Generative Models for Low-Dimensional Video Representation and Compressive Sensing

Rakib Hyder and M. Salman Asif
Department of Electrical and Computer Engineering, University of California, Riverside
rhyde001@ucr.edu, sasif@ece.ucr.edu

***Abstract*—Generative priors have become highly effective in solving inverse problems including denoising, inpainting, and reconstruction from few and noisy measurements. With a generative model we can represent an image with a much lower dimensional latent codes. In the context of compressive sensing, if the unknown image belongs to the range of a pretrained generative network, then we can recover the image by estimating the underlying compact latent code from the available measurements. However, recent studies revealed that even untrained deep neural networks can work as a prior for recovering natural images. These approaches update the network weights keeping latent codes fixed to reconstruct the target image from the given measurements. In this paper, we optimize over network weights and latent codes to use untrained generative network as prior for video compressive sensing problem. We show that by optimizing over latent code, we can additionally get concise representation of the frames which retain the structural similarity of the video frames. We also apply low-rank constraint on the latent codes to represent the video sequences in even lower dimensional latent space. We empirically show that our proposed methods provide better or comparable accuracy and low computational complexity compared to the existing methods.**

I. Introduction

A. Motivation and Related Works

Compressive sensing refers to a broad class of problems in which we aim to recover a signal from a small number of measurements [1]–[3]. Suppose we are given a sequence of measurements for $t = 1, \dots, T$ as

$$y_t = A_t x_t + e_t, \tag{1}$$

where x_t denotes the t^{th} frame in the unknown video sequence, y_t denotes its observed measurements, A_t denotes the respective measurement operator, and e_t denotes noise or error in the measurements. Our goal is to recover the video sequence (x_t) from the available measurements (y_t). The recovery problem becomes especially challenging as the number of measurements (in y_t) becomes very small compared to the number of unknowns (in x_t).

Classical signal priors exploit sparse and low-rank structures in images and videos for their reconstruction [4]–[16]. However, the natural images exhibits far richer nonlinear structures than sparsity alone. We focus on a newly emerging generative priors that learn a function that maps vectors drawn from a certain distribution in a low-dimensional space into images in a high-dimensional space.

The generative model and optimization problems we use are inspired by recent work on using generative models for compressive sensing in [17]–[23]. Compressive sensing using generative models was introduced in [17], which used a trained deep generative network as a prior for image reconstruction from compressive measurements. Afterwards deep image prior (DIP) used an untrained convolutional generative model as a prior for solving inverse problems such as inpainting and denoising because of their tendency to generate natural images [22]; the reconstruction problem involves optimization of generator network parameters. Inspired by these observations, a number of methods have been proposed for solving compressive sensing problem by optimizing generator network weights while keeping the latent code fixed at a random value [19], [20]. Both DIP [22] and deep decoder [20] update the model weights to generate a given image; therefore, the generator can reconstruct wide range of images. One key difference between the two approaches is that the network used in DIP is highly overparameterized, while the one used in deep decoder is underparameterized.

We observed two main limitations in the DIP and deep decoder-based video recovery that we seek to address in this paper. (1) The latent codes in DIP and deep decoder methods are initialized at random and stay fixed throughout the recovery process. Therefore, we cannot infer the structural similarities in the images from the structural similarities in the latent codes. (2) Both of these methods train one network per image. A naive approach to train one network per frame in a video will be computationally prohibitive, and if we train a single network to generate the entire video sequence, then their performance degrades.

Therefore, we propose joint optimization over network weights γ and the latent codes z_t to reconstruct video sequence. Thus we learn a single generator and a set of latent codes to represent a video sequence. We observe that when we optimize over latent code alongside network weights, the temporal similarity in the video frames is reflected in the latent code representation. To exploit similarities among the frames in a video sequence, we also include low-rank constraints on the latent codes. An illustration of different types of representations we use in this paper are shown in Figure 1.

B. Our Contribution

In this paper, we reconstruct a video sequence from the compressive measurements in (1) by jointly optimizing over the latent codes z_t and the network parameters γ. Since the frames in a video sequence exhibit rich redundancies in their representation, we impose a low-rank constraint on the latent codes to represent the video sequence with a more compact representation of the latent codes.

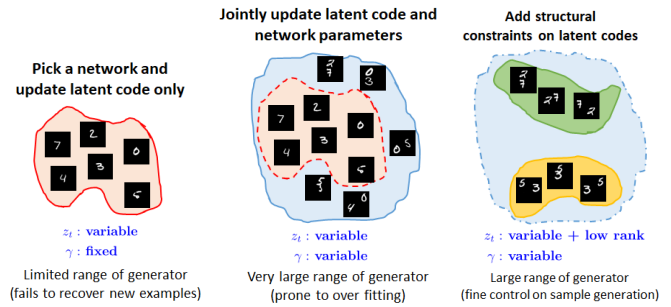


Fig. 1: An illustration of different generative priors discussed in the paper: (a) Optimizing latent codes can only reconstruct images in the range of the generative network. (b) Jointly optimizing latent code and network weights enables recovery of a larger range of images. (c) Low-rank and similarity constraints on latent code further regularize the problem and potentially explain other structures in data.

The key contributions of this paper are as follows.

- We demonstrate that joint optimization allows us to learn a single generator network for an entire video sequence and corresponding latent codes simultaneously. We demonstrate that this approach has lower computational complexity and requires less number of parameters to reliably generate the entire video sequence. Furthermore, joint optimization retains the similarity structure of the video frames in their latent representation which leaves further scope for different tasks which involves latent space manipulation.
- Consecutive frames in a video sequence share lot of similarities. To encode similarities among the reconstructed frames, we introduce low-rank constraints on the generator latent codes. This enables us to represent a video sequence with even smaller number of parameters in the latent space. We show that, in some cases, the low-rank structure on the latent codes also provides a nice low-dimensional manifold.

II. Proposed Method

A. Joint Latent Codes and Generator Optimization

For a single image reconstruction, deep image prior solve the following optimization to obtain optimal $\hat{\gamma}$,

$$\arg\min_{\gamma} \ \|y - AG_\gamma(z)\|_2^2 \tag{2}$$

In this optimization, z is initialized randomly and kept unaltered. To jointly optimize the latent codes and generator parameters for a video sequence, we use the similar formulation as in (2) but optimize it over the z_t and γ. The resulting optimization problem can be written as

$$\underset{z_1,\ldots,z_T;\gamma}{\text{minimize}} \ \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2. \tag{3}$$

The reconstructed video sequence can be generated using the estimated latent codes $(\hat{z}_1, \ldots, \hat{z}_T)$ and generator weights $(\hat{\gamma})$ as $\hat{x}_t = G_{\hat{\gamma}}(\hat{z}_t)$.

We initialize latent codes with samples drawn from a Gaussian distribution and normalize them to have unit norm. We initialize γ with random weights using the initialization scheme in [24]. Initilizing the generator with a pretrained set of weights can potentially serve as a good initialization and lead to good and faster convergence. We tested both variants, but observed little difference in performance; therefore, we use random initialization of parameters in this paper. Each iteration of joint optimization consists of two steps: 1) latent code optimization and 2) network parameter optimization. After every gradient descent update of the latent codes, z_t, we update the model parameters with stochastic gradient descent. In all of our experiments with joint optimization, we learned a single set of network weights for the entire sequence. We note that it is possible to divide a longer video sequences into small segments and learn different sets of network weights for each of them. At the end of our reconstruction process, we have a single set of trained weights $\hat{\gamma}$, reconstructed frames $\hat{x}_t$ and their corresponding optimal latent codes $\hat{z}_t$.

B. Low Rank Constraint

As we optimize over the latent codes and the network weights in joint optimization, the latent codes capture the temporal similarity of the video frames. To further exploit the redundancies in a video sequence, we assume that the variation in the sequence of images are localized and the latent codes sequence can be represented in a low-dimensional space compared to their ambient dimension. Let us define a matrix Z with all the latent codes as

$$Z = [z_1 \ z_2 \ \ldots \ z_T],$$

where z_t is the latent code corresponding to t^{th} image of the sequence. To impose a low-rank constraint, we solve the following constrained optimization:

$$\begin{aligned} \underset{z_1,\ldots,z_T;\gamma}{\text{minimize}} \ & \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2 \\ \text{s.t.} \ & \ \text{rank}(Z) = r. \end{aligned} \tag{4}$$

We solve (4) using a projected gradient descent method in which we project the latent code estimates after every iteration to a manifold of rank-r matrices. To do that, we compute Z matrix and its rank-r approximation using principal component analysis (PCA) or singular value decomposition (SVD).

In this manner, we can express each of the latent codes in terms of r orthogonal basis vectors vectors $u_1, \ldots, u_r$ as

$$z_i = \sum_{j=1}^{r} \alpha_{ij} u_j \tag{5}$$

where α_{ij} is the weight of the corresponding basis vector. We can represent a video sequence with T frames with r orthogonal codes, and the lowrank representation of latent codes requires $r \times k + r \times T$ parameters compared to $T \times k$. This offers $r(\frac{1}{T} + \frac{1}{k})$ times compression to our latent code representation. As we observe later, we use $r = 4$ for $k = 256$ and $T = 32$ which gives us compression of 0.14 in latent code representation.

III. Experimental Results

A. Setup and Performance Comparison

In this paper we report the results for one synthetic sequence which we refer to as '*Rotating MNIST*'. In this sequence, we resize one MNIST digit to 64×64 and rotate by 2° per frame for a total of 32 frames. We experiment on different real video

Algorithm 1 Generative Models for Low Rank Representation and Recovery of Videos

Input: Measurements y_t, measurement matrices A_t, A generator structure $G_\gamma(\cdot)$

Initialize the latent codes z_t and generator weights γ randomly and normalize z_t with its 2-norm.

repeat

Compute gradients w.r.t. z_t via backpropagation.

Update latent code matrix $Z = [z_1 \ \cdots \ z_T]$.

Truncate Z to a rank-r matrix via SVD or PCA.

Compute gradients w.r.t. γ via backpropagation.

Update network weights γ.

until convergence or maximum epochs

Output: Latent codes: $z_1, \ldots, z_T$ and network weights: γ

sequences from publicly available KTH human action video dataset [25] and UCF101 dataset [26]. In Table I, we report our results for '*Handclapping*', '*Handwaving*' and '*Walking*' video sequences from KTH dataset; '*Archery*', '*Apply Eye Makeup*' and '*Band Marching*' video sequences from UCF101 dataset. We centered and resized every frame in KTH videos to 64×64 and UCF101 videos to 256×256 pixels.

We used the well-known DCGAN architecture [27] for our generators, except that we do not use any batch-normalization layer. The latent code dimensions for grayscale 64×64, RGB 64×64 and RGB 256×256 video sequences are 64, 256 and 512 respectively. We use Adam optimizer for generator weights optimization and SGD for latent code optimization. Unless otherwise mentioned, we use rank=4 constraint as low rank constraint because we empirically found that we need a least rank=4 for a video sequence with 32 frames to get comparable performance.

We show comparison with classical total variation minimization based TVAL3D (3D extension of TVAL3 [28]) algorithm and state-of-the-art untrained generative prior based deep decoder [20] on denoising, inpainting, and compressive sensing tasks. We use two different deep decoder settings: underparameterized deep decoder (UP deep decoder) and overparameterized deep decoder (OP deepdecoder). Although the authors suggested deep decoder to be UP, we report the results for OP deep decoder as well because it shows better performance and its hyperparameters are tuned by the authors of deep decoder.

Other then denoising and inpainting, we performed compressive random projection experiments where we used separable measurements, $Y = P^T X P$, where X, Y are reshaped versions of x, y as 2D matrices, P is a random projection matrix.

We report the results for denoising experiment at 20 dB SNR noise, inpainting experiment for 80% missing pixels and compressive sensing experiments for 20% available measurements in Table I. From the results, we can observe that joint optimization with/without low-rank constraint outperform TVAL3D algorithm and UP deep decoder. It performs at par with OP deep decoder. In Figure 2, we show reconstruction performance for denoising, inpainting and compressive sensing at different measurement rate or noise level for '*Handwaving*' video sequence. We can get the similar observation from these curves as well. We report some reconstruction results for '*Handwaving*' sequence in Figure 2. From the reconstructions, we can say that joint optimization performs at par with the comparing algorithms. It especially performs well in reconstructing details from masked frames.

TABLE I: Reconstruction performance measured in terms of PSNR for different compressive sensing problems. We show comparison with TVAL3D (3D extension of TVAL3 [28]) and deep decoder [20]. The results are average over five experiments with different random measurement matrices (or noise in the case of denoising).

Video Sequence	Rotating MNIST	Handclapping	Handwaving	Walking	Apply Eye Makeup	Archery	Band Marching
Denoising for additive Gaussian noise of 20dB SNR							
TVAL3D	35.8	32.2	30.4	30.5	34.5	31.5	30.6
UP deep decoder	28.9	28.4	25.6	28.3	28.1	29.6	28.1
OP deep decoder	36.6	31.1	30	31	34.4	**33**	31.6
Joint optimization	**36.9**	**32.7**	30.7	**31.2**	36.1	32.1	31.3
Joint opt+lowrank	36.8	32.3	**30.8**	30.7	**36.4**	32	**31.7**
Inpainting with 80% pixels randomly missing							
TVAL3D	21.1	29.2	23.4	24.5	28.2	27.1	24.8
UP deep decoder	25.5	26.5	23.3	26.3	27.2	29	23.3
OP deep decoder	30.1	30.2	26.7	27.9	32.4	**32.5**	26.2
Joint optimization	29.3	**34.9**	**28.1**	**28.9**	35.8	32	26.8
Joint opt+lowrank	**29.5**	34.3	27.3	27.8	**36.6**	30.4	**27.6**
Spatial compressive sensing with compression rate = 0.2							
TVAL3D	29.8	32.1	28.9	28	33.9	28.4	27.8
UP deep decoder	30	27	24.9	26.7	26.2	27.6	22.5
OP deep decoder	35.2	32.9	**30.6**	29	33.1	**31.2**	27.4
Joint optimization	35.3	**35.6**	29.7	28.9	**36**	29.3	27.8
Joint opt+lowrank	**35.4**	34.7	29	**29.1**	35.9	28.8	**29.1**

B. Computational Complexity

The computational complexity of our proposed methods vary with the choice of the generator structure. We have chosen DCGAN generator structure for our experiments. We calculate memory requirement for gradient descent using `torchsummary` package [29]. For a single 64×64 RGB image, memory requirement for UP deep decoder, OP deep decoder and joint optimization is 2.75 MB, 66.48 MB and 2.06 MB respectively. For a single 256×256 RGB image, memory requirement for UP deep decoder, OP deep decoder and joint optimization is 44.03 MB, 1239.75 MB and 10.88 MB respectively. For a RGB video seqence with 32 frames, UP deep decoder will require $11,304 \times 32$ (361,728) parameters while OP deep decoder will have $397,056 \times 32$ (12.7M) parameters. On the other hand we need 4,852,736 and 6,988,544 network parameters to represent RGB 64×64 and 256×256 video sequences, respectively in joint optimization method with DCGAN generator.

Because of the huge memory requirement, OP deep decoder is not suitable for optimization over entire video sequence whereas the low capacity hinders UP deep decoder from generating entire video sequence.

C. Similarity Structure in the Latent Codes

To investigate the similarity structure in the latent codes obtained by joint optimization, we performed another experiment in which we concatenated 16 frames from each of the

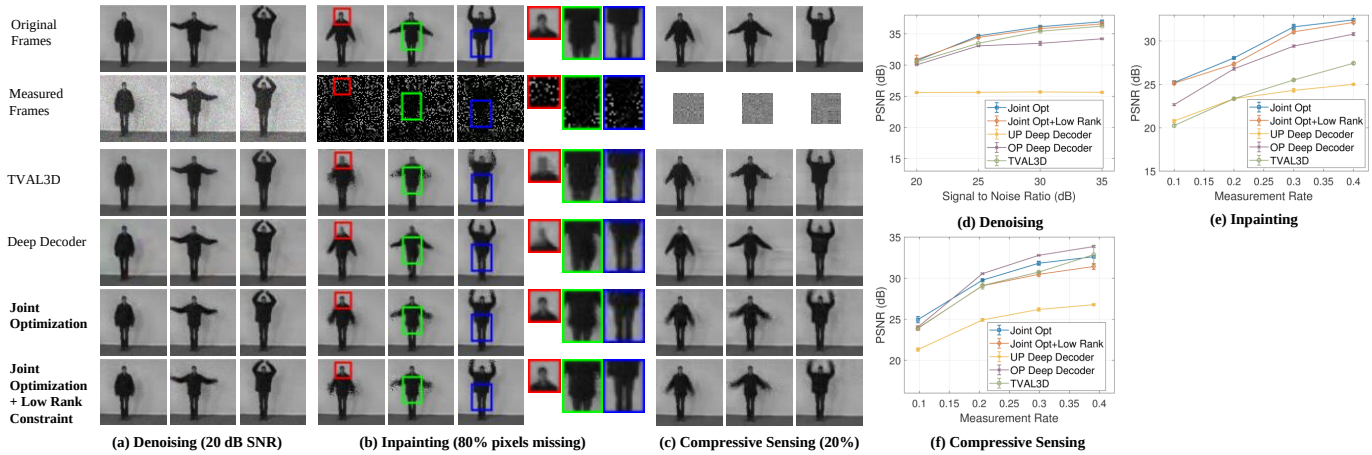
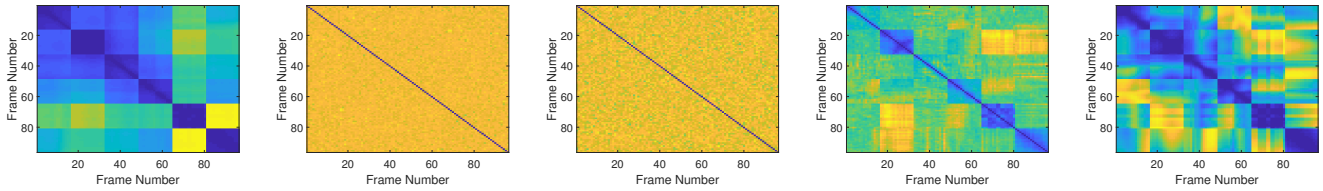


Fig. 2: (a)-(c) are some reconstructions for Handwaving sequence for (a) denoising with 20 dB SNR noise level, (b) inpainting from masked frames with 80% missing pixels, (c) compressive sensing from 20% measurements. The deep decoder reconstruction here correspond to OP deep decoder structure because quantitative reconstruction results for UP deep decoder are significantly worse. Joint optimization with/without low rank constraint performs at par with other comparing algorithms. (d)-(f) are reconstruction quality curves for (d) denoising (e) inpainting (f) compressive sensing for Handwaving sequence. We can observe that joint optimization performs consistently for different measurement rates or noise level.



(a) Original Frames **(b)** Compressive Frames **(c)** Latent Matrix for Fixed Random Latent Codes **(d)** Latent Matrix of Joint Optimization **(e)** Latent Matrix of Joint Optimization + Low Rank Constraint

Fig. 3: Pairwise cosine similarity between frames, measurements or latent codes for extended mixed video sequence where 16 frames of 6 different video sequences (Handwaving, Handclapping, Walking, Archery, Apply Eye Makeup ,Band Marching in order) are concatenated in the temporal dimension. Blue indicates highest similarity whereas yellow indicates lowest similarity. We observe that adding low rank constraint further bolster the similarity observed in the the frames of same video sequences found by joint optimization.

six different video sequences ('*Handwaving*', '*Handclapping*', '*Walking*', '*Archery*', '*Apply Eye Makeup*', and '*Band Marching*', in the same order) to create a new sequence with 96 frames. We performed compressive sensing experiment on this video sequence with 20% measurements.

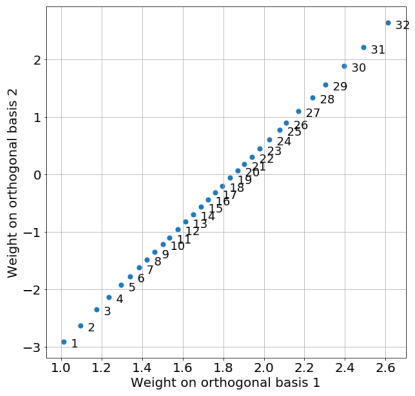


Fig. 4: Linear representation of latent space of Rotating MNIST sequence using rank=2 constraint for image inpainting task with 80% missing pixels. The annotations for each point denotes the latent code corresponding to t^{th} frame.

The cosine similarity matrices for the video frames, compressive measurements, latent codes for with fixed latent codes (random), latent codes for joint optimization, and latent codes for joint optimization with low-rank are presented in Figure 3(a)–(e). We can distinguish the video sequences from the pairwise similarity matrices of the latent codes we estimate with joint optimization. We also observe that the low-rank constraint improves the similarity matrix.

We mentioned that using low rank constraint we can represent the video sequences in much lower dimensional space. If the generator function is continuous it makes sense that the latent space representation of a video sequence will retain their sequential structure in some low dimensional representation. We demonstrate one such example using rank=2 constraint on the latent codes while reconstructing '*Rotating MNIST*' sequence from its masked version with 80% pixels missing. As we are enforcing, rank=2 constraint taking mean and first principal component, the latent codes should fall on line. In Figure 4, we represent the latent codes in a 2D plane using 2 orthogonal basis vectors. t^{th} point in Figure 4, represent latent code of t^{th} frame. We can observe that latent codes are maintaining sequence in their 2D dimensional representation. For complex motions, it might take higher dimensional representation to observe such sequential pattern.

REFERENCES

[1] E. J. Candes, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, 2011.

[2] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[3] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE transactions on information theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[4] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[5] R. Baraniuk and P. Steeghs, "Compressive radar imaging," in *Radar Conference, 2007 IEEE*. IEEE, 2007, pp. 128–133.

[6] F. Yang, H. Jiang, Z. Shen, W. Deng, and D. Metaxas, "Adaptive low rank and sparse decomposition of video using compressive sensing," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*. IEEE, 2013, pp. 1016–1020.

[7] J. V. Shi, A. C. Sankaranarayanan, C. Studer, and R. G. Baraniuk, "Video compressive sensing for dynamic mri," *BMC neuroscience*, vol. 13, no. 1, p. P183, 2012.

[8] C. Zhao, S. Ma, J. Zhang, R. Xiong, and W. Gao, "Video compressive sensing reconstruction via reweighted residual sparsity," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 6, pp. 1182–1195, 2017.

[9] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The jpeg2000 still image coding system: an overview," *IEEE transactions on consumer electronics*, vol. 46, no. 4, pp. 1103–1127, 2000.

[10] A. Puri and A. Eleftheriadis, "Mpeg-4: An object-based multimedia coding standard supporting mobile applications," *Mobile Networks and Applications*, vol. 3, no. 1, pp. 5–32, 1998.

[11] G. J. Sullivan, P. N. Topiwala, and A. Luthra, "The h. 264/avc advanced video coding standard: Overview and introduction to the fidelity range extensions," in *Applications of Digital Image Processing XXVII*, vol. 5558. International Society for Optics and Photonics, 2004, pp. 454–475.

[12] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[13] M. S. Asif, F. Fernandes, and J. Romberg, "Low-complexity video compression and compressive sensing," in *2013 Asilomar Conference on Signals, Systems and Computers*. IEEE, 2013, pp. 579–583.

[14] Y. Li, W. Dai, J. Zou, H. Xiong, and Y. F. Zheng, "Structured sparse representation with union of data-driven linear and multilinear subspaces

[23] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, "Optimizing the latent space of generative networks," in *Proc. Int. Conf. Machine Learning*, 2018.

model for compressive video sampling," *IEEE Transactions on Signal Processing*, vol. 65, no. 19, pp. 5062–5077, 2017.

[15] W. Dai, Y. Li, J. Zou, H. Xiong, and Y. F. Zheng, "Fully decomposable compressive sampling with joint optimization for multidimensional sparse representation," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 603–616, 2017.

[16] J. F. Mota, N. Deligiannis, A. C. Sankaranarayanan, V. Cevher, and M. R. Rodrigues, "Adaptive-rate reconstruction of time-varying signals with application in compressive foreground extraction," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3651–3666, 2016.

[17] A. Bora, A. Jalal, E. Price, and A. Dimakis, "Compressed sensing using generative models," *Proc. Int. Conf. Machine Learning*, 2017.

[18] A. C. Gilbert, Y. Zhang, K. Lee, Y. Zhang, and H. Lee, "Towards understanding the invertibility of convolutional neural networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1703–1710.

[19] D. Van Veen, A. Jalal, E. Price, S. Vishwanath, and A. G. Dimakis, "Compressed sensing with deep image prior and learned regularization," *arXiv preprint arXiv:1806.06438*, 2018.

[20] R. Heckel and P. Hand, "Deep decoder: Concise image representations from untrained non-convolutional networks," *Proc. Int. Conf. Learning Representations (ICLR)*, 2018.

[21] V. Shah and C. Hegde, "Solving linear inverse problems using gan priors: An algorithm with provable guarantees," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, 2018.

[22] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, 2018, pp. 9446–9454.

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

[25] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3. IEEE, 2004, pp. 32–36.

[26] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[27] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *Proc. Int. Conf. Learning Representations (ICLR)*, 2016.

[28] C. Li, W. Yin, H. Jiang, and Y. Zhang, "An efficient augmented lagrangian method with applications to total variation minimization," *Computational Optimization and Applications*, vol. 56, no. 3, pp. 507–530, 2013.

[29] "Keras style model.summary() in pytorch," available at github: https://github.com/sksq96/pytorch-summary, Accessed: 7 August, 2019.