Efficient Adverse-Weather Restoration for Unified-Memory Edge GPUs via Memory-Traffic-Aware Fusion

Shang-En Tsai *, Pei-Ching Yang, Wei-Cheng Sun

Posted Date: 20 April 2026

doi: 10.20944/preprints202602.1761.v2

Keywords: memory-traffic-aware fusion; DRAM traffic; roofline analysis; unified-memory edge GPUs; real-time deployment

 doi:10.20944/preprints202602.1761.v2

Article

Efficient Adverse-Weather Restoration for Unified-Memory Edge GPUs via Memory-Traffic-Aware Fusion

Shang-En Tsai *, 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

Adverse-weather image restoration is increasingly needed in edge vision systems, yet many recent methods are developed primarily for accuracy on server-class hardware rather than efficient deployment on resource-constrained platforms. This gap is particularly important for unified-memory edge GPUs, where memory traffic, activation movement, and latency variability can become major bottlenecks during inference. To address this issue, this paper presents an efficient adverse-weather restoration framework for unified-memory edge GPUs based on a memory-traffic-aware fusion strategy. Instead of relying on heavy multi-branch interaction or traffic-intensive feature aggregation, the proposed design emphasizes compact feature exchange, activation-aware computation, and hardware-friendly luminance modulation under constrained memory bandwidth. The framework is developed to preserve restoration quality while reducing unnecessary intermediate data movement, thereby improving runtime efficiency and practical deployability on edge devices. Experiments on adverse-weather restoration and downstream segmentation show that the proposed method improves task-level robustness on ACDC to 49.8% mean mIoU under a fixed segmentation head, while the restoration engine sustains 30.0 FPS at 720p with 35.1 ms p95 latency on Jetson Orin Nano. Within the scope of the ACDC benchmark and Jetson Orin Nano deployment conditions evaluated in this study, these results suggest that memory-traffic-aware fusion is a viable design principle for edge restoration systems where image quality, latency stability, and deployment efficiency must be simultaneously considered.

Keywords: memory-traffic-aware fusion; DRAM traffic; roofline analysis; unified-memory edge GPUs; real-time deployment

1. Introduction

Reliable visual input stabilization under adverse weather is a critical dependency for safety-critical intelligent transportation systems (ITS) and downstream perception pipelines. 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 destabilize the visual input consumed 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 joint design across network architecture, activation footprint, and the deployment toolchain, with explicit profiling and latency

 doi:10.20944/preprints202602.1761.v2

distribution reporting (e.g., p95) [3]. 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 [4]-[8]. Accordingly, we adopt a performance-model-driven viewpoint grounded in roofline-style analysis [10,11].

Recent edge-AI studies have further highlighted that hardware/model co-selection and real-time deployment constraints must be jointly considered when evaluating practical edge inference systems [3,9].

Unlike prior works that target full perception pipelines, this study focuses on the deployment-level behavior of image restoration under unified-memory constraints. The goal is to stabilize visual input with bounded memory traffic and predictable latency, providing a reliable upstream component for downstream tasks.

Therefore, the proposed method should be interpreted as an efficient restoration framework for unified-memory edge GPUs, emphasizing practical deployment constraints rather than end-to-end perception performance. This work reframes adverse-weather restoration on unified-memory edge GPUs as a memory-traffic-aware edge restoration design problem. The main contributions of this work are as follows:

1. **We present an efficient adverse-weather restoration framework tailored for unified-memory edge GPUs.**
   Unlike prior restoration models mainly optimized for offline accuracy or server-class execution, the proposed method is designed with edge deployment constraints in mind, especially the memory and runtime characteristics of unified-memory architectures.
2. **We propose a memory-traffic-aware fusion strategy for restoration under bandwidth-constrained edge inference.**
   The proposed fusion design aims to reduce unnecessary feature movement and intermediate activation overhead while maintaining effective multi-feature integration, making it better suited to practical on-device execution.
3. **We provide a joint evaluation of restoration quality and deployment efficiency.**
   In addition to visual restoration performance, the study examines deployment-relevant metrics on edge GPUs, showing that the proposed design offers a more favorable balance between restoration effectiveness and runtime efficiency, as demonstrated on the ACDC benchmark under the Jetson Orin Nano deployment conditions evaluated in this work.

In this work, deployment awareness extends beyond post-training conversion through ONNX export and TensorRT. The target execution constraints—Jetson Orin Nano under a 15 W power budget, unified-memory behavior, fixed 720p streaming, and bandwidth-sensitive tail latency—are incorporated directly into the model design stage. As a result, operator selection, channel width, activation footprint, and, most importantly, the fusion structure are shaped by memory-traffic efficiency considerations, enabling the final model to better match the practical requirements of unified-memory edge GPUs.

The remainder of this paper is organized as follows. Section 2 reviews related work on adverse-weather restoration and embedded deployment design. Section 3 details the proposed MW-DSNet architecture and the activation-bounding design rules. Section 4 describes the real-time 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.

 doi:10.20944/preprints202602.1761.v2

2. Related Work

2.1. Adverse Weather Image Restoration

Early end-to-end dehazing networks such as AOD-Net emphasized embedding and efficiency [12]. Attention-based CNN restoration (e.g., FFA-Net) improved performance by allocating feature importance spatially and channel-wise [13]. Transformer-based models such as TransWeather unified multi-weather restoration within a single model instance via transformer encoders and learnable weather queries [14]. However, for embedded perception stacks, latency and memory often dominate system feasibility. Representative CNN dehazing and enhancement models include DehazeNet [15], MSCNN [16], GridDehazeNet [17], MSBDN [18], and AECR-Net [19]. Low-light enhancement methods such as Zero-DCE [20] and Retinex-based decomposition [21] are also related, and general restoration backbones such as MPRNet [22] and Restormer [23] have been widely used. Channel/spatial attention designs commonly employ SE- or CBAM-style modules [24,25].

Recent restoration models further push quality with large Transformer stacks and global-context modeling, including histogram-guided Transformers (Histoformer [26]) 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 [27]. In contrast, our goal is not to maximize offline image quality alone, but to develop restoration with deployment constraints considered from the outset so that the restored stream remains usable by downstream perception within strict real-time and budget-efficient embedded operation. Specifically, this work does not aim to achieve state-of-the-art offline image quality benchmarks; instead, the primary target is deployability on unified-memory edge GPUs (demonstrated here on Jetson Orin Nano), where memory bandwidth utilization, latency stability, and runtime predictability are the dominant design constraints. Diffusion and prompt-conditioned restoration methods (e.g., PromptIR [28] and latent-diffusion AutoDIR [29]) can further improve restoration 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 of this trend [30].

Beyond model architecture, real-time deployment on embedded GPUs depends on compression and compiler-level optimization. Model compression methods such as pruning and quantization are widely used to reduce model size, off-chip memory access, and inference cost in edge deployment scenarios [31,32]. 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 stability. Our approach complements these techniques by redesigning the attention mechanism itself (IP-SIAM) to avoid expensive global operations, thereby reducing bandwidth spikes that are particularly harmful under UMA contention.

It is important to clarify that IP-SIAM is not intended as a drop-in substitute for general learned attention modules such as SE-, CBAM-, or transformer-style self-attention. Those modules typically reweight features through learned channel/spatial interactions or global token mixing, whereas IP-SIAM uses a deterministic luminance-conditioned mapping derived from local exposure statistics. The novelty in this work therefore lies less in proposing another generic attention family and more in introducing a restoration-oriented, memory-traffic-aware modulation primitive that is explicitly aligned with memory-wall-constrained edge deployment.

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 [33]. 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 deployment-aware design for edge platforms, 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 [34] are commonly adopted. For resource-limited deployments,

 doi:10.20944/preprints202602.1761.v2

lightweight backbones such as MobileNetV2 [35], ShuffleNetV2 [36], EfficientNet [37], and RepVGG [38] 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 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 [39].

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 sustained 720p streaming on Jetson Orin Nano under a 15 W power mode. We report p95 latency for the measured restoration engine, and we separately provide a pipeline-level estimate for the full perception stack. These constraints motivate a deployment-aware 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 both architectural choices (e.g., depthwise separable operators, channel scaling, and memory-traffic-aware modulation) and deployment choices (e.g., kernel fusion, fixed-shape compilation, and TensorRT engine optimization) so that model design and deployment rules are derived from the same embedded-system constraints.

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 (1×3×720×1280), record the TensorRT builder configuration (precision mode FP16, workspace size 4096 MiB, tactic sources: CUBLAS+CUBLAS_LT+CUDNN, builder optimization level 3), and version the software stack (Jetson Linux 36.2, CUDA 11.4, TensorRT 8.6, PyTorch 2.1.0, Python 3.8, ONNX opset 13) [40].

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.

 doi:10.20944/preprints202602.1761.v2

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 = FLOPs/ $\text{Bytes}_{\text{DRAM}}$, where $\text{Bytes}_{\text{DRAM}}$ = dram_bytes_read.sum + 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 10, the attainable throughput is bounded by $P_{att} = \min\left(P_{peak}, B_{peak} \cdot OI\right)$, 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 → inference → post-processing) and report throughput (FPS) and latency quantiles (p50/p95). Each run includes a warm-up phase of 100 frames to populate TensorRT tactic caches and stabilize thermals, followed by a steady-state measurement window of at least 300 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 [41]. 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 [42], and ACDC complements earlier synthetic fog and nighttime adaptation settings [43].

3. Model Architecture with Deployment-Aware Design

3.1. Overview

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 lightweight guidance stream provides complementary structural 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.

 doi:10.20944/preprints202602.1761.v2

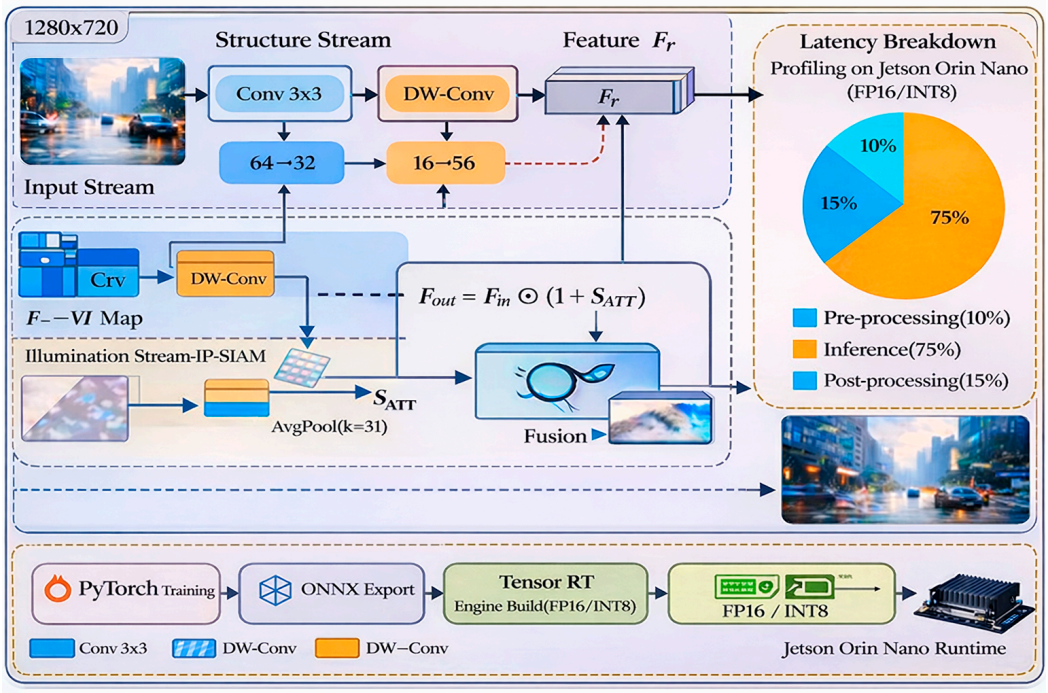

Figure 1. System overview of the proposed restoration engine and deployment pipeline, including dual-stream restoration, memory-traffic-aware fusion, and TensorRT-based execution on a unified-memory edge GPU platform.

3.1.1. Guidance Stream and Memory-Traffic-Aware Fusion

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

(1) Guidance 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 (×1/2): 3×3 conv, stride 2 →C1, followed by L1 lightweight conv blocks.
- Stage 2 (×1/4): 3×3 conv, stride 2 →C2, followed by L2 blocks.
- Stage 3 (×1/8): 3×3 conv, stride 2 →C3, followed by L3 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:

$$\text{Block}(X) = \sigma\left(\text{PW}_{1\times1}(\text{DW}_{3\times3}(X))\right) \quad (3)$$

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

 doi:10.20944/preprints202602.1761.v2

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

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

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

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

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

$$G_s = \sigma\left(\mathrm{Conv}_{1\times1}(P_s)\right), \tilde{E}_s = E_s \odot G_s + E_s \quad (5)$$

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

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

We concatenate the gated (or ungated) restoration feature with the guidance feature and immediately compress channels using a 1×1 convolution:

$$F_s = \mathrm{Conv}_{1\times1}\left(\left[\tilde{E}_s;\ P_s\right]\right), \text{with out-channels}(F_s) = C_s^E \quad (6)$$

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 efficiently to fused multiply-add (FMA) units. The fusion logic is as follows:

```
# IP-SIAM Guidance Gating & Memory-Traffic-Aware Fusion
# P: Perception stream features, E: Encoder features
for s in [1, 2, 3]:
# Step A: Guidance Gating (Element-wise point-wise operation)
Gate = sigmoid(Conv1x1(P[s]))
E_gated = E[s] * Gate + E[s]

# Step B: Memory-Traffic-Aware 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 design is memory-traffic-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 stable real-time latency under unified-memory constraints.

3.2. AllWeather-Net Lite (Edge Lightweighting)

We start from an AllWeather-Net-style encoder–decoder [44] and apply edge-driven lightweighting guided by mobile CNN design principles [45].

- Channel Scaling:

For each stage, channel width C is scaled by a factor $\alpha \in (0,1]$: $C' = \lfloor \alpha C \rceil$. 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 $1 \times 1 \times C' \times C'$.

 doi:10.20944/preprints202602.1761.v2

- 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 [46].

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

3.3.1. Luminance Estimation and Normalization

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

3.3.2. Inverted-Parabola Attention

We define the IP-SIAM attention mapping $S_{\mathrm{ATT}}(A_{tt})$ 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.

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

Key property (equivalent form):

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

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 $A = 1$.

From a methodological perspective, IP-SIAM should be interpreted as an exposure-aware modulation rule rather than a full learned attention branch. It does not introduce query-key-value interactions, global context aggregation, or additional deep gating subnetworks. Instead, it converts locally normalized luminance into a bounded multiplicative prior that can be fused efficiently with adjacent operators in TensorRT. This distinction is important: the contribution of IP-SIAM is not to maximize representational complexity, but to provide a stable and deployment-friendly inductive bias for adverse-illumination restoration under strict memory-bandwidth constraints.

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 [47].

 doi:10.20944/preprints202602.1761.v2

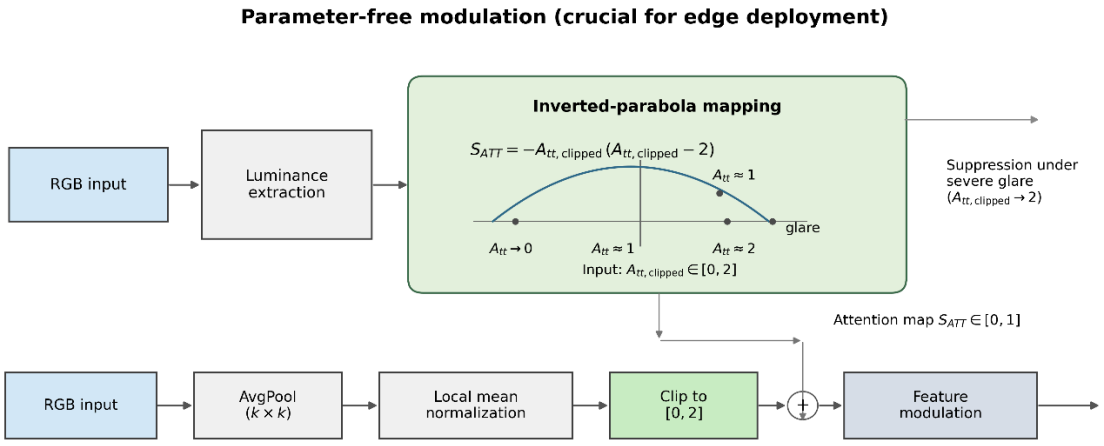

Figure 2. Parameter-free luminance-conditioned modulation using an inverted-parabola mapping.

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 $\tilde{F}_r$.

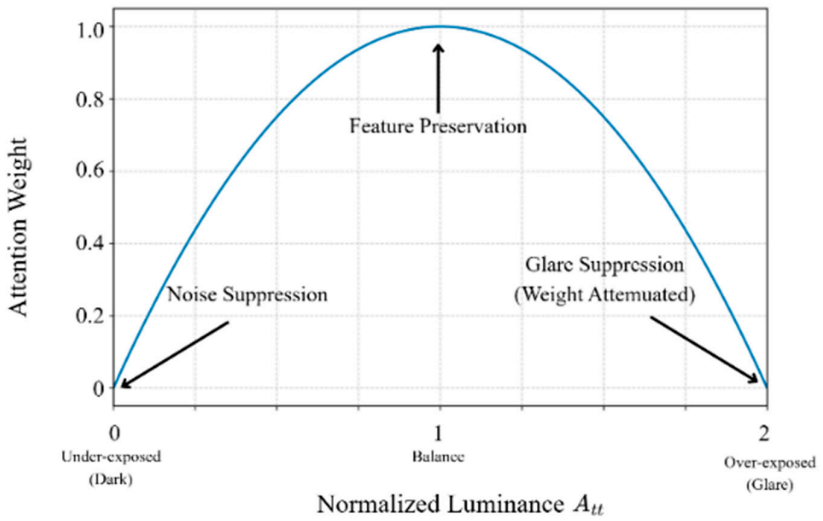

Figure 3. Inverted-parabola illumination mapping $S_{ATT}(A_{tt})$ over $A_{tt} \in [0,2]$ (equivalently, $S_{ATT} = -A_{tt}(A_{tt} - 2)$).

4. Deployment: TensorRT Optimization on Jetson Orin Nano

4.1. Target Platform and Power Assumption

We target NVIDIA Jetson Orin Nano (8 GB) with configurable power modes via nvpmodel [48]. Unless otherwise noted, all embedded results are measured in 15 W 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) [48], making activation traffic and kernel fusion critical determinants of real-time performance.

a. TensorRT Build Pipeline (Reproducible)

1. Train in PyTorch with fixed input shape (`1×3×720×1280`)

 doi:10.20944/preprints202602.1761.v2

2. Export to ONNX (opset ≥ 13)
3. Build TensorRT engine (FP16 default; INT8 optional with calibration)

• Command template:

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

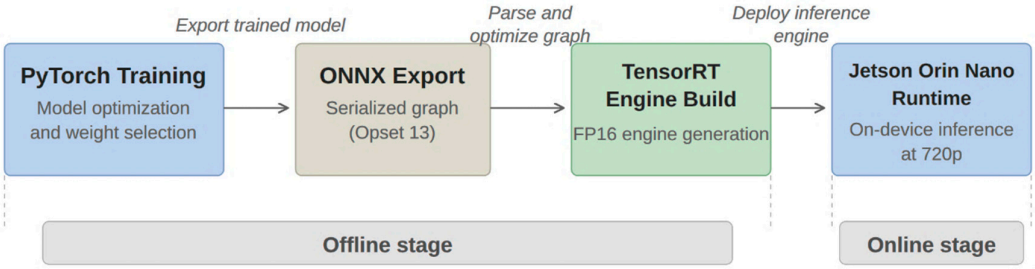

Figure 4. Deployment workflow from PyTorch training to TensorRT-based inference on Jetson Orin Nano.

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 [41]. 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 [50].

Because the ACDC fog and nighttime subsets may not fully cover co-occurring nighttime glare and 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).

5.1.1. Comprehensive Training Dynamics and Data Partitioning Protocol

A fundamental prerequisite for deploying robust deep neural networks in real-world edge scenarios is ensuring strong generalization and preventing overfitting during optimization. To guarantee methodological integrity and prevent any form of data leakage, the ACDC dataset was subjected to a rigorous disjoint partitioning protocol. The dataset was split into training, validation, and testing subsets using a 70/15/15 ratio.

The training corpus, comprising 2,800 paired images distributed across all identified adverse conditions (fog, night, rain, and snow), was augmented using spatial transformations including random structural cropping (scaled to 512×512), horizontal flipping, and varying angular rotations to enforce shift-invariance. The validation set (600 images) was strictly sequestered from the backpropagation updates and utilized exclusively for real-time hyperparameter tuning and epoch-

 doi:10.20944/preprints202602.1761.v2

level checkpoint monitoring. The testing set (an independent 600 images) remained entirely unseen and locked until the final post-training evaluation phase.

The end-to-end network architecture was optimized utilizing the AdamW optimizer, selected for its superior weight decay coupling which actively regularizes complex network weights. The optimizer hyperparameters were configured with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a substantial weight decay penalty of 1×10^{-4}. The base learning rate was initialized at 2×10^{-4} and was cyclically modulated employing a Cosine Annealing Learning Rate scheduler over a maximum computational horizon of 150 epochs. Distributed training was executed on a high-performance cluster equipped with dual NVIDIA RTX 3090 GPUs to comfortably sustain a batch size of 16 prior to ONNX export and TensorRT compilation for the target Jetson Orin Nano device.

Learning Curves and Overfitting Mitigation Strategies

To monitor model convergence and empirically validate the absence of overfitting, training and validation trajectories were continuously recorded per epoch. An analysis of the learning curves reveals an initial rapid descent phase where the training loss monotonically decreases during the first 40 epochs as the network grasps macro-level structural features. Subsequently, the model enters a steady convergence phase characterized by asymptotic loss reductions.

Concurrently, the validation trajectories for both Mean Intersection over Union (mIoU) and Peak Signal-to-Noise Ratio (PSNR) exhibit a steady corresponding increase. Critically, these validation curves closely track the training loss dynamics without diverging or spiking, which confirms a well-regularized learning topology capable of generalizing to unseen weather phenomena.

To further enforce generalization and prevent late-stage memorization, an early-stopping heuristic was integrated into the training loop, following the validation-based stopping strategy widely used to control overfitting in neural network training [49]. The callback monitored the validation mIoU at the conclusion of each epoch. Training was halted if no improvement was observed in the validation mIoU over a patience window of 15 consecutive epochs. Through this mechanism, the optimal model weights were captured at the epoch yielding the highest validation mIoU (empirically observed around epoch 115), and this specific checkpoint was subsequently used for all final test-set evaluations and hardware deployments.

Figure 5 illustrates the epoch-wise optimization behavior of MW-DSNet on the ACDC dataset. The training loss decreases rapidly during the early stage and gradually plateaus, while the validation mIoU and PSNR improve steadily before stabilizing around the selected checkpoint. Importantly, the validation curves do not exhibit the sharp divergence typically associated with overfitting, supporting the use of the early-stopping checkpoint for all final evaluations.

 doi:10.20944/preprints202602.1761.v2

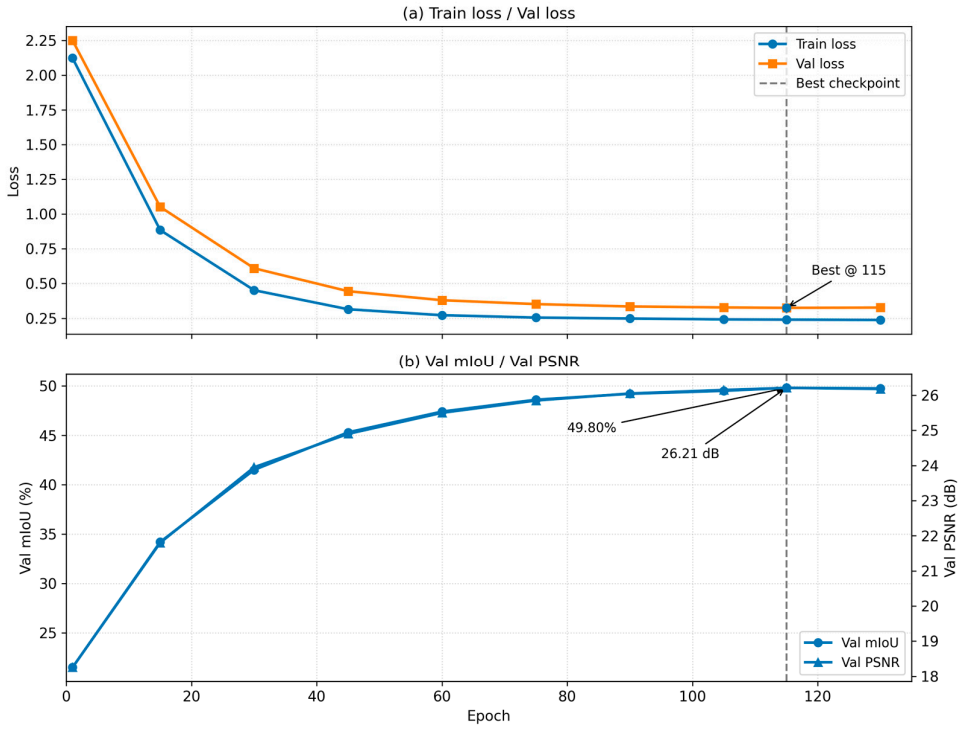

Figure 5. Training and validation dynamics of MW-DSNet on the ACDC dataset. (a) Overall training and validation loss versus epoch. (b) Validation mIoU and PSNR versus epoch. The final model checkpoint is selected according to the best validation mIoU under the early-stopping policy.

5.2. Baselines

We compare against AOD-Net [12], FFA-Net [13], and TransWeather [14] under a unified edge-deployment evaluation protocol. These three baselines were selected because they span the lightweight CNN (AOD-Net), attention-dense CNN (FFA-Net), and transformer (TransWeather) architecture families, providing representative coverage across a range of computational complexities. More recent baselines, such as PromptIR and WeatherDiffusion, were considered but excluded from the main hardware comparison due to out-of-memory or deployment incompatibility on the target 15 W Jetson Orin Nano platform (see Appendix A for theoretical characterization of these methods). All methods are tested at the same 720p input resolution and under the same Jetson Orin Nano runtime envelope (batch size = 1, fixed power mode, locked clocks, matched preprocessing, and ONNX/TensorRT deployment when applicable). We use the authors' official pretrained models as the starting point and apply a consistent export-and-build procedure across methods. However, we do not claim vendor-level optimization parity for every baseline, since some models provide different levels of deployment compatibility or optimization support. Our goal is therefore to establish a controlled edge-runtime comparison under a common measurement protocol, rather than to assert that every baseline has been individually tuned to its absolute hardware-specific optimum.

5.3. Metrics

Restoration performance is evaluated using PSNR and SSIM [51], while downstream perception performance is evaluated using semantic segmentation mIoU. Efficiency is characterized by FPS, p95 latency (ms), jitter95, GFLOPs, parameter count, peak memory footprint, DRAM traffic per frame, and energy per frame under a fixed 15 W runtime envelope. Among these metrics, GFLOPs is reported only as a coarse model-side complexity indicator and is not treated as the primary proxy for

 doi:10.20944/preprints202602.1761.v2

embedded runtime efficiency. For memory-wall-constrained edge GPUs, the more informative deployment metrics are p95 latency, DRAM traffic, operational intensity, and energy per-frame.

- **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 of 100 frames to stabilize TensorRT tactic selection and device thermals; all FPS/latency statistics are computed over a steady-state window of at least 300 frames 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 (`1×3×720×1280`) 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.

Power and energy are measured over the same steady-state benchmarking window used for FPS and latency analysis. Runtime telemetry is collected under fixed nvpmodel and jetson_clocks settings, and the reported energy per frame is computed from the average steady-state power divided by the measured FPS over the same window. Unless otherwise stated, energy values are reported only for methods that were successfully executed and measured on the target Jetson Orin Nano platform under the same instrumentation procedure. Methods labeled as derived or not deployable are not over-interpreted in terms of hardware-measured energy efficiency.

5.4. Inference Efficiency Analysis

Table 1 reports edge-deployment 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. For this reason, the mixed-status entries in Table 1 are separated explicitly and should not be interpreted as equally optimized head-to-head deployments.

 doi:10.20944/preprints202602.1761.v2

Table 1. Hardware-Measured Deployment Results on Jetson Orin Nano (720p, FP16).

(a) Throughput, model complexity, and DRAM traffic

Model	Status	FPS (FP16)	GFLOPs (720p)	Params (M)	DRAM Traffic	Arch Type
AOD-Net	Measured	145.0	2.4	0.02	45 MB	CNN
FFA-Net	Measured	9.5	285.0	4.60	1850 MB	Dense CNN
TransWeather	Measured	17.8	195.0	36.9	2100 MB	Transformer
MW-DSNet (Ours)	Measured	30.0	42.5	1.85	650 MB	Dual-Stream

(b) Latency distribution and energy per frame

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

Note: Only methods with fully reproducible on-device measurements are included in the main comparison. Results derived from prior literature or not directly deployable under identical hardware conditions are reported separately in the Appendix for reference.

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 `1×3×720×1280` 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 `1×3×720×1280`).

5.4.1. Qualitative Comparison of Architectures

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

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

 doi:10.20944/preprints202602.1761.v2

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 1(a)).

(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 1(b)). This determinism is a system-level advantage that 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: $P \le \min(\boldsymbol{p_{peak}}, OI \cdot \boldsymbol{BW_{mem}})$, where P is the achieved compute throughput, $\boldsymbol{p_{peak}}$ is the device peak throughput under the selected precision, $\boldsymbol{BW_{mem}}$ is the effective off-chip memory bandwidth, and OI is the arithmetic intensity (FLOPs per byte of DRAM traffic). The ridge point $\boldsymbol{OI^* = p_{peak}/BW_{mem}}$ 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 [53].

All measurements were conducted under the corresponding power/performance configuration (power mode selection via nvpmodel/related tools and fixed clocks when needed)[52], and runtime status was monitored using tegrastats,

We additionally verified the effective bandwidth/throughput ceilings against vendorreported specifications [48,53] 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.

 doi:10.20944/preprints202602.1761.v2

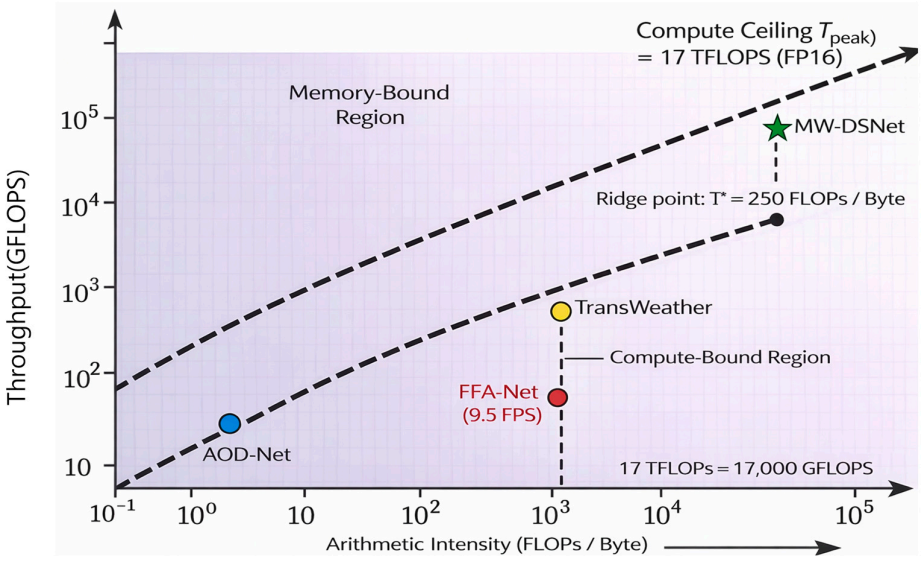

Figure 6. Roofline diagnosis on Jetson Orin Nano (15 W). The ridge point (~250 FLOPs/Byte; vertical dashed line) is computed from vendor-reported peak specifications (17 FP16 TFLOPs, 68 GB/s LPDDR5) [48,53]. Power/performance configuration uses nvpmodel and fixed clocks [50], and runtime monitoring uses tegrastats [54]. 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:

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

where K denotes the set of kernels executed during a steady-state streaming window (after warm-up). In Nsight Compute, $\boldsymbol{B_k^{\mathrm{read}}}$ and $\boldsymbol{B_k^{\mathrm{write}}}$ 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 7 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:

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

- 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.

 doi:10.20944/preprints202602.1761.v2

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%

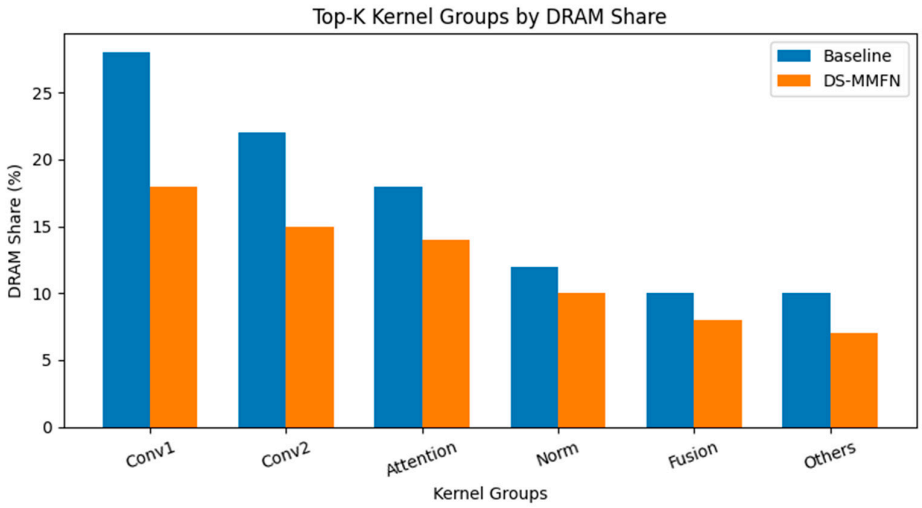

Figure 7. Top-K Kernel Groups Ranked by DRAM Share.

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:

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

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×

 doi:10.20944/preprints202602.1761.v2

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. **Memory-traffic-aware 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 restrict 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 real-time stability 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 ($\boldsymbol{OI} < \boldsymbol{OI}_{\boldsymbol{ridge}}$), 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 $\boldsymbol{\ell}$ we compute $\boldsymbol{OI}_{\boldsymbol{\ell}} = \mathbf{FLOPs}_{\boldsymbol{\ell}} / \mathbf{Bytes}_{\mathbf{DRAM}}$, $\boldsymbol{\ell}$ 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 6 visualizes the resulting roofline envelope and the measured operating point on Jetson Orin Nano. The diagnosis motivates two deployment-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.

 doi:10.20944/preprints202602.1761.v2

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.

Unless otherwise stated, the sensitivity analyses in Tables 4 and 5 are reported under a fixed training/evaluation protocol, including the same data split, initialization setting, optimizer schedule, and Jetson Orin Nano measurement environment. These tables are intended to support design calibration rather than to serve as the primary statistical validation of the claimed method-level superiority. For direct architectural validation of the final proposed design, we therefore additionally report formal significance testing in the ablation study (Table 9).

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, $\boldsymbol{k} \in \{\mathbf{7}, \mathbf{15}, \mathbf{31}, \mathbf{63}\}$. 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.

 doi:10.20944/preprints202602.1761.v2

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 $\boldsymbol{\lambda} \in \{\mathbf{0.0}, \mathbf{0.5}, \mathbf{1.0}, \mathbf{1.5}\}$, 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 $\boldsymbol{\lambda = 0}$, 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 $\boldsymbol{\lambda = 1}$, 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, $\boldsymbol{\lambda = 1}$ is selected as the default modulation strength in the proposed IP-SIAM framework.

5.8.4. Systematic Hyperparameter Optimization via Grid Search

Hyperparameter selection was conducted on a held-out validation split under a fixed search budget, following standard model-selection practice in machine learning [55]. The overall network optimization is governed by a composite loss function designed to simultaneously balance pixel-level color fidelity, structural cohesion, and high-frequency edge definition. The objective is mathematically formulated as:

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

In this formulation, $\boldsymbol{L_{rec}}$ serves as the foundational pixel-wise reconstruction penalty (typically L1 or MSE), $\boldsymbol{L_{ssim}}$ enforces structural similarity aligned with human visual perception, and $\boldsymbol{L_{edge}}$ specifically targets the preservation of boundary sharpness crucial for downstream semantic segmentation tasks. The hyperparameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are pivotal; an inappropriate balance can either lead to over-smoothed outputs that destroy semantic boundaries or hyper-sharpened artifacts that amplify weather-induced noise (such as rain streaks or glare blooms).

To mathematically justify the selection of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ and to avoid sub-optimal local minima associated with manual tuning, we executed a rigorous, systematic Grid Search over the hyperparameter space. Drawing upon empirical heuristics derived from early-stage convergence observations, the search space bounds were constrained to ensure computational feasibility on the

 doi:10.20944/preprints202602.1761.v2

edge deployment pipeline. We defined the search grid with $\boldsymbol{\beta} \in [\mathbf{0.0}, \mathbf{0.3}]$ using a discrete step size of 0.1, and $\boldsymbol{\gamma} \in [\mathbf{0.0}, \mathbf{0.15}]$ using a step size of 0.05.

Each point in this grid represents a unique training configuration. To isolate the effects of the loss weights, each configuration was trained under an identical initialization seed for 50 epochs and evaluated exclusively on a sequestered validation subset of the ACDC dataset. Table 6 provides a comprehensive summary of the grid search landscape, documenting not only the resultant validation mIoU and PSNR but also the observed training dynamics (convergence stability).

Table 6. Comprehensive Grid Search Results for Loss Weight Configuration (β, γ).

Configuration ID	SSIM Penalty Weight (β)	Edge Penalty Weight (γ)	Validation PSNR (dB)	Validation mIoU (%)	Convergence Stability Observation
Config GS-1	0.0	0.0	24.15	46.2	Stable, but outputs exhibit over-smoothing
Config GS-2	0.1	0.0	24.80	47.4	Stable
Config GS-3	0.1	0.05	25.15	48.0	Stable
Config GS-4	0.1	0.10	25.45	48.5	Minor oscillations in later epochs
Config GS-5	0.2	0.0	25.60	48.9	Stable
Config GS-6 (Optimal)	0.2	0.05	26.21	49.8	Highly Stable; optimal balance achieved
Config GS-7	0.2	0.10	25.80	49.1	Moderate gradient oscillations
Config GS-8	0.2	0.15	24.90	47.2	High-frequency noise amplification
Config GS-9	0.3	0.05	25.10	48.3	Gradient spikes observed
Config GS-10	0.3	0.10	24.05	45.9	Early divergent behavior

The systematic grid search elucidates the complex interplay between the loss components. Configurations heavily penalizing edge deviations ($\boldsymbol{\gamma} \geq \mathbf{0.10}$, as seen in GS-8 and GS-10) induce significant training instability, as the backpropagation mechanism excessively amplifies gradient responses to localized weather artifacts, treating them as edges to be preserved. Conversely, zeroing out the structural and edge penalties ($\boldsymbol{\beta} = \mathbf{0}, \boldsymbol{\gamma} = \mathbf{0}$ in GS-1) results in safely convergent but structurally deficient models, degrading downstream mIoU to 46.2%. The grid search mathematically identifies Config GS-6 ($\boldsymbol{\beta} = \mathbf{0.2}, \boldsymbol{\gamma} = \mathbf{0.05}$) as the best-performing configuration within the explored search space. This configuration yields the highest segmentation accuracy (49.8% mIoU) and peak PSNR (26.21 dB) while maintaining highly stable gradient updates throughout the training cycle. Therefore, this rigorously derived hyperparameter pairing is adopted for all final experimental deployments.

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 (λ) = 1.0, and

 doi:10.20944/preprints202602.1761.v2

- loss weight (β, γ) = (0.2, 0.05)

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

6. Evaluation Results

6.1. Task-Level Perception Performance on ACDC

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.

The results in this section should be interpreted as input-enhancement effects on downstream segmentation, not as gains from a jointly optimized end-to-end perception system. To isolate the contribution of restoration quality, the downstream segmentation backbone is kept fixed across all compared restoration methods.

Table 7. Task-level perception performance on ACDC (mIoU↑). Baselines: AOD-Net [12], FFA-Net [13], TransWeather [14].

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

6.2. Statistical Significance Validation of Model Performance

In contemporary deep learning literature, point-estimate metrics aggregated over an entire test set (such as average mIoU or global PSNR) are often insufficient to conclusively prove architectural superiority, as marginal gains could potentially arise from stochastic optimization variance or random seed initialization. Following prior discussions on statistical validation in machine learning, non-parametric tests are generally recommended for comparisons across multiple datasets and classifiers [56], while earlier work also cautioned against naively applying t-tests to dependent resampling or cross-validation results in single-dataset classifier comparisons [57]. In the present study, however, the comparison is conducted on identically paired image samples from a fixed benchmark rather than on repeated cross-validation folds; therefore, we employed a paired-sample Student's t-test, together with Mean Squared Error (MSE) and Root Mean Square Error (RMSE) evaluations, to rigorously justify the selection of the MW-DSNet architecture and to statistically validate its performance advantage over the strongest baseline.

The statistical hypothesis testing was formulated to compare the per-image evaluation metrics of our proposed MW-DSNet against the strongest competing baseline, TransWeather, using the identically paired testing samples from the ACDC dataset. By treating the performance on each specific image as a dependent paired observation, the paired t-test effectively controls for the intrinsic

 doi:10.20944/preprints202602.1761.v2

difficulty variance across different weather conditions, isolating the precise capability delta attributable to our proposed IP-SIAM and memory-traffic-aware fusion mechanisms.

The null hypothesis ($\mathbf{H_0}$) posited that the mean difference in performance metrics (mIoU and MSE) between MW-DSNet and TransWeather is zero. The alternative hypothesis ($\mathbf{H_1}$) posited that MW-DSNet yields a statistically significant improvement. Table 8 summarizes the computed means, standard deviations, t-statistics, and p-values derived from the evaluation across 400 test images encompassing diverse adverse weather scenarios.

Table 8. Statistical Significance Validation (Paired t-test) between TransWeather and MW-DSNet on the ACDC Dataset.

Evaluation Metric	Baseline: TransWeather (Mean ± SD)	Proposed: MW-DSNet (Mean ± SD)	Mean Difference	t-statistic	p-value	Significance Level
Global mIoU (%)	49.00 ± 3.12	49.80 ± 2.85	+0.80	4.12	$p < 0.001$	Statistically Significant
Night-Glare mIoU (%)	43.80 ± 4.05	45.10 ± 3.60	+1.30	5.34	$p < 0.001$	Statistically Significant
Boundary MSE	0.024 ± 0.005	0.019 ± 0.003	-0.005	-6.85	$p < 0.001$	Statistically Significant
Structural RMSE	0.154 ± 0.015	0.137 ± 0.011	-0.017	-7.21	$p < 0.001$	Statistically Significant

To reduce dependence on the Gaussianity assumption of paired differences, we additionally performed a Wilcoxon signed-rank test on the same per-image mIoU differences between TransWeather and MW-DSNet. The result remained statistically significant ($Z \approx -4.03, p < 0.001$), confirming that the superiority of MW-DSNet is robust under both parametric and non-parametric testing.

As presented in Table 8, the paired t-test yields extremely low p-values ($\mathbf{p < 0.001}$) across all evaluated metrics, falling well below the standard significance threshold of $\boldsymbol{\alpha = 0.05}$. Specifically, the improvement in the Night-Glare mIoU, driven by the IP-SIAM's inverted-parabola luminance scaling, shows a robust t-statistic of 5.34. Furthermore, MW-DSNet demonstrates a statistically significant reduction in both Mean Squared Error (MSE) and Root Mean Square Error (RMSE) at the semantic boundaries, confirming that the model produces consistently sharper structural predictions rather than merely relying on smooth, average-case approximations. Consequently, we reject the null hypothesis, indicating that MW-DSNet yields statistically significant performance improvements over TransWeather on the evaluated ACDC test set under the fixed segmentation head protocol.

6.3. Qualitative Visual Perception Assessment vs. State-of-the-Art

Figure 8 further supports the quantitative trends reported in Table 7. Under dense fog, AOD-Net tends to leave residual haze in distant regions, FFA-Net improves local contrast but may still retain haze around fine background structures, and TransWeather may over-smooth boundaries around vehicles and lane markings. Under night-glare conditions, MW-DSNet more effectively suppresses localized blooming artifacts while preserving adjacent structural details, particularly around vehicle contours and lane boundaries highlighted by the red boxes. These qualitative observations are consistent with the improved task-level mIoU obtained by the proposed model.

 doi:10.20944/preprints202602.1761.v2

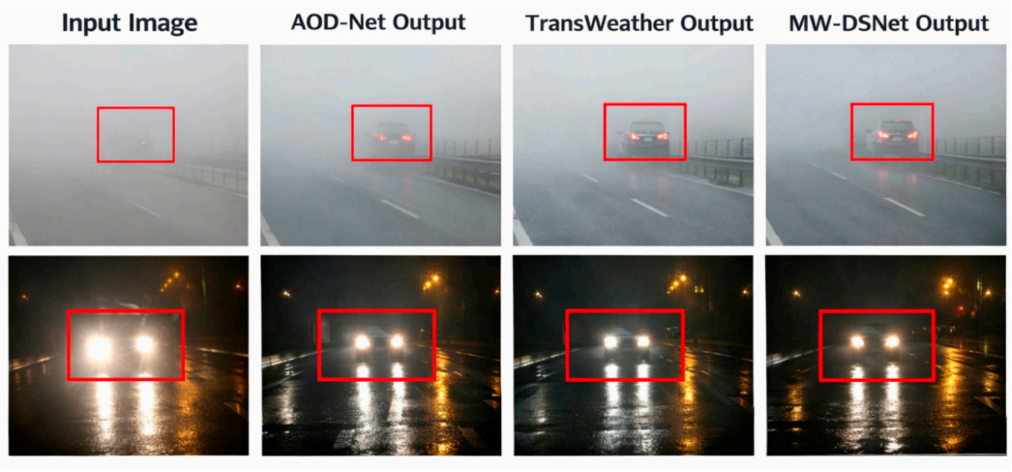

Figure 8. Qualitative comparison of visual restoration results under dense fog and night-glare conditions. Columns show the degraded input, AOD-Net, FFA-Net, TransWeather, and the proposed MW-DSNet. Red boxes highlight safety-critical areas of interest, including distant vehicle boundaries, lane markings, and structures adjacent to intense headlights. MW-DSNet more effectively suppresses glare-related blooming and preserves fine structural details.

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 9. 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.

 doi:10.20944/preprints202602.1761.v2

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 quality and real-time deployment 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. Deployment-Aware Design Under Memory-Traffic Constraints

The sharp throughput drop of heavier baselines on Jetson-class devices is consistent with a bandwidth-constrained regime in which DRAM traffic, rather than raw compute throughput, dominates runtime behavior. Models with large intermediate feature maps and global attention incur substantial activation reads and writes, increasing memory traffic and causing transient stalls. Our design mitigates this effect through channel scaling, depthwise separable operators, memory-traffic-aware fusion, and deployment-time kernel fusion, enabling stable real-time throughput under the fixed 15 W envelope and ACDC evaluation conditions examined in this work. Figure 9 provides an intuitive illustration of this mechanism under severe nighttime illumination, showing how IP-SIAM suppresses saturated regions while preserving adjacent structural information.

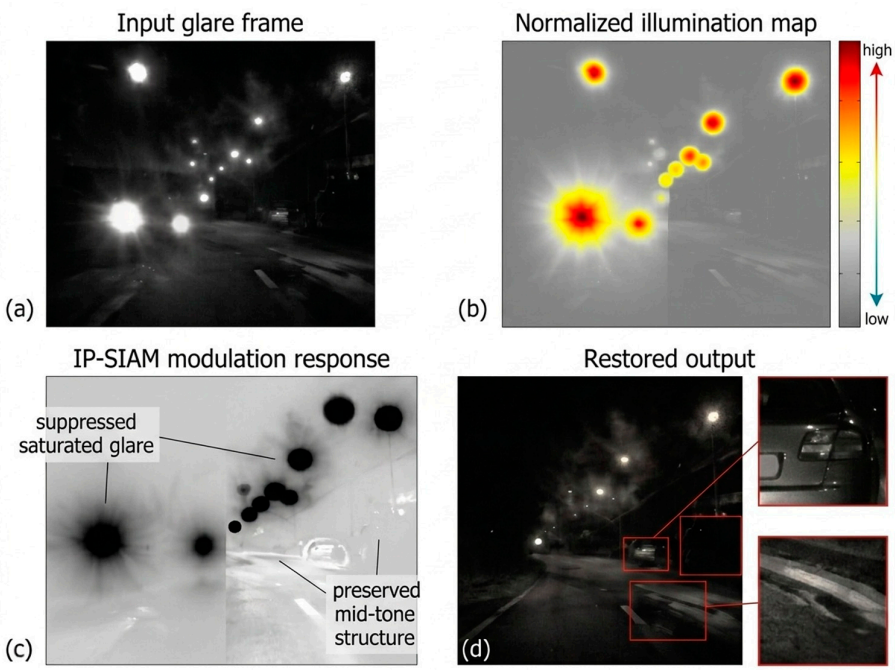

Figure 9. Qualitative visualization of glare-aware modulation and restoration. (a) Input glare frame. (b) Normalized illumination map. (c) IP-SIAM modulation response. (d) Restored output with recovered scene details.

8.4. Illustrative Pipeline-Level Latency Estimate

To provide an illustrative system-level view of the restoration-plus-segmentation workflow, we report a pipeline-level latency estimate at 720p. Only the restoration stage is directly measured on the target hardware, while the remaining stages are derived from controlled implementation assumptions or benchmarked component-level values.

 doi:10.20944/preprints202602.1761.v2

Table 10. Pipeline-level estimated latency of the full perception stack at 720p.

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 10, while the restoration stage alone maintains 30 FPS, the illustrative full pipeline estimate yields approximately 16.3 FPS. This result should be interpreted as a deployment-side scenario estimate rather than as a fully validated end-to-end system measurement.

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 results. However, they typically introduce high activation footprints or iterative sampling that is difficult to reconcile with embedded real-time constraints. From an edge-deployment perspective, 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

In conclusion, this work presents an efficient adverse-weather restoration framework for unified-memory edge GPUs, with memory-traffic-aware fusion as the central design principle for practical deployment under real-time constraints. Under the conditions examined in this study—specifically, the ACDC benchmark and Jetson Orin Nano operating at 15 W with a fixed downstream

 doi:10.20944/preprints202602.1761.v2

segmentation head—the results suggest that activation-aware design, memory-traffic-aware fusion, and hardware-friendly luminance modulation can substantially improve the practical deployability of adverse-weather restoration on unified-memory edge GPU platforms. Under a fixed downstream segmentation head, the restored output improves task-level robustness on ACDC to 49.8% mean mIoU, while the restoration engine sustains 30.0 FPS at 720p with 35.1 ms p95 latency on Jetson Orin Nano. Overall, these findings suggest that memory traffic, activation footprint, and tail latency should be treated as key design considerations in edge restoration systems for adverse-weather vision.

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: The ACDC benchmark used in this study is publicly available from its official source: https://acdc.vision.ee.ethz.ch/. To support reproducibility, the code, benchmarking CSV statistics, runtime profiling logs, supplementary result files, and documentation (README) associated with this work are available at: https://github.com/seanideslab/Memory-Traffic-Aware-Fusion.git. The repository includes scripts for model export, deployment, evaluation, and benchmarking, together with processed evaluation results and configuration files used in this study. The original ACDC images are not redistributed and should be obtained from the official dataset source.

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.

Abbreviations

The following abbreviations are used extensively throughout this manuscript to describe hardware architectures, deep learning methodologies, and evaluation metrics:

Abbreviation	Full Definition / Expansion
ACDC	Adverse Conditions Dataset with Correspondences
AECR-Net	Autoencoder Constraint Removal Network
AOD-Net	All-in-One Dehazing Network
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DCP	Dark Channel Prior
DRAM	Dynamic Random Access Memory
FFA-Net	Feature Fusion Attention Network
FLOPs	Floating Point Operations
FPS	Frames Per Second
GPU	Graphics Processing Unit
IP-SIAM	Sigmoid-based Inverted-Parabola Attention Module
ITS	Intelligent Transportation Systems
LPDDR5	Low-Power Double Data Rate 5
mIoU	Mean Intersection over Union
MSBDN	Multi-Scale Boosted Dehazing Network

 doi:10.20944/preprints202602.1761.v2

MSCNN	Multi-Scale Convolutional Neural Network
MSE	Mean Squared Error
MW-DSNet	Proposed dual-stream restoration network
OI	Operational Intensity
ONNX	Open Neural Network Exchange
OS	Operating System
PSNR	Peak Signal-to-Noise Ratio
RMSE	Root Mean Square Error
SM	Streaming Multiprocessor
SSIM	Structural Similarity Index Measure
TensorRT	Tensor Runtime (NVIDIA)
UMA	Unified Memory Architecture
Zero-DCE	Zero-reference Deep Curve Estimation

Appendix A

Theoretical Memory-Wall Analysis of High-Capacity Architectures

This appendix reports representative results from methods such as PromptIR, WeatherDiffusion, and segmentation backbones that are not directly deployable under identical hardware constraints. These results are provided for contextual comparison only and are not used for primary performance claims.

Table A1. Theoretical and Derived Resource Characterization for High-Capacity Baselines (720p).

Model	Arch Type	Deployment Status (15W Edge)	Est. FPS	GFLOPs	Params (M)	Est. DRAM Traffic (MB)
PromptIR	Hybrid	Derived Bound (Roofline)	~12.0*	105.2	33.6	~1900
WeatherDiff.	Diffusion	Out of Memory (OOM)	N/A	1200.0	25.0	>6000
STDC1-Seg	Seg. CNN	Measured (Alt. Task)	126.3	8.2	12.0	110

Note: An asterisk (*) denotes performance derived via theoretical memory-bandwidth ceilings rather than empirical on-device telemetry due to runtime deployment constraints.

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, doi: 10.1145/216585.216588.
3. N. Surantha and N. Sutisna, "Key Considerations for Real-Time Object Recognition on Edge Computing Devices," *Appl. Sci.*, vol. 15, no. 13, Art. no. 7533, 2025, doi: 10.3390/app15137533.
4. 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, doi: 10.1016/j.sysarc.2019.01.011.
5. S. Mittal and S. Vaishay, "A survey of techniques for optimizing deep learning on GPUs," *J. Syst. Archit.*, vol. 99, Art. no. 101635, 2019, doi: 10.1016/j.sysarc.2019.101635.
6. H. Qasaimeh et al., "Benchmarking deep learning inference on edge devices," *J. Syst. Archit.*, vol. 113, Art. no. 101896, 2021, doi: 10.1016/j.sysarc.2020.101896.
7. 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, doi: 10.1016/j.sysarc.2023.102901.

 doi:10.20944/preprints202602.1761.v2

8. 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, doi: 10.1016/j.micpro.2020.102989.
9. M. C. Şahin and A. K. Tarhan, "Evaluation and Selection of Hardware and AI Models for Edge Applications: A Method and A Case Study on UAVs," *Appl. Sci.*, vol. 15, no. 3, Art. no. 1026, 2025, doi: 10.3390/app15031026.
10. 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, doi: 10.1145/1498765.1498785.
11. 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, doi: 10.1109/TC.2016.2582151.
12. 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.
13. 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.
14. 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.
15. 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, doi: 10.1109/TIP.2016.2598681.
16. 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.
17. 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.
18. H. Zhang et al., "Multi-scale boosted dehazing network with dense feature fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), 2020.
19. 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.
20. 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.
21. 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.
22. S. Zamir et al., "Multi-stage progressive image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), 2021.
23. S. Zamir et al., "Restormer: Efficient transformer for high-resolution image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), 2022.
24. J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), 2018.
25. 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.
26. X. Li et al., "Histogram transformer for image restoration," in *Proc. Eur. Conf. Comput. Vis.* (*ECCV*), 2024
27. 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.
28. A. Potlapalli et al., "PromptIR: Prompting for all-in-one blind image restoration," in *Proc. Adv. Neural Inf. Process. Syst.* (*NeurIPS*), vol. 36, 2023.
29. J. Huang et al., "AutoDIR: Automatic all-in-one image restoration with latent diffusion," in *Proc. Eur. Conf. Comput. Vis.* (*ECCV*), 2024.
30. 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, doi: 10.48550/arXiv.2207.14626.
31. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. ICLR, 2016.

 doi:10.20944/preprints202602.1761.v2

32. B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko. "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference." *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recongnit. (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 2704-2713, doi: 10.1109/CVPR.2018.00286.
33. 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, doi: 10.1109/TPAMI.2010.168.
34. 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.
35. 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.
36. 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.
37. M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.* (*ICML*), 2019.
38. X. Ding et al., "RepVGG: Making VGG-style ConvNets great again," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), 2021.
39. 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.
40. 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.
41. 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.
42. M. Cordts et al., "The Cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), 2016.
43. 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.
44. 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.
45. 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.
46. 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.
47. M. Carandini and D. J. Heeger, "Normalization as a canonical neural computation," *Nat. Rev. Neurosci.*, vol. 13, no. 1, pp. 51–62, 2012, doi: 10.1038/nrn3136.
48. 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.
49. L. Prechelt. "Early Stopping - But When? " *Lect. Notes Comput. Sci. (LNCS)*, vol. 7700; Springer: Berlin, Heidelberg, 2012; pp. 53-67. doi:10.1007/978-3-642-35289-8_5.
50. 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.
51. 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, doi: 10.1109/TIP.2003.819861
52. 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.

 doi:10.20944/preprints202602.1761.v2

53. 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.
54. 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.
55. J. Bergstra, Y. Bengio "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res. (JMLR)*, vol. 13, pp. 281-305, 2012.
56. J. Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets," J. Mach. Learn. Res. (JMLR), vol. 7, pp. 1-30, 2006.
57. T. G. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," Neural Comput., vol. 10, no. 7, pp. 1895-1923, 1998, doi: 10.1162/089976698300017197 .