DynaVSR: Dynamic Adaptive Blind Video Super-Resolution

Suyoung Lee* Myungsub Choi* Kyoung Mu Lee
ASRI, Department of ECE, Seoul National University
{esw0116, cms6539, kyoungmu}@snu.ac.kr

Abstract

*Most conventional supervised super-resolution (SR) algorithms assume that low-resolution (LR) data is obtained by downscaling high-resolution (HR) data with a fixed known kernel, but such an assumption often does not hold in real scenarios. Some recent blind SR algorithms have been proposed to estimate different downscaling kernels for each input LR image. However, they suffer from heavy computational overhead, making them infeasible for direct application to videos. In this work, we present **DynaVSR**, a novel meta-learning-based framework for real-world video SR that enables efficient downscaling model estimation and adaptation to the current input. Specifically, we train a multi-frame downscaling module with various types of synthetic blur kernels, which is seamlessly combined with a video SR network for input-aware adaptation. Experimental results show that DynaVSR consistently improves the performance of the state-of-the-art video SR models by a large margin, with an order of magnitude faster inference time compared to the existing blind SR approaches.*

1. Introduction

Widespread usage of high-resolution (HR) displays in our everyday life has led to increasing popularity of super-resolution (SR) technology, which allows for enhancing the resolution of visual contents from low-resolution (LR) inputs. Recent advances in deep-learning-based SR approaches for images [7, 8, 12, 18, 19, 20, 22, 27, 41] and videos [5, 16, 17, 31, 35] are driving this trend, showing excellent performance on public SR benchmarks [1, 4, 14, 24, 26, 38]. However, majority of the models are trained under the assumption that the LR images are downscaled from the ground truth HR images with a *fixed known kernel*, such as MATLAB bicubic. It has been shown in Shocher *et al.* [29] that SR performance of the existing models significantly deteriorates if test images do not match such training settings.

SR problem focusing on real-world scenarios with unknown downscaling kernels is called *blind* SR. Numerous methods have been proposed to accurately estimate image-specific kernels for each input [3, 11, 25]. These methods, however, require training the estimation network from scratch at inference time and typically runs in minutes [3, 6], or sometimes up to an hour [25] to handle a single image. Such heavy computational overhead makes the existing approaches impractical to run on a frame-by-frame basis for video SR, as even a short video clip typically contains over hundreds of frames.

Note that real-world LR video frames contain various different types of degradations, including spatial downsampling and motion blurs. To solve this problem by learning, we need to collect enough training data for all kinds of degradations, which is computationally infeasible. However, it can be greatly alleviated if we could effectively estimate the characteristics of the current input video, and build an adaptive model that can adjust its parameters at test time.

In this work, we propose an efficient framework for blind video SR named **DynaVSR** that can flexibly adapt to dynamic input videos. Our proposed framework is based on novel downscaling kernel estimation and input-aware adaptation by meta-learning. It first estimates an approximate downscaling process given input LR video sequences, and generates further downscaled version of the LR frames, which we call *Super LR*, or SLR in short. Then, using the constructed SLR-LR pairs, the parameters of the video SR (VSR) network as well as the downscaling network are jointly updated. The final output HR video is obtained by inference through the VSR model with the parameters adapted to the input LR video. The HR output predicted by DynaVSR for a real-world example sequence is shown in Figure 1, compared to the recent blind SR methods [3, 11, 15]. We observe that DynaVSR greatly improves the output quality upon the VSR baselines that are trained with bicubic downsampled data, and shows more visually pleasing results than the existing approaches, even with significantly faster running time (see Sec. 5.3.1).

Overall, our contributions are summarized as follows:

- We propose DynaVSR, a novel adaptive framework for real-world VSR that combines the estimation of the

*indicates equal contribution.

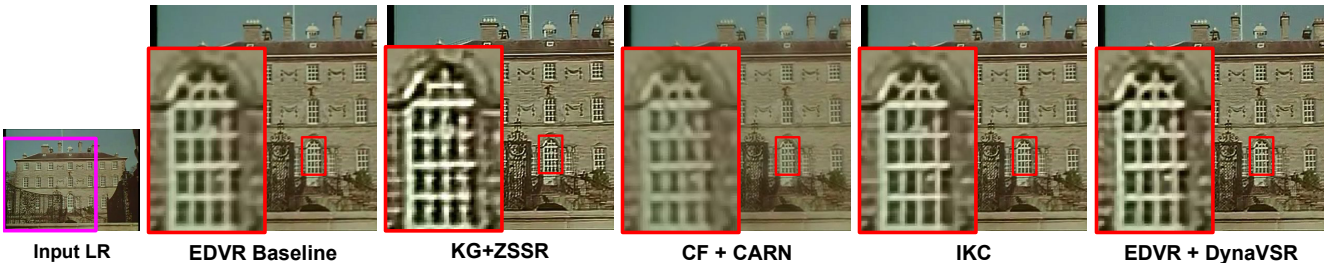


Figure 1: Sample results of the proposed DynaVSR on *real* video with unknown degradation kernels. The state-of-the-art video SR model (EDVR [35]) and recent blind SR models (KG (KernelGAN) [3], CF (CorrectionFilter) [15], and IKC [11]) either show blurry outputs or generate unpleasing artifacts, while DynaVSR shows a much clearer result.

unknown downscaling process with test-time adaptation via meta-learning.

- We greatly reduce the computational complexity of estimating the real downscaling process, thereby enabling real-time execution of SR in videos.
- DynaVSR is generic and can be applied to any existing VSR model for consistently improved performance.

2. Related Works

While the field of super-resolution (SR) has a long history, in this section, we concentrate on more relevant deep-learning-based approaches and review recent adaptive methods applicable in real-world scenarios.

2.1. Single-Image SR (SISR)

Since Dong *et al.* [8] (SRCNN) have shown that a deep learning approach can substantially outperform previous optimization based approaches [4, 33, 34, 37, 38], great advances have been made in SISR including VDSR [18], EDSR [22], ESRGAN [21], and others [2, 7, 12, 19, 20, 27, 40, 41]. However, despite huge performance boost, many works are limited to only performing well on LR images downscaled by a fixed kernel such as bicubic, and otherwise produce undesirable artifacts.

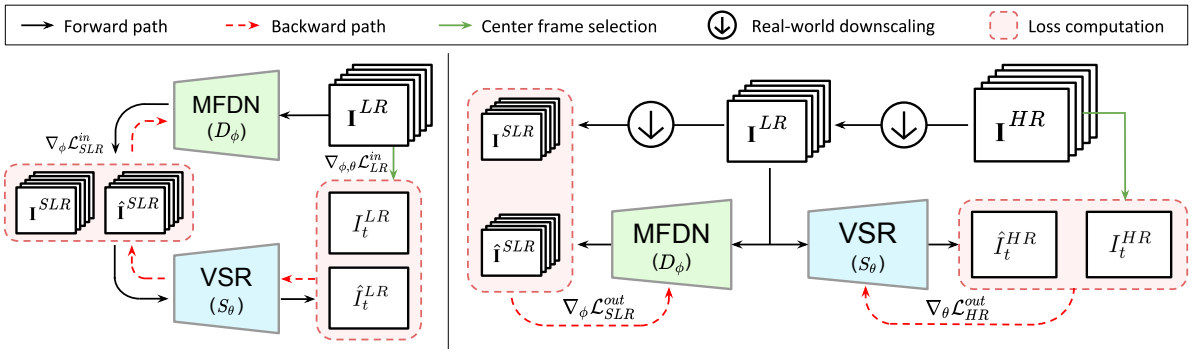
To overcome this issue, several approaches train SR networks which are applicable to multiple types of degradations, assuming that we already know the degradation kernel (*a.k.a.* non-blind SR). SRMD [39] used the LR image and its corresponding degradation kernel as the model inputs to generate high-quality HR images. ZSSR [29] instead apply the same kernel used to generate the LR image to make a smaller LR image, then train an image-specific network. Park *et al.* [28] and Soh *et al.* [30] greatly reduce the time required for input-aware adaptation by incorporating meta-learning. All methods, however, cannot perform well unless we know the exact downscaling kernel, which is unavailable in real-world cases. To this end, numerous *blind* SR methods have been proposed.

Blind SR methods first estimate the unknown kernels in a self-supervised manner, and then apply the predicted kernels to the non-blind SR models. Existing kernel estimation approaches either exploit self-similarity [10, 14, 42] (with the hypothesis that similar patterns and structures across different scales appear in natural images) or design an iterative self-correction scheme [11, 15]. Michaeli and Irani [25] first propose to estimate the downscaling kernel by exploiting the patch-recurrence property of a single image, which is further improved in KernelGAN [3] by utilizing the Internal-GAN [29]. IKC [11] introduce an iterative correction scheme and successfully generates high-quality SR images. Hussein *et al.* [15] also correct the downscaling kernel for many iterations, and use the final kernel in the same way as their non-blind settings.

Since most of the aforementioned methods require training the model from scratch to estimate an unknown kernel, they suffer from heavy computational overhead at test time. IKC does not need training at inference, but still requires many iterations for refining its initial output. On the other hand, our proposed framework directly integrates the input-aware kernel estimation process with video SR models and achieves better results with faster running time, enabling practical application of blind SR techniques to videos.

2.2. Video SR (VSR)

Video SR is different from SISR in that the input frames contain temporal information. Kappeler *et al.* [17] first propose a convolutional neural network (CNN) based VSR method by allowing the network input to be a sequence of frames. Caballero *et al.* [5] and Tao *et al.* [31] incorporate optical flow estimation models to explicitly account for the motion between neighboring frames. TOFlow [36] introduce task-oriented flow, a computationally lighter flow estimation module that is applicable to various video processing tasks. Since the flow-based methods are highly dependent on the motion estimation accuracy, DUF [16] propose the dynamic upsampling filter network, avoiding explicit calculation of the motion information. EDVR [35] also handles motion implicitly with a unified framework, including the Pyramid, Cascading and Deformable convolution (PCD) alignment and the Temporal and Spatial Attention (TSA) fusion processes. In this work, we use TOFlow,



(a) Inner loop update (Train, Test) (b) Outer loop update (Train)

Figure 2: Overall training procedure for the proposed DynaVSR framework. (a) Both MFDN and VSR network are jointly updated in the inner loop. (b) The base parameters, ϕ and θ, are separately updated in the outer loop.

DUF, and EDVR as the baseline VSR models and show how DynaVSR can consistently improve the performance of those models.

3. Preliminary on MAML

Before diving into our main framework, we briefly summarize model-agnostic meta-learning (MAML) [9] algorithm that we use for test-time adaptation. The goal of meta-learning is to rapidly adapt to novel tasks with only few examples. For MAML, the adaptation process is modeled with a few gradient updates to the parameters.

Specifically, MAML first samples a set of examples $\mathcal{D}_{\mathcal{T}_i}$ from the current task $\mathcal{T}_i \sim p(\mathcal{T})$, where $p(\mathcal{T})$ denotes a distribution of tasks. Then, adaptation to the current task is done by fine-tuning the model parameters θ:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(S_\theta(\mathcal{D}_{\mathcal{T}_i})), \tag{1}$$

where $\mathcal{L}_{\mathcal{T}_i}$ is a loss function, and S_θ can be any parameterized model. After adapting to each task $\mathcal{T}_i$, new examples $\mathcal{D}'_{\mathcal{T}_i}$ are sampled from the same task to test the generalization capability and update the base parameters:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(S_{\theta_i'}(\mathcal{D}'_{\mathcal{T}_i})). \tag{2}$$

Note that Eq. 1, inner loop update, is performed both at training and inference, while the outer loop update (Eq. 2, *a.k.a. meta-update*) is only executed during training.

4. Dynamic Adaptive Blind VSR Framework

In this section, we first summarize the overall framework and the problem formulation, and describe in detail how meta-learning is integrated for efficient adaptation.

4.1. Overall Framework

For blind VSR application, we define a *task* as super-resolving each input video sequence. Since only the input LR frames are available at test time, we further downscale the input to form *super*-low-resolution (SLR) frames. Then, we can update the model by making it predict the LR frames well given the SLR frames. The resulting *adapted* parameters perform especially well on the current inputs, generating high-quality HR frames given the LR inputs.

In the *blind* SR setting, each LR input may have come through a different downscaling process, therefore test-time adaptation is crucial. For video application, real-time execution of estimating the downscaling process is also critical. Thus, we introduce an efficient Multi-Frame Downscaling Network (MFDN), and combine it with the VSR network. The proposed framework is named as **DynaVSR**, as it can adapt well to each of the dynamically-varying input videos.

Figure 2 illustrates the overall training process of DynaVSR, which consists of three stages: 1) estimation of the unknown downscaling process with MFDN, 2) joint adaptation of MFDN and VSR network parameters *w.r.t.* each input video, and 3) meta-updating the base parameters for MFDN and VSR network. At test time, only 1) and 2) are processed, and the updated parameters of the VSR network is used to generate the final super-resolved images. The detailed training and test processes are described in Sec. 4.4.

4.2. Blind VSR Problem Formulation

The goal of VSR is to accurately predict the HR frames $\left\{\hat{I}_t^{HR}\right\}$ given the input LR frames $\left\{I_t^{LR}\right\}$, where t denotes the time step. In practice, many recent models such as [16, 32, 35] estimate the *center* HR frame $\hat{I}_t^{HR}$ given the surrounding $(2N+1)$ LR frames $I_{t\in\mathbb{T}}^{LR}$, where $\mathbb{T} = \{t-N, \cdots, t+N\}$, and generate the HR sequence in a sliding window fashion. Thus, VSR problem for a single time step t can be formulated as:

$$\hat{I}_t^{HR} = S_\theta\left(I_{t\in\mathbb{T}}^{LR}\right), \tag{3}$$

In a fixed-kernel SR setting, a large number of training pairs is available since $\left\{I_t^{LR}\right\}$ can be easily obtained

by applying the designated downscaling kernel to $\left\{I_t^{HR}\right\}$. When tackling *blind* SR, however, the downscaling process is unknown and acquiring a large training set becomes impractical. As previously studied in KernelGAN [3], correct estimation of the downscaling process is crucial to the SR performance. This can be formalized as:

$$\hat{I}_t^{LR} = D_\phi\left(I_t^{HR}\right), \tag{4}$$

where D_ϕ denotes a downscaling model parameterized by ϕ. Existing blind SR approaches typically find a good D_ϕ by learning ϕ in a self-supervised manner. Then, using D_ϕ that is optimized to the current inputs, the final SR results are obtained in the same way as a non-blind SR setting.

When it comes to blind *video* SR, efficiency becomes the key issue, since videos may contain several hundreds and thousands of frames. Existing blind *image* SR techniques require long processing time for finding the downscaling model D_ϕ, and are therefore computationally infeasible (see Sec. 5.3.1 for runtime comparison). To this end, we design a new light-weight model, named Multi-Frame Downscaling Network (MFDN), for effective estimation of the downscaling process in real-time.

4.3. Multi-Frame Downscaling Network (MFDN)

Though video clips at different time steps can be affected by various different types of degradations (*e.g.* motion blurs, noises), in this work, we primarily focus on the *downscaling* process. Consequently, we assume that each LR frame I_t^{LR} is generated from the corresponding HR frame I_t^{HR} following the same but unknown downscaling process within a single video sequence.

To model D_ϕ in Eq. 4, we propose MFDN that receives a multi-frame LR inputs and produces the corresponding **further downscaled** multi-frame outputs. This process is formulated as:

$$\hat{I}_{t\in\mathbb{T}}^{SLR} = D_\phi\left(I_{t\in\mathbb{T}}^{LR}\right), \tag{5}$$

where $I_{t\in\mathbb{T}}^{LR}$ is an input LR sequence, and $\hat{I}_{t\in\mathbb{T}}^{SLR}$ denotes an estimated *Super LR* (SLR) sequence which is a further downscaled version. To model various kernels *w.r.t.* different inputs while maintaining efficiency, we model MFDN with a 7-layer CNN including non-linearities. For handling multi-frame information, 3-D convolutions are used for the first and the last layers of MFDN, and 2-D convolution layers for the rest. Contrary to the existing methods [3, 15], MFDN does not require additional training at test time. This greatly improves the efficiency in estimating the unknown downscaling process and thereby enables application to computation-heavy problems like VSR.

To accurately predict the SLR frames for diverse cases with a single discriminative model, MFDN is first pretrained with various synthetic kernels and later employed to our meta-learning framework for further adaptation to each input. The training LR-SLR pairs are generated by random sampling of the anisotropic Gaussian kernels. Note that, the LR frames $I_{t\in\mathbb{T}}^{LR}$ are themselves generated from the ground truth HR frames $I_{t\in\mathbb{T}}^{HR}$ by applying randomly selected kernel in the synthetic kernel set. The corresponding SLR frames $I_{t\in\mathbb{T}}^{SLR}$ are then generated from LR with the same kernel. Pre-training MFDN is done by minimizing the pixel-wise loss between $I_{t\in\mathbb{T}}^{SLR}$ and the estimated output $\hat{I}_{t\in\mathbb{T}}^{SLR}$.

After pretraining MFDN, it is further fine-tuned during the meta-training process to be readily adaptable to each input. Though we only use synthetic kernels for training MFDN, it can generalize to the real inputs reasonably well, as we show in the experiments (see Sec. 5.4). Further analysis on the effects of MFDN is shown in Sec. 5.5.1, where we compare it to the other downscaling methods.

4.4. Meta-Learning for Blind VSR

4.4.1 Meta-training

For the inner loop update, we first generate $\hat{I}_{t\in\mathbb{T}}^{SLR}$ with MFDN using Eq. (5). The generated SLR sequence is then fed into the VSR network as the input:

$$\hat{I}_t^{LR} = S_\theta\left(\hat{I}_{t\in\mathbb{T}}^{SLR}\right) = S_\theta\left(D_\phi\left(I_{t\in\mathbb{T}}^{LR}\right)\right). \tag{6}$$

We introduce two loss terms to update ϕ and θ: LR fidelity loss ($\mathcal{L}_{LR}^{in}$) and SLR guidance loss ($\mathcal{L}_{SLR}^{in}$). The LR fidelity loss ($\mathcal{L}_{LR}^{in}$) indicates the difference between $\hat{I}_t^{LR}$ and I_t^{LR}, and we match the type of loss function used for each backbone VSR network (denoted as L_{VSR}). However, $\mathcal{L}_{LR}^{in}$ alone cannot guarantee that the updated MFDN would produce the correct SLR frames. Inaccurate downscaling estimation can generate erroneous SLR frames, and can also give wrong update signals to the VSR network. To cope with this issue, SLR guidance loss ($\mathcal{L}_{SLR}^{in}$) is proposed to make sure that MFDN outputs do not move far away from the actual SLR frames. In practice, $\mathcal{L}_{SLR}^{in}$ is calculated as the ℓ_1 distance between generated SLR frames $\hat{I}_{t\in\mathbb{T}}^{SLR}$ and the ground truth $I_{t\in\mathbb{T}}^{SLR}$. The total loss for the inner loop update is computed as a sum of the two terms:

$$\mathcal{L}^{in} = \mathcal{L}_{LR}^{in} + \mathcal{L}_{SLR}^{in} \tag{7}$$

$$= L_{VSR}\left(\hat{I}_t^{LR}, I_t^{LR}\right) + \left\|\hat{I}_{t\in\mathbb{T}}^{SLR} - I_{t\in\mathbb{T}}^{SLR}\right\|_1. \tag{8}$$

This process corresponds to the left part of Figure 2.

For the outer loop, the *base* parameter values of ϕ and θ (before inner loop updates) are adjusted to make the models more adaptive to new inputs. Given the input LR sequence $I_{t\in\mathbb{T}}^{LR}$, VSR network and MFDN generate the HR and SLR predictions, correspondingly, as follows:

$$\hat{I}_t^{HR} = S_{\theta'}\left(I_{t\in\mathbb{T}}^{LR}\right), \quad \hat{I}_{t\in\mathbb{T}}^{SLR} = D_{\phi'}\left(I_{t\in\mathbb{T}}^{LR}\right). \tag{9}$$

Algorithm 1: DynaVSR training

Require: $p(\mathcal{T})$: uniform distribution over videos
Require: α, β: inner / outer-loop learning rates

Initialize parameters θ and ϕ
while *not converged* **do**
  Sample a batch of sequences $\mathcal{T}_i \sim p(\mathcal{T})$
  foreach *i* **do**
    Generate $\left\{I_t^{HR}\right\}_i, \left\{I_t^{LR}\right\}_i, \left\{I_t^{SLR}\right\}_i$ from $\mathcal{T}_i$ using random synthetic kernels
    Generate $\left\{\hat{I}_t^{SLR}\right\}_i$ using Eq. (5)
    Calculate $\nabla_{\phi,\theta}\mathcal{L}^{in}$ using Eq. (8)
    Compute adapted parameters ϕ' and θ' with:
    $\phi' = \phi - \alpha\nabla_\phi\mathcal{L}^{in}, \;\; \theta' = \theta - \alpha\nabla_\theta\mathcal{L}^{in}$
    Save $\left\{I_t^{HR}\right\}_i, \left\{I_t^{SLR}\right\}_i$ for meta-update
  end
  Update $\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{HR}^{out}$ using Eq. (10)
  Update $\phi \leftarrow \phi - \beta\nabla_\phi \sum_{\mathcal{T}_i} \mathcal{L}_{SLR}^{out}$ using Eq. (11)
end

From the two predictions, we can define the two loss terms:

$$\mathcal{L}_{HR}^{out} = L_{VSR}\left(\hat{I}_t^{HR}, I_t^{HR}\right), \tag{10}$$

$$\mathcal{L}_{SLR}^{out} = \left\|\hat{I}_{t\in\mathbb{T}}^{SLR} - I_{t\in\mathbb{T}}^{SLR}\right\|_1, \tag{11}$$

where each loss is used to update the parameters in corresponding networks. Note that the loss is calculated with updated parameters, $D_{\phi'}$ and $S_{\theta'}$, but the gradient is calculated *w.r.t.* ϕ and θ, respectively. The right part of Figure 2 depicts the outer update mechanism.

Algorithm 1 summarizes the full procedure for training DynaVSR. Compared to the existing blind SISR approaches, the proposed algorithm has multiple advantages: 1) DynaVSR does not require a necessary number of iterations as a hyperparameter, achieving maximum performance with only a single gradient update, leading to improved computational efficiency compared to IKC [11] or CorrectionFilter [15]; 2) DynaVSR is generic and can be applied to any existing VSR models, while the other methods need specific SR network architectures; 3) DynaVSR can handle multiple frames as inputs for video application.

4.4.2 Meta-test

At test time, only the inner loop update is performed to adapt the MFDN and VSR network parameters to the test input frame sequence. Since there are no ground truth (GT) SLR frames, we replace it with the SLR frames predicted by our pretrained MFDN. Although we do not use the real GT SLR frames, we show in experiments that the pseudo GT frames generated by MFDN are still valid (see Sec. 5.3).

The final output HR frame $\hat{I}_t^{HR}$ can then be generated using the updated VSR network.

5. Experiments

5.1. Dataset

We use three popular VSR datasets for our experiments: REDS [26], Vimeo-90K [36], and Vid4 [23]. Also, many low-resolution videos are gathered from YouTube to demonstrate the performance of DynaVSR on real-world scenarios. **REDS** dataset is composed of 270 videos each consisting of 100 frames, and each frame has 1280 $\times$ 720 spatial resolution. Out of 270 train-validation videos, we use 266 sequences for training and the other 4 sequences (REDS-val) for testing, following the experimental settings in Wang *et al.* [35] (EDVR). Videos from REDS dataset typically contain large and irregular motion, which makes it challenging for VSR. **Vimeo-90K** dataset contains 91,707 short video clips, each containing 7 frames. We use Vimeo-90K only for training, using the training split of 64,612 clips. Although the resolution of each frame is low (448 $\times$ 256), Vimeo-90K is one of the most frequently used dataset for training VSR models. **Vid4** dataset is widely used for evaluation purposes only; many previous works [13, 32, 35] train their models with Vimeo-90K and report their performance on Vid4 dataset, and we follow the same setting.

5.2. Implementation Details

DynaVSR can be applied to any deep-learning-based VSR model, and we show its effectiveness using EDVR [35], DUF [16], and TOFlow [36] as backbone VSR networks. All models are initialized with pretrained parameters for scale factor $s = 2$, with a known downscaling process, MATLAB bicubic downsampling with antialiasing. Separate models are trained for each training dataset, Vimeo-90K and REDS. We denote these pretrained models as (bicubic) ***Baseline***, and report their performance to show how existing approaches using this ideal downscaling kernel fail in synthetic and real-world settings.

When pretraining MFDN and meta-training DynaVSR, diverse kinds of downscaling kernels are used to generate the HR-LR-SLR patches for each iteration. Specifically, we select $\sigma_1, \sigma_2 \in \mathcal{U}[0.2, 2.0]$, and $\theta \in \mathcal{U}[-\pi, \pi]$ independently for randomly generating many anisotropic Gaussian kernels. More details are described in supplemantary slides. The source code along with our pretrained models is made public to facilitate reproduction and further research.[1]

5.3. Quantitative Results

We thoroughly evaluate the performance improvements of DynaVSR *w.r.t.* the bicubic baselines in three different kinds of synthetic blur kernels: isotropic Gaussian,

[1] https://github.com/esw0116/DynaVSR

Table 1: **Quantitative results, running time comparison for meta-training with recent VSR models and blind SISR methods**. We evaluate the benefits of DynaVSR algorithm on Vid4 [23] and REDS-val [26] dataset. Performance is measured in PSNR (dB). Red denotes the best performance, and blue denotes the second best. The right part indicates the running time to make a single HD frame. DynaVSR shows the shortest time among the existing algorithms on all three VSR baselines.

	Method		Vid4 [23]			REDS-val [26]			Time (s)		
			Iso.	Aniso.	Mixed	Iso.	Aniso.	Mixed	Preprocessing	Super-resolution	Total
Blind SISR	KG [3] + ZSSR [29]		25.92	24.36	22.62	29.05	27.30	25.23	58.96	92.07	151.03
	CF [15] + DBPN [12]*		27.30	25.61	24.03	30.37	29.30	28.02	664.77	0.03	664.78
	CF [15] + CARN [2]*		27.95	26.82	25.62	30.98	30.47	29.70	106.11	**0.02**	106.13
	IKC [11]		**29.46**	26.19	27.82	**34.10**	30.11	31.41	-	2.21	2.21
Video SR	EDVR [35]	Baseline	25.35	25.84	26.27	29.14	29.66	30.21	-	0.40	0.40
		DynaVSR	**28.72**	**28.81**	**29.25**	**32.45**	**33.41**	**33.67**	0.88	0.40	**1.28**
	DUF [16]	Baseline	25.26	25.70	26.47	29.01	29.38	30.14	-	1.00	1.00
		DynaVSR	27.43	**27.54**	27.77	31.23	31.29	31.36	**0.30**	1.00	1.30
	TOFlow [36]	Baseline	25.27	25.66	26.48	29.04	29.46	30.40	-	0.98	0.98
		DynaVSR	27.16	27.07	27.69	31.50	**31.56**	**31.87**	0.96	0.98	1.94
	Average PSNR gain		**+2.48**	**+2.07**	**+1.83**	**+2.66**	**+2.59**	**+2.05**	-	-	-

anisotropic Gaussian, and mixed Gaussian. For all experiments in this section, we report the standard peak signal-to-noise ratio (PSNR).

For ***isotropic*** Gaussians, we adapt the *Gaussian8* setting from Gu *et al.* [11], which consists of eight isotropic Gaussian kernels of $\sigma \in [0.8, 1.6]$ for scale factor 2, originally proposed for evaluating blind image SR methods. The HR image is first blurred by the Gaussian kernel and then downsampled by bicubic interpolation. Since it is unclear from Gu *et al.* [11] how to handle the boundary σ-s, we evaluate on nine different kernel widths including both $\sigma = 0.8$ and $\sigma = 1.6$ with step size 0.1. However, isotropic Gaussian kernels are insufficient to represent various types of degradations in the real world. Thus, we also evaluate DynaVSR on ***anistropic*** Gaussian settings, where we fix the Gaussian kernel widths to the boundary values of *Gaussian8* so that $(\sigma_x, \sigma_y) = (0.8, 1.6)$. Evaluation is done on 4 cases with different rotations ($0°, 45°, 90°$, and $135°$), and the average performance is reported. Lastly, we introduce a ***mixed*** setting which consists of randomly generated kernels. Each sequence is individually downscaled by random Gaussian kernels with $\sigma_1, \sigma_2 \in \mathcal{U}[0.2, 2.0]$ and $\theta \in \mathcal{U}[-\pi, \pi]$ with direct downsampling. Note that the sampled downscaling kernel is kept same for the entire sequence.

Table 1 compares the results of DynaVSR with its baselines and other blind SR methods. Compared to the bicubic baseline, DynaVSR consistently improves the performance over all evaluation settings by a large margin (over 2dB on average). This proves the effectiveness of adaptation via meta-training, since we use the same architecture without introducing any additional parameters. Compared to blind SISR models, DynaVSR with any baseline VSR network performs favorably against existing methods in general. IKC performs well in isotropic Gaussian settings, but DynaVSR with EDVR ranks the second while greatly outperforming IKC for the other evaluation settings. Note that, while IKC is specifically optimized with isotropic Gaussian kernels only, the reported performance for DynaVSR is from our final model trained also with various other kernels including anisotropic and rotated Gaussians.

5.3.1 Time Complexity Analysis

The right part of Table 1 demonstrates the running time for generating a single HD resolution (1280×720) frame from a $\times 2$ downscaled LR frame sequence *w.r.t* each blind SR method. *Preprocessing* indicates the steps required to prepare the input LR frames for putting through the SR network, which may include kernel estimation (KG [3]) or iterative correction of the inputs to modify their characteristics (CF [15]). For IKC [11], it is difficult to explicitly separate each step, so we include the runtime for iterative correction to *Super-Resolution* category, which reports the inference time for each SR network.

Since MFDN is highly efficient, DynaVSR shows much faster preprocessing time compared to existing blind SISR models. Recent approaches, KG [3] and CF [15], require minutes of preprocessing time because both models need to train from scratch at test time, which is very expensive even with a small network. On the other hand, DynaVSR

*Since CF require too much time for preprocessing, we report the performance for random 10% of the validation set. For fair comparison, we show the results of the other models on the same 10% validation set in the supplementary document. We can observe that the overall trend in performance is almost same as the full evaluation.

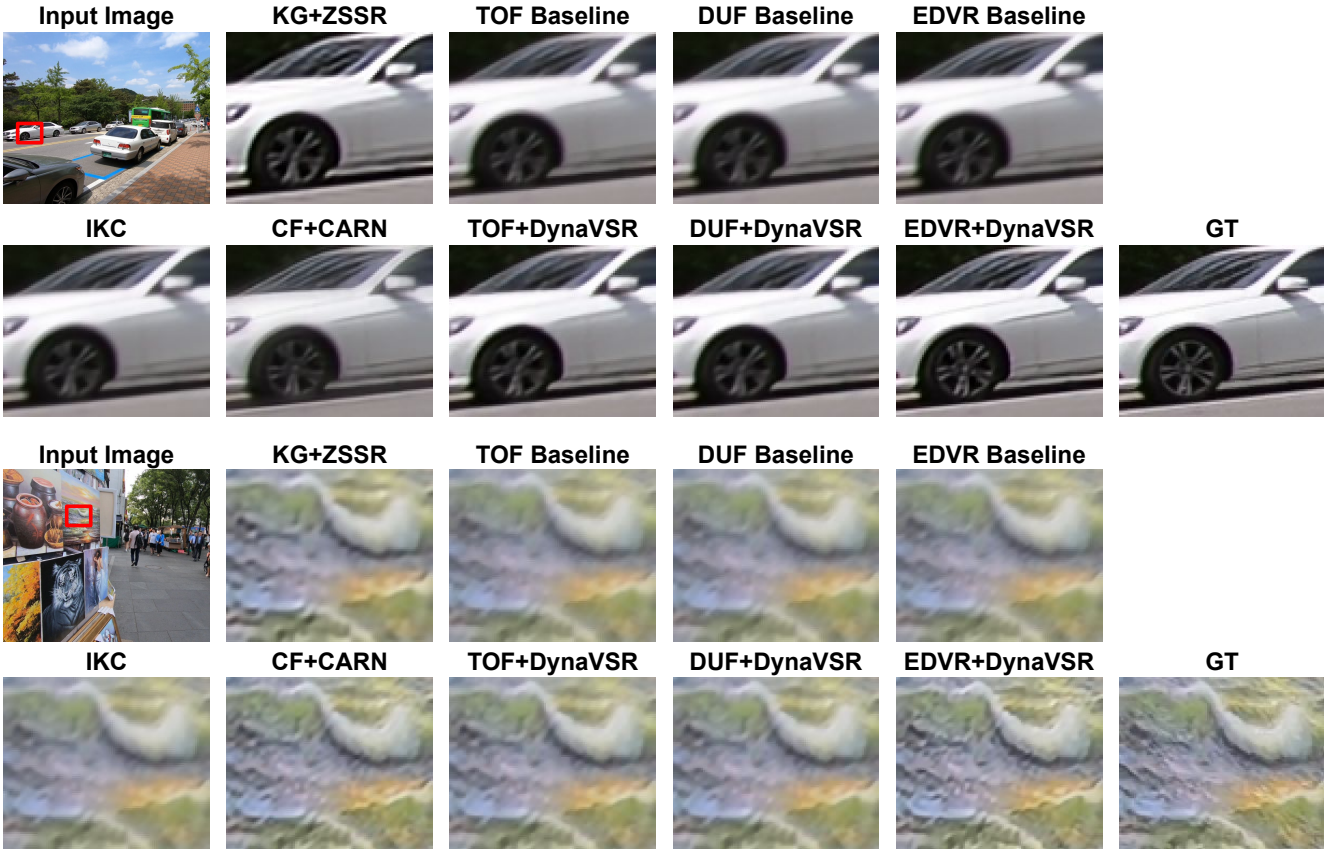


Figure 3: **Qualitative results on REDS-val [26] dataset.** DynaVSR consistently improves the visual details upon all baseline VSR models, and also produces visually more pleasing outputs compared to recent blind SR approaches.

Table 2: **Quantitative comparison *w.r.t.* different downscaling models.** We evaluate on REDS-val using EDVR baseline. The right 3 columns indicate the SR performance for each downscaling model in our framework, where the joint training with MFDN shows the best results.

Downscaling models	SLR	Isotropic	Anisotropic	Mixed
Bicubic	34.78	31.82	33.26	33.39
KernelGAN [3]	40.84	31.99	33.31	33.48
SFDN	45.36	32.34	33.41	33.63
MFDN	45.71	32.45	33.41	33.67
GT	∞	32.87	33.84	34.27

requires only a single gradient update to the model parameters and successfully reduces the preprocessing time to less than a second (**more than** $\times 60$ **faster than KG, and** $\times 100$ **faster than CF**). Note that, the preprocessing time for DynaVSR is highly dependent on the architecture of the base VSR network, and the efficiency can be further improved by using more light-weight VSR models. IKC reports the shortest runtime among the other previous methods, but it still needs multiple iterations for kernel correction, and EDVR+DynaVSR shows more than 40% shorter runtime.

5.4. Visual Comparison

The qualitative results for REDS-val dataset using random synthetic downscaling kernels are shown in Figure 3. DynaVSR greatly improves the visual quality over all baselines by well adapting to the input frames. Notably, blurry edges from the bicubic baselines are sharpened, and texture details become much clearer. Outputs of DynaVSR also show visually more pleasing results compared to three recent blind SISR models, which are shown in the left part of Figure 3. Results for *real-world* low-quality videos from YouTube, where ground truth frames are unavailable, are illustrated in Figure 1, 4. Although these videos contain various types of unknown degradations, DynaVSR is robust in producing visually pleasing outputs. For results on Vid4 dataset, additional results on REDS-val, and more extensive qualitative analysis on many real-world videos, please check our supplementary materials.

5.5. Analysis

5.5.1 Varying Downscaling Models

To analyze the effects of end-to-end training of VSR model with a downscaling network, we substitute the MFDN part

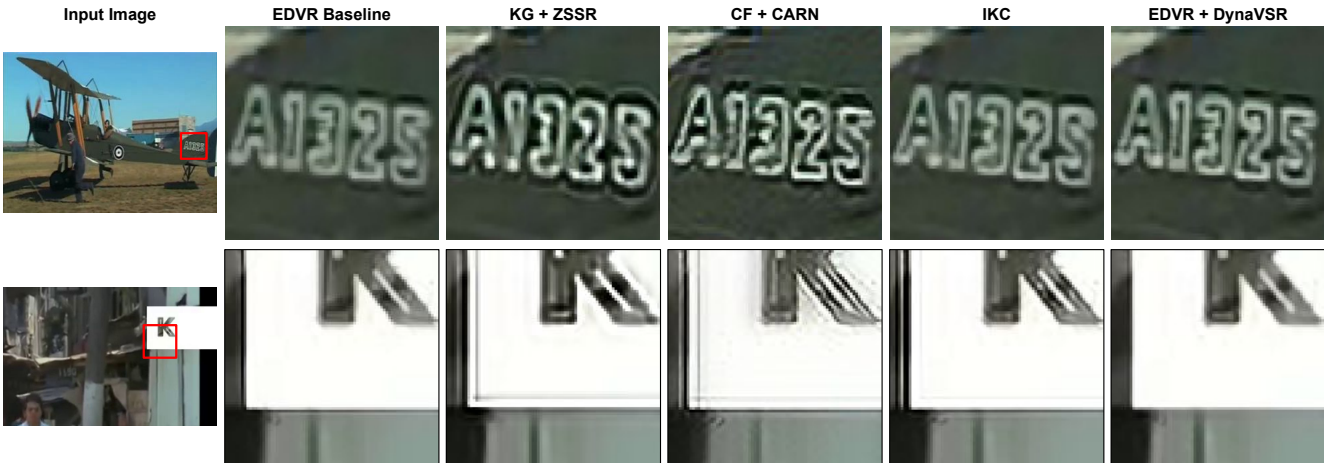


Figure 4: **Qualitative results on *real-world* videos from YouTube** where no ground truth HR frames exist. We use EDVR as the base VSR network. DynaVSR generates cleaner results while the other blind SISR methods show severe artifacts or blurry outputs. Note that, in the second row, all blind SR methods except DynaVSR generate additional boundaries near the corner due to strong ringing artifacts, which does not exist in the original LR frames.

of DynaVSR with four different downscaling models: bicubic downscaling, KernelGAN [3], single-frame variant of MFDN (SFDN, Single Frame Downscaling Network), and the ground truth SLR frame downscaled from the LR frame with the correct kernel. We also evaluate the performance of SLR frame estimation, and report the average PSNR values of VSR network for same settings. The results are summarized in Table 2.

For generating the SLR frames, we first pretrain MFDN and SFDN until convergence. SFDN is a single-frame variant of MFDN with the same number of parameters, which regards the temporal dimension as the same as batch dimension, using only 2-D convolutions. As shown in the leftmost column of Table 2, the MFDN achieves the best performance as a stand-alone downscaler. We believe that it is due to the multi-frame nature of MFDN, since multiple input frames with similar kernels can be observed to make it easier to recognize the actual downscaling patterns.

The final SR performance after jointly training with the VSR model is also more favorable to MFDN, achieving the closest performance to the ideal case of adapting VSR model with GT SLR frames.

5.5.2 Varying the Number of Inner Loop Updates

We also modify the number of inner loop updates and compare the results. Table 3 shows the performance while changing the number of iterations within 1, 3, and 5 steps. The best performance is achieved when we set the number of updates to 1, and more inner loop iterations led to diminishing results on average. We believe this phenomenon is because of overfitting to the input video sequence and forgetting the general super-resolution capability. Although the adaptation to each specific input is a crucial step in blind SR problems, it is only beneficial when the model maintains its generalization performance. Our results show that adaptation with too many inner loop updates drive the model to fall into local optima. How to regularize the inner loop optimization well to circumvent this issue is beyond the scope of this paper and can be an interesting future direction.

Table 3: **Effect of the number of inner loop iterations.** We evaluate on REDS-val using EDVR. Just 1 inner loop update yields the best performance in general.

#	Isotropic	Anisotropic	Mixed
1	32.45	**33.41**	**33.67**
3	**32.75**	32.38	32.96
5	32.17	32.23	32.57

6. Conclusions

In this paper, we propose DynaVSR, a novel adaptive blind video SR framework which seamlessly combines the downscaling kernel estimation model into meta-learning-based test-time adaptation scheme in an end-to-end manner. Compared to existing kernel estimation models for blind SISR, our MFDN extremely improves the computational efficiency and better estimates the downscaling process. Also, the excessive computation needed for input-aware adaptation of network parameters is minimized to a single gradient update by incorporating meta-learning. We demonstrate that DynaVSR gives substantial performance gain regardless of the VSR network architecture in various experimental settings including isotropic and anisotropic Gaussian blur kernels. Furthermore, we empirically show that DynaVSR can be readily applied to real-world videos with unknown downscaling kernels even though it is only trained with synthetic kernels.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshop.*, 2017.

[2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV.*, 2018.

[3] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *NeurIPS.*, 2019.

[4] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC.*, 2012.

[5] Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *CVPR.*, 2017.

[6] Ayan Chakrabarti. A neural approach to blind motion deblurring. In *ECCV.*, 2016.

[7] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *CVPR.*, 2019.

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 2016.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML.*, 2017.

[10] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV.*, 2009.

[11] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. Blind super-resolution with iterative kernel correction. In *CVPR.*, 2019.

[12] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *CVPR.*, 2018.

[13] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *CVPR.*, 2019.

[14] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR.*, 2015.

[15] Shady Abu Hussein, Tom Tirer, and Raja Giryes. Correction filter for single image super-resolution: Robustifying off-the-shelf deep super-resolvers. In *CVPR*, 2020.

[16] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *CVPR.*, 2018.

[17] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Trans. on Computational Imaging*, 2016.

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR.*, 2016.

[19] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR.*, 2016.

[20] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR.*, 2017.

[21] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR.*, 2017.

[22] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshop.*, 2017.

[23] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *TPAMI*, 2013.

[24] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV.*, 2001.

[25] Tomer Michaeli and Michal Irani. Nonparametric blind super-resolution. In *ICCV.*, 2013.

[26] 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 *CVPR Workshop.*, 2019.

[27] Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen. Single image super-resolution via a holistic attention network. In *ECCV.*, 2020.

[28] Seobin Park, Jinsu Yoo, Donghyeon Cho, Jiwon Kim, and Tae Hyun Kim. Fast adaptation to super-resolution networks via meta-learning. *arXiv:2001.02905*, 2020.

[29] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *CVPR.*, 2018.

[30] Jae Woong Soh, Sunwoo Cho, and Nam Ik Cho. Meta-transfer learning for zero-shot super-resolution. In *CVPR.*, 2020.

[31] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *ICCV.*, 2017.

[32] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally-deformable alignment network for video super-resolution. In *CVPR.*, 2020.

[33] Radu Timofte, Vincent De Smet, and Luc Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV.*, 2013.

[34] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV.*, 2014.

[35] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *CVPR Workshop.*, 2019.

[36] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 2019.

[37] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE trans. on Image Processing*, 2010.

[38] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, 2010.

[39] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR.*, 2018.

[40] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV.*, 2018.

[41] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR.*, 2018.

[42] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *CVPR.*, 2011.