Recursions Are All You Need: Towards Efficient Deep Unfolding Networks

Rawwad Alhejaili[1,2] , Motaz Alfarraj[1,2], Hamzah Luqman[2,3] , and Ali Al-Shaikhi[1]
{g202112950, motaz, hluqman, shaikhi}@kfupm.edu.sa
[1] Electrical Engineering Department
[2] SDAIA-KFUPM Joint Research Center for Artificial Intelligence
[3] Information and Computer Science Department
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Abstract

The use of deep unfolding networks in compressive sensing (CS) has seen wide success as they provide both simplicity and interpretability. However, since most deep unfolding networks are iterative, this incurs significant redundancies in the network. In this work, we propose a novel recursion-based framework to enhance the efficiency of deep unfolding models. First, recursions are used to effectively eliminate the redundancies in deep unfolding networks. Secondly, we randomize the number of recursions during training to decrease the overall training time. Finally, to effectively utilize the power of recursions, we introduce a learnable unit to modulate the features of the model based on both the total number of iterations and the current iteration index. To evaluate the proposed framework, we apply it to both ISTA-Net+ and COAST. Extensive testing shows that our proposed framework allows the network to cut down as much as 75% of its learnable parameters while mostly maintaining its performance, and at the same time, it cuts around 21% and 42% from the training time for ISTA-Net+ and COAST respectively. Moreover, when presented with a limited training dataset, the recursive models match or even outperform their respective non-recursive baseline. Codes and pretrained models are available at https://github.com/Rawwad-Alhejaili/Recursions-Are-All-You-Need.

1. Introduction

Compressive sensing (CS) is an emerging field that challenges the conventions of digital data acquisition. The Nyquist sampling rate (the golden standard in signal acquisition) states that any signal x can be recovered when it is sampled at double its highest frequency. However, sampling the signal at this rate can pose some challenges. For example, in data centers, the Nyquist rate limits the amount of data stored on storage drives leading to insufficient utilization of the servers. Also, in seismic surveys, sampling the signal at the Nyquist rate is challenging either due to geological or economical constraints [15]. As such, while the Nyquist rate ensures perfect reconstruction of the signal, it can be unappealing in many cases. However, CS promises to recover signals below the Nyquist rate [1] [8] [2]. Mathematically, the recovery problem can be formulated as:

$$\mathbf{y} = \mathbf{\Phi x} \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the measurements, $\mathbf{x} \in \mathbb{R}^N$ is the original signal, and $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ is the sampling matrix.

Compressive sensing aims to reconstruct the signal (or image) when the samples $M \ll N$. This can be guaranteed with high probability when the signal is sparse enough in one domain [1] [8] [2]. This is why CS emphasizes sparsity. Representing the collapsed signal is done by:

$$\mathbf{y} = \mathbf{\Phi \Psi s} \tag{2}$$

where $\mathbf{\Psi} \in \mathbb{R}^{N \times N}$ is the sparsifying basis and $\mathbf{s}$ is the signal $\mathbf{x}$ in the sparse domain.

Although compressive sensing presents theoretical guarantees about the recoverability of the signal [2], it is still a challenging problem to solve. Finding the sparsest solution requires iterative calculations which in turn makes CS computationally expensive to run [24] [6] [23].

However, applying deep learning (DL) based approaches to compressive sensing proved to be fruitful. Neural networks are able to jointly learn the sampling matrix and the inverse mapping of the measurements y to the original signal x [21] [19] [29] [3]. Neural network-based CS methods demonstrated both better performance than handcrafted techniques and more importantly, at a lower computational complexity [28] [21] [27].

For deep learning-based CS models, deep unfolding networks are an attractive choice. Those models utilize the rich literature in the CS field by taking existing iterative handcrafted CS methods and supercharge them with the use of

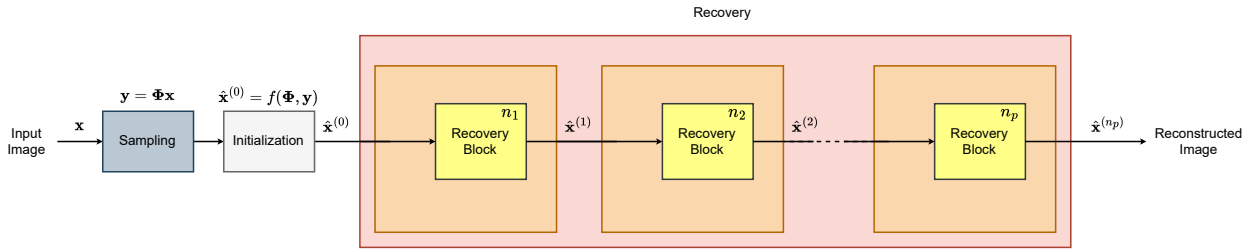


Figure 1. General structure of deep unfolding models.

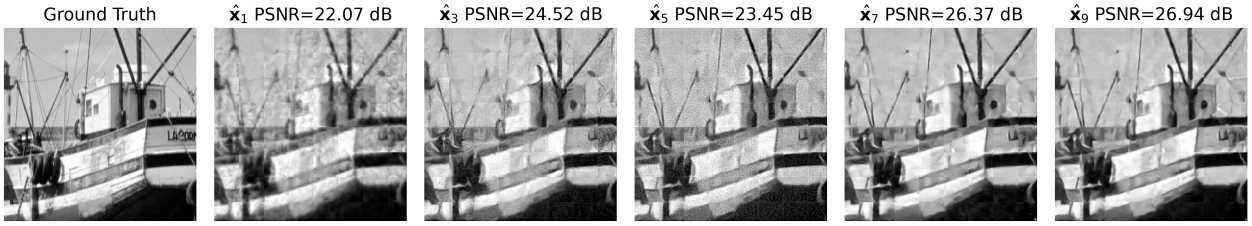


Figure 2. Intermediate CS reconstruction results of ISTA-Net [28].

neural networks [28] [22] [27]. This is done by mapping each iteration to its own neural network block. The advantage this provides is both interpretability and simplicity.

Deep unfolding models usually exhibit improved performance as the number of recovery blocks increases [28] [27]. However, we argue that this increase in performance could be attributed to the increased number of iterations rather than the increase in recovery blocks. Therefore, increasing the number of recovery blocks unnecessarily increases the number of parameters and redundancies. This makes the model more prone to overfitting necessitating a larger and more diverse training dataset [26] and can also limit its deployment options [4].

In this work, we propose a recursive framework to increase the efficiency of deep unfolding networks in terms of training time, number of parameters, and training data efficiency. Specifically, our contribution is three-fold:

- First, we cut down on the redundancies in the deep unfolding networks by introducing recursions. This allows the model to increase its utilization of each layer's capacity before moving on to the next layer or block.
- Secondly, we propose the use of random recursions during training to decrease the training time and allow the model to work with varying degrees of recursions.
- Finally, we acknowledge that a naive implementation of recursions may not be optimally efficient. Therefore, we leverage a learnable unit to modulate the features of the model based on the total number of iterations and the current iteration index.

The proposed framework is applied to both ISTA-Net+ [28] and COAST [27]. The results on COAST show that the proposed framework decreases the training time by 42% and the learnable parameters by almost 75% while sustaining a minimal impact on its performance. In the case of ISTA-Net+, we observe similar trends. The recursive framework decreases the training time by 21% and the learnable parameters by 66%. However, we find that the recursive ISTA-Net+ model slightly outperforms its baseline model. Furthermore, when the size of the training data is limited, the recursive models either match their respective baselines (which is the case for COAST) or even outperform the baseline in the case of ISTA-Net+. Therefore, the recursive models are more resilient to overfitting thanks to the inherent reduction of their learning parameters.

2. Literature Review

Previous work in compressive sensing can be categorized into two categories: handcrafted iterative optimization-based methods and data-driven deep learning-based methods. The data-driven methods can also be categorized into two categories, i.e., deep straightforward networks and deep unfolding networks.

2.1. Iterative Optimization Based Methods

Generally, inverse problems such as in Eq. (1) are impossible to solve as the number of unknowns N is usually much greater than the number of measurements M. However, compressive sensing theory states that if the signal x is sparse in some domain, then it can be recovered with high probability. Ideally, the recovered signal will have the low-

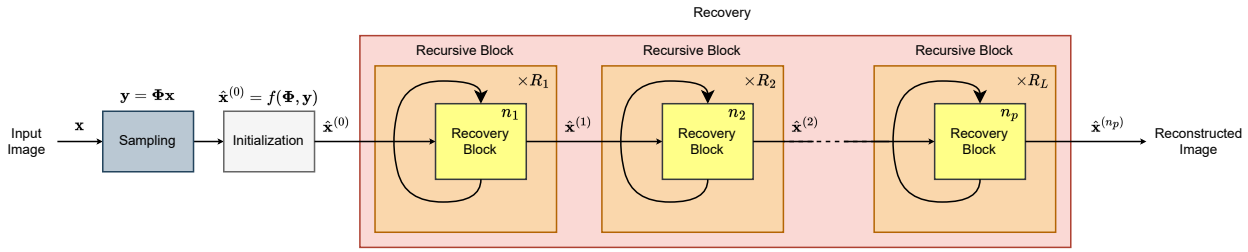


Figure 3. General architecture of the recursive framework. Compared to general deep unfolding models such as COAST [27] and ISTA-Net+ [28], R_i recursions are used in each recovery block i in the recovery subnet.

est l_0-norm [8]. Though, l_0-norm minimization is a non-convex optimization problem and is prone to combinatorial complexity [8]. As such, computing the minimum l_0-norm suffers from a severe computational cost as the signal x increases in size. Others have proposed minimizing the l_1-norm instead [2], which translates the problem into a convex optimization problem. Such methods include orthogonal matching pursuit (OMP) [24], least absolute shrinkage and selector operator (LASSO) [23], and iterative shrinkage-thresholding algorithm (ISTA) [6]. Other works have proposed minimizing the total variation (TV) instead, which include methods such as TVAL3 [12] and denoising-based approximate message passing (D-AMP) [14]. Overall, while handcrafted methods have made strides in decreasing their associated computational complexity, they still remain computationally demanding due to their reliance on iterative optimizations.

2.2. Deep Straightforward Models

Briefly, deep straightforward networks model the CS problem purely for neural networks and are not based on a handcrafted CS method. [16] proposes stacked denoising autoencoder (SDA), which uses multi-layer perceptrons (MLPs) for both the sampling and reconstruction stages. ReconNet [11] uses a random Gaussian matrix in the sampling stage and then uses convolutional neural networks (CNNs) for reconstruction followed by a conventional denoiser (BM3D [5]). Shi *et al*. [21] proposed CSNet. It jointly learns both the sampling matrix and the inverse mapping from the measurements to the original signal. Notably, it introduced the use of convolutions to learn the sampling matrix. Furthermore, other works [19] enforce constraints on the learned sampling matrices (such as limiting them to binary numbers), so that they can be applied more easily to hardware solutions. In [19], they show that the learned sampling matrices can improve the reconstruction results even when applied to handcrafted methods.

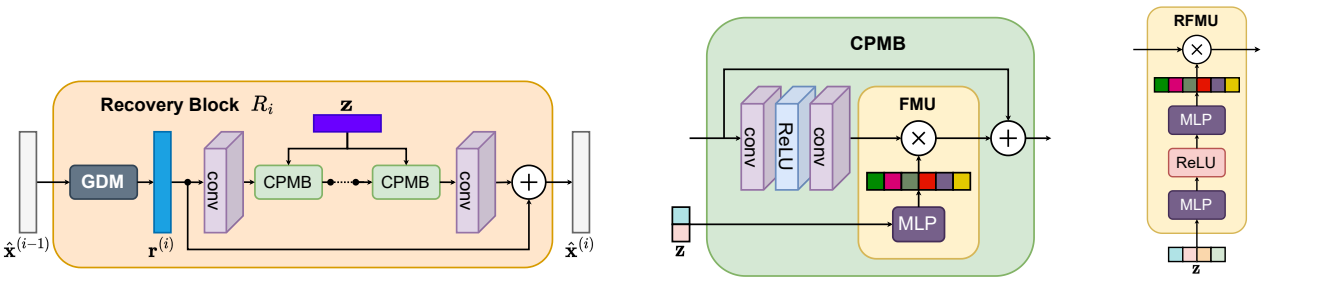
Many CS architectures suffer from being limited to a single CS ratio (and have to be retrained for each specific ratio). Wuzhen Shi *et al*. proposes SCSNet [20], which solves the problem by adding enhancement layers. This allows the model to tackle multiple CS ratios without the need for retraining. A more recent approach by Vladislav Kravets & Adrian Stern [10] solves this by progressively sampling the image at different scales where each scale builds on the measurements obtained from the last one. In other words, they reconstruct the low frequencies first and then progressively reconstruct higher frequencies. As such, the model can handle different CS ratios by design.

2.3. Deep Unfolding Models

Deep unfolding models are neural networks that are designed with handcrafted iterative CS techniques as their basis. They map each iteration of the handcrafted technique to a neural network block which allow them to utilize the speed and performance of neural networks while preserving the interpretability of the handcrafted techniques. It also helps to keep the network design fairly simple.

One such network is ISTA-Net [28]. As the name implies, the network revolves around the iterative shrinkage-thresholding algorithm (ISTA) [6]. The performance of ISTA highly depends on the sparsity of the input, so finding the best domain that ensures the highest sparsity is crucial for this algorithm. As such, ISTA-Net maps each iteration of ISTA to a corresponding block and utilizes the representative power of CNNs to find the sparsest possible transform. By default, the network uses 9 blocks (with an identical structure) to reconstruct the image. However, the network requires training for each specific CS ratio meaning a single model cannot handle multiple CS ratios. Also, at low CS ratios, the network suffers from blocking artifacts due to its block-based recovery. OPINE-Net [29] modifies ISTA-Net by letting it learn its optimal sampling matrix.

COAST [27] is another deep unfolding network that is built on top of ISTA-Net. It solves the main two issues with ISTA-Net. First, to promote the network to recover the image from different CS ratios, they introduce random projection augmentation (RPA), which exposes the network during training to many different sampling matrices at different CS ratios. Furthermore, they add a controllable unit



(a) Recovery block of COAST [27]. (b) CPMB block of COAST [27]. (c) The proposed RFMU unit.

Figure 4. Detailed look into the recovery block of COAST [27] and the RFMU unit. Note that the RFMU unit has two extra parameters in $\mathbf{z}$ compared to the FMU unit in COAST [27], and they are the current recursion index R_{cur} and the total number of iterations R_{tot}.

that modulates the features based on the CS ratio. Secondly, they address the blockification issue of ISTA-Net by introducing a plug-and-play deblocking module that deblockifies the features before feeding them to the CNN layers. On top of solving the issues of ISTA-Net, the model also introduces changes to the recovery stage of ISTA-Net for optimal recovery, and instead of using 9 blocks for recovery, the model now uses 20 blocks (with identical structure).

Regarding recursions specifically, there have been works that used recursions to decrease the number of learnable parameters. DPDNN [7] is one such network. It uses an encoder-decoder network in the recovery block to reconstruct the image over multiple blocks. However, all blocks share the same learning parameters, and thus, it can be described by Fig. 3 as having a single recovery block but with $R_1 = 6$. The authors of ISTA-Net [28] have also tested sharing the learnable parameters across all recovery blocks, which can be described by Fig. 3 as having a single recovery block and $R_1 = 9$, and they found that the model still performed competently despite the significant decrease in the number of parameters.

With that said, the naive implementation of recursions leaves a lot to be desired. For one, the previous methods used a fixed number of recursions during training, and thus, did not enjoy any reduction to the training time. Secondly, the features of the model were not modulated to account for recursions. This is a crucial point as different features need to be emphasized depending on the iteration index and will enable efficient use out of recursions. Finally, recursions on those models were applied when the network only had a single recovery block, thus, its recovery potential was limited to the capacity of that single block.

Now, we move on to what would be our major contribution in this work. From Fig. 1, we see that in general, deep unfolding models reconstruct the image iteratively over multiple blocks but with identical structures. Fig. 2 shows how such an iterative model can improve the results in each phase or block, and how in the specific case of ISTA-Net [28], the model can possibly improve its results with extra phases. However, this approach is unattractive as it increases the number of parameters, and as such, runs the risk of vanishing gradients and overfitting, which is especially unappealing when the size of the training dataset is small [26].

To reduce the redundancy in the recovery phases, we propose the use of recursions. The intuition behind this is instead of passing the reconstructed signal to the next recovery block in each iteration, we will feed it back to the same block for R_i iterations until the point of diminishing returns (ideally). This should provide us with three major advantages.

- First, we would be able to realize more of the capacity of each recovery block, allowing for more efficient use of them, which will lead to a lower number of trainable parameters.
- Secondly, the training time can be decreased by randomizing the number of recursions per recovery block in each training iteration.
- Thirdly, due to the decrease in the number of parameters, recursive models are more resilient to overfitting and thus, can perform better when the training dataset is small.

Finally, do note that the use of recursions should not be limited to compressed sensing applications. We believe that it might be generalizable to most image restoration tasks such as reconstruction, denoising, and interpolation as long as both the input and the output are in the same domain. Deep unfolding models lend themselves especially well for recursions due to their iterative design, which usually ensures that both the input and the output remain in the same domain.

3. Methodology

COAST [27] will be used to demonstrate how the recursive framework can be applied to a deep unfolding network.

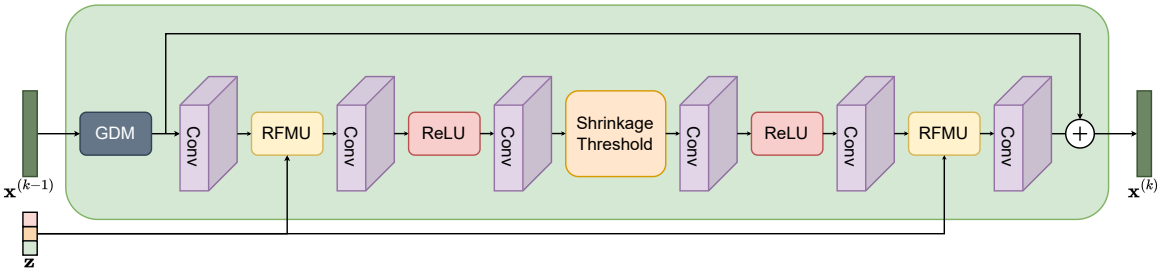


Figure 5. Recovery block of ISTA-Net+ [28] after the addition of the RFMU unit.

We will discuss the general details of COAST and how the recursive framework was applied to it. Then, we will discuss how the framework modulates the features of the network based on recursions. Finally, we discuss how we modulate the features of the network based on recursions.

3.1. COAST

The COAST model follows the general structure outlined in Fig. 1. The image is sampled using random Gaussian matrices with orthogonal rows, and the initialization is done by:

$$\hat{\mathbf{x}}^{(0)} = \mathbf{\Phi}^T \mathbf{y} \tag{3}$$

Afterward, the initial solution is fed to the recovery network.

The fundamental recovery block of COAST is illustrated in Fig. 4a. First, the image is fed to a gradient descent module (GDM) to preserve the ISTA [6] structure. It is computed by:

$$\mathbf{r}^i = \hat{\mathbf{x}}^{(i-1)} - \rho^{(i)} \mathbf{\Phi}^T (\mathbf{\Phi} \hat{\mathbf{x}}^{(i-1)} - \mathbf{y}) \tag{4}$$

where ρ is the learning rate. Next, the image is fed to a convolutional layer to project it to the feature space with $C = N_f$. Then, a stack of controllable proximal mapping blocks (CPMB) is used to solve the proximal mapping problem of ISTA, which is defined as:

$$\mathbf{prox}_{\lambda\psi}(\mathbf{r}) = \arg\min_{\hat{\mathbf{x}}} \frac{1}{2} \|\hat{\mathbf{x}} - \mathbf{r}\|_2^2 + \lambda \psi(\hat{\mathbf{x}}) \tag{5}$$

where λ is a regularization parameter, and $\psi(\cdot)$ is a transformation function that is now learned by the CPMB block. Furthermore, to let the model better adapt to the different CS ratios and noise levels, [27] proposes a feature modulation unit (FMU) that modulates the features of the CPMB block based on the CS ratio and noise level. Specifically, the model employs a linear layer that takes those two parameters and outputs N_f modulation coefficients where each one is used to modulate its respective feature in the CPMB block. Finally, a convolutional layer is used to project the image from the feature space back to the spatial domain.

3.2. Recursions

Since COAST [27] follows the general structure of deep unfolding models (shown in Fig. 1), introducing recursions to it is fairly straightforward. A feedback connection is added from the output of each recovery phase back to its input, and R_i denotes the number of iterations per layer (IPL). The modified network will now have the same structure seen in Fig. 3.

To further promote the model to learn recursions, we will introduce random recursions. In each training iteration, we will randomize the IPL of each layer (with R_i being the highest number of IPL and 1 being the lowest). This will allow the model to generalize better to a wide range of recursions instead of being trained for a single set of recursions. Additionally, it will decrease the training time since on average fewer FLOPS will be computed during training.

3.3. Recursion-based Feature Modulation Unit

To further tailor the model for recursions, we introduce a recursion-based feature modulation unit (RFMU) During training, it was observed that although naively adding recursions was sufficient to enhance the performance of the framework, it was not an optimal approach. Each phase cannot anticipate the level of quality of its input, and therefore, its features will not optimally enhance the results. Inspired by [27], we extend the feature modulation unit seen in Fig. 4b by adding recursion statistics to $\mathbf{z}$. Specifically, the proposed RFMU unit will now modulate the features of the network based on two extra parameters. First, on how many iterations have been completed so far (denoted as R_{cur}), and second, the total number of iterations (which is $R_{tot} = \sum_i R_i$). For that end, we employ two linear layers separated by ReLU [17] that take the input vector $\mathbf{z}$ with the extra two parameters and extract N_f modulation coefficients (denoted as σ_c). Finally, each feature in the output of the CPMB block will be modulated by its corresponding modulation coefficient σ_c. The figure of the RFMU unit is seen in Fig. 4c.

Originally, ISTA-Net+ [28] does not use a feature modulation unit, however, adding the RFMU to it is fairly simple and is shown in Fig. 5. The only modification done to the

Training Data %	Model	Configuration	Layers	IPL	CS Ratio 10%	20%	30%	40%	50%	Avg. Score	Parameters
100 %	ISTA-Net+ [28]	Baseline [28]	9	1	25.18/0.6971	28.11/0.8161	30.21/0.8769	32.12/0.9161	33.94/0.9421	29.91/0.8497	336,978
		Recursive	3	3	**25.31/0.7010**	**28.21/0.8177**	**30.30/0.8786**	**32.21/0.9169**	**34.05/0.9428**	**30.02/0.8514**	**114,630**
	COAST [27]	Baseline [27]	20	1	**26.24/0.7378**	**28.93/0.8378**	**30.99/0.8918**	**32.82/0.9255**	**34.58/0.9484**	**30.71/0.8683**	1,122,024
		Recursive	5	4	26.14/0.7329	28.79/0.8330	30.83/0.8879	32.68/0.9226	34.46/0.9463	30.58/0.8645	**281,674**
3 %	ISTA-Net+ [28]	Baseline [28]	9	1	24.28/0.6614	27.06/0.7823	29.04/0.8498	30.87/0.8960	32.54/0.9264	28.76/0.8232	336,978
		Recursive	3	3	**24.71/0.6799**	**27.57/0.8017**	**29.65/0.8659**	**31.49/0.9066**	**33.20/0.9341**	**29.32/0.8376**	**114,630**
	COAST [27]	Baseline [27]	20	1	**25.88**/0.7226	28.55/0.8275	**30.63/0.8848**	**32.44/0.9201**	**34.14/0.9440**	**30.33**/0.8598	1,122,024
		Recursive	5	4	25.87/**0.7252**	**28.57/0.8283**	30.61/0.8846	32.41/0.9198	34.13/0.9438	30.32/**0.8603**	**281,674**

Table 1. PSNR/SSIM scores of the models on the BSD68 [13] dataset.

Model	Configuration	Layers	IPL	Training Time (Hours)	Inference Time (FPS)	Parameters
ISTA-Net+ [28]	Baseline [28]	9	1	3.22	**35.62**	336,978
	Recursive	3	3	**2.53**	29.29	**114,630**
COAST [27]	Baseline [27]	20	1	18.14	**12.27**	1,122,024
	Recursive	5	4	**10.50**	12.20	**281,674**

Table 2. Training time and number of parameters of the models in Tab. 1. Note that the training time for ISTA-Net+ [28] was computed on a single CS ratio (10%).

network was the addition of the RFMU after the first convolutional layer and another RFMU before the last convolutional layer. Other than this and the inclusion of recursions, the recursive ISTA-Net+ model is identical to its reference in [28].

4. Experimental Results

For a fair comparison, all models were trained using the same dataset used in ISTA-Net+ [28] and COAST [27], which features 88,912 blocks of dimension 33×33 obtained from 91 images. All models were trained using CS ratios from 10% up to 50%. The sampling matrix used was the random Gaussian matrix with orthogonal rows. For COAST however, we have generated multiple random Gaussian matrices for each CS ratio to faithfully follow their RPA training strategy [27]. The cost function was the mean-square error (MSE). The optimizer used was Adam [9] with the momentum set to 0.9, weight decay set to 0.999, and the learning rate set to 1×10^{-4}. The number of epochs was 400 for COAST, but for ISTA-Net+ [28], it was trained for 200 epochs specifically as this is what was used in the original work [28]. All models were implemented using PyTorch [18], and they were trained using a single Nvidia RTX 3090 GPU. For validation, we use the Set11 dataset [11] while for testing, we use the BSD68 dataset [13]. As for the evaluation metrics, we use the peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) [25].

4.1. COAST vs. Recursive COAST

To demonstrate the effectiveness of the recursive framework, we apply it to COAST [27]. The first model will use the same configuration as in [27] (labeled COAST). That is, it will have 20 layers without recursions. The second model will have 5 layers and 4 iterations per layer (labeled R-COAST).

Tab. 1 shows the PSNR/SSIM scores for the BSD68 dataset. When both models have access to the entire training dataset, we observe that despite the recursive model only using 25% the number of parameters of the baseline, it still achieves competitive results in terms of both the PSNR and SSIM metrics. However, when we limit the size of the training dataset to only 3% of its original size, we find that the gap is mostly eliminated and both models achieve largely the same performance.

Finally, although the recursive model sees a minimal impact on its performance, from Tab. 2, we see that it takes 41% less time to train and sees almost a 75% reduction in its learnable parameters with a negligible impact on the inference time. Therefore, we can conclude that the recursive framework increased the efficiency of the COAST model with a minimal impact on its performance.

4.2. ISTA-Net+ vs Recursive ISTA-Net+

To investigate the generalizability of the recursive framework, we will compare the ISTA-Net+ [28] model with and without the recursive framework (labeled RISTA-Net+ and ISTA-Net+ respectively). We will train two models. One

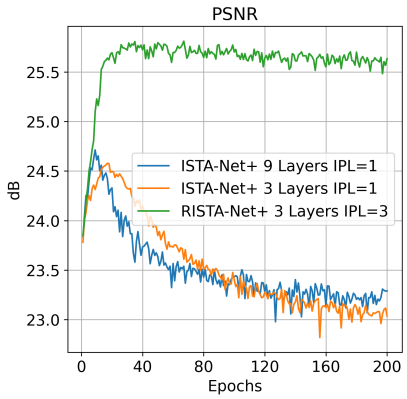
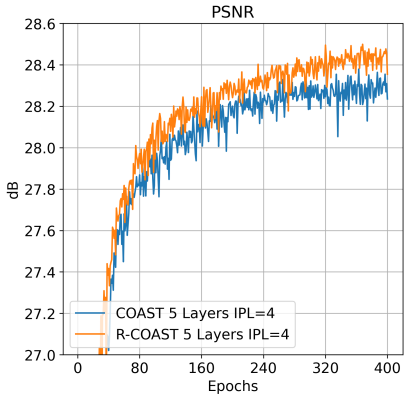
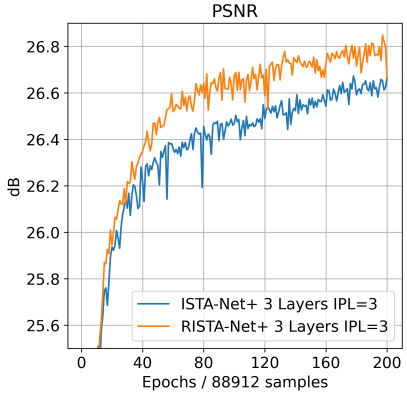


Figure 6. The learning curves of ISTA-Net+ [28] with and without recursions on the validation dataset (Set11 [11]). All models were trained using a CS ratio of 10% and with access to only 3% of the training dataset.



(a) COAST [27] with and without the RFMU



(b) ISTA-Net+ [28] with and without the RFMU

Figure 7. The learning curves of COAST [27] and ISTA-Net+ [28] with the RFMU (labeled R-COAST and RISTA-Net+ respectively) and without it (labeled COAST and ISTA-Net+ respectively) on the validation dataset (Set11 [11]) and at a CS ratio of 10%.

will use 9 layers with no recursions (the same configuration as in [28]), and the other model will use 3 layers and 3 IPL. The results are shown in Tab. 1.

On the testing dataset, we find that on average, the recursive configuration slightly outperforms the baseline model while using 66% fewer parameters. However, when the training dataset is limited to only 3% of its samples, we observe that the recursive configuration clearly outperforms the baseline. This is thanks to its inherently small number of parameters and therefore, it is more resilient to overfitting. This is also confirmed when looking at its learning curve in Fig. 6. Here, we observe that the recursive configuration is able to generalize better than the non-recursive model. We also include the results of a model with 3 layers (like RISTA-Net+) but without recursions, and the results in Fig. 6 clearly show that the PSNR improvement RISTA-Net+ sees is *not* attributed to its smaller size but rather its recursive design.

Tab. 2 shows the training time and the number of parameters of the models. Here, it is observed that the recursive configuration takes 21% less time during training thanks to the use of randomized recursions during training while only sustaining a minimal impact to the inference time. In general, we can conclude that applying the recursive framework to ISTA-Net+ [28] allows it to decrease its training time and number of parameters while maintaining the same performance when the training dataset is large, but when it is limited, the recursive model outperforms the baseline.

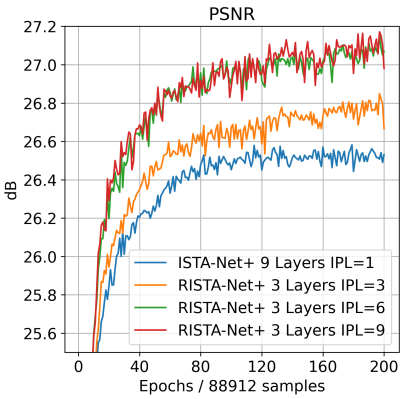
5. Ablation Study

5.1. RFMU Enhances the Efficiency of Recursions

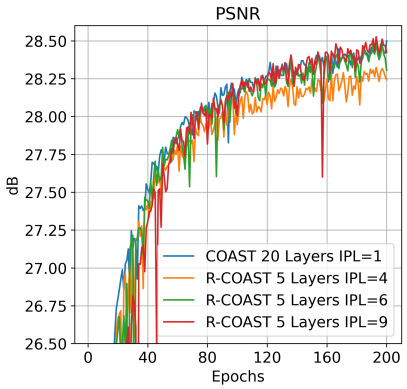
One of the problems in the naive implementation of recursions is that there can be a drastic difference in quality between each iteration (especially in the first iterations). More importantly, the model *cannot* anticipate the level of quality of the input as it does not have access to the iteration index. Thus, it cannot modulate its features accordingly. In other words, the first iteration should have different weights than the last iteration, or at least some weights should be emphasized over the others. We proposed that modulating the features of the model based on recursion statistics improves its results. To investigate this, we compare the use of the RFMU in both COAST [27] and ISTA-Net+ [28]. The results are shown in Fig. 7.

Fig. 7a shows the results of COAST. Both models have 5 layers and 4 IPL, but one utilizes the RFMU (labeled R-COAST) while the other model does not (labeled COAST). Here, it is shown that the R-COAST model performs better as it achieved a higher PSNR score of around 28.50 dB while the COAST model without the RFMU scored 28.38 dB. From this, it can be concluded that the RFMU does indeed improve the performance of the model.

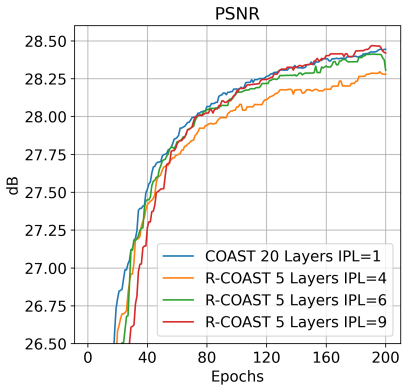
The RFMU results of ISTA-Net+ are shown in 7b. We trained both models to have 3 layers and 3 IPL, but again,



(a) Learning curves of ISTA-Net+ [28].



(b) Learning curves of COAST [27].



(c) Learning curves of COAST [27] after passing through a median filter (to enhance visualization).

Figure 8. The learning curves of ISTA-Net+ [28] and COAST [27] using different IPL values.

one model foregoes the use of the RFMU (labeled ISTA-Net+) while the other model does (labeled RISTA-Net+). From Fig. 7b, we observe that RISTA-Net+ can outperform ISTA-Net+ thanks to its use of the RFMU.

Given the results in Fig. 7, we can conclude that the addition of the RFMU allows the model to more efficiently utilize the recursive connection to achieve better results.

5.2. Increasing the Number of Recursions

In the main results section, the maximum number of iterations per layer we performed was 4. The IPL was specifically chosen so that the total number of iterations for both the recursive model and the baseline remains the same. This raises the question: Can we see improved performance by increasing the IPL? And at what point would we experience diminishing returns?

To investigate this, we trained both RISTA-Net+ and R-COAST using three different IPLs. For RISTA-Net+, all configurations used 3 layers and the chosen IPLs were 3, 6, and 9. For R-COAST, we settled on using 5 layers and the IPLs were 4, 6, and 9. The resulting learning curves in the validation dataset for both models are seen in Fig. 8 respectively.

From the results in Fig. 8, it is observed that both models see an increase in performance when the IPL is increased to 6. However, increasing the IPL to 9 sees a minimal increase in the performance, and thus, the tests indicate that 6 iterations per layer is the highest IPL possible before experiencing diminishing returns. Interestingly, when the R-COAST model uses 6 IPL, it actually closes the gap between it and the baseline model as it achieves roughly the same performance.

6. Conclusion

In this work, we have discussed how most deep unfolding networks suffer from redundancies due to their iterative design. We argued that this can be mostly eliminated by using recursions. To demonstrate this, we applied our proposed framework to both COAST [27] and ISTA-Net+ [28]. In general, our findings confirm that the iterative design of such networks suffers from redundancies and that the use of the recursive framework successfully eliminates a lot of those redundancies. It can also decrease the training time by using randomized IPL during training. We believe that the most significant part of the recursive framework is its simplicity as this could allow it to be generalized to other networks since the core concept of the framework revolves around recursions, which are fairly simple to implement to other iterative networks. We believe that it is possible to extract even more utilization out of recursions as we have shown how implementing a simple block such as the RFMU can improve the efficiency of the framework. For future work, we would like to perform a more involved recursive framework and implement the recursions on the feature space instead of the spatial domain, as the increase of features could allow for more efficient use of recursions.

7. Acknowledgement

Author(s) would like to acknowledge the support received from Saudi Data and AI Authority (SDAIA) and King Fahd University of Petroleum and Minerals (KFUPM) under SDAIA-KFUPM Joint Research Center for Artificial Intelligence.

References

[1] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022. 1

[2] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006. 1, 3

[3] Bin Chen and Jian Zhang. Content-aware scalable deep compressed sensing. *IEEE Transactions on Image Processing*, 31:5412–5426, 2022. 1

[4] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017. 2

[5] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 3

[6] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004. 1, 3, 5

[7] Weisheng Dong, Peiyao Wang, Wotao Yin, Guangming Shi, Fangfang Wu, and Xiaotong Lu. Denoising prior driven deep neural network for image restoration. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2305–2318, 2018. 4

[8] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer New York, 2013. 1, 3

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[10] Vladislav Kravets and Adrian Stern. Progressive compressive sensing of large images with multiscale deep learning reconstruction. *Scientific Reports*, 12(1):7228, May 2022. 3

[11] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458, 2016. 3, 6, 7

[12] Chengbo Li, Wotao Yin, Hong Jiang, and Yin Zhang. An efficient augmented lagrangian method with applications to total variation minimization. *Computational Optimization and Applications*, 56(3):507–530, 2013. 3

[13] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001. 6

[14] Christopher A Metzler, Arian Maleki, and Richard G Baraniuk. From denoising to compressed sensing. *IEEE Transactions on Information Theory*, 62(9):5117–5144, 2016. 3

[15] W.A. Mousa. *Advanced Digital Signal Processing of Seismic Data*. Cambridge University Press, 2020. 1

[16] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *2015 53rd annual allerton conference on communication, control, and computing (Allerton)*, pages 1336–1343. IEEE, 2015. 3

[17] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 5

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[19] Wuzhen Shi, Feng Jiang, Shaohui Liu, and Debin Zhao. Image compressed sensing using convolutional neural network. *IEEE Transactions on Image Processing*, 29:375–388, 2019. 1, 3

[20] Wuzhen Shi, Feng Jiang, Shaohui Liu, and Debin Zhao. Scalable convolutional neural network for image compressed sensing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12282–12291, 2019. 3

[21] Wuzhen Shi, Feng Jiang, Shengping Zhang, and Debin Zhao. Deep networks for compressed image sensing. *CoRR*, abs/1707.07119, 2017. 1, 3

[22] Jian Sun, Huibin Li, Zongben Xu, et al. Deep admm-net for compressive sensing mri. *Advances in neural information processing systems*, 29, 2016. 2

[23] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 1, 3

[24] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007. 1, 3

[25] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[26] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019. 2, 4

[27] Di You, Jian Zhang, Jingfen Xie, Bin Chen, and Siwei Ma. Coast: Controllable arbitrary-sampling network for compressive sensing. *IEEE Transactions on Image Processing*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[28] Jian Zhang and Bernard Ghanem. Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In *CVPR*, pages 1828–1837, 2018. 1, 2, 3, 4, 5, 6, 7, 8

[29] Jian Zhang, Chen Zhao, and Wen Gao. Optimization-inspired compact deep compressive sensing. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):765–774, 2020. 1, 3