How to warm-start your unfolding network

Vicky Kouni[†,*]
[†] *Mathematical Institute, University of Oxford, kouni@maths.ox.ac.uk*
[*] *Isaac Newton Institute for Mathematical Studies, University of Cambridge, vk428@cam.ac.uk*

***Abstract*—We present a new ensemble framework for boosting the performance of overparameterized unfolding networks solving the compressed sensing problem. We combine a state-of-the-art overparameterized unfolding network with a continuation technique, to warm-start a crucial quantity of the said network's architecture; we coin the resulting continued network C-DEC. Moreover, for training and evaluating C-DEC, we incorporate the log-cosh loss function, which enjoys both linear and quadratic behavior. Finally, we numerically assess C-DEC's performance on real-world images. Results showcase that the combination of continuation with the overparameterized unfolded architecture, trained and evaluated with the chosen loss function, yields smoother loss landscapes and improved reconstruction and generalization performance of C-DEC, consistently for all datasets.**

***Index Terms*—deep unfolding network, compressed sensing, continuation, overparameterization, warm-start, loss landscape.**

I. Introduction

Deep unfolding networks (DUNs) [1], [2], [3] burst into the deep learning scene [4], by merging merits of model-based [5] and data-driven approaches [6]. A DUN is formed by casting the iterations of a model-based algorithm as layers of a deep neural network (DNN), solving an inverse problem [7], [8]. A particular example is the compressed sensing (CS) problem [9], [10], which pertains to reconstructing data $x \in \mathbb{R}^n$ from incomplete, noisy measurements $y = Ax + e \in \mathbb{R}^m$, $m < n$. Given y, a DUN implements a *decoder* $h : \mathbb{R}^m \mapsto \mathbb{R}^n$, so that $h(y) = \hat{x} \approx x$ [11], [12]. As such, the CS reconstruction problem resembles a regression one [13].
DUNs enjoy a set of advantages compared to standard DNNs [14], [15] for CS: they are interpretable [16] and incorporate prior knowledge of data structure [17], while they significantly reduce the time complexity and increase the reconstruction quality [18]. Still, boosting the performance of DUNs is an ongoing research direction. To that end, DUNs are infused with elements from deep-learning architectures. For instance, [19], [20], and [21] explore the combination of DUNs with LSTM-type gates units, attention mechanisms, and generative data priors, respectively. Nevertheless, the deployment of model-based tools is quite unexplored, although its relevance seems natural, by definition of DUNs.
In fact, there are optimization techniques for boosting the performance of model-based iterative algorithms, with a prime example being continuation [22], [23], [24], [25], also known as "warm-starting" [26], [27]. Coarsely speaking, continuation hinges upon making better and better guesses on the values of parameters/variables used in a CS reconstruction algorithm, in order to improve the algorithm's performance. Since continuation appeals to model-based methods, it appears intuitive to also apply it in the unfolding regime.
Our work is inspired by [28], [29], [30]. In [28], continuation is utilized to warm-start the proposed model-based CS reconstruction algorithm. Moreover, [29] explores how model-based choices, e.g., different matrices A, affect the behavior of DUNs, while [30] introduces a new loss function for training and evaluating the proposed DUN. Similarly, we will leverage model-based techniques to increase efficiency of DUNs. We differentiate our approach by exploiting continuation to warm-start a key quantity of a state-of-the-art overparameterized DUN, namely, DECONET [11], which solves the CS problem. To our knowledge, warm-starting DUNs for improving their performance has not yet been explored. We coin this new framework of continuation applied on DUN as *continued decoder* (C-DEC). In contrast to standard metrics for assessing DUNs' performance, e.g., the mean-squared loss [12], [31], [32], we train and evaluate C-DEC using the log-cosh loss function [33], to increase the reconstruction and generalization performance of the proposed DUN. Log-cosh is a well-known metric for regression problems [34] – though never used before in the context of DUNs – and combines merits of both mean-absolute error and mean-squared error (cf. Sec. III). Finally, we experiment with C-DEC on real-world images. Results indicate that continuation and deployment of log-cosh as a metric for testing the proposed DUN, improve reconstruction and generalization performance of C-DEC – and especially in the case of loss landscapes [35], where continuation and overparameterization induce a "smoothing" effect.
Our contributions read as follows: a) we propose a new ensemble framework, which comprises of warm-starting a state-of-the-art overparameterized DUN for CS, resulting in a continued DUN dubbed C-DEC, whose performance is measured in terms of the log-cosh loss; the latter has not been used before in the context of DUNs b) we provide empirical evidence confirming the beneficial role of continuation (and overparameterization) in C-DEC's reconstruction and generalization performance, when the log-cosh loss is employed. We also highlight that continuation and overparameterization smooth out C-DEC's loss landscapes; to our knowledge, we are the first to visualize the loss landscapes of DUNs.

The author would like to thank the Isaac Newton Inst. for Math. Sci. for supporting her during her INI Postdoctoral Research Fellowship in the Mathematical Sciences, especially during the programme "Representing, calibrating & leveraging prediction uncertainty from statistics to machine learning". This work was funded by the EPSRC (Grant Number EP/V521929/1).

Notation: For $x \in \mathbb{R}$, $\tau > 0$, the soft-thresholding operator $\mathcal{S}_\tau : \mathbb{R} \mapsto \mathbb{R}$ and the truncation operator $\mathcal{T}_\tau : \mathbb{R} \mapsto \mathbb{R}$ are defined in closed form as $\mathcal{S}_\tau(x) = \operatorname{sign}(x) \max(0, |x| - \tau)$ and $\mathcal{T}_\tau(x) = \operatorname{sign}(x) \min\{|x|, \tau\}$, respectively. For $x \in \mathbb{R}^n$, both $\mathcal{S}_\tau(\cdot)$ and $\mathcal{T}_\tau(\cdot)$ act component-wise. For $x \in \mathbb{R}$, the hyperbolic cosine function is given as $\cosh(x) = (e^x + e^{-x})/2$.

Algorithm 1: [28, Listing 6]

```
Input : $y \in \mathbb{R}^m$
Output: solution $\hat{x} \in \mathbb{R}^n$ of (1)
1 Initialize $x_0 \in \mathbb{R}^n$, $z_0^1 \in \mathbb{R}^N$, $z_0^2 \in \mathbb{R}^m$;
2 $\theta_0 \leftarrow 1$, $u_0^1 \leftarrow z_0^1$, $u_0^2 \leftarrow z_0^2$;
3 for $k = 0, 1, \dots$ do
4     $x_k \leftarrow x_0 + \mu^{-1}((1-\theta_k)W^T u_k^1 + \theta_k W^T z_k^1 - (1-\theta_k)A^T u_k^2 - \theta_k A^T z_k^2)$;
5     $z_{k+1}^1 \leftarrow \mathcal{T}_{\theta_k^{-1} t_k^1}((1-\theta_k)u_k^1 + \theta_k z_k^1 - \theta_k^{-1} t_k^1 W x_k)$;
6     $z_{k+1}^2 \leftarrow \mathcal{S}_{\theta_k^{-1} t_k^2 \varepsilon}((1-\theta_k)u_k^2 + \theta_k z_k^2 - \theta_k^{-1} t_k^2 (y - Ax_k))$;
7     $u_{k+1}^1 \leftarrow (1-\theta_k)u_k^1 + \theta_k z_{k+1}^1$;
8     $u_{k+1}^2 \leftarrow (1-\theta_k)u_k^2 + \theta_k z_{k+1}^2$;
9     $\theta_{k+1} \leftarrow 2/(1 + (1 + 4/(\theta_k)^2)^{1/2})$;
```

II. Main results

Model-based CS: CS pertains to reconstructing $x \in \mathbb{R}^n$ from $y = Ax + e \in \mathbb{R}^m$, $m < n$, with $A \in \mathbb{R}^{m \times n}$ and $e \in \mathbb{R}^m$, $\|e\|_2 \leq \varepsilon$. The data x is assumed to be sparse after the application of a *known* transform W, to ensure exact/approximate reconstruction; in this paper, we consider the case of a *redundant* $W \in \mathbb{R}^{N \times n}$. In optimization terms, this is translated into minimizing a smooth, sparsity-inducing objective function satisfying a data-fitting term. In other words, a common approach [28] for formulating the CS problem is:

$$\min_{x \in \mathbb{R}^n} \|Wx\|_1 + \frac{\mu}{2}\|x - x_0\|_2^2 \quad \text{s. t.} \quad \|y - Ax\|_2 \leq \varepsilon, \tag{1}$$

with $x_0 \in \mathbb{R}^n$ being an initial guess on x, and $\mu > 0$. A state-of-the-art algorithm solving (1) is Algorithm 1. The latter takes as input y, introduces dual variables $z^1 \in \mathbb{R}^N$, $z^2 \in \mathbb{R}^m$ and step sizes t^1, t^2, θ, and produces an iterative scheme, which after some iterations outputs $\hat{x} \approx x$.
Due to the appearance of the term $(\mu/2)\|x - x_0\|_2^2$ in (1), the performance of the solver depends on the choices of x_0 (and μ); to alleviate this effect, one can use continuation. The latter is an algorithmic method deployed to increase the solver's performance and hinges upon iteratively solving (1) with better and better guesses on x_0 (and μ). In other words, continuation solves a sequence of similar but easier problems, using the results of each "subproblem" to "warm start", i.e., initialize the next one. The combination of Algorithm 1 with continuation results in Algorithm 2, presented here slightly differently from the one in [28], to satisfy our paper's formulation. We focus on updating x_0, and keep μ fixed, since updating the latter is optional [28]. Of particular interest is the fact that continuation is quite efficient when coupled with Algorithm 1 and boosts its performance, in terms of reconstruction error [28].
Deep-unfolding CS: To reformulate Algorithm 1 as a DUN, we firstly consider W to be *unknown* and learned by a set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^s \overset{\text{i.i.d.}}{\sim} \mathcal{D}^s$, with $\mathcal{D}$ being an unknown distribution. Then, by treating the number of iterations k as layers, Algorithm 1 is cast as an unfolded architecture called DECONET [11], with outputs of the kth layer given by

$$f_1^W(y) = \sigma_1^W(y) \tag{2}$$
$$f_k^W(v) = D_{k-1}v + \Theta_{k-1}\sigma_{k-1}^W(v), \quad k = 2, \dots, L, \tag{3}$$

with $v \in \mathbb{R}^{(2N+2m)\times 1}$ being an auxiliary variable and L the total number of layers. Formally, y is passed through all subsequent layers, but for the sake of readability, we only write the dependence on v here. The matrices $\{D_k, \Theta_k\}_{k\geq 1} \in \mathbb{R}^{(2N+2m)\times(2N+2m)}$ and the nonlinear function $\sigma(\cdot)$ depend on the step sizes and the optimization variables of Algorithm 1 (see [11] for more implementation details). By composing the outputs layer after layer, DECONET implements a decoder $h^W : \mathbb{R}^m \mapsto \mathbb{R}^n$ for CS, such that $h^W(y) := h(y) = \hat{x} \approx x$. The learnable W is shared across all layers, and due to its redundancy, it renders DECONET as an overparameterized DUN. Since both DECONET and Algorithm 1 solve (1), we can replace the latter with the former inside the continuation loop of Algorithm 2. This tantamounts to a new unfolded architecture, presented in Algorithm 3, where we combine the continuation loop with DECONET. We coin this hybrid model-based unfolded architecture *continued decoder* (C-DEC) and write $g_{x_0}^W(y)$ for the decoder it implements, to distinguish it from the decoder implemented by DECONET.
Given a training sequence of s pair-samples, we aim to measure the difference between x_i and $\hat{x}_i^\star = g_{x_0}^W(y_i)$, $i = 1, \dots, s$. To that end, we choose the log-cosh loss function [33]:

$$\mathcal{L}_{\text{train}} = \frac{1}{s}\sum_{i=1}^s \log\cosh(g_{x_0}^W(y_i) - x_i), \tag{4}$$

as opposed to DECONET, which is trained and tested with the mean-squared error (MSE) [11]. Our choice of log-cosh [34] is attributed to its behavior as the MSE close to the origin, and as the mean-absolute error [36] far from the origin. Hence, log-cosh incorporates properties of both loss functions, which are standard metrics for DUNs' performance. Based on this aspect and the success of continuation when coupled with the iterative solver for CS, we believe that C-DEC will demonstrate an improved performance compared to its vanilla counterpart. In Sec. III, we provide empirical evidence, confirming the superiority of the proposed ensemble framework.

Algorithm 2: Alg. 1 with continuation [28, Listing 11]

```
Input : $y \in \mathbb{R}^m$
Output: Warm-started solution $\hat{x}^\star \in \mathbb{R}^n$ of (1)
1 Initialize $x_0^0 \in \mathbb{R}^n$, $\mu_0 > 0$;
2 for $j = 0, 1, \dots$ do
3     $\hat{x}^{j+1} \leftarrow$Algorithm 1$(y)$;
4     $x_0^{j+1} \leftarrow \hat{x}^j + \frac{j}{j+3}(\hat{x}^{j+1} - \hat{x}^j)$;
5     (optional) linear increase or decrease of $\mu_j$;
```

III. Experiments

Settings: We train and test C-DEC on MNIST (60000 training and 10000 test 28×28 image examples) and CIFAR10 (50000

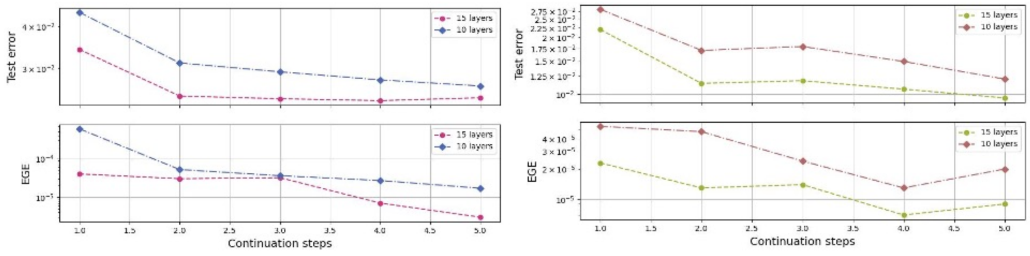


Fig. 1. Performance plots of C-DEC, with $W \in \mathbb{R}^{10n\times n}$, on MNIST (left) and CIFAR10 (right) datasets.

training and 10000 test 32×32 coloured image examples). We transform the CIFAR10 images into grayscale ones. For both datasets, we consider the vectorized form of the images. We fix $m = 0.25 \cdot n$, use a Gaussian $\tilde{A} := A/\sqrt{m} \in \mathbb{R}^{m\times n}$, and add zero-mean Gaussian noise e with standard deviation $\text{std} = 10^{-4}$ to the measurements y, so that $y = \tilde{A}x + e$. We set $\varepsilon = \|y - \tilde{A}x\|_2$ and $x_0 = A^T y$, which are standard algorithmic setups for Algorithm 1. We investigate C-DEC with varying number of layers and continuation steps. For the learnable sparsifying transform $W \in \mathbb{R}^{N\times n}$ we consider two instances, with $N = 10 \cdot n$ and $N = 50 \cdot n$, both initialized with a Beta distribution [37]. The rest of C-DEC's hyperparameters, e.g., μ, t^1, t^2, θ, are inherited by DECONET and thus set according to [11]. All models are implemented in PyTorch [38] and trained using *Adam* algorithm [39], with batch size 128 and learning rates $\eta = 10^{-3}$ for MNIST and $\eta = 10^{-4}$ for CIFAR10. For our experiments, we report the test log-cosh loss $\mathcal{L}_{\text{test}}$ – which is essentially the log-cosh loss evaluated on a set of d test data not used during training – and the *empirical generalization error* (EGE) $\mathcal{L}_{\text{gen}} = |\mathcal{L}_{\text{test}} - \mathcal{L}_{\text{train}}|$, with $\mathcal{L}_{\text{train}}$ as in (4). We train C-DEC, on all datasets, employing an early stopping technique [40] with respect to $\mathcal{L}_{\text{gen}}$. We repeat all the experiments at least 10 times and average the results over the runs. Since our main objective is to showcase how the model-based continuation ripples out to the behavior of the unfolded network, we only employ DECONET as a baseline and leave other DUNs as future work, especially since [11] has demonstrated that DECONET outperforms two other state-of-the-art DUNs. For the loss landscapes, we follow the standard framework of [35] and perturb the learned W in two random directions, with scalars $Alpha$ and $Beta$ representing the extent of the perturbation in each direction.

Results & discussion: We examine the reconstruction and generalization performance of 10- and 15-layer C-DEC, with $W \in \mathbb{R}^{10n\times n}$ and varying number of continuation steps, on both datasets, and illustrate the findings in Fig. 1. From a model-based viewpoint, the decays in test errors seem reasonable: as the number of continuation steps (and layers) increases, the solver's reconstruction ability is improved [28]. This behavior conforms with our intuition presented in Sec. II. Interestingly, we find that the number of continuation steps is mildly reversely proportional to the generalization error. We surmise that this is due to the nested architecture of C-DEC compared to DECONET, although further mathematical exploration is needed. Additionally, we compare C-DEC and DECONET, both with $W \in \mathbb{R}^{50n\times n}$, for 10 and 50 layers, on all datasets, and present the findings in Table I. Both the test and generalization errors are always lower for our proposed DUN, consistently for both datasets. Based on the results, we obtain a two-fold insight: a) we confirm our model-based intuition that continuation boosts the solver's performance, even in the unfolding regime b) we justify our choice of the log-cosh loss as an adequate metric for training and evaluating the proposed DUN, compared to MSE, which is deployed for DECONET. Overall, results highlight that continuation is a favorable component of C-DEC's architecture, with the log-cosh loss being a better choice over standard metrics, e.g., the MSE, for measuring DUNs' performance.

Finally, we examine how continuation ripples out to C-DEC's loss landscapes. We compare a 5-layer C-DEC without continuation – which is architecturally equivalent to DECONET – and a 5-layer C-DEC with 5 steps of continuation. We examine both DUNs with $W \in \mathbb{R}^{10n\times n}$ and $W \in \mathbb{R}^{50n\times n}$, and report the results in Fig. 2. The illustrations demonstrate a very intriguing phenomenon. Firstly, we observe that for both datasets, DECONET (equivalent to C-DEC with 1-step

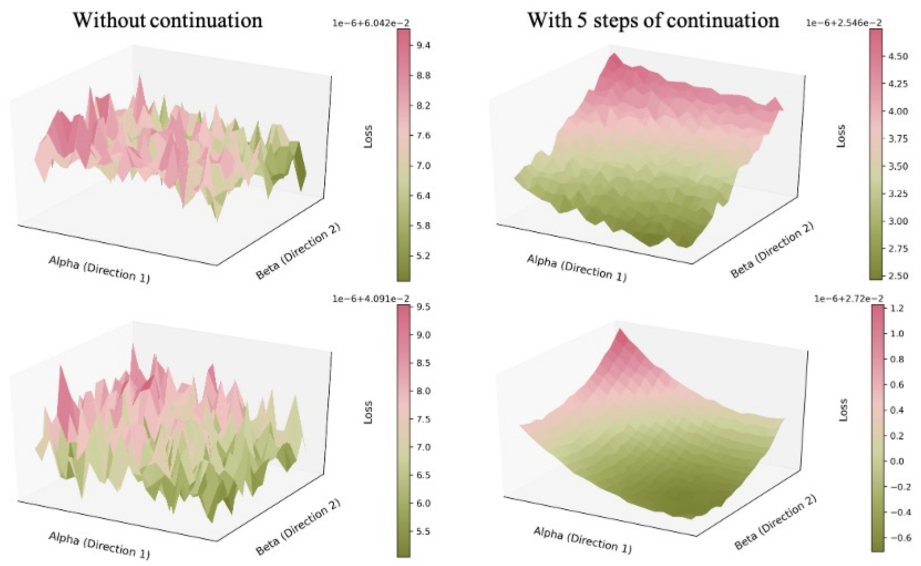
Algorithm 3: C-DEC

```
   Input  : $y \in \mathbb{R}^m$
   Output: Warm-started solution $\hat{x}^\star \in \mathbb{R}^n$ of (1)
1  Initialize $x_0^0 \in \mathbb{R}^n$;
2  for $j = 0, 1, \ldots$ do
3      $\hat{x}^{j+1} \leftarrow h(y)$;
4      $x_0^{j+1} \leftarrow \hat{x}^j + \frac{j}{j+3}(\hat{x}^{j+1} - \hat{x}^j)$;
```

TABLE I
PERFORMANCE COMPARISONS BETWEEN DECONET (MEASURED WITH THE MSE) AND C-DEC (MEASURED WITH THE LOG-COSH LOSS), ON ALL DATASETS, WITH $W \in \mathbb{R}^{50n\times n}$, $L = 10$ (TOP) AND $L = 50$ (BOTTOM). BOLD LETTERS INDICATE THE BEST PERFORMANCE.

Dataset / DUN	Test error		Emp. gen. error	
	MNIST	CIFAR10	MNIST	CIFAR10
C-DEC (proposed)	**0.029893**	**0.014344**	**0.000018**	**0.000008**
DECONET	0.055963	0.038947	0.000214	0.000139

Dataset / DUN	Test error		Emp. gen. error	
	MNIST	CIFAR10	MNIST	CIFAR10
C-DEC (proposed)	**0.025576**	**0.013918**	**0.000086**	**0.000074**
DECONET	0.061135	0.027518	0.000117	0.000161


(a) Top: $W \in \mathbb{R}^{10n \times n}$. Bottom: $W \in \mathbb{R}^{50n \times n}$.

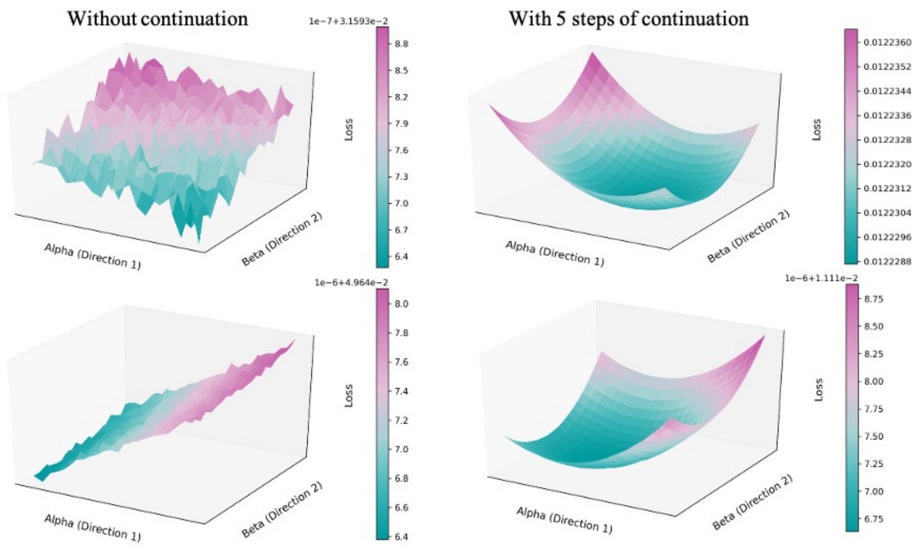

(b) Top: $W \in \mathbb{R}^{10n \times n}$. Bottom: $W \in \mathbb{R}^{50n \times n}$.

Fig. 2. Loss landscapes of 5-layer C-DEC on (a) MNIST and (b) CIFAR10 datasets.

continuation) suffers from a highly nonsmooth loss landscape, with many peaks and valleys. On the other hand, C-DEC with 5 steps of continuation enjoys a much smoother loss landscape, with the effect being highly noticeable for the CIFAR10 images, as depicted in Fig. 2b. Then, for the MNIST dataset illustrated in Fig. 2a, we notice that overparameterization also smooths out the landscape, as N increases from $10n$ to $50n$. Although this effect is negligible for DECONET, it is more noticeable for C-DEC, showcasing the potential of coupling continuation with overparameterization in the context of DUNs. All in all, results indicate that continuation (and overparameterization) has a strong positive effect on the optimization process of DUNs, and could spark an interesting mathematical theory regarding DUNs' generalization.

IV. Conclusion

In this paper, we applied a continuation technique to warm-start a state-of-the-art overparameterized unfolding network solving the compressed sensing problem. We measured the performance of the resulting continued network with an enhanced loss function, as opposed to standard metrics for unfolding networks. Empirical results highlighted the superiority of the proposed framework, thereby paving the way for creating hybrid unfolded architectures, with tools stemming from model-based methods. In the future, we would like to mathematically explore the properties of the proposed network. This could include a comprehensive optimization-based analysis similar to [41], or the study of continuation's effect on the network's robustness and generalization ability.

References

[1] M. Borgerding, P. Schniter, and S. Rangan. "AMP-Inspired Deep Networks for Sparse Linear Inverse Problems". In: *IEEE Trans. Signal Process.* 65.16 (2017), pp. 4293–4308.

[2] J. Zhang and B. Ghanem. "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing". In: *Proc. IEEE Comput. Vision and Pattern Recognit.* 2018, pp. 1828–1837.

[3] Y. Yang et al. "ADMM-CSNet: A deep learning approach for image compressive sensing". In: *Trans. Pattern Anal. and Mach. Intell.* 42.3 (2018), pp. 521–538.

[4] N. Shlezinger et al. "Model-based deep learning". In: *Proc. IEEE* 111.5 (2023), pp. 465–499.

[5] V. Kouni, H. Rauhut, and T. Theoharis. "Star DGT: a robust Gabor transform for speech denoising". In: *Sampling Theory, Signal Process., and Data Anal.* 21.1 (2023), p. 14.

[6] J. Guo et al. "ASCONVSR: Fast and lightweight super-resolution network with assembled convolutions". In: *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognit.* 2023, pp. 1582–1592.

[7] X. Li et al. "Pansharpening method based on deep nonlocal unfolding". In: *IEEE Trans. Geoscience and Remote Sensing* 61 (2023), pp. 1–11.

[8] Meihuang Wang et al. "A Deep Proximal-Unfolding Method for Monaural Speech Dereverberation". In: *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC).* IEEE. 2022, pp. 324–329.

[9] R. G. Baraniuk et al. "Model-based compressive sensing". In: *IEEE Trans. Inf. Theory* 56.4 (2010), pp. 1982–2001.

[10] V. Kouni and H. Rauhut. "Spark Deficient Gabor Frame Provides a Novel Analysis Operator for Compressed Sensing". In: *Int. Conf. Neural Inf. Process.* Springer. 2021, pp. 700–708.

[11] V. Kouni and Y. Panagakis. "DECONET: an Unfolding Network for Analysis-based Compressed Sensing with Generalization Error Bounds". In: *IEEE Trans. Signal Process.* 71 (2023), pp. 1938–1951. DOI: 10.1109/TSP.2023.3272286.

[12] V. Kouni and Y. Panagakis. "Generalization analysis of an unfolding network for analysis-based Compressed Sensing". In: *arXiv preprint arXiv:2303.05582, under review in Appl. & Comput. Harmon. Anal.* (2023).

[13] A. Landi et al. "Artificial neural networks for nonlinear regression and classification". In: *10th Int. Conf. Intell. Syst. Design and Appl.* IEEE. 2010, pp. 115–120.

[14] H. Yao et al. "Dr2-net: Deep residual reconstruction network for image compressive sensing". In: *Neurocomputing* 359 (2019), pp. 483–493.

[15] J. Gurrola-Ramos, O. Dalmau, and T. E. Alarcon. "A residual dense u-net neural network for image denoising". In: *IEEE Access* 9 (2021), pp. 31742–31754.

[16] H. Van Luong et al. "Designing interpretable recurrent neural networks for video reconstruction via deep unfolding". In: *IEEE Trans. Image Process.* 30 (2021), pp. 4099–4113.

[17] Z. Zhang et al. "AMP-Net: Denoising-Based Deep Unfolding for Compressive Image Sensing". In: *IEEE Trans. Image Process.* 30 (2021), pp. 1487–1500.

[18] J. Song, B. Chen, and J. Zhang. "Dynamic path-controllable deep unfolding network for compressive sensing". In: *IEEE Trans. Image Process.* 32 (2023), pp. 2202–2214.

[19] T. Li et al. "Gates-Controlled Deep Unfolding Network for Image Compressed Sensing". In: *IEEE Trans. Comput. Imag.* (2024).

[20] X. Wang and H. Gan. "UFC-Net: Unrolling Fixed-point Continuous Network for Deep Compressive Sensing". In: *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognit.* 2024, pp. 25149–25159.

[21] X. Wei et al. "Deep unfolding with normalizing flow priors for inverse problems". In: *IEEE Trans. Signal Process.* 70 (2022), pp. 2962–2971.

[22] E. T. Hale, W. Yin, and Y. Zhang. "A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing". In: *CAAM TR07-07, Rice University* 43.44 (2007), p. 2.

[23] E. T. Hale, W. Yin, and Y. Zhang. "Fixed-point continuation for ℓ_1-minimization: Methodology and convergence". In: *SIAM J. on Optim.* 19.3 (2008), pp. 1107–1130.

[24] S. Becker, J. Bobin, and E. J. Candes. "NESTA: A fast and accurate first-order method for sparse recovery". In: *SIAM J. on Imag. Scie.* 4.1 (2011), pp. 1–39.

[25] A. Y. Yang et al. "Fast ℓ_1-Minimization Algorithms for Robust Face Recognition". In: *IEEE Trans. Image Process.* 22.8 (2013), pp. 3234–3246.

[26] E. A. Yildirim and S. J. Wright. "Warm-start strategies in interior-point methods for linear programming". In: *SIAM J. Optim.* 12.3 (2002), pp. 782–810.

[27] A. Forsgren. "On warm starts for interior methods". In: *System Modeling and Optimization: Proc. of 22nd IFIP TC7 Conference held from July 18–22, 2005, in Turin, Italy 22.* Springer. 2006, pp. 51–66.

[28] S. R. Becker, E. J. Candes, and M. C. Grant. "Templates for convex cone problems with applications to sparse signal recovery". In: *Math. Prog. Comput.* 3.3 (2011), pp. 165–218.

[29] E. Chen et al. "Comprehensive Examination of Unrolled Networks for Linear Inverse Problems". In: *arXiv preprint arXiv:2501.04608* (2025).

[30] Yu Zhou et al. "Denoiser-Regulated Deep Unfolding Compressed Sensing with Learnable Fixed-Point Projections". In: *IEEE Trans. Circuits and Syst. for Video Technol.* (2024).

[31] B. Joukovsky et al. "Generalization error bounds for deep unfolding RNNs". In: *Uncertainty in Artif. Intell.* PMLR. 2021, pp. 1515–1524.

[32] C. Lyons, R. G. Raj, and M. Cheney. "On Generalization Bounds for Deep Compound Gaussian Neural Networks". In: *arXiv preprint arXiv:2402.13106* (2024).

[33] A. Jadon, A. Patil, and S. Jadon. "A comprehensive survey of regression-based loss functions for time series forecasting". In: *Int. Conf. Data Manage., Analytics & Innov.* Springer. 2024, pp. 117–147.

[34] R. A. Saleh and A.K. Saleh. "Statistical properties of the log-cosh loss function used in machine learning". In: *arXiv preprint arXiv:2208.04564* (2022).

[35] H. Li et al. "Visualizing the loss landscape of neural nets". In: *Adv. Neural Inf. Process. Syst.* 31 (2018).

[36] J. Wang et al. "A deep unfolding method for satellite super resolution". In: *IEEE Trans. Computa. Imag.* 8 (2022), pp. 933–944.

[37] L. Lu et al. "Dying relu and initialization: Theory and numerical examples". In: *arXiv preprint arXiv:1903.06733* (2019).

[38] N. Ketkar. "Introduction to pytorch". In: *Deep learning with python*. Springer, 2017, pp. 195–208.

[39] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[40] L. Prechelt. "Early stopping-but when?" In: *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[41] C. Liu, L. Zhu, and M. Belkin. "Loss landscapes and optimization in over-parameterized non-linear systems and neural networks". In: *Appl. and Comput. Harmon. Anal.* 59 (2022), pp. 85–116.