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

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

Zoltán Ádám Milacski*, Barnabás Póczos**, András Lőrincz*

* Department of Artificial Intelligence,
ELTE Eötvös Loránd University,
Budapest, Hungary
** Machine Learning Department,
Carnegie Mellon University,
Pittsburgh, USA

Spotlight

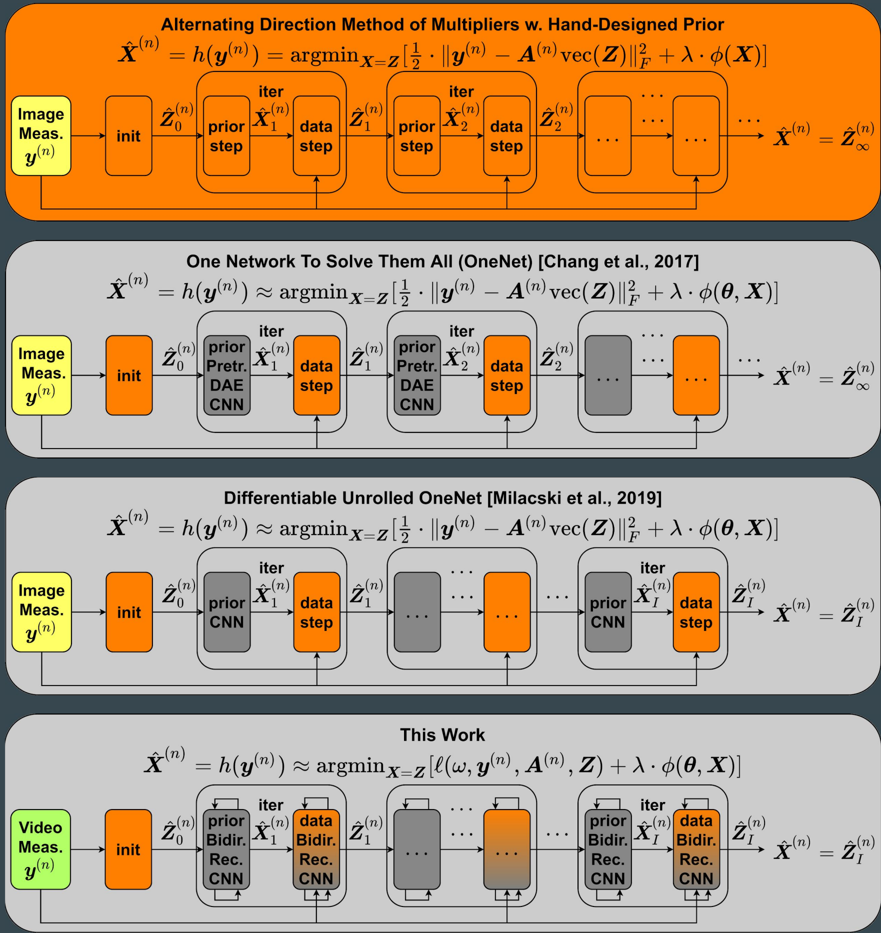
Linear Video Restoration Problem:

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

Reconstruct a video $\boldsymbol{X}^{*\,(n)}$ from $\boldsymbol{y}^{(n)}$.
Various measurement operators $\boldsymbol{A}^{(n)}$!
Single DNN (Self-Supervised Multi-Task Learn.)!

Code: https://github.com/srph25/videoonenet
Email: srph25@gmail.com

Related Works & This Work:



The Challenge: Linear Video Restoration Problem

Reconstruct a video $\boldsymbol{X}^{*(n)} \in \mathbb{R}^{T \times H \times W \times F}$ from $\boldsymbol{y}^{(n)} \in \mathbb{R}^{d^{(n)}}$ of the form:

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

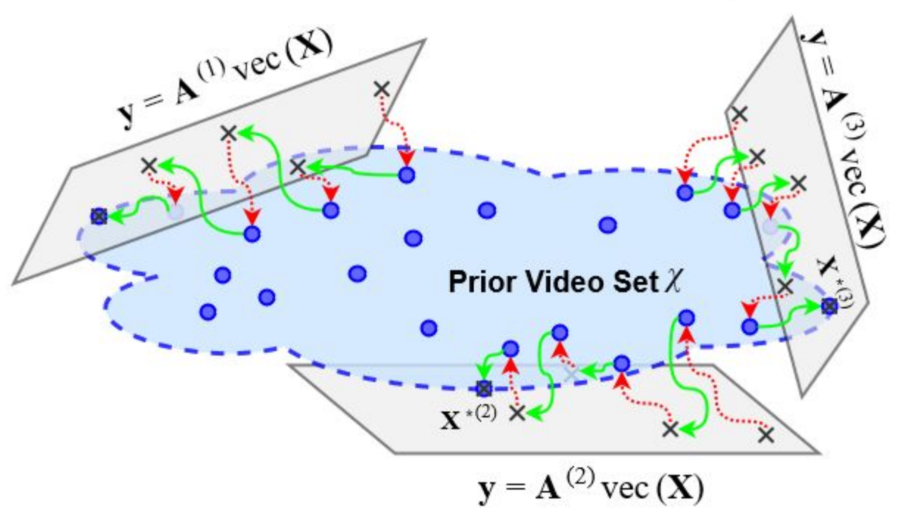
- $\boldsymbol{A}^{(n)}$ is often overcomplete (more columns than rows)...
- The corresponding linear system is underdetermined...
- The null space of $\boldsymbol{A}^{(n)}$ is nontrivial...
- There are an infinite number of feasible solutions...

- ... How to recover $\boldsymbol{X}^{*(n)}$?

Solution: Regularization with Signal Prior

Regularization with a **signal prior** penalty $\phi: \mathbb{R}^{(T \cdot H \cdot W \cdot F)} \rightarrow \mathbb{R}$ (promote intersection points with a prior set of real videos):

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



The Challenge 2.0

- ... How to recover $\boldsymbol{X}^{*\,(n)}$?

$\Leftrightarrow$

- ... How to choose ϕ ?

Solution 1: Sparsity in Hand-Designed Transformation Space

$$\phi\left(\boldsymbol{X}^{(n)}\right) = \left\|\boldsymbol{W}\,\mathrm{vec}\left(\boldsymbol{X}^{(n)}\right)\right\|_1$$

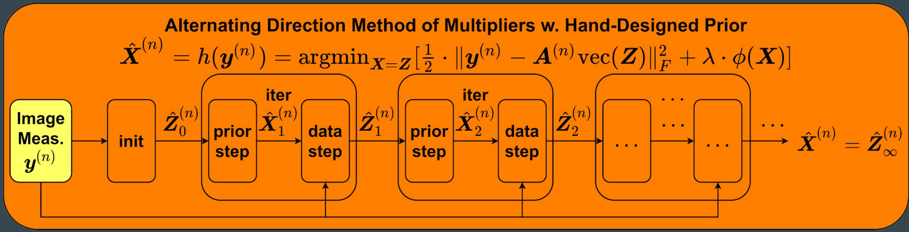
- ℓ_1 norm promotes sparsity
- ... How to choose $\boldsymbol{W}$?
  - Wavelet Transform
    - 2D for images
    - 3D for videos
  - Spatial Gradient (a.k.a. Total Variation)
- All of these are theoretically motivated, but the penalty is low for many fake images/videos, too...
- Poor empirical performance

Solution 1.5: Alternating Direction Method of Multipliers

- Apply ADMM [Boyd et al., 2011] (dual ascent with approx. evaluated robustified dual function) to

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

- The Proximal operator of ϕ appears in (c).
- This is separated from $\boldsymbol{A}^{(n)}$, which appears only in (d)...
- The input of the prior proximal operator is a noisy image/video...



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

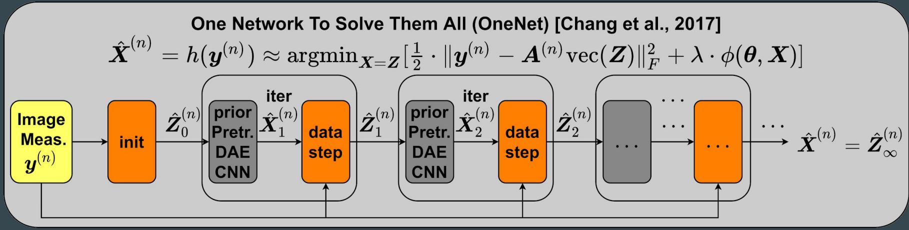
Input: $\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}, \rho^{(n)}, \lambda, \phi(\cdot)$
{Initialization}
$\boldsymbol{Z}^{(n,0,\cdot)} = \arg\min_{\boldsymbol{Z}} \frac{1}{2}\left\|\boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)}\,\mathrm{vec}\,(\boldsymbol{Z})\right\|_2^2$ (a)
$\boldsymbol{U}^{(n,0,\cdot)} = \boldsymbol{0}$ (b)
{Iterations}
for $i = 1, \ldots, I$ **do**
{Prior Step}
{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}
{Least Squares}
$\boldsymbol{Z}^{(n,i,\cdot)} = \arg\min_{\boldsymbol{Z}} \frac{1}{2}\left\|\boldsymbol{y}^{(n)} - \boldsymbol{A}^{(n)}\,\mathrm{vec}\,(\boldsymbol{Z})\right\|_2^2 + \frac{\rho^{(n)}}{2}\left\|\boldsymbol{X}^{(n,i,\cdot)} - \boldsymbol{Z} + \boldsymbol{U}^{(n,i-1,\cdot)}\right\|_2^2$ (d)
{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: $\boldsymbol{Z}^{(n,I,\cdot)}$

Solution 2: OneNet... Learned Prior! [Chang et al., 2017]

- Apply ADMM [Boyd et al., 2011] (dual ascent with approx. evaluated robustified dual function) to

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

- The Proximal operator of ϕ appears in (c).
- This is separated from $\boldsymbol{A}^{(n)}$, which appears only in (d)...
- The input of the prior proximal operator is a noisy image/video...
- ... Plug in a pretrained adversarial denoising autoencoder instead!
- Competitive with problem-specific Self-Supervised Context Encoders [Pathak et al., 2016]
- Very slow: suboptimal denoising due to separate training from ADMM... $I \approx 300$ iterations



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}
$$\boldsymbol{X}^{(n,i,\cdot)} = p_{\text{CNN}}\left(\boldsymbol{\theta}, \boldsymbol{Z}^{(n,i-1,\cdot)} - \boldsymbol{U}^{(n,i-1,\cdot)}\right) \quad \text{(c)}$$
{Data Step}
(d)-(e) of Algorithm 1
end for
Output: $\boldsymbol{Z}^{(n,I,\cdot)}$

Solution 3: OneNet… End-to-End! [Milacski et al., 2019]

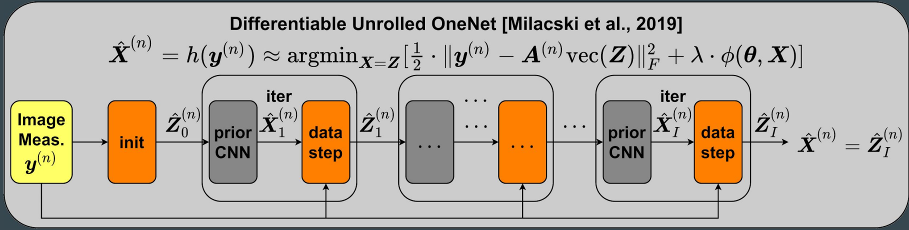
- Apply ADMM [Boyd et al., 2011] (dual ascent with approx. evaluated robustified dual function) to

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

- The Proximal operator of ϕ appears in (c).
- This is separated from $\boldsymbol{A}^{(n)}$, which appears only in (d)…
- … The procedure is actually end-to-end differentiable… Train the CNN with ADMM!
- Self-Supervised Multi-Task Learning vs ground truth:

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

- Much faster ($I \leq 13$ iterations) while also better!
- Requires $\boldsymbol{A}^{(n)}$ at training time…



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

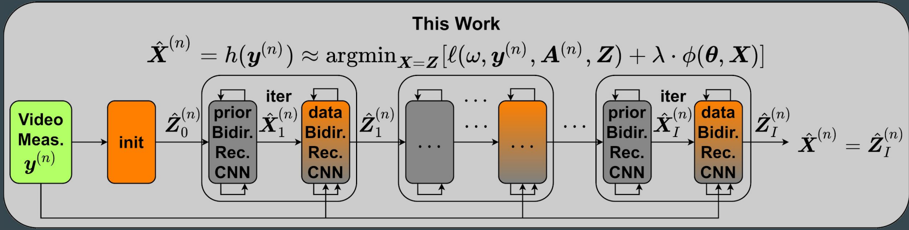
Input: $\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}, \rho^{(n)}, p_{\text{CNN}}(\boldsymbol{\theta}, \cdot)$
{Initialization}
(a)-(b) of Algorithm 1
{Iterations}
for $i = 1, \ldots, I$ **do**
{Prior Step}
{Deep Time Distributed Convolutional Prior Network}
$\boldsymbol{X}^{(n,i,\cdot)} = p_{\text{CNN}}\left(\boldsymbol{\theta},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

This Work: Bidirectional Recurrence, Learned Data Step

1. Replace the prior CNN with a Bidirectional Recurrent CNN (BCRNN)
    - Better suited for videos due to forward propagation across frames
2. Add a 2nd BCRNN on top of the Data Step.
    - Basically we augment the other proximal operator and dual ascent with learned parts.
    - Ideally one would directly input $\boldsymbol{y}^{(n)}$ and $\boldsymbol{A}^{(n)}$ to this network, but their row size varies.
    - Solution: concatenated inputs of fixed size.

Still train the whole thing end-to-end with Self-Supervised Multi-Task Learning vs ground truth:

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



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

---

Input: $\boldsymbol{y}^{(n)}, \boldsymbol{A}^{(n)}, \rho^{(n)}, p_{\text{BCRNN}}(\boldsymbol{\theta}, \cdot), q_{\text{BCRNN}}(\boldsymbol{\omega}, \cdot)$

{Initialization}

(a)-(b) of Algorithm 1

$\tilde{\boldsymbol{Z}}^{(n,0,\cdot)} = \text{reshape}\left(\boldsymbol{A}^{(n)T} \boldsymbol{y}^{(n)}, (1, 1, T, H, W, F)\right)$ (ã)

{Iterations}

for $i = 1, \ldots, I$ **do**

{Prior Step}

{Deep Bidirectional Convolutional Recurrent Prior Network}

$\boldsymbol{X}^{(n,i,\cdot)} = p_{\text{BCRNN}}\left(\boldsymbol{\theta}, \left[\boldsymbol{Z}^{(n,i-1,\cdot)}, \boldsymbol{U}^{(n,i-1,\cdot)}\right]\right)$ (c)

{Data Step}

(d)-(e) of Algorithm 1

{Deep Bidirectional Convolutional Recurrent Data Network}

$\boldsymbol{C}^{(n,i,\cdot)} = \left[\boldsymbol{X}^{(n,i,\cdot)}, \boldsymbol{Z}^{(n,0,\cdot)}, \tilde{\boldsymbol{Z}}^{(n,0,\cdot)}, \boldsymbol{Z}^{(n,i-1,\cdot)}, \boldsymbol{Z}^{(n,i,\cdot)}, \boldsymbol{U}^{(n,i-1,\cdot)}, \boldsymbol{U}^{(n,i,\cdot)}\right]$

$\left[\boldsymbol{Z}^{(n,i,\cdot)}, \boldsymbol{U}^{(n,i,\cdot)}\right] = q_{\text{BCRNN}}\left(\boldsymbol{\omega}, \boldsymbol{C}^{(n,i,\cdot)}\right)$ (f)

end for

Output: $\boldsymbol{Z}^{(n,I,\cdot)}$

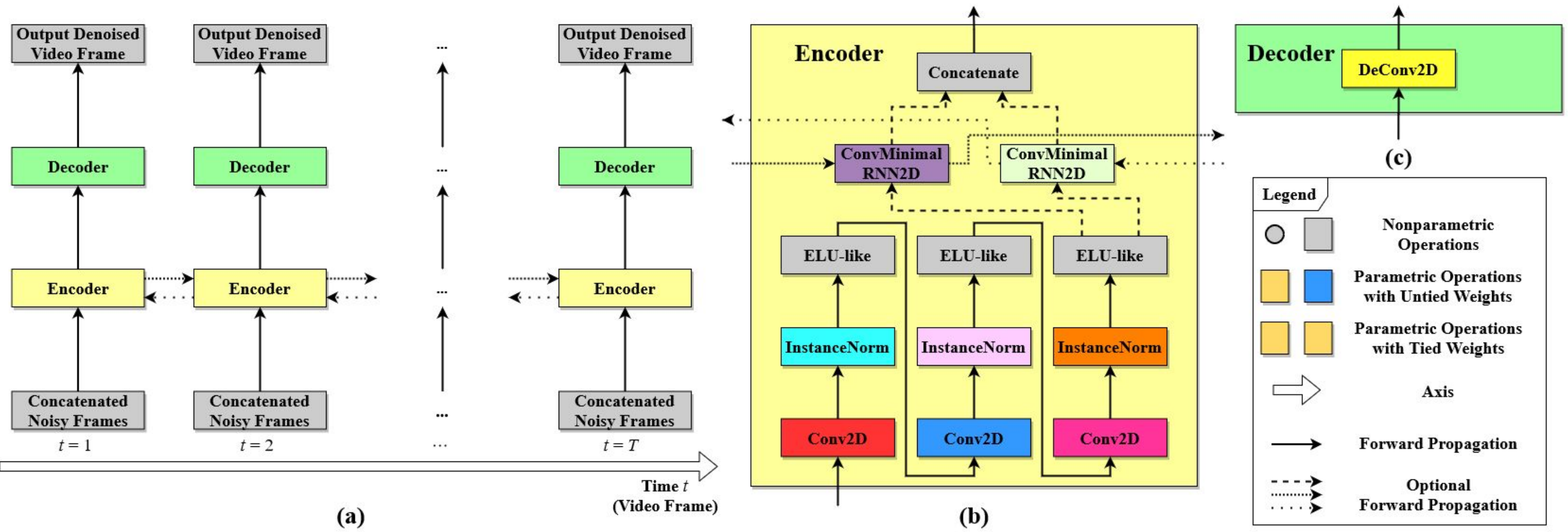
Experimental Setup

- Data sets (small ones, as $\boldsymbol{A}^{(n)}$ is large):
  - Rotated MNIST: 9 frames of size 14x14x1
  - Scanned CIFAR-10: 4 frames of size 16x16x3
  - UCF-101: 4 frames of size 32x32x3
- Baselines:
  - 3D Wavelet Sparsity
  - End-to-end OneNet [Milacski et. al., 2019]
  - Ablation study for our 2 contributions
- Settings:
  - $I = 13$ unrolled iterations
  - Train with ADAM SGD
- Hardware (GPUs):
  - Rotated MNIST: Tesla K80 12 GB VRAM
  - Scanned CIFAR-10: Tesla V100 16 GB VRAM
  - UCF-101: 2xTesla P40 24 GB VRAM
- Evaluation: PSNR score
  - Reverse log-scale (very sensitive to large errors)

- Restoration Problems:
  - Pixelwise Inpaint.-Den. (PID; 50%, std=0.1)
  - Blockwise Inpainting (BI; side 40%)
  - Scattered Inpainting (SI; 10 blocks side 20%)
  - Super Resolution (SR; 2x & 4x)
  - 4D Compressive Sensing (CS; 10%)
  - Video Compressive Sensing (VCS; frame differences compressed to 10%)
  - 2D Spatial & 1D Temporal Disk Deblurring (DD & TDD; filter size 4x4 and 4, radius 2)
  - 2D Spatial Motion Deblurring (MD; filter size 7x7
  - Frame Interpolation (FI; from 2x to 4.5x)
  - Frame Prediction (FP; from (9/7)x to 4x)
  - Colorization (C; from grayscale).

Experimental Setup

- Standard BCRNN architecture for both prior and data networks:



Quantitative Results (Mean±std PSNR, including Ablation)

Rotated MNIST $(10{,}000 \times 9 \times 14 \times 14 \times 1)$

	PID	BI	SI	SR 2×	SR (14/3)×	CS	VCS	DD	TDD	MD	FI 2.25×	FI 4.5×	FP (9/7)×	FP 1.8×	FP 3×
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

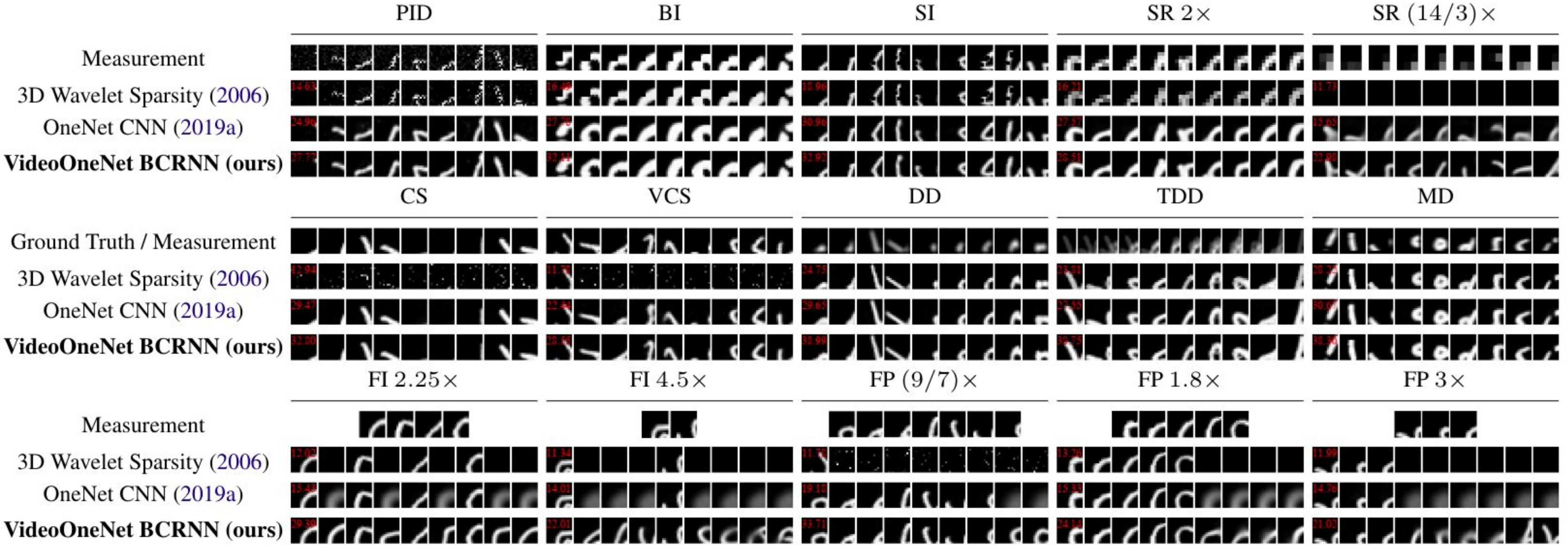
Scanned CIFAR-10 $(10{,}000 \times 4 \times 16 \times 16 \times 3)$

	PID	BI	SI	SR 2×	SR 4×	CS	VCS	DD	TDD	MD	FI 2×	FP (4/3)×	FP 2×	FP 4×	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

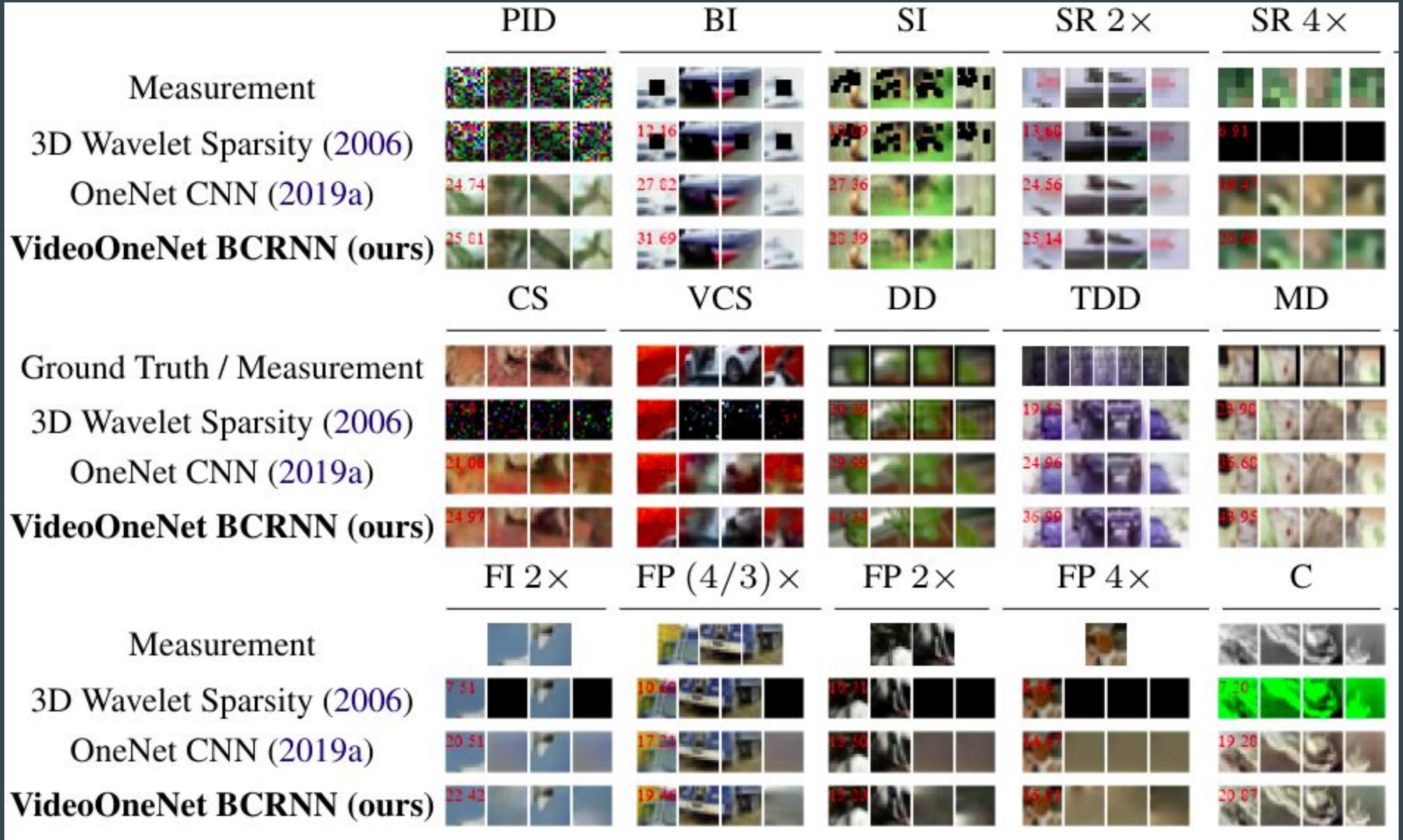
UCF-101 $(3{,}783 \times 4 \times 32 \times 32 \times 3)$

	PID	BI	SI	SR 2×	SR 4×	CS	VCS	DD	TDD	MD	FI 2×	FP (4/3)×	FP 2×	FP 4×	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

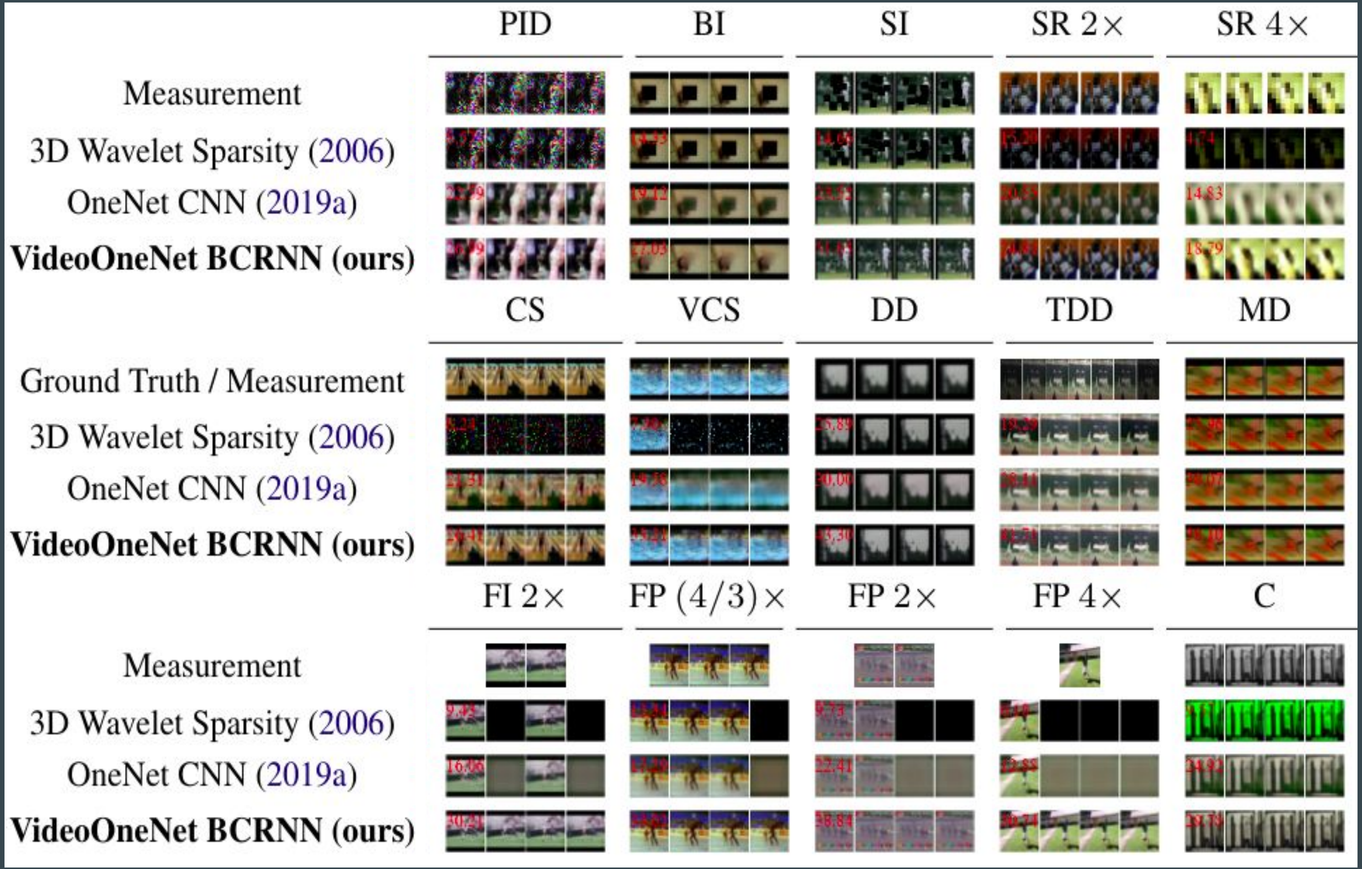
Qualitative Results: Rotated MNIST



Qualitative Results: Scanned CIFAR-10



Qualitative Results: UCF-101



Conclusions

- Improved OneNet for video data
  - Contribution 1: BCRNN prior network
  - Contribution 2: BCRNN data network
  - Consistent yields for both on 2 semi-toy problems and a real data set across 15 tasks
  - Best results when combined
- Future work:
  - Scaling to larger sample and frame sizes (reducing the computational burden, e.g., via fewer ADMM iterations or avoiding least squares)
  - Adversarial training (some restorations are blurry due to MSE loss)
  - Meta-Learning: learning to learn new few-shot problems (avoiding overfitting to test problems)
  - Pretraining for classification (so many tasks may actually yield good features)
- Source code (results are fully reproducible): https://github.com/srph25/videoonenet

Thank you!

Thank you for your attention!

See you in the live chat session! Questions are welcome!

I am looking for a postdoc position.

Email: srph25@gmail.com