Time Without Time: Pseudo-Temporal Representation for Space-Time Super-Resolution

Supplementary Material

6. Datasets

We describe datasets used in our experiments. A summary of each dataset is illustrated in Table 6. If a dataset does not officially provide low-resolution videos (images), we create low-resolution videos (images) using the bicubic interpolation method.

ImageNet [10]. ImageNet is a popular image classification dataset. Due to its large volume, we only adopt the 10% train split of [3] for pretraining.

Vimeo-90K [60]. Vimeo-90K `septuplet` consists of seven-frame video clips, where its train and test datasets include 64,612 and 7,815 clips, respectively. The spatial resolution of the original video is 448×256. We follow the common protocol of measuring performance on 3 sequences of the testset proposed in [23]. Here, the testset is divided into 3 categories according to the average motion flow magnitude, fast, medium and slow, which include 1225, 4977, and 1613 clips.

REDS [42]. REDS is composed of 300 video clips with dynamic scenes, and each video has 100 frames. The training, validation and testing sets include 240, 30, and 30 videos, respectively. The spatial dimension of low-resolution and high-resolution frames are 320×180 and 1080×720, respectively. We adopt the validation set for testing as high resolution videos of the testset is not publicly available. We split each 100-frame video into 14 sets with non-overlapping 7 frames ($14\,\text{sets} \times 7\,\text{frames/set} = 98\,\text{frames}$). For the remaining 2 frames (99th and 100th), we create 2 sets by grouping the last 7 frames together, counting backward from the end.

Vid4 [38]. Vid4 is a small dataset and exclusively used for evaluation purposes. It consists of four sequences, walk (740×480, 47 frames), foliage (740×480, 49 frames), city (704×576, 34 frames), and calendar (720×576, 41 frames).

Inter4K [48]. Inter4K provides a diverse set of 5-second videos with standardized UHD resolution. We split each video into clips with non-overlapping 7 frames, and randomly divide them into 649 train clips and 353 test clips.

7. Implementation Details of Our Framework

7.1. Pretraining and Finetuning Details

We provide implementation details of pretraining and finetuning Zooming [58], TMNet [59], RSTT [18] and SAFA [31]. For each network, we run experiments using their official implementation and optimal hyperparameters with minor modifications such as a batch size and the number of GPUs used for training. We find their main results are successfully reproducible with these modifications. We first describe the common settings for pretraining in our framework and then illustrate each network-specific details. For the SOTA comparison in Table 5, we finetuned the models for 200 epochs, whereas all other experiments followed the default settings described below.

Masking. We follow spatio-temporal random masking strategy of [15]. For completeness, we provide details. We divide each input frame into non-overlapping 4×4 blocks. We randomly sample a subset of blocks with a masking ratio of p (default $p = 0.5$) for each frame. Here, we zero out the sampled blocks.

Visibility count. The masking block in the input is a size of 4×4, so we set the corresponding block size for visibility counting in the output to $(4 \times \text{scale}_s) \times (4 \times \text{scale}_s)$ when the spatial upsampling factor is scale_s. Therefore, for every pixel location within an output block, the visibility count is the same.

Zooming. Random crop and resize, horizontal flip and random rotation of 90°, 180°, or 270° augmentation strategies are used for creating input frames. The spatial input resolution is 64×64 for the default $\times 4$ upsampling task and 128×128 for the $\times 2$ upsampling experiment. Adam optimizer [34] is adopted with weight decay of $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The network is trained using cosine annealing with restart [20] with initial learning rate of 4×10^{-4} and minimum of 10^{-7}. For every 15,000 iterations, the restart is performed. For pretraining, we use 64 batches. When finetuning, we train the network with a batch size of 32 for the default spatial upsampling of $\times 4$ and 24 for the upsampling of $\times 2$. All experiments are conducted on two NVIDIA A100 GPUs. We replace the built-in deformable convolutional network (DCN) module [9] with the implementation from MatthewHowe/DCNv2.

TMNet. Random crop and resize, horizontal flip and random rotation of 90°, 180°, or 270° augmentation strategies are used for creating input frames. The spatial input resolution is 32×32 for the default $\times 4$ upsampling and 64×64 for the $\times 2$ upsampling experiment. We adopt the Adam optimizer [34] with 0.9 for β_1 and 0.999 for β_2. For training scheduling, cosine annealing with restart [20]

Table 6. **Dataset information.** The table summarizes the information of each dataset. For some datasets, small subsets are used for finetuning, so the percentage is noted in the fifth column (%). For video datasets, the last column indicates (number of clips) × (frames per each clip).

Dataset	Data Type	Resolution	Split	%	#Images
ImageNet [10]	Image	various	train	10	128,116
Vimeo-90K [60]	Video	448×256	train	100	64,612×7
				10	6,462×7
				1	647×7
			fast	100	1,225×7
			medium	100	4,977×7
			slow	100	1,613×7
REDS [42]	Video	1080×720	train	100	3,840×7
				10	384×7
			validation	100	480×7
Vid4 [38]	Video	740×480	walk	100	1×47
		740×480	foliage	100	1×49
		704×576	city	100	1×34
		720×576	calendar	100	1×41
Inter4K [48]	Video	3840×2160	train	100	649×7
			test	100	353×7

every 15,000 iterations was used. The learning rate started at 4×10^{-4} and decayed to 10^{-7}. In the pretraining phase, we trained the network for 200 epochs with a batch size of 512 on eight NVIDIA A100 GPUs. In the finetuning phase, we used 50 epochs with a batch size of 24 for $\times 2$ upsampling experiments and 64 or 128 for $\times 4$ upsampling experiments. All finetuning experiments were performed on two NVIDIA A100 GPUs.

RSTT. Random crop, horizontal and temporal flip augmentations are applied to create input frames. The spatial input resolution is 64×64 for the default upsampling and 128×128 for the $\times 2$ upsampling experiment. Adam optimizer [34] is adopted with weight decay of $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The network is trained using cosine annealing with restart [20] with initial learning rate of 2×10^{-4} and minimum of 10^{-7}. For every 40,000 iterations, the restart is performed. Different from the original recipe, we train the network on two NVIDIA A100 GPUs with a batch size of 24. The default pretraining and finetuning epochs are set to 200 and 50. When finetuning, multiple learning rates ranging from 2×10^{-4} to 9×10^{-4} are swept over and the best results are reported.

SAFA. To ensure a fair comparison with other approaches, we adapt the original SAFA implementation by constructing each training sample with 7 consecutive frames rather than the original 9-frame configuration, while keeping all other settings consistent with the authors' implementation. For spatial augmentation, we follow the authors' setup by applying random cropping to 128×128, random horizontal and vertical flips, and random RGB channel reversal. The cropped high-resolution patches are then downsampled by a factor of $1/4$ and subsequently upsampled back to 128×128 using bicubic interpolation to produce the low-resolution inputs. For the temporal dimension, we randomly select one intermediate frame, whose corresponding timestep is drawn from $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$, while the first and last frames serve as input. This modification corresponds to the authors' original strategy of choosing intermediate frames from $\frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}$ when using 9-frame samples. During training, the learning rate is gradually reduced from 2×10^{-4} to 0 following a cosine annealing schedule. During the pretraining phase, we trained the model for 200 epochs with a batch size of 48 on eight NVIDIA A100 GPUs, and in the subsequent finetuning stage, we reduced the batch size to 24.

7.2. Partial Finetuning Details

RSTT [18] consists of encoder and decoder layers. For all self-supervised pretrained models, we use pretrained representations of encoder layers, and decoder layers are trained from scratch. We use the same setups for finetuning such as optimizer, learning rate settings, batch size and epochs described in Appendix 7.1.

7.3. Finetuning Details for More Downstream Tasks

RSTT-S network is used for validating transfer performance of $4\times$ video super-resolution (VSR), $2\times$ video frame interpolation (VFI) and deblur plus $4\times$ VSR (DSR) tasks. We follow the same setup for finetuning STVSR network described in Appendix 7.1. We measure VFI performance using 2nd, 4th and 6th output frames after downsampling them to the original input spatial dimension of the STVSR network, and 1st, 3rd, 5th and 7th frames are evaluated for

VSR. For DSR experiments, we adopt REDS-Deblur [42] dataset and test protocol of VSR.

8. Implementation Details of Other Self-Supervised Learning Methods

To compare with other self-supervised methods, we employ RSTT-S [18] network architecture. As MAE-ST [15], VideoMAE [50], SIGMA [47] and EVEREST [32] are designed for ViT [11] and MoCo-v3 [4] takes images as input, we reimplement the three methods by ourselves. Our implementation involves applying the core idea of each method to pretraining RSTT-S.

8.1. MAE-ST, VideoMAE, SIGMA and EVEREST

We first describe the common settings for mask autoencoder based methods of MAE-ST, VideoMAE, SIGMA and EVEREST illustrate the method-specific masking strategy and additional module.

Input. In masked autoencoder based frameworks, each input frame is divided into small patches. The core idea involves random sampling a subset of patches, applying masking to the selected ones and reconstructing the masked patches from unmasked ones. In our reimplementation, we also use videos as input, but masking means filling the masked blocks with zeros, not dropping those blocks. We follow their optimal masking ratio of 0.9. In order to accommodate the small input resolution of 64×64, we adopt a small block size of 4×4, whereas patch dimension is 16×16 for the input size of 224×224 in the original masked autoencoder approaches.

Decoders. Decoder layers are only used at pretraining in the masked autoencoder methods. For a fair comparison, we maintain the original architecture of RSTT-S as possible. In particular, we use the original decoder layers except for the last three convolution layers. These three layers are designed for spatial upsampling, which are not necessary for the reconstruction task adopted in the masked autoencoders. Therefore, we replace the three layers with one 1×1 convolution layer to match the temporal output dimension of the network with the temporal input dimension. When finetuning, only the encoder features are adopted and decoder layers are initialized from scratch.

Hyperparameters. We use the same setups described in Appendix 7.1 such as an optimizer, a learning schedule, a batch size and train epochs. Multiple learning rates ranging from 2×10^{-4} to 9×10^{-4} are swept over and the best results are reported.

Masking strategy. We use the optimal masking strategy proposed by each approach. Spatio-temporal-agnostic random masking is applied for MAE-ST and tube masking is used for VideoMAE and SIGMA. EVEREST proposes a mask generator to select tokens with a large disparity in the previous time dimension.

Additional module. Only SIGMA has a projection module for computing Sinkhorn features, whereas the other methods do not include any additional modules.

8.2. MoCo-V3

MoCo-V3 is originally designed for image pretraining, but we use videos as input with a size of $4 \times 3 \times 64 \times 64$ (4-frame RGB input with a resolution of 64×64). A query encoder consists of a backbone, a projection head and a prediction head, and a key encoder consists of a backbone and a projection head. In our reimplementation, we use the encoder layers of RSTT-S [18] as backbone. The projection head is 3-layer multi-linear perceptron (MLP), and the prediction head is 2-layer MLP. The hidden dimension of MLP is 4096, and the output layer dimension is 384. The momentum coefficient is set to 0.9, and the temperature

Table 7. **Effectiveness of the proposed pretraining on a different upsampling task.** Various networks are pretrained for the task of upsampling spatial dimension by $\times4$ and temporal dimension by $\times2$ using Vimeo-90K [60] trainset. We then finetune the learned representation for the task of $\times2$ spatial and $\times2$ temporal upsampling on 1% and 10% train splits of Vimeo-90K, REDS [42] trainset and its 10% splits. For Vimeo-90K experiments, we report the performance on the fast sequence of the testset. Overall, our pretraining shows better results than no pretraining (i.e., training from scratch), and the proposed method is more effective when the quantity of finetuning dataset is limited. We note that our pretrained representations are successfully transferred to a different upsampling task.

Method	Vimeo-90K 1%			Vimeo-90K 10%			REDS 10%			REDS 100%		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Zooming [58]	34.04	0.9089	0.1401	38.12	0.9507	0.0718	27.49	0.7695	0.4236	29.71	0.8043	0.2927
Zooming + **Ours**	37.33	0.9284	0.1049	39.20	0.9542	0.0913	29.42	0.7942	0.3075	29.91	0.8062	0.2884
TMNet [59]	35.48	0.9145	0.1077	39.16	0.9552	0.0690	28.02	0.7797	0.3739	29.97	0.8093	0.2945
TMNet + **Ours**	38.16	0.9493	0.0783	39.59	0.9601	0.0596	29.63	0.8058	0.2868	30.13	0.8131	0.2576
RSTT [18]	36.73	0.9204	0.1023	39.41	0.9534	0.0904	28.86	0.7835	0.3033	29.91	0.8064	0.2837
RSTT + **Ours**	38.38	0.9393	0.0908	40.22	0.9624	0.0799	29.58	0.7966	0.2938	30.25	0.8145	0.2600

Table 8. **Higher-scale super-resolution results.** RSTT-S [18] network is pretrained for the task of upsampling spatial dimension by $\times 8$ and temporal dimension by $\times 2$ on REDS [42] dataset using various self-supervised methods and then finetuned on five non-overlapping subsets (1%) of Vimeo-90K [60] trainsets. For each method, the average performance on fast split of Vimeo-90K testset is reported using PSNR and SSIM [54]. Our method consistently achieves much higher performance than other self-supervised approaches using videos.

Pretrain Dataset	Method	Pretrain Data Type	Vimeo-Fast PSNR↑	Vimeo-Fast SSIM↑
None	Scratch	-	29.07	0.8233
REDS	Ours	Image	**29.93**	**0.8383**
	MAE-ST [15]	Video	29.12	0.8237
	VideoMAE [50]	Video	29.13	0.8237
	MoCo-v3 [4]	Video	29.29	0.8271
	SIGMA [47]	Video	29.04	0.8224
	EVEREST [32]	Video	29.43	0.8307

parameter for the loss function is 0.2. The total pretrain epoch is set to 200 with a warmup epoch of 40. We follow the linear scaling rule with $1.5 \times 10^{-4} \times (\text{batch size})/256$, where 1.5×10^{-4} is a base learning rate. The AdamW optimizer [43] is used with a weight decay of 10^{-6}, after the warmup period, learning rate follows a cosine decay schedule [41].

9. More Analysis

9.1. Performance on Different Upsampling Task

We test finetuning performance of Vimeo-90K [60] pretrained representations to a different STVSR task. The target task is upsampling the spatial dimension by $\times 2$ and temporal dimension by $\times 2$. We reuse the pretrained weights for the task of increasing spatial resolution by $\times 4$ and frame rate by $\times 2$. Due to the architecture difference between pretraining and finetuning, the last three convolution layers related to spatial upsampling are trained from scratch for the finetuning stage. We demonstrate the effectiveness of the proposed pretraining across various STVSR networks such as Zooming [58], TMNet [59] and RSTT [18] on multiple datasets (Vimeo-90K 1% and 10%, REDS 10% and REDS trainsets). We use their official implementations and optimal hyperparameter settings provided by the authors. Table 7 compares our pretraining vs. no pretraining. For Vimeo-90K, results on the fast test sequence are illustrated. Similar to Table 1, we see that our pretrained models consistently yield better results for all architectures, datasets and evaluation metrics. Also, the proposed pretraining is more effective when the target finetuning dataset is limited. We note that pretrained representations for a different upsampling rate can be successfully transferred.

9.2. Higher-Scale Super-Resolution Results

We further evaluate the proposed method on a task with $\times 2$ temporal and $\times 8$ spatial upsampling to test its effectiveness for higher-scale super-resolution. As done in Figure 1, RSTT-S [18] network is pretrained on REDS [42] dataset using various self-supervised pretraining methods and then finetuned on five non-overlapping subsets (1%) of Vimeo-90K [60] trainsets for space-time video super-resolution task. For each method, the average performance on fast split of Vimeo-90K testset is reported using PSNR and SSIM [54]. Table 8 shows that our method consistently achieves much higher performance than other self-supervised approaches using videos.

9.3. Comparison with Same-Task Pretraining

We compare the performance of our framework with same-task representation learning on Vimeo-90K [60] using RSTT-S [18]. Here, same-task representation learning indicates that the target network itself is pretrained on REDS [42] dataset to perform the target task (spatial $\times 4$ and temporal $\times 2$). When implementing the proposed framework, we consider two datasets, REDS and ImageNet [10]. REDS trainset consists of 3840 video clips with each clip of 7-frames, and we use one randomly selected frame for each clip in our pretraining. For a fair comparison, we select 3840 images in ImageNet trainset and pretrain the network on this small subset. Finetuning is performed on Vimeo-90K 10% train splits and we report PSNR results. Table 9 illustrates that our method is slightly better than same-task representation learning. Our pretrained model is as effective as the model trained on a different dataset for the target space-time video super-resolution task, and it has the advantage of being able to use image datasets.

Table 9. **Comparison with same-task representation learning.** RSTT-S [18] network is pretrained using different pretraining methods. Same-task representation learning (Same-task) indicates that the network itself is pretrained on REDS videos to perform STVSR. Ours-R and Ours-I denote our pretraining on REDS [42] and our pretraining on a ImageNet [10] subset, respectively. Finetuning is performed on Vimeo-90K 10% train splits for each method and average PSNR values are reported. Our pretrained model is as effective as a model trained on a different dataset for the target STVSR task, and it has the advantage of using images.

Method	Data Type	#Images	Vimeo-90K Fast	Vimeo-90K Medium	Vimeo-90K Slow
Scratch	None	0	34.56	33.88	32.13
Same-task	Video	3,840×7	35.27	34.29	32.40
Ours-R	Image	3,840×1	**35.30**	**34.46**	**32.57**
Ours-I	Image	3,840×1	35.27	34.38	32.50

9.4. Results for Various Number of Input Frames and Masking Ratios

We investigate how the finetuning performance varies with the masking ratio used in our pretraining for different numbers of input-output frame combinations. Due to computational constraints, we use small-scale REDS dataset [42] for pretraining for scenarios with smaller input-output frames and follow the rest of experimental setups of Table 3b. Table 10 shows that finetuning performance (PSNR) on Vimeo-90K fast split of testset is robust as long as masking ratio is not too large (≤ 0.7) for all input-output frame combinations.

9.5. Finetuning Performance on High-Resolution Dataset

We further demonstrate the effectiveness of the proposed framework on Inter4K [48], a small-scale, high-resolution (4K) with rapid motion (Figure 4 in [48]) dataset, for STVSR task. We compare our method with other self-supervised learning methods as in Section 4.4. We pretrain RSTT-S[18] on REDS [42] using various self-supervised methods and then perform finetuning on Inter4K. Table 11 shows that transfer performance of our pretrained weight is best.

Table 10. **Finetuning performance with different masking ratios and input-output frame combinations.** RSTT-S [18] network is pretrained on REDS [42] training set using the proposed framework and finetuned on five Vimeo-90K [60] 1% train splits. The PSNR results are averaged over five splits. Finetuning performance is not sensitive for all input-output frame combinations as long as masking ratio is not too large.

Masking Ratio	Input Frames $\rightarrow$ Output Frames		
	$4 \rightarrow 7$	$3 \rightarrow 5$	$2 \rightarrow 3$
Scratch	31.66	32.29	32.38
0.3	33.46	33.63	33.62
0.5	33.58	33.60	33.62
0.7	33.57	33.64	33.52
0.9	33.07	33.06	33.01

Table 11. **Finetuning performance of various self-supervised models on Inter4K.** RSTT-S [18] network is pretrained on REDS [42] dataset using various self-supervised methods. The results after finetuning on Inter4K dataset for STVSR task are reported. Transfer performance of our method is better than other approaches.

Method	Pretrain Data Type	#Pretrain Images	PSNR↑	SSIM↑	LPIPS↓
Scratch	None	0	33.72	0.9088	0.1404
MAE-ST [15]	Video	3,840×7	33.69	0.9081	0.1443
VideoMAE [50]	Video	3,840×7	33.24	0.9024	0.1608
MoCo-v3 [4]	Video	3,840×7	33.38	0.9038	0.1575
Ours	Image	3,840×1	**34.75**	**0.9226**	**0.1262**

9.6. Pretrain Time

As our pretraining method uses videos as input, it takes long train time similar to models designed for video-related tasks. The per-epoch pretrain time is 48 minutes on 2 NVIDIA A100 GPUs with 24 batches using Vimeo-90K trainset (64,612 videos) in our framework. However, our method does not require long epochs (see Table 3c and 16c) and the proposed pretraining significantly reduces finetuning time (see Figure 6).

9.7. Per-epoch Finetuning Performance

Figure 6 provides a per-epoch finetuning results of our pretraining and no pretraining. Pretraining and finetuning is performed on Vimeo-90K [60] trainset and Vimeo-90K 10% train splits using RSTT-S [18], respectively. The best performance for training from scratch (PSNR $=$ 34.55dB at 50 epochs) is obtained at 8-epoch finetuning of our pretrained weights (PSNR $=$ 34.61dB). This indicates that the proposed framework helps significantly reduce finetuning time to achieve similar results compared to training from scratch. Moreover, if we finetune our pretrained model longer, then pretraining clearly outperforms no pretraining.

10. Full Tables for Comparison with Other Pretraining Methods

10.1. Partial Finetuning Performance

Table 12 presents the complete results of the various self-supervised pretraining methods and partial finetuning settings. The Figure 3 provides a visualization of a subset of these results. Using weights pretrained for 200 epochs on the REDS dataset, we freeze the encoder and finetune only the decoder for 50 epochs on the Vimeo-90K 1% splits. Performance is evaluated on the Fast, Medium, and Slow

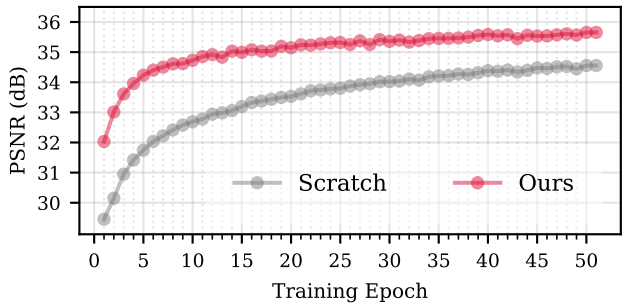

Figure 6. **Per-epoch finetuning performance.** RSTT-S [18] is pretrained on Vimeo-90K [60] trainset and finetuning is performed on Vimeo-90K 10% train splits. The best PSNR performance for training from scratch is obtained at 8-epoch finetuning of our pretrained weights. Furthermore, if we finetune our pretrained model longer, then pretraining clearly outperforms no pretraining.

Table 12. **Partial finetuning performance of various self-supervised methods.** In the RSTT-S network, the encoder is frozen and only the decoder is finetuned using weights pretrained on the REDS dataset with different self-supervised approaches. The table summarizes the evaluation results on the Fast, Medium, and Slow splits of Vimeo-90K after finetuning on the 1% Vimeo-90K splits.

Method	Pretrain Data Type	#Pretrain Images	Fast			Medium			Slow		
			PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
MAE-ST [15]	video	64612×7	30.15	0.8403	0.2599	30.19	0.8526	0.2731	29.33	0.8339	0.3014
VideoMAE [50]	video	64612×7	31.51	0.8639	0.1953	31.66	0.8819	0.1862	30.66	0.8680	0.2057
MoCo-v3 [4]	video	64612×7	30.63	0.8497	0.2375	30.72	0.8630	0.2449	29.79	0.8452	0.2708
SIGMA [47]	video	64612×7	29.98	0.8379	0.2637	30.00	0.8483	0.2834	29.17	0.8302	0.3131
EVEREST [32]	video	64612×7	31.49	0.8655	0.2155	31.38	0.8752	0.2125	30.33	0.8587	0.2364
Ours	image	64612×1	**32.67**	**0.8807**	**0.1833**	**32.48**	**0.8950**	**0.1675**	**31.24**	**0.8793**	**0.1863**

subsets of Vimeo-90K. This shows that, with our pretrained weights, even finetuning only part of the model on a small amount of data can achieve better image quality than other representation learning methods.

10.2. More Downstream Performance

Due to space limitations, we provide additional finetuning results for other self-supervised models (EVEREST [32] and SIGMA [47]) in Table 13, complementing Table 4. Our method outperforms other methods on 2× video frame interpolation (VFI) at Vimeo-90K and 4× video super-resolution (VSR) at REDS. On the DSR task (deblur plus 4× VSR), our approach shows a slight performance degradation relative to EVEREST.

10.3. Quality of the Pretrained Representation

We also provide RankMe [17] values of the features of the last encoder layer (embeddings) obtained from SIGMA [47] and EVEREST [32] and finetuning results on Vimeo-90K [60] 1% and 10% trainsets in Table 14. This serves as a complement to Figure 4. Our pretrained representation achieves a substantially higher RankMe value, which has been shown to correlate positively with downstream performance [17]. This further demonstrates the effectiveness of our pretraining approach.

Table 13. **Finetuning performance of various self-supervised models.** RSTT-S [18] network is pretrained on REDS [42] dataset using different self-supervised methods. The PSNR results after finetuning for video frame interpolation (VFI), video super-resolution (VSR) and deblur plus video super-resolution (DSR) are reported. Transfer performance of our method is better than other approaches across different downstream tasks except for DSR.

Method	Pretrain Data Type	#Pretrain Images	Vimeo-90K VFI	REDS VSR	REDS DSR
Scratch	none	0	28.13	35.50	26.28
MAE-ST [15]	Video	3,840×7	28.10	35.48	26.31
VideoMAE [50]	Video	3,840×7	28.05	35.44	26.34
MoCo-v3 [4]	Video	3,840×7	28.34	35.60	26.29
SIGMA [47]	Video	3,840×7	28.12	35.52	26.30
EVEREST [32]	Video	3,840×7	28.77	35.97	**26.70**
Ours	Image	3,840×1	**30.90**	**37.06**	26.60

11. Full Tables for Ablation Studies

We have shown ablation results using PSNR on Table 2 and 3. Here, we provide the ablation results using SSIM [54] metric in Table 15 and 16.

12. More Visualization Examples

We provide more visualization examples for finetuned models initialized with different pretraining methods. See Figure 7, 8 and 9. Full-resolution visualizations are available at https://huggingface.co/spaces/twt2026/vis.

Table 14. **Quality of pretrained representation.** RankMe [17] values of REDS [42] pretrained encoder embeddings from different self-supervised learning models and their finetuning performance on Vimeo-90K [60] 1% and 10% trainsets is reported. Our method shows a significantly higher rank and finetuning performance, which indicates superiority of our pretraining.

Method	Pretrain Data Type	#Pretrain Images	RankMe ↑	Vimeo-90K 1%		Vimeo-90K 10%	
				PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
MAE-ST [15]	Video	3,840×7	13.08	31.60	0.8672	34.56	0.9133
VideoMAE [50]	Video	3,840×7	11.57	31.64	0.8681	34.54	0.9130
MoCo-v3 [4]	Video	3,840×7	67.19	31.79	0.8715	34.61	0.9141
SIGMA [47]	Video	3,840×7	2.64	31.66	0.8687	34.56	0.9132
EVEREST [32]	Video	3,840×7	188.26	32.15	0.8756	34.70	0.9154
Ours	Image	3,840×1	**237.64**	**33.46**	**0.8958**	**35.30**	**0.9239**

Table 15. **Ablation on pretraining dataset.** RSTT-S [18] is pretrained with different datasets, and finetuning is performed on 1% and 10% of Vimeo-90K [60] trainset and REDS [42] trainset and its 10% splits. The SSIM [54] results show that the finetuned networks initialized from our pretrained weights surpass models trained from scratch. Also, ImageNet [10] pretrained representation is effectively transferred to space-time video super-resolution tasks. Our method is effective even when pretraining is performed with a small-scale dataset like REDS.

Pretrain Dataset	#Pretrain Images	Vimeo-90K 1%			Vimeo-90K 10%			REDS 10%	REDS 100%
		Fast	Medium	Slow	Fast	Medium	Slow		
Scratch	0	0.8682	0.8820	0.8641	0.9133	0.9172	0.8943	0.6823	0.7188
ImageNet 10% [10]	128,116	**0.9078**	0.9126	0.8907	**0.9278**	**0.9263**	**0.9031**	0.7148	**0.7343**
Vimeo-90K [60]	64,612	0.9067	**0.9131**	**0.8928**	0.9276	0.9262	**0.9031**	**0.7170**	0.7339
REDS [42]	3,840	0.8958	0.9050	0.8831	0.9239	0.9250	0.9016	0.7101	0.7296

Table 16. **Ablation experiments on Vimeo-90K.** RSTT-S [18] network is pretrained on Vimeo-90K [60] training set using the proposed method and finetuned on five Vimeo-90K 1% train splits. The SSIM [54] results are averaged over the five splits.

(a) **Modulating factor.** We compare different modulating factors. The proposed difficulty-adaptive strategy works best.

Method	Vimeo-90K		
	Fast	Medium	Slow
Scratch	0.8682	0.8820	0.8641
Equal	0.8957	0.9088	0.8922
Onehot	0.8935	0.9069	0.8901
Ours	**0.9067**	**0.9131**	**0.8928**

(b) **Masking ratio.** Our method is not sensitive to masking ratio as long as sufficient amount of input information is available (small totally masked region).

Masking Ratio	Totally Masked	Vimeo-90K		
		Fast	Medium	Slow
0.3	0.81%	0.9058	0.9139	0.8944
0.5	6.25%	0.9067	0.9131	0.8928
0.7	24.01%	0.9068	0.9128	0.8914
0.9	65.61%	0.9002	0.9062	0.8851

(c) **Pretrain epochs.** Our framework does not require long pretraining epochs to learn representative features. The finetuning results for our 50-epoch pretrained model are similar to those for the model initialized from 200-epoch pretraining for all three test sequences.

Epochs	Vimeo-90K		
	Fast	Medium	Slow
0	0.8682	0.8820	0.8641
50	0.9053	0.9125	0.8918
100	0.9056	0.9130	0.8928
200	0.9067	0.9131	0.8928

(d) **Model size.** Our method improves the performance across all models, regardless of the number of parameters (Params), particularly for larger networks.

Embed Dim.	Params (M)	Method	Vimeo-90K		
			Fast	Medium	Slow
32	0.77	Scratch	0.8623	0.8668	0.8498
		Ours	**0.8816**	**0.8924**	**0.8735**
64	2.18	Scratch	0.8647	0.8783	0.8597
		Ours	**0.9009**	**0.9094**	**0.8885**
96	4.49	Scratch	0.8682	0.8820	0.8641
		Ours	**0.9067**	**0.9131**	**0.8928**

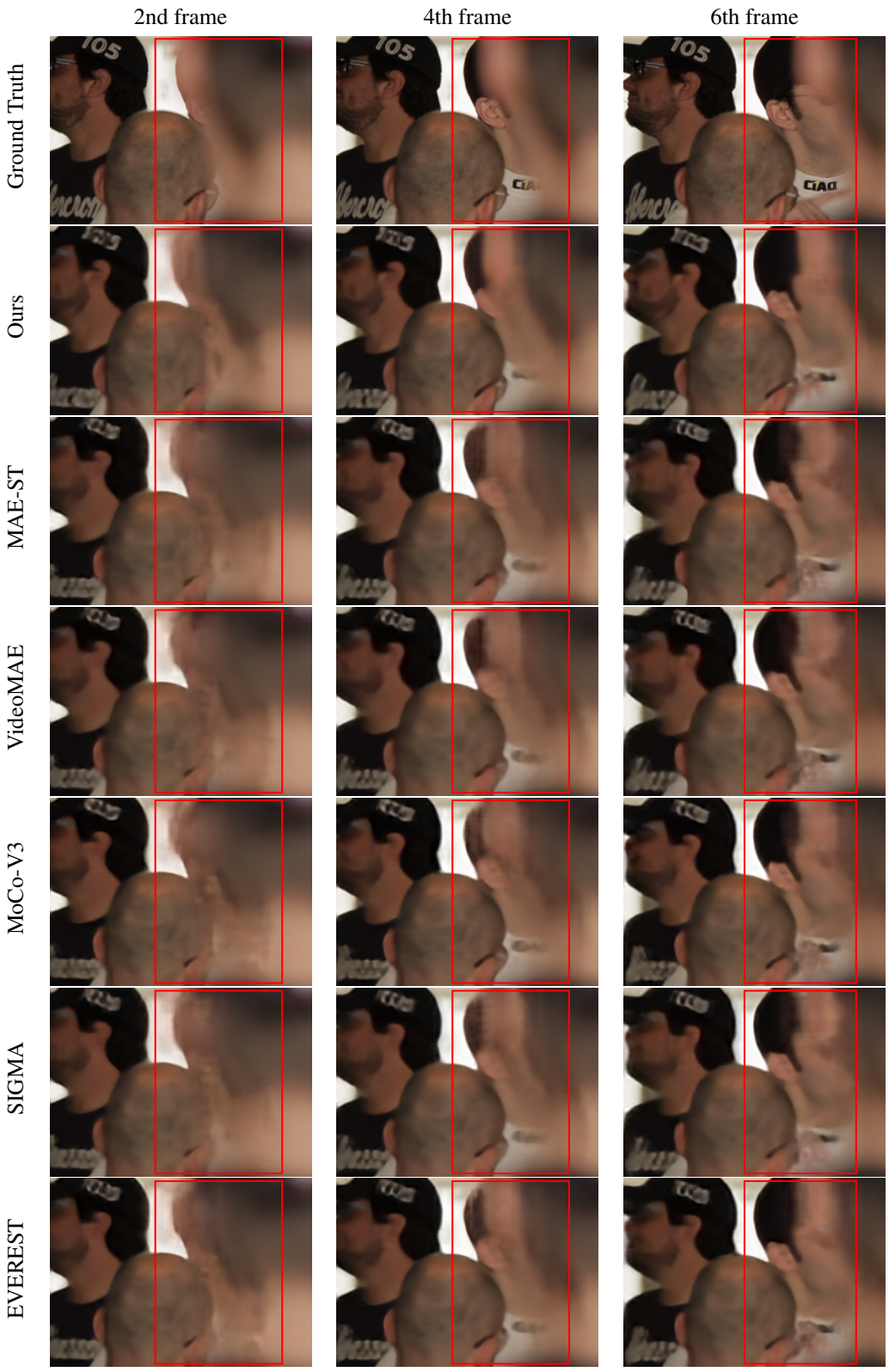

Figure 7. **Visualization results for finetuned networks initialized with different REDS [42] pretrained models.** RSTT-S [18] initialized with our pretraining method shows higher visual quality in scenes with occlusion (see man's hair color, back of the head, and glasses in the red boxed area). The three frames are upsampled in both temporal and spatial dimensions (2nd, 4th and 6th frames in the 7 output frames).

Figure 8. **Visualization results for finetuned models initialized with different pretraining methods.** RSTT-S [18] initialized with our pretraining method shows higher visual quality in scenes with occlusion (see man's cheek and nose in the red boxed area). The three visualized frames are 2nd, 4th and 6th frames from the 7-frame output, where both frame interpolation and super-resolution are performed.

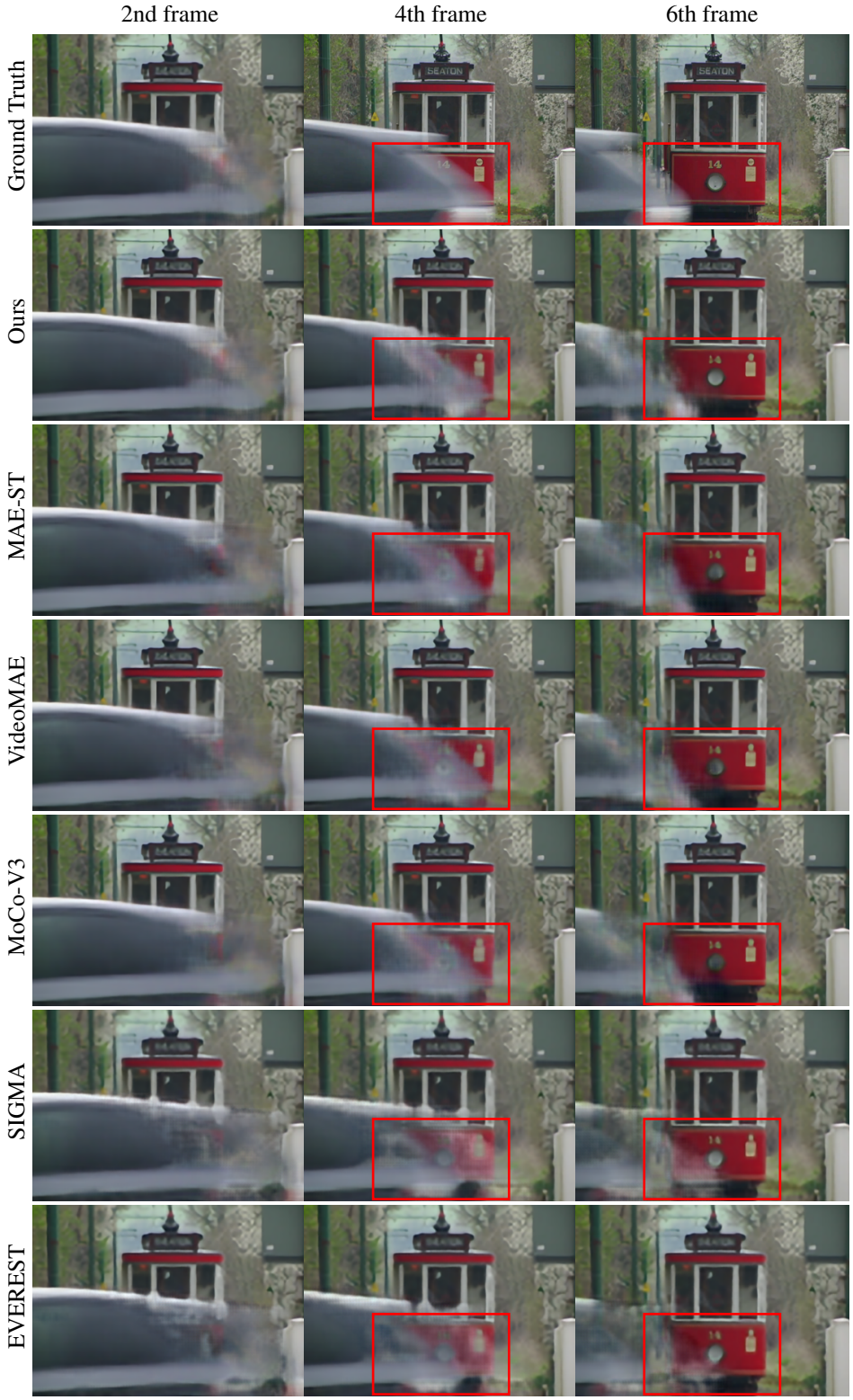

Figure 9. **Visualization results for finetuned models initialized with different pretraining methods.** RSTT-S [18] initialized with our pretraining method shows higher visual quality in scenes with occlusion (see the lower part of the tram in the red boxed area). The three visualized frames are 2th, 4th and 6th frames from the 7-frame output, where both frame interpolation and super-resolution are performed.