Deep Video Super-Resolution Network Using Dynamic Upsampling Filters Without Explicit Motion Compensation

Younghyun Jo Seoung Wug Oh Jaeyeon Kang Seon Joo Kim
Yonsei University

Abstract

Video super-resolution (VSR) has become even more important recently to provide high resolution (HR) contents for ultra high definition displays. While many deep learning based VSR methods have been proposed, most of them rely heavily on the accuracy of motion estimation and compensation. We introduce a fundamentally different framework for VSR in this paper. We propose a novel end-to-end deep neural network that generates dynamic upsampling filters and a residual image, which are computed depending on the local spatio-temporal neighborhood of each pixel to avoid explicit motion compensation. With our approach, an HR image is reconstructed directly from the input image using the dynamic upsampling filters, and the fine details are added through the computed residual. Our network with the help of a new data augmentation technique can generate much sharper HR videos with temporal consistency, compared with the previous methods. We also provide analysis of our network through extensive experiments to show how the network deals with motions implicitly.

1. Introduction

The goal of super-resolution (SR) is to generate a high-resolution (HR) image or video from its corresponding low-resolution (LR) image or video. SR is widely used in many fields ranging from medical imaging [7] to satellite imaging [2] and surveillance [38]. With the advances in the display technology, the video super-resolution (VSR) for LR videos are becoming more important as the ultra high definition televisions target 4K (3840×2160) and 8K resolution (7680×4320), but the contents that match such high resolution are still scarce.

With the success of deep learning in computer vision as in image classification [19] and image segmentation [25], the deep learning based single-image super-resolution (SISR) methods have emerged [3, 17, 31, 21, 33, 20, 35, 30]. These methods are showing state-of-the-art performances in terms of the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [37].

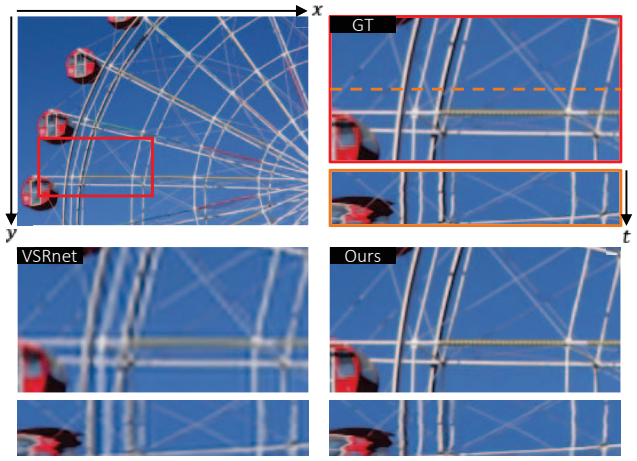


Figure 1. $\times 4$ VSR for the scene *ferriswheel*. To visualize the temporal consistency in 2D, we also plot the transition of the dotted orange horizontal scanline over time in the orange box with the $x-t$ axis. We can observe that our method produces much sharper and temporally consistent HR frames compared with VSRnet [16].

A straightforward way to perform VSR is to run SISR frame by frame. However, since the SISR methods do not consider the temporal relationship between frames, there is a high possibility that consecutive frames are not connected naturally, resulting in the flickering artifact.

Traditional VSR (or multi-image super-resolution) algorithms take multiple LR frames as inputs and output HR frames by taking into account subpixel motions between the neighboring LR frames [28, 4, 23, 26]. All deep learning based VSR methods follow similar steps and are composed of two steps: a motion estimation and compensation procedure followed by an upsampling process [16, 22, 1, 24, 34]. One problem with this two-step approach is that the results rely heavily on the accurate motion estimation. Another potential problem with this type of approach is that the HR output frame is produced by mixing the values from multiple motion compensated input LR frames through a convolutional neural networks (CNN), which can lead to a blurry output HR frame.

2575-7075/18 $31.00 © 2018 IEEE
DOI 10.1109/CVPR.2018.00340

In this paper, we propose a novel end-to-end deep neural network that is fundamentally different from the previous methods. Instead of explicitly computing and compensating for motion between input frames, the motion information is implicitly utilized to generate dynamic upsampling filters. With the generated upsampling filters, the HR frame is directly constructed by local filtering to the input center frame (Fig. 2). As we do not rely on explicit computation of motions and do not directly combine values from multiple frames, we can generate much sharper and temporally consistent HR videos.

Using a large number of training videos with a new data augmentation process, we achieve the state-of-the-art performance compared to the previous deep learning based VSR algorithms. Fig. 1 shows one example where our method produces much sharper frames with less flickering compared to one of the state-of-the-art methods, VSRnet [16].

2. Related Works

Single-Image Super-Resolution. Dong *et al*. [3] proposed SRCNN, which is one of the earliest works to apply deep learning to SISR. SRCNN is a relatively shallow network that consists of 3 convolutional layers. Kim *et al*. [17] later developed a deeper network with 20 convolutional layers called VDSR that employs residual learning. SRGAN proposed by Ledig *et al*. [21] and EnhanceNet proposed by Sajjadi *et al*. [30] produced plausible high-frequency details by using the structure of ResNet [9] with the generative adversarial network [6]. Also, Tai *et al*. [33] proposed DRRN, which beats previous SISR methods in terms of PSNR and SSIM using recursive residual blocks. Recently, Tong *et al*. [35] proposed SRDenseNet, which showed good performance by using DenseNet [10].

Video Super-Resolution. Non deep learning based traditional VSR methods modelled the VSR problem by putting the motion between HR frames, the blurring process, and the subsampling altogether into one framework and focused on solving for sharp frames using an optimization [28, 4, 23, 26]. Liu and Sun [23] proposed a Bayesian approach to estimate HR video sequences that also computes the motion fields and blur kernels simultaneously.

Recently, deep learning based VSR methods that exploit motion information in videos have been proposed. Huang *et al*. [11] proposed BRCN using recurrent neural networks to model long-term contextual information of temporal sequences. Specifically, they used bidirectional connection between video frames with three types of convolutions: the feedforward convolution for spatial dependency, the recurrent convolution for long-term temporal dependency, and the conditional convolution for long-term contextual information.

Liao *et al*. [22] proposed DECN, which reduces computational load for motion estimation by employing a non-iterative framework. They generated SR drafts by several hand-designed optical flow algorithms, and a deep network produced final results. Likewise, Kappeler *et al*. [16] proposed VSRnet, which compensates motions in input LR frames by using a hand-designed optical flow algorithm as a preprocessing before being fed to a pretrained deep SR network.

Caballero *et al*. [1] proposed VESPCN, an end-to-end deep network, which learns motions between input LR frames and improves HR frame reconstruction accuracy in real-time. VESPCN estimates the optical flow between input LR frames with a learned CNN to warp frames by a spatial transformer [13], and finally produces an HR frame through another deep network.

In Liu *et al*. [24], the method also learns and compensates the motion between input LR frames. But after the motion compensation, they adaptively use the motion information in various temporal radius by temporal adaptive neural network. The network is composed of several SR inference branches for each different temporal radius, and the final output is generated by aggregating the outputs of all the branches.

Tao *et al*. [34] used motion compensation transformer module from VESPCN [1] for the motion estimation, and proposed a subpixel motion compensation layer for simultaneous motion compensation and upsampling. For following SR network, an encoder-decoder style network with skip-connections [27] is used to accelerating training and ConvLSTM module [32] is used since video is sequential data.

Previous end-to-end CNN based VSR methods focus on explicit motion estimation and compensation to better reconstruct HR frames. Unlike the previous works, we do not have explicit motion estimation and compensation steps. Using the adaptive upsampling with dynamic filters that depend on input frames, we achieve a VSR performance that goes beyond those of existing works.

3. Method

The goal of the VSR is to estimate HR frames $\{\hat{Y}_t\}$ from given LR frames $\{X_t\}$. The LR frames $\{X_t\}$ are downsampled from the corresponding GT frames $\{Y_t\}$, where t denotes the time step. With the proposed VSR network G and the network parameters θ, the VSR problem is defined as

$$\hat{Y}_t = G_\theta(X_{t-N:t+N}), \tag{1}$$

where N is the temporal radius. An input tensor shape for G is $T \times H \times W \times C$, where $T = 2N + 1$, H and W are the height and the width of the input LR frames, and C is the number of color channels. Corresponding output tensor shape is $1 \times rH \times rW \times C$, where r is the upscaling factor.

As shown in Fig. 3, our network produces two outputs to

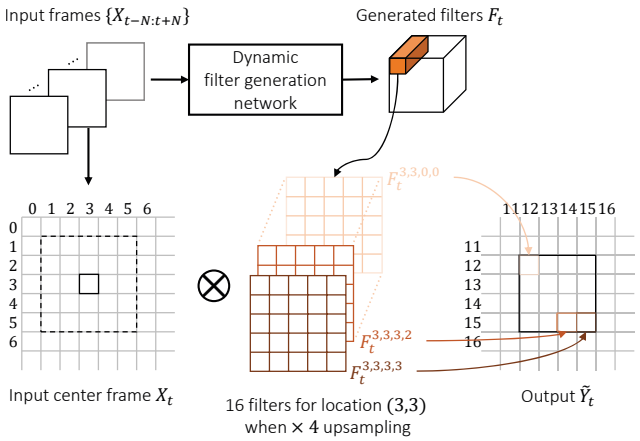


Figure 2. Dynamic upsampling without explicit motion compensation. The figure illustrates an example of upscaling a pixel at location $(3, 3)$ in the center input frame X_t by the upscaling factor $r = 4$. 16 generated filters from $F_t^{3,3,0,0}$ to $F_t^{3,3,3,3}$ are used to create 16 pixels in the region $(12, 12)$ to $(15, 15)$ of the filtered HR frame $\tilde{Y}_t$.

generate a final HR frame $\hat{Y}_t$ from a set of input LR frames $\{X_{t-N:t+N}\}$: the dynamic upsampling filters F_t and the residual R_t. The input center frame X_t is first locally filtered with the dynamic upsampling filters F_t and then the residual R_t is added to upsampled result $\tilde{Y}_t$ for the final output $\hat{Y}_t$.

3.1. Dynamic Upsampling Filters

The filter kernels for traditional bilinear or bicubic upsampling are basically fixed, with the only variation being the shift of the kernel according to the location of newly created pixel in an upsampled image. For the $\times 4$ upsamling, a set of 16 fixed kernels are used for those traditional upsampling process. They are fast but hardly restore sharp and textured regions.

Contrary to this, we propose to use dynamic upsampling filters inspired by the dynamic filter network (DFN) [15]. The upsampling filters are generated locally and dynamically depending on the spatio-temporal neighborhood of each pixel in LR frames.

The overview of the proposed dynamic upsampling procedure for VSR is shown in Fig. 2. First, a set of input LR frames $\{X_{t-N:t+N}\}$ (7 frames in our network: $N = 3$) is fed into the dynamic filter generation network. The trained network outputs a set of r^2HW upsampling filters F_t of a certain size (5×5 in our network), which will be used to generate new pixels in the filtered HR frame $\tilde{Y}_t$ [1]. Finally, each output HR pixel value is created by local filtering on an LR pixel in the input frame X_t with the corresponding filter $F_t^{y,x,v,u}$ as follows:

$$\tilde{Y}_t(yr+v, xr+u) = \sum_{j=-2}^{2}\sum_{i=-2}^{2} F_t^{y,x,v,u}(j+2, i+2) X_t(y+j, x+i), \quad (2)$$

where y and x are the coordinates in LR grid, and v and u are the coordinates in each $r \times r$ output block ($0 \le v, u \le r-1$). Note that this operation is similar to deconvolution (or transposed convolution), thus our network can be trained end-to-end as it allows back-propagation.

Our approach is essentially different from the previous deep learning based SR methods, where a deep neural network learns to reconstruct HR frames through a series of convolution in the feature space. Instead, we use the deep neural network to learn the best upsampling filters, which is then used to directly reconstruct HR frames from given LR frames. Conceptually, the dynamic filters are created depending on the pixel motions as the filters are generated by looking at spatio-temporal neighborhoods of pixels, enabling us to avoid explicit motion compensation.

3.2. Residual Learning

The result after applying the dynamic upsampling filters alone lacks sharpness as it is still a weighted sum of input pixels. There may be unrecoverable details through linear filtering. To address this, we additionally estimate a residual image to increase high frequency details. In [17], a residual was added to the bicubically upsampled baseline to produce a final output. Our approach is different from [17] as the residual image is made from multiple input frames rather than a single input frame, and we use the dynamically upsampled frame as a better baseline which is then added with the computed residual. By combining these complementary components, we are able to achieve spatial sharpness and temporal consistency in the resulting HR frames.

3.3. Network Design

As shown in Fig. 3, our filter and residual generation network are designed to share most of the weights thus we can reduce the overhead coming from producing the two different outputs. The shared part of our network is designed inspired by a dense block [10] and it is modified properly to our problem. Specifically, we replace 2D convolutional layers with 3D convolutional layers to learn spatio-temporal features from video data. A 3D convolutional layer is known to be more suitable than a 2D convolutional layer in human action recognition [14] and generic spatio-temporal feature extraction [36] on video data. Each part of the dense block is composed of batch normalization (BN) [12], ReLU [5], $1\times1\times1$ convolution, BN, ReLU, and $3\times3\times3$ convolution in order. As done in [10], each part

[1] We use the same upsampling kernel for all the color channels

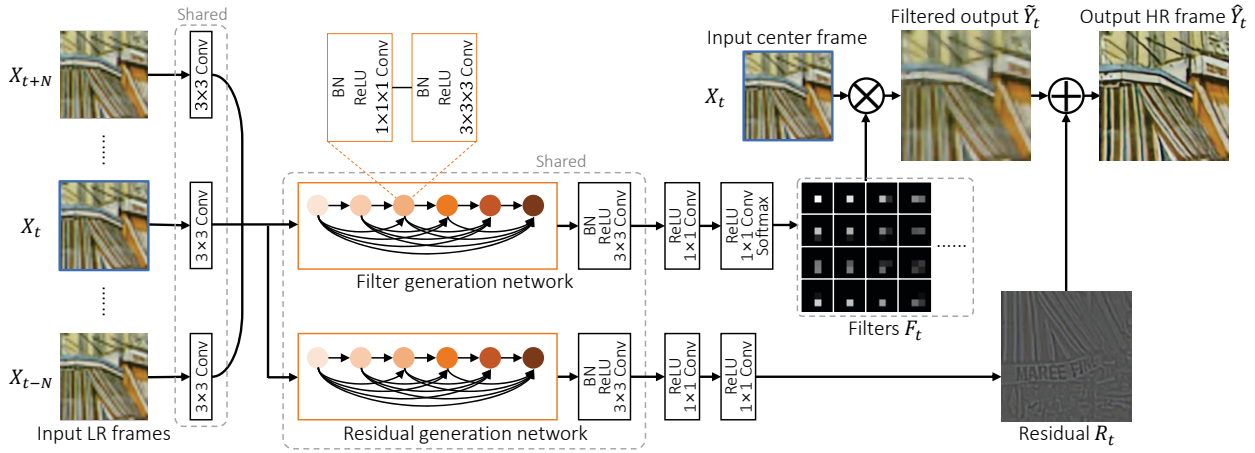


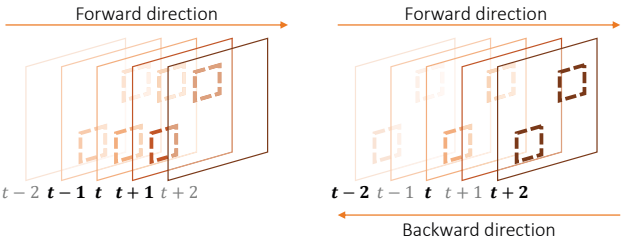
Figure 3. Network architecture. The weights of filter and residual generation networks are shared for efficiency. The dynamic upsampling filters F_t are generated adaptively to local motion, and residual R_t adds high-frequency details.

takes all preceding feature maps as input.

Each input LR frame is first processed by a shared 2D convolutional layer and concatenated along temporal axis. The resulting spatio-tempral feature maps go through our 3D dense block, and are then processed at separate branches that consist of several 2D convolutional layers to generate the two outputs. To produce the final output $\hat{Y}_t$, the filtered output $\tilde{Y}_t$ is added with the generated residual R_t.

3.4. Temporal Augmentation

In order to make the proposed network to fully understand various and complex real-world motions, we need corresponding training data. To create such training data, we apply a data augmentation in the temporal axis on top of the general data augmentation like random rotation and flipping. Here, we introduce the variable TA that determines sampling interval of the temporal augmentation. With $TA = 2$, for example, we will sample every other frames to simulate faster motion. We can also create a new video sample in the reverse order when we set the TA value as negative (Fig. 4). Using various sizes of TA (from -3 to 3 in our work), we can create training data with rich motion. Note that the VSR performance decreases in case of $|TA| > 3$ as the displacement of an object gets too large.



(a) $TA = 1$. (b) $TA = \{-2, 2\}$.

Figure 4. Sampling data from a video with the temporal radius $N = 1$. Training data with faster or reverse motion can be sampled with the temporal augmentation.

4. Implementation

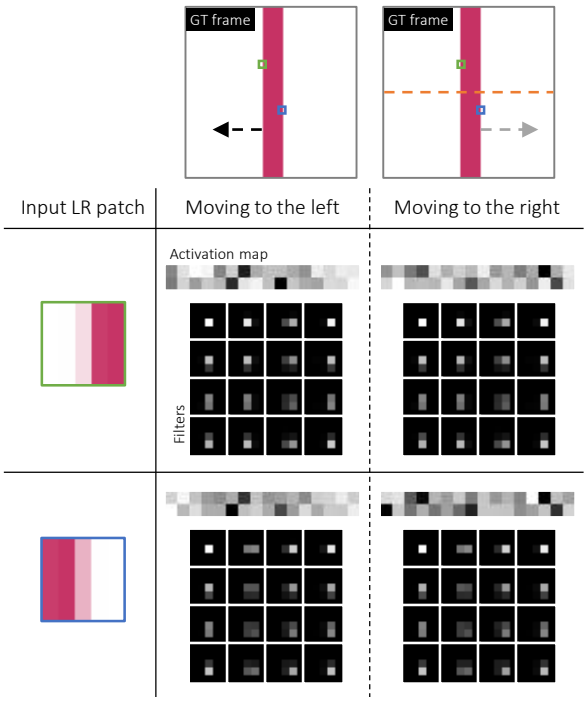
Datasets. One of the most important elements of deep learning is the quantity and the quality of training data. To achieve good generalization, videos in the training dataset must contain various and complex real-world motions. But for the VSR task, a dataset like ImageNet [29] does not exist. Therefore, we collected a total of 351 videos from the Internet with various contents including wildlife, activity, and landscape. The collected videos also include various textures and motions. We sample $160,000$ ground truth training data with the spatial resolution of 144×144 by selecting areas with sufficient amount of motion. For the validation set, we use 4 videos, *coastguard*, *foreman*, *garden*, and *husky* from the Derf's collection[2], which we name as *Val4*. For the test set, *Vid4* from [23] is used to compare with other methods.

Training. To obtain LR inputs, the ground truth training data are first smoothed with a Gaussian filter and then subsampled with respect to the upscaling factor r. The spatial resolution of the input patch is fixed as 32×32. We set the size of mini-batch to 16, and all variables are initialized by using the method in [8].

To train our network G_θ, we use the Huber loss as the cost function for stable convergence:

$$\mathcal{L} = \mathcal{H}(\hat{Y}_t, Y_t) = \begin{cases} \frac{1}{2}||\hat{Y}_t - Y_t||_2^2 & ||\hat{Y}_t - Y_t||_1 \leq \delta, \\ \delta||\hat{Y}_t - Y_t||_1 - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{3}$$

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



(a) The results of synthetic motion test.

(b) $x-t$ graph for the dotted orange scanline.

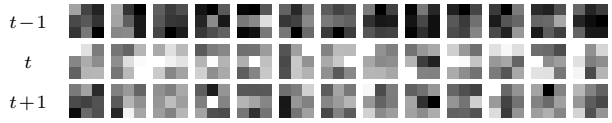
Figure 5. Synthetic motion experiments for $\times 4$ VSR. Two videos with the same object but different motion are synthesized to verify that our method can implicitly handle motion for VSR task. (a) The results indicate that our network exploits the temporal information properly. See text for details. (b) Our method can maintain temporal consistency. Best viewed in zoom.

and set the threshold $\delta = 0.01$. We use Adam optimizer [18] and initially set learning rate to 0.001 and multiply by 0.1 after every 10 epochs. Initial accuracy of our approach is quite high due to the direct upsample filtering and we get a good starting point for the training. Combined with the residual learning, the convergence speed is very fast and we can complete the training in only 22 epochs. In the test phase, we zero padded inputs in the temporal axis to prevent the number of frames from decreasing.

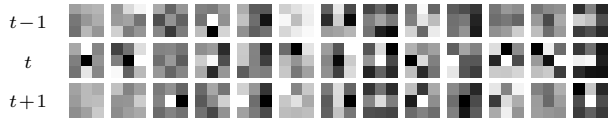
5. Experimental Results

5.1. Visualization of Learned Motion

Synthetic Motion Test. In order to verify that the proposed method exploits temporal information of a given video properly without the explicit motion compensation, videos with a vertical bar moving to the left and to the right



(a) Learned filters from the first $3\times3\times3$ convolutional layer.



(b) Learned filters from the last $3\times3\times3$ convolutional layer.

Figure 6. Example learned filters of our filter and residual generation network. They can bring information from front and rear as well as center of an input tensor since the weights are evenly high at all time steps. Each 3×3 filter represents spatial weight and total 3 filters to temporal axis are shown in each column. Each $3\times3\times3$ filter is normalized to better visualization.

by 1 pixel per frame in GT frames are synthesized. For the same input patch, the activation maps of both videos should be different since each bar has different motions. But the HR frame to be generated is the same, so the upsampling filters should be same. In Fig. 5(a), we plot the activation maps of the first $3\times3\times3$ convolutional layer and the generated dynamic filters for the different two regions. As expected, our learned network shows different activations for different motions in the same input patches and the weights of the generated dynamic filters are almost the same.

We also plot the transition of the dotted orange horizontal scanline over time in Fig. 5(b) to see whether the temporal consistency is well preserved. Since VDSR [17] does not consider temporal information, it shows a bumpy result. The reconstructed bar of VSRnet [16] also shows zig-zag artifacts, and both will cause severe flickering artifact when playing video. In comparison, our result looks sharper and shows the ability to maintain the temporal consistency, and closest to the ground truth. We recommend the readers to see our supplementary video for better comparisons.[3]

Learned Filters. Another way to check our method can learn and use temporal information is seeing learned filters. To do this, learned filters in $3\times3\times3$ convolutional layers of our method should fetch some values from front and rear as well as center of inputs to temporal axis. We can see that our network can evenly fetch some values from front and rear as well as center of inputs to temporal axis with various structures since the filters have higher weights at time steps $t-1$ and $t+1$ as well as t as shown in Fig. 6. This shows our method is capable to learn and use temporal information between input frames implicitly for generating

[3]Codes and the video are available on our project page https://github.com/yhjo09/VSR-DUF

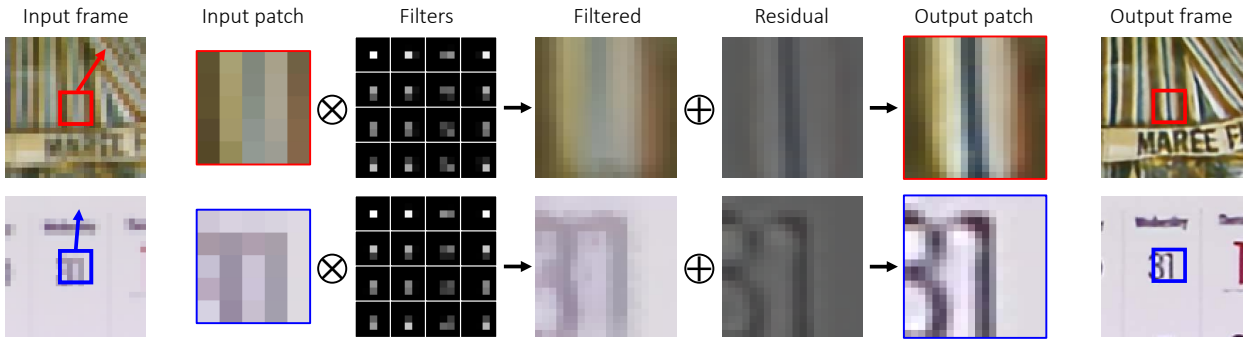


Figure 7. Examples of upsampling process in detail. The direction of the colored arrows indicates the direction of local motion. Two sets of 16 upsampling filters are for the center pixel of each input patch, and they are different since each input has different texture and local motion.

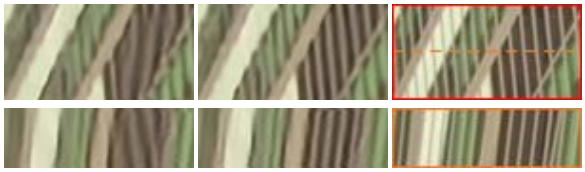


Figure 8. With our algorithm, increasing the number of layers provide better result. The scene from [34].

accurate output frame.

Dynamic Upsampling Filters. The generated upsampling filters should have different weights for different regions to adaptively handle local motion. Fig. 7 shows examples of the proposed upsampling process for two different regions of the scene *Calendar* in details, and the filter weights for each newly created pixel positions are different. For two input patches with different texture and motion direction, our method adaptively generates upsampling filters for accurate output reconstruction.

5.2. Comparisons with Other Methods

With our basic network consisting of 16 layers (Ours-16L), networks with 28 layers (Ours-28L) and 52 layers (Ours-52L) are also tested. As with most super-resolution methods, scenes with thin lines that are close together are quite challenging as shown in Fig. 8. With our algorithm, increasing the number of layers provide better results for this type of challenging scenes.

Quantitative Evaluation. Quantitative comparison with other state-of-the-art VSR methods is shown in Table 1. The results show the tendency that the network with increased depth performs better, and the PSNR value of Ours-28L is increased by 0.18dB from Ours-16L with 0.2M additional parameters. Our approach works well even when stacking up to 52 layers and the PSNR value for *Vid4* is improved to 27.34dB, which is 0.53dB higher than that of Ours-16L. Even with Ours-16L, we outperforms all other methods by a large margin in terms of PSNR and SSIM for all upscale factors. For example, the PSNR of Ours-16L is 0.8dB higher than the second highest result [34] ($r = 4$).

Qualitative Comparisons. Some qualitative examples are shown in Fig. 9. Fine details and textures are better reconstructed using our method. In Fig. 10, we also compare a result from [34] with the results using our network with different depth. We outperform the previous work and can also observe the increase in the performance with more depth. More qualitative comparisons with other state-of-the-art VSR methods on *Vid4* is shown in Fig. 11. Our results show sharper outputs with more smooth temporal transition compared to other works. Sharp and smooth edges in the $x - t$ images indicate that the video has much less flickering.

6. Conclusion

In this paper, we introduce a new deep learning based framework for VSR that learns to output dynamic upsampling filters and the residual simultaneously. We achieve the state-of-the-art performance with our new framework and recover sharp HR frames and also maintain the temporal consistency. Through experiments, we have shown that our deep network can implicitly handle the motion without explicit motion estimation and compensation.

It takes about 2 days to train our network and the runtimes to generate a single 1920×1080 output frame from 7 input frames size of 480×270 are $0.4030s$, $0.8382s$, and $2.8189s$ for Ours-16L, Ours-28L, and Ours-52L respectively using NVidia GeForce GTX 1080 Ti. About a half of the runtime is spent on local filtering in our inference process. In the future, we would like to focus on accelerating our method to achieve a real-time performance. We would also like to extend our work to increase the temporal resolution in addition to the spatial resolution, for example, creating a 60fps UHD video from a 30fps SD video.

Upscale	Metric	Bicubic	VSRnet[16]	VESPCN[1]	Tao *et al*. [34]	Liu *et al*. [24]	Ours-16L	Ours-28L	Ours-52L
×2	PSNR	28.43	31.30	-	-	-	**33.73**	-	-
	SSIM	0.8676	0.9278	-	-	-	**0.9554**	-	-
×3	PSNR	25.28	26.79	27.25	-	-	**28.90**	-	-
	SSIM	0.7329	0.8098	0.8447	-	-	**0.8898**	-	-
×4	PSNR	23.79	24.84	25.35	26.01	25.24	**26.81**	**26.99**	**27.34**
	SSIM	0.6332	0.7049	0.7557	0.7744	-	**0.8145**	**0.8215**	**0.8327**

Table 1. Quantitative comparison with other state-of-the-art VSR methods on *Vid4*. Mean PSNR and SSIM are measured excluding spatial border pixels (8 pixels), the first two, and the last two frames as done in [16]. The results for [34] is re-computed from the provided output images in the same way. All other values are taken from the original papers. The results of our method are shown in bold, and show the best performance for all the upscale factors.

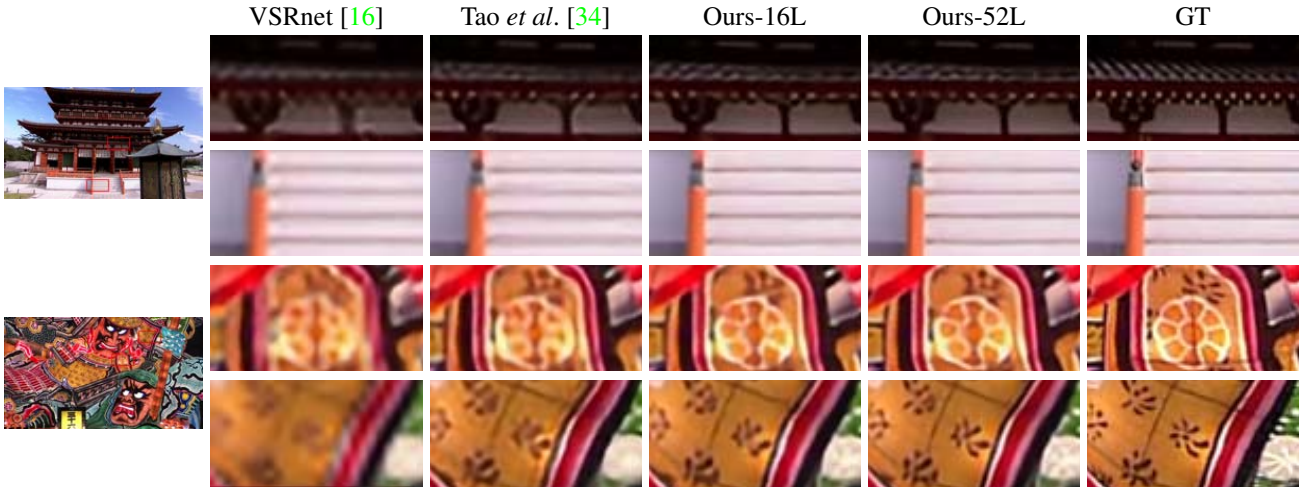


Figure 9. Qualitative comparison on videos from [34].

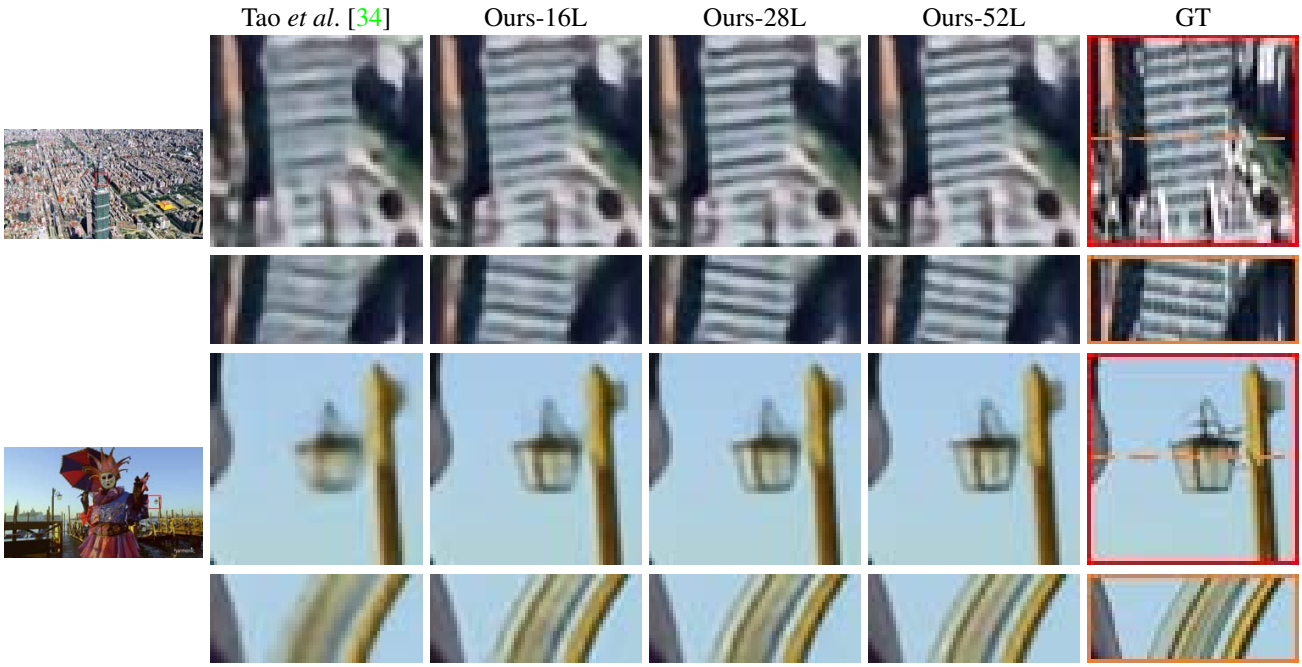


Figure 10. Qualitative comparison of our methods on videos from [34].

Figure 11. Qualitative comparison on *Vid4*.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2016R1A2B4014610).

References

[1] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In

CVPR, 2017. 1, 2, 7, 8

[2] H. Demirel and G. Anbarjafari. Discrete wavelet transform-based satellite image resolution enhancement. *IEEE Transactions on Geoscience and Remote Sensing*, 49(6):1997–2004, 2011. 1

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199, 2014. 1, 2

[4] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004. 1, 2

[5] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, volume 15, pages 315–323, 2011. 3

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680. 2014. 2

[7] H. Greenspan. Super-resolution in medical imaging. *The Computer Journal*, 52(1):43–63, 2009. 1

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015. 4

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2

[10] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2, 3

[11] Y. Huang, W. Wang, and L. Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In *NIPS*, pages 235–243, 2015. 2

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

[13] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015. 2

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 35(1):221–231, 2013. 3

[15] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *NIPS*, pages 667–675, 2016. 3

[16] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016. 1, 2, 5, 7

[17] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016. 1, 2, 3, 5

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[20] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 1

[21] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 1, 2

[22] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia. Video super-resolution via deep draft-ensemble learning. In *ICCV*, pages 531–539, 2015. 1, 2

[23] C. Liu and D. Sun. On bayesian adaptive video super resolution. *TPAMI*, 36(2):346–360, 2014. 1, 2, 4

[24] D. Liu, Z. Wang, Y. Fan, X. Liu, Z. Wang, S. Chang, and T. Huang. Robust video super-resolution with learned temporal dynamics. In *ICCV*, 2017. 1, 2, 7

[25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 1

[26] Z. Ma, R. Liao, X. Tao, L. Xu, J. Jia, and E. Wu. Handling motion blur in multi-frame super-resolution. In *CVPR*, pages 5224–5232, 2015. 1, 2

[27] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, pages 2802–2810, 2016. 2

[28] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *IEEE Signal Processing Magazine*, 20(3):21–36, 2003. 1, 2

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 4

[30] M. S. M. Sajjadi, B. Scholkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, 2017. 1, 2

[31] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016. 1

[32] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810, 2015. 2

[33] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017. 1, 2

[34] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia. Detail-revealing deep video super-resolution. In *ICCV*, 2017. 1, 2, 6, 7, 8

[35] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *ICCV*, 2017. 1, 2

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015. 3

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 1

[38] L. Zhang, H. Zhang, H. Shen, and P. Li. A super-resolution reconstruction algorithm for surveillance images. *Signal Processing*, 90(3):848–859, 2010. 1