DVDNET: A FAST NETWORK FOR DEEP VIDEO DENOISING

Matias Tassano, Julie Delon, Thomas Veit

DVDNET: A FAST NETWORK FOR DEEP VIDEO DENOISING

Matias Tassano[⋆†] *Julie Delon*[⋆] *Thomas Veit*[†]

[⋆] MAP5, Université Paris Descartes
[†] GoPro France

ABSTRACT

In this paper, we propose a state-of-the-art video denoising algorithm based on a convolutional neural network architecture. Previous neural network based approaches to video denoising have been unsuccessful as their performance cannot compete with the performance of patch-based methods. However, our approach outperforms other patch-based competitors with significantly lower computing times. In contrast to other existing neural network denoisers, our algorithm exhibits several desirable properties such as a small memory footprint, and the ability to handle a wide range of noise levels with a single network model. The combination between its denoising performance and lower computational load makes this algorithm attractive for practical denoising applications. We compare our method with different state-of-art algorithms, both visually and with respect to objective quality metrics. The experiments show that our algorithm compares favorably to other state-of-art methods. Video examples, code and models are publicly available at `https://github.com/m-tassano/dvdnet`.

Index Terms— video denoising, CNN, residual learning, neural networks, image restoration

1. INTRODUCTION

We introduce a network for Deep Video Denoising: DVDnet. The algorithm compares favorably to other state-of-the-art methods, while it features fast running times. The outputs of our algorithm present remarkable temporal coherence, very low flickering, strong noise reduction, and accurate detail preservation.

1.1. Image Denoising

Compared to image denoising, video denoising appears as a largely underexplored domain. Recently, new image denoising methods based on deep learning techniques have drawn considerable attention due to their outstanding performance. Schmidt and Roth proposed in [1] the cascade of shrinkage fields method that unifies the random field-based model and half-quadratic optimization into a single learning framework. Based on this method, Chen and Pock proposed in [2] a trainable nonlinear reaction diffusion model. This model can be expressed as a feed-forward deep network by concatenating a fixed number of gradient descent inference steps. Methods such as these two attain denoising performances comparable to those of well-known algorithms such as BM3D [3] or non-local Bayes (NLB [4]). However, their performance is restricted to specific forms of prior. Additionally, many hand-tuned parameters are involved in the training process. In [5], a multi-layer perceptron was successfully applied for image denoising. Nevertheless, a significant drawback of all these algorithms is that a specific model must be trained for each noise level.

Another popular approach involves the use of convolutional neural networks (CNN), e.g. RBDN [6], DnCNN [7], and FFDNet [8]. Their performance compares favorably to other state-of-the-art image denoising algorithms, both quantitatively and visually. These methods are composed of a succession of convolutional layers with nonlinear activation functions in between them. This type of architecture has been applied to the problem of joint denoising and demosaicing of RGB and raw images by Gharbi et al. in [9]. Contrary to other deep learning denoising methods, one of the remarkable features that these CNN-based methods present is the ability to denoise several levels of noise with only one trained model. Proposed by Zhang et al. in [7], DnCNN is an end-to-end trainable deep CNN for image denoising. This method is able to denoise different noise levels (e.g. with standard deviation $\sigma \in [0, 55]$) with only one trained model. One of its main features is that it implements residual learning [10], i.e. it estimates the noise existent in the input image rather than the denoised image. In a following paper [8], Zhang et al. proposed FFDNet, which builds upon the work done for DnCNN.

1.2. Video Denoising

As for video denoising, the method proposed by Chen et al. in [11] is one of the few to approach this problem with neural networks—recurrent neural networks in their case. However, their algorithm only works on grayscale images and it does not achieve satisfactory results, probably due to the difficulties associated with training recurring neural networks [12]. Vogels et al. proposed in [13] an architecture based on kernel-

predicting neural networks able to denoise Monte Carlo rendered sequences. The state-of-the-art in video denoising is mostly defined by patch-based methods. Kokaram et al. proposed in [14] a 3D Wiener filtering scheme. We note in particular an extension of the popular BM3D to video denoising, V-BM4D [15], and Video non-local Bayes (VNLB [16]). Nowadays, VNLB is the best video denoising algorithm in terms of quality of results, as it outperforms V-BM4D by a large margin. Nonetheless, its long running times render the method impractical—it could take several minutes to denoise a single frame. The performance of our method compares favorably to that of VNLB for moderate to large values of noise, while it features significantly faster inference times.

2. OUR METHOD

Methods based on neural networks are nowadays state-of-the-art in image denoising. However, state-of-the-art in video denoising still consists of patch-based methods. Generally speaking, most previous approaches based on deep learning have failed to employ the temporal information existent in image sequences effectively. Temporal coherence and the lack of flickering are vital aspects in the perceived quality of a video. Most state-of-the-art algorithms in video denoising are extensions of their image denoising counterparts. Such is the case, for example, of V-BM4D and BM3D, or VNLB and NLB. There are mainly two factors in these video denoising approaches which enforce temporal coherence in the results, namely the extension of search regions from spatial neighborhoods to volumetric neighborhoods, and the use of motion estimation. In other words, the former implies that when denoising a given pixel (or patch), the algorithm is going to look for similar pixels (patches) not only in the same frame, but also in adjacent frames of the sequence. Secondly, the use of motion estimation and/or compensation has been shown to help improving video denoising performance [17, 16, 15]. We thus incorporated these two elements into our algorithm, as well as different aspects of other relevant CNN-based denoising architectures [8, 9, 13]. Thanks to all these characteristics, our algorithm improves the state-of-the-art results, while featuring fast inference times.

Figure 1 displays a simplified diagram of the architecture of our method. When denoising a given frame, its $2T$ neighboring frames are also taken as inputs. The denoising process of our algorithm can be split in two stages. Firstly, the $2T + 1$ frames are individually denoised with a spatial denoiser. Although each individual frame output at this stage features relatively good image quality, they present evident flickering when considered as a sequence. In the second stage of the algorithm, the $2T$ denoised temporal neighbors are registered with respect to the central frame. We use optical flow for this purpose. Splitting denoising in two stages allows for an individual pre-processing of each frame. On top of this, motion compensation is performed on pre-denoised images, which facilitates the task. Finally, the $2T + 1$ aligned frames are concatenated and input into the temporal denoising block. Using temporal neighbors when denoising each frame helps to reduce flickering as the residual error in each frame will be correlated. We also add a noise map as input to the spatial and temporal denoisers. The inclusion of the noise map as input allows the processing of spatially varying noise [18]. Contrary to other denoising algorithms, our denoiser takes no other parameters as inputs apart from the image sequence and the estimation of the input noise.

Observe that experiments presented in this paper focus on the case of additive white Gaussian noise (AWGN). Nevertheless, this algorithm can be straightforwardly extended to other types of noise, e.g. spatially varying noise (e.g. Poissonian). Let $\mathbf{I}$ be a noiseless image, while $\tilde{\mathbf{I}}$ is its noisy version corrupted by a realization of zero-mean white Gaussian noise $\mathbf{N}$ of standard deviation σ, then

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{N} \,. \tag{1}$$

2.1. Spatial and Temporal Denoising Blocks

The design characteristics of the spatial and temporal blocks make a good compromise between performance and fast running times. Both blocks are implemented as standard feed-forward networks, as shown in fig. 2. The architecture of the spatial denoiser is inspired by the architectures in [8, 9], while the temporal denoiser also borrows some elements from [13].

The spatial and temporal denoising blocks are composed of $D_{spa} = 12$, and $D_{temp} = 6$ convolutional layers, respectively. The number of feature maps is set to $W = 96$. The outputs of the convolutional layers are followed by pointwise ***ReLU*** [19] activation functions $ReLU(\cdot) = \max(\cdot, 0)$. At training time, batch normalization layers (***BN*** [20]) are placed between the convolutional and ***ReLU*** layers. At evaluation time, the batch normalization layers are removed, and replaced by an affine layer that applies the learned normalization. The spatial size of the convolutional kernels is 3×3, and the stride is set to 1. In both blocks, the inputs are first downscaled to a quarter resolution. The main advantage of performing the denoising in a lower resolution is the large reduction in running times and memory requirements, without sacrificing denoising performance [8, 18]. The upscaling back to full resolution is performed with the technique described in [21]. Both blocks feature residual connections [10], which have been observed to ease the training process [18].

3. TRAINING DETAILS

The spatial and temporal denoising parts are trained separately, with the spatial denoiser trained first as its outputs are used to train the temporal denoiser. Both blocks are trained using crops of images, or patches. The size of the patches should be larger than the receptive field of the networks. In

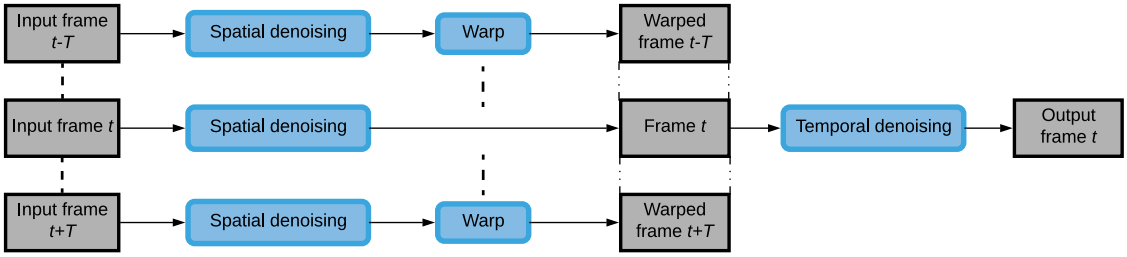

Fig. 1. Simplified architecture of our method.

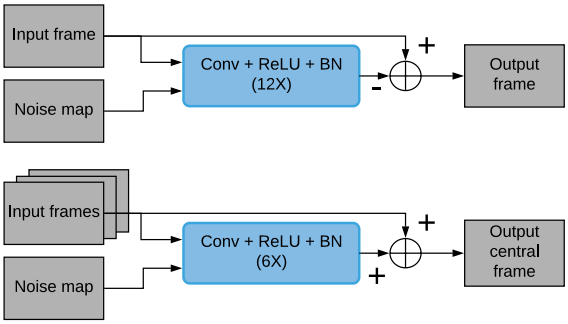

Fig. 2. Simplified architecture of the spatial (top) and temporal (bottom) denoising blocks.

the case of the spatial denoiser, the training dataset is composed of pairs of input-output patches $\left\{\left((\tilde{\mathbf{I}}^j, \mathbf{M}^j),\, \mathbf{I}^j\right)\right\}_{j=0}^{m_s}$ which are generated by adding AWGN with standard deviation $\sigma \in [0,\, 55]$ to the clean patches $\mathbf{I}^j$ and building the corresponding noise map $\mathbf{M}^j$ (which is in this case constant with all its elements equal to σ). A total of $m_s = 1024000$ patches are extracted from the Waterloo Exploration Database [22]. The patch size is 50×50. Patches are randomly cropped from randomly sampled images of the training dataset. Residual learning is used, which implies that if the network outputs an estimation of the input noise $\mathcal{F}_{spa}(\,\tilde{\mathbf{I}}; \theta_{spa}\,) = \hat{\mathbf{N}}$, then the denoised image is computed by subtracting the output noise to the noisy input

$$\hat{\mathbf{I}}(\,\tilde{\mathbf{I}}; \theta_{spa}\,) = \tilde{\mathbf{I}} - \mathcal{F}_{spa}(\,\tilde{\mathbf{I}}; \theta_{spa}\,)\,. \tag{2}$$

The loss function of the spatial denoiser writes

$$\mathcal{L}_{spa}(\theta_{spa}) = \frac{1}{2m_s} \sum_{j=1}^{m_s} \left\| \hat{\mathbf{I}}^j(\,\tilde{\mathbf{I}}^j; \theta_{spa}\,) - \mathbf{I}^j \right\|^2, \tag{3}$$

where θ_{spa} is the collection of all learnable parameters.

As for the temporal denoiser, the training dataset consists of input-output pairs

$$P_t^j = \left\{\left(\left(\left({}^w\hat{\mathbf{I}}_{t-T}^j, \ldots, \hat{\mathbf{I}}_t^j, \ldots, {}^w\hat{\mathbf{I}}_{t+T}^j\right), \mathbf{M}^j\right),\, \mathbf{I}_t^j\right)\right\}_{j=0}^{m_t},$$

where $\left({}^w\hat{\mathbf{I}}_{t-T}^j, \ldots, \hat{\mathbf{I}}_t^j, \ldots, {}^w\hat{\mathbf{I}}_{t+T}^j\right)$ is a collection of $2T+1$ spatial patches cropped at the same location in contiguous frames. These are generated by adding AWGN of $\sigma \in [0,\, 55]$ to clean patches of a given sequence, and denoising them using the spatial denoiser. Then, the $2T$ patches contiguous to the central reference patch $\mathbf{I}_t^j$ are motion-compensated with respect to the latter, i.e. ${}^w\hat{\mathbf{I}}_l^j = \texttt{compensate}(\hat{\mathbf{I}}_l^j, \hat{\mathbf{I}}_t^j)$. To compensate frames, we use the DeepFlow algorithm [23] for the estimation of the optical flow between frames. The noise map $\mathbf{M}^j$ is the same as the one used in the spatial denoising stage. A total of $m_t = 450000$ training samples are extracted from the training set of the DAVIS database [24]. The spatial size of the patches is 44×44, while the temporal size is $2T + 1 = 5$. The loss function for the temporal denoiser is

$$\mathcal{L}_{temp}(\theta_{temp}) = \frac{1}{2m_t} \sum_{j=1}^{m_t} \left\| \hat{\mathbf{I}}_{temp,\,t}^j - \mathbf{I}_t^j \right\|^2, \tag{4}$$

where $\hat{\mathbf{I}}_{temp,\,t}^j = \mathcal{F}_{temp}(P_t^j; \theta_{temp})$.

In both cases, the ADAM algorithm [25] is applied to minimize the loss function, with all its hyper-parameters set to their default values. The number of epochs is set to 80, and the mini-batch size is 128. The scheduling of the learning rate is also common to both cases. It starts at $1\mathrm{e}{-3}$ for the first 50 epochs, then changes to $1\mathrm{e}{-4}$ for the following 10 epochs, and finally switches to $1\mathrm{e}{-6}$ for the remaining of the training. Data is augmented five times by introducing rescaling by different scale factors and random flips. During the first 60 epochs, the orthogonalization of the convolutional kernels is applied as a means of regularization. It has been observed that initializing the training with orthogonalization may be beneficial to performance [8, 18].

4. RESULTS

Two different testsets were used for benchmarking our method: the DAVIS-test testset, and Set8, which is composed of 4 color sequences from the *Derf's Test Media collection*[1] and 4 color sequences captured with a GoPro camera. The DAVIS set contains 30 color sequences of resolution

[1] `https://media.xiph.org/video/derf`

(b) Noisy $\sigma = 50$ (c) V-BM4D (d) VNLB (e) Neat Video (e) DVDnet (ours)

Fig. 3. *Comparison of results*. Left to right: noisy frame (PSNR$_{seq} = 14.15dB$), output by V-BM4D (PSNR$_{seq} = 24.91dB$), output by VNLB (PSNR$_{seq} = 26.34dB$), output by Neat Video (PSNR$_{seq} = 23.11dB$), output by DVDnet (PSNR$_{seq} = 26.62dB$). Note the clarity of the denoised text, and the lack of low-frequency residual noise and chroma noise for DVDnet. Best viewed in digital format.

854×480. The sequences of Set8 have been downscaled to a resolution of 960×540. In all cases, sequences were limited to a maximum of 85 frames. We used the DeepFlow algorithm to compute flow maps for DVDnet and VNLB. We also compare our method to a commercial blind denoising software, Neat Video (NV [26]).

In general, DVDnet outputs sequences which feature remarkable temporal coherence. Flickering rendered by our method is notably small, especially in flat areas, where patch-based algorithms often leave behind low-frequency residual noise. An example can be observed in fig. 3 (which is best viewed in digital format). Temporally decorrelated low-frequency noise in flat areas appears as particularly annoying in the eyes of the viewer. More video examples can be found in the website of the algorithm. The reader is encouraged to watch these examples to compare the visual quality of the results of our method.

Tables 1 and 2 show a comparison of $PSNR$ on the Set8 and DAVIS dataset, respectively. It can be observed that for smaller values of noise, VNLB performs better. In effect, DVDnet tends to over denoise in some of these cases. However, for larger values of noise DVDnet surpasses VNLB.

Table 1. Comparison of $PSNR$ on the Set8 testset.

	DVDnet	VNLB	V-BM4D	NV
$\sigma = 10$	36.08	**37.26**	36.05	35.67
$\sigma = 20$	33.49	**33.72**	32.19	31.69
$\sigma = 30$	**31.79**	31.74	30.00	28.84
$\sigma = 40$	**30.55**	30.39	28.48	26.36
$\sigma = 50$	**29.56**	29.24	27.33	25.46

4.1. Running times

Our method achieves fast inference times, thanks to its design characteristics and simple architecture. DVDnet takes less than $8s$ to denoise a 960×540 color frame, which is about 20 times faster than V-BM4D, and about 50 times faster than VNLB. Even running on CPU, DVDnet is about an order of magnitude faster than these methods. Of the $8s$ it takes to denoise a frame, $6s$ are spent on compensating motion of the temporal neighboring frames. Table 3 compares the running times of different state-of-the-art algorithms.

Table 2. Comparison of $PSNR$ on the DAVIS testset.

	DVDnet	VNLB	V-BM4D
$\sigma = 10$	38.13	**38.85**	37.58
$\sigma = 20$	**35.70**	35.68	33.88
$\sigma = 30$	**34.08**	33.73	31.65
$\sigma = 40$	**32.86**	32.32	30.05
$\sigma = 50$	**31.85**	31.13	28.80

Table 3. *Comparison of running times.* Time to denoise a color frame of resolution 960×540. Note: values displayed for VNLB do not include the time required to estimate motion.

Method	V-BM4D	VNLB	DVDnet (CPU)	DVDnet (GPU)
Time (s)	156	420	19	8

5. CONCLUSIONS

In this paper, we presented DVDnet, a video denoising algorithm which improves the state-of-the-art. Denoising results of DVDnet feature remarkable temporal coherence, very low flickering, and excellent detail preservation. The algorithm achieves running times which are at least an order of magnitude faster than other state-of-the-art competitors. Although the results presented in this paper hold for Gaussian noise, our method could be extended to denoise other types of noise.

6. REFERENCES

[1] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," 2014, number 8, pp. 2774–2781.

[2] Y. Chen and T. Pock, "Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration," *IEEE Trans. PAMI*, vol. 39, no. 6, pp. 1256–1272, 2017.

[3] K Dabov, A Foi, and V Katkovnik, "Image denoising by sparse 3D transformation-domain collaborative filtering," *IEEE Trans. IP*, vol. 16, no. 8, pp. 1–16, 2007.

[4] M. Lebrun, A. Buades, and J. M. Morel, "A Nonlocal Bayesian Image Denoising Algorithm," *SIAM Journal IS*, vol. 6, no. 3, pp. 1665–1688, 2013.

[5] H.C. Burger, C.J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?," 2012, pp. 2392–2399.

[6] V. Santhanam, V.I. Morariu, and L.S. Davis, "Generalized Deep Image to Image Regression," in *Proc. CVPR*, 2016.

[7] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. IP*, vol. 26, no. 7, pp. 3142–3155, 2017.

[8] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising," *IEEE Trans. IP*, vol. 27, no. 9, pp. 4608–4622, 2018.

[9] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM Trans. Graphics*, vol. 35, no. 6, pp. 1–12, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[11] Xinyuan Chen, Li Song, and Xiaokang Yang, "Deep rnns for video denoising," in *Applications of Digital Image Processing XXXIX*. International Society for Optics and Photonics, 2016, vol. 9971, p. 99711T.

[12] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013, pp. 1310–1318.

[13] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák, "Denoising with kernel prediction and asymmetric loss functions," *ACM Trans. Graphics*, vol. 37, no. 4, pp. 124, 2018.

[14] Anil Christopher Kokaram, *Motion picture restoration*, Ph.D. thesis, University of Cambridge, 1993.

[15] Matteo Maggioni, Giacomo Boracchi, Alessandro Foi, and Karen Egiazarian, "Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms," *IEEE Trans. IP*, vol. 21, no. 9, pp. 3952–3966, 2012.

[16] Pablo Arias and Jean-Michel Morel, "Video denoising via empirical Bayesian estimation of space-time patches," *Journal of Mathematical Imaging and Vision*, vol. 60, no. 1, pp. 70—-93, 2018.

[17] Antoni Buades and Jose-Luis Lisani, "Patch-Based Video Denoising With Optical Flow Estimation," *IEEE Trans. IP*, vol. 25, no. 6, pp. 2573–2586, 2016.

[18] Matias Tassano, Julie Delon, and Thomas Veit, "An Analysis and Implementation of the FFDNet Image Denoising Method," *IPOL*, vol. 9, pp. 1–25, 2019.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *NIPS*, pp. 1–9, 2012.

[20] Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. ICML*. 2015, pp. 448–456, JMLR.org.

[21] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proc. CVPR*, 2016, pp. 1874–1883.

[22] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, "Waterloo Exploration Database: New Challenges for Image Quality Assessment Models," *IEEE Trans. IP*, vol. 26, no. 2, pp. 1004–1016, 2017.

[23] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *IEEE ICCV*, Sydney, Australia, Dec. 2013.

[24] Anna Khoreva, Anna Rohrbach, and Bernt Schiele, "Video object segmentation with language referring expressions," in *ACCV*, 2018.

[25] D.P. Kingma and J.L. Ba, "ADAM: a Method for Stochastic Optimization," *Proc. ICLR*, pp. 1–15, 2015.

[26] ABSoft, "Neat Video," `https://www.neatvideo.com`, 1999–2019.