Flow Equivariant State Space Model

Chengrong Ye[2], **Xinyan Gong**[2,3], **Lingwei Zhang**[2,3], **Yue Song**[1*]
[1]Tsinghua University, [2]Shanghai Jiao Tong University, [3]University of Michigan

Abstract

Selective state space models such as Mamba process such data efficiently, but their recurrent state is updated at fixed spatial coordinates. As a result, evidence from a moving object can be accumulated across inconsistent locations rather than in the object's moving reference frame. To address this problem, we introduce Flow-SSM, a flow-equivariant selective state space model that lifts recurrent memory to candidate motion branches and transports each branch along its corresponding flow before applying the selective update. To handle real-world sequences where broad motion and local deformation often occur at different scales, we further introduce Hierarchical Flow-SSM, which organizes flow-aligned memory in a coarse-to-fine hierarchy rather than a single flat representation. We validate our architecture progressively: demonstrating exact transformation capture on Moving and Rotating MNIST, interpretable flow structures on KTH action videos, improved prediction on UCF-101 natural videos with unconstrained motion, and the necessity of our multi-scale design for complex forecasting on the SEVIR weather dataset. Beyond merely improving predictive accuracy, our analyses confirm that the explicit flow alignment and hierarchical communication contribute measurably to the observed gains. These results support transported recurrence as an effective inductive bias for spatiotemporal sequence modeling. Code of this paper is available at `https://github.com/ycr-sjtu05/Flow-Equivariant-State-Space-Model`.

1 Introduction

Natural sequences are not just ordered collections of independent snapshots. A moving object in a video, a rotating digit, or a drifting storm system is often better described as an underlying signal transported over time [1, 2, 3]. This suggests a foundational principle for modeling dynamic sequential data: recurrent memory should be accumulated along the motion trajectories of the underlying signal, rather than at static coordinates.

This perspective reveals a concrete limitation of modern state space models such as Mamba [4], which builds on a line of efficient SSM sequence models [5, 6, 7, 8]. Consider a simple translated sequence $x_t(p) = x_0(p - vt)$. A standard spatial SSM recurrence stores hidden memory $h_t(p)$ at fixed spatial coordinates, so the same coordinate observes different parts of the moving object over time. Consequently, temporal evidence for one object may be accumulated across inconsistent spatial locations, producing disconnected traces in memory. Our Flow-SSM addresses this mismatch by lifting Mamba's state to $h_t(\nu, p)$ over candidate motion trajectories ν, then transporting each memory slice along its trajectory before the selective update. Crucially, accumulating memory along trajectories reveals a deep connection to symmetry: when an input signal undergoes a global motion shift, its accumulated memory traces are not distorted across fixed coordinates, but simply shift intact across trajectory branches. Thus, trajectory-aligned recurrence naturally achieves flow equivariance [9, 10, 11, 12, 13, 14] while preserving SSM's input-dependent dynamics.

However, a flat lifted representation over all positions and all candidate velocities is unnecessarily rigid for real-world dynamics, since real-world motion often exhibits scale-dependent behavior, with

*Denotes corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

distinct dynamical characteristics at the macro and micro levels We therefore introduce Hierarchical Flow-SSM as a coarse-to-fine version of Flow-SSM. The model maintains states across matched levels of velocity and spatial resolution, pairing coarser spatial grids with broader velocity bins so that coarse levels track global motion patterns with fewer flow hypotheses while fine levels resolve local residual motion. Cross-scale messages are restricted to equivariant communicators [15], so information can move across levels without breaking the flow structure. This hierarchical construction connects to broader ideas in geometric deep learning and equivariant hierarchy [10, 16].

We evaluate the proposed models across increasingly realistic settings. On Moving and Rotating MNIST [1, 9], Flow-SSM reduces forecasting error under controlled translation and rotation flows. On KTH action recognition [17, 9] with induced camera motion, it improves classification accuracy and its velocity branches show stronger alignment with the ground truth motion. On UCF-101 natural-video prediction [18], Flow-SSM outperforms Mamba and FERNN [9], with larger gains on high-motion sequences where motion is not constrained to predefined velocity branches. On SEVIR weather forecasting [19], Hierarchical Flow-SSM achieves best forecasting scores among the evaluated baselines, and probing analyses further show that cross-scale messages improve the prediction. Overall, Flow-SSM replaces order-only recurrence with transported recurrence over candidate motion branches, providing a selective state space model whose memory update explicitly follows the modeled flow.

2 Method

2.1 Preliminaries

2.1.1 Standard State Space Model

A simplified discrete selective state space update, following mamba [4, 8], has the form

$$h_{t+1} = \bar{A}_t h_t + \bar{B}_t x_t, \qquad y_t = C_t h_t + D_t x_t, \tag{1}$$

where the discretized parameters $\bar{A}_t, \bar{B}_t, C_t, D_t$ are generated from the input x_t. Yet it does not imply geometric consistency when processing flowing sequential data: The recurrence in Eq. (1) accumulates memory at fixed spatial positions along temporal order, rather than following the spatial trajectory of a moving signal.

2.1.2 Flow Equivariance Formalized via Lie Algebra

To formalize trajectory-aligned memory accumulation across general motions (*e.g.,* translations and rotations), we use Lie groups and Lie algebras as a formal mathematical language to parameterize continuous trajectories [9].

Let a Lie group G act on a spatial domain X as an action of group [11, 15, 10] and $x_t \in \mathcal{F}(X; \mathbb{R}^C)$ denote a frame at time t, where $x_t(p) \in \mathbb{R}^C$ is the signal value at spatial position $p \in X$. For a group element $g \in G$, the induced left action on a signal is defined by pulling back spatial coordinates:

$$(g \cdot x_t)(p) := x_t(g^{-1} \cdot p), \qquad p \in X.$$

Intuitively, as the signal moves under g, the value arriving at position p is read from its upstream location $g^{-1} \cdot p$ prior to the motion.

Let $\mathfrak{g}$ be the commutative Lie algebra of G, and let $V \subseteq \mathfrak{g}$ denote the set of commuting generators, such as velocities. For a generator $\nu \in V$, we write

$$\phi_t(\nu) = \exp(t\nu) \in G$$

as the cumulative transformation after time t. Since V is commutative, the flow satisfies:

$$\phi_s(\nu) \cdot \phi_t(\nu) = \phi_{s+t}(\nu), \qquad \phi_t(\nu_1) \cdot \phi_t(\nu_2) = \phi_t(\nu_1 + \nu_2).$$

The action of this flow on an entire sequence $x_{0:T-1}$ is defined frame-wise as

$$(\phi(\nu) \cdot x)_t(p) := x_t\big(\phi_t(\nu)^{-1} \cdot p\big)\,.$$

Here, the Lie algebra generator $\nu \in V$ formalizes a constant-velocity trajectory $\phi_t(\nu) = \exp(t\nu)$. A model that accumulates memory along trajectories should respond consistently when the entire sequence is transported along ν, a property formalized as Flow Equivariance:

Definition 1 (Flow Equivariance [9])**.** *A sequence model Φ is* flow equivariant *with respect to a set of generators V if, for any $\nu \in V$, applying the corresponding flow to the input sequence commutes with applying the model:*

$$\Phi[\phi(\nu) \cdot x] = \phi(\nu) \cdot \Phi[x], \qquad \forall \nu \in V.$$

2.2 Flow Equivariant State Space Model

2.2.1 Flow-SSM Components

To accumulate Mamba's memory along signal trajectories, we modify its static recurrence into a trajectory-aligned scan across moving reference frames (see Figure 1). Each reference frame is indexed by a candidate trajectory generator ν, whose recurrent memory is transported along $\phi_1(\nu)$ before receiving the selective update. To ensure that spatial features and selective gates themselves respect spatial geometry, we build them from standard G-equivariant maps [11, 15].

Definition 2 (G-Equivariant Map)**.** *Let G act on the input domain X and output domain Y. The induced actions on input fields $x \in \mathcal{F}(X; \mathbb{R}^C)$ and output fields $f \in \mathcal{F}(Y; \mathbb{R}^D)$ are defined by*

$$(g_0 \cdot x)(p) := x(g_0^{-1} \cdot p), \qquad (g_0 \cdot f)(q) := f(g_0^{-1} \cdot q),$$

where $p \in X$, $q \in Y$, and $g_0 \in G$. A map

$$\Gamma : \mathcal{F}(X; \mathbb{R}^C) \to \mathcal{F}(Y; \mathbb{R}^D)$$

is G-equivariant *if*

$$\Gamma(g_0 \cdot x) = g_0 \cdot \Gamma(x), \qquad \forall g_0 \in G.$$

Equivalently, in pointwise form,

$$\Gamma(g_0 \cdot x)(q) = \Gamma(x)(g_0^{-1} \cdot q), \qquad \forall g_0 \in G,\ q \in Y.$$

Now we build our model part by part.

Input lifting. We first lift each input frame from the spatial domain to the product space $V \times G$, analogous to lifting operations in group-equivariant networks [11, 12]. This is done by a simple V-shared lift:

$$u_t(\nu, g) := \Gamma_{\text{in}}(x_t)(g),$$

where Γ_{in} is a G-equivariant network. This lift injects the same local evidence into every reference frame. A more expressive flow-aligned temporal lift is discussed in Appendix A.1.

Lifted state space. After lifting the input, Flow-SSM maintains recurrent memory on the same product space of flow generators and spatial locations. Instead of using a single order-indexed hidden state, we define a dual order-indexed hidden state as

$$s_t \in \mathcal{F}(V \times G; \mathbb{R}^N), \qquad s_t(\nu, g) \in \mathbb{R}^N.$$

Here $\nu \in V$ indexes a candidate moving reference frame, while $g \in G$ indexes spatial position. Intuitively, each ν-slice stores the memory of the sequence as seen from one possible motion hypothesis. To make the lifted state compatible with the flow action, we use the representation

$$(\phi(\hat{\nu}) \cdot s)_t(\nu, g) := s_t(\nu - \hat{\nu}, \phi_t(\hat{\nu})^{-1} g).$$

The generator-axis shift reflects the change from absolute to relative motion under the transformed input flow. Applying a flow $\hat{\nu}$ shifts the state both in space and along the generator axis.

Selective parameter fields. We keep Mamba-style input-dependent dynamics, but require them to respect the group action. Let a G-equivariant parameter generator produce the selective fields

$$(\Delta_t, B_t, C_t, D_t) = \Gamma(x_t),$$

where $\Delta_t(g), B_t(g), C_t(g), D_t(g)$ are evaluated pointwise on G. The discretized SSM parameters are obtained pointwise:

$$\bar{A}_t(g) = F_A(\Delta_t(g), A), \qquad \bar{B}_t(g) = F_B(\Delta_t(g), A, B_t(g)).$$

The selective fields depend only on spatial position and are shared across flow branches. Detailed parameterizations and discretization choices are given in Appendix A.1.

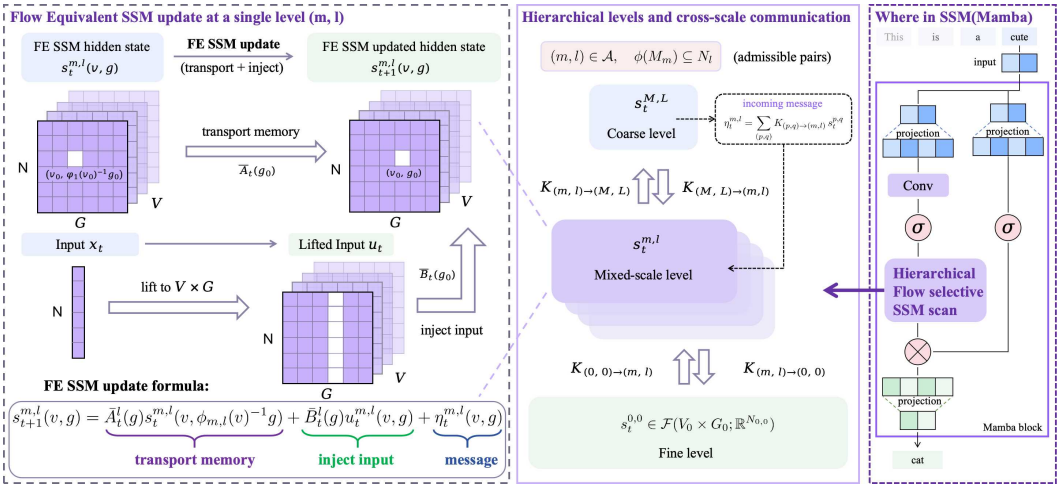

Figure 1: Overview of Hierarchical Flow-SSM. **Left:** The input x_t is first lifted to the lifted space $V \times G$, producing the lifted input u_t. The recurrent hidden state $s_t^{m,\ell}(v, g)$ is then transported along the one-step quotient flow, so each branch reads memory from $(v, \phi_{m,\ell}(v)^{-1}g)$. Finally, the selective SSM update combines transported memory, injected lifted input and cross-scale message $\eta_t^{m,\ell}(v, g)$. **Middle:** hierarchical organization of lifted states across admissible pairs $(m, \ell) \in \mathcal{A}$. Coarse levels capture global transport, while fine levels refine local details at higher spatial resolution. Cross-scale communicators $K_{(p,q)\to(m,\ell)}$ exchange information between levels while preserving flow equivariance. **Right:** integration into a standard SSM (Mamba) block, where the proposed hierarchical flow selective SSM scan replaces the conventional order-only selective SSM layer.

2.2.2 Flow-SSM Scan and Equivariance Guarantee

Combining the lifted input, the lifted state, the transport operator and the shared selective fields gives the Flow-SSM scan:

$$s_{t+1}(\nu, g) = \bar{A}_t(g) s_t(\nu, \phi_1(\nu)^{-1} g) + \bar{B}_t(g) u_t(\nu, g).$$

The first term is the transported recurrent memory: the state at (ν, g) reads from the location that would have moved to g after one step under generator ν. The second term injects the current lifted input using the same Mamba-style selective dynamics as in standard SSMs.

The readout is computed pointwise in the lifted space:

$$y_t(\nu, g) = C_t(g) s_t(\nu, g) + D_t(g) u_t(\nu, g).$$

When an invariant representation is required, lifted output can be pooled over the generator axis V.

Theorem. Under G-equivariant parameter generators and pointwise SSM discretization, the Flow-SSM scan is flow equivariant. In particular, for any $\hat{\nu} \in V$,

$$s_t[\phi(\hat{\nu}) \cdot x](\nu, g) = s_t[x](\nu - \hat{\nu}, \phi_t(\hat{\nu})^{-1} g).$$

Here, accumulating memory along trajectories $\phi_1(\nu)^{-1} g$ directly yields flow equivariance: in a comoving frame following trajectory ν, spatial transport cancels out the motion, so transforming the input sequence by flow $\hat{\nu}$ simply shifts the accumulated memory intact to the relative trajectory branch $\nu - \hat{\nu}$. The full proof is provided in Appendix A.2.

2.3 Hierarchical Flow-SSM

Flat Flow-SSM aligns recurrent memory with motion over a single product space $V \times G$, representing all motion branches at one fixed spatial resolution. However, real-world motion often exhibits scale-dependent behavior, with distinct dynamical characteristics at the macroscopic and microscopic levels. Modeling such sequences on a single flat resolution cannot simultaneously capture global trajectory structure and fine-grained local evolution. We therefore introduce Hierarchical Flow-SSM as a coarse-to-fine factorization of Flow-SSM, illustrated in Figure 1: the model replaces one flat lifted state with multiple states defined on different motion and spatial resolutions.

2.3.1 Quotient Hierarchies and Admissible Scale Pairs

In multi-scale tracking, coarse spatial grids only need coarse trajectory hypotheses, as fine velocity differences cannot be resolved at low spatial resolution. Group quotients [15, 11] provide a rigorous tool to formalize this resolution matching by quotienting out fine-scale variations. Let $V^\star$ be the ambient additive generator space and $G^\star$ the ambient spatial transformation group. We choose subgroup chains

$$\{0\} = M_0 \subset M_1 \subset \cdots \subset M_M \subset V^\star, \qquad \{e\} = N_0 \triangleleft N_1 \triangleleft \cdots \triangleleft N_L \triangleleft G^\star,$$

and define quotient levels

$$V_m := V^\star/M_m, \quad \pi_m^V : V^\star \to V_m, \qquad G_l := G^\star/N_l, \quad \pi_l^G : G^\star \to G_l.$$

Larger m or l corresponds to coarser generator or spatial resolution.

Not every generator level V_m is compatible with every spatial level G_l. Let $\phi : V^\star \to G^\star$ be the one-step ambient flow map. We call (m, l) *admissible* if

$$\phi(M_m) \subseteq N_l.$$

Intuitively, any fine velocity differences merged into the same coarse velocity bin at level m cause a one-step spatial drift smaller than a single grid cell at level l, ensuring that all trajectories within the same bin land in the same spatial location. Such flow descends to a well-defined quotient map

$$\phi_{m,l} : V_m \to G_l, \qquad \phi_{m,l}(\pi_m^V(v)) := \pi_l^G(\phi(v)).$$

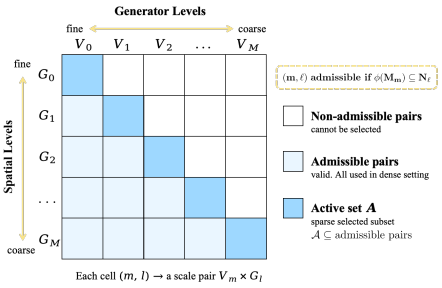

Figure 2: Admissible scale pairs.

As a simple example, consider 2D translations on a 32×32 periodic grid. Let G_0, G_1, G_2 be spatial grids of size 32×32, 16×16, and 8×8, respectively. Let V_0, V_1, V_2 be velocity levels at matching resolutions: V_0 keeps pixel-level velocities, V_1 groups velocities into 2-pixel bins, and V_2 groups them into 4-pixel bins. A pair (V_m, G_l) is admissible if the velocity differences discarded by V_m cannot be resolved at spatial level G_l. In this example, admissibility is simply $m \leq l$ in Figure 2. Thus, coarse velocity bins can only be used on sufficiently coarse spatial grids. We choose the active set $\mathcal{A}$ as a subset of all admissible pairs. A dense $\mathcal{A}$ approaches the full admissible hierarchy, while in practice we often use a sparse near-diagonal choice.

2.3.2 Levelwise Flow-SSM Updates

For each pair $(m, l) \in \mathcal{A}$, Hierarchical Flow-SSM maintains a lifted state on its own quotient generator and spatial spaces:

$$s_t^{m,l} \in \mathcal{F}(V_m \times G_l; \mathbb{R}^{N_{m,l}}).$$

Since admissibility gives a well-defined quotient flow $\phi_{m,l} : V_m \to G_l$, the Flow-SSM update from Section 2.2.2 can be applied directly and independently at each level:

$$s_{t+1}^{m,l}(v, g) = \bar{A}_t^l(g) s_t^{m,l}(v, \phi_{m,l}(v)^{-1} g) + \bar{B}_t^l(g) u_t^{m,l}(v, g).$$

Here the selective fields are generated from the spatial feature $x_t^{(l)}$ at level G_l, and are shared across V_m. Coarse levels capture global transport with fewer states, while fine levels refine local details at higher resolution.

2.3.3 Equivariant Cross-Scale Communication

Once multiple levels are introduced, states could communicate across scales. However, passing messages across scales cannot be an arbitrary mapping: it must keep moving reference frames aligned between levels, routing memory from a source trajectory and spatial cell only to compatible trajectories and locations at the target scale. Formally, for a cross-scale communication $\alpha \to \beta$ from $\alpha = (p, q)$ to $\beta = (m, l)$, let

$$K_{\alpha\to\beta} : \mathcal{F}(V_p \times G_q; \mathbb{R}^{N_\alpha}) \to \mathcal{F}(V_m \times G_l; \mathbb{R}^{N_\beta})$$

Algorithm 1 Hierarchical Flow-SSM

Require: sequence $\{x_t\}_{t=0}^{T-1}$ on $G^\star$, active set $\mathcal{A}$, edge set $\mathcal{E}$, quotient flows $\phi_{m,l}$

initialize $s_0^{m,l}(v,g) \leftarrow 0$ for all $(m,l) \in \mathcal{A}$
for $t = 0, \dots, T-1$ **do**
  for each spatial level l **do**
    compute G_l-equivariant fields $(\Delta_t^l, B_t^l, C_t^l, D_t^l)$
    discretize pointwise to obtain $\bar{A}_t^l, \bar{B}_t^l$
  end for
  for each active pair $(m,l) \in \mathcal{A}$ **do**
    form lifted input $u_t^{m,l}(v,g)$
    compute cross-scale message

$$\eta_t^{m,l} = \sum_{(p,q):(p,q)\to(m,l)\in\mathcal{E}} K_{(p,q)\to(m,l)} s_t^{p,q}$$

    update transported recurrent state

$$s_{t+1}^{m,l}(v,g) = \bar{A}_t^l(g) s_t^{m,l}\big(v, \phi_{m,l}(v)^{-1} g\big) + \bar{B}_t^l(g) u_t^{m,l}(v,g) + \eta_t^{m,l}(v,g)$$

    compute levelwise output $y_t^{m,l}$
  end for
  fuse levelwise outputs to obtain y_t^{multi}
end for
return $\{y_t^{\mathrm{multi}}\}_{t=0}^{T-1}$

be the communicator. To preserve trajectory alignment under motion shifts, we require $K_{\alpha\to\beta}$ to act as an intertwiner [15]:

$$K_{\alpha\to\beta} T_\alpha^t(\hat{v}) = T_\beta^t(\hat{v}) K_{\alpha\to\beta},$$

where $T_\alpha^t(\hat{v})$ and $T_\beta^t(\hat{v})$ are the induced flow actions on the source and target levels. In words, communicating after applying a flow should give the same result as applying the corresponding flow after communication. The total message received by level (m,l) is

$$\eta_t^{m,l} = \sum_{\alpha:\alpha\to(m,l)} K_{\alpha\to(m,l)} s_t^\alpha.$$

We give the exact quotient-space definition of T_α^t, the quotient broadcast formula, and the formal proof in Appendix A.4.

2.3.4 Hierarchical Flow-SSM

We now put the quotient hierarchy, levelwise Flow-SSM updates, and equivariant cross-scale messages together. Each active level updates its transported recurrent state, receives messages from connected levels, computes a levelwise readout, and contributes to the final multi-scale output. The full procedure is summarized in Algorithm 1.

Theorem. Suppose all active scale pairs are admissible and all cross-scale communicators are intertwiners. Then the Hierarchical Flow-SSM is flow equivariant. In particular, for each active level (m,l),

$$s_t^{m,l}[\phi(\hat{v}) \cdot x](v,g) = s_t^{m,l}[x]\left(v - \pi_m^V(\hat{v}), \phi_{m,l,t}(\pi_m^V(\hat{v}))^{-1} g\right).$$

In other words, an input motion $\hat{v}$ induces a synchronized shift across the entire hierarchy, where each level (m,l) tracks the trajectory at its own velocity and spatial granularity via $\pi_m^V(\hat{v})$ and $\phi_{m,l,t}(\pi_m^V(\hat{v}))$. The proof is given in Appendix A.3 and Appendix A.4.

3 Experiment

3.1 Controlled validation on Moving and Rotating MNIST

Task overview. We use Moving and Rotating MNIST as controlled prediction benchmarks [1, 9]. In both datasets, each sequence is constructed by compositing two MNIST digits on a shared canvas

Table 1: MNIST prediction results. Lower test MSE is better.

(a) Moving MNIST

Experiment	Test MSE ↓
FERNN, $10 \to 10$	1.1×10^{-4}
FERNN, $10 \to 50$	2.7×10^{-4}
Flow-SSM, $10 \to 10$	6.3×10^{-5}
Flow-SSM, $10 \to 50$	1.5×10^{-4}

(b) Rotating MNIST

Experiment	Test MSE ↓
FERNN, $10 \to 10$	6.1×10^{-4}
FERNN, $10 \to 50$	9.4×10^{-4}
Flow-SSM, $10 \to 10$	2.4×10^{-4}
Flow-SSM, $10 \to 50$	4.7×10^{-4}

and evolving them under constant-velocity motion. For Moving MNIST, each digit follows a constant 2D translation with velocity sampled from

$$\{-2, -1, 0, 1, 2\} \times \{-2, -1, 0, 1, 2\}. \tag{2}$$

For Rotating MNIST, each digit rotates with a constant angular velocity sampled from

$$\{-40, -20, 0, 20, 40\} \tag{3}$$

degrees per frame. We consider two direct forecasting protocols: given 10 frames, predict 10 future frames, and given 10 frames, predict 50 future frames. The first setting is trained and evaluated on 20-frame sequences, while the second is trained on 60-frame sequences for 10-to-50 prediction.

Results. Table 1 shows that Flow-SSM achieves very low prediction error on both Moving and Rotating MNIST. In the 10-to-10 setting, it matches or improves upon FERNN, indicating that the flow-aligned selective scan behaves accurately when the motion structure is explicit. It also remains accurate in the 10-to-50 setting, showing that the model can stably capture longer consistent motion trajectories, as illustrated qualitatively in Figure 3.

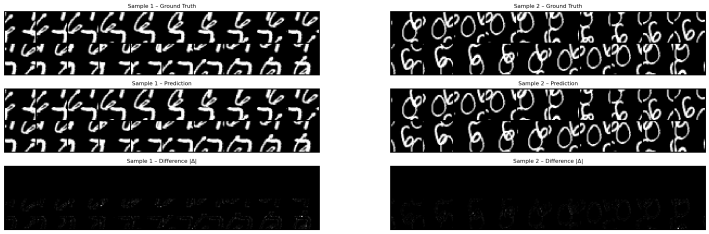

Figure 3: Qualitative prediction examples on Moving MNIST. The first row shows the ground-truth future sequence, the second row shows the model prediction, and the third row visualizes the absolute error. The consistently small error maps indicate that the predicted frames closely follow the ground-truth.

3.2 KTH action recognition under induced motion

Task overview. We use KTH action recognition [17] under induced motion as an intermediate benchmark between controlled synthetic sequences and more complex real-world spatiotemporal data. The benchmark is built from KTH videos of 25 subjects performing 6 action classes. On top of the original action content, we impose additional translation flows on each clip to simulate camera motion. We consider two motion settings:

$$V_1 = \{-1, 0, 1\} \times \{-1, 0, 1\}, \qquad V_2 = \{-2, -1, 0, 1, 2\} \times \{-2, -1, 0, 1, 2\}. \tag{4}$$

Example clips under induced translation are shown in Figure 4. We compare plain Mamba, FERNN, and Flow-SSM under the same classification protocol. In addition to accuracy, we analyze internal branch behavior to test whether the model uses flow-aligned reference frames in a motion-consistent way.

Main classification results. Table 2 reports the main classification accuracy under induced motion. Here $V_i \to V_j$ denotes training on clips augmented with induced translation velocities from V_i and

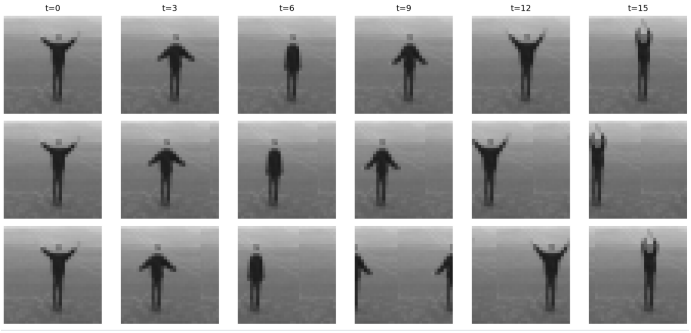

Figure 4: KTH action clips under induced translation. Rows correspond to induced horizontal velocities $v_x = 0$, $v_x = 1$, and $v_x = 2$, while columns correspond to time steps $t = 0, 3, 6, 9, 12, 15$.

testing on clips augmented with velocities from V_j, therefore $V_1 \to V_2$ evaluates generalization to a broader test-time motion range, while $V_2 \to V_2$ is the matched-motion setting. Flow-SSM substantially outperforms both FERNN and plain Mamba in both cases.

Table 2: KTH action recognition under induced motion. Higher classification accuracy is better.

Model	Parameters	$V_1 \to V_2$	$V_2 \to V_2$
Mamba	0.53M	0.477	0.497
FERNN	0.54M	0.540	0.571
Flow-SSM	0.53M	**0.697**	**0.729**

Internal flow mechanism analysis Classification accuracy alone does not show whether the model organizes its flow branches in a meaningful way. We therefore evaluate representative checkpoints with branch-level diagnostics. The definition of reported metrics in Table 3 are in the appendix.

Table 3: Representative KTH branch-level mechanism analysis. Higher is better for all metrics.

Model	Setting	Hit@1 ↑	Top-5 ↑	Drop-GT Acc. ↑
FERNN	$V_1 \to V_2$	0.299	0.882	0.137
Flow-SSM	$V_1 \to V_2$	**0.308**	**0.944**	**0.379**
FERNN	$V_2 \to V_2$	0.183	0.562	0.204
Flow-SSM	$V_2 \to V_2$	**0.320**	**0.961**	**0.520**

The mechanism analysis suggests that Flow-SSM develops a more structured and robust use of its velocity branches. In both $V_1 \to V_2$ and $V_2 \to V_2$, the flow-consistent branch is more frequently ranked among the high-importance branches, indicating better alignment between the model and the ground truth. At the same time, Flow-SSM maintains higher accuracy after the ground-truth branch is removed. This suggests that predictive information is distributed across nearby flow branches rather than concentrated in a single velocity hypothesis.

3.3 Natural-video prediction on UCF-101

Task overview. We further evaluate Flow-SSM on self-supervised video prediction using UCF-101 [18]. Unlike the controlled motion in MNIST or the induced translations in KTH, natural videos contain camera motion, nonrigid human motion, and spatially heterogeneous dynamics that are not constrained to our discrete velocity branches. We observe 8 frames and predict the next 4, comparing Mamba, FERNN, and Flow-SSM under comparable computational budgets. We additionally evaluate a high-motion subset to examine robustness under more pronounced motion.

Table 4: UCF-101 video prediction results. Higher PSNR and SSIM indicate better prediction quality.

Model	PSNR↑	SSIM↑	High-motion PSNR↑	High-motion SSIM↑
Mamba	19.874	0.679	17.416	0.554
FERNN	19.738	0.673	17.278	0.556
Flow-SSM	**21.064**	**0.717**	**19.106**	**0.644**

Results. Table 4 shows that Flow-SSM outperforms both Mamba and FERNN on natural-video prediction. Compared with Mamba, Flow-SSM improves overall PSNR by 1.190 dB, with a larger gain of 1.690 dB on the high-motion subset. The corresponding SSIM improvements further support this advantage. These results suggest that trajectory-aligned recurrence remains effective under complex, nonrigid motion, even when the underlying trajectories do not exactly match the predefined velocity branches.

3.4 SEVIR weather nowcasting

Task overview. We use SEVIR [19] as a realistic large-scale nowcasting benchmark [2] beyond controlled motion datasets. SEVIR is a storm event dataset for short-term weather forecasting, and we focus on its VIL modality, which represents vertically integrated liquid water content from radar observations. In this task, a model observes past VIL radar frames and predicts future frames. We compare four models under the same forecasting protocol: Mamba, FERNN, Flow-SSM, and Hierarchical Flow-SSM. We report threshold-based nowcasting scores, including CSI74 and CSI133, which evaluate precipitation prediction at two intensity levels. Detailed data preprocessing, input-output configurations, and training settings are provided in the appendix.

Main forecasting results. We train all models under the $8 \to 8$ protocol, where $a \to b$ means observing a past VIL frames and predicting the next b frames. Table 5 reports both the matched $8 \to 8$ setting and a longer $12 \to 12$ evaluation. We use CSI74 and CSI133 as threshold-based precipitation scores.

Table 5: SEVIR VIL nowcasting results. CSI74 and CSI133 are threshold-based precipitation scores; higher is better. Results are mean $\pm$ sample standard deviation over five independent seeds, except for the single-run Mamba baseline.

Model	Parameters	$8 \to 8$ CSI74	$8 \to 8$ CSI133	$12 \to 12$ CSI74	$12 \to 12$ CSI133
Mamba	1.221M	0.597	0.226	0.540	0.160
FERNN	1.700M	0.640 ± 0.007	0.258 ± 0.006	0.582 ± 0.005	0.173 ± 0.005
Flow-SSM	1.333M	0.638 ± 0.006	0.251 ± 0.008	0.581 ± 0.006	0.172 ± 0.010
Hier Flow-SSM	1.340M	$\mathbf{0.652} \pm 0.004$	$\mathbf{0.264} \pm 0.005$	$\mathbf{0.591} \pm 0.008$	$\mathbf{0.180} \pm 0.009$

All the flow-aware models consistently outperform plain Mamba, showing that motion-aligned recurrence improves forecasting performance on realistic radar sequences. Hier Flow-SSM gives the best scores across all reported settings with nearly the same parameter count as Flow-SSM. Its gain on CSI133 suggests that hierarchy helps combine coarse storm motion with local high-intensity precipitation structure; additional cross-scale probes are provided in Appendix B.

4 Discussion

Limitations and Future Work. Hierarchical Flow-SSM may benefit spatiotemporal forecasting and video understanding through motion-consistent sequence modeling. However, deployment in safety-critical settings still requires calibration, uncertainty awareness, and human oversight, since prediction errors may affect downstream decisions. The model has several limitations: the exact formulation relies on a discrete set of constant-velocity flow generators V. This shift representation

makes memory and computation scale with the size of the generator set, and truncating V to a finite set can introduce boundary equivariance errors for motions outside the represented range. A natural future direction is to reduce the large generator sets through frequency-domain structure. Since the comoving formulation turns flow transformations into shifts along the generator axis, Fourier or band-limited representations of V could provide a multi-resolution compression of motion hypotheses while preserving a structured description of the flow action. Finally, real physical systems may involve acceleration, deformation, splitting and merging structures, or other non-autonomous dynamics beyond the constant or locally persistent motions studied here. Future work should explore continuous or steerable flow generators [20, 21], principled dynamic routing with controllable approximate-equivariance error, and sparse tensor or custom CUDA implementations that preserve the hardware efficiency of state space models.

Related Work. Our work sits at the intersection of state space sequence models, dynamic equivariance, and geometric deep learning. State space models [22, 6, 7], and particularly Mamba [4, 8], have revolutionized sequence modeling through input-dependent selective scans and hardware-efficient linear-time complexity. However, by processing sequences in a flat, single-scale manner, they accumulate disconnected traces at fixed spatial coordinates rather than tracking moving entities, leaving them structurally unaware of the geometric and dynamic symmetries underlying physical data [23, 24, 25]. Recently, FERNNs [9] addressed this by introducing a formal framework for time-parameterized symmetries, demonstrating that lifting hidden states to explicitly moving reference frames allows models to track continuous flows safely. Yet, FERNNs operate exclusively on a single spatial and velocity scale, fundamentally missing the hierarchical organization of natural dynamics where local interactions form macroscopic structures. Other efforts to incorporate continuous time or motion symmetries [26, 27, 28] typically face similar structural bottlenecks. In parallel, geometric deep learning has explored multiscale equivariance through various mechanisms: hierarchical graph networks utilizing E(n)-equivariant pooling [16], steerable filters on arbitrary continuous Lie groups [29, 30, 31, 32, 33], hierarchical representations in hyperbolic spaces via Cartan networks [34], and equivariant normalizing flows [35, 36, 37]. While powerful, these models predominantly focus on static spatial transformations rather than temporal flows. Hierarchical Flow-SSM extends flow-equivariant recurrent modeling to selective state-space architectures with hierarchical transport. By formalizing a multiscale framework that supports exact cross-scale communication under continuous flow transformations, we endow modern state space models with both dynamic flow equivariance and explicit structural hierarchy.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 62676211).

References

[1] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning (ICML)*, 2015.

[2] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[3] Haixu Wu, Zhiyu Yao, Jianmin Wang, and Mingsheng Long. MotionRNN: A flexible model for video prediction with spacetime-varying motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15435–15444, 2021.

[4] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*, 2024.

[5] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[6] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

[7] Jimmy T.H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[8] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024.

[9] T. Anderson Keller. Flow equivariant recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[10] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[11] Taco S Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns. In *International Conference on Learning Representations (ICLR)*, 2018.

[12] Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2018.

[13] Daniel E. Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[14] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[15] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[16] Jiaqi Han, Wenbing Huang, Tingyang Xu, and Yu Rong. Equivariant graph hierarchy-based neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[17] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local SVM approach. In *International Conference on Pattern Recognition (ICPR)*, 2004.

[18] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[19] Mark Veillette, Siddharth Samsi, and Christopher Mattioli. SEVIR: A storm event imagery dataset for deep learning applications in radar and satelite meteorology. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. In *International Conference on Learning Representations (ICLR)*, 2023.

[21] Bahar Azari and Deniz Erdogmus. Equivariant deep dynamical model for motion prediction. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 11655–11668. PMLR, 2022.

[22] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[24] Wenhao Zhang, Ying Nian Wu, and Si Wu. Translation-equivariant representation in recurrent networks with a continuous manifold of attractors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[25] Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *International Conference on Learning Representations (ICLR)*, 2021.

[26] Alex Zihao Zhu, Ziyun Wang, and Kostas Daniilidis. Motion equivariant networks for event cameras with the temporal normalization transform. *arXiv preprint arXiv:1902.06820*, 2019.

[27] T. Anderson Keller and Max Welling. Topographic VAEs learn equivariant capsules. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[28] T Anderson Keller and Max Welling. Neural wave machines: Learning spatiotemporally structured representations with locally coupled oscillatory recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 2023.

[29] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew G Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning (ICML)*, 2020.

[30] David W Romero, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. Attentive group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, 2020.

[31] Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build E(N)-equivariant steerable cnns. In *International Conference on Learning Representations (ICLR)*, 2022.

[32] Maksim Zhdanov, David Ruhe, Maurice Weiler, Ana Lucic, Johannes Brandstetter, and Patrick Forré. Clifford-steerable convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2024.

[33] Nichita Diaconu and Daniel Worrall. Learning to convolve: A generalized weight-tying approach. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1586–1595. PMLR, 2019.

[34] Federico Milanesio, Matteo Santoro, Pietro Giuseppe Fre, and Guido Sanguinetti. Cartan networks: Group theoretical hyperbolic deep learning. *arXiv preprint arXiv:2505.24353*, 2025.

[35] Victor Garcia Satorras, Emiel Hoogeboom, Fabian B Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[36] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning (ICML)*, 2020.

[37] Robert Mahony, Pieter van Goor, and Tarek Hamel. Observer design for nonlinear systems with equivariance. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:221–252, 2022.

[38] NOAA Warning Decision Training Division. Vertically integrated liquid (vil). `https://vlab.noaa.gov/web/wdtd/-/vertically-integrated-liquid-vil-`.

Appendix Contents

A Additional Method Details and Equivariance Proofs **14**

A.1 Selective Discretization and Input Lifts 14

A.2 Flow Equivariance of Plain Flow-SSM 15

A.2.1 Left-comoving coordinates 16

A.2.2 Covariance of coefficients and input lifts 16

A.3 Equivariance of Hierarchical Cross-Scale Communication 18

A.3.1 Hierarchical quotient spaces and shift representations 18

A.3.2 Communication requirement 18

A.4 An Exact Quotient-Pullback Communicator 19

B Experiment Details **20**

B.1 Moving and Rotating MNIST 20

B.2 KTH motion recognition 23

B.3 UCF101 Natural-Video Prediction 25

B.4 SEVIR forecast 26

B.5 Computational Resources 29

A Additional Method Details and Equivariance Proofs

This appendix provides additional details omitted from the main method section. We first specify the selective state-space discretization and the input-lift variants used by Flow-SSM. We then prove the flow equivariance of plain Flow-SSM in left-comoving coordinates, and finally prove that the static hierarchical communication preserves equivariance when each cross-scale communicator is an intertwiner.

A.1 Selective Discretization and Input Lifts

Selective fields. Let G be the spatial group and let $x_t \in \mathcal{F}(G;\mathbb{R}^C)$ be the input frame at time t. At each spatial location $g \in G$, the selective fields are generated by G-equivariant maps:

$$\Delta_t(g) = \mathrm{softplus}\big(\Phi_\Delta(x_t)(g)\big), \tag{5}$$
$$B_t(g) = \Phi_B(x_t)(g), \tag{6}$$
$$C_t(g) = \Phi_C(x_t)(g), \tag{7}$$
$$D_t(g) = \Phi_D(x_t)(g). \tag{8}$$

Here $\Delta_t(g) > 0$ is the step size used to discretize the continuous state-space dynamics, while $B_t(g), C_t(g), D_t(g)$ are the input, readout, and skip coefficients.

Pointwise selective SSM discretization. Let $A \in \mathbb{R}^{N\times N}$ be the continuous-time state matrix [5, 6, 7] shared over space and time. In the diagonal Mamba-style parameterization, A is usually taken to be diagonal or channel-wise, with stable entries. At each spatial location g, we discretize the continuous-time system

$$\frac{ds(\tau)}{d\tau} = As(\tau) + B_t(g)u(\tau) \tag{9}$$

with a zero-order hold over an interval of length $\Delta_t(g)$. This gives

$$\bar{A}_t(g) = F_A\big(\Delta_t(g), A\big) := \exp\big(\Delta_t(g)A\big), \tag{10}$$
$$\bar{B}_t(g) = F_B\big(\Delta_t(g), A, B_t(g)\big) := \int_0^{\Delta_t(g)} \exp\big((\Delta_t(g)-\tau)A\big)\, d\tau\, B_t(g). \tag{11}$$

When A is invertible, the second expression can be written as

$$\bar{B}_t(g) = A^{-1}\big(\exp(\Delta_t(g)A) - I\big)B_t(g). \tag{12}$$

For diagonal $A = \mathrm{diag}(a_1, \dots, a_N)$, this becomes the elementwise formula

$$\bar{A}_{t,i}(g) = \exp\big(\Delta_t(g)a_i\big), \tag{13}$$
$$\bar{B}_{t,i}(g) = \begin{cases} \dfrac{\exp(\Delta_t(g)a_i) - 1}{a_i} B_{t,i}(g), & a_i \neq 0, \\ \Delta_t(g)B_{t,i}(g), & a_i = 0. \end{cases} \tag{14}$$

In the main text we write this compactly as

$$\bar{A}_t(g) = F_A(\Delta_t(g), A), \qquad \bar{B}_t(g) = F_B(\Delta_t(g), A, B_t(g)). \tag{15}$$

The equivariance proofs below only require that F_A and F_B are applied pointwise in g. Therefore the proof is unchanged for stabilized diagonal implementations or other pointwise selective-scan discretizations [4, 8].

Trivial input lift. The simplest input lift [11, 12, 15] copies the same spatial feature to every generator channel:

$$u_t[x](v, g) := \Phi_{\mathrm{in}}(x_t)(g), \qquad v \in V, g \in G. \tag{16}$$

This is the safest exact lift: it does not introduce generator-dependent spatial mixing before the recurrent scan.

Non-trivial spatial input lift. A richer input lift can mix nearby spatial positions while preserving equivariance. Let $\mathcal{N}_G \subset G$ be a finite local support set, and let U_a be learned channel matrices for $a \in \mathcal{N}_G$. Define

$$u_t[x](v, g) := \sum_{a \in \mathcal{N}_G} U_a \, \Phi_{\mathrm{in}}(x_t)(ga). \tag{17}$$

The right multiplication by a means that the same local stencil is used at every spatial location. Hence this is a group-convolutional input lift and is equivariant under the left action of G. For image translations, Eq. (17) reduces to an ordinary local convolutional lift.

Flow-aligned temporal input lift. The input lift may also mix previous frames after aligning them according to the candidate flow generator. A flow-aligned FIR lift is

$$u_t[x](v, g) := \sum_{k=0}^{K-1} \sum_{a \in \mathcal{N}_G} U_{k,a} \, \Phi_{\mathrm{in}}(x_{t-k})\big(\psi_k(v)^{-1} ga\big). \tag{18}$$

This lift first transports the past frame x_{t-k} by $\psi_k(v)^{-1}$, thereby placing it in the reference frame associated with generator v, and then applies a fixed local spatial stencil. Eq. (16) is recovered from Eq. (18) by taking $K = 1$ and $\mathcal{N}_G = \{e\}$.

A.2 Flow Equivariance of Plain Flow-SSM

Spatial action and flow action. Let G be the spatial group. We use the standard left action [15, 10] of G on signals $x \in \mathcal{F}(G; \mathbb{R}^C)$:

$$(\gamma \cdot x)(g) := x(\gamma^{-1} g), \qquad \gamma, g \in G. \tag{19}$$

Let V be the generator set, and let $\psi_t(v) \in G$ be the time-t flow element generated by $v \in V$. For a fixed input-flow generator $\hat{v} \in V$, define

$$(\mathcal{T}_{\hat{v}} x)_t(g) := x_t\big(\psi_t(\hat{v})^{-1} g\big). \tag{20}$$

Thus frame t is transformed by the time-dependent group element $\psi_t(\hat{v})$.

Plain Flow-SSM recurrence. The lifted state is

$$s_t[x] \in \mathcal{F}(V \times G; \mathbb{R}^N), \qquad s_t[x](v, g) \in \mathbb{R}^N. \tag{21}$$

We use the convention that s_t is the state after processing frame x_t. The transition is

$$s_t[x](v, g) = \bar{A}_t[x](g) s_{t-1}[x]\big(v, \psi_1(v)^{-1} g\big) + \bar{B}_t[x](g) u_t[x](v, g). \tag{22}$$

The readout is

$$y_t[x](v, g) = C_t[x](g) s_t[x](v, g) + D_t[x](g) u_t[x](v, g). \tag{23}$$

Conditions. The following conditions are satisfied by the additive flow setting used in our model. First, the generator set is closed under relative shifts:

$$v - \hat{v} \in V, \qquad \forall v, \hat{v} \in V. \tag{24}$$

Second, for each fixed generator v, the flow satisfies the same-generator composition law

$$\psi_0(v) = e, \qquad \psi_a(v) \psi_b(v) = \psi_{a+b}(v). \tag{25}$$

In particular,

$$\psi_1(v)^{-1} \psi_t(v) = \psi_{t-1}(v). \tag{26}$$

Third, the flow family satisfies the relative-flow identity

$$\psi_t(\hat{v})^{-1} \psi_t(v) = \psi_t(v - \hat{v}). \tag{27}$$

For translation flows, this is simply the statement that removing a background velocity $\hat{v}$ from v gives the relative velocity $v - \hat{v}$.

Fourth, the selective parameter generators are G-equivariant. For any generator $\Phi \in \{\Phi_\Delta, \Phi_B, \Phi_C, \Phi_D, \Phi_{\mathrm{in}}\}$,

$$\Phi[\gamma \cdot z](g) = \Phi[z](\gamma^{-1} g). \tag{28}$$

Since the SSM discretization is pointwise in g, the discretized coefficients inherit the same covariance:

$$\bar{A}_t[\mathcal{T}_{\hat{v}} x](g) = \bar{A}_t[x]\big(\psi_t(\hat{v})^{-1} g\big), \tag{29}$$

$$\bar{B}_t[\mathcal{T}_{\hat{v}} x](g) = \bar{B}_t[x]\big(\psi_t(\hat{v})^{-1} g\big), \tag{30}$$

and similarly for C_t and D_t. Finally, the initial state is equivariant; in the standard case we use zero initialization.

A.2.1 Left-comoving coordinates

The direct proof in the original coordinates (v, g) is cumbersome because Eq. (22) contains the transported recurrent lookup $s_{t-1}(v, \psi_1(v)^{-1}g)$. We remove this transport by defining the left-comoving state

$$r_t[x](v, h) := s_t[x]\big(v, \psi_t(v)h\big). \tag{31}$$

Equivalently,

$$s_t[x](v, g) = r_t[x]\big(v, \psi_t(v)^{-1}g\big). \tag{32}$$

The coordinate h is the spatial coordinate in the reference frame moving with generator v.

Substituting $g = \psi_t(v)h$ into Eq. (22), we get

$$r_t[x](v, h) = \bar{A}_t[x]\big(\psi_t(v)h\big)s_{t-1}[x]\big(v, \psi_1(v)^{-1}\psi_t(v)h\big) \tag{33}$$

$$+ \bar{B}_t[x]\big(\psi_t(v)h\big)u_t[x]\big(v, \psi_t(v)h\big). \tag{34}$$

Using Eq. (26),

$$s_{t-1}[x]\big(v, \psi_1(v)^{-1}\psi_t(v)h\big) = s_{t-1}[x]\big(v, \psi_{t-1}(v)h\big) = r_{t-1}[x](v, h). \tag{35}$$

Define

$$\widetilde{A}_t[x](v, h) := \bar{A}_t[x]\big(\psi_t(v)h\big), \tag{36}$$

$$\widetilde{B}_t[x](v, h) := \bar{B}_t[x]\big(\psi_t(v)h\big), \tag{37}$$

$$\widetilde{u}_t[x](v, h) := u_t[x]\big(v, \psi_t(v)h\big). \tag{38}$$

Then the recurrence becomes pointwise in (v, h):

$$r_t[x](v, h) = \widetilde{A}_t[x](v, h)r_{t-1}[x](v, h) + \widetilde{B}_t[x](v, h)\widetilde{u}_t[x](v, h). \tag{39}$$

This is the key simplification. It uses only the natural same-generator flow composition identity and does not require any commutation between h and $\psi_t(v)$.

A.2.2 Covariance of coefficients and input lifts

We first show that the comoving coefficients transform by a shift on the generator axis. For $\widetilde{A}_t$,

$$\widetilde{A}_t[\mathcal{T}_{\hat{v}}x](v, h) = \bar{A}_t[\mathcal{T}_{\hat{v}}x]\big(\psi_t(v)h\big) \tag{40}$$

$$= \bar{A}_t[x]\big(\psi_t(\hat{v})^{-1}\psi_t(v)h\big) \tag{41}$$

$$= \bar{A}_t[x]\big(\psi_t(v - \hat{v})h\big) \tag{42}$$

$$= \widetilde{A}_t[x](v - \hat{v}, h), \tag{43}$$

where we used Eq. (27). The same argument gives

$$\widetilde{B}_t[\mathcal{T}_{\hat{v}}x](v, h) = \widetilde{B}_t[x](v - \hat{v}, h), \tag{44}$$

$$\widetilde{C}_t[\mathcal{T}_{\hat{v}}x](v, h) = \widetilde{C}_t[x](v - \hat{v}, h), \tag{45}$$

$$\widetilde{D}_t[\mathcal{T}_{\hat{v}}x](v, h) = \widetilde{D}_t[x](v - \hat{v}, h). \tag{46}$$

For the trivial input lift in Eq. (16),

$$\widetilde{u}_t[\mathcal{T}_{\hat{v}}x](v, h) = \Phi_{\text{in}}((\mathcal{T}_{\hat{v}}x)_t)\big(\psi_t(v)h\big) \tag{47}$$

$$= \Phi_{\text{in}}(x_t)\big(\psi_t(\hat{v})^{-1}\psi_t(v)h\big) \tag{48}$$

$$= \Phi_{\text{in}}(x_t)\big(\psi_t(v - \hat{v})h\big) \tag{49}$$

$$= \widetilde{u}_t[x](v - \hat{v}, h). \tag{50}$$

For the non-trivial flow-aligned lift in Eq. (18), we have

$$\widetilde{u}_t[x](v, h) = \sum_{k=0}^{K-1} \sum_{a \in \mathcal{N}_G} U_{k,a}\, \Phi_{\text{in}}(x_{t-k})\big(\psi_k(v)^{-1}\psi_t(v)ha\big) \tag{51}$$

$$= \sum_{k=0}^{K-1} \sum_{a \in \mathcal{N}_G} U_{k,a}\, \Phi_{\text{in}}(x_{t-k})\big(\psi_{t-k}(v)ha\big). \tag{52}$$

Under the transformed input,

$$\widetilde{u}_t[\mathcal{T}_{\hat{v}}x](v,h) = \sum_{k=0}^{K-1}\sum_{a\in\mathcal{N}_G} U_{k,a}\,\Phi_{\mathrm{in}}((\mathcal{T}_{\hat{v}}x)_{t-k})\big(\psi_k(v)^{-1}\psi_t(v)ha\big) \tag{53}$$

$$= \sum_{k=0}^{K-1}\sum_{a\in\mathcal{N}_G} U_{k,a}\,\Phi_{\mathrm{in}}(x_{t-k})\big(\psi_{t-k}(\hat{v})^{-1}\psi_k(v)^{-1}\psi_t(v)ha\big) \tag{54}$$

$$= \sum_{k=0}^{K-1}\sum_{a\in\mathcal{N}_G} U_{k,a}\,\Phi_{\mathrm{in}}(x_{t-k})\big(\psi_{t-k}(\hat{v})^{-1}\psi_{t-k}(v)ha\big) \tag{55}$$

$$= \sum_{k=0}^{K-1}\sum_{a\in\mathcal{N}_G} U_{k,a}\,\Phi_{\mathrm{in}}(x_{t-k})\big(\psi_{t-k}(v-\hat{v})ha\big) \tag{56}$$

$$= \widetilde{u}_t[x](v-\hat{v},h). \tag{57}$$

Thus both trivial and non-trivial flow-aligned input lifts transform by the same generator-axis shift.

Theorem 1 (Flow equivariance of plain Flow-SSM)**.** *Under the conditions above, the plain Flow-SSM hidden state is flow-equivariant in left-comoving coordinates:*

$$r_t[\mathcal{T}_{\hat{v}}x](v,h) = r_t[x](v-\hat{v},h), \qquad \forall t,\ v,\hat{v}\in V,\ h\in G. \tag{58}$$

The comoving readout satisfies the same transformation rule:

$$\widetilde{y}_t[\mathcal{T}_{\hat{v}}x](v,h) = \widetilde{y}_t[x](v-\hat{v},h). \tag{59}$$

Equivalently, in original lifted coordinates,

$$s_t[\mathcal{T}_{\hat{v}}x](v,g) = s_t[x]\big(v-\hat{v},\psi_t(\hat{v})^{-1}g\big). \tag{60}$$

Proof. We prove Eq. (58) by induction on t.

Base case. With zero initialization, $r_0[x](v,h)=0$ and $r_0[\mathcal{T}_{\hat{v}}x](v,h)=0$. Therefore

$$r_0[\mathcal{T}_{\hat{v}}x](v,h) = r_0[x](v-\hat{v},h). \tag{61}$$

Inductive step. Assume that

$$r_{t-1}[\mathcal{T}_{\hat{v}}x](v,h) = r_{t-1}[x](v-\hat{v},h). \tag{62}$$

Using the comoving recurrence Eq. (39) under the transformed input,

$$r_t[\mathcal{T}_{\hat{v}}x](v,h) = \widetilde{A}_t[\mathcal{T}_{\hat{v}}x](v,h)r_{t-1}[\mathcal{T}_{\hat{v}}x](v,h) \tag{63}$$

$$+\,\widetilde{B}_t[\mathcal{T}_{\hat{v}}x](v,h)\widetilde{u}_t[\mathcal{T}_{\hat{v}}x](v,h). \tag{64}$$

By coefficient covariance, input-lift covariance, and the induction hypothesis,

$$r_t[\mathcal{T}_{\hat{v}}x](v,h) = \widetilde{A}_t[x](v-\hat{v},h)r_{t-1}[x](v-\hat{v},h) \tag{65}$$

$$+\,\widetilde{B}_t[x](v-\hat{v},h)\widetilde{u}_t[x](v-\hat{v},h). \tag{66}$$

The right-hand side is exactly Eq. (39) evaluated at $(v-\hat{v},h)$. Thus

$$r_t[\mathcal{T}_{\hat{v}}x](v,h) = r_t[x](v-\hat{v},h). \tag{67}$$

This proves the induction.

For the readout, define $\widetilde{y}_t[x](v,h) := y_t[x](v,\psi_t(v)h)$. Then

$$\widetilde{y}_t[x](v,h) = \widetilde{C}_t[x](v,h)r_t[x](v,h) + \widetilde{D}_t[x](v,h)\widetilde{u}_t[x](v,h). \tag{68}$$

Using covariance of $\widetilde{C}_t$ and $\widetilde{D}_t$, state equivariance, and input-lift covariance gives Eq. (59).

Finally, converting back to original coordinates using Eq. (32),

$$s_t[\mathcal{T}_{\hat{v}}x](v,g) = r_t[\mathcal{T}_{\hat{v}}x]\big(v,\psi_t(v)^{-1}g\big) \tag{69}$$

$$= r_t[x]\big(v-\hat{v},\psi_t(v)^{-1}g\big). \tag{70}$$

Let $w=v-\hat{v}$. Since $s_t[x](w,g') = r_t[x](w,\psi_t(w)^{-1}g')$, choose

$$g' = \psi_t(w)\psi_t(v)^{-1}g. \tag{71}$$

Using $\psi_t(\hat{v})^{-1}\psi_t(v) = \psi_t(v-\hat{v})$, we have

$$\psi_t(w)\psi_t(v)^{-1} = \psi_t(\hat{v})^{-1}. \tag{72}$$

Therefore Eq. (60) follows. □

A.3 Equivariance of Hierarchical Cross-Scale Communication

We now prove that the static hierarchical extension [15, 11] preserves equivariance when every cross-scale communicator is an intertwiner [15]. The local recurrence at each scale pair follows from Theorem 1; we therefore focus on the communication term.

A.3.1 Hierarchical quotient spaces and shift representations

Let $V^\star$ be an ambient generator group and choose a subgroup chain

$$\{0\} = M_0 \subset M_1 \subset \cdots \subset M_M \subset V^\star. \tag{73}$$

The generator level m is

$$V_m := V^\star/M_m, \qquad \pi_m^V : V^\star \to V_m. \tag{74}$$

Similarly, let $G^\star$ be an ambient spatial group and choose normal subgroups

$$\{e\} = N_0 \lhd N_1 \lhd \cdots \lhd N_L \lhd G^\star. \tag{75}$$

The spatial level ℓ is

$$G_\ell := G^\star/N_\ell, \qquad \pi_\ell^G : G^\star \to G_\ell. \tag{76}$$

For an active scale pair (m, ℓ), the comoving state is

$$r_t^{m,\ell}[x] \in \mathcal{F}(V_m \times G_\ell; \mathbb{R}^{N_{m,\ell}}). \tag{77}$$

An ambient generator $\hat{v} \in V^\star$ induces a shift on V_m through $\pi_m^V(\hat{v})$. Define

$$(T_m(\hat{v})r)(v, h) := r\big(v - \pi_m^V(\hat{v}), h\big), \qquad v \in V_m,\ h \in G_\ell. \tag{78}$$

The target equivariance statement is

$$r_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = T_m(\hat{v})r_t^{m,\ell}[x]. \tag{79}$$

A.3.2 Communication requirement

For each active pair, write the hierarchical update as

$$r_t^{m,\ell} = \mathrm{Local}_t^{m,\ell} + \eta_t^{m,\ell}, \tag{80}$$

where $\mathrm{Local}_t^{m,\ell}$ is the within-scale Flow-SSM update and

$$\eta_t^{m,\ell}[x] := \sum_{(p,q)\to(m,\ell)} K_{(p,q)\to(m,\ell)} r_{t-1}^{p,q}[x]. \tag{81}$$

The communicator has type

$$K_{(p,q)\to(m,\ell)} : \mathcal{F}(V_p \times G_q; \mathbb{R}^{N_{p,q}}) \to \mathcal{F}(V_m \times G_\ell; \mathbb{R}^{N_{m,\ell}}). \tag{82}$$

Because $\eta_t^{m,\ell}$ is added directly to the target state update, it must transform under the same target representation T_m. Therefore we require

$$K_{(p,q)\to(m,\ell)} T_p(\hat{v}) = T_m(\hat{v}) K_{(p,q)\to(m,\ell)}, \qquad \forall \hat{v} \in V^\star. \tag{83}$$

This is precisely the intertwining relation between the source and target shift representations.

Lemma 1 (Equivariance of cross-scale messages)**.** *Suppose every communicator satisfies Eq.* (83)*. If every source state satisfies*

$$r_{t-1}^{p,q}[\mathcal{T}_{\hat{v}}x] = T_p(\hat{v}) r_{t-1}^{p,q}[x], \tag{84}$$

then the total message into (m, ℓ) *satisfies*

$$\eta_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = T_m(\hat{v}) \eta_t^{m,\ell}[x]. \tag{85}$$

Proof. Using the definition of the message,

$$\eta_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = \sum_{(p,q)\to(m,\ell)} K_{(p,q)\to(m,\ell)} r_{t-1}^{p,q}[\mathcal{T}_{\hat{v}}x] \tag{86}$$

$$= \sum_{(p,q)\to(m,\ell)} K_{(p,q)\to(m,\ell)} T_p(\hat{v}) r_{t-1}^{p,q}[x] \tag{87}$$

$$= \sum_{(p,q)\to(m,\ell)} T_m(\hat{v}) K_{(p,q)\to(m,\ell)} r_{t-1}^{p,q}[x] \tag{88}$$

$$= T_m(\hat{v}) \sum_{(p,q)\to(m,\ell)} K_{(p,q)\to(m,\ell)} r_{t-1}^{p,q}[x] \tag{89}$$

$$= T_m(\hat{v}) \eta_t^{m,\ell}[x]. \tag{90}$$

□

Theorem 2 (Flow equivariance of static hierarchical Flow-SSM)**.** *Suppose each active scale pair satisfies the local Flow-SSM equivariance conditions at its quotient level, and suppose every cross-scale communicator satisfies the intertwining relation in Eq.* (83)*. Then the static hierarchical Flow-SSM state is flow-equivariant at every active scale pair:*

$$r_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = T_m(\hat{v}) r_t^{m,\ell}[x], \qquad \forall (m,\ell),\ t,\ \hat{v} \in V^\star. \tag{91}$$

Equivalently,

$$r_t^{m,\ell}[\mathcal{T}_{\hat{v}}x](v,h) = r_t^{m,\ell}[x]\big(v - \pi_m^V(\hat{v}), h\big). \tag{92}$$

Proof. We prove the claim by induction over time, simultaneously for all active scale pairs.

Base case. With zero initialization, $r_0^{m,\ell}[x] \equiv 0$ for every active pair. Hence

$$r_0^{m,\ell}[\mathcal{T}_{\hat{v}}x] = T_m(\hat{v}) r_0^{m,\ell}[x]. \tag{93}$$

Inductive step. Assume that at time $t-1$, every active source pair satisfies

$$r_{t-1}^{p,q}[\mathcal{T}_{\hat{v}}x] = T_p(\hat{v}) r_{t-1}^{p,q}[x]. \tag{94}$$

Fix a target pair (m,ℓ). Under the transformed input,

$$r_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = \mathrm{Local}_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] + \eta_t^{m,\ell}[\mathcal{T}_{\hat{v}}x]. \tag{95}$$

The local term is equivariant by the scale-wise version of Theorem 1:

$$\mathrm{Local}_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = T_m(\hat{v})\, \mathrm{Local}_t^{m,\ell}[x]. \tag{96}$$

The message term is equivariant by Lemma 1:

$$\eta_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = T_m(\hat{v}) \eta_t^{m,\ell}[x]. \tag{97}$$

Therefore

$$r_t^{m,\ell}[\mathcal{T}_{\hat{v}}x] = T_m(\hat{v})\, \mathrm{Local}_t^{m,\ell}[x] + T_m(\hat{v}) \eta_t^{m,\ell}[x] \tag{98}$$

$$= T_m(\hat{v}) \big(\mathrm{Local}_t^{m,\ell}[x] + \eta_t^{m,\ell}[x]\big) \tag{99}$$

$$= T_m(\hat{v}) r_t^{m,\ell}[x]. \tag{100}$$

This completes the induction. □

A.4 An Exact Quotient-Pullback Communicator

We now give a concrete communicator satisfying Eq. (83). Suppose the source pair (p,q) is coarser than the target pair (m,ℓ). Then there are quotient maps

$$\pi_{m\to p}^V : V_m \to V_p, \qquad \pi_{\ell\to q}^G : G_\ell \to G_q. \tag{101}$$

Define the coarse-to-fine communicator

$$\big(K_{(p,q)\to(m,\ell)} r\big)(v,h) := W_{(p,q)\to(m,\ell)} r\big(\pi_{m\to p}^V(v), \pi_{\ell\to q}^G(h)\big), \tag{102}$$

where $W_{(p,q)\to(m,\ell)} \in \mathbb{R}^{N_{m,\ell}\times N_{p,q}}$ is a learned channel matrix. The target fine cell (v,h) reads from the source coarse cell $(\pi_{m\to p}^V(v), \pi_{\ell\to q}^G(h))$ and then applies a learned channel mixing.

Proposition. The quotient-pullback communicator in Eq. (102) is an exact intertwiner.

That is,

$$K_{(p,q)\to(m,\ell)}T_p(\hat{v}) = T_m(\hat{v})K_{(p,q)\to(m,\ell)}, \qquad \forall \hat{v} \in V^\star. \tag{103}$$

Proof. For brevity write $K = K_{(p,q)\to(m,\ell)}$ and $W = W_{(p,q)\to(m,\ell)}$. Then

$$(KT_p(\hat{v})r)(v,h) = W(T_p(\hat{v})r)\big(\pi^V_{m\to p}(v), \pi^G_{\ell\to q}(h)\big) \tag{104}$$

$$= Wr\big(\pi^V_{m\to p}(v) - \pi^V_p(\hat{v}), \pi^G_{\ell\to q}(h)\big). \tag{105}$$

Because the quotient projection is compatible with generator shifts,

$$\pi^V_{m\to p}(v) - \pi^V_p(\hat{v}) = \pi^V_{m\to p}\big(v - \pi^V_m(\hat{v})\big). \tag{106}$$

Therefore

$$(KT_p(\hat{v})r)(v,h) = Wr\big(\pi^V_{m\to p}(v - \pi^V_m(\hat{v})), \pi^G_{\ell\to q}(h)\big) \tag{107}$$

$$= (Kr)\big(v - \pi^V_m(\hat{v}), h\big) \tag{108}$$

$$= (T_m(\hat{v})Kr)(v,h). \tag{109}$$

Thus $KT_p(\hat{v}) = T_m(\hat{v})K$, proving the claim. □

Why this communicator is natural. The communicator in Eq. (102) is a quotient pullback: each fine cell receives information from the coarse cell containing it. It uses only fixed quotient maps and a learned channel matrix, so it introduces no input-dependent routing that could break the equivariance relation. The quotient map commutes with generator shifts, and the channel matrix acts only on features, not on generator or spatial coordinates. Therefore "shift then communicate" and "communicate then shift" are exactly the same operation.

B Experiment Details

B.1 Moving and Rotating MNIST

Dataset and task. We use Moving MNIST and Rotating MNIST[1, 9] as controlled sequence prediction benchmarks with known flow structure. Each sequence is constructed by compositing two MNIST digits on a shared canvas and evolving them under independently sampled constant motions.

For Moving MNIST, each frame has shape

$$x_t \in \mathbb{R}^{1\times 28\times 28},$$

and each digit follows a constant two-dimensional translation velocity sampled from

$$V_T = \{(v_x, v_y) : v_x, v_y \in \{-2, -1, 0, 1, 2\}\}.$$

For Rotating MNIST, each frame has shape

$$x_t \in \mathbb{R}^{1\times 34\times 34},$$

and each digit rotates with a constant angular velocity sampled from

$$V_R = \{-40, -20, 0, 20, 40\},$$

where each element is measured in degrees per frame.

In both benchmarks, the model observes the first $T_{\text{in}} = 10$ frames and predicts future frames under two direct forecasting protocols:

$$10 \to 10 \qquad \text{and} \qquad 10 \to 50.$$

The $10 \to 10$ setting uses sequences of total length 20, with

$$X_{\text{in}} \in \mathbb{R}^{10\times C\times H\times W}, \qquad X_{\text{out}} \in \mathbb{R}^{10\times C\times H\times W}.$$

The $10 \to 50$ setting uses sequences of total length 60, with

$$X_{\text{in}} \in \mathbb{R}^{10\times C\times H\times W}, \qquad X_{\text{out}} \in \mathbb{R}^{50\times C\times H\times W}.$$

Thus, the long-horizon setting is trained directly under the $10 \to 50$ protocol rather than evaluated only as zero-shot extrapolation from a $10 \to 10$ model.

Implementation details (Moving MNIST). For Moving MNIST, let $x_t \in \mathbb{R}^{B\times C\times H\times W}$ denote the input frame at time t. Let $M_T = |V_T|$.

FERNN. Let C_h denote the hidden width. FERNN lifts the hidden state over the translation generator set:

$$h_t \in \mathbb{R}^{B\times M_T\times C_h\times H\times W}.$$

Given x_t, an input convolution produces $u_t = \mathrm{Conv}_u(x_t)$, which is shared across velocity branches. For each branch $v \in V_T$, the previous hidden state is transported by the one-step translation

$$\tilde{h}^{(v)}_{t-1}(p) = h^{(v)}_{t-1}(p - v),$$

and then updated by a shared convolutional recurrence:

$$h^{(v)}_t = \sigma\Big(u_t + \mathrm{Conv}_h(\tilde{h}^{(v)}_{t-1})\Big), \qquad v \in V_T.$$

The branch dimension is reduced by max-pooling and decoded into the next frame:

$$f_t = \max_{v\in V_T} h^{(v)}_t, \qquad \hat{x}_{t+1} = g_\theta(f_t).$$

Flow-SSM. Flow-SSM uses the same translation branch set V_T, but replaces the convolutional recurrence with a selective state-space update. The frame encoder gives

$$u_t = \mathrm{Enc}(x_t) \in \mathbb{R}^{B\times D\times H\times W},$$

and the lifted recurrent state is

$$s_t \in \mathbb{R}^{B\times M_T\times N\times H\times W}.$$

From u_t, the model generates branch-shared selective fields Δ_t, B_t, C_t. With a stable diagonal parameterization

$$A = -\exp(\log A_{\mathrm{real}}), \qquad \bar{A}_t = \exp(\Delta_t \odot A),$$

each branch first transports its previous state,

$$\tilde{s}^{(v)}_{t-1}(p) = s^{(v)}_{t-1}(p - v),$$

and then applies the selective update

$$s^{(v)}_t = \bar{A}_t \odot \mathrm{Mix}(\tilde{s}^{(v)}_{t-1}) + \Delta_t \odot (B_t u_t).$$

The branch-wise readout is

$$y^{(v)}_t = C_t s^{(v)}_t + D_{\mathrm{skip}} u_t,$$

followed by branch max-pooling and decoding:

$$f_t = \max_{v\in V_T} y^{(v)}_t, \qquad \hat{x}_{t+1} = g_\theta(f_t).$$

Thus, FERNN and Flow-SSM use the same moving reference frames, while Flow-SSM replaces the additive convolutional recurrent update with branch-shared selective state-space dynamics.

For Moving MNIST, we use

$$C = 1, \qquad H = W = 28, \qquad T_{\mathrm{in}} = 10, \qquad T_{\mathrm{pred}} \in \{10, 50\},$$

$$V_T = \{-2,-1,0,1,2\} \times \{-2,-1,0,1,2\}, \qquad M_T = 25.$$

For Moving MNIST Flow-SSM, we set

$$D = 64, \qquad N = 32.$$

Implementation details (Rotating MNIST). For Rotating MNIST, let $x_t \in \mathbb{R}^{B\times C\times H\times W}$ denote the input frame at time t, and let N_{rot} be the number of discrete orientation channels.

FERNN. Let C_h be the base hidden width and define $C_{\mathrm{rot}} = C_h N_{\mathrm{rot}}$. The rotating FERNN lifts the hidden state over angular velocity branches:

$$h_t \in \mathbb{R}^{B\times M_R\times C_{\mathrm{rot}}\times H\times W}.$$

The input frame is mapped into a rotation-lifted feature field $u_t = \Gamma_u(x_t)$, which is shared across angular branches. For each $\omega \in V_R$, the branch transport consists of a cyclic shift of the orientation channels and a spatial rotation by the one-step angle. Denote this combined action by T_ω:

$$\tilde{h}^{(\omega)}_{t-1} = T_\omega h^{(\omega)}_{t-1}.$$

The recurrent update is

$$h^{(\omega)}_t = \sigma\Big(u_t + \Gamma_h(\tilde{h}^{(\omega)}_{t-1})\Big), \qquad \omega \in V_R,$$

where Γ_u and Γ_h are rotation-equivariant convolutional maps. The angular branches are fused by max-pooling and decoded:

$$f_t = \max_{\omega\in V_R} h^{(\omega)}_t, \qquad \hat{x}_{t+1} = g_\theta(f_t).$$

Flow-SSM. Rotation Flow-SSM keeps the same angular branch structure and transport operator T_ω, but uses a selective state-space recurrence in the rotation-lifted feature space. Let D denote the base feature width and set $C_{\mathrm{rot}} = DN_{\mathrm{rot}}$. The encoded feature and recurrent state are

$$u_t \in \mathbb{R}^{B\times C_{\mathrm{rot}}\times H\times W}, \qquad s_t \in \mathbb{R}^{B\times M_R\times C_{\mathrm{rot}}\times H\times W}.$$

The model predicts branch-shared scalar gate fields Δ_t, b_t, c_t, where each scalar gate acts uniformly over the N_{rot} orientation channels within a feature block. With

$$A = -\exp(\log A_{\mathrm{real}}), \qquad \bar{A}_t = \exp(\Delta_t \odot A),$$

each branch is updated by

$$\tilde{s}^{(\omega)}_{t-1} = T_\omega s^{(\omega)}_{t-1},$$
$$s^{(\omega)}_t = \bar{A}_t \odot \mathrm{Mix}(\tilde{s}^{(\omega)}_{t-1}) + b_t \odot u_t.$$

The branch readout is

$$y^{(\omega)}_t = c_t \odot s^{(\omega)}_t + \alpha \odot u_t,$$

where α is a learned skip scale. Finally,

$$f_t = \max_{\omega\in V_R} y^{(\omega)}_t, \qquad \hat{x}_{t+1} = g_\theta(f_t).$$

Thus, compared with rotating FERNN, rotating Flow-SSM keeps the same branch-wise rotation transport, but replaces the convolutional recurrence by a selective state-space update.

For Rotating MNIST, we use

$$C = 1, \qquad H = W = 34, \qquad T_{\mathrm{in}} = 10, \qquad T_{\mathrm{pred}} \in \{10, 50\},$$
$$V_R = \{-40, -20, 0, 20, 40\}, \qquad M_R = 5.$$

For Rotating MNIST Flow-SSM, we set

$$D = 32, \qquad N_{\mathrm{rot}} = 8,$$

so that

$$C_{\mathrm{rot}} = 256.$$

Training and evaluation. All Moving and Rotating MNIST experiments are trained with mean squared error loss:

$$\mathcal{L} = \frac{1}{T_{\mathrm{pred}} HW} \sum_{t=1}^{T_{\mathrm{pred}}} \|\hat{x}_t - x_t\|_2^2.$$

All models are trained for 200 epochs with learning rate 10^{-4}. We report test MSE averaged over all predicted frames. Therefore, for $10 \to 10$, the metric is averaged over 10 predicted frames, while for $10 \to 50$, it is averaged over 50 predicted frames.

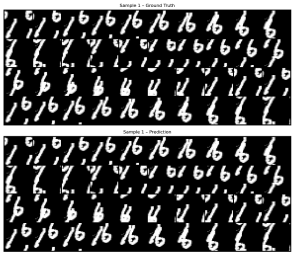

Figure 5: Qualitative long-horizon prediction on Moving MNIST under the directly trained $10 \rightarrow 50$ protocol. Selected frames from the 50 predicted future steps are shown.



Figure 6: Qualitative long-horizon prediction on Rotating MNIST under the directly trained $10 \rightarrow 50$ protocol. Selected frames from the 50 predicted future steps are shown.

More results. We additionally provide qualitative long-horizon predictions under the directly trained $10 \rightarrow 50$ protocol. The examples visualize selected frames from the 50-step rollout and compare the predicted sequence against the ground-truth trajectory. These visualizations complement the MSE results by showing whether the models preserve the prescribed translation or rotation structure over extended autoregressive prediction.

B.2 KTH motion recognition

Dataset and induced-motion construction. We use the person-based KTH[17, 9] split: subjects 1–16 for training, 17–20 for validation, and 21–25 for testing. Each video is converted to grayscale, resized to 32×32, normalized to $[0, 1]$, and sampled as a $T = 16$-frame clip with temporal stride 2.

For the induced-motion setting, each sampled clip is assigned a constant translation velocity

$$v = (v_y, v_x) \in V_i,$$

where V_i is one of the velocity sets defined in the main text. The entire frame is translated by the same accumulated displacement at each time step. Concretely, if x_t denotes the original frame at clip time t, the motion-augmented frame is

$$\tilde{x}_t(p_y, p_x) = x_t((p_y - tv_y) \bmod H,\ (p_x - tv_x) \bmod W)\,.$$

Thus, every pixel follows the same prescribed translation flow. The modulo operation implements circular spatial padding: pixels shifted beyond one image boundary re-enter from the opposite boundary, so the transformation is a global translation on a discrete torus rather than a crop with zero padding. This construction aligns the data augmentation with the circular branch transport used by the flow-aware recurrent models.

Implementation details. Let

$$X \in \mathbb{R}^{B \times T \times C \times H \times W}$$

denote an input clip. Let V_{model} be the velocity-branch set used by a flow-aware model, and let $M = |V_{\text{model}}|$.

Plain Mamba. Plain Mamba removes the velocity branch dimension and therefore does not explicitly transport memory across space. Each frame is first encoded by a small convolutional encoder,

$$u_t = \text{Enc}(x_t) \in \mathbb{R}^{B \times D \times H \times W}.$$

The model then applies a spatial selective state-space update with a single recurrent state

$$s_t \in \mathbb{R}^{B \times N \times H \times W}.$$

In contrast to the flow-aware models below, the state s_t remains at fixed spatial coordinates. After processing the full clip, the last-frame feature is globally average-pooled and fed to a linear classifier:

$$\hat{y} = \text{FC}(\text{GAP}(u_T)) \in \mathbb{R}^6.$$

This baseline tests the effect of Mamba-style selective recurrence without flow-aligned memory transport.

FERNN. The FERNN classifier lifts its recurrent hidden state over the velocity branch set:

$$h_t^\ell \in \mathbb{R}^{B\times M\times C_h\times H\times W},$$

where ℓ indexes recurrent layers and C_h is the hidden width. For the first layer, the input frame is encoded by a convolutional input map and copied to every velocity branch:

$$u_t^1(v) = \mathrm{Enc}(x_t), \qquad v \in V_{\mathrm{model}}.$$

For each branch $v = (v_y, v_x)$, the previous hidden state is transported by one step before the recurrent update:

$$\tilde{h}_{t-1}^{\ell,v}(p) = h_{t-1}^{\ell,v}(p - v).$$

In implementation, this transport is realized by fixed shift kernels with circular padding, so the branch state is shifted on the same discrete torus used by the induced-motion augmentation. The recurrent update is

$$h_t^{\ell,v} = \tanh\Big(u_t^\ell(v) + W_h^\ell * \tilde{h}_{t-1}^{\ell,v}\Big), \qquad v \in V_{\mathrm{model}},$$

where W_h^ℓ is shared across velocity branches. For deeper layers, $u_t^\ell(v)$ is obtained by applying a convolutional input map to the previous layer's branch-wise hidden output.

The FERNN recurrent block returns branch-indexed features for all time steps. For classification, we take the last time step, pool over velocity branches, and then apply global average pooling and a linear classifier:

$$z_T = \max_{v\in V_{\mathrm{model}}} h_T^v, \qquad \hat{y} = \mathrm{FC}(\mathrm{GAP}(z_T)) \in \mathbb{R}^6.$$

In our KTH experiments, the FERNN hidden width is $C_h = 128$.

Flow-SSM. Flow-SSM uses the same velocity-branch interpretation as FERNN, but replaces the convolutional recurrent update with a branch-shared selective state-space update. Each frame is encoded as

$$u_t = \mathrm{Enc}(x_t) \in \mathbb{R}^{B\times D\times H\times W}.$$

At layer ℓ, the lifted state is

$$s_t^\ell \in \mathbb{R}^{B\times M\times N\times H\times W}.$$

From the current feature u_t^ℓ, the model generates branch-shared selective fields

$$\Delta_t^\ell, \quad B_t^\ell, \quad C_t^\ell,$$

which are shared across all velocity branches. The diagonal continuous-time state matrix is parameterized as

$$A = -\exp(\log A_{\mathrm{real}}), \qquad \bar{A}_t^\ell = \exp(\Delta_t^\ell \odot A).$$

For each branch v, the previous state is first transported:

$$\tilde{s}_{t-1}^{\ell,v}(p) = s_{t-1}^{\ell,v}(p - v),$$

then mixed spatially and updated by

$$s_t^{\ell,v} = \bar{A}_t^\ell \odot \mathrm{Mix}(\tilde{s}_{t-1}^{\ell,v}) + \Delta_t^\ell \odot (B_t^\ell u_t^\ell).$$

The branch-wise readout is

$$y_t^{\ell,v} = C_t^\ell s_t^{\ell,v} + \mathrm{Skip}(u_t^\ell).$$

Between Flow-SSM layers, the velocity branches are fused by max-pooling:

$$u_t^{\ell+1} = \max_{v\in V_{\mathrm{model}}} y_t^{\ell,v}.$$

For classification, we use the final time step, pool over velocity branches, and apply global average pooling followed by a linear classifier:

$$z_T = \max_{v\in V_{\mathrm{model}}} y_T^v, \qquad \hat{y} = \mathrm{FC}(\mathrm{GAP}(z_T)) \in \mathbb{R}^6.$$

In our KTH experiments, Flow-SSM uses $D = 40$, $N = 6$, and three recurrent layers. The selective fields are shared across velocity branches, so increasing the number of velocity branches mainly increases activation memory and computation rather than duplicating the model parameters.

Training and evaluation. All six KTH runs reported in the main text use a resampled induced-motion protocol. During training, the induced velocity of a clip is sampled when the clip is accessed, rather than fixed once for each underlying KTH subsequence. This removes deterministic clip–velocity associations and gives a stricter evaluation of motion robustness. All compared models are trained under the same protocol.

For evaluation, we use `eval_repeat` $= 4$, corresponding to four temporal windows per valid action segment. The evaluation velocity set is determined by the reported setting, e.g., $V_1 \to V_2$ evaluates clips with induced velocities from V_2. Models are trained with cross-entropy loss for six-way action classification:

$$\mathcal{L}_{\text{cls}} = -\log p_\theta(y \mid X).$$

We report test accuracy from the checkpoint selected by validation accuracy.

Branch-level analysis. For FERNN and Flow-SSM, the final recurrent representation is indexed by velocity branches. Let

$$Y_i(v) \in \mathbb{R}^{D\times H\times W}$$

denote the final branch feature of sample i for velocity branch v. We measure the importance of a branch by its average activation magnitude:

$$E_i(v) = \text{mean}_{d,h,w} \left|Y_i(v,d,h,w)\right|.$$

Let $v_i^\star$ be the induced velocity used to generate sample i. If $v_i^\star$ is represented in the model branch set, we use the corresponding branch as the ground-truth branch; otherwise, it is mapped to the nearest branch in the model velocity lattice. In the reported V_2-branch experiments, the test velocities are represented by the model lattice.

Hit@1 measures whether the ground-truth branch has the largest energy:

$$\text{Hit@1} = \frac{1}{N_{\text{test}}} \sum_i \mathbf{1}\left[v_i^\star = \arg\max_v E_i(v)\right].$$

Top-5 measures whether the ground-truth branch appears among the five highest-energy branches:

$$\text{Top5} = \frac{1}{N_{\text{test}}} \sum_i \mathbf{1}\left[v_i^\star \in \text{Top5}_v\, E_i(v)\right].$$

For Drop-GT Acc., we mask the ground-truth velocity branch before the final branch pooling and recompute classification accuracy:

$$\hat{y}_i^{\text{drop}} = \Phi(X_i;\, Y_i(v_i^\star) = 0).$$

This diagnostic tests whether the model stores action-relevant information only in the flow-consistent branch or distributes it across nearby and alternative moving reference frames. A higher Drop-GT accuracy indicates stronger robustness after removing the most directly aligned branch.

B.3 UCF101 Natural-Video Prediction

Dataset and task. We use UCF101 to evaluate video prediction on natural motion sequences. We adopt the official split 1, reserving 10% of the training videos for validation. Each video is sampled into 12 RGB frames with temporal stride 2, resized and cropped to 64×64. Models observe 8 frames and autoregressively predict the next 4 frames.

Implementation details. We compare Mamba, FERNN, and Flow-SSM using a shared convolutional encoder-decoder architecture. Flow-SSM uses feature dimension $D = 64$, state dimension $N = 16$, and a translation velocity grid

$$V = \{-1, 0, 1\} \times \{-1, 0, 1\}.$$

Each velocity branch transports its recurrent state in latent space, with selective parameters shared across branches. Branch outputs are aggregated by max-pooling.

Training and evaluation. All models are trained using AdamW with learning rate 3×10^{-4} and batch size 32. We minimize the mean L1 prediction loss without teacher forcing and select checkpoints by validation MSE. We report PSNR, SSIM, and final-frame PSNR. To examine performance under

Table 6: UCF101 video prediction results. Higher is better for all metrics.

Model	PSNR ↑	SSIM ↑	Last PSNR ↑	High-motion PSNR ↑	High-motion SSIM ↑	High-motion Last PSNR ↑
Mamba	19.874	0.679	18.989	17.416	0.554	16.407
FERNN	19.738	0.673	18.755	17.278	0.556	16.272
Flow-SSM	**21.064**	**0.717**	**20.651**	**19.106**	**0.644**	**17.945**

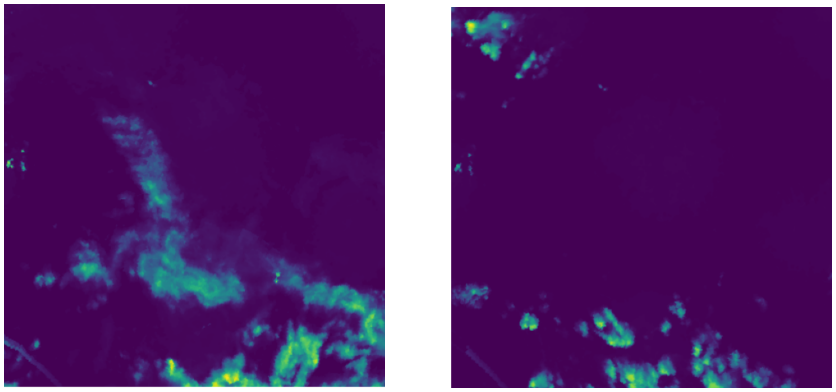

Figure 7: Examples of SEVIR VIL radar observations. Brighter regions indicate stronger convective intensity.

stronger motion, we additionally evaluate the highest-motion third of test clips, ranked by mean absolute grayscale frame differences in the observed context.

Results. Table 6 summarizes the prediction results. Flow-SSM consistently outperforms both baselines across all metrics. The improvement is particularly pronounced on high-motion sequences, supporting the benefit of flow-aligned recurrence for natural video prediction.

B.4 SEVIR forecast

Dataset and forecasting task. We use the VIL modality of SEVIR for radar nowcasting. SEVIR is a storm-event imagery dataset constructed from spatially and temporally aligned meteorological observations, including GOES-16 satellite channels, NEXRAD radar-derived VIL mosaics, and lightning measurements [19]. Each SEVIR event covers a $384\,\mathrm{km} \times 384\,\mathrm{km}$ region over a 4-hour time window. In this work, we focus only on the VIL sequence, since it directly represents the radar-observed convective storm structure used for precipitation nowcasting. VIL, or vertically integrated liquid, is a radar-derived estimate of liquid water content obtained from the vertical profile of radar reflectivity [38]. Intuitively, larger VIL values indicate stronger vertically integrated precipitation or convective intensity. We represent each VIL event as a single-channel radar sequence

$$Z \in \mathbb{R}^{T_{\mathrm{all}} \times 1 \times H \times W}.$$

From each event, we construct an input–target pair

$$X = Z_{1:T_{\mathrm{in}}} \in \mathbb{R}^{T_{\mathrm{in}} \times 1 \times H \times W}, \qquad Y = Z_{T_{\mathrm{in}}+1:T_{\mathrm{in}}+T_{\mathrm{pred}}} \in \mathbb{R}^{T_{\mathrm{pred}} \times 1 \times H \times W}.$$

We write $a \to b$ to denote the protocol with $T_{\mathrm{in}} = a$ observed frames and $T_{\mathrm{pred}} = b$ predicted frames. The main training setting is $8 \to 8$, and we additionally evaluate longer forecasting behavior under $12 \to 12$.

We use year-based train/validation/test splits built from the indexed SEVIR events. During training, VIL values are normalized using the `zero_one_255` convention,

$$Z_{\mathrm{norm}} = Z/255.$$

For threshold-based evaluation, predictions are mapped back to the original VIL scale. We report the critical success index (CSI) at thresholds 74 and 133. For a threshold τ, pixels are classified as active if their VIL value is at least τ. Let TP_τ, FP_τ, and FN_τ denote the number of hit, false-alarm, and missed active pixels. Then

$$\mathrm{CSI}_\tau = \frac{\mathrm{TP}_\tau}{\mathrm{TP}_\tau + \mathrm{FP}_\tau + \mathrm{FN}_\tau}.$$

We use CSI_{74} to measure prediction quality for moderate-to-strong radar echoes and CSI_{133} to emphasize stronger convective regions.

Implementation details. All SEVIR models use the same forecasting interface. Given an input VIL sequence X, a pyramid encoder maps it into a three-level latent sequence

$$X \xrightarrow{\text{PyramidEncoder}} \{z^0_{1:T_{\text{in}}}, z^1_{1:T_{\text{in}}}, z^2_{1:T_{\text{in}}}\},$$

where $\ell = 0$ denotes the finest latent level and $\ell = 1, 2$ denote progressively coarser levels. A temporal core scans the context latents and rolls out a predicted latent pyramid,

$$\{z^\ell_{1:T_{\text{in}}}\}^2_{\ell=0} \xrightarrow{\text{TemporalCore}} \{\hat{z}^\ell_{1:T_{\text{pred}}}\}^2_{\ell=0},$$

and a pyramid decoder maps the predicted latents back to VIL frames:

$$\{\hat{z}^\ell_{1:T_{\text{pred}}}\}^2_{\ell=0} \xrightarrow{\text{PyramidDecoder}} \hat{Y}.$$

Thus, all four models operate on the same three-level latent pyramid; they differ only in the temporal core.

Table 7: SEVIR temporal-core comparison. All models use the same encoder–decoder pyramid.

Model	Three-level latent pyramid	Flow branches	Cross-level communication
Mamba	Yes	No	No
FERNN	Yes	Yes	No
Flow-SSM	Yes	Yes	No
Hier Flow-SSM	Yes	Yes	Yes

Mamba. The Mamba baseline uses a non-flow temporal core. At each latent level ℓ, it maintains a recurrent state at fixed spatial coordinates,

$$s^\ell_t(p),$$

and updates it using input-dependent selective state-space dynamics. Since no velocity branch is introduced, the recurrent memory is not explicitly transported along candidate storm motions.

FERNN. FERNN replaces the fixed-coordinate recurrence with a flow-indexed recurrent state. At each pyramid level, it maintains

$$h^\ell_t(v, p), \qquad v \in V,$$

where V is the discrete translation-velocity branch set. Before the recurrent update, each branch is transported according to its velocity:

$$\tilde{h}^{\ell,v}_{t-1}(p) = h^{\ell,v}_{t-1}(p - v).$$

The transported state and the current latent input are then updated by a shared convolutional recurrent cell:

$$h^{\ell,v}_t = \text{ConvRNN}^\ell\left(u^\ell_t, \tilde{h}^{\ell,v}_{t-1}\right), \qquad v \in V.$$

The branch outputs are pooled over v to produce a levelwise latent prediction. FERNN therefore tests the effect of explicit flow-aligned recurrence without Mamba-style selective state-space dynamics.

Flow-SSM. Flow-SSM keeps the velocity-indexed state but replaces the convolutional recurrent update with a selective state-space update. At each pyramid level,

$$s^\ell_t(v, p), \qquad v \in V,$$

is maintained independently. For each branch, the previous state is transported first:

$$\tilde{s}^{\ell,v}_{t-1}(p) = s^{\ell,v}_{t-1}(p - v).$$

Branch-shared selective fields generated from the current latent feature u^ℓ_t update all branches:

$$s^{\ell,v}_t = \bar{A}^\ell_t \odot \text{Mix}^\ell(\tilde{s}^{\ell,v}_{t-1}) + \bar{B}^\ell_t u^\ell_t.$$

A readout maps the branch states to branch outputs, and max-pooling over v gives the levelwise latent output. In contrast to Hier Flow-SSM, the three pyramid levels are updated independently and do not exchange messages inside the temporal core.

Hier Flow-SSM. Hier Flow-SSM augments the flow-aligned temporal core with explicit communication across pyramid levels. At each active level, it maintains a flow-indexed state

$$s_t^\ell(v,p), \qquad \ell \in \{0,1,2\}, \quad v \in V_\ell,$$

where V_ℓ denotes the velocity lattice used at level ℓ. The levelwise transported state is

$$\tilde{s}_{t-1}^{\ell,v}(p) = s_{t-1}^{\ell,v}(p - \phi_\ell(v)),$$

where $\phi_\ell(v)$ is the one-step translation induced by branch v at level ℓ. In addition to the local Flow-SSM update, Hier Flow-SSM receives cross-level messages from connected source levels:

$$\eta_t^\ell(v,p) = \sum_{k:\, k\to\ell} K_{k\to\ell}\big(s_t^k\big)\,(v,p),$$

where $K_{k\to\ell}$ denotes the scale communicator from level k to level ℓ. The hierarchical state update is then

$$s_t^{\ell,v} = \bar{A}_t^\ell \odot \mathrm{Mix}^\ell(\tilde{s}_{t-1}^{\ell,v}) + \bar{B}_t^\ell u_t^\ell + \eta_t^\ell(v,p).$$

The corresponding branch readout is

$$y_t^{\ell,v} = C_t^\ell s_t^{\ell,v} + D_t^\ell u_t^\ell.$$

Branch pooling gives a levelwise prediction,

$$\hat{z}_t^\ell = \max_{v\in V_\ell} y_t^{\ell,v}.$$

The predicted pyramid

$$\{\hat{z}_t^0, \hat{z}_t^1, \hat{z}_t^2\}$$

is passed to the decoder to reconstruct the VIL forecast.

This architecture makes hierarchy explicit: coarse levels summarize broad storm displacement and organization, while fine levels preserve local precipitation structure. The cross-level message term η_t^ℓ lets coarse storm-scale information correct fine-resolution predictions and lets fine-level evidence inform higher-level latent evolution. This is the main architectural difference between Flow-SSM and Hier Flow-SSM on SEVIR.

Training and evaluation. All SEVIR models are trained under the same forecasting protocol. The main training setting is $8 \to 8$: the model observes 8 past VIL frames and predicts the next 8 frames. We train each model for 30 epochs using AdamW with learning rate 10^{-4}. The training objective is the mean absolute error on the predicted VIL sequence:

$$\mathcal{L}_{\mathrm{train}} = \frac{1}{T_{\mathrm{pred}} HW} \sum_{t=1}^{T_{\mathrm{pred}}} \|\hat{Y}_t - Y_t\|_1.$$

The best checkpoint is selected by validation loss.

At evaluation time, we report the matched $8 \to 8$ setting and an additional longer $12 \to 12$ setting. The notation $12 \to 12$ means that the model scans 12 observed context frames and rolls out 12 future frames using the same autoregressive forecasting interface. This evaluates whether the learned temporal core remains stable under a longer context and prediction horizon.

We report threshold-based nowcasting metrics on the original VIL scale. Since models are trained with normalized VIL values,

$$Z_{\mathrm{norm}} = Z/255,$$

the predictions are mapped back to raw VIL units before computing threshold scores. Following the main text, we report CSI_{74} and CSI_{133}, where CSI_{133} emphasizes stronger convective regions. All models use the same train/validation/test splits, normalization, checkpoint selection rule, and evaluation pipeline.

Table 8: Additional SEVIR reconstruction results. We report raw-scale VIL MSE; lower is better.

Model	$8 \to 8$	$10 \to 10$	$12 \to 12$
Mamba	241.88	323.40	389.15
FERNN	203.23	283.35	344.65
Flow-SSM	213.83	294.88	355.88
Hier Flow-SSM	209.82	290.05	350.50

Additional reconstruction results and hierarchy probes. In addition to threshold-based CSI scores in the main text, we report raw-scale VIL MSE in Table 8. MSE complements CSI by measuring global reconstruction error over all pixels, while CSI focuses on thresholded precipitation regions.

We further probe the hierarchy module in Hier Flow-SSM. Let $\hat{z}_0^\ell$ denote the predicted latent at scale ℓ before applying the hierarchical correction, and let Δ_e denote the additive correction from a hierarchy path e. The hierarchy-corrected latent can be written as

$$\hat{z}_{\text{hier}}^\ell = \hat{z}_0^\ell + \sum_{e:\,\text{dst}(e)=\ell} \Delta_e.$$

We use three diagnostics. First, counterfactual removal decodes the prediction after removing all hierarchy paths. We measure the output change by

$$\text{OutputDelta} = \frac{\|\hat{Y}_{\text{normal}} - \hat{Y}_{\text{zero}}\|_2}{\|\hat{Y}_{\text{normal}}\|_2}.$$

Second, target alignment measures whether a correction points toward the target latent residual:

$$\text{Align}(e) = \cos\Big(\Delta_e,\ z_{\text{gt}}^{\ell(e)} - \hat{z}_0^{\ell(e)}\Big), \qquad \ell(e) = \text{dst}(e).$$

Third, spatial localization measures whether correction magnitude concentrates near current prediction-error regions:

$$\text{SpatialCorr}(e) = \text{corr}\Big(|\Delta_e(p)|,\ |\hat{Y}(p) - Y(p)|\Big).$$

Table 9: Hierarchy probe diagnostics for Hier Flow-SSM. The probes test whether hierarchical corrections affect the output, align with target residuals, and localize around error-prone regions. All metrics are evaluated over mixed-length protocols ($5 \to 5, 8 \to 8, 10 \to 10, 12 \to 12$), and the table reports the weighted aggregate results.

Probe	Metric	Value
Remove hierarchy paths	OutputDelta; MSE; CSI_{133}	0.106; 177.4→181.8; 0.301→0.277
Coarsest self correction	Target alignment	0.083
Total fine-scale correction	Spatial corr. with error	0.390
Cross-scale path $2 \to 1$	Spatial corr. with error	0.317

These probes show that the hierarchy is not only a passive architectural addition. Removing all hierarchical paths changes the decoded forecast and degrades both raw MSE and strong-echo CSI, while the retained corrections show positive target alignment and spatial concentration near prediction-error regions.

B.5 Computational Resources

All experiments reported in this paper were conducted on a shared public cluster equipped with NVIDIA H100 GPUs, each with 80GB of GPU memory. All reported models and training protocols fit on a single H100 GPU.

For Moving and Rotating MNIST, the $10 \to 10$ protocols were trained in approximately 7 hours, while the longer $10 \to 50$ protocols required approximately 12 hours due to the larger memory and

rollout cost. For KTH action recognition, training was substantially lighter: each model was trained in approximately 1 hour, and all six reported runs can be completed in about 7 hours. For SEVIR VIL nowcasting, all non-hierarchical models were trained in roughly 10 hours. Hierarchical Flow-Mamba required approximately 17 hours under a similar parameter budget, mainly due to the additional cross-scale communication operations.

The current implementation uses standard PyTorch operations for the hierarchical communication and scan updates. We expect that a fused implementation with custom CUDA kernels, especially for the scan and cross-scale communication operations, would substantially improve both training and inference efficiency.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are clearly stated in the abstract and Sec. 1, including flow-equivariant SSM and hierarchical multi-scale modeling.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses limitations, including modeling assumptions and computational scaling, in Sec. 4.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All assumptions are stated in Sec. 2, and full proofs are provided in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Comprehensive experimental results across multiple datasets are reported in Sec. 3 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We use standard datasets (MNIST, KTH recognition, SEVIR). The code repository of the paper can be accessed at the URL in the abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper specifies the experimental setup, including datasets, evaluation protocols, and metrics in Sec. 3, with additional training and implementation details provided in the appendix B

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report multi-seed results for the SEVIR experiments, including the mean and standard deviation across independent runs, to quantify experimental variability.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resource details are provided in the appendix B

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics and does not involve sensitive data, human subjects, or ethical risks.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both potential positive applications and risks within the limitations discussion in Sec. 4.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not involve the release of models or datasets that pose a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets used in this work, including datasets and prior methods, are properly credited through citations, and used in accordance with their respective licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release new datasets, models, or code assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or experiments with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects or crowdsourcing, and therefore IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as part of the core methodology or model design in this work.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.