Towards High-Quality and Efficient Video Super-Resolution via Spatial-Temporal Data Overfitting

Gen Li¹†, Jie Ji¹†, Minghai Qin†, Wei Niu², Bin Ren², Fatemeh Afghah¹, Linke Guo¹, Xiaolong Ma¹
¹Clemson University ²William & Mary
{gen,xiaolom}@clemson.edu

Abstract

As deep convolutional neural networks (DNNs) are widely used in various fields of computer vision, leveraging the overfitting ability of the DNN to achieve video resolution upscaling has become a new trend in the modern video delivery system. By dividing videos into chunks and overfitting each chunk with a super-resolution model, the server encodes videos before transmitting them to the clients, thus achieving better video quality and transmission efficiency. However, a large number of chunks are expected to ensure good overfitting quality, which substantially increases the storage and consumes more bandwidth resources for data transmission. On the other hand, decreasing the number of chunks through training optimization techniques usually requires high model capacity, which significantly slows down execution speed. To reconcile such, we propose a novel method for high-quality and efficient video resolution upscaling tasks, which leverages the spatial-temporal information to accurately divide video into chunks, thus keeping the number of chunks as well as the model size to minimum. Additionally, we advance our method into a single overfitting model by a data-aware joint training technique, which further reduces the storage requirement with negligible quality drop. We deploy our models on an off-the-shelf mobile phone, and experimental results show that our method achieves real-time video super-resolution with high video quality. Compared with the state-of-the-art, our method achieves 28 fps streaming speed with 41.6 PSNR, which is 14× faster and 2.29 dB better in the live video resolution upscaling tasks. Code available in https://github.com/coulsonlee/STDO-CVPR2023.git.

1. Introduction

Being praised by its high image quality performance and wide application scenarios, deep learning-based super-resolution (SR) becomes the core enabler of many incredible, cutting-edge applications in the field of image/video reparation [10, 11, 39, 40], surveillance system enhancement [9], medical image processing [35], and high-quality video live streaming [20]. Distinct from the traditional methods that adopt classic interpolation algorithms [15, 45] to improve the image/video quality, the deep learning-based approaches [10, 11, 21, 24, 28, 40, 44, 47, 57, 60] exploit the advantages of learning a mapping function from low-resolution (LR) to high-resolution (HR) using external data, thus achieving better performance due to better generalization ability when meeting new data.

Such benefits have driven numerous interests in designing new methods [5, 17, 50] to deliver high-quality video stream to users in the real-time fashion, especially in the context of massive online video and live streaming available. Among this huge family, an emerging representative [13, 16, 31, 38] studies the prospect of utilizing SR model to upscale the resolution of the LR video in lieu of transmitting the HR video directly, which in many cases, consumes tremendous bandwidth between servers and clients [19]. One practical method is to deploy a pretrained SR model on the devices of the end users [25, 54], and perform resolution upscaling for the transmitted LR videos, thus obtaining HR videos without causing bandwidth congestion. However, the deployed SR model that is trained with limited data usually suffers from limited generalization abil-
In this work, we present a novel approach towards high-quality and efficient video resolution upsampling via Spatial-Temporal Data Overfitting, namely STDO, which for the first time, utilizes the spatial-temporal information to accurately divide video into chunks. Inspired by the work proposed in [1, 14, 23, 46, 58] that images may have different levels of intra- and inter-image (i.e., within one image or between different images) information density due to varied texture complexity, we argue that the unbalanced information density within or between frames of the video universally exists, and should be properly managed for data overfitting. Our preliminary experiment in Figure 1 shows that the PSNR values at different locations in a video frame forms certain pattern regarding the video content, and exhibits different patterns along the timeline. Specifically, at the server end, each frame of the video is evenly divided into patches, and then we split all the patches into multiple chunks by PSNR regarding all frames. Independent SR models will be used to overfit the video chunks, and then delivered to the clients. Figure 2 demonstrates the overview of our proposed method. By using spatial-temporal information for data overfitting, we reduce the number of chunks as well as the overfitting models since they are bounded by the nature of the content, which means our method can keep a minimum number of chunks regardless the duration of videos. In addition, since each chunk has similar data patches, we can actually use smaller SR model without handcrafted modules for the overfitting task, which reduces the computation burden for devices of the end-user. Our experimental results demonstrate that our method achieves real-time video resolution upsampling from 270p to 1080p on an off-the-shelf mobile phone with high PSNR.

Note that STDO encodes different video chunks with independent SR models, we further improve it by a Joint training technique (JSTDO) that results in one single SR model for all chunks, which further reduces the storage requirement. We design a novel data-aware joint training technique, which trains a single SR model with more data from higher information density chunks and less data from their counterparts. The underlying rationale is consistent with the discovery in [46, 58], that more informative data contributes majorly to the model training. We summarize...
our contributions as follows:

• We discover the unbalanced information density within video frames, and it universally exists and constantly changes along the video timeline.

• By leveraging the unbalanced information density in the video, we propose a spatial-temporal data overfitting method STDO for video resolution upscaling, which achieves outperforming video quality as well as real-time execution speed.

• We propose an advanced data-aware joint training technique which takes different chunk information density into consideration, and reduces the number of SR models to a single model with negligible quality degradation.

• We deploy our models on an off-the-shelf mobile phone, and achieve real-time super-resolution performance.

2. Related Works

2.1. Single Image Super Resolution (SISR)

For SISR tasks, SRCNN [10] is the pioneer of applying DNN to image super resolution. Then, followed by FSR-CNN [11] and ESPCN [40], both of them make progress in efficiency and performance. After this, with the development of deep neural networks, more and more network backbones are used for SISR tasks. For example, VDSR [21] uses the VGG [41] network as the backbone and adds residual learning to further improve the effectiveness. Similarly, SRResNet [24] proposed a SR network using ResNet [18] as a backbone. EDVR [49] uses the batch norm in residual blocks by finding that the use of batch norm will ignore absolute differences between image pixels (or features). WDSR [57] finds that ReLU will impede information transfer, so the growth of the number of filters before ReLU increases the width of the feature map. With the emergence of channel attention mechanisms networks represented by SENet [36], various applications of attention mechanisms poured into the area of image super resolution [7, 34, 60, 61]. After witnessing the excellent performance of transformer [12] in the field of computer vision, more and more researchers apply various vision transformer models into image super resolution tasks [3, 27, 32].

2.2. Video Super Resolution (VSR)

The VSR methods mainly learn from SISR framework [29]. Some aforementioned works like EDSR and WDSR are all present results on VSR. Some of the current VSR works perform alignment to estimate the motions between images by computing optical flow by DNNs [2, 22, 39, 42]. The Deformable convolution (DConv) [6] was first used to deal with geometric transformation in vision tasks because the sampling operation of CNN is fixed.

TDAN [43] applies DConv to align the input frames at the feature level, which avoids the two-stage process used in previous optical flow based methods. EDVR [49] uses their proposed Pyramid, Cascading and Deformable convolutions (PCD) alignment module and the temporal-spatial attention (TSA) fusion module to further improve the robustness of alignment and take account of the visual informativeness of each frame. Other works like DNLLN [48] and D3Dnnet [56] also apply Deconv in their model to achieve a better alignment performance.

2.3. Content-Aware DNN

It is impractical to develop a DNN model to work well on all the video from Internet. NAS [55] was the first proposed video delivery framework to consider using DNN models to overfit each video chunk to guarantee reliability and performance. Other livestreaming and video streaming works [4, 8, 20, 51, 53] leverage overfitting property to ensure excellent performance at the client end. [20] proposes a live video ingest framework, which adds an online learning module to the original NAS [55] framework to further ensure quality. NEMO [53] selects key frames to apply super-resolution. This greatly reduces the amount of computation on the client sides. CaFM [30] splits a long video into different chunks by time and design a handcrafted layer along with a joint training technique to reduce the number of SR models and improve performance. EMT [26] proposes to leverage meta-tuning and challenge patches sampling technique to further reduce model size and computation cost. SRVC [19] encodes video into content and time-varying model streams. Our work differentiates from these works by taking spatial information as well as temporal information into account, which exhibits better training effects for the overfitting tasks.

3. Proposed Method

3.1. Motivation

To tackle the limited generalization ability caused by using only one general SR model to super-resolve various videos, previous works [20, 30, 55] split the video by time and train separate SR models to overfit each of the video chunks. With more fine-grained video chunks over time, the better overfitting quality can be obtained, which makes these approaches essentially a trade-off between model expressive power and the storage efficiency. For a specific video, more chunks will surely improve the overfitting quality, but it also inevitably increases the data volume as well as system overhead when performing SR inference.

In the aforementioned methods, images are stacked according to the timeline to form the video. However, patches have spatial location information [37], and these patches are fed into the neural network indiscriminately for training,
which may cause redundancy that contradicts with overfitting property. As illustrated in Figure 1, when using a general SR model to super-resolve an LR video, the values of PSNR at different patch locations form a clear boundary, and are strongly related to the content of the current video frame (i.e., spatial information). Meanwhile, diverse boundary patterns can be seen in different frames (i.e., temporal information). This observation motivates us to use the spatial-temporal information to accurately divide video into chunks, which exhibits a different design space to overfit video data. With the different levels of information density within each patches, the key insight is to cluster patches that has similar texture complexity across all frames, and use one SR model to overfit each patch group. In this way, the number of video chunks and their associated SR models are effectively reduced, which improves the encoding efficiency regardless the duration of videos. Meanwhile, a compact SR model can be adopted without causing quality degradation because each SR model only overfits one specific video content with similar texture complexity. Additionally, when the spatial-temporal data is properly scheduled, our method can be extended to a joint training manner which generates a single SR model for the entire video.

### 3.2. Spatial-Temporal Data Overfitting

In this section, we introduce a novel spatial-temporal data overfitting approach, STDO, which efficiently encodes HR videos into LR videos and their corresponding SR models, and achieves outperforming super-resolution quality & speed at user end.

Suppose the video time length is $T$. General method to train an SR model would firstly divide the video into frames, and slice each frame into multiple non-overlapping image patches. All patches across all dimensions such as their locations in the frame or time compose a complete video. For a given video with the dimension $W \times H$, and the desired patch size $w \times h$, the patch is denoted as $P_{i,j,t}$, where $i \in [0, I)$, $j \in [0, J)$, and $t \in [0, T)$. Note that $I = \lfloor \frac{W}{w} \rfloor$ and $J = \lfloor \frac{H}{h} \rfloor$ are integer numbers, then the training set for this specific video is $D = \{P_n\}^N$ where $N = I \times J \times T$ is the total number of patches.

Note that $D$ contains all patches across all dimensions. We use a pretrained SR model $f_0(\cdot)$ to super-resolve all of the LR patches and compute their PSNRs with the HR patches. As illustrated in Figure 1, we find that the distribution of the PSNR is usually not uniform, and shows a clear boundary regarding the content of the video. We divide the training set $D$ into multiple chunks by grouping patches with similar PSNRs, and form a set of chunks as $\Omega = \{D_0, D_1, \ldots, D_k\}$, in which

$$\hat{D}_k = \{P_n | P_n \in D, PSNR(f_0(P_n)) \in [\lambda_{k_1}, \lambda_{k_2})\}, \quad (1)$$

where $\lambda$ is the threshold. We set the first chunk $\hat{D}_0$ to be group of the patches with lowest PSNRs, and we list all chunks in ascending order, which means $\hat{D}_k$ to be the patches with the highest PSNRs. In this way, we separate training data by their spatial-temporal information in a one-shot manner, which is usually done in seconds and can be considered negligible compared to the training process. In this paper, we empirically divide $D$ evenly. Finally, we train an SR model $f_{sr}(w_k; \hat{D}_k)$ to overfit each video chunk $\hat{D}_k$. Experimental results indicate better performance on both video quality and execution speed. Our empirical analysis is that by accurately identifying and grouping the data with similar information density (i.e., texture complexity) into chunks, each SR model becomes easier to “memorize” similar data in an overfitting task, and subsequently demands smaller SR models that can be executed in a real-time fashion.

### 3.3. Data-Aware Joint Training

In Section 3.2, our method significantly reduces the number of video chunks and overfitting SR models by effectively utilizing spatial-temporal information of each patch in the video. In this section, we extend our method by generating a single SR model for the entire video, which further reduces the storage requirement with negligible quality drop. From the set of chunks $\Omega \in \mathbb{R}^N$ and all SR models, we demonstrate PSNR in Table 1 by using only one SR model to super-resolve the entire video. Somehow surprisingly, we find out that using the model trained with $\hat{D}_0$ (i.e., the most informative chunk) experiences a moderate quality drop, and achieves similar or higher PSNR compared to the model trained with all patches. Meanwhile, the model trained with $\hat{D}_k$ has a severe quality degradation. We argue that low PSNR patches usually contain rich features, and contribute more to the model learning, which improves the generalization ability to the less informative data [46, 58].

Motivated by the above observation, we propose a joint training technique, which carefully schedules the spatial-temporal data participated in training to train a single SR model that overfits the entire video. Concretely, we keep all patches for $\hat{D}_0$, and remove the entire $\hat{D}_k$. For the rest of the chunks, we randomly sample a portion of the patches from each chunk, while gradually decreasing the proportion

### Table 1. Video super-resolution results comparison of vanilla STD0 training and using only one model from STDO trained with the most informative chunk $\hat{D}_0$ and least informative chunk $\hat{D}_k$, respectively. We also include the video super-resolution results with one model trained with all data [54].

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>vlog-15s</th>
<th>vlog-45s</th>
</tr>
</thead>
<tbody>
<tr>
<td>awDNN [54]</td>
<td>$\times 2$</td>
<td>49.24</td>
<td>45.30</td>
</tr>
<tr>
<td>WDSR</td>
<td>STDO</td>
<td>50.58</td>
<td>46.43</td>
</tr>
<tr>
<td>STDO$\hat{D}_0$</td>
<td>$\times 3$</td>
<td>50.42</td>
<td>45.99</td>
</tr>
<tr>
<td>STDO$\hat{D}_k$</td>
<td>$\times 4$</td>
<td>46.89</td>
<td>42.63</td>
</tr>
</tbody>
</table>
Table 2. Comparison results of STDO with different data overfitting methods on different SR backbones.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data Scale</th>
<th>game-15s $\times 2$</th>
<th>game-15s $\times 3$</th>
<th>game-15s $\times 4$</th>
<th>inter-15s $\times 2$</th>
<th>inter-15s $\times 3$</th>
<th>inter-15s $\times 4$</th>
<th>vlog-15s $\times 2$</th>
<th>vlog-15s $\times 3$</th>
<th>vlog-15s $\times 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPCN</td>
<td>awDNN [54]</td>
<td>37.94</td>
<td>32.85</td>
<td>29.97</td>
<td>40.43</td>
<td>35.36</td>
<td>29.91</td>
<td>46.41</td>
<td>42.90</td>
<td>39.65</td>
</tr>
<tr>
<td></td>
<td>NAS [55]</td>
<td>37.58</td>
<td>32.71</td>
<td>30.59</td>
<td>40.62</td>
<td>35.42</td>
<td>30.43</td>
<td>46.53</td>
<td>43.01</td>
<td>39.98</td>
</tr>
<tr>
<td></td>
<td>CaFM [30]</td>
<td>38.07</td>
<td>33.14</td>
<td>30.96</td>
<td>40.71</td>
<td>35.54</td>
<td>30.47</td>
<td>47.02</td>
<td>43.20</td>
<td>40.16</td>
</tr>
<tr>
<td>STDO</td>
<td>38.61</td>
<td>33.57</td>
<td>31.30</td>
<td>42.65</td>
<td>35.63</td>
<td>30.63</td>
<td>47.11</td>
<td>43.25</td>
<td>40.73</td>
<td></td>
</tr>
<tr>
<td>SRCNN</td>
<td>awDNN [54]</td>
<td>36.08</td>
<td>31.94</td>
<td>29.90</td>
<td>40.46</td>
<td>33.95</td>
<td>28.78</td>
<td>46.69</td>
<td>42.41</td>
<td>39.71</td>
</tr>
<tr>
<td></td>
<td>NAS [55]</td>
<td>36.27</td>
<td>32.08</td>
<td>29.94</td>
<td>40.70</td>
<td>34.01</td>
<td>28.84</td>
<td>46.78</td>
<td>42.53</td>
<td>39.76</td>
</tr>
<tr>
<td></td>
<td>CaFM [30]</td>
<td>36.63</td>
<td>32.21</td>
<td>29.98</td>
<td>40.76</td>
<td>34.08</td>
<td>29.93</td>
<td>46.98</td>
<td>42.62</td>
<td>39.81</td>
</tr>
<tr>
<td>STDO</td>
<td>37.59</td>
<td>32.67</td>
<td>30.64</td>
<td>42.28</td>
<td>34.26</td>
<td>30.05</td>
<td>47.06</td>
<td>42.78</td>
<td>39.90</td>
<td></td>
</tr>
<tr>
<td>VDSR</td>
<td>awDNN [54]</td>
<td>41.27</td>
<td>35.03</td>
<td>32.16</td>
<td>44.16</td>
<td>35.99</td>
<td>30.65</td>
<td>48.18</td>
<td>43.03</td>
<td>41.07</td>
</tr>
<tr>
<td></td>
<td>NAS [55]</td>
<td>42.53</td>
<td>35.97</td>
<td>33.86</td>
<td>44.71</td>
<td>36.57</td>
<td>31.05</td>
<td>48.49</td>
<td>43.41</td>
<td>41.33</td>
</tr>
<tr>
<td></td>
<td>CaFM [30]</td>
<td>43.13</td>
<td>37.04</td>
<td>34.47</td>
<td>45.35</td>
<td>38.66</td>
<td>35.70</td>
<td>49.30</td>
<td>45.03</td>
<td>43.12</td>
</tr>
<tr>
<td>STDO</td>
<td>43.56</td>
<td>36.71</td>
<td>35.02</td>
<td>45.16</td>
<td>36.81</td>
<td>33.43</td>
<td>48.75</td>
<td>43.82</td>
<td>41.71</td>
<td></td>
</tr>
<tr>
<td>EDSR</td>
<td>awDNN [54]</td>
<td>42.24</td>
<td>35.88</td>
<td>33.44</td>
<td>43.06</td>
<td>37.89</td>
<td>34.94</td>
<td>48.87</td>
<td>44.51</td>
<td>42.58</td>
</tr>
<tr>
<td></td>
<td>NAS [55]</td>
<td>42.82</td>
<td>36.42</td>
<td>34.00</td>
<td>45.06</td>
<td>38.38</td>
<td>35.47</td>
<td>49.10</td>
<td>44.80</td>
<td>42.83</td>
</tr>
<tr>
<td></td>
<td>CaFM [30]</td>
<td>43.13</td>
<td>37.04</td>
<td>34.47</td>
<td>45.35</td>
<td>38.66</td>
<td>35.70</td>
<td>49.30</td>
<td>45.03</td>
<td>43.12</td>
</tr>
<tr>
<td>STDO</td>
<td>44.93</td>
<td>37.80</td>
<td>35.47</td>
<td>45.91</td>
<td>39.26</td>
<td>36.76</td>
<td>50.24</td>
<td>45.68</td>
<td>43.46</td>
<td></td>
</tr>
<tr>
<td>WDSR</td>
<td>awDNN [54]</td>
<td>43.36</td>
<td>37.12</td>
<td>34.62</td>
<td>44.83</td>
<td>39.05</td>
<td>35.23</td>
<td>49.24</td>
<td>43.30</td>
<td>43.33</td>
</tr>
<tr>
<td></td>
<td>NAS [55]</td>
<td>44.17</td>
<td>38.23</td>
<td>36.02</td>
<td>45.43</td>
<td>39.71</td>
<td>36.54</td>
<td>49.98</td>
<td>45.63</td>
<td>43.51</td>
</tr>
<tr>
<td></td>
<td>CaFM [30]</td>
<td>44.23</td>
<td>38.55</td>
<td>36.30</td>
<td>45.71</td>
<td>39.92</td>
<td>36.87</td>
<td>50.12</td>
<td>45.87</td>
<td>43.79</td>
</tr>
<tr>
<td>STDO</td>
<td>45.75</td>
<td>40.17</td>
<td>38.62</td>
<td>46.34</td>
<td>41.13</td>
<td>38.76</td>
<td>50.58</td>
<td>46.43</td>
<td>44.62</td>
<td></td>
</tr>
</tbody>
</table>

of the data sampled. We train a single model by solving the following optimization problem using the joint dataset:

\[
\begin{align*}
\text{minimize} & \quad f_{\text{joint}}(w; D_{\text{joint}}) \\
\text{subject to} & \quad D_{\text{joint}} \in \{ \hat{D}_0, \rho_1 \odot \hat{D}_1, \ldots, \rho_{k-1} \odot \hat{D}_{k-1} \}, \\
& \quad \sum_{i=0}^{k-1} \| \rho_i \odot \hat{D}_i \| = \mu,
\end{align*}
\]

(2)

where $\odot$ is the sampling operation with pre-defined proportion $\rho$, and $\mu$ is a hyper-parameter that controls the size of the joint dataset.

4. Experimental Results

In this section, we conduct extensive experiments to prove the advantages of our proposed methods. To show the effects of our methods, we apply our proposed STDO and JSTDO to videos with different scenes and different time lengths. The detailed information on video datasets and implementations are shown in Section 4.1. In Section 4.2, we compared our method with time-divided method using different videos and different SR models, which show that STDO achieves outperforming video super-resolution quality as well as using lowest computation budgets. In section 4.3, we demonstrate the results of our single SR model obtained by JSTDO and show that JSTDO effectively exploits heterogeneous information density among different video chunks to achieve better training performance. In Section 4.4, we deploy our model on an off-the-shelf mobile phone to show our model can achieve real-time video super-resolution. In Section 4.5, we show our ablation study on key parameters used in STDO and JSTDO methods, such as the different number of chunks, training data scheduling,
Table 3. Computation cost for different backbones with VSD4K video game-45s. We include the computation cost for the models with different resolution upscaling factors.

<table>
<thead>
<tr>
<th>Model</th>
<th>Scale</th>
<th>FLOPs</th>
<th>CaFM [30]</th>
<th>STDO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPCN</td>
<td>×2</td>
<td>0.14G</td>
<td>36.09</td>
<td>37.75</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>0.15G</td>
<td>31.06</td>
<td>32.29</td>
</tr>
<tr>
<td></td>
<td>×4</td>
<td>0.16G</td>
<td>29.05</td>
<td>29.96</td>
</tr>
<tr>
<td>SRCNN</td>
<td>×2</td>
<td>0.64G</td>
<td>35.49</td>
<td>36.74</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>1.45G</td>
<td>30.63</td>
<td>31.46</td>
</tr>
<tr>
<td></td>
<td>×4</td>
<td>2.58G</td>
<td>28.66</td>
<td>29.37</td>
</tr>
<tr>
<td>VDSR</td>
<td>×2</td>
<td>6.15G</td>
<td>41.92</td>
<td>42.65</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>13.85G</td>
<td>35.56</td>
<td>36.23</td>
</tr>
<tr>
<td></td>
<td>×4</td>
<td>20.62G</td>
<td>33.16</td>
<td>33.76</td>
</tr>
<tr>
<td>EDSR</td>
<td>×2</td>
<td>3.16G</td>
<td>43.32</td>
<td>45.65</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>3.60G</td>
<td>37.19</td>
<td>39.93</td>
</tr>
<tr>
<td></td>
<td>×4</td>
<td>4.57G</td>
<td>34.61</td>
<td>37.24</td>
</tr>
<tr>
<td>WDSR</td>
<td>×2</td>
<td>2.73G</td>
<td>43.97</td>
<td>45.71</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>2.74G</td>
<td>37.64</td>
<td>40.33</td>
</tr>
<tr>
<td></td>
<td>×4</td>
<td>2.76G</td>
<td>35.12</td>
<td>37.76</td>
</tr>
</tbody>
</table>

and long video with multiple scene conversions.

4.1. Datasets and Implementation Details

In the previous video super-resolution works, most video datasets [33, 52] for super-resolution only provide several adjacent frames as a mini-video. Those mini-video sets are not suitable for a network to overfit. Therefore, we adopt the VSD4K collected in [30]. In this video dataset, there are 6 video categories including: vlog, game, interview, city, sports, dance. Each of the category contains various video lengths. We set the resolution for HR videos to 1080p, and LR videos are generated by bicubic interpolation to match different scaling factors.

We apply our approach to several popular SR models including ESPCN [40], SRCNN [10], VDSR [21], EDSR [28], and WDSR [57]. During training, we use patch sizes of 48×54 for scaling factor 2 and 4, and 60×64 for scaling factor 3 to accommodate HR video size, and the threshold value $\lambda$ is set to split the patches evenly into chunks. Regarding the hyperparameter configuration of training the SR models, we follow the setting of [28, 30, 57]. We adopt Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.009$, $\epsilon = 10^{-8}$ and we use L1 loss as loss function. For learning rate, we set $10^{-3}$ for WDSR and $10^{-4}$ for other models with decay at different training epochs. We conduct our experiment on EDSR model with 16 resblocks and WDSR model with 16 resblocks.

4.2. Compare with the State-of-the-Art Methods

In this section, we compare our method with the state-of-the-art (SOTA) that either use general model overfitting or time-divided model overfitting on different SR backbones. Due to space limits, we sample three video categories from VSD4K, and test on two different video time lengths as 15s and 45s. Our results are shown in Table 2. We compare with the state-of-the-art neural network-based SR video delivery methods, such as awDNN [54] where a video is overfitted by a model, NAS [55] that splits a video into multiple chunks in advance and overfit each of the time-divided chunk with independent SR model, and CaFM [30] that uses time-divided video chunk and single SR model with handcrafted module to overfit videos. For our implementation of STDO, we divide the spatial-temporal data into 2, 4, and 6 chunks respectively, and report the best results. We adjust batch size while training to keep the same computation cost, and we show the comparison by computing the PSNR of each method. It can be seen that our method can exceed the SOTA works consistently on different backbones.

With STDO, each SR model is only overfitting one video chunk that has similar information density, which makes it suitable to use smaller and low capacity SR models that has low computations. In Table 3, we demonstrate the computation cost on each model. From the results, we notice that with the relatively new model such as VDSR, EDSR, and WDSR, when the computation drops below 3 GFLOPs, time-divided method experiences significant quality degradation, while STDO maintains its performance or even achieves quality improvements. When using extremely small networks such as ESPCN or SRCNN, time-divided methods PSNR drops quickly, while STDO still achieves 0.7 ~ 1.7 dB better performance.

4.3. Data-Aware Joint Training

In this part, we show the results of reducing the number of SR models to a single model by data-aware joint training with the spatial-temporal data overfitting (JSTDO) in
Table 4. Comparison between STDO and JSTDO regarding PSNR and total number of model parameters with game-15s from VSD4K. We compute the PSNR difference of the two methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>#Chunks</th>
<th>#Models</th>
<th>#Param.</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDSR × 2</td>
<td>STDO 4</td>
<td>4</td>
<td>4.8M</td>
<td>45.75</td>
<td></td>
</tr>
<tr>
<td>JSTDO 4</td>
<td>1</td>
<td>1.2M</td>
<td>45.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ΔPSNR: 0.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDSR × 3</td>
<td>STDO 4</td>
<td>4</td>
<td>4.8M</td>
<td>40.17</td>
<td></td>
</tr>
<tr>
<td>JSTDO 4</td>
<td>1</td>
<td>1.2M</td>
<td>39.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ΔPSNR: 0.30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDSR × 4</td>
<td>STDO 4</td>
<td>4</td>
<td>4.8M</td>
<td>38.62</td>
<td></td>
</tr>
<tr>
<td>JSTDO 4</td>
<td>1</td>
<td>1.2M</td>
<td>38.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ΔPSNR: 0.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the results, we can clearly see that JSTDO effectively reduces the overall model size from 4.8 MB to 1.2 MB, while the PSNR of JSTDO has only negligible degradation compared with STDO. Please note that even with some minor quality degradation, JSTDO still outperforms baseline methods in both super-resolution quality and total parameter counts.

We also plot the PSNR heatmaps when using STDO and JSTDO for video super-resolution, and compare with the traditional time-divided method. As showing in Figure 3, we randomly select one frame in a vlog-15s video that is super-resolved from 270p to 1080p (×4) by CaFM [30], STD, and JSTDO. The heatmaps clearly demonstrate that our methods achieve better PSNR in the key content zone in the frame. Meanwhile, another key observation can be drawn: the JSTDO heatmap has similar patterns with the one using STDO, which further proves that the joint training technique using carefully scheduled spatial-temporal data effectively captures the key features, while not losing the expressive power towards the low information density data. We also show the qualitative comparison in Figure 4.

4.4. Deployment on Mobile Devices

One of the many benefits by using STDO is that we can use smaller (i.e., low model capacity & complexity) SR models to perform data overfitting. The reason is that the patches in each chunk are relatively similar, especially for some short videos, which makes it easier for smaller models to “memorize” them. Subsequently, unlike CaFM [30], no handcrafted modules are needed for both STDO and JSTDO methods, which further reduces the compilation burden on the end-user devices.

We deploy the video chunks alongside with the overfit-
ting models of STDO on a Samsung Galaxy S21 with Snapdragon 888 to test execution performance. Each patch will have a unique index to help assemble into frames. Our results are shown in Figure 5. We set the criteria for real-time as latency less than 50 ms and FPS greater than 20 on the mobile devices according to [59]. The result shows that our method achieves 28 FPS when super-resolving videos from 270p to 1080p by WDSR, and it is significantly faster in speed and better in quality than other models such as EDSR or VDSR that are originally used in other baseline methods [30, 53–55]. Please note that the capability of using small scale SR models to accelerate execution speed is ensured by the high super-resolution quality achieved by spatial-temporal data overfitting method.

4.5. Ablation Study

▷ Different number of chunks in STDO. Previous experiments have proved that STDO brings performance improvement when we take account of the spatial-temporal information. In this ablation study, we vary the number of chunks and evaluate their video super-resolution quality. We set the number of chunks with the range of 1 (i.e., single model overfitting) to 8, and we plot the PSNR trends using ESPCN and WDSR in Figure 7a. We observe that ESPCN and WDSR demonstrate similar trends when the number of chunks increases, and better results can be obtained when we divide video into ∼4 chunks, which consolidates our claim that STDO uses fewer chunks compared to time-divided methods.

▷ Data scheduling in joint training. In JSTDO, we vary the sampling proportion by increasing patches from $D_0$ while decreasing the proportion of patches in $D_k$, and adjusting sampling proportion for other chunks accordingly to maintain the same amount of training patches. The evaluation results are shown in Figure 7b, where we observe that when more informative data participates in training, the overall video super-resolution quality increases. Same patterns can be seen in Figure 6, where the heatmaps show relatively high PSNR in the key content zone when the SR model is trained with more informative data.

▷ Long video with multiple scene conversions. We combine the game-45s video and the vlog-45s video together into a 90s long video which contains multiple scene conversions. The STDO result of the long video trained on WDSR is 39.92 dB with scaling factor 4, which is close to the average value of overfitting two videos (43.99 dB and 37.76 dB). Therefore, it can be proved that our design can still maintain good performance for long videos where multiple scene conversions exist.

5. Conclusion

In this paper, we introduce a novel spatial-temporal data overfitting approach towards high-quality and efficient video resolution upscaling tasks at the user end. We leverage the spatial-temporal information based on the content of the video to accurately divide video into chunks, then overfit each video chunk with an independent SR model or use a novel joint training technique to produce a single SR model that overfits all video chunks. We successfully keep the number of chunks and the corresponding SR models to a minimum, as well as obtaining high super-resolution quality with low capacity SR models. We deploy our method on the mobile devices from the end-user and achieve real-time video super-resolution performance.

6. Acknowledgment

This work is partly supported by the National Science Foundation IIS-1949640, CNS-2008049, CNS-2232048, and CNS-2204445, and Air Force Office of Scientific Research FA9550-20-1-0090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and AFOSR.
References


