# Learning Spatio-Temporal Downsampling for Effective Video Upscaling Supplementary Material

Xiaoyu Xiang[1], Yapeng Tian[2], Vijay Rengaranjan[1], Lucas D. Young[1], Bo Zhu[1], and Rakesh Ranjan[1]

[1] Meta Reality Labs, [2] University of Texas at Dallas
{xiangxiaoyu,apvijay,bozhufrl,rakeshr}@fb.com
{tianyapeng92,lucasyoung482}@gmail.com

In this appendix, we first discuss the difference between video frame interpolation and temporal upscaling in Sec. A. Then, we provide more module design details in Sec. B. Besides, we provide the background knowledge of space-time Fourier analysis in Sec. C to help the readers understand the relevant part in the main paper. Experimental details are described in Sec. D. Sec. E includes more comparisons with the state-of-the-art (SOTA) methods and the results of integrating other network structures in the Space-Time Anti-Aliasing (STAA) framework. Lastly, we show the training setting and more results of the extensive applications in Sec. F.

## A Difference between Video Frame Interpolation and Temporal Upscaling

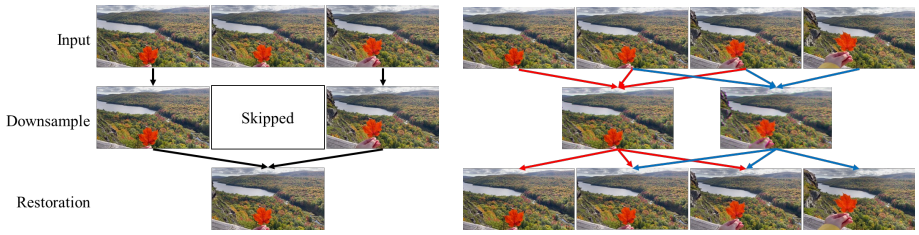
In this section, we explain the difference between the video frame interpolation (VFI) task and the temporal upscaling proposed in this paper.

Fig. S1 shows the illustration of these two different settings: given an input sequence of frames $I_1, I_2, I_3, I_4, \ldots$, VFI adopts nearest-neighbor downsampling to acquire the downsampled sequence $I_1, I_2, \ldots$, as shown in (a). The acquired sequence explicitly corresponds to the timestep in the original input. In the restoration step, it needs to synthesize the missing frames from neighbors in the downsampled sequence: *e.g.*

$$I_1, I_3 \rightarrow \widetilde{I}_2. \tag{1}$$

Thus, in the final sequence with a total length of $2n-1$, $n-1$ frames are synthesized and $n$ frames are directly copied from the original input. So VFI methods cannot perfectly reconstruct a sequence with an even number of frames.

Temporal upscaling is more like spatial-upscaling: given an input sequence $I_1, I_2, I_3, I_4, \ldots$, the anti-aliasing filter blends the information from nearby frames, thus each frame of the downsampled sequence should contain information from multiple frames of the original input. Thus, it does not explicitly correspond to the timestep in the original input. So we annotate the downsampled frames as $I_{123}, I_{234}, \ldots$. Since this downsampled representation contains the motion information at each timestep of the original sequence, we should be able to reconstruct




(a) Video Frame Interpolation (b) Temporal Upscaling

Fig. S1: *Illustration of the difference between video frame interpolation and temporal upscaling.* In VFI settings, each downsampled frame corresponds to only a single frame of the original input. While for temporal upscaling, we allow the downsampled frame to assimilate information from the adjacent frames of the original input. Thus, we could reconstruct a full sequence from it.

them with proper design. For the restoration output, all frames in the sequence are synthesized:

$$I_{123}, I_{234} \rightarrow \widetilde{I}_1, \widetilde{I}_2, \widetilde{I}_3, \widetilde{I}_4. \tag{2}$$

The temporal upscaling method can reconstruct a sequence with either an even or odd number of frames.

Due to the above differences, it is unfair to directly compare the output between these two methods. So in this paper, we only compare the frames that are synthesized by both methods (*e.g.* in the above example, $\widetilde{I}_2, \widetilde{I}_4, \ldots$) to measure the reconstruction capability of the network. For the full sequence, we compare the temporal profile to evaluate the reconstructed motion patterns.

## B Module Design Details

In this section, we provide more details about the two modules of the downsampler: the space-time anti-aliasing filter and the differentiable quantization layer, and the two modules of the upsampler: the deformable temporal modeling (DTM) and the residual dense block with 3D convolutions.

### B.1 Space-Time Anti-Aliasing Downsampler

Fig. S2 illustrates the two operations in the downsampler: filtering with the learned 3D space-time filter and downsampling with stride. In our implementation, the above two steps are simplified into a convolution: the convolution kernel size is the window size in space/time; the convolution weights are learnable with custom constraints; the downsampling in each dimension is controlled by stride.

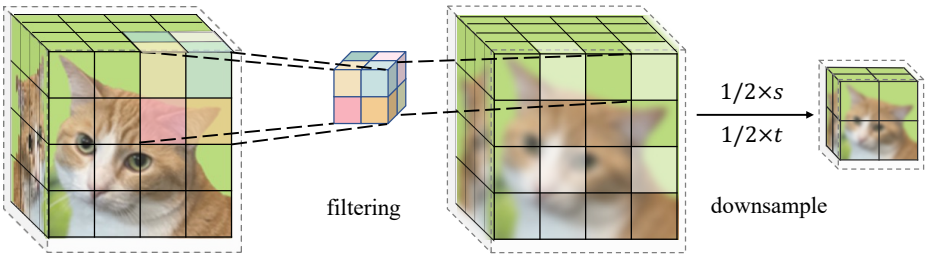



Fig. S2: Illustration of our learned space-time anti-aliasing downsampler.

### B.2 Differentiable Quantization Layer

The direct output of our downsampler is a floating-point tensor, while in practical applications, images are usually encoded as 8-bit RGB (uint8). To make our downsampled frames compatible with popular image data storage and transmission pipelines, we propose a quantization layer and include it in our end-to-end training process.

Directly casting the tensor type to uint8 does not work: such operations are not differentiable and thus cannot be used to train our network in an end-to-end manner. Consequently, if we directly send a uint8 input to the upsampler, there is going to be a performance drop due to the precision gap between the float and the uint8. So we split the quantization for tensors into two steps:

(1) **Clipping**: limit the values to a certain range (*e.g.* [0, 255] for uint8) to avoid blown-out colors. There is a serious problem: the derivative out of the $[min, max]$ value is masked to be 0, which makes the training at early iterations unstable, as reported in [4]. To solve this problem, we modify the clipping function $Q(\cdot)$ into:

$$\begin{aligned} \hat{x} &= x + \mathrm{ReLU}(min - x), \\ Q(x) &= \hat{x} - \mathrm{ReLU}(\hat{x} - max), \end{aligned} \tag{3}$$

where $x$ is our input value, $[min, max]$ denotes the clipping range, and $Q(x)$ is the clipped output. The derivative of the clipping function becomes:

$$\frac{dQ(x)}{dx} = \begin{cases} 2, & \text{if } x \leq min \\ 1, & \text{if } x \in (min, max) \\ 2, & \text{if } x \geq max \end{cases} \tag{4}$$

This modification does not influence clipping results but makes the derivative non-zero everywhere.

(2) **Round**: round the values to the closest integer. The gradient of this function is zero almost everywhere, which is bad for optimization. To avoid this issue, we override the gradient map to 1 during backward propagation.

As a result, the quantization operation's derivative is non-zero everywhere, thus enabling end-to-end training. Fig. S3 shows the training and validation loss of our framework. It shows our proposed method can be trained stably from scratch.


Fig. S3: *Plot of our training progress.* With the aid of the differentiable quantization layer, our network's training is stable from scratch.

### B.3 Deformable Temporal Modeling (DTM)

Here we describe the detailed structure of the deformable temporal module: as illustrated in the main paper, the temporal correspondences are propagated in both forward and backward directions. At each step $i$, it takes a current feature $f_i$ as input and refine it with the last step's output $f'_{i-1}$ with deformable alignment function $T(\cdot)$:

$$f'_i = T(f_i, f'_{i-1}). \tag{5}$$

The deformable alignment function $T(\cdot)$ is basically a deformable convolution, which takes an input feature map $f'_{i-1}$ and an offset map $\Delta p_i$:

$$T(f_i, f'_{i-1}) = DConv(f'_{i-1}, \Delta p_i), \tag{6}$$

where the offset $\Delta p_i$ is estimated from the refined feature map from the last step and the current step with convolutional function $g(\cdot)$:

$$\Delta p_i = g(f'_{i-1}, f_i). \tag{7}$$

In this way, the previous feature maps are aligned with the current one, which enhances our network's capability of handling motions. Then we aggregate the current feature $f_i$ and the aligned feature $f'_i$ to get the refined feature: $r(f_i) = c(f_i, f'_i)$ with the ConvLSTM $c$. After bi-directional propagation, for each timestep $i$, there will be two refined features from each direction. The aggregated output can be acquired by a blending function:

$$DTM(f_i) = w_f * r_f(f_i) + w_b r_b(f_i), \tag{8}$$

where $w_f, w_b$ are conv$1 \times 1$ kernels and $*$ denotes the convolutional operator. In this way, we get a well-aligned sequence as output.

### B.4 Residual Dense Block with 3D Convolutions

As the major building block of the reconstruction trunk in our upsampler, the 3D convolutions are organized exactly the same way as the residual dense block in [12, 16] while replacing the 2D convolutions with 3D, as shown in Fig. S4. The skip-connection of the residual block is with a residual scaling parameter $\beta$; in the dense block, features of different hierarchical levels are concatenated.

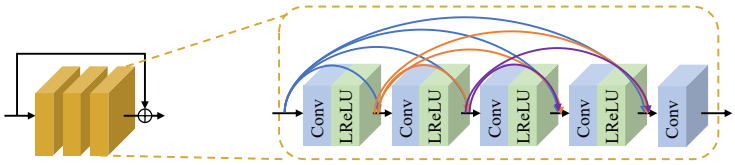



Fig. S4: Illustration of the residual dense block with 3D convolution.

## C Background of Space-Time Fourier Analysis

Given a video signal $f(x, y, t)$, we can analyze its space-time frequency characteristics through Fourier transform ($\mathcal{F}(\cdot)$):

$$F(\Omega_x, \Omega_y, \Omega_t) = \mathcal{F}_{x,y,t}(f(x, y, t)), \tag{9}$$

where $\Omega$ denotes the frequency for each dimension, and $F$ is the Fourier transform of $f$. For the uniform-velocity case as discussed in our main paper, we can denote the constant velocities as $v_x$ and $v_y$ for each direction. Thus, the object signal is translated through time by:

$$f(x, y, t) = g(x - v_x t, y - v_y t), \tag{10}$$

where $g(x, y)$ is the signal of the 2D object. We denote its Fourier transform as $G$. Correspondingly, the Fourier transform in Eq. 9 becomes:

$$\begin{aligned} F(\Omega_x, \Omega_y, \Omega_t) &= \mathcal{F}_{x,y,t}[g(x - v_x t, y - v_y t)] \\ &= G(\Omega_x \Omega_y) \int e^{-2\pi i t(\Omega_x v_x + \Omega_y v_y + \Omega_t)} dt \\ &= G(\Omega_x \Omega_y)\delta(\Omega_x v_x + \Omega_y v_y + \Omega_t). \end{aligned} \tag{11}$$

This explains why moving the 2D object would result in a shearing in the frequency domain: theoretically, all the non-zero frequency components should lie on the plane $\Omega_x v_x + \Omega_y v_y + \Omega_t$ in the 3D Fourier space, which reflects the coupling of space and time dimensions. However, our digital image is discrete – our simulated temporal profile has jigsaw-like boundaries between the moving object and the background due to the motion being at the pixel level. As a result, the transformed figure has two small sub-bands, as shown in the main paper. For visualization purposes, our analysis in the main paper is on the $xt$ 2D signals, thus Eq. 11 can be written as:

$$F(\Omega_x, \Omega_t) = G(\Omega_x)\delta(\Omega_x v_x + \Omega_t), \tag{12}$$

which shows the spectrum is limited by the delta function to a single line $\Omega_x v_x + \Omega_t = 0$. The velocity $v_x$ decides the slope of this line. For the $xt$ 2D signal, convolving with a filter $h(\cdot)$ equals to a multiplication with $H(\cdot)$ in the Fourier domain:

$$H_F(\Omega_x, \Omega_t) = G(\Omega_x)\delta(\Omega_x v_x + \Omega_t)H(\Omega_x, \Omega_t). \tag{13}$$

This explains the effect of different filters: Gaussian filter's Fourier transform is still a Gaussian function, which gradually attenuates the spatio-temporal high frequency; box filter (motion blur)'s Fourier transform is a sinc function, which attenuates certain frequency components.

## D Experimental Details

**Datasets.** Vimeo-90k dataset [15] is adopted to train the proposed framework. It consists of more than 60,000 training video sequences, and each video sequence has seven frames. These frames are used as both the inputs and outputs of the auto-encoder. Consistent with previous works, we evaluate our method on the Vid4 [6] dataset and the test set of Vimeo-90k to compare with the state-of-the-art VFI, VSR, and STVSR methods.

**Implementation Details.** In our proposed STAA framework, we adopt a filter of shape $3 \times 3 \times 3$ for the downsampler. In the upsampler, one 3D convolutional layer is used to convert the input frame sequence to the feature domain. We adopt the deformable ConvLSTM [13,14] for temporal propagation. Five 3D residual-dense blocks are stacked to build the reconstruction module. During the training phase, we augment the training frames by randomly flipping horizontally and 90° rotations. The training patch size is $128 \times 128$, and the batch size is set to be 32. We train the network for 100 epochs using the Adam [5] optimizer. The initial learning rate is set to $2 \times 10^{-4}$ which scales down by a factor of 0.2 each at epochs, 50 and 80. Our network is implemented in PyTorch [8]. Our STAA framework is trained end-to-end with the $L1$-loss: $L(V, \widetilde{V}) = ||V - \widetilde{V}||_1$.

$$L(V, \widetilde{V}) = ||V - \widetilde{V}||_1. \tag{14}$$

## E More Experiments

### E.1 More Comparisons with State-of-the-Art Methods

In this section, we first show more visual results that compare the previous reconstruction methods with our joint-learned STAA in Fig. S5 and Fig. S6.

Fig. S5 shows the temporal upscaling results ($1 \times s, 2 \times t$). Compared with previous VFI methods, the reconstruction results of our proposed STAA framework have more vivid spatial textures while preserving the correct motion patterns. Because our STAA downsampler can encode the motion of each frame in the original input sequence, our reconstruction result maintains the fast motions that are lost in the previous nearest-neighbor sampling (see the pigeon in the second row).

Fig. S6 shows the space-time upscaling results ($4 \times s, 2 \times t$). Benefiting from the joint encoding of space-time dimensions, our reconstruction result has much richer spatial details even without the aid of perceptual or adversarial loss, which is almost impossible to be restored by previous methods. These results demonstrate the great advantage of co-designing the downsampler with the upsampler, which could be a potential direction for the video reconstruction task.

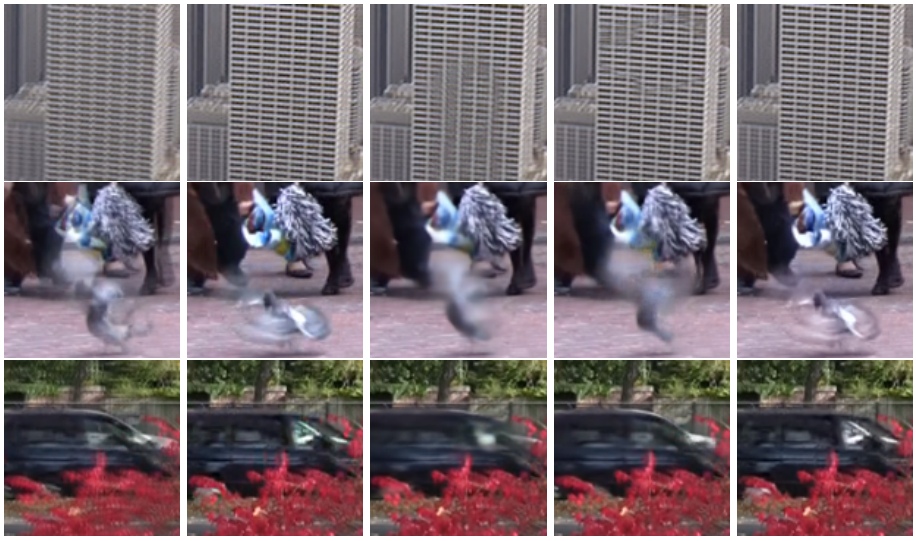



Fig. S5: Qualitative results of $1 \times s, 2 \times t$ upscaling on Vid4 dataset. Compared with previous video frame interpolation methods, the reconstruction results of our proposed STAA framework have better textures while preserving the motion patterns due to the co-design of downsampler and upsampler.

We also compare the computational efficiency of different space-time video upscaling methods by computing the FLOPs per million pixels (MP) using the open-source tool fvcore [9], as shown in Table S1. Compared with the two-stage methods with cascaded VFI and VSR networks, the one-stage space-time super-resolution network is more efficient. Our proposed upsampler requires the least computation cost among all compared methods.

### E.2 Our Framework Can be Generalized to More Networks

To compare the influence of the joint-learning mechanism of our framework, we switch the upsampler to an STVSR network ZSM [1], which converts 4 LR frames to 7 HR frames. In the updated STAA-ZSM setup, the 4 LR frames are generated by our STAA downsampler and then sent to the ZSM for reconstruction. Compared with the original ZSM in Table S2, we can observe that the STAA downsampled representation improves the PSNR by 2.14 dB and SSIM by 0.0348. These results demonstrate that the improvement brought by the joint learning framework can be generalized to other networks.

To evaluate the effectiveness of our upsampler design, we train another framework under the setting $1 \times s, 2 \times t$ by substituting the upsampler with FLAVR [3].

[1] Note that the reconstructed result is not comparable with our video upscaling setting of $4 \times s, 2 \times t$, since our network decodes 8 frames from 4 LR inputs, which is more challenging than decoding 7 out of 4.

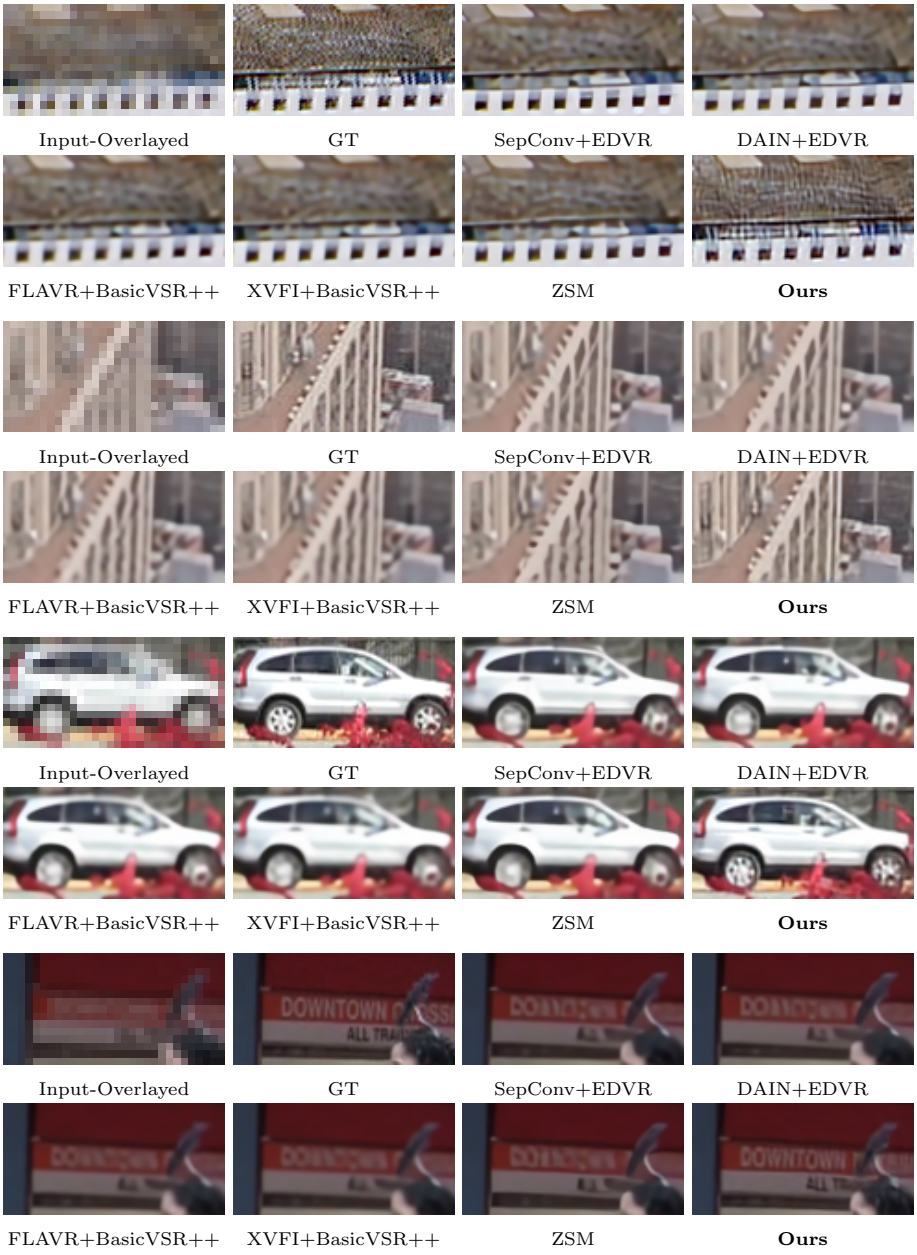



Fig. S6: Qualitative results of $4\times s, 2\times t$ upscaling on Vid4 dataset. Compared with previous two-stage and one-stage space-time super-resolution methods, the reconstruction results of our proposed STAA framework have much better textures while preserving the motion patterns due to the co-design of downsampler and upsampler.

Table S1: Comparison of computational costs among cascaded VFI and VSR methods, and space-time super-resolution, and our method.

| Method | | GFLOPs/MP |
|---|---|---|
| VFI | VSR | |
| FLAVR [3] | BasicVSR++ [2] | 185.34+140.29 |
| XVFI [10] | BasicVSR++ [2] | 676.65+140.29 |
| ZSM [13] | | 198.51 |
| Ours | | 163.98 |

Table S2: Using space-time anti-aliasing filter for space-time super-resolution.

| Method | Vimeo-90k | |
|---|---|---|
| | PSNR | SSIM |
| ZSM [13] | 33.48 | 0.9178 |
| STAA-ZSM | **35.62** | **0.9526** |

We modify the input and output numbers of the FLAVR to make it compatible with the temporal upscaling ratio. From Table S3, we can see that our proposed upsampler outperforms FLAVR by a large margin. Considering that both networks adopt 3D convolution as the basic building block, we believe that such improvement should attribute to the deformable temporal modeling.

### E.3 Influence of Filter Size

In the aformentioned experiments, we empirically set the downsampling filter size as $3 \times 3 \times 3$. Here we experimented different sizes from $1 \times 3 \times 3$ to $5 \times 7 \times 7$ (as shown in Tab S4) and found that larger sizes (e.g., $3 \times 5 \times 5$) can achieve better results but increase model complexity. For this downsampling setting $2 \times t$, $4 \times s$, keeping increasing the filter size does not bring better performance.

Table S3: Performance comparison for temporal upscaling: $2 \times t$.

| Method | Vimeo-90k | | Params(M) |
|---|---|---|---|
| | PSNR | SSIM | |
| STAA-FLAVR [3] | 44.72 | 0.9915 | 42.1 |
| STAA-Ours | **46.59** | **0.9942** | 15.9 |

Table S4: Ablation study of different downsampling filter kernel sizes ($2 \times t$, $4 \times s$).

| kernel(t, s) | Params/k | GFLOPs/MP | PSNR | SSIM |
|---|---|---|---|---|
| 1x3x3 | 0.081 | 0.027 | 32.17 | 0.9291 |
| 3x3x3 | 0.243 | 0.081 | 34.56 | 0.9413 |
| 3x5x5 | 0.675 | 0.225 | **35.15** | **0.9437** |
| 5x3x3 | 0.405 | 0.135 | 34.46 | 0.9403 |
| 5x5x5 | 1.125 | 0.375 | 34.99 | 0.9427 |
| 5x7x7 | 2.205 | 0.735 | 35.13 | 0.9425 |

## F Extensive Applications

In this section, we provide the training details for each application, and show more results and analysis. Besides, we also discuss the trade-off between the data storage and the reconstruction performance.

### F.1 Frame Rate Conversion

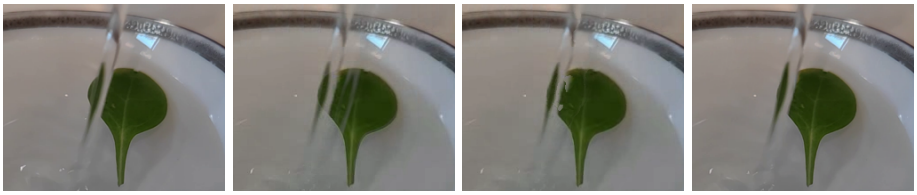
**Training Details.** We adopt REDS [7] as our training set: it contains 270 videos of dynamic scenes at $720 \times 1280$ resolution, in which 240 videos are split as the training set and 30 videos as the validation set. In this application, we train our upsampler network with the training set. During the training, we take the 5-frame sequence as input and the 6-frame sequence as the supervision for output.

To quantitatively evaluate the frame rate conversion on in-the-wild videos of our network, we took 120-fps video clips using a high-speed camera. In this way, we can get the input and corresponding ground truth for 20-fps and 24-fps frames by sampling every 6 or 5 frames, respectively. We measure the PSNR and SSIM of the output and show the results in Table S5. We also take the outputs of several popular video editing tools and software for comparison: ffmpeg [11] (skip/duplicate frames), and Adobe Premiere Pro [1]. Adobe Premiere Pro has three options for frame rate conversion: frame sampling, frame blending, and optical flow warping. Among these compared methods, only "blending" and "warping" can synthesize new frames at the new timestamp. We show these synthesized frame in Fig. S7.

From Table S5, we can conclude that the frame duplication and sampling perform the worst since it cannot synthesize frames at the new timestamp. Frame blending is a little bit better in terms of PSNR and SSIM; still, it suffers from ghosting artifacts (transparent edges due to the overlayed objects) as shown in the second figure in Fig. S7. Optical flow warping can synthesize the correct motion for the new timestamp and thus improve the reconstruction PSNR and SSIM. However, the reconstruction performance relies on the estimated optical flow field: if the estimation accuracy is low, then the warped frames would have holes, as shown in the third figure. Compared with the above methods, our network does not have these artifacts and achieves the highest reconstruction quality in terms of PSNR and SSIM.

Table S5: Performance comparison for frame rate conversion: 20 to 24 fps.

| Method | ffmpeg [11] | premiere-sampling | premiere-blend | premiere-warp | Ours |
|---|---|---|---|---|---|
| PSNR | 32.44 | 32.39 | 33.90 | 34.54 | **36.99** |
| SSIM | 0.9514 | 0.9474 | 0.9536 | 0.9588 | **0.9784** |



GT premiere-blend premiere-warp **Ours**

Fig. S7: Comparison of the intermediate frame synthesized by different methods. The blending result has obvious ghosting artifacts (transparent edges), while the optical-flow warping result has holes in the leaf. Compared with the other two methods, our result does not have these artifacts.

### F.2 Blurry Frame Reconstruction

**Training Details.** For this application, we adopt REDS [7] as our training set: it provides blurry videos at 24-fps and the corresponding clean videos at both 24-fps or 120-fps. We conduct two experiments: in the main paper, we show the results of $4 \times s, 2 \times t$, which are trained with bicubic-downsampled blurry videos at 12-fps as input, and clean HR videos at 24-fps as output. Besides, we also experiment on another setting with temporal-upscaling only: $1 \times s, 5 \times t$, which means the model needs to decode 5 clean frames per input frame. This network is trained with 24-fps blurry videos as input and 120-fps clean videos as output. To handle the motion ambiguity, we take 2 frames as input and make the network generate $2 \times t$ frames as output. We use the training subset of REDS to train the networks and show results on the validation subset.

To further illustrate the effectiveness of our upsampler design, We compare our upsampler's structure with FLAVR [3] under the setting $1 \times s, 5 \times t$ in Table S6 trained with the same hyperparameters. Our method exceeds it by a large margin in terms of PSNR and SSIM. We also provide the qualitative results of the validation videos from REDS in Fig. S8. As we can see, our method can reduce the motion blur and produce clean and crisp frames. Compared with FLAVR, our network generates vivid textures for each frame while keeping a good motion pattern at each timestep.

### F.3 Efficient Video Storage and Transmission

For this application, we assume the video capturing and reconstruction steps are asynchronous: the captured video can be downsampled and saved or transmitted

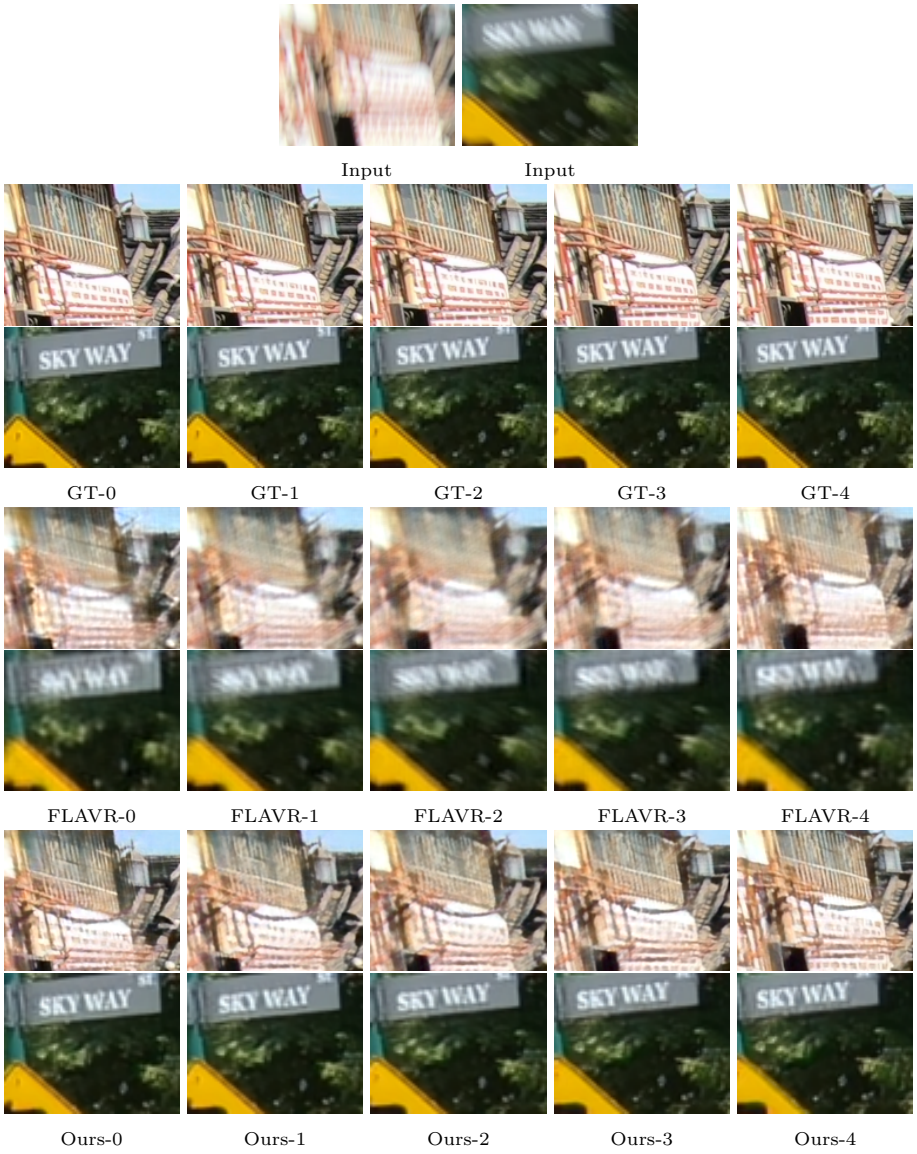



Fig. S8: Qualitative results of the blurry frame reconstruction. The first row is the input blurry frame, and the other rows are corresponding clean frames. Our method can generate clean frames with vivid textures and correct motions at each timestep.

Table S6: Performance comparison for blurry frame upscaling: $5 \times t$.

| Method | REDS | |
|---|---|---|
| | PSNR | SSIM |
| FLAVR [3] | 28.50 | 0.8337 |
| Ours | **32.29** | **0.9153** |

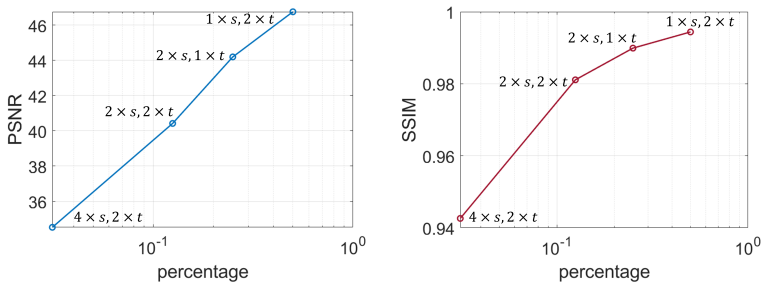
for later reconstruction. In this process, the downsampling filter can help save the data to be stored or transmitted while keeping high-quality reconstruction results. Our proposed downsampling method is a better way to reduce the number of pixels in space and time dimensions, and it is compatible with the previous image and video compression methods in the standard ISP pipeline. In practical applications, a video can be first resized by our STAA downsampler and then compressed with video codecs for transmission. Correspondingly, it requires the upsampler to be trained with the compressed data for better handling the compressed video reconstruction. We do not conduct the above experiments since it is beyond the scope of this paper. It can be a promising direction for future works.

For efficient video storage and transmission, we care about the downsampling ratio since it directly influences the amount of data to be stored or transmitted: intuitively, the restoration quality should increase with the sampling ratio. In the extreme case when the ratio = 1.0, the stored/transmitted video is exactly the original input. We discuss the trade-off between the data storage and reconstruction performance as follows.

**Trade-off between Storage and Reconstruction Performance.** We evaluate the percentage of pixels kept in the downsampled representation: for temporal downsampling ratio of $t$, there will be $1/t$ pixels kept; for spatial downsampling ratio of $s$, there will be $1/s^2$ pixels kept because of the reduction in both height and width. We conduct a series of experiments with different combinations of space and time downsampling factors and plot the influence of *pixel percentage in downsampled representation* to the *reconstruction PSNR/SSIM* calculated on Vimeo-90k's testset in Fig. S9, where the x-axis is the base-10 logarithmic scale of the pixel percentage, and the y-axis refers to PSNR and SSIM. All the experiments are done with the same network structure and trained with the same hyperparameters on Vimeo-90k.

From plot (a), we can observe that the reconstruction PSNR is almost linearly correlated to the log of pixel percentage. The SSIM plot increases slower when the percentage becomes higher, and it will eventually reach the top-right corner when the percentage is 100% and SSIM=1.0 (original video). These plots reflect the following interesting facts:

1. Under our STAA framework with jointly learned downsampler and upsampler, the reconstruction performance only relates to the percentage of pixels kept in the downsampled representation, regardless of whether the reduc-




(a) PSNR (b) SSIM

Fig. S9: *Plot of reconstruction PSNR/SSIM for different percentages of pixels in the downsampled representation.* With the increase of pixel percentage, the reconstruction results become better.

tion happens on time or space dimension. This trend validates our method's effectiveness in maintaining the spatio-temporal characteristics of the video;

2. With this plot, we can estimate a rough range of reconstruction performance given a specific downsampling ratio. This trade-off can guide the design of the video restoration system by answering the following question: what is the lowest percentage of pixels that satisfies the reconstruction quality requirement;
3. This plot can serve as a benchmark of the network's restoration capability: a better network should move the plot to the top left (reconstruct the best quality with the least percentage of pixels). Since the plot is acquired through different space/time downsampling settings, it reflects the general restoration ability beyond specific tasks.

## References


1. Adobe Inc.: Adobe premiere pro, https://www.adobe.com/products/premiere.html
2. Chan, K.C., Zhou, S., Xu, X., Loy, C.C.: Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. arXiv preprint arXiv:2104.13371 (2021)
3. Kalluri, T., Pathak, D., Chandraker, M., Tran, D.: Flavr: Flow-agnostic video representations for fast frame interpolation. arXiv preprint arXiv:2012.08512 (2020)
4. Kim, H., Choi, M., Lim, B., Lee, K.M.: Task-aware image downscaling. In: European Conference on Computer Vision. pp. 399–414 (2018)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Liu, C., Sun, D.: A bayesian approach to adaptive video super resolution. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 209–216. IEEE (2011)
7. Nah, S., Baik, S., Hong, S., Moon, G., Son, S., Timofte, R., Lee, K.M.: Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (June 2019)
8. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, pp. 8024–8035 (2019)
9. Research, M.: fvcore. https://github.com/facebookresearch/fvcore (2019)
10. Sim, H., Oh, J., Kim, M.: Xvfi: extreme video frame interpolation. In: IEEE International Conference on Computer Vision (2021)
11. Tomar, S.: Converting video formats with ffmpeg. Linux Journal **2006**(146), 10 (2006)
12. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., Change Loy, C.: Esrgan: Enhanced super-resolution generative adversarial networks. In: Proceedings of the European conference on computer vision (ECCV) workshops. pp. 0–0 (2018)
13. Xiang, X., Tian, Y., Zhang, Y., Fu, Y., Allebach, J.P., Xu, C.: Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3370–3379 (2020)
14. Xiang, X., Tian, Y., Zhang, Y., Fu, Y., Allebach, J.P., Xu, C.: Zooming slowmo: An efficient one-stage framework for space-time video super-resolution. arXiv preprint arXiv:2104.07473 (2021)
15. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision **127**(8), 1106–1125 (2019)
16. Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y.: Residual dense network for image super-resolution. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2472–2481 (2018)