VideoOneNet: Bidirectional Convolutional Recurrent OneNet with Trainable Data Steps for Video Processing

Zoltán Á. Milacski [1] **Barnabás Póczos** [2] **András Lőrincz** [1]

Abstract

Deep Neural Networks (DNNs) achieve the state-of-the-art results on a wide range of image processing tasks, however, the majority of such solutions are problem-specific, like most AI algorithms. The One Network to Solve Them All ('OneNet') procedure has been suggested to resolve this issue by exploiting a DNN as the proximal operator in Alternating Direction Method of Multipliers (ADMM) solvers for various imaging problems. In this work, we make two contributions, both facilitating end-to-end learning using backpropagation. First, we generalize OneNet to videos by augmenting its convolutional prior network with bidirectional recurrent connections; second, we extend the fixed fully connected linear ADMM data step with another trainable bidirectional convolutional recurrent network. In our computational experiments on the Rotated MNIST, Scanned CIFAR-10 and UCF-101 data sets, the proposed modifications improve performance by a large margin compared to end-to-end convolutional OneNet and 3D Wavelet sparsity on several video processing problems: pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting, super resolution, compressive sensing, deblurring, frame interpolation, frame prediction and colorization. Our two contributions are complementary, and using them together yields the best results.

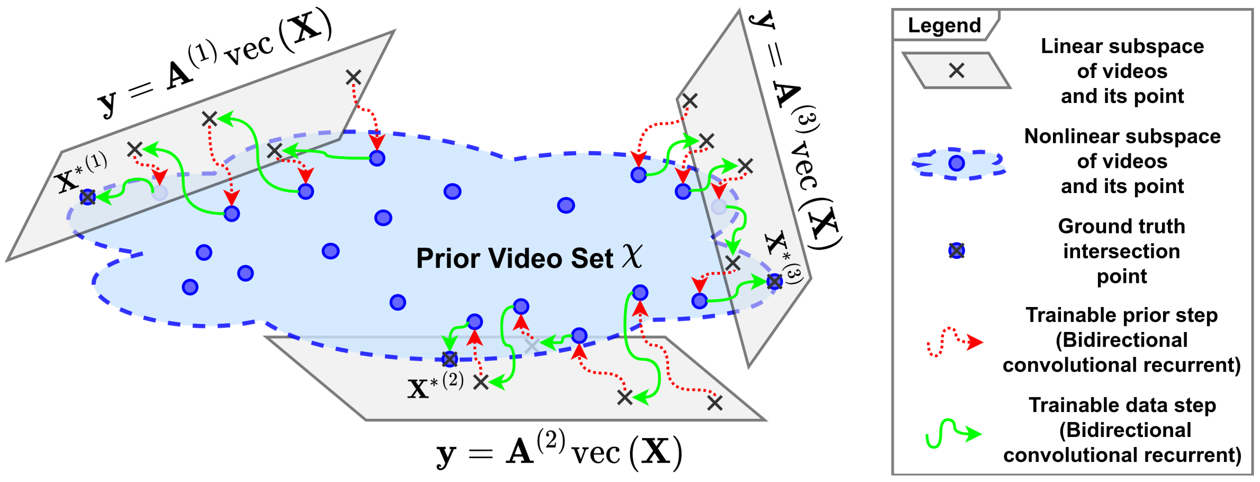
1. Introduction

In previous years, Deep Neural Networks (DNNs, see LeCun et al. (2015) and the references therein) have become the state-of-the-art on many data processing tasks, including image (Krizhevsky et al., 2012; He et al., 2016; 2017; Sabour et al., 2017), text (Peters et al., 2018; Howard & Ruder, 2018; Devlin et al., 2018), speech (Xiong et al., 2018; Chiu et al., 2018) problems, among many others. These results mostly rely on large data sets and applying Supervised Learning or Unsupervised Learning via gradient descent with minimal feature engineering for each task independently. One imminent downside of this practice is the lack of Multi-Task Learning (Ruder, 2017): the regularization effects of feature and weight sharing across the various tasks are ignored, while training individual problem-specific networks from scratch is computationally expensive and time consuming. Recently, the One Network to Solve Them All (OneNet) (Chang et al., 2017) procedure emerged as a possible solution for images, which involves using the same Convolutional Neural Network (CNN) as the prior proximal operator of Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) solvers for various linear inverse restoration problems. The original procedure used a pretrained adversarial denoising autoencoder, but lately, it was extended as an end-to-end trained architecture to prevent ADMM convergence issues (Milacski et al., 2019a).

In contrast to the image case addressed by OneNet, linear inverse problems for videos are more challenging: one has to deal with larger tensors, exploit temporal correlations, while the measurement process can involve bigger and more complex linear operators acting between-frames (e.g., temporal convolutions, dropping frames) or both within- and between-frames (e.g., video compressive sensing (Zheng & Jacobs, 2009)). Therefore it is unlikely that the straightfoward time distributed (i.e., framewise repeated) extension of OneNet may be optimal under such circumstances, due to the lack of temporal propagation. Furthermore, it is also questionable whether the mappings in the OneNet data step, which are kept from the original ADMM algorithm to leverage the measurement and the corresponding linear operator (i.e., the proximal operator of the reconstruction loss and dual ascent), are optimal. Although the ADMM update rules are theoretically motivated, they are still manually engineered, fully-connected and linear, hence they may not fully take advantage of all available features, especially when the mea-

[1]Department of Artificial Intelligence, ELTE Eötvös Loránd University, Budapest, Hungary [2]Machine Learning Department, Carnegie Mellon University, Pittsburgh, USA. Correspondence to: Zoltán Á. Milacski <srph25@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).



$\mathbf{y} = \mathbf{A}^{(1)}\,\mathrm{vec}\,(\mathbf{X})$
$\mathbf{y} = \mathbf{A}^{(2)}\,\mathrm{vec}\,(\mathbf{X})$
$\mathbf{y} = \mathbf{A}^{(3)}\,\mathrm{vec}\,(\mathbf{X})$
$\mathbf{X}^{*(1)}$
$\mathbf{X}^{*(2)}$
$\mathbf{X}^{*(3)}$
Prior Video Set $\mathcal{X}$
Legend
Linear subspace of videos and its point
Nonlinear subspace of videos and its point
Ground truth intersection point
Trainable prior step (Bidirectional convolutional recurrent)
Trainable data step (Bidirectional convolutional recurrent)

Figure 1. Schematic diagram of our proposed method. Best viewed in color. We consider solving linear inverse problems for videos, i. e., recovering ground truth videos $\boldsymbol{X}^{*(n)} \in \mathbb{R}^{T\times H\times W\times F}$ from measurement vectors $\boldsymbol{y}^{(n)} \in \mathbb{R}^{d^{(n)}}$ observed through very different linear operators $\boldsymbol{A}^{(n)} \in \mathbb{R}^{d^{(n)}\times(T\cdot H\cdot W\cdot F)}$, $n = 1, \ldots, N$. As most points in the linear subspaces $\boldsymbol{y} = \boldsymbol{A}^{(n)}\,\mathrm{vec}\,(\boldsymbol{X})$ (gray) are fake videos, we implicitly learn a nonlinear prior video set χ (blue) of real videos that reduces the problems to finding intersection points between the linear subspaces and the prior set. This is achieved by learning two alternating nonlinear projections: the prior step (red dotted arrow) and the data step (green solid arrow). Each sequence of steps is trained end-to-end to approximate $\boldsymbol{X}^{*(n)}$ where both projections are implemented as bidirectional convolutional recurrent networks to improve performance and convergence. The bidirectional convolutional recurrent property of projections is explained in Algorithm 3 and Figure 2.

surement operator acts locally (e.g., spatial and temporal convolution, super resolution) or when the ADMM iteration count is small. This latter problem generalizes beyond the video case.

Contributions. In this work, we test the hypotheses whether convolutional recurrent architectures and learned data steps can improve the performance of the OneNet procedure in the video setting. We start from the time distributed version of OneNet and propose two modifications.

- We augment the convolutional network in the prior step of OneNet with bidirectional convolutional recurrent connections (Liang & Hu, 2015; Xingjian et al., 2015). This facilitates information propagation across video frames.
- We complement the predefined fully connected linear ADMM update rules in the data step by introducing a second DNN on top of their outputs. This technique can be understood as an extended ADMM (falling back to classical ADMM when our second network learns the identity function). Similar to the prior network, we implement this as a Bidirectional Convolutional Recurrent Neural Network (BCRNN), however, with multiple concatenated inputs.

Our scheme is summarized in Figure 1. We empirically compare our method with the end-to-end convolutional OneNet procedure and 3D Wavelet sparsity on the Rotated MNIST, Scanned CIFAR-10 and UCF-101 data sets on an ensemble of video processing tasks: pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting, super resolution, compressive sensing, disk and motion deblurring, frame intepolation, frame prediction and colorization. Furthermore, we also perform an ablation study to test the efficiency of different components of our model. The source code enabling the reproduction of our results is available[1].

2. Theoretical Background and Related Works

2.1. Notation

Throughout the paper, we use the following conventions. A bold face letter refers to a multi-dimensional tensor, a tensor superscript denotes a sample index and a tensor subscript indexes the tensor across the corresponding axes. Following standard machine learning notation, for a 6-dimensional tensor $\boldsymbol{Z} \in \mathbb{R}^{N\times I\times T\times H\times W\times F}$, $z_{h,w,f}^{(n,i,t)} \in \mathbb{R}$ refers to the pixel value of the n^{th} sample video in the i^{th} ADMM iteration for the t^{th} frame at the h^{th} row, w^{th} column and f^{th} feature. N, I, T, H, W, F stands for the total number of videos, ADMM iterations, video frames, frame rows (height), frame columns (width) and features (channels), accordingly. When it does not cause confusion, we refer to (h, w) as pixel indices. For simplicity, we either omit or designate by "dot" $(\cdot)$ all indices across an axis, e.g., for a 6-dimensional tensor we use $\boldsymbol{Z}$, $\boldsymbol{Z}^{(\cdot)}$, $\boldsymbol{Z}^{(\cdot,\cdot)}$, $\boldsymbol{Z}^{(\cdot,\cdot,\cdot)}$, $\boldsymbol{Z}^{(\cdot,\cdot,\cdot)}_{\cdot}$, $\boldsymbol{Z}^{(\cdot,\cdot,\cdot)}_{\cdot,\cdot}$ and $\boldsymbol{Z}^{(\cdot,\cdot,\cdot)}_{\cdot,\cdot,\cdot}$ interchangeably, where the dots indicate the number of indexed

[1]https://github.com/srph25/videoonenet

dimensions being relevant in the expression. Specifying some of the axes allows us to take subtensors. For example, $\boldsymbol{Z}_{h,w}^{(n,i,t)} = \boldsymbol{Z}_{h,w,\cdot}^{(n,i,t)} = \left(z_{h,w,f}^{(n,i,t)}\right)_{f\in\{1,\dots,F\}}$ denotes a one dimensional tensor, i.e., it is a vector. Functions are specified by normal typesetting, where again each "dot" $(\cdot)$ represents the respective argument. $\|\cdot\|_p$ is the ℓ_p norm, $\|\cdot\|_F$ is the Frobenius norm (vectorized ℓ_2 norm), vec stands for the vectorization operator (flattening), $\mathrm{reshape}$ designates the reshaping operator (including the inverse of vec as a special case).

2.2. Linear Inverse Problems and Signal Priors

The goal of a *linear inverse problem* (also called *restoration problem*, see Engl et al. (1996) and the references therein) is to reconstruct a video[2] $\boldsymbol{X}^{*(n)} \in \mathbb{R}^{T\times H\times W\times F}$ from a measurement vector $\boldsymbol{y}^{(n)} \in \mathbb{R}^{d^{(n)}}$ of the form

$$\boldsymbol{y}^{(n)} = \boldsymbol{A}^{(n)} \operatorname{vec}\left(\boldsymbol{X}^{*(n)}\right) + \boldsymbol{\xi}^{(n)}, \tag{1}$$

where $\boldsymbol{A}^{(n)} \in \mathbb{R}^{d^{(n)}\times(T\cdot H\cdot W\cdot F)}$ is the linear measurement operator and $\boldsymbol{\xi}^{(n)} \in \mathbb{R}^{d^{(n)}}$ is the zero-mean noise (e.g., Gaussian). For example, in image inpainting, $\boldsymbol{A}^{(n)}$ is a pixelwise mask operator; in super resolution, $\boldsymbol{A}^{(n)}$ is a downsampling operator averaging nearby pixels; in compressive sensing, $\boldsymbol{A}^{(n)}$ is a random Gaussian matrix mixing multiple frames; in frame prediction and interpolation, $\boldsymbol{A}^{(n)}$ is a framewise mask operator. Notice how $\boldsymbol{A}^{(n)}$ is large, as it acts across all video pixels simultaneously, leading to computational challenges. $\boldsymbol{A}^{(n)}$ is often overcomplete, i.e., the corresponding linear system is underdetermined, thus the null space of $\boldsymbol{A}^{(n)}$ is nontrivial and there are an infinite number of feasible solutions. The index n represents that one may solve many such problems together, where $\boldsymbol{A}^{(n)}$ may vary considerably.

In order to choose a particular solution from the infinitely large feasible set, the reconstruction loss is often regularized with a *signal prior* penalty (Golub et al., 1999):

$$\min_{\boldsymbol{X}^{(n)}} \frac{1}{2}\left\|\boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)}\operatorname{vec}\left(\boldsymbol{X}^{(n)}\right)\right\|_2^2 + \lambda\phi\left(\boldsymbol{X}^{(n)}\right), \tag{2}$$

where $\phi\colon \mathbb{R}^{(T\cdot H\cdot W\cdot F)} \to \mathbb{R}$ is the signal prior and $\lambda \geq 0$. Note that we apply the same ϕ for each sample index n, so we assume that a common latent structure is present for all $\boldsymbol{X}^{(n)}$. Originally, analytically derived, convex, predefined signal priors were widely used as ϕ in the literature. A natural choice is sparsity in some transformation domain, i.e., $\phi\left(\boldsymbol{X}^{(n)}\right) = \left\|\boldsymbol{W}\operatorname{vec}\left(\boldsymbol{X}^{(n)}\right)\right\|_1$, where $\boldsymbol{W}$ is an operator representing either wavelet transformation (Donoho, 1995; Mallat, 1999), spatial gradient (Chan et al., 2006), or some other similar linear operation.

2.3. Alternating Direction Method of Multipliers

The *Alternating Direction Method of Multipliers* (*ADMM*)[3] (Boyd et al., 2011) is an optimization method for solving linear inverse problems in the form of (2). ADMM relies on variable splitting: an auxiliary variable $\boldsymbol{Z}^{(n)}$ is introduced that is constrained to be equal to $\boldsymbol{X}^{(n)}$. This gives us the following optimization problem:

$$\min_{\boldsymbol{X}^{(n)},\boldsymbol{Z}^{(n)}} \frac{1}{2}\left\|\boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)}\operatorname{vec}\left(\boldsymbol{Z}^{(n)}\right)\right\|_2^2 + \lambda\phi\left(\boldsymbol{X}^{(n)}\right)$$
$$\text{s.t.}\quad \boldsymbol{X}^{(n)} = \boldsymbol{Z}^{(n)}, \tag{3}$$

which is equivalent to the original regularized problem (2). The scaled form of the augmented Lagrangian of (3) can be written as

$$L(\boldsymbol{X}^{(n)},\boldsymbol{Z}^{(n)},\boldsymbol{U}^{(n)}) = \frac{1}{2}\left\|\boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)}\operatorname{vec}\left(\boldsymbol{Z}^{(n)}\right)\right\|_2^2 + \lambda\phi\left(\boldsymbol{X}^{(n)}\right) + \frac{\rho^{(n)}}{2}\left\|\boldsymbol{X}^{(n)} - \boldsymbol{Z}^{(n)} + \boldsymbol{U}^{(n)}\right\|_2^2, \tag{4}$$

where $\rho^{(n)} > 0$ is the penalty parameter of the constraint $\boldsymbol{X}^{(n)} = \boldsymbol{Z}^{(n)}$, and $\boldsymbol{U}^{(n)}$ is the tensor of dual variables divided by $\rho^{(n)}$. By alternatively optimizing $L\left(\boldsymbol{X}^{(n)},\boldsymbol{Z}^{(n)},\boldsymbol{U}^{(n)}\right)$ over $\boldsymbol{X}^{(n)}$, $\boldsymbol{Z}^{(n)}$ and $\boldsymbol{U}^{(n)}$, ADMM is composed of the iterations in Algorithm 1.

Algorithm 1 Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011)

```
Input: $\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}, \rho^{(n)}, \lambda, \phi(\cdot)$
  {Initialization}
  $\boldsymbol{Z}^{(n,0,\cdot)} = \arg\min_{\boldsymbol{Z}} \frac{1}{2}\left\|\boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)}\operatorname{vec}(\boldsymbol{Z})\right\|_2^2$    (a)
  $\boldsymbol{U}^{(n,0,\cdot)} = \boldsymbol{0}$    (b)
  {Iterations}
  for $i = 1,\dots,I$ do
    {Prior Step}
      {Prior Proximal Operator}
      $\boldsymbol{X}^{(n,i,\cdot)} = \arg\min_{\boldsymbol{X}} \frac{\rho^{(n)}}{2}\left\|\boldsymbol{X} - \boldsymbol{Z}^{(n,i-1,\cdot)} + \boldsymbol{U}^{(n,i-1,\cdot)}\right\|_2^2$
                $+ \lambda\phi(\boldsymbol{X})$    (c)
    {Data Step}
      {Reconstruction Loss Proximal Operator (Least Squares)}
      $\boldsymbol{Z}^{(n,i,\cdot)} = \arg\min_{\boldsymbol{Z}} \frac{1}{2}\left\|\boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)}\operatorname{vec}(\boldsymbol{Z})\right\|_2^2$
                $+ \frac{\rho^{(n)}}{2}\left\|\boldsymbol{X}^{(n,i,\cdot)} - \boldsymbol{Z} + \boldsymbol{U}^{(n,i-1,\cdot)}\right\|_2^2$    (d)
      {Dual Ascent (Cumulative Summation)}
      $\boldsymbol{U}^{(n,i,\cdot)} = \boldsymbol{U}^{(n,i-1,\cdot)} + \boldsymbol{X}^{(n,i,\cdot)} - \boldsymbol{Z}^{(n,i,\cdot)}$    (e)
  end for
Output: $\boldsymbol{Z}^{(n,I,\cdot)}$
```

[2]Throughout this paper, we present and use the generalization of the $T = 1$ case of linear inverse problems (Chang et al., 2017) for arbitrary T. Yet, we introduced the vec operator for $T > 1$ and the superscript (n) indexing various triplets $\left[\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}, \boldsymbol{X}^{*(n)}\right]$.

[3]Throughout this paper, we present and use the generalization of the $T = 1$ case of ADMM (Boyd et al., 2011) for arbitrary T.

The update (c) of $\boldsymbol{X}^{(n)}$ in Algorithm 1 is the proximal operator of the signal prior ϕ with penalty $\frac{\rho^{(n)}}{\lambda}$, denoted as $\operatorname{prox}_{\phi,\,\rho^{(n)}/\lambda}\left(\boldsymbol{Z}^{(n,i-1,\cdot)} - \boldsymbol{U}^{(n,i-1,\cdot)}\right)$. When the signal prior uses ℓ_1-norm, this is simply a soft-thresholding. The update (d) of $\boldsymbol{Z}^{(n)}$ is the proximal operator of the reconstruction loss, which is a least squares problem; while the update (e) of $\boldsymbol{U}^{(n)}$ is dual ascent, a cumulative summation resembling residual skip connections (He et al., 2016).

Notice that the ADMM algorithm separates the signal prior ϕ from the linear operators $\boldsymbol{A}^{(n)}$, $n = 1, \ldots, N$. This enables learning a signal prior that can be used with many linear operators.

2.4. One Network to Solve Them All and Unrolled Optimization

Chang et al. (2017) argue that in each iteration i, the input $\left(\boldsymbol{Z}^{(n,i-1,\cdot)} - \boldsymbol{U}^{(n,i-1,\cdot)}\right)$ to the prior proximal operator resembles a noisy video, which is ideally mapped to a noiseless video. Consequently, the authors proposed to pretrain an adversarial denoising convolutional autoencoder $p_{\text{CNN}}(\boldsymbol{\theta}, \cdot)$ (Makhzani et al., 2015) and use it during inference phase in ADMM in place of the prior proximal operator as in Algorithm 2 (c).

Algorithm 2 One Network to Solve Them All (OneNet) (Chang et al., 2017; Milacski et al., 2019a)

```
Input: $\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}, \rho^{(n)}, p_{\text{CNN}}(\boldsymbol{\theta}, \cdot)$
  {Initialization}
  (a)-(b) of Algorithm 1
  {Iterations}
  for $i = 1, \ldots, I$ do
    {Prior Step}
      {Deep Time Distributed Convolutional Prior Network}
      $\boldsymbol{X}^{(n,i,\cdot)} = p_{\text{CNN}}\left(\boldsymbol{\theta}, \boldsymbol{Z}^{(n,i-1,\cdot)} - \boldsymbol{U}^{(n,i-1,\cdot)}\right)$        (c)
    {Data Step}
      (d)-(e) of Algorithm 1
  end for
Output: $\boldsymbol{Z}^{(n,I,\cdot)}$
```

In other words, Chang et al. (2017) suggested a modified ADMM with a learned (but pretrained) nonconvex prior ϕ. They call their framework *One Network to Solve Them All* (*OneNet*)[4] as they are using the same pretrained network $p_{\text{CNN}}(\boldsymbol{\theta}, \cdot)$ with multiple linear inverse problem matrices $\boldsymbol{A}^{(n)}$ (compressive sensing, pixelwise inpainting-denoising, scattered inpainting, blockwise inpainting and super resolution). They showed either significantly improved or on par performance while also being less prone to changes in $\boldsymbol{A}^{(n)}$ and $\boldsymbol{\xi}^{(n)}$ compared to 2D Wavelet sparsity and more common problem-specific Context Encoder networks (Pathak et al., 2016), depending on the task. As their prior is trained separately from the linear operators, it generalizes well to new problems without retraining, however, it also underfits those observed during testing phase.

Recently, OneNet was enhanced by end-to-end training (Milacski et al., 2019a). In the original OneNet procedure, the network is trained independently from ADMM, consequently updates are suboptimal and thus many (up to 300) ADMM iterations are required for convergence, preventing real-time applications. Therefore, Milacski et al. (2019a) adopted the *Unrolled Optimization with Deep Priors* (Diamond et al., 2017) framework based on Deep Unfolding (Hershey et al., 2014), in which they truncate a classical iterative optimization algorithm (e.g., Proximal Gradient (Chen et al., 2015), Iterative Shrinkage Thresholding (Beck & Teboulle, 2009; Gregor & LeCun, 2010) and even ADMM (Boyd et al., 2011; Sun et al., 2016)) and propose using deep convolutional prior architectures $p_{\text{CNN}}(\boldsymbol{\theta}, \cdot)$ within the unrolled optimization, facilitating end-to-end backpropagation. They also leveraged Self-Supervised Multi-Task Learning (Doersch & Zisserman, 2017; Jing & Tian, 2020): given a target video $\boldsymbol{X}^{*(n)}$, they randomly pick a linear operator $\boldsymbol{A}^{(n)}$ from a prescribed problem set and automatically generate the measurement $\boldsymbol{y}^{(n)}$ via (1); then use $\boldsymbol{A}^{(n)}$ and $\boldsymbol{y}^{(n)}$ as inputs to Algorithm 2 for computing the approximate inverse $\boldsymbol{Z}^{(n,I,\cdot)}$. They have shown greatly improved results in terms of Peak Signal-to-Noise Ratio (PSNR) score while also requiring much fewer ADMM iterations ($13 \ll 300$), due to the stabilizing effect of end-to-end training on convergence. As they tailor their prior to specific linear operators, it works much better for those, but does not generalize well to new problems without retraining anymore.

Simultaneously with the original OneNet paper, similar procedures using pretrained deep prior objectives have been developed for other variable splitting solvers, including Half Quadratic Splitting (HQS) (Zhang et al., 2017), Primal-Dual Hybrid Gradient (PDHG) (Meinhardt et al., 2017) and Proximal Gradient Descent (PGD) (Shah & Hegde, 2018). Specifically, Meinhardt et al. (2017) proves the equivalence of these with ADMM. Hence, we focus on OneNet.

2.5. Deep Neural Network Solvers for Specific Linear Inverse Problems

Related to our method, there are several DNNs in the literature that can solve individual linear inverse problems, even if the particular linear operator $\boldsymbol{A}^{(n)} = \boldsymbol{A}$ for $n = 1, \ldots, N$ is not constructed explicitly as in (1). Hereby, we refer the reader to the following short collection of papers. See e.g., for images: pixelwise inpainting-denoising (Vincent et al., 2008; Srivastava, 2013), blockwise and scattered inpainting (Pathak et al., 2016), super resolution (Dong et al., 2015; Ledig et al., 2017), compressive sensing (Kulkarni et al., 2016; Bora et al., 2017; Mousavi & Baraniuk, 2017), deblurring (Xu et al., 2014; Kupyn et al., 2018) and colorization (Zhang et al., 2016); as well as for videos: inpainting (Kim

[4]Throughout this paper, we present and use the generalization of the $T = 1$ case of OneNet (Chang et al., 2017) for arbitrary T.

et al., 2019), compressive sensing (Iliadis et al., 2018; Xu & Ren, 2018), deblurring (Su et al., 2017; Zhang et al., 2018), frame interpolation (Niklaus et al., 2017; Jiang et al., 2018), frame prediction (Mathieu et al., 2015; Lotter et al., 2016; Finn et al., 2016) and colorization (Thasarathan et al., 2019). These techniques are tailored to their specific problems, hence they are unable to deal with others without retraining. Training networks for each case separately ignores the regularization induced by sharing parameters between similar tasks, while it is also wasteful from the point of view of computational power and storage.

In contrast, ADMM with predefined signal priors supports new tasks by altering the matrix $\boldsymbol{A}^{(n)}$, however, it is less efficient. Considering the linear inverse scenario, OneNet is the golden mean between the two extremes, as it is trainable, and works well on multiple problems.

2.6. Multi-Task Learning

Multi-Task Learning (see the overview of Ruder (2017)) relies on sharing weights of DNNs between related problems to improve generalization capabilities on individual ones. *Hard parameter sharing* (Caruana, 1997) employs separate models for problems with tied weights at middle layers and untied weights at bottom and top layers, facilitating distributed training across tasks (Doersch & Zisserman, 2017). *Soft parameter sharing* (Argyriou et al., 2008; Duong et al., 2015) uses completely untied weights, which are promoted to be similar with regularization. In both cases, tasks can be quite general, as they are implicitly predefined by input-target pairs, yet adapting to additional problems requires retraining.

Our setup operates with hard parameter sharing: we use separate ADMM models, where network weights are tied, and only the linear operator $\boldsymbol{A}^{(n)}$ is untied for different problems, which is not a trainable parameter but part of the input. Therefore, our scheme is trainable on infinitely many different tasks by randomizing $\boldsymbol{A}^{(n)}$. However, our technique is not expected to generalize well to new problems.

2.7. Self-Supervised Learning

Self-Supervised Learning (Jing & Tian, 2020) is a type of Unsupervised Learning, where one uses automated generation of input-target pairs (e.g., producing one from the other, or both from some 3rd variable) in order to leverage techniques from Supervised Learning. It is often employed in Transfer Learning (Weiss et al., 2016) scenarios: *source tasks* (called *pretext tasks* in the self-supervised setting) are used for pretraining before performing actual supervised *target tasks* with human annotated data.

Our scheme is an instance of Self-Supervised Learning, as we generate the inputs automatically: we first pick the target $\boldsymbol{X}^{*(n)}$ from the training set, then sample $\boldsymbol{A}^{(n)}$ randomly, to finally compute $\boldsymbol{y}^{(n)}$ from those using (1). However, in this work we focus on the linear inverse problems instead of transferring knowledge to later supervised tasks.

3. Methods

3.1. Proposed Solution

In this section, we describe our two contributions to OneNet (Section 2.4): the generalization to videos via the convolutional recurrent prior network, and the trainable data step. Our method is summarized in Algorithm 3.

Algorithm 3 VideoOneNet: Bidirectional Convolutional Recurrent OneNet with Trainable Data Steps for Videos

```
Input: $\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}, \rho^{(n)}, p_{\mathrm{BCRNN}}(\boldsymbol{\theta}, \cdot), q_{\mathrm{BCRNN}}(\boldsymbol{\omega}, \cdot)$
  {Initialization}
  (a)-(b) of Algorithm 1
  $\tilde{\boldsymbol{Z}}^{(n,0,\cdot)} = \mathrm{reshape}\left(\boldsymbol{A}^{(n)T}\boldsymbol{y}^{(n)}, (1,1,T,H,W,F)\right)$    (ã)
  {Iterations}
  for $i = 1, \ldots, I$ do
    {Prior Step}
      {Deep Bidirectional Convolutional Recurrent Prior Network}
      $\boldsymbol{X}^{(n,i,\cdot)} = p_{\mathrm{BCRNN}}\left(\boldsymbol{\theta}, \left[\boldsymbol{Z}^{(n,i-1,\cdot)}, \boldsymbol{U}^{(n,i-1,\cdot)}\right]\right)$    (c)
    {Data Step}
      (d)-(e) of Algorithm 1
      {Deep Bidirectional Convolutional Recurrent Data Network}
      $\boldsymbol{C}^{(n,i,\cdot)} = \left[\boldsymbol{X}^{(n,i,\cdot)}, \boldsymbol{Z}^{(n,0,\cdot)}, \tilde{\boldsymbol{Z}}^{(n,0,\cdot)}, \boldsymbol{Z}^{(n,i-1,\cdot)}, \boldsymbol{Z}^{(n,i,\cdot)},$
                    $\boldsymbol{U}^{(n,i-1,\cdot)}, \boldsymbol{U}^{(n,i,\cdot)}\right]$
      $\left[\boldsymbol{Z}^{(n,i,\cdot)}, \boldsymbol{U}^{(n,i,\cdot)}\right] = q_{\mathrm{BCRNN}}\left(\boldsymbol{\omega}, \boldsymbol{C}^{(n,i,\cdot)}\right)$    (f)
  end for
Output: $\boldsymbol{Z}^{(n,I,\cdot)}$
```

First, we replace $p_{\mathrm{CNN}}(\boldsymbol{\theta}, \cdot)$ with a bidirectional convolutional recurrent network $p_{\mathrm{BCRNN}}(\boldsymbol{\theta}, \cdot)$ (Liang & Hu, 2015; Xingjian et al., 2015) in (c). This is a standard architecture that can leverage temporal information between frames via local spatio-temporal connectivity. This is a rather straightforward extension of OneNet to videos. One notable difference is in the input to the network: instead of employing $\left(\boldsymbol{Z}^{(n,i-1,\cdot)} - \boldsymbol{U}^{(n,i-1,\cdot)}\right)$, we concatenate the two parts across channels, i.e., use the more general $\left[\boldsymbol{Z}^{(n,i-1,\cdot)}, \boldsymbol{U}^{(n,i-1,\cdot)}\right]$.

Second, we extend ADMM by adding a second bidirectional convolutional recurrent network denoted by $q_{\mathrm{BCRNN}}(\boldsymbol{\omega}, \cdot)$ on top of the original operations within the data step in (f). In other words, we aim to improve upon both the proximal operator of the reconstruction loss and dual ascent. This is also reminiscent of OneNet, as we use a trainable architecture instead of a fixed, analytically derived, linear one, in a vein similar to the learned prior step. We precompute and reshape the matrix $\boldsymbol{A}^{(n)T}\boldsymbol{y}^{(n)}$ in (ã), denoted by $\tilde{\boldsymbol{Z}}^{(n,0,\cdot)}$. Then again, we use the concatenation $\Big[\boldsymbol{X}^{(n,i,\cdot)}, \boldsymbol{Z}^{(n,0,\cdot)}, \tilde{\boldsymbol{Z}}^{(n,0,\cdot)}, \boldsymbol{Z}^{(n,i-1,\cdot)}, \boldsymbol{Z}^{(n,i,\cdot)}, \boldsymbol{U}^{(n,i-1,\cdot)}, \boldsymbol{U}^{(n,i,\cdot)}\Big]$ across channels as input to the network in order to efficiently

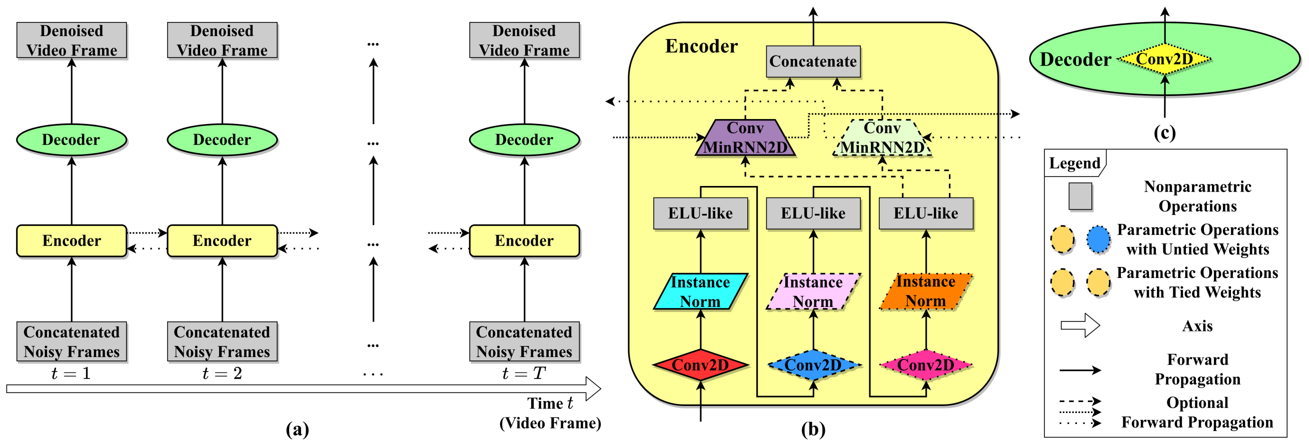


Figure 2. Schematic diagram of our proposed Deep Neural Network architecture used for both prior and data step networks. Best viewed in color, same color and border means tied weights. (a) Base architecture. It processes concatenated noisy input video frames using the (optionally bidirectional convolutional recurrent) encoder and the decoder in order to output denoised video frames. (b) Details of our encoder. It consists of 3 convolutional layers each followed by instance normalization and ELU-like smooth activation function. Optionally there can be a bidirectional convolutional recurrent mechanism added on top that can propagate temporal information between frames. (c) Details of our decoder. It is a single linear transposed convolutional layer.

utilize information about the measurement $\boldsymbol{y}^{(n)}$ and the linear operator $\boldsymbol{A}^{(n)}$.[5] Making the data step nonlinear and trainable should improve performance and convergence.

Network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are trained end-to-end to minimize the mean squared error (MSE) between the final output $\boldsymbol{Z}^{(n,I,\cdot)}$ and the ground truth solution $\boldsymbol{X}^{*(n)}$ for each training sample:

$$\min_{\boldsymbol{\theta},\boldsymbol{\omega}} \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{X}^{*(n)} - \boldsymbol{Z}^{(n,I,\cdot)}\right\|_F^2. \qquad (5)$$

We also adopt the self-supervised sample generation of end-to-end OneNet in Section 2.4 and Section 2.7 for Algorithm 3.

3.2. Experimental Setup

We followed the experimental setups of Chang et al. (2017) and Milacski et al. (2019a). Data sets were taken from Santana et al. (2017). Hyperparameters were chosen according to the increased hardware requirements. We compared the performance of our VideoOneNet with 3D Wavelet sparsity (Wakin et al., 2006) and end-to-end convolutional OneNet (Milacski et al., 2019a). We also performed an ablation study to quantify the individual effects of our bidirectional convolutional recurrent prior step and our learned data step.

Our linear inverse problem set consisted of various dense and sparse linear operators $\boldsymbol{A}^{(n)}$: pixelwise inpainting-denoising (PID; randomly replacing 50% of the pixels by zeros and adding Gaussian noise with 0.1 standard deviation), blockwise inpainting (BI; replacing the center block of side length 40% with zeros), scattered inpainting (SI; randomly replacing 10 blocks of side length 20% with zeros), super resolution (SR; downsampling to side length 50% or 25% using box-averaging), 4D spatio-temporal compressive sensing (CS; to size 10% by multiplying the vector of all RGB channel values with an overcomplete random Gaussian matrix), video compressive sensing (VCS; the initial frame is given and frame differences are spatially compressed to 10% (Zheng & Jacobs, 2009)), 2D spatial and 1D temporal disk deblurring (DD and TDD; filter size 4×4 and 4 with radius 2, accordingly), 2D spatial motion deblurring (MD; filter size 7×7), frame interpolation (FI; estimating the rest from every 2nd or 4th frame), frame prediction (FP; from the initial 75%, 50% and 25% of frames) and frame colorization (C; from grayscale). For details, see the cited papers in Section 2.5.

We set the ADMM iteration count $I = 13$ and the least squares regularization constant $\rho^{(n)} = 0.3$ for all cases. All least squares problems were implemented by precomputing the Moore – Penrose pseudoinverses of the corresponding matrices.

Our prior and data network architectures are summarized in Figure 2. For both networks we used identical setups: we

[5]In principle, one could directly feed $\boldsymbol{y}^{(n)}$ and $\boldsymbol{A}^{(n)}$ as inputs to the data network $q_{\mathrm{BCRNN}}(\boldsymbol{\omega}, \cdot)$. Although, at first sight, this seems appealing, the varying row size $d^{(n)}$ hampers implementation. Thus, we concatenate variables of fixed size $N \times 1 \times T \times H \times W \times F$ in each iteration. This way we treat $\boldsymbol{y}^{(n)}$ and $\boldsymbol{A}^{(n)}$ as latent variables, but it is easy to realize.

employed 3 time distributed convolutional 2D layers and optionally 1 bidirectional convolutional MinimalRNN 2D (Chen, 2017; Milacski et al., 2019b) layer in the encoder, with a single linear time distributed transposed convolutional 2D decoder layer. All encoder convolutional layers were followed by instance normalization (Ulyanov et al., 2016) and exponential linear-like (ELU-like) smooth $\frac{\text{softplus}(2x+2)}{2} - 1$ activation (Gulrajani et al., 2017). We used Adam optimizer (Kingma & Ba, 2014) with learning rate 10^{-4}, $\beta_1 = 0$ and $\beta_2 = 0.9$. Due to potential training instability problems via exploding gradients, we employed gradient clipping to norm 1 (Pascanu et al., 2013). All methods were trained over 50 epochs with early stopping patience 5.

Each training/validation/test split was defined by partitioning the ground truth video set. All methods were ensured to train and validate using the exact same randomly sampled $\left(\left[\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}\right], \boldsymbol{X}^{*(n)}\right)$ triplets (except for early stopping). Following related papers, we employed the PSNR score as the evaluation metric, which is basically MSE on reverse logarithmic scale (being even more sensitive to large errors): a unit higher PSNR means an order of magnitude better MSE . A PSNR value > 25 is typically acceptable. Test set PSNR scores were computed for each problem separately, to see how performance varied.

We implemented our scheme and the baseline procedures in Python 3.7 using NumPy 1.18.1 and Keras 2.2.4 (Chollet et al., 2015) with Tensorflow 1.14.1 (Abadi et al., 2015) backend.

Regarding hardware requirements, we note that all discussed methods compute 6D tensors $\boldsymbol{X}$, $\boldsymbol{Z}$, $\boldsymbol{U}$ for each batch using both DNNs and massive pseudoinverses, which demand large computational power and memory. While the algorithms may be scalable in the cloud in their current form, our intention was to achieve good performance, without addressing computational issues. For now, to fit the 6D tensors into the accessible GPUs ($2 \times 24\,\text{GB}$), we restricted ourselves to smaller problems: we specifically picked data sets with limited sizes, and for each of those, our batch size N, frame count T and frame size $H \times W$ were chosen to be as big as possible for execution. We employed single-precision format (FP32).[6] For details, see Section 3.3.

3.3. Data Sets

We evaluated our method and the baseline procedures on two semi-toy problems, namely, Rotated MNIST and Scanned CIFAR-10 (Santana et al., 2017); and on the real video data set UCF-101[7] (Soomro et al., 2012).

[6]We tried automatic mixed precision training, but memory gains were marginal as only few operations were converted to half-precision format (FP16).

[7]https://www.crcv.ucf.edu/data/UCF101.php

For Rotated MNIST, we rotated images of size 28×28 by $40°$ steps starting from a random angle, and cropping the top left $H \times W = 14 \times 14$ pixels, resulting in videos of $T = 9$ frames. We left out 6,000 randomly selected training videos for validation. All convolutional 2D layers consisted of 64 filters with size 3 in the encoder and size 11 in the decoder. We applied $N = 8$ batch size. This experiment was executed on Tesla K80 GPUs with 12 GB VRAM.

For Scanned CIFAR-10, we took the $H \times W = 16 \times 16$ corners of the 32×32 images, creating $T = 4$ frame videos. We held out 5,000 randomly selected training videos for validation. Each convolutional 2D layer had 256 filters with size 5 in the encoder and size 7 in the decoder. We set $N = 4$ batch size. This study was carried out using Tesla V100 GPUs with 16 GB VRAM.

UCF-101 consists of 25 groups of videos of people performing one of 101 actions. Following the official data split 1, we employed groups $\{8, \dots, 22\}$ for training, $\{23, 24, 25\}$ for validation and $\{1, \dots, 7\}$ for testing; resulting in 7,953, 1,584 and 3,783 videos, respectively for the 3 sets. We randomly sampled $T = 4$ consecutive frames resized to $H \times W = 32 \times 32$ from $N = 2$ videos in each batch. We used the same network setup from Scanned CIFAR-10. Due to the increased memory requirements, this analysis was performed on 2 Tesla P40 GPUs with 24 GB VRAM each, exploiting data parallelism.

4. Results and Discussion

Table 1 shows our quantitative results. Table 2 depicts the corresponding qualitative figures for illustrative purposes.

Overall, one can see both quantitatively and qualitatively that our proposed VideoOneNet using both of our novelties performed the best on all data sets. By visual inspection, our method generates semantically correct, sharper videos compared to the two baselines schemes. 3D Wavelet sparsity only performed well on deblurring, while it completely failed on every other task. End-to-end convolutional ADMM OneNet produced appealing results on spatial domain problems, but broke down on frame interpolation and prediction. Our scheme worked successfully in the aforementioned cases.

Regarding the ablation study, both of our modifications boosted performance on their own. Recurrent connections attained huge gains in temporal domain, thanks to the ability of propagating information across the frames. The trainable data step gave the biggest improvements in deblurring. This was because the local spatial connections suited the inversion of 2D convolution, in contrast to a predefined dense layer. Yet, both add-ons were able to give minor boosts on other tasks, too. When combined together, all problems observed even further gains, many of which became quite

Table 1. Mean and population standard deviation Peak Signal-to-Noise Ratio (PSNR) scores computed over test samples on the Rotated MNIST, Scanned CIFAR-10 and UCF-101 data sets. Reverse logarithmic scale (higher is much better). PID, BI, SI, SR, (V)CS, (T)DD, MD, FI, FP and C stand for pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting, super resolution, (video) compressive sensing, (temporal) disk deblurring, motion deblurring, frame interpolation, frame prediction and colorization tasks, respectively (see Section 3.2 for detailed description). An ablation study is included showing the individual effects of our contributions. Standard deviations are large, due to the reverse logarithmic scale and the differences in the difficulties of the test samples. The winning numbers and methods are highlighted in bold.

	Rotated MNIST ($10{,}000 \times 9 \times 14 \times 14 \times 1$)														
	PID	BI	SI	SR $2\times$	SR $(14/3)\times$	CS	VCS	DD	TDD	MD	FI $2.25\times$	FI $4.5\times$	FP $(9/7)\times$	FP $1.8\times$	FP $3\times$
3D Wavelet Sparsity (2006)	13.0±3.6	21.6±4.6	20.0±4.5	17.1±4.1	10.3±3.2	10.1±3.2	10.7±3.3	24.2±4.9	24.5±4.9	29.5±5.4	12.8±3.6	11.3±3.4	17.0±4.1	13.8±3.7	12.0±3.5
OneNet CNN (2019a)	23.4±4.8	30.3±5.5	32.4±5.7	27.4±5.2	16.4±4.1	24.2±4.9	22.1±4.7	28.7±5.4	28.9±5.4	32.4±5.7	15.2±3.9	13.6±3.7	19.2±4.4	16.1±4.0	14.3±3.8
OneNet BCRNN (ours)	23.9±4.9	32.0±5.7	32.7±5.7	27.6±5.3	17.8±4.2	25.5±5.0	25.0±5.0	28.9±5.4	29.4±5.4	32.5±5.7	22.2±4.7	17.1±4.1	26.6±5.2	20.3±4.5	17.0±4.1
VideoOneNet CNN (ours)	23.0±4.8	30.0±5.5	31.8±5.6	26.9±5.2	16.5±4.1	24.0±4.9	22.9±4.8	31.4±5.6	32.8±5.7	34.4±5.9	15.4±3.9	13.8±3.7	19.3±4.4	16.2±4.0	14.4±3.8
VideoOneNet BCRNN (ours)	**26.3**±5.1	**36.1**±6.0	**35.0**±5.9	**29.7**±5.4	**21.8**±4.7	**27.9**±5.3	**27.7**±5.3	**39.9**±6.3	**39.0**±6.2	**41.5**±6.4	**28.4**±5.3	**20.2**±4.5	**31.2**±5.6	**22.3**±4.7	**18.0**±4.2

	Scanned CIFAR-10 ($10{,}000 \times 4 \times 16 \times 16 \times 3$)														
	PID	BI	SI	SR $2\times$	SR $4\times$	CS	VCS	DD	TDD	MD	FI $2\times$	FP $(4/3)\times$	FP $2\times$	FP $4\times$	C
3D Wavelet Sparsity (2006)	8.6±2.9	14.3±3.8	11.6±3.4	13.9±3.7	6.2±2.5	5.7±2.4	7.2±2.7	20.0±4.5	19.6±4.4	24.3±4.9	8.8±3.0	11.7±3.4	8.9±3.0	7.2±2.7	7.2±2.7
OneNet CNN (2019a)	24.2±4.9	27.4±5.2	26.1±5.1	25.2±5.0	19.6±4.4	21.8±4.7	19.6±4.4	27.3±5.2	26.4±5.1	31.6±5.6	16.9±4.1	20.0±4.5	16.8±4.1	14.7±3.8	23.1±4.8
OneNet BCRNN (ours)	24.9±5.0	28.8±5.4	26.8±5.2	26.0±5.1	20.4±4.5	22.9±4.8	20.5±4.5	29.1±5.4	28.5±5.3	34.3±5.9	18.7±4.3	22.4±4.7	18.5±4.3	15.7±4.0	24.1±4.9
VideoOneNet CNN (ours)	25.0±5.0	**29.9**±5.5	**28.4**±5.3	**27.2**±5.2	**21.5**±4.6	23.8±4.9	21.5±4.6	38.2±6.2	36.8±6.1	39.1±6.3	17.2±4.1	20.9±4.6	17.1±4.1	14.9±3.9	**24.6**±5.0
VideoOneNet BCRNN (ours)	**25.1**±5.0	**29.9**±5.5	**28.4**±5.3	**27.2**±5.2	**21.5**±4.6	**24.0**±4.9	**21.9**±4.7	**38.9**±6.2	**37.8**±6.2	**40.5**±6.4	**19.3**±4.4	**22.8**±4.8	**18.6**±4.3	**15.9**±4.0	24.4±4.9

	UCF-101 ($3{,}783 \times 4 \times 32 \times 32 \times 3$)														
	PID	BI	SI	SR $2\times$	SR $4\times$	CS	VCS	DD	TDD	MD	FI $2\times$	FP $(4/3)\times$	FP $2\times$	FP $4\times$	C
3D Wavelet Sparsity (2006)	10.1±3.2	15.4±3.9	12.5±3.5	14.6±3.8	7.6±2.7	7.1±2.7	8.6±2.9	22.7±4.8	20.2±4.5	25.6±5.1	10.3±3.2	13.2±3.6	10.3±3.2	8.5±2.9	8.7±2.9
OneNet CNN (2019a)	24.2±4.9	23.1±4.8	22.3±4.7	23.7±4.9	18.3±4.3	22.3±4.7	18.5±4.3	25.6±5.1	27.7±5.3	28.2±5.3	16.5±4.1	19.2±4.4	16.5±4.1	14.7±3.8	22.0±4.7
OneNet BCRNN (ours)	26.4±5.1	25.2±5.0	27.0±5.2	26.7±5.2	20.6±4.5	24.7±5.0	30.4±5.5	29.0±5.4	33.9±5.8	33.7±5.8	33.8±5.8	35.4±5.9	32.4±5.7	30.2±5.5	24.1±4.9
VideoOneNet CNN (ours)	23.2±4.8	25.1±5.0	24.1±4.9	26.4±5.1	21.6±4.6	20.7±4.6	20.6±4.5	31.0±5.6	32.3±5.7	32.4±5.7	17.8±4.2	20.7±4.5	17.8±4.2	16.0±4.0	24.4±4.9
VideoOneNet BCRNN (ours)	**27.3**±5.2	**26.4**±5.1	**31.2**±5.6	**27.5**±5.2	**22.1**±4.7	**27.1**±5.2	**31.6**±5.6	**38.6**±6.2	**40.2**±6.3	**41.0**±6.4	**35.5**±6.0	**36.9**±6.1	**33.4**±5.8	**30.7**±5.5	**24.7**±5.0

significant.

It is important to note that there is a minor exception to the trend on Scanned CIFAR-10 colorization. Our experimental setup allows methods to greedily adapt to certain tasks, at the expense of sacrificing others, which is exactly what happened in that unique case.

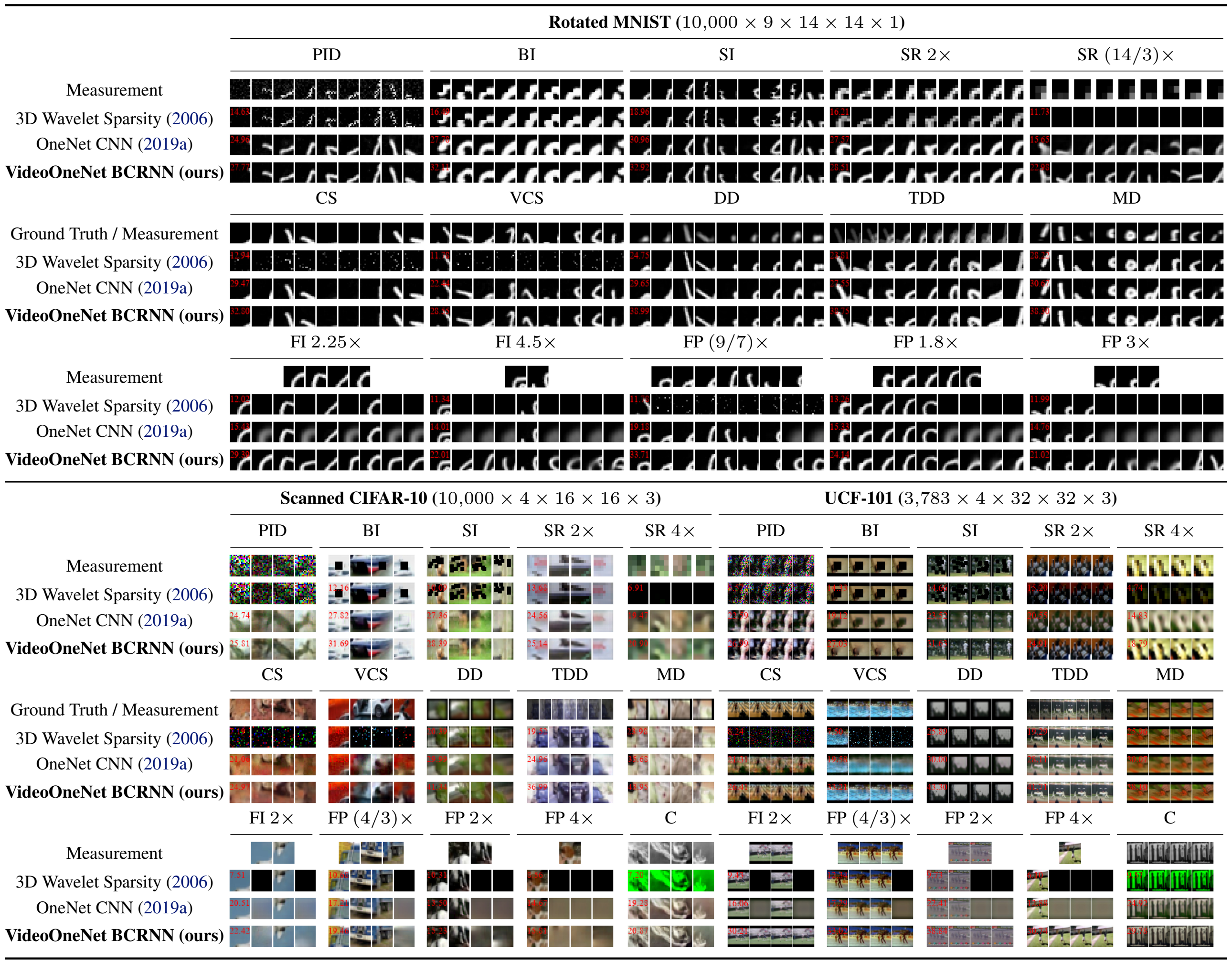
5. Conclusions

We proposed an improved OneNet framework for video data, exploiting a convolutional recurrent prior network and a learned nonlinear data step. Considering pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting, super resolution, compressive sensing, disk and motion deblurring, frame interpolation, frame prediction and colorization, our method significantly outperformed the more recent end-to-end trainable variant of the convolutional OneNet procedure and 3D Wavelet sparsity in the new video setting, even though the baselines already give reasonable results (both quantitatively and qualitatively) on our tasks. The best results were achieved when our two contributions were used together.

Our studies have been limited and certain possibilities for further extensions need to be mentioned. We list some possible directions. First, extending the scope of this paper to larger sample and frame sizes is important for real-world applications, calling for more advanced hardware. However, more carefully designed lightweight variants of our approach (e.g., using fewer iterations, or avoiding least squares) may save memory and computation. Second, some of our generated videos are blurry. This is due to the MSE objective failing to capture uncertainty. Fortunately, adversarial training procedures (Goodfellow et al., 2014) could readily solve this problem. Third, our prior and data networks both try to denoise frames by learning identity-like mappings. This suggests that a residual architecture (He et al., 2016) could enhance performance, since it includes a hard-coded identity. Fourth, Meta-Learning (Vanschoren, 2018), i.e., estimating the optimal $\boldsymbol{\theta}^{(n)}$ and $\boldsymbol{\omega}^{(n)}$ specifically for each task with a DNN from few samples, may prevent overfitting to a fixed set of problems. Finally, as we work with many self-supervised tasks jointly, Transfer Learning to supervised tasks (e.g., classification) may be worth investigating (see Section 2.7).

Our method is a step towards achieving Multi-Task Learning exploiting video data. It has relevance when storing and running pretrained weights of several problem specific DNNs is infeasible (e.g., when processing camera recordings on mobile devices and in robots or self-driving cars): instead, one can simply construct appropriate linear operators by a computer program on-the-fly and deploy our two networks. We believe that our trainable data step idea may gain some attention outside the video processing domain, too, as it

Table 2. Sample qualitative and quantitative experimental test results for the Rotated MNIST, Scanned CIFAR-10 and UCF-101 data sets. Best viewed zoomed in and in color. PID, BI, SI, SR, (V)CS, (T)DD, MD, FI, FP and C stand for pixelwise inpainting-denoising, blockwise inpainting, scattered inpainting, super resolution, (video) compressive sensing, (temporal) disk deblurring, motion deblurring, frame interpolation, frame prediction and colorization tasks, respectively (see Section 3.2 for detailed description). Since the measurement of CS and VCS can not be visualized, we show the ground truth instead. The Peak Signal-to-Noise Ratio (PSNR) values against the ground truth are shown in top of the first frame with red color for each predicted video. The winning methods are highlighted in bold.


Rotated MNIST ($10{,}000 \times 9 \times 14 \times 14 \times 1$)
PID
BI
SI
SR 2×
SR (14/3)×
Measurement
3D Wavelet Sparsity (2006)
OneNet CNN (2019a)
VideoOneNet BCRNN (ours)
CS
VCS
DD
TDD
MD
Ground Truth / Measurement
3D Wavelet Sparsity (2006)
OneNet CNN (2019a)
VideoOneNet BCRNN (ours)
FI 2.25×
FI 4.5×
FP (9/7)×
FP 1.8×
FP 3×
Measurement
3D Wavelet Sparsity (2006)
OneNet CNN (2019a)
VideoOneNet BCRNN (ours)
Scanned CIFAR-10 ($10{,}000 \times 4 \times 16 \times 16 \times 3$)
UCF-101 ($3{,}783 \times 4 \times 32 \times 32 \times 3$)
PID
BI
SI
SR 2×
SR 4×
PID
BI
SI
SR 2×
SR 4×
Measurement
3D Wavelet Sparsity (2006)
OneNet CNN (2019a)
VideoOneNet BCRNN (ours)
CS
VCS
DD
TDD
MD
CS
VCS
DD
TDD
MD
Ground Truth / Measurement
3D Wavelet Sparsity (2006)
OneNet CNN (2019a)
VideoOneNet BCRNN (ours)
FI 2×
FP (4/3)×
FP 2×
FP 4×
C
FI 2×
FP (4/3)×
FP 2×
FP 4×
C
Measurement
3D Wavelet Sparsity (2006)
OneNet CNN (2019a)
VideoOneNet BCRNN (ours)

solves a more general issue of ADMM.

Acknowledgments

This work was completed in the ELTE Institutional Excellence Program supported by the National Research, Development and Innovation Office (NKFIH-1157-8/2019-DT).

We also want to thank Balázs Pintér, Ellák Somfai and the anonymous reviewers for their highly valuable feedback, allowing us to considerably improve this manuscript.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/, 2015.

Argyriou, A., Evgeniou, T., and Pontil, M. Convex Multi-Task Feature Learning. *Machine Learning*, 73(3):243–272, 2008.

Beck, A. and Teboulle, M. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. Compressed Sensing using Generative Models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 537–546. JMLR.org, 2017.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Caruana, R. Multitask Learning. *Machine Learning*, 28(1): 41–75, 1997.

Chan, T. F., Shen, J., and Zhou, H.-M. Total Variation Wavelet Inpainting. *Journal of Mathematical Imaging and Vision*, 25(1):107–125, 2006.

Chang, J., Li, C.-L., Póczos, B., Vijaya Kumar, B., and Sankaranarayanan, A. C. One Network to Solve Them All–Solving Linear Inverse Problems Using Deep Projection Models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5888–5897, 2017.

Chen, M. MinimalRNN: Toward More Interpretable and Trainable Recurrent Neural Networks. *arXiv preprint arXiv:1711.06788*, 2017.

Chen, Y., Yu, W., and Pock, T. On Learning Optimized Reaction Diffusion Processes for Effective Image Restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5261–5269, 2015.

Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, E., et al. State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778. IEEE, 2018.

Chollet, F. et al. Keras. https://keras.io, 2015.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Diamond, S., Sitzmann, V., Heide, F., and Wetzstein, G. Unrolled Optimization with Deep Priors. *arXiv preprint arXiv:1705.08041*, 2017.

Doersch, C. and Zisserman, A. Multi-Task Self-Supervised Visual Learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2051–2060, 2017.

Dong, C., Loy, C. C., He, K., and Tang, X. Image Super-Resolution using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2015.

Donoho, D. L. De-Noising by Soft-Thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

Duong, L., Cohn, T., Bird, S., and Cook, P. Low Resource Dependency Parsing: Cross-Lingual Parameter Sharing in a Neural Network Parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 845–850, 2015.

Engl, H. W., Hanke, M., and Neubauer, A. *Regularization of Inverse Problems*, volume 375. Springer Science & Business Media, 1996.

Finn, C., Goodfellow, I., and Levine, S. Unsupervised Learning for Physical Interaction Through Video Prediction. In *Advances in Neural Information Processing Systems*, pp. 64–72, 2016.

Golub, G. H., Hansen, P. C., and O'Leary, D. P. Tikhonov Regularization and Total Least Squares. *SIAM Journal on Matrix Analysis and Applications*, 21(1):185–194, 1999.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Gregor, K. and LeCun, Y. Learning Fast Approximations of Sparse Coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 399–406. Omnipress, 2010.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

Hershey, J. R., Roux, J. L., and Weninger, F. Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures. *arXiv preprint arXiv:1409.2574*, 2014.

Howard, J. and Ruder, S. Universal Language Model Fine-Tuning for Text Classification. *arXiv preprint arXiv:1801.06146*, 2018.

Iliadis, M., Spinoulas, L., and Katsaggelos, A. K. Deep Fully-Connected Networks for Video Compressive Sensing. *Digital Signal Processing*, 72:9–18, 2018.

Jiang, H., Sun, D., Jampani, V., Yang, M.-H., Learned-Miller, E., and Kautz, J. Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9000–9008, 2018.

Jing, L. and Tian, Y. Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Kim, D., Woo, S., Lee, J.-Y., and Kweon, I. S. Deep Video Inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5792–5801, 2019.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Kulkarni, K., Lohit, S., Turaga, P., Kerviche, R., and Ashok, A. ReconNet: Non-Iterative Reconstruction of Images from Compressively Sensed Measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 449–458, 2016.

Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., and Matas, J. DeblurGAN: Blind Motion Deblurring using Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8183–8192, 2018.

LeCun, Y., Bengio, Y., and Hinton, G. Deep Learning. *Nature*, 521(7553):436, 2015.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-Realistic Single Image Super-Resolution using a Generative Adversarial Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4681–4690, 2017.

Liang, M. and Hu, X. Recurrent Convolutional Neural Network for Object Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3367–3375, 2015.

Lotter, W., Kreiman, G., and Cox, D. Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. *arXiv preprint arXiv:1605.08104*, 2016.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial Autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Mallat, S. *A Wavelet Tour of Signal Processing*. Elsevier, 1999.

Mathieu, M., Couprie, C., and LeCun, Y. Deep Multi-Scale Video Prediction Beyond Mean Square Error. *arXiv preprint arXiv:1511.05440*, 2015.

Meinhardt, T., Moller, M., Hazirbas, C., and Cremers, D. Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1781–1790, 2017.

Milacski, Z. Á., Póczos, B., and Lőrincz, A. Differentiable Unrolled Alternating Direction Method of Multipliers for OneNet. In *30th British Machine Vision Conference (BMVC)*. BMVA, 2019a.

Milacski, Z. Á., Póczos, B., and Lőrincz, A. Group k-Sparse Temporal Convolutional Neural Networks: Unsupervised Pretraining for Video Classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. IEEE, 2019b.

Mousavi, A. and Baraniuk, R. G. Learning to Invert: Signal Recovery via Deep Convolutional Networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2272–2276. IEEE, 2017.

Niklaus, S., Mai, L., and Liu, F. Video Frame Interpolation via Adaptive Separable Convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 261–270, 2017.

Pascanu, R., Mikolov, T., and Bengio, Y. On the Difficulty of Training Recurrent Neural Networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. Context Encoders: Feature Learning by Inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep Contextualized Word Representations. *arXiv preprint arXiv:1802.05365*, 2018.

Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*, 2017.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems*, pp. 3856–3866, 2017.

Santana, E., Emigh, M. S., Zegers, P., and Principe, J. C. Exploiting Spatio-Temporal Structure with Recurrent Winner-Take-All Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3738–3746, 2017.

Shah, V. and Hegde, C. Solving Linear Inverse Problems Using GAN Priors: An Algorithm with Provable Guarantees. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4609–4613. IEEE, 2018.

Soomro, K., Zamir, A. R., and Shah, M. UCF101: A Dataset of 101 Human Actions Classes from Videos in The Wild. *arXiv preprint arXiv:1212.0402*, 2012.

Srivastava, N. Improving Neural Networks with Dropout. *University of Toronto*, 182(566):7, 2013.

Su, S., Delbracio, M., Wang, J., Sapiro, G., Heidrich, W., and Wang, O. Deep Video Deblurring for Hand-Held Cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1279–1288, 2017.

Sun, J., Li, H., Xu, Z., et al. Deep ADMM-Net for Compressive Sensing MRI. In *Advances in Neural Information Processing Systems*, pp. 10–18, 2016.

Thasarathan, H., Nazeri, K., and Ebrahimi, M. Automatic Temporally Coherent Video Colorization. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pp. 189–194. IEEE, 2019.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Vanschoren, J. Meta-Learning: A Survey. *arXiv preprint arXiv:1810.03548*, 2018.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103, 2008.

Wakin, M., Laska, J. N., Duarte, M. F., Baron, D., Sarvotham, S., Takhar, D., Kelly, K. F., and Baraniuk, R. G. Compressive Imaging for Video Representation and Coding. In *Picture Coding Symposium*, volume 1, 2006.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. A Survey of Transfer Learning. *Journal of Big Data*, 3(1):9, 2016.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, pp. 802–810, 2015.

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., and Stolcke, A. The Microsoft 2017 Conversational Speech Recognition System. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5934–5938. IEEE, 2018.

Xu, K. and Ren, F. CSVideoNet: A Real-Time End-to-End Learning Framework for High-Frame-Rate Video Compressive Sensing. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1680–1688. IEEE, 2018.

Xu, L., Ren, J. S., Liu, C., and Jia, J. Deep Convolutional Neural Network for Image Deconvolution. In *Advances in Neural Information Processing Systems*, pp. 1790–1798, 2014.

Zhang, K., Zuo, W., Gu, S., and Zhang, L. Learning Deep CNN Denoiser Prior for Image Restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3929–3938, 2017.

Zhang, K., Luo, W., Zhong, Y., Ma, L., Liu, W., and Li, H. Adversarial Spatio-Temporal Learning for Video Deblurring. *IEEE Transactions on Image Processing*, 28(1): 291–301, 2018.

Zhang, R., Isola, P., and Efros, A. A. Colorful Image Colorization. In *European Conference on Computer Vision*, pp. 649–666. Springer, 2016.

Zheng, J. and Jacobs, E. L. Video Compressive Sensing using Spatial Domain Sparsity. *Optical Engineering*, 48 (8):087006, 2009.