Signal Denoising Using a New Class of Robust Neural Networks

Ana Neacsu[1,2§], Kavya Gupta[2§], Jean-Christophe Pesquet[2], and Corneliu Burileanu[1]
[1] *Speech and Dialogue Laboratory*
University Politehnica of Bucharest, Bucharest, Romania
[2] *Université Paris-Saclay, CentraleSupélec,*
Centre de Vision Numérique, Inria, Gif-sur-Yvette, France

***Abstract*—In this work, we propose a novel neural network architecture, called Adaptive Convolutional Neural Network (ACNN), which can be viewed as an intermediate solution between a standard convolutional network and a fully connected one. A constrained training strategy is developed to learn the parameters of such a network. The proposed algorithm allows us to control the Lipschitz constant of our ACNN to secure its robustness to adversarial noise. The resulting learning approach is evaluated for signal denoising based on a database of music recordings. Both qualitative and quantitative results show that the designed network is successful in removing Gaussian noise with unknown variance.**

***Index Terms*—stability, fully connected networks, audio denoising, perturbations.**

I. Introduction

Convolutive Neural Networks (CNNs) and their variants have been applied ubiquitously to a variety of learning problems in various domains such as image classification [1], [2], face recognition [3], object tracking [4], natural language processing [5], speech enhancement [6], [7] etc. Despite their massive success with respect to fully connected neural networks (FCNs), CNNs are difficult to analyze theoretically and thus raise some robustness issues.

Deep neural networks (DNNs) are known to be potentially very sensitive to small perturbations of the input, which can mislead the network and decrease its performance [8], [9]. The main challenge nowadays is to develop high-performance systems that are reliable and safe; this aspect is even more critical while developing real-life systems such as autonomous cars, aircraft flight control, voice-controlled systems [10], [11], etc. A well-known metric which is used in the literature to assess the robustness of neural networks to small perturbations is their Lipschitz constant [12]–[15]. This constant provides an upper bound on the ratio between output and input variations for a given distance, thus being a measure of the sensitivity of the non-linear function of interest with respect to input perturbations. An accurate computation of the Lipschitz bounds can be useful in either assessing robustness of the trained model [16], [17] or favoring robustness during the training of the model [18], [19]. However, as DNNs have highly complex and non-linear structures, an accurate and efficient estimation of the Lipschitz constant remains a challenge. In the recent literature [15], [20], different techniques have been proposed to derive Lipschitz bounds for deep feed-forward neural networks using non-expansive activation operators. These techniques may, however, be challenging to apply to CNNs.

This work introduces a new class of neural networks, which can be seen as intermediate between CNNs and FCNs. Learning capabilities of CNNs being well-investigated and proven, we take advantage of this potential by structuring the weights of our network in a similar manner. A significant difference is that the network makes use of filters which are no longer time/space invariant, similarly to what is done in adaptive filtering. Thus, such network architecture appears more flexible. In addition, we show that we can obtain tight Lipschitz certificates for ensuring its robustness. More precisely, we control the Lipschitz constant of the network to reach a good performance-robustness balance by training the system under suitable spectral norm constraints. Our approach focuses on neural networks with positive weights [21], for which we can efficiently compute an optimal Lipschitz constant [20]. We demonstrate the good performance of the designed architecture for denoising music signals corrupted with various levels of noise.

The rest of the paper is organized as follows. The theoretical background of our work and a literature review are presented in Section II. Section III describes our proposed architecture and then develops a constrained optimization method to train it. The applications and experimental results are reported in Section IV. Some concluding remarks are drawn in Section V.

II. Background

A. Related Work

Many recent developments in the field of deep learning emphasize the importance of convolutions in neural network architectures [22] and the improvement in their performance [23]. To the best of our knowledge, there are few works that propose ways for improving FCN architectures. In [24], the performance increase is achieved by reducing the sparsity of the network and improving the gradient flow in it. In our work, we also aim at improving the performance of a FCN, but we follow a different approach. The linear layers of our network have a structure similar to convolutive ones, in the sense that

[§]Equal contribution

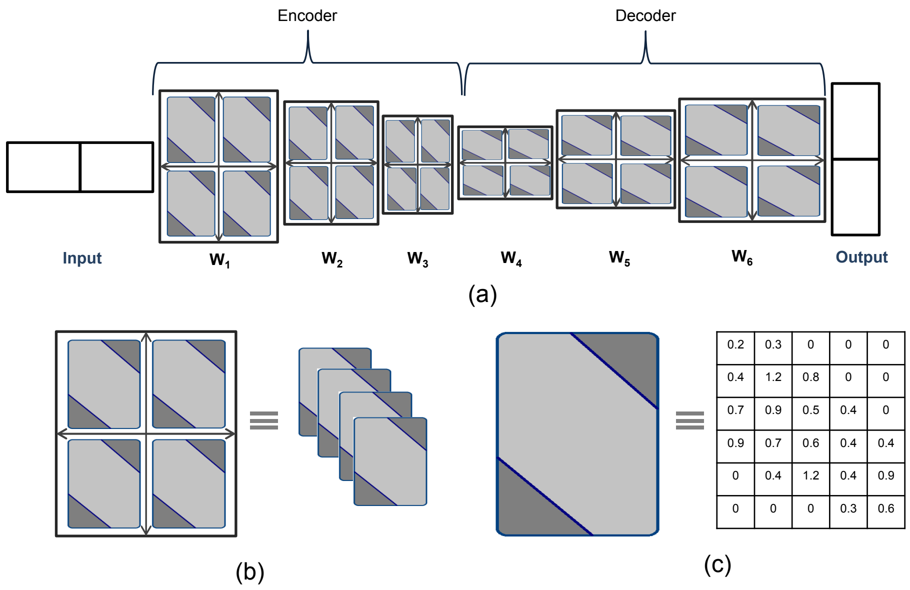

Fig. 1. Proposed architecture of Adaptive Convolutional Neural Network (ACNN). a) An encoder-decoder architecture composed of a 6-layer FCN followed by ReLU activation function. b) Relation between proposed FCNs and CNNs; the weights are split into sub-matrices simulating convolutive filters in CNNs c) Each of the sub-matrices is constrained to have a band structure as shown in this example. The dark grey area marks the zero-entries, while the light-grey colour corresponds to the ones that are allowed to be non-zero.

each neuron input is a weighted sum of the outputs of a given limited number of neurons in the preceding layer and the processing is split into multiple channels operating in a parallel manner. In contrast to convolutive layers, ACNN uses multiple kernels at each channel leading to more flexibility in the choice of the weights. Not only we reach a good performance, but we also keep it robust to input perturbations by providing Lipschitz certificates.

B. Quantifying the Robustness of the Network

Consider an m-layer feedforward neural network T, having N_i neurons at layer $i \in \{1, \dots, m\}$. Let $x \in \mathbb{R}^{N_0}$ be the input and $T(x) \in \mathbb{R}^{N_m}$ the associated output. If the input is altered by an additive perturbation vector $z \in \mathbb{R}^{N_0}$, the error on the perturbation of the output is upper bounded as follows:

$$\|T(x+z) - T(x)\| \leq \theta_m \|z\|, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm and $\theta_m \geq 0$ is the Lipschitz constant of the network. The smaller this constant, the more robust the network with respect to perturbations. Thus θ_m allows us to assess the sensitivity of the network to adversarial examples. A standard Lipschitz constant estimate [25] is given by

$$\theta_m = \prod_{i=1}^{m} \|W_i\|_{\mathrm{S}}, \tag{2}$$

where $\|\cdot\|_{\mathrm{S}}$ denotes the *spectral norm*. However, this bound is often over-pessimistic and tighter Lipschitz constant estimates have been derived in the literature [20].

If all the weights of a feed-forward neural network are non-negative, an optimal Lipschitz constant reduces to $\|W_m \cdots W_1\|_{\mathrm{S}}$ [20]. In other words, it has the same Lipschitz constant as a linear network where the identity function is substituted for all the activation operators. For a network with weights having arbitrary signs, $\vartheta_m = \|W_m \cdots W_1\|_{\mathrm{S}}$ only constitutes a lower bound on the Lipschitz constant [20].

III. Proposed approach

A. Making the bridge between CNNs and fully-connected networks

In this work, we aim at filling the gap between FCNs and CNNs. In terms of signal processing concepts, a convolutive layer is a Multiple-Input Multiple-Output (MIMO) filter. For one-dimensional signals, each of these filters can be viewed as a Tœplitz matrix generated by the impulse response of the filter, which is applied to the vector of signal samples. If the filter length is short, large upper and lower triangular parts of this matrix are null. In our proposed approach, we will keep this band structure for the weight matrix, which is equivalent to performing local processing at each time within a sliding window. However, in order to add more flexibility in this architecture, we will allow all the nonzero coefficients of this matrix to be fully optimized.

The proposed architecture is depicted in Figure 1. Figure 1.b is the graphical representation of the presented concept. As it can be observed, the weights are split to emulate a MIMO system. Each kernel is associated with a specific shape of the matrix, which is depicted in Figure 1.c. The lower and upper triangular null parts are displayed in dark gray, while the light gray central part contains overlap and may use different weight values.

B. Learning algorithm

For training the proposed ACNN (Adaptive Convolutional Neural Network), we use a stochastic gradient-like optimization based on the popular ADAM method [26]. Consider the vector of parameters of the network, $\eta = (\eta_i)_{1\leq i\leq m}$, such that, for each layer $i \in \{1,\dots,m\}$, η_i represents a vector of dimension $N_i(N_{i-1}+1)$, composed of the elements of weight matrix W_i and the components of the bias vector b_i.

To secure the conditions of robustness while imposing the desired structure for our network, the parameter vector η is projected onto a closed set $\mathcal{S}$ that expresses all these constraints. The parameter update at epoch $n > 0$ is performed for mini-batches $(\mathbb{M}_{q,n})_{1\leq q\leq Q}$. If the training data are denoted by $(z_k)_{1\leq k\leq K}$, where z_k is the k-th pair of inputs and their associated outputs, the operations performed during the n-th epoch reads:

```
Partition $\{1,\dots,K\}$ into mini-batches $(\mathbb{M}_{q,n})_{1\leq q\leq Q}$
For every $q \in \{1,\dots,Q\}$
    $t = (n-1)Q + q$
    For every $i \in \{1,\dots,m\}$
        $g_{i,t} = \sum_{k\in\mathbb{M}_{q,n}} \nabla_i \ell\big(z_k, (\eta_{i,t})_{1\leq i\leq m}\big)$
        $\mu_{i,t} = \beta_1\mu_{i,t-1} + (1-\beta_1)g_{i,t}$
        $\nu_{i,t} = \beta_2\nu_{i,t-1} + (1-\beta_2)g_{i,t}^2$
        $\gamma_t = \gamma\sqrt{1-\beta_2^t}/(1-\beta_1^t)$
        $\eta_{i,t+1} = \mathsf{P}_{\mathcal{S}_{i,t}}\Big(\eta_{i,t} - \gamma_t\mu_{i,t}/(\sqrt{\nu_{i,t}}+\epsilon)\Big)$,
```

where the square, the square root, and the division are performed component-wise, and

$$\mathcal{S}_{i,t} = \big\{\eta_i \mid [(\eta_{j,t+1}^\top)_{j<i} \ \ \eta_i^\top \ \ (\eta_{j,t}^\top)_{j>i}]^\top \in \mathcal{S}\big\}. \tag{3}$$

In the presented algorithm, ℓ denotes the loss function, ∇_i represents the gradients with respect to η_i; $\mu_{i,t}$ and $\nu_{i,t}$ represent the first and second momentum estimates at the iteration t, initialized with $\mu_{i,0} = \nu_{i,0} = 0$. $\mathsf{P}_{\mathcal{S}_{i,t}}$ designates the projection onto the constraint set $\mathcal{S}_{i,t}$. Although the set $\mathcal{S}$ is non-convex, the sets $\mathcal{S}_{i,t}$ will be defined as the intersection of three closed and convex constraint sets, as detailed next. The parameters used for learning are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\gamma = 0.001$, and $\epsilon = 10^{-12}$.

To ensure the computation of a tight robustness bound, as explained in Section II, we impose non-negative weights for every $i \in \{1,\dots,m\}$ by considering the constraint set:

$$\mathcal{D}_i = \{W_i \in \mathbb{R}^{N_i\times N_{i-1}} \mid W_i \geq 0\} \tag{4}$$

Let R_i (resp. Q_i) be the number of output (resp. input) channels used in layer $i \in \{1,\dots,m\}$. The proposed algebraic structure of each weight operator is guaranteed by splitting the corresponding matrix W_i into $R_i \times Q_i$ sub-matrices of dimension $N_i' \times M_i'$ (with $N_i' = N_i/R_i$ and $M_i' = N_{i-1}/Q_i$), denoted by $(W_i^{(r,q)})_{1\leq r\leq R_i, 1\leq q\leq Q_i}$. The desired band structure of the sub-matrix $W_i^{(r,q)}$ is ensured by imposing that it belongs to the following vector space:

$$\mathcal{E}_i = \{(V_{u,v})_{1\leq u\leq N_i', 1\leq v\leq M_i'} \in \mathbb{R}^{N_i'\times M_i'} \mid \forall(u,v) \text{ s.t. } |(M_i'-1)(u-1)-(N_i'-1)(v-1)| \geq d_i, V_{u,v} = 0\}.$$

Herein, d_i is an integer and, if $N_i' = M_i' > 1$, $2\lfloor d_i/(N_i'-1)\rfloor - 1$ plays a role similar to a kernel length in standard CNNs.

Finally, to control the robustness, we need to limit the Lipschitz constant of the network to a given value $\overline{\vartheta} > 0$. The related constraint can be expressed as:

$$\mathcal{C}_{i,t} = \{W_i \in \mathbb{R}^{N_i\times N_{i-1}} \mid \|A_{i,t}W_iB_{i,t}\|_{\mathrm{S}} \leq \overline{\vartheta}\} \tag{5}$$

$$A_{i,t} = W_{m,t}\cdots W_{i+1,t}, \tag{6}$$

$$B_{i,t} = W_{i-1,t+1}\cdots W_{1,t+1} \tag{7}$$

with the convention that for the first layer ($i = 1$), $B_{i,t}$ is the $N_i \times N_i$ identity matrix Id_{N_i} and for the last layer ($i = m$), $A_{i,m} = \mathrm{Id}_{N_i}$. Hereinabove, $(W_{j,t})_{1\leq j\leq m}$ designates the estimates of the weight matrices at iteration t of the projected ADAM optimizer.

The projection onto the intersection of the above three closed convex sets has no closed-form expression. The intersection $\mathcal{D}_i \cap \mathcal{E}_i^{R_i\times Q_i}$ is however quite simple to handle since the projection onto this set reduces to $\mathsf{P}_{\mathcal{D}_i} \circ \mathsf{P}_{\mathcal{E}_i^{R_i\times Q_i}}$. To compute the final projection onto $\mathcal{C}_{i,t} \cap (\mathcal{D}_i \cap \mathcal{E}_i^{R_i\times Q_i})$ of a weight matrix W_i, we use the dual forward-backward algorithm, as presented in [27], [28].

IV. Experimental Evaluation

The proposed network has been evaluated for denoising music signals.

A. Dataset Description

We train our proposed ACNN on a dataset consisting of musical exercises and songs performed on a Ronald organ. The organ covers 5 octaves (range `C2--C7`), each octave having 12 semitones, generating a total of 61 different possible notes. For the recordings, the whole range of notes is used. The songs are recorded in MIDI format using MidiEditor, in the following manner: the recording mode from MidiEditor is activated before each song being played and is stopped after the song is finished (so there is silence at the beginning and end of each recording). In total, the dataset contains 100 MIDI recordings, with a sampling frequency $F_s = 44100$ Hz, constituting 1 h and 17 min of audio. The data set is available online[§] .

The dataset is divided into training, validation, and test sets. The training set contains 90 clips with variable length, ranging from 6 s up to 150 s, having in total over an hour (67 min) of audio recordings. Some songs are repeated on different octaves to obtain a minimum number of occurrences for all notes. The validation dataset contains 9 songs with length between 12 and 120 s, forming around 10 minutes of audio signals. The test set has one 2-minute long clip.

[§]https://speed.pub.ro/downloads/

					PSNR	**MSE**	**CC**
Noisy Signal					18.25	1.18×10^{-2}	0.76
Denoised Signal	Baseline - Wavelet-based denoiser				20.66	$1.00 \times 10-3$	0.80
	ACNN denoiser	Scenario (i)	$\overline{\vartheta} = 1$		24.27	3.73×10^{-3}	0.96
			$\overline{\vartheta} = 5$		29.03	1.25×10^{-3}	0.97
			$\overline{\vartheta} = 10$		33.76	6.53×10^{-4}	0.98
		Scenario (ii)	$\overline{\vartheta} = 1$		25.87	3.12×10^{-3}	0.96
			$\overline{\vartheta} = 5$		30.63	8.63×10^{-4}	0.98
			$\overline{\vartheta} = 10$		36.02	2.23×10^{-4}	0.99
	Standard FCN denoiser		$\overline{\vartheta} = 1$		23.38	4.59×10^{-3}	0.90

TABLE I
COMPARISON OF DIFFERENT VARIANTS OF THE PROPOSED METHOD WITH BASELINES.

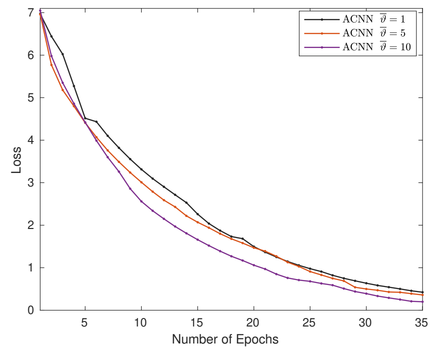

Fig. 2. Convergence profile of the proposed method.

B. Experimental setup

The noisy data for training, validating, and testing is generated by adding white Gaussian noise to the original samples. The noise has zero mean and its standard deviation is randomly chosen so that the resulting signal-to-noise ratio (SNR) varies between 5 and 30dB. The dataset samples are normalized between 0 and 1. We extract the frequency features from the audio signal using a STFT. The network estimates the STFT coefficients of the samples and an ISTFT is performed as the post-processing step. We consider a `Hanning` sliding analysis window of length $T = 23$ ms, with an overlap between two consecutive windows of 50%. The STFT is performed on 1024 points. In total, from each audio segment, a vector of length $L = 513$ frequency coefficients is obtained, constituting the input of our ACNN.

The denoising is performed using a 6-layer ACNN architecture, as presented in Figure 1. The network has an encoder-decoder structure. Each layer employs *ReLU* as activation function followed by a *batch normalization* step that acts as a regularizer and prevents the model from overfitting.

C. Simulations and results

In order to measure the performance of our proposed ACNN architecture, we perform two sets of experiments. In the first set, we control the Lipschitz constant of the architecture for three values $\overline{\vartheta}$ equal to 1, 5, and 10. In the second experiment, we test our architecture by varying the number of channels, i.e. the way we split each weight matrix. Note that for both scenarios we consider a network with $m = 6$ layers and $\forall i \in \{1, \ldots, 6\}$, $d_i = d'_i(\max\{N'_i, M'_i\} - 1)$, having the following characteristics:

(i) $R_1 = 2 = Q_6$, $Q_1 = 1 = R_6$, $\forall i \in \{2, \ldots, 5\}$ $R_i = Q_i = 2$, $(N_i)_{1 \leq i \leq 6} = (400, 200, 100, 200, 400, 513)$, $(d'_i)_{1 \leq i \leq 6} = (237, 80, 30, 30, 80, 237)$;

(ii) $R_1 = 3 = Q_6$, $Q_1 = 1 = R_6$, $\forall i \in \{2, \ldots, 5\}$ $R_i = 3$, $Q_i = 3$, $(N_i)_{1 \leq i \leq 6} = (630, 510, 420, 510, 630, 513)$, $(d'_i)_{1 \leq i \leq 6} = (237, 85, 65, 65, 85, 237)$.

We evaluate the performance on 3 standard metrics: *Peak to Signal Noise Ratio (PSNR)*, *Mean squared error (MSE)*, and *Cross-correlation (CC)*, as shown in the Table I. We compare our method with a standard denoising technique, based on a *Wavelet* decomposition. We employ a 5-level decomposition using $Symlet8$ filters combined with *SureShrink* thresholding. We also compare ACNN with a FCN implementation, for which we ensured the Lipschitz bound $\overline{\vartheta} = 1$. Here, the FCN has no structural constraints but, in addition to the imposed Lipschitz property, it has positive weights.

Table I reports the quantitative results obtained by all three approaches, evaluated over the test set. Our method outperforms the baseline on all measures by a significant margin. ACNN also outperforms classical FCN, inferring that structure imposed on the weight matrix for ACNN leads to a model with better generalization power, whereas FCN implementation may be prone to overfitting. We observed that, without any Lipschitz constraint, FCNs tend to have a very high Lipschitz constant of the order $10^5 - 10^6$, which emphasizes the importance of controlling the Lipschitz behaviour of the system. The Lipschitz constant of the network is also closely related to the expressiveness of the trained model. Architectures with tight robustness constraint have fewer degrees of freedom and are thus expected to be less accurate, and lead to a slower convergence. The convergence profile of our proposed method is shown in Figure 2.

V. CONCLUSION

This paper is a step towards bridging the gap between FCNs and CNNs. We split the weight matrix at each layer of a FCN in such a way that it acts similarly to multichannel convolutive filters in a CNN. In the training process, we constrained the weights of the network to be non-negative while letting each filter to have adaptive coefficients. We also ensured the robustness of the network against input perturbations by controlling accurately its global Lipschitz constant. We verified the effectiveness of the proposed architecture by showing its denoising capabilities on music clips. It is worth noting that the method is not only limited to such a regression problem.

References

[1] L. Huang, X. He, L. Fang, H. Rabbani, and X. Chen, "Automatic classification of retinal optical coherence tomography images with layer guided convolutional neural network," *IEEE Signal Process. Lett.*, vol. 26, no. 7, pp. 1026–1030, 2019.

[2] Y. Chen, K. Zhu, L. Zhu, X. He, P. Ghamisi, and J. A. Benediktsson, "Automatic design of convolutional neural network for hyperspectral image classification," *IEEE Geosci. Rem. Sens. Lett.*, vol. 57, no. 9, pp. 7048–7066, 2019.

[3] S. Saparudin, R. F. Malik, E. Erwin, M. Fachrurrozi, S. Sukemi, P. Rachmawati, A. Susanto, and B. Suprihatin, "Convolutional neural networks for realtime multi-faces verification with occlusion," in *Proc. Int. Conf. Electr. Eng. Comp. Sci.*, Batam Island, Indonesia, 2–3 Oct 2019, pp. 235–240.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 8-16 Oct. 2016, pp. 850–865.

[5] N. Widiastuti, "Convolution neural network for text mining and natural language processing," in *IOP Conf.: Mater. Sci. Eng.*, vol. 662, no. 5, 2019, p. 052010.

[6] S. R. Park and J. Lee, "A fully convolutional neural network for speech enhancement," *Proc. Interspeech*, pp. 1993–1997, 20–24 Aug 2017.

[7] Z. Ouyang, H. Yu, W.-P. Zhu, and B. Champagne, "A fully convolutional neural network for complex spectrogram processing in speech enhancement," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, 12–17 May 2019, pp. 5756–5760.

[8] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. Int. Conf. Learning Representations*, Toulon, France, 24–26 Apr 2017.

[9] M. Takeru, K. Toshiki, K. Masanori, and Y. Yuichi, "Spectral normalization for generative adversarial networks," in *Int. Conf. Learning Representations*, Vancouver, Canada, 30 Apr–3 May 2018.

[10] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *USENIX Security Symposium*, Austin, TX, USA, 10–12 Aug 2016, pp. 513–530.

[11] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 1–7.

[12] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," in *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, Montreal, Canada, 2–8 Dec 2018, pp. 3839–3848.

[13] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz function approximation," in *Proc. Int. Conf. Mach. Learn.*, Long Beach, California, USA, 9–15 Jun 2019, pp. 291–301.

[14] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 06–11 Aug 2017, pp. 854–863.

[15] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," in *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, 13–14 Dec 2019, pp. 11 423–11 434.

[16] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," in *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, Montreal, Canada, 2–8 Dec 2018, pp. 10 877–10 887.

[17] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *arXiv preprint arXiv:1903.01287*, 2019.

[18] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *Proc. Int. Conf. Machine Learning*, vol. 1, no. 2, pp. 5283–5292, 6–11 Aug 2017.

[19] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," in *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, Montreal, Canada, 2–8 Dec 2018, pp. 6541–6550.

[20] P. L. Combettes and J.-C. Pesquet, "Lipschitz certificates for layered network structures driven by averaged activation operators," *to appear in SIAM J. Math. Data Sc., arXiv:1903.01014*, 2019.

[21] J. Chorowski and J. M. Zurada, "Learning understandable neural networks with nonnegative weight constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 62–69, 2015.

[22] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, "Do deep convolutional nets really need to be deep and convolutional?" *Proc. Int. Conf. Learning Representations*, 24 –26 April 2017.

[23] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *Proc. Int. Conf. Learning Representations*, 24–26 Apr 2017.

[24] Z. Lin, R. Memisevic, and K. Konda, "How far can we go without convolution: Improving fully-connected networks," *Proc. Int. Conf. Learning Representations*, 2–4 May 2016.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Int. Conf. Learning Representations*, Banff, AB, Canada, 14–16 Apr 2014.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. Int. Conf. Learning Representations*, 7–9 May 2015.

[27] F. Abboud, E. Chouzenoux, J.-C. Pesquet, J.-H. Chenot, and L. Laborelli, "Dual block-coordinate forward-backward algorithm with application to deconvolution and deinterlacing of video sequences," *J. Math. Imaging Vision*, vol. 59, no. 3, pp. 415–431, 2017.

[28] A. Neacsu, J.-C. Pesquet, and C. Burileanu, "Accuracy-robustness trade-off for positively weighted neural networks," *Proc. Int. Conf. Acoust. Speech Signal Process.*, 4–7 May 2020.