Hyperspherical Quantization: Toward Smaller and More Accurate Models

Dan Liu[1], Xi Chen[1], Chen Ma[2], Xue Liu[1]
[1]McGill University, [2]City University of Hong Kong
dan.liu4@mail.mcgill.ca,
{xi.chen11,xue.liu}@mcgill.ca,
chenma@cityu.edu.hk

Abstract

Model quantization enables the deployment of deep neural networks under resource-constrained devices. Vector quantization aims at reducing the model size by indexing model weights with full-precision embeddings, i.e., codewords, while the index needs to be restored to 32-bit during computation. Binary and other low-precision quantization methods can reduce the model size up to 32×, however, at the cost of a considerable accuracy drop. In this paper, we propose an efficient framework for ternary quantization to produce smaller and more accurate compressed models. By integrating hyperspherical learning, pruning and reinitialization, our proposed Hyperspherical Quantization (HQ) method reduces the cosine distance between the full-precision and ternary weights, thus reducing the bias of the straight-through gradient estimator during ternary quantization. Compared with existing work at similar compression levels (∼30×, ∼40×), our method significantly improves the test accuracy and reduces the model size.

1. Introduction

Despite promising results in real-world applications, deep neural network (DNN) models usually contain a large number of parameters, making them impossible to deploy on edge devices. A significant amount of research has been made to reduce the size and computational overhead of DNN models through quantization and pruning.

Pruning brings high sparsity, but cannot take advantage of compression and acceleration without customized hardware [25]. Cluster-based quantization, such as vector quantization and product quantization, remarkably reduces the model disk footprint [67, 53, 7], but its memory footprint is larger than that of the low-precision quantization method [12, 60, 10, 83, 37], as the actual weight values involved in computation remain full-precision [12]. Ultra-low-precision quantization, *e.g.*, binary [33, 11, 10], ternary [37, 83], and 2-bit quantization [82, 8, 18], has fast inference and low memory footprint [60], but it usually leads to a significant accuracy drop, due to the inaccurate weights [22] and gradients [78].

Gradually discretizing the weights can overcome such non-differentiability [52, 34, 9], *i.e.*, reducing the discrepancy between the quantized weights in the forward pass and the full-precision weights in the backward pass. However, it only performs well with 4-bit (or higher) precision as the ultra-low bit quantizer may seriously damage the weight magnitude leading to unstable weights [22]. Intuitively, ternary quantization barely affects the sign of the weights, making the direction of weight vectors [62] changes relatively more stable than their magnitude. Recently, many studies [48, 49, 47, 15, 13, 59, 5] show that the angular information [45] preserves the key semantics in feature maps.

We propose hyperspherical quantization (HQ), a method combining pruning and reinitialization [20, 81] to produce accurate ternary DNN models with a smaller memory/disk footprint. We first pre-train a DNN model with a hyperspherical learning method [49] to preserve the direction information [62] of the model weights, then apply our proposed approach to push the full-precision weights close to their ternary counterparts, and lastly, we combine the straight-through estimator (STE) [2] with a gradually increased threshold to fulfill the ternary quantization process. Our main contributions are summarized as follows:

- We demonstrate that simply integrating pruning and reinitialization can significantly reduce the impact of weight discrepancy caused by the ternary quantizer. We unify the pruning and quantization thresholds to one to further optimize the quantization process.

- Our method significantly outperforms existing works in terms of the size-accuracy trade-off of DNN models. For example, on ImageNet, our method can compress a ResNet-18 model from 45 MB to 939 KB (48× compressed) while the accuracy is only 4% lower than the original accuracy. It is the best result among the existing results (43×, 6.4% accuracy drop).

2. Related Work

2.1. Hyperspherical Learning

Hyperspherical learning aims to study the impact of the direction [62] of weight vectors on DNN models. [62] discovers that detaching the model weight direction information from its magnitude can accelerate training. [49] shows that the direction information of weight vectors, in contrast to weight magnitude, preserves useful semantic meanings in feature maps. [47, 15, 72] propose to apply regularization to angular representations on a hypersphere to enhance the model generalization ability in face recognition tasks. [46, 47, 15, 13, 59, 5, 44] further study the empirical generalization ability of hyperspherical learning.

2.2. Quantization

Low-bit quantization methods convert float values of weights and activations to lower bit values [10, 11, 60, 54]. These methods make it possible to substantially reduce the computational cost during CPU inference [71]. For example, binary quantization [10, 11] compresses full-precision weights into a 1-bit representation, thus significantly reducing the memory footprint by 32$\times$.

Clustering-based weight quantization, such as product quantization [21] and vector quantization [23, 4, 67] focus on optimizing the size-accuracy trade-off and can significantly compress the disk footprint by grouping weight values to a codebook. Common approaches cluster weights through k-means [67, 75, 66, 23] and further finetune the clustering center by minimizing the reconstruction error in the network [19, 67, 75]. The compression ratio and accuracy trade-off can be adjusted by changing the number of groups. [26] applies k-means-based vector quantization with pruning and Huffman coding to further reduce the model size. However, the codebook usually consists of 32-bit float numbers [67, 53], the memory footprint during computation is uncompressed.

Some mixed-precision quantization methods overcome the shortcomings of low-bit and clustering based methods by means of reinforcement learning [73], integer programming [3], and differentiable neural architecture search [74], so as to apply different bit-widths in model weights to optimize inference time and model size. However, mixed-precision still cannot effectively compress the model size due to the use of 8 to 32-bit weights. Other mixed-precision quantization methods assign different bit widths to layer weights according to various measures, including hardware [73, 77] and second-order information [17, 65].

2.3. Pruning

Pruning consists of structured and unstructured methods. It can greatly compress redundancy and maintain high accuracy. Unstructured pruning brings high sparsity, but cannot take advantage of acceleration without customized hardware [25]. Only structured pruning methods can reduce the inference latency and are easier to accelerate [30, 38] because the original weight structures of the model are preserved. Unstructured pruning uses criteria, such as gradient [56, 35], and magnitude [26, 55] information, to remove individual weights; structured pruning [38, 32, 1] aims to remove unimportant channels of the convolutional layer based on similar criteria. The lottery ticket hypothesis [20] shows that there exists sparse subnetworks that can be trained from scratch and achieve the same performance as the full network. [81] studies the lottery ticket hypothesis from the perspective of weight reinitialization and points out that the key premise is the sign of weight values.

Re-training after pruning [20, 81] reveals the link between the network structure and performance. Furthermore, our findings show that training after pruning and reinitialization can be used to produce more accurate and highly compressed ternary weights, which surpasses the current model compression methods and has a wide range of application scenarios.

3. Preliminary and Notations

3.1. Hyperspherical Model

A general representation of a hyperspherical neural network layer [49] is:

$$\mathbf{y} = \phi(\mathbf{W}^T\mathbf{x}), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n\times m}$ is the weight matrix, $\mathbf{x} \in \mathbb{R}^n$ is the input vector to the layer, ϕ represents a nonlinear activation function, and $\mathbf{y} \in \mathbb{R}^m$ is the output feature vector. The input vector $\mathbf{x}$ and each column vector $\mathbf{w}_j \in \mathbb{R}^n$ of $\mathbf{W}$ satisfy $\|\mathbf{w}_j\|_2 = 1$,$\|\mathbf{x}\|_2 = 1$ for all $j = 1, ..., m$.

3.2. Ternary Quantizer

In this work, the ternary quantizer is:

$$\hat{\mathbf{W}} = \texttt{Ternary}(\mathbf{W}, \Delta) = \begin{cases} \dfrac{1}{\sqrt{|\mathbf{I}_\Delta|}} : w_{ij} > \Delta, \\ \qquad 0 : |w_{ij}| \leq \Delta, \\ -\dfrac{1}{\sqrt{|\mathbf{I}_\Delta|}} : w_{ij} < -\Delta, \end{cases} \tag{2}$$

where $\mathbf{W}$ is the full-precision weights, Δ is a threshold, $\mathbf{I}_\Delta = \{i||w_{ij}| > \Delta\}$ [37], and $|\mathbf{I}_\Delta|$ denotes total non-zero values in the j-th column vector $\mathbf{w}_j$. With $\Delta = 0$, $\texttt{Ternary}(\cdot)$ becomes a variant of $\texttt{Binary}(\cdot)$ operation. And $\phi(\hat{\mathbf{w}}_j^T\mathbf{x}) = \phi(\frac{1}{\sqrt{|\mathbf{I}_\Delta|}}\bar{\mathbf{w}}_j^T\mathbf{x})$ s.t. $\bar{\mathbf{w}} \in \{-1, 0, 1\}$.

3.3. Pruning

The unstructured pruning [26] is defined by:

$$\mathbf{W}' = \texttt{Prune}(\mathbf{W}, r) = \mathbf{W} \odot \mathbf{M}, \tag{3}$$

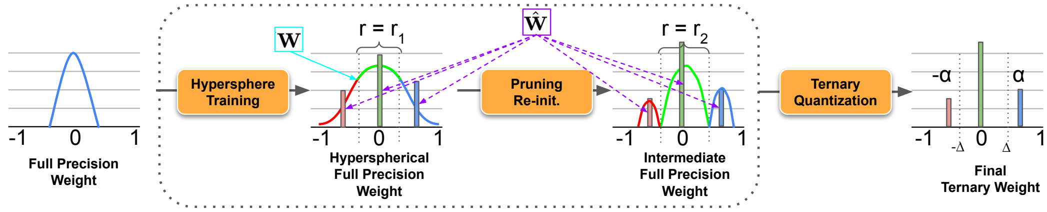


Figure 1: The overall preprocessing and ternary quantization process of our proposed method (Section 4.1). $\alpha = \frac{1}{\sqrt{|\mathbf{I}_\Delta|}}$, $r_1 = r_l$, and $r_2 = r_h$. The dotted-line square denotes the preprocessing step. The "**Pruning Re-init.**" process can increase $\mathcal{S}(\hat{\mathbf{W}}, \mathbf{W})$, *i.e.*, reduce $\mathcal{D}$ (Section 4.1.1, 4.1.2). We seek to minimize $\mathcal{D}$ before "**Ternary Quantization**".

where $\odot$ denotes the element-wise multiplication. Mask $\mathbf{M}$ selects the top r percent of the smaller weights in $\mathbf{W}$:

$$\mathbf{M} = \texttt{Mask}(\mathbf{W}, r) = \texttt{Sign}(|\texttt{Ternary}(\mathbf{W}, \Delta)|), \quad (4)$$

where $\Delta = \texttt{threshold}(\mathbf{W}, r)$, and $0 \leq r < 1$. The $\texttt{threshold}(\mathbf{W}, r)$ returns the corresponding minimum value of matrix $\mathbf{W}$ based on the percentage r. We use pruning to represent unstructured pruning for simplicity.

3.4. Cosine Similarity $\mathcal{S}$ on HyperSphere

Based on Eq. (2) and hyperspherical learning, we have $\|\hat{\mathbf{w}}_j\|_2 = 1$ and $\|\mathbf{w}_j\|_2 = 1$. The vector-wise cosine similarity between $\mathbf{w}_j$ and $\hat{\mathbf{w}}_j$ is:

$$\mathcal{S}(\hat{\mathbf{w}}_j, \mathbf{w}_j) = \frac{\hat{\mathbf{w}}_j \cdot \mathbf{w}_j}{\|\hat{\mathbf{w}}_j\|_2 \|\mathbf{w}_j\|_2} = \sum_{i=1}^{n} \frac{1}{\sqrt{|\mathbf{I}_\Delta|}} |w_{ij}|. \quad (5)$$

If without pruning, $\mathbf{W}$ and $\hat{\mathbf{W}}$ will not contain 0. With $\Delta = 0$, we have $|\mathbf{I}_\Delta| = n$ and the cosine similarity between $\hat{\mathbf{W}}$ and $\mathbf{W}$ becomes:

$$\mathcal{S}(\hat{\mathbf{W}}, \mathbf{W}) = \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{1}{\sqrt{n}} |w_{ij}|. \quad (6)$$

After applying pruning (Eq. (3)) to $\mathbf{W}$:

$$\mathcal{S}(\hat{\mathbf{W}}', \mathbf{W}') = \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{1}{\sqrt{|\mathbf{I}_\Delta|}} |w'_{ij}|. \quad (7)$$

The cosine distance between full-precision and ternary weights is:

$$\mathcal{D} = 1 - \frac{1}{l} \sum_{k=1}^{l} \mathcal{S}(\hat{\mathbf{W}}_k, \mathbf{W}_k), \quad (8)$$

where l denotes the number of quantized layers.

4. Hyperspherical Quantization

In this section, we propose using pruning to increase the cosine similarity $\mathcal{S}$ between the full-precision weights $\mathbf{W}$ and the ternary weights $\hat{\mathbf{W}}$. We show how we can effectively quantize such discrepancy reduced model weights.

Our proposed method includes the preprocessing and quantization steps (Fig. 1). In the preprocessing step, we use iterative pruning with gradually increasing sparsity and reinitialization [81] to push $\mathbf{W}$ close to its ternary counterpart $\hat{\mathbf{W}}$. In the quantization step, as $\mathbf{W}$ is close to $\hat{\mathbf{W}}$ after the first step, it is easy to obtain a more accurate $\hat{\mathbf{W}}$ by using regular STE-based ternary quantization methods. We unify the thresholds of pruning and quantization as one single threshold during ternary quantization process.

4.1. Increasing $\mathcal{S}$ by Preprocessing

We show that pruning on hypersphere can increase the cosine similarity $\mathcal{S}(\hat{\mathbf{W}}, \mathbf{W})$, thus pushing the full-precision weight close to its ternary version. But the decayed weights during training cause unstable $\mathcal{S}$, which makes ternary quantization infeasible. Then the reinitialization is proposed to stabilize $\mathcal{S}$.

4.1.1 Cosine Similarity and Hyperspherical Pruning

Given a full-precision $\mathbf{W}$ and its ternary form $\hat{\mathbf{W}}$, we seek to optimize the following problem under hyperspherical learning settings:

$$\max_{r} \ \mathcal{S}(\hat{\mathbf{W}}', \mathbf{W}') \quad (9)$$

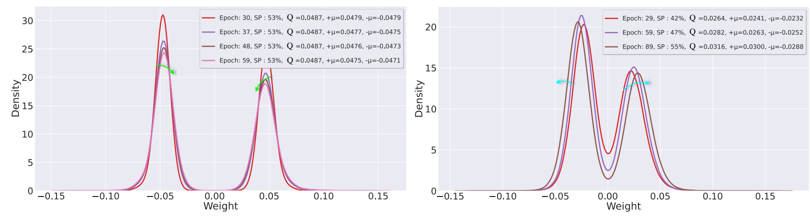
$$s.t. \ \ 0 < \mathcal{S} \leq 1,$$

where $\mathbf{W}' = \texttt{Prune}(\mathbf{W}, r)$ and $\hat{\mathbf{W}}' = \texttt{Ternary}(\mathbf{W}', 0)$. Obviously, if $\|\mathbf{W}'\|_0 = 1$ then $\mathcal{S}(\hat{\mathbf{W}}', \mathbf{W}') = 1$, but it is meaningless. Although there is no explicit solution for r, we can increase $\mathcal{S}$ by gradually increasing r. Since the model is trained with hyperspherical learning, based on Eq. (6)-(7), with pruning ratio r, we have $\frac{1}{\sqrt{|\mathbf{I}_\Delta|}} \geq \frac{1}{\sqrt{n}}$ and:

$$|w'_{ij}| = \frac{|w_{ij}|}{\|\mathbf{w}'\|_2} \geq \frac{|w_{ij}|}{\|\mathbf{w}\|_2} = |w_{ij}|, \quad (10)$$

where $\|\mathbf{w}'\|_2 \leq \|\mathbf{w}\|_2 = 1$. Therefore:

$$\mathcal{S}(\hat{\mathbf{W}}', \mathbf{W}') \geq \mathcal{S}(\hat{\mathbf{W}}, \mathbf{W}) \quad (11)$$



(a) ResNet-50 with the fixed sparsity. (b) ResNet-50 with iterative processing.

Figure 2: Weight distribution of a hidden convolutional layer. The zero-drifting is relieved by iterative pruning and reinitialization. $Q = |\hat{w}| = |\frac{1}{\sqrt{\mathbf{I}_\Delta}}|$, $\pm\mu$ denotes the mean values of $w^+ \in \mathbb{R}_{>0}$ and $w^- \in \mathbb{R}_{<0}$ during training. (a) When the sparsity is fixed, due to the decayed weights, $|\mu|$ is heading towards zero. (b) When we iteratively performe pruning and reinitialization, $|\mu|$ grows with the sparsity and keeps moving closer to $|\hat{w}|$ as epoch increases.

Eq. (10)-(11) indicate that pruning with ratio r can increase the cosine similarity $\mathcal{S}(\hat{\mathbf{W}}', \mathbf{W}')$. However, we still need a proper pruning ratio to maintain the model's performance, as the over-pruned weight has a higher $\mathcal{S}$ but may not perform well. For example, $\mathbf{w} = [0.3, 0.2, 0.0001]$, $\mathbf{w}' = [0.3, 0.2, 0]$, $\hat{\mathbf{w}} = [1, 1, 1]$, and $\hat{\mathbf{w}}' = [1, 1, 0]$, then $\mathcal{S}(\hat{\mathbf{w}}', \mathbf{w}') = 0.98$ is greater than $\mathcal{S}(\hat{\mathbf{w}}, \mathbf{w}) = 0.80$. If we further apply pruning to $\mathbf{w}'$ then we have $\mathbf{w}'' = [0.3, 0, 0]$, $\hat{\mathbf{w}}'' = [1, 0, 0]$, and $\mathcal{S}(\hat{\mathbf{w}}'', \mathbf{w}'') = 1.0$. It is obvious that $\hat{\mathbf{w}}''$ cannot perform as well as $\hat{\mathbf{w}}'$.

4.1.2 Zero-Drifting and Iterative Reinitialization

In addition to the pruning ratio r, Eq. (5) indicates that $\mathcal{S}$ is also related to $|w_{ij}|$. In this section, we discuss how the decayed weights has negative impact on $\mathcal{S}$, and how to mitigate such impact through periodical reinitialization.

Weight decay and learning rate push the weight values toward and around zeros [16, 58, 31]. Learning rate plays a similar role as weight decay (see Eq. (8) in the work of [27]). As w_{ij} drifting toward zero during training (Fig. 2a), $\mathcal{S}(\hat{\mathbf{W}}, \mathbf{W})$ is also reduced (Eq. (5)). Intuitively, this zero-drifting phenomenon hinders accurate ternary quantization as the reduced $\mathcal{S}(\hat{\mathbf{W}}, \mathbf{W})$ enlarges the quantization searching space. We reinitialize the weights by $\mathbf{W} = \texttt{Ternary}(\mathbf{W}, 0)$ to neutralize the negative effect of the decayed weights.

Fig. 2a shows the zero-drifting phenomenon with fixed pruning sparsity. Our experimental results show that iterative pruning and reinitializing can increase $\mathcal{S}(\hat{\mathbf{W}}, \mathbf{W})$ and prevent the negative impact of decayed weights (Table 5,6). Once we periodically prune, reinitialize and re-train the model, we always have a larger $\mathcal{S}$ (Table 5,6) and the zero-drifting is relieved (Fig. 2b, Table 5).

4.2. Quantization with a Unified Threshold

With $\mathbf{W}$ close to $\hat{\mathbf{W}}$ in the preprocessing step, we still need to perform weight quantization and pruning to further increase $\mathcal{S}(\hat{\mathbf{W}}, \mathbf{W})$. We introduce a gradually increasing quantization threshold Δ (Eq. 2) to unify pruning and quantization, as Δ can be seen as a pruning threshold. The non-differentiability of $\texttt{Ternary}(\cdot)$ is bypassed with STE [2]:

Forward:

$$\hat{\mathbf{W}} = \texttt{Ternary}(\mathbf{W}, \Delta) \tag{12}$$

Backward:

$$\frac{\partial E}{\partial \mathbf{W}} = \frac{\partial E}{\partial \hat{\mathbf{W}}} \frac{\partial \hat{\mathbf{W}}}{\partial \mathbf{W}} \underset{STE}{\approx} \frac{\partial E}{\partial \hat{\mathbf{W}}}. \tag{13}$$

STE essentially ignores the quantization operation and approximates it with an identity function.

The threshold Δ should gradually increase along with the training error, and such increase should slow down after the model converges (Fig. 3). Therefore, we directly use the averaged gradients to update the pruning threshold:

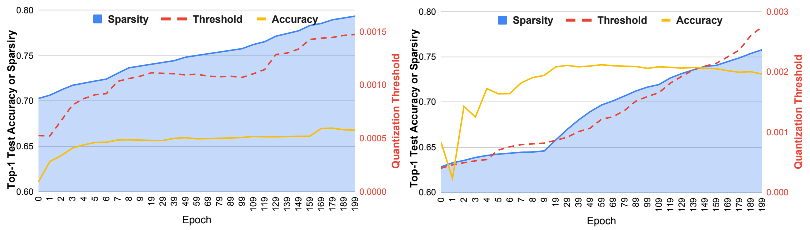
$$\boldsymbol{\Delta} = \sum_{i=1}^{n} \frac{\partial L}{\partial w_i}, w_i \in \mathbf{w} \text{ and } w_i \neq 0. \tag{14}$$

The gradient of $\frac{\partial L}{\partial w_i}$ where $w_i = 0$ is ignored. The training error is very large at the beginning, leading to a rapid increase of the threshold. As the model gradually converges, the training error will decrease and the threshold growth slows down (Fig. 3). The model sparsity is gradually increasing along with the learned threshold. The accuracy starts to decrease if the quantization continues after the model converges.

4.3. Implementation Details

4.3.1 Training Algorithm

Our proposed method is shown in Algorithm 1. We prune the model with a ratio from $r_l = 0.3$ to $r_h = 0.7$ based on [50, 83]. The overall process can be summarized as: i) Pre-training with hyperspherical learning architecture [49]; ii) Iterative preprocessing the model weights, *i.e.*, prune the model to target sparsity r_h and reset the weights via $\mathbf{W} = \texttt{Ternary}(\mathbf{W}, 0)$ after each pruning (Fig. 1); iii) Ternary quantization, updating the weights and Δ through STE.



(a) ResNet-18 on ImageNet. (b) ResNet-50 on ImageNet.

Figure 3: The trend of sparsity, threshold Δ and accuracy during ternary quantization. The blue area denotes model sparsity.

Algorithm 1 HQ training approach

1: **Input:** Input $\mathbf{x}$, a hyperspherical neural network $\phi(\mathbf{W}, \cdot)$, $r_l = 0.3$, $r_h = 0.7$, and step size $\delta = 0.01$.
2: **Result:** Quantized ternary network for inference
3: **1. Preprocessing:**
4: $r = r_l$
5: **while** $r < r_h$ **do** ▷ Iterative pruning and reinitializing
6: $\quad \mathbf{M} = \texttt{Mask}(\mathbf{W}, r)$ ▷ Obtain the pruning mask
7: $\quad \mathbf{W} = \mathbf{W} \odot \mathbf{M}$ ▷ Pruning
8: $\quad \mathbf{W} = \texttt{Ternary}(\mathbf{W}, 0)$ ▷ Reinitialization
9: $\quad$ **while** *not converged* **do**
10: $\quad\quad y = \phi((\mathbf{M} \odot \mathbf{W}), \mathbf{x})$ ▷ Eq. (1)
11: $\quad\quad$ Perform SGD, calculate $\frac{\partial L}{\partial \mathbf{W}}$, and update $\mathbf{W}$
12: $\quad$ **end while**
13: $\quad r += \delta$ ▷ Increase the pruning ratio r
14: **end while**
15: **2. Ternary Quantization:**
16: **while** *not converged* **do**
17: $\quad \hat{\mathbf{W}} = \texttt{Ternary}(\mathbf{W}, \Delta)$
18: $\quad y = \phi((\mathbf{M} \odot \hat{\mathbf{W}}), \mathbf{x})$
19: $\quad$ Get $\frac{\partial L}{\partial \mathbf{W}}$ via SGD; update $\mathbf{W}$, Δ ▷ Eq. (13,14)
20: **end while**

4.3.2 Training Settings and Training Time

For image classification, the batch size is 128. The weight decay is 0.0001, and the momentum of stochastic gradient descent (SGD) is 0.9. We use the cosine annealing schedule with restarts every 10 epochs [51] to adjust the learning rates. The initial learning rate is 0.001. All of the experiments use 16-bit half-precision from PyTorch to accelerate the training process. Thus, all parameters (except codebook with 2-bit) are stored as 16-bit precision values.

It takes about 50 epochs to convert a Pytorch model to a hyperspherical one. The SGD loop (Line 5-14 in Algorithm 1) takes at least 200 epochs. The ternary quantization loop (Line 16-20 in Algorithm 1) takes about 200 epochs.

4.3.3 Compression Strategy

Inspired by vector quantization, we use codebook as the compression method [67, 53]. Unlike the vector quantization methods that use a learned codebook, we use the Huffman table [70] as a fixed codebook (Table 2 in the Appendix). We use gzip to finalize the model file as in other work [69, 67]. It is shown that the Huffman table can maximize the compression effect when each codeword consists of three ternary values, *e.g.*, $\{0, 0, 0\}$. Note that the Huffman table can only boost the compression when the high-frequency patterns exist, such as $\{0, 0, 0\}$. For other works with low-sparsity or non-sparse quantization [67, 7, 53, 83], applying the Huffman table may not help compress the model. We find inconsistent compression results in ABGD [67] and PQF [53]. The actual size of their models (obtained from github in gzip format) is shown in Table 2 is different from the results stated in their paper.

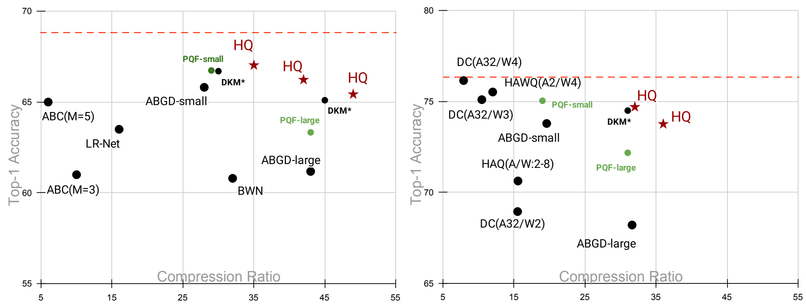
5. Experiments

Our experiments involve image classification and object detection tasks. We evaluate our method on ImageNet data set [61] with MobileNetV2 [63] and ResNet-18/50 [29]. For object detection, we use the MS COCO [42] dataset and Mask R-CNN [28, 76]. The pre-trained weights are provided by the PyTorch zoo and Detectron2 [76].

5.1. Image Classification

Following the practices of mainstream model compression work [67, 17, 53], when compressing the model size, we quantize all of the weights of the convolution and the fully-connected (FC) layers to 2-bit (except the first layer). We compare our results with leading compression results from PQF [53], ABGD [67], BRECQ [41], HAWQ [17], and TQNE [19]. We also compare our work with milestone approaches, such as ABC-Net [43], Deep Compression (DC) [26], Hardware-Aware Automated Quantization (HAQ) [73], Hessian AWare Quantization (HAWQ) [17], LR-Net [64], and BWN [60].

Our method significantly outperforms leading model compression methods in terms of bit-width, compression



(a) ResNet-18 on ImageNet.

(b) ResNet-50 on ImageNet.

Figure 4: The compression ratio and accuracy of ResNet-18/50 on ImageNet. Our method achieves much higher accuracy and compression ratio compared to other work. The dash-line is the baseline accuracy from PyTorch zoo.

Table 1: Ternary quantization results on ImageNet. We leave the last FC layer as full-precision like other works.

Models	Methods	Acc.
ResNet-18 Acc.: 69.76	**HQ (Ours)**	**68.5**
	TWN (2016) [37]	61.8
	TTQ (2016) [83]	66.6
	INQ (2017) [80]	66.0
	ADMM (2018) [36]	67.0
	LQ-Net (2018) [79]	68.0
	AdaRound (2020) [57]	55.9
	BRECQ (2021) [41]	66.3
	RTN (2020) [40]	68.5
ResNet-50 Acc.: 76.15	**HQ (Ours)**	**75.2**
	TWN[37]	72.5
	LQ-Net [79]	75.1
	AdaRound [57]	47.5
	BRECQ [41]	72.4

ratio and accuracy (Fig. 4, Table 2). It compresses ResNet-18 from 45 MB to 1.28MB (35× compressed) while maintaining high accuracy (67.03% vs. 69.7% of the original model), and compress ResNet-50 from 99 MB to 3.1MB (32× compressed) with an accuracy of 74.7% (vs. 76.15% of the original model). In the extreme cases, our method produces much smaller models with less than 4% accuracy drop, for example, 935KB out of 45 MB (48× compressed) for ResNet-18 and 2.6 MB out of 99 MB (37× compressed) for ResNet-50. Although DKM's results [7] are close to ours, their models cannot reduce the memory footprint.

For MobileNetV2, our method performs better at above 15× compression level. Quantizing the pointwise layer of MobileNet leads to significant accuracy loss [24]. That is the reason why the fully quantized 2-bit precision methods [26, 41] have low accuracy. Whereas, HAQ [73] applies mixed-precision to improve accuracy. In addition, the works [14, 24, 68] show that a less complex model is sensitive to ternary quantization due to the potential lack of redundant representation capability.

We also compare our work with conventional ternary quantization works which leave the last FC layer as full-precision (Table 1). Our work achieves comparable results with other leading methods.

5.2. Object Detection and Segmentation

Similar to previous work [67, 53, 41], we test our method on the Mask R-CNN [28] architecture with ResNet-50 backbone to verify its generalizability. The source code, hyperparameters and the pre-trained model are provided by Detectron2 [76]. We apply our method to the entire model except for the first layer. We compare against recent baselines, such as the ABGD [67], PQF [53], and BRECQ [41]. As shown in Table 3, compared to ABGD and PQF, our method gives a higher compression ratio and a similar or better recognition result.

5.3. Model Size and Accuracy

The accuracy, sparsity and size of the quantized ResNet-18 models are shown in Table 4 and Fig. 5. The results show that the model accuracy will increase with the sparsity until it reaches a certain level (the triangle symbol in Fig. 5). Then the accuracy starts to decrease as the pruning continues, which is the same as the Figure 5 in TTQ [83]. This phenomenon is different from pruning, where the accuracy decreases linearly as the sparsity increases. One possible reason is that the capacity [68] of the quantized model changes with the portion of 0 and $\pm\frac{1}{\sqrt{|\mathbf{I}_\Delta|}}$. The model capacity is low when the number of $\pm\frac{1}{\sqrt{|\mathbf{I}_\Delta|}}$ is dominant (close to binary). As the sparsity increases, the weight tends to become ternary, whose capacity is higher than binary weights. Pruning does not have this issue since it has full-precision weight values. As the proportion of 0 keeps increasing, the capacity will drop, leading to accuracy drop.

Table 2: Model compression results on ImageNet. “Bits (W/A)” denotes the bit-width of weight and activation. “Ratio” denotes the storage compression level. “+” denotes 16-bit weight precision in FC layer. “Size” denotes disk size. “*” indicates gzip-compressed publicly available models. The detailed bit allocations can be found in the Appendix.

Models	Comp. Level	Methods	Bits (W/A)	Acc.	Size	Ratio
ResNet-18 Acc.: 69.76 Size: 45 MB	~30×	**HQ (Ours)**	**2/16**	**67.03**	**1.23 MB**	**37×**
		DKM (2022)[7]	32/32	66.7	1.49 MB	30×
		PQF* (2021)[53]	32/32	66.74	1.54 MB	30×
		ABGD* (2020)[67]	32/32	65.81	1.51 MB	30×
	~40×	**HQ (Ours)**	**2/16**	**65.48**	**939 KB**	**48×**
		DKM(2022)[7]	32/32	65.1	1 MB	45×
		PQF* (2021)[53]	32/32	63.33	1.04 MB	43×
		ABGD* (2020)[67]	32/32	61.18	1.01 MB	45×
ResNet-50 Acc.: 76.15 Size: 99 MB	10−20×	**HQ (Ours)**	**$2/16^{+}$**	**75.2**	6.89 MB	14×
		HAWQ (2019)[17]	2~8/4~8	75.4	7.96 MB	12×
		PQF* (2021)[53]	32/32	75.04	5.10 MB	19×
		TQNE (2020)[19]	32/32	74.3	-	19×
		ABGD* (2020)[67]	32/32	73.79	5.01 MB	20×
		HAQ (2019)[73]	2~8/2~8	70.63	6.30MB	16×
		DC (2015)[26]	2/32	68.95	6.32MB	16×
	~30×	**HQ (Ours)**	**2/16**	**73.87**	**2.63 MB**	**38×**
		HQ (Ours)	**2/16**	**74.7**	**3.01MB**	**33×**
		DKM (2022)[7]	32/32	74.5	3.32 MB	29×
		PQF* (2021)[53]	32/32	72.18	3.26 MB	30×
		TQNE (2020)[19]	32/32	68.8	-	32×
		ABGD* (2020)[67]	32/32	68.21	3.16 MB	31×
MobileNetV2 Acc.: 71.88 Size: 14 MB	15−20×	**HQ (Ours)**	2/16	58.74	**0.71 MB**	**20×**
		HAQ (2019)[73]	2~8/2~8	66.75	0.95 MB	15×
		BRECQ (2021)[41]	2/8	56.29	0.83 MB	17×
		Han *et al.* [26, 73] (2015)	2/32	58.07	0.96 MB	17×

Table 3: The model size and Average Precision (AP) with bounding box (bb) and mask (mk) are compared.

Methods	Bits (W/A)	AP^{bb}	AP^{mk}	Size	Ratio
Baseline	32/32	37.9	34.6	170MB	1×
HQ (Ours)	2/16	35.0	31.7	**4.92MB**	**34×**
ABGD (2020)	32/32	33.9	30.8	6.6MB	26×
PQF (2021)	32/32	36.3	33.5	6.6MB	26×
BRECQ (2021)	2/8	34.23	-	-	-

Table 4: Size-accuracy results of ResNet-18 on ImageNet.

Epoch	Accuracy (%)	Sparsity (%)	Size
19	65.64	74.06	1.50MB
99	65.97	75.77	1.40MB
179	67.03	78.96	1.30MB
259	66.66	81.86	1.20MB
299	66.23	84.67	1.10MB
389	65.37	87.26	0.94MB

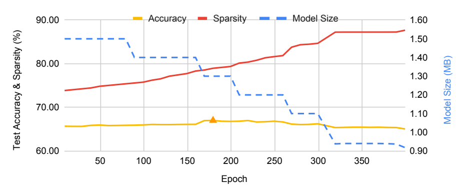


Figure 5: The trend lines of size-accuracy and sparsity of ResNet-18. The blue dashed line denotes the model size.

6. Ablation Study

We experiment with the criteria including different pruning settings, reinitialization, and hyperspherical learning. “**Hyper**” means hyperspherical training. “**Pruning+Reinit**” means pruning with reinitialization. “**Baseline/BL**” means the pre-trained model from PyTorch or Detectron[76]. Experiments in the Section 6.2 and 6.3 apply full-precision to the last FC layer.

6.1. Hyperspherical Pruning and Reinitialization

We study the change of the cosine distance $\mathcal{D}$ (Eq. (8)) brought by our proposed method: applying hyperspherical preprocessing method prior to ternary quantization.

Table 5 shows that pruning can reduce $\mathcal{D}$ with or without hyperspherical learning and $\mathcal{D}$ tends to decrease along

Table 5: The cosine distance $\mathcal{D}$ of convolutional layers of ResNet-50 on ImageNet.

Methods	$\mathcal{D}$	Accuracy	Sparsity
Baseline	0.262	76.15	0.0
Hyper	0.288	76.18	0.0
Baseline+Pruning	0.187	76.06	0.4
Hyper+Pruning	0.155	76.99	0.4
Hyper+Pruning+Reinit	**0.068**	77.04	0.4
Baseline+Pruning	0.149	76.09	0.6
Hyper+Pruning+Reinit	**0.056**	77.03	0.6

Table 6: Distance comparison of convolutional layers on the object detection task.

Methods	$\mathcal{D}$	$\mathbf{AP}^{bb}$	$\mathbf{AP}^{mk}$	Sparsity
Baseline	0.263	41.0	37.2	0.0
Hyper	0.246	41.03	37.54	0.0
Hyper+Pruning	0.193	41.33	37.94	0.8
Hyper+Pruning+Reinit$_1$	**0.186**	40.92	37.33	0.8
Hyper+Pruning+Reinit$_{10}$	**0.113**	40.31	36.86	0.8

Table 7: Quantization accuracy of ResNet-18 on ImageNet.

Initial Models (Accuracy)	Hyper	Non-Hyper
Baseline (69.76)	66.45	60.46
BL+Pruning+Reinit (69.63)	67.17	66.11
Hyper+BL+Pruning+Reinit (69.67)	67.50	65.24

with the growing sparsity. "**Pruning+Reinit**" significantly enlarges the distance gap with the pruning-only results and can improve model's performance [81, 20].

Table 6 shows the difference between applying reinitialization once (**Reinit**$_1$) and 10 times (**Reinit**$_{10}$) to the target sparsity of 0.8. "**Reinit**$_{10}$" has a smaller average distance than "**Reinit**$_1$", indicating that iterative pruning and reinitialization encourage model weights close to its ternary version (Section 4.1, Fig. 2b).

6.2. Hyperspherical Ternary Quantization

We perform ternary quantization on ResNet-18 to examine the impact of hyperspherical learning and "**Pruning+Reinit**". The models are initialized by three different pre-trained weights (Table 7). The initialized models are quantized with hyperspherical learning and regular training ("**Non-Hyper**") by 100 epochs. Table 7 shows significant improvements brought by hyperspherical learning and "**Pruning+Reinit**". The figure of trend lines can be found in the Appendix.

6.3. The impact of Pruning Settings

We further study the impact of different pruning ranges (r) and step sizes (δ, line 14 of Algorithm 1) on ResNet18. The Figure 5 of TTQ [83] indicates that a proper pruning range before ternary quantization is from 0.3 to 0.7, and we take that as a reference. During preprocessing, r changes from 0.3 to 0.7 or 0.4 to 0.8. The step sizes can be 0.01, 0.02, or controlled by cosine annealing. The "**Pruning+Reinit**" starts at the 20-th epoch. The total training epochs are 100. Fig. 6 shows that pruned models with larger step sizes and ratios perform poorly and take longer to recover. Using cosine annealing [51] method to adjust the step size δ improves the overall performance. Fig. 7 shows the following ternary quantization results. Cosine annealing accelerates the convergence.

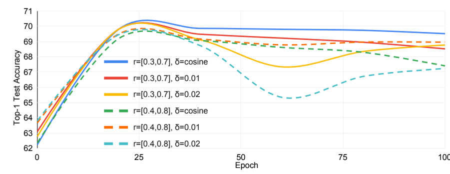


Figure 6: The accuracy trends of ResNet-18 with different pruning settings during preprocessing.

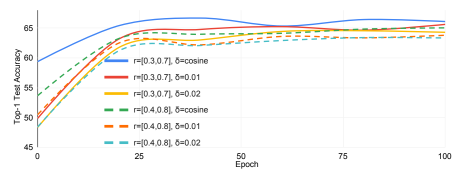


Figure 7: The following ternary quantization accuracy of ResNet-18 after preprocessing.

7. Conclusion

We propose a novel method, Hyperspherical Quantization, to construct sparse ternary weights by unifying pruning, reinitialization and ternary quantization on the hypersphere. The proposed iterative pruning and reinitialization strategy greatly outperform state-of-the-art model compression results in terms of size-accuracy trade-offs. A major contribution of our method is the use of hyperspherical learning to enhance the compression capability. Our work further reveals and demonstrates that pruning and quantization are linked through hypersphere. Our work also explores a new way to extremely compress DNN models without using clustering. Future work may combine our method with ternary activation quantization [6, 39] to further speed up the inference.

References

[1] Jose M Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, pages 856–867, 2017.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic

neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.

[4] Miguel A Carreira-Perpinán and Yerlan Idelbayev. Model compression as constrained optimization, with application to neural nets. part ii: Quantization. *arXiv preprint arXiv:1707.04319*, 2017.

[5] Beidi Chen, Weiyang Liu, Zhiding Yu, Jan Kautz, Anshumali Shrivastava, Animesh Garg, and Anima Anandkumar. Angular visual hardness. In *International Conference on Machine Learning*, 2020.

[6] Peng Chen, Bohan Zhuang, and Chunhua Shen. Fatnn: Fast and accurate ternary neural networks, 2020.

[7] Minsik Cho, Keivan A Vahid, Saurabh Adya, and Mohammad Rastegari. Dkm: Differentiable k-means clustering layer for neural network compression. *International Conference on Learning Representations*, 2022.

[8] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *International Conference on Learning Representations*, 2018.

[9] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *International Conference on Learning Representations*, 2017.

[10] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[11] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *Advances in Neural Information Processing Systems*, 2016.

[12] Steve Dai, Rangha Venkatesan, Mark Ren, Brian Zimmer, William Dally, and Brucek Khailany. Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. *Proceedings of Machine Learning and Systems*, 3:873–884, 2021.

[13] Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. In *Uncertainty in Artificial Intelligence*, 2018.

[14] Hassan Dbouk, Hetul Sanghvi, Mahesh Mehendale, and Naresh Shanbhag. Dbq: A differentiable branch quantizer for lightweight deep neural networks. In *European Conference on Computer Vision*, pages 90–106. Springer, 2020.

[15] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[16] Google Developers. Regularization for simplicity: Lambda. `https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/lambda`, 2020.

[17] Zhen Dong, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision, 2019.

[18] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *International Conference on Learning Representations*, 2020.

[19] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. *International Conference on Learning Representations*, 2021.

[20] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations*, 2019.

[21] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):744–755, 2013.

[22] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv:2103.13630*, 2021.

[23] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *International Conference on Learning Representations*, 2015.

[24] Dibakar Gope, Jesse Beu, Urmish Thakker, and Matthew Mattina. Ternary mobilenets via per-layer hybrid filter banks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 708–709, 2020.

[25] Kaiyuan Guo, Song Han, Song Yao, Yu Wang, Yuan Xie, and Huazhong Yang. Software-hardware codesign for efficient neural network acceleration. *IEEE Micro*, 37(2):18–25, 2017.

[26] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[27] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, 1:177–185, 1988.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[31] Geoffrey E Hinton et al. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[32] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *Computer Vision and Pattern Recognition*, 2016.

[33] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[34] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*, 2017.

[35] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[36] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[37] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *In International (NIPS) Workshop on EMDNN*, 2016.

[38] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations*, 2017.

[39] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2019.

[40] Yuhang Li, Xin Dong, Sai Qian Zhang, Haoli Bai, Yuanpeng Chen, and Wei Wang. Rtn: Reparameterized ternary network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4780–4787, 2020.

[41] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *International Conference on Learning Representations*, 2021.

[42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[43] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in neural information processing systems*, pages 345–353, 2017.

[44] Weiyang Liu, Rongmei Lin, Zhen Liu, James M. Rehg, Li Xiong, Adrian Weller, and Le Song. Orthogonal over-parameterized training. In *Computer Vision and Pattern Recognition*, 2021.

[45] Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Schölkopf, and Adrian Weller. Learning with hyperspherical uniformity. In *International Conference On Artificial Intelligence and Statistics*, pages 1180–1188. PMLR, 2021.

[46] Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M Rehg, and Le Song. Decoupled networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2771–2779, 2018.

[47] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[48] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, page 7, 2016.

[49] Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. Deep hyperspherical learning. *Advances in neural information processing systems*, 30, 2017.

[50] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *International Conference on Learning Representations*, 2018.

[51] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations*, 2017.

[52] Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. *International Conference on Learning Representations*, 2019.

[53] Julieta Martinez, Jashan Shewakramani, Ting Wei Liu, Ioan Andrei Bârsan, Wenyuan Zeng, and Raquel Urtasun. Permute, quantize, and fine-tune: Efficient compression of neural networks. *Computer Vision and Pattern Recognition*, 2021.

[54] Mark D McDonnell. Training wide residual networks for deployment using a single bit for each weight. *International Conference on Learning Representations*, 2018.

[55] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *International Conference on Machine Learning*, 2017.

[56] Michael C Mozer and Paul Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1(1):3–16, 1989.

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

[58] Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992.

[59] Sung Woo Park and Junseok Kwon. Sphere generative adversarial network based on geometric moment matching. In *Computer Vision and Pattern Recognition*, 2019.

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

[61] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy,

Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[62] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, pages 901–909, 2016.

[63] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[64] Oran Shayer, Dan Levi, and Ethan Fetaya. Learning discrete weights using the local reparameterization trick. *International Conference on Learning Representations*, 2018.

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

[66] Sanghyun Son, Seungjun Nah, and Kyoung Mu Lee. Clustering convolutional kernels to compress deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 216–232, 2018.

[67] Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou. And the bit goes down: Revisiting the quantization of neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[68] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488*, 2015.

[69] Tensorflow. Pruning in keras example. `https://www.tensorflow.org/model_optimization/guide/pruning/pruning_with_keras`, 2022.

[70] Jan Van Leeuwen. On the construction of huffman trees. In *ICALP*, pages 382–410, 1976.

[71] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. 2011.

[72] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018.

[73] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

[74] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

[75] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[76] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[77] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021.

[78] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *International Conference on Learning Representations*, 2019.

[79] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018.

[80] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *International Conference on Learning Representations*, 2017.

[81] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32, 2019.

[82] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.

[83] Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. *International Conference on Learning Representations*, 2017.