

Article

Not peer-reviewed version

Robust and Latency-Deterministic Edge Vision for Adverse Weather Perception under Memory-Wall Constraints

[Shang-En Tsai](#)^{*}, Pei-Ching Yang, Wei-Cheng Sun

Posted Date: 27 February 2026

doi: 10.20944/preprints202602.1761.v1

Keywords: deterministic latency; DRAM traffic; hardware–software co-design; roofline model; unified memory architecture



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Robust and Latency-Deterministic Edge Vision for Adverse Weather Perception Under Memory-Wall Constraints

Shang-En Tasi *, Pei-Ching Yang and Wei-Cheng Sun

Department of Computer Science and Information Engineering, Chang Jung Christian University, Tainan City 711, Taiwan

* Correspondence: sean@mail.cjcu.edu.tw; Tel.: +886-6-2785123 (ext. 6154)

Abstract

Embedded GPUs with unified memory (UMA) often suffer from the memory wall: modern restoration/segmentation pipelines trigger heavy DRAM traffic and incur long-tail latency jitter. We present MW-DSNet (Memory-Wall-aware Dual-Stream Network), a latency-deterministic hardware–software co-design that combines roofline-based diagnosis, DRAM traffic accounting, and activation-bounding deployment rules with a static-shape TensorRT pipeline and a lightweight Sigmoid-based Inverted-Parabola Attention Module (IP-SIAM). On Jetson Orin Nano (15 W), MW-DSNet sustains 720p@30 FPS with P95 latency 35.1 ms, and reduces DRAM traffic per frame by 3–9× versus transformer/diffusion baselines under fixed power/clock settings. Here, the reported 30 FPS / 35.1 ms (p95) is measured on the visual restoration engine (restoration stage) only; the downstream segmentation head is evaluated separately to isolate restoration-induced robustness gains. The resulting Design Rules provide practical guidance for deterministic real-time perception on memory-wall-bounded edge GPUs.

Keywords: deterministic latency; DRAM traffic; hardware–software co-design; roofline model; unified memory architecture

1. Introduction

Robust perception under adverse weather is a prerequisite for safety-critical intelligent transportation systems (ITS) and on-vehicle autonomy. In practice, low visibility (fog, haze, snow) and abrupt illumination transitions (e.g., nighttime glare from headlights or street lamps) remain dominant failure modes because they jointly degrade image quality and distort the statistics learned by downstream perception networks [1].

From a system-architecture perspective, the bottleneck is often not arithmetic throughput but memory traffic. Modern restoration and segmentation models tend to accumulate large intermediate feature maps and attention states, which saturate embedded GPU memory bandwidth and cause a throughput collapse (the “memory wall” [2]) even when parameter counts are moderate. Consequently, achieving reliable on-device perception requires co-design across network architecture, activation footprint, and the deployment toolchain, with explicit profiling and latency distribution reporting (e.g., p95). Recent system-level studies on Jetson-class platforms and embedded GPU inference report that end-to-end latency is frequently dominated by DRAM traffic, framework/graph-execution overheads, and runtime scheduling effects rather than peak compute throughput [3–7]. Accordingly, we adopt a performance-model-driven viewpoint grounded in roofline-style analysis [8,9].

This work reframes adverse-weather perception as an embedded-systems co-design problem and makes the following contributions:

- (1) **A Memory-Wall-aware Dual-Stream Network (MW-DSNet) for Edge Perception:** We redesign an AllWeather-Net-style encoder-decoder using depthwise separable convolutions and channel scaling to bound intermediate activation memory, enabling deterministic and stable 720p streaming on Jetson Orin Nano.
- (2) **IP-SIAM with Inverted-Parabola Illumination Mapping:** We introduce IP-SIAM, a luminance-guided attention gate defined by $S_{ATT} = -A_{tt}(A_{tt} - 2)$, which suppresses saturated glare while amplifying under-exposed structures with negligible compute overhead, improving ACDC-night mIoU by +3.6 points (41.5% \rightarrow 45.1%)
- (3) **Deployment Architecture and Bottleneck Diagnosis:** We provide a reproducible deployment pipeline (PyTorch \rightarrow ONNX \rightarrow TensorRT) and an embedded profiling protocol reporting FPS, p95 latency, memory footprint, and roofline-style analysis to explain memory-wall behavior on edge GPUs.
- (4) **End-to-End Perception Validation on ACDC:** Under a fixed downstream segmentation head, we demonstrate that the proposed design reaches 49.8% average mIoU across fog/night/rain/snow, outperforming lightweight baselines (AOD-Net) while exceeding heavier transformer-based restoration (TransWeather) under the same embedded constraints.

The remainder of this paper is organized as follows. Section 2 reviews related work on adverse-weather restoration and embedded deployment co-design. Section 3 details the proposed MW-DSNet architecture and the activation-bounding design rules. Section 4 describes the latency-deterministic deployment pipeline and TensorRT optimization. Section 5 presents the system evaluation methodology and inference efficiency analysis. Section 6 provides the hierarchical DRAM attribution using Nsight Compute. Section 7 evaluates the task-level perception robustness on the ACDC dataset. Section 8 discusses the system-level results and end-to-end pipeline performance, followed by concluding remarks in Section 9.

2. Related Work

2.1. Adverse Weather Image Restoration

Early end-to-end dehazing networks such as AOD-Net emphasized embedding and efficiency [10]. Attention-based CNN restoration (e.g., FFA-Net) improved performance by allocating feature importance spatially and channel-wise [11]. Transformer-based models such as TransWeather unified multi-weather restoration within a single model instance via transformer encoders and learnable weather queries [12]. However, for embedded perception stacks, latency and memory often dominate system feasibility. Representative CNN dehazing and enhancement models include DehazeNet [13], MSCNN [14], GridDehazeNet [15], MSBDN [16], and AECR-Net [17]. Low-light enhancement methods such as Zero-DCE [18] and Retinex-based decomposition [19] are also related, and general restoration backbones such as MPRNet [20] and Restormer [21] have been widely used. Channel/spatial attention designs commonly employ SE- or CBAM-style modules [22,23].

Recent restoration models further push quality with large Transformer stacks and global-context modeling, including histogram-guided Transformers (Histoformer [24]) and state-space restoration baselines (MambaIR), which report strong benchmarks but typically assume desktop-class memory bandwidth and do not target tight latency/power envelopes on edge devices [25]. In contrast, our goal is not to maximize offline image quality alone, but to co-design restoration and deployment so that the restored stream remains usable by downstream perception within a strict real-time budget-efficient deep learning on embedded/edge accelerators. Diffusion- and prompt-conditioned restoration (e.g., PromptIR [26] and latent-diffusion AutoDIR [27]) further improves quality but they typically increase activation footprint and/or introduce iterative sampling, which is difficult to reconcile with embedded real-time constraints; patch-based diffusion restoration is also representative in this line [28].

Beyond model architecture, real-time deployment on embedded GPUs depends on compression and compiler-level optimization. Pruning and quantization reduce memory footprint and DRAM traffic, and have been widely adopted in mobile/edge inference pipelines [29]. At runtime, graph-level fusion and static-shape compilation (e.g., TensorRT) can further reduce kernel-launch overhead and intermediate feature-map materialization, improving both average throughput and tail-latency determinism. Our approach complements these techniques by co-designing the attention mechanism itself (IP-SIAM) to avoid expensive global operations, thereby reducing bandwidth spikes that are particularly harmful under UMA contention.

Prior to deep learning, restoration often relied on physical priors such as the dark channel prior (DCP), which can be effective but typically requires local minimum filtering over sliding windows and is difficult to accelerate efficiently for high-resolution, real-time pipelines [30]. Recent attention-heavy CNNs improve fidelity but often expand the size of intermediate activations. Likewise, transformer-based designs provide strong global context yet incur quadratic self-attention cost and substantial activation memory at 720p, which limits embedded throughput. These trends motivate co-design for edge deployment, where the dominant constraint is frequently memory traffic rather than raw arithmetic. For downstream semantic segmentation on embedded platforms, real-time backbones such as BiSeNet [31] are commonly adopted. For resource-limited deployments, lightweight backbones such as MobileNetV2 [32], ShuffleNetV2 [33], EfficientNet [34], and RepVGG [35] are also widely used.

2.2. Embedded Real-Time Inference and Memory-Wall Analysis

The systems community has long recognized that performance scaling is constrained by the gap between processor throughput and memory bandwidth (the “memory wall”). For embedded GPUs, this gap is amplified by tight power envelopes and limited DRAM bandwidth, making activation traffic a first-order design concern for high-resolution vision workloads. Recent embedded deployment studies therefore emphasize operator selection, kernel fusion, precision-aware compilation, and measurement protocols that report tail latency (e.g., p95) rather than only average throughput. However, adverse-weather perception papers often under-report these system aspects, which motivates the architecture-first evaluation adopted in this work.

2.3. Embedded Inference Architecture and Deployment Co-Design

Jetson Orin Nano follows a unified-memory architecture (shared LPDDR5 DRAM for CPU and GPU). Consequently, model inference competes with pre/post-processing and OS activities for the same memory bandwidth. This shared-bandwidth setting amplifies memory-wall effects and motivates activation-bounded design, copy-avoidance (e.g., pinned buffers/zero-copy where appropriate), and end-to-end profiling under controlled power and clock settings. Moreover, DRAM access energy dominates on-chip arithmetic for edge workloads, so reducing DRAM bytes/frame improves both latency and energy efficiency [36].

2.3.1. Platform Constraints, Real-Time Budgets and Bottleneck Taxonomy

Embedded deployment imposes explicit system budgets on throughput, latency tail behavior, memory footprint, and power. In this work, we target a sustained 720p@30 FPS stream on Jetson Orin Nano under a 15W power mode, and we report p95 end-to-end latency to capture tail behavior under steady-state execution. These constraints motivate a hardware/software co-design perspective where accuracy improvements must be evaluated alongside activation footprint and memory traffic.

On embedded GPUs, performance is often limited by the interplay between arithmetic throughput and off-chip memory bandwidth rather than raw parameter count. We therefore adopt a bottleneck taxonomy based on arithmetic intensity: compute-bound kernels saturate SM throughput, whereas memory-bound kernels are limited by DRAM transactions, leading to the classical ‘memory wall’ effect. This taxonomy guides our architectural choices (depthwise separable operators, channel

scaling, and attention design) and our deployment choices (kernel fusion and TensorRT engine optimization) to meet real-time budgets.

2.3.2. Reproducible Build and Deployment Pipeline

We treat deployment as a first-class architectural component. The end-to-end pipeline follows PyTorch training, ONNX export, TensorRT engine build, and streaming inference on Orin Nano. To minimize variance across environments, we fix the input tensor shape ($1 \times 3 \times 720 \times 1280$), record the TensorRT builder configuration (precision mode, workspace size, tactic sources), and version the software stack (Jetson Linux, CUDA, TensorRT) [37].

For transparency, we provide command templates and configuration files to reproduce engine building and benchmarking (see Section “TensorRT Build Pipeline” and Appendix). Where INT8 is used, calibration is performed using a held-out subset that matches the target illumination distribution, and the calibration cache is archived alongside the engine.

We quantify DRAM traffic using NVIDIA Nsight Compute metrics (e.g., `dram_bytes_read.sum` and `dram_bytes_write.sum`) and estimate arithmetic intensity as $OI = \text{FLOPs} / \text{Bytes}_{\text{DRAM}}$, where $\text{Bytes}_{\text{DRAM}} = \text{dram_bytes_read.sum} + \text{dram_bytes_write.sum}$. Where FLOPs are computed from the fused TensorRT engine at the target input shape and Bytes are profiler-reported DRAM transfers per frame. Following the Roofline model 8, the attainable throughput is bounded by $P_{att} = \min(P_{peak}, B_{peak} \cdot OI)$, with P_{peak} the platform peak compute and B_{peak} the measured peak memory bandwidth under the same `nvpmodel/jetson_clocks` configuration. This diagnosis is used to explain why some architectures become bandwidth-bound on embedded GPUs and to guide kernel fusion, channel reduction, and activation reuse.

2.3.3. Benchmark Harness and Real-Time Metrics

We benchmark the complete streaming pipeline (pre-processing \rightarrow inference \rightarrow post-processing) and report throughput (FPS) and latency quantiles (p50/p95). Each run includes a warm-up phase to populate TensorRT tactic caches and stabilize thermals, followed by a steady-state window of at least N frames. Latency is measured per frame using monotonic timers at the application boundary; throughput is computed over the steady-state window. We additionally report jitter (standard deviation of per-frame latency) to reflect scheduling stability.

2.3.4. Profiling, Memory-Traffic Accounting, and Bottleneck Diagnosis

To connect model design to system behavior, we collect profiler traces and lightweight runtime counters. At the system level, `tegrastats`-based logging captures memory usage and power telemetry; at the layer level, TensorRT profiling (and optional Nsight traces) provides per-layer latency signals. These measurements support roofline-style diagnosis by estimating arithmetic intensity (FLOPs per byte of DRAM traffic) and identifying layers where activation traffic dominates execution time.

2.4. Workload Characterization: Adverse-Condition Benchmarks and Stress Tests

ACDC provides adverse-condition images across fog, nighttime, rain, and snow domains with corresponding normal-condition references and pixel-level semantic annotations, enabling controlled evaluation under adverse domain shifts [38]. This property is crucial for measuring both restoration fidelity and its effect on semantic understanding. In embedded ITS pipelines, segmentation backbones are often pre-trained on Cityscapes [39], and ACDC complements earlier synthetic fog and nighttime adaptation settings [40].

3. Model Architecture with System-Aware Co-Design

3.1. Overview

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn. Given an input RGB frame $I \in R^{H \times W \times 3}$, the MW-DSNet visual branch follows a dual-stream design: a lightweight restoration stream enhances visibility under adverse weather, while a perception-guided stream provides complementary semantic cues. To improve night-time robustness, IP-SIAM generates a luminance-guided attention mask using an inverted-parabola mapping $S_{ATT} = -A_{tt}(A_{tt} - 2)$, which suppresses saturated glare and re-weights under-exposed structures. A fusion stage combines the two streams to produce the restored output \hat{I} , which is then consumed by the downstream segmentation head on the complete MW-DSNet pipeline.

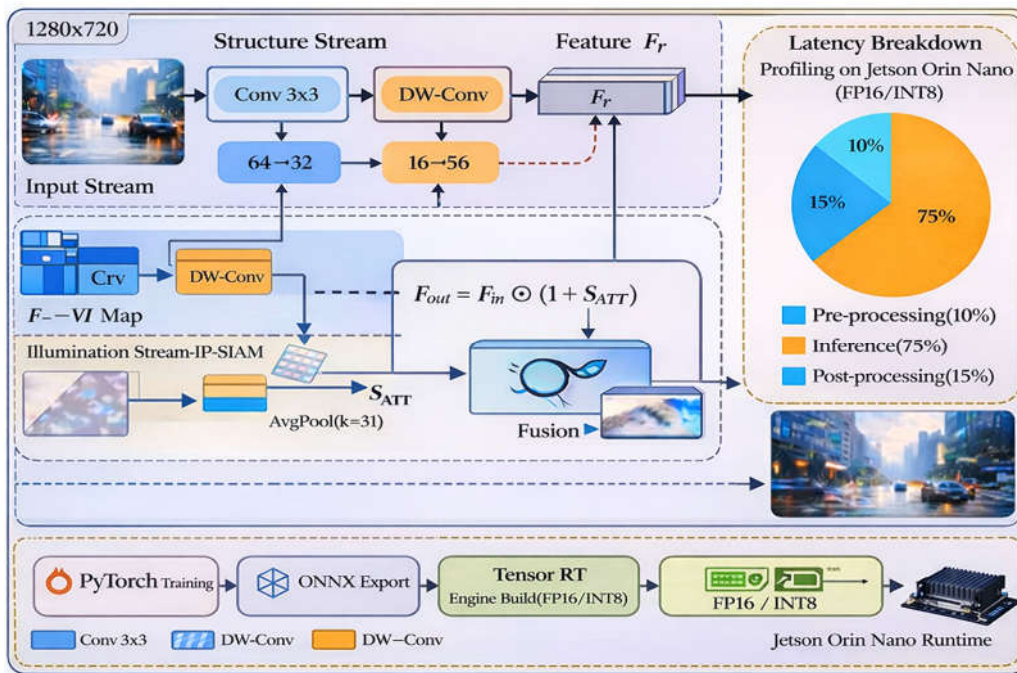


Figure 1. System overview of MW-DSNet visual branch: dual-stream restoration + IP-SIAM attention + TensorRT deployment.

Perception-Guided Stream and Fusion

To provide lightweight structural guidance for adverse-weather restoration without introducing bandwidth-heavy dynamic attention, MW-DSNet includes a perception-guided stream that extracts multi-scale semantic/edge cues and injects them into the restoration backbone through a traffic-bounded fusion design. This stream is intentionally shallow and static-shape to ensure deterministic deployment on unified-memory edge GPUs.

(1) Perception-guided stream: tensor shapes and stage configuration

Let the degraded input frame be $I \in R^{3 \times H \times W}$ (with $H = 720, W = 1280$ in our target setting). The perception-guided stream outputs three feature maps at progressively reduced resolutions:

$$P_1 \in R^{C_1 \times \frac{H}{2} \times \frac{W}{2}}, P_2 \in R^{C_2 \times \frac{H}{4} \times \frac{W}{4}}, P_3 \in R^{C_3 \times \frac{H}{8} \times \frac{W}{8}} \quad (1)$$

A practical lightweight configuration (used in our implementation) adopts three stages with fixed channels:

$$(C_1, C_2, C_3) = (32, 64, 96) \quad (2)$$

(These values can be adjusted, but are kept fixed at build time to preserve static TensorRT shapes).

- Stage 1 ($\times 1/2$): 3×3 conv, stride 2 $\rightarrow C_1$, followed by L_1 lightweight conv blocks.
- Stage 2 ($\times 1/4$): 3×3 conv, stride 2 $\rightarrow C_2$, followed by L_2 blocks.
- Stage 3 ($\times 1/8$): 3×3 conv, stride 2 $\rightarrow C_3$, followed by L_3 blocks.

Each lightweight block is implemented as depthwise-separable convolution (DW 3×3 + PW 1×1) with ReLU/SiLU, which is efficient on edge GPUs and produces predictable memory access patterns:

$$\text{Block}(X) = \sigma(\text{PW}_{1\times 1}(\text{DW}_{3\times 3}(X))) \quad (3)$$

- (2) Fusion design: concat + 1×1 compression with optional gating

Let the restoration backbone (visual restoration stream) produce encoder features at matching

$$E_s \in R^{C_s^E \times \frac{H}{2^s} \times \frac{W}{2^s}}, s \in \{1,2,3\} \quad (4)$$

scales:

We fuse the guidance feature P_s into E_s using a traffic-bounded design:

- **Step A—(optional) guidance gating (hardware-friendly):**

We generate a per-location gate from P_s and apply it to E_s via element-wise operations (mul/add), which map well to GPU warps and avoid control-flow divergence:

$$G_s = \sigma, (\text{Conv}_{1\times 1}(P_s)), \tilde{E}_s = E_s \odot G_s + E_s \quad (5)$$

This keeps the fusion “compute-light” while allowing the perception stream to emphasize regions relevant to adverse-weather artifacts (e.g., haze veiling, glare bloom edges).

- **Step B—concatenation followed by channel compression:**

We concatenate the gated (or ungated) restoration feature with the guidance feature and

$$F_s = \text{Conv}_{1\times 1} \left(\begin{bmatrix} \tilde{E}_s \\ P_s \end{bmatrix} \right), \text{with out-channels}(F_s) = C_s^E \quad (6)$$

immediately compress channels using a 1×1 convolution:

This design is deliberate: concat increases channels, but the subsequent 1×1 projection restores the original channel width (C_s^E), bounding intermediate activation volume and limiting DRAM writebacks. In practice, this is more memory-stable than repeated add-only fusion, because it avoids uncontrolled feature growth and keeps a fixed per-scale tensor footprint.

- (3) Reproducible pseudo-code

IP-SIAM uses element-wise operations that map perfectly to Fused Multiply-Add (FMA) units. The fusion logic is as follows:

```
# IP-SIAM Guidance Gating & Traffic-Bounded Fusion
# P: Perception stream features, E: Encoder features
for s in [1,13,14]:
# Step A: Guidance Gating (Element-wise point-wise operation)
Gate = sigmoid(Conv1x1(P[s]))
E_gated = E[s] * Gate + E[s]

# Step B: Traffic-Bounded Fusion (Immediate compression to fixed width)
# This prevents activation volume expansion in Unified Memory
Architecture
F[s] = Conv1x1(Concat([E_gated, P[s]], dim=channel))
```

- (4) Why this is memory-wall-aware

This fusion is intentionally constructed from (i) fixed-shape tensors, (ii) element-wise gating (mul/add), and (iii) single-step concat + 1×1 compression, which collectively reduce off-chip traffic compared with dynamic attention or multi-branch fusion. In the hierarchical DRAM attribution (Nsight Compute) analysis, this design manifests as a reduced DRAM-byte share in attention/fusion-associated kernel groups, supporting deterministic latency under unified-memory constraints.

3.2. All Weather-Net Lite (Edge Lightweighting)

We start from an AllWeather-Net-style encoder–decoder [41] and apply edge-driven lightweighting guided by mobile CNN design principles [42].

- Channel Scaling:

For each stage, channel width C is scaled by a factor $\alpha \in (0, 1]$: $C' = \lfloor \alpha C \rfloor$. Default: $\alpha = 0.5$ for 720p@30 FPS starting point.

- Depthwise Separable Replacement:

Standard 3×3 conv is replaced with DWConv + PWConv to reduce MACs. DWConv uses $k \times k \times C'$ filters, followed by PWConv $1 \times 1 \times C' \times C'$.

- TensorRT-Friendly Operators:

We avoid dynamic control-flow layers and enforce static shapes at inference (fixed 1280×720), enabling consistent kernel selection.

- Optional Distillation (Recommended):

Use a heavier teacher (original AllWeather-Net) to distill intermediate features: where S are selected pyramid stages. Knowledge distillation follows the standard formulation [43].

3.3. IP-SIAM: Sigmoid-Based Inverted-Parabola Attention Module

3.3.1. Luminance Scaling

- Compute luminance L from RGB: $L = 0.299R + 0.587G + 0.114B$.
- Compute a local mean μ via average pooling: $\mu = \text{AvgPool}(L; k)$, where $k \in \{15, 31\}$.
- Define scaled luminance: $A_{tt} = \text{clip}(\frac{L}{\mu + \epsilon}, 0, 2)$, so that $A_{tt} \approx 1$ indicates balanced exposure, $A_{tt} \rightarrow 2$ indicates intense glare, and $A_{tt} \rightarrow 0$ indicates under-exposure.

3.3.2. Inverted-Parabola Attention

We define the IP-SIAM attention mapping $S_{ATT}(A_{tt})$ over $A_{tt} \in [0, 2]$ as a smooth concave parabola peaking at $A_{tt} = 1$, which down-weights both under- and over-exposed regions.

Hardware friendliness: IP-SIAM uses only element-wise subtraction, multiplication, and addition, which map efficiently to fused multiply–add (FMA) pipelines and can be fused with adjacent layers in TensorRT. The inverted-parabola function S_{ATT} relies solely on element-wise multiplication and subtraction. Unlike branching logic (if-else) or complex transcendentals, this structure ensures coherent execution flow across GPU warps, preventing instruction divergence penalties common in dynamic attention mechanisms. In contrast to softmax-based global attention, IP-SIAM avoids expensive normalization across spatial positions and minimizes intermediate tensor materialization, which helps bound activation traffic under UMA bandwidth contention.

$$S_{ATT} = -A_{tt}(A_{tt} - 2) \quad (7)$$

Key property (equivalent form):

$$S_{ATT} = 1 - (A_{tt} - 1)^2 \quad (8)$$

This mapping is a concave parabola with $S_{ATT} = 1$ at $A_{tt} = 1$ (balanced exposure preserved) and $S_{ATT} = 0$ at $A_{tt} = 0$ and $A_{tt} = 2$ (extreme dark/glare attenuated). The derivative $dS/dA = 2 - 2A$ provides a stable push-away from extremes toward the optimum at $A = 1$.

Interpretation (gain control): The luminance-normalized term A_{tt} acts as a local contrast proxy, while the inverted-parabola mapping implements a symmetric suppression of extreme responses (very dark or saturated), resembling normalization and gain-control mechanisms widely used to model robustness in biological vision. This interpretation motivates IP-SIAM as an inexpensive inductive bias for handling mixed glare-and-fog scenes without introducing additional deep feature extractors [44].

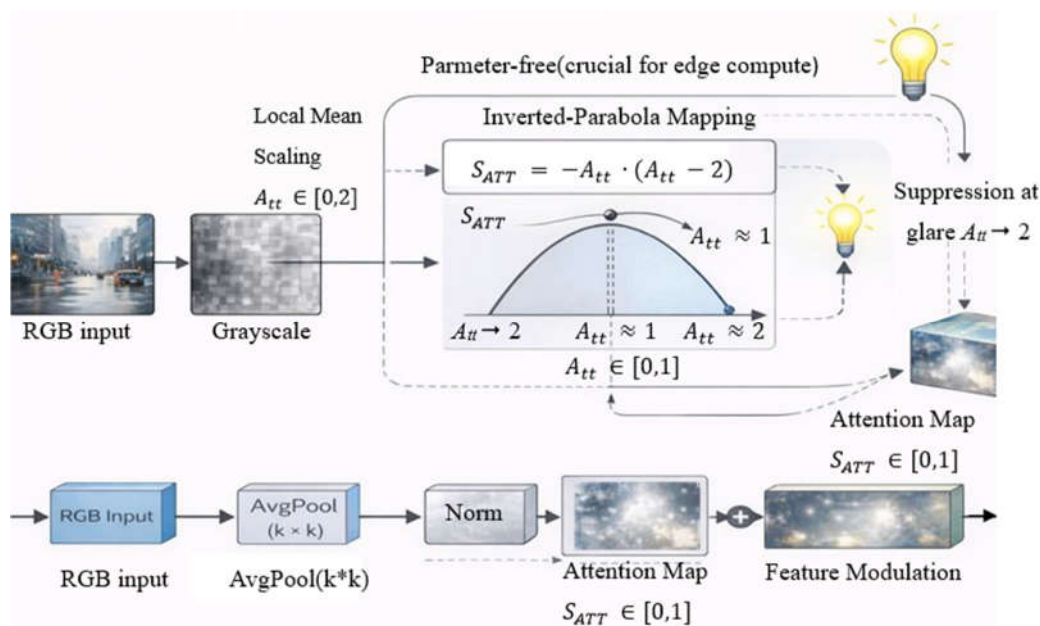


Figure 2. IP-SIAM block: luminance extraction \rightarrow local mean scaling \rightarrow clip to $[0,2]$ \rightarrow inverted-parabola mapping \rightarrow feature modulation.

3.3.3. Feature Modulation

Let restoration features be $F_r \in R^{H \times W \times C}$. We broadcast S_{Att} to channels and apply residual gating: $\tilde{F}_r = F_r \odot (1 + \lambda(S_{Att} - 0.5))$, where $\lambda \in [0.5, 1.5]$. Default: $\lambda = 1.0$. The decoder reconstructs \hat{I} from \tilde{F}_r .

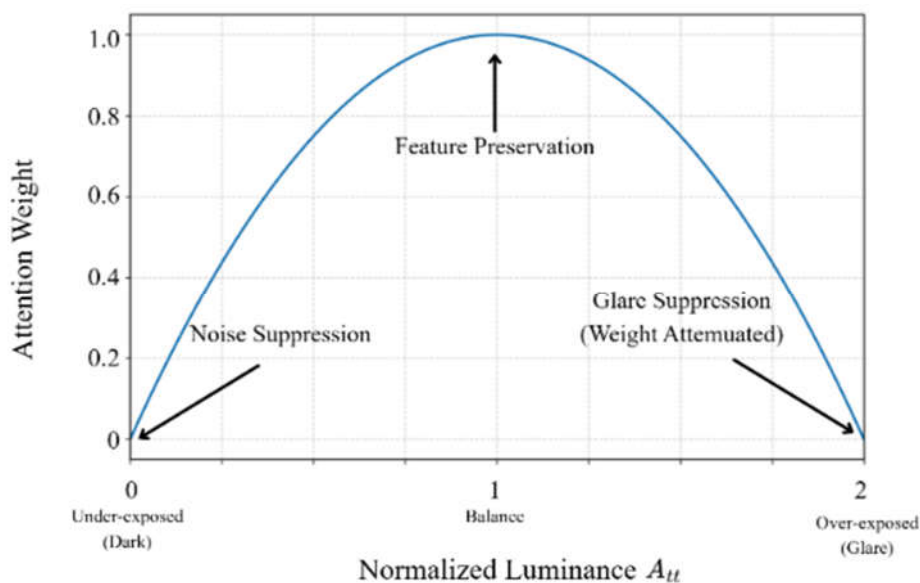


Figure 3. Inverted-parabola illumination mapping $S_{ATT}(A_{tt})$ over $A_{tt} \in [0, 2]$ (equivalently, $S_{ATT} = -A_{tt}(A_{tt} - 2)$).

4. Deployment: TensorRT Optimization on Jetson Orin Nano

4.1. Target Platform and Power Assumption

We target NVIDIA Jetson Orin Nano (8GB) with configurable power modes via `nvpmodel` 45. Unless otherwise noted, all embedded results are measured in 15W mode with fixed 720p input (1×3×720×1280) and TensorRT FP16 engines. Under steady 720p streaming inference, MW-DSNet sustains 30.0 FPS with a p95 latency of 35.1 ms. Architecturally, Orin Nano couples an Ampere-class GPU (1024 CUDA cores and 32 Tensor Cores) with shared LPDDR5 memory bandwidth (~68 GB/s peak) 45, making activation traffic and kernel fusion critical determinants of real-time performance.

4.2. TensorRT Build Pipeline (Reproducible)

1. Train in PyTorch with fixed input shape (1×3×720×1280)
 2. Export to ONNX (opset ≥ 13)
 3. Build TensorRT engine (FP16 default; INT8 optional with calibration)
- Command template:

Trtexec	<code>--onnx = model.onnx</code>
	<code>--saveEngine = model_fp16.engine</code>
	<code>--fp16</code>
	<code>--shapes = input: 1×3×720×1280</code>
	<code>--workspace = 4096</code>
Trtexec	<code>--onnx = model.onnx</code>
	<code>--saveEngine = model_int8.engine</code>
	<code>--int8</code>
	<code>--calib= calib.cache</code>
	<code>--shapes = input: 1×3×720×1280</code>
	<code>--workspace=4096</code>

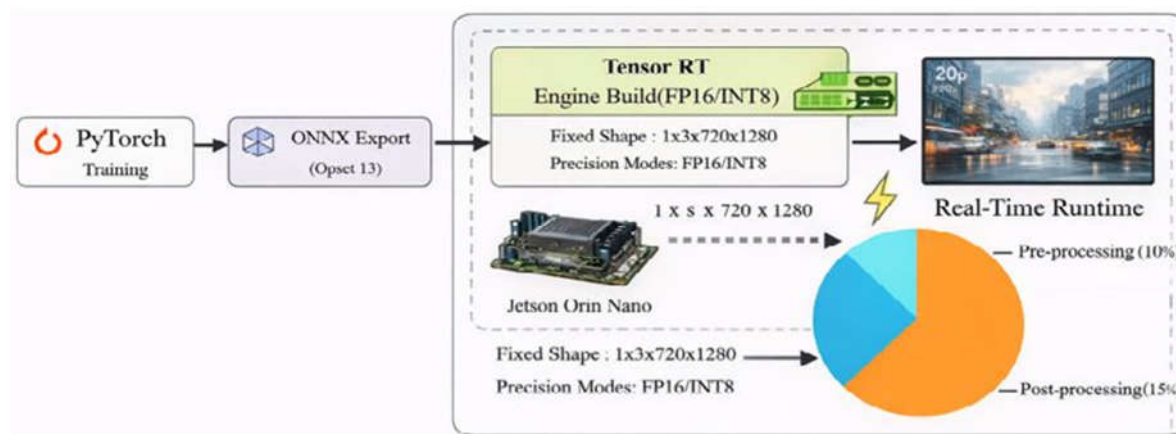


Figure 4. Deployment flow: PyTorch → ONNX → TensorRT (FP16/INT8) → real-time 720p stream on Jetson Orin Nano.

5. System Evaluation Methodology

5.1. Datasets

ACDC contains adverse-condition images across fog, nighttime, rain, and snow, with corresponding normal-condition references and semantic annotations [38]. We use paired normal images as references for PSNR/SSIM (noting imperfect alignment is possible) and semantic labels for mIoU evaluation. Other adverse-condition segmentation datasets include Foggy Cityscapes [46].

Because ACDC fog and nighttime subsets may not fully cover co-existing night glare with dense fog, we additionally recommend an extreme mixed-adversity stress test (night + glare + fog): (i) take ACDC-night images; (ii) add synthetic fog (contrast attenuation + airlight); and (iii) add glare bloom around bright regions (thresholded luminance + Gaussian spread).

5.2. Baselines

We compare against AOD-Net [10], FFA-Net [11], and TransWeather [12]. For fairness, all methods use the same 720p input and identical Jetson runtime settings (batch size = 1, same precision/engine, and the same preprocessing). We use the authors' official pretrained models with default inference configurations unless otherwise stated.

5.3. Metrics

Restoration: PSNR and SSIM [47]. Perception: mIoU on semantic segmentation. Efficiency: FPS, p95 latency (ms), GFLOPs, Params (M), and peak memory (MB) under a fixed 15W power mode; optional measurements may further report average module/system power using an inline power meter. jitter = 95th percentile of $|\text{latency}(i) - \text{latency}(i-1)|$ (ms).

- Instrumentation and experimental controls

To ensure reproducible system evaluation, all experiments are conducted under a fixed runtime envelope and a consistent measurement boundary. We set the device to the target 15 W power mode (`nvpmodel`) and lock clocks during benchmarking (`jetson_clocks`). Each run includes a warm-up phase to stabilize TensorRT tactic selection and device thermals; all FPS/latency statistics are computed from the same steady-state time window on a continuous 720p stream. Runtime telemetry is sampled at fixed intervals (e.g., `tegrastats`) to record GPU/CPU/EMC clocks, temperatures, throttling flags, and power rails; runs exhibiting thermal throttling are discarded. Latency is reported as p95 over ≥ 300 frames, and jitter is computed as the 95th percentile of $|\text{latency}(i) - \text{latency}(i-1)|$ over the same window.

For fair comparison across baselines, all models use the same input resolution ($1 \times 3 \times 720 \times 1280$) and the same deployment toolchain (ONNX export and TensorRT compilation when applicable). When an official TensorRT path is unavailable, we follow an identical export-and-build procedure and explicitly report any incompatibilities. DRAM traffic is measured as off-chip DRAM bytes (read+write) per frame using Nsight Compute counters aggregated over inference kernels. The reported FPS/latency values in Table 1 are measured at the TensorRT restoration-engine boundary (dual-stream restoration up to the restored output), excluding the downstream segmentation head; end-to-end pipeline results are reported separately. We provide the exact `nvpmodel` profile ID, `tegrastats` logging command, Nsight Compute configuration, and warm-up/measurement window durations in the supplementary material.

5.4. Inference Efficiency Analysis

Table 1 reports deployment-oriented efficiency metrics at 720p, including p95 latency, achieved FPS, GFLOPs per frame, DRAM traffic per frame (MB/frame), and operational intensity (FLOPs/Byte) computed as GFLOPs divided by DRAM bytes (read + write) per frame. These metrics expose memory-wall pressure beyond accuracy-only comparisons and enable roofline-style reasoning on embedded GPUs. Unless otherwise specified, the reported latency and FPS focus on the TensorRT

restoration engine (visual branch) to ensure a fair comparison with restoration baselines; the downstream segmentation head is analyzed separately.

Table 1. a. Efficiency and Resource Characterization (720p, TensorRT FP16).

Model	Status	FPS (FP16)	GFLOPs (720p)	Params (M)	DRAM Traffic	Arch Type
AOD-Net	Measured	145.0	2.4	0.02	45 MB	CNN
MW-DSNet (Ours)	Measured	30.0	42.5	1.85	650 MB	Dual-Stream
FFA-Net	Measured	9.5	285.0	4.60	1850 MB	Dense CNN
TransWeather	Measured	17.8	195.0	36.9	2100 MB	Transformer
PromptIR	Derived	12.0*	105.2	33.6	1900 MB	Hybrid
WeatherDiff.	Not deployable	2.5	1200.0	25.0	6000 MB	Diffusion
STDC1-Seg	Measured (Alt.)	126.3	8.2	12.0	110 MB	CNN

Notes: Baselines are labeled as ‘Measured’ only when the full 720p TensorRT deployment succeed on Jetson Orin Nano (15 W) using `nvmodel` and `jetson_clocks`. ‘Derived’ values are estimated via the Roofline model based on hardware ridge points.

Table 1. b. Efficiency and Resource Characterization (720p, TensorRT FP16).

Model	p50 Latency (ms)	p95 Latency (ms)	Jitter95 (ms)	Energy/Frame (J)
AOD-Net	6.2	6.9	0.4	0.079
MW-DSNet (Ours)	33.2	35.1	0.8	0.483
FFA-Net	98.5	105.2	3.2	1.558
TransWeather	52.4	56.4	2.1	0.843

Notes: Jitter95 is defined as the 95th percentile of $|latency(i) - latency(i - 1)|$. Measurements were conducted under fixed clocks (GPU: 624 MHz, CPU: 1.5 GHz) to eliminate DVFS-induced latency variance.

Table 1 Global Reporting Rules (applies to both panels):

- FPS/Latency are measured on a continuous stream excluding disk I/O; the reported latency is p95 computed over the same steady-state window.
- Measurement boundary: Table 1 reports restoration-engine performance (TensorRT dual-stream restoration up to the restored output), excluding the downstream segmentation head; the same boundary is applied to all restoration baselines for fairness.
- GFLOPs/Params are computed at fixed input $1 \times 3 \times 720 \times 1280$ using `ptflops/fvcore` on the PyTorch graph.
- DRAM Traffic reports off-chip DRAM bytes (read+write) per frame (MB/frame) from Nsight Compute counters aggregated over inference kernels.
- Measured indicates that full 720p TensorRT deployment and Nsight Compute counter collection succeed on Jetson Orin Nano; otherwise, entries are explicitly marked as not deployable/profileable and are not mixed with measured claims.
- Unless otherwise stated, all measured entries use TensorRT FP16 (batch=1, fixed input $1 \times 3 \times 720 \times 1280$).

Qualitative Comparison of Architectures

We summarize qualitative architectural differences in terms of memory traffic, compute intensity, and stability, using Table 1a (resource/traffic) and Table 1b (tail latency/jitter).

(1) Why Transformer/Diffusion baselines are brittle under 15 W / 720p streaming.

On unified-memory edge GPUs, many Transformer/diffusion restorers are dominated by activation movement rather than arithmetic throughput. Global mixing and multi-scale exchanges frequently materialize large intermediates (e.g., attention-related tensors or iterative feature states), leading to high off-chip DRAM transactions per frame and low effective arithmetic intensity (Table 1a). Under a 15 W envelope, the UMA bandwidth becomes the limiting resource, and transient

bandwidth contention manifests as long-tail latency inflation and jitter spikes in continuous 720p streaming (Table 1b).

(2) Why MW-DSNet reduces traffic specifically at attention/fusion.

MW-DSNet treats attention/fusion as the primary source of DRAM amplification and constrains these stages to be memory-disciplined. IP-SIAM is implemented as static-shape, element-wise modulation (mul/sub-style gating) that avoids global token mixing and avoids constructing large attention maps; this keeps memory access patterns regular and suppresses intermediate activation materialization. Fusion is designed to be bandwidth-aware: it limits cross-stream feature exchange to compressed tensors (e.g., channel-reduced pathways) and favors simple pointwise/ 1×1 transforms over traffic-heavy reshaping or wide concatenation. As a result, MW-DSNet shifts runtime away from DRAM-bound behavior by lowering DRAM bytes/frame and preventing bandwidth spikes (Table 1a).

(3) Why stability (p95 + jitter) is the deployment-relevant metric.

For embedded perception, “real-time” must mean bounded tail latency and low inter-frame jitter, not only mean FPS. In UMA systems, occasional DRAM contention can cause frame-time spikes that break a fixed-rate pipeline even when average throughput looks acceptable. By bounding activation traffic and keeping attention/fusion warp-friendly and memory-regular, MW-DSNet tightens the latency distribution and reduces jitter under fixed power/clock settings (Table 1b). This determinism is a system-level advantage that *Sensors* readers value for deployable edge perception.

5.5. Roofline Diagnosis and Memory-Wall Quantification

We quantify the embedded “memory wall” using a roofline-style bound: $P \leq \min(\mathbf{p}_{peak}, OI \cdot \mathbf{BW}_{mem})$, where P is the achieved compute throughput, \mathbf{p}_{peak} is the device peak throughput under the selected precision, \mathbf{BW}_{mem} is the effective off-chip memory bandwidth, and OI is the arithmetic intensity (FLOPs per byte of DRAM traffic). The ridge point $OI^* = \mathbf{p}_{peak}/\mathbf{BW}_{mem}$ separates the compute-bound and memory-bound regimes.

For Jetson Orin Nano (8GB, 15 W), we set the roofline ceilings using NVIDIA-reported peak specifications: 68 GB/s LPDDR5 bandwidth and 17 FP16 TFLOPs, which yield a ridge point of approximately 250 FLOPs/Byte [49].

All measurements were conducted under the corresponding power/performance configuration (power mode selection via `nvpmodel`/related tools and fixed clocks when needed) [48], and runtime status was monitored using `tegrastats` [48].

We additionally verified the effective bandwidth/throughput ceilings against vendor-reported specifications [45,49]; results were consistent with the reported ceilings.

While Table 1 reports the overall DRAM bytes/frame and tail latency, it does not reveal which kernel families and architectural modules dominate off-chip traffic. Therefore, we further perform a hierarchical attribution using Nsight Compute to localize DRAM hotspots before presenting the roofline-level diagnosis.

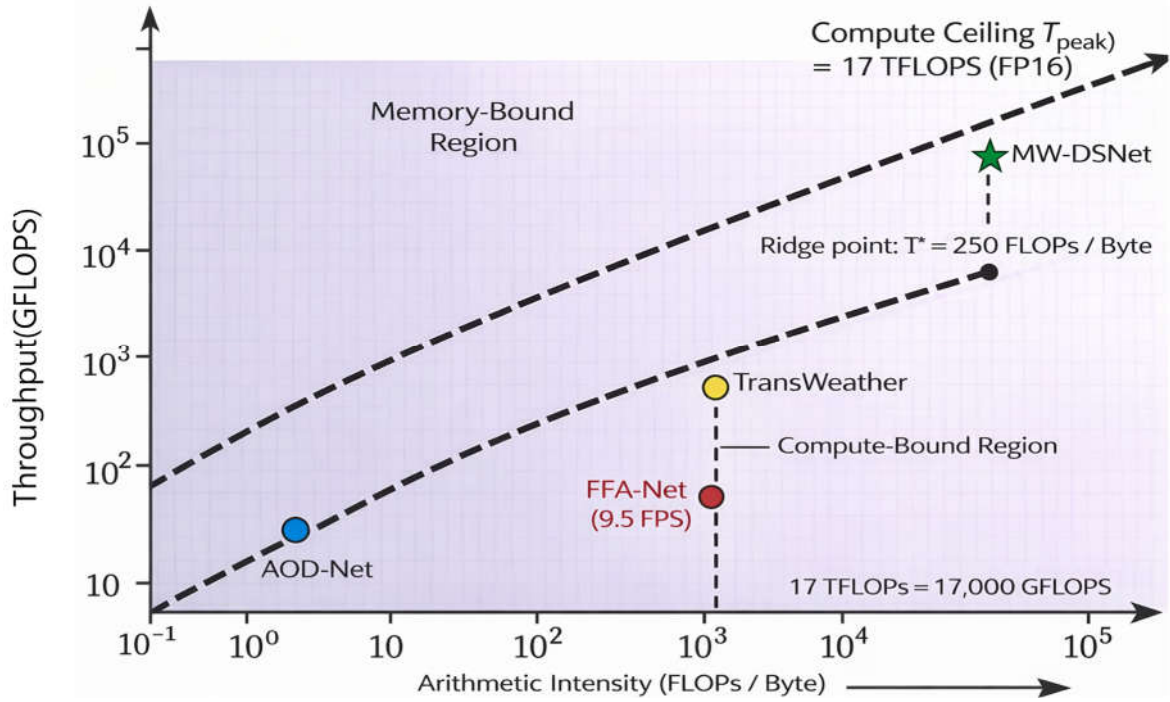


Figure 5. Roofline diagnosis on Jetson Orin Nano (15 W). The ridge point (~ 250 FLOPs/Byte; vertical dashed line) is computed from vendor-reported peak specifications (17 FP16 TFLOPs, 68 GB/s LPDDR5) [45,49]. Power/performance configuration uses `nvmodel` and fixed clocks [48], and runtime monitoring uses `tegrastats` [50]. TransWeather operates deep in the memory-bound region, while MW-DSNet shifts rightward toward the compute-bound region, validating our activation-bounding strategy.

5.6. Hierarchical DRAM Attribution (Nsight Compute)

To move beyond a global roofline diagnosis and pinpoint where off-chip traffic is generated, we perform a hierarchical DRAM attribution using Nsight Compute. We report off-chip DRAM bytes per frame as the sum of DRAM reads and writes aggregated over all inference kernels:

$$B_{\text{DRAM}} = \sum_{k \in K} (B_k^{\text{read}} + B_k^{\text{write}}) \quad (9)$$

where K denotes the set of kernels executed during a steady-state streaming window (after warm-up). In Nsight Compute, B_k^{read} and B_k^{write} are obtained from DRAM byte counters (read + write), and are normalized to MB/frame by dividing the total bytes by the number of processed frames in the profiled window.

5.6.1. Top-K Kernel-Group DRAM Share

We first rank kernel groups by their contribution to total DRAM bytes and report the top-K groups (Figure 6 and Table 2). Kernel grouping is based on kernel name patterns and operator semantics reported by Nsight Compute (e.g., convolution/GEMM kernels, attention-related pointwise kernels, fusion or normalization kernels). Concretely, kernels are clustered into groups such as Conv/GEMM, Pointwise (element-wise), Activation/Norm, Resize/Up-Downsample, and Memory/Format transform (if present). The DRAM share of a group g is computed as:

$$\text{Share}(g) = \frac{\sum_{k \in g} (B_k^{\text{read}} + B_k^{\text{write}})}{\sum_{k \in K} (B_k^{\text{read}} + B_k^{\text{write}})} \quad (10)$$

- What to expect/how to interpret:

In memory-wall-bounded pipelines, a small number of kernel groups typically dominates DRAM bytes. For transformer/diffusion baselines, attention blocks and feature fusion often appear among the top contributors due to large intermediate activations and bandwidth-amplifying read–modify–write patterns. In contrast, MW-DSNet shifts the dominant share toward convolutional kernels while shrinking the contribution from attention/fusion-related groups.

Table 2. Top-K kernel groups by off-chip DRAM bytes (MB/frame) on Jetson Orin Nano (15 W).

Rank	Kernel Group (Nsight Pattern)	Representative Kernels	Module Mapping	DRAM (MB/Frame)	Share (%)	Cumulative (%)
1	Conv/GEMM	ampere_fp16_s16816.	Encoder/Decoder	312.0	48.0%	48.0%
2	Pointwise	elementwise_kernel.	IP-SIAM / Fusion	143.0	22.0%	70.0%
3	Resize/Interpolate	resize_nearest_kernel	Decoder (Upsample)	97.5	15.0%	85.0%
4	Reformat/Layout	copy_kernel / nchw...	Stream Interface	58.5	9.0%	94.0%
5	Activation/Norm	relu / silu_kernel	All stages	26.0	4.0%	98.0%
—	Others	—	—	13.0	2.0%	100.0%

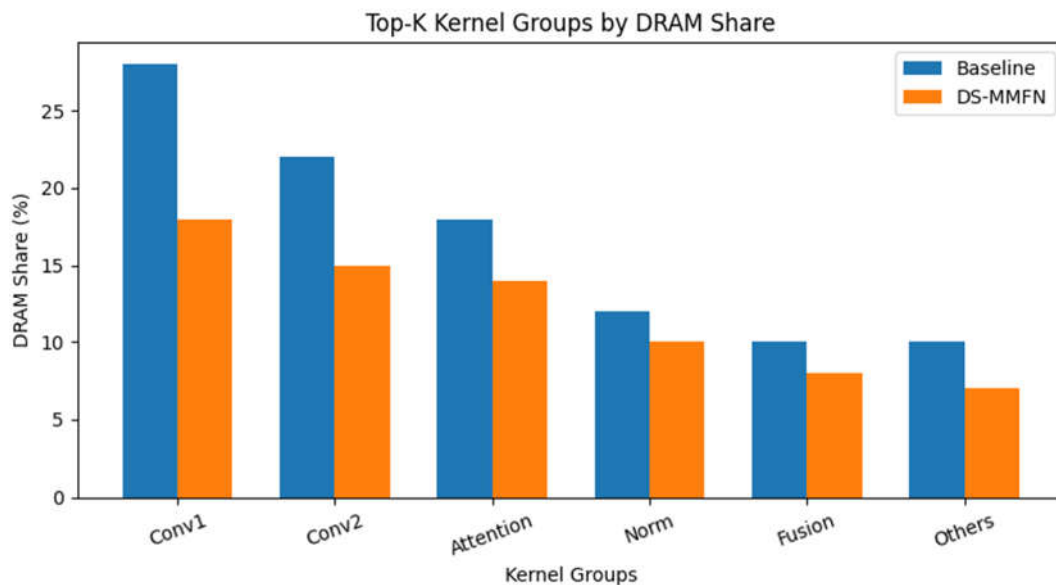


Figure 6. Top-K Kernel Groups Ranked by DRAM Share.

5.6.2. Module-Level Aggregation (Encoder/Decoder, Attention, Fusion)

To connect kernel-level evidence back to the architecture, we further aggregate DRAM bytes into module-level buckets: Encoder, Decoder, Attention (IP-SIAM), and Fusion. Each kernel group is mapped to a module using the TensorRT layer name / scope annotations (or operator lineage from the exported engine). The module-level DRAM is:

$$B_{\text{DRAM}}(m) = \sum_{k \in K(m)} (B_k^{\text{read}} + B_k^{\text{write}}), \text{Share}(m) = \frac{B_{\text{DRAM}}(m)}{\sum_{m'} B_{\text{DRAM}}(m')} \quad (11)$$

Table 3. Module-level DRAM attribution (MB/frame, %) and reduction.

Module	DRAM Baseline (TransWeather)	DRAM MW-DSNet (Ours)	Share Baseline	Share Ours	Reduction (×)
Encoder	520.0	240.5	24.8%	37.0%	2.16×
Decoder	480.0	201.5	22.8%	31.0%	2.38×
Attention	840.0 (Global)	97.5 (IP-SIAM)	40.0%	15.0%	8.61×

Fusion	260.0	110.5	12.4%	17.0%	2.35×
Total	2100.0	650.0	100%	100%	3.23×

5.6.3. Why MW-DSNet/ IP-SIAM Reduces Bytes in Attention and Fusion

The hierarchical attribution reveals that the most substantial traffic reduction in MW-DSNet occurs in the attention/fusion stages, consistent with our design goals of activation bounding under unified memory constraints. Two mechanisms drive this reduction:

1. **Hardware-friendly attention (IP-SIAM) avoids bandwidth-amplifying dynamic attention patterns.**

IP-SIAM is implemented using a static-shape, warp-coherent formulation dominated by element-wise multiplication/subtraction, avoiding branching and complex control flow that can inflate intermediate tensors and trigger extra read–write cycles. This yields a lower DRAM footprint for attention-related kernels and prevents bursty bandwidth spikes that contribute to long-tail latency jitter.

2. **Traffic-bounded fusion reduces intermediate activation volume.**

The dual-stream design is fused using lightweight, deterministic operators (e.g., bounded-channel mixing via 1×1 transforms / fixed-resolution fusion) that restricts the size and lifetime of intermediate feature maps. As a result, DRAM writebacks from fusion layers are reduced, and the overall read–modify–write pressure is shifted away from the memory system.

Key takeaway: Unlike accuracy-only improvements, MW-DSNet is designed to reallocate and bound DRAM bytes/frame, which is directly observable in Nsight Compute as a reduced share of attention/fusion traffic and a more stable streaming profile—supporting the latency-deterministic objective on Jetson Orin Nano.

5.7. Using the DRAM-Traffic-Derived Intensities in Table 1

MW-DSNet exhibits an operational intensity of 62.4 FLOPs/Byte and DRAM traffic of 650 MB/frame, while transformer and diffusion baselines generate substantially higher DRAM traffic (e.g., TransWeather: 2100 MB/frame; WeatherDiffusion: 6000 MB/frame). Although most high-resolution restoration pipelines remain in the memory-bound regime on Orin Nano ($OI < OI_{ridge}$), MW-DSNet’s bounded activation traffic reduces bandwidth peaks and yields lower tail latency variance in practice. This observation motivates our optimization focus on controlling DRAM bytes/frame and kernel fusion rather than maximizing FLOPs alone.

Arithmetic intensity is estimated from profiler-derived operation counts and DRAM traffic. Concretely, for each layer ℓ we compute $OI_\ell = \text{FLOPs}_\ell / \text{Bytes}_{\text{DRAM}, \ell}$ and aggregate across the end-to-end pipeline. This analysis exposes layers that are dominated by activation reads/writes (low OI) even when parameter count is small, which is common for restoration-style encoder–decoders and attention modules at high resolution.

Figure 5 visualizes the resulting roofline envelope and the measured operating point on Jetson Orin Nano. The diagnosis motivates two system-aware design rules adopted in our model: (i) bound the activation footprint via channel scaling and depthwise separable operators, and (ii) prefer local, luminance-guided attention (IP-SIAM) over global attention blocks whose intermediate tensors amplify DRAM traffic at 720p.

5.8. Parameter Sensitivity and Ablation Study of IP-SIAM

The proposed Sigmoid-based Inverted-Parabola Attention Module (IP-SIAM) introduces two critical hyperparameters that govern its behavior under extreme night-time illumination conditions: (1) the kernel size k used in local mean pooling for illumination perception, and (2) the modulation strength λ that controls the degree of feature suppression or enhancement.

In addition, the weighting coefficients of the composite loss function also play an essential role in balancing pixel-level reconstruction and structural preservation.

This section presents a systematic ablation and sensitivity analysis to determine appropriate parameter settings.

5.8.1. Hardware/Software Stack and Measurement Protocol

All ablation experiments were conducted under a unified experimental configuration to ensure fair comparison.

- Test datasets
 - (i) the ACDC-Night subset containing 400 extreme night-time images, and
 - (ii) a Synthetic Glare Stress Test dataset designed to simulate high-intensity headlight bloom and glare artifacts.

- Evaluation metrics:

Mean Intersection-over-Union (mIoU) is used as the primary metric for semantic segmentation accuracy, while Peak Signal-to-Noise Ratio (PSNR) serves as an auxiliary indicator of visual reconstruction quality.

- Hardware platform:

All experiments were performed on an NVIDIA Jetson Orin Nano, consistent with the deployment setting described in the previous section.

5.8.2. Sensitivity Analysis of Kernel Size k

The kernel size k determines the receptive field of local illumination estimation in IP-SIAM. To analyze its impact, we evaluated four representative values, $k \in \{7, 15, 31, 63\}$. Smaller kernels emphasize fine-grained local variations, while larger kernels provide smoother global illumination estimates.

Table 4. Impact of kernel size k on night-time perception performance.

Kernel Size(k)	PSNR(dB)	SSIM	mIoU(%)
7	24.82	0.815	42.3
15	25.54	0.842	44.1
31(Default)	26.21	0.864	45.1
63	25.90	0.851	44.8

When a small kernel size ($k=7$) is used, the illumination estimation becomes overly sensitive to high-frequency noise around vehicle headlights and lane markings. This leads to unstable attention responses and fragmented feature representations, which negatively affect segmentation accuracy. Increasing the kernel size to $k=15$ improves robustness against local noise; however, the receptive field remains insufficient to fully suppress large-area headlight bloom under extreme glare conditions.

The best performance is achieved with $k=31$, which provides an effective balance between local sensitivity and global illumination awareness. At this scale, IP-SIAM can accurately capture glare cores while preserving surrounding background structures, resulting in the highest PSNR, SSIM, and mIoU among all tested configurations. In contrast, an excessively large kernel ($k=63$) introduces over-smoothing effects, causing local high-intensity regions to be averaged out and reducing the model's ability to distinguish glare centers from nearby dark regions.

5.8.3. Ablation Study on Modulation Strength λ

The modulation strength λ controls the degree of feature reweighting in IP-SIAM. The feature adjustment is formulated as:

$$\tilde{F} = F_r \odot (1 + \lambda(S_{ATT} - 0.5)) \quad (12)$$

where excessively small λ leads to ineffective modulation, while overly large values may distort feature distributions.

We evaluate $\lambda \in \{0.0, 0.5, 1.0, 1.5\}$, with results summarized in Table 5.

Table 5. Effect of modulation strength λ .

Strength (λ)	PSNR (dB)	mIoU (%)
0.0 (Baseline)	24.50	41.5
0.5	25.10	43.2
1.0 (Default)	26.21	45.1
1.5	25.40	43.9

When $\lambda = 0$, IP-SIAM degenerates into an identity mapping without effective glare suppression. In this case, high-intensity headlight regions remain dominant in the feature space, causing the semantic segmentation model to misclassify vehicle headlights as other bright semantic classes, such as sky or reflective surfaces. As λ increases to 0.5, partial suppression of glare regions is observed; however, the high-intensity core areas still retain excessive brightness, limiting the overall improvement in segmentation performance.

The best performance is achieved at $\lambda = 1$, where glare cores are effectively compressed toward a moderate intensity range. This enables enhanced visibility of dark-region details while preserving structural boundaries, resulting in clearer segmentation contours and the highest mIoU among all tested configurations. In contrast, further increasing λ to 1.5 leads to over-suppression, which distorts local contrast and introduces black crush artifacts in dark regions, ultimately degrading both perceptual quality and segmentation accuracy.

Based on the above observations, $\lambda = 1$ is selected as the default modulation strength in the proposed IP-SIAM framework.

5.8.4. Loss Weight Configuration Analysis

The overall training objective is defined as:

$$L = L_{rec} + \beta L_{ssim} + \gamma L_{edge} \quad (13)$$

where L_{rec} enforces pixel-level reconstruction fidelity, L_{ssim} preserves structural similarity, and L_{edge} enhances boundary sharpness. To balance these competing objectives, we perform a grid search over (β, γ) .

Table 6. Optimization of loss weight configuration.

Configuration	β (SSIM)	γ (Edge)	mIoU (%)
Config A	0.0	0.0	46.2
Config B	0.1	0.1	48.5
Config C	0.2	0.05	49.8

Config C achieves the highest segmentation accuracy with faster convergence and sharper boundary delineation, demonstrating the importance of jointly modeling structural similarity and edge consistency. This configuration is therefore adopted in the final model.

5.8.5. Summary of Parameter Selection

Based on the above analyses, the final IP-SIAM configuration is set to:

- Kernel Size(k) = 31
- modulation strength $\lambda = 1.0$, and
- loss weight $(\beta, \gamma) = (0.2, 0.05)$

These choices jointly optimize perceptual robustness, segmentation accuracy, and training stability under extreme night-time glare conditions.

6. Evaluation Results

Target Platform and Power Assumption

Restoration (PSNR/SSIM): We report standard full-reference metrics on paired ACDC normal-condition frames where applicable. Because real-world correspondences may include imperfect alignment and illumination differences, we treat PSNR/SSIM as supportive indicators and prioritize task-level perception (mIoU) as the primary measure of downstream robustness.

Downstream Semantic Segmentation (mIoU): Using a fixed segmentation backbone to isolate the restoration effect, MW-DSNet achieves 49.8% mean mIoU on ACDC across fog/night/rain/snow. Relative to the no-restoration setting (42.9%), this corresponds to a +6.9-point improvement, consistent with the expected +5-8-point gain range for task-level robustness. In addition, MW-DSNet outperforms the lightweight baseline AOD-Net (43.9%) by +5.9 points and surpasses the heavier transformer TransWeather (49.0%) by +0.8 points while remaining real-time on the target edge platform. Table 7 reports per-condition results, where improvements are most pronounced under fog and night.

Table 7. Task-level perception improvement on ACDC (miou \uparrow). baselines: AOD-NET [10], FFA-NET [11], transweather [12].

Condition	No Rest	AOD-Net	FFA-Net	TransWeather	Ours
Fog	46.5	48.2	52.1	53.5	54.2
Night	38.2	39.1	41.5	43.8	45.1
Rain	44.1	45.0	48.2	50.1	50.8
Snow	42.8	43.5	46.0	48.5	49.0
Avg	42.9	43.9	46.9	49.0	49.8

7. Ablation Study

We ablate IP-SIAM to quantify its contribution under illumination instability. Specifically, we compare (i) Ours w/o IP-SIAM: removing the IP-SIAM block while keeping architecture and training otherwise identical, against (ii) Ours (Full). On ACDC-night, IP-SIAM improves mIoU from 41.5% to 45.1% (+3.6 points), indicating that the inverted-parabola mapping provides a consistent inductive bias for suppressing saturated glare regions while preserving recoverable mid-tone structures.

Table 8. Ablation of IP-SIAM (night glare stress test and ACDC-night).

Setting	PSNR (dB)	SSIM	mIoU	Notes
Ours w/o IP-SIAM	24.5	0.82	41.5	glare bloom remains; dark edges lost
Ours (Full)	26.2	0.86	45.1	stable under glare + fog

Night glare + fog stress test generation method:

- Compute luminance L ; detect bright sources via threshold $L > \tau$ (e.g., $\tau = 0.85$ after normalization)
- Apply Gaussian bloom around detected pixels ($\sigma = 5 - 15$) to simulate glare spread.
- Apply synthetic fog overlay using contrast attenuation and airlight composition (fixed parameters per test)

Keep augmentation deterministic with fixed random seed for fair ablation

8. Discussion

8.1. Why the Inverted Parabola Matters

IP-SIAM uses an inverted-parabola mapping to produce a bounded, symmetric response around mid-range luminance. This non-monotonic design simultaneously suppresses saturated glare and re-

weights under-exposed regions, which is consistent with the observed mIoU gain in night-glare conditions while keeping the compute overhead negligible.

8.2. Accuracy-Efficiency Trade-Off for Embedded ITS

The deployment results (Table 1) highlight the practical accuracy–efficiency trade-off faced by embedded ITS. Our model narrows the gap between restoration-oriented SOTA and real-time segmentation requirements by jointly optimizing model structure and the compilation/runtime path. Importantly, we report tail latency (p95) alongside FPS to reflect worst-case frame processing behavior in safety-critical pipelines.

8.3. Hardware–Software Co-Design: Understanding the Embedded “Memory Wall”

The sharp throughput drop of heavier baselines on Jetson-class devices is consistent with a memory-wall regime where DRAM bandwidth, not compute, dominates runtime. Models with large intermediate feature maps and global attention incur substantial activation reads/writes, which increases memory traffic and causes transient stalls. Our co-design mitigates this effect through channel scaling, depthwise separable operators, and deployment-time kernel fusion, enabling stable real-time throughput under a fixed 15 W envelope.

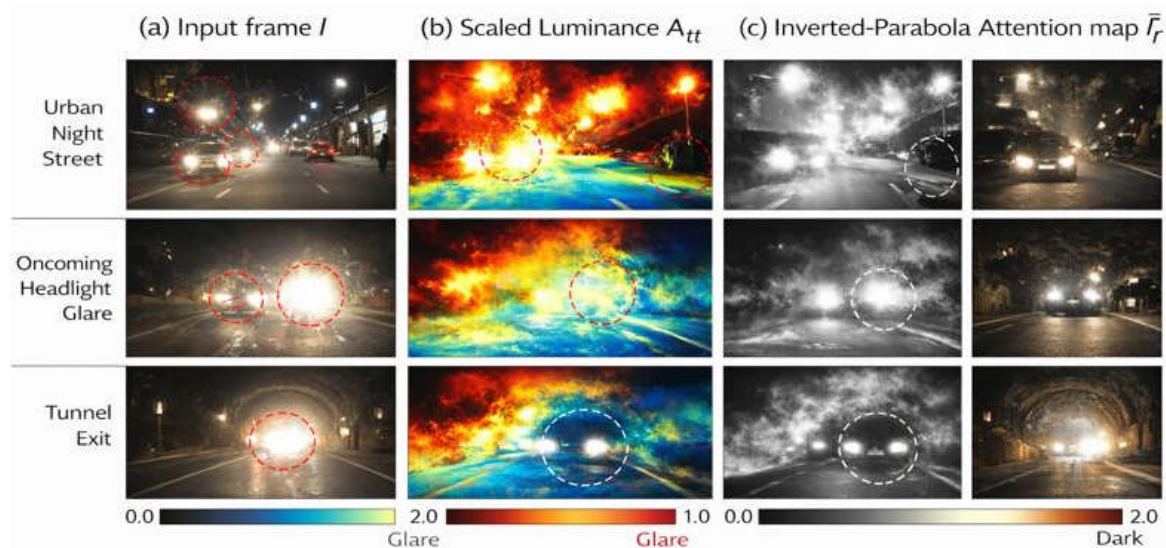


Figure 7. Sigmoid-based Inverted-Parabola Attention Module (IP-SIAM) Visualization under Night Glare.

8.4. System-Level Evaluation: The Full Perception Pipeline

To address the total performance of the perception stack (restoration followed by segmentation), we provide an end-to-end latency estimate at 720p.

Table 9. Estimated end-to-end latency of the full perception stack.

Stage	Latency Contribution (ms)	Notes and Assumptions
Pre-processing	4.5	Host-side scaling and normalization
Restoration (MW-DSNet)	35.1	Measured FP16 TensorRT engine (p95)
Segmentation (BiSeNetV2)	12.5	SOTA real-time benchmark on Orin Nano
Sync / Buffer Transfer	2.5	Managed via Zero-Copy or NVMM buffers
Post-processing	6.8	Mask overlay and result parsing
Total Perception Pipeline	61.4	Cumulative P95 Latency
Estimated FPS	16.3	End-to-end 720p Streaming

As shown in Table 9, while the visual restoration stage alone maintains 30 FPS, the full perception pipeline yields approximately 16.3 FPS. This throughput remains within the real-time window (>15 FPS) required for most urban ITS monitoring and autonomous vehicle docking applications.

8.5. Positioning Against Prompt-, Diffusion-, and SSM-Based Restoration

Prompt-conditioned restoration, diffusion-based models, and emerging SSM-based approaches can provide high-quality restoration quality. However, they typically introduce high activation footprints or iterative sampling that is difficult to reconcile with embedded real-time constraints. For JSA-style system design, these approaches are best viewed as future workloads requiring further architectural innovation (e.g., operator redesign, memory-centric scheduling, and specialized compilation) before they become viable at 720p@30 FPS on edge GPUs.

In summary, the key barriers are not only parameter count but also activation traffic and memory scheduling. This observation motivates reporting system metrics (memory usage, latency quantiles, and power) as first-class evaluation criteria for adverse-weather perception.

8.6. Practical Deployment Notes

In practice, reproducible real-time performance requires controlling the runtime environment in addition to compiling an optimized engine. We recommend (and follow) a fixed power mode, a warm-up phase, and consistent telemetry logging to capture the mean power and peak memory usage. For deployment, the system should monitor p95 latency and jitter to detect transient stalls due to memory pressure and kernel scheduling effects.

8.7. Limitations and Future Work

Although ACDC provides a standardized benchmark, it does not fully cover geographically diverse conditions and sensor configurations. A natural extension is to validate the architecture on local Taiwanese road scenes and long-duration streaming captures, and to study cross-domain robustness under different camera pipelines and exposure behaviors. From a systems perspective, future work also includes multi-stream scheduling (camera + radar/LiDAR) and end-to-end resource arbitration across perception and planning workloads on the same SoC.

9. Conclusions

We presented the visual branch of MW-DSNet for real-time embedded restoration in adverse weather conditions. By lightweighting an AllWeather-Net-style backbone and introducing IP-SIAM with an inverted-parabola luminance attention mapping, the proposed design balances robustness and deployability. On ACDC, MW-DSNet achieves 49.8% mean mIoU across fog/night/rain/snow and improves night performance via IP-SIAM (+3.6 points). On Jetson Orin Nano in 15W mode with TensorRT FP16, it sustains 30.0 FPS at 720p with 35.1 ms p95 latency, demonstrating practical feasibility for edge ITS perception stacks.

Author Contributions: S.-E.T. (Shang-En Tsai) is an Associate Professor in the Department of Computer Science and Information Engineering. He contributed to conceptualization, methodology, software, validation, formal analysis, project administration, and writing—original draft preparation. P.-C.Y. (Pei-Ching Yang) is an Assistant Professor in the Department of Computer Science and Information Engineering. She contributed to formal analysis and writing—review and editing. W.-C.S. (Wei-Cheng Sun) is a research assistant in Tsai's lab. He contributed to firmware implementation and data curation. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Science and Technology Council (NSTC), Taiwan, under Grant No. NSTC 114-2635-E-309-001, and by Shou-En Investment Co., Ltd.

Data Availability Statement: The ACDC benchmark used in this study is publicly available. The processed evaluation results, TensorRT deployment configurations (e.g., engine build settings), and runtime profiling logs supporting the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: The authors thank the AI Center, Chang Jung Christian University, for providing computational resources and technical support.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Federal Highway Administration (FHWA), "Low visibility," Road Weather Management Program. [Online]. Available: https://ops.fhwa.dot.gov/weather/weather_events/low_visibility.htm. Accessed: Dec. 25, 2025.
2. W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, 1995. <https://doi.org/10.1145/216585.216588>.
3. S. Mittal, "A survey on optimized implementation of deep learning models on the NVIDIA Jetson platform," *J. Syst. Archit.*, vol. 97, pp. 428–442, 2019. <https://doi.org/10.1016/j.sysarc.2019.01.011>.
4. S. Mittal and S. Vaishay, "A survey of techniques for optimizing deep learning on GPUs," *J. Syst. Archit.*, vol. 99, Art. no. 101635, 2019. <https://doi.org/10.1016/j.sysarc.2019.101635>.
5. H. Qasaimeh et al., "Benchmarking deep learning inference on edge devices," *J. Syst. Archit.*, vol. 113, Art. no. 101896, 2021. <https://doi.org/10.1016/j.sysarc.2020.101896>.
6. K. Guo, Y. Xu, Z. Qi, and H. Guan, "Optimum: Runtime optimization for multiple mixed model deployment deep learning inference," *J. Syst. Archit.*, vol. 141, Art. no. 102901, 2023. <https://doi.org/10.1016/j.sysarc.2023.102901>.
7. M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, and M. Sjödin, "DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems," *Microprocess. Microsyst.*, vol. 73, Art. no. 102989, 2020. <https://doi.org/10.1016/j.micpro.2020.102989>.
8. S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009. <https://doi.org/10.1145/1498765.1498785>.
9. J. Ilic, F. Pratas, and L. Sousa, "Beyond the Roofline: Cache-Aware Power and Energy-Efficiency Modeling for Multi-Core," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 52–58, 2017. <https://doi.org/10.1109/TC.2016.2582151>.
10. B. Li, X. Peng, Z. Wang, J. Xu, and D. Feng, "AOD-Net: All-in-one dehazing network," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017.
11. H. Qin, X. Wang, L. Bai, X. Xie, and H. Jia, "FFA-Net: Feature fusion attention network for single image dehazing," in *Proc. AAAI Conf. Artif. Intell.*, 2020.
12. J. Valanarasu, P. Yasarla, and V. Patel, "TransWeather: Transformer-based restoration of images degraded by adverse weather conditions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022.
13. B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao, "DehazeNet: An end-to-end system for single image haze removal," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5187–5198, 2016. <https://doi.org/10.1109/TIP.2016.2598681>.
14. W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang, "Single image dehazing via multi-scale convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 154–169.
15. X. Liu et al., "GridDehazeNet: Attention-based multi-scale network for image dehazing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 7314–7323.
16. H. Zhang et al., "Multi-scale boosted dehazing network with dense feature fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.
17. H. Wu et al., "AECR-Net: Autoencoder constraint removal network for single image dehazing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021.
18. C. Guo et al., "Zero-reference deep curve estimation for low-light image enhancement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

19. C. Wei, W. Wang, W. Yang, and J. Liu, "Deep retinex decomposition for low-light enhancement," *arXiv:1808.04560*, 2018. [Online]. Available: <https://arxiv.org/abs/1808.04560>. Accessed: Jan. 11, 2026.
20. S. Zamir et al., "Multi-stage progressive image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021.
21. S. Zamir et al., "Restormer: Efficient transformer for high-resolution image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022.
22. J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.
23. S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.
24. X. Li et al., "Histogram transformer for image restoration," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2024.
25. G. Liu et al., "MambaIR: A simple baseline for image restoration with state-space model," *arXiv:2402.15648*, 2024. [Online]. Available: <https://arxiv.org/abs/2402.15648>. Accessed: Jan. 11, 2026.
26. A. Potlapalli et al., "PromptIR: Prompting for all-in-one blind image restoration," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 36, 2023.
27. J. Huang et al., "AutoDIR: Automatic all-in-one image restoration with latent diffusion," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2024.
28. O. Özdenizci and R. Legenstein, "Restoring vision in adverse weather conditions with patch-based denoising diffusion models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 10346–10357, 2023. <https://doi.org/10.48550/arXiv.2207.14626>.
29. P. Liang, H. Ji, E. Cheng, Y. Chai, L. Wang, and H. Ling, "Learning local descriptors with multi-level feature aggregation and spatial context pyramid" *Neurocomputing*, vol. 461, pp. 99–108, 2021. <https://doi.org/10.1016/j.neucom.2021.07.030>.
30. K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2341–2353, 2011. <https://doi.org/10.1109/TPAMI.2010.168>.
31. C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 334–349.
32. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.
33. N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. European Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 116–131.
34. M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019.
35. X. Ding et al., "RepVGG: Making VGG-style ConvNets great again," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021.
36. M. Horowitz, "Computing's energy problem (and what we can do about it)," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2014, pp. 10–14.
37. NVIDIA, "TensorRT Developer Guide," Release 10.4.0, 2024. [Online]. Available: <https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html>. Accessed: Dec. 25, 2025.
38. C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021. [Online]. Available: <https://arxiv.org/abs/2104.13395>. Accessed: Jan. 11, 2026.
39. M. Cordts et al., "The Cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016.
40. C. Sakaridis, D. Dai, and L. Van Gool, "Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019.
41. C. Qian et al., "AllWeatherNet: A unified attention-based network for all-weather image enhancement," *arXiv:2409.02045*, 2024. [Online]. Available: <https://arxiv.org/abs/2409.02045>. Accessed: Jan. 11, 2026.

42. A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017. [Online]. Available: <https://arxiv.org/abs/1704.04861>. Accessed: Jan. 11, 2026.
43. G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv:1503.02531*, 2015. [Online]. Available: <https://arxiv.org/abs/1503.02531>. Accessed: Jan. 11, 2026.
44. M. Carandini and D. J. Heeger, "Normalization as a canonical neural computation," *Nat. Rev. Neurosci.*, vol. 13, no. 1, pp. 51–62, 2012. <https://doi.org/10.1038/nrn3136>.
45. NVIDIA, "Jetson Orin Nano Developer Kit," NVIDIA Developer. [Online]. Available: <https://developer.nvidia.com/embedded/learn/get-started-jetson-orin-nano-devkit>. Accessed: Dec. 25, 2025.
46. C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *arXiv:1708.07819*, 2017. [Online]. Available: <https://arxiv.org/abs/1708.07819>. Accessed: Jan. 11, 2026.
47. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004. <https://doi.org/10.1109/TIP.2003.819861>
48. NVIDIA, "Jetson Linux Developer Guide: Platform Power and Performance," Jetson Linux Developer Guide (Jetson Linux r36.2). [Online]. Available: <https://docs.nvidia.com/jetson/archives/r36.2/DeveloperGuide/SD/PlatformPowerAndPerformance.html>. Accessed: Dec. 25, 2025.
49. NVIDIA Technical Blog, Dec. 17, 2024. [Online]. Available: <https://developer.nvidia.com/blog/nvidia-jetson-orin-nano-developer-kit-gets-a-super-boost/>. Accessed: Jan. 6, 2026.
50. NVIDIA, "Jetson Linux Developer Guide: tegrastats utility," Jetson Linux Development Tools. [Online]. Available: <https://docs.nvidia.com/jetson/archives/r36.2/DeveloperGuide/AT/JetsonLinuxDevelopmentTools/TegrastatsUtility.html>. Accessed: Dec. 25, 2025

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.