GameSR: Real-Time Super-Resolution for Interactive Gaming

Haseeb Ur Rehman, Shervin Shirmohammadi, *Fellow, IEEE*, Ihab Amer, *Senior Member, IEEE*, and Mohamed Hefeeda, *Senior Member, IEEE*

***Abstract*—High-resolution gaming demands significant computational resources, with challenges further amplified by bandwidth and latency constraints in cloud gaming. Existing upscalers, such as NVIDIA DLSS and AMD FSR, reduce rendering costs but require engine integration, making them unavailable for most titles and inapplicable to cloud gaming, where the client receives only compressed video and has no access to engine data. We present GameSR, a lightweight, engine-independent super-resolution model that operates directly on encoded game frames. The architecture of GameSR combines reparameterized convolutional blocks, PixelUnshuffle, and a lightweight ConvLSTM to deliver real-time upscaling with high perceptual quality. Extensive objective and subjective evaluations on popular games, such as *Counter-Strike 2*, *Overwatch 2*, *FC24* and *Team Fortress 2*, show that GameSR reduces cloud gaming bandwidth usage by 35–56% while meeting target perceptual qualities, achieves real-time performance of up to 240 FPS for $2\times$ scaling to 1080p ($\sim$4 ms/frame) and remains within real-time budgets for 4K upscaling (7–15 ms/frame), substantially outperforms existing super-resolution models in the literature, and achieves competitive no-reference perceptual quality compared to DLSS and FSR *without* accessing rendering engine data structures or modifying game source code, making GameSR a practical, engine-agnostic solution for upscaling both modern and legacy games in cloud gaming with no additional development effort.**

***Index Terms*—Cloud Gaming, Super-Resolution, Neural Upscaling.**

I. Introduction

Gaming is the world's largest entertainment industry, with revenues exceeding $200 billion in 2024 and projected to reach $290 billion by 2030 [1]. High-resolution, high-frame-rate gaming (e.g., 4K at 60–120 fps) is immersive but computationally demanding. High-end GPUs like the RTX 3080 Ti draw $\sim$350 W under load [2], and full gaming PCs can reach 325–380 W at ultra settings [3], consuming up to 1,400 kWh/year [4], highlighting the substantial hardware and energy costs of premium gaming.

An emerging alternative to this hardware-intensive model is cloud gaming, where games are rendered on remote servers and streamed to lightweight clients. While this shifts the computational burden away from players, it introduces substantial bandwidth and latency challenges. Unlike video streaming services such as Netflix, which stream 1080p content at around 5 Mbps (3 GB/hr) [5], platforms like NVIDIA GeForce NOW demand at least 28 Mbps for 1080p (12.6 GB/hr) [6], due to fast motion, complex animations, and latency-sensitive compression profiles (e.g., small GOPs and no B-frames). Moreover, gaming is highly interactive, requiring round-trip response within milliseconds to preserve player performance and Quality of Experience (QoE). Prior studies show first-person shooter games tolerate up to 80 ms end-to-end latency [7], while every additional 100 ms can reduce third-person game performance by 25% [8]. Latency arises from client input, server rendering/encoding, and network delay; the latter alone can consume up to 80% of the total budget [9].

A common way to reduce rendering costs is to lower spatial resolution and then upscale; however, naive upscaling degrades visual quality. Hardware vendors have therefore introduced content-aware solutions such as NVIDIA DLSS [10], AMD FSR [11], and Intel XeSS [12]. While effective, these upscalers require game engine integration and access to depth maps, motion vectors, and other internal data structures, with additional vendor restrictions (e.g., DLSS on NVIDIA hardware only). Research models like RenderSR [13], ExtraSS [14], Mob-FGSR [15], and Neural Supersampling [16] follow the same tightly coupled approach. As a result, support remains limited to a small subset of modern titles, leaving legacy engines and forward-rendered pipelines unable to adopt these upscalers.

In contrast, a large body of work on super-resolution for *general* images and videos (e.g., [17]–[21]) can operate directly on rendered frames without requiring game-engine integration. While these models achieve good upscaling quality, they are typically too slow for interactive use, with inference times far exceeding real-time budgets, as confirmed by our experiments in Section IV. As such, these models remain unsuitable as a general-purpose upscaling solution for gaming.

The goal of this paper is to introduce a video game upscaler that reduces computing cost while preserving high visual fidelity, and that operates independently of the game engine without requiring source code. Achieving this is challenging: strict latency constraints leave little tolerance for extra processing, most industrial upscalers rely on engine-level data (e.g., motion vectors, depth), and any solution must be lightweight enough to coexist with rendering, encoding, and networking in real-time. Even minor overheads risk stutter or added input-to-display delay, as modern pipelines already push frame budgets to the limit, often disabling effects like motion blur or ambient occlusion at higher frame rates. Thus, an effective upscaler must be engine-agnostic, efficient, and carefully integrated to deliver perceptual gains without breaking interactivity. We present evaluations in Section IV.

The main contributions of this paper are as follows.

- We propose GameSR (§III-B), a lightweight, engine-agnostic neural super-resolution model that operates directly on rendered frames **without requiring access to game source code or game engine data structures**,

Haseeb Ur Rehman and Shervin Shirmohammadi are with the University of Ottawa, Ottawa, ON, Canada (e-mail: haseebfazal@gmail.com; sshirmoh@uottawa.ca).

Ihab Amer is with the American University of Sharjah, Sharjah, United Arab Emirates (e-mail: iamer@aus.edu).

Mohamed Hefeeda is with Simon Fraser University, Burnaby, BC, Canada (e-mail: mhefeeda@sfu.ca).

making it readily deployable for both modern and legacy games in cloud and local gaming.
- We deploy GameSR in a real-time WebRTC cloud-gaming testbed and show that streaming at lower resolutions and upscaling with GameSR achieves the same perceptual quality at **35–56% lower bitrate**, while also reducing end-to-end latency by $\sim$19% (§IV-E).
- GameSR matches SOTA quality while running **30–60$\times$ faster** than CNN baselines and nearly **500$\times$ faster** than SwinIR, with up to an **order-of-magnitude smaller** size and memory (§IV-B).
- We demonstrate that GameSR achieves **competitive no-reference perceptual quality** compared to industrial upscalers such as DLSS and FSR, despite operating entirely without motion vectors, depth buffers, or any engine-level data (§IV-B).

The subjective study in this paper has been approved by the Research Ethics Board of our institution.

II. Background and Related Work

Characteristics of Video Game Content: Unlike camera-captured video, game frames are *synthetic* outputs of a graphics engine, and prior work shows they exhibit visual statistics and streaming behavior that differ from natural video [22], [23]. More importantly for super-resolution, a game is a largely *closed visual world*: it reuses a finite library of meshes, textures, shaders, and HUD elements across maps and stages, while viewpoints and motions are drawn from a small set of camera rigs (e.g., first-person) and a fixed action space (e.g., move, jump, shoot, ability casts). This bounded, repetitive structure makes high-frequency details (edges, text, UI glyphs, weapon geometry) highly recurring and therefore learnable from representative gameplay, as also reflected by datasets that use short sequences to capture a title's characteristic visual behavior [24]. The growing importance of competitive gaming and esports streaming further underscores the need for low-latency, high-quality video delivery [25]. GameSR leverages this property by training per title: the model learns game-specific priors that generalize across unseen matches of the same game, enabling real-time reconstruction quality that is difficult to achieve with generic SR models trained for open-world natural video.

Stand-alone Gaming and Upscalers: Most games run locally on PCs or consoles, where detailed textures, fast motion, and ray tracing demand powerful GPUs. To reduce load, super-resolution (SR) methods render at lower resolutions and upscale—far cheaper than full-resolution rendering.

Industry solutions include DLSS [26], FSR [11], XeSS [12], and Unreal Engine's built-in Temporal Super Resolution (TSR) [27]. TSR addresses many practical challenges of engine-integrated upscaling, including translucency, motion blur, ghosting, flickering, and antialiasing, with multi-platform support across PC and console hardware. More recently, NVIDIA introduced DLSS 3.5 Ray Reconstruction [28], which replaces hand-tuned denoisers with an AI-trained neural network to reconstruct higher-quality pixels in ray-traced scenes. While effective, all these solutions require integration into game source code to access engine data (motion vectors, depth, color), complicating deployment and limiting applicability.

Academic work has also advanced real-time upsampling: Neural Supersampling [16] leverages depth and motion vectors but suffers from ghosting; Li et al. [29] separate lighting and material components for temporal stability, an approach conceptually related to DLSS Ray Reconstruction [28]; and ExtraSS [14] combines supersampling with G-buffer–guided frame extrapolation. Like industrial solutions, these approaches rely heavily on engine data structures.

Limitations of Engine-Integrated Upscalers: The reliance on engine data structures limits existing upscalers to a narrow set of modern titles, particularly excluding legacy games with active communities. For instance, Team Fortress 2 (2007, Source engine) has not been ported to Source 2 and cannot expose the motion vectors, depth buffers, or temporal anti-aliasing required by DLSS 2/3 and FSR 2/3 [30], [31]. Similar restrictions apply to forward-rendered games like Counter-Strike 2, whose pipeline lacks the temporal data upscalers depend on.

As a result, DLSS, FSR, and XeSS support only a limited subset of titles with explicit developer integration [32], [33]. While Steam hosts over 86,000 games [34], only 650 support DLSS [32] and 350–400 support FSR [33]—well under 1% of the catalog. With additional titles on Epic Games Store, PlayStation Store, and Xbox Marketplace, the relative coverage is even smaller across the broader gaming landscape.

Suitability of Existing Image/Video Upscalers for Gaming: Prior work has proposed numerous SR models, including EDSR [17], LapSRN [18], IMDN [19], LatticeNet [20], and SwinIR [21], with recent focus on real-time efficiency through architectural refinements, compression, and novel training methods [35]–[38].

These lightweight models employ strategies like information distillation (IMDN), feature distillation (RFDN), Laplacian pyramids (LapSRN), and residual-attention mechanisms (LatticeNet). However, they target general efficiency rather than the millisecond-level latency demands of cloud gaming.

To quantify this gap, we evaluate existing SR models on gaming content (§IV-B). Even IMDN, the most efficient general-purpose model, takes over 120 ms to upscale a frame by $2\times$ on an NVIDIA RTX A4000, far exceeding real-time limits. RT4KSR, the most relevant real-time baseline, achieves $\sim$15 ms per frame but yields lower perceptual quality on gaming content, as shown in Table II). By contrast, GameSR takes 4.1 ms for $2\times$ scaling to 1080p on the same hardware with substantially higher quality, and latency scales proportionally for higher output resolutions (see Table IV).

Recurrent VSR methods such as RLSP [39], MRVSR [40], BasicVSR++ [41], and SSL-pruned BasicVSR [42] use heavier backbones with fixed $4\times$ scaling on small inputs and report tens of milliseconds per frame. In contrast, GameSR targets $2\times$ and $3\times$ upscaling of full HD streams (540p/720p$\rightarrow$1080p) in 4–5 ms with only 138K parameters—a more suitable operating point for real-time gaming.

Additional Challenges of Cloud Gaming: In cloud gaming, clients receive only compressed video streams, so industrial upscalers and rendering-coupled research models [13], [15],

This article has been accepted for publication in IEEE Transactions on Games. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TG.2026.3710843

Fig. 1. GameSR in cloud gaming: low-resolution streams are rendered at server-side and upscaled at client-side in real-time.

[43]–[47] cannot be applied—they require motion vectors, depth, and other engine-level data unavailable client-side. Even if executed server-side, such methods would not reduce streaming bitrate since frames must still be transmitted at display resolution.

While VOD streaming has explored SR integration via segment-specific micro-models [48]–[50], this is infeasible for *interactive* cloud gaming where frames are generated in real-time based on player inputs. Recent work by Jiang et al. [51] addresses this by maintaining a lookup table of pre-trained SR models for cloud gaming, selecting the most relevant model per video segment to reduce training overhead, though their approach targets mobile devices at lower resolutions (∼720p at 20 fps) rather than real-time HD/4K upscaling.

Feasibility of Running Upscalers on Client Devices: Most client devices possess underutilized compute capable of running upscalers: smartphones include powerful NPUs (iPhone 16 Pro: 35 TOPS; Dimensity 9400: 50 TOPS), while consoles like PS5 and Xbox Series X offer over 10 TFLOPS [52]–[55]. However, heavy VSR models remain impractical due to power, thermal, and latency constraints.

GameStreamSR [56] addresses this by upscaling only depth-defined regions on mobile, using bilinear interpolation elsewhere—but this engine-dependent strategy limits quality outside the focus area and requires render buffer access. In contrast, GameSR is a lightweight, full-frame, engine-agnostic upscaler that runs within milliseconds on both mobile and desktop hardware, compatible with post-decoder pipelines using only compressed RGB video.

III. Proposed Solution

A. Overview and Operation

We design GameSR as an engine-independent, lightweight super-resolution (SR) model for both traditional and cloud gaming. In traditional gaming, GameSR applies as a post-processing step after rendering, enhancing frames before display.

In cloud gaming (Figure 1), frames are rendered on the server, compressed, and transmitted to clients. GameSR is inserted between the decoder and display, transparently upscaling frames in real-time as they arrive.

Beyond improving perceived quality, GameSR offers three advantages for cloud gaming: (i) reduced server rendering and encoding load by operating at lower resolutions, (ii) lower transmission bitrate since fewer pixels are streamed, and (iii) no integration with game engines or decoders required, enabling deployment for both recent and legacy games.

This design makes GameSR both engine-agnostic and codec-agnostic: any title or streaming service outputting RGB video can benefit without modifying the game engine or depending on vendor-specific upscaling stacks.

The key challenge is meeting strict latency deadlines in interactive gaming. Figure 2 illustrates our high-level design; as shown in Section IV, GameSR improves quality while meeting real-time requirements. We detail the components below.

B. GameSR Details

We design GameSR as a lightweight SR model for latency-sensitive gaming content, with neural layers and components specifically designed for efficiency and effectiveness. GameSR, and SR models in general, reconstructs high-resolution (HR) frames from low-resolution (LR) inputs by optimizing a parameterized function F as follows:

$$\theta^* = \arg\min_{\theta} \sum L\left(F(y^{\text{LR}};\theta), y^{\text{HR}}\right). \tag{1}$$

Fundamentally, the function F performs three main tasks in super-resolution problems: Feature Representation, Feature Learning, and Mapping LR frames to HR ones. In our design, we extend this formulation by introducing a fourth stage—*Temporal Learning*—which leverages information from adjacent frames before the final mapping stage. We summarize each of these tasks in the following. More details can be found in Supplementary Material sec B.

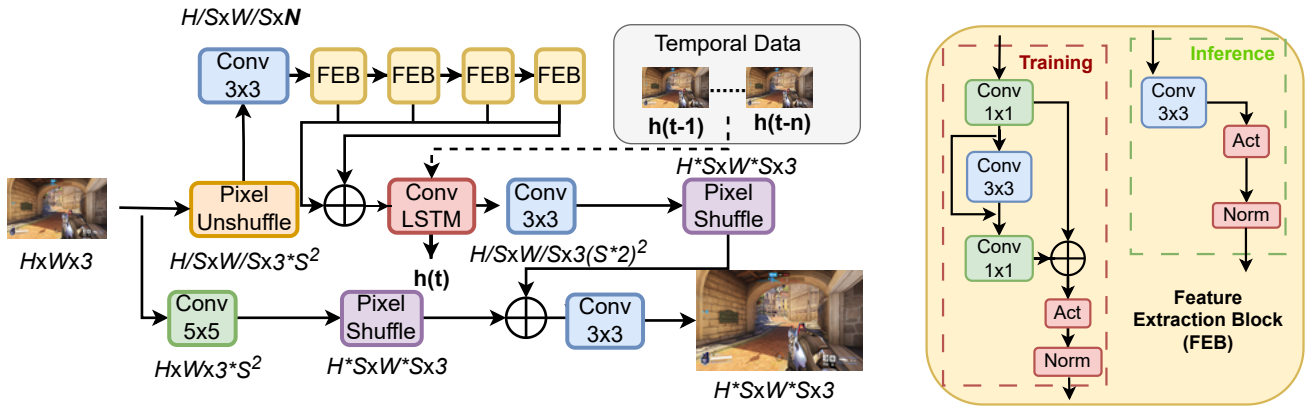
While the formulation in Eq.1 is general and GameSR could in principle be applied to other video domains, our design is motivated by the unique characteristics of gaming content. As opposed to traditional multimedia, gaming video is synthetic and exhibits recurring objects, structured environments, and repetitive motion patterns [24]. These properties enable per-game, data-centric training and make it possible to realize an extremely lightweight SR model that still achieves high perceptual quality.

Feature Representation for Upscaling: The feature representation stage employs a single 3×3 convolution and PixelUnshuffle (space-to-depth) [57], reducing spatial dimensions by a factor of s and expanding channels by s^2. Unlike conventional SR methods [19], [58], [59], this down-and-up scheme significantly reduces computational cost while capturing richer channel-wise feature relationships. The formulation is:

$$F_1(y^{LR})_{f_1 \times \frac{H}{s} \times \frac{W}{s}} = \max\Big(0,\, W_1 * \text{PixelUnshuffle}(y^{LR}) + B_1\Big) \tag{2}$$

where W_1 and B_1 are convolution weights and biases. PixelUnshuffle with scale factor s rearranges the input from $c \times H \times W$ into $c \cdot s^2 \times \frac{H}{s} \times \frac{W}{s}$, increasing the channel dimension

This article has been accepted for publication in IEEE Transactions on Games. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TG.2026.3710843



(a) GameSR architecture (b) FEB

Fig. 2. Overview of GameSR: (a) architecture where frames are downsampled with PixelUnshuffle, processed by Feature Extraction Blocks, and passed through a lightweight ConvLSTM before upsampling via PixelShuffle with a residual connection; (b) internal structure of FEB.

by s^2 while reducing spatial dimensions. These features are then passed into Feature Learning blocks.

Feature Learning: The feature learning stage captures non-linear mappings between LR and HR features using our Feature Extraction Block (FEB), which is shown in Figure 2b. During training, each FEB applies a $1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$ convolution sequence, expanding and then compressing feature dimensions. At inference, we merge these into a single convolution via reparameterization [60], greatly reducing computational load without accuracy loss.

Each FEB incorporates GeLU activations and LayerNorm [61] for stable and efficient training. Residual connections preserve spatial detail and facilitate gradient flow. We use four FEBs, chosen for the best quality–latency trade-off. After sequential FEBs, we employ multi-level feature aggregation through additive fusion, defined as:

$$RB_{final} = \sum_{i=0}^{N} RB(i), \tag{3}$$

where each FEB output is combined additively, enhancing gradient propagation, feature reuse, and memory efficiency. The aggregated features are then fed into a lightweight ConvLSTM to capture temporal information

Temporal Learning: Video super-resolution (VSR) leverages temporal information across frames to enhance quality, making it especially relevant for gaming sequences in cloud gaming. Unlike single-image SR, VSR exploits motion continuity through either explicit (e.g., optical flow [62]) or implicit alignment (e.g., 3D/deformable convolutions [63], [64]). However, most VSR models are too computationally heavy for real-time deployment.

To balance temporal modeling and efficiency, we adopt a lightweight variant of ConvLSTM [65] after feature extraction. ConvLSTM replaces matrix multiplications in standard LSTMs with convolutions, preserving spatial resolution while capturing long-range dependencies. Our design uses a single-layer structure with decoupled gates (input, forget, output, and cell), each implemented with independent 2D convolutions. This modular design enables better parallelization on modern GPUs while minimizing sequential overhead.

During inference, frames are processed sequentially using hidden states from prior frames, enabling effective motion-aware upsampling. The ConvLSTM operates over spatial features with dimensions $(C, H/s, W/s)$ and uses standard gate updates:

$$\begin{aligned} &i_t = \sigma(W_i * [x_t, h_{t-1}] + b_i), \quad f_t = \sigma(W_f * [x_t, h_{t-1}] + b_f), \\ &o_t = \sigma(W_o * [x_t, h_{t-1}] + b_o), \quad \tilde{c}_t = \tanh(W_g * [x_t, h_{t-1}] + b_g), \\ &c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t). \end{aligned} \tag{4}$$

Here, $*$ denotes 2D convolution and $[\cdot, \cdot]$ is channel-wise concatenation. This temporal module significantly improves perceptual quality under motion (see Section IV-C). Finally, the temporally enhanced features are upsampled back to display resolution.

Mapping from Low to High Resolutions: Our upsampling stage utilizes Pixel-shuffling for spatial resolution enhancement, avoiding checkerboard artifacts common in deconvolution methods [66], [67]. This approach reshapes feature channels into spatial dimensions efficiently. We incorporate a residual connection by combining upsampled ConvLSTM output with the original input, preserving fine details and textures. Formally, the operation is expressed as:

$$\hat{y}^{SR}_{c\times H\times W} = \mathrm{Conv}\Big(\mathrm{PixelShuffle}\big(\max\big(0,\, W_{\mathrm{up}} * (RB)_{f_1\times H\times W} + B_{\mathrm{up}}\big)\big)\Big). \tag{5}$$

IV. Evaluation

A. Setup and Performance Metrics

Games: We evaluate on four distinct games: Counter-Strike 2 (CS2), Overwatch 2 (OW2), FC24, and Team Fortress 2 (TF2). CS2, OW2, and FC24 represent modern, high-demand titles spanning FPS and sports genres, while TF2 serves as a legacy case. Using VirtualDub [68], we captured uncompressed 1080p gameplay at 30/60 FPS, with V-Sync disabled and high graphics-quality presets, across diverse maps, motions, camera dynamics, and lighting conditions. Five players of varying skill levels recorded five sessions each per game (25 sessions

per title), yielding 40k frames for CS2, 54k for OW2, and 30k for TF2, plus an additional FC24 corpus of comparable scale covering representative match scenarios (e.g., build-up play, counterattacks, set pieces, and goal events). We used 10 sessions per game for training and 15 unseen sessions for testing.

Performance Metrics: We evaluate quality using commonly used metrics: PSNR, SSIM, VMAF [69], and LPIPS [70]. PSNR/SSIM are pixel-based, while VMAF/LPIPS better capture perceptual quality. In gaming, reference frames are often unavailable due to engine non-determinism, floating-point variability, multithreaded scheduling, and event-driven randomness, which prevent frame-level consistency [71]. Thus, we also employ two no-reference models: NDNetGaming [72], tailored to gaming with MOS-like scores, and VSFA [73], a ResNet-50+GRU model. These are primarily used for DLSS/FSR comparisons. In addition, we measure bandwidth, and GPU usage.

Training: We trained GameSR in PyTorch 2.0.1 on an NVIDIA RTX A4000 with an Intel Xeon Gold 5220 CPU and 32 GB RAM. Training used AdamW [74] (β_1=0.9, β_2=0.999), learning rate 10^{-3} halved every 2×10^5 iterations, minibatch size 16, and Charbonnier loss. Data was split 80/20 for training/validation. For deployment, we compiled the model with Torch-TensorRT using kernel fusion and mixed precision (FP32 inputs, FP16 kernels) to improve throughput and memory efficiency.

Beyond architectural choices, the training objective affects both convergence and reconstruction quality. While prior SR work commonly uses MSE (L2) or MAE (L1), we optimize GameSR with the Charbonnier loss [75],

$$\mathcal{L}_{\text{char}} = \mathbb{E}_{(z,y)\sim P_{\text{data}}}[\rho(y - G(z))], \quad \rho(x) = \sqrt{x^2 + \epsilon^2}, \quad (6)$$

a smooth, robust surrogate for L1 that behaves L2-like for small residuals and L1-like for large errors. In our experiments, it provided more stable convergence than MSE/MAE, and we use it throughout.

B. Performance Analysis of GameSR

GameSR Performance To evaluate GameSR's performance, we utilized it to upscale diverse gameplay sessions across different maps, users, and character configurations, ensuring a wide range of visual variability. We present sample results in Figure 3 for upscaling sessions from the CS2 and OW2 games by a factor of 2X. As shown in the figure, GameSR consistently achieves high-quality results: PSNR ranges from 36–40 dB, SSIM exceeds 0.998, and VMAF scores fall within the 90–95 range—indicating excellent quality [76], [77].

GameSR vs. Commercial DLSS and FSR Upscalers: We compare GameSR against industry-standard upscalers: specifically, **FSR 1.0** on CS2, and **FSR 2.2** and **DLSS 3.5** on OW2. FSR and DLSS were applied using in-game settings, whereas we rendered frames natively at 540p and upscaled them directly using GameSR. This approach was designed to provide a realistic reference point; however, it is essential to note that this setup is not entirely fair to GameSR. While DLSS and FSR have access to additional renderer data (e.g., motion vectors, depth buffers), GameSR relies solely on the input frames for upscaling. We note that newer iterations of these industrial upscalers have since been released (e.g., DLSS 4 with Multi Frame Generation, FSR 3.1/Redstone), which offer further improvements in perceptual quality and can generate additional frames beyond upscaling. GameSR is complementary to these solutions: rather than competing directly, it enables engine-agnostic deployment across the vast majority of titles that lack engine integration for any industrial upscaler.

TABLE I
COMPARING GAMESR AGAINST DLSS AND FSR, WHICH REQUIRE ENGINE-LEVEL DATA. IN CONTRAST, GAMESR UPSCALES ENCODED STREAMS. RESULTS SHOWN FOR 2× SCALING ON CS2, OW2, AND TF2.

Model	Counter-Strike 2		Overwatch 2		Team Fortress 2	
	NDNet ↑	VSFA ↑	NDNet ↑	VSFA ↑	NDNet ↑	VSFA ↑
DLSS	**-**	**-**	**4.93**	**0.88**	-	-
FSR	**5.00**	**0.89**	4.81	0.81	-	-
GameSR	4.90	0.83	4.76	0.81	**4.79**	**0.78**

We summarize the comparison results in Table I. Sample frames produced by the considered upscalers are presented in Figure 4 for visual comparisons. We compared GameSR to FSR 1.0/2.2 and DLSS 3.5 using the same gameplay sequences, maps, and camera paths to ensure a fair comparison. In CS2, we tested GameSR against FSR 1.0 in "Performance" mode (2× upsampling), matching GameSR's scaling factor.

GameSR scored 4.9 (NDNetGaming) and 0.83 (VSFA), closely trailing FSR's 5.0 and 0.89. For OW2, which supports both FSR 2.2 and DLSS 3.5, we also used 2× upscaling factor. GameSR achieved scores of 4.76 and 0.81, nearly matching FSR (4.81, 0.81) and DLSS (4.93, 0.88).

On the evaluated titles, GameSR achieves competitive no-reference perceptual quality compared to FSR and DLSS, with differences of only 0.1 (NDNetGaming) and 0.06 (VSFA) in CS2, and within 0.05 (FSR) and 0.17 (DLSS) for NDNetGaming in OW2, despite not utilizing any game render data. The engine-independent nature makes it broadly deployable across platforms and titles. For instance, CS2 employs forward rendering and currently does not support temporal elements required by DLSS 2+ or FSR 2+, meaning those modern upscalers cannot be adopted without changes to the rendering pipeline [78].

Team Fortress 2 serves as a representative legacy title in our evaluation. Like many older games, it has not been updated to modern engines such as Source 2, which restricts compatibility with contemporary upscalers like DLSS and FSR that rely on motion vectors, depth buffers, and temporal anti-aliasing. As a result, TF2 and similar legacy titles cannot natively benefit from these industrial solutions. In contrast, GameSR operates directly on rendered frames without engine-level modifications, delivering high-fidelity upscaling comparable to modern titles and extending the visual longevity of older games while maintaining broad deployability.

Figure 4 shows side-by-side comparisons of upscaled frames on OW2 and CS2. The top two rows present Overwatch 2 sequences from the Esperança and Nepal maps, comparing DLSS 3.5, FSR 2.2, and GameSR; OW2 natively supports both DLSS and FSR, allowing direct visual comparison. The

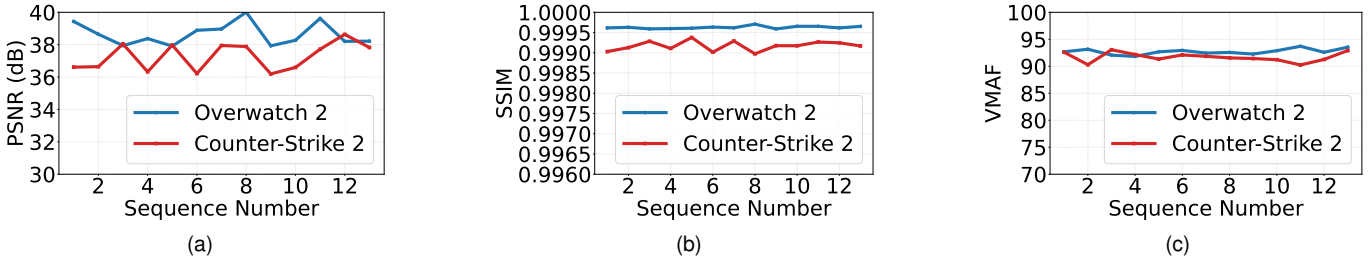


Fig. 3. Performance of GameSR on upscaling game sessions from CS2 and OW2 by a factor of 2×.

bottom row presents a Counter-Strike 2 sequence from the Dust II map, comparing FSR 1.0 and GameSR; as noted above, CS2 employs forward rendering and supports only FSR 1.0, since its pipeline lacks the temporal data required by DLSS 2+ and FSR 2+. Zoomed-in insets highlight fine detail in high-frequency regions; GameSR preserves comparable sharpness and texture fidelity to the engine-integrated upscalers across both titles. Supplemental video comparisons showing GameSR, DLSS, FSR, and native resolution across multiple gameplay sequences are available online at 30 and 60 FPS.[1]

Beyond quality, we also measured GPU usage and FPS (Figure 5c). Native 1080p runs at ≈123 FPS and ∼100% GPU, whereas 540p+GameSR runs at ≈125 FPS with only ∼82% GPU, since it shades 4× fewer pixels and adds only a lightweight SR pass. DLSS 3.5 (≈140 FPS) and FSR 2.2 (≈145 FPS) also render internally at reduced resolution but use the saved budget for higher FPS, so their GPU usage stays near 99%; in contrast, 540p+GameSR exposes headroom that, in a cloud setup with server-side rendering and client-side upscaling, can translate into meaningful compute savings. ***In summary***, GameSR not only saves bandwidth, but it also reduces the computational power needed to render games. This is achieved while providing competitive perceptual quality to DLSS/FSR on no-reference metrics, without accessing the rendering engine's data structures or modifying the game source code.

Information gap vs. model capacity: The quality difference between GameSR and engine-integrated upscalers has two sources: the *information gap* (DLSS/FSR access motion vectors, depth buffers, and renderer metadata, whereas GameSR operates solely on compressed RGB) and *model capacity* (industrial upscalers use larger architectures). Our results suggest the information gap is the primary factor. GameSR compensates for this information gap through game-specific, data-centric training that exploits the bounded visual world of each title: recurring textures, HUD elements, and structured motion patterns allow an extremely lightweight model (138K parameters, 5× less GPU memory) to achieve competitive or superior quality to substantially larger general-purpose models such as IMDN (880K parameters) and EDSR (1.37M), indicating that model size is not the bottleneck. Our ablation study (Table VIII) further shows that ConvLSTM is critical for partially bridging this gap: removing it drops PSNR by 5 dB on CS2, confirming that implicit temporal modeling recovers a substantial portion of the motion information that engine-integrated upscalers obtain through explicit motion vectors. The remaining gap likely requires access to accurate motion vectors and depth information that only engine integration can provide.

GameSR vs. State-of-the-Art Upscalers in the Literature: To assess the performance and efficiency of our lightweight model, GameSR, we conducted a comparative analysis against several state-of-the-art (SOTA) SR models, including EDSR [17], LapSRN [18], IMDN [19], LatticeNet [20], RT4KSR [79] and SwinIR [21]. In our evaluation, a scaling factor of 2 corresponds to upsampling from 540p→1080p Table II, while a scaling factor of 3 corresponds to 360p→1080p Table III.

Table II shows that GameSR achieves near state-of-the-art quality (PSNR/SSIM/LPIPS) while running orders of magnitude faster. For 2× scaling (540p→1080p), GameSR reaches ∼240+ fps (∼4.1 ms/frame) on an RTX A4000, versus < 10 fps for EDSR/LatticeNet and < 1 fps for SwinIR. Inference latency is consistent across all four titles (CS2, OW2, TF2, FC24) since the model's cost depends only on input resolution; the times in Tables II and III are averages across all four games, and latency for higher-resolution targets is reported in Table IV. We also include RT4KSR [79], the most relevant real-time baseline. The original RT4KSR paper reports times on an NVIDIA RTX 3090 Ti; for fairness, we evaluate it on the same A4000 as all other models. Additionally, GameSR benefits from Torch-TensorRT optimization (kernel fusion, mixed precision), whereas RT4KSR has no TensorRT support and is evaluated with standard PyTorch. These two factors account for the latency difference from the original paper. Despite being substantially faster (∼67 fps) than large SOTA models, RT4KSR yields noticeably lower quality on gaming content. SwinIR attains the best perceptual scores via its Transformer design but is too costly for latency-sensitive settings. For a fair lightweight-CNN comparison, we retrained IMDN on our CS2 dataset (Table VI): GameSR achieves comparable quality with only a 0.18 dB PSNR gap while using 5× fewer parameters and 4.5× less GPU memory, demonstrating the benefit of data-centric training and GameSR's temporal modeling.

GameSR's efficiency comes from three design choices: ConvLSTM captures temporal dependencies, reparameterization enables wide training but lightweight inference, and PixelUnshuffle reduces spatial cost. Together, these yield real-time performance with high visual fidelity.

[1]https://tinyurl.com/29an495a

This article has been accepted for publication in IEEE Transactions on Games. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TG.2026.3710843

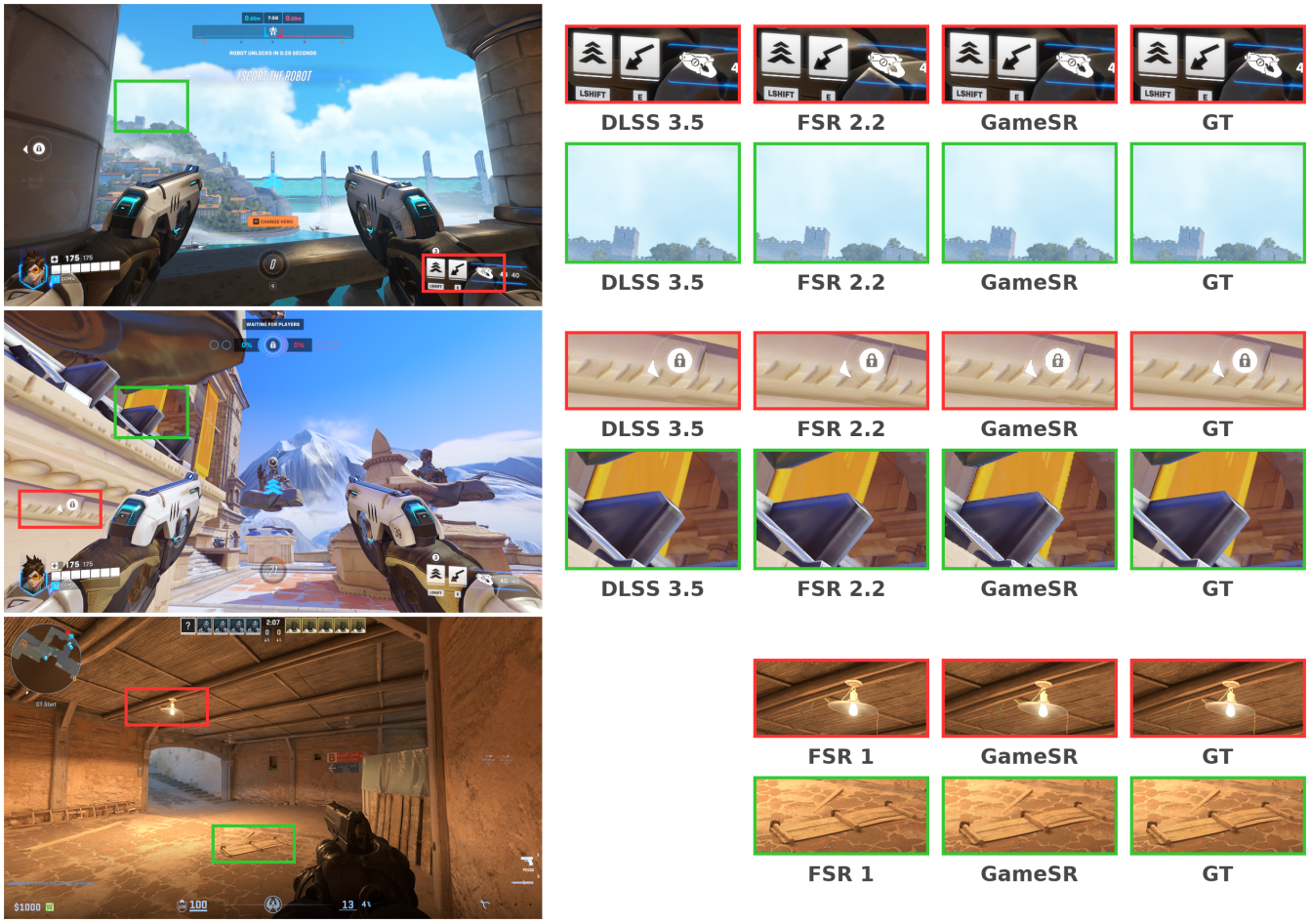


Fig. 4. Visual comparison of upsampling methods with zoomed-in insets. The top two rows show Overwatch 2 on the Esperança and Nepal maps, comparing DLSS 3.5, FSR 2.2, and GameSR. The bottom row shows Counter-Strike 2 on Dust II, comparing FSR 1.0 and GameSR; CS2 uses forward rendering and supports only FSR 1.0. Red boxes indicate the zoom regions.

Table III presents results for 3× scaling (360p→1080p). The trends mirror those observed at 2×: GameSR delivers quality on par with state-of-the-art SR models while being orders of magnitude faster. Since the 3× path starts from a smaller input (360p vs. 540p), inference time is slightly lower (∼4.09 ms vs. ∼4.12 ms), consistent with the resolution-dependent scaling of computational cost.

Beyond accuracy and runtime, we compared parameter counts and GPU memory across models. As shown in Figure 5(a,b), GameSR uses only 138K parameters and 604 MiB memory, compared to 1.37M/11.9 GiB for EDSR and 910K/3.9 GiB for SwinIR. IMDN and LatticeNet also require 5–6× more memory. At ×2 scale (and similarly at ×3), GameSR achieves order-of-magnitude savings in size and memory over SOTA upscalers.

Scalability and generality to high-resolution upscaling: We further evaluate GameSR on 2K (2560×1440) and 4K (3840×2160) CS2 gameplay sequences. Table IV reports reconstruction quality and latency for three high-resolution mappings: 720p→2K, 1080p→4K, and 720p→4K. As expected, inference time increases with the input spatial resolution: the model's computational cost is dominated by per-pixel convolutions, so larger inputs directly translate to higher latency. For 1080p→4K, GameSR achieves up to 39.25 dB PSNR, SSIM above 0.999, and VMAF above 93 with an inference time of 14.3 ms per frame (∼70 FPS), comfortably within real-time 60 FPS budgets. For 720p→2K, the smaller input size reduces latency to only 7 ms per frame (∼143 FPS), sufficient even for 120 Hz gaming. All 2K/4K clips are played by different users, collected after training, and remain strictly unseen during training and validation, mirroring the protocol used for our 1080p test set. We note that these latency figures are consistent across game titles, as inference time depends on the input resolution rather than on the visual content of the game.

Despite being trained solely on 1080p content, GameSR can be directly applied to substantially higher-resolution inputs without fine-tuning, while preserving temporal stability, maintaining high perceptual quality, and sustaining low latency. This demonstrates that GameSR scales effectively to practical high-resolution cloud gaming scenarios.

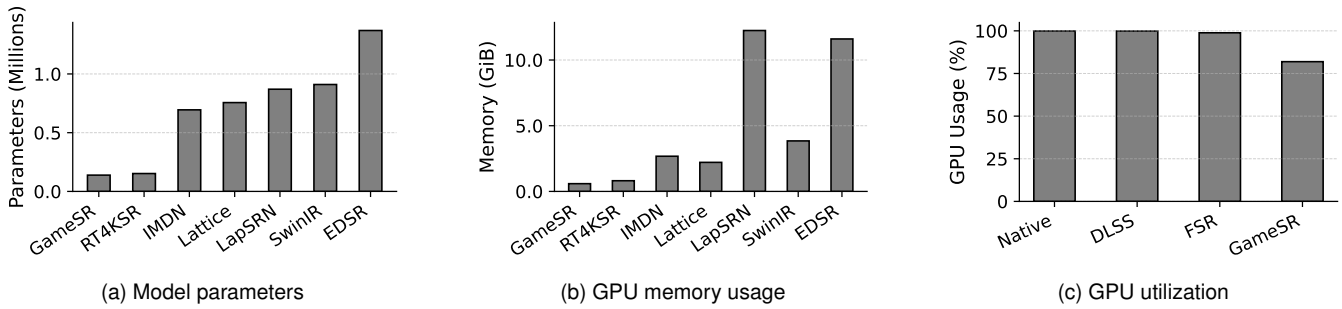
User Study. To further assess perceptual quality, we conducted a subjective evaluation following ITU-T Recommendation P.910 and ITU-R BT.500 [80], [81]. A total of **15**

TABLE II
QUANTITATIVE COMPARISON BETWEEN STATE-OF-THE-ART SUPER-RESOLUTION MODELS AND GAMESR AT 2× SCALING (540P→1080P) ON FOUR POPULAR GAMES. INFERENCE TIMES ARE AVERAGED ACROSS ALL FOUR TITLES. ALL MODELS EVALUATED ON AN NVIDIA RTX A4000 GPU UNDER IDENTICAL CONDITIONS.

Model	Inference (ms)	Counter-Strike 2			Overwatch 2			Team Fortress 2			FC24		
		PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Bicubic	-	32.50	0.998	0.189	35.22	0.999	0.128	34.07	0.998	0.128	31.82	0.997	0.190
SwinIR	1971.7	35.92	0.998	**0.084**	**40.74**	**0.999**	**0.016**	40.10	**0.999**	**0.046**	**34.91**	0.998	0.155
LapSRN	239.7	33.63	0.998	0.107	38.09	0.999	0.030	36.77	0.998	0.144	33.48	0.998	0.172
EDSR	160.0	35.30	0.998	0.091	40.16	0.999	0.018	39.41	0.999	0.050	34.56	0.998	0.157
LatticeNet	154.4	35.46	0.998	0.088	40.29	0.999	0.017	39.36	0.999	0.050	34.71	0.998	0.159
IMDN	121.2	35.38	0.998	0.089	40.33	0.999	0.018	39.36	0.999	0.050	34.71	0.998	0.154
RT4KSR	15	33.74	0.998	0.154	37.11	0.999	0.101	36.04	0.999	0.093	33.35	0.998	0.163
GameSR	**4.12**	**37.99**	**0.999**	0.095	40.36	0.999	0.021	**40.88**	0.999	0.051	34.84	**0.998**	**0.143**

TABLE III
QUANTITATIVE COMPARISON BETWEEN STATE-OF-THE-ART SUPER-RESOLUTION MODELS AND GAMESR AT 3× SCALING (360P→1080P) ON FOUR POPULAR GAMES. INFERENCE TIMES ARE AVERAGED ACROSS ALL FOUR TITLES. ALL MODELS EVALUATED ON AN NVIDIA RTX A4000 GPU UNDER IDENTICAL CONDITIONS.

Model	Inference (ms)	Counter-Strike 2			Overwatch 2			Team Fortress 2			FC24		
		PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
SwinIR	881.2	32.53	0.996	**0.178**	36.79	**0.998**	**0.052**	36.38	**0.998**	**0.106**	**31.41**	**0.992**	0.264
LatticeNet	70.5	32.25	0.996	0.189	36.42	0.9987	0.056	35.81	0.998	0.113	31.20	0.992	0.268
EDSR	76.6	32.26	0.996	0.188	36.32	0.9986	0.059	35.82	0.998	0.113	31.13	0.992	0.278
IMDN	55.2	32.29	0.996	0.186	36.40	0.9987	0.055	35.85	0.998	0.110	31.20	0.992	0.272
RT4KSR	9.0	30.99	0.995	0.253	33.75	0.997	0.194	33.17	0.997	0.174	29.77	0.991	0.279
GameSR	**4.09**	**35.46**	**0.996**	0.180	**36.99**	0.998	0.059	**38.10**	0.998	0.111	31.23	0.992	**0.253**



(a) Model parameters (b) GPU memory usage (c) GPU utilization

Fig. 5. Efficiency of GameSR compared to state-of-the-art methods for 2× scaling. (a,b) compare parameter count and GPU memory; (c) compares GPU utilization against DLSS, FSR, and native rendering.

TABLE IV
SUPER-RESOLUTION PERFORMANCE OF GAMESR ON HIGH-RESOLUTION CS2 CLIPS (E.G., 720P→2K). ALL EXPERIMENTS WERE CONDUCTED ON AN NVIDIA A4000 GPU.

Source → Target	Time (ms)	PSNR (dB)	SSIM	VMAF
720p → 2K	7.0	36.24	0.998	91
1080p → 4K	14.3	39.25	0.999	93
720p → 4K	15.3	34.83	0.999	88.13

participants took part, comprising experienced gamers (40%), occasional gamers (40%), and non-gamers (20%). The study included two conditions: (i) a *rendered* condition, where participants rated GameSR-upscaled frames directly, and (ii) a *paired cloud gaming* condition, where participants viewed both native 1080p WebRTC streams and 540p streams upscaled by GameSR at 25 Mbps, rating each on a 5-point MOS scale across **26 stimuli** per participant.

Rendered condition. In the rendered condition, participants rated the visual quality of GameSR-upscaled output without a baseline comparison. As shown in Table V, GameSR achieved an overall MOS of **4.73/5** for CS2 and **4.70/5** for OW2, with narrow 95% confidence intervals (±0.08–0.11) indicating low inter-participant variability. Scores were stable across all experience groups (Kruskal–Wallis: $p = 0.09$ for CS2, $p = 0.50$ for OW2), confirming that gamers and non-gamers alike perceive the upscaled output as high quality.

Paired cloud gaming comparison. The paired condition provides the key statistical evidence: participants rated both native 1080p WebRTC streams and GameSR-upscaled streams under identical cloud gaming conditions at 25 Mbps. Table VII summarizes the results. A TOST equivalence test with a margin of ± 0.5 MOS (half a quality category) confirms perceptual equivalence for both CS2 and OW2 ($p < 0.001$), positively asserting that the quality difference falls within an imperceptible range. Wilcoxon signed-rank tests show no

This article has been accepted for publication in IEEE Transactions on Games. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TG.2026.3710843

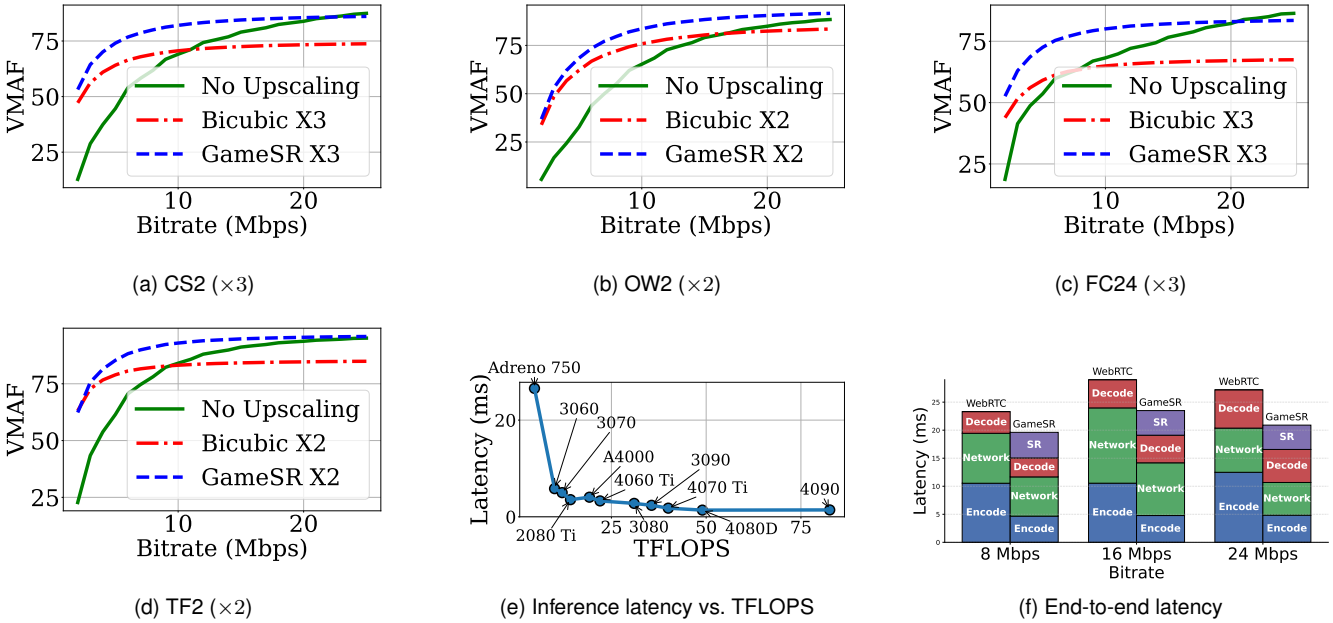


Fig. 6. VMAF vs. bitrate for all four titles (a–d), inference latency vs. compute capability (e), and end-to-end latency breakdown across bitrates (f). GameSR consistently achieves higher VMAF at lower bitrates compared to bicubic upscaling and native 1080p streaming, while adding only ∼4–5 ms of client-side SR compute and reducing overall latency by ∼19% through smaller frame sizes.

TABLE V
RENDERED CONDITION MOS FOR GAMESR ACROSS PARTICIPANT GROUPS. EXP.=EXPERIENCED, OCC.=OCCASIONAL, NO.=NON-GAMERS.

Game	MOS	95% CI	Exp.	Occ.	No.
CS2	4.73	[4.63, 4.80]	4.62	4.75	4.84
OW2	4.70	[4.59, 4.82]	4.68	4.75	4.68

TABLE VI
COMPARISON OF GAMESR AGAINST IMDN AND IMDN-G TRAINED ON CS2.

Model	PSNR	SSIM	LPIPS
IMDN	35.37	0.9988	0.0898
IMDN-G	**38.17**	**0.9990**	**0.0835**
GameSR	37.99	0.9990	0.0954

significant difference for either game (CS2: $p = 0.106$; OW2: $p = 0.582$), with mean differences of only $+0.12$ and $+0.05$ MOS—far smaller than what users can perceive. A Kruskal–Wallis test confirmed no significant differences across experience groups (all $p > 0.41$), meaning experienced gamers, occasional gamers, and non-gamers all rated GameSR quality similarly.

Paired comparisons against DLSS and FSR were not included because these upscalers execute server-side during rendering, whereas GameSR operates client-side after decoding; native 1080p WebRTC streaming serves as the appropriate upper-bound baseline. The subjective evaluation focused on CS2 and OW2 as the most visually complex titles; objective metrics (Tables II and III) and the end-to-end evaluation (§IV-E) confirm consistent trends across all four titles. At $N = 15$, the study provides 80% power to detect large effects (Cohen's $d \geq 0.78$), and TOST equivalence confirmation validates that GameSR delivers quality statistically equivalent to native 1080p streaming while enabling 35–56% bandwidth reduction.

Inference Evaluation Across Heterogeneous Hardware: We evaluate the scalability of GAMESR across a diverse set of client devices, ranging from modern mobile SoCs to high-end desktop GPUs. Figure 6e reports the inference latency of GAMESR for 2× upscaling as a function of device TFLOPS. GAMESR-M, which is designed using only mobile-friendly operations, achieves real-time performance on an Adreno 750 mobile GPU (4.7 TFLOPS), requiring 26.56 ms per 1080p frame for 2× scaling and 14.55 ms per frame for 3× scaling. Thus, the 3× configuration satisfies the 60 FPS (16.7 ms) budget. On discrete GPUs, latency decreases roughly with available compute: mid-range GPUs such as the RTX 4060 Ti, RTX 4070 Ti, and A4000 sustain about 1.7–4.0 ms per frame for 2× scaling, while high-end cards such as the RTX 3090 and RTX 4090 reduce latency to about 2.3 ms and 1.3 ms, respectively. Overall, these results highlight the hardware scalability of GAMESR and its ability to meet real-time budgets across a wide spectrum of client devices.

C. Ablation Study

To validate the contributions of each major component, we performed an ablation study removing ConvLSTM, PixelUnshuffle, and Reparameterization one at a time. Table VIII compares inference time, memory, parameters, and quality metrics. All ablation timings are reported for 2× scaling (540p→1080p) on the RTX A4000.

ConvLSTM: Temporal processing across frames proves critical. Without ConvLSTM, PSNR drops approximately 5 dB

TABLE VII
PAIRED CLOUD GAMING COMPARISON (25 MBPS): WEBRTC VS. GAMESR MOS SCORES WITH 95% CI. Δ MOS = GAMESR − WEBRTC. TOST EQUIVALENCE MARGIN: ±0.5 MOS.

Game	WebRTC MOS [CI]	GameSR MOS [CI]	Δ MOS	Wilcoxon p	TOST p
CS2	4.37 [4.20, 4.53]	4.48 [4.28, 4.69]	+0.12	0.106	<0.001
OW2	4.43 [4.27, 4.59]	4.48 [4.28, 4.69]	+0.05	0.582	<0.001

TABLE VIII
ABLATION STUDY OF GAMESR ON CS2, SHOWING THE IMPACT OF CONVLSTM, PIXELUNSHUFFLE, AND REPARAMETERIZATION ON EFFICIENCY AND QUALITY.

Model	#Params (K)	Memory (MiB)	Inference (ms)	CS2 PSNR	CS2 SSIM	CS2 LPIPS
GameSR (No ConvLSTM)	**65**	**436**	**3.05**	32.99	0.998	0.116
GameSR (No PixelUnshuffle)	125	1174	13.13	38.65	0.999	0.087
GameSR (No Reparam.)	298	608	6.29	37.99	0.998	0.095
GameSR (Final Model)	138	604	4.12	37.99	0.998	0.095

TABLE IX
CROSS-GAME GENERALIZATION OF GAMESR. MODELS TRAINED ON CS2, OW2, AND A COMBINED DATASET ARE EVALUATED ON BOTH GAMES.

Test Sequence	Game data used for training					
	CS2		OW2		CS2+OW2	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
CS2	37.80	0.999	34.63	0.999	37.53	0.999
OW2	37.17	0.999	38.81	0.999	38.74	0.999

(from 37.99 dB to 32.99 dB on CS2), with similar degradations in SSIM and LPIPS. While removing it reduces parameters to 65K and speeds inference to 3.05 ms, the substantial quality loss demonstrates the importance of temporal modeling.

PixelUnshuffle: This component balances quality and efficiency. Without it, the model achieves the highest quality (PSNR: 38.65 dB), but inference time triples to 13.13 ms and memory nearly doubles to 1174 MiB—impractical for real-time applications.

Reparameterization: This technique improves efficiency without compromising quality. Compared to the non-reparameterized version, our final model reduces parameters by 54% (298K→138K) and improves inference from 6.29 ms to 4.12 ms, while maintaining identical quality metrics.

D. Model Generalization

To assess generalization, we trained models on CS2, OW2, and their combination (Table IX). Models perform best in-domain but cross-game evaluations still yield competitive results. The combined CS2+OW2 model performs strongly on both, suggesting that shared motion and visual structures within the shooter genre [25] improve robustness and enable effective multi-game training.

E. End-to-End Cloud Gaming System

We evaluate client-side SR in a real WebRTC cloud-gaming pipeline using *GameLab* [82], an open-source aiortc-based testbed that exposes hooks to swap post-decode processing at the client and log per-stage latency (encode/transport/decode/client). The server and client are connected via a dedicated 1 Gbps Ethernet switch.

Network setup rationale. The 1 Gbps Ethernet link serves only as the physical transport medium between the server and client; realistic cloud-gaming access conditions are enforced using Linux Traffic Control (NetEm), which constrains the effective link rate to 2–25 Mbps to emulate scenarios ranging from low-bandwidth mobile access to typical broadband. At each constrained operating point, GameLab's rate controller sets the encoder target accordingly, yielding reproducible bandwidth-limited conditions. Because GameSR operates on decoded RGB frames downstream of the transport and decoding stack, network effects influence GameSR through the quality of the decoded input stream, while the SR model itself remains unchanged. This enables a fair comparison among WebRTC-1080p, LR+Bicubic, and LR+GameSR under identical bitrate-constrained end-to-end conditions.

We compare three 1080p-display configurations: (i) **WebRTC-1080p** (baseline streaming at 1080p), (ii) **LR+Bicubic** (stream 540p for ×2 / 360p for ×3 and upscale with bicubic), and (iii) **LR+GameSR** (same LR streams upscaled with GameSR). Bicubic is a strong classical baseline (many pipelines rely on interpolation scalers, often bilinear; bicubic is a common lightweight upgrade).

Across 2–25 Mbps traces, using standard VMAF targets [69], [76], [77], the graphs (Figures 6a and 6b) show that achieving *good* quality (VMAF≥80) in CS2 requires 8.2 Mbps with GameSR (×2) versus 12.5 Mbps with bicubic and 16.0 Mbps with WebRTC-1080p (34.6% and 48.9% savings), while OW2 requires 8.1 Mbps with GameSR versus 14.4 Mbps (bicubic) and 15.8 Mbps (WebRTC-1080p), yielding 43.7% and 48.4% savings. For the *excellent* target (VMAF≥90), GameSR (×2) reaches 90 at 19.4 Mbps (CS2) and 18.4 Mbps (OW2), whereas neither bicubic nor WebRTC-1080p reaches VMAF≥90 within our tested range up to 25 Mbps (max VMAF: 84.2/87.4 in CS2 and 83.5/88.4 in OW2 for bicubic/WebRTC-baseline).

We extend this evaluation to two additional titles: **FC24** and **TF2** (Team Fortress 2). Notably, neither title supports DLSS or FSR—FC24 has no upscaler integration in its Frostbite build, and TF2 runs on the legacy Source engine—making them representative cases where GameSR is the only available upscaling solution. As shown in Figures 6c and 6d, GameSR yields substantial bandwidth savings for both titles. For FC24, GameSR (×2) reaches VMAF≥80 at 8.2 Mbps versus 16.2 Mbps for bicubic and 17.7 Mbps for WebRTC-1080p (49.0% and 53.4% savings, respectively); no configuration reaches VMAF≥90 within the tested range, reflecting FC24's high visual complexity (fast camera motion, detailed grass and

crowd textures). For TF2, GameSR ($\times 2$) achieves VMAF$\geq$80 at just 3.7 Mbps compared to 5.6 Mbps (bicubic) and 8.5 Mbps (WebRTC-1080p), yielding 33.4% and 55.7% savings. TF2 also reaches the *excellent* target (VMAF$\geq$90) at 7.1 Mbps, whereas WebRTC-1080p requires 14.1 Mbps (50.0% savings) and bicubic never reaches this threshold (max VMAF: 84.9). The higher absolute VMAF values for TF2 reflect its stylized, lower-complexity visuals, which compress and reconstruct more efficiently.

Finally, Figure 6f shows that although GameSR adds $\sim$4–5 ms of client-side SR compute, it reduces end-to-end latency by $\sim$**19% on average** (across 8/16/24 Mbps), driven by lower encoder, transport, and decoding times when streaming fewer pixels at lower resolutions.

V. Conclusion

We introduced GameSR, a fast and engine-agnostic super-resolution model designed for cloud and local gaming. Unlike engine-integrated upscalers such as DLSS and FSR, GameSR requires no renderer data (motion vectors, depth buffers, or source code access), enabling deployment across both modern and legacy titles. Deployed in a real-time WebRTC cloud-gaming pipeline, GameSR reduces bandwidth by 35–56% and end-to-end latency by $\sim$19%, while maintaining high perceptual quality. Through efficient feature extraction, reparameterization, and lightweight temporal modeling via ConvLSTM, GameSR achieves $\sim$4 ms inference for $2\times$ scaling to 1080p and 7–15 ms for scaling to 4K, both within real-time budgets. Objective and subjective evaluations further show that GameSR achieves competitive no-reference perceptual quality compared to DLSS and FSR, despite operating without any engine-level information. Overall, GameSR offers a practical, deployable path toward high-quality, low-cost, and real-time cloud gaming.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant ALLRP 556311-20. During the preparation of this work, the authors used Claude (Anthropic) to improve the language and readability of the manuscript; the authors reviewed and edited all content and take full responsibility for the publication.

References

[1] Newzoo, "Newzoo's global games market report 2024," https://newzoo.com/resources/trend-reports/newzoos-global-games-market-report-2024-free-version, 2024.

[2] Igor's Lab, "NVIDIA GeForce RTX 3080 Ti FE in test: Almost an RTX 3090 but with halved memory expansion for gamers," https://www.igorslab.de/en/nvidia-geforce-rtx-3080-ti-fe-in-test-almost-an-rtx-3090-but-with-halved-memory-expansion-for-gamers/, 2021.

[3] Mezha, "PC Power Consumption in Games: How Many Watts Does Your Computer Use?" https://mezha.media/en/articles/pc-power-consumption-in-games/, 2024.

[4] E. Mills and N. Mills, "Taming the energy use of gaming computers," *Energy Efficiency*, vol. 8, pp. 865–885, 2015.

[5] Netflix, "How to control how much data netflix uses," 2022. [Online]. Available: https://help.netflix.com/en/node/87

[6] NVIDIA, "GeForce NOW System Requirements," https://www.nvidia.com/en-us/geforce-now/system-reqs/, 2025.

[7] M. Amiri, H. A. Osman, and S. Shirmohammadi, "Resource optimization through hierarchical sdn-enabled inter data center network for cloud gaming," in *MMSys '20*. ACM, 2020, pp. 166–177.

[8] M. Claypool and D. Finkel, "The effects of latency on player performance in cloud-based games," in *2014 13th Annual Workshop on Network and Systems Support for Games*. IEEE, 2014, pp. 1–6.

[9] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*. IEEE, 2012, pp. 1–6.

[10] NVIDIA, "Deep Learning Super Sampling (DLSS," in *Proceedings of the Game Developers Conference (GDC)*, 2019, available: https://web.archive.org/web/20240826000000/https://www.nvidia.com/en-us/geforce/technologies/dlss/.

[11] AMD, "AMD FidelityFX^{TM} Super Resolution (FSR," https://www.amd.com/en/products/graphics/technologies/fidelityfx/super-resolution.html, 2025.

[12] Intel, "Intel X^{e} Super Sampling (XeSS," https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/arc/technology/xess.html, 2024.

[13] T. Dong, H. Yan, M. Parasar, and R. Krisch, "RenderSR: A Lightweight Super-Resolution Model for Mobile Gaming Upscaling," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22) Workshops*, 2022, pp. 3086–3094. [Online]. Available: https://ieeexplore.ieee.org/document/9857374/

[14] S. Wu, S. Kim, Z. Zeng, D. Vembar, S. Jha, A. Kaplanyan, and L.-Q. Yan, "ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation," in *SIGGRAPH Asia 2023 Conference Papers*, ser. SA '23, 2023.

[15] S. Yang, Q. Zhu, J. Zhuge, Q. Qiu, C. Li, Y. Yan, H. Xu, L.-Q. Yan, and X. Jin, "Mob-FGSR: Frame Generation and Super Resolution for Mobile Real-Time Rendering," in *Proceedings of the ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–11.

[16] L. Xiao, S. Nouri, M. Chapman, A. Fix, D. Lanman, and A. Kaplanyan, "Neural supersampling for real-time rendering," *ACM Transactions on Graphics*, vol. 39, 2020.

[17] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," *CoRR*, vol. abs/1707.02921, 2017.

[18] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," *CoRR*, vol. abs/1704.03915, 2017. [Online]. Available: http://arxiv.org/abs/1704.03915

[19] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight image super-resolution with information multi-distillation network," in *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2019.

[20] X. Luo, Y. Xie, Y. Zhang, Y. Qu, C. Li, and Y. Fu, "LatticeNet: Towards Lightweight Image Super-Resolution with Lattice Block," in *Computer Vision – ECCV 2020*. Springer, 2020, pp. 272–289.

[21] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image Restoration Using Swin Transformer," 2021. [Online]. Available: https://arxiv.org/abs/2108.10257

[22] N. Barman, S. Zadtootaghaj, M. G. Martini, S. Möller, and S. Lee, "A comparative quality assessment study for gaming and non-gaming videos," in *2018 QoMEX*. IEEE, 2018, pp. 1–6.

[23] N. Barman, S. Zadtootaghaj, S. Schmidt, M. G. Martini, and S. Moller, "An objective and subjective quality assessment study of passive gaming video streaming," *International Journal of Network Management*, vol. 30, no. 3, p. e2054, 2020.

[24] S. Zadtootaghaj, S. Schmidt, N. Barman, S. Möller, and M. G. Martini, "A classification of video games based on game characteristics linked to video coding complexity," in *2018 16th Annual Workshop on Network and Systems Support for Games (NetGames)*. IEEE, 2018, pp. 1–6.

[25] V. J. Hodge, N. Sherkat, A. Sherkat, and H. Liu, "Win prediction in multiplayer esports: Live professional match prediction," *IEEE Trans. Games*, vol. 13, no. 4, pp. 368–379, 2021.

[26] NVIDIA, "Deep Learning Super Sampling (DLSS) Technology - Nvidia," 2022. [Online]. Available: https://www.nvidia.com/en-us/geforce/technologies/dlss/

[27] Epic Games, "Temporal super resolution in Unreal Engine," Unreal Engine 5 Documentation, 2022. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/temporal-super-resolution-in-unreal-engine

[28] NVIDIA, "NVIDIA DLSS 3.5: Ray reconstruction," Integration Guide, 2023. [Online]. Available: https://github.com/NVIDIA/DLSS/blob/main/doc/DLSS-RR%20Integration%20Guide.pdf

[29] J. Li, Z. Chen, X. Wu, L. Wang, B. Wang, and L. Zhang, "Neural super-resolution for real-time rendering with radiance demodulation," 2024. [Online]. Available: https://arxiv.org/abs/2308.06699
[30] AMD GPUOpen, "FidelityFX Super Resolution 2: Temporal Upscaling," https://gpuopen.com/manuals/fidelityfx_sdk/techniques/super-resolution-temporal/, 2025.
[31] NVIDIA, "Deep Learning Super Sampling (DLSS) SDK," https://developer.nvidia.com/dlss, 2025.
[32] ——, "RTX Games, Engines, and Apps," https://www.nvidia.com/en-us/geforce/news/nvidia-rtx-games-engines-apps/, 2025.
[33] AMD, "AMD FidelityFX Super Resolution: Supported Games," https://www.amd.com/en/products/graphics/technologies/fidelityfx/supported-games.html, 2025.
[34] SQ Magazine. (2025) Steam statistics. [Online]. Available: https://sqmagazine.co.uk/steam-statistics/
[35] A. Ignatov, R. Timofte *et al.*, "Real-time quantized image super-resolution on mobile npus, mobile ai & aim 2021 challenge: Report," 2021.
[36] ——, "Efficient and accurate quantized image super-resolution on mobile npus, mobile ai aim 2022 challenge: Report," 2022.
[37] Y. Li, K. Zhang, R. Timofte *et al.*, "Ntire 2022 challenge on efficient super-resolution: Methods and results," in *2022 IEEE/CVF CVPRW*, 2022, pp. 1061–1101.
[38] M. V. Conde, E. Zamfir, R. Timofte *et al.*, "Efficient deep models for real-time 4k image super-resolution. ntire 2023 benchmark and report," in *Proceedings of the IEEE/CVF CVPR Workshops*, 2023, pp. 1495–1521.
[39] D. Fuoli, S. Gu, and R. Timofte, "Efficient video super-resolution through recurrent latent space propagation," in *Proceedings of the IEEE/CVF ICCVW*, 2019.
[40] B. N. Chiche *et al.*, "Stable long-term recurrent video super-resolution," in *Proceedings of the IEEE/CVF CVPR*, 2022.
[41] K. C. Chan, S. Zhou, X. Xu, and C. C. 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)*, 2022, pp. 5972–5981.
[42] L. Wang *et al.*, "Structured sparsity learning for efficient video super-resolution," in *Proceedings of the IEEE/CVF CVPR*, 2023.
[43] H. Meyer *et al.*, "Super-resolution by predicting offsets: An ultra-efficient super-resolution network for rasterized images," in *Proceedings of the ECCV*, 2022.
[44] C. Zheng *et al.*, "Efficient video super-resolution for real-time rendering with decoupled g-buffer guidance," in *Proceedings of the IEEE/CVF CVPR*, 2025.
[45] Z. Zhong, J. Zhu, Y. Dai, C. Zheng, G. Chen, Y. Huo, H. Bao, and R. Wang, "FuseSR: Super Resolution for Real-time Rendering through Efficient Multi-resolution Fusion," in *ACM SIGGRAPH Asia 2023 Conference Papers*. ACM, 2023, pp. 1–10.
[46] S. Yang, Y. Zhao, Y. Luo, H. Wang, H. Sun, C. Li, B. Cai, and X. Jin, "MNSS: Neural Supersampling Framework for Real-time Rendering on Mobile Devices," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 7, pp. 4271–4284, 2023.
[47] H. Zhang, J. Guo, J. Zhang, H. Qin, Z. Feng, M. Yang, and Y. Guo, "Deep fourier-based arbitrary-scale super-resolution for real-time rendering," in *ACM SIGGRAPH 2024 Conference Papers*. ACM, 2024, pp. 1–11.
[48] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *13th USENIX Symposium on OSDI 18*, 2018, pp. 645–661.
[49] D. Baek, M. Dasari, S. R. Das, and J. Ryoo, "dcSR: practical video quality enhancement using data-centric super resolution," in *Proceedings of the 17th CoNEXT*, 2021, pp. 336–343.
[50] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "Nemo: enabling neural-enhanced video streaming on commodity mobile devices," in *Proceedings of the 26th MobiCom*, 2020, pp. 1–14.
[51] S. Jiang, Z. Han, H. Tan, X. Jiang, Y. Yang, X. Zhang, H. Ni, Y. Yang, and X.-Y. Li, "Real-time neural-enhancement for online cloud gaming," *arXiv preprint arXiv:2501.06880*, 2025.
[52] Apple Inc., "iPhone 16 Pro – Technical Specifications," 2024. [Online]. Available: https://www.apple.com/iphone-16-pro/specs/
[53] MediaTek Inc., "MediaTek Dimensity 9400 – Flagship 5G Agentic AI Platform," 2024. [Online]. Available: https://www.mediatek.com/dimensity-9400
[54] Sony Interactive Entertainment, "PlayStation 5 Specifications," 2024. [Online]. Available: https://www.playstation.com/en-us/ps5/
[55] Microsoft, "Xbox Series X – Console Specs," 2024. [Online]. Available: https://www.xbox.com/en-US/consoles/xbox-series-x
[56] S. Bhuyan, Z. Ying, M. T. Kandemir, M. Gowda, and C. R. Das, "GameStreamSR: Enabling Neural-Augmented Game Streaming on Commodity Mobile Platforms," in *Proceedings of the 51st Annual ISCA*, 2024.
[57] W. Shi, J. Caballero, F. Huszár *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," *CoRR*, vol. abs/1609.05158, 2016.
[58] J. Liu, J. Tang, and G. Wu, "Residual feature distillation network for lightweight image super-resolution," 2020.
[59] Z. Du, D. Liu, J. Liu, J. Tang, G. Wu, and L. Fu, "Fast and memory-efficient network towards efficient image super-resolution," 2022. [Online]. Available: https://arxiv.org/abs/2204.08397
[60] W. Deng, H. Yuan, L. Deng, and Z. Lu, "Reparameterized residual feature network for lightweight image super-resolution," in *2023 IEEE/CVF CVPRW*, 2023, pp. 1712–1721.
[61] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.
[62] A. Dosovitskiy, P. Fischer *et al.*, "FlowNet: Learning Optical Flow with Convolutional Networks," in *Proceedings of the IEEE ICCV*, 2015, pp. 2758–2766.
[63] X. Ying, L. Wang *et al.*, "Deformable 3d convolution for video super-resolution," *CoRR*, vol. abs/2004.02803, 2020.
[64] S. Shi, J. Gu, L. Xie, X. Wang, Y. Yang, and C. Dong, "Rethinking alignment in video super-resolution transformers," 2022. [Online]. Available: https://arxiv.org/abs/2207.08494
[65] X. Shi, Z. Gao, L. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 802–810. [Online]. Available: https://arxiv.org/abs/1506.04214
[66] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard/
[67] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE CVPR*, 2015, pp. 3431–3440.
[68] A. Lee. (2024) VirtualDub: Video capture/processing utility for Windows. [Online]. Available: https://www.virtualdub.org/
[69] Netflix Technology Blog, "VMAF: The Journey Continues," https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12, 2018.
[70] S. Ghazanfari, S. Garg, P. Krishnamurthy, F. Khorrami, and A. Araujo, "R-LPIPS: An adversarially robust perceptual similarity metric," *arXiv preprint arXiv:2307.15157*, 2023.
[71] G. Chance, A. Ghobrial, K. McAreavey, S. Lemaignan, T. Pipe, and K. Eder, "On determinism of game engines used for simulation-based autonomous vehicle verification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 20 538–20 552, 2022.
[72] M. Utke, S. Zadtootaghaj, S. Schmidt, S. Bosse, and S. Möller, "ND-NetGaming - development of a no-reference deep CNN for gaming video quality prediction," *Multimedia Tools and Applications*, vol. 81, pp. 3181–3203, 2022.
[73] D. Li, T. Jiang, and M. Jiang, "Quality Assessment of In-the-Wild Videos," in *Proceedings of the 27th ACM International Conference on Multimedia (MM '19)*, ser. MM '19, 2019, pp. 2351–2359.
[74] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://arxiv.org/abs/1711.05101
[75] J. T. Barron, "A more general robust loss function," *CoRR*, vol. abs/1701.03077, 2017.
[76] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, "Quality-aware strategies for optimizing abr video streaming qoe and reducing data usage," in *MMSys '19*. ACM, 2019, pp. 189–200.
[77] Elecard, "Interpretation of Metrics: PSNR, SSIM, VMAF," https://www.elecard.com/page/article_interpretation_of_metrics, 2023.
[78] Valve. (2024) Source 2 - Valve Developer Community. [Online]. Available: https://developer.valvesoftware.com/wiki/Source_2
[79] E. Zamfir, M. V. Conde, and R. Timofte, "RT4KSR: Real-Time 4K Image Super-Resolution Baseline (NTIRE Real-Time 4K SR Challenge," in *Proceedings of the IEEE/CVF CVPRW*, 2023, nTIRE Real-Time 4K SR Challenge baseline.
[80] ITU-T, "Recommendation P.910: Subjective video quality assessment methods for multimedia applications," International Telecommunication Union, ITU-T Recommendation, 2022.
[81] ITU-R, "Recommendation BT.500-15: Methodologies for the subjective assessment of the quality of television images," International Telecommunication Union, ITU-R Recommendation, 2023.
[82] "GameLab: An Open-Source WebRTC Cloud Gaming Testbed," https://github.com/haseebfazal/ai-testbed.