NanoVSR: Towards Real-Time Video Super-Resolution on Edge Devices

Filip Pawlicki[1], Marcel Kańduła[2], Marcin Pucek[1], and Kamil Dobies[1]

[1] Gdańsk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Department of Computer Architecture, Gdańsk, Poland
{s198371, s197893, s197875}@student.pg.edu.pl
[2] Gdańsk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Department of Software Engineering, Gdańsk, Poland
s197677@student.pg.edu.pl

Abstract. Recent Video Super-Resolution (VSR) methods rely heavily on transformers and explicit optical flow, creating computational overhead and custom operations that hinder deployment on hardware accelerators like TensorRT. To address this, we introduce NanoVSR, a scalable, fully convolutional architecture designed for resource-constrained edge devices. Using structural reparameterization, NanoVSR collapses into standard convolutions during inference, ensuring seamless hardware compatibility and negligible runtime overhead. Furthermore, despite lacking explicit motion compensation, it maintains competitive restoration quality by implicitly learning spatio-temporal alignments through progressive training. Evaluated on the REDS4 benchmark, NanoVSR demonstrates an exceptional balance between accuracy and computational efficiency, significantly improving the trade-off for compact architectures. Our NanoVSR-644k baseline yields 28.64 dB PSNR on REDS4 while delivering 27.20 FPS on the NVIDIA Jetson Orin NX 16GB (25W), offering massive speed gains over heavier models. The scaled NanoVSR-1.7M variant reaches 29.15 dB with a throughput of 19.58 FPS, providing superior, edge-optimized upscaling.

Keywords: Video Super-Resolution · Edge AI · Structural Reparameterization

1 Introduction

Super-resolution (SR) is a fundamental computer-vision problem: reconstructing high-resolution (HR) detail from a low-resolution (LR) input that no longer contains it. While single image super-resolution focuses on recovering missing high-frequency details from a static image [8, 51], it is inherently limited by the absence of external information. Video super-resolution (VSR) transcends this limitation by introducing the temporal dimension, leveraging the rich, redundant information scattered across consecutive frames to recover details that are physically absent in a single frame. The fundamental mechanism of VSR involves the effective aggregation of this temporal data; however, because objects and

cameras move, simply stacking neighboring frames results in ghosting and blurring. Consequently, the core challenge of VSR lies in finding effective methods for motion compensation.

Leading VSR methods [3, 20, 44] have produced architectures capable of achieving impressive visual quality, successfully recovering details and maintaining temporal consistency across frames. To achieve this high standard of restoration, contemporary methods typically employ very deep neural networks that process video sequences using bidirectional propagation. Crucially, they rely on sophisticated alignment modules, utilizing dense optical flow estimation or complex attention mechanisms to maximize information extraction from multiple frames and ensure correct alignment before reconstruction.

However, this visual quality imposes a heavy resource demand. While manageable on high-end GPUs, these architectures require memory and compute far exceeding typical edge devices, so on constrained platforms the execution overhead exceeds real-time limits, rendering them impractical where efficient processing is as critical as restoration quality.

To address these problems, we propose NanoVSR, a lightweight VSR network optimized for edge platforms like the NVIDIA Jetson. Motivated by on-device AI, where real-time processing under strict resource constraints is paramount [45], our architecture draws inspiration from highly efficient designs [13,24,32,39]. We introduce a purely convolutional, bidirectional recurrent framework built upon structural reparameterization, allowing NanoVSR to train as a multi-branch network and collapse into an optimized stream of standard convolutions at inference. We further discard expensive optical flow and feature concatenation in favor of a direct additive propagation scheme. To compensate for the reduced capacity and lack of explicit alignment, we employ a progressive two-stage training method: the network is first pre-trained on short sequences for spatial feature extraction, then fine-tuned on extended sequences, forcing the recurrent structure to implicitly learn residual motion and long-term dependencies.

Our main contributions are as follows:

- We propose NanoVSR, a highly efficient, purely convolutional VSR architecture explicitly designed for edge devices, eliminating the reliance on memory-intensive explicit alignment and custom CUDA operations.
- We introduce a direct additive propagation scheme for bidirectional recurrent VSR that avoids channel concatenation, reducing memory bandwidth and enabling the full network—including temporal propagation—to collapse into a plain convolution stream via structural reparameterization.
- We conduct extensive experiments across standard benchmarks (REDS4, Vid4, Vimeo-90K-T) and provide hardware-specific profiling on two NVIDIA Jetson Orin NX configurations (8GB/15W and 16GB/25W), demonstrating that NanoVSR delivers a highly competitive balance of restoration quality and inference throughput compared to existing baselines.

2 Related Work

In this section, we review the literature relevant to our proposed NanoVSR architecture. We first discuss the datasets commonly used for training and evaluating video super-resolution models, highlighting the shift towards more complex motion profiles. Then, we provide an overview of existing VSR methodologies, ranging from traditional sliding-window networks to modern, computationally intensive transformers. Finally, we situate our work within the context of efficient network design, specifically structural reparameterization.

2.1 Video Super-Resolution Datasets

The evaluation of VSR algorithms depends on datasets reflecting real-world degradation and complex temporal dynamics. Early works relied on Vid4 [21] and SPMCS [40], which lack the diversity and motion complexity needed to test modern architectures. Larger benchmarks have since become standard: Vimeo-90K [48] provides over 90,000 clips serving as the primary training ground for most VSR networks, while the REDS dataset [26] from the NTIRE 2019 Challenge offers sequences with heavy motion blur and large displacements. We benchmark NanoVSR on three widely adopted test sets: Vid4 for historical comparisons, the Vimeo-90K test set (Vimeo-90K-T) for diverse motion, and the challenging REDS4 split (clips 000, 011, 015, 020 from REDS dataset) for highly dynamic scenes.

2.2 Existing VSR Methods

VSR architectures have transitioned from traditional sliding-window CNNs to recurrent frameworks and, more recently, heavy attention-based transformers.

Sliding-Window and Alignment-Based CNNs. Early VSR models processed a local temporal window to reconstruct a central frame. Methods like VESPCN [1] and TOFlow [48] relied on explicit optical flow. To mitigate flow estimation errors, TDAN [41] and EDVR [44] adapted Deformable Convolutions (DCNs) [6, 52]. However, computing dense optical flow or DCN offsets for every frame imposes a severe computational bottleneck, rendering these methods unsuitable for real-time edge deployment.

Recurrent Architectures. Early models like FRVSR [31] and RLSP [10] relied on unidirectional propagation, limiting their access to future temporal context. This was largely overcome by BasicVSR [3] and its variants [4], which use bidirectional propagation and flow-based alignment. While more efficient, their reliance on sequential flow estimation (*e.g.*, SPyNet [30]) leads to suboptimal inference speeds on constrained hardware.

Transformer-Based VSR. Benefiting from highly effective spatio-temporal modeling, transformer-based architectures have established leading performance on video restoration benchmarks. For instance, VRT [19] uses mutual attention for feature alignment, while RVRT [20] employs guided deformable attention. Although RVRT sets the state-of-the-art (SOTA), its sheer parameter count and

quadratic attention complexity make it computationally prohibitive and strictly confined to high-end GPUs.

Efficient VSR and Structural Reparameterization. Developing fast VSR architectures has attracted increasing attention, with efficiency-oriented designs spanning reparameterizable architectures (RepNet-VSR [47]), neural architecture search (EVSRNet [22]), and sliding-window recurrence (SWRN [18]), alongside scaled-down baselines (*e.g.*, EDVR-M [44]) and compact recurrent designs (*e.g.*, RSDN [16], MobileRNN [14]). However, most of these methods still depend on operations or model capacities that hinder strict real-time deployment on embedded accelerators. To address this gap, we draw inspiration from structural reparameterization [7], training the network as an expressive multi-branch model while executing it as a single chain of plain convolutions at inference.

3 Methodology

In this section, we detail the structural design and optimization strategy of NanoVSR, our lightweight video super-resolution network. We first introduce the overall bidirectional recurrent architecture, designed to maximize inference throughput by entirely bypassing explicit motion compensation and custom CUDA operations (Sec. 3.1). Next, we describe the structural reparameterization technique [7] used to collapse a multi-branch training topology into a minimalist, single-branch inference model (Sec. 3.2). Finally, we describe the two-stage training protocol that enables our network to implicitly learn robust spatio-temporal alignments and achieve competitive restoration quality (Sec. 3.3).

3.1 Architecture Overview

Unlike heavyweight Video Super-Resolution (VSR) models that rely on explicit alignment modules (optical flow or deformable convolution) [3,31,44,48] or computationally expensive transformer-based attention mechanisms [2, 9, 19, 20, 34], NanoVSR is designed as a purely convolutional, bidirectional recurrent network constructed exclusively from standard tensor operations. By explicitly avoiding custom CUDA modules like deformable convolutions and memory-intensive warping functions, NanoVSR ensures native ONNX [29] compatibility. This maximizes data throughput on edge devices by fully utilizing deep fusion optimizations provided by hardware accelerators (*e.g.*, TensorRT [28]).

The overall pipeline of NanoVSR is illustrated in Fig. 1. Given a sequence of low-resolution (LR) input frames $x \in \mathbb{R}^{T\times 3\times H\times W}$, where T denotes the sequence length, the network processes the sequence through three main stages: shallow feature extraction, bidirectional temporal propagation, and high-resolution (HR) reconstruction.

Shallow Feature Extraction. Initially, the temporal dimension is folded into the batch dimension to process all frames independently and efficiently. A single feature extraction block maps the RGB input into a higher-dimensional latent space:

$$f_i = \mathcal{H}_{ext}(x_i), \quad \forall i \in \{1, \dots, T\}, \tag{1}$$

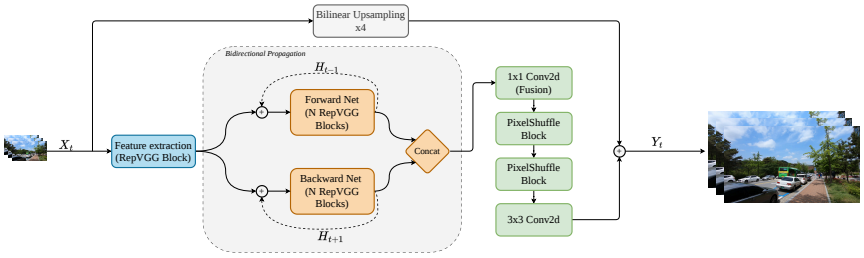

Fig. 1: Detailed schematic of the NanoVSR architecture. LR input frames are embedded into a latent space and passed through a bidirectional recurrent structure built with reparameterizable blocks (see Fig. 2). Crucially, to minimize memory bandwidth, temporal propagation relies on element-wise addition instead of standard feature concatenation. The refined features are bottlenecked and upsampled, with a global bilinear residual added to reconstruct the final HR sequence.

where $f_i \in \mathbb{R}^{C\times H\times W}$ represents the extracted features for the i-th frame, C is the number of feature channels, and $\mathcal{H}_{ext}$ denotes our multi-branch extraction block (which is detailed in Sec. 3.2). After this operation, the features are reshaped back to restore the temporal dimension.

Bidirectional Temporal Propagation. To aggregate temporal context without explicit motion compensation, we employ a bidirectional recurrent structure. In standard recurrent architectures [3,10,16,31], hidden states are typically concatenated with input features. However, concatenation doubles the channel dimension, leading to a substantial increase in computational cost and heavily burdening memory bandwidth during subsequent convolutions. To mitigate this, NanoVSR employs a direct element-wise additive propagation scheme. The forward and backward hidden states, denoted as $h_i^{\rightarrow}$ and $h_i^{\leftarrow}$ respectively, are updated sequentially as follows:

$$h_i^{\rightarrow} = \mathcal{H}_{forward}\left(f_i + h_{i-1}^{\rightarrow}\right), \tag{2}$$

$$h_i^{\leftarrow} = \mathcal{H}_{backward}\left(f_i + h_{i+1}^{\leftarrow}\right), \tag{3}$$

where both $\mathcal{H}_{forward}$ and $\mathcal{H}_{backward}$ consist of a sequence of structural blocks configured with the LeakyReLU activation function. This additive formulation forces the network to implicitly learn residual motion representations, significantly reducing the memory footprint during inference while maintaining robust temporal coherence. The hidden states $h_0^{\rightarrow}$ and $h_{T+1}^{\leftarrow}$ are initialized as zero tensors.

Fusion and Reconstruction. For each time step i, the forward and backward hidden states are channel-wise concatenated and fused via a 1×1 convolution to bottleneck the channel dimension back to C:

$$f_i^{fused} = \mathcal{H}_{fusion}\left([h_i^{\rightarrow}, h_i^{\leftarrow}]\right), \tag{4}$$

where $[\cdot,\cdot]$ denotes the channel-wise concatenation operator. The fused features are subsequently upsampled through a cascaded sub-pixel convolution module

(PixelShuffle) [35] consisting of two progressive $2\times$ spatial expansion stages. These stages are separated by PReLU [11] activations, before a final 3×3 convolution projects the refined representations back into the 3-channel RGB domain.

To ease the learning process, preserve absolute spatial scale, and allow the core network to focus strictly on high-frequency detail synthesis, we employ a global residual connection. The final HR output $\hat{y}_i$ is generated by adding the network's processed residual output to the upsampled LR input. To ensure maximum hardware efficiency and minimize latency, we substitute traditional bicubic upsampling with highly optimized bilinear interpolation for this base projection:

$$\hat{y}_i = \mathcal{H}_{up}(f_i^{fused}) + \text{Bilinear}_{\uparrow 4}(x_i). \tag{5}$$

3.2 Structural Reparameterization for Inference

On edge accelerators, throughput is limited less by the raw parameter count than by how frequently the device must access memory and by the many small, separately-launched kernels a branched graph produces [24]. While multi-branch architectures provide strong representational capacity and stable gradient flow during training [12, 38], they fragment memory access and introduce severe CUDA kernel launch overheads during inference. To resolve this disparity, NanoVSR is constructed utilizing structural reparameterization, allowing us to completely decouple the training topology from the deployment architecture.

During the training phase, every fundamental building block operates as a multi-branch topology. The input features are processed via three parallel pathways: a 3×3 convolution, a 1×1 convolution, and an identity mapping, each immediately followed by a Batch Normalization (BN) layer [15].

Upon completing the training phase, the model is prepared for hardware deployment. Because convolution and batch normalization act as linear and affine transformations during inference, we apply a mathematically equivalent transformation to collapse the entire block. Following the standard reparameterization formulation, the parallel branches and their respective absorbed BN statistics are zero-padded, spatially aligned, and linearly fused into a single set of weights and biases.

Consequently, the complex multi-branch topology completely vanishes during edge deployment, collapsing into a single, highly optimized 3×3 convolution, as visually depicted in Fig. 2. The inference forward pass is executed as a continuous stream of plain convolutions, completely eliminating memory fragmentation and making it perfectly suited for deep fusion optimizations provided by accelerators like TensorRT [28].

3.3 Two-Stage Training Strategy

Training an implicit-alignment VSR network from scratch on long temporal sequences is notoriously unstable. To prevent gradient instability and ensure convergence, we adopt a two-phase progressive curriculum strategy [43].

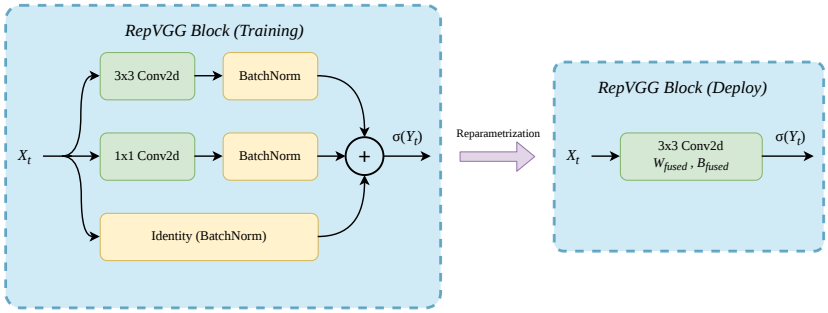

Fig. 2: Diagram of the structural reparameterization process [7]. During training, the block utilizes a multi-branch topology (3×3, 1×1, and identity pathways) to ensure robust feature extraction. For deployment, Batch Normalization statistics are first absorbed into their respective kernels. Finally, the parallel branches are mathematically fused into a single, dense 3×3 convolution, eliminating memory fragmentation and significantly accelerating inference.

Phase 1: Short-Sequence Pre-training. For the initial 50,000 iterations, the network is trained on 7-frame sequences from the Vimeo-90K dataset. This phase focuses on establishing robust spatial feature extraction and short-term temporal fusion. The limited motion in these clips prevents early gradient instability, providing a solid initialization for the structural blocks.

Phase 2: Long-Sequence Fine-tuning. Subsequently, we transition the training corpus to the REDS dataset and expand the temporal window to 30 continuous frames for the remaining 100,000 iterations. We employ a sliding window approach to sample these sequences. This phase is crucial for developing the network's long-term recurrence capabilities, allowing the bidirectional hidden states to propagate information across complex motion trajectories.

Optimization Details. The network is trained for 150,000 iterations using a global batch size of 12 and spatial patches of size 256×256. We optimize the model end-to-end using the Charbonnier penalty [5] ($\epsilon = 1 \times 10^{-6}$) and the Adam optimizer [17] ($\beta_1 = 0.9, \beta_2 = 0.99$). A single Cosine Annealing scheduler [23] governs the learning rate across both phases, decaying from 3×10^{-4} to 1×10^{-7}. To maintain high throughput and memory efficiency, we utilize Automatic Mixed Precision (AMP) [25] with the BFloat16 format and gradient clipping with a maximum L_2 norm of 0.5 throughout the training process. Data augmentation includes geometric transformations (flips, rotations, temporal reversal) and the CutBlur strategy [49].

4 Experiments

This section evaluates NanoVSR across several standard benchmarks. Our analysis focuses on two primary dimensions: reconstruction quality and hardware-specific efficiency on edge devices. Restoration quality is comprehensively quanti-

fied using fidelity (PSNR/SSIM [46]), perceptual (LPIPS [50]/VMAF [27]), and temporal (ST-RRED [36]) metrics.

4.1 Experimental Setup

Models are trained with the progressive curriculum of Sec. 3.3 on Vimeo-90K and REDS. The four sequences (000, 011, 015, 020) comprising the REDS4 benchmark were strictly excluded from the training set, and performance is evaluated on three standard benchmarks: REDS4, Vid4, and Vimeo-90K-T. For REDS4 we report PSNR and SSIM in the RGB color space; for Vid4 and Vimeo-90K-T, metrics are computed on the luminance (Y) channel. All models are evaluated at a $4\times$ upsampling factor.

Quantitative and inference benchmarks are conducted on a single NVIDIA H100 GPU, with the inference latency of all baselines re-evaluated on our hardware at FP32 precision for a fair comparison. For edge deployment, we use two configurations of the NVIDIA Jetson Orin NX: an 8GB variant at 15W TDP and a 16GB variant at 25W TDP. All models are exported via ONNX and compiled with NVIDIA TensorRT 10.3.0.30 [28]. To maximize edge performance, we employ FP16 precision and a chunk-based execution pattern with a temporal window of 15 frames ($T = 15$), ensuring efficient bidirectional context aggregation within the device's memory and power constraints.

4.2 Quantitative Results

We compare NanoVSR against state-of-the-art and efficiency-oriented VSR baselines, considering restoration quality and inference latency for a 180×320 input. As detailed in Tab. 1, NanoVSR offers a significant speed advantage over existing methods. While heavyweight transformer models like RVRT [20] achieve the highest PSNR, they incur substantial inference delay (47.867 ms), making them unsuitable for real-time edge applications. In contrast, our ultra-lightweight NanoVSR-226k operates at just 1.910 ms per frame, and our flagship NanoVSR-644k reaches 28.64 dB on REDS4 at only 2.982 ms, approaching significantly heavier architectures. Scaling up, NanoVSR-1.7M attains 29.15 dB on REDS4 while maintaining a low runtime of 4.268 ms per frame. Qualitatively (Figs. 3 and 4), NanoVSR recovers structural and organic textures well, scaling gracefully with model capacity. While the finest details are best recovered by the heaviest models, all methods achieve only moderate quality on alphanumeric details in motion-heavy scenes. We further contextualize NanoVSR against efficient single-image and recurrent baselines. The SISR model SPAN [42], applied frame-by-frame, reaches 25.43 dB on Vid4 at 5.820 ms per frame. NanoVSR brackets this operating point: NanoVSR-226k trades 0.17 dB on Vid4 (25.26 vs. 25.43 dB) for roughly three times lower latency (1.910 ms), whereas NanoVSR-1.7M surpasses SPAN on all three benchmarks (*e.g.*, 26.44 vs. 25.43 dB on Vid4, 29.15 vs. 28.48 dB on REDS4) while remaining faster (4.268 ms). This confirms that temporal aggregation yields a markedly better speed-accuracy trade-off than per-frame restoration. To provide a verifiable comparison against ultra-fast edge

Table 1: Quantitative comparison on Vid4 [21], REDS4 [26], and Vimeo-90K-T [48]. Metrics are PSNR (dB) / SSIM; runtime is ms/frame for a 180×320 input on an NVIDIA H100 in FP32, re-measured on our hardware for all methods. For baselines with separate released models, REDS4 uses the REDS-trained model and Vid4/Vimeo-90K-T the Vimeo-90K-trained model; single-model baselines use one model throughout. Restoration metrics for BasicVSR [3] and RVRT [20] are taken from their original papers, and those for EDVR-M [44], IconVSR [3] and EDVR [44] from the comparison tables therein; SPAN [42] and MobileRNN [14], lacking public scores under this protocol, were evaluated by us. * SPAN is a single-image model applied frame-by-frame. † MobileRNN is omitted from REDS4, as its official weights were trained on the full REDS set (including the REDS4 clips).

Model	**Params**	**Runtime (ms)**	**Vid4** (Y-channel)	**REDS4** (RGB)	**Vimeo-90K-T** (Y-channel)
NanoVSR-226k	226k	1.910	25.26 / 0.7252	28.23 / 0.8057	34.31 / 0.9130
NanoVSR-644k	644k	2.982	26.05 / 0.7761	28.64 / 0.8215	35.00 / 0.9226
NanoVSR-1.7M	1.7M	4.268	26.44 / 0.7964	29.15 / 0.8364	35.49 / 0.9294
NanoVSR-5.4M	5.4M	8.547	26.76 / 0.8089	29.73 / 0.8526	35.85 / 0.9335
EDVR-M	3.3M	27.343	27.10 / 0.8186	30.53 / 0.8699	37.09 / 0.9446
BasicVSR	6.2M	15.160	27.24 / 0.8251	31.42 / 0.8909	37.18 / 0.9450
IconVSR	8.7M	28.881	27.39 / 0.8279	31.67 / 0.8948	37.47 / 0.9476
RVRT	10.8M	47.867	27.99 / 0.8462	32.75 / 0.9113	38.15 / 0.9527
EDVR	20.6M	73.994	27.35 / 0.8264	31.09 / 0.8800	37.61 / 0.9489
SPAN*	498k	5.820	25.43 / 0.7357	28.48 / 0.8109	35.06 / 0.9213
MobileRNN†	474k	5.676	25.42 / 0.7369	— / —	34.45 / 0.9156

architectures, we also evaluate MobileRNN [14], the official baseline model from the Mobile AI 2022 VSR Challenge. For a fair comparison, we report MobileRNN only on benchmarks unseen during its training; we therefore omit REDS4, as the official weights were trained on the full REDS dataset. On the remaining benchmarks, its 474k-parameter variant attains 25.42 dB on Vid4 and 34.45 dB on Vimeo-90K-T, but its reliance on standard recurrent operations incurs a noticeable inference latency of 5.676 ms per frame. At a comparable operating point, NanoVSR-226k reaches 25.26 dB on Vid4 (-0.16 dB vs. MobileRNN) at nearly three times the speed (1.910 ms), while our flagship NanoVSR-644k outperforms MobileRNN on both valid benchmarks ($+0.63$ dB on Vid4, $+0.55$ dB on Vimeo-90K-T) while remaining nearly twice as fast. We also considered SWRN [18], a sliding-window recurrent network notable for its very low latency, which we reproduced at 0.66 ms per frame, although under our evaluation protocol we obtained markedly lower PSNR than its reported 27.92 dB on the REDS validation split. To avoid reporting numbers that may not reflect the method's intended setup, we omit it from Tab. 1.

Beyond fidelity-oriented PSNR/SSIM, perceptual and temporal behavior is essential for video restoration. We therefore report LPIPS, VMAF, and ST-RRED on Vid4 and REDS4 (Tab. 2). On REDS4, NanoVSR-644k surpasses the frame-by-frame SISR baseline SPAN in perceptual quality (LPIPS 0.2546 vs. 0.2757) and temporal consistency (ST-RRED 6.64×10^{-5} vs. 8.91×10^{-5}), as it

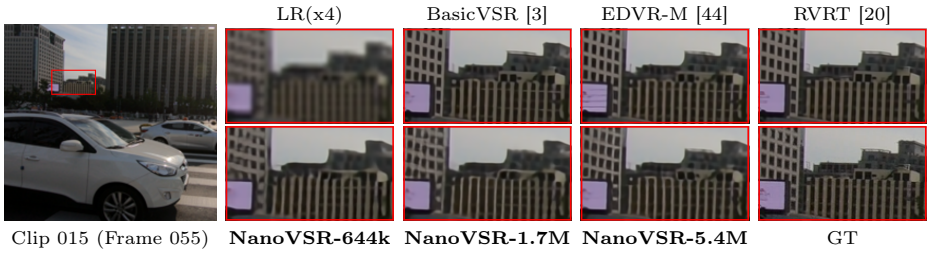

Fig. 3: Qualitative comparison on the REDS4 [26] dataset (Clip 015). Reconstruction of the regular window structures improves visibly with model capacity across the NanoVSR variants; while RVRT recovers the sharpest geometry, the larger NanoVSR models render the pattern with reasonable fidelity. Left: full frame; right: magnified patches.

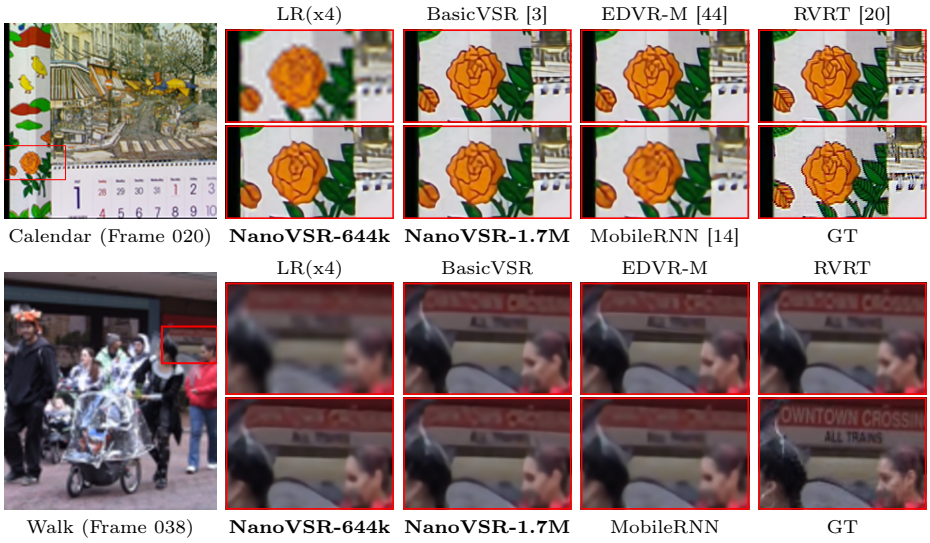

Fig. 4: Qualitative comparison on the Vid4 [21] dataset. Top (Calendar sequence): NanoVSR reconstructs the organic floral texture well, preserving its overall structure as model capacity scales; the finest detail is still best recovered by heavier models. Bottom (Walk sequence): on the text-heavy background under motion, all methods, lightweight and state-of-the-art alike, achieve only moderate reconstruction quality, reflecting the persistent difficulty of alphanumeric detail in motion-heavy content. For each row, left: full frame; right: magnified patches.

does on Vid4 (LPIPS 0.2889 vs. 0.3288; ST-RRED 1.93×10^{-4} vs. 3.31×10^{-4}), while the heavyweight RVRT achieves the strongest perceptual and temporal scores overall. VMAF and ST-RRED improve monotonically with model capacity on both datasets and remain stable across our suite, indicating that the

Table 2: Perceptual and temporal metrics on Vid4 [21] and REDS4 [26], reported as LPIPS [50] (perceptual quality) together with VMAF [27] and ST-RRED [36] (spatio-temporal consistency), under the same evaluation setup as Tab. 1. As these three metrics are not reported by the baseline works, they were computed by us on the generated outputs for all methods; the accompanying PSNR is reproduced from Tab. 1.

	Vid4				REDS4			
Model	**PSNR ↑**	**LPIPS ↓**	**VMAF ↑**	**ST-RRED ↓**	**PSNR ↑**	**LPIPS ↓**	**VMAF ↑**	**ST-RRED ↓**
SPAN [42]	25.43	0.3288	71.35	3.31×10^{-4}	28.48	0.2757	88.95	8.91×10^{-5}
BasicVSR [3]	27.24	0.2401	82.32	7.10×10^{-5}	31.42	0.1655	98.30	8.13×10^{-6}
RVRT [20]	27.99	0.2246	85.65	6.05×10^{-5}	32.75	0.1312	99.49	4.45×10^{-6}
NanoVSR-226k	25.26	0.3283	65.79	4.25×10^{-4}	28.23	0.2695	85.15	9.27×10^{-5}
NanoVSR-644k	26.05	0.2889	73.28	1.93×10^{-4}	28.64	0.2546	88.37	6.64×10^{-5}
NanoVSR-1.7M	26.44	0.2644	76.99	1.28×10^{-4}	29.15	0.2374	91.26	4.91×10^{-5}

implicit alignment scheme preserves temporal coherence without introducing flickering, despite the absence of explicit motion compensation.

4.3 Efficiency Analysis

To assess real-world applicability on embedded hardware, we conduct an efficiency analysis across two configurations of the NVIDIA Jetson Orin NX: an 8GB variant restricted to a 15W TDP and a 16GB variant operating at a 25W TDP. We measure inference latency and throughput (FPS) for a temporal window of $T = 15$ frames at two input resolutions, 180×320 and 270×480, with all models compiled via the TensorRT 10.3 execution engine. Since both devices were operated at their maximum performance profiles and memory consumption remained within hardware limits due to static engine optimization, we prioritize throughput as the primary metric for edge-device viability.

As summarized in Tab. 3, heavyweight architectures like BasicVSR [3] fail to reach real-time performance, achieving 8.04 FPS for a 180×320 input on the Orin NX (16GB) while demanding significantly more resources. In contrast, NanoVSR-226k meets stringent streaming constraints, delivering 43.86 FPS on 180×320 and 19.55 FPS on 270×480 inputs, while our flagship NanoVSR-644k sustains 27.20 FPS on 180×320. Even the scaled NanoVSR-1.7M maintains competitive edge performance within the strict hardware limits of edge computing.

4.4 Model Scaling Analysis

To evaluate the scalability of the proposed architecture, we investigate the relationship between the number of parameters and the restoration quality on the challenging REDS4 dataset. As illustrated in Fig. 5, we scale the network capacity from an ultra-lightweight 22k parameters up to 9.6M parameters by progressively adjusting the number of temporal propagation blocks and internal feature channels.

Table 3: Efficiency scalability across NVIDIA Jetson Orin NX configurations. We compare average inference latency per frame (ms) and throughput (FPS) between the 8GB variant (15W TDP) and the 16GB variant (25W TDP). All models are compiled via TensorRT with a temporal window of $T = 15$.

		Orin NX (8GB)		**Orin NX (16GB)**	
Model	**Resolution**	**Runtime (ms)**	**FPS**	**Runtime (ms)**	**FPS**
NanoVSR-226k	180×320	42.46	23.55	22.80	43.86
	270×480	95.15	10.51	51.14	19.55
NanoVSR-644k	180×320	62.04	16.12	36.77	27.20
	270×480	139.03	7.19	77.99	12.82
NanoVSR-1.7M	180×320	86.46	11.57	51.06	19.58
	270×480	196.40	5.09	112.77	8.87
NanoVSR-5.4M	180×320	190.71	5.24	115.50	8.66
	270×480	427.95	2.34	263.82	3.79
BasicVSR [3]	180×320	229.28	4.36	124.37	8.04
	270×480	528.38	1.89	289.13	3.46

Initial Capacity Gains. The performance trajectory exhibits steep initial gains in the low-parameter regime. Scaling the network from the NanoVSR-22k variant to the NanoVSR-644k baseline yields a substantial improvement of 1.02 dB (from 27.62 dB to 28.64 dB). As the architectural capacity increases beyond the 1M parameter mark, the network continues to improve visual fidelity steadily, reaching 29.73 dB for the 5.4M variant.

Performance Saturation. Nevertheless, expanding the architecture from the NanoVSR-5.4M to the NanoVSR-9.6M variant yields diminishing returns. Despite a near two-fold increase in parameter count, it provides only a marginal PSNR improvement of 0.17 dB, effectively plateauing at 29.90 dB. This saturation point indicates that while the NanoVSR topology can easily scale to achieve higher-quality reconstructions, our baseline (NanoVSR-644k) strikes the most practical balance between computational efficiency and restoration quality, making it the definitive choice for edge deployment.

4.5 Ablation Studies

To explore the effectiveness of different architectural design choices, we conduct ablation studies on the REDS4, Vid4, and Vimeo-90K-T datasets. Due to the substantial computational cost of video training, all ablations were conducted using our highly efficient NanoVSR-226k variant. We empirically observed that these design insights and relative performance improvements transfer consistently to our larger flagship model, NanoVSR-644k.

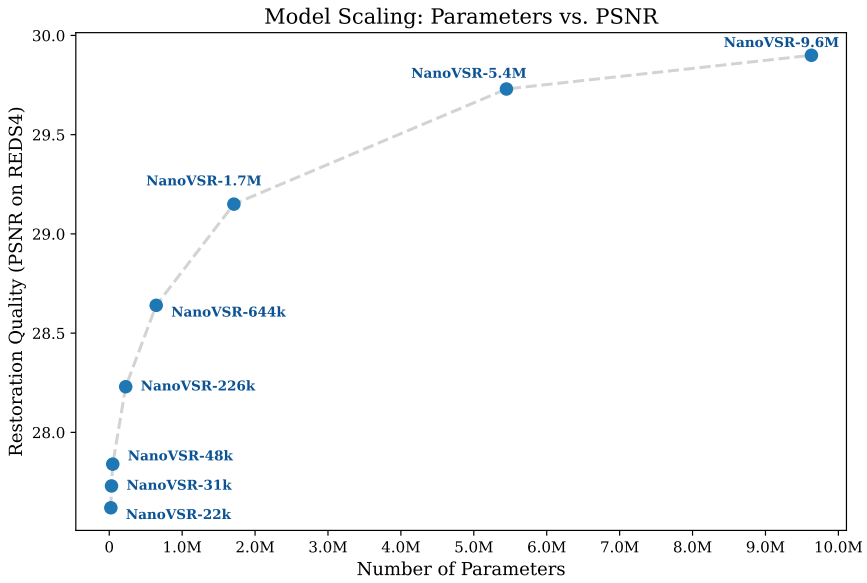

Fig. 5: Model scaling analysis on the REDS4 [26] dataset. The plot illustrates the trade-off between the number of parameters and restoration quality (PSNR). While scaling up the architecture consistently improves performance, the curve demonstrates diminishing returns beyond 5.4M parameters.

The impact of structural reparameterization. In NanoVSR, structural reparameterization decouples the training topology from the inference architecture. As shown in Tab. 4, the baseline NanoVSR-226k and its non-fused counterpart NanoVSR-226k-NOFUSE achieve mathematically identical restoration quality, yet the fusion reduces the parameter count from 245k to 226k and nearly halves the inference time (from 3.471 ms to 1.910 ms). Conversely, NanoVSR-226k-SINGLE, trained from scratch as a strict single-branch architecture, drops by 0.1 dB PSNR on REDS4. Although it yields a marginally higher SSIM on Vid4, this reflects a smoothing bias toward lower-frequency predictions under constrained capacity rather than improved fidelity: the single-branch topology lacks the representational power to reconstruct the fine detail captured by our multi-branch reference. This confirms that the rich multi-branch training phase is essential for learning robust representations.

The impact of curriculum pre-training. Our two-stage progressive training scheme is evaluated by comparing the NanoVSR-226k model against NanoVSR-226k-NOPRET, which is trained exclusively on REDS, omitting the Vimeo-90K phase. While both models reach an identical PSNR on REDS4, the pre-trained variant naturally performs better on Vimeo-90K-T due to prior exposure. For zero-shot generalization on the external Vid4 dataset, skipping this initial phase leads to a PSNR drop from 25.26 dB to 25.17 dB. Although NanoVSR-

Table 4: Comprehensive ablation study on the NanoVSR architecture. We evaluate the impact of structural reparameterization (-NOFUSE, -SINGLE), curriculum pretraining (-NOPRET), explicit motion alignment (-SPYNET [30]), as well as the strictly unidirectional inference mode (-ONEWAY). Runtime is measured in milliseconds per frame for a 180×320 input on an NVIDIA H100 GPU. Restoration quality is reported as PSNR (dB) / SSIM.

Model	**Params**	**Runtime (ms)**	**Vid4 [21]** (Y-channel)	**REDS4 [26]** (RGB)	**Vimeo-90K-T [48]** (Y-channel)
NanoVSR-226k (Reference)	226k	1.910	25.26 / 0.7252	28.23 / 0.8057	34.31 / 0.9130
NanoVSR-226k-NOFUSE	245k	3.471	25.26 / 0.7252	28.23 / 0.8057	34.31 / 0.9130
NanoVSR-226k-SINGLE	226k	1.885	25.25 / 0.7296	28.13 / 0.8027	34.25 / 0.9134
NanoVSR-226k-NOPRET	226k	1.887	25.17 / 0.7271	28.23 / 0.8054	34.21 / 0.9130
NanoVSR-226k-SPYNET	1.7M	3.953	25.95 / 0.7701	29.44 / 0.8460	34.85 / 0.9220
NanoVSR-226k-ONEWAY	151k	1.470	25.00 / 0.7107	28.04 / 0.8005	33.98 / 0.9086

226k-NOPRET yields a marginally higher SSIM here (0.7271 vs. 0.7252), the overall metrics demonstrate that initial exposure on diverse sequences improves overall generalization and restoration quality.

The impact of explicit alignment. We analyze the architectural trade-off of omitting explicit motion compensation by introducing NanoVSR-226k-SPYNET. This variant shares the identical core architecture and is trained under the same progressive curriculum, with the sole addition of an integrated SPyNet [30] module. As indicated in Tab. 4, explicit alignment significantly boosts the restoration fidelity, yielding a substantial gain of 1.21 dB over NanoVSR-226k on REDS4. However, this performance surge incurs severe computational penalties. The total parameter count inflates to over 1.7M (of which approximately 1.5M belong strictly to the SPyNet estimator), and the inference time more than doubles to 3.953 ms per frame. Given the strict power and inference speed constraints inherent to edge devices, the NanoVSR architecture intentionally foregoes explicit optical flow.

The impact of the bidirectional window. To quantify the contribution of bidirectional temporal aggregation, we evaluate a strictly unidirectional variant, NanoVSR-226k-ONEWAY, which removes the 15-frame look-ahead. Discarding future context eliminates the buffering delay and reduces the parameter count to 151k (1.470 ms per frame), but lowers the Vid4 PSNR by 0.26 dB (from 25.26 dB to 25.00 dB). This explicitly quantifies the quality cost of zero-latency streaming and confirms that the bidirectional window provides a meaningful restoration benefit, while the architecture remains directly deployable in latency-critical scenarios where this trade-off is acceptable.

5 Limitations and Societal Impact

While NanoVSR achieves a competitive efficiency-quality balance, it inherits limitations of implicit-alignment architectures. The absence of explicit optical flow can hinder reconstruction of high-frequency textures under extreme, non-rigid local motion. Moreover, bidirectional propagation requires a temporal look-

ahead window ($T = 15$), introducing a buffering delay that makes the model unsuitable for strict frame-by-frame use without degrading to a unidirectional mode. Extending our evaluation to higher input resolutions remains future work.

From a societal perspective, our focus on efficiency broadens access to high-quality video processing and reduces the environmental footprint of large-scale deployment [33, 37], while also extending the lifecycle of legacy low-resolution hardware. However, such enhancement is inherently dual-use and could facilitate non-consensual surveillance. As super-resolution synthesizes plausible details rather than recovering true signals, outputs must be treated as enhancements rather than identifiable evidence. We therefore advocate deploying VSR within bounded, consent-driven domains under robust ethical guidelines against unverified monitoring.

6 Conclusion

In this work, we presented NanoVSR, a fully convolutional architecture for video super-resolution on resource-limited edge devices. By replacing optical flow and attention with a progressive curriculum and structural reparameterization, our bidirectional recurrent network collapses into a stream of standard convolutions at inference, ensuring seamless TensorRT compatibility and low latency. Across REDS4, Vid4, and Vimeo-90K-T, NanoVSR redefines the accuracy-efficiency trade-off for compact models: NanoVSR-226k delivers 43.86 FPS on an NVIDIA Jetson Orin NX (25W), the flagship NanoVSR-644k sustains 27.20 FPS at higher quality, and NanoVSR-1.7M reaches 29.15 dB on REDS4 at 4.268 ms per frame on an H100. We hope this encourages further research into sustainable, globally accessible video enhancement.

Acknowledgements

The calculations were carried out using the computers of the Centre of Informatics Tricity Academic Supercomputer & Network.

The research was supported in part by project "Cloud Artificial Intelligence Service Engineering (CAISE) platform to create universal and smart services for various application areas", No. KPOD.05.10-IW.10-0005/24, as part of the European IPCEI-CIS program, financed by NRRP (National Recovery and Resilience Plan) funds.

References

1. Caballero, J., Ledig, C., Aitken, A., Acosta, A., Totz, J., Wang, Z., Shi, W.: Real-time video super-resolution with spatio-temporal networks and motion compensation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2848–2857 (2017). https://doi.org/10.1109/CVPR.2017.304
2. Cao, J., Li, Y., Zhang, K., Van Gool, L.: Video super-resolution transformer (2023), arXiv preprint arXiv:2106.06847

3. Chan, K.C., Wang, X., Yu, K., Dong, C., Loy, C.C.: BasicVSR: The search for essential components in video super-resolution and beyond. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4945–4954 (2021). https://doi.org/10.1109/CVPR46437.2021.00491
4. Chan, K.C., Zhou, S., Xu, X., Loy, C.C.: BasicVSR++: Improving video super-resolution with enhanced propagation and alignment. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5962–5971 (2022). https://doi.org/10.1109/CVPR52688.2022.00588
5. Charbonnier, P., Blanc-Feraud, L., Aubert, G., Barlaud, M.: Two deterministic half-quadratic regularization algorithms for computed imaging. In: Proceedings of 1st International Conference on Image Processing. vol. 2, pp. 168–172 (1994). https://doi.org/10.1109/ICIP.1994.413553
6. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 764–773 (2017). https://doi.org/10.1109/ICCV.2017.89
7. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: RepVGG: Making VGG-style convnets great again. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13728–13737 (2021). https://doi.org/10.1109/CVPR46437.2021.01352
8. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(2), 295–307 (2016). https://doi.org/10.1109/TPAMI.2015.2439281
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021)
10. Fuoli, D., Gu, S., Timofte, R.: Efficient video super-resolution through recurrent latent space propagation. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3476–3485 (2019). https://doi.org/10.1109/ICCVW.2019.00431
11. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 1026–1034 (2015). https://doi.org/10.1109/ICCV.2015.123
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
13. Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., Le, Q.: Searching for MobileNetV3. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1314–1324 (2019). https://doi.org/10.1109/ICCV.2019.00140
14. Ignatov, A., Timofte, R., Chiang, C.M., Kuo, H.K., Xu, Y.S., Lee, M.Y., Lu, A., Cheng, C.M., Chen, C.C., Yong, J.Y., Shuai, H.H., Cheng, W.H., Jia, Z., Xu, T., Zhang, Y., Bao, L., Sun, H., Zhang, D., Gao, S., Liu, S., Wu, B., Zhang, X., Zheng, C., Lu, K., Wang, N., Sun, X., Wu, H., Liu, X., Zhang, W., Yan, C., Du, H., Zheng, Q., Wang, Q., Chen, W., Duan, R., Sun, M., Zhu, D., Chen, G., Cho, H., Kim, S., Yue, S., Li, C., Zhuge, Z., Chen, W., Wang, W., Zhou, Y., Cai, X., Cai, H., Xu, K., Liu, L., Cheng, Z., Lian, W., Lian, W.: Power Efficient Video Super-Resolution on Mobile NPUs with Deep Learning, Mobile AI & AIM 2022 Challenge: Report. In: Computer Vision – ECCV 2022 Workshops: Tel Aviv, Israel,

October 23–27, 2022, Proceedings, Part III. pp. 130–152. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-25066-8_6
15. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 448–456. PMLR, Lille, France (07–09 Jul 2015)
16. Isobe, T., Jia, X., Gu, S., Li, S., Wang, S., Tian, Q.: Video super-resolution with recurrent structure-detail network. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 645–660. Springer International Publishing, Cham (2020)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
18. Lian, W., Lian, W.: Sliding window recurrent network for efficient video super-resolution. In: Karlinsky, L., Michaeli, T., Nishino, K. (eds.) Computer Vision – ECCV 2022 Workshops. pp. 591–601. Springer Nature Switzerland, Cham (2023)
19. Liang, J., Cao, J., Fan, Y., Zhang, K., Ranjan, R., Li, Y., Timofte, R., Van Gool, L.: VRT: A video restoration transformer. IEEE Transactions on Image Processing **33**, 2171–2182 (2024). https://doi.org/10.1109/TIP.2024.3372454
20. Liang, J., Fan, Y., Xiang, X., Ranjan, R., Ilg, E., Green, S., Cao, J., Zhang, K., Timofte, R., Van Gool, L.: Recurrent video restoration transformer with guided deformable attention. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2022)
21. Liu, C., Sun, D.: On bayesian adaptive video super resolution. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(2), 346–360 (2014). https://doi.org/10.1109/TPAMI.2013.127
22. Liu, S., Zheng, C.y., Lu, K., Gao, S., Wang, N., Wang, B., Zhang, D., Zhang, X., Xu, T.: EVSRNet: Efficient video super-resolution with neural architecture search. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 2480–2485 (2021)
23. Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (2017)
24. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In: Computer Vision – ECCV 2018. pp. 122–138. Springer International Publishing, Cham (2018)
25. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., Wu, H.: Mixed precision training. In: International Conference on Learning Representations (2018)
26. Nah, S., Baik, S., Hong, S., Moon, G., Son, S., Timofte, R., Lee, K.M.: NTIRE 2019 challenge on video deblurring and super-resolution: Dataset and study. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1996–2005 (2019). https://doi.org/10.1109/CVPRW.2019.00251
27. Netflix, Inc.: VMAF - video multi-method assessment fusion (2016), https://github.com/Netflix/vmaf, last accessed 2026-06-25
28. NVIDIA Corporation: NVIDIA TensorRT (2017), https://developer.nvidia.com/tensorrt, last accessed 2026-06-25
29. ONNX Contributors: ONNX: Open neural network exchange (2017), https://github.com/onnx/onnx, last accessed 2026-06-25
30. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2720–2729 (2017). https://doi.org/10.1109/CVPR.2017.291

31. Sajjadi, M.S.M., Vemulapalli, R., Brown, M.: Frame-recurrent video super-resolution. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6626–6634 (2018). https://doi.org/10.1109/CVPR.2018.00693
32. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4510–4520. IEEE Computer Society, Los Alamitos, CA, USA (Jun 2018). https://doi.org/10.1109/CVPR.2018.00474
33. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green AI. Commun. ACM **63**(12), 54–63 (Nov 2020). https://doi.org/10.1145/3381831
34. Shi, S., Gu, J., Xie, L., Wang, X., Yang, Y., Dong, C.: Rethinking alignment in video super-resolution transformers. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2022)
35. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1874–1883 (2016). https://doi.org/10.1109/CVPR.2016.207
36. Soundararajan, R., Bovik, A.C.: Video quality assessment by reduced reference spatio-temporal entropic differencing. IEEE Transactions on Circuits and Systems for Video Technology **23**(4), 684–694 (2013). https://doi.org/10.1109/TCSVT.2012.2214933
37. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: Korhonen, A., Traum, D., Màrquez, L. (eds.) Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3645–3650. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1355
38. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (2015). https://doi.org/10.1109/CVPR.2015.7298594
39. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019)
40. Tao, X., Gao, H., Liao, R., Wang, J., Jia, J.: Detail-revealing deep video super-resolution. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 4482–4490 (2017). https://doi.org/10.1109/ICCV.2017.479
41. Tian, Y., Zhang, Y., Fu, Y., Xu, C.: TDAN: Temporally-deformable alignment network for video super-resolution. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3357–3366 (2020). https://doi.org/10.1109/CVPR42600.2020.00342
42. Wan, C., Yu, H., Li, Z., Chen, Y., Zou, Y., Liu, Y., Yin, X., Zuo, K.: Swift parameter-free attention network for efficient super-resolution. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 6246–6256 (2024). https://doi.org/10.1109/CVPRW63382.2024.00628
43. Wang, X., Chen, Y., Zhu, W.: A survey on curriculum learning. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(9), 4555–4576 (2022). https://doi.org/10.1109/TPAMI.2021.3069908

44. Wang, X., Chan, K.C., Yu, K., Dong, C., Loy, C.C.: EDVR: Video restoration with enhanced deformable convolutional networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1954–1963 (2019). https://doi.org/10.1109/CVPRW.2019.00247
45. Wang, X., Tang, Z., Guo, J., Meng, T., Wang, C., Wang, T., Jia, W.: Empowering edge intelligence: A comprehensive survey on on-device ai models. ACM Comput. Surv. **57**(9) (Apr 2025). https://doi.org/10.1145/3724420
46. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004). https://doi.org/10.1109/TIP.2003.819861
47. Wu, B., Zhang, D., Liu, S., Gao, S., Zheng, C., Wang, N.: RepNet-VSR: Reparameterizable architecture for high-fidelity video super-resolution (2025), arXiv preprint arXiv:2504.15649
48. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision **127**(8), 1106–1125 (2019). https://doi.org/10.1007/s11263-018-01144-2
49. Yoo, J., Ahn, N., Sohn, K.A.: Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8372–8381 (2020). https://doi.org/10.1109/CVPR42600.2020.00840
50. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 586–595 (2018). https://doi.org/10.1109/CVPR.2018.00068
51. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. pp. 294–310. Springer International Publishing, Cham (2018)
52. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets v2: More deformable, better results. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9300–9308 (2019). https://doi.org/10.1109/CVPR.2019.00953