LEARNED CONVOLUTIONAL SPARSE CODING

Hillel Sreter   *Raja Giryes* *

School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel, {hillelsr@mail., raja@}tau.ac.il

ABSTRACT

We propose a convolutional recurrent sparse auto-encoder model. The model consists of a sparse encoder, which is a convolutional extension of the learned ISTA (LISTA) method, and a linear convolutional decoder. Our strategy offers a simple strategy for learning a task-driven sparse convolutional dictionary (CD), and producing an approximate convolutional sparse code (CSC) over the learned dictionary. We trained the model to minimize reconstruction loss via gradient decent with back-propagation and have achieved competitve results to KSVD image denoising and to leading CSC methods in image inpainting requiring only a small fraction of their runtime.

Index Terms— Sparse Coding, ISTA, LISTA, Convolutional Sparse Coding, Neural networks.

1. INTRODUCTION

Sparse coding (SC) is a powerful and popular tool used in a variety of applications from classification and feature extraction, to signal processing tasks such as enhancement, super-resolution, etc. A classical approach to use SC with a signal is to split it into segments (or patches), and solve for each

$$\min ||\mathbf{z}||_0 \ s.t. \ \mathbf{x} = \mathbf{Dz}, \tag{1}$$

where $\mathbf{z} \in \mathbb{R}^m$ is the sparse representation of the (column stacked) patch $\mathbf{x} \in \mathbb{R}^n$ in the dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$. There are two painful points in this approach: (i) performing SC over all the patches tends to be a slow process; and (ii) learning a dictionary over each segment independently loses spatial information outside it such as shift-invariance in images.

One prominent approach for addressing the first point is using approximate sparse coding models such as the learned iterative shrinkage and thresholding algorithm (LISTA) [1]. LISTA is a recurrent neural network designed to mimic ISTA [2], which is an iterative algorithm for approximating the solution of (1). As to the second point, one may impose an additional prior on the learned dictionary such as being convolutional, i.e., a concatenation of Toeplitz matrices. In this case each element (known also as atom) of the dictionary is learned based on the whole signal. Moreover, the resulting dictionary is shift-invariant due to it being convolutional.

In this paper, we introduce a learning method for task driven CD learning. We design a convolutional neural network that both learns the SC of a family of signals and their CSCs. We demonstrate the efficiency of our new approach in the tasks of image denoising and inpainting.

2. THE SPARSE CODING PROBLEM

2.1. Sparse coding and ISTA

Solving (1) directly is a combinatorial problem, thus, its complexity grows exponentially with m. To resolve this deficiency, many approximation strategies have been developed for solving (1). A popular one relaxes the ℓ_0 pseudo-norm using the ℓ_1 norm yielding (the unconstrained form):

$$\arg\min_{\mathbf{D},\mathbf{z}} \frac{1}{2}||\mathbf{x} - \mathbf{Dz}||_2^2 + \lambda||\mathbf{z}||_1. \tag{2}$$

A popular technique to minimize (2) is the ISTA [2] iteration:

$$\mathbf{z_{k+1}} = S_{\lambda/L}(\mathbf{z_k} + \frac{1}{L}\mathbf{D}^T(\mathbf{Dx} - \mathbf{z}_k)), \tag{3}$$

where $L \le \sigma_{max}(\mathbf{D}^T\mathbf{D})$, $\sigma_{max}(\mathbf{A})$ is the largest eigenvalue of $\mathbf{A}$ and $S_\theta(x)$ is the soft thresholding operator:

$$S_\theta(x) = \text{sign}(x)\max(|x| - \theta, 0). \tag{4}$$

ISTA iterations are stopped when a defined convergence criterion is satisfied.

2.2. Learned ISTA (LISTA)

In approximate SC, one may build a non-linear differentiable encoder that can be trained to produce quick approximate SC of a given signal. In [1], an ISTA like recurrent network denoted by LISTA is introduced. By rewriting (3) as

$$\mathbf{z_{k+1}} = S_{\lambda/L}((\mathbf{I} - \mathbf{D}^T\mathbf{D})\mathbf{z_k} + \mathbf{D}^T\mathbf{x}), \tag{5}$$

LISTA can be derived by substituting in (5) $\mathbf{W_e} \in \mathbb{R}^{m \times n}$ for $\mathbf{D}^T$, $\mathbf{S} \in \mathbb{R}^{m \times m}$ for $I - \frac{1}{L}\mathbf{D}^T\mathbf{D}$ and $\theta \in \mathbb{R}_+^m$ for $\frac{\lambda}{L}$ (using a different threshold value for each entry instead of a single threshold for all of them [3]). Thus, LISTA iteration reads:

$$\mathbf{z}_0 = 0, \ k = 0..K - 1$$
$$\mathbf{z}_{k+1} = S_\theta(\mathbf{Sz}_k + \mathbf{W_e x}) \tag{6}$$
$$\mathbf{z}_{LISTA} = \mathbf{z}_K,$$

ICASSP 2018

where K is the number of LISTA iterations. The parameters $\mathbf{W_e}$, $\mathbf{S}$ and θ are learned by minimizing:

$$\mathcal{L} = \frac{1}{2}||\mathbf{z}^* - \mathbf{z}_{LISTA}||_2^2, \qquad (7)$$

where $\mathbf{z}^*$ is either the optimal SC of the input signal (if attainable) or the ISTA final solution after its convergence.

There have been lots of work on LISTA like architectures. In [3], a LISTA auto-encoder is introduced. Rather than training to approximate the optimal SC, LISTA is trained directly to minimize the optimization objective (2) instead of (7). A discriminative non-negative version of LISTA is introduced in [4]. It contains a decoder that reconstructs the approximate SC back to the input signal and a classification unit that receives the approximate SC as an input. Thus, the model described in [4] is named discriminative recurrent sparse auto-encoder and the quality of the produced approximate SC is quantified by the success of the decoder and classifier. The work in [5] used a cascaded sparse coding network with LISTA building blocks to fully exploit the sparsity of an image for the super resolution problem. In [6] and [7], a theoretical explanation is provided for the reason LISTA like models are able to accelerate SC iterative solvers.

3. CONVOLUTIONAL SPARSE CODING

The CSC model [8], [9], [10], [11], [12] can be derived from the classical SC model by substituting matrix multiplication with the convolution operator:

$$\mathbf{x} = \sum_{i=0}^{m-1} \mathbf{d}_i * \mathbf{z}_i, \qquad (8)$$

where $\mathbf{x} \in \mathbb{R}^{n_1 \times n_2}$ is the input signal, $\mathbf{d}_i \in \mathbb{R}^{s \times s}$ a local convolution filter and $\mathbf{z}_i \in \mathbb{R}^{n_1 \times n_2}$ a sparse feature map of the convolutional atom $\mathbf{d}_i$. The ℓ_1 minimization problem in (2) for CSC may be formulated as:

$$\arg\min_{\mathbf{d},\mathbf{z}} \frac{1}{2}||\mathbf{x} - \sum_{i=0}^{m-1} \mathbf{d}_i * \mathbf{z}_i||_2^2 + \lambda \sum_{i=0}^{m-1} ||\mathbf{z}_i||_1. \qquad (9)$$

It is important to note that unlike traditional SC, $\mathbf{x}$ is not split into patches (or segments) but rather the CSC is of the full input signal. The CSC model is inherently spatially invariant, thus, a learned atom of a specific edge orientation can globally represent all edges of that orientation in the whole image. Unlike CSC, in classical SC multiple atoms tend to learn the same oriented edge for different offsets in space.

Various methods have been proposed for solving (9). The strategies in [8] and [9] involve transformation to the frequency domain and optimization using Alternating Direction Method of Multipliers (ADMM). These methods tend to optimize over the whole train-set at once. Thus, the whole train-set must be held in memory while learning the CDC, which of course limits the train-set size. Moreover, inferring $\mathbf{z}_i$ from $\mathbf{x}$ is an iterative process that may require a large number of iterations, thus, less suitable for real-time tasks. Work on speeding up the ADMM method for CD learning is done in [5]. In [13], a consensus-based optimization framework has been proposed that makes CSC tractable on large-scale datasets. In [14], a thorough performance comparison among different CD learning methods is done as well as proposing a new learning approach. More work on CD learning has been done in [10] and [15]. In [16], the effect of solving in the frequency domain on boundary artifact is studied, offering different types of solutions. The work in [17] shows the potential of CSC for image super-resolution.

4. LEARNED CONVOLUTIONAL SPARSE CODING

In order to have computationally efficient CSC model, we extend the approximate SC model LISTA [1] to the CSC model. We perform training in an end-to-end task driven fashion. Our proposed approach shows competitive performance to classical SC and CSC methods in different tasks but with order of magnitude fewer computations. Our model is trained via stochastic gradient-descent, thus, naturally it can learn the CD over very large datasets without any special adaptations. This of course helps in learning a CD that can better represent the space in which the input signals are sampled from.

4.1. Learning Approximate CSC

Due to the linear properties of convolutions and the fact that CSC model can be thought of as a classical SC model, where the dictionary $\mathbf{D}_{conv}$ is a concatenation of Toepltiz matrices, the CSC model can be viewed as a specific case of classical SC. Thus, the objective in (9) can be formatted to be like (2), by substituting the general dictionary $\mathbf{D}$ with $\mathbf{D}_{conv}$. Obviously, naively reformulating CSC model as matrix multiplication is very inefficient both memory-wise because $\mathbf{D}_{conv} \in \mathbb{R}^{(n_1 n_2) \times (n_1 n_2 m)}$ and computation wise, as each element of $\mathbf{x}$ is computed with $n_1 n_2 m$ multiply and accumulate operations (MACs) versus the convolution formation, where only $s^2 m$ MACs are needed (realistically assuming $s \ll \{n_1, n_2\}$).

Thus, instead of using standard LISTA directly on $\mathbf{D}_{conv}$, we reformulate ISTA to the convolutional case and then propose its LISTA version. ISTA iterations for CSC reads as:

$$\mathbf{z}_{k+1} = S_{\lambda/L}(\mathbf{z}_k + \frac{1}{L}\mathbf{d} \star (\mathbf{x} - \mathbf{d} * \mathbf{z}_k)), \qquad (10)$$

where $\mathbf{d} \in \mathbb{R}^{s \times s \times m}$ is an array of m $s \times s$ filters, $\mathbf{d} \star \mathbf{x} = [\text{flip}(\mathbf{d}_0) * \mathbf{x}, ..., \text{flip}(\mathbf{d}_{m-1}) * \mathbf{x}]$ and $\mathbf{d} * \mathbf{z} = \sum_{i=0}^{m-1} \mathbf{d}_i * \mathbf{z}_i$. The operartion $\text{flip}(\mathbf{d}_i)$ reverses the order of entries in $\mathbf{d}_i$ in both dimensions. Modeling (10) in a similar way to (6) (with some differences) leads to the convolutional LISTA structure:

$$\mathbf{z}_{k+1} = S_\theta(\mathbf{z}_k + \mathbf{w_e} * (\mathbf{x} - \mathbf{w_d} * \mathbf{z}_k)), \qquad (11)$$

where $\mathbf{w_e} \in \mathbb{R}^{s \times s \times c \times m}$, $\mathbf{w_d} \in \mathbb{R}^{s \times s \times m \times c}$ and $\theta \in \mathbb{R}_+^m$ are fully trainable and independent variables and the $\star$ operation has been replaced with $*$ since $\mathbf{w_e}$ is not constrained to be the transpose of $\mathbf{w_d}$. Note that we have added the variable c that takes into account having multiple channels in the original signal (e.g., color channels). In this case, a convolution of a 3D $input$ tensor with a 4D $filter$ tensor results with a 3D $output$ tensor defined as $output[i, j, k] = \sum_{di, dj, q} input[i + di, j + dj, q] \cdot filter[di, dj, q, k]$.

4.2. Learning the CD

When learning a CD, we expect to produce $\hat{x}$ as close as possible to $\mathbf{x}$ given the approximate CSC (ACSC) produced by the model described in (11). Thus, we learn the CD by adding a linear encoder that consists of a filter array $\mathbf{d}$ at the end of the convolutional LISTA iterations. This leads to the following network that integrates both the calculation of the CSC and the application of the dictionary:

$$
\begin{aligned}
&\mathbf{z}_0 = 0, \quad , k = 0 : K - 1 \\
&\mathbf{z}_{k+1} = S_\theta(\mathbf{z}_k + \mathbf{w_e} * (\mathbf{x} - \mathbf{w_d} * \mathbf{z}_k)) \\
&\mathbf{z}_{ACSC} = \mathbf{z}_K \\
&\hat{\mathbf{x}} = \mathbf{d} * \mathbf{z}_{ACSC}
\end{aligned}
\tag{12}
$$

4.3. Task driven convolutional sparse coding

This formulation makes it possible to train a sparse induced auto-encoder, where the encoder learns to produce ACSC and the decoder learns the correct filters to reconstruct the signal from the generated ACSC. The whole model is trained via stochastic gradient decent aiming at minimizing:

$$
\mathcal{L} = dist(\mathbf{x}, \hat{\mathbf{x}}),
\tag{13}
$$

where $\mathbf{x}$ is the target signal and $\hat{\mathbf{x}}$ is the one calculated in (12). We tested different types of distance functions and found (15) to yield best results. Unlike the sparse auto-encoder proposed in [18], where the encoder is a feed-froward network and sparsity is induced via adding sparsity promoting regularization to the loss function, our model is inherently biased to produce sparse encoding of its input due to the special design of its architecture. From a probabilistic point of view, as shown in [19], sparse auto-encoder can be thought of as a generative model that has latent variables with a sparsity prior. Thus, the joint distribution of the model, the output and the CSC is

$$
p_{model}(\mathbf{x}, \mathbf{z}) = p_{model}(\mathbf{z}) p_{model}(\mathbf{x}|\mathbf{z}).
\tag{14}
$$

The soft thresholding operation used in the network encourages $p_{model}(\mathbf{z})$ to be large when $\mathbf{z}$ is sparse. Thus, when training our model we found it sufficient to minimize a reconstruction term representing $-\log(p_{model}(\mathbf{x}|\mathbf{z}))$ without the need to add a sparsity inducing term to the loss.

Image	Proposed ACSC	KSVD
Lena	**32.11**	32.09
House	32.55	**32.7**
Pepper	30.65	**30.89**
Couple	**30.14**	30.05
Fingerprint	27.44	**28.48**
Boat	30.3	**30.37**
Hill	**30.23**	29.5
Man	**30.29**	29.67
Barbara	28.91	**30.57**

Table 1: Denoising PSNR results for $\sigma_n = 20$

	ACSC CPU	ACSC GPU	OMP CPU
runtime [sec]	0.81	0.03	4.21

Table 2: Average run time on the image denoising task. OMP represents the sparse coding step at KSVD denoising [24]

5. EXPERIMENTS

5.1. Learned CSC network parameters

We used our model as specified in (12). We found 3 recurrent steps to be sufficient for our tasks. The filters used in the experiments have the following dimensions: $\mathbf{w_e} \in \mathbb{R}^{7 \times 7 \times 1 \times 64}$, $\mathbf{w_d} \in \mathbb{R}^{7 \times 7 \times 64 \times 1}$, $\theta \in \mathbb{R}_+^{1 \times 64}$, $\mathbf{d} \in \mathbb{R}^{7 \times 7 \times 64 \times 1}$.

As initialization is important for convergence, $\mathbf{w_d}$ and $\mathbf{d}$ are initialized with the same random values and $\mathbf{w_e}$ is initialized with the transposed and flipped filters of $\frac{1}{10}\mathbf{w_d}$. The factor $\frac{1}{10}$ takes into consideration $\frac{1}{L}$ from (3). We initialized the threshold θ to $\frac{1}{10}$, thus implicitly initializing λ to 1. We tested different types of reconstruction loss including standard ℓ_1 and ℓ_2 losses. We found the loss function proposed in [20] to retrieves the best image quality,

$$
\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \alpha(1 - ms\_ssim(\mathbf{x}, \hat{\mathbf{x}})) + (1 - \alpha)||\mathbf{x} - \hat{\mathbf{x}}||_1, \tag{15}
$$

where ms_ssim is the multiscale SSIM loss [21]. Model is trained with $\alpha = 0.8$ and the Adam optimizer [22] using its default parameters [20]. We use PASCAL VOC dataset [23] due to its large amount of high quality images. Images are normalized to the range [0 ,1] before being fed to the model.

5.2. Image denoiseing

To test our model for image denoising we have added random noise to a given original image $\mathbf{x}$ producing $\mathbf{y} = \mathbf{x} + \epsilon$, where $\sigma_n = 20$ and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Table 1 and Fig. 1 show that the learned CD generalizes well and has a competitive recovery to KSVD denoising both qualitatively and quantitatively [24].

We show the atoms of the learned CD in Fig. 2. The learned CD does not have any image specific atoms but rather a mixture of high pass DCT like and Gabor like filters. Table 2 presents the average run time per image. We used KSVD's publicly available code [25]. Our model is faster by an order of magnitude on a CPU and by two orders on a GPU.

Fig. 1: Denoising of *Lena*. From left to right: original, noisy, ACSC denoising, KSVD denoising.
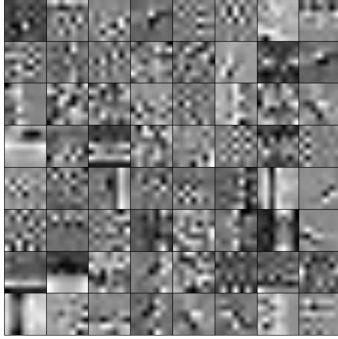
Fig. 2: Convolutional dictionary learned on denoising task containing 64 7×7 filters with gabor and high-pass structure.

Image	Heide et al. [8]	Bristow et al. [9]	Proposed ACSC
1	28.76	28.36	**28.84**
2	31.54	30.52	**31.95**
3	**30.59**	30.22	30.48
4	27.41	27.45	**27.64**
5	33.65	32.91	**33.81**
6	33.03	32.21	**33.27**
7	**28.69**	28.25	28.33
8	**30.39**	30.04	30.23
9	28.07	27.79	**28.10**
10	31.59	30.57	**31.65**
11	**29.77**	29.09	29.60

Table 3: Inpainting PSNR for test-set from [8].

	ACSC CPU	ACSC GPU	[8] CPU
runtime [sec]	0.6	0.023	163

Table 4: Average runtime on image inpainting.

Fig. 3: Inpainting of test image #10 from test-set in [8] after local contrast normalization. Left: ACSC. Right: Heide et al.

5.3. Image inpainting

We further test our model on the inpainting problem, in which $\mathbf{y} = \mathbf{M} \odot \mathbf{x}$, where $\odot$ is as an element wise multiplication operator and $\mathbf{M}$ is a binary mask such that $\mathbf{y}$ contains only part of the pixels in $\mathbf{x}$. Rewriting (8) while taking $\mathbf{M}$ into consideration, we have

$$\mathbf{y} = \mathbf{M} \odot \mathbf{D} * \mathbf{z}. \tag{16}$$

Thus, (5) becomes,

$$\mathbf{z}_{k+1} = S_{\lambda/L}(\mathbf{z}_k + \frac{1}{L}\mathbf{d} \star (\mathbf{M} \odot (\mathbf{x} - \mathbf{d} * \mathbf{z}_k))). \tag{17}$$

The ACSC version of (17) is given by

$$\mathbf{z}_{k+1} = S_\theta(\mathbf{z}_k + \mathbf{w_e} * (\mathbf{M} \odot (\mathbf{x} - \mathbf{w_d} * \mathbf{z}_k))). \tag{18}$$

In our experiment $M(i,j) \sim Ber(0.5)$, thus, we randomly sample half of the input pixels. The objective is to reconstruct $\mathbf{x}$. We took the pre-trained ACSC network of the denoising task, plugged in it $\mathbf{M}$ to be the form of (18), and optimized it for the inpainting task over the PASCAL VOC [23] dataset. We compare our inpainting results to [8] over the same test images used there. The image numbering convention is consistent to [8]. All test images are preprocessed with local contrast as in [8]. Table. 3 shows that our model produces competitive results to the ones of [8] and [9] . Notice in Fig. 3 that our model is able to reconstruct the finer details of the road that are lost in [8]. The main advantage of the proposed approach as can be seen in Table 4 is its significant speed-up in running time (by more than three orders of magnitude). [1]

6. CONCLUSION

Approximate convolutional sparse coding as proposed in this work is a powerful tool. It combines both the computational capabilities and approximation power of convolutional neural networks and the strong theory of sparse coding. We demonstrated the efficiency of this strategy both in the tasks of image denoising and inpainting.

---

[1] We compare only to the runtime of [8] as no code was available for [9].

7. REFERENCES

[1] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 399–406.

[2] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on pure and applied mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[3] P. Sprechmann, A.M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1821–1833, 2015.

[4] J. Rolfe and Y. LeCun, "Discriminative recurrent sparse auto-encoders," *ICLR*, 2013.

[5] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 370–378.

[6] R. Giryes, Y. Eldar, A.M. Bronstein, and G. Sapiro, "Tradeoffs between convergence speed and reconstruction accuracy in inverse problems," *arXiv preprint arXiv:1605.09232*, 2016.

[7] T. Moreau and J. Bruna, "Adaptive acceleration of sparse coding via matrix factorization," *ICLR*, 2017.

[8] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5135–5143.

[9] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 391–398.

[10] V. Papyan, Y. Romano, J. Sulam, and M. Elad, "Convolutional dictionary learning via local processing," *ICCV*, 2017.

[11] H. Bristow and S. Lucey, "Optimization methods for convolutional sparse coding," *arXiv preprint arXiv:1406.2407*, 2014.

[12] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus, "Deconvolutional networks," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2528–2535.

[13] B. Choudhury, R. Swanson, F. Heide, G. Wetzstein, and W. Heidrich, "Consensus convolutional sparse coding," 2017.

[14] C. Garcia-Cardona and B. Wohlberg, "Convolutional dictionary learning," *arXiv preprint arXiv:1709.02893*, 2017.

[15] B. Wohlberg, "Efficient convolutional sparse coding," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7173–7177.

[16] B. Wohlberg, "Boundary handling for convolutional sparse representations," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1833–1837.

[17] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[18] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[19] I. Goodfellow, Y. Bengio, and Courville A., *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.

[20] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.

[21] Zhou Wang, Eero P Simoncelli, and Alan C Bovik, "Multiscale structural similarity for image quality assessment," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*. Ieee, 2003, vol. 2, pp. 1398–1402.

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[24] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[25] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit," *Cs Technion*, vol. 40, no. 8, pp. 1–15, 2008.