Non-Smooth Regularization: Improvement to Learning Framework through Extrapolation

Sajjad Amini, Mohammad Soltanian, Mostafa Sadeghi, Shahrokh Ghaemmaghami

▶ **To cite this version:**

Sajjad Amini, Mohammad Soltanian, Mostafa Sadeghi, Shahrokh Ghaemmaghami. Non-Smooth Regularization: Improvement to Learning Framework through Extrapolation. IEEE Transactions on Signal Processing, 2022, 70, pp.1213 - 1223. 10.1109/TSP.2022.3154969 . hal-03586153

HAL Id: hal-03586153
https://inria.hal.science/hal-03586153

Submitted on 23 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Non-Smooth Regularization: Improvement to Learning Framework through Extrapolation

Sajjad Amini, Mohammad Soltanian, Mostafa Sadeghi, and Shahrokh Ghaemmaghami, *Senior Member, IEEE*

***Abstract*—Deep learning architectures employ various regularization terms to handle different types of priors. Non-smooth regularization terms have shown promising performance in the deep learning architectures and a learning framework has recently been proposed to train autoencoders with such regularization terms. While this framework efficiently manages the non-smooth term during training through proximal operators, it is limited to autoencoders and suffers from low convergence speed due to several optimization sub-problems that must be solved in a row. In this paper, we address these issues by extending the framework to general feed-forward neural networks and introducing variable extrapolation which can dramatically increase the convergence speed in each sub-problem. We show that the proposed update rules converge to a critical point of the objective function under mild conditions. To compare the resulting framework with the previously proposed one, we consider the problem of training sparse autoencoders and robustifying deep neural architectures against both targeted and untargeted attacks. Simulations show superior performance in both convergence speed and final objective function value.**

***Index Terms*—Deep neural networks, training, regularizer, extrapolation, robustness, sparsity, proximal operator.**

I. Introduction

DEEP learning is responsible for the most of recent advances in machine vision [1], [2], natural language processing [3], and time series prediction [4]. In contrast to traditional learning architecture, deep learning architectures are equipped with a hierarchy of layers that gradually transfer input pattern representation from low level ones in the starting layers to more abstract representations at the higher layers of architecture [5].

Although the idea of learning via a hierarchy of layers was traditionally in use, basic difficulties in the learning of these architectures, such as gradient vanishing and random weight initialization blocked their applicability [6]. This condition continued until 2006 when Hinton proposed a smart initialization method for deep generative models using repetitive utilization of a shallow generative model named restricted Boltzmann machine (RBM) [7]. In the sequel, deterministic models based on autoencoders were also introduced to initialize deep fully connected architectures [8].

Deep neural networks can be categorized into recurrent and feed-forward. While data flow for inference is unidirectional in feed-forward neural networks (FFNNs), recurrent neural networks (RNNs) have at least one loop [5]. FFNNs include two main groups, namely generative and discriminative models. Variational autoencoders (VAEs), generative adversarial networks (GANs) are among generative FFNNs, while fully-connected and convolutional neural networks (FCNNs and CNNs) are examples of discriminative FFNNs [8].

Different types of regularization have been proposed to enhance the efficiency of FFNNs. Introduction of sparsity regularizer has successfully improved the initialization quality of deep fully-connected neural networks [9], [10]. Low-coherence regularization has also shown impressive results in improving CNNs accuracy [11], which was previously explored for dictionary learning [12]. Different regularization strategies have also been proposed in the literature targeting the robustification of CNNs [13], [14]. The aforementioned examples can be considered as mathematical regularizers, while various structural regularization schemes are also widely used in different architectures such as dropout, batch normalization and batch re-normalization [15].

Smooth mathematical regularizers are preferable in the sense that they meet smoothness constraint assumed by many optimization algorithms used to learn network parameters [6]. Recently, a new learning framework has been proposed which aims at eliminating the smoothness constraint for the regularizers [9]. In this framework, instead of smoothness, the regularizer must be proximable which makes many non-smooth regularizers applicable to the problem of training FFNNs. This framework has shown impressive results for training sparse AEs [9] and robustifying CNNs [13].

However, the framework proposed in [9] is limited to AE architecture and does not consider a general framework for the class of FFNNs. It also suffers from low convergence speed. In this paper, we extend the framework given in [9] by making the following improvements:

- While the framework introduced in [9] is limited to autoencoders, in this paper we propose a new framework applicable to the general category of feed-forward neural networks, including fully-connected and convolutional neural networks.
- The framework proposed in [9] exhibit slow convergence speed which necessitates numerous iterations that limits its application for large scale datasets. In contrast, in this paper an extrapolation strategy is developed that results in much fewer iterations for training and expands the applicability of the resulting framework. The convergence analysis of the proposed framework is also established, which is not trivial with respect to [9] in the case of extrapolated variables.

S. Amini (e-mail: s_amini@sharif.edu) and S. Ghaemmaghami (e-mail: ghaemmag@sharif.edu) are with the Electronics Research Institute (ERI) and Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran. M. Soltanian (e-mail: m.soltanian@khu.ac.ir) is with Department of Computer Science, Faculty of Mathematical Sciences and Computer, Kharazmi University, Tehran, Iran. M. Sadeghi (e-mail: mostafa.sadeghi@inria.fr) is with the Multispeech team, Inria Nancy - Grand Est, France.

- Conducting extensive experiments by distinct regularizers and over different datasets, the performance of the proposed framework is tested especially against the framework proposed in [9], and its superiority in terms of objective function value and convergence speed is demonstrated for extrapolated variables.

The rest of the paper is organized as follows. The notation used throughout the paper and preliminaries are reviewed in Section II. We review the regularizers proposed in the literature for training sparse AEs and robustifying CNNs in Section III and define the modifications to the framework for the general case of FFNN with non-smooth regularization and extrapolation in Section IV. In Section V, we investigate performance of the modified framework for sparse AE training and robustifying CNNs and compare it to the original framework and the previously proposed methods over different benchmark datasets. We conclude the paper in Section VI.

II. Notations and preliminaries

In this paper, bold uppercase and bold lowercase letters are used to show matrices and vectors, respectively. The i-th element of vector $\mathbf{x}$ is denoted by x_i and w_{ij} denotes the (i,j) element of matrix $\mathbf{W}$. $\text{vec}(\cdot)$ indicates vectorizing operator that produces column-wise concatenation of a matrix. Column-wise concatenation of m vectors is denoted by $\mathbf{y}=[\mathbf{y}_1,\ldots,\mathbf{y}_m]$. As for norms, $\|\cdot\|_k$ and $\|\cdot\|_F$ indicate the ℓ_k vector norm and matrix Frobenius norm, respectively.

In the rest of this section, we introduce some definitions and lemmas needed throughout the paper.

Definition 1 (Proximal Operator [16])**.** *Let $f:\mathbb{R}^n\to\mathbb{R}\cup\{+\infty\}$ be a proper and lower semi-continuous convex function. The proximal operator (mapping) $\text{Prox}_f:\mathbb{R}^n\to\mathbb{R}^n$ of f at $\mathbf{v}$ is defined as: $\text{Prox}_f(\mathbf{v})=\arg\min_{\mathbf{x}} f(\mathbf{x})+\frac{1}{2}\|\mathbf{x}-\mathbf{v}\|_2^2$*

Based on the above definition, we can calculate proximal operator for different functions.

Lemma 1 (Proximal operator for ℓ_∞ norm [16])**.** *Let $f(\mathbf{x})$ be scaled ℓ_∞ norm defined as $f(\mathbf{x})=\lambda\max_i|x_i|$. The proximal operator for $f(\mathbf{x})$ is defined as $\text{Prox}_f(\mathbf{v})=\mathbf{v}-\mathcal{P}_{\mathcal{B}_i^\lambda}(\mathbf{v})$ where $\mathcal{P}_{\mathcal{B}_i^\lambda}(\cdot)$ is the projection onto ℓ_1 norm ball with radius λ.*

Lemma 2 (Proximal operator for ℓ_∞ indicator norm [13])**.** *Let $f(\mathbf{x})$ be the scaled ℓ_∞ indicator norm defined as ($\lambda>0$):*

$$f(\mathbf{x})=\lambda\mathcal{I}\left\{\|\mathbf{x}\|_\infty\le\xi\right\}=\begin{cases}0 & \text{if } \|\mathbf{x}\|_\infty\le\xi\\ \infty & \text{if } \|\mathbf{x}\|_\infty>\xi\end{cases}.$$

The proximal operator for $f(\mathbf{x})$ is denoted $\text{Prox}_f(\mathbf{v})=\mathbf{u}$ where $\mathbf{u}$ is:

$$u_i=\begin{cases}v_i & \text{if } |v_i|\le\xi\\ \text{sgn}(v_i)\xi & \text{if } |v_i|>\xi\end{cases}.$$

Definition 2 (Lipschitz Gradient [17])**.** *A function $f:\mathbb{R}^d\to\mathbb{R}$ has a Lipschitz gradient if $\|\nabla f(\mathbf{x})-\nabla f(\mathbf{y})\|\le L\|\mathbf{x}-\mathbf{y}\|$ for all $\mathbf{x}$ and $\mathbf{y}$. The smallest constant L that satisfies the inequality, is the Lipschitz constant of the gradient.*

A twice differentiable function $f:\mathbb{R}^d\to\mathbb{R}$ with Hessian matrix denoted $\mathbf{H}$ has Lipschitz gradient if $\mathbf{M}=L\mathbf{I}-\mathbf{H}$ is positive semi-definite. The smallest L making $\mathbf{M}$ positive semi-definite is the Lipschitz constant of the gradient [18].

Definition 3 (Critical Point [19])**.** *A point $\mathbf{x}^*$ is called a critical point of f if $\mathbf{0}\in\partial f(\mathbf{x}^*)$ where $\partial f(\mathbf{x}^*)$ is the limiting Fréchet subdifferential at $\mathbf{x}^*$.*

In the convergence analysis presented in this work, to transfer from subsequence convergence to the whole sequence convergence, one key factor is the Kurdyka-Łojasiewicz property defined as follows.

Definition 4 (Kurdyka-Łojasiewicz (KL) property [20])**.** *A function $f(\mathbf{x})$ satisfies the KL property at point $\bar{\mathbf{x}}\in\text{dom}(\partial f)$ if there exist $\eta>0$, a neighborhood $\mathcal{B}_\rho(\bar{\mathbf{x}})\triangleq\{\mathbf{x}:\|\mathbf{x}-\bar{\mathbf{x}}\|<\rho\}$, and a concave function $g(a)=c\cdot a^{1-\theta}$ for some $c>0$ and $\theta\in[0,1)$ such that the KL inequality holds*

$$g(|f(\mathbf{x})-f(\bar{\mathbf{x}})|)\text{dist}(\mathbf{0},\partial f(\mathbf{x}))\ge 1,$$
$$\forall\mathbf{x}\in\mathcal{B}_\rho(\bar{\mathbf{x}})\cap\text{dom}(\partial f)\text{ and }f(\bar{\mathbf{x}})<f(\mathbf{x})<f(\bar{\mathbf{x}})+\eta,$$

where $\text{dom}(\partial f)=\{\mathbf{x}|\partial f(\mathbf{x})\neq\emptyset\}$ and $\text{dist}(\mathbf{0},\partial f(\mathbf{x}))=\min\{\|\mathbf{y}\|_2|\mathbf{y}\in\partial f(\mathbf{x})\}$.

We use the resulting extended framework to robustify a deep CNN against untargeted attacks. We use four types of untargeted attacks, namely random perturbation (*Random*), regular perturbation (*Regular*) [21], fast gradient sign perturbation (*Fast*) [14], and projected gradient descent perturbation (*PGD*), which are defined as follows.

Definition 5 (Regular adversary)**.** *Let $d(\widehat{\mathbf{y}},\mathbf{y})$ denote a loss function used to train a FFNN, which is usually a distance measure between target vector $\mathbf{y}$ and network output vector $\widehat{\mathbf{y}}$ corresponding to input pattern $\mathbf{x}$, and $f(\mathbf{p},\mathbf{x})$ represents the mapping that evaluates output of the network based on input pattern $\mathbf{x}$, and network variables $\mathbf{p}$. The regular adversary $\widetilde{\mathbf{x}}$ corresponding to clean pattern $\mathbf{x}$ is defined as [21]:*

$$\widetilde{\mathbf{x}}=\mathbf{x}+\epsilon\,\nabla_{\mathbf{x}}d\left(f\left(\mathbf{p},\mathbf{x}\right),\mathbf{y}\right),\tag{1}$$

where ϵ controls the signal contamination in the sense of signal-to-noise ratio (SNR). The resulting SNR is:

$$\text{SNR}=20\log_{10}\frac{\|\mathbf{x}\|_2}{\|\epsilon\,\nabla_{\mathbf{x}}d\left(f\left(\mathbf{p},\mathbf{x}\right),\mathbf{y}\right)\|_2}.\tag{2}$$

Fast gradient sign method [14] uses the sign information of gradient to design perturbation as:

$$\widetilde{\mathbf{x}}=\mathbf{x}+\epsilon\,\text{sign}\left(\nabla_{\mathbf{x}}d(f\left(\mathbf{p},\mathbf{x}\right),\mathbf{y})\right).\tag{3}$$

To design the perturbation in *PGD* method, (1) is iteratively applied while the resulting $\widetilde{\mathbf{x}}$ is projected on the feasible input pattern range. In the *Random* method, the perturbation is designed irrespective of the network architecture using a predefined distribution, e.g., normal.

III. Prior Art

FFNNs play an important role in learning hidden structures of patterns such as images where all the samples are processed simultaneously [8]. Various priors are introduced in the literature which are generally imposed through regularizers aiming

at improving the different performance metrics of FFNNs such as generalizations error, robustness, etc. [14], [11], [13]

Autoencoder, an encoder-decoder fully-connected neural network, is among the well-known FFNNs suitable for the initialization of deep fully-connected networks. Due to the high capacity of fully-connected layers, autoencoder provides high capacity for learning which necessitates architecture regularization, e.g., sparse autoencoders [22]. Although the ℓ_0 (pseudo) norm is the first option to impose sparsity, the resulting optimization problem is hard to deal with, due to the discrete nature of the ℓ_0 norm [23]. So, smooth and continuous approximations are considered in the literature, e.g., Kullback-Leibler divergence (KLD) [22] and Student-t distribution approximation [24]. The cooperative effect of non-negativity of weight matrices in the sparsity of representation in an autoencoder based on KLD has also been shown in [25]. Besides, smooth approximations of sparsity, ℓ_1 norm, as the tightest norm below ℓ_0 on any ℓ_p-unit ball ($p \in [1, \infty]$) [26], has shown promising results in sparse representation and dictionary learning [23]. Because of non-smoothness, many learning frameworks of FFNNs are not able to handle this term [8]. In [27], an algorithm to train sparse autoencoder based on ℓ_1 norm regularization of code vectors has been proposed. A general purpose framework to regularize autoencoders based on non-smooth regularization was proposed in [9] where warm start [28] is used to gradually impose the regularizer term.

In addition to the sparse autoencoder, robustification of deep CNNs is also a potential problem, to which non-smooth regularizer can be impressive [13]. Error back-propagation, an effective method to evaluate the derivative of objective function with respect to network parameters, can expose CNNs to potential adversarial attacks, aiming at maximally changing the network output by imperceptibly manipulating the input patterns [21], [29], [30]. *Regular*, *Fast* and *PGD* are examples of such attacks. A review of different attack scenarios can be found in [31].

Several algorithms have been proposed to robustify an FFNN. The first idea was based on enriching the training patterns using adversarial samples known as *Adversarial Training* [21], [29]. Constraining the network to output similar representations for a pattern and its adversarial counterpart was proposed in *virtual adversarial training* [32]. *Stability training* also utilizes a stability objective $L_{\text{objective}}$ to enforce similarity between the outputs of training patterns and their adversarial counterparts [33]. Among other methods for robustification of deep architectures are *Adversarial Noise Propagation* [34], input transformations [35], total variation norm regularization [35], input super-resolution [36] and knowledge Distillation [37].

Formulation of regular adversary in (1) indicates that regularizing $\|\nabla_{\mathbf{x}} d\left(f\left(\mathbf{p}, \mathbf{x}\right), \mathbf{y}\right)\|_2^2$ defined in (1), could be a generic way for deep architectures robustification [38], which would increase the smoothness of network input-output mapping. Similarly, constraining the Lipschitz constant of the network was also proposed to improve robustness [21], but no practical method to evaluate the Lipschitz constant was presented. In [14] the idea to control the Lipschitz constant of a network for robustification was followed and *Parseval Networks* were proposed. In Parseval networks, convolutional and fully-connected layers are formulated by a matrix product notation as $\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}$, where $\mathbf{z}^{(l)}$ is the vectorized representation in layer l, and $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are weight matrix and bias vector, respectively. The authors argue that the Lipschitz constant of a network can be limited by controlling the layer-wise spectral norm of weight matrices in convolutional and fully-connected layers. Therefore, they suggest the following regularizer for these layers:

$$R_\beta(\mathbf{W}^{(l)}) = \frac{\beta}{2}\left\|\mathbf{W}^{(l)T}\mathbf{W}^{(l)} - \mathbf{I}\right\|_2^2, \tag{4}$$

where, $\mathbf{I}$ is the identity matrix and β is the regularization hyper-parameter.

In [13] two formulations, namely *InfNorm* and *InfInd* are proposed to robustify the deep architecture by controlling the spectral norm of weight matrices in each layer. InfNorm and InfInd use ℓ_∞ norm and ℓ_∞ indicator norm, respectively, to impose Parseval tightness over weight matrices and control the overall Lipschitz constant. These regularizers are non-smooth and a variant of the framework proposed in [9] is utilized to train the network parameters. Both the proposed methods present impressive results for targeted and untargeted attacks, but at the expense of increased training time.

In the sequel of this section, we focus on reviewing the framework proposed in [9] as the building block for the improvements proposed in this paper. The general form of the problem to train an autoencoder under non-smooth regularization can be written as:

$$\min_{\mathbf{p}} E_p\left(\left\{F\left(\mathbf{p}, \mathbf{x}_i\right)\right\}_{i=1}^N, \left\{\mathbf{y}_i\right\}_{i=1}^N\right) + \mathrm{r}_1\left(g(\mathbf{p})\right), \tag{5}$$

where $\mathbf{p}$ is the autoencoder variables vector, $\mathbf{x}_i$ is the i-th input training pattern, $F\left(\mathbf{p}, \mathbf{x}_i\right)$ is the autoencoder output represented by $\widehat{\mathbf{y}}_i$ for the input pattern $\mathbf{x}_i$, $E_p(\cdot,\cdot)$ is a distance metric used to calculate the distance between network output vector $\widehat{\mathbf{y}}_i$ and target vector $\mathbf{y}_i$, N is the number of training patterns, $\mathrm{r}_1(\cdot)$ is a non-smooth regularizer, and $g(\cdot)$ is a function that selects a subset of network variables or parameters. To simplify the notation, the first term in Problem (5) is shown by $E_p(\mathbf{p})$. Then Problem (5) is modified using the following steps to obtain a new relaxed version.

Step 1: Introduce a new vector variable $\mathbf{s}$ as:

$$\min_{\mathbf{p},\mathbf{s}} E_p(\mathbf{p}) + \mathrm{r}_1(\mathbf{s}) \quad \text{s.t. } g(\mathbf{p}) = \mathbf{s}. \tag{6}$$

Problem (6) is equivalent to Problem (5).

Step 2: Relax Problem (6) based on quadratic penalty method as:

$$\min_{\mathbf{p},\mathbf{s}} E_p(\mathbf{p}) + \mathrm{r}_1(\mathbf{s}) + \frac{1}{2\mu}\|g(\mathbf{p}) - \mathbf{s}\|_2^2, \tag{7}$$

where μ is a penalty parameter. We solve a series of subproblems, corresponding to a decreasing sequence of μ. If we denote the solutions to the Problem (7) by $\mathbf{p}^{(\mu)}$ and $\mathbf{s}^{(\mu)}$, and the solution to the Problem (5) by $\mathbf{p}^\star$, given that we find the global minimizer of each subproblem, then [19, Theorem 17.1]:

$$\lim_{\mu\to 0}\mathbf{p}^{(\mu)} = \mathbf{p}^\star, \quad \lim_{\mu\to 0}\mathbf{s}^{(\mu)} = g(\mathbf{p}^\star).$$

Step 3: For convergence purposes, the amplitudes of the network variables are limited by adding the indicator function of the ℓ_∞ norm to the variables vector $\mathbf{p}$ as follows:

$$\min_{\mathbf{p},\mathbf{s}} \mathrm{T}_\mu(\mathbf{p},\mathbf{s}) = \mathrm{E}_\mu(\mathbf{p},\mathbf{s}) + \mathrm{r}_1(\mathbf{s}) + \mathrm{r}_2(\mathbf{p}), \tag{8}$$

where $\mathrm{E}_\mu(\mathbf{p},\mathbf{s}) = E_p(\mathbf{p}) + \frac{1}{2\mu}\|g(\mathbf{p}) - \mathbf{s}\|_2^2$ contains all the smooth terms of the objective function, and $\mathrm{r}_2(\cdot)$ is defined as:

$$\mathrm{r}_2(\mathbf{p}) \triangleq \begin{cases} 0 & \text{if } \|\mathbf{p}\|_\infty \le M \\ \infty & \text{if } \|\mathbf{p}\|_\infty > M \end{cases}. \tag{9}$$

The authors in [9] use an alternating minimization method over the two vector variables to solve Problem (8). For each vector variable, minimization of an upper bound for the second order approximation of the objective function is used to update the variables which results in the following update formulas:

$$\begin{aligned} \mathbf{s}_{k+1} &= \arg\min_{\mathbf{s}} \ \nabla_{\mathbf{s}}^T \mathrm{E}_\mu(\mathbf{p}_k,\mathbf{s}_k)(\mathbf{s}-\mathbf{s}_k) + \frac{1}{2\mu_{s,k}}\|\mathbf{s}-\mathbf{s}_k\|^2 + \mathrm{r}_1(\mathbf{s}), \\ \mathbf{p}_{k+1} &= \arg\min_{\mathbf{p}} \ \nabla_{\mathbf{p}}^T \mathrm{E}_\mu(\mathbf{p}_k,\mathbf{s}_{k+1})(\mathbf{p}-\mathbf{p}_k) + \frac{1}{2\mu_{p,k}}\|\mathbf{p}-\mathbf{p}_k\|^2 + \mathrm{r}_2(\mathbf{p}), \end{aligned} \tag{10}$$

where $\mu_{s,k} \in (0, 1/L_{s,k}]$ and $\mu_{p,k} \in (0, 1/L_{p,k}]$, and $L_{s,k}$ and $L_{p,k}$ are the Lipschitz constants for $\nabla_s \mathrm{E}_\mu(\mathbf{p},\mathbf{s})\Big|_{s_k,p_k}$ and $\nabla_p \mathrm{E}_\mu(\mathbf{p},\mathbf{s})\Big|_{s_{k+1},p_k}$, respectively.

The above framework can be extended in two directions. The first extension is to include a general FFNN, instead of autoencoders, as the architectures that the framework supports. The second extension deals with variable extrapolation to speed up convergence. The effect of extrapolation on convergence speed is studied extensively in [39], [17]. In this paper, we follow the above directions and develop an extrapolation-based learning framework to train FFNNs in the presence of non-smooth regularizers. We then evaluate the new framework comparatively for training sparse autoencoders and robustifying deep CNNs.

IV. Proposed Method

The problem of training an FFNN can be formulated as the problem shown in (5). For the sake of completeness, we restate this problem here:

$$\mathcal{P} : \min_{\mathbf{p}} E_p\left(\{F(\mathbf{p},\mathbf{x}_i)\}_{i=1}^N, \{\mathbf{y}_i\}_{i=1}^N\right) + \mathrm{r}_1\left(g(\mathbf{p})\right). \tag{11}$$

Following similar steps mentioned in the previous section for reaching to (8) starting from Problem (5), we can define a relaxed version of problem $\mathcal{P}$ in (11) based on quadratic penalty method as:

$$\mathcal{P}_\mu : \min_{\mathbf{p},\mathbf{s}} \mathrm{T}_\mu(\mathbf{p},\mathbf{s}) = \mathrm{E}_\mu(\mathbf{p},\mathbf{s}) + \mathrm{r}_1(\mathbf{s}) + \mathrm{r}_2(\mathbf{p}). \tag{12}$$

In the new framework, we extend the update formulas in (10) by using an extrapolated version of variables. Doing so, the new problem for updating variable $\mathbf{s}$ is:

$$\mathbf{s}_{k+1} = \arg\min_{\mathbf{s}} \ \nabla_{\mathbf{s}}^T \mathrm{E}_\mu(\mathbf{p}_k,\widehat{\mathbf{s}}_k)(\mathbf{s}-\widehat{\mathbf{s}}_k) + \frac{1}{2\mu_{s,k}}\|\mathbf{s}-\widehat{\mathbf{s}}_k\|^2 + \mathrm{r}_1(\mathbf{s}), \tag{13}$$

where the extrapolation for variable $\mathbf{s}$ is defined as:

$$\widehat{\mathbf{s}}_k = \mathbf{s}_k + w_{s,k}\cdot(\mathbf{s}_k - \mathbf{s}_{k-1}). \tag{14}$$

Here, $w_{s,k}$ is the extrapolation weight for $\mathbf{s}$ at iteration k. The update formula for variable $\mathbf{p}$ can be derived similarly. Adding a constant term $(1/(2\mu_{s,k}))\|\mu_{s,k}\nabla_s \mathrm{E}_\mu(\mathbf{p}_k,\widehat{\mathbf{s}}_k)\|_2^2$ to the optimization Problem (13), we have the following equivalent form:

$$\mathbf{s}_{k+1} = \arg\min_{\mathbf{s}} \ \frac{1}{2\mu_{s,k}}\|(\mathbf{s}-\widehat{\mathbf{s}}_k) + \mu_{s,k}\nabla_s \mathrm{E}_\mu(\mathbf{p}_k,\widehat{\mathbf{s}}_k)\|_2^2 + \mathrm{r}_1(\mathbf{s}). \tag{15}$$

Using the definition of proximal mapping, the closed-form solution for the vector variables $\mathbf{s}$ and $\mathbf{p}$ can be written as:

$$\begin{aligned} \mathbf{s}_{k+1} &= \mathrm{Prox}_{\mu_{s,k}\mathrm{r}_1}(\widehat{\mathbf{s}}_k - \mu_{s,k}\nabla_s \mathrm{E}_\mu(\mathbf{p}_k,\widehat{\mathbf{s}}_k)), \\ \widehat{\mathbf{s}}_k &\triangleq \mathbf{s}_k + w_{s,k}\cdot(\mathbf{s}_k - \mathbf{s}_{k-1}), \\ \\ \mathbf{p}_{k+1} &= \mathrm{Prox}_{\mu_{p,k}\mathrm{r}_2}(\widehat{\mathbf{p}}_k - \mu_{p,k}\nabla_p \mathrm{E}_\mu(\widehat{\mathbf{p}}_k,\mathbf{s}_{k+1})), \\ \widehat{\mathbf{p}}_k &\triangleq \mathbf{p}_k + w_{p,k}\cdot(\mathbf{p}_k - \mathbf{p}_{k-1}). \end{aligned} \tag{16}$$

In the following, we show that the sequence generated by the update rules in (16) for Problem (12) converges to a critical point of the objective function.

A. Convergence Analysis

In this paper, we consider an FFNN to be composed of fully-connected and convolutional layers, max and average pooling, batch normalization and batch re-normalization and residual modules (see [8] for definition of different elements of FFNN). The activation functions are also constrained to be analytic. As an exception and due to popularity of the non-analytic rectified linear units (ReLU) activation function, we discuss using it as the activation function. The regularizer $\mathrm{r}_1(\cdot)$ is assumed to be proper lower semi-continuous and a KL function.

The first step is to show the Lipschitz continuity of the partial gradients of $\mathrm{E}_\mu(\mathbf{p},\mathbf{s})$ in an FFNN.

Proposition 1. *For a general FFNN with smooth input-output mapping,* $\mathrm{E}_\mu(\mathbf{p},\mathbf{s})$ *has Lipschitz gradients with respect to* $\mathbf{p}$ *and* $\mathbf{s}$*.*

Proof: Following the same procedure as the one in Proposition 1 in [9], one can easily show that for an FFNN without max pooling layer and ReLU activation function, $\mathrm{E}_\mu(\mathbf{p},\mathbf{s})$ is twice differentiable with respect to $\mathbf{p}$ and $\mathbf{s}$. ■

It is to be noted that Proposition 1 limits the application of ReLU activation function and max pooling layer, but one can easily replace them with their smooth approximations, i.e., softplus activation function and Log-Sum-Exp function, respectively [40].

The next step is to show the absolute summability of the sequence generated by the proposed update rules in (16). To proceed, we need the following lemma which can be trivially proved.

Lemma 3. *Let* $\mathbf{x}$*,* $\mathbf{y}$ *and* $\mathbf{z}$ *be arbitrary vectors in* $\mathbb{R}^p$*. Then, the following equality holds:*

$$\|\mathbf{x}-\mathbf{z}\|_2^2 - \|\mathbf{y}-\mathbf{z}\|_2^2 = 2(\mathbf{x}-\mathbf{y})^T(\mathbf{y}-\mathbf{z}) + \|\mathbf{x}-\mathbf{y}\|_2^2.$$

Proposition 2. *Let $\{\mathbf{p}_k, \mathbf{s}_k\}$ be the sequence generated by the update rules given in* (16)*, then we have:*

$$\sum_{k=0}^{\infty} \alpha_k \|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2^2 + \beta_k \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 < \infty \tag{17}$$

for non-negative $\alpha_k = \frac{1}{8} L_{p,k}$ and $\beta_k = \frac{1}{8} L_{s,k}$.

Proof: To show the absolute summability of the sequence generated by the proposed update rules in (16), we use different lemmas presented in [17]. From the Lipschitz continuity of the gradient of $\mathrm{E}_\mu(\mathbf{p}, \mathbf{s})$ with respect to $\mathbf{s}$, we have:

$$\begin{aligned}\mathrm{E}_\mu(\mathbf{p}_k, \mathbf{s}_{k+1}) \leq \mathrm{E}_\mu(\mathbf{p}_k, \mathbf{s}_k) + \nabla_s^T \mathrm{E}_\mu(\mathbf{p}_k, \mathbf{s}_k)(\mathbf{s}_{k+1} - \mathbf{s}_k) \\ + \frac{L_{s,k}}{2} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2. \end{aligned} \tag{18}$$

From (13), we get:

$$\begin{aligned}&\nabla_s^T \mathrm{E}_\mu(\mathbf{p}_k, \widehat{\mathbf{s}}_k)(\mathbf{s}_{k+1} - \widehat{\mathbf{s}}_k) + \frac{1}{2\mu_{s,k}} \|\mathbf{s}_{k+1} - \widehat{\mathbf{s}}_k\|_2^2 + \mathrm{r}_1(\mathbf{s}_{k+1}) \\ \leq &\nabla_s^T \mathrm{E}_\mu(\mathbf{p}_k, \widehat{\mathbf{s}}_k)(\mathbf{s}_k - \widehat{\mathbf{s}}_k) + \frac{1}{2\mu_{s,k}} \|\mathbf{s}_k - \widehat{\mathbf{s}}_k\|_2^2 + \mathrm{r}_1(\mathbf{s}_k). \end{aligned} \tag{19}$$

Summing the inequalities in (18) and (19) while assuming $\mu_{s,k} \leq \frac{1}{2L_{s,k}}$ we have:

$$\begin{aligned}&[\mathrm{E}_\mu(\mathbf{p}_k, \mathbf{s}_k) + \mathrm{r}_1(\mathbf{s}_k) + \mathrm{r}_2(\mathbf{p}_k)] - [\mathrm{E}_\mu(\mathbf{p}_k, \mathbf{s}_{k+1}) + \mathrm{r}_1(\mathbf{s}_{k+1}) + \mathrm{r}_2(\mathbf{p}_k)] \\ \overset{1}{\geq} &[\nabla_s \mathrm{E}_\mu(\mathbf{p}_k, \widehat{\mathbf{s}}_k) - \nabla_s \mathrm{E}_\mu(\mathbf{p}_k, \mathbf{s}_k)]^T (\mathbf{s}_{k+1} - \mathbf{s}_k) \\ &+ 2L_{s,k}(\mathbf{s}_{k+1} - \mathbf{s}_k)^T(\mathbf{s}_k - \widehat{\mathbf{s}}_k) + \frac{L_{s,k}}{2} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 \\ \overset{2}{\geq} &- \left[\|\nabla_s \mathrm{E}_\mu(\mathbf{p}_k, \widehat{\mathbf{s}}_k) - \nabla_s \mathrm{E}_\mu(\mathbf{p}_k, \mathbf{s}_k)\|_2 + 2L_{s,k}\|\mathbf{s}_k - \widehat{\mathbf{s}}_k\|_2\right] \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2 \\ &+ \frac{L_{s,k}}{2} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 \\ \overset{3}{\geq} &- 3L_{s,k} w_{s,k} \|\mathbf{s}_k - \mathbf{s}_{k-1}\|_2 \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2 + \frac{L_{s,k}}{2} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 \\ \overset{4}{\geq} &- 9L_{s,k} w_{s,k}^2 \|\mathbf{s}_k - \mathbf{s}_{k-1}\|_2^2 + \frac{L_{s,k}}{4} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2. \end{aligned} \tag{20}$$

We have four inequalities in (20) that are detailed below. Inequality 1 is deduced from Lemma 3, where inequality 2 comes from the Cauchy-Schwarz inequality. For inequality 3, we use the definition of Lipschitz gradient along with the definition of $\widehat{\mathbf{s}}_k$ in (14). Based on Young's inequality [41], for $\epsilon > 0$, we have $ab \leq \frac{a^2}{2\epsilon} + \frac{\epsilon b^2}{2}$. Inequality 4 results from applying Young's inequality to the first term in penultimate line of (20) with $\epsilon = L_{s,k}/2$, $a = 3L_{s,k} w_{s,k} \|\mathbf{s}_k - \mathbf{s}_{k-1}\|_2$ and $b = \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2$.

We choose the weight $w_{s,k}$ to be related to the Lipschitz constant $L_{s,k}$ and $L_{s,k-1}$ as follows:

$$w_{s,k} \leq \sqrt{\frac{1}{72} \frac{L_{s,k-1}}{L_{s,k}}}. \tag{21}$$

Using this upper-bound of $w_{s,k}$ in (20), we have:

$$\begin{aligned}&\mathrm{T}_\mu(\mathbf{p}_k, \mathbf{s}_k) - \mathrm{T}_\mu(\mathbf{p}_k, \mathbf{s}_{k+1}) \\ \geq &\frac{1}{4} L_{s,k} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 - \frac{1}{8} L_{s,k-1} \|\mathbf{s}_k - \mathbf{s}_{k-1}\|_2^2. \end{aligned} \tag{22}$$

Following a similar procedure for $\mathbf{p}$, we can show the following:

$$\begin{aligned}&\mathrm{T}_\mu(\mathbf{p}_k, \mathbf{s}_{k+1}) - \mathrm{T}_\mu(\mathbf{p}_{k+1}, \mathbf{s}_{k+1}) \\ \geq &\frac{1}{4} L_{p,k} \|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2^2 - \frac{1}{8} L_{p,k-1} \|\mathbf{p}_k - \mathbf{p}_{k-1}\|_2^2. \end{aligned} \tag{23}$$

Summing up the two sides of inequalities in (22) and (23) results in the following inequality:

$$\begin{aligned}&\mathrm{T}_\mu(\mathbf{p}_k, \mathbf{s}_k) - \mathrm{T}_\mu(\mathbf{p}_{k+1}, \mathbf{s}_{k+1}) \\ \geq &\frac{1}{4} L_{s,k} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 - \frac{1}{8} L_{s,k-1} \|\mathbf{s}_k - \mathbf{s}_{k-1}\|_2^2 \\ &+ \frac{1}{4} L_{p,k} \|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2^2 - \frac{1}{8} L_{p,k-1} \|\mathbf{p}_k - \mathbf{p}_{k-1}\|_2^2, \end{aligned} \tag{24}$$

from which, we can write:

$$\begin{aligned}&\mathrm{T}_\mu(\mathbf{p}_0, \mathbf{s}_0) - \mathrm{T}_\mu(\mathbf{p}_{K+1}, \mathbf{s}_{K+1}) \\ \geq &\sum_{k=0}^{K} \left[\frac{1}{8} L_{s,k} \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 + \frac{1}{8} L_{p,k} \|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2^2\right]. \end{aligned} \tag{25}$$

Then, the inequality (17) is followed from (25) for $\alpha_k = \frac{1}{8} L_{p,k}$ and $\beta_k = \frac{1}{8} L_{s,k}$, if $K \to \infty$. ∎

According to Proposition 2, we can conclude that the Euclidean distance between values of each vector variable in successive iterations will tend to zero as $k \to \infty$.

Proposition 3. *The sequence $\{\mathbf{p}_k, \mathbf{s}_k\}$ generated by the update rules* (16) *is bounded.*

Proof: As given in [9], the sequence $\mathbf{p}_k$ is bounded due to the constraint applied by $\mathrm{r}_2(\cdot)$. For the sequence $\mathbf{s}_k$, we start by simplifying the update rule. Using the definition of $\mathrm{E}_\mu(\mathbf{p}, \mathbf{s})$ in (12), we can easily show that $\nabla_s \mathrm{E}_\mu(\mathbf{p}, \mathbf{s}) = \frac{1}{\mu}(\mathbf{s} - g(\mathbf{p}))$ and write:

$$\begin{aligned}\mathbf{s}_{k+1} &= \mathrm{Prox}_{\mu_{s,k} \mathrm{r}_1}\left(\widehat{\mathbf{s}}_k - \frac{\mu_{s,k}}{\mu}\left(\widehat{\mathbf{s}}_k - g(\mathbf{p}_k)\right)\right) \\ &= \mathrm{Prox}_{\mu_{s,k} \mathrm{r}_1}\left(\frac{\mu - \mu_{s,k}}{\mu} \widehat{\mathbf{s}}_k + \frac{\mu_{s,k}}{\mu} g(\mathbf{p}_k)\right). \end{aligned} \tag{26}$$

The boundedness of the $\mathbf{s}_k$ sequence can be shown through an inductive reasoning. Assuming $\mathbf{s}_0 = \mathbf{s}_{-1} = \mathbf{0}$, we can show:

$$\mathbf{s}_1 = \mathrm{Prox}_{\mu_{s,0} \mathrm{r}_1}\left(\frac{\mu_{s,0}}{\mu} g(\mathbf{p}_0)\right), \tag{27}$$

and due to the boundedness of $\{\mathbf{p}_k\}$, $g(\mathbf{p}_k)$ is bounded and so is $\mathbf{s}_1$.

Now, having $\mathbf{s}_k$ and $\mathbf{s}_{k-1}$ bounded, then $\widehat{\mathbf{s}}_k$ is also bounded. Boundedness of $\{\mathbf{p}_k\}$ also guarantees $g(\mathbf{p}_k)$ to be bounded. As $\mu_{s,k} \leq \frac{1}{2L_{s,k}}$ and $L_{s,k} = \frac{1}{\mu}$, we can easily show that the following inequality holds:

$$\frac{1}{2} \leq \frac{\mu - \mu_{s,k}}{\mu} < 1, \; 0 < \frac{\mu_{s,k}}{\mu} \leq \frac{1}{2} \tag{28}$$

Due to the boundedness of $\widehat{\mathbf{s}}_k$ and $g(\mathbf{p}_k)$, and the inequalities in (28), the argument of proximal mapping in (26) is bounded which implies the boundedness of $\widehat{\mathbf{s}}_{k+1}$. ∎

Now, we prove that using the update rule with extrapolation in (16), there exists a subsequence $\{\mathbf{p}_k, \mathbf{s}_k\}_{k \in \mathcal{I}}$ converging to a critical point of the objective function, where $\mathcal{I}$ stands for the subsequence index set.

Theorem 1. *Let $\{\mathbf{p}_k, \mathbf{s}_k\}_{k \geq 1}$ be a sequence generated by the update rules* (16)*. Then:*

1) *There exists a finite limit point (namely $\{\bar{\mathbf{p}}, \bar{\mathbf{s}}\}$) for the sequence.*
2) *Regularizers satisfy the following relations:*

$$\lim_{\mathcal{I} \ni k \to \infty} \mathrm{r}_1(\mathbf{s}_k) = \mathrm{r}_1(\bar{\mathbf{s}}), \lim_{\mathcal{I} \ni k \to \infty} \mathrm{r}_2(\mathbf{p}_k) = \mathrm{r}_2(\bar{\mathbf{p}}).$$

3) Any limit point of the sequence is a critical point of $\mathrm{T}_\mu(\mathbf{p},\mathbf{s})$ *(defined in* (12)*).*

4) If a subsequence $\{\mathbf{p}_k,\mathbf{s}_k\}_{k\in\mathcal{I}}$ *converges to* $\{\bar{\mathbf{p}},\bar{\mathbf{s}}\}$*, then* $\lim_{\mathcal{I}\ni k\to\infty}\mathrm{T}_\mu(\mathbf{p}_k,\mathbf{s}_k)=\mathrm{T}_\mu(\bar{\mathbf{p}},\bar{\mathbf{s}})$.

Proof: Each part of the theorem is proved separately.

1) Due to boundedness of the sequence generated by the update rules (16), Bolzano-Weierstrass theorem [42] implies the existence of a finite limit point for the sequence.

2) $\{\bar{\mathbf{p}},\bar{\mathbf{s}}\}$ is a limit point of $\{\mathbf{p}_k,\mathbf{s}_k\}$, thus there exists a subsequence $\{\mathbf{p}_k,\mathbf{s}_k\}_{k\in\mathcal{I}}$ converging to $\{\bar{\mathbf{p}},\bar{\mathbf{s}}\}$. From Proposition 2, we have $\|\mathbf{p}_{k+1}-\mathbf{p}_k\|\to 0$ and $\|\mathbf{s}_{k+1}-\mathbf{s}_k\|\to 0$. This suggests that for any integer $\gamma\geq 0$, we have:

$$\{\mathbf{p}_{k+\gamma},\mathbf{s}_{k+\gamma}\}_{k\in\mathcal{I}}\to\{\bar{\mathbf{p}},\bar{\mathbf{s}}\}. \tag{29}$$

From (13), the update problem for $\mathbf{s}$ is:

$$\mathbf{s}_{k+1}=\arg\min_{\mathbf{s}}\ \nabla_{\mathbf{s}}^T\mathrm{E}_\mu(\mathbf{p}_k,\widehat{\mathbf{s}}_k)(\mathbf{s}-\widehat{\mathbf{s}}_k)+\frac{1}{2\mu_{s,k}}\|\mathbf{s}-\widehat{\mathbf{s}}_k\|^2+\mathrm{r}_1(\mathbf{s}). \tag{30}$$

Thus we have:

$$\begin{aligned}&\nabla_{\mathbf{s}}^T\mathrm{E}_\mu(\mathbf{p}_k,\mathbf{s}_k)(\mathbf{s}_{k+1}-\widehat{\mathbf{s}}_k)+\frac{1}{2\mu_{s,k}}\|\mathbf{s}_{k+1}-\widehat{\mathbf{s}}_k\|^2+\mathrm{r}_1(\mathbf{s}_{k+1})\\ \leq&\nabla_{\mathbf{s}}^T\mathrm{E}_\mu(\mathbf{p}_k,\widehat{\mathbf{s}}_k)(\mathbf{s}-\widehat{\mathbf{s}}_k)+\frac{1}{2\mu_{s,k}}\|\mathbf{s}-\widehat{\mathbf{s}}_k\|^2+\mathrm{r}_1(\mathbf{s})\ ,\ \forall\mathbf{s}.\end{aligned} \tag{31}$$

As $\mathcal{I}\ni k\to\infty$, the extrapolation term $\mathbf{s}_k-\mathbf{s}_{k-1}$ tends to zero in the ℓ_2 norm sense and thus similar to [9], we have:

$$\begin{aligned}\lim_{\mathcal{I}\ni k\to\infty}\mathrm{r}_1(\mathbf{s}_{k+1})\leq&\nabla_{\mathbf{s}}^T\mathrm{E}_\mu(\bar{\mathbf{p}},\bar{\mathbf{s}})(\mathbf{s}-\bar{\mathbf{s}})+\frac{1}{2\bar{\mu}_s}\|\mathbf{s}-\bar{\mathbf{s}}\|^2\\ &+\mathrm{r}_1(\mathbf{s})\ ,\ \forall\mathbf{s},\end{aligned} \tag{32}$$

where $\bar{\mu}_s$ is a constant in the interval $(0,1/2\bar{L}_s]$ and $\bar{L}_s$ is the Lipschitz constant of $\nabla_{\mathbf{s}}\mathrm{E}_\mu(\mathbf{p},\mathbf{s})$ at $\mathbf{p}=\bar{\mathbf{p}}$ and $\mathbf{s}=\bar{\mathbf{s}}$. Inequality (32) for $\mathbf{s}=\bar{\mathbf{s}}$ results in the following inequality:

$$\lim_{\mathcal{I}\ni k\to\infty}\mathrm{r}_1(\mathbf{s}_{k+1})\leq\mathrm{r}_1(\bar{\mathbf{s}}) \tag{33}$$

Using (33), along with the lower semi-continuity assumption for $\mathrm{r}_1(\cdot)$, leads:

$$\lim_{\mathcal{I}\ni k\to\infty}\mathrm{r}_1(\mathbf{s}_k)=\mathrm{r}_1(\bar{\mathbf{s}}). \tag{34}$$

Similarly, we can show the following:

$$\lim_{\mathcal{I}\ni k\to\infty}\mathrm{r}_2(\mathbf{p}_k)=\mathrm{r}_2(\bar{\mathbf{p}}). \tag{35}$$

3) See Theorem 1 Part 3 in [9].

4) See Theorem 1 Part 4 in [9].

■

Theorem 1 proves that there exists a convergent subsequence to a critical point of $\mathrm{T}_\mu(\mathbf{p},\mathbf{s})$ in the sequence generated by the update rule (16). To generalize subsequence convergence to whole sequence convergence, we show that the KL property holds for the objective function in (12).

Proposition 4. *The objective function* $\mathrm{T}_\mu(\mathbf{p},\mathbf{s})$*, defined in* (12)*, is a KL function.*

Proof: Based on the assumption for the architecture of a FFNN, the input-output mapping is analytic (see [9] for more details). Thus, the smooth term $\mathrm{E}_\mu(\mathbf{p},\mathbf{s})$ in (12) is analytic and has the KL property. Besides, $\mathrm{r}_1(\cdot)$ in (11) has the KL property by assumption. $\mathrm{r}_2(\cdot)$ is also the indicator function of a semi-algebraic set which guarantees its KL property [9]. Thus, all the terms in Problem (12) have KL property which means that $\mathrm{T}_\mu(\mathbf{p},\mathbf{s})$ is a KL function. ■

Algorithm 1 Final Algorithm

procedure SOLVING PROBLEM $\mathcal{P}$ IN (11)
Input: $\{\mathbf{x}_i,\mathbf{y}_i\}$, $\mu_{\max}$, $\mu_{\min}$, s (scale value for μ), M in (9), termination threshold (ζ).
Initialization: $\mathbf{p}_{0,0}=\mathbf{p}_{-1,0}=\mathrm{Xavier}$, $\mathbf{s}_{0,0}=\mathbf{s}_{-1,0}=\mathbf{0}$, $i=0$
Output: FFNN parameters
  $\mu=\mu_{\max}$
  Flag $\leftarrow$ True
  while Flag **do**
    $k=0$
    while $k<N$ **do**
      $\widehat{\mathbf{s}}_{k,i}=\mathbf{s}_{k,i}+w_{s,k,i}(\mathbf{s}_{k,i}-\mathbf{s}_{k-1,i})$
      $\widehat{\mathbf{p}}_{k,i}=\mathbf{p}_{k,i}+w_{p,k,i}(\mathbf{p}_{k,i}-\mathbf{p}_{k-1,i})$
      $\mathbf{s}_{k+1,i}=\mathrm{Prox}_{\mu_{s,k,i}\mathrm{r}_1}(\widehat{\mathbf{s}}_{k,i}-\mu_{s,k,i}\nabla_s\mathrm{E}_\mu(\mathbf{p}_{k,i},\widehat{\mathbf{s}}_{k,i}))$
      $\mathbf{p}_{k+1,i}=\mathrm{Prox}_{\mu_{p,k,i}\mathrm{r}_2}(\widehat{\mathbf{p}}_{k,i}-\mu_{p,k,i}\nabla_p\mathrm{E}_\mu(\widehat{\mathbf{p}}_{k,i},\mathbf{s}_{k+1,i}))$
      $k\leftarrow k+1$
    $\mathbf{p}_{-1,i+1},\mathbf{p}_{0,i+1}\leftarrow\mathbf{p}_{N,i}$
    $\mathbf{s}_{-1,i+1},\mathbf{s}_{0,i+1}\leftarrow\mathbf{s}_{N,i}$
    $i\leftarrow i+1$
    $\mu\leftarrow s\cdot\mu$
    if $\mu<\mu_{\min}$ **then**
      $\mu\leftarrow\mu_{\min}$
    if $\|g(\mathbf{p}_i)-\mathbf{s}_i\|_2^2\leq\zeta$ **then**
      Flag $\leftarrow$ False

The next steps of the proof of convergence are similar to those in [9], which are repeated here to complete the proof. These steps are detailed in theorems 2 and 3 below.

Theorem 2. *Consider Problem* (12) *where* $\mathrm{T}_\mu(\mathbf{p},\mathbf{s})$ *is proper and bounded in* $\mathrm{dom}(\mathrm{T}_\mu)$*,* $\mathrm{E}_\mu(\mathbf{p},\mathbf{s})$ *is continuously differentiable,* r_1 *and* r_2 *are proper lower semi-continuous,* $\mathrm{E}_\mu(\mathbf{p},\mathbf{s})$ *has Lipschitz gradient with respect to both* $\mathbf{p}$ *and* $\mathbf{s}$*, and the problem has a critical point* $\mathbf{x}^*$*, where* $\mathbf{0}\in\partial\mathrm{T}(\mathbf{x}^*)$*. Let* $\{\mathbf{p}_k,\mathbf{s}_k\}_{k\geq 1}$ *be generated from the update rules* (16)*. Assume:*

1) $\{\mathbf{p}_k,\mathbf{s}_k\}_{k\geq 1}$ *has a finite limit point* $\{\bar{\mathbf{p}},\bar{\mathbf{s}}\}$*,*

2) $\mathrm{T}_\mu(\mathbf{p},\mathbf{s})$ *is a KL function.*

Then: $\lim_{k\to\infty}\{\mathbf{p}_k,\mathbf{s}_k\}=\{\bar{\mathbf{p}},\bar{\mathbf{s}}\}$.

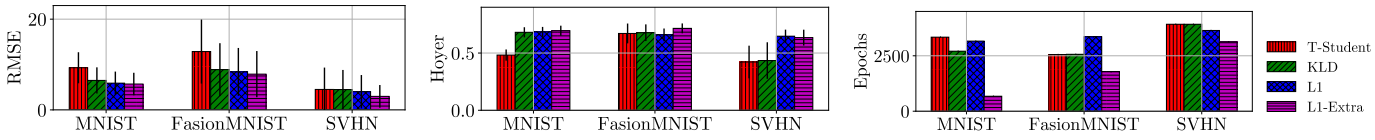
Proof: See Theorem 2.7 in [17]. ■

Using Theorem 2, we can prove that the whole sequence generated by the update rules in (16) converges to a critical point of the objective function in (12).

Theorem 3. *The sequence* $\{\mathbf{p}_k,\mathbf{s}_k\}_{k\geq 1}$ *generated by the update rules in* (16) *converges to a critical point of problem* $\mathcal{P}_\mu$ *defined in* (12)*.*

Proof: See Theorem 3 in [9]. ■

Hence, using update rules in (16), we can solve problem $\mathcal{P}_\mu$ and find one of its critical points. To solve problem $\mathcal{P}$, we



(a) Reconstruction error (b) Hoyer sparseness measure (c) Convergence epochs

Fig. 1: Average and standard deviation of reconstruction error, sparsity (Hoyer) and number of epochs needed for convergence over test set of benchmark datasets.

solve a series of the problem $\mathcal{P}_\mu$ with decreasing values for μ based on warm-start [28], as shown in Algorithm 1. In this algorithm, each problem $\mathcal{P}_\mu$ is indexed by the variable i and thus this variable appears in the update rules and extrapolated variables calculations.

It should be noted that while $L_{s,k}$ can easily be calculated, computing $L_{p,k}$ and consequently $w_{p,k}$ is not straightforward, due to the complex network architecture in FFNNs. This necessitates to employ a backtracking procedure. More implementation details could be found in [9].

V. Experimental results

In this section, we evaluate the performance of the proposed framework summarized in Algorithm 1 to train the parameters of FFNNs. As also done in [9], sparse autoencoder training is used to measure efficiency of the framework. We also study the effect of extrapolation weight parameter in the update rules (16). In the sequel, we consider two more complex CNNs and use the updated framework for the robustification based on the formulations given in [13].

TABLE I: Performance comparison over clean test set for C-CNN architecture. $\mathrm{SN}_{\max}(l)$, $\mathrm{SN}_{\min}(l)$ represent the maximum and minimum spectral norm across all convolutional and fully-connected layers, respectively.

	Method	Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra
SVHN	Accuracy	93.07	93.14	93.42	93.40	**93.45**
	Iteration	85	72	77	67	**57**
	Time (ms)	**0.13**	0.22	0.31	0.25	0.26
	$\mathrm{SN}_{\max}(l)$	**1.00(F1)**	1.01($F1$)	1.01($F1$)	1.01($F1$)	1.02($C2$)
	$\mathrm{SN}_{\min}(l)$	**1.00(F3)**	1.00($F3$)	1.00($F3$)	1.00($F3$)	1.00($F3$)
CIFAR10	Accuracy	77.14	77.93	78.11	78.29	**78.52**
	Iteration	72	57	84	**51**	71
	Time (ms)	**0.13**	0.21	0.18	0.24	0.23
	$\mathrm{SN}_{\max}(l)$	1.01($F1$)	1.00($F1$)	**1.00(F1)**	1.00($F1$)	1.00($F1$)
	$\mathrm{SN}_{\min}(l)$	1.00($F3$)	1.00($F3$)	**1.00(F3)**	1.00($F3$)	1.00($F3$)
CIFAR100	Accuracy	42.06	42.37	42.33	**43.62**	43.12
	Iteration	122	98	94	93	**63**
	Time (ms)	**0.24**	0.24	0.31	0.43	0.99
	$\mathrm{SN}_{\max}(l)$	1.03($F1$)	1.03($F1$)	1.03($F1$)	**1.01(F1)**	1.02($F1$)
	$\mathrm{SN}_{\min}(l)$	1.01($F3$)	1.01($F3$)	1.01($C1$)	**1.00(F2)**	1.01($F3$)

A. Sparse Autoencoder

Training a sparse autoencoder can be formulated using problem $\mathcal{P}$ in (11) with $\mathrm{r}_1(g(\mathbf{p})) = \lambda \sum_i \|\mathbf{z}_i\|_1$ and $\mathbf{z}_i$ being the code representation of the i-th training pattern. This option satisfies all the conditions needed for $\mathrm{r}_1(\cdot)$ in the proposed framework. This means that we can use Algorithm 1 to train a sparse autoencoder. We use three benchmark datasets, MNIST [43], Fashion-MNIST [44] and SVHN [45], and train a sparse autoencoder over each of them using KLD [22], smooth approximation of ℓ_1 norm by Student-t penalty (T-Student) [24], ℓ_1 regularizer based on framework proposed in [9] (L1), and Algorithm 1 (L1-Extra).

In this experiment, the input vector size is 784 for the MNIST and FasionMNIST datasets and 1024 for the SVHN dataset. The code vector length is set to 144 for all datasets and the sigmoid function is used for the encoder and decoder activation functions. We use a multiplicative schedule to change learning rate and penalty parameter. The extrapolation weight of 0.9 is used in all datasets. Initial penalty parameter and penalty factor are 20 and 0.1, respectively, while penalty parameters is clipped at 0.01 for all datasets. Figure 1 shows the training results over three benchmark datasets using three different metrics (reconstruction error via root mean square error, sparsity via Hoyer measure [46], and convergence speed via epochs needed for convergence). Here, the extrapolation with weight 0.9 is used for the network variable vector $\mathbf{p}$ and the vector $\mathbf{s}$ is updated using the rule proposed in [9]. The updated algorithm significantly outperforms other methods in all three metrics on MNIST and Fashion-MNIST datasets, while it presents just almost equal sparsity in Hoyer measure, relative to L1, on SVHN. The performance improvement achieved confirms that using extrapolation, which results in a convergent learning framework, can also boost the learning convergence speed in the FFNNs architectures.

Figure 2 shows the learning curve of the framework proposed in [9] with updated version presented in this paper. It is clear that extrapolation has significantly decreased the number of iterations required for convergence and final value of the objective function. For instance, the numbers of iterations needed for SVHN-L1 and SVHN-L1-Extra to reach the same objective value are about 3500 and 500, respectively. This makes the updated framework more applicable to real world applications.

To explore effect of the extrapolation weight on the performance of the proposed framework, we adopt a $\mathcal{P}_\mu$ problem with $\mu = 1$ and solve the resulting problem using the update rules proposed in (16). Figure 3 shows the learning curve over test samples for different datasets. As shown, fixing all parameters of the optimization, the learning performance in terms of convergence iterations and final objective function value is monotonically improved by increasing the extrapolation weight toward unity.

B. Robust CNNs

Thus far, we have shown how extrapolation can improve performance of the sparse autoencoder training. Now, we

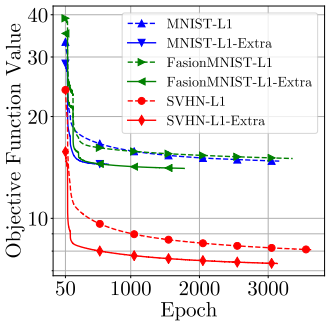
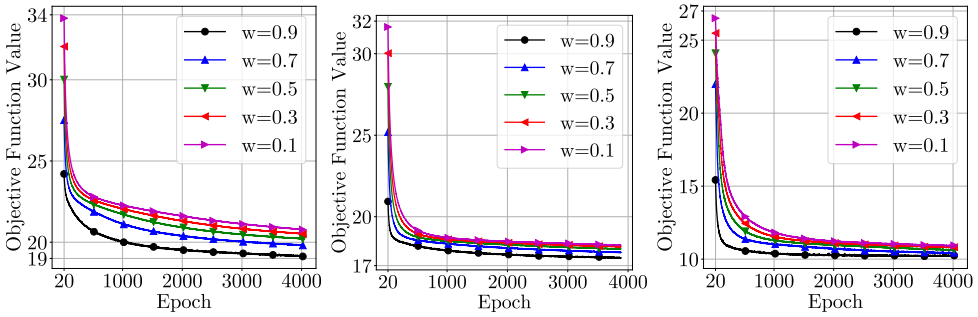


Fig. 2: Comparison of learning objective function with and without extrapolation.



(a) MNIST dataset (b) FasionMNIST dataset (c) SVHN dataset

Fig. 3: The effect of extrapolation weight in solving problem $\mathcal{P}_\mu$ defined in (12) over different datasets.

TABLE II: Performance of robust training of C-CNN architecture against different untargeted attacks over benchmark datasets.

Attack type		Random					Fast					Regular					PGD				
Method		Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra	Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra	Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra	Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra
SVHN	40	93.02	93.11	93.44	93.41	**93.44**	65.56	64.18	64.13	67.21	**68.34**	51.63	50.17	50.74	54.00	**55.52**	51.49	50.05	50.60	53.88	**55.21**
	50	93.05	93.13	**93.43**	93.42	93.43	85.48	84.94	85.26	86.08	**86.32**	80.71	79.87	80.13	81.59	**82.04**	81.04	80.32	80.50	81.79	**82.28**
	60	93.06	93.13	93.42	93.40	**93.43**	90.93	90.83	91.15	91.36	**91.48**	89.69	89.47	89.88	90.11	**90.27**	89.90	89.69	90.09	90.32	**90.45**
	70	93.07	93.14	93.42	93.40	**93.44**	92.41	92.46	92.75	92.80	**92.87**	92.06	92.07	92.40	92.46	**92.54**	92.11	92.10	92.44	92.49	**92.56**
CIFAR10	40	77.10	77.93	78.02	78.22	**78.41**	37.82	**39.41**	37.81	39.04	37.98	25.00	**26.25**	25.12	25.96	24.84	24.73	**25.88**	24.79	25.65	24.54
	50	77.12	77.87	78.11	78.34	**78.54**	62.96	64.06	63.58	**64.21**	63.79	56.04	56.87	56.80	**57.50**	55.73	55.53	56.51	56.29	**56.95**	55.46
	60	77.13	77.93	78.13	78.26	**78.52**	72.42	73.29	73.45	**74.04**	73.71	70.15	70.77	71.06	**71.59**	71.24	70.27	70.91	71.20	**71.65**	71.36
	70	77.14	77.91	78.13	78.29	**78.53**	75.56	76.47	76.56	76.80	**77.02**	74.80	75.72	75.78	76.08	**76.22**	74.82	75.74	75.81	76.17	**76.27**
CIFAR100	40	42.03	42.30	42.45	**43.58**	43.09	10.38	11.40	12.29	12.54	**15.86**	5.60	6.04	7.07	7.26	**9.22**	5.51	5.85	6.74	7.04	**7.69**
	50	42.04	42.35	42.40	**43.69**	43.10	26.53	27.15	27.80	28.79	**30.66**	20.60	21.81	22.30	23.02	**25.94**	20.54	21.60	22.38	23.08	**25.95**
	60	42.05	42.36	42.37	**43.66**	43.09	36.29	36.58	36.52	38.26	**38.74**	33.51	34.24	34.33	35.69	**36.75**	33.82	34.55	34.53	36.05	**36.99**
	70	42.05	42.37	42.35	**43.65**	43.13	39.86	40.59	40.30	**41.96**	41.62	39.04	39.56	39.28	**41.02**	40.87	39.17	39.71	39.45	**41.16**	40.97

TABLE III: Performance of robust training of Mobilenetv2 architecture against tageted attacks over benchmark datasets.

Attack type		CIFAR10-PGD(ℓ_0)					CIFAR10-PGD($\ell_0+\ell_\infty$)					CIFAR100-PGD(ℓ_0)					CIFAR100-PGD($\ell_0+\ell_\infty$)				
Method		Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra	Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra	Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra	Parseval	InfInd	InfNorm	InfInd-Extra	InfNorm-Extra
Metrics	Accuracy	93.86	93.79	93.93	**94.91**	94.76	93.86	93.79	93.93	**94.91**	94.76	75.87	75.99	75.79	76.79	**77.11**	75.87	75.99	75.79	76.79	**77.11**
	Time	0.40	0.38	0.42	1.76	**2.58**	0.53	0.44	0.53	2.30	**3.29**	0.38	0.43	0.38	**0.87**	0.78	0.68	0.73	0.64	**1.42**	1.28
	ℓ_0	173.7	171.8	178.8	194.6	**203.0**	230.2	229.8	233.5	241.7	**245.9**	211.8	212.0	211.1	228.4	**229.0**	257.3	257.2	255.6	263.7	**263.7**
	ℓ_1	10.1	10.2	10.6	11.3	**12.3**	7.4	7.4	7.6	9.8	**10.7**	**13.6**	13.1	13.4	12.7	13.1	8.8	8.9	8.6	**9.5**	9.5
	ℓ_2	0.81	0.82	0.84	0.86	**0.92**	0.48	0.48	0.50	0.63	**0.69**	**1.00**	0.96	0.98	0.91	0.94	0.55	0.56	0.54	**0.60**	0.60
	ℓ_∞	0.145	0.144	0.146	0.154	**0.163**	0.035	0.035	0.036	0.058	**0.066**	**0.166**	0.159	0.165	0.155	0.161	0.039	0.041	0.038	**0.051**	0.050

continue with investigating the extrapolation performance in training robust CNNs. In [13], the framework proposed in [9] is used to impose Parseval tightness over weight matrices of convolutional and fully-connected layers to robustify the architecture similar to that of Parseval networks [14]. The regularizers used in [9] are:

$$\text{Infinity Norm Indicator}: \mathrm{r}_1(g(\mathbf{p})) = \sum_{i\in\mathcal{S}} \mathcal{I}\left\{\|\mathbf{W}^{(l)T}\mathbf{W}^{(l)} - \mathbf{I}\|_\infty \le \xi\right\}$$

$$\text{Infinity Norm}: \mathrm{r}_1(g(\mathbf{p})) = \lambda \sum_{i\in\mathcal{S}} \|\mathbf{W}^{(l)T}\mathbf{W}^{(l)} - \mathbf{I}\|_\infty, \quad (36)$$

where λ and ξ are hyperparameters, $\mathbf{W}^{(l)}$ is the weight matrix for convolutional and fully-connected layers in the vectorized setting, $\mathbf{I}$ is the identity matrix with proper dimension and $\mathcal{S}$ is the set of regularized layers. Both regularizers in (36) meet the conditions needed to apply Algorithm 1 and the proximal mapping presented in Lemma 1 and Lemma 2. Henceforth, the proposed framework can be employed to train robust CNNs.

We train a custom CNN (C-CNN) architecture over SVHN [45], CIFAR10 and CIFAR100 [47] datasets. Let $Ck(D, N, F)$ represent convolutional layer k with depth D, filter number N, filter size F, and let $Fk(I, O)$ show fully connected layer k with input dimension I and output dimension O. Using these notations, the architecture for C-CNN used in this paper consists of six layers C1(3,30,5), C2(30,100,5), C3(100,500,3), F1(2000,500), F2(500,100), and F3(100,C), respectively, and each layer is followed by a batch normalization layer, where non-overlapping max-pooling with filter size of 2 is employed after the convolutional layers. Variable C in the last layer represents the number of classes in the dataset. Finally, the network is terminated by a SoftMax layer. The network is

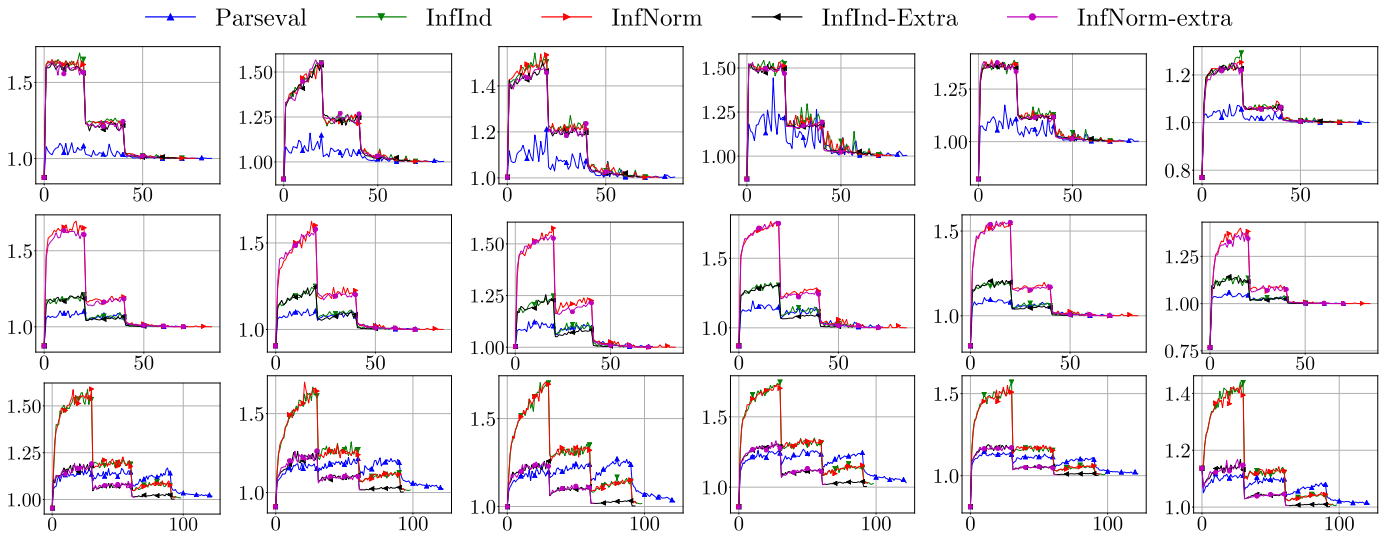


Fig. 4: Spectral norm for convolutional and fully-connected layers of C-CNN architecture (The rows correspond to SVHN, CIFAR10, and CIFAR100 datasets, respectively, and columns correspond to C1, C2, C3, F1, F2 and F3, respectively.)

trained using a multiplicative schedule for both learning rate and penalty parameters. The triplet consisting of initial, final, and factor for the learning rate, respectively, is (10^3,1,0.3) for both methods over SVHN and InfNorm method over CIFAR10 dataset, while it is (10^2,0.1,0.3) for both methods over CIFAR100 and InfInd method over CIFAR10 dataset.

Table I shows the training and generalization performance over clean test set for different methods including Parseval [14], InfInd [13], InfNorm [13], InfInd-Extra (first regularizer in (36)) and InfNorm-Extra (second regularizer in (36)). All methods can almost perfectly recover the spectral norm throughout the layers with unit value, and the result of the proposed Algorithm 1 outperforms the result provided by the framework given in [13].

Table II compares different methods in terms of their robustness against untargeted attacks and random perturbation (Random) with Gaussian distribution for four SNR values of 40-70 dB. As shown, the result provided by Algorithm 1 outperforms the results of other methods on most of the datasets and the SNR values.

Figure 4 presents the spectral norm along with training epochs using different methods on all selected datasets. We observe that the learning starts by increasing the spectral norm, and then the regularizer plays its role to control the spectral norm and decrease it. An important point is that Algorithm 1 works better than the framework proposed in [9] and, in some cases, they present almost equal performance. This shows that extrapolation leads to improved convergence speed while the resulting accuracy over test samples and spectral norm are also improved, as compared to the Parseval network and results presented in [13].

In the sequel, we evaluate the performance of robust training algorithms over CIFAR10 and CIFAR100 datasets for Mobilenetv2 architecture proposed in [48] against PGD(ℓ_0) and PGD($\ell_0+\ell_\infty$) targeted adversarial attacks introduced in [49]. We regularize all convolutional and fully-connected layers in Mobilenetv2. To train Mobilenetv2 in all of the experiments, the values for $\mu_{\max}$, $\mu_{\min}$ and w are 10^6, 10^2, and 0.95, respectively. The learning rate follows the cosine annealing schedule, with starting value of 0.01 and 0.012 for CIFAR10 and CIFAR100 datasets, respectively. For InfInd method, the value of ξ sets to 0.2 and 0.1 for CIFAR10 and CIFAR100 datasets, respectively, while in InfNorm mehtod, the values of 0.1 is selected for λ in both datasets.

For CIFAR10, we randomly select 100 images per class and generate attacks to all 9 wrong labels which results in a total number of 9000 attacks. In the CIFAR100 dataset, we select 10 images per class and generate attacks to 10 randomly selected wrong labels resulting in 10000 attacks. Table III represents the accuracy of methods over clean test set and average attack metric over all attacks. The proposed methods present superior performance in accuracy over clean test samples. At the same time the adversary needs much more time to attack the architectures trained using the proposed methods while adversarial perturbation is more perceptible in the proposed methods in terms of different distortion metrics including ℓ_0, ℓ_1, ℓ_2 and ℓ_∞ norms in most of the cases.

VI. Conclusion

This paper has improved the framework presented in [9] by equipping it with extrapolation to boost its learning performance. While the framework given in [9] is devoted to the case of autoencoders, our new framework extends it in two major directions. First, our framework can be employed in a general case of FFNN that may include the popular family of CNNs. Second, by using extrapolation for updating the learning parameters, the new framework significantly enhances the learning performance in terms of convergence speed and final objective function value. We evaluated the proposed framework in two applications, including sparse autoencoder training and robustifying CNNs, and compared the results with those of the previous frameworks [9], [13], over three benchmark datasets. The results of our comparative study show the superiority of our proposed framework based on certain measures of the system performance, including reconstruction error, Hoyer sparseness, convergence speed and robustness.

ACKNOWLEDGMENT

This research has been supported by the Iran's National Elites Foundation and we acknowledge this support.

REFERENCES

[1] C. Tian, Y. Xu, and W. Zuo, "Image denoising using deep cnn with batch renormalization," *Neural Networks*, vol. 121, pp. 461–473, 2020.
[2] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[3] E. Gentet, B. David, S. Denjean, G. Richard, and V. Roussarie, "Neutral to lombard speech conversion with deep learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7739–7743.
[4] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.
[5] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[7] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[8] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
[9] S. Amini and S. Ghaemmaghami, "A new framework to train autoencoders through non-smooth regularization," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1860–1874, 2019.
[10] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, "Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 12, pp. 2486–2498, 2016.
[11] S. Amini and S. Ghaemmaghami, "Lowering mutual coherence between receptive fields in convolutional neural networks," *Electronics Letters*, 2018.
[12] M. Sadeghi and M. Babaie-Zadeh, "Dictionary learning with low mutual coherence constraint," *Neurocomputing*, vol. 407, pp. 163–174, 2020.
[13] S. Amini and S. Ghaemmaghami, "Towards improving robustness of deep neural networks to adversarial perturbations," *IEEE Transactions on Multimedia*, 2020.
[14] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 854–863.
[15] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, pp. 1–39, 2020.
[16] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.
[17] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *Journal of Scientific Computing*, pp. 1–35, 2017.
[18] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, "Accelerated methods for non-convex optimization," *arXiv preprint arXiv:1611.00756*, 2016.
[19] S. Wright and J. Nocedal, "Numerical optimization," *Springer Science*, vol. 35, pp. 67–68, 1999.
[20] K. Kurdyka, "On gradients of functions definable in o-minimal structures," in *Annales de l'institut Fourier*, vol. 48, no. 3, 1998, pp. 769–783.
[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2014.
[22] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.
[23] M. Elad, *Sparse and redundant representation*. Springer, 2010.
[24] N. Jiang, W. Rong, B. Peng, Y. Nie, and Z. Xiong, "An empirical analysis of different sparse penalties for autoencoder in unsupervised feature learning," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.
[25] J. Chorowski and J. M. Zurada, "Learning understandable neural networks with nonnegative weight constraints," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 1, pp. 62–69, 2015.
[26] T. Bittar, J.-P. Chancelier, and M. De Lara, "Best convex lower approximations of the l 0 pseudonorm on unit balls," *arXiv preprint arXiv:2105.14983*, 2021.
[27] S. Amini and S. Ghaernmaghami, "Sparse autoencoders using non-smooth regularization," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 2000–2004.
[28] E. T. Hale, W. Yin, and Y. Zhang, "Fixed-point continuation for ℓ_1-minimization: Methodology and convergence," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1107–1130, 2008.
[29] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, vol. abs/1412.6572, 2015.
[30] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.
[31] N. Akhtar and A. S. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.
[32] T. Miyato, A. M. Dai, and I. J. Goodfellow, "Adversarial training methods for semi-supervised text classification," in *ICLR 2017*, 2016.
[33] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 4480–4488.
[34] A. Liu, X. Liu, C. Zhang, H. Yu, Q. Liu, and J. He, "Training robust deep neural networks via adversarial noise propagation," *arXiv preprint arXiv:1909.09034*, 2019.
[35] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Enhancing robustness of machine learning systems via data transformations," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2018, pp. 1–5.
[36] A. Mustafa, S. H. Khan, M. Hayat, J. Shen, and L. Shao, "Image super-resolution as a defense against adversarial attacks," *CoRR*, vol. abs/1901.01677, 2019.
[37] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.
[38] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[39] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.
[40] G. C. Calafiore, S. Gaubert, and C. Possieri, "Log-sum-exp neural networks and posynomial models for convex and log-log-convex data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 3, pp. 827–838, 2019.
[41] W. H. Young, "On classes of summable functions and their fourier series," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 87, no. 594, pp. 225–229, 1912.
[42] H. H. Sohrab, *Basic real analysis*. Springer, 2003, vol. 231.
[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[45] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, 2011.
[46] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of machine learning research*, vol. 5, no. Nov, pp. 1457–1469, 2004.
[47] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
[49] F. Croce and M. Hein, "Sparse and imperceivable adversarial attacks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4724–4732.