

883 Appendices

884 Table of Contents

886	A Extended Proofs	22
887	A.1 Equivariance of Group-Lifting Convolution & Group-Convolution	22
888	A.2 Equivariance of Group Equivariant RNN	23
889	A.3 Frame-wise Flow Equivariance	23
890	A.4 Group Equivariant RNNs are not Flow Equivariant	24
891	A.5 Equivariance of Lifting Flow Convolution	26
892	A.6 Equivariance of Flow Convolution	26
893	A.7 Equivariance of FERNN	26
894	B Experiment Details	28
895	B.1 Flowing MNIST: Dataset Creation	28
896	B.2 Flowing MNIST: Models	29
897	B.3 Flowing MNIST: Next-Step Prediction Training & Evaluation	30
898	B.4 Moving KTH: Datasets	31
899	B.5 Moving KTH: Models	31
900	B.6 Moving KTH: Action Recognition Training & Evaluation	32
901	B.7 Moving KTH Samples	32
902	B.8 Compute	33
903	C Variations of FERNNs	34
904	C.1 FERNN with Non-Trivial Lift	34
905	D Related Work	35
906	D.1 Flow Equivariance without a Hidden State	35
907	D.2 ‘Statically Equivariant’ Sequence Models	35
908	D.3 Neuroscience and Biologically Inspired Neural Networks	35
909	D.4 Reference Frames in Neural Networks	36
910	D.5 Broadly Related Work	36
911	D.6 Equivariant Dynamical Systems	37
912	E Extended Results: Flowing MNIST	38
913	E.1 In-Distribution Next-Step Prediction (Table 1 & Figure 2)	38
914	E.2 Length Generalization Visualizations	38
915	E.3 Velocity Generalization Visualizations	40

919 A Extended Proofs

920 A.1 Equivariance of Group-Lifting Convolution & Group-Convolution

921 We can verify that group convolutional layers given by Equation 3 are equivariant according to
 922 Equation 2. While these are well-known facts, we re-derive them here in our own notation for
 923 completeness and to lay the groundwork for our later proofs. First, for the lifting convolution, we
 924 wish to prove:

$$[(\hat{g} \cdot f) \star_G \mathcal{W}^i](g) = \hat{g} \cdot [f \star_G \mathcal{W}^i](g) \quad \forall g, \hat{g} \in G, f \in \mathcal{F}_K(X). \quad (15)$$

925 *Proof.* (Group-Lifting Conv. is Group Equivariant)

$$[(\hat{g} \cdot f) \star_G \mathcal{W}^i](g) = \sum_{x \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{g}^{-1} \cdot x) \mathcal{W}_k^i(g^{-1} \cdot x) \quad (\text{by def. action, Eqn. 1}) \quad (16)$$

$$= \sum_{\hat{x} \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{x}) \mathcal{W}_k^i(g^{-1} \cdot (\hat{g} \cdot \hat{x})) \quad (\text{where } \hat{x} = \hat{g}^{-1} \cdot x) \quad (17)$$

$$= \sum_{\hat{x} \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{x}) \mathcal{W}_k^i((g^{-1} \cdot \hat{g}) \cdot \hat{x}) \quad (\text{by associativity}) \quad (18)$$

$$= \sum_{\hat{x} \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{x}) \mathcal{W}_k^i((\hat{g}^{-1} \cdot g)^{-1} \cdot \hat{x}) \quad (\text{by defn. inverse}) \quad (19)$$

$$= [f \star_G \mathcal{W}^i](\hat{g}^{-1} \cdot g) = \hat{g} \cdot [f \star_G \mathcal{W}^i](g) \quad (\text{by Eqn. 3 \& defn. action}) \quad (20)$$

926 □

927 In line 17 we use the fact that because the group G acts on X by a bijection (see our earlier definition
 928 of the input representation $\pi_X : G \rightarrow \text{Aut}(X)$ in §2), the substitution $\hat{x} = g^{-1} \cdot x$ is just a relabeling
 929 of the index set, and hence the sum over $x \in X$ is the same as the sum over $\hat{x} \in X$.

930 Next for the group convolution, the proof is virtually identical. We wish to prove:

$$[(\hat{g} \cdot f) \star_G \mathcal{W}^i](g) = \hat{g} \cdot [f \star_G \mathcal{W}^i](g) \quad \forall g, \hat{g} \in G, f \in \mathcal{F}_K(X). \quad (21)$$

931 *Proof.* (Group-Conv. is Group Equivariant)

$$[(\hat{g} \cdot f) \star_G \mathcal{W}^i](g) = \sum_{h \in G} \sum_{k=1}^K f_k(\hat{g}^{-1} \cdot h) \mathcal{W}_k^i(g^{-1} \cdot h) \quad (\text{by def. action, Eqn. 1}) \quad (22)$$

$$= \sum_{\hat{h} \in G} \sum_{k=1}^K f_k(\hat{h}) \mathcal{W}_k^i(g^{-1} \cdot (\hat{g} \cdot \hat{h})) \quad (\text{where } \hat{h} = \hat{g}^{-1} \cdot h) \quad (23)$$

$$= \sum_{\hat{h} \in G} \sum_{k=1}^K f_k(\hat{h}) \mathcal{W}_k^i((g^{-1} \cdot \hat{g}) \cdot \hat{h}) \quad (\text{by associativity}) \quad (24)$$

$$= \sum_{\hat{h} \in G} \sum_{k=1}^K f_k(\hat{h}) \mathcal{W}_k^i((\hat{g}^{-1} \cdot g)^{-1} \cdot \hat{h}) \quad (\text{by defn. inverse}) \quad (25)$$

$$= [f \star_G \mathcal{W}^i](\hat{g}^{-1} \cdot g) = \hat{g} \cdot [f \star_G \mathcal{W}^i](g) \quad (\text{by Eqn. 3 \& defn. action}) \quad (26)$$

932 □

933 Again, we use the fact that the group G is closed under the group action, so the substitution $\hat{h} = \hat{g}^{-1} \cdot h$
 934 is just a relabeling of the index set, and hence the sum over $h \in G$ is the same as the sum over $\hat{h} \in G$.

A.2 Equivariance of Group Equivariant RNN

We wish to prove that a recurrent neural network built with group-equivariant convolutional layers (Equation 4) is equivariant according to Equation 2. If we write the hidden state as a function of the input signal $h_{t+1}[f_{\leq t}](g) \in \mathcal{F}_{K'}(Y)$, where $Y = G$, then we can write the the equivariance condition as:

$$h_{t+1}[\hat{g} \cdot f_{\leq t}](g) = \hat{g} \cdot h_{t+1}[f_{\leq t}](g) \quad \forall t \in \mathbb{Z}_+, f \in \mathcal{F}_K(X), \hat{g}, g \in G, \quad (27)$$

where $\hat{g} \cdot f_{\leq t} := \{\hat{g} \cdot f_i \mid i \in \mathbb{Z}_+ \leq t\}$, denotes the input signal generated by applying the same group element to each timestep. We prove this by induction for all $t \in \mathbb{Z}_+$:

Proof. (G-RNN is Group Equivariant)

We assume (i) $\hat{\star}_G$ and \star_G are the group-lifting convolution and group convolution defined in Equation 3 (ii) σ is a G-equivariant non-linearity (e.g. pointwise), and (iii) g acts linearly on h -space. Since h is defined by the lifting and recurrent convolutions, we see that $Y = G$. We further assume (iv) h_0 is initialized to be invariant to the group action, i.e. $\hat{g} \cdot h_0(g) = h_0(\hat{g}^{-1} \cdot g) = h_0(g)$, and (v) the input signal is zero before time zero, i.e. $f_{<0} = \mathbf{0}$.

Base Case: We can see that the base case is trivially true from the initial condition, since the initial condition is not a function of the input sequence, so:

$$h_0[\hat{g} \cdot f_{<0}](g) = h_0[f_{<0}](g) \quad (\text{since } h_0 \text{ is indep. of input } f_{<0}) \quad (28)$$

$$= h_0[f_{<0}](\hat{g}^{-1} \cdot g) \quad (\text{by initialization}) \quad (29)$$

$$= \hat{g} \cdot h_0[f_{<0}](g) \quad (\text{by defn. action}) \quad (30)$$

Inductive Step: Assuming $h_t[\hat{g} \cdot f_{<t}](g) = \hat{g} \cdot h_t[f_{<t}](g)$, we wish to prove this holds also for $t + 1$:

$$h_{t+1}[\hat{g} \cdot f_{\leq t}](g) = \sigma([h_t[\hat{g} \cdot f_{<t}] \star_G \mathcal{W}](g) + [\hat{g} \cdot f_t] \hat{\star}_G \mathcal{U})(g) \quad (\text{by defn. G-RNN}) \quad (31)$$

$$= \sigma([\hat{g} \cdot h_t[f_{<t}]] \star_G \mathcal{W})(g) + [\hat{g} \cdot f_t] \hat{\star}_G \mathcal{U}(g) \quad (\text{by inductive hyp.}) \quad (32)$$

$$= \sigma(\hat{g} \cdot [h_t[f_{<t}] \star_G \mathcal{W}](g) + \hat{g} \cdot [f_t \hat{\star}_G \mathcal{U}](g)) \quad (\text{by equivar. of G-conv.}) \quad (33)$$

$$= \hat{g} \cdot \sigma([h_t[f_{<t}] \star_G \mathcal{W}](g) + [f_t \hat{\star}_G \mathcal{U}](g)) \quad (\text{by equivar. non-lin.}) \quad (34)$$

$$= \hat{g} \cdot h_{t+1}[f_{\leq t}](g) \quad (\text{by defn. G-RNN, Eqn 4}) \quad (35)$$

951

□

We note that the step on the fourth line assumes that the group action is linear, in that it distributes over the addition operation between the previous hidden state and the input. Traditionally, group equivariant neural network architectures have been constructed to exhibit linear representations in the latent space, and therefore this is a natural assumption. However, if one allowed g to act by non-linear maps (i.e. dropped the linear-representation assumption), then neither distributivity nor σ -equivariance would hold in general.

A.3 Frame-wise Flow Equivariance

We wish to prove that any G-equivariant map $\phi : \mathcal{F}_K(X) \rightarrow \mathcal{F}_{K'}(Y)$, that is applied ‘frame-wise’ to a space-time function $f \in \mathcal{F}_K(X, \mathbb{Z})$ is also flow equivariant according to Definition 3.1. Specifically, let $\Phi[f] = [\phi(f_0), \phi(f_1) \dots \phi(f_T)]$ be a sequence model built by concatenating the output of a G-equivariant map ϕ applied to each timestep t of the input signal f_t . Furthermore, let ϕ be equivariant to the individual group elements generated by vector fields $\nu \in V$, i.e. $\phi(\psi_t(\nu) \cdot f_t) = \psi_t(\nu) \cdot \phi(f_t) \forall t \in \mathbb{Z}, \nu \in V, f_t \in \mathcal{F}_K(X)$. Then:

Proof. (Frame-wise Equivariant Maps are Flow Equivariant)

$$\Phi(\psi(\nu) \cdot f) = [\phi(\psi_0(\nu) \cdot f_0), \phi(\psi_1(\nu) \cdot f_1) \dots \phi(\psi_T(\nu) \cdot f_T)] \quad (\text{by defn. flow action}) \quad (36)$$

$$= [\psi_0(\nu) \cdot \phi(f_0), \psi_1(\nu) \cdot \phi(f_1) \dots \psi_T(\nu) \cdot \phi(f_T)] \quad (\text{by G-equivariance of } \phi) \quad (37)$$

$$= \psi(\nu) \cdot [\phi(f_0), \phi(f_1) \dots \phi(f_T)] \quad (\text{by defn. flow action, eqn. 5}) \quad (38)$$

$$= (\psi(\nu) \cdot \Phi(f)) \quad (\text{by defn. } \Phi) \quad (39)$$

966

□

967 A.4 Group Equivariant RNNs are not Flow Equivariant

968 In this subsection we prove Theorem 3.1, that the group-equivariant RNN as defined in Equation 4,
969 with non-zero \mathcal{W} , is not flow equivariant according to Definition 3.1, except in the degenerate flow
970 invariant case.

971 We will prove this in two parts: (i) First we will prove that a G-RNN with constant kernels is indeed
972 flow invariant through induction (the degenerate case). Then (ii), we will proceed with a proof by
973 contradiction to show that this is the only such flow equivariant G-RNN possible.

974 To begin, we recall that the degenerate flow invariant case is defined as a G-RNN where both \mathcal{W} and
975 \mathcal{U} are constant over G . We will denote such kernels $\bar{\mathcal{U}}$ and $\bar{\mathcal{W}}$, such that:

$$\bar{\mathcal{W}}(g) = \bar{\mathcal{W}}(g'), \quad \forall g, g' \in G \quad \& \quad \bar{\mathcal{U}}(g \cdot x) = \bar{\mathcal{U}}(g' \cdot x) \quad \forall g, g' \in G, x \in X. \quad (40)$$

976 We wish to prove that such a network satisfies Definition 3.1, but more specifically that it is actually
977 invariant to the flow action, meaning:

$$h_t[\psi(\nu) \cdot f_{<t}] = h_t[f_{<t}] \quad \forall f \in \mathcal{F}_K(X), \nu \in V, t \in \mathbb{Z}_+ \quad (41)$$

978 We can see that this is a special case of Definition 3.1 where the group action on the output space is
979 trivial, i.e. given by the identity: $\psi_t(\nu) \cdot h_t[f_{<t}] = h_t[f_{<t}]$. We can prove this by induction as before.
980 First we introduce the following lemma for convenience:

981 *Lemma A.1.* (Lifting convolution with $\bar{\mathcal{U}}$ is group invariant) The result of applying the lifting
982 convolution ($\hat{\star}_G$) defined in 3 with a constant kernel $\bar{\mathcal{U}}$ to a signal $f \in \mathcal{F}_K(X)$ is invariant to flow
983 element action on f , i.e.

$$[(\psi_t(\nu) \cdot f) \hat{\star}_G \bar{\mathcal{U}}^i](g) = [f \hat{\star}_G \bar{\mathcal{U}}^i](g) \quad \forall \nu \in V, f \in \mathcal{F}_K(X). \quad (42)$$

984 *Proof.* (Lemma A.1)

$$[(\psi_t(\nu) \cdot f) \hat{\star}_G \bar{\mathcal{U}}^i](g) = \sum_{x \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\psi_t(\nu)^{-1} \cdot x) \bar{\mathcal{U}}_k^i(g^{-1} \cdot x) \quad (\text{by def. action, Eqn. 1}) \quad (43)$$

$$= \sum_{\hat{x} \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{x}) \bar{\mathcal{U}}_k^i(g^{-1} \cdot (\psi_t(\nu) \cdot \hat{x})) \quad (\text{where } \hat{x} = \psi_t(\nu)^{-1} \cdot x) \quad (44)$$

$$= \sum_{\hat{x} \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{x}) \bar{\mathcal{U}}_k^i((\psi_t(\nu)^{-1} \cdot g)^{-1} \cdot \hat{x}) \quad (\text{by associativity and inv.}) \quad (45)$$

$$= \sum_{\hat{x} \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{x}) \bar{\mathcal{U}}_k^i(\hat{g}^{-1} \cdot \hat{x}) \quad (\text{by closure, } \hat{g} = \psi_t(\nu)^{-1} \cdot g \in G) \quad (46)$$

$$= \sum_{\hat{x} \in \mathbb{Z}^2} \sum_{k=1}^K f_k(\hat{x}) \bar{\mathcal{U}}_k^i(g^{-1} \cdot \hat{x}) \quad (\text{by defn } \bar{\mathcal{U}}(g^{-1} \cdot \hat{x}) = \bar{\mathcal{U}}(\hat{g}^{-1} \cdot \hat{x})) \quad (47)$$

$$= [f \hat{\star}_G \bar{\mathcal{U}}^i](g) \quad (\text{by Eqn. 3}) \quad (48)$$

985

□

986 We have again used the fact that the group acts on X by a bijection as in §A.1. We can then proceed
987 with the proof by induction to prove that the G-RNN with constant kernels is flow invariant:

988 *Proof.* (G-RNN with Constant Kernels is Flow Invariant)

989 Let $h_t[f_{<t}]$ be defined as in Equation 4 with $\mathcal{U} = \bar{\mathcal{U}}$ and $\mathcal{W} = \bar{\mathcal{W}}$. We again assume (i) $\hat{\star}_G$ and \star_G are
990 the group-lifting convolution and group convolution defined in Equation 3, (ii) σ is a G-equivariant
991 non-linearity (e.g. pointwise), (iii) g acts linearly on h -space, (iv) h_0 is initialized to be invariant to
992 the group action of the individual flow elements, i.e. $\psi_t(\nu) \cdot h_0(g) = h_0(\psi_t(\nu)^{-1} \cdot g) = h_0(g)$, and
993 (v) the input signal is zero before time zero, i.e. $f_{<0} = 0$.

994 **Base Case:** We can see that the base case is again trivially true from the initial condition, since the
 995 initial condition is not a function of the input sequence, so:

$$h_0[\psi(\nu) \cdot f_{<0}](g) = h_0[f_{<0}](g) \quad (\text{since } h_0 \text{ is indep. of input } f_{<0}) \quad (49)$$

996 **Inductive Step:** Assuming the inductive hypothesis that $h_t[\psi(\nu) \cdot f_{<t}] = h_t[f_{<t}]$, we wish to prove
 997 this holds for $t + 1$:

$$h_{t+1}[\psi(\nu) \cdot f_{\leq t}](g) = \sigma([h_t[\psi(\nu) \cdot f_{<t}] \star_G \bar{\mathcal{W}}](g) + [\psi_t(\nu) \cdot f_t] \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (\text{by defn. G-RNN}) \quad (50)$$

$$= \sigma([h_t[f_{<t}] \star_G \bar{\mathcal{W}}](g) + [\psi_t(\nu) \cdot f_t] \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (\text{by inductive hyp.}) \quad (51)$$

$$= \sigma([h_t[f_{<t}] \star_G \bar{\mathcal{W}}](g) + [f_t \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (\text{by Lemma A.1}) \quad (52)$$

$$= h_{t+1}[f_{\leq t}](g) \quad (\text{by eqn. 4}) \quad (53)$$

998

□

999 Finally then, we use a proof by contradiction to show that this degenerate flow invariant case is the
 1000 only possible case for the G-RNN to be flow equivariant.

1001 *Proof.* (Theorem 3.1) G-RNNs are not Generally Flow Equivariant)

1002 Assert the converse, that the G-RNN defined in Equation 4 is flow equivariant, and that the kernels
 1003 \mathcal{W} and \mathcal{U} , are not constant, i.e. $\mathcal{W} \neq \bar{\mathcal{W}}$ & $\mathcal{U} \neq \bar{\mathcal{U}}$ (as defined in Equation 40). Then, it should be the
 1004 case that:

$$h_{t+1}[\psi(\nu) \cdot f_{\leq t}](g) = (\psi(\nu) \cdot h[f_{\leq t}])_{t+1}(g) = h_{t+1}[f_{\leq t}](\psi_t(\nu)^{-1} \cdot g) \quad (54)$$

1005 For $t = 0$ we have the following:

$$h_1[\psi(\nu) \cdot f_{\leq 0}](g) = \sigma([h_0[\psi(\nu) \cdot f_{<0}] \star_G \bar{\mathcal{W}}](g) + [\psi_0(\nu) \cdot f_0] \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (55)$$

$$= \sigma([h_0[f_{<0}] \star_G \bar{\mathcal{W}}](g) + [f_0 \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (\text{by defn. } h_0 \text{ & } \psi_0) \quad (56)$$

$$= h_1[f_{\leq 0}](g) \quad (57)$$

1006 For the next step, we get:

$$h_2[\psi(\nu) \cdot f_{\leq 2}](g) = \sigma([h_1[\psi(\nu) \cdot f_{\leq 0}] \star_G \bar{\mathcal{W}}](g) + [\psi_1(\nu) \cdot f_1] \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (58)$$

$$= \sigma([h_1[f_{\leq 0}] \star_G \bar{\mathcal{W}}](g) + \psi_1(\nu) \cdot [f_1 \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (\text{by above & g-conv.}) \quad (59)$$

1007 If we look at the action of the flow on the output space, according to flow equivariance, we should
 1008 have:

$$(\psi(\nu) \cdot h[f_{\leq 2}])_2(g) = \sigma(\psi_1(\nu) \cdot [h_1[f_{\leq 0}] \star_G \bar{\mathcal{W}}](g) + [\psi_1(\nu) \cdot f_1] \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (60)$$

$$= \sigma(\psi_1(\nu) \cdot [h_1[f_{\leq 0}] \star_G \bar{\mathcal{W}}](g) + \psi_1(\nu) \cdot [f_1 \hat{\star}_G \bar{\mathcal{U}}](g)) \quad (\text{by g-conv.}) \quad (61)$$

1009 Setting $h_2[\psi(\nu) \cdot f_{\leq 2}](g) = (\psi(\nu) \cdot h[f_{\leq 2}])_2(g)$, as per our assumption, we see the input terms
 1010 match exactly, but the h_{t-1} terms imply the following equivalence:

$$[h_1[f_{\leq 0}] \star_G \bar{\mathcal{W}}](g) = \psi_1(\nu) \cdot [h_1[f_{\leq 0}] \star_G \bar{\mathcal{W}}](g) \quad (62)$$

$$= [h_1[f_{\leq 0}] \star_G \bar{\mathcal{W}}](\psi_1(\nu)^{-1} \cdot g) \quad (\text{by action defn.}) \quad (63)$$

1011 We see this is precisely the ‘lagging’ hidden state that we visualized in Figure 1. In order for this
 1012 equality to hold for all $\nu \in V$, the output of the convolution must be constant along the flows
 1013 generated by all ν . We can see that this is only satisfied by $\mathcal{W} = \bar{\mathcal{W}}$, a contradiction.

1014

□

1015 A.5 Equivariance of Lifting Flow Convolution

1016 In this section we verify that the flow-lifting convolution given by Equation 7 is equivariant to action
 1017 of the individual flow elements, with the following representation of the action:

$$\psi_t(\nu) \cdot [f_t \star_{V \times G} \mathcal{U}^i](\nu, g) = [(\psi_t(\nu) \cdot f_t) \star_{V \times G} \mathcal{U}^i](\nu, g) := [f_t \star_{V \times G} \mathcal{U}^i](\nu, \psi_t(\nu)^{-1} \cdot g) \quad (64)$$

1018 Since the flow lifting convolution is a trivial lift, equivalent to the group-lifting convolution with an
 1019 extra duplicated index, this proof is a trivial replication of the group-lifting convolution proof:

1020 *Proof.* (Flow-Lifting Conv. is Flow Equivariant)

$$[(\psi_t(\nu) \cdot f_t) \star_{V \times G} \mathcal{U}^i](\nu, g) = \sum_{x \in X} \sum_{k=1}^K f_k(\psi_t(\nu)^{-1} \cdot x) \mathcal{U}_k^i(g^{-1} \cdot x) \quad (\text{by defn. action}) \quad (65)$$

$$= \sum_{\hat{x} \in X} \sum_{k=1}^K f_k(\hat{x}) \mathcal{U}_k^i(g^{-1} \cdot (\psi_t(\nu) \cdot \hat{x})) \quad (\text{where } \hat{x} = \psi_t(\nu)^{-1} \cdot x) \quad (66)$$

$$= \sum_{\hat{x} \in X} \sum_{k=1}^K f_k(\hat{x}) \mathcal{U}_k^i((\psi_t(\nu)^{-1} \cdot g)^{-1} \cdot \hat{x}) \quad (\text{by associativity \& inv.}) \quad (67)$$

$$= [f_t \star_{V \times G} \mathcal{U}^i](\nu, (\psi_t(\nu)^{-1} \cdot g)) \quad (68)$$

1021 \square

1022 A.6 Equivariance of Flow Convolution

1023 In this section, we prove that the flow convolution in Equation 9 is equivariant to the action of the
 1024 individual flow elements yielding:

$$[(\psi_t(\nu) \cdot h) \star_{V \times G} \mathcal{W}^i](\nu, g) = \psi_t(\nu) \cdot [h \star_{V \times G} \mathcal{W}^i](\nu, g) \quad (69)$$

1025 *Proof.* (Flow Conv. is Flow Equivariant)

$$[(\psi_t(\nu) \cdot h) \star_{V \times G} \mathcal{W}^i](\nu, g) = \sum_{\gamma \in V} \sum_{m \in G} \sum_{k=1}^{K'} h_k(\gamma, \psi_t(\nu)^{-1} \cdot m) \mathcal{W}_k^i(\gamma - \nu, g^{-1} \cdot m) \quad (70)$$

$$(\text{where } \hat{m} = \psi_t(\nu)^{-1} \cdot m) = \sum_{\gamma \in V} \sum_{\hat{m} \in G} \sum_{k=1}^{K'} h_k(\gamma, \hat{m}) \mathcal{W}_k^i(\gamma - \nu, g^{-1} \cdot (\psi_t(\nu) \cdot \hat{m})) \quad (71)$$

$$(\text{by associativity \& inverse}) = \sum_{\gamma \in V} \sum_{\hat{m} \in G} \sum_{k=1}^{K'} h_k(\gamma, \hat{m}) \mathcal{W}_k^i(\gamma - \nu, (\psi_t(\nu)^{-1} \cdot g)^{-1} \cdot \hat{m}) \quad (72)$$

$$(\text{by Eqn. 9}) = [h \star_{V \times G} \mathcal{W}^i](\nu, \psi_t(\nu)^{-1} \cdot g) \quad (73)$$

$$(\text{by action}) = \psi_t(\nu) \cdot [h \star_{V \times G} \mathcal{W}^i](\nu, g) \quad (74)$$

1026 \square

1027 A.7 Equivariance of FERNN

1028 In this section, we prove Theorem 4.1 by induction. We restate the theorem below:

1029 *Theorem.* (FERNNs are flow equivariant) Let $h[f] \in \mathcal{F}_{K'}(Y, \mathbb{Z})$ be a FERNN as defined in Equations
 1030 7, 9, and 11 with hidden-state initialization invariant to the group action and constant in the flow
 1031 dimension, i.e. $h_0(\nu, g) = h_0(\nu', g) \forall \nu', \nu \in V$ and $\psi_1(\nu) \cdot h_0(\nu, g) = h_0(\nu, g) \forall \nu \in V, g \in G$.
 1032 Then, $h[f]$ is flow equivariant according to Definition 3.1 with the following representation of the
 1033 action of the flow in the output space for $t \geq 1$:

$$(\psi(\hat{\nu}) \cdot h[f])_t(\nu, g) = h_t[f](\nu - \hat{\nu}, \psi_{t-1}(\hat{\nu})^{-1} \cdot g) \quad (75)$$

1034 We note for the sake of completeness, that this then implies the follow equivariance relations:

$$h_t[\psi(\hat{\nu}) \cdot f](\nu, g) = h_t[f](\nu - \hat{\nu}, \psi_{t-1}(\hat{\nu})^{-1} \cdot g) = \psi_{t-1}(\hat{\nu}) \cdot h_t[f](\nu - \hat{\nu}, g) \quad (76)$$

1035 *Proof.* (Theorem 4.1 FERNs are flow equivariant)

1036 Identical to the G-RNN, we assume that σ is a G-equivariant non-linearity, g acts linearly on h -space,
1037 $h_0(\nu, g)$ is defined constant as above, and the input signal is zero before time zero, i.e. $f_{<0} = \mathbf{0}$.

1038 Base Case: The base case is trivially true from the initial condition:

$$h_0[\psi(\hat{\nu}) \cdot f_{<0}](\nu, g) = h_0[f_{<0}](\nu, g) \quad (\text{by initial cond.}) \quad (77)$$

$$= h_0[f_{<0}](\nu - \hat{\nu}, \psi_{t-1}(\hat{\nu})^{-1} \cdot g) \quad (\text{by constant init.}) \quad (78)$$

1039 Inductive Step: Assuming $h_t[\psi(\hat{\nu}) \cdot f](\nu, g) = h_t[f](\nu - \hat{\nu}, \psi_{t-1}(\hat{\nu})^{-1} \cdot g) \forall \nu \in V, g \in G$, for some
1040 $t \geq 0$, we wish to prove this also holds for $t + 1$:

1041 Using the FERNN recurrence (Eqn. 11) on the transformed input, and letting $H = V \times G$, we get:

$$h_{t+1}[\psi(\hat{\nu}) \cdot f](\nu, g) = \sigma\left(\psi_1(\nu) \cdot [h_t[\psi(\hat{\nu}) \cdot f_{<t}]] \star_H W\right)(\nu, g) + [(\psi_t(\hat{\nu}) \cdot f_t) \hat{\star}_H U](\nu, g) \quad (79)$$

$$(\text{by inductive hyp.}) = \sigma\left(\psi_1(\nu) \cdot [(\psi_{t-1}(\hat{\nu}) \cdot h_t[f_{<t}])] \star_H W\right)(\nu - \hat{\nu}, g) + [(\psi_t(\hat{\nu}) \cdot f_t) \hat{\star}_H U](\nu, g) \quad (80)$$

$$(\text{by trivial inp. lift}) = \sigma\left(\psi_1(\nu) \cdot [(\psi_{t-1}(\hat{\nu}) \cdot h_t[f_{<t}])] \star_H W\right)(\nu - \hat{\nu}, g) + [(\psi_t(\hat{\nu}) \cdot f_t) \hat{\star}_H U](\nu - \hat{\nu}, g) \quad (81)$$

$$(\text{by equiv. flow-conv}) = \sigma\left(\psi_1(\nu) \cdot \psi_{t-1}(\hat{\nu}) \cdot [h_t[f_{<t}]] \star_H W\right)(\nu - \hat{\nu}, g) + \psi_t(\hat{\nu}) \cdot [f_t \hat{\star}_H U](\nu - \hat{\nu}, g) \quad (82)$$

$$(\text{by flow properties}) = \sigma\left(\psi_t(\hat{\nu}) \cdot \psi_1(\nu - \hat{\nu}) [h_t[f_{<t}]] \star_H W\right)(\nu - \hat{\nu}, g) + \psi_t(\hat{\nu}) \cdot [f_t \hat{\star}_H U](\nu - \hat{\nu}, g) \quad (83)$$

$$(\text{by equiv. non-lin.}) = \psi_t(\hat{\nu}) \cdot \sigma\left(\psi_1(\nu - \hat{\nu}) [h_t[f_{<t}]] \star_H W\right)(\nu - \hat{\nu}, g) + [f_t \hat{\star}_H U](\nu - \hat{\nu}, g) \quad (84)$$

$$(\text{by FERNN Eqn. 11}) = \psi_t(\hat{\nu}) \cdot h_{t+1}[f](\nu - \hat{\nu}, g) \quad (85)$$

$$= h_{t+1}[f](\nu - \hat{\nu}, \psi_t(\hat{\nu})^{-1} \cdot g), \quad (86)$$

1042 Thus, assuming the inductive hypothesis for time t implies the desired relation at time $t + 1$; together
1043 with the base case this completes the induction and proves Theorem 4.1 \square

B Experiment Details

In this section we describe the datasets, models, training and evaluation procedures used in §5 of the main text. We additionally include samples from each dataset for visualization. The full code to reproduce these experiments is available at: <https://github.com/Neurips-FERNN/FERNN>

B.1 Flowing MNIST: Dataset Creation

As described in the main text, we construct sequences from the Flowing MNIST dataset by applying a flow generators ν randomly picked from an admissible set V_{train} , V_{val} , & V_{test} to samples from the corresponding train / validation / test split of the original MNIST dataset [LeCun et al., 1998]. The training sequences are always composed of $T = 20$ time-steps, and identically for the test sequences (except in the length-generalization experiments where the test sequence length is increased to 40 time-steps). We similarly generally set $V_{train} = V_{val} = V_{test}$, except in the velocity generalization experiments, where we specify the training and test flows explicitly. Note that the set V which defines the set of flows to which the FERNNs are equivariant can be anything, and must not match the V_{train} of the dataset. Explicitly, denoting the value of the i th MNIST training image at pixel coordinate (x, y) as $m_{train}^{(i)}((x, y))$, a sample from the training set can be written as:

$$f_t^{(i)}((x, y)) = \psi_t(\nu) \cdot m_{train}^{(i)}((x, y)) \quad \forall (x, y) \in \mathbb{Z}^2, t \in \mathbb{Z}_+ \leq T \quad \text{where } \nu \sim V_{train} \quad (87)$$

We note that above, as in the main paper, we have written the signals and the kernels on the infinite domain \mathbb{Z}^2 . In practice, however, both are only non-zero on a small portion of this domain (i.e. between $(0, 0)$ & $(28, 28)$ for MNIST digits). We do this to simplify the analysis, analogous to prior work [Cohen and Welling, 2016a].

Translating MNIST. For the planar 2D-translation group variant of Flowing MNIST, we consider the group

$$G = T(2) = (\mathbb{Z}^2, +), \quad \text{with binary operation} \quad (x, y) + (x', y') = (x + x', y + y'). \quad (88)$$

We use flow generators which are elements of the Lie algebra of this group, $\nu \in \mathfrak{t}(2)$, which we similarly denote as vectors in \mathbb{Z}^2 , with the Lie bracket $[\gamma, \nu] = 0 \quad \forall \gamma, \nu \in \mathfrak{t}(2)$, meaning the translations are commutative. We note that in a matrix representation, we embed the elements of $T(2)$ into the affine group with homogeneous 3×3 matrices:

$$(x, y) \mapsto \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}, \quad (89)$$

where the corresponding Lie algebra elements are

$$X(\nu) = \begin{pmatrix} 0 & 0 & \nu_1 \\ 0 & 0 & \nu_2 \\ 0 & 0 & 0 \end{pmatrix}, \quad \nu = (\nu_1, \nu_2) \in \mathbb{Z}^2. \quad (90)$$

The exponential map is then simply: $\exp(X(\nu)) = I + X(\nu)$. We see that the flow is then given as: $\psi_t(\nu) = \exp(tX(\nu)) = I + tX(\nu)$, and the action of the flow on a given pixel coordinate $g = (x, y)$ is given as:

$$\psi_t(\nu) \cdot g = t \begin{pmatrix} 1 & 0 & \nu_1 \\ 0 & 1 & \nu_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & x + t\nu_1 \\ 0 & 1 & y + t\nu_2 \\ 0 & 0 & 1 \end{pmatrix}, \quad (91)$$

i.e. a shift of the pixel coordinates by velocity ν for t time-steps. In practice, order to be able to generate long sequences without excessively large images, we perform all translations with cyclic boundary conditions on our input and hidden states, i.e. $\psi_t(\nu) \cdot (x, y) = ((x + t\nu_1) \bmod W, (y + t\nu_2) \bmod H)$, for image size (H, W) . Since all of our convolutions are also performed with cyclic boundary conditions, this does not impact performance.

In our experiments, to define the sets of generators that we are interested in, we always use the integer lattice up to velocities $N \frac{\text{pixels}}{\text{step}}$. We denote these sets with the notation:

$$V_N^T := \{\nu \in \mathbb{Z}^2 \mid \|\nu\|_\infty \leq N\}. \quad (92)$$

For example V_2^T is the set of all 2D translation vectors with maximal velocity component ± 2 in either dimension, i.e. $V_2^T = \{(-2, -2), (-2, -1), (-2, 0), \dots, (2, 2)\}$.

1082 **Rotating MNIST.** For the planar rotation variant of Flowing MNIST, we consider the group
 $G = SO(2) = (\mathbb{R}, +)$, with binary operation $\theta + \theta' = (\theta + \theta') \bmod 2\pi$. (93)

1083 Elements of the Lie algebra $\mathfrak{so}(2)$ are one-parameter generators of in-plane rotations, each of the
 1084 form $\nu = \omega J$, where $J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ and $\omega \in \mathbb{Z}$ denotes the (integer-scaled) angular velocity.
 1085 Because $\mathfrak{so}(2)$ is abelian, the Lie bracket $[\gamma, \nu] = 0 \quad \forall \gamma, \nu \in \mathfrak{so}(2)$.

1086 In homogeneous coordinates we embed $SO(2)$ into the affine group with 3×3 matrices:

$$\theta \mapsto \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (94)$$

1087 whose corresponding Lie-algebra elements are

$$X(\nu) = \begin{pmatrix} 0 & -\omega & 0 \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \nu = \omega J, \quad \omega \in \mathbb{Z}. \quad (95)$$

1088 Since $X(\nu)^2 = -\omega^2 I_{2 \times 2}$, the exponential map is the usual matrix exponential for planar rotations:

$$\exp(X(\nu)) = I + \sin(\omega) J + (1 - \cos(\omega)) J^2.$$

1089 Hence the flow generated by ν is $\psi_t(\nu) = \exp(t X(\nu))$, and its action on a pixel coordinate $g = (x, y)$
 1090 (in homogeneous form) is

$$\psi_t(\nu) \cdot g = \begin{pmatrix} \cos(t\omega) & -\sin(t\omega) & 0 \\ \sin(t\omega) & \cos(t\omega) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \cos(t\omega) - y \sin(t\omega) \\ x \sin(t\omega) + y \cos(t\omega) \\ 1 \end{pmatrix}, \quad (96)$$

1091 i.e. a rotation about the image centre by angle $t\omega$.

1092 Following our experimental protocol, we discretize the angular velocity at $\Delta\theta = 10^\circ$ intervals and
 1093 collect the set of generators

$$V_N^R := \{ \nu = k \Delta\theta J \mid k \in \mathbb{Z}, |k| \leq N \}. \quad (97)$$

1094 For instance, $V_4^R = \{-40^\circ, -30^\circ, -20^\circ, -10^\circ, 0^\circ, 10^\circ, 20^\circ, 30^\circ, 40^\circ\}$. In practice, to implement
 1095 the spatial rotation we use the Pytorch function `F.grid_sample` with zero-padding and bilinear
 1096 interpolation. We additionally zero-pad all images with 6-pixels on each side (resulting in images of
 1097 size (40×40)) to allow for the rotation to fit within the full image frame.

1098 B.2 Flowing MNIST: Models

1099 For the Flowing MNIST datasets we compare three types of models: standard group-equivariant
 1100 RNNs (G-RNNs) as defined in Equation 4, FERNs with equivariance to a subset of the training
 1101 flows (i.e. $V_{model} = V_1^T$ and $V_{train} = V_2^T$), and FERNs with full equivariance to the training flows
 1102 ($V_{model} = V_{train}$). For all models on each dataset we use the same model architecture, and since
 1103 there are no extra parameters introduced by the FERN model, all models have the same number of
 1104 trainable parameters.

1105 **Translating MNIST G-RNN.** For the translation group, the corresponding group-convolution is
 1106 the standard 2D convolution. We therefore build a G-RNN exactly as written in Equation 4 with
 1107 regular convolutional layers in place of the group-convolution. We use kernel sizes of 3×3 for
 1108 both \mathcal{U} and \mathcal{W} with no bias terms, strides of 1 and circular padding of 1, resulting in a hidden state
 1109 with spatial dimensions equal to the input spatial dimensions: 28×28 . We use 128 output channels
 1110 for our convolutional ‘encoder’ \mathcal{U} , and similarly 128 input and output channels for our recurrent
 1111 kernel \mathcal{W} . This results in a hidden state $h \in \mathbb{R}^{128 \times 28 \times 28}$. We use a ReLU activation function:
 1112 $\sigma(h) = \max(0, h)$. We initialize all hidden states to zero: $h_0 = \mathbf{0}$, satisfying our equivariance
 1113 proof assumptions. At each timestep, we decode the updated hidden state to predict the next input
 1114 through a 4-layer CNN decoder: $g_\theta(h_{t+1}) = \hat{f}_{t+1}$. The CNN decoder is composed of three primary
 1115 convolutional layers each with 128 input and output channels, kernel size 3×3 , stride 1, padding 1,
 1116 followed by ReLU activations. A final identical convolutional layer, but with 1 output channel, is
 1117 used to predict \hat{f}_{t+1} .

Translating MNIST FERNs. For the FERNs, we use the exact same RNN and decoder architecture, with the only difference being that we extend the hidden state with an extra V dimension for each of the flows in V_{model} : $h(\nu, g)$. Explicitly then, through the trivial lift, the input is copied identically to each of these ν channels, and the flow convolution on the hidden state also operates identically on each ν channel. As stated in the main text, we define the flow convolution to not mix the V channels at all, explicitly: $\mathcal{W}_k^i(\nu, g) = \delta_{\nu=e} \mathcal{W}_k^i(g)$, thus giving us constant parameter count. Our lifted hidden state is thus $h \in \mathbb{R}^{|V_{model}| \times 128 \times 28 \times 28}$. The flow action $\psi_1(\nu) \cdot$ in the recurrence is implemented practically as a Roll of the hidden state tensor by (ν_1, ν_2) steps along the (x, y) spatial dimensions. To achieve invariance of the reconstruction to the input flows, and thereby achieve the generalization we report, we max-pool over the V dimensions before decoding. Explicitly, the output of the model for each time step is computed as: $g_\theta(\max_\nu h_{t+1}(\nu, g)) = \hat{f}_{t+1}$.

Rotating MNIST G-RNN. For the rotating MNIST models, we use a nearly identical setup to the translation experiments, with the only difference being that we use $SE(2)$ group convolutions in place of the standard convolutional layers to achieve the necessary rotation equivariance. In practice, we use the *escnn* library to implement the $SE(2)$ convolutions [Cesa et al., 2022]. We discretize the rotation group into $\Delta\theta = 10^\circ$ rotations, yielding a cyclic group with 36 elements: C_{36} . We lift the input to this space and assert a regular representation of the group action on the output. Due to the increased dimensionality of the hidden state from the lift to the discrete rotation group, we decrease the number of hidden state channels to 32 due to hardware limitations. This then yields a hidden state: $h \in \mathbb{R}^{36 \times 32 \times 40 \times 40}$.

Rotating MNIST FERNs. For the FERN models, we follow the same procedure as for the translating MNIST FERNs, except with the action of the group $\psi_1(\nu) \cdot$ in the recurrence now taking the form of the regular representation of rotation in the $SE(2)$ equivariant CNN output space. Explicitly, this means that in addition to rotating the inputs, we also permute along the lifted rotation channel for each angular velocity ν . Again this yields a hidden state of size: $h \in \mathbb{R}^{|V_{model}| \times 36 \times 32 \times 40 \times 40}$.

B.3 Flowing MNIST: Next-Step Prediction Training & Evaluation

For the in-distribution next step prediction experiments, (Table 1 and Figure 2) of the main text, we use $V_{train} = V_{val} = V_{test} = V_2^T$ for the translating MNIST experiments, and $V_{train} = V_{val} = V_{test} = V_4^R$ for the rotating MNIST experiments. We set the training sequence length to 20 steps, providing the models with 10 time-steps as input, and computing the next-step prediction reconstruction loss (MSE) of the model output on the remaining 10 time-steps. Explicitly:

$$\mathcal{L} = \frac{1}{10} \sum_{t=11}^{20} \|f_t - g_\theta(h_t)\|_2^2 \quad (98)$$

During training, we use ‘teacher forcing’ [Williams and Zipser, 1989] with 50% probability, meaning that at each time-step of the 10 forward prediction steps, the model has a 50% probability of receiving the true input as f_t , and a 50% probability of receiving it’s own autoregressive prediction \hat{f}_t as input. This technique is designed to improve model convergence of autoregressive prediction models by avoiding loss plateaus induced by poor forward prediction. Note that this procedure is not used during validation or testing, ensuring pure autoregressive next-step performance is evaluated.

All models are trained for 50 epochs, with a learning rate of 1×10^{-4} using the Adam optimizer [Kingma and Ba, 2017]. For translation flows, we use a batch size of 128, and clip gradient magnitudes at 1 in all models for additional stability. For rotation flows we use a batch size of 32 due to memory constraints, and find gradient clipping not necessary. For evaluation, we save the model with the best performance on the validation set (over epochs), and report its corresponding performance on the held-out test set. For each model we train with 3 random initializations (3 seeds) and report the mean and standard deviation of the test set performance from the best saved models.

Length Generalization. For the out-of-distribution length generalization experiments (Figure 3) we again use $V_{train} = V_{val} = V_{test} = V_2^T$ for the translating MNIST experiments, and $V_{train} = V_{val} = V_{test} = V_4^R$ for the rotating MNIST experiments. Where the length of training and validation sequences is again set to $T = 20$ as before. At test time, we increase the length of the sequences to

1167 $T = 40$, but continue to only feed the models the first 10 time-steps as input. In Figure 3 we show
 1168 the loss of the model for each of these remaining 30 time steps ahead.

1169 **Velocity Generalization.** For the out-of-distribution velocity generalization experiments (Figure 4),
 1170 we leave the train and test sequence length at 20, but we instead set $V_{train} = V_{val} \subset V_{test}$. Explicitly,
 1171 in Figure 4 for rotating MNIST, we set $V_{train} = V_{val} = V_1^R$ and $V_{test} = V_5^R$. For translating
 1172 MNIST, we set $V_{train} = V_{val} = V_1^T$ and $V_{test} = V_2^T$. This tests the ability of models to generalize
 1173 to new flows not seen during training, and we see that the FERNs perform significantly better on
 1174 this test set when they are made equivariant to these velocities.

1175 B.4 Moving KTH: Datasets

1176 To test the benefits of flow equivariance on a sequence classification task with real image se-
 1177 quences, we opted to use the KTH action recognition dataset [Schuldt et al., 2004], obtained from
 1178 <http://www.csc.kth.se/cvap/actions/>. The dataset is composed of 2391 videos of 25 people
 1179 performing 6 different actions: running, jogging, walking, boxing, hand clapping, and hand waving.
 1180 The original videos are provided at a resolution of 160×120 with 25 frames per second, and an av-
 1181 erage clip length of 4 seconds. The training, validation, and test sets are constructed from this dataset
 1182 by taking the videos from the first 16 people as training, the next 4 people as validation, and the last
 1183 5 people as test. We split the videos into clips of 32-frames each, downsample to a spatial resolution
 1184 of 32×32 , and subsample them by half in time, yielding a final set of clips which are 16 steps long.

1185 Since the videos from this dataset are taken entirely from a stationary camera viewpoint, there are no
 1186 global flows of the input space which our model might benefit from. We call this original dataset
 1187 KTH with V_0^T (no motion). To test the benefits of flow equivariance in an action recognition setting
 1188 with a moving viewpoint, we construct two additional variants of the KTH dataset, augmented by
 1189 translation flows from the sets V_1^T and V_2^T . These sets are identical to those described in the Flowing
 1190 MNIST examples: translation with circular boundary conditions. Again, since we use convolution
 1191 with circular boundary conditions in all our models, this does not impact model performance.

1192 B.5 Moving KTH: Models

1193 We compare five models on the KTH dataset: a 3D-CNN, a G-RNN, two FERN variants, and an
 1194 ablation of the FERN (denoted G-RNN+). We describe these in detail below:

1195 **Moving KTH 3D-CNN.** The spatio-temporal 3D-CNN baseline is built as a sequence of five 3D
 1196 convolution layers, interleaved with 3D batchnorm and ReLU activations. Each layer has kernels of
 1197 shape $(3 \times 3 \times 3)$, no bias, and padding 1. The first layer has 16 output channels, a temporal stride of
 1198 2, and a spatial stride of 1. Layer 2 has 32 output channels, temporal stride of 1, and spatial stride of
 1199 2. Layer 3 has 32 output channels, temporal stride of 1, and spatial stride of 1. Layer 4 has 64 output
 1200 channels, temporal stride of 1, and spatial stride of 2. Layer 5 has 64 output channels, temporal
 1201 stride of 1, and spatial stride of 1. This final layer is followed by a global average pooling over the
 1202 remaining $(8 \times 8 \times 8)$ space-time feature map dimensions, yielding a single vector of dimensionality
 1203 64 which is passed through a linear layer to predict the logits for the 6 classes.

1204 **Moving KTH G-RNN.** For the baseline G-RNN, each grayscale input frame $f_t \in \mathbb{R}^{1 \times 32 \times 32}$ is
 1205 passed through a three-layer convolutional encoder that preserves spatial resolution:

$$\text{Conv}_{5 \times 5}^{1 \rightarrow 32} \xrightarrow{\text{BN+ReLU}} \text{Conv}_{3 \times 3}^{32 \rightarrow 64} \xrightarrow{\text{BN+ReLU}} \text{Conv}_{3 \times 3}^{64 \rightarrow 128} \xrightarrow{\text{BN+ReLU}},$$

1206 all with stride 1 and circular padding chosen so that the hidden state has the same spatial dimensions
 1207 as the input (32×32). The output defines the encoder feature map $\mathcal{U} * f_t$. The hidden state
 1208 $h_t \in \mathbb{R}^{128 \times 32 \times 32}$ is updated with a single recurrent layer using a circularly padded 3×3 convolution
 1209 where $\mathcal{W} \in \mathbb{R}^{128 \times 128 \times 3 \times 3}$ contains no bias terms. The recurrent convolution is also followed by
 1210 a batch-norm and ReLU non-linearity for added expressivity. The non-linearity σ is tanh. Initial
 1211 states are zero, $h_0 = \mathbf{0}$, satisfying the assumptions of our equivariance proof. At the final timestep of
 1212 the sequence ($t = 16$) we take the hidden state of RNN, perform global average pooling to a 1×1
 1213 spatial dimensions, and feed the resulting 128-dimensional vector through a fully-connected layer:

$$g_\theta(h_T) = \text{FC}_{128 \rightarrow 6}(\text{SpatialAvgPool}(h_T)) \in \mathbb{R}^6,$$

1214 producing logits for the six KTH action classes.

Moving KTH FERNNs. For the FERNNs, we use the exact same architecture, but use the trivial lift to lift the corresponding sets of flows V_1^T for FERNN- V_1^T and V_2^T for FERNN- V_2^T (denoted FERNN-1 and FERNN-2 in Figure 5). Concretely, for every translation velocity $\nu \in V_{\text{model}}$ we allocate an additional velocity channel, so that the hidden state becomes $h_t(\nu, g) \in \mathbb{R}^{|V_{\text{model}}| \times 128 \times 32 \times 32}$, where $g = (x, y)$ indexes spatial position. Input frames are trivially lifted by copying the same encoder features into each ν -channel. Following Equation 11, the flow action in the recurrence is implemented again as a Roll of the spatial dimensions of the hidden tensor by (ν_x, ν_y) pixels, and the flow convolution uses weight sharing across velocities, $\mathcal{W}_k^i(\nu, g) = \delta_{\nu=e} \mathcal{W}_k^i(g)$, so the total number of trainable parameters is identical to the G-RNN. As in the Flowing-MNIST experiments, we consider two settings: (i) *partial* equivariance with $V_{\text{model}} = V_1^T \subset V_{\text{train}} = V_2^T$ and (ii) *full* equivariance where $V_{\text{model}} = V_{\text{train}}$. To achieve flow-invariant action classification we take the maximum over the velocity dimension after the final FERNN layer, $\max_{\nu} h_t(\nu, g)$, followed by the same global-average-pooling and linear classifier used for the G-RNN. This design ensures that any translation-induced shifts present at test time are pooled over, yielding the generalization results reported in Figure 5.

Moving KTH G-RNN+. To ensure that the observed performance improvement of the FERNN was not simply due to the increased number of hidden state activations and the associated max-pooling, but instead could be attributed the precise flow equivariant form of the recurrence introduced in Equation 11, we built a third baseline which is as close as possible to the best performing FERNN (FERNN- V_2^T), while removing precise flow equivariance. Specifically, while keeping all other architectural components of the FERNN- V_2^T identical, we replaced the single-step action of the flow in the recurrence ($\psi_1(\nu) \cdot$) with convolution by a separate learned 5×5 convolutional kernel for each ν channel (randomly initialized). Since the action of $\psi_1(\nu)$ is a simple translation (a local linear operation), this can indeed be represented by such a kernel. However, as we see in practice (Figure 5), the model fails to learn such kernels and instead overfits to the training data distribution.

B.6 Moving KTH: Action Recognition Training & Evaluation

Models are trained to minimize the cross entropy loss between the predicted class and the ground truth label using the Adam optimizer. Due to the small dataset size, all models are trained for 500 epochs, with a batch size of 32. We search over learning rates in the set $\{3 \times 10^{-3}, 1 \times 10^{-3}, 3 \times 10^{-4}, 1 \times 10^{-4}\}$, for each model, running three random initialization seeds for each. For each random seed of each hyper-parameter setting, we store the model with the best validation loss. We then pick the best performing model based on the mean value of the best validation loss across all three seeds. We then report in Table 2 the mean test loss of the models (3 seeds) saved at the best validation loss epoch for the best identified learning rate. We generally find the lower learning rates (3×10^{-4}) work better for the RNN models, while the higher learning rates (1×10^{-3}) work better for the CNNs.

B.7 Moving KTH Samples

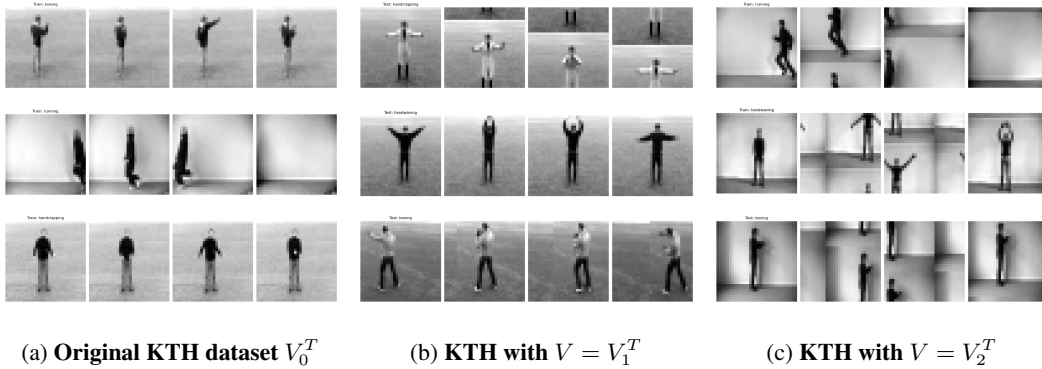


Figure 6: Samples of the original KTH dataset and its two motion-augmented variants.

1251 B.8 Compute

1252 All experiments in this paper were performed on a private cluster containing a mixture of NVIDIA
1253 A100 and H100 GPUs, each having 40GB and 80GB of VRAM respectively. No parallelization of
1254 individual models across GPUs was required, i.e. most models and training paradigms were able to
1255 fit on a single A100 GPU, with the larger models on a single H100 GPU. The cluster nodes allocated
1256 up to 24 CPU cores and 375GB of RAM per job, although only a small fraction of this was required
1257 for training and evaluation. The majority of models were able to train fully in less than 24 hours.
1258 For example, the FERNN- V_2^T models on KTH trained in 7 hours, and the FERNN- V_2^T models on
1259 MNIST trained in 15 hours, with all other models training faster. The significant exception to this
1260 were the FERNN- V_4^R models on Rotating MNIST which took roughly 67 hours to complete 50
1261 epochs (although they converged much more quickly than this, see Figure 2 we ran them to the same
1262 number of epochs as the G-RNN for consistency). The reason for this increased computational time
1263 was an inefficient implementation of the rotation operation and our custom recurrence, which could
1264 both be accelerated in future work. Specifically, we used a naive vanilla Pytorch implementation of
1265 our custom FERNN recurrence (Equation 11) using ‘for loops’, which dramatically slowed down
1266 training for all models. In future work, implementation of the model with a scan operation in JAX, or
1267 a custom CUDA kernel would dramatically improve runtime performance. Overall, we estimate the
1268 computational requirements necessary to develop the models and run all experiments for this paper
1269 totaled approximately 30 days of H100 compute time.

1270 C Variations of FERNNs

1271 C.1 FERNN with Non-Trivial Lift

1272 As noted in §4, it is possible to build a FERNN with a non-trivial lift, such that the lifting convolution
1273 itself incorporates the flow transformation for each ν dimension.

1274 Explicitly, we can define such a lift as:

$$[f_t \star_{V \times G} \mathcal{U}^i](\nu, g) = \sum_{x \in X} \sum_{k=1}^K f_k(x) \mathcal{U}_k^i(g^{-1} \cdot \psi_t(\nu)^{-1} \cdot x) \quad (99)$$

1275 Another way to think of this, is that there is a time-parameterized input kernel defined on the full
1276 space $V \times G$, i.e. $\hat{\mathcal{U}}(\nu, g, t) = \mathcal{U}(\psi_t(\nu)^{-1} \cdot g)$; however, we find this viewpoint less elegant given
1277 that the kernel then depends on time.

1278 We see than when the flow is incorporated into the lifting convolution, the output of the convolution is
1279 no longer constant along the ν index (as it was in the trivial lift). Instead, it is now flowing according
1280 to the ν 'th vector field. Therefore, when processing an input undergoing a given flow $\psi(\hat{\nu})$, this input
1281 flow will combine with the flows of the lifting convolution to yield a shift along the V dimensions,
1282 similar to what we previously observed in the hidden state, i.e.:

$$\begin{aligned} [(\psi_t(\hat{\nu}) \cdot f_t) \star_{V \times G} \mathcal{U}^i](\nu, g) &= \sum_{x \in X} \sum_{k=1}^K f_k(\psi_t(\hat{\nu})^{-1} \cdot x) \mathcal{U}_k^i(g^{-1} \cdot \psi_t(\nu)^{-1} \cdot x) \quad (100) \\ &= \sum_{\hat{x} \in X} \sum_{k=1}^K f_k(\hat{x}) \mathcal{U}_k^i(g^{-1} \cdot \psi_t(\nu - \hat{\nu})^{-1} \cdot \hat{x}) \quad (\text{where } \hat{x} = \psi_t(\hat{\nu})^{-1} \cdot x) \\ &\quad (101) \end{aligned}$$

$$= [f_t \star_{V \times G} \mathcal{U}^i](\nu - \hat{\nu}, g) \quad (102)$$

1283 Notably then, keeping the flow convolution from Equation 9 unchanged, we see that we must
1284 remove the additional $\psi_1(\nu)$ shift from the original FERNN in order to maintain flow equivariance.
1285 Specifically, the new non-trivial-lift recurrence relation is then given simply as:

$$h_{t+1}(\nu, g) = \sigma([h_t \star_{V \times G} \mathcal{W}](\nu, g) + [f_t \star_{V \times G} \mathcal{U}](\nu, g)). \quad (103)$$

1286 In this setting, the action of the flow on the output space changes to just a permutation of the V
1287 dimension, with no corresponding flow on g :

$$(\psi(\hat{\nu}) \cdot h[f])_t(\nu, g) = h_t[f](\nu - \hat{\nu}, g) \quad (104)$$

1288 In a sense, this model can be seen as ‘undoing’ the action of each flow on the input when lifting. We
1289 find this to be somewhat analogous to the traditional group-equivariant CNN design choice where
1290 the transformation can *either* be applied to the filter or the input. In the FERNN setting, the ‘filter’
1291 is now defined by the full recurrence relation, so we can either apply the flow transformation to the
1292 input sequence, or to the hidden state sequence.

1293 Overall, we find this to be a slightly less elegant construction since the indexing of the kernel in the
1294 convolution is then dependent on the time index explicitly. In the trivial-lift setting introduced in the
1295 main text, this time-dependence is rather implicitly imposed by the recurrence of the hidden state
1296 itself, therefore allowing us to only require the instantaneous one-step flows during each recurrent
1297 update. Regardless, we are interested in future work which may explore this non-trivial lift setting
1298 more fully, and other interpretations of the FERNN model as described.

D Related Work

In this section we provide an overview of work which is related to flow equivariance and equivariance with respect to time-parameterized symmetries generally.

D.1 Flow Equivariance without a Hidden State

As mentioned in the main text, it is possible to achieve flow equivariance without an explicitly flow-equivariant sequence model. The two primary methods for accomplishing this are through frame-wise application of an equivariant model (as described in §3) and through group-convolution over the entire space-time block (as described in §6). Both of these methods are verifiably flow-equivariant, however they are fundamentally a different class of model than what we have described in this work. They are not recurrent sequence models, and therefore intrinsically have a finite temporal context or ‘receptive field’ which can be used when computing any output. By contrast, recurrent networks can theoretically have an infinite temporal context, if need be, through the maintenance of a hidden-state. Given this is a fundamental distinction between recurrent and non-recurrent networks which is the subject of research beyond the domains of equivariance, we find it to be beyond the scope of this work to compare with these models explicitly. Instead, we propose flow equivariant RNNs as an extension of existing equivariant network theory to this class of models which maintain a hidden state and can operate in the online recurrent setting.

The list of prior work which can be included in this category of ‘flow equivariance without a hidden state’ is quite broad, since it encompasses most of the equivariant deep learning literature to date, however we list a few notable examples here. The Lorentz equivariant work of [Bogatskiy et al., 2020], [Gong et al., 2022] is the most relevant, while other applied work has developed 3-D convolutional networks which are equivariant with respect to Galilean shifts (our translation flows) in the context of event-based cameras [Zhu et al., 2019], or rotations over time in the context of medical imaging [Zhu et al., 2024]. Early work developed Minkowski CNNs [Choy et al., 2019], which are equivariant with respect to 4D translations, thus making them equivariant to axis-aligned motions. Further, Clifford Steerable CNNs [Zhdanov et al., 2024] have also been developed to achieve Poincaré-equivariance on Minkowski spacetime. Related work on equivariance for PDE solving / forecasting has built dynamics models which are equivariant with respect to galilean transformations in the sense that the model is equivariant if the input vector field has a global additive constant [Wang et al., 2021]. While this is valid for neural networks applied to vector field data as input, it is clearly not the same as our method in more general settings. The method of [Wang et al., 2021] can be interpreted as viewing a dynamical system which has an unknown global current introduced, while ours is better interpreted as viewing the dynamical system from a moving reference frame – the two concepts are compatible and may even be combined.

D.2 ‘Statically Equivariant’ Sequence Models

The second broad category of related work includes sequence models which are equivariant with respect to instantaneous static group transformations, but are not equivariant with respect to time-parameterized group transformations, as our FERNN is. Examples in this category include [Azari and Erdoğan, 2022], [Nguyen et al., 2023], [Basu et al., 2023], which introduce equivariance to transformations such as static rotations in sequence models including RNNs. For example, these models train on one frame of reference and then test on a ‘rotated’ frame of reference. Our work can be seen to generalize these models to instead allow them to be tested on ‘rotating’ frames of reference. This class of prior work can most readily be compared to the group equivariant RNN (G-RNN) we describe in §2. Other researchers have developed sequence to sequence models which are equivariant with respect to fixed permutations and demonstrated that this is beneficial in the context of language modeling [Gordon et al., 2020]. Further, recent work has developed an equivariant sequence autoencoder which uses group convolutions in an LSTM to achieve static equivariance for the purpose of PDE modeling [Fromme et al., 2025].

D.3 Neuroscience and Biologically Inspired Neural Networks

The study of symmetry has also grown increasingly relevant in the computational neuroscience literature. [Zhang et al., 2022] have studied equivariant representations in biological systems in

terms of continuous attractor networks. Interestingly, these models also use convolutional recurrent dynamics; however, again, this work only considers static translations and not motion over time. Related work in the domain of neuroscience has studied how waves can be used to represent motion explicitly [Shaw and Kilpatrick, 2023]. Interestingly, these waves can be seen to be instantiations of the regular representation of the group action that is used in the hidden state of our FERNN model (see [Keller et al., 2024b] for a description of the high level intuition for how equivariance, regular representations, and waves are related). Waves have also been shown to model visual illusions [Barch et al., 2010, Chemla et al., 2018], and some models have even demonstrated bow-wave like effects in neural fields [Glaser and Barch, 1999, Barch and Glaser, 2002], arguing that these effects are potentially beneficial for motion detection. Scientists have also measured doppler shifts in neural responses as a function of moving stimuli [Zhang et al., 2016]. Finally, traveling waves in visual cortex are also known to be induced by movement, such as saccades [Zanos et al., 2015], an effect which we believe holds some relation to the representation of the flow action on our FERNN’s hidden state.

One class of models which is highly related to the FERNN in both theory and implementation comes from a line of biologically inspired work aiming to learn symmetries from data. One of the first models in this category, the Topographic VAE [Keller and Welling, 2022] can be seen as similar to our FERNN but with an explicitly imposed translation flow of a single velocity in the latent space. This makes these models equivariant to input transformations which are isomorphic to the translation group on the integers modulo the capsule length. Interestingly, the authors find that by simply imposing this translation flow in the latent space, the model learns to encode dataset symmetries into these flows in order to better model the dataset. This seems to imply that simply imposing flow symmetries in sequence models without a priori knowledge of the structure of the flow symmetries in the input may still be beneficial. Similar results were shown with the Neural Wave Machine [Keller and Welling, 2023], where flows in the latent space were implemented implicitly through a bias towards traveling wave dynamics. Finally, perhaps most interestingly, related work on traveling waves in simple recurrent neural networks (the wave-RNN) [Keller et al., 2024a] and SSMs [Keller, 2025] has demonstrated that waves implemented through similar ‘roll’ operations have significant benefits for long-term memory in recurrent neural network architectures. Incredibly, these models are identical to translation flow-equivariant RNNs in implementation, but without any mention of equivariance, and applied to an entirely different set of tasks. It is therefore of great interest to study if there is something unique to translation flows which benefit memory performance, or if similar performance benefits may be gained from any latent flow symmetry.

D.4 Reference Frames in Neural Networks

As mentioned in §4, one way to interpret the flow-equivariant RNN is that its hidden state lives in a number of moving reference frames simultaneously (one for each $\nu \in V$). Thus, for moving inputs, the corresponding co-moving hidden state reference frame will see the input as stationary, and process it as normal. This idea of reference frames in neural networks is not new, and significant interesting related work should be noted.

Specifically, Spatial Transformer Networks [Jaderberg et al., 2016], and Recurrent Spatial Transformer Networks [Sønderby et al., 2015] can be seen to predict a frame of reference for a given input and then switch to that reference frame to gain invariance properties. However, these models do not discuss moving reference frames. Other models have been built in this vein with respect to other symmetries, such as polar coordinate networks which are inherently equivariant with respect to scale and rotation [Esteves et al., 2018]. Related in theory is the idea of Capsule Networks [Sabour et al., 2017, Hinton et al., 2018]. These models do have an explicit notion of reference frames, similar to ours, and use this to gain a structured equivariant representation, but again this is defined only in the spatial context.

D.5 Broadly Related Work

More broadly, prior work has looked at the integration of recurrence and motion modeling specifically for vision [Wu et al., 2021, Gehrig and Scaramuzza, 2023]. These models do not have any mention of motion equivariance, and therefore are highly unlikely to provide the strong generalization benefits such as those that we present in this paper. Other work has studied ego motion for action recognition [López-Cifuentes et al., 2020]; and relevant work has looked at equivariance in the context of object tracking [Gupta et al., 2020, Sosnovik et al., 2020a].

1403 D.6 Equivariant Dynamical Systems

1404 In the dynamical systems literature, equivariance is typically defined for autonomous (or homo-
1405 geneous) dynamical systems which have no ‘input’ or ‘driving force’. Abstractly, for a system
1406 $\frac{dx}{dt} = f(x)$, we say the dynamical system is equivariant if $f(g \cdot x) = g \cdot f(x)$ for all $g \in G$. Then
1407 for any $x(t)$ that solves the differential equation, we also know that $g \cdot x(t)$ solves the differential
1408 equation for the full group orbit $g \in G$ [Moehlis and Knobloch, 2007]. Similar to the equivariant
1409 neural network setting, we see that the ‘output’ of an equivariant dynamical system (interpreted as the
1410 particular solution) transforms in a predictable well-behaved manner for a given transformation of the
1411 input. As the simplest example, $\frac{dx}{dt} = x + x^3$, has a sign flip symmetry in x , meaning that if take the
1412 sign flipped version of the trajectory $x(t)$, the solution to the equation also inherits a sign flip. It is
1413 straightforward to see from the above definition that if the function $f(x)$ defining the time derivative
1414 is equivariant with respect to G , then the system is considered equivariant, since the definitions are
1415 equivalent.

1416 In this work, we are interested in recurrent neural networks, which generally operate in the “forced”
1417 or non-autonomous setting. The study of symmetries in non-autonomous dynamical systems has
1418 been previously explored in the control theory literature, and has proven highly valuable for the
1419 design of robust and high performing equivariant filters [Mahony et al., 2022]. In this setting, the
1420 dynamical system is defined as $\frac{dx}{dt} = f(x, u)$ for some driving force u , and the equivariance property
1421 is then defined as $g \cdot f(x, u) = f(g \cdot x, g \cdot u)$. We see that our FERNs indeed are equivariant
1422 non-autonomous dynamical systems by this definition.

1423 The difference of our current work with this prior control-observer work, is that [Mahony et al., 2022]
1424 treats time-parameterised symmetries as known biases inside a hand-written dynamical model and
1425 uses lifts/adjoint operators to keep the estimation error autonomous. The FERNN instead learns
1426 the dynamics, lifts the hidden state to a group-indexed field, and enforces equivariance by a simple
1427 group-convolution weight-sharing rule. This removes the need for handwritten dynamical models and
1428 allows learning flexible non-linear dynamics – capabilities not covered by existing observer theory.

E Extended Results: Flowing MNIST

E.1 In-Distribution Next-Step Prediction (Table 1 & Figure 2)

In Figure 7 we show the sequence predictions of the models presented in Figure 2 & Table 1 trained on Translating MNIST V_2^T and Rotating MNIST V_4^R , and evaluated on the same flows. We see that when tested in-distribution, all models appear to perform well from visual inspection. Full sequences of 20 steps (input and forward prediction steps) are presented, wrapped over 2 lines.

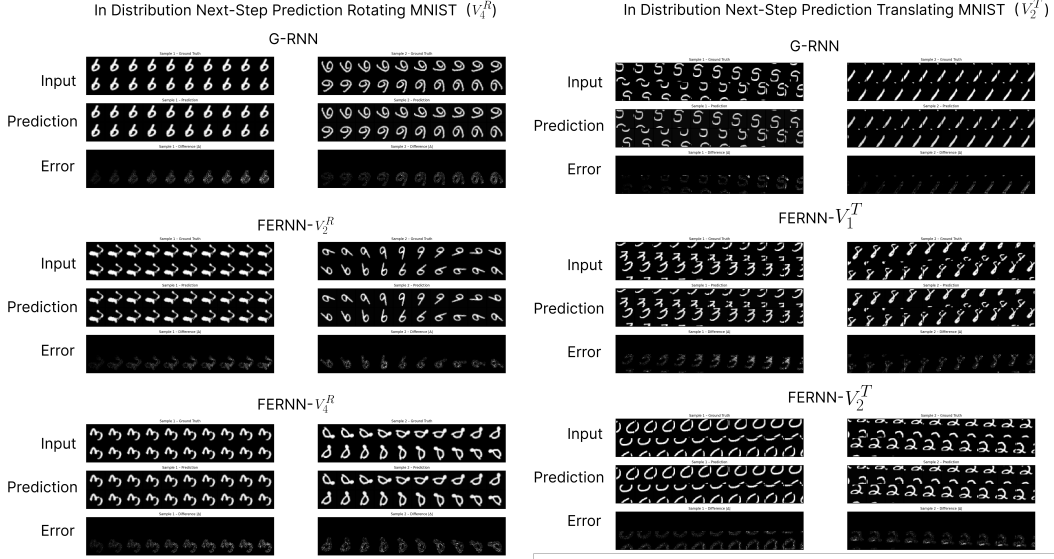


Figure 7: In-distribution sequence predictions for the models from Table 1 & Figure 2, trained on Rotating MNIST V_4^R (left) and Translating MNIST V_2^T (right), evaluated on the same flows.

E.2 Length Generalization Visualizations

In Figure 8 we show the length generalization plot, analogous plot to Figure 3 (right), but for Rotating MNIST. We see that the length generalization performance gap is not as significant on Rotating MNIST compared with Translating MNIST. We suspect that this is due to the accumulation of errors induced by repeated interpolation when performing rotation by small angles on a discrete grid. Despite this, we see that the FERNN- V_4^R still achieves strong generalization up to 20-time steps forward, significantly outperforming the non-equivariant model.

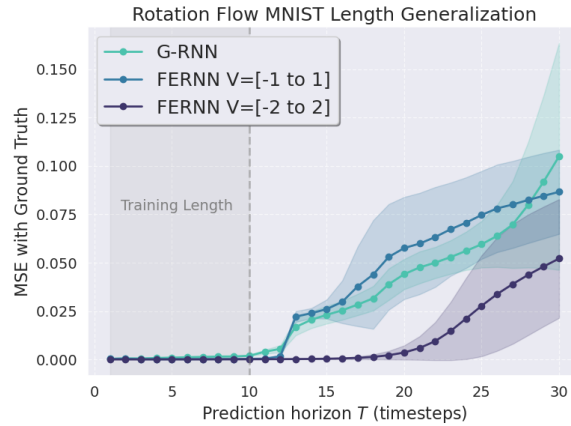


Figure 8: MSE vs. Forward prediction horizon for models on Rotating MNIST. Analogous plot to Figure 3 (right) but for Rotating MNIST.

In Figure 9 we show the sequence predictions of the same models presented in Table 1 & Figure 2 trained on Translating MNIST V_2^T and Rotating MNIST V_4^R , but tested in the length generalization setting. We plot the 30-forward prediction steps here, subsampled by half in time (giving 15 elements). We plot the ground truth sequences on top, and the forward predictions below, with the error (in blue-red color scheme) on bottom. We see the FERNNs significantly outperform the G-RNNs in length generalization on Translating MNIST, and also appear to perform noticeably better on Rotating MNIST, mirroring the quantitative results in Figures 8 & 3.

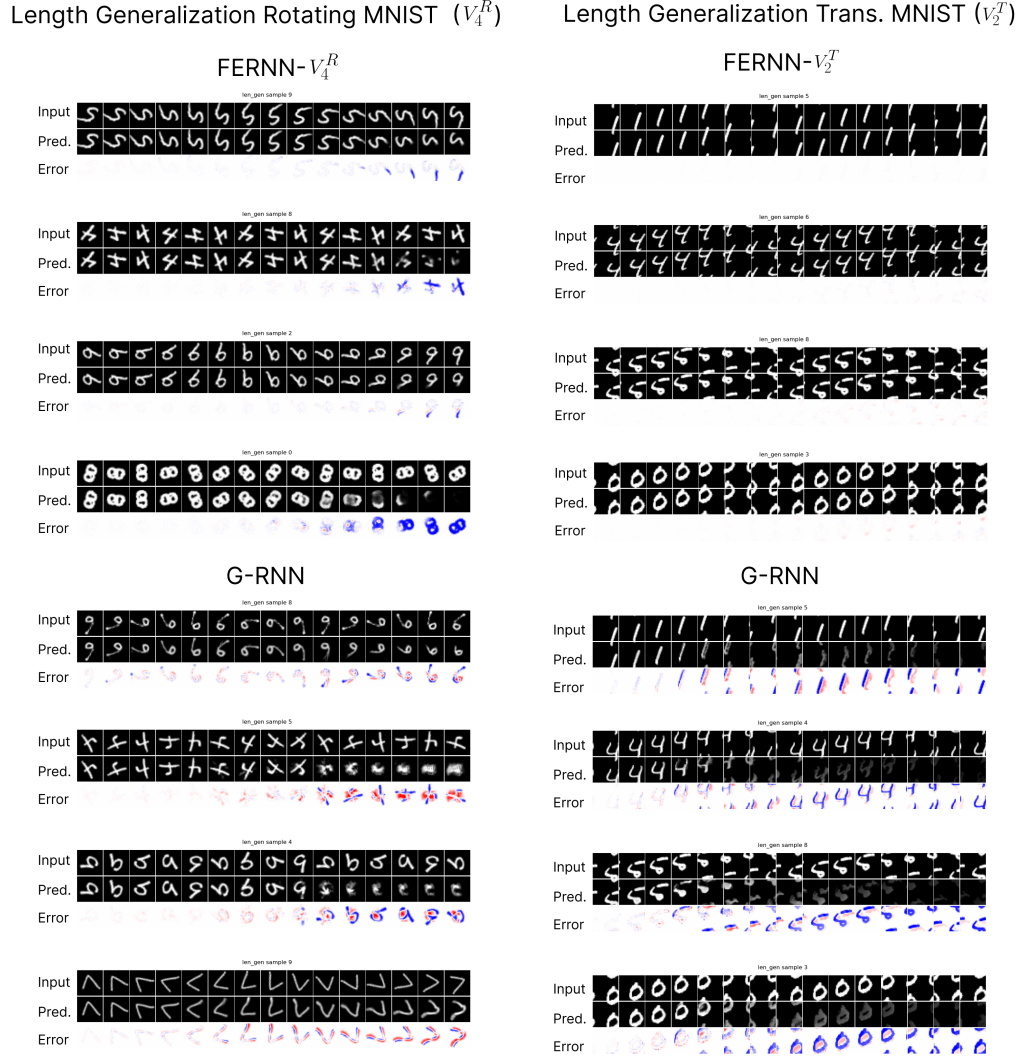


Figure 9: Samples from models trained on Rotating MNIST V_4^R (left) and Translating MNIST V_2^T (right) with training sequence lengths of 20, tested on sequence lengths of 40. We plot the 30-forward prediction steps here, subsampled by half in time (giving 15 elements).

1449 E.3 Velocity Generalization Visualizations

1450 In Figure 10 we show the sequence predictions of the models presented in Figure 4, trained on
 1451 Translating MNIST V_1^T and Rotating MNIST V_1^R , but evaluated on V_2^T and V_4^R respectively.
 1452 These figures show that FERNNs which are equivariant to flows beyond their training distribution are
 1453 able to automatically generalize to these transformations at test time, achieving near perfect next-step
 1454 prediction performance where non-flow equivariant RNNs fail.



Figure 10: Samples from models trained on Rotating MNIST V_1^R (left) and Translating MNIST V_1^T (right), and tested on sequences with significantly higher velocity flows (V_4^R and V_2^T respectively). We see that the FERNN (bottom rows) has no problem generalizing to these new velocities despite never having seen them in training, while the G-RNN (top) fails. The specific flows plotted in this example are $-40 \frac{deg}{step}$ for Rotating MNIST (left), and $\nu = (+2, +1)$ for Translating MNIST.