Model-blind Video Denoising Via Frame-to-frame Training

Thibaud Ehret Axel Davy Jean-Michel Morel
Gabriele Facciolo Pablo Arias
CMLA, ENS Cachan, CNRS
Université Paris-Saclay, 94235 Cachan, France
thibaud.ehret@ens-cachan.fr

Figure 1: From the *same* starting point and *only* using the video, our fine-tuned network is able to denoise different noises without any artifact. The top images are the noisy and the bottom ones the denoised. From left to right: Gaussian noise, Poisson type noise, salt and pepper type noise and JPEG compressed Gaussian noise.

Abstract

Modeling the processing chain that has produced a video is a difficult reverse engineering task, even when the camera is available. This makes model based video processing a still more complex task. In this paper we propose a fully blind video denoising method, with two versions off-line and on-line. This is achieved by fine-tuning a pre-trained AWGN denoising network to the video with a novel frame-to-frame training strategy. Our denoiser can be used without knowledge of the origin of the video or burst and the post-processing steps applied from the camera sensor. The on-line process only requires a couple of frames before achieving visually pleasing results for a wide range of perturbations. It nonetheless reaches state-of-the-art performance for standard Gaussian noise, and can be used off-line with still better performance.

1. Introduction

Denoising is a fundamental image and video processing problem. While the performance of denoising methods and imaging sensors has steadily improved over decades of research, new challenges have also appeared. High-end cameras still acquire noisy images in low lighting conditions. High-speed video cameras use short exposure times, reducing the SNR of the captured frames. Cheaper, lower quality sensors are used extensively, for example in mobile phones or surveillance cameras, and require denoising even with a good scene illumination.

A plethora of approaches have been proposed for image and video denoising: PDE and variational methods [36, 7], bilateral filters [41], domain transform methods [31, 33], non-local patch-based methods [3]. In the last decade, most research focused on modeling image patches [51, 45, 15] or groups of similar patches [13, 27, 22, 17, 5]. Recently the

focus has shifted towards neural networks.

The first neural network with results competitive with patch-based methods was introduced in [5], and consisted of a fully connected network trained to denoise image patches. More recently, [48] proposed DnCNN a deep CNN with 17 to 20 convolutional layers with 3×3 filters and reported a significant improvement over the state-of-the-art. The authors also trained a blind denoising network that can denoise an image with an unknown noise level $\sigma \in [0, 55]$, and a multi-task network that can handle blindly three types of noise. A lighter version of DnCNN was proposed in [49], which allows a spatially variant noise variance by adding the noise variance map $\sigma^2(x)$ as an additional input. The architectures of DnCNN and FFDnet keep the image size throughout the network. Other networks have been proposed [30, 38, 8] that use pooling and up-convolutional layers in a U-shaped architecture [35]. Other works proposed neural networks with an architecture obtained by unrolling optimization algorithms such as those used for MAP inference with MRFs probabilistic models [2, 39, 10, 43]. For textures formed by repetitive patterns, non-local patch-based methods still perform better than "local" CNNs. To remedy this, some attempts have been made to include the non-local patch similarity in a CNN framework [34, 10, 24, 44, 11].

The most widely adopted assumption in the literature is that of additive white Gaussian noise (AWGN). This is justified by the fact that the noise generated by the photon count process at the imaging sensor can be modeled as Poisson noise, which in turn can be approximated by AWGN after a variance stabilizing transform (VST) [1, 29, 28]. However, in many practical applications the data available is not the raw data straight from the sensor. The camera output is the result of a processing pipeline, which can include quantization, demosaicking, gamma correction, compression, etc. The noise at the end of the pipeline is spatially correlated and signal dependent, and it is difficult to model. Furthermore the details of the processes undergone by an image or video are usually unknown. To make things even more difficult, a large amount of images and videos are generated by mobile phone applications which apply their own processing of the data (for example compression, filters, or effects selected by the user). The specifics of this processing are unknown, and might change with different releases.

The literature addressing this case is much more limited. The works [23, 16] address denoising noisy compressed images. RF3D [26] handles correlated noise in infrared videos. Data-driven approaches provide an interesting alternative when modeling is not challenging. CNNs have been applied successfully to denoise images with non-Gaussian noise [48, 8, 18]. In applications in which the noise type is unknown, one could use *model-blind* networks such as DnCNN-3 [48] trained to denoise several types of noise, or the blind denoiser of [18]. These however have two important limitations. First, the performance of such *model-blind* denoising networks very often drops with respect to *model-specific* networks [48]. Second, training the network requires a dataset of images corrupted with each type of noise that we wish to remove (or the ability to generate it synthetically [18]). Generating ground truth data for real photographs is not straightforward [32, 8]. Furthermore, in many occasions we do not have access to the camera, and a single image or a video is all that we have.

In this work we show that, for certain kinds of noise, in the context of video denoising one video is enough: a network can be trained from a single noisy video by considering the video itself as a dataset. Our approach is inspired by two works: the one-shot object segmentation method [6] and the noise-to-noise training proposed in the context of denoising by [25].

The aim of one-shot learning is to train a classifier network to classify a new class with only a very limited amount of labeled examples. Recently Caelles *et al*. [6] suggested a one-shot framework for object segmentation in video, where an object is manually segmented on the first frame and the objective is to segment it in the rest of the frames. Their main contribution is the use of a pre-trained classification network, which is fine-tuned to a manual segmentation of the first frame. This fine-tuned network is then able to segment the object in the rest of the frames. This generalizes the one-shot principle from classification to other types of problems. Borrowing the concept from [6], our work can be interpreted as a one-shot blind video denoising method: a network can denoise an unseen noise type by fine-tuning it to a single video. In our case, however, we do not require "labels" (i.e. the ground truth images without noise). Instead, we benefit from the noise-to-noise training proposed by [25]: a denoising network can be trained by penalizing the loss between the predicted output given a noisy and a second noisy version of the same image, with an independent realization of the noise. We benefit from the temporal redundancy of videos and use the noise-to-noise training between adjacent frames to fine-tune a pre-trained denoising network. That is, the network is trained by minimizing the error between the predicted frame and the past (or future) frame. The noise used to pre-train the network can be very different from the type of noise in the video.

We present the different tools, namely one of the state-of-the-art denoising network DnCNN [48] and a training principle for denoising called noise2noise [25], necessary to derive our refined model in Section 2. We present our truly blind denoising principle in Section 3. We compare the quality of our blind denoiser to the state of the art in Section 4. Finally we conclude and open new perspectives for this type of denoising in Section 5.

2. Preliminaries

The proposed model-blind denoiser builds upon DnCNN and the noise-to-noise training. In this section we provide a brief review of these works, plus some other related work.

2.1. DnCNN

DnCNN [48] was the first neural network to report a significant improvement over patch-based methods such as BM3D [13] and WNNM [17]. It has a simple architecture inspired by the VGG network [40], consisting of 17 convolutional layers. The first layer consists of 64 3×3 followed by ReLU activations and outputs 64 feature maps. The next 15 layers also compute 64 3×3 convolutions, followed by batch normalization [19] and ReLU. The output layer is simply a 3×3 convolutional layer.

To improve training, in addition to the batch normalization layers, DnCNN uses *residual learning*, which means that network is trained to predict the noise in the input image instead of the clean image. The intuition behind this is that if the mapping from the noisy input f to the clean target u is close to the identity function, then it is easier for the network to learn the *residual mapping*, $f \mapsto f - u$.

DnCNN provides state-of-the-art image denoising for Gaussian noise with a rather simple architecture. For this reason we will use it for all our experiments.

2.2. Noise-to-noise training

The usual approach for training a neural network for denoising (or other image restoration problems) is to synthesize a degraded image f_i from a clean one u_i according to a noise model. Training is then achieved by minimizing the empirical risk which penalizes the loss between the network prediction $\mathcal{F}_\theta(f_i)$ and the clean target u_i. This method cannot be applied for many practical cases where the noise model is not known. In these settings, noise cannot be synthetically added to a clean image. One can generate noisy data by acquiring it (for example by taking pictures with a camera), but the corresponding clean targets are unknown, or are hard to acquire [9, 32].

Lehtinen *et al*. [25] recently pointed out that for certain types of noise it is possible to train a denoising network from pairs of noisy images (f_i, g_i) corresponding to the same clean underlying data and independent noise realizations, thus eliminating the need for clean data. This allows learning networks for noise that cannot be easily modeled (an appropriate choice of the loss is still necessary though so that the network converges to a good denoising).

Assume that the pairs (f, u) are distributed according to $p(f, u) = p(u|f)p(f)$. For a dataset of infinite size, the empirical risk of an estimator $\mathcal{F}$ converges to the Bayesian risk, i.e. the expected loss: $\mathcal{R}(\mathcal{F}) = \mathbb{E}_{f,u}\{\ell(\mathcal{F}(u), f)\}$. The optimal estimator F^* depends on the choice of the loss. From Bayesian estimation theory [20] we know that:[1]

$$\ell = L_2 \quad \Rightarrow \quad \mathcal{F}^*(f) = \mathbb{E}\{u|f\} \tag{1}$$

$$\ell = L_1 \quad \Rightarrow \quad \mathcal{F}^*(f) = \text{median}\{u|f\} \tag{2}$$

$$\ell = L_0 \quad \Rightarrow \quad \mathcal{F}^*(f) \approx \text{mode}\{u|f\} \tag{3}$$

Here $\mathbb{E}\{u|f\}$ denotes by the expectation of the posterior distribution $p(u|f)$ given the noisy observation f. During training, the network learns to approximate the mapping $f \mapsto F^*(f)$.

The key observation leading to noise-to-noise training is that the same optimal estimators apply when the loss is computed between $\mathcal{F}(f)$ and g, a second noisy version of u. In this case we obtain the mean, median and mode of the posterior $p(g|f)$. Then, for example if the noise is such that $\mathbb{E}\{g|f\} = \mathbb{E}\{u|f\}$, then the network can be trained by minimizing the MSE loss between $F(f)$ and a second noisy observation g. If the median (resp. the mode) is preserved by the noise, then the L_1 loss (resp. the L_0) loss can be used.

3. Model-blind video denoising

In this section we show how one can use a pre-trained denoising network learned for an arbitrary noise and fine-tune it to other target noise types using a single video sequence, attaining the same performance as a network trained specifically for the target noise for classic noise. This fine tuning can be done off-line (using the whole video as a dataset) or on-line, i.e. frame-by-frame, depending on the application and the computational resources at hand.

As we will show in Section 4, starting from a pre-trained network is key for the success of the proposed training, as we do not have a large dataset available as in [25], but only a single video sequence. The use of a pre-trained network is in part motivated by works on transfer learning such as Zamir *et al*. [47]. Denoising different noise models are related tasks. Our intuition is that a part of the network focuses on the noise type while the rest encodes features of natural images.

Our approach is inspired by the one-shot video object segmentation approach of [6], where a classification network is fine-tuned using the manually segmented first frame, and then applied to the other frames. As opposed to the segmentation problem, we do not assume that we have a ground truth (clean frames). Instead, we adapt the noise-to-noise training to a single video.

We need pairs of independent noisy observations of the same underlying clean image. For that we take advantage of the temporal redundancy in videos: we consider consecutive frames as observations of the same underlying clean

[1] The median and mode are taken element-wise. For a continuous random variable the L_0-loss is defined as a limit. See [20] and [25].

signal transformed by the motion in the scene. To account for the motion we need to estimate it and warp one frame to the other. We estimate the motion using an optical flow. We use the TV-L1 optical flow [46] with an implementation available in [37]. This method is reasonably fast and is quite robust to noise when the flow is computed at a coarser scale.

Let us denote by v_t the optical flow from frame f_t to frame f_{t-1}. The warped f_{t-1} is then $f^w_{t-1}(x) = f_{t-1}(x + v_t(x))$ (we use bilinear interpolation). Similarly, we define the warped clean frame u^w_{t-1}. We assume

(i) that the warped clean frame u^w_{t-1} matches u_t, i.e. $u_t(x) \approx u^w_{t-1}(x)$, and

(ii) that the noise of consecutive frames is independent.

Occluded pixels in the backward flow from t to $t-1$ do not have a correspondence in frame $t-1$. Nevertheless, the optical flow assigns them a value. We use a simple occlusion detector to eliminate these false correspondences from our loss. A simple way to detect occlusions is to determine regions where the divergence of the optical flow is large [4]. We therefore define a binary occlusion mask as

$$\kappa_t(x) = \begin{cases} 0 \text{ if } |\operatorname{div} v_t(x)| > \tau \\ 1 \text{ if } |\operatorname{div} v_t(x)| \leq \tau. \end{cases} \tag{4}$$

Pixels with an optical flow that points out of the image domain are considered occluded. In practice, we compute a more conservative occlusion mask by dilating the result of Eq. (4).

We then compute the loss masking out occluded pixels. For example, for the L_1 loss we have:

$$\ell_1(f, g, \kappa) = \sum_x \kappa(x) \left| f(x) - g(x) \right|. \tag{5}$$

Similarly one can define masked versions of other losses. In the noise-to-noise setting, the choice of the loss depends on the properties of the noise [25]. The noise types that can be handled by each loss in noise-to-noise have a precise characterization (the mean/median/mode of the *noisy posterior* $p(g|f)$ have to be equal to those of the *clean posterior* $p(u|f)$). Verifying this requires some knowledge about noise distribution. In the absence of such knowledge, since the method is reasonably fast, an alternative would be to test different losses and see which one gives the best result.

In principle our method is able to deal with the same noise types as noise-to-noise. In practice we have some limitation imposed by the registration as it degrades for severe noise. For this reason we do not show examples with non-median preserving noise requiring the L_0 loss. For all our experiments we use the masked L_1 loss since it has better training properties than the L_2 [50]. Most relevant noise types often encountered in practice (poisson, jpeg-compressed, low-freq. noise) can be handled by the L_1 loss and the registration.

We now have pairs of images $(f_t, f^w_{t_1})$ and the corresponding occlusion masks κ_t and we apply the noise-to-noise principle to fine-tune the network on this dataset. In order to increase the number of training samples the symmetric warping can also be done, i.e. warping f_{t+1} to f_t using the forward optical flow from f_t to f_{t+1}. This allows to double the amount of data used for the fine-tuning. We consider two settings: off-line and on-line training.

Off-line fine-tuning. We denote the network as a parametrized function $\mathcal{F}_\theta$, where θ is the parameter vector. In the off-line setting we fine-tune the network parameters θ by doing a fixed number N of steps of the minimization of the masked loss over all frames in the video:

$$\theta^{\text{ft}} = \arg\min_\theta{}^{N,\theta_0} \sum_{t=1}^{T} \ell_1(\mathcal{F}_\theta(f_t), f^w_{t-1}, \kappa_t) \tag{6}$$

where by $\arg\min_\theta{}^{N,\theta_0} E(\theta)$ we denote an operator which does N optimization steps of function E starting from θ_0 and following a given optimization algorithm (for instance gradient descent, Adam [21], etc.). The initial condition for the optimization is the parameter vector of the pre-trained network. The fine-tuned network is then applied to the rest of the video.

On-line fine-tuning In the on-line setting we train the network in a frame-by-frame fashion. As a consequence we denoise each frame with a different parameter vector θ^{ft}_t. At frame t we compute θ^{ft}_t by doing N optimization steps corresponding to the minimization of the loss between frames t and $t-1$:

$$\theta^{\text{ft}}_t = \arg\min_\theta{}^{N,\theta^{\text{ft}}_{t-1}} \ell_1(\mathcal{F}_\theta(f_t), f^w_{t-1}, \kappa_t). \tag{7}$$

The initial condition for this iteration is given by the fine-tuned parameter vector at the previous frame θ^{ft}_t. The first frame is denoised using the pre-trained network. The fine-tuning starts for the second frame. A reasonable concern is that the network overfits the given realization of the noise and the frame at each step. This is indeed the case if we use a large number of optimization iterations N at a single frame. A similar behavior is reported in [42], which trains a network to minimize the loss on a single data point. We prevent this from happening by using a small number of iterations (e.g. $N = 20$). We have observed that the parameters fine-tuned at t can be applied to denoise any other frame without any significant drop in performance.

The on-line fine-tuning addresses the problem of *life-long learning* [47] by continuously adapting the network to changes in the distribution of noise and signal. This is particularly useful when the statistics of the noise depend on time-varying parameters (such as imaging sensors affected by temperature).

4. Experiments

In this section we demonstrate the flexibility of the proposed fine-tuning blind denoising approach with several experimental results. For all these experiments the starting point for the fine-tuning process is a DnCNN network trained for an additive white Gaussian noise of standard variation $\sigma = 25$. In all cases we use the same hyper-parameters for the fine tuning: a learning rate of 5.10^{-5} and $N = 20$ iterations of the Adam optimizer. For the off-line case we use the entire video. The videos used in this section come from Derf's database[2]. They've been converted to grayscale by averaging the three color channels and downscaled by a factor two in each direction to ensure that they contain little to no noise. The code and data to reproduce the results presented in this section are available on https://github.com/tehret/blind-denoising.

To the best of our knowledge there is not any other blind video denoising method in the literature. We will compare with state-of-the-art methods on different types of noise. Most methods have been crafted (or trained) for a specific noise model and often a specific noise level. We will also compare with an image denoising method proposed by Lebrun *et al.* [23] which assumes a Gaussian noise model with variance depending on the intensity and the local frequency of the image. This model was proposed for denoising of compressed noisy images. We cannot compare with some more recent blind denoising methods, such as [9], because there is no code available. We will compare with DnCNN [48] and VBM3D [12]. VBM3D is a video denoising algorithm. All the other methods are image denoising applied frame-by-frame (perspectives for videos are mentioned in Section 5).

The goal of the first experiment is to compare against reference networks trained for these noises the regular way. The per-frame PSNRs are presented in Figure 2. We applied the proposed learning process to a sequence contaminated with AWGN with standard deviation $\sigma = 25$, which is precisely the type of noise the network was trained on and verified that it does not deteriorate the pre-training. The off-line fine-tuning performs on par with the pre-trained network. The PSNR of the on-line process has a higher variance, with some significant drops for some frames. For $\sigma = 50$, we can see that both fine-tuned networks perform better than the pre-trained network for $\sigma = 25$. In fact their performance

[2]https://media.xiph.org/video/derf/

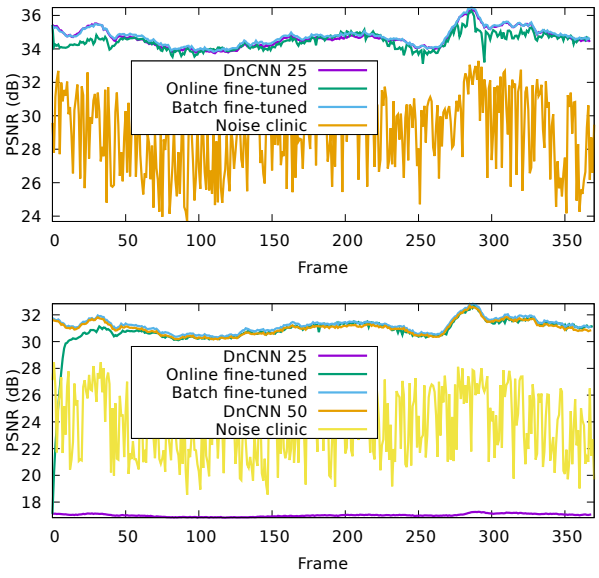

Figure 2: The fine-tuning process is done on a sequence corrupted by an additive Gaussian noise of standard deviation $\sigma = 25$ (top) or $\sigma = 50$ (bottom). The fine-tuned networks (offline and online) achieve comparable performance than the reference networks.

is as good as the DnCNN network trained specifically for $\sigma = 50$ (actually the off-line trained performs even slightly better than the reference network). Our process also outperforms the "noise clinic" of [23].

We have also tested the proposed fine-tuning on four other types of noise: multiplicative Gaussian, correlated, salt and pepper and compressed Gaussian. We present the corresponding per-frame PSNRs in Figure 3. The multiplicative Gaussian noise is given by

$$f_t(x) = u_t(x) + r_t(x)u_t(x), \tag{8}$$

where $r_t(x)$ is white Gaussian noise with standard deviation of $75/255$ (the images are within the range [0,1]). The resulting variance $\sigma_t^2(x)$ depends on the pixel intensity $u_t(x)$. The correlated noise is obtained by convolving AWGN with a disk kernel. The resulting standard deviation is $\sigma = 25$. The salt and pepper uniform noise is like the one used [25], obtained by replacing with probability 0.25 the value of a pixel with a value sampled uniformly in $[0, 1]$. Finally, the compressed Gaussian noise, results from compressing an image corrupted by an AWGN of $\sigma = 25$ with JPEG. The last one is particularly interesting because it is a realistic use case for which the noise model is quite hard to estimate [16]. While in this case the noise can be generated synthetically for training a network over a dataset, this is not possible with other compression tools (for example for proprietary technologies). We can see the effectiveness of the

Method	walk	crowd	football	station	Average
DnCNN 25	17.02	11.24	15.09	13.86	14.30
DnCNN 50	31.02	**25.83**	**31.67**	30.09	29.65
Online fine-tuned	30.84	25.58	31.33	29.90	29.59
Batch fine-tuned	**31.22**	**25.83**	31.54	**30.39**	**29.75**
Noise Clinic	23.85	22.13	24.57	24.39	23.74
VBM3D	*31.57*	*27.02*	*31.97*	*31.33*	*30.47*

Table 1: PSNR values for 4 sequences with AGWN of standard deviation $\sigma = 50$.

Method	walk	crowd	football	station	Average
DnCNN 25	32.62	**27.31**	32.48	31.48	30.97
Online fine-tuned	32.86	27.20	32.79	30.88	30.94
Batch fine-tuned	**33.28**	27.19	**32.91**	**31.58**	**31.24**
Noise Clinic	27.62	25.17	27.20	26.89	26.72
VBM3D	*34.16*	*28.95*	*33.83*	*33.53*	*32.62*

Table 2: PSNR values on JPEG compressed AWGN noise with $\sigma = 25$ and compression factor 10.

fine-tuning for all examples. The off-line training is more stable (smaller variance) and gives slightly better results, although the difference is small.

A visual comparison with other methods is shown in Figure 4 for JPEG compressed noise and in Figure 5 for AWGN with $\sigma = 50$. Visual examples on real data are presented in the supplementary material. The result of the fine-tuned network has no visible artifacts and is visually pleasing even though the network has never seen this type of noise before the fine-tuning. A limitation of the method is the oversmoothing of texture. Indeed DnCNN has a tendency of oversmoothing textures. Using a network designed for video denoising should help recover more texture and improve temporal consistency [14]. Another cause is the optical flow. Since it is computed on downscaled noisy frames it is imprecise around edges. This leads to false correspondences between frames and introduces some blur. Improved registration should lead to sharper results.

In Tables 1 and 2 we show the PSNR of the results obtained on 4 sequences for AWGN of $\sigma = 50$ and JPEG compressed AWGN of $\sigma = 25$ and compression factor 10. For the case of AWGN the fine-tuned networks attain the performance of the DnCNN trained for that specific noise. For JPEG compressed Gaussian noise, the batch fine-tuned network is on average $0.3dB$ above the pre-trained network.

Figure 6 shows the impact of different parameters of the method. The main parameters of the proposed training are the learning rate and the number of per-frame iterations. Fewer iterations require more frames for convergence. In turn the result has smaller variance. A similar analysis can be done for the learning rate. We also show the importance of using a pre-trained network compared to a random initialization. There is a $2dB$ gap in favor of the pre-trained network. The other important parameter is the number of frames used for fine-tuning. The fine-tuning is stopped at a frame t_0 and $\theta^{\text{ft}}_{t_0}$ is used to process the remaining frame. We can see that the more frames used for the fine-tuning the better the performance.

Finally Figure 7 shows examples of lifelong learning. The first example shows a slowly evolving noise (starting with a Gaussian noise with standard deviation $\sigma = 25$ that linearly increases up to $\sigma = 50$). The fine-tuned network performs better than the two reference networks for respectively $\sigma = 25$ and $\sigma = 50$. The second example shows a sudden change (starting with a Gaussian noise with standard deviation $\sigma = 50$ and changes to Salt and pepper noise at frame 200). In that case the fine-tuned network adapts quickly to the new noise model.

The running time depends on the network. We used DnCNN but other networks can be used instead and trained with the proposed method. Each fine-tuning iteration runs a back-propagation step which takes $0.33s$ on a NVIDIA Titan XP for DnCNN for a 960×540 frame. Fifty frames with 20 iterations per frame take 5 mins. For comparison, training DnCNN from scratch over a dataset requires around $6h$ (and a dataset). By using a lighter network and reducing the per-frame-iterations it might be possible to achieve real time frame rates. Moreover, the fine-tuning can be done on a fixed number of frames at the beginning or run in the background each number of frames for cases when the computational efficiency is important.

5. Discussion and perspectives

Denoising methods based on deep learning often require large datasets to achieve state-of-the-art performance. Lehtinen *et al*. [25] pointed out that in many cases the clean ground truth images are not necessary, thus simplifying the acquisition of the training datasets. With the framework presented in this paper we take a step further and show that a single video is often enough, removing the need for a dataset of images. By applying a simple frame-to-frame training on a generic pre-trained network (for example a DnCNN network trained for additive Gaussian noise with fixed standard deviation), we successfully denoise a wide range of different noise models even though the network has *never* seen the video nor the noise model before its fine-tuning. This opens the possibility to easily process data from any unknown origin.

We think that the current fine tuning process can still be

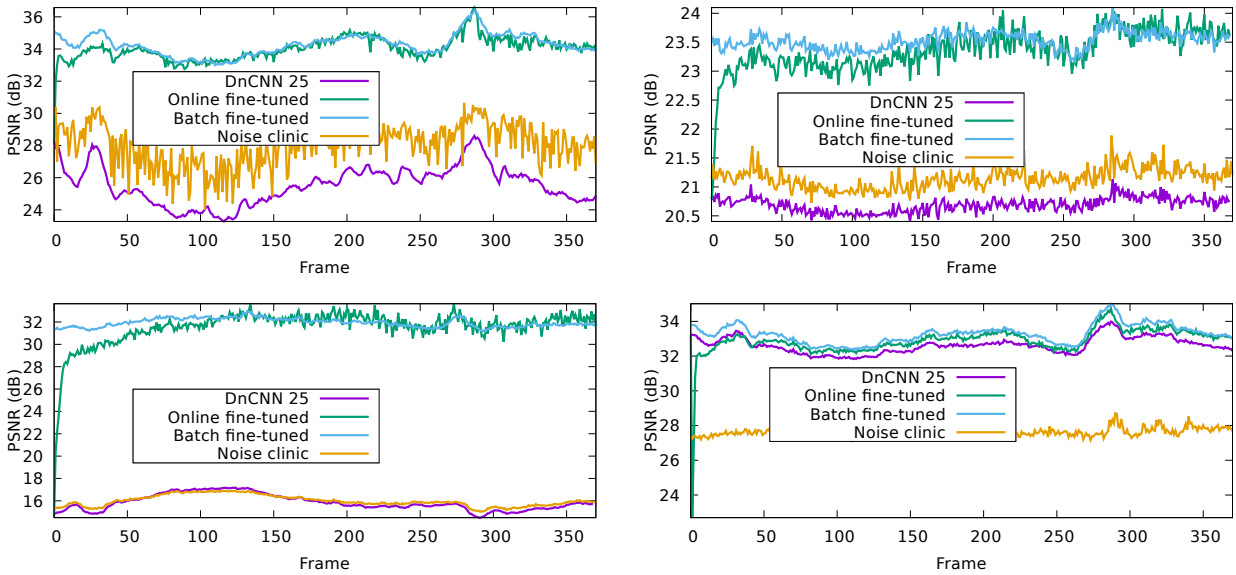

Figure 3: Different types of noise. From top-left to bottom right: multiplicative Gaussian noise, correlated Gaussian noise, salt and pepper noise, Gaussian noise after JPEG compression. The fine-tuned network (both online and batch) always performs better than the original network.

Figure 4: Example of denoising of an image corrupted by a JPEG compressed Gaussian noise. The fine-tuned network doesn't produce any visible artifacts, contrary to the original DnCNN used for the fine-tuning process. From left to right, top to bottom: Noisy, fine-tuned, VBM3D, ground truth, DnCNN trained for a Gaussian noise, noise clinic.

improved. First, given that the application is video denoising, it is expected that better results will be achieved by a video denoising network (the DnCNN network processes each frame independent of the others). Using the temporal information could improve the denoising quality, just like video denoising methods improve over frame-by-frame image denoising methods, but also might stabilize the variance of the result for the on-line fine-tuning.

Acknowledgment

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp

Figure 5: Example of denoising of an image corrupted by a Gaussian noise of standard deviation $\sigma = 50$. The fine-tuned network doesn't produce any visible artifact, the results are comparable to a DnCNN trained for this particular type of noise. From left to right, top to bottom: Noisy, fine-tuned, DnCNN trained for a Gaussian noise with $\sigma = 50$, VBM3D, ground truth, noise clinic, DnCNN trained for a Gaussian noise with $\sigma = 25$.

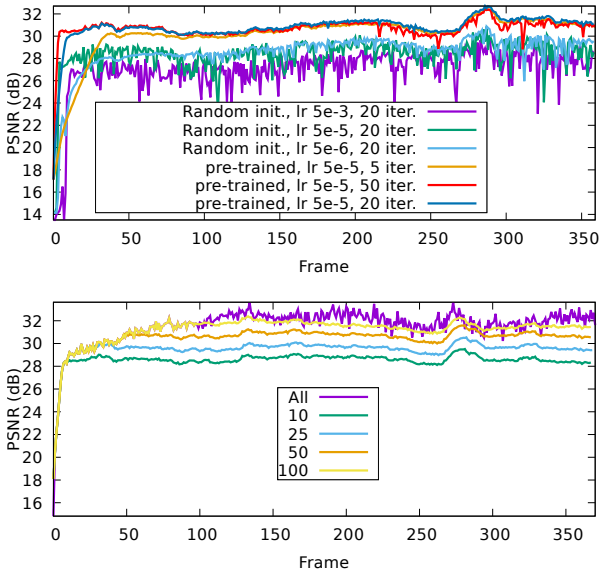

Figure 6: Impact of parameters. Top: Impact of the learning rate and the number of iterations. It also shows the gap between using a pre-trained network and random initialization. Bottom: Impact of the number of frames used for fine-tuning.

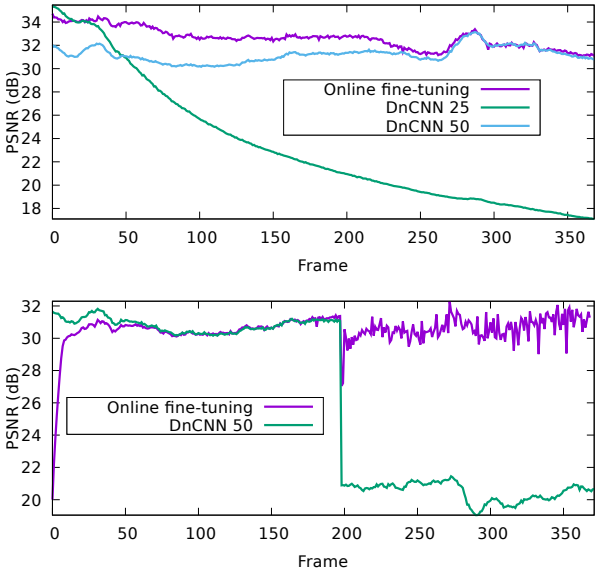

Figure 7: Lifelong learning. Top: Slow change. Bottom: sudden change. The fine-tuned network adapts without difficulty to slow changes and sudden changes. See text for more details

GPU used for this research. Work partly financed by IDEX Paris-Saclay IDI 2016, ANR-11-IDEX-0003-02, Office of Naval research grant N00014-17-1-2552, DGA Astrid project «filmer la Terre» n^o ANR-17-ASTR-0013-01, MENRT.

References

[1] F. J. Anscombe. The transformation of poisson, binomial and negative-binomial data. *Biometrika*, 35(3/4):246–254, 1948. 2

[2] A. Barbu. Training an active random field for real-time image denoising. *Transactions on Image Processing*, 18(11):2451–2462, 2009. 2

[3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *CVPR*, volume 2, pages 60–65. IEEE, 2005. 1

[4] A. Buades, J.-L. Lisani, and M. Miladinović. Patch-based video denoising with optical flow estimation. *Transactions on Image Processing*, 25(6):2573–2586, 2016. 4

[5] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *CVPR*, pages 2392–2399. IEEE, 2012. 1, 2

[6] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, pages 221–230. IEEE, 2017. 2, 3

[7] A. Chambolle and P.-L. Lions. Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188, 1997. 1

[8] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to see in the dark. In *CVPR*, pages 3291–3300. IEEE, 2018. 2

[9] J. Chen, J. Chen, H. Chao, and M. Yang. Image blind denoising with generative adversarial network based noise modeling. In *CVPR*, pages 3155–3164. IEEE, 2018. 3, 5

[10] Y. Chen and T. Pock. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration. *Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 6 2017. 2

[11] C. Cruz, A. Foi, V. Katkovnik, and K. Egiazarian. Nonlocality-reinforced convolutional neural networks for image denoising. *Signal Processing Letters*, 25(8):1216–1220, 2018. 2

[12] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3d transform-domain collaborative filtering. In *European Signal Processing Conference*, pages 145–149. IEEE, 2007. 5

[13] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3d filtering. In *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064, page 606414. International Society for Optics and Photonics, 2006. 1, 3

[14] A. Davy, T. Ehret, G. Facciolo, J. Morel, and P. Arias. Non-local video denoising by CNN. *CoRR*, abs/1811.12758, 2018. 6

[15] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Transactions on Image processing*, 15(12):3736–3745, 2006. 1

[16] M. González, J. Preciozzi, P. Musé, and A. Almansa. Joint denoising and decompression using cnn regularization. In *CVPR Workshops*, pages 2598–2601. IEEE, 2018. 2, 5

[17] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, pages 2862–2869. IEEE, 2014. 1, 3

[18] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang. Toward convolutional blind denoising of real photographs. *CVPR*, 2019. 2

[19] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456. PMLR, 2015. 3

[20] S. Kay. Fundamentals of statistical processing, volume i: Estimation theory: Estimation theory v. 1, 1993. 3

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[22] M. Lebrun, A. Buades, and J.-M. Morel. A Nonlocal Bayesian Image Denoising Algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013. 1

[23] M. Lebrun, M. Colom, and J.-M. Morel. The noise clinic: a blind image denoising algorithm. *Image Processing On Line*, 5:1–54, 2015. 2, 5

[24] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *CVPR*, pages 5882–5891. IEEE, 2017. 2

[25] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2noise: Learning image restoration without clean data. In *ICML*, pages 2971–2980. PMLR, 2018. 2, 3, 4, 5, 6

[26] M. Maggioni, E. Sánchez-Monge, and A. Foi. Joint removal of random and fixed-pattern noise through spatiotemporal video filtering. *Transactions on Image Processing*, 23(10):4282–4296, 2014. 2

[27] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *CVPR*, pages 2272–2279. IEEE, 2009. 1

[28] M. Makitalo and A. Foi. A closed-form approximation of the exact unbiased inverse of the anscombe variance-stabilizing transformation. *Transactions on Image Processing*, 20(9):2697–2698, 2011. 2

[29] M. Makitalo and A. Foi. Optimal inversion of the anscombe transformation in low-count poisson image denoising. *Transactions on Image Processing*, 20(1):99–109, 2011. 2

[30] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2802–2810. Curran Associates, Inc., 2016. 2

[31] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes using generalized Gaussian and complexity priors. *Transactions on Information Theory*, 45(3):909–919, 1999. 1

[32] T. Plotz and S. Roth. Benchmarking Denoising Algorithms with Real Photographs. In *CVPR*, pages 1586–1595. IEEE, 2017. 2, 3

[33] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *Transactions on Image Processing*, 12(11):1338–1351, 2003. 1

[34] P. Qiao, Y. Dou, W. Feng, R. Li, and Y. Chen. Learning non-local image diffusion for image denoising. In *International Conference on Multimedia*, pages 1847–1855. ACM, 2017. 2

[35] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[36] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992. 1

[37] J. Sánchez Pérez, E. Meinhardt-Llopis, and G. Facciolo. Tv-l1 optical flow estimation. *Image Processing On Line*, 2013:137–150, 2013. 4

[38] V. Santhanam, V. I. Morariu, and L. S. Davis. Generalized deep image to image regression. In *CVPR*, pages 5609–5619. IEEE, 2017. 2

[39] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *CVPR*, pages 2774–2781. IEEE, 2014. 2

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[41] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846. IEEE, 1998. 1

[42] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *CVPR*, pages 9446–9454. IEEE, 2018. 4

[43] R. Vemulapalli, O. Tuzel, and M. Liu. Deep gaussian conditional random field network: A model-based deep network for discriminative denoising. In *CVPR*, pages 4801–4809. IEEE, 2016. 2

[44] D. Yang and J. Sun. Bm3d-net: A convolutional neural network for transform-domain collaborative filtering. *Signal Processing Letters*, 25(1):55–59, 2018. 2

[45] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. *Transactions on Image Processing*, 21(5), 2012. 1

[46] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*. Springer, 2007. 4

[47] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018. 3, 5

[48] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *Transactions on Image Processing*, 26(7):3142–3155, 7 2017. 2, 3, 5

[49] K. Zhang, W. Zuo, and L. Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *Transactions on Image Processing*, 27(9):4608–4622, 2018. 2

[50] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss Functions for Image Restoration With Neural Networks. *Transactions on Computational Imaging*, 3:47–57, 3 2017. 4

[51] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, pages 479–486. IEEE, 2011. 1