A VARIATIONAL INEQUALITY MODEL FOR LEARNING NEURAL NETWORKS

Patrick L. Combettes[1] *Jean-Christophe Pesquet*[2] *Audrey Repetti*[3]

[1] *Department of Mathematics, North Carolina State University*, Raleigh, USA

[2] *Centre de Vision Numérique, CentraleSupélec, Inria, Unversité Paris-Saclay*, Gif sur Yvette, France

[3] *Maxwell Inst. for Mathematical Sciences, Inst. of Sensors, Signals and Systems, Heriot-Watt University*, Edinburgh, UK

ABSTRACT

Neural networks have become ubiquitous tools for solving signal and image processing problems, and they often outperform standard approaches. Nevertheless, training the layers of a neural network is a challenging task in many applications. The prevalent training procedure consists of minimizing highly non-convex objectives based on data sets of huge dimension. In this context, current methodologies are not guaranteed to produce global solutions. We present an alternative approach which foregoes the optimization framework and adopts a variational inequality formalism. The associated algorithm guarantees convergence of the iterates to a true solution of the variational inequality and it possesses an efficient block-iterative structure. A numerical application is presented.

Index Terms— Block-iterative algorithm, MRI, neural networks, transfer learning, variational inequality.

1. INTRODUCTION

Deep learning techniques have become very successful in solving a great variety of tasks in data science; see for instance [1, 2, 16, 17, 19, 25, 26]. Deep neural networks rely on highly parametrized nonlinear systems. Standard methods for learning the vector of parameters $\boldsymbol{\theta}$ of a neural network $T_{\boldsymbol{\theta}}$ are mainly based on stochastic algorithms such as the stochastic gradient descent (SGD) or Adam methods [18,26], and they are implemented in toolboxes such as PyTorch or TensorFlow. In this context, the standard approach to learn the parameter vector $\boldsymbol{\theta}$ is to minimize a training loss. Specifically, given a finite training data set consisting of ground truth input/output pairs $(x_k, y_k)_{1\leqslant k\leqslant K}$, a discrepancy measure is computed between the ground truth outputs $(y_k)_{1\leqslant k\leqslant K}$ and the outputs $(T_{\boldsymbol{\theta}}x_k)_{1\leqslant k\leqslant K}$ of the neural network driven by inputs $(x_k)_{1\leqslant k\leqslant K}$. Thus, if we denote by $\boldsymbol{\Theta}$ the parameter space, the objective of these methods is to

$$\underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\text{minimize}} \;\; \sum_{k=1}^{K} \ell\big(T_{\boldsymbol{\theta}}x_k, y_k\big). \tag{1}$$

One of the main weaknesses of such an approach is that it typically leads to a nonconvex optimization problem, for which existing algorithms seldom offer strong guarantees of optimality for the delivered output parameters. In other words, the solution methods do not guarantee true solutions but only local ones that may be hard to interpret in terms of the original objectives in (1).

The contribution of this work is to introduce an alternative training approach which is not based on an optimization approach but, rather, seeks the parameter vector $\boldsymbol{\theta}$ as the solution of equilibrium problems defined by variational inequalities (see [11] for background). Nonlinear analysis tools for neural network modeling have been employed in [3,7,9,10,15,20,21,23,24]. Here, we show that training a layer of a feedforward neural network can be modeled as a variational inequality problem and solved efficiently via iterative techniques such as the deterministic block-iterative forward-backward algorithm of [8]. This algorithm displays two attractive features. First, it guarantees convergence of the iterates to a true equilibrium, and not to a local solution as in the minimization setting. Second, it lends itself to a batch implementation, which is indispensable to deal with large data sets. The strategy of foregoing standard optimization in favor of more general forms of equilibria in the form of variational inequalities was first adopted in [13] in a quite different context, namely signal recovery in the presence of nonlinear observations.

The paper is organized as follows. Section 2 describes our new training method, the design of a minibatch algorithm to solve the associated variational inequality, and convergence properties of this algorithm. In Section 3, we apply the proposed approach to a transfer learning problem in which the last layer of neural network is optimized to denoise magnetic resonance (MR) images. Some conclusions are drawn in Section 4.

The work of P. L. Combettes was supported by the National Science Foundation under grant CCF-2211123. The work of J.-C. Pesquet was supported by the ANR Chair in AI BRIDGEABLE. The work of A. Repetti was supported by an RSE Personal Research Fellowship from the Royal Society of Edinburgh, by the CVN, INRIA/OPIS and CentraleSupélec.

2. PROPOSED VARIATIONAL INEQUALITY MODEL

2.1. Variational inequality model for a single layer

We first consider a single layer, modeled by an operator T_θ acting between an input Euclidean space $\mathcal{H}$ and output Euclidean space $\mathcal{G}$, and parametrized by a vector θ which is constrained to lie in a closed convex subset C of a Euclidean space Θ. More specifically,

$$T_\theta \colon \mathcal{H} \to \mathcal{G} \colon x \mapsto R(Wx + b), \tag{2}$$

where $W \colon \mathcal{H} \to \mathcal{G}$ is a linear weight operator, $b \in \mathcal{G}$ a bias vector, and $R \colon \mathcal{G} \to \mathcal{G}$ a known activation operator. The objective is to learn W and b from a training data set $(x_k, y_k)_{1 \leqslant k \leqslant K} \in (\mathcal{H} \times \mathcal{G})^K$. Our model assumes that the parametrization $\theta \mapsto (W, b)$ is linear. Further, we set

$$(\forall k \in \{1, \dots, K\}) \quad L_k \colon \Theta \to \mathcal{G} \colon \theta \mapsto W x_k + b. \tag{3}$$

Thus, the ideal problem is to

$$\text{find } \theta \in C \text{ such that } (\forall k \in \{1, \dots, K\}) \ T_\theta x_k = y_k, \tag{4}$$

that is, to

$$\text{find } \theta \in C \text{ such that } (\forall k \in \{1, \dots, K\}) \ R(L_k \theta) = y_k. \tag{5}$$

In practice, this ideal formulation has no solution and one must introduce a meaningful relaxation of it. This is usually done via optimization formulations such as (1), which leads to the pitfalls discussed in Section 1.

The approach we propose to construct a relaxation of (5) starts with the observation made in [9] that most activation operators are firmly nonexpansive in the sense that, for every $(z_1, z_2) \in \mathcal{G}^2$, $\langle z_1 - z_2 \mid Rz_1 - Rz_2 \rangle \geqslant \|Rz_1 - Rz_2\|^2$. As shown in [12], (5) can be relaxed into the variational inequality problem

$$\text{find } \theta \in C \text{ such that } (\forall \vartheta \in C) \left\langle \vartheta - \theta \,\middle|\, \sum_{k=1}^{K} \omega_k L_k^*\big(R(L_k\theta) - y_k\big) \right\rangle \geqslant 0 \tag{6}$$

where, for every $k \in \{1, \dots, K\}$, $L_k^* \colon \mathcal{G} \to \Theta$ is the adjoint of L_k and $\omega_k \in \left]0, 1\right[$, and $\sum_{l=1}^{K} \omega_l = 1$. This relaxation is exact in the sense that, if (5) has solutions, they are the same as those of (6). We assume that (6) has solutions, which is true under mild conditions [12].

2.2. Block-iterative forward-backward splitting

We solve the variational inequality problem (6) by adapting a block-iterative forward-backward algorithm proposed in [8]. This algorithm splits the computations associated with the different linear operators $(L_k)_{1 \leqslant k \leqslant K}$ using a block-iterative approach. At iteration $n \in \mathbb{N}$, a subset $\mathbb{K}_n$ of $\{1, \dots, K\}$ is selected and, for every $k \in \mathbb{K}_n$, a forward step in the direction of the vector $L_k^*(R\big(L_k\theta_n\big) - y_k)$ is performed. The forward steps are then averaged and projected onto the constraint set C.

Algorithm 1 Take $\gamma \in \left]0, 2/\max_{1 \leqslant k \leqslant K} \|L_k\|^2\right[$, $\theta_0 \in \Theta$, and $(\vartheta_{k,0})_{1 \leqslant k \leqslant K} \in \Theta^K$. Iterate

$$\begin{array}{l} \text{for } n = 0, 1, \dots \\ \quad \text{select } \varnothing \neq \mathbb{K}_n \subset \{1, \dots, K\} \\ \quad \text{for every } k \in \mathbb{K}_n \\ \quad\quad \vartheta_{k,n+1} = \theta_n - \gamma L_k^*\big(R(L_k\theta_n) - y_k\big) \\ \quad \text{for every } k \in \{1, \dots, K\} \smallsetminus \mathbb{K}_n \\ \quad\quad \vartheta_{k,n+1} = \vartheta_{k,n} \\ \quad \theta_{n+1} = \operatorname{proj}_C \left(\displaystyle\sum_{k=1}^{K} \omega_k \vartheta_{k,n+1} \right). \end{array} \tag{7}$$

We then derive the following result from [8].

Proposition 2 *Suppose that, for some $P \in \mathbb{N}$, every index $k \in \{1, \dots, K\}$ is selected at least once within any P consecutive iterations, i.e., $(\forall n \in \mathbb{N}) \bigcup_{k=0}^{P-1} \mathbb{K}_{n+k} = \{1, \dots, K\}$. Then the sequence $(\theta_n)_{n \in \mathbb{N}}$ generated by Algorithm 1 converges to a solution to* (6).

2.3. Proposed deterministic batch algorithm

In a neural network context, batch approaches are necessary for training purposes. Towards this goal, we modify Algorithm 1 into a batch-based deterministic forward-backward scheme to solve (6).

Let us form a partition $(\mathbb{K}_j)_{1 \leqslant j \leqslant J}$ of $\{1, \dots, K\}$ and assume that, at each iteration $n \in \mathbb{N}$, only one batch index $j_n \in \{1, \dots, J\}$ is selected. Then, to avoid keeping in memory all values of $(\vartheta_{k,n})_{1 \leqslant k \leqslant K}$, Algorithm 1 is rewritten below (Algorithm 3) so that only J variables are kept in memory. Given $j_n \in \{1, \dots, J\}$, $\overline{\theta}_{j_n,n} \in \Theta$ denotes the stored variable associated with subset $\mathbb{K}_{j_n}$.

Algorithm 3 Take $\gamma \in \left]0, 2/\max_{1 \leqslant k \leqslant K} \|L_k\|^2\right[$, $\theta_0 \in \Theta$, and $(\overline{\theta}_{j,0})_{1 \leqslant j \leqslant J} \in \Theta^J$. Iterate

$$\begin{array}{l} \text{for } n = 0, 1, \dots \\ \quad \text{select } j_n \in \{1, \dots, J\} \\ \quad \overline{\theta}_{j_n,n+1} = \displaystyle\sum_{k \in \mathbb{K}_{j_n}} \omega_k \Big(\theta_n - \gamma L_k^*\big(R(L_k\theta_n) - y_k\big)\Big) \\ \quad \text{for } j \in \{1, \dots, J\} \smallsetminus \{j_n\} \\ \quad\quad \overline{\theta}_{j,n+1} = \overline{\theta}_{j,n} \\ \quad \overline{\theta}_{n+1} = \overline{\theta}_n - \overline{\theta}_{j_n,n+1} + \overline{\theta}_{j_n,n} \\ \quad \theta_{n+1} = \operatorname{proj}_C \overline{\theta}_{n+1}. \end{array} \tag{8}$$

The sequence $(\theta_n)_{n\in\mathbb{N}}$ in (3) converges to a solution to (6) under the mild condition that there exist $P \in \mathbb{N}$ such that $(\forall n \in \mathbb{N}) \bigcup_{k=0}^{P-1}\{j_{n+k}\} = \{1,\dots,J\}$ [12].

2.4. The case of general feedforward neural networks

Let $\mathcal{H}_0,\dots,\mathcal{H}_M$ be Euclidean spaces. A feedforward neural network $T_{\boldsymbol{\theta}}\colon \mathcal{H}_0 \to \mathcal{H}_m$ consists of a composition of M layers

$$T_{\boldsymbol{\theta}} = T_{M,\theta_M} \circ \cdots \circ T_{1,\theta_1} \tag{9}$$

where the operators $(T_{m,\theta_m})_{1\leqslant m\leqslant M}$ are as in Section 2.1: $\theta_m \in \Theta_m$ is a vector linearly parametrizing a weight operator $W_m\colon \mathcal{H}_{m-1} \to \mathcal{H}_m$ and a bias vector $b_m \in \mathcal{H}_m$, and $R_m\colon \mathcal{H}_m \to \mathcal{H}_m$ is a firmly nonexpansive activation operator. For convenience, we gather the learnable parameters of the network in a vector $\boldsymbol{\theta} = (\theta_m)_{1\leqslant m\leqslant M}$. Given a training sequence $(x_k, y_k)_{1\leqslant k\leqslant K} \in (\mathcal{H}_0 \times \mathcal{H}_M)^K$, the approach proposed in Section 2 is used to train the last layer of the neural network. For this layer the input sequence is defined by $(\forall k \in \{1,\dots,K\})\ \widetilde{x}_k = T_{M-1,\theta_{M-1}} \circ \cdots \circ T_{1,\theta_1} x_k$. Two learning scenarios are of special interest:

- Greedy training [5]. Layers are added one after the other to form a deep neural network.
- Transfer learning [4, 6, 14, 22]. The goal is to retrain the last layer of a trained neural network to allow it to be applied to a different data set or a different task.

3. TRANSFER LEARNING: FINE-TUNING LAST LAYER OF A DENOISING NEURAL NETWORK

We apply the proposed variational inequality model to a transfer learning problem for building a denoising neural network. Transfer learning [4, 6, 14, 22] is often used in practice to tailor a neural network that has been trained on a particular data set, for a different type of data and improve its performance [22].

3.1. General setting for denoising neural networks

We consider a denoising neural network $T_{\boldsymbol{\theta}}\colon \mathcal{H} \to \mathcal{H}$ with M layers, defined as in (9). $T_{\boldsymbol{\theta}^*}$ has been pretrained as a denoiser, such that

$$\boldsymbol{\theta}^* \in \operatorname*{Argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{k=1}^{K'} \ell\big(T_{\boldsymbol{\theta}} u_k, v_k\big), \tag{10}$$

where each $(u_k, v_k) \in \mathcal{H}^2$ is a pair of noisy/ground truth images, and $\ell\colon \mathcal{H}\times\mathcal{H} \to \mathbb{R}$ a loss function. The objective is to retrain only the last layer of $T_{\boldsymbol{\theta}^*}$ in order to use it on a different type of images. For instance, if the network has been trained on natural images, it can be fine-tuned to denoise medical images obtained by modalities such as MR or computed tomography.

3.2. Simulation setting

In our experiments, $T_{\boldsymbol{\theta}^*}$ is a DnCNN with $M = 20$ layers, of the form of (9), where $\mathcal{H}_0 = \mathbb{R}^{N\times N}$, $\mathcal{H}_1 = \cdots = \mathcal{H}_{19} = \mathbb{R}^{64\times N\times N}$, and $\mathcal{H}_{20} = \mathbb{R}^{N\times N}$. The layers of the networks are as follows. For the first layer, W_1 represents a convolutional layer with 1 input, 64 outputs, and a kernel of size 3×3. For every $m \in \{2,\dots,19\}$, W_m represents a multi-input multi-output convolutional layer with 64 inputs, 64 outputs, and a kernel of size 3×3. Finally, W_{20} represents a convolutional layer with 64 inputs, 1 output and a kernel of size 3×3. We use LeakyReLU activation functions with negative slope parameter 10^{-2}. As shown in [9], this operator is firmly nonexpansive. In addition, we take $b_1 = \cdots = b_{20} = 0$.

The network $T_{\boldsymbol{\theta}^*}$ is trained on the $50,000$ ImageNet test data set converted to grayscale images and normalized between 0 and 1. The ground truth images $(v_k)_{1\leqslant k\leqslant K'}$ correspond to patches of size 50×50 selected randomly during training. For every $k \in \{1,\dots,K'\}$, the degraded images are obtained as $u_k = v_k + \sigma b_k$, where $\sigma = 0.02$ and $b_k \in \mathbb{R}^{50\times 50}$ is a realization of a random standard white Gaussian noise.

We propose to fine-tune the network $T_{\boldsymbol{\theta}^*}$ to denoise MR images. We thus focus on the training of the last layer $T_{20,\theta_{20}}$. We choose W_{20} to be a convolutional layer with 64 inputs, 1 output, and kernels w of size 7×7. In addition, R_{20} is a LeakyReLU activation function with negative slope parameter 10^{-3}. In this context, (3) assumes the form

$$L_k\colon \mathbb{R}^{64\times 1\times 7\times 7} \to \mathbb{R}^{N\times N}\colon w \mapsto \widetilde{x}_k * w, \tag{11}$$

where $\widetilde{x}_k \in \mathbb{R}^{64\times N\times N}$ is the output of the 19th layer.

Three training strategies for $T_{20,\theta_{20}}$ are considered: the standard SGD and the Adam algorithm for minimizing an ℓ^1 loss, as well as the approach proposed in Section 2 with $C = \Theta_{20} = \mathbb{R}^{64\times 1\times 7\times 7}$.

For the three methods, the training set consists of the first 300 images of the fastMRI train data set, and we test the resulting networks on the next 300 images of the same data set. The dimension of the images in this data set is 320×320. We train the networks on patches, by splitting the images into 16 patches of size 80×80. The patches are randomly shuffled every time the algorithm has seen all the patches of the data set. Moreover, we split the training set into batches containing 10 patches located at the same position in 10 images of the train set. Batches are normalized between 0 and 1, and corrupted with an additive white Gaussian noise with stan-

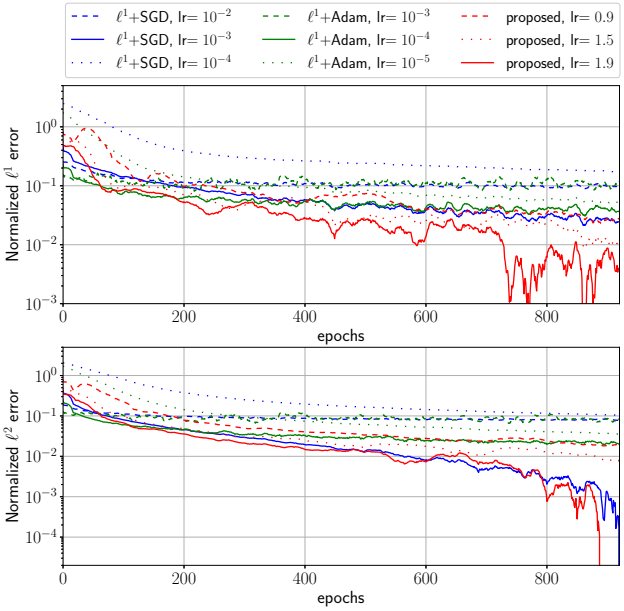

Fig. 1: Convergence profiles showing the normalized averaged ℓ^1 (top) and ℓ^2 (bottom) errors (in log scales) with respect to epochs, for the ℓ^1+SGD method (blue), the ℓ^1+Adam method (green), and the proposed approach (red). Continuous lines show best step-size (i.e., learning rate) for each method. Dashed and dotted lines show inaccurate choice of step-size.

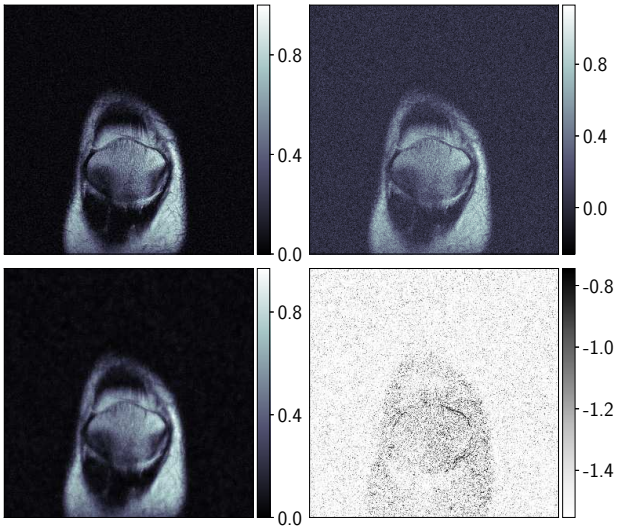

Fig. 2: Denoising results on the test set corresponding to slice 399 of the fastMRI training set. First row: ground truth (left) and corresponding noisy observation (right) with PSNR $= 23.09$ dB. Second row: output of the DnCNN trained with the proposed procedure (left), with PNSR $= 29.31$ dB, and the corresponding error map, in log-scale (right).

Table 1: Average SSIM (and standard deviation) obtained for the first 300 images of the fastMRI training set, and the next 300 images of the same set.

	SSIM	
method	Training set	Test set
proposed	**0.6647 (±0.0721)**	**0.6630 (±0.0597)**
ℓ^1+SGD	0.6641 (±0.0770)	0.6627 (±0.0629)
ℓ^1+Adam	0.6598 (±0.0703)	0.6239 (±0.0346)

dard deviation 0.07. One epoch is completed when the algorithm has seen all the batches at the same location (i.e., 30 batches generated as explained above). All algorithms are run over 1000 epochs.

The learning rate in SGD and Adam has been tuned manually to achieve best performance. In Algorithm 3, each ω_k is set to $1/K$ and $\gamma = 1.9/\max_{1\leqslant k\leqslant K}\|L_k\|^2$, where each $\|L_k\|$ is computed via power iteration.

3.3. Simulation results

Since the proposed method is not devised as a minimization method, we assess the behavior of the three learning procedure during training by monitoring the ℓ^p errors $\sum_{k=1}^K \|T_{20,\theta_{20,e}}\widetilde{x}_k - y_k\|_p^p$ for $p \in \{1,2\}$ with respect to the epochs $e \in \{1,\ldots,1000\}$. We observe that, for any choice of the step-size value γ (even not determined optimally), our method reaches a lower ℓ^1 error more quickly than SGD and Adam, for any choice of the learning rate. For the ℓ^2 error, any choice of step-size will lead to faster convergence than Adam. For this example, an accurate choice of learning rate for SGD leads to a performance which is similar to that of the proposed approach. However, choosing an inaccurate learning rate results in extremely slow convergence (to a local solution) or diverging behavior for SGD, while our method converges to a true solution of (6) for any choice of step-size as long as it satisfies the conditions given in Algorithm 3.

The SSIM values for the 300 training images and the 300 test images are shown in Table 1. We observe that our approach yields slightly better results for both data sets. One image slice of the test data set is displayed in Fig. 2 to show the good visual quality of the proposed transfer learning approach.

4. CONCLUSION

A new framework has been proposed to train neural network layers based on a variational inequality model. The effectiveness of this approach has been illustrated through simulations on a transfer learning problem. In future work, we plan to explore further algorithmic developments and consider various applications of the proposed technique to other training problems.

5. REFERENCES

[1] J. Adler and O. Öktem, "Learned primal-dual reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, pp. 1322–1332, 2018.

[2] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery," *IEEE Signal Process. Mag.*, vol. 37, pp. 105–116, 2020.

[3] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," *Proc. Conf. Adv. Neural Inform. Process. Syst.*, vol. 32, pp. 690–701, 2019.

[4] B. Banerjee and P. Stone, "General game learning using knowledge transfer," in: *Proc. 20th Int. Joint Conf. Artificial Intell.*, pp. 672–677, 2007.

[5] E. Belilovki, M. Eickenberg, and E. Oyallon, "Decoupled greedy learning of CNNs," in: *Proc. 37th Int. Conf. Machine Learning*, vol. 119, pp. 736–745, 2020.

[6] J. J. Bird, J. Kobylarz, D. R. Faria, A. Ekárt, and E. P. Ribeiro, "Cross-domain MLP and CNN transfer learning for biological signal processing: EEG and EMG," *IEEE Access*, vol. 8, pp. 54789–54801, 2020.

[7] R. Cohen, M. Elad, and P. Milanfar, "Regularization by denoising via fixed-point projection (RED-PRO)," *SIAM J. Imaging Sci.*, vol. 14, pp. 1374–1406, 2021.

[8] P. L. Combettes and L. E. Glaudin, "Solving composite fixed point problems with block updates," *Adv. Nonlinear Anal.*, vol. 10, pp. 1154–1177, 2021.

[9] P. L. Combettes and J.-C. Pesquet, "Deep neural network structures solving variational inequalities," *Set-Valued Var. Anal.*, vol. 28, pp. 491–518, 2020.

[10] P. L. Combettes and J.-C. Pesquet, "Lipschitz certificates for layered network structures driven by averaged activation operators," *SIAM J. Math. Data Sci.*, vol. 2, pp. 529–557, 2020.

[11] P. L. Combettes and J.-C. Pesquet, "Fixed point strategies in data science," *IEEE Trans. Signal Process.*, vol. 69, pp. 3878–3905, 2021.

[12] P. L. Combettes, J.-C. Pesquet, and A. Repetti, "A variational inequality model for learning neural networks," extended version.

[13] P. L. Combettes and Z. C. Woodstock, "A variational inequality model for the construction of signals from inconsistent nonlinear equations," *SIAM J. Imaging Sci.*, vol. 15, pp. 84–109, 2022.

[14] C. B. Do and A. Y. Ng, "Transfer learning for text classification," in: *Proc. Conf. Adv. Neural Inform. Process. Syst.*, vol. 18, pp. 299–306, 2015.

[15] M. Hasannasab, J. Hertrich, S. Neumayer, G. Plonka, S. Setzer, and G. Steidl, "Parseval proximal neural networks," *J. Fourier Anal. Appl.*, vol. 26, art. 59, 2020.

[16] S. Haykin, *Neural Networks: A Comprehensive Foundation,* 2nd ed. Pearson Education, Singapore, 1998.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proc. Int. Conf. Comput. Vision,* pp. 1026–1034, 2015.

[18] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in: *Proc. 3rd Intern. Conf. Learn. Represent.*, 2015.

[19] Y. A. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, pp. 436–444, 2015.

[20] J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux, "Learning maximally monotone operators for image recovery," *J. Imaging Sci.*, vol. 14, pp. 1206–1237, 2021.

[21] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comput. Imaging*, vol. 5, pp. 52–67, 2019.

[22] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, pp. 1–40, 2016.

[23] E. Winston and J. Z. Kolter, "Monotone operator equilibrium networks," *Proc. Conf. Adv. Neural Inform. Process. Syst.*, vol. 33, pp. 10718–10728, 2020.

[24] X. Xu, Y. Sun, J. Liu, B. Wohlberg, and U. S. Kamilov, "Provable convergence of plug-and-play priors with MMSE denoisers," *IEEE Signal Process. Lett.*, vol. 27, pp. 1280–1284, 2020.

[25] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, pp. 6360–6376, 2021.

[26] K. Zhang, W. Zuo, and L. Zhang, "Deep plug-and-play super-resolution for arbitrary blur kernels," in: *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 1671–1681, 2019.