Time Consistency for Video Super-Resolution

Chaitanya Ravuri
MIT
Cambridge, MA
cravuri@mit.edu

Abstract

Video super-resolution is an ill-posed task, in which a low resolution input video is converted into a high resolution output video. Previous approaches have achieved great results in single-image super-resolution, but applying these methods directly to video results in artifacts due to temporal inconsistency. In this work, we propose a convolutional neural network that uses information from multiple input frames, in addition to the previous output frame to achieve high quality, temporally consistent results. Our evaluations and comparisons to the more naive frame-by-frame method show that our proposed framework significantly outperforms the baseline method.

1. Introduction

The task of video super-resolution is to reconstruct a high resolution (HR) video from a low resolution (LR) one. This problem is highly useful, especially in the domain of streaming. If a streaming site like Youtube or Netflix applies super-resolution to videos locally, they only have to send low resolution videos from their servers to the user, greatly decreasing their necessary bandwidth and storage space.

The problem of video super-resolution is difficult because, in contrast to single image super-resolution, the reconstructed HR video must be coherent both spatially and temporally. If the frames are just processed independently, while individual HR frames may be well constructed, consecutive frames are unlikely to be reconstructed in the same way, resulting in unpleasant artifacts.

Additionally, simply applying image super-resolution to each frame independently does not leverage the additional information inherent to the video format. Barring sudden cuts between scenes, adjacent frames in a video are generally quite similar to each other. Thus, consecutive frames would contain information about the current one, which may help fill in the gaps of the HR reconstruction, and improve overall image quality.

(a) LR frame (b) Our result (c) True HR frame

Figure 1. Side-by-side comparison of the low resolution input, our reconstructed output, and the high resolution ground truth.

In this work, we propose a framework that uses these two observations to generate high-quality and temporally consistent HR videos. Rather than applying image super-resolution to each frame independently, our input will consist of a sequence of LR frames, that will all be used to predict a single HR frame. To make sure that consecutive frames of the output are temporally consistent with each other, we will also pass in the HR frame that was previously generated by the network.

2. Related Work

Prior approaches to super-resolution have included a variety of methods, such as simple upscaling through interpolation [2], or more complex algorithms that use coupled dictionaries to learn a mapping between LR and HR images [6, 7]. With progress in deep learning, approaches like the one used by Dong *et al.* [1] have become the new state-of-the-art in single image super-resolution, using convolutional neural network architectures to improve reconstructions.

Video super-resolution has benefited from these techniques as well. Our method was heavily inspired by previous deep-learning based approaches, many of which used information from multiple frames to generate a single HR output frame. For example, Kappeler *et al.* [3] uses three consecutive frames to construct a HR version of the middle frame.

We were also inspired by the method used by Sajjadi *et*

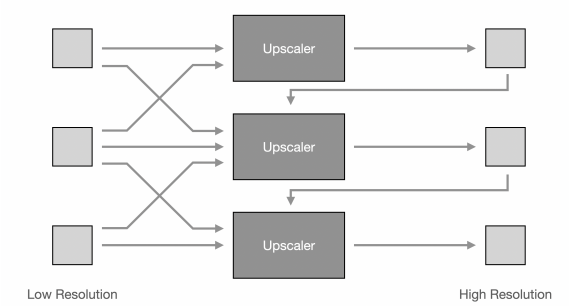

Figure 2. Overview of proposed framework. The $2n+1$ adjacent LR frames, and the previous HR frame are passed in to the network to generate the current HR frame.

al. [5], who use a recurrent architecture that feeds in the previous HR frame estimate to generate the current frame.

Our goal is to combine these two methods together in the hope that the improved information from multiple frames and the temporal consistency from the recurrent architecture will give more accurate and more qualitatively pleasing reconstructions.

3. Approach

Our proposed framework will take in an LR frame (I_t^{LR}), its neighboring frames $(I_{t-n}^{LR}, \ldots, I_{t+n}^{LR})$, and the previously generated HR image (I_{t-1}^{est}) to generate a new HR image (I_t^{est}). This pipeline is shown in Figure 2.

Each LR image has dimensions $W \times H \times 3$. There are a total of $(2n+1)$ LR frames in the input, and we can concatenate them all along the channel dimension to get an input of size $W \times H \times (6n+3)$.

The HR image has dimensions $sW \times sH \times 3$, where s is the scaling factor. This can be mapped to LR space with the space-to-depth transformation:

$$SD : \mathbb{R}^{sW \times sH \times 3} \rightarrow \mathbb{R}^{W \times H \times 3s^2}$$

This transformation takes each pixel in an $s \times s$ block of the original image, and places them in separate channels. As we are now in LR space, we can concatenate $SD(I_{t-1}^{est})$ with the LR frames to get the full input, which now has dimensions $W \times H \times (3s^2+6n+3)$. Our network will then take this input image and generate I_t^{est}, the HR estimation for the current frame. In our experiments, we set $s = 4$ and $n = 1$, so we used the three adjacent images to generate an image that is 4 times as large.

Our network architecture is similar to the architecture used by Sajjadi *et al.* [4], a Super-Resolution Residual Network. An overview of this architecture is shown in Figure 3. The network first has a convolutional layer to go from $(3s^2 + 6n + 3)$ channels to 64. Then, the network has a series of residual blocks (in our case 16 blocks), that compute

Output size	Layer
$w \times h \times c$	Input I_{LR}
$w \times h \times 64$	Conv, ReLU
	Residual: Conv, ReLU, Conv
	...
$2w \times 2h \times 64$	2x nearest neighbor upsampling Conv, ReLU
$4w \times 4h \times 64$	2x nearest neighbor upsampling Conv, ReLU
	Conv, ReLU
$4w \times 4h \times c$	Conv
	Residual image I_{res}
	Output $I_{\text{est}} = I_{\text{bicubic}} + I_{\text{res}}$

Figure 3. Upscaler. The architecture for a super-resolution network from [4], consisting of a series of 16 residual blocks, then a number of upsampling convolutional layers.

a feature map of the input images. Finally, the network has two nearest neighbor convolutional upsampling layers to go from a $W \times H$ image to a $4W \times 4H$ image.

As a loss function, we use pixel-wise MSE to train our network, taking the sum of the squares of the differences between each pixel in the estimated HR image and the actual HR image:

$$\mathcal{L} = \|I_t^{est} - I_t^{HR}\|_2^2$$

To train our network, we use the RealVSR dataset, which consists of 50 high resolution and corresponding low resolution videos in RGB format. We split this dataset into 40 videos to be used for training, and 10 videos to be used for testing. Each video consists of around 50 frames, so at a frame rate of 24 fps, each video is 2-3 seconds in length.

We used Google Colab's free GPUs to train our network. Because of the limited compute, we chose not to use the full 1080p video resolution in training our network. Instead, we first downsampled all our input frames so they were 240×240 pixels in size, and then downsampled them further to 60×60 for the low resolution videos.

We trained for 100 epochs, or a total of 250,000 iterations. The results of our training are described in the following section.

4. Results

To evaluate our system, we compare our model with three separate baselines that use the same super-resolution network but take in different inputs. Our first baseline is the naive method, which computes super-resolution on each LR frame independently to then generate an HR frame:

$$\text{Naive} : I_t \rightarrow I_t^{est}$$

We made two improvements to this naive method, and so for our other two baselines, we wanted to test these two

Method	PSNR (dB)
Naive	23.06
Adjacent	24.12
Recurrent	23.31
Both	**24.28**

Table 1. Mean PSNR of various models on the RealVSR dataset. The naive model performs the worst, and our combined method performs the best.

improvements individually, to see if they provide value on their own. Our first improvement was to use the adjacent LR frames to help construct the current HR frame. We'll call this method the adjacent method:

$$\text{Adjacent} : I_{t-n}...I_{t+n} \rightarrow I_t^{est}$$

Our second improvement was using the previously reconstructed HR frame, in addition to the current LR frame to generate the current HR frame. We'll call this method the recurrent method:

$$\text{Recurrent} : I_t; I_{t-1}^{est} \rightarrow I_t^{est}$$

We test these baselines against our full model, which combines both of these methods, using both the adjacent LR frames and the previous HR frame to generate the current HR frame:

$$\text{Both} : I_{t-n}...I_{t+n}; I_{t-1}^{est} \rightarrow I_t^{est}$$

The results of our experiments are shown in Table 1. We used the same dataset to train and evaluate all four models. We chose to use the Peak Signal-to-Noise Ratio (PSNR) as our metric of evaluating the accuracy of a reconstruction. The calculation for PSNR is as follows:

$$PSNR = 10 \cdot \log_{10}\left(\frac{\max(I)^2}{MSE}\right)$$

Here, $\max(I)$ is the maximum value of any pixel in the image (which in our case is 1.0). We use the same MSE calculation as our loss function, so PSNR captures the average loss very well. From Table 1, we can see that the naive method had the lowest PSNR, and our combined method had the highest. Greater PSNR means a more accurate reconstruction of the image, so our method performs the best out of the four we tested.

An interesting thing to note is that although both the adjacent method and the recurrent method improve upon the naive method, the adjacent method offers the greater improvement, resulting in a PSNR of 24.12 compared to the recurrent method's PSNR of 23.31. PSNR measures the accuracy of reconstruction for each frame, but it does not capture how temporally consistent a video is across frames.

Through qualitative observation, the recurrent method gives a more temporally consistent video, with less flickering and artifacts, than the adjacent method. This makes sense, as the recurrency was meant to make the output video more temporally consistent, while the adjacent method was meant to improve the reconstruction of individual frames. Of course, using both methods together produces the best results both quantitatively and qualitatively.

5. Conclusion and Future Work

We have proposed an architecture that performs significantly better than the naive method by both quantitative and qualitative measures. However, there are still limitations that our approach has not addressed.

Quick cuts in a video, or very large amounts of motion will cause our system to fail. As our system is now, the filters in the super-resolution network observe the same section of the image for each of the channels in the input. We rely on the fact that there is usually very little motion between frames to make use of adjacent frames, but if there is a large amount of motion, the same locations in consecutive frames will no longer correspond, and so any extra information they provide will be useless.

To improve our system in the future, we should take any motion in the scene into account, and correct adjacent frames, essentially inverting that motion so the information they provide is still useful.

Additionally, the way we measure temporal consistency is not particularly scientific. We use visual comparison to see which videos have the least amount of artifacts, which is qualitative and subjective. Having a metric that can quantitatively measure how smooth a video is can help in both evaluation and training, as using that metric as a loss function could result in an increase in quality.

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. Dec. 2014. 1

[2] Claude E. Duchon. Lanczos filtering in one and two dimensions, Aug 1979. 1

[3] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K. Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016. 1

[4] Mehdi S M Sajjadi, Bernhard Schölkopf, and Michael Hirsch. EnhanceNet: Single image super-resolution through automated texture synthesis. Dec. 2016. 2

[5] Mehdi S. M. Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[6] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas Huang. Coupled dictionary training for image

super-resolution. *IEEE Transactions on Image Processing*, 21(8):3467–3478, 2012. 1

[7] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010. 1