Flow Equivariant Transformers

Ibrahim Khaliliya [1] **T. Anderson Keller** [1 2]

Abstract

Many sequences exhibit a geometric 'flow' structure that both relates time steps within the sequence via a time-parameterized Lie-group action, and relates sequences to one another via the Lie algebra of the underlying flow. In this work, we illustrate how Transformers do not inherently respect this type of flow structure, causing otherwise related sequences to be processed as unrelated inputs, i.e., the same sequence viewed from different moving reference frames may be seen entirely differently. We then introduce a provably Flow Equivariant Transformer and demonstrate empirically that it achieves the generalization benefits expected of geometric inductive biases.

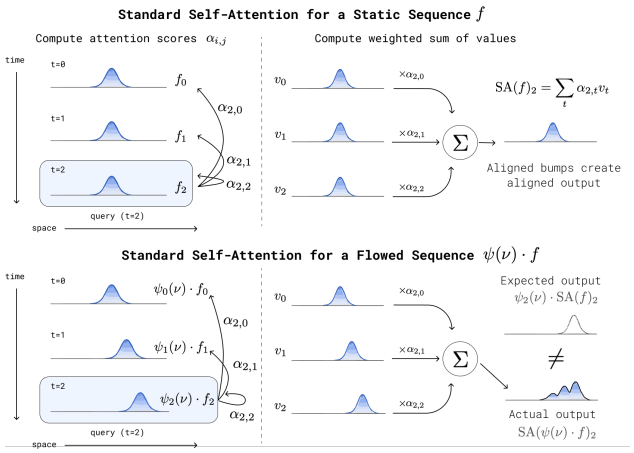

Figure 1. **Standard self-attention is not flow equivariant.** The output for a flowed input sequence is not equivalent to flowing the output of a static sequence, i.e. $\mathrm{SA}(\psi(\nu) \cdot f)_t \neq \psi(\nu)_t \cdot \mathrm{SA}(f)_t$.

1. Introduction

Many sequence-modeling problems exhibit a *flow* structure: a one-parameter Lie group $\psi_t(\nu)$ acting on signals over time, such that two sequences related by such a family describe the same underlying process viewed from different time-parametrized reference frames. The flow may be a translation (an object moving with constant velocity across a video frame), but equally a continuous transposition of a melodic motif up in pitch, a slow time-dependent rotation acting on a high-dimensional state, or any other smooth time-parameterized symmetry. Crucially, the underlying signal being flowed need not itself be static, but may be a dynamic process, such as a swimming fish, a talking face, or a time-dependent melody that can equivalently be transported by $\psi_t(\nu)$. In all cases, the observed sequence is the composition of the signal's intrinsic dynamics with the flow. Within a single example, points along the sequence length are related by the action of $\psi_t(\nu)$; across different examples, sequences that share the same intrinsic signal but differ in their flow are related by elements ν of the underlying Lie algebra $\mathfrak{g}$.

Deep neural network architectures that hard-code a relevant symmetry are often denoted 'equivariant', and have been shown to generalize from less data and even scale more favorably than comparable unconstrained counterparts (Cohen & Welling, 2016; Kondor & Trivedi, 2018; Cohen et al., 2019; Bronstein et al., 2021; Ngo & Ravanbakhsh, 2026). Recently, Keller (2025) introduced *flow equivariance* as the natural extension of equivariant deep learning to such time-parameterized symmetries; and demonstrated that unconstrained temporal integration, such as standard recurrence, generally breaks flow equivariance.

In state-of-the-art sequence modeling, however, Transformers have displaced recurrent networks as the dominant sequence architecture. In this work, we demonstrate that Transformers are not naturally flow equivariant, just as RNNs are not, but can be made so, like the Flow Equivariant RNN (FERNN), by making all operations group-equivariant and operating on a flow-lifted hidden state. Our work is thus related to equivariant attention mechanisms and equivariant Transformers, including Lie-group self-attention and SE(3)/E(3)-equivariant Transformer architectures (Hutchinson et al., 2021; Fuchs et al., 2020; Liao & Smidt, 2023), but specific to time-parameterized symmetries. Concretely, we introduce the *Flow Equivariant Transformer* (FET), prove its flow equivariance analytically, and validate this theory with a simple 1-D toy task that demonstrates the significant

[1]Harvard University [2]The Kempner Institute for the Study of Natural and Artificial Intelligence, Harvard University. Correspondence to: T. Anderson Keller <t.anderson.keller@gmail.com>.

Workshop on Weight-Space Symmetries, held in conjunction with the 43rd International Conference on Machine Learning, Seoul, South Korea. 2026. Copyright 2026 by the author(s).

generalization benefits afforded by this construction.

2. Background

In the following, we introduce flow equivariance in the general setting. Our constructions in this paper apply to any group G for which equivariant building blocks, such as G-CNNs (Cohen & Welling, 2016), are available.

Space-time signals and group actions. We consider discrete-time space-time signals $f \in \mathcal{F}_K(X, \mathbb{Z})$, where $\mathcal{F}_K(X, \mathbb{Z}) := \{f : X \times \mathbb{Z} \to \mathbb{R}^K\}$ denotes the space of K-channel signals defined over a 'spatial' domain X and discrete time $t \in \mathbb{Z}$. We write $f_t(x) := f(x, t)$ for the signal at time t, and in practice restrict to finite sequences with $t \in \{0, \ldots, T-1\}$. We assume that the signal domain X admits the action of a symmetry group G. For $g \in G$, the corresponding left action on a single time-slice is written

$$(g \cdot f_t)(x) := f_t(g^{-1} \cdot x), \qquad g \in G,\ x \in X. \quad (1)$$

This follows standard convention: transforming the signal by g is equivalent to evaluating the original signal at the inverse-transformed coordinate.

Group convolution. A standard way to construct maps equivariant to the static group action is through group convolution. For a kernel $W^i : G \to \mathbb{R}^K$, group convolutions between functions on G are defined by

$$[f \star_G W^i](g) := \sum_{h \in G} \sum_{k=1}^{K} f_k(h)\, W^i_k(g^{-1}h). \quad (2)$$

We state the group-convolution formula for functions on G; typically, the initial embedding *lifts* a signal on X to a function on G, i.e. it maps $\mathcal{F}_K(X) \to \mathcal{F}(G)$, after which all further convolutions are of the $G \to G$ form above. Importantly, these convolutions commute with the action of G: $(g \cdot f) \star_G W = g \cdot (f \star_G W)$. We will thus make use of these in the following to construct the flow equivariant transformer block; however, we highlight that this is just one possible construction of a group equivariant layer, and other, e.g., steerable, group representations could equivalently be used (Cohen & Welling, 2017; Weiler & Cesa, 2019).

Flows. Following Keller (2025), we view motion as a *flow* over space X: a one-parameter subgroup of G generated by an element ν of the corresponding Lie algebra $\mathfrak{g}$. We work in the matrix Lie group setting of Hall (2015), for which the exponential map $\exp : \mathfrak{g} \to G$ is well defined. We denote the time-t group element generated by ν as $\psi_t(\nu) := \exp(t\nu) \in G$. By construction, $\psi_0(\nu) = e$ and $\psi_s(\nu)\psi_t(\nu) = \psi_{s+t}(\nu)$. For a sequence $f \in \mathcal{F}_K(X, \mathbb{Z})$, the action of the full flow $\psi(\nu)$ is defined frame-wise by

$$\big(\psi(\nu) \cdot f\big)_t(x) := f_t\big(\psi_t(\nu)^{-1} \cdot x\big)\,. \quad (3)$$

Throughout, we use $\psi(\nu) \cdot f$ to denote the action of the entire time-parameterized transformation on the sequence, while $\psi_t(\nu)$ denotes the instantaneous group element acting at time t. In all our experiments G is the compact, abelian group of circular translations $\mathbb{Z}/S\mathbb{Z}$ (a discretization of the translation group $\cong \mathbb{R}$), the generator $\nu \in \mathbb{Z}$ is an integer velocity, $\mathfrak{g} \cong \mathbb{R}$, and $\psi_t(\nu) = \exp(t\nu)$ acts as the circular shift by $t\nu \bmod S$. In practice, we consider a finite set of flow generators $\mathcal{V}_{\text{model}} \subset \mathfrak{g}$, and build the model to be equivariant with respect to flows generated by elements of this set. We write $\mathcal{V}_{\text{train}}$ and $\mathcal{V}_{\text{test}}$ for the sets of generators used when constructing the training and evaluation sequences.

Flow equivariance. A sequence model $\Phi : \mathcal{F}_K(X, \mathbb{Z}) \to \mathcal{F}_{K'}(Y, \mathbb{Z})$, where, as for X, the output domain Y also admits an action of G, is *flow-equivariant* with respect to $\mathcal{V} \subseteq \mathfrak{g}$ if

$$\psi(\nu) \cdot \Phi[f] = \Phi[\psi(\nu) \cdot f], \quad \forall \nu \in \mathcal{V},\ f \in \mathcal{F}_K(X, \mathbb{Z}). \quad (4)$$

In words, applying a flow to the input sequence and then processing it with the model must produce the same result as first processing the original sequence and then applying the corresponding flow to the output sequence; Figure 1 shows that standard self-attention is not generally flow equivariant.

3. The Flow Equivariant Transformer

For the purpose of analysis, consider the Transformer (Vaswani et al., 2017) as a map between discrete-time sequences of spatially-continuous functions $\Phi : \mathcal{F}_K(X, \mathbb{Z}) \to \mathcal{F}_{K'}(Y, \mathbb{Z})$. This map is composed of an embedding ϕ, followed by multiple transformer 'blocks' each composed of self-attention, layer normalization, a multilayered perceptron (MLP), and residual connections. In the following, we will describe how to modify each of these components from their original formulation in order to make the full map Φ flow equivariant, satisfying Equation 4.

3.1. Flow-Lifting Embedding Layer

Similar to Keller (2025), the key step is to let the hidden state of the model live in a set of moving reference frames. Let $\phi : \mathcal{F}_K(X) \to \mathcal{F}_C(Y)$ be a G-equivariant embedding (e.g. Eqn. 2), where Y is the feature domain on which G acts. Denoting the embedded sequence as $z_t = \phi(f_t)$, we then lift each sequence element to the set of flows $\mathcal{V}_{\text{model}}$ by applying the inverse of each $\nu \in \mathcal{V}_{\text{model}}$ to each time step:

$$(Lz)_t(\nu) := \psi_t(\nu)^{-1} \cdot z_t, \quad \forall \nu \in \mathcal{V}_{\text{model}}. \quad (5)$$

Thus slice ν of the lifted state is the embedded sequence viewed from the reference frame moving according to $\psi(\nu)$. The lift L thus converts input flows into shifts of the flow index, which we denote $\nu \ominus \hat{\nu}$. For commuting translation

or rotation flows this is simply $\nu - \hat{\nu}$. With this notation,

$$\begin{aligned} L(\psi(\hat{\nu}) \cdot z)_t(\nu) &= \psi_t(\nu)^{-1} \cdot (\psi_t(\hat{\nu}) \cdot z_t) \\ &= (\psi_t(\hat{\nu})^{-1}\psi_t(\nu))^{-1} \cdot z_t = (Lz)_t(\nu \ominus \hat{\nu}). \end{aligned} \tag{6}$$

We write $(\rho_{\hat{\nu}} h)_t(\nu) := h_t(\nu \ominus \hat{\nu})$ for this flow-index shift. Exact equivariance requires the modeled set of generators to be closed under the relative-flow operation. In finite implementations, we store only $\mathcal{V}_{\text{model}}$, so equivariance is exact when $\rho_{\hat{\nu}}$ permutes the stored slices, and otherwise holds away from the boundaries, similar to (Keller, 2025).

3.2. Flow-Equivariant Transformer Block

After the lift, the model state has an additional set of $\mathcal{V}_{\text{model}}$ 'channels', i.e. $h \in \mathcal{F}_C(\mathcal{V}_{\text{model}} \times Y, \mathbb{Z})$. We write $h^0 := Lz$ for this lifted input and h^l for the state after l blocks, so the superscript indexes block depth. The Flow Equivariant Transformer (FET) block we introduce below is a standard pre-norm transformer block applied independently in each moving reference frame. All parameters are shared across ν, and no operation mixes different ν-slices. Thus the block is a bank of identical transformers, one for each flow, tied together by weight sharing over the flow dimension. The flow-mixing variant in Section 3.5 relaxes this no-mixing restriction while preserving the same flow-index equivariance. Concretely, the block B_l is defined as:

$$\bar{h}^l = h^l + \text{MSA}_G\big(N_G(h^l)\big)\,, \tag{7}$$

$$h^{l+1} = \bar{h}^l + \text{MLP}_G\big(N_G(\bar{h}^l)\big)\,. \tag{8}$$

Here N_G is any normalization that commutes with the action of G, MSA_G is multi-head self-attention whose projections are G-equivariant, and MLP_G is the usual position-wise feed-forward map with equivariant linear layers. In our experiments, we instantiate N_G as channel-only LayerNorm, reducing over the channel axis only (pointwise in (t, ν, y)) with a per-channel affine transform independent of y; this commutes with the G-action (Appendix A, Lemma A.1).

Writing $\eta_\nu = N_G(h(\nu))$ for the full time sequence in slice ν, a single attention head, queries, keys, and values are computed frame-wise by equivariant layers,

$$Q_\nu = \eta_\nu \star_G W_Q, \quad K_\nu = \eta_\nu \star_G W_K, \quad V_\nu = \eta_\nu \star_G W_V.$$

Self-attention then proceeds exactly as in a standard transformer along the time axis, per ν slice independently:

$$\text{SA}_G(\eta)_\nu = \left[\text{softmax}\left(\frac{Q_\nu K_\nu^\top}{\sqrt{d}}\right) V_\nu\right] \star_G W_O, \tag{9}$$

Multi-head attention repeats this construction for several heads and applies the final equivariant output projection after concatenation. Any temporal positional information is shared across ν and is independent of token 'space' y.

The feed-forward sublayer is defined in the same way. It is applied independently to each time and flow slice, with the ordinary linear maps replaced by equivariant layers:

$$\text{MLP}_G(\eta)_{t,\nu} = [\sigma(\eta_{t,\nu} \star_G W_1 + b_1)] \star_G W_2 + b_2. \tag{10}$$

3.3. Output Head

The FET core produces a lifted sequence h^N. The output head (R) is task-dependent and should implement the desired representation of the flow action on the output space. For invariant sequence-level prediction (as in our experiments), a simple choice is to pool over the feature domain, time, and flow index before applying a final prediction map:

$$\hat{y} = r\big(\text{Pool}_{t,\nu,y}\, N_G(h^N_{t,\nu})(y)\big)\,. \tag{11}$$

Complexity and memory. The lift replaces the hidden state with a stack of $V = |\mathcal{V}_{\text{model}}|$ slices; for a d-dimensional flow truncated at $|\nu_i| \leq K$ this is $V = (2K{+}1)^d$. Per layer, both compute and activation memory scale *linearly* in V (the lifted state is $B \times T \times V \times C \times |Y|$), while the parameter count is independent of V because all slices share weights. Self-attention is the usual $O(T^2)$ per slice, hence $O(VT^2)$ in total. The lift thus trades a factor V in memory for exact equivariance.

3.4. Equivariance

Let $C = B_N \circ \cdots \circ B_1$ be the transformer core composed of blocks B_l. Since ϕ is frame-wise G-equivariant, Equation 6 gives $L\phi(\psi(\hat{\nu}) \cdot f) = \rho_{\hat{\nu}} L\phi(f)$. To demonstrate each block commutes with $\rho_{\hat{\nu}}$, consider attention first. Shared per-slice group-equivariant projections give

$$Q(\rho_{\hat{\nu}}\eta)_\nu = (\rho_{\hat{\nu}}\eta)_\nu \star_G W_Q = \eta_{\nu\ominus\hat{\nu}} \star_G W_Q = Q(\eta)_{\nu\ominus\hat{\nu}},$$

and likewise for K and V. Thus the attention scores, softmax, value mixing, and output projection are simply reindexed, so $\text{MSA}_G\rho_{\hat{\nu}} = \rho_{\hat{\nu}}\text{MSA}_G$. The same per-slice weight sharing gives $N_G\rho_{\hat{\nu}} = \rho_{\hat{\nu}} N_G$ and $\text{MLP}_G\rho_{\hat{\nu}} = \rho_{\hat{\nu}}\text{MLP}_G$. Residual additions commute because $\rho_{\hat{\nu}}$ is linear. Hence $B\rho_{\hat{\nu}} = \rho_{\hat{\nu}} B$, and therefore $C\rho_{\hat{\nu}} = \rho_{\hat{\nu}} C$.

If the output head R realizes the desired output action, $R\rho_{\hat{\nu}} = \chi_{\hat{\nu}} R$, then the full model $\Phi = RCL\phi$ satisfies

$$\begin{aligned} \Phi(\psi(\hat{\nu}) \cdot f) &= RCL\phi(\psi(\hat{\nu}) \cdot f) = RC\rho_{\hat{\nu}} L\phi(f) \\ &= R\rho_{\hat{\nu}} CL\phi(f) = \chi_{\hat{\nu}}\Phi(f). \end{aligned} \tag{12}$$

For the invariant head in Equation 11, $\chi_{\hat{\nu}} = I$. As in finite group lifts, exact equivariance requires the induced shift to stay within $\mathcal{V}_{\text{model}}$; otherwise boundary slices are truncated.

3.5. Flow-Mixing Construction

While the above 'flow-independent' construction is sufficient for the following experiments, there exists an alternative more general 'flow-mixing' construction that allows interaction between flow slices. Similar to the FERNN, this construction is built by replacing all group convolution layers with *flow-convolution* layers, e.g. group convolutions

that also convolve over the $\mathcal{V}$ dimension, denoted $\star_{\mathcal{V}\times G}$ (see (Keller, 2025), Equation 9). In addition, the self-attention layer can then be made to attend *across ν slices*, with the following attention weight between time step t in flow slice ν and a source token at time step j in flow slice μ:

$$\alpha^{\nu}_{t,j,\mu} = \mathrm{softmax}_{j,\mu}\left(\frac{\langle q^{\nu}_{t}, k^{\mu}_{j}\rangle}{\sqrt{d}} + b_{t-j,\mu\ominus\nu}\right). \tag{13}$$

The output of the layer for token at time t flow ν is then:

$$\mathrm{SA}_{\nu-\mathrm{mix}}(\eta)^{\nu}_{t} = \left(\sum_{\mu\in V_{\mathrm{model}}}\sum_{j=0}^{T-1}\alpha^{\nu}_{t,j,\mu}v^{\mu}_{j}\right)\star_{\mathcal{V}\times G} W_O. \tag{14}$$

Here, analogous to relative-position self-attention (Shaw et al., 2018), the bias/mask may depend only on relative time and relative flow $\mu\ominus\nu$, so a flow-induced reindexing of both μ and ν leaves the logits unchanged up to permutation. The flow-independent attention in Eq. 9 is recovered by masking to $\mu = \nu$ and using diagonal flow-convolution kernels. We leave evaluation of this variant to future work.

4. Experiments

To validate the construction above, we aim to test whether the proposed FET achieves the canonical out-of-distribution generalization abilities expected of equivariant models. Specifically, in the following, we train both standard and Flow Equivariant Transformers on a simple bump detection task in the presence of noise. For evaluation, we measure if flow equivariant models generalize outside the set of trained velocities better than non-equivariant counterparts.

4.1. Task: 1-D bump detection

Each example is a sequence of $T = 16$ frames on a circular spatial lattice of size $S = 64$ (see examples, Figure 3). The single-channel signal is generated by one of two procedures:

- **Coherent flow:** a Gaussian bump of width $\sigma = 3$ and unit amplitude is placed at a random initial position $p_0 \in \mathbb{Z}/S\mathbb{Z}$ and propagated for T frames at a fixed velocity ν drawn from the dataset's velocity set. Frame t contains the bump at position $(p_0 + t\nu) \bmod S$.
- **Incoherent:** each of the T frames consists of an independent Gaussian bump at a uniformly random position. The marginal per-frame statistics are identical to the positive class; only the temporal coherence differs.

We then add i.i.d. Gaussian noise $\mathcal{N}(0, 0.3^2)$ to every spatiotemporal sample. The model must output a single logit predicting the binary label COHERENT/INCOHERENT. Thus, solving the task requires integrating information across time and cannot be solved per frame. The training set contains

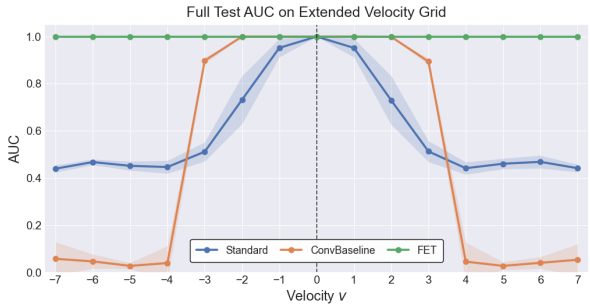

Figure 2. Per-velocity AUC with mean ± std. over 5 seeds on $\mathcal{V}_{\text{test}} = \{-7, \ldots, 7\}$. The FET (green) is flat at 1.000 across the grid. The Standard Transformer (blue) degrades rapidly away from v=0, the in-domain set. The ConvBaseline (orange) retains high AUC only for moderate $|v|$; at large $|v|$, AUC degrades.

10,000 examples, and the validation set contains 2,000 examples, both with $\mathcal{V}_{\text{train}} = \mathcal{V}_{\text{val}} = \{0\}$ (stationary bumps only). Test evaluation uses 2,000 fresh examples per integer velocity in $\mathcal{V}_{\text{test}} = \{-7, -6, \ldots, +6, +7\}$.

All three models are trained for 30 epochs with Adam (Kingma & Ba, 2015) (lr $= 10^{-3}$, weight decay 10^{-4}), cosine annealing to zero, and batch size 64. Each model is trained under 5 different random seeds. For each seed, we retain the checkpoint at the highest validation accuracy.

4.2. Models

We compare our FET construction with two baselines: a 'standard' Transformer with all fully-connected layers; and a 'ConvBaseline' Transformer with all 1-D convolutional layers – matching the construction of the FET but without the flow lift. All models are composed of two pre-norm Transformer blocks (Xiong et al., 2020), and share the same two-layer 1D circular-convolution embedding network construction with kernel size 5, ReLU activations, and 32 hidden channels. The number of output channels is determined by the architecture.

The Standard Transformer is built with $d_{\text{model}} = 64$, 4 heads, $d_{\text{ff}} = 128$, and standard LayerNorm (Ba et al., 2016). The encoder outputs 32 channels, and all channels are flattened into the spatial dimension before being passed to the transformer block. The ConvBaseline is a Transformer built with 1-D convolutions in place of all standard linear layers (with kernel size 3). Each token now has a spatial dimension of 64, and the output of the encoder is set to 224 to account for the extra effective activations of the FET (while keeping training stable). The model is built with ChannelLN, and the MLP is replaced with a CNN that has 448 hidden channels. The architecture is per-frame translation-equivariant by construction but *not* flow-equivariant. The FET uses the same convolutional block structure as the ConvBaseline, but with 32 channels per flow slice and $V_{model} = \{-7, \ldots, +7\}$.

4.3. Results

All three models reach $\geq 99.9\%$ test accuracy on the in-distribution velocity $\nu = 0$, confirming the task is learnable (Tables 1 and 2); however, as expected, the three models exhibit distinct generalization behaviors on the extended grid $\mathcal{V}_{\text{test}} = \{-7, \ldots, 7\}$. In Figure 2, we plot the AUC of all three models as a function of velocity on the test set. We see that the FET generalizes perfectly across the full tested velocity range, up to $\nu = \pm 7$, while both baselines degrade outside a small neighborhood of the training velocity.

5. Conclusion

In this work, we have illustrated how standard self-attention generally breaks flow equivariance in a standard Transformer. We proposed an equivariant alternative to address this and demonstrate that it yields expected generalization benefits on a small toy sequence classification task. We present these results as a preliminary validation of the FET's flow generalization capabilities and our equivariant design.

Limitations. Our construction has three principal limitations. *(i) Scaling of the lift:* since the slice count $V = (2K+1)^d$ grows exponentially in the flow dimension d and polynomially in the truncation K, modeling rich or high-dimensional flows is memory-bound. *(ii) Constant-velocity flows only:* a single Lie-algebra generator ν yields the one-parameter subgroup $\psi_t(\nu) = \exp(t\nu)$, i.e. constant-speed motion; accelerating or otherwise time-varying flows are not captured by one generator and would require lifting over a larger subgroup or time-dependent generators. *(iii) Boundary truncation:* equivariance is exact only when the relative-flow shift permutes stored slices, so for test generators outside $\mathcal{V}_{\text{model}}$ (e.g. $|\nu| > K$) the identity breaks at the boundary and the FET loses its guarantee.

In future work, we intend to explore more efficient alternatives to the discrete flow-lifting embedding layer and evaluate the performance of Flow Equivariant Transformers on more complex tasks such as those involving spatiotemporal forecasting or world modeling (Lillemark et al., 2026).

Acknowledgements

This work has been made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University. T.A.K. is funded by a Kempner Research Fellowship.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Cohen, T. S. and Welling, M. Group equivariant convolutional networks, 2016. URL https://arxiv.org/abs/1602.07576.

Cohen, T. S. and Welling, M. Steerable CNNs. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rJQKYt5ll.

Cohen, T. S., Geiger, M., and Weiler, M. A general theory of equivariant CNNs on homogeneous spaces. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. SE(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1970–1981, 2020.

Hall, B. C. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, volume 222 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 2015.

Hutchinson, M., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. LieTransformer: Equivariant self-attention for lie groups. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4533–4543. PMLR, 2021.

Keller, T. A. Flow equivariant recurrent neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=N1KPOlcN6P.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2747–2755. PMLR, 2018.

Liao, Y.-L. and Smidt, T. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KwmPfARgOTD.

Lillemark, H. J., Huang, B., Zhan, F., Du, Y., and Keller, T. A. Flow equivariant world models: Memory for partially observed dynamic environments, 2026. URL https://arxiv.org/abs/2601.01075.

Ngo, K. and Ravanbakhsh, S. Scaling laws and symmetry, evidence from neural force fields. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qyjaVda7t2.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 464–468, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Weiler, M. and Cesa, G. General E(2)-equivariant steerable CNNs. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 2020.

A. Proofs

Lemma A.1 (Channel-only LayerNorm is G-equivariant)**.** *Let* $\mathrm{ChannelLN}$ *act on a lifted feature* $h_{t,\nu,c}(y)$ *by normalizing over the channel index* c *at each fixed* (t, ν, y) *and applying a per-channel affine* (γ_c, β_c) *that does not depend on* y*. Then for every* $g \in G$*,*

$$\mathrm{ChannelLN}(g \cdot h) \;=\; g \cdot \mathrm{ChannelLN}(h). \tag{15}$$

Proof. The action of g relabels the feature coordinate, $(g \cdot h)_{t,\nu,c}(y) = h_{t,\nu,c}(g^{-1} \cdot y)$. The channel mean and variance at (t, ν, y) computed from $g \cdot h$ therefore equal those computed at $(t, \nu, g^{-1} \cdot y)$ from h, and the affine parameters (γ_c, β_c) are independent of y. Hence the normalization commutes with the relabeling $y \mapsto g^{-1} \cdot y$, i.e. with the action of g. □

B. Experiment Details

B.1. Data generation

The 1-D bump-detection dataset is generated procedurally with a fixed seed per split. For each example we sample a binary label $y \in \{0, 1\}$ uniformly. Conditioned on y we generate a clean spatiotemporal signal $f \in \mathbb{R}^{T \times S}$ as follows:

- **Coherent** ($y = 1$)**:** draw an initial position $p_0 \sim \mathrm{Uniform}(\mathbb{Z}/S\mathbb{Z})$ and a velocity ν from the split's velocity set. For each frame $t \in \{0, \ldots, T-1\}$, place a Gaussian bump of width σ at position $p_t = (p_0 + t\nu) \bmod S$:

$$f_{t,s} \;=\; \exp\Big(-\tfrac{1}{2\sigma^2}\, d_S(s, p_t)^2\Big), \tag{16}$$

  where d_S is the circular distance on $\mathbb{Z}/S\mathbb{Z}$.

- **Incoherent** ($y = 0$)**:** for each frame t independently, draw $p_t \sim \mathrm{Uniform}(\mathbb{Z}/S\mathbb{Z})$ and place the same Gaussian bump at p_t. The marginal per-frame statistics match the coherent class exactly; only the temporal coupling differs.

We then add i.i.d. Gaussian noise $\eta \sim \mathcal{N}(0, \sigma_{\text{noise}}^2)$ to every (t, s) entry. We use $S = 64$, $T = 16$, $\sigma = 3.0$, and $\sigma_{\text{noise}} = 0.3$ throughout.

The training set (10,000 examples) draws velocities from $\mathcal{V}_{\text{train}} = \{0\}$. The in-distribution validation set (2,000 examples) also uses $v = 0$. The held-out test sets (2,000 examples each) use every integer $v \in \{-7, -6, \ldots, +6, +7\}$. Different RNG seeds are used for train, val, and each test split, so no example overlap is possible.

B.2. Architectures

Shared encoder. All three models use the shared $\mathrm{ConvEncoder}$ module: two $\mathrm{CircularConv1d}$ layers ($1 \to 32 \to C_{\text{out}}$) with kernel size 5 and ReLU, applied independently per frame, with $C_{\text{out}} = 32$ for the Standard and FET and $C_{\text{out}} = 224$ for the ConvBaseline (Section 4.2). The Standard Transformer reshapes the $32 \times S$ feature map per time step, applies a learned linear map into $d_{\text{model}} = 64$, and adds learned positional embeddings; spatial structure is therefore available to attention, but neither flow nor translation equivariance is enforced in deeper blocks.

Standard Transformer. The encoder output $z \in \mathbb{R}^{T \times 32 \times S}$ is flattened to $z \in \mathbb{R}^{T \times (32 \cdot S)}$ per time step, projected by a learned linear map to dimension $d_{\text{model}} = 64$, and combined with learned absolute positional embeddings along T. We then apply $N = 2$ standard pre-norm transformer blocks with 4 heads, $d_{\text{ff}} = 128$, and standard $\mathrm{LayerNorm}$. A final $\mathrm{LayerNorm}$ is followed by mean pooling over all tokens and a linear classifier to a single logit. Total parameters: 204,641.

ConvBaseline (untied 224-channel). A second baseline that retains the FET's convolutional building blocks but removes the lift, isolating the contribution of explicit flow equivariance. The encoder produces $z \in \mathbb{R}^{T \times 224 \times S}$ with $224 = 32 \cdot 7$ channels. We then apply $N = 2$ pre-norm convolutional transformer blocks. Each block contains:

- $\mathrm{ChannelLN}$: a channel-wise LayerNorm applied per (t, s) slice;
- Single-head dot-product attention over the time axis with $\mathrm{CircularConv1d}$-based Q, K, V projections of kernel size 3 (the per-token feature is the flattened (C, S) map, scaled by $\sqrt{C \cdot S}$);

- Position-wise MLP implemented as two $\mathrm{CircularConv1d}$ layers ($224 \to 448 \to 224$, kernel size 3) with ReLU activations.

After the N blocks we apply a final $\mathrm{ChannelLN}$, mean-pool over (t, s), and project to a single logit. The architecture is per-frame translation-equivariant by construction (every layer is a $\mathrm{CircularConv1d}$ or a $\mathrm{ChannelLN}$, both of which commute with σ_d), but it is not flow-equivariant: a flow $\psi(\hat{\nu})$ shifts each frame by a different amount, and no operation in the network compensates for this. Total parameters: 2,148,801 (approximately $44\times$ the FET).

Flow Equivariant Transformer. The encoder output $z \in \mathbb{R}^{T\times 32\times S}$ is lifted (Section 3) to $h^0 \in \mathbb{R}^{T\times V\times 32\times S}$ with $\mathcal{V}_{\text{model}} = \{-7, -6, \ldots, 6, 7\}$ ($|\mathcal{V}_{model}| = 15$). We then apply $N = 2$ pre-norm flow-equivariant blocks. Each block contains:

- $\mathrm{ChannelLN}$ as in the ConvBaseline, applied per (t, ν, s) slice;
- Single-head dot-product self-attention over the time axis with $\mathrm{CircularConv1d}$-based Q, K, V projections of kernel size 3, applied independently per velocity slice;
- Position-wise MLP implemented as two $\mathrm{CircularConv1d}$ layers ($32 \to 64 \to 32$, kernel size 3) with ReLU, applied independently per (t, ν) slice;

After the N blocks we apply a final $\mathrm{ChannelLN}$, mean-pool over (t, ν, s), and project to a single logit. Total parameters: 49,089.

B.3. Training

We train all three models with Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$), peak learning rate 10^{-3}, weight decay 10^{-4}, batch size 64, and cosine annealing to 0 over 30 epochs. Loss is binary cross-entropy with logits. The ConvBaseline additionally uses 3 epochs of linear learning-rate warmup (ramping the LR from $\frac{1}{3} \cdot 10^{-3}$ to 10^{-3} over the first 3 epochs, before the cosine schedule runs for the remaining 27) and global gradient clipping at norm 1.0; with these stabilizers all 5 seeds converge to the in-distribution optimum. Without them, some seeds fail to escape a saturated initialization, which we attribute to the larger softmax-feature dimension ($C \cdot S = 224 \cdot 64$ vs $32 \cdot 64$ for the FET) at the same peak learning rate. The Standard Transformer and FET converge reliably without warmup or clipping. Validation is run every epoch on the in-distribution ($v = 0$) set; we retain the checkpoint achieving the highest validation accuracy, and when multiple late epochs tie at that maximum we keep the last one so the linear head reflects calibration after accuracy saturates.

B.4. Multi-seed protocol

For each architecture we train 5 independent runs with seeds $\{42, 43, 44, 45, 46\}$. The seed controls (a) PyTorch and NumPy RNGs for parameter initialization and dropout, (b) the data-loader shuffle order, and (c) the dataset RNG for any per-epoch resampling. The same seed produces identical per-epoch validation accuracy across two re-runs of the same configuration on the same hardware (verified by CLI test). Test data are deterministic across seeds, so cross-seed variance is purely model-side.

B.5. Extended per-velocity metrics

Tables 1 and 2 list mean $\pm$ std over 5 seeds for every integer test velocity in $\{-7, \ldots, 7\}$. We see the FET achieves perfect accuracy and AUC throughout the range, while the ConvBaseline and Standard Transformer struggle. The ConvBaseline can retain threshold-free separability (AUC) at some shifted velocities while losing calibration (accuracy) under the fixed validation threshold; at larger velocities, its scores become systematically anti-correlated with the label.

B.6. Dataset Samples

Table 1. Full test accuracy (%) on the extended velocity grid.

v	Standard	ConvBaseline	FET
-7	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
-6	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
-5	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
-4	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
-3	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
-2	55.7 ± 3.5	50.0 ± 0.0	100.0 ± 0.0
-1	79.6 ± 3.5	99.7 ± 0.6	100.0 ± 0.0
0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
1	79.4 ± 3.2	99.7 ± 0.5	100.0 ± 0.0
2	56.0 ± 3.8	50.0 ± 0.0	100.0 ± 0.0
3	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
4	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
5	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
6	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
7	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0

Table 2. Full test AUC on the extended velocity grid.

v	Standard	ConvBaseline	FET
-7	0.440 ± 0.015	0.058 ± 0.071	1.000 ± 0.000
-6	0.468 ± 0.012	0.047 ± 0.030	1.000 ± 0.000
-5	0.452 ± 0.019	0.028 ± 0.013	1.000 ± 0.000
-4	0.447 ± 0.027	0.040 ± 0.072	1.000 ± 0.000
-3	0.512 ± 0.040	0.898 ± 0.013	1.000 ± 0.000
-2	0.732 ± 0.101	1.000 ± 0.000	1.000 ± 0.000
-1	0.952 ± 0.039	1.000 ± 0.000	1.000 ± 0.000
0	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
1	0.952 ± 0.039	1.000 ± 0.000	1.000 ± 0.000
2	0.730 ± 0.103	0.999 ± 0.000	1.000 ± 0.000
3	0.513 ± 0.044	0.894 ± 0.013	1.000 ± 0.000
4	0.442 ± 0.026	0.046 ± 0.081	1.000 ± 0.000
5	0.461 ± 0.022	0.028 ± 0.016	1.000 ± 0.000
6	0.469 ± 0.027	0.041 ± 0.025	1.000 ± 0.000
7	0.442 ± 0.016	0.054 ± 0.066	1.000 ± 0.000

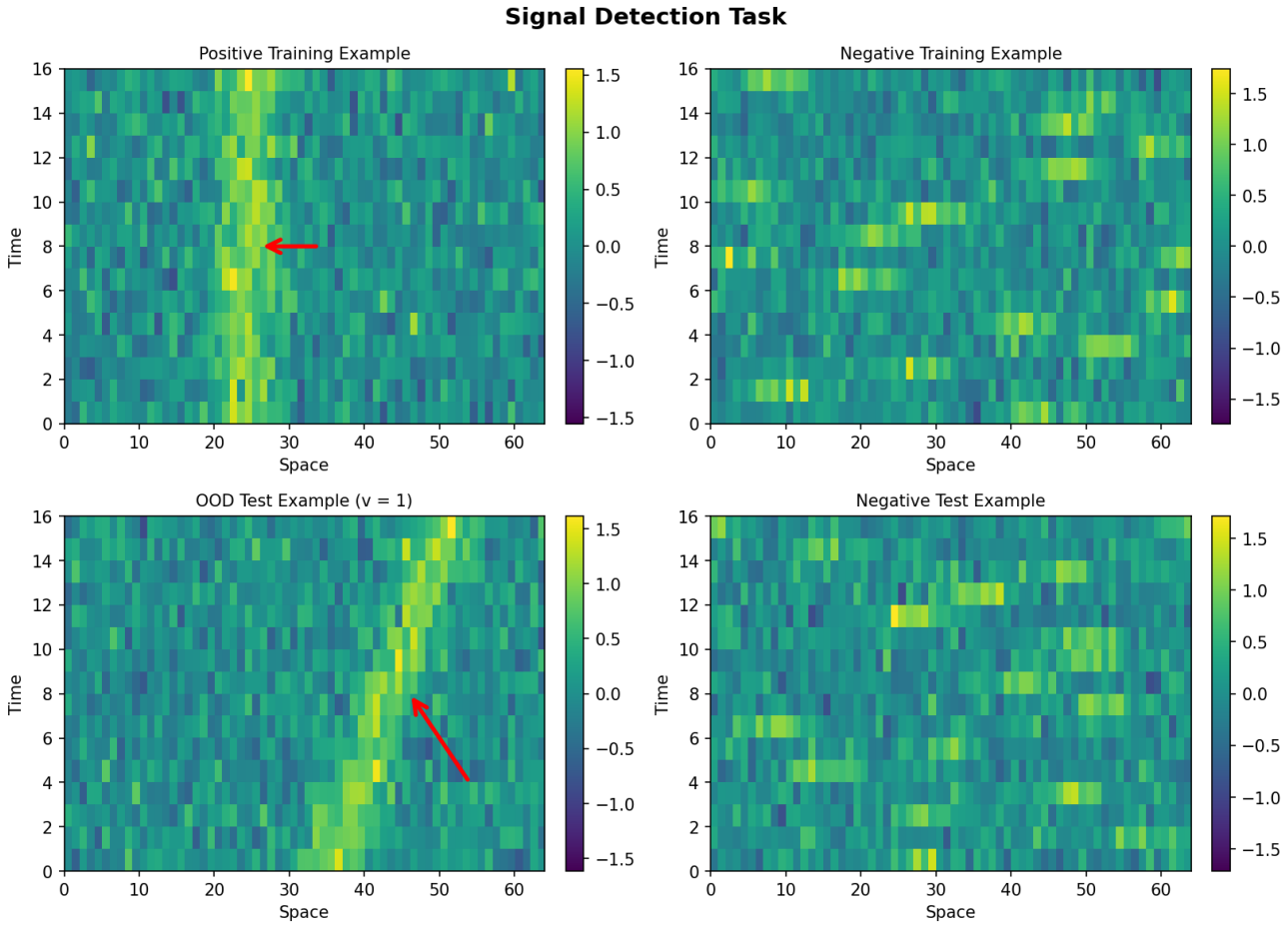

Figure 3. Examples from the 1-D bump detection task. We show a positive training example (top left), a stationary bump, and a negative training example (top right) an incoherent bump. In the bottom left we show a corresponding sample from the test set with a velocity of 1. On the bottom right we show a corresponding negative example from the test set.