Fast Convex Pruning of Deep Neural Networks

Alireza Aghasi* Afshin Abdi† Justin Romberg†

Abstract

We develop a fast, tractable technique called Net-Trim for simplifying a trained neural network. The method is a convex post-processing module, which prunes (sparsifies) a trained network layer by layer, while preserving the internal responses. We present a comprehensive analysis of Net-Trim from both the algorithmic and sample complexity standpoints, centered on a fast, scalable convex optimization program. Our analysis includes consistency results between the initial and retrained models before and after Net-Trim application and guarantees on the number of training samples needed to discover a network that can be expressed using a certain number of nonzero terms. Specifically, if there is a set of weights that uses at most s terms that can re-create the layer outputs from the layer inputs, we can find these weights from $\mathcal{O}(s \log N/s)$ samples, where N is the input size. These theoretical results are similar to those for sparse regression using the Lasso, and our analysis uses some of the same recently-developed tools (namely recent results on the concentration of measure and convex analysis). Finally, we propose an algorithmic framework based on the alternating direction method of multipliers (ADMM), which allows a fast and simple implementation of Net-Trim for network pruning and compression.

Keywords: Pruning Neural Networks, Deep Neural Networks, Compressed Sensing, Bowling Scheme, Rademacher Complexity

1 Introduction

Deep neural networks are becoming a prominent tool to learn data structures of arbitrary complexity. This success is mainly thanks to their flexible, yet compact nonlinear formulation, and the development of computational and architectural techniques to improve their training (c.f. [Sch15, GBC16] for a comprehensive review). Increasing the number of layers, and the number of neurons within each layer is generally the most standard way of adding more flexibility to a neural network. While adding such flexibility is capable of improving the fitting of the model to the training data (i.e., reducing the model bias), it makes the models prone to over-parameterization and overfitting (i.e., increasing the model variance), which in turn can degrade the predictive capability of the network.

To simplify or stabilize neural networks, various regularizing techniques and pruning strategies have been considered. Inspired by the classic regularizers for linear models, such as Ridge ([HK70]) and Lasso ([Tib96]), the training of neural networks is also equipped with ℓ_2 or ℓ_1 penalties ([NH92, GJP95]) to control their variance and complexity. Adding randomness to the training process is also shown to have regularizing effects, relevant to which we may refer to Dropout ([SHK+14]) and DropConnect ([WZZ+16]), which randomly remove active connections in the training phase and are likely to produce pruned networks. Batch normalization ([IS15]), associated with stochastic gradient descent-type fitting techniques, can also be considered as a tool of similar nature, where in the training process the updates of the hidden units are weighted by the standard deviation of the random examples included in the mini-batch.

In this paper, we advocate a different approach. We train the network using standard techniques. We then extract the internal outputs (the intermediate features) at each layer and find a sparse set of weights that reproduces these features across all the training data. The philosophy here is that the most important product of

*(Corresponding Author) Robinson College of Business, Georgia State University, Atlanta, GA. Email: aaghasi@gsu.edu
†School of Electrical and Computer Engineering, Georgia Tech, Atlanta, GA. Emails: {abdi,jrom}@ece.gatech.edu.

training the network is the features that it extracts, not the weights that it settles on to produce those features. For large networks, there will be many sets of weights that produce exactly the same internal features; of those weights, we choose the simplest.

Our method for finding sparse sets of weights, presented in detail in Section 3, is related to well-known techniques for sparse regression, e.g. the Lasso ([Tib96]) in statistics and compressed sensing ([CRT06]) in signal processing. The main difference is the non-linearity in the mapping of internal features from one layer to another. If this non-linearity is piecewise linear and convex (as is the rectified linear unit, $\texttt{ReLU}(\boldsymbol{x}) = \max(\boldsymbol{x}, \boldsymbol{0})$, that we use in all of our analysis below), then there is a natural way to recast the condition that the outputs and inputs of a layer match as a set of linear inequality constraints. There is a similar way to recast an approximate matching as inclusion in a convex set. Using the ℓ_1 norm as a proxy for sparsity, the entire program becomes convex. This opens the door for a thorough analysis of how well and under what conditions we can expect Net-Trim to perform well, and allows us to leverage decades of research in convex optimization to find a scalable algorithm with predictable convergence behavior.

The theory in Section 4 presents an upper bound on the number of training samples needed to discover a weight matrix that is sparse. Given a set of layer input vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P$ and output vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_P$, we solve the program

$$\underset{\boldsymbol{W}}{\text{minimize}} \ \|\boldsymbol{W}\|_1 \quad \text{subject to} \quad \texttt{ReLU}(\boldsymbol{W}^\top \boldsymbol{x}_p) = \boldsymbol{y}_p, \quad (p = 1, \ldots, P), \tag{1}$$

where $\|\boldsymbol{W}\|_1 = \sum_{n,m} |w_{n,m}|$ is the sum of the absolute values of the entries in a matrix $\boldsymbol{W} \in \mathbb{R}^{N \times M}$. As the ℓ_1 norm is convex and the $\texttt{ReLU}(\cdot)$ function is piecewise linear, meaning that constraints in the program above can be broken into a series of linear equality and inequality constraints, the program above is convex. We show that if the $\boldsymbol{x}_p$ are independent samples of a subgaussian random vector that is non-degenerate (meaning that the correlation matrix is full-rank) and there exists a $\boldsymbol{W}_\star$ with maximally s-sparse columns that does indeed satisfy $\boldsymbol{y}_p = \texttt{ReLU}(\boldsymbol{W}_\star^\top \boldsymbol{x}_p)$ for all p, then the solution to (1) is exactly $\boldsymbol{W}_\star$ when the number of training samples P is (almost) proportional to the sparsity s: we require

$$P \ \gtrsim \ s \log(N/s).$$

We also show that if the $\boldsymbol{x}_p$ are subgaussian, then so are the $\boldsymbol{y}_p$. As a results, the theory can be applied layer-by-layer, yielding a sampling result for networks of arbitrary depth. (When we apply the algorithm in practice, the equality constraints in (1) are relaxed; this is discussed in detail in Section 3.1.)

Along with these theoretical guarantees, Net-Trim offers state-of-the-art performance on realistic networks. In Section 6, we present numerical experiments that show that compression factors between 10x and 50x (removing 90% to 98% of the connections) are possible with very little loss in test accuracy. In fact, for small compression factors (5x–10x), the test accuracy is actually higher than for the original network. The compression factor (sparsity of the connections), which can be adjusted in a straightforward manner in Net-Trim, acts as a trade-off between the model bias and variance; these results suggest that trimming the weights rebalances these terms in a favorable way by reducing the variance while not comparable increasing the bias.

Contributions and relations to previous work. This paper provides a full description of the Net-Trim method from both a theoretical and algorithmic perspective. In Section 3, we present our convex formulation for sparsifying the weights in the linear layers of a network; we describe how the procedure can be applied layer-by-layer in a deep network either in parallel or serially (cascading the results), and present consistency bounds for both approaches. Section 4 presents our main theoretical result, stated precisely in Theorem 4. This result derives an upper bound on the number of data samples we need to reliably discover a layer that has at

most s connections in its linear layer — we show that if the data samples are random, then these weights can be learned from $\mathcal{O}(s \log N/s)$ samples. Mathematically, this result is comparable to the sample complexity bounds for the Lasso in performing sparse regression on a linear model (also known as the compressed sensing problem), and our analysis draws on a similar mix of tools from optimization and probability theory.

There are several other examples of techniques for simplifying networks by re-training in the recent literature. These techniques are typically presented as model compression tools (e.g., [HPTD15, CWT+15, HMD15]) for removing the inherent model redundancies. In what is perhaps the most closely related work to what we present here, [HPTD15] proposes a pruning scheme that simply truncates small weights of an already trained network, and then re-adjusts the remaining active weights using another round of training. In contrast, our optimization scheme ensures that the layer inputs and outputs stay consistent as the network is pruned.

The Net-Trim framework was first presented in ([AANR17]). This paper provides a far more rigorous and complete analysis (sample complexity bound) of the Net-Trim algorithm for networks with multiple layers (the previous work only considered a single layer of the network). In addition, we present a scalable (yet relatively simple) implementation of Net-Trim using the alternation direction method of multipliers (ADMM). This is an iterative method with each iteration requiring a small number of matrix-vector multiplies. The code, along with all the examples presented in the paper, is available online[1].

Notation. We use lowercase and uppercase boldface for vectors and matrices, respectively. Specifically, the notation $\boldsymbol{I}$ is reserved for the identity matrix. For a matrix $\boldsymbol{A}$, $\boldsymbol{A}_{\Gamma_1,:}$ denotes the submatrix formed by restricting the rows of $\boldsymbol{A}$ to the index set Γ_1. Similarly, $\boldsymbol{A}_{:,\Gamma_2}$ restricts the columns of $\boldsymbol{A}$ to Γ_2, and $\boldsymbol{A}_{\Gamma_1,\Gamma_2}$ is formed by extracting both rows and columns. Given a vector $\boldsymbol{x}$ (or matrix $\boldsymbol{X}$), supp $\boldsymbol{x}$ (or supp $\boldsymbol{X}$) is the set of indices with non-zero entries, and supp^c $\boldsymbol{x}$ (or supp^c $\boldsymbol{X}$) is the complement set.

For $\boldsymbol{X} = [x_{m,n}] \in \mathbb{R}^{M\times N}$, the matrix trace is denoted by $\mathrm{tr}(\boldsymbol{X})$. Furthermore, we use $\|\boldsymbol{X}\|_1 \triangleq \sum_{m=1}^{M}\sum_{n=1}^{N}|x_{m,n}|$ as a notation for the sum of absolute entries[2], and $\|\boldsymbol{X}\|_F$ as the Frobenius norm. The neural network activation used throughout the paper is the rectified linear unit (ReLU), which is applied component-wise to vectors and matrices,

$$\left(\texttt{ReLU}\left(\boldsymbol{X}\right)\right)_{m,n} = \max\left(x_{m,n}, 0\right).$$

We will sometimes use the notation $\boldsymbol{X}^+$ as shorthand for $\texttt{ReLU}(\boldsymbol{X})$. For an index set $\Omega \subseteq \{1,\cdots,M\}\times\{1,\cdots,N\}$, $\boldsymbol{W}_\Omega$ represents a matrix of identical size as $\boldsymbol{W} = [w_{m,n}]$ with entries

$$\left(\boldsymbol{W}_\Omega\right)_{m,n} = \begin{cases} w_{m,n} & (m,n)\in\Omega \\ 0 & (m,n)\notin\Omega \end{cases}.$$

Finally, we use $\mathbb{S}^N$ to denotes the unit sphere in $\mathbb{R}^{N+1}$; and the notation $f \gtrsim \hat{f}$ (or $f \lesssim \hat{f}$) when there exists an absolute constant C such that $f \geq C\hat{f}$ (or $f \leq C\hat{f}$).

Outline. The remainder of the paper is structured as follows. In Section 2, we briefly overview the neural network architecture considered. Section 3 presents the pruning idea and the consistency results between the initial and retrained networks. The statistical architecture of the network and the general sample complexity results are presented in Section 4. To implement the Net-Trim underlying convex program, in Section 5 we present an ADMM scheme applicable to the original Net-Trim formulation. A collection of experiments and comparisons is presented in Section 6, which is followed by some concluding remarks in Section 7. All the technical proofs of the theorems and results presented in this paper are moved to Section 8.

[1] The link to the code and related material: `https://dnntoolbox.github.io/Net-Trim/`

[2] The notation $\|\boldsymbol{X}\|_1$ should not be confused with the matrix induced ℓ_1 norm

2 Feedforward Network Model

In this section, we briefly overview the topology of the feedforward network model considered. The training of the network is performed via P samples $\boldsymbol{x}_p$, $p = 1, \cdots, P$, where $\boldsymbol{x}_p \in \mathbb{R}^N$ is the network input. To compactly represent the training samples, we form a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times P}$, structured as $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_P]$. Considering L layers in the network, the output of the network at the final layer is denoted by $\boldsymbol{X}^{(L)} \in \mathbb{R}^{N_L \times P}$, where each column in $\boldsymbol{X}^{(L)}$ is a response to the corresponding training column in $\boldsymbol{X}$.

In a ReLU network, the output of the ℓ-th layer is $\boldsymbol{X}^{(\ell)} \in \mathbb{R}^{N_\ell \times P}$, generated by applying the affine transformation $\boldsymbol{W}_\ell^\top(\cdot) + \boldsymbol{b}^{(\ell)}$ to each column of the previous layer $\boldsymbol{X}^{(\ell-1)}$, followed by a ReLU activation:

$$\boldsymbol{X}^{(\ell)} = \texttt{ReLU}\left(\boldsymbol{W}_\ell^\top \boldsymbol{X}^{(\ell-1)} + \boldsymbol{b}^{(\ell)} \boldsymbol{1}^\top\right), \qquad \ell = 1, \cdots, L. \tag{2}$$

Here $\boldsymbol{W}_\ell \in \mathbb{R}^{N_{\ell-1} \times N_\ell}$, $\boldsymbol{X}^{(0)} = \boldsymbol{X}$ and $N_0 = N$. By adding an additional row to $\boldsymbol{W}_\ell$ and $\boldsymbol{X}^{(\ell-1)}$, one can absorb the intercept term and compactly rewrite (2) as

$$\boldsymbol{X}^{(\ell)} = \texttt{ReLU}\left(\boldsymbol{W}_\ell^\top \boldsymbol{X}^{(\ell-1)}\right), \qquad \ell = 1, \cdots, L. \tag{3}$$

Often the last layer of a neural network skips an activation by merely going through the affine transformation. As a matter of fact, the results presented in this paper also apply to such architecture (see analysis examples in [AANR17]). A neural network that follows the model in (3) can be fully identified by $\boldsymbol{X}$ and $\hat{\boldsymbol{W}}_\ell$, $\ell = 1, \cdots, L$. Throughout the paper, such network will be denoted by $\mathcal{N}et(\{\boldsymbol{W}_\ell\}_{\ell=1}^L; \boldsymbol{X})$.

3 The Net-Trim Pruning Algorithm

Net-Trim is a post processing scheme which prunes a neural network after the training phase. Similar to many other regularization techniques, Net-Trim is capable of simplifying trained models at the expense of a controllable increase in the bias.

After the training phase and learning $\boldsymbol{W}_\ell$, Net-Trim retrains the network so that for the same training data the layer outcomes stay more or less close to the initial model, while the redesigned network is sparser, i.e.,

$$\ell = 1, \cdots, L: \ \text{nnz}\left(\hat{\boldsymbol{W}}_\ell\right) \ll \text{nnz}\left(\boldsymbol{W}_\ell\right), \quad \text{while} \ \ \hat{\boldsymbol{X}}^{(\ell)} \approx \boldsymbol{X}^{(\ell)}.$$

Here, nnz(.) denotes the number of nonzero entries, and $\hat{\boldsymbol{W}}_\ell$ and $\hat{\boldsymbol{X}}^{(\ell)}$ are respectively the redesigned layer matrices and the corresponding layer outcomes.

Aside from the post-processing nature and some differences in the convex formulations, Net-Trim shares many similarities with the Lasso (least absolute shrinkage and selection operator, [Tib96]), as they both use an ℓ_1 proxy to promote model sparsity. In the remainder of this section we overview the Net-Trim formulation and the corresponding pruning schemes.

3.1 Pruning a Single Layer

Consider $\boldsymbol{X}^{in} \in \mathbb{R}^{N \times P}$ and $\boldsymbol{X}^{out} \in \mathbb{R}^{M \times P}$ to be a layer input and output matrices after the training, which based on the model in (2) (or (3)) are connected via

$$\boldsymbol{X}^{out} = \texttt{ReLU}\left(\boldsymbol{W}^\top \boldsymbol{X}^{in}\right).$$

To explore a sparser coefficient matrix, we may consider the minimization

$$\underset{\boldsymbol{U}}{\text{minimize}} \quad \|\boldsymbol{U}\|_1 \quad \text{subject to} \quad \left\| \texttt{ReLU}\left(\boldsymbol{U}^\top \boldsymbol{X}^{in}\right) - \boldsymbol{X}^{out} \right\|_F \leq \epsilon, \tag{4}$$

which may potentially generate a sparser $\boldsymbol{W}$-matrix relating $\boldsymbol{X}^{in}$ and $\boldsymbol{X}^{out}$, at the expense of a (controllable) discrepancy between the layer outcomes before and after the retraining.

Despite the convex objective, the constraint set in (4) is non-convex. Using the fact that the entries of $\boldsymbol{X}^{out}$ are either zero or strictly positive quantities, [AANR17] propose the following convex proxy to (4):

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{U}}{\arg\min} \ \|\boldsymbol{U}\|_1 \quad \text{subject to} \quad \begin{cases} \left\| \left(\boldsymbol{U}^\top \boldsymbol{X}^{in} - \boldsymbol{X}^{out}\right)_\Omega \right\|_F \leq \epsilon \\ \left(\boldsymbol{U}^\top \boldsymbol{X}^{in}\right)_{\Omega^c} \leq \boldsymbol{0} \end{cases}, \tag{5}$$

where

$$\Omega = \text{supp } \boldsymbol{X}^{out} = \left\{ (m,p) : \left[\boldsymbol{X}^{out}\right]_{m,p} > 0 \right\}.$$

The main idea behind this convex surrogate is imposing similar activation patterns before and after the retraining via the second inequality in (5), i.e.,

$$\left(\boldsymbol{W}^\top \boldsymbol{X}^{in}\right)_{\Omega^c}^+ = \left(\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in}\right)_{\Omega^c}^+ = \boldsymbol{0},$$

and allowing the ϵ-discrepancy only on the set Ω. For a more compact presentation of the convex constraint set, for given matrices $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{V}$ we use the notation

$$\boldsymbol{U} \in \mathcal{C}_\epsilon\left(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{V}\right) \iff \begin{cases} \left\| \left(\boldsymbol{U}^\top \boldsymbol{X} - \boldsymbol{Y}\right)_\Omega \right\|_F \leq \epsilon \\ \left(\boldsymbol{U}^\top \boldsymbol{X}\right)_{\Omega^c} \leq \boldsymbol{V}_{\Omega^c} \end{cases}, \text{ for } \Omega = \text{supp } \boldsymbol{Y}. \tag{6}$$

Using this notation, the convex program in (5) may be cast as

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{U}}{\arg\min} \ \|\boldsymbol{U}\|_1 \quad \text{subject to} \quad \boldsymbol{U} \in \mathcal{C}_\epsilon\left(\boldsymbol{X}^{in}, \boldsymbol{X}^{out}, \boldsymbol{0}\right). \tag{7}$$

3.2 Pruning the Network

Having access to the tools to retrain any layer within the network, exclusively based on the input and the output, we may consider *parallel* or *cascade* frameworks to retrain the entire network.

The parallel Net-Trim is a straightforward application of the convex program (7) to each layer in the network. Basically, each layer is processed independently based on the initial model input and output, without taking into account the retraining result from the previous layers. Specifically, denoting $\boldsymbol{X}^{(\ell-1)}$ and $\boldsymbol{X}^{(\ell)}$ as the input and output of the ℓ-th layer of the initial trained network, we propose to retrain the coefficient matrix $\boldsymbol{W}_\ell$ via the convex program

$$\hat{\boldsymbol{W}}_\ell = \underset{\boldsymbol{U}}{\arg\min} \ \|\boldsymbol{U}\|_1 \quad \text{subject to} \quad \boldsymbol{U} \in \mathcal{C}_{\epsilon_\ell}\left(\boldsymbol{X}^{(\ell-1)}, \boldsymbol{X}^{(\ell)}, \boldsymbol{0}\right), \quad \ell = 1, \cdots, L. \tag{8}$$

An immediate question would be if each layer of a network is retrained via (8) and one replaces $\mathcal{N}et(\{\boldsymbol{W}_\ell\}_{\ell=1}^L; \boldsymbol{X})$ with the retrained network $\mathcal{N}et(\{\hat{\boldsymbol{W}}_\ell\}_{\ell=1}^L; \boldsymbol{X})$, how do the discrepancies ϵ_ℓ propagate across the network, and how far apart would be the final responses of the two networks to $\boldsymbol{X}$? The following result addresses this question.

Theorem 1 (Parallel Net-Trim) *Consider a normalized network $\mathcal{N}et(\{\boldsymbol{W}_\ell\}_{\ell=1}^L;\boldsymbol{X})$, such that $\|\boldsymbol{W}_\ell\|_1=1$ for $\ell=1,\cdots,L$. Solve (8) for each layer and form the retrained network $\mathcal{N}et(\{\hat{\boldsymbol{W}}_\ell\}_{\ell=1}^L;\boldsymbol{X})$. Denoting the outcomes of the retrained network by $\hat{\boldsymbol{X}}^{(\ell)}=\texttt{ReLU}(\hat{\boldsymbol{W}}_\ell{}^\top\hat{\boldsymbol{X}}^{(\ell-1)})$ where $\hat{\boldsymbol{X}}^{(0)}=\boldsymbol{X}^{(0)}=\boldsymbol{X}$, the layer outcomes of the original and retrained networks will obey*

$$\left\|\hat{\boldsymbol{X}}^{(\ell)}-\boldsymbol{X}^{(\ell)}\right\|_F\leq\sum_{j=1}^{\ell}\epsilon_j,\quad \ell=1,\cdots,L. \tag{9}$$

It is noteworthy that the normalization assumption $\|\boldsymbol{W}_\ell\|_1=1$ in Theorem 1 is made with no loss in generality, and is only a way of presenting the result in a standard form. This is simply because $\texttt{ReLU}(|\alpha|x)=|\alpha|\texttt{ReLU}(x)$, and a scaling of any of the weight matrices $\boldsymbol{W}_\ell$ would scale $\boldsymbol{X}^{(L)}$ (or $\boldsymbol{X}^{(\ell')}$ where $\ell'\geq\ell$) by the same amount. Specifically, the outcomes of the network before and after the process obey

$$\left\|\hat{\boldsymbol{X}}^{(L)}-\boldsymbol{X}^{(L)}\right\|_F\leq\sum_{j=1}^{L}\epsilon_j,$$

which makes parallel Net-Trim a stable process, producing a controllable overall discrepancy.

A more adaptive way of retraining a network, which we would refer to as the cascade Net-Trim, incorporates the outcome of the previously pruned layers to retrain a target layer. Basically, in a cascade Net-Trim, retraining $\boldsymbol{W}_\ell$ takes place by exploring a path between the input/output pairs $(\hat{\boldsymbol{X}}^{(\ell-1)},\boldsymbol{X}_\ell)$ instead of $(\boldsymbol{X}^{(\ell-1)},\boldsymbol{X}_\ell)$. Due to some feasibility concerns, that will be detailed in the sequel, a cascade formulation does not simply happen by replacing $\boldsymbol{X}^{(\ell-1)}$ with $\hat{\boldsymbol{X}}^{(\ell-1)}$ in (8), and the formulation requires some modifications.

To derive the cascade formulation, consider starting the process by retraining the first layer via

$$\hat{\boldsymbol{W}}_1=\underset{\boldsymbol{U}}{\arg\min}\ \|\boldsymbol{U}\|_1\quad\text{subject to}\quad\boldsymbol{U}\in\mathcal{C}_{\epsilon_1}\left(\boldsymbol{X},\boldsymbol{X}^{(1)},\boldsymbol{0}\right). \tag{10}$$

Setting $\hat{\boldsymbol{X}}^{(1)}=\texttt{ReLU}(\hat{\boldsymbol{W}}_1{}^\top\boldsymbol{X})$, to adaptively prune the second layer, one would ideally consider the program

$$\underset{\boldsymbol{U}}{\text{minimize}}\ \|\boldsymbol{U}\|_1\quad\text{subject to}\quad\boldsymbol{U}\in\mathcal{C}_{\epsilon_2}\left(\hat{\boldsymbol{X}}^{(1)},\boldsymbol{X}^{(2)},\boldsymbol{0}\right). \tag{11}$$

It is not hard to see that the simple generalization in (11) is not guaranteed to be feasible, that is, there exists a matrix $\boldsymbol{W}$ such that for $\Omega=\text{supp}\boldsymbol{X}^{(2)}$:

$$\begin{cases}\left\|\left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{(1)}-\boldsymbol{X}^{(2)}\right)_\Omega\right\|_F\leq\epsilon_2\\ \left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{(1)}\right)_{\Omega^c}\leq\boldsymbol{0}\end{cases}. \tag{12}$$

If instead of $\hat{\boldsymbol{X}}^{(1)}$ the constraint set (12) was parameterized by $\boldsymbol{X}^{(1)}$, a natural feasible point would have been $\boldsymbol{W}=\boldsymbol{W}_2$. Now that $\hat{\boldsymbol{X}}^{(1)}$ is a perturbed version of $\boldsymbol{X}^{(1)}$, the constraint set needs to be properly slacked to maintain the feasibility of $\boldsymbol{W}_2$. In this context, one may easily verify that $\boldsymbol{W}_2$ is feasible for the slacked program

$$\underset{\boldsymbol{U}}{\text{minimize}}\ \|\boldsymbol{U}\|_1\quad\text{subject to}\quad\boldsymbol{U}\in\mathcal{C}_{\epsilon_2}\left(\hat{\boldsymbol{X}}^{(1)},\boldsymbol{X}^{(2)},\boldsymbol{W}_2^\top\hat{\boldsymbol{X}}^{(1)}\right), \tag{13}$$

as long as for some $\gamma\geq1$,

$$\epsilon_2=\gamma\left\|\left(\boldsymbol{W}_2^\top\hat{\boldsymbol{X}}^{(1)}-\boldsymbol{X}^{(2)}\right)_\Omega\right\|_F.$$

The γ-coefficient is a free parameter, which we refer to as the *inflation rate*. When $\gamma = 1$, the matrix $\boldsymbol{W}_2$ is only tightly feasible for (13) and the feasible set can at the very least become a singleton. However, increasing the inflation rate would expand the set of permissible matrices and makes (13) capable of producing sparser solutions.

The process applied to the second layer may be generalized to the subsequent layers and form a cascade paradigm to prune the network layer by layer. The pseudocode in Algorithm 1 summarizes the Net-Trim cascade scheme, where we set $\epsilon_1 = \epsilon$ for the first layer, and consider the inflation rates γ_ℓ, $\ell = 2, \cdots, L$, for the subsequent layers.

Algorithm 1: Cascade Net-Trim

$\hat{\boldsymbol{W}}_1 \leftarrow \arg\min_{\boldsymbol{U}} \ \|\boldsymbol{U}\|_1 \quad \text{subject to} \quad \boldsymbol{U} \in \mathcal{C}_\epsilon\left(\boldsymbol{X}, \boldsymbol{X}^{(1)}, \boldsymbol{0}\right)$

$\hat{\boldsymbol{X}}^{(1)} \leftarrow \texttt{ReLU}\left(\hat{\boldsymbol{W}}_1{}^\top \boldsymbol{X}\right)$

for $\ell = 2, \cdots, L$ **do**

$\Omega \leftarrow \text{supp}\boldsymbol{X}^{(\ell)}; \quad \epsilon_\ell \leftarrow \gamma_\ell \left\| \left(\boldsymbol{W}_\ell^\top \hat{\boldsymbol{X}}^{(\ell-1)} - \boldsymbol{X}^{(\ell)}\right)_\Omega \right\|_F$

$\hat{\boldsymbol{W}}_\ell \leftarrow \arg\min_{\boldsymbol{U}} \ \|\boldsymbol{U}\|_1 \quad \text{subject to} \quad \boldsymbol{U} \in \mathcal{C}_{\epsilon_\ell}\left(\hat{\boldsymbol{X}}^{(\ell-1)}, \boldsymbol{X}^{(\ell)}, \boldsymbol{W}_\ell^\top \hat{\boldsymbol{X}}^{(\ell-1)}\right)$

$\hat{\boldsymbol{X}}^{(\ell)} \leftarrow \texttt{ReLU}\left(\hat{\boldsymbol{W}}_\ell{}^\top \hat{\boldsymbol{X}}^{(\ell-1)}\right)$

end

Similar to the parallel scheme, we can show a bounded discrepancy between the outcomes of the initial network $\mathcal{N}et(\{\boldsymbol{W}_\ell\}_{\ell=1}^L; \boldsymbol{X})$ and the retrained network $\mathcal{N}et(\{\hat{\boldsymbol{W}}_\ell\}_{\ell=1}^L; \boldsymbol{X})$, as follows.

Theorem 2 (Cascade Net-Trim) *Consider a normalized network $\mathcal{N}et(\{\boldsymbol{W}_\ell\}_{\ell=1}^L; \boldsymbol{X})$, such that $\|\boldsymbol{W}_\ell\|_1 = 1$ for $\ell = 1, \cdots, L$. If the network is retrained according to Algorithm 1, the layer outcomes of the original and retrained networks will obey*

$$\left\| \hat{\boldsymbol{X}}^{(\ell)} - \boldsymbol{X}^{(\ell)} \right\|_F \leq \epsilon \prod_{j=2}^{\ell} \gamma_j. \tag{14}$$

Specifically, when an identical inflation rate is used across all the layers, one would have $\|\hat{\boldsymbol{X}}^{(L)} - \boldsymbol{X}^{(L)}\|_F \leq \gamma^{(L-1)}\epsilon$, which is a controllably small quantity, given that γ can be selected arbitrarily close to 1. For instance when $\gamma = 1.01$ and $L = 10$, the total network discrepancy would be still less than 1.1ϵ. As will be demonstrated in the experiments section, for the same level of total network discrepancy, the cascade Net-Trim is capable of producing sparser networks. However, such reduction is achieved at the expense of the loss in distributability, which makes the parallel scheme computationally more attractive for big data problems.

4 Sample Complexity Bounds Using Subgaussian Random Flow

In the previous section we discussed and analyzed the convex retraining scheme and its consistency with the reference model. In this section we analyze the sample complexity of the proposed retraining framework. Basically, the goal of this section is addressing the following question: *if there exists a sparse transformation matrix relating the input and output of a layer, how many random samples are sufficient to recover it via the proposed retraining scheme?*

As will be detailed in the sequel, we will show that retraining each neuron within the network is possible with fewer samples than the neuron degrees of freedom. More specifically, for a trained neuron with N input ports, if generating an identical response is possible with $s \ll N$ nonzero weights, Net-Trim is able to recover such model with only $\mathcal{O}(s \log(N/s))$ random samples. This result is valid for the neurons of any layer within the network, as long as some standard statistical properties can be established for the input samples.

Unlike the previous work ([AANR17]), which establishes a similar result for only the neurons within the first layer, here thanks to some favorable tail properties of subgaussian random vectors, we are able to generalize the result to the entire network. Basically, we will show that when the network input samples are independently drawn from a standard normal (or any other subgaussian) distribution, the input samples at all subsequent layers remain independent and subgaussian (what we refer to as a subgaussian flow). By carefully using some technical tools from the structured signal recovery literature ([Tro15, Men14]), we are able to present the main sample complexity result in a general form.

To present the results, we first start with a brief overview of subgaussian random variables. For a more comprehensive overview, the reader is referred to ([Ver12] and §2.2 of [vdVW96]).

Definition 1 (subgaussian random variable) *A random variable φ is subgaussian*[3] *if there exists a constant κ, such that for all $t \geq 0$,*

$$\mathbb{P}\left\{|\varphi| > t\right\} \leq \exp\left(1 - \frac{t^2}{\kappa^2}\right). \tag{15}$$

Equivalently, φ is subgaussian if there exists a constant $\hat{\kappa}$ such that

$$\mathbb{E}\exp\left(\frac{\varphi^2}{\hat{\kappa}^2}\right) \leq e. \tag{16}$$

The subgaussian norm of φ, also referred to as the Orlicz norm, is denoted by $\|\varphi\|_{\psi_2}$, and defined as

$$\|\varphi\|_{\psi_2} \triangleq \sup_{p \geq 1} p^{-\frac{1}{2}} \left(\mathbb{E}|\varphi|^p\right)^{\frac{1}{p}}.$$

While calculating the exact Orlicz norm can be challenging, if either one of the properties (15) or (16) hold, $\|\varphi\|_{\psi_2}$ is the smallest possible number (κ or $\hat{\kappa}$) in either one of these inequalities, up to an absolute constant.

Definition 2 (subgaussian random vector) *A random vector $\boldsymbol{\varphi} \in \mathbb{R}^N$ is subgaussian if for all $\boldsymbol{\alpha} \in \mathbb{R}^N$ (or equivalently all $\boldsymbol{\alpha} \in \mathbb{S}^{N-1}$), the one-dimensional marginals $\boldsymbol{\alpha}^\top \boldsymbol{\varphi}$ are subgaussian.*

The notion of Orlicz norm also generalizes to the vector case as

$$\|\boldsymbol{\varphi}\|_{\psi_2} \triangleq \sup_{\boldsymbol{\alpha} \in \mathbb{S}^{N-1}} \|\boldsymbol{\alpha}^\top \boldsymbol{\varphi}\|_{\psi_2} = \sup_{\boldsymbol{\alpha} \in \mathbb{S}^{N-1}} \sup_{p \geq 1} p^{-\frac{1}{2}} \left(\mathbb{E}\left|\boldsymbol{\alpha}^\top \boldsymbol{\varphi}\right|^p\right)^{\frac{1}{p}}. \tag{17}$$

We are now ready to state the first result, which warrants a subgaussian random flow across the network, as long as the network input samples are independently drawn from a standard Gaussian (or subgaussian) distribution.

Theorem 3 *Consider a network with fixed parameters $\boldsymbol{W}_\ell$, $\boldsymbol{b}^{(\ell)}$, where the input and output to each layer are related via*

$$\boldsymbol{x}^{(\ell)} = \texttt{ReLU}\left(\boldsymbol{W}_\ell^\top \boldsymbol{x}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\right), \qquad \ell = 1, \cdots, L. \tag{18}$$

If the network is fed with i.i.d sample vectors $\boldsymbol{x}_1^{(0)}, \cdots, \boldsymbol{x}_P^{(0)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, the response samples at each layer output remain i.i.d subgaussian.

[3] In general, the right-hand expression in (15) can be replaced with $c\exp\left(-t^2/\kappa^2\right)$ using two absolute constants c and κ

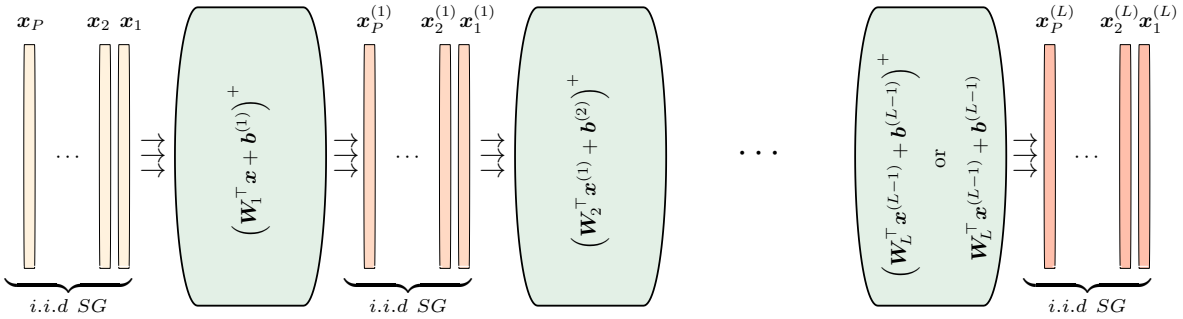


Figure 1: When $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_P$, the input samples to the proposed neural network, are independently drawn from a subgaussian distribution, the input/output vectors of every subsequent layer remain i.i.d and subgaussian

As shown in the proof, the result of Theorem 3 still holds when the network input samples are independently drawn from a subgaussian distribution instead of a standard normal, and/or when the last layer skips a ReLU activation. Specifically, when the network is fed with $\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}, \cdots, \boldsymbol{x}_P^{(0)}$, independently drawn from a subgaussian distribution, the resulting responses $\boldsymbol{x}_1^{(\ell)}, \boldsymbol{x}_2^{(\ell)}, \cdots, \boldsymbol{x}_P^{(\ell)}$ at any layer ℓ remain independent and subgaussian. The diagram in Figure 1 demonstrates such statistical structure among the layer inputs across the network.

Having independent subgaussian samples at any layer input port allows us to relate the number of samples to the recovery of a reduced model. Exchanging the layer index with the general input/output notation, when $\boldsymbol{X}^{in} \in \mathbb{R}^{N\times P}$ and $\boldsymbol{X}^{out} \in \mathbb{R}^{M\times P}$ are respectively the input and output to a layer, related via $\boldsymbol{X}^{out} = \texttt{ReLU}(\boldsymbol{W}^\top \boldsymbol{X}^{in})$, obtaining the pruned layer matrix $\hat{\boldsymbol{W}} \in \mathbb{R}^{N\times M}$ is performed via

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\arg\min} \ \|\boldsymbol{W}\|_1 \quad \text{subject to} \quad \boldsymbol{W} \in \mathcal{C}_\epsilon\left(\boldsymbol{X}^{in}, \boldsymbol{X}^{out}, \boldsymbol{0}\right). \tag{19}$$

When $\epsilon = 0$, the program in (19) decouples into M individual convex programs each retraining a column in $\boldsymbol{W}$. Basically, instead of solving (19) for $\hat{\boldsymbol{W}}$, if $\boldsymbol{x}^{out^\top} \in \mathbb{R}^P$ is a row in $\boldsymbol{X}^{out}$, the corresponding column in $\hat{\boldsymbol{W}}$ can be calculated via

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \|\boldsymbol{w}\|_1 \quad \text{subject to} \quad \boldsymbol{w} \in \mathcal{C}_0\left(\boldsymbol{X}^{in}, \boldsymbol{x}^{out^\top}, \boldsymbol{0}\right), \tag{20}$$

reducing (19) to retraining each of the M output neurons, individually. Our focus on the case of $\epsilon = 0$ (which also makes the cascade and parallel schemes equivalent) is working in an underdetermined regime, where the required samples are shown to be much less than the layer (neuron) degrees of freedom. In this case, the relationship between $\boldsymbol{X}^{in}$ and $\boldsymbol{X}^{out}$ can be established via infinitely many $\boldsymbol{W}$ matrices and one seeks a unique sparse solution via (19).

Before stating the main technical result, we would like to introduce some notions used in the presentation. When a neuron is initially trained via a vector $\boldsymbol{w}_0 \in \mathbb{R}^N$ and fed with i.i.d instances of $\boldsymbol{x}$, the activation pattern of the neuron is fully controlled by the sign of $\boldsymbol{w}_0^\top \boldsymbol{x}$. In this case, one expects to gain the main retraining information from the cases when ReLU is in the linear mode (i.e., $\boldsymbol{w}_0^\top \boldsymbol{x} > 0$). In this regard, corresponding to the random input $\boldsymbol{x}$, we define the random *virtual input* as

$$\boldsymbol{v} = \boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x} > 0} = \begin{cases} \boldsymbol{x} & \boldsymbol{w}_0^\top \boldsymbol{x} > 0 \\ \boldsymbol{0} & \boldsymbol{w}_0^\top \boldsymbol{x} \le 0 \end{cases}.$$

The virtual random vector plays a key role in our presentation. Our presentation also depends on the smallest

eigenvalue of the virtual covariance matrix, which follows the standard definition:

$$\lambda_{\min}\left(\operatorname{cov}\left(\boldsymbol{\upsilon}\right)\right)=\inf_{\boldsymbol{\alpha}\in\mathbb{S}^{N-1}}\boldsymbol{\alpha}^{\top}\operatorname{cov}\left(\boldsymbol{\upsilon}\right)\boldsymbol{\alpha},\quad\text{where}\quad\operatorname{cov}\left(\boldsymbol{\upsilon}\right)=\mathbb{E}\left(\boldsymbol{\upsilon}\boldsymbol{\upsilon}^{\top}\right)-\mathbb{E}\left(\boldsymbol{\upsilon}\right)\mathbb{E}\left(\boldsymbol{\upsilon}\right)^{\top}.$$

Theorem 4 *For the model in* (3)*, consider a trained neuron obeying* $\boldsymbol{x}^{out}=\texttt{ReLU}(\boldsymbol{X}^{in\,\top}\boldsymbol{w}_0)$*, where* $\boldsymbol{X}^{in}=[\boldsymbol{x}_1,\cdots,\boldsymbol{x}_P]\in\mathbb{R}^{N\times P}$ *and* $\boldsymbol{x}$*,* $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_P$ *are independent samples of a subgaussian distribution. Assume, an s-sparse vector* $\boldsymbol{w}^*\in\mathbb{R}^N$ *is capable of generating an identical response to* $\boldsymbol{X}^{in}$ *as* $\boldsymbol{x}^{out}$*. Fix* $\beta\geq 1/2$ *and* $t\geq 0$*, then if*

$$P\gtrsim C_{\beta,\upsilon}\left(s\log\left(\frac{N}{s}\right)+s+1+t\right),\tag{21}$$

retraining the neuron via (20) *recovers* $\boldsymbol{w}^*$ *with probability exceeding* $1-e^{-ct}$*. The absolute constant c is universal and the constant* $C_{\beta,\upsilon}$ *relates to the statistics of the virtual input* $\boldsymbol{\upsilon}=\boldsymbol{x}1_{\boldsymbol{w}_0^{\top}\boldsymbol{x}>0}$ *via*

$$C_{\beta,\upsilon}=(1+\beta)^2\left(\frac{\|\boldsymbol{\upsilon}-\mathbb{E}\boldsymbol{\upsilon}\|_{\psi_2}^2}{\lambda_{\min}(\operatorname{cov}(\boldsymbol{\upsilon}))}\right)^{3+\frac{1}{\beta}}.\tag{22}$$

We would like to highlight some technical details related to Theorem 4. To establish the result we use the *bowling scheme* proposed by [Tro15], which discusses the recovery of a structured (e.g., sparse) signal from independent linear measurements. We are able to make a connection between our nonlinear problem and the linear setup required in such scheme. While we used the compact model (3) for a more concise presentation, the model in (2) is still covered by Theorem 4, treating the intercept as a constant feature appended to the neuron input.

It is important to note that due to the application of the ReLU at each layer, the random samples entering the next layer are non-centered and this requires a careful analysis of the problem. In fact, the majority of the measurement systems in the structured recovery literature work with centered random measurements, as some of the powerful analysis tools, such as the restricted isometry property ([Can08, CRT06]), the certificate of duality ([CP11, Gro11]), and the Mendelson's small ball method – which stands as the backbone for the bowling scheme ([KM15, Men14, Men17]) rely on such setup. In the presentation of Theorem 4, the constant is related to the statistics of the centered virtual input, regardless of the mean shift that the previous activation units have caused to the input[4].

Finally, Theorem 4 can be used as a general and powerful tool to estimate the retraining sample complexity for any layer within the network. To establish the $\mathcal{O}(s\log(N/s))$ rate for a given layer, we only need to show that for the corresponding input $\boldsymbol{x}$ and initially trained weights $\boldsymbol{w}_0$, the virtual input $\boldsymbol{\upsilon}=\boldsymbol{x}1_{\boldsymbol{w}_0^{\top}\boldsymbol{x}>0}$ satisfies the following two conditions:

$$\lambda_{\min}\left(\operatorname{cov}\left(\boldsymbol{\upsilon}\right)\right)\gtrsim 1,\qquad\text{and}\qquad\|\boldsymbol{\upsilon}-\mathbb{E}\boldsymbol{\upsilon}\|_{\psi_2}\lesssim 1.\tag{23}$$

As an insightful example, we go through the exercise of establishing the bounds in (23) for a layer fed with i.i.d Gaussian samples. This could be a realistic feature setup for the first layer of a neural network. As will be detailed in Section 4.1 below, using standard tools to verify the conditions in (23), conveniently proves the $\mathcal{O}(s\log(N/s))$ rate for such layer.

For a network fed with i.i.d Gaussian samples, going through a similar exercise for the subsequent layers (say layer $\ell>1$, with independent copies of the random input $\boldsymbol{x}^{(\ell)}$), requires tracing the statistics of $\boldsymbol{\upsilon}^{(\ell)}=\boldsymbol{x}^{(\ell)}1_{\boldsymbol{w}_0^{\top}\boldsymbol{x}^{(\ell)}>0}$ down to the Gaussian input $\boldsymbol{x}^{(0)}$. In such case, warranting the conditions in (23) would require stating realistic conditions on the initially trained $\boldsymbol{W}_j$ for $j=1,\cdots,\ell$. Such generalization could be application specific and beyond the current load of the paper, which is left as a potential future work.

[4]This is important because the Orlicz norm of a noncentered random vector can easily become dimension-dependent. For instance, if the components of $\boldsymbol{x}\in\mathbb{R}^N$ are i.i.d standard Gaussians, one can easily verify that $\|\boldsymbol{x}^+\|_{\psi_2}=\mathcal{O}(\sqrt{N})$, while $\|\boldsymbol{x}^+-\mathbb{E}\boldsymbol{x}^+\|_{\psi_2}=\mathcal{O}(1)$.

4.1 Feeding a Neuron with i.i.d Gaussian Samples

In this section we go through the exercise of establishing the conditions in (23) for a neuron fed with independent copies of $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{x} \in \mathbb{R}^N$. Below, we go through each bound in (23), separately. In all the calculations, $\boldsymbol{w}_0 \neq \boldsymbol{0}$ is a fixed vector that corresponds to the initially trained model.

4.1.1 Step 1: Bounding the Covariance Matrix

To evaluate the virtual input covariance matrix we have

$$\begin{aligned}
\lambda_{\min}\left(\operatorname{cov}\left(\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)\right) &= \lambda_{\min}\left(\mathbb{E}\boldsymbol{x}\boldsymbol{x}^\top 1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \left(\mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)\left(\mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)^\top\right) \\
&\geq \lambda_{\min}\left(\mathbb{E}\boldsymbol{x}\boldsymbol{x}^\top 1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right) + \lambda_{\min}\left(-\left(\mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)\left(\mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)^\top\right) \\
&= \lambda_{\min}\left(\mathbb{E}\boldsymbol{x}\boldsymbol{x}^\top 1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right) - \left\|\mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right\|^2, \qquad (24)
\end{aligned}$$

where the second line is thanks to the Weyl's inequality. To conveniently calculate the required moments, we can make use of the following lemma, which reduces the calculations to the bivariate case.

Lemma 1 *Consider $\boldsymbol{x} = (x_1, \cdots, x_N)^\top \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and let $g(.) : \mathbb{R} \to \mathbb{R}$ be a real-valued function. Then, for any fixed vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{S}^{N-1}$:*

$$\mathbb{E}_{\boldsymbol{x}}\ g\left(\boldsymbol{\alpha}^\top \boldsymbol{x}\right) 1_{\boldsymbol{\beta}^\top \boldsymbol{x}>0} = \mathbb{E}_{x_1,x_2}\ g\left(\left(\boldsymbol{\alpha}^\top \boldsymbol{\beta}\right) x_1 + \sqrt{1-\left(\boldsymbol{\alpha}^\top \boldsymbol{\beta}\right)^2} x_2\right) 1_{x_1>0}. \qquad (25)$$

With no loss of generality we can assume $\boldsymbol{w}_0 \in \mathbb{S}^{N-1}$, and apply (25) to the first right-hand side term in (24) to get

$$\begin{aligned}
\lambda_{\min}\left(\mathbb{E}\boldsymbol{x}\boldsymbol{x}^\top 1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right) &= \inf_{\boldsymbol{\alpha}\in\mathbb{S}^{N-1}} \mathbb{E}\left(\boldsymbol{\alpha}^\top \boldsymbol{x}\right)^2 1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} \\
&= \inf_{\boldsymbol{\alpha}\in\mathbb{S}^{N-1}} \mathbb{E}_{x_1,x_2}\left(\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_0\right) x_1 + \sqrt{1-\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_0\right)^2} x_2\right)^2 1_{x_1>0} \\
&= \inf_{\boldsymbol{\alpha}\in\mathbb{S}^{N-1}} \frac{1}{2}\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_0\right)^2 + \frac{1}{2}\left(1-\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_0\right)^2\right) \\
&= \frac{1}{2}
\end{aligned}$$

For the second term in (24) we have

$$\begin{aligned}
\left\|\mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right\| &= \sup_{\boldsymbol{\alpha}\in\mathbb{S}^{N-1}} \mathbb{E}\left(\boldsymbol{\alpha}^\top \boldsymbol{x}\right) 1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} \\
&= \sup_{\boldsymbol{\alpha}\in\mathbb{S}^{N-1}} \mathbb{E}_{x_1,x_2}\left(\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_0\right) x_1 + \sqrt{1-\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_0\right)^2} x_2\right) 1_{x_1>0} \\
&= \sup_{\boldsymbol{\alpha}\in\mathbb{S}^{N-1}} \frac{1}{\sqrt{2\pi}}\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_0\right) \\
&= \frac{1}{\sqrt{2\pi}},
\end{aligned}$$

as a result of which one has $\lambda_{\min}\left(\operatorname{cov}\left(\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)\right) \geq 1/2 - 1/(2\pi)$.

4.1.2 Step 2: Bounding the Orlicz Norm

To bound the Orlicz norm of the centered virtual input by a constant, we only need to introduce a constant κ such that for all $\boldsymbol{\alpha} \in \mathbb{S}^{N-1}$ the marginals $\boldsymbol{\alpha}^\top(\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0})$ obey (15). To this end, one has

$$\begin{aligned}\forall \boldsymbol{\alpha} \in \mathbb{S}^{N-1} : \left|\boldsymbol{\alpha}^\top \left(\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)\right| &\le \left|\boldsymbol{\alpha}^\top \boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right| + \left\|\mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right\| \\ &\le \left|\boldsymbol{\alpha}^\top \boldsymbol{x}\right| + \frac{1}{\sqrt{2\pi}}.\end{aligned}$$

As a result, for $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and any fixed $\boldsymbol{\alpha} \in \mathbb{S}^{N-1}$:

$$\begin{aligned}\forall t \ge 0 : \quad \mathbb{P}\left\{\left|\boldsymbol{\alpha}^\top \left(\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)\right| > t\right\} &\le \mathbb{P}\left\{\left|\boldsymbol{\alpha}^\top \boldsymbol{x}\right| + \frac{1}{\sqrt{2\pi}} > t\right\} \\ &= \mathbb{P}\left\{\left|\boldsymbol{\alpha}^\top \boldsymbol{x}\right| > \max\left(t - \frac{1}{\sqrt{2\pi}}, 0\right)\right\} \\ &\le \exp\left(-\frac{1}{2}\max\left(t - \frac{1}{\sqrt{2\pi}}, 0\right)^2\right),\end{aligned}$$

where in the last inequality we used the fact that $\boldsymbol{\alpha}^\top \boldsymbol{x} \sim \mathcal{N}(0,1)$ and for a standard normal variable z, $\mathbb{P}\{|z| \ge t\} \le \exp(-t^2/2)$ for all $t \ge 0$. Finally we can use the basic inequality stated in Lemma 2 of the proofs section to get

$$\forall t \ge 0 : \quad \mathbb{P}\left\{\left|\boldsymbol{\alpha}^\top \left(\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right)\right| > t\right\} \le \exp\left(1 - \frac{t^2}{2 + \frac{1}{2\pi}}\right),$$

which implies that $\left\|\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \mathbb{E}\boldsymbol{x}1_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}\right\|_{\psi_2} \lesssim 1$.

In ([AANR17]), the authors go through a chain of techniques to prove an $\mathcal{O}(s \log N)$ rate by carefully constructing a dual certificate for the convex program. Here, thanks to Theorem 4, such process is markedly reduced to establishing the conditions in (23), which is conveniently fulfilled using the standard tools above.

5 Net-Trim Implementation

In this section we discuss details of an ADMM implementation for the Net-Trim convex program. The approach that we suggest here is based on the *global variable consensus* (see §7.1 of [BPC+11]). This technique is useful in addressing convex optimizations with additively separable objectives.

For $\boldsymbol{W} \in \mathbb{R}^{N \times M}$, $\boldsymbol{X}^{in} \in \mathbb{R}^{N \times P}$, and $\Omega \subseteq \{1, \cdots, M\} \times \{1, \cdots, P\}$ the Net-Trim central program

$$\underset{\boldsymbol{W}}{\text{minimize}} \ \|\boldsymbol{W}\|_1 \quad \text{subject to} \quad \begin{cases} \left\|\left(\boldsymbol{W}^\top \boldsymbol{X}^{in} - \boldsymbol{X}^{out}\right)_\Omega\right\|_F \le \epsilon \\ \left(\boldsymbol{W}^\top \boldsymbol{X}^{in}\right)_{\Omega^c} \le \boldsymbol{V}_{\Omega^c} \end{cases}, \tag{26}$$

can be cast as the equivalent form

$$\underset{\substack{\boldsymbol{W}^{(1)} \in \mathbb{R}^{M \times P} \\ \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)} \in \mathbb{R}^{N \times M}}}{\text{minimize}} \ f_1\left(\boldsymbol{W}^{(1)}\right) + f_2\left(\boldsymbol{W}^{(2)}\right) \quad \text{subject to} \quad \begin{cases} \boldsymbol{W}^{(1)} = {\boldsymbol{W}^{(3)}}^\top \boldsymbol{X}^{in} \\ \boldsymbol{W}^{(2)} = \boldsymbol{W}^{(3)} \end{cases}, \tag{27}$$

where

$$f_1(\boldsymbol{W}) = \mathcal{I}_{\left\|\boldsymbol{W}_\Omega - \boldsymbol{X}_\Omega^{out}\right\|_F \le \epsilon}(\boldsymbol{W}) + \mathcal{I}_{\boldsymbol{W}_{\Omega^c} \le \boldsymbol{V}_{\Omega^c}}(\boldsymbol{W}), \text{ and } f_2(\boldsymbol{W}) = \|\boldsymbol{W}\|_1.$$

Here $\mathcal{I}_C(\cdot)$ represents the indicator function of the set C,

$$\mathcal{I}_C(\boldsymbol{W}) = \begin{cases} 0 & \boldsymbol{W} \in C \\ +\infty & \boldsymbol{W} \notin C \end{cases}.$$

For the convex program (27), the ADMM update for each variable at the k-th iteration follows the standard forms

$$\boldsymbol{W}_{k+1}^{(1)} = \arg\min_{\boldsymbol{W}} \; f_1(\boldsymbol{W}) + \frac{\rho}{2}\left\|\boldsymbol{W} + \boldsymbol{U}_k^{(1)} - {\boldsymbol{W}_k^{(3)}}^\top \boldsymbol{X}^{in}\right\|_F^2, \tag{28}$$

$$\boldsymbol{W}_{k+1}^{(2)} = \arg\min_{\boldsymbol{W}} \; f_2(\boldsymbol{W}) + \frac{\rho}{2}\left\|\boldsymbol{W} + \boldsymbol{U}_k^{(2)} - \boldsymbol{W}_k^{(3)}\right\|_F^2, \tag{29}$$

$$\boldsymbol{W}_{k+1}^{(3)} = \left(\boldsymbol{X}^{in}\boldsymbol{X}^{in\top} + \boldsymbol{I}\right)^{-1}\left(\boldsymbol{X}^{in}\left(\boldsymbol{W}_{k+1}^{(1)} + \boldsymbol{U}_k^{(1)}\right)^\top + \boldsymbol{W}_{k+1}^{(2)} + \boldsymbol{U}_k^{(2)}\right), \tag{30}$$

and the dual updates are performed via

$$\boldsymbol{U}_{k+1}^{(1)} = \boldsymbol{U}_k^{(1)} + \boldsymbol{W}_{k+1}^{(1)} - {\boldsymbol{W}_{k+1}^{(3)}}^\top \boldsymbol{X}^{in}, \quad \boldsymbol{U}_{k+1}^{(2)} = \boldsymbol{U}_k^{(2)} + \boldsymbol{W}_{k+1}^{(2)} - \boldsymbol{W}_{k+1}^{(3)}.$$

The update stated in (30) is derived by finding the minimizer of the augmented Lagrangian with respect to $\boldsymbol{W}^{(3)}$, which amounts to the minimization

$$\underset{\boldsymbol{W}}{\text{minimize}} \; \frac{\rho}{2}\left\|\boldsymbol{W}_{k+1}^{(1)} + \boldsymbol{U}_k^{(1)} - \boldsymbol{W}^\top \boldsymbol{X}^{in}\right\|_F^2 + \frac{\rho}{2}\left\|\boldsymbol{W}_{k+1}^{(2)} + \boldsymbol{U}_k^{(2)} - \boldsymbol{W}\right\|_F^2.$$

While the updates for $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$, as in (28) and (29), are stated in the general form, they can be further simplified and presented in closed form. To this end, a first observation is that (28) can be decoupled into independent minimizations in terms of $\boldsymbol{W}_\Omega$ and $\boldsymbol{W}_{\Omega^c}$, i.e.,

$$\begin{aligned}\boldsymbol{W}_{k+1}^{(1)} = &\underset{\boldsymbol{W}_\Omega:\, \left\|\boldsymbol{W}_\Omega - \boldsymbol{X}_\Omega^{out}\right\|_F \le \epsilon}{\arg\min} \frac{\rho}{2}\left\|\boldsymbol{W}_\Omega + \left(\boldsymbol{U}_k^{(1)} - {\boldsymbol{W}_k^{(3)}}^\top \boldsymbol{X}^{in}\right)_\Omega\right\|_F^2 \\ &+ \underset{\boldsymbol{W}_{\Omega^c}:\, \boldsymbol{W}_{\Omega^c} \le \boldsymbol{V}_{\Omega^c}}{\arg\min} \frac{\rho}{2}\left\|\boldsymbol{W}_{\Omega^c} + \left(\boldsymbol{U}_k^{(1)} - {\boldsymbol{W}_k^{(3)}}^\top \boldsymbol{X}^{in}\right)_{\Omega^c}\right\|_F^2.\end{aligned} \tag{31}$$

The first minimization on the right-hand side of (31) is basically the problem of finding the closest point of an ϵ-radius Euclidean ball to a given point. For the non-trivial case that the given point is outside the ball, the solution is the intersection of the ball surface with the line connecting the point to the center of the ball. More specifically, for fixed $\boldsymbol{Y}$ and $\boldsymbol{Z}$,

$$\underset{\boldsymbol{W}_\Omega:\, \|\boldsymbol{W}_\Omega - \boldsymbol{Z}_\Omega\|_F \le \epsilon}{\arg\min} \frac{\rho}{2}\|\boldsymbol{W}_\Omega - \boldsymbol{Y}_\Omega\|_F^2 = \begin{cases} \boldsymbol{Y}_\Omega & \text{if } \|\boldsymbol{Y}_\Omega - \boldsymbol{Z}_\Omega\|_F \le \epsilon \\ \boldsymbol{Z}_\Omega + \epsilon\frac{\boldsymbol{Y}_\Omega - \boldsymbol{Z}_\Omega}{\|\boldsymbol{Y}_\Omega - \boldsymbol{Z}_\Omega\|_F} & \text{else} \end{cases}.$$

The second term in (31) is an instance of a projection onto an orthant and can be delivered in closed form as

$$\underset{\boldsymbol{W}_{\Omega^c}:\, \boldsymbol{W}_{\Omega^c} \le \boldsymbol{V}_{\Omega^c}}{\arg\min} \frac{\rho}{2}\|\boldsymbol{W}_{\Omega^c} - \boldsymbol{Y}_{\Omega^c}\|_F^2 = \boldsymbol{Y}_{\Omega^c} - (\boldsymbol{Y}_{\Omega^c} - \boldsymbol{V}_{\Omega^c})^+.$$

Finally, the solution to (29) is the standard soft thresholding operator (e.g., see §4.4.3 of [BPC+11]), which reduces the update to

$$\left(\boldsymbol{W}_{k+1}^{(2)}\right)_{n,m} = S_{1/\rho}\left(\left(\boldsymbol{W}_{k}^{(3)} - \boldsymbol{U}_{k}^{(2)}\right)_{n,m}\right), \text{ where } S_c(w) = \begin{cases} w-c & w > c \\ 0 & |w| \le c \\ w+c & w < -c \end{cases}.$$

After combining the steps above, we propose Algorithm 2 as a computational scheme to address the Net-Trim central program[5]. The only computational load of the proposed scheme is the linear solve (30), for which the coefficient matrix $\boldsymbol{X}^{in}\boldsymbol{X}^{in\top} + \boldsymbol{I}$ only needs to be calculated once. As observable, the processing time for each ADMM step is relatively low, and only involves few matrix multiplications.

Algorithm 2: Implementation of the Net-Trim Central Program

Input: $\boldsymbol{X}^{in} \in \mathbb{R}^{N\times P}$, $\boldsymbol{X}^{out} \in \mathbb{R}^{M\times P}$, Ω, $\boldsymbol{V}_\Omega$, ϵ, ρ
initialize $\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}$ and $\boldsymbol{W}^{(3)}$ `% all initializations can be with 0`
$\boldsymbol{C} \leftarrow \boldsymbol{X}^{in}\boldsymbol{X}^{in\top} + \boldsymbol{I}$
while *not converged* **do**
  $\boldsymbol{Y} \leftarrow \boldsymbol{W}^{(3)\top}\boldsymbol{X}^{in} - \boldsymbol{U}^{(1)}$
  if $\left\|\boldsymbol{Y}_\Omega - \boldsymbol{X}_\Omega^{out}\right\|_F \le \epsilon$ **then**
    $\boldsymbol{W}_\Omega^{(1)} \leftarrow \boldsymbol{Y}_\Omega$
  else
    $\boldsymbol{W}_\Omega^{(1)} \leftarrow \boldsymbol{X}_\Omega^{out} + \epsilon\left\|\boldsymbol{Y}_\Omega - \boldsymbol{X}_\Omega^{out}\right\|_F^{-1}\left(\boldsymbol{Y}_\Omega - \boldsymbol{X}_\Omega^{out}\right)$
  end
  $\boldsymbol{W}_{\Omega^c}^{(1)} \leftarrow \boldsymbol{Y}_{\Omega^c} - (\boldsymbol{Y}_{\Omega^c} - \boldsymbol{V}_{\Omega^c})^+$
  $\boldsymbol{W}^{(2)} \leftarrow S_{1/\rho}\left(\boldsymbol{W}^{(3)} - \boldsymbol{U}^{(2)}\right)$ `%` $S_{1/\rho}$ `applies to each element of the matrix`
  $\boldsymbol{W}^{(3)} \leftarrow \boldsymbol{C}^{-1}\left(\boldsymbol{X}^{in}(\boldsymbol{W}^{(1)} + \boldsymbol{U}^{(1)})^\top + \boldsymbol{W}^{(2)} + \boldsymbol{U}^{(2)}\right)$
  $\boldsymbol{U}^{(1)} \leftarrow \boldsymbol{U}^{(1)} + \boldsymbol{W}^{(1)} - \boldsymbol{W}^{(3)\top}\boldsymbol{X}^{in}$
  $\boldsymbol{U}^{(2)} \leftarrow \boldsymbol{U}^{(2)} + \boldsymbol{W}^{(2)} - \boldsymbol{W}^{(3)}$
end
Output: $\boldsymbol{W}^{(3)}$

6 Experiments

The theoretical results above give us a mathematical understanding of Net-Trim's performance. In this section, we present a series of numerical experiments, which demonstrate that Net-Trim is also an efficient and reliable tool for model reduction of real-world neural networks.

Below, we compare how Net-Trim performs in both cascade and parallel modes; the former is slightly more accurate, while the latter can be made extremely computationally efficient through parallelization. We then compare Net-Trim against some other well-known tools for network sparsification, including Dropout, ℓ_1 regularized training, and the pruning using weight truncation.

Our first set of experiments corresponds to a comparison between the cascade and parallel frameworks. In this experiment we also assess the possibility of applying Net-Trim to only a portion of the training data. More specifically, if $\boldsymbol{X}$ is the entire training matrix used for the initial model, how much the Net-Trim retrained models

[5] To access the algorithm implementation, visit: `https://dnntoolbox.github.io/Net-Trim/`

differ using the entire $\boldsymbol{X}$, versus using a subset of the columns in $\boldsymbol{X}$. Clearly, working with smaller $\boldsymbol{X}$ matrices is computationally more desirable.

For this purpose we use a fully connected (FC) neural network of size $784 \times 300 \times 1000 \times 100 \times 10$ (composed of four layers: $\boldsymbol{W}_1 \in \mathbb{R}^{784\times 300}, \boldsymbol{W}_2 \in \mathbb{R}^{300\times 1000}$, etc), trained to classify hand-written digits of the mixed national institute of standards and technology (MNIST) dataset. Throughout the entire section we refer to this network as the FC model.

Panels (a) and (b) in Figure 2 summarize the outcome of applying the Net-Trim parallel scheme to the trained FC model. By varying the value of ϵ, one may explore different levels of layer sparsity and discrepancy. Panel (a) reports the relative value of the overall discrepancy (i.e., $\|\hat{\boldsymbol{X}}^L - \boldsymbol{X}^L\|_F / \|\boldsymbol{X}^L\|_F$) as a function of the relative sparsity at each layer (i.e., percentage of zeros in $\hat{\boldsymbol{W}}_\ell$). Each plot is obtained by varying ϵ for a range of values and retraining the FC model with 10K, 20K, 30K and the entire 55K training samples. As expected, allowing more discrepancy improves the level of sparsity. Since practically the overall discrepancy is not a good indication of the changes in the model accuracy, in panel (b) we replace it with the test accuracy of the retrained models. An interesting observation is that retraining the models with fewer samples does not significantly degrade the test accuracies and even in some cases (e.g., 30K versus 55K) it causes a slight improvement in the accuracy of the retrained models.

Panels (c) and (d) report a similar set of experiments for the cascade Net-Trim, where increasing the inflation rate away from one allows producing sparser networks. A quick comparison between the range of relative discrepancies in panels (c) and (a) reveals that for more or less similar sparsity rates, cascade Net-Trim causes a smaller overall discrepancy compared to the parallel scheme (note the axis ranges). This may be considered as the return for going through the more careful and non-distributable cascade scheme. On the other hand, a comparison of the test accuracies in panels (d) and (b), and especially for larger values of the sparsity ratio, shows a less significant difference between the test accuracies of the two schemes.

Employing the parallel scheme (thanks to its distributable nature), and the use of a subset of the training data in the Net-Trim retraining process are both computationally attractive paths, and the experiments in Figure 2 indicate that at least for a reasonable sparsity range, they could be both explored without much degradation of the model accuracies. In the remainder of the experiments in this section, we will consistently use the parallel scheme for our retraining purposes, and will no more reference to the Net-Trim parallel or cascade nature.

In the next set of experiments, we investigate the additional pruning that Net-Trim brings to the architecture of neural networks beyond Dropout and ℓ_1 regularization. These two technique are well-known tools that tend to regularize the training process and produce pruned networks. For this purpose we consider the application of an ℓ_1 regularization, Dropout and a combination of both to the training of our standard FC model.

We also apply a similar set of tools to the LeNet convolutional network ([LBBH98]), which is composed of two convolutional layers (32 filters of size 5×5 at the first layer, and 64 filters of similar size at the second layer, both followed by 2×2 max pooling units), and two fully connected layers ($3136 \times 512 \times 10$). While the linearity of the convolution operator immediately allows the application of Net-Trim, in our experiments we omit retraining the convolutional layers as the number of parameters in such layers is much less than the fully connected layers.

For both network architectures we vary λ (the ℓ_1 penalty), and p (the Dropout probability of keeping) in a range of values that tend to produce reasonably high test accuracies. The statistics reported in Table 1 correspond to the FC and LeNet models, which resulted in the highest test accuracies. For both architectures the best results happened when the Dropout and ℓ_1 regularization were applied simultaneously. The first row reports the initial model statistics and the subsequent rows correspond to the application of the Net-Trim using different values of ϵ. In this experiment the third column of each architecture section corresponds to an additional fine-tuning step (FT) after Net-Trim prunes the network. This (optional) step uses the Net-Trim solution as an initialization for a secondary training, which only applies to the non-zero weights identified by the Net-Trim. Such fine-tuning often

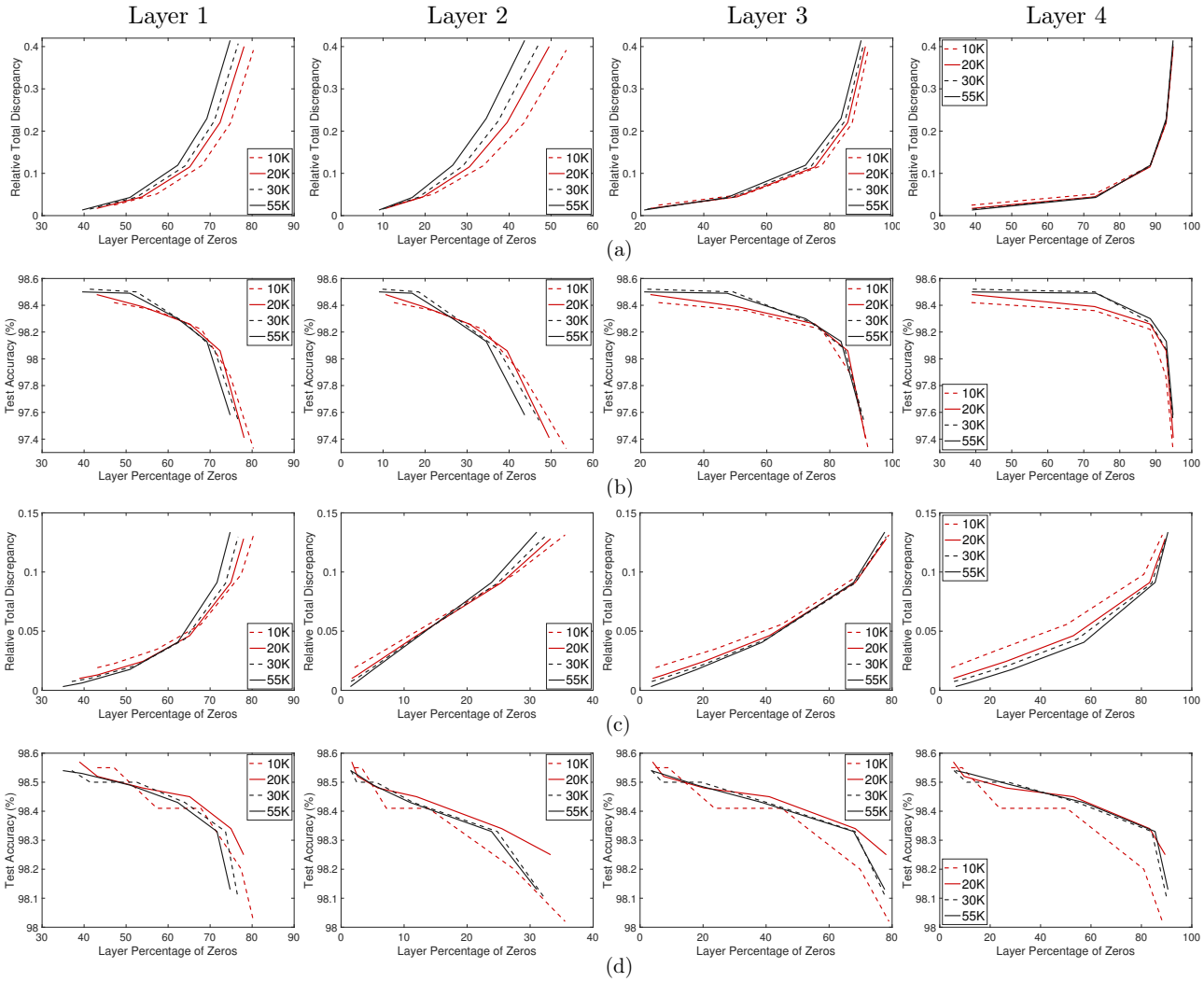


Figure 2: Retraining the FC network of size $784 \times 300 \times 1000 \times 100 \times 10$ that is initially trained with the full MNIST training set and retrained with 10K, 20K, 30K and 55K samples (each column corresponds to a layer); (a) network relative total discrepancy (RTD) vs the layer percentage of zeros (LPZ) after a parallel scheme; (b) network test accuracy vs LPZ after a parallel scheme; (c) network RTD vs LPZ after a cascade scheme; (d) network test accuracy vs LPZ after a cascade scheme;

allows an improvement in the generalization error without changing the sparsity of the network.

A quick assessment of Table 1 reveals that applying Net-Trim can significantly improve the sparsity (and even at the same time the accuracy) of the models. For instance, in the FC model we can improve the test accuracy to 98.76%, and at the same time increase the percentage of network zeros from 43.69% to 71.93%. A similar trend holds for the LeNet model. If we allow some degradation in the test accuracy, the percentage of zeros can be

Table 1: Application of Net-Trim for different values of ϵ to the standard (FC) and convolutional (LeNet) networks trained via careful choice of the ℓ_1 regularization and Dropout probability ($\lambda = 10^{-5}$, $p = 0.75$ for the FC model, and $\lambda = 10^{-5}$, $p = 0.5$ for the LeNet architecture); improved quantities compared to the initial models are highlighted in bold

		FC			LeNet		
		Network Zeros (%)	Test Acc. (No FT)	Test Acc. With FT	Network Zeros (%)	Test Acc. (No FT)	Test Acc. With FT
Initial Model		43.69	98.65	–	33.65	99.57	–
Net-Trim	$\epsilon = 0.01$	**71.93**	**98.65**	**98.76**	**83.80**	**99.59**	**99.60**
	$\epsilon = 0.02$	**76.13**	**98.65**	**98.72**	**88.76**	**99.60**	**99.57**
	$\epsilon = 0.04$	**80.02**	98.56	**98.66**	**92.75**	99.54	99.53
	$\epsilon = 0.06$	**81.98**	98.54	98.59	**94.46**	99.49	99.47
	$\epsilon = 0.08$	**83.34**	98.36	98.48	**95.40**	99.35	99.44
	$\epsilon = 0.1$	**84.30**	98.08	98.38	**96.01**	98.26	99.35
	$\epsilon = 0.2$	**86.99**	96.76	97.88	**97.37**	98.83	99.22
	$\epsilon = 0.3$	**88.61**	94.69	97.31	**97.89**	98.61	99.07

significantly increased to 88.61% in the FC model, and 97.89% in the LeNet architecture.

Table 1 only reports the Net-Trim performance on the most accurate models. A more comprehensive set of experiments on the two network architectures is reported in Figure 3. In these experiments the mean test accuracy and initial model sparsity are reported for the cases of Dropout, ℓ_1 regularization, and a combination of both. For each setup the tuning parameters (λ, p, or both) are varied in a range of values and unlike Table 1, the mean quantities are reported. For instance, panel (a) indicates that applying the Dropout to the FC model with $0.3 \leq p \leq 0.8$ yields an average network zero percentage of 0.07%, and approximately 97.5% test accuracy. However, applying Net-Trim along with the FT step, can elevate the average accuracy to around 98%, and at the same time increase the network percentage of zeros to almost 45%. The plot also reveals that with no loss in the model accuracies, we can improve the sparsity of the models to up to 56% (corresponding to the point where the red and the dashed lines intersect).

An assessment of all panels (specifically the crossing of the red curves and the dashed lines) reveals that in all three scenarios (Dropout, ℓ_1 regularization and a combination of both), and for both architectures (FC and LeNet), an additional application of Net-Trim can improve the models both in terms of accuracy and the number of underlying parameters. Even in cases that the accuracy is degraded to some extent, but the model is significantly pruned, the pruned network may be considered a more reliable model. In Figure (4) we have demonstrated the FC and LeNet models initially trained with Dropout and retrained using Net-Trim. Despite an accuracy loss of 1.3% for the FC model, and 1.7% for the LeNet model, the percentage of zeros have been increased to 63.32% and 96.8%, respectively. As a result of this reduction, when the models are tested with different noisy versions of the original test set, the reduced models exhibit a lower accuracy degradation (i.e., more robustness) to the noise increase.

Thanks to the simple implementation of Net-Trim (as outline in Algorithm 2), in the aforementioned experiments, the retraining of the layer matrices was only in order of few minutes on a standard desktop computer, while in the majority of the cases, the initial training of the networks took much longer. We would like to note that we did not make any efforts to optimize the Net-Trim code and fully exploit the parallel features (e.g., matrix products, processing of layers in parallel, etc). The distributable nature of Algorithm 2 supports yet much faster software than the one currently present.

We also compare Net-Trim with the more recent and well-known algorithm by Han, Pool, Tran and Dally (HPTD: [HPTD15]). The HPTD algorithm is a heuristic tool used for network compression, which truncates the small weights across a trained network and performs another round of training on the active weights (same as the

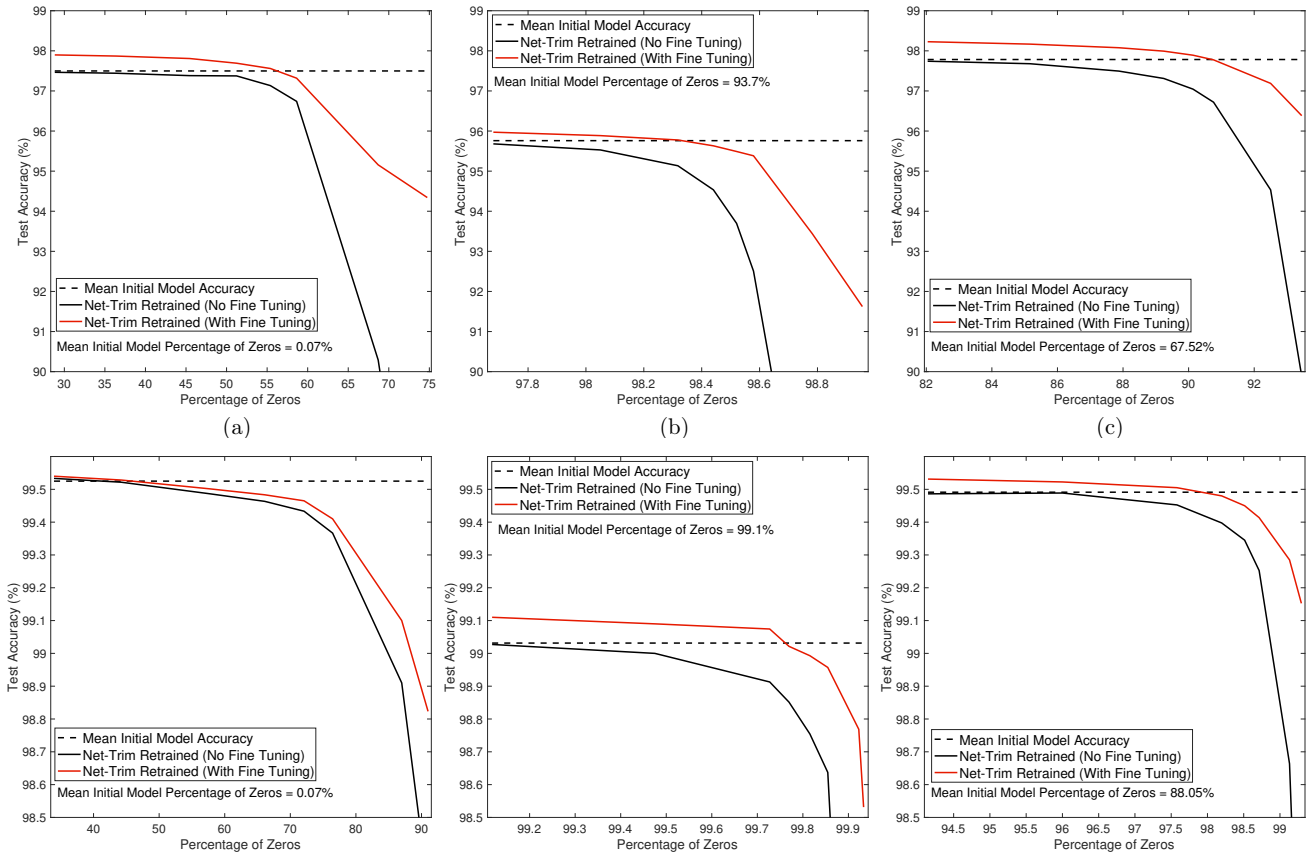
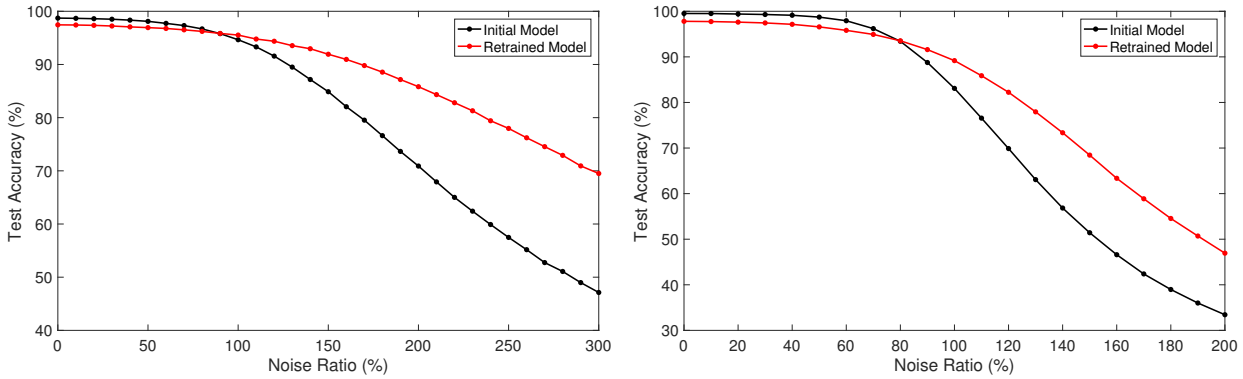


Figure 3: Mean test accuracy vs mean model sparsity after the application of Net-Trim for networks initially regularized via ℓ_1 penalty, Dropout, or both (the regularization parameter and Dropout probability are picked from a range of values and mean quantities are reported); (a) FC models trained with Dropout only: $0.3 \leq p \leq 0.8$; (b) FC models trained with ℓ_1 penalty only: $10^{-5} \leq \lambda \leq 5 \times 10^{-3}$; (c) FC models trained with Dropout and ℓ_1: $10^{-5} \leq \lambda \leq 2 \times 10^{-4}$, $0.5 \leq p \leq 0.75$; (d) LeNet models trained with Dropout only: $0.3 \leq p \leq 0.8$; (e) LeNet models trained with ℓ_1 penalty only: $10^{-5} \leq \lambda \leq 5 \times 10^{-3}$; (f) LeNet models trained with Dropout and ℓ_1: $10^{-5} \leq \lambda \leq 2 \times 10^{-4}$, $0.5 \leq p \leq 0.75$;

fine-tuning scheme explained above). The algorithm does not come with any performance guarantees, however, the basic implementation idea has made it a widespread tool in the network compression community. Using tools such as quantization and Huffman coding, more advanced frameworks such as the Deep Compression ([HMD16]) have been developed later. However, their focus is mainly compressing the network parameters on the memory, and HPTD pruning scheme is yet the most relevant single-module framework that could be compared with Net-Trim.

Figure 5 presents a comprehensive comparison between the Net-Trim and HPTD on the FC and LeNet models. For the Net-Trim we use different values of ϵ to prune the trained networks. To compare the method with the HPTD, after each application of the Net-Trim and counting the number of zeros, the same number of elements are



(a) (b)

Figure 4: Noise robustness of initial and retrained networks; (a) FC; (b) LeNet

truncated from the initial network to be used for the HPTD implementation. HPTD is followed by a fine-tuning step after the truncation, which is also an optional task for Net-Trim. Nevertheless, both algorithms are compared without fine-tuning, or with fine-tuning using 10 or 30 epochs. The left plots in panels (a) and (b) show that in all scenarios Net-Trim outperforms HPTD in generating more accurate models when the levels of sparsity are matched. The middle plots also show the improvements in the accuracy as a function of the number of epochs required in the fine-tuning process for the two schemes. In both scenarios, Net-Trim requires only few epochs to achieve the top accuracy, while achieving such level of accuracy for the HPTD is either not feasible or takes many fine-tuning epochs.

One of the main drawbacks with the HPTD is the truncation based on the magnitude of the weights, which in many cases may discard connections to the important features and variables in the network. That is mainly the reason that Net-Trim consistently outperforms this method, as the Net-Trim pruning takes place along with matching the intermediate responses of the initial and retrained networks. In fact, Net-Trim can also present vital information about the data structure and important features that are not immediately available using other techniques.

In Figure 6 we have depicted the retrained $\hat{\boldsymbol{W}}_1$ matrix of the FC model after applying Net-Trim and HPTD. In panel (b) we can see many columns that are fully zero. Plotting the histogram of the MNIST samples (as in panel (d)), one would immediately observe that the zero columns in $\hat{\boldsymbol{W}}_1$ correspond to the boundary pixels with the least level of information. As HPTD only relies on the truncation based on the weight magnitudes, despite the similar number of zeros in panels (b) and (c), the latter does not highlight such data structure. To obtain a similar pattern as in panel (b), the authors in ([HPTD15]) suggest an iterative pruning path with a fine-tuning after truncating a portion of the network weights. This is however not a computationally efficient path as it requires retraining the network multiple times, which can take a lot of time for large data sets and is not guaranteed to identify the right structures.

Figure 7 demonstrates another set of comparative experiments between Net-Trim and HPTD, performed on a much larger augmented dataset. The reference training set is the CIFAR10 color-image database, which contains 50K samples of size 32×32 from ten classes ([Kri10, KSH12]).

In order to obtain higher test accuracies, the training images are multiplicated by taking 24×24 windows to randomly crop them, and each cropped image is horizontally flipped with probability 0.5. This process augments

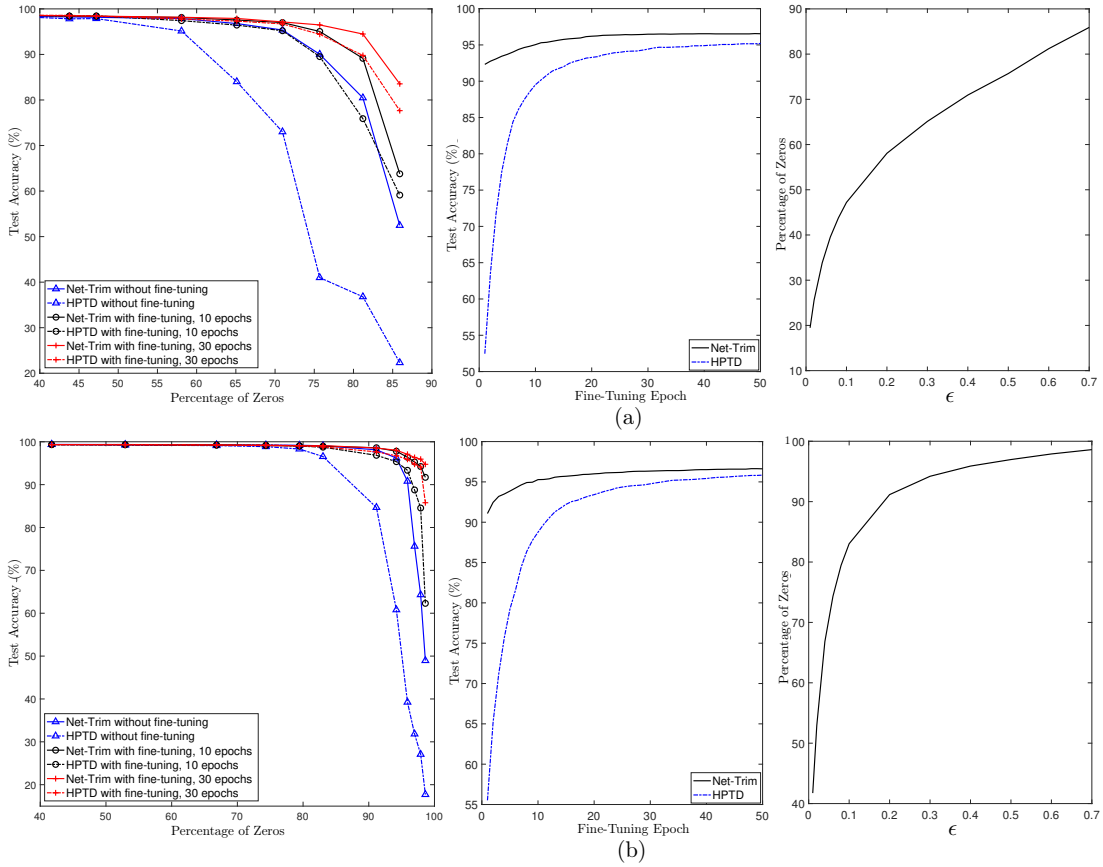


Figure 5: Comparison of Net-Trim and HPTD in different settings for (a) FC model, (b) LeNet model; the left panels compare Net-Trim and HPTD test accuracy vs percentage of zeros, without fine-tuning, and with fine-tuning using 10 and 30 epochs; middle panels show the number of fine-tuning epochs and the acquired accuracy using Net-Trim and HPTD; the right panels indicate the percentage of zeros as a function ϵ for Net-Trim

the training set to 6,400,000 samples. The neural network employed to address the initial classification problem is convolutional, where the first layer of the trained network uses 64 filters of size $5 \times 5 \times 3$, followed by a max pooling unit (size: 3×3, stride: 2×2). The second layer is also convolutional with 64 filters of size $5 \times 5 \times 64$ and a similar max pooling unit. The remainder of the network contains three fully connected layers ($3136 \times 384 \times 192 \times 10$).

For this relatively large dataset we also go through the exercise of retraining the Net-Trim with only part of the training samples, specifically 25K, 50K and 75K samples of the entire 6.4M training set. A similar set of comparisons between the Net-Trim and the HPTD as in Figure 5 is performed, noting that the fine-tuning step for both schemes is carried out using all the training samples. Similar to the previous experiment, Net-Trim consistently outperforms HPTD in all similar setups. Aside from such superiority, we highlight the possibility of retraining Net-Trim using only part of the training samples. For instance, a comparison of panels (a) and (b) shows that almost identical results can be achieved in terms of accuracy versus sparsity, when Net-Trim is solved

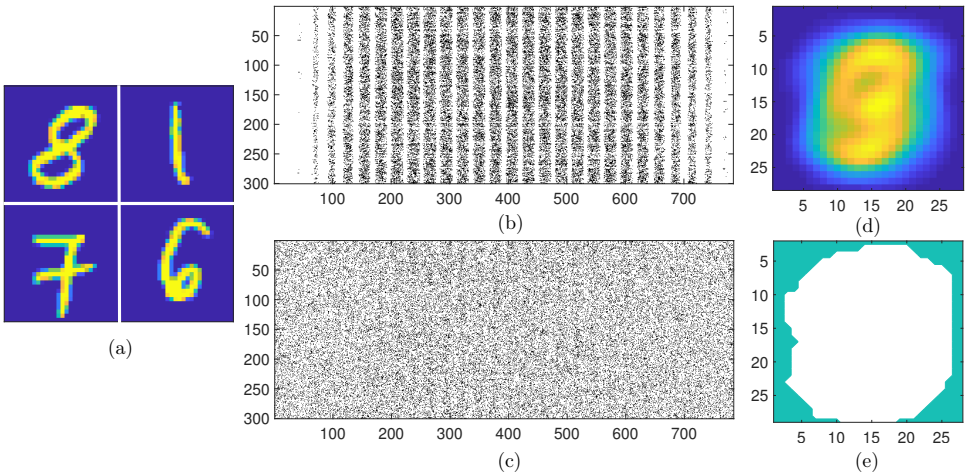


Figure 6: Instant identification of important features using Net-Trim; (a) samples from MNIST dataset; (b) visualization of $\hat{\boldsymbol{W}}_1{}^\top$ in the FC retrained model using Net-Trim; (c) similar visualization of $\hat{\boldsymbol{W}}_1{}^\top$ in the FC retrained model, using HPTD with a single fine-tuning step; (d) histogram of the pixel values in the MNIST data set; (e) the green mask corresponding to the zero columns in panel (b)

with 25K samples instead of 50K samples. For instance, for both panels, a Net-Trim application followed by a single fine-tuning step can increase the percentage of the zeros in the network to more than 80%, with almost no loss in the model accuracy. Basically, as also discussed previously with reference to Figure 2, for large data sets formulating the Net-Trim with only a portion of the data can be considered as a general computation shortcut.

Another interesting observation, which is more apparent on the left plot of panel (c), is that fine-tuning does not always improve the accuracy of the models after the application of Net-Trim, and especially in low pruning regimes may cause degrading the accuracy due to phenomena such as overfitting. For example, in panel (c), up to a pruning percentage of almost 65%, a fine-tuning step after the Net-Trim slightly degrades the accuracy. While a fine-tuning step is likely to help in the majority of cases, our access to both Net-Trim's plain outcome, and the fine-tuned version provides the flexibility of picking the most compressed and accurate model among the two.

7 Concluding Remarks

The main motivation for Net-Trim is introducing a class of techniques which work as post-training modules for neural networks, and in return simplify the regularization process, and provide an understanding of when and how well they work. Because of the complex structure of neural networks, to date the majority of techniques that are designed to regularize or improve the process of training neural networks are based on heuristics or empirical success, and lack a clear performance analysis.

Net-Trim can be generalized to a large class of problems, where the architecture of each layer in a trained network is restructured via a program of the type

$$\underset{\boldsymbol{U}\in\mathbb{R}^{N\times M}}{\text{minimize}} \quad \mathcal{R}\left(\boldsymbol{U}\right) \qquad \text{subject to} \qquad \sigma\left(\boldsymbol{U}^\top \boldsymbol{X}^{in}\right) \approx \boldsymbol{X}^{out}. \tag{32}$$

The objective $\mathcal{R}(\cdot)$ aims to promote a desired structure, and the constraint enforces a consistency between the initial and retrained models. While in this paper we merely emphasized on $\mathcal{R}(\boldsymbol{U}) = \|\boldsymbol{U}\|_1$, a variety of other

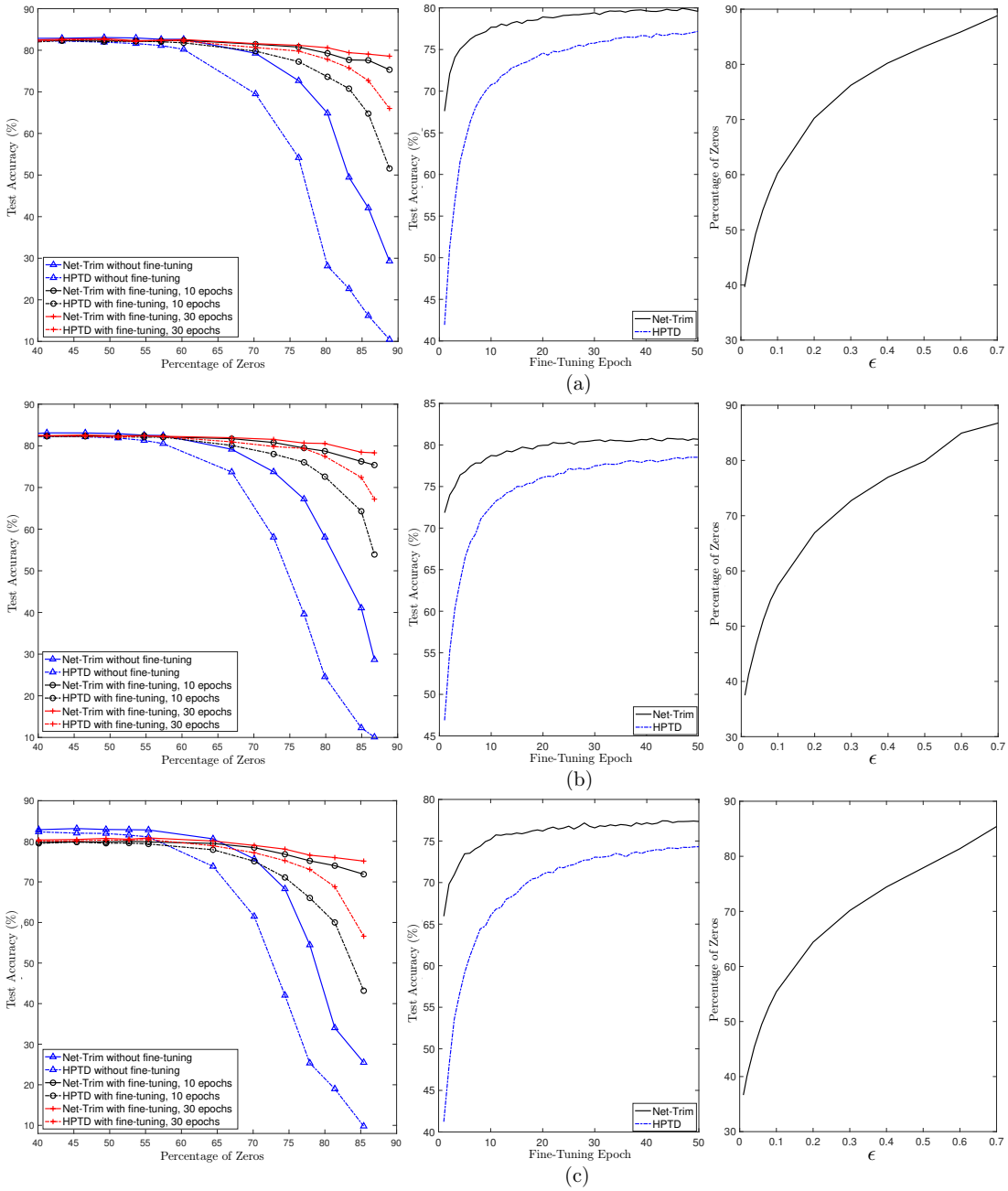


Figure 7: Similar comparison plots as in Figure 5 for CIFAR-10: (a) retraining of Net-Trim and HPTD performed using 25K samples; (b) retraining performed using 50K samples; (c) retraining performed using 75K samples

structures may be explored by adaptively selecting the objective. For instance, other than the Ridge and the elastic net penalties as regularizing tools, choosing $\mathcal{R}(\boldsymbol{U}) = \|\boldsymbol{U}\|_{2,1} = \sum_{n=1}^{N} \|\boldsymbol{U}_{n,:}\|$ can promote selection of a subset of the rows in $\boldsymbol{X}^{in}$, and act as a feature selection tool for each layer. Total variation, or rank penalizing objectives may also directly apply to network compression problems.

While in this paper we specifically focused on $\sigma = \texttt{ReLU}(\cdot)$ to exploit the convex formulation, in principal other forms of activation may be explored. Even if a convex (re)formulation is suboptimal or not possible, powerful tools from non-convex analysis would still allow us to have an understanding of when and how well programs of type (32) work. Clearly, the techniques used for such type of analysis might be initialization-sensitive, and different than those used in this paper.

Net-Trim can specifically become a useful tool when the number of training samples is limited. While overfitting is likely to happen in this situation, Net-Trim allows reducing the complexity of the models, yet maintaining the consistency with the original model. From a different perspective, Net-Trim may simplify the process of determining the network size, and for large networks that are trained with insufficient samples, employing Net-Trim can reduce the size of the models to an order matching the data.

8 Proofs

Before we start a detailed proof of the results, we would like to state two inequalities that will be frequently used throughout this section:

$$\forall\ \boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}: \qquad \left\|\boldsymbol{X}^+\right\|_F \leq \|\boldsymbol{X}\|_F\,, \tag{33}$$

$$\forall\ \boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{d_1 \times d_2}: \qquad \left\|\boldsymbol{X}^+ - \boldsymbol{Y}^+\right\|_F \leq \|\boldsymbol{X} - \boldsymbol{Y}\|_F\,. \tag{34}$$

The first inequality is straightforward to verify. To verify (34) we note that for all $x, y \in \mathbb{R}$:

$$x^+ = (x - y + y)^+ \leq (x-y)^+ + y^+ \leq |x-y| + y^+,$$

which is interchangeable in x and y, and yields $|x^+ - y^+| \leq |x-y|$.

8.1 Proof of Theorem 1

The central convex program (7) requires that for $\Omega = \text{supp}\ \boldsymbol{X}^{out}$:

$$\left\|\left(\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in} - \boldsymbol{X}^{out}\right)_\Omega\right\|_F \leq \epsilon, \quad \text{and} \quad \left(\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in}\right)_{\Omega^c} \leq 0. \tag{35}$$

As the first step, notice that for $\tilde{\boldsymbol{X}}^{out} = (\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in})^+$ one has

$$\begin{aligned}
\left\|\tilde{\boldsymbol{X}}^{out} - \boldsymbol{X}^{out}\right\|_F^2 &= \left\|\left(\tilde{\boldsymbol{X}}^{out} - \boldsymbol{X}^{out}\right)_\Omega\right\|_F^2 + \left\|\left(\tilde{\boldsymbol{X}}^{out} - \boldsymbol{X}^{out}\right)_{\Omega^c}\right\|_F^2 \\
&= \left\|\left(\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in}\right)_\Omega^+ - \boldsymbol{X}_\Omega^{out}\right\|_F^2 \\
&= \left\|\left(\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in}\right)_\Omega^+ - \left(\boldsymbol{X}^{out}\right)_\Omega^+\right\|_F^2 \\
&\leq \left\|\left(\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in} - \boldsymbol{X}^{out}\right)_\Omega\right\|_F^2 \\
&\leq \epsilon^2,
\end{aligned} \tag{36}$$

where the first inequality is thanks to (34). Now consider $\hat{\boldsymbol{X}}^{in}$ be any matrix such that

$$\left\|\hat{\boldsymbol{X}}^{in} - \boldsymbol{X}^{in}\right\|_F \leq \epsilon_{in},$$

then for $\hat{\boldsymbol{X}}^{out} = (\hat{\boldsymbol{W}}^\top \hat{\boldsymbol{X}}^{in})^+$ one has

$$\begin{aligned}
\left\|\hat{\boldsymbol{X}}^{out} - \boldsymbol{X}^{out}\right\|_F &\leq \left\|\hat{\boldsymbol{X}}^{out} - \tilde{\boldsymbol{X}}^{out}\right\|_F + \left\|\tilde{\boldsymbol{X}}^{out} - \boldsymbol{X}^{out}\right\|_F \\
&\leq \left\|\left(\hat{\boldsymbol{W}}^\top \hat{\boldsymbol{X}}^{in}\right)^+ - \left(\hat{\boldsymbol{W}}^\top \boldsymbol{X}^{in}\right)^+\right\|_F + \epsilon \\
&\leq \left\|\hat{\boldsymbol{W}}^\top \left(\hat{\boldsymbol{X}}^{in} - \boldsymbol{X}^{in}\right)\right\|_F + \epsilon \\
&\leq \left\|\hat{\boldsymbol{W}}\right\|_F \left\|\hat{\boldsymbol{X}}^{in} - \boldsymbol{X}^{in}\right\|_F + \epsilon \\
&\leq \epsilon_{in} + \epsilon. \qquad (37)
\end{aligned}$$

To present the last inequality we used the fact that

$$\left\|\hat{\boldsymbol{W}}\right\|_F \leq \left\|\hat{\boldsymbol{W}}\right\|_1 \leq \|\boldsymbol{W}\|_1 = 1.$$

We may now complete the proof via a simple induction. For the parallel scheme sketched in (8), inequality (36) implies that $\|\hat{\boldsymbol{X}}^{(1)} - \boldsymbol{X}^{(1)}\|_F \leq \epsilon_1$. Also, (36) requires that $\|\hat{\boldsymbol{X}}^{(\ell)} - \boldsymbol{X}^{(\ell)}\|_F \leq \epsilon_\ell$, and assuming that $\|\hat{\boldsymbol{X}}^{(\ell-1)} - \boldsymbol{X}^{(\ell-1)}\|_F \leq \sum_{j=1}^{\ell-1} \epsilon_j$, (37) yields

$$\left\|\hat{\boldsymbol{X}}^{(\ell)} - \boldsymbol{X}^{(\ell)}\right\|_F \leq \sum_{j=1}^{\ell} \epsilon_j.$$

8.2 Proof of Theorem 2

For the cascade scheme outlined in Algorithm 1, replacing the ℓ indexing with the *in*/*out* notation, the layer retraining takes place by addressing the convex program

$$\hat{\boldsymbol{W}} = \underset{\boldsymbol{U}}{\arg\min} \ \|\boldsymbol{U}\|_1 \quad \text{subject to} \quad \boldsymbol{U} \in \mathcal{C}_\epsilon\left(\hat{\boldsymbol{X}}^{in}, \boldsymbol{X}^{out}, \boldsymbol{W}^\top \hat{\boldsymbol{X}}^{in}\right), \qquad (38)$$

where $\hat{\boldsymbol{X}}^{in}$ is the retrained model input, $\boldsymbol{X}^{out} = (\boldsymbol{W}^\top \boldsymbol{X}^{in})^+$ is the initially trained model output, and for $\Omega = \text{supp } \boldsymbol{X}^{out}$,

$$\epsilon = \gamma \left\|\left(\boldsymbol{W}^\top \hat{\boldsymbol{X}}^{in} - \boldsymbol{X}^{out}\right)_\Omega\right\|_F.$$

The central convex program (38), hence requires that

$$\left\|\left(\hat{\boldsymbol{W}}^\top \hat{\boldsymbol{X}}^{in} - \boldsymbol{X}^{out}\right)_\Omega\right\|_F \leq \gamma \left\|\left(\boldsymbol{W}^\top \hat{\boldsymbol{X}}^{in} - \boldsymbol{X}^{out}\right)_\Omega\right\|_F, \qquad (39)$$

$$\left(\hat{\boldsymbol{W}}^\top \hat{\boldsymbol{X}}^{in}\right)_{\Omega^c} \leq \left(\boldsymbol{W}^\top \hat{\boldsymbol{X}}^{in}\right)_{\Omega^c}. \qquad (40)$$

For the output of the initial and retrained models, one has

$$\left\|\hat{\boldsymbol{X}}^{out}-\boldsymbol{X}^{out}\right\|_F^2=\left\|\left(\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{X}}^{in}\right)_\Omega^+-\boldsymbol{X}_\Omega^{out}\right\|_F^2+\left\|\left(\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{X}}^{in}\right)_{\Omega^c}^+\right\|_F^2. \tag{41}$$

For the first term in (41) thanks to (34) and (39), one has

$$\begin{aligned}\left\|\left(\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{X}}^{in}\right)_\Omega^+-\boldsymbol{X}_\Omega^{out}\right\|_F^2&=\left\|\left(\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{X}}^{in}\right)_\Omega^+-\left(\boldsymbol{X}^{out}\right)_\Omega^+\right\|_F^2\\&\le\left\|\left(\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{X}}^{in}-\boldsymbol{X}^{out}\right)_\Omega\right\|_F^2\\&\le\gamma^2\left\|\left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{in}-\boldsymbol{X}^{out}\right)_\Omega\right\|_F^2.\end{aligned} \tag{42}$$

The second term in (41) can also be bounded by

$$\begin{aligned}\left\|\left(\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{X}}^{in}\right)_{\Omega^c}^+\right\|_F^2\le\left\|\left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{in}\right)_{\Omega^c}^+\right\|_F^2&=\left\|\left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{in}\right)_{\Omega^c}^+-\left(\boldsymbol{W}^\top\boldsymbol{X}^{in}\right)_{\Omega^c}^+\right\|_F^2\\&\le\left\|\left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{in}-\boldsymbol{W}^\top\boldsymbol{X}^{in}\right)_{\Omega^c}\right\|_F^2.\end{aligned} \tag{43}$$

Using $\boldsymbol{X}_\Omega^{out}=(\boldsymbol{W}^\top\boldsymbol{X}^{in})_\Omega$, and applying the results of (42) and (43) to (41) yields

$$\begin{aligned}\left\|\hat{\boldsymbol{X}}^{out}-\boldsymbol{X}^{out}\right\|_F^2&\le\gamma^2\left\|\left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{in}-\boldsymbol{W}^\top\boldsymbol{X}^{in}\right)_\Omega\right\|_F^2+\left\|\left(\boldsymbol{W}^\top\hat{\boldsymbol{X}}^{in}-\boldsymbol{W}^\top\boldsymbol{X}^{in}\right)_{\Omega^c}\right\|_F^2\\&\le\gamma^2\left\|\boldsymbol{W}^\top\left(\hat{\boldsymbol{X}}^{in}-\boldsymbol{X}^{in}\right)\right\|_F^2\\&\le\gamma^2\left\|\hat{\boldsymbol{X}}^{in}-\boldsymbol{X}^{in}\right\|_F^2.\end{aligned} \tag{44}$$

In a cascade Net-Trim, the first layer goes through the standard retraining (10) with $\epsilon_1=\epsilon$, for which Theorem 1 warrants $\|\hat{\boldsymbol{X}}^{(1)}-\boldsymbol{X}^{(1)}\|_F\le\epsilon$. On the other hand, for $\ell\ge2$, (44) warrants $\|\hat{\boldsymbol{X}}^{(\ell)}-\boldsymbol{X}^{(\ell)}\|_F\le\gamma_\ell\|\hat{\boldsymbol{X}}^{(\ell-1)}-\boldsymbol{X}^{(\ell-1)}\|_F$, which together with the discrepancy of the first layer yield the advertised result in (14).

8.3 Proof of Theorem 3

It suffices to show the following statements:

- If $\boldsymbol{x}\in\mathbb{R}^N$ is a subgaussian vector, then for given $\boldsymbol{W}\in\mathbb{R}^{N\times M}$ and $\boldsymbol{b}\in\mathbb{R}^M$, the random vector $\boldsymbol{y}=\boldsymbol{W}^\top\boldsymbol{x}+\boldsymbol{b}$ is subgaussian.
- If $\boldsymbol{x}\in\mathbb{R}^N$ is a subgaussian vector, $\boldsymbol{y}=\boldsymbol{x}^+$ is also subgaussian.

We start by proving the first statement. The subgaussianity of $\boldsymbol{x}$ implies that there exists a constant κ such that for any given $\boldsymbol{\alpha}\in\mathbb{S}^{N-1}$:

$$\forall t\ge0\ :\ \mathbb{P}\left\{\left|\boldsymbol{\alpha}^\top\boldsymbol{x}\right|>t\right\}\le c\exp\left(-\frac{t^2}{\kappa^2}\right). \tag{45}$$

Now considering $\boldsymbol{\alpha} \in \mathbb{S}^{N-1}$ we have

$$\begin{aligned}\left|\boldsymbol{\alpha}^\top \boldsymbol{y}\right| &\leq \left|\left(\boldsymbol{W}\boldsymbol{\alpha}\right)^\top \boldsymbol{x}\right| + \left|\boldsymbol{\alpha}^\top \boldsymbol{b}\right| \\ &= \left\|\boldsymbol{W}\boldsymbol{\alpha}\right\| \left|\left(\frac{\boldsymbol{W}\boldsymbol{\alpha}}{\left\|\boldsymbol{W}\boldsymbol{\alpha}\right\|}\right)^\top \boldsymbol{x}\right| + \left|\boldsymbol{\alpha}^\top \boldsymbol{b}\right| \\ &\leq \left\|\boldsymbol{W}\right\| \left|\left(\frac{\boldsymbol{W}\boldsymbol{\alpha}}{\left\|\boldsymbol{W}\boldsymbol{\alpha}\right\|}\right)^\top \boldsymbol{x}\right| + \left\|\boldsymbol{b}\right\|,\end{aligned}$$

which immediately implies that

$$\forall \boldsymbol{\alpha} \in \mathbb{S}^{N_\ell - 1} : \left\{\boldsymbol{x} : \left|\boldsymbol{\alpha}^\top \left(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}\right)\right| > t\right\} \subseteq \left\{\boldsymbol{x} : \left\|\boldsymbol{W}\right\| \left|\left(\frac{\boldsymbol{W}\boldsymbol{\alpha}}{\left\|\boldsymbol{W}\boldsymbol{\alpha}\right\|}\right)^\top \boldsymbol{x}\right| + \left\|\boldsymbol{b}\right\| > t\right\}.$$

By the measure comparison we get

$$\begin{aligned}\mathbb{P}\left\{\left|\boldsymbol{\alpha}^\top \left(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}\right)\right| > t\right\} &\leq \mathbb{P}\left\{\left\|\boldsymbol{W}\right\| \left|\left(\frac{\boldsymbol{W}\boldsymbol{\alpha}}{\left\|\boldsymbol{W}\boldsymbol{\alpha}\right\|}\right)^\top \boldsymbol{x}\right| + \left\|\boldsymbol{b}\right\| > t\right\} \\ &= \mathbb{P}\left\{\left|\left(\frac{\boldsymbol{W}\boldsymbol{\alpha}}{\left\|\boldsymbol{W}\boldsymbol{\alpha}\right\|}\right)^\top \boldsymbol{x}\right| > \max\left(\frac{t - \left\|\boldsymbol{b}\right\|}{\left\|\boldsymbol{W}\right\|}, 0\right)\right\} \\ &\leq c \exp\left(-\max\left(\frac{t - \left\|\boldsymbol{b}\right\|}{\kappa \left\|\boldsymbol{W}\right\|}, 0\right)^2\right).\end{aligned}$$

Using Lemma 2 below, for $\kappa'^2 \geq \kappa^2 \left\|\boldsymbol{W}\right\|^2 + \left\|\boldsymbol{b}\right\|^2$ and $c' = ce$, the following should hold:

$$\forall t \geq 0 : \quad \mathbb{P}\left\{\left|\boldsymbol{\alpha}^\top \left(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}\right)\right| > t\right\} \leq c' \exp\left(-\frac{t^2}{\kappa'^2}\right),$$

which completes the first part of the proof.

Lemma 2 *Fix $a > 0$ and $b \geq 0$. Then, for $c^2 \geq a^2 + b^2$,*

$$\forall t \geq 0 \ : \ \exp\left(-\max\left(\frac{t-b}{a}, 0\right)^2\right) \leq \exp\left(1 - \frac{t^2}{c^2}\right). \tag{46}$$

Proof:
For $t \leq b$, the proposed conditions require $1 - t^2/c^2 > 0$, for which (46) automatically holds. In the case of $t > b$, to establish (46) it suffices to show that

$$\left(\frac{t-b}{a}\right)^2 \geq \frac{t^2}{c^2} - 1,$$

or in a simplified form

$$\left(1 - \frac{a^2}{c^2}\right) t^2 - 2bt + a^2 + b^2 \geq 0. \tag{47}$$

The discriminant of the quadratic expression in (47) is $4a^2\left((a^2+b^2)/c^2-1\right)$, which is never positive and the expression always takes an identical sign to $1-a^2/c^2$. $\square$

We next show the subgaussianity of $\boldsymbol{x}^+$ for a subgaussian random vector $\boldsymbol{x}\in\mathbb{R}^N$. For this purpose we introduce a constant κ' such that $\mathbb{E}\exp((\boldsymbol{\alpha}^\top\boldsymbol{x}^+)^2/\kappa'^2)\le e$ for all $\boldsymbol{\alpha}\in\mathbb{S}^{N-1}$. To this end, we first bound the magnitude of the marginals as

$$(\boldsymbol{\alpha}^\top\boldsymbol{x}^+)^2\le\|\boldsymbol{\alpha}\|^2\|\boldsymbol{x}^+\|^2\le\|\boldsymbol{x}\|^2. \tag{48}$$

We now make use of the following lemma borrowed from ([HKZ12]; see Theorem 2.1 and Remark 2.3 therein).

Lemma 3 *Let $\boldsymbol{A}\in\mathbb{R}^{N\times N}$ be a matrix and $\boldsymbol{\Sigma}=\boldsymbol{A}^\top\boldsymbol{A}$. Suppose $\boldsymbol{x}\in\mathbb{R}^N$ is a random vector such that for some $\boldsymbol{\mu}\in\mathbb{R}^N$ and $\kappa\ge 0$*

$$\forall\boldsymbol{\alpha}\in\mathbb{R}^N:\quad\mathbb{E}\exp\left(\boldsymbol{\alpha}^\top(\boldsymbol{x}-\boldsymbol{\mu})\right)\le\exp\left(\|\boldsymbol{\alpha}\|^2\kappa^2/2\right), \tag{49}$$

then for $\kappa'^2\ge 2\kappa^2\|\boldsymbol{\Sigma}\|$,

$$\mathbb{E}\exp\left(\frac{\|\boldsymbol{A}\boldsymbol{x}\|^2}{\kappa'^2}\right)\le\exp\left(\kappa^2 tr(\boldsymbol{\Sigma})\,\kappa'^{-2}+\frac{\kappa^4 tr\left(\boldsymbol{\Sigma}^2\right)\kappa'^{-4}+\|\boldsymbol{A}\boldsymbol{\mu}\|^2\kappa'^{-2}}{1-2\kappa^2\|\boldsymbol{\Sigma}\|\kappa'^{-2}}\right).$$

Condition (49) is technically a certificate of the subgaussianity of $\boldsymbol{x}$ ([HKZ12, Ver12]). Setting $\boldsymbol{A}$ in Lemma 3 to the identity matrix and making use of (48) verify that for $\boldsymbol{\alpha}\in\mathbb{S}^{N-1}$ and $\kappa'^2\ge 2\kappa^2$,

$$\mathbb{E}\exp\left(\frac{(\boldsymbol{\alpha}^\top\boldsymbol{x}^+)^2}{\kappa'^2}\right)\le\mathbb{E}\exp\left(\frac{\|\boldsymbol{x}\|^2}{\kappa'^2}\right)\le\exp\left(N\kappa^2\kappa'^{-2}+\frac{N\kappa^4\kappa'^{-4}+\|\boldsymbol{\mu}\|^2\kappa'^{-2}}{1-2\kappa^2\kappa'^{-2}}\right). \tag{50}$$

By selecting κ' sufficiently large, specifically $\kappa'\gtrsim\max(\sqrt{N}\kappa,\|\boldsymbol{\mu}\|)$, one can upper bound the right-hand side expression of (50) by e.

8.4 Proof of Theorem 4

With reference to (20), for $\boldsymbol{X}^{in}=[\boldsymbol{x}_1,\cdots,\boldsymbol{x}_P]\in\mathbb{R}^{N\times P}$ and

$$\Omega=\left\{p:x_p^{out}>0\right\}=\left\{p:\boldsymbol{x}_p^\top\boldsymbol{w}_0>0\right\},$$

we need to derive the conditions that $\boldsymbol{w}^*$ is the unique solution to

$$\min_{\boldsymbol{w}}\ \|\boldsymbol{w}\|_1\qquad\text{subject to}\qquad\begin{cases}\boldsymbol{X}_{:,\Omega}^{in\,\top}\boldsymbol{w}=\boldsymbol{x}_\Omega^{out}\\ \boldsymbol{X}_{:,\Omega^c}^{in\,\top}\boldsymbol{w}\preceq\boldsymbol{0}\end{cases}. \tag{51}$$

For general $\boldsymbol{X},\boldsymbol{X}_0\in\mathbb{R}^{N\times P}$ and $\Omega\subseteq\{1,\cdots,P\}$, consider the operator

$$\mathcal{T}_\Omega^{\boldsymbol{X}_0}\boldsymbol{X}\triangleq\boldsymbol{X}\mathrm{diag}\left(\mathbb{1}_\Omega\right)+\boldsymbol{X}_0,$$

where $\mathbb{1}_\Omega\in\mathbb{R}^P$ is the indicator of the set Ω. Simply, $\mathcal{T}_\Omega^{\boldsymbol{0}}\boldsymbol{X}$ replaces columns of $\boldsymbol{X}$ indexed by Ω^c with zero vectors. Exploiting the notion of minimum conic singular value, we first state a unique optimality result for (51), which generally holds regardless of the specific structure of $\boldsymbol{X}^{in}$.

Lemma 4 *Fix $\boldsymbol{\mu} \in \mathbb{R}^N$ and $\sigma \in \mathbb{R} - \{0\}$. Consider $\boldsymbol{w}^* \in \mathbb{R}^N$ to be a (sparse) feasible vector for* (51)*, and define the descent cone*

$$\mathcal{D} = \bigcup_{\tau>0} \left\{ \begin{pmatrix} \boldsymbol{y} \\ z \end{pmatrix} \in \mathbb{R}^{N+1} : \|\boldsymbol{w}^* + \tau \boldsymbol{y}\|_1 \leq \|\boldsymbol{w}^*\|_1 \right\}.$$

For $\boldsymbol{\Phi} = \mathcal{T}_\Omega^{-\boldsymbol{\mu}\mathbf{1}^\top} \boldsymbol{X}^{in}$, if

$$\inf \ \left\{ \left\| \begin{pmatrix} \boldsymbol{\Phi}^\top & \sigma \mathbf{1} \end{pmatrix} \boldsymbol{v} \right\| : \ \boldsymbol{v} \in \mathcal{D} \cap \mathbb{S}^N \right\} > 0, \tag{52}$$

then $\boldsymbol{w}^*$ is the unique solution to (51)*.*

Proof:
Showing the following three statements would complete the proof:

(S.1) If $\boldsymbol{w}^*$ is feasible for (51), then the pair $(\boldsymbol{w}^*, \sigma^{-1}\boldsymbol{\mu}^\top \boldsymbol{w}^*)$ is feasible for the convex program:

$$\underset{(\boldsymbol{w},u)}{\text{minimize}} \ \ \|\boldsymbol{w}\|_1 \qquad \text{subject to} \qquad \begin{pmatrix} \boldsymbol{\Phi}^\top & \sigma \mathbf{1} \end{pmatrix} \begin{pmatrix} \boldsymbol{w} \\ u \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_\Omega^{out} \\ \mathbf{0} \end{pmatrix}. \tag{53}$$

(S.2) For any pair $(\boldsymbol{w}^*, u^*)$ that is feasible for (53), if condition (52) holds, then $(\boldsymbol{w}^*, u^*)$ is the unique solution to (53).

(S.3) If $(\boldsymbol{w}^*, \sigma^{-1}\boldsymbol{\mu}^\top \boldsymbol{w}^*)$ is the unique solution to (53), then $\boldsymbol{w}^*$ is the unique solution to (51).

Based on the definition $\boldsymbol{\Phi} = \mathcal{T}_\Omega^{-\boldsymbol{\mu}\mathbf{1}^\top} \boldsymbol{X}^{in}$, verifying (S.1) is trivial. Claim (S.2) is a direct application of the minimum conic singular value result (e.g., see Prop. 2.2 of [CRPW12], or Prop. 2.6 of [Tro15]). To prove (S.3), suppose under the proposed assumption, (51) has a different solution $\hat{\boldsymbol{w}}$, where $\|\hat{\boldsymbol{w}}\|_1 \leq \|\boldsymbol{w}^*\|_1$. Then (S.1) requires $(\hat{\boldsymbol{w}}, \sigma^{-1}\boldsymbol{\mu}^\top \hat{\boldsymbol{w}})$ to be feasible for (53). However the objective for this feasible point is less than $\|\boldsymbol{w}^*\|_1$, which is in contradiction with $(\boldsymbol{w}^*, \sigma^{-1}\boldsymbol{\mu}^\top \boldsymbol{w}^*)$ being the unique solution to (53). □

Using Lemma 4 and the bowling scheme sketched in ([Tro15]), we continue with lower-bounding the minimum conic singular value away from zero, and relating the conditions to the number of samples, P.

To this end, we may look into the structure of the matrix $\boldsymbol{\Phi}$ in Lemma 4 as being populated with independent copies of $\boldsymbol{x}\mathbf{1}_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \boldsymbol{\mu}$ as the columns, and exploit the independence required for the bowling scheme. To assure centered columns, we choose $\boldsymbol{\mu} = \mathbb{E}\boldsymbol{x}\mathbf{1}_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}$, making columns of $\boldsymbol{\Phi} = [\boldsymbol{\varphi}_1, \cdots, \boldsymbol{\varphi}_P]$ independent copies of the centered subgaussian[6] random vector

$$\boldsymbol{\varphi} \triangleq \boldsymbol{x}\mathbf{1}_{\boldsymbol{w}_0^\top \boldsymbol{x}>0} - \mathbb{E}\boldsymbol{x}\mathbf{1}_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}.$$

For reasons that become apparent later in the proof, our arbitrary choice of σ in Lemma 4 is narrowed to

$$\sigma_0 \triangleq \sqrt{2}\|\boldsymbol{\varphi}\|_{\psi_2}.$$

In a random setting, to lower-bound the minimum conic singular value, we adapt the following result from ([Tro15]; Prop. 5.1), – or see Theorem 5.4 of ([Men14]) for the original statement.

Theorem 5 *Fix a set $E \subset \mathbb{R}^d$. Let $\boldsymbol{\phi}$ be a random vector on $\mathbb{R}^d$, and let $\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_P$ be independent copies of $\boldsymbol{\phi}$. For $\xi \geq 0$, suppose the marginal tail relation below holds:*

$$\inf_{\boldsymbol{v} \in E} \mathbb{P}\left\{ \left| \boldsymbol{\phi}^\top \boldsymbol{v} \right| \geq \xi \right\} \geq C_\xi > 0.$$

[6] Since $|\boldsymbol{\alpha}^\top \boldsymbol{x}\mathbf{1}_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}| \leq |\boldsymbol{\alpha}^\top \boldsymbol{x}|$ and for $t \geq 0$, $\mathbb{P}\{|\boldsymbol{\alpha}^\top \boldsymbol{x}\mathbf{1}_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}| > t\} \leq \mathbb{P}\{|\boldsymbol{\alpha}^\top \boldsymbol{x}| > t\}$, (15) confirms that $\boldsymbol{x}$ being subgaussian implies $\boldsymbol{x}\mathbf{1}_{\boldsymbol{w}_0^\top \boldsymbol{x}>0}$ to be subgaussian.

Let $\varepsilon_1, \cdots, \varepsilon_P$ be independent Rademacher random variables, independent from everything else, and define the mean empirical width of the set E:

$$\mathcal{W}_P\left(E; \boldsymbol{\phi}\right) \triangleq \mathbb{E}\ \sup_{\boldsymbol{v}\in E} \langle \boldsymbol{h}, \boldsymbol{v}\rangle, \quad \textit{where} \quad \boldsymbol{h} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \varepsilon_p \boldsymbol{\phi}_p. \tag{54}$$

Then, for any $\xi > 0$ and $t > 0$, with probability at least $1 - \exp(-t^2/2)$:

$$\inf_{\boldsymbol{v}\in E} \left(\sum_{p=1}^{P} \left(\boldsymbol{\phi}_p^\top \boldsymbol{v} \right)^2 \right)^{\frac{1}{2}} \geq \frac{\xi}{2} C_\xi \sqrt{P} - 2\mathcal{W}_P(E; \boldsymbol{\phi}) - \frac{\xi}{2} t. \tag{55}$$

For a more compact (and inline) notation, we use the following notation for the concatenation of a vector $\boldsymbol{w}$ and a scalar u,

$$\boldsymbol{w}^\frown u \triangleq \begin{pmatrix} \boldsymbol{w} \\ u \end{pmatrix}.$$

Also, for a given objective and point $\boldsymbol{v}_0 \in \mathbb{R}^d$, we denote the descent cone by

$$\mathcal{D}_{\boldsymbol{v}}\left(f(\boldsymbol{v}); \boldsymbol{v}_0\right) = \bigcup_{\tau>0} \left\{ \boldsymbol{y} \in \mathbb{R}^d : f(\boldsymbol{v}_0 + \tau \boldsymbol{y}) \leq f(\boldsymbol{v}_0) \right\}.$$

To show that condition (52) holds for the prescribed s-sparse vector $\boldsymbol{w}^*$, we will show that for sufficiently large P, the right-hand side expression in (55) can be bounded away from zero. To apply Theorem 5 to our problem in (52), the random vector $\boldsymbol{\phi}$ and the set E to consider are

$$\boldsymbol{\phi} = \boldsymbol{\varphi}^\frown \sigma_0, \quad \text{and} \quad E = \mathcal{D}_{\boldsymbol{w}^\frown u}\left(\|\boldsymbol{w}\|_1; \boldsymbol{w}^{*\frown} u^*\right) \cap \mathbb{S}^N,$$

where

$$\begin{aligned} \mathcal{D}_{\boldsymbol{w}^\frown u}\left(\|\boldsymbol{w}\|_1; \boldsymbol{w}^{*\frown} u^*\right) &= \bigcup_{\tau>0} \left\{ \boldsymbol{y}^\frown z \in \mathbb{R}^{N+1} : \|\boldsymbol{w}^* + \tau \boldsymbol{y}\|_1 \leq \|\boldsymbol{w}^*\|_1 \right\} \\ &= \mathcal{D}_{\boldsymbol{w}}\left(\|\boldsymbol{w}\|_1; \boldsymbol{w}^*\right) \times \mathbb{R}, \end{aligned}$$

and

$$u^* = \sigma_0^{-1} \left(\mathbb{E} \boldsymbol{x} 1_{\boldsymbol{w}_0^\top \boldsymbol{x} > 0} \right)^\top \boldsymbol{w}^*.$$

Note that in the formulation above, $\mathcal{D}_{\boldsymbol{w}^\frown u}(.;.) \subset \mathbb{R}^{N+1}$, while $\mathcal{D}_{\boldsymbol{w}}(.;.) \subset \mathbb{R}^N$. The remainder of the proof focuses on bounding the contributing terms on the right-hand side expression of (55).

8.4.1 Bounding the Mean Empirical Width

In this section of the proof, we aim to upper-bound

$$\mathcal{W}_P\left(\mathcal{D}_{\boldsymbol{w}^\frown u}\left(\|\boldsymbol{w}\|_1; \boldsymbol{w}^{*\frown} u^*\right) \cap \mathbb{S}^N; \boldsymbol{\varphi}^\frown \sigma_0 \right),$$

where following the formulation in (54) we have

$$\boldsymbol{h} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \varepsilon_p \boldsymbol{\varphi}_p^\frown \sigma_0 = \underbrace{\begin{pmatrix} \mathbf{0} \\ \frac{\sigma_0}{\sqrt{P}} \sum_{p=1}^{P} \varepsilon_p \end{pmatrix}}_{\boldsymbol{h}_u} + \underbrace{\begin{pmatrix} \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \varepsilon_p \boldsymbol{\varphi}_p \\ 0 \end{pmatrix}}_{\boldsymbol{h}_{\boldsymbol{w}}^\frown 0}.$$

Using the compact notations

$$\mathcal{K}_{\boldsymbol{w}^*,u^*} = \mathcal{D}_{\boldsymbol{w}^\frown u}\left(\|\boldsymbol{w}\|_1; \boldsymbol{w}^{*\frown} u^*\right), \quad \text{and} \quad \mathcal{K}_{\boldsymbol{w}^*} = \mathcal{D}_{\boldsymbol{w}}\left(\|\boldsymbol{w}\|_1; \boldsymbol{w}^*\right),$$

ones has

$$\begin{aligned}\mathcal{W}_P\left(\mathcal{K}_{\boldsymbol{w}^*,u^*} \cap \mathbb{S}^N; \boldsymbol{\varphi}^\frown \sigma_0\right) &= \mathbb{E} \sup_{\boldsymbol{v}\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N} \langle \boldsymbol{h}, \boldsymbol{v}\rangle \\ &\leq \mathbb{E} \sup_{\boldsymbol{v}\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N} \langle \boldsymbol{h}_u, \boldsymbol{v}\rangle + \mathbb{E} \sup_{\boldsymbol{v}\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N} \langle \boldsymbol{h}_{\boldsymbol{w}}^\frown 0, \boldsymbol{v}\rangle. \end{aligned} \tag{56}$$

For the first term in (56) note that

$$\begin{aligned}\mathbb{E} \sup_{\boldsymbol{w}^\frown u\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^n} \langle \boldsymbol{h}_u, \boldsymbol{w}^\frown u\rangle &= \mathbb{E} \sup_{\boldsymbol{w}^\frown u\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N} \left(\frac{\sigma_0}{\sqrt{P}}\sum_{p=1}^P \varepsilon_p\right) u \\ &= \mathbb{E}\left|\frac{\sigma_0}{\sqrt{P}}\sum_{p=1}^P \varepsilon_p\right| \\ &\leq \frac{\sigma_0}{\sqrt{P}}\left(\mathbb{E}\left(\sum_{p=1}^P \varepsilon_p\right)^2\right)^{\frac{1}{2}} \\ &= \sigma_0. \end{aligned} \tag{57}$$

To bound the second term in (56), we proceed by first showing that for any fixed $\boldsymbol{h}_{\boldsymbol{w}} \in \mathbb{R}^N$,

$$\sup_{\boldsymbol{w}^\frown u\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N} \langle \boldsymbol{h}_{\boldsymbol{w}}^\frown 0, \boldsymbol{w}^\frown u\rangle = \sup_{\boldsymbol{w}\in\mathcal{K}_{\boldsymbol{w}^*}\cap\mathbb{S}^{N-1}} \langle \boldsymbol{h}_{\boldsymbol{w}}, \boldsymbol{w}\rangle. \tag{58}$$

For this purpose only the following two cases need to be considered:

– **case 1:** $\langle \boldsymbol{h}_{\boldsymbol{w}}, \boldsymbol{w}\rangle \leq 0, \forall \boldsymbol{w} \in \mathcal{K}_{\boldsymbol{w}^*}$.

In this case the supremum value for both sides of (58) is zero, which may be attained by picking $\boldsymbol{w} = \boldsymbol{0}$ and $u = 1$.

– **case 2:** $\exists \boldsymbol{w} \in \mathcal{K}_{\boldsymbol{w}^*}$, such that $\langle \boldsymbol{h}_{\boldsymbol{w}}, \boldsymbol{w}\rangle > 0$.

To show the equality in this case, we only need to show that if $\hat{\boldsymbol{v}} = \hat{\boldsymbol{w}}^\frown \hat{u}$ is a point at which the (positive) supremum is attained, i.e.,

$$\sup_{\boldsymbol{v}\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N} \langle \boldsymbol{h}_{\boldsymbol{w}}^\frown 0, \boldsymbol{v}\rangle = \langle \boldsymbol{h}_{\boldsymbol{w}}^\frown 0, \hat{\boldsymbol{v}}\rangle = \langle \boldsymbol{h}_{\boldsymbol{w}}, \hat{\boldsymbol{w}}\rangle,$$

then we must have $\hat{u} = 0$. If $\hat{u} \neq 0$, then the condition $\hat{\boldsymbol{v}} \in \mathbb{S}^N$ requires that $\|\hat{\boldsymbol{w}}\| < 1$. In this case the alternative feasible point $\tilde{\boldsymbol{v}} = \|\hat{\boldsymbol{w}}\|^{-1}\hat{\boldsymbol{w}}^\frown 0$ produces a greater inner product:

$$\langle \boldsymbol{h}_{\boldsymbol{w}}^\frown 0, \tilde{\boldsymbol{v}}\rangle = \frac{1}{\|\hat{\boldsymbol{w}}\|}\langle \boldsymbol{h}_{\boldsymbol{w}}, \hat{\boldsymbol{w}}\rangle > \langle \boldsymbol{h}_{\boldsymbol{w}}, \hat{\boldsymbol{w}}\rangle,$$

which cannot be possible. Therefore $\hat{u} = 0$, and for both sides of (58) the supremum value is $\langle \boldsymbol{h}_{\boldsymbol{w}}, \hat{\boldsymbol{w}}\rangle$. Combining cases 1 and 2 establishes the claim in (58).

Now, employing (58) and (57) in (56) certifies that for

$$\boldsymbol{h}_{\boldsymbol{w}} = \frac{1}{\sqrt{P}}\sum_{p=1}^P \varepsilon_p \boldsymbol{\varphi}_p,$$

and some absolute constant C, one has

$$\mathcal{W}_P\left(\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N;\boldsymbol{\varphi}\frown\sigma_0\right)\leq\sigma_0+\mathbb{E}\sup_{\boldsymbol{w}\in\mathcal{K}_{\boldsymbol{w}^*}\cap\mathbb{S}^{N-1}}\langle\boldsymbol{h}_{\boldsymbol{w}},\boldsymbol{w}\rangle$$
$$\leq C\|\boldsymbol{\varphi}\|_{\psi_2}\left(\sqrt{s\log\left(\frac{N}{s}\right)+s}+1\right). \tag{59}$$

The last line in (59) is thanks to the following inequality (see §6.6 of [Tro15]), which relates the mean empirical width of a centered subgaussian random vector $\boldsymbol{\varphi}$ to the Gaussian width:

$$\mathbb{E}\sup_{\boldsymbol{w}\in\mathcal{K}_{\boldsymbol{w}^*}\cap\mathbb{S}^{N-1}}\langle\boldsymbol{h}_{\boldsymbol{w}},\boldsymbol{w}\rangle\lesssim\|\boldsymbol{\varphi}\|_{\psi_2}\mathbb{E}\sup_{\substack{\boldsymbol{w}\in\mathcal{K}_{\boldsymbol{w}^*}\cap\mathbb{S}^{N-1}\\ \boldsymbol{g}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}}\langle\boldsymbol{g},\boldsymbol{w}\rangle\lesssim\|\boldsymbol{\varphi}\|_{\psi_2}\sqrt{s\log\left(\frac{N}{s}\right)+s}.$$

8.4.2 Relating the Marginal Tail Bound and the Virtual Covariance

As the next step in lower-bounding the right-hand side expression in (55), noting that

$$\inf_{\boldsymbol{v}\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N}\mathbb{P}\left\{\left|\boldsymbol{v}^\top\boldsymbol{\varphi}\frown\sigma_0\right|\geq\xi\right\}\geq\inf_{\boldsymbol{v}\in\mathbb{S}^N}\mathbb{P}\left\{\left|\boldsymbol{v}^\top\boldsymbol{\varphi}\frown\sigma_0\right|\geq\xi\right\}, \tag{60}$$

in this section we focus on lower bounding the right-hand side expression in (60) in terms of $\|\boldsymbol{\varphi}\|_{\psi_2}$ and the minimum eigenvalue of the virtual covariance matrix. To this end, using the notation

$$\tilde{\lambda}_{\min}\triangleq\lambda_{\min}\left(\operatorname{cov}\left(\boldsymbol{v}\right)\right)=\lambda_{\min}\left(\mathbb{E}\boldsymbol{\varphi}\boldsymbol{\varphi}^\top\right),$$

one has

$$\mathbb{E}\boldsymbol{\varphi}\frown\sigma_0\boldsymbol{\varphi}\frown\sigma_0{}^\top=\begin{pmatrix}\mathbb{E}\boldsymbol{\varphi}\boldsymbol{\varphi}^\top & \boldsymbol{0}\\ \boldsymbol{0}^\top & \sigma_0^2\end{pmatrix}\succeq\min\left(\tilde{\lambda}_{\min},\sigma_0^2\right)\boldsymbol{I}. \tag{61}$$

On the other hand, from the subgaussian properties of $\boldsymbol{\varphi}$ we have

$$\|\boldsymbol{\varphi}\|_{\psi_2}\geq\sup_{\boldsymbol{w}\in\mathbb{S}^{N-1}}2^{-\frac{1}{2}}\left(\mathbb{E}\left|\boldsymbol{w}^\top\boldsymbol{\varphi}\right|^2\right)^{\frac{1}{2}}\geq\inf_{\boldsymbol{w}\in\mathbb{S}^{N-1}}2^{-\frac{1}{2}}\left(\mathbb{E}\left|\boldsymbol{w}^\top\boldsymbol{\varphi}\right|^2\right)^{\frac{1}{2}}=\sqrt{\frac{\tilde{\lambda}_{\min}}{2}},$$

which simply implies that $\sigma_0^2\geq\tilde{\lambda}_{\min}$ and combining with (61) yields

$$\inf_{\boldsymbol{v}\in\mathbb{S}^N}\mathbb{E}\left|\boldsymbol{v}^\top\boldsymbol{\varphi}\frown\sigma_0\right|^2\geq\min\left(\tilde{\lambda}_{\min},\sigma_0^2\right)=\tilde{\lambda}_{\min}. \tag{62}$$

Considering a positive random variable χ and a fixed $\xi\geq0$, we can derive a variant of the Paley-Zygmund inequality by writing $\chi^2=\chi^2 1_{\{\chi^2<\xi^2\}}+\chi^2 1_{\{\chi^2\geq\xi^2\}}$, which using the Hölder's inequality naturally yields

$$\mathbb{E}\chi^2\leq\xi^2+\left(\mathbb{P}\left\{\chi\geq\xi\right\}\right)^{\frac{\beta}{1+\beta}}\left(\mathbb{E}\chi^{2(1+\beta)}\right)^{\frac{1}{1+\beta}},\quad\beta>0.$$

Subsequently, selecting $\xi\in[0,\sqrt{\tilde{\lambda}_{\min}}]$ warrants that

$$\forall\boldsymbol{v}\in\mathbb{S}^N:\qquad\mathbb{P}\left\{\left|\boldsymbol{v}^\top\boldsymbol{\varphi}\frown\sigma_0\right|\geq\xi\right\}\geq\left(\frac{\tilde{\lambda}_{\min}-\xi^2}{\left(\mathbb{E}\left|\boldsymbol{v}^\top\boldsymbol{\varphi}\frown\sigma_0\right|^{2(1+\beta)}\right)^{\frac{1}{\beta+1}}}\right)^{1+\frac{1}{\beta}}.$$

We can also use the subgaussian properties of $\boldsymbol{\varphi}$ to bound the denominator as follows

$$\begin{aligned}\forall \boldsymbol{w}\frown u \in \mathbb{S}^N,\ \alpha \geq 1: \quad \left(\mathbb{E}\left|\boldsymbol{w}\frown u^\top \boldsymbol{\varphi}\frown \sigma_0\right|^\alpha\right)^{\frac{1}{\alpha}} &= \left(\mathbb{E}\left|\boldsymbol{w}^\top\boldsymbol{\varphi}+\sigma_0 u\right|^\alpha\right)^{\frac{1}{\alpha}} \\ &\leq \left(\mathbb{E}\left|\boldsymbol{w}^\top\boldsymbol{\varphi}\right|^\alpha\right)^{\frac{1}{\alpha}} + \sigma_0|u| \\ &= \|\boldsymbol{w}\|\left(\mathbb{E}\left|\frac{\boldsymbol{w}^\top}{\|\boldsymbol{w}\|}\boldsymbol{\varphi}\right|^\alpha\right)^{\frac{1}{\alpha}} + \sigma_0|u| \\ &\leq \sqrt{\alpha}\|\boldsymbol{\varphi}\|_{\psi_2}\|\boldsymbol{w}\| + \sqrt{2}\|\boldsymbol{\varphi}\|_{\psi_2}|u| \\ &\leq \sqrt{\alpha+2}\|\boldsymbol{\varphi}\|_{\psi_2},\end{aligned}$$

where the first inequality is a direct application of the Minkowski inequality, the second inequality uses the subgaussian definition (17) and the last bound is thanks to the Cauchy-Schwarz inequality. As a result for $\xi \in [0, \sqrt{\tilde{\lambda}_{\min}}]$

$$\inf_{\boldsymbol{v}\in\mathbb{S}^N} \mathbb{P}\left\{\left|\boldsymbol{v}^\top\boldsymbol{\varphi}\frown\sigma_0\right| \geq \xi\right\} \geq \left(\frac{\tilde{\lambda}_{\min}-\xi^2}{2(2+\beta)\|\boldsymbol{\varphi}\|_{\psi_2}^2}\right)^{1+\frac{1}{\beta}}. \tag{63}$$

8.4.3 Combining the Bounds

We can now combine the bounds (59) and (63), and use Theorem 5 to state that with probability at least $1-\exp(-t^2/2)$:

$$\begin{aligned}\inf_{\boldsymbol{v}\in\mathcal{K}_{\boldsymbol{w}^*,u^*}\cap\mathbb{S}^N} \left(\sum_{p=1}^P \left(\boldsymbol{\varphi}_p\frown\sigma_0^\top\boldsymbol{v}\right)^2\right)^{\frac{1}{2}} &\geq \frac{\xi}{2}\sqrt{P}\left(\frac{\tilde{\lambda}_{\min}-\xi^2}{2(2+\beta)\|\boldsymbol{\varphi}\|_{\psi_2}^2}\right)^{1+\frac{1}{\beta}} \\ &\quad - 2C\|\boldsymbol{\varphi}\|_{\psi_2}\left(\sqrt{s\log\left(\frac{N}{s}\right)+s}+1\right) - \frac{\xi}{2}t.\end{aligned}$$

Selecting $\xi = \sqrt{\tilde{\lambda}_{\min}}/3$ would bound the expression above away from zero, as long as

$$P \geq \frac{36\left(9\left(1+\frac{\beta}{2}\right)\|\boldsymbol{\varphi}\|_{\psi_2}^2\right)^{2+\frac{2}{\beta}}}{\tilde{\lambda}_{\min}\left(2\tilde{\lambda}_{\min}\right)^{2+\frac{2}{\beta}}}\left(2C\|\boldsymbol{\varphi}\|_{\psi_2}\left(\sqrt{s\log\left(\frac{N}{s}\right)+s}+1\right)+\frac{\sqrt{\tilde{\lambda}_{\min}}}{6}t\right)^2. \tag{64}$$

Noting that $\tilde{\lambda}_{\min} \leq 2\|\boldsymbol{\varphi}\|_{\psi_2}^2$ and using the basic inequality $(a+b)^2 \leq 2a^2+2b^2$ twice yields

$$\left(2C\|\boldsymbol{\varphi}\|_{\psi_2}\left(\sqrt{s\log\left(\frac{N}{s}\right)+s}+1\right)+\frac{\sqrt{\tilde{\lambda}_{\min}}}{6}t\right)^2 \lesssim \|\boldsymbol{\varphi}\|_{\psi_2}^2\left(s\log\left(\frac{N}{s}\right)+s+1+\frac{t^2}{72C^2}\right).$$

Also, since for $\beta \geq 1$,

$$\left(1+\frac{\beta}{2}\right)^{\frac{2}{\beta}} \lesssim 1,$$

one has

$$\frac{36\left(9\left(1+\frac{\beta}{2}\right)\|\boldsymbol{\varphi}\|_{\psi_2}^2\right)^{2+\frac{2}{\beta}}}{\tilde{\lambda}_{\min}\left(2\tilde{\lambda}_{\min}\right)^{2+\frac{2}{\beta}}} \lesssim \left(1+\frac{\beta}{2}\right)^2 \frac{\|\boldsymbol{\varphi}\|_{\psi_2}^{4+\frac{4}{\beta}}}{\tilde{\lambda}_{\min}^{3+\frac{2}{\beta}}}.$$

Therefore, the desired condition in (52) holds, as long as

$$P \gtrsim \left(1+\frac{\beta}{2}\right)^2 \frac{\|\boldsymbol{\varphi}\|_{\psi_2}^{6+\frac{4}{\beta}}}{\tilde{\lambda}_{\min}^{3+\frac{2}{\beta}}} \left(s\log\left(\frac{N}{s}\right)+s+1+\frac{t^2}{72C^2}\right).$$

Finally, setting $\beta' = \beta/2$ and $t' = t^2/(72C^2)$ yields the advertised claim in (21).

8.5 Proof of Lemma 1

We follow a similar line of argument as §5.3.1 of ([LLC17]). To evaluate

$$I = \mathbb{E}_{\boldsymbol{x}}\; g\left(\boldsymbol{\alpha}^\top \boldsymbol{x}\right) 1_{\boldsymbol{\beta}^\top \boldsymbol{x}>0} = \frac{1}{(2\pi)^{\frac{N}{2}}} \int_{\boldsymbol{\beta}^\top \boldsymbol{x}>0} g\left(\boldsymbol{\alpha}^\top \boldsymbol{x}\right) \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2}\right) \mathrm{d}\boldsymbol{x},$$

we assume that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are not aligned (for the aligned case a similar procedure applies to merely $\boldsymbol{\alpha}$). We consider the unitary matrix $\boldsymbol{E} = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_N]$, where

$$\boldsymbol{e}_1 = \boldsymbol{\beta}, \qquad \boldsymbol{e}_2 = \frac{\boldsymbol{\alpha} - \left(\boldsymbol{\beta}^\top \boldsymbol{\alpha}\right)\boldsymbol{\beta}}{\sqrt{1-\left(\boldsymbol{\beta}^\top \boldsymbol{\alpha}\right)^2}},$$

and $\boldsymbol{e}_3, \cdots \boldsymbol{e}_N$ are any completion of the ortho-basis. Setting $\boldsymbol{x} = \boldsymbol{E}\boldsymbol{z}$ yields $\boldsymbol{\beta}^\top \boldsymbol{x} = z_1$ and

$$\boldsymbol{\alpha}^\top \boldsymbol{x} = \left(\boldsymbol{\beta}^\top \boldsymbol{\alpha}\right) z_1 + \sqrt{1-\left(\boldsymbol{\beta}^\top \boldsymbol{\alpha}\right)^2}\, z_2.$$

Taking into account the injectivity of the linear map $\boldsymbol{E}$, we can reformulate the integral in the $\boldsymbol{z}$-domain as (see Theorem 263D of [Fre00] for the formal statement)

$$I = \frac{1}{2\pi} \int_{z_1>0} g\left(\left(\boldsymbol{\beta}^\top \boldsymbol{\alpha}\right) z_1 + \sqrt{1-\left(\boldsymbol{\beta}^\top \boldsymbol{\alpha}\right)^2}\, z_2\right) \exp\left(-\frac{z_1^2+z_2^2}{2}\right) \mathrm{d}z_1 \mathrm{d}z_2.$$

Acknowledgement: A. Aghasi would like to thank Roman Vershynin and Richard Kueng for the insightful suggestions and communications.

References

[AANR17] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. In *Advances in Neural Information Processing Systems 31*, pages 3180–3189. Curran Associates, Inc., 2017.

[BPC+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[Can08] E. Candes. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

[CP11] E. Candes and Y. Plan. A probabilistic and ripless theory of compressed sensing. *IEEE Transactions on Information Theory*, 57(11):7235–7254, 2011.

[CRPW12] V. Chandrasekaran, B. Recht, P. Parrilo, and A. Willsky. The convex geometry of linear inverse problems. *Foundations of Computational mathematics*, 12(6):805–849, 2012.

[CRT06] E. Candes, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[CWT+15] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

[Fre00] D.H. Fremlin. Measure theory. *Torres Fremlin*, 2, 2000.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[GJP95] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

[Gro11] D. Gross. Recovering low-rank matrices from few coefficients in any basis. *IEEE Transactions on Information Theory*, 57(3):1548–1566, 2011.

[HK70] A.E. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[HKZ12] D. Hsu, S. Kakade, and T. Zhang. A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17, 2012.

[HMD15] S. Han, H. Mao, and W. J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[HMD16] S. Han, H. Mao, and W. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[HPTD15] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[IS15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[KM15] V. Koltchinskii and S. Mendelson. Bounding the smallest singular value of a random matrix without concentration. *International Mathematics Research Notices*, 2015(23):12991–13008, 2015.

[Kri10] A. Krizhevsky. Convolutional deep belief networks on cifar-10. 2010.

[KSH12] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LLC17] C. Louart, Z. Liao, and R. Couillet. A random matrix approach to neural networks. *arXiv preprint arXiv:1702.05419*, 2017.

[Men14] S. Mendelson. Learning without concentration. In *Conference on Learning Theory*, pages 25–39, 2014.

[Men17] S. Mendelson. Learning without concentration for general loss functions. *Probability Theory and Related Fields*, 2017.

[NH92] S. Nowlan and G. Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992.

[Sch15] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[Tro15] J. Tropp. Convex recovery of a structured signal from independent random linear measurements. In *Sampling Theory, a Renaissance*, pages 67–101. Springer, 2015.

[vdVW96] A.W. van der Vaart and J. Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer Science & Business Media, 1996.

[Ver12] R. Vershynin. *Introduction to the non-asymptotic analysis of random matrices*, page 210–268. Cambridge University Press, 2012.

[WZZ+16] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.