKGROUND

2.1 Weight Sharing and Quantization

The scalar weight sharing introduced by [11] is the first quantization approach. Regard the weights of DNN as a set of vectors $W = \{W_1, \dots, W_m\}$, where $W_i \in \mathbb{R}^{n_i}$. Accordingly, the result of scalar weight sharing can be expressed as $C = [C_1, \dots, C_m] \in \mathbb{R}^{K \times m}$, where K is the number of clusters and vector C_i contains the K cluster centers. The uncompressed DNN needs $32 \sum_{i=1}^m n_i$ bits for storing the weights, as each weight is expressed as a 32-bit floating-point number. With the weight sharing, each element of W_i is represented by an element in $C_i \in \mathbb{R}^K$. So, we just need $\log_2 K \cdot \sum_{i=1}^m n_i$ bits to encode the index and $32mK$ bits to store the cluster centers. This scalar clustering leads to the compression ratio:

$$r = \frac{32 \sum_{i=1}^m n_i}{\log_2 K \cdot \sum_{i=1}^m n_i + 32mK}, \quad (1)$$

which approximates $32/\log_2 K$.

A naive approach for weight sharing regards the problem as the K-means clustering of the trained weights. However, the weights often follow the Gaussian distribution, which is consistent with the phenomenon observed by [3] that the weights of learned convolutional filters are typically smooth. This is shown in Fig. 1(a) as an example, and means the learned weights may be unsuitable for clustering or quantization. Thus, the network has to be retrained to compensate for the accuracy loss caused by weight sharing [11]. Because for each weight, the cluster it belongs to is fixed during the retraining, this is the same as what's done for HashedNet [2], i.e randomly grouping each weight and then training, and could not produce a good DNN model [19].

Training DNN and performing weight sharing can be formulated as a single optimization problem. If the loss function of training

DNN is $L(W)$, the optimization problem can be expressed as:

$$\begin{aligned} & \min_{W, C} L(W), \\ & \text{s. t. } C = [C_1, C_2, \dots, C_m] \in \mathbb{R}^{K \times m}, \\ & W = \{W_1, W_2, \dots, W_m\}, \text{ and } \forall i, j, W_{i,j} \in C_i. \end{aligned} \quad (2)$$

The constraint means every element in vector W_i appears in the vector C_i containing cluster centers. This formulation is also suitable for a general quantization method. For a non-weight-sharing approach (e.g. low-bit representation), a problem with more constraints added to (2) is solved.

If the weights are regarded as vectors for clustering, it becomes the vector weight sharing problem, which was investigated in [24]. Vector weight sharing produces larger compression ratio than (1) (even > 32), but it may induce sacrifice on accuracy as compared with the scalar weight sharing. A non-weight-sharing quantization approach can be regarded as a variant of scalar weight sharing. It trades off smaller parameter space for faster inference computation.

There are mainly two kinds of approaches for the quantization problem. One converts (2) to a formulation with regularization item. The other takes the constraints into account with quantization function. They are briefly introduced in the following two subsections.

2.2 The Approach with Regularization

The optimization problem (2) is difficult to solve. One can convert the constraints to a regularization item in the loss function. Then, solving the new formulation results in a clustering-friendly model.

For simplicity, we just consider the problem with $m = 1$ where the weight vector $W' \in \mathbb{R}^n$ is clustered to K centers c_1, c_2, \dots, c_K . For (2), the constraints can be converted to a regularization item $\lambda \sum_{i=1}^n \min_{1 \leq k \leq K} (W'_i - c_k)^2$ in the loss function, where λ is a Lagrange multiplier. The regularization item corresponds to the loss of K-means clustering problem, which is generally NP-hard for vector data. An approximate algorithm for K-means clustering was proposed in [27], which relaxed the problem to minimizing $Tr(W'^T W') - Tr(F^T W'^T W' F)$ with constraint $F^T F = I$. Here, Tr denotes the matrix trace, $F \in \mathbb{R}^{n \times K}$, and I is the identity matrix. A singular value decomposition (SVD) based algorithm was proposed in [27] to obtain the closed-form solution of the relaxed problem. Based on this, an approach called Deep K-means was proposed for the weight-sharing compression of DNN, which solves [24]:

$$\begin{aligned} & \min_{W', F} \{L(W') + \lambda [Tr(W'^T W') - Tr(F^T W'^T W' F)]\}, \\ & \text{s. t. } F \in \mathbb{R}^{n \times K}, F^T F = I. \end{aligned} \quad (3)$$

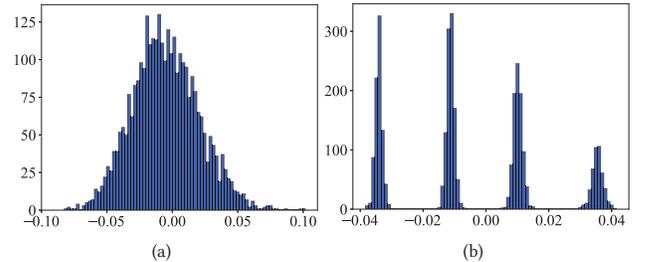


Figure 1: The histograms of weights in the FreshNet model's first convolution layer obtained from (a) a normal training, and (b) a clustering-friendly training.

An iterative procedure was proposed in [24] to solve (3), which updates W' and F alternatively. W' is updated with the stochastic gradient descent (SGD) approach at each iteration, while F is updated by computing K -truncated SVD after training every t epochs.

2.3 The Approach with Quantization Function

The quantization function is also used to model the effect of clustering in (2). Suppose we have a quantization function Q_i for each W_i and C_i , where $Q_i(x) = \operatorname{argmin}_{c \in C_i} |x - c|$. Then, the quantization functions can be plugged into the neural network directly. The optimization problem for quantization (2) is converted to:

$$\begin{aligned} \min_{W, C} L(Q_1(W_1), Q_2(W_2), \dots, Q_m(W_m)), \quad (4) \\ \text{s. t. } C = [C_1, C_2, \dots, C_m] \in \mathbb{R}^{K \times m}, \\ \forall i, j, Q_i(W_{i,j}) = \operatorname{argmin}_{c \in C_i} |W_{i,j} - c|. \end{aligned}$$

Once the cluster centers C is given, W can be optimized by SGD. However, the $\frac{\partial Q_i(W_{i,j})}{\partial W_{i,j}}$ is zero almost everywhere, which makes $\frac{\partial L}{\partial W_{i,j}} = 0$ and training neural network is infeasible. A common solution to this is a so-called straight-through estimator (STE) technique [4], which approximates $\frac{\partial L}{\partial W_{i,j}}$ with $\frac{\partial L}{\partial Q_i(W_{i,j})}$.

The remaining problem is how to choose C . Let L be the bit length for quantization, i.e. $K = 2^L$. With LQ-Net [28], which is an effective DNN quantization scheme with quantization function, a quantizer basis $v_i \in \mathbb{R}^L$ and an encoding matrix $B_i \in \{-1, 1\}^{n_i \times L}$ for each vector W_i are found at each iteration to minimize $\|B_i v_i - W_i\|_2^2$. Then, for each quantization function Q_i vector $C_i = B_i^* v_i$, where matrix $B_i^* \in \mathbb{R}^{K \times L}$ contains all vectors from set $\{-1, 1\}^L$.

3 DYNAMIC PROGRAMMING ASSISTED APPROACHES FOR DNN COMPRESSION

In this section, we first propose the dynamic programming based algorithm for the scalar clustering problem. Then, we present its applications to improve the aforementioned two approaches for DNN quantization.

3.1 A DP Based Algorithm for Scalar Clustering

As we know, the K -means clustering problem is NP-hard for general vector data. Therefore, the global optimum for the vector quantization cannot be found in reasonable time. Nevertheless, we find out that the optimal solution of the scalar K -means clustering can be obtained in polynomial time, based on the following Theorem 1.

THEOREM 1. *Let $x_1 \leq x_2 \leq \dots \leq x_N$ be N scalars which need to be clustered into K classes. The clustering result is expressed as an integer index set $p = \{p_1, p_2, \dots, p_N\}$, $1 \leq p_i \leq K$, which means x_i belongs to the p_i -th cluster. If the K -means clustering is to minimize*

$$g(p, c) = \sum_{i=1}^N (x_i - c_{p_i})^2, \quad (5)$$

$$\text{s. t. } c = \{c_1, c_2, \dots, c_K\}, c_1 < c_2 < \dots < c_K,$$

an optimal solution p satisfies: $1 = p_1 \leq p_2 \leq \dots \leq p_N = K$.

PROOF. Suppose the ascending array $c = \{c_i\}$ stands for the cluster centers for the optimal solution. Let $c'_0 = -\infty$, $c'_1 = (c_1 + c_2)/2$, $c'_2 = (c_2 + c_3)/2$, \dots , $c'_{K-1} = (c_{K-1} + c_K)/2$, $c'_K = \infty$. We can construct a clustering solution by setting $p_i = j$, for any x_i satisfying $c'_{j-1} \leq$

$x_i < c'_j$ (meaning c_j is the closest cluster center for x_i). Therefore, this solution minimizes $g(p, c)$ and is an optimal solution satisfying $1 = p_1 \leq p_2 \leq \dots \leq p_N = K$. \square

Theorem 1 infers that by suitable interval partition we can get the optimal K -means clustering of the weights (see Fig. 2). Let $x_1 \leq \dots \leq x_N$ be the sorted weights for quantization, and $G_{n,k}$ be the minimum loss for clustering the first n weights into k clusters. Based on Theorem 1, we have

$$G_{n,k} = \min_{k-1 \leq i < n} \{G_{i,k-1} + h(i+1, n)\}, \quad 1 < k \leq K, k \leq n \leq N, \quad (6)$$

where $h(l, q) = \min_c \sum_{i=l}^q (x_i - c)^2$, meaning the minimum clustering error (loss) for clustering x_l, x_{l+1}, \dots, x_q to one class. For the trivial situation with $k = 1$, $G_{n,1} = h(1, n)$. Now, we consider how to obtain the optimal clustering corresponding to $G_{n,k}$ for the situations with $k > 1$. Following (6), we need to enumerate all possible i which represents the largest index of scalar not belonging to the k -th cluster. The minimum loss for clustering (i.e. quantization error) includes two parts: the minimum loss for clustering x_1, \dots, x_i into $k-1$ clusters, i.e. $G_{i,k-1}$, and the minimum quantization error that quantizing x_{i+1}, \dots, x_n into a single value, i.e. $h(i+1, n)$. The latter part can be easily calculated, and the mean of scalars should be the cluster center (quantized value). So,

$$\begin{aligned} h(l, q) &= \sum_{i=l}^q (x_i - \frac{1}{q-l+1} \sum_{j=l}^q x_j)^2 \quad (7) \\ &= \sum_{i=l}^q x_i^2 - \frac{1}{q-l+1} (\sum_{i=l}^q x_i)^2. \end{aligned}$$

For clustering, we need to know the optimal solution $\{c_i\}$, instead of the clustering error. We can use an auxiliary array z to depict the optimal clustering obtained by solving (6):

$$z_{n,k} = \operatorname{argmin}_{k-1 \leq i < n} \{G_{i,k-1} + h(i+1, n)\} + 1. \quad (8)$$

$z_{n,k}$ is the index of the first scalar in the last class, when the first n scalars in $\{x_i\}$ are clustered into k classes optimally. During the recursive procedure of solving (6) we can get the $z_{n,k}$ values. And, for example, with $z_{N,K}$ the last cluster center c_K can be obtained.

The core idea of dynamic programming is breaking a complicated problem down into simpler sub-problems in a recursive manner [6]. From the above discussion, we see that (6) reflects the optimal substructure for solving the scalar K -means clustering problem, and (8) guides us to find an optimal solution. Along with (7) and other dynamic programming skills, we derive Algorithm 1 for optimally solving the scalar quantization problem.

Based on Theorem 1, the related derivation (6) through (8), and the principle of dynamic programming, we can prove the optimality of the DP based algorithm for scalar quantization, i.e. Theorem 2.

THEOREM 2. *The DP based algorithm for scalar quantization (Algorithm 1) obtains the optimal solution for the K -means clustering problem of scalar data (i.e., the minimization of (5)).*

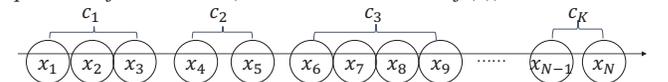


Figure 2: Illustration of the optimal solution of clustering N scalars into K classes.

Algorithm 1 DP based scalar quantization

Input: N scalars $x_1 \leq x_2 \cdots \leq x_N$, number of clusters K .

Output: The K cluster centers in the optimal solution.

```

1: Define two  $N \times K$  arrays  $G$  and  $z$ .
2: for  $i \leftarrow 1$  to  $N$  do
3:   Pre-compute  $h(j, i)$  for  $1 \leq j \leq i$  based on (7).
4:    $G_{i,1} \leftarrow h(1, i)$ ,  $z_{i,1} \leftarrow 1$ .
5:   Calculate  $G_{i,k}$  and  $z_{i,k}$  based on (6) and (8) for  $1 < k \leq K$ .
6: end for
7:  $n \leftarrow N$ 
8: for  $i \leftarrow K$  downto 1 do
9:    $c_i \leftarrow (\sum_{j=z_{n,i}}^n x_j) / (n - z_{n,i})$ .
10:   $n \leftarrow z_{n,i} - 1$ .
11: end for
12: return  $c_1, c_2, \dots, c_K$ .
```

For Step 3 of Alg. 1, we just need enumerate j from i to 1, and use two variables to store $\sum_{k=j}^i x_k^2$ and $\sum_{k=j}^i x_k$. Then, with (7) we can obtain $h(j, i)$ for all $j < i$, with a time complexity of $O(i)$ for a given i . The time complexity of Step 5 is $O(iK)$ for a given i , since (6) needs $O(i)$ time for calculating each $G_{i,k}$ and $z_{i,k}$ with the pre-computed $h(j, i)$. This derives that the time complexity of Alg. 1 is $O(N^2K)$, where N is the numbers of scalars. We will make sure that N is not very large to save computation, when applying it to the weight quantization. It should be pointed out that, Alg. 1 can be easily extended to other kinds of clustering problem, such as that if the L_2 norm in the loss function (5) is replaced with L_1 norm.

3.2 DP Assisted Approach with Regularization

Inspired by Deep K-means [24], we propose a DP assisted approach with regularization (called DPR) for DNN compression, which directly optimizes the Lagrangian function of (2) during the training process with the help of the proposed DP based algorithm. The problem is an unconstrained optimization:

$$\min_{W, C} \{L(W) + \lambda \sum_{i=1}^m \sum_{j=1}^{n_i} \min_{1 \leq k \leq K} (W_{i,j} - C_{i,k})^2\}, \quad (9)$$

where λ is the Lagrange multiplier. After solving it, we can obtain a clustering-friendly network (an example is shown in Fig. 1(b)).

The problem is solved through alternatively optimizing W and C . After every t epochs of SGD based optimization of W , we optimize C by solving a scalar K-means clustering with the DP based algorithm (Alg. 1). In practice, for an fully-connected (FC) layer with $n_{fc} \times m_{fc}$ weights, we divide them into n_{fc} parts (each contains $n = m_{fc}$ weights). We cluster the weights row by row. For a convolutional layer with $n_{conv} \times m_{conv} \times h_{conv} \times w_{conv}$ weights, we divide the weights into n_{conv} parts and each part contains $n = m_{conv} \times h_{conv} \times w_{conv}$ weights. In this way, the number of weights on which we do clustering will be no more than 10000 for most mainstream DNNs. So, the computational time for the DP based algorithm is affordable. This fine-grained scalar clustering improves the accuracy if compared with clustering all weights as a whole, and just induces negligible drop of compression ratio. Compared with Deep K-means [24], we do not relax the original optimization problem and thus may achieve better accuracy.

The DPR is also applicable to the robust DNN models. We just need to add the clustering error, i.e. the minimum loss $G_{N,K}$, as a regularization term to the loss function optimized during the training process. We consider the state-of-the-art TRADES (TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization) model for robust DNN [29]. The new formulation for training becomes:

$$\min_{W, C} \{L(f(X; W), Y) + \max_{X' \in \mathbb{B}(X, \epsilon)} \gamma L(f(X; W), f(X'; W)) + \lambda \sum_{i=1}^m \sum_{j=1}^{n_i} \min_{1 \leq k \leq K} (W_{i,j} - C_{i,k})^2\}, \quad (10)$$

where $f(X; W)$ is the output vector of learning model given parameters W , L denotes the cross-entropy loss function, $\mathbb{B}(x, \epsilon)$ represents a neighborhood of x : $\{x' : \|x' - x\|_2 \leq \epsilon\}$, and γ is a regularization parameter to trade off between accuracy and robustness. With this formulation, we can train a clustering-friendly robust model.

3.3 DP Assisted Approach with Quantization Function

Based on the formulation (4), we propose a DP assisted approach with quantization function (called DPQ). The core idea is performing DP based algorithm to find an accurate quantizer, i.e. obtaining the optimal C_i to minimize $\|Q_i(W_i) - W_i\|_2^2$. This is the standard scalar K-means clustering problem. Then, we solve (4) directly.

We do not consider the constraints for quantizer in LQ-Net [28], and thus search in a larger parameter space. During the solution of (4), we do not perform the proposed DP based algorithm at each iteration to save computation. Instead, it is executed every t epochs while for other iteration we perform the Lloyd's algorithm for clustering. Thanks to the small perturbation of weights incurred by SGD, the cluster centers C in last iteration is a good initial guess of Lloyd's algorithm. The fine-grained scalar clustering strategy in last subsection is also used to improve the runtime efficiency.

This approach is also applicable to the robust DNN model. We just need to plug the quantization function into the loss function for robust model (such as TRADES) to derive the optimization problem:

$$\min_{W, C} \{L(f(X; Q(W)), Y) + \max_{X' \in \mathbb{B}(X, \epsilon)} \gamma L(f(X; Q(W)), f(X'; Q(W)))\}. \quad (11)$$

And, the STE technique is used during the training, for either the normal model or robust model.

4 EXPERIMENTAL RESULTS

In this section, we present the experimental results to demonstrate the effectiveness of the proposed DPR and DPQ approaches for DNN compression. We first conduct experiments with CIFAR-10 dataset [16] for compressing the normal DNN models in [7], [3], and the Wide ResNet [26]. The results show the benefit of using the proposed DP based algorithm to replace the Lloyd's algorithm, and the advantage of the proposed approaches over the counterparts. Then, the experiments are carried out with the ImageNet dataset, for compressing GoogLeNet [23] and ResNet-18 [13]. Lastly, the experiments on compressing robust DNN models are presented.

The proposed approaches are implemented with Python 3.6. The DPR approach has two hyperparameters: λ , the regularization factor in (9); t , the clustering frequency during training. We choose

$\lambda = 100$ for all experiments. The value of t varies for different datasets because it affects the number of training epochs to reach convergence. The DPQ approach has only one hyperparameter t , which is set to 5 for all experiments. The training and inference are conducted on PyTorch. In all experiments, the compression ratio (CR) of the proposed approaches is obtained with (1) and then rounded to an integer.

4.1 Compressing Normal Models on CIFAR-10

The TT-Conv model [7] contains six convolutional layers and one fully-connected (FC) layer. The authors of TT-Conv used tensor train decomposition to compress the convolutional layer by 4X which makes the inference accuracy decreases by 2%. In [3], a FreshNet approach is proposed to quantize the weights of a DNN model’s weights on the frequency domain, based on the observation that the learned convolutional weights are smooth and low-frequency. It achieves a CR of 16, with a 6.51% drop of inference accuracy. In [24], Deep K-means is used to compress the TT-Conv model and FreshNet model, which achieves less accuracy loss with same compression ratio.

We first test the proposed DPR and DPQ with these two models. We choose $t = 20$ for DPR and use SGD with cosine annealing for training. The learning rate reduces from 0.05 to 0.01 during the training. The inference accuracy of the pretrained models we obtained for TT-Conv and FreshNet model are 91.45% and 87.51%, respectively. They are higher than those reported in [24]. For the TT-Conv model, we train 300 epochs. Because the FreshNet model is prone to overfitting [3], we just train 150 epochs and obtain a better result than training 300 epochs. The number of training epochs used for our DP assisted approaches is consistent with the pretrained model. We also use a variant of DPR called LR as a baseline, which employs Lloyd’s algorithm to do clustering during training. The other details of LR are consistent with DPR. The experimental results of DPR, DPQ, and the baselines are listed in Table 1.

The results in Table 1 demonstrate two important phenomena. Firstly, the inference accuracy of the model compressed by DPR is always higher than that by LR, which means that the proposed DP based algorithm is helpful to improve the accuracy of compressed model. Secondly, the models compressed with our DPR performs the best, and even can exhibit better accuracy than the pretrained uncompressed model. The possible reason is that the constraint of quantization may suppress overfitting, like what a L_2 -norm regularized factor often behaves. Therefore, we believe that performing a suitable weight quantization cannot only reduces the size of model, but also improve the performance of DNN.

Table 1: The results of compressing the models in [7] and [3]. Δ means the change of inference accuracy compared to the pretrained model.

TT-Conv model [7]			FreshNet model [3]		
Approach	CR	Δ (%)	Approach	CR	Δ (%)
TT Decomposition [7]	4	-2.00	Hashed Net [2]	16	-9.79
Deep K-means [24]	2	+0.05	FreshNet [3]	16	-6.51
Deep K-means [24]	4	-0.04	Deep K-means [24]	16	-1.30
LR (3 bits)	10	-0.87	LR (2 bits)	16	-0.76
DPR (3 bits)	10	+0.31	DPR (2 bits)	16	-0.57
DPQ (3 bits)	10	+0.22	DPQ (2 bits)	16	-1.56

Compared with Deep K-means [24], the model compressed by DPR exhibits much higher accuracy with same or larger CR. For the proposed DPQ, it performs the same as DPR for the TT-Conv model, but worse for the FreshNet model. This is possibly caused by the network architecture of FreshNet and the small size of dataset. FreshNet has very simple structure, which makes the overfitting easy to happen. We will show the experiments with more popular networks (GoogLeNet and ResNet-18) and the larger dataset (ImageNet) in the following subsections.

An additional experiment is carried out, following that was done in [24]. We combine the proposed approach with the pruning and Huffman coding to compress the Wide ResNet [26] on CIFAR-10. The sparsity for each layer and the pruning method are the same as those in [24], where a total CR=47 is achieved with 2.23% drop of inference accuracy and CR=50 is achieved with 4.49% drop of accuracy. We use DPR to replace Deep K-means for 2-bit quantization, resulting in CR of 77 with only 2.94% accuracy drop.

4.2 Compressing Normal Models on ImageNet

The proposed approaches are first tested on GoogLeNet trained on the ImageNet ILSVRC2012 dataset. We use the PyTorch official model as a pretrained model, whose top-1 accuracy is 69.78% and top-5 accuracy is 89.53%. We quantize the all convolutional layer and fully-connected layer. Our experiment settings are consistent with the PyTorch example¹ of ImageNet except the number of training epochs. We just train 30 epochs and make the learning rate decay 10X at the 20th and 25th epoch respectively, because the pretrained model provides a good initial solution. The learning rate is initialized at 0.001. We choose $t = 3$ for DPR because three epochs are enough for SGD to converge when the cluster centers are fixed. The results are listed in Table 2. It shows that the proposed approaches result in better accuracy than Deep K-means with 2.5X larger compression as well. Similar to those for TT-Conv, the models compressed by our DPR and DPQ even exhibit better inference accuracy than the pretrained uncompressed model. While comparing DPR and DPQ, we see that the latter performs remarkably better than the former.

Table 2: The results of compressing GoogLeNet. Δ -top1 and Δ -top5 are the changes of top1-accuracy and top5-accuracy compared to the pretrained model, respectively.

Approach	CR	Δ -top1 (%)	Δ -top5 (%)
Deep K-means [24]	4	-1.95	-1.14
DPR (3 bits)	10	-1.56	-0.88
DPR (4 bits)	7	+0.30	+0.20
DPQ (3 bits)	10	+0.04	+0.05
DPQ (4 bits)	7	+1.85	+1.02

Then, we conduct the experiment of compressing ResNet-18 with ImageNet dataset. The proposed approaches are compared with LQ-Net and other approaches, whose results are obtained from [28]. For fairness, we quantize all the convolutional layer and fully-connected layer except the first and last layers and do not employ a pretrained model, the same as in [28]. The results are listed in Table 3. From it we see that DPQ surpasses the others when compressing ResNet-18 to more than 2 bits. For 2-bit quantization of ResNet-18, the accuracy of the model compressed with DPQ is comparable to

¹<https://github.com/pytorch/examples/tree/master/imagenet>

Table 3: The results of compressing ResNet-18. acc-top1 and acc-top5 are the top1-accuracy and top5-accuracy on ImageNet, respectively.

Approach	CR	acc-top1 (%)	acc-top5 (%)
TTQ (2 bits) [30]	16	66.6	87.2
ADMM (2 bits) [18]	16	67.0	87.5
LQ-Net (2 bits) [28]	16	68.0	88.0
LQ-Net (3 bits) [28]	10	69.3	88.8
LQ-Net (4 bits) [28]	7	70.0	89.1
DPR (2 bits)	16	63.0	84.5
DPR (3 bits)	10	69.2	88.6
DPR (4 bits)	7	70.3	89.5
DPQ (2 bits)	16	67.7	87.9
DPQ (3 bits)	10	69.8	89.3
DPQ (4 bits)	7	70.5	89.6

that compressed with LQ-Net. And for the quantization with 4 bits, the accuracy of the model compressed with DPR is also better than that with LQ-Net. This shows that the proposed approach, especially DPQ, has superior or comparable performance of compression to the recent LQ-Net approach.

4.3 Compressing Robust Models

The proposed DPR and DPQ are extended to compress the robust DNN models. They are denoted by DPR⁺ and DPQ⁺ respectively, and train the compressed model through solving (10) and (11) based on the TRADES technique [29]. The training settings and evaluating settings are consistent with the public codes² of TRADES. We choose $t = 5$ for DPR⁺ because their training epochs are small. We first test a small network proposed by [1], which consists of four convolutional layers and three FC layers. We called it SmallCNN and train it on the MNIST dataset. The pretrained model achieves 99.54% accuracy on normal testing data and 96.91% accuracy under a powerful attack algorithm named PGD (projected gradient descent) [22]. Then, we train a robust ResNet-18 model on CIFAR-10, which achieves 92.44% accuracy on normal testing data and 46.74% accuracy under the PGD attack.

Because there is few work on compressing a robust model with quantization approach, we consider a baseline called DP₀ for comparison. DP₀ directly quantizes the weights of the pretrained model with the proposed DP based algorithm to realize compression. Table 4 shows the accuracy drop of different compressed robust models obtained with DP₀, DPR⁺ and DPQ⁺. From the table, we see that DP₀ employing the simple weight-clustering quantization cannot preserve the accuracy of the robust model. It results in the accuracy drop on natural example and adversarial example up to 22.6%. With the proposed DPR⁺ and DPQ⁺ approaches, the accuracy drop caused by the compressed model is remarkably reduced (no more than 4.81%). This is due to the proposed formulations (10) and (11) for training the compressed robust model. On the other hand, we find out that DPR⁺ performs better than DPQ⁺. Although they have comparable accuracy drop on the natural examples, DPR⁺ produces the model with 2% less accuracy drop than DPQ⁺ on the adversarial example. With 2-bit quantization, DPR⁺ obtains a compressed robust ResNet-18 model which exhibits **less than 3%** accuracy drop on both natural and adversarial examples.

²<https://github.com/yaodongyu/TRADES>

Table 4: The results of compressing the robust SmallCNN (on MNIST) and ResNet-18 (on CIFAR-10). Δ_{nat} and Δ_{adv} are the accuracy changes for natural images and adversarial images compared to the pretrained model, respectively.

Approach	CR	Model	Δ_{nat} (%)	Δ_{adv} (%)
DP ₀ (2 bits)	14	SmallCNN	-0.44	-5.90
DPR ⁺ (2 bits)	14	SmallCNN	-0.11	-2.38
DPQ ⁺ (2 bits)	14	SmallCNN	-0.37	-4.44
DP ₀ (2 bits)	16	ResNet-18	-22.48	-22.60
DPR ⁺ (2 bits)	16	ResNet-18	-0.98	-2.77
DPQ ⁺ (2 bits)	16	ResNet-18	+0.40	-4.81

5 CONCLUSIONS

A dynamical programming based algorithm for scalar clustering and two DNN compression schemes, DPR and DPQ, are proposed. DPR includes training a clustering-friendly network with a formulation including the regularization item and the DP based algorithm obtaining the optimal solution of scalar weight clustering. DPQ includes using the DP based algorithm to find a better quantizer and training/compressing the DNN with a formulation with quantization function. DPR and DPQ are also extended to compress robust DNNs, through a combination with the TRADES approach [29].

Exhaustive experiments have been carried out to show the advantages of the proposed approaches over existing counterparts for compressing normal and robust DNNs. Experimental results also show, the DPR approach performs better for robust models while the DPQ approach is more suitable for large DNN models.

6 ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Plan of China (2020AAA0103502), and Beijing National Research Center for Information Science and Technology (BNR2019ZS01001).

REFERENCES

- [1] N. Carlini and D. Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proc. IEEE Symposium on Security and Privacy*. 39–57.
- [2] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. 2015. Compressing neural networks with the hashing trick. In *Proc. ICML*. 2285–2294.
- [3] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. 2016. Compressing convolutional neural networks in the frequency domain. In *Proc. SIGKDD*. 1475–1484.
- [4] 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. *arXiv preprint arXiv:1602.02830* (2016).
- [5] R. Dai, L. Li, and W. Yu. 2018. Fast training and model compression of gated RNNs via singular value decomposition. In *Proc. IJCNN*. 1–7.
- [6] S. E. Dreyfus and A. M. Law. 1977. *Art and Theory of Dynamic Programming*. (1977).
- [7] T. Garipov, D. Podoprikin, A. Novikov, and D. Vetrov. 2016. Ultimate tensorization: Compressing convolutional and FC layers alike. *arXiv preprint arXiv:1611.03214* (2016).
- [8] Y. Gong, L. Liu, M. Yang, and L. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115* (2014).
- [9] I. J. Goodfellow, J. Shlens, and C. Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
- [10] S. Gui, H. N. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu. 2019. Model compression with adversarial robustness: A unified optimization framework. In *Proc. NIPS*. 1285–1296.
- [11] S. Han, H. Mao, and W. J. Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149* (2015).
- [12] S. Han, J. Pool, J. Tran, and W. Dally. 2015. Learning both weights and connections for efficient neural network. In *Proc. NIPS*. 1135–1143.
- [13] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proc. CVPR*. 770–778.

- [14] G. Hinton, O. Vinyals, and J. Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [15] T. Hu, T. Chen, H. Wang, and Z. Wang. 2019. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. In *Proc. ICLR*.
- [16] A. Krizhevsky, G. Hinton, et al. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.
- [17] Y. LeCun, J. S. Denker, and S. A. Solla. 1990. Optimal brain damage. In *Proc. NIPS*. 598–605.
- [18] 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 *Proc. AAAI*. 3466–3473.
- [19] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. 2019. Rethinking the value of network pruning. In *Proc. ICLR*.
- [20] S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Information Theory* 28, 2 (1982), 129–137.
- [21] Y. Ma, R. Chen, W. Li, F. Shang, W. Yu, M. Cho, and B. Yu. 2019. A unified approximation framework for compressing and accelerating deep neural networks. In *Proc. ICTAI*. 376–383.
- [22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *Proc. ICLR*.
- [23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. Going deeper with convolutions. In *Proc. CVPR*. 1–9.
- [24] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin. 2018. Deep k -Means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In *Proc. ICML*. 5363–5372.
- [25] W. Yu, Y. Gu, and Y. Li. 2018. Efficient randomized algorithms for the fixed-precision low-rank matrix approximation. *SIAM J. Matrix Anal. Appl.* 39, 3 (2018), 1339–1359.
- [26] S. Zagoruyko and N. Komodakis. 2016. Wide residual networks. In *Proc. BMVC*.
- [27] H. Zha, X. He, C. Ding, M. Gu, and H. D. Simon. 2002. Spectral relaxation for k -means clustering. In *Proc. NIPS*. 1057–1064.
- [28] D. Zhang, J. Yang, D. Ye, and G. Hua. 2018. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. In *Proc. ECCV*. 365–382.
- [29] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *Proc. ICML*. 7472–7482.
- [30] C. Zhu, S. Han, H. Mao, and W. Dally. 2017. Trained ternary quantization. In *Proc. ICLR*.