Rethinking Super-Resolution for Near-Sensor Object Detection

Lingjia Shi* Jinendra Malekar* Ramtin Zand
University of South Carolina
lingjia@email.sc.edu, jmalekar@email.sc.edu
ramtin@cse.sc.edu

Abstract

Deploying computer vision models on edge devices requires carefully balancing accuracy, latency, and data movement. In this work, we investigate super-resolution (SR) not as a means to improve visual quality, but as a near-sensor preprocessing primitive for optimizing end-to-end system efficiency in object detection pipelines. We introduce a split SR framework based on a lightweight hierarchical architecture (Swin-HIER), where the encoder is deployed near the sensor to generate compact feature representations, and the decoder operates on the edge device. This design reduces the need to transmit full-resolution images over bandwidth-limited wireless links, thereby lowering overall latency.

Through experiments on the COCO dataset using a lightweight detector (YOLO26n), we show that applying $2\times$ *SR without retraining leads to only a modest drop in detection performance (from 0.401 to 0.369 mAP@[0.5:0.95]), indicating that SR preserves competitive accuracy despite introducing a distribution shift. At the same time, system-level evaluation on NVIDIA Jetson hardware demonstrates up to* $1.26\times$ *latency reduction compared to conventional pipelines that transmit raw images. Our results suggest that, in standard-resolution regimes, SR is more effective as a system optimization tool than as a performance enhancer. This work highlights the importance of co-designing data representation, model architecture, and deployment strategy for efficient edge vision systems.*

1. Introduction

Modern computer vision systems are increasingly deployed on edge platforms such as drones, robots, and IoT devices [3, 10, 16, 17, 23, 25, 27, 28]. These systems operate under strict constraints on power, latency, and bandwidth, making data movement a dominant bottleneck [2, 6, 9, 22]. As a result, there is growing interest in on-sensor and near-sensor vision, where part of the computation is performed close to the sensing hardware [4, 18, 19, 26, 29]. A fundamental question in this paradigm is: *what representation should a sensor output?* Traditional pipelines transmit raw pixel data, often incurring significant latency when using wireless links such as WiFi or Bluetooth [13, 20, 21]. In contrast, emerging approaches aim to produce intermediate representations near the sensor that reduce data movement costs while preserving useful information for downstream tasks [5, 7, 11, 15].

In this work, we revisit super-resolution (SR) not as a tool for visual enhancement, but as a deployment-friendly intermediate representation for reducing data movement in edge vision systems. We propose to split an SR model (Swin-HIER) into encoder and decoder components, where the encoder is deployed near the sensor. The encoder produces a compact feature representation that can be transmitted efficiently (e.g., via PCIe), after which the decoder reconstructs the image for downstream processing. Compared to transmitting full-resolution images over wireless links and running the complete SR model off-device, this split design significantly reduces end-to-end latency by minimizing expensive data transfers. We evaluate this pipeline on the COCO dataset using a lightweight detector (YOLO26n [24]), applied directly to SR outputs without any additional training or fine-tuning. Our results show that SR preprocessing introduces only a modest drop in detection performance (from 0.401 to 0.369 mAP@[0.5:0.95]), indicating that the proposed system preserves competitive accuracy while enabling more efficient deployment. To complement accuracy analysis, we benchmark the latency of both the full SR model and the split encoder–decoder configuration on an NVIDIA Jetson device using TensorRT. Our measurements demonstrate that placing the encoder near the sensor and reducing wireless data transfer leads to substantial latency improvements compared to transmitting full image data. Our contributions are as follows:

- We propose a split SR pipeline with an encoder deployed near the sensor to reduce data movement overhead.
- We demonstrate that transmitting encoder features instead of full images significantly lowers end-to-end la-

*Equal contribution

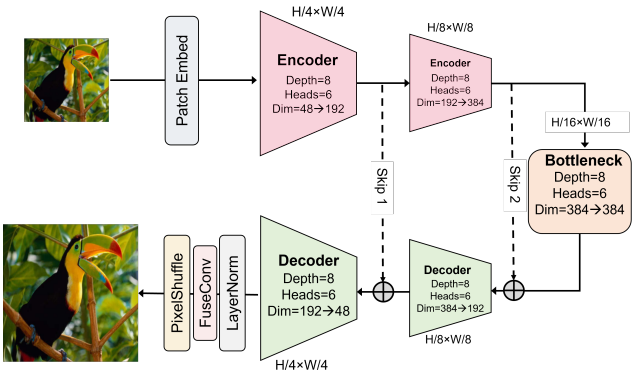

Figure 1. **Swin-HIER architecture.** We replace all attention blocks with convolutional token mixers within a hierarchical encoder–decoder. This design preserves multi-scale representations while avoiding window-induced artifacts.

tency compared to WiFi/Bluetooth-based pipelines.

- We show that SR-based preprocessing can be directly integrated with object detectors without retraining, with only a limited decrease in accuracy.
- We provide a detailed latency analysis of SR deployment on edge hardware, highlighting system-level trade-offs.

These findings provide practical insights into designing efficient near-sensor vision systems by co-optimizing computation placement and data movement.

2. Swin-HIER Architecture

2.1. Overview

Window-based self-attention (WSA) in Swin-style models restricts interactions to fixed $M \times M$ windows:

$$\mathrm{WSA}(Q,K,V) = \mathrm{Softmax}\left(\frac{QK^\top}{\sqrt{D}} + B\right)V, \quad (1)$$

which inherently limits cross-window communication and leads to regional drift, i.e., texture discontinuities and phase inconsistency across neighboring regions.

Unlike prior Swin-based SR methods [1, 8, 12, 14] that rely on shifted windows or deeper hierarchies to alleviate this issue, we remove windowed attention altogether. We propose **Swin-HIER**, a hierarchical encoder–decoder that retains token organization but replaces attention with a convolutional token mixer, achieving both improved efficiency and more stable spatial consistency. As shown in Fig. 1, our design is built on three key principles: **(1) token-aligned geometry** to eliminate boundary artifacts, **(2) hierarchical token pyramid** for scalable context aggregation, and **(3) cross-scale fusion** for consistent reconstruction.

2.2. Hierarchical Token-Aligned Architecture

Given an input image, a stride-p convolution produces a token grid of size $(H/p) \times (W/p)$ with channel dimension C.

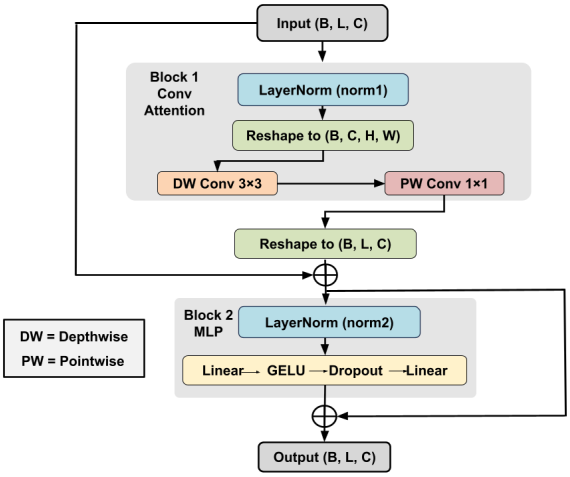

Figure 2. **Convolutional token mixer.** The proposed block replaces self-attention with depthwise and pointwise convolutions while preserving the transformer-style residual structure.

The encoder progressively downsamples tokens via *Patch Merging* while increasing channel capacity:

$$(H/4, W/4) \to (H/8, W/8) \to (H/16, W/16),$$

as illustrated in Fig. 1. The decoder mirrors this process using *Patch Expand*, with skip connections fusing encoder features at matched resolutions.

A key design choice is **exact token alignment**. We enforce symmetric padding such that all spatial resolutions are divisible by the patch stride, ensuring that encoder and decoder tokens remain strictly aligned. This eliminates boundary inconsistency, a common issue in window-based methods. The hierarchy reduces token count by $4\times$ per stage while maintaining constant per-stage complexity:

$$H_{i+1}W_{i+1}C_{i+1}^2 = H_iW_iC_i^2.$$

This enables efficient multi-scale representation without increasing computational cost.

2.3. Convolutional Token Mixer

We introduce a **convolutional token mixer** (Fig. 2) as a drop-in replacement for multi-head self-attention (MHSA). Instead of computing pairwise token interactions, we perform spatial and channel mixing directly in the feature domain. Given token input (B, L, C), features are normalized and reshaped into (B, C, H, W). A depthwise 3×3 convolution captures local spatial structure, followed by a pointwise 1×1 convolution for channel interaction. The result is projected through an MLP with residual connections.

Key insight. For image restoration, preserving local phase and structure is more critical than modeling long-range token correlations. The proposed mixer leverages

convolutional inductive bias to achieve this while avoiding the instability and overhead of attention.

Compared to WSA, our design eliminates window partitioning and cross-window inconsistency, reduces complexity by removing attention computation, and improves training stability via fully convolutional operations.

2.4. Reconstruction and Complexity

After decoding, tokens are reshaped and fused:

$$F = \mathrm{Conv}_{3\times3}(\mathrm{Reshape}(T_0)), \tag{2}$$

followed by PixelShuffle upsampling. For feature size (H, W, C), the proposed mixer has complexity:

$$\mathcal{O}(\text{ConvMix}) = HWC^2.$$

In contrast, WSA incurs:

$$\mathcal{O}(\text{WSA}) = HWC^2 + HWM^2C.$$

While both scale linearly with spatial size, WSA introduces an additional window-dependent term. Our method removes this overhead entirely. With $M{=}8$, this corresponds to $64C$ extra operations per token, making our approach significantly more efficient while preserving hierarchical expressiveness.

2.5. Implementation Details

We implement a heterogeneous pipeline that splits Swin-HIER into encoder and decoder components across the sensing and edge platforms.

Raw image data is first transferred to a near-sensor microprocessor unit (MPU) via a high-bandwidth, low-latency PCIe interface. The encoder, deployed on the MPU, converts this data into a compact intermediate representation, thereby reducing the volume of information that must be sent off-device. The resulting encoded features are then transmitted over a USB 3.0 interface to an NVIDIA Jetson AGX Orin, where the decoder stage of Swin-HIER is executed, followed by object detection using YOLOv6n [24].

3. Experiments and Analysis

3.1. Detection Performance under SR

We evaluate detection performance on the COCO dataset under SR preprocessing. We use Swin-HIER with a $2\times$ upscaling factor to transform all input images, and directly apply a pretrained YOLO26n [24] detector without any fine-tuning.

As shown in Table 1, SR preprocessing results in a moderate performance drop from 0.401 to 0.369 mAP@[0.5:0.95], with AP50 and AP75 decreasing from 0.556/0.435 to 0.540/0.394, respectively. This degradation is consistent across metrics and reflects a distribution shift introduced by SR outputs. Unlike low-resolution or degraded-input scenarios where SR can recover missing information, COCO images already contain sufficient spatial detail. In this setting, SR does not add new information but instead alters image statistics such as texture frequency, edge sharpness, and noise patterns. Since the detector is trained on natural images, these changes lead to suboptimal feature extraction and reduced detection accuracy.

We observe a larger drop in AP75 compared to AP50, suggesting that SR affects precise localization more than coarse detection. This indicates that SR-induced artifacts (e.g., slight boundary distortions or phase inconsistencies) can degrade bounding box regression accuracy even when object presence is correctly identified.

Table 1. Detection performance comparison under different input settings.

Method	Input Size	mAP@[0.5:0.95]	AP50	AP75
YOLO26n (baseline)	640×480	0.401	0.556	0.435
YOLO26n + SR ($2\times$)	1280×960	0.369	0.540	0.394

Table 2. Latency breakdown (ms) of baseline and proposed split SR pipeline.

System	Resolution		
	128×128	256×256	512×512
Baseline			
Raw Image Transfer	6.22	24.88	99.53
SR Processing	24.28	89.73	350.18
End-to-End	30.50	114.61	449.71
Proposed (Split Pipeline)			
Encoder (Near Sensor)	13.31	46.81	183.37
Feature Transfer		Low / negligible	
Decoder (Edge)	11.39	44.08	173.90
End-to-End	24.70	90.89	357.27

3.2. Latency and System-Level Evaluation

We evaluate end-to-end latency on the NVIDIA Jetson hardware, comparing a baseline pipeline (full SR on edge AI accelerator) with the proposed split pipeline (encoder on near-sensor MPU + feature transfer + decoder on edge AI accelerator). The detailed breakdown is shown in Table 2.

Let X denote the baseline latency, which includes raw image transmission and full SR processing. In the proposed design, latency is decomposed into encoder computation and feature transfer, followed by decoder execution:

$$X = T_{\text{raw}} + T_{\text{SR}}, \quad Y = T_{\text{Enc}} + T_{\text{feature}} + T_{\text{Dec}},$$

where $Y < X$ mainly due to the reduction in data movement.

Results and speedup. The proposed pipeline consistently outperforms the baseline across all resolutions. At 128×128, 256×256, and 512×512, we achieve speedups of $1.23\times$, $1.26\times$, and $1.26\times$, respectively. At 512×512, latency is reduced from 449.71 ms to 357.27 ms.

From Table 2, we observe that the baseline is dominated by SR computation and data transfer, while the proposed method redistributes computation between encoder and decoder and eliminates expensive transmission in favor of compact feature transfer. The advantage of the proposed design becomes more pronounced at higher resolutions. This is because data transmission scales with image size, whereas the encoder produces a compressed representation through hierarchical token reduction, significantly lowering transfer cost. In addition to per-frame latency reduction, the split architecture enables pipelined execution, where encoding, transfer, and decoding overlap across frames. This improves throughput and helps to hide communication latency, making the system more suitable for edge deployment.

3.3. Security and Privacy Implications

An additional advantage of the proposed split pipeline is that the transmitted representation inherently provides a level of data obfuscation. Instead of sending raw images, the system transmits intermediate encoder features that are not directly interpretable without the corresponding decoder. This makes it difficult to reconstruct or visually inspect the original scene from intercepted data, offering a lightweight form of privacy preservation.

While this should not be considered a replacement for standard encryption or secure communication protocols, the use of encoded feature representations reduces the exposure of sensitive visual information during transmission. This property is particularly beneficial in edge deployment scenarios involving surveillance, personal devices, or bandwidth-limited wireless communication, where both efficiency and data privacy are important considerations.

4. Conclusion

We investigated SR as a near-sensor preprocessing primitive for edge-based object detection. Our results on COCO show that applying $2\times$ SR without retraining leads to only a modest performance drop (0.401 to 0.369 mAP@[0.5:0.95]), indicating that SR preserves competitive accuracy despite the distribution shift. We further proposed a split SR pipeline using Swin-HIER, where the encoder operates near the sensor and the decoder runs on the edge device. This design reduces data movement by transmitting compact features instead of full images, achieving up to $1.26\times$ latency reduction on Jetson hardware. Overall, our findings suggest that, under bandwidth-constrained settings, SR can serve as an effective system-level optimization for reducing communication overhead while maintaining competitive detection performance. This highlights the importance of co-designing data representation and deployment strategies for edge vision systems.

References

[1] Xiangyu Chen, Xintao Wang, Wenlong Zhang, Xiangtao Kong, Yu Qiao, Jiantao Zhou, and Chao Dong. Hat: Hybrid attention transformer for image restoration. *arXiv e-prints*, pages arXiv–2309, 2023. 2

[2] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1):127–138, 2016. 1

[3] Ruth Cordova-Cardenas, Daniel Amor, and Álvaro Gutiérrez. Edge ai in practice: A survey and deployment framework for neural networks on embedded systems. *Electronics*, 14(24):4877, 2025. 1

[4] Gourav Datta, Zeyu Liu, Md Abdullah-Al Kaiser, Souvik Kundu, Joe Mathai, Zihan Yin, Ajey P Jacob, Akhilesh R Jaiswal, and Peter A Beerel. In-sensor & neuromorphic computing are all you need for energy efficient computer vision. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023. 1

[5] Amir Erfan Eshratifar, Amirhossein Esmaili, and Massoud Pedram. Bottlenet: A deep learning architecture for intelligent mobile cloud computing services. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2019. 1

[6] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016. 1

[7] Maruf Hassan, Steven Davy, Owais Bin Zuber, and Nouman Ashraf. Spikebottlenet: Spike-driven feature compression architecture for edge-cloud co-inference. In *Intelligent Systems Conference*, pages 342–355. Springer, 2025. 1

[8] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6185–6194, 2023. 2

[9] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pages 10–14. IEEE, 2014. 1

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[11] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45 (1):615–629, 2017. 1

[12] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1844, 2021. 2

[13] Yating Liu, Xiaojie Wang, Zhaolong Ning, MengChu Zhou, Lei Guo, and Behrouz Jedari. A survey on semantic communications: Technologies, solutions, applications and challenges. *Digital Communications and Networks*, 10(3):528–545, 2024. 1

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 2

[15] Yoshitomo Matsubara, Matteo Mendula, and Marco Levorato. A multi-task supervised compression model for split computing. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4913–4922. IEEE, 2025. 1

[16] Mohammadreza Mohammadi, Sheng-En Huang, Titon Barua, Ioannis Rekleitis, Md Jahidul Islam, and Ramtin Zand. Caveline detection at the edge for autonomous underwater cave exploration and mapping. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 1392–1398. IEEE, 2023. 1

[17] Mohammadreza Mohammadi, Adnan Abdullah, Aishneet Juneja, Ioannis Rekleitis, Md Jahidul Islam, and Ramtin Zand. Edge-centric real-time segmentation for autonomous underwater cave exploration. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 1404–1411. IEEE, 2024. 1

[18] Mohammadreza Mohammadi, Mehrdad Morsali, Sepehr Tabrizchi, Brendan Reidy, Arman Roohi, Shaahin Angizi, and Ramtin Zand. Pixelprune: Optimizing aiot vision systems via in-sensor segmentation and adaptive data transfer. In *Proceedings of the Great Lakes Symposium on VLSI 2025*, pages 312–319, 2025. 1

[19] Brendan Reidy, Sepehr Tabrizchi, Mohammadreza Mohammadi, Shaahin Angizi, Arman Roohi, and Ramtin Zand. Hirise: High-resolution image scaling for edge ml via in-sensor compression and selective roi. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024. 1

[20] Mahadev Satyanarayanan. The emergence of edge computing. *computer*, 50(1):30–39, 2017. 1

[21] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016. 1

[22] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017. 1

[23] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[24] Ultralytics. Yolo26: Real-time object detection and image segmentation model. https://github.com/ultralytics/ultralytics, 2025. Accessed: 2026-04-10. 1, 3

[25] Xubin Wang and Weijia Jia. Optimizing edge ai: A comprehensive survey on data, model, and system strategies. *arXiv e-prints*, pages arXiv–2501, 2025. 1

[26] Wenqiang Xu, Zhenjun Yu, Han Xue, Ruolin Ye, Siqiong Yao, and Cewu Lu. Visual-tactile sensing for in-hand object reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8803–8812, 2023. 1

[27] Yiwen Xu, Tariq M Khan, Yang Song, and Erik Meijering. Edge deep learning in computer vision and medical diagnostics: a comprehensive survey. *Artificial Intelligence Review*, 58(3):93, 2025. 1

[28] Yin Zhang, Chi Jiang, Binglei Yue, Jiafu Wan, and Mohsen Guizani. Information fusion for edge intelligence: A survey. *Information Fusion*, 81:171–186, 2022. 1

[29] Yue Zhou, Jiawei Fu, Zirui Chen, Fuwei Zhuge, Yasai Wang, Jianmin Yan, Sijie Ma, Lin Xu, Huanmei Yuan, Mansun Chan, et al. Computational event-driven vision sensors for in-sensor spiking neural networks. *Nature Electronics*, 6 (11):870–878, 2023. 1