VidCNN

Learning Blind Video Denoising

by

Michele Claus

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 30, 2018 at 1:00 PM.

Student number:	4754751	
Project duration:	February 5, 2018 – August 29, 2018	
Thesis committee:	Dr. Marcel J.T. Reinders,	EEMCS-Interactive Intelligence, TU Delft
	Dr. Jan van Gemert,	EEMCS-Interactive Intelligence, TU Delft
	Dr. Cynthia Liem,	EEMCS-Interactive Intelligence, TU Delft
	Dr. Ildiko Suveg,	Bosch Security Systems, Eindhoven
	Mr. Chengqiu Zhang,	Bosch Security Systems, Eindhoven

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

eit Digital

BOSCH Security Systems

Contents

Acronyms

ADC Analog to Digital Converter.

AWGN Additive White Gaussian Noise.

BN Batch Normalization.

BSD Berkeley Segmentation Dataset.

CNN Convolutional Neural Network.

DND Darmstadt Noise Dataset.

HD High Definition.

LSE Least Squares Error.

MRF Markov Random Field.

NN Neural Network.

NSS Non-local Self Similarity.

PSN Photon Shot Noise.

PSNR Peak Signal to Noise Ratio.

ReLU Rectified Linear Unit.

RNN Recurrent Neural Network.

SSIM Structural Similarity.

UHD Ultra High Definition.

VidCNN - Learning Blind Video Denoising

Michele Claus
TU Delft - EEMCS
Mekelweg 4, Delft, The Netherlands
claus.michele@hotmail.it

Abstract

We propose a novel Convolutional Neural Network (CNN) for Video Denoising called VidCNN, which is capable to denoise videos without prior knowledge on the noise distribution (Blind). VidCNN is a flexible model, since it tackles multiple noise types, artificial and real. The CNN architecture uses a combination of spatial and temporal filtering, which learns how to spatially denoise the frames first and how to combine their temporal information, handling camera and objects motion, brightness changes, low-light conditions and temporal inconsistencies at the same time. We demonstrate the importance of the data used for CNNs training, creating for this purpose a specific dataset for low-light conditions. We test VidCNN on videos commonly used for benchmarking and on self-collected data, achieving good results comparable with the state-of-the-art in video denoising. Our model can be easily adapted to different noise models, keeping the same temporal denoising network, maintaining performance in terms of accuracy and speed.

1. Introduction

Image and video denoising are classical computer vision tasks, which aim to obtain the original signal X from the available noisy observations Y. Noise can influence greatly not only the perceived visual quality, but also segmentation tasks [1] and compression algorithms [2]. Thus, denoising is an important step for subsequent processes. With X as the original signal, N as the noise and Y as the available noisy observation, the noise degradation model can be described as $Y = X + N$, for an additive type of noise, or as $Y = H(X) + N$, if the noise model is signal dependent, with H as the degradation function. In low-light conditions occurs typically signal dependent noise, which is more visible in dark regions than bright areas. Noise is present in all imaging system due to thermal effects, sensor imperfections and low-light conditions.

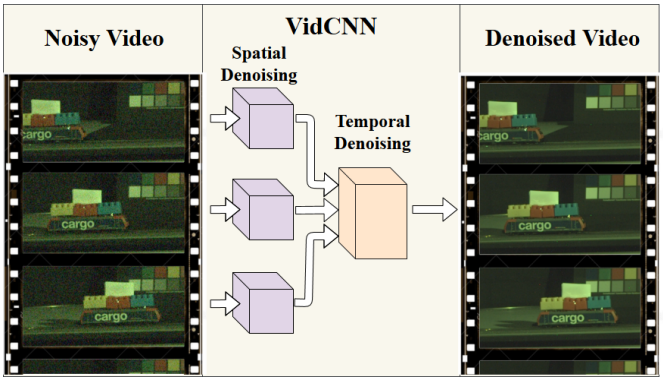


Figure 1: The VidCNN approach to Video Denoising: combining two similar networks performing first Single Frame Spatial Denoising and subsequently Temporal Denoising over a window of three frames, all in a single feed-forward process.

Currently, when developing new cameras in order to apply spatial and temporal filtering, the tuning of multiple filters parameters is required for each gain level. This phase takes time and effort, but is fundamental to get the best result in terms of quality and bandwidth usage. Therefore, in this paper, we focus on automating the denoising procedure with a Convolutional Neural Network for flexible and efficient video denoising, capable to blindly remove artificial and real noise.
Having a noise removal algorithm working in "blind" conditions is essential, since in a real-world scenario, color and light conditions can change suddenly, producing a different noise distribution for each frame.
Nowadays, we can distinguish between two main types of denoising algorithms: based on statistics or based on (deep) learning. Statistical models include the algorithms modeling the image priors for denoising purposes: various mathematical approaches have been used, including Markov Random Field models [3], gradient models [4], sparse models [5] and Nonlocal Self-Similarity (NSS). The latter is currently used in state-of-the-art techniques such as BM3D [6],

LSSC [7], NCSR [8] and WNNM [9]. Even though they achieve respectable denoising performance, most of those algorithms have some drawbacks. Firstly, they are generally designed to tackle specific noise models and levels, limiting their usage in blind denoising. Secondly, they include many handcrafted parameters and complex optimization procedures. Instead, Convolutional Neural Networks are able to mimic complex procedures as denoising in a feed-forward process, learning the parameters in the training phase.

On one hand, a lot of work has been done in the image denoising field. On the other hand, few algorithms have been specifically designed for videos. Typically, video frames are strongly correlated, and the information redundancy is the key assumption for video denoising algorithms. The most basic video denoising technique consists in taking the temporal average over various subsequent frames. Even though it works for steady scenes, it blurs motion parts generating artifacts.

VBM4D [10], from the same authors of BM3D [6], is considered the current state-of-the-art in video denoising. The main difference from the image version consist of the search of similar patches, not only in spatial but also in temporal domain. However, searching for similar patches in more frames drastically increases the processing time.

We propose a convolutional neural network for blind video denoising (VidCNN), capable to denoise videos with synthetic or real noise without prior knowledge over the noise model and the video content. For comparison purpose, experiments have been run on publicly available and on self captured videos, which will also publicly released at the project web page [1].

The main contributions of our work are:

- We created a novel CNN architecture capable to blind denoise videos, combining spatial and temporal information of multiple frames with one single feed-forward process.
- We demonstrate the flexibility of VidCNN, testing it on Additive White Gaussian Noise (AWGN) and real data in low-light conditions.
- We further show that VidCNN can handle motion in challenging situations, detecting temporal inconsistencies and using only the correct information to improve the final outcome.
- We created a low-light dataset for a specific Bosch security camera, with sample pairs of noise free and noisy images, using a simple yet efficient technique to get clean raw data.

[1]Project web page: https://github.com/clausmichele/VidCNN---Learning-Blind-Video-Denoising

2. Related Work

2.1. Background

A great step forward in the Computer Vision world has been achieved with Convolutional Neural Networks (CNNs). CNNs are inspired by the organization of the animal visual cortex: single neurons respond to stimuli only in a limited region of the visual field known as Receptive Field. The application to the image and video denoising fields are recent but based on the same concepts.

2.2. CNNs for Image Denoising

The first work on image denoising using CNNs is from Jain and Seung [11] published in 2008. From that date onwards, there have been enormous improvements of the proposed method and architecture, thanks to newly available hardware with more computational power and high quality datasets recently released. In 2012, Burger *et al*. [12] showed how even a simple Multi Layer Perceptron can obtain comparable results with BM3D [6], even though a huge dataset was required for training [13]. Recently, in 2016, Zhang *et al*. [14] used residual learning and Batch Normalization [15] for image denoising in their DnCNN architecture. With its simple yet effective architecture, it has shown to be flexible for tasks as blind Gaussian denoising, JPEG deblocking and image inpainting. The same research group of DnCNN released in 2017 a new CNN based image denoiser called FFDNet [16], which compared to the previous can handle an extended range of noise levels and has the ability to remove spatially variant noise, *i.e.* images containing parts with different amounts of Gaussian noise. However, it does not additionally tackle JPEG deblocking and image inpainting at the same time as DnCNN does. Ulyanov *et al*. [17] showed how, with their Deep Prior, they can enhance a given image with no prior training data other than the image itself, which can be seen as a "blind" denoising. There have been also some works on CNNs directly inspired by BM3D such as [18, 19]. In [20], Ying *et al*. propose a deep persistent memory network called MemNet that obtains valid results, introducing a memory block, motivated by the fact that human thoughts are persistent. However, the network structure remains complex and not easily reproducible. A recent CNN architecture consisting in an encoder-decoder with skipping connections, the U-Net, has been successfully used for image denoising in the work of Xiao-Jiao *et al*. [21] and in the most recent work on image denoising of Guo *et al*. [22] called CBDNet. With their novel approach, CBDNet reaches extraordinarily results in real world blind image denoising. NVIDIA, in cooperation with the Aalto University, recently proposed an innovative CNN model for blind image denoising. Their Noise2Noise [23], based on the encoder-decoder structure, obtains almost the same result using only noisy images for training,

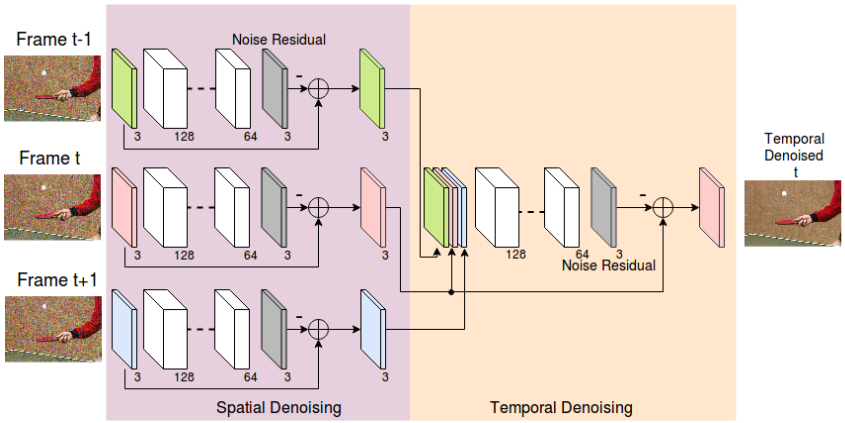


Figure 2: The architecture of the proposed VidCNN network. Every frame will go through a spatial denoising CNN. The temporal CNN takes as input three spatially denoised frames and outputs the final estimate of the central frame. Both CNNs estimate first the noise residual, *i.e.* the unwanted values noise adds to an image, and then subtracts them from the noisy input ($\oplus$ means addition of the two signals, and "-" the negation). VidCNN is composed only by Convolutional Layers. The number of feature maps is written at the bottom of each layer.

instead of clean-noisy pairs. Even though this could be particularly useful for cases where the ground truth is not available, such as low-light conditions, there is no explicit test on this noise type. All these new methods show how much the image denoising field has advanced during only ten years thanks to neural networks. However, our goal is video denoising, which adds other constraints to the process.

2.3. Video and Deep Neural Networks

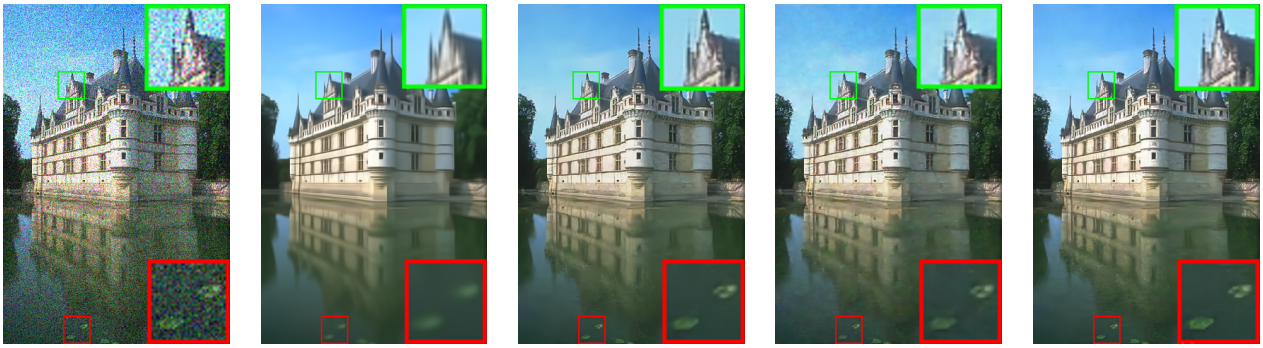
Video denoising using deep learning is still an underexplored research area. The seminal work of Xinyuan *et al*. [24], is currently the only one using neural networks (Recurrent Neural Networks in this case) to address video denoising. Even though neural networks have great generalization capabilities, their algorithm works only on gray level videos with Additive White Gaussian Noise (AWGN) and did not achieve state-of-art results. Here we present a method addressing color video denoising, with comparable results to the state-of-art. Similar tasks have been addressed using CNNs, such as Video Frame Enhancement, Interpolation, Deblurring and Super-Resolution. The key component in all those applications is how to handle motion and temporal changes. For frame interpolation, Niklaus *et al*. [25] use a pre-computed optical flow to feed motion information to a frame interpolation CNN. Meyer *et al*. [26] use instead phase based features to describe motion. Caballero *et al*. [27] developed a network which estimate the motion by itself for video super resolution. Similarly, in Multi Frame Quality Enhancement (MFQE), Yang *et al*. [28] use a Motion Compensation Network and a Quality Enhancement Network, considering three non-consecutive frames for H265 compressed videos. Specifically for video deblurring, Su *et al*. [29] developed a network called DeBlurNet: a U-Net CNN which takes three frames stacked together as input. Similarly, we also use three stacked frames in our VidCNN. Additionally, we have also investigated the use of different frame numbers and present the comparison results.

2.4. Real World Datasets

An image or video denoising algorithm, has to be effective on real world data to be successful. However, it is hard to obtain the ground truth for real pictures, since perfect sensors and channels do not exist. In 2014, Anaya and Barbu, created a dataset for low-light conditions called RENOIR [30]: they use different exposure times and ISO levels to get noisy and clean images of the same static scene. Similarly, in 2017, Plotz and Roth created a dataset called DND [31]. In this case, only the noisy samples have been released, whereas the noise free ones are kept undisclosed for benchmarking purposes. Recently, two other related papers have been published. The first, written by Abdelhamed *et al*. [32] concerns the creation of a smartphone image dataset of noisy and clean images, which at the time of writing is not yet publicly available. The second, written by Chen *et al*. [33], presents a new CNN based algorithm capable to enhance the quality of low-light raw images. They created a dedicated dataset of two camera types similarly to [30].

3. Proposed Method

In this section, we present and describe the proposed blind video denoising CNN architecture VidCNN. It is composed by two main subnetworks: spatial and temporal denoising CNN.



(a) Noisy frame (18.54/0.5225) (b) CBM3D[34] (29.26/0.9194) (c) DnCNN-B[14] (28.72/0.9355) (d) CBDNet[22] (28.48/0.9161) (e) Our result (30.37/0.9361)

Figure 3: Comparison of blind spatial denoising of an image from the CBSD68 dataset corrupted with 2, with Ag=64 and Dg=4. AWGN based method as CBM3D and DnCNN does not achieve optimal result. The first blurs excessively the image. CBDNet generates many artifacts. Using the proper noise model for training leads to a better result. (PSNR [dB]/SSIM)

3.1. Spatial Denoising CNN

We started following [14], which showed great flexibility tackling multiple degradation types at the same time, and experimented with the same architecture for blind spatial denoising. It is shown, that this architecture can achieve state-of-art results for Gaussian denoising. We found out that using a first layer of depth 128 helps when the network has to handle different noise models a the same time. So, the principal difference is the first convolutional layer, which has 128 feature maps instead of 64. The network depth is set to 20 and Batch Normalization (BN) [15] is used. The activation function is ReLU (Rectified Linear Unit). We also investigated the use of Leaky ReLU as activation function, which can be more effective [35], without improvement over ReLU. Comparison results are provided in the supplementary material. Our Spatial-CNN uses Residual Learning, which has been firstly introduced in [14] to tackle image denoising. Instead of forcing the network to output directly the denoised frame, the residual architecture predicts the residual image, which consist in the difference between the original clean image and the noisy observation. The loss function is the L2-norm in equation 1, also known as least squares error (LSE). It is the sum of the square of the differences S between the target value Y and the estimated values Y_{est}. In this case the difference S represents the noise residual image estimated by the Spatial-CNN.

$$L = \sum_{x}\sum_{y}\Big(\underbrace{Y(x,y) - Y_{est}(x,y)}_{\text{Noise Residual}}\Big)^2 \qquad (1)$$

3.1.1 Real Noise Model

The denoising performance of a spatial denoising CNN depends greatly on the data used for training. Real noise distribution differs from Gaussian, since it is not purely additive but it contains a *signal dependent* part. For this reason, CNN models trained only on Additive White Gaussian Noise (AWGN) fail to denoise real world images [22]. Our goal is to achieve a good balance between performance and flexibility, using the same trained network for multiple noise models. As shown in Table 1, our Spatial-CNN can handle blind gaussian denoising: we will further investigate its generalization capabilities, introducing a signal dependent noise model. This specific noise model, in equation 2, is composed by two main contributions, the Photon Shot Noise (PSN) and the Read Noise. The PSN is the main noise source in low-light condition, where N_{sat} accounts the saturation number of electrons. The Read Noise is mainly due to the quantization process in the Analog to Digital Converter (ADC), used to transform the analog light signal into a digital image. $CT1_n$ represents the normalized value of the noise contribution due to the Analog Gain, whereas $CT2_n$ represents the additive normalized part.

$$M = \sqrt{\underbrace{\frac{Ag * Dg}{N_{sat} * s}}_{\text{PSN}} + \underbrace{Dg^2 * (Ag * CT1_n + CT2_n)^2}_{\text{Read Noise}}} \qquad (2)$$

$$NoisyImage = s + \mathcal{N}(0,1) * M(s) \qquad (3)$$

Equation 2 represent the real noise model, where the relevant terms for the considered Sony sensor are: Ag (Analog Gain), in range [0,64], Dg (Digital Gain), in range [0,32] and s, the image that will be degraded. The remaining values are $CT1_n$=1.25^{-4}, $CT2_n$=1.11^{-4} and N_{sat}=7489. The noisy image is generated multiplying observations of a normal distribution $\mathcal{N}(0,1)$ with the same shape of the reference image s, with the Noise Model M in equation 3. In Figure 3 we can appreciate how algorithms based on AWGN such as CBM3D and DnCNN do not achieve an optimal result on a different noise model. The first, in its blind version, i.e. with the supposed AWGN standard deviation

σ set to 50, over-smooths the image, getting a low SSIM (Structural Similarity, the higher the better) score, whereas DnCNN preserves more structure. The recent CBDNet, scores lower than the others in this scenario. Our result shows that, to get the best denoising results, a proper training dataset with the correct noise model is necessary.

3.2. Temporal Denoising CNN

The temporal denoising part of VidCNN is similar in structure to the spatial one, having the same number of layers and feature maps. However, in this case we will consider three frames stacked together as input.
We consider only three frames for several reasons:

- Other works in the literature considered three frames [27, 28, 29] for similar applications.
- We want to keep the network as efficient as possible, lowering the needed memory and computational resources.
- Empirical results show that considering more frames, i.e. five, does not guarantee a better result.

Considering a frame with dimensions $w \times h \times c$, the new input will have dimension of $w \times h \times 3c$. The neural network will learn how to combine the previous and the following frame to enhance the quality of the current one, taking care of temporal inconsistencies. As the Spatial-CNN, even this one uses Residual Learning and will estimate the noise residual image of the central input frame, combining the information of three subsequent frames. With our work we show that residual learning is an efficient solution not only for image denoising, but also for video denoising.

4. Experiments

4.1. Low-Light Dataset Creation

A dataset for image denoising consist in pairs of clean and noisy images. For low-light conditions, creating couples of noisy and noise-free images is more challenging and the publicly available data is scarce. We used the Renoir Dataset [30] and additionally our self-collected dataset. Josue Anaya and Adrian Barbu [30] propose to use two different ISO values and exposure times to get reference and distorted images. However, many camera settings and parameters are involved in this process. We wanted to use a simpler process: grabbing many noisy images of the same scene and then simply averaging to get an estimated ground truth. We used a Bosch Autodome IP 5000 IR, a security camera capable to record raw images, *i.e.* without any type of processing. The setting involved a static scene and a light source with color temperature $3460K$, which has variable intensity between 0 and 255. We varied the light intensity in 12 steps, from the lowest acceptable light condition of value 46, below of which the camera showed noise only, up to the maximum with value 255. For every different light intensity, we recorded 200 raw images in a row. This process, even if simple and effective for our purpose, requires much disk space and is time consuming.
Additionally, we recorded six video sequences in different light conditions, consisting in three or four frames with moving objects or light changes: for each frame we had to record 200 images, which results in a total of 4200 images. To show VidCNN performance, we could just record one image per frame. However, we needed a reference ground truth for comparison with other denoising algorithms.

(a) Low-light Noisy Image (b) Reference Ground Truth

Figure 4: Sample detail of noisy-clean image pairs of our own low-light dataset, collected with a Bosch Autodome IP 5000 IR security camera. The ground truth is obtained averaging 200 raw images collected in the same light conditions.

4.2. Spatial CNN Training

The training phase of VidCNN is divided in two main parts: firstly, we train the spatial denoising CNN and later, when we achieve a satisfying result with a steady loss function, we train the temporal denoising CNN. Our ideal model has to tackle multiple degradation types at the same time, such as AWGN and real noise model (as in equation 3) including low-light conditions. During the training phase, our neural network will learn how to estimate the residual noise content of the input noisy image, using the clean one as reference. Therefore, we require couples of clean and noisy images. which are easily created for AWGN and the real noise model in equation 3. For those two degradation types we use the Waterloo Exploration Dataset [36], containing 4744 pristine images divided in seven different categories: Human, Animal, Plant, Landscape, Cityscape, Still-life and Transportation. The amount of available images helps greatly to generalize and allows us to keep a good part of it for testing. The dataset is randomly divided in two parts, 70% for training and 30% for testing. Half of the images are being added with AWGN with σ=[0,55]. The second half are processed with equation 3, the realistic noise model, with Analog Gain Ag=[0,64] and Digital Gain Dg=[0,32].
The network will be trained with $50x50x3$ patches follow-

ing [14]. We obtained 120000 patches from the Waterloo training set, containing AWGN and real noise type, using data augmentation such as rotating, flipping and mirroring. For low-light conditions, we used five noisy images for each light level from our own training dataset, obtaining 60 pairs of noisy-clean images for training. The patches extracted are 80000. From the Renoir dataset, we used the subset T3 and randomly cropped 40000 patches. For low-light testing, we will use 5 images from our camera of a different scene, not present in the training set, and part of the Renoir T3 set. We trained our network with 100 epochs, using a batch of 128 and Adam Optimizer [37] with a learning rate of 10^{-3} for the first 20 epochs and 10^{-4} for the latest 80.

4.3. Image Denoising Benchmarks

We performed two types of benchmarking tests on our trained Spatial-CNN. Firstly, we compared Blind Gaussian Denoising with the original implementation of DnCNN, on which ours is based. From our test in Table 1 on the BSD68 test set, we notice how the result of our blind model and the one proposed by the paper [14] are comparable.

	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$	$\sigma = 50$
Spatial-CNN*	39.73	35.92	33.66	30.99	29.34	27.63
DnCNN-B* [14]	39.79	35.87	33.57	30.69	28.74	26.53
DnCNN-B [14]	40.62	36.14	33.88	31.22	29.57	27.91

Table 1: Comparison of Blind Gaussian Denoising on the CBSD68 dataset. Our modified version of DnCNN for spatial denoising has comparable results with the original one. The values represent *PSNR[dB]*, the higher the better. DnCNN results obtained with the provided Matlab implementation [38]. CBSD68 available here [39]. *Noisy images clipped in range [0,255].

Secondly, to understand the effectiveness of our training set on real-world images, we denoised the sRGB DND dataset [31] and submitted for evaluation. The result [40] is encouraging, since our trained model (called *128-DnCNN Tensorflow* in the DND webpage) scored an average of $37.0343dB$ for the PSNR and 0.9324 for the SSIM, placing it in the first 10 positions. Interestingly, the authors of DnCNN submitted their result of a fine-tuned model, called *DnCNN+*, a week later, achieving the overall highest score for SSIM, which further confirms its flexibility.

4.4. Temporal CNN Training

Here we are describing the training process of the Temp3-CNN, which is the temporal denoising network considering three subsequent frames.
If previously we needed pairs of clean and noisy images, now we need pairs of videos. For artificially added noise as Additive White Gaussian Noise (AWGN) or the real noise model in equation 3, is easy to create such couples. However, for real-world and low-light conditions videos is almost impossible. For this reason, this kind of video dataset, offering pairs of noisy and noise-free sequences, are not available. Therefore, we decided to proceed according to this sequence:

1. We selected 31 publicly available videos from [41].
2. We divided the videos in sequences of three frames each.
3. Every sequence was added with either Gaussian noise with σ=[0,55] or real noise 3 with Ag=[0,64] and Dg=[0,32].
4. All the sequences went through the Spatial-CNN and the results were stored.
5. Pairs of spatially-denoised and clean sequences will be used for training.

We followed the same training procedure as the Spatial-CNN, even though now the network will be trained with patches of dimension $50 \times 50 \times 9$, containing three patches coming from three subsequent frames.
The 31 selected videos contain 8922 frames, which means 2974 sequences of three frames and a final number of patches of 300032. We ran the training for 60 epochs with a batch size of 128, Adam optimizer with learning rate of 10^{-4} and *LeakyReLU* as activation function. It is shown LeakyReLU can outperform ReLU [35]. However, we did not use Leaky Relu in the spatial CNN, because ReLU performed better. We present the comparison result in the supplementary material. In the final version of Temp3-CNN, Batch Normalization (BN) was not used: experiments show it slows down the training and denoising process. BN did not improve the final result in terms of PSNR. Moreover, denoising without BN requires around 5% less time. On one hand, BN helps for spatial denoising, because inputs have a wide range of noise levels in the same batch. On the other hand, BN is not helpful with this setting for temporal denoising, because the inputs have less variance, having been already spatially denoised from the same Spatial-CNN model. Figure 5 represents the evolution of the L2-loss for the Temp3-CNN: avoiding the normalization step makes

	PSNR [dB]	SSIM
Spatial-CNN	37.0343	0.9324
CBDNet [22]	38.0564	0.9421
DnCNN+ [14]	37.9018	0.943
FFDNet+ [16]	37.6107	0.9415
BM3D [34]	34.51	0.8507

Table 2: Results of the DND benchmark [31] on real-world noisy images. It shows that our dataset, containing different noise models, is valid for real-world image denoising, placing our Spatial-CNN in the top 10 for sRGB denoising.

the loss starting immediately at a low value. We trained the same network with Leaky ReLU, Leaky ReLU+BN and ReLU+BN and present the comparison results in the supplementary material.

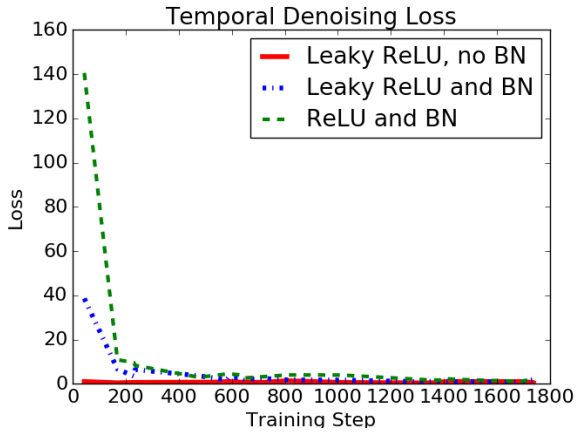


Figure 5: Evolution of the L2-Loss during the training of the Temp3-CNN. Batch Normalization (BN) does not help, adding a computation overhead without any improvement over the final result. With Leaky ReLU as activation function and with no BN, the loss starts immediately around 1 and decreases to 0.5 after 60 epochs. Denoising without BN takes around 5% less time. We show only the first 1800 steps for a better visualization.

4.5. Exp 1: Evaluating the Optimal Video Denoising CNN Architecture

The final proposed version of VidCNN consists in two CNNs in a pipeline, performing first spatial and then temporal denoising. To get the final architecture, we trained VidCNN with different structures and tested it on two famous benchmarking videos and on one we personally recorded with a Blackmagic Design URSA Mini 4.6K, capable to record raw videos. The videos have various levels of Additive White Gaussian Noise (AWGN). We will answer to three critical questions.

4.5.1 Q1: Is Temp3-CNN able to learn both temporal and spatial denoising?

- **Spatial-CNN**: image denoising CNN, similar to DnCNN[14] but with 128 feature maps in the first convolutional layer.
- **Temp3-CNN**: temporal denoising CNN, same architecture of the spatial one, but with three noisy frames as input.

We compare the Spatial-CNN with the Temp3-CNN, which in this case tries to perform spatial and temporal denoising at the same time.
Answer: referring to Table 3, we notice how using Temp3-CNN alone leads to a worse result compared to the simpler Spatial-CNN.

	Foreman		*Tennis*		*Strijp-S* *
Res./Frames	288×352 / 300		240×352 / 150		656×1164/787
σ	25	55	25	55	25
Spatial-CNN	32.18	28.27	29.46	26.15	32.73
Temp3-CNN	31.56	27.45	29.32	25.63	31.13

Table 3: Comparison of Spatial-CNN and Temp3-CNN over videos with different levels of AWGN. The Temp3-CNN alone can not outperform the Spatial-CNN. Results expressed in terms of *PSNR[dB]*. *Self-recorded Raw video converted to RGB.

4.5.2 Q2: In which order performing spatial and temporal denoising?

Knowing that using Temp3-CNN alone is not enough, we now have to compare different combination of spatial and temporal denoising.
Answer: looking at Table 4, we can confirm that using temporal denoising improves the result over spatial denoising, with the best performing combination as Spatial-CNN followed by Temp3-CNN.

	Foreman		*Tennis*		*Strijp-S*
Res./Frames	288×352 / 300		240×352 / 150		656×1164/787
σ	25	55	25	55	25
Spatial-CNN	32.18	28.27	29.46	26.15	32.73
Temp3-CNN & Spatial-CNN	32.09	28.37	29.21	25.98	32.28
Spatial-CNN & Temp3-CNN	33.12	29.56	30.36	27.18	34.07

Table 4: The combination of Spatial-CNN + Temp3-CNN is the best performing, showing consistent improvements of $\sim 1dB$ over the spatial-only denoising. Results expressed in terms of *PSNR[dB]*.

4.5.3 Q3: How many frames do we have to consider?

We have to investigate how many frames are needed for the best trade-off between quality and complexity, which increases with every additional frame we consider. We compare now the introduced Temp3-CNN with Temp5-CNN, which considers a time window of five frames.
Answer: results in Table 5 shows that considering more frames could improve the result, but this is not guaranteed. Therefore, since using a bigger time window means

more memory and time needed, we decided to use the three frames model for a better trade-off. For comparison, using the Temp5-CNN on the video *Foreman* took 6.5% more time than using the Temp3-CNN, $21.17s$ vs $19.85s$ on GPU. The difference does not seem much, but if we would apply this denoiser to real videos, which are bigger in size and longer in time, the difference would increase and the Temp5-CNN could not fit in the memory.

	Foreman		*Tennis*		*Strijp-S*
Res./Frames	288×352 / 300		240×352 / 150		656×1164/787
σ	25	55	25	55	25
Spatial-CNN	32.18	28.27	29.46	26.15	32.73
Spatial-CNN & Temp3-CNN	33.12	29.56	30.36	27.18	34.07
Spatial-CNN & Temp5-CNN	33.03	29.87	30.72	27.70	33.97

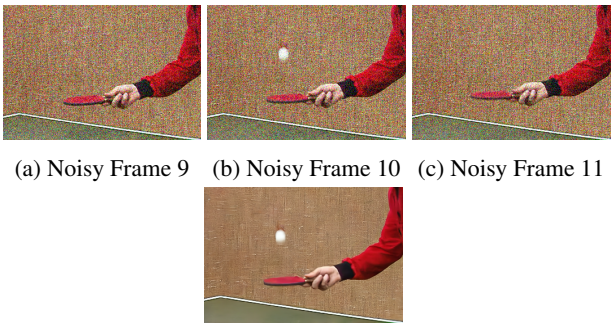
Table 5: Comparison of architectures using 3 or 5 frames. Using a bigger time window, i.e. five frames, may slightly improve the final result or even worsen it. Hence, we decided to proceed using a 3-frames architecture. Results expressed in terms of *PSNR[dB]*.

4.6. Exp 2: Evaluating Sensitivity to Temporal Inconsistency

To correctly combine the information coming from the two other frames, VidCNN has to understand if the secondary frames contain useful information to improve the reference frame or not. We can appreciate how powerful is our model with this simple experiment:

- We consider the video *Tennis* from [41], adding Gaussian noise with standard deviation σ=40.
- We artificially removed the white ball, covering it with part of the background.
- We ran VidCNN and check the difference between the experiment and the output with non-modified noisy frames 9 and 11.

Figure 6 shows the modified input frames and VidCNN output. In terms of PSNR value, we got the same value for both normal and experimental case: $27.28dB$. This result confirms that the network does what we expected: it uses part of the secondary frames and combine them with the reference, but only where the pixel content is similar enough. Thus, in this scenario the outcome does not change, since VidCNN would not have used the information in the ball area anyway.



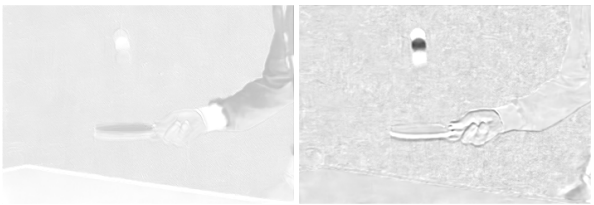
(a) Noisy Frame 9 (b) Noisy Frame 10 (c) Noisy Frame 11

(d) VidCNN Frame 10

Figure 6: VidCNN achieves the same PSNR value of $27.28dB$ for frame 10 of the video *Tennis* with AWGN σ=40, even if we manually cancel the white ball from the secondary frames. The network understands which part has to take into consideration and which not, i.e. the ball area.

4.6.1 Visualization of temporal filters

The novel architecture Temp3-CNN is capable to detect the temporal inconsistencies, but without a visual reference it is difficult to understand the process. Hence, we want to show in detail what our model detects and how we can interpret this visualization. In Figure 7, we show the output of two of the 128 filters in the first layer of Temp3-CNN, where the network highlights different features of the concatenated input frames. In Figure 7b, we see in black the table-tennis ball of the current frame, whereas the ones in the previous and subsequent frame are in white. In Figure 7a instead, we see how this filter highlights flat areas with similar colors and shows mostly the ball of the current frame in white. Therefore, Temp3-CNN gives different importance to similar and different areas among the three frames. This is a simple indication on how the CNN handles motion and temporal inconsistencies.

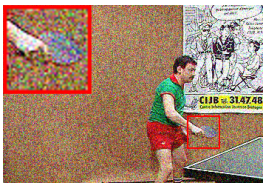


(a) Filter 59 (b) Filter 90

Figure 7: Visualization of filters 59 and 90 output of Temp3-CNN first convolutional layer. This layer is composed by 128 filters. We used frames number 9, 10 and 11 from the video *Tennis* as input. Filter 59 highlights the reference ball and other areas with similar colors, whereas filter 90 seems to highlight mostly contours and the ball at the reference position in frame 10. The images have been color-inverted for a better visualization.

(a) Noisy frame 148

(b) Noisy frame 150



(c) Noisy frame 149
(16.48/0.6023)

(d) CBM3D[6]
(25.60/0.9482)

(e) VBM4D[10]
(24.51/0.9292)

(f) VidCNN-G
(27.19/0.9605)

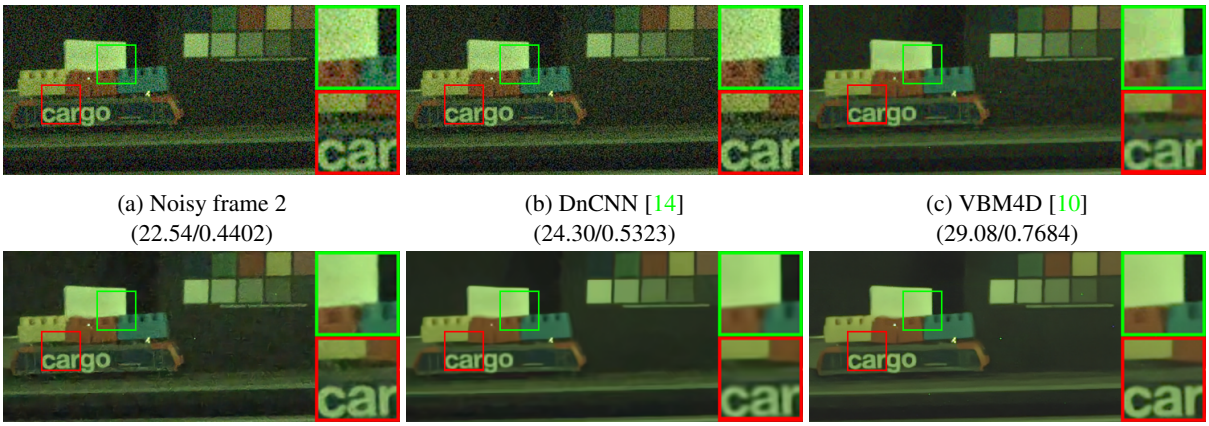
Figure 8: Blind video denoising comparison on *Tennis* [41] corrupted with AWGN σ=40 and values clipped between [0,255]. We show the result of two competitors, VBM4D and CBM3D, which scored respectively second and third (see Table 6) on this test video. VidCNN performs well in challenging situations, even if the previous frame is completely different 8a, thanks to the temporal inconsistency detection. VBM4D suffers from the change of view, creating artifacts. Results in brackets are referred to the single frame 149 (PSNR [dB]/SSIM).

	Tennis			*Old Town Cross*			*Park Run*			*Stefan*		
Res./Frames	240×352 / 150			720×1280 / 500			720×1280 / 504			656×1164 / 300		
σ	5	25	40	15	25	40	15	25	40	15	25	55
VidCNN	35.51	29.97	28.00	32.15	30.91	29.41	31.04	28.44	25.97	32.06	29.23	24.63
VidCNN-G	**37.81**	**30.36**	**28.44**	32.39	**31.29**	**29.97**	**31.25**	**28.72**	**26.36**	**32.37**	**29.59**	**25.06**
VBM4D [10]	34.64	29.72	27.49	**32.40**	31.21	29.57	29.99	27.90	25.84	29.90	27.87	23.83
CBM3D [34]	27.04	26.37	25.62	28.19	27.95	27.35	24.75	24.46	23.71	26.19	25.89	24.18
DnCNN [14]	35.49	27.47	25.43	31.47	30.10	28.35	30.66	27.87	25.20	32.20	29.29	24.51

Table 6: Comparison of VidCNN with a video denoising algorithm, VBM4D [10], and two image denoising algorithms, DnCNN [14] and CBM3D [34]. VidCNN-G is the model trained specificly for blind gaussian denoising. Test videos have different length, size and level of Additive White Gaussian Noise. VidCNN performs better than blind denoising algorithms CBM3D, DnCNN and VBM4D, which has been used with the *low complexity* setup due to our memory limitations. Best results are highlighted in bold. Original videos are publicly available here [41]. Results expressed in terms of *PSNR[dB]*.

4.7. Exp 3: Evaluating Performance on Gaussian Video Denoising

Currently, most of the video and image denoising algorithms have been developed to tackle Additive White Gaussian Noise (AWGN). We will compare VidCNN with the state-of-art algorithm for gaussian video denoising VBM4D [10] and additionally with CBM3D [34] and DnCNN [14] for single frame denoising. We used the algorithms in their blind version: for VBM4D we activated the noise estimation, for CBM3D we set the sigma level to 50 and for DnCNN we use the blind model provided by the authors. We compare two versions of VidCNN, where VidCNN-G is the model trained specifically for AWGN denoising and VidCNN is the final model tackling multiple noise models, including low-light conditions. The videos have been stored as uncompressed png frames, added with AWGN and saved again in loss-less png format. From the results in Table 6 we notice that VBM4D achieves superior results compared to its spatial counterpart CBM3D, which is probably due to the effectiveness of the noise estimator implemented in VBM4D. CBM3D suffers from the wrong noise std. deviation (σ) level for low noise intensities, whereas for high levels achieves comparable results. Overall, our implemented VidCNN in its gaussian specific version performs better than the general blind model, even though the difference is limited. VidCNN-G scores the best results, as highlighted in bold in Table 6, confirming our blind video denoising network as a valid approach, which achieves state-of-art results.



(a) Noisy frame 2 (22.54/0.4402)
(b) DnCNN [14] (24.30/0.5323)
(c) VBM4D [10] (29.08/0.7684)
(d) CBDNet [22] (30.75/0.8710)
(e) CBM3D[34] (31.11/0.8982)
(f) VidCNN (34.14/0.9158)

Figure 9: Detailed comparison of denoising algorithms on the low-light video *Train* with light intensity at 50/255. Our VidCNN shows superior performance in this light condition, preserving edges and correctly smoothing flat areas. Results referred to frame 2, expressed in terms of (PSNR [dB]/SSIM).

	Train		Mountains	Windmill		
Res./Frames	212×1091 / 4		1080×1920 / 4	1080×1920 / 3		
Light	50/255	55/255	[55,75]/255	44.6 lux	118 lux	212 lux
VidCNN	**34.05**	**36.96**	**40.84**	**32.96**	**35.42**	**36.69**
VBM4D[10]	29.10	33.48	37.34	26.62	30.69	32.92
CBDNet[22]	30.89	34.56	39.91	29.56	34.31	36.22
CBM3D[34]	31.27	34.06	40.20	29.81	34.06	35.74
DnCNN[14]	24.33	29.87	32.39	21.73	25.55	27.87

Table 7: Comparison of state-of-art denoising algorithms over six low-light sequences recorded with a Bosch Autodome IP 5000 IR in raw mode, without any type of filtering activated. Every sequence is composed of 4 or 3 frames, with ground truths obtained averaging over 200 images. The *Windmill* sequences has been recorded with a different light source, where we were able to measure the light intensity. Highlighted in bold our VidCNN results, which performs significantly better. Results expressed in terms of *PSNR[dB]*.

4.8. Exp 4: Evaluating Performance on Low-Light Video Denoising

Along with the low-light dataset creation, we also recorded six sequences of three or four frames each:

- Two sequences of the same scene, with a moving toy train, in two different light intensities.
- A sequence of an artificial mountain landscape with increasing light intensity.
- Three sequences of the same scene, with a rotating toy windmill and a moving toy truck, in three different light conditions.

Those sequences are not part of the training set and have been recorded separately, with the same Bosch Autodome IP 5000 IR camera. In Table 7 we present the results of VidCNN, highlighted in bold, in comparison with other state-of-art denoising algorithms on the low-light test set. We compare our method with VBM4D [10], CBM3D [34], DnCNN [14] and CBDNet [22]. VidCNN achieves substantial improvements over the competitors, especially for the lowest light intensities. Surprisingly, the single frame denoiser CBM3D performs better than the video version VBM4D: the difference may arose, because CBM3D in its blind version uses $\sigma = 50$, whereas VBM4D has a built-in noise level estimator, which may perform worse with a completely different noise model from the supposed Gaussian one. CBDNet achieves also remarkable results as expected, since it is suitable for real-world noisy images. In this case we can appreciate how powerful CNNs are, in conditions where designing a specific denoising algorithm for a specific sensor would be difficult and time consuming.

5. Discussion

5.1. Summary

In this paper, we presented a novel CNN architecture for Blind Video Denoising called VidCNN. We use spatial and temporal information in a feed-forward process, combining three consecutive frames to get a clean version of the middle frame. We perform temporal denoising in simple yet efficient manner, where our Temp3-CNN learns how to handle objects motion, brightness changes, camera motion and temporal inconsistencies. We define our model as *Blind*, since it can tackle different noise models at the same time, without any prior knowledge nor analysis of the input signal. We created a dataset containing multiple noise models, showing how the right mixture of training data can improve image denoising on real world data, such as on the DND Benchmarking Dataset [31]. We achieve state-of-art results in blind gaussian video denoising, comparing our outcome with the competitors available in the literature. We show how it is possible, with the proper hardware, to address low-light video denoising with the use of a Convolutional Neural Network, which would ease the tuning of new sensors and camera models. Collecting the proper training data would be the most time consuming part. However, defining an automatic framework with predefined scenes and light conditions would simplify the process, allowing to further reduce the needed time and resources. Our technique for acquiring clean and noisy low-light image pairs has proven to be effective and simple, requiring no specific exposure tuning.

5.2. Limitations and Future Works

One of the biggest limitations of VidCNN, if we would use it in real-world scenarios, is the needed computational power. At the current state, even with an high-end graphic card as the Nvidia Titan X, we were able to reach a maximum speed of $\sim 3fps$ on HD videos. However, most of the current cameras work with Full HD or even UHD (4K) resolutions, with always higher frame rates. We did not try to implement VidCNN on a mobile device supporting Tensorflow Lite, which converts model and weights to a lighter version more suitable for handled devices. Hence, this could be a possible new development and challenging question to investigate on, since every week the available hardware in the market gets more powerful. Something that we still have to analyze in depth are the generalization capabilities of the proposed method: we were able to collect real data from a single sensor type and we should investigate further its performance, using data from multiple sensors at the same time.

References

[1] Ivana Despotovi, Bart Goossens, and Wilfried Philips. Mri segmentation of the human brain: Challenges, methods, and applications. In *Computational and Mathematical Methods in Medicine*, volume 2015, pages 1–23. 05 2015. 2

[2] Peter Goebel, Ahmed Nabil Belbachir, and Michael Truppe. A study on the influence of image dynamics and noise on the jpeg 2000 compression performance for medical images. In *Computer Vision Approaches to Medical Image Analysis: Second International ECCV Workshop*, volume 4241, pages 214–224. 05 2006. 2

[3] Stefan Roth and Michael Black. Fields of experts: A framework for learning image priors. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 860–867. 01 2005. 2

[4] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Image super-resolution using gradient profile prior. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 06 2008. 2

[5] Huibin Li and Feng Liu. Image denoising via sparse and redundant representations over learned dictionaries in wavelet domain. In *2009 Fifth International Conference on Image and Graphics*, pages 754–758. 09 2009. 2

[6] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. In *IEEE transactions on image processing*, volume 16, pages 2080–95. 09 2007. 2, 3, 10

[7] Julien Mairal, Francis Bach, J Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2272–2279. 09 2009. 3

[8] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin li. Nonlocally centralized sparse representation for image restoration. In *IEEE transactions on image processing*, volume 22. 12 2012. 3

[9] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2862–2869. 06 2014. 3

[10] Matteo Maggioni, Giacomo Boracchi, Alessandro Foi, and Karen Egiazarian. Video denoising, deblocking, and enhancement through separable 4-d nonlocal spatiotemporal transforms. In *IEEE transactions on image processing*, volume 21, pages 3952–66. 05 2012. 3, 10, 11

[11] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 769–776. Curran Associates, Inc., 2009. 3

[12] H.C. Burger, C.J. Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2392–2399. 06 2012. 3

[13] Bryan C. Russell, Antonio Torralba, Kevin Murphy, and William T. Freeman. Labelme: A database and web-based tool for image annotation. In *International Journal of Computer Vision*, volume 77. 05 2008. 3

[14] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. In *IEEE Transactions on Image Processing*, volume 26, pages 3142–3155. 07 2017. 3, 5, 7, 8, 10, 11

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of Machine Learning Research*, volume 37, pages 448–456. 02 2015. 3, 5

[16] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn based image denoising. In *IEEE Transactions on Image Processing*, volume 27, pages 4608 – 4622. 09 2018. 3, 7

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. hhttps://arxiv.org/abs/1711.10925, 11 2017. Accessed: 20-08-2018. 3

[18] Stamatis Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5882 – 5891. 07 2017. 3

[19] Dong Yang and Jian Sun. Bm3d-net: A convolutional neural network for transform-domain collaborative filtering. In *IEEE Journals & Magazines*, volume 25, pages 55 – 59. 01 2018. 3

[20] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. pages 4539–4547, 10 2017. 3

[21] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image denoising using very deep fully convolutional encoder-decoder networks with symmetric skip connections. In *Advances in Neural Information Processing Systems 29*, pages 2802–2810. 12 2016. 3

[22] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. https://arxiv.org/abs/1807.04686, 07 2018. Accessed: 20-08-2018. 3, 5, 7, 11

[23] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. 2018. 3

[24] Xiaokang Yang Xinyuan Chen, Li Song. Deep rnns for video denoising. In *Proc. SPIE*, volume 9971. 09 2016. 4

[25] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018. 4

[26] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018. 4

[27] Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2848–2857, 07 2017. 4, 6

[28] Ren Yang, Mai Xu, Zulin Wang, and Tianyi Li. Multi-frame quality enhancement for compressed video. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 4, 6

[29] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang. Deep video deblurring for hand-held cameras. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 237–246, 07 2017. 4, 6

[30] Josue Anaya and Adrian Barbu. Renoir a dataset for real low-light image noise reduction. In *Journal of Visual Communication and Image Representation*, volume 51, 09 2014. 4, 6

[31] Tobias Plotz and Stefan Roth. Benchmarking denoising algorithms with real photographs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 07 2017. 4, 7, 12

[32] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018. 4

[33] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018. 4

[34] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space. In *2007 IEEE International Conference on Image Processing*, volume 1, pages I – 313–I – 316, 09 2007. 5, 7, 10, 11

[35] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. 2015. 5, 7

[36] Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang. Waterloo Exploration Database: New challenges for image quality assessment models. volume 26, pages 1004–1016, 02 2017. 6

[37] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 12 2014. 7

[38] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Dncnn matlab implementation. https://github.com/cszn/DnCNN, 2016. Accessed: 20-08-2018. 7

[39] CBSD68 benchmark dataset. https://github.com/clausmichele/CBSD68-dataset. Accessed: 20-08-2018. 7

[40] DND Benchmark Results. https://noise.visinf.tu-darmstadt.de/benchmark/. Accessed: 20-08-2018. 7

[41] Xiph.org Video Test Media [derf's collection]. https://media.xiph.org/video/derf/. Accessed: 20-08-2018. 7, 9, 10

Supplementary Material of VidCNN - Learning Blind Video Denoising

Michele Claus
TU Delft - EEMCS
Mekelweg 4, Delft, The Netherlands
claus.michele@hotmail.it

1 Background on Convolutional Neural Networks

In this section we want to give a high-level introduction to the concepts used in our Video Denoising CNN. For this reason, we will only go through the types of components we used. For a better and more in-depth explanation we suggest the book [1]. Convolutional Neural Networks (CNNs) are a type of feed-forward artificial Neural Networks, which are mostly used to analyze and elaborate multi-dimensional signals such as images. In fact, an image can be composed of a single channel for gray level, three or four for color images and more for multi-spectral data.
CNNs have been successfully applied to image recognition, object detection and also image denoising tasks. In this section we want to present the components of the CNN architecture we used for the implementation of VidCNN, the blind video denoising network.

1.1 Multi Dimensional Discrete Convolution

The basic and fundamental operation of CNNs is the Convolution. Since we are in the digital domain, our data has been quantized and we will work with discrete values, which means we use a particular type of convolution. The general form of the multi dimensional discrete valued convolution can be written as

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^{C} K_{m,n,c} \cdot I_{i+m,j+n,c} + b\,, \tag{1}$$

where $I \in \mathbb{R}^{H \times W \times C}$ represents our image signal with C number of channels, $K \in \mathbb{R}^{k_1 \times k_2 \times C \times D}$ represents the *kernel* with D filters and biases b, one for each filter. The operation is composed only by multiplication and addition, with complexity depending on the size of the image and the kernel. With a gray-scale image of size $(H \times W)$ and a kernel $(k \times k)$, the complexity will result as $O(HWkk)$. If the kernel is *separable*, which means it can be written as the convolution of two one-dimensional vectors $K = K_1 * K_2$, the complexity decreases to $O(2HWk)$. In the Neural Network field, the kernel is also called *receptive field*.

1.2 Training Process

1.2.1 Gradient Descent

The Gradient Descent is a generic optimization algorithm capable of finding optimal solutions in various problems. The main idea of Gradient Descent is to tune parameters iteratively to minimize a pre-defined loss function. The loss function is the L2-norm in equation 2, also known as least squares error (LSE). It is the sum of the square of the differences S between the target value Y and the estimated values Y_{est}.

$$L = \sum_x \sum_y (Y(x,y) - Y_{est}(x,y))^2 \tag{2}$$

The Gradient Descent measures the local gradient of the error function with regards to the filter weights $\mathbf{w}$, which have been randomly initialized, and step by step follows the highest gradient

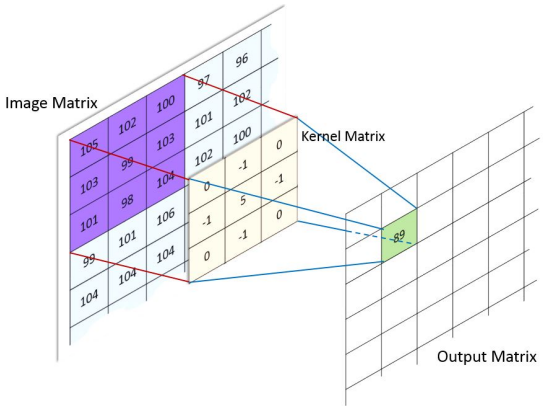


Figure 1: Graphical representation of a 2D discrete valued convolution with a 3×3 kernel. Source [2].

(which we can see as a measure of "steepness") attempting to decrease the loss function, until the algorithm converges to a minimum. A necessary parameter of the Gradient Descent algorithm is the *learning rate*, which determines the size of each update step: if too small it will take many iterations to converge, whereas if too high could lead to divergence instead of convergence. There can be several *hills* and *valleys* the optimizer has to overcome to get to the absolute minimum. A gradient descent variant that attempts to mimic the physical behavior of a ball rolling down the loss-function surface with momentum is the *Adam* optimizer, which tries to avoid getting stuck in local optima *valley*. We use Adam optimizer during our CNN training phase, with predefined first order momentum of 0.9.

1.2.2 Tensorflow Autodiff

Tensorflow has built-in functions which can automatically and efficiently compute the gradients: we do not have to use any loop for multiple derivatives. There are multiple approaches to compute gradients automatically and Tensorflow uses the specific *Reverse-mode autodiff*. You can refer to [1] for more details.

1.2.3 The Backpropagation Process

In 1986, D.E. Rumelhart *et al.* published an article about an innovative training algorithm called *backpropagation* [3]. We can describe it as a Gradient Descent using reverse-mode autodiff. For each training instance, the algorithm feeds the data to the CNN and computes the output of every consecutive convolutional layer. This step is called *forward pass* and it is the same we use for estimation (denoising in our specific case). Afterwards, computing the Loss, the algorithm measures the difference between the reference sample (in our case a patch of clean image) and the actual output of the CNN (the denoised image patch). Subsequently, it measures how much of the error comes from each *neuron* in the last convolutional layer. It proceeds computing the contribution of the previous layers, until the process reaches the input layer. This reverse pass measures efficiently the loss gradient across all the layers by propagating it backward in the network. The name of the algorithm comes from this final step.

1.2.4 Vanishing and Exploding Gradients Problems

The backpropagation algorithm propagates the error gradient from the output layer up to the input layer. The gradients are used for updating the weights in each neuron, but unfortunately gradients value gets often smaller and smaller as we reach the lower layers. As outcome, the algorithm leaves the lower layer connection weights almost the same, avoiding the training to reach an optimal solution. This problem is called *Vanishing Gradient*. The opposite, the *Exploding Gradient*, can also happen, making the gradients increase too steeply and the algorithm to diverge. There are different solutions to overcome this problem, which involve the use of particular activation functions or normalization steps, such as Batch Normalization.

1.2.5 Batch Normalization

Thanks to particular activation functions as ReLU and LeakyReLU, the vanishing/exploding gradient problem is attenuated at the beginning of the training phase. However, it is not guaranteed it will not reappear during training. In 2015, Sergey Ioffe and Christian Szegedy published a paper [4], where they proposed a technique called *Batch Normalization* (BN) to address this issue, which they refer as *Internal Covariate Shift*. The technique consist of adding a normalization step before the activation function of each layer. This operation zero-centers and normalizes the input and then scales and shifts the results using a parameter for each operation. Hence, it learns the optimal scale and mean of the input for each layer.

1.3 Leaning the parameters

The result of a convolution depends on the kernel type: the size and values will determine the final result. There are many different types of kernels, which can extract contours, increase sharpness or contrast, blur and perform morphological operations. However, in the case of Convolutional Neural Networks, the type of kernel is not predefined, but it will be *learned* during the training phase, hence it is called receptive field. The process used for training the kernels weights is based on the gradient descent calculation and is called *backpropagation*, because it will start computing the gradient at the last layer and propagate it back to the higher ones. Before explaining the backpropagation process we have to introduce some other parameters and definitions.

1.3.1 Padding and Stride

Performing a 2D discrete value convolution consist of multiple multiplications and a final sum, for each output value (or pixel). The kernel will *slide* over the image, and the result will be placed in the central position, as shown in Figure 1. However, placing the kernel on a border pixel, will leave some of the kernel values without a multiplication factor from the image. Using for instance a (3×3) kernel and an image of $(H \times W)$, the output will shrink of 1 pixel on each side with a final size of $(H-1 \times W-1)$. Avoiding this effect and obtaining the same image size as output is possible by padding the input image before the convolutional step with zeros: this will increase the input size to $(H+1 \times W+1)$ and allow to obtain the expected size as output.

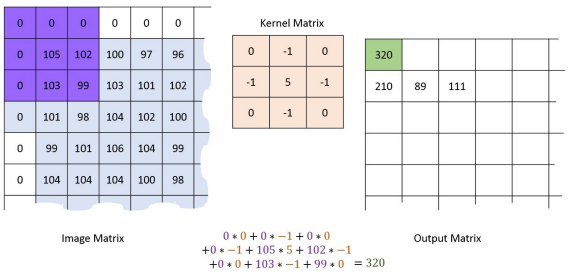


Figure 2: Graphical representation of a zero padded input image: all the kernel values have a multiplication factor on the border. Source [2].

Another fundamental parameter, which controls the output size, is the stride. The stride indicates how much the kernel will move at each step along each axis: using a stride of [1,1], for the 2D case, means that the kernel will move of just one pixel each step. However, if we increase the stride value, the kernel will move with bigger steps and the obtained pixels will decrease.

1.4 Activation Function

The activation function is typically used after every Convolutional Layer to introduce non-linearity. Our CNN has to learn from real-world data, which is mostly non-linear and therefore, we have to use a particular function to transform our output. In 2011, Glorot and Bengio published a paper [5] explaining how the vanishing/exploding gradient problem was partly due to the poor choice of the activation function. Until then, most researchers supposed that the sigmoid function was the best choice, since it was similar to the one in biological neurons. However, it turned out that other

activation functions behave much better in deep neural networks. The ReLU distinguished from others, because it does not saturate for positive values and it is fast to compute. In the VidCNN architecture two similar activation functions are used: ReLU and LeakyReLU.

1.4.1 ReLU

The Rectified Linear Unit function, introduced recently in the Machine Learning field [5], sets to zero all the negative values, according to the formula:

$$ReLU(x) = \max(o, x) \tag{3}$$

We used this function in the Spatial-CNN architecture, following [6], where they show its effectiveness for image denoising tasks, among others. An advantage of this activation function, compared to the *Sigmoid* (commonly used in machine learning), lays in their range: the ReLU has range $[0, +\infty]$, whereas the Sigmoid $[0, 1]$, which is more suitable when working with probabilities. Another important factor to take into account, is that the gradient of the ReLU function does not vanish as we increase x. On the other hand, the gradient of the sigmoid function vanishes as we increase or decrease x. When training deep neural networks with Backpropagation, this has major benefits.

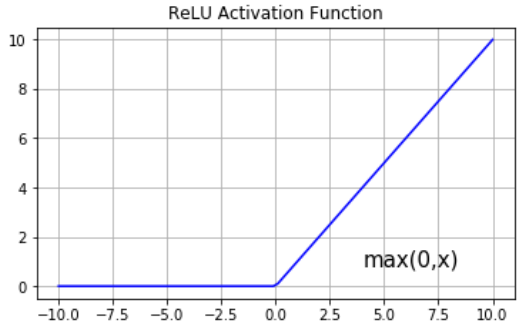


Figure 3: The ReLU activation function. All negative input values are set to zero, introducing non-linearity efficiently.

1.4.2 Leaky ReLU

The introduction of ReLU gave major benefit for training networks with deep architectures. However, ReLU introduces an issue called *dying*: when the dot product of the ReLU input with its weights is negative, the neuron *dies* and its output will be always zero. Subsequently, the gradient will be always zero. The loss, back-propagated from later layers gets multiplied by zero, hence no error signal arrives to earlier layers. Leaky ReLU has been introduced to overcome this problem: instead of setting to zero all the negative values, it maps them according to a parameter, usually set to 0.01. Leaky ReLU allows a small gradient even when the unit is inactive.

$$LeakyReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases} \tag{4}$$

We used Leaky ReLU in final version of the Temp3-CNN, the network performing temporal denoising, which gave better results than ReLU.

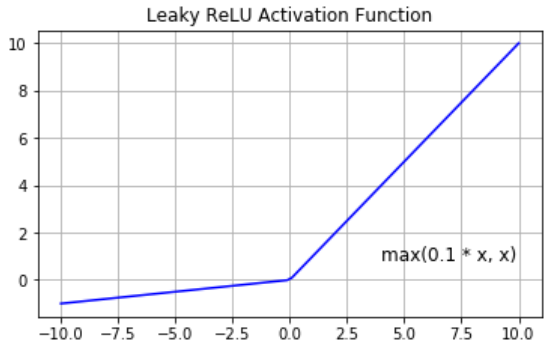


Figure 4: The Leaky ReLU activation function. It allows small negative values, letting the gradient being non-zero also in the left part of the plot.

2 Supplementary Results

2.1 Hardware details

All the developing, training and testing have been run on the same machine with the following specifications:

- O.S.: Ubuntu 16.04
- CPU: Intel Xeon E5-1620 v4
- RAM: 32GB
- GPU: Nvidia Titan X (Pascal)
- Deep Learning Library: Tensorflow 1.9 (Python 2.7)

2.2 Spatial-CNN: comparison results with ReLU and LeakyReLU

Here we present the comparison result of the Spatial-CNN, trained with two different activation functions. We tested both on the same benchmarking dataset, the CBSD68. Even though Leaky ReLU gave us better result for the Temp3-CNN architecture, in this scenario the simpler ReLU is still the best choice among them.

	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$	$\sigma = 50$
Spatial-CNN - ReLU	39.73	35.92	33.66	30.99	29.34	27.63
Spatial-CNN - Leaky ReLU	38.64	35.22	33.06	30.47	28.78	26.99

Table 1: Comparison of Spatial-CNN, trained with ReLU and Leaky RelU, on the CBSD68 dataset corrupted with Additive White Gaussian Noise of multiple levels. The same network, trained in the same way, perform better using the ReLU activation function, instead of the more recent Leaky ReLU.

2.3 Temp3-CNN: comparison results with BN and without

In Table 2, we present a comparison of the same VidCNN trained with three different settings:

1. Using Leaky ReLU as activation function and no normalization.
2. Using Leaky ReLU as activation function and Batch Normalization.
3. Using ReLU as activation function and Batch Normalization.

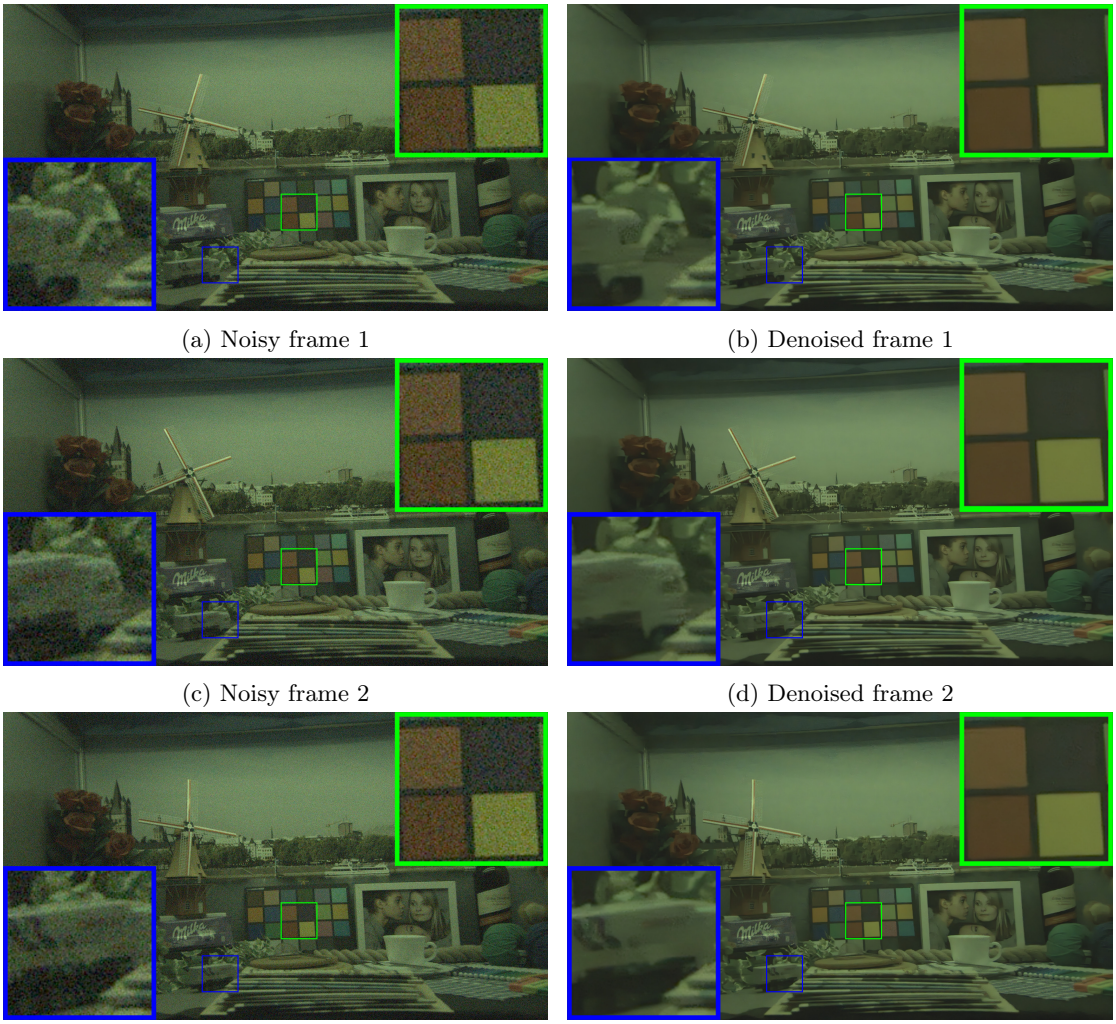
The results show how, using the more complex Leaky ReLU to train our temporal denoising network Temp3-CNN, we obtain the best results. In Table 2, we also show the difference in time needed to denoise the same videos, where the use of Batch Normalization increases it. We notice also that, due its lower computational complexity, the Temp3-CNN trained with ReLU and Batch Normalization is faster than the one trained with Leaky ReLU and BN.

	Train		Mountains	Windmill		
Res./Frames	212×1091/4		1080×1920/4	1080×1920/3		
Light	50/255	55/255	[55,75]/255	44.6 lux	118 lux	212 lux
VidCNN Leaky ReLU	34.05	36.96	40.84	32.96	35.42	36.69
Elapsed Time [s]	1.6361	1.6164	6.4080	5.8228	6.3080	6.1896
VidCNN Leaky ReLU + BN	34.07	36.98	40.76	32.87	35.34	36.81
Elapsed Time [s]	2.0089	2.3251	7.2746	6.6507	6.5803	6.6682
VidCNN ReLU + BN	33.88	36.57	40.42	32.77	35.27	36.40
Elapsed Time [s]	1.7450	1.7567	6.4485	5.4045	5.3711	5.4232

Table 2: Comparison of VidCNN trained with different settings on the same low-light test set. Two activation functions have been used, Leaky ReLU and ReLU. Batch Normalization has been used in two settings as normalization step. The best result in terms of Peak Signal to Noise Ratio (PSNR) and elapsed time is achieved by VidCNN trained with Leaky ReLU and no Batch Normalization.

3 Low-light Test Sequences

In this section we present the remaining low-light test sequences we recorded, not shown previously in the main article. The sequence *Windmill* contains many different objects and flat areas, plus the windmill and the toy truck that are moving. The sequence *Mountains* does not contain moving parts, but the illumination increases in time.

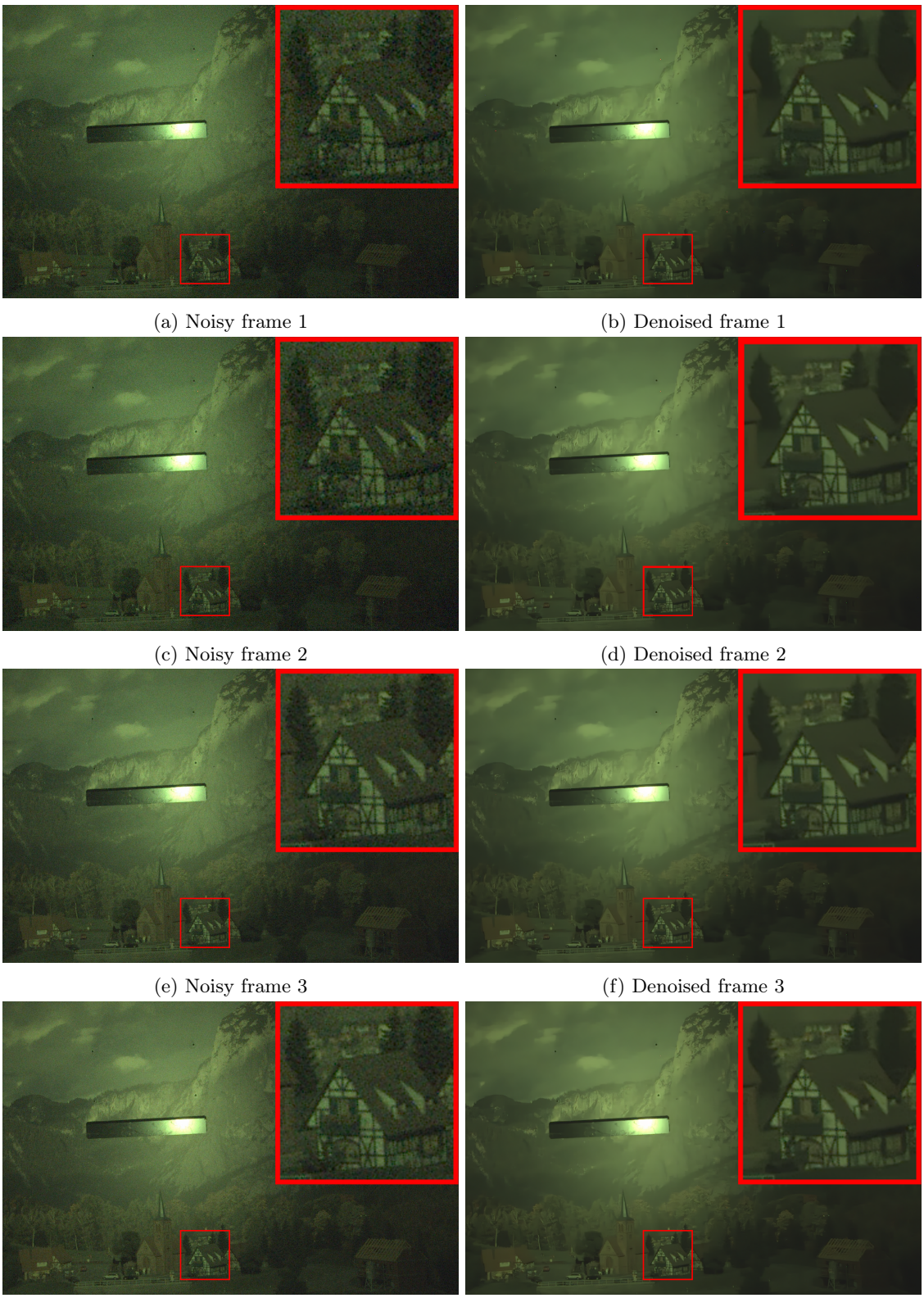


(a) Noisy frame 1 (b) Denoised frame 1

(c) Noisy frame 2 (d) Denoised frame 2

(e) Noisy frame 3 (f) Denoised frame 3

Figure 5: The test sequence *Windmill*, with light intensity of 212 lux. On the left we see the noisy sequence and on the right the VidCNN denoised output, which can correctly preserve edges and smooth flat areas.



(a) Noisy frame 1

(b) Denoised frame 1

(c) Noisy frame 2

(d) Denoised frame 2

(e) Noisy frame 3

(f) Denoised frame 3

(g) Noisy frame 4

(h) Denoised frame 4

Figure 6: The test sequence *Mountains*, with light intensity increasing at each frame. We can appreciate how VidCNN can preserve the frame features looking at the house in details.

References

[1] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[2] Figure 1 and 2 source. http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html. Accessed: 20-08-2018.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back propagating errors. In *Nature*, volume 323, pages 533–536, 10 1986.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of Machine Learning Research*, volume 37, pages 448–456. 02 2015.

[5] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[6] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. In *IEEE Transactions on Image Processing*, volume 26, pages 3142–3155. 07 2017.