Understanding Generalization and Robustess of Learned Representations of Chaotic Dynamical Systems

Luã Streit [1 2] **Vikram Voleti** [3 4] **Tegan Maharaj** [2 3]

Abstract

We investigate the generalization capabilities of different methods of learning representations via an extensible synthetic dataset of real-world chaotic dynamical systems introduced by Gilpin (2021). We propose an evaluation framework built on top of this dataset, called ValiDyna, which uses probes and multi-task learning to study robustness and out-of-distribution (OOD) generalization of learned representations across a range of settings, including changes in losses, architecture, etc. as well as changes in the distribution of the dynamical systems' initial conditions and parameters. Our evaluation framework is of interest for generalization and robustess broadly, but we focus our assessment here on evaluating learned representations of ecosystem dynamics, with the goal of using these representations in ecological impact assesments, with applications to biodiversity conservation and climate change mitigation.

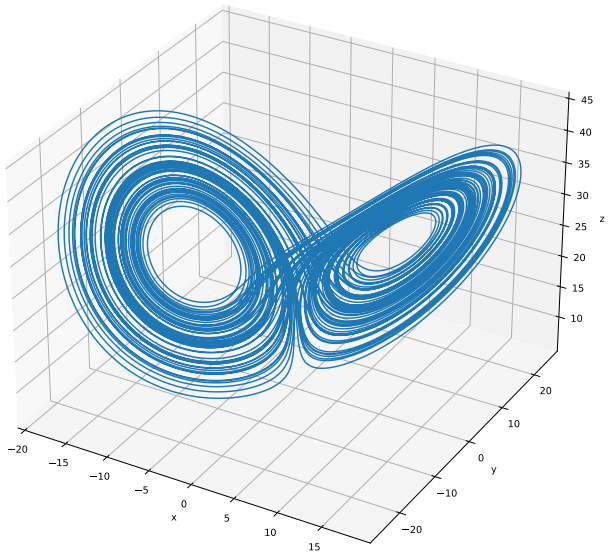


Figure 1. A single trajectory from the Lorenz dynamical system.

1. Introduction

Chaos is typically defined as extreme sensitivity to initial conditions; with this property even deterministic systems become impossible to predict in the limit. However, chaotic systems do exhibit patterns, such as characteristic shapes or subsequences; for example the classic butterfly shape of the Lorentz attractor shown in Fig. 1. This makes chaotic systems a compelling challenge for representation learning: can representations be learned which capture such high-level patterns, making the representations useful for downstream tasks such as forecasting, generative modelling, infilling, few-shot transfer learning, etc.? Which losses, architectures, regularization, and other design choices lead to better representations? How can we be sure the learned representations generalize well, and are not merely picking up on spurious correlations? How robust are the representations to differences in initial conditions, environmental settings, or shifts in distribution? These are some of the questions we seek to answer using our evaluation framework, ValiDyna.

We are motivated to validate learned reprsentations of dynamical systems because we wish to use them in a real-world setting: for Ecological Impact Assessments (EIAs). EIAs seek to characterize risks or impacts of environmental change on ecosystems or society. This lengthy and expensive process is typically done manually by a team of expert practitioners, and an important component of the process often (and increasingly) involves computational modelling of the ecosystem. This is typically done by collecting a series of environmental measurements, fitting an Ordinary Differential Equation (ODE) based on previously published models of similar ecosystems, modeling a proposed change (or often, a small set of possible changes) to the ecosystem in terms of changed variables in the ODE, solving the new ODE(s), and examining the effect on variables of interest

[1]Faculty of Computer Science, Ecole Polytechnique Fédérale de Lausanne, Switzerland [2]Faculty of Information, University of Toronto, Canada [3]Mila, Montreal, Canada [4]University of Montreal, Canada. Correspondence to: Luã Streit <lua.streit@proton.me>, Tegan Maharaj <tegan.maharaj@utoronto.ca>.

Published at the ICML 2022 Workshop on Spurious Correlations, Invariance, and Stability. Baltimore, Maryland, USA.

such as the population levels of keystone species. Representation learning could help uncover patterns that transfer across ecosystems, and provide a 'foundation' model that could be adapted to new settings with relatively less data, enabling computational EIAs to be performed more quickly and in a wider range of situations.

2. Related Work

Gilpin (2021) present a dataset of ODEs that show chaotic behaviour under certain conditions. They benchmark ML models typically used with time series for a variety of tasks, including forecasting, and also highlight some relationships between model performance and the properties of the chaotic ODE.

Many works have studied generalization (in particular OOD) in the context of dynamical systems. Arjovsky (2021) discuss how generalization relies on assumptions about the test data, and how they depend on the specifics of the task (No Free Lunch theorem). They discuss how having multiple training environments allows models to ignore spurious correlations, and to produce better predictions based on invariant features. Yin et al. (2022) propose a framework to learn contextual dynamics by decomposing $f_e(x_t^e) = \frac{dx_t^e}{dt}$ into $f_e = f + g_e$ and regularizing g_e in order to maximize the information in f, which is common to all environments. Gulrajani & Lopez-Paz (2021) discuss model selection for Domain Generalization (DG) as non-trivial and integral to any DG algorithm. They propose model selection algorithms and develop a framework (DomainBed) for testing domain generalization. Raissi et al. (2019) employ partial differential equations as a regularization mechanism that enforces the shape of the neural network output. Lee et al. (2021) enforce the learning of explicit context features that are then used by the Reinforcement Learning (RL) agent to predict the dynamics. Wang et al. (2020) show that ML techniques generalize badly when the distribution of the data or the system parameters in the test set are not included in the train set.

We aim to support works of this nature, by providing a relatively generic and lightweight set of exploratory analyses and evaluation approaches, together with standards for their use. Our primary objective is to make this toolbox of practical use in evaluating learned representations for scientific applications, like EIAs.

3. The ValiDyna evaluation framework

Our framework is built upon the Pytorch and Pytorch-Lightning libraries, and emphasizes the use of probes and multi-task learning for evaluating representations, as well as simplifying more standard robustness and sensitivity analyses. In summary, it allows evaluation of the following components:

- Architecture (e.g. RNN, Transformer, N-BEATS)
- Supervised loss (e.g. MSE, Cross-Entropy, MLE)
- Regularization (e.g. Dropout, Zoneout, L2, L1)
- System parameters (e.g. coefficients, scale parameters)

Via the following means:

- Sensitivity/robustness analysis: Perturbation of initial conditions/system parameters via different noise distributions
- Validation suites: Validating on a set of related test sets generated via different noise distributions
- Probes: Using a linear layer or another feed-forward network attached to different layers of the representation to assess performance at the task defined by that probe (e.g. classification, forecasting)
- Multi-task performance: Training with multiple objectives, e.g. reconstruction, classification, and forecasting, and tracking performance across all of them
- Transfer/few-shot learning: Pre-training on one task and then fine-tuning (only on a few examples, for few-shot learning) on another, and measuring performance
- Generative modelling: Sampling from the learned model and using various measures of similarity to the training data distribution

4. Experiments

In this initial study, we demonstrate several components of the ValiDyna framework, focusing on validating learned representations for their use in EIAs. First, we discuss the general setting of our experiments. Next, in section 4.2, we perform a robustness analysis to random data splits. Then, we perform a one-shot learning experiment to compare the quality of representations accross attractors in subsection 4.3. Finally, in subsection 4.4 we assess multi-task performance, training for forecasting with a self-supervised task of environment classification, and vice-versa.

4.1. Setting

We focus on the representation capabilities of ML models that are often applied to time series:

- Recurrent Neural Networks (RNNs) such as LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs are likely the most popular ML architecture applied to

time series, given that these models can process series of arbitraty time length.

- The Transformer (Vaswani et al., 2017) is a fully-attentional model that has shown state-of-the art performance in sequence-to-sequence translation tasks, and has replaced RNNs for many tasks.

- N-BEATS (Oreshkin et al., 2020) achieves state-of-the-art performance on forecasting tasks, thanks to its use of neural-basis expansions and double residual connections. Unlike RNNs and Transformers, N-BEATS needs the number of input and output time steps to be fixed.

Due to N-BEATS' limitation of requiring a fixed number of input time steps, we configure all our models to take input time series of fixed length T_{in}, and for the forecasting task to predict an output time series of fixed length T_{out}. See Appendix A for more details on how models are adapted to a Multi-Task setting.

As not all dynamical systems have the same space dimension (i.e. vector size per time step), we need to either train a different model for each possible space dimension, or to pad low dimensional time series with zeros. We focus on the chaotic attractors of dimension 3, which are the most common (100 out of 134 in the dataset).

For the classification task, we train our models to predict the chaotic dynamical system from which the given trajectory chunk comes from, using a standard cross-entropy loss. For the forecasting task, the models predict the next T_{out} time steps given the previous T_{in} ones.

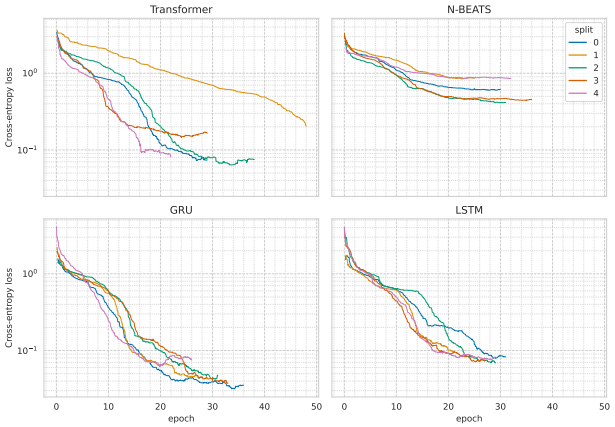
All models are trained with an AdamW optimizer, the learning rate starts at 0.01, and is divided by 5 whenever the validation loss stagnates, with 1 epoch of patience. We also perform early stopping by monitoring the validation loss with patience 5.

4.2. Robustness Analysis

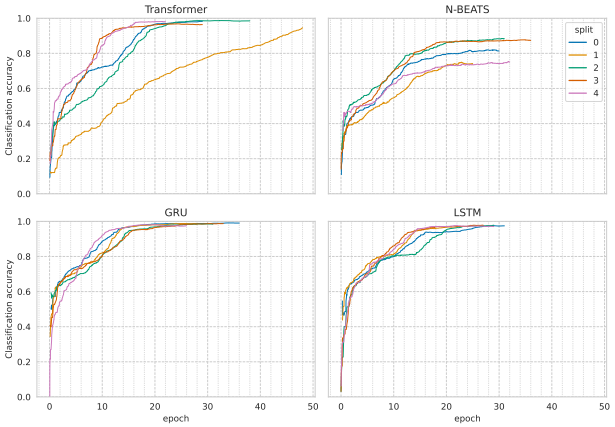
In this experiment, we measure the effect of the random train / validation split by performing 5 different ones for the classification task. The results in Fig. 2 show that some models, namely the Transformer and N-BEATS, have a huge variability in performance depending on the data split, and likely overfit spurrious correlations.

4.3. One-shot Learning

In this experiment, we focus on the attractor called SprottE. We consider a set of four attractors having similar differential equations to SprottE, namely SprottA-D, and another set of four other attractors very different to SprottE.



(a) Validation cross-entropy loss



(b) Validation accuracy

Figure 2. Validation loss and accuracy during training for the classification task, for different model architectures and 5 different random splits. A running average of length 100 is used for readability. The two RNN models seem to consistently achieve a good performance; N-BEATS shows more variability and worse results; the Transformer shows the most variability although it ends up achieving good results.

Some of our models are pre-trained on one set of attractors (similar or different) and then SprottE is added, while other models are trained directly on the corresponding set of five attractors.

We seek to evaluate how the one-shot performance of the models on SprottE changes as a function of the other attractors in the training set, as well as the effect of pre-training on performance. To measure that, we use metrics that only reflect how well the model is doing for SprottE and not for the other attractors: for the forecasting task, we only measure the MSE loss for SprottE trajectories; for the classification task, we measure the classification sensitivity (true positive rate) and specificity (true negative rate).

The results in Fig. 3 show that the relationship between

task performance and the one-shot experiment setting is complicated, and practices leading to good forecasting performance might not be the best ones for classification.

4.4. Multi-Task Transfer-Learning

In this experiment, we seek to evaluate the usefulness of transfering learned representations from one task to another. More specifically, we consider the tasks of classification and forecasting, and we train our models in one, then freeze the feature extractor weights (see Appendix A) and train the head corresponding to the other task. With our framework, this procedure can be simply written as:

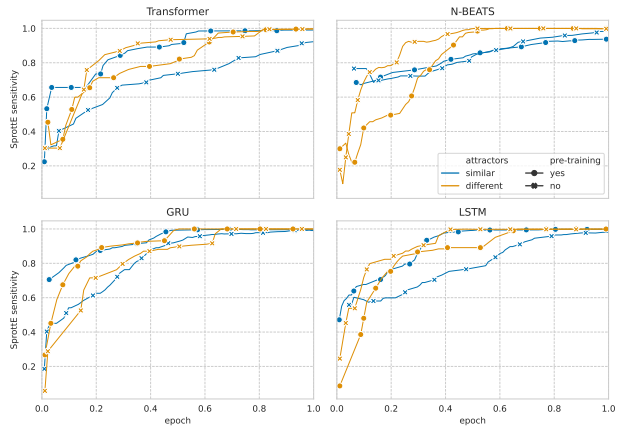
```
# Python 3.9 pseudocode
Model: Type[MultiTaskModel]
model = Model()
classifier = Classifier(model)
classifier.fit(class_data)
model.freeze_featurizer()
forecaster = Forecaster(model)
forecaster.fit(forecast_data)
```

The results in Fig. 4 suggest that the representations learned by N-BEATS are not useful accross tasks, presumably because we are not exploiting the neural basis expansions to the fullest. However, the representations learned for classification by the Transformer do transfer well to forecasting, allowing for a better performance compared to no pre-training.

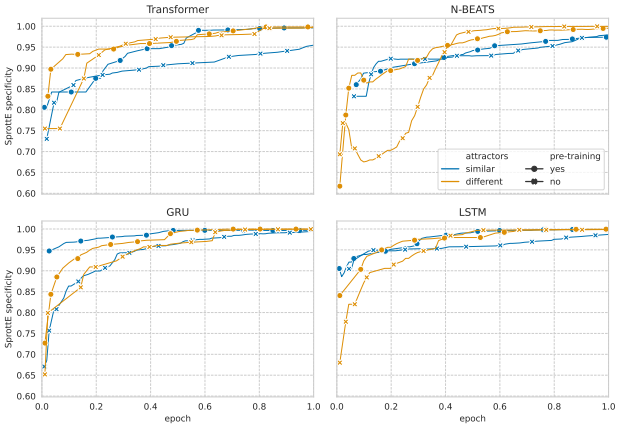
5. Discussion & Conclusions

Experts in various scientific fields are increasingly making use of learned respresentations. An important barrier to application for many researchers is the lack of standards and reusable code for even relatively basic exploration and evaluation of reprsentation quality and robustness. Our ValiDyna evaluation framework provides a lightweight set of tools and standards for assessing learned representations of dynamical systems in ways relevant to their real-world use.

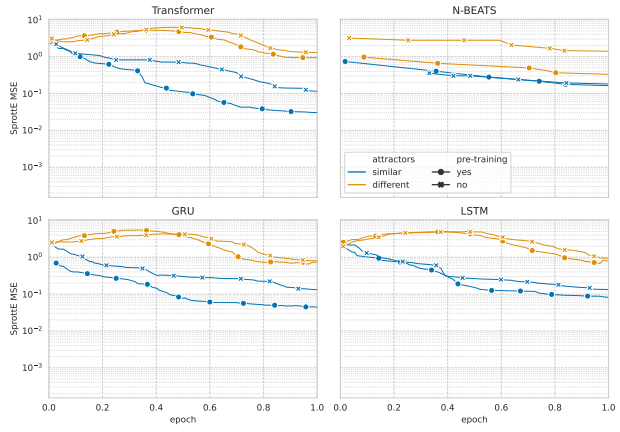
We demonstrate preliminary results using the framework to evaluate representations for use in EIAs, and we believe ecology and climate change are important areas of application where ValiDyna can be useful. ValiDyna provides tools and evaluation standards that are also useful more broadly in understanding learned representations of dynamical systems, for example supporting research in interpretability, empirical exploration of generalization and learning dynamics, as well as responsible AI practices such as audit trails and identification of failure modes and undesirable behaviour.



(a) Classification sensitivity

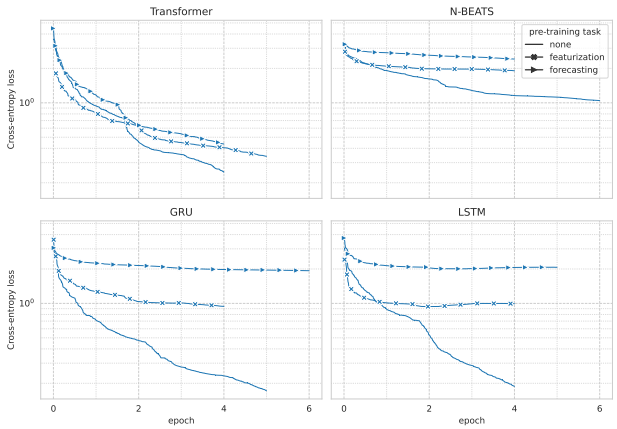


(b) Classification specificity

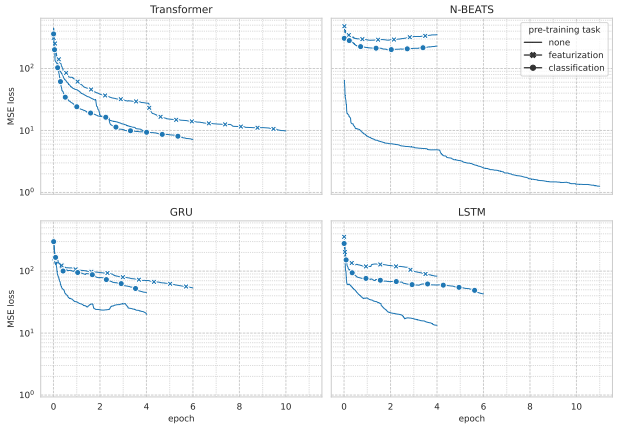


(c) Forecasting MSE

Figure 3. Classification sensitivity/specificity and forecasting loss for the SprottE attractor as a function of the other attractors present in the training set, and whether the models were pre-trained without SprottE (running average of length 20). Pre-training on a similar set of attractors seems to yield the best forecasting perfomance, while pre-training on different attractors results in slightly better results for classification.



(a) Classification for different pre-training tasks



(b) Forecasting loss for different pre-training tasks

Figure 4. Training losses for the classification and forecasting task for different pre-training tasks (running average with window size 200). For both classification and forecasting, all models perform better when not pre-trained on other tasks. Only the Transformer seems to achieve a comparable performance with pre-training. We speculate that the complete freeze of the feature extraction layer is too unflexible, and better results could be obtained by allowing some learning to occur at the feature extraction layer.

References

Arjovsky, M. Out of Distribution Generalization in Machine Learning. *arXiv:2103.02667 [cs, stat]*, March 2021. URL http://arxiv.org/abs/2103.02667. arXiv: 2103.02667.

Gilpin, W. Chaos as an interpretable benchmark for forecasting and data-driven modelling. *arXiv:2110.05266 [nlin]*, October 2021. URL http://arxiv.org/abs/2110.05266. arXiv: 2110.05266.

Gulrajani, I. and Lopez-Paz, D. IN SEARCH OF LOST DOMAIN GENERALIZATION. pp. 29, 2021.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Lee, K., Seo, Y., Lee, S., Lee, H., and Shin, J. Context-aware Dynamics Model for Generalization in Model-Based Reinforcement Learning. pp. 10, 2021.

Lorenz, E. N. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963. doi: 10.1175/1520-0469(1963)020⟨0130:DNF⟩2.0.CO;2. URL https://journals.ametsoc.org/jas/article/20/2/130/16956/Deterministic-Nonperiodic-Flow.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

Raissi, M., Perdikaris, P., and Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019. ISSN 00219991. doi: 10.1016/j.jcp.2018.10.045. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999118307125.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, R., Maddix, D., Faloutsos, C., Wang, Y., and Yu, R. Learning Dynamical Systems Requires Rethinking Generalization. pp. 10, 2020.

Yin, Y., Ayed, I., de Bézenac, E., Baskiotis, N., and Gallinari, P. LEADS: Learning Dynamical Systems that Generalize Across Environments. *arXiv:2106.04546 [cs, stat]*, February 2022. URL http://arxiv.org/abs/2106.04546. arXiv: 2106.04546.

A. Adapting Models for Multi-Task Learning

The main challenge of our Multi-Task setup is the need to adapt our model architectures for tasks that they were not originally built for.

In the case of RNNs, this is not particularly problematic: RNNs are built for feature extraction, and it is easy to feed these features (i.e. the last hidden outputs) to a classification or forecasting head. However, N-BEATS is designed for forecasting, so in order to adapt it to other tasks, we split its architecture into a module that is a "natural" feature extractor (the neural-basis expansion of each block) and another one that forecasts future time steps. We can then attach a simple classification head to the feature extractor. The Transformer requires a similar workaround: we consider that its encoder module is a natural feature extractor, to which we can attach a classification or forecasting head, and so the decoder module is unused.

Note that the natural number of features of a model (N_{natural}) depends on some of its hyper-parameters, for instance it is equal to the number of hidden layers for an RNN. Thus, in order to be able to set the same number of features (N_{features}) for all models (to allow comparisons), we attach to the natural feature extractor a simple linear layer, whose number of output units is N_{features}, as shown in Fig. 5.

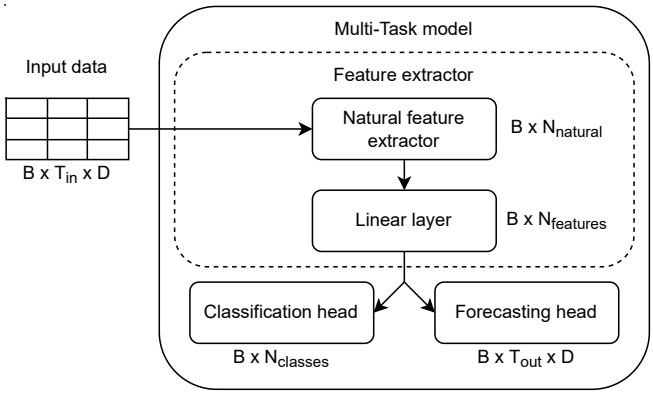


Figure 5. The inner structure of Transformer and RNNs adapted to Multi-Task. B is the batch size, D is the space dimension, and T represents a number of time steps. N_{natural} depends on the model hyper-parameters, but N_{features} can be chosen. The picture is similar for N-BEATS, but for forecasting the model works normally, without the extra linear layer and forecasting head.

B. Data Processing

All the tasks described in section 4 take only chunks of the given trajectories as input, and not the trajectories themselves. It is thus useful to formally define T_e^n as a set of trajectories of length n coming from the dynamical system e, and $chunks(T_e^t, k) = \bigcup_{t \in T_e^n} \bigcup_{i=1}^{n-k} (t_j)_{i \leq j \leq i+k}$ as the set of chunks (or slices) of length k that results from T_e^n.

For in-distrubution generalization tasks, we construct one data set per chaotic dynamical system, consisting of n trajectories of length l where each initial condition is chosen randomly and close to the original one. Formally, $T_e^n = \{Integrate(e.ODE, e.ic + r, l)\}_{i=1}^n$, where $e.ic$ is the initial condition, and r is uniform in $[-\epsilon, \epsilon[$.

For OOD generalization tasks, we can construct training data sets where the initial conditions are picked randomly from some space quadrant, and test data sets where the initial conditions come from other quadrants. In addition, we can explore OOD generalization by generating trajectories with different ODE parameters. For instance, the Lorenz system (Lorenz, 1963) depends on the parameters β, σ, ρ, and is known to exhibit chaotic behaviour for $\beta = 8/3, \sigma = 10, \rho = 28$.