QBasicVSR: Temporal Awareness Adaptation Quantization for Video Super-Resolution

Zhenwei Zhang[1], **Fanhua Shang**[1]*, **Hongying Liu**[2, 3]*, **Liang Wan**[2, 4],
Wei Feng[1], **Yanming Hui**[1]

[1]School of Computer Science and Technology, Tianjin University [2]Medical School, Tianjin University
[3]Peng Cheng Lab, Shenzhen [4]School of Computer Software, Tianjin University

{microsoftzzw, fhshang, hyliu2009, lwan, wfeng, ymhui}@tju.edu.cn

Abstract

While model quantization has become pivotal for deploying super-resolution (SR) networks on mobile devices, existing works focus on quantization methods only for image super-resolution. Different from image SR quantization, the temporal error propagation, shared temporal parameterization, and temporal metric mismatch significantly degrade the quantization performance of a video SR model. To address these issues, we propose the first quantization method, QBasicVSR, for video super-resolution. A novel temporal awareness adaptation post-training quantization (PTQ) framework for video super-resolution with the flow-gradient video bit adaptation and temporal shared layer bit adaptation is presented. Moreover, we put forward a novel fine-tuning method for VSR with the supervision of the full-precision model. Our method achieves extraordinary performance with state-of-the-art efficient VSR approaches, delivering up to $\times$200 faster processing speed while utilizing only 1/8 of the GPU resources. Additionally, extensive experiments demonstrate that the proposed method significantly outperforms existing PTQ algorithms on various datasets. For instance, it attains a 2.53 dB increase on the UDM10 benchmark when quantizing BasicVSR to 4-bit with 100 unlabeled video clips. The code and models will be released on GitHub.

1 Introduction

Super-resolution (SR) is a classic low-level computer vision task focused on reconstructing high-resolution image (s) from their low-resolution (LR) counterparts. In video scenarios, it extends to aggregating spatiotemporal information from multiple misaligned frames within a sequence, requiring motion compensation and frame alignment to achieve coherent detail enhancement across the temporal domain. Despite demonstrating impressive performance, state-of-the-art video super-resolution (VSR) networks [3, 4, 28, 29] face deployment challenges on resource-constrained edge devices due to their high computational and memory demands. Furthermore, the growing industry demand for real-time large-scale VSR (e.g., 4K video streaming) introduces quadratically increasing computational complexity. To enhance computational efficiency in VSR, researchers have explored various efficient methods, such as reparameterization [21] and network pruning [25, 42, 46], which aim to reduce model size or simplify network structures. In parallel, model quantization has proven to be another powerful efficiency-driven technique [14, 26, 37, 38, 47], by leveraging low-precision arithmetic to reduce latency, memory footprint, and consumption cost on AI accelerators [6, 41]. However, despite its advantages, quantization has not yet been explored in the domain of VSR.

*Corresponding authors

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Although existing efficient VSR methods have achieved notable progress in reducing parameter sizes and improving inference speed, their critical limitation stems from reliance on 32-bit floating-point (FP) operations. This high computational precision inherently restricts their ability to fully exploit the capabilities of modern AI accelerators, which are optimized for low bit-width computations. Furthermore, these efficient methods in VSR necessitate processing entire datasets during training, resulting in prohibitively expensive computational overhead that persists despite their architectural optimizations. As shown in Table 4, the SOTA efficient BasicVSR method [42] requires training 303,380 iterations with a batch size of 8, which takes approximately 15 days on 8 NVIDIA A6000 GPUs. Quantization has emerged as a critical, efficient strategy, systematically replacing 32-bit FP representations in network parameters and activation tensors with reduced-precision arithmetic, thereby achieving up to $4\times$ memory footprint compression and $2-3\times$ latency reduction in hardware-accelerated inference engines.

Current mainstream quantization methods can be broadly categorized into two paradigms [11]: quantization-aware training (QAT) approaches that require complete training datasets for end-to-end model retraining, and post-training quantization (PTQ) methods that directly leverage pre-trained models with only a few unlabeled calibration samples. While image SR quantization methods [14, 16, 26, 37, 38, 47] have achieved advanced performance with minimal degradation in accuracy, there are seldom works on quantization for VSR. When transitioning from image to VSR, three fundamental challenges emerge [3, 4, 28, 29] : (1) Temporal error propagation: the recurrent architecture, which is adopted in almost all VSR models, accumulates quantization errors across inter-frame dependencies; (2) Shared temporal parameterization: weight-sharing in VSR across time steps introduces compounded quantization effects absent in single image processing; (3) Temporal metric mismatch: conventional image quality assessments fail to capture temporal consistency degradation. Therefore, existing quantization methods developed for image SR can not be directly transferred to VSR tasks. Crucially, even when attempting to adapt QAT to VSR, significant barriers persist. While QAT could theoretically recover quantized model performance, it demands full retraining with the same iterations as original FP models. Each training step incurs exacerbated GPU memory and prolonged duration due to gradient accumulation through recurrent units and quantization-aware gradient calculations. Given that VSR training already requires orders of magnitude more computation than image SR, such additional overhead makes QAT-based retraining unrealistic for practical deployment, particularly in resource-constrained scenarios. Therefore, we concentrate on the PTQ methods in this paper.

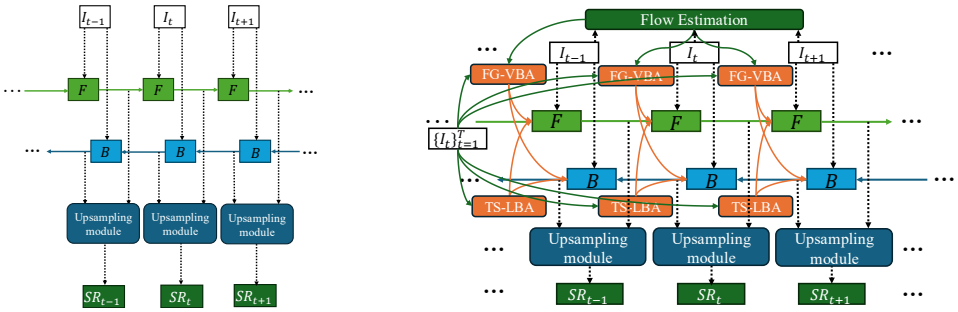
To address the issues above, we propose the first VSR quantization method for an accurate model in this paper. Our framework decouples the adaptive bit allocation mechanisms between video streams and parameter-shared layers, allowing independent optimization of bit-width adaptation to reduce the heavy computation. For video streams, we propose a flow-gradient video bit adaptation strategy driven by a spatiotemporal complexity metric. Videos exceeding the upper threshold are assigned a positive factor that increases bit-width allocation. Calibration datasets are then utilized to refine the threshold through backward propagation. Subsequently, by exploiting the spatiotemporal correlations inherent in video data, a temporal shared layer bit adaptation module is presented to evaluate the quantization sensitivity of each layer through temporal-aware analysis. This analysis captures both intra-frame features and inter-frame temporal dependencies, ultimately determining the bit factors for the weight-sharing layers. The layer-wise sensitivity is calculated by processing the calibration datasets with the pre-trained FP network. Layers exhibiting sensitivity values exceeding the upper threshold are identified as critical components and assigned positive adaptation factors. These layer-specific adaptation factors undergo further calibration through direct fine-tuning using the same calibration datasets, enabling precise adjustment of quantization parameters while preserving temporal coherence across video frames. This dual-optimization architecture, strategically combining temporal-aware bit allocation (for both video and VSR network layers) with calibration-driven sensitivity refinement, establishes optimized quantization scheme designs that holistically balance spatial-textural details, temporal evolution patterns, and layer-specific representation requirements throughout the video restoration pipeline. Compared to traditional VSR methods, which rely on complete labeled ground truth (GT), our method achieves rapid quantization with unprecedented efficiency, completing the entire process using only a small collection of unlabeled video clips. The contributions of this work are summarized as follows:

(1) To the best of our knowledge, this is the first work that explicitly addresses model quantization for video super-resolution task.

(2) We propose a novel dual-optimization framework with a flow-gradient video bit adaptation strategy guided by spatiotemporal complexity metrics, coupled with a temporal shared layer bit adaptation module that dynamically coordinates layer-wise bit-width distribution.

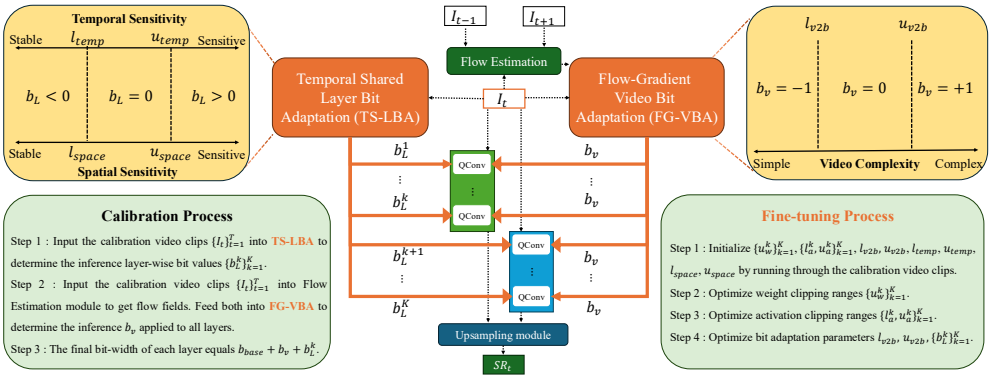
(3) Moreover, we present a novel fine-tuning method for VSR tasks with the supervision of the full-precision model. Our method outperforms existing efficient VSR approaches. Furthermore, our method outperforms existing PTQ techniques by a substantial margin.

(4) To promote collaborative studies and industrial deployments, we also provide QBasicVSR, a novel quantization VSR library. This new library includes the complete implementation, such as training protocols, quantization-aware modules, and pre-trained model weights.



(a) Bidirectional VSR

(b) The proposed temporal awareness adaptation quantization for VSR



(c) The proposed TS-LBA and FG-VBA modules

Figure 1: 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. (c) The TS-LBA and FG-VBA modules dynamically modulate the bit-widths of layers.

2 Related Work

Model quantization has emerged as a critical technique for compressing and accelerating deep neural networks, garnering significant research attention and achieving ubiquitous adoption in modern AI systems due to its effectiveness in enabling efficient deployment across resource-constrained platforms. PAMS [26] addresses dynamic activation variations by adaptively updating quantization ranges to match their statistical characteristics. Consequently, DDTB [47] successfully pushes the compression boundary to ultra-low 4-bit precision without catastrophic accuracy degradation by developing test-time quantization range adaptation through dynamic parameterization. Furthermore, recent works, CADyQ [14] and CABM [37], propose adaptive quantization frameworks employing content-aware bit-width prediction modules that dynamically adjust quantization levels based on input image characteristics. However, these methods all require QAT with a complete dataset and often incur even higher computational costs than training full-precision models, particularly problematic for videos where processing high-dimensional temporal data makes such retraining procedures prohibitively expensive in both time and computational resources.

To overcome these limitations, PTQ enables efficient model compression without retraining by calibrating quantization parameters using only a few unlabeled samples. Tu et al. [38] proposed the first PTQ method for image SR, which introduces the density-based dual clipping to cut off the outliers by analyzing the asymmetric bounds of activations. To promote adaptive inference, Cheeun et al. [16] proposed an adaptive quantization method, which achieves competitive performance with the previous adaptive quantization methods. However, these methods are specifically designed for image SR and cannot be directly applied to video models. To this end, we propose a video quantization framework that requires only a few unlabeled video clips to determine quantization parameters automatically.

3 Methodology

Our method, illustrated in Fig. 1, proposes a novel adaptive quantization method, temporal awareness adaptation quantization for VSR. The bit allocation strategy comprises two adaptive modules: a temporal shared layer bit adaptation, which dynamically assigns layer-wise bit-widths based on spatial and temporal sensitivities, and a flow-gradient video bit adaptation, which globally adjusts bit-widths guided by a unified spatiotemporal complexity metric. After the adaptive bit-width configurations are determined by the two modules and finalized during the calibration phase, we design a novel fine-tuning step. This fine-tuning further optimizes the quantization parameters, significantly enhancing reconstruction accuracy while maintaining computational efficiency. During inference, the layer-wise bit-widths remain fixed, while the bit-widths are adaptively adjusted based on the input video complexity through the flow-gradient video bit adaptation module.

3.1 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 [14, 15, 37, 47]. Given a floating-point tensor x with bit-width b, the quantization process is formulated as:

$$\begin{aligned} x_c &= \text{Clamp}(x, l, u) = \min(\max(x, l), u), \\ x_{int} &= \lfloor \frac{x_c - l}{S} \rceil, S = \frac{u - l}{2^b - 1}, \\ x_q &= x_{int} \cdot S + l, \end{aligned} \tag{1}$$

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 symmetry quantizer by setting $l = -u$.

3.2 Flow-Gradient Video Bit Adaptation

Our goal is to determine a universal flow-gradient synergized adaptation factor, which dynamically adjusts the uniform bit-width for all layers based on spatiotemporal complexity characteristics of input video streams. To achieve this, we design a joint gradient-flow complexity metric that models spatial texture richness and temporal motion dynamics. Below, we formalize the components of this metric and their mathematical definitions.

Spatial Gradient Analysis: For each frame I_t in a video stream of T frames, where $t \in \{1, \ldots, T\}$, we compute spatial complexity by aggregating gradient magnitudes across the frame. Let $\Omega = \{(i, j) \mid 0 \leq i < H, 0 \leq j < W\}$ denote the set of all pixel coordinates in the t-th frame with height H and width W. The spatial gradient $\nabla I_t(i, j)$ at pixel (i, j) is defined as a vector containing horizontal and vertical derivatives, computed via finite differences:

$$\frac{\partial I_t}{\partial x}(i, j) = I_t(i + 1, j) - I_t(i, j), \quad \frac{\partial I_t}{\partial y}(i, j) = I_t(i, j + 1) - I_t(i, j). \tag{2}$$

The spatial complexity S_t for frame t is then calculated as the mean ℓ_1-norm of gradients across all pixels, scaled by 10^3 to amplify small magnitudes:

$$S_t = \frac{10^3}{|\Omega|} \sum_{(i,j) \in \Omega} \|\nabla I_t(i, j)\|_1 , \tag{3}$$

where $\nabla I_t(i,j) = \left[\frac{\partial I_t}{\partial x}(i,j), \frac{\partial I_t}{\partial y}(i,j)\right]^\top$, $|\Omega| = H \times W$, denotes the total number of pixels, and $\|\cdot\|_1$ represents the ℓ_1-norm. This method emphasizes high-frequency spatial details (e.g., edges and textures), critical indicators of visual complexity in videos.

Temporal Flow-Field Dynamics: To holistically model video complexity, we characterize temporal motion dynamics through bidirectional flow-field analysis. Let $\mathcal{B}_t = \{\mathbf{F}_\mathrm{f}, \mathbf{F}_\mathrm{b}\}$ denote the set of forward $(t \to t+1)$ and backward $(t+1 \to t)$ flow fields estimated by SPyNet [34], where each vector $\mathbf{F}(i,j) = (u(i,j), v(i,j)) \in \mathbb{R}^2$ represents horizontal and vertical displacements of pixel (i,j). The temporal complexity is decomposed into complementary magnitude and consistency components. The flow magnitude term quantifies average motion intensity across bidirectional flows:

$$T_{\text{magnitude},t} = \frac{1}{|\mathcal{B}_t| \cdot |\Omega|} \sum_{\mathbf{F} \in \mathcal{B}_t} \sum_{(i,j) \in \Omega} \|\mathbf{F}(i,j)\|_2 \,, \tag{4}$$

where $|\mathcal{B}_t| = 2$ accounts for bidirectional flows, $|\Omega| = H \times W$ is the pixel count, and $\|\cdot\|_2$ computes the ℓ_2-norm of displacement vectors. Beyond motion magnitude, we quantify flow consistency by measuring the spatial smoothness of flow vectors. For each flow field $\mathbf{F} \in \mathcal{B}_t$, we compute the Jacobian matrix $J_\mathbf{F}(i,j) \in \mathbb{R}^{2\times 2}$ at pixel (i,j), defined as:

$$J_\mathbf{F}(i,j) = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}, \tag{5}$$

where partial derivatives are approximated via finite differences:

$$\frac{\partial u}{\partial x}(i,j) = u(i+1,j) - u(i,j), \quad \frac{\partial u}{\partial y}(i,j) = u(i,j+1) - u(i,j), \tag{6}$$

and similarly for v-components. The element-wise ℓ_1-norm $\|J_\mathrm{F}(i,j)\|_{1,1}$ aggregates absolute gradients of the flow field. The consistency term aggregates the element-wise ℓ_1-norm of Jacobians across all flows:

$$T_{\text{consistency},t} = \frac{1}{|\mathcal{B}_t| \cdot |\Omega|} \sum_{\mathbf{F} \in \mathcal{B}_t} \sum_{(i,j) \in \Omega} \|J_\mathbf{F}(i,j)\|_{1,1} \,. \tag{7}$$

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

$$T_t = T_{\text{magnitude},t} + \gamma \cdot T_{\text{consistency},t}, \tag{8}$$

where γ is a balancing coefficient that determines the relative importance of motion magnitude versus motion irregularity. To holistically model video complexity and bridge spatial texture details with temporal motion dynamics, we introduce a spatiotemporal complexity metric C_video, which unifies spatial and temporal components into a single guiding signal for bit adaptation. The overall flow-gradient complexity metric is defined as:

$$C_\text{video} = \frac{1}{T} \sum_{t=1}^{T} S_t + \lambda \cdot \frac{1}{T-1} \sum_{t=1}^{T-1} \left(T_{\text{magnitude},t} + \gamma \cdot T_{\text{consistency},t}\right), \tag{9}$$

where S_t represents the spatial complexity of the t-th frame (defined in Equation (3)), $T_{\text{magnitude},t}$ and $T_{\text{consistency},t}$ denote the temporal magnitude and consistency terms at time t (defined in Equations (4) and (7), respectively. The parameter λ balances the relative contributions of spatial texture richness and temporal motion dynamics. Finally, we design a flow-gradient driven bit-width adaptation module that maps C_video to a global bit-width factor. This metric integrates spatial texture details and temporal motion dynamics, enabling a unified characterization of video complexity. During inference, the video-to-bit allocation function converts the computed complexity C_video into a global bit-width adaptation factor b_v, which dynamically adjusts the layer-wise bit-widths of the VSR network. The allocation mechanism operates as follows: if C_video exceeds an upper threshold u_{v2b}, the bit-width factor b_v is set to +1, globally increasing the bit-widths to preserve fidelity in scenes with intricate textures or intense motion; if C_video falls below a lower threshold l_{v2b}, b_v is set to -1, reducing bit-widths to optimize computational efficiency for simpler videos; for values within the thresholds, b_v remains 0, maintaining the default quantization configuration. The thresholds l_{v2b} and u_{v2b} are initialized using percentile-based statistics from a small set of calibration video clips. Specifically, l_{v2b} corresponds to the p_V-th percentile of the video complexity distribution across calibration data, while u_{v2b} is set to the $(100 - p_V)$-th percentile. While the statistical thresholds offer a practical baseline for determining video-adaptive bit-width allocations, we achieve superior optimization performances through threshold fine-tuning.

3.3 Temporal Shared Layer Bit Adaptation

It is widely acknowledged that different layers in neural networks exhibit varying sensitivity to quantization [7, 9, 35, 39]. Layers with lower sensitivity can tolerate aggressive bit-width reduction without significant accuracy degradation, enabling better computational efficiency. To address the unique challenges of VSR networks, where parameters are shared across frames, we propose a temporal shared layer bit adaptation module. For each convolutional layer k, we define two complementary sensitivity metrics that jointly guide layer-wise bit-width adaptation. Given the input video sequences with T frames, we stack activations into $\mathcal{X}^k_{\text{stacked}} \in \mathbb{R}^{T\times B\times C\times H\times W}$ (B: batch size, C: channels, $H \times W$: spatial dimensions), the spatial sensitivity s^k_{space} quantifies intra-frame activation variability through channel-wise standard deviations:

$$s^k_{\text{space}} = \mathbb{E}_{t,b,h,w}\left[\sigma_c\left(\mathcal{X}^k_{\text{stacked},t,b,:,h,w}\right)\right], \tag{10}$$

where $\mathbb{E}_{t,b,h,w}$ denotes averaging over time step t, batch index b and spatial positions (h, w). $\sigma_c(\cdot)$ computes the standard deviation across the channel dimension (dim=2). This captures spatial texture complexity: higher values indicate regions with edges or fine details requiring higher bit-widths. Moreover, the temporal sensitivity s^k_{temp} models inter-frame dynamics as:

$$s^k_{\text{temp}} = \mathbb{E}_{b,c,h,w}\left[\sigma_t\left(\mathcal{X}^k_{\text{stacked},:,b,c,h,w}\right)\right], \tag{11}$$

where $\mathbb{E}_{b,c,h,w}$ averages over batch b, channels c, and spatial coordinates (h, w). $\sigma_t(\cdot)$ calculates the unbiased standard deviation along the temporal dimension (dim=0). Elevated s^k_{temp} reflects dynamic regions with significant motion between frames. The layer-wise bit adaptation factor b^k_L is determined by joint thresholding:

$$b^k_L = \begin{cases} -1, & \text{if } (s^k_{\text{space}}, s^k_{\text{temp}}) \in [0, l_{\text{space}}] \times [0, l_{\text{temp}}] \\ +1, & \text{if } (s^k_{\text{space}}, s^k_{\text{temp}}) \in [u_{\text{space}}, \infty) \times [u_{\text{temp}}, \infty) \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

where spatial thresholds l_{space} and u_{space} are initialized as the p_{space}-th and $(100 - p_{\text{space}})$-th percentiles of spatial sensitivity values $\left\{s^k_{\text{space}}\right\}$ from the calibration dataset, while temporal thresholds l_{temp} and u_{temp} correspond to the p_{temp}-th and $(100 - p_{\text{temp}})$-th percentiles of temporal sensitivities $\left\{s^k_{\text{temp}}\right\}$. The Cartesian product $[0, l_{\text{space}}] \times [0, l_{\text{temp}}]$ identifies stable regions where sensitivities fall below lower percentiles, enabling bit reduction without quality degradation. Conversely, $[u_{\text{space}}, \infty) \times [u_{\text{temp}}, \infty)$ defines regions where both spatial texture complexity and temporal motion dynamics exceed upper percentiles, necessitating increased bit-widths.

3.4 Calibration

For initializing the clipping ranges of quantized activations, we collect minimum and maximum activation statistics [20] by forwarding calibration video clips through the FP network, which are smoothed using the exponential moving average (EMA) to stabilize the lower and upper bounds of each quantizer. Our framework decouples the adaptive bit allocation mechanisms between input video streams and parameter-shared layers. Notably, the calibration phase determines the layer-wise factors $\{b^k_L\}^K_{k=1}$, which are pre-determined and fixed during inference, leaving only the flow-gradient video bit adaptation module active. After $\{b^k_L\}^K_{k=1}$ are determined, we further refine the activation clipping ranges using the OMSE [8] criterion. In short, the bit-width for quantizing a video stream V in layer k is dynamically adjusted through a combination of a flow-gradient video bit adaptation factor and a temporal shared layer bit adaptation factor, expressed as:

$$b^k_V = b_{\text{base}} + b_v + b^k_L, \tag{13}$$

where b^k_V denotes the bit-width for video stream V in layer k, b_{base} is the baseline bit-width, b_v accounts for video complexity, and b^k_L reflects the quantization sensitivity of each layer.

3.5 Fine-tuning

While calibration provides a reasonable starting point for the quantized VSR network, we further fine-tune the bit adaptation modules through calibration data to achieve better adaptation. Since

layer-wise bit adaptation factors $\{b_L^k\}_{k=1}^K$ are pre-configured and remain static during inference, we optimize them directly through gradient descent. In contrast, the video-wise bit adaptation factor b_v is dynamically determined during inference based on video complexity. We therefore fine-tune the flow-gradient video bit adaptation module, which is parameterized by threshold values l_{v2b} and u_{v2b}. To overcome the non-differentiability of the thresholding operation in this module, we employ a tanh-based gradient approximation during backpropagation, inspired by the approach in [12]. Simultaneously, we optimize the clipping boundaries for activation quantization (l_a, u_a) and weight quantization $(-u_w, u_w)$. For the non-differentiable rounding operation in the quantizer, we utilize the Straight-Through Estimator (STE) [1], maintaining the gradient flow by approximating the rounding function as an identity operator during backward passes while preserving its exact behavior in the forward path. During optimization, the super-resolution network's weights remain fixed to maintain the pre-trained knowledge of the full-precision model $\mathcal{P}$. Inspired by the Charbonnier loss [5] in VSR, we design the reconstruction loss as follows:

$$\mathcal{L}_{\text{pix}} = \frac{1}{T}\sum_{i=1}^{T}\sqrt{\left\|P(I_i^{LR}) - Q(I_i^{LR})\right\|_2^2 + \epsilon^2}, \tag{14}$$

where T denotes the number of input video frames, I_i^{LR} represents the i-th low-resolution frame, $P(\cdot)$ and $Q(\cdot)$ correspond to the full-precision (FP) and quantized networks respectively, $\|\cdot\|_2$ is the $\mathcal{L}_2$ norm measuring feature discrepancies, and ϵ is a small positive constant introduced to ensure numerical stability and maintain differentiability of the loss function, particularly when $\mathcal{L}_{\text{pix}}$ approaches zero. For feature-level supervision, we extend the pixel transfer loss [38] to video domains by incorporating temporal dimension awareness. Given intermediate features from the FP network $\mathcal{P}$ and quantized network $\mathcal{Q}$, we first perform layer-wise $\mathcal{L}_2$ normalization across T consecutive frames, and then we calculate the mean square error of these two feature maps across all the blocks and temporal frames. The pixel transfer loss is formulated as follows:

$$\mathcal{L}_{\text{skt}} = \frac{1}{T\cdot K}\sum_{i=1}^{T}\sum_{k=1}^{K}\left\|\frac{F_{\mathcal{P}}^{i,k}}{\left\|F_{\mathcal{P}}^{i,k}\right\|_2} - \frac{F_{\mathcal{Q}}^{i,k}}{\left\|F_{\mathcal{Q}}^{i,k}\right\|_2}\right\|_2^2, \tag{15}$$

where T denotes the number of temporal frames, K represents the total number of feature layers, $F_{\mathcal{P}}^{i,k} \in \mathbb{R}^{C_k\times H_k\times W_k}$ and $F_{\mathcal{Q}}^{i,k} \in \mathbb{R}^{C_k\times H_k\times W_k}$ denote the k-th layer features for the i-th frame from $\mathcal{P}$ and $\mathcal{Q}$ respectively, with (C_k, H_k, W_k) indicating the channel depth and spatial dimensions of the k-th layer outputs. The $\mathcal{L}_2$ normalization ensures scale-invariant feature comparison, while the temporal summation over T frames explicitly models inter-frame relationships in video sequences. Such a supervision paradigm operates solely on low-resolution inputs, eliminating the dependency on high-resolution ground-truth references. Finally, we can get the total loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pix}} + \lambda_{\text{skt}}\mathcal{L}_{\text{skt}}, \tag{16}$$

with λ_{skt} balancing frame-wise reconstruction and temporal feature consistency in recurrent video networks. This formulation enables optimization of quantization parameters across both spatial and temporal dimensions, effectively suppressing error propagation in cyclic computation while preserving video texture details through adaptive adjustment. The quantization parameters are iteratively optimized through three consecutive steps after the calibration phase: initially updating weight clipping ranges $\{u_w^k\}_{k=1}^K$ with activation and bit adaptation module parameters frozen, subsequently refining activation clipping ranges $\{l_a^k, u_a^k\}_{k=1}^K$, and finally adjusting bit adaptive module parameters $l_{v2b}, u_{v2b}, \{b_L^k\}_{k=1}^K$. Each step exclusively optimizes the parameter subset while strictly maintaining the others in fixed states, completing the fine-tuning within only three epochs.

4 Experiments

4.1 Implementation Details

We build the calibration datasets by randomly sampling 100 LR video clips from the REDS [33] and Vimeo-90K [44]. For REDS, we use REDS4, containing four clips, as our test set. In addition, we utilize Vid4 [30], UDM10 [45], and Vimeo-90K-T [44] as test sets along with Vimeo-90K. We train and test models with 4× downsampling using two degradations, bicubic (BI) and blur

downsampling (BD), as BasicVSR [3] did. The calibration datasets calibrate and fine-tune our bit adaptation modules and the quantization ranges. The quantization range for activations is initialized using MinMax [20], and weights OMSE [8]. The parameters γ and λ in the flow-gradient complexity metric are set to 200 and 10. The hyperparameters for calibrating the bit adaptation modules, p_V, p_{space}, and p_{temp} are set to 10, 30 and 30. The constant ϵ in the reconstruction loss is set to 10^{-6}. The parameter λ_{skt} in total loss is set to 0.1. After freezing the network weights, we sequentially fine-tune individual components for one epoch each: optimizing weight clipping ranges in the first epoch, activation clipping ranges in the second epoch, and bit adaptation module parameters in the third epoch. Each stage uses the Adam optimizer [24] with a batch size of 2. The initial learning rates are set to 1×10^{-3} for weight clipping ranges, 1×10^{-5} for activation clipping ranges, and 0.1 for both temporal shared layer bit adaptation and flow-gradient video bit adaptation modules. Each learning rate is decayed by 0.9 every epoch. All experiments follow the same configuration described above, and all parameters are fixed across datasets and models to ensure a fair and consistent evaluation.

4.2 Comparisons with Efficient Video Super-Resolution

We measure reconstruction accuracy using the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) for evaluation metrics. As shown in Table 1, our method achieves superior performance and establishes a new state-of-the-art among efficient VSR approaches. Compared to KSNet [21], our model requires fewer parameters yet outperforms it by 0.12 dB in PSNR on the REDS4 dataset. Compared to the SSL [42] method, our approach maintains comparable model complexity while eliminating the need for ground-truth (GT) videos during training. Notably, we achieve performance gains of 0.2 dB on REDS4 and 0.31 dB on Vimeo90K-T (BD degradation). Moreover, we retrain a unidirectional variant of BasicVSR (BasicVSR-uni) by removing its backward propagation branch, yielding a distinct yet functionally consistent architecture to assess the generalizability of our quantization framework. These quantitative improvements across multiple benchmarks convincingly demonstrate the effectiveness of our quantization framework.

Table 1: **Quantitative comparison (PSNR / SSIM)**. All results are calculated on the Y-channel except REDS4 (RGB-channel). Blank entries correspond to results that cannot be reported. The results for PQ and FQ correspond to Partial Quantized and Fully Quantized models, respectively.

Methods	Params(M)	GT	W / A	BI degradation			BD degradation		
				REDS4 [33]	Vimeo-90K-T [44]	Vid4 [30]	UDM10 [45]	Vimeo-90K-T [44]	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 [44]	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 [45]	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 [43]	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 [42]	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 [46]	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 [42]	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
Ours-PQ	**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**
Ours-FQ	**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**
Ours-PQ	**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**
Ours-FQ	**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**
BasicVSR-uni [3]	2.6	✓	32 / 32	30.54 / 0.8694	36.99 / 0.9429	27.03 / 0.8163	39.29 / 0.9646	37.27 / 0.9473	27.54 / 0.8419
BasicVSR-uni-lite [42]	0.7	✓	32 / 32	29.95 / 0.8561	36.38 / 0.9372	26.68 / 0.8012	38.24 / 0.9586	36.38 / 0.9388	26.87 / 0.8157
L_1-norm-uni [25]	0.7	✓	32 / 32	29.97 / 0.8570	36.45 / 0.9381	26.70 / 0.8031	38.43 / 0.9601	36.53 / 0.9405	26.89 / 0.8187
ASSL-uni [46]	0.7	✓	32 / 32	30.02 / 0.8589	36.49 / 0.9385	26.76 / 0.8051	38.48 / 0.9603	36.61 / 0.9416	27.02 / 0.8236
KSNet-uni [21]	1.6	✓	32 / 32	30.69 / 0.8724	-	27.14 / 0.8208	-	37.34 / 0.9490	-
SSL-uni [42]	0.5	✓	32 / 32	30.24 / 0.8633	36.56 / 0.9392	27.01 / 0.8148	38.68 / 0.9615	36.77 / 0.9429	27.18 / 0.8296
Ours-uni-PQ	**0.8**	✗	**6 / 6MP**	**30.40 / 0.8665**	**36.81 / 0.9413**	**26.91 / 0.8101**	**38.97 / 0.9633**	**37.11 / 0.9460**	**27.40 / 0.8372**
Ours-uni-FQ	**0.55**	✗	**6 / 6MP**	**30.35 / 0.8646**	**36.63 / 0.9396**	**26.81 / 0.8060**	**38.70 / 0.9610**	**36.83 / 0.9434**	**27.29 / 0.8310**
Ours-uni-PQ	**0.7**	✗	**4 / 4MP**	**29.57 / 0.8460**	**35.96 / 0.9323**	**26.32 / 0.7784**	**37.78 / 0.9550**	**36.22 / 0.9366**	**26.71 / 0.8067**
Ours-uni-FQ	**0.4**	✗	**4 / 4MP**	**29.55 / 0.8434**	**35.75 / 0.9296**	**26.15 / 0.7674**	**37.29 / 0.9517**	**35.72 / 0.9319**	**26.53 / 0.7938**

4.3 Comparison with Static Quantization

To substantiate the effectiveness of our method, we benchmark against conventional static quantization techniques without QAT. Comparisons include standard PTQ approaches MinMax [20] and Percentile [27]. To ensure fair comparison, we extend these methods by incorporating fine-tuning (FT) steps to existing PTQ approaches, denoted as MinMax+FT and Percentile+FT, where FT employs

our parameter optimization strategy described in Sec. 3.5. In addition, we also adapt DBDC [38], the first image super-resolution quantization method, into the video setting. Since video SR networks typically adopt a recurrent architecture with parameter sharing across frames, image-specific quantization strategies cannot be directly applied. To address this, we apply DBDC's clipping-boundary estimation independently on each frame, and then aggregate the per-frame boundaries via an EMA strategy across the temporal dimension. This modification preserves the core idea of DBDC (dynamic clipping with EMA) while making it compatible with recurrent video SR models, thus providing a meaningful baseline for comparison. As Table 2 demonstrates, our adaptive quantization surpasses all counterparts in 4-bit weights. Extended experiments validating the applicability of our method are included in the supplementary materials. Qualitative comparisons are shown in Fig. 5, where it can be seen that only our method successfully restores the contours.

Table 2: **Comparisons with static quantization on VSR**. All results are calculated on the Y-channel except REDS4 (RGB-channel). "Time (min)" indicates the total processing time for each method.

Methods	FT	W / A	Time (min)	BI degradation REDS4 [33]	BI degradation Vimeo-90K-T [44]	BI degradation Vid4 [30]	BD degradation UDM10 [45]	BD degradation Vimeo90K-T [44]	BD degradation Vid4 [30]
BasicVSR [3]	-	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-MinMax [20]	✗	4 / 4	6	28.12 / 0.7896	34.89 / 0.9182	26.03 / 0.7585	35.37 / 0.9251	34.32 / 0.9106	25.64 / 0.7412
BasicVSR-Percentile [27]	✗	4 / 4	3	27.78 / 0.7841	34.37 / 0.9159	25.23 / 0.7305	35.06 / 0.9341	34.11 / 0.9135	25.08 / 0.7275
BasicVSR-DBDC [38]	✗	4 / 4	30	28.07 / 0.7817	35.23 / 0.9225	26.24 / 0.7720	35.96 / 0.9339	34.77 / 0.9172	26.01 / 0.7581
BasicVSR-MinMax+FT	✓	4 / 4	70	29.21 / 0.8238	35.22 / 0.9212	26.17 / 0.7601	36.44 / 0.9393	34.88 / 0.9176	26.11 / 0.7584
BasicVSR-Percentile+FT	✓	4 / 4	70	28.30 / 0.8054	34.40 / 0.9159	25.26 / 0.7314	35.32 / 0.9362	34.20 / 0.9142	25.26 / 0.7340
BasicVSR-DBDC+FT	✓	4 / 4	95	29.24 / 0.8232	35.45 / 0.9243	26.31 / 0.7703	36.82 / 0.9434	35.25 / 0.9228	26.42 / 0.7764
BasicVSR-Ours	✓	**4 / 4MP**	**90**	**30.26 / 0.8637**	**35.82 / 0.9311**	**26.29 / 0.7752**	**37.59 / 0.9536**	**35.95 / 0.9339**	**26.81 / 0.8025**

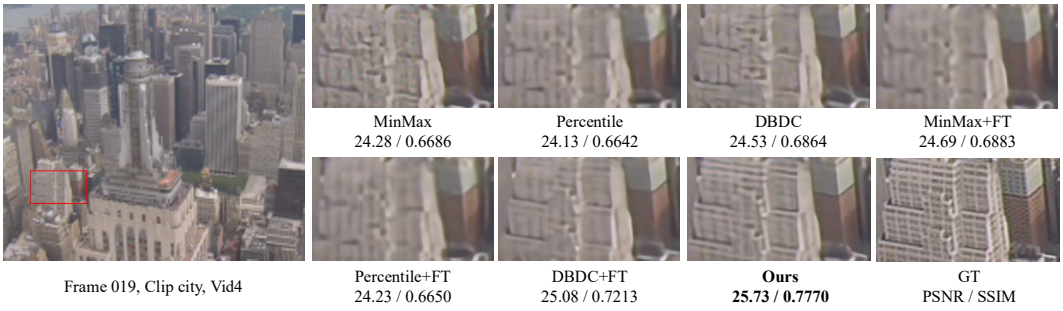


Figure 2: Qualitative comparison on Vid4 [30]. Only our method reconstructs both structural contours and fine-grained details. (**Zoom-in for best view**)

4.4 Ablation Study

Table 3 reports a component-wise ablation on REDS4 with 4-bit fully quantized BasicVSR. The first row corresponds to the full-precision (FP32) baseline, where no calibration is performed and the network therefore runs without quantization. Activating calibration alone (row 2) applies 4-bit quantization and causes the expected accuracy drop. Fine-tuning on the small calibration set (row 3) restores most of the accuracy lost after calibration. Enabling FG-VBA or TS-LBA individually (rows 4–5) provides further gains, and combining both adaptation modules (row 6) delivers the best overall performance, surpassing the FP32 model in perceptual quality, while being constrained to 4-bit quantization. These results demonstrate that each component, calibration, fine-tuning, and the two adaptation modules, contributes significantly to the final reconstruction accuracy.

Table 3: **Ablation study** on each attribute of our method evaluated on REDS4 with 4-bit FQ BasicVSR. FG-VBA and TS-LBA denote the flow-gradient video bit adaptation and temporal shared layer bit adaptation modules. Calib is the calibration, and FT denotes fine-tuning. The full configuration gives the best reconstruction accuracy.

FG-VBA	TS-LBA	Calib	FT	PSNR	SSIM
-	-	-	-	31.42	0.8909
-	-	✓	-	28.34	0.7933
-	-	✓	✓	29.05	0.8180
✓	-	✓	✓	29.96	0.8502
-	✓	✓	✓	30.14	0.8606
✓	✓	✓	✓	**30.26**	**0.8637**

Table 4: **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 [42]	✓	26600	✓	8	303,380	370hrs	54ms	8
Ours	✓	**100**	✗	**2**	**150**	**90min**	**8ms**	**1**

4.5 Computational Efficiency Comparison

We analyze the processing time of our framework in comparison to existing SOTA efficient methods in Table 4. In terms of processing time, our method is over $\times 200$ faster than SSL [42]. The calculation is as follows: 370 hours $\times$ 60 = 22,200 minutes; 22,200 minutes / 90 minutes $\approx 247\times$. Furthermore, our method only requires 1 GPU, whereas SSL [42] requires 8 GPUs. These results demonstrate that our method requires significantly less processing time and fewer GPU resources compared to other SOTA efficient VSR compression methods. Note that this comparison specifically evaluates model compression techniques applied to pretrained VSR models, rather than comparing quantization with core VSR training. Both SSL and our method are compression approaches operating on the same pretrained VSR backbone. Beyond the drastic reduction in tuning cost and GPU resource demand, our method also achieves a superior runtime of 8 ms. This is notably faster than other methods, making the quantized model highly efficient for practical deployment. Combined with its minimal data requirement (only 100 video clips without the need for GT), our approach presents a highly practical and efficient compression solution for VSR models. To the best of our knowledge, this work presents the first successful quantization scheme for video super-resolution models, delivering a breakthrough in efficiency while maintaining competitive performance.

4.6 Error Bars Evaluation

PTQ for VSR requires a calibration dataset of unlabeled LR video clips to estimate activation statistics. An important consideration for practical deployment is the sensitivity of the quantized model's performance to the composition of this dataset. Notably, these calibration data can be sourced directly from the target application's video streams, eliminating the need for separate data collection. In our main experiments, calibration data were sampled using a fixed random seed for consistency. To rigorously evaluate the robustness of our method, we constructed multiple calibration datasets by sampling 100 LR video clips using different random seeds (0, 1, 2, 3, and 321), where seed 1 served as the default in our main experiments. The performance variations across these datasets are quantified in Table 5. The results show consistently high performance with low variance across different calibration sets, confirming that our method is robust to variations in the calibration data. This demonstrates that in practice, any representative sample from the deployment domain can be effectively used without requiring carefully curated data. Our approach therefore provides both superior quantitative performance and practical robustness for real-world applications.

Table 5: **Error bars evaluation with 4-bit fully quantized BasicVSR (PSNR / SSIM)**. All results are calculated on the Y-channel except REDS4 (RGB-channel). The random seeds are taken from 0, 1, 2, 3, 321 to generate calibration datasets.

Methods	seed	BI degradation			BD degradation		
		REDS4 [33]	Vimeo-90K-T [44]	Vid4 [30]	UDM10 [45]	Vimeo90K-T [44]	Vid4 [30]
BasicVSR [3]	-	31.42 / 0.8909	37.18 / 0.9450	27.24 / 0.8251	39.96 / 0.9694	37.53 / 0.9498	27.96 / 0.8553
Ours	0	30.09 / 0.8562	36.03 / 0.9335	26.29 / 0.7772	37.78 / 0.9554	35.93 / 0.9336	26.80 / 0.8040
	1	**30.26 / 0.8637**	**35.82 / 0.9311**	**26.29 / 0.7752**	**37.59 / 0.9536**	**35.95 / 0.9339**	**26.81 / 0.8025**
	2	29.90 / 0.8511	35.81 / 0.9311	26.22 / 0.7738	37.94 / 0.9563	36.02 / 0.9347	26.77 / 0.8032
	3	30.19 / 0.8618	35.93 / 0.9320	26.19 / 0.7712	37.85 / 0.9557	36.08 / 0.9356	26.93 / 0.8114
	321	30.09 / 0.8598	35.77 / 0.9299	25.84 / 0.7530	37.78 / 0.9546	35.97 / 0.9335	26.88 / 0.8056
	Mean	30.11 / 0.8578	35.87 / 0.9315	26.17 / 0.7701	37.79 / 0.9567	35.99 / 0.9343	26.84 / 0.8053
	Std	0.12 / 0.0038	0.10 / 0.0011	0.17 / 0.0088	0.12 / 0.0027	0.05 / 0.0019	0.06 / 0.0032

5 Conclusion

In this paper, we present the first quantization method for VSR with a few unlabeled calibration video clips that adaptively learns the bit adaptation strategies. Different from image SR, the temporal error propagation, shared temporal parameterization, and temporal metric mismatch significantly degrade the performance of the quantized VSR model. To alleviate these problems, we propose a temporal awareness adaptation post-training quantization framework for VSR. With the flow-gradient video bit adaptation and temporal shared layer bit adaptation, we could get the initial bit-width for videos and layers in VSR models, then optimize them with calibration and a novel fine-tuning method with the supervision of the full-precision model. The results demonstrate that our method not only achieves superior accuracy compared to SOTA methods but also significantly reduces the tuning cost and achieves an ultrafast runtime, underscoring its high practicality.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (Nos. 62276182, 62476196), Emerging Frontiers Cultivation Program of Tianjin University Interdisciplinary Center, Tianjin Natural Science Foundation (Nos. 24JCYBJC01230, 24JCYBJC01460), and Peng Cheng Lab Program (No. PCL2023A08).

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4778–4787, 2017.

[3] Kelvin C.K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4947–4956, June 2021.

[4] Kelvin C.K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5972–5981, June 2022.

[5] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st international conference on image processing*, volume 2, pages 168–172. IEEE, 1994.

[6] Wei-Hao Chen, Chunmeng Dou, Kai-Xiang Li, Wei-Yu Lin, Pin-Yi Li, Jian-Hao Huang, Jing-Hong Wang, Wei-Chen Wei, Cheng-Xin Xue, Yen-Cheng Chiu, et al. Cmos-integrated memristive non-volatile computing-in-memory for ai edge processors. *Nature Electronics*, 2(9):420–428, 2019.

[7] Weihan Chen, Peisong Wang, and Jian Cheng. Towards mixed-precision quantization of neural networks via constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5350–5359, 2021.

[8] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019.

[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 293–302, 2019.

[10] Dario Fuoli, Shuhang Gu, and Radu Timofte. Efficient video super-resolution through recurrent latent space propagation. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3476–3485. IEEE, 2019.

[11] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[12] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4852–4861, 2019.

[13] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3897–3906, 2019.

[14] Cheeun Hong, Sungyong Baik, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Cadyq: Content-aware dynamic quantization for image super-resolution. In *European Conference on Computer Vision*, pages 367–383. Springer, 2022.

[15] Cheeun Hong, Heewon Kim, Sungyong Baik, Junghun Oh, and Kyoung Mu Lee. Daq: Channel-wise distribution-aware quantization for deep image super-resolution networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2675–2684, January 2022.

[16] Cheeun Hong and Kyoung Mu Lee. Adabm: on-the-fly adaptive bit mapping for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2641–2650, 2024.

[17] Takashi Isobe, Xu Jia, Shuhang Gu, Songjiang Li, Shengjin Wang, and Qi Tian. Video super-resolution with recurrent structure-detail network. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pages 645–660. Springer, 2020.

[18] Takashi Isobe, Songjiang Li, Xu Jia, Shanxin Yuan, Gregory Slabaugh, Chunjing Xu, Ya-Li Li, Shengjin Wang, and Qi Tian. Video super-resolution with temporal group attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8008–8017, 2020.

[19] Takashi Isobe, Fang Zhu, Xu Jia, and Shengjin Wang. Revisiting temporal modeling for video super-resolution. *arXiv preprint arXiv:2008.05765*, 2020.

[20] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[21] Shuo Jin, Meiqin Liu, Chao Yao, Chunyu Lin, and Yao Zhao. Kernel dimension matters: To activate available kernels for real-time video super-resolution. In *Proceedings of the 31st ACM International Conference on Multimedia*, MM '23, page 8617–8625, New York, NY, USA, 2023. Association for Computing Machinery.

[22] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3224–3232, 2018.

[23] Sergey Kastryulin, Jamil Zakirov, Denis Prokopenko, and Dmitry V. Dylov. Pytorch image quality: Metrics for image quality assessment, 2022.

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

[25] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[26] Huixia Li, Chenqian Yan, Shaohui Lin, Xiawu Zheng, Baochang Zhang, Fan Yang, and Rongrong Ji. Pams: Quantized super-resolution via parameterized max scale. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 564–580. Springer, 2020.

[27] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2810–2819, 2019.

[28] Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. Vrt: A video restoration transformer. *IEEE Transactions on Image Processing*, 33:2171–2182, 2024.

[29] Jingyun Liang, Yuchen Fan, Xiaoyu Xiang, Rakesh Ranjan, Eddy Ilg, Simon Green, Jiezhang Cao, Kai Zhang, Radu Timofte, and Luc V Gool. Recurrent video restoration transformer with guided deformable attention. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 378–393. Curran Associates, Inc., 2022.

[30] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *IEEE transactions on pattern analysis and machine intelligence*, 36(2):346–360, 2013.

[31] Anish Mittal, Anush K Moorthy, and Alan C Bovik. Blind/referenceless image spatial quality evaluator. In *2011 conference record of the forty fifth asilomar conference on signals, systems and computers (ASILOMAR)*, pages 723–727. IEEE, 2011.

[32] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012.

[33] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019.

[34] 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.

[35] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Wen Ji, Yaowei Wang, and Wenwu Zhu. Mixed-precision neural network quantization via learned layer-wise importance. In *European conference on computer vision*, pages 259–275. Springer, 2022.

[36] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE international conference on computer vision*, pages 4472–4480, 2017.

[37] Senmao Tian, Ming Lu, Jiaming Liu, Yandong Guo, Yurong Chen, and Shunli Zhang. Cabm: Content-aware bit mapping for single image super-resolution network with large input. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1756–1765, 2023.

[38] Zhijun Tu, Jie Hu, Hanting Chen, and Yunhe Wang. Toward accurate post-training quantization for image super resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5856–5865, 2023.

[39] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

[40] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019.

[41] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

[42] Bin Xia, Jingwen He, Yulun Zhang, Yitong Wang, Yapeng Tian, Wenming Yang, and Luc Van Gool. Structured sparsity learning for efficient video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22638–22647, June 2023.

[43] Zeyu Xiao, Xueyang Fu, Jie Huang, Zhen Cheng, and Zhiwei Xiong. Space-time distillation for video super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2113–2122, 2021.

[44] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127:1106–1125, 2019.

[45] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, and Jiayi Ma. Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3106–3115, 2019.

[46] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Aligned structured sparsity learning for efficient image super-resolution. *Advances in Neural Information Processing Systems*, 34:2695–2706, 2021.

[47] Yunshan Zhong, Mingbao Lin, Xunchao Li, Ke Li, Yunhang Shen, Fei Chao, Yongjian Wu, and Rongrong Ji. Dynamic dual trainable bounds for ultra-low precision super-resolution networks. In *European Conference on Computer Vision*, pages 1–18. Springer, 2022.

A More Details on Datasets and Evaluation Metrics

Training and testing datasets. For video super-resolution at a scale factor of 4, we train the model using two distinct datasets. In one setting, we generate LR frames by applying the MATLAB `imresize` function with bicubic interpolation, and the model is trained on the REDS dataset [33], using the REDS4 subset, which consists of clips 000, 011, 015, and 020, for evaluation. In another setting, the model is trained on the Vimeo-90K dataset [44] using two types of degradation: bicubic interpolation and a blur-downsampling approach that applies a Gaussian blur with a standard deviation of 1.6 followed by subsampling. The evaluation is conducted on three datasets, where Vimeo-90K-T [44] and Vid4 [30] are tested under both degradation settings, while UDM10 [45] is evaluated using only the blur-downsampling setting.

REDS [33] is a standard proposed high-quality video dataset with a resolution of 1280×720, designed for video super-resolution. It includes 270 clips that serve both training and validation purposes. Following the setting in [33], we use REDS4, which consists of four representative clips (000, 011, 015 and 020), as the evaluation set, while the remaining 266 clips are used for training. This dataset is employed for training video super-resolution models under bicubic degradation.

Vimeo-90K [44] is a widely used video dataset with a resolution of 448×256, commonly adopted for video super-resolution tasks. It provides 64,612 clips for training and 7,824 clips for testing, with the test set referred to as Vimeo-90K-T. This dataset is utilized for training video super-resolution models under both bicubic interpolation and blur-downsampling degradations.

Vid4 [30] is a classic dataset commonly used for evaluating video restoration methods. It comprises four video clips named calendar, city, foliage, and walk, each consisting of at least 34 frames with a resolution of 720×480.

UDM10 [45] is a standard benchmark dataset introduced for evaluating video super-resolution methods. It consists of 4 video clips covering diverse scenes, with each clip containing 32 frames at a resolution of 1272×720.

Evaluation metrics. Following [3, 33], the evaluation metrics are computed on the RGB channel for REDS4 [33], while for Vimeo-90K-T [44], Vid4 [30], and UDM10 [45], the calculations are performed on the Y channel.

B Additional Experiments

B.1 Per-stage Fine-tuning Analysis

As detailed in Section 3.5, our fine-tuning process consists of three sequential stages. This design ensures stable optimization by progressively refining different subsets of quantization parameters. To provide a comprehensive analysis, Table 6 reports the performance and feature average bit-width (FAB), which is calculated across all video clips in the test dataset after each stage, with the analysis conducted on the BasicVSR architecture under an FQ setting.

Stage 1: Weight Clipping Optimization. In this initial stage, we exclusively optimize the clipping parameters ($\{u_w^k\}_{k=1}^K$) for weights, while freezing the quantization parameters ($\{l_a^k, u_a^k\}_{k=1}^K$) for activations and the bit adaptation parameters ($\{b_L^k\}_{k=1}^K, l_{v2b}, u_{v2b}$). The fact that this stage alone already surpasses strong baselines like DBDC [38] in accuracy, while maintaining a lower FAB, demonstrates a key finding: our framework, even with its bit adaptation modules not yet finely tuned, can achieve a superior performance-efficiency trade-off purely through optimized static quantization of weights. This establishes a strong and efficient baseline.

Stage 2: Activation Clipping Optimization. We then refine the clipping ranges ($\{l_a^k, u_a^k\}_{k=1}^K$) for activations, with other parameters fixed. This step addresses the dynamic, input-dependent nature of activation distributions. We observe a consistent performance improvement across all datasets with a negligible change in FAB, indicating that the activation quantization parameters are being effectively tuned without compromising efficiency.

Stage 3: Bit Adaptation Modules Optimization. The final stage unlocks the full potential of our framework by optimizing the bit adaptation parameters, including the TS-LBA factors ($\{b_L^k\}_{k=1}^K$) and the FG-VBA thresholds (l_{v2b}, u_{v2b}). As shown in Table 6, this stage yields a significant per-

formance boost by enabling dynamic bit-width allocation, at the cost of a moderate and controlled increase in FAB. This introduces a tunable trade-off: users can opt for the high-efficiency model from Stage 1 or 2, or the high-performance model from Stage 3, depending on their specific deployment constraints. Note that the weight bit-width remains fixed at the base precision (e.g., 4 bits in "w4a4") throughout all stages; only the activation bit-width is dynamically adapted.

Table 6: **Performance and efficiency analysis across different fine-tuning stages. Results are reported in metrics of PSNR / SSIM / FAB.** All the results are calculated on the Y-channel except REDS4 (RGB-channel).

Methods	W/A	Time (min)	BI degradation			BD degradation		
			REDS4 [33]	Vimeo-90K-T [44]	Vid4 [30]	UDM10 [45]	Vimeo90K-T [44]	Vid4 [30]
DBDC [38]	4 / 4	30	28.07 / 0.7817 / 4.00	35.23 / 0.9225 / 4.00	26.24 / 0.7720 / 4.00	35.96 / 0.9339 / 4.00	34.77 / 0.9172 / 4.00	26.01 / 0.7581 / 4.00
Ours-Stage1	**4 / 4MP**	**40**	**29.82 / 0.8466 / 3.98**	**35.46 / 0.9266 / 3.95**	**26.12 / 0.7666 / 3.75**	**36.69 / 0.9464 / 3.70**	**35.45 / 0.9272 / 3.95**	**26.32 / 0.7716 / 3.75**
Ours-Stage2	**4 / 4MP**	**65**	**29.86 / 0.8479 / 3.98**	**35.51 / 0.9270 / 3.95**	**26.21 / 0.7695 / 3.75**	**36.76 / 0.9471 / 3.70**	**35.54 / 0.9283 / 3.95**	**26.38 / 0.7741 / 3.75**
Ours-Stage3	**4 / 4MP**	**90**	**30.26 / 0.8637 / 5.50**	**35.82 / 0.9311 / 5.03**	**26.29 / 0.7752 / 4.69**	**37.59 / 0.9536 / 4.78**	**35.95 / 0.9339 / 5.08**	**26.81 / 0.8025 / 4.98**

B.2 Full Quantization vs. Partial Quantization

In this work, we compare VSR networks with existing static quantization methods that do not utilize QAT. Meanwhile, the majority of SR quantization approaches employ a partial quantization scheme, restricting quantization exclusively to the network's body module. Therefore, we analyze the effect of fully quantizing the network in Table 7. Despite its limited impact on reducing total computational cost, partial quantization yields notably higher reconstruction accuracy. Overall, our method delivers higher accuracy while maintaining comparable computational cost in both partial and full quantization settings. For the evaluation metric, we measure reconstruction accuracy using PSNR, SSIM, and FAB. We further validate our quantization framework on the unidirectional variant, BasicVSR-uni, to assess its generalizability across architectures. As shown in the lower part of the table, our method consistently outperforms existing quantization schemes on both BasicVSR and BasicVSR-uni, confirming the robustness of the proposed design.

B.3 Comprehensive Error Bars Evaluation with FAB

In our main experiments, calibration data were sampled using a fixed random seed (Seed 1) for consistency. To provide a more comprehensive robustness analysis, we extend the error bars evaluation presented in Table 5 of the main text by further including the variation in FAB. As in the main text, we use random seeds (0, 1, 2, 3, 321) to sample calibration datasets. The results, presented in Table 8, demonstrate that our method exhibits not only stable performance (PSNR / SSIM) but also a stable FAB across different calibration video clips. The low standard deviation in FAB confirms that the bit-width allocation is robust and does not fluctuate excessively with the calibration data.

B.4 Comprehensive Quantitative Comparisons with Perceptual Quality Metrics

As GT is not available for real-world videos, full-reference metrics such as PSNR and SSIM cannot be applied in such cases. To provide a more comprehensive assessment, we additionally report two widely used no-reference perceptual quality metrics, NIQE [32] and BRISQUE [31], on the REDS4 [33] and Vid4 [30] datasets. The corresponding results are provided in the Table 9. Compared to the efficient VSR method SSL [42], our 6-bit models (1.0-1.3M) are competitive or better in PSNR / SSIM while using far lower bits. Compared to the FP BasicVSR (32 / 32, 4.9M), our 6-bit models narrow the fidelity gap to within 0.2-0.3 dB on average while using 4-5$\times$ fewer parameters, and with perceptual scores that are comparable or better in several cases. In addition, under the same 4-bit FQ setting, our method consistently delivers higher fidelity and strong perceptual quality than standard PTQ baselines, including DBDC [38]. On REDS4 [33], our method (4 / 4, FQ) improves PSNR over DBDC+FT by +1.04 dB (30.26 vs. 29.22) with a lower NIQE (20.71 vs. 20.98). On Vid4 [30] (BI), we achieve a comparable PSNR (26.28 vs. 26.31) but with better NIQE and BRISQUE (21.20 / 45.30 vs. 21.91 / 46.83). On Vid4 [30] (BD), our method outperforms DBDC+FT by +0.39 dB in PSNR (26.81 vs. 26.42) and achieves a lower NIQE (21.27 vs. 22.01). Furthermore, to validate the generalization of our method across different architectures, we also evaluate on BasicVSR-uni variant. On this architecture, our quantized models maintain competitive performance against the full-precision baseline while utilizing significantly fewer parameters and consistently outperform other PTQ methods, demonstrating robust generalization capability. In summary, unlike other PTQ methods (MinMax [20], Percentile [27], DBDC [38]) that often trade

Table 7: **Comparisons between fully quantization and partial quantization on VSR in terms of PSNR, SSIM and FAB**. All the results are calculated on the Y-channel except REDS4 (RGB-channel) and FQ denotes Full Quantization models.

Methods	FQ	W/A	BI degradation			BD degradation		
			REDS4 [33]	Vimeo-90K-T [44]	Vid4 [30]	UDM10 [45]	Vimeo90K-T [44]	Vid4 [30]
BasicVSR [3]	-	32 / 32	31.42 / 0.8909 / 32.00	37.18 / 0.9450 / 32.00	27.24 / 0.8251 / 32.00	39.96 / 0.9694 / 32.00	37.53 / 0.9498 / 32.00	27.96 / 0.8553 / 32.00
BasicVSR-MinMax [20]	✗	4 / 4	28.16 / 0.7915 / 4.00	34.95 / 0.9189 / 4.00	26.06 / 0.7594 / 4.00	35.48 / 0.9272 / 4.00	34.41 / 0.9120 / 4.00	25.66 / 0.7426 / 4.00
BasicVSR-Percentile [27]	✗	4 / 4	28.89 / 0.8208 / 4.00	36.08 / 0.9345 / 4.00	26.51 / 0.7909 / 4.00	37.59 / 0.9537 / 4.00	35.96 / 0.9348 / 4.00	26.62 / 0.8000 / 4.00
BasicVSR-DBDC [38]	✗	4 / 4	28.11 / 0.7836 / 4.00	35.29 / 0.9232 / 4.00	26.27 / 0.7725 / 4.00	36.04 / 0.9350 / 4.00	34.84 / 0.9182 / 4.00	26.03 / 0.7589 / 4.00
BasicVSR-MinMax+FT	✗	4 / 4	29.21 / 0.8217 / 4.00	35.25 / 0.9219 / 4.00	26.11 / 0.7612 / 4.00	36.42 / 0.9399 / 4.00	34.86 / 0.9178 / 4.00	26.05 / 0.7570 / 4.00
BasicVSR-Percentile+FT	✗	4 / 4	29.92 / 0.8513 / 4.00	36.17 / 0.9357 / 4.00	26.60 / 0.7948 / 4.00	37.77 / 0.9573 / 4.00	36.18 / 0.9375 / 4.00	26.78 / 0.8075 / 4.00
BasicVSR-DBDC+FT	✗	4 / 4	29.22 / 0.8223 / 4.00	35.45 / 0.9245 / 4.00	26.12 / 0.7636 / 4.00	36.91 / 0.9450 / 4.00	35.32 / 0.9241 / 4.00	26.41 / 0.7795 / 4.00
BasicVSR-Ours	✗	**4 / 4MP**	**30.34 / 0.8657 / 5.48**	**35.93 / 0.9315 / 3.95**	**26.26 / 0.7764 / 3.75**	**38.15 / 0.9576 / 4.75**	**36.46 / 0.9387 / 5.05**	**27.02 / 0.8161 / 4.96**
BasicVSR-MinMax [20]	✓	4 / 4	28.12 / 0.7896 / 4.00	34.89 / 0.9182 / 4.00	26.03 / 0.7585 / 4.00	35.37 / 0.9251 / 4.00	34.32 / 0.9106 / 4.00	25.64 / 0.7412 / 4.00
BasicVSR-Percentile [27]	✓	4 / 4	27.78 / 0.7841 / 4.00	34.37 / 0.9159 / 4.00	25.23 / 0.7305 / 4.00	35.06 / 0.9341 / 4.00	34.11 / 0.9135 / 4.00	25.08 / 0.7275 / 4.00
BasicVSR-DBDC [38]	✓	4 / 4	28.07 / 0.7817 / 4.00	35.23 / 0.9225 / 4.00	26.24 / 0.7720 / 4.00	35.96 / 0.9339 / 4.00	34.77 / 0.9172 / 4.00	26.01 / 0.7581 / 4.00
BasicVSR-MinMax+FT	✓	4 / 4	29.21 / 0.8238 / 4.00	35.22 / 0.9212 / 4.00	26.17 / 0.7601 / 4.00	36.44 / 0.9393 / 4.00	34.88 / 0.9176 / 4.00	26.11 / 0.7584 / 4.00
BasicVSR-Percentile+FT	✓	4 / 4	28.30 / 0.8054 / 4.00	34.40 / 0.9159 / 4.00	25.26 / 0.7314 / 4.00	35.32 / 0.9362 / 4.00	34.20 / 0.9142 / 4.00	25.26 / 0.7340 / 4.00
BasicVSR-DBDC+FT	✓	4 / 4	29.24 / 0.8232 / 4.00	35.45 / 0.9243 / 4.00	26.31 / 0.7703 / 4.00	36.82 / 0.9434 / 4.00	35.25 / 0.9228 / 4.00	26.42 / 0.7764 / 4.00
BasicVSR-Ours	✓	**4 / 4MP**	**30.26 / 0.8637 / 5.50**	**35.82 / 0.9311 / 5.03**	**26.29 / 0.7752 / 4.69**	**37.59 / 0.9536 / 4.78**	**35.95 / 0.9339 / 5.08**	**26.81 / 0.8025 / 4.98**
BasicVSR-MinMax [20]	✗	6 / 6	31.19 / 0.8846 / 6.00	36.97 / 0.9429 / 6.00	27.16 / 0.8199 / 6.00	39.53 / 0.9665 / 6.00	37.25 / 0.9470 / 6.00	27.79 / 0.8486 / 6.00
BasicVSR-Percentile [27]	✗	6 / 6	31.04 / 0.8845 / 6.00	36.77 / 0.9427 / 6.00	26.94 / 0.8121 / 6.00	38.81 / 0.9664 / 6.00	36.95 / 0.9469 / 6.00	27.39 / 0.8376 / 6.00
BasicVSR-DBDC [38]	✗	6 / 6	31.12 / 0.8842 / 6.00	37.01 / 0.9435 / 6.00	27.17 / 0.8214 / 6.00	39.61 / 0.9672 / 6.00	37.29 / 0.9476 / 6.00	27.80 / 0.8503 / 6.00
BasicVSR-MinMax+FT	✗	6 / 6	31.22 / 0.8848 / 6.00	36.97 / 0.9426 / 6.00	27.17 / 0.8201 / 6.00	39.50 / 0.9663 / 6.00	37.23 / 0.9466 / 6.00	27.79 / 0.8465 / 6.00
BasicVSR-Percentile+FT	✗	6 / 6	31.07 / 0.8851 / 6.00	36.80 / 0.9427 / 6.00	26.97 / 0.8137 / 6.00	38.85 / 0.9663 / 6.00	36.97 / 0.9465 / 6.00	27.41 / 0.8370 / 6.00
BasicVSR-DBDC+FT	✗	6 / 6	31.23 / 0.8848 / 6.00	36.99 / 0.9430 / 6.00	27.16 / 0.8091 / 6.00	39.59 / 0.9670 / 6.00	37.28 / 0.9471 / 6.00	27.85 / 0.8479 / 6.00
BasicVSR-Ours	✗	**6 / 6MP**	**31.26 / 0.8879 / 7.48**	**37.07 / 0.9440 / 7.07**	**27.18 / 0.8215 / 6.73**	**39.64 / 0.9680 / 6.77**	**37.37 / 0.9485 / 7.06**	**27.84 / 0.8495 / 6.97**
BasicVSR-MinMax [20]	✓	6 / 6	31.11 / 0.8821 / 6.00	36.79 / 0.9411 / 6.00	27.10 / 0.8172 / 6.00	39.14 / 0.9637 / 6.00	36.96 / 0.9441 / 6.00	27.71 / 0.8438 / 6.00
BasicVSR-Percentile [27]	✓	6 / 6	28.49 / 0.8098 / 6.00	34.70 / 0.9211 / 6.00	25.42 / 0.7451 / 6.00	35.56 / 0.9409 / 6.00	34.49 / 0.9196 / 6.00	25.41 / 0.7485 / 6.00
BasicVSR-DBDC [38]	✓	6 / 6	31.07 / 0.8822 / 6.00	36.86 / 0.9421 / 6.00	27.13 / 0.8194 / 6.00	39.30 / 0.9651 / 6.00	37.05 / 0.9453 / 6.00	27.72 / 0.8460 / 6.00
BasicVSR-MinMax+FT	✓	6 / 6	31.12 / 0.8816 / 6.00	36.77 / 0.9402 / 6.00	27.07 / 0.8151 / 6.00	39.12 / 0.9632 / 6.00	36.95 / 0.9434 / 6.00	27.67 / 0.8438 / 6.00
BasicVSR-Percentile+FT	✓	6 / 6	29.02 / 0.8313 / 6.00	34.75 / 0.9219 / 6.00	25.51 / 0.7497 / 6.00	35.88 / 0.9438 / 6.00	34.65 / 0.9218 / 6.00	25.59 / 0.7571 / 6.00
BasicVSR-DBDC+FT	✓	6 / 6	31.14 / 0.8826 / 6.00	36.82 / 0.9412 / 6.00	27.09 / 0.8143 / 6.00	39.28 / 0.9647 / 6.00	37.06 / 0.9448 / 6.00	27.72 / 0.8417 / 6.00
BasicVSR-Ours	✓	**6 / 6MP**	**31.17 / 0.8849 / 7.47**	**36.79 / 0.9409 / 6.99**	**27.05 / 0.8140 / 6.65**	**39.22 / 0.9646 / 6.66**	**37.02 / 0.9444 / 6.95**	**27.69 / 0.8405 / 6.86**
BasicVSR-uni [3]	-	32 / 32	30.54 / 0.8694 / 32.00	36.99 / 0.9429 / 32.00	27.03 / 0.8163 / 32.00	39.29 / 0.9646 / 32.00	37.27 / 0.9473 / 32.00	27.54 / 0.8419 / 32.00
BasicVSR-uni-MinMax [20]	✗	4 / 4	28.48 / 0.8012 / 4.00	34.40 / 0.9103 / 4.00	25.63 / 0.7342 / 4.00	35.59 / 0.9291 / 4.00	34.60 / 0.9153 / 4.00	25.84 / 0.7550 / 4.00
BasicVSR-uni-Percentile [27]	✗	4 / 4	29.28 / 0.8321 / 4.00	35.45 / 0.9274 / 4.00	26.11 / 0.7717 / 4.00	37.19 / 0.9516 / 4.00	35.78 / 0.9330 / 4.00	26.41 / 0.7963 / 4.00
BasicVSR-uni-DBDC [38]	✗	4 / 4	28.39 / 0.7964 / 4.00	34.96 / 0.9187 / 4.00	25.89 / 0.7524 / 4.00	35.89 / 0.9337 / 4.00	34.78 / 0.9180 / 4.00	25.98 / 0.7622 / 4.00
BasicVSR-uni-MinMax+FT	✗	4 / 4	28.88 / 0.8157 / 4.00	34.86 / 0.9167 / 4.00	25.80 / 0.7441 / 4.00	36.34 / 0.9397 / 4.00	34.93 / 0.9188 / 4.00	26.00 / 0.7581 / 4.00
BasicVSR-uni-Percentile+FT	✗	4 / 4	29.43 / 0.8366 / 4.00	35.49 / 0.9275 / 4.00	26.07 / 0.7681 / 4.00	37.32 / 0.9519 / 4.00	35.78 / 0.9326 / 4.00	26.50 / 0.7979 / 4.00
BasicVSR-uni-DBDC+FT	✗	4 / 4	28.80 / 0.8119 / 4.00	35.02 / 0.9187 / 4.00	25.90 / 0.7506 / 4.00	36.42 / 0.9404 / 4.00	34.50 / 0.9199 / 4.00	26.07 / 0.7613 / 4.00
BasicVSR-uni-Ours	✗	**4 / 4MP**	**29.57 / 0.8460 / 5.25**	**35.96 / 0.9323 / 5.12**	**26.32 / 0.7784 / 4.75**	**37.78 / 0.9550 / 5.00**	**36.22 / 0.9366 / 5.03**	**26.71 / 0.8067 / 4.90**
BasicVSR-uni-MinMax [20]	✓	4 / 4	28.45 / 0.7997 / 4.00	34.31 / 0.9088 / 4.00	25.56 / 0.7308 / 4.00	35.53 / 0.9274 / 4.00	34.55 / 0.9142 / 4.00	25.82 / 0.7531 / 4.00
BasicVSR-uni-Percentile [27]	✓	4 / 4	28.12 / 0.7984 / 4.00	33.94 / 0.9096 / 4.00	24.81 / 0.7030 / 4.00	34.76 / 0.9318 / 4.00	33.97 / 0.9111 / 4.00	24.87 / 0.7138 / 4.00
BasicVSR-uni-DBDC [38]	✓	4 / 4	28.36 / 0.7953 / 4.00	34.91 / 0.9182 / 4.00	25.86 / 0.7510 / 4.00	35.85 / 0.9327 / 4.00	34.74 / 0.9172 / 4.00	25.96 / 0.7608 / 4.00
BasicVSR-uni-MinMax+FT	✓	4 / 4	28.86 / 0.8135 / 4.00	34.70 / 0.9146 / 4.00	25.72 / 0.7384 / 4.00	36.33 / 0.9392 / 4.00	34.88 / 0.9182 / 4.00	26.14 / 0.7667 / 4.00
BasicVSR-uni-Percentile+FT	✓	4 / 4	28.35 / 0.8074 / 4.00	34.03 / 0.9107 / 4.00	24.92 / 0.7105 / 4.00	35.05 / 0.9341 / 4.00	34.06 / 0.9125 / 4.00	25.09 / 0.7242 / 4.00
BasicVSR-uni-DBDC+FT	✓	4 / 4	28.81 / 0.8119 / 4.00	34.81 / 0.9159 / 4.00	25.79 / 0.7432 / 4.00	35.85 / 0.9327 / 4.00	34.89 / 0.9192 / 4.00	25.98 / 0.7618 / 4.00
BasicVSR-uni-Ours	✓	**4 / 4MP**	**29.55 / 0.8434 / 5.25**	**35.75 / 0.9296 / 5.12**	**26.15 / 0.7674 / 4.75**	**37.29 / 0.9517 / 5.07**	**35.72 / 0.9319 / 5.09**	**26.53 / 0.7938 / 4.97**
BasicVSR-uni-MinMax [20]	✗	6 / 6	30.40 / 0.8649 / 6.00	36.79 / 0.9409 / 6.00	26.94 / 0.8121 / 6.00	38.90 / 0.9619 / 6.00	36.99 / 0.9442 / 6.00	27.36 / 0.8331 / 6.00
BasicVSR-uni-Percentile [27]	✗	6 / 6	30.36 / 0.8660 / 6.00	36.51 / 0.9405 / 6.00	26.75 / 0.8065 / 6.00	38.40 / 0.9619 / 6.00	36.72 / 0.9440 / 6.00	27.05 / 0.8252 / 6.00
BasicVSR-uni-DBDC [38]	✗	6 / 6	30.40 / 0.8653 / 6.00	36.79 / 0.9411 / 6.00	26.92 / 0.8127 / 6.00	38.91 / 0.9620 / 6.00	36.99 / 0.9444 / 6.00	27.37 / 0.8340 / 6.00
BasicVSR-uni-MinMax+FT	✗	6 / 6	30.35 / 0.8643 / 6.00	36.74 / 0.9404 / 6.00	26.89 / 0.8091 / 6.00	38.85 / 0.9614 / 6.00	36.93 / 0.9436 / 6.00	27.35 / 0.8314 / 6.00
BasicVSR-uni-Percentile+FT	✗	6 / 6	30.33 / 0.8654 / 6.00	36.50 / 0.9401 / 6.00	26.73 / 0.8052 / 6.00	38.37 / 0.9617 / 6.00	36.77 / 0.9444 / 6.00	27.06 / 0.8268 / 6.00
BasicVSR-uni-DBDC+FT	✗	6 / 6	30.39 / 0.8642 / 6.00	36.74 / 0.9401 / 6.00	26.91 / 0.8092 / 6.00	38.79 / 0.9611 / 6.00	36.95 / 0.9440 / 6.00	27.33 / 0.8321 / 6.00
BasicVSR-uni-Ours	✗	**6 / 6MP**	**30.40 / 0.8665 / 6.23**	**36.81 / 0.9413 / 7.05**	**26.91 / 0.8101 / 6.68**	**38.97 / 0.9633 / 7.07**	**37.11 / 0.9460 / 7.09**	**27.40 / 0.8372 / 6.97**
BasicVSR-uni-MinMax [20]	✓	6 / 6	30.29 / 0.8616 / 6.00	36.64 / 0.9393 / 6.00	26.89 / 0.8091 / 6.00	38.67 / 0.9601 / 6.00	36.80 / 0.9422 / 6.00	27.30 / 0.8302 / 6.00
BasicVSR-uni-Percentile [27]	✓	6 / 6	28.77 / 0.8237 / 6.00	34.42 / 0.9168 / 6.00	25.07 / 0.7222 / 6.00	35.19 / 0.9373 / 6.00	34.32 / 0.9163 / 6.00	25.04 / 0.7237 / 6.00
BasicVSR-uni-DBDC [38]	✓	6 / 6	30.35 / 0.8633 / 6.00	36.67 / 0.9398 / 6.00	26.88 / 0.8102 / 6.00	38.70 / 0.9604 / 6.00	36.81 / 0.9426 / 6.00	27.31 / 0.8315 / 6.00
BasicVSR-uni-MinMax+FT	✓	6 / 6	30.26 / 0.8612 / 6.00	36.60 / 0.9385 / 6.00	26.90 / 0.8075 / 6.00	38.65 / 0.9595 / 6.00	36.73 / 0.9413 / 6.00	27.30 / 0.8277 / 6.00
BasicVSR-uni-Percentile+FT	✓	6 / 6	29.01 / 0.8328 / 6.00	34.56 / 0.9195 / 6.00	25.23 / 0.7345 / 6.00	35.48 / 0.9402 / 6.00	34.46 / 0.9186 / 6.00	25.22 / 0.7352 / 6.00
BasicVSR-uni-DBDC+FT	✓	6 / 6	30.30 / 0.8621 / 6.00	36.61 / 0.9389 / 6.00	26.89 / 0.8070 / 6.00	38.63 / 0.9599 / 6.00	36.74 / 0.9416 / 6.00	27.28 / 0.8288 / 6.00
BasicVSR-uni-Ours	✓	**6 / 6MP**	**30.35 / 0.8646 / 7.25**	**36.63 / 0.9396 / 7.12**	**26.81 / 0.8060 / 6.75**	**38.70 / 0.9610 / 6.91**	**36.83 / 0.9430 / 6.94**	**27.29 / 0.8310 / 6.82**

Table 8: **Error bars evaluation with 4-bit fully quantized BasicVSR (PSNR / SSIM / FAB)**. All results are calculated on the Y-channel except REDS4 (RGB-channel). The random seeds are taken from 0, 1, 2, 3, 321 to generate calibration datasets.

Methods	seed	BI degradation			BD degradation		
		REDS4 [33]	Vimeo-90K-T [44]	Vid4 [30]	UDM10 [45]	Vimeo90K-T [44]	Vid4 [30]
BasicVSR [3]	-	31.42 / 0.8909 / 32.00	37.18 / 0.9450 / 32.00	27.24 / 0.8251 / 32.00	39.96 / 0.9694 / 32.00	37.53 / 0.9498 / 32.00	27.96 / 0.8553 / 32.00
Ours	0	30.09 / 0.8562 / 4.98	36.03 / 0.9335 / 5.05	26.29 / 0.7772 / 4.70	37.78 / 0.9554 / 4.95	35.93 / 0.9336 / 5.02	26.80 / 0.8040 / 4.95
	1	**30.26 / 0.8637 / 5.50**	**35.82 / 0.9311 / 5.03**	**26.29 / 0.7752 / 4.69**	**37.59 / 0.9536 / 4.78**	**35.95 / 0.9339 / 5.08**	**26.81 / 0.8025 / 4.98**
	2	29.90 / 0.8511 / 5.00	35.81 / 0.9311 / 5.11	26.22 / 0.7738 / 5.00	37.94 / 0.9563 / 5.05	36.02 / 0.9347 / 5.11	26.77 / 0.8032 / 4.95
	3	30.19 / 0.8618 / 5.50	35.93 / 0.9320 / 5.07	26.19 / 0.7712 / 4.68	37.85 / 0.9557 / 4.80	36.08 / 0.9356 / 5.06	26.93 / 0.8114 / 5.00
	321	30.09 / 0.8598 / 4.96	35.77 / 0.9299 / 5.07	25.84 / 0.7530 / 4.75	37.78 / 0.9546 / 5.00	35.97 / 0.9335 / 5.07	26.88 / 0.8056 / 5.00
	Mean	30.11 / 0.8578 / 5.19	35.87 / 0.9315 / 5.07	26.17 / 0.7701 / 4.76	37.79 / 0.9551 / 4.92	35.99 / 0.9343 / 5.07	26.84 / 0.8053 / 4.98
	Std	0.12 / 0.0038 / 0.24	0.10 / 0.0011 / 0.03	0.17 / 0.0088 / 0.12	0.12 / 0.0027 / 0.11	0.05 / 0.0019 / 0.03	0.06 / 0.0032 / 0.02

off distortion and perception, ours achieves reconstructions that are both quantitatively accurate and perceptually natural.

B.5 On-device Latency and Memory Consumption

We present CPU evaluation across three benchmark datasets, REDS4 [33], UDM10 [45], and Vid4 [30] running CPU-only inference on both Intel Xeon Gold 5218R (x86; Linux; 40 threads used) and Apple M1 Pro (ARM; macOS; 8 threads used). Our quantized models use 6-bit weights (64 levels) and dynamic 6-bit activation precision, executed via INT8 kernels due to current deployment constraints. Our method achieves consistent speedups across both platforms. On Intel, we observe 1.28–1.45$\times$ speedups over BasicVSR [3] and 1.05–1.14$\times$ over SSL [42] across all datasets.

Table 9: **Quantitative comparison (PSNR↑ / SSIM↑ / NIQE↓ / BRISQUE↓).** PSNR and SSIM are calculated on Y-channel except REDS4 [33] (RGB-channel). NIQE is computed on grayscale, and BRISQUE on RGB, following the PIQ [23] implementation.

Methods	FQ	Params(M)	GT	W / A	REDS4 [33] (BI)	Vid4 [30] (BI)	Vid4 [30] (BD)
BasicVSR [3]	-	4.9	✓	32 / 32	31.42 / 0.8909 / 20.4533 / 42.0603	27.24 / 0.8251 / 21.2263 / 46.6820	27.96 / 0.8553 / 20.9968 / 43.2148
SSL [42]	-	1.0	✓	32 / 32	31.06 / 0.8833 / 20.5275 / 42.5137	27.15 / 0.8208 / 21.2449 / 46.4442	27.56 / 0.8431 / 20.9479 / 44.2100
Ours	✗	**1.3**	✗	**6 / 6MP**	**31.26 / 0.8879 / 20.5416 / 42.6729**	**27.18 / 0.8215 / 21.1865 / 46.8981**	**27.84 / 0.8495 / 21.0283 / 44.0302**
Ours	✓	**1.0**	✗	**6 / 6MP**	**31.17 / 0.8849 / 20.5608 / 41.9935**	**27.05 / 0.8140 / 21.1727 / 45.7232**	**27.69 / 0.8405 / 21.0390 / 43.9466**
Ours	✗	**1.0**	✗	**4 / 4MP**	**30.34 / 0.8657 / 20.9095 / 46.9496**	**26.26 / 0.7764 / 21.0902 / 47.0901**	**27.02 / 0.8161 / 21.5588 / 44.8271**
Ours	✓	**0.7**	✗	**4 / 4MP**	**30.26 / 0.8637 / 20.7069 / 46.0018**	**26.29 / 0.7752 / 21.1959 / 45.3042**	**26.81 / 0.8025 / 21.2659 / 44.4966**
MinMax [20]	✓	0.7	✗	4 / 4	28.12 / 0.7896 / 20.5620 / 42.6086	26.03 / 0.7585 / 21.7752 / 44.2720	25.64 / 0.7412 / 21.6568 / 40.8614
Percentile [27]	✓	0.7	✗	4 / 4	27.78 / 0.7841 / 20.6565 / 46.4639	25.23 / 0.7305 / 21.0079 / 48.0859	25.08 / 0.7275 / 21.3866 / 45.0816
DBDC [38]	✓	0.7	✗	4 / 4	28.07 / 0.7817 / 20.6225 / 39.5703	26.24 / 0.7720 / 21.5619 / 44.7354	26.01 / 0.7581 / 21.6323 / 41.6941
MinMax+FT	✓	0.7	✗	4 / 4	29.21 / 0.8238 / 21.1095 / 45.4638	26.17 / 0.7601 / 21.9205 / 45.7476	26.11 / 0.7584 / 22.2404 / 43.5693
Percentile+FT [27]	✓	0.7	✗	4 / 4	28.30 / 0.8054 / 20.8440 / 46.1643	25.26 / 0.7314 / 21.3187 / 48.1076	25.26 / 0.7340 / 21.5298 / 45.3654
DBDC+FT	✓	0.7	✗	4 / 4	29.24 / 0.8232 / 20.9791 / 44.9058	26.31 / 0.7703 / 21.9100 / 46.8330	26.42 / 0.7764 / 22.0107 / 43.9439
BasicVSR-uni [3]	-	2.6	✓	32 / 32	30.54 / 0.8694 / 20.6602 / 44.0094	27.03 / 0.8163 / 21.0566 / 46.3993	27.54 / 0.8419 / 20.9163 / 43.7744
Ours	✗	**0.8**	✗	**6 / 6MP**	**30.40 / 0.8665 / 20.6781 / 44.1256**	**26.91 / 0.8101 / 21.2396 / 46.7119**	**27.40 / 0.8372 / 20.8535 / 44.3251**
Ours	✓	**0.55**	✗	**6 / 6MP**	**30.35 / 0.8646 / 20.7043 / 43.1907**	**26.81 / 0.8060 / 21.0356 / 46.2837**	**27.29 / 0.8310 / 20.8523 / 44.2879**
Ours	✗	**0.7**	✗	**4 / 4MP**	**29.57 / 0.8460 / 20.8091 / 45.0827**	**26.32 / 0.7784 / 21.7651 / 47.2370**	**26.71 / 0.8067 / 21.3894 / 45.7664**
Ours	✓	**0.4**	✗	**4 / 4MP**	**29.55 / 0.8434 / 20.7997 / 43.7369**	**26.15 / 0.7674 / 21.7846 / 46.7945**	**26.53 / 0.7938 / 21.2562 / 43.8501**
MinMax [20]	✓	0.4	✗	4 / 4	28.45 / 0.7997 / 21.1814 / 45.3900	25.56 / 0.7308 / 21.2691 / 43.8372	25.82 / 0.7531 / 21.3604 / 43.3835
Percentile [27]	✓	0.4	✗	4 / 4	28.12 / 0.7984 / 20.8874 / 47.1888	24.81 / 0.7030 / 21.2692 / 47.6955	24.87 / 0.7138 / 21.2414 / 46.6296
DBDC [38]	✓	0.4	✗	4 / 4	28.36 / 0.7953 / 21.1184 / 46.4919	25.86 / 0.7510 / 21.3411 / 45.2436	25.96 / 0.7608 / 21.1965 / 43.5195
MinMax+FT [20]	✓	0.4	✗	4 / 4	28.86 / 0.8135 / 21.2081 / 44.9757	25.72 / 0.7384 / 22.2928 / 45.5211	26.14 / 0.7667 / 21.7386 / 45.0558
Percentile+FT [27]	✓	0.4	✗	4 / 4	28.35 / 0.8074 / 20.9143 / 43.3349	24.92 / 0.7105 / 21.5808 / 48.3883	25.09 / 0.7242 / 21.3034 / 46.2203
DBDC+FT [38]	✓	0.4	✗	4 / 4	28.81 / 0.8119 / 20.9934 / 46.2085	25.79 / 0.7432 / 22.4006 / 46.1070	25.98 / 0.7618 / 21.4305 / 45.4257

The ARM platform shows even more significant gains, with speedups of 1.32–5.79×against BasicVSR [3] and 1.07–1.53× against SSL [42]. Notably, on UDM10 [45], our ARM implementation achieves a remarkable 5.79× speedup over BasicVSR [3] while reducing peak memory by approximately 17% compared to SSL [42]. The flow-warping (alignment) operator runs in FP32 with lightweight quantization overhead, preserving reconstruction quality while maintaining efficiency. We observe that the ARM platform demonstrates superior scalability for our method, where the unified memory architecture avoids NUMA penalties and cross-socket traffic overhead present in the dual-socket Intel system. These results demonstrate that our temporal awareness adaptation quantization approach delivers consistent acceleration and memory efficiency across diverse hardware architectures and datasets, while maintaining the reconstruction quality reported in the main paper.

Table 10: Inference latency and memory usage on x86 and ARM Platforms

Dataset	method	**Intel Xeon Gold 5218R**			**Apple M1 Pro**		
		Memory (MB)	Latency (s)	Speedup	Memory (MB)	Latency (s)	Speedup
REDS4 [33]	BasicVSR [3]	18823.56	2677.44	1.00×	10506.25	1293.22	1.00×
	SSL [42]	19047.57	2203.05	1.22×	10122.02	1009.10	1.28×
	Ours-FQ	**19068.67**	**1963.63**	**1.36×**	**10034.67**	**659.82**	**1.96×**
UDM10 [45]	BasicVSR [3]	14811.98	1252.88	1.00×	11346.16	3292.11	1.00×
	SSL [42]	15020.86	909.86	1.38×	10337.38	850.59	3.87×
	Ours-FQ	**15083.24**	**866.65**	**1.45×**	**8549.67**	**568.83**	**5.79×**
Vid4 [30]	BasicVSR [3]	3688.39	2095.47	1.00×	4470.97	422.10	1.00×
	SSL [42]	4318.05	1860.86	1.13×	4511.02	340.19	1.24×
	Ours-FQ	**3605.30**	**1631.11**	**1.28×**	**3435.47**	**318.79**	**1.32×**

C Overall Algorithm

Our Temporal Awareness Adaptation Quantization for VSR is summarized in Alg 1.

D Related Work

We provide a detailed comparison with AdaBM [16] to further clarify the distinctions in methodology and contributions. It is important to note that AdaBM was originally designed for image SR and introduced an efficient on-the-fly adaptive quantization framework that significantly reduces processing time. While we acknowledge the conceptual inspiration our work drew from AdaBM in terms of bit-level adaptation, we would like to emphasize that our framework is fundamentally redesigned to address the unique challenges of VSR, making it a substantially different and purpose-built solution. The core technical divergence lies in the incorporation of temporal modeling. Ad-

Algorithm 1 Temporal Awareness Adaptation Quantization for Video Super-Resolution

Input: Pretrained 32-bit VSR network $\mathcal{P}$ of K parameter-shared layers, calibration dataset $\mathcal{D}_{\text{cal}} = \left\{V_{LR}^{i}\right\}_{i=1}^{N}$, spatial percentile parameter p_{space}, temporal percentile parameter p_{temp}.
Output: Quantized VSR network $\mathcal{Q}$.

Calibration Phase:
Initialize weight clipping ranges $\{u_w^k\}_{k=1}^K$ using OMSE [8];
Initialize activation clipping ranges $\{l_a^k, u_a^k\}_{k=1}^K$ with the EMA smoothed minimum and maximum statistics [20] by running calibration dataset through the FP network;
for $i = 1$ to N **do**
    Calculate the video flow-gradient complexity metric C_{video} (see Eq.\ (9));
    Calculate the spatial sensitivity $\left\{s_{\text{space}}^k\right\}_{k=1}^K$ (see Eq.\ (10));
    Calculate the temporal sensitivity $\left\{s_{\text{temp}}^k\right\}_{k=1}^K$ (see Eq.\ (11));
end for
Initialize spatial thresholds l_{space} and u_{space} according to p_{space}-th and $(100-p_{\text{space}})$-th percentiles of the spatial sensitivity values $\left\{s_{\text{space}}^k\right\}_{k=1}^K$;
Initialize temporal thresholds l_{temp} and u_{temp} according to p_{temp}-th and $(100 - p_{\text{temp}})$-th percentiles of the temporal sensitivity values $\left\{s_{\text{temp}}^k\right\}_{k=1}^K$;
Initialize temporal shared layer bit adaptation factor $\left\{b_L^k\right\}_{k=1}^K$ (see Eq.\ (12));
Update activation clipping ranges $\{l_a^k, u_a^k\}_{k=1}^K$ using OMSE [8] depending on the temporal shared layer bit adaptation factor $\left\{b_L^k\right\}_{k=1}^K$ assigned per layer;
Fine-tuning Phase:
Update weight clipping ranges $\{u_w^k\}_{k=1}^K$ using Eq.\ (16);
Update activation clipping ranges $\{l_a^k, u_a^k\}_{k=1}^K$ using Eq.\ (16);
Update bit adaptive module parameters $l_{v2b}, u_{v2b}, \{b_L^k\}_{k=1}^K$ using Eq.\ (16);
Output the final quantized VSR model $\mathcal{Q}$.

aBM, targeting single-image SR, relies solely on spatial sensitivity for bit allocation. In contrast, our method explicitly models spatio-temporal variance through components like FG-VBA and TS-LBA. This temporal awareness is critical for maintaining coherence across frames in VSR but was absent in AdaBM. Furthermore, the training strategies differ significantly due to the inherently higher computational cost of processing video sequences compared to single images. AdaBM's requirement for ten full epochs of fine-tuning, while feasible for image SR models, becomes prohibitively expensive for VSR. Our method addresses this by employing an efficient three-stage progressive optimization schedule. This approach updates parameters in distinct stages, progressing through weight clipping, activation clipping, and finally bit adaptation modules optimization, which enables fast convergence in merely 3 epochs and demonstrates more stable optimization behavior, making it far more practical for VSR model quantization. The design of the loss function also reflects our focus on the temporal dimension. We extend beyond the spatial reconstruction loss used in image SR by introducing a temporal structure consistency loss. This addition is crucial for ensuring smooth transitions between frames and preventing artifacts like flickering, thereby enhancing the perceptual quality of the output video. Lastly, our approaches diverge in the handling of the bit space. Unlike AdaBM, which incorporates a bit-penalization loss to explicitly encourage lower bit-widths, we intentionally forgo such a constraint. We found that an explicit penalization could lead to optimization instability in VSR quantization. Instead, our method allows the optimal bit allocation to emerge more naturally through a data-driven optimization process, which we found to be more suitable for the complex spatio-temporal features in video data. In conclusion, while AdaBM provided valuable insights for adaptive quantization in image SR, our framework introduces critical innovations, from its temporal-aware modules and efficient training strategy to its specialized loss design and the removal of explicit bit-penalization, that collectively constitute a novel and substantially different solution tailored for the demands of VSR.

E More Visual Comparison

More visual results of the 4-bit FQ BasicVSR model are presented below. Our algorithm achieves the highest PSNR value and restores the most optimal results across all datasets. Moreover, after applying our designed FT strategy described in Sec 3.5, the performance of other benchmark algorithms is significantly improved, further validating the effectiveness of the proposed FT method.

F Limitation and Future Direction

Currently, we only investigate quantization for video super-resolution. It is better to generalize the quantized networks for other video restoration applications. Additionally, similar architectural considerations apply across various video processing tasks. These methodologies commonly eliminate Batch Normalization layers to maintain dynamic range adaptability while minimizing generative anomalies. Our ongoing research will include comprehensive empirical validation and investigate the broader applicability of this approach to pixel-to-pixel video tasks through systematic experimentation in subsequent studies.

G Code

We have provided code to reproduce the results in this work.

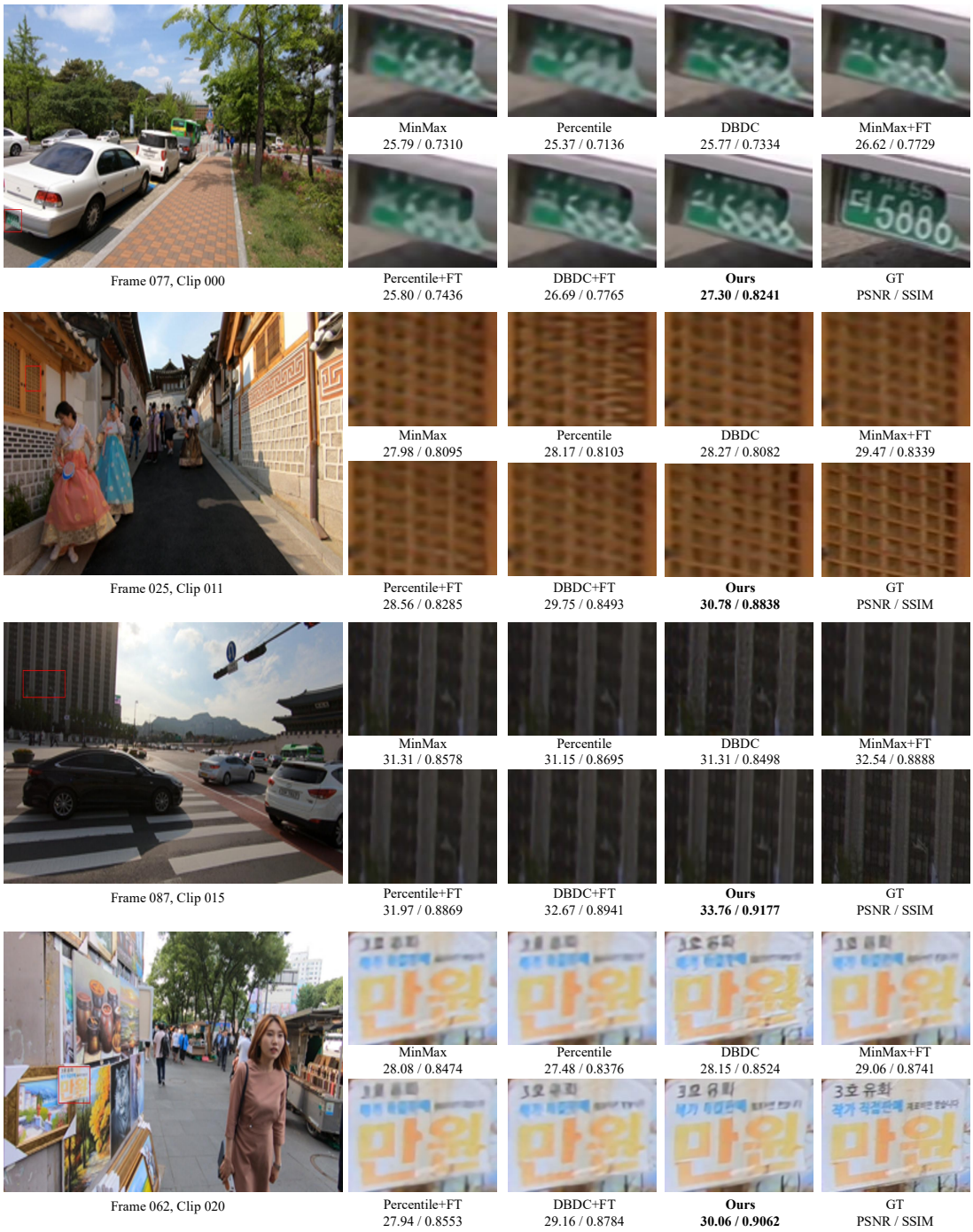


Figure 3: Qualitative comparison on REDS4 [33] (**Zoom-in for best view**)

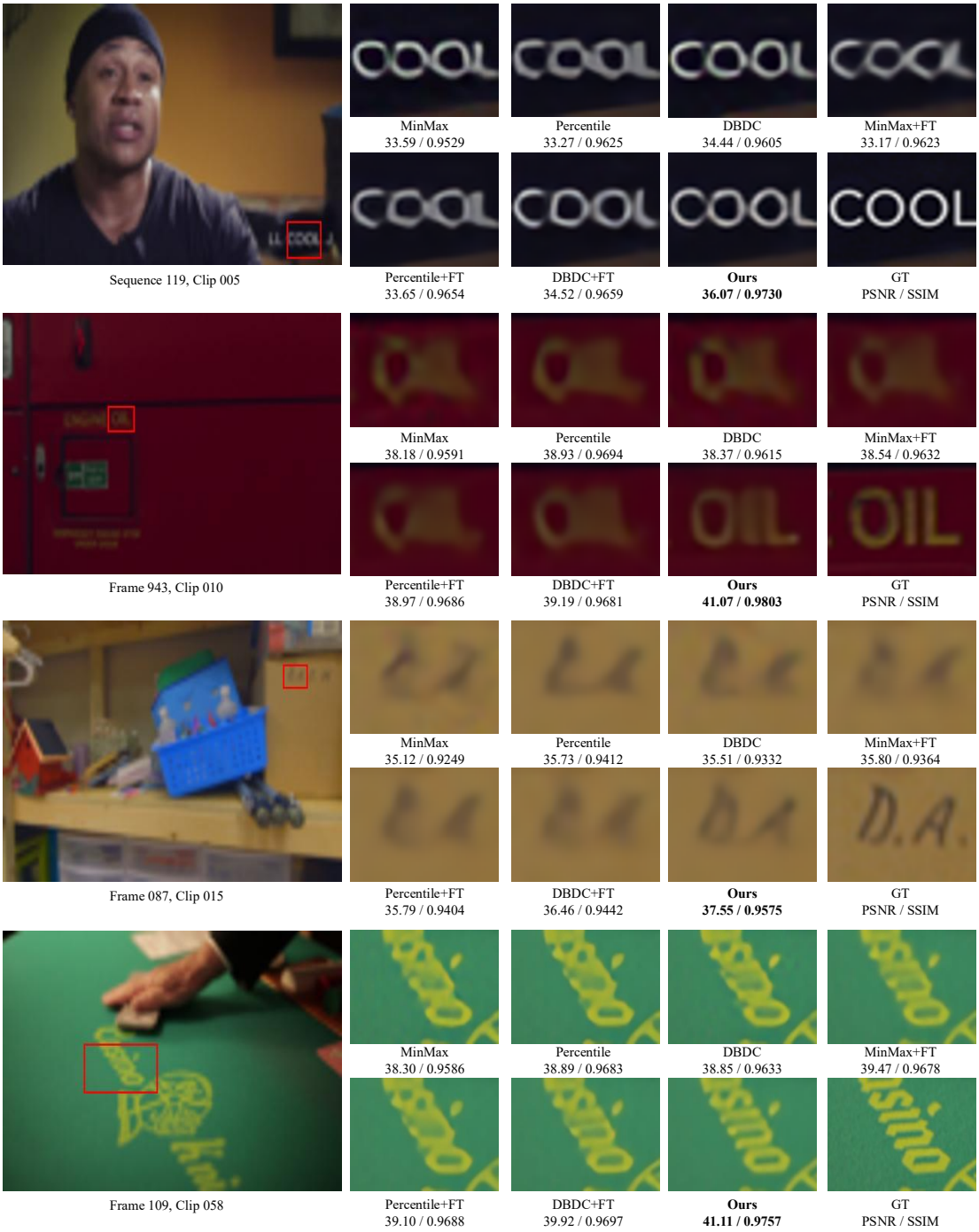


Figure 4: Qualitative comparison on Vimeo-90K [44] (**Zoom-in for best view**)

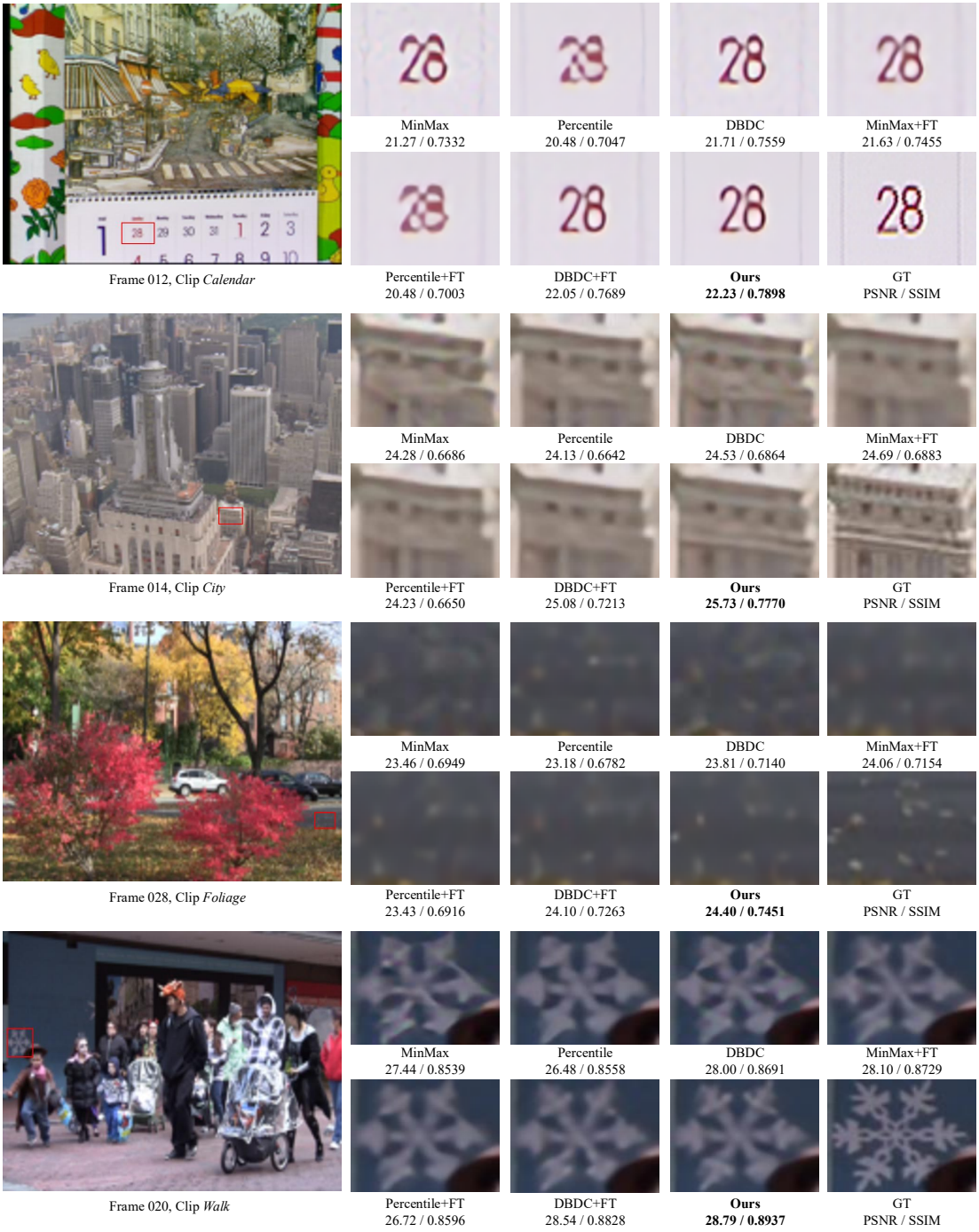


Figure 5: Qualitative comparison on Vid4 [30] (**Zoom-in for best view**)

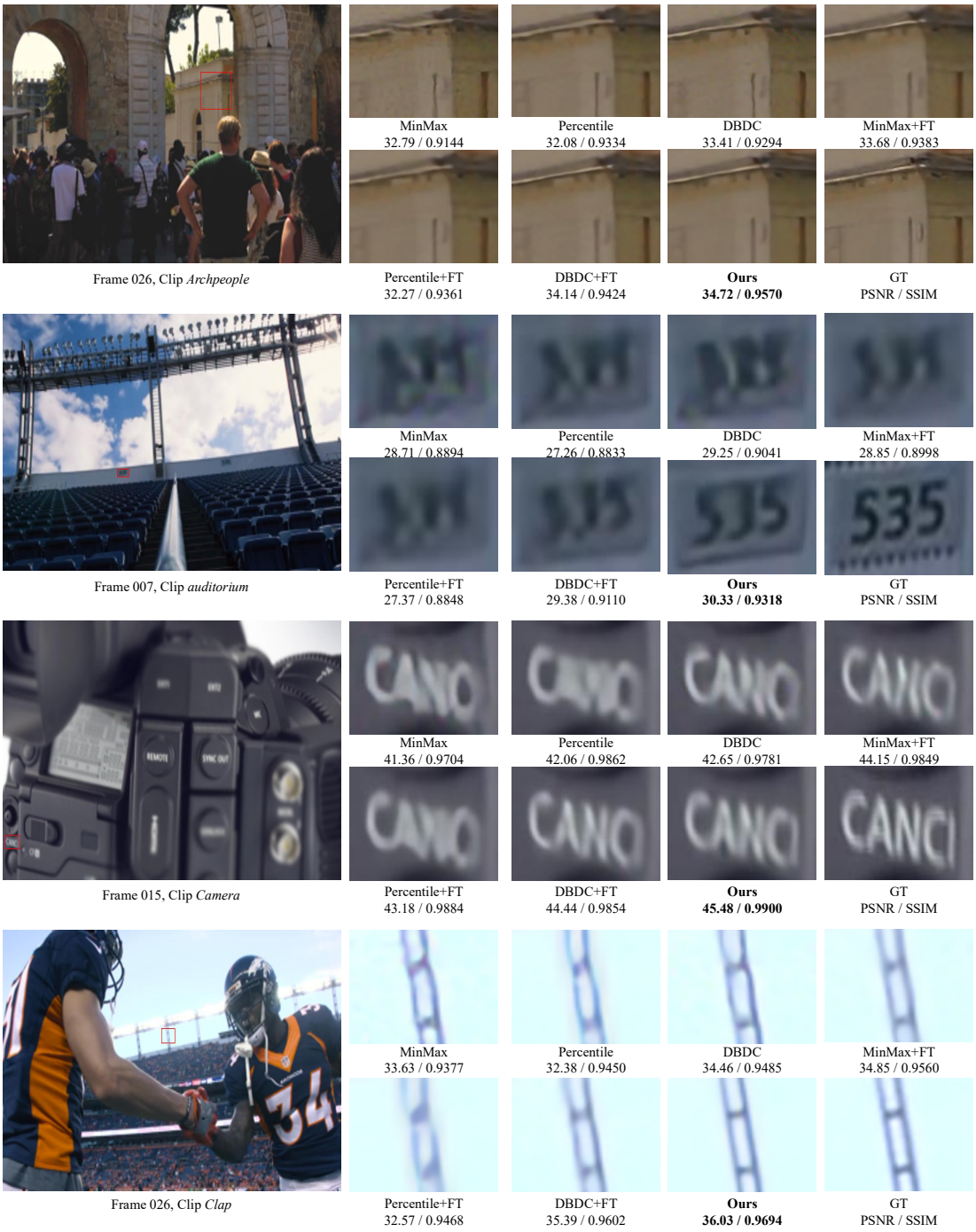


Figure 6: Qualitative comparison on UDM10 [45] (**Zoom-in for best view**)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix F.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our method does not involve theory assumptions or proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 4.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code will be released to QBasicVSR library.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 4.1 and Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Section 4.6 and Appendix B.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 4.5 and Appendix B.5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, we confirm.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There are no potential negative social impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There are no such problems in our task.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Section 4.1.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We use existing public benchmark datasets for experiments.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No such experiments or research are involved in our work.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No such experiments or research are involved in our work.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs are not used in any part of this research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.