

Disentanglement for improved data-driven modeling of dynamical systems

Author Efstathios (Stathi) Fotiadis
Submitted November 30th 2024

Supervisor Prof. Anil A. Bharath

A thesis submitted in fulfillment of the requirements for the degree of **Doctor of
Philosophy (PhD)** of Imperial College London

Department of Bioengineering
Imperial College London

Life can only be understood backwards, but it must
be lived forwards

Søren Kierkegaard, Journals

These 5 years have shaped my life.
It has been an educational journey as a whole.

This thesis is dedicated to my wife Katerina, and to my parents, Vasiliki and Periklis.

Declaration of Originality

This thesis presents original ideas that were developed through collaboration between the author and their collaborators. The author affirms that to the best of their knowledge, the work is original and proper attribution has been provided where necessary. The contributions made by others have been acknowledged through co-authorship of the relevant papers described in this thesis.

The work in [Chapter 4](#) was presented at the the [International Conference on Machine Learning \(ICML\)](#) Workshop on Deep Neural Models and Differential Equations in 2020 ([Fotiadis et al., 2020](#)). Part of the work in [Chapters 5](#) and [6](#) was originally presented at the [International Conference on Learning Representations \(ICLR\)](#) Workshop on Semi-Supervised Learning for Reasoning and Perception in 2021 ([Fotiadis et al., 2021](#)). That work was further developed and was accepted at the proceedings of the [ICML](#) in 2023 ([Fotiadis et al., 2023](#)).

The author is the exclusive writer of this document, unless stated otherwise. All tables and figures were produced by the author, including the formatting and styling of the document. This thesis has been submitted in fulfillment of the requirements for the degree of Philosophiae Doctor at Imperial College London only. It has not been submitted for any other degree or professional qualification.

The formatting for this PhD thesis, is adapted from the open-source repository <https://github.com/mlnotebook> ([Robinson, 2020](#)).

Copyright

I declare that this thesis is my own research work except where explicitly stated, and that I have acknowledged the work of others.

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgements

I feel incredibly lucky to cross paths with so many amazing people throughout this journey and I want to express my gratitude to all of them.

First and foremost, I want to thank my supervisor, Prof. Anil Bharath, for introducing me to the fascinating domain where physics and machine learning intersect. He mentored me every step of the way, and challenged me to think more critically. This thesis wouldn't have been the same (if at all) without his guidance and encouragement.

I'm very grateful to my collaborators - Dr. Mario Lino, Dr. Chris Cantwell, Dr. Eduardo Pignatelli, and Dr. Stef Garasto - for being part of my intellectual growth. Our conversations were invaluable, and I owe a great deal to their insights and support. I would, also, like to sincerely thank my thesis examiners, Prof. Mauricio Barahona and Dr. Nicola Perra, for their insightful feedback and thoughtful suggestions, which helped strengthen the quality of this thesis.

To my group mates (Kai, Sam, Tianhong, Konstantinos, Antoine, Christoforos, Krithika, and Henry) thank you for creating such a collaborative and supportive atmosphere. Working alongside you has been a joy. I also want to thank the many master's students who have been part of our group, especially Shunlong, Alexia, and Joe. Their fresh perspectives and enthusiasm were very important.

To my wife, Katerina: thank you for your unwavering love and for pushing me to keep going when things got tough. Your patience and belief in me have been a constant source of motivation and it was a blessing having you by my side.

I want to express my gratitude to my parents, Periklis and Vasiliki, and to my extended family. Their love, resilience, and courage gave me roots that I'm grateful for.

Special thanks go to Yasmin Frantzola for being there during the ups and downs of my PhD journey. Talking with her often helped me find clarity and stay on track. I'd also like to thank Nicola Pogson for her support.

This research was made possible by the Department of Bioengineering at Imperial College London, and completed in its entirety at Imperial. I want to thank the Department, the Imperial Trust, and the Old Centralian's Trust for their financial support, which allowed me to travel and broaden my horizons.

Accessibility

We have used vector-based figures to increase clarity for zooming-in, a color palette that is easily distinguishable by colorblind people and different line styles. We have also curated citations to refer to the corresponding conference or journal publications instead of the pre-prints where possible.

Software and Data

We provide all the necessary code to reproduce our experiments at https://github.com/stathius/wave_propagation and <https://github.com/stathius/sd-vae>. The repository contains code and instructions for generating all the datasets and training all the models presented in this work using the hyperparameters that are clearly presented in the paper. This should significantly help others reproduce our experiments. For any further clarifications, readers are encouraged to contact the corresponding author(s).

Abstract

Modeling dynamical systems is essential in science and engineering, requiring accurate predictions, robustness to varying conditions, and interpretability. Traditional data-driven methods often falter in long-term forecasting, generalization to out-of-distribution (OOD) scenarios, and providing insight into system behavior. This thesis investigates the integration of supervised disentanglement into deep learning as a solution to these challenges.

We first advance the modeling of wave propagation governed by the Saint-Venant equations using deep autoencoders (U-Nets) and tailored training strategies. These models show marked improvements in prediction accuracy. Through OOD testing, we reveal the limitations of conventional deep learning models and demonstrate that embedding domain knowledge into architectures and training can enhance performance.

Next, we extend supervised disentanglement to high-dimensional data by incorporating it into Recurrent State-Space Models (RSSMs) and U-Nets. These adaptations are evaluated on tasks like pendulum dynamics from images and wave propagation. Results show that supervised disentanglement improves model robustness and is feasible in complex settings.

We analyze the latent representations using established disentanglement metrics, finding that supervised disentanglement yields more structured, interpretable latent spaces. This can facilitate better parameter inference and understanding of the system's internal dynamics.

However, we observe trade-offs: in some high-dimensional cases, short-term prediction accuracy slightly decreases. These results emphasize the importance of context and system characteristics when applying disentanglement techniques.

Overall, this thesis shows that combining physical insights with data-driven models via supervised disentanglement leads to more accurate, generalizable, and interpretable modeling of dynamical systems. This approach has potential in fields like climate science, engineering, and healthcare, where understanding complex dynamics is vital. The work lays the groundwork for further exploration into integrating domain knowledge and learning disentangled representations in deep models for dynamic systems.

Contents

1	Introduction	26
1.1	Background and Motivation	26
1.1.1	Dynamical Systems	26
1.1.2	First-Principles Modeling	27
1.1.3	Data-Driven Methods	28
1.1.4	Limitations of Data-Driven Methods	28
1.1.5	Domain-aware data-driven methods	29
1.1.6	Disentangled Representation Learning as a Domain-Aware Data-Driven Method	30
1.2	Research Objectives	32
1.2.1	Contributions	32
1.3	Thesis Organization	34
1.4	Publications	36
1.4.1	Publications in this thesis	36
1.4.2	Related publications	36
1.4.3	Other publications	37
1.5	Concluding Remarks	38
2	Background	39
2.1	Time series forecasting	40
2.1.1	Problem formulation	40
2.1.2	Spatiotemporal forecasting	40
2.2	Probabilistic forecasting	41
2.2.1	Probabilistic Graphical Models	41
2.2.2	State Space Models	42
2.2.3	Prediction with State Space Models (SSMs)	44
2.2.4	Learning SSMs	44
2.3	Forecasting with Neural Networks	48
2.3.1	Deep Neural Networks	48
2.3.2	Deep Probabilistic Graphical Models	49
2.3.3	Long term forecasting with Deep Neural Networks (DNNs)	50
2.4	Disentangled Representation Learning	53

3	Related Work	55
3.1	Deep Learning for Spatiotemporal Forecasting	56
3.1.1	Convolutional Neural Networks	56
3.1.2	Recurrent Neural Networks	56
3.1.3	Deep State Space Models	57
3.1.4	Other Architectures	59
3.2	Physics-Aware Deep Learning	60
3.2.1	Physics-Informed Neural Networks	60
3.2.2	Neural Operators	60
3.2.3	Physics-Inspired Models	61
3.2.4	Identification of Physical Systems	61
3.3	Fluid Flow Approximation	63
3.3.1	Eulerian and Lagrangian Approaches	63
3.3.2	Wave Propagation	63
3.4	Disentanglement in Deep Learning	64
3.4.1	VAEs and Disentanglement	64
3.4.2	VAEs for Sequence Modeling	64
3.4.3	Disentanglement in Sequence Modeling	65
3.5	Concluding Remarks	66
4	Forecasting Wave Propagation with Neural Networks	67
4.1	Introduction	68
4.2	Motivation for FFNNs	71
4.2.1	Problem formulation	71
4.2.2	Markovian property of dynamical systems	71
4.2.3	Effectiveness of Feed-forward Neural Networks (FFNNs) in sequence modelling	72
4.3	Neural Architectures	73
4.3.1	Baseline: ED-LSTM	74
4.3.2	Convolutional LSTM	76
4.3.3	PredRNN++	77
4.3.4	U-Net	78
4.4	Dataset acquisition	80
4.4.1	OOD Datasets	80
4.5	Experiments	83
4.5.1	Data pre-processing and split	83
4.5.2	Training and model selection	83
4.5.3	Code and dataset availability	84
4.5.4	Model size and training time	84

4.5.5	Evaluation Metrics	84
4.6	Results	86
4.6.1	Long-term prediction	86
4.6.2	Generalization to other physical settings	91
4.6.3	Internal Representations	95
4.6.4	Physical Information in Latent Representation	96
4.7	Concluding remarks	99
5	Disentanglement for dynamical system forecasting	100
5.1	Introduction	101
5.2	Problem Formulation	103
5.2.1	Definition of OOD Dataset	104
5.3	Motivating Disentanglement	106
5.3.1	Theoretical Motivation for Disentanglement	106
5.3.2	Experimental Motivation	107
5.4	Methods	108
5.4.1	Variational Autoencoders SSMs	108
5.4.2	Forecasting with Variational Autoencoders (VAEs)	108
5.4.3	Disentangling VAEs	109
5.4.4	Supervised Disentanglement for Dynamical Systems	110
5.5	Datasets	112
5.5.1	Visualizing the Distribution Shift	114
5.6	Experiments	116
5.6.1	Modeling Decisions	116
5.6.2	Training	117
5.7	Results	120
5.7.1	Long-Term and OOD Prediction for VAE and SD-VAE	120
5.7.2	Disentangling AEs	123
5.7.3	Implications for Model Stability	124
5.7.4	Disentanglement of Representations	124
5.7.5	Linear Correlation between Latents and Parameters	126
5.7.6	Latent Space Traversals	128
5.8	Concluding remarks	129
6	Disentanglement for spatiotemporal forecasting	130
6.1	Introduction	131
6.1.1	Main Contributions	132
6.2	Pendulum Experiments	133
6.2.1	Dataset	133

6.2.2	Method	133
6.2.3	Experiments	134
6.2.4	Evaluation Metrics	135
6.2.5	Results	136
6.3	Wave Propagation	144
6.3.1	Dataset	144
6.3.2	Method	145
6.4	Experiments	148
6.4.1	Model Architecture and Training	148
6.4.2	Evaluation Protocol	149
6.5	Results	150
6.5.1	Choice of f, g	150
6.5.2	Disentanglement and invariance	150
6.5.3	In-Distribution Test Set	150
6.5.4	Smaller Tank Dataset	151
6.5.5	Larger Tank Dataset	151
6.5.6	Discussion	151
6.6	Concluding Remarks	154
7	Conclusion	156
	References	161
	Appendices	179
7.1	Proof of Lemma (2.13)	179
7.2	Wave Propagation Results	181
7.2.1	Results for all models	181
7.2.2	Baselines	181
7.3	Disentanglement Results	187

List of Figures

1.1	Spectrum of Dynamical System Modeling Approaches. Modeling methods for dynamical systems are categorized based on their reliance on first-principles modeling versus data-driven techniques. The left ellipse represents first-principles modeling, the right ellipse represents purely data-driven methods, and the overlapping central region highlights domain-aware data-driven methods.	31
2.1	Probabilistic Graphical Model for time series modeling. Stochastic variables are pictured with circles where $\mathbf{z}$ are the latent and $\mathbf{x}$ the observed variables. The relationships between latent variables are denoted with arrows. The model has first-order Markovian structure that aligns with the factorization in Equations (2.2) to (2.5).	42
2.2	Graphical models of SSM (left) and Recurrent Neural Network (RNN)(right). The two models have an identical structure, though the hidden state of the RNN is deterministic (represented with a diamond), whereas that of the SSM is stochastic (represented with a circle).	46
2.3	Graphical models of FFNNs with stochastic (left) and deterministic latent space (right). Solid lines represent the generative process $p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x} \mathbf{z})$, while the dashed lines symbolize the inference $q_{\phi}(\mathbf{z} \mathbf{x})$. In the deterministic case, the inference is assumed to have zero variance (i.e. a Dirac delta function) but the generative process can still be stochastic.	49
2.4	Glss2s modelling with RNNs and Deep SSMs. As each element in an input sequence is processed, the recurrent cell updates its hidden state, capturing a "memory" of the data seen so far. This hidden state can be unrolled without further input, allowing the recurrent cell to produce predictions or arbitrary length during the output phase.	51
2.5	Autoregressive prediction. A sliding window approach can be used with FFNNs to predict a sequence of arbitrary length. At each step fixed-length window of past observations are used to predict the future values over a fixed-length window. This window slides forward as the horizon extends and the model uses it's own predictions as input, in an autoregressive manner. . .	52

4.1	Visualization of spatiotemporal sequences of wave propagation obtained through numerical simulation of the Saint-Venant Equations (S-V equations). Frames are rendered from the raw simulations through an observation model that uses light positioned at 45° azimuth and 20° altitude.	70
4.2	The core mechanism of an Long Short-Term Memory (LSTM) network involves its ability to regulate information flow through three distinct gates: the forget gate f_t , input gate i_t , and output gate o_t . The forget gate decides which information to discard from the cell state, the input gate determines what new information to add from the input x_t , and the output gate decides what part of the current cell state to output. The updated internal and cell states h_t, c_t get passed from one time step to the next. Image adapted from Olah, 2015	73
4.3	Schematic of the encoder and decoder used in the ED-LSTM model (Sorteberg et al., 2018). Dimensions of the channels are shown at the left and top, the number of channels at each layer are shown the bottom. Downscaling/ upscaling is performed between each layer of the encoder/decoder.	74
4.4	Time unrolling of the baseline model (Sorteberg et al., 2018). The encoder provides the first LSTM with low-dimensional representation of each frame. The first LSTM then passes it's state to the second LSTM which unroll up to timestep $t = 9$, after that the third LSTM takes over, forecasting up to an arbitrary sequence length. At each step the output of the LSTMs is mapped back to pixel space by the decoder.	75
4.5	Convolutional LSTM (ConvLSTM) model. The model employs a stack of three cells. A Convolutional Neural Network (CNN) encoder before each cell is used for dimensionality reduction. Temporally the model works in sequence-to-sequence (seq2seq) fashion as described in Section 2.3.	76
4.6	Graphical model of ConvLSTM and Improved Predictive Recurrent Neural Network (PredRNN++). The two models have the same graphical structure. The hidden state is deterministic (represented with a diamond) while the number of layers K can be varied.	77
4.7	PredRNN++ Our implementation vertically stacks a convolutional layer, a max-pooling layer, and four CausalLSTM cells. A state propagated across the stacked cells, enables the flow of spatial information As with the previous RNN models we use seq2seq prediction.	78
4.8	U-Net architecture The encoder (left) progressively downsamples input using convolutional blocks and max-pooling. The decoder (right) uses bilinear interpolation for upsampling. Skip connections (dashed lines) copy feature maps without dimensionality reduction, combining coarse and fine-grained information.	79

4.9	Sample sequences from the out-of-distribution (OOD) datasets in which wavefront shape (b-c), observation model (d-e), and parameters describing the physical properties of the tank (f-h) are changed.	81
4.10	Long-term forecasting on the test set. All proposed model outperform the baseline. The U-Net outperforms the recurrent architectures indicating that FFNNs can be competitive in forecasting dynamical systems, challenging the conventional assumption that RNNs are inherently better suited for this task. The shaded area denotes one standard deviation computed across the test set.	86
4.11	Visual comparison of U-Net and PredRNN++ forecasts against ground truth. Both models produce visually similar and physically plausible forecasts on a test sequence.	88
4.12	Detailed analysis at time step 80 (correspond to the last column of Figure 4.11). Top: visual comparison. Bottom: intensity plot along the yellow line highlights subtle differences.	89
4.13	Visualizing the error of the U-Net (top) and PredRNN++ (bottom) across a sample sequence. The error is computed as the absolute difference between ground truth and model prediction. More examples can be found in the Appendix.	90
4.14	Inherent Variation of OOD Datasets. We use a flat image as a baseline to measure Root Mean Square Error (RMSE) for each frame, quantifying dataset difficulty. Since datasets vary in complexity, this inherent variation must be considered when assessing a model's generalization capabilities.	92
4.16	Generalization Under OOD Physical Conditions. Top: While the U-Net demonstrates robustness to illumination changes, its RMSE on other out-of-distribution datasets is significantly higher than on the in-distribution test set. Bottom: The U-Net appears to adapt better to illumination variations, while the PredRNN++ is better when wave speed is altered (Shallow Depth, Big & Small Tank).	93
4.17	Detailed comparison between the U-Net and the PredRNN++ on the Lines test set at time step 80. Top: visual comparison. Bottom: the intensity plot corresponds the line with the highest variation (in yellow) highlights subtle differences.. In this particular case, the PredRNN++ has missed the characteristic time constant.	94
4.18	U-Net Predictions on OOD Datasets (at time $t = 80$). Note performance variations across datasets: a single blurry wavefront in the double drop scenario, and incorrect characteristic time constants for larger and smaller tanks. This highlights the impact of dataset-specific challenges on the model's generalization.	94

4.19	Cumulative Feature Map Reconstruction for PredRNN++ (top) and U-Net (bottom) (prediction at the 80th timestep). Feature maps are ordered by the absolute magnitude of their weights from the most important to the least. PredRNN++ uses a gradual prediction buildup. In contrast, U-Net uses maps focusing on peaks (white) and troughs (black). These are combined in the final layer to produce the prediction.	95
4.20	Schematic of the Tank Size Prediction Model The model combines the U-Net encoder (Section 4.3.4) with a new head comprised of two ResNet blocks and a fully connected layer with a single output neuron. This architecture enables to do regression on the tank size using features extracted by the encoder.	98
5.1	Pendulum length parameter distributions. Training, validation, and test set trajectories are simulated with pendulum lengths drawn from $\ell \sim \mathcal{U}(1.0, 1.5)$. The OOD-Easy and OOD-Hard datasets are considered out-of-distribution because their pendulum lengths are sampled from distinct, non-overlapping uniform distributions outside the training range. Specifically, OOD-Easy uses longer pendulum lengths, resulting in slower angular velocities, while OOD-Hard employs shorter lengths that lead to faster, more challenging oscillations for prediction (see Figure 5.4).	104
5.2	The SD-VAE Model This model directly predicts future states of a system based on t observations of its phase space. During training, an additional loss term, applied to a subset of the latent variables, encourages a more disentangled representation.	109
5.3	Parameter distributions of Lotka-Volterra system (L-V system) datasets. Two-dimensional slices of the 4D distributions are shown (colors represent regions, not boundaries). OOD Easy and Hard test set parameters do not overlap with the original test set.	114
5.4	Phase space diagrams (top) and time evolution (bottom) for random pendulum trajectories. The OOD-Hard test set exhibits greater variability in θ and ω trajectories, particularly evident in the bottom row's time evolution.	114
5.5	Phase space diagrams (top) and time evolution (bottom) for random L-V system trajectories. OOD test sets demonstrate progressively wider coverage of the phase-space and time domains.	115
5.6	Position plots (top) and time evolution (bottom) of random 3-body system trajectories (body one only). OOD test sets demonstrate a wider trajectory range, reflected in greater domain coverage within the Cartesian plot.	115

5.7	Accuracy Improvement with Supervised Disentanglement. Normalized relative Mean Absolute Error (MAE) between Supervised Disentanglement VAE (SD-VAE) and the plain VAE. Negative values indicate lower SD-VAE error. Results depict the mean and one standard deviation interval across the best 5 models (selected by cumulative validation MAE).	122
5.8	Absolute MAE for Model Predictions. Performance comparison between prediction and ground-truth trajectories across different systems and datasets.	123
5.9	Disentanglement of Representations. The x-axis shows the system parameters (factors of variation in the data), and the y-axis represents the latent variables. The color scale indicates the importance weight value. Values correspond to importance weights of a tree regressor trained to predict the parameters from the latent codes of each model. Higher values (yellow) represent latent variables with stronger predictive power for the corresponding ground-truth value of each parameter. We present the top 3 models for each method and dataset. The SD-VAE latent space exhibits clear disentanglement, with highly-predictive and non-predictive latents. In contrast, the VAE encoding shows no such separation.	126
5.10	Disentanglement of Representations (a), (c) Importance weights of a random forest regressor trained to predict ground-truth parameter values using latents extracted from the model over the training set. High importance weights signify latent variables with strong predictive ability over corresponding ground-truth values. (b, d) Absolute Pearson correlation coefficient between system parameters and latents.	127
5.11	Traversals of the latent space of SD-VAE on the pendulum system. The purple line shows a linear interpolation between the latents of the two trajectories with extreme length values ($l = 1.0$ and $l = 1.5$). The blue line corresponds to the latents extracted directly from trajectories with different lengths. The real latents seem to have a linear trend which is very close to the interpolation. For this experiment, the initial conditions were kept constant across all lengths.	128
6.1	Parameter distributions for the pendulum datasets. The top subfigure illustrates how the pendulum length and gravity vary among the different datasets, emphasizing the distribution shifts in the OOD test sets. The bottom subfigure shows that the initial angle and angular velocity distributions remain consistent across all datasets.	139
6.2	Example frame sequence from the pendulum data set	140

6.3	Structural Similarity Index (SSIM) Comparison Across Pendulum Datasets. We compare RSSM and SD-RSSM on three pendulum datasets (in-distribution, OOD Easy, OOD Hard) using SSIM (higher is better). While RSSM excels in short-term similarity, SD-RSSM preserves long-term structure more effectively, especially under distribution shifts, underscoring the benefits of disentanglement.	141
6.4	Peak Signal-to-Noise Ratio (PSNR) Comparison Across Pendulum Datasets. (higher is better) comparison between RSSM and SD-RSSM across three test datasets. RSSM demonstrates superior performance in short-term predictions, whereas SD-RSSM excels in capturing long-term dynamics. As the distribution shift increases from in-distribution to OOD Easy and OOD Hard, SD-RSSM outperforms RSSM over longer ranges, highlighting the effectiveness of disentanglement in OOD scenarios.	142
6.5	Samples of Predicted Trajectories. Difference between ground truth (gray) and predictions (white) on the pendulum dataset test sets: (a) In-Distribution Test Set, (b) OOD Easy Test Set, and (c) OOD Hard Test Set. These figures illustrate how SD-RSSM better captures the longer-term dynamics of the pendulum, maintaining closer alignment with the ground truth trajectory even under distributional shifts.	143
6.6	Sample sequences from the wave propagation datasets. This figure illustrates how varying the physical size of the tank across different datasets affects the wave dynamics captured in the sequences. (a) Sequences from the Training, Validation, and In-Distribution Test sets with tank sizes ranging from 10 m to 20 m. (b) Sequences from the Larger Tank Dataset with tank sizes between 20 m and 40 m. (c) Sequences from the Smaller Tank Dataset with tank sizes between 5 m and 10 m. The differences in tank sizes lead to distinct wave patterns and dynamics observable in the sequences.	145
6.7	Wave Propagation Experiments RMSE Comparison Root Mean Square Error (RMSE, lower is better) comparison among baseline U-Net, SD-UNet, and SDI-UNet across three datasets. In the in-distribution test set and the Larger Tank, both SD-UNet and SDI-UNet consistently outperform the baseline. For the Smaller Tank, SDI-UNet better maintains baseline performance. These results highlight the effectiveness of disentanglement and invariance in enhancing model robustness and generalization.	152
7.1	Out-of-Distribution (OOD) Evaluation Results for All Models. (Continued on next page)	182
7.1	(Continued from previous page)	183
7.1	(Continued from previous page)	184

7.1	(Continued from previous page)	185
7.2	UNet and PredRNN++ against naive baselines.	186
7.3	Model predictions in phase space. Trajectories are taken from the OOD-Hard set of each system. The trajectory observations are noisy, denoted by the grey '×' markers. The orange circle and the orange and bold '×' markers denote the start and end of the ground-truth trajectories respectively	187
7.4	Model predictions in phase space. Trajectories are taken from the OOD-Hard set of each system. The trajectory observations are noisy, denoted by the grey '×' markers. The orange circle and the orange and bold '×' markers denote the start and end of the ground-truth trajectories respectively	188

List of Tables

4.1	Comparison of Feed-Forward and Recurrent Neural Networks	72
4.2	The Train/Val/Test datasets were used for training, model selection, and in-distribution evaluation. The other datasets were used to evaluate the OOD generalisation capabilities of the models.	82
4.3	Hyper-parameters of the best performing model for each architecture. The input and output of the model correspond to one full forward pass. During sampling multiple samples per sequence are used per backward pass to reduce variance. The patience is how many epochs it takes without validation error improvement before the learning rate is reduced.	83
4.4	Model size and training times. The table illustrates that U-Net, a feed-forward model, achieves the shortest epoch times, likely due to its streamlined architecture, in contrast to the recurrent nature of LSTM, ConvLSTM, and PredRNN++, which require more complex computations over sequences. Model size and training time per epoch are vital for assessing computational demands and practicality.	85
4.5	RMSE comparison of model forecasting at at various time steps. Best accuracy at each time step is highlighted in bold. Standard errors are reported across 4 models trained with different random seeds. U-Net’s advantage may stem from its feed-forward Markovian design, which allows it to make accurate one-step predictions conditioned only on the most recent frames, thus avoiding the accumulation of errors that can arise in recurrent models like PredRNN++ over long horizons.	88
4.6	Computational efficiency of models. Time per frame and speedups relative to the numerical simulator are shown. U-Net demonstrates the fastest inference time. The data-driven models take advantage of GPU parallelization while the numerical simulator runs on the CPU. While our setup consisted of commercial off-the-self hardware, results may vary across hardware configurations.	89
4.7	OOD generalization of U-Net across datasets. Values correspond to RMSE at specific time steps. Performance varies across different physical settings.	92

4.8	Predicting Tank Size from U-Net’s Latent Space. The pre-trained model demonstrates the ability to estimate tank size with relatively low error. In contrast, the randomly initialized encoder performs poorly, only slightly outperforming a dummy regressor (which always predicts the dataset’s mean tank size). However, both pre-trained and random encoders fail to extrapolate to smaller and larger tank sizes.	98
5.1	Datasets. In the L-V system and 3-body OOD test sets, at least one domain parameter is outside of the parameter range used for training.	113
5.2	Number of experiments with phase space data. Each experiment corresponds to a distinct configuration of hyperparameters. All configurations were run once, all of them with the same seed.	117
5.3	Pendulum hyperparameters	118
5.4	L-V system hyperparameters	118
5.5	3-body system hyperparameters	119
5.6	MAE average over 800 steps. Mean performance of the top 5 models of each architecture, selected by validation MAE. SD-VAE consistently outperforms VAE and all other models. Note: LSTM diverged during testing on the L-V system.	121
5.7	Model Stability. Percentage of models whose predictions diverge (to positive or negative infinity) during testing across all trajectories (out of all trained models).	124
5.8	Disentanglement Metrics. SD-VAE consistently outperforms VAE across several disentanglement metrics. Scores represent averages over the top 3 models (selected by validation MAE).	125
6.1	Hyperparameters for the RSSM and SD-RSSM models. The table lists the hyperparameter values used in our experiments. For SD-RSSM, the supervision coefficient δ is tuned between 0.01 , 0.1 , and 1 . For RSSM, we use 3 random seeds to achieve the same number of experiments (24) for a fair comparison.	135
6.2	Model comparison in observation-space pendulum. Metrics are calculated between ground truth and prediction of the models at the 800th timestep.	138
6.3	Hyperparameter configurations and the number of experiments conducted for wave propagation disentanglement.	148

Acronyms

CNN Convolutional Neural Network. 16, 56, 57, 59, 63, 69, 75, 76, 130

ConvLSTM Convolutional LSTM. 16, 22, 68, 73, 76, 77, 84–87

DCI Disentanglement - Completeness - Informativeness. 124, 125, 129

Deep SSM Deep State Space Model. 55

DL Deep Learning. 56, 91, 101

DNN Deep Neural Network. 11, 39, 44, 46, 48–51, 59, 100, 101, 107, 108, 130, 131

ELBO Evidence Lower Bound. 47, 108, 110

FFNN Feed-forward Neural Network. 12, 15, 17, 48–52, 55, 67–69, 71–73, 78, 86, 87, 99, 109

GAN Generative Adversarial Network. 59, 63

ICLR International Conference on Learning Representations. 4

ICML International Conference on Machine Learning. 4

KL Kullback-Leibler. 53, 54, 108–111, 116

L-V system Lotka-Volterra system. 18, 23, 103, 104, 112–115, 118, 121–125, 127

LSTM Long Short-Term Memory. 16, 22, 56, 57, 63, 73–76, 84–86, 116, 120, 129, 157

MAE Mean Absolute Error. 19, 23, 48, 120–123, 128

MIG Mutual Information Gap. 124, 129

MSE Mean Square Error. 48, 56, 59, 108

ODE Ordinary Differential Equation. 71, 101, 103, 110, 112

OOD out-of-distribution. 17, 18, 22, 28, 32–35, 66, 69, 80, 81, 87, 92–94, 96, 97, 99–102, 104, 105, 112, 114–116, 120, 121, 129–131, 156–158

PDE Partial Differential Equation. 56, 57, 68, 103

PGM Probabilistic Graphical Model. 39, 41, 42, 49, 50, 107, 111

PredRNN++ Improved Predictive Recurrent Neural Network. 16–18, 22, 68, 73, 77, 78, 84–88, 90, 91, 93–95

PSNR Peak Signal-to-Noise Ratio. 20, 134–138, 142

RMSE Root Mean Square Error. 17, 22, 84, 86–88, 91–93, 96, 99, 134, 149

RNN Recurrent Neural Network. 15–17, 46, 48, 50, 51, 55–60, 67–69, 71–74, 78, 84, 86, 87, 99

RSSM Recurrent State Space Model. 33–35, 157

S-V equations Saint-Venant Equations. 16, 68–70, 80

SAP Attribute Predictability Score. 124, 129

SD-VAE Supervised Disentanglement VAE. 19, 23, 111, 116–129

seq2seq sequence-to-sequence. 16, 50, 76–78

SSIM Structural Similarity Index. 20, 134–138, 141

SSM State Space Model. 11, 13, 15, 39, 42–46, 48–51, 74, 108, 109

STF spatiotemporal forecasting. 40, 69, 71–73, 76

VAE Variational Autoencoder. 13, 19, 23, 32, 34, 35, 53, 55, 64, 65, 101, 102, 107–111, 116–126, 128, 129, 157

Introduction

1.1 Background and Motivation

1.1.1 Dynamical Systems

Dynamical systems are processes that evolve over time under deterministic or stochastic rules ([Strogatz, 2018](#)). They are fundamental in understanding and predicting a wide range of natural and engineered processes, including weather patterns, mechanical systems, biological populations, and financial markets ([Brunton & Kutz, 2019](#)). Accurate modeling and forecasting of dynamical systems enable us to analyze system behavior, design control strategies, and make informed decisions in various scientific and engineering applications.

For example, in engineering, dynamical systems modeling is essential for the design and control of mechanical and electrical systems, enhancing stability and efficiency ([Ogata, 2001](#)). In biology, modeling the dynamic behaviors of cells, organs, or whole organisms provides insights into disease progression, drug interactions, and biological responses to external stimuli ([Alon, 2006](#)). Financial markets, characterized by intricate interactions and feedback loops, are examples of complex dynamical systems where accurate modeling informs investment strategies and risk assessment ([Mantegna & Stanley, 1999](#)). Climate science relies heavily on dynamical systems to understand atmospheric interactions, aiding in weather forecasting and climate change predictions ([Baines, 2008](#)). Even social systems, from the spread of information to population dynamics, are studied through the lens of dynamical systems ([Newman, 2010](#)).

Given the diverse applications and complexities of dynamical systems, modeling approaches can be viewed along a spectrum ranging from traditional first-principles modeling to purely data-driven methods, with domain-aware data-driven methods bridging the gap between the two (see [Figure 1.1](#)). This spectrum illustrates how different methods balance reliance on domain knowledge versus data-driven techniques in capturing system dynamics. First-principles modeling relies on fundamental physical laws and domain expertise to construct ODEs and PDEs that describe system behavior. In contrast,

data-driven methods utilize machine learning techniques to learn system dynamics directly from data, often without explicit consideration of underlying physical principles. Domain-aware data-driven methods seek to combine these approaches by integrating physical laws or constraints into data-driven models, thereby enhancing their accuracy and generalizability.

1.1.2 First-Principles Modeling

Historically, the exploration of dynamical systems has relied heavily on first-principles modeling using ordinary differential equations (ODEs) and partial differential equations (PDEs) (Hairer et al., 1993). These mathematical constructs, built from fundamental physical laws and domain knowledge, have long served as the cornerstones of our understanding (Evans, 2010). First-principles models offer deep insights into the underlying mechanisms of dynamical systems and provide interpretability and predictive capabilities across a range of conditions, assuming the models accurately capture the system dynamics.

However, traditional methods face inherent limitations that constrain their applicability to complex, real-world scenarios. These limitations include:

Model Complexity and Incompleteness: Many real-world systems are highly nonlinear and multidimensional, and exhibit chaotic behavior, making mathematical modeling challenging (Strogatz, 2018). Even models grounded in fundamental physical principles may not capture the full spectrum of a system's behavior and simplifying assumptions may omit critical dynamics (Haken, 2012).

Computational Demands: High-resolution numerical methods, especially those involving multidimensional PDEs, are computationally intensive. This computational burden becomes a significant bottleneck for time-critical applications such as weather forecasting, fluid dynamics, or control systems (Karniadakis & Sherwin, 2005).

Unmodeled or Poorly Understood Systems: In many complex systems explicit mathematical models may not exist, especially in emerging fields or domains where data and understanding is scarce. This makes traditional modeling approaches infeasible and requires data-driven methods capable of learning system behaviors directly from available observations (Brunton & Kutz, 2019).

Given these challenges, there is a pressing need for more flexible and adaptive modeling approaches that can handle complexity, leverage available data effectively, and provide accurate predictions in real-world scenarios.

1.1.3 Data-Driven Methods

At the other end of the spectrum (see Figure 1.1), purely data-driven methods leverage machine learning, particularly deep learning, to model dynamical systems directly from data without explicit knowledge of the underlying equations (Goodfellow et al., 2016). These methods approximate complex nonlinear relationships and have been applied to time series prediction, system identification, and control (Brunton & Kutz, 2019; Chen et al., 2018b; Raissi et al., 2017b). A major reason for the increased interest in data-driven methods for modeling dynamical systems is the abundance of data in the modern era (Brunton & Kutz, 2019).

Data-driven modeling offers several key advantages over traditional methods. Machine learning models, especially deep neural networks, can capture relationships directly from data, providing useful representations of real-world systems that can be difficult to describe analytically (Goodfellow et al., 2016). This makes them valuable for studying complex or poorly understood systems (Brunton et al., 2020). Furthermore, machine learning models can uncover underlying patterns or structures in data that might be non-obvious, bringing new insights into system dynamics (Ljung et al., 2020).

Once trained, data-driven models can generate rapid predictions, often orders of magnitude faster than traditional simulations (Bouallège et al., 2024). This makes them suitable for applications where numerical methods may be computationally prohibitive (Fotiadis et al., 2025). Additionally, data-driven methods can be designed to handle noisy and incomplete data effectively through various techniques (Huber, 1992; Shorten & Khoshgoftaar, 2019; Srivastava et al., 2014). In contrast, many traditional numerical methods assume precise initial conditions and parameters and may be sensitive to data imperfections, which can lead to significant errors in simulations (Tarantola, 2005).

1.1.4 Limitations of Data-Driven Methods

Despite their potential, purely data-driven machine learning techniques face challenges when applied to dynamical systems:

Long-Term Prediction Accuracy and Stability: Data-driven models often suffer from error accumulation over time, leading to divergence from true trajectories in long-term forecasts (Girin et al., 2020). This issue is especially pronounced in systems with sensitive dependence on initial conditions and chaotic dynamics, where small errors can grow exponentially (Pathak et al., 2018). Stability is critical in applications such as climate modeling, control systems, and financial forecasting, where accurate and reliable predictions over extended periods are essential.

Out-of-Distribution OOD Generalization: Data-driven models are susceptible to overfit-

ting, learning noise in the training data rather than the true underlying dynamics (Goodfellow et al., 2016; Hendrycks & Dietterich, 2019). Overfitted models may struggle to generalize beyond the conditions present in the training data (Snoek et al., 2019; Zhang et al., 2021). Predicting system behavior under a wide range of operating conditions, including rare or extreme events, is essential in dynamical systems (Baker et al., 2019). However, models often exhibit poor performance when faced with out-of-distribution inputs, limiting their applicability in real-world scenarios where conditions can vary significantly.

Large Data Requirements and Complexity of Training: Data-driven models often require large amounts of high-quality data to learn accurate representations of the underlying dynamics (Brunton & Kutz, 2019). In many practical situations, collecting sufficient data is expensive, time-consuming, or even impossible, especially for systems where measurements are difficult or intrusive (Proctor et al., 2016). Training complex machine learning models can also be computationally intensive, requiring significant resources and time (Schmidhuber, 2015). This demand can be a barrier to practical implementation, especially when rapid model updates are needed in response to new data.

Ignoring Physical Laws and Interpretability: Purely data-driven models may violate known physical laws or constraints, leading to physically implausible predictions (Krishnapriyan et al., 2021; Raissi et al., 2019). Incorporating prior knowledge and ensuring that models adhere to known laws poses significant challenges (Willard et al., 2020). Additionally, deep learning models are often considered black boxes, making it difficult to understand their predictions and extract meaningful insights about system dynamics (Rudin, 2019). This lack of interpretability can hurt the adoption of these models in applications where understanding the reasoning behind predictions is necessary.

Overall, these limitations highlight the need for approaches that combine data-driven techniques with physical insights, leading to hybrid models that leverage the strengths of both methodologies.

1.1.5 Domain-aware data-driven methods

Bridging the gap between first-principles modeling and purely data-driven methods are **domain-aware data-driven methods**. These approaches incorporate domain-specific knowledge into data-driven models to address many challenges faced by purely data-driven techniques. By embedding physical laws, constraints, or known system properties, models can be guided toward more accurate and physically consistent representations (Karniadakis et al., 2021).

Several methods have been proposed to achieve this integration, which can be broadly categorized into two groups:

Methods Informed by Known Equations: This category includes methods that explicitly incorporate known governing equations into the modeling process. Techniques like Physics-Informed Neural Networks (PINNs) (Raissi et al., 2017c; 2019) integrate known physical laws by embedding differential equations directly into the loss function during training. By enforcing the governing equations of the system, these models ensure that predictions are physically consistent and adhere to known dynamics.

Methods Incorporating Physical Priors or System Properties: The second category consists of methods that incorporate physical priors or system properties without explicitly embedding the governing equations. This includes approaches that integrate physical principles, such as conservation laws, symmetries, or invariances, into the model architecture or training process (Otto et al., 2023). For example, Hamiltonian and Lagrangian Neural Networks (Cranmer et al., 2020; Greydanus et al., 2019) embed energy conservation and symplectic structures to model mechanical systems.

One promising direction for incorporating domain knowledge into machine learning models is through disentangled representation learning. In this thesis, we explore how disentanglement can utilize domain knowledge by embedding information about the system without requiring explicit knowledge of the governing equations. This positions our work within the broader context of integrating physical knowledge into machine learning models, complementing existing methods and addressing some of the key limitations of purely data-driven approaches.

1.1.6 Disentangled Representation Learning as a Domain-Aware Data-Driven Method

Disentangled representation learning is an approach within machine learning that aims to create representations where distinct underlying factors of variation in the data are captured by separate latent variables. The core idea is to learn a model that can identify and isolate these individual factors, allowing for more interpretable and meaningful features. In the context of dynamical systems, this means the model attempts to separate system parameters (like physical constants) from dynamic states (such as positions or velocities).

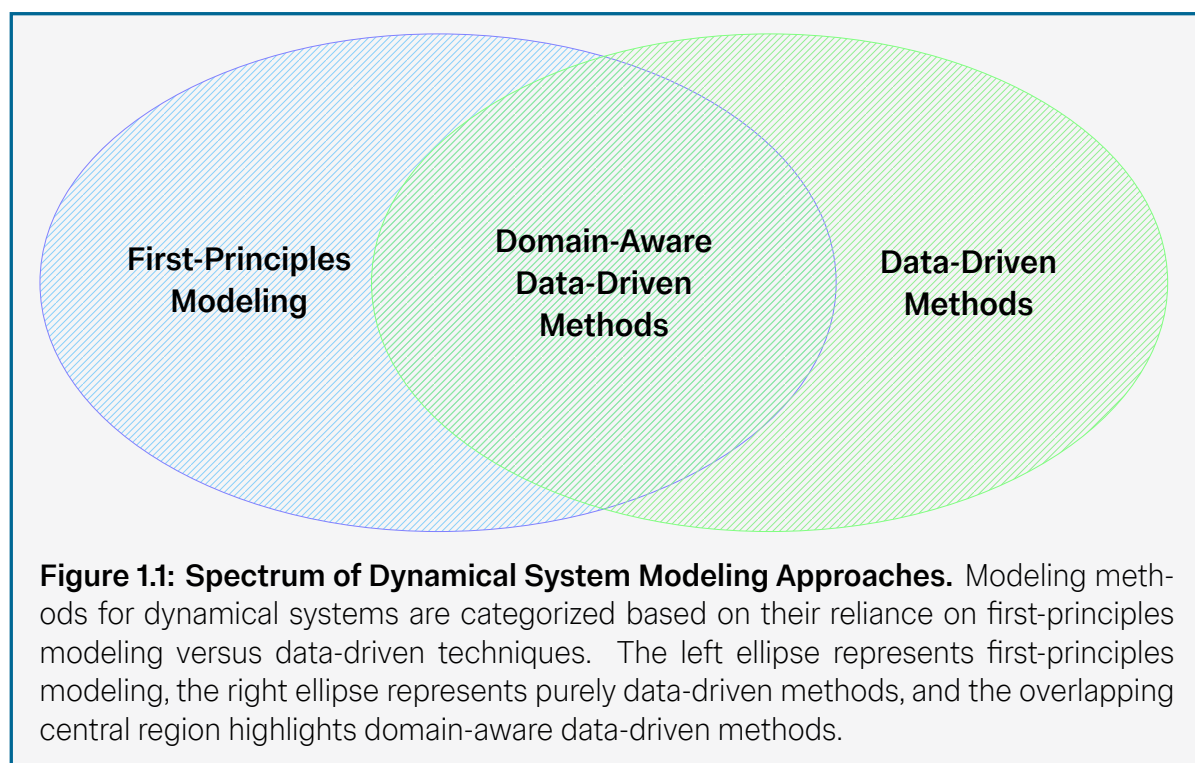
Disentanglement has been proposed as a means to improve both the interpretability and generalization of deep learning models (Bengio et al., 2013; Higgins et al., 2017). By separating underlying factors of variation in data into distinct latent variables, models can capture meaningful and interpretable features (Eastwood & Williams, 2018). This structure allows for better generalization to new combinations of factors and facilitates tasks such as transfer learning and controllable generation (Locatello et al., 2019a).

Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) have been ex-

tensively studied for learning disentangled representations. Modifications to the VAE framework, such as the β -VAE (Higgins et al., 2017), introduce constraints or regularization terms that encourage disentanglement in the latent space (Tschannen et al., 2018). However, achieving disentanglement in an unsupervised manner is challenging and often relies on strong assumptions about the data distribution (Locatello et al., 2019a). Incorporating supervision, even partially, can significantly improve the learning of disentangled representations (Shu et al., 2020). By providing information about the factors of variation, models can be guided to encode these factors explicitly in the latent space, enhancing interpretability and potentially improving performance on downstream tasks.

In the context of dynamical systems, disentangling representations allows models to separate system parameters (e.g., physical constants) from dynamic states (e.g., positions, velocities) (Iten et al., 2020). This separation improves long-term prediction accuracy, robustness to new conditions, and interpretability of learned representations.

In this thesis we explore a supervised disentanglement approach, positioning it as a domain-aware data-driven method. This means we are integrating domain-specific knowledge (the physical parameters) into deep learning models for dynamical systems, aiming to combine the strengths of first-principles modeling (understanding of underlying mechanisms) with data-driven techniques (learning complex patterns from data). The goal of this disentanglement is to enhance model generalization, stability in long-term predictions, and interpretability, thereby addressing some key limitations of purely data-driven methods.



1.2 Research Objectives

This thesis is situated within the broader context of integrating physical knowledge into machine learning models. Our approach is to integrate knowledge about the system into deep learning models via supervised disentanglement. We focus on improving long-term prediction accuracy, robustness to OOD scenarios, and interpretability of learned representations in both low-dimensional systems with observable states and high-dimensional spatiotemporal data. Specifically, our objectives are:

- **Develop techniques for supervised disentanglement** that incorporate physical priors into deep learning models to improve interpretability and generalization.
- **Enhance long-term prediction accuracy and stability** by reducing error accumulation over time.
- **Improve robustness to OOD scenarios** to generalize beyond training conditions by explicitly encoding physical parameters.
- **Increase the interpretability** of learned representations the model's behavior.

1.2.1 Contributions

The primary contributions of this thesis are as follows:

Advancing State-of-the-Art in Wave Propagation Modeling: We develop deep learning models that improve the state-of-the-art performance in predicting wave propagation governed by the Saint-Venant equations. We repurposed U-Net, a convolutional neural network primarily designed for image segmentation that has an encoder-decoder architecture (Ronneberger et al., 2015). By carefully designing training strategies, we achieve significant improvements in wave prediction accuracy. We perform a thorough OOD analysis, highlighting issues with deep learning models in capturing complex spatio-temporal dynamics (Chapter 4). Our work demonstrates how integrating domain knowledge through architectural design and training practices can enhance model performance.

Introducing Supervised Disentanglement into VAEs for Dynamical Systems: We propose a method to incorporate supervised disentanglement into VAEs (Kingma & Welling, 2014), encouraging the model to learn latent representations that separately encode system parameters from dynamics (Chapter 5). This method involves adding a supervised regression loss on the latent space to guide the encoding of known physical parameters. By guiding the learning process with known physical parameters, we aim to separate system

parameters from dynamic states in the latent space, enhancing interpretability and generalization (Iten et al., 2020; Locatello et al., 2019a). We demonstrate that this approach can improve long-term prediction accuracy, generalization to OOD scenarios and improved explainability.

Scaling Supervised Disentanglement to High-Dimensional Observations: We adapt our supervised disentanglement approach to convolutional neural network architectures to handle high-dimensional spatiotemporal data. Specifically, we integrate supervised disentanglement into the [Recurrent State Space Model \(RSSM\)](#) (Hafner et al., 2019) and U-Net architectures, evaluating their performance on modeling pendulum motion from images and wave propagation dynamics ([Chapter 6](#)). Our work demonstrates the applicability of supervised disentanglement in complex, high-dimensional settings.

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2: Background

We provide the necessary technical background on dynamical systems, machine learning models for time series prediction, probabilistic graphical models, and variational inference. We discuss how [VAEs](#), [RSSM](#), U-Net and other deep learning models can be applied to modeling dynamical systems.

Chapter 3: Related Work

We review the literature on machine learning approaches to dynamical systems modeling, focusing on methods that incorporate physical knowledge or structure, such as physics-informed neural networks ([Raissi et al., 2017b](#)) and domain-aware methods ([Chen et al., 2018b](#); [Greydanus et al., 2019](#)). We, further, discuss the strengths and limitations of these methods, situating our work within this context. We discuss previous work on disentangled representations and their application to domain-aware modeling of dynamical systems.

Chapter 4: Forecasting Wave Propagation with Neural Networks

In this chapter, we explore the use of deep learning models to predict wave propagation dynamics, a high-dimensional dynamical system governed by the Saint-Venant equations. We discuss the challenges associated with modeling such systems, including the nonlinear and dispersive nature of waves and the influence of system parameters ([Aloev et al., 2023](#); [Coron et al., 2007](#)). We present our models, primarily based on the U-Net architecture, for predicting wave evolution from spatiotemporal data. By carefully designing the training strategies we achieve improvements over the state-of-the-art performance. We also perform [OOD](#) analysis, recognizing robustness issues with deep learning models in capturing complex spatiotemporal dynamics.

Chapter 5: Disentanglement for dynamical system forecasting

We introduce our method for integrating supervised disentanglement into VAEs for dynamical systems modeling. We formulate the loss function, discuss implementation details, and explain how this approach can improve long-term prediction accuracy and generalization. We provide theoretical insights into why disentanglement aids in modeling dynamical systems, drawing on concepts from representation learning (Bengio et al., 2013) and information theory (Tishby & Zaslavsky, 2015). We test our method on three dynamical systems of increased complexity, showcasing that it can produce more interpretable representations and lead to better long-term and OOD prediction accuracy.

Chapter 6: Disentanglement for Spatiotemporal Forecasting

We extend our supervised disentanglement approach to high-dimensional spatiotemporal data by incorporating it into CNN-based architectures. We test our method in two scenarios. First, we integrate supervised disentanglement into the RSSM (Hafner et al., 2019) and evaluate its performance on image sequences of pendulum motion. Secondly, we incorporate supervised disentanglement into the U-Net architecture to model wave propagation dynamics from high-dimensional observations. In both cases, we analyze the model's ability to improve long-term predictions and robustness to distributional shifts, demonstrating that disentanglement can enhance model interpretability and accuracy for long-term predictions.

Chapter 7: Conclusions

We summarize our findings, discussing the implications of supervised disentanglement for modeling dynamical systems using deep learning. We highlight the benefits, limitations, and potential applications of our approach. We propose directions for future research, such as exploring semi-supervised or unsupervised disentanglement methods, integrating physical constraints directly into the learning process, and applying the method to more complex systems.

1.4 Publications

This section contains a compilation of research papers that I contributed to throughout my doctoral journey. These papers were published in various academic outlets, including journals and conference proceedings.

1.4.1 Publications in this thesis

This is the list the publications that form the main contributions of this thesis and are presented in [Chapter 4](#), [Chapter 5](#) and [Chapter 6](#).

Fotiadis, S., Pignatelli, E., Lino, M., Cantwell, C., Storkey, A., & Bharath, A. A. (2020). Comparing recurrent and convolutional neural networks for predicting wave propagation. **International Conference on Learning Representations (ICLR) 2020 Workshop on Deep Learning and Differential Equations.**

Fotiadis, S., Lino, M., Cantwell, C., & Bharath, A. (2021). Disentangling ODE parameters from dynamics in VAEs. **International Conference on Machine Learning (ICML) 2021 Workshop on Semi-Supervised Learning.**

Fotiadis, S., Lino, M., Hu, S., Garasto, S., Cantwell, C., & Bharath, A. A. (2023). Disentangled Generative Models for Robust Prediction of System Dynamics. **Proceedings of the International Conference on Machine Learning (ICML) 2023**, 10222–10248.

1.4.2 Related publications

This is a list of publications on the topic of modelling dynamical systems, that do not form part of the thesis.

Lino, M., Cantwell, C., Fotiadis, S., Pignatelli, E., & Bharath, A. A. (2020). Simulating surface wave dynamics with convolutional networks. **NeurIPS 2020 Workshop on Interpretable Inductive Biases and Physically Structured Learning.**

Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. D. (2022a). Multi-scale rotation-equivariant graph neural networks for unsteady Eulerian fluid dynamics. **Physics of Fluids**, 34(8).

Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. (2022b). REMuS-GNN: A

Rotation-Equivariant Model for Simulating Continuum Dynamics. **International Conference on Learning Representations (ICLR) 2022 Workshop on Geometrical and Topological Representation Learning.**

Lino, M., Fotiadis, S., Bharath, A., & Cantwell, C. (2022c). Towards fast simulation of environmental fluid mechanics with multi-scale graph neural networks. **International Conference on Learning Representations (ICLR) Workshop on AI for Earth and Space Science.**

Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. D. (2023). Current and emerging deep-learning methods for the simulation of fluid dynamics. **Proceedings of the Royal Society A**, 479(2275), 20230058.

This is a publication I contributed to during my internship at NVIDIA, Santa Clara, CA related to modelling multi-scale physical phenomena like weather.

Fotiadis, S., Brenowitz, N. D., Geffner, T., Cohen, Y., Pritchard, M., Vahdat, A., & Mardani, M. (2025). Adaptive Flow Matching for Resolving Small-Scale Physics. **Proceedings of the International Conference on Machine Learning (ICML).**

1.4.3 Other publications

These are publications I contributed to during my internship at MediaTek Research, Cambridge.

Das*, A., Fotiadis*, S., Batra, A., Nabiei, F., Liao, F., Vakili, S., Shiu, D.-S., & Bernacchia, A. (2023). Image generation with shortest path diffusion. **Proceedings of the International Conference on Machine Learning (ICML) *equal contribution.**

Garcia, J. R., Freddi, F., Fotiadis, S., Li, M., Vakili, S., Bernacchia, A., & Hennequin, G. (2023). Fisher-Legendre (FishLeg) optimization of deep neural networks. **Proceedings of the International Conference on Learning Representations (ICLR) 2023, Oral.**

Xia, G., Danier, D., Das, A., Fotiadis, S., Nabiei, F., Sengupta, U., & Bernacchia, A. (2023). Score Normalization for a Faster Diffusion Exponential Integrator Sampler. **NeurIPS 2023 Workshop on Diffusion Models.**

1.5 Concluding Remarks

In this introduction, we have outlined the critical role that dynamical systems play in modeling and predicting complex processes across a wide range of scientific and engineering disciplines. While traditional modeling approaches based on differential equations have been foundational, they face significant limitations when dealing with highly nonlinear, multidimensional, or poorly understood systems. The advent of data-driven methods, particularly deep learning, offers powerful alternatives capable of capturing intricate patterns directly from data without explicit knowledge of the underlying equations. However, purely data-driven models often struggle with long-term prediction accuracy, generalization to out-of-distribution scenarios, large data requirements, and a lack of interpretability.

To address these challenges, integrating domain-specific knowledge into machine learning models emerges as a promising avenue. By embedding physical laws, constraints, or known system properties, models can be guided toward more accurate and physically consistent representations. Disentangled representation learning, especially when supervised, offers a way to incorporate domain knowledge into deep learning models by separating underlying factors of variation in data into distinct latent variables. This approach enhances interpretability, improves generalization, and facilitates tasks such as transfer learning and controllable generation.

In this thesis, we aim to bridge the gap between purely data-driven approaches and traditional physics-based methods by developing domain-aware data-driven models for dynamical systems. Our contributions include advancing state-of-the-art modeling techniques, introducing supervised disentanglement into variational autoencoders, and scaling these methods to high-dimensional observations. By doing so, we contribute to the development of more accurate, generalizable, and interpretable models for dynamical systems.

We hope that the techniques and insights presented here will inspire further advancements in the modeling and understanding of complex dynamical systems across various fields. Future research could explore semi-supervised or unsupervised disentanglement methods, integrate physical constraints directly into the learning process, and apply these methods to more complex systems.

Chapter 2

Background

In this section, we lay the groundwork for the rest of the thesis by defining key concepts. We will establish the problem of time series forecasting and introduce the framework of [Probabilistic Graphical Models \(PGMs\)](#) which will serve as a consistent reference throughout the thesis. Next, we will introduce [State Space Models \(SSMs\)](#), a specific instance of [PGMs](#) widely used for analyzing time series. The discussion also touches on [Deep Neural Networks \(DNNs\)](#), that can be integrated into [SSMs](#) to learn complex and non-linear temporal dynamics from data. Furthermore, we will cover the main types of [DNNs](#) and discuss long-term forecasting methods.

2.1 Time series forecasting

2.1.1 Problem formulation

Time series forecasting involves using a model to **forecast**, or **predict**, future values in a time series based on previously observed values. This framework, also applicable to dynamical systems, has been extensively reviewed in the literature (Benidis et al., 2022).

The problem of forecasting is set up as follows. Given an observed variable $\mathbf{x} \in \mathbb{R}$, indexed by discrete time $\mathbf{t}$, evolving under some unknown dynamics and a sequence of past observations $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{t-1})$, the aim is to predict the next k future values $\mathbf{x}_{t:t+k}$. Using a probabilistic formulation allows to naturally embrace uncertainty in forecasting time series. In the general setting, time series forecasting seeks to approximate the following conditional probability distribution:

$$p(\mathbf{x}_t, \dots, \mathbf{x}_{t+k} \mid \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) \quad (2.1)$$

using a model p_θ parameterized by θ (Benidis et al., 2022). The task is termed **univariate** when $\mathbf{x} \in \mathbb{R}$ and **multivariate** for $\mathbf{x} \in \mathbb{R}^n, n \geq 2$.

The often complex and non-stationary nature of time series data makes forecasting a challenging task for machine learning (Chomiak & Hu, 2024). Directly estimating the distribution in Equation (2.1) is intractable except in the most trivial cases. Therefore, practical approaches rely on modeling decisions that simplify the conditional distribution while maintaining good predictive performance.

2.1.2 Spatiotemporal forecasting

Time series forecasting is often univariate, considering a single real-valued variable. However, many applications require predicting multiple values simultaneously. Modeling multiple variables and their typically unknown correlations is called multivariate forecasting.

A special case of multivariate forecasting occurs when the data exhibit some form of spatial structure over a grid. **Spatiotemporal forecasting (STF)** techniques leverage this structure to better capture dynamics in both space and time. Typically, the data resides on a k -dimensional grid, represented as $\mathbf{x}_t \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_k}$ for a k -dimensional grid. The grid does not need to be equidistant, and flexible structures such as graphs have been employed (Battaglia et al., 2018; Lino et al., 2022a), while recently other topological structures have been explored (Papamarkou et al., 2024). Extensions to higher dimensions are also crucial for forecasting complex systems like weather prediction (Price et al., 2023).

2.2 Probabilistic forecasting

2.2.1 Probabilistic Graphical Models

PGMs consists of a framework that can be used for modelling complex probability distributions like Equation (2.1). They offer tools for representing dependencies in data, including temporal ones (Barber, 2011; Koller & Friedman, 2009). In their graphical structure they use nodes to denote stochastic variables and edges to mark the relationships between them. Variables be categorized as **observed** or **hidden**, the latter also called **latent** or unobserved. This is important for time series forecasting, as we're often interested in modeling how both observed and unobserved factors influence forecasting. PGMs that contain unobserved variables are called **latent variable models**.

Graphical models can be described as directed acyclic graphs (DAGs), i.e. graphs where the edges have a direction, indicated by arrows, and there are no cycles (i.e. you can't follow the arrows and end up back at the starting node). These properties make DAGs particularly well-suited for modeling dynamical systems. Directionality ensures consistency with real-world notions of causality, where effects cannot precede their causes while the absence of cycles prevents feedback loops that could lead to non-causal behavior in the model. Importantly, DAGs allow us to factorize complex joint probability distributions into a product of smaller conditional distributions, where each variable depends only on its parent nodes in the graph. This factorization simplifies both the representation and the computations involved in learning the model.

As an example, we factorize the joint of the latent variable model in Figure 2.1. We start by factorizing the joint in the following form, where variables $\mathbf{z}$ and $\mathbf{x}$ alternate:

$$\begin{aligned} p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) &= p(\mathbf{z}_1) \cdot p(\mathbf{x}_1 | \mathbf{z}_1) \cdot p(\mathbf{z}_2 | \mathbf{x}_1, \mathbf{z}_1) \cdot p(\mathbf{x}_2 | \mathbf{x}_1, \mathbf{z}_{1:2}) \dots \\ &= \prod_{t=1}^T p(\mathbf{z}_t | \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) \cdot p(\mathbf{x}_t | \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) \end{aligned} \quad (2.2)$$

From the graph we can read the following modelling (and independency) assumptions of the model:

$$p(\mathbf{z}_t | \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = p_\phi(\mathbf{z}_t | \mathbf{x}_{t-1}) \quad (2.3)$$

$$p(\mathbf{x}_t | \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = p_\theta(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{z}_t) \quad (2.4)$$

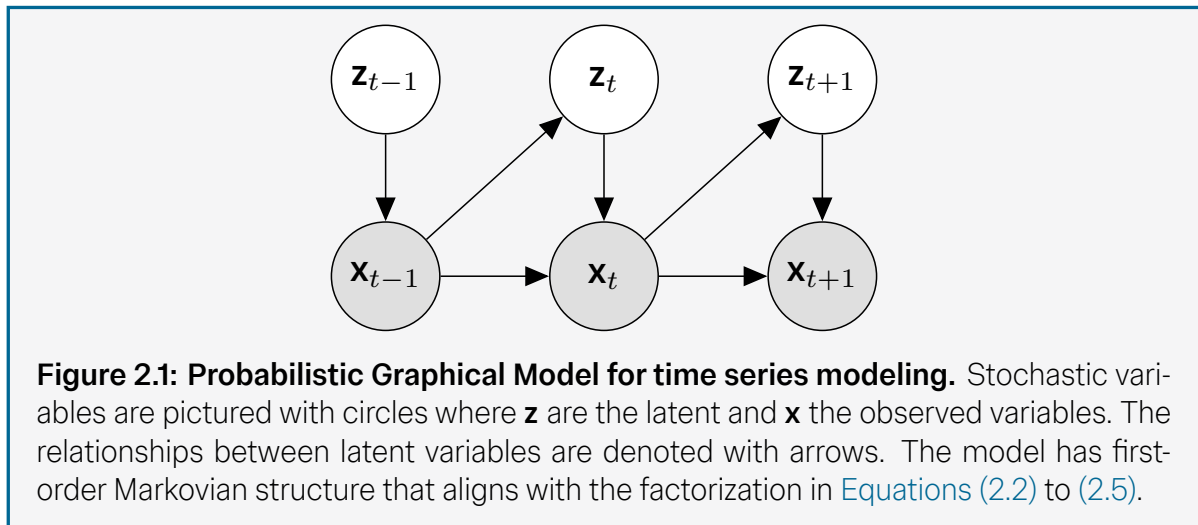
leading to a simplified model for the factorization of the joint distribution:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^T p_{\phi}(\mathbf{z}_t \mid \mathbf{x}_{t-1}) \cdot p_{\theta}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{z}_t) \quad (2.5)$$

where θ, ϕ denote the parameters of the model.

A central task in PGMs is **inference**, where we estimate the latent states of a system based on observed data. This necessitates **learning** a suitable model $p_{\phi}(\mathbf{z} \mid \cdot)$ from the data. Finally, **generation** allows us to use the learned model to synthesize new samples that reflect the patterns in the training data, drawing from the conditional distribution $p_{\theta}(\mathbf{x} \mid \cdot)$.

Overall, the use of PGMs in modeling discrete-time, continuous-valued time series has been extensively studied (Durbin & Koopman, 2001). Their ability to represent temporal causality combined with the flexibility to integrate knowledge about system dynamics, result in computationally efficient learning and inference algorithms. This makes DAGs a practical choice for tackling complex time series forecasting problems, especially for dynamical systems. As we will see in Section 2.2.2 predicting the next step in a time-series can be seen as a generation step according to Equation (2.4).



2.2.2 State Space Models

SSMs are a class of discrete-time continuous-valued latent variable DAGs developed for forecasting and control of dynamical systems (Durbin & Koopman, 2001; Kalman, 1960). State space refers to the collection of all possible configurations a dynamical system can occupy. Each unique state corresponds to a specific point within this space. For example, in the case of an idealized pendulum its state is fully described by its angle and angular velocity. Therefore, the state space consists of all possible combinations of angle and velocity, which can be visualized as an ellipse. In the context of statistics and machine learning SSMs are

also known as Hidden Markov Models (Rabiner, 1989).

SSMs incorporate several key assumptions about the systems they model. Firstly, they assume **latent dynamics**, i.e. latent variables fully encapsulate the dynamics of the system. Next, they presuppose **first-order Markovian** dynamics, where Markovian means that the current state of a system depends only on a finite set of previous states. In other words, a short history of states holds enough information to predict the system's future. The number of required previous states defines the order of the Markovian dynamics. Furthermore, SSMs include an **observation model** that explicitly links the latent states to the observed variables. Lastly, SSMs assume the presence of **additive noise** in both the system dynamics and the observation process. While it lies out of the scope of this thesis, we note that input variables can also be used to model control mechanisms. These assumptions are reflected graphically in Figure 2.1 and in the state and observations equations that define SSMs:

$$\mathbf{z}_t = \mathcal{F}(\mathbf{z}_{t-1}) + \boldsymbol{\varepsilon}_t \quad (2.6)$$

$$\mathbf{x}_t = \mathcal{G}(\mathbf{z}_t) + \boldsymbol{\eta}_t \quad (2.7)$$

where $\mathbf{x}_t \in R^n$, $\mathbf{z}_t \in R^d$ are the observed and latent variables respectively, $\boldsymbol{\varepsilon} \in R^d$ and $\boldsymbol{\eta} \in R^n$ are the state and observation noise processes, while $\mathcal{F}$ and $\mathcal{G}$ are vector functions. The state Equation (2.6) models evolution of the latent state, while the observation Equation (2.7) links these latent states to observations.

In probabilistic terms, the graphical model of the SSM in Figure 2.2 and Equations (2.6) to (2.7) imply the following independency assumptions:

$$p_\phi(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}) = p_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \quad (2.8)$$

$$p_\theta(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \quad (2.9)$$

leading to the factorization of the joint distribution for a series of length T :

$$p_{\theta,\phi}(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^T p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \cdot p_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \quad (2.10)$$

Using the Bayes rule and the conditional independence of Equation (2.8) we deduce that:

$$p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}) = \prod_{t=1}^T p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \quad (2.11)$$

It's worth noting that Equation (2.6) represents a first-order autoregressive (AR) process, but higher-order AR structures can be accommodated by expanding the state vector. For ex-

ample, we could define $\tilde{\mathbf{x}}_t = [\mathbf{x}_t, \mathbf{x}_{t-1}]$ to capture second-order dynamics. Furthermore, the noise terms allow **SSMs** to model stochastic processes of unknown dynamics. While this thesis focuses on forecasting systems governed by **deterministic** ODEs and PDEs (implying $\varepsilon_t = 0$), we still consider the possibility of **observational noise** in our models.

2.2.3 Prediction with SSMs

Before moving on to discuss how such a model can be learned, it is illuminating to understand how it can be used to solve our initial forecasting problem of Equation (2.1). Since **SSMs** consider one-step prediction, the distribution of interest is $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})$. Assuming we have access to p_θ, p_ϕ the forecasting can start from a single observation. By using the law of total probability to integrate out the latent variables we get:

$$\begin{aligned}
 p_\theta(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}) &= \int_{\mathbf{z}_{1:t}} p_\theta(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) \cdot p_\phi(\mathbf{z}_{1:t} \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_{1:t} \\
 &= \int_{\mathbf{z}_{1:t}} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \cdot p_\phi(\mathbf{z}_t, \mathbf{z}_{1:t-1} \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_{1:t} \\
 &= \int_{\mathbf{z}_{1:t}} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \cdot p_\phi(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}) \cdot p_\phi(\mathbf{z}_{1:t-1} \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_{1:t} \\
 &= \int_{\mathbf{z}_{1:t}} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \cdot p_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \cdot p_\phi(\mathbf{z}_{1:t-1} \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_{1:t} \\
 &= \int_{\mathbf{z}_t} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \cdot p_\phi(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_t
 \end{aligned} \tag{2.12}$$

where Equations (2.8) to (2.6) are used to simplify terms, and on the last step the conditionals of $\mathbf{z}_{1:t-1}$ are marginalized out. Many modelling assumptions allow the straightforward calculation of moments of the probability $p_\theta(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})$, while in some case its even possible to get closed form solution [Durbin and Koopman, 2001](#); [Särkkä and Svensson, 2023](#). Learned models can also be used for generative purposes, drawing from $p_\phi(\mathbf{z}_1)$.

2.2.4 Learning SSMs

Learning in **SSMs** centers on approximating the joint distribution $p_\theta(\mathbf{x}_{1:t}, \mathbf{z}_{1:t})$ using available data ([Barber, 2011](#); [Girin et al., 2020](#)). Directly tackling this problem is often intractable and, generally, requires assumptions about the system dynamics, observation mechanisms, and noise characteristics. We explore two approximation techniques: Bayesian filtering and variational inference. While both are valuable, variational inference becomes particularly relevant when using **DNNs** to model **SSMs**.

Bayesian Filtering

In Bayesian filtering (Kalman, 1960; Särkkä & Svensson, 2023), the objective is to estimate the posterior distribution of the state given the observations, $p_\phi(\mathbf{z}_t \mid \mathbf{x}_{1:t})$, a process known as inference. A key feature of filtering methods is their ability to iteratively update their approximation, incorporating new observations as they become available. This adaptability makes them well-suited for online learning scenarios, where the model learns iteratively as new data arrives, and control problems Durbin and Koopman, 2001. Additionally, filtering methods are increasingly applied in offline learning settings, where the model is learned from a fixed, previously collected dataset (Doucet, Johansen, et al., 2009; Duan et al., 2023).

One of the simplest SSMs is the linear Gaussian model. Here, the state and observation functions are linear transformations (represented by matrices), and noise follows zero-mean Gaussian distributions with parameterized covariance matrices. In this specific scenario, the Kalman filter provides the optimal, closed-form solution for parameter estimation (Kalman, 1960; Roweis & Ghahramani, 1999). For non-linear non-Gaussian models, alternative approaches like Markov Chain Monte Carlo (MCMC) (Dellaortas & Roberts, 2003) and sequential Monte Carlo (SMC) methods (Doucet, Johansen, et al., 2009) have been used.

Various objectives can be used for learning the parameters. Maximum likelihood (MLE) and maximum a posteriori (MAP) provide point estimation of parameters. This can be done using iterative algorithms like expectation-maximization. A fully Bayesian treatment of parameter learning is also possible (Särkkä & Svensson, 2023); in this case learning is also an inference process.

Learning parameters in SSMs can be accomplished using various objectives. Maximum likelihood (MLE) and maximum a posteriori (MAP) estimation provide point estimates of the parameters, often found using iterative algorithms like expectation-maximization (EM). Fully Bayesian approach to parameter learning can also be applied (Särkkä & Svensson, 2023). In this Bayesian framework, the parameters are treated as random variables, making the learning process itself a form of inference.

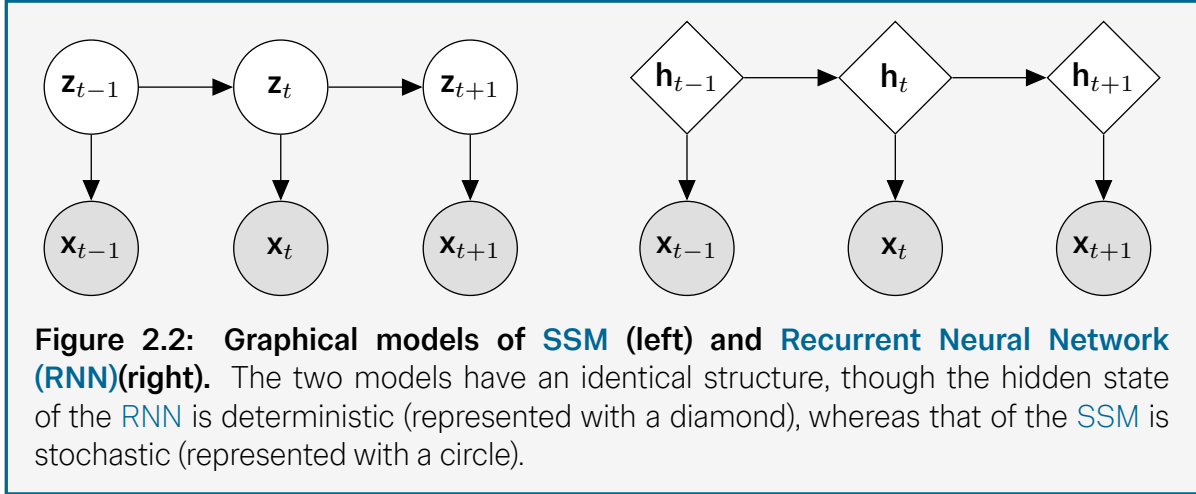
To efficiently learn the posterior distribution in SSMs, filtering techniques can leverage their autoregressive structure. This property is formally described in the following lemma:

Lemma.

$$p_\phi(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \propto p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \int p_\phi(\mathbf{z}_t \mid \mathbf{z}_{t-1}) p_\phi(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_{t-1} \quad (2.13)$$

This lemma (proof in Section 7.1 of the appendix) plays a crucial role in filtering. It allows us to iteratively refine the approximation of the state dynamics by incorporating new observations.

The update step relies on Equation (2.13) (the filtering distribution) to integrate this new observation. Predictions then leverage the updated parameters. Through this iterative process of updating and predicting, the model's parameters adapt to new information.



Variational Inference

Variational Inference (VI) is a broad inference technique applicable not just to dynamical systems, but to a wide range of Bayesian models where calculating the exact posterior distribution is intractable (Blei et al., 2017). Unlike filtering, VI remains efficient when the dynamics or observation model are non-linear and/or high-dimensional. This flexibility makes VI a especially useful for learning DNN-based SSMs. DNNs can capture complex nonlinear dependencies in high-dimensional spaces, offering greater flexibility and power than traditional linear models.

In Variational Inference, we begin by specifying a family of approximate (or variational) distributions, denoted by $q_\phi(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \in \mathcal{Q}$. The objective is to find the distribution within this family that most closely resembles the true posterior $p(\mathbf{z}_t \mid \mathbf{x}_{1:t})$. To ensure computational tractability, the family of approximate posteriors $\mathcal{Q}$ is typically chosen to be simple (e.g., Gaussian). While this simplifies learning and inference, it can potentially introduce inaccuracies if the true posterior cannot be well-represented by the chosen family (Tomczak & Welling, 2018).

To measure the similarity (or rather, dissimilarity) between distributions, we use the Kullback-Leibler divergence (KL). With VI, learning is reframed as an optimization task where we aim to minimize the KL divergence between our approximate posterior, $q_\phi(\mathbf{z})$, and the true posterior subject to the constraint that $q_\phi(\mathbf{z}) \in \mathcal{Q}$.

$$\arg \min_{\theta, \phi} \text{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid p_\theta(\mathbf{z} \mid \mathbf{x})) \quad (2.14)$$

The KL divergence can be decomposed as follows:

$$\begin{aligned}
 \text{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) &= \mathbb{E}_{q_\phi} [\log q_\phi(\mathbf{z} \mid \mathbf{x}) - \log p_\theta(\mathbf{z} \mid \mathbf{x})] \\
 &= \mathbb{E}_{q_\phi} [\log q_\phi(\mathbf{z} \mid \mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z}) + \log p_\theta(\mathbf{x})] \\
 &= \mathbb{E}_{q_\phi} [\log q_\phi(\mathbf{z} \mid \mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})] + \log p_\theta(\mathbf{x}) \quad (2.15)
 \end{aligned}$$

We define **Evidence Lower Bound (ELBO)** as the quantity:

$$\text{ELBO}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})] \quad (2.16)$$

and rearrange the terms to solve for $\log p_\theta(\mathbf{x})$:

$$\log p_\theta(\mathbf{x}) = \text{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) + \text{ELBO}(\theta, \phi; \mathbf{x}) \quad (2.17)$$

Since the KL divergence is always non-negative ($\text{KL} \geq 0$), this shows that the **ELBO** is indeed a lower bound on the log likelihood of the data under the model. Maximizing the **ELBO** with respect to θ and ϕ effectively minimizes the KL divergence, bringing our approximate posterior closer to the true posterior and improves the approximation. Furthermore, the objective function depends on the parameters of both the variational distribution and the generative model. When these models are parameterized by neural networks, this allows us to learn both simultaneously with methods like gradient descent.

2.3 Forecasting with Neural Networks

In the previous [Section 2.2.2](#) we discussed [SSMs](#), an important tool for modelling dynamical systems and time series in general. However, these model are hard to scale to higher dimension without assuming linear system dynamics. This section introduces [DNNs](#), a non-linear function approximate that can be used in conjunction with [SSMs](#) to overcome some of their limitations.

2.3.1 Deep Neural Networks

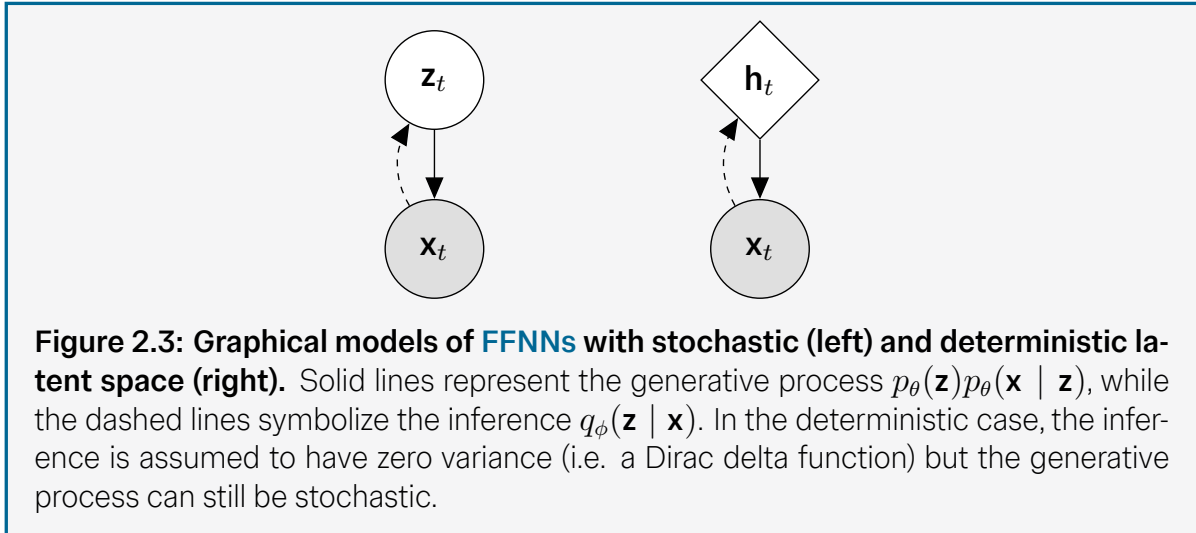
[Deep Neural Networks \(DNNs\)](#) are machine learning models that, inspired by the human brain, process information through layers of interconnected **neurons**. [DNNs](#) incorporate multiple layers, enabling them to learn complex patterns and hierarchical representations of data. This makes them very effective for tasks ranging from image and speech recognition to natural language processing, offering significant advancements in artificial intelligence applications ([Goodfellow et al., 2016](#)). [DNNs](#) can be categorized based on how they process information:

Feed-forward Neural Networks (FFNNs) feature a unidirectional flow of information, from the input layer through hidden layers to the output layer. This architecture makes them ideal for tasks where data points are assumed independent. A prime example is image classification, where the network analyzes each image in isolation. Forecasting, however, relies on dependencies within sequential data, and FFNNs are considered less suitable for this domain. Despite this limitation, FFNNs are known for their computational efficiency and widespread use in various applications.

Recurrent Neural Networks (RNNs) possess a hidden "memory" due to feedback loops in their architecture. This memory allows [RNNs](#) to process sequences of data by incorporating information from both the current input and the network's hidden state, derived from previous inputs. This design makes [RNNs](#) powerful for tasks like language translation or time series prediction, where the context and order of inputs significantly influence the output. However, [RNNs](#) often face challenges when handling very long sequences due to issues like vanishing and exploding gradients.

For forecasting, both types of networks can be optimized to minimize prediction errors, e.g. using the [Mean Square Error \(MSE\)](#) or [Mean Absolute Error \(MAE\)](#). This is a common approach in regression settings. From a probabilistic perspective, using the [MSE](#) implies modeling the errors with a Gaussian distribution while [MAE](#) is derived by the Laplacian distribu-

tion.



2.3.2 Deep Probabilistic Graphical Models

While DNNs might not seem inherently probabilistic, reinterpreting them through the lens of the Probabilistic Graphical Model (PGM) framework offers a valuable perspective. This creates a bridge to traditional PGM and SSM methods and opens the possibility of synergies, allowing us to leverage the strengths of both approaches.

Deep SSMs

For example, DNNs offer a promising solution to address the limitations of traditional learning methods for SSMs. By parameterizing the functions $\mathcal{F}$ and $\mathcal{G}$ in the state (2.6) and observation (2.7) equations, DNNs introduce several key advantages. Primarily, they allow us to directly learn complex system dynamics from observed data, eliminating the need for hand-crafted mathematical models. Furthermore, advancements in hardware and optimization algorithms enable DNNs to scale effectively to high-dimensional non-linear systems. Their flexible structure also makes it easier to incorporate domain knowledge (such as grid information in spatiotemporal settings) (Surace et al., 2019; Yi et al., 2024).

The integration of DNNs into SSMs leads to a class of models known as Deep SSMs. While some Deep SSMs share the graphical structure of Figure 2.2 (left) (Krishnan et al., 2015), more elaborate structures have also been proposed (Girin et al., 2020; Hafner et al., 2019).

RNNs

The PGM framework also helps us understand how RNNs can be viewed as probabilistic models (Choe et al., 2017). In this context, the graphical models of RNNs and SSMs have a one-to-one correspondence, where a deterministic hidden state $\mathbf{h}_t$ represents the latent state variable $\mathbf{z}_t$ (see Figure 2.2). The RNN's transition function (which updates the hidden state) mirrors the SSM's state equation, and the RNN's output mapping aligns with the SSM's observation equation. The key distinction is that basic RNNs lack the stochastic elements often found in traditional SSMs.

FFNNs

We can also consider an FFNNs as a conditional probability distribution where the input layer represents observed variables. We can view the collection of neurons across all hidden layers as representing a single, multi-dimensional latent variable. The network's weights and biases then define the parameters governing the conditional probability distribution between this latent variable and the input. Finally, the output layer represents the probability distribution over the target variables, conditioned on this latent variable. While in many cases FFNNs use deterministic neurons Bengio et al., 2013, versions with stochastic latent variables have also been emerging (Chen et al., 2018b; Ho et al., 2020; Kingma & Welling, 2014).

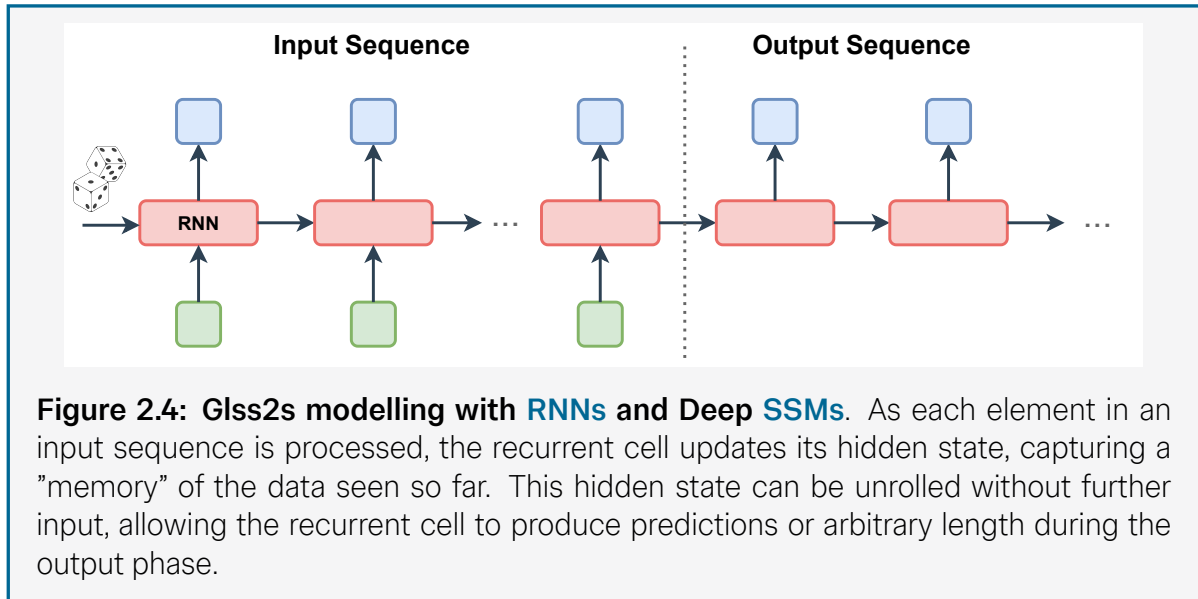
In forecasting, where we aim to predict future values of the observed variables, FFNNs can be viewed as graphical models. Since the target variables are the same as the observed variables, they collapse into a single node, as illustrated in Figure 2.3.

2.3.3 Long term forecasting with DNNs

The architectural difference between RNNs and Deep SSMs on one hand, and FFNNs on the other, also affects how long-term forecasting is achieved.

Sequence-to-sequence (seq2seq) is a framework that maps a sequence of inputs to a sequence of outputs. It excels in tasks where both the input and output are sequences that may vary in length, such as machine translation, language modelling, time series and dynamical systems forecasting (Ehrhardt et al., 2017; Sutskever et al., 2014). RNNs and Deep SSMs, due to their inherent hidden state mechanism, are well-suited for seq2seq modeling. As each element in an input sequence is processed, the recurrent cell updates its hidden state, capturing a "memory" of the data seen so far (illustrated in Figure 2.4). This evolving hidden state, can be unrolled without further input, allowing the recurrent cell to produce sequential predictions or arbitrary length during the output phase.

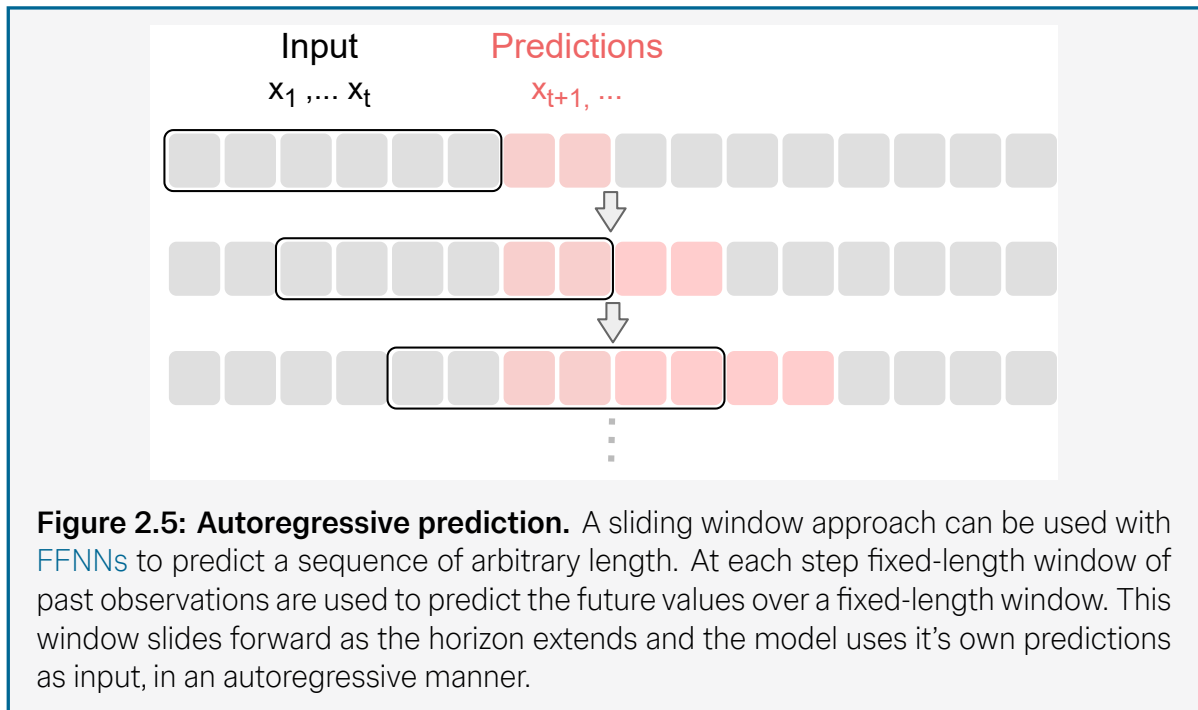
Autoregressive prediction. While FFNNs inherently lack the sequential memory mechanism of RNNs and Deep SSMs, they can still model time series data in a few ways. Notably,



a sliding window approach can be used. Here, the FFNN processes a fixed-length window of past observations to predict the future values over a fixed-length window:

$$\hat{\mathbf{x}}[t], \hat{\mathbf{x}}[t+1], \dots, \hat{\mathbf{x}}[t+k] = \text{NN}_{\theta}(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \dots, \mathbf{x}_{t-l+1}) \quad (2.18)$$

Where, l, k the input and output window length respectively, and θ the DNN parameters. The window slides forward as the horizon extends and the model uses its own predictions as input, in an autoregressive manner as Figure 2.5 shows. While computationally efficient this approach has limitations since the window size inherently limits the range of dependencies the model can capture.



2.4 Disentangled Representation Learning

In many machine learning applications, a critical goal is to learn meaningful representations of data that can facilitate downstream tasks such as prediction, classification, or generation. An important paradigm within representation learning is **disentangled representation learning**. The core objective of disentanglement is to learn representations where individual latent variables, or distinct dimensions of the latent space, correspond to independent, interpretable factors of variation in the data distribution. This approach is often motivated by the idea of aligning the learned representation with the true underlying causal factors of the data-generating process (Schölkopf, 2019). By achieving such a separation, models can capture more robust, meaningful, and interpretable features (Eastwood & Williams, 2018).

The pursuit of disentangled representations is driven by several potential benefits. Firstly, **interpretability** is enhanced. When latent variables correspond to distinct real-world generative factors (e.g., object shape, color, or in dynamical systems, specific physical parameters), the model's internal workings become more transparent and understandable. Secondly, models with disentangled representations are often hypothesized to achieve better **generalization** to unseen data, particularly new combinations of the learned factors of variation (Bengio et al., 2013; Higgins et al., 2017), which can lead to improved robustness in out-of-distribution (OOD) scenarios. Lastly, disentangled latents can facilitate tasks such as **controllability and transfer learning**, where specific attributes of the output can be manipulated by changing corresponding latent variables, and can also be beneficial for transfer learning (Locatello et al., 2019a).

Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) are a prominent class of generative models extensively studied for learning disentangled representations. VAEs consist of an encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$ that maps input data $\mathbf{x}$ to a latent distribution (typically Gaussian), and a decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$ that reconstructs the data from samples of this latent distribution. They are trained by maximizing the Evidence Lower Bound (ELBO) on the marginal log-likelihood of the data:

$$\mathcal{L}_{\phi, \theta}^{\text{VAE}}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})} [\log p_\theta(\mathbf{x} \mid \mathbf{z})] - \text{KL} (q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))$$

where the first term is the expected reconstruction log-likelihood, and the second term is the Kullback-Leibler (KL) divergence between the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ and a chosen prior $p(\mathbf{z})$ (often an isotropic Gaussian $\mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$). Several strategies have been developed to encourage disentanglement within the VAE framework and other models. **Unsupervised disentanglement** methods often involve modifications to the VAE objective,

such as in the β -VAE (Higgins et al., 2017), which weights the KL divergence term to place a stronger constraint on the latent space, encouraging it to factorize along the data's axes of variation. Other information-theoretic approaches also fall under this category (Kim & Mnih, 2018). However, purely unsupervised disentanglement can be challenging, often relying on specific model architectures, inductive biases, or assumptions about the data (Locatello et al., 2019a; Tschannen et al., 2018). In contrast, **supervised and weakly-supervised disentanglement** techniques leverage available labels or information about the ground-truth factors of variation. Even partial supervision can significantly improve disentanglement (Locatello et al., 2020b; Mathieu et al., 2016a; Shu et al., 2020) by guiding the model to explicitly encode these known factors in specific parts of the latent space, often through additional loss terms.

In the context of dynamical systems, disentangled representation learning offers a powerful means to separate static system parameters (e.g., physical constants like mass or length, or coefficients in governing equations) from the time-varying dynamic states of the system (e.g., positions, velocities, or population counts) (Iten et al., 2020). This separation can be particularly beneficial for improving long-term prediction accuracy, as the model can better distinguish between invariant system properties and evolving states. It can also enhance robustness to OOD scenarios, where the system parameters might differ from those seen during training, and increase the interpretability of the learned representations, allowing for a clearer understanding of how different parameters influence the system's behavior. By treating system parameters as distinct factors of variation, supervised disentanglement techniques can be employed to explicitly encode these parameters in the latent space, thereby creating more robust and domain-aware models for dynamical systems.

Chapter 3

Related Work

This chapter offers a survey of the related work essential for understanding the research presented in this thesis. We begin by exploring deep learning methodologies for spatiotemporal forecasting, with a focus on [Feed-forward Neural Networks \(FFNNs\)](#), [Recurrent Neural Networks \(RNNs\)](#) and [Deep State Space Models \(Deep SSMs\)](#). We then investigate the incorporation of physical laws into neural network architectures through physics-informed neural networks (PINNs) and related approaches. The chapter proceeds to discuss disentanglement in deep learning, particularly within [Variational Autoencoders \(VAEs\)](#) and their role in sequence modeling.

3.1 Deep Learning for Spatiotemporal Forecasting

Spatiotemporal forecasting entails predicting future states of a system based on data that vary over both space and time. [Deep Learning \(DL\)](#) methods have been extensively applied to spatiotemporal forecasting across various domains ([Oprea et al., 2022](#)), including the numerical solution of [Partial Differential Equations \(PDEs\)](#) ([Lino et al., 2023](#)). These methods exploit the capability of deep neural networks to model complex, nonlinear relationships within data, making them well-suited for predicting the evolution of dynamical systems.

3.1.1 Convolutional Neural Networks

[Convolutional Neural Networks \(CNNs\)](#) are feed-forward architectures that have achieved remarkable success in image-related tasks and have been extended to video prediction applications ([Oprea et al., 2022](#)). In [Goroshin et al., 2015](#), 2D convolutional filters are employed to learn latent representations of two consecutive frames, with the temporal difference extrapolated to generate future predictions. Similarly, [Mathieu et al., 2016b](#) explored multi-scale 2D [CNNs](#) and introduced an adversarial loss to mitigate the blurriness commonly associated with [Mean Square Error \(MSE\)](#) loss in predictions.

Utilizing 3D convolutions enables the direct extraction of spatiotemporal correlations, eliminating the need for additional manipulations ([Vondrick et al., 2016](#)). While recurrent models have traditionally been considered superior for sequence modeling, recent studies have demonstrated that dilated convolutions can outperform recurrent networks in various one-dimensional tasks ([Bai et al., 2018](#)). Motivated by these findings, we evaluate a [CNN](#) based on 2D dilated convolutions for wave propagation in [Chapter 4](#).

3.1.2 Recurrent Neural Networks

While [CNNs](#) excel at capturing spatial features, spatiotemporal prediction inherently involves temporal dynamics. [RNNs](#), such as [Long Short-Term Memorys \(LSTMs\)](#), possess internal states that provide memory capabilities, making them well-suited for sequence modeling ([Hochreiter & Schmidhuber, 1997](#); [Shi & Yeung, 2018](#)). However, standard [RNNs](#) are primarily designed for one-dimensional sequences and scaling them to high-dimensional spatiotemporal is computationally inefficient.

One approach to apply [RNNs](#) to high-dimensional data involves first reducing the data to a lower-dimensional vector representation via some transformation and then processing the

reduced data recurrently (Ehrhardt et al., 2017; Srivastava et al., 2015). This is achieved by adopting an encoder-forecaster architecture, where the encoder learns a latent representation of the input, and the forecaster propagates this representation into the future. In Srivastava et al., 2015, the input frames are concatenated into a single vector and fed into a stack of LSTMs to predict the next frame. However, concatenating frames increases the input size exponentially, leading to a significant growth in network parameters. To alleviate this issue, several studies have employed convolutional encoders to reduce the dimensionality before feeding data into recurrent networks (Cricri et al., 2016; Oh et al., 2015).

Combining the spatial feature extraction capabilities of CNNs with the temporal modeling strengths of LSTMs has proven effective, but it presents some limitations. Firstly, processing spatial and temporal signals independently may overlook potential interactions between them. Secondly, even with dimensionality reduction, the memory and computational requirements of these models remain substantial. To address these challenges, the **Convolutional LSTM** (ConvLSTM) replaces the fully connected gates within the LSTM with convolutional operations (Shi et al., 2015). Originally applied to predict short-term meteorological phenomena, ConvLSTMs have been widely adopted in state-of-the-art video prediction techniques (Finn et al., 2016; Villegas et al., 2017). PredRNN introduces the **Causal LSTM**, a cell based on ConvLSTM that adds a global memory variable transferring information not only between time steps but also across layers (Wang et al., 2017). PredRNN++ further enhances this architecture by incorporating a gradient highway unit, facilitating more efficient backpropagation through time (BPTT) (Wang et al., 2018).

Although both RNNs and CNNs have been successfully applied to the prediction of PDEs, there is a scarcity of studies directly comparing these two categories from a representation learning perspective. Our comparative study in Chapter 4 aims to bridge this gap.

3.1.3 Deep State Space Models

State space models (SSMs) are foundational frameworks for modeling dynamical systems and time series data, incorporating latent variables to capture the underlying state of the system (Durbin & Koopman, 2001; Kalman, 1960). Traditional SSMs, such as the Kalman filter (Kalman, 1960), rely on linear dynamics and Gaussian noise assumptions, limiting their applicability to complex, nonlinear systems.

To overcome these limitations, deep learning techniques have been integrated into SSMs, resulting in **deep state space models** (DSSMs). DSSMs leverage the representational power of deep neural networks to model complex, nonlinear system dynamics and observation processes (Girin et al., 2020; Krishnan et al., 2015).

Deep Learning Approaches to State Space Models

[Krishnan et al., 2015](#) introduced the **Deep Kalman Filter** (DeepKF), where both the state transition and emission functions are parameterized by neural networks. This approach extends traditional linear Gaussian SSMs to nonlinear, non-Gaussian settings, enabling the modeling of complex time series data through variational inference to approximate the intractable posterior distributions. Similarly, the **Variational Recurrent Neural Network** (VRNN) proposed by [\(Chung et al., 2014\)](#) combines recurrent neural networks with variational inference to model sequential data with latent variables. The VRNN extends the variational autoencoder (VAE) framework [\(Kingma & Welling, 2014\)](#) to sequences, capturing both temporal dependencies and latent stochasticity. [Fraccaro et al., 2017](#) developed the **Deep Variational Bayes Filter** (DVBF), introducing an inference model that combines an [RNN](#) with variational inference for latent variables. The DVBF is designed to learn latent dynamics in an unsupervised manner, capturing complex temporal dependencies in time series data.

DSSMs have been applied across various domains requiring the modeling of complex dynamical systems. For example, [\(Hafner et al., 2019\)](#) introduced the **Recurrent State-Space Model** (RSSM), which learns latent dynamics from high-dimensional pixel observations, enabling model-based planning and control in reinforcement learning tasks. [Girin et al., 2020](#) provided a comprehensive review of **Dynamical Variational Autoencoders** (DVAEs), highlighting their applications in modeling videos, speech signals, and other time series data.

Relation to Other Models

DSSMs bridge the gap between traditional probabilistic graphical models and deep learning approaches. While traditional SSMs provide a probabilistic framework for modeling time series data, they are limited in capturing complex, nonlinear dynamics. Deep learning models, such as [RNNs](#), excel at modeling sequences but often lack a probabilistic interpretation and the ability to model uncertainty.

By integrating latent variables and probabilistic inference with deep neural networks, DSSMs offer a powerful framework for modeling complex dynamical systems, capturing both nonlinear dynamics and uncertainty. Despite their advantages, DSSMs present challenges in scalability, training stability, and interpretability [\(Girin et al., 2020\)](#). Exploiting the Markovian properties of physical systems under study, in [Chapter 4](#) we propose the re-purposing of a feed-forward neural network for spatiotemporal forecasting.

3.1.4 Other Architectures

Beyond [CNNs](#) and [RNNs](#), other architectures have been proposed for spatiotemporal prediction, particularly those that handle inherent uncertainty by employing generative models. [Generative Adversarial Networks \(GANs\)](#) can implicitly model the probability distribution of predictions ([Goodfellow et al., 2014](#)). Training [Deep Neural Networks \(DNNs\)](#) on video using a simple [MSE](#) loss is known to produce blurry results ([Mathieu et al., 2016b](#)), prompting methods that combine an L2 loss with an adversarial loss to produce sharper predictions ([Vondrick et al., 2016](#)).

3.2 Physics-Aware Deep Learning

Integrating physical knowledge into deep learning models enhances their ability to predict and generalize in physical systems. Physics-aware approaches, whether through incorporating partial differential equations directly into loss functions (Raissi et al., 2019) or designing architectures that respect conservation laws (Greydanus et al., 2019), provide powerful tools for modeling complex dynamical systems. These methods bridge the gap between data-driven models and traditional physics-based simulations, offering improved accuracy, interpretability, and computational efficiency.

3.2.1 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) explicitly incorporate physical laws into neural network architectures, leveraging prior knowledge of governing equations to enhance learning efficiency and accuracy (Raissi et al., 2019; Yang & Perdikaris, 2018). PINNs have been particularly effective in solving forward and inverse problems involving PDEs, offering a mesh-free and flexible framework for modeling complex physical phenomena.

RNNs have been combined with physics-informed methodologies to address spatiotemporal PDEs. Ren et al., 2022 introduced, a model that integrates physical domain knowledge with convolutional and recurrent layers to solve spatiotemporal PDEs efficiently. (Geneva & Zabaras, 2019) developed physics-constrained deep autoregressive networks to model the dynamics of PDE systems. By incorporating physical constraints directly into the network architecture, they enhanced prediction accuracy and ensured consistency with known physical laws.

3.2.2 Neural Operators

Deep Operator Networks (DeepONets) are designed to learn operators mapping between infinite-dimensional function spaces, providing a framework for modeling complex system behaviors (Lu et al., 2021). They excel in capturing underlying patterns of physical systems by learning relationships between input functions and their corresponding outputs. Similarly, **Fourier Neural Operators** utilize spectral methods to capture global information effectively (Li et al., 2021). By operating in the frequency domain, these models efficiently learn mappings between functions, which is advantageous for problems with periodic or oscillatory solutions.

The application of **Koopman operator theory** in neural networks offers a linear representa-

tion of nonlinear dynamical systems in higher-dimensional spaces (Lusch et al., 2018). The key idea is to learn a mapping from the original system state to a higher-dimensional latent space where the time evolution becomes approximately linear. A neural network encoder maps inputs to this latent space, a linear operator models the dynamics over time, and a decoder reconstructs the system's future states. This allows complex nonlinear behaviors to be predicted using simple linear evolution, improving interpretability and often making long-term forecasting more stable.

3.2.3 Physics-Inspired Models

Physics-inspired deep learning makes less assumptions about the underlying system than PINNs and Neural Operators. Still, domain knowledge is embedded in these methods e.g. in the loss function or in the architectural design. Physics-inspired methods have been applied to various domains, including uncertainty quantification and reliability estimation in nonlinear dynamical systems such as weather modeling (Yang et al., 2024). By integrating physical models with deep learning, these approaches quantify uncertainties arising from model approximations and data noise. Thoreau et al., 2022 proposed a physics-integrated generative model for semantic segmentation of optical remote sensing images, combining variational autoencoders with physical constraints to improve segmentation accuracy.

Natural video prediction techniques often assume minimal prior knowledge; however, many real-world spatiotemporal data arise from well-defined physical processes. Leveraging prior physical knowledge can lead to better statistical models. (Cressie & Wikle, 2015) demonstrated that building autoregressive models inspired by PDEs outperforms traditional models in spatiotemporal prediction tasks. Similarly, (Raissi et al., 2017a) employed Gaussian Processes with covariance functions derived from temporal discretizations of PDEs to quantify uncertainty in noisy data.

Forecasting architectures can also be derived from fundamental physical laws, such as conservation of energy or motion laws, by applying appropriate constraints (Stewart & Ermon, 2017). For instance, Greydanus et al., 2019 proposed Hamiltonian Neural Networks, which incorporate Hamiltonian mechanics to predict conservative physical systems, ensuring adherence to energy conservation laws. However, the effectiveness of such techniques in large-scale, non-conservative models remains an open question.

3.2.4 Identification of Physical Systems

Learning the parameters of an underlying physical process is akin to system identification. Wu et al., 2016 used neural networks to infer mass, density, and friction from unlabeled videos and subsequently employed a physics simulator to predict future states. The drawback is the necessity of a physics engine for simulations. Other researchers have trained neural

networks to answer questions such as determining the number of degrees of freedom in a system ([Iten et al., 2020](#)). Identifying the PDEs governing a system from data is also possible using Gaussian Processes ([Raissi & Karniadakis, 2018](#)) or neural networks ([Long et al., 2019](#)).

Intuitive Physics

Intuitive physics, a field in cognitive psychology, studies how humans perceive the physical world and make predictions about physical phenomena. Deep learning models have been developed to predict the effects of forces on parts of an image ([Mottaghi et al., 2016](#)) or assess the stability of structures like towers of blocks ([Lerer et al., 2016](#)). While these methods capture certain aspects of physical reasoning, they often do not model the temporal evolution of the environment explicitly.

3.3 Fluid Flow Approximation

3.3.1 Eulerian and Lagrangian Approaches

Simulation methods for fluid flow are generally divided into two broad categories: Eulerian methods, which predict the velocity and pressure fields over a discrete mesh, and Lagrangian methods, which consider the properties and interactions between individual particles ([Zhang & Chen, 2007](#)). Traditional Lagrangian methods like Smoothed Particle Hydrodynamics (SPH) simulate a large number of particles to achieve accurate predictions. However, handling such a vast number of objects poses challenges for current deep learning architectures. To address this, [Jeong et al., 2018](#) employed regression forests and hand-crafted features to predict the velocity of each particle in the next time step, thereby accelerating the simulation.

Eulerian methods involve a projection step where the pressure field is calculated by solving the Poisson equation, which is computationally expensive. Machine learning techniques can replace numerical computations to expedite the simulation process. [Tompson et al., 2017](#) designed CNNs motivated by the structure of linear systems to speed up the pressure projection step. U-Net-based architectures have been employed in the prediction of physical systems, such as sea temperature predictions ([de Bézenac et al., 2018](#)) and accelerating the simulation of the Navier-Stokes equations around an airfoil. ([Lino et al., 2020](#); [Thuerey et al., 2018](#)). Other architectures for spatiotemporal prediction include GANs for fluid simulations ([Kim et al., 2018](#)) and Graph Networks for wind-farm power estimation ([Lino et al., 2022b](#); [Park & Park, 2019](#)). A comprehensive review of the literature can be found in [Lino et al., 2023](#).

3.3.2 Wave Propagation

Wave propagation is a fundamental phenomenon in physics, described by PDEs such as the Saint-Venant equations. Accurate prediction of wave propagation is essential in numerous scientific and engineering applications. Recurrent neural networks have proven effective for this task due to their inherent ability to capture temporal correlations. ([Srivastava et al., 2015](#)) used a convolutional encoder-decoder architecture where an LSTM module is used to propagate the latent space into the future. Variations of this technique have been successfully applied to the long-term prediction of physical systems, such as sliding objects ([Ehrhardt et al., 2017](#)). Similar techniques have been employed for modeling wave propagation in shallow water ([Sorteberg et al., 2018](#)).

3.4 Disentanglement in Deep Learning

Disentanglement in deep learning refers to learning representations where each latent variable corresponds to a distinct, interpretable factor that influences the observed data. For instance, in a physical system, these factors could include parameters like mass, damping coefficient, or interaction strength, each affecting the system's behavior in a specific way. The goal is to recover these underlying variables in a way that reflects their independent roles in generating the data. This approach aligns with the notion of identifying the true causal model of the data-generating process (Schölkopf, 2019), where each latent variable maps to a separate causal mechanism.

While disentangled representations can be learned in supervised settings (Mathieu et al., 2016a) much of the recent focus has been on unsupervised approaches using generative models, particularly VAEs. These methods typically modify the VAE objective to encourage statistical independence across latent dimensions, often through information-theoretic regularization (Higgins et al., 2017; Kim & Mnih, 2018). However, recent results show that unsupervised disentanglement is not identifiable without additional inductive biases from the model architecture or data (Locatello et al., 2019a). This result has shifted the field toward semi- and weakly-supervised approaches (Locatello et al., 2020b). In summary, disentanglement aims to align latent variables with meaningful causal properties of the data distribution. The next sections review VAE-based approaches to disentanglement and how they extend to sequential and dynamical settings.

3.4.1 VAEs and Disentanglement

In physical sciences, hierarchical priors have been proposed to learn disentangled representations of high-dimensional spatial fields (Jacobsen & Duraisamy, 2022). To assess the strength of disentanglement, simulated datasets are often used because simulations provide access to ground-truth factors of variation (e.g., color or shape of an object in image data). Various metrics have been proposed to quantify disentanglement, including predictor-based (Eastwood & Williams, 2018) and information-theoretic measures (Chen et al., 2018a), although the task remains challenging (Carbonneau et al., 2020).

3.4.2 VAEs for Sequence Modeling

Dynamical VAEs (Girin et al., 2020) have long been used to model sequence dynamics. Combining VAEs with state-space modeling, they can model stochastic differential equations (Zhong & Meidani, 2022). VAEs have garnered interest in modeling physical systems due to

two main motivations. First, they offer various ways to incorporate inductive biases obtained from prior knowledge of the physical system. Second, since their latent space is of lower dimension, one can more easily assess whether these inductive biases result in more interpretable representations. Methods to incorporate inductive biases include (i) bottlenecks based on the degrees of freedom of the physical system (Iten et al., 2020), (ii) the use of geometric and topological information of the dynamical system responses to shape the manifold of the latent representations (Lopez & Atzberger, 2022), and (iii) using physics-informed priors (Takeishi & Kalousis, 2021). Furthermore, VAEs can be combined with recurrent neural networks to improve accuracy while learning highly disentangled representations of dynamical systems (Yeo et al., 2021).

3.4.3 Disentanglement in Sequence Modeling

Applying disentanglement methods to sequence modeling has recently gained interest as well. Using a bottleneck corresponding to the degrees of freedom of the physical system, (Iten et al., 2020) learned interpretable representations using a VAE. However, their model produced physically inconsistent predictions on out-of-distribution (OOD) data (Barber et al., 2021). Disentangling content from dynamics has been explored in deep state-space models (Fraccaro et al., 2017; Li & Mandt, 2018), but these methods focus mainly on modeling variations in the appearance of moving objects, neglecting the dynamics.

Unsupervised techniques have also been proposed. Assuming domain stationarity, (Miladinović et al., 2019) separated dynamics from sequence-wide properties in dynamical systems like Lotka-Volterra, but they did not fully evaluate the OOD performance of their model. (Yeo et al., 2021) suggested that learning a hierarchy of semantic concepts leads to feature abstraction and enhanced disentanglement, while (Li et al., 2022) proposed a model for time-series generation whose representation is disentangled by minimizing the pairwise total correlation between the latent variables.

While unsupervised methods have advantages, they may overlook valuable information that can be readily collected from simulated data, a gap that our method aims to fill. Furthermore, addressing OOD generalization for sequence modeling of physical systems can be benefited from models that can disentangle causal factors of variation and capture the underlying physical processes governing the data. Incorporating domain knowledge through physics-informed models and supervised disentanglement, as discussed in Section 3.4, enhances the robustness and interpretability of deep learning models in OOD settings.

3.5 Concluding Remarks

In this chapter, we reviewed key concepts and recent advancements in deep learning methods for spatiotemporal forecasting, physics-informed neural networks and disentanglement methods in deep learning. We highlighted the importance of integrating physical knowledge into neural network architectures to improve prediction accuracy and generalization, particularly in the context of dynamical systems governed by partial differential equations. The discussions in this chapter provide the necessary background for the methodologies and experiments presented in the subsequent chapters of this thesis.

Despite these advances, several research gaps remain. Many deep learning models struggle to generalize under distribution shifts, lack interpretability, or require explicit physical equations that are unavailable in many real-world systems. While disentangled representations offer a pathway toward interpretability and robustness, their use in modeling dynamical systems, especially in forecasting tasks involving spatiotemporal data, remains underexplored.

This thesis addresses these gaps by introducing supervised disentanglement as a domain-aware modeling strategy for dynamical systems. In [Chapter 4](#), we investigate the benefits and limitations of deep learning models (specifically U-Nets) for forecasting wave propagation, and identify challenges related to [out-of-distribution \(OOD\)](#) generalization. In [Chapter 5](#), we incorporate disentangled latent spaces into state-space models, showing how supervision on physical factors can enhance long-term prediction accuracy, improve stability, and yield more interpretable representations. Finally, in [Chapter 6](#), we extend this approach to high-dimensional dynamical systems using variational and recurrent models, demonstrating its applicability to spatiotemporal data observed through complex observation functions.

Chapter 4

Forecasting Wave Propagation with Neural Networks

Dynamical systems, often modeled by partial differential equations (PDEs), are essential in numerous scientific and engineering fields. In this chapter we explore the ability of [Recurrent Neural Networks \(RNNs\)](#) and [Feed-forward Neural Networks \(FFNNs\)](#) to predict, or forecast, the propagation of surface waves governed by the Saint-Venant equations. Our findings demonstrate improved long-term prediction accuracy compared to existing methods, while achieving inference times significantly faster than traditional numerical simulations. Notably, [FFNNs](#) match the performance of [RNN](#), despite their reduced complexity and computational cost. This result, observed with equivalent hyperparameter tuning, suggests a potential re-evaluation of architectural assumptions for dynamical system prediction. We further evaluate the generalization capabilities of these networks under extended time horizons and varying physical conditions.

The methods and part of the results discussed in the present chapter have been published in [Fotiadis et al., 2020](#). For researchers and practitioners interested in the underlying methodologies and experimental results, the code and datasets have been made publicly available at https://github.com/stathius/wave_propagation.

4.1 Introduction

Accurate forecasting of wave propagation is a core challenge in many scientific and engineering domains, with applications ranging from climate modeling and geophysics to biomedical engineering. While deep learning methods have shown promising results in sequence prediction tasks, their application to modeling physical wave phenomena remains relatively underexplored (Carleo et al., 2019).

In this chapter, we adapt and benchmark a feed-forward neural network architecture, the U-Net, for forecasting surface wave propagation (Ronneberger et al., 2015). Originally developed for image segmentation tasks, the U-Net is repurposed here for long-term forecasting of physical dynamics which is a domain traditionally dominated by RNNs. This adaptation is novel in the context of dynamical systems, particularly when evaluated against conventional recurrent models such as Convolutional LSTM (ConvLSTM) and Improved Predictive Recurrent Neural Network (PredRNN++). Our study offers new insights into the inductive biases of these architectures, showing that feed-forward models can match or even surpass recurrent alternatives in both forecasting accuracy and generalization performance.

Furthermore, we introduce a comprehensive evaluation suite that systematically varies physical properties (e.g., wave speed, tank size, excitation shape) and rendering conditions to study the generalization behavior of these models under controlled distribution shifts. Our findings show that FFNNs, despite lacking explicit memory, can achieve competitive or even superior performance to complex recurrent models—especially in inference efficiency and select generalization scenarios. These results challenge conventional assumptions about the necessity of recurrence in forecasting dynamical systems and motivate re-evaluating architectural choices for data-driven modeling. We focus on surface wave propagation as a representative case of Partial Differential Equation (PDE)-driven dynamics. This system offers several advantages for benchmarking forecasting models:

- **Diverse Applications:** From computer animation (Watt, 1990), communication technologies (Samad et al., 2023), and thermonuclear fusion (Heuraux et al., 2015), to earth science in areas like seismology (Serguei, 2023), hydrology (Moussa & Bocquillon, 2000), and avalanche, flood, and tsunami modeling (Ersoy et al., 2017; Mangeney-Castelnau et al., 2003; PONS et al., 2016), the applications are widespread.
- **Theoretical Foundation:** The Lyapunov stability of Saint-Venant Equations (S-V equations) numerical solutions is well-established (Aloev et al., 2023; Coron et al., 2007). This provides a controlled setting for training and evaluating data-driven models.

- **Broader Implications:** Since the [S-V equations](#) are derived from the Navier-Stokes equations, our findings offer insights relevant to the broader field of computational fluid dynamics. [Figure 4.1](#) provides a visual example of the spatiotemporal wave sequences used in our study.

The task at hand is a spatiotemporal forecasting as defined in [Section 2.1](#) and our primary goal is to develop a model that outperforms previous data-driven efforts in predicting wave propagation. The desiderata for the model are that it needs to accurately predict long sequences of future frames using only a short sequence of recent visual inputs, it must operate without prior knowledge of the physical system's specifications and demonstrate strong generalization to unseen data distributions. While we consider purely data-driven methods here, a key research question surrounds identifying an inductive bias that leverages the inherent determinism of the underlying system, which will be discussed further in [Chapter 5](#).

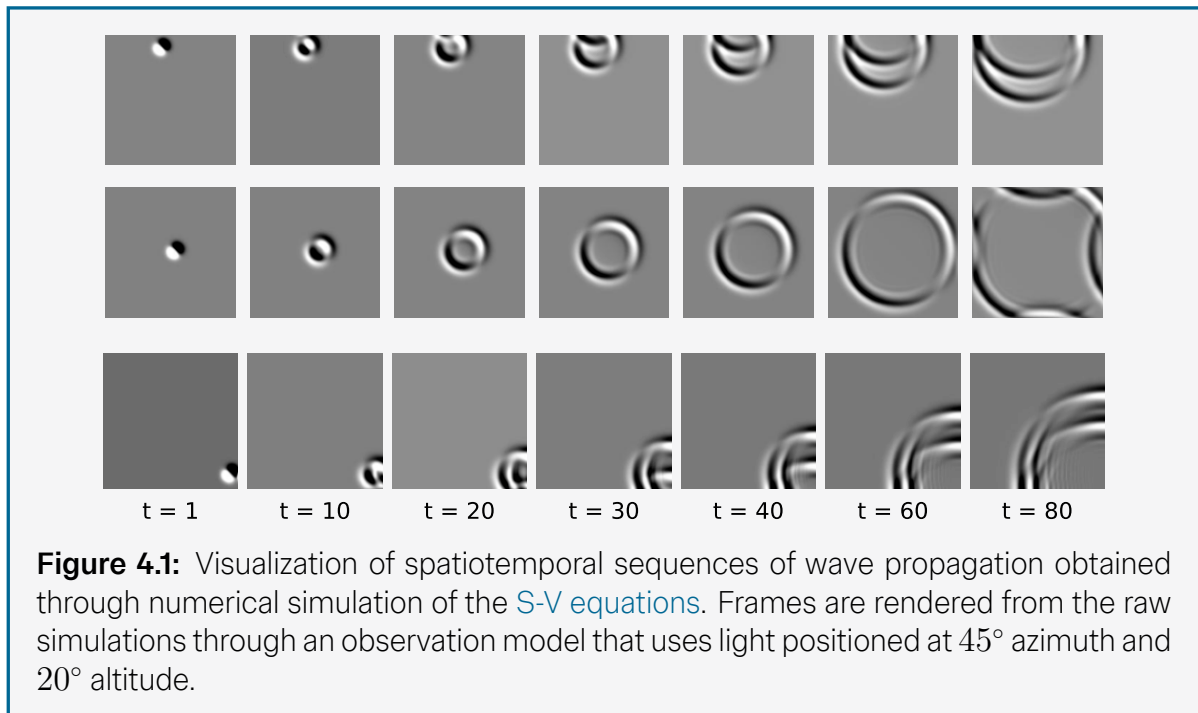
This study offers three main contributions:

Comparison of [RNNs](#) vs [FFNNs](#): We hypothesize that because the dynamics are Markovian in time, the long-term memory support of [RNNs](#) is not necessary in systems like wave propagation. By comparing the two classes of models we demonstrate that purely [Convolutional Neural Networks \(CNNs\)](#) can match the performance of [RNNs](#) in spatiotemporal prediction of [S-V equations](#).

Competitive Forecasting Performance: We identify three neural network architectures well-suited to [spatiotemporal forecasting \(STF\)](#) that outperform previous methods in long-term accuracy ([Sorteberg et al., 2018](#)) while maintaining significantly faster inference than numerical solvers.

Generalization Analysis: We critically assess the ability of our models to generalize to an extensive range of scenarios unseen during training and identify potential limitations.

This chapter is organised as follows. [Section 4.4](#) describes how the datasets for training and testing were created, [Section 4.3](#) describes the architecture of the evaluated models and [Section 4.5](#) outlines the details of the training procedure and the evaluation protocol. Next, results are presented in [Section 4.6](#). Specifically, [Section 4.6.1](#) analyses the accuracy of the models in long-term prediction, [Section 4.6.2](#) explores the [out-of-distribution \(OOD\)](#) generalization in new physical settings and [Section 4.6.4](#) analyzes the information contained in the learned representations. Finally, concluding remarks are given in [Section 4.7](#).



4.2 Motivation for FFNNs

In this section we formulate the problem of wave propagation and contrary to widespread understanding, we motivate the use of [FFNNs](#) for forecasting dynamical systems.

4.2.1 Problem formulation

Our specific aim is to forecast the evolution of wave patterns from video-like sequences. This is a [STF](#) setting, where each frame represents a two-dimensional snapshot ($M \times N$ grid) of that process (Figure 4.1). We denote these frames as tensors $\mathbf{x}_t \in \mathbb{R}^{M \times N}$. The problem formulation is the same as in [Section 2.1](#) with the only difference lying in the dimensions and structure of the variable $\mathbf{x}_t$. Briefly, given a sequence of L past observations $(\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_{t-L+1})$, the goal is to accurately predict the next k future frames. As we discussed in [2.2](#), we use the Markovian assumption to simplify the underlying distribution, which allows the use of deep [FFNN](#) and [RNN](#) models.

4.2.2 Markovian property of dynamical systems

The primary reason for utilizing [FFNNs](#) to model system dynamics lies in the Markovian nature of deterministic dynamical systems: when the full state is observable, future states depend only on the current state and the governing equations and not on the entire history. We can illustrate this with a simple [Ordinary Differential Equation \(ODE\)](#) which models the system's evolution:

$$\frac{d\mathbf{x}(t)}{dt} = \mathcal{F}(\mathbf{x}, t) \quad (4.1)$$

where $\mathcal{F}(\mathbf{x}, t)$ describes the system evolution over time, $\mathbf{x}$ is the state variable and t is time.

The Markovian property implies that if two trajectories $\mathbf{x}_0(t)$ and $\mathbf{x}_s(t)$ of a deterministic ODE intersect at any time t , they will evolve identically thereafter: $\mathbf{x}_0(t + s) = \mathbf{x}_s(t)$ for all $s \geq 0$. This reflects a fundamental causal property of dynamical systems: the future is entirely determined by the present state. Importantly, it also implies that system trajectories in state space cannot cross unless they coincide exactly, a condition that preserves the uniqueness of solutions.

In practice, however, many real-world systems are only partially observed or subject to stochastic influences. In such cases, the system's true state may be only partially accessible or inherently noisy, breaking the strict determinism assumed by the Markovian

Table 4.1: Comparison of Feed-Forward and Recurrent Neural Networks

Feature	Feed-Forward (FFN)	Recurrent (RNN)
Structure	Acyclic Graph	Graph with Cycles
Memory	None	Internal State
Ideal Tasks	Independent Inputs	Sequences, Time Series
Limitations	Temporal dependencies	Long-term sequences, training

property. Nonetheless, even in these settings, [FFNNs](#) may succeed by learning effective embeddings of recent observations that approximate sufficient statistics of the latent state. Thus, while [FFNNs](#) lack an explicit memory mechanism, their structure may still capture the causal sufficiency needed for accurate short-term forecasting. This study investigates the extent to which this holds true for spatiotemporal dynamics.

4.2.3 Effectiveness of [FFNNs](#) in sequence modelling

Traditionally, [RNNs](#) have been favored for modeling dynamical systems due to their perceived ability to handle longer-term dependencies ([Babaeizadeh et al., 2018](#); [Ehrhardt et al., 2017](#); [Finn et al., 2016](#); [Fragkiadaki et al., 2016](#); [Shi et al., 2015](#); [Wang et al., 2017](#)). In recent years, though, this perceived advantage has been challenged with research revealing that the theoretical **infinite memory support** of [RNNs](#) does not really hold in practice and that feed-forward approaches can be competitive in various 1-dimensional sequence modelling tasks like language ([Bai et al., 2018](#)). In addition, the Markovian property of dynamical systems indicates that the need for memory might not be so important. This opens the door to exploring [FFNNs](#) for modelling dynamical systems and study is a step towards that direction by assessing whether they can be competitive in [STF](#). Differences between [FFNNs](#) and [RNNs](#) are summarized in [Table 4.1](#).

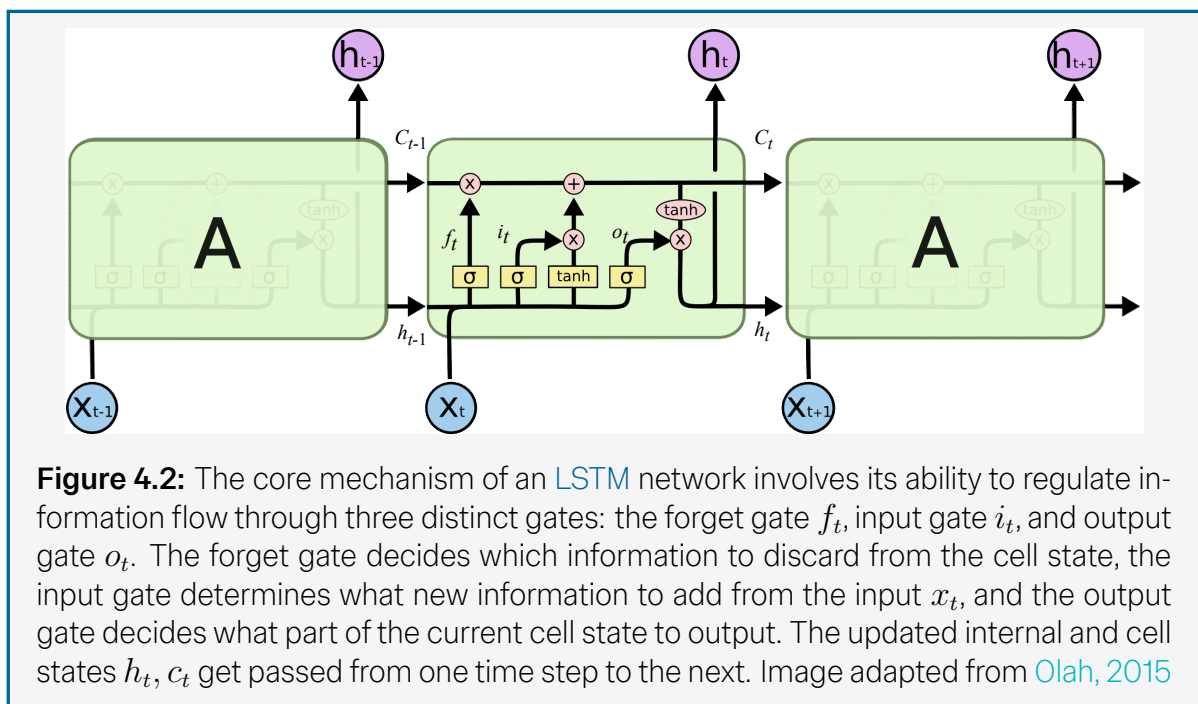
4.3 Neural Architectures

This section introduces the four deep neural network models that are evaluated in this study. Three of them are **RNNs** (ED-LSTM, **ConvLSTM**, **PredRNN++**) and one **FFNN** (U-Net).

Our baseline is a model specifically designed for predicting wave propagation (Sorteberg et al., 2018). It's based on a **Long Short-Term Memory (LSTM)** model (Hochreiter & Schmidhuber, 1997). It features a convolutional encoder-decoder architecture with three **LSTMs** modules, each handling a different portion of the sequence.

Two more **RNN** based models were chosen due to their success in short-term **STF** tasks: **ConvLSTM** (Shi et al., 2015) and **PredRNN++** (Wang et al., 2018). Both models integrate convolutions within their recurrent cells, enabling a synergy between spatial and temporal modeling. **PredRNN++** additionally incorporates a spatial memory mechanism for enhanced short-term precision.

Lastly, the **FFNN** model is built upon the U-Net architecture (Ronneberger et al., 2015) that has been employed in spatiotemporal modelling tasks such as the estimation of optical flow (Dosovitskiy et al., 2015) and motion field (de Bézenac et al., 2018), and velocity field prediction (Thuerey et al., 2018). In contrast to previous methods, we prioritize long-term over short-term performance. For this, we implement an end-to-end training approach, conditioning the network on its own predictions.



4.3.1 Baseline: ED-LSTM

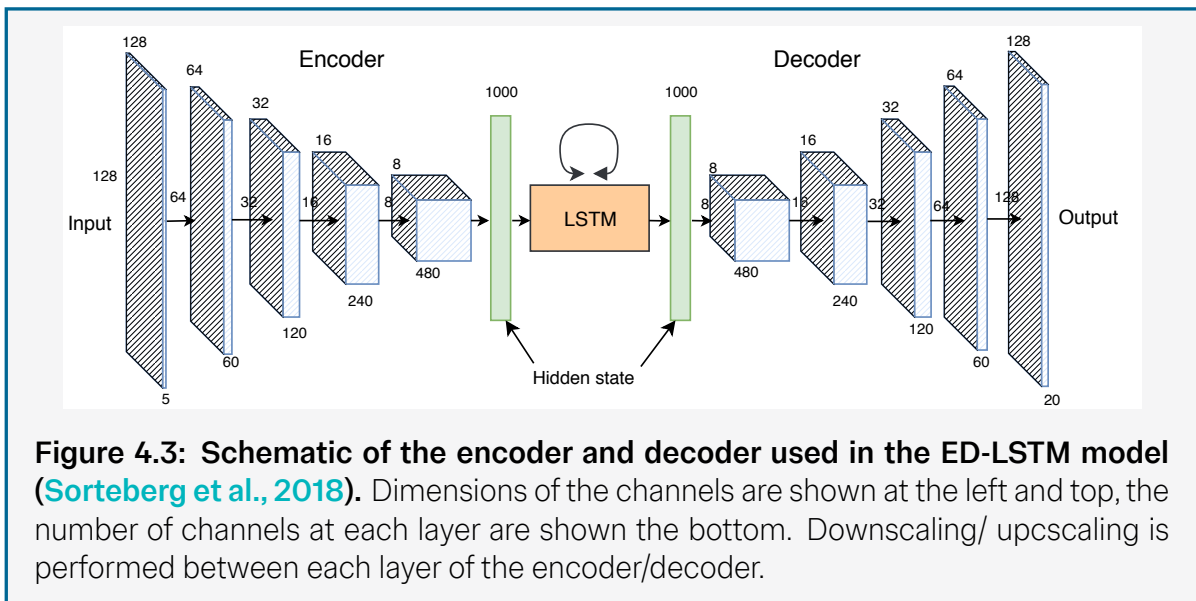
The baseline model is built upon a convolutional encoder-decoder (ED) and [LSTMs](#).

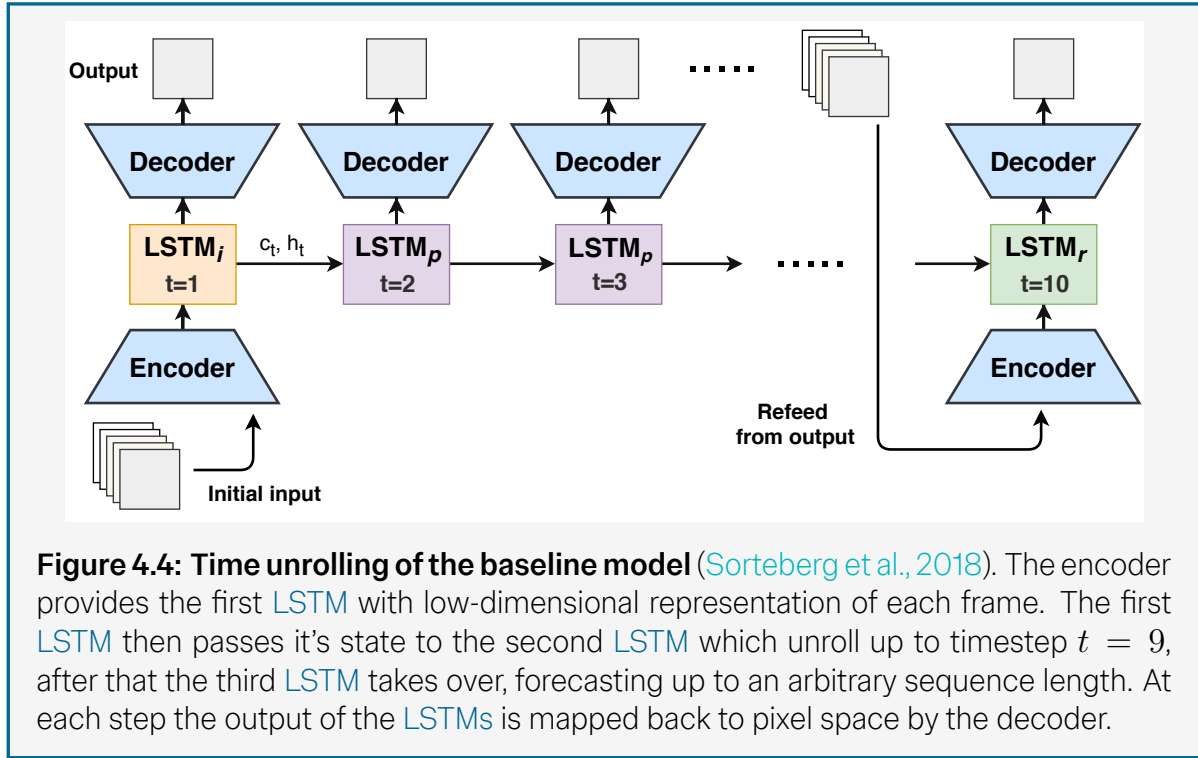
Long Short-Term Memory (LSTM) Although [RNNs](#) are well-suited for modeling sequential data in practice they suffer from vanishing/exploding gradient problems during training. [LSTMs](#) are specialized [RNNs](#) variants that try to mitigate this problems while maintaining the capability to capture long-term dependencies. The structure and independence assumptions of [LSTMs](#) mirror those of [State Space Models \(SSMs\)](#), as illustrated in [Figure 2.2](#) and described by [Equations \(2.6\) to \(2.7\)](#). Unlike standard [RNNs](#), [LSTMs](#) partition the latent state into two segments, $\mathbf{h}_t$ and $\mathbf{c}_t$, detailed in [Figure 4.2](#) and modeled by the following state and observation equations:

$$p_{\phi}(\mathbf{h}_t, \mathbf{c}_t \mid \mathbf{x}_{1:t}, \mathbf{h}_{1:t-1}) = p_{\phi}(\mathbf{h}_t, \mathbf{c}_t \mid \mathbf{h}_{t-1}) \quad (4.2)$$

$$p_{\theta}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{h}_{1:t}) = p_{\theta}(\mathbf{x}_t \mid \mathbf{h}_t, \mathbf{c}_t) \quad (4.3)$$

As in all deterministic models, the probability distributions can be assumed to have zero variance (i.e. Dirac deltas). The core mechanism of an [LSTM](#) network involves its ability to regulate information flow through three distinct gates: the forget gate, input gate, and output gate ([Figure 4.2](#)). The forget gate decides which information to discard from the cell state, the input gate determines what new information to add, and the output gate decides what part of the current cell state to output. This design allows [LSTMs](#) to selectively retain and update information over extended time steps and mitigate issues with gradients during learning. [LSTMs](#) have been extensively used in sequence modeling tasks like time series and video prediction ([Ehrhardt et al., 2017](#); [Shi & Yeung, 2018](#); [Srivastava et al., 2015](#)). For the temporal part, the baseline model employs a three-LSTM configuration, responsible for han-

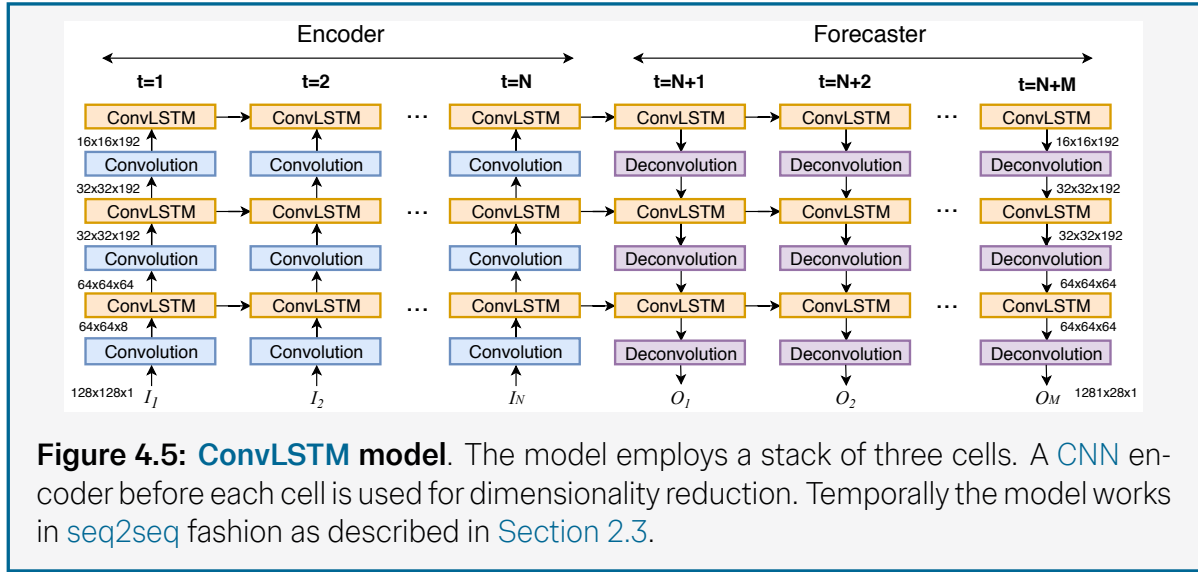




dling the first input, predicting the 10th frame (midway), and subsequent frames respectively. All of them use an internal representation of length 1000.

Convolutional Encoder and Decoder Using LSTM to model the system in pixel space, while feasible, has two main drawbacks. First of all the computational cost. Representing 5 frames with dimensions 128×128 pixels as a flattened vector would necessitate $n = 81,920$ elements and modeling this directly would require $\mathcal{O}(n^2)$ LSTM parameters. Furthermore, flattening would neglect the inherent spatial structure of images and would fail to capture crucial correlations between adjacent pixels. To address these issues, CNN encoder and decoder are incorporated for their efficiency in managing spatial data hierarchies. Convolutional layers utilize filters to detect patterns and textures in data, while the shared weights and local connectivity reduces the number of parameters, making them more efficient and less prone to overfitting compared to fully connected networks.

Architecture The model architecture builds directly on the work of Sorteberg et al., 2018. The encoder comprises of 4 convolutional layers with feature map counts progressively increasing as 60, 120, 240, and 480. Kernel sizes are 7, 3, 3, and 3 with padding 2, 1, 1, and 1 pixels respectively. Each convolutional layer incorporates batch normalization, followed by a **tanh** non-linearity, except the final layer. Stride of size 2 in all layers facilitates a halving of the reduction. Dropout (25% units) in the final layer aids in regularization. The encoder concludes with a fully connected layer 1000 neurons wide, producing the latent input for the three LSTMs. The decoder mirrors the encoder with respect to feature map sizes, employing deconvolutions that progressively double spatial dimensions until the initial pixel dimensions



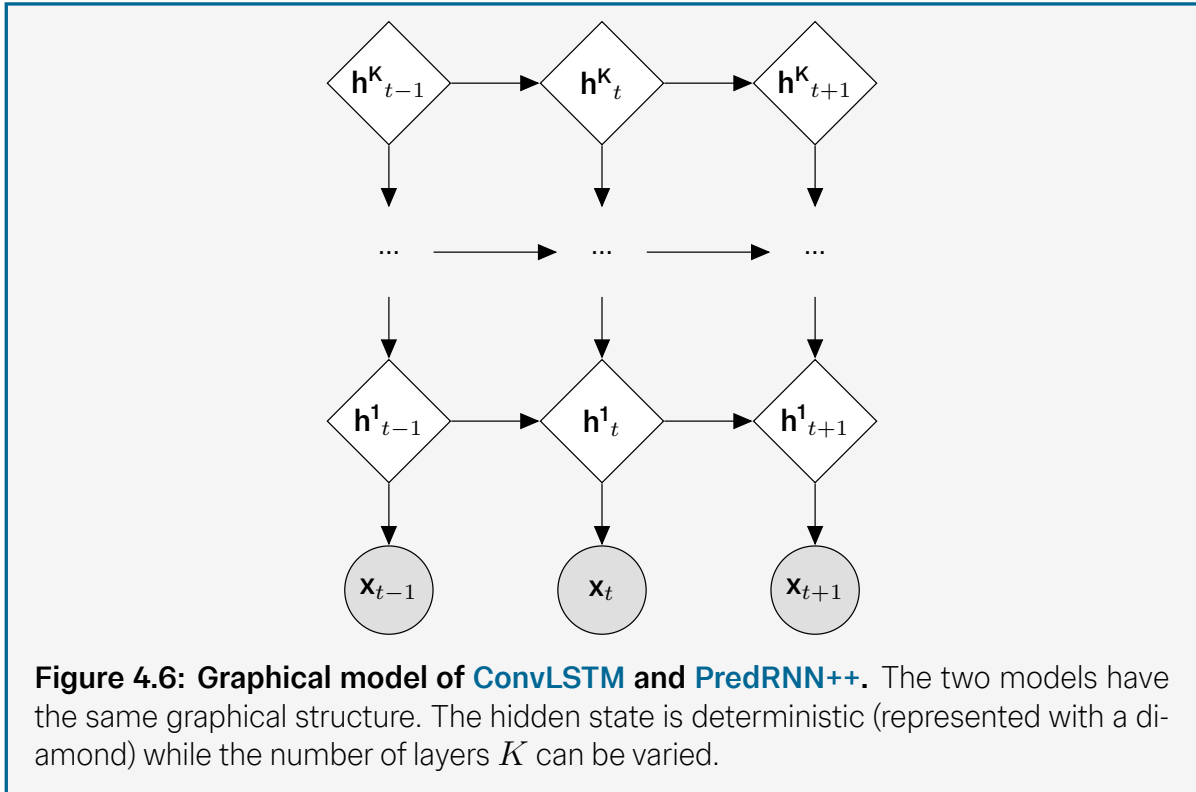
are reached. Kernel size 3, padding 1, and stride 2 (for upsampling) are used throughout. A visual representation of the model can be seen in Figure 4.3.

To generate predictions, the model employs a *sequence-to-sequence (seq2seq)* rollout, similar to the process outlined in Section 2.3. The encoder first provides a compact representation of each frame to the initial LSTM. This LSTM predicts the first future frame and then passes its state to a second LSTM, which unrolls up to timestep $t=9$. Subsequently, the third LSTM takes over, enabling forecasting for an arbitrary sequence length. At each step, a decoder maps the LSTM output back to pixel space. Refer to Figure 4.4 for a visual representation of this process.

4.3.2 Convolutional LSTM

The second model employed in this study is a ConvLSTM. ConvLSTMs try to mitigate the same problems of LSTMs mentioned in Section 4.3.1 namely the inability to capture spatial correlations and the quadratic scaling of parameters with respect to the internal state size. They achieve this by replacing fully connected layers within the LSTM cell with convolutional gates (Shi et al., 2015). Due to the weight sharing in CNNs, this reduces the number of parameters while at the same time exploits the structural information on the data. Information flow within a ConvLSTM cell mirrors that of a standard LSTM (Figure 4.2), but 3D tensors and convolutional operations replace 2D matrices and matrix multiplications. Originally designed for precipitation nowcasting, ConvLSTMs have since been successfully applied to other STF tasks like video prediction (Shi & Yeung, 2018).

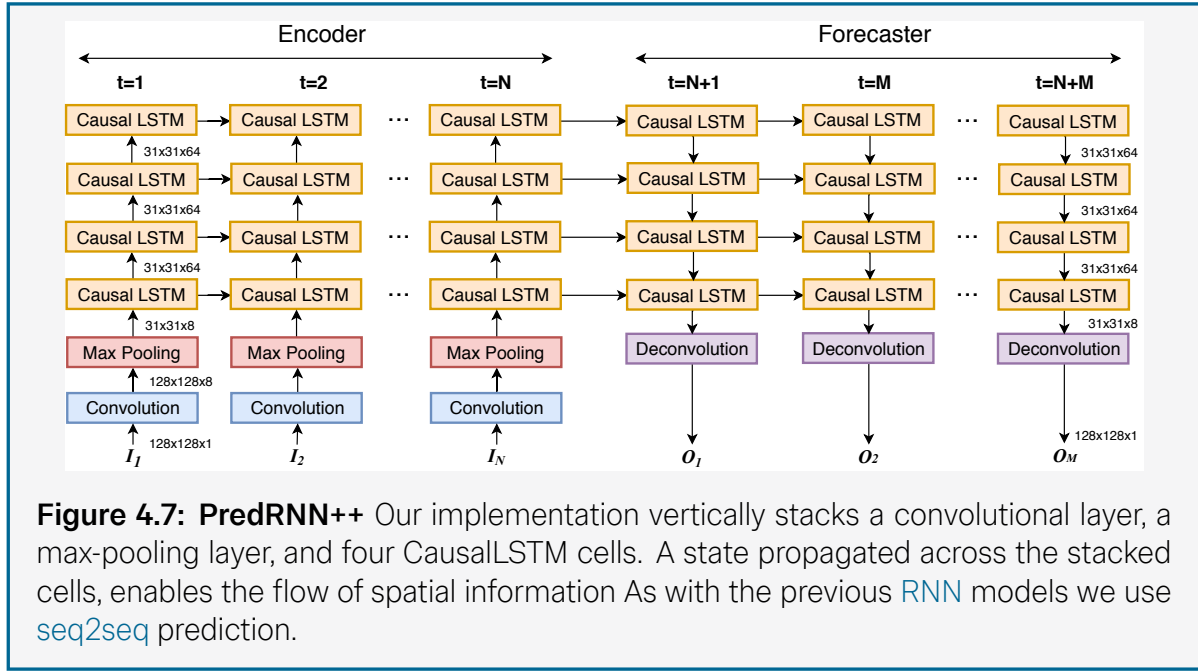
The architecture we evaluate in this study uses a stack of 3 ConvLSTM cells as shown in Figure 4.5. Convolutional encoders between these cells downsample spatial dimensions for computational efficiency. These encoders employ 4 layers (8, 64, 192 feature maps), 3×3



kernels, zero padding (width 1), and a stride of 2. These choices follow [Shi et al., 2015](#) but were adapted to accommodate hardware memory constraints and keep training time reasonable. The rest of the hyperparameters were set after some initial tuning. The activation function is a Leaky Rectified Linear Unit (ReLU) ([Xu et al., 2015](#)) with slope 0.2. At the final layer, the output is represented by a $16 \times 16 \times 192$ tensor. Inside the ConvLSTMs we use kernels of size 3 and zero padding of 1 pixel. This padding adds a single pixel of zeros around the edge of each input feature map, ensuring that the convolution operation does not reduce the spatial dimensions from one layer to the other. The decoder uses deconvolutions with stride 2 to double up the pixel dimensions in each layer. To generate predictions, the model employs a [seq2seq](#) rollout, similar to the process outlined in Section 2.3.

4.3.3 PredRNN++

The third and final recurrent model evaluated in this study leverages CausalLSTM cells from the PredRNN++ architecture ([Wang et al., 2018](#)). CausalLSTM cells, similar to ConvLSTM cells, employ convolutional gates. Furthermore the PredRNN++ architecture is very similar to the ConvLSTM one since models have a hidden state that comprises of many layers as shown in Figure 4.6. The difference is that PredRNN++ incorporates a state propagated across stacked cells, enabling the flow of spatial information as well. This design suggests potential advantages for long-term prediction of deterministic systems like wave propagation, where small short-term accuracy gains can significantly mitigate error accu-



mulation.

Our implementation follows and adapts the [PredRNN++](#) design for our specific use case with hardware limitations and parameter tuning guiding our choices. We vertically stack a convolutional layer, a max-pooling layer, and four CausalLSTM cells. The initial convolutional layer uses a 3x3 kernel, zero padding, and produces 8 feature maps. To accommodate GPU memory constraints, we downsample the input from 128×128 to 31×31 pixels using max-pooling with stride 4. We maintain four CausalLSTM cells but reduce the channel size to 64. These cells utilize 3x3 convolutional kernels. For prediction, a deconvolutional layer (kernel size 7, stride 4) restores the original dimensions. As with the previous RNN models we use [seq2seq](#) prediction. Refer to Figure 4.7 for a schematic and rollout illustration.

4.3.4 U-Net

U-Net

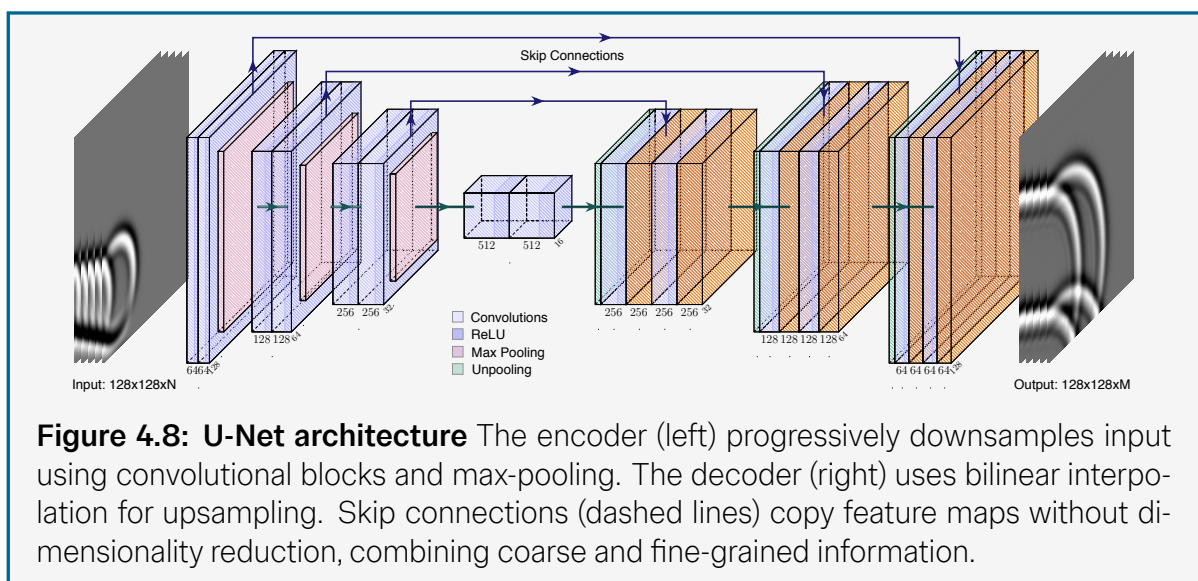
The FFNN proposed in this study draws inspiration from the U-Net architecture ([Ronneberger et al., 2015](#)), originally developed for biomedical image segmentation. In this context, U-Net receives a single image and outputs a dense per-pixel classification map, identifying object classes such as organs or lesions. Its success lies in the symmetric encoder-decoder structure with skip connections that preserve spatial resolution across scales.

Adaptations of U-Net have extended its utility beyond segmentation. In optical flow estimation, for instance, models such as FlowNet ([Dosovitskiy et al., 2015](#)) take two consecutive video frames as input (e.g., frames at t and $t + 1$) and predict a per-pixel motion field rep-

representing the displacement between the two. Unlike segmentation, which is a classification task, optical flow is a regression task with vector-valued outputs that encode both direction and magnitude of motion. This versatility has led to the successful application of U-Net variants to forecasting physical processes, such as predicting surface temperature fields (De Bézenac et al., 2019) and modeling fluid flow dynamics (Thuerey et al., 2018). In this work, we adopt a U-Net-based architecture to forecast wave propagation in a purely feed-forward manner, providing a strong baseline for spatiotemporal prediction.

As with all FFNNs, the U-Net’s graphical representation is straightforward: it directly maps observations $\mathbf{x}$ (sequences of frames in this case) to a latent representation $\mathbf{z}$, without explicit temporal dependencies (see Figure 2.3). Core in the U-Net is its encoder-decoder architecture, making it natural to identify the intermediate layers as the latent space $\mathbf{z}$. Additionally, the U-Net leverages skip connections, a technique popularized in ResNet architectures (He et al., 2016). These connections enhance the model by allowing the flow of information across different layers, aiding the propagation of gradients during training and the recovery of fine-grained details during testing.

The encoder consists of four blocks, each containing two convolutional layers (3×3 kernels, padding 1, ReLU non-linearities) followed by max-pooling (stride 2) for downsampling. Feature map count doubles in each block. The decoder uses bilinear interpolation (scale factor 2) for upsampling instead of deconvolutions to reduce the parameter count. Importantly, we retain the dimensions of feature maps copied via skip connections, contrary to the original design. This allows for the fusion of high-level, coarse features with fine-grained local information throughout the whole domain, without losing information in the boundaries. Hyperparameters were selected with some experimentation, using as initial point the original paper of Ronneberger et al., 2015. See Figure 4.8 for the a detailed depiction of the network architecture.



4.4 Dataset acquisition

The datasets were generated by simulating the [S-V equations](#), which describe shallow water wave dynamics. They are derived from the Navier-Stokes equations in the limit where the horizontal scale is much higher than the the vertical scale. Here we use the conservative [S-V equations](#) without the viscosity coefficient:

$$\begin{aligned} h_t + ((H + h)u)_x + ((H + h)v)_y &= 0 \\ u_t + uu_x + vu_y + gh_x - \nu(u_{xx} + u_{yy}) &= 0 \\ v_t + uv_x + vv_y + gh_y - \nu(v_{xx} + v_{yy}) &= 0 \end{aligned} \tag{4.4}$$

where h is the water height, H the reference height of the surface, u, v correspond to the horizontal velocity across the x and y axes respectively, g is the gravity acceleration, f is the Coriolis coefficient and ν the kinematic viscosity. ss

We employed the `triflow` package ([Cellier, 2019](#)) in `python` to simulate surface wave propagation governed by [Equation \(4.4\)](#). Coriolis effects and viscosity were omitted, and kinematic viscosity was set to $10^{-6} \text{ m}^2/\text{s}$ (approximating water at 20°C). Water depth (H) was fixed at 10 m. Tank size was randomized between 10 m and 20 m for each simulation, and initial wave excitations were modeled as Gaussian droplets placed randomly over the grid. We used reflective boundary conditions at the edge of the domain to study how wave interference is captured by the models. Frames were rendered using lighting positioned at 45° azimuth and 20° altitude. We structured each sequence to span 100 steps, with a temporal resolution of 0.01 seconds. Altogether, a total of 3, 000 sequences were crafted. The rendering resolution grid was set to 184×184 pixels.

4.4.1 OOD Datasets

To explore the models' ability to generalize [OOD](#), additional datasets were created by altering the physical characteristics of the simulations. More specifically we created the following datasets: **double droplet** was created by using two initial excitations at different positions, allowing to better study wave interference, **lines** where the initial excitation changed from circular to linear, **opposite and random illumination** where the illumination angle was changed, **shallow depth** where the tank size was half of the original one and **small and big tank** where the dimensions of the medium where altered, effectively changing the speed of the wave front. Details for all the datasets can be found in [Table 4.2](#) and visualizations of sample sequences from each dataset can be seen in [Figure 4.9](#).

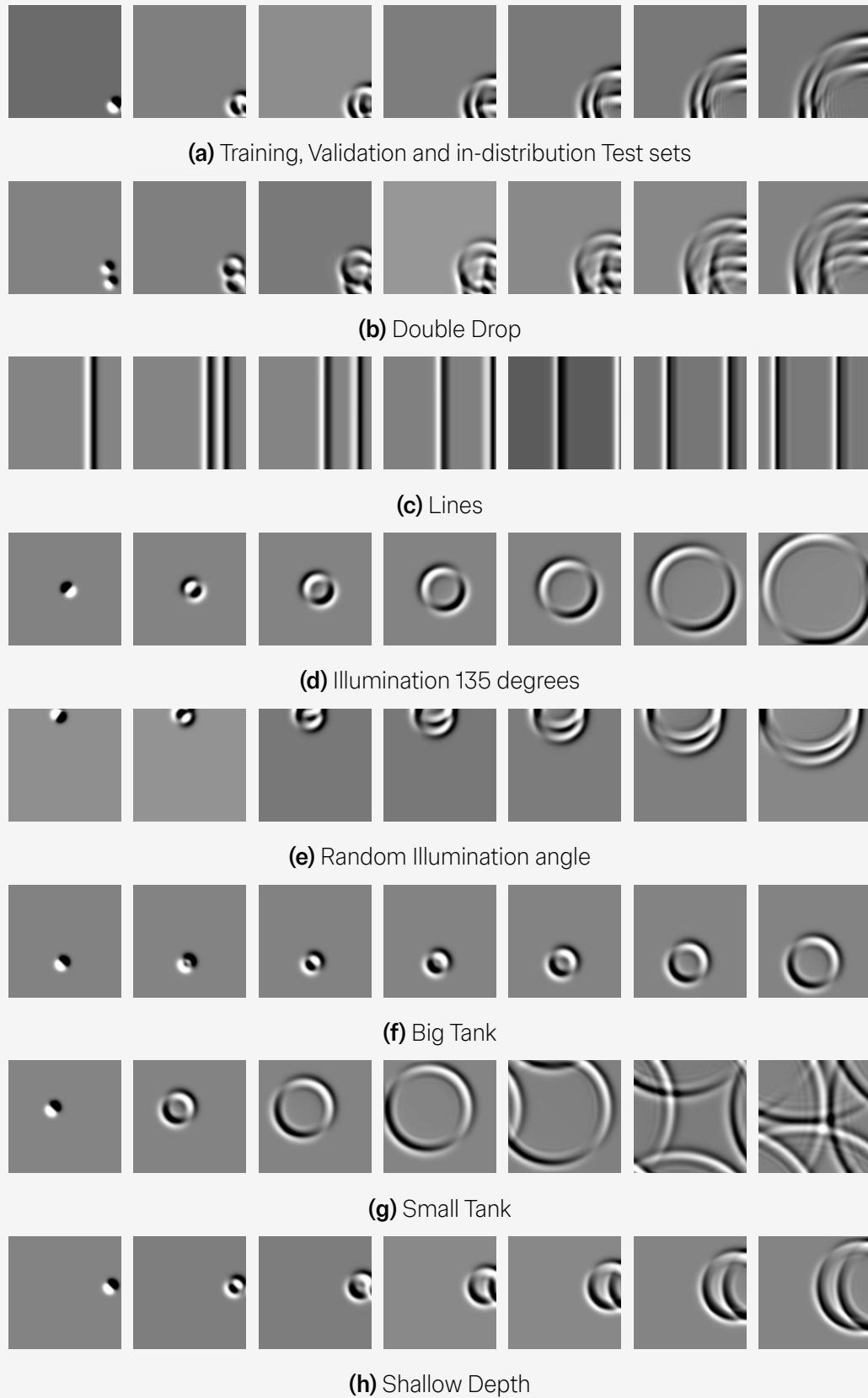


Figure 4.9: Sample sequences from the OOD datasets in which wavefront shape (b-c), observation model (d-e), and parameters describing the physical properties of the tank (f-h) are changed.

Table 4.2: The Train/Val/Test datasets were used for training, model selection, and in-distribution evaluation. The other datasets were used to evaluate the OOD generalisation capabilities of the models.

Dataset Name	Initial Cond.	Height	Tank Size	Illum. Angle	Sequences
Train/Val/Test	Droplet	10	[10, 20]	45°	3000
Double Drop	Double Droplet	10	[10, 20]	45°	500
Lines	Line Wave	10	[10, 20]	45°	500
Opposite Illumin.	Droplet	10	[10, 20]	135°	500
Random Illumin.	Droplet	10	[10, 20]	Random	500
Shallow Depth	Droplet	5	[10, 20]	45°	500
Small Tank	Droplet	10	[5, 10]	45°	500
Big Tank	Droplet	10	[20, 40]	45°	500

4.5 Experiments

4.5.1 Data pre-processing and split

To reduce the computational burden and allow for faster iterations of experiments the images were downsampled to a resolution of 128×128 . To enhance image prediction performance, we applied image normalization. Scaling the pixel values to a zero mean and unit standard deviation yielded optimal results. It's important to note that the normalization parameters were calculated solely from the training set and then applied correspondingly to the validation and test sets. Augmentation techniques (horizontal and vertical flips) were applied on a per-sequence basis. The original dataset of 3,000 sequences, was partitioned into 70% for training, 15% for validation, and 15% for testing.

4.5.2 Training and model selection

Assume that N is the number of input and M the number of output frames of the model. For each training iteration, we randomly selecting K sub-sequences of length $N + M$ from each simulated sequence. The models were trained to minimise the MSE over their respective output length M . In each iteration, the weights are updated using an Adam optimiser while a scheduling scheme adjusts the learning rate (LR) by a scaling factor of 10^{-2} if there is no improvement in validation error after a given amount of epochs (patience). The hyperparameters of interest are the input length $N \in \{3, 5, 10\}$, training output length $M \in \{1, 5, 10, 20\}$, samples per sequence between weight updates $K \in \{1, 3, 5, 10\}$, batch size $b \in \{16, 32\}$, LR $\in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and the learning rate schedule had patience $p \in \{3, 5, 7\}$ (based on validation loss) and scaling factor 0.1.

Model	Input Length	Output Length	Samples per Sequence	Batch Size	Learning Rate	Optimizer Patience
LSTM	5	20	10	16	10^{-4}	5
ConvLSTM	5	10	5	8	10^{-3}	7
PredRNN++	5	20	5	4	10^{-4}	3
U-Net	5	20	10	16	10^{-4}	7

Table 4.3: Hyper-parameters of the best performing model for each architecture. The input and output of the model correspond to one full forward pass. During sampling multiple samples per sequence are used per backward pass to reduce variance. The patience is how many epochs it takes without validation error improvement before the learning rate is reduced.

Grid search was used to find the best set of hyperparameters of each model. The training budget was fixed to 24h hours on a single GPU for all models and the best checkpoint was selected using a validation set. Since the objective is accurate long-term prediction, during model selection, we use prediction error over 50 frames regardless of the output size of the model M which is smaller. As mentioned in [Section 2.3](#) to obtain an arbitrary long prediction we feed the output as the next input. The final hyperparameters and model sizes can be found in [Table 4.3](#).

4.5.3 Code and dataset availability

Models were implemented in PyTorch. For researchers and practitioners interested in the underlying methodologies and in replicating the experimental results, the code for creating all the datasets and for running the experiments has been made publicly available at github.com/stathius/wave_propagation. We trained the models on a GTX 1060 GPU with 6GB of memory for 24h. Total training time includes evaluation overhead.

4.5.4 Model size and training time

Model size and training time per epoch matter because they directly impact the cost and feasibility of training deep learning models. The table shows a trade-off between model complexity and training speed. The [LSTM](#), despite having the most trainable parameters (88.2M), has the fastest epoch time (12m) among the [RNN](#) models. Conversely, the [PredRNN++](#) and [ConvLSTM](#) have significantly fewer parameters (12.3M and 2.5M respectively) due to their convolutional layers. However, they demonstrate much higher computational demands per epoch, likely because their architectures are more complex than the [LSTM](#)'s. Interestingly, the U-Net trains significantly faster per epoch than the other models. This is likely because the U-Net's feed-forward architecture is less computationally intensive than the recurrent nature of the [RNNs](#).

4.5.5 Evaluation Metrics

For model evaluation we use [Root Mean Square Error \(RMSE\)](#). RMSE is a standard way to measure the error of a model in predicting quantitative data. It represents the square root of the average of squared differences between predicted and actual values.

The RMSE between two images x and y is defined as:

$$\text{RMSE}(x, y) = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n [x(i, j) - y(i, j)]^2} \quad (4.5)$$

where m and n are the dimensions of the images, and $x(i, j)$ and $y(i, j)$ are the pixel values at position (i, j) in the two images.

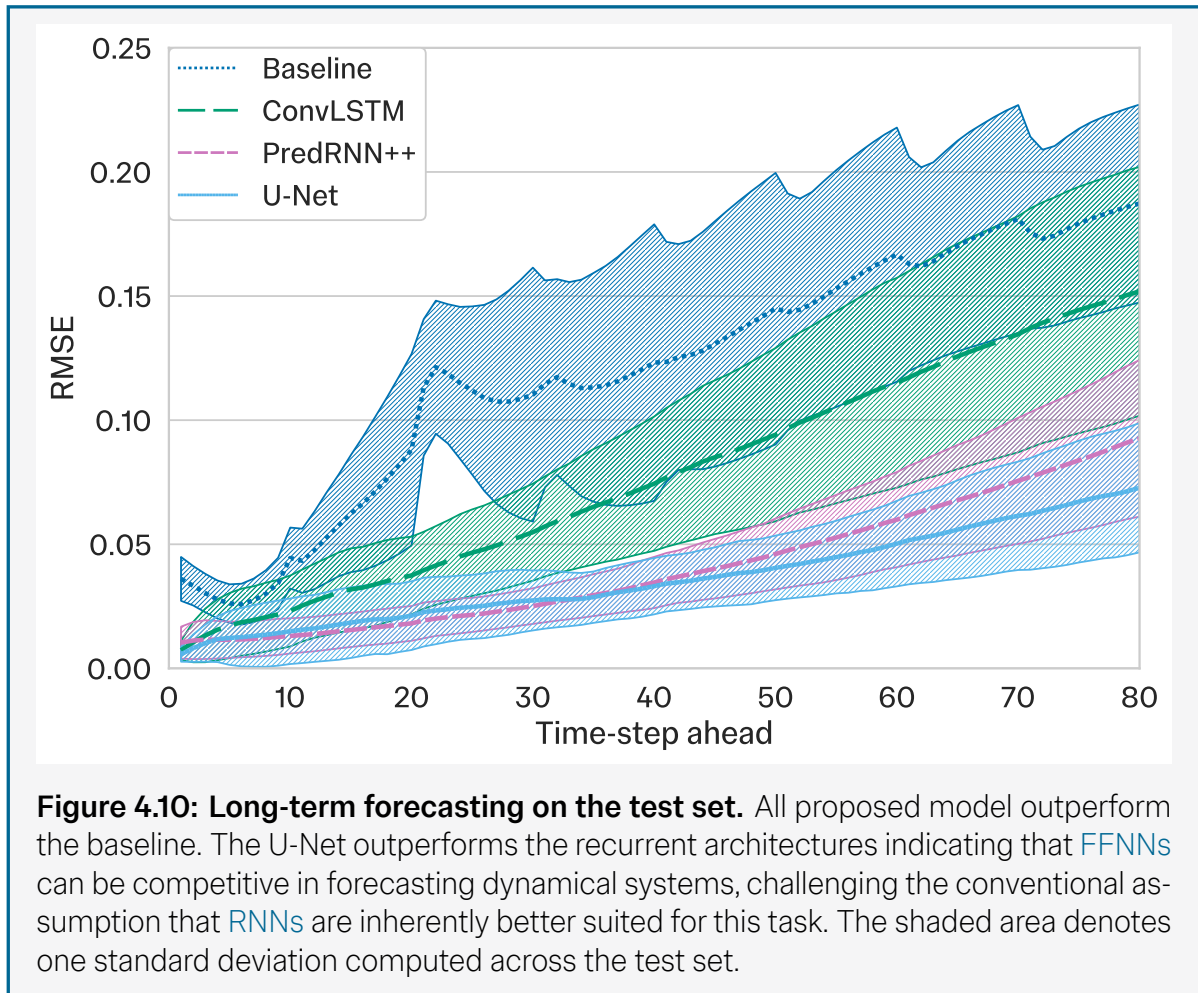
RMSE provides a measure of the average magnitude of the errors between predicted and observed values, with lower values indicating better performance.

Model	Trainable Parameters	Epoch Time	Num. Epochs	Best Epoch	Total Training Time
LSTM	88.2M	12m	75	71	24h
ConvLSTM	12.3M	36m	24	18	24h
PredRNN++	2.5M	33m	43	36	24h
U-Net	7.8M	8m	171	166	24h

Table 4.4: Model size and training times. The table illustrates that U-Net, a feed-forward model, achieves the shortest epoch times, likely due to its streamlined architecture, in contrast to the recurrent nature of [LSTM](#), [ConvLSTM](#), and [PredRNN++](#), which require more complex computations over sequences. Model size and training time per epoch are vital for assessing computational demands and practicality.

4.6 Results

4.6.1 Long-term prediction



The first objective is to evaluate how well the models extrapolate in time. Given ground-truth simulations of 100 frames in length, we tested the model predictions up to 80 time-steps, significantly longer than the maximum of 20 time-steps that the models sees upon training. The [RMSE](#) at each time step is calculated as an average over all the test sequences.

Results in [Figure 4.10](#) and [Table 4.5](#) show that the all the proposed models improve upon baseline. For the baseline the [RMSE](#) sharply raises after 10 frames and reaches 0.10 only after only 21 time-steps. A probable cause is the usage of three distinct [LSTMs](#) as described [Section 4.3.1](#), potentially hindering learning. The [ConvLSTM](#) offers an improvement staying below 0.1 [RMSE](#) for 53 time-steps with a gradual, near-linear error increases. [PredRNN++](#) further surpasses this, with a maximum [RMSE](#) of 0.091 at time-step 80, significantly lower than the baseline (0.186) and the [ConvLSTM](#) (0.150). This aligns with the findings of [Wang](#)

et al., 2017, that [PredRNN++](#) is more efficient than [ConvLSTM](#). The U-Net model excels in long-term prediction. While the [PredRNN++](#) remains competitive until about frame 35, the U-Net ultimately achieves superior long-term prediction, with an [RMSE](#) of 0.071 at frame 80. This translates to 62% error reduction compared to the baseline. To obtain uncertainty estimates we train each model 4 times with a different random seed. These results suggest that [FFNNs](#) like U-Net can be highly competitive in forecasting dynamical systems, challenging the conventional assumption that [RNNs](#) are inherently better suited for this task.

Figure 4.12 provides a spatial analysis of model predictions by plotting pixel intensity along the yellow line at timestep 80. While both models qualitatively capture the wave structure, discrepancies with the ground truth still occur. Error accumulation often manifests as dampening of wave amplitudes and/or phase shifts over extended forecasting horizons. Additionally, models may struggle to preserve sharp gradients or high-frequency components, leading to smoothed peaks. Inaccuracies in simulating wave reflections near domain boundaries—particularly if the analyzed slice is near the edge—can also compound over time.

At the same time, notable quantitative differences emerge between the two models. In this particular example, the U-Net’s wavefront appears to propagate slightly faster than the ground truth, resulting in a phase shift—its peaks arrive a few pixels earlier. This misalignment is most visible around pixels 30 to 80, where the wavefront exhibits the highest amplitude. In contrast, [PredRNN++](#) aligns more closely with the ground truth in terms of timing, capturing the peak locations more accurately.

This example illustrates how small deviations in wave propagation speed can lead to spatially localized errors, particularly around the center of the wavefront, where energy is concentrated and the dynamics are most sensitive. As we will see in the [OOD](#) analysis in [Section 4.6.2](#), such deviations play an important role in overall model performance.

Importantly, data-driven models also offer significant computational speed advantages over the numerical solver used for data generation. As shown in [Table 4.6](#), speedups range from $40\times$ for the baseline to $240\times$ for the U-Net. This establishes U-Net as the top-performing and most computationally efficient model. All runtimes are measured using commercial off-the-shelf hardware: a single GTX 1080ti GPU and an Intel i7 CPU with 8 cores. The data-driven models benefit from GPU parallelization, whereas the numerical simulator runs on the CPU. While these results already highlight significant efficiency gains, actual speedups may vary depending on specific hardware configurations.

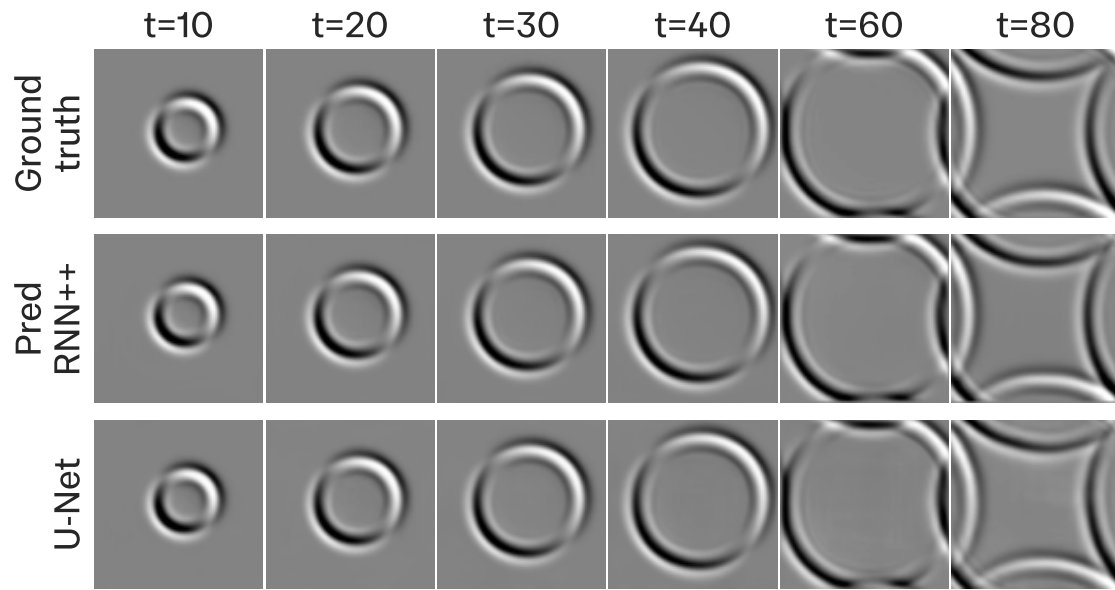


Figure 4.11: Visual comparison of U-Net and [PredRNN++](#) forecasts against ground truth. Both models produce visually similar and physically plausible forecasts on a test sequence.

Time-step ahead	10	20	40	60	80
Baseline	3.4 ± 0.3	8.2 ± 0.6	12.0 ± 1.4	16.4 ± 2.2	18.6 ± 3.1
ConvLSTM	2.2 ± 0.2	3.6 ± 0.3	7.2 ± 0.4	11.3 ± 0.5	15.0 ± 0.7
PredRNN++	1.3 ± 0.2	1.8 ± 0.4	3.4 ± 0.8	5.9 ± 1.1	9.1 ± 1.4
U-Net	1.4 ± 0.1	2.0 ± 0.2	3.2 ± 0.5	4.9 ± 0.8	7.1 ± 1.3

Table 4.5: RMSE comparison of model forecasting at various time steps. Best accuracy at each time step is highlighted in bold. Standard errors are reported across 4 models trained with different random seeds. U-Net's advantage may stem from its feed-forward Markovian design, which allows it to make accurate one-step predictions conditioned only on the most recent frames, thus avoiding the accumulation of errors that can arise in recurrent models like [PredRNN++](#) over long horizons.

Method	Time per frame (ms)	Speed-up
Numerical simulator	630.7	-
LSTM	15.0	40x
ConvLSTM	4.5	141x
PredRNN++	9.2	68x
U-Net	2.6	241x

Table 4.6: Computational efficiency of models. Time per frame and speedups relative to the numerical simulator are shown. U-Net demonstrates the fastest inference time. The data-driven models take advantage of GPU parallelization while the numerical simulator runs on the CPU. While our setup consisted of commercial off-the-self hardware, results may vary across hardware configurations.

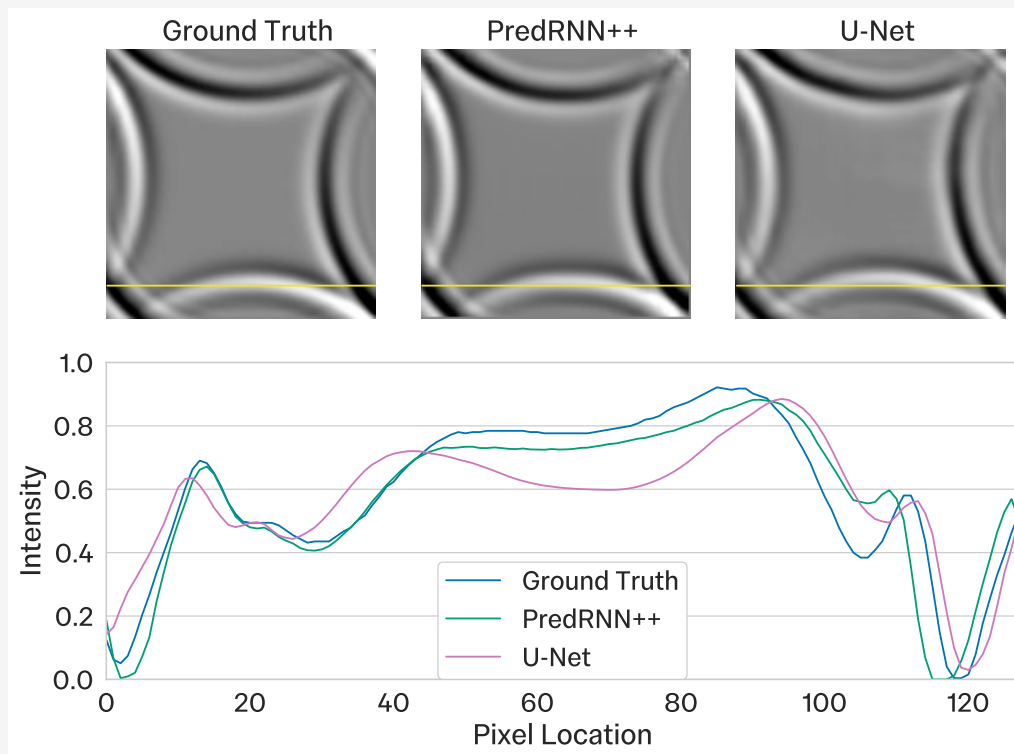
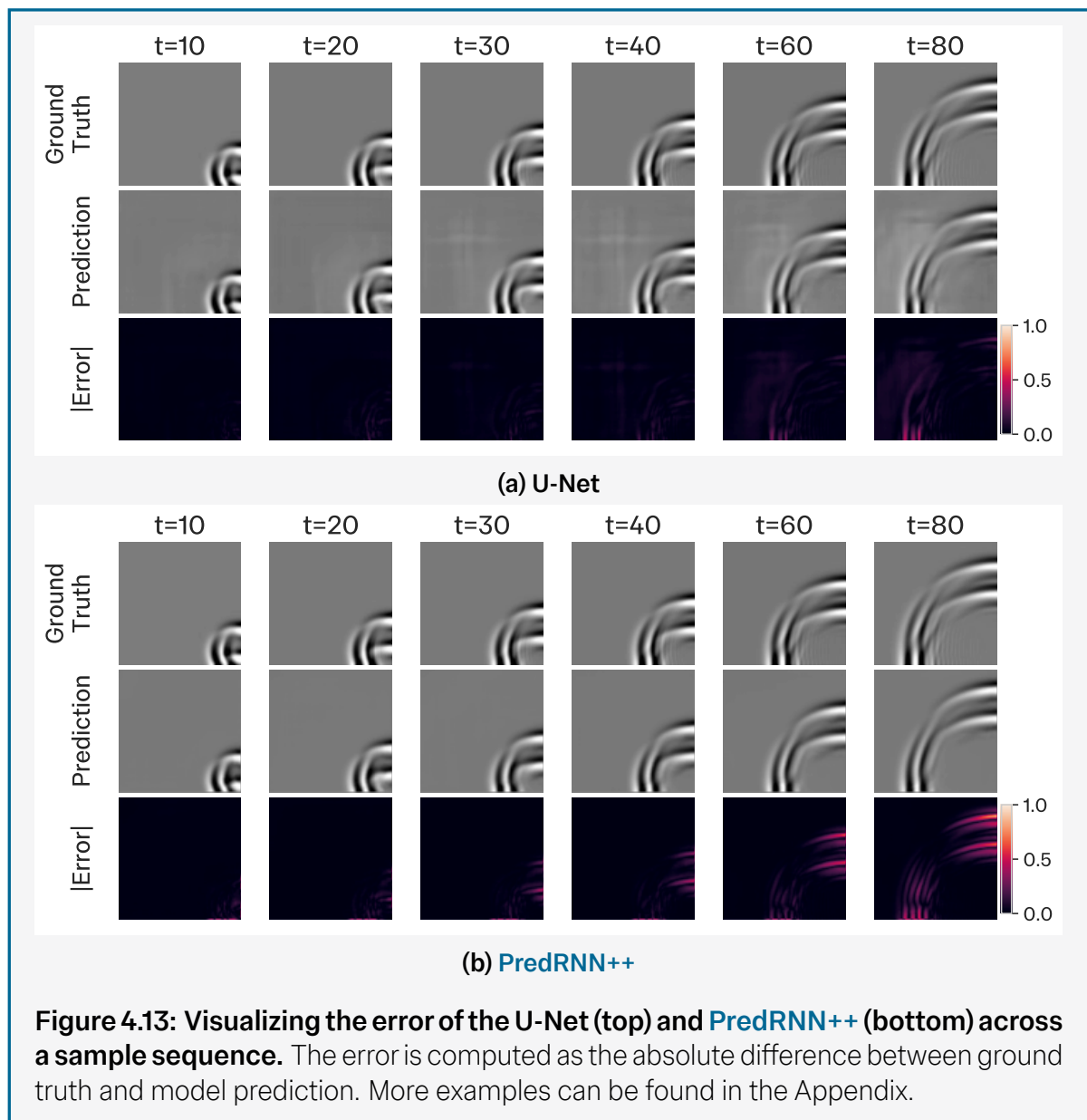


Figure 4.12: Detailed analysis at time step 80 (correspond to the last column of [Figure 4.11](#)). **Top:** visual comparison. **Bottom:** intensity plot along the yellow line highlights subtle differences.



4.6.2 Generalization to other physical settings

In this section, we evaluate the capabilities and limitations of our models by testing under different conditions. These conditions include changes in the initial excitation, illumination angle, tank dimensions and depth. To enable this, we created specific datasets for each case, as detailed in [Section 4.4.1](#). The conclusions we draw from the results are similar for all the models. For conciseness, we only present here the results of the U-Net and the [PredRNN++](#) and leave the rest of the models for the Appendix.

Before analyzing results, it's important to note that [RMSE](#) comparisons across datasets must be interpreted with caution. Each dataset has a unique level of intrinsic variation that impacts RMSE values. To illustrate this, we calculate the [RMSE](#) relative to a flat medium ([Figure 4.14](#)). The results illustrate higher RMSE in regions with complex dynamics, such as the small tank with its faster waves or areas near lines with stronger excitation. Alternative dataset variation metrics exist (like using prior frames) but we'll discuss those details in the Appendix.

Results ([Figure 4.15a](#), [Table 4.7](#)) demonstrate the U-Net's ability to generalize to unseen conditions. Particularly, U-Net seems to handle well changes in illumination angles used in rendering. While random illumination slightly increases the [RMSE](#) (from 0.07 to 0.10 at time step 80), remarkably, a drastic shift in illumination angle (from 45° to 135°) has no impact on the model's performance. This robustness is likely to arise from data augmentation techniques (like horizontal and vertical flipping), making the model invariant to orthogonal variations in illumination. Image augmentation techniques, such as flipping, rotation etc act as regularization and can significantly improve the generalization ability of [Deep Learning \(DL\)](#) models in image-based tasks ([Shorten & Khoshgoftaar, 2019](#)). Fundamentally, this type of augmentation acts as an inductive bias that enforces spatial symmetry i.e. the idea that the essential features and dynamics of a system should remain consistent even when orientation changes. This helps prevent overfitting and encourages the model to learn robust representations that focus on truly invariant aspects of the data.

On the other hand, modifications to the initial excitation present a greater challenge for the U-Net compared to illumination shifts. With linear waves, the [RMSE](#) reaches 0.11 at 20 frames, noticeably exceeding the 0.02 value observed on the in-distribution test set. Moreover, visualizations in [Figure 4.17](#) reveal qualitative differences in the prediction morphology, namely circular artifacts that resemble the training data. In the case of drops, while the [RMSE](#) of 0.04 at 20 timesteps is lower, the model sometimes struggles to resolve two distinct wave fronts, leading to blurring ([Figure 4.18](#)). The most challenging scenarios for the model arise when wave speed is altered, a consequence of changes in tank dimensions and depth. Reducing tank size leads to faster waves, resulting in [RMSE](#) reaching 0.10 within just 5 frames. Predictions in [Figure 4.18](#) illustrate how the network incorrectly propagates waves, failing to capture the system's characteristic time constant.

Time-step ahead	20	40	60	80
Test set	0.02	0.03	0.05	0.07
Opposite Illum.	0.02	0.03	0.05	0.07
Random Illum.	0.4	0.06	0.08	0.10
Double Drop	0.04	0.07	0.10	0.13
Lines	0.11	0.16	0.18	0.19
Shallow Depth	0.04	0.09	0.13	0.16
Big Tank	0.08	0.14	0.16	0.17
Small Tank	0.19	0.22	0.23	0.23

Table 4.7: OOD generalization of U-Net across datasets. Values correspond to [RMSE](#) at specific time steps. Performance varies across different physical settings.

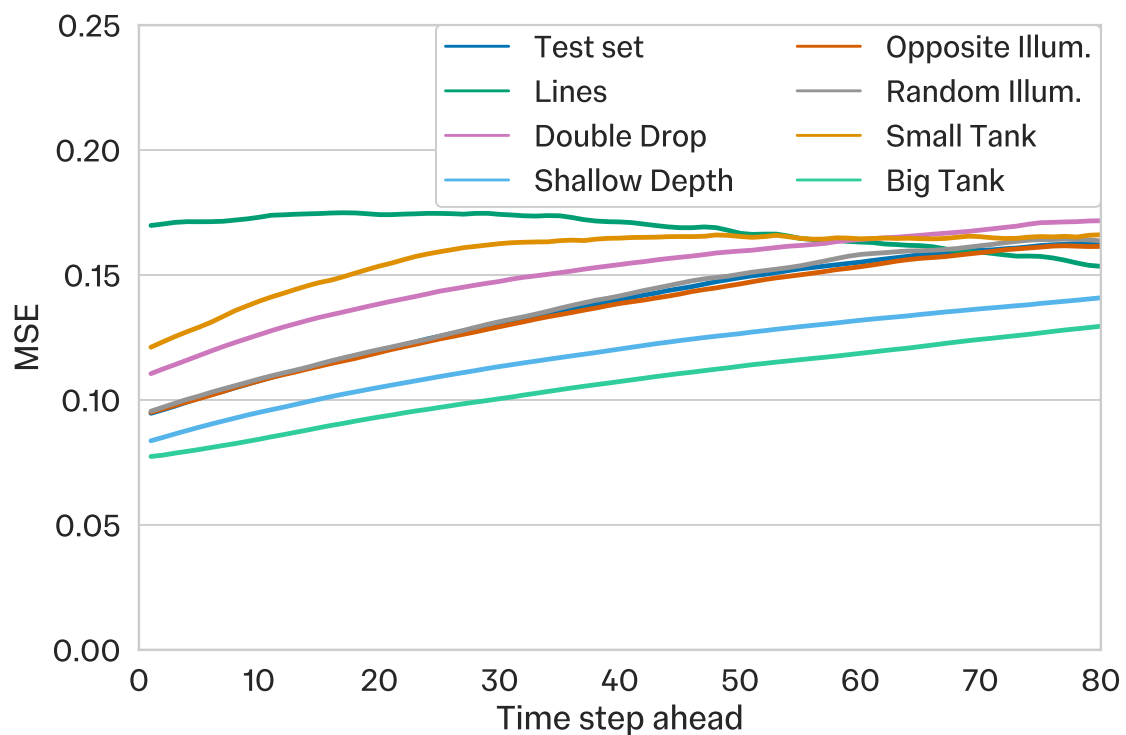
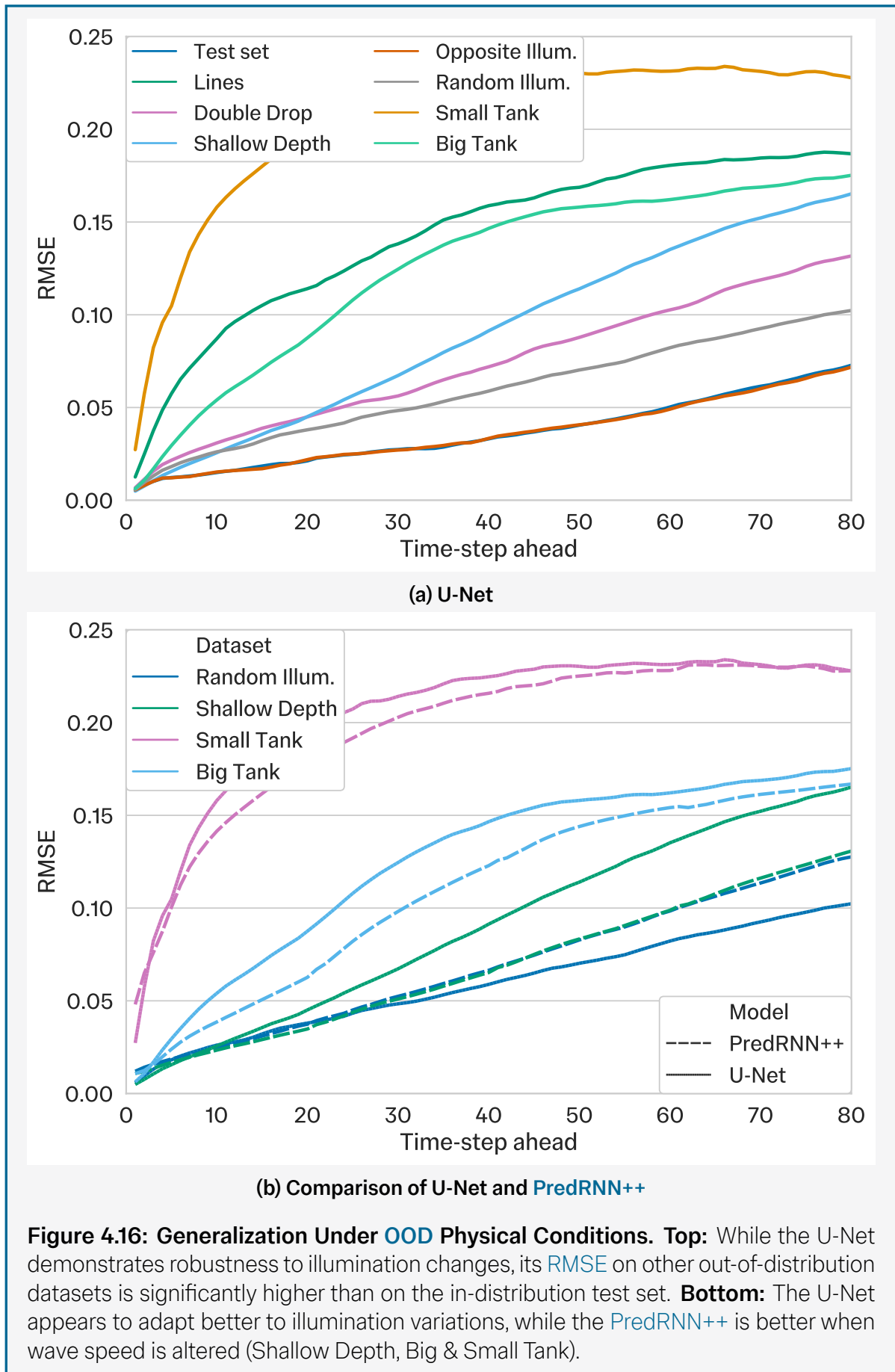


Figure 4.14: Inherent Variation of OOD Datasets. We use a flat image as a baseline to measure [RMSE](#) for each frame, quantifying dataset difficulty. Since datasets vary in complexity, this inherent variation must be considered when assessing a model's generalization capabilities.



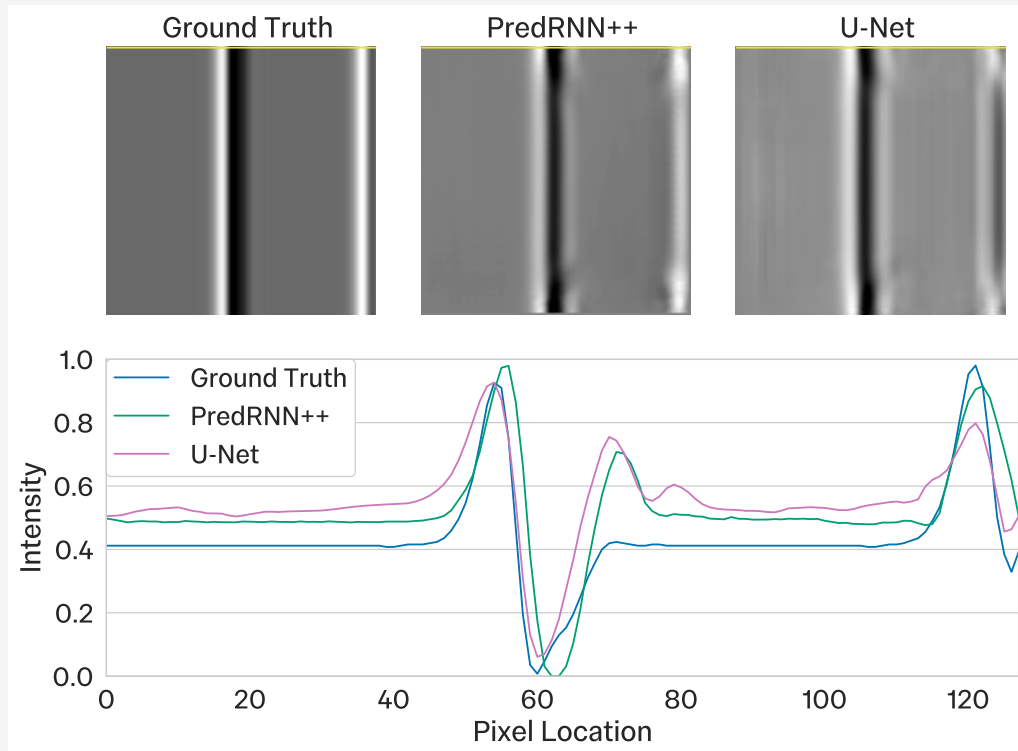


Figure 4.17: Detailed comparison between the U-Net and the [PredRNN++](#) on the Lines test set at time step 80. Top: visual comparison. **Bottom:** the intensity plot corresponds the line with the highest variation (in yellow) highlights subtle differences.. In this particular case, the [PredRNN++](#) has missed the characteristic time constant.

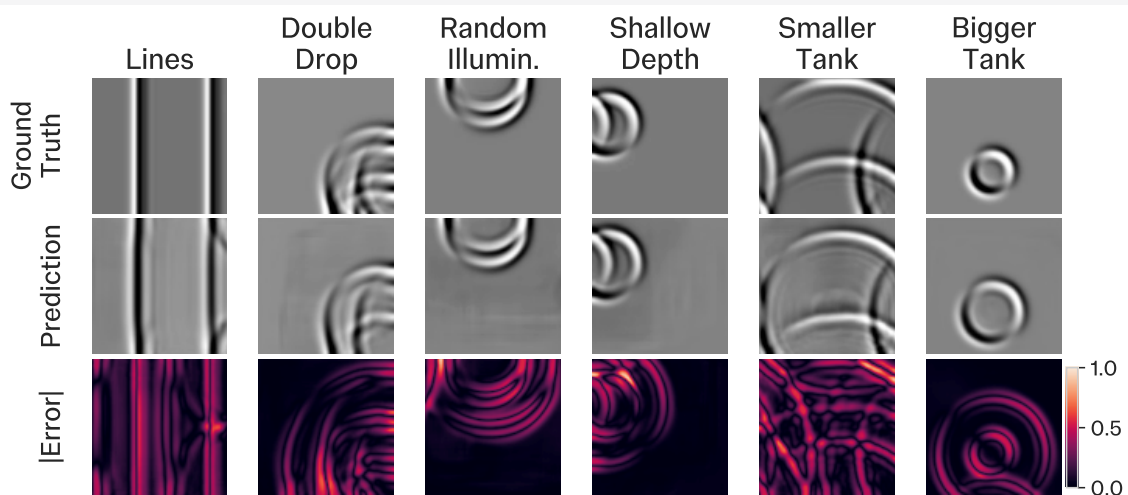
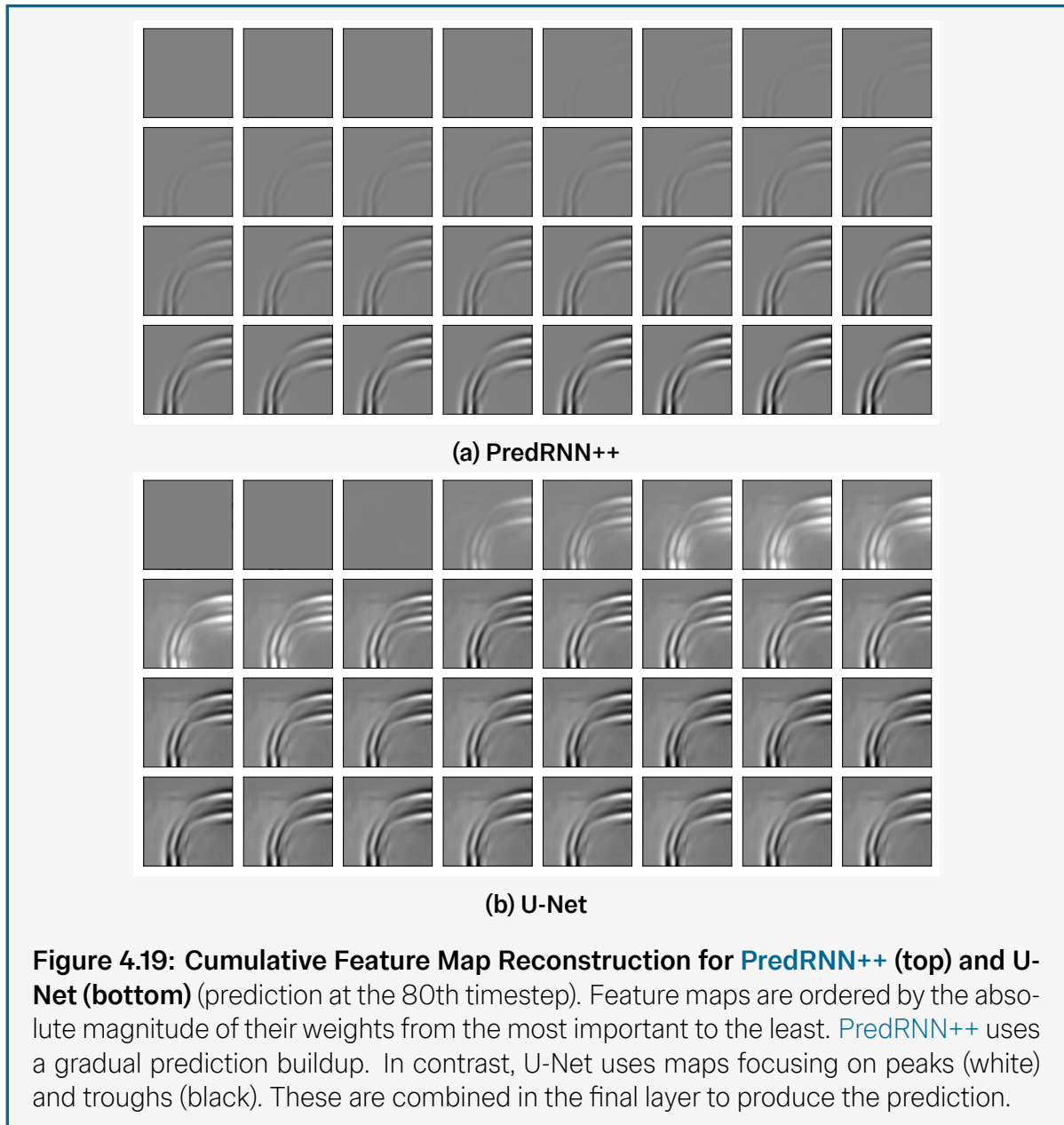


Figure 4.18: U-Net Predictions on [OOD Datasets](#) (at time $t = 80$). Note performance variations across datasets: a single blurry wavefront in the double drop scenario, and incorrect characteristic time constants for larger and smaller tanks. This highlights the impact of dataset-specific challenges on the model's generalization.

4.6.3 Internal Representations

In this section, we investigate the internal representations of the U-Net and **PredRNN++** during prediction. Specifically, we examine activations of the feature maps within the final convolutional layer of each model's decoder. To generate the final prediction frames, the models linearly combine these feature maps (without an activation function). While both networks have significantly more layers, this analysis provides insight into how they construct their predictions. As seen in [Figure 4.19](#), both models share some similarities but exhibit differences. Both possess flat feature maps, potentially serving as a foundation to build upon. However, their reconstruction methods are very different. **PredRNN++** appears to build its prediction incrementally in an additive process, while the U-Net exhibits a different behaviour with important maps corresponding to wave troughs while the less important maps to crests.



4.6.4 Physical Information in Latent Representation

In this section, we investigate whether the trained U-Net has developed any understanding of the system's physical properties, specifically the tank size (which inversely relates to wave speed). This analysis has two motivations:

- **Understanding Generalization Failure.** Explore potential reasons behind the U-Net's inability to extrapolate to different tank sizes.
- **Accessible Ground Truth:** Tank size information is already available, as each dataset sequence corresponds to a unique, square tank. Our training and testing datasets used tank sizes $s_i \in [10, 20]$ meters, with smaller tanks ($s_i \in [5, 10]$ meters) and larger tanks ($s_i \in [20, 40]$ meters) for out-of-distribution evaluation. This removes the need to create new datasets and facilitates comparison.

The key question is whether the U-Net's latent representation encodes any information about the underlying physical parameters of the system. Among these, we focus specifically on the tank size s_i , as it directly influences wave speed and governs the temporal dynamics of the system. Our decision to study s_i stems from the observation that variations in tank size led to the most substantial degradation in forecasting performance under OOD conditions, as indicated by significantly higher RMSE compared to other shifts such as illumination or excitation shape. This makes tank size a critical and informative variable for probing the limits of the model's internal representation and generalization capacity.

To explore this encoding, we take a pre-trained U-Net encoder and add a head consisting of two ResNet blocks (He et al., 2016) and a fully connected layer with a single output neuron (Figure 4.20). We train this network to predict tank size while freezing the encoder weights, allowing the new head to perform regression using the features extracted by the encoder. For comparison, we repeat this process with a randomly initialized encoder. We also compare with a dummy regressor that always predicts the mean tank size for each dataset (i.e., 15 for the test set, 7.5 for the small tank, and 30 for the big tank).

The results presented in Table 4.8 offer insights into how the U-Net encoder, pre-trained for wave propagation forecasting, internally represents physical parameters like tank size.

The encoder's strong performance in estimating tank size **within the training distribution** (RMSE of 0.14, significantly outperforming the random encoder and dummy regressor) can be explained by its implicit learning during the primary forecasting task. To accurately predict wave evolution, the U-Net must learn to process visual cues that are inherently modulated by the tank size. These cues include the speed of the wave front, the characteristics of wave reflections from the boundaries, and the overall scale of the wave patterns relative to the frame. The encoder, therefore, learns to extract features from the input frames that are cor-

related with these visual manifestations of different tank sizes. The subsequent regression head is then able to find a mapping from these learned, informative features in the latent space to the specific tank size values encountered during training. The random encoder, lacking this task-specific feature learning, provides no useful representation for the regression head, and the dummy regressor simply predicts the mean, which is a high-error strategy when there is variation.

However, the failure of this same pre-trained encoder to extrapolate to **OOD tank sizes** (i.e., smaller and larger tanks, where its RMSE is worse than the dummy regressor) highlights a crucial limitation. This suggests that the encoder has not learned a fundamental or causal understanding of how tank size dictates wave physics in a generalizable way. Instead, it has likely learned a set of correlations between visual patterns and tank sizes that are **specific to the range of tank sizes seen during training** ([10, 20] meters).

When faced with waves from significantly different tank sizes (e.g., [5, 10] meters or [20, 40] meters), the visual characteristics of the waves (e.g., much faster or slower wave propagation, different reflection timings) can fall outside the manifold of features the encoder was trained on. The latent representations generated for these **OOD** inputs may therefore be unpredictable or inconsistent with the patterns learned by the regression head. The mapping learned by the regression head is essentially brittle and overfitted to the feature distribution of the training data and cannot reliably extrapolate when the input features themselves shift due to OOD conditions.

The dummy regressor performs better in these **OOD** cases because its strategy is to predict the mean tank size of the **specific dataset it is being tested on**. For instance, if the **OOD small tank** dataset has a mean size of 7.5m, the dummy regressor predicts 7.5m. If the pre-trained encoder, confronted with unfamiliar wave patterns from these small tanks, produces latent features that lead the regression head to predict, for example, 15m (a value typical of the training set), its error will be substantial compared to the dummy regressor's simple, dataset-adapted guess.

In essence, while the pre-trained encoder's latent space successfully captures features that are **indicative** of tank size within its training experience, this representation is not robust enough to generalize its understanding of **tank size** to new physical regimes where the visual manifestations of wave dynamics differ significantly. The learned features are correlational shortcuts valid for the seen data, not an abstracted physical principle. This underscores the challenge of OOD generalization for deep learning models, a theme that motivates the exploration of techniques like supervised disentanglement in [Chapters 5 and 6](#).

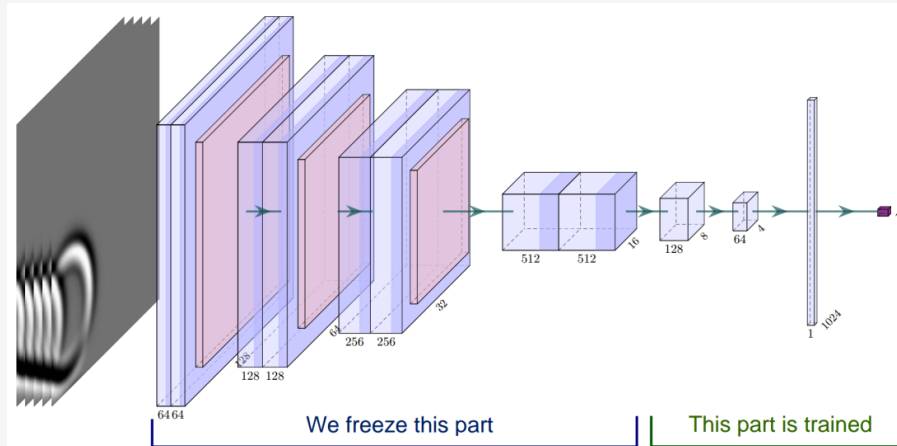


Figure 4.20: Schematic of the Tank Size Prediction Model The model combines the U-Net encoder (Section 4.3.4) with a new head comprised of two ResNet blocks and a fully connected layer with a single output neuron. This architecture enables to do regression on the tank size using features extracted by the encoder.

	Test set	Bigger Tank	Smaller Tank
Pre-trained encoder	0.14	6.65	2.23
Random encoder	2.27	14.21	6.35
Dummy regressor	2.45	5.19	1.22

Table 4.8: Predicting Tank Size from U-Net’s Latent Space. The pre-trained model demonstrates the ability to estimate tank size with relatively low error. In contrast, the randomly initialized encoder performs poorly, only slightly outperforming a dummy regressor (which always predicts the dataset’s mean tank size). However, both pre-trained and random encoders fail to extrapolate to smaller and larger tank sizes.

4.7 Concluding remarks

In this work, we demonstrated the potential of data-driven methods, specifically deep networks, for approximating wave propagation. Our U-Net architecture achieved a significant reduction in long-term [RMSE](#) to 0.071, outperforming the previous baseline of 0.186. Notably, it surpassed state-of-the-art recurrent architectures, suggesting that [FFNNs](#) can be highly competitive in forecasting dynamical systems. This challenges the common assumption that [RNNs](#) are inherently superior for such tasks.

While [RNNs](#) are designed to capture long-term temporal dependencies, their advantage may be less pronounced when the dynamics are primarily local in time. We attribute the U-Net's success partly to this locality, as simulated data from low-order Markovian solvers may not require the extended memory mechanisms of [RNNs](#). This aligns with previous findings on the efficacy of [FFNNs](#) for modeling 1-dimensional sequences ([Bai et al., 2018](#)). However, a deeper understanding of the U-Net's performance in this domain remains an open question.

One of the key strengths of the U-Net architecture is its computational efficiency. In our experiments, U-Net achieved a 240-fold speedup compared to the fourth-order numerical simulator, with both systems running on commercial off-the-shelf hardware (GTX 1080ti GPU for the models and Intel i7 CPU for the simulator). This acceleration, coupled with low prediction error, underscores the potential of deep learning models to effectively balance precision and computational resource requirements.

Crucially, we devoted some attention to [OOD](#) generalization. Our evaluation underscored the challenge of data-driven models to generalize to unseen scenarios. To mitigate this limitation, [Chapter 5](#) explores simpler systems and generative models, introducing a dynamics-based latent space disentanglement approach designed to improve model robustness and explainability.

Chapter 5

Disentanglement for dynamical system forecasting

Deep Neural Networks (DNNs) excel at modeling complex systems, but achieving long-term prediction accuracy and generalization to unseen scenarios remains challenging. In this chapter, we address these issues by treating system parameters as key factors that cause variations in the data. By incorporating actual parameter values during training, we help disentangle the representations learned by deep generative models. We tested this approach on three different dynamical systems, and our results show that latent-space supervision significantly improves the accuracy and robustness of long-term predictions, even for OOD data.

The methods and part of the results discussed in the present chapter have been published in (Fotiadis et al., 2021) and (Fotiadis et al., 2023). For researchers and practitioners interested in the underlying methodologies and experimental results, the code and datasets have been made publicly available at <https://github.com/stathius/sd-vae>.

5.1 Introduction

One of the main challenges in machine learning and engineering is predicting how dynamical systems will behave. Robust forecasting of such systems is crucial for everything from weather modeling (Garg et al., 2021) to innovations in autonomous systems (Hafner et al., 2019), and engineering design (Brunton et al., 2020).

Data-driven methods like DNNs are promising tools for sequence modeling (Girin et al., 2020) but face two core robustness challenges: **long-term prediction** and **OOD generalization**. In many dynamical systems, error accumulation leads to forecast divergence over time (Raissi et al., 2019; Zhou et al., 2021). This problem also impacts traditional numerical solvers. Additionally, machine learning models often struggle with poor OOD performance when faced with new scenarios, as noted by Goyal and Bengio, 2020. We have explored and discussed both of these issues in Chapter 4.

In modeling dynamical systems, it's crucial to realize that training data encompasses two distinct elements: the underlying structure of the model, and the specific parameters that govern those dynamics. Here, we concentrate on systems governed by ODEs and start our analysis by giving a concrete definition of what constitutes an OOD scenario. We further observe that DL approaches fail to differentiate between dynamics and parameters, potentially leading to entangled representations. This entanglement can result in overfitting and ultimately hurt forecasting robustness (Bengio et al., 2013).

Our proposed method combines two concepts. Firstly, we leverage the ability of Variational Autoencoders (VAEs) (Kingma & Welling, 2014) to learn disentangled representations unsupervised (Higgins et al., 2017), a feature successfully applied in image analysis (Kim & Mnih, 2018). Secondly, we incorporate ground-truth factors during training, a technique known to further enhance disentanglement in image modeling (Locatello et al., 2019b). We provide theoretical justification for the use of disentangled representations through an analysis of dynamical systems in the scope of state space modeling (see Section 2.2.2). Practically, we treat system parameters as key factors of variation and use their ground-truth values to guide latent space disentanglement. While unsupervised disentanglement has been explored for dynamical systems (Li & Mandt, 2018; Miladinović et al., 2019), our approach is the first to utilize supervised disentanglement in this context. Furthermore, unlike system identification methods that demand complete knowledge of the system (Ayyad et al., 2020), our technique requires only the system parameters themselves.

The main contributions of this study are the following:

Defining OOD in Dynamical Systems: We introduce a clear definition of OOD behavior in dynamical systems. OOD scenarios occur when a model encounters parameter ranges of the governing differential equations that were not present during its training. This means the model has only observed a subset of the possible system dynamics. Our definition contrasts with previous work where individual parameters from the same range were treated as OOD (Miladinović et al., 2019).

Supervised Disentanglement in Dynamical Systems: We highlight the crucial role of parameter identification for robust forecasting of dynamical systems. To address this challenge, we introduce supervised disentanglement, a novel approach that leverages ground-truth system parameters to guide the disentanglement of the latent space within generative models like VAEs.

Improved Accuracy and Disentangled Representation: We encourage latent space disentanglement using supervision as a form of regularization. We conduct experiments with VAEs trained with noisy observations of the phase-space of three dynamical systems and show improved accuracy in- and out-of-distribution. We analyze the latent space of disentangled models and show that they are effectively disentangled compared to normally trained models.

However, the practical importance of our method is currently limited by the label cost. It would be worth assessing the model in the semi-supervised or sim2real setting, as that would be better suited for real-world applications. All the necessary code to reproduce our experiments is provided at <https://github.com/stathius/sd-vae>.

5.2 Problem Formulation

Governing Equations We consider a class $\mathcal{C}$ of deterministic dynamical systems described by a set of ODEs:

$$\frac{d\mathbf{x}_i}{dt} = \mathcal{F}(\mathbf{x}_i, \boldsymbol{\xi}_C) \quad (5.1)$$

Here, $\mathbf{x}_i \in \mathbb{R}^d$ represents the system state, $\mathcal{F}$ encodes the governing dynamics, and $\boldsymbol{\xi}_C \in \mathbb{R}^k$ holds the system parameters. While these equations define the system's evolution, finding analytical solutions is often impossible for real-world problems. Consequently, we usually rely on numerical methods to simulate the system's behavior over time, given the equations and an initial state. In general, long-term prediction is non-trivial. As numerical errors accrue, at some point predictions diverge from the ground truth.

Modeling Observational Noise Experimental observations $\tilde{\mathbf{x}}$ inevitably contain some degree of noise: $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}_t \in \mathbb{R}^d$ represents random, uncorrelated noise. To simulate this in our experiments, we add white Gaussian noise to the data (see Section 5.5).

Experimental Case Studies Our experiments focus on dynamical systems described by ODEs, though our methods could potentially be extended to PDEs. We investigate three diverse case studies:

$$\text{Simple pendulum: } \ddot{\theta} = -\frac{g}{\ell} \sin \theta \quad (5.2)$$

$$\begin{aligned} \text{Lotka-Volterra system (L-V system): } \dot{x} &= \alpha x - \beta xy \\ \dot{y} &= \delta xy - \gamma y \end{aligned} \quad (5.3)$$

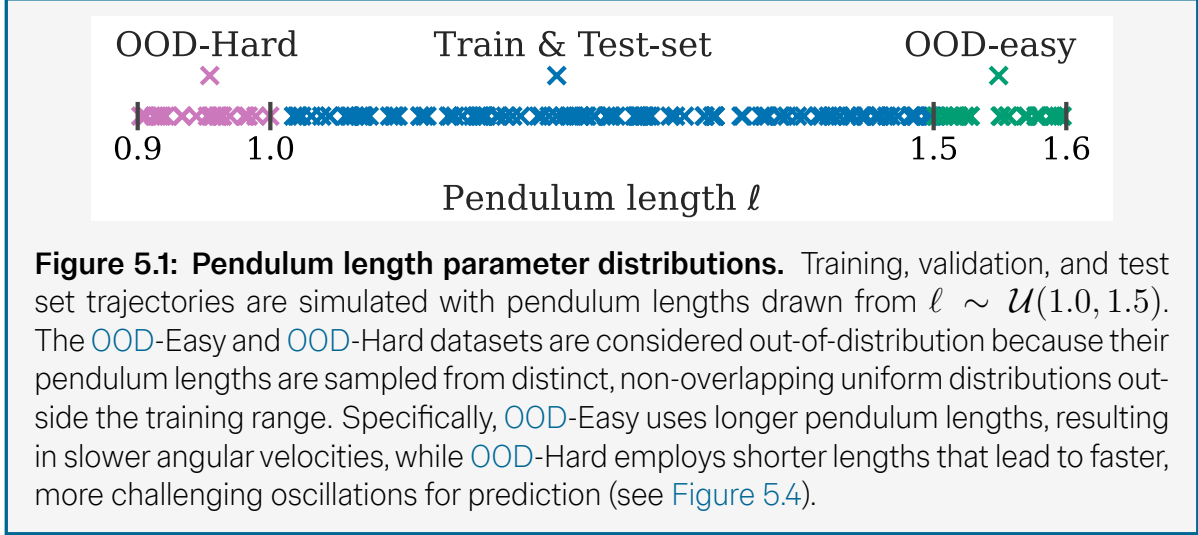
$$\text{3-body system: } \dot{\mathbf{v}}_i = K_1 \sum_{j \neq i} m_j \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^3} \quad (5.4)$$

$$\begin{aligned} \dot{\mathbf{x}}_i &= K_2 \mathbf{v}_i \\ \mathbf{v}_i, \mathbf{x}_i &\in \mathbb{R}^2, i \in [1, 2, 3] \end{aligned}$$

These systems were chosen to capture the range of complexities found in dynamical systems. They differ in their degrees of freedom, the nature of their governing equations, and the number of parameters that influence their behavior.

Observable Quantities and Parameters

Swinging Pendulum: The angle of the pendulum from its vertical resting position is denoted by θ . Gravitational acceleration (g) remains constant in our experiments, while the pendulum's length (ℓ) serves as the primary variable parameter.



L-V system (Predator-Prey Dynamics): This system models the populations of prey (x) and predators (y). The parameters $\xi_C = [\alpha, \beta, \gamma, \delta]$ model the birth rates, death rates, and interactions between the two species.

Planar 3-Body System: We track the positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ of the three bodies. The system's parameters ξ_C contain the constants K_1, K_2 and the masses of the three bodies (m_1, m_2, m_3). Employing two constants related to the gravitational force helps in rendering the system dimensionless (Section 5.5).

5.2.1 Definition of OOD Dataset

A dynamical system's evolution is fundamentally shaped by the form of $\mathcal{F}$ in Equation (5.1). With $\mathcal{F}$ fixed, the next most influential factor is the set of parameters, ξ_C . If these parameters are drawn from a distribution $\xi_C \sim p(\xi_C)$, the resulting trajectories will follow a corresponding distribution $\mathbf{x}_{1:t-1} \sim p(\mathbf{x}_{1:t-1} \mid \xi_C)$, where $\mathbf{x}_{1:t}$ denotes the system's state evolution up to time t . Due to the nature of dynamical systems, even small variations in parameters can lead to vastly different trajectories (Lai & Winslow, 1994) and changes in the parameter distribution will significantly impact the trajectory distribution. This is visualized in Section 5.5.

We define an OOD dataset as one containing trajectories generated from a parameter distribution that is disjoint from the one used to produce the training data. For each system, we initially sample parameters ξ_C (and initial conditions I_C for the pendulum) from a uniform distribution. This creates our in-distribution training, validation, and test sets. We subsequently create two additional OOD datasets by sampling parameters from distinct, non-overlapping uniform distributions. The first dataset, referred to as OOD-Easy, uses longer pendulum lengths, which result in slower angular velocities compared to the in-distribution sequences. In contrast, the second dataset, OOD-Hard, employs shorter pendulum lengths, leading to faster angular velocities that are more challenging for the model to accurately

predict than the slower ones. [Figure 5.1](#) illustrates the parameter distributions for pendulum lengths in each OOD dataset.

Clarification on Terminology: The labels **OOD-Easy** and **OOD-Hard** refer to the proximity of the parameter distribution to the in-distribution range, rather than prediction difficulty. Specifically, **OOD-Easy** corresponds to parameters closer to the training set distribution, while **OOD-Hard** involves parameters further away. While these shifts may correlate with predictive difficulty in some systems (e.g., faster dynamics in shorter pendulums), the naming reflects distributional distance, not empirical performance.

Realistically, a single training set cannot encompass the entire distribution of possible trajectories ([Fotiadis et al., 2020](#)). Robust OOD prediction often demands additional inductive biases ([Barber et al., 2021](#); [Bird & Williams, 2019](#)). Our method introduces such a bias by leveraging ground-truth parameters to guide the disentanglement of the latent representation. Contrary to previous work, which defines out-of-distribution (OOD) parameters as those sampled from the same distribution but not encountered during training ([Cranmer et al., 2020](#); [Miladinović et al., 2019](#)), we broaden the notion of OOD to a more challenging setting where the distributions are entirely disjoint. For details, see [Table 5.1](#).

5.3 Motivating Disentanglement

Disentanglement offers both empirical and theoretical benefits for dynamical system prediction. By aiding parameter and initial condition estimation, it reduces uncertainty in the system's dynamics and allows us to simplify the learning problem.

5.3.1 Theoretical Motivation for Disentanglement

Forecasting dynamical systems can be framed as a time-series modeling problem, as we discussed in [Section 2.1](#). We consider a class C of deterministic systems with unknown time-invariant dynamics, parameterized by unobserved parameters ξ_C , and aim to predict the system's state evolution up to time $t+n$, given observations up to time t . Initial conditions I_C are also unobserved and introduce another source of uncertainty. Under this probabilistic framework, we seek to estimate $P(\mathbf{x}_{t:t+n} \mid \mathbf{x}_{1:t-1}; \xi_C, I_C)$.

In practice, we often lack access to ξ_C and I_C . Here are two potential modeling approaches. With a Bayesian approach, we could assign priors on ξ_C, I_C and marginalize over them to estimate $p(\mathbf{x}_{t:t+n} \mid \mathbf{x}_{1:t-1})$. However, the potential divergence of trajectories for different parameters (and initial conditions) makes both assigning accurate priors and efficient marginalization challenging. Alternatively, we could directly estimate ξ_C and I_C , and then model the conditional distribution. This approach becomes more appealing if we can obtain good estimates of ξ_C and I_C , and it's where disentanglement could offer benefits.

Pendulum Example: Consider a pendulum where only its length, $l \in \xi_C$, varies across trajectories. This is actually the only free parameter in the pendulum model, assuming that gravity is constant. Let all other parameters and initial conditions remain fixed and known. Under the assumption that we have a good estimator for l , then the conditional $p(\mathbf{x}_{t:t+n} \mid \mathbf{x}_{1:t-1}, l)$ is less complex than the marginal $p(\mathbf{x}_{t:t+n} \mid \mathbf{x}_{1:t-1})$ and presents a better modeling target. This is because knowing the length removes a source of uncertainty from the system's dynamics.

Role of Disentanglement:

Role of Disentanglement The motivation for disentangled representations comes from practical constraints. In theory, with unlimited data and computational power, a sufficiently powerful model could implicitly learn all relevant factors of variation. However, in real-world settings where data and capacity are limited, data-driven models benefit from guidance. By supervising specific latent dimensions with known factors, such as the pendulum's length

l , we reduce the burden on the model to infer everything from scratch. This supervision effectively unveils a previously unseen quantity, narrowing the search space and improving sample efficiency. Framing DNNs as Probabilistic Graphical Models (PGMs), this approach allows us to decompose the latent space into two parts: $\mathbf{z}_{1:t}$, which captures the system’s evolving dynamics, and $\mathbf{z}_l$, which encodes static, interpretable parameters like length. The resulting conditional distribution $p(\mathbf{x}_{t:t+n} \mid \mathbf{z}_{1:t}, \mathbf{z}_l)$ is both easier to learn and more faithfully mirrors the true data-generating process. If the model successfully encodes l into $\mathbf{z}_l$, it can concentrate its capacity on modeling the remaining uncertainty in the system’s dynamics. This decomposition has been empirically shown to improve predictive performance (Section 5.3.2). A practical method to achieve this supervision in VAEs is introduced in Section 5.4.

5.3.2 Experimental Motivation

Disentangled representations have a proven track record of enhancing downstream task performance and reducing overfitting (Bengio et al., 2013). In image generation, they enable controlled image synthesis based on specific attributes. For image reconstruction or inpainting, they preserve existing features while filling in missing information (Higgins et al., 2017; Locatello et al., 2020a). Disentangled representations also offer advantages for time-series prediction. They can effectively separate appearance from underlying dynamics (Li & Mandt, 2018) or disentangle trends, seasonality, and noise (Li et al., 2022).

Specifically in dynamical systems, previous work demonstrates that VAEs can recover physical system parameters within their latent space (Iten et al., 2020). Our own experiments corroborate this, revealing that even a standard VAE’s latent space implicitly contains significant information about system parameters (see Informativeness scores in Table 5.8). This highlights the crucial role of parameter inference for accurate prediction. In this case, latent space supervision acts as a regularizer, enforcing the inference of the parameters. Our results (Table 5.8) suggest that this can lead to disentangled representations. Since disentangled representations are known to improve prediction performance, this provides a strong experimental motivation for our approach (Bengio et al., 2013).

5.4 Methods

5.4.1 Variational Autoencoders SSMs

Variational Autoencoders (VAEs) (Kingma & Welling, 2014) are a type of latent variable model, discussed in Section 2.2.1, that uses neural networks to perform the tasks of inference and generation. VAEs consist of an encoder, $q_\phi(\mathbf{z} \mid \mathbf{x})$, which infers latent representations $\mathbf{z}$ from input data $\mathbf{x}$, and a decoder, $p_\theta(\mathbf{x} \mid \mathbf{z})$, that reconstructs the data from these latent representations. Both encoder and decoder are parameterized using DNNs, and since they are non-linear models, this makes the direct computation of marginal likelihoods infeasible. Therefore, VAEs are trained using approximate inference by maximizing the Evidence Lower Bound (ELBO) on the marginal data likelihood (see Equations (2.16) to (2.17)):

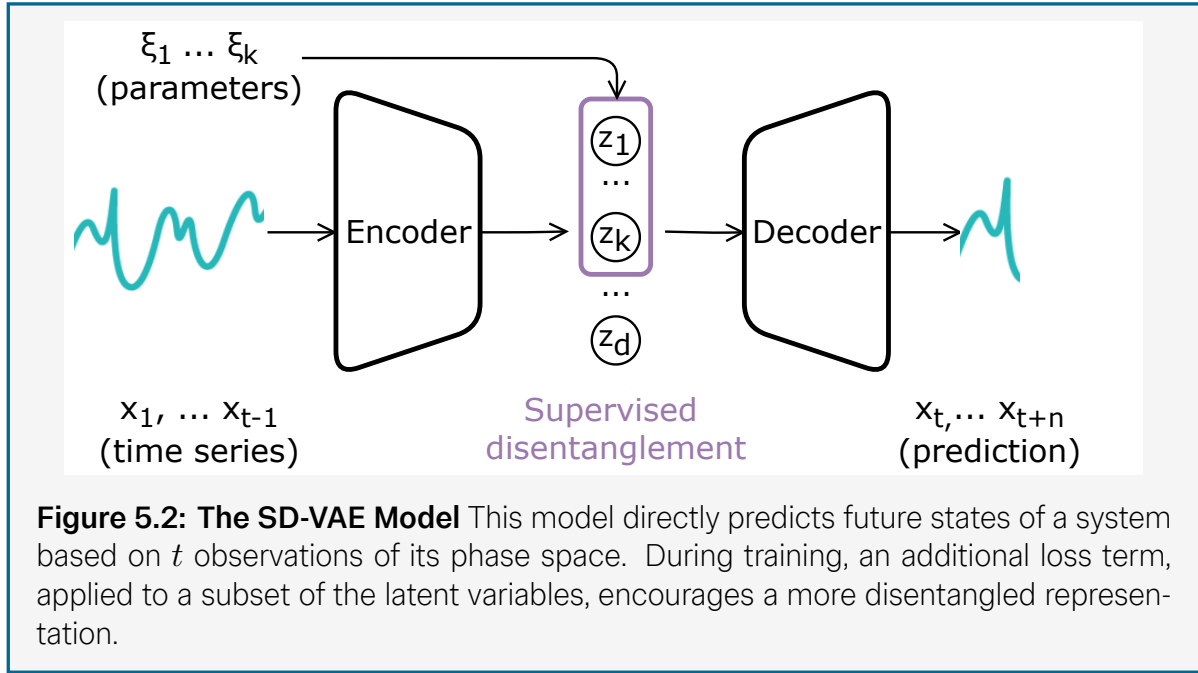
$$\begin{aligned}\mathcal{L}_{\phi, \theta}^{\text{VAE}}(\mathbf{x}) &= \text{ELBO}(\mathbf{x}; \theta, \phi) \\ &= \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})] \\ &= \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{x} \mid \mathbf{z})] - \text{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))\end{aligned}\quad (5.5)$$

In the formulation of Equation (5.5), we can see that the ELBO balances two key components. Firstly, the left term is the reconstruction loss that measures how faithfully the model reconstructs the input data. Secondly, the Kullback-Leibler (KL) divergence quantifies the difference between the approximate posterior distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ and a chosen prior distribution $p(\mathbf{z})$.

The original VAE work by Kingma and Welling, 2014 used an isotropic Gaussian prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$. This choice was later shown to promote unsupervised disentanglement within the learned representation (Higgins et al., 2017). Subsequent research has explored alternative prior distributions as well (Semenova et al., 2022; Tomczak & Welling, 2018). The approximate posterior distribution is also commonly modeled as Gaussian $q_\phi(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$ with a diagonal covariance matrix (Kingma & Welling, 2014). Using Gaussian distributions for both prior and posterior enables a closed-form solution for the KL divergence, making the VAE framework computationally tractable. Additionally, the decoder distribution $p_\theta(\mathbf{x} \mid \mathbf{z})$ can be modeled as Gaussian, which corresponds to using the Mean Square Error (MSE) as the reconstruction loss function.

5.4.2 Forecasting with VAEs

To adapt the VAE framework from reconstruction to prediction, we modify the VAE loss in Equation (5.5) to take time-dependence into account. We replace the approximate posterior



to reflect the time dependency on observations $q_\phi(\mathbf{z} \mid \mathbf{x}_{1:t-1})$ and the reconstruction term with a prediction term, $p_\theta(\mathbf{x}_{t:t+n} \mid \mathbf{z})$, leading to the following training objective:

$$\mathcal{L}_{\phi,\theta}(\mathbf{x}_{1:t+n}) = \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{x}_{t:t+n} \mid \mathbf{z})] - \text{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}_{1:t-1}) \parallel p(\mathbf{z})) \quad (5.6)$$

where t is the time step when prediction starts, and n is the length of the prediction.

Autoregressive Forecasting

While many Deep *SSMs* utilize a distinct latent representation for each time step (Girin et al., 2020; Li & Mandt, 2018), our adapted *VAE* (Equation 5.6) extracts a single latent vector and uses it to perform multi-step prediction. This approach aligns conceptually with *FFNNs*, enabling us to generate arbitrarily long forecasts using an autoregressive strategy (discussed in Section 2.3.3 and depicted in Figure 2.5). This autoregressive technique, where the model's own predictions are fed back as inputs, has proven effective in domains such as wave propagation (Fotiadis et al., 2020) and weather forecasting (Lam et al., 2022).

5.4.3 Disentangling VAEs

Unsupervised vs. Supervised Disentanglement

One of the reasons that we chose *VAEs* as our base model is because they have been extensively studied in the literature in terms of latent space disentanglement (Higgins et al., 2017; Kim & Mnih, 2018; Li et al., 2022; Li & Mandt, 2018; Locatello et al., 2019a; 2020b). *VAEs* can, in principle, achieve disentangled representations without explicit supervision. This can

involve using priors with uncorrelated variables (e.g., standard normal distribution), weighting the KL divergence in the loss function (Higgins et al., 2018), or restricting the latent space dimensionality to match the known factors of variation (Iten et al., 2020). However, as noted by Locatello et al., 2020a, unsupervised disentanglement relies on exploitable biases within the data itself.

When information about the factors of variation in the data is available, supervised disentanglement offers a powerful alternative. This technique has been shown to improve image generation by leveraging ground-truth labels of scene properties (Locatello et al., 2019b).

Supervised Disentanglement in VAEs

We will begin by deriving the general objective for supervised disentanglement in VAEs without assuming time dependency. Later, we will tailor it specifically for dynamical system forecasting. As discussed, VAEs are trained by maximizing the ELBO. To enforce a specific structure on the VAE's latent space, we can employ constrained optimization. Assuming the Karush-Kuhn-Tucker conditions hold, we can express the constrained optimization objective in its Lagrangian form, where constraints transform into regularization terms. The following objective encapsulates the majority of disentanglement methods using this approach (see Tschannen et al., 2018 for a comprehensive review):

$$\begin{aligned} \mathcal{L}_{\phi, \theta}^{\text{D-VAE}}(\mathbf{x}) &= \mathcal{L}_{\phi, \theta}^{\text{VAE}}(\mathbf{x}) \\ &\quad + \lambda_1 \cdot \mathbb{E}_{p(\mathbf{z})} [R_1(q_\phi(\mathbf{z} | \mathbf{x}))] \\ &\quad + \lambda_2 \cdot R_2(q_\phi(\mathbf{z})) \end{aligned} \quad (5.7)$$

Here, R_1, R_2 are regularization terms, dependent on the application, and λ_1, λ_2 are the Lagrange coefficients.

5.4.4 Supervised Disentanglement for Dynamical Systems

Here, we tailor the supervised disentanglement objective of Equation (5.7) specifically for modeling dynamical systems. We employ $R_1 = \text{KL}(q_\phi(\mathbf{z} | \mathbf{x}) || p(\mathbf{z}))$, a standard choice for unsupervised disentanglement originally proposed in beta-VAE (Higgins et al., 2017).

Our main motivation from Section 5.3 suggests that we can treat the ODE parameters as factors of variation within the data. Since our datasets link each trajectory with the ground-truth values of the ODE parameters ξ_C , we use these values in our second regularization term: $\mathcal{L}^{\text{SD}}(\mathbf{z}_{1:k}, \xi_C)$. This supervised regression term depends on both the ground-truth parameters $\xi_C \in \mathbb{R}^k$ and a selected subset of the latent variables, $\mathbf{z}_{1:k}$. Section 5.7 discusses the specific choice of $\mathcal{L}_{\xi_C}$. Finally, we incorporate the time-dependency from

Equation (5.6) to reach the final Supervised Disentanglement VAE (SD-VAE) objective:

$$\begin{aligned} \mathcal{L}_{\phi, \theta}^{\text{SD-VAE}}(\mathbf{x}_{1:t+n}) &= \mathcal{L}_{\phi, \theta}^{\text{VAE}}(\mathbf{x}_{1:t+n}) \\ &\quad - \beta \cdot \text{KL}(q_{\phi}(\mathbf{z} \mid \mathbf{x}_{1:t-1}) \parallel p(\mathbf{z})) \\ &\quad + \delta \cdot \mathcal{L}^{\text{SD}}(\mathbf{z}_{1:k}, \boldsymbol{\xi}_C) \end{aligned} \quad (5.8)$$

The tunable parameters β and δ allow more flexibility in balancing prediction and disentanglement. These parameters respectively weight the KL divergence term and the supervised disentanglement term. From a PGM perspective, weighting the KL term can also be seen as modifying the variance of the decoder. We refer to a VAE model trained with the objective specified in Equation (5.8) as a SD-VAE. Figure 5.2 provides a schematic representation of this architecture.

5.5 Datasets

Here we outline how we build datasets of three dynamical systems for our experiments. We generate these datasets by numerically simulating the ODEs in Equations (5.2) to (5.4). For each system, we start by sampling parameters ξ_C (and initial conditions I_C for the pendulum) from a uniform distribution. This gives us our standard training, validation, and test datasets. To rigorously assess generalization, we then create two OOD datasets (OOD-Easy and OOD-Hard) by sampling parameters from different uniform distributions that don't overlap with the initial distribution. This is visualized for the pendulum length in Figure 5.1, and for all four parameters of the L-V system system in Figure 5.3.

Pendulum simulations involve random sampling of the length l and initial angle θ . Angular velocity and acceleration due to gravity remain constant at $\omega = 0$ and $g = 9.81 \text{ m/s}^2$, respectively. We only vary l because only the g/l ratio is relevant. For each trajectory of the L-V system, we sample the parameters $\alpha, \beta, \gamma, \delta$ independently.

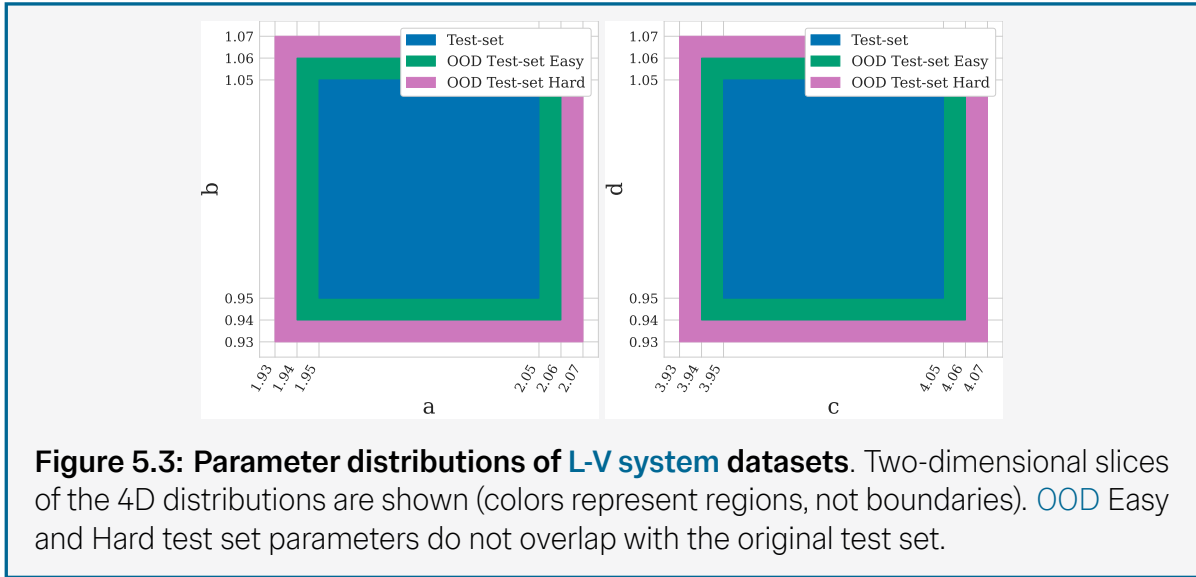
In the case of the 3-body system, we first render the system dimensionless for improved convergence and stability during numerical integration. This also simplifies parameter understanding by providing a reference scale, especially for object masses. We use the Alpha and Proxima Centauri star system for this process. Dimensionless quantities are obtained by dividing by reference values: mass by the solar mass ($m_{nd} = 1.99 \cdot 10^{30} \text{ kg}$), distance by the maximum Alpha Centauri separation ($r_{nd} = 5.33 \cdot 10^{12} \text{ m}$), time by the Alpha Centauri orbital period ($t_{nd} = 1.29 \cdot 10^9 \text{ s}$), and velocity by Earth's solar orbital velocity ($v_{nd} = 30000 \text{ m/s}$). We employ the universal gravitation constant $G = 6.67 \cdot 10^{-11} \text{ N} \cdot \text{m}^2/\text{kg}^2$. Dimensionless constants are calculated as: $K_1 = G \cdot t_{nd} \cdot m_{nd} / (r_{nd}^2 \cdot v_{nd})$ and $K_2 = v_{nd} \cdot t_{nd} / r_{nd}$. Only the ratio between K_1 and K_2 is significant, so we fix K_1 and independently sample K_2 and the three body masses in experiments. The distributions of parameters are selected to avoid pathological trajectories. For the same reason, initial conditions for the L-V system and 3-body systems are held constant, in contrast to the pendulum.

For simulations, we employ the adaptive Dormand-Prince integrator (a member of the Runge-Kutta family) with a 0.01-second time step. Pendulum trajectories span 20 seconds (2000 time steps), while L-V system and 3-body system trajectories have a 10-second duration. We simulate 10,000 trajectories for each system, randomly dividing them into train, validation, and test sets of 8000, 1000, and 1000 samples, respectively. Both OOD Test Set Easy and Hard consist of 1000 trajectories each. To all trajectories, we add a small amount of zero-mean Gaussian noise. Specific details for each system/dataset combination are provided in Table 5.1. The distributions of the pendulum length and the L-V

system parameters can also be seen in Figure 5.1 and Figure 5.3.

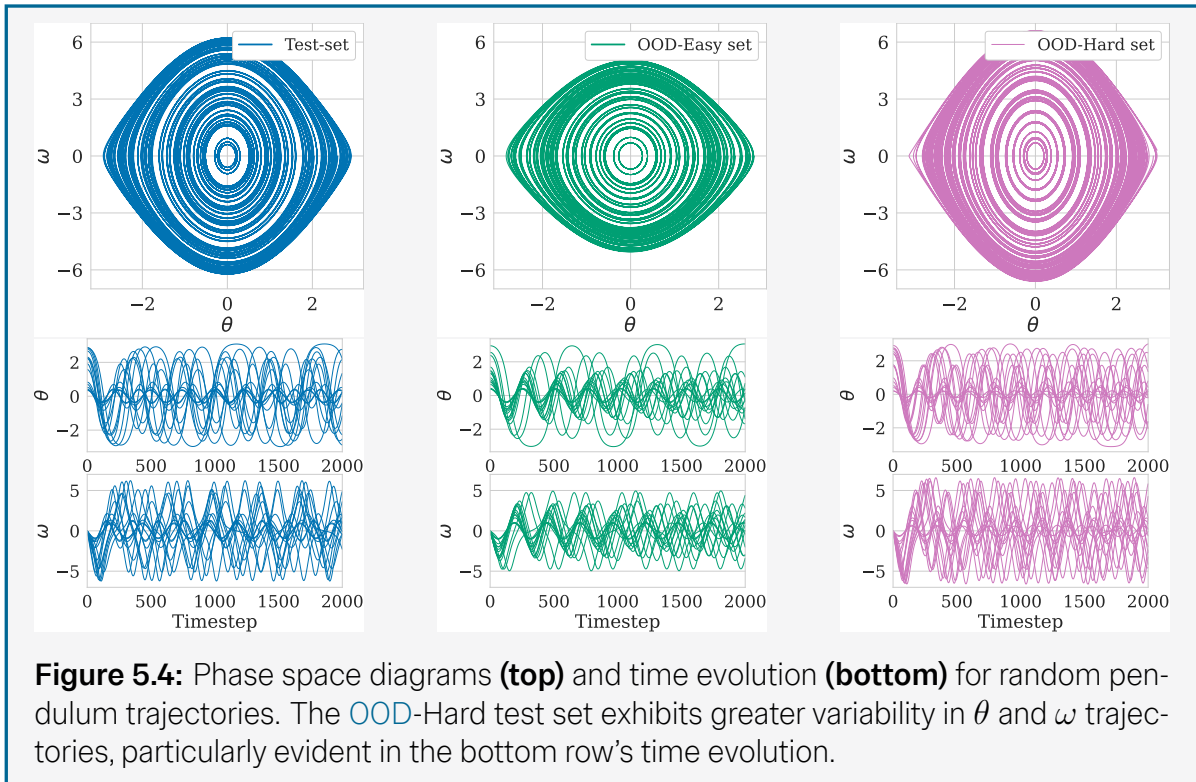
Table 5.1: Datasets. In the L-V system and 3-body OOD test sets, at least one domain parameter is outside of the parameter range used for training.

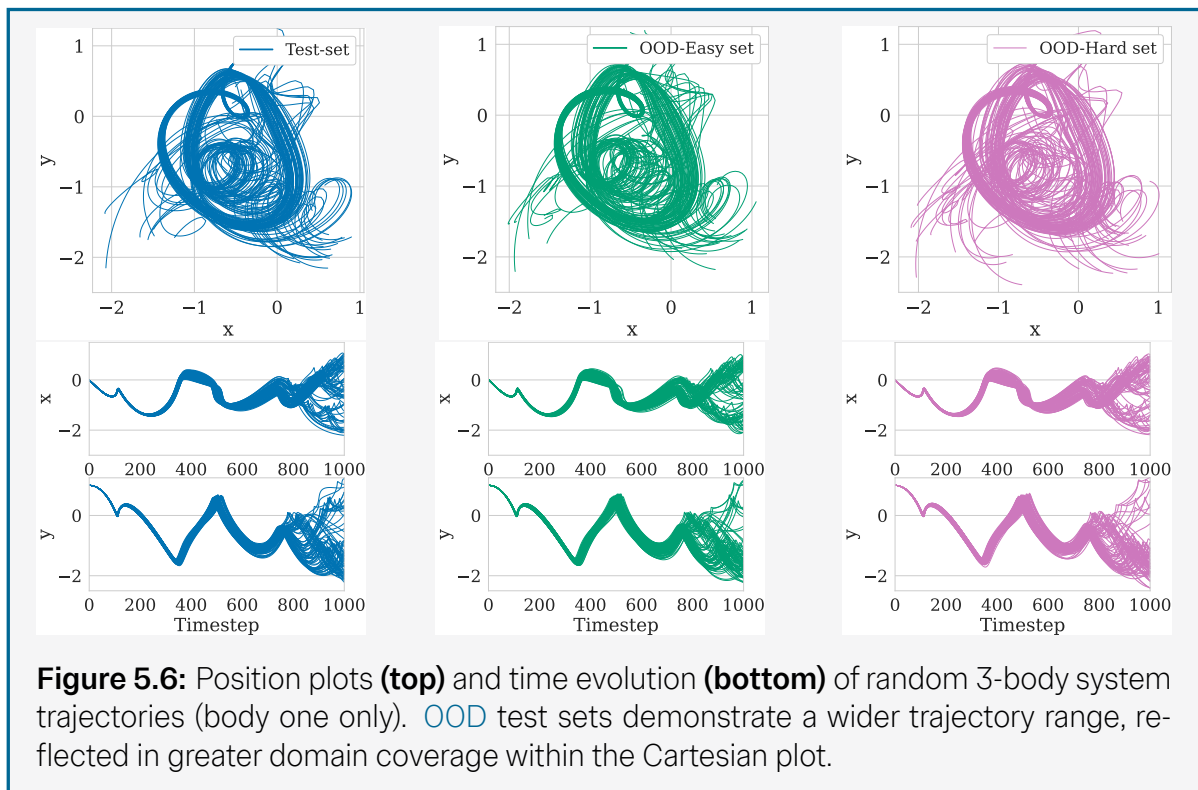
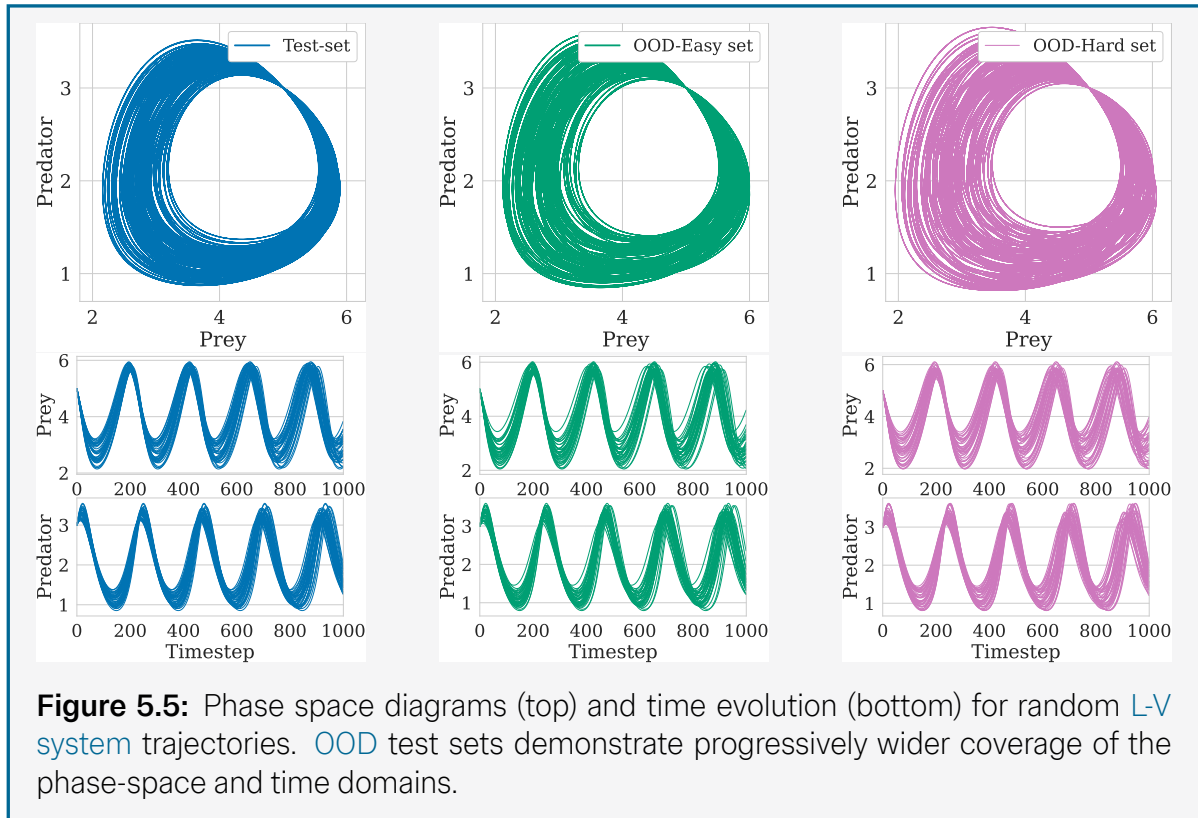
	Pendulum	L-V system	3-body system
ODEs	$\ddot{\theta} + \frac{g}{\ell} \sin \theta = 0$	$\dot{x} = \alpha x - \beta xy$ $\dot{y} = \delta xy - \gamma y$	$\dot{\mathbf{v}}_i = K_1 \sum_{j \neq i} m_j \frac{\mathbf{r}_{ij}}{ \mathbf{r}_{ij} ^3}$ $\dot{\mathbf{x}}_i = K_2 \mathbf{v}_i$ $\mathbf{v}_i, \mathbf{x}_i \in \mathbb{R}^2, i \in [1, 2, 3]$
Number of ODEs	1	2	6
Independent Variables	θ, ω	$x(\text{prey}), y(\text{predator})$	$\mathbf{x}_i, \mathbf{v}_i$
Initial Conditions	$\theta \in [10^\circ - 170^\circ]$ $\omega = 0$	$x = 5, y = 3$	$\mathbf{x}_1 = (-1, -1), \mathbf{v}_1 = (0.0, 0.5)$ $\mathbf{x}_2 = (1, -1), \mathbf{v}_2 = (0.5, -0.5)$ $\mathbf{x}_3 = (0, 1), \mathbf{v}_3 = (-0.5, 0.0)$
Timestep	0.01	0.01	0.01
Sequence length	2000	1000	1000
Noise σ^2	0.05	0.05	0.01
Fixed Parameters	$g = 9.81 \text{ m/s}^2$		$K_1 = 1.0$
Sampled Parameters	$l(\text{length})$	$\alpha, \beta, \gamma, \delta$	m_1, m_2, m_3
		$\alpha \in [1.95, 2.05]$ $\beta \in [0.95, 1.05]$ $\gamma \in [3.95, 4.05]$ $\delta \in [1.95, 2.04]$ $\Omega_{\text{train}} = (\alpha \times \beta \times \gamma \times \delta)$	$K_2 \in [1.95, 2.05]$ $m_1 \in [1.95, 2.05]$ $m_2 \in [1.95, 2.05]$ $m_3 \in [1.95, 2.05]$ $\Omega_{\text{OOD-Hard}} =$ $(K_2 \times m_1 \times m_2 \times m_3)$
Train/Val/Test	$l \in [1.0 - 1.5]$		
		$\alpha \in [1.94, 2.06]$ $\beta \in [0.94, 1.06]$ $\gamma \in [3.94, 4.06]$ $\delta \in [1.94, 2.06]$ $\Omega_{\text{OOD-Easy}} =$ $(\alpha \times \beta \times \gamma \times \delta) \setminus \Omega_{\text{train}}$	$K_2 \in [1.94, 2.06]$ $m_1 \in [1.94, 2.06]$ $m_2 \in [1.94, 2.06]$ $m_3 \in [1.94, 2.06]$ $\Omega_{\text{OOD-Hard}} =$ $(K_2 \times m_1 \times m_2 \times m_3) \setminus \Omega_{\text{train}}$
OOD Test Set Easy	$l \in [1.5 - 1.6]$		
		$\alpha \in [1.93, 2.07]$ $\beta \in [0.93, 1.07]$ $\gamma \in [3.93, 4.07]$ $\delta \in [1.93, 2.07]$ $\Omega_{\text{OOD-Hard}} =$ $(\alpha \times \beta \times \gamma \times \delta) \setminus$ $(\Omega_{\text{train}} \cup \Omega_{\text{OOD-Easy}})$	$K_2 \in [1.93, 2.07]$ $m_1 \in [1.93, 2.07]$ $m_2 \in [1.93, 2.07]$ $m_3 \in [1.93, 2.07]$ $\Omega_{\text{OOD-Hard}} =$ $(K_2 \times m_1 \times m_2 \times m_3) \setminus$ $(\Omega_{\text{train}} \cup \Omega_{\text{OOD-Easy}})$
OOD Test Set Hard	$l \in [0.9 - 1.0]$		
Number of sequences			
Train/Val/Test		8000/1000/1000	
OOD Test Set Easy		1000	
OOD Test Set Hard		1000	



5.5.1 Visualizing the Distribution Shift

Visualizing distribution shifts between datasets can be challenging, particularly for dynamical system trajectories where visual representations are less familiar compared to domains like natural images. To enable qualitative comparisons, we depict phase space diagrams from the three dynamical systems. We provide separate figures for each dynamical system and dataset, highlighting the differences between in and out-of-distribution datasets Figures 5.4 to 5.6. Besides phase space diagrams, we also visualize trajectories across time, offering an alternative perspective to differentiate dynamical characteristics.





5.6 Experiments

5.6.1 Modeling Decisions

VAE and SD-VAE models We use the same model choices for both VAE and SD-VAE. Our prior is an isotropic Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$, which facilitates the disentanglement of the learned representation (Higgins et al., 2017). The approximate posterior, parameterized by the encoder, is a Gaussian with diagonal covariance: $q_\phi(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z} \mid \mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$. This choice allows for a closed-form KL divergence (Kingma & Welling, 2014). The decoder employs a Laplace distribution $p_\theta(\mathbf{x} \mid \mathbf{z}) = \text{Laplace}(\mathbf{x} \mid \boldsymbol{\mu}_x, \mathbf{I})$, equivalent to using an L_1 prediction loss. Preliminary experiments confirmed that the L_1 loss gives better results than L_2 . This aligns with findings in image modeling where L_1 produces sharper results (Mathieu et al., 2019), and its use in time-series forecasting (Tang & Matteson, 2021). The decoder’s covariance remains constant and isotropic, avoiding the need for scaling due to the KL divergence weighting in Equation (5.8).

Scaling of Ground-truth Parameters We select the L_1 loss for the disentanglement regression loss based on preliminary experiments. Since ground-truth parameters have non-zero values, regressing latents on them conflicts with the KL divergence term (which encourages a zero-mean Gaussian). To mitigate this, we scale ground-truth parameters to $[0,1]$ using the equation:

$$\hat{\xi}_C^i = \frac{\xi_C^i - \min(\xi_C^i)}{\max(\xi_C^i) - \min(\xi_C^i)} \quad (5.9)$$

Here, $i \in \{1, 2, \dots, k\}$ indexes domain parameters for system class C . Minimum and maximum values are calculated across the training set. The scaled output $\hat{\xi}_C^i$ is the regression loss target during training. This scaling simplifies the regression task and offers modest improvements. However, it might potentially impact OOD generalization performance.

Other Models We incorporate several additional models in our experiments for comparison. Among these is the LSTM, a widely-used recurrent architecture for modeling low-dimensional sequences (Yu et al., 2019). To evaluate the broader applicability of supervised disentanglement, we extend its use to autoencoders (AEs) with deterministic latent spaces. In total, we train and compare the following models: VAE, SD-VAE, AE, SD-AE, and LSTM.

5.6.2 Training

Early experiments revealed significant performance variance across models, highlighting the importance of careful hyperparameter selection. To ensure fair comparisons, we take several steps. Firstly, we ensure similar network capacities across models. VAE and AE architectures share an encoder with two hidden layers (400 and 200 neurons) and a mirrored decoder. The LSTM employs two stacked LSTM cells with 100 hidden units each, maintaining a comparable parameter count. This leads to models with approximately equal numbers of learned parameters.

We carefully tune our models across a range of hyperparameters, including input/output step counts, hidden layer sizes and counts, latent space size, learning rate, batch size, scheduler patience/reduction factor, and decoder variance. To ensure fair comparisons, we employ grid-search for hyperparameter tuning, running an equal number of experiments per architecture. Detailed hyperparameter choices and experimental counts are provided in Tables 5.3 to 5.5. Our large-scale study involves training a total of 1200 models, requiring over 5,000 CPU-hours Table 5.2.

Typically, our training sequences have a minimum length of 1000 steps. We perform data augmentation by selecting a random starting point t for every batch. We employ an Adam optimizer ($b_1 = 0.9$, $b_2 = 0.999$) with a learning rate scheduler. While the maximum epoch count is set to 2000, we use early stopping based on a validation set, often resulting in fewer epochs.

Table 5.2: Number of experiments with phase space data. Each experiment corresponds to a distinct configuration of hyperparameters. All configurations were run once, all of them with the same seed.

	AE	SD-AE	VAE	SD-VAE	LSTM	Total
Pendulum	72	72	72	72	72	360
L-V	72	72	72	72	72	360
3-body	96	96	96	96	96	480
Total experiments						1200

Table 5.3: Pendulum hyperparameters

	AE	SD-AE	VAE	SD-VAE	LSTM
Input Size			10, 50		
Output Size			1, 10		
Hidden Layers			[400, 200]		50,100,200
Latent Size			4, 8, 16		-
Nonlinearity			Leaky ReLU		Sigmoid
Num. Layers	-	-	-	-	1,2,3
Learning rate			10^{-3}		
Batch size	16, 32	16	16, 32	16	16, 64
Sched. patience	20, 30, 40	20,30	20	20	30
Sched. factor	0.3	0.3	0.3	0.3	0.3
Gradient clipping	No	1.0	1.0		
Layer norm (latent)	No	No	Yes	Yes	No
Teacher Forcing	-	-	-	-	Partial
Decoder γ	-	-	$10^{-3}, 10^{-4}, 10^{-5}$	$10^{-3}, 10^{-4}$	-
Sup. scaling	-	Linear	-	Linear	-
Supervision δ	-	0.1, 0.2, 0.3	-	0.01, 0.1, 0.2	-
# of experiments	72	72	72	72	72

Table 5.4: L-V system hyperparameters

	AE	SD-AE	VAE	SD-VAE	LSTM
Input Size			50		
Output Size			10		
Hidden Layers			[400, 200]		50,100
Latent Size			8, 16, 32		-
Nonlinearity			Leaky ReLU		Sigmoid
Num. Layers	-	-	-	-	1,2,3
Learning rate	$10^{-3}, 10^{-4}$	$10^{-3}, 10^{-4}$	$10^{-3}, 10^{-4}$	10^{-3}	10^{-3}
Batch size	16, 32, 64	16, 32	16, 32	16	10, 64, 128
Sched. patience	20, 30	20, 30	20	20	20, 30
Sched. factor	0.3, 0.4	0.3	0.3	0.3	0.3
Gradient clipping	No	No	0.1, 1.0	0.1, 1.0	No
Layer norm (latent)	No	No	No	No	No
Teacher Forcing	-	-	-	-	Partial, No
Decoder γ	-	-	$10^{-4}, 10^{-5}, 10^{-6}$	$10^{-4}, 10^{-5}, 10^{-6}$	-
Sup. scaling	-	Linear	-	Linear	-
Supervision δ	-	0.1, 0.2, 0.3	-	0.01, 0.1, 0.2, 0.3	-
# of experiments	72	72	72	72	72

Table 5.5: 3-body system hyperparameters

	AE	SD-AE	VAE	SD-VAE	LSTM
Input Size			50		
Output Size			10		
Hidden Layers		[400, 200]			50,100
Latent Size		8, 16, 32			-
Nonlinearity		Leaky ReLU			Sigmoid
Learning rate	$10^{-3}, 10^{-4}$	$10^{-3}, 10^{-4}$	$10^{-3}, 10^{-4}$	10^{-3}	
Batch size	16, 32	16	16	16	16, 64, 128
Sched. patience	30, 40, 50, 60	30, 40, 50, 60	30, 40, 50, 60	30, 40, 50, 60	20, 30
Sched. factor	0.3, 0.4	0.3	0.3, 0.4	0.3, 0.4	0.3
Gradient clipping	No	No	No	No	No
Layer norm (latent)	No	No	No	No	No
Decoder γ	-	-	$10^{-5}, 10^{-6}$	$10^{-5}, 10^{-6}$	-
Sup. scaling	-	Linear	-	Linear, Sigmoid	-
Supervision δ	-	0.05, 0.1, 0.2, 0.3	-	0.1, 0.2	-
# of experiments	96	96	96	96	96

5.7 Results

The primary objective of the experiments in this section is to evaluate whether supervised disentanglement enhances long-term forecasting accuracy. We specifically focus on whether these benefits extend to OOD scenarios. To investigate this, we directly compare VAEs with our proposed SD-VAE method. For comparison, we include the LSTM as a baseline. Additionally, we employ quantitative and qualitative techniques to analyze and compare the latent space representations learned by VAEs and SD-VAEs. Finally, we explore the potential of supervised disentanglement within the framework of deterministic autoencoders (AEs).

5.7.1 Long-Term and OOD Prediction for VAE and SD-VAE

We evaluate the prediction accuracy of VAE and SD-VAE across the three dynamical systems introduced in Section 5.2. For each system, we compare performance on three datasets: the in-distribution test set, OOD-Easy set, and OOD-Hard set, representing increasing degrees of distribution shift from training data. We employ the Mean Absolute Error (MAE) between prediction and ground truth, a standard metric for sequence prediction (Girin et al., 2020) that we also used during training.

Models generate long-term predictions (800 steps) in an autoregressive fashion (Section 2.3.3). This length is considered sufficiently long-term by exceeding the single-forward pass output by a factor of 16-80x (depending on the model's output length). We limit predictions to 800 steps to reserve 200 initial steps of the trajectories for random input starting point selection. To address randomness in training, we select the 5 models with the best average validation MAE and report the mean and standard deviation across the top 5 models for each method.

Results (Table 5.6 and Figure 5.7) demonstrate that SD-VAE consistently outperforms VAE across all systems and datasets. Notably, error reduction exceeds 30% in the pendulum system. Improvements are most pronounced in long-term predictions for both the pendulum and 3-body systems, suggesting that SD-VAE better captures system dynamics. While both models experience absolute performance declines in OOD-Easy and OOD-Hard sets, SD-VAE maintains its advantage (see Figure 5.8). This indicates that disentangling domain parameters may be a beneficial inductive bias, improving in-distribution forecasting without hurting OOD generalization. Overall, SD-VAE achieves better accuracy in long-term and OOD forecasting across all models tested (VAE, AE, SD-AE, and LSTM), demonstrating the value of supervised disentanglement in modeling system dynamics.

Oscillatory dynamics. Interestingly, both the [L-V system](#) and 3-body systems exhibit an oscillatory pattern in the relative [MAE](#) improvement of the [SD-VAE](#) over the [VAE](#). However, the underlying causes appear distinct.

In the [L-V system](#) system, the oscillations are consistently present across all datasets and closely mirror the system's inherent population cycles. These periodic dynamics affect both the prediction difficulty and the relative benefit of disentangled representations over time. The peaks and troughs in relative error align with phases of rapid population change, where latent variable supervision provides varying levels of advantage. On the other hand, the 3-body system shows oscillatory behavior only in the [OOD-Hard](#) setting, where larger parameter shifts occur. We hypothesize that these fluctuations are linked to **transient orbital configurations**: temporary alignments or interactions between the bodies that periodically amplify the system's sensitivity to its governing parameters. During such intervals, disentangled representations capture key latent factors more effectively, resulting in sharper dips in [MAE](#). At other times, the short-term dynamics may dominate prediction difficulty, reducing the relative advantage. These contrasting behaviors are also reflected in the time evolution plots of the datasets (see [Figure 5.5](#) for the [L-V system](#) system and [Figure 5.6](#) for the 3-body system) where the periodicity in the former and the more erratic transitions in the latter become visually apparent.

This suggests that disentanglement is especially helpful in capturing structured, time-varying dependencies, and that its advantages can amplify when the system re-enters previously seen dynamics, even in an [OOD](#) parameter regime. These findings highlight an intriguing interaction between latent space structure and dynamic phase transitions.

Table 5.6: MAE average over 800 steps. Mean performance of the top 5 models of each architecture, selected by validation [MAE](#). [SD-VAE](#) consistently outperforms [VAE](#) and all other models. Note: LSTM diverged during testing on the [L-V system](#).

	Pendulum			L-V system			3-body system		
	Test Set	OOD Easy	OOD Hard	Test Set	OOD Easy	OOD Hard	Test Set	OOD Easy	OOD Hard
LSTM	0.829	1.318	1.985	-	-	-	0.061	0.082	0.099
AE	0.635	1.097	1.420	0.103	0.140	0.157	0.064	0.079	0.093
SD-AE	0.687	1.088	1.442	0.104	0.141	0.157	0.053	0.067	0.084
VAE	0.673	1.128	1.386	0.104	0.142	0.159	0.060	0.075	0.089
SD-VAE	0.443	0.819	1.185	0.100	0.138	0.155	0.048	0.062	0.080

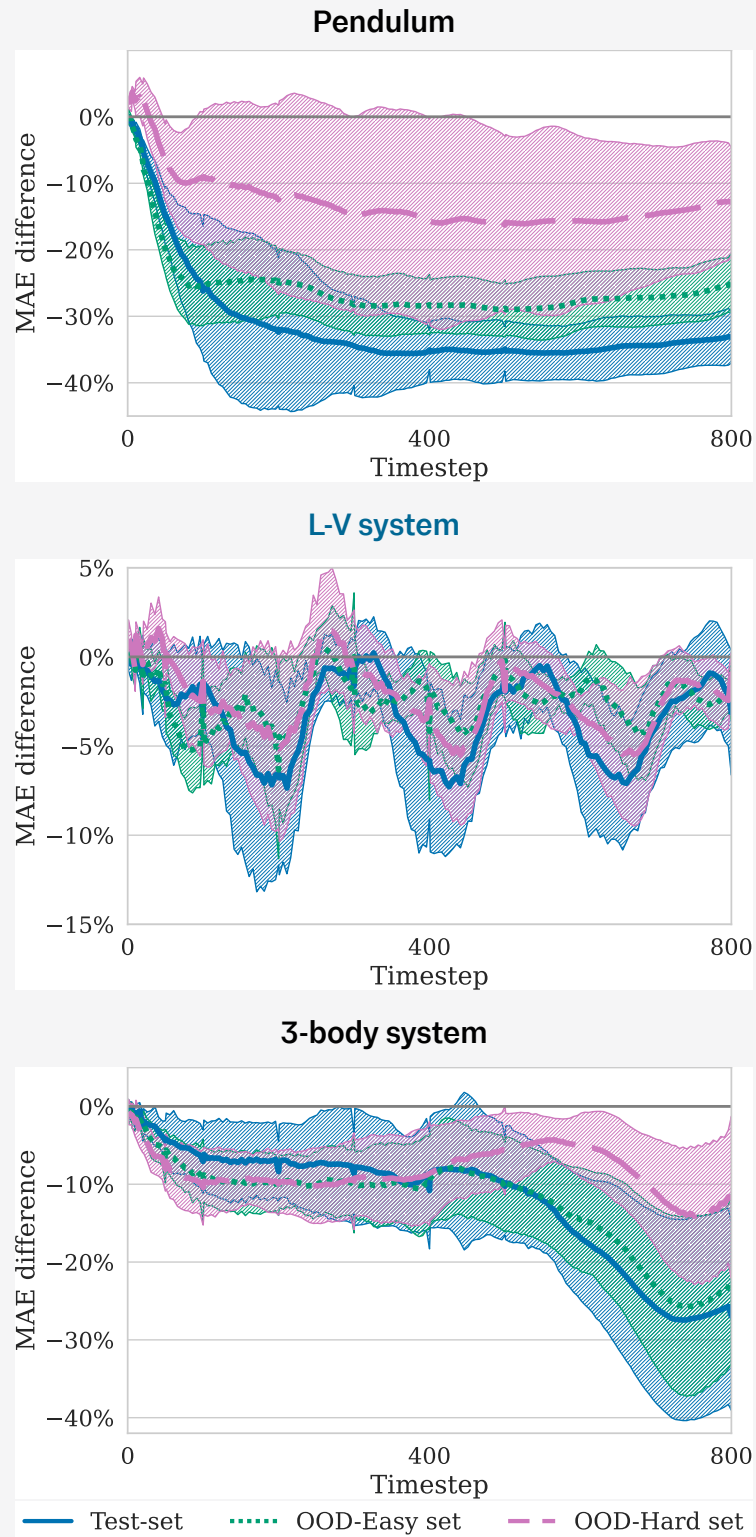
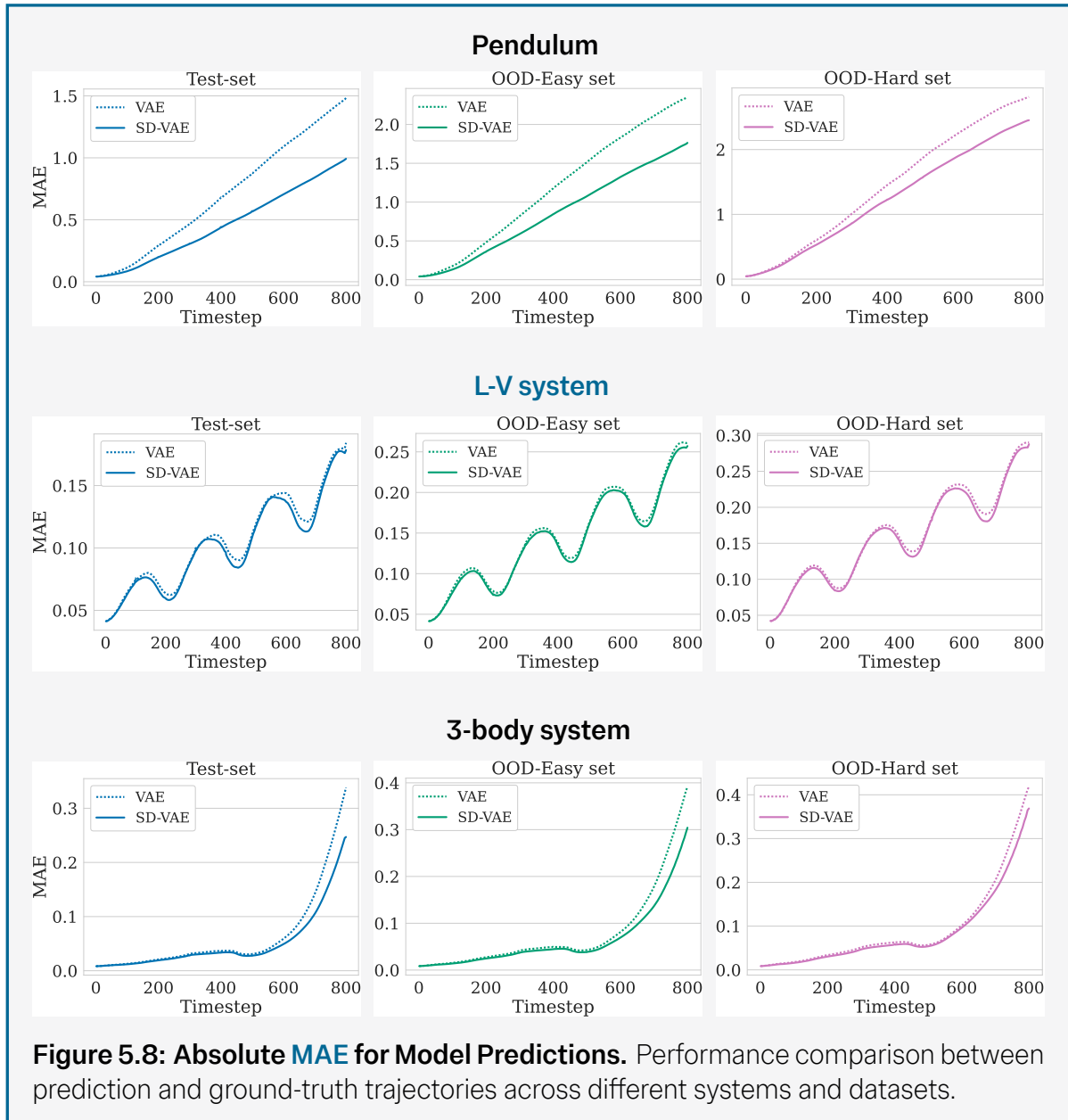


Figure 5.7: Accuracy Improvement with Supervised Disentanglement. Normalized relative MAE between SD-VAE and the plain VAE. Negative values indicate lower SD-VAE error. Results depict the mean and one standard deviation interval across the best 5 models (selected by cumulative validation MAE).



5.7.2 Disentangling AEs

We investigate the applicability of supervised disentanglement to deterministic autoencoders (AEs). Comparisons between AE, SD-AE, VAE, and SD-VAE models (see Table 5.6) reveal that disentangling AEs can lead to accuracy improvements in specific cases, such as the 3-body system. However, improvements appear minimal or absent in the other systems. This suggests that probabilistic models may generally be better suited for capturing data distribution variability. Furthermore, the findings highlight the complexities of latent space disentanglement and motivate future work to deepen our understanding of its practical efficacy.

5.7.3 Implications for Model Stability

Data-driven methods can be susceptible to instability (i.e., outputs diverging to infinity) due to error accumulation. Our experiments demonstrate the severe instability of LSTMs for the [L-V system](#), where none of the 72 trained models could generate long-term (800-step) predictions without divergence. This highlights an important consideration for the potential impact of supervised disentanglement on model stability. In our experiments, [SD-VAE](#) was the most stable model in the pendulum and 3-body systems (>90% stable models) while the [VAE](#) produces more stable models for the [L-V system](#).

5.7.4 Disentanglement of Representations

In this section, our aim is to understand how latent space supervision affects the representations learned by [VAEs](#) and [SD-VAEs](#). Since quantifying disentanglement is challenging, we use a multifaceted approach. Primarily, we employ the [Disentanglement - Completeness - Informativeness \(DCI\)](#) framework, which assesses disentanglement along three distinct axes ([Eastwood & Williams, 2018](#)). This framework takes advantage of the ground-truth labels. Briefly, Disentanglement measures how well factors of variation are separated in the representation, Completeness measures whether each factor is captured by a single latent variable, and Informativeness quantifies how much information about the factors of variation the representation holds (irrespective if it's disentangled or not). Recent studies support that random forests-based [DCI](#) is a robust all-around metric ([Zaidi et al., 2020](#)). We also compute the [Attribute Predictability Score \(SAP\)](#) ([Kumar et al., 2018](#)), which leverages learned predictor importance and mutual information. Lastly, we compute the [Mutual Information Gap \(MIG\)](#) ([Chen et al., 2018a](#)) which we adapt to the continuous setting. [MIG](#) is based on information-theoretic principles and does not require ground-truth labels.

Quantitative Results Disentanglement metrics in ([Table 5.8](#)) demonstrate that [SD-VAE](#) produces significantly more disentangled representations than [VAE](#) across all three systems. We see important improvement across Disentanglement, Completeness, [SAP](#), and [MIG](#) met-

Table 5.7: Model Stability. Percentage of models whose predictions diverge (to positive or negative infinity) during testing across all trajectories (out of all trained models).

	Pendulum	L-V system	3-body system
LSTM	86%	100%	53%
AE	42%	14%	50%
SD-AE	69%	29%	58%
VAE	3%	10%	15%
SD-VAE	2%	48%	9%

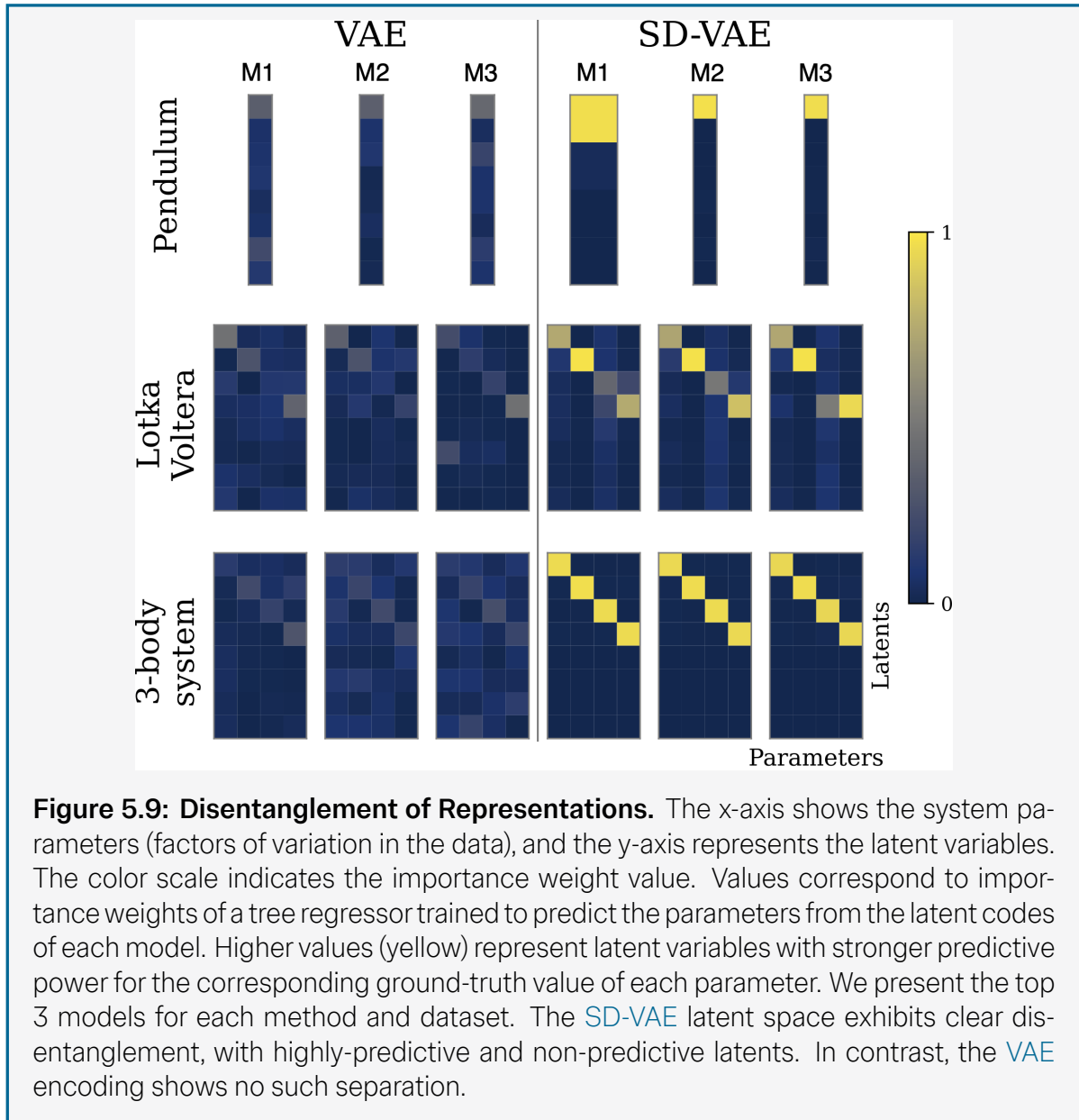
rics. Both VAEs achieve near-optimal Informativeness, implying that VAE representations contain the necessary parameter information, although it is dispersed across latent dimensions. Note that Disentanglement isn't calculated for the pendulum as it has a single factor of variation (length).

Qualitative Comparison To compute DCI metrics, we trained a set of gradient-boosted decision tree regressors, each tasked with predicting a single ground-truth system parameter from the latent representation $\mathbf{z}$. For each parameter $\xi_i \in \xi_C$, a separate regressor is fit using the training set. We employ the XGBoost library with default hyperparameters, chosen for its strong performance and robustness in low-data, high-variance settings. After training, we extract feature importance scores, which quantify the contribution of each latent dimension to the prediction of each parameter. These scores form the basis for calculating the Disentanglement and Completeness components of the DCI framework and also offer an interpretable view of how parameter information is distributed across the latent space.

In Figure 5.9, we visualize the importance scores for the top three VAE and SD-VAE models, truncated to the first 8 latent dimensions for clarity. Color intensity reflects the relative predictive contribution of each latent variable. For a full view including all latent dimensions, as well as a Pearson correlation-based comparison, see Figure 5.10 (b). To aid interpretability, we sort the rows in the VAE plots such that the latent with the highest weight for the i -th parameter appears in the i -th row. These visualizations reveal that SD-VAEs tend to assign most of the predictive responsibility to a small subset of latent variables, while the VAEs distribute information more diffusely across the latent space. Together with the quantitative disentanglement metrics, these patterns underscore the effectiveness of latent supervision in producing more interpretable and disentangled representations.

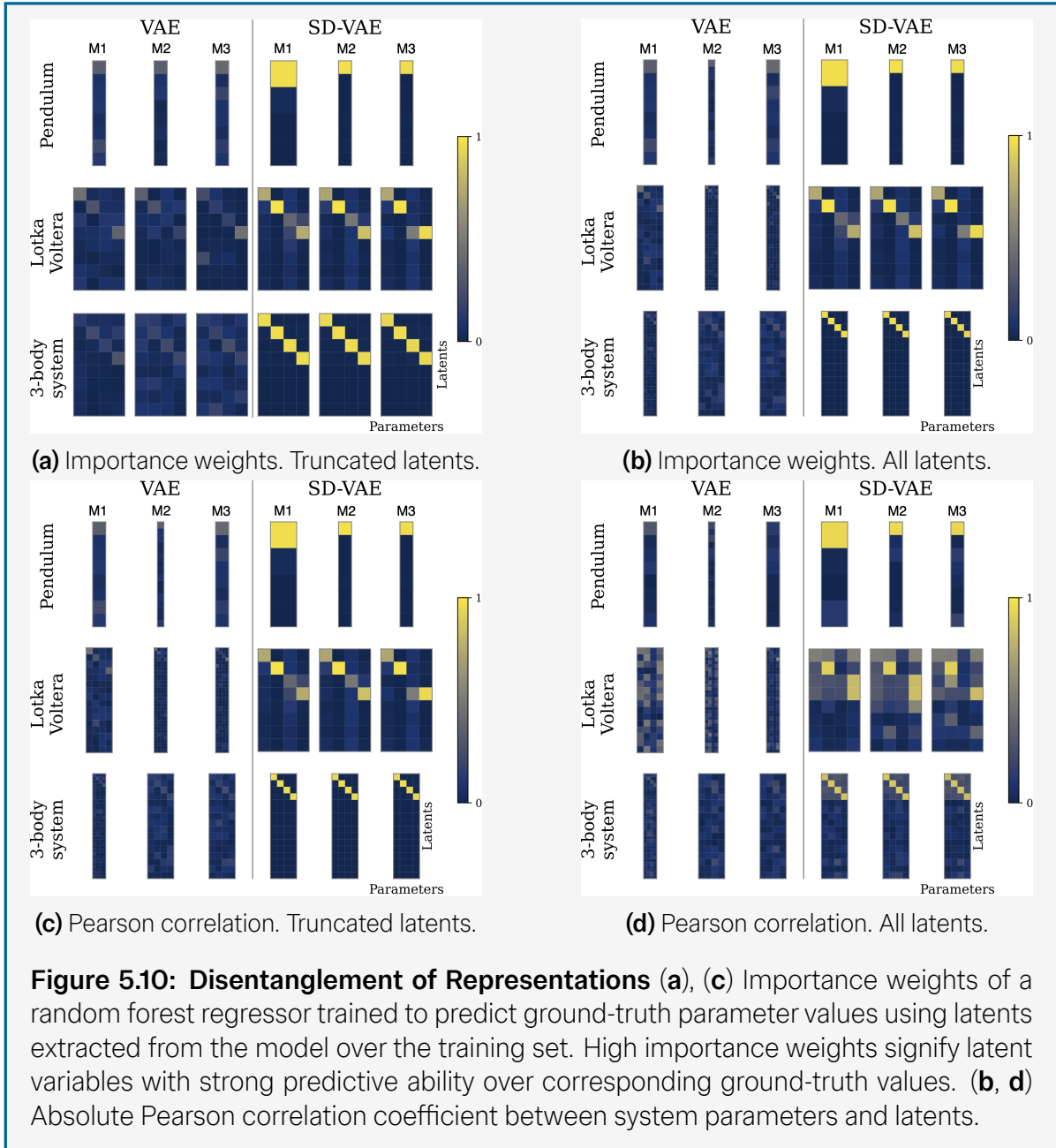
Table 5.8: Disentanglement Metrics. SD-VAE consistently outperforms VAE across several disentanglement metrics. Scores represent averages over the top 3 models (selected by validation MAE).

	Pendulum		L-V system		3-body system	
	VAE	SD-VAE	VAE	SD-VAE	VAE	SD-VAE
Disentanglement	-	-	0.27	0.53	0.20	0.90
Completeness	0.17	0.90	0.20	0.57	0.13	0.90
Informativeness	0.94	0.99	1.00	1.00	1.00	1.00
SAP	0.03	0.87	0.04	0.21	0.01	0.67
MIG	0.01	0.17	0.00	0.03	0.00	0.08



5.7.5 Linear Correlation between Latents and Parameters

The importance weights analyzed previously (Section 5.7.4, Figures 5.9 to 5.10) indicate a substantial correlation between latents ($\mathbf{z}$) and parameters (ξ_C). Understanding the nature of this relationship could enhance our understanding and could enable us to manipulate the latent space. Since tree-based models capture both linear and non-linear dependencies, we need further experiments to determine the precise nature of this relationship. To quantify linearity, we fit a linear regression models for each $\mathbf{z}_i, \xi_{C_j}$ pair and compute the absolute Pearson correlation coefficient (r) between them. The Pearson's r is chosen here due to its ability to measure the linear relationship between the latent space variables and the ground-truth parameters. An absolute value of r close to 1 denotes a highly linear correlation. Visualization of Pearson's r values in Figure 5.10 (b, d) confirms a predominantly linear relationship



between supervised latents and parameters. The absolute values of r are:

Pendulum: $r_l = 0.94$

L-V system: $r_a = 0.52, r_b = 0.91, r_c = 0.23, r_d = 0.82$

3-body system: $r_K = 0.84, r_{m_1} = 0.87, r_{m_2} = 0.87, r_{m_3} = 0.85$

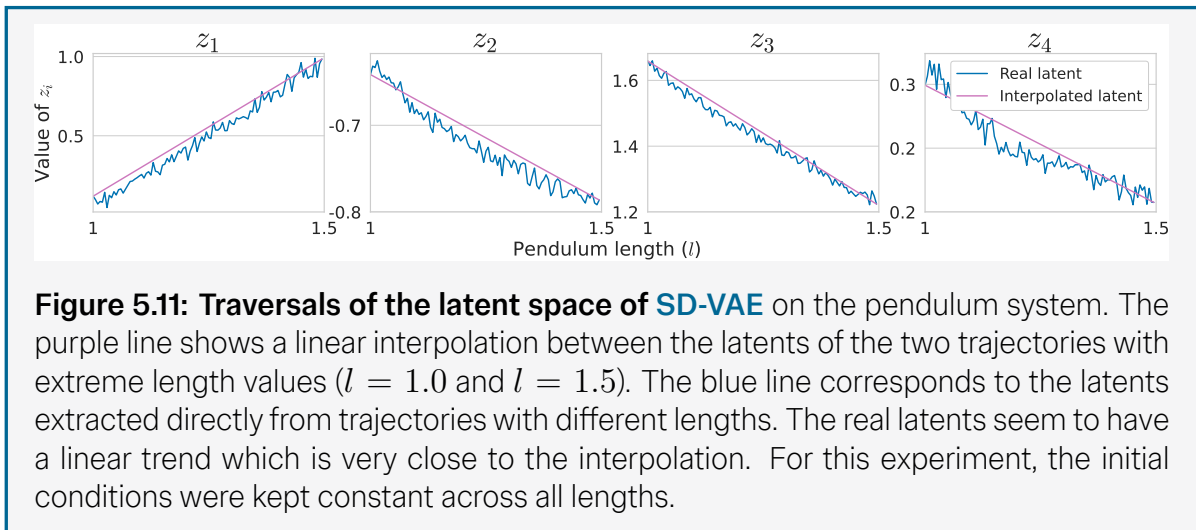
These results demonstrate a predominantly linear relationship between supervised latents and parameters, further strengthening our conclusions about the disentanglement achieved by SD-VAE. This linearity aligns with our experimental findings that linear scaling gives the best performance for the disentanglement loss (see Section 5.6.1). We leverage this linearity for latent space traversals in the following Section 5.7.6.

5.7.6 Latent Space Traversals

Being able to traverse between two points in the latent space suggests that meaningful representations have been learned in mapping from data to latent space. Interpolation in latent space can produce meaningful images in properly disentangled VAEs (Higgins et al., 2017). While images have easily recognizable visual components, similar traversals for dynamical systems are harder to portray.

Here we study whether interpolating between two points in the latent space of SD-VAE can produce meaningful trajectories. For this, we create a new pendulum dataset containing 100 trajectories with linearly spaced length in the range $l \in [1.0 - 1.5]$. The initial conditions are kept constant ($\theta = \frac{\pi}{2}, \omega = 0$) for all the trajectories to facilitate comparisons and we use the same noise level as in the training dataset. We use the encoder of our best SD-VAE model to extract the latent variables for each trajectory. For each trajectory, the encoder produces 4 latent variables $z_1 \dots z_4$. We interpolate between the latents of the two extreme trajectories ($l = 1.0$ and $l = 1.5$). The interpolation we use is linear, driven by our findings that latents and parameters have a strongly linear correlation (Section 5.7.5).

Next, we feed the real and interpolated latents to the decoder and predict up to 1000 time steps. We find that MAE is 0.29 when the real latents are used (i.e., extracted with the encoder) and 0.33 with the interpolated ones. The error is very close, indicating that linear latent space interpolation produces meaningful latent codes. This is further corroborated by plotting the real and interpolated latents together in Figure 2.5. We can see that the relationship between the real latents z_i and pendulum length l is highly linear, which further explains why the linear interpolation method works well.



5.8 Concluding remarks

In this chapter, we explored a novel supervised disentanglement methodology for [VAEs](#) and its impact on forecasting dynamical systems. Our findings demonstrate significant improvements in model robustness and forecasting accuracy, particularly in [OOD](#) scenarios.

Firstly, our experiments demonstrated that supervised disentanglement in [SD-VAEs](#) enables more stable and robust long-term predictions compared to traditional [VAEs](#) and other baseline models like [LSTMs](#). [SD-VAEs](#) consistently outperformed other architectures across various dynamical systems, achieving error reduction and enhanced performance stability in both in-distribution and [OOD](#) settings. Our approach is particularly valuable in applications where reliable predictions over long time horizons or under novel conditions are crucial. It also offers practical advantages in real-world applications where the dynamics of the system are often unknown, but some knowledge of the system's parameters is available. By incorporating this knowledge into the disentanglement process, our approach efficiently utilizes available data, leading to improved model adaptability and interpretability.

Additionally, the quantitative assessments using disentanglement metrics like [DCI](#), [SAP](#), and [MIG](#) show that [SD-VAEs](#) provide a more structured and interpretable latent space than [VAEs](#). This structure not only helps with model performance and robustness but also facilitates a deeper understanding of the underlying dynamics of the modeled systems.

Moreover, the practical implementation of our [SD-VAEs](#) models, supported by publicly available code and datasets, offers a valuable resource for further research and application in the field. It allows for reproducibility and iterative improvement, essential for the ongoing advancement of robust forecasting in dynamical systems.

In summary, integrating supervised disentanglement into [VAEs](#) presents a promising path for enhancing the accuracy and reliability of data-driven models in forecasting dynamical systems. Future research directions include extension to the semi-supervised or unsupervised setting, aiming to develop methods capable of leveraging minimal or no supervision for improved predictive performance. Additionally, assessing the method in systems with more complex dynamics is important for establishing its real-world applicability. In [Chapter 6](#), we build upon the findings of this chapter and explore the utility of our method in high-dimensional systems where an observation model is also present.

Chapter 6

Disentanglement for spatiotemporal forecasting

This chapter investigates the role of supervised disentanglement in enhancing generative models for capturing system dynamics from spatiotemporal data. We introduce disentanglement loss to CNN-based models to independently encode key physical parameters. Experiments on wave propagation demonstrate that the proposed training outperforms the baselines in long-term prediction accuracy and robustness to OOD shifts. Additional experiments with pendulum motion elucidate the tradeoffs of disentanglement. Furthermore, supervised disentanglement enhances the interpretability and controllability of latent representations. Overall, our findings highlight the benefits and tradeoffs of supervised disentanglement for improving the generalization and interpretability of convolutional DNNs in dynamical system modeling from spatiotemporal data.

6.1 Introduction

Predicting the behavior of complex dynamical systems from high-dimensional spatiotemporal data is a significant challenge in machine learning and engineering. While deep generative models have shown promise in modeling such systems, their ability to generalize to long-term predictions and OOD scenarios is often limited, especially when dealing with high-dimensional inputs.

In [Chapter 4](#), we demonstrated that supervised disentanglement can enhance the robustness and accuracy of deep generative models in forecasting dynamical systems with low-dimensional states (up to $\mathbb{R}^4$). By incorporating latent-space supervision to produce disentangled representations, we observed improvements in long-term prediction accuracy and generalization to OOD data using fully connected DNNs.

Building upon this foundation, we address the more challenging problem of modeling dynamical systems from high-dimensional spatiotemporal observations, where the system's state $\mathbf{u}_t$ is only available through high-dimensional renderings on a uniform grid $\mathbf{f}_t \in \mathbb{R}^{h \times w}$ at each time step t . This setting introduces additional complexity due to both the higher dimensionality and the presence of an observation model, rendering fully connected networks computationally infeasible.

To tackle this, we adapt our supervised disentanglement approach to convolutional neural network (CNN)-based models, which are better suited for handling high-dimensional, grid-structured inputs. Specifically, we integrate supervised disentanglement into two different CNN-based architectures: the Recurrent State Space Model (RSSM) ([Hafner et al., 2019](#)), applied to image sequences of a swinging pendulum, and a U-Net ([Ronneberger et al., 2015](#)), trained to predict wave propagation dynamics in shallow water. By applying our method to these models, we aim to evaluate its effectiveness across different systems with varying dynamics and observation models, as well as different neural network architectures.

Through experiments on pendulum motion and wave propagation datasets, we examine the impact of supervised disentanglement on modeling dynamical systems from high-dimensional observations. Our results show that incorporating supervised disentanglement can improve long-term prediction accuracy and robustness to OOD shifts in certain cases. However, we also observe trade-offs, such as potential decreases in short-term prediction accuracy in some scenarios, indicating that the benefits of disentanglement may vary depending on the specific system and model architecture.

6.1.1 Main Contributions

The main contributions of this chapter are as follows:

Extending Supervised Disentanglement to High-Dimensional Observations: We extend the concept of supervised disentanglement to dynamical systems where the state is observed through high-dimensional spatiotemporal data, addressing the challenges associated with higher dimensionality and varied observation models.

Adapting the Approach to CNN-Based Models: We adapt our supervised disentanglement method to convolutional neural network architectures, specifically integrating it into the Recurrent State Space Model (RSSM) and U-Net, to handle high-dimensional inputs and evaluate its effectiveness across different architectures.

Empirical Evaluation on Two Dynamical Systems: We conduct experiments on pendulum motion and wave propagation, analyzing the effects of supervised disentanglement on long-term prediction accuracy and robustness to distributional shifts, and highlighting both the benefits and trade-offs observed.

These contributions advance the understanding of how supervised disentanglement can influence the modeling of dynamical systems from high-dimensional observations. By exploring its applicability across different systems and architectures, our work provides insights into the potential and limitations of this approach in improving the performance and interpretability of models in control, engineering, and physical sciences.

We present the experiments in two sections, first for the pendulum and then for the wave propagation.

6.2 Pendulum Experiments

6.2.1 Dataset

In this experiment, we use the swinging pendulum as our dynamical system—a common benchmark for modeling dynamics in image sequences (Barber et al., 2021; Brunton et al., 2020). The dataset consists of sequences of images depicting a moving pendulum. The pendulum’s positions are computed via numerical simulation using an adaptive Runge-Kutta integrator with a timestep of 0.05 seconds, and then rendered as 64×64 pixel frames.

For each trajectory, the pendulum length ℓ , gravity strength g , and initial conditions—initial angle θ and angular velocity ω —are set to different values drawn from uniform distributions, causing each trajectory to differ slightly. Specifically, the initial angle ranges from 30° to 170° and the initial velocity ranges from -2 to 2 rad/s.

For the training, validation, and in-distribution testing sets, gravity g ranges from 8.0 to 12.0 m/s^2 , and pendulum length ℓ ranges from 1.20 to 1.40 m. We create 10,000 sequences split in 8,000/1,000/1,000 between training, validation and test sets. In the out-of-distribution (OOD) testing sets, we alter the distributions of ℓ and g while keeping the same distributions for θ and ω as in training. In the easy OOD testing set, gravity ranges from 12.0 to 12.5 m/s^2 and pendulum length from 1.40 to 1.45 m. We call this dataset “**easy**” because due to higher length the angular velocity is lower, resulting to dynamics that are easier for the model to capture. Same naming logic applies for the OOD Hard test set, where gravity ranges from 12.5 to 13.0 m/s^2 and pendulum length from 1.45 to 1.50 m. Each OOD set comprises of 1,000 sequences. More details about the dataset are provided in Section 6.2.1, and parameter distributions are illustrated in Figure 6.1.

6.2.2 Method

We showcase our method by extending the latent-space supervision from Chapter 5 to a state-of-the-art generative model called Recurrent State Space Model (RSSM) (Hafner et al., 2019). RSSM has been successfully used for planning from pixels and is considered state-of-the-art model in long-term spatiotemporal prediction (Saxena et al., 2021). Furthermore, RSSM is a hybrid model combining deterministic and stochastic components, and this allows us to assess disentanglement outside VAEs. We use the same formulation of the loss function as in the original implementation of Hafner et al., 2019, with the addition of the supervised disentanglement loss, similarly to what we do in Equation (5.8). RSSM includes both stochastic and deterministic latents. We use supervised disentanglement on

the stochastic part, and term that model SD-RSSM. Since the model has latent variables for each time-step, we apply a disentanglement loss on all of them. The loss function for SD-RSSM is:

$$\begin{aligned} \mathcal{L}_{SD-RSSM}(\mathbf{x}_{1:t}) = \sum_{t=1}^T & \left(\underbrace{\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:t-1})} [\ln p(\mathbf{x}_t | \mathbf{z}_t)]}_{\text{reconstruction}} \right. \\ & - \underbrace{\mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1})} [\text{KL} [q(\mathbf{z}_t | \mathbf{x}_{1:t-1}) || p(\mathbf{z}_t | \mathbf{z}_{t-1})]]}_{\text{prediction}} \\ & \left. + \underbrace{\delta \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:t-1})} [\|\boldsymbol{\xi} - \mathbf{z}_t^{1:k}\|_2]}_{\text{supervised disentanglement loss}} \right) \end{aligned} \quad (6.1)$$

where $\mathbf{x}_t$ is the observations, $\mathbf{z}_t$ the stochastic latent variables at time t , $\boldsymbol{\xi}$ are the k dimensional domain parameters and δ tunes the supervised disentanglement strength. Next, we train SD-RSSM and train in on on image sequences of the swinging pendulum dataset.

6.2.3 Experiments

The RSSM model we use follows the architecture as described in (Hafner et al., 2019) & (Saxena et al., 2021). Disentanglement is applied to all four parameters (length ℓ , gravity g , initial position θ and velocity ω), but only length and gravity vary between datasets as explained in Section 6.2.1. We use an L_2 loss for the disentanglement term because preliminary results showed it performs better than L_1 and BCE. All models were trained for 300 epochs with a learning rate of 10^{-3} and an Adam optimizer ($b_1 = 0.9$ and $b_2 = 0.999$). We tune the length of the input frames, the batch size and the decoder standard deviation. For SD-RSSM we also tune the supervision strength $\delta = 0.01, 0.1, 1$. To make comparisons fair for RSSM we use 3 seeds to achieve the same number of experiments (24) for both models (detailed hyperparameters in Table 6.1). During testing the model uses 50 frames as context.

For evaluating the models we use Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and RMSE following our analysis in Chapter 4. **SSIM** measures structural similarity, accounting for luminance, contrast, and structural information. Values range from -1 to 1 , with higher values indicating greater similarity. Evaluates the ratio between the maximum possible signal power and the power of corrupting noise. Higher values signify better image quality. **RMSE** quantifies the average magnitude of prediction errors. Lower values denote more accurate reconstructions. These metrics collectively provide a comprehensive evaluation of image quality, capturing both perceptual similarity and pixel-level accuracy.

Table 6.1: Hyperparameters for the RSSM and SD-RSSM models. The table lists the hyperparameter values used in our experiments. For SD-RSSM, the supervision coefficient δ is tuned between 0.01 , 0.1 , and 1 . For RSSM, we use 3 random seeds to achieve the same number of experiments (24) for a fair comparison.

Batch Size	50, 100
Decoder std dev	1.0, 2.0
Input Length	50, 100

6.2.4 Evaluation Metrics

To assess the quality of image predictions and reconstructions, we employ two widely recognized evaluation metrics: **SSIM** (Wang et al., 2004) and Peak Signal-to-Noise Ratio (PSNR).

SSIM

SSIM is a perceptual metric that quantifies image quality degradation caused by processing such as data compression or transmission losses. Unlike traditional metrics like MSE or PSNR, **SSIM** considers changes in structural information, luminance, and contrast, which are more aligned with human visual perception (Golestani & Ghanbari, 2014).

The **SSIM** between two images x and y is defined as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (6.2)$$

where:

- μ_x and μ_y are the average pixel values of images x and y , respectively.
- σ_x^2 and σ_y^2 are the variances of x and y .
- σ_{xy} is the covariance of x and y .
- $C_1 = (k_1L)^2$ and $C_2 = (k_2L)^2$ are stabilization constants to avoid division by zero, with L being the dynamic range of the pixel values and k_1, k_2 small constants.

SSIM values range from -1 to 1 , where 1 indicates perfect similarity between the two images.

Peak Signal-to-Noise Ratio (PSNR)

PSNR is a metric used to measure the quality of reconstruction of lossy compression codecs. It is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.

The PSNR is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\max(I)^2}{\text{MSE}} \right) \quad (6.3)$$

where:

- $\max(I)$ is the maximum possible pixel value of the image (e.g., 255 for an 8-bit image).
- MSE is the Mean Squared Error between the original and the reconstructed image, calculated as:

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n [I(i, j) - K(i, j)]^2 \quad (6.4)$$

where m and n are the dimensions of the images, and $I(i, j)$ and $K(i, j)$ are the pixel values at position (i, j) in the original and reconstructed images, respectively.

Higher PSNR values generally indicate better reconstruction quality, with typical values ranging from 20 to 50 dB for various applications.

6.2.5 Results

We evaluate the performance of the standard Recurrent State-Space Model (RSSM) and our proposed Supervised Disentangled RSSM (SD-RSSM) on the pendulum dataset, focusing on both in-distribution and out-of-distribution (OOD) scenarios. For quantitative evaluation, we use the SSIM and PSNR, which are widely used for assessing image quality in dynamical system prediction from spatiotemporal data (Pant & Farimani, 2020; Wang et al., 2004). Unlike pixel-wise metrics such as MSE and MAE, SSIM considers qualitative characteristics of the images, making it more suitable for our analysis.

We select the best models based on the average SSIM over the validation set and plot SSIM and PSNR as functions of the timestep (Figures 6.3 and 6.4). As shown in Figure 6.3a, in the in-distribution test set, RSSM outperforms SD-RSSM in maintaining structural similarity during short-term predictions. However, beyond 400 timesteps, SD-RSSM excels in preserving structural integrity over long-term dynamics. This pattern is consistent across all three datasets.

In the OOD Easy test set, which features a longer pendulum length and slower angular velocity, SD-RSSM demonstrates superior structural similarity in capturing long-term dynam-

ics starting around 350 timesteps, outperforming RSSM (Figure 6.3b). In the more challenging OOD Hard test set, with a shorter pendulum length and higher angular velocity, SD-RSSM maintains higher structural similarity in long-term predictions after approximately 200 timesteps, outperforming RSSM under significant distributional shifts (Figure 6.5c).

We observe similar trends when comparing PSNR values (Figure 6.4). While RSSM performs better in short-term predictions, SD-RSSM consistently outperforms RSSM in capturing long-term dynamics, especially under increasing distribution shifts. Quantitative metrics at the 800th timestep are summarized in Table 6.2, where SD-RSSM achieves higher SSIM and PSNR values across all test sets.

The crossover in performance observed around 300-400 timesteps, where SD-RSSM begins to outperform RSSM in long-term predictions, can be attributed to a combination of factors. In the initial prediction phase (short-term), the standard RSSM might leverage its full latent capacity to model immediate pixel-level transitions and very short-term dynamics effectively, potentially giving it a slight edge. However, as predictions are made autoregressively, any minor inaccuracies or entangled representations of the underlying physical parameters (such as pendulum length l , gravity g , initial angle θ , and initial velocity ω) within the standard RSSM can begin to accumulate. This error accumulation often becomes more pronounced over longer horizons. Conversely, the SD-RSSM is explicitly trained to disentangle these physical parameters within its latent space. While this explicit supervision acts as a regularizer and might initially constrain the model slightly in the very short term (as it balances reconstruction with parameter regression), it leads to a more robust and physically grounded internal representation of the system's governing factors. After a certain number of timesteps (around 300-400 in this case), the cumulative errors in the standard RSSM, stemming from less structured representations of these parameters, likely surpass any initial short-term predictive advantage. At this point, the SD-RSSM's more stable and accurate encoding of the fundamental physical parameters allows it to maintain a more coherent and accurate trajectory over extended predictions, better capturing the true long-term dynamics of the system.

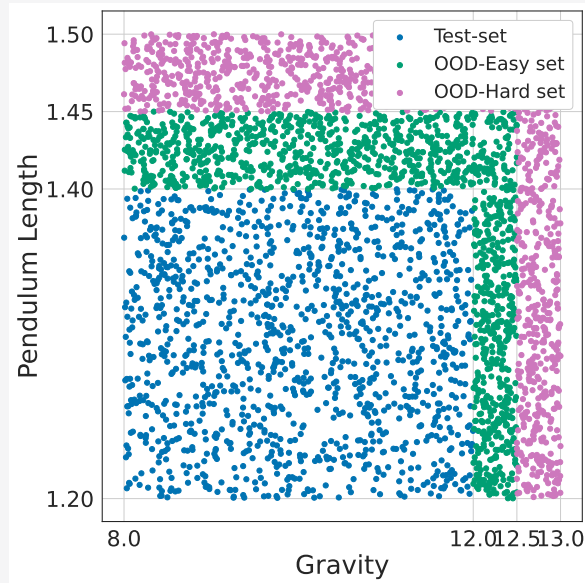
Qualitative results further support these findings. As illustrated in Figure 6.5, both models produce accurate short-term predictions and capture the appearance of the pendulum in long-term predictions. However, SD-RSSM better captures the longer-term dynamics, maintaining closer alignment with the ground truth trajectory. Specifically, the difference between ground truth (gray) and predictions (white) highlights how SD-RSSM more effectively preserves the pendulum's motion over time, even under significant distribution shifts in the OOD Hard test set.

We hypothesize that the improved long-term performance of SD-RSSM is due to its ability to disentangle latent spaces, reducing overfitting to the relatively short training horizon. This

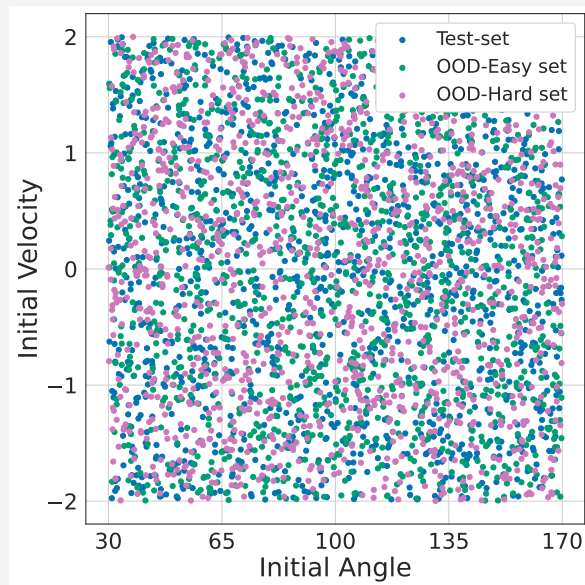
disentanglement allows SD-RSSM to generalize better to unseen conditions, explaining its robustness under increasing distribution shifts. Overall, our results suggest that supervised disentanglement can enhance the modeling of dynamical systems in observation-space sequences, leading to improved long-term predictions and OOD performance.

Table 6.2: Model comparison in observation-space pendulum. Metrics are calculated between ground truth and prediction of the models at the 800th timestep.

	SSIM			PSNR		
	Test-set	OOD Easy	OOD Hard	Test-set	OOD Easy	OOD Hard
RSSM	0.795	0.787	0.783	13.36	12.71	12.26
SD-RSSM	0.813	0.808	0.794	14.16	13.82	12.90



(a) Distributions of pendulum length ℓ and gravity g for training, in-distribution test, and out-of-distribution (OOD) test sets. The in-distribution test set samples from the same parameter range as training. The two OOD test sets sample from non-overlapping ranges that lie progressively farther from the training distribution. We label them **OOD-Easy** and **OOD-Hard** based on the degree of distributional shift—**Easy** being closer and **Hard** farther away. These labels refer to the magnitude of the shift in parameter space, not necessarily the difficulty of the prediction task (see [Section 5.2.1](#)).



(b) Distributions of initial angle θ and angular velocity ω for all datasets. These initial conditions are drawn from the same uniform distributions across training, in-distribution test, and OOD test sets.

Figure 6.1: Parameter distributions for the pendulum datasets. The top subfigure illustrates how the pendulum length and gravity vary among the different datasets, emphasizing the distribution shifts in the OOD test sets. The bottom subfigure shows that the initial angle and angular velocity distributions remain consistent across all datasets.

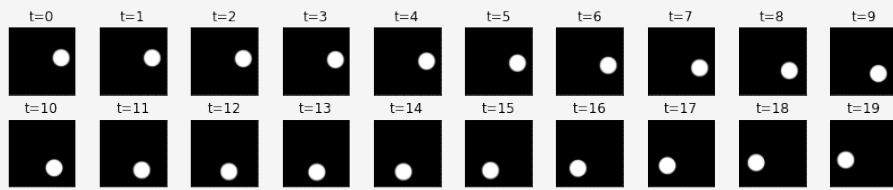
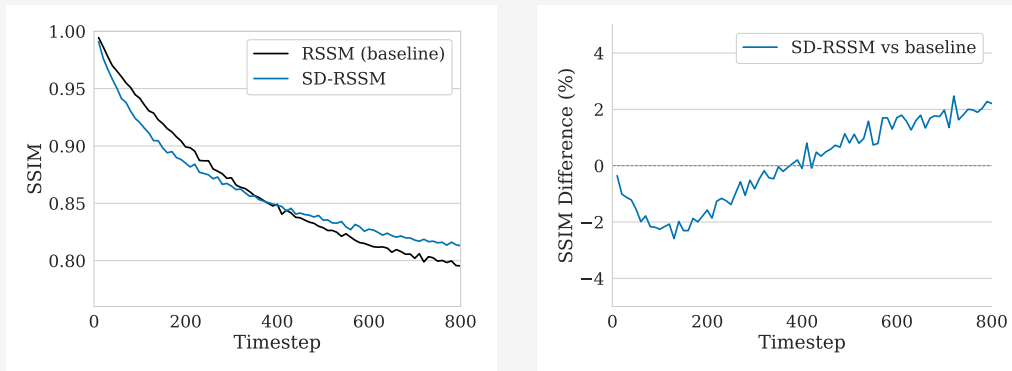
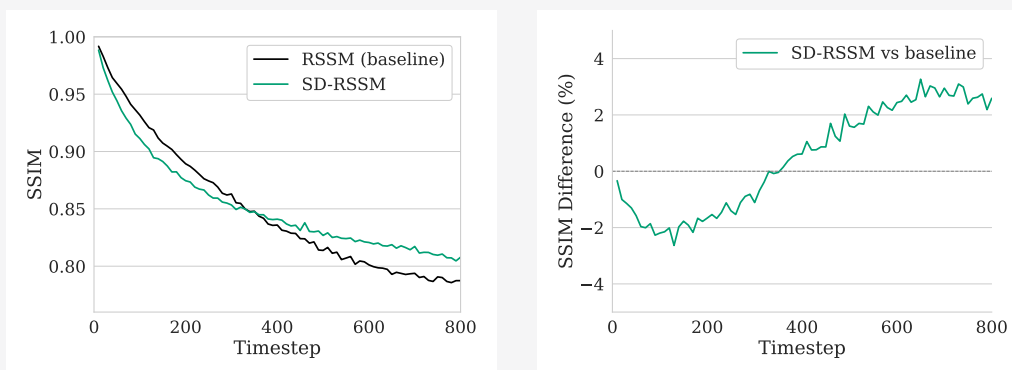


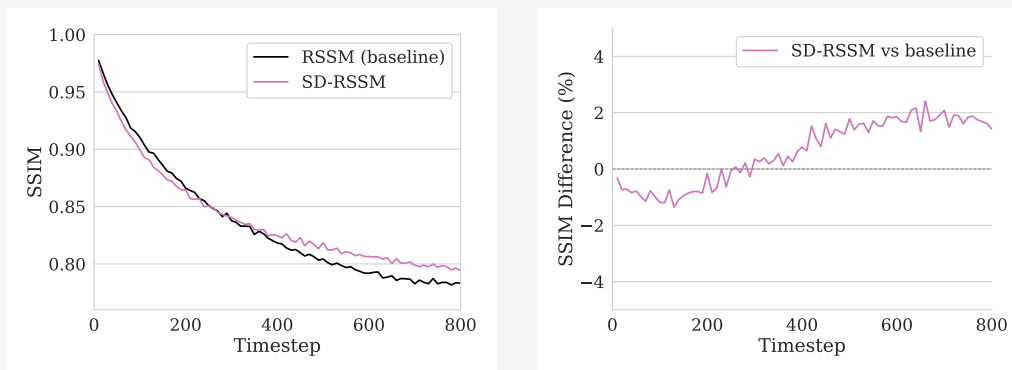
Figure 6.2: Example frame sequence from the pendulum data set



(a) In-Distribution Test Set. RSSM performs better in maintaining structural similarity in short-term predictions, whereas SD-RSSM excels in preserving structural integrity over long-term dynamics beyond 400 timesteps.

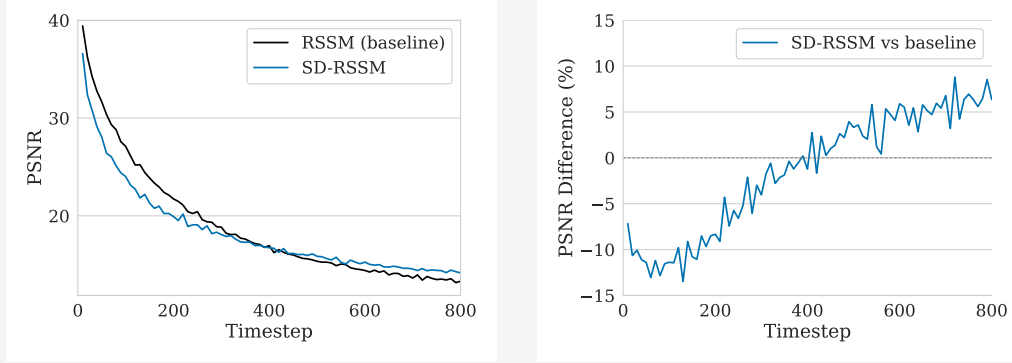


(b) OOD Easy Test Set. In this out-of-distribution scenario with a longer pendulum length and slower angular velocity, SD-RSSM demonstrates superior structural similarity in capturing long-term dynamics starting around 350 timesteps, outperforming RSSM.

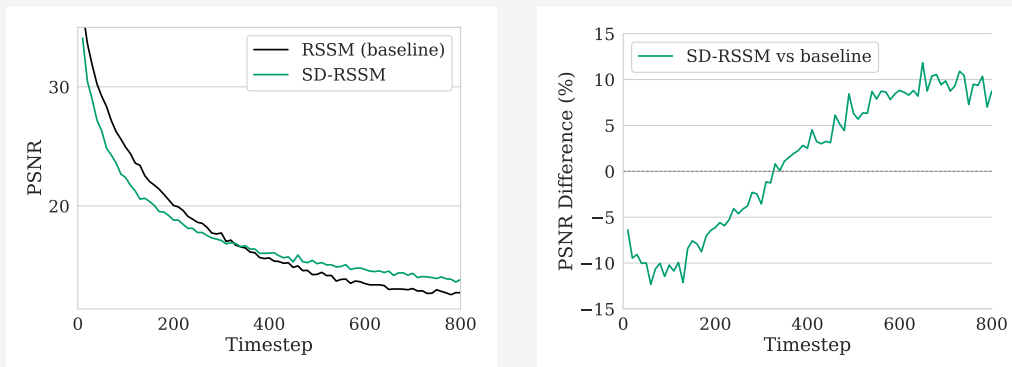


(c) OOD Hard Test Set. This challenging OOD scenario has a shorter pendulum length and higher angular velocity. SD-RSSM maintains higher SSIM in long-term predictions after approximately 200 timesteps, outperforming RSSM.

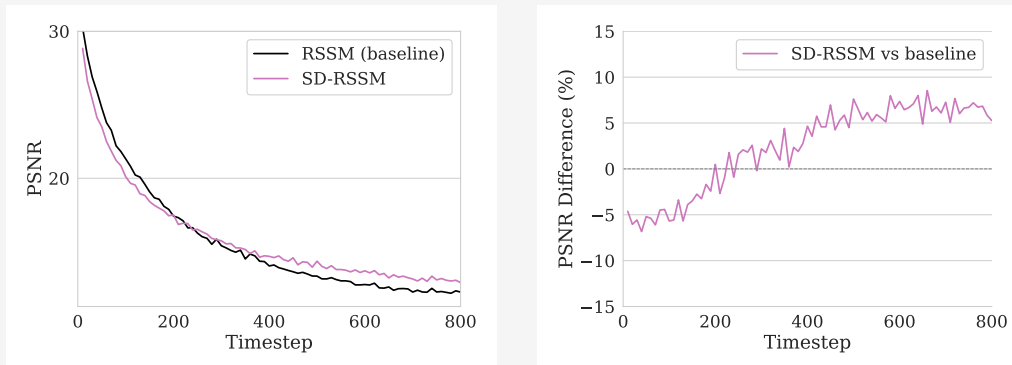
Figure 6.3: SSIM Comparison Across Pendulum Datasets. We compare RSSM and SD-RSSM on three pendulum datasets (in-distribution, OOD Easy, OOD Hard) using SSIM (higher is better). While RSSM excels in short-term similarity, SD-RSSM preserves long-term structure more effectively, especially under distribution shifts, underscoring the benefits of disentanglement.



(a) In-distribution test set. RSSM is better in short term prediction while SD-RSSM is better in capturing long term dynamics after 400 timesteps.



(b) OOD Easy Test Set. In this scenario with a longer pendulum length and slower angular velocity, SD-RSSM surpasses RSSM in capturing long-term dynamics starting around 350 timesteps.



(c) OOD Hard Test Set. This challenging scenario features a shorter pendulum length and higher angular velocity. SD-RSSM outperforms RSSM in capturing long-term dynamics after about 200 timesteps.

Figure 6.4: PSNR Comparison Across Pendulum Datasets. (higher is better) comparison between RSSM and SD-RSSM across three test datasets. RSSM demonstrates superior performance in short-term predictions, whereas SD-RSSM excels in capturing long-term dynamics. As the distribution shift increases from in-distribution to OOD Easy and OOD Hard, SD-RSSM outperforms RSSM over longer ranges, highlighting the effectiveness of disentanglement in OOD scenarios.

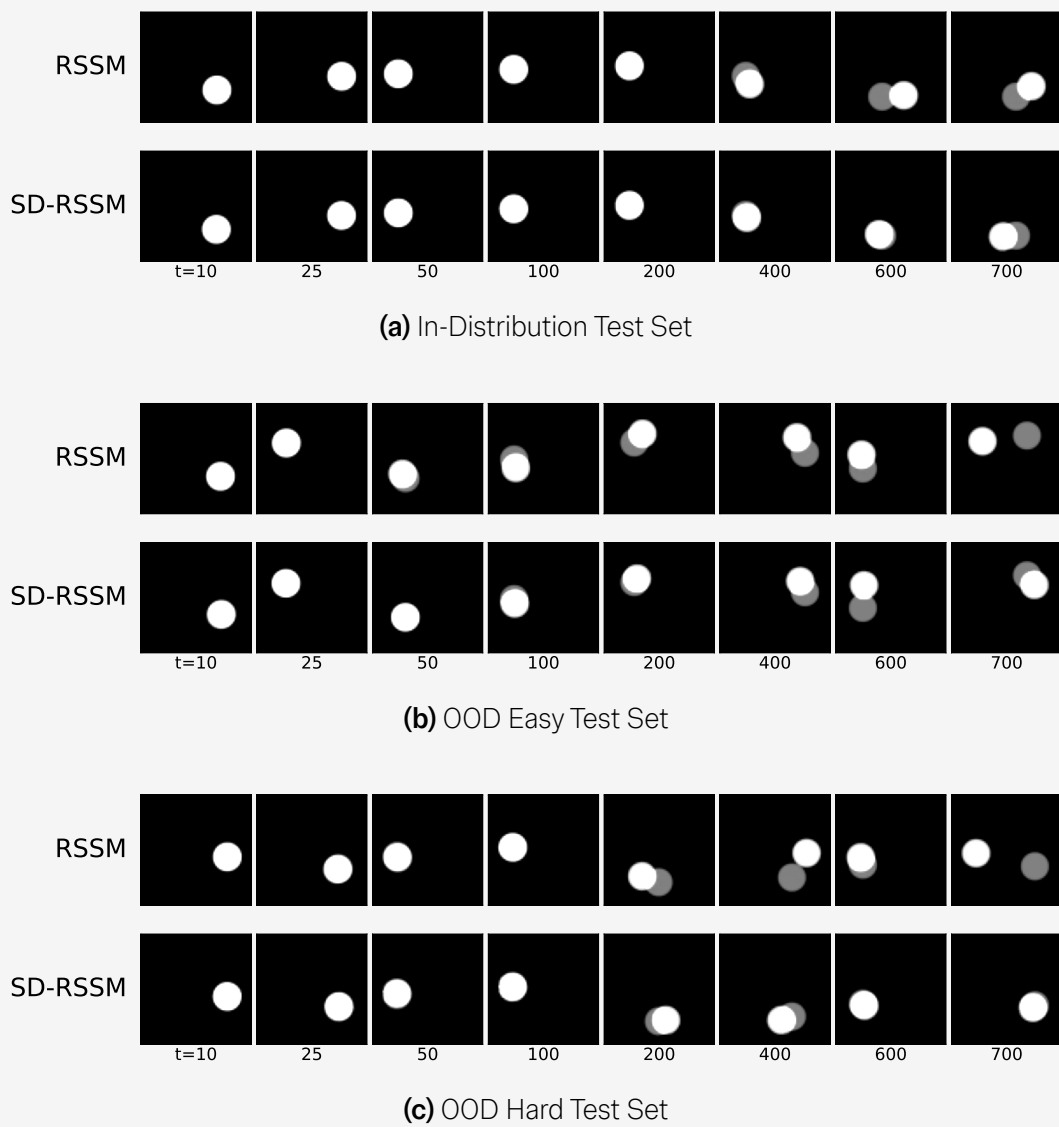


Figure 6.5: Samples of Predicted Trajectories. Difference between ground truth (gray) and predictions (white) on the pendulum dataset test sets: (a) In-Distribution Test Set, (b) OOD Easy Test Set, and (c) OOD Hard Test Set. These figures illustrate how SD-RSSM better captures the longer-term dynamics of the pendulum, maintaining closer alignment with the ground truth trajectory even under distributional shifts.

6.3 Wave Propagation

6.3.1 Dataset

To further investigate the impact of supervised disentanglement in modeling complex dynamical systems, we extended our experiments to wave propagation governed by the Saint-Venant equations, which describe shallow water dynamics (Equation (4.4)). This system presents a higher-dimensional spatiotemporal challenge compared to the pendulum, involving intricate wave interactions and boundary effects.

We generated datasets by simulating the Saint-Venant equations using the `triflow` package in `python` (Cellier, 2019). Unless specified otherwise, we employed the same simulation parameters as in Chapter 4. Reflective boundary conditions were implemented to emulate a closed tank environment, and initial wave excitations were modeled as single Gaussian droplets randomly positioned within the grid to introduce variability in the wave patterns. The tank depth was fixed at $H = 10$ meters.

Each simulation produced a sequence of frames representing the water surface over time. Frames were rendered with lighting positioned at 45° azimuth and 20° altitude to enhance the visualization of the waveforms. Sequences spanned 100 time steps with a temporal resolution of 0.01 seconds, capturing the dynamic evolution of the wave fronts. The rendering resolution was set to 184×184 pixels to ensure sufficient spatial detail for modeling.

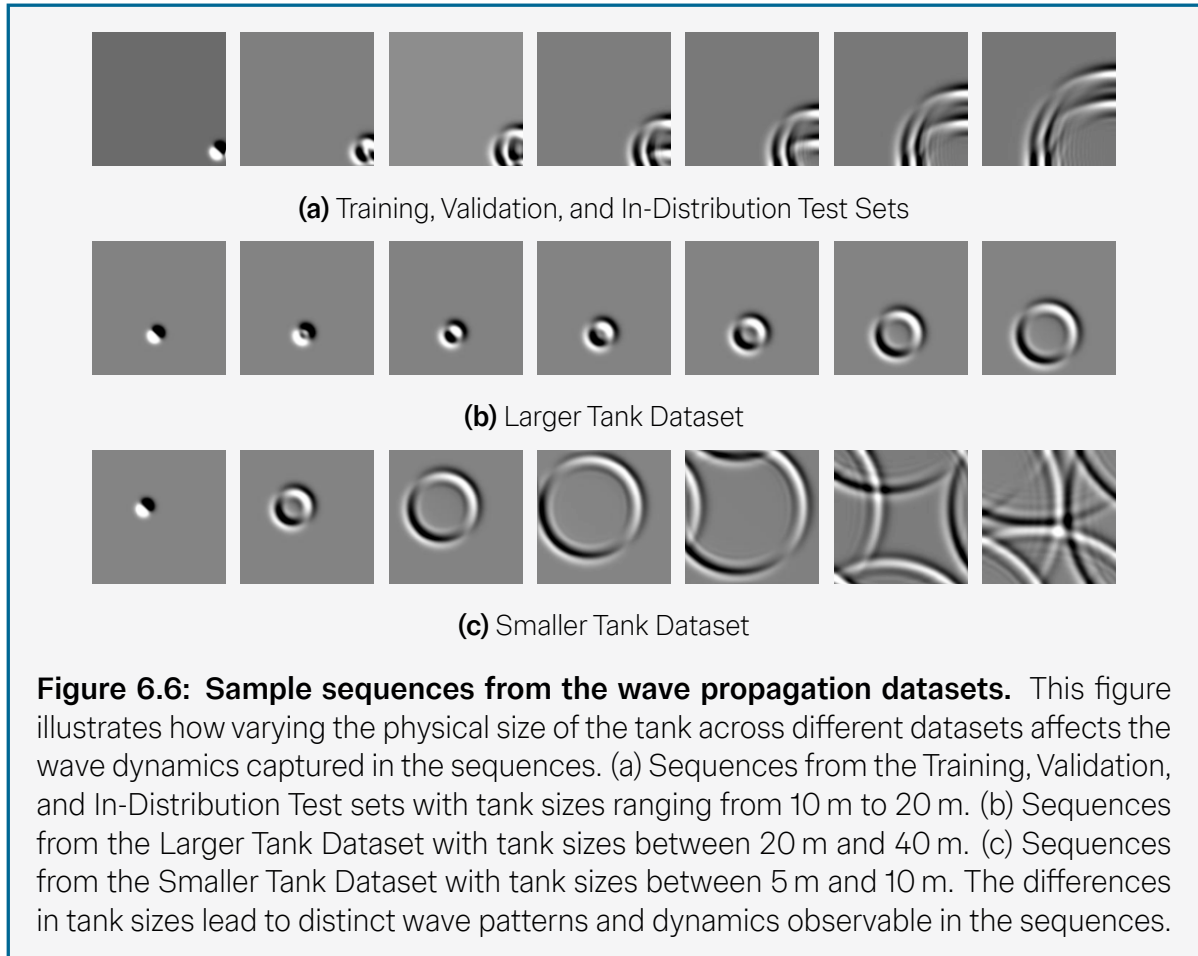
To examine the effects of supervised disentanglement on out-of-distribution (OOD) generalization, we varied the physical size of the tank between datasets. One in-distribution and two OOD test sets were created by altering the tank size distributions:

- **In-distribution train/val/test sets:** Tank sizes drawn from $\mathcal{U}(10, 20)$ meters
- **Smaller Tank Dataset:** Tank sizes drawn from $\mathcal{U}(5, 10)$ meters
- **Larger Tank Dataset:** Tank sizes drawn from $\mathcal{U}(20, 40)$ meters

The in-distribution dataset comprised 3,000 sequences, split into 2,500 for training, 200 for validation, and 300 for testing. Each OOD dataset contained 300 sequences. Sample sequences from each dataset are illustrated in Figure 6.6, showcasing the diversity in wave dynamics resulting from different tank sizes.

By varying the tank size—a key physical parameter affecting wave behavior—we introduced a controlled distribution shift. This setup allowed us to evaluate the ability of models, trained with and without supervised disentanglement, to generalize to unseen

physical conditions, which is crucial for applications in fields like oceanography, meteorology, and engineering.



6.3.2 Method

To model the spatiotemporal dynamics of wave propagation from high-dimensional observations, we employed a U-Net architecture (Ronneberger et al., 2015). The U-Net effectively captures spatial hierarchies in image data due to its convolutional layers and encoder-decoder structure with skip connections. It is well-suited for tasks requiring detailed pixel-level predictions, making it appropriate for modeling the evolution of physical systems (de Bézenac et al., 2018). In our prior study of wave propagation modeling in Chapter 4, the U-Net demonstrated strong performance. The encoder-decoder structure provides an information bottleneck, resulting in a latent space of reduced dimensionality—an ideal setting for applying disentanglement techniques.

We modified the U-Net to incorporate supervised disentanglement by supervising its latent space representation. Specifically, we introduced a disentanglement loss that encourages certain functions of the latent code to explicitly encode the key physical parameter—the tank size—while conditioning the rest of the latent space to be invariant to this parameter. This

design aims to separate the influence of different factors of variation within the data, allowing the remaining latent dimensions to focus on capturing the system's dynamics independently of the tank size.

The general form of the loss function we employed is:

$$\mathcal{L}(\mathbf{x}_{1:t}) = \mathcal{L}_{\text{pred}}(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \delta_1 \mathcal{L}_{\text{dis}}(f(\mathbf{z}), \boldsymbol{\xi}) - \delta_2 \mathcal{L}_{\text{inv}}(g(\mathbf{z}), \boldsymbol{\xi}) \quad (6.5)$$

where:

- $\mathcal{L}_{\text{pred}}$ is the prediction loss of the U-Net, computed as the mean squared error between the predicted frame $\hat{\mathbf{x}}_t$ and the ground truth frame $\mathbf{x}_t$.
- $f(\mathbf{z})$ and $g(\mathbf{z})$ are functions that operate on different portions of the latent representation $\mathbf{z}$.
- $\mathcal{L}_{\text{dis}}(f(\mathbf{z}), \boldsymbol{\xi})$ is the disentanglement loss encouraging $f(\mathbf{z})$ to predict the tank size $\boldsymbol{\xi}$.
- $\mathcal{L}_{\text{inv}}(g(\mathbf{z}), \boldsymbol{\xi})$ is the invariance loss discouraging $g(\mathbf{z})$ from containing information about $\boldsymbol{\xi}$.
- δ_1 and δ_2 are positive parameters controlling the strengths of the disentanglement and invariance losses, respectively.

By using a negative invariance loss, we discourage $g(\mathbf{z})$ from capturing information about the tank size, effectively promoting invariance.

We choose $\mathcal{L}_{\text{pred}}$, $\mathcal{L}_{\text{dis}}$ and $\mathcal{L}_{\text{inv}}$ to be L2 losses which is a typical choice in this setting (Mathieu et al., 2016b; Tschannen et al., 2018). We also define $\hat{\boldsymbol{\xi}} = f(\mathbf{z})$ and $\tilde{\boldsymbol{\xi}} = g(\mathbf{z})$. In this case the loss takes the following form:

$$\mathcal{L}_{\text{NN}}(\mathbf{x}_{1:t}) = \mathcal{L}_{\text{pred}} + \delta_1 \left\| \hat{\boldsymbol{\xi}} - \boldsymbol{\xi} \right\|_2^2 - \delta_2 \left\| \tilde{\boldsymbol{\xi}} - \boldsymbol{\xi} \right\|_2^2 \quad (6.6)$$

Overall, this approach provides a flexible framework for disentangling physical parameters from the latent representation by factorizing the latent space around a known variable (in this case, tank size). Concretely, the model dedicates a specific portion of the latent space to encoding tank size while keeping the remainder invariant to it. This ensures that the model focuses on capturing the underlying dynamics more effectively, since the key physical parameter is handled explicitly.

We explored two specific implementations for the functions f and g :

Selector Functions

In the first approach, f and g are used to just select specific portions of $\mathbf{z}$.

- $f(\mathbf{z})$ just extracts the first k channels of $\mathbf{z}$:

$$\hat{\xi} = f(\mathbf{z}) = \mathbf{z}^k \quad (6.7)$$

- $g(\mathbf{z})$ selects the remaining channels $\mathbf{z}^{k+1:D}$:

$$g(\mathbf{z}) = \mathbf{z}^{k+1:D} \quad (6.8)$$

where D is the total number of channels in $\mathbf{z}$. The full loss function in this case is:

$$\mathcal{L}_{\text{ID}}(\mathbf{x}_{1:t}) = \mathcal{L}_{\text{pred}} + \delta_1 \|\mathbf{z}^k - \xi\|_2^2 + \delta_2 \|\mathbf{z}^{k+1:D}\|_2^2 \quad (6.9)$$

Neural Networks

In the second approach, both f and g are neural networks trained alongside the main U-Net architecture.

- $f(\mathbf{z})$ is a neural network that takes a designated portion $\mathbf{z}^k$ of the latent space as input and predicts the tank size parameter ξ :

$$\hat{\xi} = f(\mathbf{z}) = \text{NN}(\mathbf{z}^k) \quad (6.10)$$

- $g(\mathbf{z})$ is another neural network that takes the remaining portion $\mathbf{z}^{k+1:D}$ of the latent space and attempts to predict ξ . We aim to minimize its ability to do so:

$$\tilde{\xi} = g(\mathbf{z}) = \text{NN}(\mathbf{z}^{k+1:D}) \quad (6.11)$$

In general f, g can be parameterized by different or the same neural network and we explore this choice in our experiments.

6.4 Experiments

6.4.1 Model Architecture and Training

To evaluate the effectiveness of supervised disentanglement in modeling wave propagation from high-dimensional observations, we conducted experiments comparing a baseline U-Net model with a U-Net incorporating our proposed disentanglement loss. Both models were trained on the in-distribution training set, aiming to predict future frames based on sequences of input frames, thereby learning the spatiotemporal evolution of the wave surface.

We employed a U-Net architecture consistent with [Chapter 4](#). Specifically, the encoder consists of 4 blocks, each containing 2 convolutional layers with 3×3 kernels (padding of 1) followed by ReLU activations and max-pooling layers of stride 2. We double the number of feature maps at each block. The decoder mirrors this structure using bilinear upsampling (scale factor of 2) instead of deconvolutions to reduce the number of parameters. The bottleneck layer has 128 channels with spatial dimensions of 8×8 . For the input, the models received sequences of 10 frames depicting the initial wave conditions. The task was to predict the subsequent 20 frames, effectively modeling the waves spatiotemporal dynamics.

Training was performed over 200 epochs using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 16, which were chosen based on preliminary experiments. Hyperparameters were fine-tuned based on validation set performance. We conducted a grid search to optimize the disentanglement coefficient δ_1 and the invariance coefficient δ_2 . For experiments where f and g were neural networks, we use 2-layer fully connected networks and empirically select the number of hidden neuron. We also explored using a shared neural network versus separate networks for f and g . These hyperparameter combinations resulted in a total of 64 experiments as detailed in [Table 6.3](#).

Table 6.3: Hyperparameter configurations and the number of experiments conducted for wave propagation disentanglement.

Hyperparameter	Values	Number of Values
Disentanglement coeff. δ_1	0.001, 0.01, 0.1, 1.0	4
Invariance coeff. δ_2	0.001, 0.01, 0.1, 1.0	4
Functions f, g	shared NN, separate NNs	2
Hidden layer size	512, 1024	2
Total Experiments		64

6.4.2 Evaluation Protocol

During testing, models were provided with an initial sequence of 10 frames and tasked with predicting up to 70 future frames without additional input, allowing us to evaluate their long-term predictive capabilities. We assessed model performance on three datasets:

- **In-Distribution Test Set:** Sequences that were not used during training but whose distribution of tank sizes is the same with the training set.
- **Smaller Tank Dataset:** Evaluates generalization to a smaller physical domain than those seen during training.
- **Larger Tank Dataset:** Assesses performance on a larger physical domain, testing the model's extrapolation abilities.

Data acquisition details can be found in [Section 6.3.1](#). Model accuracy was measured using the [RMSE](#) between predicted frames and ground truth, consistent with the metrics used in our previous analyses in [Chapter 4](#).

6.5 Results

6.5.1 Choice of f, g

Our experiments aimed to assess the impact of supervised disentanglement in the ability to accurately predict wave dynamics and generalize to new physical conditions. We initially explored using selector functions f and g that simply selected portions of the latent space. However, this approach did not yield satisfactory results—the models struggled to effectively disentangle the tank size parameter using identity mappings, likely due to the complexity of the relationships between the latent representations and the physical parameters.

We then experimented with parameterizing f and g as neural networks. Initially, we tried using a shared neural network for both f and g , but this configuration also failed to produce significant improvements. Finally, we achieved better performance by using two separate neural networks for f and g , allowing each to learn distinct mappings tailored to their respective roles in the disentanglement process.

6.5.2 Disentanglement and invariance

To analyze the impact of the disentanglement and invariance losses, we defined two variants of our model:

- **Supervised Disentangled U-Net (SD-UNet):** Incorporates the disentanglement loss ($\delta_1 > 0$) while setting the invariance loss coefficient to zero ($\delta_2 = 0$).
- **Supervised Disentangled and Invariant U-Net (SDI-UNet):** Utilizes both the disentanglement and invariance losses ($\delta_1, \delta_2 > 0$).

After the hyperparameter tuning, we chose the best SD-UNet and SDI-UNet configurations and compare them with the baseline model. We evaluated the performance of these models across the prediction horizon on all datasets.

6.5.3 In-Distribution Test Set

Figure 6.7a shows that both the SD-UNet and SDI-UNet achieved lower RMSE values than the standard U-Net throughout the prediction horizon on the in-distribution test set. This improvement indicates that supervised disentanglement enhances the model's ability to capture and predict the underlying dynamics within the training distribution. Interestingly, the SD-UNet and SDI-UNet exhibited nearly identical performance in this setting, suggesting

that the invariance loss does not significantly affect in-distribution accuracy.

6.5.4 Smaller Tank Dataset

On the Smaller Tank dataset, which represents a distribution shift to smaller physical domains, the SD-UNet's performance was slightly worse than the baseline U-Net, as shown in Figure 6.7b. However, the SDI-UNet, which includes the invariance loss, improved upon the SD-UNet's performance and closely matched that of the baseline U-Net. This suggests that while the SD-UNet may have overfitted to the tank sizes seen during training, the inclusion of the invariance loss in the SDI-UNet helped mitigate this issue by encouraging the latent representation to be invariant to the tank size parameter, thus enhancing generalization to unseen smaller tanks.

6.5.5 Larger Tank Dataset

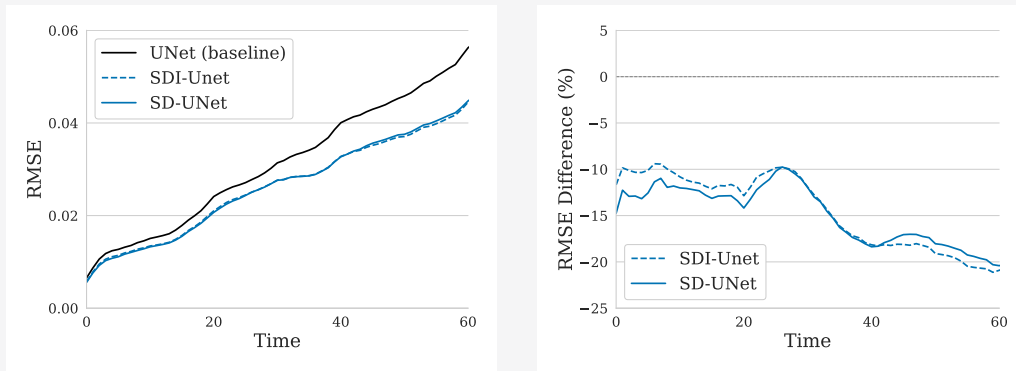
In the Larger Tank dataset, representing a shift to larger physical domains, both the SD-UNet and SDI-UNet maintained lower RMSE values than the baseline U-Net (see Chapter 2) up to approximately 45 prediction steps, as depicted in Figure 6.7c. Beyond this point, the SDI-UNet's performance became comparable to the baseline, while the SD-UNet's RMSE increased slightly. This indicates that the models incorporating supervised disentanglement were better at generalizing to larger tanks in the short to medium term, with the invariance loss in the SDI-UNet providing additional benefits in maintaining performance over longer prediction horizons.

6.5.6 Discussion

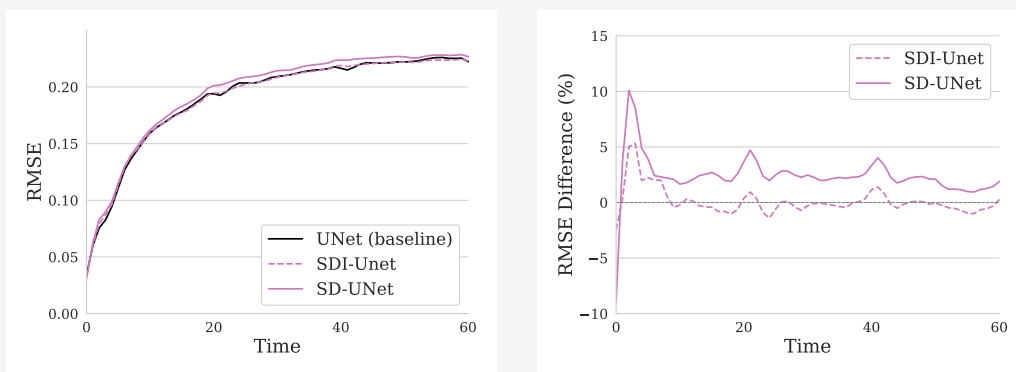
The experimental results indicate that incorporating supervised disentanglement into the U-Net architecture enhances the model's ability to predict complex spatiotemporal dynamics, both within the training distribution and when generalizing to new physical conditions. The improved performance of the SD-UNet on the in-distribution test set suggests that disentangling key physical parameters helps the model focus on the underlying dynamics, leading to more accurate predictions.

The inclusion of the invariance loss in the SDI-UNet proved beneficial in scenarios involving out-of-distribution data, particularly when the physical parameters shifted to values not seen during training. By encouraging the latent representation to be invariant to the tank size parameter, the SDI-UNet demonstrated improved generalization to both smaller and larger tank sizes compared to the SD-UNet and the baseline model.

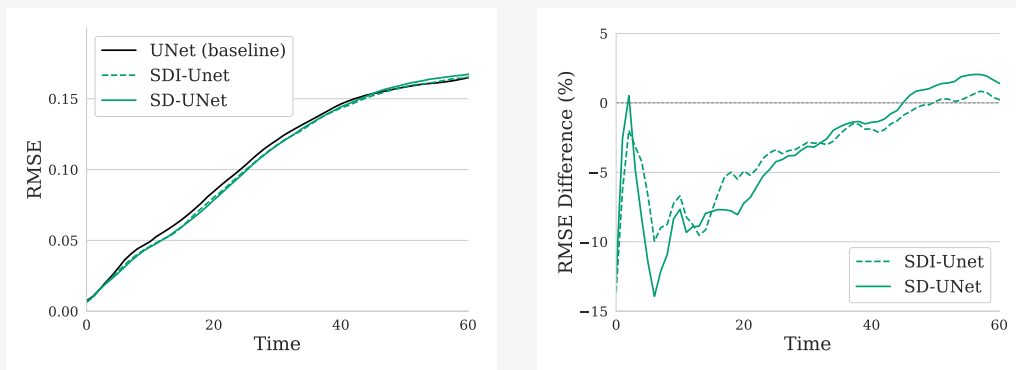
These findings stand in contrast to the results observed in Chapter 4, where models trained without disentanglement struggled to generalize to new physical conditions. In that set-



(a) In-Distribution Test Set. Both SD-UNet and SDI-UNet outperform the baseline U-Net, achieving lower RMSE values throughout the prediction horizon.



(b) Smaller Tank. The SD-UNet slightly underperforms the baseline U-Net, indicating potential overfitting. However, the SDI-UNet closely matches the baseline performance, indicating that the invariance loss enhances generalization.



(c) Bigger Tank. The SDI-UNet maintain lower RMSE values than the baseline U-Net (see Chapter 2) up to approximately 45 prediction steps. Beyond this horizon, the SDI-UNet continues to perform comparably to the baseline, while the SD-UNet's performance slightly degrades. This demonstrates the effectiveness of supervised disentanglement and invariance in generalizing to larger tank sizes over extended prediction horizons.

Figure 6.7: Wave Propagation Experiments RMSE Comparison Root Mean Square Error (RMSE, lower is better) comparison among baseline U-Net, SD-UNet, and SDI-UNet across three datasets. In the in-distribution test set and the Larger Tank, both SD-UNet and SDI-UNet consistently outperform the baseline. For the Smaller Tank, SDI-UNet better maintains baseline performance. These results highlight the effectiveness of disentanglement and invariance in enhancing model robustness and generalization.

ting, the U-Net baseline showed strong performance in-distribution but exhibited noticeable degradation when tested under unseen boundary conditions or parameter shifts. The current results suggest that this degradation can be partially mitigated by explicitly guiding the model to separate invariant structural features from variable physical parameters.

These findings highlight the potential of supervised disentanglement to improve the robustness and generalization capabilities of models tasked with predicting the evolution of complex dynamical systems from high-dimensional observations. The approach shows promise for applications where the system may encounter conditions different from those seen during training, such as in real-world scenarios where environmental variables can vary widely.

6.6 Concluding Remarks

In this chapter, we investigated the role of supervised disentanglement in enhancing generative models for modeling dynamical systems from spatiotemporal data. By integrating a disentanglement loss into convolutional neural network architectures, we aimed to encourage the encoding of key physical parameters in a more structured and interpretable latent space.

Our experiments on the pendulum and wave propagation datasets provided insights into the benefits and trade-offs of supervised disentanglement. In the pendulum experiments, the SD-RSSM demonstrated improved performance in long-term predictions, particularly beyond 350 time steps, and showed robustness to out-of-distribution (OOD) shifts in long-term forecasting. However, in short-term predictions, the standard RSSM performed better than the SD-RSSM. This suggests that while supervised disentanglement can enhance long-term prediction capabilities and OOD robustness, it may introduce trade-offs in short-term accuracy.

In the wave propagation experiments, both the supervised disentangled U-Net (SD-UNet) and the variant with an invariance loss (SDI-UNet) outperformed the baseline U-Net on the in-distribution test set. For the OOD datasets, the SDI-UNet, which includes an invariance loss, demonstrated better generalization to unseen physical conditions, particularly in the larger tank dataset. This highlights the potential of combining supervised disentanglement with invariance to improve model robustness in OOD scenarios.

These findings suggest that supervised disentanglement can be a valuable tool in modeling dynamical systems from high-dimensional observations, potentially improving long-term predictions and robustness to distributional shifts. However, the benefits may come with trade-offs in short-term prediction accuracy, as observed in the pendulum experiments. The impact of supervised disentanglement may also depend on the specific architecture and the nature of the dynamical system being modeled.

Future work could explore balancing the trade-offs between short-term and long-term prediction accuracy and investigate how to optimize the disentanglement and invariance losses for different systems and architectures. Additionally, extending the approach to semi-supervised or unsupervised settings, where labeled data for the physical parameters is limited or unavailable, would enhance its applicability to real-world scenarios.

In conclusion, supervised disentanglement shows promise in enhancing generative models for dynamical systems, particularly in improving long-term predictions and robustness to

OOD shifts. Careful consideration of the trade-offs and further research into optimizing the approach are needed to fully realize its potential across various scientific and engineering applications.

Conclusion

Summary of Contributions

In this thesis, we have investigated how incorporating supervised disentanglement into deep learning models can enhance the modeling of dynamical systems. We focused on critical challenges such as achieving accurate long-term predictions, generalizing to OOD scenarios, and interpreting the learned representations. Our research encompasses both low-dimensional systems with directly observable states and high-dimensional systems where observations are derived from complex spatiotemporal data.

The main contributions of this thesis are as follows:

Advancing State-of-the-Art in Wave Propagation Modeling: We developed deep learning models that improve the state-of-the-art performance in predicting wave propagation. By utilizing U-Net architectures along with innovative training strategies, we significantly improved prediction accuracy. Our analysis of OOD scenarios revealed the limitations of deep learning models in capturing intricate spatiotemporal dynamics and emphasized the necessity of integrating domain knowledge into data-driven approaches.

Introducing Supervised Disentanglement in Dynamical Systems Modeling: We have proposed a novel approach that employs supervised disentanglement to guide the learning process of deep generative models in capturing the underlying dynamics of low-dimensional dynamical systems. By supervising the latent space with ground-truth parameters we achieved better long-term forecasting performance and enhanced robustness when confronted with OOD conditions.

Analysis of Latent Space Representations: We performed an in-depth examination of the latent space representations using various disentanglement metrics. The results demonstrated that supervised disentanglement yields more structured and interpretable latent spaces, improving parameter inference and contributing to model interpretability.

Extension to High-Dimensional Spatiotemporal Data: Building on our experiments with low-dimensional systems, we extended the supervised disentanglement approach to high-dimensional observations, including videos and spatiotemporal sequences. Our experiments confirmed the method's applicability in complex, high-dimensional contexts and its advantages in enhancing model robustness.

Investigating Trade-offs and Limitations: We investigated the potential trade-offs associated with supervised disentanglement. While it improves long-term predictive capabilities and OOD robustness, we observed that in specific scenarios, there may be a slight reduction in short-term prediction accuracy. This finding highlights the importance of carefully considering the system and model architecture when applying supervised disentanglement.

Discussion of Findings

Our research illustrates that embedding supervised disentanglement into deep learning models substantially elevates their proficiency in modeling dynamical systems. By explicitly encoding known physical parameters into the latent space, these models concentrate more effectively on capturing the essential dynamics, leading to enhanced long-term prediction capabilities and better generalization to unseen scenarios.

In low-dimensional systems, such as the swinging pendulum and the Lotka–Volterra predator–prey model, the supervised disentangled [VAEs](#) outperformed baseline models like standard [VAEs](#) and [LSTMs](#) in long-term forecasting tasks. By integrating physical parameters into the latent representation, the models more accurately captured critical dynamics, maintaining higher fidelity in predictions across longer time horizons.

For high-dimensional systems, particularly wave propagation governed by the Saint-Venant equations, incorporating supervised disentanglement into architectures like the [Recurrent State Space Model \(RSSM\)](#) and U-Net led to notable improvements in predictive performance. The models demonstrated enhanced resilience to variations in system parameters and were better equipped to handle the complexities of high-dimensional spatiotemporal data.

A notable benefit of our approach is the improved generalization to [OODs](#) scenarios. Models equipped with supervised disentanglement maintained superior prediction accuracy when evaluated on datasets with system parameters outside the training distribution. For instance, in pendulum experiments, the supervised disentangled [RSSM](#) (SD-RSSM) demonstrated greater robustness to changes in pendulum length and gravitational acceleration, outperforming the standard [RSSM](#) in structural similarity and peak signal-to-noise ratio metrics. This finding underscores the efficacy of supervised disentanglement in enhancing model adaptability to novel conditions.

Furthermore, the interpretability of the learned representations was significantly enhanced. By separating physical parameters from dynamic states in the latent space, our models provided clearer insights into how specific parameters influence system behavior. Analysis using metrics such as Disentanglement, Completeness, and Informativeness (DCI), the Score of Attribute Predictability (SAP), and the Mutual Information Gap (MIG) confirmed that su-

pervised disentanglement led to more structured and meaningful latent spaces. This interpretability is valuable in scientific and engineering contexts, where understanding the underlying mechanisms is as important as making accurate predictions.

However, we observed certain trade-offs associated with the integration of supervised disentanglement. In some scenarios, particularly with the swinging pendulum modeled from spatiotemporal data, the approach led to slight reductions in short-term prediction accuracy. This indicates that while our method enhances long-term performance and generalization, it may necessitate a balance between immediate accuracy and long-term benefits. Adjusting the disentanglement strength and carefully tuning hyperparameters are essential steps to manage these trade-offs effectively.

In summary, our findings highlight the advantages of incorporating supervised disentanglement into deep learning models for dynamical systems. By bridging data-driven modeling with physical understanding, we have advanced the development of models that are more accurate, robust, interpretable, and applicable to a wider range of scenarios. These insights provide a strong foundation for future exploration and emphasize the importance of integrating domain knowledge into machine learning models for complex systems.

Implications for Dynamical Systems Modeling

The integration of supervised disentanglement into data-driven models can offers advancements in the modeling of dynamical systems. By explicitly encoding system parameters into the latent space, models achieve improved generalization and allowed for more accurate predictions in out-of-distribution (OOD) scenarios where system parameters differ from those encountered during training.

Supervised disentanglement mitigates error accumulation over time, contributing to robust long-term predictions. This robustness is crucial for forecasting and control tasks across various domains, such as meteorology, financial markets, and autonomous navigation. Accurate long-term forecasts facilitate improved strategic planning and decision-making in these areas.

Furthermore, supervised disentanglement enhances the interpretability of models by separating different factors of variation that influence system behavior. This interpretability is essential in areas like healthcare, where understanding how specific parameters affect patient outcomes can inform treatment decisions and personalized medicine.

The ability to make accurate long-term predictions, maintain robustness to OOD scenarios, and offer interpretable insights makes supervised disentanglement an important tool in data-driven modeling of dynamical systems. These advancements could have significant implications across diverse fields that rely on understanding and predicting complex system

behaviors.

Limitations and Challenges

Despite the significant benefits, our approach presents certain challenges and limitations. The dependency on ground-truth parameter labels for supervised disentanglement imposes additional data requirements. Obtaining accurate parameter labels can be expensive or impractical in some real-world contexts. Extending disentanglement to semi-supervised or unsupervised scenarios, where parameter labels are scarce or unavailable, remains a challenging but important direction for future research.

While the incorporation of disentanglement techniques has enhanced long-term prediction accuracy, the performance of our data-driven models still falls short of traditional numerical methods. This gap reflects a general limitation of data-driven approaches, which often struggle to match the precision and reliability of established numerical algorithms in modeling dynamical systems.

In high-dimensional experiments, we observed that enhancements in long-term and OOD performance, but in one case it was accompanied by a slight decrease in short-term prediction accuracy. This indicates that the success of supervised disentanglement may depend on the specific characteristics of the dynamical system and the architecture employed. Systems exhibiting highly nonlinear dynamics or chaotic behavior may pose additional difficulties. Navigating these trade-offs necessitates careful model tuning and parameter adjustment.

Future Research Directions

The findings of this thesis open several promising directions for future work. Developing methods that require fewer or no parameter labels would enhance the applicability of supervised disentanglement in real-world scenarios. Techniques such as weak supervision, self-supervised learning, or leveraging physical constraints could be explored to reduce label dependency.

Exploring adaptive learning strategies that enable models to refine their representations in response to new information or evolving system dynamics would enhance robustness and adaptability. Extending the supervised disentanglement framework to alternative architectures, including graph neural networks or transformers, may offer advantages in modeling systems characterized by complex interactions or long-range dependencies.

Applying our methodology to real-world systems—such as climate modeling, fluid dynamics, or biological processes—would provide valuable insights into its practicality and effectiveness in intricate environments. Further investigation into the balance between short-

term and long-term prediction accuracy could yield more versatile models. Techniques like multi-objective optimization or adaptive loss weighting might help mitigate associated trade-offs.

Developing more robust and informative metrics for evaluating disentanglement, particularly in high-dimensional contexts, would facilitate the assessment and comparison of different methods. Incorporating uncertainty quantification mechanisms into predictions could enhance model reliability and offer valuable insights into forecast confidence levels.

Enhancing the interpretability of latent spaces and enabling control over specific system behaviors through manipulation of latent variables could have significant practical implications. Addressing computational challenges associated with high-dimensional data and complex models is essential for making the approach more accessible and scalable for large-scale applications.

Closing Remarks

This thesis has demonstrated that supervised disentanglement can play a pivotal role in advancing the modeling of dynamical systems. By guiding models to learn structured and interpretable representations that capture key system parameters, we can achieve improved prediction accuracy, robustness to distributional shifts, and enhanced interpretability.

The integration of supervised disentanglement into both low-dimensional and high-dimensional modeling frameworks underscores its versatility and potential for broader applicability. While challenges remain, particularly in balancing trade-offs and reducing dependency on labeled data, the insights gained from this work provide a strong foundation for future exploration.

In an era where data-driven approaches are increasingly vital for understanding complex systems, the development of models that not only predict but also provide interpretable insights is of great importance. Supervised disentanglement offers a promising pathway toward this goal, bridging the gap between data and knowledge in the study of dynamical systems.

We hope that this thesis inspires further research into integrating domain knowledge into machine learning models, exploring new methods for disentangled representation learning, and applying these techniques to a wider range of dynamical systems. By continuing to merge data-driven models with physical understanding, we move toward more reliable, interpretable and effective tools for modeling the dynamic processes that shape our world.

References

- Aloev, R., Abdullah, I., Akbarova, A., & Juraev, S. (2023). Lyapunov stability of the numerical solution of the Saint-Venant equation. **AIP Conference Proceedings**, 2484(1).
- Alon, U. (2006). **An Introduction to Systems Biology: Design Principles of Biological Circuits**. Taylor & Francis.
- Ayyad, A., Chehadeh, M., Awad, M. I., & Zweiri, Y. (2020). Real-Time System Identification Using Deep Learning for Linear Processes With Application to Unmanned Aerial Vehicles. **IEEE Access**, 8, 122539–122553.
- Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., & Levine, S. (2018). Stochastic Variational Video Prediction. **6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings**.
- Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. **ArXiv preprint**, abs/1803.01271.
- Baines, P. G. (2008). Lorenz, E.N. 1963: Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences* 20, 130–41.1. **Progress in Physical Geography: Earth and Environment**, 32(4), 475–480.
- Baker, N., Alexander, F., Bremer, P.-T., et al. (2019). **Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence** (tech. rep.). USDOE Office of Science (SC), Washington, DC (United States).
- Barber, D. (2011). **Bayesian Reasoning and Machine Learning** (04-2011) [In press]. Cambridge University Press.
- Barber, G., Haile, M. A., & Chen, T. (2021). Joint Parameter Discovery and Generative Modeling of Dynamic Systems. **ArXiv preprint**, abs/2103.10905.
- Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A.,

- Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., ... Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks.
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. **IEEE transactions on pattern analysis and machine intelligence**, **35**, 1798–1828.
- Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Aubet, F.-X., Callot, L., & Januschowski, T. (2022). Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. **ACM Comput. Surv.**, **55**(6).
- Bird, A., & Williams, C. K. (2019). Customizing sequence generation with multitask dynamical systems. **arXiv**.
- Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. **Journal of the American statistical Association**, **112**(518), 859–877.
- Bouallègue, Z. B., Clare, M. C. A., Magnusson, L., Gascón, E., Maier-Gerber, M., Janoušek, M., Rodwell, M., Pinault, F., Dramsch, J. S., Lang, S. T. K., Raoult, B., Rabier, F., Chevallier, M., Sandu, I., Dueben, P., Chantry, M., & Pappenberger, F. (2024). The Rise of Data-Driven Weather Forecasting: A First Statistical Assessment of Machine Learning–Based Weather Forecasts in an Operational-Like Context. **Bulletin of the American Meteorological Society**, **105**(6), E864–E883.
- Brunton, S., & Kutz, J. (2019). **Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control**. Cambridge University Press.
- Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine Learning for Fluid Mechanics. **Annual Review of Fluid Mechanics**, **52**(1), 477–508.
- Carbonneau, M.-A., Zaidi, J., Boilard, J., & Gagnon, G. (2020). Measuring Disentanglement: A Review of Metrics. **ArXiv preprint**, **abs/2012.09276**.
- Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., & Zdeborová, L. (2019). Machine learning and the physical sciences. **ArXiv preprint**, **abs/1903.10563**.
- Cellier, N. (2019). Scikit-fdiff / skfdiff [[Online; accessed 11-8-2019]].

- Chen, T. Q., Li, X., Grosse, R. B., & Duvenaud, D. (2018a). Isolating Sources of Disentanglement in Variational Autoencoders. **Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada**, 2615–2625.
- Chen, T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018b). Neural Ordinary Differential Equations. **Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada**, 6572–6583.
- Choe, Y. J., Shin, J., & Spencer, N. (2017). Probabilistic interpretations of recurrent neural networks. **Probabilistic Graphical Models**.
- Chomiak, T., & Hu, B. (2024). Time-series forecasting through recurrent topology. **Communications Engineering**, 3(1), 9.
- Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv preprint arXiv:1412.3555**.
- Coron, J.-M., d'Andrea-Novell, B., & Bastin, G. (2007). A Strict Lyapunov Function for Boundary Control of Hyperbolic Systems of Conservation Laws. **IEEE Transactions on Automatic Control**, 52(1), 2–11.
- Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., & Ho, S. (2020). Lagrangian Neural Networks. **ArXiv preprint, abs/2003.04630**.
- Cressie, N., & Wikle, C. K. (2015). **Statistics for spatio-temporal data**. John Wiley & Sons.
- Cricri, F., Ni, X., Honkala, M., Aksu, E., & Gabbouj, M. (2016). Video ladder networks. **ArXiv preprint, abs/1612.01756**.
- Das*, A., Fotiadis*, S., Batra, A., Nabiei, F., Liao, F., Vakili, S., Shiu, D.-S., & Bernacchia, A. (2023). Image generation with shortest path diffusion. **Proceedings of the International Conference on Machine Learning (ICML) *equal contribution**.
- De Bézenac, E., Pajot, A., & Gallinari, P. (2019). Deep learning for physical processes: Incorporating prior scientific knowledge. **Journal of Statistical Mechanics: Theory and Experiment**, 2019(12), 124009.
- de Bézenac, E., Pajot, A., & Gallinari, P. (2018). Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge. **6th International Conference on Learning Repre-**

- sentations, **ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings**.
- Dellaportas, P., & Roberts, G. O. (2003). An Introduction to MCMC. In **Spatial Statistics and Computational Methods** (pp. 1–41). Springer New York.
- Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., & Brox, T. (2015). FlowNet: Learning Optical Flow with Convolutional Networks. **2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015**, 2758–2766.
- Doucet, A., Johansen, A. M., et al. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. **Handbook of nonlinear filtering**, 12(656-704), 3.
- Duan, J.-C., Li, S., & Xu, Y. (2023). Sequential Monte Carlo optimization and statistical inference. **Wiley Interdisciplinary Reviews: Computational Statistics**, 15(3), e1598.
- Durbin, J., & Koopman, S. (2001). **Time series analysis by state space methods**. Oxford University Press.
- Eastwood, C., & Williams, C. K. I. (2018). A Framework for the Quantitative Evaluation of Disentangled Representations. **6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings**.
- Ehrhardt, S., Monszpart, A., Mitra, N. J., & Vedaldi, A. (2017). Learning a physical long-term predictor. **ArXiv preprint, abs/1703.00247**.
- Ersoy, M., Lakkis, O., & Townsend, P. (2017). A Saint-Venant shallow water model for overland flows with precipitation and recharge.
- Evans, L. C. (2010). **Partial differential equations**. American Mathematical Society.
- Finn, C., Goodfellow, I. J., & Levine, S. (2016). Unsupervised Learning for Physical Interaction through Video Prediction. **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**, 64–72.
- Fotiadis, S., Pignatelli, E., Lino, M., Cantwell, C., Storkey, A., & Bharath, A. A. (2020). Comparing recurrent and convolutional neural networks for predicting wave propagation.

International Conference on Learning Representations (ICLR) 2020 Workshop on Deep Learning and Differential Equations.

- Fotiadis, S., Lino, M., Cantwell, C., & Bharath, A. (2021). Disentangling ODE parameters from dynamics in VAEs. **International Conference on Machine Learning (ICML) 2021 Workshop on Semi-Supervised Learning.**
- Fotiadis, S., Lino, M., Hu, S., Garasto, S., Cantwell, C., & Bharath, A. A. (2023). Disentangled Generative Models for Robust Prediction of System Dynamics. **Proceedings of the International Conference on Machine Learning (ICML) 2023**, 10222–10248.
- Fotiadis, S., Brenowitz, N. D., Geffner, T., Cohen, Y., Pritchard, M., Vahdat, A., & Mardani, M. (2025). Adaptive Flow Matching for Resolving Small-Scale Physics. **Proceedings of the International Conference on Machine Learning (ICML).**
- Fraccaro, M., Kamronn, S., Paquet, U., & Winther, O. (2017). A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning. **Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA**, 3601–3610.
- Fragkiadaki, K., Agrawal, P., Levine, S., & Malik, J. (2016). Learning Visual Predictive Models of Physics for Playing Billiards. **4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.**
- Garcia, J. R., Freddi, F., Fotiadis, S., Li, M., Vakili, S., Bernacchia, A., & Hennequin, G. (2023). Fisher-Legendre (FishLeg) optimization of deep neural networks. **Proceedings of the International Conference on Learning Representations (ICLR) 2023, Oral.**
- Garg, S., Rasp, S., & Thuerey, N. (2021). WeatherBench Probability: Medium-range weather forecasts with probabilistic machine learning methods. **EGU General Assembly Conference Abstracts**, Egu21–11448.
- Geneva, N., & Zabaras, N. (2019). Modeling the Dynamics of PDE Systems with Physics-Constrained Deep Auto-Regressive Networks. **Journal of Computational Physics**, **403**, 109056.
- Girin, L., Leglaive, S., Bie, X., Diard, J., Hueber, T., & Alameda-Pineda, X. (2020). Dynamical Variational Autoencoders: A Comprehensive Review. **ArXiv preprint, abs/2008.12595.**

- Golestani, H. B., & Ghanbari, M. (2014). Window size influence on SSIM fidelity. **7'th International Symposium on Telecommunications (IST'2014)**, 355–360.
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., & Bengio, Y. (2014). Generative Adversarial Nets. **Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada**, 2672–2680.
- Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). **Deep Learning**. MIT Press.
- Goroshin, R., Mathieu, M., & LeCun, Y. (2015). Learning to Linearize Under Uncertainty. **Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada**, 1234–1242.
- Goyal, A., & Bengio, Y. (2020). Inductive Biases for Deep Learning of Higher-Level Cognition. **ArXiv preprint, abs/2011.15091**.
- Greydanus, S., Dzamba, M., & Yosinski, J. (2019). Hamiltonian Neural Networks. **Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada**, 15353–15363.
- Hafner, D., Lillicrap, T. P., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). Learning Latent Dynamics for Planning from Pixels. **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, 97**, 2555–2565.
- Hairer, E., Nørsett, S., & Wanner, G. (1993). **Solving Ordinary Differential Equations**. Springer-Verlag.
- Haken, H. (2012). **Advanced Synergetics: Instability Hierarchies of Self-Organizing Systems and Devices**. Springer Berlin Heidelberg.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. **2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016**, 770–778.

- Hendrycks, D., & Dietterich, T. G. (2019). Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. **7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.**
- Heuraux, S., da Silva, F., Ribeiro, T., Despres, B., Campos Pinto, M., Jacquot, J., Faudot, E., Wengertowsky, S., Colas, L., & Lu, L. (2015). Simulation as a tool to improve wave heating in fusion plasmas. **Journal of Plasma Physics**, **81**(5), 435810503.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. **5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.**
- Higgins, I., Sonnerat, N., Matthey, L., Pal, A., Burgess, C. P., Bosnjak, M., Shanahan, M., Botvinick, M., Hassabis, D., & Lerchner, A. (2018). SCAN: Learning Hierarchical Compositional Visual Concepts. **6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.**
- Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. **Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.**
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. **Neural computation**, **9**(8), 1735–1780.
- Huber, P. J. (1992). Robust Estimation of a Location Parameter. **Breakthroughs in Statistics**, **1**, 492–518.
- Iten, R., Metger, T., Wilming, H., del Rio, L., & Renner, R. (2020). Discovering Physical Concepts with Neural Networks. **Physical Review Letters**, **124**(1).
- Jacobsen, C., & Duraisamy, K. (2022). Disentangling Generative Factors of Physical Fields Using Variational Autoencoders. **Frontiers in Physics**, **10**, 536.
- Jeong, S. H., Solenthaler, B., Pollefeys, M., Gross, M., et al. (2018). Data-driven fluid simulations using regression forests. **ArXiv preprint**, [abs/1806.01261](https://arxiv.org/abs/1806.01261).

- Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. **Journal of Basic Engineering**, **82**(1), 35–45.
- Karniadakis, G., & Sherwin, S. J. (2005). **Spectral/hp Element Methods for Computational Fluid Dynamics** (Second). Oxford University Press.
- Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. **Nature Reviews Physics**, **3**(6), 422–440.
- Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., & Solenthaler, B. (2018). Deep Fluids: A Generative Network for Parameterized Fluid Simulations. **ArXiv preprint, abs/1806.02071**.
- Kim, H., & Mnih, A. (2018). Disentangling by Factorising. **Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018**, **80**, 2654–2663.
- Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. **2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings**.
- Koller, D., & Friedman, N. (2009). **Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning**. The MIT Press.
- Krishnan, R. G., Shalit, U., & Sontag, D. (2015). Deep Kalman Filters. **ArXiv preprint, abs/1511.05121**.
- Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. **Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual**, 26548–26560.
- Kumar, A., Sattigeri, P., & Balakrishnan, A. (2018). Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. **6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings**.

- Lai, Y.-C., & Winslow, R. L. (1994). Extreme sensitive dependence on parameters and initial conditions in spatio-temporal chaotic dynamical systems. **Physica D: Nonlinear Phenomena**, **74**(3-4), 353–371.
- Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., et al. (2022). GraphCast: Learning skillful medium-range global weather forecasting. **ArXiv preprint**, [abs/2212.12794](https://arxiv.org/abs/2212.12794).
- Lerer, A., Gross, S., & Fergus, R. (2016). Learning Physical Intuition of Block Towers by Example. **Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016**, **48**, 430–438.
- Li, Y., Lu, X., Wang, Y., & Dou, D. (2022). Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement. **Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022**.
- Li, Y., & Mandt, S. (2018). Disentangled Sequential Autoencoder. **Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018**, **80**, 5656–5665.
- Li, Z., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. M., & Anandkumar, A. (2021). Fourier Neural Operator for Parametric Partial Differential Equations. **9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021**.
- Lino, M., Cantwell, C., Fotiadis, S., Pignatelli, E., & Bharath, A. A. (2020). Simulating surface wave dynamics with convolutional networks. **NeurIPS 2020 Workshop on Interpretable Inductive Biases and Physically Structured Learning**.
- Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. D. (2022a). Multi-scale rotation-equivariant graph neural networks for unsteady Eulerian fluid dynamics. **Physics of Fluids**, **34**(8).
- Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. (2022b). REMuS-GNN: A Rotation-Equivariant Model for Simulating Continuum Dynamics. **International Conference on Learning Representations (ICLR) 2022 Workshop on Geometrical and Topological Representation Learning**.
- Lino, M., Fotiadis, S., Bharath, A., & Cantwell, C. (2022c). Towards fast simulation of environmental fluid mechanics with multi-scale graph neural networks. **International Con-**

ference on Learning Representations (ICLR) Workshop on AI for Earth and Space Science.

- Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. D. (2023). Current and emerging deep-learning methods for the simulation of fluid dynamics. **Proceedings of the Royal Society A**, **479**(2275), 20230058.
- Ljung, L., Andersson, C., Tiels, K., & Schön, T. B. (2020). Deep Learning and System Identification [21st IFAC World Congress]. **IFAC-PapersOnLine**, **53**(2), 1175–1181.
- Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., & Bachem, O. (2019a). Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA**, **97**, 4114–4124.
- Locatello, F., Tschannen, M., Bauer, S., Rätsch, G., Schölkopf, B., & Bachem, O. (2019b). Disentangling Factors of Variation Using Few Labels. **ArXiv preprint**, **abs/1905.01258**.
- Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., & Bachem, O. (2020a). A Sober Look at the Unsupervised Learning of Disentangled Representations and their Evaluation. **J. Mach. Learn. Res.**, **21**, 209:1–209:62.
- Locatello, F., Poole, B., Rätsch, G., Schölkopf, B., Bachem, O., & Tschannen, M. (2020b). Weakly-Supervised Disentanglement Without Compromises. **Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event**, **119**, 6348–6359.
- Long, Z., Lu, Y., & Dong, B. (2019). PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. **Journal of Computational Physics**, **399**.
- Lopez, R., & Atzberger, P. J. (2022). GD-VAEs: Geometric Dynamic Variational Autoencoders for Learning Nonlinear Dynamics and Dimension Reductions. **ArXiv preprint**, **abs/2206.05183**.
- Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning Nonlinear Operators via Deep Neural Networks: A Diffusion Model View. **ArXiv preprint**, **abs/2104.02406**.
- Lusch, B., Kutz, J. N., & Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. **Nature Communications**, **9**(1).

- Mangeney-Castelnau, A., Vilotte, J.-P., Bristeau, M. O., Perthame, B., Bouchut, F., Simeoni, C., & Yerneni, S. (2003). Numerical modeling of avalanches based on Saint Venant equations using a kinetic scheme. **Journal of Geophysical Research: Solid Earth**, **108**(B11).
- Mantegna, R., & Stanley, H. (1999). **Introduction to Econophysics: Correlations and Complexity in Finance**. Cambridge University Press.
- Mathieu, E., Rainforth, T., Siddharth, N., & Teh, Y. W. (2019). Disentangling Disentanglement in Variational Autoencoders. **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA**, **97**, 4402–4412.
- Mathieu, M., Zhao, J., Sprechmann, P., Ramesh, A., & LeCun, Y. (2016a). Disentangling factors of variation in deep representations using adversarial training. **ArXiv preprint, abs/1611.03383**.
- Mathieu, M., Couprie, C., & LeCun, Y. (2016b). Deep multi-scale video prediction beyond mean square error. **4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings**.
- Miladinović, Đ., Gondal, M. W., Schölkopf, B., Buhmann, J. M., & Bauer, S. (2019). Disentangled State Space Representations. **ArXiv preprint, abs/1906.03255**.
- Mottaghi, R., Bagherinezhad, H., Rastegari, M., & Farhadi, A. (2016). Newtonian scene understanding: Unfolding the dynamics of objects in static images. **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 3521–3529.
- Moussa, R., & Bocquillon, C. (2000). Approximation zones of the Saint-Venant equations of flood routing with overbank flow. **Hydrology and Earth System Sciences Discussions**, **4**(2), 251–260.
- Newman, M. E. J. (2010). **Networks: an introduction**. Oxford University Press.
- Ogata, K. (2001). **Modern Control Engineering** (4th). Prentice Hall PTR.
- Oh, J., Guo, X., Lee, H., Lewis, R. L., & Singh, S. P. (2015). Action-Conditional Video Prediction using Deep Networks in Atari Games. **Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada**, 2863–2871.

- Olah, C. (2015). Understanding LSTM Networks [[Online; accessed 11-8-2019]].
- Oprea, S., Martinez-Gonzalez, P., Garcia-Garcia, A., Castro-Vargas, J., Orts-Escolano, S., Garcia-Rodriguez, J., & Argyros, A. (2022). A Review on Deep Learning Techniques for Video Prediction. **IEEE Transactions on Pattern Analysis & Machine Intelligence**, **44**(06), 2806–2826.
- Otto, S. E., Zolman, N., Kutz, J. N., & Brunton, S. L. (2023). A Unified Framework to Enforce, Discover, and Promote Symmetry in Machine Learning.
- Pant, P., & Farimani, A. B. (2020). Deep Learning for Efficient Reconstruction of High-Resolution Turbulent DNS Data. **ArXiv preprint**, **abs/2010.11348**.
- Papamarkou, T., Birdal, T., Bronstein, M., Carlsson, G., Curry, J., Gao, Y., Hajj, M., Kwitt, R., Liò, P., Lorenzo, P. D., Maroulas, V., Miolane, N., Nasrin, F., Ramamurthy, K. N., Rieck, B., Scardapane, S., Schaub, M. T., Veličković, P., Wang, B., ... Zamzmi, G. (2024). Position Paper: Challenges and Opportunities in Topological Deep Learning.
- Park, J., & Park, J. (2019). Physics-induced graph neural network: An application to wind-farm power estimation. **Energy**, **187**, 115883.
- Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. (2018). Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. **Phys. Rev. Lett.**, **120**, 024102.
- PONS, K., MARCER, R., & GOLAY, F. (2016). The Saint-Venant model for tsunami propagation simulation: Application to the 2011 Tohoku event. **XIV èmes Journées Nationales Génie Côtier-Génie Civil, Toulon**, 160.
- Price, I., Sanchez-Gonzalez, A., Alet, F., Ewalds, T., El-Kadi, A., Stott, J., Mohamed, S., Battaglia, P., Lam, R., & Willson, M. (2023). GenCast: Diffusion-based ensemble forecasting for medium-range weather.
- Proctor, J. L., Brunton, S. L., & Kutz, J. N. (2016). Dynamic mode decomposition with control. **SIAM Journal on Applied Dynamical Systems**, **15**(1), 142–161.
- Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. **Proceedings of the IEEE**, **77**(2), 257–286.
- Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. **Journal of Computational Physics**, **357**, 125–141.

- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017a). Numerical Gaussian processes for time-dependent and non-linear partial differential equations. **ArXiv preprint, abs/1703.10230**.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017b). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. **ArXiv preprint, abs/1711.10561**.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017c). Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. **ArXiv preprint, abs/1711.10566**.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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.
- Ren, P., Rao, C., Liu, Y., Wang, J.-X., & Sun, H. (2022). PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. **Computer Methods in Applied Mechanics and Engineering, 389**, 114399.
- Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. **Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, 32**, 1278–1286.
- Robinson, R. D. (2020). Reliable Machine Learning for Medical Imaging Data through Automated Quality Control and Data Harmonization.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. **International Conference on Medical image computing and computer-assisted intervention, 234–241**.
- Roweis, S., & Ghahramani, Z. (1999). A unifying review of linear Gaussian models. **Neural computation, 11(2)**, 305–345.
- Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. **Nature Machine Intelligence, 1(5)**, 206–215.

- Samad, M. A., Choi, S.-W., Kim, C.-S., & Choi, K. (2023). Wave Propagation Modeling Techniques in Tunnel Environments: A Survey. **IEEE Access**, **11**, 2199–2225.
- Särkkä, S., & Svensson, L. (2023). **Bayesian filtering and smoothing** (Vol. 17). Cambridge university press.
- Saxena, V., Ba, J., & Hafner, D. (2021). Clockwork Variational Autoencoders. **Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual**, 29246–29257.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. **Neural Networks**, **61**, 85–117.
- Schölkopf, B. (2019). Causality for Machine Learning. **ArXiv preprint**, [abs/1911.10500](https://arxiv.org/abs/1911.10500).
- Semenova, E., Xu, Y., Howes, A., Rashid, T., Bhatt, S., Mishra, S., & Flaxman, S. (2022). Prior-VAE: encoding spatial priors with variational autoencoders for small-area estimation. **Journal of the Royal Society Interface**, **19**(191), 20220094.
- Serguei, B. (2023). Seismology in the Light of Fundamental Sciences. **Open Journal of Earthquake Research**, **13**(1), 84–112.
- Shi, X., & Yeung, D.-Y. (2018). Machine learning for spatiotemporal sequence forecasting: A survey. **ArXiv preprint**, [abs/1808.06865](https://arxiv.org/abs/1808.06865).
- Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. **Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada**, 802–810.
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. **Journal of Big Data**, **6**(1), 60.
- Shu, R., Chen, Y., Kumar, A., Ermon, S., & Poole, B. (2020). Weakly Supervised Disentanglement with Guarantees. **8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020**.
- Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J. V., Ren, J., & Nado, Z. (2019). Can you trust your model's uncertainty? Evaluating predictive un-

- certainty under dataset shift. **Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada**, 13969–13980.
- Sorteberg, W. E., Garasto, S., Pouplin, A. S., Cantwell, C. D., & Bharath, A. A. (2018). Approximating the solution to wave propagation using deep neural networks. **ArXiv preprint, abs/1812.01609**.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, 15(56), 1929–1958.
- Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). Unsupervised Learning of Video Representations using LSTMs. **Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015**, 37, 843–852.
- Stewart, R., & Ermon, S. (2017). Label-Free Supervision of Neural Networks with Physics and Domain Knowledge. **Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA**, 2576–2582.
- Strogatz, S. H. (2018). **Nonlinear Dynamics and Chaos with Student Solutions Manual: With Applications to Physics, Biology, Chemistry, and Engineering**. CRC Press.
- Surace, S. C., Kutschireiter, A., & Pfister, J.-P. (2019). How to avoid the curse of dimensionality: Scalability of particle filters with and without importance weights. **SIAM review**, 61(1), 79–91.
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. **Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada**, 3104–3112.
- Takeishi, N., & Kalousis, A. (2021). Physics-Integrated Variational Autoencoders for Robust and Interpretable Generative Modeling. **Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual**, 14809–14821.
- Tang, B., & Matteson, D. S. (2021). Probabilistic Transformer For Time Series Analysis. **Advances in Neural Information Processing Systems 34: Annual Conference on Neu-**

- ral Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 23592–23608.
- Tarantola, A. (2005). **Inverse Problem Theory and Methods for Model Parameter Estimation**. Society for Industrial; Applied Mathematics.
- Thoreau, R., Risser, L., Achard, V., Berthelot, B., & Briottet, X. (2022). p³VAE: a physics-integrated generative model. Application to the pixel-wise classification of airborne hyperspectral images.
- Thuerey, N., Weissenow, K., Prantl, L., & Hu, X. (2018). Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows. **ArXiv preprint, abs/1810.08217**.
- Tishby, N., & Zaslavsky, N. (2015). Deep Learning and the Information Bottleneck Principle.
- Tomczak, J. M., & Welling, M. (2018). VAE with a VampPrior. **International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain, 84**, 1214–1223.
- Tompson, J., Schlachter, K., Sprechmann, P., & Perlin, K. (2017). Accelerating Eulerian Fluid Simulation With Convolutional Networks. **Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, 70**, 3424–3433.
- Tschannen, M., Bachem, O., & Lucic, M. (2018). Recent Advances in Autoencoder-Based Representation Learning. **ArXiv preprint, abs/1812.05069**.
- Villegas, R., Yang, J., Hong, S., Lin, X., & Lee, H. (2017). Decomposing Motion and Content for Natural Video Sequence Prediction. **5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings**.
- Vondrick, C., Pirsiavash, H., & Torralba, A. (2016). Generating Videos with Scene Dynamics. **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**, 613–621.
- Wang, Y., Long, M., Wang, J., Gao, Z., & Yu, P. S. (2017). PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. **Advances in Neural Informa-**

- tion Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 879–888.
- Wang, Y., Gao, Z., Long, M., Wang, J., & Yu, P. S. (2018). PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning. **Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, 80**, 5110–5119.
- Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. (2004). Image quality assessment: from error visibility to structural similarity. **IEEE Trans on Image Proc.**, **13**(4), 600–612.
- Watt, M. (1990). Light-water interaction using backward beam tracing. **ACM SIGGRAPH Computer Graphics**, **24**(4), 377–385.
- Willard, J., Jia, X., Xu, S., Steinbach, M., & Kumar, V. (2020). Integrating physics-based modeling with machine learning: A survey. **ArXiv preprint, abs/2003.04919**.
- Wu, J., Lim, J. J., Zhang, H., Tenenbaum, J. B., & Freeman, W. T. (2016). Physics 101: Learning Physical Object Properties from Unlabeled Videos. **Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016**.
- Xia, G., Danier, D., Das, A., Fotiadis, S., Nabiei, F., Sengupta, U., & Bernacchia, A. (2023). Score Normalization for a Faster Diffusion Exponential Integrator Sampler. **NeurIPS 2023 Workshop on Diffusion Models**.
- Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network.
- Yang, R., Hu, J., Li, Z., Mu, J., Yu, T., Xia, J., Li, X., Dasgupta, A., & Xiong, H. (2024). Interpretable Machine Learning for Weather and Climate Prediction: A Survey.
- Yang, Y., & Perdikaris, P. (2018). Physics-informed deep generative models. **ArXiv preprint, abs/1812.03511**.
- Yeo, K., Grullon, D. E., Sun, F.-K., Boning, D. S., & Kalagnanam, J. R. (2021). Variational inference formulation for a model-free simulation of a dynamical system with unknown parameters by a recurrent neural network. **SIAM Journal on Scientific Computing**, **43**(2), A1305–a1335.
- Yi, Z., Wei, Y., Cheng, C. X., He, K., & Sui, Y. (2024). Improving sample efficiency of high dimensional Bayesian optimization with MCMC.

- Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. **Neural Computation**, **31**(7), 1235–1270.
- Zaidi, J., Boilard, J., Gagnon, G., & Carboneau, M.-A. (2020). Measuring Disentanglement: A Review of Metrics. **ArXiv preprint**, [abs/2012.09276](https://arxiv.org/abs/2012.09276).
- Zhang, Y., Lin, Z., Wang, J., & Sato, I. (2021). On the generalization of neural ordinary differential equations. **International Conference on Learning Representations**.
- Zhang, Z., & Chen, Q. (2007). Comparison of the Eulerian and Lagrangian methods for predicting particle transport in enclosed spaces. **Atmospheric environment**, **41**(25), 5236–5248.
- Zhong, W., & Meidani, H. (2022). PI-VAE: Physics-Informed Variational Auto-Encoder for stochastic differential equations. **ArXiv preprint**, [abs/2203.11363](https://arxiv.org/abs/2203.11363).
- Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. **Thirty-Fifth AAAI Conference on Artificial Intelligence, Virtual Event, February 2-9, 2021**, 11106–11115.

Appendices

7.1 Proof of Lemma (2.13)

Lemma. If the posterior distribution $p(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1})$ at time $t - 1$ is available then the posterior at time t is given by:

$$p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \propto p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{z}_t) \int p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{z}_{t-1}) p_{\mathbf{z}}(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_{t-1}$$

Proof. We can then see that:

$$\begin{aligned} p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{x}_{1:t}) &= \frac{p_{\mathbf{z}}(\mathbf{x}_{1:t} \mid \mathbf{z}_t) \cdot p_{\mathbf{z}}(\mathbf{z}_t)}{p(\mathbf{x}_{1:t})} \\ &= \frac{p_{\mathbf{x}}(\mathbf{x}_{1:t-1}, \mathbf{x}_t \mid \mathbf{z}_t) \cdot p_{\mathbf{z}}(\mathbf{z}_t)}{p_{\mathbf{x}}(\mathbf{x}_{1:t-1}, \mathbf{x}_t)} \\ &= \frac{p_{\mathbf{x}}(\mathbf{x}_{1:t-1} \mid \mathbf{z}_t) \cdot p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_t) \cdot p_{\mathbf{z}}(\mathbf{z}_t)}{p_{\mathbf{x}}(\mathbf{x}_{1:t-1}) \cdot p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})} \\ &= \frac{p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) \cdot p_{\mathbf{x}}(\mathbf{x}_{1:t-1}) \cdot p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_t) \cdot p_{\mathbf{z}}(\mathbf{z}_t)}{p_{\mathbf{z}}(\mathbf{z}_t) \cdot p_{\mathbf{x}}(\mathbf{x}_{1:t-1}) p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})} \\ &= \frac{p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) \cdot p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{z}_t)}{p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})} \\ &= \frac{p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) \cdot p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{z}_t)}{p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})} \\ &\propto p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) \cdot p_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{z}_t) \end{aligned}$$

where the first equality is the Bayes rule, second and third from factorizing the joints, fourth from the application of Bayes rule on $p_{\mathbf{x}}(\mathbf{x}_{1:t-1} \mid \mathbf{z}_t)$, and sixth by substituting [Equation \(2.9\)](#).

Furthermore using the Bayes rule and the law of total probability we can write the given

posterior as:

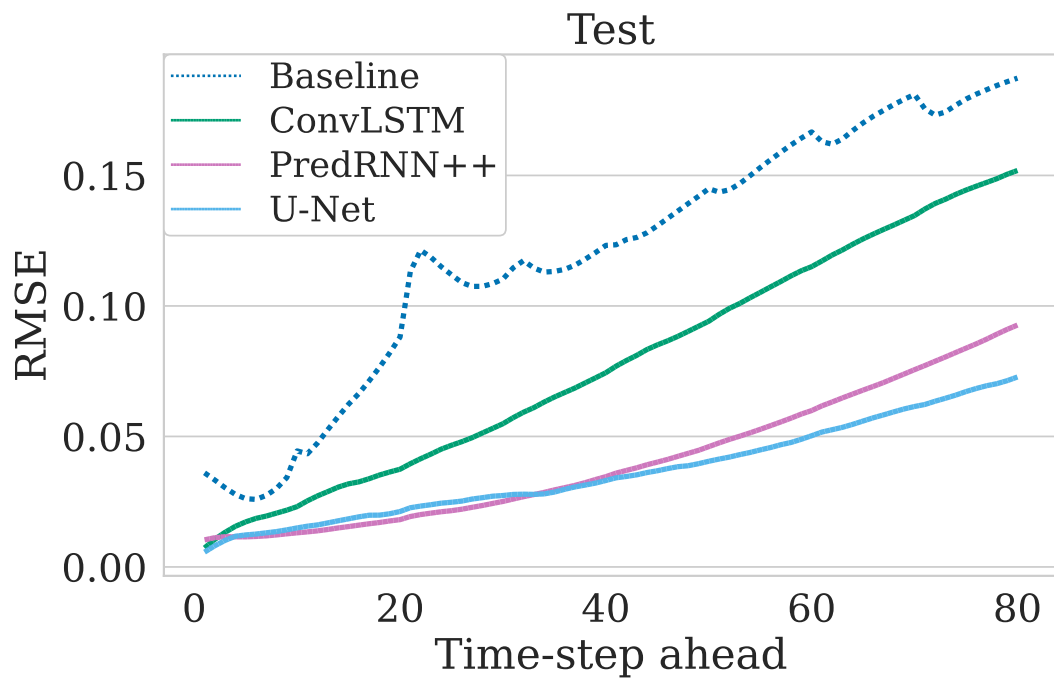
$$p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) = \int p_{\mathbf{z}}(\mathbf{z}_t \mid \mathbf{z}_{t-1}) p_{\mathbf{z}}(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1}) d\mathbf{z}_{t-1}$$

completing the proof. □

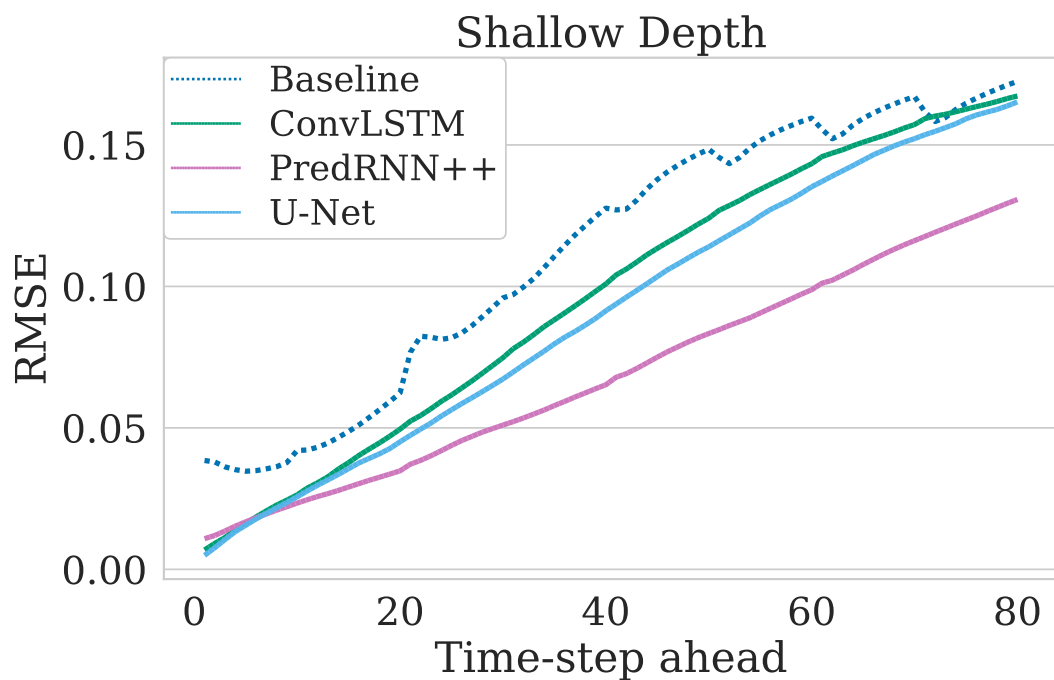
7.2 Wave Propagation Results

7.2.1 Results for all models

7.2.2 Baselines

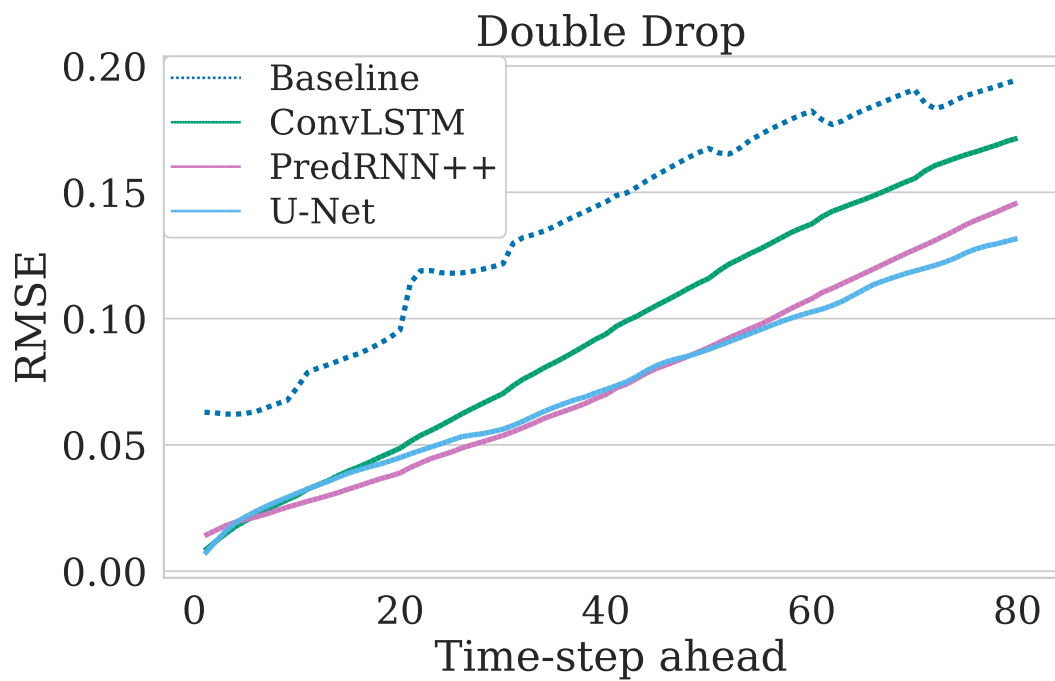


(a) Test Dataset Results

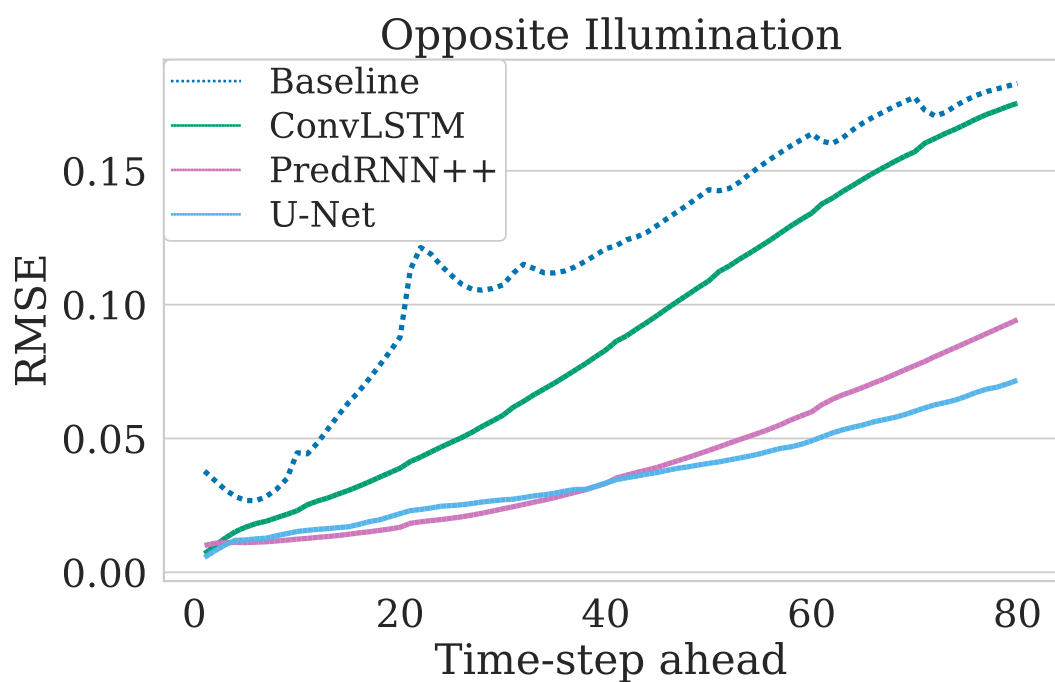


(b) Shallow Depth Dataset Results

Figure 7.1: Out-of-Distribution (OOD) Evaluation Results for All Models. (Continued on next page)

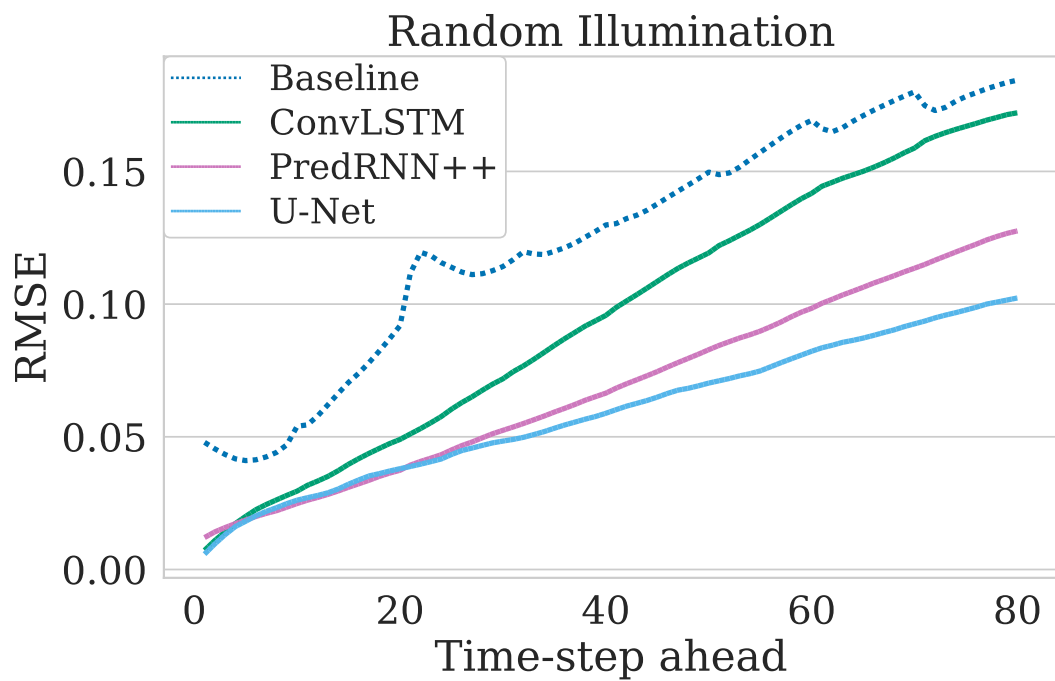


(c) Double Drop Dataset Results

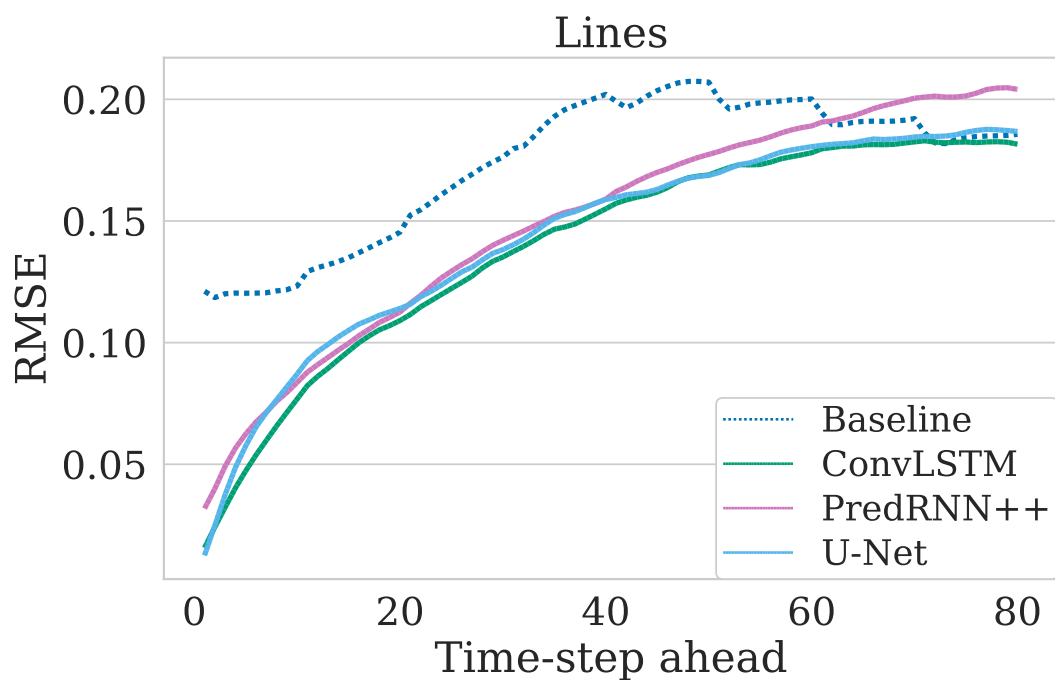


(d) Illumination Angle 135° Dataset Results

Figure 7.1: (Continued from previous page)

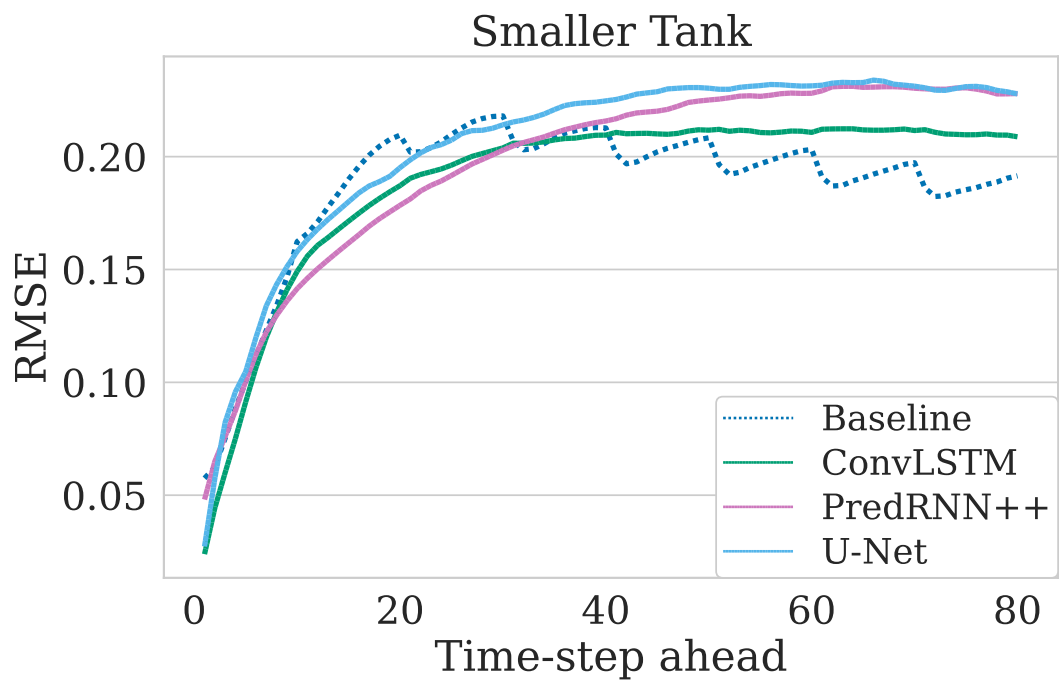


(e) Random Illumination Dataset Results

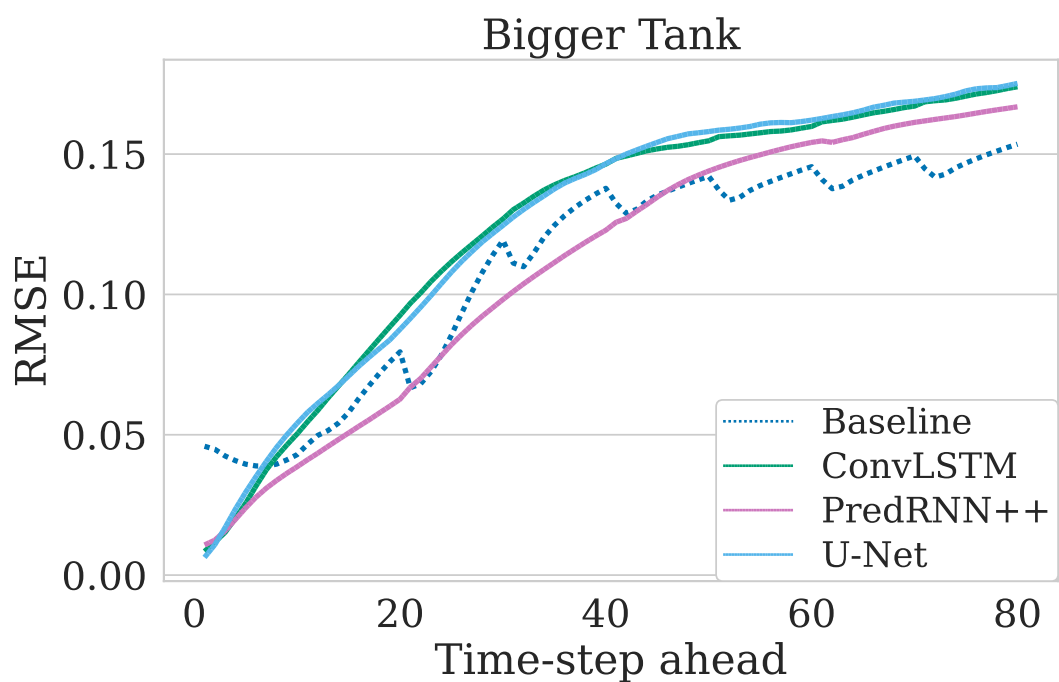


(f) Lines Dataset Results

Figure 7.1: (Continued from previous page)

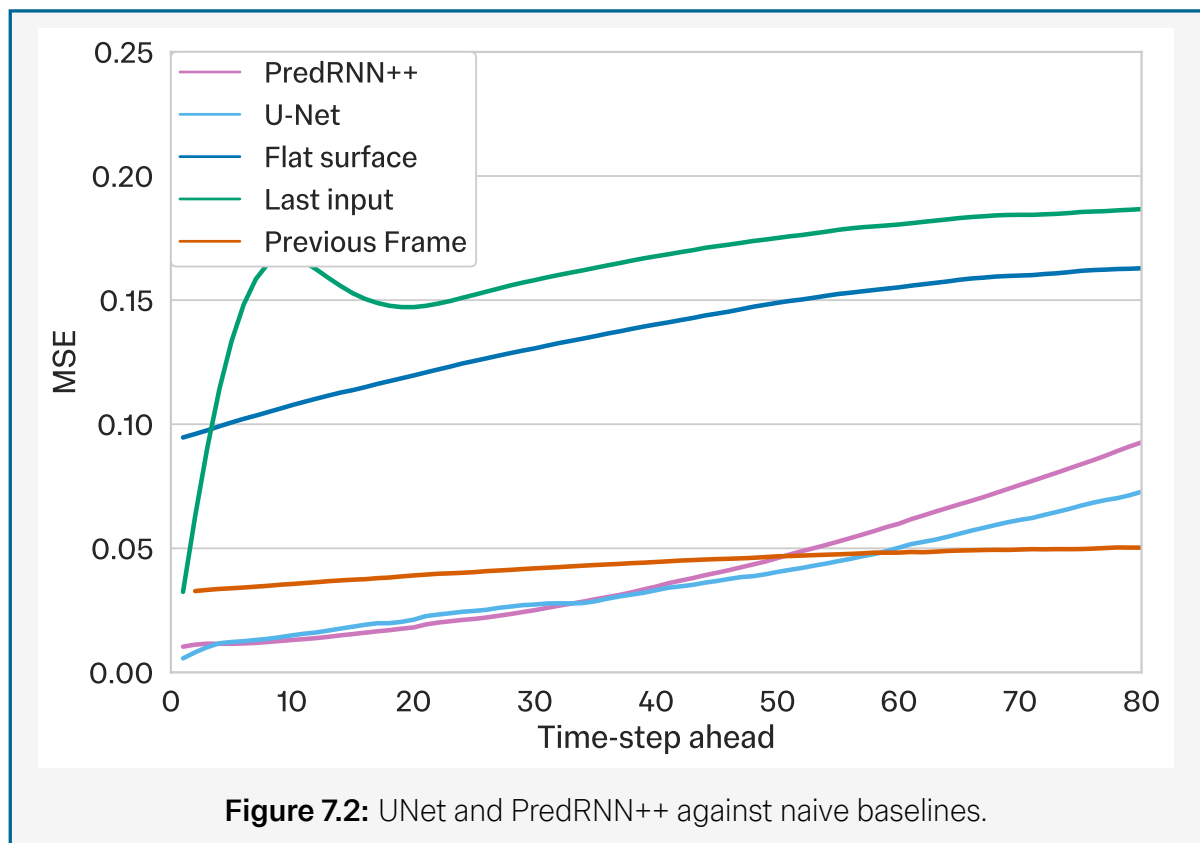


(g) Smaller Tank Dataset Results



(h) Bigger Tank Dataset Results (Duplicate)

Figure 7.1: (Continued from previous page)



7.3 Disentanglement Results

