Deep RNNs for Video Denoising

Xinyuan Chen\textsuperscript{a}, Li Song\textsuperscript{a}, and Xiaokang Yang\textsuperscript{a}

\textsuperscript{a}Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Cooperative Medianet Innovation Center, Shanghai, China

ABSTRACT

Video denoising can be described as the problem of mapping from a specific length of noisy frames to clean one. We propose a deep architecture based on Recurrent Neural Network (RNN) for video denoising. The model learns a patch-based end-to-end mapping between the clean and noisy video sequences. It takes the corrupted video sequences as the input and outputs the clean one. Our deep network, which we refer to as deep Recurrent Neural Networks (deep RNNs or DRNNs), stacks RNN layers where each layer receives the hidden state of the previous layer as input. Experiment shows (i) the recurrent architecture through temporal domain extracts motion information and does favor to video denoising, and (ii) deep architecture have large enough capacity for expressing mapping relation between corrupted videos as input and clean videos as output, furthermore, (iii) the model has generality to learned different mappings from videos corrupted by different types of noise (e.g., Poisson-Gaussian noise). By training on large video databases, we are able to compete with some existing video denoising methods.

Keywords: Deep RNNs, Video Denoising, patch-based method, spatial-temporal processing

1. INTRODUCTION

Video denoising is a classical problem in computer vision and video processing, as well as an assessment for low-level video modeling algorithm. It has been widely studies in the literature, yet remains an active topic. A video denoising procedure takes a corrupted video $Y = X + N$ as input, where $X$ is the clean video of $Y$ and $N$ is the additive noise. It outputs the video where the noise has been reduced. By accommodating different types of noise distribution, the same model could extend to suit for different kinds of low-level video processing problems, like deblur, super-resolution, inpainting, etc.

Nowadays, deep learning has made great progress in computer vision and pattern recognition application (e.g., image classification using deep convolutional networks\textsuperscript{1}), thanks to its enormous expression ability for representation and fast running on Graphics Processing Units (GPUs). How to apply deep learning on exploring time sequence data and mining temporal information is a popular topic. RNNs are a superset of feedforward neural networks, with the ability to pass information across time steps. They are first widely used in language processing domain like speech recognition,\textsuperscript{2} image description.\textsuperscript{3} In computer vision domain, Nitish Srivastava et al.\textsuperscript{4} verifies that RNNs have ability to learn both motion information and profile feature from videos and successfully exploits such representation to perform pattern recognition, which motivates us to propose a novel patch-based deep RNNs model for video sequence denoising. To the best of our knowledge, our method is the first one proposing an end-to-end deep RNNs for video denoising processing. The procedure is to take a few continuous noisy frames as input and outputs images where the noise has been reduced. Results show that removing on additive white Gaussian noise are competitive with the current state of the art. The approach is equally valid for other type of noise.

In summary, following reasons inspire us to proposed deep RNNs for video denoising:

Further author information: (Send correspondence to Xinyuan Chen, Li Song)

Xinyuan Chen: E-mail: xychen91@sjtu.edu.cn, Telephone: +86 15800767939
Li Song: E-mail: song.li@sjtu.edu.cn, Telephone: +86 13816840561
Xiaokang Yang: E-mail: xkyang@sjtu.edu.cn, Telephone: +86 13761192352
1. RNNs have ability to extract motion information and profile feature by temporal sequences, which helps perform video denoising.

2. Deep architecture have large capacity for expressing mapping relation between corrupted videos as input and clean videos as output.

3. Generality for different types of noise can be simulated.

The contribution of this paper goes that we present a patch-based denoising algorithm that is learned on a large dataset with a deep recurrent neural network. Results on additive white Gaussian noise (AWGN) approach to the current state of the art. The model is equally valid for other types of noise and other mixed noise.

2. RELATED WORK

In the last decades, great progress has been made in image denoising attempted by various methods, e.g. sparse coding method, conditional random fields, variation techniques, patch based methods, etc. Among these, BM3D is known as a state of the art approach. It exploits similar images patches based on $\ell_2$ norm distance, and then noise is removed after wavelet shrinkage or Wiener filter in 3D transform domain.

Video denoising is different as video sequences have motion information and high temporal redundancy to be exploited during noise removing procedure. A common method of patch-based image denoising can be applied on video denoising by searching for similar patched on different frames over time. Liu and Freeman also use motion vectors and group patches across adjacent frames but in a different manner. Instead of comparing patches to the reference patch, these are compared in each frame with the compensated patch of the reference one. NL-means is applied to this group of collected patches. The proposed algorithm adaptively computes the noise model of the sequence, which is an important issue for real applications. Gijis and Zhou took the whole adjacent frames into account by using motion estimation for global motion compensation. Then wavelet transform is applied to the piled frames including current frame and past/future frames. Mairal et al. learnt multi-scale sparse representations for video restoration. VBM3D evolves from BM3D method by matching similar patches both within images and over different timestep frames by predictive search block-matching. VBM4D, the state-of-the-art for white noise removal in video, evolved from VBM3D by exploiting similarity between 3D spatio-temporal volumes instead of 2D patches and grouped similar volumes together by stacking them along an additional fourth dimension, thus producing a 4D structure. Collaborative filtering is realized by transforming each group through a decorrelating 4D separable transform and then by shrinkage and inverse transformation. Low rank matrix completion is another practicable method combining patch-based methodology. It grouped similar patches in temporal-spatial field and minimize the nuclear norm (sum of all singular values) of the matrix with linear constraints, leads to state-of-the-art performance in mixed noises denoising. Another way to utilize spatiotemporal information is to combine with motion estimation. The model simultaneously captured the local correlations between the wavelet coefficients of natural video sequences across both space and time, strengthened with a motion compensation process.

As recurrent neural network (RNN) can model long-term contextual information for video sequence, a great amount of successes have been made for video processing based on RNN range from high-level recognition, middle level to low level. Recently, Yan Huang has made breakthrough on multi-frames super-resolution by using a bidirectional recurrent convolutional network for efficient multi-frame SR, it’s a remarkable work for using RNNs model in low-level video processing. To our knowledge, there’s no video denoising by RNNs at current, so our work is the first succeeding in proposing RNNs based model for video denoising.

3. MODEL DESCRIPTION

3.1 Deep Recurrent Neural Networks

Recurrent neural networks are a powerful family of connectionist models that capture time dynamics via cycles in the graph. Information can cycle inside the networks during timesteps. Details about recurrent neural networks can be found in a review of RNN. Briefly speaking, A single RNN unit is shown in Figure 1(a). At time $t$,
Figure 1. Structure of Recurrent Neural Networks. (a) A simple recurrent network, the delay nodes represent recurrent in time. (b) Deep Recurrent Neural Networks of 2 layers.

hidden layer units $h^{(t)}$ receive activation from input $x^{(t)}$ at current timestep and hidden states $h^{(t-1)}$ at last timestep. The output $y^{(t)}$ is computed from hidden units $h^{(t)}$ at time $t$:

$$h^{(t)} = \sigma \left( W_{hx} x^{(t)} + W_{hh} h^{(t-1)} + b_h \right)$$  \hspace{1cm} (1)

$$y^{(t)} = \sigma \left( W_{yh} h^{(t)} + b_y \right)$$  \hspace{1cm} (2)

The weight matrices $W_{hx}, W_{hh}, W_{yh}$ and vector-valued biases $b_h, b_y$ parameterize the RNN, the activation function $\sigma (\cdot)$ (e.g. tanh or sigmoid function) operates component-wise. In our model, All of activation function is hyperbolic tangent function except the output layer uses linear function. The greatest difference between RNN and normal neural networks is that the recurrent hidden units’ states are influenced by not only at current inputs but also previous hidden state. As a result, hidden units can been seen as container carry information of previous time sequences.

Deep recurrent neural networks (Deep RNNs) is an extended deeper RNN by stacking one input layer, multiple hidden layers and one output layer. In fact, deep RNNs model stacked different layers in depth with the same way to multi layer perceptions (MLP). If we get rid of the cycle procedure through timesteps, Deep RNNs becomes to multi layer perceptions. A two-hidden-layers deep RNNs is shown in Figure 1(b). Generally speaking, Deep RNNs have more hidden layers compared to simple RNN. Hidden layer $h^{(t)}_l$ ($l > 2$) receives layer below $h^{(t-1)}_{l-1}$ and previous hidden states $h^{(t-1)}_l$ at current layer as input:

$$h^{(t)}_l = \sigma \left( W_{h_l} h^{(t)}_{l-1} + W_{h_{l-1}} h^{(t)}_{l-2} + b_h \right)$$  \hspace{1cm} (3)

3.2 Deep RNNs for Video Denoising

In this section, we propose an end-to-end model that uses Deep RNNs for video denoising. The model consists of two recurrent layers as shown in Figure 2. The input to the model is a sequence of vectors (corrupted video cubes), and the target output is groundtruth vector (clean image patches), which means to denoise a given frame’s noise we take several future/past adjacent frames into account.

Let $\chi = \{x^t\}_{t=1}^T$ be the image sequence with $T$ frames. Each image $x^t$ can be treat as a sum of its underlying clean image $y^t$ and the noise $n^t$:

$$x^t = y^t + n^t$$  \hspace{1cm} (4)

The goal of video denoising is to construct a mapping into $\mathcal{Y} = \{y^t\}_{t=1}^T$ by removing $n^t$ from $x^t$.

We adopt patch-based method in our model. In order to incorporate temporal redundancy of video into deep RNNs, we take in video cubes as input. Here, cube refers multi-dimensional array of data with specific patch size and timestep picked from video. More precisely, it stacks from several timing continuous patches (see blue
cube in Figure 2 (a)). The Deep RNNs aim to map such cubes from noisy video into clean image patches of the midmost timestep: \( \hat{y}(T+1/2) = F(\chi; \Theta) \), where \( \Theta \) denotes to network parameters. They are updated by the BPTT algorithm\(^{26} \) minimizing the quadratic error between the mapped noisy patch and clean patch:

\[
L = \| F(\chi; \Theta) - Y \| \tag{5}
\]

The intuitive processing of our Deep RNNs denoising model can be described as follow. The first recurrent hidden layer reads input data in sequence and delivers the representation of the input video to second layer. Then the second recurrent hidden layer tries to extract higher level representation and learning more powerful expression from the representation. At last, the output layer is being asked to construct the clean image from this representation. In order to do so, the representation must retain information about the appearance of the objects and the background as well as the motion contained in the video.

### 3.3 Applying Deep RNNs for Video Denoising

To denoise video, we decompose a given noisy video into overlapping patch frame-by-frame. Then we feed its corresponding cubes (a list of patches in the same position each frame) into trained model. In other words, inputs can be obtained by sliding through video volume with a 3D window of specific spatial size and temporal step. The denoised video is obtained by placing the denoised patches at the location their noisy counterparts averaging the overlapping regions. When we pick overlapping patches, the stride set to be 3 as denosing performance is almost equally good with smaller stride.

### 4. EXPERIMENTAL SETUP

We design experiment to accomplish the following objectives:

- Compare DRNNs with different model variants to illustrate DRNNs exploit latent temporal-spatial information for video denoising.
- Compare with state-of-the-art video noise benchmark on removing addictive white Gaussian noise.
- Measure the generality of DRNNs for other type of noise (e.g., Poisson-Gaussian noise).
4.1 Dataset

Training set: The UCF-101 dataset\cite{dataset} is a standard benchmark dataset for human action recognition. It has 13320 videos of variable length belonging to 101 human action categories, and each frame has size 160×320 pixels. We use it to train deep RNNs for video denoising because it’s large enough and consist of natural videos containing motion and the spatial information well.

We performed all our experiments on gray-scale video. We randomly pick 17×17 patches from 7 continuous frames to generate 17×17×7 cubes as raw sequences samples. Then we manually add noise on these samples to generate training samples. The groundtruth of each training samples is corresponding to clean patches from the middle of timestep.

Testing set: We use standard test sequence the same with test sequence in paper of VBM3D\cite{vbm3d} and VB-M4D,\cite{vbm4d} which can be downloaded by the author’s website. Figure 3 displays the central frame of the used sequences.

4.2 Implementation

Python library Theano\cite{theano} is used to construct deep RNNs (DRNNs) model. All models were trained on a single NVIDIA Titan X GPU. A two layer deep RNNs took about 10 days to converge. In order to make training procedure more efficient, we apply some common tricks:

- Data normalization: The pixel values are transformed to have approximately mean zero and variance one. More precisely, we subtract 0.5 from each pixel and divide by 0.2, assuming pixel values between 0 and 1.

- Weight initialization: The neural net’s weights are initialized from a uniform distribution $U \left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ where $n$ is the size of the previous layer.\cite{weight_init}

- Learning Strategy: The learning rate is initialized as 0.03 and stopped as 0.0001, while momentum is initialized as 0.9 and stopped as 0.999. Both of them are decreased at the same step in every epoch. To avoid waiting for max epochs when validation error stopped improving, we use early stopping strategy. We stop training when validation error stopped improve in the latest 200 epoches, and save model at the epoch where validation error reached the lowest.

5. RESULTS

5.1 Comparison of Different Model Variants

In this set of experiment, we compare different model variants to verify if recurrent unit and deep architecture play important roles in denoising. All the models are trained on video cubes corrupted with Gaussian noise with $\sigma = 25$: $n^t \sim N (0, \sigma^2 I)$. The training set contains about 10 million cubes of 500 videos randomly picked from UCF-101, twenty percent of which is chosen as validation set. Table 1 shows the denoising testing results using peak signal to noise ratio (PSNR) as metric.
To compare performance of single layer RNN and deep RNNs, we trained an RNN model of $17 \times 17 - 1024 - 17 \times 17$ layers (shown in Figure 4(a)) and deep RNNs model of $17 \times 17 - 1024 - 1024 - 17 \times 17$ layers with 5 timesteps, notating as RNN-5 and DRNNs-5. As we can see, DRNNs-5 outperforms RNN-5 on all the test sequence, which means deep architecture have better mapping capacity from noisy video to clean one.

To find if recurrent unit helps for temporal information extracting and motion analysis, we trained a model of 2 layers MLP of 7 timesteps pooling in the last layer (notating as MLP-7 shown in Figure 4(b)) and a 2 layers deep RNNs of 7 timesteps (DRNNs-7). Also we use MLPs$^{30}$ methods as comparison. The denoising MLPs model is obtained from author’s website and applied on corrupted video frame by frame to remove noise. MLPs has 4 layers with 2047 hidden units in each layer. The stride is 3 on both DRNNs and MLPs when clipping overlapping patches. Result shows DRNNs-7’s denoising PSNR is 1.91dB higher than MLPs-7 and 2.53 higher than MLPs on average.

Noticing that two layers MLPs-X shared the same order of magnitudes numbers of parameters with single layer RNN-X (X notating as number of timesteps), because recurrent layer occupies the same space complexity of one fully connected layer. In table 1 two layer MLPs-7’s performance is even poorer than RNN-5’s with narrow temporal windows, from which we draw the conclusion that recurrent layer extracts temporal information and does motion analyse for video denoising which is superior to simply stacking a sequence of frames and feeding into network without timing connection. DRNNs outperforms MLPs on all of test videos, which illustrates temporal information improves denoising performance. The performance is inferior to VBM3D at a small distance.

---

**Figure 4.** Sketch of model variants. (a) Single layer RNN for video denosing. (b) MLP-7 for video denoising without recurrent layer.

**Table 1.** Denoising results of PSNR (dB) by variants of DRNNs on testing videos corrupted addictive Gaussian noise with standard deviation $\sigma = 25$

<table>
<thead>
<tr>
<th>Video name:</th>
<th>Salesm.</th>
<th>Miss Am.</th>
<th>Coastg.</th>
<th>Foreman</th>
<th>Bus</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame size</td>
<td>288×352</td>
<td>288×352</td>
<td>144×176</td>
<td>288×352</td>
<td>288×352</td>
<td></td>
</tr>
<tr>
<td>Frame number</td>
<td>50</td>
<td>150</td>
<td>300</td>
<td>300</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>MLPs$^{30}$</td>
<td>28.68</td>
<td>32.87</td>
<td>27.71</td>
<td>29.63</td>
<td>27.34</td>
<td>29.25</td>
</tr>
<tr>
<td>MLPs-7</td>
<td>29.98</td>
<td>34.73</td>
<td>28.43</td>
<td>29.97</td>
<td>26.22</td>
<td>29.87</td>
</tr>
<tr>
<td>RNN-5</td>
<td>31.07</td>
<td>36.02</td>
<td>29.23</td>
<td>30.81</td>
<td>27.01</td>
<td>30.83</td>
</tr>
<tr>
<td>DRNNs-5</td>
<td>31.43</td>
<td>36.64</td>
<td>29.72</td>
<td>31.31</td>
<td>27.54</td>
<td>31.33</td>
</tr>
<tr>
<td>DRNNs-7</td>
<td>32.27</td>
<td>37.01</td>
<td>30.06</td>
<td>31.66</td>
<td>27.88</td>
<td>31.78</td>
</tr>
</tbody>
</table>
Figure 5. From left to right and from top to bottom, visual comparison of the denoising performance of R-NL, VBM3D, VBM4D and DRNNs on the sequence of salesm., Miss Am., and foreman corrupted by white Gaussian noise with standard deviation $\sigma=35$.

By comparing DRNNs-5’s and DRNNs-7’s results, we find that by enlarging the timestep windows, noise can be removed more thoroughly as the model has more reference frames. The performance might be improved further if temporal windows become larger. But considering exponentially increasing time cost, we haven’t done experiment with wider temporal windows.

5.2 Comparison with State-of-the-art Performance

We compared 7 timesteps $17 \times 17$--$1024$--$1024$--$17 \times 17$ deep RNNs with state-of-the-art algorithm. The results of algorithm R-NL$^{14}$ are also displayed for comparison. The Matlab implementation of VBM3D, VBM4D and R-NL was obtained from the authors web site. The default parameters were used in the tests if the author haven’t provided results in their paper. All the test video sequence are provided by VBM3D’s author on his web site. We compared the performance of R-NL, VBM3D, VBM4D and the proposed algorithm (DRNNs) both visually and numerically in several image sequences. Figure 3 displays the central frame of the used sequences. Table 2 shows the videos’ sizes and frame number. We trained two networks processing with noisy video corrupted by AWGN with $\sigma = 25$ and $\sigma = 35$, both of whose training strategy and tricks have been mentioned in Section 4.

Table 2 reports denoising performance in terms of PSNR using different methods. DRNNs reaches an approximate result with VBM4D methods in less than 1 dB difference on average and outperforms R-NL methods. Figure 5 displays the visual results of the compared algorithms. The figures also display the difference of the denoised image with the noisy one, and the difference of the denoised image to the original one. The difference with the noisy image displays the noise removed by each algorithm. The results of R-NL smooth the image overmuch while remove too much details and textures. DRNNs seemly performs similar with VBM3D and VBM4D by a first glimpse while its weakness is some slight noise spot still on the images, but details are protected better especially comparing with R-NL.

5.3 DRNNs for Mix Noises

Videos are not always corrupted by addictive white Gaussian noise. In some situations, the video processing might be corrupted by mix noise, like Gaussian and Poisson noise. Most existing denoising removers are quite sensible to the noise model with certain type of noise. DRNNs allows us to effectively learn a denoising model for a given noise of any types, if noise can be simulated. In this set of experiment, we trained a DRNNs model for removing mix noise in Poisson-Gaussian noise:

$$n_t = n^g_t + n^p_t$$  \hspace{1cm} (6)
Table 2. Denoising performance of R-NL, VBM3D, VBM4D and DRNNs. The PSNR (in dB) are reported. The test videos are corrupted by AWGN with corresponding standard deviation $\sigma$ showed on left row.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>Video name</th>
<th>Salesm.</th>
<th>Miss Am.</th>
<th>Coastg.</th>
<th>Foreman</th>
<th>Bus</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Frame size</td>
<td>288×352</td>
<td>288×352</td>
<td>144×176</td>
<td>288×352</td>
<td>288×352</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Frame number</td>
<td>50</td>
<td>150</td>
<td>300</td>
<td>300</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>R-NL$^{14}$</td>
<td>31.81</td>
<td>36.91</td>
<td>30.32</td>
<td>31.53</td>
<td>27.48</td>
<td>31.61</td>
</tr>
<tr>
<td></td>
<td>VBM3D$^{12}$</td>
<td>32.79</td>
<td>37.10</td>
<td>30.62</td>
<td>32.19</td>
<td>28.48</td>
<td>32.24</td>
</tr>
<tr>
<td></td>
<td>VBM4D$^{17}$</td>
<td>32.66</td>
<td>37.24</td>
<td>30.81</td>
<td>32.61</td>
<td>29.10</td>
<td>32.48</td>
</tr>
<tr>
<td></td>
<td>DRNNs</td>
<td>32.27</td>
<td>37.01</td>
<td>30.06</td>
<td>31.66</td>
<td>27.88</td>
<td>31.78</td>
</tr>
<tr>
<td>35</td>
<td>R-NL$^{14}$</td>
<td>29.88</td>
<td>35.60</td>
<td>28.51</td>
<td>29.80</td>
<td>25.55</td>
<td>29.87</td>
</tr>
<tr>
<td></td>
<td>VBM3D$^{12}$</td>
<td>30.72</td>
<td>35.87</td>
<td>28.92</td>
<td>30.50</td>
<td>26.91</td>
<td>30.60</td>
</tr>
<tr>
<td></td>
<td>VBM4D$^{17}$</td>
<td>30.99</td>
<td>35.98</td>
<td>29.17</td>
<td>31.11</td>
<td>27.39</td>
<td>31.04</td>
</tr>
<tr>
<td></td>
<td>DRNNs</td>
<td>30.51</td>
<td>35.88</td>
<td>28.43</td>
<td>30.09</td>
<td>26.02</td>
<td>30.19</td>
</tr>
</tbody>
</table>

where $n^g_t$ denotes Gaussian noise with zero mean and pixel independent variance $\sigma = 25$, $n^p_t$ denotes Poisson noise (shot noise) with zero mean and variance $\lambda = ky^t$. We use the formulation $I = kP(y_t^p/k)^{31}$ and in our experiment, we set $k = 15$. Figure 6(a) shows the noisy frame from video ‘salesman’ in the middle of the sequence (the 25th frame).

As there’s no specific methods to remove video’s Poisson-Gaussian noise, we compare results to three existing video denoising methods, two of those are VBM3D method and R-NL mentioned above. Another is low rank matrix completion$^{18}$ method for video denoising. Figure 6 shows the visual results. As VBM3D is designed for AWGN with known variance, we first perform VBM3D given $\sigma = 25$, and it turns out to be still noisy (PSNR=21.50dB). Then we feed the processed video as input to R-NL method to remove part of Poisson noise(by setting the corresponding Poisson noise $\lambda = 15y^t$. The result in Figure6(d) shows to be smooth too much (PSNR=26.97dB). We also set different value of variance in VBM3D to various from 25 to 50 and find the best denoising results when the variance to be $\sigma = 45$ (PSNR=28.44dB). R-NL method and low rank method are shown in Figure6(f) and (g), it seems R-NL method (PSNR=27.79dB) remove too much texture and features of the image while low rank method (PSNR=24.37dB) still preserve some noisy black spot in the image. Overall, deep RNNs methods (PSNR=30.09dB) outperform other methods in both PSNR and visual result.

6. CONCLUSION

Deep recurrent neural networks can exploit temporal-spatial information of video to removing video noise, as well as deep architecture have large capacity for expressing mapping relation between corrupted videos as input and clean videos as output. Results of our method approach to state-of-the-art video denoising performance. Our proposed method does not assume any specific statistical properties of noise and is robust to map relation between corrupted video as input and clean video as output.

ACKNOWLEDGMENTS

This work was supported by NSFC (61527804,61521062,61420106008), the 111 Project (B07022 and Sheitc No. 150633) and the Shanghai Key Laboratory of Digital Media Processing and Transmissions.

REFERENCES


Figure 6. Visual comparison of the denoising performance by different methods on the sequence of salesman corrupted by Poisson-Gaussian noise with $\sigma = 25$ and $\lambda = 15y^k$. The PSNR from (b) to (h) is 16.33dB, 21.50dB, 26.97dB, 28.44dB, 27.79dB, 24.37dB, 30.09dB.


