Video Super-Resolution With Convolutional Neural Networks

Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K. Katsaggelos, *Fellow, IEEE*

***Abstract*—Convolutional neural networks (CNN) are a special type of deep neural networks (DNN). They have so far been successfully applied to image super-resolution (SR) as well as other image restoration tasks. In this paper, we consider the problem of video super-resolution. We propose a CNN that is trained on both the spatial and the temporal dimensions of videos to enhance their spatial resolution. Consecutive frames are motion compensated and used as input to a CNN that provides super-resolved video frames as output. We investigate different options of combining the video frames within one CNN architecture. While large image databases are available to train deep neural networks, it is more challenging to create a large video database of sufficient quality to train neural nets for video restoration. We show that by using images to pretrain our model, a relatively small video database is sufficient for the training of our model to achieve and even improve upon the current state-of-the-art. We compare our proposed approach to current video as well as image SR algorithms.**

***Index Terms*—Deep Learning, Deep Neural Networks, Convolutional Neural Networks, Video Super-Resolution.**

I. Introduction

IMAGE and video or multiframe super-resolution is the process of estimating a high resolution version of a low resolution image or video sequence. It has been studied for a long time, but has become more prevalent with the new generation of Ultra High Definition (UHD) TVs (3,840 $\times$ 2,048). Most video content is not available in UHD resolution. Therefore SR algorithms are needed to generate UHD content from Full HD (FHD) (1,920 $\times$ 1080) or lower resolutions.

SR algorithms can be divided into two categories, model-based and learning-based algorithms. Model-based approaches [1]–[5] model the Low Resolution (LR) image as a blurred, subsampled version of the High Resolution (HR) image with additive noise. The reconstruction of the HR image from the LR image is an ill-posed problem and therefore needs to be regularized. In a Bayesian framework, priors controlling the smoothness or the total variation of the image are introduced in order to obtain the reconstructed HR image. For example, Babacan *et al.* [1] utilize the Bayesian framework to reconstruct an HR image from multiple LR observations, subject to rotation and translation amongst them. Belekos *et al.* [2] and later Liu and Sun [3] also use the Bayesian framework to derive an algorithm that is able to deal with complex motion and real world video sequences. With all these algorithms, the motion field and the HR reconstructed image, along with additionally required model parameters are estimated simultaneously from the observed data. Ma *et al.* [5] presented an algorithm that extended the same idea to handle motion blur.

Learning-based algorithms learn representations from large training databases of HR and LR image pairs [6]–[11] or exploit self-similarities within an image [10]–[13]. Dictionary based approaches utilize the assumption that natural image patches can be sparsely represented as a linear combination of learned dictionary patches or atoms. Yang *et al.* [6] were among the first to use two coupled dictionaries to learn a nonlinear mapping between the LR and the HR images. Improvements and variations of [6] were represented in [7]–[10], [13]. Song *et al.* [14] propose a dictionary approach to video super-resolution where the dictionary is learned on the fly. However, the authors assumed that sparsely existing keyframes in HR are available. Learning-based methods generally learn representations of patches and therefore also reconstruct an image patch by patch. In order to avoid artifacts along the patch edges, overlapping patches are used which leads to a considerable computational overhead.

Inspired by the recent successes achieved with CNNs [15], [16], a new generation of image SR algorithms based on deep neural nets emerged [17]–[21], with very promising performances. The training of CNNs can be done efficiently by parallelization using GPU-accelerated computing. Neural networks are capable of processing and learning from large training databases such as ImageNet [22], while training a dictionary on a dataset this size can be challenging. Moreover, once a CNN is trained, super-resolving an image is a purely feed-forward process, which makes CNN based algorithms much faster than traditional approaches. In this paper, we introduce a CNN framework for video SR.

In the classification and retrieval domains, CNNs have been successfully trained on video data [23], [24]. Training for recovery purposes remains a challenging problem because the video quality requirements for the training database are high since the output of the CNN is the actual video rather than just a label. Suitable videos for the SR task are uncompressed, feature-rich and should be separated by shots/scenes. We show that by pretraining the CNN with images we can bypass the creation of a large video database. Our proposed algorithm requires only a small video database for training to achieve very promising performance.

Manuscript received August 13, 2015; revised February 03, 2016; accepted February 10, 2016. Date of publication March 30, 2016; date of current version May 03, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alessandro Foi.

The authors are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: a.kappeler@u.northwestern.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCI.2016.2532323

The proposed CNN uses multiple LR frames as input to reconstruct one HR output frame. There are several ways of extracting the temporal information inside the CNN architecture. We investigated different variations of combining the frames and demonstrate the advantages and disadvantages of these variations. Our main contributions can be summarized in the following aspects:

- We introduce a video SR framework based on a CNN.
- We propose three different architectures, by modifying each time a different layer of a reference SR CNN architecture.
- We propose a pretraining procedure whereby we train the reference SR architecture on images and utilize the resulting filter coefficients to initialize the training of the video SR architectures. This improves the performance of the video SR architecture both in terms of accuracy and speed.
- We introduce Filter Symmetry Enforcement, wich reduces the training time of VSRnet by almost 20% without sacrificing the quality of the reconstructed video.
- We apply an adaptive motion compensation scheme to handle fast moving objects and motion blur in videos

The Caffe [41] model as well as the training and testing protocols are available at http://ivpl.eecs.northwestern.edu/software.

The rest of the paper is organised as follows. We briefly introduce deep learning and review existing deep learning based image SR techniques in Section II. In Section III we explain our proposed framework. Sections IV contains our results and their evaluation and Section V concludes the paper.

II. Related Work

A. Super-Resolution

Most of the state-of-the-art image SR algorithms are learning-based algorithms that learn a nonlinear mapping between LR and HR patches using coupled dictionaries [6]–[9]. Overcomplete HR and LR dictionaries are jointly trained on HR and LR image patches. Each LR image patch can be represented as a sparse linear combination of atoms from the LR dictionary. The dictionaries are coupled via common coefficients a.k.a. representation weights. The dictionaries and the coefficients can be found with standard sparse coding techniques such as K-SVD [25]. An HR patch can then be recovered by finding the sparse coefficients for an observed LR patch and applying them to the HR dictionary. Timofte *et al.* [10] considered replacing the single large overcomplete dictionary with several smaller complete dictionaries to remove the computationally expensive sparse coding step. This led to a significant faster algorithm while maintaining the reconstruction accuracy. A variation of [10] was recently proposed by Schulter *et al.* [11]. A random forest model was trained instead of the coupled dictionaries for the LR to HR patch mapping. Glasner *et al.* [13] did not learn a dictionary from sample images. Instead they created a set of downscaled versions of the LR image with different scaling factors. Then patches from the LR image were matched to the downscaled version of itself and its HR 'parent' patch was used to construct the HR image. Learning-based algorithms, although popular for image SR, are not very well explored for video SR. In [14] a dictionary based algorithm is applied to video SR. The video is assumed to contain sparsely recurring HR keyframes. The dictionary is learned on the fly from these keyframes while recovering HR video frames.

Many of the early works in multiframe SR have focussed on reconstructing one HR image from a series of LR images using a Bayesian framework [1]–[4]. The LR images were obtained by blurring and subsampling the HR image and then applying different motions to each LR image, such as translation and rotation. These algorithms generally solve two problems: Registration estimation, where the motion between the LR images is estimated, and image recovery, where the HR image is estimated using the information recovered in the first step. Bayesian video SR methods [2], [3] followed the same concept but used a more sophisticated optical flow algorithm [3] or a hierarchical block matching method [2] to find the motion field, in order to be able to deal with real world videos with more complex motion schemes. Ma *et al.* [5] extended the previously mentioned work in order to handle videos with motion blur. They introduced a temporal relative sharpness prior, which excludes pixels that are severely blurred. Because the image recovery process is an ill-posted problem, image priors such as constraints on the total variation [26] are introduced and then a Bayesian framework is used to recover the HR image. An alternative method to the conventional motion estimation and image restoration scheme is presented in [27]. Instead of explicit motion estimation, a 3-D Iterative Steering Kernel Regression is proposed. The video is divided and processed in overlapping 3D cubes (time and space). The method then recovers the HR image by approximating the pixels in the cubes with a 3D Taylor series.

Most video SR algorithms depend on an accurate motion estimation between the LR frames. There is a plethora of techniques in the literature for estimating a dense motion field [28]–[30]. Optical flow techniques assume that the optical flow is preserved over time. This information is utilized to form the optical flow equation connecting spatial and temporal gradients. Assuming local constancy of the optical flow, an over-determined system of equations is solved for determining the translational motion components per pixel with sub-pixel accuracy.

B. Deep Learning-Based Image Reconstruction

DNNs have achieved state-of-the-art performance on a number of image classification and recognition benchmarks, including the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC-2012) [15], [16]. However, they are not very widely used yet for image reconstruction, much less for video reconstruction tasks. Research on image reconstruction using DNNs includes denoising [31]–[33], impainting [31], deblurring [34] and rain drop removal [35]. In [17]–[21], deep learning is applied to the image SR task. Dong *et al.* [17] pointed out that each step of the dictionary based SR algorithm can be reinterpreted as a layer of a deep neural network. Representing an image patch of size $f \times f$ with a dictionary with n atoms can be interpreted as applying n filters with kernel size $f \times f$ on

the input image, which in turn can be implemented as a convolutional layer in a CNN. Accordingly, they created a CNN that directly learns the nonlinear mapping from the LR to the HR image by using a purely convolutional neural network with two hidden layers and one output layer. Their approach is described in more detail in Section III-A.

Wang *et al.* [19] introduced a patch-based method where a convolutional autoencoder [36] is used to pretrain a SR model on 33×33 pixel, mean-subtracted, normalized LR/HR patch pairs. Then the training patches are clustered according to their similarity and one sub-model is fine-tuned on self-similar patch pairs for each cluster. As opposed to [18], which uses standard fully connected autoencoders, they used convolutional based autoencoders which exploit the 2-dimensional data structure of an image. The training data was augmented with translation, rotation, and different zoom factors in order to allow the model to learn more visually meaningful features. Although this measure does increase the size of the training dataset, these augmentations do not occur in real image superresolution tasks. Moreover, although a convolutional architecture is used, the images have to be processed patch by patch due to the sub-models, whereas this is not necessary for our proposed algorithm.

Cui *et al.* [18] proposed an algorithm that gradually increases the resolution of the LR image up to the desired resolution. It consists of a cascade of stacked collaborative local autoencoders (CLA). First, a non-local self-similarity search (NLSS) is performed in each layer of the cascade to reconstruct high frequency details and textures of the image. The resulting image is then processed by an autoencoder to remove structure distortions and errors introduced by the NLSS step. The algorithm works with 7×7 pixel overlapping patches, which leads to an overhead in computation. Besides, as opposed to [17] and our proposed algorithm, this method is not designed to be an end-to-end solution, since the CLA and NLSS of each layer of the cascade have to be optimized independently.

Cheng *et al.* [20] introduced a patch-based video SR algorithm using fully connected layers. The network has two layers, one hidden and one output layer and uses 5 consecutive LR frames to reconstruct one center HR frame. The video is processed patchwise, where the input to the network is a $5 \times 5 \times 5$ volume and the output a reconstructed 3×3 patch from the HR image. The 5×5 patches or the neighboring frames were found by applying block matching using the reference patch and the neighboring frames. As opposed to our proposed SR method, [18] and [20] do not use convolutional layers and therefore do not exploit the two-dimensional data structure of images.

Liao *et al.* [21] apply a similar approach which involves motion compensation on multiple frames and combining frames using a convolutional neural network. Their algorithm works in two stages. In the first stage, two motion compensation algorithms with 9 different parameter settings were utilized to calculate SR drafts in order to deal with motion compensation errors. In the second stage, all drafts are combined using a CNN. However, calculating several motion compensations per frame is computationally very expensive. Our proposed adaptive motion compensation only requires one compensation and is still able to deal with strong motion blur (see Figure 10).

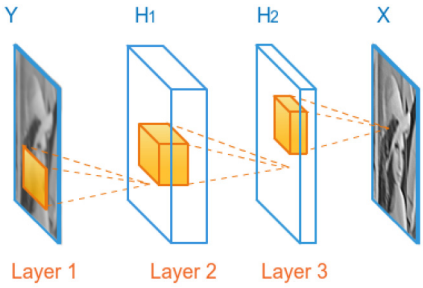

Fig. 1. Reference architecture for image super-resolution consisting of three convolutional layers.

III. Video Super-Resolution With Convolutional Neural Network

A. Single Frame/Image Super-Resolution

Before we start the training of the video SR model, we pretrain the model weights on images. For the image pretraining, we use a model for image SR, henceforth referred to as a reference model, with the network architecture parameters proposed in [17]. It has only convolutional layers which has the advantage that the input images can be of any size and the algorithm is not patch-based. The setup is shown in Figure 1. In it Y represents the input LR image and X the output HR image. It consists of three convolutional layers, where the two hidden layers H1 and H2 are followed by a Rectified Linear Unit (ReLU) [37]. The first convolutional layer consists of $1 \times f_1 \times f_1 \times C_1$ filter coefficients, where $f_1 \times f_1$ is the kernel size and C_1 the number of kernels in the first layer. We use this notation to indicate that the first dimension is defined by the number of input images, which is 1 for the image SR case. The filter dimensions of the second and third layers are $C_1 \times f_2 \times f_2 \times C_2$ and $C_2 \times f_3 \times f_3 \times 1$, respectively. The last layer can only have one kernel in order to obtain an image as output. Otherwise an additional layer with one kernel otherwise a postprocessing or aggregation step is required. The input image Y is bicubically upsampled so that the input (LR) and output (HR) images have the same resolution. This is necessary because upsampling with standard convolutional layers is not possible. A typical image classification architecture often contains pooling and normalization layers, which helps to create compressed layer outputs that are invariant to small shifts and distortions of the input image. In the SR task, we are interested in creating more image details rather than compressing them. Hence the introduction of pooling and normalization layers would be counter productive. The model is trained on patches extracted from images from the ImageNet detection dataset [38], which consists of around 400,000 images.

B. Video Super-Resolution Architectures

It has been shown for model-based approaches that including neighboring frames into the recovery process is beneficial for video SR [2]–[4]. The motion between frames is modeled and estimated during the recovery process and additional information is gained due to the subpixel motions among frames. The additional information conveyed by these small

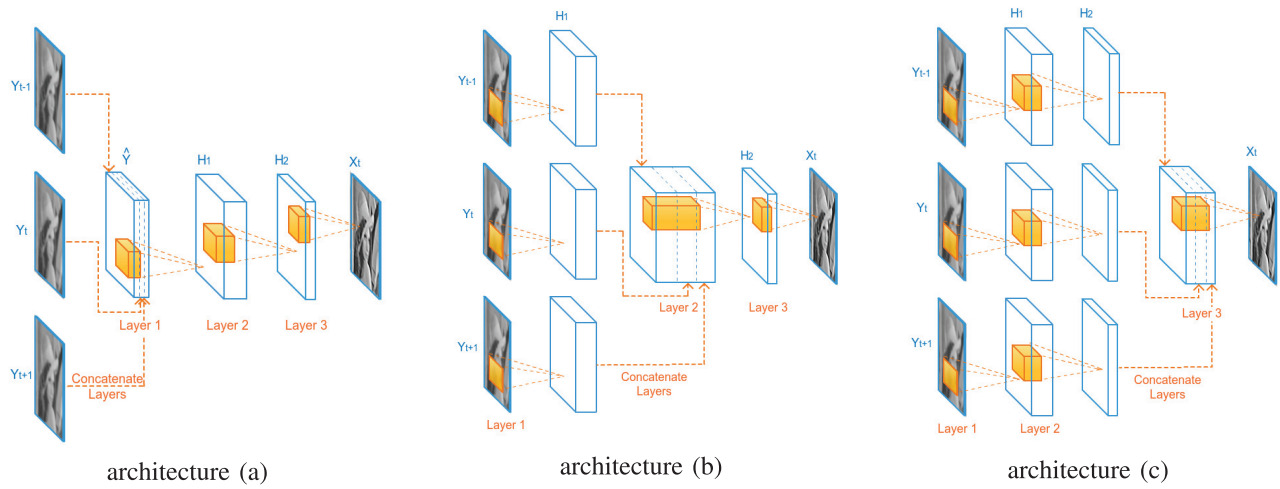

Fig. 2. Video SR architectures: In figure (a), the three input frames are concatenated (Concat Layer) before layer 1 is applied. Architecture (b) concatenates the data between layers 1 and 2, and (c) between layers 2 and 3.

frame differences can also be captured by a learning-based approach, if multiple frames are included in the training procedure.

For the video SR architecture, we include the neighboring frames into the process. Figure 2 shows three options for incorporating the previous and next frames into the process. For simplicity, we only show the architecture for three input frames, namely the previous $(t-1)$, current (t), and next $(t+1)$ frames. Clearly, any number of past and future frames can be accomodated (for example, we use five input frames in the experimental section). In order to use more than one forward- and backward-frame, the architectures in Figure 2 can be extended with more branches. A single input frame has dimensions $1 \times M \times N$, where M and N are the width and height of the input image, respectively. For the architecture in (a), the three input frames are concatenated along the first dimension before the first convolutional layer is applied. The new input data for Layer 1 is 3-dimensional with size $3 \times M \times N$. In a similar fashion we can combine the frames *after* the first layer, which is shown in architecture (b). The output data of layer 1 is again concatenated along the first dimension and then used as input to layer 2. In architecture (c) layers 1 and 2 are applied separately and the data is concatenated between layers 2 and 3. Not only the data size but also the filter dimensions are larger for the video SR architectures. The new filter dimension for architecture (a) for the first layer is $3 \times f_1 \times f_1 \times C_1$, since now we have 3 input frames. The dimensions of layers 2 and 3 do not change. In architecture (b), the filter coefficients of layer 2 increase to $3C_1 \times f_2 \times f_2 \times C_2$, whereas layers 1 and 3 remain the same. Similarly, for architecture (c) the new filter dimension of layer 3 is $3C_2 \times f_3 \times f_3 \times 1$.

C. Weight Transfer From Pretraining

The kernel width, height and their number used in the reference (Fig. 1) and video (Fig. 2) SR networks have to be equal, so that the pretrained filter values from the reference model can be transferred to the video SR models. The only difference is the filter depth in layer 1 for architecture (a), layer 2 for architecture (b) and layer 3 in (c). Using three input frames instead of one, the filter depth in the above mentioned layers is three times larger in the video SR network than in the reference SR one.

The filters of layers 2 and 3 in architecture (a) have the same dimensions as in the reference SR architecture and can be transferred directly from the reference model. The filter dimension of layer 1 in the pretrained model (Figure 1) and the video model are different. The first dimension in the video model is three times larger than in the reference model, as the three input frames are concatenated along the temporal dimension. Furthermore, the output data of layer 1 should be similar to the output data obtained by the single frame SR, as layers 2 and 3 remain the same as in the single frame SR. To properly initialize the video SR model, let us assume that instead of using 3 consecutive frames of a video, we use the same frame three times as input. Hence the output result of the video SR and the image SR systems should be identical. Because layers 2 and 3 are the same in architecture (a) and the reference image SR architecture, we just need to ensure that the input data to layer 2 (output of layer 1) is identical for the two systems. The output data of layer 1, denoted by H_1, for the image SR system has dimensions $M \times N \times C$, where C is the number of kernels[1]. Its elements $h(i,j,c)$ are calculated as

$$h(i,j,c) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} w(m,n,t,c) y_t(i-m, j-n) + b(c), \tag{1}$$

where $w(.)$ are the filter weights, $b(.)$ the biases, c is the kernel index and y_t is the input frame at time t. The weight dimensions are $M \times N \times 1 \times C$. The third dimension is 1 because there is

[1]The input image is zero-padded with $(f_1 - 1)/2$ zeros on each side in order to have the same size for the input and output of the convolution.

only one grayscale input image at time t. The same data for the video SR architecture (a) is calculated as

$$h_v(i,j,c) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{t'=t-1}^{t+1} w_v(m,n,t',c) \times \hat{y}(i-m, j-n, t') + b_v(c), \quad (2)$$

where $w_v(.)$ and $b_v(.)$ are the weights and biases of the video SR model and $\hat{y}$ contains the three consecutive frames y_{t-1},y_t, and y_{t+1}, which are concatenated to a 3 dimensional cube with dimension $M \times N \times 3$. Equation 2 can be expressed in terms of the input images y_{t-1}, y_t and y_{t+1} as

$$\begin{aligned} h_v(i,j,c) = &\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} w_v(m,n,t-1,c) y_{t-1}(i-m, j-n) \\ &+ \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} w_v(m,n,t,c) y_{t}(i-m, j-n) \\ &+ \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} w_v(m,n,t+1,c) y_{t+1}(i-m, j-n) \\ &+ b_v(c) \end{aligned} \quad (3)$$

By setting $h_v = h$ in Equation 3 and replacing y_{t+1} and y_{t-1} by y_t, it is clear that the right-hand side of Equations 1 and 3 are equal, as long as the following two conditions are met for all m, n, c

$$\begin{aligned} w(m,n,t,c) &= w_v(m,n,t-1,c) + w_v(m,n,t,c) \\ &\quad + w_v(m,n,t+1,c) \\ b(c) &= b_v(c), \qquad \forall m,n,c \end{aligned} \quad (4)$$

In our experiments, we initialize the video filter weights $w_v(.)$ and the biases $b_v(.)$ as

$$\begin{aligned} w_v(m,n,t-1,c) &= w_v(m,n,t,c) = w_v(m,n,t+1,c) \\ &= \frac{1}{3} w(m,n,t,c) \\ b_v(c) &= b(c), \qquad \forall m,n,c \end{aligned} \quad (5)$$

which is equivalent to averaging the input images before applying the first convolution layer. The same equations can be applied for the concatenated layers of architectures (b) and (c).

D. Filter Symmetry Enforcement (FSE)

An ideal motion compensated frame would be identical to its reference frame. Hence all input frames would be the same. Training a neural network as shown in Figure 2 with such frames would theoretically lead to equal weights for the separate layers $(t-1)$, (t), and $(t+1)$ in layers 1 and 2 in architecture (c) and in layer 1 in architecture (b). However, even the most advanced optical flow algorithm will not be error free, particularly in cases where frames contain occluded and/or non-rigid objects. If we super-resolve each frame of a video sequence, then each frame will at some point be in the position

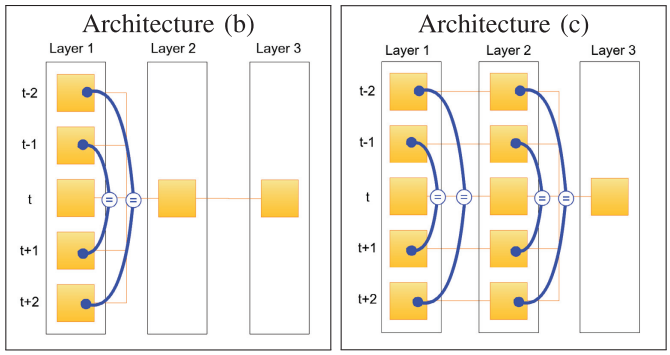

Fig. 3. A schematic display of the filters (yellow squares) in architecture (b) and (c) with filter symmetry enforcement for five input frames. The blue lines connect the filter pairs that are the same.

of the current (t), preceding $(t-1)$ or following $(t+1)$ frames. Statistically the motion compensation error from frame $t-1$ to the current frame t and from frame $t+1$ to frame t should therefore be the same. Thus, we assume that the trained model should end up learning temporally symmetric filters, meaning that the filter weights of filter $(t-1)$ and $(t+1)$ in Layer 1 should be the same. Similarly, all filters in Layer 2 should be the same. Hence, we enforced the weights of the convolutional layer 1 for the pairs $(t-1, t+1)$ and $(t-2, t+2)$ in architecture (b) to be the same. Similarly for architecture (c) we enforced the same weights for the pairs mentioned above for layers 1 and 2. The fact the same filter applied at a given spatial location in the $t-1$ frame can also be applied at the same location in the $t+1$ frame, expressing the same local correlations, indeed extends the convolutional nature of the network in the temporal dimension. The symmetric filter pairs are connected with the blue lines in Figure 3. Architecture (a) has no duplicated filters as the concatenation of the frames happens before the first convolution is applied. For 3 input frames, we simply omit the frame pairs $(t-2, t+2)$.

In order to implement the filter symmetry, we apply the gradients from the backpropagation of the symmetric filters to both filters at the same time. In other words, the symmetric filters share the same weights.

E. Motion Compensation

We tested a number of optical flow estimation algorithms [28]. Both, accuracy of the motion estimates and speed of implementation were taken into account. We chose Druleas algorithm [30] for our framework. The algorithm uses a Combined Local-Global approach with Total Variation (CLG-TV) and demonstrates good results even when large displacements are present.

1) Adaptive Motion Compensation: Motion compensation can be difficult if large motion or motion blur occurs in the video. This can lead to undesired boundary effects and artifacts in the HR reconstruction and will therefore reduce performance. We propose the use of an adaptive motion compensation (AMC) scheme that reduces the influence of neighboring frames for the reconstruction in case of misregistration.

Motion-compensation is applied according to the following equation

$$y_{t-T}^{amc}(i,j) = (1 - r(i,j))y_t(i,j) + r(i,j)y_{t-T}^{mc}(i,j), \quad (6)$$

where $r(i,j)$ controls the convex combination between the reference and the neighboring frame at each pixel location (i,j). y_t is the center frame, y_{t-T}^{mc} is the motion compensated neighboring frame and y_{t-T}^{amc} is the neighboring frame after applying adaptive motion compensation. Similarly to [39], $r(i,j)$ is defined as

$$r(i,j) = exp(-ke(i,j)), \quad (7)$$

where k is a constant parameter and $e(i,j)$ is the motion compensation or misregistration error. Large errors can be due to large motion, occlusion, blurring of the object, or due to the fact that (i,j) is close to a motion boundary. According to Equation 6 and 7 when the motion compensation error $e(i,j)$ is large at the location (i,j), the corresponding weight $r(i,j)$ is small, which means that the adaptively motion compensated pixel is just the pixel in the current frame y_t. In other words, the information from the neighboring frames is not used, since it is not reliable. The image of $r(i,j)$ in Figure 9 provides a map of the accuracy of the motion estimation and compensation. Dark pixels correspond to values close to zero, which means no information of the neighboring frames is utilized. Using the adaptive motion compensation helped to improve the performance for challenging videos, as will be shown in the experimental section.

IV. Experimental Section

In this section we first compare the proposed algorithm to state-of-the-art image and video SR algorithms. Next, we investigate qualitatively and quantitatively the performance of the different proposed video SR architectures, followed by a study of the effects of pretraining, FSE, and motion compensation. At last, we analyze the execution time of our method and compare it to other algorithms.

A. Datasets

We used a publicly available video database [40] which provides high quality movies. The *Myanmar* video sequence was used for the training and testing of our algorithm. The video contains 59 scenes from which we use 53 for training and 6 for testing. We use 4 frames from each test sequence and calculate the average PSNR and SSIM values from the 24 (6×4) resulting test frames as a performance measure. The *Myanmar* video is uncompressed and has 4K resolution (3840×2160 pixels). We downsampled the video by a factor of 4 resulting in a 960×540 pixel resolution, in order to better compare to the state-of-the-art SR algorithms, as most of the published ones (Video and Image) work with images of similar resolution.

Although we do not use the same shots of the *Myanmar* video sequence for testing and training, it can be argued that the different shots share a similar style as they are from the same video. We therefore chose to test our algorithm on a set of other videos from a different source. Our second dataset, referred to as *Videoset4*, consists of the four test videos *walk*, *foliage*, *city*, and *calendar* which were also used in [3]. The specifications of the videos are shown in Table I. We skip the first and last two frames of each video in order to always have a complete set of 5 consecutive frames for the video SR process.

TABLE I
Specifications of Videoset4

	walk	foliage	city	calendar
Resolution	720×480	720×480	704×576	720×576
Nr Frames	43	45	30	37

B. CNN Model Parameters

We implemented our proposed algorithm with the Caffe framework [41]. The image SR network for the pretraining was implemented with the settings proposed in [17]. The network has 3 convolutional layers (see Figure 1), where layers 1 and 2 are each followed by a ReLU. Layer 1 has 64 kernels with a kernel size of 9×9, Layer 2 has 32 kernels with size 5×5 and the third layer has one 5×5 kernel. The filters of the video SR architectures have the same configurations as the image SR architecture.

C. Training Procedure

Following the literature ([6], [17]), we converted the images and videos into the YCbCr colorspace and only used the luminance channel (Y) for training, testing, and PSNR/SSIM calculations. For the color images shown in this paper, we bicubically upsampled the chrominance channels, Cb and Cr. In order to create the video training set, we extracted sets of 5 consecutive frames from the training video scenes. Then, we downsampled them by the desired factors 2, 3, or 4 using the Matlab implementation of *imresize* and upsampled the resulting LR frames with bicubic interpolation to their original resolution. Afterwards, we calculated the optical flow from the first and the last two frames towards the center frame and computed the motion compensated frames. From the resulting 5 frames (4 motion compensated and one center frame) we extracted $36 \times 36 \times 5$ data cubes, that is, 36×36 pixel patches from 5 consecutive frames. We dismissed patches/data cubes if they did not contain sufficient structures. Patches were excluded if their pixel variance did not exceed 0.0035[2]. The pixel variance may not be the best measure of patch structure, but it is sufficient for our purpose. The created training database consists of about 900,000 data cubes.

In order to optimize the filter weights and biases in the training phase, we need to define a loss function that will be minimized. The Euclidean distance between the output image and the ground truth image, which is known for the training dataset, is the measure we use, since also the Peak Signal-to-Noise Ratio (PSNR) performance measure is directly related to the Euclidean distance. In order to avoid border effects during

[2]The image intensity values are scaled to the range [0,1].

TABLE II
AVERAGE PSNR AND SSIM VALUES FOR THE *Myanmar* TEST SEQUENCES. THE RESULT OF THE PROPOSED METHOD (VSRNET) IS SHOWN IN THE LAST COLUMN

MYANMAR	Scale	Image SR Algorithms				Video SR Algorithms				Own
		Bicubic	SrSC [6]	A+ [10]	SRCNN [17]	ANN [20]	Bayesian [3]	Bayesian-MB [5]	Enhancer [42]	VSRnet
PSNR	2	34.59	-	37.19	37.79	35.18	35.56	36.41	35.94	**38.48**
SSIM	2	0.9458	-	0.9638	0.9640	0.9501	0.9515	0.9599	0.9588	**0.9679**
PSNR	3	31.59	32.71	33.48	33.88	32.55	32.20	32.74	32.50	**34.42**
SSIM	3	0.8957	0.9127	0.9191	0.9198	0.9095	0.9203	0.9174	0.9099	**0.9247**
PSNR	4	29.53	-	30.88	31.26	29.94	30.68	29.29	30.23	**31.85**
SSIM	4	0.8526	-	0.8777	0.8777	0.8560	**0.8895**	0.8639	0.8681	0.8834

TABLE III
AVERAGE PSNR AND SSIM VALUES FOR THE *Videoset4* DATASET. THE RESULT OF THE PROPOSED METHOD (VSRNET) IS SHOWN IN THE LAST COLUMN

Videoset4	Scale	Image SR Algorithms				Video SR Algorithms				Own
		Bicubic	SrSC [6]	A+ [10]	SRCNN [17]	ANN [20]	Bayesian [3]	Bayesian-MB [5]	Enhancer [42]	VSRnet
PSNR	2	28.43	-	30.53	30.70	29.04	29.69	30.63	30.40	**31.30**
SSIM	2	0.8676	-	0.9154	0.9172	0.8835	0.9055	0.9226	0.9151	**0.9278**
PSNR	3	25.28	26.01	26.36	26.51	25.94	25.82	26.43	26.34	**26.79**
SSIM	3	0.7329	0.7788	0.7904	0.7933	0.7705	**0.8323**	0.8071	0.7948	0.8098
PSNR	4	23.79	-	24.59	24.69	23.97	**25.06**	24.14	24.55	24.84
SSIM	4	0.6332	-	0.6889	0.6918	0.6437	**0.7466**	0.6864	0.6877	0.7049

the training, we can either add zero padding to the 36×36 pixel patches (which we do not do) or allow the output of the convolution to be of smaller size; that is the output patch shrinks with each convolutional layer by (filter size - 1). This shrunk output patch corresponds to the 20×20 center pixels of the original patch. These center pixels are then used for the calculation of the loss function. The model weights provided by SRCNN [17] were used for the image SR pretraining. In the video SR training we used a batch size of 240, a learning rate of 10^{-4} for the first two layers and 10^{-5} for the last layer and a weight decay rate of 0.0005 [41]. All the results shown in the experimental Section are evaluated after 200,000 iterations if pretraining was used, and 400,000 iterations otherwise. In all cases we tested convergence has been achieved with such a number of iterations. A reduction of the learning rate did not lead to any further improvements. Therefore, we kept the learning rate constant throughout the whole training. It took about 22 hours to train a network model as used in Tables II and III, not including the pretraining.

D. Comparison to the State-of-the-Art

We compare our algorithm, henceforth refered to as Video SR network (VSRnet), to single frame and video SR algorithms. Bicubic interpolation is included as a baseline. Super-resolving a video can be achieved by simply applying Image SR to each frame separately. Therefore we also compared our algorithm to the state-of-the-art image SR algorithms. These are dictionary-based Sparse Coding SR (ScSR) [6], adjusted anchored neighbor regression (A+) [10] and SRCNN [17]. The implementations and parameters we used were provided by the authors of the paper. Furthermore we compared to the state-of-the-art video SR algorithms. All video SR methods were tested using $\pm$ 2 neighboring frames. The *Enhancer* [42] is a commercially available software, which we tested on the highest quality settings. The source code of the Bayesian adaptive video SR method from [3] (Bayesian) is unavailable, thus, we used the reimplementation provided and used by Ma *et al.* [5] in their paper. We also tested the Bayesian method described in [5] (Bayesian-MB), which can handle motion blurred videos. The parameters of the Bayesian methods were tuned for each video. The implementation of the Artificial Neural Network (ANN) architecture described in [20] was not available either.

We reimplemented it with two changes which improved its performance. The first change in our implementation is that we used 19×19 pixel bicubically upsampled input patches instead of the 5×5 pixel input patches used in [20]. The reason for this is that we wanted to perform a direct comparison with our approach which uses a bicubically interpolated input. The second change was to replace the sigmoid activation function used in [20] with ReLU since the latter provides faster convergence. Both changes led to a better PSNR. We used the same database for the training of the two networks so we believe that the experimental comparison is fair.

We used architecture (b) of VSRnet for the results in Tables II and III since it provides the best performance among the three proposed architectures. Table II shows the PSNR and SSIM values for the different algorithms tested on the *Myanmar* sequence. The same algorithms have been tested on the *Videoset4* dataset with the results shown in Table III. The proposed VSRnet provides the highest average PSNR in all experiments except for upscale factor 4 and Videoset4, where the Bayesian method outperforms VSRnet by 0.22 dB. However, in average VSRnet outperforms the Bayesian method by 1.45 dBs per testcase. In terms of SSIM, VSRnet is only outperformed by the Bayesian method in three cases. On the *Myanmar* dataset the PSNR margin to the second best algorithm (SRCNN) is 0.69 dB for upscale factor 2, 0.54 dB for factor 3 and 0.59 dB for factor 4. The gain for the *Videoset4* is lower with 0.60 dB and 0.28 dB for upscale factors 2 and 3, respectively. The results indicate that the gain with Video SR is higher for lower upscaling factors, which is most likely due to the more accurate motion compensation.

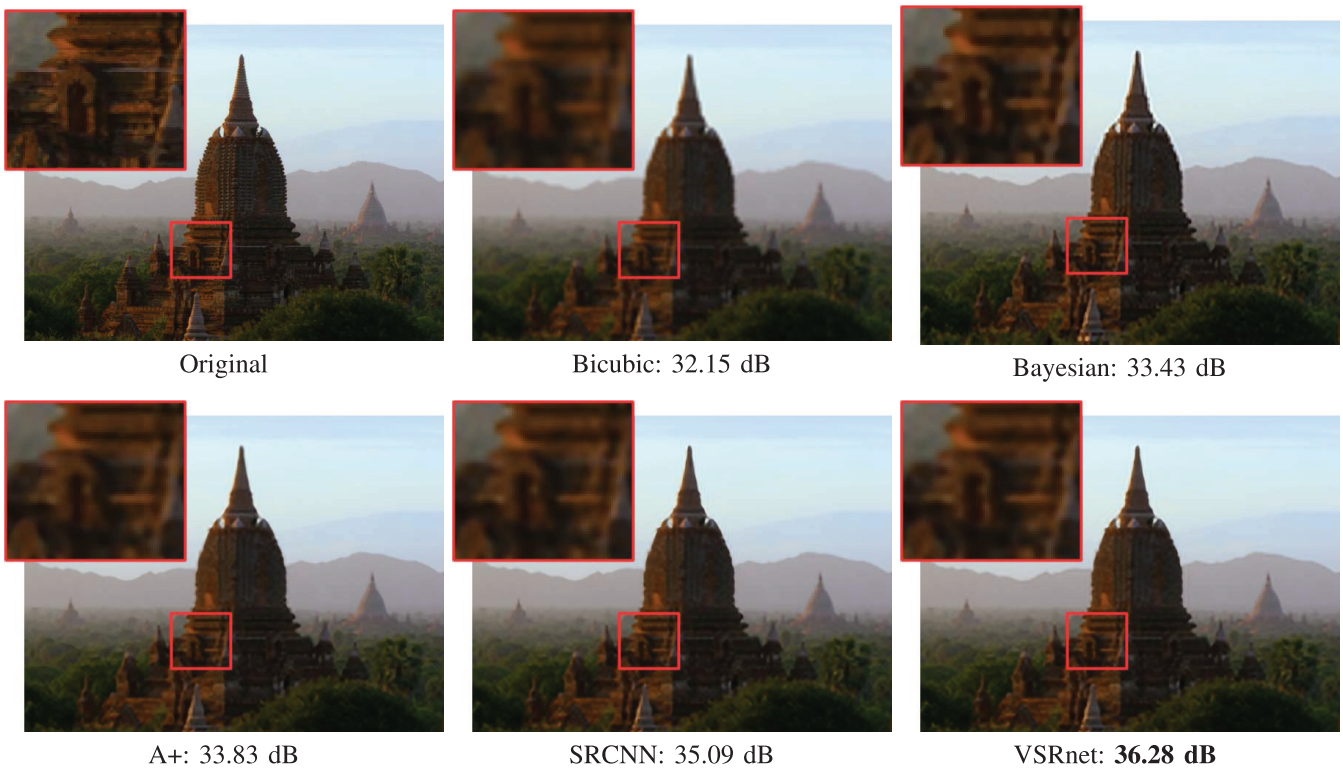

Fig. 4. Comparison to the State-of-the-Art: SR frames from the Myanmar video compared to our method (VSRnet) for upscale factor 4.

Figure 4 shows example frames and their corresponding PSNR values from the *Myanmar* test set using upscale factor 4. Figure 5 shows an example frame from the *city* and *calendar* videos in *Videoset4* for upscale factor 2. We show the original frame and the result obtained by bicubic interpolation, and our proposed algorithm (VSRnet). In addition, we show the results of SRCNN, A+ and the Bayesian method. The door shown in the zoomed rectangle in Figure 4 was reconstructed more accurately by the proposed VSRnet algorithm compared to the other algorithms. We can also see a more accurate line reconstruction in the *city* frame and a better reconstruction of the letter "M" in the *calendar*. The Bayesian method applies too much deblurring which is why it appears to create sharp images despite the lower PSNR. There are visible ringing artifacts around the tower in the Bayesian reconstruction of the *Myanmar* sequence and ripples in the *Calendar* sequence.

E. Architecture Comparison

In this section we compare the different configuration parameters of the proposed SRnet. The PSNR results for the *Myanmar* test set are shown in Table IV. The concatenation of the multiple frames can happen at different layers of the network. Figures 2 (a), (b) and (c) show the configurations where the frames are concatenated at the first, second and third layer, respectively. In order to show the benefits of the video training, we added the last column (AVG), where we apply single frame SR to each frame separately and then average the resulting images. The best PSNR is achieved with architecture (b); however, the performance of all three architecture types is similar. We also show the advantage of the proposed architectures over single frame SR.

Figure 6 shows the PSNR versus the training time for 200,000 iterations for each architecture type with 3 input frames (red) and 5 input frames (blue). The PSNR performances are evaluated on the *Myanmar* test video sequences shown in Table IV. The graph shows that architecture (a) requires the least training time, followed by architecture (b) and (c). Combining the frames at a later stage introduces more weights to the network and therefore leads to the longer training time. The number of weights in layer 1 for example is the same in architectures (a), (b) and (c). Layer 2 however has 3 times more weights to learn in (b) and (c) than in (a). Equivalently, Layer 3 in (c) has 3 times more weights than in (a) and (b). Furthermore, using more input frames also leads to more weights, hence to a longer training time. Using more input frames only affects the first layer in architecture (a), the increase in training time is therefore not as big as for (b) and (c), which is visible in Figure 6.

F. Pretraining

In this section we demonstrate the importance of pretraining. Table V shows the PSNR for architecture (b) with and without pretraining. The filters of the models without pretraining were initialized with random Gaussian distributed values with standard deviation of 0.001 and were trained for 400,000 iterations. The pretrained models were initialized with the filter weights from the single frame SR models. The performance of these models did not improve anymore after 200,000 iterations due

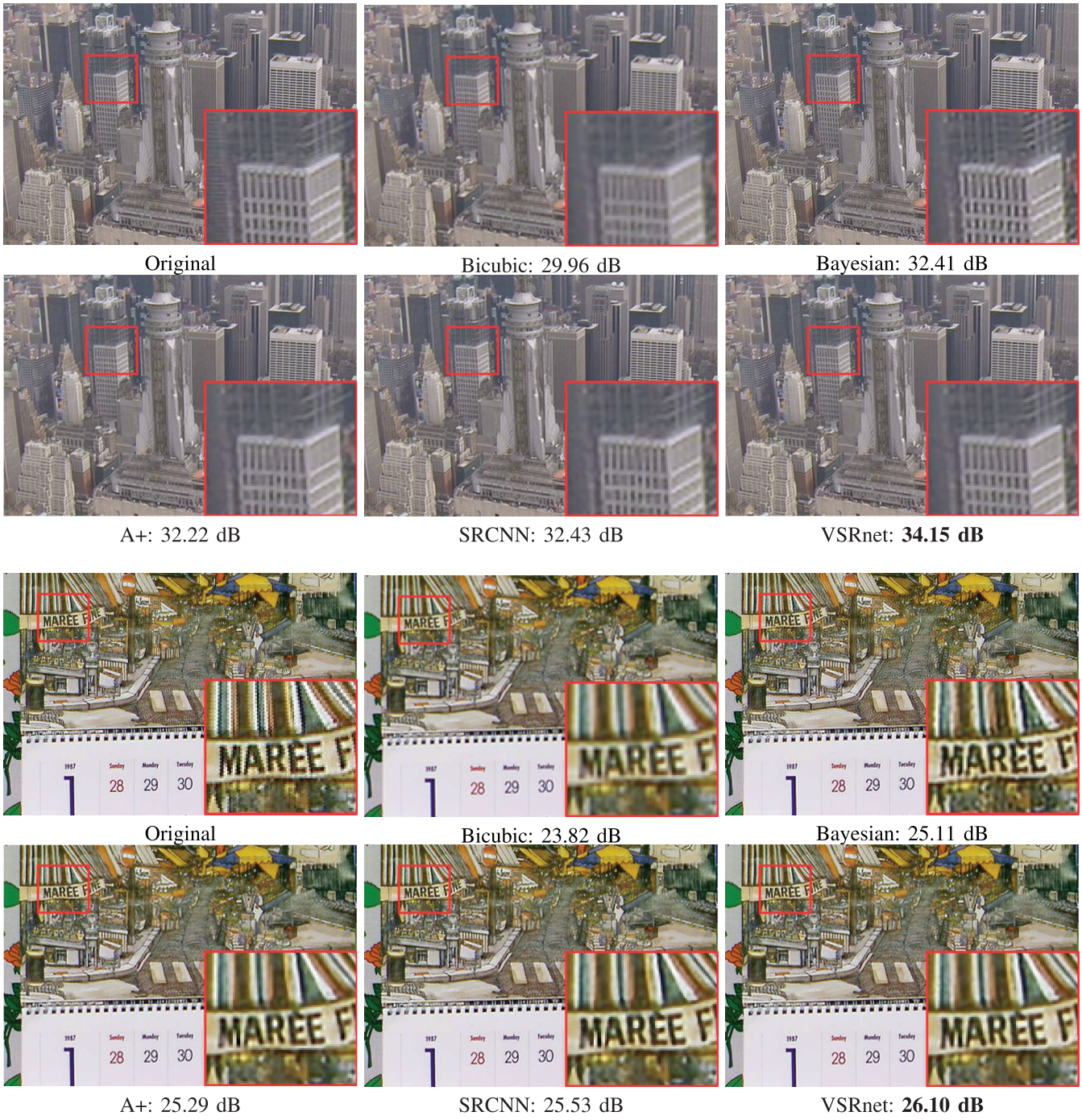

Fig. 5. Comparison to the State-of-the-Art: SR frames from the *city* and *calendar* videos compared to our method (VSRnet) for upscale factor 2.

to the pretrained values. We can see that even without pretraining we achieve PSNR values comparable to the state-of-the-art. However using pretraining improves the PSNR by an additional 0.62 dBs.

In Figure 7, we show the filter values of the first convolutional layer of architecture (b) trained for upscale factor 3 and 3 input frames. We only show the first layer because these filters are two dimensional and can be easily displayed as images. The first row shows the filters learned with pretraining after 200'000 iterations, and the second row the same filters trained without pretraining using random initialization after 400'000 iterations. The resulting filters with pretraining did not change much from the initial values. The filter for all 3 time instances $(t-1)$, t, and $(t+1)$ look very similar, since these filters were all initialized with the same values from pretraining. The resulting filters after 400,000 iterations are shown in the second row. We can see in this case that different filter pattern emerge for the different time instances. Due to the independent random initialization of the filter coefficient values, they do not develop any symmetric patterns. We can see a diagonal pattern in the filters of the center instance t whereas the filters in the instances $(t-1)$ and $(t+1)$ tend to be more round shaped. Note that

TABLE IV
COMPARISON OF THE DIFFERENT ARCHITECTURES SHOWN IN FIGURE 2. THE TABLE SHOWS THE AVERAGE PSNR OBTAINED FROM THE *Myanmar* DATASET. AS A REFERENCE, WE SHOW THE LAST COLUMN (AVG), WHERE WE APPLY SINGLE FRAME SR TO EACH FRAME AND THEN AVERAGE THE RESULTING FRAMES. IF ONLY ONE FRAME IS USED, ARCHITECTURES (A), (B), AND (C) ARE ALL EQUIVALENT TO THE REFERENCE SINGLE FRAME ARCHITECTURE DESCRIBED IN FIGURE 1, HENCE THE PSNRS FOR ONLY ONE FRAME ARE ALL THE SAME

Architecture	(a)	(b)	(c)	AVG
1 frame	-	-	-	31.26
3 frames	31.77	**31.81**	31.80	31.31
5 frames	31.80	**31.85**	31.82	31.33

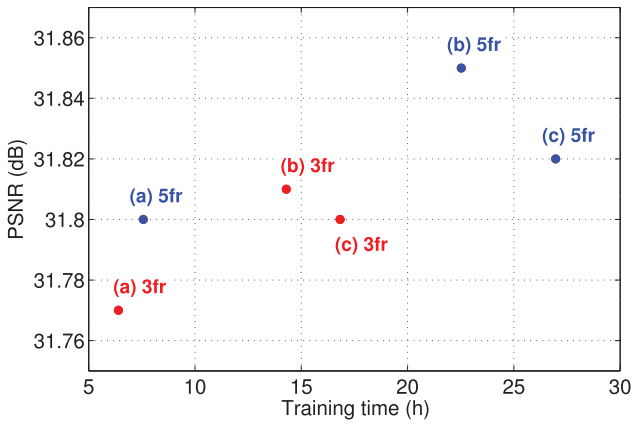

Fig. 6. Training time versus PSNR of different architectures.

TABLE V
PSNR OF THE MYANMAR TESTSET OF ARCHITECTURE (B) FOR UPSCALE FACTOR 3 WITH 3 INPUT FRAMES WITH AND WITHOUT PRETRAINING

	Pretraining 200,000 Iter.	No Pretraining 400,000 Iter
PSNR	34.33	33.71

some of the filters appeared to be "dead". The same observation was made in [17] and [43]. The uniform or random filters (dead filters) are a side-effect of the ReLU unit. A large update step can cause the weights to update in a way that the neuron will never activate again. Future gradients will therefore always be zero and the neuron will not be updated anymore, which leads to the observed dead filter. There are different ways to reduce the number of dead filters such as leaky ReLU [37], gradient clipping, reduction of learning rate and different initialization values and constant biases. Using pretraining definitely helps reducing their number. From these methods, reducing the learning rate and lowering the standard deviation of the initialization values proofed to be most effective.

G. Filter Symmetry Enforcement

In this section, We trained the architecture (c) with 3 input frames to demonstrate the effect of the Filter Symmetry Enforcement (FSE) described in Section III-D. Figure 8 shows the reconstruction error during training with and without FSE. After 200,000 iterations, both models converged to the same reconstruction error. The PSNR difference of models trained with and without FSE lay within ≤ 0.01 dBs; however we can see that the same reconstruction error achieved after 100,000 iterations without FSE only requires 82,000 iterations when using FSE. The faster convergence during the training phase can be explained with the reduced number of unknown weights. Using FSE accelerates the training procedure by almost 20% while maintaining the same performance.

H. Motion Compensation (MC)

We trained the VSRnet models with and without motion compensated frames to show the effect of MC. We calculated the resulting average PSNRs for both cases. We can see that using MC improves the PSNR by 0.22 dBs. Training with MC and testing without MC or vice versa is not consistent and has a negative impact on the PSNR, as expected. By applying motion compensation, the neighboring frames become very similar to the center frame. The remaining differences between the reference and the motion compensated frames contain information that is beneficial for the video SR learning process. In order to verify this, we trained a model where we replaced the 5 consecutive input frames with the center frame. Therefore for this training there is no temporal information available. This model is comparable with the single frame architecture in Figure 1 but with 5 times more filter coefficients. This model achieved an average PSNR of 30.18 dB, which is 0.42 dB lower than the MC result presented in Table VI.

1) Adaptive Motion Compensation (AMC): The average PSNR of AMC on our test videos is 30.43 dB, which is 0.17 dB lower than with normal MC. However, the use of AMC led to significant improvements on frames with strong motion blur and fast moving objects. Figure 9 shows two example frames which experience strong motion blur. The first frame is from the *walk* sequence where two pigeons are flying through the scene and the second one is from the *foreman* sequence, briefly showing a hand of the foreman. The images on the right of the original images show the r values from equation 7 averaged over the four neighbor frames, where a dark color corresponds to a small r or to a large motion compensation error, respectively. The pigeons and the hand are clearly visible as dark spots in the r-value images. We tested the frames using the proposed VSRnet algorithm with AMC and standard MC, where we set the constant k from Equation 7 to $1/8$. In addition, we tested the Bayesian-MB method [5], which is designed to handle motion blurred frames. The results are shown in Figure 10. The standard MC approach fails to estimate the motion of these objects which leads to a poor SR reconstruction. Even the Bayesian-MB method produces artifacts in the shape of small dots in both examples. The AMC on the other hand successfully reconstructs the pigeon and the hand with a PSNR improvement of 0.27 dB and 0.95 dB respectively.

I. Execution Time Analysis

We measured the average runtime to super-resolve one frame from a video sequence with a resolution of 704×576 pixels. We used the average PSNR value of upscale factor 3 from all four videos from the *Videoset4* and the *Myanmar* test sequence.

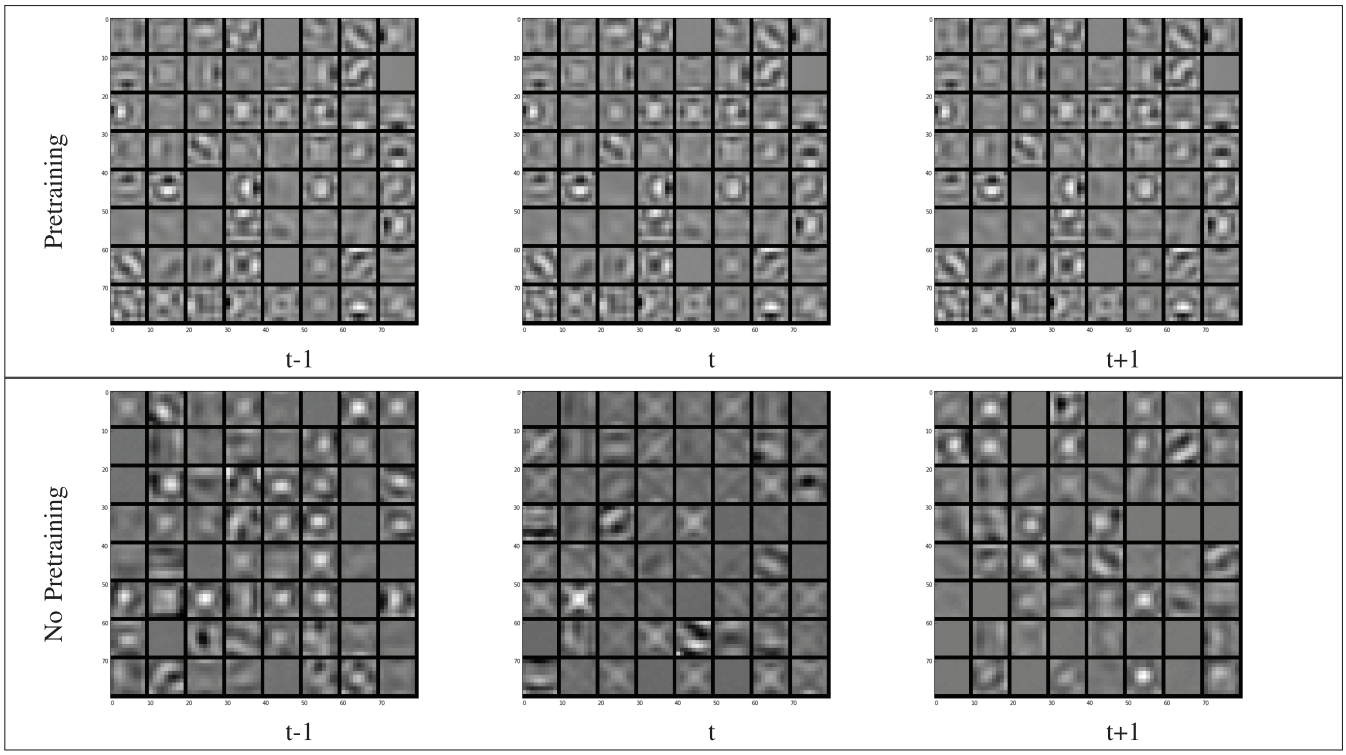

Fig. 7. Filters from the first layer of architecture (b) for upscale factor 3. The first row shows the filter coefficients after 200,000 iterations with pretraining. Each figure shows 64 filters each of size 9×9. The second and row shows the resulting filter coefficients after training with random initialization after 400,000 iterations.

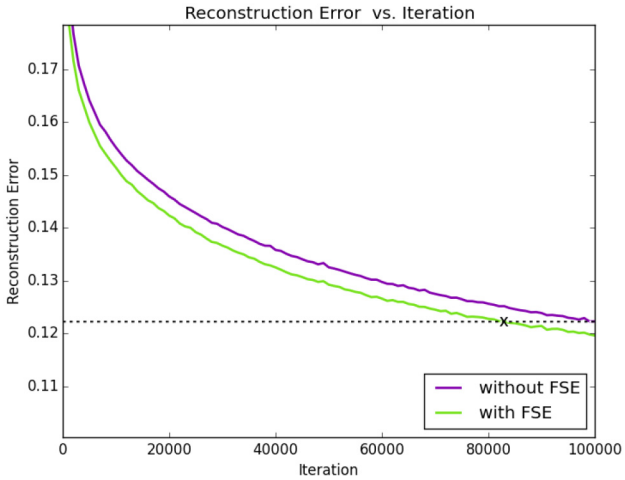

Fig. 8. The reconstruction error of architecture (c) with 3 frames during the training with and without Filter Symmetry Enforcement (FSE).

The test was performed on a Linux workstation with an Intel Xeon E5-2630 processor with 2.4 GHz and 128 GB RAM, and a NVIDIA Geforce GTX Titan Z graphics card with 2880 cores per GPU unit[3]. The Enhancer and the Bayesian-MB algorithm was only available for the Windows operating system and had to be tested on a different workstation which runs with an Intel Xeon E3-1245 3.3 GHz processor. The average runtime of the Enhancer on an equivalent workstation can therefore

[3]The Titan Z card has 5760 cores split between two GPU units. We were utilizing only one GPU unit for the testing.

TABLE VI
AVERAGE PSNR WITH AND WITHOUT MOTION COMPENSATION (MC) FOR UPSCALE FACTOR OF 3 FOR THE *Myanmar* AND THE *Videoset4* VIDEOS COMBINED. THE MOTION COMPENSATION WAS APPLIED TO BOTH THE TRAINING AND TEST SETS. ALL POSSIBLE COMBINATION OF TRAINING AND TESTING WERE EVALUATED

		Training Set	
		No MC	MC
Test Set	No MC	30.38	29.43
	MC	30.31	**30.60**

be slightly different on a system specified before. Figure 11 shows the runtime and the performance of the different algorithms as well as the proposed VSRnet algorithm. Image SR algorithms are shown in red color, video SR algorithms in magenta and the proposed VSRnet in blue. The runtime of VSRnet, ANN and SRCNN was measured with and without GPU acceleration. Despite GPU support, we have a slow runtime for the ANN algorithm, because the video data has to be processed patch-wise. The other algorithms were not available with GPU acceleration and were therefore measured without GPU support.

The runtime of VSRnet is mainly determined by the motion compensation, which takes about 55 second per frame. The neural net computation itself only takes 0.24 seconds with GPU or 16 seconds without GPU support. VSRnet-NoMC and VSRnet-NoMC-GPU show the runtime and the PSNR of the proposed algorithm without using motion compensation (MC). Despite the PSNR loss of about 0.23 dB, VSRnet-noMC still

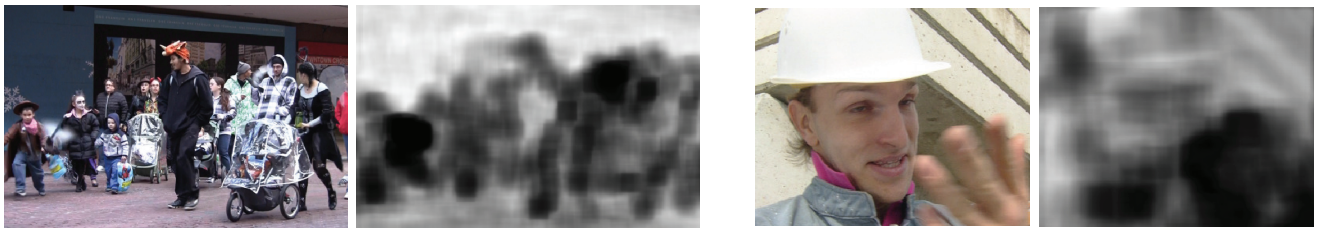

Fig. 9. The original center frame and its average r value of the four motion compensated neighbor frames for the *walk* and *foreman* sequence.

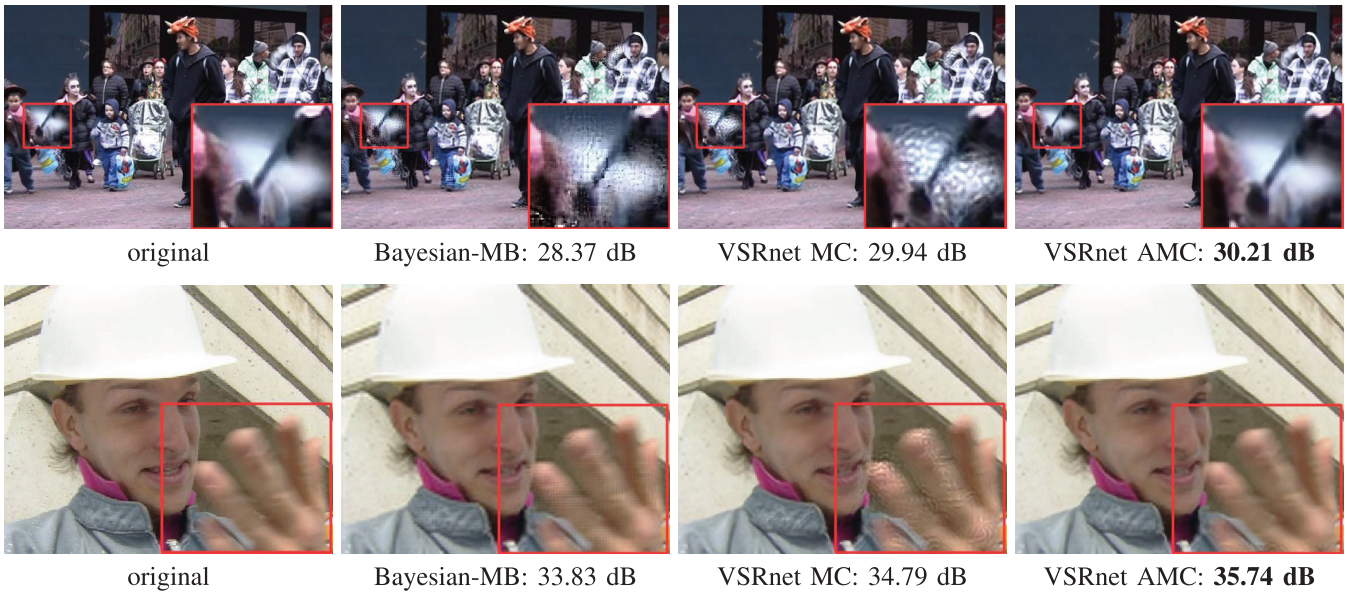

Fig. 10. Frames with motion blurred objects from the *walk* and *foreman* sequence, reconstructed with MC and AMC for upscale factor 3.

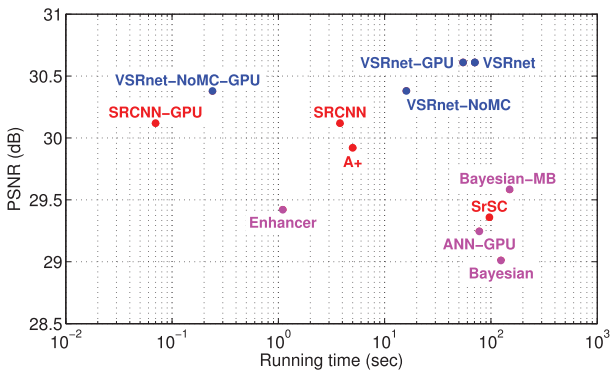

Fig. 11. Runtime/PSNR comparison. The runtime for SRCNN, ANN and VSRnet are also shown with GPU acceleration which is denoted with (-GPU). VSRnet-NoMC is the runtime of the proposed algorithm without using motion compensation.

outperforms the state-of-the art while using only about 240 milliseconds to compute one frame.

Methods such as SrSC [6] or the Bayesian approach [3] are iterative algorithms and therefore computationally expensive which leads to slow running times. The output of a CNN-based approach on the other hand is calculated as a single feed-forward process. Their computation time is very fast even without GPU acceleration. Using GPU support, the runtime of the VSRnet can be further reduced from 16 seconds per frame without GPU to 0.24 seconds per frame with GPU, which is about 66 times faster.

V. Conclusion

In this paper we have introduced a video SR algorithm using convolutional neural nets. Our proposed CNN exploits spatial as well as temporal information. We have investigated different architectures and have shown their advantages and disadvantages. Using motion compensated input frames, filter symmetry enforcement and a pretraining method we were able to improve the reconstruction quality and reduce the training time. Finally, we introduced an adaptive motion compensation scheme to deal with motion blur and fast moving objects. We presented an algorithm that outperforms the current state-of-the art algorithms in video SR.

References

[1] S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Variational Bayesian super resolution," *IEEE Trans. Image Process.*, vol. 20, no. 4, pp. 984–999, Apr. 2011.

[2] S. P. Belekos, N. P. Galatsanos, and A. K. Katsaggelos, "Maximum a posteriori video super-resolution using a new multichannel image prior," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1451–1464, Jun. 2010.

[3] C. Liu and D. Sun, "On Bayesian adaptive video super resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 346–360, Feb. 2014.

[4] A. K. Katsaggelos, R. Molina, and J. Mateos, "Super resolution of images and video," *Synth. Lect. Imag. Video Multimedia Process.*, vol. 1, no. 1, pp. 1–134, 2007.
[5] Z. Ma, J. Jia, and E. Wu, "Handling motion blur in multi-frame super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, vol. 1.
[6] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.
[7] J. Yang, Z. Wang, Z. Lin, X. Shu, and T. Huang, "Bilevel sparse coding for coupled feature spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 2360–2367.
[8] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces*. New York, NY, USA: Springer, 2012, pp. 711–730.
[9] S. Wang, L. Zhang, Y. Liang, and Q. Pan, "Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 2216–2223.
[10] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. IEEE Asian Conf. Comput. Vis.*, 2014, pp. 1920–1927.
[11] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3791–3799.
[12] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 5197–5206.
[13] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2009, pp. 349–356.
[14] B. C. Song, S. C. Jeong, and Y. Choi, "Video super-resolution algorithm using bi-directional overlapped block motion compensation and on-the-fly dictionary training," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 3, pp. 274–285, Mar. 2011.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[16] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. Int. Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1–9.
[17] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.
[18] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2014, pp. 1–16.
[19] Z. Wang *et al.*, "Self-tuned deep super resolution," in *Proc. Comput. Vis. Pattern Recog. Workshops*, 2015, pp. 1–8.
[20] M. Cheng, N. Lin, K. Hwang, and J. Jeng, "Fast video super-resolution using artificial neural networks," in *Proc. 8th Int. Symp. Commun. Syst. Netw. Digital Signal Process. (CSNDSP)*, 2012, pp. 1–4.
[21] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, "Video super-resolution via deep draft-ensemble learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp 531–539.
[22] J. Deng, W. Dong, R. Socher, and L. Li, "A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 248–255.
[23] A. Karpathy and G. Toderici, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1725–1732.
[24] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2012.
[25] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[26] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D, Nonlinear Phenom.*, vol. 60, nos. 1–4, pp. 259–268, 1992.
[27] H. Takeda, P. Milanfar, M. Protter, and M. Elad, "Super-resolution without explicit subpixel motion estimation," *IEEE Trans. Image Process.*, vol. 18, no. 9, pp. 1958–1975, Sep. 2009.
[28] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.
[29] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 2432–2439.
[30] M. Drulea and S. Nedevschi, "Total variation regularization of local-global optical flow," in *Proc. IEEE Conf. Intell. Transp. Syst. (ITSC)*, 2011, pp. 318–323.
[31] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1–9.
[32] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 2392–2399.
[33] V. Jain and H. Seung, "Natural image denoising with convolutional networks." in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2008, pp. 1–8.
[34] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Scholkopf, "A machine learning approach for non-blind image deconvolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1067–1074.
[35] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2013, pp. 633–640.
[36] J. Masci, U. Meier, D. Cirean, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning—ICANN*. New York, NY, USA: Springer, vol. 1, pp. 52–59, 2011.
[37] A. Maas, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, vol. 28, p. 1.
[38] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[39] M. K. Park, M. G. Kang, and A. K. Katsaggelos, "Regularized high-resolution image reconstruction considering inaccurate motion information," *Opt. Eng.*, vol. 46, no. 11, p. 117004, 2007.
[40] Myanmar 60p, Harmonic Inc. (2014). [Online]. Available: http://www.harmonicinc.com/resources/videos/4k-video-clip-center.
[41] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," arXiv preprint arXiv:1408.5093, 2014.
[42] Infognition. (2010). *Video Enhancer* [Online]. Available: http://www.infognition.com/videoenhancer
[43] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Comput. Vis. (ECCV'14)*, 2014, vol. 8689, pp. 818–833.

Armin Kappeler received the B.S. degree in electrical engineering from the Hochschule fuer Technik Rapperswil (HSR), Rapperswil, St. Gallen, Switzerland, in 2004, and the M.S. degree in electrical engineering from the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA, in 2012, where he is currently pursuing the Ph.D. degree in electrical engineering. In 2015, he joined Yahoo Inc., Sunnyvale, CA, USA, where he is currently working on an image classification algorithm using deep neural networks. His research interests include deep neural networks for image and video restoration and classification, computer vision, and machine learning.

Seunghwan Yoo received the B.E and M.S degrees in electrical engineering from Sogang University, Seoul, South Korea, in 2005 and 2007, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at Northwestern University, Evanston, IL, USA. He joined the Image and Video Processing Laboratory in 2013. His research interests include image and video super-resolution and deconvolution.

Qiqin Dai received the B.S. degree in automation from Zhejiang University, Hangzhou, China, in 2012. He is currently pursuing the Ph.D. degree at the Image and Video Processing Laboratory, Northwestern University, Evanston, IL, USA. His research interests include digital image processing, computer vision, computational photography, and high dynamic range imaging.

Aggelos K. Katsaggelos (F'98) received the diploma degree in electrical and mechanical engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece, in 1979, and the M.S. and Ph.D. degrees in electrical engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 1981 and 1985, respectively. In 1985, he joined the Department of Electrical Engineering and Computer Science, Northwestern University, where he is currently a Professor of the Joseph Cummings Chair. He was previously the Ameritech Chair of the Information Technology and the AT&T Chair. He is also a member of the Academic Staff, NorthShore University Health System, Evanston, IL, USA, an Affiliated Faculty with the Department of Linguistics and he has an appointment with the Argonne National Laboratory. He has authored extensively in the areas of multimedia signal processing and communications (over 230 journal papers, 500 conference papers, and 40 book chapters) and is the holder of 25 international patents. He is the coauthor of *Rate-Distortion Based Video Compression* (Kluwer, 1997), *Super-Resolution for Images and Video* (Claypool, 2007), *Joint Source-Channel Video Transmission* (Claypool, 2007), and *Machine Learning Refined* (Cambridge University Press, 2016). He has supervised 54 Ph.D. theses so far. Among his many professional activities Prof. Katsaggelos was the Editor-in-Chief of the *IEEE Signal Processing Magazine* (1997–2002), a BOG Member of the IEEE Signal Processing Society (1999–2001), a member of the Publication Board of the IEEE Proceedings (2003–2007), and he is currently a Member of the Award Board of the IEEE Signal Processing Society. He is a Fellow of the SPIE (2009). He was the recipient of the IEEE Third Millennium Medal (2000), the IEEE Signal Processing Society Meritorious Service Award (2001), the IEEE Signal Processing Society Technical Achievement Award (2010), an IEEE Signal Processing Society Best Paper Award (2001), an IEEE ICME Paper Award (2006), an IEEE ICIP Paper Award (2007), an ISPA Paper Award (2009), and a EUSIPCO Paper Award (2013). He was a Distinguished Lecturer of the IEEE Signal Processing Society (2007–2008).