An Information-Theoretic Justification for Model Pruning

Berivan Isik
Stanford University
berivan.isik@stanford.edu

Tsachy Weissman
Stanford University
tsachy@stanford.edu

Albert No
Hongik University
albertno@hongik.ac.kr

Abstract

We study the neural network (NN) compression problem, viewing the tension between the compression ratio and NN performance through the lens of rate-distortion theory. We choose a distortion metric that reflects the effect of NN compression on the model output and derive the tradeoff between rate (compression) and distortion. In addition to characterizing theoretical limits of NN compression, this formulation shows that *pruning*, implicitly or explicitly, must be a part of a good compression algorithm. This observation bridges a gap between parts of the literature pertaining to NN and data compression, respectively, providing insight into the empirical success of model pruning. Finally, we propose a novel pruning strategy derived from our information-theoretic formulation and show that it outperforms the relevant baselines on CIFAR-10 and ImageNet datasets.

1 Introduction

The recent success of NNs in various machine learning applications has come with their over-parameterization. Deployment of such over-parameterized models on edge devices is challenging as these devices have limited storage, computation, and power resources. Motivated by this, there has been significant interest in NN compression by the research community. The most established NN compression techniques can be broadly grouped into five categories: quantization (Li et al., 2016; Banner et al., 2018; Jacob et al., 2018; Jung et al., 2019; Wang et al., 2019b; Choi et al., 2020; Young et al., 2020; Idelbayev et al., 2021) and coding (Wiedemann et al., 2020; Zhe et al., 2021) of NN parameters, pruning (Han et al., 2016; Molchanov et al., 2016; Carreira-Perpinan and Idelbayev, 2018; Liu et al., 2018; Yu et al., 2018; Lin et al., 2019; Peng et al., 2019; Xiao et al., 2019; Zhao et al., 2019; Blalock et al., 2020; Elsen et al., 2020; Park et al., 2020; Renda et al., 2020), Bayesian compression (Federici et al., 2017; Louizos et al., 2017a,b; Molchanov et al., 2017; Dai et al., 2018), distillation (Hinton et al., 2015; Polino et al., 2018; Wang et al., 2019a), and low-rank matrix factorization (Sainath et al., 2013; Ioannou et al., 2015; Idelbayev and Carreira-Perpinan, 2020). The success of these techniques in compressing NN models without a significant performance loss brings a theoretical question: *what is the fundamental limit of NN compression while maintaining a target performance?*

A similar question arises in the classical data compression problem as well (Salomon, 2004). Shannon (1948) introduced the mathematical formulation of the data compression problem, where the goal is to describe a source sequence with the minimum number of bits. In an information-theoretic sense, entropy is the limit of how much a source sequence can be losslessly compressed. However, *in practice*, there are many sources such as image, video, and audio, where lossless compression cannot achieve a high enough compression rate. In such cases, we need to compress the source sequence in a *lossy* manner allowing some *distortion* between the source and reconstruction. This is where rate-distortion theory comes into the picture. For lossy compression, rate-distortion theory gives the limit of how much a source sequence can be compressed without exceeding a target distortion level (Berger, 2003).

In this work, we connect these two lines of research and study the theoretical limits of lossy NN compression via rate-distortion theory. In particular, we consider a classical lossy compression problem to compress NN weights while minimizing the perturbation in the NN output space. We first (1) define a distortion metric that upper bounds the output perturbation due to compression, then (2) find a probability distribution that fits NN parameters, and finally (3) derive the

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

rate-distortion function for the chosen distortion metric and distribution. This function describes the theoretical tradeoff between rate (compression ratio) and NN output perturbation, thus provides insight into how compressible NN models are. Furthermore, our findings indicate that the compressed model that reaches the optimal achievable compression ratio must be sparse. This suggests that a good NN compression algorithm must, implicitly or explicitly, involve a pruning step, complimenting the empirical success of pruning strategies (Gale et al., 2019). Therefore, we provide theoretical support for pruning as a rate-distortion theoretic compression scheme that maintains the model output.

Inspired by this observation, we propose a practical lossy compression algorithm for NN models. The reconstruction of our algorithm is a sparse model, which naturally induces a novel pruning strategy. Our algorithm is based on *successive refinability* – a property that often helps to reduce the complexity of lossy compression algorithms (Equitz and Cover, 1991). Our strategy differs from previous score-based pruning methods as it relies solely on an information-theoretic approach to a data compression problem with additional practical benefits that we cover in Section 6. We also prove that the proposed algorithm is sound from a rate-distortion theoretic perspective. We demonstrate the efficacy of our pruning strategy on CIFAR-10 and ImageNet datasets. Lastly, we show that our strategy provides a tool for compressing NN gradients as well, an important objective in communication-efficient federated learning (FL) settings (Kairouz et al., 2019). The contributions of our paper can be summarized as:

- We take a step in bridging the gap between NN compression and data compression.
- We present the rate-distortion theoretical limit of achievable NN compression given a target distortion level and show that pruning is an essential part of a good compression algorithm.
- We propose a novel pruning strategy derived from our findings, which outperforms relevant baselines.

2 Related Work

This section is devoted to prior work on NN compression that has the same flavor as ours, in particular, we touch on (a) data compression approaches to NN compression and (b) pruning. We cover related works in classical data compression as we go through the methodology in Sections 3, 4, and 5.

From Data Compression to NN Compression. To date, several works have proposed to minimize the bit-rate (compressed size) of NNs with quantization techniques (Wang et al., 2019b; Idelbayev et al., 2021; Stock et al., 2021). Some recent work has shown promising results to go beyond quantization using tools from data compression. For instance, Havasi et al. (2019) and Oktay et al. (2019) have trained a model to jointly optimize compression and performance of the model using tools from minimum description length principle (Grünwald and Grunwald, 2007) and a recently advanced image compression framework (Ballé et al., 2016), respectively. While we share the same goal with these papers, our focus is on compressing NN models *post-training*. With this distinction, our work is most related to (Gao et al., 2019), where the authors have put the first attempt to approach NN compression from a rate-distortion theoretic perspective. Although they have shown achievability results on one-layer networks, their results do not generalize to deeper networks without first-order Taylor approximations. Moreover, their formulation relies on the assumption that NN weights follow Gaussian distribution, which currently lacks empirical evidence. On the other hand, we show achievable compression ratios generalized to multi-layer networks without making linear approximations and provide strong empirical evidence for our choice of *Laplacian* distribution for NN weights.

Pruning. The overparameterized nature of NNs has motivated researchers to explore ways to find and remove redundant parameters (Cun et al., 1990; Hassibi et al., 1993). The idea of iterative magnitude pruning was shown to be remarkably successful in deep NNs first by Han et al. (2016), and since then, NN pruning research has accelerated. To improve upon the iterative magnitude pruning scheme of (Han et al., 2016), researchers have looked for different ways to adjust the pruning ratios across layers. For instance, Zhu and Gupta (2017) have suggested pruning the parameters uniformly across layers. Gale et al. (2019), on the other hand, have shown better results when the first convolutional layer is excluded from the pruning and the last fully-connected layer is not pruned more than 80%. Layerwise pruning ratio has also been investigated for NNs pruned at initialization since the explosion of the Lottery Ticket Hypothesis (Frankle and Carbin, 2019; Morcos et al., 2019). Evci et al. (2020) have shown promising results on NNs pruned at initialization where the pruning ratio across layers is adjusted by Erdős-Rényi kernel method, as introduced by Mocanu et al. (2018). More recently, Lee et al. (2021) have proposed adjusting the pruning threshold for each layer based on the norm of the weights at that layer. We follow a similar methodology in (Lee et al., 2021) to normalize the parameters prior to applying our *novel* pruning algorithm. Unlike other pruning strategies, our algorithm outputs

a pruned (sparse) model, without an explicit score-based pruning step. Instead, our reconstruction goes from the coarsest (sparsest) to the finest representation of the model. Parallel to our work, a recent study has proposed a heuristic bottom-up approach as opposed to the common top-down pruning approach and provided promising empirical results (Chen et al., 2021). To the best of our knowledge, our work is the first to provide a rate-distortion theoretic justification for pruning.

3 Preliminaries

In this section, we present the problem setup and briefly introduce the rate-distortion theory and the successive refinement concept.

3.1 Problem Statement

We study a NN compression problem where the network $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$ characterizes a prediction from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$, parameterized by weights $\mathbf{w}$. Our goal is to minimize the difference between $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$ and $\hat{\mathbf{y}} = f(\mathbf{x}; \hat{\mathbf{w}})$, where $\hat{\mathbf{w}}$ is a compressed version of the trained parameters $\mathbf{w}$. In Section 4.1, we define an appropriate distortion function $d(\mathbf{w}, \hat{\mathbf{w}})$ that reflects the perturbation in the output space $\|f(\mathbf{x}; \mathbf{w}) - f(\mathbf{x}; \hat{\mathbf{w}})\|_1$. This is a lossy compression problem where the distortion is a measure of the distance between the original model and the compressed model, and the rate is the number of bits required to represent one weight. In information-theoretic term, rate distortion theory characterizes the minimum achievable rate given the target distortion.

3.2 Notation

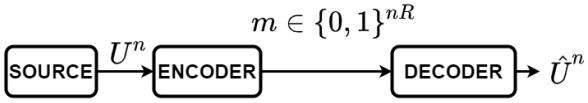
Throughout the paper, $\mathbf{w} \in \mathbb{R}^n$ is the weights of a trained model. Logarithms are natural logarithms. Rate is defined as nats (the unit of information obtained from natural logarithm) per symbol (weight in our case). We use lower case u to denote the realization of a scalar random variable U and $\mathbf{u} = u^n = (u_1, \ldots, u_n)$ to denote the realization of a random vector $\mathbf{U} = U^n = (U_1, \ldots, U_n)$. We use the term "perturbation" for the change in the model output due to compression, whereas "distortion" $d(\mathbf{w}, \hat{\mathbf{w}})$ refers to the change in the *parameter* space. Lastly, $d(u^n, \hat{u}^n) = \frac{1}{n}\sum_{i=1}^{n} d(u_i, \hat{u}_i)$ is the regular extension of the distortion function for an n dimensional vector.

3.3 Rate-Distortion Theory

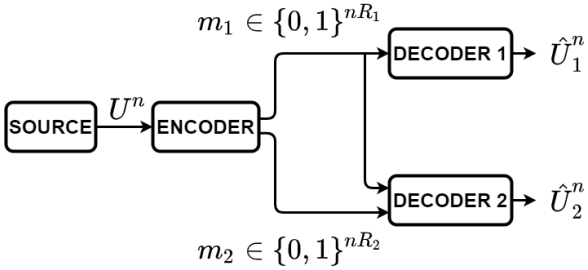
Let $U_1, \ldots, U_n \in \mathcal{U}$ be a source sequence generated by i.i.d. $\sim p(u)$ where $p(u)$ is a probability density function and $\mathcal{U} = \mathbb{R}$. The encoder $f_e : \mathcal{U}^n \to \{0,1\}^{nR}$ describes this sequence in nR bits, where this binary representation is called a "message" m. The decoder $f_d : \{0,1\}^{nR} \to \hat{\mathcal{U}}^n$ reconstructs an estimate $\hat{\mathbf{u}} = \hat{u}^n \in \hat{\mathcal{U}}^n$ based on $m \in \{0,1\}^{nR}$ where $\hat{\mathcal{U}} = \mathbb{R}$ as well. This process, summarized in Figure 1(a), is called lossy source coding. The number of bits per source symbol ($\frac{nR}{n} = R$ in this case) and the "distance" $d(\mathbf{u}, \hat{\mathbf{u}}) = d(u^n, \hat{u}^n) = \frac{1}{n}\sum_{i=1}^{n} d(u_i, \hat{u}_i)$ between $\mathbf{u}$ and $\hat{\mathbf{u}}$ are named as rate and distortion, respectively. Ideally, we would like to keep both rate and distortion low, but there is a tradeoff between these two quantities, which is characterized by the rate-distortion function (Shannon, 1948; Berger, 2003; Cover and Thomas, 2006) as:

$$R(D) = \min_{p(\hat{u}|u):\mathbb{E}[d(u,\hat{u})]\leq D} I(U;\hat{U}) \tag{1}$$

where $I(U;\hat{U})$ is the mutual information between U and $\hat{U}$, and $d(\cdot,\cdot)$ is a predefined distortion metric, e.g. ℓ_2 distance. The rate-distortion function $R(D)$ in Eq. 1 is the minimum achievable rate at distortion D, and the conditional distribution $p(\hat{u}|u)$ that achieves $I(U;\hat{U}) = R(D)$ explains how an optimal encoder-decoder pair should operate for the source $p(u)$. We can also define the inverse, namely the distortion-rate function $D(R)$, which is the minimum achievable distortion at rate R. Clearly, source distribution has a critical role in the solution of the rate distortion problem. We discuss possible assumptions for the distribution of NN weights in Section 4.2.



(a) Lossy Source Coding.



(b) Successive Refinement.

Figure 1: (a) Source Coding, (b) Successive Refinement with 2 Decoders.

3.4 Successive Refinement

In the successive refinement problem, summarized in Figure 1(b), the encoder wants to describe the source

to two decoders, where each decoder has its own target distortion, D_1 and D_2. Instead of having separate encoding schemes for each decoder, the successive refinement encoder encodes a message m_1 for Decoder 1 (with higher target distortion, D_1), and encodes an extra message m_2 where the second decoder gets both m_1 and m_2. Receiving both m_1 and m_2, Decoder 2 reconstructs $\hat{\mathbf{U}}_2$ with distortion D_2. Since the message m_1 is re-used, the performance of successive refinement encoder is sub-optimal in general. However, in some cases, the successive refinement encoder achieves the optimum rate-distortion tradeoff as if dedicated encoders were used separately. In such a case, we call the source (distribution) and the distortion pair successively refinable (Koshelev, 1980; Equitz and Cover, 1991). In Section 5.1, we discuss how to achieve low complexity via successive refinement.

4 Rate-Distortion Theory for Neural Network Parameters

In this section, we first derive the distortion metric to be used in the rate-distortion function, then we estimate the source distribution (probability density of NN weights), and finally, we present the rate-distortion function associated with the chosen distortion metric and the source distribution.

4.1 Distortion Metric

Our objective is to minimize the difference between the output of the original NN model and the compressed model. Formally, we would like to keep the output perturbation $\|f(\mathbf{x};\mathbf{w}) - f(\mathbf{x};\hat{\mathbf{w}})\|_1$ small. Since the effect of a weight distortion on the output space $f(\mathbf{x};\mathbf{w})$ is intractable for deep NNs, we seek to find a distortion function on parameter space that upper bounds $\|f(\mathbf{x};\mathbf{w}) - f(\mathbf{x};\hat{\mathbf{w}})\|_1$.

Prior work has derived an upper bound for the ℓ_2 norm of the output perturbation as the Frobenius norm of the difference between $\mathbf{w}$ and $\hat{\mathbf{w}}$ when only a single layer is compressed (Lee et al., 2021). More precisely, consider a fully connected NN model with d layers and ReLU activation. Let $\mathbf{w}$ be the weights of the original trained model and $\hat{\mathbf{w}}$ be a compressed version of $\mathbf{w}$ where $\hat{\mathbf{w}}$ is the same with $\mathbf{w}$ except in the l-th layer. In such a case, i.e., when only a single layer is compressed, the output perturbation is bounded by

$$\sup_{\|\mathbf{x}\|_2\leq 1} \|f(\mathbf{x};\mathbf{w}) - f(\mathbf{x};\hat{\mathbf{w}})\|_2 \leq \frac{\|\mathbf{w}^{(l)} - \hat{\mathbf{w}}^{(l)}\|_F}{\|\mathbf{w}^{(l)}\|_F} \cdot \left(\prod_{k=1}^{d} \|\mathbf{w}^{(k)}\|_F\right) \tag{2}$$

where $\mathbf{w}^{(l)}$ indicates the weights of the l-th layer. Inspired by Eq. 2, Lee et al. (2021) have introduced Layer-Adaptive Magnitude-based Pruning (LAMP) score $(\mathbf{w}_i^{(l)})^2 / \left(\sum_j (\mathbf{w}_j^{(l)})^2\right)$ to measure the importance of the weight $\mathbf{w}_i^{(l)}$ for pruning. Notice that Eq. 2 holds only when a single layer is pruned.

In this work, we follow a similar strategy to relate the "ℓ_1 norm of perturbation on the output space" to "ℓ_1 norm of the weight distortion after compression", but not limited to single-layer compression.

Theorem 1. *Suppose $f(\cdot;\mathbf{w})$ is a fully-connected NN model with d layers and 1-Lipschitz activations $\sigma(\cdot)$ such that $\sigma(0) = 0$, e.g., ReLU. Let $\hat{\mathbf{w}}$ be the reconstructed weights (after compression) where all layers are subject to compression. If $\|\mathbf{w}^{(l)}\|_1 \geq \|\hat{\mathbf{w}}^{(l)}\|_1$ for all $1 \leq l \leq d$ [1], then, we have the following bound on the output perturbation:*

$$\sup_{\|x\|_1\leq 1} \|f(\mathbf{x},\mathbf{w}) - f(\mathbf{x},\hat{\mathbf{w}})\|_1 \leq \left(\sum_{l=1}^{d} \frac{\|\mathbf{w}^{(l)} - \hat{\mathbf{w}}^{(l)}\|_1}{\|\mathbf{w}^{(l)}\|_1}\right)\left(\prod_{k=1}^{d} \|\mathbf{w}^{(k)}\|_1\right) \tag{3}$$

i.e., the output perturbation is bounded by the ℓ_1 distortion of the normalized weights.

The matrix norm $\|\cdot\|_1$ is an induced norm by ℓ_1 vector norm. The proof is given in Appendix A. In Section 5.2 (Remark 2), we show that the proposed compression algorithm satisfies the additional assumption $\|\mathbf{w}^{(l)}\|_1 \geq \|\hat{\mathbf{w}}^{(l)}\|_1$ for all $1 \leq l \leq d$. Since the last term in Eq. 3, $\left(\prod_{k=1}^{d} \|\mathbf{w}^{(k)}\|_1\right)$, is independent of the compression, we do not include this term in our weight distortion function. Then, one distortion function that naturally arises from Theorem 1 is $d(\mathbf{w},\hat{\mathbf{w}}) = \sum_{l=1}^{d} \frac{\|\mathbf{w}^{(l)} - \hat{\mathbf{w}}^{(l)}\|_1}{\|\mathbf{w}^{(l)}\|_1}$. By changing the notation slightly, we would like to minimize the following distortion function

$$d(\mathbf{u},\hat{\mathbf{u}}) = \frac{1}{n}\sum_{i=1}^{n} |u_i - \hat{u}_i| \tag{4}$$

where $\mathbf{u}$ is the normalized weights arisen from the normalization in Eq. 3, i.e., $\mathbf{u}^{(l)} = \frac{\mathbf{w}^{(l)}}{\|\mathbf{w}^{(l)}\|_1}$ for $l = 1,\ldots,d$. In the next section, we derive the rate-distortion function with the distortion metric in Eq. 4, which approximates the perturbation (ℓ_1 loss) on the output space due to compression.

[1] We provide a symmetric version of Theorem 1 in Appendix B, which essentially implies the same upper bound on the output perturbation without requiring the additional condition of $\|\mathbf{w}\|_1 \geq \|\hat{\mathbf{w}}\|_1$

4.2 Rate-Distortion Function for Neural Network Parameters

Since we define our distortion function as the ℓ_1 distortion between $\mathbf{u}$ and $\hat{\mathbf{u}}$ as in Eq. 4, where $\mathbf{u}$ is the normalized NN weights, we can formulate the compression problem as a lossy compression of the normalized NN weights. Before deriving the rate-distortion function, we need a source distribution that fits the normalized weights $\mathbf{u}$. Figure 2 shows that Laplacian distribution is a good fit for pretrained NN weights after normalization as opposed to the common Gaussian assumption in the prior work (Gao et al., 2019). For Figure 2, we use PyTorch's pretrained models with no further training.

Now that we have a distortion metric and a source distribution, suitable for NN compression problem, we can finally derive the rate-distortion function. We consider i.i.d. Laplacian source sequence $u_1, \dots, u_n$ distributed according to $f_L(u;\lambda) = \frac{\lambda}{2} e^{-\lambda|u|}$ with zero-mean and scale factor of λ, reconstructed sequence $v_1, \dots, v_n$, and ℓ_1 distortion given in Eq. 4 with $\hat{\mathbf{u}} = \mathbf{v}$. The rate-distortion function, which is the minimum achievable rate given the target distortion D follows by:

Lemma 1 (Berger (2003))**.** *The rate-distortion function for a Laplacian source with ℓ_1 distortion is given by*

$$R(D) = \begin{cases} -\log(\lambda D), & 0 \le D \le \frac{1}{\lambda} \\ 0, & D > \frac{1}{\lambda} \end{cases} \tag{5}$$

with the following optimal conditional probability distribution that achieves the minimum rate:

$$f_{\mathbf{U}|\mathbf{V}}(u|v) = \frac{1}{2D} e^{-|u-v|/D}. \tag{6}$$

Moreover, the marginal distribution of $\mathbf{V}$ for the optimal reconstruction is

$$f_{\mathbf{V}}(v) = \lambda^2 D^2 \cdot \delta(v) + (1 - \lambda^2 D^2) \cdot \frac{\lambda}{2} e^{-\lambda|v|}, \tag{7}$$

where $\delta(v)$ is a Dirac measure.

The proof of Lemma 1 is given in Appendix D. The rate-distortion function in Eq. 5 describes the tradeoff between NN compression ratio and weight distortion D – which upper bounds the *output* perturbation. Lemma 1 further indicates that:

(1) The rate-distortion theoretic optimal encoder-decoder pair makes the reconstruction sparse as the optimal marginal distribution in Lemma 1 is a sparse Laplacian distribution with sparsity $\lambda^2 D^2$. Therefore, unless a compression scheme involves an implicit or explicit pruning step (to make the reconstruction sparse), the reconstruction does not follow the optimal marginal distribution. This would leave a sub-optimal compression scheme since the mutual information $I(U;\hat{U})$ between the source and reconstruction would be strictly larger than the rate-distortion function.

(2) Once $\mathbf{V}$ is reconstructed at the decoder, the error term on the encoder side, $\mathbf{U} - \mathbf{V}$, follows a Laplacian distribution with parameter $1/D$ (see the conditional distribution in Lemma 1). This allows for a practical coding scheme with low complexity based on successive refinement. That is, we can iteratively [2] describe NN weights with reasonable complexity.

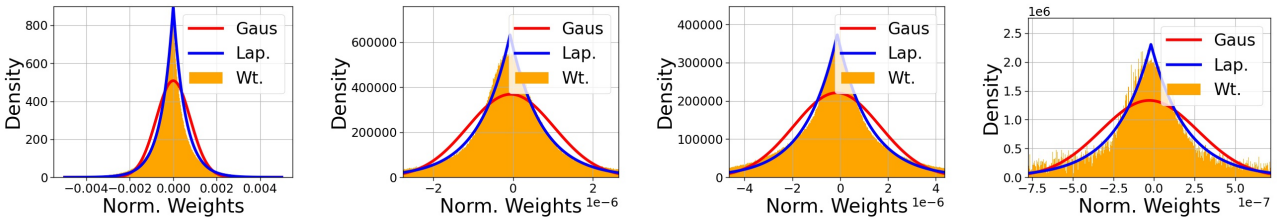
In Theorem 1, we add another constraint that the norm of the reconstructed weights at each layer is smaller than the norm of the original weights at the same layer ($\|\mathbf{w}^{(l)}\|_1 \ge \|\hat{\mathbf{w}}^{(l)}\|_1$). This is mainly because (1) sign change in the NN weights can significantly affect the NN output, hence sign bits must be protected to maintain the performance (Isik et al., 2021); and (2) this inequality ($\|\mathbf{w}^{(l)}\|_1 \ge \|\hat{\mathbf{w}}^{(l)}\|_1$) is necessary to apply the iterative compression algorithm based on successive refinement (to be discussed in Section 5).

In the next section, we develop a NN compression algorithm merging (i) our theoretical findings in Lemma 1 for *optimality* and (ii) successive refinement property for *practicality*.

5 Successive Refinement for Pruning

Rate-distortion theory, although, gives the limit of lossy compression and suggests that pruning must be a part of a good compression algorithm, does not explicitly give the optimal compression algorithm. In *theory*, a compression algorithm could be designed by letting the encoder pick the closest codeword from a random codebook generated according to the marginal distribution of $\mathbf{V}$ in Lemma 1, as suggested by Shannon (1948). However, such a compressor would not be practical due to the size of the randomly generated codebook $|\mathcal{C}| = 2^{nR(D)}$ (exponential in n – number of weights in our case). While designing practical compression algorithms without sacrificing the optimality is a fundamental dilemma in data compression, recent studies have shown that it is possible to design theoretically optimal schemes with low complexity for certain source distributions. In particular, for a successively refinable source, an optimal compression algorithm can also be practical (No et al., 2016). We exploit this idea for the Laplacian source and develop a practical iterative

[2] The term "iterative" in our proposed algorithm is different from the "iterative" magnitude pruning concept.



(a) ResNet-18. (b) ResNet-50. (c) ResNet-152. (d) Wide ResNet-50.

Figure 2: Density of normalized weights. (a) ResNet-18, (b) ResNet-50, (c) ResNet-152, and (d) Wide ResNet-50. Gaus: Gaussian, Lap.: Laplacian, Wt.: Normalized NN weights. We use PyTorch's pretrained models with no further training.

compression algorithm that is rate-distortion theoretically optimal. We call it Successive Refinement for Pruning (SuRP) since it also outputs a sparse model, which can be viewed as a pruned model (although we do not explicitly prune the model). We first present the successive refinement scheme for Laplacian source that shows the core idea to achieve lower complexity, but still impractical. We then push further to provide the practical algorithm and prove the optimality in a rate-distortion theoretic sense.

5.1 Successive Refinement with Randomly Generated Codebooks

Instead of a successive refinement scheme with two decoders as described in Section 3, we consider successive refinement with L decoders. Let $\lambda = \lambda_1 < \cdots < \lambda_L$ where $D_t = 1/\lambda_{t+1}$ is the target distortion at the t-th decoder. This is because the error term at iteration t has a Laplacian distribution with parameter $\lambda_{t+1} = 1/D_t$ in an optimal compression scheme (see Lemma 1). We begin by setting $\mathbf{U}^{(1)} = u^n$. At the t-th iteration, the encoder finds $\mathbf{V}^{(t)}$ that minimizes the distance $d(\mathbf{U}^{(t)}, \mathbf{V}^{(t)})$ from a codebook $\mathcal{C}^{(t)}$, then computes the residual $\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} - \mathbf{V}^{(t)}$. The t-th codebook $\mathcal{C}^{(t)}$ consists of $2^{nR/L}$ codewords generated by the marginal distribution in Lemma 1:

$$f_{\mathbf{V}^{(t)}}(v) = \frac{\lambda_t^2}{\lambda_{t+1}^2} \cdot \delta(v) + \left(1 - \frac{\lambda_t^2}{\lambda_{t+1}^2}\right) \cdot \frac{\lambda_t}{2} e^{-\lambda_t |v|}$$

Since $\mathbf{U}^{(t+1)}$ is again an i.i.d. Laplacian random sequence with parameter $\lambda_{t+1} = 1/D_t$ (from the conditional probability in Lemma 1), the encoder can keep applying the same steps for Laplacian sources at each iteration. In summary, for $1 \le t \le L-1$, the information-theoretic successive refinement encoder performs the following steps iteratively: (1) find $\mathbf{V}^{(t)} \in \mathcal{C}^{(t)}$ that minimizes $d(\mathbf{U}^{(t)}, \mathbf{V}^{(t)})$; and (2) update $\mathbf{U}$ as $\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} - \mathbf{V}^{(t)}$. The decoder, on the other hand, reconstructs $\hat{\mathbf{U}}^{(t)} = \sum_{\tau=1}^{t} \mathbf{V}^{(\tau)}$ at iteration t. This scheme has a complexity of $L \cdot 2^{nR/L}$ (the total size of the codebooks in L iterations), which is lower than the naive random coding strategy (2^{nR} at once). At the same time, it still achieves the rate-distortion limit, i.e., does not sacrifice the optimality, thanks to successive refinability of Laplacian source. However, the complexity is still exponential in n, which is impractical. We fix this in the next section.

5.2 SuRP Algorithm

The algorithm in Section 5.1 is rate-distortion theoretic optimal with lower complexity thanks to successive refinability, but still impractical due to the exponential size of the codebooks. In this section, we develop a new algorithm SuRP, that enjoys both practicality and optimality. Concretely, SuRP does not require a random codebook or a search for the nearest codeword $\mathbf{V}^{(t)}$ from $\mathbf{U}^{(t)}$ at each iteration, yet still rate-distortion theoretic optimal. With the same initialization $\mathbf{U}^{(1)} = u^n$ and $\lambda_1 = \lambda$, new iterative coding scheme for $1 \le t \le L-1$ is as follows:

(1) Find indices (i, j) such that $\mathbf{U}_i^{(t)} \ge \frac{1}{\lambda_t} \log \frac{n}{2\beta}$ and $\mathbf{U}_j^{(t)} \le -\frac{1}{\lambda_t} \log \frac{n}{2\beta}$. If there are more than one such indices, pick (i, j) randomly. Encode (i, j) as m_t.

(2) Let $\mathbf{V}^{(t)}$ be an n-dimensional all-zero vector except $\mathbf{V}_i^{(t)} = \frac{1}{\lambda_t} \log \frac{n}{2\beta}$ and $\mathbf{V}_j^{(t)} = -\frac{1}{\lambda_t} \log \frac{n}{2\beta}$.

(3) Let $\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} - \mathbf{V}^{(t)}$.

(4) Set $\lambda_{t+1} = \frac{n}{n - 2 \log \frac{n}{2\beta}} \cdot \lambda_t$.

Here, $\beta > 1$ is a tunable parameter. Similar to the algorithm in Section 5.1, the reconstruction at t-th iteration would be $\hat{\mathbf{U}}^{(t)} = \sum_{\tau=1}^{t} \mathbf{V}^{(\tau)}$. We note that the encoder still communicates a sparse vector $\mathbf{V}^{(t)}$ with two nonzero entries by sending $m_t = (i, j)$. We give the pseudocode for SuRP in Appendix E.

This coding scheme is equivalent to the original scheme in Section 5.1, where $\frac{\lambda_t}{\lambda_{t+1}} = \frac{n-2\log\frac{n}{2\beta}}{n}$ for $1 \leq t \leq L-1$ except the fact that the encoder does not do a search over a randomly generated codebook with exponential size, i.e., SuRP is practical. However, there is still an *implicit* codebook $\mathcal{C}^{(t)}$ at every iteration t, which consists of n-dimensional all-zero vectors except for two nonzero elements of values $\pm\frac{1}{\lambda_{t-1}}\log\frac{n}{2\beta}$. The size of this codebook is $n(n-1)$ (not exponential anymore). Since these *implicit* codebooks are not directly generated from the optimal marginal distribution in Lemma 1, it is not obvious that SuRP is rate-distortion theoretic optimal. However, we prove the optimality under some criteria in Section 5.3.

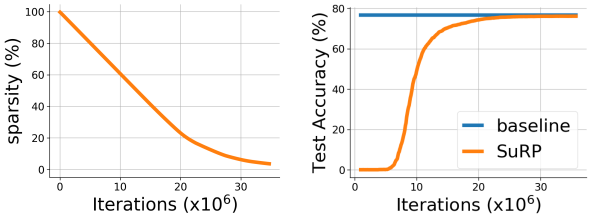
We highlight that our scheme follows a bottom-up approach, in that sparsity in the reconstructed weights starts from 100% at the first iteration and it decreases as the decoder receives new indices from the encoder (see Figure 3(a)). This is similar to the progressive/hierarchical image compression techniques (Lewis and Knowles, 1992; Rabbani, 2002). Similarly, from Figure 3(b), accuracy increases through the iterations.

As a practical issue, when there is no index i or j such that $\mathbf{U}_i^{(t)} \geq \frac{1}{\lambda_{t-1}}\log\frac{n}{2\beta}$ or $\mathbf{U}_j^{(t)} \leq -\frac{1}{\lambda_{t-1}}\log\frac{n}{2\beta}$, the encoder re-estimates λ and sends a refreshed value to the decoder. Obviously, these refreshments must be avoided to preserve optimality. We have seen in our experiments that this is a rare situation (20 refreshments in 20M iterations) and hence has a negligible effect on the overall optimality. In fact, we control the probability of this undesired situation (when there is need for refreshment) with the tunable parameter β. More precisely, the probability that all Laplacian random variables are smaller than $\frac{1}{\lambda}\log\frac{n}{2\beta}$ in magnitude (i.e., no index i or j found) is

$$
\begin{aligned}
&P\left[\max X_i < \frac{1}{\lambda}\log\frac{n}{2\beta} \text{ or } \min X_i > -\frac{1}{\lambda}\log\frac{n}{2\beta}\right] \\
&\leq P\left[\max X_i < \frac{1}{\lambda}\log\frac{n}{2\beta}\right] + P\left[\min X_i > -\frac{1}{\lambda}\log\frac{n}{2\beta}\right] \\
&= 2\left(1-\frac{1}{2}\frac{2\beta}{n}\right)^n \approx 2e^{-\beta}.
\end{aligned} \tag{8}
$$

We set $\beta = \log n$ to bound this probability by $\frac{2}{n}$, which converges to 0 as n increases. We discuss the choice of β in more detail with empirical results in Appendix E.1.

Remark 1. *From the extreme value theory, the maximum of n Laplacian random variables concentrates near $\frac{1}{\lambda}\log\frac{n}{2}$, which is the case of $\beta = 1$. Thus, one iteration of SuRP can be viewed as finding a near-maximum (and minimum) element. From this perspective, magnitude pruning can be viewed as a special case of SuRP.*



(a) Sparsity. (b) Accuracy.

Figure 3: Sparsity and accuracy of the reconstructed ResNet-50 on ImageNet during one cycle of SuRP. Iterations correspond to iterations running inside SuRP.

Remark 2. *SuRP guarantees $\|\mathbf{U}^{(t)}\|_1 \geq \|\mathbf{V}^{(t)}\|_1$ for all t. This implies that the magnitude of weights in $\mathbf{w}$ is always larger than the magnitude of weights in $\hat{\mathbf{w}}$. From Theorem 1, we can say that the ℓ_1 weight distortion of SuRP algorithm is an upper bound to the NN model's output perturbation.*

5.3 Zero-Rate Optimality of SuRP

In this section, we prove that SuRP is a zero-rate optimal compression algorithm. Given that SuRP uses an *implicit* codebook of size $n(n-1)$ at each iteration, the rate is found as $R_n = \frac{\log n(n-1)}{n}$. We note that R_n gets arbitrarily close to zero as n increases. Moreover, the decrement in the distortion at each iteration is given as $D_n = \frac{2}{n\lambda}\log\frac{n}{2\beta_n}$, where $\beta_n = \beta$ as before. We start with the definition of *zero-rate optimality*, which states that a sub-linear number of bits (in our case $\log n(n-1)$ nats) is being used optimally in the rate-distortion theoretic sense.

Definition 1 (Zero-rate optimality)**.** *A scheme with rate R_n, distortion decrement D_n, and distortion-rate function $D(\cdot)$, is zero-rate optimal if $\lim_{n\to\infty} R_n = 0$ and $\lim_{n\to\infty}\frac{D_n}{R_n} = D'(0)$.*

This implies that a zero-rate optimal scheme achieves the "slope" of the distortion-rate function at zero rate $R = 0$. In the case of Laplacian source, this slope is $D'(0) = -\frac{1}{\lambda}$ since the distortion-rate function is $D(R) = \frac{1}{\lambda}e^{-R}$, which can be derived from the rate-distortion function in Lemma 1. Finally, the following theorem states that a single iteration of SuRP is zero-rate optimal.

Theorem 2. *An iteration of SuRP is zero-rate optimal if $\lim_{n\to\infty}\frac{\log 2\beta_n}{\log n(n-1)} = 0$ holds.*

Proof. In an iteration of SuRP, where $R_n = \frac{\log n(n-1)}{n}$

Table 1: Accuracy of VGG-16, ResNet-20, and DenseNet-121 on CIFAR-10. Results are averaged over five runs.

	Pruning Ratio:	93.12%	95.60%	97.19%	98.20%	98.85%	99.53%	99.70%	99.81%	99.88%
VGG-16	Global	91.30	90.80	89.28	85.55	81.56	41.91	31.93	21.87	11.72
	Uniform	91.47	90.78	88.61	84.17	55.68	26.41	16.75	11.58	9.95
	Adaptive	91.54	91.20	90.16	89.44	87.85	84.84	82.41	74.54	24.46
	RiGL	92.34	91.99	91.66	91.15	90.55	88.21	86.73	84.85	81.50
	LAMP	92.24	92.06	91.71	91.66	91.07	89.64	88.75	87.07	84.90
	SuRP (ours)	**92.55**	**92.13**	**91.95**	**91.72**	**91.21**	**90.65**	**89.70**	**87.28**	**85.04**
	Pruning Ratio:	79.03%	86.58%	91.41%	94.50%	96.48%	97.75%	98.56%	99.41%	99.62%
ResNet-20	Global	87.48	86.97	86.29	85.02	83.15	80.52	76.28	47.47	12.02
	Uniform	87.24	86.70	86.09	84.53	82.05	77.19	64.24	20.45	13.35
	Adaptive	87.30	87.00	86.27	85.00	83.23	80.40	76.40	52.06	20.19
	RiGL	87.63	87.49	86.83	85.84	84.08	81.76	78.70	66.42	50.90
	LAMP	87.54	87.12	86.56	85.64	84.18	81.56	78.63	67.01	51.24
	SuRP (ours)	**91.37**	**90.44**	**89.00**	**88.87**	**87.05**	**83.98**	**79.00**	**70.64**	**54.22**
	Pruning Ratio:	94.50%	95.60%	96.48%	97.18%	97.75%	98.20%	98.56%	99.08%	99.26%
DenseNet-121	Global	90.16	89.52	88.83	88.00	86.85	85.32	77.68	49.65	20.96
	Uniform	90.24	89.50	88.44	87.94	86.83	85.00	82.16	66.46	48.71
	Adaptive	90.25	89.70	89.03	88.22	87.40	86.26	84.55	69.25	58.91
	RiGL	90.21	89.79	88.92	88.20	87.25	86.22	84.11	59.06	59.07
	LAMP	90.89	90.11	89.72	89.12	88.39	87.75	86.53	82.92	79.23
	SuRP (ours)	**91.42**	**90.75**	**90.30**	**89.62**	**88.77**	**88.06**	**86.71**	**83.18**	**79.45**
	Pruning Ratio:	59.00%	73.80%	83.20%	89.30%	93.13%	95.60%	97.18%	98.20%	99.26%
EfficientNet-B0	Global	89.66	89.55	88.80	87.64	84.36	79.25	11.09	10.62	10.00
	Uniform	88.99	88.26	86.48	83.40	23.65	10.83	10.00	10.00	10.00
	Adaptive	89.18	88.03	86.71	84.16	36.64	10.45	10.00	10.19	10.00
	RiGL	89.54	90.09	90.01	89.62	88.82	87.08	84.72	81.53	13.40
	LAMP	89.52	89.95	89.97	90.21	89.91	89.79	89.30	88.51	65.76
	SuRP (ours)	**90.96**	**90.94**	**90.89**	**90.75**	**90.31**	**90.08**	**89.88**	**89.02**	**70.76**

and

$$\frac{D_n}{R_n} = -\frac{\frac{2}{\lambda}\log\frac{n}{2\beta_n}}{\log n(n-1)} = -\frac{1}{\lambda}\frac{\log n^2}{\log n(n-1)} + \frac{1}{\lambda}\frac{2\log 2\beta_n}{\log n(n-1)}.$$

If $\lim_{n\to\infty}\frac{\log 2\beta_n}{\log n(n-1)} = 0$, it is clear that $\frac{D_n}{R_n}$ converges to $D'(0) = -\frac{1}{\lambda}$ as n increases. Therefore, SuRP is zero-rate optimal under the condition that $\lim_{n\to\infty}\frac{\log 2\beta_n}{\log n(n-1)} = 0$. □

In Section 5.2, we choose $\beta_n = \log n$ to keep the probability in Eq. 8 small. With this choice of β_n, $\lim_{n\to\infty}\frac{\log 2\beta_n}{\log n(n-1)} = 0$ holds. Therefore, from Theorem 2, our implementation of SuRP is indeed zero-rate optimal.

Remark 3. *In pure information-theoretic compression setting (main concern is not NN compression), similar zero-rate optimal schemes were proposed for Gaussian source under mean squared error (Venkataramanan et al., 2014; No and Weissman, 2016).*

6 Experiments

In this section, we empirically investigate the performance of SuRP compared to recent pruning strategies in terms of accuracy-sparsity tradeoff. We emphasize that the main contribution of our paper is to provide an information-theoretic justification for pruning. SuRP is designed solely to show that an algorithm derived with an information-theoretic approach indeed outputs a pruned model, as suggested by our findings. This also provides theoretical support for the recent success of pruning strategies.

For our NN compression experiments, we consider two image datasets: CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009). For CIFAR-10, we use four architectures: VGG-16 (Simonyan and Zisserman, 2014), ResNet-20 (He et al., 2016), DenseNet-121 (Iandola et al., 2014), and EfficientNet-B0 (Tan and Le, 2019). For ImageNet, we use ResNet-50 (He et al., 2016; Paszke et al., 2019). We give additional details on model architectures and hyperparameters in Appendix J. We present experimental results averaged

over 3-5 runs (see Appendix K for complete results).

NN Compression/Pruning: In Tables 1 and 2, we compare our scheme with the recent pruning papers. We apply iterative pruning, meaning that we apply SuRP in repeating cycles (see Appendix K for details). As baselines, we consider Global (Morcos et al., 2019), Uniform (Zhu and Gupta, 2017), and Adaptive (Gale et al., 2019) pruning techniques and LAMP (Lee et al., 2021). Additionally, we include comparisons to recent works on weight rewinding and dynamic sparsity, in particular SNIP (Lee et al., 2018), DSR (Mostafa and Wang, 2019), SNFS (Dettmers and Zettlemoyer, 2019), and RiGL (Evci et al., 2020).

We present the performance of pruned VGG-16, ResNet-20, and DenseNet-121 architectures on CIFAR-10 in Table 1 and ResNet-50 on ImageNet in Table 2 . As can be seen from Table 1, SuRP outperforms prior work in all sparsity levels. From Table 2, SuRP and Adaptive pruning perform similarly (with $\pm 0.06\%$ difference), and they both outperform other baselines.

Table 2: Accuracy of ResNet-50 on ImageNet (3 runs).

Pruning Ratio:	80%	90%
Adaptive	75.60	73.90
SNIP	72.00	67.20
DSR	73.30	71.60
SNFS	74.90	72.90
RiGL	74.60	72.00
LAMP	74.96	73.22
SuRP (ours)	**75.54**	**73.95**

We provide a comparison for lower pruning ratios in Appendix K.

7 Discussion and Conclusion

In this work, we connected two lines of research, namely, data compression and NN compression. We investigated the theoretical tradeoff between the compression ratio and output perturbation of NN models, and found out that the rate-distortion theoretic formulation introduces a theoretical foundation for pruning. Guided by this, we developed a NN compression algorithm that outputs a pruned model and outperforms prior work.

We note that our algorithm SuRP has an additional advantage in optimizing the bitrate of the model thanks to the rate-distortion theoretic basis of our approach. In particular, the decoder has only access to a list of indices, and these indices represent the whole (compressed) model – more efficiently than describing the precise values of surviving weights. However, our current implementation does not exploit this efficiency to the full extent due to retraining steps after each pruning iteration. Like many, we will also look for ways to prune NN models without a retraining step afterward. That way, SuRP can be improved to provide a better accuracy-bitrate tradeoff, together with the already demonstrated sparsity-accuracy gain. We give more details on this and share experimental results in Appendices H and K. Finally, to give an idea about the bitrate efficiency of SuRP, we apply it for compressing gradients in a federated learning setting. Since the compressed gradients are not exposed to fine-tuning (like retraining in pruning), SuRP provides a substantial improvement on the bitrate compared to prior work. We elaborate more on this in the next paragraph.

Compression for Federated Learning (FL): FL is a distributed training setting where edge devices are responsible for doing local training and sending local gradients to a central server (Kairouz et al., 2019). Given the resource limitations of edge devices, gradient communication is a significant bottleneck in FL, and gradient compression is crucial (Konečný et al., 2016). We show in Appendix I that Laplacian distribution is a good fit for NN gradients. Therefore, SuRP is applicable to this problem as well. Our preliminary experiments with LeNet-5-Caffe on MNIST (LeCun et al., 2010) compare SuRP with DGC (Lin et al., 2017) and rTop-k (Barnes et al., 2020). We compute the communication budget for prior work by assuming a naive encoding with $k(\log n + 32)$ bits (n is the model size) since no other method is provided. With the same sparsity ratio 99.9%, DGC achieves 98.5% accuracy with **2.05KB** of budget, rTop-k achieves 99.1% accuracy with **2.05KB** of budget, and SuRP achieves 99.1% accuracy with **218B** of budget. Thus, SuRP provides $10\times$ times improvement in the gradients' compression rate while achieving the same accuracy as rTop-k.

Limitations and Broader Impact: When we evaluated our strategy, we only considered accuracy as a metric. However, compression might have an impact on other properties of the model as well, such as fairness, as pointed out by Hooker et al. (2020). We agree that this issue deserves more attention from the community.

The codebase for this work is open-sourced at `https://github.com/BerivanIsik/SuRP`.

8 Acknowledgement

This work was supported in part by a Sony Stanford Graduate Fellowship, a National Science Foundation (NSF) award, a Meta (formerly Facebook) resarch award, and a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A105956711).

References

Aji, A. F. and Heafield, K. (2017). Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*.

Ballé, J., Laparra, V., and Simoncelli, E. P. (2016). End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*.

Banner, R., Hubara, I., Hoffer, E., and Soudry, D. (2018). Scalable methods for 8-bit training of neural networks. In *Advances in neural information processing systems*, pages 5145–5153.

Barnes, L. P., Inan, H. A., Isik, B., and Özgür, A. (2020). rtop-k: A statistical estimation approach to distributed sgd. *IEEE Journal on Selected Areas in Information Theory*, 1(3):897–907.

Berger, T. (2003). Rate-distortion theory. *Wiley Encyclopedia of Telecommunications*.

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. (2020). What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*.

Carreira-Perpinan, M. A. and Idelbayev, Y. (2018). "learning-compression" algorithms for neural net pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8532–8541.

Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. (2021). Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations, 2021a. URL https://openreview. net/forum*.

Choi, Y., El-Khamy, M., and Lee, J. (2020). Universal deep neural network compression. *IEEE Journal of Selected Topics in Signal Processing*.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA.

Cun, Y. L., Denker, J. S., and Solla, S. A. (1990). *Optimal Brain Damage*, page 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Dai, B., Zhu, C., Guo, B., and Wipf, D. (2018). Compressing neural networks using the variational information bottleneck. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1135–1144. PMLR.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Dettmers, T. and Zettlemoyer, L. (2019). Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*.

Elsen, E., Dukhan, M., Gale, T., and Simonyan, K. (2020). Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14629–14638.

Equitz, W. H. and Cover, T. M. (1991). Successive refinement of information. *IEEE Transactions on Information Theory*, 37(2):269–275.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. (2020). Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR.

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

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*.

Gale, T., Elsen, E., and Hooker, S. (2019). The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Gallager, R. and Van Voorhis, D. (1975). Optimal source codes for geometrically distributed integer alphabets (corresp.). *IEEE Transactions on Information theory*, 21(2):228–230.

Gao, W., Liu, Y.-H., Wang, C., and Oh, S. (2019). Rate distortion for model compression: From theory to practice. In *International Conference on Machine Learning*, pages 2102–2111. PMLR.

Golomb, S. (1966). Run-length encodings (corresp.). *IEEE transactions on information theory*, 12(3):399–401.

Grünwald, P. D. and Grunwald, A. (2007). *The minimum description length principle*. MIT press.

Guo, Y., Yao, A., and Chen, Y. (2016). Dynamic network surgery for efficient dnns. In *Advances in neural information processing systems*, pages 1379–1387.

Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*.

Hassibi, B., Stork, D. G., Wolff, G., and Watanabe, T. (1993). Optimal brain surgeon: Extensions and performance comparisons. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, page 263–270, San Francisco, CA, USA.

Havasi, M., Peharz, R., and Hernández-Lobato, J. M. (2019). Minimal random code learning: Getting bits

back from compressed model parameters. In *International Conference on Learning Representations (ICLR)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hooker, S., Moorosi, N., Clark, G., Bengio, S., and Denton, E. (2020). Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*.

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., and Keutzer, K. (2014). Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*.

Idelbayev, Y. and Carreira-Perpinan, M. A. (2020). Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Idelbayev, Y., Molchanov, P., Shen, M., Yin, H., Carreira-Perpinan, M. A., and Alvarez, J. M. (2021). Optimal quantization using scaled codebook. In *Proc. of the 2021 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'21), Virtual*.

Ioannou, Y., Robertson, D., Shotton, J., Cipolla, R., and Criminisi, A. (2015). Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*.

Isik, B., Choi, K., Zheng, X., Weissman, T., Ermon, S., Wong, H. S. P., and Alaghi, A. (2021). Neural network compression for noisy storage devices. *arXiv preprint arXiv:2102.07725*.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

Jung, S., Son, C., Lee, S., Son, J., Han, J.-J., Kwak, Y., Hwang, S. J., and Choi, C. (2019). Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4350–4359.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2019). Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Konečný, J., McMahan, H. B., Yu, F. X., Richtarik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*.

Koshelev, V. N. (1980). Hierarchical coding of discrete sources. *Problemy peredachi informatsii*, 16(3):31–49.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

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., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database.

Lee, J., Park, S., Mo, S., Ahn, S., and Shin, J. (2021). Layer-adaptive sparsity for the magnitude-based pruning. *International Conference on Learning Representations*.

Lee, N., Ajanthan, T., and Torr, P. H. (2018). Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Lewis, A. S. and Knowles, G. (1992). Image compression using the 2-d wavelet transform. *IEEE Transactions on image Processing*, 1(2):244–250.

Li, F., Zhang, B., and Liu, B. (2016). Ternary weight networks. *arXiv preprint arXiv:1605.04711*.

Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., and Doermann, D. (2019). Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. (2017). Deep gradient compression: Reducing the communication bandwidth for distributed training. *International Conference on Learning Representations (ICLR)*.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. (2018). Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Louizos, C., Ullrich, K., and Welling, M. (2017a). Bayesian compression for deep learning. *arXiv preprint arXiv:1705.08665*.

Louizos, C., Welling, M., and Kingma, D. P. (2017b). Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. (2018). Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12.

Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

Morcos, A. S., Yu, H., Paganini, M., and Tian, Y. (2019). One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv preprint arXiv:1906.02773*.

Mostafa, H. and Wang, X. (2019). Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR.

No, A., Ingber, A., and Weissman, T. (2016). Strong successive refinability and rate-distortion-complexity tradeoff. *IEEE Transactions on Information Theory*, 62(6):3618–3635.

No, A. and Weissman, T. (2016). Rateless lossy compression via the extremes. *IEEE transactions on information theory*, 62(10):5484–5495.

Oktay, D., Ballé, J., Singh, S., and Shrivastava, A. (2019). Scalable model compression by entropy penalized reparameterization. In *International Conference on Learning Representations (ICLR)*.

Park, S., Lee, J., Mo, S., and Shin, J. (2020). Lookahead: A far-sighted alternative of magnitude-based pruning. *International Conference on Learning Representations (ICLR)*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Peng, H., Wu, J., Chen, S., and Huang, J. (2019). Collaborative channel pruning for deep networks. In *International Conference on Machine Learning*, pages 5113–5122. PMLR.

Polino, A., Pascanu, R., and Alistarh, D. (2018). Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.

Rabbani, M. (2002). Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286.

Renda, A., Frankle, J., and Carbin, M. (2020). Comparing fine-tuning and rewinding in neural network pruning. In *International Conference on Learning Representations*.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., and Ramabhadran, B. (2013). Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE.

Salomon, D. (2004). *Data compression: the complete reference*. Springer Science & Business Media.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

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

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

Stock, P., Fan, A., Graham, B., Grave, E., Gribonval, R., Jegou, H., and Joulin, A. (2021). Training with quantization noise for extreme model compression. In *International Conference on Learning Representations*.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR.

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

Venkataramanan, R., Sarkar, T., and Tatikonda, S. (2014). Lossy compression via sparse linear regression: Computationally efficient encoding and decoding. *IEEE transactions on information theory*, 60(6):3265–3278.

Verdu, S. (1996). The exponential distribution in information theory. *Problemy peredachi informatsii*, 32(1):100–111.

Wang, H., Sievert, S., Liu, S., Charles, Z. B., Papailiopoulos, D. S., and Wright, S. (2018). Atomo: Communication-efficient learning via atomic sparsification. In *NeurIPS*.

Wang, J., Bao, W., Sun, L., Zhu, X., Cao, B., and Philip, S. Y. (2019a). Private model compression via knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1190–1197.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. (2019b). Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620.

Wangni, J., Wang, J., Liu, J., and Zhang, T. (2018). Gradient sparsification for communication-efficient distributed optimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 1299–1309. Curran Associates, Inc.

Wiedemann, S., Kirchhoffer, H., Matlage, S., Haase, P., Marban, A., Marinč, T., Neumann, D., Nguyen, T., Schwarz, H., Wiegand, T., Marpe, D., and Samek, W. (2020). Deepcabac: A universal compression algorithm for deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):700–714.

Xiao, X., Wang, Z., and Rajasekaran, S. (2019). Autoprune: Automatic network pruning by regularizing auxiliary parameters. *Advances in neural information processing systems*, 32.

Young, S. I., Zhe, W., Taubman, D., and Girod, B. (2020). Transform quantization for cnn compression. *arXiv preprint arXiv:2009.01174*.

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., and Davis, L. S. (2018). Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203.

Zhao, C., Ni, B., Zhang, J., Zhao, Q., Zhang, W., and Tian, Q. (2019). Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2780–2789.

Zhe, W., Lin, J., Aly, M. S., Young, S., Chandrasekhar, V., and Girod, B. (2021). Rate-distortion optimized coding for efficient cnn compression. In *2021 Data Compression Conference (DCC)*, pages 253–262. IEEE.

Zhu, M. and Gupta, S. (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

Supplementary Material: An Information-Theoretic Justification for Model Pruning

A Proof of Theorem 1

In this section, we provide the proof of Theorem 1. The fully connected d-layer NN model with 1-Lipschitz activations $\sigma(\cdot)$ is given by

$$f(\mathbf{x};\mathbf{w}) = \mathbf{w}^{(d)}\sigma(\mathbf{w}^{(d-1)}\sigma(\cdots\mathbf{w}^{(2)}\sigma(\mathbf{w}^{(1)}\mathbf{x}))).$$

We let $\mathbf{w}^{(1:i)} = \{\mathbf{w}^{(1)},\ldots,\mathbf{w}^{(i)}\}$ for $1 \le i \le d$ where $\mathbf{w}^{(1:d)} = \mathbf{w}$. Furthermore, we define the first i layer of the network by

$$f(\mathbf{x};\mathbf{w}^{(1:i)}) = \mathbf{w}^{(i)}\sigma(\mathbf{w}^{(i-1)}\sigma(\cdots\mathbf{w}^{(2)}\sigma(\mathbf{w}^{(1)}\mathbf{x}))).$$

Then, the output perturbation is bounded by

$$\begin{aligned}
&\|f(\mathbf{x};\mathbf{w}^{(1:d)}) - f(\mathbf{x};\hat{\mathbf{w}}^{(1:d)})\|_1 \\
=&\|\mathbf{w}^{(d)}\sigma(f(\mathbf{x};\mathbf{w}^{(1:d-1)})) - \hat{\mathbf{w}}^{(d)}\sigma(f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)}))\|_1 \\
\le&\|\mathbf{w}^{(d)}\sigma(f(\mathbf{x};\mathbf{w}^{(1:d-1)})) - \hat{\mathbf{w}}^{(d)}\sigma(f(\mathbf{x};\mathbf{w}^{(1:d-1)}))\|_1 \\
&+\|\hat{\mathbf{w}}^{(d)}\sigma(f(\mathbf{x};\mathbf{w}^{(1:d-1)})) - \hat{\mathbf{w}}^{(d)}\sigma(f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)}))\|_1 &(9)\\
\le&\|\mathbf{w}^{(d)} - \hat{\mathbf{w}}^{(d)}\|_1\cdot\|\sigma(f(\mathbf{x};\mathbf{w}^{(1:d-1)}))\|_1 + \|\hat{\mathbf{w}}^{(d)}\|_1\cdot\|\sigma(f(\mathbf{x};\mathbf{w}^{(1:d-1)})) - \sigma(f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)}))\|_1 &(10)\\
\le&\|\mathbf{w}^{(d)} - \hat{\mathbf{w}}^{(d)}\|_1\cdot\|f(\mathbf{x};\mathbf{w}^{(1:d-1)})\|_1 + \|\hat{\mathbf{w}}^{(d)}\|_1\cdot\|f(\mathbf{x};\mathbf{w}^{(1:d-1)}) - f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1 &(11)\\
\le&\|\mathbf{w}^{(d)} - \hat{\mathbf{w}}^{(d)}\|_1\cdot\prod_{l=1}^{d-1}\|\mathbf{w}^{(l)}\|_1\cdot\|\mathbf{x}\|_1 + \|\hat{\mathbf{w}}^{(d)}\|_1\cdot\|f(\mathbf{x};\mathbf{w}^{(1:d-1)}) - f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1 &(12)\\
\le&\|\mathbf{w}^{(d)} - \hat{\mathbf{w}}^{(d)}\|_1\cdot\prod_{l=1}^{d-1}\|\mathbf{w}^{(l)}\|_1 + \|\hat{\mathbf{w}}^{(d)}\|_1\cdot\|f(\mathbf{x};\mathbf{w}^{(1:d-1)}) - f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1 &(13)\\
\le&\|\mathbf{w}^{(d)} - \hat{\mathbf{w}}^{(d)}\|_1\cdot\prod_{l=1}^{d-1}\|\mathbf{w}^{(l)}\|_1 + \|\mathbf{w}^{(d)}\|_1\cdot\|f(\mathbf{x};\mathbf{w}^{(1:d-1)}) - f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1 &(14)
\end{aligned}$$

where Eq. 9 is due to triangle inequality, and Eq. 10 holds from the property of ℓ_1-norm (and induced norm). Eq. 11 is from 1-Lipshitzness of activation $\sigma(\cdot)$, i.e., $\|\sigma(\mathbf{x})\|_1 \le \|\mathbf{x}\|_1$. Eq. 12 holds from the following lemma.

Lemma 2. *For all $1 \le i \le d$, we have $\|f(\mathbf{x};\mathbf{w}^{(1:i)})\|_1 \le \prod_{j=1}^{i}\|\mathbf{w}^{(j)}\|_1\cdot\|\mathbf{x}\|_1$.*

Proof. From the property of ℓ_1-norm, we have

$$\begin{aligned}
\|f(\mathbf{x};\mathbf{w}^{(1:i)})\|_1 \le&\|\mathbf{w}^{(i)}\|_1\cdot\|\sigma(f(\mathbf{x};\mathbf{w}^{(1:i-1)}))\|_1 &(15)\\
\le&\|\mathbf{w}^{(i)}\|_1\cdot\|f(\mathbf{x};\mathbf{w}^{(1:i-1)})\|_1 &(16)
\end{aligned}$$

where the last inequality is due to 1-Lipshitzness of σ. Then, we can keep applying the same inequality, which concludes the proof. □

Eq. 13 follows by the constraint $\|\mathbf{x}\|_1 \le 1$ in Theorem 1. Finally Eq. 14 is from the assumption $\|\hat{\mathbf{w}}^{(l)}\|_1 \le \|\mathbf{w}^{(l)}\|_1$ for all $1 \le l \le d$.

Thus, we have

$$\left(\prod_{l=1}^{d}\frac{1}{\|\mathbf{w}^{(l)}\|_1}\right)\|f(\mathbf{x};\mathbf{w}^{(1:d)})-f(\mathbf{x};\hat{\mathbf{w}}^{(1:d)})\|_1$$
$$\leq\frac{\|\mathbf{w}^{(d)}-\hat{\mathbf{w}}^{(d)}\|_1}{\|\mathbf{w}^{(d)}\|_1}+\left(\prod_{l=1}^{d-1}\frac{1}{\|\mathbf{w}^{(l)}\|_1}\right)\|f(\mathbf{x};\mathbf{w}^{(1:d-1)})-f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1. \tag{17}$$

We can repeat the same procedure, and get

$$\left(\prod_{l=1}^{d}\frac{1}{\|\mathbf{w}^{(l)}\|_1}\right)\|f(\mathbf{x};\mathbf{w}^{(1:d)})-f(\mathbf{x};\hat{\mathbf{w}}^{(1:d)})\|_1\leq\sum_{l=1}^{d}\frac{\|\mathbf{w}^{(l)}-\hat{\mathbf{w}}^{(l)}\|_1}{\|\mathbf{w}^{(l)}\|_1}. \tag{18}$$

This completes the proof.

B Modified Theorem 1

In this section, we provide a symmetric version of Theorem 1, which essentially implies the same upper bound on the output perturbation without requiring the additional condition of $\|\mathbf{w}\|_1\geq\|\hat{\mathbf{w}}\|_1$.

Theorem 3. *Suppose $f(\cdot;\mathbf{w})$ is a fully-connected NN model with d layers and 1-Lipschitz activations $\sigma(\cdot)$ such that $\sigma(0)=0$, e.g., ReLU. Let $\hat{\mathbf{w}}$ be the reconstructed weights (after compression) where all layers are subject to compression. Then, we have the following bound on the output perturbation:*

$$\sup_{\|x\|_1\leq 1}\|f(\mathbf{x},\mathbf{w})-f(\mathbf{x},\hat{\mathbf{w}})\|_1\leq\left(\sum_{l=1}^{d}\frac{\|\mathbf{w}^{(l)}-\hat{\mathbf{w}}^{(l)}\|_1}{\max\{\|\hat{\mathbf{w}}^{(l)}\|_1,\|\mathbf{w}^{(l)}\|_1\}}\right)\left(\prod_{k=1}^{d}\max\{\|\hat{\mathbf{w}}^{(k)}\|_1,\|\mathbf{w}^{(k)}\|_1\}\right). \tag{19}$$

By rearranging the terms in Eq. 19, we get the following relation:

$$\sup_{\|x\|_1\leq 1}\frac{\|f(\mathbf{x},\mathbf{w})-f(\mathbf{x},\hat{\mathbf{w}})\|_1}{\prod_{k=1}^{d}\max\{\|\hat{\mathbf{w}}^{(k)}\|_1,\|\mathbf{w}^{(k)}\|_1\}}\leq\left(\sum_{l=1}^{d}\frac{\|\mathbf{w}^{(l)}-\hat{\mathbf{w}}^{(l)}\|_1}{\max\{\|\hat{\mathbf{w}}^{(l)}\|_1,\|\mathbf{w}^{(l)}\|_1\}}\right), \tag{20}$$

which implies that the normalized output perturbation is bounded by the normalized weight differences. With the additional condition of $\|\mathbf{w}\|_1\geq\|\hat{\mathbf{w}}\|_1$, we can simply recover Theorem 1 from Theorem 3.

$$\sup_{\|x\|_1\leq 1}\frac{\|f(\mathbf{x},\mathbf{w})-f(\mathbf{x},\hat{\mathbf{w}})\|_1}{\prod_{k=1}^{d}\max\{\|\hat{\mathbf{w}}^{(k)}\|_1,\|\mathbf{w}^{(k)}\|_1\}}\leq\left(\sum_{l=1}^{d}\frac{\|\mathbf{w}^{(l)}-\hat{\mathbf{w}}^{(l)}\|_1}{\max\{\|\hat{\mathbf{w}}^{(l)}\|_1,\|\mathbf{w}^{(l)}\|_1\}}\right) \tag{21}$$
$$\leq\sum_{l=1}^{d}\frac{\|\mathbf{w}^{(l)}-\hat{\mathbf{w}}^{(l)}\|_1}{\|\mathbf{w}^{(l)}\|_1}, \tag{22}$$

which is compatible with the rest of our results. The proof of Theorem 3 is almost identical to the proof of Theorem 1.

Proof of Theorem 3. Since Eq. 13 still holds without the additional condition $\|\mathbf{w}\|_1\geq\|\hat{\mathbf{w}}\|_1$,

$$\|f(\mathbf{x};\mathbf{w}^{(1:d)})-f(\mathbf{x};\hat{\mathbf{w}}^{(1:d)})\|_1$$
$$\leq\|\mathbf{w}^{(d)}-\hat{\mathbf{w}}^{(d)}\|_1\cdot\prod_{l=1}^{d-1}\|\mathbf{w}^{(l)}\|_1+\|\hat{\mathbf{w}}^{(d)}\|_1\cdot\|f(\mathbf{x};\mathbf{w}^{(1:d-1)})-f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1 \tag{23}$$
$$\leq\|\mathbf{w}^{(d)}-\hat{\mathbf{w}}^{(d)}\|_1\cdot\prod_{l=1}^{d-1}\max\{\|\mathbf{w}^{(l)}\|_1,\|\hat{\mathbf{w}}^{(l)}\|_1\}+\max\{\|\mathbf{w}^{(d)}\|_1,\|\hat{\mathbf{w}}^{(d)}\|_1\}\cdot\|f(\mathbf{x};\mathbf{w}^{(1:d-1)})-f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1, \tag{24}$$

which implies

$$\frac{\|f(\mathbf{x};\mathbf{w}^{(1:d)}) - f(\mathbf{x};\hat{\mathbf{w}}^{(1:d)})\|_1}{\prod_{l=1}^{d}\max\{\|\mathbf{w}^{(l)}\|_1, \|\hat{\mathbf{w}}^{(l)}\|_1\}} \leq \frac{\|\mathbf{w}^{(d)} - \hat{\mathbf{w}}^{(d)}\|_1}{\max\{\|\mathbf{w}^{(d)}\|_1, \|\hat{\mathbf{w}}^{(d)}\|_1\}} + \frac{\|f(\mathbf{x};\mathbf{w}^{(1:d-1)}) - f(\mathbf{x};\hat{\mathbf{w}}^{(1:d-1)})\|_1}{\prod_{l=1}^{d-1}\max\{\|\mathbf{w}^{(l)}\|_1, \|\hat{\mathbf{w}}^{(l)}\|_1\}}. \tag{25}$$

Similar to the proof of Theorem 1, we can recursively apply the above inequality to obtain Eq. 20. □

C Density Estimation for Neural Network Parameters without Normalization

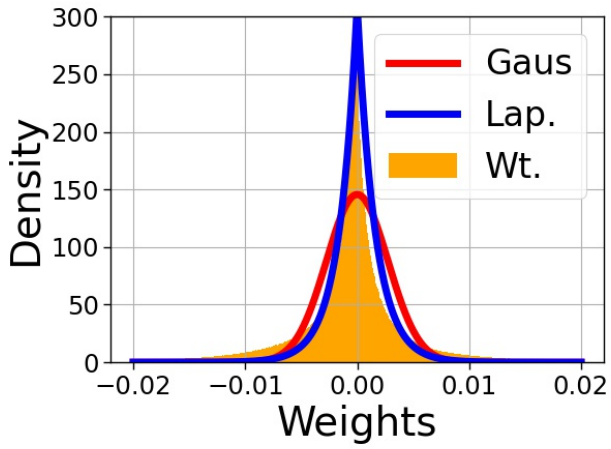


Figure 4: Weight Density of ResNet-18 (trained on CIFAR-10) before normalization.

In Section 4, we have justified our assumption of Laplacian distribution over normalized NN weights through density plots for three distinct architectures. We have also emphasized that Laplacian would be a good fit for unnormalized NN weights as well. We give the density plots of unnormalized weights of ResNet-18 in Figure 4 to justify our claim empirically. This claim implies that SuRP can also be applied to NNs without normalization and it would achieve rate-distortion theoretic optimal performance for reconstructing the NN weights back. However, recall from Theorem 1 that ℓ_1 distortion of normalized weights upper bounds the output perturbation. Since we care more about maintaining the outputs rather than the weights themselves, we have applied SuRP after the normalization.

We have additionally observed that weights of layers closer to the input tend to follow a Gaussian-like distribution. In contrast, the weights of layers closer to the output behave like Laplacian random variables. Since the last layers have larger number of parameters in the architectures used in this work, we see a Laplacian distribution over the weights globally. Therefore, different pruning strategies might be necessary for layers with Gaussian and Laplacian behaviour for a layer by layer pruning approach.

D Proof of Lemma 1

In this section, we briefly describe the proof outline of Lemma 1 in Section 4, which is provided in (Berger, 2003). Consider the Laplacian source $U \sim P_U$ with parameter λ, and the target distortion D satisfies $0 \leq D \leq 1/\lambda$. Then,

$$\begin{aligned} R(D) &= \min_{\mathbb{E}[d(U,\hat{U})]\leq D} I(U;\hat{U}) \\ &= \inf_{\mathbb{E}[|U-\hat{U}|]\leq D} I(U;\hat{U}). \end{aligned}$$

Let Q be another conditional distribution where $Q_{U|\hat{U}}(u|\hat{u}) = \frac{1}{D}e^{-|u-\hat{u}|/D}$. Then,

$$\begin{aligned}
I(U;\hat{U}) =& D_{KL}(P_{U|\hat{U}}\|P_U|P_{\hat{U}}) \\
=& D_{KL}(P_{U|\hat{U}}\|Q_{U|\hat{U}}|P_{\hat{U}}) + \mathbb{E}_{P_{U,\hat{U}}}\left[\log\frac{Q_{U|\hat{U}}(U|\hat{U})}{P_U(U)}\right] \\
\geq& \mathbb{E}_{P_{U,\hat{U}}}\left[\log\frac{Q_{U|\hat{U}}(U|\hat{U})}{P_U(U)}\right] \qquad (26)\\
=& -\log(\lambda D) - \frac{1}{D}\mathbb{E}[|U-\hat{U}|] + \lambda\mathbb{E}[|U|] \\
\geq& -\log(\lambda D) \qquad (27)
\end{aligned}$$

where Eq. 26 is due to nonnegativity of KL divergence, and Eq. 27 is because $\mathbb{E}[|U|] = \frac{1}{\lambda}$ and $\mathbb{E}[|U-\hat{U}|] \leq D$. This implies that $R(D) \geq -\log(\lambda D)$. We note that we followed a technique inspired by Verdu's proof for rate-distortion function of exponential source (Verdu, 1996). The same lower bound can also be achieved via Shannon lower bound (SLB) (Shannon, 1959).

On the other hand, we need to show that the lower bound $R(D) \geq -\log(\lambda D)$ is indeed tight. Let V be a mixture of point measure and Laplacian random variable, where the probability density function is given by

$$P_V(v) = \lambda^2 D^2 \cdot \delta(v) + (1-\lambda^2 D^2)\cdot\frac{\lambda}{2}e^{-\lambda|v|}.$$

We further let N be a Laplacian random variable with parameter $1/D$, where V and N are independent. Then, the Laplace transform of P_V and P_N are given by

$$\begin{aligned}
\mathbb{E}[e^{-sV}] =& \lambda^2 D^2 + (1-\lambda^2 D^2)\frac{\lambda^2}{\lambda^2 - s^2} \\
\mathbb{E}[e^{-sN}] =& \frac{1/D^2}{1/D^2 - s^2}.
\end{aligned}$$

Consider the sum of two random variables $V+N$. Since they are independent, Laplace transform of the density of $V+N$ is a product of the above two terms.

$$\begin{aligned}
\mathbb{E}[e^{-s(V+N)}] =& \mathbb{E}[e^{-sV}]\cdot\mathbb{E}[e^{-sN}] \\
=& \frac{\lambda^2}{\lambda^2 - s^2}.
\end{aligned}$$

Since it coincides with the Laplace transform of P_U, we conclude that $U \overset{(d)}{=} V+N$. Thus, by letting $U = V+N$, we obtain the conditional distribution $Q_{U|V}(u|v) = \frac{1}{D}e^{-|u-v|/D}$. It is clear that $Q_{U|V}$ satisfies the equality conditions in Eq. 26 and Eq. 27, and therefore it achieves the lower bound $I(U;\hat{U}) = -\log(\lambda D)$ with $\hat{U} = V$.

To sum, the optimal rate-distortion tradeoff is $R(D) = -\log(\lambda D)$ and it can be achieved with a reconstruction that follows

$$P_V(v) = \lambda^2 D^2 \cdot \delta(v) + (1-\lambda^2 D^2)\cdot\frac{\lambda}{2}e^{-\lambda|v|}. \qquad (28)$$

E Algorithms

We give the algorithm described in Section 5 in Algorithm 1. For the experiments in Section 6, we slightly modified Algorithm 1 and used Algorithm 2.

As mentioned in Section 6, these two algorithms are equivalent except the fact that Algorithm 2 applies the same compression scheme after taking the absolute value of the normalized weights. Furthermore, Algorithm 2 is rate-distortion theoretic optimal too. To see this, it is enough to follow the same steps in Sections 4 and 5 for exponential source instead of Laplacian source since the magnitude of Laplacian source sequence follows an

Algorithm 1 SuRP

Hyperparameters: β
Inputs: weights $w_1, \ldots, w_n$ in d layers
Output: reconstructed weights $w_1^{recon}, \ldots, w_n^{recon}$

Normalization:
for $l = 1, \ldots, d$ **do**
  $u^{(l)} \leftarrow \frac{w^{(l)}}{\|w^{(l)}\|_1}$
end for

$(u_1^{recon}, \ldots, u_n^{recon}) \leftarrow 0$
$\lambda \leftarrow \mathsf{ParamEst}((u_1, \ldots, u_n))$
Encoder sends λ to the Decoder.
for $t = 1, \ldots, L$ **do**
  Encoder:
  $m_{max} \leftarrow$ (indices of the components in $(u_1, \ldots, u_n)$ that are larger than $\frac{1}{\lambda} \cdot \log \frac{n}{2\beta}$.)
  $m_{min} \leftarrow$ (indices of the components in $(u_1, \ldots, u_n)$ that are smaller than $-\frac{1}{\lambda} \cdot \log \frac{n}{2\beta}$.)
  if m_{max} or m_{min} is empty **then**
    $\lambda \leftarrow \mathsf{ParamEst}((u_1, \ldots, u_n))$
    sends λ to the Decoder.
  end if
  $m_1 \leftarrow$ (a random index from m_{max})
  $m_{-1} \leftarrow$ (a random index from m_{min})
  sends m_1 and m_{-1} to the Decoder.
  $u_{m_1} = u_{m_1} - \frac{1}{\lambda} \cdot \log \frac{n}{2\beta}$
  $u_{m_{-1}} = u_{m_{-1}} + \frac{1}{\lambda} \cdot \log \frac{n}{2\beta}$
  $\lambda \leftarrow \frac{n}{n - 2\log \frac{n}{2\beta}} \cdot \lambda$
  Decoder:
  receives m_1 and m_{-1} from the Encoder.
  $u_{m_1}^{recon} = u_{m_1}^{recon} + \frac{1}{\lambda} \cdot \log \frac{n}{2\beta}$
  $u_{m_{-1}}^{recon} = u_{m_{-1}}^{recon} - \frac{1}{\lambda} \cdot \log \frac{n}{2\beta}$
  $\lambda \leftarrow \frac{n}{n - 2\log \frac{n}{2\beta}} \cdot \lambda$
end for
$w_1^{recon}, \ldots, w_n^{recon} \leftarrow$ (denormalize $u_1^{recon}, \ldots, u_n^{recon}$.)

$\mathsf{ParamEst}((u_1, \ldots, u_n))$:
  $1/\lambda \leftarrow$ mean of $(|u_1|, \ldots, |u_n|)$
  return λ

exponential distribution. We now give the rate-distortion function for exponential source (magnitude of normalized weights). We consider i.i.d. exponential source sequence $u_1, \ldots, u_n$ with distribution $f_{exp}(u; \lambda) = \lambda e^{-\lambda u}$ for $u \geq 0$, reconstruction $v_1, \ldots, v_n$, and one-sided ℓ_1 distortion given by:

$$d(u, v) = \begin{cases} u - v, & \text{if } u \geq v \\ \infty, & \text{otherwise.} \end{cases}$$

Then, the rate-distortion function is given in Lemma 3:

Lemma 3. *(Verdu, 1996) The rate-distortion function for an exponential source with one-sided distortion is given by*

$$R(D) = \begin{cases} -\log(\lambda D), & 0 \leq D \leq \frac{1}{\lambda} \\ 0, & D > \frac{1}{\lambda} \end{cases} \tag{29}$$

Algorithm 2 SuRP-modified

Hyperparameters: β
Inputs: weights $w_1, \dots, w_n$ in d layers
Output: reconstructed weights $w_1^{recon}, \dots, w_n^{recon}$

Normalization:
for $l = 1, \dots, d$ **do**
  $u^{(l)} \leftarrow \frac{|w^{(l)}|}{\|w^{(l)}\|_1}$
end for

$(u_1^{recon}, \dots, u_n^{recon}) \leftarrow 0$
$\lambda \leftarrow \mathsf{ParamEst}((u_1, \dots, u_n))$
Encoder sends λ to the Decoder.
for $t = 1, \dots, L$ **do**
  Encoder:
  $m_{inds} \leftarrow$ (indices of the components in $(u_1, \dots, u_n)$ that are larger than $\frac{1}{\lambda} \cdot \log \frac{n}{\beta}$.)
  if m_{inds} is empty **then**
    $\lambda \leftarrow \mathsf{ParamEst}((u_1, \dots, u_n))$
    sends λ to the Decoder.
  end if
  $m \leftarrow$ (a random index from m_{inds})
  sends m to the Decoder.
  $u_m = u_m - \frac{1}{\lambda} \cdot \log \frac{n}{\beta}$
  $\lambda \leftarrow \frac{n}{n - \log \frac{n}{\beta}} \cdot \lambda$
  Decoder:
  receives m from the Encoder.
  $u_m^{recon} = u_m^{recon} + \frac{1}{\lambda} \cdot \log \frac{n}{\beta}$
  $\lambda \leftarrow \frac{n}{n - \log \frac{n}{\beta}} \cdot \lambda$
end for
$w_1^{recon}, \dots, w_n^{recon} \leftarrow$ (denormalize $u_1^{recon}, \dots, u_n^{recon}$ and add sign bits.)

$\mathsf{ParamEst}((u_1, \dots, u_n))$:
  $1/\lambda \leftarrow$ mean of $(u_1, \dots, u_n)$
  return λ

with the following optimal conditional probability distribution that achieves the minimum mutual information

$$f_{\mathbf{U}|\mathbf{V}}(u|v) = \begin{cases} \frac{1}{D} e^{-(u-v)/D}, & u \geq v \geq 0 \\ 0, & \textit{otherwise.} \end{cases} \tag{30}$$

Moreover, the marginal distribution of $\mathbf{V}$ *is as follows*

$$f_{\mathbf{V}}(v) = \lambda D \cdot \delta(v) + (1 - \lambda D) \cdot \lambda e^{-\lambda v} \tag{31}$$

where $\delta(v)$ *is a Dirac measure at 0.*

Proof of Lemma 3 can be found in (Verdu, 1996). It is clear to see from Eq.s 30 and 31 that exponential source has the same nice properties as Laplacian:

1. It suggests pruning as an essential step in a good compression algorithm since Eq. 31 is a sparse distribution.
2. It is successively refinable, allowing for a both practical and rate-distortion theoretic optimal algorithm (see Algorithm 2).

Following the same steps in Section 5, it can be proven that Algorithm 2, which we used in our experiments, is zero-rate optimal with $\beta = \log n$.

E.1 Effect of β

Table 3 shows the effect of the hyperparameter β on the model accuracy, the number of SuRP iterations needed to achieve the desired sparsity, and the number of required refreshment for the Laplacian parameter λ. We perform one-shot pruning experiments without retraining, i.e., apply SuRP once, with different β values as shown in the table. We can consider the accuracy as a measure of distortion; and 'the number of iterations' and 'the number of parameter refreshments' as a measure of rate. More concretely, we first fix a target sparsity for all the experiments. If the number of iterations to achieve this sparsity is large, then the compression amount is small since higher number of indices represents the same model. During one running of SuRP, there might be a need for re-estimating the parameter λ of the underlying Laplacian distribution, as we stated in Section 5.2. This is an undesirable situation since this requires the encoder to send the re-estimated parameter λ to the decoder; we call this a "refreshment". The numbers in this table verify our theoretical analysis that as β increases, the number of refreshments becomes negligible compared to the total number of iterations (see Eq. 8). However, for very large β such as $\beta = (\log n)^2$, zero-rate optimality is not as strong as $\beta = \log n$ (see Theorem 2). This can also be seen from the table since the number of iterations needed is significantly larger for $\beta = (\log n)^2$, indicating that the bit-rate is large and we do not compress the model much. Since the accuracy is similar across different β values, we can conclude that $c \cdot \log n$ is indeed a reasonable choice for β as it balances the two factors (number of iterations and number of refreshments) that contribute to the rate.

Table 3: Effect of the choice of β on the model accuracy, the number of SuRP iterations needed to achieve the desired sparsity, and the number of required refreshment for the Laplacian parameter λ. The experiments are run with VGG-16 model on CIFAR-10 dataset. The sparsity is 95% in all cases. Note that these experiments do not involve multiple pruning steps or retraining, that is why the accuracy is slightly smaller than the numbers in Table 1.

β :	$\sqrt{\log n}$	$1/2 \cdot \log n$	$\log n$	$2 \cdot \log n$	$(\log n)^2$
Accuracy	89.2%	90.02%	90.02%	90.00%	90.00%
Required Number of Iterations	$1.1M$	$1.2M$	$1.2M$	$1.3M$	$20M$
Number of Parameter Refreshment	$20K$	500	22	20	10

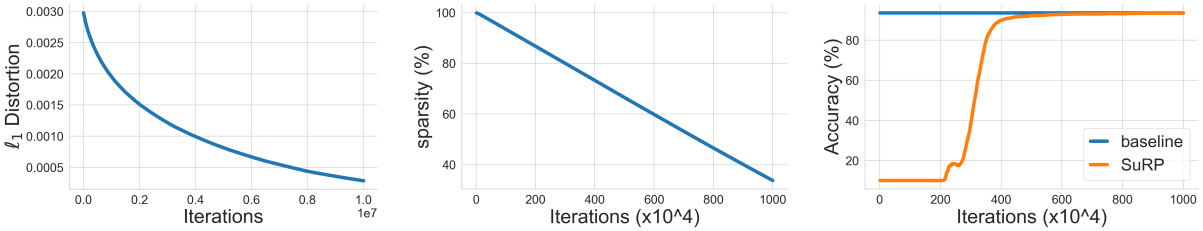
F Visualization of SuRP

Figure 5 shows the decreasing ℓ_1 distortion and sparsity, and increasing accuracy of the reconstructed model through the iterations (running inside SuRP). We note that SuRP is applied only once in Figure 5 and iterations correspond to the iterations running inside SuRP. However, as stated in Section 6, we adopted iterative pruning approach, where each pruning iteration corresponds to running SuRP one time. After each pruning iteration, SuRP outputs a sparse model, and a retraining procedure is applied to the sparse model. When we do iterative pruning, we apply SuRP several times by increasing the target sparsity ratio every time. For instance, let us say we want to prune 90% of the parameters in the first pruning iteration. Then, as shown in Figure 6, SuRP stops once the sparsity ratio drops to 90%. Before starting the next iteration (next round of SuRP), we retrain the sparse model by excluding the pruned parameters (as proposed in (Han et al., 2016)). In the next iteration, as shown in Figure 7, the sparsity can never be lower than 90% no matter how long we run the algorithm since 90% of the parameters are already pruned in the previous iteration of the pruning. As we typically desire a higher sparsity ratio in the later iterations, we need to stop SuRP at the target sparsity (which is higher than 90%).

G Proof of Theorem 2

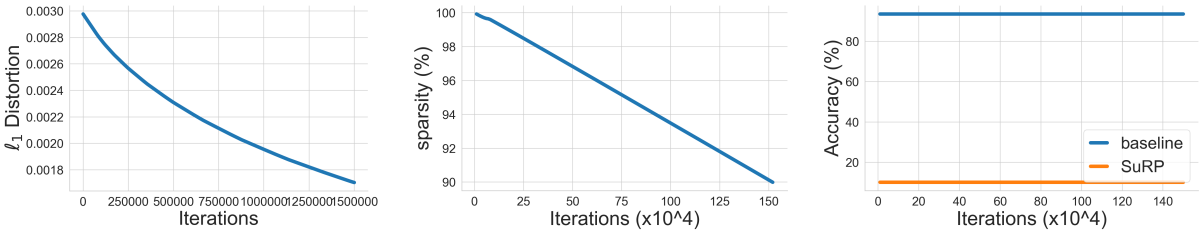
In this section, we provide the proof of Theorem 2. In an iteration of SuRP, where $R_n = \frac{\log n(n-1)}{n}$ and $D_n = \frac{2}{n\lambda} \log \frac{n}{2\beta_n}$, we have

$$\begin{aligned} \frac{D_n}{R_n} &= -\frac{\frac{2}{\lambda} \log \frac{n}{2\beta_n}}{\log n(n-1)} \\ &= -\frac{1}{\lambda} \frac{\log n^2}{\log n(n-1)} + \frac{1}{\lambda} \frac{2 \log 2\beta_n}{\log n(n-1)}. \end{aligned}$$



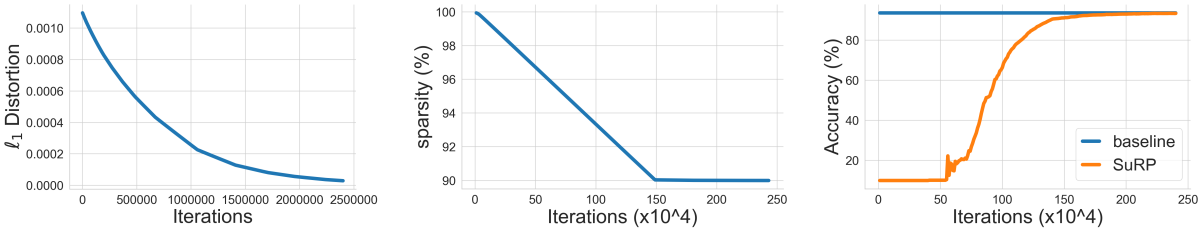
(a) ℓ_1 Distortion. (b) Sparsity. (c) Accuracy.

Figure 5: (a) Average ℓ_1 distortion, (b) sparsity and (c) accuracy of the reconstructed VGG-16 when SuRP is applied once (no iterative pruning). Baseline: fully-trained model without compression.



(a) ℓ_1 Distortion. (b) Sparsity. (c) Accuracy.

Figure 6: (a) Average ℓ_1 distortion, (b) sparsity and (c) accuracy of the reconstructed VGG-16 during SuRP (first iteration of the iterative pruning). SuRP stops at the desired sparsity 90%. Baseline: fully-trained model without compression.



(a) ℓ_1 Distortion. (b) Sparsity. (c) Accuracy.

Figure 7: (a) Average ℓ_1 distortion, (b) sparsity and (c) accuracy of the reconstructed VGG-16 during SuRP (after the first iteration of the iterative pruning). The sparsity cannot be lower than 90% and SuRP must stop at the desired sparsity (which is higher than 90%). Baseline: fully-trained model without compression.

If $\lim_{n\to\infty} \frac{\log 2\beta_n}{\log n(n-1)} = 0$, it is clear that $\frac{D_n}{R_n}$ converges to $D'(0) = -\frac{1}{\lambda}$ as n increases. Therefore, SuRP is zero-rate optimal under the condition that $\lim_{n\to\infty} \frac{\log 2\beta_n}{\log n(n-1)} = 0$.

H Optimizing the Bit Rate

In this section, we highlight a useful byproduct of SuRP as a way to minimize the bit rate of the pruned model. Recall that SuRP requires transmitting two indices $i, j \in \{1, \dots, n\}$ from the encoder to the decoder for each iteration. This means that SuRP automatically gives the integer (indices are integers from $1, \dots, n$) representation of the model. Therefore, without dealing with floating points, i.e., precise values of the weights, we can reconstruct the model back using these indices. In order to further optimize this, we need a lossless compression scheme, namely entropy coding, to represent these indices as binary sequences. In information theory, the optimal entropy

coding method can be found when the source distribution is known in advance (Huffman, 1952). Although there are universal codes that encode any source regardless of the distribution, they are preferable only when the source distribution is unknown since an entropy coding that matches the source distribution is always better than a universal code. Fortunately, our coding scheme for Laplacian (also for exponential) source induces a well-defined distribution that allows us to choose an optimal entropy coding method. Notice that randomly picking two indices i, j from $\{k : \mathbf{U}_k^{(t)} \geq \frac{1}{\lambda_t} \cdot \log \frac{n}{2\beta}\}$ and $\{k : \mathbf{U}_k^{(t)} \leq -\frac{1}{\lambda_t} \cdot \log \frac{n}{2\beta}\}$ is equivalent to; (1) first randomly permuting $\mathbf{U}^{(t)}$, and (2) selecting the minimum indices i, j from $\{k : \mathbf{U}_k^{(t)} \geq \frac{1}{\lambda_t} \cdot \log \frac{n}{2\beta}\}$ and $\{k : \mathbf{U}_k^{(t)} \leq -\frac{1}{\lambda_t} \cdot \log \frac{n}{2\beta}\}$. The second approach induces a geometric distribution under the i.i.d. assumption on the indices where small indices are always more probable to be selected. For geometric sources, there are two standard entropy coding methods: unary coding and Golomb coding (Golomb, 1966; Gallager and Van Voorhis, 1975). In our additional experiments in Appendix K, for comparing SuRP with other works on accuracy-bit rate tradeoff, we use Golomb coding. Now, we give more details on both methods.

Unary Coding. Unary coding is a prefix-free code that is optimally efficient for the following geometric distribution:

$$P_B(b) = 2^{-b} \tag{32}$$

where b is a positive integer. In the simplest term, unary coding encodes an integer b with single 1 followed by $b-1$ consecutive 0's. For instance, 72 would be uniquely encoded as 100000010. In our problem, indices follow the distribution in Eq. 32 only when the fraction of normalized weights larger than $1/\lambda_t \cdot \log \frac{n}{2\beta}$ in magnitude is exactly equal to $1/2$. Since this is not the case in every iteration, unary coding is not the optimal entropy coding method for indices in SuRP.

Golomb Coding. Golomb coding is an optimal prefix-free code for any geometric source, i.e., it is more general than unary coding. The construction of Golomb codes can be found in (Golomb, 1966). In our additional experiments in Appendix K, we implemented Golomb coding to represent NN models as binary arrays.

I Compression for Federated Learning

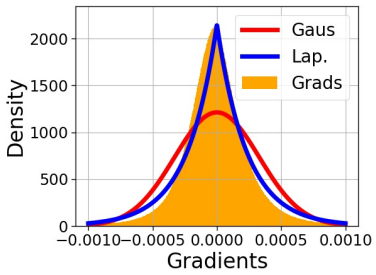
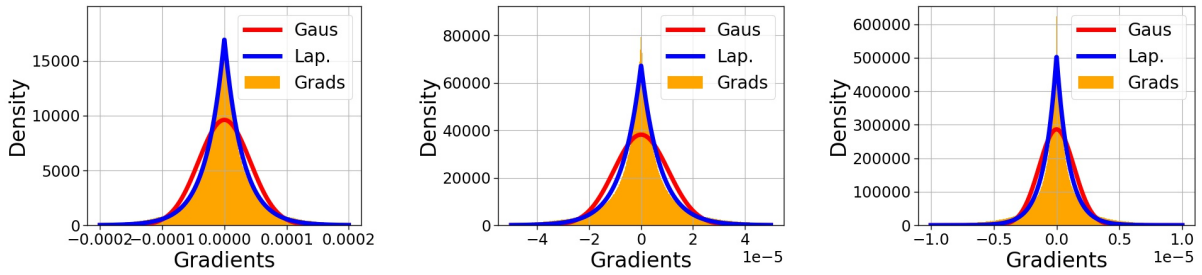


Figure 8: Density of gradients of ResNet-50 trained on ImageNet. We present only the gradients from the late stages of training since we use a pretrained ResNet-50.

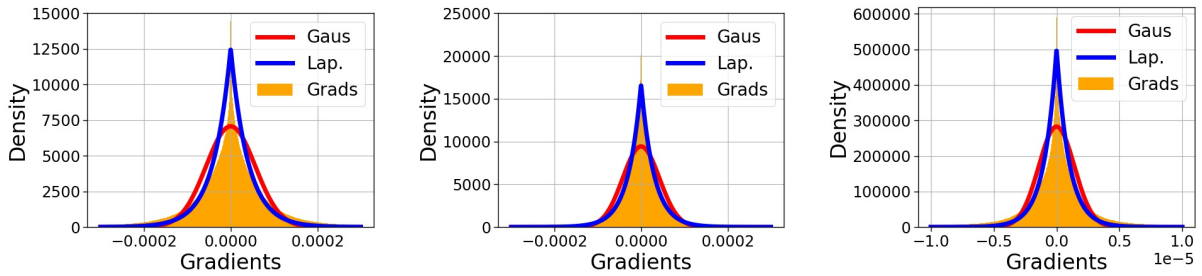
In Section 6, we have applied SuRP to compress gradients in federated learning. In Figures 8, 9, and 10, we justify that Laplacian is a good fit for gradients of ResNet-50 trained on ImageNet, ResNet-18 trained on CIFAR-10, and VGG-16 trained on CIFAR-10. Since we need to compress the gradients before each communication round of federated training, SuRP requires the gradients to follow a Laplacian distribution throughout the learning process. In other words, although the parameter of the Laplacian distribution might change, we must be able to fit a Laplacian distribution to the gradients in every round. We provide the density estimation of gradients in early-, mid-, and late-stages of training in Figures 8, 9, and 10 to show that Laplacian distribution is a good fit for gradients starting from the early stages of training till the training ends. Therefore, we can apply SuRP to compress gradients at every communication round. In our experiments, we update the parameter of the Laplacian distribution (λ) at every communication round.

Among other gradient sparsification methods for federated learning (Aji and Heafield, 2017; Lin et al., 2017; McMahan et al., 2017; Wang et al., 2018; Wangni et al., 2018), SuRP is most similar to rTop-k (Barnes et al., 2020),



(a) Gradients at Early Stage. (b) Gradients at Mid Stage. (c) Gradients at Late Stage.

Figure 9: Density of gradients of ResNet-18 trained on CIFAR-10 during (a) early stages of training (epoch 32), (b) middle stages of training (epoch 155), (c) late stages of training (epoch 336).



(a) Gradients at Early Stage. (b) Gradients at Mid Stage. (c) Gradients at Late Stage.

Figure 10: Density of gradients of small VGG-16 trained on CIFAR-10 during (a) early stages of training (epoch 16), (b) middle stages of training (epoch 100), (c) late stages of training (epoch 170).

in that they also communicate a random subset of the large gradients. Different from our work, they approach the communication-efficient federated learning problem from a distributed statistical estimation point of view. By modeling the gradients with a sparse Bernoulli distribution, they show that the optimal compression strategy for each user (device) is to communicate a random k/r fraction of the r largest gradients. In contrast, we study the gradient compression problem from an information-theoretic approach and assume Laplacian distribution over the gradients. With this assumption, we conclude that each user must communicate the parameter of underlying Laplacian distribution of the local gradients and a list of indices that are randomly selected among the gradients larger than $\frac{1}{\lambda_t} \cdot \log \frac{n}{2\beta}$ in magnitude at iteration t. Since the threshold $\frac{1}{\lambda_t} \cdot \log \frac{n}{2\beta}$ is decreasing at each iteration, SuRP assigns a higher probability for larger (in magnitude) gradients to be selected, whereas rTop-k picks the gradients uniformly random from the large gradients.

J Additional Experimental Details

We conducted our experiments on NVIDIA Titan X (MNIST and CIFAR-10) and NVIDIA Titan Xp (ImageNet) GPUs on an internal cluster server. We used 1 GPU for MNIST and CIFAR-10 experiments and 2 GPUs for ImageNet experiments. We set the target sparsity of each SuRP round so that at each pruning iteration, 20% of the surviving parameters will be pruned, e.g., sparsity schedule is as follows $20\%, 36\%, 48.8\%, 59.04\%, \ldots$.

J.1 MNIST:

We provide the architectural details and hyperparameters for LeNet-5 Caffe in Table 4 (LeCun et al., 1998). We use a batch size of 100 and train for 100 epochs, early stopping at the best accuracy on validation set. We use the Adam optimizer with learning rate $= 0.001$, and $\beta_1 = 0.9, \beta_2 = 0.999$ with weight decay $= 5e^{-4}$.

Table 4: LeNet-5 Caffe convolutional architecture.

Name	Component
conv1	[5×5 conv, 20 filters, stride 1], ReLU, 2×2 max pool
conv2	[5×5 conv, 50 filters, stride 1], ReLU, 2×2 max pool
Linear	Linear $800 \rightarrow 500$, ReLU
Output Layer	Linear $500 \rightarrow 10$

J.2 CIFAR-10:

We provide the architectural details and hyperparameters for the ResNet-20 (He et al., 2016) and (small) VGG-16 (Simonyan and Zisserman, 2014) in Tables 5 and 6, respectively. For both ResNet-20 and VGG-16, we use a batch size of 128, we train ResNet-20 for 350 epochs and VGG-16 for 200 epochs, early stopping at the best accuracy on validation set. We use SGD with learning rate $= 0.1$, momentum $= 0.9$, and weight decay $= 5e^{-4}$. We note that VGG-16 architecture is a smaller version of the original VGG architecture in (Simonyan and Zisserman, 2014). We retrain both models for 20 epochs at the end of each pruning iteration.

Table 5: Slim ResNet-20 architecture.

Name	Component
conv1	3×3 conv, 16 filters. stride 1, BatchNorm
Residual Block 1	$\begin{bmatrix} 3 \times 3 \text{ conv, } 16 \text{ filters} \\ 3 \times 3 \text{ conv, } 16 \text{ filters} \end{bmatrix} \times 2$
Residual Block 2	$\begin{bmatrix} 3 \times 3 \text{ conv, } 32 \text{ filters} \\ 3 \times 3 \text{ conv, } 32 \text{ filters} \end{bmatrix} \times 2$
Residual Block 3	$\begin{bmatrix} 3 \times 3 \text{ conv, } 64 \text{ filters} \\ 3 \times 3 \text{ conv, } 64 \text{ filters} \end{bmatrix} \times 2$
Output Layer	7×7 average pool stride 1, fully-connected, softmax

Table 6: VGG-16 architecture.

Name	Component
conv1-2	[3×3 conv, 64 filters. stride 1, BatchNorm, ReLU] $\times 2$
max pool	2×2, stride 2
conv3-4	[3×3 conv, 128 filters. stride 1, BatchNorm, ReLU] $\times 2$
max pool	2×2, stride 2
conv5-7	[3×3 conv, 256 filters. stride 1, BatchNorm, ReLU] $\times 3$
max pool	2×2, stride 2
conv8-10	[3×3 conv, 512 filters. stride 1, BatchNorm, ReLU] $\times 3$
max pool	2×2, stride 2
conv11-13	[3×3 conv, 512 filters. stride 1, BatchNorm, ReLU] $\times 3$
max pool	2×2, stride 2
Output Layer	1×1 average pool stride 1, fully-connected, softmax

J.3 ImageNet:

We provide the architectural details and hyperparameters for the ResNet-50 used in our experiments in Table 7. We use the pretrained ResNet-50 from PyTorch (`https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py`), with a batch size of 64. At the end of each pruning iteration, we retrain the model for 15 epochs. We use SGD with learning rate $= 0.001$, momentum $= 0.9$ and weight decay $= 5e^{-4}$.

Table 7: ResNet-50 architecture.

Name	Component
conv1	3×3 conv, 64 filters. stride 1, BatchNorm
Residual Block 1	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \text{ filters} \\ 3 \times 3 \text{ conv, } 64 \text{ filters} \\ 1 \times 1 \text{ conv, } 256 \text{ filters} \end{bmatrix} \times 3$
Residual Block 2	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \text{ filters} \\ 3 \times 3 \text{ conv, } 128 \text{ filters} \\ 1 \times 1 \text{ conv, } 512 \text{ filters} \end{bmatrix} \times 4$
Residual Block 3	$\begin{bmatrix} 1 \times 1 \text{ conv, } 256 \text{ filters} \\ 3 \times 3 \text{ conv, } 256 \text{ filters} \\ 1 \times 1 \text{ conv, } 1024 \text{ filters} \end{bmatrix} \times 6$
Residual Block 4	$\begin{bmatrix} 1 \times 1 \text{ conv, } 512 \text{ filters} \\ 3 \times 3 \text{ conv, } 512 \text{ filters} \\ 1 \times 1 \text{ conv, } 2048 \text{ filters} \end{bmatrix} \times 3$
Output Layer	4×4 average pool stride 1, fully-connected, softmax

K Additional Experimental Results

We give a more detailed version of Table 1 in Tables 8, 9, 10 and 11 and a more detailed version of Table 2 in Table 12 with confidence intervals included in SuRP results.

Table 8: Accuracy of VGG-16 on CIFAR-10. Results are averaged over five runs.

Pruning Ratio:	93.12%	95.60%	97.19%	98.20%	98.85%	99.26%	99.53%	99.70%	99.81%	99.88%
Global (Morcos et al., 2019)	91.30	90.80	89.28	85.55	81.56	54.58	41.91	31.93	21.87	11.72
Uniform (Zhu and Gupta, 2017)	91.47	90.78	88.61	84.17	55.68	38.51	26.41	16.75	11.58	9.95
Adaptive (Gale et al., 2019)	91.54	91.20	90.16	89.44	87.85	86.53	84.84	82.41	74.54	24.46
RiGL (Evci et al., 2020)	92.34	91.99	91.66	91.15	90.55	89.51	88.21	86.73	84.85	81.50
LAMP (Lee et al., 2021)	92.24	92.06	91.71	91.66	91.07	90.49	89.64	88.75	87.07	84.90
SuRP (ours)	**92.55** $\pm$ 0.19	**92.13** $\pm$ 0.20	**91.95** $\pm$ 0.21	**91.72** $\pm$ 0.28	**91.21** $\pm$ 0.24	**90.73** $\pm$ 0.21	**90.65** $\pm$ 0.27	**89.70** $\pm$ 0.32	**87.28** $\pm$ 0.32	**85.04** $\pm$ 0.35

Table 9: Accuracy of ResNet-20 on CIFAR-10. Results are averaged over five runs.

Pruning Ratio:	79.03%	86.58%	91.41%	94.50%	96.48%	97.75%	98.56%	99.08%	99.41%	99.62%
Global (Morcos et al., 2019)	87.48	86.97	86.29	85.02	83.15	80.52	76.28	70.69	47.47	12.02
Uniform (Zhu and Gupta, 2017)	87.24	86.70	86.09	84.53	82.05	77.19	64.24	47.97	20.45	13.35
Adaptive (Gale et al., 2019)	87.30	87.00	86.27	85.00	83.23	80.40	76.40	69.31	52.06	20.19
RiGL (Evci et al., 2020)	87.63	87.49	86.83	85.84	84.08	81.76	78.70	74.40	66.42	50.90
LAMP (Lee et al., 2021)	87.54	87.12	86.56	85.64	84.18	81.56	78.63	74.20	67.01	51.24
SuRP (ours)	**91.37** $\pm$ 0.24	**90.44** $\pm$ 0.26	**89.00** $\pm$ 0.21	**88.87** $\pm$ 0.26	**87.05** $\pm$ 0.28	**83.98** $\pm$ 0.20	**79.00** $\pm$ 0.34	**74.86** $\pm$ 0.29	**70.64** $\pm$ 0.38	**54.22** $\pm$ 0.42

Table 10: DenseNet-121 on CIFAR-10. Results are averaged over five runs.

Pruning Ratio:	94.50%	95.60%	96.48%	97.18%	97.75%	98.20%	98.56%	98.85%	99.08%	99.26%
Global (Morcos et al., 2019)	90.16	89.52	88.83	88.00	86.85	85.32	77.68	45.30	49.65	20.96
Unif. (Zhu and Gupta, 2017)	90.24	89.50	88.44	87.94	86.83	85.00	82.16	70.13	66.46	48.71
Adap. (Gale et al., 2019)	90.25	89.70	89.03	88.22	87.40	86.26	84.55	81.87	69.25	58.91
RiGL (Evci et al., 2020)	90.21	89.79	88.92	88.20	87.25	86.22	84.11	81.82	59.06	59.07
LAMP (Lee et al., 2021)	90.89	90.11	89.72	89.12	88.39	87.75	86.53	85.13	82.92	79.23
SuRP (ours)	**91.42** $\pm$ 0.11	**90.75** $\pm$ 0.08	**90.30** $\pm$ 0.20	**89.62** $\pm$ 0.17	**88.77** $\pm$ 0.08	**88.06** $\pm$ 0.22	**86.71** $\pm$ 0.15	**85.34** $\pm$ 0.27	**83.18** $\pm$ 0.24	**79.45** $\pm$ 0.36

We also provide a comparison between SuRP and LAMP at lower pruning rates in Table 13.

Additionally, we provide accuracy-bit rate comparisons between SuRP and relevant baselines such as Deep Comp. (Han et al., 2016), DeepCABAC (Wiedemann et al., 2020), DNS (Guo et al., 2016), and SWS (Ullrich et al., 2017) in Table 14. It is seen from Table 14 that SuRP outperforms the baselines both in terms of accuracy-sparsity and accuracy-bit rate tradeoffs.

Table 11: EfficientNet-B0 on CIFAR-10. Results are averaged over five runs.

Pruning Ratio:	59.00%	73.80%	83.20%	89.30%	93.13%	95.60%	97.18%	98.20%	98.85%	99.26%
Global (Morcos et al., 2019)	89.66	89.55	88.80	87.64	84.36	79.25	11.09	10.62	10.00	10.00
Uniform (Zhu and Gupta, 2017)	88.99	88.26	86.48	83.40	23.65	10.83	10.00	10.00	10.00	10.00
Adaptive (Gale et al., 2019)	89.18	88.03	86.71	84.16	36.64	10.45	10.00	10.19	10.00	10.00
RiGL (Evci et al., 2020)	89.54	90.09	90.01	89.62	88.82	87.08	84.72	81.53	51.31	13.40
LAMP (Lee et al., 2021)	89.52	89.95	89.97	90.21	89.91	89.79	89.30	88.51	86.79	65.76
SuRP (ours)	**90.96** $\pm$ 0.10	**90.94** $\pm$ 0.12	**90.89** $\pm$ 0.12	**90.75** $\pm$ 0.16	**90.31** $\pm$ 0.21	**90.08** $\pm$ 0.20	**89.88** $\pm$ 0.27	**89.02** $\pm$ 0.38	**87.80** $\pm$ 0.0.36	**70.76** $\pm$ 0.52

Table 12: Accuracy of ResNet-50 on ImageNet. Results are averaged over three runs.

Pruning Ratio:	80%	90%
Adaptive (Gale et al., 2019)	75.60	73.90
SNIP (Lee et al., 2018)	72.00	67.20
DSR (Mostafa and Wang, 2019)	73.30	71.60
SNFS (Dettmers and Zettlemoyer, 2019)	74.90	72.90
RiGL (Evci et al., 2020)	74.60	72.00
SuRP (ours)	**75.54** $\pm$ 0.05	**73.93** $\pm$ 0.04

Table 13: Additional Results with Low Pruning Ratios.

Pruning Ratio:		20%	36%	49%	59%	67%	79%
VGG-16	LAMP (Lee et al., 2021)	93.12	93.08	93.05	92.89	92.81	92.75
	SuRP (ours)	**93.72**	**93.75**	**93.72**	**93.63**	**93.64**	**93.56**
ResNet-20	LAMP (Lee et al., 2021)	89.12	88.81	88.67	88.27	87.95	87.54
	SuRP (ours)	**92.47**	**92.43**	**92.29**	**92.30**	**91.98**	**91.37**

Table 14: Comparison of SuRP with other pruning strategies in terms of accuracy, sparsity and size (bit rate).

Model (Original size)	Original Acc. (%)	Method	Sparsity $\frac{\lvert w=0\rvert}{\lvert w\rvert}$ (%)	Comp. Size	Comp. Acc. (%)
LeNet-5-Caffe MNIST (1.72 MB)	99.14	Deep Comp. (Han et al., 2016)	92.0	44 KB (×39)	99.3
		DNS (Guo et al., 2016)	99.1	16 KB (×107)	99.1
		SWS (Ullrich et al., 2017)	99.5	11 KB (×156)	99.0
		DeepCABAC (Wiedemann et al., 2020)	98.1	12 KB (×143)	99.1
		SuRP (ours)	99.2	**7 KB (**×246**)**	99.3 (± 0.0)
		SuRP (ours)	99.3	**5 KB (**×344**)**	98.2 (± 0.1)
ResNet-18 CIFAR-10 (44.70 MB)	95.60	SuRP (ours)	90.0	3.1 MB (×15)	95.1 (± 0.0)
		SuRP (ours)	95.0	1.1 MB (×42)	92.2 (± 0.1)
		SuRP (ours)	97.0	**875 KB (**×53**)**	90.0 (± 0.2)
Small VGG-16 CIFAR-10 (58.91 MB)	93.60	DeepCABAC (Wiedemann et al., 2020)	92.4	956 KB (×61)	91.0
		SuRP (ours)	95.0	**1.1 MB (**×54**)**	92.4 (± 0.1)
		SuRP (ours)	90.0	3.0 MB (×20)	93.5 (± 0.1)
ResNet-50 ImageNet (102.23 MB)	76.60	Deep Comp. (Han et al., 2016)	71.0	6.1 MB (×16)	76.1
		DeepCABAC (Wiedemann et al., 2020)	74.6	6.1 MB (×16)	74.1
		SuRP (ours)	71.0	**6.1 MB (**×16**)**	76.4 (± 0.0)