Deep Fully-Connected Networks for Video Compressive Sensing

Michael Iliadis*
Northwestern University, EECS
miliad@u.northwestern.edu

Leonidas Spinoulas*
Northwestern University, EECS
leonisp@u.northwestern.edu

Aggelos K. Katsaggelos
Northwestern University, EECS
aggk@eecs.northwestern.edu

Abstract

In this work we present a deep learning framework for video compressive sensing. The proposed formulation enables recovery of video frames in a few seconds at significantly improved reconstruction quality compared to previous approaches. Our investigation starts by learning a linear mapping between video sequences and corresponding measured frames which turns out to provide promising results. We then extend the linear formulation to deep fully-connected networks and explore the performance gains using deeper architectures. Our analysis is always driven by the applicability of the proposed framework on existing compressive video architectures. Extensive simulations on several video sequences document the superiority of our approach both quantitatively and qualitatively. Finally, our analysis offers insights into understanding how dataset sizes and number of layers affect reconstruction performance while raising a few points for future investigation.

Code is available at Github: https://github.com/miliadis/DeepVideoCS

1 Introduction

The subdivision of time by motion picture cameras, the frame rate, limits the temporal resolution of a camera system. Even though frame rate increase above 30 Hz may be imperceptible to human eyes, high speed motion picture capture has long been a goal in scientific imaging and cinematography communities. Despite the increasing availability of high speed cameras through the reduction of hardware prices, fundamental restrictions still limit the maximum achievable frame rates.

Video compressive sensing (CS) aims at increasing the temporal resolution of a sensor by incorporating additional hardware components to the camera architecture and employing powerful computational techniques for high speed video reconstruction. The additional components operate at higher frame rates than the camera's native temporal resolution giving rise to low frame rate multiplexed measurements which can later be decoded to extract the unknown observed high speed video sequence. Despite its use for high speed motion capture [24], video CS also has applications to coherent imaging (e.g., holography) for tracking high-speed events [41] (e.g., particle tracking, observing moving biological samples). The benefits of video CS are even more pronounced for non-visible light applications where high speed cameras are rarely available or prohibitively expensive (e.g., millimeter-wave imaging, infrared imaging) [2, 4].

*Indicates equal contribution.

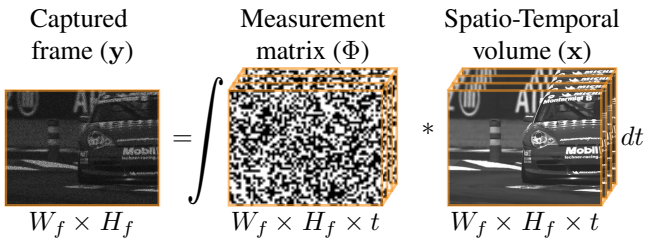

Figure 1: Temporal compressive sensing measurement model.

Video CS comes in two incarnations, namely, spatial CS and temporal CS. Spatial video CS architectures stem from the well-known single-pixel-camera [8], which performs spatial multiplexing per measurement, and enable video recovery by expediting the capturing process. They either employ fast readout circuitry to capture information at video rates [5] or parallelize the single-pixel architecture using multiple sensors, each one responsible for sampling a separate spatial area of the scene [4, 39].

In this work, we focus on temporal CS where multiplexing occurs across the time dimension. Figure 1 depicts this process, where a spatio-temporal volume of size $W_f \times H_f \times t = N_f$ is modulated by t binary random masks during the exposure time of a single capture, giving rise to a coded frame of size $W_f \times H_f = M_f$.

We denote the vectorized versions of the unknown signal and the captured frame as $\mathbf{x} : N_f \times 1$ and $\mathbf{y} : M_f \times 1$, respectively. Each vectorized sampling mask is expressed as $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_t$ giving rise to the measurement model

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

where $\Phi = [diag(\boldsymbol{\phi}_1), \ldots, diag(\boldsymbol{\phi}_t)] : M_f \times N_f$ and $diag(\cdot)$ creates a diagonal matrix from its vector argument.

Various successful temporal CS architectures have been proposed. Their differences mainly involve the implementation of the random masks on the optical path (i.e., the measurement matrix in Figure 1). Digital micromirror devices (DMD), spatial light modulators (SLM) and liquid crystal on silicon (LCoS) were used in [4, 39, 10, 22, 31] while translating printed masks were employed in [16, 23]. Moreover, a few architectures have eliminated additional optical elements by directly programming the chip's readout mode through hardware circuitry modifications [9, 28, 35].

Despite their reasonable performance, temporal CS architectures lack practicality. The main drawback is that existing reconstruction algorithms (e.g., using sparsity models [4, 13], combining sparsity and dictionary learning [22] or using Gaussian mixture models [44, 45]) are often too computationally intensive, rendering the reconstruction process painfully slow. Even with parallel processing, recovery times make video CS prohibitive for modern commercial camera architectures.

In this work, we address this problem by employing deep learning and show that video frames can be recovered in a few seconds at significantly improved reconstruction quality compared to existing approaches.

Our contributions are summarized as follows:

1. We present the first deep learning architecture for temporal video CS reconstruction approach, based on fully-connected neural networks, which learns to map directly temporal CS measurements to video frames. For such task to be practical, a measurement mask with a repeated pattern is proposed.

2. We show that a simple linear regression-based approach learns to reconstruct video frames adequately at a minimal computational cost. Such reconstruction could be used as an initial point to other video CS algorithms.

3. The learning parading is extended to deeper architectures exhibiting reconstruction quality and computational cost improvements compared to previous methods.

2 Motivation and Related Work

Deep learning [19] is a burgeoning research field which has demonstrated state-of-the-art performance in a multitude of machine learning and computer vision tasks, such as image recognition [12] or object detection [30].

In simple words, deep learning tries to mimic the human brain by training large multi-layer neural networks with vast amounts of training samples, describing a given task. Such networks have proven very successful in problems where analytical modeling is not easy or straightforward (e.g., a variety of computer vision tasks [17, 21]).

The popularity of neural networks in recent years has led researchers to explore the capabilities of deep architectures even in problems where analytical models often exist and are well understood (e.g., restoration problems [3, 34, 42]). Even though performance improvement is not as pronounced as in classification problems, many proposed architectures have achieved state-of-the-art performance in problems such as deconvolution, denoising, inpainting, and super-resolution.

More specifically, investigators have employed a variety of architectures: deep fully-connected networks or multi-layer perceptrons (MLPs) [3, 34]; stacked denoising auto-encoders (SDAEs) [42, 1, 6, 38], which are MLPs whose layers are pre-trained to provide improved weight initialization; convolutional neural networks (CNNs) [39, 36, 7, 20, 32, 43] and recurrent neural networks (RNNs) [14].

Based on such success in restoration problems, we wanted to explore the capabilities of deep learning for the video CS problem. However, the majority of existing architectures involve outputs whose dimensionality is smaller than the input (e.g., classification) or have the same size (e.g., denoising/deblurring). Hence, devising an architecture that estimates N_f unknowns, given M_f inputs, where $M_f \ll N_f$ is not necessarily straightforward.

Two recent studies, utilizing SDAEs [26] or CNNs [18], have been presented on spatial CS for still images exhibiting promising performance. Our work constitutes the first attempt to apply deep learning on temporal video CS. Our approach differs from prior 2D image restoration architectures [3, 34] since we are recovering a 3D volume from 2D measurements.

3 Deep Networks for Compressed Video

3.1 Linear mapping

We started our investigation by posing the question: can training data be used to find a linear mapping W such that $\mathbf{x} = W\mathbf{y}$? Essentially, this question asks for the inverse of Φ in equation (1) which, of course, does not exist. Clearly, such a matrix would be huge to store but, instead, one can apply the same logic on video blocks [22].

We collect a set of training video blocks denoted by $\mathbf{x}_i$, $i \in \mathbb{N}$ of size $w_p \times h_p \times t = N_p$. Therefore, the measurement model per block is now $\mathbf{y}_i = \Phi_p \mathbf{x}_i$ with size $M_p \times 1$, where $M_p = w_p \times h_p$ and Φ_p refers to the corresponding measurement matrix per block.

Collecting a set of N video blocks, we obtain the matrix equation

$$Y = \Phi_p X, \tag{2}$$

where $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$, $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ and Φ_p is the same for all blocks. The linear mapping $X = W_p Y$ we are after can be calculated as

$$\min_{W_p} \|X - W_p Y\|_2^2 \rightarrow W_p = \left(XY^T\right)\left(YY^T\right)^{-1}, \tag{3}$$

where W_p is of size $N_p \times M_p$.

Intuitively, such an approach would not necessarily be expected to even provide a solution due to ill-posedness. However, it turns out that, if N is sufficiently large and the matrix Φ_p has at least one nonzero in each row (i.e., sampling each spatial location at least once over time), the estimation of $\mathbf{x}_i$'s by the $\mathbf{y}_i$'s provides surprisingly good performance.

Specifically, we obtain measurements from a test video sequence applying the same Φ_p per video block and then reconstruct all blocks using the learnt W_p. Figure 2 depicts the average peak signal-to-noise ratio (PSNR) and structural similarity metric (SSIM) [40] for the reconstruction of 14 video

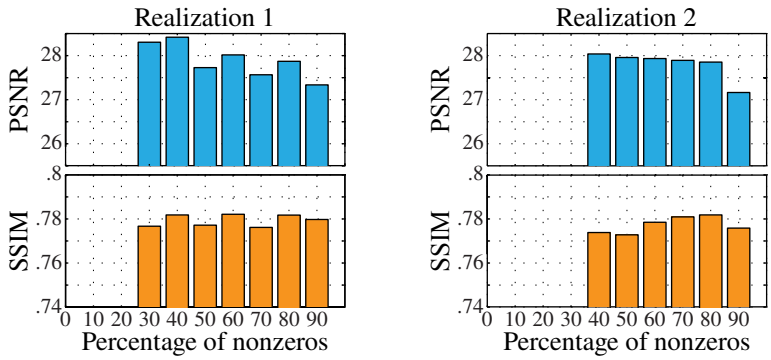

Figure 2: Average reconstruction performance of linear mapping for 14 videos (unrelated to the training data), using measurement matrices Φ_p with varying percentages of nonzero elements.

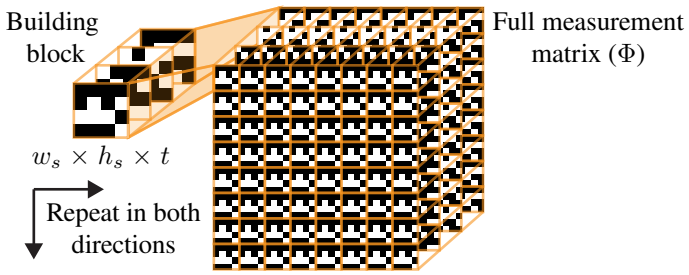

Figure 3: Construction of the proposed full measurement matrix by repeating a three dimensional random array (building block) in the horizontal and vertical directions.

sequences using 2 different realizations of the random binary matrix Φ_p for varying percentages of nonzero elements. The empty bars for $10-20\%$ and $10-30\%$ of nonzeros in realizations 1 and 2, respectively, refer to cases when there was no solution due to the lack of nonzeros at some spatial location. In these experiments $w_p \times h_p \times t$ was selected as $8 \times 8 \times 16$ simulating the reconstruction of 16 frames by a single captured frame and $N = 10^6$.

3.2 Measurement Matrix Construction

Based on the performance in Figure 2, investigating the extension of the linear mapping in (3) to a nonlinear mapping using deep networks seemed increasingly promising. In order for such an approach to be practical, though, reconstruction has to be performed on blocks and each block must be sampled with the same measurement matrix Φ_p. Furthermore, such a measurement matrix should be realizable in hardware. Hence we propose constructing a Φ which consists of repeated identical building blocks of size $w_s \times h_s \times t$, as presented in Figure 3. Such a matrix can be straightforwardly implemented on existing systems employing DMDs, SLMs or LCoS [4, 39, 10, 22, 31]. At the same time, in systems utilizing translating masks [16, 23], a repeated mask can be printed and shifted appropriately to produce the same effect.

In the remainder of this paper, we select a building block of size $w_s \times h_s \times t = 4 \times 4 \times 16$ as a random binary matrix containing 50% of nonzero elements and set $w_p \times h_p \times t = 8 \times 8 \times 16$, such that $N_p = 1024$ and $M_p = 64$. Therefore, the compression ratio is $1/16$. In addition, for the proposed matrix Φ, each $4 \times 4 \times 16$ block is the same allowing reconstruction for overlapping blocks of size $8 \times 8 \times 16$ with spatial overlap of 4×4. Such overlap can usually aid at improving reconstruction quality. The selection of 50% of nonzeros was just a random choice since the results of Figure 2 did not suggest that a specific percentage is particularly beneficial in terms of reconstruction quality.

3.3 Multi-layer Network Architecture

In this section, we extend the linear formulation to MLPs and investigate the performance in deeper structures.

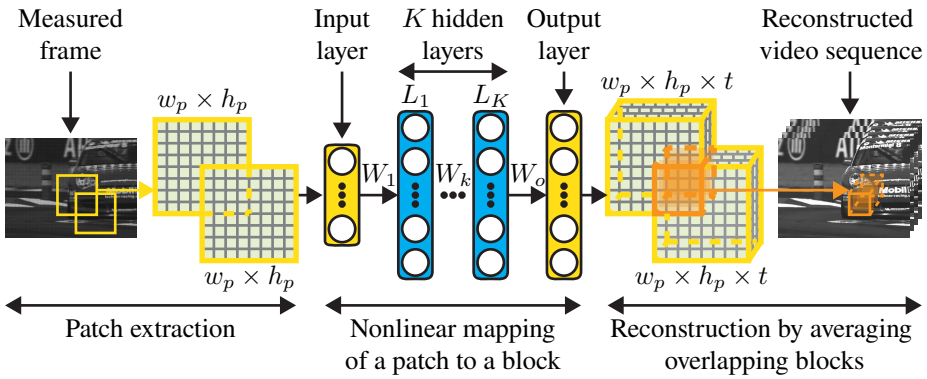

Figure 4: Illustration of the proposed deep learning architecture for video compressive sensing.

Choice of Network Architecture. We consider an end-to-end MLP architecture to learn a nonlinear function $f(\cdot)$ that maps a measured frame patch $\mathbf{y}_i$ via several hidden layers to a video block $\mathbf{x}_i$, as illustrated in Figure 4. The MLP architecture was chosen for the problem of video CS reconstruction due to the following two considerations;

1. The first hidden layer should be a fully-connected layer that would provide a 3D signal from the compressed 2D measurements. This is necessary for temporal video CS as in contrast to the super-resolution problem (or other related image reconstruction problems) where a low-resolution image is given as input, here we are given CS encoded measurements. Thus, convolution does not hold and therefore a convolutional layer cannot be employed as a first layer.
2. Following that, one could argue that the subsequent layers could be 3D Convolutional layers [37]. Although that would sound reasonable for our problem, in practice, the small size of blocks used in this paper ($8 \times 8 \times 16$) do not allow for convolutions to be effective. Increasing the size of blocks to $32 \times 32 \times 16$, so that convolutions can be applied, would dramatically increase the network complexity in 3D volumes such as in videos. For example, if we use a block size of 32×32 as input, the first fully-connected layer would contain $(32 \times 32 \times 16) \times (32 \times 32) = 16,777,216$ parameters! Besides, such small block sizes ($8 \times 8 \times 16$) have provided good reconstruction quality in dictionary learning approaches used for CS video reconstruction [22]. It was shown that choosing larger block sizes led to worse reconstruction quality.

Thus, MLPs (i.e., apply fully-connected layers for the entire network) were considered more reasonable in our work and we found that when applied to $8 \times 8 \times 16$ blocks they capture the motion and spatial details of videos adequately.

It is interesting to note here that another approach would be to try learning the mapping between $\hat{\mathbf{x}}_i = \Phi_p^T \mathbf{y}_i$ and $\mathbf{x}_i$, since matrix Φ_p is known [25]. Such approach could provide better pixel localization since $\Phi_p^T \mathbf{y}$ places the values in $\mathbf{y}$ in the corresponding pixel locations that were sampled to provide the summation in the t direction. However, such an architecture would require additional weights between the input and the first hidden layer since the input would now be of size $(8 \times 8 \times 16)$ instead of (8×8). Such approach was tested and resulted in almost identical performance, albeit with a higher computational cost, hence it is not presented here.

Network Architecture Design. As illustrated in Figure 4, each hidden layer L_k, $k = 1, \ldots, K$ is defined as

$$h_k(\mathbf{y}) = \sigma(\mathbf{b}_k + W_k \mathbf{y}), \tag{4}$$

where $\mathbf{b}_k \in \mathbb{R}^{N_p}$ is the bias vector and W_k is the output weight matrix, containing linear filters. $W_1 \in \mathbb{R}^{N_p \times M_p}$ connects $\mathbf{y}_i$ to the first hidden layer, while for the remaining hidden layers, $W_{2-K} \in \mathbb{R}^{N_p \times N_p}$. The last hidden layer is connected to the output layer via $\mathbf{b}_o \in \mathbb{R}^{N_p}$ and $W_o \in \mathbb{R}^{N_p \times N_p}$ without nonlinearity. The non-linear function $\sigma(\cdot)$ is the rectified linear unit (ReLU) [27] defined as,

Figure 5: Example frames from the video sequences used for training.

$\sigma(y) = \max(0, y)$. In our work we considered two different network architectures, one with $K = 4$ and another with $K = 7$ hidden layers.

To train the proposed MLP, we learn all the weights and biases of the model. The set of parameters is denoted as $\theta = \{\mathbf{b}_{1-K}, \mathbf{b}_o, W_{1-K}, W_o\}$ and is updated by the backpropagation algorithm [33] minimizing the quadratic error between the set of training mapped measurements $f(\mathbf{y}_i; \theta)$ and the corresponding video blocks $\mathbf{x}_i$. The loss function is the mean squared error (MSE) which is given by

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \|f(\mathbf{y}_i; \theta) - \mathbf{x}_i\|_2^2. \tag{5}$$

The MSE was used in this work since our goal is to optimize the PSNR which is directly related to the MSE.

4 Experiments

We compare our proposed deep architecture with state-of-the-art approaches both quantitatively and qualitatively. The proposed approaches are evaluated assuming noiseless measurements or under the presence of measurement noise. Finally, we investigate the performance of our methods under different network parameters (e.g., number of layers) and size of training samples. The metrics used for evaluation were the PSNR and SSIM.

4.1 Training Data Collection

For deep neural networks, increasing the number of training samples is usually synonymous to improved performance. We collected a diverse set of training samples using 400 high-definition videos from Youtube, depicting natural scenes. The video sequences contain more than 10^5 frames which were converted to grayscale. All videos are unrelated to the test set. We randomly extracted 10 million video blocks of size $w_p \times h_p \times t$ while keeping the amount of blocks extracted per video proportional to its duration. This data was used as output while the corresponding input was obtained by multiplying each sample with the measurement matrix Φ_p (see subsection 3.2 for details). Example frames from the video sequences used for training are shown in Figure 5.

4.2 Implementation Details

Our networks were trained for up to 4×10^6 iterations using a mini-batch size of 200. We normalized the input per-feature to zero mean and standard deviation one. The weights of each layer were initialized to random values uniformly distributed in $(-1/\sqrt{s}, 1/\sqrt{s})$, where s is the size of the previous layer [11]. We used Stochastic Gradient Descent (SGD) with a starting learning rate of 0.01, which was divided by 10 after 3×10^6 iterations. The momentum was set to 0.9 and we further used ℓ_2 norm gradient clipping to keep the gradients in a certain range. Gradient clipping is a widely used technique in recurrent neural networks to avoid exploding gradients [29]. The threshold of gradient clipping was set to 10.

4.3 Comparison with Previous Methods

We compare our method with the state-of-the-art video compressive sensing methods:

Table 1: Average performance for the reconstruction of the first 32 frames for 14 video sequences using several methods. Maximum values are highlighted for each side (left/right) of the table. The time (at the bottom row) refers to the average time for reconstructing a sequence of 16 frames using a single captured frame.

		Reconstruction Method					
Video Sequence	Metric	W-10M	FC7-10M	GMM-4 [45]	GMM-1 [44]	FC7-10M +MMLE	GMM-1 +MMLE [44]
Electric Ball	PSNR	40.97	**43.62**	40.16	40.27	**43.81**	41.18
	SSIM	0.9796	**0.9882**	0.9747	0.9754	**0.9885**	0.9802
Horse	PSNR	29.08	**31.65**	29.00	29.20	**31.58**	29.97
	SSIM	0.7869	**0.8586**	0.7747	0.7803	**0.8556**	0.8016
Bow & Arrow	PSNR	35.59	**41.88**	39.27	40.06	**42.96**	41.77
	SSIM	0.9773	**0.9885**	0.9810	0.9838	**0.9902**	0.9881
Bus	PSNR	18.92	**20.10**	19.01	19.20	**20.22**	19.35
	SSIM	0.4583	**0.5316**	0.4640	0.4817	**0.5375**	0.4815
Dogs	PSNR	35.64	**42.40**	38.03	39.29	**43.50**	42.39
	SSIM	0.9712	**0.9889**	0.9739	0.9796	**0.9929**	0.9919
City	PSNR	22.28	**23.29**	22.39	22.55	**23.16**	22.55
	SSIM	0.5127	**0.6279**	0.5196	0.5302	**0.6408**	0.5579
Crew	PSNR	29.74	**32.48**	29.68	29.89	**33.35**	30.42
	SSIM	0.8362	**0.8771**	0.8371	0.8450	**0.8943**	0.8621
Filament	PSNR	42.02	**51.43**	47.95	49.33	**55.03**	52.75
	SSIM	0.9945	**0.9974**	0.9963	0.9965	**0.9989**	0.9988
Hammer	PSNR	31.11	**38.04**	34.45	34.99	**38.59**	36.68
	SSIM	0.9304	**0.9666**	0.9360	0.9423	**0.9696**	0.9553
Football	PSNR	19.84	**21.58**	20.25	20.46	**21.85**	20.80
	SSIM	0.4793	**0.5642**	0.5009	0.5277	**0.5834**	0.5378
Kayak	PSNR	26.49	**30.46**	27.41	27.66	**30.55**	28.74
	SSIM	0.7188	**0.8128**	0.7326	0.7458	**0.8142**	0.7638
Porsche	PSNR	26.17	**29.52**	26.14	26.37	**30.15**	27.45
	SSIM	0.9310	**0.9640**	0.9270	0.9328	**0.9675**	0.9491
Golf	PSNR	26.77	**29.41**	28.30	28.58	**29.89**	29.14
	SSIM	0.9050	**0.9401**	0.9235	0.9319	**0.9507**	0.9440
Basketball	PSNR	22.35	**25.15**	22.80	23.00	**25.53**	23.66
	SSIM	0.6412	**0.7687**	0.6640	0.6868	**0.7860**	0.7156
	Time	$\sim 1s$	$\sim 10s$	$\sim 100s$	$\sim 10^3 s$	$\sim 10^3 - 10^4 s$	$\sim 10^3 - 10^4 s$

- GMM-TP, a Gaussian mixture model (GMM)-based algorithm [45].
- MMLE-GMM, a maximum marginal likelihood estimator (MMLE), that maximizes the likelihood of the GMM of the underlying signals given only their linear compressive measurements [44].

For temporal CS reconstruction, data driven models usually perform better than standard sparsity-based schemes [44, 45]. Indeed, both GMM-TP and MMLE-GMM have demonstrated superior performance compared to existing approaches in the literature such as Total-Variation (TV) or dictionary learning [22, 44, 45], hence we did not include experiments with the latter methods.

In GMM-TP [45] we followed the settings proposed by the authors and used our training data (randomly selecting $20,000$ samples) to train the underlying GMM parameters. We found that our training data provided better performance compared to the data used by the authors. In our experiments we denote this method by GMM-4 to denote reconstruction of overlapping blocks with spatial overlap of 4×4 pixels, as discussed in subsection 3.2.

MMLE [44] is a self-training method but it is sensitive to initialization. A satisfactory performance is obtained only when MMLE is combined with a good starting point. In [44], the GMM-TP [45] with full overlapping patches (denoted in our experiments as GMM-1) was used to initialize the MMLE. We denote the combined method as GMM-1+MMLE. For fairness, we also conducted experiments in the case where our method is used as a starting point for the MMLE.

In our methods, a collection of overlapping patches of size $w_p \times h_p$ is extracted by each coded measurement of size $W_f \times H_f$ and subsequently reconstructed into video blocks of size $w_p \times h_p \times t$. Overlapping areas of the recovered video blocks are then averaged to obtain the final video

reconstruction results, as depicted in Figure 4. The step of the overlapping patches was set to 4×4 due to the special construction of the utilized measurement matrix, as discussed in subsection 3.2.

We consider six different architectures:

- W-10M, a simple linear mapping (equation (3)) trained on 10×10^6 samples.
- FC4-1M, a $K = 4$ MLP trained on 1×10^6 samples (randomly selected from our 10×10^6 samples).
- FC4-10M, a $K = 4$ MLP trained on 10×10^6 samples.
- FC7-1M, a $K = 7$ MLP trained on 1×10^6 samples (randomly selected from our 10×10^6 samples).
- FC7-10M, a $K = 7$ MLP trained on 10×10^6 samples.
- FC7-10M+MMLE, a $K = 7$ MLP trained on 10×10^6 samples which is used as an initialization to the MMLE [44] method.

Note that the subset of randomly selected 1 million samples used for training FC4-1M and FC7-1M was the same.

Our test set consists of 14 video sequences. They involve a set of videos that were used for dictionary training in [22], provided by the authors, as well as the "Basketball" video sequence used by [44]. All video sequences are unrelated to the training set (see subsection 4.1 for details). For fair comparisons, the same measurement mask was used in all methods, according to subsection 3.2. All code implementations are publicly available provided by the authors.

4.4 Reconstruction Results

Quantitative reconstruction results for all video sequences with all tested algorithms are illustrated in Table 1 and average performance is summarized in Figure 7. The presented metrics refer to average performance for the reconstruction of the first 32 frames of each video sequence, using 2 consecutive captured coded frames through the video CS measurement model of equation (1). In both, Table 1 and Figure 7, results are divided in two parts. The first part lists reconstruction performance of the tested approaches without the MMLE step, while the second compares the performance of the best candidate in the proposed and previous methods, respectively, with a subsequent MMLE step [44]. In Table 1 the best performing algorithms are highlighted for each part while the bottom row presents average reconstruction time requirements for the recovery of 16 video frames using 1 captured coded frame.

Our FC7-10M and FC7-10M+MMLE yield the highest PSNR and SSIM values for all video sequences. Specifically, the average PSNR improvement of FC7-10M over the GMM-1 [44] is 2.15 dB. When these two methods are used to initialize the MMLE [44] algorithm, the average PSNR gain of FC7-10M+MMLE over the GMM-1+MMLE [44] is 1.67 dB. Notice also that the FC7-10M achieves 1.01 dB higher than the combined GMM-1+MMLE. The highest PSNR and SSIM values are reported in the FC7-10M+MMLE method with 33.58 dB average PSNR over all test sequences. However, the average reconstruction time for the reconstruction of 16 frames using this method is almost two hours while for the second best, the FC7-10M, is about 12 seconds, with average PSNR 32.93 dB. We conclude that, when time is critical, FC7-10M should be the preferred reconstruction method.

Qualitative results of selected video frames are shown in Figure 6. The proposed MLP architectures, including the linear regression model, favorably recover motion while the additional hidden layers emphasize on improving the spatial resolution of the scene (see supplementary material for example reconstructed videos). One can clearly observe the sharper edges and high frequency details produced by the FC7-10M and FC7-10M+MMLE methods compared to previously proposed algorithms.

Due to the extremely long reconstruction times of previous methods, the results presented in Table 1 and Figure 7 refer to only the first 32 frames of each video sequence, as mentioned above. Figure 8 compares the PSNR for all the frames of 6 video sequences using our FC7-10M algorithm and the fastest previous method GMM-4 [45], while Figure 9 depicts representative snapshots for some of them. The varying PSNR performance across the frames of a 16 frame block is consistent for both algorithms and is reminiscent of the reconstruction tendency observed in other video CS papers in the literature [16, 23, 44, 45].

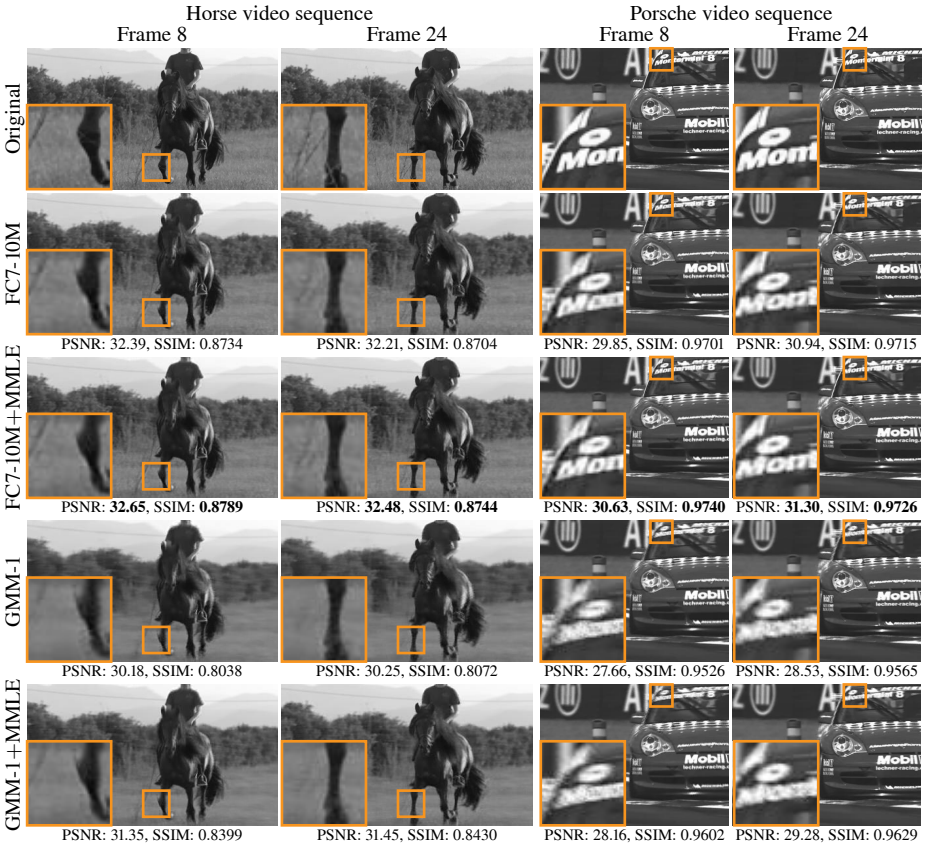

Figure 6: Qualitative reconstruction comparison of frames from two video sequences between our methods and GMM-1 [44], GMM-1+MMLE [44].

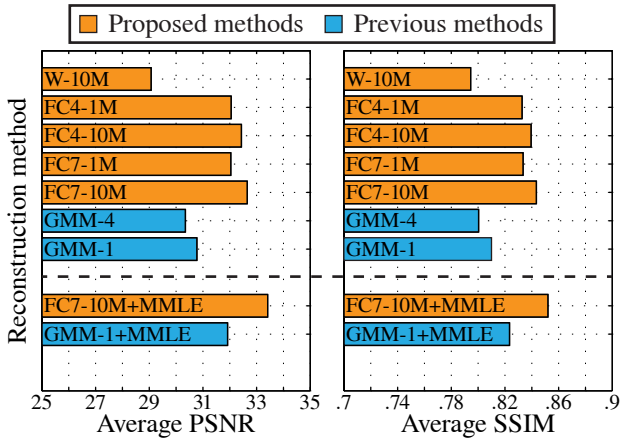

Figure 7: Average PSNR and SSIM over all video sequences for several methods.

4.5 Reconstruction Results with Noise

Previously, we evaluated the proposed algorithms assuming noiseless measurements. In this subsection, we investigate the performance of the presented deep architectures under the presence of measurement noise. Specifically, the measurement model of equation (1) is now modified to

$$\mathbf{y} = \Phi\mathbf{x} + \mathbf{n}, \tag{6}$$

where $\mathbf{n} : M_f \times 1$ is the additive measurement noise vector.

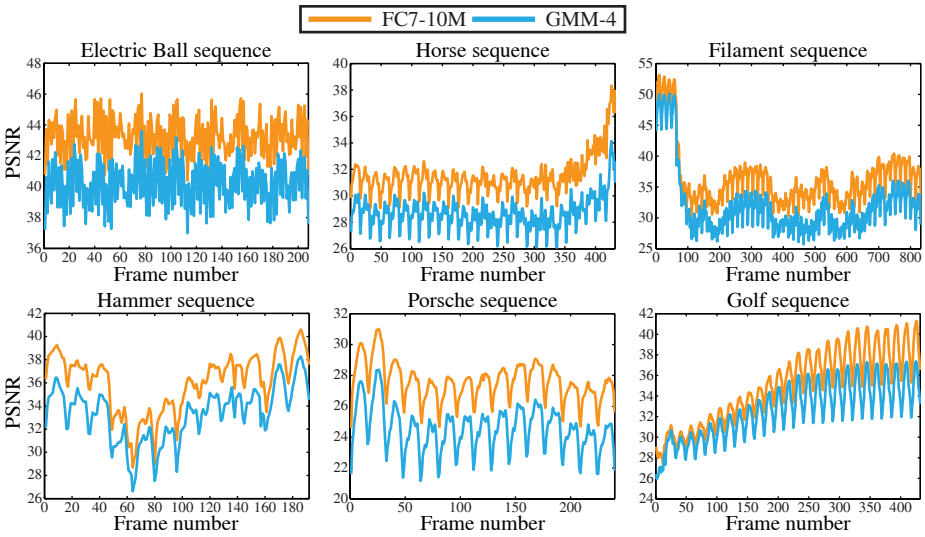

Figure 8: PSNR comparison for all the frames of 6 video sequences between the proposed method FC7-10M and the previous method GGM-4 [45].

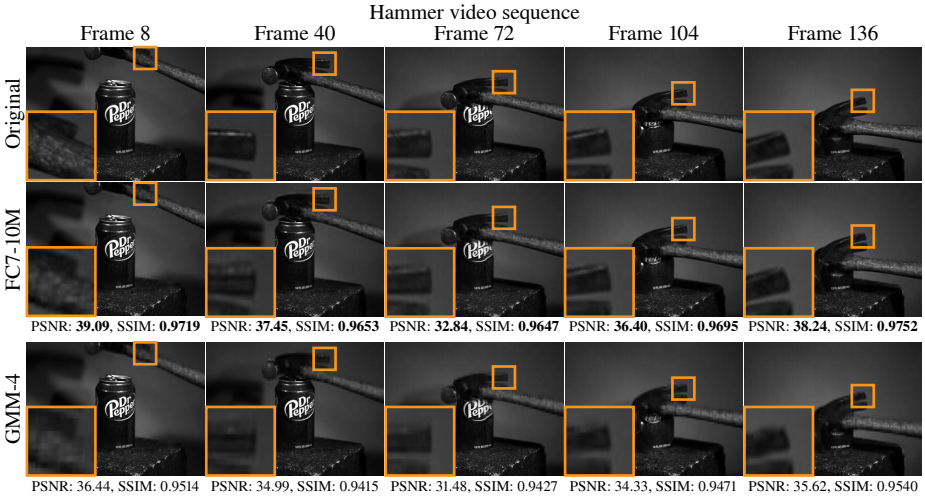

Figure 9: Qualitative reconstruction performance of video frames between the proposed method FC7-10M and the previous method GMM-4 [45]. The corresponding PSNR results for all video frames are shown in Figure 8.

We employ our best architecture utilizing $K = 7$ hidden layers and follow two different training schemes. In the first one, the network is trained on the 10×10^6 samples, as discussed in subsection 4.3 (i.e., the same FC7-10M network as before) while in the second, the network is trained using the same data pairs $\{\mathbf{y}_i, \mathbf{x}_i\}$ after adding random Gaussian noise to each vector $\mathbf{y}_i$. Each vector $\mathbf{y}_i$ was corrupted with a level of noise such that signal-to-noise ratio (SNR) is uniformly selected in the range between $20 - 40$ dB giving rise to a set of 10×10^6 noisy samples for training. We denote the network trained on the noisy dataset as FC7N-10M.

We now compare the performance of the two proposed architectures with the previous methods GMM-4 and GMM-1 using measurement noise. We did not include experiments with the MMLE counterparts of the algorithms since, as we observed earlier, the performance improvement is always related to the starting point of the MMLE algorithm. Figure 10 shows the average performance comparison for the reconstruction of the first 32 frames of each tested video sequence under different levels of measurement noise while Figure 11 depicts example reconstructed frames.

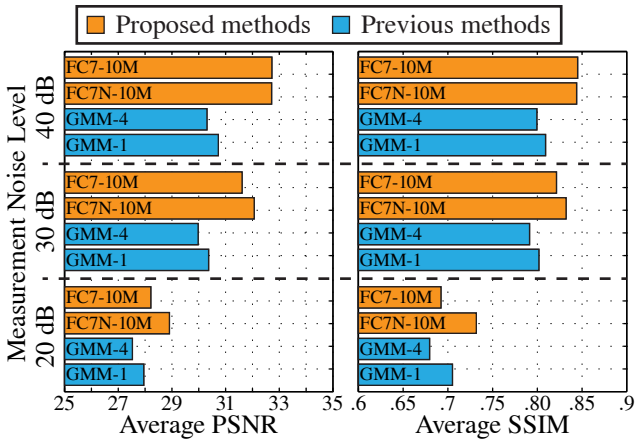

Figure 10: Average PSNR and SSIM over all video sequences for several methods under different levels of measurement noise.

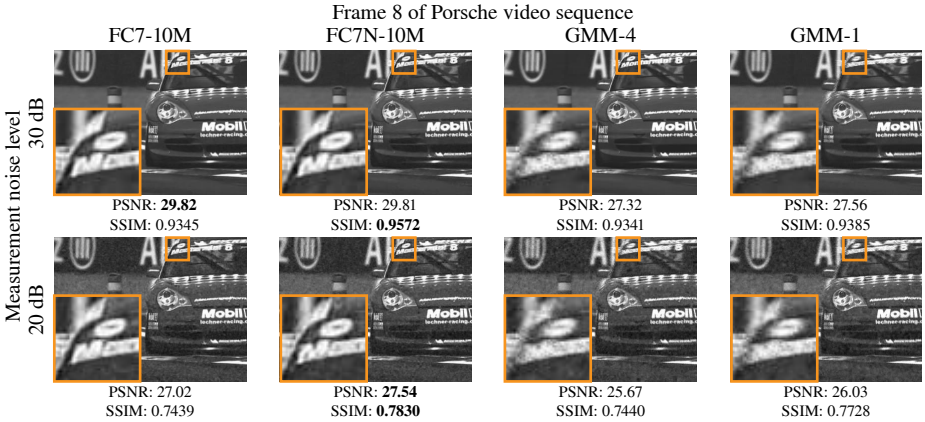

Figure 11: Qualitative reconstruction comparison between our methods and GMM-4 [45], GMM-1 [44] under different levels of measurement noise. The original frame and corresponding inset are presented in Figure 6.

As we can observe, the network trained on noiseless data (FC7-10M) provides good performance for low measurement noise (e.g., 40 dB) and reaches similar performance to GMM-1 for more severe noise levels (e.g., 20 dB). The network trained on noisy data (FC7N-10M), proves more robust to noise severity achieving better performance than GMM-1 under all tested noise levels.

Despite proving more robust to noise, our algorithms in general recover motion favorably but, for high noise levels, there is additive noise throughout the reconstructed scene (observe results for 20 dB noise level in Figure 11). Such degradation could be combated by cascading our architecture with a denoising deep architecture (e.g., [3]) or denoising algorithm to remove the noise artifacts. Ideally, for a specific camera system, data would be collected using this system and trained such that the deep architecture incorporates the noise characteristics of the underlying sensor.

4.6 Run Time

Run time comparisons for several methods are illustrated at the bottom row of Table 1. All previous approaches are implemented in MATLAB. Our deep learning methods are implemented in Caffe package [15] and all algorithms were executed by the same machine. We observe that the deep learning approaches significantly outperform the previous approaches in order of several magnitudes. Note that a direct comparison between the methods is not trivial due to the different implementations. Nevertheless, previous methods solve an optimization problem during reconstruction while our MLP is a feed-forward network that requires only few matrix-vector multiplications.

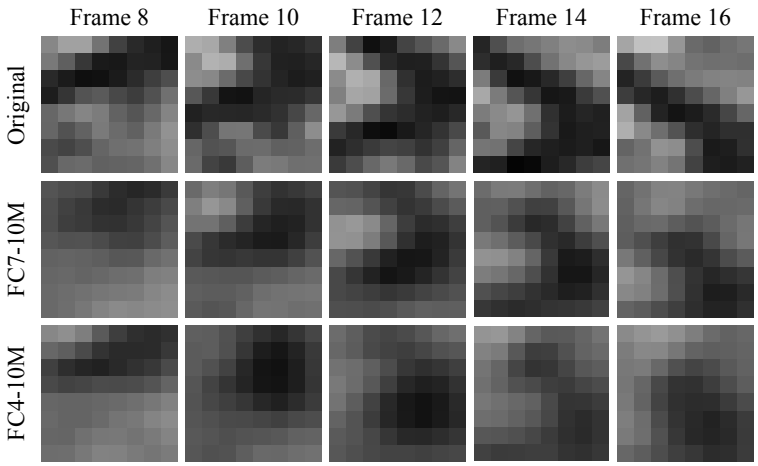

Figure 12: Qualitative reconstruction comparison for a video block of the training set. First row shows 5 patches from the original video block of size $8 \times 8 \times 16$; second row shows the reconstruction using the trained network with 7 hidden layers (FC7-10M); third row shows the reconstruction using the trained network with 4 hidden layers (FC4-10M). The slight improvement in reconstruction quality using network FC7-10M is apparent while the ℓ_2 norm reconstruction error is 3.05 and 4.11 for FC7-10M and FC4-10M, respectively.

4.7 Number of Layers and Dataset Size

From Figure 7 we observe that as the number of training samples increases the performance consistently improves. However, the improvement achieved by increasing the number of layers (from 4 to 7) for architectures trained on small datasets (e.g., 1M) is not significant (performance is almost the same). This is perhaps expected as one may argue that in order to achieve higher performance with extra layers (thus, more parameters to train) more training data would be required. Intuitively, adding hidden layers enables the network to learn more complex functions. Indeed, reconstruction performance in our 10 million dataset is slightly higher in FC7-10M than in FC4-10M. The average PSNR for all test videos is 32.66 dB for FC4-10M and 32.91 dB for FC7-10M. This suggests that 4-hidden layers are sufficient to learn the mappings in our 10M training set. However, we wanted to explore the possible performance benefits of adding extra hidden layers to the network architecture.

In order to provide more insights regarding the slight performance improvement of FC7-10M compared to FC4-10M we visualize in Figure 12 an example video block from our training set and its respective reconstruction using the two networks. We observe that FC7-10M is able to reconstruct the patches of the video block slightly better than FC4-10M. This suggests that the additional parameters help in fitting the training data more accurately. Furthermore, we observed that reconstruction performance of our validation set was better in FC7-10M than in FC4-10M. Note that a small validation set was kept for tuning the hyper-parameters during training and that we also employed weight regularization (ℓ_2 norm) to prevent overfitting. Increasing the number of hidden layers further did not help in our experiments as we did not observe any additional performance improvement based on our validation set. Thus, we found that learning to reconstruct training patches accurately was important in our problem.

5 Conclusions

To the best of our knowledge, this work constitutes the first deep learning architecture for temporal video compressive sensing reconstruction. We demonstrated superior performance compared to existing algorithms while reducing reconstruction time to a few seconds. At the same time, we focused on the applicability of our framework on existing compressive camera architectures suggesting that their commercial use could be viable. We believe that this work can be extended in three directions: 1) exploring the performance of variant architectures such as RNNs, 2) investigate the training of deeper architectures and 3) finally, examine the reconstruction performance in real video sequences acquired by a temporal compressive sensing camera.

References

[1] F. Agostinelli, M. R. Anderson, and H. Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Adv. Neural Inf. Process. Syst. 26*, pages 1493–1501. Curran Associates, Inc., 2013.

[2] S. D. Babacan, M. Luessi, L. Spinoulas, A. K. Katsaggelos, N. Gopalsami, T. Elmer, R. Ahern, S. Liao, and A. Raptis. Compressive passive millimeter-wave imaging. In *IEEE Int. Conf. Image Processing*, pages 2705–2708, Sept 2011.

[3] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 2392–2399, June 2012.

[4] H. Chen, M. S. Asif, A. C. Sankaranarayanan, and A. Veeraraghavan. FPA-CS: Focal plane array-based compressive imaging in short-wave infrared. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 2358–2366, June 2015.

[5] H. Chen, Z. Weng, Y. Liang, C. Lei, F. Xing, M. Chen, and S. Xie. High speed single-pixel imaging via time domain compressive sampling. In *CLEO: 2014*, page JTh2A.132. Optical Society of America, 2014.

[6] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen. Deep network cascade for image super-resolution. In *Computer Vision – ECCV 2014*, volume 8693 of *Lecture Notes in Computer Science*, pages 49–64. Springer International Publishing, 2014.

[7] C. Dong, C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, Feb. 2016.

[8] 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 Process. Mag.*, 25(2):83–91, Mar. 2008.

[9] C. Fernandez-Cull, B. M. Tyrrell, R. D'Onofrio, A. Bolstad, J. Lin, J. W. Little, M. Blackwell, M. Renzi, and M. Kelly. Smart pixel imaging with computational-imaging arrays. In *Proc. SPIE*, volume 9070, pages 90703D–90703D–13, 2014.

[10] L. Gao, J. Liang, C. Li, and L. V. Wang. Single-Shot compressed ultrafast photography at one hundred billion frames per second. *Nature*, 516:74–77, 2014.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proc. Int. Conf. Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 770–778, June 2016.

[13] J. Holloway, A. C. Sankaranarayanan, A. Veeraraghavan, and S. Tambe. Flutter shutter video camera for compressive sensing of videos. In *IEEE Int. Conf. Comp. Photography*, pages 1–9, April 2012.

[14] Y. Huang, W. Wang, and L. Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Adv. Neural Inf. Process. Syst. 28*, pages 235–243. Curran Associates, Inc., 2015.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM Int. Conf. Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.

[16] R. Koller, L. Schmid, N. Matsuda, T. Niederberger, L. Spinoulas, O. Cossairt, G. Schuster, and A. K. Katsaggelos. High spatio-temporal resolution video with compressed sensing. *Opt. Express*, 23(12):15992–16007, June 2015.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Adv. Neural Inf. Process. Syst. 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[18] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok. ReconNet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 449–458, June 2016.

[19] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[20] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec. 1989.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, Nov. 1998.

[22] D. Liu, J. Gu, Y. Hitomi, M. Gupta, T. Mitsunaga, and S. K. Nayar. Efficient space-time sampling with pixel-wise coded exposure for high-speed imaging. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(2):248–260, Feb 2014.

[23] P. Llull, X. Liao, X. Yuan, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. J. Brady. Coded aperture compressive temporal imaging. *Opt. Express*, 21(9):10526–10545, May 2013.

[24] P. Llull, X. Yuan, X. Liao, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. J. Brady. *Temporal Compressive Sensing for Video*, pages 41–74. Springer International Publishing, Cham, 2015.

[25] J. Mehta and A. Majumdar. Rodeo: Robust de-aliasing autoencoder for real-time medical image reconstruction. *Pattern Recognition*, 63:499 – 510, 2017.

[26] A. Mousavi, A. B. Patel, and R. G. Baraniuk. A deep learning approach to structured signal recovery. In *Annual Allerton Conf. Communication, Control, and Computing*, pages 1336–1343, Sept 2015.

[27] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz and T. Joachims, editors, *Proc. Int. Conf. Machine Learning*, pages 807–814. Omnipress, 2010.

[28] G. Orchard, J. Zhang, Y. Suo, M. Dao, D. T. Nguyen, S. Chin, C. Posch, T. D. Tran, and R. Etienne-Cummings. Real time compressive sensing video reconstruction in hardware. *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, 2(3):604–615, Sept. 2012.

[29] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In S. Dasgupta and D. McAllester, editors, *Proc. Int. Conf. Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[30] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *Proc. Int. Conf. Neural Inf. Process. Systems*, NIPS'15, pages 1990–1998, Cambridge, MA, USA, 2015. MIT Press.

[31] D. Reddy, A. Veeraraghavan, and R. Chellappa. P2C2: Programmable pixel compressive camera for high speed imaging. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 329–336, June 2011.

[32] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Adv. Neural Inf. Process. Syst. 28*, pages 901–909. Curran Associates, Inc., 2015.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[34] C. Schuler, H. Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 1067–1074, June 2013.

[35] L. Spinoulas, K. He, O. Cossairt, and A. Katsaggelos. Video compressive sensing with on-chip programmable subsampling. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition Workshops*, pages 49–57, June 2015.

[36] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 769–777, June 2015.

[37] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *IEEE Int. Conf. Computer Vision*, pages 4489–4497, Dec 2015.

[38] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked Denoising Autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, Dec. 2010.

[39] J. Wang, M. Gupta, and A. C. Sankaranarayanan. LiSens- A scalable architecture for video compressive sensing. In *Proc. IEEE Conf. Comp. Photography*, pages 1–9, April 2015.

[40] Z. Wang, A. C. Bovik, H. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, April 2004.

[41] Z. Wang, L. Spinoulas, K. He, L. Tian, O. Cossairt, A. K. Katsaggelos, and H. Chen. Compressive holographic video. *Opt. Express*, 25(1):250–262, Jan 2017.

[42] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Adv. Neural Inf. Process. Syst. 25*, pages 341–349. Curran Associates, Inc., 2012.

[43] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Adv. Neural Inf. Process. Syst. 27*, pages 1790–1798. Curran Associates, Inc., 2014.

[44] J. Yang, X. Liao, X. Yuan, P. Llull, D. J. Brady, G. Sapiro, and L. Carin. Compressive sensing by learning a gaussian mixture model from measurements. *IEEE Trans. Image Processing*, 24(1):106–119, Jan. 2015.

[45] J. Yang, X. Yuan, X. Liao, P. Llull, D. J. Brady, G. Sapiro, and L. Carin. Video compressive sensing using gaussian mixture models. *IEEE Trans. Image Processing*, 23(11):4863–4878, Nov. 2014.