Self-Conditioned Probabilistic Learning of Video Rescaling

Yuan Tian\textsuperscript{1} Guo Lu\textsuperscript{2} Xiongkuo Min\textsuperscript{1} Zhaohui Che\textsuperscript{1} Guangtao Zhai\textsuperscript{1,}\textsuperscript{*} Guodong Guo\textsuperscript{3} Zhiyong Gao\textsuperscript{1}

\textsuperscript{1}Shanghai Jiao Tong University \textsuperscript{2}Beijing Institute of Technology \textsuperscript{3}Baidu
{ee.tianyuan, minxiongkuo, chezhaohui, zhaiguangtao, zhiyong.gao}@sjtu.edu.cn

DOI 10.1109/ICCV48922.2021.00445

Abstract

Bicubic downscaling is a prevalent technique used to reduce the video storage burden or to accelerate the downstream processing speed. However, the inverse upscaling step is non-trivial, and the downsampled video may also deteriorate the performance of downstream tasks. In this paper, we propose a self-conditioned probabilistic framework for video rescaling to learn the paired downscaling and upscaling procedures simultaneously. During the training, we decrease the entropy of the information lost in the downscaling by maximizing its probability conditioned on the strong spatial-temporal prior information within the downsampled video. After optimization, the downsampled video by our framework preserves more meaningful information, which is beneficial for both the upscaling step and the downstream tasks, e.g., video action recognition task. We further extend the framework to a lossy video compression system, in which a gradient estimator for non-differential industrial lossy codecs is proposed for the end-to-end training of the whole system. Extensive experimental results demonstrate the superiority of our approach on video rescaling, video compression, and efficient action recognition tasks.

1. Introduction

High-resolution videos are widely used over various computer vision tasks \cite{62,7,35,33,50,64}. However, considering the increased storage burden or the high computational cost, it is usually required to first downscale the high-resolution videos. Then we can either compress the output low-resolution videos for saving storage cost or feed them to the downstream tasks to reduce the computational cost. Despite that this paradigm is prevalent, it has the following two disadvantages. First, it is non-trivial to restore the original high-resolution videos from the (compressed) low-resolution videos, even we use the latest super-resolution methods \cite{30,63,48,57}. Second, it is also a challenge for the downstream tasks to achieve high performance based on these low-resolution videos. Therefore, it raises the question that whether the downsampling operation can facilitate the reconstruction of the high-resolution videos and also preserve the most meaningful information.

Recently, this question has been partially studied as a single image rescaling problem \cite{24,27,47,60}, which learns the down/up scaling operators jointly. However, how to adapt these methods from image to video domain and leverage the rich temporal information within videos are still open problems. More importantly, modeling the lost information during downscaling is non-trivial. Current methods either ignore the lost information \cite{24,27,47} or assume it as an independent distribution in the latent space \cite{60}, while neglecting the internal relationship between the downsampled image and the lost information. Besides, all literature above has not explored how to apply the rescaling technique to the lossy image/video compression.

In this paper, we focus on building a video rescaling framework and propose a self-conditioned probabilistic learning approach to learn a pair of video downscaling and upscaling operators by exploiting the information dependency within the video itself. Specifically, we first design a learnable frequency analyzer to decompose the original high-resolution video into its downsampled version and the corresponding high-frequency component. Then, a Gaussian mixture distribution is leveraged to model the high-frequency component by conditioning on the downsampled video. For accurate estimation of the distribution parameters, we further introduce the local and global temporal aggregation modules to fuse the spatial information from adjacent downsampled video frames. Finally, the original video can be restored by a frequency synthesizer from the downsampled video and the high-frequency component sampled from the distribution. We integrate the components above as a novel self-conditioned video rescaling framework termed SelfC and optimize it by minimizing the negative log-likelihood for the distribution.

Furthermore, we apply our proposed SelfC in two practical applications, \textit{i.e.}, lossy video compression and video action recognition. In particular, to integrate our framework with the existing non-differential video codecs \textit{(e.g.,...}

\textsuperscript{*}Corresponding author.
We propose a probabilistic learning framework dubbed SelfC for the video rescaling task, which models the lost information during downscaling as a dynamic distribution conditioned on the downsampled video. Our approach exploits rich temporal information in downsampled videos for an accurate estimation of the distribution parameters by introducing the specified local and global temporal aggregation modules. We propose a gradient estimation method for non-differential lossy codecs based on the control variates method and Monte Carlo sampling technique, extending the framework to a video compression system.

2. Related Work

Video Upscaling after Downscaling. Previous SR works [30, 63, 48, 57, 17, 23] mainly leverage a heavy neural network to hallucinate the lost details during downscaling, only achieving unsatisfactory results. Taking the video downscaling model into consideration may help mitigate the ill-posedness of the video upsampling procedure.

There are already a few works on single image rescaling task following this spirit. For example, Kim et al. [24] proposed a task-aware downscaling model based on an auto-encoder framework. Li et al. [27] proposed to use a neural network to estimate the downsampled low-resolution images for a given super-resolution method. Yang et al. [65] proposed a superpixel-based downsampling/upsampling scheme to effectively preserve object boundaries. Recently, Xiao et al. [60] proposed to leverage a deep invertible neural network (INN) to model the problem, which maps the complex distribution of the lost information to an independent and fixed normal distribution.

However, these methods neither leverage the temporal information between adjacent frames, which is important for video-related tasks, nor consider the fact that the components of different frequencies in natural images or videos are conditionally dependent [53, 45, 38, 54].

Video Compression. Several traditional video compression algorithms have been proposed and widely deployed, such as H.264 [59] and H.265 [46]. Most of them follow the predictive coding architecture and rely on the sophisticated hand-crafted transformations to analyze the redundancy within the videos. Recently, fully end-to-end video codecs [32] [15][6][66][2][29] [31][19][34] have been proposed by optimizing the rate-distortion trade-off. They demonstrate promising performance and may be further improved by feeding more ubiquitous videos in the wild. However, they haven’t been widely used by industrial due to computational inefﬁciency. In contrast, our framework can be readily integrated with the best traditional video codecs and further saves the video storage space signiﬁcantly.

Video Action Recognition. Simonyan et al. [44] first proposed the two-stream framework. Feichtenhofer et al. [9] then improved it. Later, wang et al. [56] proposed a new sparse frame sampling strategy. Recently, 3D networks [51][5][16][52][39][8][49] also show promising performance. Our work can accelerate the off-the-shelf action CNNs by 3-4 times while preserving the comparable performance. We mainly conduct experiments on light-weight 2D action CNNs (e.g., TSM [28]) for efficiency.

3. Proposed Method

An overview of our proposed SelfC framework is shown in Fig. 1 (a). During the downscaling procedure, given a high-resolution (HR) video, a frequency analyzer (FA) (Section 3.1) first converts it into video features \( f \), where the first 3 channels are low-frequency (LF) component \( f_l \), the last \( 3 \cdot k^2 \) channels are high-frequency (HF) component \( f_h \), and \( k \) is the downscaling ratio. Then, \( f_l \) is quantized to a LR video \( x_l \) for storage. \( f_h \) is discarded in this procedure.

During the upsampling procedure, given the LR video \( x_l \), the spatial-temporal prior network (STP-Net) (Section 3.3) predicts the probability density function of the HF component \( f_h \):

\[
p(f_h|x_l) = \text{STP-Net}(x_l).
\]  

We model \( p(f_h|x_l) \) as a continuous mixture of the parametric Gaussian distributions (Section 3.2). Then, a case of the HF component \( f_h \) related to LR video \( x_l \) is drawn from the distribution. Finally, we reconstruct the HR video from the concatenation of HF component \( f_h \) and LR video \( x_l \) by the frequency synthesizer (FS).

3.1. Frequency Analyzer and Synthesizer

As shown in Fig. 1 (b), we first decompose the HR input video \( x \) as the \( k \) times downsampled low-frequency component \( c_l := \text{Down}(x) \in \mathbb{R}^{l \times 3 \times \frac{h}{k} \times \frac{w}{k}} \) and the residual high-frequency component \( c_h := \text{PixelUnshuffle}(x - \text{Up}(c_l)) \in \mathbb{R}^{l \times 3 \cdot k^2 \times \frac{h}{k} \times \frac{w}{k}} \), where \( h \times w \) denotes the spatial scale of the
original video and \( t \) denotes the video length. Down and Up represent bicubic downsampling and upsampling operations with the scaling ratio \( k \). PixelUnshuffle is the inverse operation of the pixel shuffling operation proposed in [43], where the scaling ratio is also \( k \). Then, we use a learnable operation \( \mathcal{T} \) to transform \( c_l \) and \( c_h \) to the output features \( f = \mathcal{T}(c_l \oplus c_h) \), where \( \oplus \) denotes the channel concatenation operation. The produced video feature \( f \) consists of LF component \( f_l \) and HF component \( f_h \). The network architecture for \( \mathcal{T} \) is very flexible in our framework and we use multiple stacking Dense2D-T blocks to implement it by default. The Dense2D-T block is modified from the vanilla Dense2D block [20] by replacing the last spatial convolution with the temporal convolution.

The architecture of the frequency synthesizer is symmetric with the analyzer, as shown in Fig. 1 (c). Specifically, we use channel splitting, bicubic upsampling, and pixel shuffling operations to synthesize the final high-resolution videos based on the reconstructed video feature \( \hat{f} \).

### 3.2. A Self-conditioned Probabilistic Model

Directly optimizing \( p(f_h | x_l) \) in Eq. (1) through gradient descent is unstable due to the unsmooth gradient [24] of the quantization module. Thus, we optimize \( p(f_h | f_l) \) instead during the training procedure. Specifically, we represent the high-frequency component \( f_h \) as a continuous multi-modal probability distribution conditioned on the low-frequency component \( f_l \), which is formulated as:

\[
p(f_h | f_l) = \prod_{o} p(f_h(o) | f_l),
\]

where \( o \) denotes the spatial-temporal location. We use a continuous Gaussian Mixture Model (GMM) [40] to approximate \( p \) with component number \( K = 5 \). The distributions are defined by the learnable mixture weights \( w_{o}^{k} \), means \( \mu_{o}^{k} \) and log variances \( \sigma_{o}^{k} \). With these parameters, the distributions can be accurately defined as:

\[
p(f_h(o) | f_l) = \sum_{k=1}^{K} w_{o}^{k} p_{y}(f_h(o) | \mu_{o}^{k}, \sigma_{o}^{k}),
\]

where

\[
p_{y}(f | \mu, \sigma^{2}) = \frac{1}{\sqrt{2\pi} \sigma} e^{-\frac{(f - \mu)^2}{2\sigma^2}}.
\]

### 3.3. Spatial-temporal Prior Network (STP-Net)

As shown in Fig. 1 (d), to estimate the parameters of the distribution above, we propose the STP-Net to model both the local and global temporal information within the downscaled video. We first utilize the Dense2D-T block to extract the short-term spatial-temporal features for each input frame. In this stage, only information from local frames, i.e., the previous or the next frames, are aggregated into the current frame, while the temporally long-range dependencies in videos are neglected. Therefore, we further introduce the attention mechanism for modeling the global temporal information. More specifically, the spatial dimension of the short-term spatial-temporal features is first reduced...
by a spatial aggregator, which is implemented as an average pooling operation followed by a full-connected (FC) layer. The output scale of the pooling operation is 32×32. Then we use the dot-producing operation to generate the attention map, which represents the similarity scores between every two frames. Finally, we refine the local spatial-temporal features based on the similarity scores. We repeat the following procedure six times to extract better video features. After that, a three-layer multi-layer perceptron (MLP) is used to estimate the parameters of the GMM distribution.

3.4. Quantization and Storage Medium

We use rounding operation as the quantization module, and store the output LR videos by lossless format, i.e., H.265 lossless mode. The gradient of the module is calculated by the Straight-Through Estimator [3]. We also discuss how to adapt the framework to more practical lossy video formats such as H.264 and H.265 in Section 3.6.

3.5. Training Strategy

Building a learned video rescaling framework is non-trivial, especially the generated low-resolution videos are expected to benefit both the upscaling procedure and the downstream tasks. We consider the following objectives.

**Learning the self-conditioned probability.** First, to make sure the STP-Net can obtain an accurate estimation for the HF component $f_h$, we directly minimize the negative log-likelihood of $p(f_h|f_l)$ in Eq. (2):

$$L_c = -\sum_{i=0}^{N} \log(p(f_h^{i}|f_l^{i})), \quad (5)$$

where $N$ is the number of the training samples.

**Mimicking Bicubic downscaling.** Then the downscaled video is preferred to be similar to the original video, making its deployment for the downstream tasks easier. Therefore, we regularize the downscaled video before quantization, i.e., $f_l$, to mimic the bicubic downsampled $x$:

$$L_{\text{mimic}} = ||x_{\text{bicubic}} - f_l||_2, x_{\text{bicubic}} = \text{Bicubic}(x). \quad (6)$$

**Penalizing $\mathcal{T}$.** Without any extra constraint, Eq. (5) can be easily minimized by tuning $f_h$ to one constant tensor for any input video. Thus, to avoid the trivial solution, the CNN parts of the frequency analyzer and synthesizer are penalized by the photo-parametric loss (i.e., $\ell_2$ loss) between the video directly reconstructed from $f$ and the original input $x$:

$$L_{\text{pen}} = ||x - \text{FS}(f)||_2. \quad (7)$$

**Minimizing reconstruction difference.** Finally, the expected difference between the reconstructed video sampled from the model and the original video should be minimized:

$$L_{\text{recons}} = \ell(x, \hat{x}), \hat{x} = \text{FS}(x_l \circ f_h), \quad (8)$$

where $\ell$ denotes a photo-metric loss (i.e., $\ell_1$ loss), $\circ$ denotes the channel-wise concatenation operation. In each training iteration, $f_h$ is sampled from the distribution constructed from the parameters output by STP-Net, conditioning on the LR video $x_l$. To enable an end-to-end optimization, we apply the “reparametrization trick” [26, 41, 14] to make the sampling procedure differentiable.

The total loss is then given by:

$$L_{\text{selfc}} = \lambda_1 L_c + \lambda_2 \cdot k^2 L_{\text{mimic}} + \lambda_3 L_{\text{pen}} + \lambda_4 L_{\text{recons}}, \quad (9)$$

where $\lambda_1$, $\lambda_2$, $\lambda_3$ and $\lambda_4$ are the balancing parameters, and $k$ is the scaling ratio. The loss function of our framework may seem a little bit complicated. However, we want to mention that the performance of our framework is not sensitive to these hyper-parameters, and directly setting all the parameters to 1 already achieves reasonable performance.

3.6. Application I: Video Compression

In this section, we extend the proposed SelfC framework to a lossy video compression system, which aims to demonstrate the effectiveness of our approach in reducing the video storage space. The whole system is shown in Fig. 2. Specifically, we first use the SelfC to generate the downscaled video $x_l$, which will be compressed by using the off-the-shelf industrial video codecs, e.g., H.265 codec. Then at the decoder side, the compressed videos will be decompressed and upscaled to the full resolution video.

Considering the traditional video codecs are non-differential, we further propose a novel optimization strategy. Specifically, we introduce a differentiable surrogate video perturbator $\phi$, which is implemented as a deep neural network (DNN) consisting of 6 Dense2D-T blocks. During the back-propagation stage, the gradient of the codec can be approximated by that of $\phi$, which is tractable. During the test stage, the surrogate DNN is removed and we directly use the H.265 codec for compression and decompression.

According to the control variates theory [11, 13], $\phi$ can be a low-variance gradient estimator for the video codec (i.e., $\eta$) when (1) the differences between the outputs of the two functions are minimized and (2) the correlation coefficients $\rho$ of the two output distributions are maximized.

Therefore, we introduce these two constraints to the optimization procedure of the proposed SelfC based video compression system. The loss function for the surrogate video codec as an example.
perturbator is formulated as:

\[ \mathcal{L}_{\text{codec}} = ||\eta(x_i) - \phi(x_i)||^2_2 - \lambda_\rho \rho(\eta, \phi), \quad (10) \]

where \( \lambda_\rho \) is set to a small value, \( i.e., 10^{-5} \), and \( \rho \) is estimated within each batch by Monte Carlo sampling:

\[ \rho(\eta, \phi) = \frac{\sum_{k=1}^{N}(\eta(x_i^k) - \mathbb{E}[\eta])(\phi(x_i^k) - \mathbb{E}[\phi])}{\sqrt{\sum_{k=1}^{N}(\eta(x_i^k) - \mathbb{E}[\eta])^2 \sum_{k=1}^{N}(\phi(x_i^k) - \mathbb{E}[\phi])^2}}, \quad (11) \]

where

\[ \mathbb{E}[\eta] = \frac{1}{N} \sum_{k=1}^{N} \eta(x_i^k), \quad \mathbb{E}[\phi] = \frac{1}{N} \sum_{k=1}^{N} \phi(x_i^k), \quad (12) \]

and \( N \) denotes the batch size. Note that \( \rho \) is separately computed on each spatial-temporal location and then averaged.

Finally, the total loss function for the SelfC based video compression system is given by:

\[ \mathcal{L}_{\text{compression}} = \mathcal{L}_{\text{selfC}} + \lambda_\text{codec}\mathcal{L}_{\text{codec}}, \quad (13) \]

where \( \lambda_\text{codec} \) is the balancing weight.

### 3.7. Application II: Efficient Action Recognition

We further apply the proposed SelfC framework to the video action recognition task. Specifically, we adopt the LR videos (\( i.e., x_i \)) downscaled by our framework as the input of action recognition CNNs for efficient action recognition. Considering the downsampler of our approach can preserve meaningful information for the downstream tasks and the complexity of itself can be rather low, inserting the downsampler before the off-the-shelf action CNNs can reduce the huge computational complexity of them with negligible performance drop. Moreover, the light-weightness of the rescaling framework makes the joint optimization tractable. In fact, compared with bicubic downsampling operation, our downsampler in SelfC framework can still generate more informative low-resolution videos for the action recognition task even without the joint training procedure. Please see Section 4.5 for more experimental results.

### 4. Experiments

#### 4.1. Dataset

We use the videos in Vimeo90K dataset [63] as our training data, which are also adopted by the recent video super resolution methods [21, 22, 57] and learnable video compression methods [32, 18, 31]. For video rescaling task, the evaluation datasets are the test set of Vimeo90K (denoted by Vimeo90K-T), the widely-used Vid4 benchmark [30] and SPMCs-30 dataset [48]. For video compression task, the evaluation datasets include UVG [1], MCL-JCV [55], and HEVC Class B [46], which contain high-quality videos with resolution 1920×1080. For video recognition task, we train and evaluate it on two large scale datasets requiring temporal relation reasoning, \( i.e., \) Something V1&V2 [12][36].

#### 4.2. Implementation Details

**Video rescaling.** \( \lambda_1, \lambda_2, \lambda_3 \) and \( \lambda_4 \) are set as 0.01, 1 and 1, respectively. \( \lambda_1 \) is decayed to 0 in the last 200K iterations. Each training clip consists of 7 consecutive RGB patches of size 224×224. The batch size is set as 8. We augment the training data with random horizontal flips and 90° rotations. We train our model with Adam optimizer [25] by setting \( \beta_1 \) as 0.9, \( \beta_2 \) as 0.999, and learning rate as \( 10^{-4} \). The total training iteration number is about 400K. The learning rate is divided by 2 every 100K iterations. We implement the models with the PyTorch framework [37] and train them with 2 NVIDIA 2080Ti GPUs. We draw 5 times from the generated distribution for each evaluation and report the averaged performance. We leverage the invertible neural network (INN) architecture to implement the CNN parts of the paired frequency analyzer and synthesizer for a fair comparison with IRN because INN will cut the number of parameters by 50%. We propose the two following models: the SelfC-small and SelfC-large, which consist of 2 and 8 invertible Dense2D-T blocks. The invertible architecture follows [60]. Training the SelfC-large model takes ~6days.

**Video compression.** The rescaling ratio of the SelfC is set to 2. We adopt H.265 as the embedded codec. \( \lambda_4 \) is set as 0.1 to make sure the statistical distribution of the downscalled videos is more closed to the natural images, which stabilizes the performance of the whole system. \( \lambda_\text{codec} \) is empirically set as 4. The CRF value of the H.265 codec is randomly selected from \{11,13,17,21\} during training while it is set to a fixed value during evaluation. The input video clip length is set as 3. The other details follow that of video rescaling task. The models are initialized from SelfC-large model but the number of invertible Dense2D-T blocks is reduced to 4. The surrogate CNN is randomly initialized, and is jointly optimized with other components of SelfC online. It takes about 5days to train the model.

**Action recognition.** We insert the downscaaler of our framework before the action recognition CNN, \( i.e., \) TSM [28]. The data augmentation pipeline also follows it. The downsampling ratio of SelfC is 2. At inference time, we used just 1 clip per video and each clip contains 8 frames. We adopt 2 plain Dense2D-T blocks as the CNN part of the frequency analyzer. Note that the downscaaler is first pretrained on Vimeo90K dataset with the rescaling task.

#### 4.3. Results of Video Rescaling

As shown in Tab. 1 and Tab. 2, our method outperforms the recent state-of-the-art video super resolution methods on Vid4, SPMCs-30 and Vimeo90K-T by a large margin in terms of both PSNR and SSIM. For example, the result on the Vid4 dataset for our SelfC-large model is 32.85dB, while the state-of-the-art video super resolution approach RSDN only achieves 27.92dB. Furthermore, we also provide the results of the image rescaling method, \( i.e., \) IRN, in
Table 1: Quantitative comparison (PSNR(Y) and SSIM-Y) on SPMCs-30 and Vimeo90K-T for downscaling. The average PSNR(Y) and SSIM-Y are calculated by averaging the respective measures across 8 downscaling modes. The first mode is for the offline downsampling of the video rescaling paradigm. The temporal relationship between downscaling modes is designed to be the same as H.265. We also show the qualitative comparison with other methods on Vid4 in Fig. 3. Our method demonstrates much better details than both video super-resolution methods and image rescaling methods, proving the superiority of the video rescaling paradigm.

4.4. Results of Video Compression

In our framework, we adopt the H.265 codec in default or zerolatency modes. The first mode is for the offline storage of video data while the other is oriented to the online low-latency video streaming services. The evaluation metrics are PSNR and MS-SSIM.

Tab. 1 and Tab. 2. It is obvious that our model outperforms IRN significantly while also reduces the computational complexity by ~3x (SelfC-large) or 6x (SelfC-small). This result clearly proves that it is necessary to exploit the temporal relationship for video rescaling task, while the existing image rescaling methods like IRN ignored this temporal cue. We also show the qualitative comparison with other methods on Vid4 in Fig. 3. Our method demonstrates much better details than both video super-resolution methods and image rescaling methods, proving the superiority of the video rescaling paradigm.

---

1 ffmpeg -i input.yuv -vf scale=200:400 -c:v libx265 -preset veryfast -x265-params "aq=Q"

2 ffmpeg -i input.yuv -vf scale=200:400 -c:v libx265 -preset veryfast -tune zerolatency -x265-params "aq=Q"

---

Table 2: BDBR results using H.265 as the anchor. The low-complexity or low-latency mode of H.265 codec. We also evaluate the Bjntegaard Delta Bit-Rate (BDBR) [4] by using H.265 as the anchor method. As shown in Tab. 3, our method saves the bit cost or the storage space by over 30% averagely under the same MS-SSIM, compared with the vanilla H.265 codec. Notably, we reduce the bit cost by 8.41%.

Table 3: BDBR results using H.265 as the anchor. The lower value, the more bit cost reduced.

---

Fig. 4 shows the experimental results. It is obvious that our method outperforms both the traditional methods and learning-based methods (DVC [32], Yang et al. [66], Hu et al. [18], and Lu et al. [31]) on video compression task by a large margin. Although our method is only optimized by $\ell_1$ loss, it demonstrates strong performances in terms of both PSNR and MS-SSIM metrics. It should also be mentioned that our models generalize well to default mode although only trained with the zerolatency mode of H.265 codec. We also evaluate the Bjntegaard Delta Bit-Rate (BDBR) [4] by using H.265 as the anchor method. As shown in Tab. 3, our method saves the bit cost or the storage space by over 30% averagely under the same MS-SSIM, compared with the vanilla H.265 codec.

---

Table 3: BDBR results using H.265 as the anchor. The lower value, the more bit cost reduced.
over 45% on the UVG dataset. This proves that the video rescaling technique is a novel and effective way to improve video compression performance, without considering much about the complicated details of the industrial lossy codecs.

We perform more analysis to verify the effectiveness of the “Video rescaling+Codec” paradigm and the proposed gradient estimation method. As shown in Fig. 5, it is observed that using Bicubic as the downsampler and upscaler in the video compression system (i.e., H.265+Bicubic) leads to a much inferior result than the baseline. We also try to improve the result by using a state-of-the-art video super resolution method, i.e., TGA [22]. The performance is indeed improved though still lower than the baseline method H.265. Considering the network parameters of TGA are 5.87M while ours are only 0.88M, this result further demonstrates the effectiveness of our SelfC framework. Finally, we provide experimental results (i.e., SelfC w/o gradient) when directly using the biased Straight-Through Estimator [3] for approximating the gradient of H.265. The results show that the proposed gradient estimation method in Section 3.6 can bring nearly 0.3dB improvements.

Finally, We give the complexity analysis of the proposed video compression system. Although it seems like that our method adds extra computation costs upon the H.265 codec, our system is indeed more efficient because the input videos to codec are downsampled. Concretely, under the zerolatency mode, for one frame with resolution 1920×1080, the average encoding time of our method is 108ms, including 21ms for 2×downscaling and 87ms consumed by the embedded H.265 codec. Our system improves the efficiency of the vanilla H.265 codec (116ms) and is also ∼5× faster than the learnable codec DVC (522ms).

4.5. Results of Efficient Video Action Recognition

We show the video action recognition results in Tab. 4. In the first group, when directly testing the action model pretrained on full resolution videos, we observe the performances of low-resolution videos downsampled by Bicubic and ours are both dropped drastically, because the classification networks are rather sensitive to the absolute scale of the input. However, our downscaler still performs better (30.4% vs. 32.7% on Something V1 dataset).

In the second group, we provide the experimental results when the action recognition CNN is fine-tuned on the low-resolution videos downsampled by Bicubic and our downscaler. It is obvious that our method clearly outperforms Bicubic downsample by about 1.5% in terms of Top1 accuracy on the two datasets. Notably, our downscaler is learnable. Therefore, we then jointly fine-tune action recognition CNNs and our downscaler. The results in the last group show that the end-to-end joint training strategy further improves the performance by an obvious margin. On Something V2, the ultimate performances of our method nearly achieve that of performing recognition directly on HR videos, and our method improves the efficiency by over 3 times. The downscaler of IRN can not improve the efficiency of this task because its computation cost is even larger than the HR setting. We try to decrease the layer number of IRN but it no longer converges.

| Method | Action CNN | FLOPs | Params | V1  | V2  |
|--------|------------|-------|--------|-----|-----|
|        |            |       |        | Top1 (%) | Top5 (%) | Top1 (%) | Top5 (%) |
| HR     | TSM        | 33.3G | 24.31M | 45.6 | 74.2 | 58.8 | 85.4 |
| Bicubic| TSM        | 8.6G  | 24.31M | 30.4 | 57.5 | 40.1 | 68.7 |
| Ours   | TSM        | 10.8G | 24.35M | 32.7 | 59.5 | 41.8 | 71.5 |
| Ours (FT) | TSM    | 8.6G  | 24.35M | 42.1 | 72.1 | 56.0 | 83.9 |
| Ours (E2E) | TSM     | 10.8G | 24.35M | 43.5 | 72.8 | 57.4 | 84.8 |
|        | Ours (E2E) |       |        | 44.6 | 73.3 | 58.3 | 85.5 |

Table 4: Comparison between our method and Bicubic downscaler. FT denotes only fine-tuning the action recognition CNN. E2E denotes also fine-tuning the downscaler.

4.6. Ablation Studies on the Framework

In this section, we conduct experiments on video rescaling task to verify the effectiveness of the components in
Figure 6: Visualization of the high-frequency component $\hat{f}_h$ sampled from the learned distribution $p(f_h|x_l)$. We also compare the difference of the downsampled video by our learnable downscaler and the Bicubic downsampler. Since the magnitude of the difference is small, we amplify it by 10 times for better visualization.

Table 5: Ablation studies on $4 \times$ video rescaling. All blocks adopt the INN architecture for a fair comparison with IRN. * indicates the model is re-trained on Vimeo90K.

| Methods | Backbone | Probability model | Param(M) | Vidi4-Y | PSNR(dB) | SSIM |
|---------|----------|--------------------|----------|---------|----------|------|
| Auto-Enc | 16×Dense2D-T | - | 3.63 | 28.91 | 0.8797 |
| IRN* | 16×Dense2D | Normal | 4.36 | 30.68 | 0.9067 |
| IRN-T | 16×Dense2D-T | Normal | 3.63 | 30.42 | 0.9004 |
| SelfC-basic | 2×Dense2D | GMM(K=1) | 1.77 | 30.62 | 0.9214 |
| SelfC-basicT | 2×Dense2D-T | GMM(K=1) | 1.61 | 31.29 | 0.9268 |
| SelfC-small | 2×Dense2D-T | GMM(K=5) | 1.76 | 31.61 | 0.9317 |
| SelfC-large | 8×Dense2D-T | GMM(K=5) | 3.37 | 32.30 | 0.9402 |

4.7. Visualization Results

While the previous quantitative results validate the superiority of the proposed self-conditioned modeling scheme on several tasks, it is interesting to investigate the intermediate components output by our model, especially the distribution of the high-frequency (HF) component predicted by STP-Net. Note that the distribution is a mixture of Gaussian and includes multiple channels, we draw two samples of $\hat{f}_h$ from $p(f_h|x_l)$ and randomly select 1 channel of them for visualization. The $\hat{f}_h$ output from the frequency analyzer is adopted as the ground-truth sample.

As shown in Fig. 6, we first see that the LR video $x_l$ downscaled by our method is modulated into some mandatory information for reconstructing the HF components more easily, compared to Bicubic. Also, the sampled HF components can restore the ground-truth of that accurately in terms of key structures, i.e., the windows of the building, while retaining a certain degree of randomness. This is consistent with our learning objectives.

5. Conclusion

We have proposed a video-rescaling framework to learn a pair of downscaling and upscaling operations. Extensive experiments demonstrated that our method can outperform the previous methods with a large margin while with much fewer parameters and computational costs. Moreover, the learned operators facilitates the tasks of video compression and efficient action recognition significantly.

Acknowledgement This work was supported by the National Science Foundation of China (61831015, 61527804 and U1908210).
References

[1] Ultra video group test sequences. http://ultravideo.cs.tut.fi, accessed: 2019-11-06.

[2] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In CVPR, 2020.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv, 2013.

[4] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. VCEG-M33, 2001.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In CVPR, 2017.

[6] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In ICCV, 2019.

[7] Xin Fan, Xingju Lu, Dian Li, and Yuanliu Liu. Video-based emotion recognition using cnn-rnns and 3d hybrid networks. In ACM MI, 2016.

[8] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In ICCV, 2019.

[9] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In CVPR, 2016.

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

[11] Peter W Glynn and Roberto Szechman. Some new perspectives on the method of control variates. In Monte Carlo and Quasi-Monte Carlo Methods 2000. 2002.

[12] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yiannilos, Moritz Mueller-Freitag, et al. The\'s something something\' video database for learning and evaluating visual common sense. In ICCV, 2017.

[13] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. arXiv, 2017.

[14] Alex Graves. Stochastic backpropagation through mixture density distributions. arXiv, 2016.

[15] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In ICCV, 2019.

[16] Kengo Haru, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In ICCVW, 2017.

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

[18] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. Improving deep video compression by resolution-adaptive flow coding. In ECCV, 2020.

[19] Zhihao Hu, Guo Lu, and Dong Xu. Fvc: A new framework towards deep video compression in feature space. In CVPR, 2021.

[20] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efﬁcient convnet descriptor pyramids. In arXiv, 2014.

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

[22] 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.

[23] 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.

[24] Heewon Kim, Myungsung Choi, Bee Lim, and Kyoung Mu Lee. Task-aware image downscaling. In ECCV, 2018.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In arXiv, 2014.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv, 2013.

[27] Yue Li, Dong Liu, Houqiang Li, Li Li, Zhu Li, and Feng Wu. Learning a convolutional neural network for image compact-resolution. In TIP, 2018.

[28] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efﬁcient video understanding. In ICCV, 2019.

[29] Jianping Lin, Dong Li, Houqiang Li, and Feng Wu. M-ivc: multiple frames prediction for learned video compression. In CVPR, 2020.

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

[31] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In ECCV, 2020.

[32] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In CVPR, 2019.

[33] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Zhiyong Gao, and Ming-Ting Sun. Deep kalman filtering network for video compression artifact reduction. In ECCV, 2018.

[34] Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Li Chen, Zhiyong Gao, and Dong Xu. An end-to-end learning framework for video compression. TPAMI, 2020.

[35] Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Dong Xu, Li Chen, and Zhiyong Gao. Deep non-local kalman network for video compression artifact reduction. TIP, 2019.

[36] Farzaneh Mahdisoltani, Guillaume Berger, Waseem Gharbieh, David Fleet, and Roland Memisevic. Fine-grained video classiﬁcation and captioning. In arXiv, 2018.
