Structured Sparsity Learning for Efficient Video Super-Resolution

Bin Xia [1], Jingwen He [2], Yulun Zhang [3], Yitong Wang [4],
Yapeng Tian [5], Wenming Yang [1]*, and Luc Van Gool [3]
[1] Tsinghua University, [2] Shanghai AI Laboratory, [3] ETH Zürich,
[4] ByteDance Inc, [5] University of Texas at Dallas

Abstract

The high computational costs of video super-resolution (VSR) models hinder their deployment on resource-limited devices, e.g.*, smartphones and drones. Existing VSR models contain considerable redundant filters, which drag down the inference efficiency. To prune these unimportant filters, we develop a structured pruning scheme called Structured Sparsity Learning (SSL) according to the properties of VSR. In SSL, we design pruning schemes for several key components in VSR models, including residual blocks, recurrent networks, and upsampling networks. Specifically, we develop a Residual Sparsity Connection (RSC) scheme for residual blocks of recurrent networks to liberate pruning restrictions and preserve the restoration information. For upsampling networks, we design a pixel-shuffle pruning scheme to guarantee the accuracy of feature channel-space conversion. In addition, we observe that pruning error would be amplified as the hidden states propagate along with recurrent networks. To alleviate the issue, we design Temporal Finetuning (TF). Extensive experiments show that SSL can significantly outperform recent methods quantitatively and qualitatively. The code is available at* https://github.com/Zj-BinXia/SSL.

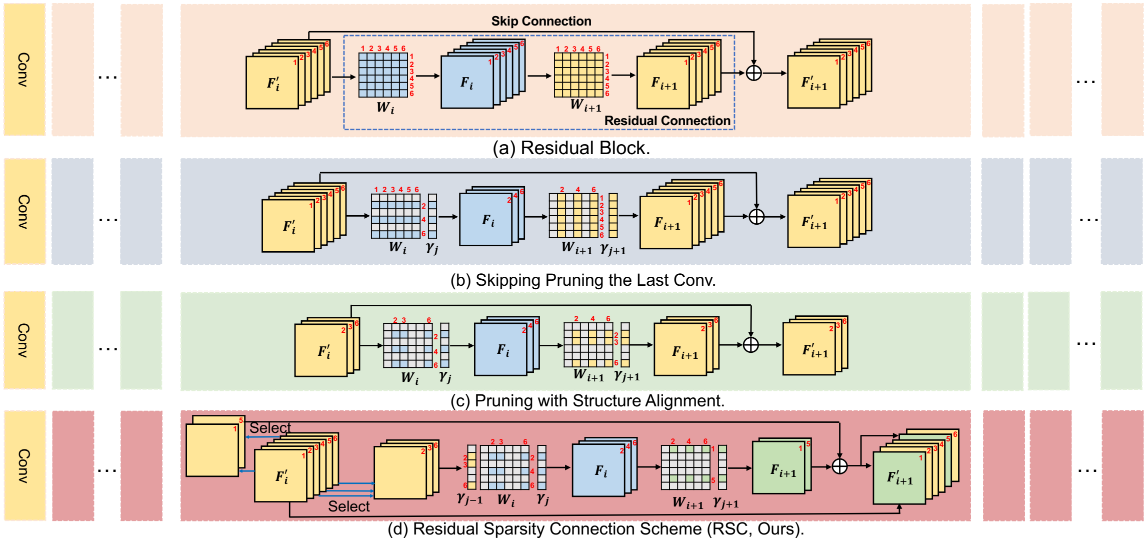
1. Introduction

Video super-resolution (VSR) aims to generate a high-resolution (HR) video from its corresponding low-resolution (LR) observation by filling in missing details. With the popularity of intelligent edge devices such as smartphones and small drones, performing VSR on these devices is in high demand. Although a variety of VSR networks [20, 24, 29, 44, 51] can achieve great performance, these models are usually difficult to be deployed on edge devices with limited computation and memory resources.

To alleviate this issue, we explore a new direction for effective and efficient VSR. To reduce the redundancy of Conv kernels [4, 5, 36, 38] obtaining a more efficient VSR network, we develop a neural network pruning scheme for the VSR task for the first time. Since structured pruning [14, 23, 46, 57] (focusing on filter pruning) can achieve an actual acceleration [41,46] superior to unstructured pruning [11,12] (focusing on weight-element pruning), we adopt structured pruning principle to develop our VSR pruning scheme. Given a powerful VSR network, our pruning scheme can find submodels under presetting pruning rate without significantly compromising performance.

Structured pruning is a general concept, and designing a concrete pruning scheme for VSR networks is challenging. **(1)** Recurrent networks are widely used in VSR models to extract temporal features, consisting of residual blocks (*e.g.*, BasicVSR [2] has 60 residual blocks). However, it is hard to prune the residual blocks because the skip and residual connections ought to share the same indices [23] (Fig. 1 (a)). As shown in Fig. 1 (b), quite a few structured pruning schemes [23,34] do not prune the last Conv layer of the residual blocks, which restricts the pruning space. Recently, as shown in Fig. 1 (c), ASSL [57] and SRPN [58] introduce regularization and prune the same indices on skip and residual connections to keep channel alignment (local pruning scheme,*i.e.*, each layer pruning the same ratio of filters). However, ASSL and SRPN still cannot achieve the potential of pruning residual blocks on recurrent networks. The recurrent networks take the previous output as later input (Fig. 2 (a)). This requires the pruned indices of the first and last Convs in recurrent networks to be the same. But ASSL and SRPN cannot guarantee filter indices are aligned. Besides, many SR methods [45, 56] have shown that the information contained in front Conv layers can help the restoration feature extraction of later Conv layers. Thus, we design a Residual Sparsity Connection (RSC) for VSR recurrent networks, which preserves all channels of the input and output feature maps and selects the important channels for operation (Fig. 1 (d)). Compared with other pruning schemes [57, 58], RSC does not require the pruned indices of the first and last Convs of recurrent networks to be the same, can preserve the information contained in all layers, and liberates the pruning space of the last Conv of the residual blocks without adding extra calculations. Notably, RSC can prune residual blocks globally (*i.e.*, the filters in various layers are compared together to remove unimportant ones).

*Corresponding Author



(a) Residual Block.

(b) Skipping Pruning the Last Conv.

(c) Pruning with Structure Alignment.

(d) Residual Sparsity Connection Scheme (RSC, Ours).

Figure 1. Illustration of different schemes for pruning residual blocks of recurrent networks. **(a)** Structure of the residual block in the VSR network. **(b)** The residual block pruning schemes [7, 23, 42] do not prune the last Conv. **(c)** ASSL [57] and SRPN [58] prunes the same indices on skip and residual connections to keep channel alignment, which abandons some channels of input and output feature maps. **(d)** RSC preserves all channels of input and output feature maps, which does not need to align the pruned indices on the first and last Convs in recurrent networks, can fully use restoration information, and can prune the first and last Convs of residual blocks without restrictions.

(2) We observe that the upsampling network accounts for 22% of the total calculations in BasicVSR [2], which is necessary to be pruned to reduce redundancy. Since the pixel-shuffle [37] operation in VSR networks converts the channels to space, pruning the pixel-shuffle without any restrictions would cause the channel-space conversion to fail. Thus, we specially design a pixel-shuffle pruning scheme by taking four consecutive filters as the pruning unit for 2× pixel-shuffle. **(3)** Furthermore, we observe that the error of pruning VSR networks would accumulate with propagation steps increasing along with recurrent networks, which limits the efficiency and performance of pruning. Thus, we further introduce Temporal Finetuning (TF) to constrain the pruning error accumulation in recurrent networks. Overall, our main contributions are threefold:

- Our work is necessary and timely. There is an urgent need to compress VSR models for deployment. To the best of our knowledge, we are one of the first to design a structured pruning scheme for VSR.
- We propose an integral VSR pruning scheme called Structured Sparsity Learning (SSL) for various components of VSR models, such as residual blocks, recurrent networks, and pixel-shuffle operations.
- We employ SSL to train VSR models, which surpass recent pruning schemes and lightweight VSR models.

2. Related Work

2.1. Video Super-Resolution

VSR models can exploit additional information from neighboring LR frames for restoration [3, 8, 16, 17, 28, 48, 49, 52–54]. Earlier VSR methods [1, 39, 49] estimate the optical flow between LR frames and perform spatial warping for alignment. Later methods resort to a more sophisticated approach of implicit alignment. Instead of image-level motion alignment, TDAN [40] and EDVR [44] work at the feature level. TDAN [40] first adopted deformable Conv [6] in VSR to align the features of different frames. EDVR [44] extended TDAN by introducing coarse-to-fine deformable alignment and a new spatial-temporal attention fusion module. RSDN [16] adopted a recurrent detail-structural block and a hidden state adaptation module to reduce the effect of appearance changes and error accumulation. Recently, BasicVSR [2] found that bidirectional propagation coupled with a simple optical flow-based feature alignment can further improve performance. Similarly, Yi *et al.* [50] used the bidirectional propagation framework to exploit LR frames and estimated hidden states from the past, present, and future. To compress the VSR model, Xiao *et al.* designed a space-time knowledge distillation scheme [47]. However, these VSR methods require high computational costs impeding their application on resource-limited devices. Different from previous methods, we focus on designing SSL to compress VSR models by pruning redundant filters.

2.2. Network Pruning

Network pruning [4, 5, 34, 36, 38] is widely used to remove a set of redundant parameters for network acceleration. Pruning methods can be divided into two branches, structured pruning [10, 14, 23, 46] and unstructured pruning [11, 12]. Structured pruning methods prune the network at the level of filters, channels, and even layers, which can obtain regular sparsity after pruning. This is beneficial for acceleration. In contrast, unstructured methods focus on pruning weights, leading up to much irregular sparsity. This is beneficial for compression but tends not to yield an actual acceleration [41, 46]. Specifically, Li *et al.* [23] applied the L_1-norm to measure the importance of different filters and then removed the less important ones. Afterward, Liu *et al.* [30] added a sparsity-inducing penalty term on scaling factors of the batch normalization layers to enforce the channels with lower scaling factors to be the less informative ones. Recently, ASSL [57] and SRPN [58] utilized aligned structured sparsity learning for structured pruning of residual blocks. In addition, Luo *et al.* [32] developed a residual block pruning scheme for image classification using the Convs on skip connections. However, the residual blocks of VSR networks do not have such Convs. Lin *et al.* [26] conducted runtime neural network pruning according to the input image. Besides, Wang *et al.* [43] designed an unstructured pruning scheme for single image SR tasks by using sparse Conv to skip redundant computations. Since we cannot directly apply a general structured pruning method for VSR, we explore the properties of VSR networks and develop a VSR pruning scheme in this paper.

3. Methodology

3.1. Overview

Figure 2 (a) shows VSR networks based on the bidirectional recurrent structures, such as BasicVSR [2]. Given a LR frame I_t, the forward network concatenates I_t and the previous hidden state $H_{F,t-1}$ to extract features from I_t and aggregate the reference information from $H_{F,t-1}$. Similarly, the backward network extracts features from I_t and aggregates the reference information from the future hidden state $H_{B,t+1}$. Note that both the forward and backward networks consist of numerous residual blocks. Then, the features generated by forward and backward networks are fed into the upsampling network, which consists of multiple pixel-shuffle operations and Convs, to obtain the recovered frame SR_t. However, SOTA VSR networks [2, 3, 50] need massive computation and memory resources, limiting their deployment on resource-limited devices.

To pursue more efficient VSR networks, we specially design a VSR structured pruning scheme called Structured Sparsity Learning (SSL), according to the properties of VSR networks. Specifically, SSL has three stages, including pretraining, pruning, and finetuning. In the pretraining stage, we train a powerful VSR network. Since current VSR networks do not use BatchNorm [15], we introduce a scaling factor in pretrained VSR models to tune the sparsity of each channel and filter. In the pruning stage, we select the unimportant filters according to the pruning criterion and apply sparsity-inducing regularization on corresponding scaling factors. In addition, we propose a Residual Sparsity Connection (RSC) scheme to liberate the restrictions on pruning residual blocks of recurrent networks and preserve all restoration information contained in channels of feature maps for better performance. Moreover, for the upsampling networks, we specially develop a pruning scheme for the pixel-shuffle operation to guarantee the accuracy of channel-space conversion after pruning. Besides, we observe that the error of the hidden state would be amplified with the propagation in recurrent networks after pruning. Thus, in the finetuning stage, we design Temporal Finetuning (TF) to alleviate the error accumulation.

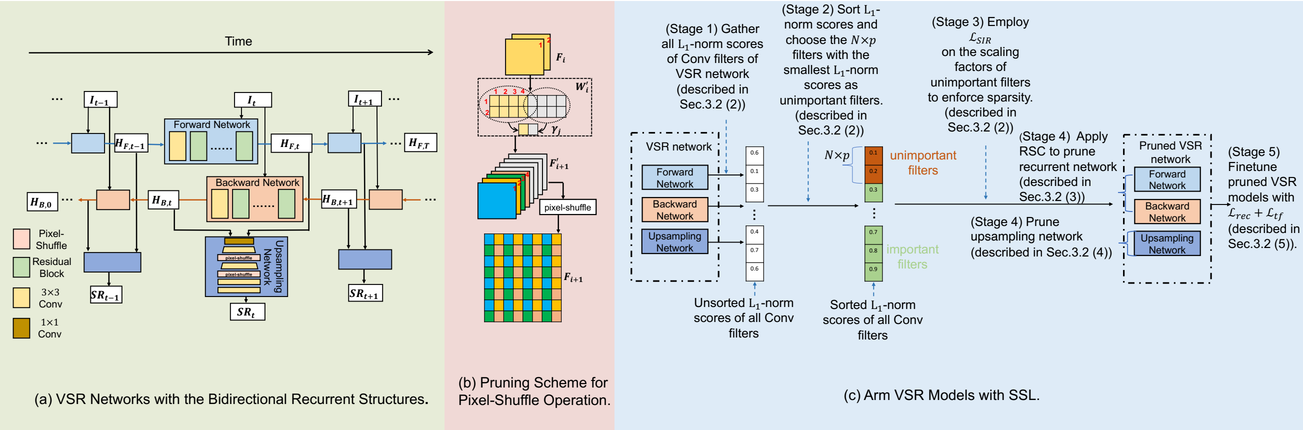
3.2. Structured Sparsity Learning

Structured Sparsity Learning (SSL) is a structured pruning scheme specially designed for VSR. It can reduce the redundancy of neural filters and obtain more efficient VSR submodels. Next, we will explain our SSL in detail.

(1) Scaling Factor. Structured pruning aims to remove Conv filters based on a designed importance criterion. In the classification task, quite a few works use scale parameters of BatchNorm [15] to control the throughput of each filter. Zero scale parameters make the value of corresponding channels vanish. As a result, they contribute nothing to the subsequent Convs and can be removed. By regularizing the scale parameter, we can assess and tune the importance of each filter. However, the BatchNorm is not useful for SR tasks [25], and SOTA VSR networks [2, 3, 50] do not utilize it. Therefore, it is infeasible to apply the existing pruning schemes directly. In our pruning scheme, as shown in Fig. 1 (d) and Fig. 2 (b), we multiply the scaling factors $\boldsymbol{\gamma}$ before or after Convs. Then, we perform regularization on scaling factors to enforce sparsity for pruning.

(2) Pruning Criterion and Regularization Form. To remove the redundant filters, we need to select unimportant scaling factors $\boldsymbol{\gamma}$ to induce sparsity. In previous works, ASSL [57] and SRPN [58] had to adopt a local pruning scheme (namely, scaling factors are only compared within the same layer, and each layer has the same pruning ratio) to guarantee that skip and residual connections keep the same number of filters and indices for the adding operation. Given that the importance of the Conv filters in various layers is different and our RSC does not have restrictions as ASSL, we can adopt the global pruning scheme (*i.e.*, scaling factors of different layers are compared together).

For the pruning criterion, we adopt the simple and practical L_1-norm. Specifically, given Conv kernel $\boldsymbol{W_i} \in \mathbb{R}^{C_{out} \times C_{in} \times K_h \times K_w}$ in the i-th layer, we calculate the absolute weight sum of k-th Conv filter $\boldsymbol{W_i}[k, ...] \in \mathbb{R}^{C_{in} \times K_h \times K_w}$ with $s_{i,k} = \sum |\boldsymbol{W_i}[k, ...]|$. In particular, for our RSC in Fig. 1 (d), we require to additionally prune the input channels for the first Conv, and calculate its L_1-norm score with $s'_{i,k} = \sum |\boldsymbol{W_i}[:, k, ...]|$, where $\boldsymbol{W_i}[:, k, ...] \in \mathbb{R}^{C_{out} \times K_h \times K_w}$. Moreover, for the Conv


Time
I_{t-1}
I_t
I_{t+1}
Forward Network
$H_{F,t-1}$
$H_{F,t}$
$H_{F,T}$
Backward Network
$H_{B,0}$
$H_{B,t}$
$H_{B,t+1}$
Upsampling Network
pixel-shuffle
pixel-shuffle
SR_{t-1}
SR_t
SR_{t+1}
Pixel-Shuffle
Residual Block
3×3 Conv
1×1 Conv
F_i
W'_i
γ_j
F'_{i+1}
pixel-shuffle
F_{i+1}
(Stage 1) Gather all L_1-norm scores of Conv filters of VSR network (described in Sec.3.2 (2))
(Stage 2) Sort L_1-norm scores and choose the $N\times p$ filters with the smallest L_1-norm scores as unimportant filters. (described in Sec.3.2 (2))
(Stage 3) Employ $\mathcal{L}_{SIR}$ on the scaling factors of unimportant filters to enforce sparsity. (described in Sec.3.2 (2))
VSR network
Forward Network
Backward Network
Upsampling Network
0.6
0.1
0.3
0.4
0.7
0.6
$N\times p$
0.1
0.2
0.3
0.7
0.8
0.9
unimportant filters
important filters
(Stage 4) Apply RSC to prune recurrent network (described in Sec.3.2 (3))
(Stage 4) Prune upsampling network (described in Sec.3.2 (4))
Pruned VSR network
Forward Network
Backward Network
Upsampling Network
(Stage 5) Finetune pruned VSR models with $\mathcal{L}_{rec}+\mathcal{L}_{tf}$ (described in Sec.3.2 (5)).
Unsorted L_1-norm scores of all Conv filters
Sorted L_1-norm scores of all Conv filters

(a) VSR Networks with the Bidirectional Recurrent Structures.

(b) Pruning Scheme for Pixel-Shuffle Operation.

(c) Arm VSR Models with SSL.

Figure 2. **(a)** The basic architecture of the VSR methods with the bidirectional recurrent network. The forward and backward networks both consist of numerous residual blocks. The upsampling networks contain multiple pixel-shuffle operations and Convs. **(b)** The pruning scheme for the pixel-shuffle. For the 2× upsampling pixel-shuffle [37] operation, we take four channels with consecutive indices as the pruning unit to guarantee the accuracy of channel-space conversion after pruning. **(c)** The application of SSL on VSR models.

before the pixel-shuffle operation, we take four consecutive filters as a pruning unit and calculate the score with $s_{i,k} = \sum |\boldsymbol{W_i}[4k:4(k+1),...]|$. Then, given the pruning ratio p and the total number of filters or channels N, we sort all their L_1-norm scores s together and choose the $N \times p$ filters with the smallest L_1-norm values as unimportant filters or channels, denoted as set S (Fig. 2 (c)).

After identifying the unimportant filters and channels set S, we apply sparsity-inducing regularization (SIR) to the corresponding scaling factors, denoted as set S_{sf}. It is notable that we do not enforce sparsity-inducing regularization to the important filters and channels since they will remain in the network. Specifically, we employ L_2 regularization on the scaling factors to enforce sparsity (Fig. 2 (c)):

$$\mathcal{L}_{SIR} = \alpha_\gamma \sum_{\gamma \in S_{sf}} \gamma^2, \tag{1}$$

where γ is the scalar selected from $\boldsymbol{\gamma} \in \mathbb{R}^C$ corresponding to unimportant filters or channels; α_γ is a scalar. We increment α_γ by a presetting constant Δ every T_1 iterations. When α_γ reaches the pre-defined upper limit τ, we keep α_γ constant and continue training T_2 iterations.

(3) Pruning Scheme for Residual Blocks of Recurrent Networks. VSR recurrent networks consist of residual blocks. Residual blocks are difficult to prune because the addition operations require the pruned filter indices between the skip and residual connections to be the same. As shown in Fig. 1 (b), quite a few pruning schemes [23, 30, 34] simply skipped the pruning of the last Conv in residual blocks, which restricted the pruning space. Recently, as shown in Fig. 1 (c), ASSL [57] and SRPN [58] pruned the last Conv in the residual block by using regularization. However, recurrent networks take the previous output as later input. Thus, pruning recurrent networks require the pruned indices of the first and last Convs to be the same. ASSL and SRPN cannot guarantee the indices of important filters in the first and last Convs of recurrent networks are aligned. Besides, ASSL and SRPN have to remove some channels in output feature maps and adopt a local pruning scheme, which limits pruning space and restoration information utilization (RDN [56] and ESRGAN [45] have shown restoration information of front layers can guide feature extraction of later layers). To break the restriction of pruning and fully use restoration information contained in channels of front layers, we propose the RSC to prune residual blocks in recurrent networks (Fig. 1 (d)). As we can see, RSC preserves all channels of input $\boldsymbol{F'_i}$ and output $\boldsymbol{F'_{i+1}}$ in the residual blocks. For the first Conv, we select the important channels (the indices not in S) to participate in the Conv operation, which can be expressed as Eq. 2. After the last Conv, we obtained $\boldsymbol{F_{i+1}}$ and add $\boldsymbol{F_{i+1}}$ to $\boldsymbol{F'_i}$ on the corresponding channel indices to obtain $\boldsymbol{F'_{i+1}}$ (Eq. 3 and Eq. 4). Furthermore, we do not prune the 1×1 Conv of upsampling networks (Fig. 2 (a)) to aggregate all preserved restoration information in $H_{F,t}, H_{B,t} \in \mathbb{R}^{C\times H\times W}$.

$$\boldsymbol{F_i} = \boldsymbol{F'_i} \otimes (\boldsymbol{\gamma_{j-1} W_i \gamma_j}), \tag{2}$$

$$\boldsymbol{F_{i+1}} = \boldsymbol{F_i} \otimes (\boldsymbol{W_{i+1} \gamma_{j+1}}), \tag{3}$$

$$\boldsymbol{F'_{i+1}} = \boldsymbol{F_{i+1}} + \boldsymbol{F'_i}, \tag{4}$$

where $\otimes$ indicates Conv. $\boldsymbol{F'_i}, \boldsymbol{F'_{i+1}} \in \mathbb{R}^{C\times H\times W}$ are the input and output feature maps of the residual block, respectively. $\boldsymbol{F_i}, \boldsymbol{F_{i+1}} \in \mathbb{R}^{C_p\times H\times W}$ are intermediate feature maps. $\boldsymbol{W_i}, \boldsymbol{W_{i+1}} \in \mathbb{R}^{C_{out}\times C_{in}\times K_h\times K_w}$ are weights of Conv kernels. $\boldsymbol{\gamma_{j-1}} \in \mathbb{R}^{C_{in}}$ and $\boldsymbol{\gamma_j}, \boldsymbol{\gamma_{j+1}} \in \mathbb{R}^{C_{out}}$ are scaling factors to apply sparsity-inducing regularization. $\boldsymbol{F'_{i+1}}$ prunes some channels, and $\boldsymbol{F_i}$ keeps all channels. In Eq. 4, $\boldsymbol{F'_i}$ adds $\boldsymbol{F_{i+1}}$ on corresponding kept channels. It is notable that our RSC does not introduce extra parameters and computational cost.

(4) Pruning Scheme for Pixel-Shuffle. The upsampling network of the VSR network uses Conv to increase channels of feature maps and adopts the pixel-shuffle [37] operation to convert the channels to space realizing upsampling. As shown in Fig. 2 (b), given the input feature map $F_i \in \mathbb{R}^{C\times H\times W}$, we expand its channels 4× by a Conv with weight W_i to obtain $F'_{i+1} \in \mathbb{R}^{4C\times H\times W}$. Then, the pixel-shuffle operation takes four channels as a group to convert F'_{i+1} to $F_{i+1} \in \mathbb{R}^{C\times 2H\times 2W}$ realize 2× upsampling. We observe that, if we prune the Conv before pixel-shuffle without any restriction, the pruned feature maps will be spatially disordered after passing the pixel-shuffle operation and lead to performance drop. To address the problem, we specially design a strong and simple pruning scheme for the pixel-shuffle operation. Given the input feature map, we take four filters as a pruning unit to evaluate their importance and then impose the scaling factor $\boldsymbol{\gamma_j}$ on filters to enforce sparsity (described in (2) pruning criterion and regularization form):

$$\boldsymbol{W_i'} = \boldsymbol{W_i}[4k:4(k+1),...]\boldsymbol{\gamma_j}[k], k \in [0, C_{in}), \tag{5}$$

where $\boldsymbol{W_i} \in \mathbb{R}^{4C_{in}\times C_{in}\times K_h \times K_w}$ is the weights of Conv kernel. $\boldsymbol{\gamma_j}$ is the scaling factor.

(5) Temporal Finetuning. We observe that the pruned VSR network generates a minor error in hidden state H_F and H_B (Fig. 2 (a)), which will be amplified as the hidden state propagates along with recurrent networks. To solve the issue, we introduce Temporal Finetuning (TF) by enforcing the hidden states of pruned networks to align with the accurate hidden states of unpruned networks:

$$\mathcal{L}_{tf} = \left\| H_{F,T} - H'_{F,T} \right\| + \left\| H_{B,0} - H'_{B,0} \right\|, \tag{6}$$

where T is the number of input frames, and $H_{F,T}$ and $H'_{F,T}$ are the final hidden states after T frames forward propagation in pruned and original VSR networks, separately. Similarly, $H_{B,0}$ and $H'_{B,0}$ are the final hidden states after backward propagation in the pruned and original VSR networks.

To train the whole VSR network, we use the Charbonnier loss [2, 44], which can be formulated as:

$$\mathcal{L}_{rec} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\left\| SR_t - HR_t \right\|^2 + \varepsilon^2}, \tag{7}$$

where ε is set to 10^{-6}. SR_t and HR_t are t-th reconstructed and HR frames, respectively. The overall loss function for pruned network finetuning is designed as:

$$\mathcal{L}_{all} = \mathcal{L}_{rec} + \mathcal{L}_{tf}. \tag{8}$$

3.3. Arm VSR Models with SSL

Our SSL can be used for VSR networks. BasicVSR [2] and BasicVSR++ [3] are two strong SOTA VSR methods. The dense deformable Conv of BasicVSR++ requires reading a large amount of irregular memory data, which is unsuitable for deployment on resource-limited devices without GPUs. Besides, the Second-Order Grid Propagation of BasicVSR++ will force a delay of two frames, which further impedes its usage in real-time devices.

In our study, we use BasicVSR for VSR pruning, which is more suitable for applications on edge devices. In addition, we further propose unidirectional BasicVSR (BasicVSR-uni), obtained by removing the backward network, for online inference. Since the SpyNet of BasicVSR is used for flow estimation, we do not apply our pruning scheme SSL on it. In the pruning stage, as shown in Fig. 2 (c), we first add the scaling factor to the Conv and residual blocks as described in Sec. 3.2. Then we use the pruning criterion to select unimportant filters globally and apply sparsity-inducing regularization to the corresponding scaling factor. Afterward, we remove the unimportant Conv filters and finetune the pruned VSR network with T_3 iterations. We provide more details in Alg. 1 of supplementary.

4. Experiments

4.1. Experimental Settings

We adopt two widely used datasets for training: REDS [33] and Vimeo-90K [49]. For REDS, we use REDS4 containing 4 clips as our test set. Additionally, we adopt REDSval4 as our validation set, which contains 4 clips selected from the REDS validation set. The remaining clips of REDS are used for training. In addition, we utilize Vid4 [27], UDM10 [52], and Vimeo-90K-T [49] as test sets along with Vimeo-90K. We train and test models with 4× downsampling using two degradations Bicubic (BI) and Blur Downsampling (BD) as BasicVSR did. For BI, the MATLAB function "imresize" is used for downsampling. For BD, we blur the HR images by a Gaussian filter with σ =1.6, followed by a subsampling every four pixels.

We pretrain the unidirectional BasicVSR (BasicVSR-uni) as done for BasicVSR. In sparsity-inducing regularization, the iterations T_1 and T_2 are set to 5 and $3,375$ separately. The scalars Δ and τ are set to 10^{-4} and 0.1, respectively. Note that we fix the parameters of the flow estimator in sparsity-inducing regularization. In the pruned VSR network finetuning, we set T_3 to $300,000$. We adopt the Adam optimizer [22] and Cosine Annealing scheme [31]. The initial learning rate of the flow estimator is 2.5×10^{-5}. The learning rate for all other modules is 2×10^{-4}. The patch size of input LR frames is 64×64. Experiments are conducted on a server with PyTorch 1.10 and V100 GPUs.

4.2. Quantitative and Qualitative Comparisons

Since BasicVSR violates causality and cannot be evaluated online, we construct the unidirectional BasicVSR (BasicVSR-uni) by removing the backward network for online inference. We compare the proposed SSL with three other pruning schemes at pruning ratio $p = 0.5$: L_1-norm pruning [23] (which simply removes filters with the smallest L_1-norms and is the most prevailing filter pruning method now), and ASSL [57]. We apply these pruning schemes on BasicVSR, thus obtaining L_1-norm-bi, ASSL-bi, and SSL-bi separately. In addition, we use these

Table 1. Quantitative comparison (average PSNR/SSIM). Pruning schemes applied on bidirectional and unidirectional BasicVSR ("bi" and "uni") and marked in rouse and gray, respectively. $*$ means the space-time knowledge distillation scheme [47]. We mark the best results among comparing pruning schemes in bold. The FLOPs and runtime are computed based on an LR size of 180×320.

				BI degradatioin			BD degradatioin		
Methods	Params (M)	FLOPs (G)	Runtime (ms)	REDS4 [33]	Vimeo-90K-T [49]	Vid4 [27]	UDM10 [52]	Vimeo-90K-T [49]	Vid4 [27]
Bicubic	-	-	-	26.14/0.7292	31.32/0.8684	23.78/0.6347	28.47/0.8253	31.30/0.8687	21.80/0.5246
VESPCN [1]	-	-	-	-	-	25.35/0.7557	-	-	-
SPMC [39]	-	-	-	-	-	25.88/0.7752	-	-	-
TOFlow [49]	1.4	274.9	1610	27.98/0.7990	33.08/0.9054	25.89/0.7651	36.26/0.9438	34.62/0.9212	-
DUF [21]	5.8	1645.8	974	28.63/0.8251	-	-	38.48/0.9605	36.87/0.9447	27.38/0.8329
RBPN [13]	12.2	8516.0	1507	30.09/0.8590	37.07/0.9435	27.12/0.8180	38.66/0.9596	37.20/0.9458	-
EDVR-M [44]	3.3	304.2	118	30.53/0.8699	37.09/0.9446	27.10/0.8186	39.40/0.9663	37.33/0.9484	27.45/0.8406
PFNL [51]	3.0	940.0	295	29.63/0.8502	36.14/0.9363	26.73/0.8029	38.74/0.9627	-	27.16/0.8355
TGA [18]	5.8	694.1	236	-	-	-	-	37.59/0.9516	27.63/0.8423
RLSP [9]	4.2	82.3	49	-	-	-	38.48/0.9606	36.49/0.9403	27.48/0.8388
RSDN [16]	6.2	355.7	94	-	-	-	39.35/0.9653	37.23/0.9471	27.92/0.8505
RRN [19]	3.4	108.7	45	-	-	-	38.96/0.9644	-	27.69/0.8488
FastDVDnet* [47]	2.6	64.3	-	-	36.12/0.9348	26.14/0.8029	-	-	-
BasicVSR [2]	4.9	338.5	57	31.42/0.8909	37.18/0.9450	27.24/0.8251	39.96/0.9694	37.53/0.9498	27.96/0.8553
BasicVSR-lite	1.3	85.5	24	30.56/0.8738	36.57/0.9397	26.86/0.8125	38.98/0.9645	36.78/0.9431	27.27/0.8327
L_1-norm-bi [23]	1.3	85.5	24	30.66/0.8766	36.69/0.9406	26.87/0.8121	39.04/0.9650	36.84/0.9437	27.29/0.8335
ASSL-bi [57]	1.3	85.5	24	30.74/0.8770	36.75/0.9414	27.01/0.8176	39.15/0.9660	36.93/0.9450	27.40/0.8400
SSL-bi (Ours)	1.0	92.1	24	**31.06/0.8933**	**36.82/0.9419**	**27.15/0.8208**	**39.35/0.9665**	**37.06/0.9458**	**27.56/0.8431**
BasicVSR-uni [2]	2.6	218.1	39	30.56/0.8698	36.95/0.9429	27.01/0.8164	39.25/0.9645	37.25/0.9472	27.57/0.8424
BasicVSR-uni-lite	0.7	62.4	18	29.95/0.8561	36.38/0.9372	26.68/0.8012	38.24/0.9586	36.38/0.9388	26.87/0.8157
L_1-norm-uni [23]	0.7	62.4	18	29.97/0.8570	36.45/0.9381	26.70/0.8031	38.43/0.9601	36.53/0.9405	26.89/0.8187
ASSL-uni [57]	0.7	62.4	18	30.02/0.8589	36.49/0.9385	26.76/0.8051	38.48/0.9603	36.61/0.9416	27.02/0.8236
SSL-uni (Ours)	0.5	63.9	18	**30.24/0.8633**	**36.56/0.9392**	**27.01/0.8148**	**38.68/0.9615**	**36.77/0.9429**	**27.18/0.8296**

pruning schemes on BasicVSR-uni, obtaining L_1-norm-uni, ASSL-uni, and SSL-uni. Moreover, we reduce the channels of BasicVSR and BasicVSR-uni to obtain lightweight VSR models BasicVSR-lite and BasicVSR-uni-lite, respectively. Furthermore, we compare our pruned BasciVSR and BasicVSR-uni with other lightweight VSR networks, including TOFlow [49], EDVR-M [44], RLSP [9], RSDN [16], *etc*. Since we only prune VSR networks, the parameters and FLOPs of the optical flow network, SPyNet [35] (Params 1.4M, Flops 19.6G) are not included.

The quantitative results measures performance (PSNR and SSIM), the number of parameters, runtime, and FLOPs on the different methods, which are shown in Tab. 1. **(1)** Compared with competitive lightweight VSR networks, our SSL-bi obtains 0.53dB gain on REDS4 over EDVR-M. Note that, different from careful network designs like EDVR-M, we merely prune the BasicVSR, a simple backbone with 60 residual blocks, obtaining superior performance while only consuming less the FLOPs of EDVR-M. **(2)** Our SSL-bi surpasses the BasicVSR-lite by 0.5dB, and SSL-uni surpasses the BasicVSR-uni-lite by 0.29dB. This demonstrates the effectiveness of applying SSL for offline and online VSR network pruning. **(3)** We also adapt some SOTA pruning schemes, such as the L_1-norm and ASSL, to VSR networks for comparison. As a result, our SSL achieves superior performance on BasicVSR and BasicVSR-uni to other pruning schemes. This shows that SSL can make better use of the sparsity of the network and increases the efficiency of the learned network parameters. **(4)** Besides, comparing our pruning scheme and other VSR model compression method (space-time knowledge distillation scheme [47], FastDVDnet*), our SSL-uni (0.5M parameters) surpasses the FastDVDnet* (2.6M parameters) by 0.87dB on Vid4, which further demonstrates the effectiveness of our structured pruning scheme. Moreover, with our SSL strategy, we do not have to train a teacher network as knowledge distillation [47] methods did.

The qualitative results are shown in Fig. 3. Our SSL-bi achieves the best visual quality containing more realistic details. These visual comparisons are consistent with the quantitative results, showing the superiority of SSL. SSL can learn to remove the redundant filters to compress a network to a smaller one while maintaining the most restoration ability. More visual results are given in supplementary.

5. Ablation Study

The Validation of Components in SSL. We conduct an ablation study to demonstrate the effectiveness of our SSL by progressively adding components. The results are shown in Tab. 2. SSL_1 uses the aligned pruning [57] scheme for residual blocks, while SSL_4 adopts our RSC. Comparing SSL_1 and SSL_4, we can see that our RSC is superior to the advanced residual block pruning scheme. That is because RSC can break the pruning restrictions and preserve all information contained in feature channels for VSR. For SSL_2, we halve the number of filters in upsampling networks to keep the same model size as other models. As we can see, SSL_3 outperforms SSL_2. This is because introducing the pruning scheme for the pixel-shuffle operation can increase the available pruning space in upsampling networks. Com-

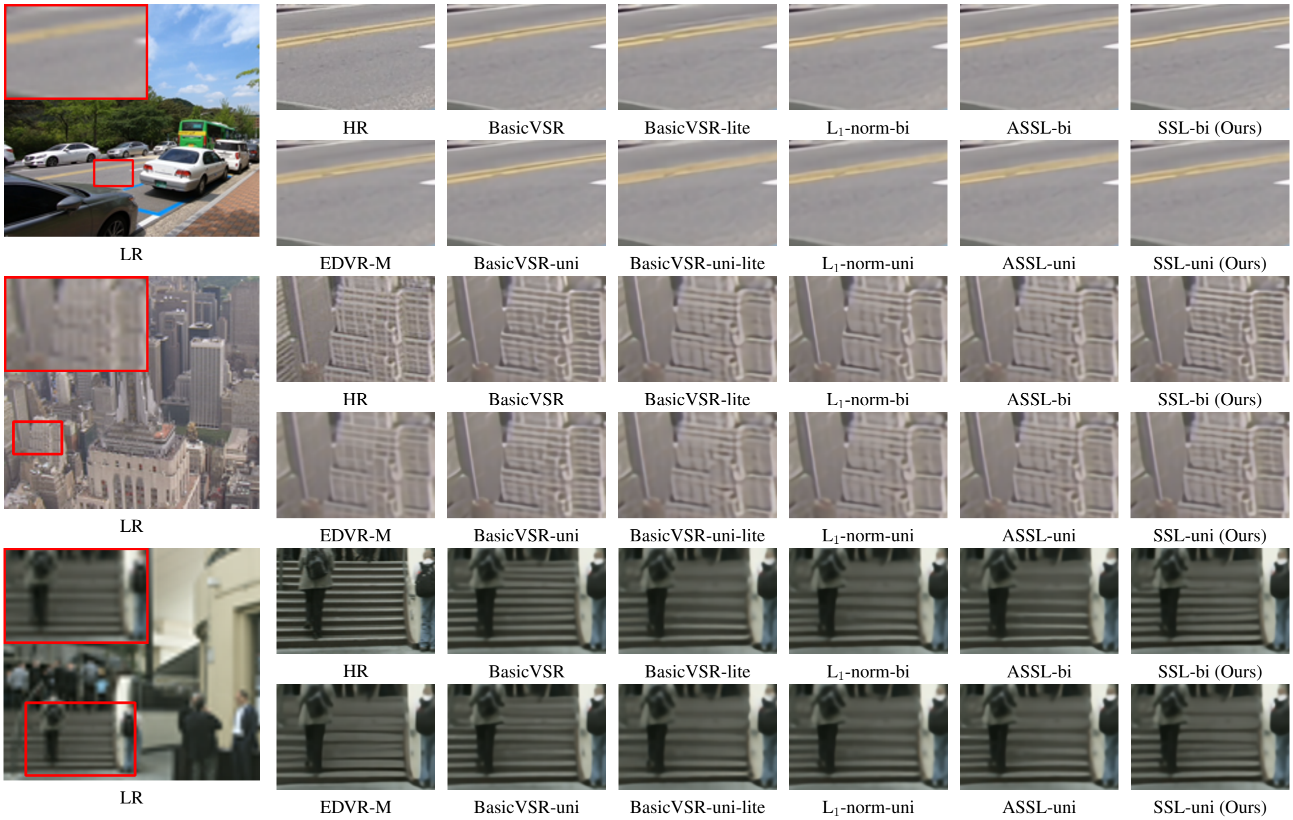


Figure 3. Qualitative comparison between various VSR and pruning schemes on REDS4 [33], Vid4 [27], and Vimeo90K-T [49], separately.

Table 2. Validation of the components in our SSL. PSNR (dB) results evaluated on REDS4 [33] (4×). The backbone is BasicVSR [2], and the pruning ratio is set to 0.5.

Methods	SSL_1	SSL_2	SSL_3	SSL_4 (Ours)
Aligned Pruning [57]	✓			
Residual Sparsity Connection		✓	✓	✓
Pixel-Shuffle Pruning	✓		✓	✓
Temporal Finetuning	✓			✓
PSNR (dB)	30.86	30.82	30.98	31.06

paring SSL_4 and SSL_3, we can see that adopting TF can bring a 0.08 dB improvement, reducing the error accumulation of hidden states in the recurrent network after pruning.

Pruning Methods with Various Pruning Ratios. To further demonstrate SSL's effectiveness, we compare it with widely used pruning schemes, including L_1-norm [23], ASSL [57] at different pruning ratios. We use these pruning schemes to obtain numerous submodels with different FLOPs. Besides, we also adjust the channels of BasicVSR to obtain BasicVSR-lite with different FLOPs. The results are shown in Fig. 4. **(1)** Our SSL achieves the best performance compared with other pruning schemes at different pruning ratios and FLOPs. Note that SSL even surpasses the BasicVSR-lite in the same model size by 0.73 dB on submodels with around 3.2G FLOPs. This demonstrates the superiority of SSL on VSR. **(2)** With the pruning ratio increasing and FLOPs decreasing, the performance advantage brought by our SSL becomes more evident compared with BasicVSR-lite, L_1-norm and ASSL.

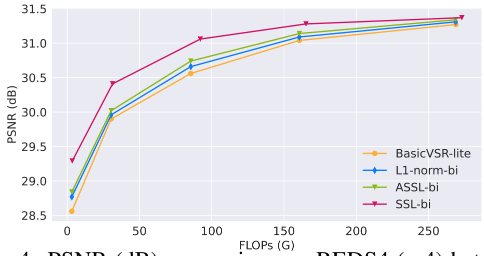
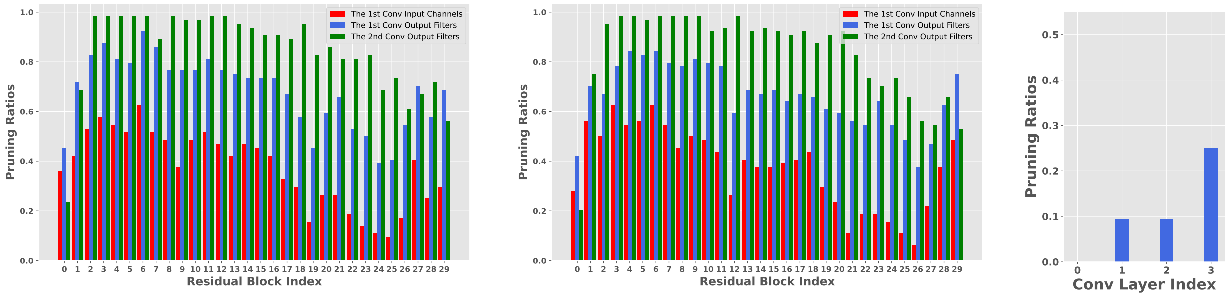


Figure 4. PSNR (dB) comparison on REDS4 (×4) between SSL and three other methods obtaining the *same* small network.

Comparison with Different Pruning Criteria. We explore the influence of different pruning criteria on the pruned VSR model at different pruning ratios. Specifically, we select and remove the unimportant filters globally (namely, comparing all filters from all layers together) with minimum L_1-norm scores, which is expressed as "Min + Global". In addition, we select and remove the unimportant filters locally (namely, filters are compared with each other in the same layer, and each layer has the same pruning ratio) with maximum L_1-norm scores, which is expressed as "Max + Local". Similarly, we determine "Max + Global" and "Min + local". Furthermore, we randomly remove the unimportant filters as "Rand". Then, we com-



(a) Forward Network.

(b) Backward Network.

(c) Upsampling Network.

Figure 5. **(a)**, **(b)**, and **(c)** show the pruning ratios of Conv layers in forward, backward and upsampling networks, respectively.

Table 3. PSNR (dB) comparison on REDS4 (4×) for our pruning scheme (SSL) with different pruning criteria and pruning ratios. The unpruned model is BasicVSR [2] baseline.

Pruning Ratios	Min Global (Ours)	Max Global	Min Local	Max Local	Rand
0.3	31.28	30.59	31.22	30.97	31.10
0.5	31.06	28.90	30.83	30.38	30.68
0.7	30.41	25.89	30.20	29.51	30.09

pare all pruning criteria at 0.3, 0.5, and 0.7 pruning ratios. The results are shown in Tab. 3. **(1)** The "Min + Global" pruning criterion achieves the best performance at different pruning ratios. It implies that the filters with minimum L_1-norm scores are relatively unimportant for VSR. Besides, this shows that the importance of filters in different layers are different and it is better to compare them together to select unimportant filters (pruning globally) for VSR. **(2)** The performance of "Max + Global" and "Max + local" are both inferior to "Rand" for the removal of more important filters. This implies that filters with large L_1-norm scores are more important than those with small ones for VSR networks.

Pruning Ratios of Different Layers. We take BasicVSR pruned by SSL at 0.5 pruning ratio as an example. We visualize pruning ratios in different layers and show results in Fig. 5. **(1)** In the forward and backward networks, the pruning ratios of the first Conv input channels (corresponding to the $\boldsymbol{\gamma_{j-1}}$ in Fig. 1 (d)) are lower than the pruning ratios of second Conv output filters (corresponding to the $\boldsymbol{\gamma_{j+1}}$ in Fig. 1 (d)). It implies that VSR networks tend to aggregate information from the numerous input channels into several important output channels. **(2)** The residual blocks in deeper position of forward and backward networks tend to have minor pruning ratio. This means that residual blocks in deeper position contribute more to VSR. **(3)** The average pruning ratio of upsampling network is 0.2 (less than 0.5), indicating that upsampling network plays a quite important role in VSR performance. Previous works [2, 9, 25, 44, 45, 51, 55] have paid much attention to the design of feature extraction modules. In latter VSR research, paying more attention to upsampling network design is likely to improve VSR performance more.

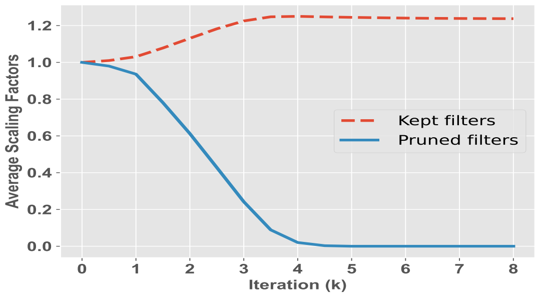


Figure 6. The SSL pruning process of Convs in the BasicVSR [2].

Regularization Visualization. To understand how sparsity-inducing regularization works, we plot the average scaling factors in the BasicVSR [2] during applying regularization at 0.5 pruning ratio (Fig. 6). The average scaling factor is split into two parts, pruned and kept. As seen, the average scaling factor γ of the pruned filters decreases as the corresponding penalty term α_γ and iterations become larger. Besides, it is interesting that the average scaling factor of the kept filters will increase without any regularization term to enforce them to be larger. It means that, as the unimportant filters are removed, the network will strengthen the kept filters to compensate for the performance, which is similar to the compensation effect in the human brain.

6. Conclusion

In this work, we propose a structured pruning scheme called SSL for efficient VSR in resource-limited situations. Specifically, for the difficulty of pruning residual blocks of recurrent networks, we propose the RSC. Compared with previous pruning schemes for residual blocks, RSC does not have pruning restrictions as other pruning schemes and can fully utilize restoration information in all channels for better performance. In addition, for the pixel-shuffle operation in the upsampling network, we specially design a pruning scheme by grouping filters to guarantee the accuracy of channel-space conversion after pruning. Furthermore, we propose Temporal Finetuning to reduce the error accumulation in recurrent networks. We apply SSL on the BasicVSR, and SSL achieves superior performance to that of recent SOTA methods, quantitatively and qualitatively.

References

[1] 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. 2, 6

[2] 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*, 2021. 1, 2, 3, 5, 6, 7, 8

[3] Kelvin CK Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. *arXiv preprint arXiv:2104.13371*, 2021. 2, 3, 5

[4] Jian Cheng, Pei-song Wang, Gang Li, Qing-hao Hu, and Han-qing Lu. Recent advances in efficient computation of deep convolutional neural networks. *Frontiers of Information Technology & Electronic Engineering*, 2018. 1, 2

[5] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 2018. 1, 2

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 2

[7] Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive cnn width optimization. In *ICML*, 2019. 2

[8] Dario Fuoli, Shuhang Gu, and Radu Timofte. Efficient video super-resolution through recurrent latent space propagation. In *ICCVW*, 2019. 2

[9] Dario Fuoli, Shuhang Gu, and Radu Timofte. Efficient video super-resolution through recurrent latent space propagation. In *ICCVW*, 2019. 6, 8

[10] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *ICLR*, 2018. 2

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1, 2

[12] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *NeurIPS*, 28, 2015. 1, 2

[13] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *CVPR*, 2019. 6

[14] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 1, 2

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3

[16] Takashi Isobe, Xu Jia, Shuhang Gu, Songjiang Li, Shengjin Wang, and Qi Tian. Video super-resolution with recurrent structure-detail network. In *ECCV*, 2020. 2, 6

[17] Takashi Isobe, Xu Jia, Xin Tao, Changlin Li, Ruihuang Li, Yongjie Shi, Jing Mu, Huchuan Lu, and Yu-Wing Tai. Look back and forth: Video super-resolution with explicit temporal difference modeling. In *CVPR*, 2022. 2

[18] Takashi Isobe, Songjiang Li, Xu Jia, Shanxin Yuan, Gregory Slabaugh, Chunjing Xu, Ya-Li Li, Shengjin Wang, and Qi Tian. Video super-resolution with temporal group attention. In *CVPR*, 2020. 6

[19] Takashi Isobe, Fang Zhu, Xu Jia, and Shengjin Wang. Revisiting temporal modeling for video super-resolution. *BMVC*, 2020. 6

[20] 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. 1

[21] 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. 6

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ICLR*, 2017. 1, 2, 3, 4, 5, 6, 7

[24] Sheng Li, Fengxiang He, Bo Du, Lefei Zhang, Yonghao Xu, and Dacheng Tao. Fast spatio-temporal residual network for video super-resolution. In *CVPR*, 2019. 1

[25] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017. 3, 8

[26] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *NeurIPS*, 2017. 3

[27] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *TPAMI*, 2013. 5, 6, 7

[28] Ding Liu, Zhaowen Wang, Yuchen Fan, Xianming Liu, Zhangyang Wang, Shiyu Chang, and Thomas Huang. Robust video super-resolution with learned temporal dynamics. In *ICCV*, 2017. 2

[29] Hongying Liu, Peng Zhao, Zhubo Ruan, Fanhua Shang, and Yuanyuan Liu. Large motion video super-resolution with dual subnet and multi-stage communicated upsampling. *arXiv preprint arXiv:2103.11744*, 2021. 1

[30] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 3, 4

[31] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5

[32] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In *CVPR*, 2020. 3

[33] 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. 5, 6, 7

[34] Junghun Oh, Heewon Kim, Seungjun Nah, Cheeun Hong, Jonghyun Choi, and Kyoung Mu Lee. Attentive fine-grained structured sparsity for image restoration. In *CVPR*, 2022. 1, 2, 4

[35] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017. 6

[36] Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 1993. 1, 2

[37] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 2, 4, 5

[38] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks. *Synthesis Lectures on Computer Architecture*, 2020. 1, 2

[39] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *ICCV*, 2017. 2, 6

[40] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally-deformable alignment network for video super-resolution. In *CVPR*, 2020. 2

[41] Huan Wang, Xinyi Hu, Qiming Zhang, Yuehai Wang, Lu Yu, and Haoji Hu. Structured pruning for efficient convolutional neural networks via incremental regularization. *IEEE Journal of Selected Topics in Signal Processing*, 2019. 1, 3

[42] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. *ICLR*, 2021. 2

[43] Longguang Wang, Xiaoyu Dong, Yingqian Wang, Xinyi Ying, Zaiping Lin, Wei An, and Yulan Guo. Exploring sparsity in image super-resolution for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4917–4926, 2021. 3

[44] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *CVPRW*, 2019. 1, 2, 5, 6, 8

[45] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCVW*, 2018. 1, 4, 8

[46] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *NeurIPS*, 2016. 1, 2, 3

[47] Zeyu Xiao, Xueyang Fu, Jie Huang, Zhen Cheng, and Zhiwei Xiong. Space-time distillation for video super-resolution. In *CVPR*, 2021. 2, 6

[48] Zeyu Xiao, Zhiwei Xiong, Xueyang Fu, Dong Liu, and Zheng-Jun Zha. Space-time video super-resolution using temporal profiles. In *ACM MM*, 2020. 2

[49] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 2019. 2, 5, 6, 7

[50] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, Tao Lu, Xin Tian, and Jiayi Ma. Omniscient video super-resolution. In *ICCV*, 2021. 2, 3

[51] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, and Jiayi Ma. Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations. In *ICCV*, 2019. 1, 6, 8

[52] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, and Jiayi Ma. Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations. In *ICCV*, 2019. 2, 5, 6

[53] Jiyang Yu, Jingen Liu, Liefeng Bo, and Tao Mei. Memory-augmented non-local attention for video super-resolution. In *CVPR*, 2022. 2

[54] Haochen Zhang, Dong Liu, and Zhiwei Xiong. Two-stream action recognition-oriented video super-resolution. In *ICCV*, 2019. 2

[55] 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. 8

[56] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018. 1, 4

[57] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Aligned structured sparsity learning for efficient image super-resolution. *NeurIPS*, 2021. 1, 2, 3, 4, 5, 6, 7

[58] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Learning efficient image super-resolution networks via structure-regularized pruning. In *ICLR*, 2021. 1, 2, 3, 4