Rate Distortion For Model Compression: From Theory To Practice

Weihao Gao[*], Yu-Han Liu[†], Chong Wang[†], Sewoong Oh[‡]

January 25, 2019

Abstract

The enormous size of modern deep neural networks makes it challenging to deploy those models in memory and communication limited scenarios. Thus, compressing a trained model without a significant loss in performance has become an increasingly important task. Tremendous advances has been made recently, where the main technical building blocks are parameter pruning, parameter sharing (quantization), and low-rank factorization. In this paper, we propose principled approaches to improve upon the common heuristics used in those building blocks, namely pruning and quantization.

We first study the fundamental limit for model compression via the rate distortion theory. We bring the rate distortion function from data compression to model compression to quantify this fundamental limit. We prove a lower bound for the rate distortion function and prove its achievability for linear models. Although this achievable compression scheme is intractable in practice, this analysis motivates a novel model compression framework. This framework provides a new objective function in model compression, which can be applied together with other classes of model compressor such as pruning or quantization. Theoretically, we prove that the proposed scheme is optimal for compressing one-hidden-layer ReLU neural networks. Empirically, we show that the proposed scheme improves upon the baseline in the compression-accuracy tradeoff.

1 Introduction

Deep neural networks have been successful, for example, in the application of computer vision (Krizhevsky et al., 2012), machine translation (Wu et al., 2016) and game playing (Silver et al., 2017). With increasing data and computational power, the number of weights in practical neural network model also grows rapidly. For example, in the application of image recognition, the LeNet-5 model (LeCun et al., 1998) only has 400K weights. After two decades, AlexNet (Krizhevsky et al., 2012) has more than 60M weights, and VGG-16 net (Simonyan and Zisserman, 2014) has more than 130M weights. Coates et al. (2013) even tried a neural network with 11B weights. The huge size of neural networks brings many challenges, including large storage, difficulty in training, and large energy consumption. In particular, deploying such extreme models to embedded mobile systems is not feasible.

Several approaches have been proposed to reduce the size of large neural networks while preserving the performance as much as possible. Most of those approaches fall into one of the two broad categories. The first category designs novel network structures with small number of parameters, such as SqueezeNet Iandola et al. (2016) and MobileNet Howard et al. (2017). The other category directly compresses a given large neural network using pruning, quantization, and matrix factorization, including LeCun et al. (1990); Hassibi and Stork (1993); Han et al. (2015b,a); Cheng et al. (2015). There are also advanced methods to train the neural network using Bayesian methods to help pruning or quantization at a later stage, such as Ullrich et al. (2017); Louizos et al. (2017); Federici et al. (2017).

[*] Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign. Email: wgao9@illinois.edu, work done as an intern in Google Inc.

[†] Google Inc. Email: {yuhanliu, chongw}@google.com

[‡] Allen School of Computer Science and Engineering. Email: sewoong@cs.washington.edu

As more and more model compression algorithms are proposed and compression ratio becomes larger and larger, it motivates us to think about the fundamental question — How well can we do for model compression? The goal of model compression is to trade off the *number of bits* used to describe the model parameters, and the *distortion* between the compressed model and original model. We wonder *at least* how many bits is needed to achieve certain distortion? Despite many successful model compression algorithms, these theoretical questions still remain unclear.

In this paper, we fill in this gap by bringing tools from rate distortion theory to identify the fundamental limit on how much a model can be compressed. Specifically, we focus on compression of a pretrained model, rather than designing new structures or retraining models. Our approach builds upon rate-distortion theory introduced by Shannon (1959) and further developed by Berger (1971). The approach also connects to modeling neural networks as random variables in Mandt et al. (2017), which has many practical usages (Cao et al., 2018).

Our contribution for model compression is twofold: theoretical and practical. We first apply theoretical tools from rate distortion theory to provide a lower bound on the fundamental trade-off between *rate* (number of bits to describe the model) and *distortion* between compressed and original models, and prove the tightness of the lower bound for a linear model. This analysis seamlessly incorporate the structure of the neural network architecture into model compression via backpropagation. Motivated by the theory, we design an improved objective for compression algorithms and show that the improved objective gives optimal pruning and quantization algorithm for one-hidden-layer ReLU neural network, and has better performance in real neural networks as well.

The rest of the paper is organized as follows.

- In Section 2, we briefly review some previous work on model compression.
- In Section 3, we introduce the background of the rate distortion theory for data compression, and formally state the rate distortion theory for model compression.
- In Section 4, we give a lower bound of the rate distortion function, which quantifies the fundamental limit for model compression. We then prove that the lower bound is achievable for linear model.
- In Section 5, motivated by the achievable compressor for linear model, we proposed an improved objective for model compression, which takes consideration of the sturcture of the neural network. We then prove that the improved objective gives optimal compressor for one-hidden-layer ReLU neural network.
- In Section 6, we demonstrate the empirical performance of the proposed objective on fully-connected neural networks on MNIST dataset and convolutional networks on CIFAR dataset.

2 Related work on model compression

The study of model compression of neural networks appeared as long as neural network was invented. Here we mainly discuss the literature on directly compressing large models, which are more relevant to our work. They usually contain three types of methods — pruning, quantization and matrix factorization.

Pruning methods set unimportant weights to zero to reduce the number of parameters. Early works of model pruning includes biased weight decay (Hanson and Pratt, 1989), optimal brain damage (LeCun et al., 1990) and optimal brain surgeon (Hassibi and Stork, 1993). Early methods utilize the Hessian matrix of the loss function to prune the weights, however, Hessian matrix is inefficient to compute for modern large neural networks with millions of parameters. More recently, Han et al. (2015b) proposed an iterative pruning and retraining algorithm that works for large neural networks.

Quantization, or weight sharing methods group the weights into clusters and use one value to represent the weights in the same group. This category includes fixed-point quantization by Vanhoucke et al. (2011), vector quantization by Gong et al. (2014), HashedNets by Chen et al. (2015), Hessian-weighted quantizaiton by Choi et al. (2016).

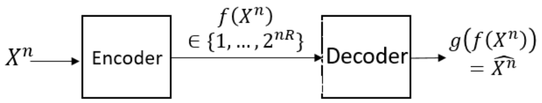

Figure 1: An illustration of encoder and decoder.

Matrix factorization assumes the weight matrix in each layer could be factored as a low rank matrix plus a sparse matrix. Hence, storing low rank and sparse matrices is cheaper than storing the whole matrix. This category includes Denton et al. (2014) and Cheng et al. (2015).

There are some recent advanced method beyond pruning, quantization and matrix factorization. Han et al. (2015a) assembles pruning, quantization and Huffman coding to achieve better compression rate. Bayesian methods Ullrich et al. (2017); Louizos et al. (2017); Federici et al. (2017) are also used to retrain the model such that the model has more space to be compressed. He et al. (2018) uses reinforcement learning to design a compression algorithm.

Despite these aforementioned works for model compression, no one has studied the fundamental limit of model compression, as far as we know. More specifically, in this paper, we focus on the study of theory of model compression for pretrained neural network models and then derive practical compression algorithms given the proposed theory.

3 Rate distortion theory for model compression

In this section, we briefly introduce the rate distortion theory for data compression. Then we extend the theory to compression of model parameters.

3.1 Review of rate distortion theory for data compression

Rate distortion theory, firstly introduced by Shannon (1959) and further developed by Berger (1971), is an important concept in information theory which gives theoretical description of lossy data compression. It addressed the minimum average number of R bits, to transmit a random variable such that the receiver can reconstruct the random variable with distortion D.

Precisely, let $X^n = \{X_1, X_2 \ldots X_n\} \in \mathcal{X}^n$ be i.i.d. random variables from distribution P_X. An encoder $f_n : \mathcal{X}^n \to \{1, 2, \ldots, 2^{nR}\}$ maps the message X^n into codeword, and a decoder $g_n : \{1, 2, \ldots, 2^{nR}\} \to \mathcal{X}^n$ reconstruct the message by an estimate $\hat{X}^n$ from the codeword. See Figure 1 for an illustration.

A distortion function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ quantifies the difference of the original and reconstructed message. Distortion between sequence X^n and $\hat{X}^n$ is defined as the average distortion of X_i's and $\hat{X}_i$'s. Commonly used distortion function includes Hamming distortion function $d(x, \hat{x}) = \mathbb{1}[x \neq \hat{x}]$ for $\mathcal{X} = \{0, 1\}$ and square distortion function $d(x, \hat{x}) = (x - \hat{x})^2$ for $\mathcal{X} = \mathbb{R}$.

Now we are ready to define the rate-distortion function for data compression.

Definition 1 *A rate-distortion pair (R, D) is* achievable *if there exists a series of (probabilistic) encoder-decoder (f_n, g_n) such that the alphabet of codeword has size 2^{nR} and the expected distortion $\lim_{n\to\infty} \mathbb{E}[d(X^n, g_n(f_n(X^n)))] \leq D$.*

Definition 2 *Rate-distortion function $R(D)$ equals to the infimum of rate R such that rate-distortion pair (R, D) is achievable.*

The main theorem of rate-distortion theory (Cover and Thomas (2012, Theorem 10.2.1)) states as follows,

Theorem 1 *Rate distortion theorem for data compression.*

$$R(D) \quad = \quad \min_{P_{\hat{X}|X}:\mathbb{E}[d(X,\hat{X})]\leq D} I(X;\hat{X}) \ . \tag{1}$$

The rate distortion quantifies the fundamental limit of data compression, i.e., *at least* how many bits are needed to compress the data, given the quality of the reconstructed data. Here is an example for rate-distortion function.

Example 1 *If $X \sim \mathcal{N}(0,\sigma^2)$, the rate distortion function is given by*

$$R(D) \quad = \quad \begin{cases} \frac{1}{2}\log_2(\sigma^2/D) & \text{if } D \leq \sigma^2 \\ 0 & \text{if } D > \sigma^2 \end{cases} \ .$$

If the required distortion D is larger than the variance of the Gaussian variable σ^2, we simply transmit $\hat{X} = 0$; otherwise, we will transmit $\hat{X}$ such that $\hat{X} \sim \mathcal{N}(0, \sigma^2 - D)$, $X - \hat{X} \sim \mathcal{N}(0, D)$ where $\hat{X}$ and $X - \hat{X}$ are independent.

3.2 Rate distortion theory for model compression

Now we extend the rate distortion theory for data compression to model compression. To apply the rate distortion theory to model compression, we view the weights in the model as a multi-dimensional random variable $W \in \mathbb{R}^m$ following distribution P_W. The randomness comes from multiple sources including different distributions of training data, randomness of training data and randomness of training algorithm. The compressor can also be random hence we describe the compressor by a conditional probability $P_{\hat{W}|W}$. Now we define the distortion and rate in model compression, analogously to the data compression scenario.

Distortion. Assume we have a neural network f_w that maps input $x \in \mathbb{R}^{d_x}$ to $f_w(x)$ in output space $\mathcal{S}$. For regressors, $f_w(x)$ is defined as the output of the neural network on $\mathbb{R}^{d_y}$. Analogous to the square distortion in data compression, We define the distortion to be the expected ℓ_2 distance between f_w and $f_{\hat{w}}$, i.e.

$$d(w,\hat{w}) \equiv \mathbb{E}_X\left[\|f_w(X) - f_{\hat{w}}(X)\|_2^2\right] \ . \tag{2}$$

For classfiers, $f_w(x)$ is defined as the output probability distribution over C classes on the simplex Δ^{C-1}. We define the distortion to be the expected distance between f_w and $f_{\hat{w}}$, i.e.

$$d(w,\hat{w}) \equiv \mathbb{E}_X\left[D(f_{\hat{w}}(X)||f_w(X))\right] \ . \tag{3}$$

Here D could be any statistical distance, including KL divergence, Hellinger distance, total variation distance, etc. Such a definition of distortion captures the difference between the original model and the compressed model, averaged over data X, and measures the quality of a compression algorithm.

Rate. In data compression, the rate is defined as the description length of the bits necessary to communicate the compressed data $\hat{X}$. The compressor outputs $\hat{X}$ from a finite *code book* $\mathcal{X}$. The description consists the *code word* which are the indices of $\hat{x}$ in the code book, and the description of the *code book*.

In rate distortion theory, we ignore the code book length. Since we are transmitting a sequence of data X^n, the code word has to be transmitted for each X_i but the code book is only transmitted once. In asymptotic setting, the description length of code book can be ignored, and the rate is defined as the description length of the code word.

In model compression, we also define the rate as the code word length, by assuming that an underlying distribution P_W of the parameters exists and infinitely many models whose parameters are i.i.d. from P_W will be compressed. In practice, we only compress the parameters once so there is no distribution of the parameters. Nevertheless, the rate distortion theory can also provide important intuitions for one-time compression, explained in Section 5.

Now we can define the rate distortion function for model compression. Analogously to Theorem 1, the rate distortion function for model compression is defined as follows,

Definition 3 *Rate distortion function for model compression.*

$$R(D) = \min_{P_{\hat{W}|W}:\mathbb{E}_{W,\hat{W}}\left[d(W,\hat{W})\right]\leq D} I(W;\hat{W}). \tag{4}$$

In the following sections we establish a lower bound of the rate-distortion function.

4 Lower bound and achievability for rate distortion function

In this section, we study the lower bound for rate distortion function in Definition 3. We provide a lower bound for the rate distortion function, and prove that this lower bound is achivable for linear regression models.

4.1 Lower bound for linear model

Assume that we are going to compress a linear regression model $f_w(x) = w^T x$. We assume that the mean of data $x \in \mathbb{R}^m$ is zero and the covariance matrix is diagonal, i.e., $\mathbb{E}_X[X_i^2] = \lambda_{x,i} > 0$ and $\mathbb{E}_X[X_i X_j] = 0$ for $i \neq j$. Furthermore, assume that the parameters $W \in \mathbb{R}^m$ are drawn from a Gaussian distribution $\mathcal{N}(0, \Sigma_W)$. The following theorem gives the lower bound of the rate distortion function for the linear regression model.

Theorem 2 *The rate-distortion function of the linear regression model $f_w(x) = w^T x$ is lower bounded by*

$$R(D) \geq \underline{R}(D) = \frac{1}{2}\log\det(\Sigma_W) - \sum_{i=1}^{m}\frac{1}{2}\log(D_i),$$

where

$$D_i = \begin{cases} \mu/\lambda_{x,i} & \text{if} \mu < \lambda_{x,i}\mathbb{E}_W[W_i^2]\,, \\ \mathbb{E}_W[W_i^2] & \text{if} \mu \geq \lambda_{x,i}\mathbb{E}_W[W_i^2]\,, \end{cases}$$

where μ is chosen that $\sum_{i=1}^m \lambda_{x,i} D_i = D$.

This lower bound gives rise to a "weighted water-filling" approach, which differs from the classical "water-filling" for rate distortion of colored Gaussian source in Cover and Thomas (2012, Figure 13.7). The details and graphical explanation of the "weighted water-filling" can be found in Appendix A.

4.2 Achievability

We show that, the lower bound give in Theorem 2 is achievable. Precisely, we have the following theorem.

Theorem 3 *There exists a class of probabilistic compressors $P^{(D)}_{\hat{W}^*|W}$ such that $\mathbb{E}_{P_W \circ P^{(D)}_{\hat{W}^*|W}}\left[d(W,\hat{W}^*)\right] = D$ and $I(W;\hat{W}^*) = \underline{R}(D)$.*

The optimal compressor is Algorithm 1 in Appendix A. Intuitively, the optimal compressor does the following

- Find the optimal water levels D_i for "weighted water filling", such that the expected distortion $D = \mathbb{E}_{W,\hat{W}}[d(W,\hat{W})] = \mathbb{E}_{W,\hat{W}}[\hat{W}^T\Sigma_X(W-\hat{W})]$ is minimized given certain rate.

- Add a noise Z_i which is independent of $\hat{W}_i = W_i + Z_i$ and has a variance proportional to the water level. That is possible since W is Gaussian.

We can check that the compressor makes all the inequalities become equality, hence achieve the lower bound. The full proof of the lower bound and achievability can be found in Appendix A.

5 Improved objective for model compression

In the previous sections, we study the rate-distortion theory for model compression. In rate-distortion theory, we assume that there exists a prior distribution P_W on the weights W, and prove the tightness of the lower bound in the asymptotic scenario. However, in practice, we only compress one particular pre-trained model, so there are no prior distribution of W. Nonetheless, we can still learn something important from the achivability of the lower bound, by extracting two "golden rules" from the optimal algorithm for linear regression.

5.1 Two golden rules

Recall that for linear regression model, to achieve the smallest rate given certain distortion (or, equivalently, achieve the smallest distortion given certain rate), the optimal compressor need to do the following: (1) find appropriate "water levels" such that the expected distortion $E_{W,\hat{W}}[d(W,\hat{W})] = \mathbb{E}_{W,\hat{W},X}[(W^TX - \hat{W}^TX)^2] = \mathbb{E}_{W,\hat{W}}[(W-\hat{W})^T\Sigma_X(W-\hat{W})]$ is minimized. (2) make sure that $\hat{W}_i$ is independent with $W_i - \hat{W}_i$, in other words, $\mathbb{E}_{W,\hat{W}}[\hat{W}^T\Sigma_X(W-\hat{W})] = 0$. Hence, we extract the following two "golden rules":

1. $\mathbb{E}_{W,\hat{W}}[\hat{W}^T\Sigma_X(W-\hat{W})] = 0$
2. $\mathbb{E}_{W,\hat{W}}[(W-\hat{W})^T\Sigma_X(W-\hat{W})]$ should be minimized, given certain rate.

For practical model compression, we adopt these two "golden rules", by making the following amendments. First, we discard the expectation over W and $\hat{W}$ since there is only one model to be compressed. Second, the distortion can be written as $d(w,\hat{w}) = (w-\hat{w})^T\Sigma_X(w-\hat{w})$ only for linear models. For non-linear models, the distortion function is complicated, but can be approximated by a simpler formula. For non-linear regression models, we take first order Taylor expansion of the function $f_{\hat{w}}(x) \approx f_w(x) + (\hat{w}-w)^T\nabla_w f_w(x)$, and have

$$\begin{aligned} d(w,\hat{w}) &= \mathbb{E}_X\left[\|f_w(X) - f_{\hat{w}}(X)\|_2^2\right] \\ &\approx \mathbb{E}_X\left[(w-\hat{w})^T\nabla_w f_w(X)(\nabla_w f_w(X))^T(w-\hat{w})\right] \\ &= (w-\hat{w})^T I_w (w-\hat{w}) \end{aligned}$$

where the "weight importance matrix" defined as

$$I_w = \mathbb{E}_X\left[\nabla_w f_w(X)(\nabla_w f_w(X))^T\right], \tag{5}$$

quantifies the relative importance of each weight to the output. For linear regression models, weight importance matrix I_w equals to Σ_X.

For classification models, we will first approximate the KL divergence. Using the Taylor expansion $x\log(x/a) \approx (x-a) + (x-a)^2/(2a)$ for $x/a \approx 1$, the KL divergence $D_{KL}(P||Q)$ for can be approximated by $D_{KL}(P||Q) \approx \sum_i (P_i - Q_i) + (P_i - Q_i)^2/(2P_i) = \sum_i (P_i - Q_i)^2/(2P_i)$, or in vector form $D_{KL}(P||Q) \approx \frac{1}{2}(P-Q)^T\mathrm{diag}[P^{-1}](P-Q)$. Therefore,

$$\begin{aligned} d(w,\hat{w}) &= \mathbb{E}_X\left[D_{KL}(f_{\hat{w}}(X)||f_w(X))\right] \\ &\approx \frac{1}{2}\mathbb{E}_X\left[(f_w(X) - f_{\hat{w}}(X))^T\mathrm{diag}[f_w^{-1}(X)](f_w(X) - f_{\hat{w}}(X))\right] \\ &\approx \frac{1}{2}\mathbb{E}_X\left[(w-\hat{w})^T(\nabla_w f_w(X))\mathrm{diag}[f_w^{-1}(X)](\nabla_w f_w(X))^T(w-\hat{w})\right]. \end{aligned}$$

So the weight importance matrix is given by

$$I_w = \mathbb{E}_X \left[(\nabla_w f_w(X)) \mathrm{diag}[f_w^{-1}(X)] (\nabla_w f_w(X))^T \right]. \tag{6}$$

This weight importance matrix is also valid for many other statistical distances, including reverse KL divergence, Hellinger distance and Jenson-Shannon distance.

Now we define the two "golden rules" for practical model compression algorithms,

1. $\hat{w}^T I_w (w - \hat{w}) = 0$,
2. $(w - \hat{w})^T I_w (w - \hat{w})$ is minimized given certain constraints.

In the following subsection we will show the optimality of the "golden rules" for a one-hidden-layer neural network.

5.2 Optimality for one-hidden-layer ReLU network

We show that if a compressor of a one-hidden-layer ReLU network satisfies the two "golden rules", it will be the optimal compressor, with respect to mean-square-error. Precisely, consider the one-hidden layer ReLU neural network $f_w(x) = ReLU(w^T x)$, where the distribution of input $x \in \mathbb{R}^m$ is $\mathcal{N}(0, \Sigma_X)$. Furthermore, we assume that the covariance matrix $\Sigma_X = \mathrm{diag}[\lambda_{x,1}, \dots, \lambda_{x,m}]$ is diagonal and $\lambda_{x,i} > 0$ for all i. We have the following theorem.

Theorem 4 *If compressed weight $\hat{w}^*$ satisfies $\hat{w}^* I_w (\hat{w}^* - w) = 0$ and*

$$\hat{w}^* = \arg\min_{\hat{w} \in \hat{\mathcal{W}}} (w - \hat{w})^T I_w (w - \hat{w}),$$

where $\hat{\mathcal{W}}$ is some class of compressors, then

$$\hat{w}^* = \arg\min_{\hat{w} \in \hat{\mathcal{W}}} \mathbb{E}_X \left[(f_w(X) - f_{\hat{w}}(X))^2 \right].$$

The proof uses the techniques of Hermite polynomials and Fourier analysis on Gaussian spaces, inspired by Ge et al. (2017). The full proof can be found in Appendix B.

Here $\hat{\mathcal{W}}$ denotes a class of compressors, with some constraints. For example, $\hat{\mathcal{W}}$ could be the class of pruning algorithms where no more than 50% weights are pruned, or $\hat{\mathcal{W}}$ could be the class of quantization algorithm where each weight is quantized to 4 bits. Theoretically, it is not guaranteed that the two "golden rules" can be satisfied simultaneously for every $\hat{\mathcal{W}}$, but in the following subsection we show that they can be satisfied simultaneously for two of the most commonly used class of compressors — pruning and quantization. Hence, minimizing the objective $(w - \hat{w})^T I_w (w - \hat{w})$ will be optimal for pruning and quantization.

5.3 Improved objective for pruning and quantization

Pruning and quantization are two most basic and useful building blocks of modern model compression algorithms, For example, DeepCompress Han et al. (2015a) iteratively prune, retrain and quantize the neural network and achieve state-of-the-art performances on large neural networks.

In pruning algorithms, we choose a subset $S \in [m]$ and set $\hat{w}_i = 0$ for all $i \in S$ and $\hat{w}_i = w_i$ for $i \notin S$. The compression ratio is evaluated by the proportion of unpruned weights $r = (m - |S|)/m$. Since either $\hat{w}_i$ or $w_i - \hat{w}_i$ is zero, so the first "golden rule" is automatically satisfied, so we have the following corollary.

Corollary 1 *For any fixed r, let*

$$\hat{w}_r^* = \arg\min_{S: \frac{d-|S|}{d} = r} (w - \hat{w})^T I_w (w - \hat{w}),$$

Then

$$\hat{w}_r^* = \arg\min_{S:\frac{d-|S|}{d}=r} \mathbb{E}_X\left[(f_w(X)-f_{\hat{w}}(X))^2\right].$$

In quantization algorithms, we cluster the weights into k centroids $\{c_1,\ldots,c_k\}$. The algorithm optimize the centroids as long as the assignments of each weight $A_i \in [k]$. The final compressed weight is given by $\hat{w}_i = c_{A_i}$. Usually k-means algorithm are utilized to minimize the centroids and assignments alternatively. The compression ratio of quantization algorithm is given by

$$r = \frac{mb}{m\sum_{j=1}^k \frac{m_j}{m}\lceil\log_2\frac{m}{m_j}\rceil + kb},$$

where m is the number of weights and b is the number of bits to represent one weight before quantization (usually 32). By using Huffman coding, the average number of bits for each weight is given by $\sum_{j=1}^k (m_j/m)\lceil\log_2(m/m_j)\rceil$, where m_j is the number of weights assigned to the j-th cluster.

If we can find the optimal quantization algorithm with respect to $(w-\hat{w})^T I_w(w-\hat{w})$, then each centroids c_j should be optimal, i.e.

$$0 = \frac{\partial}{\partial c_j}(w-\hat{w})^T I_w(w-\hat{w}) = -2\left(\sum_{i:A_i=j} e_i^T\right) I_w(w-\hat{w})$$

where e_i is the i-th standard basis. Therefore, we have

$$\hat{w}I_w(\hat{w}-w) = \left(\sum_{j=1}^k c_j(\sum_{i:A_i=j} e_i)\right)^T I_w(w-\hat{w}) = \sum_{j=1}^k c_j\left((\sum_{i:A_i=j} e_i^T)I_w(w-\hat{w})\right) = 0.$$

Hence the first "golden rule" is satisfied if the second "golden rule" is satisfied. So we have

Corollary 2 *For any fixed number of centroids k, let*

$$\hat{w}_k^* = \arg\min_{\{c_1,\ldots,c_k\},A\in[k]^m} (w-\hat{w})^T I_w(w-\hat{w}),$$

then

$$\hat{w}_k^* = \arg\min_{\{c_1,\ldots,c_k\},A\in[k]^m} \mathbb{E}_X\left[(f_w(X)-f_{\hat{w}}(X))^2\right].$$

As corollaries of Theorem 4, we proposed to use $(w-\hat{w})^T I_w(w-\hat{w})$ as the objective for pruning and quantization algorithms, which can achieve the minimum MSE for one-hidden-layer ReLU neural network.

6 Experiments

In the previous section, we proved that a pruning or quantization algorithm that minimizes the objective $(w-\hat{w})^T I_w(w-\hat{w})$ also minimizes the MSE loss for one-hidden-layer ReLU neural network. In this section, we show that this objective can also improve pruning and quantization algorithm for larger neural networks on real data.[1]

We test the objectives on the following neural network and datasets.

1. 3-layer fully connected neural network on MNIST.

[1] We leave combinations of pruning, model retraining and quantization like Han et al. (2015a) as future work.

2. Convolutional neural network with 5 convolutional layers and 3 fully connected layers on CIFAR 10 and CIFAR 100.

We load the pretrained models from `https://github.com/aaron-xichen/pytorch-playground`.

In Section 6.1, we use the weight importance matrix for classification in Eq. (6), which is derived by approximating the distortion of KL-divergence. This weight importance matrix does not depend on the training labels, so the induced pruning/quantization algorithms is called "unsupervised compression". Furthermore, if the training labels are available, we treat the loss function $\mathcal{L}_w(X,Y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ as the function to be compressed, and derive several pruning/quantization objectives. The induced pruning/quantization methods are called "supervised compression" and are studied in Section 6.2.

6.1 Unsupervised Compression Experiments

Recall that for classification problems, the weight importance matrix is defined as

$$I_w \quad = \quad \mathbb{E}_X \left[\nabla_w f_w(X) \mathrm{diag}[f_w^{-1}(X)] (\nabla_w f_w(X))^T \right].$$

For computational simplicity, we drop the off-diagonal terms of I_w, and simplify the objective to $\sum_{i=1}^m \mathbb{E}_X[\frac{(\nabla_{w_i} f_w(X))^2}{f_w(X)}](w_i - \hat{w}_i)^2$. To minimize the proposed objective, a pruning algorithm just prune the weights with smaller $\mathbb{E}_X[\frac{(\nabla_{w_i} f_w(X))^2}{f_w(X)}]w_i^2$ greedily. A quantization algorithm uses the weighted k-means algorithm Choi et al. (2016) to find the optimal centroids and assignments. We compare the proposed objective with the baseline objective $\sum_{i=1}^m (w_i - \hat{w}_i)^2$, which were used as building blocks in DeepCompress Han et al. (2015a). We compare the objectives in Table 6.1.

Name	Minimizing objective
Baseline	$\sum_{i=1}^m (w_i - \hat{w}_i)^2$
Proposed	$\sum_{i=1}^m \mathbb{E}_X[\frac{(\nabla_{w_i} f_w(X))^2}{f_w(X)}](w_i - \hat{w}_i)^2$

Table 1: Comparison of unsupervised compression objectives.

For pruning experiment, we choose the same compression rate for every convolutional layer and fully-connected layer, and plot the test accuracy and test cross-entropy loss against compression rate. For quantization experiment, we choose the same number of clusters for every convolutional and fully-connected layer. Also we plot the test accuracy and test cross-entropy loss against compression rate. To reduce the variance of estimating the weight importance matrix I_w, we use the *temperature scaling* method introduced by Guo et al. (2017) to improve model calibration.

We show that results of pruning experiment in Figure 2, and the results of quantization experiment in Figure 3. We can see that the proposed objective gives better validation cross-entropy loss than the baseline, for every different compression ratios. The proposed objective also gives better validation accuracy in most scenarios. We relegate the results for CIFAR100 in Appendix C.

6.2 Supervised Compression Experiments

In the previous experiment, we only use the training data to compute the weight importance matrix. But if we can use the training label as well, we can further improve the performance of pruning and quantization algorithms. If the training label is available, we can view the cross-entropy loss function $\mathcal{L}(f_w(x), y) = \mathcal{L}_w(x,y)$ as a function from $\mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, and define the distortion function as

$$d(w, \hat{w}) \quad = \quad \mathbb{E}_{X,Y} \left[(\mathcal{L}_w(X,Y) - \mathcal{L}_{\hat{w}}(X,Y))^2 \right].$$

Taking first order approximation of the loss function gives the supervised weight importance matrix,

$$I_w \quad = \quad \mathbb{E} \left[\nabla_w \mathcal{L}_w(X,Y) (\nabla_w \mathcal{L}_w(X,Y))^T \right].$$

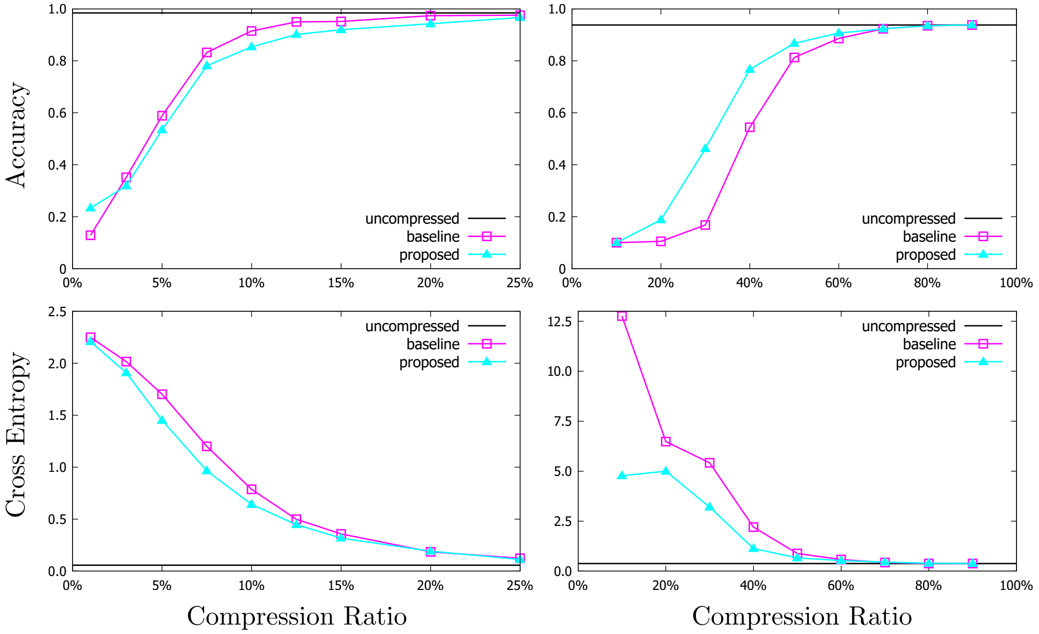

Figure 2: Result for unsupervised pruning experiment. Left: fully-connected neural network on MNIST (Top: test accuracy, Bottom: test cross entropy loss). Right: convolutional neural network on CIFAR10 (Top: test accuracy, Bottom: test cross entropy loss).

We write $\mathbb{E}$ instead of $\mathbb{E}_{X,Y}$ for simplicity. Similarly, we drop the off-diagonal terms for ease of computation, and simplify the objective to $\sum_{i=1}^{m}\mathbb{E}[(\nabla_{w_i}\mathcal{L}_w(X,Y))^2](w_i-\hat{w}_i)^2$, which is called gradient-based objective. Note that for well-trained model, the expected value of gradient $\mathbb{E}[\nabla_w\mathcal{L}_w(X,Y)]$ is closed to zero, but the second moment of the gradient $\mathbb{E}[\nabla_w\mathcal{L}_w(X,Y)(\nabla_w\mathcal{L}_w(X,Y))^T]$ could be large. We compare this objective with the baseline objective $\sum_{i=1}^{m}(w_i-\hat{w}_i)^2$. We also compare with the hessian-based objective $\sum_{i=1}^{m}\mathbb{E}[\nabla^2_{w_i}\mathcal{L}_w(X,Y)](w_i-\hat{w}_i)^2$, which is used in LeCun et al. (1990) and Hassibi and Stork (1993) for network pruning and Choi et al. (2016) for network quantization. To estimate the diagonal entries of the Hessian matrix of the loss function with respect to the model parameters, we implemented Curvature Propagation Martens et al. (2012) treating each layer and activation as a node. The running time is proportional to the running time of the usual gradient back-propagation by a factor that does not depend on the size of the model. Manually optimizing the local Hessian calculation at each node reduces memory usage and allows us to use larger batch size and larger number of samples for more accurate estimates.

Furthermore, if we take second order approximation of the loss function, and drop the off-diagonal terms of the squared gradient matrix and squared hessian tensor, we have the following approximation

$$\begin{aligned} d(w,\hat{w}) &= \mathbb{E}\big[(\mathcal{L}_w(X,Y)-\mathcal{L}_{\hat{w}}(X,Y))^2\big] \\ &\approx \mathbb{E}\big[(\nabla_w\mathcal{L}_w(X,Y)^T(w-\hat{w})+\frac{1}{2}(w-\hat{w})^T\nabla^2_w\mathcal{L}_w(X,Y)(w-\hat{w}))^2\big] \\ &\approx \sum_{i=1}^{m}\mathbb{E}[(\nabla_{w_i}\mathcal{L}_w(X,Y))^2](w_i-\hat{w}_i)^2+\frac{1}{4}\sum_{i=1}^{m}\mathbb{E}[(\nabla^2_{w_i}\mathcal{L}_w(X,Y))^2](w_i-\hat{w}_i)^4, \end{aligned}$$

which is called gradient+hessian based objective. For pruning algorithm, we can prune the weights with smaller $\mathbb{E}[(\nabla_{w_i}\mathcal{L}_w(X,Y))^2]w_i^2+\frac{1}{4}\mathbb{E}[(\nabla^2_{w_i}\mathcal{L}_w(X,Y))^2]w_i^4$ greedily. For quantization algorithm, we use an alternatice minimization algorithm in Appendix C to find the minimum. We conclude the different supervised objectives in Table 6.2.

We show that results of pruning experiment in Figure 4, and the results of quantization experiment in

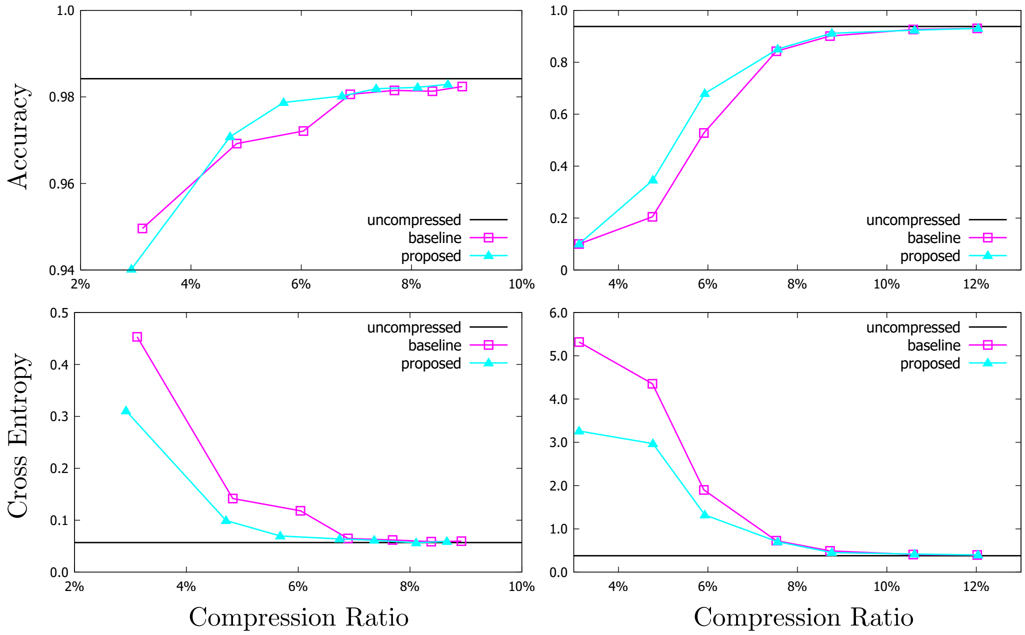

Figure 3: Result for unsupervised quantization experiment. Left: fully-connected neural network on MNIST (Top: test accuracy, Bottom: test cross entropy loss). Right: convolutional neural network on CIFAR10 (Top: test accuracy, Bottom: test cross entropy loss).

Name	Minimizing objective
Baseline	$\sum_{i=1}^{m}(w_i - \hat{w}_i)^2$
Gradient	$\sum_{i=1}^{m} \mathbb{E}[(\nabla_{w_i}\mathcal{L}_w(X,Y))^2](w_i - \hat{w}_i)^2$
Hessian	$\sum_{i=1}^{m} \mathbb{E}[\nabla^2_{w_i}\mathcal{L}_w(X,Y)](w_i - \hat{w}_i)^2$
Gradient + Hessian	$\sum_{i=1}^{m} \mathbb{E}[(\nabla_{w_i}\mathcal{L}_w(X,Y))^2](w_i - \hat{w}_i)^2$ $+\frac{1}{4}\sum_{i=1}^{m} \mathbb{E}[(\nabla^2_{w_i}\mathcal{L}_w(X,Y))^2](w_i - \hat{w}_i)^4$

Table 2: Comparison of supervised compression objectives.

Figure 5. Generally, the gradient objective and hessian objective both give better performance than baseline objective , while gradient objective is slightly than hessian objective at some points. Gradient + hessian objective gives the best overall performance. We relegate the results for CIFAR100 in Appendix C.

Remark. Here we define the supervised distortion function as $d(w,\hat{w}) = \mathbb{E}_{X,Y}\left[(\mathcal{L}_w(X,Y) - \mathcal{L}_{\hat{w}}(X,Y))^2\right]$, analogously to the distortion of regression. However, since the goal of classification is to minimize the loss function, the following definition of distortion function $\tilde{d}(w,\hat{w}) = \mathbb{E}_{X,Y}\left[\mathcal{L}_{\hat{w}}(X,Y) - \mathcal{L}_w(X,Y)\right]$ is also valid and has been adopted in LeCun et al. (1990) and Choi et al. (2016). The main difference is — $d(w,\hat{w})$ focus on the quality of *compression algorithm*, i.e., how similar is the compressed model compared to uncompressed model, whereas $\tilde{d}(w,\hat{w})$ focus on the quality of *compressed model*, i.e. how good is the compressed model. So $d(w,\hat{w})$ is a better criteria for the compression algorithm. Additionally, by taking second order approximation of $d(w,\hat{w})$, we have gradient+hessian objective, which shows better empirical performance than hessian objective, derived by taking second order approximation of $\tilde{d}(w,\hat{w})$.

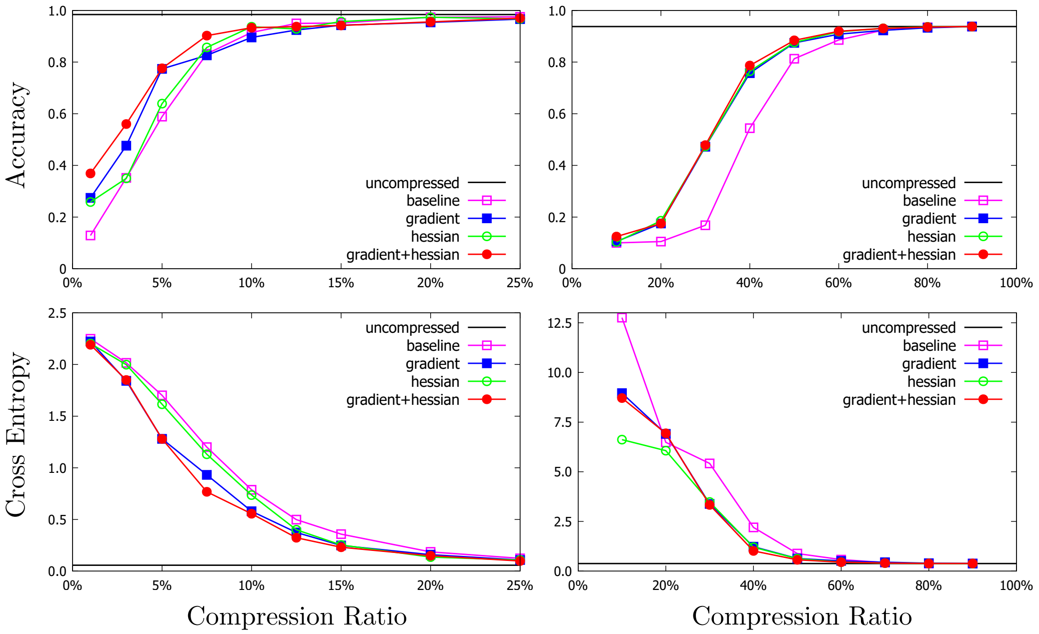

Figure 4: Result for supervised pruning experiment. Left: fully-connected neural network on MNIST (Top: test accuracy, Bottom: test cross entropy loss). Right: convolutional neural network on CIFAR10 (Top: test accuracy, Bottom: test cross entropy loss).

7 Conclusion

In this paper, we investigate the fundamental limit of neural network model compression algorithms. We prove a lower bound for the rate distortion function for model compression, and prove its achievability for linear model. Motivated by the rate distortion function, we propose the weight importance matrtix, and show that for one-hidden-layer ReLU network, pruning and quantization that minimizes the proposed objective is optimal. We also show the superiority of proposed objective in real neural networks.

Acknowledgement

The authors thank Denny Zhou for initial comments and helpful discussions.

References

Berger, T. (1971). Rate distortion theory: A mathematical basis for data compression.

Cao, W., Wang, X., Ming, Z., and Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing*, 275:278–287.

Chen, W., Wilson, J., Tyree, S., Weinberger, K., and Chen, Y. (2015). Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294.

Cheng, Y., Yu, F. X., Feris, R. S., Kumar, S., Choudhary, A., and Chang, S.-F. (2015). An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2857–2865.

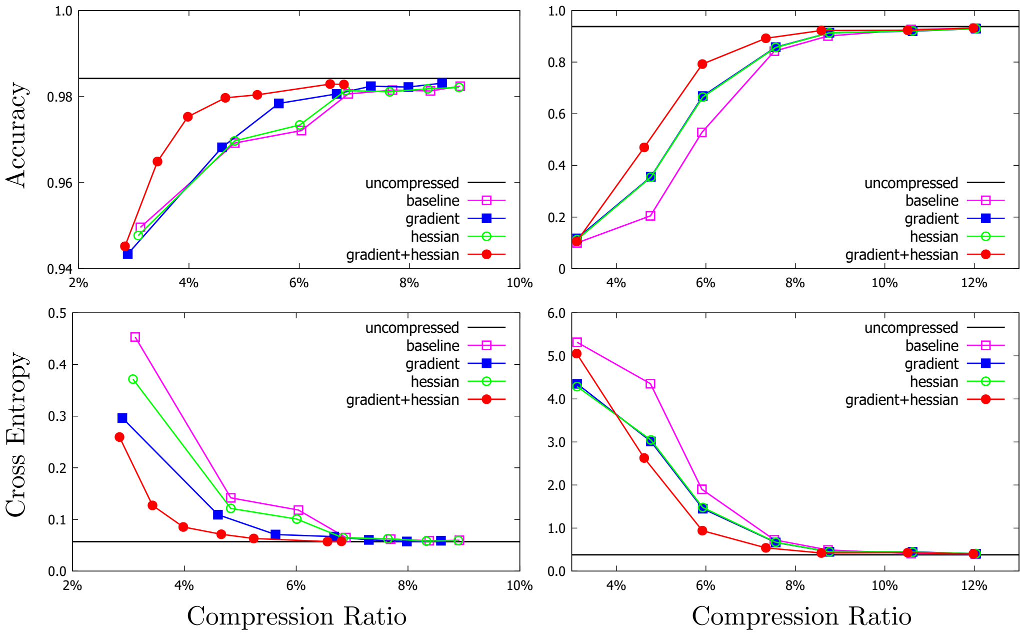

Figure 5: Result for supervised quantization experiment. Left: fully-connected neural network on MNIST (Top: test accuracy, Bottom: test cross entropy loss). Right: convolutional neural network on CIFAR10 (Top: test accuracy, Bottom: test cross entropy loss).

Choi, Y., El-Khamy, M., and Lee, J. (2016). Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*.

Coates, A., Huval, B., Wang, T., Wu, D., Catanzaro, B., and Andrew, N. (2013). Deep learning with cots hpc systems. In *International Conference on Machine Learning*, pages 1337–1345.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277.

Federici, M., Ullrich, K., and Welling, M. (2017). Improved bayesian compression. *arXiv preprint arXiv:1711.06494*.

Ge, R., Lee, J. D., and Ma, T. (2017). Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*.

Gong, Y., Liu, L., Yang, M., and Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Han, S., Mao, H., and Dally, W. J. (2015a). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Han, S., Pool, J., Tran, J., and Dally, W. (2015b). Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143.

Hanson, S. J. and Pratt, L. Y. (1989). Comparing biases for minimal network construction with back-propagation. In *Advances in neural information processing systems*, pages 177–185.

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices.

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

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Jiao, J., Gao, W., and Han, Y. (2017). The nearest neighbor information estimator is adaptively near minimax rate-optimal. *arXiv preprint arXiv:1711.08824*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

Louizos, C., Ullrich, K., and Welling, M. (2017). Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907.

Martens, J., Sutskever, I., and Swersky, K. (2012). Estimating the hessian by back-propagating curvature. *arXiv preprint arXiv:1206.6464*.

McDonald, R. and Schultheiss, P. (1964). Information rates of gaussian signals under criteria constraining the error spectrum. *Proceedings of the IEEE*, 52(4):415–416.

Shannon, C. E. (1959). Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, 4(142-163):1.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Ullrich, K., Meeds, E., and Welling, M. (2017). Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*.

Vanhoucke, V., Senior, A., and Mao, M. Z. (2011). Improving the speed of neural networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, volume 1, page 4. Citeseer.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

A Lower bound for rate distortion function

In this section, we finish the proof of the lower bound and achievability in Section 4. Our approach is based on the water-filling approach McDonald and Schultheiss (1964).

A.1 General lower bound

First, we establish establishes a lower bound of the rate distortion function, which works for general models..

Lemma 1 *The rate-distortion function $R(D) \geq \underline{R}(D) = h(W) - C$, where C is the optimal value of the following optimization problem.*

$$\begin{array}{cl} \max\limits_{P_{\hat{W}|W}} & \sum_{i=1}^{m} \min\left\{h(W_i), \frac{1}{2}\log(2\pi e \mathbb{E}_{W,\hat{W}}[(W_i - \hat{W}_i)^2])\right\} \\ \text{s.t.} & E_{W,\hat{W}}\left[d(W,\hat{W})\right] \leq D. \end{array}$$

where $h(W) = -\int_{w\in\mathcal{W}} P_W(w)\log P_W(w)dw$ is the differential entropy of W and $h(W_i)$ is the differential entropy of the i-th entry of W.

A.1.1 Proof of Lemma 1

Recall that the rate distortion function for model compression is defined as $R(D) = \min_{P_{\hat{W}|W}:\mathbb{E}_{W,\hat{W}}[d(W,\hat{W})]\leq D} I(W;\hat{W})$. Now we lower bound the mutual information $I(W,\hat{W})$ by

$$\begin{aligned} I(W;\hat{W}) &= h(W) - h(W \,|\, \hat{W}), \\ &= h(W) - \sum_{i=1}^{m} h(W_i \,|\, W_1,\ldots,W_{i-1},\hat{W}_i,\ldots,\hat{W}_m) \\ &\geq h(W) - \sum_{i=1}^{m} h(W_i \,|\, \hat{W}_i). \end{aligned}$$

Here the last inequality comes from the fact that conditioning does not increase entropy. Notice that the first term $h(W)$ does not depend on the compressor. For the last term, we upper bound each term $h(W_i \,|\, \hat{W}_i)$ in two ways. On one hand, $h(W_i \,|\, \hat{W}_i)$ is upper bounded by $h(W_i)$ because conditioning does not increase entropy. On the other hand, $h(W_i \,|\, \hat{W}_i) = h(W_i - \hat{W}_i \,|\, \hat{W}_i) \leq h(W_i - \hat{W}_i)$, and by Cover and Thomas (2012, Theorem 8.6.5), differential entropy is maximized by Gaussian distribution, for given second moment. We then have:

$$\begin{aligned} h(W_i \,|\, \hat{W}_i) &\leq \min\left\{h(W_i), h(W_i - \hat{W}_i)\right\} \\ &\leq \min\left\{h(W_i), \frac{1}{2}\log\left(2\pi e \mathbb{E}_{W,\hat{W}}[(W_i - \hat{W}_i)^2]\right)\right\} \\ &= \min\left\{h(W_i), \frac{1}{2}\log(2\pi e \mathbb{E}_{W,\hat{W}}[(W_i - \hat{W}_i)^2])\right\}. \end{aligned}$$

Therefore, the lower bound of the mutual information is given by,

$$I(W;\hat{W}) \geq h(W) - \sum_{i=1}^{m} \min\left\{h(W_i), \frac{1}{2}\log(2\pi e \mathbb{E}_{W,\hat{W}}[(W_i - \hat{W}_i)^2])\right\}.$$

A.2 Lower bound for linear model

For complex models, the general lower bound in Lemma 1 is difficult to evaluate, due to the large dimension of parameters. It was shown by Jiao et al. (2017) that the sample complexity to estimate differential entropy is exponential to the dimension. It's even harder to design an algorithm to achieve the lower bound. But for linear model, the lower bound can be simplified. For $f_w(x) = w^T x$, the distortion function $d(w, \hat{w})$ can be written as

$$\begin{aligned} d(w, \hat{w}) &= \mathbb{E}_X \left[(f_w(X) - f_{\hat{w}}(X))^2\right] = \mathbb{E}_X \left[(w^T X - \hat{w}^T X)^2\right] \\ &= \mathbb{E}_X \left[(w - \hat{w})^T X X^T (w - \hat{w})\right] = (w - \hat{w})^T \mathbb{E}_X[XX^T](w - \hat{w}). \end{aligned}$$

Since we assumed that $\mathbb{E}[X] = 0$, $\mathbb{E}[X_i^2] = \lambda_{x,i} > 0$ and $\mathbb{E}[X_i X_j] = 0$, so the constraint in Lemma 1 is given by

$$\begin{aligned} D &\geq \mathbb{E}_{W,\hat{W}} \left[(W - \hat{W})^T \mathbb{E}_X[XX^T](W - \hat{W})\right] \\ &= \sum_{i=1}^m \lambda_{x,i} \underbrace{\mathbb{E}_{W,\hat{W}} \left[(W_i - \hat{W}_i)^2\right]}_{D_i}. \end{aligned}$$

Then the optimization problem in Lemma 1 can be written as follows

$$\begin{aligned} \max_{p(\hat{w}|w)} \quad & \sum_{i=1}^m \min\{h(W_i), \frac{1}{2}\log(2\pi e D_i)\} \\ \text{s.t.} \quad & \sum_{i=1}^m \lambda_{x,i} D_i \leq D. \end{aligned}$$

Here W_i is a Gaussian random variable, so $h(W_i) = \frac{1}{2}\log(2\pi e \mathbb{E}[W_i^2])$. The Lagrangian function of the problem is given by

$$\begin{aligned} & \mathcal{L}(D_1, \dots, D_m, \mu) \\ = & \sum_{i=1}^m \Big(\min\{\frac{1}{2}\log \mathbb{E}[W_i^2], \frac{1}{2}\log D_i\} + \frac{1}{2}\log(2\pi e) - \mu \lambda_{x,i} D_i\Big). \end{aligned}$$

By setting the derivative w.r.t. D_i to 0, we have

$$0 = \frac{\partial \mathcal{L}}{\partial D_i} = \frac{1}{2D_i} - \mu \lambda_{x,i}.$$

for all D_i such that $D_i < \mathbb{E}[W_i^2]$. So the optimal D_i should satisfy that $D_i \lambda_{x,i}$ is constant, for all D_i such that $D_i < \mathbb{E}[W_i^2]$. Also the optimal D_i is at most $\mathbb{E}[W_i^2]$. Also, since $h(W) = \frac{m}{2}\log(2\pi e) + \frac{1}{2}\log\det(\Sigma_W)$ the lower bound is given by

$$R(D) \geq \frac{1}{2}\log\det(\Sigma_W) - \sum_{i=1}^m \frac{1}{2}\log(D_i),$$

where

$$D_i = \begin{cases} \mu/\lambda_{x,i} & \text{if} \mu < \lambda_{x,i}\mathbb{E}_W[W_i^2] \;, \\ \mathbb{E}_W[W_i^2] & \text{if} \mu \geq \lambda_{x,i}\mathbb{E}_W[W_i^2] \;, \end{cases}$$

where μ is chosen that $\sum_{i=1}^m \lambda_{x,i} D_i = D$.

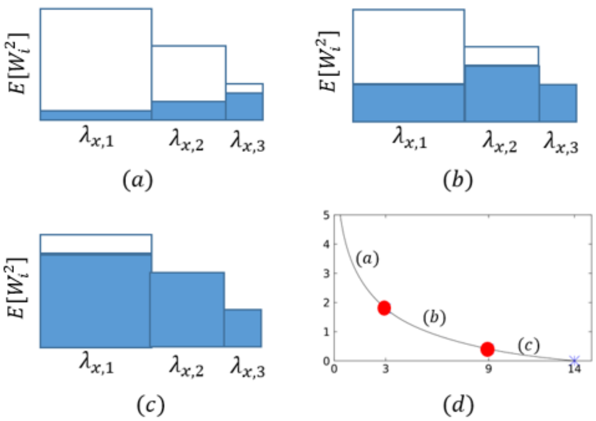

Figure 6: Illustration of “weighted water-filling” process.

This lower bound gives rise to a “weighted water-filling”, which differs from the classical “water-filling” for rate-distortion of colored Gaussian source in Cover and Thomas (2012, Figure 13.7), since the water level's D_i are proportional to $1/\lambda_{x,i}$, which is related to the input of the model rather than the parameters to be compressed. To illustrate the “weighted water-filling” process, we choose a simple example where $\Sigma_W = \Sigma_X = \text{diag}[3, 2, 1]$. In Figure 6, the widths of each rectangle are proportional to $\lambda_{x,i}$, and the heights are proportional to $\Sigma_W = [3, 2, 1]$. The water level in each rectangle is D_i and the volume of water is μ. As D starts to increase from 0, each rectangle is filled with same volume of water (μ is the same), but the water level D_i's increase with speed $1/\lambda_{x,i}$ respectively (Figure 6.(a)). This gives segment (a) of the rate distortion curve in Figure 6.(d). If D is large enough such that the third rectangle is full, then D_3 is fixed to be $\mathbb{E}[W_3^2] = 1$, whereas D_1 and D_2 continuously increase (Figure 6.(b)). This gives segment (b) in Figure 6.(d). Keep increasing D until the second rectangle is also full, then D_2 is fixed to be $\mathbb{E}[W_2^2] = 2$ and D_1 continuous increasing (Figure 6 (c)). This gives segment (c) in Figure 6.(d). The entire rate-distortion function is shown in Figure 6(d), where the first red dot corresponds to the moment that the third rectangle is exactly full, and the second red dot corresponds to moment that the second rectangle is exactly full.

A.3 Achievability

We prove that this lower bound is achievable. To achieve the lower bound, we construct the compression algorithm in Algorithm 1,

Intuitively, the optimal compressor does the following: (1) Find the optimal water levels D_i for “weighted water filling”. (2) For the entries where the corresponding rectangles are full, simply discard the entries; (3) for the entries where the corresponding rectangles are not full, add a noise which is independent of $\hat{W}_i$ and has a variance proportional to the water level. That is possible since W is Gaussian. (4) Combine the conditional probabilities.

To see that this compressor is optimal, we will check that the compressor makes all the inequalities become equality. Here is all the inequalities used in the proof.

- $h(W_i \,|\, W_1, \ldots, W_{i-1}, \hat{W}_i, \ldots, \hat{W}_m) \leq h(W_i|\hat{W}_i)$ for all $i = 1...m$. It becomes equality by $P_{\hat{W}|W} = \prod_{i=1}^m P_{\hat{W}_i|W}$.
- Either
  - $h(W_i|\hat{W}_i) \leq h(W_i)$. It becomes equality for those $\hat{W}_i = 0$.
  - $h(W_i - \hat{W}_i|\hat{W}_i) \leq h(W_i - \hat{W}_i) \leq \frac{1}{2}\log(2\pi e \mathbb{E}_{W,\hat{W}}[(W_i - \hat{W})^2])$. It becomes equality for those $\hat{W}_i$'s such that $W_i - \hat{W}_i$ is independent of $\hat{W}_i$ and $W_i - \hat{W}_i$ is Gaussian.

Algorithm 1 Optimal compression algorithm for linear regression

```
Input: distortion $D$, covariance matrix of parameters $\Sigma_W$, covariance matrix of data $\Sigma_X = \text{diag}[\lambda_{x,1}, \ldots, \lambda_{x,m}]$.
Choose $D_i$'s such that
    $D_i = \begin{cases} \mu/\lambda_{x,i} & \text{if} \mu < \lambda_{x,i}\mathbb{E}_W[W_i^2] \ , \\ \mathbb{E}_W[W_i^2] & \text{if} \mu \geq \lambda_{x,i}\mathbb{E}_W[W_i^2] \ , \end{cases}$
where $\sum_{i=1}^{m} \lambda_{x,i} D_i = D$.
for $i = 1$ to $m$ do
  if $D_i = \mu/\lambda_{x,i}$ then
    Choose $\hat{W}_i = 0$
  else
    Choose a conditional distribution $P_{\hat{W}_i|W_i}$ such that $W_i = \hat{W} + Z_i$ where $Z_i \sim \mathcal{N}(0, D_i)$, $\hat{W}_i \sim \mathcal{N}(0, \mathbb{E}_W[W_i^2] - D_i)$ and $\hat{W}_i$ is independent of $Z_i$.
  end if
end for
Combine the conditional probability distributions by $P_{\hat{W}|W} = \prod_{i=1}^{m} P_{\hat{W}_i|W_i}$.
```

- The "water levels" D_i. It becomes equality by choosing the D_i's according to Lagrangian conditions.

Therefore, Algorithm 1 gives a compressor $P_{\hat{W}|W}^{(D)}$ such that $\mathbb{E}_{P_W \circ P_{\hat{W}|W}^{(D)}}[d(W, \hat{W})] = D$ and $I(W; \hat{W}) = \underline{R}(D)$, hence the lower bound is tight.

B Proof of Theorem 4

In this section, we provide the proof of Theorem 4. For simplicity let $\sigma(t) = t\mathbb{I}\{t \geq 0\}$ denotes the ReLU activation function. First we deal with the objective of the compression algorithm,

$$\begin{aligned} (w - \hat{w})^T I_w (w - \hat{w}) &= (w - \hat{w})^T \mathbb{E}_X \left[\nabla_w f_w(x) \nabla_w f_w(x)^T \right] (w - \hat{w}) \\ &= (w - \hat{w})^T \mathbb{E}_X \left[\nabla_w \sigma(w^T x) \nabla_w \sigma(w^T x)^T \right] (w - \hat{w}) \\ &= (w - \hat{w})^T \mathbb{E}_X \left[x^T (\sigma'(w^T x))^2 x \right] (w - \hat{w}) \\ &= \mathbb{E}_X \left[\mathbb{I}\{w^T x \geq 0\} ((w - \hat{w})^T x)^2 \right] \end{aligned}$$

Notice that x is jointly Gaussian random variable with zero mean and non-degenerate variance, so the distribution of x is equivalent to the distribution of $-x$. Therefore,

$$\mathbb{E}_X[\mathbb{I}\{w^T x \geq 0\}((w - \hat{w})^T x)^2] = \int_{x: w^T x \geq 0} ((w - \hat{w}^T)x)^2 dx$$

$$= \frac{1}{2} \left(\int_{x: w^T x \geq 0} ((w - \hat{w}^T)x)^2 dx + \int_{x: w^T x \leq 0} ((w - \hat{w}^T)x)^2 dx \right)$$

$$= \frac{1}{2} \int_{x \in \mathbb{R}}^{d} ((w - \hat{w}^T)x)^2 dx = \frac{1}{2} (w - \hat{w})^T \Sigma_X (w - \hat{w})$$

So minimizing the gradient-squared based loss is equivalent to minimizing $(w - \hat{w})^T \Sigma_X (w - \hat{w})$. Similarly, the condition $\hat{w} I_w (w - \hat{w}) = 0$ is equivalent to $\hat{w} \Sigma_X (w - \hat{w}) = 0$. Now we deal with the MSE loss function $\mathbb{E}[(f_w(x) - f_{\hat{w}}(x))^2]$. We utilize the Hermite polynomials and Fourier analysis on Gaussian space. We use the following key lemma,

Lemma 2 *(Ge et al. (2017, Claim 4.3)) Let f, g be two functions from $\mathbb{R}$ to $\mathbb{R}$ such that $f^2, g^2 \in L^2(\mathbb{R}, e^{-x^2/2})$. The for any unit vectors u, v, we have that*

$$\mathbb{E}_{x \in \mathcal{N}(0, I_{d\times d})}[f(u^T x) g(v^T x)] = \sum_{p=0}^{\infty} \hat{f}_p \hat{g}_p (u^T v)^p$$

where $\hat{f}_p = \mathbb{E}_{x\in\mathcal{N}(0,1)}[f(x)h_p(x)]$ is the p-th order coefficient of f, where h_p is the p-th order probabilists' Hermite polynomial.

Please see Section 4.1 in Ge et al. (2017) for more backgrounds of the Hermite polynomials and Fourier analysis on Gaussian space. For ReLU function, the coefficients are given by $\hat{\sigma}_0 = \frac{1}{\sqrt{2\pi}}$, $\hat{\sigma}_1 = \frac{1}{2}$. For $p \geq 2$ and even, $\hat{\sigma}_p = \frac{((p-3)!!)^2}{\sqrt{2\pi p!}}$. For $p \geq 2$ and odd, $\hat{\sigma}_p = 0$. Since $X \sim \mathcal{N}(0, \Sigma_X)$, we can write $x = \Sigma_X^{1/2} z$, where $z \sim \mathcal{N}(0, I_d)$. So for any compressed weight $\hat{w}$, we have

$$\begin{aligned}
& \mathbb{E}_X\left[(f_w(x) - f_{\hat{w}}(x))^2\right] = \mathbb{E}_X\left[(\sigma(w^T x) - \sigma(\hat{w}^T x))^2\right] \\
= \; & \mathbb{E}_{z\in\mathcal{N}(0,I_d)}[(\sigma(w^T \Sigma_X^{1/2} z) - \sigma(\hat{w}^T \Sigma_X^{1/2} z))^2] \\
= \; & \mathbb{E}_{z\in\mathcal{N}(0,I_d)}[\sigma(w^T \Sigma_X^{1/2} z)^2] - 2\mathbb{E}_{z\in\mathcal{N}(0,I_d)}[\sigma(w^T \Sigma_X^{1/2} z)\sigma(\hat{w}^T \Sigma_X^{1/2} z)] + \mathbb{E}_{z\in\mathcal{N}(0,I_d)}[\sigma(\hat{w}^T \Sigma_X^{1/2} z)^2] \\
= \; & \sum_{p=0}^{\infty} \hat{\sigma}_p^2 (w^T \Sigma_X w)^p - 2\sum_{p=0}^{\infty} \hat{\sigma}_p^2 (w^T \Sigma_X \hat{w})^p + \sum_{p=0}^{\infty} \hat{\sigma_p}^2 (\hat{w}^T \Sigma_X \hat{w})^p \\
= \; & \sum_{p=0}^{\infty} \hat{\sigma}_p^2 \left(\underbrace{(w^T \Sigma_X w)^p - 2(w^T \Sigma_X \hat{w})^p + (\hat{w}^T \Sigma_X \hat{w})^p}_{D_p(w,\hat{w})} \right)
\end{aligned}$$

Now we can see that $D_0(w, \hat{w}) = 0$. $D_1(w, \hat{w}) = w^T \Sigma_X w - 2w^T \Sigma_X \hat{w} + \hat{w}^T \Sigma_X w = (w - \hat{w})^T \Sigma_X (w - \hat{w})$, is just the objective. The following lemma gives the minimizer of $D_p(w, \hat{w})$ for higher order p.

Lemma 3 *If $\hat{w}^*$ satisfies $\hat{w}^* \Sigma_X(\hat{w} - w) = 0$ and*

$$\hat{w}^* = \arg\min_{\hat{s}\in\mathcal{W}} D_1(w, \hat{w})$$

for some constrained set $\mathcal{W}$. Then for any $p \geq 2$ and even, we have

$$\hat{w}^* = \arg\min_{\hat{w}\in\mathcal{W}} D_p(w, \hat{w})$$

Since the coefficients $\hat{\sigma}_p$ is zero for $p \geq 3$ and odd, so if a compressed weight $\hat{w}$ satisfied $\hat{w}\Sigma_X(\hat{w} - w) = 0$ and minimizes $D_1(\hat{w}, w) = (\hat{w} - w)^T \Sigma_X (\hat{w} - w)$, then it is the minimizer for all $D_p(w, \hat{w})$ for even p, therefore a minimizer of the MSE loss.

B.1 Proof of Lemma 3

For simplicity of notation, define $A = w^T \Sigma_X w$, $B = \hat{w}^T \Sigma_X (\hat{w} - w)$ and $C = D_1(w, \hat{w}) = (\hat{w} - w)^T \Sigma_X (\hat{w} - w)$. For all compressors, we have $C \leq A$. Therefore, $w^T \Sigma_X \hat{w} = A + B - C$ and $\hat{w}^T \Sigma_X \hat{w} = A + 2B - C$. So

$$D_p(w, \hat{w}) \quad = \quad A^p - 2(A + B - C)^p + (A + 2B - C)^p$$

First notice that

$$\frac{\partial D_p(w, \hat{w})}{\partial B} = 2p((A + 2B - C)^{p-1} - (A + B - C)^{p-1}).$$

For even $p \geq 2$, x^{p-1} is monotonically increasing, so $(A + 2B - C)^{p-1} > (A + B - C)^{p-1}$ if $B > 0$ and vice versa. Therefore, for fixed A and C, $D_p(w, \hat{w})$ is monotonically increasing for positive B and decreasing for negative B. Therefore, $D_p(w, \hat{w})$ is minimized when $B = 0$, and the minimal value is $D_p(w, \hat{w}) = A^p - 2(A - C)^p + (A - C)^p = A^p - (A - C)^p$, which is monotonically increasing with respect to C. So if $\hat{w}^*$ satisfies $B = 0$ and is a minimzer of $C = D_1(w, \hat{w})$, it is also a minimizer for $D_p(w, \hat{w})$ for all $p \geq 2$ and even.

C Details of the experiments

In this appendix, we give some details of the experiment and additional experiments which are omitted in the main text.

C.1 Additional experiment results

We present the experiment results for CIFAR100 here, due to page limit of the main text.

In Figure 7 and Figure 8, we show the result for unsupervised pruning and quantization, introduced in Section 6.1. We can see that, similar to the experiments of MNIST and CIFAR10, the proposed objectives gives better accuracy and smaller loss than the baseline.

In Figure 9 and Figure 10, we show the result for supervised pruning and quantization, introduced in Section 6.2. Due to the slow running speed for estimating the Hessian $\nabla^2_{w_i} \mathcal{L}_w(x, y)$, we only compare two objectives — baseline and gradient. It is shown that the gradient objective gives better accuracy and smaller loss.

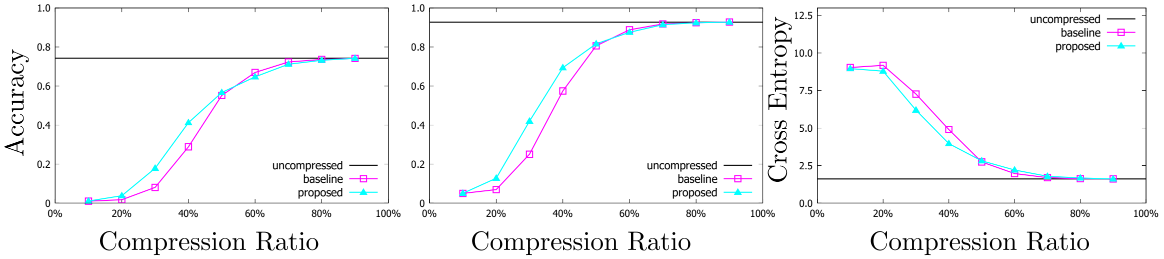

Figure 7: Result for unsupervised pruning experiment for CIFAR 100 experiment. Left: top-1 accuracy. Middle: top-5 accuracy. Right: cross entropy loss.

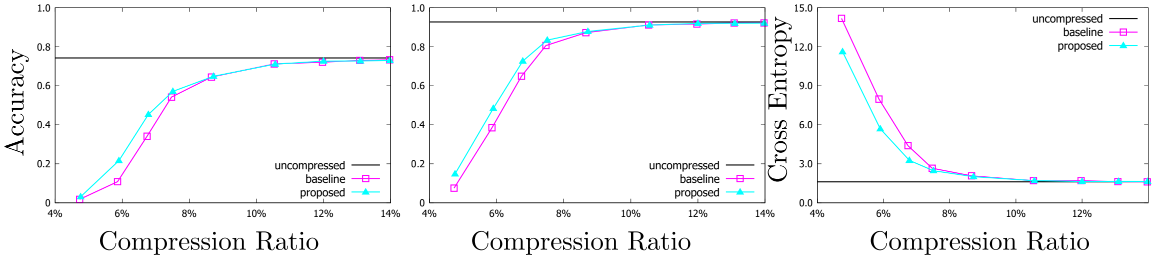

Figure 8: Result for unsupervised quantization experiment for CIFAR 100 experiment. Left: top-1 accuracy. Middle: top-5 accuracy. Right: cross entropy loss.

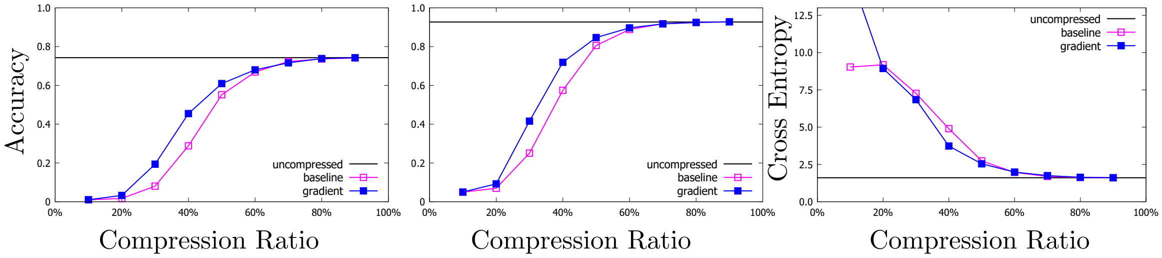

Figure 9: Result for supervised pruning experiment for CIFAR 100 experiment. Left: top-1 accuracy. Middle: top-5 accuracy. Right: cross entropy loss.

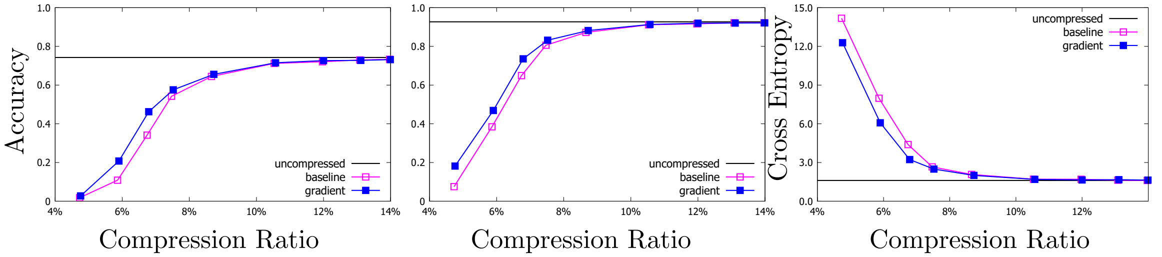

Figure 10: Result for supervised quantization experiment for CIFAR 100 experiment. Left: top-1 accuracy. Middle: top-5 accuracy. Right: cross entropy loss.

C.2 Algorithm for finding optimal quantization

We present a variation of k-means algorithm which are used to find the optimal quantization for the following objective,

$$\min_{c_1,\ldots,c_k,A\in[k]^m} \sum_{i=1}^{m} \left(I_i(w_i - c_{A_i})^2 + H_i(w_i - c_{A_i})^4 \right)$$

where I_i is positive weight importance for quadratic term and H_i is positive weight importance for quartic term. Basic idea of the algorithm is — the assignment step finds the optimal assignment given fixed centroids, and the update step finds the optimal centroids given fixed assignments. This is used for gradient+hessian objective in Section 6.2.

Here we show that the cubic equation in Algorithm 2 has only one real root. It was know that if the determinant $\Delta_0 = b^2 - 3ac$ of a cubic equation $ax^3 + bx^2 + cx + d = 0$ is negative, then the cubic equation is strictly increasing or decreasing, hence only have one real root. Now we show that the determinant is negative in this case (we drop the subsripts of the summation for simplicity).

$$\begin{aligned} \Delta_0 &= (\sum 12H_i w_i)^2 - 3(\sum 4H_i)(\sum 12H_i w_i^2 + 2I_i) \\ &= 144\left((\sum H_i w_i)^2 - (\sum H_i)(\sum H_i w_i^2) \right) - 24(\sum H_i)(\sum I_i) \end{aligned}$$

The first term is non-positive because of Cauchy-Schwarz inequality. The second term is negative since H_i's and I_i's are all positive. Hence the determinant is negative.

Algorithm 2 Quartic weighted k-means

```
input Weights $\{w_1, \dots, w_m\}$, weight importances $\{I_1, \dots, I_m\}$, quartic weight importances $\{H_1, \dots, H_m\}$, number of clusters $k$, iterations $T$
  Initialize the centroid of $k$ clusters $\{c_1^{(0)}, \dots, c_k^{(0)}\}$
  for $t = 1$ to $T$ do
    Assignment step:
    for $i = 1$ to $m$ do
      Assign $w_i$ to the nearest cluster centroid, i.e. $A_i^{(t)} = \arg\min_{j \in [k]} (w_i - c_j^{(t-1)})^2$.
    end for
    Update step:
    for $j = 1$ to $k$ do
      Find the only real root $x^*$ of the cubic equation
      $$(\sum_{i:A_i^{(t)}=j} 4H_i)x^3 - (\sum_{i:A_i^{(t)}=j} 12H_i w_i)x^2 + (\sum_{i:A_i^{(t)}=j} (12H_i w_i^2 + 2I_i))x - (\sum_{i:A_i^{(t)}=j} (4H_i w_i^3 + 2I_i w_i)) = 0$$
      Update the cluster centroids $c_j^{(t)}$ be the real root $x^*$.
    end for
  end for
output Centroids $\{c_1^{(T)}, \dots, c_k^{(T)}\}$ and assignments $A^{(T)} \in [k]^m$.
```

C.3 Effects of hyperparameters

Here we briefly talk about the hyperparameters used in estimating the gradients $\mathbb{E}[\nabla_{w_i} \mathcal{L}_w(X, Y)]$ and hessians $\mathbb{E}[\nabla^2_{w_i} \mathcal{L}_w(X, Y)]$.

C.3.1 Temperature scaling method

The temperature scaling method proposed by Guo et al. (2017), aims to improve the confidence calibration of a classification model. Denote $z_w(x) \in \mathbb{R}^C$ is the output of the neural network, and classical softmax gives $f_w^{(c)}(x) = \frac{\exp\{z_w^{(c)}(x)\}}{\sum_{c \in C} \exp\{z_w^{(c)}(x)\}}$. The temperature sclaed softmax gives

$$f_w^{(c)}(x) = \frac{\exp\{z_w^{(c)}(x)/T\}}{\sum_{c \in C} \exp\{z_w^{(c)}(x)/T\}}$$

by choosing different T, the prediction of the model does not change, but the cross entropy loss may change. Hence, we can finetune T to get a better model calibration. In our experiment, we found that in MNIST experiment, the model is poorly calibrated. Hence, the variance of estimating gradient and hessian is very large. To solve this, we adopt a temperature $T > 1$ such that the loss from correctly-predicted data can also be backpropagated.

In Figure 11, we show the effect of T for supervised pruning for MNIST. We can see that as T increases from 1, the performance become better at first, then become worse. In our experiment, we choose $T \in \{1.0, 2.0, \dots, 9.0\}$ which gives best accuracy.

C.3.2 Regularizer of hessian

In the experiments, we estimate the hessians $\mathbb{E}[\nabla^2_{w_i} \mathcal{L}_w(X, Y)]$ using the curvature propagation algorithm Martens et al. (2012). However, due to the sparsity introduced by ReLU, there are many zero entries of the estimated hessians, which hurts the performance of the algorithm. Hence, we add a constant $\mu > 0$ to the estimated hessians. In Figure 12, we show that effect of μ for supervised pruning for CIFAR10.

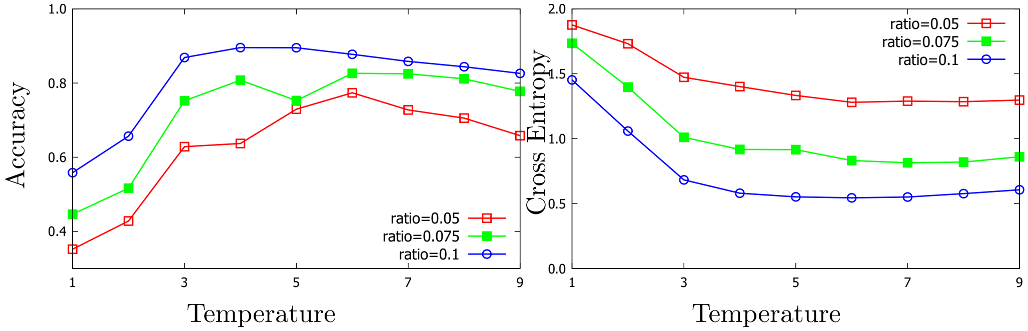

Figure 11: Effect of the temperature T. Left: accuracy of supervised pruning for MNIST. Right: cross entropy of supervised pruning for MNIST. Different lines denote different compression ratio $\in \{0.05, 0.075, 0.1\}$

We can see that as μ increases from 0, the performance increase first then decrease. We use simple binary search to find the best μ.

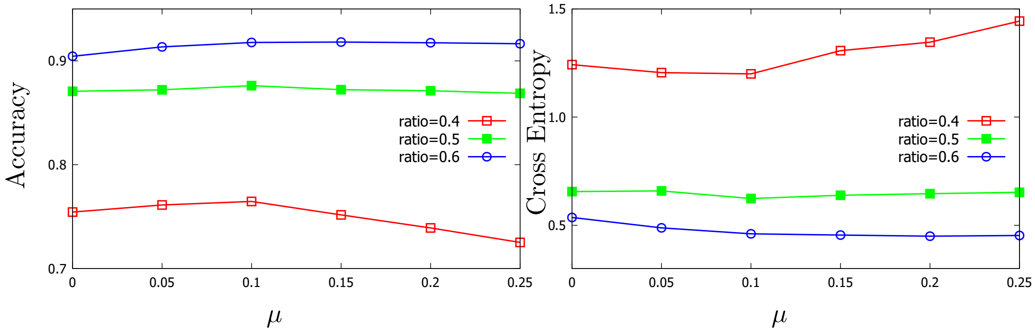

Figure 12: Effect of the regularizer μ. Left: accuracy of supervised pruning for CIFAR10. Right: cross entropy of supervised pruning for CIFAR10. Different lines denote different compression ratio $\in \{0.4, 0.5, 0.6\}$