Tianjin University

NEURAL INFORMATION PROCESSING SYSTEMS

QBasicVSR: Temporal Awareness Adaptation Quantization for Video Super-Resolution

Zhenwei Zhang[1], Fanhua Shang[1*], Hongying Liu[2*], Liang Wan[2,3], Wei Feng[1], Yanming Hui[1]

[1]School of Computer Science and Technology, Tianjin University [2]Medical School, Tianjin University
[3]School of Computer Software, Tianjin University

Background

Quantization is essential for mobile deployment, but previous works only focused on image super-resolution (SR). Video SR quantization, however, faces temporal error propagation, shared temporal parameterization, and temporal metric mismatch challenges. There is no quantization method for video super-resolution. To address these issues, we introduce QBasicVSR, the first quantization method specifically designed for VSR.

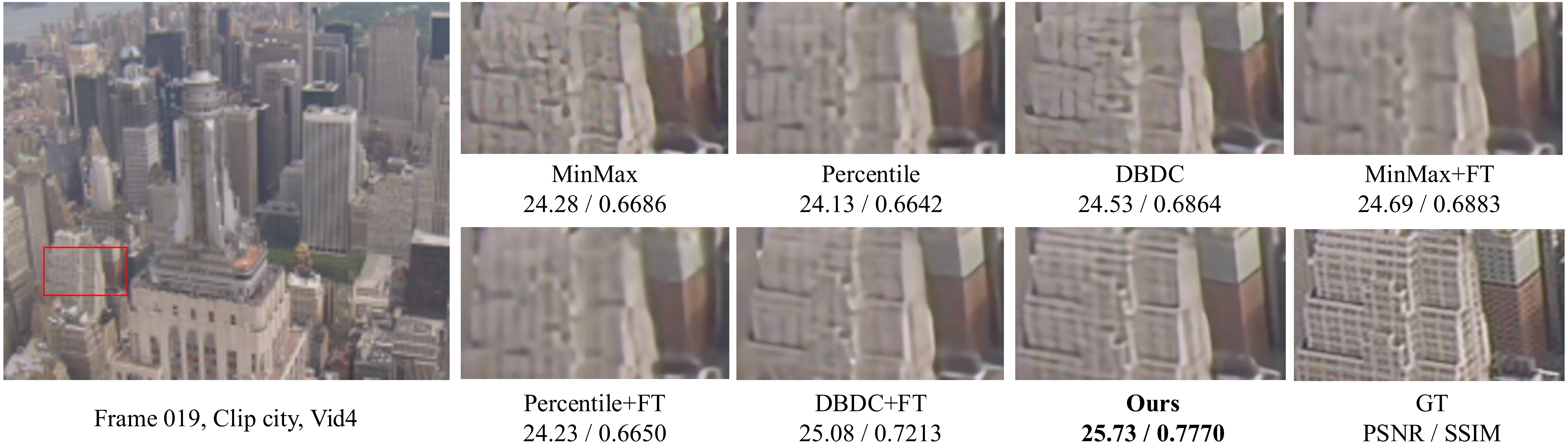


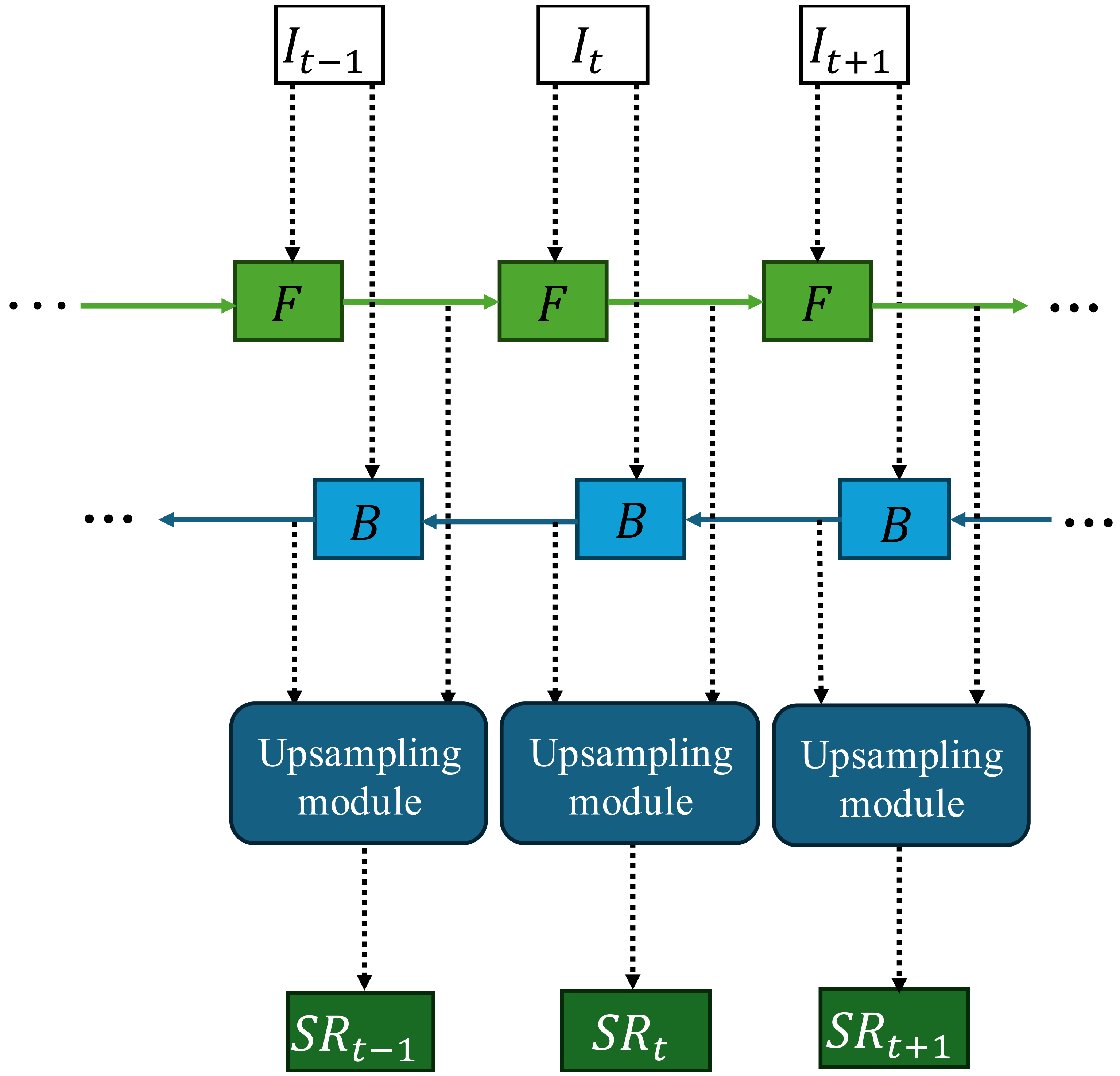
Fig. 1: Qualitative comparison on Vid4. Only our method reconstructs both structural contours and fine-grained details.

Contributions

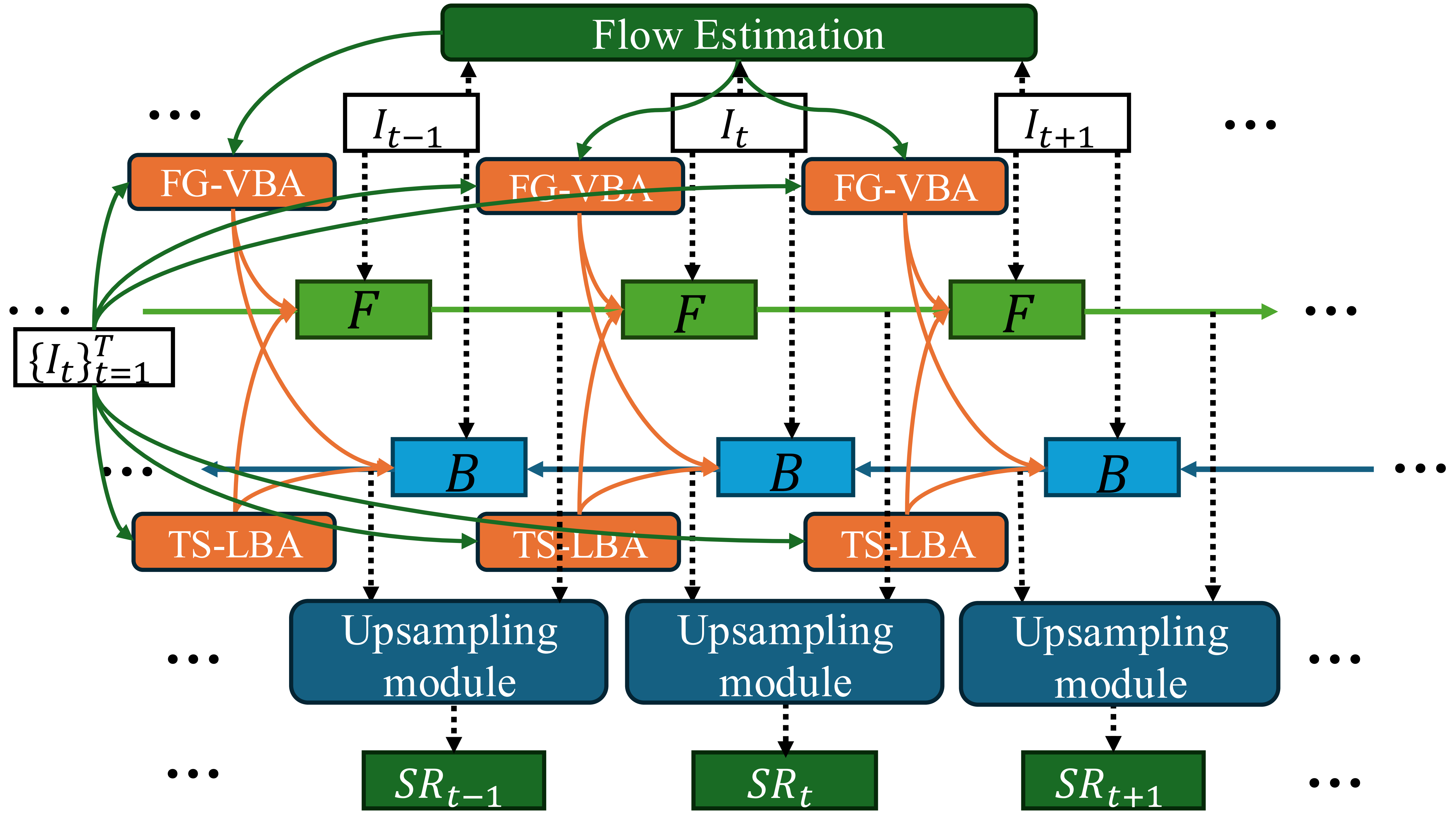
- To the best of our knowledge, this is the first work that explicitly addresses quantization for video super-resolution.
- We propose a novel dual-optimization framework (QBasicVSR) with flow-gradient and temporal-shared bit adaptation modules.
- A new proposed fine-tuning scheme further enhances performance under full-precision supervision, achieving 200 times faster processing speed with only 1/8 GPU usage.
- We also release a novel quantization VSR library.

Methodology

Overview



(a) Bidirectional VSR (before quantization)



(b) The proposed temporal awareness adaptation quantization for VSR

Fig. 2: Overview of the proposed temporal awareness adaptation quantization for VSR. Parameters are shared within green and blue blocks separately. (a) The frames I_{t-1}, I_t, and I_{t+1} represent consecutive LR video frames. F and B denote the forward and backward modules. (b) During inference, the TS-LBA module is fixed, and the FG-VBA module is adapted according to the video complexity.

Methodology

Overview

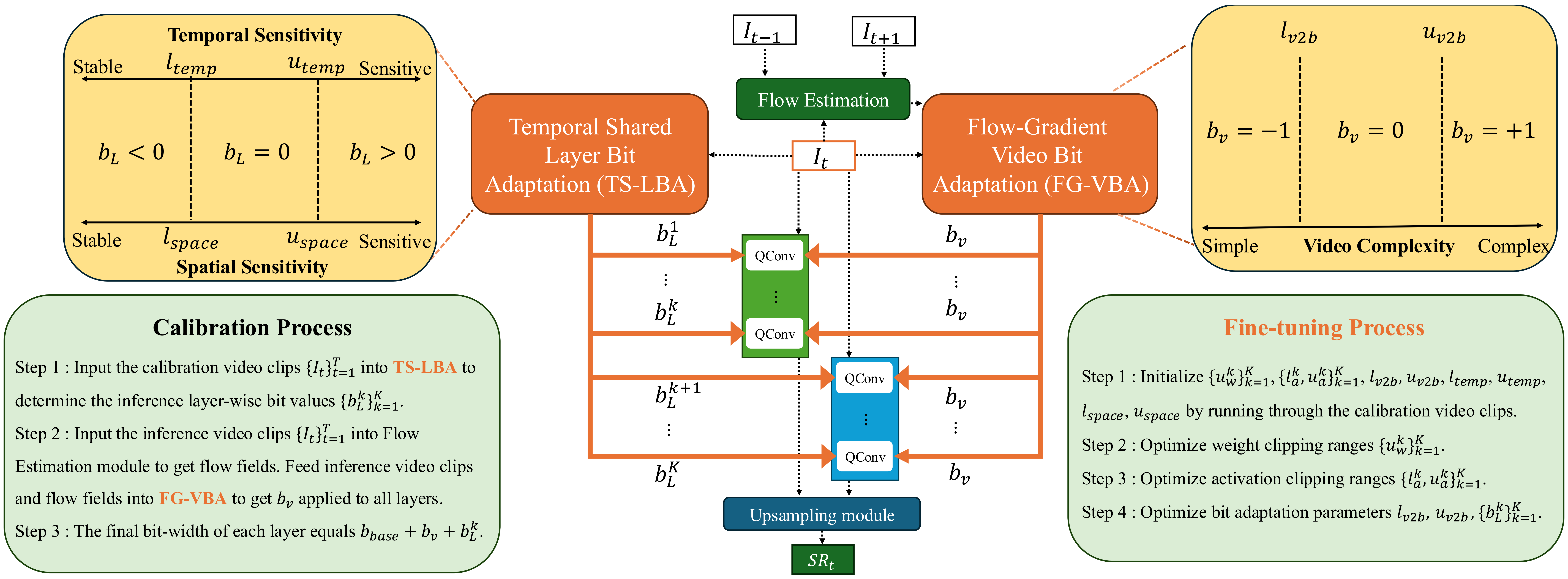


Fig. 3: The proposed TS-LBA and FG-VBA modules dynamically modulate the bit-widths of layers.

Methodology

Preliminaries

- We briefly introduce the quantization approach, focusing on asymmetric quantization for activations, as is standard practice for SR networks due to their asymmetric activation distributions. The quantization process is formulated as:

$$x_c = \min(\max(x, l), u), \quad x_{int} = \left\lfloor \frac{x_c - l}{S} \right\rceil, \quad S = \frac{u - l}{2^b - 1}, \quad x_q = x_{int} \cdot S + l\,.$$

where l and u denote the lower and upper clipping thresholds, b is the quantization bit-width, and S represents the scaling factor that maps the clamped tensor x_c (confined to $[l, u]$) to 2^b discrete levels. The operator $\lfloor \cdot \rceil$ implements nearest-integer rounding, while x_{int} corresponds to the hardware-friendly integer representation. For weights, we enforce the symmetry quantizer by setting $l = -u$.

Methodology

Flow-Gradient Video Bit Adaptation

- The spatial complexity S_t for the frame t is then calculated as the mean ℓ_1-norm of gradients across all pixels:

$$S_t = \frac{10^3}{|\Omega|} \sum_{(i,j)\in\Omega} \left\| \nabla I_t(i,j) \right\|_1, \quad \nabla I_t(i,j) = \left[\frac{\partial I_t}{\partial x}(i,j), \frac{\partial I_t}{\partial y}(i,j)\right]^\top, \quad |\Omega| = H \times W.$$

- Let $\mathcal{F}_t = \left\{\mathbf{F}_f, \mathbf{F}_b\right\}$ denote the set of forward $(t \to t+1)$ and backward $(t+1 \to t)$ flow fields estimated by SPyNet [1]. Each flow field $\mathbf{F} \in \mathcal{F}_t$ is a two-channel map, where the vector at pixel (i,j), $\mathbf{F}(i,j) = (u(i,j), v(i,j))^\top \in \mathbb{R}^2$, represents the horizontal and vertical displacements of that pixel. The flow magnitude term quantifies average motion intensity across bidirectional flows:

$$T_{mag,t} = \frac{1}{|\mathcal{F}_t| \cdot |\Omega|} \sum_{\mathbf{F}\in\mathcal{F}_t} \sum_{(i,j)\in\Omega} \left\| \mathbf{F}(i,j) \right\|_2.$$

[1] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4161–4170, 2017.

Methodology

Flow-Gradient Video Bit Adaptation

- For each flow field $\mathbf{F} \in \mathscr{F}_t$, we compute the Jacobian matrix $J_{\mathbf{F}}(i,j) \in \mathbb{R}^{2\times 2}$ at a pixel (i,j).

$$T_{cons,t} = \frac{1}{|\mathscr{F}_t| \cdot |\Omega|} \sum_{\mathbf{F}\in\mathscr{F}_t} \sum_{(i,j)\in\Omega} \left\| J_{\mathbf{F}}(i,j) \right\|_{1,1}.$$

- Then, the temporal complexity T_t combines the magnitude and consistency terms through a weighted summation:

$$T_t = T_{mag,t} + \gamma \cdot T_{cons,t}.$$

- The overall flow-gradient complexity metric is defined as:

$$C_V = \frac{1}{T} \sum_{t=1}^{T} S_t + \lambda \cdot \frac{1}{T-1} \sum_{t=1}^{T-1} \left(T_{mag,t} + \gamma \cdot T_{cons,t} \right).$$

- The video-to-bit allocation dynamically adjusts global bit-width by mapping video complexity C_V to $b_V \in \{-1, 0, +1\}$.

Methodology

Temporal Shared Layer Bit Adaptation

- For each convolutional layer k, given the input video sequences with T frames, we stack activations into $\mathcal{X}^k_{stacked} \in \mathbb{R}^{T\times B\times C\times H\times W}$. The spatial sensitivity s^k_{space} quantifies intra-frame activation variability:

$$s^k_{space} = \mathbb{E}_{t,b,h,w}\left[\sigma_c\left(\mathcal{X}^k_{stacked,t,b,:,h,w}\right)\right].$$

- The temporal sensitivity s^k_{temp} models inter-frame dynamics as follows:

$$s^k_{temp} = \mathbb{E}_{b,c,h,w}\left[\sigma_t\left(\mathcal{X}^k_{\text{stacked},:,b,c,h,w}\right)\right].$$

- The layer-wise bit adaptation factor b^k_L is determined by joint thresholding:

$$b^k_L = \begin{cases} -1, & if(s^k_{space}, s^k_{temp}) \in [0,l_{space}] \times [0,l_{temp}] \\ +1, & if(s^k_{space}, s^k_{temp}) \in [u_{space}, \infty) \times [u_{temp}, \infty) \\ 0, & otherwise \end{cases}.$$

Methodology
Calibration

- The b_L^k of each layer is pre-determined and fixed during test time. Finally, the adapted bit-width is expressed as follows:

$$b_V^k = b_{base} + b_V + b_L^k .$$

where b_V^k represents the bit-width for video stream V in layer k, b_{base} is the baseline bit-width serving as the starting point, b_V adjusts for the complexity of the video, and b_L^k adapts to the quantization sensitivity of each layer.

Methodology

Fine-tuning

- Inspired by the Charbonnier loss in VSR, we design the reconstruction loss as follows:

$$\mathscr{L}_{pix} = \frac{1}{T}\sum_{i=1}^{T}\sqrt{\left\| P(I_i^{LR}) - Q(I_i^{LR}) \right\|_2^2 + \epsilon^2}\,.$$

where $P(\cdot)$ and $Q(\cdot)$ correspond to the full-precision (FP) and quantized networks, respectively.

- For feature-level supervision, the pixel transfer loss to video domains is formulated as follows:

$$\mathscr{L}_{skt} = \frac{1}{T \cdot K}\sum_{i=1}^{T}\sum_{k=1}^{K}\left\| \frac{F_P^{i,k}}{\left\| F_P^{i,k} \right\|_2} - \frac{F_Q^{i,k}}{\left\| F_Q^{i,k} \right\|_2} \right\|_2^2 .$$

where T denotes the number of temporal frames and K represents the total number of feature layers.

- Finally, we can get the total loss:

$$\mathscr{L}_{total} = \mathscr{L}_{pix} + \lambda_{skt}\mathscr{L}_{skt}\,.$$

Experiments

Table 1: Quantitative comparison (PSNR / SSIM). All results are calculated on the Y-channel except REDS4 (RGB-channel). The results for PQBasicVSR (Ours) and FQBasicVSR (Ours) correspond to Partial Quantized and Fully Quantized models, respectively.

Methods	Params(M)	GT	W / A	BI degradation			BD degradation		
				REDS4 [33]	Vimeo-90K-T [45]	Vid4 [30]	UDM10 [46]	Vimeo90K-T [45]	Vid4 [30]
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 [2]	-	✓	32 / 32	-	-	25.35 / 0.7557	-	-	-
SPMC [36]	-	✓	32 / 32	-	-	25.88 / 0.7752	-	-	-
TOFlow [45]	1.4	✓	32 / 32	27.98 / 0.7990	33.08 / 0.9054	25.89 / 0.7651	36.26 / 0.9438	34.62 / 0.9212	-
DUF [22]	5.8	✓	32 / 32	28.63 / 0.8251	-	-	38.48 / 0.9605	36.87 / 0.9447	27.38 / 0.8329
RBPN [13]	12.2	✓	32 / 32	30.09 / 0.8590	37.07 / 0.9435	27.12 / 0.8180	38.66 / 0.9596	37.20 / 0.9458	-
EDVR-M [40]	3.3	✓	32 / 32	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 [47]	3.0	✓	32 / 32	29.63 / 0.8502	36.14 / 0.9363	26.73 / 0.8029	38.74 / 0.9627	-	27.16 / 0.8355
TGA [18]	5.8	✓	32 / 32	-	-	-	-	37.59/ 0.9516	27.63 / 0.8423
RLSP [10]	4.2	✓	32 / 32	-	-	-	38.48 / 0.9606	36.49 / 0.9403	27.48 / 0.8388
RSDN [17]	6.2	✓	32 / 32	-	-	-	39.35 / 0.9653	37.23 / 0.9471	27.92 / 0.8505
RRN [19]	3.4	✓	32 / 32	-	-	-	38.96 / 0.9644	-	27.69 / 0.8488
FastDVDnet [44]	2.6	✓	32 / 32	-	36.12 / 0.9348	26.14 / 0.8029	-	-	-
BasicVSR [3]	4.9	✓	32 / 32	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 [43]	1.3	✓	32 / 32	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 [25]	1.3	✓	32 / 32	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 [48]	1.3	✓	32 / 32	30.74 / 0.8770	36.75 / 0.9414	27.01 / 0.8176	39.15 / 0.9660	36.93 / 0.9450	27.40 / 0.8400
KSNet [21]	1.6	✓	32 / 32	31.14 / 0.8862	-	27.22 / 0.8245	-	37.54 / 0.9503	-
SSL [43]	1.0	✓	32 / 32	31.06 / 0.8833	36.82 / 0.9419	27.15 / 0.8208	39.35 / 0.9665	37.06 / 0.9458	27.56 / 0.8431
PQBasicVSR (Ours)	**1.3**	✗	**6 / 6MP**	**31.26 / 0.8879**	**37.07 / 0.9440**	**27.18 / 0.8215**	**39.64 / 0.9680**	**37.37 / 0.9485**	**27.84 / 0.8495**
FQBasicVSR (Ours)	**1.0**	✗	**6 / 6MP**	**31.17 / 0.8849**	**36.79 / 0.9409**	**27.05 / 0.8140**	**39.22 / 0.9646**	**37.02 / 0.9444**	**27.69 / 0.8405**
PQBasicVSR (Ours)	**1.0**	✗	**4 / 4MP**	**30.34 / 0.8657**	**35.93 / 0.9315**	**26.26 / 0.7764**	**38.15 / 0.9576**	**36.46 / 0.9387**	**27.02 / 0.8161**
FQBasicVSR (Ours)	**0.7**	✗	**4 / 4MP**	**30.26 / 0.8637**	**35.82 / 0.9311**	**26.29 / 0.7752**	**37.59 / 0.9536**	**35.95 / 0.9339**	**26.81 / 0.8025**

Experiments

Table 2: Comparison with SOTA efficient VSR methods. Pretraining denotes whether a pretrained model has been used, Data means the number of video clips required for training, GT denotes the requirement for ground-truth HR videos, BS is the batch size during the fine-tuning phase, and GPUs denotes the number of GPUs used. The processing time is measured on A6000 GPUs. The runtime is computed based on an LR size of 180 * 320.

Methods	Pretraining	Data	GT	BS	Iterations	Processing Time	Runtime	GPUs
BasicVSR	-	26600	✓	8	300,000	116hrs	53ms	2
KSNet [21]	✗	26600	✓	8	600,000	96hrs	-	4
SSL [43]	✓	26600	✓	8	303,380	370hrs	54ms	8
Ours	✓	**100**	✗	**2**	**150**	**90min**	**8ms**	**1**

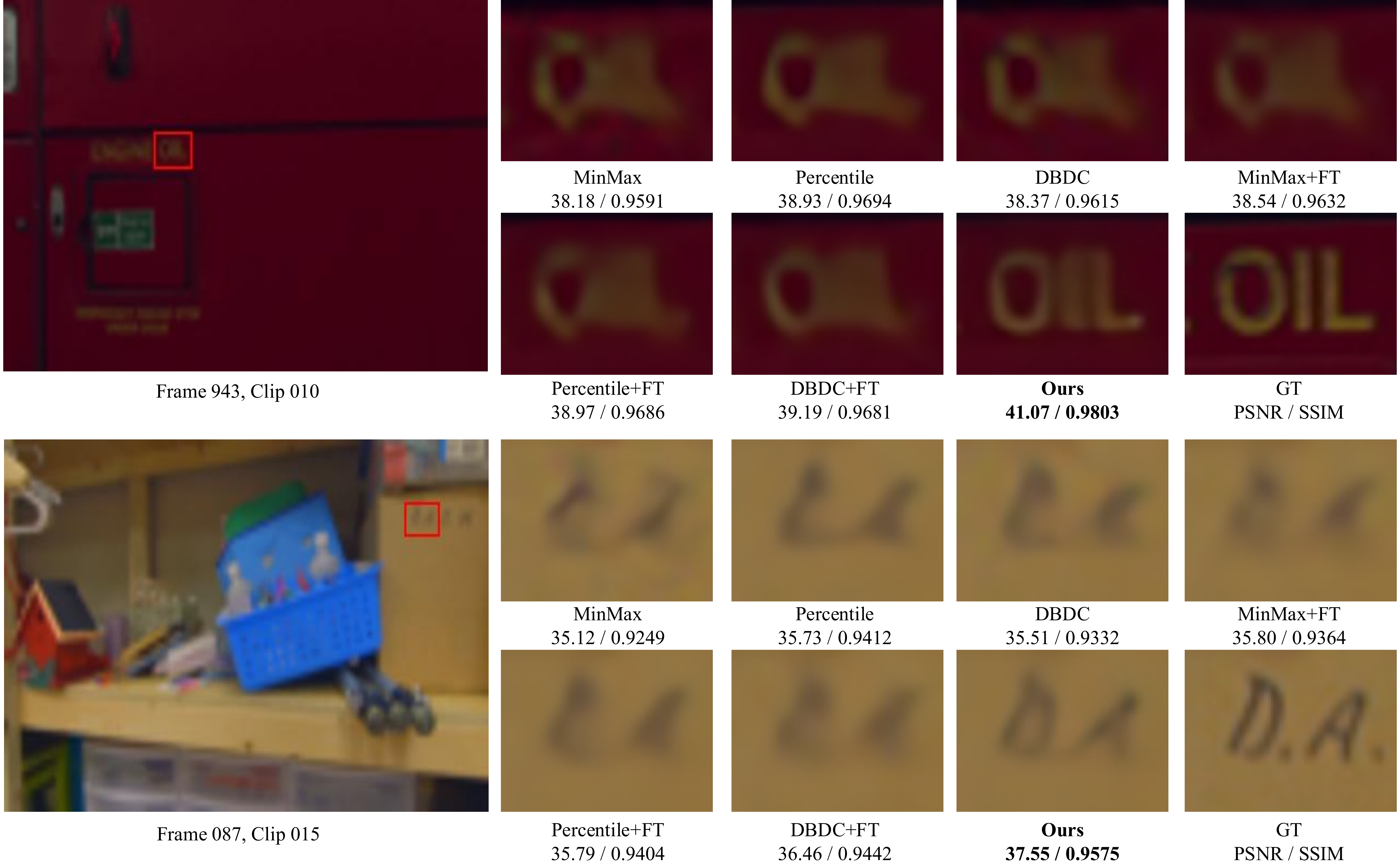


Fig. 4: Qualitative comparison on Vimeo-90K.

Thank you