LAYER SPARSITY IN NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparsity has become popular in machine learning, because it can save computational resources, facilitate interpretations, and prevent overfitting. In this paper, we discuss sparsity in the framework of neural networks. In particular, we formulate a new notion of sparsity that concerns the networks' layers and, therefore, aligns particularly well with the current trend toward deep networks. We call this notion layer sparsity. We then introduce corresponding regularization and refitting schemes that can complement standard deep-learning pipelines to generate more compact and accurate networks.

1 INTRODUCTION

The number of layers and the number of nodes in each layer are arguably among the most fundamental parameters of neural networks. But specifying these parameters can be challenging: deep and wide networks, that is, networks with many layers and nodes, can describe data in astounding detail, but they are also prone to overfitting and require large memory, CPU, energy, and so forth. The resource requirements can be particularly problematic for real-time applications or applications on fitness trackers and other wearables, whose popularity has surged in recent years. A promising approach to meet these challenges is to fit networks sizes adaptively, that is, to allow for many layers and nodes in principle, but to ensure that the final network is "simple" in that it has a small number of connections, nodes, or layers (Changpinyo et al., 2017; Han et al., 2016; Kim et al., 2016; Liu et al., 2015; Wen et al., 2016).

Popular ways to fit such simple and compact networks include successively augmenting small networks (Ash, 1989; Bello, 1992), pruning large networks (Simonyan & Zisserman, 2015), or explicit sparsity-inducing regularization of the weight matrices, which we focus on here. An example is the ℓ_1-norm, which can reduce the number of connections. Another example is the ℓ_1-norm grouped over the rows of the weight matrices, which can reduce the number of nodes. It has been shown that such regularizers can indeed produce networks that are both accurate and yet have a small number of nodes and connections either in the first layer (Feng & Simon, 2017) or overall (Alvarez & Salzmann, 2016; Liu et al., 2015; Scardapane et al., 2017). Such sparsity-inducing regularizers also have a long-standing tradition and thorough theoretical underpinning in statistics (Hastie et al., 2015).

But while sparsity on the level of connections and nodes has been studied in some detail, sparsity on the level of layers is much less understood. This lack of understanding contrasts the current trend to deep network architectures, which is supported by state-of-the-art performances of deep networks (LeCun et al., 2015; Schmidhuber, 2015), recent approximation theory for ReLU activation networks (Liang & Srikant, 2016; Telgarsky, 2016; Yarotsky, 2017), and recent statistical theory (Golowich et al., 2017; Kohler & Langer, 2019; Taheri et al., 2020). Hence, a better understanding of sparsity on the level of layers seems to be in order.

Therefore, we discuss in this paper sparsity with a special emphasis on the networks' layers. Our key observation is that for typical activation functions such as ReLU, a layer can be removed if all its parameter values are non-negative. We leverage this observation in the development of a new regularizer that specifically targets sparsity on the level of layers, and we show that this regularizer can lead to more compact and more accurate networks.

Our three main contributions are:

1. We introduce a new notion of sparsity that we call *layer sparsity*.
2. We introduce a corresponding regularizer that can reduce network sizes.

3. We introduce an additional refitting step that can further improve prediction accuracies.

In Section 2, we specify our framework, discuss different notions of sparsity, and introduce our refitting scheme. In Section 3, we establish a numerical proof of concept. In Section 4, we conclude with a discussion. In Appendix A, we establish the proof for our theoretical result.

2 Sparsity in Neural Networks

We first state our framework, then discuss different notions of sparsity, and finally introduce a refitting scheme.

2.1 Mathematical Framework

To fix ideas, we consider feedforward neural networks that model data according to

$$y_i = \boldsymbol{f}^1\Big[W^1\boldsymbol{f}^2\big[...\boldsymbol{f}^l[W^l\boldsymbol{x}_i]\big]\Big] + u_i \ , \tag{1}$$

where $i \in \{1,\dots,n\}$ indexes the n different samples, $y_i \in \mathbb{R}$ is the output, $\boldsymbol{x}_i \in \mathbb{R}^d$ is the corresponding input with d the input dimension, l is the number of layers, $W^j \in \mathbb{R}^{p_j \times p_{j+1}}$ for $j \in \{1,\dots,l\}$ are the weight matrices with $p_1 = 1$ and $p_{l+1} = d$, $\boldsymbol{f}^j : \mathbb{R}^{p_j} \to \mathbb{R}^{p_j}$ for $j \in \{1,\dots,l\}$ are the activation functions, and $u_i \in \mathbb{R}$ is the random noise.

We summarize the parameters in $\boldsymbol{W} := (W^1,\dots,W^l) \in \mathcal{V} := \{\boldsymbol{V} = (V^1,\dots,V^l) : V^j \in \mathbb{R}^{p_j \times p_{j+1}}\}$, and we write for ease of notation

$$f_{\boldsymbol{V}}[\boldsymbol{x}_i] := \boldsymbol{f}^1\Big[V^1\boldsymbol{f}^2\big[...\boldsymbol{f}^l[V^l\boldsymbol{x}_i]\big]\Big] \tag{2}$$

for $\boldsymbol{V} \in \mathcal{V}$.

Neural networks are usually fitted based on regularized estimators in Lagrange

$$\widehat{\boldsymbol{W}} \in \underset{\boldsymbol{V}\in\mathcal{V}}{\operatorname{argmin}}\big\{\mathrm{DataFit}[y_1,\dots,y_n,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n] + h[\boldsymbol{V}]\big\} \tag{3}$$

or constraint form

$$\widehat{\boldsymbol{W}} \in \underset{\substack{\boldsymbol{V}\in\mathcal{V}\\ h[\boldsymbol{V}]\leq 1}}{\operatorname{argmin}}\big\{\mathrm{DataFit}[y_1,\dots,y_n,\boldsymbol{x}_1,\dots,\boldsymbol{x}_n]\big\} \ , \tag{4}$$

where $\mathrm{DataFit} : \mathbb{R}^n \times \mathbb{R}^{n\times d}$ is a data-fitting function such as least-squares $\sum_{i=1}^n (y_i - f_{\boldsymbol{V}}[\boldsymbol{x}_i])^2$, and $h : \mathcal{V} \to [0,\infty)$ is a regularizer such as the elementwise ℓ_1-norm $\sum_{j,k,l}|(V^j)_{kl}|$. We are particularly interested in regularizers that induce sparsity.

2.2 Standard Notions of Sparsity

We first state two regularizers that are known in deep learning and the corresponding notions of sparsity.

Connection sparsity Consider the vanilla ℓ_1-regularizer

$$h^{\mathrm{C}}[\boldsymbol{V}] := \sum_{j=1}^{l}(r^{\mathrm{C}})_j\|V^j\|_1 := \sum_{j=1}^{l}(r^{\mathrm{C}})_j\sum_{v=1}^{p_j}\sum_{w=1}^{p_{j+1}}|(V^j)_{vw}| \ ,$$

where $\boldsymbol{r}^{\mathrm{C}} \in [0,\infty)^l$ is a vector of tuning parameters. This regularizer is the deep learning equivalent of the lasso regularizer in linear regression (Tibshirani, 1996) and has received considerable attention recently (Barron & Klusowski, 2018b; 2019; Kim et al., 2016). The regularizer acts on each individual connection, pruning a full network (first network from the left in Figure 1) to a more sparsely connected network (second network in Figure 1). We, therefore, propose to speak of *connection sparsity.*

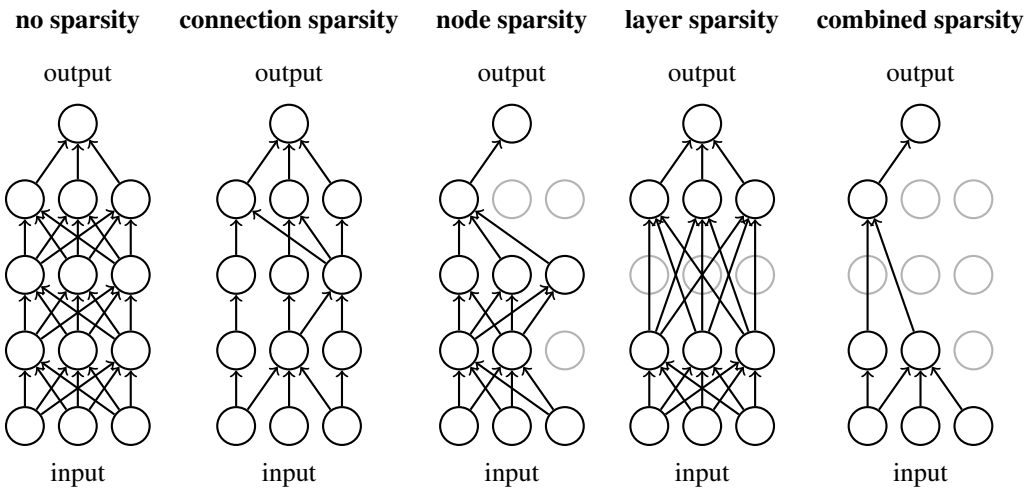

Figure 1: feedforward neural networks with different types of sparsity

Node sparsity Consider a grouped version of the above regularizer

$$h^{\mathrm{N}}[\boldsymbol{V}] := \sum_{j=1}^{l} (r^{\mathrm{N}})_j \|V^j\|_{2,1} := \sum_{j=1}^{l} (r^{\mathrm{N}})_j \sum_{v=1}^{p_j} \sqrt{\sum_{w=1}^{p_{j+1}} |(V^j)_{vw}|^2} \ ,$$

where $\boldsymbol{r}^{\mathrm{N}} \in [0,\infty)^l$ is again a vector of tuning parameters. This regularizer is the deep learning equivalent of the group lasso regularizer in linear regression (Bakin, 1999) and has received some attention recently (Alvarez & Salzmann, 2016; Feng & Simon, 2017; Scardapane et al., 2017). The regularizer acts on all connections that go into a node simultaneously, rendering entire nodes inactive (third network in Figure 1). We, therefore, propose to speak of *node sparsity.*

2.3 Layer Sparsity

We now complement the two existing regularizers and notions of sparsity with a new, third notion.

Layer sparsity Consider the regularizer

$$h^{\mathrm{L}}[\boldsymbol{V}] := \sum_{j=1}^{l-1} (r^{\mathrm{L}})_j \|V^j\|_{2,+} := \sum_{j=1}^{l-1} (r^{\mathrm{L}})_j \sqrt{\sum_{v=1}^{p_j} \sum_{w=1}^{p_{j+1}} \big(\mathrm{neg}[(V^j)_{vw}]\big)^2} \ , \tag{5}$$

where $\boldsymbol{r}^{\mathrm{L}} \in [0,\infty)^{l-1}$ is a vector of tuning parameters, and $\mathrm{neg}[a] := \min\{a, 0\}$ is the negative part of a real value $a \in \mathbb{R}$. This regularizers does not have an equivalent in linear regression, and it is also new in deep learning.

We argue that the regularizer can give rise to a new type of sparsity. The regularizer can be disentangled along the layers according to

$$h^{\mathrm{L}}[\boldsymbol{V}] = \sum_{j=1}^{l-1} (r^{\mathrm{L}})_j h^{\mathrm{L},j}[V^j]$$

with

$$h^{\mathrm{L},j}[V^j] := \sqrt{\sum_{v=1}^{p_j} \sum_{w=1}^{p_{j+1}} \big(\mathrm{neg}[(V^j)_{vw}]\big)^2} \qquad \text{for } j \in \{1,\ldots,l-1\} \ .$$

We then focus on an individual layer that corresponds an index $j \in \{1,\ldots,l-1\}$. We assume that 1. the coordinates of the activation functions of the jth and $(j+1)$th layers disentangle and 2. the individual coordinate functions of the $(j+1)$th layer are positive homogeneous. Precisely, we assume that

for every $\boldsymbol{t} \in \mathbb{R}^{p_j}$, $\boldsymbol{c} \in \mathbb{R}^{p_{j+1}}$, and $a \in [0, \infty)$, it holds that $\boldsymbol{f}^j[\boldsymbol{t}] = ((\boldsymbol{f}^j)_1[t_1], \dots, (\boldsymbol{f}^j)_{p_j}[t_{p_j}])^\top$, $\boldsymbol{f}^{j+1}[\boldsymbol{c}] = ((\boldsymbol{f}^{j+1})_1[c_1], \dots, (\boldsymbol{f}^{j+1})_{p_{j+1}}[c_{p_{j+1}}])^\top$, and $a(\boldsymbol{f}^{j+1})_m[c_m] = (\boldsymbol{f}^{j+1})_m[ac_m]$ for all $m \in \{1, \dots, p_{j+1}\}$ and some real-valued functions $(\boldsymbol{f}^j)_1, \dots, (\boldsymbol{f}^j)_{p_j}, (\boldsymbol{f}^{j+1})_1, \dots, (\boldsymbol{f}^{j+1})_{p_{j+1}}$. Standard examples that fit this framework are ReLU and leaky ReLU networks (Glorot et al., 2011; Hahnloser, 1998; Hahnloser et al., 2000; Salinas & Abbott, 1996).

We can now show that the regularizer h^{L} induces sparsity on the level of layers.

Theorem 1 (Layer Sparsity)**.** *Define a merged weight matrix $V^{j,j+1} \in \mathbb{R}^{p_j \times p_{j+2}}$ through $V^{j,j+1} := p_{j+1} V^j V^{j+1}$ and a merged activation function $\boldsymbol{f}^{j,j+1} : \mathbb{R}^{p_j} \to \mathbb{R}^{p_j}$ through $\boldsymbol{f}^{j,j+1} := ((\boldsymbol{f}^{j,j+1})_1, \dots, (\boldsymbol{f}^{j,j+1})_{p_j})^\top$ with*

$$(\boldsymbol{f}^{j,j+1})_q[t] := (\boldsymbol{f}^j)_q \left[\frac{1}{p_{j+1}} \sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m[t] \right] \qquad \text{for } q \in \{1, \dots, p_j\}, t \in \mathbb{R} \ .$$

It holds that

$$h^{\mathrm{L},j}[V^j] = 0 \quad \Rightarrow \quad \boldsymbol{f}^j \left[V^j \boldsymbol{f}^{j+1}[V^{j+1} \boldsymbol{z}] \right] = \boldsymbol{f}^{j,j+1} \left[V^{j,j+1} \boldsymbol{z} \right] \qquad \text{for all } \boldsymbol{z} \in \mathbb{R}^{p_{j+2}} \ .$$

That positive homogeneity can allow for moving weight matrices had been observed in Barron & Klusowski (2018a); here, we use the positive homogeneity to merge layers. The idea is as follows: $h^{\mathrm{L},j}[V^j] = 0$ means in view of the stated theorem that we can redefine the network of depth l as a network of depth $l-1$ by replacing $\boldsymbol{f}^j$ and V^j by $\boldsymbol{f}^{j,j+1}$ and $V^{j,j+1}$, respectively, and removing the $(j+1)$th layer.

One can verify readily that the coordinates of the merged activation functions $\boldsymbol{f}^{j,j+1}$ can still be disentangled. Moreover, the individual coordinate functions of the merged activation functions are positive homogeneous as long as the coordinates of the activation functions that correspond to the jth layer are positive homogenous. Hence, the regularization can merge not only one but many layers into one. In conclusion, our new regularizer h^{L} acts on all nodes and connections of each layer simultaneously, rendering entire layers inactive (fourth network in Figure 1). We, therefore, propose to speak of *layer sparsity.*

The concept of layer sparsity and Theorem 1 in particular do not hinge on the exact choice of the regularizer in (5): one can take any function h^{L} that can be disentangled along the layers as described and that ensures the fact that $h^{\mathrm{L},j}[V^j] = 0$ implies $\min_{k,l}(V^j)_{kl} \geq 0$.

We illustrate layer sparsity with two examples.

Example 1 (Identity Activation)**.** *We first highlight the meaning of layer sparsity in a simplistic setting. We consider identity activation, that is, $(\boldsymbol{f}^j)_q[t] = t$ for all $j \in \{2, \dots, l\}$, $q \in \{1, \dots, p_j\}$, and $t \in \mathbb{R}$. The networks in* (2) *can then be written as*

$$f_{\boldsymbol{V}}[\boldsymbol{x}_i] = \boldsymbol{f}^1[V^1 \cdots V^l \boldsymbol{x}_i] \ .$$

In other words, the initial l-layer network can be compressed into a one-layer network with activation function $\boldsymbol{f}^1$ and parameter matrix $V^1 \cdots V^l \in \mathbb{R}^{1 \times d}$. This setting with identity activation is, of course, purely academic, but it motivates an important question: can parts of networks be compressed similarly in the case of practical activation functions such as ReLU?

Theorem 1 gives an answer to this question: if $h^{\mathrm{L},j}[V^j] = 0$, then the jth and $(j+1)$th layers can be combined. In the extreme case $h^{\mathrm{L},2}[V^2] = \cdots = h^{\mathrm{L},l}[V^l] = 0$, the network can be condensed into a one-layer network just as in the linear case. In this sense, one can understand our layer regularizer is as a measure for the networks' "distance to linearity."

Example 2 (ReLU Activation)**.** *We now illustrate how layer sparsity compresses and, therefore, simplifies networks in the case of ReLU activation. ReLU activation means that $(\boldsymbol{f}^j)_q[t] = \max\{t, 0\}$ for all $j \in \{2, \dots, l\}$, $q \in \{1, \dots, p_j\}$, and $t \in \mathbb{R}$. One can verify readily that the merged activation functions of the jth and $(j+1)$th layer in Theorem 1 are then proportional to the original activation functions of the jth layer:*

$$(\boldsymbol{f}^{j,j+1})_q[t] = (\boldsymbol{f}^j)_q[t] \qquad \text{for } q \in \{1, \dots, p_j\}, t \in \mathbb{R} \ .$$

We show in the following that this leads to concise networks.

We fix an initial network $f_{\boldsymbol{N}}$ parameterized by $\boldsymbol{N} \in \mathcal{V}$. We identify the active layers of the network by

$$\mathcal{S} \equiv \mathcal{S}[\boldsymbol{N}] := \left\{ j \in \{1, \dots, l-1\} \,:\, h^{\mathrm{L},j}[N^j] \neq 0 \right\} \cup \{l\} \ . \tag{6}$$

Thus, $\mathcal{S}$ and $\{1, \dots, l\} \setminus \mathcal{S}$ contain the indexes of the relevant and irrelevant layers, respectively. The level of sparsity, that is, the number of active layers, is $s := |\mathcal{S}| \leq l$.

We now denote the indexes in $\mathcal{S}$ in an orderly fashion: $j_1, \dots, j_s \in \mathcal{S}$ such that $j_1 < \dots < j_s = l$. We then define scaled versions of the corresponding merged matrices: if $j_i - 1 \in \mathcal{S}$ or $j_i = 1$, we do the "trivial merge" $M^{j_i} := N^{j_i} \in \mathbb{R}^{p_{j_i} \times p_{j_i+1}}$; otherwise, we do the "non-trivial merge"

$$M^{j_i} := (p_{j_{i-1}+2} \cdots p_{j_i}) N^{j_{i-1}+1} \cdots N^{j_i} \in \mathbb{R}^{p_{j_{i-1}+1} \times p_{j_i+1}} \ .$$

In other words, we merge all irrelevant layers between the j_{i-1}th and j_ith layers into the j_ith layer.

We can then compress the data-generating model in (1) *into*

$$y_i = \boldsymbol{f}^{j_1}\left[M^{j_1} \boldsymbol{f}^{j_2}\left[\dots \boldsymbol{f}^{j_s}[M^{j_s} \boldsymbol{x}_i]\right]\right] + u_i$$

with $\boldsymbol{M} := (M^{j_1}, \dots, M^{j_s}) \in \mathcal{V}_{\mathcal{S}} := \{\boldsymbol{V} = (V^1, \dots, V^s) : V^i \in \mathbb{R}^{p_{j_{i-1}+1} \times p_{j_i+1}}\}$. Formulated differently, we can condense the original network according to

$$f_{\boldsymbol{N}}[\boldsymbol{x}_i] = f_{\boldsymbol{M}}[\boldsymbol{x}_i] = \boldsymbol{f}^{j_1}\left[M^{j_1} \boldsymbol{f}^{j_2}\left[\dots \boldsymbol{f}^{j_s}[M^{j_s} \boldsymbol{x}_i]\right]\right] \ ,$$

that is, we can formulate the initial ReLU activation network with l layers as a new ReLU activation network with s layers.

The new network is still a ReLU activation network but has a smaller number of layers if $s < l$ and, consequently, a smaller number of parameters in total: the total number of parameters in the initial network is $\sum_{j=1}^{l}(p_j \times p_{j+1})$, while the total number of parameters in the transformed network is only $\sum_{i=1}^{s}(p_{j_{i-1}+1} \times p_{j_i+1})$.

Our concept for regularizing layers is substantially different from existing ones: our layer-wise regularizer induces weights to be *non-negative*, whereas existing layer-wise regularizers induce weights to be *zero* (Wen et al., 2016, Section 3.3). The two main advantages of our approach are that it (i) does not require shortcuts to avoid trivial networks and (ii) does not implicitly enforce connection or node sparsity. We thus argue that our layer sparsity is a much more natural and appropriate way to capture and regularize network depths.

Layer sparsity more closely relates to ResNets (He et al., 2016). The recent popularity of ResNets is motivated by two observations: 1. Solvers seem to struggle with finding good minima of deep networks; even training accuracies can deteriorate when increasing the number of layers. 2. Allowing for linear mappings that short-circuit parts of the network seem to help solvers in finding better minima. From our viewpoint here, one can argue that ResNets use these linear mappings to regulate network depths adaptively and, therefore, are related to layer sparsity. But importantly, ResNets are even more complex than the networks they are based on, while our notion simplifies networks.

Since, as one can verify again readily, all three regularizers are convex, any combination of them is also convex. Such combinations can be used to obtain networks that are sparse in two or all three aspects (last network in Figure 1).

2.4 Refitting

Layer-sparse estimators such as (3) with $h = h^{\mathrm{L}}$ provide networks that can be condensed into smaller networks. But it is well-known that regularization generates bias. A question is, therefore, whether we can get the best of both worlds: can we construct a method that yields smaller networks yet avoids bias?

We propose a pipeline that complements the regularized estimators with an additional step: after estimating a layer-sparse network, we condense it and re-adjust its parameters by using an unregularized estimator such as least-squares. In line with terminology in linear regression (Chzhen et al., 2019; Lederer, 2013), we call the additional step *refitting*.

While this refitting idea applies very generally, we formulate a version of it in the framework of Example 2 to keep the notation light.

Example 3 (ReLU Activation Cont.). *Consider the model in* (1) *with the specifications of Example 2, and consider a corresponding layer-sparse estimator* $\widehat{\boldsymbol{W}}$ *of the parameters such as* (3) *with* $h = h^{\mathrm{L}}$. *In line with* (6)*, we denote the set of the active layers by*

$$\mathcal{S} = \mathcal{S}[\widehat{\boldsymbol{W}}] = \big\{ j \in \{1, \ldots, l-1\} \,:\, h^{\mathrm{L},j}[\widehat{W}^j] \neq 0 \big\} \cup \{l\}$$

and the corresponding parameter space of the condensed network by $\mathcal{V}_{\mathcal{S}} \equiv \mathcal{V}_{\mathcal{S}}[\widehat{\boldsymbol{W}}] = \{\boldsymbol{V} = (V^1, \ldots, V^s) : V^i \in \mathbb{R}^{p_{j_{i-1}+1} \times p_{j_i+1}}\}$*, where* $s := |\mathcal{S}|$. *The least-squares refitted estimator for the parameters in the condensed network is then*

$$\widehat{\boldsymbol{W}}_{\mathcal{S}} \in \operatorname*{argmin}_{\boldsymbol{V} \in \mathcal{V}_{\mathcal{S}}} \bigg\{ \sum_{i=1}^{n} \big(y_i - f_{\boldsymbol{V}}[\boldsymbol{x}_i] \big)^2 \bigg\} \ . \tag{7}$$

Hence, the estimator $\widehat{\boldsymbol{W}}$ *complemented with least-squares refitting yields the network*

$$f_{\widehat{\boldsymbol{W}}_{\mathcal{S}}}[\boldsymbol{x}_i] = \boldsymbol{f}^{j_1} \Big[\widehat{W}_{\mathcal{S}}^{j_1} \boldsymbol{f}^{j_2} \big[\ldots \boldsymbol{f}^{j_s} [\widehat{W}_{\mathcal{S}}^{j_s} \boldsymbol{x}_i] \big] \Big] \ . \tag{8}$$

Of course, the refitting scheme can be adapted to any type of sparsity. But in the following section, we focus on layer-sparsity and show that layer-sparse estimator with refitting can both reduce network sizes and prediction errors simultaneously.

3 Simulation Study

We now confirm in a brief simulation study that layer regularization can 1. improve prediction accuracies and 2. reduce the number of active layers.

3.1 Simulation Framework

We generate data according to the model in (1). The most outside activation function $\boldsymbol{f}^1$ is the identity function, and the coordinates of all other activation functions $\boldsymbol{f}^2, \ldots, \boldsymbol{f}^l$ are ReLU functions. The input vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are jointly independent and standard normally distributed in d dimensions; the noise random variables $u_1, \ldots, u_n$ are independent of the input, jointly independent, and standard normally distributed in one dimension. For a given sparsity level $s_W \in [0, 1]$, a vector $\boldsymbol{s} \in \{0, 1\}^{l-1}$ with independent Bernoulli distributed entries that have success parameter s_W is generated. The entries of the parameter matrix W^1 are sampled independently from the uniform distribution on $(-2, 2)$, and the entries of the parameter matrices $W^2, \ldots, W^l$ are sampled independently from the uniform distribution on $(0, 2)$ if $s_j = 0$ and on $(-2, 2)$ otherwise. Hence, the parameter s_W controls the level of the layer sparsity: the smaller s_W, the higher the network's layer sparsity.

In concrete numbers, the input dimension is $d = 2$, the network widths are $p_2 = \cdots = p_l = 5$, the number of hidden layers is $l - 1 \in \{10, 25\}$, and the sparsity level is $s \in \{0.1, 0.3, 0.9\}$. Our settings and values represent, of course, only a very small part of possible networks in practice, but given the generality of our concepts, any attempt of an exhaustive simulation study must fail, and the simulations at least allow us (i) to corroborate our theoretical insights and (ii) to indicate that our concepts can be very useful in practice.

Datasets of 150 samples are generated; $n = 100$ of the samples are assigned to training and the rest to testing. The relevant measures for an estimate $\widehat{\boldsymbol{W}}$ of the network's parameters are the empirical mean squared error

$$\widehat{\mathrm{mse}} \equiv \widehat{\mathrm{mse}}[\widehat{\boldsymbol{W}}] := \frac{1}{|\mathcal{T}|} \sum_{(y, \boldsymbol{x}) \in \mathcal{T}} \big(y - f_{\widehat{\boldsymbol{W}}}[\boldsymbol{x}] \big)^2 \ ,$$

over the test set $\mathcal{T}$ with cardinality $|\mathcal{T}| = 50$ and the level of sparsity among the hidden layers

$$\hat{\mathrm{s}} \equiv \hat{\mathrm{s}}[\widehat{\boldsymbol{W}}] := \big| \big\{ j \in \{1, \ldots, l-1\} \,:\, h^{\mathrm{L},j}[\widehat{W}^j] \neq 0 \big\} \big| \ .$$

Reported are the medians (and third quantiles in paranthesis) over 30 simulation runs for each setting.

	$d = 2, n = 100$									
	$l - 1 = 10$						$l - 1 = 25$			
	$\mathrm{s_W} = 0.1$		$\mathrm{s_W} = 0.3$		$\mathrm{s_W} = 0.9$		$\mathrm{s_W} = 0.1$		$\mathrm{s_W} = 0.3$	
Method	$\widehat{\mathrm{mse}}$	$\hat{\mathrm{s}}$	$\widehat{\mathrm{mse}}$	$\hat{\mathrm{s}}$	$\widehat{\mathrm{mse}}$	$\hat{\mathrm{s}}$	$\widehat{\mathrm{mse}}$	$\hat{\mathrm{s}}$	$\widehat{\mathrm{mse}}$	$\hat{\mathrm{s}}$
LS	1.123 (1.396)	—	1.112 (1.448)	—	1.060 (1.349)	—	0.989 (1.263)	—	0.988 (1.260)	—
SLS	0.018 (0.023)	—	0.035 (0.616)	—	0.203 (0.830)	—	0.142 (1.020)	—	0.147 (0.985)	—
ILS	0.006 (0.011)	1 (2)	0.008 (0.023)	3 (4)	0.907 (1.272)	9 (10)	0.003 (0.024)	2 (4)	0.050 (1.012)	7 (9)
FLS	0.007 (0.013)	1 (1)	0.008 (0.128)	2 (4)	0.364 (1.137)	8 (9)	0.009 (0.985)	2 (5)	0.021 (1.120)	4 (13)

Table 1: encouraging layer sparsity can reduce the prediction error and the model complexity

3.2 METHODS

Our first method (SLS) is a standard least-squares complemented with the layer regularizer (5) in Lagrange form $\widehat{\boldsymbol{W}}$. The baseline for this estimator is vanilla least-squares (LS). Since our estimator—in contrast to least-squares—allows for merging layers, we can also complement it with our refitting scheme of Section 2.4 (FLS). The baseline for our refitted estimator is the least-squares estimator that "knows" the relevant layers beforehand (ILS), that is, a least-squares on the relevant layers $\mathcal{V}_{\mathcal{S}}[\boldsymbol{W}]$ with $\boldsymbol{W}$ the true parameter—see Example 2. The latter estimator cannot be used in practice, but it can serve as a benchmark here in the simulations.

The objective functions are optimized by using mini-batch gradient descent with batch size 10, learning rate 10^{-2}, and number of epochs 200 (for $l = 10$, $\mathrm{s_W} = 0.1, 0.3$), 300 (for $l = 10$, $\mathrm{s_W} = 0.9$), 400 (for $l = 25$, $\mathrm{s_W} = 0.1$), and 500 (otherwise). The tuning parameters $(r^{\mathrm{L}})_j$ are 0.2 (for $l = 10$, $\mathrm{s_W} = 0.1$), 0.12 (for $l = 10$, $\mathrm{s_W} = 0.3$), 0.07 (for $l = 10$, $\mathrm{s_W} = 0.9$), and 0.05 (for $l = 25$).

3.3 RESULTS

The numerical results show that our layer-regularized version SLS can improve on the prediction accuracy of the standard least-squares LS considerably ($\widehat{\mathrm{mse}}$-columns of the first and second rows in Table 1). The results also show that the refitting in FLS can improve the prediction accuracy further, and that refitted estimator can rival the infeasible ILS in terms of prediction ($\widehat{\mathrm{mse}}$-columns of the third and fourth rows). The results finally show that the layer regularization can detect the correct number of layers ($\hat{\mathrm{s}}$-columns of the third and fourth rows). In summary, our layer-regularized estimator outmatches the standard least-squares, and the refitted version of our estimator rivals the infeasible least-squares that knows which are the relevant layers beforehand—both in terms of prediction accuracy and sparsity. Hence, layer regularization can condense networks effectively.

The results also reveal that the prediction accuracies of our estimators increase with $\mathrm{s_W}$ decreasing ($\widehat{\mathrm{mse}}$-columns across different $\mathrm{s_W}$). This trend is expected: the higher the layer sparsity, the more layer-regularization can condense the networks. This behavior is confirmed in the sparsities ($\hat{\mathrm{s}}$-columns across different $\mathrm{s_W}$). In other words, the layer regularization is adaptive to the true layer sparsity.

The tuning parameters $(r^{\mathrm{L}})_j$ have been calibrated very roughly by hand. We expect that a more careful calibration of $\boldsymbol{r}^{\mathrm{L}}$ based on cross-validation, for example, accentuates the positive effects of layer sparsity even further. But since our goal in this section is a general proof of concept for layer sparsity rather than the optimization of a specific deep learning pipeline, we do not pursue this further here.

The tuning parameters of the descent algorithm, such as the batch size, number of epochs, learning rate, and so forth, have also been calibrated very roughly by hand. An observation is the fact that all methods can sometimes provide accurate prediction if the number of epochs is extremely large, but our layer-regularized methods SLS and FLS generally lead to accurate prediction after much less epochs than their unregularized counterpart LS. This observation indicates that layer regularization also impacts the algorithmic aspects of deep learning beneficially.

4 DISCUSSION

We have shown that layer sparsity can compress layered networks effectively both in theory (Section 2) and practice (Section 3).

Related concepts such as ResNets add complexity to the network descriptions and, therefore, are very hard to analyze statistically. Layer sparsity, in contrast, simplify network architectures and seem amenable to statistical analyses via recent techniques for regularized deep learning (Taheri et al., 2020). A statistical treatment of layer sparsity, therefore, seems a feasible topic for further research.

Another topic for further research is the practical calibration of the tuning parameters. Tuning parameter calibration is an active topic in machine learning; thus, we expect that one can use ideas developed in other frameworks, such as (Chichignoud et al., 2016).

In summary, layer sparsity complements other notions of sparsity that concern individual connections or nodes. All of these concepts can help to fit networks that are efficient in terms of memory and computations and easy to interpret.

REFERENCES

J. Alvarez and M. Salzmann. Learning the number of neurons in deep networks. In *Adv. Neural Inf. Process Syst.*, pp. 2270–2278, 2016.

T. Ash. Dynamic node creation in backpropagation networks. *Connect. Sci.*, 1(4):365–375, 1989.

S. Bakin. *Adaptive regression and model selection in data mining problems*. PhD thesis, The Australian National University, 1999.

A. Barron and J. Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv:1809.03090*, 2018a.

A. Barron and J. Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv:1809.03090*, 2018b.

A. Barron and J. Klusowski. Complexity, statistical risk, and metric entropy of deep nets using total path variation. *arXiv:1902.00800*, 2019.

M. Bello. Enhanced training algorithms, and integrated training/architecture selection for multilayer perceptron networks. *IEEE Trans. Neural Netw.*, 3(6):864–875, 1992.

S. Changpinyo, M. Sandler, and A. Zhmoginov. The power of sparsity in convolutional neural networks. *arXiv:1702.06257*, 2017.

M. Chichignoud, J. Lederer, and M. Wainwright. A practical scheme and fast algorithm to tune the lasso with optimality guarantees. *J. Mach. Learn. Res.*, 17(1):1–20, 2016.

E. Chzhen, M. Hebiri, and J. Salmon. On Lasso refitting strategies. *Bernoulli*, 25(4A):3175–3200, 2019.

J. Feng and N. Simon. Sparse-input neural networks for high-dimensional nonparametric regression and classification. *arXiv:1711.07592*, 2017.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proc. Mach. Learn. Res.*, pp. 315–323, 2011.

N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. *arXiv:1712.06541*, 2017.

R. Hahnloser. On the piecewise analysis of networks of linear threshold neurons. *Neural Networks*, 11(14):691–697, 1998.

R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, and H. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951, 2000.

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

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: The lasso and generalizations*. CRC press, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, pp. 770–778, 2016.

J. Kim, V. Calhoun, E. Shim, and J.-H. Lee. Deep neural network with weight sparsity control and pre-training extracts hierarchical features and enhances classification performance: Evidence from whole-brain resting-state functional connectivity patterns of schizophrenia. *Neuroimage*, 124: 127–146, 2016.

M. Kohler and S. Langer. On the rate of convergence of fully connected very deep neural network regression estimates. *arXiv:1908.11133*, 2019.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

J. Lederer. Trust, but verify: benefits and pitfalls of least-squares refitting in high dimensions. *arXiv:1306.0113*, 2013.

S. Liang and R. Srikant. Why deep neural networks for function approximation? *arXiv:1610.04161*, 2016.

B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, pp. 806–814, 2015.

E. Salinas and L. Abbott. A model of multiplicative neural responses in parietal cortex. *Proc. Natl. Acad. Sci. USA*, 93(21):11956–11961, 1996.

S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

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

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

M. Taheri, F. Xie, and J. Lederer. Statistical guarantees for regularized networks. *arXiv:2006.00294*, 2020.

M. Telgarsky. Benefits of depth in neural networks. In *Annual Conference on Learning Theory*, volume 49 of *Proc. Mach. Learn. Res.*, pp. 1517–1539, 2016.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, 58(1):267–288, 1996.

W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Adv. Neural Inf. Process Syst.*, pp. 2082–2090, 2016.

D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94: 103–114, 2017.

A PROOFS

We now provide a proof for the theorem.

Proof of Theorem 1. A main prerequisite of the proof is that positive weights can be pulled insight activation functions that are positive homogeneous.

We first show that $h^{\mathrm{L},j}[V^j] = 0$ allows us to pull the weight matrix V^j inside the activation function $\boldsymbol{f}^{j+1}$: If $h^{\mathrm{L},j}[V^j] = 0$, then $(V^j)_{qm} \geq 0$ for all $q \in \{1, \dots, p_j\}$, $m \in \{1, \dots, p_{j+1}\}$. Hence, we find for all $\boldsymbol{c} \in \mathbb{R}^{p_{j+1}}$ and $q \in \{1, \dots, p_j\}$ that

$$\begin{aligned}\big(V^j \boldsymbol{f}^{j+1}[\boldsymbol{c}]\big)_q &= \sum_{m=1}^{p_{j+1}} (V^j)_{qm} (\boldsymbol{f}^{j+1})_m [c_m] \\ &= \sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[(V^j)_{qm} c_m\big] \quad .\end{aligned}$$

We have used the assumed positive homogeneity of $\boldsymbol{f}^{j+1}$ in the second line.

Using this result with $\boldsymbol{c} = V^{j+1}\boldsymbol{z}$ then yields

$$\begin{aligned}&(\boldsymbol{f}^j)_q \Big[\big(V^j \boldsymbol{f}^{j+1}[V^{j+1}\boldsymbol{z}]\big)_q\Big] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[(V^j)_{qm} (V^{j+1}\boldsymbol{z})_m\big]\right] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[\langle (V^j)_{q.}, V^{j+1}\boldsymbol{z}\rangle\big]\right] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[\langle (V^{j+1})^\top (V^j)_{q.}, \boldsymbol{z}\rangle\big]\right] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[\langle (V^{j+1})^\top ((V^j)^\top)_{.q}, \boldsymbol{z}\rangle\big]\right] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[\langle \big((V^{j+1})^\top (V^j)^\top\big)_{.q}, \boldsymbol{z}\rangle\big]\right] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[\langle \big((V^j V^{j+1})^\top\big)_{.q}, \boldsymbol{z}\rangle\big]\right] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[\langle (V^j V^{j+1})_{q.}, \boldsymbol{z}\rangle\big]\right] \\ &= (\boldsymbol{f}^j)_q \left[\sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[(V^j V^{j+1}\boldsymbol{z})_q\big]\right] \quad .\end{aligned}$$

Using the positive homogeneity of $\boldsymbol{f}^{j,j+1}$ once more then gives

$$(\boldsymbol{f}^j)_q \Big[\big(V^j \boldsymbol{f}^{j+1}[V^{j+1}\boldsymbol{z}]\big)_q\Big] = (\boldsymbol{f}^j)_q \left[\frac{1}{p_{j+1}} \sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[(p_{j+1} V^j V^{j+1}\boldsymbol{z})_q\big]\right] \quad .$$

Together with the definitions of $V^{j,j+1}$ and $\boldsymbol{f}^{j,j+1}$, this yields

$$\begin{aligned}&(\boldsymbol{f}^j)_q \Big[\big(V^j \boldsymbol{f}^{j+1}[V^{j+1}\boldsymbol{z}]\big)_q\Big] \\ &\qquad = (\boldsymbol{f}^j)_q \left[\frac{1}{p_{j+1}} \sum_{m=1}^{p_{j+1}} (\boldsymbol{f}^{j+1})_m \big[(V^{j,j+1}\boldsymbol{z})_q\big]\right] = (\boldsymbol{f}^{j,j+1})_q \big[(V^{j,j+1}\boldsymbol{z})_q\big] \quad .\end{aligned}$$

This identity concludes the proof in view of the assumed forms of $\boldsymbol{f}^j$ and $\boldsymbol{f}^{j,j+1}$. □