Beyond Sequential Scanning: Rethinking Visual State Propagation with Neighborhood State Interaction

Anonymous authors
Paper under double-blind review

Abstract

Visual state space models (VSSMs) provide an efficient mechanism for modeling long-range dependencies in vision tasks. However, many existing visual state space models mainly rely on sequential scanning to organize state propagation, which introduces indirect interactions between spatially adjacent locations and makes local information exchange dependent on intermediate states. In this work, we rethink visual state propagation beyond sequential scanning and introduce Neighborhood State Interaction (NSI), a novel state interaction mechanism that enables direct interactions among neighboring states during each update. Instead of propagating states along predefined scan paths, NSI performs synchronous neighborhood interactions on two-dimensional feature maps and progressively expands contextual information through multiple propagation steps. To make multi-step state interaction efficient and stable, we generate input-conditioned interaction kernels and share them across propagation steps, while using state-dependent retention to adaptively control accumulated information and incremental readout to combine representations from different propagation stages. Extensive experiments on image restoration tasks demonstrate that our approach consistently improves over SS2D, achieving PSNR gains of 0.50 dB on the GoPro dataset for deblurring, 1.15 dB on the O-HAZE dataset for dehazing, and 1.37 dB on the Haze4K dataset for dehazing, while reducing MACs by 25.1% and 30.2% for deblurring and dehazing models, respectively.

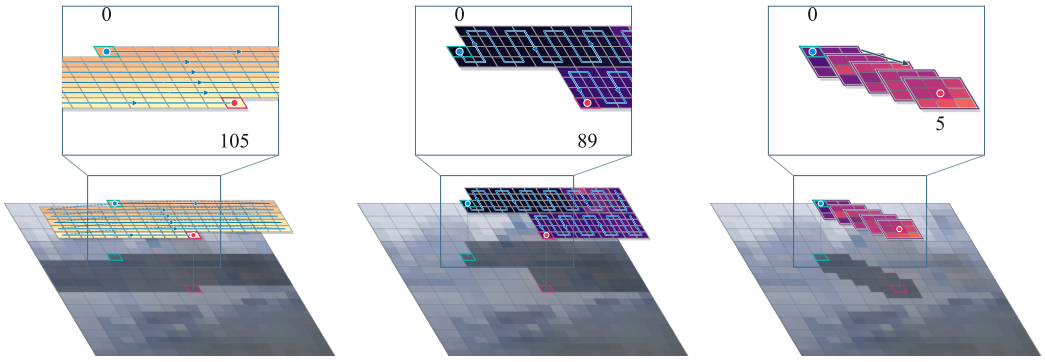
1 Introduction

Visual state space models (VSSMs) model long-range dependencies efficiently through selective state propagation, where input-dependent parameters decide how much information is retained and passed on (Gu & Dao, 2024; Liu et al., 2024). Because this propagation runs over one-dimensional sequences, many VSSMs organize visual state propagation by sequential scanning, flattening the two-dimensional grid along ordered paths. Which positions exchange states directly, and hence which transfers selection gates, depends on the scan order rather than spatial adjacency. This matters for image restoration, where local structures and broader context jointly guide reconstruction.

Several VSSMs improve the scans themselves, with VMamba's 2D Selective Scan (SS2D) combining four directional scans (Liu et al., 2024), and MaIR and EAMamba refining scan paths and directional coverage for restoration (Li et al., 2025; Lin et al., 2025). These designs strengthen spatial interactions and change which neighbors are consecutive, but within any one-dimensional scan branch, at most two of a position's eight spatial neighbors are consecutive with it, and the others exchange states only through intermediate positions (Figure 1(a,b)). In SS2D, diagonal neighbors, unlike axial ones, are never consecutive in any of the four scans. A depthwise convolution mixes their features before the scan, yet in the first token mixer of a trained SS2D model, state components keep a median of over 96% of their magnitude across one axial step but under 0.2% along diagonal paths (Appendix C.3).

This limitation stems from tying selective state propagation to one-dimensional sequences, so we rethink visual state propagation beyond sequential scanning: selection should act on whole neighborhoods at once, not along a propagation order. To implement this idea, we introduce Neighborhood State Interaction (NSI), a token mixer for image restoration that updates a two-dimensional state grid

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) 105 steps (b) 89 steps (c) 5 steps

Figure 1: **Ordered scan propagation and direct neighborhood state interaction** on a 24×20 grid: (a) a selected SS2D branch from VMamba, (b) a shifted NSS branch used in MaIR, and (c) NSI. Blue and red points mark the start and end positions. A step denotes one recurrence along the displayed scan branch in (a,b) and one synchronous neighborhood update in (c), in which every position exchanges states with its axial and diagonal neighbors. Panel (c) shows consecutive local state grids; NSI supports any chosen number K of propagation steps.

synchronously. At each update, every position exchanges states directly with its axial and diagonal neighbors, so local information exchange no longer depends on intermediate states, and repeated updates progressively expand the gathered context (Figure 1(c)).

Selection across these repeated updates must stay efficient and stable. A scan selects once per position from its token; a synchronous update revisits every position with the same input but an evolving state, so NSI selects from both. Input-conditioned interaction kernels, generated once and shared across steps much as CSPN reuses input-predicted affinities (Cheng et al., 2020), select spatial interactions from the input. State-dependent retention selects how much accumulated information each update carries forward, attenuating the previous state more strongly as its magnitude grows. Regenerating kernels atop retention costs 24.5–32.9% more MACs without consistently improving PSNR (Section 5.2). A repeated input term keeps input features available, and incremental readout combines the state increments of different stages, since early steps gather local context and later steps broader context. Propagation gain control keeps the summed kernel gain at each position below one.

State-dependent retention makes each update nonlinear, so this gain bound alone does not guarantee contraction. For a fixed module input, we show that the gain bound suffices once retention acts on groups of at most eight channels; with the prescribed readout weights, this contraction bounds incremental readout independently of the number of steps. In controlled deblurring and dehazing experiments that replace only the token mixer of VIVNet backbones (Cui et al., 2026), NSI achieves higher PSNR than SS2D with fewer complete-model MACs and outperforms a CSPN-style mixer while using 4.0–5.4% more MACs. Ablations show task-dependent effects of retention and readout.

We summarize our contributions as follows:

- **Efficient selective neighborhood propagation.** We propose NSI, a token mixer that replaces scan-ordered recurrence with synchronous neighborhood updates. Input-conditioned interaction kernels shared across steps, together with state-dependent retention, enable context-dependent selection without regenerating kernels at every step; incremental readout combines the state increments across propagation steps.

- **Stability guarantees for nonlinear propagation and readout.** For a fixed module input, we establish sufficient conditions on channel grouping and propagation gain for contraction of the complete nonlinear update. Given these conditions, zero initialization and the prescribed readout weights bound the internal state and incremental readout norms independently of the number of propagation steps. The readout bound holds before output scaling and projection.

- **Improved restoration quality and efficiency.** With the same backbone architecture and training budget within each task, NSI improves PSNR over SS2D by 0.50 dB on GoPro (Nah et al., 2017), 1.15 dB on O-HAZE (Ancuti et al., 2018), and 1.37 dB on Haze4K, while reducing complete-model MACs by 25.1% for deblurring and 30.2% for dehazing.

2 Related Work

2.1 Visual State Space Models

State space models (SSMs) summarize the history of a sequence in a recurrent hidden state. S4, a structured SSM, makes long-sequence modeling efficient while keeping its learned SSM parameters fixed across sequence positions (Gu et al., 2022). Mamba builds on this formulation by making selected SSM parameters depend on the current input, so that information retention and propagation can vary with token content (Gu & Dao, 2024). Adapting these sequence models to vision also requires organizing state interactions on a two-dimensional grid. VMamba's SS2D combines four scan sequences, and MaIR uses nested S-shaped paths within and between strips (Liu et al., 2024; Li et al., 2025). EAMamba scans channel groups horizontally, vertically, and diagonally (Lin et al., 2025). Mamba2D combines preceding states along two axes in a wavefront recurrence (Baty et al., 2024), whereas Spatial-Mamba fuses neighboring states after a unidirectional scan (Xiao et al., 2025). NSI instead updates the entire grid through repeated synchronous neighborhood interactions, without a scan path. Each updated state guides retention at the next step, which couples spatial information exchange with state-dependent selection.

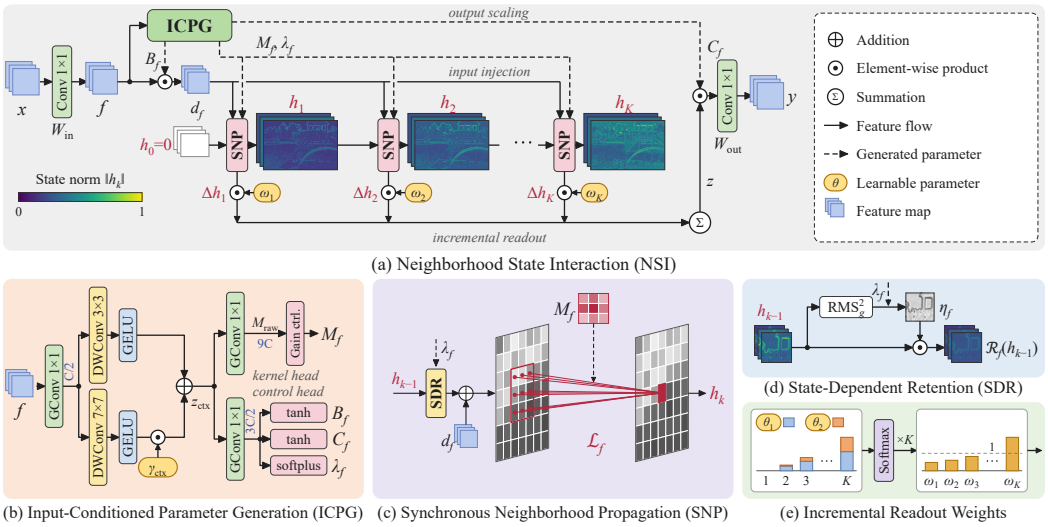
2.2 Spatial Propagation

Spatial propagation networks refine depth estimates with learned affinities. CSPN reuses input-predicted affinities and also supports stereo cost volumes and multiscale features, and NLSPN learns nonlocal neighborhoods (Cheng et al., 2020; Park et al., 2020). DySPN adapts affinities across iterations, whereas MSPN updates depth estimates and propagation masks through local attention (Lin et al., 2022; Jun et al., 2024). NSI instead addresses selective propagation of latent visual states. It keeps input-conditioned interaction kernels fixed across propagation steps, adapts retention to the evolving state, and combines state increments rather than depth predictions. NSI separates the spatial connections from the state-dependent decision of how much accumulated information to retain.

DRCN shares convolutional parameters across recursive steps and combines the super-resolution predictions of different recursion depths (Kim et al., 2016). Stability analysis also has precedents: CSPN constrains its propagation coefficients, and NLSPN relates affinity normalization to the Jacobian of the update (Cheng et al., 2020; Park et al., 2020). NSI contributes the joint design and analysis of state-dependent retention and incremental readout with shared interaction kernels. Our analysis includes the derivative of retention in the complete nonlinear update, derives sufficient conditions on channel grouping and propagation gain for contraction, and bounds both the state and readout norms independently of the number of propagation steps.

2.3 Image Restoration

Image restoration methods balance local detail recovery, contextual modeling, and computational efficiency. MPRNet refines degraded images progressively through multiple stages that exchange features (Zamir et al., 2021). NAFNet simplifies convolutional restoration blocks with multiplicative gating and channel reweighting (Chen et al., 2022). SwinIR uses shifted-window attention, and Restormer combines attention across channels with depthwise convolutions for high-resolution restoration (Liang et al., 2021; Zamir et al., 2022). VIVNet embeds a compact visual module in each block of a U-shaped architecture; this module combines depthwise convolutions at multiple receptive fields, similarity-aware weighting, and repeated elementwise multiplication to model higher-order feature interactions (Cui et al., 2026).



(a) Neighborhood State Interaction (NSI)

(b) Input-Conditioned Parameter Generation (ICPG) (c) Synchronous Neighborhood Propagation (SNP) (d) State-Dependent Retention (SDR) (e) Incremental Readout Weights

Figure 2: **Overview of NSI.** (a) Generator $\mathcal{G}$ (ICPG) produces B_f, M_f, λ_f, and C_f once; K synchronous neighborhood propagation (SNP) steps update the state from $h_0 = 0$, and readout sums $\omega_k \Delta h_k$ into z before output scaling and projection. (b) $\mathcal{G}$ with kernel and control heads (Appendix B.1). (c) One step, $h_k = \mathcal{L}_f(d_f + \mathcal{R}_f(h_{k-1}))$. (d) State-dependent retention (SDR) $\mathcal{R}_f$. (e) Readout weights ω_k from learned θ_1, θ_2. Amber chips mark learned static quantities.

3 Method

3.1 Notation and Overview

NSI is a token mixer for image restoration (Figure 2). A 1×1 convolution projects input features x to $f = W_{\text{in}}(x) \in \mathbb{R}^{C \times H \times W}$. The state h_k has the same shape as f, with $h_0 = 0$ and $k = 1, \ldots, K$ indexing propagation steps.

A shared parameter generation network $\mathcal{G}$ computes $(B_f, M_{\text{raw}}, C_f, \lambda_f) = \mathcal{G}(f)$ once. Its kernel head produces the raw interaction matrices M_{raw}, which Section 3.2 normalizes to M_f; its control head produces the input scaling B_f, the output scaling C_f, and the decay parameters λ_f. All of these input-conditioned quantities remain fixed across propagation steps.

Let $d_f = B_f \odot f$ denote the repeated input term, $\mathcal{R}_f(h) = \eta_f(h) \odot h$ the position-wise retention operator, and $\mathcal{L}_f$ the shared neighborhood aggregation operator. Each position reads the previous state from itself, its four axial neighbors, and its four diagonal neighbors, and all positions update synchronously:

$$h_k = \mathcal{L}_f(d_f + \mathcal{R}_f(h_{k-1}))\,. \tag{1}$$

At each step, the retention coefficients $\eta_f(h_{k-1})$ are recomputed from the evolving state before aggregation; thus $h_1 = \mathcal{L}_f(d_f)$. After K steps, incremental readout forms z, which is then scaled and projected to produce the output (Section 3.4). With the generated coefficients fixed, these steps reach at most a $(2K+1) \times (2K+1)$ region before boundary truncation; parameter generation and the backbone provide additional context.

3.2 Input-conditioned Interaction Kernels

The shared feature extractor in $\mathcal{G}$ uses a grouped 1×1 projection and 3×3 and 7×7 depthwise branches, and its context features feed both heads. NSI arranges the C channels into $C/2$ pairs. Each spatial connection transforms one pair; W_{in} and W_{out} mix channels across pairs outside propagation.

Let $v^{(c)}[p] \in \mathbb{R}^2$ denote pair c at position p, and let $\mathcal{N}_3$ be the set of nine neighborhood offsets. The aggregation at receiving position p is

$$(\mathcal{L}_f v)^{(c)}[p] = \sum_{\delta \in \mathcal{N}_3} M_f[c, \delta, p]\, v^{(c)}[p + \delta], \tag{2}$$

where $M_f[c, \delta, p] \in \mathbb{R}^{2\times 2}$ transforms the pair at sending position $p + \delta$, and values outside the grid are zero. The matrices vary across positions, pairs, and offsets but are shared across steps.

Propagation gain control. At each receiving position, the kernel head predicts two scalars $a = a_{\text{raw}}[c, \delta, p]$ and $b = b_{\text{raw}}[c, \delta, p]$ from the local context features. These scalars define a raw matrix, which is normalized jointly over offsets for each (c, p):

$$M_{\text{raw}}[c, \delta, p] = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}, \qquad M_f[c, \delta, p] = \frac{\rho\, M_{\text{raw}}[c, \delta, p]}{1 + \sum_{\delta' \in \mathcal{N}_3} \left\| M_{\text{raw}}[c, \delta', p] \right\|_2}. \tag{3}$$

Here, $\rho = 0.95$ by default and $\| \cdot \|_2$ is the spectral norm. For every pair and position, this scaling ensures $\sum_\delta \|M_f[c, \delta, p]\|_2 < \rho$, which bounds the propagation gain while preserving the relative strengths of the connections. Appendix B.1 gives the generator configuration.

3.3 Input Injection and State-dependent Retention

The control head generates one input scaling coefficient $B_f[c, p]$ per pair and position, which is broadcast to both channels of the pair. Injecting $d_f = B_f \odot f$ at every step keeps the input features available while context accumulates. Retention operates on groups of g channel pairs ($g = 4$ by default). For the state vector $v \in \mathbb{R}^{2g}$ of group j at position p, we define

$$\text{RMS}_g^2(v) = \frac{\|v\|_2^2}{2g}, \qquad \eta_{f,j,p}(v) = \left(1 + \lambda_f[j, p]\, \text{RMS}_g^2(v)\right)^{-1/2}. \tag{4}$$

The control head generates the nonnegative group decay $\lambda_f[j, p]$ once. At each step, $\eta_{f,j,p}$ is recomputed and broadcast to all $2g$ channels of the group, giving $\mathcal{R}_f(h) = \eta_f(h) \odot h$. When the decay is positive, a larger group RMS lowers retention; because the scaling is scalar, it preserves the direction of the group and the relative values of its channels. The pair-energy convention of Appendix B.2 uses $E_k = 2\,\text{RMS}_g^2(h_k)$ and $\lambda_f = 2\bar{\lambda}_f$, which yields the same retention.

Retention acts at each sending position $p + \delta$: the retained state is added to d_f at that position, transformed by $M_f[c, \delta, p]$, and summed at receiving position p. This is the composition in Eq. (1), which Appendix B.2 expands position by position.

3.4 Incremental Readout

As successive steps gather broader context, NSI combines the state increments:

$$z = \sum_{k=1}^{K} \omega_k \Delta h_k, \qquad \Delta h_k = h_k - h_{k-1}. \tag{5}$$

Summing the states directly counts their shared component repeatedly, whereas weighting the increments adjusts the net contribution of each step. With unit weights, the sum telescopes to $z = h_K - h_0 = h_K$; nonuniform weights change this combination (Appendix A.3).

For $K > 1$, the weights are generated from two learned scalars as $\omega_k = K\,\text{softmax}_k(\theta_1^2 \tau_k + \theta_2^2 \tau_k^2)$ with $\tau_k = (k-1)/(K-1)$; for $K = 1$, $\omega_1 = 1$. The weights are positive, nondecreasing, and sum to K. They do not depend on the input or the state and are shared across samples, positions, and channels. Appendix B.2 gives the initialization details.

The control head generates one output scaling coefficient $C_f[c, p]$ per pair and position, broadcast to both channels; the module output is $y = W_{\text{out}}(C_f \odot z)$. Section 3.5 bounds z before output scaling and projection.

3.5 Stability of Multi-step Propagation

For fixed f, we analyze the complete nonlinear map $\mathcal{T}_f(h) = \mathcal{L}_f(d_f + \mathcal{R}_f(h))$, including state-dependent retention. Define the maximum channel-pair norm $\|h\|_{\infty,2} = \max_{c,p} \|h^{(c)}[p]\|_2$ and actual kernel gain $q_f = \max_{c,p} \sum_\delta \|M_f[c, \delta, p]\|_2$. Equation (3) ensures $q_f < \rho < 1$; ρ is the design bound.

Proposition 1 (Contractivity and uniform magnitude bounds of multi-step propagation)**.** *Assume zero boundary conditions, parameters shared across propagation steps, $g \leq 4$ channel pairs per group, $\lambda_f \geq 0$, and $q_f \leq \rho < 1$. For any states h, v,*

$$\|\mathcal{T}_f(h) - \mathcal{T}_f(v)\|_{\infty,2} \leq q_f \|h - v\|_{\infty,2} \leq \rho \|h - v\|_{\infty,2}. \tag{6}$$

If $h_0 = 0$ and the readout weights are nonnegative, nondecreasing, and sum to K, then, for every $K \geq 1$,

$$\max\{\|h_K\|_{\infty,2}, \|z\|_{\infty,2}\} \leq \frac{q_f}{1 - q_f} \|d_f\|_{\infty,2} \leq \frac{\rho}{1 - \rho} \|d_f\|_{\infty,2}. \tag{7}$$

For $g \leq 4$, the retention Jacobian has induced norm at most one, so retention is non-expansive (Appendix A.2), and gain control then makes the complete update contractive. Applying this map repeatedly gives $\|\Delta h_k\|_{\infty,2} \leq q_f^{k-1} \|\Delta h_1\|_{\infty,2}$, which controls incremental readout under the stated weights. Appendix A proves these results and derives the state bound under weaker assumptions.

4 Experiments

4.1 Experimental Settings

Controlled comparison. To evaluate NSI as a token mixer, we replace VIVNet's complete token mixer, including its projections (Cui et al., 2026), and keep the backbone, reconstruction layer, normalization, residual connections, and gated feedforward branch fixed. Parameter and MAC counts cover the complete model, and MACs are measured on a $1 \times 3 \times 256 \times 256$ input.

We also compare MaIR and Spatial-Mamba within VIVNet on O-HAZE. Appendix D.1 reports operator replacements separately from full-block transfers, since the latter also change normalization, residual connections, and feedforward computation. Appendix D.2 tests eight input resolutions.

Training and evaluation. Models are trained from scratch, with data, augmentation, and budgets matched within each task (Table 1). The O-HAZE evaluation resolution was fixed before test inspection. We report the mean per-image RGB PSNR and SSIM and, for GoPro, the mean $\pm$ sample standard deviation across seeds. Appendices B.3–B.6 give the full protocols.

Table 1: Training and evaluation settings. Crop is the side length of the square training crop.

Dataset	Backbone	Train/Test	Steps	Crop	Seeds	Evaluation
GoPro	VIVNet-B	2,103/1,111	49,350	128	42–44	Original size
O-HAZE	VIVNet-S	40/5	55,800	256	42	Long side resized to 512
Haze4K	VIVNet-S	3,000/1,000	100,000	256	42	Original size

4.2 Image Restoration Results

Image deblurring. On GoPro, NSI improves mean PSNR over SS2D by 0.50 dB and achieves higher PSNR under each of the three random seeds (Table 2). Figure 3(a) illustrates sharper text boundaries and digit strokes in a motion-blurred region. Appendix C.2 reports the per-video results.

Table 2: GoPro deblurring on original-resolution images within VIVNet-B; three-seed mean $\pm$ standard deviation. Venue/year cites each design's origin; all variants are retrained.

Token mixer	Venue/year	PSNR (dB)↑	SSIM↑	Params (M)	MACs (G)
VIVNet-B (native)	TPAMI 2026	29.9024 ± 0.1205	0.896065 ± 0.002584	7.422	79.091
CSPN-style	ECCV 2018	30.2023 ± 0.0910	0.901086 ± 0.001702	5.792	59.091
SS2D	NeurIPS 2024	30.3674 ± 0.0685	0.904607 ± 0.001215	6.301	82.060
Ours	–	$\mathbf{30.8687 \pm 0.0482}$	$\mathbf{0.912960 \pm 0.000645}$	5.807	61.431

Image dehazing. NSI improves PSNR over SS2D by 1.15 dB on O-HAZE and 1.37 dB on Haze4K, and it also achieves higher SSIM on Haze4K (Table 3). The example in Figure 3(b) shows branch contrast and contours that match the target more closely.

Table 3: Dehazing within VIVNet-S: complete O-HAZE images at long side 512 and original-resolution Haze4K images. Venue/year cites each design's origin; all variants are retrained.

		O-HAZE		Haze4K			
Token mixer	Venue/year	PSNR (dB)↑	SSIM↑	PSNR (dB)↑	SSIM↑	Params (M)	MACs (G)
VIVNet-S (native)	TPAMI 2026	25.8096	0.90428	29.8088	0.980832	2.860	31.776
DWConv	CVPR 2022	25.9001	0.90154	29.6031	0.981382	2.849	28.356
Swin	ICCV 2021	25.5671	0.89718	28.0332	0.975555	2.942	34.517
CSPN-style	ECCV 2018	26.2017	0.90397	27.2493	0.973603	2.240	24.304
SS2D	NeurIPS 2024	25.6250	0.89792	28.7410	0.972697	2.470	36.697
Ours	–	**26.7755**	**0.90655**	**30.1147**	**0.982612**	2.248	25.613

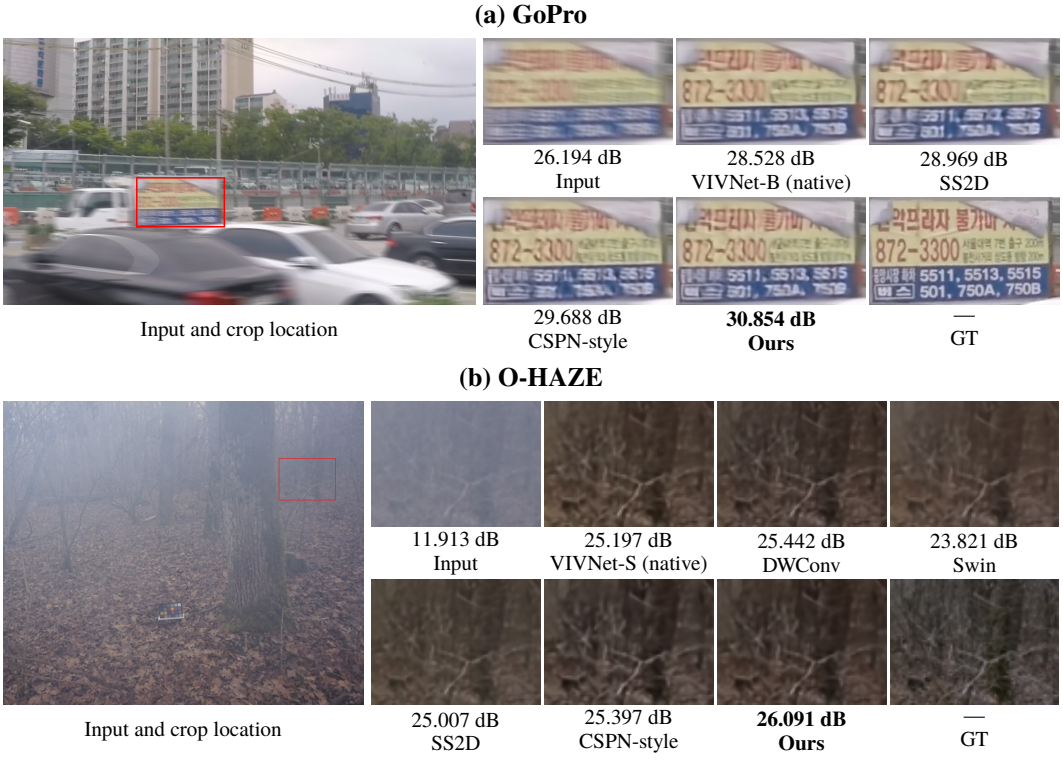


Figure 3: Restoration examples for (a) GoPro deblurring and (b) O-HAZE dehazing. Enlarged crops show the red-boxed input region; PSNR values (dB) are computed over the complete image.

4.3 Computational Efficiency

Compared with SS2D, NSI improves PSNR while reducing complete-model MACs by 25.1% for deblurring and 30.2% for dehazing (Tables 2 and 3). Compared with CSPN-style, it uses 4.0% and 5.4% more MACs for PSNR gains of 0.67 and 0.57 dB on GoPro and O-HAZE, respectively. Figure 4 compares module costs across spatial sizes, channel widths, and numbers of NSI propagation steps.

For paired inference on a 720×1280 GoPro image on an RTX 5090 (FP32, TF32 disabled), the median latencies across three independent processes are 0.494–0.496 s for NSI and 0.513–0.514 s for SS2D, a reduction of 3.4%–3.7% (Appendix B.6).

5 Ablation Studies and Analysis

This section examines NSI's propagation design, kernel sharing, state-dependent retention, incremental readout, and propagation gain control. Each structural ablation is trained independently with seed 42 for the prescribed number of training steps on each dataset. Diagnostics that modify the

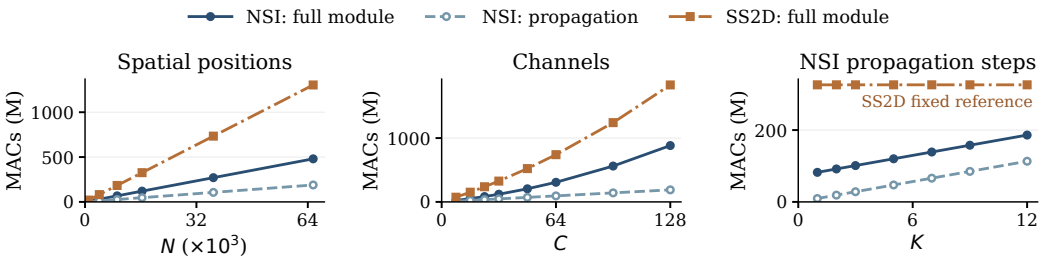


Figure 4: **Single-module MACs.** Both full modules scale linearly with spatial size, and their projections add a cost that is quadratic in the channel width. In the right panel, only the number of NSI steps varies, and SS2D serves as a fixed single-module reference. Appendix B.5 gives the settings.

computation of a trained model while keeping its parameters fixed are identified separately. In the tables, Δ PSNR is the result of a control minus that of the full model trained with the same seed. Appendix C defines the controls and reports the complete results.

5.1 Multi-step Neighborhood Interaction

On O-HAZE, PSNR peaks at five steps, while SSIM is slightly higher at seven (Table 4(a)). This supports the quality–cost trade-off of the five-step configuration, chosen before this analysis.

The five-layer feedforward stack uses the same number of neighborhood aggregation operations and similar MACs but has lower PSNR than NSI. This comparison changes kernel sharing, input injection, retention, and readout together, so it evaluates the overall propagation design. CSPN-style also achieves lower PSNR when equipped with NSI's kernel generator. Appendix C.3 examines input injection at every step, spatial propagation paths, and the timing of updates from diagonal neighbors. On one test image per dataset, NSI's effective receptive field is smaller and less axis-concentrated than SS2D's (Appendix C.4), so a larger receptive field does not explain its gains over SS2D.

5.2 Input-conditioned Interactions with Cross-step Sharing

Replacing input-conditioned interaction kernels with input-independent ones lowers PSNR on O-HAZE. Regenerating kernels from $f + h_k$ at every step raises complete-model MACs by 32.9% on O-HAZE and 24.5% on GoPro, yet lowers PSNR for dehazing and leaves deblurring PSNR similar (Table 4(b)). On Haze4K, it reaches 29.1378 dB PSNR (0.9769 dB below the full model) and 0.979345 SSIM. Kernel sharing therefore reduces repeated computation while keeping adaptation to the module input; retention coefficients are still recomputed at each step from the evolving state.

Table 4: Trade-offs between restoration quality and computational cost. Panel (a) varies the number of propagation steps on O-HAZE, and panel (b) compares propagation designs and kernel sharing. MACs are counted for the complete model on a 256×256 input.

(a) Propagation steps

K	PSNR↑	SSIM↑
1	25.8834	0.90186
3	26.4699	0.90455
5	**26.7755**	0.90655
7	26.3500	**0.90691**

(b) Propagation design and kernel sharing

Configuration	Dataset	Δ PSNR	MACs (G)
Full model	O-HAZE	0.0000	25.613
	GoPro	0.0000	61.431
	Haze4K	0.0000	25.613
Five-layer feedforward stack	O-HAZE	−0.2797	25.613
Input-independent shared kernels	O-HAZE	−0.6811	24.631
Per-step kernel regeneration	O-HAZE	−0.4427	34.037
	GoPro	+0.0562	76.498
	Haze4K	−0.9769	34.037
CSPN-style: original generator	O-HAZE	−0.5741	24.304
CSPN-style: replaced generator	O-HAZE	−2.8412	24.113

5.3 State-dependent Retention and Incremental Readout

The effects of retention and readout differ across tasks (Table 5). Independently trained controls show that state-dependent retention improves PSNR on O-HAZE but has a small effect on Haze4K and GoPro. In a separate diagnostic with the model parameters fixed, we replace each retention-coefficient map with its spatial mean and match the RMS of the projected module output. This change lowers the mean PSNR on O-HAZE by 0.2239 dB. Both PSNR and SSIM decrease on all five images, indicating that the trained model uses the spatial distribution of the retention coefficients.

Incremental readout improves PSNR on both dehazing datasets and performs similarly to final-state readout on GoPro. On Haze4K, combining state increments yields 0.2749 dB higher PSNR than combining complete states with nonnegative weights. Replacing both retention and readout reduces PSNR less than replacing readout alone, which indicates that their effects after retraining are not additive. On GoPro, the joint replacement changes PSNR by only -0.0145 dB. Appendix C.5 reports readout replacements with the parameters fixed, both with and without matched output magnitudes.

Table 5: PSNR changes (dB) from replacing components or propagation gain control. All configurations use 25.613 G MACs for dehazing and 61.431 G for GoPro; the GoPro joint-replacement difference uses unrounded values.

Configuration	O-HAZE	GoPro	Haze4K
Identity retention	-0.3326	$+0.0107$	-0.0668
Final-state readout	-0.4516	$+0.0208$	-0.7798
Identity retention + final-state readout	-0.5169	-0.0145	-0.1921
Without propagation gain control	-1.9712	-2.2770	-3.7400

5.4 Propagation Gain Control and State Evolution

NSI's propagation gain control enforces the bound on the aggregation operator used in Section 3.5. Models trained independently without this control have lower PSNR and SSIM on all three datasets (Table 5), which supports its effectiveness under the tested protocols. We also examine the internal propagation of the trained full model with its parameters held fixed.

With the norm defined in the theoretical analysis, we measure all 22 modules on the first five images in the fixed Haze4K test order. In every case, the actual kernel gain q_f is below 0.95. Over five propagation steps, the norms of the states, the state increments, and the readouts before output scaling and projection all remain within their respective bounds. Appendix C.6 reports these measurements and the largest ratios of the measured norms to their bounds.

Appendix C.6 gives O-HAZE state and increment RMS curves and tests learnable kernel scaling.

6 Conclusion

We propose Neighborhood State Interaction (NSI), a token mixer for efficient multi-step selective visual state propagation in image restoration. NSI combines synchronous neighborhood interactions with shared input-conditioned interaction kernels, repeated input injection, state-dependent retention, and incremental readout. By reusing spatial relations and adjusting how much accumulated state is passed on, NSI can use context selectively across propagation steps. Our analysis links channel grouping and propagation gain control to contraction of the complete nonlinear update at a fixed input, and controlled deblurring and dehazing experiments show that NSI achieves higher PSNR than SS2D with fewer MACs. We hope NSI inspires further research on efficient visual state propagation.

Reproducibility Statement

The main text specifies the NSI computation and the assumptions behind the theoretical results, with proofs in Appendix A. Appendix B provides parameter generation, model configurations, training and evaluation protocols, and resource measurements. Appendices C.1 and C.5 distinguish independently trained controls from interventions on trained models with fixed parameters. Appendix D.2 gives the protocol and results for varying input resolution. Supplementary records provide training configurations, per-image and per-video results, and run logs linking reported values to the corresponding experiments.

AI Use Statement

We used OpenAI Codex to assist with **writing and polishing**, including organizing sections and arguments, drafting and revising prose, arranging LaTeX figures and tables, and checking terminology. For **research ideation and execution**, it helped organize related literature, discuss mechanisms and experiments, develop and debug some experimental and diagnostic scripts, monitor runs, organize logs, and check results. For **mathematical proofs**, it assisted with developing proof ideas, derivations, mathematical checks, and checking consistency among assumptions, formulas, and implementation. The authors are responsible for verifying the references, code, results, and proofs and for the paper's final content.

References

Codruta O. Ancuti, Cosmin Ancuti, Radu Timofte, and Christophe De Vleeschouwer. O-HAZE: A dehazing benchmark with real hazy and haze-free outdoor images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 754–762, 2018.

Enis Baty, Alejandro Hernández Díaz, Rebecca Davidson, Chris Bridges, and Simon Hadfield. Mamba2D: A natively multi-dimensional state-space model for vision tasks. *arXiv preprint arXiv:2412.16146*, 2024. URL https://arxiv.org/abs/2412.16146v4. Version 4, revised March 2026.

Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple Baselines for Image Restoration. In *Computer Vision – ECCV 2022*, 2022. doi: 10.1007/978-3-031-20071-7_2. URL https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136670017.pdf.

Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proceedings of the European Conference on Computer Vision*, pp. 103–119, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Xinjing_Cheng_Depth_Estimation_via_ECCV_2018_paper.html.

Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2361–2379, 2020. doi: 10.1109/TPAMI.2019.2947374.

Yuning Cui, Wenqi Ren, Boxin Shi, and Alois Knoll. Visual-in-visual: A unified and efficient baseline for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48 (7):7981–7999, 2026. doi: 10.1109/TPAMI.2026.3669720.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.00396.

Jinyoung Jun, Jae-Han Lee, and Chang-Su Kim. Masked spatial propagation network for sparsity-adaptive depth refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19768–19778, 2024.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1637–1645, 2016. doi: 10.1109/CVPR.2016.181.

Boyun Li, Haiyu Zhao, Wenxin Wang, Peng Hu, Yuanbiao Gou, and Xi Peng. MaIR: A locality- and continuity-preserving mamba for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7491–7501, 2025.

Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. SwinIR: Image Restoration Using Swin Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2021. doi: 10.1109/iccvw54120.2021.00210. URL https://openaccess.thecvf.com/content/ICCV2021W/AIM/papers/Liang_SwinIR_Image_Restoration_Using_Swin_Transformer_ICCVW_2021_paper.pdf.

Yu-Cheng Lin, Yu-Syuan Xu, Hao-Wei Chen, Hsien-Kai Kuo, and Chun-Yi Lee. EAMamba: Efficient all-around vision state space model for image restoration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2506.22246.

Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic spatial propagation network for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1638–1646, 2022. doi: 10.1609/aaai.v36i2.20055.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021. doi: 10.1109/ICCV48922.2021.00986.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022. doi: 10.1109/CVPR52688.2022.01167.

Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3883–3891, 2017. doi: 10.1109/CVPR.2017.35.

Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-Local Spatial Propagation Network for Depth Completion. In *Computer Vision – ECCV 2020*, 2020. doi: 10.1007/978-3-030-58601-0_8. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123580120.pdf.

Chaodong Xiao, Minghan Li, Zhengqiang Zhang, Deyu Meng, and Lei Zhang. Spatial-mamba: Effective visual state space models via structure-aware state fusion. In *The Thirteenth International Conference on Learning Representations*, 2025.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. URL https://arxiv.org/abs/2102.02808.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5728–5739, 2022. doi: 10.1109/CVPR52688.2022.00564.

APPENDICES

The appendices provide proofs, implementation details, experimental protocols, and additional results. Appendix A analyzes multi-step propagation and incremental readout. Appendix B describes the module, training, evaluation, and resource measurements. Appendix C defines the controls and reports supplementary results and diagnostics. Appendix D.1 describes how other methods are incorporated into the backbone, while Appendix D.2 analyzes sensitivity to input resolution.

We use the real-valued channel pairs and transformation matrices defined in the main text. In the appendices, $g_f^{\text{in}} = B_f$ and $g_f^{\text{out}} = C_f$ denote input and output scaling, respectively; $d_f = B_f \odot f$ is the input injection term.

A THEORETICAL ANALYSIS AND PROOFS

We analyze the propagation under a fixed input f, zero boundary conditions, nonnegative decay, and parameters shared across steps. We use geometric sums, the Banach fixed-point theorem, summation by parts, and Chebyshev's sum inequality. State-dependent retention requires the separate non-expansiveness argument below.

A.1 BOUND ON THE INTERNAL STATE NORM

Write $d_f = B_f \odot f$ and $N(z) = \|z\|_{\infty,2} = \max_{b,c,p} \|z^{(c)}[b,p]\|_2$, where b, c, p index samples, channel pairs, and positions. Kernel normalization and $0 < \eta_k \leq 1$ give

$$\|\mathcal{L}_f z\|_{\infty,2} \leq \rho \|z\|_{\infty,2}, \tag{8}$$

$$\|\mathcal{R}_f(h_k)\|_{\infty,2} \leq \|h_k\|_{\infty,2}. \tag{9}$$

Here, $\mathcal{L}_f$ is aggregation, $\mathcal{R}_f$ is retention, and $\rho < 1$ bounds kernel gain. The second bound holds for any group size. With $s_k = N(h_k)$ and $d_\infty = N(d_f)$, iteration gives

$$s_{k+1} \leq \rho(d_\infty + s_k), \tag{10}$$

$$s_k \leq \rho^k s_0 + \sum_{r=1}^{k} \rho^r d_\infty = \rho^k s_0 + \frac{\rho(1-\rho^k)}{1-\rho} d_\infty. \tag{11}$$

Here, s_0 is the initial state norm and r indexes the geometric sum. Setting $s_0 = 0$ yields

$$\|h_k\|_{\infty,2} \leq \frac{\rho(1-\rho^k)}{1-\rho} \|d_f\|_{\infty,2} \leq \frac{\rho}{1-\rho} \|d_f\|_{\infty,2}. \tag{12}$$

Here, k is the step index and d_f is the repeated input term. For $\rho = 0.95$, the five-step and uniform coefficients are 4.2982 and 19.

A tighter bound replaces ρ by the actual gain $q_f = \max_{b,c,p} \sum_\delta \|M_f[b,c,\delta,p]\|_2 < \rho$, since $N(\mathcal{L}_f z) \leq q_f N(z)$.

A.2 CONTRACTIVITY OF THE COMPLETE NONLINEAR UPDATE

We differentiate the complete map $\mathcal{T}_f(h) = \mathcal{L}_f[d_f + \mathcal{R}_f(h)]$, including the dependence of retention on h.

The kernels, input term, and decay parameters are fixed for this input.

For a group state $x \in \mathbb{R}^{2g}$ containing g channel pairs, retention is

$$R(x) = \frac{x}{\sqrt{1 + \alpha \|x\|_2^2}}, \qquad \alpha = \bar{\lambda}_f / g \geq 0. \tag{13}$$

Here, α is fixed within each group and position. The group-energy parameter $\bar{\lambda}_f$ corresponds to $\lambda_f = 2\bar{\lambda}_f$ under the main-text RMS convention (Appendix B.2).

Proposition 2 (Complete nonlinear state propagation)**.** *For $g \leq 4$, $\mathcal{R}_f$ is globally non-expansive under N, so $\mathcal{T}_f$ is a q_f-contraction with a unique fixed point h_*. Any two trajectories and the increments of either trajectory satisfy*

$$N(h_k - v_k) \leq q_f^k N(h_0 - v_0), \tag{14}$$

$$N(\Delta h_k) \leq q_f^{k-1} N(\Delta h_1). \tag{15}$$

Here, h_k, v_k start from h_0, v_0, N is the maximum channel-pair norm, q_f is kernel gain, and $\Delta h_k = h_k - h_{k-1}$ for $k \geq 1$.

Proof. For $\alpha = 0$ or $x = 0$, $DR(x) = I$. Otherwise, differentiating gives

$$DR(x) = \sqrt{1-s}(I - suu^\mathsf{T}). \tag{16}$$

Here, DR is the Jacobian, I is the identity, $u = x/\|x\|_2$, and $s = \alpha\|x\|_2^2/(1+\alpha\|x\|_2^2) \in [0, 1)$. The reflection $H = I - 2uu^\mathsf{T}$ satisfies $N(Hv) \leq \|v\|_2 \leq \sqrt{g}\, N(v)$. Thus, $I - suu^\mathsf{T} = (1 - s/2)I + (s/2)H$ implies

$$\begin{aligned} \|DR(x)\|_N &\leq \sqrt{1-s}\left[1 + \frac{\sqrt{g}-1}{2}s\right] \\ &\leq \sqrt{1-s}(1 + s/2) \leq 1. \end{aligned} \tag{17}$$

Here, $\|\cdot\|_N$ is the induced matrix norm. The last two inequalities use $g \leq 4$ and $(1-s)(1+s/2)^2 \leq 1$. Integrating the Jacobian along a line segment proves groupwise non-expansiveness; taking the maximum extends it to the grid.

Composition with aggregation gives

$$N(\mathcal{T}_f(h) - \mathcal{T}_f(v)) \leq q_f N(\mathcal{R}_f(h) - \mathcal{R}_f(v)) \leq q_f N(h - v). \tag{18}$$

Here, h, v are arbitrary states. The Banach fixed-point theorem applies because this finite-dimensional normed space is complete and $q_f < 1$. Iterating the contraction proves both bounds, using $\Delta h_{k+1} = \mathcal{T}_f(h_k) - \mathcal{T}_f(h_{k-1})$ for the second. NSI uses a finite number of steps, not a fixed-point solver.

$g \leq 4$ and the mixed norm. For positive decay, $g \leq 4$ is also necessary for global non-expansiveness of retention in this norm. For $g > 1$, choose a unit vector $e \in \mathbb{R}^2$, positive a, b with $a^2 = (1-1/\sqrt{g})/2$, $b^2 = (1-a^2)/(g-1)$, and set $u = (ae, be, \ldots, be)$, $v = (e, -e, \ldots, -e)$. Choose a nonzero state x such that $x/\|x\|_2 = u$. Then $\|u\|_2 = N(v) = 1$, and the first pair of $DR(x)v$ has magnitude

$$\sqrt{1-s}\left[1 + \frac{\sqrt{g}-1}{2}s\right]. \tag{19}$$

Here, s is the normalized energy above. The right derivative of this expression with respect to s at zero is $(\sqrt{g}-2)/2 > 0$ for $g > 4$, giving expansion for sufficiently small nonzero states whenever $\alpha > 0$. Larger groups may still yield a contractive complete update if aggregation compensates; $\alpha = 0$ gives identity retention for any g.

A.3 Equivalent Form and Norm Bound for Incremental Readout

Discrete summation by parts gives the equivalent state-based readout

$$\sum_{k=1}^{K} \omega_k(h_k - h_{k-1}) = \omega_K h_K - \omega_1 h_0 + \sum_{k=1}^{K-1} (\omega_k - \omega_{k+1})h_k. \tag{20}$$

Here, h_k is the state at step k, ω_k its increment weight, and K the step count. With $h_0 = 0$, unit weights give h_K; nondecreasing weights give nonpositive coefficients on intermediate states. The identity holds for any state sequence.

Proposition 3 (Bound on the incremental readout norm)**.** *Under Proposition 2, let $h_0 = 0$ and let ω_k be nonnegative, nondecreasing, and sum to K. The readout before output scaling and projection satisfies*

$$N\left(\sum_{k=1}^{K} \omega_k \Delta h_k\right) \le N(d_f) \sum_{k=1}^{K} \omega_k q_f^k \le N(d_f) \frac{q_f(1 - q_f^K)}{1 - q_f}. \tag{21}$$

Here, N is the maximum channel-pair norm, $q_f < 1$ is kernel gain, d_f is the input term, and $\Delta h_k = h_k - h_{k-1}$. The bound includes $q_f = 0$.

Proof. Since $\Delta h_1 = \mathcal{L}_f d_f$, Eq. (15) gives $N(\Delta h_k) \le q_f^k N(d_f)$. The triangle inequality yields the first bound. Chebyshev's sum inequality for oppositely ordered sequences gives $\sum_k \omega_k q_f^k \le (\sum_k \omega_k / K) \sum_k q_f^k = \sum_k q_f^k$, proving the second by a geometric sum. Together with the state bound, this proves Proposition 1.

A.4 Bound on the Previous State's Contribution

Positive decay also bounds the contribution of arbitrary previous states.

Proposition 4 (Difference between state-dependent retention and identity retention)**.** *On the finite grid, assume $\alpha_{b,j,p} = \bar{\lambda}_f[b, j, p]/g > 0$. Set $\alpha_{\min} = \min_{b,j,p} \alpha_{b,j,p}$ and $S_f = \alpha_{\min}^{-1/2}$. For any previous state h,*

$$N\big(\mathcal{T}_f(h) - \mathcal{T}_f(0)\big) \le q_f S_f, \tag{22}$$

$$N\big(\mathcal{T}_f(h)\big) \le q_f \big[N(d_f) + S_f\big]. \tag{23}$$

Here, b, j, p index samples, groups, and positions, q_f is kernel gain, and S_f bounds retained magnitudes. Identity retention admits no such bound over arbitrary h when $\mathcal{L}_f \neq 0$, even if $q_f < 1$.

Proof. For a group state x with decay scale α,

$$\|[R(x)]^{(c)}\|_2 \le \frac{\|x\|_2}{\sqrt{1 + \alpha \|x\|_2^2}} \le \alpha^{-1/2} \le S_f. \tag{24}$$

Here, $[R(x)]^{(c)}$ is retained pair c. Thus $N(\mathcal{R}_f(h)) \le S_f$; applying the aggregation gain bound proves both claims. For identity retention, scaling any state with nonzero image under $\mathcal{L}_f$ makes its contribution arbitrarily large.

For the zero-initialized trajectory, let $d_\star = N(d_f)$ and $\phi(r) = r/\sqrt{1 + \alpha_{\min} r^2}$. Group energy dominates each pair's energy, so $N(\mathcal{R}_f(h)) \le \phi(N(h))$. Monotonicity of ϕ and induction give

$$b_0 = 0, \qquad b_{k+1} = q_f[d_\star + \phi(b_k)] \quad \Longrightarrow \quad N(h_k) \le b_k. \tag{25}$$

Here, b_k bounds the state norm, $d_\star$ is the input norm, and ϕ bounds retention. Since $0 < \phi'(r) \le 1$ and $q_f < 1$, Banach's theorem on $[0, \infty)$ gives a unique fixed point B. If $q_f, d_\star > 0$, then $0 < B < U = q_f d_\star/(1 - q_f)$ because $q_f[d_\star + \phi(U)] < U$, giving a tighter bound than identity retention.

Softplus gives positive decay for finite real inputs. However, S_f depends on the input and learned parameters and grows as decay approaches zero. Equation (23) itself needs no $q_f < 1$ assumption; contraction and readout bounds do. Table 5 separately tests restoration after training.

A.5 Parameter Sharing, Scope of the Guarantees, and Numerical Error

A shared map is essential for increment decay. Alternating $T_+(h) = \frac{1}{2}(1 + h)$ and $T_-(h) = -\frac{1}{2}(1 + h)$ gives a cycle at $1/5, -3/5$, despite both maps being $1/2$-contractions.

A state-dependent kernel needs additional derivative control: with $R(h) = h/\sqrt{1+h^2}$ and $L(h) = 0.95\tanh(2h)$, $T(h) = L(h)[1 + R(h)]$ has $T'(0) = 1.9$ although $|L(h)| < 0.95$. Table 4 tests the per-step regeneration defined in Appendix C.1.

For computed states $\widetilde{h}_{k+1} = \mathcal{T}_f(\widetilde{h}_k) + e_k$ with $N(e_k) \le \varepsilon_k$, iterating the contraction inequality gives

$$N(\widetilde{h}_k - h_k) \le q_f^k N(\widetilde{h}_0 - h_0) + \sum_{j=0}^{k-1} q_f^{k-1-j}\varepsilon_j. \tag{26}$$

Here, h_k is the exact state, ε_j bounds step-j error, and q_f is the contraction factor. Equal initial states and $\varepsilon_j \le \varepsilon$ give an error bound $\varepsilon(1-q_f^k)/(1-q_f) \le 20\varepsilon$ for $q_f \le 0.95$.

Identity retention also contracts under the same gain constraint, so contraction alone does not explain its empirical difference from state-dependent retention. All guarantees concern a fixed module input and the readout before C_f and W_{out}.

B Implementation and Experimental Protocols

B.1 Tensor Layout and Parameter Generation

The input shape is $B \times C \times H \times W$, denoting batch size, channels, height, and width. In the code, `Bin` and `Cout` implement the projections W_{in} and W_{out} before and after propagation, respectively. The operations `write` and `read` implement the input and output scaling coefficients B_f and C_f. The projections mix channels and change their number, whereas the scaling operations multiply each channel pair by a coefficient. After projection, the channels form $C/2$ pairs; the features f and states remain real-valued. Each group contains $g = 4$ channel pairs, and a grouped 1×1 convolution maps $2g$ real-valued components to g hidden features to reduce generation cost. The shared feature extractor combines 3×3 and 7×7 depthwise branches to provide nearby and broader local context for both the kernel and control heads:

$$z_{\text{ctx}} = \text{GELU}(\text{DWConv}_{3\times3}(z_{\text{in}})) + \gamma_{\text{ctx}} \odot \text{GELU}(\text{DWConv}_{7\times7}(z_{\text{in}})), \tag{27}$$

Here, z_{in} is the output of the grouped 1×1 convolution, z_{ctx} is the resulting context feature, DWConv denotes depthwise convolution, and GELU is the activation function. The learned per-channel coefficient γ_{ctx} weights the 7×7 branch. The kernel head produces $9C$ channels, reshaped into two tensors of shape $B\times(C/2)\times9\times H\times W$. Their scaled values, a_f and b_f, define each transformation. All nine offsets enter Eq. (3), with gain limit $\rho = 0.95$. A control head generates pairwise scaling coefficients and decay parameters:

$$g_f^{\text{in/out}} = 1 + 0.25\tanh(\widehat{g}_f^{\text{in/out}}), \quad \lambda_f^{\text{pair}} = \text{softplus}(\widehat{\lambda}_f^{\text{pair}}), \quad \bar{\lambda}_f[j,p] = g^{-1}\sum_{c\in\mathcal{C}_j} \lambda_f^{\text{pair}}[c,p]. \tag{28}$$

Here, hats denote the control head's raw outputs; $\text{softplus}(t) = \log(1+e^t)$ makes each pair's decay positive. The set $\mathcal{C}_j$ contains the g channel pairs in group j, p is a spatial position, and $\bar{\lambda}_f[j,p]$ is their mean decay. The sample index is omitted. The coefficients $g_f^{\text{in}} = B_f$ and $g_f^{\text{out}} = C_f$ scale input and readout pairs. Kernels, $d_f = B_f \odot f$, decay parameters, and output scaling are computed once; retention coefficients change with the state.

B.2 Synchronous Updates, State-dependent Retention, and Incremental Readout

At each position, NSI computes a retention coefficient from the average energy of the channel pairs in that group:

$$E_k[j,p] = g^{-1}\sum_{c\in\mathcal{C}_j} \|h_k^{(c)}[p]\|_2^2, \qquad \eta_k[j,p] = (1 + \bar{\lambda}_f[j,p]E_k[j,p])^{-1/2}. \tag{29}$$

Here, $E_k[j,p]$ is the mean squared pair norm in group j at position p after step k, and $\eta_k[j,p]$ is its retention coefficient. The vector $h_k^{(c)}[p]$ contains the two components of pair c. Since main-text RMS averages over $2g$ real channels, $E_k = 2\,\text{RMS}_g^2(h_k)$ and $\lambda_f = 2\bar{\lambda}_f$. Every position reads the same previous state grid, with zero values outside it. Each connection applies a 2×2 matrix; offsets are

ordered as the center followed by the other eight in row order. With $h_0 = 0$ and generated coefficients fixed, K updates allow information to propagate across a $(2K+1) \times (2K+1)$ neighborhood before boundary truncation. A sending position uses one coefficient for each group, preserving relative channel values. Appendix A.2 includes the coefficient's state derivative.

To make the sending and receiving positions explicit, let $\eta_f^{(c)}(h)[s]$ denote the retention coefficient for the group containing pair c at position s. The update becomes

$$h_k^{(c)}[p] = \sum_{\delta \in \mathcal{N}_3} M_f[c, \delta, p] \left[\eta_f^{(c)}(h_{k-1})[p+\delta]\, h_{k-1}^{(c)}[p+\delta] + d_f^{(c)}[p+\delta] \right]. \tag{30}$$

Here, p is the receiving position, $p + \delta$ is the sending position, $\mathcal{N}_3$ contains the nine offsets, and $M_f[c, \delta, p]$ transforms pair c. The factor $\eta_f^{(c)}$ retains the previous pair state, while $d_f^{(c)}$ injects input features. All terms on the right are evaluated before updating h_k.

For $K > 1$, two learnable scalars determine the readout weights for the propagation steps:

$$\tau_k = \frac{k-1}{K-1}, \qquad \omega_k = K \ \mathrm{softmax}_k\big(\theta_1^2 \tau_k + \theta_2^2 \tau_k^2\big). \tag{31}$$

Here, τ_k is the normalized step index, θ_1, θ_2 are learned scalars, and $\mathrm{softmax}_k$ normalizes over all K steps. The readout weights are positive, have mean one, and are nondecreasing with the step index. They are shared across samples, positions, and channels and do not depend on the current state. The squared tilt coefficients θ_1^2 and θ_2^2 are initialized to 0.20 and 0.01; for $K = 1$, we set $\omega_1 = 1$. At each step, we accumulate $\omega_k(h_k - h_{k-1})$. After scaling the result by C_f, we arrange the first components of all channel pairs before their second components and apply W_{out} to restore the input shape. Inference maintains the current state and accumulated readout; training stores or recomputes intermediate quantities for backpropagation. Since $h_0 = 0$, retention can be omitted at the first step. The identity retention control instead sets every retention coefficient to one at every step.

B.3 Backbone and Comparison Module Configurations

We use the backbone widths and depths from the official VIVNet implementation (Cui et al., 2026). Dehazing on O-HAZE and Haze4K uses VIVNet-S (Small), with base width $b = 32$ and $(2, 3, 4, 4, 3, 2, 4)$ blocks across seven stages, giving 22 token mixers. GoPro deblurring uses VIVNet-B (Base) from the official all-in-one implementation, with $b = 48$ and $(3, 3, 5, 5, 3, 3, 4)$ blocks, giving 26 modules. The three encoder stages, three decoder stages, and refinement stage have widths $(b, 2b, 4b, 4b, 2b, b, b)$. The original paper combines the final decoder and refinement stages and reports $(2, 3, 4, 4, 3, 6)$ and $(3, 3, 5, 5, 3, 7)$ blocks for Small and Base, respectively; these specify the same configurations. Models are trained separately for each task, and methods within a task share the backbone configuration. Supplementary records identify the official source paths and configurations used.

All core-module comparisons preserve VIVNet's surrounding block. Its reconstruction layer predicts an RGB residual with a 3×3 convolution and adds it to the input image. We retain normalization, residual connections, and the gated feedforward branch with expansion ratio 3, replacing the token mixer between them. NSI includes W_{in}, parameter generation, propagation, incremental readout, C_f, and W_{out}. SS2D uses VMamba's four-direction scan, state width 16, inner width $\lfloor 0.75C \rfloor$, time-step rank $\lceil C/16 \rceil$, a 3×3 depthwise convolution, and the oflex backend (Liu et al., 2024). For dehazing, DWConv uses a ConvNeXt-style 7×7 depthwise convolution with pointwise expansion and projection at ratio 3 (Liu et al., 2022). Swin attention uses 7×7 windows, expansion ratio 1.75, alternating shifts of 0/3, and C/b heads (Liu et al., 2021). MaIR and Spatial-Mamba are evaluated either as mixing operators within this common block or as their complete official blocks. Appendix D.1 explains these distinct comparisons.

Appendix C.1 specifies how CSPN-style generates coefficients and propagates states.

Table 6: Training configurations for each dataset. GoPro uses four gradient-accumulation steps with batch size 8, giving an effective batch size of 32. O-HAZE test images have IDs 41–45.

Configuration	GoPro	O-HAZE	Haze4K
Training/test image pairs	2,103 / 1,111	40 / 5	3,000 / 1,000
Training crop	128×128	256×256	256×256
Batch size	32 (4×8)	2	2
Training precision	BF16	FP32	FP32
Training steps	49,350	55,800	100,000
Random seeds	Main comparison: 42, 43, 44; component ablations: 42	42	42
Optimizer	AdamW	AdamW	AdamW
Initial learning rate	2×10^{-4}	10^{-3}	10^{-3}
Weight decay	0.01	10^{-3}	10^{-3}
Momentum coefficients	$(0.9, 0.999)$	$(0.9, 0.9)$	$(0.9, 0.9)$
Warm-up	Linear warm-up for the first 4,935 updates	None	None
Cosine schedule	$T_{\max} = 59,220$; minimum learning rate 0	Minimum learning rate 10^{-7}	$T_{\max} = 100,000$; minimum learning rate 10^{-7}
Gradient clipping	None	Gradient L2 norm capped at 0.01	Gradient L2 norm capped at 0.01

B.4 Data, Training, and Evaluation Protocols

On each dataset, models are trained from scratch for a fixed number of steps, as specified in Table 6. The main GoPro comparison uses three independent random seeds; GoPro component ablations and all O-HAZE configurations use seed 42. Reported standard deviations are computed across training runs.

All three tasks use paired random crops, flips, and rotations and combine pixel L1 loss with a frequency-domain loss weighted by 0.1. On GoPro, the L1 term supervises every output, while the frequency-domain term supervises only the final output. GoPro also uses a dataset enlargement ratio of 5 and paired gamma and saturation augmentation. Supplementary records provide the full augmentation settings, optimizer configurations, and launch commands.

GoPro evaluation uses one forward pass over each original 720×1280 image in FP32 with TF32 disabled, without resizing or tiling. We clamp predictions to $[0, 1]$, compute RGB PSNR and SSIM, and average equally over all 1,111 frames. Component ablations follow the same protocol. A separate test of tiled inference uses core tiles of size 384, with extension and overlap both set to 64. The O-HAZE evaluation resolution was fixed before examining test results. We resize input and target to a long side of 512 pixels, preserving aspect ratio with antialiased bicubic interpolation and corner alignment disabled. We apply reflection padding on the right and bottom to multiples of 32, run FP32 inference on the complete image, and remove the padding. Predictions are clamped to $[0, 1]$; targets retain their interpolated values. RGB PSNR uses full-image MSE. SSIM uses an 11×11 Gaussian window with standard deviation 1.5, dynamic range 1, and $K_1 = 0.01, K_2 = 0.03$, without further pooling or resizing. Both metrics are averaged equally over five images.

Haze4K dataset. Haze4K contains 3,000 training pairs and 1,000 test pairs. All configurations use seed 42 and the same sample order, paired crops, flips, and rotations, with no color augmentation. Training uses 256×256 crops and batch size 2. AdamW starts at learning rate 10^{-3}, with momentum $(0.9, 0.9)$, weight decay 10^{-3}, and gradient L2 norm clipped to 0.01. A cosine schedule without warm-up reduces the learning rate to 10^{-7} over 100,000 steps. The loss combines pixel L1 with a frequency-domain term weighted by 0.1. This term averages the absolute real and imaginary components of the difference between the two-dimensional Fourier transforms of the prediction and target.

Table 7: Training environments and checkpoint resumption for supplementary controls. Each run uses its dataset's prescribed training budget. Every GoPro component control is trained once.

Control	Environment and resumption records
O-HAZE full delay	Completed the same total budget; model, optimizer, and scheduler were restored at step 45,000, while the data loader and random states were reinitialized.
GoPro half delay	Independently trained with the same random seed and initialization procedure, retaining gradients through the historical branch.
Four GoPro controls	Identity retention, final-state readout, half-delayed diagonal states, and removal of propagation gain control were all trained on an RTX 5090 under Linux with PyTorch 2.7.0/CUDA 12.8.
GoPro joint control	Identity retention + final-state readout used an RTX 5090 with PyTorch 2.10.0/CUDA 13.0, retaining seed 42, 49,350 steps, logical batch 32, and crop 128. Training used BF16; full-image evaluation used FP32 with both TF32 options disabled.
GoPro per-step regeneration	RTX 5090; transferred from Linux to Windows at step 33,079, using activation recomputation. Model, optimizer, scheduler, and main-process random states were restored; augmentation worker random sequences restarted.

Haze4K uses the backbone specified in Appendix B.3. Both NSI and the CSPN-style baseline run five propagation steps; the baseline uses its original kernel generator. The CSPN-style: replaced generator control uses NSI's network for generating contextual features but retains the anchored propagation defined in Appendix C.1. The Without propagation gain control variant removes only kernel coefficient normalization and is trained independently for 100,000 steps with seed 42. Its final checkpoint uses the same evaluation protocol. All Haze4K training and evaluation use FP32 with TF32 disabled. For the full NSI model, custom forward and backward passes fuse the shared input heads with propagation; each control retains its verified implementation and prescribed model definition.

We evaluate Haze4K images at their original resolution. Reflection padding extends only the right and bottom to multiples of 4 and is removed after inference. Predictions are clamped to $[0, 1]$. PSNR uses RGB MSE over the complete image. SSIM uses valid 11×11 Gaussian windows with standard deviation 1.5 and $K_1 = 0.01, K_2 = 0.03$ and is averaged over RGB channels without resizing. Both metrics are then averaged equally over all 1,000 images. Evaluation uses no test-time augmentation or model ensemble, and test scores are not used to reselect checkpoints.

Visual examples. Figure 3(a) shows GoPro test image `GOPR0410_11_00/000219.png`, evaluated at its original resolution using the final seed-43 checkpoint from the main comparison. Panel (b) shows O-HAZE test image 42 using the checkpoints in Table 3, after resizing the long side to 512 pixels while preserving aspect ratio. All methods show the same crop and report PSNR for the complete image.

Training environments and checkpoint resumption. The additional GoPro controls in Table 8 use seed 42 and the same initialization procedure. Each is trained for 49,350 steps under the common GoPro data protocol and evaluated on complete images at their original resolution. The reference is the existing seed-42 full model from the main three-seed experiment. Per-step kernel regeneration raises MACs from 61.431 to 76.498 G.

B.5 Parameter and Computation Counts

We report parameter counts for the complete model and MACs for a batch of one 256^2 image. The MAC count includes convolutions, linear layers, matrix multiplications, and explicit propagation; selective scans use the official estimate. Normalization, pooling, interpolation, activations, elementwise gating, padding, and tensor rearrangement are excluded consistently across methods. All models use the same input shape and counting rules.

Module scaling. The three sweeps in Figure 4 fix $(C, K) = (32, 5)$, $(H, W, K) = (128, 128, 5)$, and $(H, W, C) = (128, 128, 32)$, respectively, with batch size one. The figure uses the same NSI and SS2D module definitions as the main comparison, including their input/output projections and ex-

cluding the shared backbone and feedforward branch. NSI uses groups of four channel pairs. SS2D uses four directions, state dimension 16, an expansion ratio of 0.75, $\lceil C/16 \rceil$ time-step projection rank, and a 3×3 depthwise convolution. We count NSI's executed convolutions and neighborhood transforms. For SS2D, we count its surrounding projections and add the official selective-scan estimate; these are operation counts, not latency measurements.

Both full modules scale linearly with the number of spatial positions. NSI's repeated neighborhood propagation is linear in channel count, while full-module projections add quadratic channel cost. Increasing NSI's step count adds the same propagation cost per step because kernel generation and input/output projections are reused. The fixed SS2D reference in the last panel is one module, not a stack of K modules or a comparison at a matched propagation range.

B.6 Optimized Implementation, Latency, and Numerical Verification

The optimized inference implementation jointly computes the coefficient and control heads from their shared hidden features. A fused recurrence kernel performs 3×3 neighborhood aggregation and accumulates state increments. It preserves the trained parameters, synchronous updates, zero padding, and groups of four channel pairs. Retention is omitted at the first step because the initial state is zero; later steps retain the original computation. In contrast, the identity retention ablation sets every coefficient to one. Output scaling is applied at the final step. Readout weights are cached and refreshed whenever model parameters are loaded or modified. The model automatically uses this implementation for CUDA FP32 evaluation with gradients disabled, with no separate preparation call. It processes complete images by default. GoPro and O-HAZE training use the original forward implementation; the Haze4K full model uses the training implementation described in Appendix B.4.

Latency is measured on a shared Windows workstation with an RTX 5090, PyTorch 2.10.0/CUDA 13.0, and Triton 3.6.0. Both models use FP32 with TF32 disabled and process the same first GoPro test frame. After warm-up, each of three independent processes performs twelve paired measurements, keeping both models in GPU memory, balancing their execution order, and reusing compilation caches. For a complete 720×1280 image, NSI's median CUDA latencies are 0.4953, 0.4961, and 0.4935 seconds; SS2D's are 0.5144, 0.5136, and 0.5127 seconds in the corresponding processes, a reduction of 3.41–3.74%. Each measurement covers one forward pass and excludes image decoding, data transfer, and metric computation. Wall-clock medians show the same ordering, and the peak increase in GPU memory per call is approximately 7604 MiB for both models.

We verify the optimized implementation against the reference using identical model weights on 55 GoPro images. Before clamping, the largest output difference is 1.26×10^{-6}, and every element satisfies the tolerance $10^{-5} + 10^{-4}|y_{\text{ref}}|$. The largest per-image changes in PSNR and SSIM are 2.39×10^{-6} dB and 1.20×10^{-7}. Tests also cover batched rectangular inputs, refresh of cached weights, and returning to training after inference. The quality tables report the verified reference implementation on the complete test sets; numerical comparison of the optimized implementation is limited to these 55 images. Supplementary records retain earlier tests and timings of tiled inference under their original protocols.

Supplementary records also contain timings from each seed's original evaluation run before optimization. These report mean synchronized forward-pass wall time, excluding image decoding, data transfer, and metric computation, but include first-call overhead and variation in workstation load. They are therefore distinguished from the paired measurements on one fixed image above. Counts of propagation steps describe state dependencies, not GPU execution, memory access, or implementation efficiency. The latency claims in the main text rely only on the paired measurements of the complete models.

C Additional Results and Mechanism Analysis

C.1 Definitions of Independently Trained Controls

We define the structural controls in Tables 4–5. Each is trained independently from scratch, keeping the dataset, backbone, training budget, seed, and evaluation protocol unchanged except where specified. Results use the final checkpoint of each run. Appendix C.5 separately reports interventions on the trained full model with its parameters fixed.

Propagation steps. We selected $K = 5$ before testing sensitivity to propagation depth. Table 4 reports models trained independently with $K = 1, 3, 5, 7$.

Five-layer feedforward stack and component replacement. The five-layer feedforward stack tests whether repeated aggregation alone explains NSI's performance. It retains the input and output projections, control quantities, context generator, and kernel normalization. The kernel generator is shared across layers, but each layer adds a learned embedding to its coefficients before normalization. Starting from $v_0 = d_f$, the stack applies $v_{k+1} = \mathcal{L}_{a_k(f)} v_k$ and reads out v_5. It omits state-dependent retention, repeated input injection, and additional GELU activations between layers. Both models perform the same five neighborhood aggregations. The stack and NSI have 2,252,300 and 2,248,140 parameters and require 25.612817 and 25.612517 G MACs, respectively. Because sharing, retention, input injection, and readout change together, Table 4 compares complete propagation designs rather than isolating one component.

Input conditioning and repeated input injection. The Input-independent shared kernels control uses learned spatial coefficients that do not depend on the input. The First-step input injection only control omits d_f after the first step.

CSPN-style and generator. The CSPN-style baseline maintains a real-valued state with C channels. A 1×1 projection with bias produces the initial state u, and a channel-wise 3×3 convolution predicts coefficients for the eight offsets excluding the center. We divide these coefficients by the sum of their absolute values plus 10^{-6} and set the center coefficient to $a_0 = 1 - \sum_\delta a_\delta$. The model then performs five updates:

$$h_0 = u, \qquad h_{k+1}[c, p] = a_0[c, p]u[c, p] + \sum_{\delta \in \mathcal{N}_3 \setminus \{(0,0)\}} a[c, \delta, p] h_k[c, p + \delta]. \tag{32}$$

Here, u is the projected input and fixed anchor; $h_k[c, p]$ is channel c at position p after step k. The coefficients $a_0[c, p]$ and $a[c, \delta, p]$ weight the anchor and neighboring states, and $\mathcal{N}_3 \setminus \{(0, 0)\}$ excludes the center. CSPN-style shares these coefficients across steps, assigns zero to states outside the grid, and projects the fifth state to form the output. To compare kernel generators, we retain this update rule, normalization, and projections but substitute NSI's grouped hidden features and $3 \times 3/7 \times 7$ context branches from Appendix B.1. The substituted head predicts $8C$ coefficients. Common parameters are reconstructed with the original constructor and random seed. These variants are named CSPN-style: original generator and CSPN-style: replaced generator.

The original-generator and replaced-generator variants contain 2.240 M and 2.231 M parameters, respectively, compared with 2.248 M for NSI.

Per-step kernel regeneration. The Per-step kernel regeneration control tests the cost and benefit of adapting spatial kernels to the evolving state. At each step it computes $a_k = \mathsf{N}_\rho(G(f + h_k))$ and updates $h_{k+1} = \mathcal{L}_{a_k}[d_f + \mathcal{R}_f(h_k)]$. Here, a_k denotes the step's normalized kernels and N_ρ applies Eq. (3). The generator G shares its parameters across steps, and the other control quantities and readout remain unchanged. This preserves the parameter count but increases MACs to 34.036777 G.

State-dependent retention and readout controls. The retention and readout controls isolate different parts of the update. Identity retention fixes all retention coefficients at one. Final-state readout sets every increment weight to one, which yields h_K from zero initialization, and the joint control makes both changes. Weighted state readout instead combines complete states using nonnegative weights that sum to one. Each Haze4K variant is trained independently with seed 42 for 100,000 steps under the same data and optimization protocol (Appendix B.4).

The GoPro joint control changes only retention and readout, is trained independently with seed 42 for 49,350 steps, and is evaluated on all 1,111 test images at their original resolution. It achieves PSNR/SSIM of 30.8772/0.912575. The reference is the existing seed-42 full model, not a newly trained run. Their unrounded PSNR values are 30.8772027053 and 30.8917520864, respectively, so the difference is -0.0145493811 dB, reported as -0.0145 dB.

Timing of contributions from diagonal neighbors. To test when diagonal information enters an update, define $q_k = d_f + \mathcal{R}_f(h_k)$. The fully delayed control uses $h_{k+1} = \mathcal{L}_{\text{ac}}(q_k) + \mathcal{L}_{\text{diag}}(q_{k-1})$

with $q_{-1} = 0$, where ac denotes the axial offsets and center. Delaying diagonal contributions by one step preserves MACs but reduces the number of positions reachable after five steps from 121 to 61.

The half-delayed control leaves the first update unchanged. From the second update onward, it uses $(q_k + q_{k-1})/2$ for diagonal contributions and q_k for axial and center contributions. This preserves the 121 reachable positions.

Propagation gain control and learnable kernel scaling. The Without propagation gain control variant uses the raw generated coefficients without normalization and preserves the rest of the model. Learnable kernel scaling instead retains normalization and multiplies each module's kernels by a positive learnable scalar initialized to one. This adds 22 parameters and removes a fixed upper bound on the total gain. It therefore tests a different modification from removing normalization entirely.

C.2 Complete Results Across Training Runs and Test Images

Table 8 summarizes the component ablations and other controls; Table 14 reports learnable kernel scaling. The main GoPro comparison reports the mean and standard deviation across three independent seeds. Each ablation and its Full model reference use one seed-42 run; different checkpoints from a run are not treated as independent repetitions. O-HAZE and Haze4K also use the final checkpoints of the prescribed seed-42 runs.

Kernel generator comparison on O-HAZE. Table 4 compares the two CSPN-style kernel generators on O-HAZE. We reevaluate the existing final checkpoints for the original generator and full model and train the replaced-generator variant from scratch. In this evaluation, their PSNR values are 26.7752, 26.2014, and 23.9343 dB for the full model, original generator, and replaced generator, respectively. Reevaluating the existing checkpoints changes PSNR by less than 0.001 dB. All Δ PSNR values in the main text use the reported full-model reference of 26.7755 dB.

Training repetitions and full-image evaluation on the GoPro dataset. Table 9 reports the three independent GoPro training runs. Pairing results by seed gives a PSNR gain of 0.5013 ± 0.0285 dB for NSI over SS2D, with a positive difference at every seed. Averaged over seeds, the difference is positive for 10 of 11 videos and is -0.0513 dB for the remaining video. The main results weight all 1,111 frames equally. Matching seed labels does not imply identical training trajectories, because the mini-batch random states are not strictly paired. Table 2 summarizes these runs using the reference forward implementation. Appendix B.6 reports paired timing of the optimized implementation. Earlier resource measurements remain available in the supplementary records under their original evaluation settings.

Kernel generation network. The replaced-generator control achieves lower PSNR than the original-generator control on four of five test images.

Per-step kernel regeneration. Table 4 reports the mean O-HAZE result. Regenerating kernels at every step yields lower PSNR than the full model on four of five images. When we exclude each image in turn, the mean PSNR difference over the remaining four images is always positive, ranging over $[0.0862, 0.5925]$ dB for the full model minus the control.

C.3 Spatial Paths and State Transmission

We examine two aspects of propagation: injecting input at every step and synchronously reading all eight neighbors. When input is injected only at the first step, independent training reduces O-HAZE PSNR from 26.7755 to 26.5319 dB but increases SSIM from 0.90655 to 0.91171. Repeated input injection therefore benefits PSNR, with a different effect on SSIM. This control only removes d_f after the first step; the controls below instead change when diagonal states contribute.

Scanning-path statistics. We measure how many recurrences connect two neighboring positions under a scan order. Let $p = (i, j)$ be a grid position, and let $\mathcal{N}_8(p)$ contain its eight neighbors, excluding the center. A scan order π assigns index $\pi(p)$ to each position, so the dependency between

Table 8: Results of component ablations and other controls, all using seed 42 and the prescribed protocol for each dataset. Panel (a) compares components across three datasets; panel (b) reports additional controls. Δ PSNR is the control minus the Full model on the same dataset, using the displayed four-decimal values except for the GoPro joint control, whose difference is computed before rounding. MACs cover the complete model with a $1 \times 3 \times 256 \times 256$ input (Appendix B.5). CSPN-style is an external baseline; Appendix C.2 explains its reevaluation.

(a) Cross-dataset component comparison

	O-HAZE		GoPro		Haze4K	
Configuration	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Full model	26.7755	0.90655	30.8918	0.9132	30.1147	0.982612
Identity retention	26.4429	0.90165	30.9025	0.9133	30.0479	0.982764
Final-state readout	26.3239	0.90463	30.9126	0.9132	29.3349	0.980952
MACs (G)	25.613		61.431		25.613	

(b) Specialized mechanism controls

Configuration	Dataset	PSNR↑	SSIM↑	Δ PSNR	MACs (G)
Five-layer feedforward stack	O-HAZE	26.4958	0.90390	-0.2797	25.613
Input-independent shared kernels	O-HAZE	26.0944	0.90394	-0.6811	24.631
Per-step kernel regeneration	O-HAZE	26.3328	0.90406	-0.4427	34.037
	GoPro	30.9480	0.9141	$+0.0562$	76.498
First-step input injection only	O-HAZE	26.5319	0.91171	-0.2436	25.613
Identity retention + final-state readout	O-HAZE	26.2586	0.90393	-0.5169	25.613
	GoPro	30.8772	0.912575	-0.0145	61.431
	Haze4K	29.9226	0.982132	-0.1921	25.613
Weighted state readout	Haze4K	29.8398	0.982006	-0.2749	25.613
Delayed diagonal states	GoPro	30.8970	0.9132	$+0.0052$	61.431
Without propagation gain control	O-HAZE	24.8043	0.89319	-1.9712	25.613
	GoPro	28.6148	0.8639	-2.2770	61.431
	Haze4K	26.3747	0.968842	-3.7400	25.613
CSPN-style: original generator	O-HAZE	26.2014	0.90397	-0.5741	24.304
	GoPro	30.2873	0.9028	-0.6045	59.091
	Haze4K	27.2493	0.973603	-2.8654	24.304
CSPN-style: replaced generator	O-HAZE	23.9343	0.78457	-2.8412	24.113
	Haze4K	27.6520	0.974879	-2.4627	24.113

Table 9: GoPro results for three independent training runs. Each entry reports PSNR/SSIM from evaluating every 720×1280 test image in FP32 with TF32 disabled.

Method	Random seed 42	Random seed 43	Random seed 44
Ours	30.8918/0.913199	30.9010/0.913451	30.8133/0.912229
SS2D	30.3735/0.904630	30.4326/0.905811	30.2960/0.903382
VIVNet native module	29.9925/0.897748	29.9491/0.897356	29.7654/0.893090
CSPN-style	30.2873/0.902812	30.1062/0.899408	30.2133/0.901037

Table 10: State propagation between neighboring positions on a 256×256 grid. SS2D and NSS follow the VMamba and MaIR scan orders; NSS uses strip width 4. Counts are computed from scan indices. For the union of two modules, each neighboring pair uses the shorter available path. Appendix C.3 defines the measurements.

Propagation method	One-step coverage		Mean propagation steps for diagonal neighbors
	eight neighboring positions	diagonal neighbors	
VMamba SS2D	50.10%	0%	256
MaIR NSS, unshifted	43.91%	0%	23.75
MaIR NSS, shifted	43.91%	0%	44.58
MaIR NSS, path union of two modules	50.10%	0%	2.496
NSI	100%	100%	1

p and q traverses $|\pi(q) - \pi(p)|$ recurrences. For a set of scanning branches that includes the reverse directions, define

$$d_{\text{scan}}(p, q) = \min_{r \in \mathcal{S}} |\pi_r(p) - \pi_r(q)|, \qquad C_1 = \frac{\#\{(p, q) : q \in \mathcal{N}_8(p),\ d_{\text{scan}}(p, q) = 1\}}{\#\{(p, q) : q \in \mathcal{N}_8(p)\}}. \tag{33}$$

Here, p, q are valid grid positions, $\pi_r(p)$ is position p's index in branch $r \in \mathcal{S}$, and d_{scan} is their shortest sequence distance. The set $\mathcal{N}_8(p)$ contains valid eight-connected neighbors; # counts ordered pairs, and C_1 is the fraction connected in one recurrence. In NSI, every update reads all eight neighbors from the previous state grid.

On an $n \times n$ grid, SS2D's row and column scans directly connect axial neighbors. The shortest sequence distances for the two diagonal orientations are $n - 1$ and $n + 1$; because their counts are equal, the mean is n. Thus, diagonal neighbors remain only $\sqrt{2}$ apart spatially, yet their mean propagation distance grows with the grid size. Table 10 compares these distances with NSS and NSI.

For NSS with strip width 4, neither the shifted nor unshifted scan directly connects diagonal neighbors. NSI's 3×3 neighborhood update connects every neighboring pair in one step. These counts measure dependency length, not execution time: they exclude parallelism, memory access, and operator implementation. Supplementary records provide complete counts and quantiles, and Appendix C.1 defines the controls that delay diagonal contributions.

Appendix C.3 complements these path counts with measurements from a trained model.

Scanning-state transmission after training. We measure state transmission in the first token mixer of an SS2D model trained on O-HAZE. For each neighboring pair and state component across the five test images, we select the causal branch that retains more of the state magnitude. Median retention is 96.45%–96.72% along one-step axial paths but only 0.105%–0.185% along diagonal paths spanning 384–448 sequence positions. Removing intermediate transitions while preserving the learned parameters and the target position's time step gives one-step retention of 98.53%–98.58%. The slowest-decaying tenth of state components retain approximately 54.8%–60.8% along the original paths. These measurements isolate transmission of the previous state; SS2D also includes local convolution, input injection, readout, and fusion across scan branches.

Timing of contributions from diagonal neighbors. Delaying diagonal states by one step reduces the number of reachable positions after five steps from 121 to 61. The independently trained control achieves 26.4166 dB PSNR and performs worse than the full model on four images. Excluding each image in turn gives a mean PSNR difference of $[0.1434, 0.4835]$ dB over the remaining four images. In a separate CPU diagnostic with parameters fixed, mean O-HAZE PSNR falls from 26.7752 to 26.4207 dB, with decreases on 4/5 images. Mean SSIM falls by 0.000421, with decreases on 3/5 images. The two experiments apply the same delay but measure performance after retraining and immediate sensitivity, respectively.

On GoPro, the half-delayed control averages the current and previous diagonal contributions from the second step onward while preserving 121 reachable positions. Its PSNR is 30.8970 dB, only 0.0052

Table 11: ERF radius r_{90} (pixels) and axis-to-diagonal ratio (A/D) at distances of 64–160 pixels, measured on the first test image of each dataset with seed-42 checkpoints. An isotropic ERF has A/D close to one. The lower rows are the MaIR and Spatial-Mamba variants of Appendix D.1.

	GoPro		O-HAZE		Haze4K	
Token mixer	r_{90}	A/D	r_{90}	A/D	r_{90}	A/D
VIVNet (native)	582.2	1.03	222.8	1.44	360.8	1.12
DWConv	–	–	111.0	1.43	124.8	1.61
Swin	–	–	121.4	1.49	110.5	1.02
CSPN-style	73.7	0.98	119.5	1.54	203.4	1.67
SS2D	480.1	24.21	216.0	8.82	319.0	2.81
Ours	91.4	1.36	121.7	1.45	159.7	1.97
MaIR core	–	–	224.0	11.56	–	–
MaIR full block	–	–	217.5	9.34	–	–
Spatial-Mamba core	–	–	243.3	3.48	–	–
Spatial-Mamba full block	–	–	265.3	2.31	–	–

dB above the full model's 30.8918 dB. Performance is therefore similar under this protocol. Neither the fully delayed nor half-delayed variant removes diagonal neighbors; Appendix C.1 specifies both updates.

C.4 Effective Receptive Fields of Trained Models

We measure the gradient-based effective receptive field (ERF) of each complete model trained with seed 42. For each dataset, we use the first image in the fixed test order (GoPro: 720×1280; O-HAZE: 512×422 after resizing; Haze4K: 480×640) with the evaluation preprocessing of Appendix B.4. The scalar target is the RGB mean of the output residual, the prediction minus the input, at the central pixel; subtracting the input removes the identity path of the image. The ERF map is $g(i,j) = \sum_c |\partial y / \partial x_c(i,j)|$ with respect to the unpadded input, and r_{90} is the smallest radius around the central pixel that contains 90% of the total gradient mass. To compare shapes, we divide the mean of g within four pixels of the horizontal and vertical lines through the central pixel by its mean within four pixels of the two diagonals, over distances of 64–160 pixels. This axis-to-diagonal ratio (A/D) is close to one for an isotropic ERF.

Table 11 and Figure 5 summarize the results. NSI's r_{90} is 91.4, 121.7, and 159.7 pixels on GoPro, O-HAZE, and Haze4K. These radii are smaller than those of SS2D (480.1, 216.0, and 319.0) and native VIVNet (582.2, 222.8, and 360.8) and comparable to those of DWConv, Swin, and CSPN-style (73.7–203.4 across datasets). Among the token mixers in Tables 2 and 3, the two with the largest r_{90} on each dataset, native VIVNet and SS2D, have lower PSNR than NSI. On O-HAZE, the four MaIR and Spatial-Mamba variants have r_{90} of 217.5–265.3 pixels but at most 23.2875 dB PSNR (Table 15). Because r_{90} is computed within the image, it is bounded by the image size: a spatially uniform map gives 613.3, 263.8, and 321.2 pixels on the three image sizes. The r_{90} of native VIVNet, SS2D on Haze4K, and the MaIR and Spatial-Mamba variants lie near or above these values, so they indicate sensitivity spread over most of the image and underestimate the untruncated radius. A larger receptive field therefore does not account for NSI's gains in these comparisons.

The ERFs also differ in shape. SS2D concentrates sensitivity along the horizontal and vertical lines through the output pixel: its A/D is 24.21, 8.82, and 2.81 on the three datasets, and the MaIR core module and full block reach 11.56 and 9.34 on O-HAZE. NSI's A/D is 1.36 and 1.45 on GoPro and O-HAZE, within the 0.98–1.67 range of native VIVNet, DWConv, Swin, and CSPN-style, and 1.97 on Haze4K, above that range but below SS2D's 2.81. SS2D's A/D exceeds NSI's on every dataset for all nine combinations of band half-width (2, 4, or 8 pixels) and distance range (32–128, 64–160, or 64–256 pixels). This contrast is consistent with the one-step coverage in Table 10: no single SS2D or NSS recurrence connects diagonal neighbors, whereas NSI reads all eight neighbors at every step. The correspondence is qualitative: the two-module MaIR path union shortens the mean diagonal path to 2.496 steps, yet the MaIR core module has the largest A/D on O-HAZE. Consistent with its unidirectional scan, Spatial-Mamba has a one-sided ERF: the gradient mass more than four rows

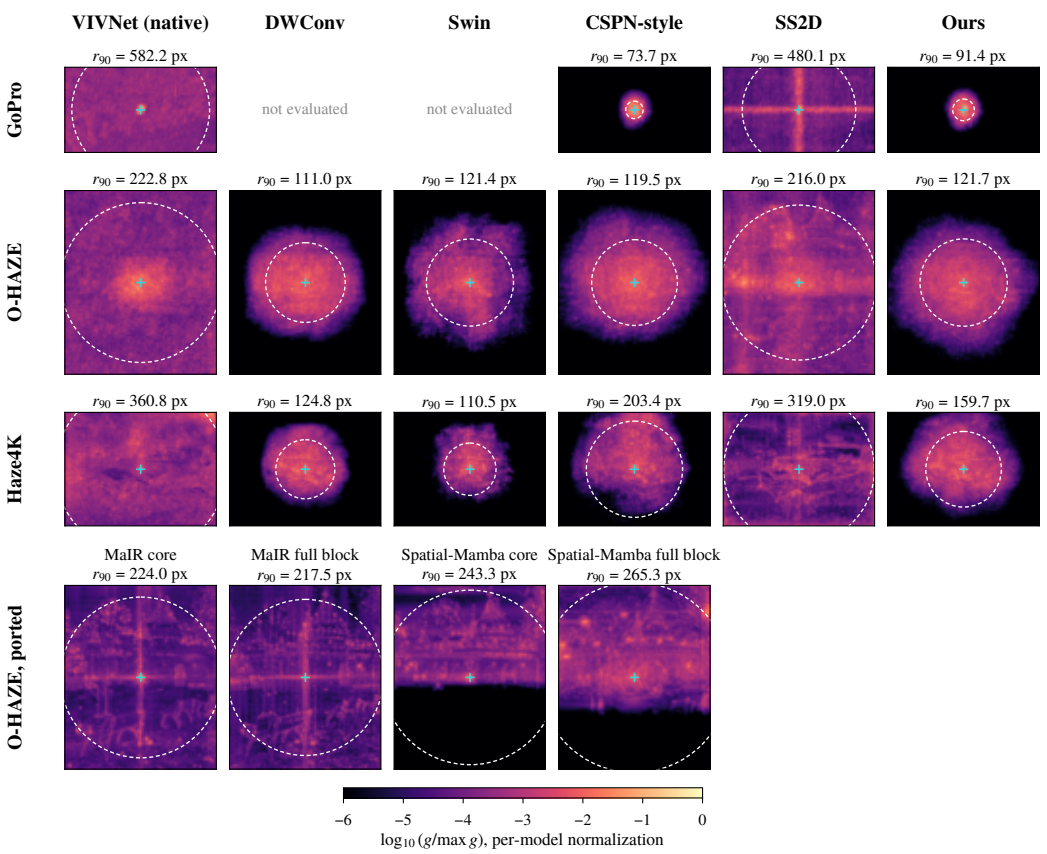


Figure 5: **Effective receptive fields of complete models.** Each map shows $\log_{10}(g/\max g)$ for the central output pixel of the first test image, normalized per model; the dashed circle has radius r_{90}. GoPro includes only the four token mixers in Table 2. The bottom row shows the MaIR and Spatial-Mamba variants on O-HAZE.

below the output pixel is 6.3% (core module) and 17.9% (full block) of that more than four rows above it, compared with 70.0%–148.3% for the other O-HAZE models.

These measurements use one image and one output position per dataset and the seed-42 checkpoints, so they describe individual cases rather than dataset averages. The gradients include the effect of the shared backbone and measure first-order sensitivity of the complete model, not the state-transfer path lengths in Table 10. Image structure also shapes the maps, and for compact ERFs the 64–160 pixel range covers the steep fall-off, where A/D varies across settings (0.91–2.47 for NSI on Haze4K); we therefore compare A/D only between models on the same image.

C.5 Fixed-weight Diagnostics of State-dependent Retention and Incremental Readout

These diagnostics test how the trained O-HAZE model depends on retention and readout without allowing it to adapt through retraining. We keep its parameters fixed and modify all 22 propagation modules. Spatial averaging replaces each group's retention map by its spatial mean at each step; cyclic translation shifts that map by half the grid size in both directions. A third intervention reuses the coefficients from the first nonzero state at every subsequent step. We also test identity retention, final-state readout, and their joint replacement. To control for changes in output magnitude, we rescale each module's output after projection and before residual addition by the ratio of reference RMS to control RMS. Both outputs use the same current module input, and RMS includes every channel and position of that sample. Table 8 separately evaluates the corresponding structures after independent training.

Table 12: Mean PSNR changes under interventions with model parameters fixed. Values are the full model minus the control in dB, so negative values favor the control. The final column counts test images whose PSNR decreases after output magnitudes are matched.

Control	Unmatched magnitude	After magnitude matching	Number of decreased images
Fixing retention coefficients in subsequent steps	1.1758	0.0634	4/5
Spatially averaged retention coefficients	0.2416	0.2239	5/5
Cyclically translated retention coefficient maps	0.3983	0.4023	4/5
Identity retention	1.7549	0.1803	5/5
Final-state readout	4.9884	1.1861	4/5
Identity retention + final-state readout	5.7265	1.1508	4/5

Matching output magnitudes does not remove the effect of spatial averaging: both PSNR and SSIM still decrease on all five images. Final-state readout and the joint replacement perform better than the full model on image 41, showing that the effect is not uniform across images. Table 8 reports the separate results after retraining.

C.6 Propagation Gain and Stepwise State Observations

Propagation gain control. The O-HAZE control uses the final checkpoint of its prescribed training run; the GoPro reference is the seed-42 full model from the main three-seed experiment. Removing propagation gain control lowers O-HAZE PSNR at every saved evaluation point; Table 5 reports the final result. On GoPro, both metrics decrease for all 11 videos, and PSNR decreases on 1,106 of 1,111 frames. The Haze4K control is trained independently for 100,000 steps and achieves PSNR/SSIM of 26.3747/0.968842 over all 1,000 test images, with PSNR 3.7400 dB below the full model. The supplementary records provide per-image and per-video results and complete training curves.

Checking the theoretical norm bounds on a trained model. To compare measured norms with the theoretical bounds, we use the full Haze4K model trained for 100,000 steps with seed 42. We evaluate the first five images in the fixed test order and record every feature position in all 22 propagation modules. Images are processed at their original resolution without cropping. Their dimensions are already divisible by 4, so no reflection padding is added. For each image and module, $N(\cdot)$ takes the maximum over samples, channel pairs, and positions, as defined in Appendix A.1. We compute q_f by summing the spectral norms of the nine 2×2 transformations at each receiving position and taking the maximum over positions and channel pairs.

Let $B_{h,k} = q_f(1 - q_f^k)N(d_f)/(1 - q_f)$, $B_{\Delta,k} = q_f^k N(d_f)$, and $B_z = q_f(1 - q_f^K)N(d_f)/(1 - q_f)$. These quantities bound the state norm at step k, its increment norm, and the readout norm after K steps, respectively; d_f is the repeated input term. Table 13 summarizes 110 image–module combinations. The actual gain q_f ranges from 0.7832 to 0.9457 and remains below 0.95. Over five steps, state norms, increment norms, and readout norms before output scaling and projection remain within their bounds. We compute each ratio using that record's q_f and $N(d_f)$ and report the maximum across records.

Learnable kernel scaling. We also test whether learning the overall propagation scale improves restoration. Each module retains kernel normalization and adds a positive learnable scale initialized to one, without a fixed upper bound (Appendix C.1). On Haze4K, this variant improves PSNR by 0.1879 dB over the full model (Table 14), and every learned scale is below one at the final checkpoint. This observation concerns one trained configuration. Unlike the construction analyzed in the propositions, the variant does not enforce a fixed upper bound on propagation gain during training.

Stepwise state observations at the training endpoint. Figure 6 uses the full model's final O-HAZE checkpoint and covers five test images, all 22 modules, and five propagation steps. For

Table 13: Verification of norm bounds over five propagation steps in the trained Haze4K model, covering five images and all 22 modules. For q_f, a violation means $q_f < 0.95$ fails; for the other quantities, it means the measured norm exceeds its finite-step bound.

Checked quantity	Number of records	Measured range or maximum bound ratio	Number of violations
Actual kernel gain q_f	110	$[0.7832,\ 0.9457]$	0
State $N(h_k)/B_{h,k}$	550	0.8614	0
Net change $N(\Delta h_k)/B_{\Delta,k}$	550	0.8614	0
Readout $N(z)/B_z$	110	0.4687	0

Table 14: Effect of learnable kernel scaling on Haze4K. Both models use seed 42, train for 100,000 steps, and process complete test images at their original resolution. Δ PSNR is the control minus the Full model, computed from the displayed four-decimal values.

Configuration	PSNR (dB)↑	SSIM↑	Δ PSNR	MACs (G)
Full model	30.1147	0.982612	0.0000	25.613
Learnable kernel scaling	**30.3026**	**0.983200**	+0.1879	25.613

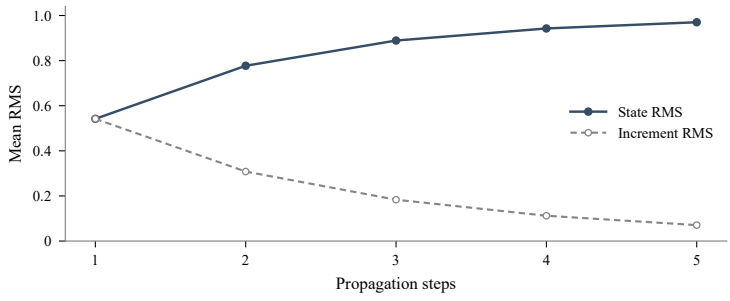


Figure 6: Evolution of state and increment magnitudes in the trained O-HAZE model. The horizontal axis shows propagation steps within a module. Curves average RMS over image–module pairs with equal weights, excluding padded regions.

Table 15: Results for replacing either the mixing operator or the complete block. Final checkpoints are evaluated on complete images resized to a long side of 512 pixels. Parameter counts and MACs cover the complete model; MACs use a 256^2 input.

Method	Ported unit	PSNR (dB)↑	SSIM↑	Params (M)	MACs (G)
MaIR	Core module	22.8164	0.764536	3.547	45.445
	Full block	22.9354	0.739838	1.932	25.545
Spatial-Mamba	Core module	22.7318	0.788609	2.946	34.968
	Full block	23.2875	0.804586	2.734	32.510

each image and module, we compute the RMS of channel-pair magnitudes after excluding reflection padding, then average equally over 110 combinations. State RMS rises from 0.54195 at step one to 0.97028 at step five, while increment RMS falls from 0.54195 to 0.07065. Thus, states accumulate information while successive changes decrease. Table 13 separately measures the maximum channel-pair norm used in Proposition 1.

D Comparisons with Related Models and Analysis of Input Resolution

D.1 Incorporating MaIR and Spatial-Mamba into VIVNet

We compare MaIR and Spatial-Mamba within the same VIVNet backbone on O-HAZE, preserving its input and output interfaces, multiscale structure, stage widths, and 22 module locations. Core-module porting replaces only the mixing operator and retains VIVNet's normalization, feedforward branch, and residual connections. Full-block porting instead inserts the other method's complete official block at those locations. Each configuration is trained once with seed 42 for 55,800 steps under the O-HAZE protocol (Appendix B). These comparisons answer different questions about operators and blocks within VIVNet; neither reproduces the other method's original network.

For MaIR, core-module porting uses the official VMM, NSS, and SSA with expansion ratio 2, stripe width 4, alternating shifts of 0/2, and stagewise state widths $(4, 8, 16, 16, 8, 4, 4)$. Full-block porting uses the dehazing VSSBlock, including normalization, residual scaling, and the feedforward branch, with state expansion ratio 1.2, feedforward expansion ratio 2, stripe width 8, and shifts of 0/4. For Spatial-Mamba, the core uses the official StructureAwareSSM with state width 1, expansion ratio 2, a 3×3 convolution, a unidirectional scan, and state fusion at dilation rates 1/3/5, merged into an 11×11 depthwise convolution at inference. Its full block also includes two positional convolutions, normalization, a feedforward branch with expansion ratio 4, and the official Linear/LayerNorm initialization.

Using complete blocks instead of core operators raises PSNR by 0.1190 dB for MaIR and 0.5557 dB for Spatial-Mamba, although MaIR's SSIM decreases. These changes jointly affect feedforward computation, normalization, residual connections, and some widths; Spatial-Mamba also changes initialization. The improvement therefore cannot be attributed to the spatial operator alone. The Spatial-Mamba core variant was trained with cuDNN TF32 enabled and matrix-multiplication TF32 disabled; all evaluations disable TF32. NSI and SS2D reuse their existing final checkpoints. Scan indices, backend computation, and transferred weights were verified, with details in the supplementary records. The resolution study below uses the complete MaIR and Spatial-Mamba blocks.

D.2 Input Resolution and Restoration Performance

Keeping model parameters fixed, we resize five test images and their targets to eight resolutions while preserving aspect ratios. Each input contains the complete scene. We use bicubic interpolation with antialiasing and reflection padding on the right and bottom to multiples of 32. After inference, we remove padding, clip predictions to $[0, 1]$, and average RGB PSNR and SSIM equally over images. All models use identical targets at each resolution. The curves reflect the model's response, changes in target detail caused by interpolation, and differences in object size between training crops and test images. They measure sensitivity to input resolution for these trained models.

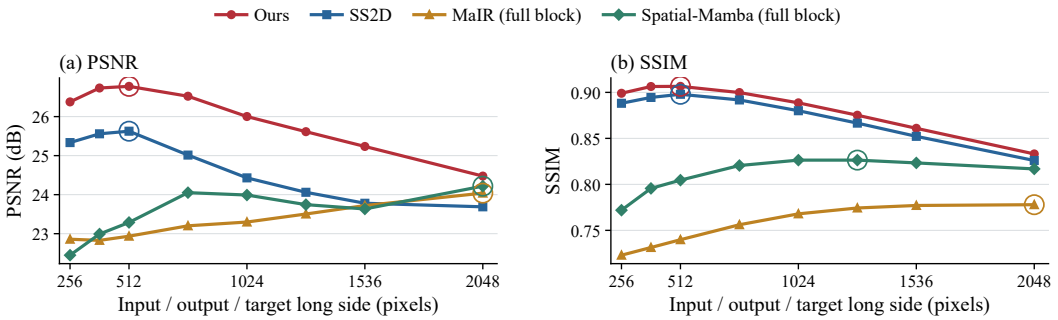


Figure 7: Restoration performance as input resolution changes. Circles mark the highest mean PSNR or SSIM for each model.

Table 16: Mean PSNR (dB) and SSIM over five images at eight input resolutions. Bold values mark each model's highest mean for each metric.

Long side (pixels)	Our method		SS2D		MaIR full block		Spatial-Mamba full block	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
256	26.3769	0.899033	25.3345	0.888156	22.8577	0.722997	22.4486	0.771924
384	26.7329	0.906378	25.5592	0.894503	22.8285	0.731358	22.9910	0.795609
512	**26.7752**	**0.906550**	**25.6259**	**0.897918**	22.9354	0.739838	23.2875	0.804586
768	26.5222	0.899813	25.0141	0.891769	23.2016	0.756108	24.0514	0.820477
1024	26.0022	0.888617	24.4301	0.880055	23.2983	0.767947	23.9902	0.826348
1280	25.6141	0.875168	24.0606	0.866570	23.5040	0.774318	23.7443	**0.826363**
1536	25.2351	0.860944	23.7798	0.852209	23.7205	0.777056	23.6334	0.823314
2048	24.4759	0.833081	23.6882	0.825948	**24.0390**	**0.777899**	**24.2180**	0.816783

Increasing the long side from 512 to 2048 pixels raises mean PSNR by 1.1035 dB for MaIR and 0.9305 dB for Spatial-Mamba, with improvements on 4/5 and 3/5 images, respectively. NSI and SS2D instead lose 2.2993 and 1.9376 dB, with decreases on all five images. NSI nevertheless has the highest PSNR and SSIM among the four models at 2048. Both metrics peak at 512 for NSI and SS2D. Spatial-Mamba's PSNR decreases between 768 and 1536, while its SSIM peaks at 1280, so its response is not monotonic.

Training crops and detail size. The difference between training and evaluation scales offers a possible explanation for these results. All four models train on random 256^2 crops from original images without random spatial scaling. For an image with a long side of 4000 pixels, a structure 100 pixels wide remains that size within a training crop but becomes approximately 13 or 51 pixels wide when the complete image is resized to 512 or 2048, respectively. Higher evaluation resolutions therefore bring local structures closer to their training sizes. This may partly explain the improvement of MaIR and Spatial-Mamba at higher resolutions.

How input resolution affects each module. Changing the input resolution changes how much of the original scene is covered by a fixed neighborhood. NSI generates kernels from $3 \times 3/7 \times 7$ local features and performs five 3×3 propagation steps, so smaller inputs allow this neighborhood to cover a larger part of the scene. SS2D's recurrence length varies with the grid size. MaIR uses NSS stripes of width 8 and SSA to combine four scan directions using global feature statistics; Spatial-Mamba fuses states at dilation rates 1/3/5. These architectural differences and learned parameters affect sensitivity to resolution.

Changes in overall color cast. We decompose RGB error into the spatial mean for each channel and a residual after subtracting that mean; their mean squared values sum to MSE. As the long side increases from 512 to 2048, 33.65% of NSI's additional MSE comes from the mean color error and 66.35% from the spatial residual. For SS2D, the shares are 63.73% and 36.27%, and both

terms increase. Changes in the size of local structures and the area covered by spatial aggregation offer possible explanations. These measurements describe the response of trained networks to input resolution; they do not establish stability or restoration quality independently of image size.