Supplemental Material: 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

1. Methods

1.1. Algorithm

Our Structured Sparsity Learning (SSL) algorithm is summarized in Alg. 1.

1.2. More Experimental Settings

When finetuning the pruned models on REDS [6], following BasicVSR, we use a sequence of 15 frames as inputs, and loss is computed for the 15 output images. When finetuning on Vimeo-90K [7], we temporally augment the sequence by flipping the original input sequence to allow longer propagation, as BasicVSR [1] did. In other words, we train with a sequence of 14 frames on Vimeo-90K. During inference, we take the whole video sequence as input. We use PyTorch to implement our models with 4 Tesla V100 GPUs. Codes will be made publicly available.

2. Experimental Results

2.1. Regularization Visualization

To understand how sparsity-inducing regularization works, in the Fig. **6** of main paper, we plot the average scaling factors of the "conv15_1" of the forward network. Here, we show more average scaling factors of the forward, backward, and upsampling networks in the BasicVSR [1] during applying regularization at 0.5 pruning ratio. 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 becomes stronger. 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.

*Corresponding Author

2.2. The Propagation Error after Pruning

We test the average value of the hidden state after pruning without finetuning on REDS4. The results are shown in Fig. 2. The forward network propagates the hidden state from index 0 to 99, and its average values of the hidden state will become larger as propagation steps increase. Similarly, the backward network propagates the hidden state from index 99 to 0, and its average values of the hidden state increase with propagation. Therefore, the error of the hidden state will accumulate with propagation steps increasing after pruning for preserving the filters with large L_1-norms. Thus, we introduce the Temporal Finetuning (TF) in finetuning stage to guarantee the accuracy of temporal information propagation to further improve the performance of the pruned VSR network.

2.3. More Visual Comparisons

As the same settings in the paper Sec. **4.1**, we use BasicVSR and BasicVSR-uni as the backbones. We apply L_1-norm, ASSL, and our SSL to the BasicVSR, obtaining L_1-norm-bi, ASSL-bi, and SSL-bi, respectively, at 0.5 pruning ratio. Similarly, we apply different pruning schemes with 0.5 pruning ratio to BasicVSR-uni, obtaining BasicVSR-uni-lite, 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. We provide more visual comparisons in Fig. 3. We can observe that our SSL better alleviate the blurring artifacts compared with other lightweight VSR method (EDVR-M) and pruning schemes. This demonstrates the superiority of our method and means that SSL can prune the redundant filters and maintain most representation ability.

3. Limitations

In our work, we mainly focus on the most commonly used recurrent network based VSR networks, which consists of numerous residual blocks. For more elaborate de-

Algorithm 1 Structured Sparsity Learning (SSL)

Input: Pretrained VSR network with parameters θ, regularization increment Δ, increment interval T_1, penalty ceiling limit τ, regularization iterations T_2, finetuning iterations T_3.
Output: Pruned model with parameters θ'.

Pruning:
Init: Insert Scaling factors $\boldsymbol{\gamma}$ in the network as described in the paper Sec. **3.2** (1) and set values to 1.
Init: set α_γ to 0.
Obtain the unimportant filters or channels set S, and corresponding scaling factors set S_{sf} according to L_1-norm pruning criterion as described in the paper Sec. **3.2** (2).
while not all α_γ reach τ, $\gamma \in S_{sf}$ **do**
  if $i\%T_1 == 0$ **then**
    $\alpha_\gamma = \min\left(\alpha_\gamma + \Delta, \tau\right)$ for $\gamma \in S_{sf}$
  end if
  Calculate $\mathcal{L}_{SIR}$ (paper Eq. **1**).
  Calculate the reconstruction loss (paper Eq. **7**).
  Loss ($\mathcal{L}_{rec} + \mathcal{L}_{SIR}$) backward and parameters update by Adam optimizer [4].
end while
for $i = 1$ to T_2 **do**
  Calculate $\mathcal{L}_{SIR}$ (paper Eq. **1**).
  Calculate the reconstruction loss (paper Eq. **7**).
  Loss ($\mathcal{L}_{rec} + \mathcal{L}_{SIR}$) backward and parameters update by Adam optimizer.
end for
Remove the unimportant filters or channels in S. Note that the pruning schemes for residual blocks and pixel-shuffle are described in paper Sec. **3.2** (3) and (4), respectively.
Remove all scaling factors (scales merged with filter weights). Rebuild to obtain the pruned model.
Finetuning:
for $i = 1$ to T_3 **do**
  Calculate the temporal finetuning loss (paper Eq. **6**).
  Calculate the reconstruction loss (paper Eq. **7**).
  Calculate the total loss (paper Eq. **8**).
  The total loss backward and parameters update by Adam optimzer.
end for
Output the final model with parameters θ'.

signed module, such as the dense connection [3] and deformable convolution [2], we have not investigated yet.

References

[1] Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond. In *CVPR*, 2021. 1, 3

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

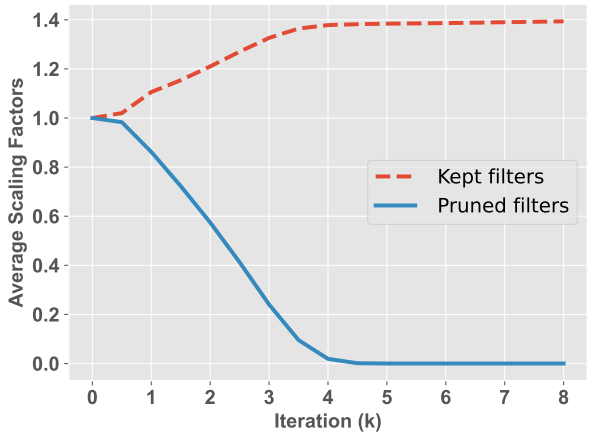
[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

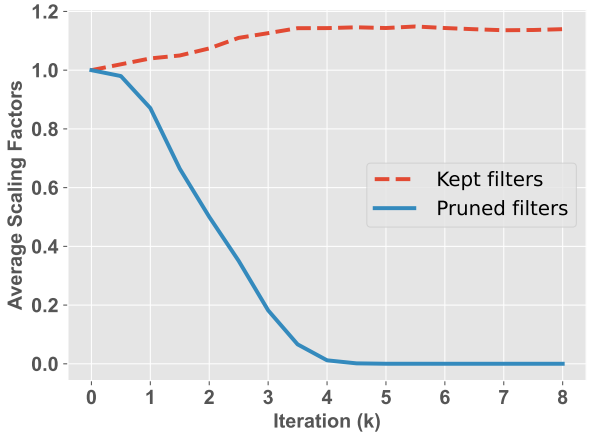
[5] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *TPAMI*, 2013. 5

[6] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPRW*, 2019. 1, 4, 5

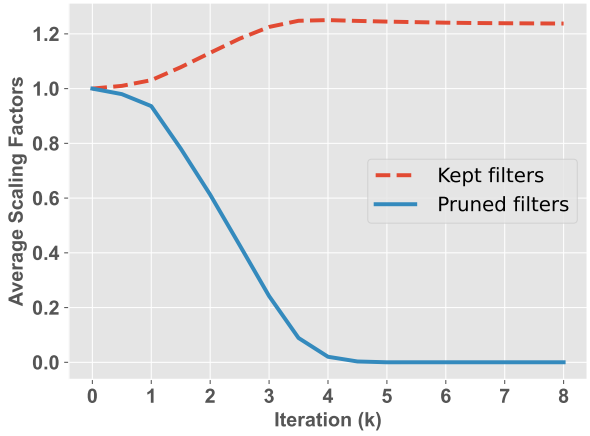
[7] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 2019. 1, 5



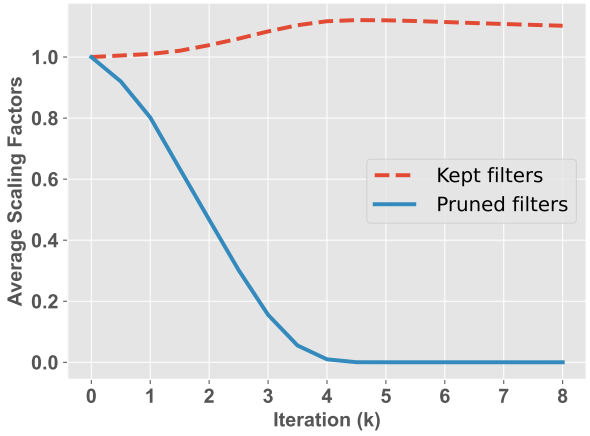
(a) backward_trunk.main.2.15.conv1



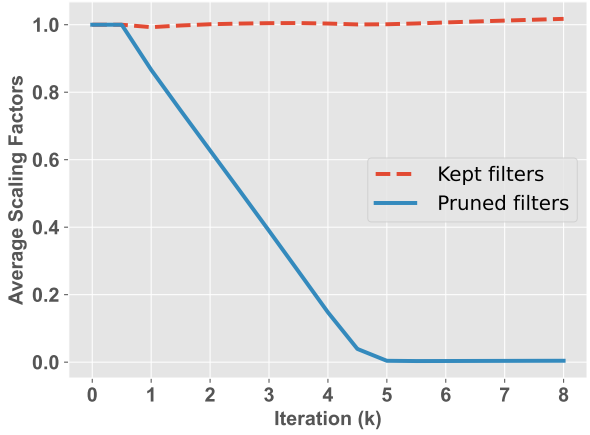
(b) backward_trunk.main.2.15.conv2



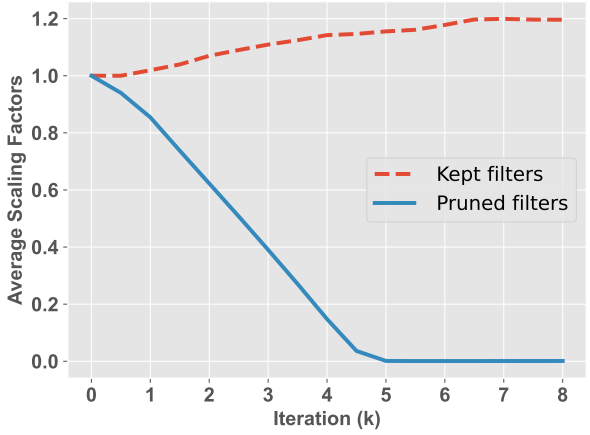
(c) forward_trunk.main.2.15.conv1



(d) forward_trunk.main.2.15.conv2

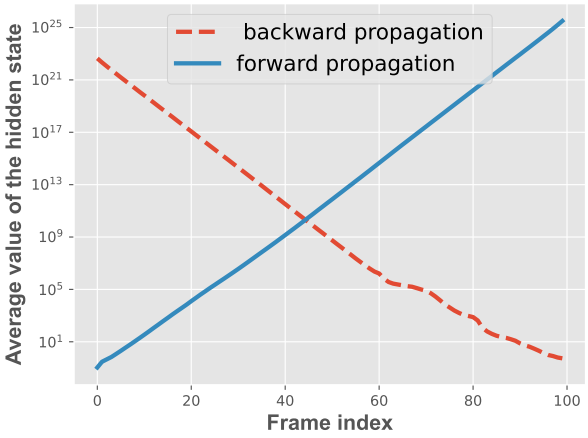


(e) upconv1

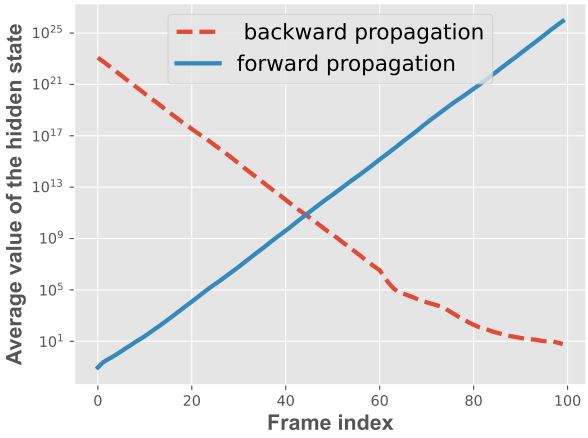


(f) upconv2

Figure 1. Illustration of the SSL pruning process of six convolution layers in the BasicVSR [1]. **(a)** and **(b)** are two convolutions in the 15-th residual block of the forward network. **(c)** and **(d)** are two convolutions in the 15-th residual block of the backward network. **(e)** and **(f)** are two convolutions before pixel-shuffle operation of the upsampling network.



(a) REDS4: 000



(b) REDS4: 011

Figure 2. The average value of the hidden state after pruning without finetuning. We apply SSL on BasicVSR at 0.5 pruning ratio, and test on 000 and 011 clips of REDS4 [6].

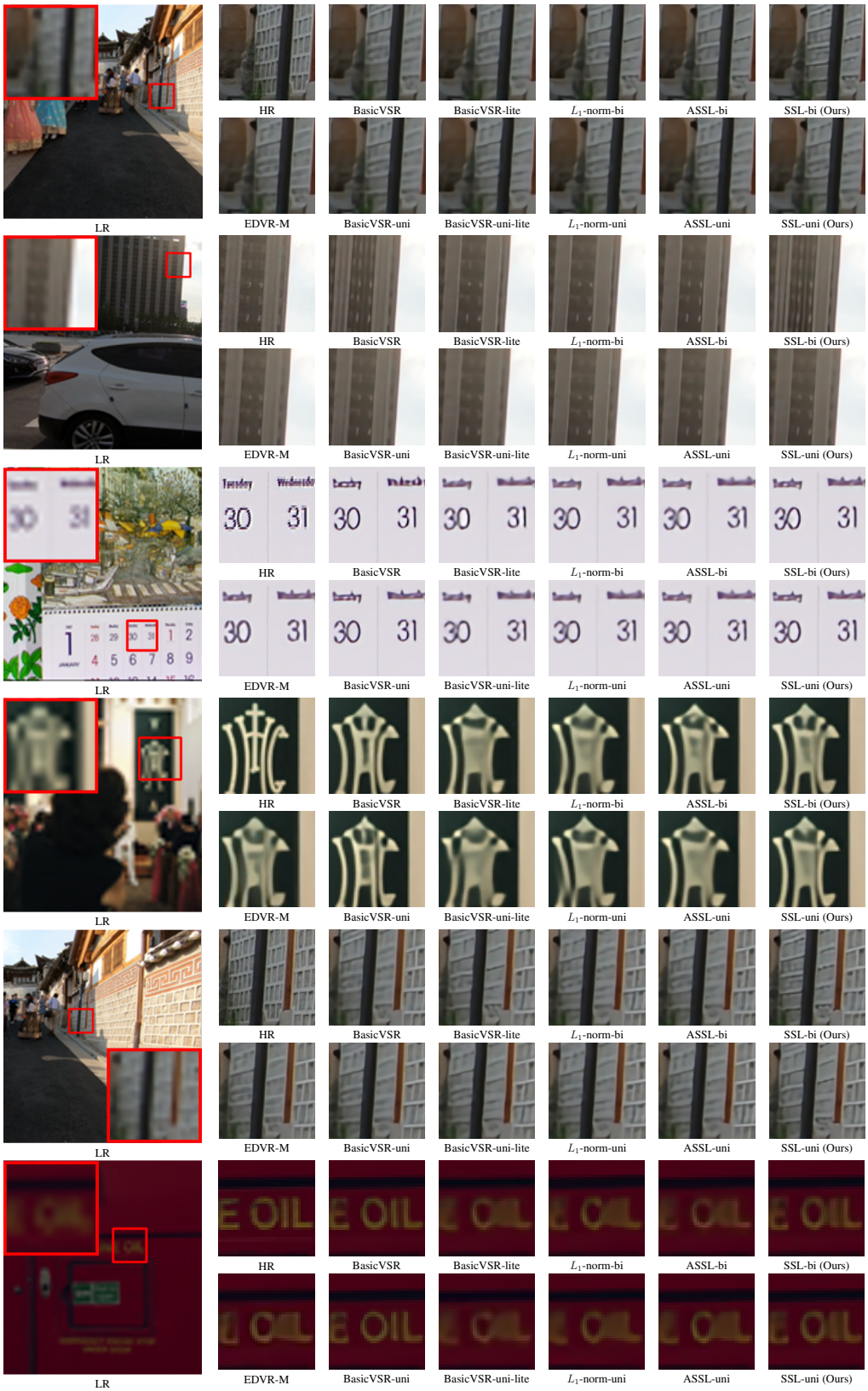


Figure 3. Qualitative comparison between different VSR and pruning methods on REDS4 [6], Vid4 [5], and Vimeo90K-T [7] benchmarks.