

# SWAP: SPARSE ENTROPIC WASSERSTEIN REGRESSION FOR ROBUST NETWORK PRUNING

**Lei You\***

Department of Engineering Technology  
 Technical University of Denmark  
 Ballerup, DK-2750, Denmark  
 leiyo@dtu.dk

**Hei Victor Cheng\***

Department of Electrical and Computer Engineering  
 Aarhus University  
 Aarhus, DK-8200, Denmark  
 hvc@ece.au.dk

## ABSTRACT

This study addresses the challenge of inaccurate gradients in computing the empirical Fisher Information Matrix during neural network pruning. We introduce SWAP, a formulation of Entropic Wasserstein regression (EWR) for pruning, capitalizing on the geometric properties of the optimal transport problem. The “swap” of the commonly used linear regression with the EWR in optimization is analytically demonstrated to offer noise mitigation effects by incorporating neighborhood interpolation across data points with only marginal additional computational cost. The unique strength of SWAP is its intrinsic ability to balance noise reduction and covariance information preservation effectively. Extensive experiments performed on various networks and datasets show comparable performance of SWAP with state-of-the-art (SoTA) network pruning algorithms. Our proposed method outperforms the SoTA when the network size or the target sparsity is large, the gain is even larger with the existence of noisy gradients, possibly from noisy data, analog memory, or adversarial attacks. Notably, our proposed method achieves a gain of 6% improvement in accuracy and 8% improvement in testing loss for MobileNetV1 with less than one-fourth of the network parameters remaining.

## 1 INTRODUCTION

The advent of deep learning has revolutionized various domains of artificial intelligence, with neural networks showing remarkable performance across an array of applications. Nonetheless, the increase in model complexity has led to escalating computational demands and substantial memory requirements. This poses significant challenges for deploying these models in resource-constrained environments such as mobile or internet of things (IoT) devices. Therefore, the concept of neural network pruning emerges as a critical solution. It aims to optimize the network by removing less important parameters, which reduces computational overhead while maintaining the performance of the original model.

In the realm of state-of-the-art deep learning, the models often exhibit substantial size and complexity, with up to trillions of parameters, as exemplified by models such as GPT-4. The immense computational demand, energy inefficiency, and the challenges with model interpretability associated with these models highlight the need for innovative and efficient optimization techniques. These techniques should ideally minimize the model size while improving their robustness and interpretability. Considering the limitations of previous work, especially those arising from the influence of noisy data and noisy gradients, the paper proposes a promising pathway for robust pruning.

Below, we inspect the network pruning problem from an optimization perspective, with a concise introduction of the most relevant existing works. Then a sketch of our approach is given.

**Related Work on Pruning as Optimization.** Denote by  $\bar{\mathbf{w}} \in \mathbb{R}^p$  a trained model and  $\mathcal{L}(\mathbf{w})$  the loss function given arbitrary model  $\mathbf{w}$ . The loss function can be locally approximated around  $\bar{\mathbf{w}}$  with

\*Correspondence to both leiyo@dtu.dk and hvc@ece.au.dk. Lei You is supported by Thomas B. Thriges Fond 5041-2402. Hei Victor Cheng is supported by the Aarhus Universitets Forskningsfond under Project AUFF 39001.

Taylor Expansion as shown in (1).

$$\mathcal{L}(\mathbf{w}) = \mathcal{L}(\bar{\mathbf{w}}) + \nabla\mathcal{L}(\bar{\mathbf{w}})^\top(\mathbf{w} - \bar{\mathbf{w}}) + \frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^\top \nabla^2\mathcal{L}(\bar{\mathbf{w}})(\mathbf{w} - \bar{\mathbf{w}}) + O(\|\mathbf{w} - \bar{\mathbf{w}}\|^3) \quad (1)$$

Consider a neural network with a loss function  $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \ell_i(\mathbf{w})$ , where  $\ell_i(\mathbf{w}) \in \mathbb{R}^p$  is the loss incurred at data point  $i$  ( $i = 1, \dots, N$ ). The goal of network pruning is to find a set of  $\mathbf{w}$  such that there are  $k$  ( $k < p$ ) elements of  $\mathbf{w}$  being zero while keeping the newly obtained model  $\mathbf{w}$ 's performance as good as possible to the original one  $\bar{\mathbf{w}}$ . Mathematically, we want to find some  $\mathbf{w} \in \mathbb{R}^p$  that satisfies both  $\mathcal{L}(\mathbf{w}) \approx \mathcal{L}(\bar{\mathbf{w}})$  and  $\|\mathbf{w}\|_0 \leq k$ , with  $k < p$ .

This line of research can be dated back to (LeCun et al., 1989), where the approximation in equation (1) is adopted. Under the assumption that gradient  $\nabla\mathcal{L}(\bar{\mathbf{w}}) \approx 0$  when the network is trained, the network weights are pruned one-by-one in decreasing order based on the value of  $(\mathbf{w} - \bar{\mathbf{w}})^\top \mathbf{H}(\mathbf{w} - \bar{\mathbf{w}})$ . In their approach, the  $\mathbf{H}$  is approximated as a diagonal matrix; this is later extended in (Hassibi & Stork, 1992) to include the whole Hessian matrix, and the authors also proposed using the Fisher information matrix (FIM) as an approximation to the Hessian. Later, (Singh & Alistarh, 2020) proposed to reduce the computation complexity by using block diagonal Hessian, and FIM is approximated using a small subset of the training data.

These approaches all use equation (1) to prune the network in a one-by-one manner, namely the weight with the least importance is set to zero according to the different approximations of equation (1). In this way, the potential interactions of pruning multiple weights are ignored. To explore this, the network pruning problem is formulated as a mixed integer quadratic programming (MIQP) in (Yu et al., 2022). Namely, an objective function

$$f(\mathbf{w}) = (\mathbf{w} - \bar{\mathbf{w}})^\top \mathbf{H}(\mathbf{w} - \bar{\mathbf{w}}) + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 \quad (\lambda \geq 0) \quad (2)$$

is minimized, and Hessian is approximated as  $\mathbf{H} \approx \nabla^2\mathcal{L}(\bar{\mathbf{w}})$ , subject to the sparsity constraint  $\|\mathbf{w}\|_0 \leq k$ , where  $\lambda$  is a regularization parameter. Although this approach shows significant improvements, it suffers from scalability issues as a full Hessian matrix is required.

**Sparse Linear Regression (LR) Formulation.** To reduce the computational complexity, the Hessian matrix can be approximated by the empirical FIM, using  $n$  samples as in (Chen et al., 2022; Benbaki et al., 2023). Denote  $\mathbf{G} = [\nabla\ell_1, \dots, \nabla\ell_n]^\top \in \mathbb{R}^{n \times p}$ , where  $\nabla\ell_i = \nabla\ell_i(\bar{\mathbf{w}})$ . For simplicity,  $\nabla\ell_i$  is used in this document to represent the derivative of the data point  $i$ 's loss at  $\bar{\mathbf{w}}$  consistently in this paper unless specified otherwise. The Hessian is approximated through the expression  $\mathbf{H} \approx (1/n) \sum_{i=1}^n \nabla\ell_i \nabla\ell_i^\top = (1/n) \mathbf{G}^\top \mathbf{G}$ , which is the so-called FIM. Denote  $x_i = \nabla\ell_i^\top \mathbf{w}$  and  $y_i = \nabla\ell_i^\top \bar{\mathbf{w}}$ , (2) is formulated to the sparse LR problem shown in (3) below.

$$\min_{\mathbf{w}} \bar{Q}(\mathbf{w}) = \sum_{i=1}^n \|x_i(\mathbf{w}) - y_i\|^2 + n\lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2, \text{ s.t. } \|\mathbf{w}\|_0 \leq k \quad (3)$$

This formulation has a computational advantage, as empirical FIM needs not to be computed explicitly. It is shown that the formulation scales to *large neural network pruning* (Chen et al., 2022).

**Motivation of Combating Against Noise.** In practice, it is not always easy to obtain the correct gradients for pruning large neural networks. There can be noise contained in the data samples, and the gradients can also be corrupted due to various reasons, e.g., distributed or federated learning (Turan et al., 2022), or adversarial attacks such as data poisoning (Steinhardt et al., 2017).

As pointed out by (Mahsereci et al., 2017; Siems et al., 2021), conditioning on the underlying true gradient  $\nabla\mathcal{L}(\bar{\mathbf{w}}) = 0$ , there are mini-batch gradients which are not informative anymore as it can be fully explained by sample noise and the vanishing gradients. These gradients would not contribute to the covariance information of the empirical FIM but serve as outliers in Hessian approximation.

In the scenarios of federated learning (FL), gradients computed by different clients are skewed and consequently, local models move away from globally optimal models (Huang et al., 2022), imposing challenges for constructing informative FIM. Besides, noise can be added to the gradient for privacy concerns in communications (Li et al., 2020). Additionally, the clients usually have inevitable noisy samples and labels, making models suffer from a significant performance drop (Tuor et al., 2021). Additionally, over-the-air communications itself suffer from unavoidable noises (Ang et al., 2020; Yang et al., 2020). These lead to concerns for network pruning with noisy gradients. Finally, analog

memory recently gained attention for deep learning model deployment (Garg et al., 2022). When neural network parameters and data are stored in these analog devices, they are susceptible to device-related noise, affecting the performance of network compression (Isik et al., 2023).

**Approach Sketch.** We revisit the MIQP network pruning optimization from a perspective of entropic Wasserstein regression (EWR), which leverages Wasserstein distance to model the dissimilarity between two distributions. In our context, it measures the dissimilarity of distributions relevant to model parameters and gradient magnitudes before and after pruning. Namely,  $\nabla\ell$  is a  $p$  dimensional distribution, capturing geometric properties of the loss at  $\bar{\mathbf{w}}$  before pruning. Both  $\bar{\mathbf{w}}$  and  $\mathbf{w}$  perform projections for  $\nabla\ell$  to a 1-D distribution respectively as  $\nabla\ell^\top \bar{\mathbf{w}}$  and  $\nabla\ell^\top \mathbf{w}$ . Computing the distance between  $\nabla\ell^\top \bar{\mathbf{w}}$  and  $\nabla\ell^\top \mathbf{w}$  falls into the framework of sliced probability divergence (Nadjahi et al., 2020). Under this framework, pruning optimization essentially fine-tunes  $\mathbf{w}$  and selectively reserves its elements such that the divergence is minimized subject to the sparsity constraint.

Our approach’s effectiveness in combating noisy gradients is established both analytically and numerically. We demonstrate that *pruning through the Wasserstein regression implicitly enacts gradient averaging using Neighborhood Interpolation*. This entails a nuanced balance between capturing gradient covariance and diminishing gradient noise. Notably, the sparse LR formulation is merely a specific instance of ours. Yet, our proposed algorithm doesn’t demand a markedly higher computational expense. This modest additional effort bestows upon us enhanced robustness.

## 2 PROBLEM SETUP AND FORMULATION

We first introduce the optimal transport (OT) problem in Kantorovich formulation with entropic regularization, which measures the distance between two distributions, defined in (4) below. The Wasserstein regression formulation as a generalization of the LR formulation is then proposed.

**The Kantorovich Problem.** Denote  $\mathcal{P}_2$  the set of probability measures with finite second moments. Let  $\mu, \nu \in \mathcal{P}_2$  and let  $\Pi(\mu, \nu)$  denote the set of probability measures in  $\mathcal{P}_2$  with marginal distributions equal to  $\mu$  and  $\nu$ . The 2-Wasserstein distance is defined as

$$W_2^2(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|^2 d\pi(x, y) + \varepsilon \int_{\mathbb{R}^d \times \mathbb{R}^d} \log \left( \frac{d\pi}{d\mu d\nu} \right) d\pi. \quad (4)$$

This is also referred to as the entropic OT problem, where the first term is the transportation cost between the two measures and the second term is the entropic regularization with multiplier  $\varepsilon$ .

**Sparse EWR Formulation.** The pruning problem formulation is defined in (5) below.

$$\min_{\mathbf{w}} Q(\mathbf{w}) = W_2^2(x(\mathbf{w}), y) + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 \quad (5a)$$

$$\text{s.t.} \quad \|\mathbf{w}\|_0 \leq k \quad (5b)$$

The term  $W_2^2(x(\mathbf{w}), y)$  is a Wasserstein distance between the two one-dimensional distributions  $x$  and  $y$  (or a sliced Wasserstein distance for  $\nabla\ell$  with two one-dimensional projections). The optimization is to alter  $\mathbf{w}$  such that the distance between the two distributions is minimized.

Let  $x$  and  $y$  follow the empirical distributions  $\{x_i\}_{i=1}^n$  and  $\{y_i\}_{i=1}^n$ . Denote by  $\mu_i$  and  $\nu_i$  the mass of the data points  $x_i$  and  $y_i$ , respectively. We use  $\mathbf{\Pi}$  to refer to a matrix representing the transportation probability between  $x$  and  $y$ , and  $\Pi$  the set of all such matrices, i.e.  $\Pi = \{\mathbf{\Pi} \mid \sum_{i=1}^n \pi_{ij} = \mu_j \forall j \text{ and } \sum_{j=1}^n \pi_{ij} = \nu_i \forall i\}$ , where  $\mu_i$  and  $\nu_j$  are marginal distributions. Then (5) reads:

$$\min_{\mathbf{w}} Q(\mathbf{w}) = \inf_{\mathbf{\Pi} \in \Pi} \left\{ \sum_{i=1}^n \sum_{j=1}^n \|x_i(\mathbf{w}) - y_j\|^2 \pi_{ij} + \varepsilon \sum_{i=1}^n \sum_{j=1}^n \log \left( \frac{\pi_{ij}}{\mu_i \nu_j} \right) \pi_{ij} \right\} + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 \quad (6a)$$

$$\text{s.t.} \quad \|\mathbf{w}\|_0 \leq k \quad (6b)$$

**LR as a Special Case.** Let  $\varepsilon = 0$ . Once we set  $\mathbf{\Pi}$  to be a diagonal matrix with constant value  $1/n$ , i.e.  $\text{diag}(1/n)$ , the mass transportation happens only between data point pairs  $(x_i, y_i)$  for  $i = 1, \dots, n$ . Therefore we have

$$Q_{\mathbf{\Pi}=\text{diag}(1/n)}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \|x_i(\mathbf{w}) - y_i\|^2 + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 = \frac{1}{n} \bar{Q}(\mathbf{w}), \quad (7)$$

i.e. the formulation (6) in this case degrades to the LR formulation in (3).

### 3 THEORETICAL ASPECTS

This section reveals some good theoretical properties of the Sparse EWR formulation for network pruning. We start with Proposition 1 below (proof in Appendix A.1) that states a geometry property of OT with squared Euclidean distance cost. Additionally, we demonstrate the Neighborhood Interpolation mechanism that happens implicitly in solving the EWR. Moreover, we show that such a mechanism strikes a balance in capturing gradient covariance and reducing gradient noise, with a brief discussion on the advantage of using entropic regularization in terms of sample complexity.

**Proposition 1** (Convex Hull Distance Equality). *Consider a set  $S$  and its convex hull  $\text{Conv}(S)$  in a Euclidean space, and an arbitrary point  $x$  in the space. For any probability measure  $\hat{\nu}$  on  $S$ , we can find a point  $y'$  in  $\text{Conv}(S)$  as  $y' = \int y d\nu(y)$  such that  $\|x - y'\|^2 = \int \|x - y\|^2 d\hat{\nu}(y)$ , where  $\nu$  is a measure on  $\text{Conv}(S)$ .*

**Neighborhood Interpolation.** In formulation (5), let  $W_x$  be the first term of  $W_2^2$  for an arbitrary given  $x$ , i.e.,  $W_x = \int \|x(\mathbf{w}) - y\|^2 d\pi(\cdot|x)(y)$ , where  $\pi(\cdot|x)(y)$  is a conditional measure given  $x$ . Now, divide the Euclidean space  $\mathbb{R}^d$  by subspaces  $S_1^y, S_2^y, \dots, S_n^y$  for  $y$ . For any conditional measure  $\pi(\cdot|x)(y)$  defined on any  $S_i^y$  ( $i = 1, \dots, n$ ), there exists a measure  $\nu(y)$  defined on  $\text{Conv}(S_i^y)$  such that the weighted distance from  $S_i^y$  to  $x$  equals the distance from  $x$  to a point  $y'$  in  $\text{Conv}(S_i^y)$ . Hence

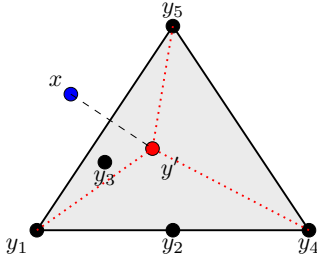
$$\begin{aligned} W_x &= \int_{S_1^y \cup S_2^y \cup \dots \cup S_n^y} \|x - y\|^2 d\pi(\cdot|x)(y) \\ &= \frac{1}{n} \sum_{i=1}^n \|x - y'_i\|^2 \text{ s.t. } y'_i = \int_{\text{Conv}(S_i^y)} y d\nu(y), \nu \in \mathbb{V}_i(x), i = 1, \dots, n. \end{aligned}$$

where  $\mathbb{V}_i(x)$  is the set of measures  $\nu$  that make the equality holds and  $\mathbb{V}_i(x) \neq \emptyset$  by Proposition 1.

Similarly, we define  $W_y$  for any given  $y$  and subspaces  $S_1^x, S_2^x, \dots, S_n^x$ ,

$$\begin{aligned} W_y &= \int_{S_1^x \cup S_2^x \cup \dots \cup S_n^x} \|x - y\|^2 d\pi(\cdot|y)(x) \\ &= \frac{1}{n} \sum_{i=1}^n \|x'_i - y\|^2 \text{ s.t. } x'_i = \int_{\text{Conv}(S_i^x)} x d\mu(x), \mu \in \mathbb{U}_i(y), i = 1, \dots, n. \end{aligned}$$

where  $\mu$  is a measure defined on  $\text{Conv}(S_i^x)$  and  $\mathbb{U}_i(y) \neq \emptyset$ .



We demonstrate the concept of “Neighborhood Interpolation” through an empirical distribution example. Define  $S$  as a subset of  $y$  such that for every element  $y_i \in S$ ,  $\pi_{x,i} > 0$ . Without loss of generality, we can denote  $S = \{y_1, y_2, y_3, y_4, y_5\}$ . The area shaded in gray, denoted as  $\text{Conv}(S)$ , represents the convex hull of  $S$ .  $W_x$  computes a weighted summation of distances between  $x$  and the points  $y_1, \dots, y_5$ . The weights  $\pi_{x,1}, \dots, \pi_{x,5}$  are decided by OT. A significant  $\pi_{x,i}$  typically implies that  $y_i$  is in proximity to  $x$ , indicating a neighborhood relation. By Proposition 1, this weighted distance is analogous to the distance between  $x$  and  $y'$ , where  $y'$  is derived from  $\text{Conv}(S)$ .

**Revisit the EWR formulation.** The integral of either  $W_x$  or  $W_y$  respectively on  $x$  or  $y$  gives the first term of  $W_2^2$ . One can then reformulate (5) as (8) below.

$$\min_{\mathbf{w}: \|\mathbf{w}\|_0 \leq k} Q(\mathbf{w}) = \frac{1}{2} \inf_{\pi} \left\{ \int W_x(\mathbf{w}) d\mu(x) + \int W_y(\mathbf{w}) d\nu(y) \right\} + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2. \quad (8)$$

Interpretation: The objective function calculates the Euclidean distance between a point  $x$  and  $n$  distinct points. These  $n$  points originate from  $n$  convex hulls, each shaped by different  $n$  subspaces within  $y$ . Similarly, the function measures the distance between each point  $y$  and  $m$  unique points derived from  $m$  convex hulls, each formed by distinct  $m$  subspaces within  $x$ .

We claim that the EWR formulation is more resilient to noisy gradient than its counterpart, the LR formulation given by (3). To understand this claim better, let us reimagine the problem using empirical distributions, as indicated by (6). In this context, we use  $x_i$  and  $y_i$  as substitutes for  $S_i^x$  and  $S_i^y$ . Moreover, the integration in both  $W_x$  and  $W_y$  is replaced with summations, offering a more insightful version of our initial EWR formulation, shown as (9).

$$\min_{\mathbf{w}: \|\mathbf{w}\|_0 \leq k} Q(\mathbf{w}) = \inf_{\Pi} \left\{ Q_{\Pi}(\mathbf{w}) + \varepsilon \sum_{i=1}^n \sum_{j=1}^n \log \left( \frac{\pi_{ij}}{\mu_i \nu_j} \right) \pi_{ij} \right\} \quad (9)$$

The notation  $Q_{\Pi}(\mathbf{w})$  defined in (10) denotes the part of the objective function given fixed  $\Pi$ :

$$Q_{\Pi}(\mathbf{w}) = \sum_{i=1}^n \sum_{j=1}^n \|x_i(\mathbf{w}) - y_j\|^2 \pi_{ij} + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 \quad (10a)$$

$$= \underbrace{\frac{1}{2} \sum_{i=1}^n \|x_i(\mathbf{w}) - y'_i\|^2}_{K_{\Pi}^{(1)}} + \underbrace{\frac{1}{2} \sum_{i=1}^n \|x'_i(\mathbf{w}) - y_i\|^2}_{K_{\Pi}^{(2)}} + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 \quad (10b)$$

In  $Q_{\Pi}(\mathbf{w})$ , for each index  $i$ , points  $x'_i$  and  $y'_i$  are chosen from the convex hulls formed by points in  $x$  and  $y$ , as per the guidelines of Proposition 1. Now, contrasting this with the LR model in (3), the objective  $\bar{Q}(\mathbf{w})$  aims for regression directly over the data points whereas every point from one empirical set is matched for Euclidean distance computation to a point derived from a convex combination of the other.

The infimum in (9) seeks the OT plan,  $\Pi$ , that aligns the empirical distributions  $x$  and  $y$  closely. In practical terms, for each data point  $x_i$ , only a subset of  $\{y_i\}_{i=1}^n$  will transport a substantial mass, rather than the entire set. This behavior of  $\Pi$  effectively defines  $n$  "neighborhoods" for each data point  $x_i$  within the empirical distribution of  $y$ . Here, a "neighborhood" refers to a group of data points in  $y$  that are proximate to a specific  $x_i$  in the Euclidean sense.

**Neighborhood Size Control.** A critical aspect of this formulation is the entropic regularization term, which is used to modulate the size of these neighborhoods. Specifically, increasing the value of  $\varepsilon$  amplifies the impact of the entropy term. This change broadens the neighborhoods, drawing more data points into the fold of the associated convex hulls. An illustrative extreme case is when  $\varepsilon = 0$ . Here, the OT does one-to-one matching, implying that each data point  $y_i$  primarily forms the convex hull independently. On the contrary, when  $\varepsilon \rightarrow \infty$ , all data points are equally weighted by  $\Pi$  and hence involved in forming the convex hull as a neighborhood.

**Capturing Covariance With Gradient Noise Reduction.** For an arbitrary  $\mathbf{w}$ , the EWR formulation essentially strikes a balance between gradient noise reduction and covariance capturing. We show the analysis for  $K_{\Pi}^{(1)}$  in (10), and  $K_{\Pi}^{(2)}$  follows similarly. Note that  $y'_i = \sum_{j=1}^n \nu_j^{(i)} y_j = \sum_{j=1}^n \nu_j^{(i)} \nabla \ell_j^{\top} \bar{\mathbf{w}}$ , where  $\nu^{(i)}$  are convex combination coefficients by Proposition 1. Denote  $\nabla \ell'_i{}^{\top} = \sum_{j=1}^n \nu_j^{(i)} \nabla \ell_j^{\top}$ , and  $\mathbf{G}' = [\nabla \ell'_1, \dots, \nabla \ell'_n]^{\top}$ . The term  $K_{\Pi}^{(1)}$  expands as follows.

$$\begin{aligned} K_{\Pi}^{(1)} &= \sum_{i=1}^n (\nabla \ell_i^{\top} \mathbf{w} - \nabla \ell'_i{}^{\top} \bar{\mathbf{w}})^{\top} (\nabla \ell_i^{\top} \mathbf{w} - \nabla \ell'_i{}^{\top} \bar{\mathbf{w}}) \\ &= \sum_{i=1}^n (\mathbf{w}^{\top} \nabla \ell_i \nabla \ell_i^{\top} \mathbf{w} - \mathbf{w}^{\top} \nabla \ell_i \nabla \ell'_i{}^{\top} \bar{\mathbf{w}} - \bar{\mathbf{w}}^{\top} \nabla \ell'_i \nabla \ell_i^{\top} \mathbf{w} + \bar{\mathbf{w}}^{\top} \nabla \ell'_i \nabla \ell'_i{}^{\top} \bar{\mathbf{w}}) \end{aligned} \quad (11)$$

Examining  $K_{\Pi}^{(1)}$  from (11), we see that it effectively replaces half of  $\nabla \ell_i$  with  $\nabla \ell'_i$ , a version obtained through weighted gradient averaging. Now let's compare the covariance between  $\nabla \ell_i$  and  $\nabla \ell'_i$ . Assume that  $\nabla \ell_i$  ( $1 \leq i \leq n$ ) are i.i.d. with the same covariance matrix  $\Sigma$ , then  $\mathbf{G}'$  is with equal or less noise than  $\mathbf{G}$ . To show this, denote the covariance matrix of each  $\nabla \ell'_i$  by

$$\Sigma'_i = \text{Cov} \left[ \sum_{j=1}^n \nu_j^{(i)} \nabla \ell_j \right] = \sum_{j=1}^n [\nu_j^{(i)}]^2 \Sigma.$$

The total variance of each gradient in  $\mathbf{G}'$  (i.e., the trace of  $\Sigma'_i$ ) is then

$$\text{trace}(\Sigma'_i) = \text{trace}\left(\sum_{j=1}^n [\nu_j^{(i)}]^2 \Sigma\right) = \sum_{j=1}^n [\nu_j^{(i)}]^2 \text{trace}(\Sigma) \leq \text{trace}(\Sigma).$$

The last inequality follows from the fact that  $\sum_{j=1}^n [\nu_j^{(i)}]^2 \leq 1$ , which is a consequence of the Cauchy-Schwarz inequality given that the coefficients  $\nu_j^{(i)}$  form a convex combination.

Originally, the covariance information of all data points is embedded in  $\nabla \ell_i \nabla \ell_i^\top$  for  $i = 1, 2, \dots, n$ . An alternative representation is  $\nabla \ell'_i \nabla \ell'_i{}^\top$ , which prioritizes noise reduction, but sacrifices some covariance information. Both  $\nabla \ell'_i \nabla \ell'_i{}^\top$  and  $\nabla \ell_i \nabla \ell_i{}^\top$  highlight a trade-off. Notably, both the original covariance  $\nabla \ell_i \nabla \ell_i{}^\top$  and its noise-reduced counterpart  $\nabla \ell'_i \nabla \ell'_i{}^\top$  are retained in (11).

**Difference From Averaging Prior to Optimization:** Next, we show that such gradient averaging differs from the averaging operation conducted prior to optimization. Let  $\mathbf{G}' = [\nabla \ell'_1, \nabla \ell'_2, \dots, \nabla \ell'_n]^\top$  such that  $\mathbf{G}'$  represents the row-wise convex combination of  $\mathbf{G}$ . Approximating the Hessian of the MIQP (2), two scenarios emerge: using  $\mathbf{G}$  that not performing averaging (case 1) and  $\mathbf{G}'$  that performs averaging before optimization (case 2).

**Case 1** is the original LR formulation (3). Denote by  $K$  below its term corresponding to  $K_{\mathbf{\Pi}}^{(1)}$ :

$$\begin{aligned} K &= (\mathbf{w} - \bar{\mathbf{w}})^\top \mathbf{G}^\top \mathbf{G} (\mathbf{w} - \bar{\mathbf{w}}) \\ &= \sum_{i=1}^n (\mathbf{w}^\top \nabla \ell_i \nabla \ell_i{}^\top \mathbf{w} - \mathbf{w}^\top \nabla \ell_i \nabla \ell_i{}^\top \bar{\mathbf{w}} - \bar{\mathbf{w}}^\top \nabla \ell_i \nabla \ell_i{}^\top \mathbf{w} + \bar{\mathbf{w}}^\top \nabla \ell_i \nabla \ell_i{}^\top \bar{\mathbf{w}}) \end{aligned} \quad (12)$$

**Case 2** uses the less-noisy row-wise convex combination matrix  $\mathbf{G}'$  instead of  $\mathbf{G}$ . Yet, the original covariance  $\nabla \ell_i \nabla \ell_i{}^\top$  is lost: Denote by  $K'$  the corresponding term, and we have

$$K' = \sum_{i=1}^n (\mathbf{w}^\top \nabla \ell'_i \nabla \ell'_i{}^\top \mathbf{w} - \mathbf{w}^\top \nabla \ell'_i \nabla \ell'_i{}^\top \bar{\mathbf{w}} - \bar{\mathbf{w}}^\top \nabla \ell'_i \nabla \ell'_i{}^\top \mathbf{w} + \bar{\mathbf{w}}^\top \nabla \ell'_i \nabla \ell'_i{}^\top \bar{\mathbf{w}}) \quad (13)$$

Inspecting the expressions, it can be observed that  $K_{\mathbf{\Pi}}^{(1)}$  (also  $K_{\mathbf{\Pi}}^{(2)}$ ) strikes a balance between  $K$  and  $K'$ . There are two notable extreme cases for  $\mathbf{\Pi}$ :

1.  $\mathbf{\Pi} = \text{diag}(1/n)$ . This corresponds to the LR formulation, as detailed in Section 2. A smaller value of  $\varepsilon$  steers the optimization in this direction.
2.  $\mathbf{\Pi} = (1/n^2)\mathbf{1} \cdot \mathbf{1}^\top$ . This arises when  $\varepsilon \rightarrow \infty$ , meaning the entropy term holds sway in the OT optimization. Here, mutual information is minimized to ensure an even contribution from data points in the convex combination. Both  $x'_i$  and  $y'_i$  are the arithmetic means of their respective sets, and all  $\nabla \ell'_i$  are equivalent to the averaged gradient over the  $n$  points. Importantly, the original covariance remains intact even in this edge case.

As  $n$  grows indefinitely, the empirical OT formulation from (6) approaches its continuous counterpart given by (5). Intuitively, a large dataset of high-quality training samples makes the empirical fisher a close approximation to the true fisher. In such situations,  $\varepsilon$  is set to zero. Brenier's theorem (Peyré, 2019) then suggests that the OT plan turns into a monotone map for costs represented by squared Euclidean distances. This means  $\mathbf{\Pi}$  tends towards  $\text{diag}(1/n)$ . Consequently, the Wasserstein distance formulation reduces to the Euclidean distance formulation, delivering optimal performance with ample data.

An advantage of employing the EWR formulation is its inherent capability of gradient averaging. This approach negates the need to manually determine the convex combination coefficients or resort to density estimation to pinpoint the nearest gradient neighbors for averaging. Importantly, this seamless trade-off has an advantage over using Euclidean distance with gradient averaging performed prior to optimization. The reason is that the original covariance information will inevitably be lost in the formulation (13), irrespective of the chosen averaging method.

**Sample Complexity** of  $W_2^2(\mu, \nu)$  is narrowed to  $O(1/\sqrt{n})$  from  $O(1/n^{\frac{1}{4}})$  by the entropic regularization term. Please see Appendix A.2 for details.

## 4 ALGORITHM DESIGN

**Algorithmic Framework.** The algorithm addresses the network pruning problem defined in (5). Drawing inspiration from (Chen et al., 2022), the algorithm incrementally adjusts the sparsity of the weights vector  $\mathbf{w}$  by using a descending sequence of non-zero elements  $k_0, \dots, k_T$ . During each sparsity level, the weights  $\mathbf{w}$  and the transportation plan  $\Pi$  (can be obtained with efficient algorithms; see Appendix A.3) are refined iteratively.

---

### Algorithm 1 Sparse Entropic Wasserstein Regression Pruning (SWAP)

---

**Input:** Number of pruning stages  $T$ , pre-pruning weights  $\bar{\mathbf{w}}$ , target sparsity  $k$ , regularization parameter  $\lambda, \varepsilon$ , batches  $\mathcal{B}_0, \mathcal{B}_1, \dots, \mathcal{B}_T$ , optimization step size  $\tau > 0$ .

**Output:** Post-pruning weights  $\mathbf{w}$ , satisfying  $\|\mathbf{w}\| \leq k$

- 1: Set  $k_0, k_1, \dots, k_T$  as a descending sequence, with  $k_0 < p$  and  $k_T = k$ .
  - 2:  $\mathbf{w}^{(0)} \leftarrow \bar{\mathbf{w}}$
  - 3: **for**  $t \leftarrow 0, 1, \dots, T$  **do**
  - 4:   Compute  $\mathbf{G} = [\nabla \ell_1(\bar{\mathbf{w}}), \dots, \nabla \ell_n(\bar{\mathbf{w}})]^\top$  with batch  $\mathcal{B}_t$
  - 5:    $\mathbf{x}, \mathbf{y} \leftarrow \mathbf{G}\mathbf{w}^{(t)}, \mathbf{G}\bar{\mathbf{w}}$
  - 6:   Compute the pairwise Euclidean distance matrix  $\mathbf{C}$  between  $\mathbf{x}$  and  $\mathbf{y}$
  - 7:   Compute OT planning  $\Pi^{(t)}$  (see Appendix A.3)
  - 8:    $\nabla Q \leftarrow \mathbf{G}^\top (\Pi(\mathbf{G}\mathbf{w}^{(t)} - \mathbf{G}\bar{\mathbf{w}})) + \lambda(\mathbf{w}^{(t)} - \bar{\mathbf{w}})$
  - 9:    $\mathbf{w}^{(t+\frac{1}{2})} \leftarrow \mathbf{w}^{(t)} - \tau \nabla Q$
  - 10:    $\mathbf{w}^{(t+1)} \leftarrow$  Select from  $\mathbf{w}^{(t+\frac{1}{2})}$   $k_t$  components having largest absolute values; Others zero
  - 11:    $\bar{\mathbf{w}} \leftarrow \mathbf{w}^{(t+1)}$
  - 12: **end for**
  - 13:  $\mathbf{w} = \mathbf{w}^{(T+1)}$
- 

**Weights Optimization.** The weights  $\mathbf{w}$  are optimized using the stochastic gradient descent (SGD) paired with the iterative hard thresholding (IHT) algorithm. We use  $\nabla Q$  to represent the derivative of  $Q(\mathbf{w})$  for brevity, with its comprehensive derivation in Appendix A.4. The expression is

$$\nabla Q = \mathbf{G}^\top (\Pi(\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}})) + \lambda(\mathbf{w} - \bar{\mathbf{w}}). \quad (14)$$

Following the weight updates driven by SGD (as seen in line 9 of Algorithm 1), the IHT method is applied. Here, the  $k_t$  components of  $\mathbf{w}$  with the largest magnitudes are retained, while the remaining are set to zero, ensuring adherence to the sparsity criteria.

A vital component of the optimization process is the choice of the stepsize  $\tau$  (referenced in line 9). Although a straightforward approach might be to set  $\tau = \frac{1}{L}$  (where  $L$  denotes the Lipschitz constant of  $Q$ ), better performance can be achieved when the stepsize is optimized using the methodology proposed in (Chen et al., 2022, Algorithm 2). For the quadratic function  $Q$ , the Lipschitz constant  $L$  is given by  $L = n\lambda + \|G\|_{op}^2$ , where  $\|\cdot\|_{op}$  indicates the foremost singular value.

Line 10 in Algorithm 1 employs the IHT method that is commonly used in sparse learning, which together with line 9, forms a projected gradient descent algorithm. It finds a sparse representation of the updated gradient in line 9. Intuitively, IHT keeps the dominant model weights and essentially preserves the most impactful aspects of the to-be-trimmed model. Although there exist alternative strategies for refining the IHT solution—including active set updates, coordinate descent, and the Woodbury formula’s back-solving—a discussion on these falls outside the scope of this paper. For in-depth exploration, especially with respect to the specialized case described in (7), one can consult (Bhatia et al., 2015; Chen & Mazumder, 2020; Hazimeh & Mazumder, 2020; Benbaki et al., 2023).

## 5 NUMERICAL RESULTS

Our method is compared with several existing SoTA methods including MP (magnitude pruning (Mozer & Smolensky, 1989)), WF (WoodFisher (Singh & Alistarh, 2020)), CBS (Combinatorial Brain Surgeon (Yu et al., 2022)), and LR (i.e. the sparse LR formulation adopted by (Chen et al., 2022)). We refer to our proposed method as EWR (i.e. sparse entropic Wasserstein regression).

Table 1: Model Pruning Accuracy Benchmarking. Five runs are taken for LR and EWR, with the mean and 95% confidence interval (in the brackets) reported. The data of MP, WF, and CBS are copied from (Yu et al., 2022). The superscript “ $+\sigma$ ” indicates that 20% of data is with noise. Bold texts imply the best performance, with 0.1 percentage as tolerance in difference. The sparsity column is the target sparsity.

Network	Sparsity	MP	WF	CBS	LR (EWR $_{\mathbf{\Pi}=\mathbf{I}/n}$ )	EWR (proposed)
MLPNet on MNIST (93.97%)	0.5	93.93	94.02	93.96	<b>95.26</b> ( $\pm 0.03$ )	<b>95.24</b> ( $\pm 0.03$ )
	0.6	93.78	93.82	93.96	<b>95.13</b> ( $\pm 0.02$ )	<b>95.13</b> ( $\pm 0.01$ )
	0.7	93.62	93.77	93.98	<b>94.93</b> ( $\pm 0.03$ )	<b>95.05</b> ( $\pm 0.04$ )
	0.8	92.89	93.57	93.90	<b>94.82</b> ( $\pm 0.04$ )	<b>94.84</b> ( $\pm 0.03$ )
	0.9	90.30	91.69	93.14	<b>94.32</b> ( $\pm 0.05$ )	<b>94.30</b> ( $\pm 0.05$ )
	0.95	83.64	85.54	88.92	<b>92.82</b> ( $\pm 0.06$ )	<b>92.86</b> ( $\pm 0.05$ )
	0.95 $^{+\sigma}$	-	-	-	90.11 ( $\pm 0.08$ )	<b>90.50</b> ( $\pm 0.07$ )
	0.98	32.25	38.26	55.45	84.43 ( $\pm 0.10$ )	<b>85.71</b> ( $\pm 0.09$ )
	0.98 $^{+\sigma}$	-	-	-	82.12 ( $\pm 0.11$ )	<b>83.69</b> ( $\pm 0.10$ )
ResNet20 on CIFAR10 (91.36%)	0.5	88.44	90.23	90.58	<b>92.06</b> ( $\pm 0.04$ )	<b>92.04</b> ( $\pm 0.03$ )
	0.6	85.24	87.96	88.88	<b>91.98</b> ( $\pm 0.09$ )	<b>91.98</b> ( $\pm 0.09$ )
	0.7	78.79	81.05	81.84	91.09 ( $\pm 0.10$ )	<b>91.89</b> ( $\pm 0.10$ )
	0.8	54.01	62.63	51.28	89.00 ( $\pm 0.12$ )	<b>90.15</b> ( $\pm 0.09$ )
	0.9	11.79	11.49	13.68	87.63 ( $\pm 0.11$ )	<b>88.82</b> ( $\pm 0.10$ )
	0.95	-	-	-	80.25 ( $\pm 0.17$ )	<b>81.33</b> ( $\pm 0.15$ )
	0.95 $^{+\sigma}$	-	-	-	77.37 ( $\pm 0.18$ )	<b>79.05</b> ( $\pm 0.16$ )
	0.98	-	-	-	68.15 ( $\pm 0.27$ )	<b>69.21</b> ( $\pm 0.24$ )
	0.98 $^{+\sigma}$	-	-	-	65.04 ( $\pm 0.27$ )	<b>68.01</b> ( $\pm 0.25$ )
ResNet50 on CIFAR10 (92.78%)	0.95	-	-	-	83.75 ( $\pm 0.14$ )	<b>84.96</b> ( $\pm 0.15$ )
	0.95 $^{+\sigma}$	-	-	-	82.34 ( $\pm 0.16$ )	<b>84.92</b> ( $\pm 0.17$ )
	0.98	-	-	-	81.04 ( $\pm 0.14$ )	<b>82.85</b> ( $\pm 0.20$ )
	0.98 $^{+\sigma}$	-	-	-	80.11 ( $\pm 0.23$ )	<b>82.94</b> ( $\pm 0.22$ )
MobileNetV1 on ImageNet (71.95%)	0.3	71.60	<b>71.88</b>	<b>71.87</b>	71.14 ( $\pm 0.08$ )	<b>71.87</b> ( $\pm 0.05$ )
	0.4	69.16	71.15	<b>71.45</b>	71.12 ( $\pm 0.10$ )	<b>71.44</b> ( $\pm 0.07$ )
	0.5	62.61	68.91	70.21	70.12 ( $\pm 0.13$ )	<b>71.12</b> ( $\pm 0.18$ )
	0.6	41.94	60.90	66.37	70.05 ( $\pm 0.22$ )	<b>70.92</b> ( $\pm 0.18$ )
	0.7	6.78	29.36	55.11	68.15 ( $\pm 0.17$ )	<b>69.26</b> ( $\pm 0.13$ )
	0.8	0.11	0.24	16.38	65.72 ( $\pm 0.19$ )	<b>66.82</b> ( $\pm 0.14$ )
	0.8 $^{+\sigma}$	-	-	-	60.29 ( $\pm 0.18$ )	<b>63.62</b> ( $\pm 0.15$ )
	0.9	-	-	-	47.65 ( $\pm 0.15$ )	<b>49.43</b> ( $\pm 0.13$ )
	0.9 $^{+\sigma}$	-	-	-	44.55 ( $\pm 0.16$ )	<b>47.98</b> ( $\pm 0.16$ )

Note that LR is a special instance of EWR, with  $\mathbf{\Pi} = \text{diag}(1/n)$ . All the methods are benchmarked on pre-trained neural networks: MLPNet (30K parameters) trained on MNIST (LeCun et al., 1998), ResNet20 (200K parameters) and ResNet50 (25M parameters) (He et al., 2016) trained on CIFAR10 (Krizhevsky et al., 2009), and MobileNetV1 (Howard et al., 2017) (4.2M parameters) trained on ImageNet (Deng et al., 2009). The experiment setup for reproducibility<sup>1</sup> is detailed in Appendix A.5. We deliver more experiments results in Appendix A.6-A.9.

**Model Accuracy Performance Benchmarking.** Table 1 compares different networks across various sparsity levels, utilizing different methods. MLPNet’s performance on MNIST is consistent across different sparsity levels, with both LR and the proposed EWR method showing superior performance. The advantages of EWR over the others are reflected by the three more challenging tasks ResNet20 and ResNet50 on CIFAR10 and MobileNetV1 on ImageNet, especially in the presence of noisy gradients. In summary, the proposed EWR method consistently outperforms or matches other methods. The LR method performs well at lower sparsity levels but is surpassed otherwise.


<sup>1</sup>The code is available on   
<https://github.com/youlei202/Entropic-Wasserstein-Pruning>



Table 2: Comparison of testing loss values (no fine-tuning) for ResNet20. The result is averaged over 25 runs. The 90% confidence interval is reported. The target sparsity is set to be 0.95.

Sparsity	10% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.95	2.83 ( $\pm 0.02$ )	<b>2.75</b> ( $\pm 0.02$ )	<b>2.87%</b>	2.86 ( $\pm 0.02$ )	<b>2.74</b> ( $\pm 0.01$ )	<b>4.35%</b>
0.84	1.58 ( $\pm 0.01$ )	<b>1.54</b> ( $\pm 0.01$ )	<b>2.73%</b>	1.62 ( $\pm 0.03$ )	<b>1.54</b> ( $\pm 0.02$ )	<b>5.15%</b>
0.74	0.66 ( $\pm 0.00$ )	<b>0.64</b> ( $\pm 0.00$ )	<b>3.03%</b>	0.66 ( $\pm 0.00$ )	<b>0.65</b> ( $\pm 0.00$ )	<b>1.32%</b>
0.63	0.35 ( $\pm 0.00$ )	0.35 ( $\pm 0.00$ )	0.00%	0.35 ( $\pm 0.00$ )	0.35 ( $\pm 0.00$ )	0.85%
Sparsity	25% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.95	2.87 ( $\pm 0.02$ )	<b>2.77</b> ( $\pm 0.01$ )	<b>3.50%</b>	2.89 ( $\pm 0.02$ )	<b>2.79</b> ( $\pm 0.02$ )	<b>3.82%</b>
0.84	1.72 ( $\pm 0.02$ )	<b>1.65</b> ( $\pm 0.02$ )	<b>4.07%</b>	1.76 ( $\pm 0.02$ )	<b>1.69</b> ( $\pm 0.02$ )	<b>4.05%</b>
0.74	0.67 ( $\pm 0.01$ )	0.67 ( $\pm 0.00$ )	0.49%	0.68 ( $\pm 0.00$ )	<b>0.67</b> ( $\pm 0.00$ )	<b>1.55%</b>
0.63	0.36 ( $\pm 0.00$ )	0.36 ( $\pm 0.00$ )	0.00%	0.35 ( $\pm 0.00$ )	0.35 ( $\pm 0.00$ )	0.00%

Table 3: Comparison of testing loss values (no fine-tuning) for MobileNetV1. The result is averaged from 10 runs. The 90% confidence interval is reported. The target sparsity is set to be 0.75.

Sparsity	10% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.75	4.54 ( $\pm 0.06$ )	<b>4.34</b> ( $\pm 0.06$ )	<b>4.41%</b>	4.62 ( $\pm 0.07$ )	<b>4.40</b> ( $\pm 0.06$ )	<b>5.02%</b>
0.63	2.53 ( $\pm 0.04$ )	<b>2.46</b> ( $\pm 0.03$ )	<b>2.61%</b>	2.56 ( $\pm 0.04$ )	<b>2.48</b> ( $\pm 0.04$ )	<b>3.23%</b>
0.53	1.64 ( $\pm 0.02$ )	<b>1.56</b> ( $\pm 0.02$ )	<b>5.16%</b>	1.66 ( $\pm 0.03$ )	<b>1.56</b> ( $\pm 0.02$ )	<b>6.01%</b>
0.42	1.30 ( $\pm 0.00$ )	1.30 ( $\pm 0.00$ )	0.00%	1.30 ( $\pm 0.00$ )	1.30 ( $\pm 0.00$ )	0.00%
Sparsity	25% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.75	4.93 ( $\pm 0.06$ )	<b>4.53</b> ( $\pm 0.06$ )	<b>8.13%</b>	5.02 ( $\pm 0.06$ )	<b>4.66</b> ( $\pm 0.05$ )	<b>7.92%</b>
0.63	2.54 ( $\pm 0.04$ )	<b>2.44</b> ( $\pm 0.03$ )	<b>4.01%</b>	2.57 ( $\pm 0.04$ )	<b>2.45</b> ( $\pm 0.03$ )	<b>5.00%</b>
0.53	1.66 ( $\pm 0.02$ )	<b>1.57</b> ( $\pm 0.02$ )	<b>5.16%</b>	1.66 ( $\pm 0.02$ )	<b>1.55</b> ( $\pm 0.02$ )	<b>6.63%</b>
0.42	1.30 ( $\pm 0.00$ )	1.30 ( $\pm 0.00$ )	0.30%	1.31 ( $\pm 0.00$ )	1.31 ( $\pm 0.00$ )	0.00%

**Robustness with Noisy Gradient.** From Section 3, EWR differs from LR in terms of gradient noise reduction achieved by solving the OT problem to obtain a group of non-trivial data pair weighting coefficients. Hence, LR that has the transportation plan  $\Pi$  fixed to  $\text{diag}(1/n)$  naturally serves as a baseline for evaluating the effectiveness of such optimization in terms of robustness against noise. In two noisy scenarios, 10%, and 25%, we evaluate loss at noise levels of  $\sigma$  and  $2\sigma$  across varying sparsity. Tables 2 and 3 contrast the loss difference between LR and EWR. EWR consistently outperforms LR in both ResNet20 and MobileNetV1, most notably in noisy conditions and at higher sparsity. The peak performance difference is 8.13% favoring EWR on MobileNetV1 at 0.75 sparsity with 25% noise. Hence, EWR outperforms LR.

## 6 CONCLUSIONS AND FUTURE IMPACT

The paper offers a novel formulation based on EWR, which strikes a balance between covariance information preservation and noise reduction. The work suggested promising avenues for applications in large-scale model compression, though it may require further empirical validation and exploration of practical implementations.

## REFERENCES

- Fan Ang, Li Chen, Nan Zhao, Yunfei Chen, Weidong Wang, and F Richard Yu. Robust federated learning with noisy communication. *IEEE Transactions on Communications*, 68(6):3452–3464, 2020.
- Riade Benbaki, Wenyu Chen, Xiang Meng, Hussein Hazimeh, Natalia Ponomareva, Zhe Zhao, and Rahul Mazumder. Fast as chita: Neural network pruning with combinatorial optimization. *arXiv preprint arXiv:2302.14623*, 2023.
- Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding. *Advances in neural information processing systems*, 28, 2015.
- Wenyu Chen and Rahul Mazumder. Multivariate convex regression at scale. *arXiv preprint arXiv:2005.11588*, 2020.
- Wenyu Chen, Riade Benbaki, Xiang Meng, and Rahul Mazumder. Network pruning at scale: A discrete optimization approach. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.
- Sahaj Garg, Joe Lou, Anirudh Jain, Zhimu Guo, Bhavin J Shastri, and Mitchell Nahmias. Dynamic precision analog computing for neural networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 29(2: Optical Computing):1–12, 2022.
- Aude Genevay, Lénaïc Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. Sample complexity of sinkhorn divergences. In *The 22nd international conference on artificial intelligence and statistics*, pp. 1574–1583. PMLR, 2019.
- Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006.
- Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.
- Hussein Hazimeh and Rahul Mazumder. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Operations Research*, 68(5):1517–1537, 2020.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. Fedtiny: Pruned federated learning towards specialized tiny models. *arXiv preprint arXiv:2212.01977*, 2022.
- Berivan Isik, Kristy Choi, Xin Zheng, Tsachy Weissman, Stefano Ermon, H-S Philip Wong, and Armin Alaghi. Neural network compression for noisy storage devices. *ACM Transactions on Embedded Computing Systems*, 22(3):1–29, 2023.
- Hicham Janati, Boris Muzellec, Gabriel Peyré, and Marco Cuturi. Entropic optimal transport between unbalanced gaussian measures has a closed form. *Advances in neural information processing systems*, 33:10468–10479, 2020.
- Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

- Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.
- Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. Early stopping without a validation set. *arXiv preprint arXiv:1703.09580*, 2017.
- Michael C Mozer and Paul Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1(1):3–16, 1989.
- Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour, and Umut Simsekli. Statistical and topological properties of sliced probability divergences. *Advances in Neural Information Processing Systems*, 33:20802–20812, 2020.
- Kimia Nadjahi, Alain Durmus, Pierre E Jacob, Roland Badeau, and Umut Simsekli. Fast approximation of the sliced-wasserstein distance using concentration of random projections. *Advances in Neural Information Processing Systems*, 34:12411–12424, 2021.
- Gabriel Peyré. Course notes on computational optimal transport. 2019.
- Galen Reeves. Conditional central limit theorems for gaussian projections. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 3045–3049. IEEE, 2017.
- Julien Niklas Siems, Aaron Klein, Cedric Archambeau, and Maren Mahsereci. Dynamic pruning of a neural network via gradient signal-to-noise ratio. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.
- Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.
- Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.
- Vladimir Nikolaevich Sudakov. Typical distributions of linear functionals in finite-dimensional spaces of higher dimension. In *Doklady Akademii Nauk*, volume 243, pp. 1402–1405. Russian Academy of Sciences, 1978.
- Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K Leung. Overcoming noisy and irrelevant data in federated learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 5020–5027. IEEE, 2021.
- Berkay Turan, César A Uribe, Hoi-To Wai, and Mahnoosh Alizadeh. Robust distributed optimization with randomly corrupted gradients. *IEEE Transactions on Signal Processing*, 70:3484–3498, 2022.
- Kai Yang, Tao Jiang, Yuanming Shi, and Zhi Ding. Federated learning via over-the-air computation. *IEEE transactions on wireless communications*, 19(3):2022–2035, 2020.
- Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In *International Conference on Machine Learning*, pp. 25668–25683. PMLR, 2022.

## A APPENDIX

### A.1 PROOF OF PROPOSITION 1

(*Convex Hull Distance Equality*) Consider a set  $S$  and its convex hull  $\text{Conv}(S)$  in a Euclidean space, and an arbitrary point  $x$  in the space. For any probability measure  $\hat{\nu}$  on  $S$ , we can find a point  $y'$  in  $\text{Conv}(S)$  as  $y' = \int y d\nu(y)$  such that  $\|x - y'\|^2 = \int \|x - y\|^2 d\hat{\nu}(y)$ , where  $\nu$  is a measure on  $\text{Conv}(S)$ .

*Proof.* We define a function  $f(\nu) = \|x - y'\|^2$ , where  $y' = \int y d\nu(y)$ . This function takes the empirical measure  $\nu$  as input and computes the squared Euclidean distance between  $x$  and  $y'$ . Similarly, we define a function  $g(\nu) = \int \|x - y\|^2 d\nu(y)$ , which computes the weighted average of squared Euclidean distances between  $x$  and the points in the set  $S$  according to the probability measure  $\nu$ .

Without loss of generality, let's assume that  $S$  is contained within its convex hull  $\text{Conv}(S)$ . Then, the right-hand side of the equation to be proved in the theorem takes the minimum and maximum values, respectively, at points within  $\text{Conv}(S)$ , i.e.,

$$g_{\min} = \inf_{\nu} \int \|x - y\|^2 d\nu(y)$$

and

$$g_{\max} = \sup_{\nu} \int \|x - y\|^2 d\nu(y).$$

The function  $f(\nu)$  takes its maximum value at  $z$ , where  $z$  is the farthest point inside  $\text{Conv}(S)$  from  $x$ , i.e.,

$$f_{\max} = \sup_{\nu} \|x - y'\|^2 = \|x - z\|^2.$$

Similarly, the minimum value is obtained at  $y' = z'$ , where  $z'$  is the closest point inside  $\text{Conv}(S)$  to  $x$ . The minimum value can reach zero if  $x$  is inside  $\text{Conv}(S)$ . Formally,

$$f_{\min} = \inf_{\mu} \{ \|x - z\|^2 \mid z \in \text{Conv}(S) \}.$$

To establish that  $f_{\max} \geq g_{\max}$ , we consider the maximum values of the two functions. The function  $f(\nu)$  takes its maximum value at  $z$ , which is the farthest point inside  $\text{Conv}(S)$  from  $x$ . This means that  $f_{\max}$  is the squared Euclidean distance between  $x$  and  $z$ , i.e.,  $f_{\max} = \|x - z\|^2$ . On the other hand, the function  $g(\nu)$  computes the weighted average of squared Euclidean distances between  $x$  and the points in  $S$ . The maximum value of  $g(\nu)$ , denoted as  $g_{\max}$ , corresponds to the squared Euclidean distance between  $x$  and the farthest point in  $S$ . Since  $\text{Conv}(S)$  contains  $S$ , it follows that  $z$  is farther from  $x$  than any point in  $S$ . Therefore,  $f_{\max} \geq g_{\max}$ . Similarly, since  $\text{Conv}(S)$  contains  $S$  and  $z'$  is the closest point in  $\text{Conv}(S)$  to  $x$ , it follows that  $f_{\min} \leq g_{\min}$ .

We apply the intermediate value theorem to finish the proof. For any measure  $\nu$ , let  $h(\mu) = f(\mu) - g(\nu)$ . Once can find two measures  $\mu_1$  and  $\mu_2$  such that  $h(\mu_1) \leq 0$  and  $h(\mu_2) \geq 0$ , hence the zero point of  $h$  exists, followed by that there is a corresponding value of  $\mu$  such that  $f(\mu) = g(\nu)$ .

Hence, the conclusion.  $\square$

### A.2 SAMPLE COMPLEXITY

The efficiency of an estimator is often measured by its ability to deliver accurate estimates with fewer samples, a trait referred to as 'good sample complexity'. The entropic Wasserstein distance formulation is posited to enhance noise reduction by optimizing this sample complexity. When  $\varepsilon = 0$ , the sample complexity of  $W_2^2(\mu, \nu)$  stands at  $O(1/n^{\frac{1}{4}})$ . Specifically,

$$\lim_{n \rightarrow \infty} \mathbb{E}[W_2^2(\mu_n, \nu_n) - W_2^2(\mu, \nu)] = O(1/n^{\frac{1}{4}})$$

as per (Nadjahi et al., 2020, Corollary 2). As  $\varepsilon$  approaches infinity,  $W_2^2(\mu, \nu)$  leverages the beneficial characteristics of the maximum mean discrepancies (MMD) (Gretton et al., 2006), which narrows the sample complexity to  $O(1/\sqrt{n})$  according to (Genevay et al., 2019, Theorem 3). The coefficient  $\varepsilon$  acts as a regulator, adjusting the sample complexity within this specified range.

### A.3 OT PLAN OPTIMIZATION

For an arbitrary  $\mathbf{w}^{(t)}$ , the transportation plan  $\Pi^{(t)}$  is obtained by solving  $W_2^2(x(\mathbf{w}^{(t)}), y)$ . The optimization of the transportation plan  $\Pi$  in line 7 of Algorithm 1 is based on the Sinkhorn-Knopp algorithm, shown in Algorithm 2 below.

---

#### Algorithm 2 Sinkhorn-Knopp Algorithm for Regularized OT

---

**Input:**  $\mathbf{C}$  ( $n \times n$  Euclidean distance matrix),  $\mu, \nu$  (probability mass of  $\mathbf{x}$  and  $\mathbf{y}$ ),  $\varepsilon$  (regularization parameter),  $\epsilon$  (tolerance for stopping criterion)

**Output:**  $\Pi$

- 1: Initialize  $\mathbf{K} = \exp(-\mathbf{C}/\varepsilon)$ ,  $\mathbf{u} = \mathbf{1}_n/n$ ,  $\mathbf{v} = \mathbf{1}_n/n$
  - 2: **repeat**
  - 3:      $\mathbf{u} = \mu/\mathbf{K}\mathbf{v}$
  - 4:      $\mathbf{v} = \nu/\mathbf{K}^\top\mathbf{u}$
  - 5: **until**  $\sup \{\|\mathbf{a} - \text{diag}(\mathbf{u})\mathbf{K}\mathbf{v}\|_\infty, \|\mathbf{b} - \text{diag}(\mathbf{v})\mathbf{K}^\top\mathbf{u}\|_\infty\} < \epsilon$
  - 6: **return**  $\Pi = \text{diag}(\mathbf{u}) \cdot \mathbf{K} \cdot \text{diag}(\mathbf{v})$
- 

The Sinkhorn-Knopp algorithm is an iterative method used to solve regularized OT problems, aiming to find a transportation plan that minimizes total cost while adhering to specific source and target probability distributions. In the algorithm, an initial matrix  $\mathbf{K}$  is formed using the exponential of the negative cost matrix divided by a regularization parameter, and two vectors are initialized as uniform distributions. These vectors are then iteratively updated using rules derived from the Kullback-Leibler divergence, seeking to align the row and column sums of the resulting matrix with the given source and target distributions. The process continues until the maximum difference between the actual and desired row and column sums falls below a specified tolerance, ensuring the solution is feasible.

Note that  $x$  and  $y$  are one-dimensional projections of high-dimensional random variables  $\nabla\ell$ , of which the dimension is the number of parameters of the neural network, a.k.a.  $p$ . Prior research has demonstrated that when certain moderate criteria are met, the distribution of lower-dimensional versions of high-dimensional random variables tends to closely follow a Gaussian (or normal) distribution (Sudakov, 1978; Reeves, 2017; Nadjahi et al., 2021). In the research by (Janati et al., 2020), it is highlighted that given  $x$  and  $y$  obeying the Gaussian distribution,  $W_2^2(x, y)$  can be reduced to a concise closed-form solution. As illustrated in Algorithm 3, the OT plan can be directly computed relying solely on the statistics of the empirical distributions  $\mathbf{x}$  and  $\mathbf{y}$ .

---

#### Algorithm 3 Closed Form Algorithm for Regularized OT

---

**Input:**  $\psi = \sqrt{\frac{\varepsilon}{2}}$ ,  $a = \text{mean}(\mathbf{x})$ ,  $b = \text{mean}(\mathbf{y})$ ,  $\sigma_a^2 = \text{var}(\mathbf{x})$ ,  $\sigma_b^2 = \text{var}(\mathbf{y})$ ,  $d_\psi = (4\sigma_a\sigma_b^2\sigma_a + \psi^4)^{\frac{1}{2}}$

**Output:**

$$\Pi \sim \mathcal{N} \left( \begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} \sigma_a^2 & \frac{1}{2}\sigma_a d_\psi \sigma_a^{-1} \\ \frac{1}{2}\sigma_a d_\psi \sigma_a^{-1} & \sigma_b^2 \end{bmatrix} \right)$$


---

Specifically, the OT plan, represented as  $\Pi$ , is governed by the mean and variance of both  $x$  and  $y$ , in tandem with the regularization parameter  $\varepsilon$ . To derive the  $n \times n$  matrix  $\Pi$ ,  $(x_i, y_i)$ , where  $i$  spans from 1 to  $n$ , are employed to extract  $n$  samples from the distribution produced in Algorithm 3.

As depicted in Figure 1, a comparative evaluation of the two algorithms in terms of attaining the OT objective,  $W_2^2$ , is presented. The entropic regularization term’s value has been omitted considering its non-impact on the pruning optimization of  $\mathbf{w}$ , given a constant  $\Pi$ . Notably, the disparity between the two algorithms in their objective optimization magnifies as  $\varepsilon$  increases. Typically, in real-world applications, the value of  $\varepsilon$  oscillates between 0 and 10 (and we use  $\varepsilon = 1$  most frequently in our experiments). The variance in the performance of the two algorithms concerning OT planning remains trivial, echoing our practical observations during the algorithmic implementation in this study.

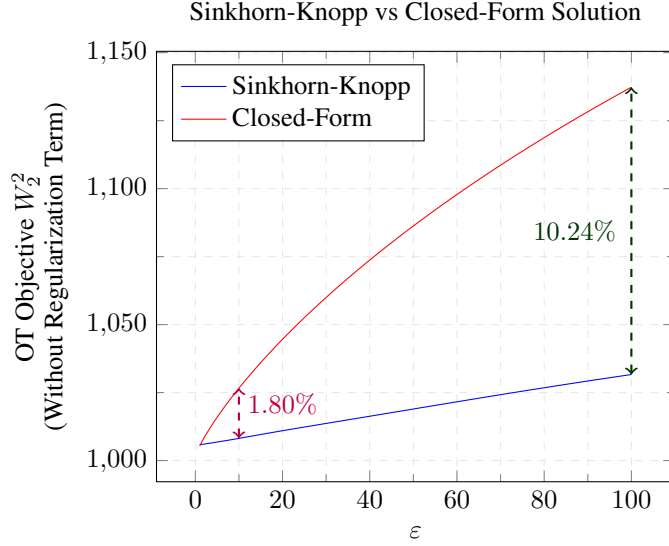


Figure 1: Comparison between the Sinkhorn-Knopp (i.e. Algorithm 2) and the closed-form solution (i.e. Algorithm 3). The plot is made based on the data of ResNet20 trained on Cifar10. The relative difference is computed by (red - blue) / blue.

#### A.4 DERIVATIVE OF $Q(\mathbf{w})$

Let’s start by revisiting the function  $Q(\mathbf{w})$ :

$$Q(\mathbf{w}) = \left\{ \sum_{i=1}^n \sum_{j=1}^n \|x_i(\mathbf{w}) - y_j\|^2 \pi_{ij} \right\} + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 \quad (15)$$

Given:

$$\begin{aligned} x_i(\mathbf{w}) &= \nabla \ell_i^\top \mathbf{w} \\ y_j &= \nabla \ell_j^\top \bar{\mathbf{w}} \end{aligned}$$

**Differentiating**  $Q(\mathbf{w})$  with respect to  $\mathbf{w}$ :

$$\begin{aligned} \nabla Q(\mathbf{w}) &= \nabla \left[ \left\{ \sum_{i=1}^n \sum_{j=1}^n \left\| \nabla \ell_i^\top \mathbf{w} - \nabla \ell_j^\top \bar{\mathbf{w}} \right\|^2 \pi_{ij} \right\} + \lambda \|\mathbf{w} - \bar{\mathbf{w}}\|^2 \right] \\ &= 2\lambda(\mathbf{w} - \bar{\mathbf{w}}) + \nabla \left[ \left\{ \sum_{i=1}^n \sum_{j=1}^n \left\| \nabla \ell_i^\top \mathbf{w} - \nabla \ell_j^\top \bar{\mathbf{w}} \right\|^2 \pi_{ij} \right\} \right] \end{aligned} \quad (16)$$

For the gradient of the inner term, consider:

$$\nabla \left[ \left\{ \sum_{i=1}^n \sum_{j=1}^n \left\| \nabla \ell_i^\top \mathbf{w} - \nabla \ell_j^\top \bar{\mathbf{w}} \right\|^2 \pi_{ij} \right\} \right] = \sum_{i=1}^n \sum_{j=1}^n 2\pi_{ij} (\nabla \ell_i^\top \mathbf{w} - \nabla \ell_j^\top \bar{\mathbf{w}}) \nabla \ell_i \quad (17)$$

**Expressing in Matrix Form.** Given matrices:

$$\mathbf{G} = \begin{bmatrix} \nabla \ell_1 \\ \nabla \ell_2 \\ \vdots \\ \nabla \ell_n \end{bmatrix}$$

$\mathbf{\Pi}$  = Matrix with elements  $\pi_{ij}$

Consider the double summation:

$$\sum_{i=1}^n \sum_{j=1}^n 2\pi_{ij}(\nabla \ell_i^\top \mathbf{w} - \nabla \ell_j^\top \bar{\mathbf{w}})\nabla \ell_i$$

For each  $i$ , the term  $\nabla \ell_i^\top \mathbf{w}$  projects vector  $\mathbf{w}$  onto  $\nabla \ell_i$ . To compute this for every  $i$  simultaneously, it is  $\mathbf{G}\mathbf{w}$ . This results in an  $n \times 1$  column vector. Similarly, for each  $j$ , it is  $\mathbf{G}\bar{\mathbf{w}}$ . This also produces an  $n \times 1$  column vector.

The difference between these two projections for each  $i$  and  $j$  is  $\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}}$ . This results in an  $n \times 1$  column vector. To incorporate the  $\pi_{ij}$  weights, we have

$$\mathbf{\Pi}(\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}})$$

This operation gives an  $n \times 1$  column vector, where each element is a summation over  $j$  for the term  $\pi_{ij}(\nabla \ell_i^\top \mathbf{w} - \nabla \ell_j^\top \bar{\mathbf{w}})$ .

To finalize the summation, we multiply it by  $\mathbf{G}^\top$ , yielding

$$\mathbf{G}^\top (\mathbf{\Pi}(\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}})). \tag{18}$$

Combining (16), (17), and (18), we get:

$$\begin{aligned} \nabla Q(\mathbf{w}) &= 2\lambda(\mathbf{w} - \bar{\mathbf{w}}) + 2\mathbf{G}^\top (\mathbf{\Pi}(\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}})) \\ &= 2[\mathbf{G}^\top (\mathbf{\Pi}(\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}})) + \lambda(\mathbf{w} - \bar{\mathbf{w}})] \end{aligned} \tag{19}$$

#### A.5 EXPERIMENT SETUP

The models MLPNet, ResNet20, and MobileNetV1 underwent a pre-training phase of 100 epochs utilizing 4 NVIDIA Tesla V100 32 GB GPUs connected with NVlink. The training times were approximately 1 hour for MLPNet, 3 hours for ResNet20, and 1 day for MobileNetV1. Pre-pruning accuracy levels for these models are detailed under their respective names in Table 1. For the pruning process, we either utilized 2 NVIDIA Tesla V100 32 GB GPUs with NVlink or a single Tesla A100 PCIE (available in 40 or 80 GB configurations). It’s worth emphasizing the time-intensive nature of training and pruning the MobileNetV1 on ImageNet; thus, harnessing multiple GPUs for concurrent training is highly recommended.

In Table 1, we set the pruning stage of LR and EWR to be 15 for MLPNet and ResNet20 and 10 for MobileNetV1. The sparsity  $k_1, k_2, \dots, k_T$  in Algorithm 1 is arranged following an exponential gradual pruning schedule

$$k_t = k_T + (k_0 - k_T) \left(1 - \frac{t}{T}\right)^3$$

with the initial sparsity  $k_0$  set to zero. The fisher sample size setup follows (Chen et al., 2022, Table 2), shown as Table 4 of this paper below.

Table 4: Comparison of Sample and Batch Sizes for Different Models

Model	MLPNet		ResNet20/50		MobileNet	
	Sample	Batch	Sample	Batch	Sample	Batch
WF & CBS	1000	1	1000	1	400	2400
LR & EWR	1000	1	1000	1	1000	16

In Table 2, Table 3, Table 6, Table 7, and additional results provided in the Appendix, sparsity is set using a linear gradual pruning strategy, progressing from 0 to 0.75 or 0.95 across ten distinct stages for MLPNet and ResNet, and from 0 to 0.75 across eight distinct stages for MobileNetV1. The values are computed with linear incremental steps, from zero to the target sparsity. Notably, all recorded loss values are captured immediately post-pruning, devoid of any subsequent fine-tuning.

This approach ensures that the loss values exclusively reflect the impact of the pruning algorithms, without being clouded by external factors. Contrasting with Table 1, where each row represents a full pruning cycle, the loss values here are recorded across the ten incremental pruning stages. The empirical fisher is computed based on 100 samples with a batch size of 1.

In calibrating noise for data in neural networks, we start with a well-trained network. First, we calculate the standard deviation  $\sigma$  of the network’s derivative. Then, we add Gaussian noise with zero mean to the data. After adding the noise, the standard deviation of the network’s derivative changes to a new value,  $\sigma'$ , which is always greater than sigma. The goal is to adjust the standard deviation of the Gaussian noise so that  $\sigma'$  becomes  $\sigma' = \sigma + \sigma$  (referred to as noise level being  $\sigma$ ) or  $\sigma' = \sigma + 2\sigma$  (referred to as noise level being  $2\sigma$ ).

Throughout the paper, we set  $\lambda$  in the optimization problem (6) to 0.01. The regularization multiplier  $\varepsilon$  is set to 1 unless specified otherwise. The noise level  $\sigma$  is set to be the standard deviation of the original gradients.

#### A.6 ABLATION STUDY

Table 5: Comparison of loss values in terms of  $\varepsilon$  for ResNet20. The results are obtained from 25 runs, with 10% Noisy data and noise level  $\sigma$ . The target sparsity is 0.95.

Sparsity	LR Loss	EWR Loss				
		$\varepsilon = 0$	$\varepsilon = 1$	$\varepsilon = 2$	$\varepsilon = 10$	$\varepsilon = \infty$
0.95	2.89	<b>2.77</b>	2.78	2.78	2.79	<b>2.97</b>
0.84	1.76	1.69	<b>1.61</b>	1.68	1.71	<b>2.62</b>
0.74	0.68	0.68	0.67	<b>0.66</b>	<b>0.66</b>	<b>0.81</b>
0.63	0.36	0.36	0.36	0.36	0.36	0.37
0.53	0.31	0.31	0.31	0.31	0.31	0.31
0.42	0.31	0.31	0.31	0.31	0.31	0.31
0.32	0.30	0.30	0.30	0.30	0.30	0.30
0.21	0.30	0.30	0.30	0.30	0.30	0.30
0.11	0.31	0.31	0.31	0.31	0.31	0.31

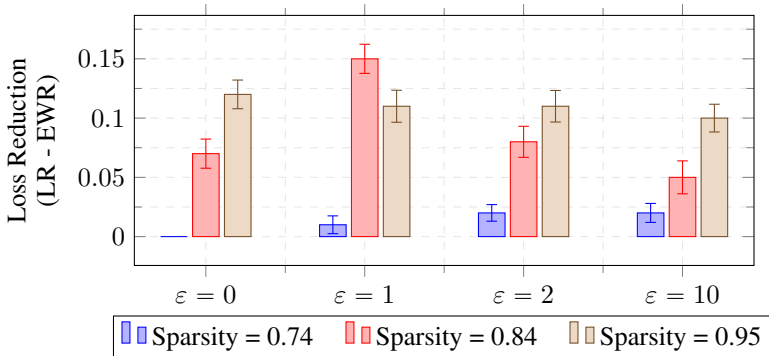


Figure 2: Loss reduction with different  $\varepsilon$ . The result is averaged over 25 runs for ResNet20, with 10% Noisy data and noise level  $\sigma$ . The error bar shows 90% confidence interval. The target sparsity is 0.95.

The ablation study centered on ResNet20 gives insight into the influence of the entropic regularization multiplier,  $\varepsilon$ , on pruning. The aim is to understand how different values of this parameter affect the loss  $\mathcal{L}(\mathbf{w})$  during the pruning process.

**Observations from Loss Values.** From Table 5, we can glean the following points. For higher sparsity levels (0.95, 0.84, 0.74), EWR Loss consistently outperforms the LR Loss, except for  $\varepsilon =$



$\infty$ . The optimum performance of the EWR Loss, across varying sparsities, tends to occur when  $\varepsilon$  is set at values between 1 and 2. As  $\varepsilon$  tends towards infinity, the EWR loss exceeds the LR loss. This phenomenon aligns with the neighborhood interpolation elaborated upon in Section 3. Specifically, incorporating an excessive number of distant data points into the interpolation detrimentally impacts performance. For lower sparsity levels (from 0.63 downwards), the differences between LR Loss and EWR Loss across different  $\varepsilon$  values are minuscule.

**Loss Reduction Insights.** Referencing Figure 2, it’s evident that the loss reduction (difference between LR and EWR) is more pronounced at higher sparsity levels. For a sparsity of 0.84,  $\varepsilon = 1$  demonstrates the most significant loss reduction. The trend of EWR loss for different  $\varepsilon$  values is consistent across varying sparsity levels. The impact  $\varepsilon$  is clearly visible at higher sparsity levels. For mid to high sparsity levels, lower values of  $\varepsilon$  (specifically 1 and 2) seem to achieve the best balance in terms of loss.

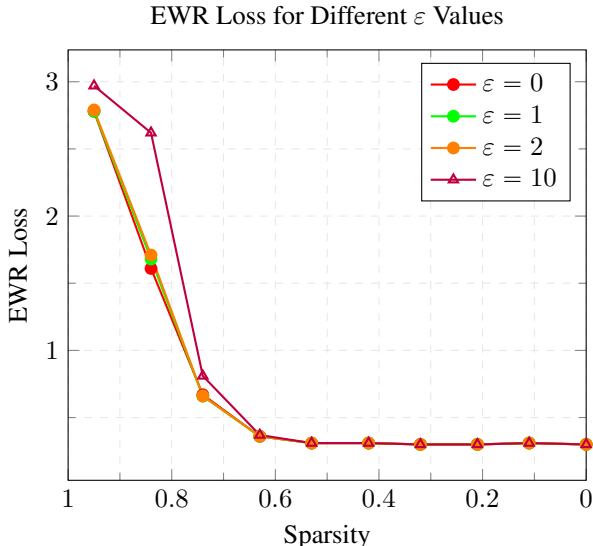


Figure 3: EWR Loss in function of the sparsity. The result is obtained over 25 runs on ResNet20, with 10% Noisy data and noise level  $\sigma$ . The target sparsity is 0.95.

Figure 3 illustrates EWR Loss for various  $\varepsilon$  values against sparsity, there is a clear trend of decreasing loss as sparsity decreases, consistent across all  $\varepsilon$  values. Particularly at  $\varepsilon = \infty$ , there’s a pronounced increase in loss at higher sparsity, suggesting that extreme entropic regularization might hinder optimal pruning. However, at low sparsity levels, the loss is consistent across all  $\varepsilon$ , emphasizing the minimal impact of  $\varepsilon$  in limited pruning scenarios. This underscores the significance of choosing an appropriate  $\varepsilon$ , balancing between regularization and pruning efficiency.

The ablation study provides insights into the role of the parameter  $\varepsilon$  in pruning. Its influence diminishes at low sparsity levels but becomes significant at extremely high sparsity. For ResNet20, an optimal range for  $\varepsilon$  appears to be between 1 and 2, ensuring effective pruning. Generally, while the exact choice of  $\varepsilon$  doesn’t drastically alter the pruning outcome, exceedingly high values, such as  $1 \times 10^8$ , might lead to less than ideal pruning decisions in very sparse networks.

### A.7 ANALYSIS OF PRUNING STAGE VERSUS PERFORMANCE

**Analysis of Loss.** Figure 4 depicts the relationship between the sparsity and the difference in loss (LR - EWR) under the influence of varying noise levels.

The left plot, representing data with a 10% noise level, depicts a prominent decrease in the difference of loss as sparsity is reduced for both noise levels  $2\sigma$  and  $\sigma$ . Notably, in the  $2\sigma$  noise scenario, there is a sharp decline in loss difference when sparsity transitions from 0.95 to 0.53. Beyond this threshold, the loss difference stabilizes and remains near zero. For the noise level  $\sigma$ , the decrease

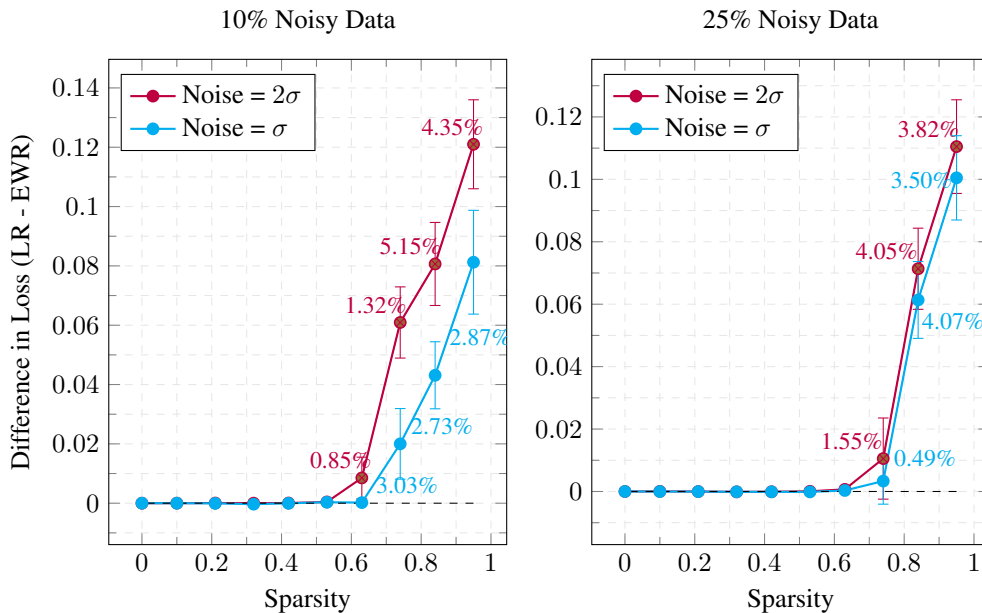


Figure 4: Difference in loss between LR and EWR for ResNet20. The data is from Table 2. The relative loss improvement of EWR over LR is reported. The target sparsity is 0.95.

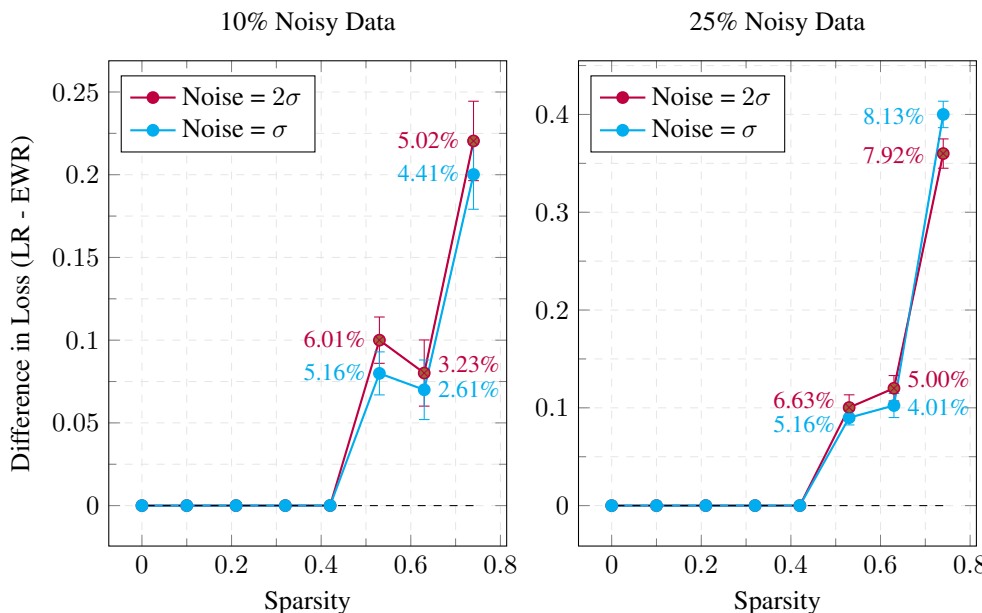


Figure 5: Difference in loss between LR and EWR for MobileNetV1. The data is from Table 3. The relative loss improvement of EWR over LR is reported. The target sparsity is 0.75.

appears more gradual. It is of interest to observe that the loss difference diminishes more swiftly for  $2\sigma$  compared to  $\sigma$ . The error bars offer insights into data variability, showcasing broader intervals at elevated sparsity levels, which suggests greater unpredictability at these levels, particularly in the  $2\sigma$  setting.

The right plot represents data with 25% noise. While the trends in loss difference share similarities with the 10% noisy data, the exact values differ slightly. In this 25% noise setting, the decline in loss difference between the two noise levels is similar. The error bars, indicating confidence intervals,

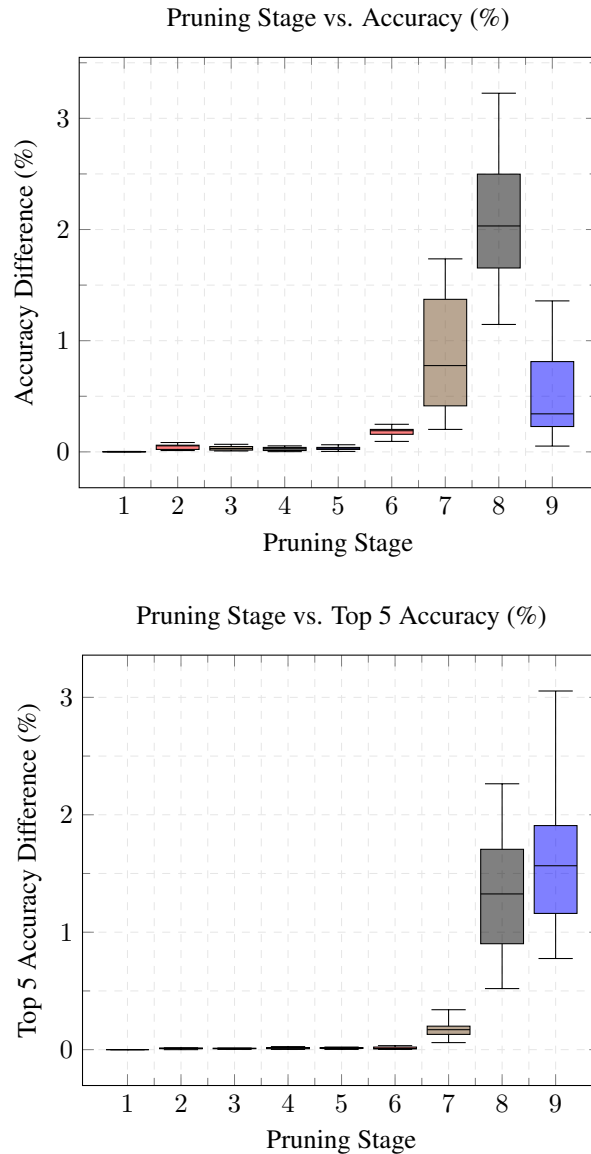


Figure 6: Accuracy difference vs. pruning stage for ResNet20. The difference is defined to be  $(\text{EWR} - \text{LR}) / \text{LR}$ . The data is obtained over 5-25 runs over combinations of different setups: noise level  $\sigma$  and  $2\sigma$ , noisy gradient proportion 10% and 25%,  $\varepsilon = 1, 2$ . The target sparsity is 0.95.

highlight the increased variability at larger sparsity levels. This variability is most noticeable for the  $2\sigma$  setting at the top sparsity levels.

Figure 5 depicts the difference in loss between LR and EWR algorithms applied to the MobileNetV1. In both two cases 10% and 25% noisy data, as sparsity increases, the loss difference diminishes, particularly when sparsity is approximately 0.42 or less. The 10% noisy data reveals that the difference in loss for noise level  $2\sigma$  is marginally higher than that of  $\sigma$  for most sparsity levels. In contrast, the 25% noisy data sometimes exhibits a reversal in this trend, especially at the highest sparsity level of 0.74.

Confidence intervals provided at select data points underscore the reliability of the data, with the 25% noisy data showing tighter intervals compared to the 10% scenario. This infers a higher consistency in the measurements or a minimized effect of outliers in the 25% noisy data. These plots accentuate the interplay between sparsity, noise, and the performance difference between the two al-

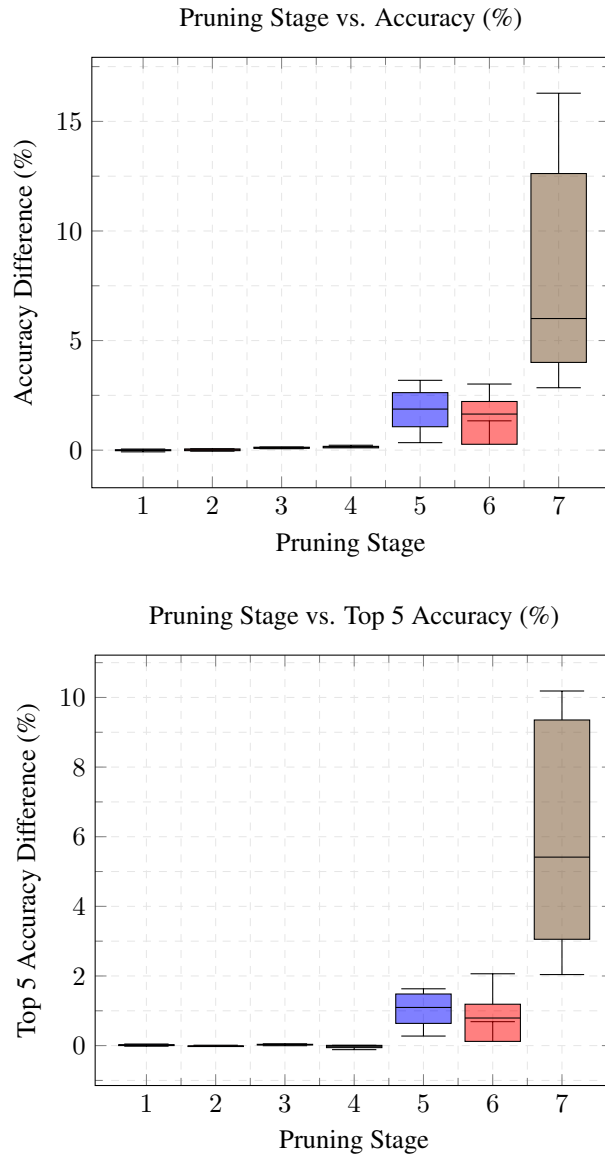


Figure 7: Accuracy difference vs. pruning stage for MobileNetV1. The difference is defined to be  $(\text{EWR} - \text{LR}) / \text{LR}$ . The data is obtained over 5-10 runs over combinations of different setups: noise level  $\sigma$  and  $2\sigma$ , noisy gradient proportion 10% and 25%,  $\varepsilon = 1, 2$ . The target sparsity is 0.95.

gorithms, emphasizing the significance of noise levels in algorithmic performance evaluations across various sparsity conditions. The underlying rationale is that as the noise intensity is too high, the performance of both LR and EWR tends to deteriorate. Consequently, their performances converge, resulting in a diminished differential between the two.

**Analysis of Accuracy.** The box plots in Figure 6 together with Table 6 show how the ResNet20 model performs at different pruning stages in terms of top-1 accuracy (also referred to as accuracy or the overall accuracy) and top-5 accuracy difference. Top-5 accuracy means any of our model’s top 5 highest probability answers match with the expected answer. The difference is computed for EWR, using LR as the baseline. When we look closely, we can see patterns that help us understand how much pruning affects the model.

In the earlier pruning stages, both the overall accuracy and the top 5 accuracy differences for EWR are minimal, suggesting that EWR remains closely aligned with the baseline LR in terms of per-

Table 6: Comparison of testing Top-1 accuracy (%) (without the fine-tuning steps imposed) values for LR and EWR for ResNet20. The result is averaged over 25 runs. The 90% confidence interval is reported. The target sparsity is set to be 0.95.

Sparsity	10% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.95	11.65 ( $\pm 0.21$ )	11.70 ( $\pm 0.15$ )	0.47%	11.53 ( $\pm 0.14$ )	<b>11.72</b> ( $\pm 0.14$ )	<b>1.70%</b>
0.84	57.01 ( $\pm 0.10$ )	<b>58.17</b> ( $\pm 0.10$ )	<b>2.03%</b>	56.03 ( $\pm 0.09$ )	<b>57.50</b> ( $\pm 0.09$ )	<b>2.63%</b>
0.74	80.53 ( $\pm 0.05$ )	<b>81.12</b> ( $\pm 0.07$ )	<b>0.73%</b>	80.23 ( $\pm 0.05$ )	<b>81.06</b> ( $\pm 0.06$ )	<b>1.03%</b>
0.63	89.85 ( $\pm 0.06$ )	89.90 ( $\pm 0.06$ )	0.00%	88.95 ( $\pm 0.06$ )	88.96 ( $\pm 0.05$ )	0.85%
Sparsity	25% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.95	11.63 ( $\pm 0.17$ )	11.98 ( $\pm 0.15$ )	<b>3.03%</b>	11.68 ( $\pm 0.18$ )	<b>12.04</b> ( $\pm 0.16$ )	<b>3.04%</b>
0.84	56.90 ( $\pm 0.09$ )	<b>58.90</b> ( $\pm 0.09$ )	<b>3.52%</b>	57.90 ( $\pm 0.07$ )	<b>59.76</b> ( $\pm 0.06$ )	<b>3.22%</b>
0.74	80.37 ( $\pm 0.05$ )	<b>80.76</b> ( $\pm 0.10$ )	<b>0.49%</b>	81.50 ( $\pm 0.08$ )	<b>82.27</b> ( $\pm 0.05$ )	<b>0.95%</b>
0.63	89.62 ( $\pm 0.05$ )	89.62 ( $\pm 0.09$ )	0.00%	90.56 ( $\pm 0.06$ )	90.56 ( $\pm 0.05$ )	0.00%

Table 7: Comparison of testing Top-1 accuracy (%) (without the fine-tuning step imposed) values for LR and EWR for MobileNetV1. The result is averaged from 10 runs. The 90% confidence interval is reported. The target sparsity is set to be 0.75.

Sparsity	10% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.75	17.10 ( $\pm 0.20$ )	<b>17.79</b> ( $\pm 0.15$ )	<b>4.02%</b>	17.05 ( $\pm 0.21$ )	<b>17.84</b> ( $\pm 0.18$ )	<b>4.62%</b>
0.63	43.92 ( $\pm 0.05$ )	<b>44.92</b> ( $\pm 0.05$ )	<b>2.27%</b>	43.27 ( $\pm 0.10$ )	<b>44.53</b> ( $\pm 0.10$ )	<b>2.92%</b>
0.53	61.06 ( $\pm 0.04$ )	<b>61.84</b> ( $\pm 0.05$ )	<b>1.28%</b>	60.51 ( $\pm 0.05$ )	<b>63.90</b> ( $\pm 0.05$ )	<b>2.30%</b>
0.42	68.09 ( $\pm 0.05$ )	68.09 ( $\pm 0.06$ )	0.00%	67.24 ( $\pm 0.05$ )	67.24 ( $\pm 0.05$ )	0.00%
Sparsity	25% Noisy Data					
	Noise = $\sigma$			Noise = $2\sigma$		
	LR	EWR	Diff	LR	EWR	Diff
0.75	13.97 ( $\pm 0.14$ )	<b>14.88</b> ( $\pm 0.26$ )	<b>6.52%</b>	13.78 ( $\pm 0.21$ )	<b>14.78</b> ( $\pm 0.25$ )	<b>7.29%</b>
0.63	43.45 ( $\pm 0.10$ )	<b>44.78</b> ( $\pm 0.09$ )	<b>3.05%</b>	43.21 ( $\pm 0.10$ )	<b>45.13</b> ( $\pm 0.11$ )	<b>4.44%</b>
0.53	60.59 ( $\pm 0.05$ )	<b>61.26</b> ( $\pm 0.04$ )	<b>1.10%</b>	60.07 ( $\pm 0.05$ )	<b>61.56</b> ( $\pm 0.05$ )	<b>2.48%</b>
0.42	67.33 ( $\pm 0.04$ )	67.46 ( $\pm 0.05$ )	0.20%	66.92 ( $\pm 0.06$ )	66.92 ( $\pm 0.05$ )	0.00%

formance. This minimal deviation can be viewed as an advantage, as it implies that even with simplifications brought by pruning, EWR retains its effectiveness compared to LR.

However, as pruning intensifies, the patterns begin to reveal more about EWR’s relative strengths. Although the accuracy difference increases, this increase in the context of the baseline suggests that EWR might be better at handling intense pruning than LR. Notably, in the top 5 accuracy, the discrepancies remain relatively low compared to the overall accuracy up until the more aggressive pruning stages. This suggests that while the model’s primary prediction confidence may decrease, the true class is still frequently among its top 5 predictions. In essence, EWR seems to retain a broader spectrum of potential correct classifications even when it’s uncertain about the primary prediction.

Figure 7 together with Table 7 shows the performance difference between EWR and LR using MobileNetV1 across various pruning stages. In the first plot showcasing the overall accuracy, there’s an evident upward trend in the median accuracy difference as pruning intensifies. Initially, the difference is marginal, which suggests that EWR’s performance closely mirrors LR during the early

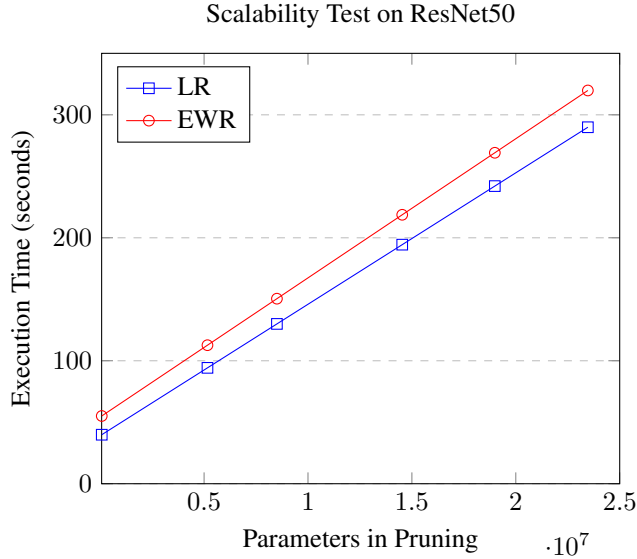


Figure 8: Scalability of Algorithm 1. This plot shows the comparison between LR and EWR of the single-round execution time of Lines 4–11. The test is performed on an NVIDIA Tesla V100 32GB GPU. The fisher sample size is set to 1000. Layers of the networks are gradually added for pruning (hence the number of model parameters  $p$  increases). The OT planning  $\Pi$  is solved using the Sinkhorn-Knopp method shown in Algorithm 2.

pruning stages. However, as we progress to the 5th and 6th stages, the accuracy difference widens considerably, indicating that EWR might be outpacing LR. The 7th stage is particularly striking, with a median accuracy difference surpassing 6%, pointing towards a potential superiority of EWR in extreme pruning scenarios.

The second plot focuses on the top 5 accuracy differences, presenting a more varied pattern. The early stages indicate a tight performance race between EWR and LR. By the 2nd stage, there’s a minor dip, hinting at EWR’s possible underperformance. This scenario changes by the 3rd stage as EWR regains momentum. The later stages, especially the 5th and 6th, denote a significant rise in the median difference in EWR’s favor. Much like the accuracy chart, the 7th stage is distinct, with EWR showcasing a considerable advantage in the top 5 accuracy over LR.

#### A.8 ALGORITHM SCALABILITY

In this section, we analyze the scalability of Algorithm 1 with respect to the number of model parameters involved in pruning. The result is shown in Figure 8. It can be observed that the execution time scales linearly with the number of pruning parameters. The extra cost of solving the OT is marginal. Theoretically, one could derive this linear scalability by inspecting Line 9, which is the most time-consuming step. The required operations can be decomposed as sequential operations: matrix-vector multiplications  $\mathbf{G}\mathbf{w}$  and  $\mathbf{G}\bar{\mathbf{w}}$  in  $O(np)$ , the vector subtraction  $\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}}$  in  $O(n)$ , a matrix-matrix multiplication  $\mathbf{\Pi}(\mathbf{G}\mathbf{w} - \mathbf{G}\bar{\mathbf{w}})$  in  $O(n^2)$ , a matrix transposition and multiplication with  $\mathbf{G}$  in  $O(np)$ , and the vector subtraction and scalar multiplication  $\lambda(\mathbf{w} - \bar{\mathbf{w}})$  in  $O(p)$ . Thus, the overall complexity is  $O(np)$ , with  $p$  significantly larger than  $n$  practically. Given fixed fisher sample size  $n$ , the loop of Algorithm 1 scales linearly with the number of pruning parameters  $p$ .

#### A.9 OPTIMAL TRANSPORT VISUALIZATION

In this section, we showcase the optimized OT plan denoted as  $\Pi$ . This was derived from the pruning applied to ResNet20. Figure 9 displays two data sets: 1)  $\{x_i\}_{i=1}^n$ : This is  $\mathbf{G}\mathbf{w}$  where  $\mathbf{w}$  is the pruned model. 2)  $\{y_i\}_{i=1}^n$ : This represents  $\mathbf{G}\bar{\mathbf{w}}$  where  $\bar{\mathbf{w}}$  is the original unpruned model.

In Figure 10, the matrix  $\Pi$  is shown through vibrant heatmaps that adjust with varying  $\varepsilon$  values. For small  $\varepsilon$ , the majority of the data remains near a diagonal. As  $\varepsilon$  increases, there's a broader data distribution, notably for central data points.

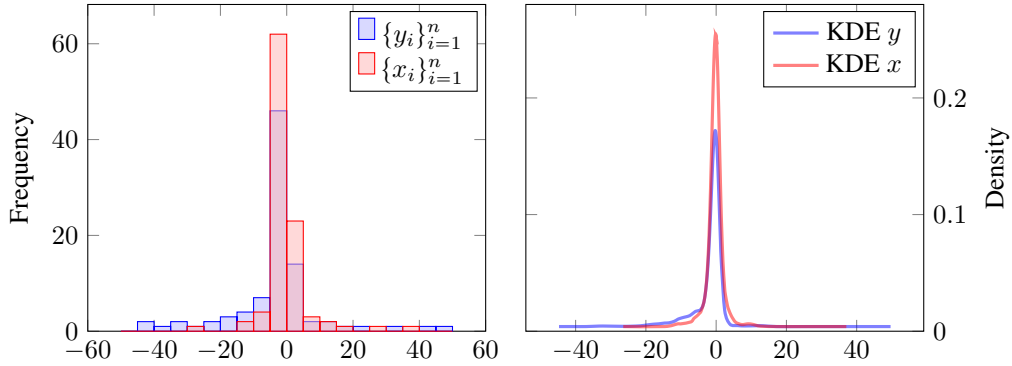


Figure 9: The histogram and kde plot of the empirical distribution  $\{x_i\}_{i=1}^n$  and  $\{y_i\}_{i=1}^n$ .

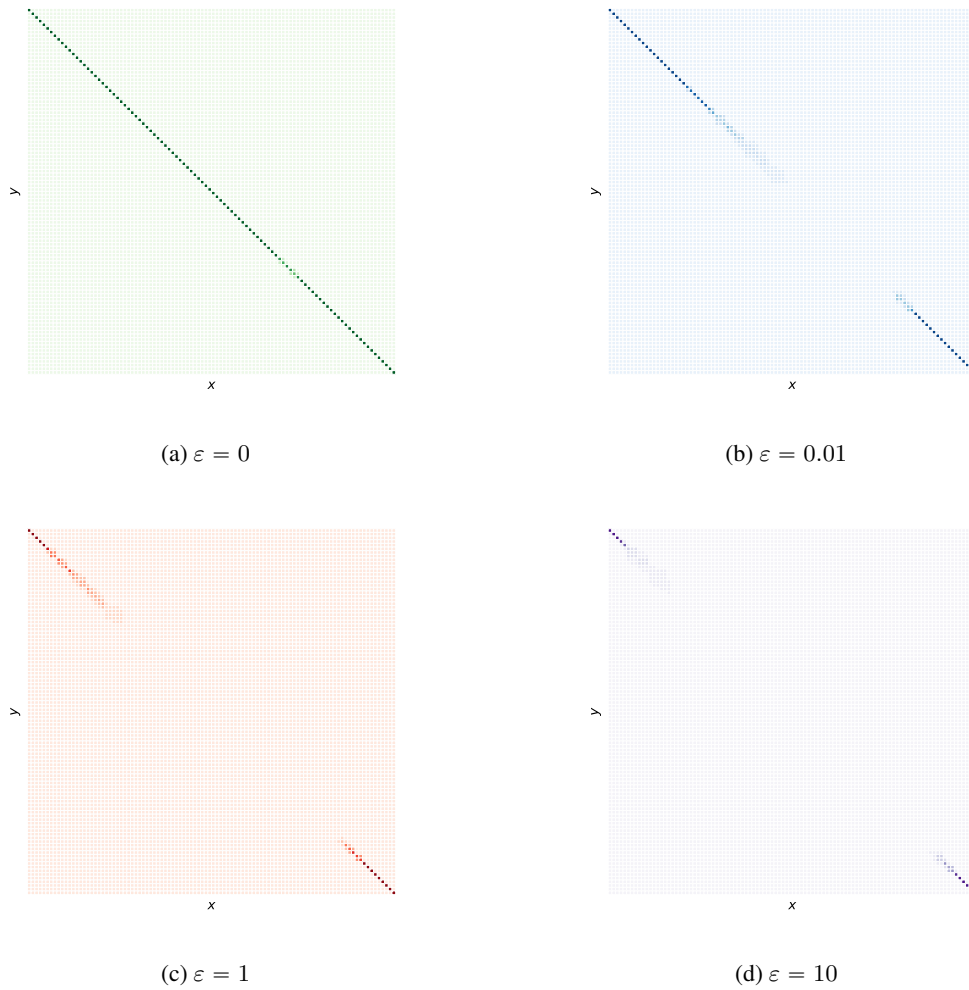


Figure 10: Heatmap of the optimized OT plan  $\Pi$  across different  $\varepsilon$ . Darker color indicate larger value in  $\Pi$ .