Title

Generative Models for Low-Dimensional Video Representation and Reconstruction

Authors

Hyder, Rakib
Asif, M Salman

Publication Date

2020

DOI

10.1109/tsp.2020.2977256

Peer reviewed

Generative Models for Low-Rank Video Representation and Reconstruction

Rakib Hyder and M. Salman Asif

Department of Electrical and Computer Engineering, University of California, Riverside

Abstract

Finding compact representation of videos is an essential component in almost every problem related to video processing or understanding. In this paper, we propose a generative model to learn compact latent codes that can efficiently represent and reconstruct a video sequence from its missing or under-sampled measurements. We use a generative network that is trained to map a compact code into an image. We first demonstrate that if a video sequence belongs to the range of the pretrained generative network, then we can recover it by estimating the underlying compact latent codes. Then we demonstrate that even if the video sequence does not belong to the range of a pretrained network, we can still recover the true video sequence by jointly updating the latent codes and the weights of the generative network. To avoid overfitting in our model, we regularize the recovery problem by imposing low-rank and similarity constraints on the latent codes of the neighboring frames in the video sequence. We use our methods to recover a variety of videos from compressive measurements at different compression rates. We also demonstrate that we can generate missing frames in a video sequence by interpolating the latent codes of the observed frames in the low-dimensional space.

1 Introduction

Deep generative networks, such as autoencoders, generative adversarial networks (GANs), and variational autoencoders (VAEs), are now commonly used in almost every machine learning and computer vision task [12, 24, 16, 35]. One key idea in these generative networks is that they can learn to transform a low-dimensional feature vector (or latent code) into realistic images and videos. The *range* of the generated images is expected to be close to the true underlying distribution of training images. Once these networks are properly trained (which remains a nontrivial task), they can generate remarkable images in the trained categories of natural scenes.

In this paper, we propose to use a deep generative model for compact representation and reconstruction of videos from a small number of linear measurements. We assume that a generative network trained on some class of images is available, which we represent as

$$x = G_\gamma(z) \equiv g_{\gamma_L} \circ g_{\gamma_{L-1}} \circ \cdots \circ g_{\gamma_1}(z). \tag{1}$$

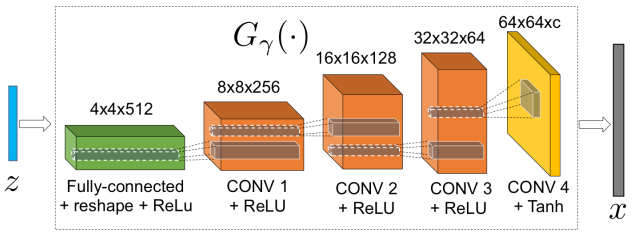


Figure 1: Generative model: $x = G_\gamma(z)$ maps a vector $z \in \mathbb{R}^k$ into an image $x \in \mathbb{R}^n$. The figure shows a DCGAN architecture that we used in our experiments with one fully connected and four convolutional layers.

$G_\gamma(z)$ denotes the overall function for the deep network with L layers that maps a low-dimensional (latent) code $z \in \mathbb{R}^k$ into an image $x \in \mathbb{R}^n$ and $\gamma = \{\gamma_1, \ldots, \gamma_L\}$ represents all the weight parameters of the deep network. $G_\gamma(\cdot)$ as given in (1) can be viewed as a cascade of L functions g_{γ_l} for $l = 1, \ldots, L$, each of which represents a mapping between input and output of respective layer. An illustration of such a generator with $L = 5$ is shown in Figure 1. Suppose we are given a sequence of measurements for $t = 1, \ldots, T$ as

$$y_t = A_t x_t + e_t, \tag{2}$$

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). To ensure quality reconstruction in such settings, we need a compact (low-dimensional) representation of the unknown signal. Thus, we use the given generative model to represent the video sequence as $x_t = G_\gamma(z_t)$ and the observed measurements as $y_t = A_t G_\gamma(z_t)$.

We first demonstrate that if a video sequence (x_t) belongs to the range of the network $G_\gamma(z_t)$, then we can reconstruct it by optimizing directly over the latent code z_t. Then we demonstrate that even if a video sequence lies outside the range of the given network $G_\gamma(z_t)$, we can still reconstruct it by jointly optimizing over network weights γ and the latent codes z_t. To exploit similarities among the frames in a video sequence, we also include low-rank and similarity constraints on the latent codes. We note that the pretrained network we used in our experiments is highly overparameterized; therefore, low-rank and similarity constraints help in regularizing the network and finding good solution presumably near the initial weights.

1.1 Motivation and Related Work

Video signals have natural redundancies along spatial and temporal dimensions that can be exploited to learn their *compact* representations. Such compact representations can then be used for compression, denoising, restoration, and other processing/transmission tasks. Historically, video representation schemes have relied on hand-crafted blocks that include motion estimation/compensation and sparsifying transforms such as discrete cosine transform (DCT) and wavelets [36, 8, 32, 22]. Recent progress in data-driven representation methods offers new opportunities to develop improved schemes for compact representation of videos [21, 25, 19].

Compressive sensing refers to a broad class of problems in which we aim to recover a signal from a small number of measurements [5, 10, 6]. The canonical compressive sensing problem in (2) is inherently underdetermined, and we need to use some prior knowledge about the signal structure. Classical signal priors exploit sparse and low-rank structures in images and videos for their reconstruction [11, 1, 37, 28, 38].

Deep generative models offer a new framework for compact representation of images and videos. A generative model can be viewed as a function that maps a given input (or latent) code into an image. For compact representation of images, we seek a generative model that can generate a variety of images with high fidelity using a very low-dimensional latent code. Recently, a number of generative models have been proposed to learn latent representation of an image with respect to a generator [20, 39, 9]. The learning process usually involves gradient decent to estimate the best representation of the latent code, where the gradients with respect to the latent code representation are backpropagated to the pixel space [3].

In recent year, generative networks have been extensively used for learning good representations for images and videos. Generative adversarial networks (GANs) and variational autoencoders (VAEs) [12, 17, 15, 2] learn a function that maps vectors drawn from a certain distribution in a low-dimensional space into images in a high-dimensional space. An attractive feature of VAEs [17] and GANs [12] is their ability to

transform feature vectors to generate a variety of images from a different set of desired distributions. Our technical approach bears some similarities with recent work on image generation and manipulation via conditional GANs and VAEs [7, 13, 29]. For example, we can create new images with same content but different articulations by changing the input latent codes [7, 23]. In [3], the authors presented a framework for jointly optimizing latent code and network parameters while training a standalone generator network. Furthermore, linear arithmetic operations in the latent space of generators can generate to meaningful image transformations. In our paper, we will apply similar principles to generate different frames in a video sequence while jointly optimizing latent codes and generator parameters but ensuring that latent codes belong to a small subspace (even a line as we show in Figure 6).

In this paper, we use a generative model as a prior for video signal representation and reconstruction. Our generative model and optimization is inspired by recent work on using generative models for compressive sensing in [4, 34, 14, 27, 33]. Recently, [4] showed that a trained deep generative network can be used as a prior for image reconstruction from compressive measurements; the reconstruction problem involves optimization over the latent code of the generator. In a related work, [33] observed that an untrained convolutional generative model can also be used as a prior for solving inverse problems such as inpainting and denoising because of their tendency to generate natural images; the reconstruction problem involves optimization of generator network weights. 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 [14, 34]. As they are allowing generator parameters to change, the generator can reconstruct wide range of images. However, as the latent codes are initialized randomly and stay the same, we cannot find a representative latent codes for images.

In our proposed method, we use the generative model in (1) to find compact representation of videos in the form of z_t. To reconstruct a video sequence from the compressive measurements in (2), we either optimize over the latent codes z_t or or optimize over the network weights γ and z_t in a joint manner. Since the frames in a video sequence exhibit rich redundancies in their representation. We hypothesize that if the generator function is continuous, then the similarity of the frames would translate into the similarity in their corresponding latent codes. Based on this hypothesis, we impose similarity and low-rank constraints on the latent codes to represent the video sequence with an even more compact representation of the latent codes. An illustration of the differences between the types of representations is shown in Figure 2.

1.2 Main Contributions

In this paper, we propose to use a low-rank generative prior for compact representation of a video sequence, which we then use to solve some video compressive sensing problems. The key contributions of this paper are as follows.

- We first demonstrate that we can learn a compact representation of a video sequence in the form of low-rank latent codes for a deep generative network similar to the one depicted in Figure 1.
- Consecutive frames in a video sequence share lots of similarities. To encode similarities among the reconstructed frames, we introduce low-rank and similarity constraints on the generator latent codes. This enables us to represent a video sequence with a very small number of parameters in the latent codes and reconstruct them from a very small number of measurements.
- Latent code optimization can only reconstruct a video sequence that belong to its range. We demonstrate that by jointly optimizing the latent codes with the network weights, we can expand the range of the

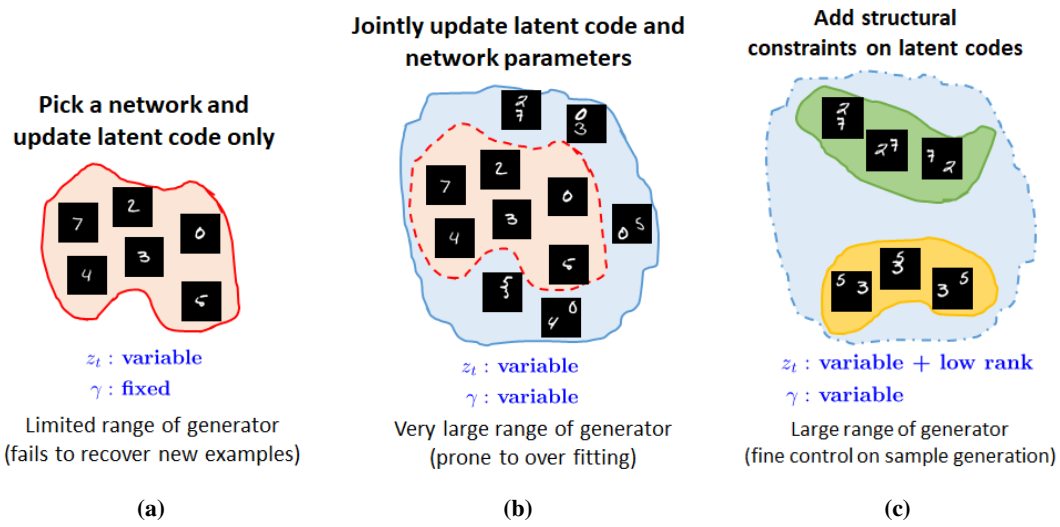


Figure 2: An illustration of different generative models discussed in the paper: (a) Optimizing latent codes can only reconstruct images in the range of the generative network. (b) Jointly optimizing latent code and network weights enables recovery of a larger range of images. (c) Low-rank and similarity constraints on latent code further regularize the problem and potentially explain other structures in data.

generator and reconstruct images that the given initial generator fails on. We show that even though the network has a very large number of parameters, but the joint optimization still converges to a good solution with similarity and low-rank constraints on latent codes.

- We show that, in some cases, the low-rank structure on the latent codes also provides a nice low-dimensional manifold that can be used to generate new frames that are similar to the given sequence.

2 Technical Approach

Let us assume that $x_t \in \mathbb{R}^n$ for $t = 1, \ldots, T$ is a sequence of video frames that we want to reconstruct from the measurements $y_t = A_t x_t + e_t$ as given in (2). The generative model as given in (1) maps a low-dimensional representation vector, $z_t \in \mathbb{R}^k$, to a high-dimensional image as $x_t = G_\gamma(z_t)$. Thus, our goal of video recovery is equivalent to solving the following optimization problem over z_t:

$$y_t = A_t G_\gamma(z_t) + e_t, \tag{3}$$

which can be viewed as a nonlinear system of equations.

2.1 Latent Code Optimization

In latent code optimization, we assume that the function $G_\gamma(\cdot)$ approximates the probability distribution of the set of natural images where our target image belongs. Thus, we can restrict our search for the underlying video sequence, x_t, only in the range of the generator. Similar problem has been studied in [4] for image compressive sensing.

Given a pretrained generator, G_γ, measurement sequence, y_t, and the measurement matrices, A_t, we can solve the following optimization problem to recover the low-dimensional latent codes: $\hat{z}_t$ for our target video

sequence, $\hat{x}_t = G_\gamma(\hat{z}_t)$, as

$$\hat{z}_1, \dots \hat{z}_T = \underset{z_1,\dots,z_T}{\arg\min} \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2. \tag{4}$$

Since we can backpropagate gradient w.r.t. the z_t through the generator, we can solve the problem in (4) using gradient descent. Although latent code optimization can solve compressive sensing problem with high probability, it cannot solve the problem when the images do not belong to the generator. As there are wide variety of images, it is difficult to represent them with a single or a few generators. In such scenarios, latent code optimization proves to be inadequate.

2.2 Joint Optimization of Latent Codes and Generator

Any generator has a limited range within which it can generate images; the range of a generator presumably depends on the types of images used during training. To highlight this limitation, we performed an experiment in which we tried to generate a video sequence that is very different from the examples on which our generator was trained on. This is not a compressive sensing experiment; we are providing original video sequences x_t to the generator and finding the best approximation of the sequence generated by them. The results are shown in Figure 3 using two video sequences: Moving MNIST and Color Wheel. In both cases, network weights are initialized with the weights of a generator that was trained on a different dataset. The pretrained network used for Moving MNIST example was trained on standard MNIST dataset, which does not include any image with two digits. Therefore, the generator trained on MNIST fails on Moving MNIST if we only optimize over the latent code because Moving MNIST dataset consists of images with two digits. The joint optimization of latent code and generator parameters, however, can recover the entire Moving MNIST sequence with high quality. For Color wheel the original generator was trained on CIFAR10 training set which contains diverse category of images. However, as we see in Figure 3, the generator fails to produce quality images Still it cannot perform well on color wheel representation just by latent code update. Joint optimization improves the reconstruction quality significantly.

The results presented in Figure 3 should not be surprising for the following reasons: We are providing a video sequence x_t to the generator $G_\gamma(z_t)$ that has k degrees of freedom for each z_t; therefore, the range of sequences that can be generated by changing the z_t is quite limited for a fixed γ. In contrast, if we let γ change while we learn the z_t, then the network can potentially generate any image in $\mathbb{R}^n$ because we have a very large degrees of freedom. Note that in our generator, the number of parameters in γ is significantly larger than the size of x_t or z_t.

The surprising thing, however, is that we can also recover quality images by jointly optimizing the latent codes z_t and network weights γ while solving the compressive sensing problem. In other words, we can overcome the range limitation of the generator by optimizing generator parameters alongside latent code to get a good reconstruction from compressive measurements as well as good representative latent codes for the video sequence even though the network is highly overparameterized. The resulting optimization problem can be written as

$$\hat{z}_1, \dots, \hat{z}_T; \hat{\gamma} = \underset{z_1,\dots,z_T;\gamma}{\arg\min} \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2, \tag{5}$$

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

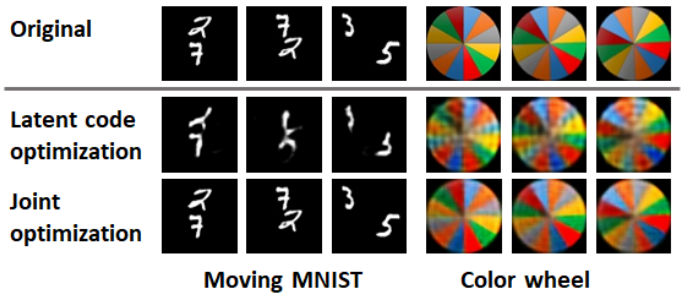


Figure 3: Comparison between optimization over latent code alone and joint optimization over latent code and network weights for representing Moving MNIST and Color wheel test sequence. Latent code optimization fails to generate quality images of sequences that are very different from the training set. Joint optimization can generate both sequences because we have very large degrees of freedom.

This joint optimization of latent code and generator parameter offer the optimization problem a lot of flexibility to generate a wide range of images. As the generator function is highly non-convex, we initialize γ with the pretrained set of weights. After every gradient descent update of the latent codes, z_t, we update the model parameters with stochastic gradient descent.

2.3 Similarity and Low Rank Constraints

2.3.1 Similarity Constraints

A generative prior gives us an opportunity to utilize the corresponding latent codes. The latent codes can be viewed as nonlinear, low-dimensional projection of the original images. In a video sequence, each frame has some similarities with the neighboring frames. Even though the similarity may seem very complex in original dimension, it can become much simpler when we encode each image to a low dimensional latent code. If the latent code is long enough to encode the changes in the image domain, then they can also be used for applying similarity constraint on the image domain.

We assume that if the images are *similar* to each other, then their corresponding latent codes must be similar too. To exploit this structure, we propose to reconstruct the following optimization problem with similarity constraints:

$$\min_{z_1,\dots,z_T;\gamma} \lambda \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2 + (1-\lambda) \sum_{t=1}^{T-1} \beta_t \|z_{t+1} - z_t\|_2^2 \tag{6}$$

where $0 < \lambda < 1$ and the β_t are the weights that represent some measure of similarity between t^{th} and $(t+1)^{th}$ frames. Assuming the adjacent frames in a sequence are close to each other, we fix $\beta_t = 1$ for all t for simplicity.

2.3.2 Low Rank Constraint

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 much lower dimensional space compared to their ambient dimension. For each minibatch, we define a matrix Z such that

$$Z = [z_1 \; z_2 \; \dots \; z_T]$$

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

$$\begin{aligned} \min_{z_1,\ldots,z_T;\gamma} \sum_{t=1}^{T} \|y_t - A_t G_\gamma(z_t)\|_2^2 \\ \text{s.t.} \quad \text{rank}(Z) = r. \end{aligned} \tag{7}$$

We implement this constraint by reconstructing Z matrix from its top r singular vectors in each iteration. Thus the rank of Z matrix formed by a sequence of images becomes r, which implies that we can express each of the latent codes in terms of r orthogonal basis vectors. For rank$(Z) = r$ embedding, we represent each latent code z as a linear combination of the r orthogonal basis vectors $u_1, \ldots, u_r$ as

$$z_i = \sum_{j=1}^{r} \alpha_{ij} u_j \tag{8}$$

where α_{ij} is the weight of the corresponding basis vector.

We can now represent a video sequence with T frames with r orthogonal codes. This offers an additional compression to our latent codes. We use the same idea to linearize motion manifold in latent space.

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

Input: Measurements y_t, measurement matrices A_t, pretrained generator $G_\gamma(\cdot)$
  Initialize the latent codes z_t.
  repeat
    Compute gradients w.r.t. z_t via backpropagation.
    Update latent code matrix $Z = [z_1 \ \cdots \ z_T]$.
    Threshold 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

3 Experiments

In this section, we describe our experimental setup.

Choice of generator: We follow the well-known DCGAN framework [23] for our generators except that we do not use any batch-normalization layer because gradient through the batch-normalization layer is dependent on the batch size and the distribution of the batch. As shown in Figure 1, in DCGAN generator framework, we project the latent code, z, to a larger vector using a fully connected network and then reshape it so that it can work as an input for the following deconvolutional layers. Instead of using any pooling layers, in DCGAN framework, authors [23] propose strided convolution. All the intermediate deconvolution layers are followed by ReLU activation. The last deconvolution layer is followed by Tanh activation function to generate the reconstructed image $x = G(z)$.

Initial generator training: We train our generators by jointly optimizing the generator parameters, γ and latent code, z using SGD optimization by following the procedure in [3]. In each iteration, we first update the generator parameters and then update the latent code using SGD. We use squared-loss function, $\ell_2(x, \hat{x}) = \|x - \hat{x}\|_2^2$ to train the generators. We keep the minibatch size fixed at 256. We use two different trained generators for our experiments: one for RGB images and another for grayscale images. The RGB image generator is trained on CIFAR10 training dataset resized to 64×64. We choose CIFAR10 because it has 10 different categories of images, which helps increase the range of the generator. The grayscale image generator is trained on MNIST digit training dataset resized to 64×64. We used SGD optimizer for optimizing both latent code and network weights. The learning rate for updating z is chosen as 1 and learning rate for updating γ as 0.01.

Measurement matrix: We used three different measurement matrices in our experiments. We first experiment with original images (i.e., A_t is an identity matrix) to test which sequences can be generated by latent code optimization and which ones require joint optimization of latent codes and network weights. Then we experiment with compressive measurements, for which we choose the entries of the A_t independently from $\mathcal{N}(0, \frac{1}{m})$ distribution. For a video sequence of T frames, we generate T independent measurement matrices. Then we experiment with missing pixels (also known as image/video inpainting problem) to show that our algorithm works on other inverse problems as well. For experiments with missing pixels, we randomly dropped a fraction of the pixels from each frame.

Datasets: We test our hypothesis on five datasets, which includes both synthetic and real video sequences. The first test set consists of 10 MNIST test digits. We rotate each digit by 2° per frame for a total of 32 frames. Second test set includes 10 Moving MNIST test sequences [31]. Each test sequence has 20 frames. For the third test set, we generate a color wheel with 12 colors by dividing a circle into 12 equal slices. We rotate the color wheel by 1° per frame for 64 frames. Finally we experiment on different real video sequences from publicly available KTH human action video dataset [26] and UCF101 dataset [30]. We show the results on a person walking video from KTH dataset in this paper because of its simplicity. We cropped the video in the temporal dimension to select 80 frames, which show only unidirectional movement. We also show results for an archery video sequence from UCF101 dataset.

Performance metric: We measure the performance of our recovery algorithms in terms of the reconstruction error PSNR. For a given image x and its reconstruction $\hat{x}$, PSNR is defined as

$$\text{PSNR}(x, \hat{x}) = 20 \log_{10} \frac{\max(x) - \min(x)}{\sqrt{\text{MSE}(x, \hat{x})}}$$

where max and min corresponds to the maximal and minimal value the image x can attain respectively, and MSE is the mean squared error.

4 Results

4.1 Compact Video Representation

In our first set of experiments, we simply generate a given video sequence using our network by optimizing only over the latent codes and by optimizing jointly over the latent codes and network parameters. In other words, A_t is an identity matrix in these experiments. A summary of our experimental results is presented in Table 1 that correspond to the case when original video sequence is used to estimate latent codes that provide best approximation of the sequence. We observe from Table 1 that adding similarity and low-rank constraints provides small improvement in the image approximation performance. This might be because of the fact that

Table 1: Results for compact video representation via generative model in terms of PSNR. In each experiment, we approximated a video sequence by either optimizing over latent codes or joint optimization over latent codes and network weights. First column (Update z_t) corresponds to the algorithm of [4]

	Latent code optimization			Joint optimization		
	Update z_t	Low-rank constraints ($r = 5$)	Similarity constraint	Update z_t and γ	Low-rank constraints ($r = 5$)	Similarity constraint
Rotating MNIST	25.82	25.73	26.81	33.75	33.78	33.9
Moving MNIST	18.55	16.99	18.51	31.17	31.16	31.15
Color Wheel	18.24	17.97	18.31	22.07	21.92	22.05
Archery	24.15	23.13	24.49	26.5	23.15	27.26
Person Walking	27.55	23.30	27.55	27.9	26.72	27.91

the frames are already slowly changing and we have enough measurements to approximate them. However, jointly optimizing both latent codes and network parameters provides a significant gain in the reconstruction PSNR.

4.2 Optimization over z_t with Constraints

In our first experiment, we test latent code optimization with and without similarity and low-rank constraints. We show some example reconstructions for the inpainting problem with 90% missing pixels in Figure 4. For similarity constraint, $\lambda = 0.6$ is chosen for both cases. For low-rank constraints, the optimal values of rank for Rotating MNIST and Person Walking are $rank = 4$ and $rank = 16$, respectively. We can observe for very low measurements, low rank generator not only represent the video sequence with lower number of parameters in latent codes (12.5% and 20% of the total frames respectively for Rotating MNIST and Person Walking), it also gives boost in reconstruction performance.

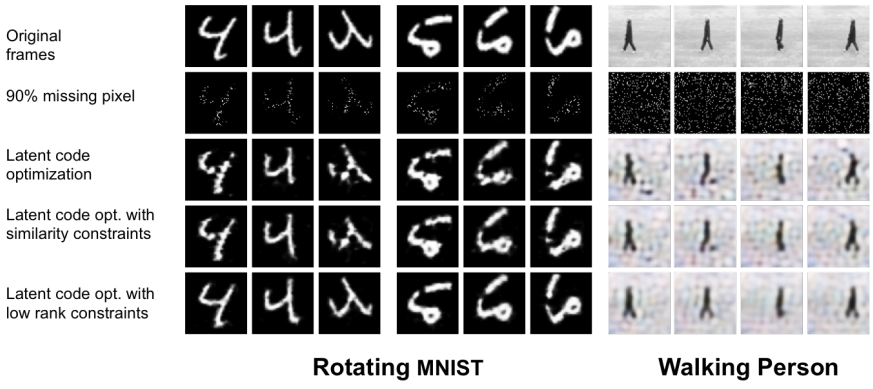


Figure 4: Example reconstruction results from inpainted video sequence with 90% missing pixels for two rotating MNIST sequences and person walking video sequence using latent code optimization.

We also performed a number of experiments for latent code optimization (with and without similarity and low-rank constraints) for different datasets and measurements. A summary of our experimental results is presented in Table 2. The results refer to experiments in which we estimate latent codes from the compressive measurements of the sequence. We observe that adding similarity or low-rank constraints in the compressive

Table 2: Reconstruction PSNR for compressive sensing problems. First four rows correspond to video recovery from m Gaussian measurements. Last five rows correspond to the recovery of videos from 80% missing pixels per frame. First column (Update z_t) corresponds to the algorithm of [4]

	Latent code optimization			Joint optimization		
	Update z_t	Low-rank constraint	Similarity constraint	Update z_t and γ	Low-rank constraint	Similarity constraint
	Experiments with compressive Gaussian measurements					
Rotating MNIST ($m = 200$)	20.35	20.75 (r=5)	22.13	30.9	31 (r=5)	32.97
Moving MNIST ($m = 512$)	16.75	16.9 (r=12)	17.57	24.43	27.03 (r=4)	27.2
Color Wheel ($m = 1024$)	16.95	17.96 (r=6)	17.09	21.92	23.71 (r=6)	21.8
Archery ($m = 512$)	21.58	23.54(r=16)	23.15	25.82	26.9 (r=21)	25.83
	Experiments with 80% Missing pixels					
Rotating MNIST	19.15	25.07(r=4)	24.45	26.54	29.58 (r=3)	28.53
Moving MNIST	16.44	16.82 (r=9)	17.34	18.65	19.02(r=9)	19.55
Color Wheel	16.54	17.85 (r=6)	16.75	18.46	19.96 (r=4)	18.88
Archery	23.15	23.8 (r=22)	23.32	23.6	23.81 (r=21)	23.57
Person Walking	25.34	26.1 (r=21)	25.9	25.8	26.17 (r=22)	25.96

sensing problems shows significant improvement in the quality of reconstruction.

4.3 Joint Optimization over z_t and γ with Constraints

As we discussed before in Figure 3, the joint optimization over z_t and γ can generate images that are very different from the images network is trained on. Table 1 refers to similar experiments in which we are given the original video sequence and we want to estimate latent codes and network weight that can best approximate the given video sequence. We observe that joint optimization offers a significant performance boost compared to latent code optimization alone. As we discussed before, this is expected because we have a lot more degrees of freedom in the case of joint optimization than what we have for latent code optimization. The similarity or low-rank constraints do not provide a significant boost while approximating the video sequence.

Table 2 summarizes results for compressive measurements, where we are only given linear measurement of the video sequence and we want to estimate the latent codes z_t and network weights γ that minimize the objectives in (5) or (7). We performed experiment on image inpainting and compressive sensing problems. For image inpainting problem, we show reconstruction results for 80% missing pixels in Table 2. We also show results for different compressive measurements for different synthetic and real video sequences. We can observe from Table 2 that with low-rank constraints on the generator, we can not only represent the whole video sequence with a very few latent codes, but also get better reconstruction than full rank cases. Similarity constraint on latent codes also show improvement in reconstruction performance when the measurements are low.

Some examples of video sequences from compressive measurements are presented in Figure 5. In each of the experiments, we compute m Gaussian measurements of each frame in a sequence and then solve the optimization problems in (5) (this corresponds to the full-rank recovery) and (7) with $r = 4$ (this corresponds to the rank-4 recovery). We observe that low-rank constraints provide a small improvement in terms of the

quality of reconstruction.

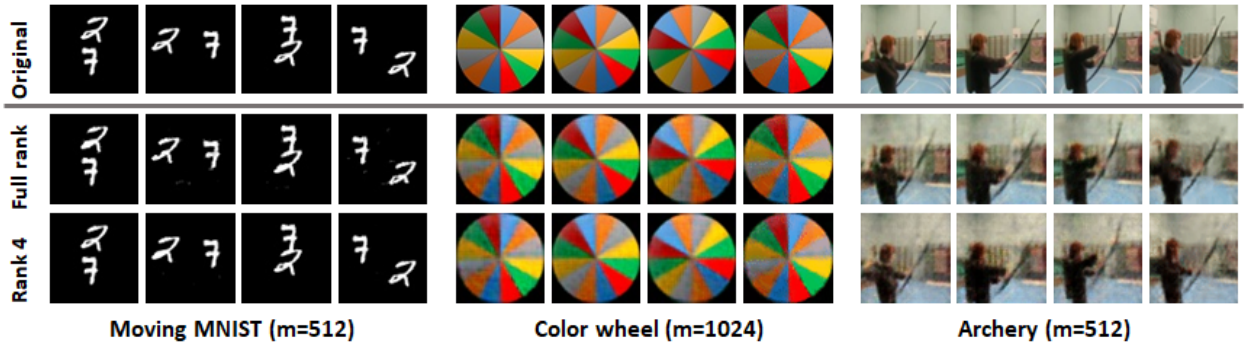


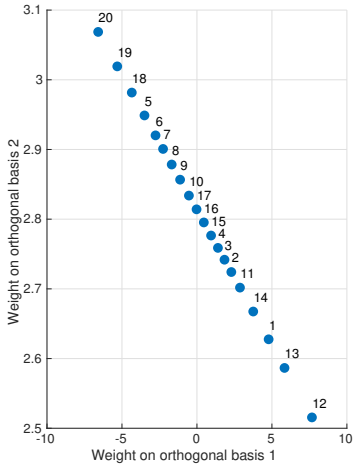
Figure 5: Examples of reconstructed images for experiments with different datasets using linear measurements. The results are from joint optimization of latent code and generator weights. First row shows samples from original video sequence. Second row shows reconstruction without low-rank constraint. Third row shows results when latent codes for each sequence are restricted to a rank-4 matrix.

4.4 Linearizing Motion Manifold via Joint Optimization and Low Rank Constraint

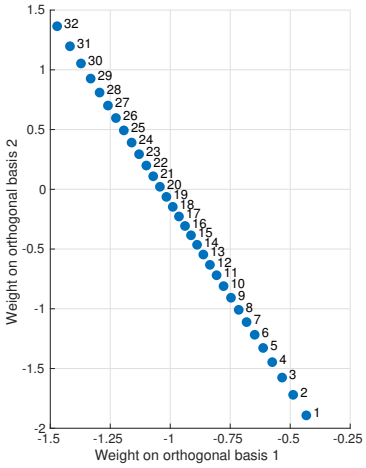
In this section, we present our preliminary experiments on linearizing articulation manifold of a video sequence by imposing low-rank structure on the latent codes. In our experiment, we force our latent codes to map on a straight line by defining each $z_t = \bar{z} + \alpha_t u$, where $\bar{z}, u \in \mathbb{R}^k$ and $\alpha_t \in \mathbb{R}$ are scalar. We impose this rank-2 constraint by solving the problem in (7) but instead of approximating the z_t using the top two singular vectors, we approximate them using their mean and first principal vector.

We further investigate the linearization of multiple video sequences while optimizing the same generator weights to generate those sequences. In this experiment, we form the Z matrix by concatenating latent codes for multiple different sequences. Then we apply rank-2 constraint on the entire Z matrix using top two singular vectors. We simultaneously apply linearity constraint on each sequence by imposing rank=2 constraint on the latent codes for each sequence separately using mean and first principal vector as mentioned above.

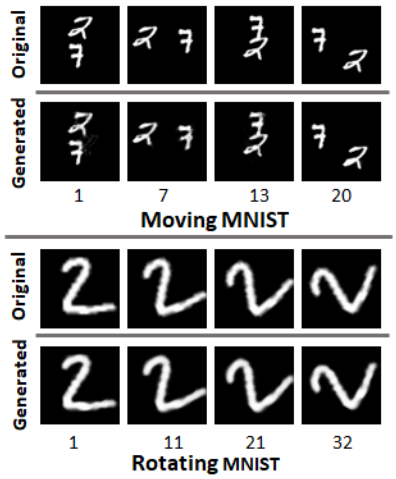
We plot the embedding of each z in terms of two orthogonal basis vectors in Figures 6a and 6b. We observe that a well-defined rotation in image domain is translated into a line in the latent space. We also observe that as we increase the rotating angles, the corresponding embedding moves along a straight line in the direction of first principal vector in an increasing order. We plot the embedding of three sequences of Rotating MNIST in Figure 7a. We observe that the rotation of different digits are translated into different lines in the 2D latent space. Furthermore, latent codes for each of the sequences preserves their sequential order. However, in the case of moving MNIST, even though we get perfect reconstruction with the line embedding, but the order of the video sequence is not preserved in the embedded space. We did not impose any constraint in our optimization to preserve the order, but we expect that if the video sequence changes in such a manner that frames that are farther in time are also farther in content, then we will see the order will be preserved. We leave this investigation for future work.



(a) Manifold of latent codes for Moving MNIST.

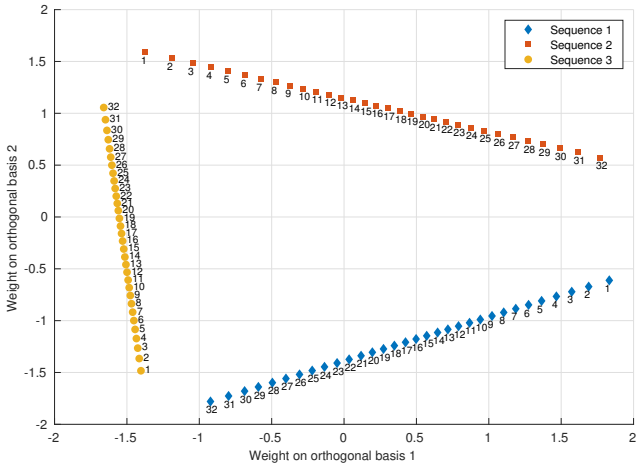


(b) Manifold of latent codes for Rotating MNIST.

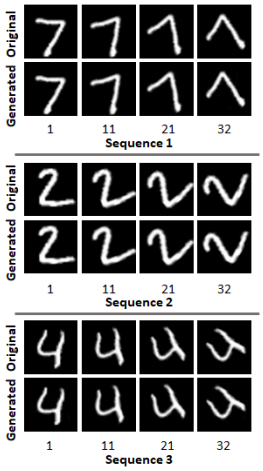


(c) Generated images for some frames of the sequences. We denote the frame number below each frame.

Figure 6: Approximated images for Moving MNIST and one Rotating MNIST video sequences from original video sequences using joint optimization in which the latent codes are constrained to lie on a straight line using PCA. In (a) and (b), these representations are linear by virtue of the constraint. Furthermore, for Rotating MNIST, the latent codes are sequentially arranged. The quality of generated frames are good (Average PSNR for Moving MNIST: 26.8 dB; Rotating MNIST: 33.6 dB).



(a) Manifold of latent codes for 3 sequences from Rotating MNIST.

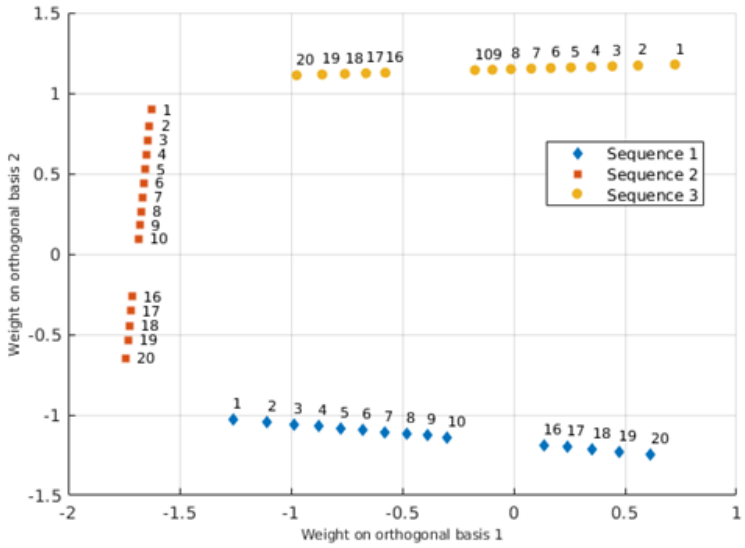


(b) Generated images for frames of 3 sequences from Rotating MNIST.

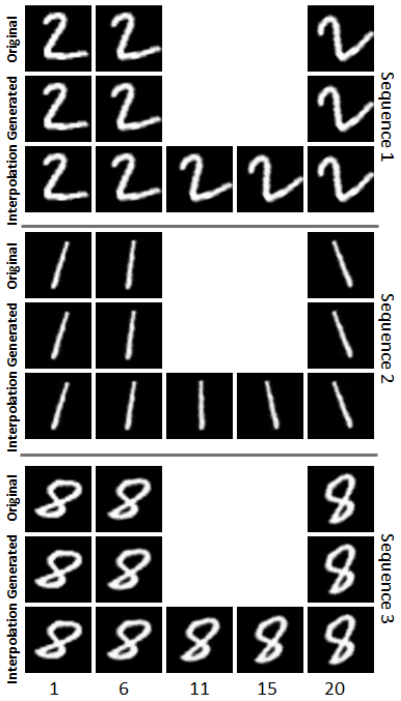
Figure 7: Approximated images and latent space representation from original video sequences for three different Rotating MNIST video sequences using joint optimization of latent codes and network weights for the same generator. Here the latent codes of each sequences are constrained to lie on a straight line using PCA. Different sequences are aligned to different lines in 2D plane. Furthermore, they maintained sequential arrangement.

4.5 Interpolation in Latent Space to Generate Missing Frames

If the latent codes follow some sequential order, it is possible to generate intermediate images between each frames. We test this idea using three Rotating MNIST sequences. Each sequence originally contained 20 frames, where, in each frame, the digit is rotated 2° from the previous frame. However, we set aside 11^{th} to 15^{th} frames while optimizing the generator to approximate those frames. We perform joint optimization of z and γ using rank=2 constraint on the latent codes and linearization constraint on the latent codes of each sequence. When we observe the latent code representation for the approximated images, we observe that the latent codes follow sequential order but there are significant gap between the latent codes of 10^{th} and 16^{th} frames. We can observe this phenomena in Figure 8a. We then try to generate 1000 frames between frame 1 and frame 20 using linear interpolation between corresponding latent codes. We keep the same network weights which is giving us the approximation of the original sequences. We can observe from Figure 8b that we can generate the missing frames in that way. However, we can choose frame 1 and 20 here as two end points for linear interpolating because the entire sequence is maintaining the sequential order in their linear latent space representation. But in cases where the sequence only maintains sequential order locally, we can select interpolation end points from the cluster of frames which maintains sequential order.



(a) Manifold of latent codes for 3 sequences from Rotating MNIST with missing frames.

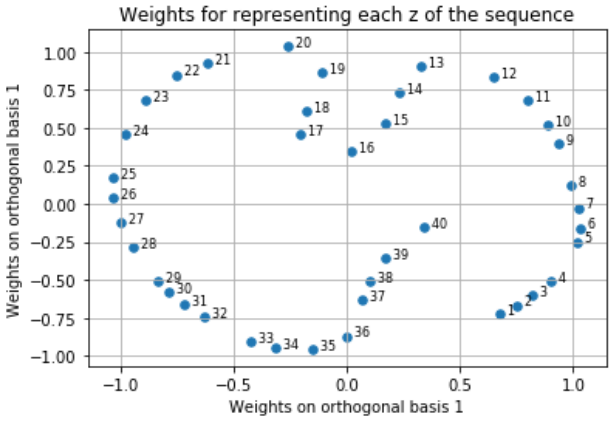


(b) Approximation and interpolation result for some frames.

Figure 8: Approximation and interpolation (in latent space) images and corresponding latent space representation for 3 Rotating MNIST sequences with missing frames. We denote the frame number for each frame. Frame 11 to Frame 15 are missing while joint optimization of z and γ.

4.6 Low-Dimensional Embedding of Complex Motion

We further experiment on a complex real life motion using spinning figures dataset from [18]. We selected a rotating bunny sequence and cropped only the bunny from the images. The bunny completes one rotation in 15 frames. We selected first 10 frames from each of the 4 full rotations and keep the similar rotations close to each other. We try to find out if this sequence maintain its sequential order in any latent space. We observe the representation of the sequence in latent space using $rank = 3$ constraint. We impose $rank = 3$ constraint by selecting mean and first two principal vectors. So, the latent codes are constrained to 2D plane in the 3D space. We show the approximation of bunny sequence using this constraint in Figure 9b and the corresponding latent space representation in Figure 9a. We can observe from the latent space representation that the sequence maintained its sequential order in this representation.



(a) Latent code representation for approximated rotating bunny sequence.

(b) Approximated images for rotating bunny sequence.

Figure 9: Approximated images and corresponding latent space for rotating bunny sequence. We constrain the latent codes to lie on 2D plane of a 3D space using mean and first two principal vectors.

5 Discussion and Future Work

We proposed a generative model for low-rank representation and reconstruction of video sequences. We presented experiments to demonstrate that video sequences can be reconstructed from compressive measurements by either optimizing over the latent code or jointly optimizing over the latent codes and network weights. We observed that adding similarity and low-rank constraints in the optimization regularizes the recovery problems and improves the quality of reconstruction. We presented some preliminary experiments to show that low-rank embedding of latent codes with joint optimization can potentially be useful in linearizing articulation manifolds of the video sequence. An implementation of our algorithm with pretrained models is available here: https://github.com/CSIPlab/gmlr.

In all our experiments, we observed that joint optimization performs remarkably well for compressive measurements as well. Even though the number of measurements are extremely small compared to the number of parameters in γ, the solution almost always converges to a good sequence. We attribute this success to a good initialization of the network weights and hypothesize that a "good set of weights" are available near the initial set of weights in all these experiments. We intend to investigate a proof of the

presence of good local minima around initialization in our future work.

References

[1] R. Baraniuk and P. Steeghs. Compressive radar imaging. In *Radar Conference, 2007 IEEE*, pages 128–133. IEEE, 2007.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Machine Intell.*, 35(8):1798–1828, 2013.

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

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

[5] E. J. Candes, Y. C. Eldar, D. Needell, and P. Randall. Compressed sensing with coherent and redundant dictionaries. *Applied and Computational Harmonic Analysis*, 31(1):59–73, 2011.

[6] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

[7] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 2172–2180, 2016.

[8] C. Christopoulos, A. Skodras, and T. Ebrahimi. The jpeg2000 still image coding system: an overview. *IEEE transactions on consumer electronics*, 46(4):1103–1127, 2000.

[9] A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–8, 2018.

[10] D. L. Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[11] 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*, 25(2):83–91, 2008.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 2672–2680, 2014.

[13] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 5767–5777, 2017.

[14] R. Heckel and P. Hand. Deep decoder: Concise image representations from untrained non-convolutional networks. *arXiv preprint arXiv:1810.03982*, 2018.

[15] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[16] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[17] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[18] R. R. Lederman and R. Talmon. Learning the geometry of common latent variables using alternating-diffusion. *Applied and Computational Harmonic Analysis*, 44(3):509–536, 2018.

[19] R. Lin, Y. Zhang, H. Wang, X. Wang, and Q. Dai. Deep convolutional neural network for decompressed video enhancement. In *Data Compression Conference (DCC), 2016*, pages 617–617. IEEE, 2016.

[20] Z. C. Lipton and S. Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017.

[21] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 1029–1038, 2016.

[22] A. Puri and A. Eleftheriadis. Mpeg-4: An object-based multimedia coding standard supporting mobile applications. *Mobile Networks and Applications*, 3(1):5–32, 1998.

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

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[25] S. Santurkar, D. Budden, and N. Shavit. Generative compression. In *2018 Picture Coding Symposium (PCS)*, pages 258–262. IEEE, 2018.

[26] 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*, volume 3, pages 32–36. IEEE, 2004.

[27] 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.

[28] J. V. Shi, A. C. Sankaranarayanan, C. Studer, and R. G. Baraniuk. Video compressive sensing for dynamic mri. *BMC neuroscience*, 13(1):P183, 2012.

[29] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *Proc. IEEE Conf. Comp. Vision and Pattern Recog. (CVPR)*, pages 2242–2251. IEEE, 2017.

[30] 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.

[31] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *Proc. Int. Conf. Machine Learning*, pages 843–852, 2015.

[32] 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*, volume 5558, pages 454–475. International Society for Optics and Photonics, 2004.

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

[34] 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.

[35] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Proc. Adv. in Neural Inf. Proc. Sys. (NIPS)*, pages 613–621, 2016.

[36] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[37] 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)*, pages 1016–1020. IEEE, 2013.

[38] 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*, 27(6):1182–1195, 2017.

[39] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *Proc. European Conf. Comp. Vision (ECCV)*, 2016.

A Supplementary Material

A.1 Image Inpainting on Additional Video Sequences

We experiment on different video sequences from all six categories (Boxing, Handclapping, Handwaving, Jogging, Running, Walking) from KTH video dataset. To reduce computational complexity, we have selected part of these videos in a batch. Table 3 includes the number of frames for our test videos. We experiment on image inpainting with 80% missing pixels. We experiment for both latent code optimization and joint optimization of latent code and network weight. In Table 3, we report experimental results and in Figure 10, we demonstrate some representational examples. Joint optimization of z and γ significantly outperforms latent code optimization because the video sequences are not from the range of the pretrained generator. Furthermore, applying rank=2 linearization constraint on latent code we observe similar performance as full rank reconstruction for joint optimization.

Table 3: Reconstruction PSNR for inpainting problem with 80% missing pixels on different KTH video sequences. We show results for latent code optimization and joint optimization of z and γ with and without linearization constraint on latent codes.

Video	No. of Frames	Latent Code optimization		Joint Optimization	
		Full rank	Rank=2 (linearized)	Full rank	Rank=2 (linearized)
Boxing	50	22.45	22.62	32.37	30.38
Handclapping	50	26.03	26.2	35.74	33.98
Handwaving	50	22.29	20.65	30.01	27.48
Jogging	30	23.82	18.58	26.4	24.01
Running	30	25.74	20.66	27.54	27.56
Walking	55	23.72	18.44	27.53	27.33

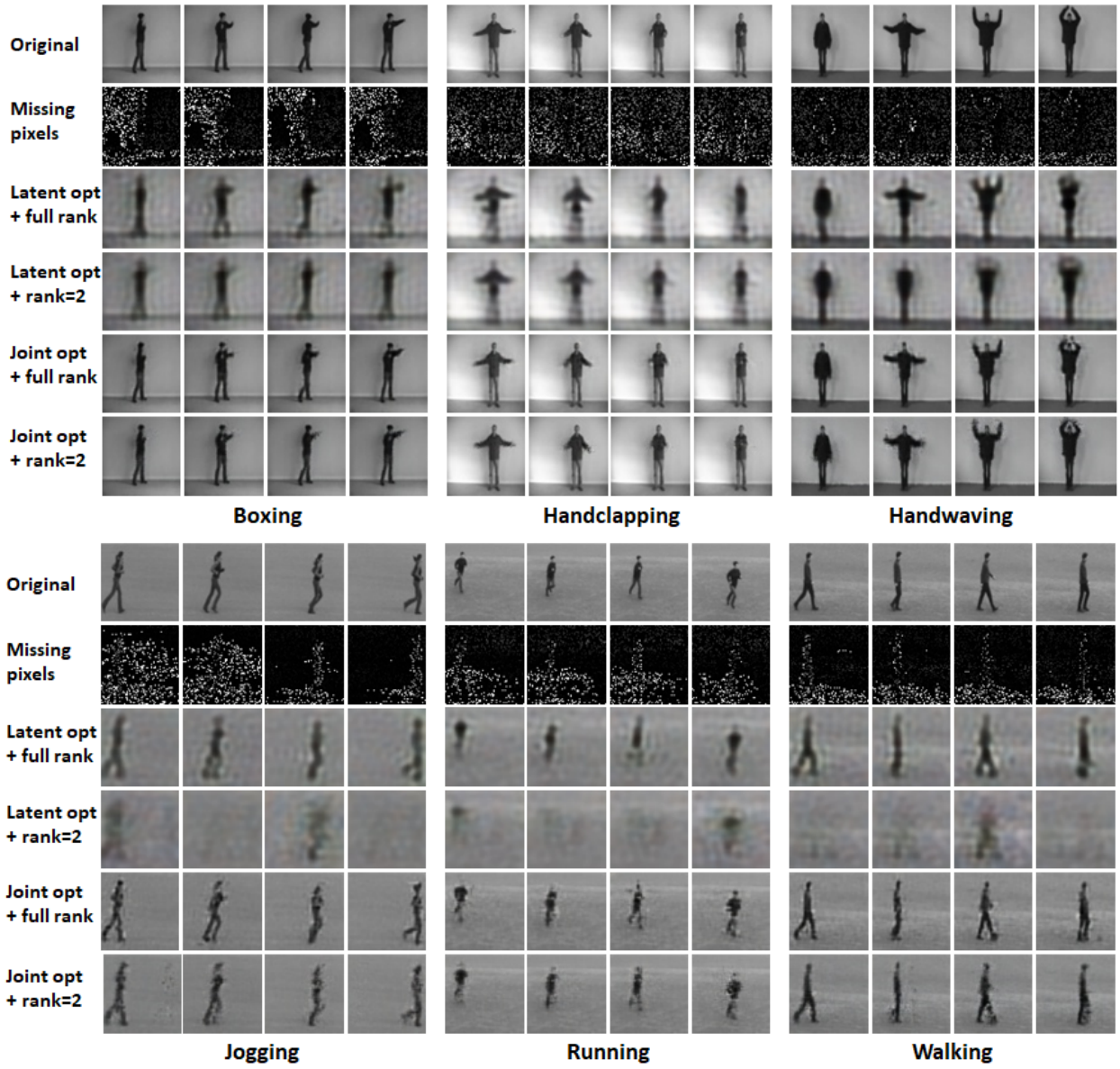


Figure 10: Reconstructions from inpainted video sequence with 80% missing pixels for different videos from KTH dataset. We enforce rank=2 with linearization constraint using PCA.

A.2 Untrained Network vs Pretrained Network for Initialization

We further experiment with untrained generator like [33, 34]. We observe that if we initialize network weights with pretrained network weights, network converges faster even for the images that were not used in the training but fall under the similar distribution. In Figure11, we show reconstruction loss vs number of iteration curve for a Rotating MNIST and a Handclapping video. We show these results for inpainting problem with 80% missing pixels. We can observe that for Rotating MNIST video, random initialization shows false convergence before finally converging. It becomes difficult for some datasets like Moving MNIST to find a convergence using untrained network weights as initialization. So, we use the weights of a pretrained network

as initialization.

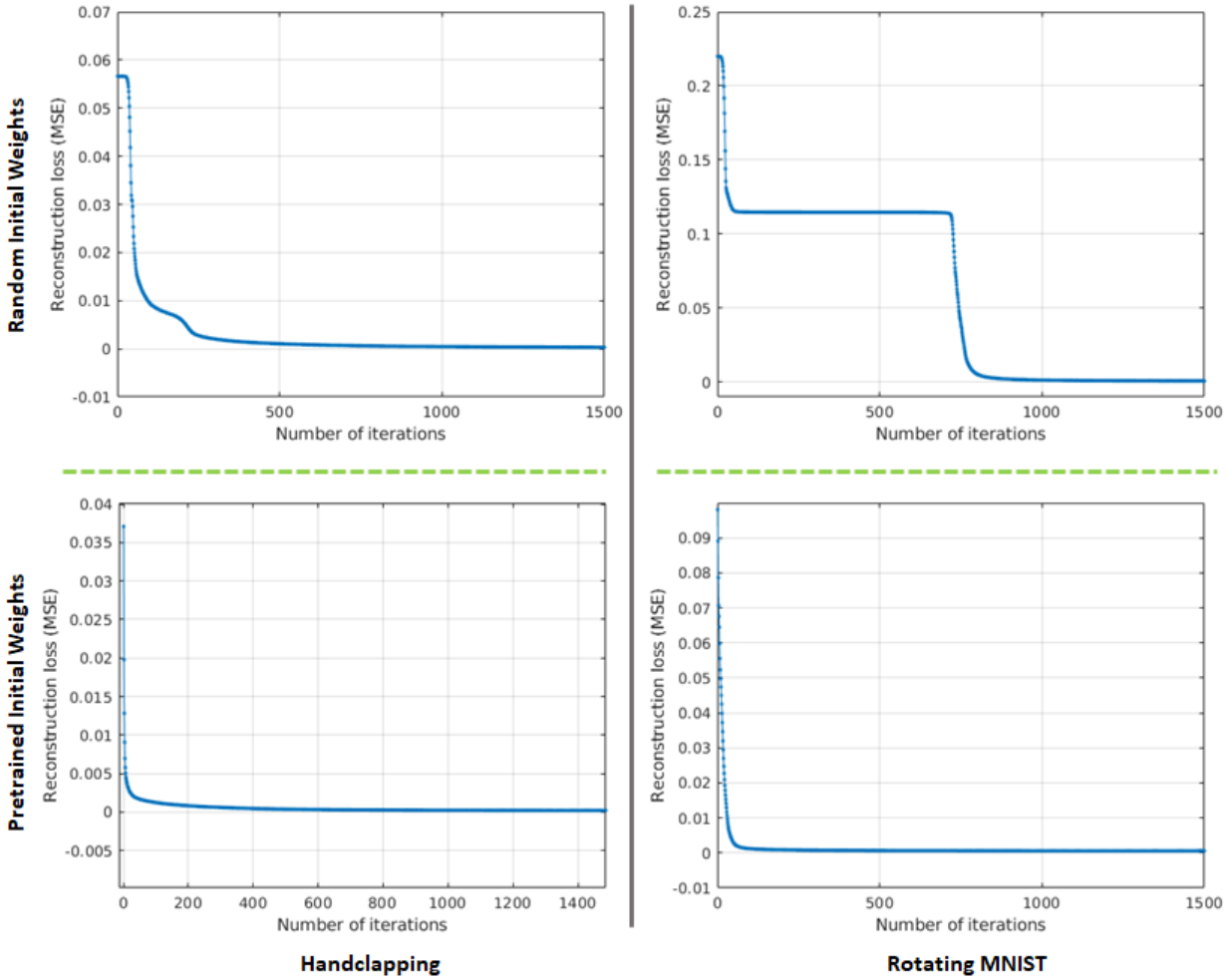


Figure 11: Reconstruction loss curve for a Handclapping and a Rotating MNIST video for initialization with untrained network weights and pretrained network weights. The loss curves are for inpainting experiments with 80% missing pixels.

A.3 Network Parameters

We use two different generator networks for RGB image generation and grayscale image generation. In both generators, we use 4×4 filters in deconvolutional layers. For RGB image generator, a 256 dimensional latent code is projected and reshaped into $512 \times 4 \times 4$ whereas for grayscale image generator, a 32 dimensional latent code is projected and reshaped into $256 \times 4 \times 4$. The number of kernel for each deconvolutional layer of RGB image generator is 256, 128, 64 and 3, respectively. For grayscale image generator, number of kernel for each deconvolutional layer is 128,64,32 and 1, respectively. The number of parameters for each generator is shown in Table 4.

Table 4: Number of parameters in different layers of the generator networks used in the experiments.

Layers	Number of Parameters	
	RGB Image Generator	Grayscale Image Generator
Fully-connected + reshape + ReLU	2,097,152	131,072
Deconv 1 + ReLU	2,097,152	524,288
Deconv 2 + ReLU	524,288	131,072
Deconv 3 + ReLU	131,072	32,768
Deconv 4 + Tanh	3,072	512
Total Parameters	4,852,736	819,712