Unsupervised Learning of Equivariant Structure from Sequences

Takeru Miyato[*1,2] **Masanori Koyama**[*1] **Kenji Fukumizu**[3,1] [*]equal contribution
[1]Preferred Networks, Inc. [2]University of Tübingen [3]The Institute of Statistical Mathematics

Abstract

In this study, we present *meta-sequential prediction* (MSP), an unsupervised framework to learn the symmetry from the time sequence of length at least three. Our method leverages the stationary property (e.g. constant velocity, constant acceleration) of the time sequence to learn the underlying equivariant structure of the dataset by simply training the encoder-decoder model to be able to predict the future observations. We will demonstrate that, with our framework, the hidden disentangled structure of the dataset naturally emerges as a by-product by applying *simultaneous block-diagonalization* to the transition operators in the latent space, the procedure which is commonly used in representation theory to decompose the feature-space based on the type of response to group actions. We will showcase our method from both empirical and theoretical perspectives. Our result suggests that finding a simple structured relation and learning a model with extrapolation capability are two sides of the same coin. The code is available at https://github.com/takerum/meta_sequential_prediction.
.

1 Introduction

The recent evolution and successes of neural networks in machine learning fields have shown the importance of symmetry-aware neural network models [18, 45, 38, 63]. In particular, symmetries in the form of geometric/algebraic constraints have been proven useful in various applications involving high-dimensional, highly-structured observations. For example, recent literature of robotics and reinforcement learning has succeeded in exploiting the knowledge of geometrical symmetries to improve the sample efficiency [62, 65] or to train a model that generalizes to unseen observations [58].

However, building an inductive bias that matches the given dataset of interest is challenging, and recent studies have been exploring the ways to learn symmetries itself from observational sequences. Many of these approaches consider settings with relatively restrictive assumptions or weak supervision. For example, [56] allows the trainer to use the knowledge of the identities of the actions used in making the transition. Meanwhile, [4, 35, 34] essentially assume that the transition velocity of all sequences in the datasets are the same.

These studies indicate that there is still much room left for the question of "what is required for dataset/model to enable the unsupervised learning of the equivariance relation corresponding to the underlying symmetry". This paper advances this investigation by showing that if the sequential dataset consists of time series with a certain stationary property (constant velocity, constant acceleration), we can learn the underlying symmetries by simply training the model to be able to predict the future with linear transition in the latent space. Our theory in Section 3 shows that this strategy can learn a model that is almost equivariant. Moreover, we will experimentally show that, by training an encoder-decoder model in a framework of meta-learning which we call **meta-sequential prediction** (MSP), we can actually learn an equivariant model. In particular, we show that we can

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

learn a hidden equivariance structure in the dataset by splitting (1) the internal training step to compute the prediction loss of linear transition in the latent space from (2) the external training step to update the encoder and decoder. We will also empirically show that, in alignment with group representation theory [42], the learned linear latent transitions in our framework can be simultaneously block-diagonalized, and that each block corresponds to a disentangled factor of variation.

2 Related works

Recently, numerous studies have explored the ways to learn symmetry in a data-driven manner. There is rich literature in unsupervised/weakly supervised approaches that use sequential datasets to exploit the structure that is shared across time, and they all differ by the types of inductive bias. For example, the object-centric approach introduces inductive bias in the form of architecture, and equips the model with pre-defined slots to be allocated to objects [39, 33, 40]. Meanwhile, [22, 1] assumes that the symmetry to be found takes the form of the energy conservation law, and learns each variable in the law as a function of observations. While this approach assumes that some *energy* is preserved in each observation, we assume that the transition parameters like velocity and acceleration are preserved within each observation. Other more indirect forms of inductive bias include those relevant to distributional sparseness and symmetry defined through algebraic constraints. [31] for instance assumed that every stationary component of a given time series is generated by a finite and independent latent time series. [41] proposed to sparsely model the transition with a distribution of large kurtosis. Our work belongs to a family of unsupervised learning that seeks to find the underlying symmetry of the dataset based on an algebraic inductive bias that the transitions can be represented linearly in some latent space. In this sense, our inductive bias also has a connection to Koopman operator [43, 25, 3]. We are different from these studies in that we are aiming to learn a common encoding function (i.e. lifting function) under which the *set* of sequences following different dynamics can be described linearly. Also [70] applied Koopman operator on pedestrian walking sequences, and [23] used Koopman operator to separate the foreground from the background. However, [70] does not set out their model to solve the extrapolation, and neither [70] nor [23] discusses the natural algebraic decomposition of the latent space that results solely from the objective to predict the unseen future.

Unsupervised learning with algebraic/geometrical constraints Many studies impose algebraic constraints that reflect some form of geometrical assumptions. [16] uses a known coordinate map parametrization of a Lie group family to construct a posterior distribution on the manifold. [54] assumes that the observations are dense enough on the data-manifold to describe its tangent space, and exploits a property of random walks on the product manifold to decompose the data space. In the analysis of sequential datasets, [13, 59, 10, 56, 12] make some Lie group type assumptions about the transition. [56] also assumes that the identity of the actions in the sequences is known. As for the approaches with less explicitly geometrical touch, [35] uses capsule structure in their probabilistic framework to model a finitely cyclic structure while retaining the computability of posterior distribution. By design, [35] assumes that all sequences in the dataset transition with the same cyclic velocity. [69] enforces the underlying transition action to be commutative. Some of these studies learn the representation so that the linear transition in the latent space can be explicitly computed [56, 12, 4]. In particular, [4] presents a theory that suggests that a representation without this feature would have topological defects, such as discontinuity. Our approach shares a similar philosophy with these works except that, instead of imposing a strong assumption about the underlying symmetry, we only make a relatively weak stationarity assumption about the dataset; although we assume each sequence to be transitioning with constant velocity/acceleration, we allow the velocity/acceleration to vary across different sequences.

Disentangled Representation Learning Disentangled structure [29, 30] is a form of symmetry that has also been actively studied. It is known that, under the i.i.d assumption of examples, *unsupervised* learning of disentanglement representation is not achievable without some inductive biases encoded in models and datasets [49]. In response to this work, subsequent works have explored different frameworks such as weakly-/semi-supervised settings [51, 50] and learning on sequential examples [41] to learn disentangled representations. For example, PhyDNet [24] disentangles the known physical dynamics from the unknown factors by preparing an explicit module called PhyCell. ICA [32] and recent works [71, 64] also discuss the identifiability property of learned representa-

tions. Classical methods like [28, 6, 36] take an approach of incorporating the inductive bias in the form of a probabilistic model.

We are different from many previous methods in that we do not equip our model with an explicit disentanglement framework. Our method achieves disentanglement as a by-product of training a model that can predict the future linearly in the latent space. The set of latent linear transformations estimated by our method for different time sequences can be simultaneously block-diagonalized, and the latent space of each block corresponds to a disentangled feature. Our data assumption about constant velocity/acceleration might be similar in taste to the setting used by [32], in which the observed time series can be split into the finite number of stationary components.

3 Learning of equivariant structure from stationary time sequences

Our goal is to learn the underlying symmetry structure of a dataset in an unsupervised way that helps us predict the future. What do us humans require for the dataset when we are tasked to, for example, predict where a thrown ball would be in the next second? We hypothesize that we solve such a prediction task by analyzing a short, past time-frame with a certain stationary property (e.g., constant velocity/acceleration). Indeed, people with good dynamic visual acuity can chase a fast-moving object, because they can identify such a short stationary time-frame and use it to predict the near future *linearly* in their latent space. Based on this intuition, we propose to provide the trainer with a dataset consisting of constant velocity/acceleration sequences. We formalize this idea below.

Dataset structure Our dataset $\mathcal{S}$ consists of sequences in some ambient space $\mathcal{X}$, so that each member $\mathbf{s} \in \mathcal{S}$ takes the form $\mathbf{s} = [s_t \in \mathcal{X}; t = 1, ..., T] \in \mathcal{S}$. Because we want $\mathbf{s}$ to be describing a sequence that transitions with constant velocity, we assume that all s_t in a given instance of $\mathbf{s} \in \mathcal{S}$ are related by a fixed transition operation $g \in \mathcal{G}$ so that $s_{t+1} = g \circ s_t$ for all t, where $\mathcal{G}$ is the set of transition operators on $\mathcal{X}$ and each $g \in \mathcal{G}$ acts on $x \in \mathcal{X}$ by sending x to $g \circ x$. We assume that $\mathcal{X}$ is closed under $\mathcal{G}$; that is, $g \circ x \in \mathcal{X}$ for all $x \in \mathcal{X}, g \in \mathcal{G}$. We allow $\mathcal{G}$ to be continuous as well, so that g might not have a finite order (For instance, if g is a rotation with speed $2\pi r$ with irrational r, any finite repetition of g would not agree with identity mapping). This way, our setting is different from those used in [35] that explores a cyclic structure using the capsules of same size. We emphasize that the transition action g is generally assumed to differ across the different members of $\mathcal{S}$. For example, if $\mathcal{G}$ is a set of rotations and $\mathcal{X}$ a set of images, then the rotational speed, direction and the initial image may all be different for any two distinct sequences, $\mathbf{s}$ and $\mathbf{s}'$. Because each instance of $\mathcal{S}$ is characterized by s_1 and g, we may write $\mathbf{s}(s_1, g)$ to denote a sequence that begins with initial frame s_1 and transitions with g. Summarizing, $\mathcal{S}$ is a subset of $\{[g^t \circ s_1; t = 0, ..., T-1]; s_1 \in \mathcal{X}, g \in \mathcal{G}\}$.

Prediction framework through equivariance Our strategy is to exploit the stationary property of each $\mathbf{s} \in \mathcal{S}$ to seek an invertible continuous function $\Phi : \mathcal{X} \to \mathbb{R}^{a \times m}$ such that there exists some $M : \mathcal{G} \to \mathbb{R}^{a \times a}$ satisfying

$$M_g \Phi(x) = \Phi(g \circ x) \text{ for all } x \in \mathcal{X} \text{ and } g \in \mathcal{G}. \tag{1}$$

This relation is known as equivariance [11], and this type of tensorial latent space has also been used in [44] as well for the unsupervised learning of the structure underlying the dataset. In other words, we seek a model in which $\mathcal{X}$ and $\mathbb{R}^{a \times m}$ are invertibly related by an equivariance relation with respect to $\mathcal{G}$, where $g \in \mathcal{G}$ acts on $\Phi(x) \in \mathbb{R}^{a \times m}$ via the map $\Phi(x) \mapsto M_g \Phi(x)$ with $M_g \in \mathbb{R}^{a \times a}$. We assume $m > 1$ in our study[1]. In this framework, we predict the sequence $\mathbf{s}(s_1, g)$ with the relation $\Phi^{-1}(M_g \Phi(s_{t-1})) = \tilde{s}_t$. When $\mathcal{G}$ is a group, (1) would imply $M_{gh}\Phi(x) = \Phi(gh \circ x) = M_g \Phi(h \circ x) = M_g M_h \Phi(x)$ for all x, and the map $g \mapsto M_g$ is called a *representation* of $\mathcal{G}$ [9, 67, 10].

Because we are aiming to establish the framework in which the representation of each action g can be explicitly computed, our philosophy has much in common with the proposition of [4]. This approach is in contrast to [35], which encodes a predefined cyclic structure in the model.

[1] We note that, when $m > 1$, the action of M_g on $\mathbb{R}^{a \times m}$ can be realized by applying the same M_g to m-copies of $\mathbb{R}^a$. In other words, our prediction framework assumes that there are m number of subspaces that react to g in the same way. This $m > 1$ assumption is also considered in [4]. The case in which there are no copies of subspaces that act in the same way is called multiplicity-free in the literature of representation theory [9, 19], and is known to be a special case that happens only under a restrictive condition on the dimension of the observations space [46]. A similar idea has been used in model architecture as well [15].

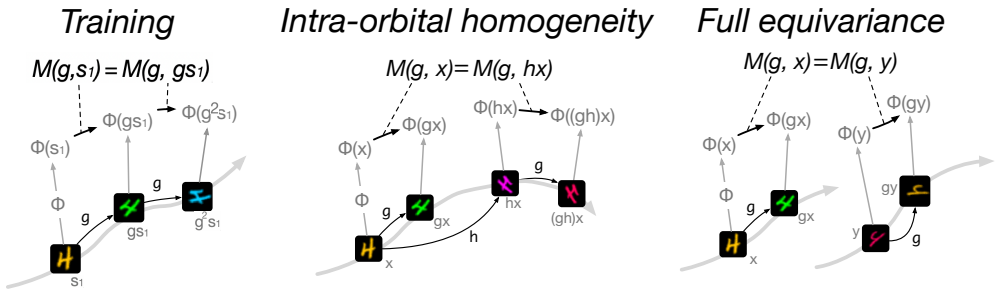


Figure 1: Visualization of intra-orbital homogeneity vs full equivariance. During the training, the model was trained to satisfy $M(g, s_1) = M(g, g \circ s_1)$ for all g and s_1. When intra-orbital homogeneity holds, $M(g, x) = M(g, h \circ x)$ for all $h, g \in G$ and x. When the full equivariance holds, $M(g, x)$ is invariant across different orbits.

3.1 Learning equivariance relation from stationary sequential dataset

However, training the model satisfying (1) with just the *constant velocity assumption* is not a trivial task, because this model assumption only assures that, for each sequence $\mathbf{s}(s_1, g)$, there is a sequence-specific operator $M(g, s_1)$ that is guaranteed only to be able to predict the sequence that transitions with g and begins from s_1 in the way of $M(g, s_1)\Phi(s_t) = \Phi(s_{t+1}) = \Phi(g \circ s_t)$ (the left most panel in Figure 1). In order to satisfy the *full* equivariance ((1) or the right most panel Figure 1), $M(g, s_1)$ shall not depend on s_1 (i.e. homogeneous with respect to s_1). At the same time, because the constant velocity assumption applies to each sequence over all time intervals, it at least assures that the latent transition M is well defined within each sequence; that is, $M(g, x) = M(g, g \circ s_1) = M(g, g^2 \circ s_1) \cdots$ and so on. It turns out that, with some regularity assumptions on the model and the choice of $\mathcal{G}$, we can extend this observation to say that M satisfies **intra-orbital homogeneity** (the middle panel in Figure 1) ; that is, $M(g, x)$ is constant on the orbit $\mathcal{G} \circ x = \{g \circ x; g \in \mathcal{G}\}$ for each x.

Proposition 3.1. *Suppose that $\Phi(s_t) = M(g, s_1)\Phi(s_{t-1})$ for all $\mathbf{s}$ and t. If $m > a$ and if $\mathcal{G}$ is a compact commutative Lie group, then M satisfies intra-orbital homogeneity.*

Also, if M satisfies intra-orbital homogeneity, $M(g, x)$ and $M(g, x')$ for any pair (x, x') in different orbits $\mathcal{G} \circ x \neq \mathcal{G} \circ x'$ can be shown to be at least similar.

Proposition 3.2. *Suppose that $M(g, x)$ satisfies intra-orbital homogeneity, and suppose that $\mathcal{G}$ is a compact connected group. If $M(g, x)$ is continuous with respect x, then for all (x, x'), there exists some P such that $PM(g, x)P^{-1} = M(g, x')$.*

Thus, much of the equivariance property can be satisfied automatically by training the representation on the set of stationary sequences. Interestingly, as we experimentally demonstrate later, our training method in the next section successfully learns a fully equivariant Φ without explicitly enforcing the change of basis P to be I.

3.2 Learning Φ via solving a *meta-sequential prediction* task

We propose a meta-learning way to learn a homeomorphic function $\Phi : \mathcal{X} \to \mathbb{R}^{a \times m}$ with equivariance property by seeking an injective Φ such that $M(g, s_1)\Phi(s_t) = \Phi(s_{t+1})$ for all $g \in \mathcal{G}$, $s_1 \in \mathcal{X}$. We learn such Φ by casting this problem as a meta-learning problem in which $M(g, s_1)$ is to be internally estimated for each $\mathbf{s}$. In other words, we seek a pair of an encoder Φ and a decoder Ψ such that $\mathcal{L}(\Phi, \Psi|\mathbf{s}) = \min_M \sum_{s_t, s_{t+1} \in \mathbf{s}} \|\Psi(M\Phi(s_t)) - s_{t+1}\|_2^2$ is optimized for each $\mathbf{s}$.

We conduct this optimization by splitting each $\mathbf{s} = \{s_1, ..., s_T\}$ into conditional time sequence $\mathbf{s}_\mathrm{c} = \{s_1, ..., s_{T_\mathrm{c}}\}$ and validation time sequence $\mathbf{s}_\mathrm{p} = \{s_{T_\mathrm{c}+1}, ..., s_T\}$, while using the former for the internal optimization of M to force the linear algebraic relation in the latent space and using the latter for the prediction loss. More precisely, we solve the following optimization problem about Φ

and Ψ:

$$\mathcal{L}^p(\Phi, \Psi) := \sum_{\mathbf{s}} \sum_{t=T_c+1}^{T} \left\| \Psi(M^*(\mathbf{s}_c|\Phi)^{t-T_c}\Phi(s_{T_c})) - s_t \right\|_2^2 \\ \text{where } M^*(\mathbf{s}_c|\Phi) = \arg\min_M \sum_{t=1}^{T_c-1} \|M\Phi(s_t) - \Phi(s_{t+1})\|_F^2 . \tag{2}$$

Since M^* is obtained from the latent sequence in the internal optimization, we call this learning framework the **meta-sequential prediction** (MSP). It might appear as if we can also set $\mathbf{s} = \mathbf{s}_c$ and optimize the following reconstruction version of Eq.(2):

$$\mathcal{L}^r(\Phi, \Psi) := \sum_{\mathbf{s}} \sum_{t=2}^{T_c} \|\Psi(M^*(\mathbf{s}|\Phi)^{t-1}\Phi(s_1)) - s_t\|_2^2. \tag{3}$$

However, as we will see in the experiment section, the use of the validation sequence $\mathbf{s}_p$ makes a substantial difference in the learned representation. This is most likely because the minimization of the validation error of M^* on $\mathbf{s}_p$ would encourage Φ to exclude the $\mathbf{s}_c$-specific information from the transition M^*. We will illustrate this effect in the experiment section. We shall note that we can also parameterize M as $M := \exp(A)$, where $A \in \mathbb{R}^{a\times a}$ is a Lie algebra element to be internally optimized. This type of approach was used in [14] for building Lie group convolutional network and in [59, 12] for predicting a sequence that is not necessarily stationary. In order to train their model, [59] used additional parameters to diagonalize each algebra element as well as hyperparameters to stabilize the training. We also experimented with a Lie-algebra style representation of M^* and used SGD to internally optimize the exponent parameters, but we needed to carefully tune the hyper-parameter for the norm regularization term to stabilize the training and never really succeeded to train the model without collapsing. Our internal optimization procedure is free of such a parameter tuning.

Internal Optimization of M^* Because the internal optimization in Eq.(2) is a linear problem, it can be solved analytically as

$$M^*(\mathbf{s}_c|\Phi) = H_{+1}H_{+0}^{\dagger}, \tag{4}$$

where $H_{+0} = [\Phi(s_1); ...; \Phi(s_{T_c-1})] \in \mathbb{R}^{a\times(T_c-1)m}$ and $H_{+1} = [\Phi(s_2); ...; \Phi(s_{T_c})] \in \mathbb{R}^{a\times(T_c-1)m}$ are the horizontal concatenations of the encoded frames and $H_{+0}^{\dagger}$ is the Moore-Penrose pseudo inverse of H_{+0}. Because $M^*(\mathbf{s}_c|\Phi)$ is a closed form with respect to Φ, the loss (2) can be directly optimized by differentiating it with respect to the parameters of both Φ and Ψ. Thus, the training is done in an end-to-end manner. Figure 2 summarizes the overall procedure to make prediction on a given sequence when $T_c = 2, T_p = t$. We note that, although we have assumed the dataset to consist of constant-velocity sequences, we can readily extend our method to the dataset consisting of the time series with higher-order stationarity, such as constant acceleration. See Section 4.4 for the detailed explanation of the model extension and the experimental results.

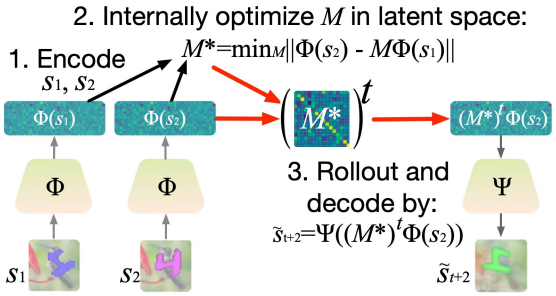


Figure 2: The overview of *meta-sequential prediction* (MSP) when $T_c = 2$ and $T_p = t$. After the encoder encodes the observations into tensor representations $\Phi(s_1), \Phi(s_2)$, the method solves the least square problem: $M^* = \arg\min_M \|M\Phi(s_1) - \Phi(s_2)\|_F^2$. The model then predicts the future observations by $\tilde{s}_{t+2} = \Psi((M^*)^t\Phi(s_2))$. These processes (including the linear problem) are all differentiable.

3.3 Irreducible decomposition of M^*s

Representation theory guarantees that, if $\mathcal{G}$ is a compact connected group, any representation $D : \mathcal{G} \to \mathbb{R}^{a\times a}$ can be simultaneously block-diagonalized; that is, there is a common change of basis U such that $V := UD_gU^{\mathrm{T}} = \bigoplus_j V_g^{(j)}$, where $V_g^{(j)}$ is called irreducible representation that cannot be block-diagonalized any further [42, 10, 67]. The equivariance of our Φ which we show in the experimental section suggests that $M^*(\mathbf{s}|\Phi)$ may be simultaneously block-diagonalizable as well. This block-diagonalization sometimes reveals disentanglement structure because any irreducible representation of $\mathcal{G}_1 \times \mathcal{G}_2$ is of form $V^{(1)} \otimes V^{(2)}$, where $V^{(k)}$ is an irreducible representation of

Figure 3: Simutaneous block diagonalization (SBD) applied to the set of M^*s obtained from 3DShapes sequences. SBD finds the common change of basis under which all matrices $V^* = UM^*U^{-1}$ simultaneously take the form of block diagonal matrices with the same block positions. For clarity, we provide in this figure the visualizations of $\overline{M^*} := M^* - I$ and $\overline{V^*} := V^* - I$ instead of M^* and V^*.

$\mathcal{G}_k$. In particular, if M^*'s irreducible representations have the form $V^{(1)} \otimes 1$ or $1 \otimes V^{(2)}$, then each block would either corresponds to the action of $\mathcal{G}_1$ or of $\mathcal{G}_2$.[2]

To find U that simultaneously block-diagonalizes all $M^*(\mathbf{s}|\Phi)$, we optimized U based on the following objective function that measures the block-ness of $V^*(\mathbf{s}) := UM^*(\mathbf{s}|\Phi)U^{-1}$ based on the normalized graph Laplacian operator Δ:

$$\mathcal{L}_{\mathrm{bd}}(V^*(\mathbf{s})) := \|\Delta(A(V^*(\mathbf{s})))\|_{\mathrm{trace}} = \textstyle\sum_{d=1}^{a} \sigma_d(A(V^*(\mathbf{s}))) \tag{5}$$

where $A(V^*(\mathbf{s})) = abs(V^*(\mathbf{s}))abs(V^*(\mathbf{s}))^{\mathrm{T}}$ with $abs(V^*(\mathbf{s}))$ representing the matrix such that $abs(V^*(\mathbf{s}))_{ij} = |V^*_{ij}|$. Our objective function is based on the fact that, if we are given an adjacency matrix A of a graph, then the number of connected components in the graph can be identified by looking at the rank of the graph Laplacian. For the derivation, please see Appendix E. Through this decomposition, we are able to uncover the hidden block structure of M^*s. See Figure 3 for the actual block-decompositions of M^*s through our simultaneous block diagonalization. We show in Section 4.3 that each block component of V^* with optimized U corresponds to the disentangled factor of variations in dataset.

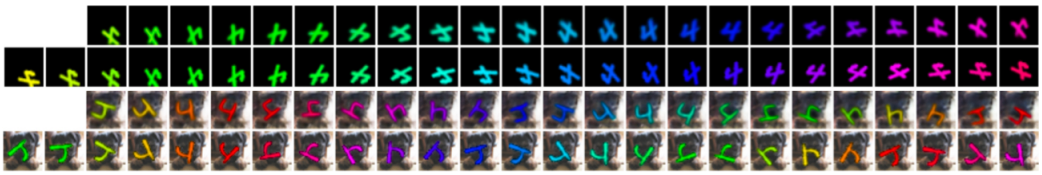
4 Experiments

We conducted several experiments to investigate the efficacy of our framework. In this section, we briefly explain the experimental settings. For more details, please see Appendix D. We tested our framework on *Sequential* MNIST, 3DShapes [5], and SmallNORB [48]. Sequential MNIST is created from MNIST dataset [47]. For all experiments, we used a ResNet [26]-based encoder-decoder architecture and we set $a = 16$ and $m = 256$ so that the latent space lives in $\mathbb{R}^{16\times 256}$.

For Sequential MNIST, we chose our $\mathcal{G}$ to be the set of all combinations of three types of transformations: shape rotation, hue rotation, and translation, and randomly sampled a single instance of $g \in \mathcal{G}$ for each sequence(See Appendix D for the examples of sequences). To create each sequence, we first resized the MNIST image to 24×24, applied repetitions of a randomly sampled, fixed member of $g \in \mathcal{G}$ and embedded the results to 32×32 images. For shape and hue rotations, we randomly sampled the velocity of angles from uniform distribution on the interval $[-\pi/2, \pi/2)$ for each sequence. For translation, we randomly sampled the start point and end point in the range of [-10, 10], and then moved the digit images on a straight line between the sampled points. We also experimented on sequential MNIST with background (Sequential MNIST-bg). For Sequantial MNIST-bg, we used the same generation rule as Sequential MNIST but we added background images behind the moving digits. For the background, we used a randomly sampled images from ImageNet [57], which were all resized to 32×32. Also, we only used the images of digit 4 for most of the experiments on Sequential MNIST/MNIST-bg. Unless otherwise noted, all evaluations in this paper for the Sequential MNIST are based on training with only *digit 4*.

3DShapes and SmallNORB are datasets with multiple factors of variation. We created a set of *constant-velocity* sequences from these datasets by varying a fixed combination of factors for each sequence. That is, on these datasets, we chose our $\mathcal{G}$ to be the set of variations of factors, and sampled each g as $\prod_i g_i^{\ell_i}$, where g_i represents the increase of i-th factor by one unit and $\ell_i \in \mathbb{Z}$. Thus, the value of ℓ_i represents the velocity in the direction of the i-th factor on the grid. For 3DShapes, we chose *wall hue*, *floor hue*, *object hue*, *scale*, and *orientation* as the factors to vary. We varied *elevation* and *azimuth* for SmallNORB. For the split of each sequence into $(\mathbf{s}_{\mathrm{c}}, \mathbf{s}_{\mathrm{p}})$, we set $T_{\mathrm{c}} = 2$ and $T_{\mathrm{p}} = 1$ on all of the constant velocity experiments.

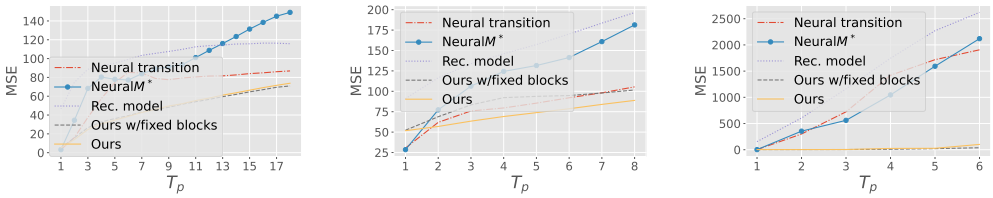
[2] $V : \mathcal{G} \to \{1\}$ is a valid irreducible representation for any $\mathcal{G}$.



(a) Generated examples with our proposed method on Sequential MNIST and MNIST-bg

(b) Comparison of variant methods

Figure 4: (a): Predictions made by *meta-sequential prediction* on Sequential MNIST and MNIST-bg. The ground truth sequence is placed below the predictions, with the first two images representing $\mathbf{s}_{\mathrm{c}}$. (b): Typical failure examples generated by the comparative methods. (1)(2)(3) and (4) are NeuralM^*, Neural transition, Rec. model and Ours w/ fixed block, respectively. See Appendix B for more examples.



(a) Sequential MNIST (b) Sequential MNIST-bg (c) 3DShapes

Figure 5: Prediction errors $\mathcal{L}^p$ with $T_{\mathrm{c}} = 2$ and $T_{\mathrm{p}} = 1, \ldots, 18$. During the training phase, models are trained to predict the observations only at $T_{\mathrm{p}} = 1$. The prediction errors at $T_{\mathrm{p}} > 1$ indicate the extrapolation performance. The results on SmallNORB can be found in Appnedix A.

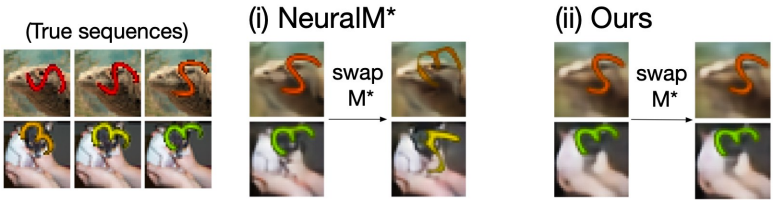
We also conducted experiments for the sequences with constant acceleration on Sequential MNIST. To create a sequence with constant acceleration, we chose a pair $g_a, g_v \in \mathcal{G}$ for each sequence, and generated $\mathbf{s}$ by setting $s_{t+1} = g_a^t g_v s_t$. We elaborate on the detail of this extension in 4.4.

As ablations, we tested several variants of our method: **fixed 2x2 blocks** (abbreviated as fixed blocks), **Neural**M^*, **Reconstruction model** (abbreviated as Rec. Model), and **Neural transition**. For the method of *fixed 2x2 blocks*, we separated the latent tensor $\Phi(s) \in \mathbb{R}^{16\times 256}$ into 8 subtensors $\{\Phi^{(k)}(s) \in \mathbb{R}^{2\times 256}\}_{k=1}^{8}$ and calculated pseudo inverse for each k to compute the transition in each $\mathbb{R}^{2\times 256}$ dimensional space. This variant yields M^* as a direct sum of eight 2×2 matrices. We tested this variant to see the effect of introducing a predetermined representation theoretic structure as in [10]. For Rec. model, we trained Φ and Ψ based on $\mathcal{L}^r$ in Eq. 3 with $T_{\mathrm{c}} = 3$. We tested this variant to see the effect of our use of T_p. For NeuralM^*, we trained an additional network M_θ that maps $\mathbf{s}_{\mathrm{c}}$ to a transition matrix, replaced M^* with M_θ in (4), and optimized θ and (Φ, Ψ) simultaneously. We may see NeuralM^* as a variant in which the *meta* part of the internal and external training is removed from our method. For Neural transition, we trained 1x1 1D-convolutional networks to be applied to latent sequences *in the past* to produce the latent tensor in the next time step; for instance, $\tilde{s}_{t+1} = \Psi(1\mathrm{DCNN}(\Phi(\mathrm{s_t}), \Phi(\mathrm{s_{t-1}})))$ when $T_{\mathrm{c}} = 2$ [3]. The 1DCNN was applied along the multiplicity dimension m. In this variant, the relation between $\Phi(s_t)$ and $\Phi(s_{t+1})$ is not necessarily linear. Section D.1 in Appendix describes each of the comparison methods more in detail. In testing all of these variants, we used the same pair of encoder and decoder architecture as the proposed method.

4.1 Qualitative and quantitative results on the prediction

Figure 4 shows the example sequences generated by the proposed model and comparative models. Figure 5 presents the prediction performance at $T_{\mathrm{c}} + T_{\mathrm{p}}$ when $T_{\mathrm{c}} = 2$. To produce this result, we

[3] This model can be seen as a simplified version of [60]



(a) Dependency of M^* on sequence. The 2x3 tiled images in the leftmost panel represents two sequences $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}$ with the same transition action g. We consider the effects of $M^{*(1)}$ and $M^{*(2)}$ inferred respectively from $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$. NeuralM^* fails in prediction when $M^{*(2)}$ is used to predict $\mathbf{s}^{(1)}$. Our method does not fail by this swap, indicating $M^{*(2)} \cong M^{*(1)}$.

	MNIST		MNIST-bg		3DShapes		SmallNORB	
Method	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$
Rec. Model	48.91	64.22	87.05	95.66	153.39	258.20	57.01	78.13
NeuralM^*	4.99	64.25	20.60	83.18	2.09	217.73	28.98	53.24
MSP (Ours)	6.42	**15.91**	27.38	**36.41**	2.74	**2.87**	31.14	**44.77**

(b) Equivariance performance based on $\mathcal{L}^p$(Eq.(2)) and $\mathcal{L}^p_{\text{equiv}}$(Eq.(6)) with $T_{\text{c}} = 2$ and $T_{\text{p}} = 1$. To evaluate equivariance errors on more difficult settings, we used all of digits in Sequential MNIST-bg for both training and test sets.

Figure 6: Quantitative and qualitative evaluation of learned equivariance.

back-propagated the prediction error at $T_{\text{p}} = 1$ to the encoder during the training, and the prediction at $T_{\text{p}} > 1$ was used to evaluate the extrapolation performance. Our method successfully predicts the images for $T_{\text{p}} \geq 1$. Neural transition and NeuralM^* had almost the same prediction performance at $T_{\text{p}} = 1$, but they both failed in extrapolation. Our *fixed 2×2 blocks* variant failed in extrapolation as well. This might be because the over-regularized structure of 2x2 block hindered with the training of the SGD optimization [17].

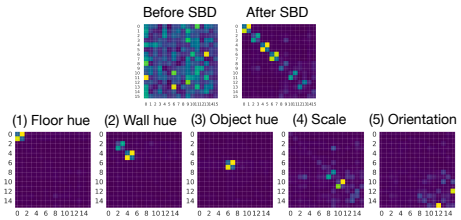
To evaluate how our learned representation relates to the structural features in the dataset, we also regressed the factors of transition from M^* and regressed the class of the digits from $\Phi(s_1)$ (Figures 10 and 11 in Appendix A). SimCLR [8] and contrastive predictive coding (CPC) [61, 27] are tested as baselines. Please see Appendix D for the detailed experimental settings for SimCLR and CPC. Our method yields the representation with better prediction performance than the comparative methods on the test datasets.

4.2 Equivariance performance

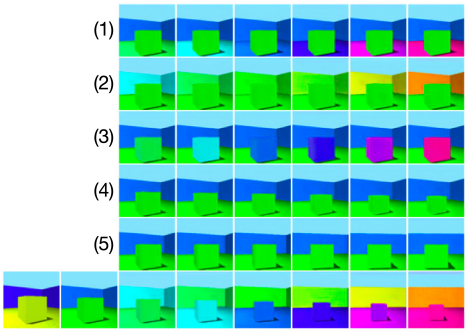
As we have described through Section 3.1 and 3.2, the equivariance is achieved when $M^*(\mathbf{s}(s_1, g)|\Phi)$ in (2) does not depend on s_1, where we recall that $\mathbf{s}(s_1, g)$ represents the sequence that begins with s_1 and transitions with g. To see how much the trained model is equivariant to the transformations in the sequential dataset, we therefore calculated the *equivariance error*, which is the prediction error from applying $M^*(\mathbf{s}|\Phi)$ to $\Phi(s'_{T_c})$ for a pair $\mathbf{s} \neq \mathbf{s}'$ that transitions with the same g. In other words, when $T_c = 2$, we compute the following;

$$\mathcal{L}^p_{\text{equiv}} := \mathbb{E}_g \mathbb{E}_{\mathbf{s},\mathbf{s}' \in \mathcal{S}(g)} [\| \Psi(M^*(\mathbf{s}|\Phi)\ \Phi(s'_2)) - s'_3 \|_2^2] \tag{6}$$

where $\mathcal{S}(g)$ represents the set of all sequences that transition with g. For each pair of $\mathbf{s} \neq \mathbf{s}'$ we set $T_{\text{c}} = 2$ and $T_{\text{p}} = 1$ as done in the experiments in the previous sections. Table 6b compares the NeuralM^* method against our method in terms of the equivariance error. Figure 6a shows the result of applying $M^*(\mathbf{s}|\Phi)$ on $\Phi(s'_2)$ and applying $M^*(\mathbf{s}'|\Phi)$ on $\Phi(s_2)$. We see that when we swap M^* this way, NeuralM^* also swaps the digits; this implies that M^* learned by NeuralM^* encodes the *sequence* specific information together with the transition. On the other hand, swapping of M^* does not affect the prediction for our method, suggesting that our method is succeeding to learn an equivariant model. This is somewhat surprising, because our model does not have the explicit mechanism to enforce the full equivariance ($P = I$ in Section 3.1).



(a) Simeltaneously block-diagonalized matrices



(b) Disentangled transition representation on 3DShapes

Figure 7: (a) Simultaneous block-diagonalization (SBD) of M^*. The top right matrix is the visualization of $abs(V^* - I)$ averaged over all of the training sequences. Each of the five matrices below is the visualization of $abs(V^* - I)$ averaged over the set of sequences on which only a single factor was varied. Coordinates are permuted for better visibility. (b) Sequences generated by applying the transformation of just one block. To produce the disentangled sequences in each row from the leftmost two images in the bottom row, we performed the internal optimization of M^* while setting *all but the block positions corresponding to each factor of variation* to be identity. We elaborate this result and the results for Sequential MNIST, MNIST-bg and SmallNORB in Appendix A.

4.3 Structures found by simultaneous block-diagonalization of M^*s

We have seen in the previous section that the trained Φ is fully equivariant to transformations $\mathcal{G}$, which implies each M^* is a representation of the corresponding transformation of $g \in \mathcal{G}$. As we describe in Section 3.3, we apply simultaneous block-diaognalization to uncover *the symmetry structure* captured by M^*s. Figure 7a shows the structure revealed by simultaneous block-diagonalization through the change of basis U trained by minimizing the average of $\mathcal{L}_{\mathrm{bd}}$ in eq.(5) over all $\mathbf{s}$. Figure 7b shows the results of applying transformation of only one block. We can see that each block only alters one factor of variation. Our results suggest that the learned M^* captures the hidden disentangled structure of the group actions behind the datasets.

4.4 Extension to the sequences with constant acceleration

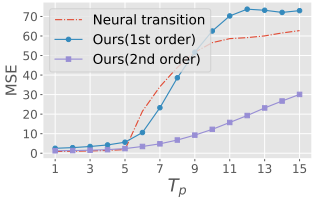
We have seen that *meta-sequential prediction* successfully learns an equivariant structure from the set of constant-velocity sequences. In this section, we show that we can extend our concept to the set of sequences sharing the stationarity of higher order (constant acceleration). By definition, the pair of $\Phi(s_t)$ and $\Phi(s_{t-1})$ encodes the information about the velocity at t. When the multiplicity m is sufficiently large[4], the velocity can be estimated by: ${}^1M_t = \Phi(s_t)\Phi(s_{t-1})^\dagger$. Because this would yield a sequence of velocities, we can simply apply our method again to estimate the constant acceleration by ${}^2M^* = {}^1M_{+1}{}^1M_{+0}^\dagger$ where ${}^1M_{+0} = [{}^1M_2; ...; {}^1M_{T_\mathrm{c}-1}] \in \mathbb{R}^{a\times(T_\mathrm{c}-2)a}$ ${}^1M_{+1} = [{}^1M_3; ...; {}^1M_{T_\mathrm{c}}] \in \mathbb{R}^{a\times(T_\mathrm{c}-2)a}$. We can then predict the future representation $\tilde{s}_t$ for $t = T_\mathrm{c} + 1, ..., T_\mathrm{p}$ by

$$\tilde{s}_t = \Psi\left(\left(\prod_{t'=T_\mathrm{c}+1}^{t} {}^1\tilde{M}_{t'}\right)\Phi(s_{T_\mathrm{c}})\right) \text{ where } {}^1\tilde{M}_{t'} = {}^2M^{*(t'-T_\mathrm{c})}\,{}^1M_{T_\mathrm{c}}. \tag{7}$$

where $\prod$ represents multiplications from left. We train Φ and Ψ by minimizing the mean squared error between $\tilde{s}_t$ and s_t for $t = T_\mathrm{c} + 1, ..., T$ as in Eq.(2). To create a sequence of constant acceleration from MNIST dataset, we only used shape and color rotations. We chose the initial velocity for these rotations randomly on the interval $[-\pi/5, \pi/5)$ for each sequence, and chose the acceleration on the interval $[-\pi/40, \pi/40)$. The results are shown in Figure 8. The Neural transition and the constant-velocity version of our method failed to predict the accelerated sequence, while the 2nd order model succeeded in predicting the sequence even after $T_\mathrm{p} > 5$. Also, Figure 15f and Table 16c

[4]If m is less than a, we cannot obtain the pseudo inverse because of the rank deficient in $\Phi(s_{t-1})\Phi(s_{t-1})^\mathrm{T}$. Thus m should be at least larger than a.

(a) Generated images on *accelerated* Sequential MNIST.



(b) Prediction error.

Figure 8: Results on accelerated Sequential MNIST. Every model was trained with $T_c = 5$, $T_p = 5$. Neural transition overfitted and collapsed from $T_p = 5$ (beyond the training horizon).

in Appendix shows that the accelerated version of our proposed model again achieves learning the equivariance relation.

5 Discussion & Limitations

How is full equivariance achieved in our method? The theoretical results we provided in section 3.1 only assure that $M(g, x)$ and $M(g, x')$ are similar when the underlying group is commutative, compact and connected. However, as we have shown experimentally, our method seems to be learning Φ for which the estimators of M satisfy $M^*(\mathbf{s}(g, x)|\Phi) \cong M^*(\mathbf{s}(g, x')|\Phi)$. This can be happening because our framework and the training method based on the internal optimization in the latent space is somehow encouraging M^* to be orthogonal (See the loss curve of orthogonality of M^* in Appendix A). Maybe this is forcing the change of matrix P such that $PM(g, x)P^{-1} = M(g, x')$ to be also rotations as well, which commutes with $M(g, x)$ itself. Also, Figure 4b and 5 show that, as reported in [34], the models trained with reconstruction loss like (3) does not well capture the group transformation behind the sequences: the encoder representation was found to be significantly worse than that of the model trained with (2). We hypothesize that (3) fails to remove the sequence-specific information from M^*, while (3) succeeds to do so by training the model to be able to predict the unseen images.

Towards learning symmetries from more realistic observations As we are making a connection between our prediction framework and group equivariance, we are essentially assuming that the transitions are always invertible, because group is closed under inversion. However, this might not be always the case in real world applications; for instance, if the image sequences are the sequential renderings of a rotating 3D object, the transitions are generally not invertible because only a part of the object is visible at each time step. We experimented Sequential ShapeNet, which is created from ShapeNet [7] dataset. A series of rendered images is generated by sequentially applying 3D rotations of different speeds for each axis. Generated results on Sequential ShapeNet (See 26 in Appendix B show that actually our current method was not able to generate the images on 3D rotated datasets. If the transitions are not invertible, some measures must be taken in order to resolve the indeterminacy, such as probabilistic modeling or additional structural inductive bias.

Broader impact Because our study generally contributes to predictions and extrapolation, it has as much potential to negatively affect the society as most other prediction methods. In particular, applications of our method to image sequence can be potentially integrated into weapon systems, for example. At the same time, our unsupervised learning of the symmetrical structure from sequential datasets may also contribute to new discoveries in the systems of finance, medical science, physics and other fields of ML such as reinforcement learning.

References

[1] F. Alet, D. Doblar, A. Zhou, J. Tenenbaum, K. Kawaguchi, and C. Finn. Noether networks: meta-learning useful conserved quantities. *NeurIPS*, pages 16384–16397, 2021.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] P. Bevanda, S. Sosnowski, and S. Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

[4] D. Bouchacourt, M. Ibrahim, and S. Deny. Addressing the topological defects of disentanglement via distributed operators. *arXiv preprint arXiv:2102.05623*, 2021.

[5] C. Burgess and H. Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

[6] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. In *NeurIPS Workshop of Learning Disentangled Features*, 2017.

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. URL `https://shapenet.org/terms`.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *1597-1607*, 2020.

[9] M. Clausen and U. Baum. *Fast Fourier Transforms*. Wissenschaftsverlag, 1993.

[10] T. Cohen and M. Welling. Learning the irreducible representations of commutative lie groups. In *ICML*, pages 1755–1763, 2014.

[11] T. Cohen and M. Welling. Group equivariant convolutional networks. In *ICML*, pages 2990–2999, 2016.

[12] M. Connor and C. Rozell. Representing closed transformation paths in encoded network latent space. In *AAAI*, pages 3666–3675, 2020.

[13] B. Culpepper and B. Olshausen. Learning transport operators for image manifolds. *NeurIPS*, 22:423–431, 2009.

[14] N. Dehmamy, R. Walters, Y. Liu, D. Wang, and R. Yu. Automatic symmetry discovery with lie algebra convolutional network. *NeurIPS*, pages 2503–2515, 2021.

[15] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *ICCV*, pages 12180–12189, 2021.

[16] L. Falorsi, P. de Haan, T. R. Davidson, N. De Cao, M. Weiler, P. Forré, and T. S. Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

[17] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

[18] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[19] W. Fulton and J. Harris. *Representation Theory: A First Course*, volume 129. Springer Science & Business Media, 1991.

[20] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, pages 315–323, 2011.

[21] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, T. Kipf, A. Kundu, D. Lagun, I. Laradji, H.-T. D. Liu, H. Meyer, Y. Miao, D. Nowrouzezahrai, C. Oztireli, E. Pot, N. Radwan, D. Rebain, S. Sabour, M. S. M. Sajjadi, M. Sela, V. Sitzmann, A. Stone, D. Sun, S. Vora, Z. Wang, T. Wu, K. M. Yi, F. Zhong, and A. Tagliasacchi. Kubric: a scalable dataset generator. In *CVPR*, 2022.

[22] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *NeurIPS*, pages 15353–15363, 2019.

[23] J. Grosek and J. N. Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*, 2014.

[24] V. L. Guen and N. Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *CVPR*, pages 11474–11484, 2020.

[25] Y. Han, W. Hao, and U. Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[27] O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. Data-efficient image recognition with contrastive predictive coding. In *ICML*, pages 4182–4192, 2020.

[28] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[29] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[30] I. Higgins, S. Racanière, and D. Rezende. Symmetry-based representations for artificial and biological general intelligence. *Frontiers in Computational Neuroscience*, page 28, 2022.

[31] A. Hyvarinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *NeurIPS*, 2016.

[32] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[33] R. Kabra, D. Zoran, G. Erdogan, L. Matthey, A. Creswell, M. Botvinick, A. Lerchner, and C. Burgess. Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. In *NeurIPS*, pages 20146–20159, 2021.

[34] T. Keller and M. Welling. Predictive coding with topographic variational autoencoders. In *ICCV workshop*, pages 1086–1091, 2021.

[35] T. Keller and M. Welling. Topographic vaes learn equivariant capsules. In *NeurIPS*, pages 28585–28597, 2021.

[36] H. Kim and A. Mnih. Disentangling by factorising. In *ICML*, pages 2649–2658, 2018.

[37] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[38] T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2016.

[39] T. Kipf, E. van der Pol, and M. Welling. Contrastive learning of structured world models. In *ICLR*, 2020.

[40] T. Kipf, G. F. Elsayed, A. Mahendran, A. Stone, S. Sabour, G. Heigold, R. Jonschkowski, A. Dosovitskiy, and K. Greff. Conditional object-centric learning from video. In *ICLR*, 2021.

[41] D. Klindt, L. Schott, Y. Sharma, I. Ustyuzhaninov, W. Brendel, M. Bethge, and D. Paiton. Towards nonlinear disentanglement in natural data with temporal sparse coding. In *ICLR*, 2021.

[42] I. R. Kondor. *Group theoretical methods in machine learning*. Columbia University, 2008.

[43] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

[44] M. Koyama, T. Miyato, and K. Fukumizu. Invariance-adapted decomposition and lasso-type contrastive learning. In *ICML 2022 Workshop on Topology, Algebra, and Geometry in Machine Learning*, 2022.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25, 2012.

[46] A. S. Leahy. A classification of multiplicity free representations. *Journal of Lie Theory*, 8:367–391, 1998.

[47] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. URL `http://yann.lecun.com/exdb/mnist/License`.

[48] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004. URL `https://cs.nyu.edu/~ylclab/data/norb-v1.0-small/`.

[49] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, pages 4114–4124, 2019.

[50] F. Locatello, B. Poole, G. Rätsch, B. Schölkopf, O. Bachem, and M. Tschannen. Weakly-supervised disentanglement without compromises. In *ICML*, pages 6348–6359, 2020.

[51] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem. Disentangling factors of variation using few labels. In *ICLR*, 2020.

[52] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[53] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[54] D. Pfau, I. Higgins, A. Botev, and S. Racanière. Disentangling by subspace diffusion. In *NeurIPS*, pages 17403–17415, 2020.

[55] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

[56] R. Quessard, T. Barrett, and W. Clements. Learning disentangled representations and group structure of dynamical environments. *NeurIPS*, pages 19727–19737, 2020.

[57] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[58] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. *arXiv preprint arXiv:2112.05124*, 2021.

[59] J. Sohl-Dickstein, C. M. Wang, and B. A. Olshausen. An unsupervised algorithm for learning lie group transformations. *arXiv preprint arXiv:1001.1027*, 2010.

[60] A. Van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[61] A. Van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, 2018.

[62] E. van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. In *NeurIPS*, pages 4199–4210, 2020.

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[64] J. Von Kügelgen, Y. Sharma, L. Gresele, W. Brendel, B. Schölkopf, M. Besserve, and F. Locatello. Self-supervised learning with data augmentations provably isolates content from style. *NeurIPS*, 34, 2021.

[65] D. Wang, R. Walters, and R. Platt. So(2) -equivariant reinforcement learning. *arXiv preprint arXiv:2203.04439*, 2022.

[66] Z. Wang and J.-C. Liu. Translating math formula images to latex sequences using deep neural networks with sequence-level training. *International Journal on Document Analysis and Recognition (IJDAR)*, 24 (1):63–75, 2021.

[67] S. H. Weintraub. *Representation Theory of Finite Groups: Algebra and Arithmetic*, volume 59. American Mathematical Society, 2003.

[68] Y. Wu and K. He. Group normalization. In *ECCV*, pages 3–19, 2018.

[69] T. Yang, X. Ren, Y. Wang, W. Zeng, and N. Zheng. Towards building a group-based unsupervised representation disentanglement framework. In *ICLR*, 2021.

[70] S. Zhang, Y. Wang, and A. Li. Cross-view gait recognition with deep universal linear embeddings. In *CVPR*, pages 9095–9104, 2021.

[71] R. S. Zimmermann, Y. Sharma, S. Schneider, M. Bethge, and W. Brendel. Contrastive learning inverts the data generating process. In *ICML*, pages 12979–12990, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The abstract and introduction reflects the contributions of this paper.
   (b) Did you describe the limitations of your work? [Yes] In the Discussion & Limitations section, we discuss the type of dataset to which our work cannot be directly applied, as well as the theoretical problem that has not been solved.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We included the broader impact discussion in Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We read the guidelines and we confirmed that the contents in our paper conformed to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We included the full set of assumptions. We provide the details of the assumptions in the supplementary material.
   (b) Did you include complete proofs of all theoretical results? [Yes] We provide complete proofs of theoretical results in Supplementary material.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provided code for reproducing the experimental results as a supplementary material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We explained how we chose the hyperparameters and how we split the dataset into training and test sets in section 4 and appendix section D.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We include the standard deviation values of equivariance performance in Appendix A (We omitted them in the main submission because of the space limitation) and the linear classification performance of the transition parameters in Appendix A.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We put the information of GPUs which we used to train the models. Also we included approximated costs for reproducing the full results of experiments in D.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We cited [47] for MNIST, [7] for ShapeNet, [36] for 3Dshapes and [48] for SmallNORB.

(b) Did you mention the license of the assets? [Yes] For each dataaset, we included the URL to the license information in the reference section.

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] For ShapeNet dataset, we are approved to download the assets.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We don't find any personally identifiable information in the images used in our experiments.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

Table of Contents

A Supplemental results **15**

A.1 Qualitative and quantitative results on the prediction 15

A.2 Equivariance performance 17

A.3 More results on simultaneous block-diagonalization 20

A.4 Orthgonality of M^* during training 22

B Generated examples **22**

C Algorithm **25**

D Experimental settings **25**

D.1 Ablation studies 25

D.2 Training details 26

D.3 Additional details of datasets 27

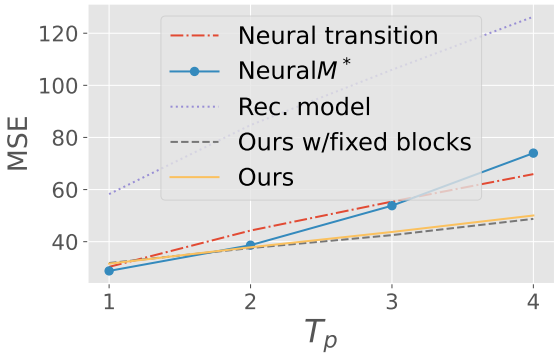
D.4 Network architecture 27

E Simultaneous Block-diagonalization **29**

F Formal statements and the proofs of the theory section **29**

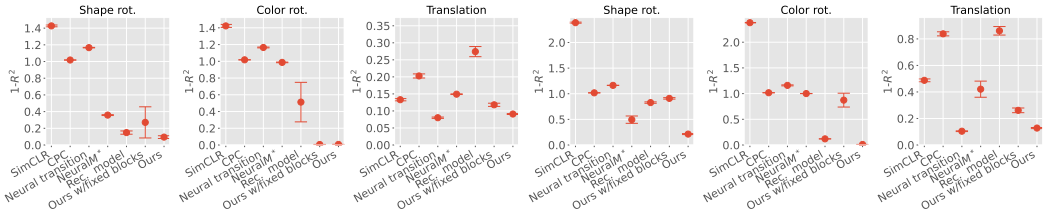
A Supplemental results

A.1 Qualitative and quantitative results on the prediction



(a) SmallNORB

Figure 9: Prediction errors on smallNORB. During the training phase, the models were trained to predict the observations only at $t_{\mathrm{p}} = 1$. The prediction errors at $t_{\mathrm{p}} > 1$ indicate the extrapolation performance.



(a) Sequential MNIST

(b) Sequential MNIST-bg

Figure 10: The results of linearly regressing the true transition parameters from M^*. For the performance evaluation, we used $1 - R^2$ scores (The value of 0 indicates the perfect prediction and 1 indicates the performance is chance level. $1 - R^2 > 1$ can happen when the model significantly overfits to the training set). For the color rotation and the shape rotation, $(\cos(v), \sin(v))$ was used as the target value where v is the angle velocity. For this experiment, we trained the models on a set of sequences generated from *digit 4 class* only, and trained/evaluated the linear regression performance on the trained models' features on a set of sequences created from all digit classes in MNIST. Because SimCLR, CPC and Neural transition do not directly compute M^*, the linear regression was computed from the concatenation of the two consecutive latent representations that were used in our method for the computation of M^*.

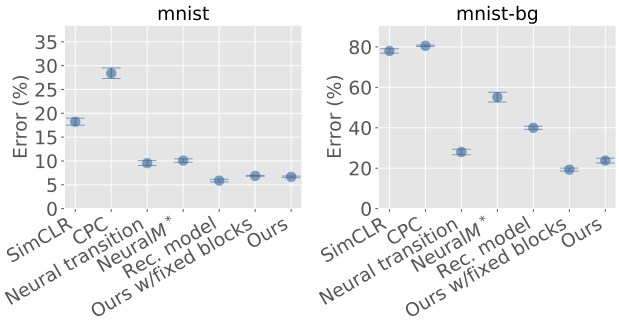


Figure 11: The results of digit classification evaluation on the sequential MNIST and MNIST-bg datasets. For this experiment, we trained the models on a set of sequences generated from only *digit 4 class*. We trained and evaluated the softmax classifier on the feature $\Phi(s_1)$ where s_1s are generated from all digit classes in MNIST.

A.2 Equivariance performance

Figure 12 shows $(M^*(\mathbf{s}) - M^*(\mathbf{s}'))^2$ for the pairs of sequences that transition with same g (e.g. $\mathbf{s} = \mathbf{s}(s_1, g), \mathbf{s}' = \mathbf{s}'(s_1', g)$). We see that M^*s computed from the representation learned by our method do not differ across $\mathbf{s}$ and $\mathbf{s}'$. This can also be confirmed visually in the generated sequences as well (Figure 13).

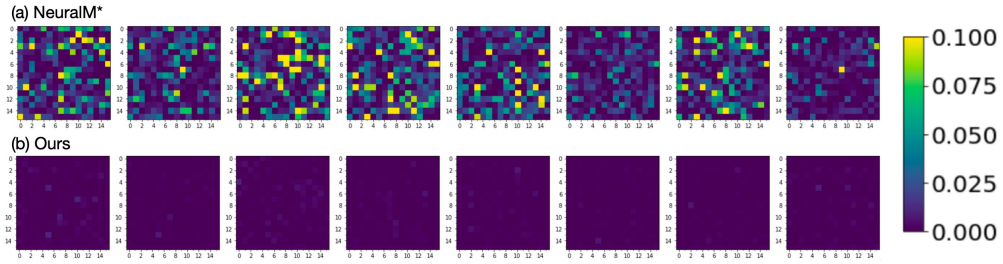
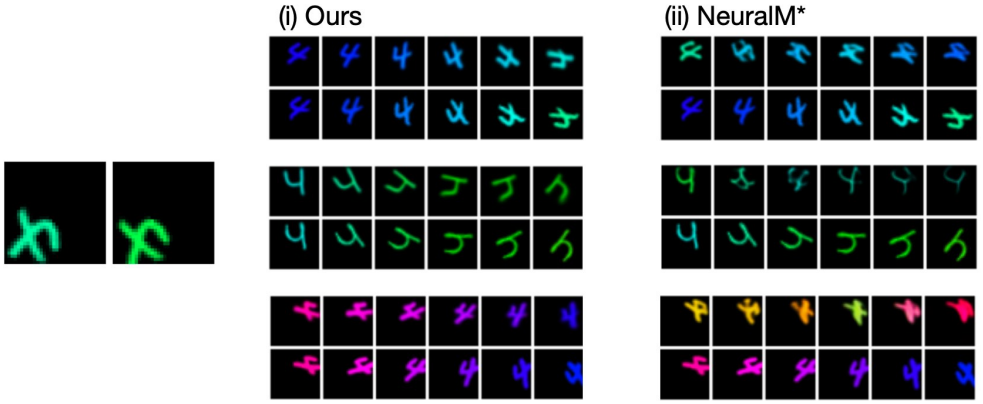
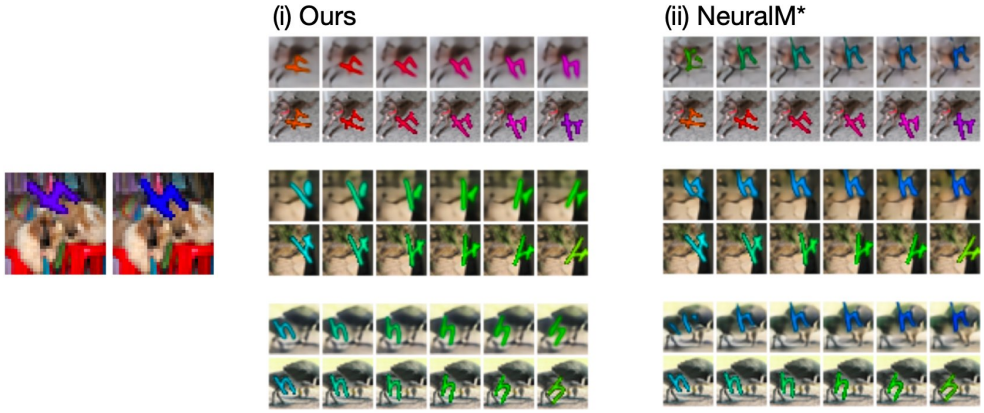


Figure 12: Visualization of $(M^*(\mathbf{s}) - M^*(\mathbf{s}'))^2$ where $\mathbf{s}, \mathbf{s}'$ that transition with the same g.

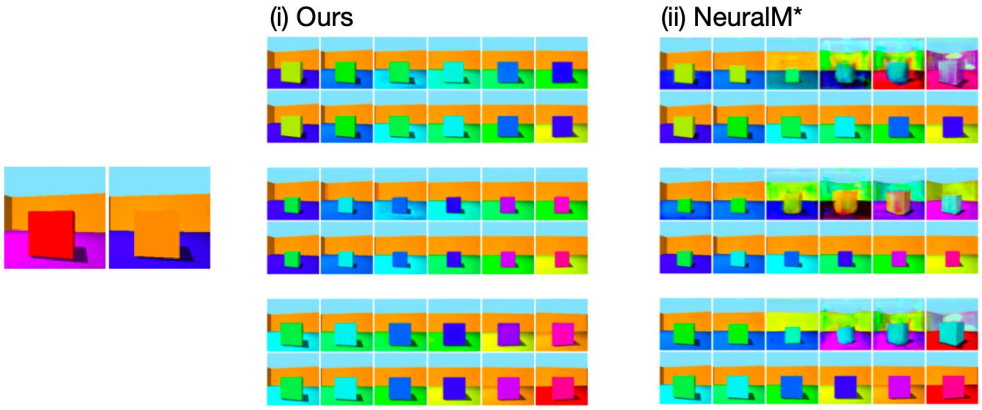


(a) Sequential MNIST

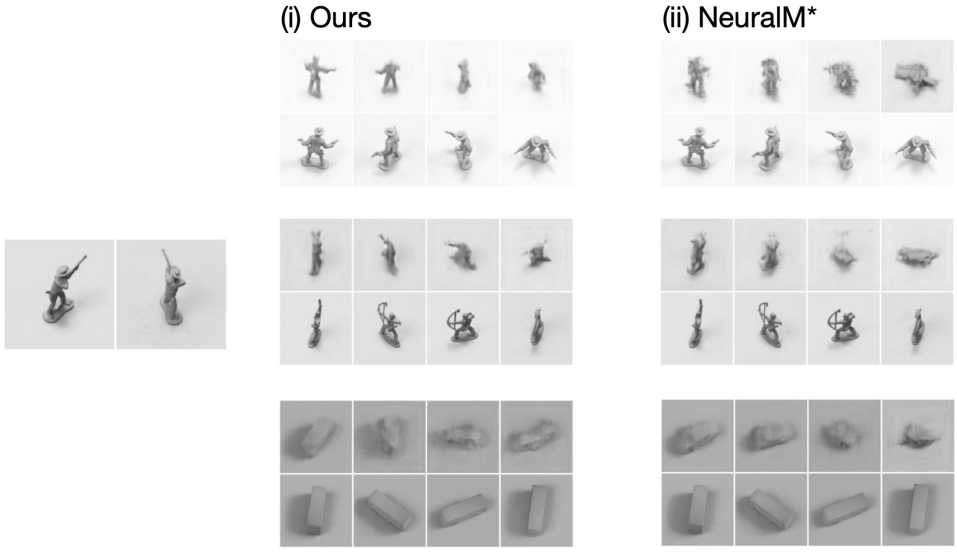


(b) Sequential MNIST-bg

Figure 13: The result of transferring M^* computed from one sequence to other sequences. For both sequential MNIST and sequential MNIST-bg, M^* was computed from the two consecutive images placed on the left edge of the figure. In each pair of rows shown on the right, the top row corresponds to the generated sequence and the bottom row corresponds to the ground truth sequence that transitions with the same g that was used to create the two consecutive images on the left. We see that each M^* computed from our representation acts on different sequences in the same way.

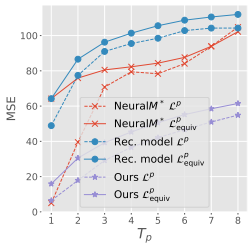


(a) 3DShapes

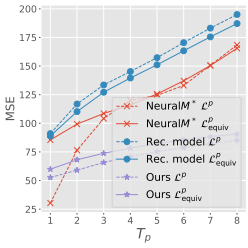


(b) SmallNORB

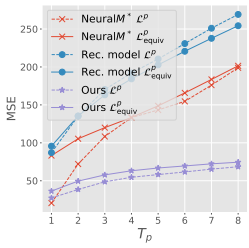
Figure 14: The result of transferring M^* on 3DShapes and SmallNORB. The visualization follows the same protocol as in Figure 13.



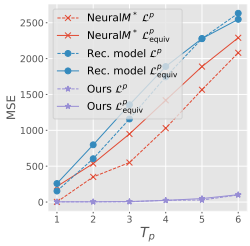
(a) Sequential MNIST



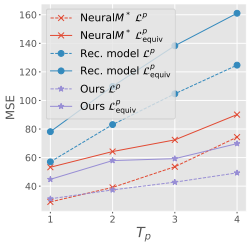
(b) Sequential MNIST w/ digit 4



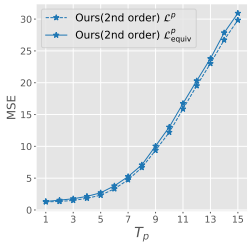
(c) Sequential MNIST w/ all digits



(d) 3DShapes



(e) SmallNORB



(f) Accelerated Sequential MNIST

Figure 15: Comparison of the prediction errrors and equivariance errors at $T_{\text{p}} \geq 1$.

Method	Seq. MNIST		Seq. MNIST-bg (w/ digit 4)	
	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$
Rec. Model	48.91±4.47	64.22±5.69	91.02±2.22	88.93±2.89
NeuralM^*	4.99±0.87	64.25±2.59	30.32±0.36	85.46±2.66
MSP (Ours)	6.42±0.21	**15.91**±0.49	52.67±0.86	**59.87**±1.37

(a) Equivariance performance on sequential MNIST and MNIST-bg w/ digit 4

Method	Seq. MNIST-bg (w/ all digits)		3DShapes	
	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$
Rec. Model	87.05±3.32	95.66±7.71	153.39±24.1	258.20±25.8
NeuralM^*	20.60±0.25	83.18±2.50	2.09±0.12	217.73±46.7
MSP (Ours)	27.38±0.14	**36.42**±0.08	2.75±0.25	**2.87**±0.30

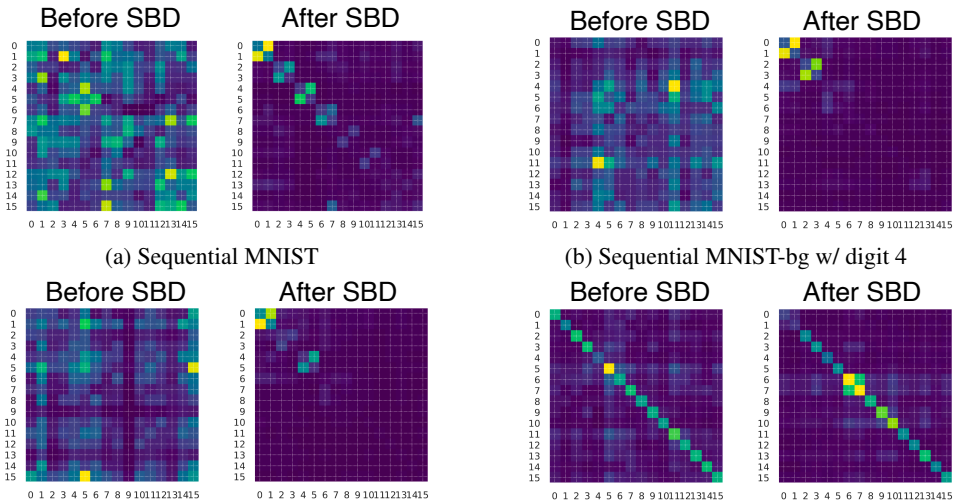
(b) Equivariance performance on sequential MNIST-bg w/ all digits and 3DShapes

Method	SmallNORB		Accelerated Seq. MNIST	
	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$	$\mathcal{L}^p$	$\mathcal{L}^p_{\text{equiv}}$
Rec. Model	57.01±2.69	78.14±4.42		
NeuralM^*	28.98±1.25	53.24±0.64		
MSP (Ours)	31.14±0.52	**44.77**±0.38	1.27± 0.02	1.34 ± 0.03

(c) Equivariance performance on SmallNORB and accelerated sequential MNIST

Figure 16: More detailed version of Fig 15 with standard deviation values. The statistics in this figure were calculated over three models initialized with different random seeds. For the definition of $\mathcal{L}^p$ and $\mathcal{L}^p_{\text{equiv}}$, see (2) and (6).

A.3 More results on simultaneous block-diagonalization



(a) Sequential MNIST

(b) Sequential MNIST-bg w/ digit 4

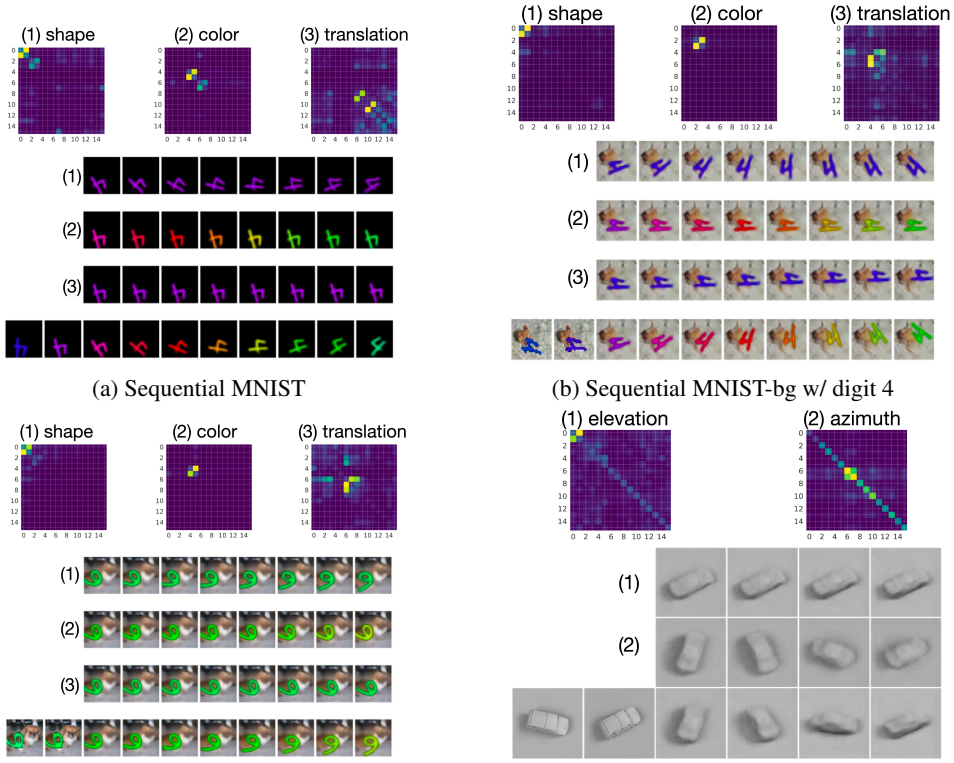
(c) Sequential MNIST-bg w/ all digits

(d) SmallNORB

Figure 17: The visualization of simultaneously block-diagonalized (SBD) matrices for Sequential MNIST/MNIST-bg and SmallNORB datasets. As in Figure 7a, our visualizations correspond to $abs(M^* - I)$ and $abs(V^* - I)$ instead of the raw matrices (V^* is the block-diagonalized version of M^*. See Section 3.3 and Section E).

Figure 17 is the visualization of the block structures revealed by the simultaneous block-diagonalization on Sequential MNIST/MNIST-bg and SmallNORB. The detail of the block-diagonalization method is provided in Section 3.3 and Section E.

To investigate what type of transformations these blocks correspond to, we studied the effect of using just one particular set of blocks in the block diagonalized transition matrix (Figure 18). To create the transformation of *one particular set of blocks*, we modified the block-diagonalized M^* by setting all block positions other than the target blocks to identity. We can visually confirm that disentanglement is achieved by the partition of block positions. See the figure captions for more details.



(a) Sequential MNIST

(b) Sequential MNIST-bg w/ digit 4

(c) Sequential MNIST-bg w/ all digits

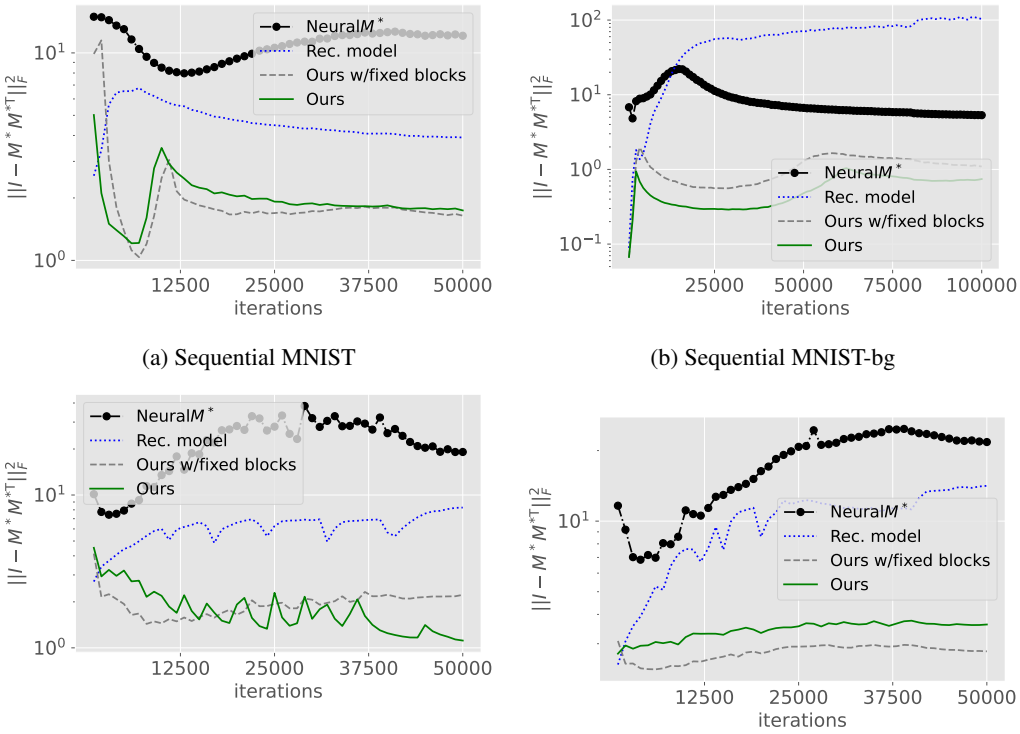
(d) SmallNORB

Figure 18: Generation of disentangled sequences. The bottom sequence in each frame of this figure is the ground truth. We generated each one of (1), (2) (and (3)) by applying the transformation corresponding to only one particular set of the blocks. To create each sequence, we first computed M^* from the first two time-steps($t = 1, t = 2$) in the ground truth, and block-diagonalized M^* to obtain $\hat{M}^*$. We then created the transformation corresponding to only *one particular set of blocks* by setting all the block positions of $\hat{M}^*$ other than the target blocks to identity. We then applied the powers of the one-block-set transformation to the image at $t = 2$ to generate the disentangled future sequence. The assignment of block positions to disentangled factors was found manually by looking at the activated blocks when we altered one factor in the ground truth sequences. See Table 1 for the correspondence between block positions and disentangled factors. We can visually confirm that disentanglement is achieved through block partitions.

dataset	The *factor-block position* correspondence
Sequential MNIST	(1) $\{0, 1, 2, 3\}$, (2) $\{4, 5, 6, 7\}$, (2) $\{8, 9, 10, 11\}$
Sequential MNIST-bg w/ digit 4	(1) $\{0, 1\}$, (2) $\{2, 3\}$, (3) $\{4, 5, 6, 7, 8\}$
Sequential MNIST-bg w/ all digits	(1) $\{0, 1, 2, 3\}$, (2) $\{4, 5\}$, (3) $\{6, 7, 8\}$,
3DShapes	(1) $\{0, 1\}$, (2) $\{2, 3, 4, 5\}$, (3) $\{6, 7\}$, (4) $\{8, 9, 10, 11\}$, (5) $\{12, 13, 14, 15\}$
SmallNORB	(1) $\{0, 1, 2, 3, 4, 5\}$, (2) $\{6, 7, 8, 9, 10, 11, 12, 13\}$

Table 1: The correspondence between block positions and disentangled factors in simultaneous block-diagonalization. For each i, " $(i)\{a_1, a_2, ...a_m\}$" means that the i-th disentangled factor has coordinates $\{a_1, a_2, ...a_m\}$. For example, the block that is positioned at coordinates $\{4, 5\}$ changes the second disentangled factor (shape rotation) in Sequential MNIST.

A.4 Orthgonality of M^* during training



(a) Sequential MNIST

(b) Sequential MNIST-bg

(c) 3DShapes

(d) SmallNORB

Figure 19: The transition of $\|I - M^* M^{*\mathrm{T}}\|_F^2$ during the training. We can observe that, for our method, the learned representation evolves in such a way that the estimated transition M^* tends to become orthogonal.

B Generated examples

Figures 20-26 show the seqeuences generated by our method and its variants for each dataset. The visualization follows the same protocol as in Figure 4.

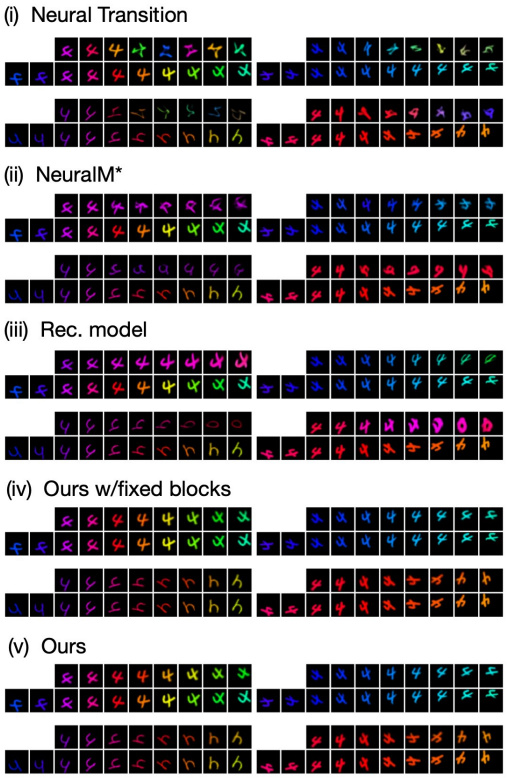


Figure 20: Sequential MNIST

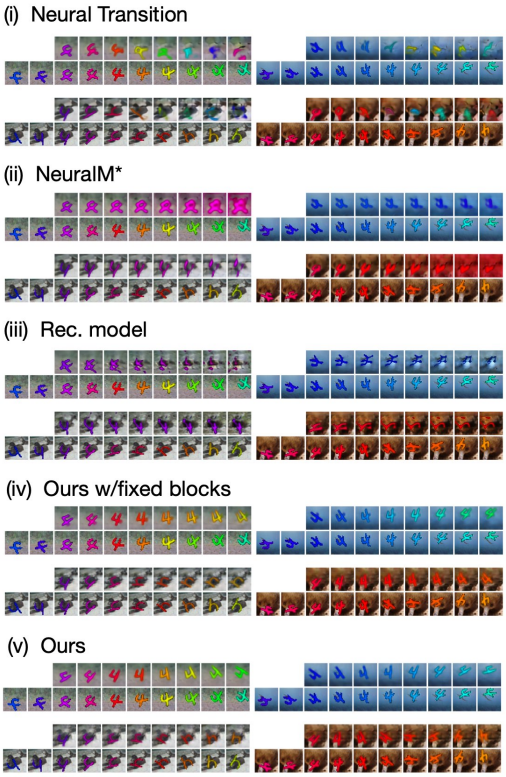


Figure 21: Seq. MNIST-bg (w/only digit 4)

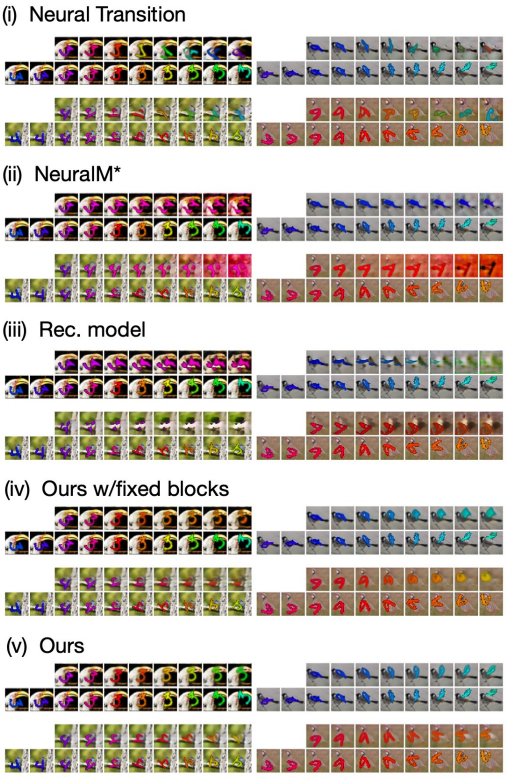


Figure 22: Seq. MNIST-bg (w/all digits)

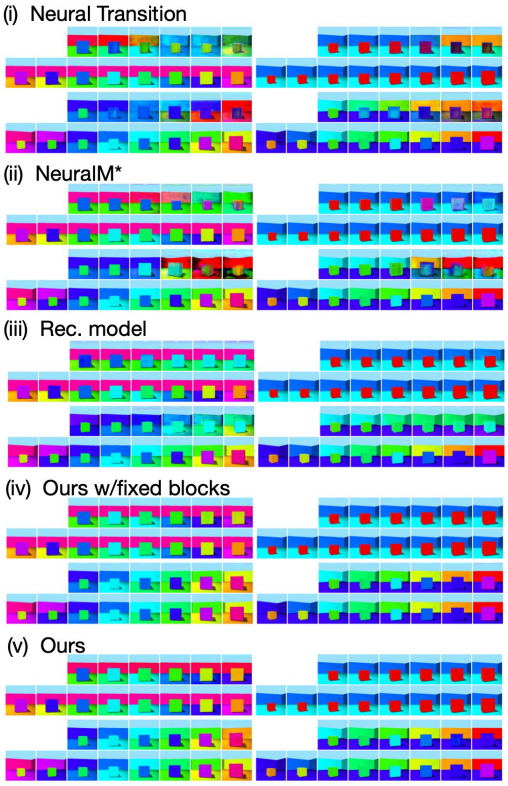


Figure 23: 3DShapes

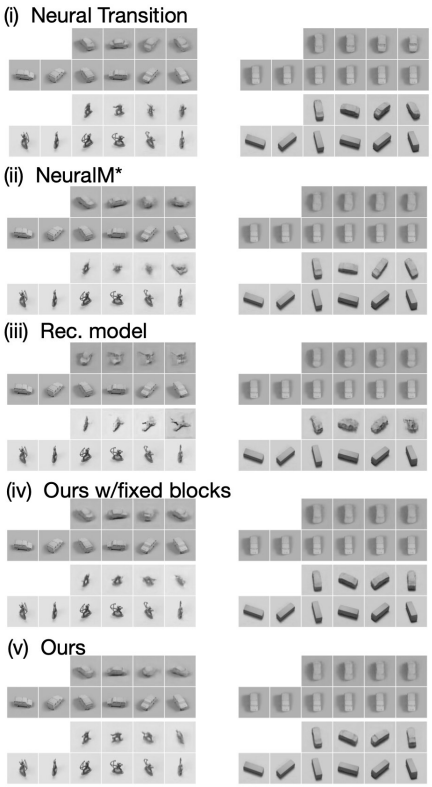


Figure 24: SmallNORB

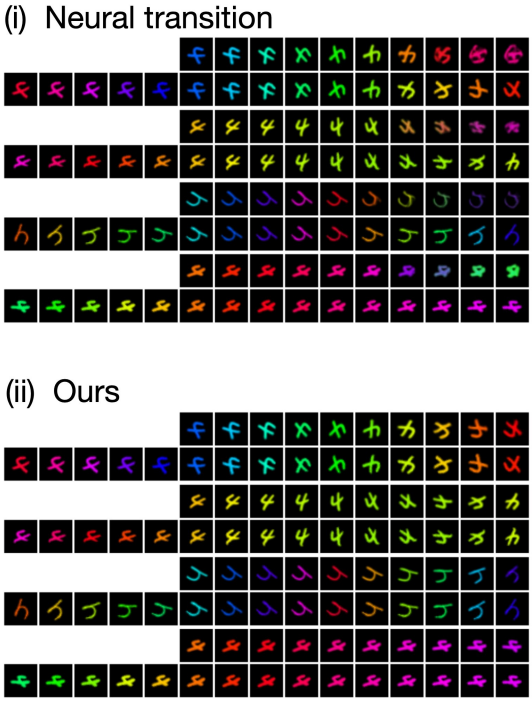


Figure 25: Accelerated Sequential MNIST. Both models were trained with $T_{\mathrm{c}} = 5$ and $T_{\mathrm{p}} = 5$. For the training procedure on this experiment, please see Section 4.4.

Figure 26: Sequential ShapeNet generated by the proposed method. The model was trained with $T_{\mathrm{c}} = 5$ and $T_{\mathrm{p}} = 5$. Our method cannot make good predictions on this dataset. Note that, unlike other datasets we studied on this paper, the transition in Sequential ShapeNet is not necessarily invertible, because some parts of a $3D$ object are often not visible in the 2D rendering.

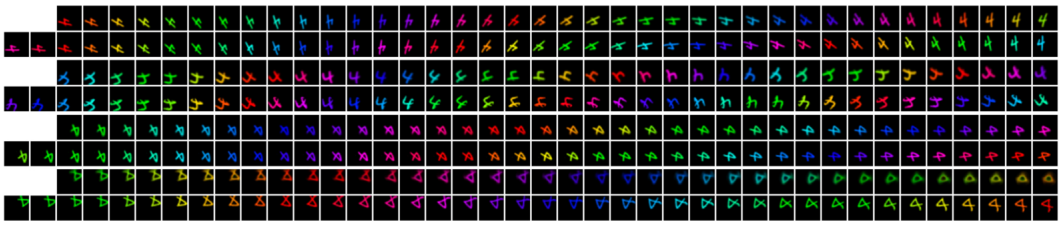


Figure 27: Extrapolation results for longer future horizon (t_{p} = 1,...,38). The visualization follows the same protocol as in Figure 4a.

C Algorithm

We provide the algorithmic description of our method. Definitions of all symbols in the table are the same as in the main sections.

Algorithm 1 Calculate the loss over Φ and Ψ.

Input: Given an encoder Φ, a decoder Ψ and a sequence of observations $\mathbf{s} = [s_1, \ldots, s_{T_{\mathrm{c}}}, s_{T_{\mathrm{c}}+1}, \ldots, s_T]$.

1. Encode the observations into latent variables $H_t = \Phi(s_t)$ for $s_1, ..., s_{T_{\mathrm{c}}}$.
2. Estimate the transition matrix by solving linear problem: $M^* = H_{+1}H_{+0}^{\dagger}$ where H_{+0} and H_{+1} are the horizontal concatenation $[H_1; H_2; ...; H_{T_{\mathrm{c}}-1}]$ and $[H_2; H_3; , ...; H_{T_{\mathrm{c}}}]$, respectively.
3. Predict the future sequence by : $\tilde{s}_t = \Psi((M^*)^{t-T_{\mathrm{c}}} H_{T_{\mathrm{c}}})$ for $t = T_{\mathrm{c}} + 1, ..., T$.
4. Calculate the loss for the sequence $\mathbf{s}$: $\sum_{t=T_{\mathrm{c}}+1}^{T} \|\tilde{s}_t - s_t\|_2^2$

D Experimental settings

D.1 Ablation studies

As ablations, we tested several variants of our method: **fixed 2x2 blocks**, **Neural**M^*, **Reconstruction model** (abbreviated as Rec. Model), and **Neural transition**. We describe each one of them below.

- **Fixed 2x2 blocks**: For this model, we separated the latent tensor $\Phi(s) \in \mathbb{R}^{16\times 256}$ into 8 subtensors $\{\Phi^{(k)}(s) \in \mathbb{R}^{2\times 256}\}_{k=1}^{8}$ and calculated the pseudo inverse for each k to compute the transition in each $\mathbb{R}^{2\times m}$ dimensional space. Essentially, this variant of our proposed method computes M^* as a direct sum of eight 2×2 matrices. In the pioneer work of [10] that endeavors to learn the symmetry in a linear system using the representation theory of commutative algebra, the authors hard-code the irreducible representations/block matrices in their model. Our study is distinctive from many applications of representation theory and symmetry learning in that we uncover the symmetry underlying the dataset not by introducing any explicit structure, but by simply seeking to improve the prediction performance. We therefore wanted to experiment how the introduction of the hard-coded symmetry like the one in [10] would affect the prediction performance.
- **Neural**M^*: Our method is "meta" in that we distinguish the internal training of M^* for each sequence from the external training of the encoder Φ. Put in another way, the internal optimization process of M^* itself is the function of the encoder. To measure how important it is to train the encoder with such a meta approach, we evaluated the performance of NeuralM^* approach. To reiterate, NeuralM^* uses a neural network M_θ^* that directly outputs M^* on the conditional sequence, and train the encoder and the decoder via

$$\sum_{t=T_c+1}^{T_c+T_p} \|\Psi(M_\theta^*(\mathbf{s}_{\mathrm{c}})^{t-T_{\mathrm{c}}}\Phi(s_{T_c})) - s_t\|_2^2,$$

thereby testing the training framework that is similar to our method "minus" the "meta" component.

- **Reconstruction model (Rec. model)**: In our default algorithm, we train our encoder and decoder with the prediction loss $\mathcal{L}^p$ in eq.(2) over the future horizon of length $T_\mathrm{p} - T_\mathrm{c}$. We therefore wanted to verify what would happen to the learned representation when we train the model with the reconstruction loss $\mathcal{L}^r$ in eq.(3) in which the model predicts the observations contained in the conditional sequence. Specifically, we trained Φ and Ψ based on $\mathcal{L}^r$ in (3) with $T = T_\mathrm{c} = 3$.
- **Neural Transition**: One important inductive bias that we introduce in our model is that we assume the latent transition to be linear. We therefore wanted to test what happens to the results of our experiment if we drop this inductive bias. For Neural transition, we trained a network with 1x1 1D-convolutions that inputs $\Phi(s)$ in the past to produce the latent tensor in the next time step; for instance, $\tilde{H}_{t+1} = \mathrm{1DCNN}(\Phi(s_t), \Phi(s_{t-1}))$ when $T_\mathrm{c} = 2$. This model can be seen as a simplified version of [60]. The 1DCNN was applied along the multiplicity dimension (m).

In testing all of these variants, we used the same pair of encoder and decoder architecture as the proposed method.

D.2 Training details

(a) Transformations (b) An example sequence

Figure 28: *Sequential MNIST* dataset. The transition in each sequence was produced by combining three families of actions: shape rotation, color rotation and translation.

For the model optimization in every experiment, we used ADAM [37]. The number of iterations for the parameter updates were $50,000$ on Sequential MNIST, 3Dshapes, and SmallNORB. The number of iterations was $100,000$ on Sequential MNIST-bg (with only digit 4 class) and accelerated Sequential MNIST. For MNIST-bg (with all digits) and Sequential ShapeNet, the number of iterations was $200,000$. We set the initial learning rate of ADAM optimizer to 0.0003 and decayed it to 0.0001 after a certain number of iterations. For Sequential MNIST, 3Dshapes, and SmallNORB, we began the decay at $40,000$-th iteration. For Sequential MNIST-bg (with digit 4 only) and accelerated Sequential MNIST, we began the decay at $80,000$-th iteration. For Sequential MNIST-bg (with all digits) and Sequential ShapeNet, we began the decay at $160,000$-th iteration.

The batchsize was set to 32 for all experiments. We conducted all experiments on NVIDIA A100 GPUs. Training our proposed model takes approximately one hour per $50,000$ iterations on a single A100 GPU. Total amount of time to reproduce the full results in our experiments is approximately 12 days on a single A100 GPU.

We found that the choice of the latent dimension ($a \times m$) does not make significant difference on the results as long as they are not too small (For example, if G is a torus group consisting of n commuting axis, a must be no less than $2n$ when all the observations are real-valued; otherwise the model will underfit the datasets. Also, we chose m to be larger than a so that $\Phi(x)$ becomes full rank almost surely. This allows us to solve M^* from $M^*\Phi(s_1) = \Phi(s_2)$ (We can compute M^* with $T_c = 2$.) Choosing $m > a$ also plays a role in the theory (Section F).

As for the NeuralM^* and Neural transition models, we also optimized the invertibility loss: $\sum_{t=1}^{T_c} \|\Psi(\Phi(s_t)) - s_t\|_2^2$ in addition to $\mathcal{L}^p$. Adding this loss to the original objective yielded better results for these models in terms of prediction error and equivariance error for all experiments.

SimCLR and CPC settings in the downstream task experiments in Figure 10 and 11 To evaluate our method as a representation learning method, we compared our method against SimCLR [8] and CPC [61, 27]. For SimCLR, we treated any pair of observations in the same sequence as a positive pair, and any pair of observations in different sequences as a negative pair. We used the

same encoder architecture as in our baseline experiment for both SimCLR and CPC. For the projection head of SimCLR, however, we used the same architecture as in the original paper. For the auto regressive network of latent representation in CPC, we used the same architecture as in Neural transition (see Section 4). The latent dimension was set to 512 for both models. We experimented with larger and smaller dimensions as well, but however large the difference, altering the dimensions did not result in significant improvements in terms of the representation quality evaluated in the experimental sections. The temperature parameters for the logit output were searched in the range of [1e-3, 1e-2, 1e-1, 1.0, 10.0]. Because SimCLR is not built for the sequential dataset, it is not expected to perform too well in terms of regression performance. We however evaluated these models as minimum performance baselines.

D.3 Additional details of datasets

Our training-test split was the same as the split in the original dataset. Therefore the train-test split of Sequential MNIST/MNIST-bg was the same as that of MNIST, and the split of the SmallNORB dataset we used was the same as that of the original SmallNORB. Meanwhile, the 3DShapes dataset does not have train-test split, so we conducted the training and the test evaluation on the same dataset for the sequential 3DShapes experiment. We also used only cubic shape examples on the 3DShapes experiments. For Sequential ShapeNet, 90% of objects in the original ShapeNetCore assets were used for the training and the rest were used for the evaluation. The input size of each example in a given sequence was $3 \times 32 \times 32$ for Sequential MNIST/MNIST-bg, $3 \times 64 \times 64$ for 3DShapes, $1 \times 96 \times 96$ for SmallNORB, and $3 \times 128 \times 128$ for Sequential ShapeNet.

To generate Sequential ShapeNet, we used Kubric [21] to render the objects in ShapeNet [7] datasets. For each sequence, we sampled one object from ShapeNetCore assets, and used 3D rotation to define the transition. The angle of 3D rotation in each axis(xyz) was sampled from the uniform distribution over the interval $[0, \pi/4)$.

D.4 Network architecture

We used ResNet-based encoder and decoder[26]. We used ReLU function [53, 20, 52] for each activation function and group normalization [68] for the normalization layer. We used weight standarization [55] for all of filters in each convolutional network. Also, we used trainable positional embedding in each block of the decoder, which was initialized to the 2D version of sinusoidal positional embeddings [66]. We provide the details of the architecture in Table 2 and Figure 29.

For the NeuralM^* method, we used the same model in the table 2a except the input channel of the network was set to 6 (and 2 for SmallNORB) because this method uses a pair of images (s_1, s_2) as an input.

For the Neural transition model, we used a network with 1x1 1D convolutions to map $[H_t, \ldots, H_{t+t'}]$ to $H_{t+t'+1}$. The network architecture is the 1x1 1D convolutional version of the table 2a without downsampling. The number of ResBlocks was set to two. We also replaced all of the group normalization layers with layer normalization [2].

	#channels or #dims	Resampling	Spatial Resolution
3x3 2DConv	32*k	-	H×W
ResBlock	64*k	Down	(H/2)×(W/2)
ResBlock	128*k	Down	(H/4)×(W/4)
ResBlock	256*k	Down	(H/8)×(W/8)
GroupNorm	256*k	-	(H/8)×(W/8)
ReLU	256*k	-	(H/8)×(W/8)
Flatten	256*k*(H/8)*(W/8)	-	-
Linear	16*256	-	-

(a) Encoder architecture

	#channels or #dims	Resampling	Spatial Resolution
Linear	256*k*(H/8)*(W/8)	-	-
Reshape	256*k	-	(H/8)×(W/8)
ResBlock	128*k	Up	(H/4)×(W/2)
ResBlock	64*k	Up	(H/2)×(W/4)
ResBlock	32*k	Up	H×W
GroupNorm	32*k	-	H×W
ReLU	32*k	-	H×W
3x3 2DConv	3 (1 for SmallNORB)	-	H×W

(b) Decoder architecture

Table 2: The detail of the encoder and decoder architecture used in our experiments. The columns of '#channels or #dims' and 'Spatial resolution' respectively represent the channels/dimensions and the spatial resolution at the end of each corresponding module. 'Resampling' column represents whether the corresponding layer performs upsampling (Up), downsampling (Down) or none of them (-). Please see Figure 29 for the detail of the ResBlock architecture. The value k in the table was set to 1 for 3DShapes, SmallNORB and Sequential ShapeNet. The value k was set to 2 for Sequential MNIST and accelerated Sequential MNIST, and 4 for Sequential MNIST-bg. For SmallNORB, we added one more downsampling ResBlock after the third ResBlock in the encoder and one more upsampling ResBlock before the first ResBlock in the decoder. For Sequential ShapeNet, we added two more downsampling ResBlock in the encoder and two more upsampling Resblock in the decoder.

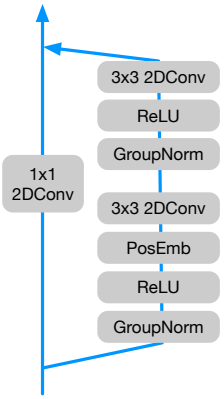


Figure 29: ResBlock architecture in the encoder and decoder. 'PosEmb' stands for the positional embedding layer which concatenates the learned positional embedding to its input. The embedding dimension was set to $32 \times H \times W$. The PosEmb layer was used only in decoder's resblock. For the encoder, we performed downsampling (mean average pooling) after the second convolution layer. For the decoder, we performed upsampling before the first convolution. Also, we added a downsampling layer (mean avegrage pooling) after the 1x1 convolution. For upsampling, we added a layer of nearest-neighbor upsampling before the 1x1 convolution. The number of groups for the group normalization layer was set to 32.

E Simultaneous Block-diagonalization

To find U that simultaneously block-diagonalizes all $M^*(\mathbf{s}|\Phi)$, we optimized U based on the objective function that measures the block-ness of $V^*(\mathbf{s}) := UM^*(\mathbf{s}|\Phi)U^{-1}$. Our objective function is based on the fact that, if we are given an adjacency matrix A of a graph, then the number of connected components in the graph can be identified by looking at the rank of graph Laplacian:

$$\dim(\mathrm{Ker}(\Delta A)) = \#\text{of blocks in } A \tag{8}$$

where Δ is the graph Laplacian operator on A. To relate our $V^* := UM^*U^{-1}$ to a graph, we see it as a bipartite graph and calculate the adjacency matrix by:

$$A(V^*(\mathbf{s})) := abs(V^*(\mathbf{s}))abs(V^*(\mathbf{s}))^{\mathrm{T}} \tag{9}$$

where $abs(V^*(\mathbf{s}))$ represents the element-wise absolute value of V^*: $abs(M^*(\mathbf{s}))_{ij} = |M^*_{ij}|$. To optimize Eq.(8) with respect to the change of basis U by continuous optimization, we used the lasso version of Eq.(8):

$$\mathcal{L}_{\mathrm{bd}}\left(V^*(\mathbf{s})\right) := \left\|\Delta\left(A\left(V^*(\mathbf{s})\right)\right)\right\|_{\mathrm{trace}} = \sum_{d=1}^{a} \sigma_d\left(A\left(V^*(\mathbf{s})\right)\right) \tag{10}$$

where $\sigma_i(\Delta A)$ is i-th singular value of ΔA. We used the symmetrically normalized version of the graph Laplacian: $\Delta A = I - D^{-1/2}AD^{-1/2}$ where D is the degree matrix of A. Summing this over all $\mathbf{s}^{(i)}$ in the dataset, we obtain: $\bar{\mathcal{L}}_{\mathrm{bd}} := \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{\mathrm{bd}}\left(V^*(\mathbf{s}^{(i)})\right)$. We search for U that simultaneously block-diagonalizes all $V^*(\mathbf{s}^{(i)})$ by minimizing $\bar{\mathcal{L}}_{\mathrm{bd}}$ w.r.t. U.

F Formal statements and the proofs of the theory section

We begin by summarizing the notations to be used in our formal statements. We use $\mathcal{X}$ to denote the space of all observations at a single time step, and $\Phi : \mathcal{X} \to \mathcal{H}$ to denote the encoder from $\mathcal{X}$ to the latent space $\mathcal{H}$. If $\mathbf{s} = [s_t \in \mathcal{X}; t = 1, ..., T]$ is one instance of video-sequence to be used in our training, we assume that, for each $\mathbf{s}$, there is an operator $g : \mathcal{X} \to \mathcal{X}$ such that $gs_t = s_{t+1}$ for each t. As such, each $\mathbf{s}$ is characterized by a pair of initial state $s_1 \in \mathcal{X}$ and $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of all operators considered. Thus, we would use $\mathbf{s}(s_1, g)$ to denote a specific sequence.

Now, given a fixed encoder $\Phi : \mathcal{X} \to \mathbb{R}^{a\times m}$, our training process computes the transition matrix M independently for each instance of $\mathbf{s}(s_1, g) = [s_t]_{t=1}^{T} = [g^{t-1}s_1]_{t=1}^{T}$. In particular, we compute

$$M^*(g, s_1|\Phi) = \arg\min_{M} \frac{1}{T}\sum_{t=1}^{T-1} \|\Phi(s_{t+1}) - M\Phi(s_t)\|_F^2. \tag{11}$$

In the theory developed here, we investigate the property of the optimal Φ when $T = \infty$ so that $M^*(g, s_1|\Phi)$ achieves

$$\|\Phi(s_{t+1}) - M^*(g, s_1|\Phi)\Phi(s_t)\|^2 = 0$$

or equivalently,

$$M^*(g, s_1|\Phi)\Phi(g^{t-1} \circ s_1) = \Phi(g^t \circ s_1)$$

for all $t \in \mathbb{N}$, $g \in \mathcal{G}$, and $s_1 \in \mathcal{X}$. We would like to know whether $M^*(g, x|\Phi)$ has no dependency on x so that $M_g = M^*(g|\Phi)$ defines an equivariance relation and hence a group representation .

We begin tackling this problem by first investigating $M^*(g, x|\Phi)$ within an orbit $\mathcal{G} \circ x := \{h \circ x \in \mathcal{X} \mid h \in \mathcal{G}\}$. That is, we check if we can say $M^*(g, x|\Phi) = M^*(g, h \circ x|\Phi)$ for any $g, h \in \mathcal{G}$ and $x \in \mathcal{X}$. We call this property *intra-orbital homogeneity*.

We assume that $\mathcal{G}$ is a compact commutative Lie group in the following result.

Proposition F.1 (Intra-orbital homogeneity)**.** *Suppose that $\mathcal{G}$ is a compact commutative Lie group, $\Phi(x) \in \mathbb{R}^{a\times m}$ has rank a, and $M(g, x) \in \mathbb{R}^{a\times a}$ satisfies*

$$M(g, x)\Phi(g^k \circ x) = \Phi(g^{k+1} \circ x) \tag{12}$$

for all $k \in \mathbb{N} \cup \{0\}, x \in \mathcal{X}$ and $g \in \mathcal{G}$. If $M(g, x)$ is continuous with respect to g and is uniformly continuous with respect to x, then

$$M(g, x) = M(g, h \circ x)$$

for all $h \in \mathcal{G}$.

Before going into the proof of this proposition, we show the following lemma about the basic properties of $M(g,x)$ that satisfies (12).

Lemma F.2. *Assume that $\Phi(x) \in \mathbb{R}^{a\times m}$ has rank a, and that $a \times a$-matrix $M(g,x)$ satisfies* (12) *for all $k \in \mathbb{N} \cup \{0\}, x \in \mathcal{X}$ and $g \in \mathcal{G}$. Then,*

(i) $M(gh,x) = M(g, h\circ x)M(h,x)$ *for any $g,h \in \mathcal{G}$ and $x \in \mathcal{X}$.*
(ii) $M(g^\ell, x) = M(g,x)^\ell$ *for any $\ell \in \mathbb{Z}$, $g \in \mathcal{G}$, and $x \in \mathcal{X}$.*
(iii) $M(g, g^\ell \circ x) = M(g,x)$ *for any $\ell \in \mathbb{Z}$, $g \in \mathcal{G}$, and $x \in \mathcal{X}$.*

Proof. First note that, from (12) with $k = 0$, we have

$$M(g,x)\Phi(x) = \Phi(g\circ x) \tag{13}$$

for any $g \in \mathcal{G}$ and $x \in \mathcal{X}$.

Using (13) repeatedly, we have

$$M(gh,x)\Phi(x) = \Phi(gh\circ x) = \Phi(g\circ(h\circ x)) = M(g,h\circ x)\Phi(h\circ x) = M(g,h\circ x)M(h,x)\Phi(x).$$

The rank assumption of Φ proves (i).

Also, (13) implies $M(e,x) = id$ for the unit $e \in \mathcal{G}$. We will first prove (ii) and (iii) with $\ell > 0$.

For (ii), note that the repeated use of (12) necessiates

$$\Phi(g^\ell \circ x) = M(g,x)\Phi(g^{\ell-1}\circ x) = \cdots = M(g,x)^\ell \Phi(x).$$

On the other hand, $\Phi(g^\ell \circ x) = M(g^\ell, x)\Phi(x)$. Equating these two expression of $\Phi(g^\ell \circ x)$ proves (ii) with $\ell > 0$.

Meanwhile, from (12) we have $\Phi(g^{\ell+1}\circ x) = M(g,x)\Phi(g^\ell \circ x)$, while

$$\Phi(g^{\ell+1}\circ x) = \Phi(g\circ(g^\ell \circ x)) = M(g, g^\ell \circ x)\Phi(g^\ell \circ x).$$

This proves the assertion (iii) for $\ell > 0$.

Now, substituting $x \leftarrow g^{-1}\circ x$ for (iii) with $\ell = 1$, we obtain $M(g,x) = M(g, g^{-1}\circ x)$. On the other hand, substituting $h \leftarrow g^{-1}$ for (i), we get $M(g, g^{-1}\circ x)M(g^{-1},x) = M(e,x) = id$. Thus,

$$M(g^{-1},x) = M(g,x)^{-1}.$$

Replacing g with g^{-1} in (ii) thus leads to $M(g^{-\ell},x) = M(g^{-1},x)^\ell = M(g,x)^{-\ell}$ for any $\ell \in \mathbb{N}$. This shows that (ii) holds for the negative integers as well. Also, substituting $g \leftarrow g^{-1}$ in (iii) yields $M(g^{-1}, g^{-\ell}\circ x) = M(g^{-1},x)$. Taking the inverse of the both sides proves the assertion (iii) for the negative integers. □

Proof of Proposition F.1. Let $h, g \in \mathcal{G}$ be given. Since $\mathcal{G}$ is a connected commutative Lie group, the exponential map $\exp : \mathfrak{g} \to \mathcal{G}$ is surjective, where $\mathfrak{g}$ is the Lie algebra of $\mathcal{G}$ [19]. Therefore, there exists some $\eta \in \mathfrak{g}$ such that $\exp(\eta) = h$. Then, for any $n \in \mathbb{N}$, we can define $h^{\frac{1}{n}} := \exp(\eta/n)$ and $h^{\frac{1}{n}} \to e$ as $n \to \infty$.

By the uniform continuity assumption on $M(\cdot,x)$, for any $\epsilon > 0$, we can choose n large enough so that

$$\|M(gh^{\frac{1}{n}}, g^{-n}\circ x) - M(g, g^{-n}\circ x)\|_F < \epsilon, \tag{14}$$

and

$$\|M(gh^{\frac{1}{n}}, h\circ x) - M(g, h\circ x)\|_F < \epsilon. \tag{15}$$

From Lemma F.2 (iii), we have

$$M(gh^{\frac{1}{n}}, g^{-n}\circ x) = M(gh^{\frac{1}{n}}, (gh^{\frac{1}{n}})^n g^{-n}\circ x),$$

and thus it follows from the commutativity assumption that

$$M(gh^{\frac{1}{n}}, g^{-n}\circ x) = M(gh^{\frac{1}{n}}, h\circ x). \tag{16}$$

At the same time, Lemma F.2 (iii) implies $M(g, g^{-n} \circ x) = M(g, x)$ so that (14) and (16) necessiates

$$\|M(gh^{\frac{1}{n}}, h \circ x) - M(g, x)\|_F < \epsilon. \tag{17}$$

Finally, the combination of (15) and (17) guarantees

$$\begin{aligned} &\|M(g, h \circ x) - M(g, x)\|_F \\ &\qquad \leq \|M(g, h \circ x) - M(gh^{\frac{1}{n}}, h \circ x)\|_F + \|M(gh^{\frac{1}{n}}, h \circ x) - M(g, x)\|_F < 2\epsilon. \end{aligned}$$

Because $\epsilon > 0$ is arbitrarily small, $\|M(g, h \circ x) - M(g, x)\|_F = 0$ necessarily holds, and the claim follows. □

Proposition F.3. *Suppose that, for a compact connected Lie group $\mathcal{G}$ and connected $\mathcal{X}$, $M : \mathcal{G} \times \mathcal{X} \to \mathbb{R}^{a \times a}$ in* (12) *satisfies the intra-orbital homogeneity, and that $\Phi(x) \in \mathbb{R}^{a \times m}$ has rank a for all x. If $M(g, x)$ is continuous with respect to x, then $M(g, x)$ is similar to $M(g, x')$ for all x, x'; that is, there is some $P \in GL(a, \mathbb{R})$ such that $PM(g, x)P^{-1} = M(g, x')$ for all $g \in \mathcal{G}$.*

Proof. From Lemma F.2, we have

$$M(gh, x) = M(g, h \circ x)M(h, x).$$

Combining this with intra-homogeneity $M(g, h \circ x) = M(g, x)$ provides

$$M(gh, x) = M(g, x)M(h, x),$$

which means that, for each fixed x,

$$M_x : \mathcal{G} \to GL(a; \mathbb{R})$$

defined by $M(g, x) = M_x(g)$ is a representation of the Lie group $\mathcal{G}$ [19]. Now, if $\mathcal{G}$ is compact and connected as assumed in the statement, $M_x(g)$ is completely reducible, and M_x is similar to a direct sum of irreducible representations. We then use the fact from character theory [19] that the multiplicity of any irreducible representation D in M_x can be computed by

$$\langle M_x | D \rangle = \int_{\mathcal{G}} tr(M(g, x))\overline{tr(D(g))}\mu(dg), \tag{18}$$

where μ is a Haar measure of $\mathcal{G}$ with volume 1, and $\overline{tr(D(g))}$ is the complex conjugate of $tr(D(g))$. Because $\langle M_x | D \rangle$ is a multiplicity, it takes an integer value. At the same time, by its definition and the continuity of $M(\cdot, x)$, this value is continuous with respect to x. Thus, $\langle M_x | D \rangle$ must be constant on $\mathcal{X}$ by the connectedness of $\mathcal{X}$. That is,

$$\langle M_x | D \rangle = \langle M_{x'} | D \rangle$$

for all $x, x' \in \mathcal{X}$. This means that, irrespective of x, $M(g, x)$ is similar to the direct sum of the same set of irreducible representations, and the claim follows. □