Supplementary Material: Rethinking Compressive Sensing: A Compression Framework for Video Super-Resolution

Ruthy Katz* Adi Teitel* Moran Mordechay Adi Falik Eli Bery Maya Mayberg
Corephotonics, Israel
{rkatz, ateitel, mmordechay, afalik, ebery, mmayberg}@corephotonics.com

1. Experimental settings

We present more details of the training settings and experiments involved in this paper. In our flow we process the frames in YUV color space. The sampling masks are applied on Y channel only. UV channels are down-sampled by $\times 4$ with bicubic down-sampler. Then, LR Y and UV channels are concatenated and converted to RGB as input to the VSR network.

For evaluation on Vimeo-90K-T [9] and Vid4 [6], we report PSNR and SSIM on the Y channel, while LPIPS is evaluated on RGB images. For Vid4, we segment videos into 7-frame sequences, following the Vimeo-90K structure, to enable efficient processing within time and resource constraints, making it suitable for sensor applications. This contrasts with VSR recurrent models, which process entire video sequences and are thus limited to post-processing.

In single-sample-per-frame we follow [2] to initialize the model with weights of a model trained on REDS dataset [7]. We fine-tune it for additional 300K iterations. For all other experiments and ablation studies we train BasicVSR without pre-train.

2. More qualitative results

We show more visual comparisons between bicubic, 16Sum and Random sampling combined with representative VSR models for one-sample-per-frame case, see Fig. 3. In Fig. 2 for the two-samples-per-frame case we show comparison between different masks layouts combined with BasicVSR model. The examples are from Vimeo-90K-T dataset.

3. Sampling with constant masks

In one experiment we trained BasicVSR [1] using random and Tetromino masks that remain constant over time, meaning the same mask is applied to all frames in the two-samples-per-frame case. This experiment validated our premise that incorporating temporal dynamics into non-regular sampling improves reconstruction performance in the VSR task. The masks are illustrated in Fig. 1, and the training settings were consistent with those used in all other experiments.

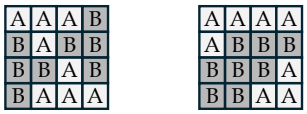

Figure 1. Constant T-Tetromino mask (a) and constant random mask (b). Pixels with the value of one in the mask are marked as A, and pixels with the value of zero in the mask (and one in the complementary mask) are marked with B.

4. Ablation study - network adaptation for two-samples-per-frame

Three configurations are explored for adding a second measurement in the two-sampels-per-frame case as input to VSR models as illustrated in Fig. 4:

1. **Flip:** During training, many VSR models flip the input sequence to simulate a long range video. We harness this configuration and use both in training and at inference time the second measurement as the flipped version instead of the first measurement. This doubles the number of processed frames.
2. **Nesting:** The frames processed in sequential order, with the odd and even masked frames processed in interleaved manner. Each two LR measurements from the same frame are paired and used in sequence. Similar to the flip configuration, the video duration is doubled.
3. **Channel stacking:** The two LR measurements are concatenated in the channels dimension. Instead of a 3-channels RGB image, we use a 6-channels input and change the first layer of the VSR model respectively. This design implicitly guides the model that the two measurements are drawn from the same frame. The output is also a 6-channels image and the last layer is

*Equal contribution

Figure 2. Visual comparison on example from Vimeo-90K-T [9] using BasicVSR [1] with various two samples-per-frame mask layouts. The frame number is shown in the figure. Zoom in for better visualization.

Figure 3. Visual comparison for an example from Vimeo-90K-T dataset [9] on different VSR models (IART [8], RVRT [5], BasicVSR [1]) with single sample-per-frame mask layouts - bicubic, 16Sum and random mask sampling, ordered from left to right. The frame number is shown in the figure. Zoom in for better visualization.

changed accordingly. We add a pointwise convolution layer to obtain a 3-channel RGB image. The video duration remains is unchanged.

We conduct ablation studies on Vimeo-90K dataset with random masks layout in two-samples-per-frame case combined with BasicVSR model. Table 1 shows quantitative evaluation of the three setups. It is shown that channel stacking is the best fit to our study case and yeilds the best performance in PSNR, SSIM and LPIPS measures both on Vid4 and Vimeo-90K-T datasets. We compared inference time of the three settings, measuring the average inference time of a 7-frame sequence from Vimeo-90K-T. The experiments were conducted on NVIDIA GeForce RTX 3090. Results are reported in Table 2. The runtine reported is the average runtime per reconstructed frame. It can be seen that not only channel stacking achieves the best reconstruction results, it is also preferable in terms of inference time and features utilization running in half of the execution time compared with the first two approaches.

5. Learnable initial reconstruction

Most VSR architectures do not explicitly use an initial reconstruction as input, instead, they rely on bilinear interpolation for initial reconstruction, with the network learning residual HR features to refine the estimate. [1, 5, 8].

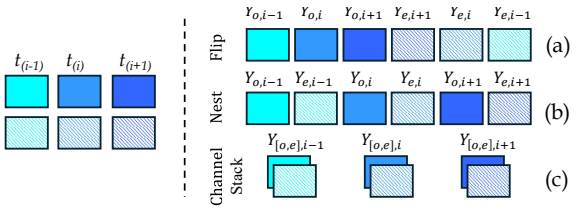

Figure 4. Two LR sequences are captured. As input to VSR network the second sequence is used as a flipped sequence (a), a separate sample paired with the first sample (b), or in channel stack configuration, as additional channels (c).

Input settings	Vimeo-90K-T	Vid4
Flip	39.9496/ 0.9730/ 0.0423	30.1695/ 0.9151/ 0.0940
Nesting	40.8813/ 0.9769/ 0.0347	30.7239/0.9257/ 0.0769
Channel stack	41.0379/ 0.9778/ 0.0335	31.2786/ 0.9292/ 0.0741

Table 1. Quantitative comparison (PSNR/SSIM/LPIPS) on Vimeo-90K-T and Vid4 of network adaptation for adding a second sample-per-frame using BasicVSR model with random masks configuration. Best result is in red.

In [3], the authors propose an end-to-end neural network-based solution for single-image super-resolution, where images are downscaled using various non-regular sampling

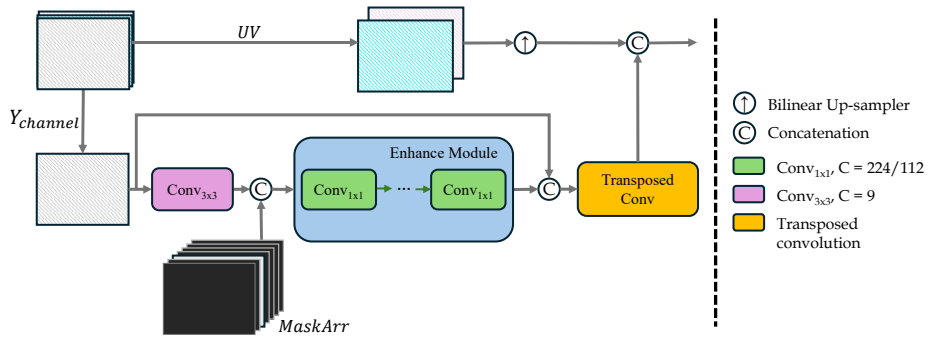

Figure 5. LFCR reconstruction network. The LR image converted to YUV. The network processes the Y channel and the output is concatenated with a bilinear upsampling of the UV channels.

Input settings	Sequence length	Input channel count	Runtime[ms] / frame
Flip	14	3	17.125
Nesting	14	3	16.862
Channel stack	7	6	9.697

Table 2. Runtime comparison for adding a second sample-per-frame using BasicVSR model with random masks configuration measured with Vimeo-90K-T dataset on NVIDIA GeForce RTX 3090. The runtine reported is the average runtime per reconstructed frame. Best result is marked in red.

methods. They replaced bicubic upsampling with a data-driven approach using a locally fully connected reconstruction (LFCR) network. Similarly, the authors of [4] employed the LFCR network for single image reconstruction and demonstrated that, when combined with their proposed sensor layout, it improves reconstruction quality.

In this section we tested whether there is an advantage in using such end-to-end solution like LFCR as the initial reconstruction instead of the bilinear up-sampler.

5.1. LFCR settings

The number of channels in the LFCR network defined in [3] considers the following elements as defined in 1:

$$C = s \times (1 - CR) \times B^2. \tag{1}$$

C is the number of channels, s is a scaling factor, CR is the compression ratio, and B is the mask block size.

Pre processing is applied so that the LFCR network is applied with the following settings:

1. Convert the LR image to YUV.
2. Y channel processed in the LFCR network.
3. UV channels are upsampled using bilinear upsampling interpolation.
4. Concatenate LFCR output with the upscaled UV.
5. Convert to RGB (if required).

We configure the LFCR parameters to align with our compression scheme, where mask block size is $B = 4$. The network settings are designed for the two-samples-per-frame case, resulting in a compression ratio of $CR = \frac{2}{B^2} = \frac{1}{8}$. We define the scaling factor as $s = 8$ or $s = 16$, which determines the network's scaling width. Consequently, the number of channels is set to $C = 112$ and $C = 224$, respectively.

To ensure a fair comparison with regular sampling schemes like bicubic downsampling, where $CR = \frac{1}{16}$, we used a single sample per frame configuration which results in the same CR. The network settings remained unchanged.

The entry stage of the LFCR network partitions the LR image into overlapping blocks and transforms each block into a vector. In our configuration 3×3 neighboring pixels are stacked into a vector, forming a $\frac{H}{4} \times \frac{W}{4} \times 9$ data block. Our sampling scheme incorporates a temporal aspect, as each frame in a 7-frame sequence is sampled using a different mask. Since each frame's initial reconstruction is processed independently, it is crucial for the upsampler to receive information about the sample structure. Instead of passing the explicit mask, we inform the model of the frame's position within the sequence. We form a tensor of size $\frac{H}{4} \times \frac{W}{4} \times 7$ with elements as in 2:

$$MaskArr\,(:,:,i) = \begin{cases} 1, & i = f \\ 0, & else \end{cases} \tag{2}$$

Where f is the index of the processed frame and $1 \leq i \leq 7$. This tensor is concatenated with the data block to produce feature maps of size $\frac{H}{4} \times \frac{W}{4} \times 16$. Then, a series of L pointwise convolutions is applied. Last, the original LR image is stacked with the output features, and transposed convolution is performed to re-order the reconstructed pixels to the HR Y image. Fig. 5 demonstrates the LFCR pipeline.

As in [3], the LFCR+BasicVSR network is trained in two steps. First, LFCR is trained for 50K iterations using Charbonnier penalty loss against the HR reference, with inputs matching the sampling-mask configuration. Then, BasicVSR is trained for 300K iterations with the same loss

Down-sampling	Up-sampling	L	Channels	Vimeo	Vid4
Bicubic	bilinear	-	-	30.46/ 0.8539/ 0.3077	23.33/ 0.6071/ 0.5000
Random	bilinear	-	-	30.34/ 0.8583/ 0.2834	23.17/ 0.6115/ 0.4736
Random	LFCR	4	224	31.72/ 0.8829/ 0.1758	23.88/ 0.6671/ 0.3345
Random	LFCR	10	112	28.94/ 0.8255/ 0.3221	22.68/ 0.6036/ 0.4440

Table 3. Quantitative comparison (PSNR/SSIM/LPIPS) on Vimeo-90K-T and Vid4 for $\times 4$ upsampling in single-sample-per-frame case, with 2 reconstruction methods (not combined with a VSR model): LFCR [3] and bilinear up-sampler using different down-sampling methods. Best result is in red.

Down-sampling	Up-sampling	L	Vimeo	Vid4
Bicubic	BasicVSR	-	37.30/ 0.9461/ 0.1006	27.16/ 0.8195/ 0.2524
Random	BasicVSR	-	39.00/ 0.9670/ 0.0544	28.91/ 0.8905/ 0.1271
Random	LFCR + BasicVSR	4	38.48/ 0.9639/ 0.0634	28.53/ 0.8823/ 0.1501
Random	LFCR + BasicVSR	10	38.21/ 0.9624/ 0.0675	28.35/ 0.8777/ 0.1599

Table 4. Quantitative comparison (PSNR/SSIM/LPIPS) on Vimeo-90K-T and Vid4 for $\times 4$ VSR in single-sample-per-frame case, with 2 reconstruction methods: LFCR [3] + BasicVSR and BasicVSR [1] using different down-sampling methods. Best result is in red.

function while keeping the LFCR layers frozen.

5.2. Results

Two configurations were explored for the LFCR pointwise convolution series: $L = 10$ as in [4] and $L = 4$. Table 3 compares bilinear upsampling and LFCR network without additional VSR network. The results support the results of [3], showing that a data-driven approach is more suitable than bilinear upsampler when combined with non-regular sampling. The optimal settings in our case were different than the ones demonstrated in [3]. However, when the LFCR network is combined with BasicVSR network, there is no advantage to the data-driven approach as shown in Table 4. The conclusion is that a simple initial reconstruction combined with a VSR model is sufficient to recover the super-resolved video, eliminating the need for an end-to-end neural network solution and simplifying the overall approach. By omitting the LFCR model, we reduce both training time and inference complexity, making our approach more efficient. Unlike SISR, VSR overcomes non-regular sampling implicitly.

6. Limitations

There are several limitations to the proposed solution in this paper:

1. **Dependency in mask order:** The trained network implicitly learns to reconstruct the video sequence without requiring the mask given as input for a given mask sequence. Consequently, altering the mask order will degrade the VSR model's performance. This limitation can be mitigated by training with cyclic mask shifts, which aligns well with hardware compression using a finite, ordered set of masks. Alternatively, the model can be trained with explicit mask inputs.
2. **Compression ratio:** in the two samples-per-frame configuration CR=$\frac{1}{8}$, which is doubled compared to regular $\times 4$ down-scaling. Doubling the input affects system bandwidth and may require twice the memory for storing measurements. To mitigate this, we propose saving one measurement along with the quantized difference between the two. Since the measurements are fairly similar, quantization is expected to have minimal impact on reconstruction quality compared to storing full measurements. This approach enables a more compact memory format than storing two separate videos.
3. **Sensor compatibility:** We demonstrated our solution with compression applied to the Y channel. To align it with a sensor-level implementation, future work should evaluate reconstruction quality when applied with the Bayer pattern and converted to RGB images.

References

[1] Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond. In *CVPR*, pages 4947–4956, 2021. 1, 2, 4

[2] Kelvin CK Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. In *CVPR*, pages 5972–5981, 2022. 1

[3] Simon Grosche, Fabian Brand, and André Kaup. A novel end-to-end network for reconstruction of non-regularly sampled image data using locally fully connected layers. In *IEEE MMSP*, pages 1–6, 2021. 2, 3, 4

[4] Simon Grosche, Andy Regensky, Jürgen Seiler, and André Kaup. Image super-resolution using T-Tetromino pixels. In *CVPR*, pages 9989–9998, 2023. 3, 4

[5] Jingyun Liang, Yuchen Fan, Xiaoyu Xiang, Rakesh Ranjan, Eddy Ilg, Simon Green, Jiezhang Cao, Kai Zhang, Radu Timofte, and Luc V Gool. Recurrent video restoration transformer with guided deformable attention. *NeurIPS*, 35:378–393, 2022. 2

[6] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *IEEE TPAMI*, 36(2):346–360, 2013. 1

[7] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPRW*, 2019. 1

[8] Kai Xu, Ziwei Yu, Xin Wang, Michael Bi Mi, and Angela Yao. Enhancing video super-resolution via implicit resampling-based alignment. In *CVPR*, pages 2546–2555, 2024. 2

[9] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 127:1106–1125, 2019. 1, 2