Mini-DLSS

Lightweight Temporal Video Super-Resolution

Technical Summary and Evaluation Report

Purpose. Mini-DLSS is a compact, reproducible study of learned 2× video super-resolution. It uses five low-resolution frames to reconstruct the high-resolution center frame with a bidirectional ConvGRU model.

Scope. The project is inspired by the temporal-reconstruction idea behind DLSS, but it is *not* an implementation of NVIDIA DLSS. It does not use renderer motion vectors, depth, jitter, exposure history, or proprietary components.

Headline result. On the repository's local Vimeo-derived REDS-style validation set, the final temporal checkpoint reaches **38.3265 dB PSNR-Y**, a **+1.5231 dB** improvement over bicubic upscaling. These are pipeline validation results, not official REDS benchmark results.

Model	BasicVSR-style center-frame predictor with bidirectional ConvGRU propagation
Input / output	Five LR RGB frames → one 2× HR center frame
Best checkpoint	80,000 training steps; 0.716M parameters
Deployment	PyTorch MP4 demo and fixed-window ONNX export
Report date	June 14, 2026

Executive Summary

Mini-DLSS addresses a practical gap between framewise image upscaling and temporally stable video reconstruction. A single-image super-resolution model can sharpen each frame independently, but it cannot explicitly aggregate sub-pixel information from neighboring frames and may introduce visible frame-to-frame variation. The project therefore treats temporal video super-resolution as a complete engineering pipeline: sequence-aware data loading, learned reconstruction, image and temporal metrics, checkpointed training, qualitative video output, and ONNX deployment.

The final model, `BasicVSRMini`, is deliberately small. A shared convolutional encoder extracts features from a five-frame low-resolution window. Two convolutional GRUs propagate context forward and backward. The center-frame feature and both recurrent states are fused, refined by residual blocks, and upsampled with PixelShuffle. The model predicts one center frame per window rather than an entire sequence.

The strongest reported checkpoint was selected at 80,000 steps from a 150,000-step training run. On the local validation set it improves PSNR-Y over bicubic from 36.8033 dB to 38.3265 dB and improves flow-warped tPSNR from 36.3441 dB to 36.6018 dB. ONNX Runtime CPU inference reduces measured latency from 34.941 ms/frame in PyTorch to 21.589 ms/frame on the same 240-frame, 64×64 low-resolution clip.

The evidence is encouraging but bounded. The validation set is generated from Vimeo sequences in a REDS-style directory layout, the available single-frame checkpoint is undertrained and not budget matched, and the model lacks explicit optical-flow or deformable alignment. The report therefore supports a reproducible engineering result, not a state-of-the-art or official benchmark claim.

Problem and Scope

Task

For an odd temporal window of $T = 2r + 1$ low-resolution frames, the model reconstructs the high-resolution center frame:

$$\widehat{\mathbf{y}}_c = f_\theta(\mathbf{x}_{c-r}, \ldots, \mathbf{x}_c, \ldots, \mathbf{x}_{c+r}), \qquad T = 5, \quad s = 2. \tag{1}$$

Here $\mathbf{x}_t \in [0,1]^{3\times h\times w}$ is an LR RGB frame, $\mathbf{y}_t \in [0,1]^{3\times 2h\times 2w}$ is its HR target, and f_θ is the learned temporal reconstruction model.

What the project demonstrates

- sequence manifests and deterministic temporal-window sampling;
- bicubic, single-frame, and temporal reconstruction paths;
- cropped Y-channel and RGB PSNR/SSIM evaluation;
- temporal diagnostics based on frame differences and optical-flow warping;
- reproducible checkpoints, JSON/Markdown reports, comparison media, and ONNX export.

What the project does not claim

- It is not NVIDIA DLSS and does not reproduce proprietary reconstruction logic.
- It is not integrated with a renderer or supplied with engine motion vectors.
- It is not an official REDS result or a state-of-the-art VSR benchmark submission.
- It does not yet establish a fair temporal-versus-single-frame ablation.

System Design

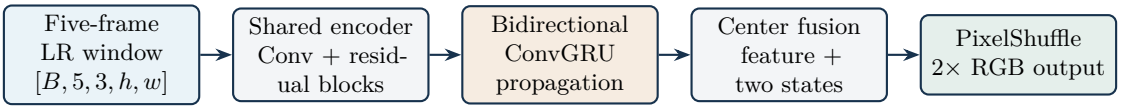

Figure 1: The `BasicVSRMini` inference path. Temporal information is propagated in both directions and fused only at the center frame.

Feature extraction and propagation

Each LR frame is encoded independently by a shared convolutional feature extractor:

$$\mathbf{e}_t = E(\mathbf{x}_t). \tag{2}$$

Forward and backward ConvGRU cells then aggregate temporal context:

$$\mathbf{h}_t^{\rightarrow} = G_f(\mathbf{e}_t, \mathbf{h}_{t-1}^{\rightarrow}), \tag{3}$$

$$\mathbf{h}_t^{\leftarrow} = G_b(\mathbf{e}_t, \mathbf{h}_{t+1}^{\leftarrow}). \tag{4}$$

At center index c, the model concatenates $[\mathbf{e}_c, \mathbf{h}_c^{\rightarrow}, \mathbf{h}_c^{\leftarrow}]$, reduces the channels with a 1×1 convolution, applies residual refinement, and reconstructs RGB with convolution plus PixelShuffle [4].

This design borrows the bidirectional propagation principle of BasicVSR [1], but omits its explicit optical-flow alignment and sequence-wide reconstruction. The omission keeps the model compact and easy to train, but limits robustness under large displacement, occlusion, and scene changes.

Training objective

The reported model uses an L_1 center-frame reconstruction loss:

$$\mathcal{L}_1(\theta) = \frac{1}{|\Omega|} \sum_{u \in \Omega} \left| f_\theta(\mathbf{x}_{c-2:c+2})_u - \mathbf{y}_{c,u} \right|. \tag{5}$$

Training uses Adam with learning rate 2×10^{-4}, batch size 4, 48 base channels, and 6 reconstruction residual blocks. The final run continued to 150,000 steps; the best validation checkpoint occurred at 80,000 steps.

Data and Evaluation Protocol

Data

Training uses a union of Vimeo-90K-style sequence data. The checked-in local evaluation set contains Vimeo-derived sequences arranged in a REDS-style HR/LR directory structure. Sequence-level manifests avoid mixing neighboring frames across scenes. During training, the loader selects aligned temporal windows, applies scale-aligned crops and flips, and returns the HR center frame as the target.

Benchmark caveat. The validation IDs and folder layout resemble REDS, but the content is Vimeo-derived. The reported values are valid for reproducing this repository's pipeline and demo; they must not be presented as official REDS benchmark results.

Image metrics

Predictions are clamped to $[0, 1]$ and cropped by two pixels on every border. The primary domain is the BT.601-style Y channel. PSNR and SSIM are computed per frame and averaged across the validation samples [6].

Temporal metrics

The report includes two output-only temporal diagnostics:

- **Difference energy**: mean absolute change between adjacent output frames. Lower is not automatically better because true scene motion also contributes.
- **tPSNR**: PSNR between adjacent outputs after Farnebäck optical-flow warping [7]. Higher generally indicates smoother motion-compensated evolution, but the score inherits optical-flow errors.

Target-relative auditing is also implemented. The final temporal model has temporal error energy 0.0105, compared with 0.0108 for bicubic and 0.0145 for the undertrained single-frame checkpoint. These diagnostics are more interpretable than raw smoothness alone because they compare predicted motion with target motion.

Results

Table 1: Local Vimeo-derived REDS-style evaluation. The single-frame model is a 300-step pipeline baseline and is not budget matched.

Method	PSNR-Y	SSIM-Y	tPSNR	Diff.	CPU ms/frame
Bicubic	36.8033	0.9601	36.3441	0.0217	0.238
Single-frame SR, 300 steps	32.3755	0.8807	33.8817	0.0183	9.105
Temporal SR, 5 frames	**38.3265**	**0.9604**	**36.6018**	**0.0201**	34.941

Primary finding. The final temporal model improves PSNR-Y by **+1.5231 dB** and tPSNR by **+0.2578 dB** over bicubic on the locked local evaluation set. It also reduces raw difference energy from 0.0217 to 0.0201.

Figure 2: Representative comparison generated by the final checkpoint. From left to right: nearest-neighbor LR visualization, bicubic, temporal SR, and HR target. The temporal output restores sharper structure than bicubic while remaining visually close to the target.

Interpretation

The PSNR result demonstrates that the temporal model learns useful information beyond deterministic interpolation on this data distribution. The final model also improves flow-warped tPSNR relative to bicubic, while its target-relative temporal error is slightly lower. Together these results support the claim that neighboring frames contribute both reconstruction quality and temporal consistency.

The single-frame row should not be used to quantify the value of temporal modeling. That checkpoint was trained for only 300 steps, whereas the final temporal model was selected from a long run. Its lower raw difference energy mainly reflects smoother, underfit outputs. A fair ablation requires matched data, optimizer, parameter count, training steps, and checkpoint selection.

Deployment and Reproducibility

Table 2: Final deployment artifacts and measured CPU latency.

Artifact	Configuration	Result
PyTorch demo	240 frames, 64×64 LR, five-frame windows	34.941 ms/frame
ONNX Runtime	CPUExecutionProvider, same clip, 10 warmups	21.589 ms/frame
ONNX graph	Dynamic batch and spatial axes; fixed temporal length	Export successful

The ONNX Runtime measurement is 38.2% lower latency than the PyTorch CPU demo on the same clip. This is a deployment comparison, not a hardware-neutral real-time claim. The current sliding-window path recomputes overlapping features and recurrent states for every output frame, leaving substantial room for streaming-state reuse.

The repository preserves the full training configuration in checkpoints and stores training and validation metrics as JSONL. The key reproducibility artifacts are:

- `configs/temporal_small.toml`;
- `results/runs/temporal_vsr_5f_small_2x/checkpoints/best.pt`;
- `results/final/tables/final_comparison.md`;
- `results/onnx/temporal_vsr_5f_small_2x.onnx`;
- final comparison images and videos under `results/final/videos/`.

Limitations and Next Steps

1. **Evaluation validity.** Rerun the locked protocol on official REDS data before making benchmark claims.
2. **Fair baseline.** Train a budget-matched single-frame model before attributing gains specifically to temporal context.
3. **Motion handling.** Add flow-guided or deformable alignment for large motion, occlusion, and fine repeated textures.
4. **Temporal objective.** Evaluate target-relative consistency or warping losses instead of relying only on center-frame L_1.
5. **Runtime.** Cache frame features and recurrent state across overlapping windows; then benchmark GPU, FP16, and quantized inference.
6. **Testing.** Add automated coverage for window indexing, metric formulas, checkpoint resume, ONNX parity, and boundary-frame padding.

Conclusion

Mini-DLSS is a credible compact implementation of temporal video super-resolution. Its main contribution is not a claim of DLSS equivalence; it is a readable, reproducible path from temporal-window data loading through training, evaluation, qualitative evidence, and ONNX deployment. The final checkpoint clearly outperforms bicubic on the locked local validation set, while the report's caveats identify exactly what remains necessary for a fair research comparison and broader deployment claim.

References

[1] K. C. K. Chan, X. Wang, K. Yu, C. Dong, and C. C. Loy, "BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond," in *CVPR*, 2021.

[2] K. C. K. Chan, S. Zhou, X. Xu, and C. C. Loy, "BasicVSR++: Improving Video Super-Resolution with Enhanced Propagation and Alignment," in *CVPR Workshops*, 2022.

[3] X. Wang, K. C. K. Chan, K. Yu, C. Dong, and C. C. Loy, "EDVR: Video Restoration with Enhanced Deformable Convolutional Networks," in *CVPR Workshops*, 2019.

[4] W. Shi et al., "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *CVPR*, 2016.

[5] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video Enhancement with Task-Oriented Flow," *International Journal of Computer Vision*, vol. 127, no. 8, 2019.

[6] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, 2004.

[7] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *Image Analysis*, 2003.

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015.