Learning World Models With Hierarchical Temporal Abstractions: A Probabilistic Perspective

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von
Vaisakh Shaj Kumar
aus Kerala (Indien)

Tag der mündlichen Prüfung: 19. November 2024

1. Referent: Prof. Dr. Gerhard Neumann
2. Referent: Prof. Dr. Kristian Kersting

Abstract

Machines that can replicate human intelligence with type 2 (Daniel 2017) reasoning capabilities should be able to reason at multiple levels of spatio-temporal abstractions and scales using internal world models (K. Friston 2008; LeCun 2022). Devising formalisms to develop such internal world models, which accurately reflect the causal hierarchies inherent in the dynamics of the real world, is a critical research challenge in the domains of artificial intelligence and machine learning. This thesis identifies several limitations with the prevalent use of state space models (SSMs) as internal world models and propose two new probabilistic formalisms namely Hidden-Parameter SSMs and Multi-Time Scale SSMs to address these drawbacks. The structure of graphical models in both formalisms facilitates scalable exact probabilistic inference using belief propagation, as well as end-to-end learning via backpropagation through time. This approach permits the development of scalable, adaptive hierarchical world models capable of representing nonstationary dynamics across multiple temporal abstractions and scales. Moreover, these probabilistic formalisms integrate the concept of uncertainty in world states, thus improving the system's capacity to emulate the stochastic nature of the real world and quantify the confidence in its predictions. The thesis also discuss how these formalisms are in line with related neuroscience literature on Bayesian brain hypothesis and predicitive processing. Our experiments on various real and simulated robots demonstrate that our formalisms can match and in many cases exceed the performance of contemporary transformer variants in making long-range future predictions. We conclude the thesis by reflecting on the limitations of our current models and suggesting directions for future research.

Abstract

Maschinen, die menschliche Intelligenz mit Fähigkeiten zur Schlussfolgerung des Typs 2 (Daniel 2017) replizieren können, sollten in der Lage sein, auf mehreren Ebenen räumlich-zeitlicher Abstraktionen und Maßstäbe zu schlussfolgern, indem sie interne Weltmodelle verwenden (K. Friston 2008; LeCun 2022). Die Entwicklung von Formalismen zur Entwicklung solcher internen Weltmodelle, die die in der Dynamik der realen Welt inhärenten kausalen Hierarchien genau widerspiegeln, stellt eine entscheidende Forschungsherausforderung in den Bereichen Künstliche Intelligenz und maschinelles Lernen dar. Diese Dissertation identifiziert mehrere Einschränkungen bei der verbreiteten Verwendung von Zustandsraummodellen (SSMs) als interne Weltmodelle und schlägt zwei neue probabilistische Formalismen vor, nämlich Versteckte-Parameter-SSMs und Mehr-Zeitskalen-SSMs, um diese Nachteile anzugehen. Die Struktur der grafischen Modelle in beiden Formalismen ermöglicht skalierbare exakte probabilistische Inferenz mittels Belief Propagation sowie End-to-End-Lernen durch Backpropagation durch die Zeit. Dieser Ansatz ermöglicht die Entwicklung skalierbarer, adaptiver hierarchischer Weltmodelle, die nichtstationäre Dynamiken über mehrere zeitliche Abstraktionen und Maßstäbe darstellen können. Darüber hinaus integrieren diese probabilistischen Formalismen das Konzept der Unsicherheit in Weltzuständen und verbessern somit die Fähigkeit des Systems, die stochastische Natur der realen Welt nachzuahmen und das Vertrauen in seine Vorhersagen zu quantifizieren. Die Arbeit diskutiert auch, wie diese Formalismen mit der verwandten neurowissenschaftlichen Literatur zur Hypothese des Bayesianischen Gehirns und zur prädiktiven Verarbeitung übereinstimmen. Unsere Experimente mit verschiedenen realen und simulierten Robotern zeigen, dass unsere Formalismen in vielen Fällen die Leistung zeitgenössischer Transformer-Varianten bei der Vorhersage langfristiger Zukünfte übertreffen können. Wir schließen die Dissertation mit einer Reflexion über die Grenzen unserer aktuellen Modelle und Vorschlägen für zukünftige Forschungsrichtungen ab.

Acknowledgement

This thesis is the result of the support of several people, to whom I am extremely grateful. I would first like to express my thanks and gratitude to my supervisor, Prof. Gerhard Neumann, for providing me the opportunity to work with him and his consistent mentoring over the last 5 years. His mentorship has helped shape my approach to research, introducing me to a probabilistic way of thinking and emphasizing the importance of rigor. He is one of the most approachable people I have met in my life and carries himself with a deep sense of humility and patience, qualities that continue to inspire me both professionally and personally.

I am also thankful for the guidance and support from several other mentors across various projects during my PhD. I extend my gratitude to Professor Marc Hanheide, Dr. Dieter Buchler, Harit Pandya, Philipp Becker, Aravind Srinivas, Ozan Demir and Niels van Duijkeren, who each played a crucial role at different stages of my doctoral journey.

My heartfelt thanks go to our administrative secretary, Christine Brand, for her kind support over the PhD, which made navigating daily challenges as a non-German speaking foreigner much simpler.

One of the highlights of my PhD journey was being able to be part of two separate research groups, at KIT in Germany and the University of Lincoln in the UK. This provided me an incredible opportunity to immerse myself in two distinct cultures. I am indebted to the several brilliant, kind, and inclusive colleagues at both these research groups with whom I had the privilege of sharing many hikes, retreats, ski trips, dinners, coffee breaks, and gossip sessions. The technical debates I had with them have significantly contributed to molding my thought process as a researcher. I would like to extend special thanks to Harit Pandya, Riccardo, Philipp Becker, Onur, Bruce (Ge Li), and Max, whose friendship and support made the initial phases of my PhD journey and the transition between two countries significantly easier.

Special thanks to my friend Sharvin Raphael, who assisted with the Photoshop-based graphics used in this thesis, enhancing the visual presentation of my research.

It was also a privilege to work with and mentor several brilliant students (Andreas Boltres, Martin Herault, Moritz Reuss, Rohit Sonker, Ruben Jacob, Stefan Geyer, Shuheng Zhang, Emiliyan Gospodinov, Thorben Comes, among others). The discussions with them were invaluable, significantly influencing my thought process. Additionally, working with them helped me discover my passion for mentoring and teaching, for which I am immensely grateful.

I would like to thank my parents, brother, friends and family in India and abroad, whose constant support and encouragement were crucial to my pursuit and perseverance in this ambitious endeavor of moving to Europe for my PhD. Finally, special thanks to Kevin for making my experience in Germany rich and colourful.

Contents

List of Figures

List of Tables

1. Introduction

The human brain maintains an intricate internal mental model of the world, a sophisticated representation that allows us to navigate complex environments, predict outcomes, and learn from interactions with minimal explicit instruction. This concept of "World Models" (Sutton 1995; Ha and Schmidhuber 2018; LeCun 2022) in machine learning research draws direct inspiration from our cognitive processes (K. Friston 2005; K. Friston 2008), aiming to endow AI systems with a similar ability to abstract, understand, and anticipate the dynamics of their surroundings, a process that remains challenging for current AI systems. Research on foundational world models is pivotal for advancing beyond the current limitations of machine learning, where systems often require vast datasets and extensive training to perform tasks that humans manage with ease.

Recent advances in transformer (Vaswani et al. 2017) based large language models (LLMs) like BERT (Devlin et al. 2018), GPT-3,4 (Brown et al. 2020; Kocoń et al. 2023) etc have impressed the AI community due to their remarkable ability to generate coherent text, translate languages, and even produce code. However, the perception that LLMs might be the definitive pathway to AGI overlooks critical nuances. At their core, LLMs excel in pattern recognition and statistical inference from vast datasets, but lack an understanding of causality, physical principles, and reasoning, essential components of human-like intelligence. Recent research (Zečević et al. 2023) on the causal nature of LLMs argues that they are heavily based on correlational statistics and do not capture the causal dynamics essential for understanding and interacting with the real world. By focusing on learning the causal structure of the world and enabling machines to simulate and predict future states, researchers can create more autonomous, efficient, and adaptable AI systems. Such a research path represents a measured step towards developing AI systems with a deeper, more intuitive grasp of the world, aligning machine learning processes more closely with natural intelligence. Advancements in learning a foundational world model promise profound implications in several fields, including autonomous vehicles, robotics, healthcare, environmental modeling, and beyond, marking a crucial step towards achieving AI that can truly understand and navigate the world as humans do (Pezzulo, Rigoli, and K. J. Friston 2018; LeCun 2022).

When discussing the foundational world model (FWM), I refer to a model (or a set of models) that embodies the following characteristics expanding on the definition of T. Gupta et al. 2024.

(i) (Representation) Conceptually understand the components, structures, and interaction dynamics within a given system at different levels of abstractions (Lee and

Mumford 2003; K. Friston 2008; Pezzulo, Rigoli, and K. J. Friston 2018; LeCun 2022);

(ii) (Veridicality) Quantitatively model the underlying laws of such a system, that enables accurate predictions of counterfactual consequences of interventions/actions;

(iii) (Probabilistic) Model uncertainties (aleatoric and epistemic) via probabilistic representations. This ensures that the FWM can handle the stochastic nature of the real world, quantify confidence in its predictions and imagine diverse plausible futures.

(iv) (Foundational) Being able to generalize and scale (i), (ii) and (iii) across diverse systems or domains encountered in the world.

The definition of FWM in T. Gupta et al. 2024 is expanded to explicitly include the "probabilistic" property. The reason for this is elaborated further in Section 1.1. Furthermore, Section 3.1 provides a reasoning for this from a computational neuroscience point of view.

1.1. Thesis Problem Statement

In this dissertation, I explore the difficulties associated with existing world model formalisms to fit into the notion of FWM and introduce the formalism(s) designed to tackle three challenges in developing foundational models of the world, as outlined below.

1. ***How can we model the world with a principled probabilistic formalism that also quantifies uncertainties in predictions in a scalable manner?***
   The real world involves a significant amount of uncertainty. It is intrinsically stochastic (aleatoric uncertainty), and we are often uncertain about the true state of the system because our observations about it are partial (epistemic uncertainty), or the prediction accuracy of the world model is imperfect due to limited training data, representational power, or computational constraints. This necessitates probabilistic world models that can imagine multiple plausible future scenarios and the associated uncertainty. Furthermore, the inference algorithms on these probabilistic models/representations should scale well for large datasets and long sequences.

2. ***How can we make sure the model adapts to changing dynamics / non-stationary situations?***
   In various real-world control scenarios, agents are presented with tasks that may exhibit different dynamics due to changing environmental factors or task requirements. For instance, a robot engaged in playing table tennis might use rackets of differing weights or sizes, or an agent tasked with bottle manipulation might deal with varying fluid levels. Traditional models, operating on a singular time scale or with flat world representations, struggle to accommodate these dynamic shifts.

Therefore, it's essential to develop formalisms capable of inferring task abstractions for seamless adaptability in multi-task environments.

3. ***Can we model the world at multiple temporal abstractions and time scales?***
   One important dimension of world models is the level of temporal granularity/time scale at which they operate. Existing literature on world models operates at a single level of temporal abstraction, typically at a fine-grained level such as milliseconds. One drawback of single-time scale world models is that they may not capture longer-term trends and patterns in the data. For efficient long-horizon prediction, reasoning, and planning, the model needs to predict at multiple levels of temporal abstractions.

The challenges outlined here are not exhaustive of all the difficulties encountered in developing foundational world models (FWMs). Instead, they specifically target three central aspects that are vital for the successful creation of FWMs.

1.2. Thesis Contribution

This thesis introduces a systematic approach to address the said challenges in a principled probabilistic framework through the development of three formalisms, each building on the other. Each formalism exhibits increasing complexity and expressiveness, with the incorporation of one utility at a time, culminating in a final model that combines the favorable properties of all three. The sequential presentation also ensures a comprehensive understanding of the progression from the foundational state space models (SSMs) to the sophisticated multi-time scale SSMs, showcasing a systematic evolution in complexity and capability.

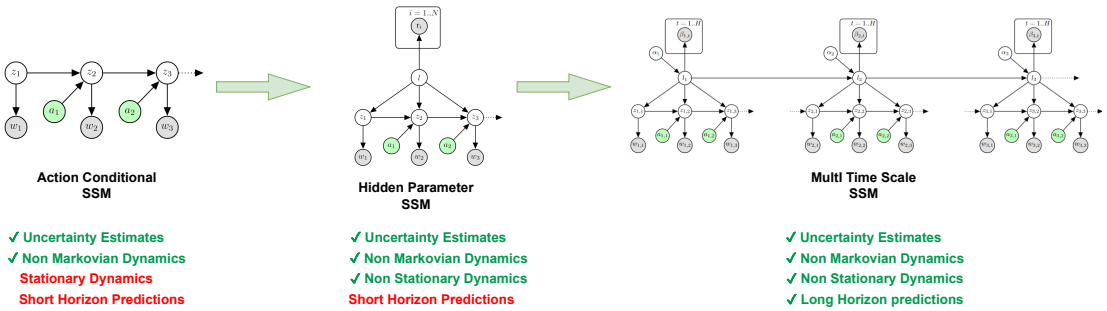


Figure 1.1.: Evolution of thesis represented in terms of the evolution of the probabilistic graphical models for each world model formalism. All formalisms infer the latent states via exact inference resulting in closed-form update rules.

The main contributions that culminate from this thesis are (i) identifying the drawbacks of the widely used SSM formalism and (ii) introducing two new formalisms / declarative representations (Koller and Friedman 2009) called HiP-SSM and Multi-Time Scale SSM to address these drawbacks. The thesis also proposes algorithms to perform scalable exact inference via message passing for forward predictions in these models. Finally, it is demonstrated that end-to-end learning is possible in all of the proposed models using a single loss function and backpropagation through time.

1.3. Thesis Outline

The remainder of this thesis is structured as follows. A visual overview of the thesis is given in Figure 1.2.

In Chapter 2, we give a list of preliminaries that will assist readers in navigating the rest of the document. This chapter begins with a concise introduction to Probabilistic Graphical Models (PGMs), a core framework utilized throughout this thesis for formalizing representations of the world dynamics and performing learning and inference in these. Additionally, we introduce fundamental concepts, including deep state space models (SSMs) and Bayesian aggregation schemes.

Chapter 3 reviews the existing literature on world models, drawing from two distinct areas: computational neuroscience and machine learning. The segment on computational neuroscience aims to highlight connections between the representations explored in this thesis and the conceptualizations of world models in the human brain set forth by the neuroscience community. Additionally, we succinctly review various methodologies adopted by the machine learning community for building world models, providing a backdrop for the thesis's further discussions.

In Chapter 4, we use the well-established formalism of state space models (SSMs) for learning single time-scale World Models. We critically analyze and extend a recent deep Kalman model (deep Gaussian SSM) known as recurrent Kalman networks (RKNs) (Becker, Pandya, et al. 2019) for the specific objective of modeling world dynamics. A significant limitation identified in the original models was their inability to systematically include control or action input in the latent dynamics. This chapter suggests a methodologically sound approach to embed action signals into state space models that respect the causal relation with the inferred latent states. Conditioning actions in a manner that adheres to causal relationships is essential, particularly for decision-making and control tasks. This approach enables precise execution of interventions ("doing") and the simulation of counterfactual scenarios ("imagining") using actions or control signals. The hypothesis is validated by modelling the dynamics of a variety of real robots with non-markovian but stationary dynamics.

In Chapter 5, we focus on the aspect of "adaptability" of world models within dynamic, non-stationary environments. Traditional State Space Models (SSMs) typically operate

under the assumption of static dynamics, a presumption that falls short in the complexity of real-world applications. Recognizing the challenge of developing distinct SSMs for all possible variation in dynamics, we introduce the formalism of Hidden Parameter State Space Models (HiP-SSMs). HiP-SSMs leverage hierarchical latent task abstractions to infer the causal factors behind environmental non-stationarity, facilitating seamless adaptation to changing conditions. Moreover, we detail a simple and computationally efficient method for learning and inference in this Gaussian graphical model that avoids approximations like variational inference.

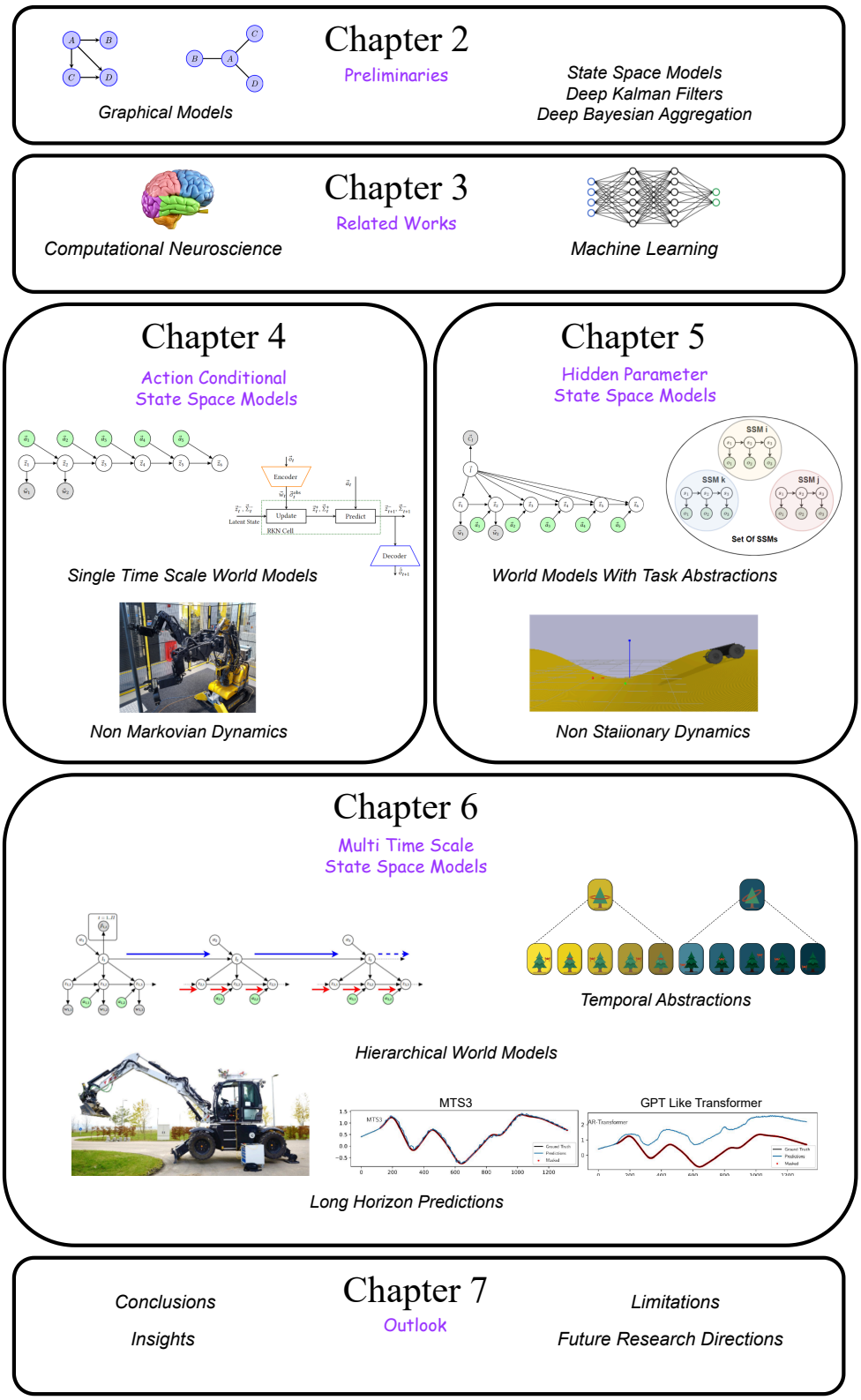


Figure 1.2.: A visual overview of the thesis.

In Chapter 6, the thesis explores the notion of "temporal depth" in the world dynamics via causal hierarchies. Learning world models with hierarchical temporal abstractions and multiple time scales is a largely unaddressed problem in existing literature for world models (deterministic or stochastic). We come up with a principled probabilistic formalism for learning such multi-time scale world models as a hierarchical sequential latent variable model called multi-time scale state space models (MTS3). MTS3 maintains all the advantages of the previous formalisms (SSM and HiP-SSM) for learning world models but can additionally make hierarchical long-horizon predictions. We also derive computationally efficient inference schemes on multiple time scales for highly accurate long-horizon predictions and uncertainty estimates spanning several seconds into the future. We show these lightweight hierarchical linear models can compete with state of the art transformers on long-horizon predictions on continuous dynamical systems.

Chapter 7 concludes the thesis by summarizing the findings, outlining the limitations, and directions for future research.

2. Preliminaries

2.1. Probabilistic Graphical Models

Probabilistic Graphical Models (PGMs) (Jordan 2004; Koller, Friedman, et al. 2007) use a graph-based representation as the basis for compactly encoding a complex distribution over a high-dimensional space. In various applied fields including bioinformatics, speech processing, image processing and control theory, statistical models have long been formulated in terms of graphs, and algorithms for computing basic statistical quantities such as likelihoods and score functions have often been expressed in terms of recursions operating on these graphs; examples include phylogenies, pedigrees, hidden Markov models, Markov random fields, and Kalman filters. These ideas can be understood, unified, and generalized within the formalism of graphical models. Indeed, graphical models provide a natural tool for formulating variations on these classical architectures, as well as for exploring entirely new families of statistical models.

The intuitive formalisms via PGMs enable effective communication between scientists across the mathematical divide by fostering substantive debate in the context of a scientific problem and ultimately facilitate the joint development of statistical and computational tools for quantitative data analysis. Consequently, in fields that involve the study of large numbers of interacting variables, graphical models are increasingly in evidence.

2.1.1. Structure Of Graph and Independencies

There is a dual perspective that can be used to interpret the structure of the graph in PGMs.

1. **The graph as a representation of a set of independencies:** From one perspective, the graph is a compact representation of a set of independencies that hold in the distribution; these properties take the form X is independent of $\boldsymbol{Y}$ given Z, denoted $\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}$), for some subsets of variables $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$.
2. **The graph as a skeleton for factorizing a distribution:** The other perspective is that the graph defines a skeleton for compactly representing a high-dimensional distribution: Rather than encode the probability of every possible assignment to all of the variables in our domain, we can "break up" the distribution into smaller factors, each over a much smaller space of possibilities.

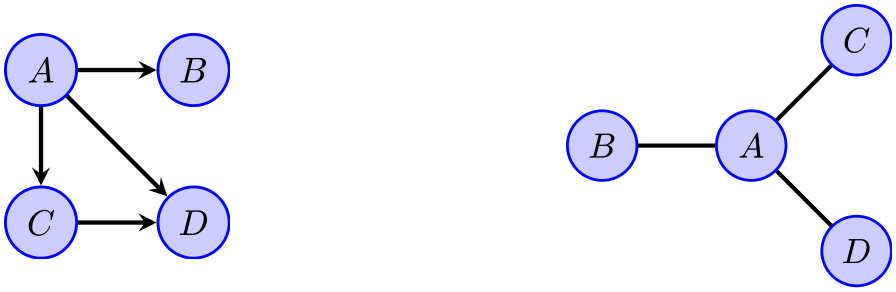


Figure 2.1.: (left) A Bayesian Network, where the directed edges indicate causal relationships. (right) A Markov network, where the undirected edges give a notion of correlation/affinity

The two perspectives are, in a deep sense, equivalent. The independence properties of the distribution are precisely what allow it to be represented compactly in a factorized form. Conversely, a particular factorization of the distribution guarantees that certain independencies hold.

2.1.2. Representation, Inference and Learning in PGMs

2.1.2.1. Representation

As detailed in section 2.1.1, the representation of distributions in graphical language exploits structure that appears in many distributions that we want to encode in practice. The graph structure allows us to factorize a high-dimensional joint distribution into a product of lower-dimensional CPDs or factors that exploit conditional independencies. This framework has many advantages, as listed below.

1. It often allows the distribution to be written down tractably, even in cases where the explicit representation of the joint distribution is astronomically large.

2. A human expert can understand and evaluate its semantics and properties. This property is important for constructing models that provide an accurate reflection of our understanding of a domain.

PGMs can be broadly classified into 2 families based on the types of their graphical structure, Bayesian and Markov networks. **Bayesian Networks** uses a directed graph that captures causal relationships between variables. **Markov Networks** uses an undirected graph and captures correlations between variables. Both representations provide the duality of independencies and factorization, but they differ in the set of independencies they can encode and in the factorization of the distribution that they induce.

In this thesis, we come up with formalisms for internal mental models of the world/environment dynamics using these graph-based representations, specifically Bayesian Networks.

2.1.2.2. Inference

The same structure often also allows the distribution to be used effectively for inference, answering queries using the distribution as our model of the world. In particular, there exist algorithms for computing the posterior probability of some variables given evidence of others. These inference algorithms work directly on the graph structure and are generally orders of magnitude faster than manipulating the joint distribution explicitly.

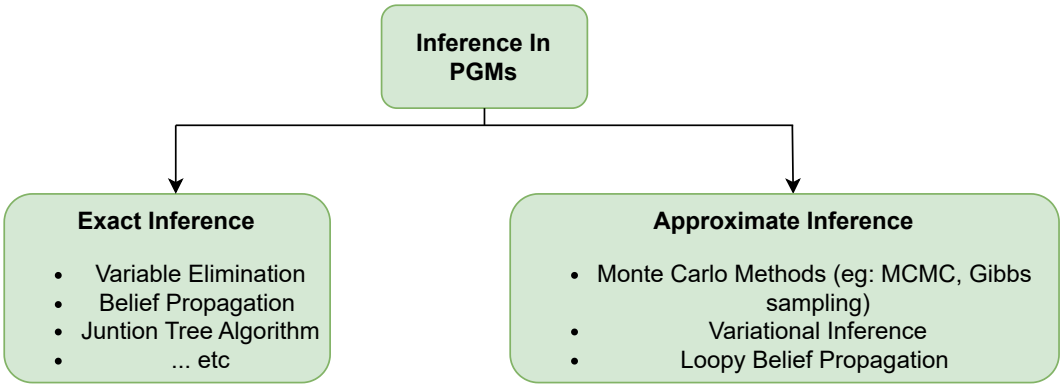


Figure 2.2.: A summary of standard inference routines on PGMs.

This thesis perform exact inference via closed form messages (belief propagation) to answer several such queries on the proposed graphical representation of the world model.

2.1.2.3. Learning

PGM framework facilitates the effective construction of approximate models by learning from past experience (in the form of observed data). In this approach, a human expert provides some rough guidelines on how to model a given domain. For example, the human usually specifies the attributes that the model should contain, often some of the main dependencies that it should encode, and perhaps other aspects. However, the details are usually filled in automatically by fitting the model to the data. The models produced by this process are usually much better reflections of the domain than models that are purely hand-constructed. In addition, they can sometimes reveal surprising connections between variables and provide novel insights into a domain. Learning in

PGMs can be broadly classified into (i) structure learning and (ii) parameter learning. **Structure learning** focuses on discovering the underlying causal or dependence structure between the variables in the model. It essentially answers the question "What variables influence each other?". **Parameter learning**, on the other hand, focuses on estimating the quantitative relationships between variables based on the discovered structure. It essentially answers the question "How strong are these relationships?".

This thesis proposes end-to-end parameter learning strategies for the Bayesian Network representations from data via backpropagation.

2.1.3. Predictive World Models As Inference In PGMs

In this thesis, we use PGM as a tool to model the causal dynamics of the world in a probabilistically principled manner. In fact, if a particular learning problem can be set up as a probabilistic graphical model, this can often serve as the first and most important step in solving it. The PGM framework offers us the flexibility that once we write down the model and pose the question, the objectives for learning and inference emerge automatically.

To set up models that reflect the causal structure of our world/environment dynamics, we choose directed graphical models (Bayesian Networks) as declarative graphical **representations**. We can then effectively use the representation to answer a wide range of questions that are of interest using well-studied and efficient **inference** algorithms. These queries may include (i) "state queries" that seek answers about the past, present, and future world states, and (ii) "causal queries" such as intervention and counterfactual queries. In this thesis, we perform both "state queries" and "causal queries". The "causal queries" primarily take the form of action-conditional future predictions. Finally, for those parameters that are unknown to the human expert, the principled **learning** objectives can be used to learn the parameters from the observed data. A summary of the schemes used in the thesis is given in Table 2.1.

Table 2.1.: The summary of the representation, inference and learning schemes used in the thesis for learning probabilistic world models.

	Types	Used In Thesis
Representation	Bayesian Network Markov Network	✓
Inference	Exact Inference Approximate Inference	✓
Learning	Parameter Learning Structural Learning Backpropagation (Gradient Descent) Expectation Maximization	✓ ✓

We start with the most widely used and studied PGM for modelling the world, commonly referred to as State Space Models (SSMs). This formalism will form the basis for all other formalisms developed in this thesis.

2.2. State Space Models

State space models (SSMs) are Bayesian probabilistic graphical models (Koller and Friedman 2009; Jordan 2004) that are popular for learning patterns and predicting behaviour in sequential data and dynamical systems as shown in Figure 2.3. Many sequential models including Kalman Filters and Hidden Markov Models (HMMs) though can be treated as variants of state space models. Kalman filters are essentially continuous counterparts of HMMs where the hidden states $\boldsymbol{z_t}$ are modeled as Gaussian distributions.

Formally, we define a state space model as a tuple $(\mathcal{Z}, \mathcal{A}, \mathcal{O}, f, h, \Delta t)$, where $\mathcal{Z}$ is the state space, $\mathcal{A}$ the action space and $\mathcal{O}$ the observation space of the SSM. The parameter Δt denotes the discretization time step and f and h the dynamics and observation models, respectively.

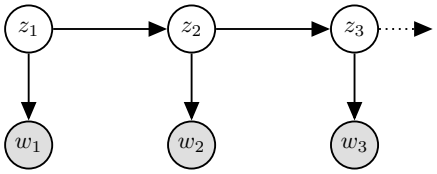


Figure 2.3.: PGM for a State Space Model (SSM).

We will consider the Gaussian state space model that is represented using the following equations [c] $\boldsymbol{z}_t = f(\boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1}) + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}^{\mathbf{trans}})$,
$\boldsymbol{o}_t = h(\boldsymbol{z}_t) + \boldsymbol{v}_t, \quad \boldsymbol{v}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}^{\mathbf{obs}})$. Here $\boldsymbol{z}_t \in \mathcal{Z}$, $\boldsymbol{a}_t \in \mathcal{A}$ and $\boldsymbol{o}_t \in \mathcal{O}$ are the latent states, actions, and observations at time t. The vectors $\boldsymbol{\epsilon}_t$ and $\boldsymbol{v}_t$ denote zero-mean Gaussian transition and observation noise, respectively.

2.2.1. Linear Gaussian SSMs (Kalman Filters).

For SSMs defined in 2.2, when f and h are linear/locally linear, inference can be performed efficiently via exact inference. Such SSMs are referred to as Kalman Filters, which were developed independently of PGM Literature. The Kalman filter (Rudolph E Kalman and Bucy 1961; Srkk 2013) works by iteratively answering two queries, (i) estimating the prior marginal (predict step) and (ii) updating the posterior given current observations (update step), which are detailed below:

2.2.1.1. Predict Step

During the prediction step the transition model $\mathbf{A}$ is used to infer the current prior state marginal $p(\boldsymbol{z}_t|\boldsymbol{w}_{1:t-1}) = \mathcal{N}(\boldsymbol{z}_t^-, \boldsymbol{\Sigma}_t^-)$, that is, a priori to the observation, from the

previous posterior estimate $p(\boldsymbol{z}_{t-1}|\boldsymbol{w}_{1:t-1}) = \mathcal{N}\left(\boldsymbol{z}_{t-1}^{+}, \boldsymbol{\Sigma}_{t-1}^{+}\right)$. The marginal computation is given as follows:

$$p(\boldsymbol{z}_t|\boldsymbol{w}_{1:t-1}) = \mathcal{N}(\boldsymbol{z}_t^{-}, \boldsymbol{\Sigma}_t^{-}) = \int p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) p(\boldsymbol{z}_{t-1}|\boldsymbol{w}_{1:t-1}) d\boldsymbol{z}_{t-1}. \tag{2.1}$$

This is an instance of forward belief propagation or variable elimination in the chain-structured graphical model (Koller, Friedman, et al. 2007). Due to the Gaussian linear assumption, the parameters of this marginal can be estimated in closed form as follows:

$$\boldsymbol{z}_t^{-} = \mathbf{A}\boldsymbol{z}_{t-1}^{+} and \quad \boldsymbol{\Sigma}_t^{-} = \mathbf{A}\boldsymbol{\Sigma}_{t-1}^{+}\mathbf{A}^T + \boldsymbol{\Sigma}^{\text{trans}}. \tag{2.2}$$

2.2.1.2. Update Step

The prior estimate is then updated using the current observation $\boldsymbol{w_t}$ and the linear observation model $\mathbf{H}$ to obtain the posterior estimate $\left(\boldsymbol{z}_t^{+}, \boldsymbol{\Sigma}_t^{+}\right)$ using the Bayes rule.

Since we have Gaussian assumptions on all random variables and a linear observation model, the posterior estimate $\left(\boldsymbol{z}_t^{+}, \boldsymbol{\Sigma}_t^{+}\right)$ can be obtained in closed form as:

$$\boldsymbol{z}_t^{+} = \boldsymbol{z}_t^{-} + \mathbf{Q}_t\left(\boldsymbol{w}_t - \mathbf{H}\boldsymbol{z}_t^{-}\right), \boldsymbol{\Sigma}_t^{+} = \left(\mathbf{I} - \mathbf{Q}_t\mathbf{H}\right)\boldsymbol{\Sigma}_t^{-}, with \quad \mathbf{Q}_t = \boldsymbol{\Sigma}_t^{-}\mathbf{H}^T\left(\mathbf{H}\boldsymbol{\Sigma}_t^{-}\mathbf{H}^T + \boldsymbol{\Sigma}^{\text{obs}}\right)^{-1}, \tag{2.3}$$

where $\mathbf{I}$ denotes the identity matrix. The matrix $\mathbf{Q}_t$ is called the Kalman gain.

The whole update step can be interpreted as a weighted average between the state and the observation estimate, where the weighting, that is, $\mathbf{Q}_t$, depends on the uncertainty about those estimates.

We refer to Srkk 2013 for detailed derivations of the Kalman filtering equations. The derivations can also be obtained using the sum-product algorithm in a factor graph of the state-space model in Figure 2.3.

2.2.2. Deep Kalman Filters

To model dynamics under partially observable scenarios, state-space models, in particular the Kalman filter (Rudolph Emil Kalman 1960), have recently been integrated with deep learning, by modeling dynamics in learned latent space. We discuss two such works (Haarnoja et al. 2016; Becker, Pandya, et al. 2019), that borrow concepts from deep learning and graphical model communities, where the architecture of the network is informed by the structure of the probabilistic state estimator. The various exact inference schemes in the learned latent spaces derived and executed in the thesis are motivated by this work.

2.2.2.1. Backprop Kalman Filter

The BackpropKF (Haarnoja et al. 2016) uses a deep convolutional encoder to encode a high-dimensional observation $\boldsymbol{o}$ to a latent observation $\boldsymbol{w}_t = \text{enc}_w(\boldsymbol{o}_t)$. In addition to the latent observation, the encoder also learns to output the uncertainty of this observation, i.e., $\boldsymbol{\sigma}_{o_t} = \text{enc}_\sigma(\boldsymbol{o}_t)$. Due to the non-linear observation encoder, a simplified linear Gaussian state space model / (extended) Kalman filter with a known transition model is used in the latent space for state estimation. The parameters of the latent state distribution are directly optimized as a deterministic computation graph via backpropagation through time. This scheme can also be motivated using the Koopman theory (Koopman 1931; Mondal et al. 2023).

2.2.2.2. Recurrent Kalman Networks (RKN)

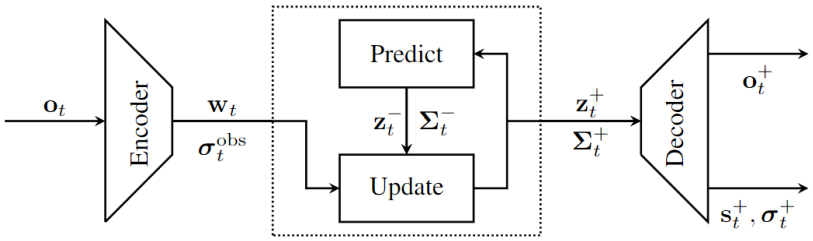


Figure 2.4.: Recurrent Kalman Network (RKN) from Becker, Pandya, et al. 2019. It embeds Kalman filter update equations into the latent space of a deep encoder-decoder network. The Kalman predict step gives the current latent prior $\left(\mathbf{z}_t^-, \boldsymbol{\Sigma}_t^-\right)$using the last posterior $\left(\mathbf{z}_{t-1}^+, \boldsymbol{\Sigma}_{t-1}^+\right)$and subsequently update the prior using the latent observation $\left(\mathbf{w}_t, \boldsymbol{\sigma}_t^{\text{obs}}\right)$. The factorized representation as shown in Figure 2.5 of Σ_t converts matrix inversions to scalar operations. Further it allows splitting the latent state $\mathbf{z}_t$ to the observable units $\mathbf{p}_t$ as well as the corresponding memory units $\mathbf{m}_t$ as discussed in Section 2.2.2.2. Finally, a decoder is tasked to reconstruct the sensory information.

Recurrent Kalman Network (RKN) (Becker, Pandya, et al. 2019), builds upon this idea but learns a locally linear transition model as opposed to a known model as in BackpropKF. Most importantly, the authors come up with clever factorized latent state representation as shown in Figure 2.5, that avoids expensive matrix inversions in the Kalman Update Step. We discuss two important implications of this factorized representation below:

1. **Matrix inversions as scalar operations:** Kalman observation update, which is an instance of Bayesian inversion via conditioning involves high dimensional matrix inversions that are expensive to evaluate and hard to backpropagate for end-to-end learning. Hence, Becker, Pandya, et al. 2019 introduce a factorization of the belief $p(\boldsymbol{z}_t|\boldsymbol{o}_{1:t}, \boldsymbol{a}_{1:t-1}) = \mathcal{N}(\boldsymbol{z}_t^+, \boldsymbol{\Sigma}_t^+)$ such that only the diagonal and one off-diagonal vector of the covariance need to be computed, i.e.

$$\boldsymbol{\Sigma}_t^+ = \begin{bmatrix} \boldsymbol{\Sigma}_t^{u,+} & \boldsymbol{\Sigma}_t^{s,+} \\ \boldsymbol{\Sigma}_t^{s,+} & \boldsymbol{\Sigma}_t^{l,+} \end{bmatrix}, \text{ with } \boldsymbol{\Sigma}_u^+ = \text{diag}(\boldsymbol{\sigma}_t^{u,+}),\ \boldsymbol{\Sigma}_l^+ = \text{diag}(\boldsymbol{\sigma}_t^{l,+})$$

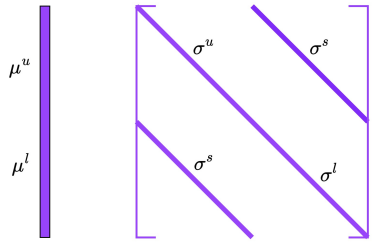


Figure 2.5.: The latent state beliefs (mean μ and covariance Σ) in Becker, Pandya, et al. 2019 is factorized into an upper (observed) and lower (unobserved) part as shown in the figure. The lower part can be dubbed as "memory states" and is assumed to keep a memory of information accumulated over time, because of the specific structure of the observation model. The side covariances (σ^s) are learnt to capture the correlation between the two halves.

and $\mathbf{\Sigma}_s^+ = \text{diag}(\boldsymbol{\sigma}_t^{s,+})$. Using this factorization, the Kalman filter update step described in Equation 2.3 can be executed using only scalar divisions, which are computationally efficient and simpler to implement in back-propagation. Consequently, the closed-form equations for updating the mean can be succinctly expressed using the following scalar equations: $\boldsymbol{z}_t^+ = \boldsymbol{z}_t^- + \begin{bmatrix} \boldsymbol{\sigma}_t^{\text{u},-} \\ \boldsymbol{\sigma}_t^{\text{l},-} \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{w}_t - \boldsymbol{z}_t^{\text{u},-} \\ \boldsymbol{w}_t - \boldsymbol{z}_t^{\text{u},-} \end{bmatrix} \oslash \begin{bmatrix} \boldsymbol{\sigma}_t^{\text{u},-} + \boldsymbol{\sigma}_t^{\text{obs}} \\ \boldsymbol{\sigma}_t^{\text{u},-} + \boldsymbol{\sigma}_t^{\text{obs}} \end{bmatrix}$,

The corresponding equations for the variance update can be expressed as the following scalar operations, $\boldsymbol{\sigma}_t^{\text{u},+} = \boldsymbol{\sigma}_t^{\text{u},-} \odot \boldsymbol{\sigma}_t^{\text{u},-} \oslash \left(\boldsymbol{\sigma}_t^{\text{u},-} + \boldsymbol{\sigma}_t^{\text{obs}}\right)$,
$\boldsymbol{\sigma}_t^{\text{s},+} = \boldsymbol{\sigma}_t^{\text{u},-} \odot \boldsymbol{\sigma}_t^{\text{s},-} \oslash \left(\boldsymbol{\sigma}_t^{\text{u},-} + \boldsymbol{\sigma}_t^{\text{obs}}\right)$,
$\boldsymbol{\sigma}_t^{\text{l},+} = \boldsymbol{\sigma}_t^{\text{l},-} - \boldsymbol{\sigma}_t^{\text{s},-} \odot \boldsymbol{\sigma}_t^{\text{s},-} \oslash \left(\boldsymbol{\sigma}_t^{\text{u},-} + \boldsymbol{\sigma}_t^{\text{obs}}\right)$, where $\odot$ denotes the elementwise vector product and $\oslash$ denotes an elementwise vector division. These factorization assumptions form the basis for the derivations of exact inference routines for all proposed models in this dissertation.

2. **Effective Estimation with Correlated Memory and Observation in Latent Variables:** Choosing the observation model as $\boldsymbol{H} = [\boldsymbol{I}, \boldsymbol{0}]$ allows for the division of the latent state vector into two distinct components. The latent state $\boldsymbol{z}_t = [\boldsymbol{p}_t^T, \boldsymbol{d}_t^T]^T$ has twice the dimensionality of the latent observation $\boldsymbol{w}_t$ and only the first half of the latent state, i.e., $\boldsymbol{p}_t$, can be observed. The upper part utilizes the identity matrix in $\boldsymbol{H} = [\boldsymbol{I}, \boldsymbol{0}]$ to directly extract information from the observations. Meanwhile, the second lower part remains unobservable and is meant to hold information inferred over time, such as velocities in ordinary dynamical systems or images. The key aspect that contributes to the effectiveness of this choice is the selection of the covariance matrix structure discussed previously. The covariance matrix is designed to incorporate both diagonal and off-diagonal elements, ensuring that the correlation between the memory and the observation parts is effectively learned in the off-diagonal part. On the contrary, if we were to utilize a pure diagonal

covariance structure, it would not update the memory units (the later half) or their variance adequately during the Observation/Task/Kalman Update step. Thus, the second half of the latent state, i.e., $\boldsymbol{d}_t$, serves as derivative or velocity units that can be used by the model to estimate the change of the observable part of the latent state.

In this thesis, we rely on the same factorization assumption for inference in all proposed formalisms to account for "memory" accumulated over time.

2.3. Deep Bayesian Aggregation.

To aggregate information from a set of observations into a consistent representation, Volpp et al. 2020 introduce Bayesian aggregation in the context of Meta-Learning and Neural Process (Garnelo et al. 2018) literature. Since the aggregation scheme was introduced in the context of multitask latent variable models, we call this latent variable l the task variable in this section. The generative model for the aggregation of observations n is shown in Figure 2.6. Volpp et al. 2020 derive a closed-form update rule for the posterior $p(\boldsymbol{l}|\boldsymbol{r}_{1:N})$ using the Bayes rule, given an observation model of the form,

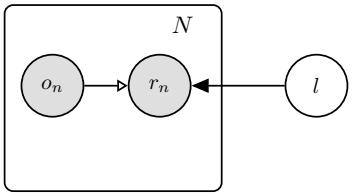


Figure 2.6.: Generative model for the latent task variable (l) inference. The hollow arrows are deterministic transformations leading to an implicit distribution r_n using a set encoder (Zaheer et al. 2017).

$$p(\boldsymbol{r}_n|\boldsymbol{l}) = \mathcal{N}\left(\boldsymbol{r}_n|\boldsymbol{H}\boldsymbol{l}, \operatorname{diag}\left(\boldsymbol{\sigma}_n\right)\right)$$

with $\boldsymbol{H} = \boldsymbol{I}$ and a prior $p(\boldsymbol{l}) = \mathcal{N}(\boldsymbol{\mu}_0, \operatorname{diag}(\boldsymbol{\sigma}_0))$.

The Gaussian assumption allows us to obtain a closed-form solution for the posterior estimate of the latent task variable $p(\boldsymbol{l}|\mathbf{C}_{\boldsymbol{l}})$, based on Gaussian conditioning. The factorization assumption further simplifies this update rule by avoiding computationally expensive matrix inversions into a simpler update rule, as follows,

$$\boldsymbol{\sigma}_{\boldsymbol{l}} = \left((\boldsymbol{\sigma}_0)^{\ominus} + \sum_{n=1}^{N} (\boldsymbol{\sigma}_n)^{\ominus}\right)^{\ominus}, \quad \boldsymbol{\mu}_{\boldsymbol{l}} = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_{\boldsymbol{l}} \odot \sum_{n=1}^{N} (\boldsymbol{r}_n - \boldsymbol{\mu}_0) \oslash \boldsymbol{\sigma}_n. \tag{2.4}$$

Here, $\ominus$, $\odot$, and $\oslash$ denote element-wise inversion, product, and division, respectively.

2.3.1. Using Deep Set Encoders To Learn Observation Uncertainties

Note that the posterior calculation is based on the assumption that we have access to information on the latent (compact) observation $\boldsymbol{r}_n$ and its corresponding uncertainty $\boldsymbol{\sigma}_{o_n}$.

We learn these using a Deep Set encoder as follows, $\mathrm{p}(\boldsymbol{r}_n|\boldsymbol{l}) = \mathcal{N}\left(\boldsymbol{r}_n|\boldsymbol{l}, \mathrm{diag}\left(\boldsymbol{\sigma}_n\right)\right), \quad \boldsymbol{r}_n = \mathrm{enc}_{\boldsymbol{r}}\left(\boldsymbol{x}_n\right), \quad \boldsymbol{\sigma}_n = \mathrm{enc}_{\boldsymbol{\sigma}}\left(\boldsymbol{x}_n\right).$

The architecture of the Bayesian Aggregation Scheme is shown in 2.7.

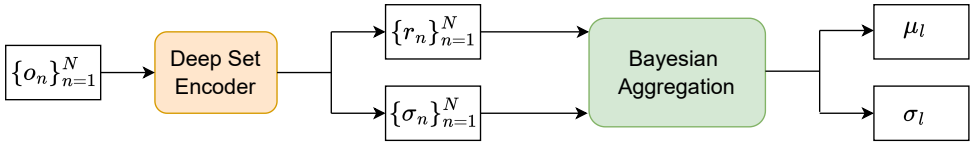


Figure 2.7.: Given a set of N observations, the deep set encoder emits a latent representation for each of the observations and their corresponding uncertainty. The set of latent representations are then aggregated via Bayesian aggregation using update rules 2.4 to obtain a posterior over the abstract latent task variable l, $p(\boldsymbol{l}|\mathbf{o}_{1:N})$. The architecture of the deep set encoder can vary depending on the type of sensory signals aggregated.

2.3.2. Bayesian Aggregation As Probabilistic Attention

Intuitively the mean of the latent task variable $\boldsymbol{\mu_l}$ is a uncertainty weighted sum of the individual latent observations $\boldsymbol{r}_n$, while the variance of the latent task variable $\boldsymbol{\sigma}_{\boldsymbol{l}}^2$ gives the uncertainty of the abstract representation of the task. The derived aggregation rules can be thought of as simple "probabilistic attention," where the learned uncertainty about the individual sensory observations in the set encoder gives the attention weights. This is in line with ideas from computational neuroscience that treat cognitive attention to different sensory signals as a form of "precision weighting" (Hohwy 2013; A. K. Seth 2014). The derived update equations have only a linear computational complexity of $O(N)$, while similar deep set operations (self-attention) in transformers Vaswani et al. 2017 have a complexity of $O(N^2)$.

Note that computing this posterior is a simplified case of the Kalman update rule used in Gaussian SSMs (Becker, Pandya, et al. 2019), with no memory units, $\boldsymbol{H} = \boldsymbol{I}$ and no dynamics.

In this thesis, we explore the concept of Bayesian Aggregation to synthesize temporally abstract representations within generative world models. We also derive, in Chapter 6, generic closed-form update equations for Bayesian aggregation that avoid the restrictive assumptions of observation models being identity matrices, as seen in Volpp et al. 2020.

3. Related Works

In this section, I explore the existing literature on internal-world models from two primary fields: computational neuroscience and machine learning. This dissertation introduces scalable, end-to-end deep learning-based approaches for generative internal-world models, along with their respective inference and learning methods. These approaches aim to implement versions of various internal world models that computational neuroscientists have studied, adapting them within the framework of modern deep learning technologies. The discussions on computational neuroscience also offer insight into why certain representations were chosen for the Bayesian networks outlined in this dissertation from a neuroscientific perspective.

3.1. Computational Neuroscience Literature

3.1.1. The Predictive Bayesian Brain

An increasingly popular theory in cognitive science claims that the brains are essentially prediction machines (Hohwy 2013). The theory is commonly known as the Bayesian brain (Knill and Pouget 2004; Polydoros, Nalpantidis, and Krüger n.d.), predictive processing (Clark 2013), and predictive mind (Hohwy 2013), among others. At its most fundamental, predictive brain theory says that perception is the result of the brain inferring the most likely causes of its sensory inputs by minimizing the difference between actual sensory signals and the signals expected on the basis of continuously updated predictive models. Arguably, predictive processing provides the most complete framework to date for explaining perception, cognition, and action in terms of fundamental theoretical principles and neurocognitive architectures (A. K. Seth 2014; Jiang and Rao 2021).

Helmholtz's Unconscious Inference Bayesian Brain Hypothesis has its origins in Hermann von Helmholtz's notion of unconscious inference (Helmholtz 1948), proposing that perception is a result of inferential processes by the brain, rather than direct imprinting of the external world onto our sensory apparatus. The observer is typically not aware of such prior assumptions but rather, they are incorporated by the neural circuits subconsciously to compute beliefs over hidden causes through the dynamics of neural activities (thereby implementing perception as "unconscious inference").

As part of the internal model, such priors can be expected to be adapted to the environment that the organism lives in. This idea laid the groundwork for the Bayesian brain hypothesis, which posits that the brain computes probabilities to infer the state of the world from ambiguous sensory inputs and prior beliefs, effectively engaging in a form of statistical reasoning or Bayesian inference.

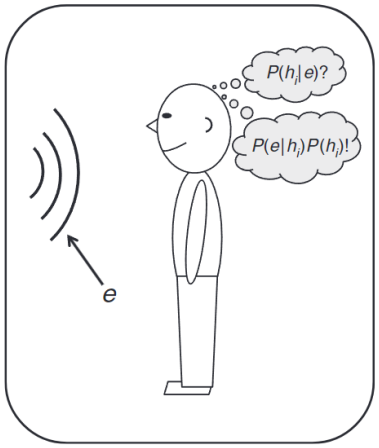


Figure 3.1.: Shows a basic perceptual inference problem of figuring out what caused a sensory signal (sound e). The mechanism is analogous to the situation for the brain(Hohwy 2013; A. K. Seth 2014). Given that the prior probability of hypothesis (cause) is $h_i : P(h_i)$ and the likelihood that the evidence e would occur, given h_i is $P(e \mid h_i)$, the brain computes the posterior probability of hypothesis h_i, given the evidence $e : P(h_i \mid e)$ using internal generative models. The figure shows simplified version of Bayes' rule that puts it together: $P(h_i \mid e) = P(e \mid h_i) P(h_i)$.

Bayesian Brain Hypothesis The Bayesian brain hypothesis (Knill and Pouget 2004) uses Bayesian probability theory to formulate perception as a constructive process based on internal or generative models. The underlying idea is that the brain has a model of the world (Von Helmholtz 2013; MacKay 1956; Neisser 2014; Gregory 1968) that it tries to optimize using sensory inputs (Ballard, G. E. Hinton, and Sejnowski 1983; Kawato, Hayakawa, and Inui 1993; Kersten, Mamassian, and Yuille 2004; Lee and Mumford 2003; K. Friston 2005). This idea is related to analysis by synthesis (Neisser 2014) and epistemological automata (MacKay 1956). In this view, the brain is an inference machine that actively predicts and explains its sensations(Von Helmholtz 2013; Gregory 1980; Dayan, G. E. Hinton, et al. 1995). Central to this hypothesis is a probabilistic model that can generate predictions against which sensory samples are tested to update beliefs about their causes. This generative model is decomposed into a likelihood (the probability of sensory data given their causes) and a prior (the a priori probability of those causes). Perception then becomes the process of inverting the likelihood model (mapping from causes to sensations) to access the posterior probability of the causes, given sensory data (mapping from sensations to causes).

3.1.2. Kalman Filters As Internal World Models / Spatiotemporal Predictive Coding

The world is dynamic; most of the time, animals receive time-varying stimuli either due to their own movement or due to other moving objects in the environment. This makes the ability to predict future stimuli essential for survival (e.g. predicting the location of predators). Rao's seminal work (Rao 1999) from the computational neuroscience literature introduces a foundational mathematical framework that underpins how organisms perceive their environment in a dynamic setting as opposed to static settings discussed previously.

This framework posits that visual perception is a stochastic, dynamic process and the task of perception is one of optimally estimating (in a Bayesian sense) the causes of visual events and on a longer time scale, learning efficient spatio-temporal internal models of the visual environment. The spatiotemporal internal generative model is elegantly represented as a stochastic linear dynamical system similar to a Kalman Filter with transition and observation matrices. By employing inference techniques rooted in Kalman filter update equations, Rao's approach provides a Bayesian optimal method for estimating the internal states that govern the dynamics of the mental world model.

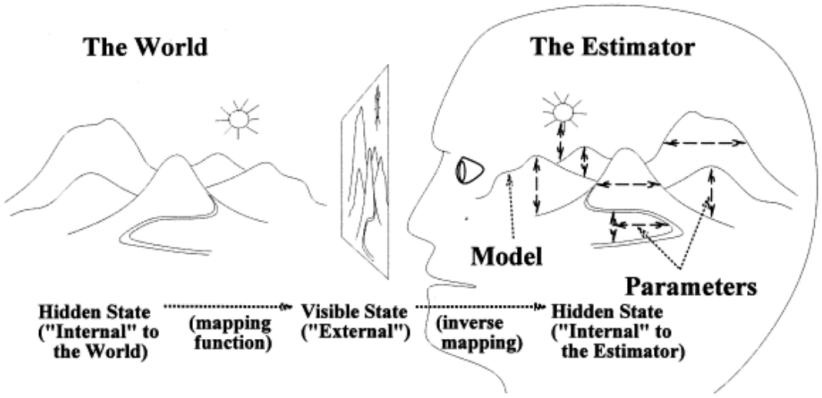


Figure 3.2.: Figure from Rao 1999 presents the challenge organisms face in perceiving the external world through internal models. Without direct access to the world's hidden internal states, organisms rely on sensory measurements to estimate these states, solving the 'inverse' problem to interpret and understand their environment. Though the paper focused on visual systems, the authors argue that the concept extends beyond visual and auditory senses to include internal models for motor systems, emphasizing the broader application of understanding and interacting with the external world through internal representations.

Combining Top-Down Predictions With Bottom-Up Sensory Signals Using Kalman Filter Updates The proposed Kalman-filter model in Rao 1999 provides a mathematical framework that demonstrates how top-down expectations (informed by the internal generative model) interact with bottom-up signals (the current sensory input) to produce a robust estimation of the visual environment. The Kalman Filter combines these two sources of information, weighted according to their reliability (inverse variances or "precisions"), to compute the posterior beliefs about the world in a Bayesian optimal manner, which is interpreted as a form of attention. The proposed model allows for the possibility that the organism or agent might want to perform internal simulations of the dynamics of the external world (e.g., for planning) by predicting how future states evolve given a starting state (and possibly actions).

Learning Of Internal Models The work also derived update rules for learning the transition and precision matrices online based on incoming input data, as opposed to hand-coded dynamics models.

Rao 1999 further provides experimental results that support the model's capability for learning dynamic and static visual stimuli, recognizing objects under various conditions, and segmenting scenes into constituent elements. The work from Rao 1999, also referred to as "spatio-temporal predictive coding" in literature, thus sets a foundational stone for exploring the synergies between computational neuroscience and the development of popular machine learning architectures like Deep Kalman Models.

The neuroscientific insights discussed above are exploited in Chapter 4 of the thesis using modern deep learning techniques.

3.1.3. Adaptive Behavior In Brain With Multiple Internal Models

One of the most notable abilities of biological creatures is their capacity to adapt their behavior to different contexts and environments (i.e., cognitive flexibility) through learning.People can learn to call on various responses depending on the situation—for example, independently move the right and left hands when playing an instrument and speak several different languages. Such multitasking abilities are particularly crucial in communication with several people, who each demand subtly different forms of interaction (Taborsky and Oliveira 2012; Parkinson and Wheatley 2015) and is a key component of exhibiting social intelligence. The work from Isomura, Parr, and K. Friston 2019 attempts to understand how the brain attains cognitive flexibility in different contexts by entertaining the possibility of distinct generative models in a context-sensitive setting. Isomura, Parr, and K. Friston 2019 further present the results of perceptual learning and inference using this form of model selection or structure learning, predicated on an ensemble of generative models. Using this setup, it is shown that Bayesian model averaging provides a plausible account of how multiple hypotheses can be combined to predict the sensorium, while Bayesian model selection enables perceptual categorization and selective learning.

The generative model proposed in Chapter 5 of the dissertation have a similar motivation of using a "set of SSMs/world models" for adaptive behaviour to changing tasks.

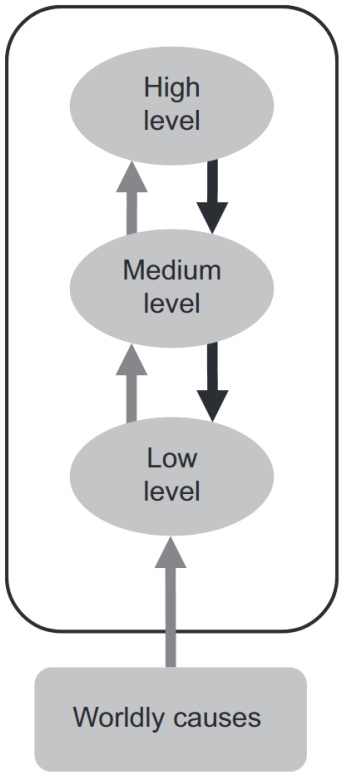


Figure 3.3.: The perceptual hierarchy (Hohwy 2013): Processing of causal regularities at different time scales influence each other in a bottom-up–top-down fashion. Sensory input (dark grey arrows pointing up) is met with prior expectations (black arrows pointing down) and perceptual inference is determined in multiple layers of the hierarchy simultaneously, building up a structured representation of the world. This figure simplifies greatly as there are of course not just three well-defined hierarchical levels in the brain.

3.1.4. Hierarchical Generative Models In The Brain

Researchers in computational neuroscience argue that the previously discussed internal predictive world models are hierarchical in nature (K. Friston 2008; Hohwy 2013; A. K. Seth 2014). The hierarchical nature adeptly mirrors the complex and layered structure of the world itself, capturing the essence of how causes and effects interrelate across different spatial and temporal scales (Hohwy 2013).

These hierarchical generative models overturn classical notions of perception that describe a largely 'bottom-up' process of evidence accumulation or feature detection. Instead, predictive processing proposes that perceptual content is determined by top-down predictive signals emerging from multilayered and hierarchically organized generative models of the causes of sensory signals (Lee and Mumford 2003). These models are continually refined by mismatches (prediction errors) between predicted signals and actual signals across hierarchical levels, which iteratively update predictive models via approximations to Bayesian inference. This means that the brain can induce accurate generative models of hidden environmental causes by operating only on signals to which it has direct access: predictions and prediction errors. It also means that even low-level perceptual content is determined via cascades of predictions flowing from very general abstract expectations, which constrain successively more fine-grained predictions.

The hierarchical generative model proposed in Chapter 6 takes inspiration from these insights.

3.2. Machine Learning Literature

We give a brief overview of notable related works in learning computational mental models of the world in the field of machine learning. A detailed discussion on several of these are provided in Sections 4.1, 5.1 and 6.1.

3.2.1. World Models Based On Gaussain Processes

Bayesian nonparametric models, such as Gaussian processes (GPs), are often the dynamics model of choice in Model-Based RL, especially in low-dimensional problems where data efficiency is critical due their robustness in handling uncertainty (Jus Kocijan et al. 2004; Ko et al. 2007; Nguyen-Tuong, Peters, and Seeger 2008; Grancharova, Juš Kocijan, and Johansen 2008; M. P. Deisenroth, Fox, and Carl Edward Rasmussen 2013; Kamthe and M. Deisenroth 2018).

However, such models introduce additional assumptions on the system, such as the smoothness assumption inherent in GPs with squared-exponential kernels (Carl Edward

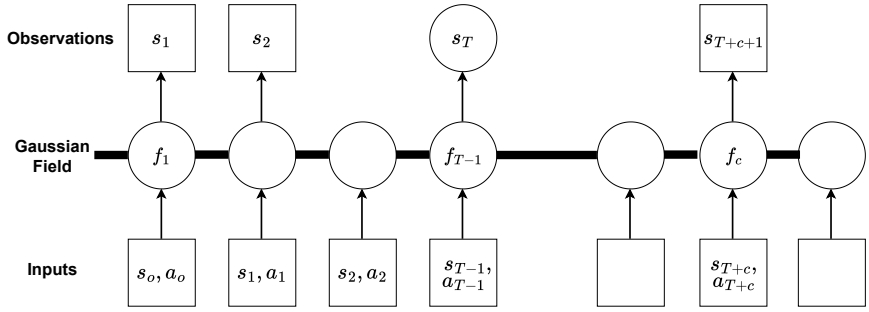


Figure 3.4.: PGM representation adapted from Carl Edward Rasmussen 2003 of a GP regression for learning dynamics. Squares represent observed variables, and circles represent unknowns. The thick horizontal bar represents a set of fully connected nodes and indicates that the forward dynamic function f_i belongs to a Gaussian field. During posterior inference f_i's act as random variables and are integrated out, which means that every prediction y_* depends on all the other inputs and observations (including the training data) making GPs often computationally challenging.

Rasmussen 2003). Also, these methods do not scale to high-dimensional environments. Furthermore, these models are inherently single-time scale and are not widely used in tasks where long-horizon predictions and planning are required.

3.2.2. World Models Based On Neural Networks

The advancements in parametric function approximators, such as neural networks (NNs), brought about by the advent of deep learning, have led to the development of a series of world models employing multi-layer perceptrons. These models are cited in several works, including Baranes and Oudeyer 2013, Fu, Levine, and Abbeel 2016, Punjani and Abbeel 2015, Lenz, Knepper, and Saxena 2015, Gal, McAllister, and Carl Edward Rasmussen 2016, Depeweg et al. 2016, Williams et al. 2017, and Nagabandi, Kahn, et al. 2018. In contrast to Gaussian processes, these deep learning-based models offer constant-time inference and manageable training with large datasets. They also possess the capability to model more intricate functions, including the non-smooth dynamics frequently encountered in robotics (Fu, Levine, and Abbeel 2016; Nagabandi, Kahn, et al. 2018). However, the majority of these studies utilizing NNs have concentrated on deterministic models,

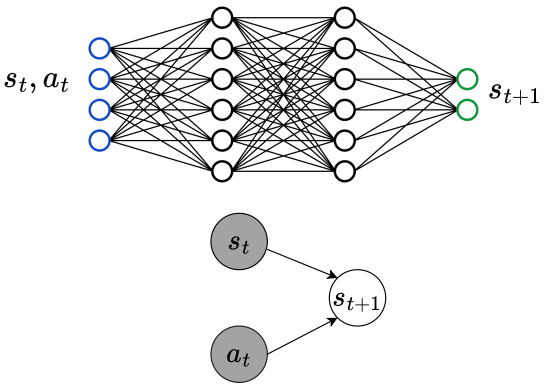


Figure 3.5.: (Top) A dynamics model based on deep neural networks. (Bottom) A graphical representation using PGM of the same model, where the shaded nodes indicate observed variables. These models operate under the assumption that the current states of the internal model of the world are already known and don't need to be inferred. This assumption is limiting, particularly when these models need to be derived from sensory information that is both high-dimensional and fraught with noise.

which do not align well with the Bayesian perspective of best guesses discussed earlier. Chua et al. 2018 proposed utilizing an ensemble of neural network models to learn probabilistic dynamics, capturing both the aleatoric uncertainty inherent in the environment and the epistemic uncertainty of the data. This approach allows for more data-efficient control and planning algorithms. Nonetheless, irrespective of being deterministic or stochastic, these NN-based dynamics models face challenges in learning with high-dimensional sensory data or in partially observable domains due to the presumption of known internal world states. Additionally, for long-term predictions, these auto-regressive models are less effective because of the accumulation of prediction errors inherent in autoregressive processes. They also presume static world dynamics and operate at a singular, fine-grained temporal scale, thereby lacking in their ability to serve as a general-purpose world model capable of addressing partial observability, non-stationarity, and hierarchical predictions across multiple time scales.

3.2.3. World Models Based On State Space Models

Classical State-Space Models (SSMs) are well-regarded for their clear inference processes and interpretable outcomes, but they often struggle with scalability and efficiency when applied to high-dimensional data or large datasets. To overcome these limitations, recent advances have introduced deep SSMs, which integrate neural networks to provide efficient and scalable inference for complex datasets. The works of Haarnoja et al. 2016; Becker, Pandya, et al. 2019 have leveraged neural network architectures to derive closed-form solutions for the forward inference algorithms in SSMs, leading to state-of-the-art performance in tasks like state estimation and dynamics prediction. Other studies, such as those of Krishnan, Shalit, and Sontag 2017; Karl et al. 2016; Fraccaro et al. 2017; Hafner et al. 2019; Becker and Neumann 2022, have focused on applying variational approximations to facilitate learning and inference within SSMs.

Despite these advances, many recurrent state-space models are based on the assumption of fixed dynamics, which does not align with the changing conditions observed in real-world scenarios, such as in robotics. To address the limitations of fixed dynamics, research by Linderman et al. 2017; Becker-Ehmck, Peters, and Van Der Smagt 2019; Dong et al. 2020 has introduced models that incorporate additional discrete switching latent variables to account for changing or multi-modal dynamics. However, these models typically operate on a single time scale, with both continuous and discrete levels operating at a fine-grained temporal resolution. They also do not adhere to a strictly top-down approach, as the top-layer latent variables depend on the predictions from the lower layer, rendering these models not suitable for long-horizon predictions and reasoning. Moreover, the introduction of discrete states complicates the learning and inference processes.

4. Action Conditional Deep SSMs: Probabilistic World Models on a Single Time Scale

This chapter is based on "Action Conditional Recurrent Kalman Networks for Forward and Inverse Dynamics Learning" (Shaj, Becker, et al. 2020).

In our quest to formalize a computational mental model of the world, we start with a simple and widely used Bayesian network called the State Space Models (SSMs). Though it operates at a single time scale (often the time scale at which observations are recorded), and hence may not satisfy the expressiveness and hierarchical structure of the mental models that the Neuroscience community advocates (Section 3.1), SSMs have several computational advantages. The chain structure of SSMs allows for tractable exact inference via message passing allowing for closed-form update rules under linear and Gaussian assumptions. When these updates are performed in the learnt latent space of a Deep Network, these linear assumptions are no longer limiting. These deep models learn a latent space where these linear assumptions work in practice. The Deep SSMs with exact inference are known in the literature as Deep Kalman models and we use one such model namely Recurrent Kalman Networks (RKN) (Becker, Pandya, et al. 2019) discussed in the preliminaries Section 2.2.2.2 as our foundation for learning a world model.

Research Objective Deep Kalman Models including RKN were primarily designed for state estimation from sensory observations in simple unactuated dynamical systems. In the real world, an agent's actions actively shape its environment, and ignoring this causal relationship in the mental models of the world can lead to inaccurate predictions and suboptimal decision-making. Hence we try to answer these questions in this chapter of the thesis:

1. Can deep Kalman models be adapted for the task of world modelling?
2. How to incorporate control action conditioning in the latent dynamics in a principled manner such that causal relations are respected?
3. How good are these models in modelling the dynamics of complex robotic systems when faced with non-markovian, partially observable and non-stationary settings?

4.1. Related Works

4.1.1. Action Conditional Predictive Models.

In model-based control and reinforcement learning (RL) problems, learning to predict future states and observations conditioned on actions is a key component. Approaches such as M. Deisenroth and Carl E Rasmussen 2011, Nagabandi, Kahn, et al. 2018, Lenz, Knepper, and Saxena 2015, Oh et al. 2015, Finn, Goodfellow, and Levine 2016 try to achieve this by using traditional models like GPs, feed forward neural networks, or LSTMs. The action conditioning is realized by concatenating the input observations and action signals (Nagabandi, Kahn, et al. 2018; M. Deisenroth and Carl E Rasmussen 2011) or via factored conditional units (G. W. Taylor and G. E. Hinton 2009), where features from some number of inputs modulate multiplicatively and are then weighted to form network outputs. In each of these approaches action conditioning happens outside of the recurrent neural network cell which leads to sub-optimal performance as observations and actions are treated similarly.

4.1.2. Predictive Models Of Robotic Agents.

Due to the increasing complexity of robot systems, analytical models are more difficult to obtain. This problem leads to a variety of model estimation techniques which allow the roboticist to acquire models from data. When learning the model, mostly standard regression techniques with Markov assumptions on the states and actions are applied to fit either the forward or inverse dynamics model to the training data. Authors used Linear Regression (Schaal, Atkeson, and Vijayakumar 2002; Haruno, Wolpert, and Kawato 2001), Gaussian Mixture Regression (Calinon et al. 2010; Khansari-Zadeh and Billard 2011), Gaussian Process Regression (Nguyen-Tuong, Seeger, and Peters 2009; Nguyen-Tuong and Peters 2010), Support Vector Regression (Ferreira et al. 2007), and feed forward neural networks (Polydoros, Nalpantidis, and Krüger n.d.). However the Markov assumptions are violated in many cases, e.g., for learning the dynamics of complex robots like hydraulically and pneumatically actuated robots or soft robots. Here, the Markov assumptions for states and actions no longer hold due to hysteresis effects and unobserved dependencies. Similarly, learning the dynamics of robots in frictional contacts with unknown objects also violates the Markov assumption and requires reliance on recurrent models which can take into account the temporal dynamic behavior of input sequences while making predictions. Recently, using RNNs such as LSTMs or GRUs has been attempted. Those scale easily to big data, available due to the high data frequencies, and allow learning in $O(n)$ time (Rueckert et al. 2017). However, the lack of a principled probabilistic modelling and action conditioning makes them less reliable to be applied for robotic control applications.

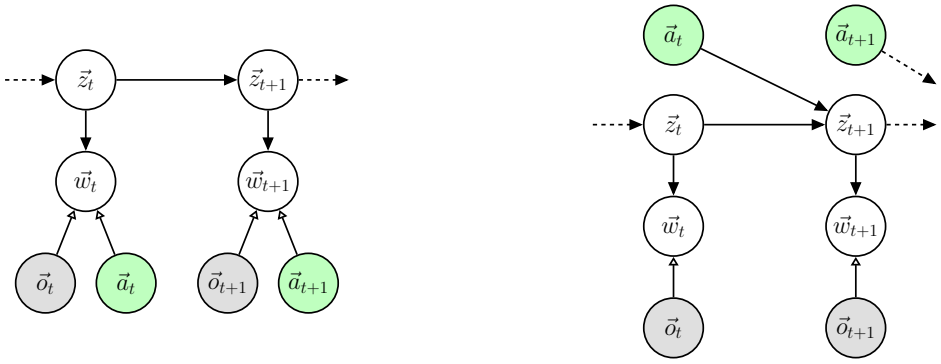


Figure 4.1.: Graphical models for actions (green). (Left) Treated as fake observations as in Becker, Pandya, et al. 2019. (Right) ac-RKN treating actions in a principled manner by capturing its causal effect on the state transition via additive interactions with the latent state. The hollow arrows denote deterministic transformation leading to implicit distributions.

4.2. Action Conditional Recurrent Kalman Networks

We extend the Recurrent Kalman Network approach to the task of world modelling / forward dynamics learning with the following innovations: **(i)** We use an **action-conditional prediction** update which provides a natural way to incorporate control inputs into the latent dynamical system. **(ii)** We modify the architecture to perform prediction instead of filtering and learn **accurate forward dynamics models** for multiple robots with complex actuator dynamics. We refer to the resulting approach as action-conditional RKN (ac-RKN).

4.2.1. Action Conditioning

To achieve action conditioning within the recurrent cell, we modify the transition model proposed in Becker, Pandya, et al. 2019 to include a control model $\vec{b}(\vec{a}_t)$ in addition to the locally linear transition model $\vec{A}_t$. The control model $\vec{b}(\vec{a}_t)$ is added to the locally linear state transition, i.e., $\vec{z}^{-}_{t+1} = \vec{A}_t \vec{z}^{+}_t + \vec{b}(\vec{a}_t)$. As illustrated in Figure 4.1, this formulation of latent dynamics captures the causal effect of the actions variables $\vec{a}_t$ on the state transitions in a more principled manner than including the action as an observation, which is the common approach for action conditioning in RNNs (Becker, Pandya, et al. 2019). The control model $\vec{b}(\vec{a}_t)$ can be represented in several ways, i.e.:

(i) Linear: $\vec{b}_l(\vec{a}_t) = \vec{B}\vec{a}_t$, where $\vec{B}$ is a linear transformation matrix.

(ii) Locally-Linear: $\vec{b}_m(\vec{a}_t) = \vec{B}_t\vec{a}_t$, where $\vec{B}_t = \sum_{k=0}^{K} \beta^{(k)}(\vec{z}_t)\vec{B}^{(k)}$ is a linear combination of k linear control models $\vec{B}^{(k)}$. A small neural network with softmax output is used to learn $\beta^{(k)}$.

(iii) Non-Linear: $\vec{b}_n(\vec{a}_t) = \vec{f}(\vec{a}_t)$, where $\vec{f}(.)$ can be any non-linear function approximator. We use a multilayer neural network regressor with ReLU activations.

Note that the prediction step for the variance is not affected by this choice of action conditioning, i.e., $\vec{\Sigma}^{-}_{t+1} = \vec{A}_t \vec{\Sigma}^{+}_{t} \vec{A}^{T}_{t} + \vec{I} \cdot \sigma^{\vec{\text{trans}}}$ as the action is known and not uncertain. Thus, unlike the state transition model $\vec{A}_t$, we do not need to constrain the model $\vec{b}$ to be linear or locally linear, as it neither affects the Kalman gain nor how the covariances are updated. Hence, we use the nonlinear approach, $\vec{b}_n(\vec{a}_t)$, as it provides the most flexibility and achieves the best performance, as shown in Section 4.3. The Kalman update step is also unaffected by the new action-conditioned prediction step and therefore remains as presented in Becker, Pandya, et al. 2019.

Our principled treatment of control signals is crucial for learning accurate forward dynamics models, as detailed in Section 4.2.2. Moreover, the disentangled representation of actions in the latent space gives us more flexibility in manipulating the control actions for different applications including inverse dynamics learning (Appendix A.1), extensions to model-based reinforcement learning, and planning. For long-term prediction using different action sequences, we can still apply the latent transition dynamics without using observations for future time steps (skipping the observation update step).

4.2.2. Ac-RKN as Single Time Scale Probabilistic World Models

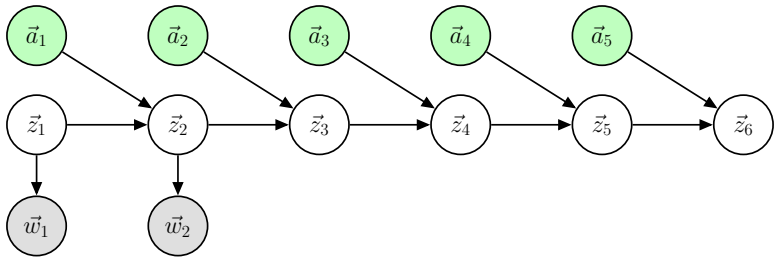


Figure 4.2.: Using internal world models to envision the future: With past and present observations ($w_{1:2}$) as a basis, Ac-RKN projects future world states ($z_{3:6}$) by considering a sequence of action signals ($a_{1:5}$).

World models attempt to learn a compact and expressive representation of the environment dynamics from observed data. These models can predict possible future world states and associated uncertainties as a function of an imagined action sequence. More formally, we want to learn a forward dynamics model of the world $f : \vec{o}_{1:t}, \vec{a}_{1:t} \mapsto \vec{o}_{t+1}$ that predicts the next observation $\vec{o}_{t+1}$ given the histories of observation $\vec{o}_{1:t}$ and actions $\vec{a}_{1:t}$.

Formulating action-conditional future predictions as inference tasks within a Gaussian State Space Model (SSM) in deep latent spaces enables the development of probabilistic world models that exhibit the following properties.

1. **The propagation of uncertainties** ($\boldsymbol{\Sigma}_t$) about the belief of our future world states in addition to the mean estimates ($\boldsymbol{z}_t$).
2. **Learning under partial observability** and missing observations. The RKN cell is well equipped to handle missing observations, as the update step can just

be omitted in this case, i.e. the posterior is equal to the prior if no observation is available. In this case, we repeatedly apply the ac-RKN prediction step while omitting the Kalman update.

3. **Capturing non-markovian dynamics** The model captures non-markovian dynamics in a principled way, since we predict the future belief states $p(\vec{z}_{t+1}|\vec{w}_{1:t}, \vec{a}_{1:t})$ conditioned on all the past observations and the control actions given so far. These beliefs are obtained in closed form using the forward inference algorithm (action conditional Kalman predict steps).

In line with theories from neuroscience discussed in Section 3.1, this results in a generative model that makes Bayesian best guesses about the world states based on the action conditional prior beliefs (top-down predictions) and bottom-up sensory signals/ observations when available. Neuroscience research (Hohwy 2013; A. K. Seth 2014; A. Seth 2021) similarly posits that the Brain is not only tasked with the challenge of figuring out the most likely causes of its sensory signals but also figuring out how relevant the sensory inputs are. Similarly, the Ac-RKN model learns to estimate the reliability of the incoming sensory information by predicting uncertainties of the sensory signals via the observation encoder (bottom-up processing). For a visual representation of these top-down and bottom-up predictions in the Ac-RKN, please see Figure 4.3.

4.2.3. End To End Learning Via Backpropagation

The network is tasked to minimize the prediction errors by maximizing the posterior predictive log-likelihood, which is given below for a single trajectory, i.e., $\mathrm{L} = \sum_{t=1}^{H} \log p(\boldsymbol{o}_{t+1}|\boldsymbol{w}_{1:t}, \boldsymbol{a}_{1:t})$
$= \sum_{t=1}^{H} \log \int p(\boldsymbol{o}_{t+1}|\boldsymbol{z}_{t+1}) p(\boldsymbol{z}_{t+1}|\boldsymbol{w}_{1:t}, \boldsymbol{a}_{1:t}) d\boldsymbol{z}_{t+1}$
The extension to multiple trajectories is straightforward and omitted to keep the notation uncluttered. Here, $\boldsymbol{o}_{t+1}$ is the ground truth observations at the time step $t+1$ which needs to be predicted from all observations up to time step t.

4.2.3.1. Approximating the likelihood

We employ a Gaussian approximation of the posterior predictive log-likelihood of the form $p(\boldsymbol{o}_{t+1}|\boldsymbol{w}_{1:t}, \boldsymbol{a}_{1:t}) \approx \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{o}_{t+1}}, \mathrm{diag}(\boldsymbol{\sigma}_{\boldsymbol{o}_{t+1}}))$ where we use the mean of the prior belief $\boldsymbol{\mu}^{-}_{z_{t+1}}$ to decode the predictive mean, i.e, $\boldsymbol{\mu}_{\boldsymbol{o}_{t+1}} = \mathrm{dec}_{\boldsymbol{\mu}}(\boldsymbol{\mu}^{-}_{z_{t+1}})$ and the variance estimate of the prior belief to decode the observation variance, i.e., $\boldsymbol{\sigma}_{o_{t+1}} = \mathrm{dec}_{\sigma}(\boldsymbol{\Sigma}^{-}_{z_{t+1}})$. This approximation can be motivated by a moment-matching perspective and allows for end-to-end optimization of the log-likelihood without using auxiliary objectives such as the ELBO (Becker, Pandya, et al. 2019). Thus the approximate Gaussian predictive log-likelihood for a single sequence is then computed as $\mathrm{L}\Big(\boldsymbol{o}_{(1:T)}\Big) =$
$1T \sum_{t=1}^{T} \log \mathcal{N}\left(\boldsymbol{o}_t \middle| \mathrm{dec}_{\mu}(\boldsymbol{z}_t^{+}), \mathrm{dec}_{\Sigma}(\boldsymbol{\sigma}_t^{\mathrm{u},+}, \boldsymbol{\sigma}_t^{\mathrm{s},+}, \boldsymbol{\sigma}_t^{\mathrm{l},+})\right)$, where $\mathrm{dec}_{\mu}(\cdot)$ and $\mathrm{dec}_{\Sigma}(\cdot)$ denote

the parts of the decoder that are responsible for decoding the latent mean and latent variance respectively.

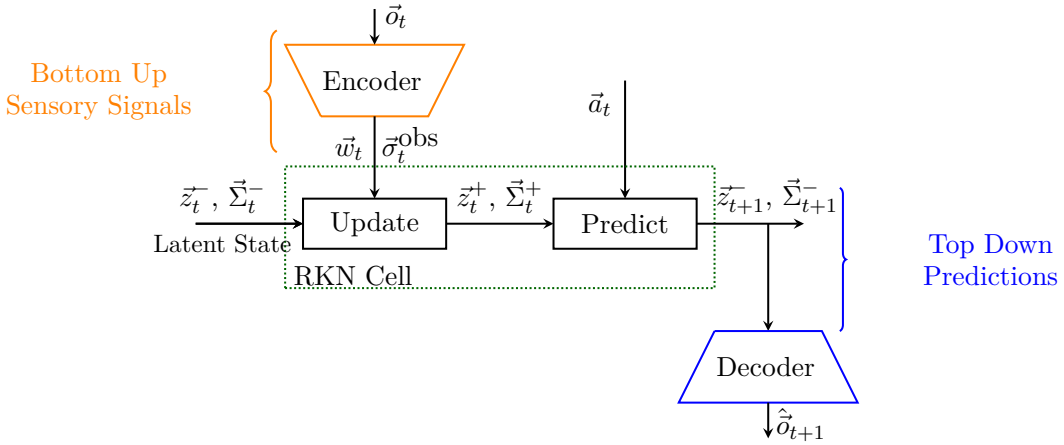


Figure 4.3.: Schematic diagram of ac-RKN for forward dynamics learning. Action conditioning is implemented by adding a latent control vector $\vec{b}(\vec{a}_t)$ to the RKN dynamics model. The output of the prediction stage, which forms the prior for the next time step $\left(\vec{z}_{t+1}^-, \vec{\Sigma}_{t+1}^-\right)$ is decoded to obtain the prediction of the next observation.

4.2.3.2. Variations Of The Learning Objective for Learning Real Robot Dynamics

In this part of the thesis, we specifically look at the learning challenges associated with modeling the dynamics of real robot systems and propose different learning objectives. We explicitly focus on non-Markovian systems, where the observations are typically joint angles and, if available, joint velocities of all degrees of freedom of the robot. In our experiments, we aim to predict the joint angles at the next time step given a sequence of joint angles and control actions. Due to unmodeled effects such as hysteresis or unknown contacts, the state transitions are non-Markovian even if joint angles and velocities are known. We can assume that the observations are almost noise-free, since the measurement errors for our observations (joint angles) are minimal. Nevertheless, as the observations do not contain the full state information of the system, we still have to model uncertainty in our latent state using the RKN.

1. **Mean Predictions** To ensure a fair comparison with deterministic world models such as LSTM and GRU, we train the model with a variation of the objective in Equaton 5.4.1, where only the mean of the latent states are decoded, while the variance units are kept fixed, resulting in an RMSE loss. $\mathrm{L}_{\text{fwd}} = \sqrt{1T \sum_{t=1}^{T} \parallel (\vec{o}_{t+1} - \vec{o}_t) - \text{dec}\left(\vec{z}_{t+1}^-\right) \parallel^2}$. We also observed that in use cases where only mean predictions are required, this results in slightly better predictions.

2. **High-frequency Predictions** The function f can be challenging to learn when the observations, i.e. joint positions and velocities, of two subsequent time steps are too similar (for example, our Franka Robot's operating frequency in 1000Hz). In this case, the action has seemingly little effect on the output, and the learned strategy is often to copy the previous state for the next-state prediction. This difficulty becomes more pronounced as the time step between states becomes smaller, e.g., 1ms, as minor errors in absolute states estimates can already create unrealistic dynamics. Therefore, a standard method for model learning is to predict the normalized differences between subsequent observations or states instead of the absolute values M. Deisenroth and Carl E Rasmussen 2011, that is, during training, the next predicted observation is $\hat{\vec{o}}_{t+1} = \vec{o}_t + \text{dec}\left(\overrightarrow{z}_{t+1}^{-}\right)$, where $\vec{o}_t$ is the true observation in t and $\text{dec}\left(\overrightarrow{z}_{t+1}^{-}\right)$ is the output of the actual decoder network. During inference, we introduce an additional memory $\vec{m}_t$ that stores the last prediction and uses it as an observation in case of a missing observation, that is, $\hat{\vec{o}}_{t+1} = \vec{m}_t + \text{dec}\left(\overrightarrow{z}_{t+1}^{-}\right)$ where we use the true observation, $\vec{m}_t = \vec{o}_t$ if available, and the predicted observation, $\vec{m}_t = \hat{\vec{o}}_t$, otherwise.

Irrespective of the learning objectives/loss functions used, we still have to model uncertainty in our latent state using the RKN as the observations do not contain the full state information of the system.

The architecture of the ac-RKN is summarized in Figure 4.3.

Note that we decode the prior mean $\overrightarrow{z}_{t+1}^{-}$ to predict $\vec{o}_{t+1}$. The prior mean $\overrightarrow{z}_{t+1}^{-}$ integrates all information up to time step t, including $\vec{a}_t$, but already denotes the belief for the next time step. Hence, as we are interested in prediction, we work with this prior. On the contrary, Becker, Pandya, et al. 2019 used posterior belief, as their goal was filtering rather than prediction.

4.2.4. Self Supervised Training For Multi-Step Prediction

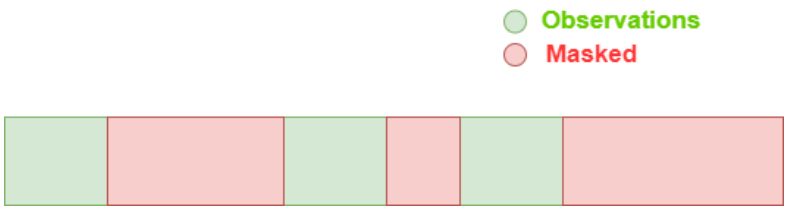


Figure 4.4.: Self-supervised training for multi-step ahead predictions by tasking the model to "fill in" the masked observations.

Using the training objective(s) discussed in Section 4.2.3 results in models that are good in one-time step prediction, but typically perform poorly in long-term predictions as the loss assumes that observations are always available up to time step t. To increase the performance of the long-term prediction, we can treat the long-term prediction problem as a case of the problem of "missing value", where the missing observations occur in future time steps. Thus, to train our model for long-term prediction, we randomly mask a fraction of observations as shown in Figure 4.4 and explicitly task the network to impute the missing observations, resulting in a strong self-supervised learning signal for long-term prediction with varying prediction horizon length.

4.3. Experiments

In this section, we discuss the experimental evaluation of ac-RKN on learning forward dynamics models on robots with different actuator dynamics. A full listing of hyperparameters can be found in the Supplementary Material. We compare ac-RKN to both standard deep recurrent neural network baselines (LSTMs, GRUs and standard RKN), analytical baselines based on classical rigid body dynamics, and nonrecurrent baselines like FFNN. For the RKN and LSTM baselines, we replaced the ac-RKN transition layer with generic LSTM and RKN layers. For recurrent models, the parameters of the recurrent cell are tuned through hyperparameter optimization using GPyOpt (authors 2016), but the size of the encoder and decoder are similar. The observations and actions are concatenated as in Nagabandi, Kahn, et al. 2018 and M. Deisenroth and Carl E Rasmussen 2011 for all baseline experiments during the forward dynamics learning, that is, we treat actions as extended observations. The data, i.e., the states or observations, actions, and targets, are normalized (zero mean and unit variance) before training. We denormalize the predicted values during inference and evaluate their performance in the test set. We evaluate the model using RMSE to ensure a fair comparison with deterministic models such as FFNNs and LSTMs.

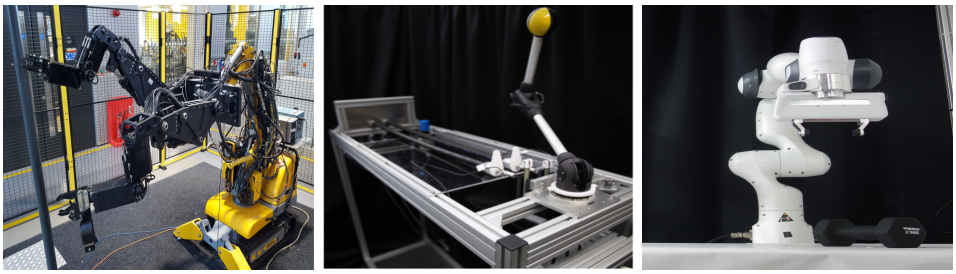
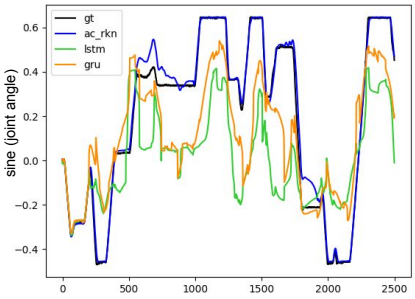


Figure 4.5.: The experiments are performed on data from robots with different actuator dynamics. From left to right, these include: Hydraulically actuated BROKK-40 (C. J. Taylor and Robertson 2013), pneumatically actuated artificial muscles (Büchler, Ott, and Peters 2016), Franka Emika Panda Robotic Arm

We evaluate the ac-RKN on three robotic systems with different actuator dynamics each of which poses unique learning challenges.

	RMSE	LL
ac-RKN	**0.0144**	**6.247**
RKN	0.0282	4.930
LSTM	0.2067	0.952
GRU	0.2015	1.186

(a) Hydraulic Brokk 40



(b) Hydraulic Arm Predictions

Figure 4.6.: **(a)** Performance comparison of various recurrent models for the BROKK-40 hydraulically actuated robot arm. **(b)** Visualization of the predicted trajectories of the hydraulic arm for 200 step ahead predictions.

1. **Hydraulically Actuated BROKK-40 Robot Arm**. The data consists of measured joint positions and the input current to the controller sampled at 100Hz from a hydraulic BROKK-40 demolition robot (C. J. Taylor and Robertson 2013). The position angle sensors are rotary linear potentiometers. We chose a similar experimental setup and metrics as in Becker, Pandya, et al. 2019 to ensure a fair comparison. We trained the model to predict the joint position 2 seconds, i.e. 200 time-steps, into the future, given only control inputs. Afterwards, the model receives the next observation and the prediction process repeated. Learning the forward model here is difficult due to inherent hysteresis associated with hydraulic control.

2. **Pneumatically Actuated Musculoskeletal Robot Arm.** This four DoF robotic arm is actuated by Pneumatic Artificial Muscles (PAMs) (Büchler, Ott, and Peters 2016). Each DoF is actuated by an antagonistic pair of PAMs, yielding a total of eight actuators. The robot arm reaches high joint angle accelerations of up to 28,000 deg/s^2 while avoiding dangerous joint limits thanks to the antagonistic actuation and limits on the air pressure ranges. The data consists of trajectories of hitting movements with varying speeds while playing table tennis (Büchler, Guist, et al. 2020) and is recorded at 100Hz. The fast motions with high accelerations of this robot are complicated to model due to hysteresis.

3. **Franka Emika Panda Arm.** We collected the data from a 7 DoF Franka Emika Panda manipulator during free motion at a sampling frequency of 1kHz. It involved a mix of movements of different velocities from slow to swift motions. The high frequency, together with the abruptly changing movements results in complex dynamics which are interesting to analyze.

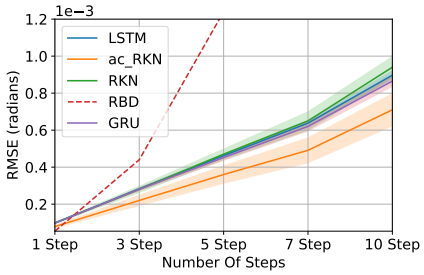


Figure 4.7.

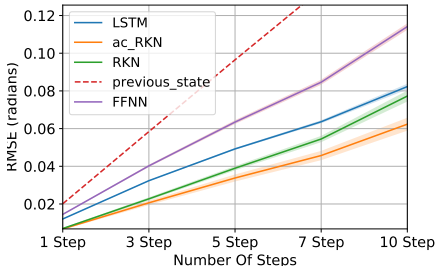


Figure 4.8.

Figure 4.9.: Plots show the comparison of different action-conditioning models discussed in Section 4.2.1 for (left) Franka Emika Panda Robot and (right) Pneumatic Muscular Robot Arm. The plots clearly show that the predictions given by our approach are by far the most accurate.

4.3.1. Multi Step Ahead Prediction

Figures 4.6 and 4.9 summarize the test set performance for each of these robots for a multi-step ahead prediction task. We benchmarked the performance of ac-RKN with state-of-the-art deterministic deep recurrent models (LSTM and GRU) and non-recurrent models like feed-forward neural network (FFNN). In all three robots, the ac-RKN gave much better multi-step ahead prediction performance than these deterministic deep models. The performance improvement was more significant for robots with hydraulic and pneumatic actuator dynamics due to explicit non-markovian dynamics and hysteresis, which is often difficult to model via analytical models. We also validated the performance improvement due to our principled action-conditioning in the latent transition dynamics by comparing it with RKN (Becker, Pandya, et al. 2019), which treats actions as part of the observations by 'concatenation'. In all three robots, our principled treatment brought significant improvement in performance. Figures 4.6b and 4.11 show the predicted trajectories for the BROKK and the pneumatically actuated robot arm.

4.3.2. Comparison with Analytical Model

We were also interested in comparing these with analytical models of Franka. For the analytical model, in addition to the inertia properties of the links, Coulomb friction was also identified for each joint. A detailed description of the same can be found in Appendix C. As seen in Figure 4.9, the performance of the analytical model outperformed ac-RKN for step-by-step predictions, but for multistep forward predictions, the data-driven models had a clear advantage over ac-RKN that provides the most accurate results.

4.3.3. Ablation Study for Action-Conditioning.

We evaluated the performance of the model for different action conditioning schemes (linear, locally-linear and non-linear) discussed in Section 4.2.1. The resulting evaluation can be seen in Figure 4.10 and shows the advantage of using nonlinear models for the additive action conditioning of the latent dynamics.

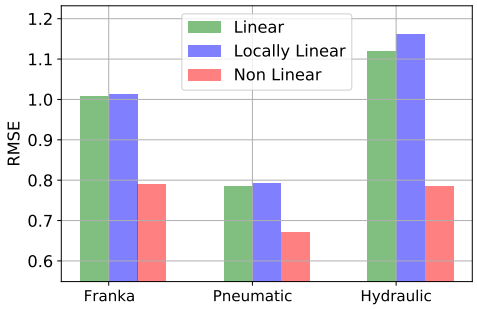


Figure 4.10.: Impact of control input models

4.3.4. Inverse Dynamics Learning

In addition, we also adapted ac-RKN for inverse dynamics learning tasks. Since this is out of scope of the main topic of the thesis of World modeling, we report the details in Appendix A.1.

4.4. CONCLUSION

In this section, we modified a contemporary deep Kalman filter approach for world modeling by incorporating action/control signals into the generative process giving them the capability to predict the sensory consequences of actions. By conditioning the actions on the latent state while maintaining the causal relationships between these entities, we enable the execution of interventions and counterfactual analyses with these signals. Such capabilities are essential for decision-making and planning processes using these models. This design facilitates the learning of precise forward dynamics models for complex robots and scenarios lacking analytical models. We validated the efficacy of our method on robots equipped with hydraulic, pneumatic, and electric actuators in settings with stationary dynamics. Initial experiments indicated that the current generative model, the ac-SSM, falls short in handling non-stationary dynamics. We plan to address this limitation in the subsequent chapter.

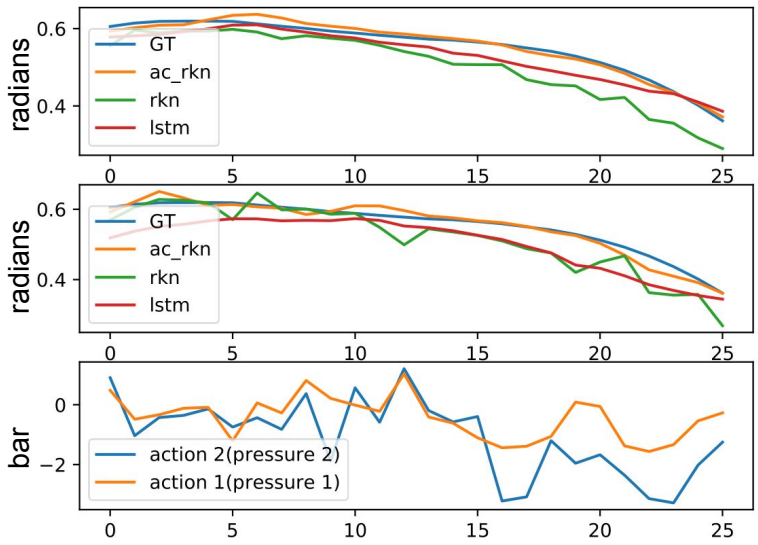


Figure 4.11.: Visualization of the predicted trajectories of the pneumatic arm for 5-step ahead (top) and 10-step ahead (middle) predictions for one of the joints. The bottom plot shows the corresponding pressure (actions) applied in bars. The figure clearly shows that the predictions given by our principled action-conditional scheme are by far the most accurate.

5. Hidden Parameter SSMs: Probabilistic World Models with Task Abstractions

This chapter is based on "Hidden-Parameter Recurrent State-Space Models for Changing Dynamics Scenarios" (Shaj, Buchler, et al. 2022).

One of the most notable abilities of intelligent biological agents is their ability to adapt their behavior to different contexts and environments (i.e., cognitive flexibility) through learning. Isomura, Parr, and K. Friston 2019 postulates that the brain maintains a collection of generative internal models of the world, and consequential flexibility arises from contextualization and selection on the basis of higher-order beliefs about the most plausible hypothesis, task, or context in play.

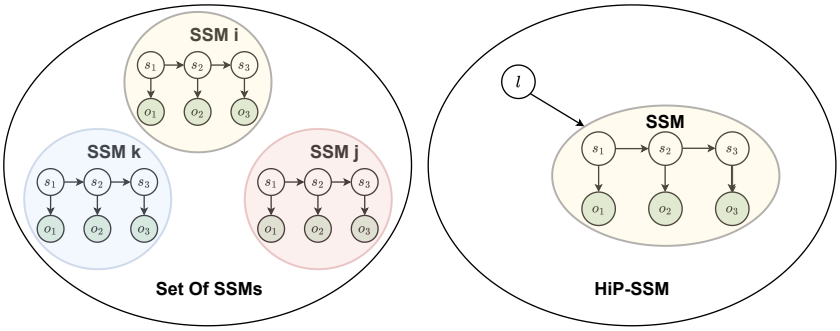


Figure 5.1.: (left) We visualize a set of internal models of the world, each depicted as an individual State-Space Model (SSM) as detailed in Chapter 4. These models cater to diverse dynamics and tasks. (right) The proposed HiP-SSM framework enables modeling of multitask dynamics using a singular, overarching model. It achieves this through a hierarchical latent task variable, denoted as l that parametrize the latent dynamics.

In this part of the thesis, we focus on the adaptability aspects of world models. For many real-world control applications, an intelligent agent must learn to solve tasks with similar, but not identical, dynamics. A robot playing table tennis may encounter several bats with different weights or lengths, while an agent manipulating a bottle may encounter bottles with varying amounts of liquid. An agent driving a car may encounter many different cars, each with unique handling characteristics. Humanoids learning to walk may face different terrains with varying slopes or friction coefficients. Any real-world

dynamical system might change over time due to multiple reasons, some of which might not be directly observable or understood. For example, in soft robotics small variations in temperature or changes in friction coefficients of the cable drives of a robot can significantly change the dynamics. Similarly, a robot may undergo wear and tear over time, which can change its dynamics. Assuming a global model as discussed in Chapter 4 that is accurate throughout the entire state space or duration of use is a limiting factor for using such models in real-world applications.

Research Objective Given the limitations of existing literature on recurrent models, particularly in modeling the dynamics of nonstationary situations, and the impracticality of employing separate State Space Models (SSMs) for each task in continuously changing systems, this research attempts to answer the following question:

1. Is it feasible to develop a formalism that enables the learning of a unified global dynamics model, applicable across multiple tasks and environments, while incorporating a hierarchical latent task variable for task-specific specialization?
2. Can we perform learning and inference in a scalable manner with such a generative model ?
3. How do these models compare against traditional Recurrent State Space Models (RSSMs) and other contemporary state-of-the-art models in environments with evolving dynamics?

We thus introduce hidden parameter state-space models (HiP-SSM), which allow capturing the variation in the dynamics of different instances through a set of hidden task parameters. We formalize the HiP-SSM and show how to perform inference in this graphical model. Under Gaussian assumptions, we obtain a probabilistically principled yet scalable approach. We name the resulting probabilistic recurrent neural network as Hidden Parameter Recurrent State Space Model (HiP-RSSM). HiP-RSSM achieves state-of-the-art performance for several systems whose dynamics change over time. Interestingly, we also observe that HiP-RSSM often exceeds traditional RSSMs in performance for dynamical systems previously assumed to have unchanging global dynamics due to the identification of unobserved causal effects in the dynamics.

5.1. Related Work

Deep State Space Models. Classical State-space models (SSMs) are popular due to their tractable inference and interpretable predictions. However, inference and learning in SSMs with high dimensional and large datasets are often not scalable. Recently, there have been several works on deep SSMs that offer tractable inference and scalability to high dimensional and large datasets. Haarnoja et al. 2016; Becker, Pandya, et al. 2019; Shaj, Becker, et al. 2020 use neural network architectures based on closed-form solutions

of the forward inference algorithm on SSMs and achieve state-of-the-art results in state estimation and dynamics prediction tasks. Krishnan, Shalit, and Sontag 2017; Karl et al. 2016; Hafner et al. 2019 perform learning and inference in SSMs using variational approximations. However, most of these recurrent state-space models assume that the dynamics is fixed, which is a significant drawback, since this is rarely the case in real-world applications such as robotics.

Recurrent Switching Dynamical Systems. Linderman et al. 2017; Becker-Ehmck, Peters, and Van Der Smagt 2019; Dong et al. 2020 tries to address the problem of changing/multimodal dynamics by incorporating additional discrete switching latent variables. However, these discrete states make learning and inference more involved. Linderman et al. 2017 uses auxiliary variable methods for approximate inference in a multi-stage training process, while Becker-Ehmck, Peters, and Van Der Smagt 2019; Dong et al. 2020 uses variational approximations and relies on additional regularization/annealing to encourage discrete state transitions. On the other hand, Fraccaro et al. 2017 uses "soft" switches, which can be interpreted as a switching linear dynamical system which interpolate linear regimes continuously rather than using truly discrete states. We take a rather different, simpler formalism for modeling changing dynamics by viewing it as a multi-task learning problem with a continuous hierarchical hidden parameter that model the distribution over these tasks. Further our experiments in appendix B.2.1 show that our model significantly outperforms the soft switching baseline (Fraccaro et al. 2017).

Hidden Parameter MDPs. Hidden Parameter Markov Decision Process (HiP-MDP) (Doshi-Velez and Konidaris 2016; Killian, Konidaris, and Doshi-Velez 2016) address the setting of learning to solve tasks with similar, but not identical, dynamics. In HiP-MDP formalism, the states are assumed to be fully observed. However, we formalize the partially observable setting where we are interested in modelling the dynamics in a latent space under changing scenarios. The formalism is critical for learning from high dimensional observations and dealing with partial observability and missing data. The formalism can be easily extended to HiP-POMDPs by including rewards into the graphical model 5.2, for planning and control in the latent space (Hafner et al. 2019; Sekar et al. 2020). However this is left as a future work.

Meta Learning For Changing Dynamics. There exists a family of approaches that attempt online model adaptation to changing dynamics scenarios via meta-learning (Nagabandi, Clavera, et al. 2018; Nagabandi, Finn, and Levine 2018). They perform online adaptation on the parameters of the dynamics models through gradient descent (Finn, Abbeel, and Levine 2017) from interaction histories. Our method fundamentally differs from these approaches in the sense that we do not perform a gradient descent step at every time step during test time, which is computationally impractical in modern robotics, where we often deal with high-frequency data. We also empirically show that our approach adapts

better, especially in scenarios with non-markovian dynamics, a property that is often encountered in real-world robots due to stiction, slip, friction, pneumatic/hydraulic/cable-driven actuators etc. Sæmundsson, Hofmann, and M. P. Deisenroth 2018; Achterhold and Stueckler 2021 on the other hand, learn context-conditioned hierarchical dynamics model for control in a formalism similar to HiP-MDPs. The former meta-learn a context-conditioned Gaussian process while the later use a context-conditioned determisitic GRU. Our method on the other hand focuses on a principled probabilistic formulation and inference scheme for context-conditioned recurrent state space models, which is critical for learning under partial observabilty/high dimensional observations, with noisy and missing data.

5.2. Hidden Parameter State Space Models (HiP-SSMs)

We denote a set of SSMs with transition dynamics $f_{\boldsymbol{l}}$ that are fully described by hidden parameters $\boldsymbol{l}$ and observation model h as a Hidden Parameter SSM (HiP-SSM). In this definition we assume the observation model h to be independent of the hidden parameter $\boldsymbol{l}$ as we only focus on cases where the dynamics change. HiP-SSMs allow us to extend SSMs to multitask settings where dynamics can vary across tasks. Defining the changes in dynamics by a latent variable unifies the dynamics across tasks as a single global function. In dynamical systems, for example, parameters can be physical quantities like gravity, friction of a surface, or the strength of a robot actuator. Their effects influence the dynamics but are not directly observed, so $\boldsymbol{l}$ is not part of the observation space and is treated as a latent task parameter vector. The Bayesian network corresponding to HiP-SSM is shown in Figure 5.2 and a formal definition is given below.

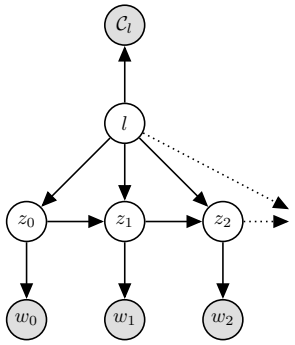


Figure 5.2.: The graphical model for a particular instance for the HiP-SSM. The transition dynamics between latent states is a function of the previous latent state, and a specific latent task parameter $\boldsymbol{l}$ which is inferred from a context set of observed past observations. Actions are omitted for brevity.

Definition: HiP-SSM

A HiP-SSM is represented by a tuple $\{\mathcal{L}, \mathcal{C}, \mathcal{Z}, \mathcal{A}, \mathcal{W}, g, f, h\}$, where $\mathcal{Z}$, $\mathcal{A}$, $\mathcal{W}$, $\mathcal{L}$ and $\mathcal{C}$ denote the sets of hidden states $\boldsymbol{z}$, actions $\boldsymbol{a}$, observations $\boldsymbol{w}$, latent tasks $\boldsymbol{l}$ and contexts C_l respectively.
Based on a task model g, transition model f and observation model g, HiP-SSM have the following causal relationship:

$$\boldsymbol{C}_l = g(\boldsymbol{l}) + \boldsymbol{v}_l, \boldsymbol{z}_t = f(\boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{l}) + \boldsymbol{\epsilon}_t, \boldsymbol{w}_t = h(\boldsymbol{z}_t) + \boldsymbol{u}_t.$$

Here, $\boldsymbol{z}_t \in \mathcal{Z}$, $\boldsymbol{a}_t \in \mathcal{A}$, and $\boldsymbol{w}_t \in \mathcal{W}$ represent latent states, actions, and observations at time t, respectively. The vector $\boldsymbol{\epsilon}_t$ denotes the process / transition noise, while $\boldsymbol{u}_t$ and $\boldsymbol{v}_l$ denote the measurement noise corresponding to observations and contexts.

Thus, a HiP-SSM describes a class of dynamics, and a particular instance of that class is obtained by fixing the parameter vector $\boldsymbol{l} \in \mathcal{L}$. The dynamics f for each instance depends on the value of the hidden parameters $\boldsymbol{l}$.

Each instance of a HiP-SSM is an SSM conditioned on $\boldsymbol{l}$. We also make the additional assumption that the parameter vector $\boldsymbol{l}$ is fixed for the duration of the task (i.e., a local segment of a trajectory), and thus the latent task parameter has no dynamics. This assumption considerably simplifies the procedure to infer hidden parameterization and is reasonable, since dynamics can be assumed to be locally consistent over small trajectories in most applications (Nagabandi, Clavera, et al. 2018).

The definition is based on the related literature on HiP-MDPs (Doshi-Velez and Konidaris 2016), where the only unobserved variable is the latent task variable. One can connect HiP-SSMs with HiP-MDPs by including rewards in the definition and formalizing HiP-POMDPs. However, this is left for future work.

5.3. Exact Inference In HiP-SSMs

We perform inference and learning in the HiP-SSM borrowing concepts from both deep learning and graphical model communities following recent work on recurrent neural network models (Haarnoja et al. 2016; Becker, Pandya, et al. 2019), where the architecture of the network is informed by the structure of the probabilistic state estimator. We denote the resulting probabilistic recurrent neural network architecture as the Hidden Parameter Recurrent State Space Model (HiP-RSSM).

The structure of the Bayesian network shown in Figure 5.3 allows a tractable inference of latent variables by the forward algorithm (Jordan 2004; Koller and Friedman 2009). Since we are dealing with continuous dynamical systems, we assume a Gaussian multivariate distribution over all variables (both observed and hidden) for the graph shown in Figure 5.3. This assumption has several advantages. Firstly, it makes the inference very similar

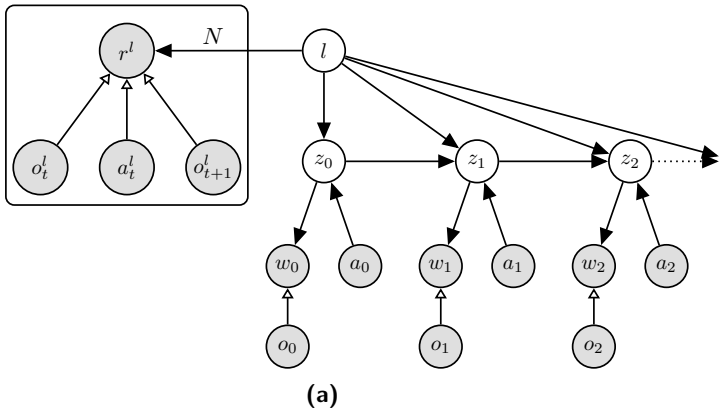


Figure 5.3.: The Gaussian graphical model corresponding to a particular instance $\boldsymbol{l} \in \mathcal{L}$ for the HiP-RSSM. The posterior of the latent task/context variable is inferred via a Bayesian aggregation procedure described in 5.3.1 based on a set of N interaction histories. The prior over the latent states $\boldsymbol{z}_t^-$ is inferred via task conditional Kalman time update described in 5.3.2 and the posterior over the latent states $\boldsymbol{z}_t^+$ is inferred via Kalman measurement update described in 5.3.3. Here, the hollow arrows denote deterministic transformation leading to implicit distributions, using the context and observation encoders.

to the well studied Kalman Filtering approaches. Secondly, the Gaussian assumptions and conditional in-dependencies allows us to have a closed form solution to each of these update procedures which are fully differentiable and can be backpropagated to the deep encoders. The update of beliefs about the hidden variables $\boldsymbol{z}_t$ and $\boldsymbol{l}$ takes place in three stages. Similar to Kalman filtering approaches, we have two recursive belief state update stages, the time update and observation update which calculate the prior and posterior belief over the latent states, respectively. However, we have an additional hierarchical latent variable $\boldsymbol{l}$ which models the (unobserved) causal factors of variation in dynamics in order to achieve efficient generalization. Hence, we have a third belief update stage to calculate the posterior over the latent task variable based on the observed context set. Each of these three stages is detailed in the following sections:

5.3.1. Inferring The Latent Task Variable (Context Update)

In this stage, we infer the posterior over the Gaussian latent task variable $\boldsymbol{l}$ based on an observed context set $\mathcal{C}_{\boldsymbol{l}}$. For any HiP-RSSM instance defined on a target sequence $\mathcal{T} = (\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1}, ..., \boldsymbol{o}_{t+N}, \boldsymbol{a}_{t+N})$, over which we intend to perform state estimation/prediction, we maintain a fixed context set $\mathcal{C}_{\boldsymbol{l}}$. $\mathcal{C}_{\boldsymbol{l}}$ in our HiP-SSM formalism can be obtained according to the algorithm designer's choice. We choose a $\mathcal{C}_{\boldsymbol{l}}$ consisting of a set of tuples $\mathcal{C}_{\boldsymbol{l}} = \{\boldsymbol{o}_{t-n}^{\boldsymbol{l}}, \boldsymbol{a}_{t-n}^{\boldsymbol{l}}, \boldsymbol{o}_{t-n+1}^{\boldsymbol{l}}\}_{n=1}^{N}$. Here each set element is a tuple consisting of the current state/observation, current action, and next state/observation for the previous N time steps.

Inferring a latent task variable $\boldsymbol{l} \in \mathcal{L}$ based on an observed context set $\mathcal{C}_{\boldsymbol{l}} = \{\mathbf{x}_n^{\boldsymbol{l}}\}_{n=1}^{N}$ has previously been explored by different neural process architectures (Gordon et al.

2018; Garnelo et al. 2018). Neural processes are multitask latent variable models that rely on deep set functions (Zaheer et al. 2017) to generate a latent representation from a varying number of context points in a permutation invariant manner. Similar to Volpp et al. 2020 we formulate the aggregation of context data as a Bayesian inference problem, where the information contained in $\mathbf{C}_l$ is directly aggregated into the statistical description of $\boldsymbol{l}$ based on a factorized Gaussian observation model of the form $p(\boldsymbol{r}_n^{\boldsymbol{l}}|\boldsymbol{l})$, where $\boldsymbol{r}_n^{\boldsymbol{l}} = \text{enc}_{\boldsymbol{r}}(\boldsymbol{o}_{t-n}^{\boldsymbol{l}}, \boldsymbol{a}_{t-n}^{\boldsymbol{l}}, \boldsymbol{o}_{t-n+1}^{\boldsymbol{l}})$,
$\boldsymbol{\sigma}_n^{\boldsymbol{l}} = \text{enc}_{\boldsymbol{\sigma}}(\boldsymbol{o}_{t-n}^{\boldsymbol{l}}, \boldsymbol{a}_{t-n}^{\boldsymbol{l}}, \boldsymbol{o}_{t-n+1}^{\boldsymbol{l}})$. Here n is the index of an element from the context set $\mathcal{C}_{\boldsymbol{l}}$. Given a prior $p_0(\boldsymbol{l}) = \mathcal{N}(\boldsymbol{l}|\boldsymbol{\mu}_0, \text{diag}(\boldsymbol{\sigma}_0))$ we can compute the posterior $p(\boldsymbol{l}|\mathbf{C}_{\boldsymbol{l}})$ using the Bayes rule. The Gaussian assumption allows us to obtain a closed-form solution for the posterior estimate of the latent task variable, $p(\boldsymbol{l}|\mathbf{C}_{\boldsymbol{l}})$ based on Gaussian conditioning. The factorization assumption further simplifies this update rule by avoiding computationally expensive matrix inversions into a simpler update rule as
$\boldsymbol{\sigma}_{\boldsymbol{l}} = \left((\boldsymbol{\sigma}_0)^{\ominus} + \sum_{n=1}^{N} \left(\boldsymbol{\sigma}_n^{\boldsymbol{l}}\right)^{\ominus} \right)^{\ominus}$,
$\boldsymbol{\mu}_{\boldsymbol{l}} = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_{\boldsymbol{l}} \odot \sum_{n=1}^{N} \left(\boldsymbol{r}_n^{\boldsymbol{l}} - \boldsymbol{\mu}_0\right) \oslash \boldsymbol{\sigma}_n^{\boldsymbol{l}}$ where $\ominus$, $\odot$ and $\oslash$ denote element-wise inversion, product, and division, respectively. Intuitively, the mean of the latent task variable $\boldsymbol{\mu}_{\boldsymbol{l}}$ is a weighted sum of the individual latent observations $\boldsymbol{r}_n^{\boldsymbol{l}}$, while the variance of the latent task variable $\boldsymbol{\sigma}_{\boldsymbol{l}}$ gives the uncertainty of this task representation.

5.3.2. Inferring Prior Latent States over the Next Time Step (Task Conditional Prediction)

The goal of this step is to compute the prior marginal $\text{p}(\boldsymbol{z}_t|\boldsymbol{w}_{1:t-1}, \boldsymbol{a}_{1:t}, \mathcal{C}_{\boldsymbol{l}}) = p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_t, \boldsymbol{l})p(\boldsymbol{z}_{t-1}|\boldsymbol{w}_{1:t-1}, \boldsymbol{a}_1$ $p(\boldsymbol{l}|\mathcal{C}_{\boldsymbol{l}})d\boldsymbol{z}_{t-1}d\boldsymbol{l}$.

We assume a dynamical model of the following form: $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_t, \boldsymbol{l}) = \mathcal{N}\left(\mathbf{A}_{t-1}\boldsymbol{z}_{t-1} + \mathbf{B}\boldsymbol{a}_t + \mathbf{C}\boldsymbol{l}, \boldsymbol{\Sigma}_{\text{trans}}\right)$ and denote the posterior from the previous time-step by $p(\boldsymbol{z}_{t-1}|\boldsymbol{w}_{1:t-1}, \boldsymbol{a}_{1:t-1}, \mathcal{C}_{\boldsymbol{l}}) = \mathcal{N}\left(\boldsymbol{z}_{t-1}^{+}, \boldsymbol{\Sigma}_{t-1}^{+}\right)$. Following Shaj, Becker, et al. 2020, we assume that the action $\boldsymbol{a}_t$ is known and not subject to noise.

At any time t, the posterior over the belief state $\boldsymbol{z}_{t-1}|\boldsymbol{w}_{1:t-1}, \boldsymbol{a}_{1:t-1}$, posterior over the latent task variable $\boldsymbol{l}|\mathcal{C}_{\boldsymbol{l}}$ and the action $\boldsymbol{a}_t$ are independent of each other since they form a "common effect" / v structure (Koller, Friedman, et al. 2007) with the unobserved variable $\boldsymbol{z}_t$. With these independencies and Gaussian assumptions, according to the Gaussian identity **??**, it can be shown that calculating the integral in equation 5.3.2 has a closed form solution as follows, $p(\boldsymbol{z}_t|\boldsymbol{w}_{1:t-1}, \boldsymbol{a}_{1:t}, \mathcal{C}_{\boldsymbol{l}}) = \mathcal{N}(\boldsymbol{z}_t^{-}, \boldsymbol{\Sigma}_t^{-})$, where $\boldsymbol{z}_t^{-} = \mathbf{A}_{t-1}\boldsymbol{z}_{t-1} + \mathbf{B}\boldsymbol{a}_t + \mathbf{C}\boldsymbol{l}$,
$\boldsymbol{\Sigma}_t^{-} = \mathbf{A}_{t-1}\boldsymbol{\Sigma}_{t-1}^{+}\mathbf{A}_{t-1}^{T} + \mathbf{C}\boldsymbol{\Sigma}_{\boldsymbol{l}}\mathbf{C}^{T} + \boldsymbol{\Sigma}_{\text{trans}}$.

Gaussian Identity 5.3.1 (Linear Combination Gaussian Marginalization)**.** *If* $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{u}} + \boldsymbol{b}, \boldsymbol{\Sigma}_{\boldsymbol{u}})$ *and* $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{v}}, \boldsymbol{\Sigma}_{\boldsymbol{v}})$ *are normally distributed independent random variables and if the conditional distribution* $p\left(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{v}\right) = \mathcal{N}(\mathbf{A}\boldsymbol{u} + \boldsymbol{b} + \mathbf{B}\boldsymbol{v}, \boldsymbol{\Sigma})$*, then marginal* $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{v})p(\boldsymbol{u})p(\boldsymbol{v}))d\boldsymbol{u}d\boldsymbol{v} = \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_{\boldsymbol{u}} + \boldsymbol{b} + \mathbf{B}\boldsymbol{\mu}_{\boldsymbol{v}}, \mathbf{A}\boldsymbol{\Sigma}_u\mathbf{A}^T + \mathbf{B}\boldsymbol{\Sigma}_{\boldsymbol{v}}\mathbf{B}^T + \boldsymbol{\Sigma})$.

Proof. The proof for the above identity is given in Appendix. □

Modelling Choices We use a locally linear transition model $\mathbf{A}_{t-1}$ as in Becker, Pandya, et al. 2019 (see Appendix A.2) and a nonlinear control model as in Shaj, Becker, et al. 2020 (Chapter 4). The local linearization around the posterior mean, can be interpreted as equivalent to an EKF. For the latent task transformation we can either use a linear, locally-linear or non-linear transformation. More details on the latent task transformation model can be found in the Appendix B.3.2. Our estimations (Figure 5.9) show that a nonlinear feedforward neural network $f(.)$ that outputs mean and variances and interacts additively gave the best performance in practice. $f(.)$ transforms the latent task moments $\boldsymbol{\mu_l}$ and $\boldsymbol{\sigma_l}$ directly into the latent space of the state-space model through additive interactions. The corresponding time update equations are given below: $\boldsymbol{z}_t^- = \mathbf{A}_{t-1}\boldsymbol{z}_{t-1}^+ + \mathbf{b}(\boldsymbol{a}_t) + \mathbf{f}(\boldsymbol{\mu}_l)$, $\boldsymbol{\Sigma}_t^- = \mathbf{A}_{t-1}\boldsymbol{\Sigma}_{t-1}^+\mathbf{A}_{t-1}^T + \mathbf{f}(\boldsymbol{\sigma_l}) + \boldsymbol{\Sigma}_{\text{trans}}$.

This chapter primarily aimed at accurately modeling mean predictions under non-stationary conditions, focusing on minimizing mean squared errors. We opted for locally linear transitions $\boldsymbol{A}_t$ and non-linear transformations f to enhance mean predictions, without prioritizing uncertainty quantification. In contrast, Chapter 6 equally emphasizes uncertainty quantification and mean predictions. Subsequent chapters adopted a simple linear model for both matrices $\boldsymbol{A}$ and $\boldsymbol{C}$, chosen for its robustness in both mean and uncertainty predictions and computational efficiency (Gu, Goel, and Re 2021; Mondal et al. 2023).

5.3.3. Inferring posterior latent states / Observation Update

The goal of this step is to compute the posterior belief $p(\boldsymbol{z}_t|\boldsymbol{o}_{1:t}, \boldsymbol{a}_{1:t}, \boldsymbol{C_l})$. We first map the observations at each time step $\boldsymbol{o}_t$ to a latent space using an observation encoder (Haarnoja et al. 2016; Becker, Pandya, et al. 2019) that emits latent features $\boldsymbol{w}_t$ along with uncertainty in those features via a variance vector $\boldsymbol{\sigma}_{obs}^t$. We then computed the posterior belief $p(\boldsymbol{z}_t|\boldsymbol{w}_{1:t}, \boldsymbol{a}_{1:t}, \boldsymbol{C_l})$, based on $p(\boldsymbol{z}_t|\boldsymbol{w}_{1:t-1}, \boldsymbol{a}_{1:t}, \boldsymbol{C_l})$ obtained from the time update, the latent observation $(\boldsymbol{w}_t, \boldsymbol{\sigma}_t^{obs})$ and the observation model $\mathbf{H}$. This is exactly the Kalman update step, which has a closed form solution as shown below for a time instant t,

$$\text{Kalman Gain: } \mathrm{Q}_t = \boldsymbol{\Sigma_t}^- \mathbf{H}^T \left(\mathbf{H}\boldsymbol{\Sigma_t}^- \mathbf{H}^T + \boldsymbol{I} \cdot {\sigma_\mathbf{t}}^{\text{obs}}\right)^{-1},$$

$$PosteriorMean : \boldsymbol{z_t}^+ = \boldsymbol{z_t}^- + \mathbf{Q}_t \left(\boldsymbol{w_t} - \mathbf{H}\boldsymbol{z_t}^-\right),$$

$$PosteriorCovariance : {\boldsymbol{\Sigma}_\mathbf{t}}^+ = \left(\mathbf{I} - \mathbf{Q}_t\mathbf{H}\right){\boldsymbol{\Sigma}_\mathbf{t}}^-,$$

where $\mathbf{I}$ denotes the identity matrix. This update is added as a layer in the computation graph (Haarnoja et al. 2016; Becker, Pandya, et al. 2019). However, the Kalman update involves computationally expensive matrix inversions of the order of $\mathcal{O}(L^3)$, where L is the dimension of the latent state $\boldsymbol{z}_t$. Thus, in order to make the

approach scalable, we follow the same factorization assumptions as in Becker, Pandya, et al. 2019. This factorization provides a simple way to reduce the observation update equation to a set of scalar operations, reducing the computational complexity from $\mathcal{O}(L^3)$ to $\mathcal{O}(L)$. More mathematical details on the simplified update equation can be found in Section 2.2.2.2.

From a computational perspective, this a Gaussian conditioning layer, similar to section 5.3.1. Both output a posterior distribution on latent variables $\boldsymbol{z}$, given a prior $p(\boldsymbol{z})$ and an observation model $p(\boldsymbol{w}|\boldsymbol{z})$, using the Bayes rule: $p(\boldsymbol{z}|\boldsymbol{w}) = p(\boldsymbol{w}|\boldsymbol{z})p(\boldsymbol{z})/p(\boldsymbol{w})$. This has a closed form solution because of Gaussian assumptions, which is coded as a layer in the neural network. The observation model is assumed to have the following structure, $P(\boldsymbol{w}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{H}\boldsymbol{z}, \boldsymbol{\Sigma}_{obs})$.

5.4. HiP-SSM as Adaptive Multi-Task World Models

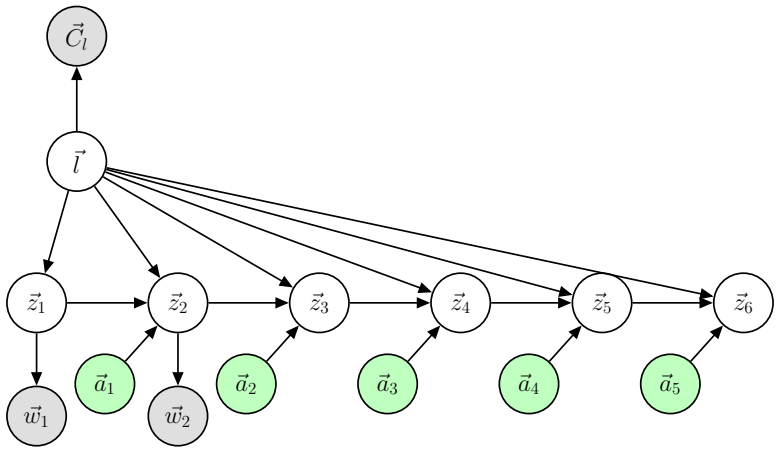


Figure 5.4.: Using adaptive internal world models to envision the future: Our predictions about future world states rely on more than just a sequence of action signals; they are also conditioned on latent task variables (l). These latent variables introduce additional causal factors that influence the dynamics, factors not considered in basic State Space Model (SSM) approaches introduced in Chapter 4. This inclusion enables a deeper and more nuanced ability to manipulate and foresee future states in diverse scenarios.

5.4.1. End To End Learning For Online Adaptation To Changes

Multitask Data Creation The latent task variable l models a distribution over functions (Garnelo et al. 2018), rather than a single function. In our case, these functions are latent dynamics functions. In order to train such a model, we use a training procedure that reflects this objective, where we form datasets consisting of timeseries, each with

different latent transition dynamics. Thus, we collect a set of trajectories over which the dynamics changes over time. These can be trajectories where a robot picks up objects of different weights or a robot traverses terrain of different slopes. Now, we introduce a multitask setting with a rather flexible definition of task, where each temporal segment of a trajectory can be considered to be a different "task" and the observed context set based on interaction histories from the past N time steps provides information about the current task setting. This definition allows us to have a potentially infinite number of tasks/local dynamical systems, and the distribution over these tasks/systems is modeled using a hierarchical latent task variable l. The formalism is based on the assumption that over these local temporal segments the dynamics is unchanging. This local consistency in dynamics holds for most real world scenarios (Nagabandi, Finn, and Levine 2018; Nagabandi, Clavera, et al. 2018). However, our formalism can model the global changes in dynamics at the test time, since we obtain a different instance of the HiP-RSSM for each temporal segment based on the observed context set. We also provide a detailed description of the multitask data set creation process in Table 1 and a pictorial illustration in Appendix 5.4.2.

Algorithm 1: Multi Task Dataset Creation For Training HiP-RSSM

Required:A set S of trajectories of changing dynamics
$D \leftarrow \phi$
for *each trajectory* $\tau \in S$ **do**
1. divide the trajectory τ into non-overlapping windows T_l of length N. Let $T = \{T_1, T_2, T_3, ., ., .\}$ be the list of all temporal segments/time-series.
for *each time window* $T_l \in T$ **do**
a) maintain a context set C_l consisting of N previous interactions;
b) update $D \leftarrow D \cup \{C_l, T_l\}$
end

end
Output: Output D consisting of batch of context and target sets.

Batch Training. Let $T \in \mathcal{R}^{B \times N \times D}$ be the batch of local temporal segments with different dynamics that we intend to model with the HiP-RSSM formalism. Given a target batch T, HiP-RSSM can be trained in a supervised manner similar to popular recurrent architectures such as LSTMs or GRUs. However, for each local temporal sequence $t \in T$, in which the dynamics is modeled, we also input a set of previous interactions N, which forms the context set $C \in \mathcal{R}^{B \times N \times D}$ to infer the latent task as explained in Section 5.3.1. Processing the context set C adds minimal additional computational / memory constraints, as we use a permutation-invariant set encoder. The set encoder allows for parallelization in processing the context set as opposed to recurrent processing of the target set.

The learnable parameters in the computation graph include the locally linear transition model $\boldsymbol{A}_t$, the nonlinear control factor $\boldsymbol{b}$, the linear/nonlinear latent task transformation model C, the transition noise $\boldsymbol{\Sigma}_{\text{trans}}$, along with the observation encoder, context encoder, and output decoder.

Loss Function. The network is tasked to minimize the prediction errors by maximizing the posterior predictive log-likelihood, which is given below for a single trajectory, i.e., $\mathrm{L} = \sum_{t=1}^{H} \log p(\boldsymbol{o}_{t+1} | \boldsymbol{w}_{1:t}, \boldsymbol{a}_{1:t}, \boldsymbol{C}_l)$
$= \sum_{t=1}^{H} \log \int p(\boldsymbol{o}_{t+1} | \boldsymbol{z}_{t+1}) p(\boldsymbol{z}_{t+1} | \boldsymbol{w}_{1:t}, \boldsymbol{a}_{1:t}, \boldsymbol{C}_l) d\boldsymbol{z}_{t+1}$

The extension to multiple trajectories is straightforward and was omitted to keep the notation uncluttered. Here, $\boldsymbol{o}_{t+1}$ are the ground truth observations at the time step $t+1$ that must be predicted from all observations up to the time step t.

Approximating the likelihood We employ a Gaussian approximation of the posterior predictive log-likelihood of the form $p(\boldsymbol{o}_{t+1} | \boldsymbol{w}_{1:t}, \boldsymbol{a}_{1:t}, \boldsymbol{C}_l) \approx \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{o}_{t+1}}, \mathrm{diag}(\boldsymbol{\sigma}_{\boldsymbol{o}_{t+1}}))$ where we use the mean of the prior belief $\boldsymbol{\mu}^{-}_{z_{t+1}}$ to decode the predictive mean, that is, $\boldsymbol{\mu}_{\boldsymbol{o}_{t+1}} = \mathrm{dec}_{\boldsymbol{\mu}}(\boldsymbol{\mu}^{-}_{z_{t+1}})$ and the variance estimate of the prior belief to decode the observation variance, that is, $\boldsymbol{\sigma}_{o_{t+1}} = \mathrm{dec}_{\sigma}(\boldsymbol{\Sigma}^{-}_{z_{t+1}})$. This approximation can be motivated by a moment matching perspective and allows end-to-end optimization of logarithmic likelihood without using auxiliary objectives such as the ELBO Becker, Pandya, et al. 2019. Thus, the approximate Gaussian predictive log-likelihood for a single sequence is then computed as $\mathrm{L}\Big(\boldsymbol{o}_{(1:T)}\Big) =$
$1T \sum_{t=1}^{T} \log \mathcal{N}\left(\boldsymbol{o}_t \middle| \mathrm{dec}_{\mu}(\boldsymbol{z}_t^{+}), \mathrm{dec}_{\Sigma}(\boldsymbol{\sigma}_t^{\mathrm{u},+}, \boldsymbol{\sigma}_t^{\mathrm{s},+}, \boldsymbol{\sigma}_t^{\mathrm{l},+})\right)$, where $\mathrm{dec}_{\mu}(\cdot)$ and $\mathrm{dec}_{\Sigma}(\cdot)$ denote the parts of the decoder that are responsible for decoding the latent mean and latent variance respectively.

Variations Of The Learning Objectives We optimize the root mean square error (RMSE) between the decoder output and the ground-truth states. As in Shaj, Becker, et al. 2020 we use the differences to the next state as our ground truth states, as this results in better performance for dynamic learning, especially at higher frequencies. In principle, we could train on the Gaussian log-likelihood instead of the RMSE and hence model the variances. Training in RMSE yields slightly better predictions and allows a fair comparison with deterministic baselines that use the same loss, such as feedforward neural networks, LSTMs and metalearning algorithms such as MAML (Finn, Abbeel, and Levine 2017). Therefore, we report results with the RMSE loss.

Gradients are computed using (truncated) backpropagation through time (BPTT) (Werbos 1990) and clipped. We optimize the objective using the Adam (Kingma and Ba 2014) stochastic gradient descent optimizer with default parameters.

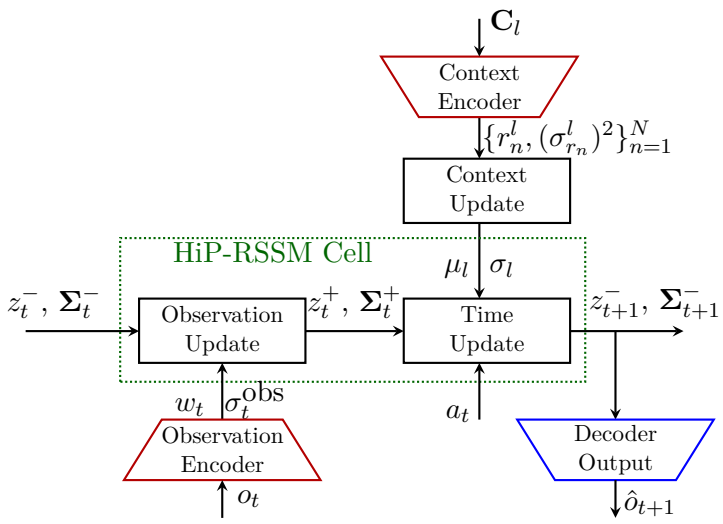


Figure 5.5.: Depicts the schematic of HiP-RSSM. The output of the task conditional 'Time Update' stage, which forms the prior for the next time step $(z_{t+1}^-, \Sigma_{t+1}^-)$ is decoded to get the prediction of the next observation.

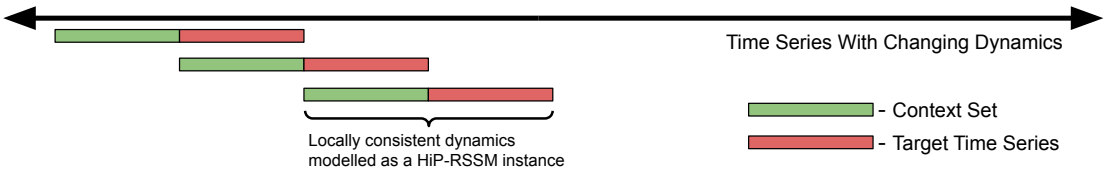


Figure 5.6.: HiP-RSSM Inference Under Changing Dynamics Scenarios

Irrespective of the learning objectives/loss functions used, we still have to model uncertainty in HiP-SSM latent state as the context set and observations do not contain the full task and state information of the system.

The architecture of the HiP-SSM is summarized in Figure 4.3.

5.4.2. HiP-RSSM during Test Time / Inference

We perform inference using HiP-RSSM at test time on a trajectory with varying dynamics using algorithm 2. A pictorial representation of the same is given in the Figure 5.6. We use this inference scheme to visualize how the latent variable l, that describe different instances of a HiP-RSSM over different temporal segments, evolve at a global level. The visualizations are reported in Figures 5.10a and 5.10b in the experiments section discussed below.

Algorithm 2: HiP-RSSM Test Time Inference

Required: Trained HiP-RSSM Model
Required: A time series τ of length $K >> N$
Divide the time series τ into non-overlapping windows T_l of length N. Let $T = \{T_1, T_2, T_3, ., ., .\}$ be the ordered list of all temporal segments, sorted in the ascending order of time of occurrence.
foreach *each time window* $T_l \in T$ **do**

1. maintain a context set C_l consisting of N previous interactions;
2. infer posterior latent task variable $p(\boldsymbol{l}|\mathcal{C}_l)$ using context update stage as in section 5.3.1;
3. using the posterior over latent task variable $\boldsymbol{l}|\mathcal{C}_l$ and observations in sequence T_l to perform sequential Bayesian inference over the state space model using Kalman observation update (5.3.3) and task conditional Kalman time update; (5.3.2)

end

5.5. Experiments

This section evaluates our approach on a diverse set of dynamical systems from the robotics domain in simulations and real systems. We show that HiP-RSSM outperforms contemporary recurrent state-space models (RSSMs) and recurrent neural networks (RNNs) by a significant margin under changing dynamics scenarios. Further, we show that HiP-RSSM outperforms these models even under situations with partial observability/ missing values. We also baseline our HiP-RSSM with contemporary multi-task models and improve performance, particularly in modelling non-Markovian dynamics and under partial observability. Finally, the visualizations of the Gaussian latent task variables in HiP-RSSM demonstrates that they learn meaningful representations of the causal factors of variations in dynamics in an unsupervised fashion.

We consider the following baselines:

- RNNs - We compare our method to two widely used recurrent neural network architectures, LSTMs (Hochreiter and Schmidhuber 1997) and GRUs (Cho et al. 2014).
- RSSMs - Among several RSSMs from the literature, we chose RKN Becker, Pandya, et al. 2019 as these have shown excellent performance for dynamics learning (Shaj, Becker, et al. 2020) and relies on exact inference as in our case.
- Multi Task Models - We also compare with state of the art multi-task learning models like Neural Processes (NPs) and MAML (Nagabandi, Clavera, et al. 2018). Both models receive the same context information as in HiP-RSSM.

	No Imputation	50% Imputation
HiP-RSSM	$\mathbf{2.30 \pm 0.043}$	$\mathbf{2.47 \pm 0.012}$
RKN	3.088 ± 0.046	3.223 ± 0.014
LSTM	3.108 ± 0.041	3.630 ± 0.097
GRU	3.287 ± 0.013	3.621 ± 0.047
FFNN	8.150 ± 0.047	-
NP	5.526 ± 0.030	-
MAML	7.314 ± 0.021	-

(a) Pneumatic RMSE $\left(10^{-3}\right)$

	No Imputation	50 % Imputation
HiP-RSSM	$\mathbf{2.833 \pm 0.024}$	$\mathbf{2.843 \pm 0.024}$
RKN	3.392 ± 0.062	3.398 ± 0.062
LSTM	3.503 ± 0.006	3.736 ± 0.062
GRU	3.407 ± 0.02	3.642 ± 0.153
FFNN	3.313 ± 0.018	-
NP	$\mathbf{2.765 \pm 0.004}$	-
MAML	3.202 ± 0.006	-

(b) Franka RMSE $\left(10^{-4}\right)$

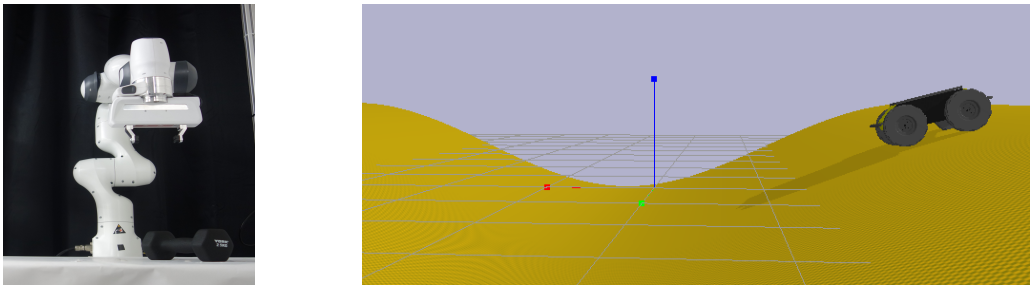
Table 5.1.: Prediction Error in RMSE for (a) pneumatic muscular arm (5.5.1) and (b) Franka Arm manipulating varying loads (5.5.2) for both fully observable and partially observable scenarios.

In case of recurrent models we replace the HiP-RSSM cell with a properly tuned LSTM, GRU and RKN Cell respectively, while fixing the encoder and decoder architectures. For the NP baseline we use the same context encoder and aggregation mechanism as in HiP-RSSM to ensure a fair comparison. We create partially observable settings by imputing 50% of the observations during inference. More details regarding the baselines and hyperparameters can be found in the Appendix D.2.

5.5.1. Soft Robot Playing Table Tennis

We first evaluate our model on learning the dynamics of a pneumatically actuated muscular robot. This four Degree of Freedom (DoF) robotic arm is actuated by Pneumatic Artificial Muscles (PAMs) (Büchler, Ott, and Peters 2016). The data consists of trajectories of hitting movements with varying speeds while playing table tennis (Büchler, Guist, et al. 2020). This robot's fast motions with high accelerations are complicated to model due to hysteresis and hence require recurrent models (Shaj, Becker, et al. 2020).

We show the prediction accuracy in the RMSE in Table B.1a. We observe that the HiP-RSSM can outperform the previous state of the art predictions obtained by recurrent models. Based on domain knowledge, we hypothesize that the latent context variable captures multiple unobserved causal factors of variation that affect the dynamics in the latent space, which are not modelled in contemporary recurrent models. These causal factors could be, in particular, the temperature changes or the friction due to a different path that the Bowden trains take within the robot. Disentangling and interpreting these causal factors can be exciting and improve generalization, but it is out of scope for the current work. Further, we find that the multitask models like NPs and MAML fail to model these dynamics accurately compared to all the recurrent baselines because of the non-markovian dynamics resulting from the high accelerations in this pneumatically actuated robot.



(a) (b)

Figure 5.7.: Figures of a subset of the agents used for collecting multi-task datasets. (left) Franka manipulator that was used to collect trajectories with different loads attached. (b) Mobile robot traversing terrain with varying slopes.

5.5.2. Robot Manipulation With Varying Loads

We collected data from a 7 DoF Franka Emika Panda manipulator carrying six different loads at its end effector. It involved a mix of movements of different velocities from slow to swift movements that covered the entire robot workspace. We chose trajectories with four different loads as training set and evaluated the performance on two unseen weights, which results in a scenario where the dynamics change over time. Here, the causal factor of variation in dynamics is the weights attached to the end-effector and assumed to be unobserved.

We show the prediction errors in RMSE in Table 5.2. HiP-RSSMs outperform all recurrent state-space models, including the RKN and deterministic RNNs, in modelling these dynamics under fully observable and partially observable conditions. The multi-task baselines of NPs and MAML perform equally well under full observability for this task because of the near Markovian dynamics of the Franka Manipulator, which often does not need recurrent models. However, HiP-RSSMs have an additional advantage in that they are naturally suited for partially observable scenarios and can predict ahead in a compact latent space, a critical component for recent success in model-based control and planning (Hafner et al. 2019).

5.5.3. Robot Locomotion In Terrains Of Different Slopes

Wheeled mobile robots are the most common types of robots being used in exploration of unknown terrains where they may face uneven and challenging terrain. We set up an environment using a Pybullet simulator (Coumans and Bai 2016) where a four-wheeled mobile robot traverses an uneven terrain of varying steepness generated by sinusoidal functions (Sonker and Dutta 2020) as shown in 5.7b. This problem is challenging due to the highly non-linear dynamics involving wheel-terrain interactions. In addition, the varying steepness levels of the terrain results in a changing dynamics scenario, which further increases the complexity of the task.

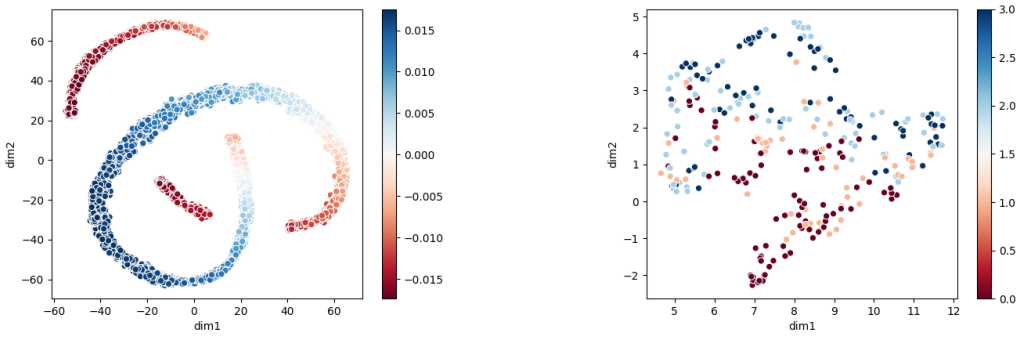


Figure 5.8.: Figure shows the tSNE (Van der Maaten and G. Hinton 2008) plots of the latent task embeddings produced from randomly sampled instances of HiP-RSSM for two different robots. (a) The wheeled robot discussed in section 5.5.3 traversing terrains of varying slopes. The color coding indicates the average gradients of the terrain for each of these instances. These can have either positive or negative values. (b) The Franka manipulator with loads of varying weights attached to the end-effector. The color coding indicated weights ranging from 0 to 3 kilograms.

We show the prediction errors in RMSE in Table 5.2. When most recurrent models, including RSSMs and deterministic RNNs, fail to model these dynamics, HiP-RSSMs are by far the most accurate in modelling these challenging dynamics in the latent space. Further, the HiP-RSSMs perform much better than state-of-the-art multi-task models like NPs and MAML.

	No Imputation	50% Imputation
HiP-RSSM	$\mathbf{2.96 \pm 0.212}$	$\mathbf{6.15 \pm 0.327}$
RKN	7.17 ± 0.017	14.66 ± 0.224
LSTM	9.14 ± 0.026	51.21 ± 0.431
GRU	9.216 ± 0.073	53.14 ± 0.242
FFNN	8.72 ± 0.021	-
NP	4.57 ± 0.013	-
MAML	5.04 ± 0.051	-

Table 5.2.: Prediction error for wheeled mobile robot trajectories in RMSE (10^{-5}) for both fully observable and partially observable scenarios.

We finally visualize the latent task representations using TSNE in Figure 5.8a. As seen in the plot, the HiP-SSM instances under similar terrain slopes cluster in the latent task space, indicating that the model correctly identifies causal effects in the dynamics in an unsupervised fashion.

5.5.4. Ablation Study for Latent Task Conditioning.

We evaluated the performance of the model for different action conditioning schemes (linear, locally-linear and non-linear) as discussed in Section B.3.2. The resulting evaluation can be seen in Figure 5.9 and shows the advantage of using nonlinear

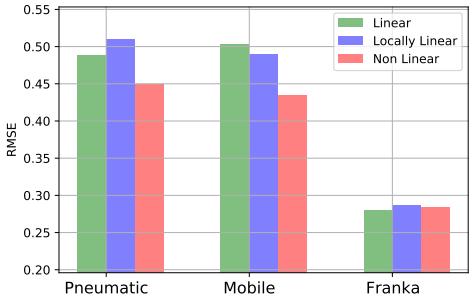


Figure 5.9.: An ablation on the performance of different task transformation models discussed in Sections 5.3.2 and B.3.2.

models for the additive task conditioning of latent dynamics to obtain accurate mean predictions under RMSE loss.

5.5.5. Ablation On Context Encoders For Inferring Task Abstractions

In table 5.3, we report the details of evaluating our Bayesian aggregation-based context set encoder (discussed in 5.3.1) against a causal / recurrent encoder that takes into account the temporal structure of the context data due to its sequential nature. We used a probabilistic recurrent encoder (Becker, Pandya, et al. 2019), whose mean and variance from the last time step is used to infer the posterior latent task distribution $p(\boldsymbol{l}|\mathcal{C}_l) = \mathcal{N}(\boldsymbol{\mu_l}, \text{diag}(\boldsymbol{\sigma}_l^2))$. The dimensions of the latent task parameters obtained from both the set and the recurrent encoders remain the same (60).
The reported experiments are conducted on data from a wheeled mobile robot discussed in Section 5.5.3. As reported in Table 5.3, the permutation-invariant set encoder outperforms the recurrent encoder by a good margin in terms of prediction accuracy for both fully and partially observable scenarios. Additionally, the set encoder is far more efficient in terms of computational time required for training, as seen from the time taken per epoch for each of these cases, as it allows for efficient parallelization.

Table 5.3.: Comparison between the permutation invariant set encoder and recurrent encoder. The performance is measured in terms of prediction RMSE (10-5) and mean of the training time per epoch (in seconds) over 5 runs.

	No Imputation RMSE	50% Imputation RMSE	Training Time Per Epoch
Set Encoder	$\mathbf{2.96 \pm 0.212}$	$\mathbf{6.15 \pm 0.327}$	**6.71**
Recurrent Encoder	5.10 ± 0.041	10.12 ± 0.112	14.13

5.5.6. Visualizing Changing Hidden Parameters At Test Time Over Trajectories With Varying Dynamics

We finally perform inference using the trained HiP-RSSM in a multi-task / changing dynamics scenario where the dynamics continuously changes over time. We use the inference procedure described in appendix 5.4.2 based on a fluid definition for "task" as the local dynamics in a temporal segment. We plot the global variation in the latent task variable captured by each instance of the HiP-RSSM over these local temporal segments using the dimensionality reduction technique UMAP (McInnes, Healy, and Melville 2018). As seen in Figures 5.10a and 5.10b, the latent task variable captures these causal factors of variations in an interpretable manner.

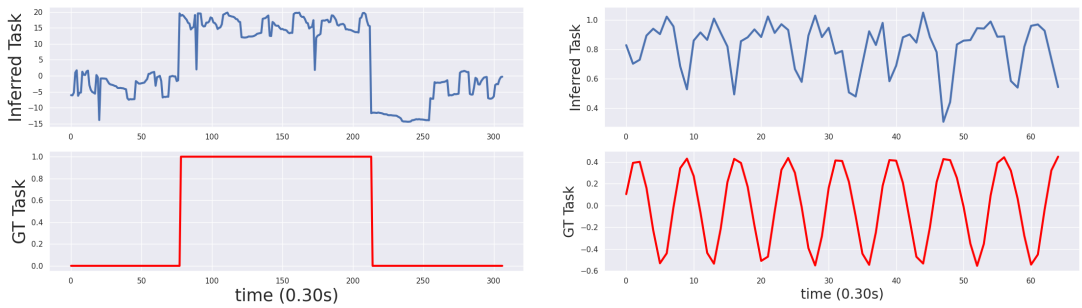


Figure 5.10.: (a) and (b) shows how the one dimensional UMAP (McInnes, Healy, and Melville 2018) embedding of the inferred latent task variable (top blue plots) changes at test time when an agent undergoes changes in its dynamics for the franka robot and mobile robot respectively. An indicator of the Ground Truth (GT) Task (bottom red plots) variables are also given. In case of the Franka Robot, the groundtruth (GT) tasks denotes the switching of dynamics between 0 kg (free motion) and 2.5 kg loads. In case of the mobile robot the groundtruth (GT) tasks denoted slopes of the terrain.

5.6. Conclusion

We proposed HiP-RSSM, a probabilistically principled recurrent neural network architecture for modelling the world under non-stationary dynamics. We start by formalizing a new framework called the Hidden Parameter State Space Model (HiP-SSM), to address the multi-task state-space modelling setting. HiP-SSM assumes a shared latent state and action space across tasks but additionally assumes latent structure in the dynamics. We exploit the structure of the resulting Bayesian network to learn a universal dynamics model with latent parameter l via exact inference and backpropagation through time. The resulting recurrent neural network, namely HiP-RSSM, learns to cluster SSM instances with similar dynamics together in an unsupervised fashion. Our experimental results on various robotic benchmarks show that HiP-RSSMs significantly outperform state-of-the-art recurrent neural network architectures on dynamics modelling tasks. We believe that modelling the dynamics in the latent space which disentangles the state, action and task representations can benefit multiple future applications including latent planning/control under non-stationary dynamics and causal factor identification.

6. Multi Time Scale SSMs: Hierarchical World Models at Multiple Temporal Abstractions

This chapter is based on "Multi Time Scale World Models" (Shaj, Gholam Zadeh, et al. 2023).

This chapter of the thesis explores the concept of "hierarchical depth" within generative models of the world by structuring them through nested causal hierarchies that span across different spatio-temporal scales (see Figure 6.1). This approach addresses a significant limitation found in the prevailing literature on world models: the reliance on a singular hierarchical depth or time scale. Present formulations in the fields of machine learning and artificial intelligence typically operate with a single, finely detailed temporal scale, such as milliseconds. Singular time scale often limits accurate prediction and planning over long hoizons (LeCun 2022). For efficient long-term prediction and planning, the model must predict at multiple levels of temporal abstractions (Sutton 1995; Precup and Sutton 1997; LeCun 2022).

The world is rife with regularities. Day follows night, seasons follow one another, milk goes sour, faulty brakes are often followed by accidents, and so on. Regularities come at different time scales, ranging from tens of milliseconds to hundreds, to seconds, minutes, and upwards towards regularities or rules that are stable over weeks, months, and years. For example, commuting to work involves planning at a higher level slow time scale (in minutes or hours) like which route to drive (or walk) and at a lower level fast time scale (in seconds or milliseconds) like how to brake/steer or choose precise skeletal muscle actions. Similarly, in robotic manipulation, the robot must be able to perform precise and coordinated movements to grasp and manipulate the object at a fast time scale while at a slower time scale the robot must also be able to recognize and utilize higher-level patterns and structures in the task, such as the shape, size, and location of objects, and the overall goal of the manipulation task. Generative models with "temporal depth" in the form of nested hierarchies offers several advantages in dealing with such scenarios (Lee and Mumford 2003; Hohwy 2013; LeCun 2022) as listed below:

- **Reflection of Causal Depth**: The world is inherently structured hierarchically (Hohwy 2013), with causes and effects nested within one another on various

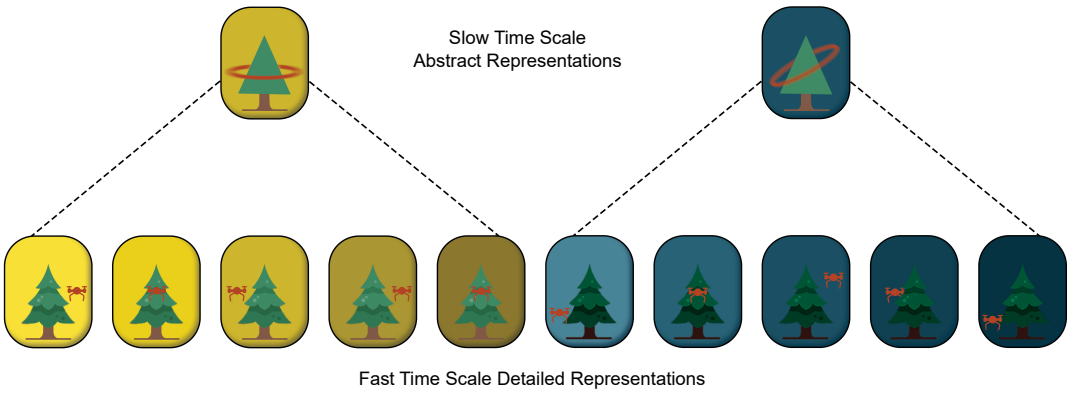


Figure 6.1.: Conceptual Depiction of Hierarchical Temporal Abstraction in World Models. At the lower level, represented by the array of yellow and blue squares, we observe a drone's movement patterns around a central tree—circular during the day (indicated by a yellow background) and elliptical at night (indicated by a blue background). The upper tier conveys a more abstracted temporal perspective, encapsulating more invariant representations without the granular detail. This abstraction captures the cyclical day-to-night transition and the drone's corresponding behavioral patterns: circular in daylight, elliptical in darkness. This model illustrates how complex, time-variant dynamics are captured within multi-scale temporal frameworks. The intuition is formalized in Section 6.2.

scales of time and space (for example, in autonomous driving applications, slow regularities like traffic congestion trends including peak and off-peak hours, holiday traffic patterns, and seasonal changes in road conditions affect the fast regularities like real-time decisions on route navigation and speed adjustments). A hierarchical model allows the representation of this causal depth, enabling intelligent agents to understand not just immediate sensory inputs but also the broader and interconnected causal relationships that govern the external world.

- **Better Long Term Prediction**: By organizing regularities in the world from faster, detailed levels to slower, more abstract levels, hierarchical models enable predictions at multiple scales of precision and time. This flexibility is fundamental to adaptive behavior, allowing for both immediate, detailed responses and longer-term planning based on abstracted patterns of regularity (K. Friston 2008; Hohwy 2013). Fast changing regularities are good for detail; slower regularities are more general and abstract. This makes sense when we consider what regularities allow us to predict. If I want to predict something with great perceptual precision, then I cannot do it very far into the future, so I need to rely on a fast changing regularity. On the other hand, predictions further into the future come at a loss of precision and often detail.

- **Adaptability and Transferability**: As seen in Chapter 5 hierarchical temporal abstractions can capture relevant task structures across dynamical systems under non-stationary conditions, which can be used to identify similarities and differences between tasks. Learning dynamics of these nested hierarchical abstractions allow

for making "predictions about predictions" and subsequent transfer of knowledge learned from one task to another (Shanahan and Mitchell 2022; LeCun 2022). This adaptability is key to navigating a world that is constantly changing, allowing for the refinement of predictions and behaviors in response to new information.

This chapter of the dissertation attempt to come up with a principled probabilistic formalism for learning such multi-time scale world models as a hierarchical sequential latent variable model. We show that such models can better capture the complex, nonlinear dynamics of a system more efficiently and robustly than models that learn on a single timescale. This is exemplified in several challenging simulated and real-world prediction tasks such as the D4RL dataset, a simulated mobile robot, and real manipulators including data from heavy machinery excavators.

6.1. Related Work

Multi Time Scale World Models One of the early works that enabled environment models at different temporal scales to be intermixed, producing temporally abstract world models was proposed by Sutton 1995. The work was limited to tabular settings but showed the importance of learning environment dynamics at multiple abstractions. However, there have been limited works that actually solve this problem at scale as discussed in LeCun 2022. A probabilistically principled formalism for these has been lacking in literature and this work is an early attempt to address this issue.

Deep State Space Models. Deep SSMs combine the benefits of deep neural nets and SSMs by offering tractable probabilistic inference and scalability to large and high-dimensional datasets. Haarnoja et al. 2016; Becker, Pandya, et al. 2019; Shaj, Becker, et al. 2020 use neural network architectures based on exact inference on SSMs and perform state estimation and dynamics prediction tasks. Shaj, Buchler, et al. 2022 extend these models to modelling non-stationary dynamics. Krishnan, Shalit, and Sontag 2017; Karl et al. 2016; Hafner et al. 2019 perform learning and inference in SSMs using variational approximations. However, most of these recurrent state-space models have been evaluated on very short-term prediction tasks in the range of a few milliseconds and model the dynamics at a single time scale.

Transformers Recent advances in Transformers (Vaswani et al. 2017; Radford et al. 2019; Brown et al. 2020), which rely on the attention mechanism, have demonstrated superior performance in capturing long-range dependency compared to RNN models in several domains, including time series forecasting (Zhou et al. 2021; Liu et al. 2022) and learning world models (Micheli, Alonso, and Fleuret 2023). Zhou et al. 2021; Liu et al. 2022; Nie et al. 2023 use transformer architectures based on a direct multistep loss (Zeng et al. 2022) and show promising results for long-term forecasting since they

avoid the accumulation of errors from autoregression. On the other hand Micheli, Alonso, and Fleuret 2023 uses a GPT-like autoregressive version of transformers to learn world models. These deterministic models, however, do not deal with temporal abstractions and uncertainty estimation in a principled manner. Nevertheless, we think Transformers that operate at multiple timescales based on our formalism can be a promising alternative research direction.

6.2. Multi Time Scale State Space Models

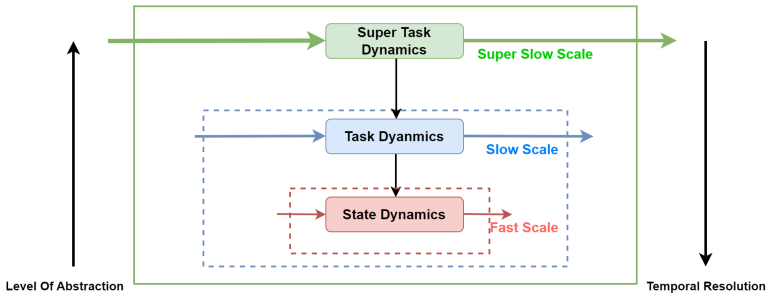


Figure 6.2.: Regularities in our world can be ordered hierarchically, from faster to slower (Hohwy 2013). Levels in the hierarchy can be connected such that certain slow regularities, at higher levels, pertain to relevant lower level, faster regularities. A complete such hierarchy would reveal the causal structure and depth of the world—the way causes interact and nest with each other across spatiotemporal scales.

Our goal is to learn a principled sequential latent variable model that can model the dynamics of artificial embodied agents (specifically robotic systems) under multiple levels of temporal abstractions based on observed sensory data. To do so, we introduce a new formalism, called Multi-Time Scale State Space (MTS3) Model, with the following desiderata: i) It is capable of modeling dynamics at multiple time scales. ii) It allows for a single global model to be learned that can be shared across changing configurations of the environments. iii) It can provide accurate long-term predictions and uncertainty estimates. iv) It is probabilistically principled, yet scalable during learning and inference.

We start by introducing a formal definition of a 2 time scale Multi-time scale SSM in Section 6.2.1 and then proceed to a general definition of an MTS3 with an arbitrary number of hierarchies/timescales in Section 6.2.4.

6.2.1. Formal Definition Of a 2-Level MTS3

An MTS3 model with 2 timescales is defined by two SSMs on a fast and a slow time scale respectively. Both SSMs are coupled via the latent state of the slow time scale SSM, which parameterizes / "reconfigures" the system dynamics of the fast time scale SSM. While the fast time scale SSM runs at the original time step Δt of the dynamical system,

the slow time scale SSM is only updated every H step, i.e., the slow time scale time step is given by $H\Delta t$. We will derive closed-form Gaussian inference for obtaining the beliefs for both time scales, resulting in variations of the Kalman update rule which are also fully differentiable and used to back-propagate the error signal (Becker, Pandya, et al. 2019; Haarnoja et al. 2016). The definition with a 2-level MTS3 along with the inference and learning schemes that we propose is directly extendable to an arbitrary number of temporal abstractions by introducing additional feudal (Dayan and G. E. Hinton 1992) hierarchies with longer discretization steps and is further detailed in Section 6.2.4.

6.2.1.1. Fast time-scale SSM

The fast time-scale (fts) SSM is given by $\mathcal{S}_{\text{fast}} = (\mathcal{Z}, \mathcal{A}, \mathcal{O}, f_{\boldsymbol{l}}^{\text{fts}}, h^{\text{fts}}, \Delta t, \mathcal{L})$. Here, $\boldsymbol{l} \in \mathcal{L}$ is a task descriptor that parameterizes the dynamics model of the SSM and is held constant for H steps. We will denote the task descriptor for the kth time window of H steps as $\boldsymbol{l}_k$. The probabilistic dynamics and observation model of the fast time scale for the tth time step in the kth window can then be described as
$\mathrm{p}(\boldsymbol{z}_{k,t}|\boldsymbol{z}_{k,t-1}, \boldsymbol{a}_{k,t-1}, \boldsymbol{l}_k) = \mathcal{N}(f_{\boldsymbol{l}}^{\text{fts}}(\boldsymbol{z}_{k,t-1}, \boldsymbol{a}_{k,t-1}, \boldsymbol{l}_k), \mathbf{Q})$, and
$p(\boldsymbol{o}_{k,t}|\boldsymbol{z}_{k,t}) = \mathcal{N}(h^{\text{fts}}(\boldsymbol{z}_{k,t}), \mathbf{R})$.

Task-conditioned marginal transition model. Moreover, we have to consider the uncertainty in the task descriptor (which will, in the end, be estimated by the slow time scale model), i.e., instead of considering a single task descriptor $\boldsymbol{l}_k$, we have to consider a distribution over task descriptors $p(\boldsymbol{l}_k)$ for inference in the fts-SSM. This distribution will be provided by the slow-time scale SSM for every time window k. We can further define the marginal task-conditioned transition model for the time window k that is given by
$\mathrm{p}_{\boldsymbol{l}_k}(\boldsymbol{z}_{k,t}|\boldsymbol{z}_{k,t-1}, \boldsymbol{a}_{k,t-1}) = \int p(\boldsymbol{z}_{k,t}|\boldsymbol{z}_{k,t-1}, \boldsymbol{a}_{k,t-1}, \boldsymbol{l}_k) p(\boldsymbol{l}_k) d\boldsymbol{l}_k$

Latent observations. Following Becker, Pandya, et al. 2019, we replace the observations by latent observations and their uncertainty, i.e., we use latent observation encoders to obtain $\boldsymbol{w}_{k,t} = \text{enc}_w(\boldsymbol{o}_{k,t})$ and an uncertainty encoder $\boldsymbol{\sigma}_{k,t} = \text{enc}_\sigma(\boldsymbol{o}_{k,t})$. The observation model is thus given by $p(\boldsymbol{w}_{k,t}|\boldsymbol{z}_{k,t}) = \mathcal{N}(h^{\text{fts}}(\boldsymbol{z}_{k,t}), \text{diag}(\boldsymbol{\sigma}_{k,t}))$.

6.2.1.2. Slow time-scale SSM

The slow time scale (sts) SSM only updates every time step H and uses the task parameter $\boldsymbol{l}$ as a latent state representation. Formally, the SSM is defined as $\mathcal{S}_{\text{slow}} = (\mathcal{L}, \mathcal{E}, \mathcal{T}, f^{\text{sts}}, h^{\text{sts}}, H\Delta t)$. It uses an abstract observation $\boldsymbol{\beta} \in \mathcal{B}$ and abstract action $\boldsymbol{\alpha} \in \mathcal{A}$ that summarize the observations and actions, respectively, throughout the current time window. The general dynamics model is hence given by $\mathrm{p}(\boldsymbol{l}_k|\boldsymbol{l}_{k-1}, \boldsymbol{\alpha}_k) = \mathcal{N}(f^{\text{sts}}(\boldsymbol{l}_{k-1}, \boldsymbol{\alpha}_k), \boldsymbol{S})$.

Although there exist many ways to implement the abstraction of observations and actions of time windows, we choose to use a consistent formulation by fusing the information from all H time steps of time window k using Gaussian conditioning.

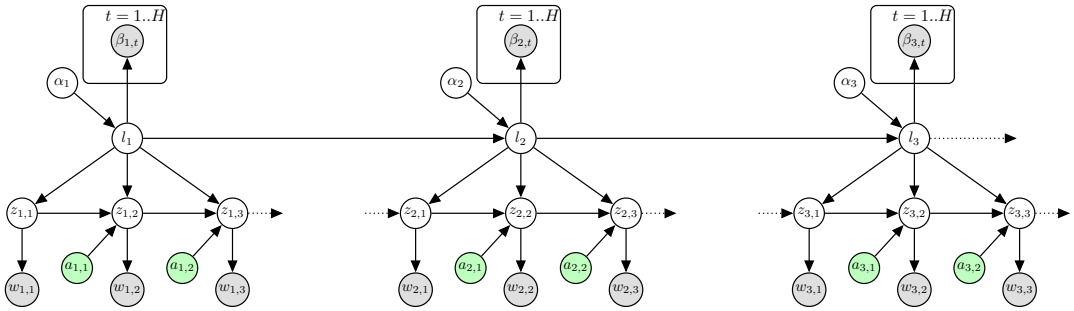


Figure 6.3.: The graphical model corresponding to an MTS3 with 2 timescales. The latent task variable $\boldsymbol{l}_k$ captures the slow changing dynamics using abstract observation inferred from $\{\boldsymbol{\beta}_{k,t}\}_{t=1}^{H}$and abstract action $\boldsymbol{\alpha}_k$ as described in Section 6.2.1.2. The inference in the fast time scale uses primitive observations $\boldsymbol{w}_{k,t}$, primitive actions $\boldsymbol{a}_{k,t}$ and the latent task descriptor l_k which parameterizes the fast-changing dynamics of $\boldsymbol{z}_{k,t}$ for a time window k as discussed in the section 6.2.2.

Observation abstraction. In terms of the abstract observation model, we choose to model H observations $\boldsymbol{\beta}_{k,t}$, $t \in [1, H]$ for a single slow-scale time step k. All these observations can then be straightforwardly integrated into the belief state representation using incremental observation updates. The abstract observation and its uncertainty for time step t is again obtained by an encoder architecture, i.e,

$$\boldsymbol{\beta}_{k,t} = \text{enc}_\beta(\boldsymbol{o}_{k,t}, t), \quad \boldsymbol{\nu}_{k,t} = \text{enc}_\nu(\boldsymbol{o}_{k,t}, t),$$

and $p(\boldsymbol{\beta}_{k,t}|\boldsymbol{l}_k) = \mathcal{N}(h^{\text{sts}}(\boldsymbol{l}_k), \text{diag}(\boldsymbol{\nu}_{k,t}))$. Hence, the abstract observation $\boldsymbol{\beta}_{k,t}$ contains the actual observation $\boldsymbol{o}_{k,t}$ at time step t as well as a temporal encoding for the time-step. Although multiple Bayesian observation updates are permutation invariant, the temporal encoding preserves the relative time information between the observations, similar to current transformer architectures.

Action abstraction. The abstract action $\boldsymbol{\alpha}_k$ causes transitions to latent task $\boldsymbol{l}_k$ from $\boldsymbol{l}_{k-1}$. It should contain relevant information of all primitive actions $\boldsymbol{a}_{k,t}$, $t \in [1, H]$ executed in the time window k. To do so, we again use Bayesian conditioning and latent action encoding. Each control action $\boldsymbol{a}_{k,t}$ and the encoding of the time step t are encoded in its latent representation and its uncertainty estimate, i.e.,

$$\boldsymbol{\alpha}_{k,t} = \text{enc}_\alpha(\boldsymbol{a}_{k,t}, t), \quad \boldsymbol{\rho}_{k,t} = \text{enc}_\rho(\boldsymbol{a}_{k,t}, t).$$

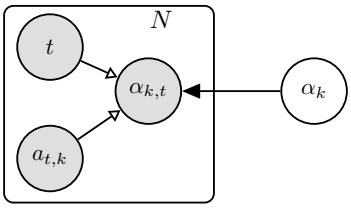


Figure 6.4.: Generative model for the abstract action α_k. The hollow arrows are deterministic transformations leading to implicit distribution $\alpha_{k,t}$ using an action set encoder.

Single latent actions $\boldsymbol{\alpha}_{k,t}$ can be aggregated into a consistent representation $\boldsymbol{\alpha}_k$ using Bayesian aggregation Volpp et al. 2020. To do so, we use the likelihood $p(\boldsymbol{\alpha}_{k,t}|\boldsymbol{\alpha}_k) = \mathcal{N}(\boldsymbol{\alpha}_k, \text{diag}(\boldsymbol{\rho}_{k,t}))$ and obtain the posterior $p(\boldsymbol{\alpha}_k|\boldsymbol{\alpha}_{k,1:H}) = \mathcal{N}(\boldsymbol{\mu}_{\alpha_k}, \boldsymbol{\Sigma}_{\alpha_k})$, which is obtained by following the standard Bayesian aggregation equations, see Appendix C.2.1. Note that our abstract action representation also contains an uncertainty estimate which can be used to express different effects of the actions on the uncertainty of the prediction. Due to the Gaussian representations, we can compute the marginal transition model $\text{p}_{\boldsymbol{\alpha}_k}(\boldsymbol{l}_k|\boldsymbol{l}_{k-1}, \boldsymbol{\alpha}_{k,1:H}) = \int p_{\boldsymbol{\alpha}_k}(\boldsymbol{l}_k|\boldsymbol{l}_{k-1}, \boldsymbol{\alpha}_k) p(\boldsymbol{\alpha}_k|\boldsymbol{\alpha}_{k,1:H}) d\boldsymbol{\alpha}_k$. This transition model is used for inference and its parameters are learned.

6.2.1.3. Connecting both SSMs via inference

In the upcoming sections, we will devise Bayesian update rules to obtain the prior $p(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k-1}, \boldsymbol{\alpha}_{1:k})$ and posterior $p(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k}, \boldsymbol{\alpha}_{1:k})$ belief state for the sts-SSM as well as the belief states for the fts-SSM. The prior belief $p(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k-1}, \boldsymbol{\alpha}_{1:k})$ contains all information up to time window $k-1$ and serves as a distribution over the task-descriptor of the fts-SSM, which connects both SSMs. This connection allows us to learn both SSMs jointly in an end-to-end manner.

The probabilistic graphical model of our MTS3 model is depicted in Figure 6.8. In the next section, we will present the detailed realization of each SSM to perform closed-form Gaussian inference and end-to-end learning on both time scales.

6.2.2. Inference in the Fast Time-Scale SSM

The fts-SSM performs inference for a given time window k of horizon length H. To keep the notation uncluttered, we will also omit the time-window index k whenever the context is clear. We use a linear Gaussian task conditional transition model, i.e, $\text{p}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{l}_k) = \mathcal{N}\left(\mathbf{A}\boldsymbol{z}_{t-1} + \mathbf{B}\boldsymbol{a}_{t-1} + \mathbf{C}\boldsymbol{l}_k, \mathbf{Q}\right)$, where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{Q}$ are state-independent but learnable parameters. In our formulation, the task descriptor can only linearly modify the dynamics which was sufficient to obtain state-of-the-art performance in our experiments, but more complex parametrizations, such as locally linear models, would also be feasible. Following Becker, Pandya, et al. 2019, we split the latent state $\boldsymbol{z}_t = [\boldsymbol{p}_t, \boldsymbol{m}_t]^T$ into its observable part $\boldsymbol{p}_t$ and a part $\boldsymbol{m}_t$ that needs to be observed over time. We also use a linear observation model $p(\boldsymbol{w}_t|\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{H}\boldsymbol{z}_t, \text{diag}(\boldsymbol{\sigma}_t))$ with $\boldsymbol{H} = [\boldsymbol{I}, \boldsymbol{0}]$.

We will assume that the distribution over the task descriptor is also given by a Gaussian distribution, i.e., $p(\boldsymbol{l}_k) = \mathcal{N}(\boldsymbol{\mu}_{l_k}, \boldsymbol{\Sigma}_{l_k})$, which will be provided by the slow-time scale (sts) SSM, see Section 6.2.3. Given these modelling assumptions, the task variable can now be integrated out in closed form, resulting in the following task-conditioned marginal transition model $\text{p}_{l_k}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1}) = \mathcal{N}\left(\mathbf{A}\boldsymbol{z}_{t-1} + \mathbf{B}\boldsymbol{a}_{t-1} + \mathbf{C}\boldsymbol{\mu}_{l_k}, \mathbf{Q} + \boldsymbol{C}\boldsymbol{\Sigma}_{l_k}\boldsymbol{C}^T\right)$, which

will be used instead of the standard dynamics equations. We follow the same factorization assumptions as in Becker, Pandya, et al. 2019 and only estimate the diagonal elements of the block matrices of the covariance matrix of the belief, see Appendix B. The update equations for the Kalman prediction and observation updates are therefore equivalent to the RKN (Becker, Pandya, et al. 2019).

6.2.3. Inference in the Slow-Time Scale SSM

Prediction Update. We follow the same Gaussian inference scheme as for the fts-SSM, i.e., we again employ a linear dynamics model $p(\boldsymbol{l}_k|\boldsymbol{l}_{k-1}, \boldsymbol{\alpha}_k) = \mathcal{N}(\boldsymbol{X}\boldsymbol{l}_{k-1} + \boldsymbol{Y}\boldsymbol{\alpha}_k, \boldsymbol{S})$, where $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{S}$ are learnable parameters. The marginalized transition model for the abstract actions is then given by

$$\mathrm{p}_{\boldsymbol{\alpha}_k}(\boldsymbol{l}_k|\boldsymbol{l}_{k-1}) = \int p(\boldsymbol{l}_k|\boldsymbol{l}_{k-1}, \boldsymbol{\alpha}_k)p(\boldsymbol{\alpha}_k)d\boldsymbol{\alpha}_k = \mathcal{N}\left(\mathbf{X}\boldsymbol{l}_{k-1} + \mathbf{Y}\boldsymbol{\mu}_{\alpha_k}, \mathbf{S} + \boldsymbol{Y}\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_k}\boldsymbol{Y}^T\right).$$

We can directly use this transition model to obtain the Kalman prediction update which computes the prior belief $p_{\boldsymbol{\alpha}_{1:k}}(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k-1}) = \mathcal{N}(\boldsymbol{\mu}_{l_k}^-, \boldsymbol{\Sigma}_{l_k}^-)$ from the posterior belief $p_{\boldsymbol{\alpha}_{1:k-1}}(\boldsymbol{l}_{k-1}|\boldsymbol{\beta}_{1:k-1}) = \mathcal{N}(\boldsymbol{\mu}_{l_{k-1}}^+, \boldsymbol{\Sigma}_{l_{k-1}}^+)$ of the previous time window, see Appendix C.2.1.

Observation/Task Update. Similarly, we will use a linear observation model for the abstract observations $p(\boldsymbol{\beta}_{k,t}|\boldsymbol{l}_k) = \mathcal{N}(\mathbf{H}\boldsymbol{l}_k, \mathrm{diag}(\boldsymbol{\nu}_{k,t}))$ with $\mathbf{H} = [\mathbf{I}, \mathbf{0}]$. As can be seen from the definition of the observation matrix $\mathbf{H}$, the latent space is also decomposed into its observable and unobservable part, i.e., $\boldsymbol{l}_k = [\boldsymbol{u}_k, \boldsymbol{v}_k]$. In difference to the standard factorized Kalman observation update given in Section 2.2.1, we have to infer with a set of observations $\vec{\beta}_{k,t}$ with $t = 1 \dots H$ for a single time window k. While in principle, the Kalman observation update can be applied incrementally H times to obtain the posterior $p_{\boldsymbol{\alpha}_{1:k}}(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k}) = \mathcal{N}(\boldsymbol{\mu}_{l_k}^+, \boldsymbol{\Sigma}_{l_k}^+)$, such an update would be very slow and also cause numerical inaccuracies. Hence, we devise a new permutation invariant version of the update rule that allows parallel processing with set encoders (Zaheer et al. 2017). We found that this update rule is easier to formalize using precision matrices. Hence, we first transform the prior covariance vectors $\boldsymbol{\sigma}_{l_k}^{u-}$, $\boldsymbol{\sigma}_{l_k}^{l-}$ and $\boldsymbol{\sigma}_{l_k}^{s-}$ to its corresponding precision representation $\boldsymbol{\lambda}_{l_k}^{u-}$, $\boldsymbol{\lambda}_{l_k}^{l-}$ and $\boldsymbol{\lambda}_{l_k}^{s-}$ which can be performed using block-wise matrix inversions of $\boldsymbol{\Sigma}_{l_k}^-$. Due to the factorization of the covariance matrix, this operation can be performed solely by scalar inversions.

As the update equations are rather lengthy, they are given in Appendix C.2.1, C.1. Subsequently, we compute the posterior precision, where only $\boldsymbol{\lambda}_{l_k}^u$ is changed by $\boldsymbol{\lambda}_{l_k}^{u+} = \boldsymbol{\lambda}_{l_k}^{u-} + \sum_{t=1}^{H} \mathbf{1} \oslash \boldsymbol{\nu}_{k,t}$ while $\boldsymbol{\lambda}_{l_k}^{l+} = \boldsymbol{\lambda}_{l_k}^{l-}$ and $\boldsymbol{\lambda}_{l_k}^{s+} = \boldsymbol{\lambda}_{l_k}^{s-}$ remain constant. The operator $\oslash$ denotes the element-wise division. From the posterior precision, we can again obtain the

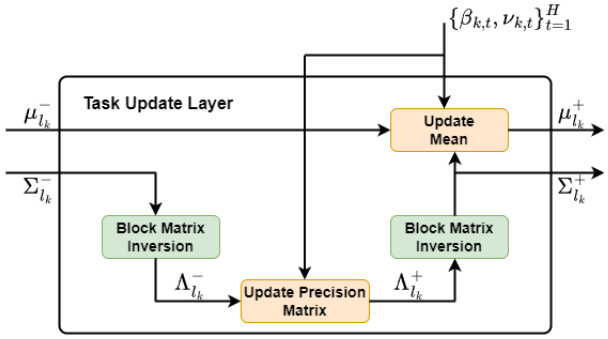


Figure 6.5.: Implementation of task update layer which performs posterior latent task inference in the sts-SSM.

posterior covariance vectors $\boldsymbol{\sigma}_{l_k}^{u+}$, $\boldsymbol{\sigma}_{l_k}^{l+}$ and $\boldsymbol{\sigma}_{l_k}^{s+}$ using only scalar inversions, see Appendix C.2.1, C.1. The posterior mean $\boldsymbol{\mu}_{l,k}^{+}$ can now be obtained from the prior mean $\boldsymbol{\mu}_{l,k}^{-}$ as

$$[c]\boldsymbol{\mu}_{l,k}^{+} = \boldsymbol{\mu}_{l,k}^{-} + \begin{bmatrix} \boldsymbol{\sigma}_{l_k}^{u+} \\ \boldsymbol{\sigma}_{l_k}^{s+} \end{bmatrix} \odot \begin{bmatrix} \sum_{t=1}^{H} \left(\boldsymbol{\beta}_{k,t} - \boldsymbol{\mu}_{l_k}^{\mathrm{u},-}\right) \oslash \boldsymbol{\nu}_{k,t} \\ \sum_{t=1}^{H} \left(\boldsymbol{\beta}_{k,t} - \boldsymbol{\mu}_{l_k}^{\mathrm{u},-}\right) \oslash \boldsymbol{\nu}_{k,t} \end{bmatrix}. \tag{6.1}$$

For $H = 1$, i.e a single observation, the given equation is equivalent to the factorized Kalman observation update (Becker, Pandya, et al. 2019). Furthermore, the given rule constitutes a unification of the batch update rule for Bayesian aggregation (Volpp et al. 2020) and the incremental Kalman update for our factorization of belief state representation (Becker, Pandya, et al. 2019) detailed in Section 2.2.2.2 and Appendix C.2.2.

6.2.4. A General Definition For an N-level MTS3

The human brain's internal representation of the real world is complex, comprising more than just two distinct causal hierarchies. Neuroscience research (Hohwy 2013; K. Friston 2008; Jiang and Rao 2021) suggests that these systems exhibit multiple hierarchical structures. Consequently, we propose a formal definition for Multi-Hierarchy Theory of Mind (MTS3) that accommodates an arbitrary number (N) of hierarchies.

Definition: N-level MTS3

An N-level MTS3 can be defined as a family of N-state space models, $\{S_0, S_1, ..., S_{N-1}\}$. Each of the state space models S_i is given by $S_i = (Z_i, A_i, O_i, f_i, h_i, H_i\Delta t, L_i)$, where Z_i is the state space, A_i the action space, and O_i the observation space of the SSM. The parameter $H_i\Delta t$ denotes the discretization time step and f_i and h_i the dynamics and observation models, respectively. Here, $l_i \in L_i$ is a task descriptor that parameterizes the dynamics model of the SSM and is constant for a local window of steps H_{i+1}. l_i is a function of the latent state of the SSM one level above it, i.e., S_{i+1}. The boundary cases can be defined as follows: for $i = 0$, $H_0 = 1$. Similarly, for $i = N - 1$, the latent task descriptor L_i is an empty set. For all i, $H_i < H_{i+1}$.

6.2.5. Inference In N-Level MTS3

As seen from the definition, an N-level MTS3 is a set of N SSMs that are strictly feudal (Dayan and G. E. Hinton 1992) from top to bottom. Top-level SSM (managers) makes decisions/predictions independently of the bottom-level SSMs, while the bottom-level SSM (worker) is conditioned on its immediate manager. Each of these SSMs performs inference via two stages at every time-step, i.e,

- **Gaussian Conditioning** Updating the posterior belief based on incoming observations.

- **Gaussian Marginalization** Estimating the prior belief for the next step via learned linear dynamics. Note that this stage is conditioned on the current belief of an immediate manager if present.

The efficient Gaussian conditioning and marginalization processes for State-Space Models (SSMs) across all levels in the MTS3 model are specific instances of two broad Gaussian principles derived within the thesis. These principles are formally presented as Gaussian Identity 6.2.1 and Gaussian Identity 6.2.2, as detailed below:

Gaussian Identity 6.2.1 (Gaussian Conditioning)**.** *Consider the graphical model given in Figure 6.6, where a set of N conditionally i.i.d observations* $\boldsymbol{r} = \{\boldsymbol{r}_i\}_{i=1}^{N}$ *are generated by a latent variable* $\boldsymbol{l}$ *and the observation model* $p(\boldsymbol{r}_i|\boldsymbol{l}) = \mathcal{N}\left(\boldsymbol{r}_i \mid \mathbf{H}\boldsymbol{l}, diag(\sigma_i^{obs})\right)$*. Assuming an observation model* $\mathbf{H} = [\mathbf{I}, \mathbf{0}]$*, the mean (*$\boldsymbol{\mu}$*) and precision matrix (*$\boldsymbol{\Lambda}$*) of the posterior over the latent variable* $\boldsymbol{l}$*,* $p(\boldsymbol{l}|\boldsymbol{r}) = \mathcal{N}\left(\boldsymbol{\mu}_l^+, \boldsymbol{\Sigma}_l^+\right) = \mathcal{N}\left(\boldsymbol{\mu}_l^+, (\boldsymbol{\Lambda}_l^+)^{-1}\right)$*, given the prior*

$p_0(\boldsymbol{l}) = \mathcal{N}\left(\boldsymbol{\mu}_l^-, \boldsymbol{\Sigma}_l^-\right) = \mathcal{N}\left(\boldsymbol{\mu}_l^-, (\boldsymbol{\Lambda}_l^-)^{-1}\right)$ *have the following permutation invariant closed form updates.*

$$\boldsymbol{\Lambda}_l^+ = \boldsymbol{\Lambda}_l^- + \begin{bmatrix} diag(\sum_{i=1}^n \frac{1}{\boldsymbol{\sigma}_i^{obs}}), & \mathbf{0} \\ \mathbf{0}, & \mathbf{0} \end{bmatrix} \boldsymbol{\mu}_l^+ = \boldsymbol{\mu}_l^- + \begin{bmatrix} \boldsymbol{\sigma}_l^{u+} \\ \boldsymbol{\sigma}_l^{s+} \end{bmatrix} \odot \begin{bmatrix} \sum_{i=1}^N \left(\boldsymbol{r}_i - \boldsymbol{\mu}_l^{\mathrm{u},-}\right) \odot \frac{1}{\boldsymbol{\sigma}_i^{obs}} \\ \sum_{i=1}^N \left(\boldsymbol{r}_i - \boldsymbol{\mu}_l^{\mathrm{u},-}\right) \odot \frac{1}{\boldsymbol{\sigma}_i^{obs}} \end{bmatrix} \tag{6.2}$$

Proof. The proof for the above identity is given in Appendix C.1.1. □

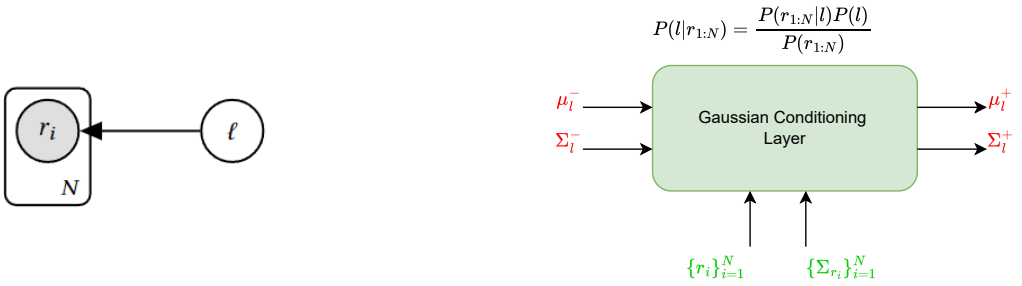


Figure 6.6.: Gaussian Conditioning Process and Implementation. **Left:** A graphical model illustrating the generative process underlying Gaussian conditioning, utilized for Bayesian inversion to infer distributions over model parameters (l) from observed data ($\boldsymbol{o} = \{\boldsymbol{o}_i\}_{i=1}^N$). This model supports various operations including observation updates, task updates, and abstract action updates, each tailored to specific conditions such as different numbers of observations (N) or observation models (H). **Right:** The Gaussian conditioning within neural network architectures across the thesis is implemented as a network layer to perform dynamic parameter updating and end-to-end learning using loss functions of choice.

Corollary 1. *The closed form updates for the resulting posterior distribution $p(l|r)$ is permutation invariant with respect to the observation set r.*

Corollary 1 is also a direct consequence of Theorem 1, which states that the sequential Bayesian update is permutation invariant. The permutation invariance/exchangeability property has both computational and theoretical implications in efficient posterior inference over latent task variables, which is further discussed in the section.

Theorem 6.2.1 (Permutation Invariance Of Bayesian Inversion)**.** *For any conditionally i.i.d model where you have a global parameter θ, and a set of observations $X = \{x_i\}_{i=1}^N$ drawn conditionally i.i.d from a distribution $p(X \mid \theta)$, then for any permutation π, the posterior $p(\theta \mid x_1, \ldots, x_N) = p(\theta \mid \pi(x_1), \ldots, \pi(x_N))$. Thus the posterior $p(\theta \mid X)$ is permutation invariant with respect to the set X.*

Proof. The proof of the above theorem can be found in Appendix C.1.3. □

The derived update equations can be coded as a layer in the neural network architecture as shown in Figure 6.6.

Gaussian Identity 6.2.2 (Linear Combination Gaussian Marginalization)**.** *Consider the graphical model in Figure 6.7, where a set of N normally distributed independent random variables* $u = \{\boldsymbol{u_i} \sim \mathcal{N}(\boldsymbol{\mu_{u_i}}, \boldsymbol{\Sigma_{u_i}})\}_{i=0}^{N}$ *forms a "common effect/V Structure" with a latent variable* $\boldsymbol{y}$*. If the conditional distribution* $p\left(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}, .., \boldsymbol{u_N}\right) = \mathcal{N}(\sum_{i=0}^{N} \mathbf{A_i}\boldsymbol{u_i}, \boldsymbol{\Sigma})$*, then marginal* $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}, .., \boldsymbol{u_N}) \prod_{i=1}^{N} p(\boldsymbol{u_i}) d\boldsymbol{u_i} = \mathcal{N}(\sum_{i=1}^{N} \mathbf{A_i}\boldsymbol{\mu_{u_i}}, \boldsymbol{\Sigma} + \sum_{i=1}^{N} \mathbf{A_i}\boldsymbol{\Sigma}_{u_i}\mathbf{A_i}^T)$.

Proof. The proof for the above identity is given in Appendix C.1.4. □

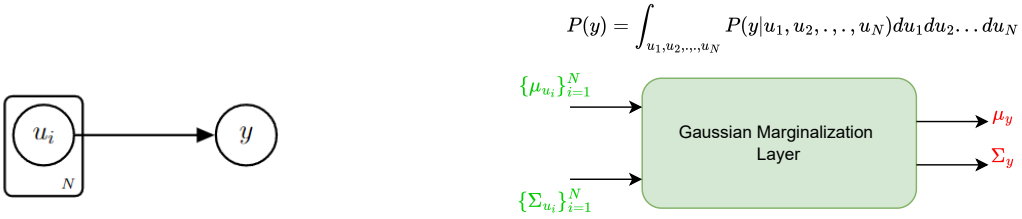


Figure 6.7.: Gaussian Marginalization Process and Implementation: **Left:** A graphical model illustrating the common effect of N normally distributed independent causes $\{\boldsymbol{c_i}\}$ on a latent variable $\boldsymbol{e}$. These causes are marginalized out to obtain $p(\boldsymbol{e})$. The marginalization can be performed in closed form as per Identity 6.2.2. The "predict" step across time scales in every SSM formalism proposed in the thesis is an instance of this operation. **Right:** The Gaussian marginalization within neural network architectures across the thesis is implemented as a neural network layer (with learnable parameters) to perform Gaussian marginalization/model averaging and end-to-end learning.

Even though our experiments focus on MTS3 models with 2 hierarchies, extensive experimentation with more hierarchies can be taken as future work.

6.3. MTS3 as a Hierarchical World Model

MTS3 allows for a natural way to build world models that can deal with partial observability, nonstationarity, and uncertainty in long-term predictions, properties which are critical for model-based control and planning. Furthermore, introducing several levels of latent variables, each working at a different time scale allows us to learn world models that can make action conditional predictions/"dreams" at multiple time scales and multiple levels of state and action abstractions.

6.3.1. Conditional Multi Time Predictions With World Model

Conditional multistep ahead predictions involve estimating plausible future states of the world resulting from a sequence of actions. Our principled formalism allows for action-conditional future predictions at multiple levels of temporal abstraction. The prediction update for the sts-SSM makes prior estimates of future latent task variables conditioned on the abstract action representations. However, the task conditional prediction update in the fts-SSM estimates the future prior latent states, conditioned on primitive actions and the inferred latent task priors, which are decoded to reconstruct future observations. To initialize the prior belief $p(\boldsymbol{z}_{k,1})$ for the first time step of the time window k, we use the prior belief $p(\boldsymbol{z}_{k-1,H+1})$ for the last time step of the time window $k-1$.

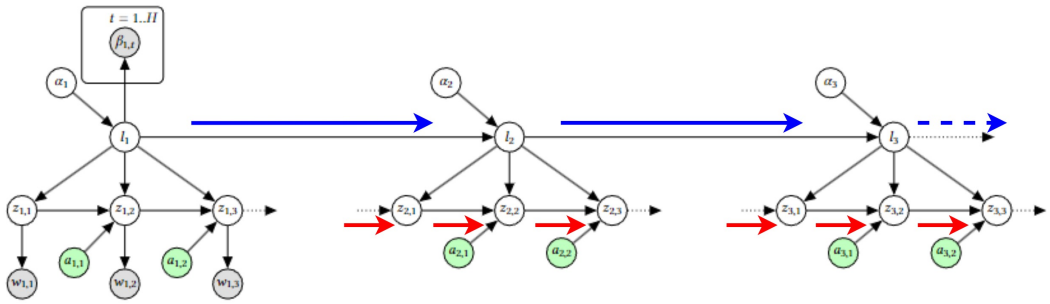


Figure 6.8.: Using hierarchical internal world models to envision the future: Our predictions about future world states rely on **top-down** predictions at multiple time scales and abstractions. Initially, the model generates abstract predictions at the highest level, denoted l_k, using abstract actions (α_k): this process is symbolized by **blue arrows**. Subsequently, each higher-level abstract state adjusts the more detailed, lower-level granular states ($z_{k,t}$), represented by **red arrows**, over a period determined by the time scale parameter H. This mechanism allows the higher level to effectively make "predictions about predictions" concerning the lower level.

6.3.2. Optimizing the Predictive Log-Likelihood

The training objective for the MTS3 involves maximizing the posterior predictive log-likelihood which is given below for a single trajectory, i.e., $\mathrm{L} = \sum_{k=1}^{N}\sum_{t=1}^{H}\log p(\boldsymbol{o}_{k,t+1}|\boldsymbol{\beta}_{1:k-1},\boldsymbol{\alpha}_{1:k},\boldsymbol{w}_{k,1:t},\boldsymbol{a}_{k,1:t})$
$= \sum_{k=1}^{N}\sum_{t=1}^{H}\log p(\boldsymbol{o}_{k,t+1}|\boldsymbol{z}_{k,t+1})p(\boldsymbol{z}_{k,t+1}|\boldsymbol{w}_{k,1:t},\boldsymbol{a}_{k,1:t},\boldsymbol{l}_k)$
$p(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k-1},\boldsymbol{\alpha}_{1:k})d\boldsymbol{z}_{k,t+1}d\boldsymbol{l}_k$
$= \sum_{k=1}^{N}\sum_{t=1}^{H}\log\int p(\boldsymbol{o}_{k,t+1}|\boldsymbol{z}_{k,t+1})p_{l_k}(\boldsymbol{z}_{k,t+1}|\boldsymbol{w}_{k,1:t},\boldsymbol{a}_{k,1:t})d\boldsymbol{z}_{k,t+1}$. The extension to multiple trajectories is straightforward and was omitted to keep the notation uncluttered. Here, $\boldsymbol{o}_{k,t+1}$ is the ground-truth observations at time step $t+1$ and time window k that needs to be predicted from all (latent and abstract) observations up to time step t. The

prior belief corresponding to the latent state $p_{l_k}(\boldsymbol{z}_{k,t+1}|\boldsymbol{w}_{k,1:t}, \boldsymbol{a}_{k,1:t})$ has a closed form solution as discussed in Section 6.2.2.

We employ a Gaussian approximation of the posterior predictive log-likelihood of the form $p(\boldsymbol{o}_{k,t+1}|\boldsymbol{\beta}_{1:k-1}, \boldsymbol{\alpha}_{1:k}, \boldsymbol{w}_{k,1:t}, \boldsymbol{a}_{k,1:t}) \approx \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{o}_{k,t+1}}, \text{diag}(\boldsymbol{\sigma}_{\boldsymbol{o}_{k,t+1}}))$ where we use the mean of the prior belief $\boldsymbol{\mu}^-_{z_{k,t+1}}$ to decode the predictive mean, i.e, $\boldsymbol{\mu}_{\boldsymbol{o}_{k,t+1}} = \text{dec}_{\boldsymbol{\mu}}(\boldsymbol{\mu}^-_{z_{k,t+1}})$ and the variance estimate of the prior belief to decode the observation variance, i.e., $\boldsymbol{\sigma}_{o_{k,t+1}} = \text{dec}_{\sigma}(\boldsymbol{\Sigma}^-_{z_{k,t+1}})$. This approximation can be motivated by a moment matching perspective and allows for end-to-end optimization of the logarithmic likelihood without using auxiliary objectives such as the ELBO Becker, Pandya, et al. 2019.

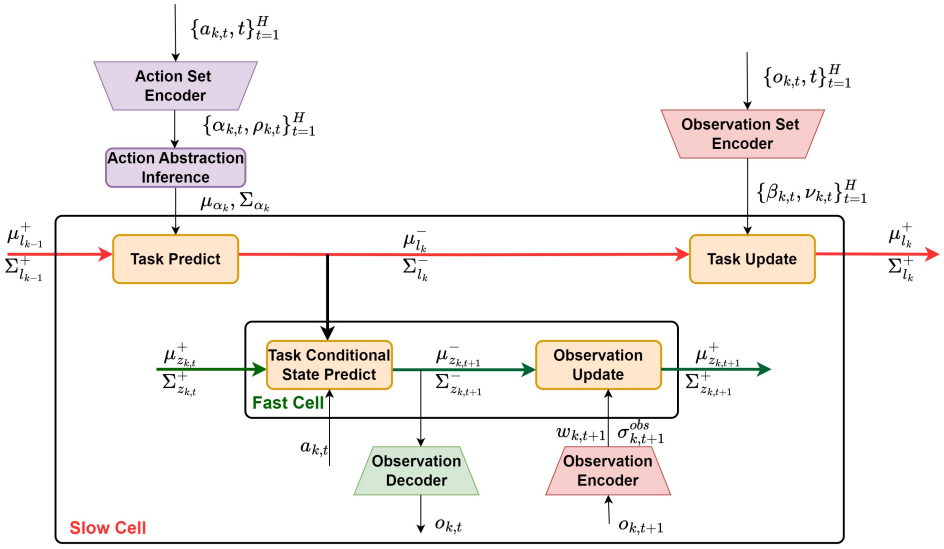


Figure 6.9.: Schematic of a 2-Level MTS3 Architecture. Inference in MTS3 takes place via closed-form equations derived using exact inference, spread across two-time scales. For the fast time scale (fts) SSM, these include the task conditional state predict and observation update stages as discussed in Section C.2.2 of the main paper. Whereas, for the slow time scale (sts) SSM, these include the task prediction and task update stages which are described in Section C.2.1. More implementation details can also be found in Appendix C.2.

Gradients are computed using (truncated) backpropagation over time (BPTT) (Werbos 1990) and clipped. We optimize the objective using the Adam (Kingma and Ba 2014) stochastic gradient descent optimizer with default parameters. A schematic of MTS3 architecture is shown in Figure 6.9. We refer to Appendix C for more details. For training, we also initialize the prior belief $p(\boldsymbol{z}_{k,1})$ with the prior belief $p_{l_{k-1}}(\boldsymbol{z}_{k-1,H+1}|\boldsymbol{w}_{k-1,1:H}, \boldsymbol{a}_{k-1,1:H})$ from the previous time window $k-1$. However, we cut the gradients for the fast time scale between time windows as this avoids vanishing gradients, and we observed a more stable learning behavior. Yet, the gradients can still flow between time windows for the fts-SSM via the sts-SSM.

6.3.3. Imputation Based Self Supervised Training For Long Term Prediction

Using the given training loss results in models that are good in one-time step prediction, but typically perform poorly in long-term predictions as the loss assumes that observations are always available up to time step t. To increase the performance of the long-term prediction, we can treat the long-term prediction problem as a case of the problem of "missing value", where the missing observations occur in future time steps. Thus, to train our model for long-term prediction, we randomly mask a fraction of observations and explicitly task the network to impute the missing observations, resulting in a strong self-supervised learning signal for long-term prediction with varying prediction horizon length. This imputation scheme is applied at both time scales, masking out single time steps or whole time windows of length H as shown in Figure 6.10. The imputation mask is also randomly resampled for every minibatch.

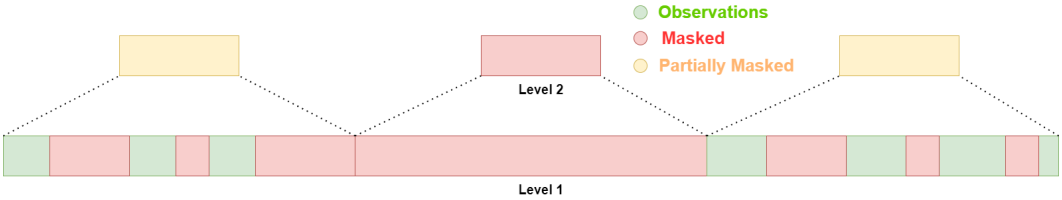


Figure 6.10.: Self-supervised training for multi-step ahead predictions by tasking the model to "fill in" the masked observations. Here masking is done at both time scales based on the available observation/set of observations in each time window.

6.4. Experiments

In this section, we evaluate our approach to a diverse set of simulated and real-world dynamical systems for long-horizon prediction tasks. Our experiments are designed to answer the following questions. (a) Can MTS3 make accurate long-term deterministic predictions (mean estimates)? (b) Can MTS3 make accurate long-term probabilistic predictions (variance estimates)? (c) How important are the modelling assumptions and training scheme?

6.4.1. Baseline Dynamics Models

While a full description of our baselines can be found in Appendix D.3, a brief description of them is given here: (a) **RNNs** - We compare our method to two widely used recurrent neural network architectures, LSTMs (Hochreiter and Schmidhuber 1997) and GRUs (Cho et al. 2014). (b) **RSSMs** - Among several RSSMs from the literature, we chose RKN (Becker, Pandya, et al. 2019) and HiP-RSSM (Shaj, Buchler, et al. 2022) as these have shown excellent performance for dynamics learning for short-term predictions

and rely on exact inference as in our case. (c) **Transformers** - We also compare with two state-of-the-art Transformer (Vaswani et al. 2017) variants. The first variant (AR-Transformer) relies on a GPT-like autoregressive prediction (Radford et al. 2019; Brown et al. 2020). Whereas the second variant (Multi-Transformer) uses direct multi-step loss (Zeng et al. 2022) from recent literature on long horizon time-series forecasting (Zhou et al. 2021; Liu et al. 2022; Nie et al. 2023). Here, multistep ahead predictions are performed using a single shot given the action sequences.

6.4.2. Environments and Datasets

We experiment with three broad datasets. In all datasets, we only use information about agent/object positions and we mask out velocities to create a partially observable setting. While full descriptions of these datasets, dataset creation procedure, and overall statistics are given in Appendix D.1, a brief description of them is as follows.

D4RL Datasets - We use a set of 3 different environments/agents from D4RL dataset (Fu, Kumar, et al. 2020), which includes the HalfCheetah, Medium Maze and Franka Kitchen environment. Each of these was chosen because of their distinct properties like sub-optimal trajectories (HalfCheetah), realistic domains / human demonstrations (Kitchen), multi-task trajectories, non-markovian collection policies (Kitchen and Maze) and availability of long horizon episodes (all three).

Manipulation Datasets - We use 2 datasets collected from a real excavator arm and a Panda robot. The highly non-linear non-markovian dynamics due to hydraulic actuators

Figure 6.11.: Figures of a subset of the agents used for collecting datasets. (top) D4RL Environments: HalfCheetah, Franka Kitchen, Maze2D-Medium (bottom) JCB Hydradig 110W Excavator

in the former and non-stationary dynamics owing to different payloads in the latter make them challenging benchmarks. Furthermore, accurate modelling of the dynamics of these complex systems is important since learning control policies for automation directly on large excavators is economically infeasible and potentially hazardous.

Mobile Robotics Dataset - We set up a simulated four-wheeled mobile robot traversing a highly uneven terrain of varying steepness generated by a mix of sinusoidal functions. This problem is challenging due to the highly non-linear dynamics involving wheel-terrain interactions and non-stationary dynamics introduced by varying steepness levels.

6.4.3. Can MTS3 make accurate long-term deterministic predictions (mean estimates)?

Here we evaluate the quality of the mean estimates for long-term prediction using our approach. The results are reported in terms of "sliding window RMSE" in Figure 6.12. We see that MTS3 gives consistently good long-term action conditional future predictions on all 6 datasets. Deep Kalman models (Becker, Pandya, et al. 2019; Shaj, Buchler, et al. 2022) which operate on a single time scale fail to give meaningful mean estimates beyond a few milliseconds. Similarly, widely used RNN baselines (Hochreiter and Schmidhuber 1997; Cho et al. 2014) which form the backbone of several world models (Ha and Schmidhuber 2018; Hafner et al. 2019) give poor action conditional predictions over long horizons. AR-Transformers also fail possibly due to error accumulation caused by the autoregression. However, Multi-Transformers are a strong baseline that outperforms MTS3 in the Medium Maze and Panda dataset by a small margin. However, on more complex tasks like the Kitchen task, which requires modelling multi-object, multi-task interactions (A. Gupta et al. 2019), MTS3 is the only model that gives meaningful long horizon predictions. A detailed description of the metric "sliding window RMSE" is given in Appendix C.3. A visualization of the predicted trajectories vs. ground truth is given in Appendix C.4.

6.4.4. Can MTS3 make accurate long-term probabilistic predictions (variance estimates)?

Next, we examine the question of whether the principled probabilistic inference translates to accurate uncertainty quantification during long-horizon predictions. We trained all the baselines with a negative log-likelihood loss and used the same as a metric to quantify the quality of uncertainty estimates. During the evaluation we rely on a sliding window approach (see Appendix C.3) and report the results for the last timestep in Table 6.1. As seen in Table 6.1, MTS3 gives the most accurate uncertainty estimates in all datasets except Medium Maze, where it is outperformed by Multi-Transformer. Also, notably, AR-Transformers and deep Kalman models fail to learn any meaningful uncertainty representation when it comes to long-term predictions.

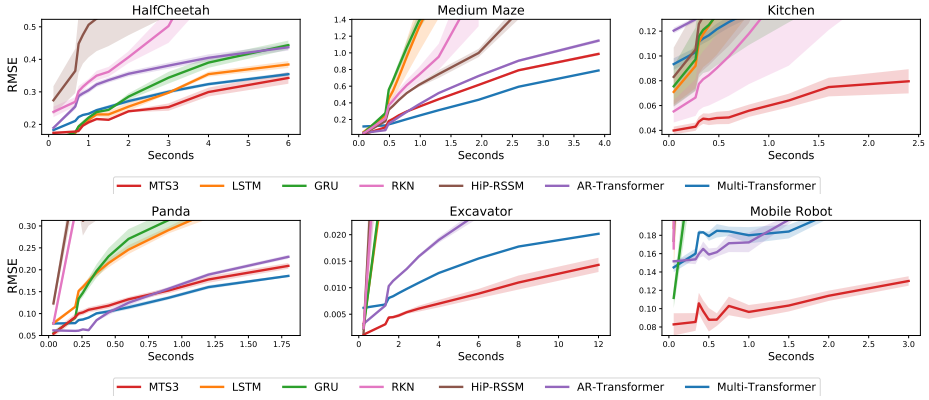


Figure 6.12.: Comparison with baselines in terms of RMSE for long horizon predictions (in seconds) as discussed in Section 6.4.3.

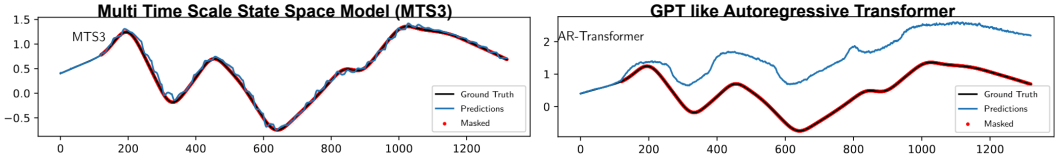


Figure 6.13.: Visualizations of the predicted trajectories vs ground truth for MTS3 and GPT like autoregressive transformer (AR-Transformer). More visualizations across algorithms and datasets can be found in Appendix C.4.

	Prediction Horizon	Algorithm						
		MTS3	**Multi-Trans**	**AR-Trans**	**LSTM**	**GRU**	**RKN**	**HiP-RSSM**
Half Cheetah	6 s	$\mathbf{-2.80 \pm 0.30}$	0.25 ± 0.05	✗	7.34 ± 0.06	7.49 ± 0.04	✗	✗
Kitchen	2.5 s	$\mathbf{-25.74 \pm 0.12}$	-7.3 ± 0.2	✗	32.45 ± 1.64	32.72 ± 0.65	✗	✗
Medium Maze	4 s	-0.21 ± 0.022	$\mathbf{-0.88 \pm 0.02}$	✗	4.03 ± 0.32	7.76 ± 0.07	✗	✗
Panda	1.8 s	$\mathbf{2.79 \pm 0.32}$	3.77 ± 0.33	✗	7.94 ± 0.39	7.91 ± 0.23	✗	✗
Hydraulic	12 s	$\mathbf{-2.64 \pm 0.12}$	-2.46 ± 0.03	✗	7.35 ± 0.061	7.35 ± 0.06	✗	✗
Mobile Robot	3 s	$\mathbf{-6.47 \pm 0.71}$	-5.17 ± 0.23	✗	11.27 ± 2.3	14.55 ± 5.6	✗	✗

Table 6.1.: Comparison in terms of Negative Log Likelihood (NLL) for long horizon predictions (in seconds). Here bold numbers indicate the top methods and ✗ denotes very high/nan values resulting from the highly divergent mean/variance long-term predictions.

6.4.5. How important are the modelling assumptions and training scheme?

Now, we look at three important modelling and training design choices: (i) splitting the latent states to include an unobservable "memory" part using observation model $h^{sts} = h^{fts} = \mathbf{H} = [\mathbf{I}, \mathbf{0}]$ as discussed in Sections 6.2.3 and 6.2.2, (ii) action abstractions in Section 6.2.1.2, (iii) training by imputation.

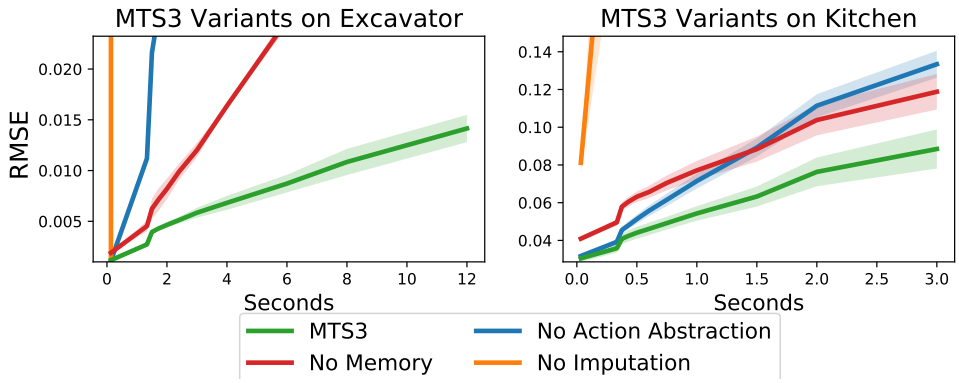


Figure 6.14.: Ablation on discretization step $H.\Delta t$ (a) The long-term prediction results in terms of RMSE, with different H values as discussed in Section 6.4.6 on the hydraulics dataset.

To analyze the importance of the memory component, we derived and implemented an MTS3 variant with an observation model of $h^{sts} = h^{fts} = \mathbf{I}$ and a pure diagonal matrix representation for the covariance matrices. As seen in Figure 6.14, this results in worse long-term predictions, suggesting that splitting the latent states in its observable and unobservable part in MTS3 is critical for learning models of non-markovian dynamical systems. Regarding (ii), we further devised another variant where MTS3 only had access to observations, primitive actions and observation abstractions, but no action abstractions. As seen in our ablation studies, using the action abstraction is crucial for long-horizon predictions.

Our final ablation (iii) shows the importance of an imputation-based training scheme discussed in Section 6.3.3. As seen in Figure 6.14 when trained for 1 step ahead predictions without imputation, MTS3 performs significantly worse for long-term prediction suggesting the importance of this training regime.

6.4.6. What is the role of the discretization step $H.\Delta t$?

Finally, we perform ablation for different values of $H.\Delta t$, which controls the time scale of the task dynamics. The higher the value of H, the slower the timescale of the task dynamics relative to the state dynamics. As seen in Figure 6.16 (left), for the hydraulics data, smaller values of H (2,3,5 and 10) give significantly worse performance. Very large values of H (like 75) also result in degradation of performance. To get an intuitive understanding, we plot the predictions given by MTS3 for different values of H on a trajectory handpicked from the hydraulics excavator dataset. As seen in Figure 6.15 for large values of H like 30 and 75, we notice that the upper level "reconfigures" the lower level every 30 and 75-step window respectively, by conditioning the lower level dynamics with the newly updated task prior. This effect is noticeable as periodic jumps or discontinuities in the predictions, occurring at 30 and 75-step intervals. Also, for a very

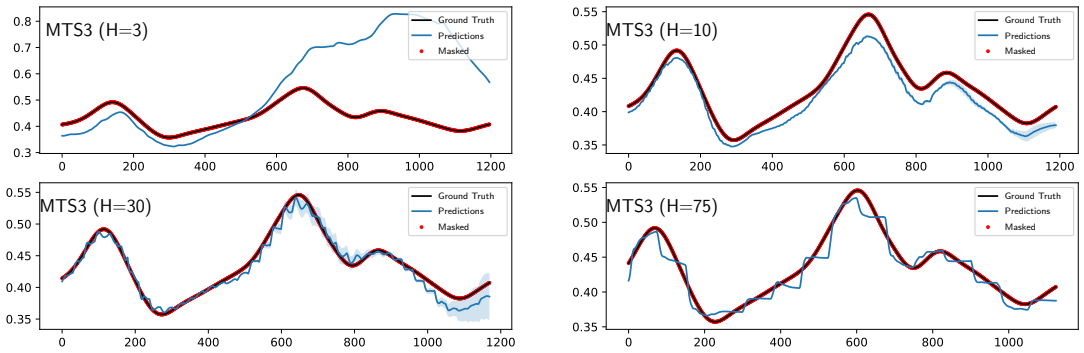


Figure 6.15.: Ablation on discretization step $H.\Delta t$. The predictions by MTS3 variants with different values of timescale parameter $H.\Delta t$ on a trajectory picked from the hydraulics excavator dataset. The top images are for $H = 3$ and $H = 10$. The bottom images are for $H = 30$ and $H = 75$. Note that the results reported in the paper are with $H = 30$.

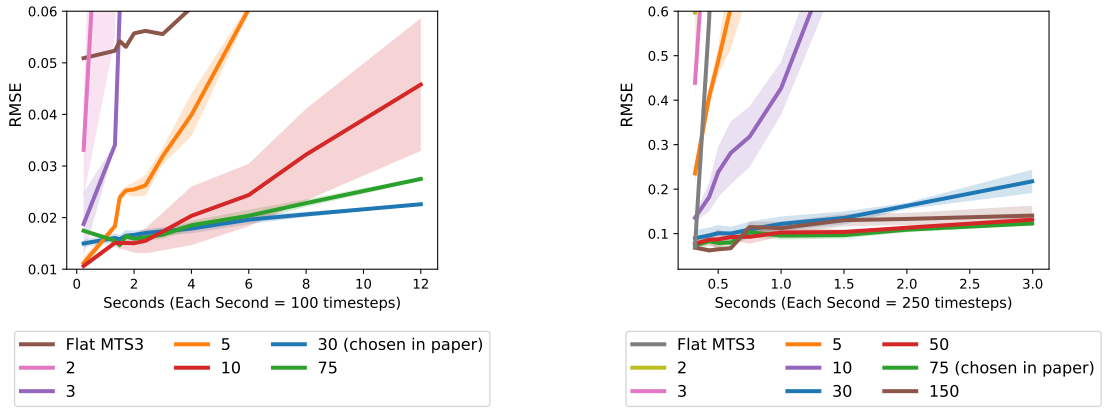


Figure 6.16.: Ablation on discretization step $H.\Delta t$. The long-term prediction results in terms of RMSE, with different H values as discussed in Section 6.4.6 on (left) the hydraulics dataset and (right) the mobile robot dataset.

large H like 75, the fast time scale SSM has to make many more steps in a longer window resulting in error accumulation and poor predictions. A similar trend was observed for the mobile robot data as seen in Figure 6.16 (right). For the mobile robot, smaller values of H (like 2,3,5 and 10) and very large values of H (like 150) gave sub-optimal performance. In the paper, we used a value of H=75.

6.5. Conclusion and Future Work

In this work, we introduce MTS3, a probabilistic formalism for learning the dynamics of complex environments at multiple time scales. By modelling the dynamics of the world at multiple levels of temporal abstraction we capture both the slow-changing long-term

trends and fast-changing short-term trends in data, leading to highly accurate predictions spanning several seconds into the future. Our experiments demonstrate that simple linear models with principled modelling assumptions can compete with large transformer model variants that require several times more parameters. Furthermore, our inference scheme also allows for principled uncertainty propagation over long horizons across multiple time scales which capture the stochastic nature of environments. We believe our formalism can benefit multiple future applications including hierarchical planning/control. We discuss the limitations and broader impacts of our work in Chapter 7.

7. Outlook

This thesis questioned the current formalisms for learning-world models due to their inability to capture some critical criteria for a foundational world model. Specifically, we looked at three aspects (i) scalable probabilistic modelling that captures causal relations in our world (Chapter 4, 5 and 6), (ii) adaptability to changing tasks (Chapter 5) and (iii) hierarchical modelling at multiple time scales and temporal abstractions (Chapter 6). In this context the thesis proposed two new formalisms Hidden Parameter State Space Models (HiP-SSM in Chapter 5) and Multi Time Scale State Space Model (MTS3 in Chapter 6) besides making an existing formalism of SSMs more robust for performing interventions/counterfactuals with action/control signals (Ac-SSM in Chapter 4).

The proposed formalisms are in line with related theory in computational neuroscience called predictive processing and Bayesian brain hypothesis (Chapter 3). In the proposed generative models, Bayesian inversion is employed to deduce the underlying causes from the incoming sensory data at various levels of abstraction. The Bayesian inversion results in update rules that use "precision weighting" (K. Friston 2009; Hohwy 2013; A. K. Seth 2014) as a mechanism to direct "attention" to relevant sensory information. The idea of "precision estimation" by learned sensory encoders is also in line with discussions in the broader neuroscience community. The world model formalisms proposed are adaptable to changing situations (embodying cognitive flexibility) and can make top-down long-horizon predictions based on nested causal hierarchies.

Machines that can replicate human intelligence and type 2 reasoning capabilities should be able to reason at multiple levels of temporal abstraction using internal world models (LeCun 2022; T. Gupta et al. 2024). The work represents a hopeful step towards developing embodied AI systems with a deeper and more intuitive grasp of the world, aligning machine learning processes more closely with natural intelligence. We believe that the findings of this dissertation will contribute to ongoing research in the creation of robust, principled, and scalable world models, while acknowledging the vast scope for further research and development in this area.

Limitations and Future Work The thesis specifically concentrated on the aspect of prediction rather than control or planning. Addressing **hierarchical planning**, much like hierarchical prediction, remains a significantly underexplored challenge. A logical progression for future research would be to extend the concept of planning within the exact inference framework (Botvinick and Toussaint 2012; Watson, Abdulsamad, and Peters 2020). This approach necessitates the derivation of backward messages and the use of the

Kalman duality principle within the learned latent spaces of a deep encoder. Furthermore, investigating variational approaches to the hierarchical models presented, and considering hierarchical planning/control as a problem of approximate inference (Toussaint et al. 2009; K. Friston 2009; Millidge et al. 2020), presents another intriguing avenue for research.

A further limitation noted in the thesis is its reliance on proprioceptive sensors for experimentation, without empirical validation using high-dimensional sensory information, such as vision. An important avenue for future research would involve evaluating the usefulness and characteristics of abstractions derived through Bayesian inversion/aggregation when applied to multimodal high-dimensional and **sensory inputs**. Conducting experiments with image-based data and exploring "non-reconstruction"-based loss functions, as suggested by LeCun 2022, represent promising directions for future work.

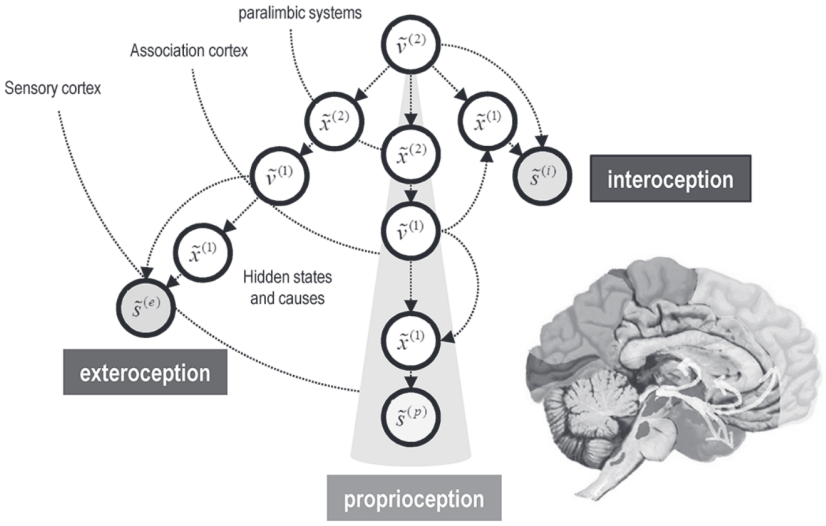


Figure 7.1.: Perception and consciousness as hierarchical inference from K. Friston 2013.

Though our hierarchical formalism outperform transformer variants on long-horizon predictions on several tasks, a limitation for these to be used as foundational world models is the computational bottleneck due to the sequential nature of dynamics. However, there have been exciting progress recently in deterministic linear SSMs (Gu, Goel, and Re 2021; Smith, Warrington, and Linderman 2022; Mondal et al. 2023) that allow efficient **parallelization** during training. Since our dynamics is linear similar to these approaches, parallelization using similar techniques as employed by deterministic SSMs can be another important direction of future research.
Our experimentation was limited to the hierarchical models with just two levels of abstraction, as this configuration proved to be adequate for numerous tasks we carried out. However, there is room for deeper exploration.The intersection of machine learning with broader cognitive theories, particularly the concept of **machine consciousness**,

presents a novel yet underexplored frontier in the field. Traditionally, consciousness has been a topic more familiar to the realms of philosophy and psychology than to machine learning and computer science. This often makes it a delicate subject within our discipline.

As I write this thesis in the Spring of 2024, the machine learning field finds itself at a crossroads, significantly influenced by the impressive capabilities of attention/transformer (Vaswani et al. 2017) based large language models (LLMs). These advancements suggests a pressing need for mathematical models or analytical frameworks to better understand machine consciousness (Bengio 2017; Chalmers 2023). Numerous theories on consciousness highlight the critical role of recurrent processing (Chalmers 2023; Lamme 2010), a feature notably absent in current LLM architectures. In this context, I believe that hierarchical generative models, particularly those extending beyond the 2-level temporal depth as explored in this thesis, represent a promising research direction for embedding consciousness priors (K. Friston 2013; Bengio 2017) into machine learning systems. This approach could offer profound insights into the mechanisms underpinning consciousness and its potential replication in artificial systems.

Bibliography

Achterhold, Jan and Joerg Stueckler (2021). "Explore the Context: Optimal Data Collection for Context-Conditional Dynamics Models". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 3529–3537.

authors, The GPyOpt (2016). *GPyOpt: A Bayesian Optimization framework in python*. `http://github.com/SheffieldML/GPyOpt`.

Ballard, Dana H, Geoffrey E Hinton, and Terrence J Sejnowski (1983). "Parallel visual computation". In: *Nature* 306.5938, pp. 21–26.

Baranes, Adrien and Pierre-Yves Oudeyer (2013). "Active learning of inverse models with intrinsically motivated goal exploration in robots". In: *Robotics and Autonomous Systems* 61.1, pp. 49–73.

Becker, Philipp and Gerhard Neumann (2022). "On Uncertainty in Deep State Space Models for Model-Based Reinforcement Learning". In: *arXiv preprint arXiv:2210.09256*.

Becker, Philipp, Harit Pandya, et al. (2019). "Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces". In: *International Conference on Machine Learning*. PMLR, pp. 544–552.

Becker-Ehmck, Philip, Jan Peters, and Patrick Van Der Smagt (2019). "Switching linear dynamics for variational bayes filtering". In: *International Conference on Machine Learning*. PMLR, pp. 553–562.

Bengio, Yoshua (2017). "The consciousness prior". In: *arXiv preprint arXiv:1709.08568*.

Bishop, Christopher M (2006). "Pattern recognition". In: *Machine learning* 128.9.

Botvinick, Matthew and Marc Toussaint (2012). "Planning as inference". In: *Trends in cognitive sciences* 16.10, pp. 485–488.

Brown, Tom et al. (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Büchler, Dieter, Simon Guist, et al. (2020). "Learning to Play Table Tennis From Scratch using Muscular Robots". In: *arXiv preprint arXiv:2006.05935*.

Büchler, Dieter, Heiko Ott, and Jan Peters (2016). "A lightweight robotic arm with pneumatic muscles for robot learning". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4086–4092.

Calinon, Sylvain et al. (2010). "Learning and reproduction of gestures by imitation". In: *IEEE Robotics & Automation Magazine* 17.2, pp. 44–54.

Chalmers, David J (2023). "Could a large language model be conscious?" In: *arXiv preprint arXiv:2303.07103*.

Cho, Kyunghyun et al. (Oct. 2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, pp. 103–111.

Chua, Kurtland et al. (2018). "Deep reinforcement learning in a handful of trials using probabilistic dynamics models". In: *Advances in neural information processing systems* 31.

Clark, Andy (2013). "Whatever next? Predictive brains, situated agents, and the future of cognitive science". In: *Behavioral and brain sciences* 36.3, pp. 181–204.

Coumans, Erwin and Yunfei Bai (2016). "Pybullet, a python module for physics simulation for games, robotics and machine learning". In.

Daniel, Kahneman (2017). *Thinking, fast and slow*.

Dayan, Peter and Geoffrey E Hinton (1992). "Feudal reinforcement learning". In: *Advances in neural information processing systems* 5.

Dayan, Peter, Geoffrey E Hinton, et al. (1995). "The helmholtz machine". In: *Neural computation* 7.5, pp. 889–904.

Deisenroth, Marc and Carl E Rasmussen (2011). "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472.

Deisenroth, Marc Peter, Dieter Fox, and Carl Edward Rasmussen (2013). "Gaussian processes for data-efficient learning in robotics and control". In: *IEEE transactions on pattern analysis and machine intelligence* 37.2, pp. 408–423.

Depeweg, Stefan et al. (2016). "Learning and policy search in stochastic dynamical systems with bayesian neural networks". In: *arXiv preprint arXiv:1605.07127*.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Dong, Zhe et al. (2020). "Collapsed amortized variational inference for switching nonlinear dynamical systems". In: *International Conference on Machine Learning*. PMLR, pp. 2638–2647.

Doshi-Velez, Finale and George Konidaris (2016). "Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations". In: *IJCAI: proceedings of the conference*. Vol. 2016. NIH Public Access, p. 1432.

Ferreira, Joao P et al. (2007). "Simulation control of a biped robot with support vector regression". In: *2007 IEEE International Symposium on Intelligent Signal Processing*. IEEE, pp. 1–6.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning*. PMLR, pp. 1126–1135.

Finn, Chelsea, Ian Goodfellow, and Sergey Levine (2016). "Unsupervised learning for physical interaction through video prediction". In: *Advances in neural information processing systems* 29, pp. 64–72.

Fraccaro, Marco et al. (2017). "A disentangled recognition and nonlinear dynamics model for unsupervised learning". In: *arXiv preprint arXiv:1710.05741*.

Friston, Karl (2005). “A theory of cortical responses”. In: *Philosophical transactions of the Royal Society B: Biological sciences* 360.1456, pp. 815–836.
– (2008). “Hierarchical models in the brain”. In: *PLoS computational biology* 4.11, e1000211.
– (2009). “The free-energy principle: a rough guide to the brain?” In: *Trends in cognitive sciences* 13.7, pp. 293–301.
– (2013). “Consciousness and hierarchical inference”. In: *Neuropsychoanalysis* 15.1, pp. 38–42.
Fu, Justin, Aviral Kumar, et al. (2020). “D4rl: Datasets for deep data-driven reinforcement learning”. In: *arXiv preprint arXiv:2004.07219*.
Fu, Justin, Sergey Levine, and Pieter Abbeel (2016). “One-shot learning of manipulation skills with online dynamics adaptation and neural network priors”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4019–4026.
Gal, Yarin, Rowan McAllister, and Carl Edward Rasmussen (2016). “Improving PILCO with Bayesian neural network dynamics models”. In: *Data-efficient machine learning workshop, ICML*. Vol. 4. 34, p. 25.
Garnelo, Marta et al. (2018). “Neural processes”. In: *arXiv preprint arXiv:1807.01622*.
Gordon, Jonathan et al. (2018). “Meta-learning probabilistic inference for prediction”. In: *arXiv preprint arXiv:1805.09921*.
Grancharova, Alexandra, Juš Kocijan, and Tor A Johansen (2008). “Explicit stochastic predictive control of combustion plants based on Gaussian process models”. In: *Automatica* 44.6, pp. 1621–1631.
Gregory, Richard Langton (1968). “Perceptual illusions and brain models”. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 171.1024, pp. 279–296.
– (1980). “Perceptions as hypotheses”. In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 290.1038, pp. 181–197.
Gu, Albert, Karan Goel, and Christopher Re (2021). “Efficiently Modeling Long Sequences with Structured State Spaces”. In: *International Conference on Learning Representations*.
Gupta, Abhishek et al. (2019). “Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning”. In: *arXiv preprint arXiv:1910.11956*.
Gupta, Tarun et al. (2024). “The Essential Role of Causality in Foundation World Models for Embodied AI”. In: *arXiv preprint arXiv:2402.06665*.
Ha, David and Jürgen Schmidhuber (2018). “World models”. In: *arXiv preprint arXiv:1803.10122*.
Haarnoja, Tuomas et al. (2016). “Backprop kf: Learning discriminative deterministic state estimators”. In: *Advances in neural information processing systems*, pp. 4376–4384.
Hafner, Danijar et al. (2019). “Learning latent dynamics for planning from pixels”. In: *International Conference on Machine Learning*. PMLR, pp. 2555–2565.
Haruno, Masahiko, Daniel M Wolpert, and Mitsuo Kawato (2001). “Mosaic model for sensorimotor learning and control”. In: *Neural computation* 13.10, pp. 2201–2220.
Helmholtz, Hermann von (1948). “Concerning the perceptions in general, 1867.” In.

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). "Long Short-Term Memory". In: *Neural Comput.* 9.8, pp. 1735–1780. ISSN: 0899-7667.

Hohwy, Jakob (2013). *The predictive mind*. OUP Oxford.

Isomura, Takuya, Thomas Parr, and Karl Friston (Dec. 2019). "Bayesian Filtering with Multiple Internal Models: Toward a Theory of Social Intelligence". In: *Neural computation* 31 (12), pp. 2390–2431. ISSN: 1530-888X. DOI: 10.1162/NECO_A_01239. URL: https://pubmed.ncbi.nlm.nih.gov/31614100/.

Jiang, Linxing Preston and Rajesh PN Rao (2021). "Predictive coding theories of cortical function". In: *arXiv preprint arXiv:2112.10048*.

Jordan, Michael I (2004). "Graphical models". In: *Statistical science* 19.1, pp. 140–155.

Kalman, Rudolph E and Richard S Bucy (1961). "New results in linear filtering and prediction theory". In.

Kalman, Rudolph Emil (1960). "A new approach to linear filtering and prediction problems". In.

Kamthe, Sanket and Marc Deisenroth (2018). "Data-efficient reinforcement learning with probabilistic model predictive control". In: *International conference on artificial intelligence and statistics*. PMLR, pp. 1701–1710.

Karl, Maximilian et al. (2016). "Deep variational bayes filters: Unsupervised learning of state space models from raw data". In: *arXiv preprint arXiv:1605.06432*.

Kawato, Mitsuo, Hideki Hayakawa, and Toshio Inui (1993). "A forward-inverse optics model of reciprocal connections between visual cortical areas". In: *Network: computation in neural systems* 4.4, p. 415.

Kersten, Daniel, Pascal Mamassian, and Alan Yuille (2004). "Object perception as Bayesian inference". In: *Annu. Rev. Psychol.* 55, pp. 271–304.

Khalil, W., M. Gautier, and Ch. Enguehard (1991). "Identifiable Parameters and Optimum Configurations for Robots Calibration". In: *Robotica* 9.1, pp. 63–70. DOI: 10.1017/S0263574700015575.

Khansari-Zadeh, S Mohammad and Aude Billard (2011). "Learning stable nonlinear dynamical systems with gaussian mixture models". In: *IEEE Transactions on Robotics* 27.5, pp. 943–957.

Killian, Taylor, George Konidaris, and Finale Doshi-Velez (2016). "Transfer learning across patient variations with hidden parameter markov decision processes". In: *arXiv preprint arXiv:1612.00475*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Knill, David C and Alexandre Pouget (2004). "The Bayesian brain: the role of uncertainty in neural coding and computation". In: *TRENDS in Neurosciences* 27.12, pp. 712–719.

Ko, Jonathan et al. (2007). "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp". In: *Proceedings 2007 ieee international conference on robotics and automation*. IEEE, pp. 742–747.

Kocijan, Jus et al. (2004). "Gaussian process model based predictive control". In: *Proceedings of the 2004 American control conference*. Vol. 3. IEEE, pp. 2214–2219.

Kocoń, Jan et al. (2023). "ChatGPT: Jack of all trades, master of none". In: *Information Fusion* 99, p. 101861.

Koller, Daphne and Nir Friedman (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Koller, Daphne, Nir Friedman, et al. (2007). "Graphical models in a nutshell". In: *Introduction to statistical relational learning* 43.

Koopman, Bernard O (1931). "Hamiltonian systems and transformation in Hilbert space". In: *Proceedings of the National Academy of Sciences* 17.5, pp. 315–318.

Krishnan, Rahul, Uri Shalit, and David Sontag (2017). "Structured inference networks for nonlinear state space models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1.

Lamme, Victor AF (2010). "How neuroscience will change our view on consciousness". In: *Cognitive neuroscience* 1.3, pp. 204–220.

LeCun, Yann (2022). "A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27". In: *Open Review* 62.

Lee, Tai Sing and David Mumford (2003). "Hierarchical Bayesian inference in the visual cortex". In: *JOSA a* 20.7, pp. 1434–1448.

Lenz, Ian, Ross A Knepper, and Ashutosh Saxena (2015). "DeepMPC: Learning deep latent features for model predictive control." In: *Robotics: Science and Systems*. Vol. 10. Rome, Italy, p. 25.

Linderman, Scott et al. (2017). "Bayesian learning and inference in recurrent switching linear dynamical systems". In: *Artificial Intelligence and Statistics*. PMLR, pp. 914–922.

Liu, Yong et al. (2022). "Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. URL: https://openreview.net/forum?id=ucNDIDRNjjv.

MacKay, Donald M (1956). "The epistemological problem for automata". In: *Automata studies*, pp. 235–51.

McInnes, Leland, John Healy, and James Melville (2018). "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426*.

Micheli, Vincent, Eloi Alonso, and François Fleuret (2023). "Transformers are Sample-Efficient World Models". In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=vhFu1Acb0xb.

Millidge, Beren et al. (2020). "On the relationship between active inference and control as inference". In: *Active Inference: First International Workshop, IWAI 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14, 2020, Proceedings 1*. Springer, pp. 3–11.

Mondal, Arnab Kumar et al. (2023). "Efficient Dynamics Modeling in Interactive Environments with Koopman Theory". In: *arXiv preprint arXiv:2306.11941*.

Nagabandi, Anusha, Ignasi Clavera, et al. (2018). "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning". In: *arXiv preprint arXiv:1803.11347*.

Nagabandi, Anusha, Chelsea Finn, and Sergey Levine (2018). "Deep online learning via meta-learning: Continual adaptation for model-based rl". In: *arXiv preprint arXiv:1812.07671*.

Nagabandi, Anusha, Gregory Kahn, et al. (2018). "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7559–7566.

Neisser, Ulric (2014). *Cognitive psychology: Classic edition*. Psychology press.

Nguyen-Tuong, Duy and Jan Peters (2010). "Using model knowledge for learning inverse dynamics". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2677–2682.

– (2011). "Incremental online sparsification for model learning in real-time robot control". In: *Neurocomputing* 74.11, pp. 1859–1867.

Nguyen-Tuong, Duy, Jan Peters, and Matthias Seeger (2008). "Local Gaussian process regression for real time online model learning". In: *Advances in neural information processing systems* 21.

Nguyen-Tuong, Duy, Matthias Seeger, and Jan Peters (2009). "Model learning with local gaussian process regression". In: *Advanced Robotics* 23.15, pp. 2015–2034.

Nie, Yuqi et al. (2023). "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers". In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Jbdc0vTOcol.

Oh, Junhyuk et al. (2015). "Action-conditional video prediction using deep networks in atari games". In: *Advances in neural information processing systems*, pp. 2863–2871.

Parkinson, Carolyn and Thalia Wheatley (2015). "The repurposed social brain". In: *Trends in Cognitive Sciences* 19.3, pp. 133–141.

Pezzulo, Giovanni, Francesco Rigoli, and Karl J Friston (2018). "Hierarchical active inference: a theory of motivated control". In: *Trends in cognitive sciences* 22.4, pp. 294–306.

Polydoros, Athanasios S, Lazaros Nalpantidis, and Volker Krüger (n.d.). "Real-time deep learning of robotic manipulator inverse dynamics". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3442–3448.

Precup, Doina and Richard S Sutton (1997). "Multi-time models for temporally abstract planning". In: *Advances in neural information processing systems* 10.

Punjani, Ali and Pieter Abbeel (2015). "Deep learning helicopter dynamics models". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3223–3230.

Radford, Alec et al. (2019). "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8, p. 9.

Rao, Rajesh PN (1999). "An optimal estimation approach to visual perception and learning". In: *Vision research* 39.11, pp. 1963–1989.

Rasmussen, Carl Edward (2003). "Gaussian processes in machine learning". In: *Summer School on Machine Learning*. Springer, pp. 63–71.

Reuss, Moritz et al. (2022). "End-to-end learning of hybrid inverse dynamics models for precise and compliant impedance control". In: *arXiv preprint arXiv:2205.13804*.

Rueckert, Elmar et al. (2017). "Learning Inverse Dynamics Models in O (n) time with LSTM networks". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, pp. 811–816.

Sæmundsson, Steindór, Katja Hofmann, and Marc Peter Deisenroth (2018). "Meta reinforcement learning with latent variable gaussian processes". In: *arXiv preprint arXiv:1803.07551*.

Schaal, Stefan, Christopher G Atkeson, and Sethu Vijayakumar (2002). "Scalable techniques from nonparametric statistics for real time robot learning". In: *Applied Intelligence* 17.1, pp. 49–60.

Sekar, Ramanan et al. (2020). "Planning to explore via self-supervised world models". In: *International Conference on Machine Learning*. PMLR, pp. 8583–8592.

Seth, Anil (2021). *Being you: A new science of consciousness*. Penguin.

Seth, Anil K (2014). "The cybernetic Bayesian brain". In: *Open mind*. Open MIND. Frankfurt am Main: MIND Group.

Shaj, Vaisakh, Philipp Becker, et al. (2020). "Action-Conditional Recurrent Kalman Networks For Forward and Inverse Dynamics Learning". In: *Conference On Robot Learning*.

Shaj, Vaisakh, Dieter Buchler, et al. (2022). "Hidden Parameter Recurrent State Space Models For Changing Dynamics Scenarios". In: *International Conference On Learning Representations*.

Shaj, Vaisakh, SALEH Gholam Zadeh, et al. (2023). "Multi Time Scale World Models". In: *Advances in Neural Information Processing Systems* 36.

Shanahan, Murray and Melanie Mitchell (2022). "Abstraction for deep reinforcement learning". In: *arXiv preprint arXiv:2202.05839*.

Smith, Jimmy TH, Andrew Warrington, and Scott Linderman (2022). "Simplified State Space Layers for Sequence Modeling". In: *The Eleventh International Conference on Learning Representations*.

Sonker, Rohit and Ashish Dutta (2020). "Adding Terrain Height to Improve Model Learning for Path Tracking on Uneven Terrain by a Four Wheel Robot". In: *IEEE Robotics and Automation Letters* 6.1, pp. 239–246.

Sousa, C. D. and R. Cortesao (2019). "Inertia Tensor Properties in Robot Dynamics Identification: A Linear Matrix Inequality Approach". In: *IEEE/ASME Transactions on Mechatronics* 24.1, pp. 406–411.

Srkk, Simo (2013). *Bayesian Filtering and Smoothing*.

Sutton, Richard S (1995). "TD models: Modeling the world at a mixture of time scales". In: *Machine Learning Proceedings 1995*. Elsevier, pp. 531–539.

Taborsky, Barbara and Rui F Oliveira (2012). "Social competence: an evolutionary approach". In: *Trends in ecology & evolution* 27.12, pp. 679–688.

Taylor, C James and David Robertson (2013). "State-dependent control of a hydraulically actuated nuclear decommissioning robot". In: *Control Engineering Practice* 21.12, pp. 1716–1725.

Taylor, Graham W and Geoffrey E Hinton (2009). "Factored conditional restricted Boltzmann machines for modeling motion style". In: *Proceedings of the 26th annual international conference on machine learning*. ACM, pp. 1025–1032.

Toussaint, Marc et al. (2009). "Probabilistic inference as a model of planned behavior." In: *Künstliche Intell.* 23.3, pp. 23–29.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Volpp, Michael et al. (2020). "Bayesian Context Aggregation for Neural Processes". In: *International Conference on Learning Representations*.

Von Helmholtz, Hermann (2013). *Treatise on Physiological Optics, volume III*. Vol. 3. Courier Corporation.

Watson, Joe, Hany Abdulsamad, and Jan Peters (2020). "Stochastic optimal control as approximate input inference". In: *Conference on Robot Learning*. PMLR, pp. 697–716.

Werbos, Paul J (1990). "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10, pp. 1550–1560.

Williams, Grady et al. (2017). "Information theoretic mpc for model-based reinforcement learning". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 1714–1721.

Zaheer, Manzil et al. (2017). "Deep sets". In: *arXiv preprint arXiv:1703.06114*.

Zečević, Matej et al. (2023). "Causal parrots: Large language models may talk causality but are not causal". In: *arXiv preprint arXiv:2308.13067*.

Zeng, Ailing et al. (2022). "Are transformers effective for time series forecasting?" In: *arXiv preprint arXiv:2205.13504*.

Zhou, Haoyi et al. (2021). "Informer: Beyond efficient transformer for long sequence time-series forecasting". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12, pp. 11106–11115.

A. Appendix: Action Conditional SSM

A.1. Inverse Dynamics Learning with Action Conditional SSM

For the inverse dynamics case we want to learn a model $f^{-1} : \vec{o}_{1:t}, \vec{a}_{1:t-1}, \vec{o}_{t+1} \mapsto \hat{\vec{a}}_t$ where $\vec{o}_{t+1}$ is the desired next observation for time step $t+1$ and $\hat{\vec{a}}_t$ is the predicted action, i.e., the one to be applied. We introduce an action decoder which decodes the latent posterior $\left(\vec{z}_t^+, \vec{\Sigma}_t^+\right)$ and estimates the action required to move to the desired next observation. The action decoder also gets information regarding the next observation as input. During training, this corresponds to the next observation in the data. For control, a desired next observation is used to obtain the action required to reach that observation. The joint angles and velocities are treated as observations in our inverse dynamics experiments.

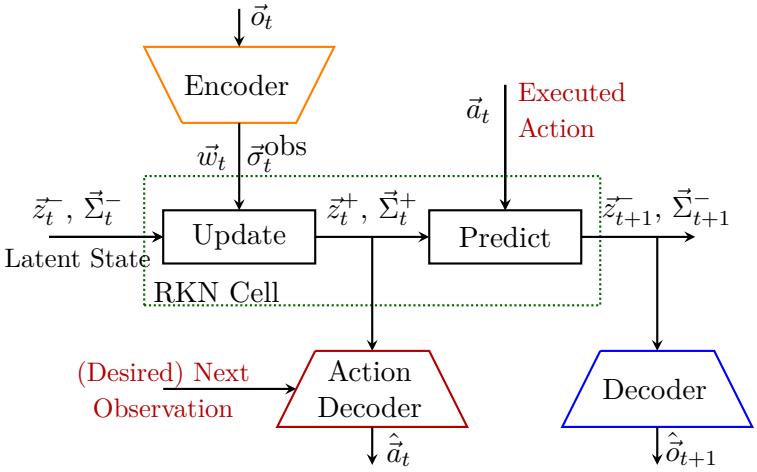


Figure A.1.: Schematic diagram of the inverse dynamics learning architecture. Here the posterior $\left(\vec{z}_t^+, \vec{\Sigma}_t^+\right)$, at the current time step is fed to an action decoder along with the desired target. In the next time-step, the executed action $\vec{a}_t$ is fed to the predict stage of the RKN cell. Further, the next predicted prior $\left(\vec{z}_{t+1}^-, \vec{\Sigma}_{t+1}^-\right)$ is decoded to get the next state $\hat{\vec{o}}_{t+1}$.

As seen in Figure A.1, instead of merely learning the inverse dynamics, our approach learns inverse and forward dynamics simultaneously by feeding back the executed action to the action-conditional predict stage, which predicts forward in latent space. Note that the action decoder also gets the same action as the target during training. However, the model sees the true action only after the prediction since the action decoder works with

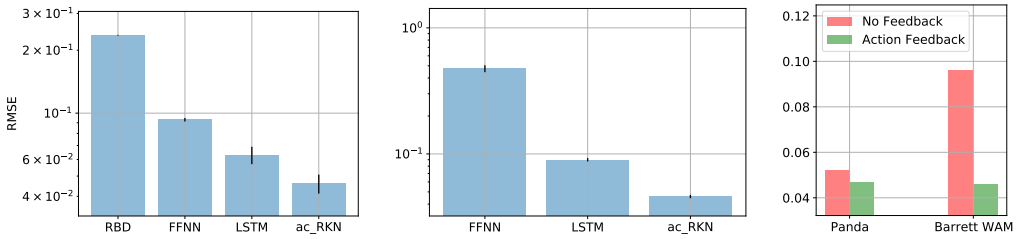
the posterior estimate $\left(\vec{z}_t^+, \vec{\Sigma}_t^+\right)$. Hence, the prediction of the inverse dynamics model for the current time step is made independent of this feedback. We found that enforcing this causal feedback, a necessary structural component of the Bayesian network of the underlying latent dynamical system, improves the performance of the inverse model, as seen in Figure A.2c.

Loss And Training. The dual output architecture for our inverse model leads to two different loss functions for the action and observation decoders that are optimized jointly. Our experiments showed that the loss of the forward model is an excellent auxiliary loss function for the inverse model, and learning this implicit forward model jointly with the inverse model results in much better performance for the inverse model. We assume that the reason for this effect is that the forward model loss provides crucial gradient information to form an informative latent state representation that is also useful for the inverse model. To deal with high-frequency data, we again chose to predict the normalized differences to the previously executed action, that is, $\hat{\vec{a}}_t = \vec{a}_{t-1} + \mathrm{dec}_{\mathrm{action}}(\vec{z}_t^+)$. Here $\vec{a}_{t-1}$ is the executed action at $t-1$ and $\mathrm{dec}_{\mathrm{action}}(\vec{z}_t^+)$ the output of the actual action decoder network. The combined loss function for the inverse dynamic case is given by $\mathrm{L}_{\mathrm{inv}} = \sqrt{1T \sum_{i=1}^{T} \parallel (\vec{a}_{t+1} - \vec{a}_t) - \mathrm{dec}_{\mathrm{action}}\left(\vec{z}_t^+\right) \parallel^2} + \lambda \mathcal{L}_{\mathrm{fwd}}$ where λ chooses the trade-off between the inverse model loss and the forward model loss. The value of λ is chosen via hyperparameter optimization using GPyOpt (authors 2016). Note that for the inverse model, the actions are decoded based on the posterior mean of the current time step, while for the forward model, the observations are decoded from the prior mean of the next time step.

We evaluated the performance of the proposed method for inverse dynamics learning on two real robots, Franka Emika Panda and Barrett WAM. Barret WAM is a robot with direct cable drives. Direct cable drives produce high torques, generating fast and dexterous movements but produce complex dynamics. The rigid body dynamics cannot accurately model these complex dynamics due to the variable stiffness and lengths of the cables (Nguyen-Tuong and Peters 2011).

Joint Torque Prediction Task. We benchmark the representational capability of the latent state posterior, $\vec{z}_t^+$, of ac-RKN in accurately modelling the inverse dynamics of these robots in comparison to deterministic models like LSTMs and FFNN. It is clear from A.2a and A.2b that ac-RKN learns highly accurate models of these in contrast with other data-driven methods. This highly precise modelling is often a requirement for high fidelity and compliant robotic control.

Impact of Action Feedback. We also perform an ablation study with and without the action feedback for the prediction step in the latent dynamics. As seen in Figure A.2c, the action feedback always results in better representational capability as this helps the implicit forward model in making better predictions by taking into account its causal effect on the state transitions.



(a) Franka Panda **(b)** Barret WAM **(c)** Action Feedback

Figure A.2.: **(a)** and **(b)** Joint torque prediction RMSE values in NM of Action-Conditional RKN, LSTM and FFNN for Panda and Barret WAM. A comparison is also provided with the analytical (RBD) model of Panda. **(c)** Comparison of ac-RKN for learning inverse dynamics with and without action feedback as discussed in Section A.1.

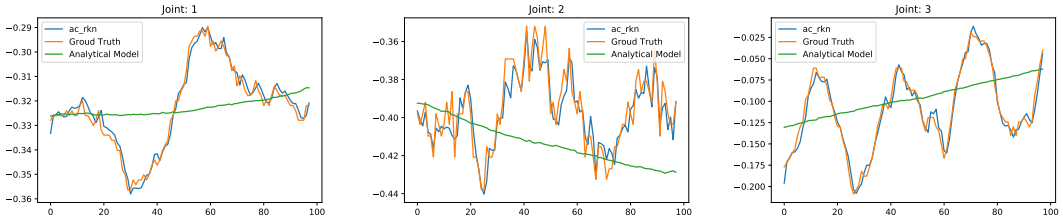


Figure A.3.: Predicted joint torques(normalized) for first 3 joints of the Panda robot arm. The learned inverse dynamics model by ac-RKN matches closely with the ground truth data, while the rigid body dynamics model cannot capture the high-frequency variations in the data.

Comparison to Analytical Models. Finally, we make a comparison with the analytical model of the Panda robot for inverse dynamics. Please refer to Appendix C for more details on the analytical model. As evident from Figure A.3 analytical models gave predictions with much lesser accuracy in comparison to ac-RKN, as it does not consider unmodelled effects such as joint and link flexibilities, backlash, stiction and actuator dynamics. In such cases, the robot would continuously have to track its current position and compensate the errors with high-gain feedback control thus making it dangerous to interact with the real world and impossible to work in human-centred environments.

A.2. Implementation Details

Locally Linear Transition Model

The state transitions in the predict stage of the Kalman filter is governed by a locally linear transition model. To obtain a locally linear transition model, the RKN learns K constant transition matrices $\vec{A}^{(k)}$ and combines them using state-dependent coefficients $\alpha^{(k)}(\vec{z}_t)$, i.e., $\vec{A}_t = \sum_{k=0}^{K} \alpha^{(k)}(\vec{z}_t)\vec{A}^{(k)}$. A small neural network with softmax output is used to learn $\alpha^{(k)}$. Each $\vec{A}^{(k)}$ is designed to consist of four band matrices as in Becker,

Pandya, et al. 2019 to reduce the number of parameters without affecting the performance.

A.3. Details Of Rigid Body Dynamics Model

The analytical model for Franka Emika Panda is a rigid-body dynamics model that was identified in its so-called base parameters (Khalil, Gautier, and Enguehard 1991). Due to the friction compensation in the joints, we observed that the viscous friction is negligible, whereas the observed Coulomb friction is very small yet included in our parameterization. This results in a model with 50 parameters. The base parameterization is computed based on the provided kinematic properties of the robotic arm and provides a linear relation between the base parameters and joint torques for a given set of joint positions, velocities and accelerations.

Due to this linearity, the regression problem can be solved using a linear least-squares method, although additional linear matrix inequality constraints must be fulfilled to ensure that the resulting parameters are physically realizable (Sousa and Cortesao 2019; Reuss et al. 2022). In order to perform a forward simulation of the robot dynamics, we numerically solve an initial value problem for the implicit set of differential equations defined by the base parameterization of the rigid-body model. Note that, the model does not parameterize actuator dynamics, nor does it model joint flexibilities, link flexibilities, or stiction. The focus here is to provide a reference baseline to show which effects the acRKN captures in comparison to a text-book robot model.

B. Appendix: Hidden Parameter SSM

B.1. Proof For Gaussian Identity 5.3.1

This section provides a proof for the Gaussian identity 5.3.1. First we derive an expression for the joint distribution $p(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{y})$.

Gaussian Identity B.1.1 (Joint Gaussian Distribution)**.** *If $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{\mu}_u + b, \boldsymbol{\Sigma}_u)$ and $v \sim \mathcal{N}(\boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v)$ are normally distributed independent random variables and if conditional distribution $p(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{v}) = \mathcal{N}(\boldsymbol{A}\boldsymbol{u} + b + \boldsymbol{B}v, \boldsymbol{\Sigma})$, the joint distribution has an expression as follows:*

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_u \\ \boldsymbol{\mu}_v \\ \mathbf{A}\mu_u + b + \mathbf{B}\mu_v \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_u & 0 & \boldsymbol{\Sigma}_u\mathbf{A}^\top \\ 0 & \boldsymbol{\Sigma}_v & \boldsymbol{\Sigma}_v\mathbf{B}^\top \\ \mathbf{A}\boldsymbol{\Sigma}_u^\top & \mathbf{B}\boldsymbol{\Sigma}_v^\top & \mathbf{A}\boldsymbol{\Sigma}_u\mathbf{A}^\top + \mathbf{B}\boldsymbol{\Sigma}_v\mathbf{B}^\top + \boldsymbol{\Sigma} \end{pmatrix}\right)$$

Proof for Identity B.1.1

Let the displacement of a variable $\mathbf{u}$ be denoted by $\Delta\mathbf{u} = \mathbf{u} - \langle\mathbf{u}\rangle$.

Since $\mathbf{u}$ and $\mathbf{v}$ are independent, covariances $\left\langle \Delta\mathbf{u}\Delta\mathbf{v}^\top \right\rangle = 0$.

We can write $\mathbf{y} = \mathbf{A}\boldsymbol{u} + b + \mathbf{B}\boldsymbol{v} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and b is a constant. Then we have covariance $\left\langle \Delta\mathbf{u}\Delta\mathbf{y}^\top \right\rangle = \left\langle \Delta\mathbf{u}(\mathbf{A}\Delta\mathbf{u} + \mathbf{B}\Delta\mathbf{v} + \Delta\epsilon)^\top \right\rangle = \left\langle \Delta\mathbf{u}\Delta\mathbf{u}^\top \right\rangle \mathbf{A}^\top + \left\langle \Delta\mathbf{u}\Delta\mathbf{v}^\top \right\rangle \mathbf{B}^\top + \left\langle \Delta\mathbf{u}\Delta\epsilon^\top \right\rangle$. Since $\left\langle \Delta\mathbf{u}\Delta\mathbf{v}^\top \right\rangle = \left\langle \Delta\mathbf{u}\Delta\epsilon^\top \right\rangle = 0$ we therefore have $\left\langle \Delta\mathbf{u}\Delta\mathbf{y}^\top \right\rangle = \boldsymbol{\Sigma}_u\mathbf{A}^\top$.

The derivations for covariances $\left\langle \Delta\mathbf{v}\Delta\mathbf{y}^\top \right\rangle$ follow in a similar way, and the corresponding covariance has the expression $\left\langle \Delta\mathbf{v}\Delta\mathbf{y}^\top \right\rangle = \boldsymbol{\Sigma}_v\mathbf{B}^\top$.

Similarly, $\left\langle \Delta\mathbf{y}\Delta\mathbf{y}^\top \right\rangle = \left\langle (\mathbf{A}\Delta\mathbf{u} + \mathbf{B}\Delta\mathbf{v} + \Delta\epsilon)(\mathbf{A}\Delta\mathbf{u} + \mathbf{B}\Delta\mathbf{v} + \Delta\epsilon)^\top \right\rangle = \mathbf{A}\left\langle \Delta\mathbf{u}\Delta\mathbf{u}^\top \right\rangle \mathbf{A}^\top + \mathbf{B}\left\langle \Delta\mathbf{v}\Delta\mathbf{v}^\top \right\rangle \mathbf{B}^\top + \left\langle \Delta\epsilon\Delta\epsilon^\top \right\rangle = \mathbf{A}\boldsymbol{\Sigma}_u\mathbf{A}^\top + \mathbf{B}\boldsymbol{\Sigma}_v\mathbf{B}^\top + \Sigma$. The result follows.

Gaussian Identity B.1.2 (Gaussian Marginalization)**.** *If*

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_u \\ \boldsymbol{\mu}_v \\ \boldsymbol{\mu}_z \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{uu} & \boldsymbol{\Sigma}_{uv} & \boldsymbol{\Sigma}_{uy} \\ \boldsymbol{\Sigma}_{uv}^\top & \boldsymbol{\Sigma}_{vv} & \boldsymbol{\Sigma}_{vy} \\ \boldsymbol{\Sigma}_{uy}^\top & \boldsymbol{\Sigma}_{vy}^\top & \boldsymbol{\Sigma}_{yy} \end{pmatrix}\right)$$

then marginal over y is given as $p(\boldsymbol{y}) = \int_{\boldsymbol{u},\boldsymbol{v}} p(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{v})p(\boldsymbol{u})p(\boldsymbol{v})d\boldsymbol{u}d\boldsymbol{v} = \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy})$

Proof For Identity B.1.2

We refer to Bishop 2006 for the derivation, which requires calculation of the Schur complement as well as completing the square of the Gaussian p.d.f. to integrate out the variable. The given derivation (Bishop 2006) for two variable multivariate Gaussians can be extended to 3 variable case WLOG.

Proof For Identity 5.3.1 is immediate from Identity B.1.1 and Identity B.1.2.

B.2. Additional Experiments

B.2.1. Comparison To Soft Switching Baseline

We also implement a soft-switching baseline similar to Fraccaro et al. 2017, where a soft mixture of dynamics is implemented in the latent transition dynamics (Kalman time update). Similar to Fraccaro et al. 2017, we now globally learn K constant transition matrices $\boldsymbol{A}^{(k)}$ and control matrices $\boldsymbol{B}^{(k)}$. An interpolation is done between these using a "dynamics parameter network" (Fraccaro et al. 2017) $\alpha_t = \boldsymbol{\alpha}_t(\mathbf{w}_{0:t-1})$. The dynamics parameter network is implemented with a recurrent neural network with LSTM cells that takes at each time step the mean of the encoded observation w_t as input and recurses $\mathbf{d}_t = LSTM(\mathbf{w}_{t-1}, \mathbf{d}_{t-1})$ and $\boldsymbol{\alpha}_t = softmax(\mathbf{d}_t)$. The output of the dynamics parameter network is weights that sum to one, $\sum_{k=1}^{K} \alpha_t^{(k)}(\mathbf{w}_{0:t-1}) = 1$. These weights choose and interpolate between K different operating modes:

$$\mathbf{A}_t = \sum_{k=1}^{K} \alpha_t^{(k)}(\mathbf{w}_{0:t-1})\,\mathbf{A}^{(k)}, \quad \mathbf{B}_t = \sum_{k=1}^{K} \alpha_t^{(k)}(\mathbf{w}_{0:t-1})\,\mathbf{B}^{(k)}$$

The authors interpret the weighted sum as a soft mixture of K different Linear Gaussian SSMs whose time-invariant matrices are combined using the time-varying weights $\boldsymbol{\alpha}_t$. In practice, each of the K sets $\left\{\mathbf{A}^{(k)}, \mathbf{B}^{(k)}\right\}$ models different/changing dynamics, which will dominate when the corresponding $\alpha_t^{(k)}$ is high.

Figure B.1 compares HiP-RSSM with different recurrent architectures, including the soft-switching baseline. As seen in Figure B.1, HiP-RSSM clearly outperforms the soft-switching baseline both in terms of convergence speed and also mult-step ahead predictions. More details on the multistep training procedure can be found in the Appendix B.2.2.

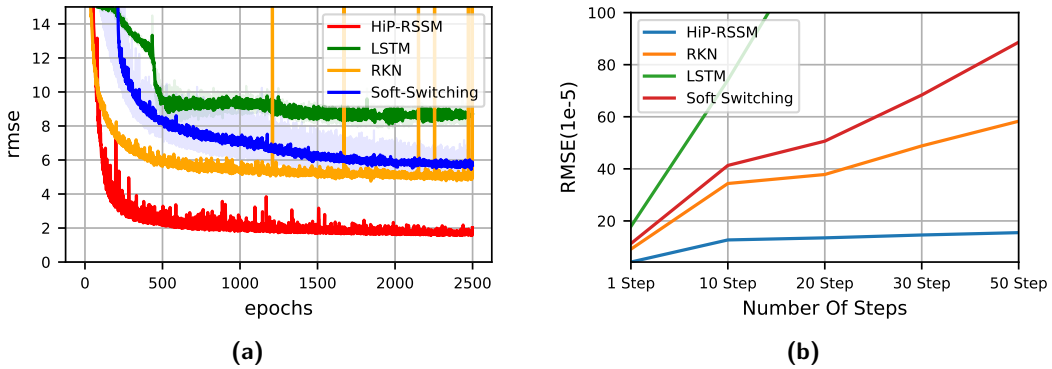


Figure B.1.: Comparison of different algorithms for Wheeled Mobile Robot in terms of (a) moving average of decoder error in normalized RMSE for the test set, plotted against training epochs (b) multi-step ahead prediction error in RMSE.

B.2.2. Multi-Step Ahead Predictions

In figure B.1b, we compare the results of multi-step ahead predictions for up to 50 steps of HiP-RSSM with recurrent baselines like RKN(Becker, Pandya, et al. 2019) and LSTM. In order to train the recurrent models for multi step ahead predictions, we removed three-quarters of the observations from the temporal sequence and tasked the models with imputing those missing observations, only based on the knowledge of available actions/control commands, i.e., we train the models to perform action conditional future predictions to impute missing observations. The imputation employs the model for multi-step ahead predictions in a convenient way (Shaj, Becker, et al. 2020). One could instead also go for a dedicated multi-step loss function as in approaches like Finn, Goodfellow, and Levine 2016.

As seen in figure B.1b, HiP-RSSM clearly outperforms contemporary recurrent models for multi-step ahead prediction tasks since it takes into account additional causal factors of variation (slopes of the terrain for this robot) in the latent dynamics in an unsupervised manner.

B.3. Implementation Details

B.3.1. Context Set Encoder and Latent Task Representation

The HiP-RSSM maps a set of previous interaction histories, $\{\boldsymbol{o}^l_{t,n}, \boldsymbol{a}^l_{t,n}, \boldsymbol{o}^l_{t+1,n}\}_{n=1}^N$, to a set of latent features and an estimate of uncertainty in these features, $\{\boldsymbol{r}^l_n, (\boldsymbol{\sigma}^l_n)^2\}_{n=1}^N$, using a context encoder. We use a feed-forward neural network as the encoder in all of our experiments, since we deal with high-dimensional vectors as our observations. However, depending upon the nature of the observations, we could use different encoder

architectures.
The set of latent features and uncertainties, $\{\boldsymbol{r}_n^l, \left(\boldsymbol{\sigma}_n^{\boldsymbol{l}}\right)^2\}_{n=1}^N$, is further aggregated in a probabilistically principled manner using the Bayesian aggregation operator discussed in 5.3.1 to obtain a Gaussian latent task variable, with a mean ($\boldsymbol{\mu}_l$) and diagonal covariance ($\boldsymbol{\sigma}_l$). Intuitively, the context encoder learns to weight the contribution from each observation in the context set based on Bayesian principles and emits a probabilistic representation of the latent task.

B.3.2. Latent Task Transformation Model

To achieve latent task conditioning within the recurrent cell, we include a task transformation model ($\boldsymbol{c}$), in addition to the locally linear transition model $\boldsymbol{A}_t$ and control model $\boldsymbol{b}$ in the time update stage (section 5.3.2). Though in section 5.3.2, we used the notation for a linear task transformation matrix, $\boldsymbol{C}$, to motivate the additive interaction of latent task variables, $\boldsymbol{\mu}_l$ and $\boldsymbol{\sigma}_l$ in the latent space, the task transformation function can be designed in several ways, i.e.:

(i) Linear: $\boldsymbol{c} = \boldsymbol{C}$, where $\boldsymbol{C}$ is a linear transformation matrix. The corresponding time update equations are given below: $\boldsymbol{z}_t^- = \mathbf{A}_{t-1}\boldsymbol{z}_{t-1}^+ + \mathbf{b}(\boldsymbol{a}_t) + \mathbf{C}\boldsymbol{\mu}_{\boldsymbol{l}}$,
$\boldsymbol{\Sigma}_t^- = \mathbf{A}_{t-1}\boldsymbol{\Sigma}_{t-1}^+\mathbf{A}_{t-1}^T + \mathbf{C}(\mathbf{I}\cdot\sigma_l)\mathbf{C}^T + \boldsymbol{\Sigma}_{\text{trans}}$.

(ii) Locally-Linear: $\boldsymbol{c} = \boldsymbol{C}_t$, where $\boldsymbol{C}_t = \sum_{k=0}^K \beta^{(k)}(\boldsymbol{z_t})\boldsymbol{C}^{(k)}$ is a linear combination of k linear control models $\boldsymbol{C}^{(k)}$. A small neural network with softmax output is used to learn $\beta^{(k)}$. The corresponding time update equations are given below:
$\boldsymbol{z}_t^- = \mathbf{A}_{t-1}\boldsymbol{z}_{t-1}^+ + \mathbf{b}(\boldsymbol{a}_t) + \mathbf{C_t}\boldsymbol{\mu}_{\boldsymbol{l}}$,
$\boldsymbol{\Sigma}_t^- = \mathbf{A}_{t-1}\boldsymbol{\Sigma}_{t-1}^+\mathbf{A}_{t-1}^T + \mathbf{C_t}(\mathbf{I}\cdot\sigma_l)\mathbf{C_t}^T + \boldsymbol{\Sigma}_{\text{trans}}$.

(iii) Non-Linear: $\boldsymbol{c} = \boldsymbol{f}$, where $\boldsymbol{f}(.)$ can be any non-linear function approximator. We use a multi-layer neural network regressor with ReLU activations, which transforms the latent task moments $\boldsymbol{\mu_l}$ and $\boldsymbol{\sigma_l}$ directly into the latent space of the state space model via additive interactions. The corresponding time update equations are given below: $\boldsymbol{z}_t^- = \mathbf{A}_{t-1}\boldsymbol{z}_{t-1}^+ + \mathbf{b}(\boldsymbol{a}_t) + \mathbf{f}(\boldsymbol{\mu}_l)$,
$\boldsymbol{\Sigma}_t^- = \mathbf{A}_{t-1}\boldsymbol{\Sigma}_{t-1}^+\mathbf{A}_{t-1}^T + \mathbf{f}(\boldsymbol{\sigma}_l) + \boldsymbol{\Sigma}_{\text{trans}}$.

In our ablation study (Figure 5.9), for the linear and locally linear task transformation models, we assume that the dimension of the latent context variable $\boldsymbol{l}$ and the latent state space $\boldsymbol{z}_t$ are equal. This allows us to work with square matrices, which are more convenient. For the non-linear transformation we are free to choose the size of the latent context variable. However for a fairer comparison we keep the dimension of latent task variable to be similar in all three cases. We choose the non-linear task transformation model in HiP-RSSM architecture as this gave the best performance in practice.

C. Appendix: Multi Time Scale SSM

C.1. Proofs and Derivations

In the following sections, vectors are denoted by a lowercase letter in bold, such as "$\mathbf{v}$", while matrices are denoted by an uppercase letter in bold, such as "$\mathbf{M}$". $\mathbf{I}$ denotes identity matrix and $\mathbf{0}$ represents a matrix filled with zeros. For any matrix $\mathbf{M}$, $\boldsymbol{m}$ denotes the corresponding vector of diagonal entries. Also, $\odot$ denotes the elementwise vector product and $\oslash$ denotes an elementwise vector division.

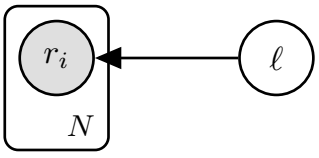


Figure C.1.: Graphical Model For Bayesian conditioning with N observations.

C.1.1. Proof For Bayesian Conditioning As Permutation Invariant Set Operations (Identity 6.2.1)

Gaussian Identity 6.2.1 (Gaussian Conditioning)**.** *Consider the graphical model given in Figure 6.6, where a set of N conditionally i.i.d observations* $\boldsymbol{r} = \{\boldsymbol{r}_i\}_{i=1}^N$ *are generated by a latent variable* $\boldsymbol{l}$ *and the observation model* $p(\boldsymbol{r}_i|\boldsymbol{l}) = \mathcal{N}\left(\boldsymbol{r}_i \mid \mathbf{H}\boldsymbol{l}, diag(\sigma_i^{obs})\right)$*. Assuming an observation model* $\mathbf{H} = [\mathbf{I}, \mathbf{0}]$*, the mean (*$\boldsymbol{\mu}$*) and precision matrix (*$\boldsymbol{\Lambda}$*) of the posterior over the latent variable* $\boldsymbol{l}$*,* $p(\boldsymbol{l}|\boldsymbol{r}) = \mathcal{N}\left(\boldsymbol{\mu}_l^+, \boldsymbol{\Sigma}_l^+\right) = \mathcal{N}\left(\boldsymbol{\mu}_l^+, (\boldsymbol{\Lambda}_l^+)^{-1}\right)$*, given the prior* $p_0(\boldsymbol{l}) = \mathcal{N}\left(\boldsymbol{\mu}_l^-, \boldsymbol{\Sigma}_l^-\right) = \mathcal{N}\left(\boldsymbol{\mu}_l^-, (\boldsymbol{\Lambda}_l^-)^{-1}\right)$ *have the following permutation invariant closed form updates.*

$$\boldsymbol{\Lambda}_l^+ = \boldsymbol{\Lambda}_l^- + \begin{bmatrix} diag(\sum_{i=1}^n \frac{1}{\boldsymbol{\sigma}_i^{obs}}), & \mathbf{0} \\ \mathbf{0}, & \mathbf{0} \end{bmatrix} \boldsymbol{\mu}_l^+ = \boldsymbol{\mu}_l^- + \begin{bmatrix} \boldsymbol{\sigma}_l^{u+} \\ \boldsymbol{\sigma}_l^{s+} \end{bmatrix} \odot \begin{bmatrix} \sum_{i=1}^N \left(\boldsymbol{r}_i - \boldsymbol{\mu}_l^{\mathrm{u},-}\right) \odot \frac{1}{\boldsymbol{\sigma}_i^{obs}} \\ \sum_{i=1}^N \left(\boldsymbol{r}_i - \boldsymbol{\mu}_l^{\mathrm{u},-}\right) \odot \frac{1}{\boldsymbol{\sigma}_i^{obs}} \end{bmatrix} \tag{6.2}$$

Note that $\boldsymbol{\Sigma}_l$ is the covariance matrix which is the inverse of the precision matrix $\boldsymbol{\Lambda}_l$. Due to the observation model assumption $\mathbf{H} = [\mathbf{I}, \mathbf{0}]$, they take block diagonal form,

$$\boldsymbol{\Sigma}_l = \begin{bmatrix} \boldsymbol{\Sigma}_l^u & \boldsymbol{\Sigma}_l^s \\ \boldsymbol{\Sigma}_l^s & \boldsymbol{\Sigma}_l^l \end{bmatrix}, \text{ with } \boldsymbol{\Sigma}_u = \mathrm{diag}(\boldsymbol{\sigma}_l^u),\ \boldsymbol{\Sigma}_l = \mathrm{diag}(\boldsymbol{\sigma}_l^l) \text{ and } \boldsymbol{\Sigma}_s = \mathrm{diag}(\boldsymbol{\sigma}_l^s).$$

Proof:

Case 1 (Single Observation): Before deriving the update rule for N conditionally iid observations, let us start with a simpler case consisting of a single observation $\boldsymbol{r}$. If the marginal Gaussian distribution for the latent variable $\boldsymbol{l}$ takes the form $p(\mathbf{l}) = \mathcal{N}\left(\mathbf{l} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right)$ and the conditional Gaussian distribution for the single observation $\boldsymbol{r}$ given $\boldsymbol{l}$ has the form , $p(\mathbf{r} \mid \mathbf{l}) = \mathcal{N}\left(\mathbf{r} \mid \mathbf{H}\mathbf{l} + \mathbf{b}, \mathbf{L}^{-1}\right)$. Then the posterior distribution over $\mathbf{l}$ can be obtained in closed form as,

$$p(\mathbf{l} \mid \mathbf{r}) = \mathcal{N}\left(\mathbf{l} \mid \boldsymbol{\Sigma}\left\{\mathbf{H}^{\mathrm{T}}\mathbf{L}(\mathbf{r} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\right\}, \boldsymbol{\Lambda}^{-1}\right), \text{where } \boldsymbol{\Lambda} = \left(\boldsymbol{\Lambda} + \mathbf{H}^{\mathrm{T}}\mathbf{L}\mathbf{H}\right). \tag{C.1}$$

We refer to Section 2.3.3 of Bishop 2006, to the proof for this standard result.

Case 2 (Set Of Observations): Now instead of a single observation, we wish to derive a closed form solution for the posterior over latent variable $\boldsymbol{l} \in \mathbb{R}^{2d}$, given a set of N conditionally i.i.d observations $r = \{r_i\}_{i=1}^{N}$. Here each element $\boldsymbol{r_i} \in \mathbb{R}^d$ of the set r is assumed to to have an observation model $\mathbf{H} = [\mathbf{I}, \mathbf{0}]$. In the derivation, we represent the set of N observations as a random vector

$$r = \begin{bmatrix} \boldsymbol{r_1} \\ \boldsymbol{r_2} \\ \boldsymbol{.} \\ \boldsymbol{.} \\ \boldsymbol{r_N} \end{bmatrix}_{Nd \times 1}.$$

Since each observation in the set r are conditionally independent, we denote the conditional distribution over the context set as $r \mid \mathbf{l} \sim \mathcal{N}\left(\bar{H}\mathrm{l}, \boldsymbol{\Sigma}_r\right)$, where the diagonal covariance matrix has the following form:

$$\boldsymbol{\Sigma}_r = \begin{bmatrix} \mathrm{diag}(\boldsymbol{\sigma}_{r_1}), & 0, & 0, & .., & 0 \\ 0, & \mathrm{diag}(\boldsymbol{\sigma}_{r_2}), & 0, & .., & 0 \\ \boldsymbol{.}, & ., & ., & .., & . \\ \boldsymbol{.}, & ., & ., & .., & . \\ 0, & 0, & 0, & .., & \mathrm{diag}(\boldsymbol{\sigma}_{r_N}) \end{bmatrix}_{Nd \times Nd}.$$

The corresponding observation model $\bar{\mathbf{H}}$ is

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \mathbf{H} \\ . \\ . \\ \mathbf{H} \end{bmatrix}_{Nd \times 2d} = \begin{bmatrix} \boldsymbol{I}, \mathbf{0} \\ \boldsymbol{I}, \mathbf{0} \\ \boldsymbol{.}, \boldsymbol{.} \\ \boldsymbol{.}, \boldsymbol{.} \\ \boldsymbol{I}, \mathbf{0} \end{bmatrix}_{Nd \times 2d}.$$

Now given the prior over the latent task variable $\mathrm{l} \sim \mathcal{N}\left(\mu_l^-, \mathbf{\Sigma}_l^-\right)$, the parameters of the posterior distribution over the task variable, $p(l|r) \sim \mathcal{N}\left(\mu_l^+, \mathbf{\Lambda}_l^+\right)$, can be obtained in closed-form substituting in Equation eq:bishop1 as follows.

$$\Lambda_l^+ = (\Sigma_l^+)^{-1} = \mathbf{\Sigma}_l^{-1} + \bar{\mathbf{H}}^T \mathbf{\Sigma}_r \bar{\mathbf{H}} = \mathbf{\Sigma}_l^{-1} + \begin{bmatrix} \mathrm{diag}(\boldsymbol{\sigma}_{r_1}), \mathrm{diag}(\boldsymbol{\sigma}_{r_2}), \mathrm{diag}(\boldsymbol{\sigma}_{r_3}), ., ., \mathrm{diag}(\boldsymbol{\sigma}_{r_N}) \\ \mathbf{0}, \quad \mathbf{0}, \quad \mathbf{0}, \quad ., ., \quad \mathbf{0} \end{bmatrix}_{2d \times nd} \quad \bar{\mathbf{H}} = \boldsymbol{\lambda}_l^- + \Big[\tag{C.2}$$

$$\mu_l^+ = \boldsymbol{\mu}_l^- + (\Lambda^+)^{-1} \bar{\mathbf{H}}^T \left(\sigma_r^{-2} \boldsymbol{I}\right) \left(\boldsymbol{y} - \bar{\mathbf{H}} \boldsymbol{\mu}_{\boldsymbol{x}}\right) = \boldsymbol{\mu}_l^- + \Sigma^+ \bar{\mathbf{H}} \left(\sigma_r^{-2} \boldsymbol{I}\right) \left(\boldsymbol{y} - \bar{\mathbf{H}} \boldsymbol{\mu}_{\boldsymbol{x}}\right) = \boldsymbol{\mu}_l^- + \Sigma^+ \begin{bmatrix} \sigma_{\boldsymbol{r_1}}^{-2} \boldsymbol{I}, \sigma_{\boldsymbol{r_2}}^{-2} \boldsymbol{I}, \sigma_{\boldsymbol{r_3}}^{-2} \boldsymbol{I} \\ \mathbf{0}, \quad \mathbf{0}, \quad \mathbf{0}, \end{bmatrix} \tag{C.3}$$

Here μ_l^+ is the posterior mean and $\mathbf{\Lambda}_l^+$ is the posterior precision matrix.

C.1.2. Derivation For Matrix Inversions as Scalar Operations

Inversion Of Block Diagonal Matrix 1. *Consider a block matrix of the following form* $\boldsymbol{A} = \begin{bmatrix} \mathrm{diag}(\boldsymbol{a}^u) & \mathrm{diag}(\boldsymbol{a}^s) \\ \mathrm{diag}(\boldsymbol{a}^s) & \mathrm{diag}(\boldsymbol{a}^l) \end{bmatrix}$. *Then inverse* $A^{-1} = \boldsymbol{B}$ *can be calculated using scalar operations and is given as,* $\boldsymbol{B} = \begin{bmatrix} \mathrm{diag}(\boldsymbol{b}^u) & \mathrm{diag}(\boldsymbol{b}^s) \\ \mathrm{diag}(\boldsymbol{b}^s) & \mathrm{diag}(\boldsymbol{b}^l) \end{bmatrix}$ *where,* $\boldsymbol{b}^u = \boldsymbol{a}_l \oslash (\boldsymbol{a}_u \odot \boldsymbol{a}_l - \boldsymbol{a}_s \odot \boldsymbol{a}_s)$
$\boldsymbol{b}^s = -\boldsymbol{a}_s \oslash (\boldsymbol{a}_u \odot \boldsymbol{a}_l - \boldsymbol{a}_s \odot \boldsymbol{a}_s)$
$\boldsymbol{b}^l = \boldsymbol{a}_u \oslash (\boldsymbol{a}_u \odot \boldsymbol{a}_l - \boldsymbol{a}_s \odot \boldsymbol{a}_s)$.

Proof: To prove this we will use the following matrix identity of a partitioned matrix from Bishop 2006, which states $\begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathrm{M} & -\mathrm{MBD}^{-1} \\ -\mathrm{D}^{-1}\mathrm{CM} & \mathrm{D}^{-1} + \mathrm{D}^{-1}\mathrm{CMBD}^{-1} \end{pmatrix}$ where M is defined as

$$\mathrm{M} = \left(\mathrm{A} - \mathrm{BD}^{-1}\mathrm{C}\right)^{-1}.$$

Here M is called the Schur complement of the Matrix on the left side of Equation C.1.2. The algebraic manipulations to arrive at scalar operations in Equation 1 are straightforward.

C.1.3. Proof for Permutation Invariance (Theorem 6.2.1)

Theorem 6.2.1 (Permutation Invariance Of Bayesian Inversion)**.** *For any conditionally i.i.d model where you have a global parameter* θ*, and a set of observations* $X = \{x_i\}_{i=1}^N$ *drawn conditionally i.i.d from a distribution* $p(X \mid \theta)$*, then for any permutation* π*, the posterior* $p(\theta \mid x_1, \ldots, x_N) = p(\theta \mid \pi(x_1), \ldots, \pi(x_N))$*. Thus the posterior* $p(\theta \mid X)$ *is permutation invariant with respect to the set* X*.*

Proof. The posterior distribution $p(\theta \mid X)$, defined by Bayes' theorem as $p(\theta \mid X) = \frac{p(X|\theta)p(\theta)}{p(X)}$, relies on the likelihood $p(X \mid \theta) = \prod_{i=1}^{N} p(x_i \mid \theta)$. Under any permutation π, the permuted set $X^{\pi} = \{\pi(x_1), \ldots, \pi(x_N)\}$ preserves the likelihood since $p(X^{\pi} \mid \theta) = \prod_{i=1}^{N} p(\pi(x_i) \mid \theta) = \prod_{i=1}^{N} p(x_i \mid \theta) = p(X \mid \theta)$. The prior $p(\theta)$ and the marginal likelihood $p(X)$ are invariant under permutation. Therefore, $p(\theta \mid X^{\pi}) = p(\theta \mid X)$, establishing the permutation invariance of the posterior. □

C.1.4. Proof for Gaussian Marginalization (Identity 6.2.2)

Gaussian Identity 6.2.2 (Linear Combination Gaussian Marginalization)**.** *Consider the graphical model in Figure 6.7, where a set of N normally distributed independent random variables* $u = \{\boldsymbol{u_i} \sim \mathcal{N}(\boldsymbol{\mu_{u_i}}, \boldsymbol{\Sigma_{u_i}})\}_{i=0}^{N}$ *forms a "common effect/V Structure" with a latent variable* $\boldsymbol{y}$*. If the conditional distribution* $p\left(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}, .., \boldsymbol{u_N}\right) = \mathcal{N}(\sum_{i=0}^{N} \mathbf{A_i}\boldsymbol{u_i}, \boldsymbol{\Sigma})$*, then marginal* $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}, .., \boldsymbol{u_N}) \prod_{i=1}^{N} p(\boldsymbol{u_i}) d\boldsymbol{u_i} = \mathcal{N}(\sum_{i=1}^{N} \mathbf{A_i}\boldsymbol{\mu_{u_i}}, \boldsymbol{\Sigma} + \sum_{i=1}^{N} \mathbf{A_i}\boldsymbol{\Sigma}_{u_i}\mathbf{A_i}^{T})$.

Proof. The proof is a staright forward extension to the Identity 5.3.1 derived in Chapter 5. First, we derive an expression for the joint distribution $p(\boldsymbol{u_1}, \boldsymbol{u_2}, .., \boldsymbol{u_N}, \boldsymbol{y})$.

Gaussian Identity C.1.1 (Joint Distribution)**.** *If* $\boldsymbol{u_1} \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $\boldsymbol{u_2} \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ *are normally distributed independent random variables and if conditional distribution* $p(\boldsymbol{y}|\boldsymbol{u_1}, \boldsymbol{u_2}) = \mathcal{N}(\boldsymbol{A_1}\boldsymbol{u_1} + \boldsymbol{A_2}\boldsymbol{u_2}, \boldsymbol{\Sigma})$*, the joint distribution has an expression as follows:*

$$\begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \vdots \\ \mathbf{u_N} \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_N \\ \sum_{n=1}^{N} \mathbf{A}_n\boldsymbol{\mu}_n \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_1 & 0 & \cdots & \mathbf{A}_1^{\top}\boldsymbol{\Sigma}_1 \\ 0 & \boldsymbol{\Sigma}_2 & \cdots & \mathbf{A}_2^{\top}\boldsymbol{\Sigma}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_1\mathbf{A}_1 & \boldsymbol{\Sigma}_2\mathbf{A}_2 & \cdots & \sum_{n=1}^{N} \mathbf{A}_n\boldsymbol{\Sigma}_n\mathbf{A}_n^{\top} + \boldsymbol{\Sigma} \end{pmatrix}\right) \tag{C.4}$$

We can write $\mathbf{y} = \sum_{n=1}^{N} \mathbf{A_n}\boldsymbol{u_n} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and b is a constant.

Let displacement of a variable $\mathbf{u}$ be denoted by $\Delta\mathbf{u} = \mathbf{u} - \langle\mathbf{u}\rangle$.

Since $\mathbf{u_i}$ and $\mathbf{u_j}$ are independent $\forall i \neq j$, the covariances

$$\left\langle \Delta\mathbf{u_i}\Delta\mathbf{u_j}^{\top} \right\rangle = 0, \forall i \neq j. \tag{C.5}$$

Similarly,

$$\left\langle \Delta\mathbf{u_i}\Delta\epsilon^{\top} \right\rangle = 0, \forall i. \tag{C.6}$$

For any i, we have the covariance

$$\left\langle \Delta \mathbf{u}_i \Delta \mathbf{y}^\top \right\rangle = \left\langle \Delta \mathbf{u}_i \left(\sum_{n=1}^{N} \mathbf{A}_n \Delta \mathbf{u}_n + \Delta \epsilon \right)^\top \right\rangle = \left\langle \Delta \mathbf{u}_i \Delta \mathbf{u}_i^\top \right\rangle \mathbf{A}_i^\top + \sum_{j=1, j\neq i}^{N} \left\langle \Delta \mathbf{u}_i \Delta \mathbf{u}_j^\top \right\rangle \mathbf{A}_j^\top + \left\langle \Delta \mathbf{u}_i \Delta \epsilon^\top \right\rangle. \tag{C.7}$$

Using equations $C.5$ and $C.6$, we therefore derive an expression for the corresponding covariance as:

$$\left\langle \Delta \mathbf{u}_i \Delta \mathbf{y}^\top \right\rangle = \mathbf{\Sigma}_i \mathbf{A}_i^\top.$$

Similarly, $\left\langle \Delta \mathbf{y} \Delta \mathbf{y}^\top \right\rangle = \left\langle \left(\sum_{n=1}^{N} \mathbf{A}_n \Delta \mathbf{u}_n + \Delta \epsilon \right) \left(\sum_{n=1}^{N} \mathbf{A}_n \Delta \mathbf{u}_n + \Delta \epsilon \right)^\top \right\rangle$
$= \sum_{n=1}^{N} \mathbf{A}_n \left\langle \Delta \mathbf{u}_n \Delta \mathbf{u}_n^\top \right\rangle \mathbf{A}_n^\top + \left\langle \Delta \epsilon \Delta \epsilon^\top \right\rangle$
$= \sum_{n=1}^{N} \mathbf{A}_n \mathbf{\Sigma}_n \mathbf{A}_n^\top + \mathbf{\Sigma}_\epsilon$. The result follows.

Gaussian Identity C.1.2 (Marginalization)**.** *If*

$$\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \\ \boldsymbol{y} \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_N \\ \boldsymbol{\mu}_y \end{pmatrix}, \begin{pmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} & \cdots & \mathbf{\Sigma}_{1N} & \mathbf{\Sigma}_{1y} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} & \cdots & \mathbf{\Sigma}_{2N} & \mathbf{\Sigma}_{2y} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{\Sigma}_{N1} & \mathbf{\Sigma}_{N2} & \cdots & \mathbf{\Sigma}_{NN} & \mathbf{\Sigma}_{Ny} \\ \mathbf{\Sigma}_{y1} & \mathbf{\Sigma}_{y2} & \cdots & \mathbf{\Sigma}_{yN} & \mathbf{\Sigma}_{yy} \end{pmatrix} \right)$$

then marginal over y is given as $p(\boldsymbol{y}) = \int p(\boldsymbol{y}, \boldsymbol{u_1}, \boldsymbol{u_2}, .., \boldsymbol{u_N}) \prod_{i=1}^{N} \boldsymbol{u_i} = \mathcal{N}(\boldsymbol{\mu}_y, \mathbf{\Sigma}_{yy})$

Proof For Identity C.1.2 We refer to Bishop 2006 for the derivation, which requires calculation of the Schur complement as well as completing the square of the Gaussian p.d.f. to integrate out the variable. The given derivation (Bishop 2006) for two variable multivariate Gaussians can be extended to the case of N variables WLOG.

Proof For Identity 6.2.2 is immediate from Identity C.1.1 and Identity C.1.2. □

C.2. Implementation Details

C.2.1. Inference In Slow Time Scale SSM

C.2.1.1. Inferring Action Abstraction (sts-SSM)

Given a set of encoded primitive actions and their corresponding variances $\{\boldsymbol{\alpha_{k,t}}, \boldsymbol{\rho_{k,t}}\}_{t=1}^{H}$, using the prior and observation model assumptions in Section 3.1.2 of main paper, we infer the latent abstract action $p(\boldsymbol{\alpha}_k | \boldsymbol{\alpha}_{k,1:H}) = \mathcal{N}(\boldsymbol{\mu}_{\alpha_k}, \mathbf{\Sigma}_{\alpha_k}) = \mathcal{N}(\boldsymbol{\mu}_{\alpha_k}, \text{diag}(\sigma_{\alpha_k}))$ as a Bayesian aggregation Volpp et al. 2020 of these using the following closed-form equations:

$$\boldsymbol{\sigma}_{\alpha_k} = \left((\boldsymbol{\sigma}_0)^{\ominus} + \sum_{n=1}^{N}\left((\boldsymbol{\rho_{k,t}})^{\ominus}\right)\right)^{\ominus},$$
$$\boldsymbol{\mu}_{\alpha_k} = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_{\alpha_k} \odot \sum_{n=1}^{N}(\boldsymbol{\alpha_{k,t}} - \boldsymbol{\mu}_0) \oslash \boldsymbol{\rho_{k,t}}$$

Here, $\ominus$, $\odot$ and $\oslash$ denote element-wise inversion, product, and division, respectively. The update equation is coded as the "abstract action inference" neural network layer as shown in Figure 1.1.

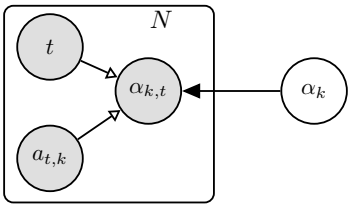


Figure C.2.: Generative model for the abstract action α_k. The hollow arrows are deterministic transformations leading to implicit distribution $\alpha_{k,t}$ using an action set encoder.

C.2.1.2. Task Prediction (sts-SSM)

The goal of this step is to update the prior marginal over the latent task variable $\boldsymbol{l}_k$, $p(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k-1}, \boldsymbol{\alpha}_{1:k})$, given the posterior beliefs from the time window $k-1$ and abstract action $\boldsymbol{\alpha}_k$.

Using the linear dynamics model assumptions from Section 3.3, we can use the following closed-form update equations to compute, $p(\boldsymbol{l}_k|\boldsymbol{\beta}_{1:k-1}, \boldsymbol{\alpha}_{1:k}) = \mathcal{N}(\boldsymbol{\mu}^-_{l_k}, \boldsymbol{\Sigma}^-_{l_k})$, where

$$[c]\boldsymbol{\mu}^-_{l_k} = \mathbf{X}\boldsymbol{\mu}^+_{l_{k-1}} + \mathbf{Y}\boldsymbol{\alpha}_k\boldsymbol{\Sigma}^-_{l_k} = \mathbf{X}\boldsymbol{\Sigma}^+_{l_{k-1}}\mathbf{X}^T + \mathbf{Y}\boldsymbol{\Sigma}_{\alpha_k}\mathbf{Y}^T + \mathbf{S}. \tag{C.8}$$

These closed-form equations are coded as the "task predict" neural net layer as shown in Figure 1.1.

C.2.1.3. Task Update (sts-SSM)

In this stage, we update the prior over l_k using an abstract observation set $\{\boldsymbol{\beta}_{k,t}\}_{t=1}^{H}$, to obtain the latent task the posterior $\mathcal{N}(\boldsymbol{\mu}^+_{z_{k,t}}, \boldsymbol{\Sigma}^+_{z_{k,t}}) = \mathcal{N}(\begin{bmatrix} \boldsymbol{\mu}^{u+}_t \\ \boldsymbol{\mu}^{l+}_t \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}^u_t & \boldsymbol{\Sigma}^s_t \\ \boldsymbol{\Sigma}^s_t & \boldsymbol{\Sigma}^l_t \end{bmatrix}^+)$, with $\boldsymbol{\Sigma}^u_{l_k} = \mathrm{diag}(\boldsymbol{\sigma}^u_{l_k})$, $\boldsymbol{\Sigma}^l_{l_k} = \mathrm{diag}(\boldsymbol{\sigma}^l_{l_k})$ and $\boldsymbol{\Sigma}^s_{l_k} = \mathrm{diag}(\boldsymbol{\sigma}^s_{l_k})$.

To do so we first invert the prior covariance matrix $\begin{bmatrix} \boldsymbol{\Sigma}^u_{l_k} & \boldsymbol{\Sigma}^s_{l_k} \\ \boldsymbol{\Sigma}^s_{l_k} & \boldsymbol{\Sigma}^l_{l_k} \end{bmatrix}^+$ to the precision matrix $\begin{bmatrix} \boldsymbol{\lambda}^u_{l_k} & \boldsymbol{\lambda}^s_{l_k} \\ \boldsymbol{\lambda}^s_{l_k} & \boldsymbol{\lambda}^l_{l_k} \end{bmatrix}^+$ for permutation invariant parallel processing. The posterior precision is then computed using scalar operations are follows, where only $\boldsymbol{\lambda}^u_{l_k}$ is changed by $\boldsymbol{\lambda}^{u+}_{l_k} = \boldsymbol{\lambda}^{u-}_{l_k} + \sum_{t=1}^{H}\mathbf{1} \oslash \boldsymbol{\nu}_{k,t}$ while $\boldsymbol{\lambda}^{l+}_{l_k} = \boldsymbol{\lambda}^{l-}_{l_k}$ and $\boldsymbol{\lambda}^{s+}_{l_k} = \boldsymbol{\lambda}^{s-}_{l_k}$ remain constant. The operator $\oslash$ denotes the element-wise division. The posterior precision is inverted back to

the posterior covariance vectors $\boldsymbol{\sigma}_{l_k}^{u+}$, $\boldsymbol{\sigma}_{l_k}^{l+}$ and $\boldsymbol{\sigma}_{l_k}^{s+}$. Now, the posterior mean $\boldsymbol{\mu}_{l,k}^{+}$ can be obtained from the prior mean $\boldsymbol{\mu}_{l,k}^{-}$ as

$$[c]\boldsymbol{\mu}_{l,k}^{+} = \boldsymbol{\mu}_{l,k}^{-} + \begin{bmatrix} \boldsymbol{\sigma}_{l_k}^{u+} \\ \boldsymbol{\sigma}_{l_k}^{s+} \end{bmatrix} \odot \begin{bmatrix} \sum_{t=1}^{H} \left(\boldsymbol{\beta}_{k,t} - \boldsymbol{\mu}_{l_k}^{\mathrm{u},-}\right) \oslash \boldsymbol{\nu}_{k,t} \\ \sum_{t=1}^{H} \left(\boldsymbol{\beta}_{k,t} - \boldsymbol{\mu}_{l_k}^{\mathrm{u},-}\right) \oslash \boldsymbol{\nu}_{k,t} \end{bmatrix}. \tag{C.9}$$

The inversion between the covariance matrix and precision matrix can be done via scalar operations leveraging block diagonal structure as derived in Appendix C.1. Figure 6.5 shows the schematic of the task update layer.

C.2.2. Inference In Fast Time Scale SSM

The inference in fts-SSM for a time-window k involves two stages as illustrated in Figure 6.9, calculating the prior and posterior over the latent state variable z_t. To keep the notation uncluttered, we will also omit the time-window index k whenever the context is clear as in Section 3.2.

C.2.2.1. Task Conditional State Prediction (fts-SSM)

Following the assumptions of a task conditional linear dynamics as in Section 3.2 of the main paper, we obtain the prior marginal for $p(\boldsymbol{z}_{k,t}|\boldsymbol{w}_{1:t-1}^{k}, \boldsymbol{a}_{1:t-1}^{k}, \boldsymbol{\beta}_{1:k-1}, \boldsymbol{\alpha}_{1:k-1}) = \mathcal{N}(\boldsymbol{\mu}_{z_{k,t}}^{-}, \boldsymbol{\Sigma}_{z_{k,t}}^{-})$ in closed form, where

$$\boldsymbol{\mu}_{z_{k,t}}^{-} = \mathbf{A}\boldsymbol{\mu}_{z_{k,t-1}}^{-} + \mathbf{B}\boldsymbol{a}_{k,t-1} + \mathbf{C}\boldsymbol{\mu}_{l_k}^{-},$$

$$\Sigma_{k,t}^{-} = \mathbf{A}\boldsymbol{\Sigma}_{k,t-1}^{+}\mathbf{A}^{T} + \mathbf{C}\boldsymbol{\Sigma}_{l_k}^{-}\mathbf{C}^{T} + \mathbf{Q}. \tag{C.10}$$

C.2.2.2. Observation Update (fts-SSM)

In this stage, we compute the posterior belief $p(\boldsymbol{z}_{k,t}|\boldsymbol{w}_{1:t}^{k}, \boldsymbol{a}_{1:t}^{k}, \boldsymbol{\beta}_{1:k}, \boldsymbol{\alpha}_{1:k-1}) = \mathcal{N}(\boldsymbol{\mu}_{z_{k,t}}^{-}, \boldsymbol{\Sigma}_{z_{k,t}}^{-})$. using the same closed-form update as in Becker, Pandya, et al. 2019. The choice of the special observation model splits the state into two parts, an upper $\boldsymbol{z}_t^{\mathrm{u}}$ and a lower part $\boldsymbol{z}_t^{\mathrm{l}}$, resulting in the posterior belief $\mathcal{N}(\boldsymbol{\mu}_{z_{k,t}}^{-}, \boldsymbol{\Sigma}_{z_{k,t}}^{-}) = \mathcal{N}(\begin{bmatrix} \boldsymbol{\mu}_t^{u+} \\ \boldsymbol{\mu}_t^{l+} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_t^{u} & \boldsymbol{\Sigma}_t^{s} \\ \boldsymbol{\Sigma}_t^{s} & \boldsymbol{\Sigma}_t^{l} \end{bmatrix}^{+})$, with $\boldsymbol{\Sigma}_t^{u} = \mathrm{diag}(\boldsymbol{\sigma}_t^{s})$, $\boldsymbol{\Sigma}_t^{l} = \mathrm{diag}(\boldsymbol{\sigma}_t^{l})$ and $\boldsymbol{\Sigma}_t^{s} = \mathrm{diag}(\boldsymbol{\sigma}_t^{s})$. Thus, the factorization allows for only the diagonal and one off-diagonal vector of the covariance to be computed and simplifies the calculation of the mean and posterior to simple scalar operations.

The closed-form equations for the mean can be expressed as the following scalar equations,

$$\boldsymbol{z}_t^{+} = \boldsymbol{z}_t^{-} + \begin{bmatrix} \boldsymbol{\sigma}_t^{\mathrm{u},-} \\ \boldsymbol{\sigma}_t^{\mathrm{l},-} \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{w}_t - \boldsymbol{z}_t^{\mathrm{u},-} \\ \boldsymbol{w}_t - \boldsymbol{z}_t^{\mathrm{u},-} \end{bmatrix} \oslash \begin{bmatrix} \boldsymbol{\sigma}_t^{\mathrm{u},-} + \boldsymbol{\sigma}_t^{\mathrm{obs}} \\ \boldsymbol{\sigma}_t^{\mathrm{u},-} + \boldsymbol{\sigma}_t^{\mathrm{obs}} \end{bmatrix},$$

The corresponding equations for the variance update can be expressed as the following scalar operations, $\boldsymbol{\sigma}_t^{\mathrm{u},+} = \boldsymbol{\sigma}_t^{\mathrm{u},-} \odot \boldsymbol{\sigma}_t^{\mathrm{u},-} \oslash \left(\boldsymbol{\sigma}_t^{\mathrm{u},-} + \boldsymbol{\sigma}_t^{\mathrm{obs}}\right)$,
$\boldsymbol{\sigma}_t^{\mathrm{s},+} = \boldsymbol{\sigma}_t^{\mathrm{u},-} \odot \boldsymbol{\sigma}_t^{\mathrm{s},-} \oslash \left(\boldsymbol{\sigma}_t^{\mathrm{u},-} + \boldsymbol{\sigma}_t^{\mathrm{obs}}\right)$,
$\boldsymbol{\sigma}_t^{\mathrm{l},+} = \boldsymbol{\sigma}_t^{\mathrm{l},-} - \boldsymbol{\sigma}_t^{\mathrm{s},-} \odot \boldsymbol{\sigma}_t^{\mathrm{s},-} \oslash \left(\boldsymbol{\sigma}_t^{\mathrm{u},-} + \boldsymbol{\sigma}_t^{\mathrm{obs}}\right)$,, where $\odot$ denotes the elementwise vector product and $\oslash$ denotes an elementwise vector division.

C.2.3. Modelling Assumptions

C.2.3.1. Control Model

To achieve action conditioning within the recurrent cell of fts-SMM, we include a control model $b(a_{k,t})$ in addition to the linear transition model A_t. $b(a_{k,t}) = f(a_{k,t})$, where $f(.)$ can be any non-linear function approximator. We use a multi-layer neural network regressor with ReLU activations Shaj, Becker, et al. 2020.

However, unlike the fts-SSM where actions are assumed to be known and subjected to no noise, in the sts-SSM, the abstract action is an inferred latent variable with an associated uncertainty estimate. Hence we use a linear control model Y, for principled uncertainty propagation.

C.2.3.2. Transition Noise

We assume the covariance of the transition noise Q and S in both timescales to be diagonal. The noise is learned and is independent of the latent state.

C.3. Metrics Used For Measuring Long Horizon Predictions

C.3.1. Sliding Window RMSE

The sliding window RMSE (Root Mean Squared Error) metric is computed for a predicted trajectory in comparison to its ground truth. At each time step, the RMSE for each trajectory is determined by taking the root mean square of the differences between the ground truth and predicted values within a sliding window that terminates at the current time step. This sliding window, with a specified size, provides a smoothed localized assessment of prediction accuracy over the entire prediction length. Mathematically, the sliding window RMSE at time step t is given by:

$$RMSE(t) = \sqrt{\frac{1}{W} \sum_{i=t-W+1}^{t} (\mathrm{gt}_i - pred_i)^2}$$

where t is the current time step, W is the window size, and gt_i and $pred_i$ are the ground truth and predicted values at time step i, respectively. The extension to multiple trajectories is straightforward and omitted to keep the notation uncluttered.

C.3.2. Sliding Window NLL

The sliding window NLL (Negative Log-Likelihood) metric is computed for a predicted probability distribution against the true distribution. At each time step, the NLL is determined by summing the negative log-likelihood values within a sliding window that terminates at the current time step. This sliding window, with a specified size, provides a smoothed localized evaluation of prediction accuracy across the entire sequence.

Mathematically, the sliding window NLL at time step t is given by:

$$\text{NLL}(t) = -\frac{1}{W} \sum_{i=t-W+1}^{t} \log \mathcal{N}\left(\text{gt}_i \mid \text{predMean}_i, \text{predVar}_i\right)$$

where t is the current time step, W is the window size. predMean_i, predVar_i, and gt_i represent the predicted mean, predicted variance, and the ground truth at time step i.

C.4. Visualization of predictions given by different models.

In this section, we plot the multistep ahead predictions (mean and variance) by different models on 3 datasets on normalized test trajectories. Not that we omit NaN values in predictions while plotting.

C.4.0.1. Franka Kitchen

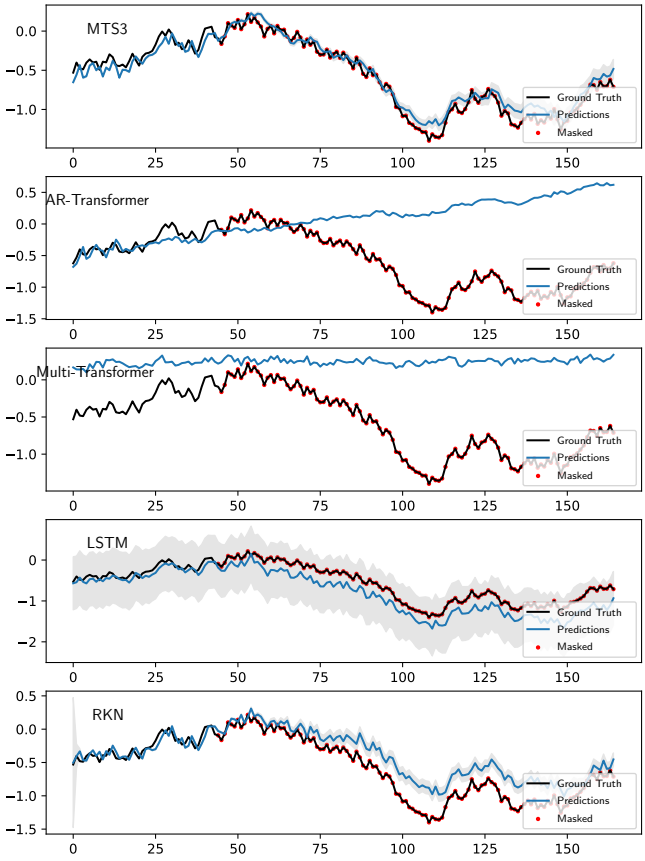


Figure C.3.: Multi-step ahead mean and variance predictions for a particular joint (joint 1) of Franka Kitchen Environment. The multi-step ahead prediction starts from the first red dot, which indicates masked observations. MTS3 gives the most reliable mean and variance estimates.

C.4.0.2. Hydraulic Excavator

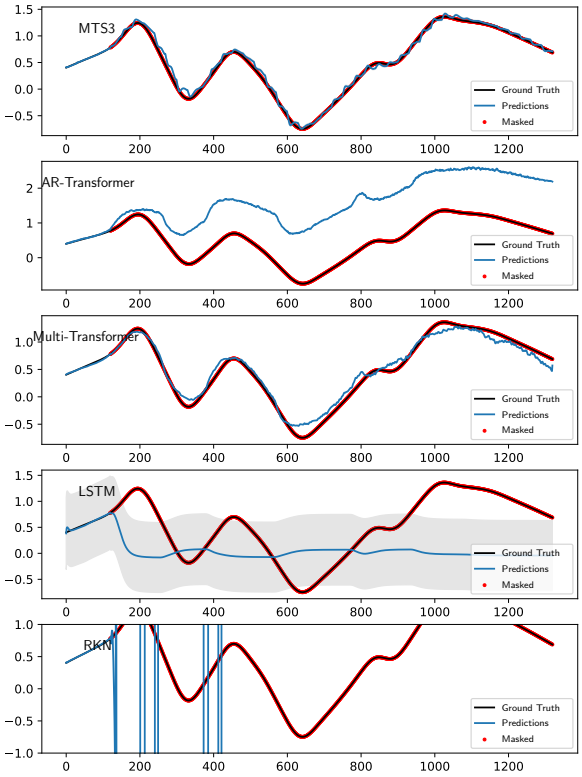


Figure C.4.: Multi-step ahead mean and variance predictions for a particular joint (joint 1) of Excavator Dataset. The multi-step ahead prediction starts from the first red dot, which indicates masked observations. MTS3 gives the most reliable mean and variance estimates even up to 12 seconds into the future. Another interesting observation can also be seen in the predictions for MTS3, where after every window k of sts-SSM, which is 0.3 seconds (30 timesteps) long, the updation of the higher-level abstractions helps in grounding the lower-level predictions thus helping in the long horizon yet fine-grained predictions.

C.4.0.3. Mobile Robot

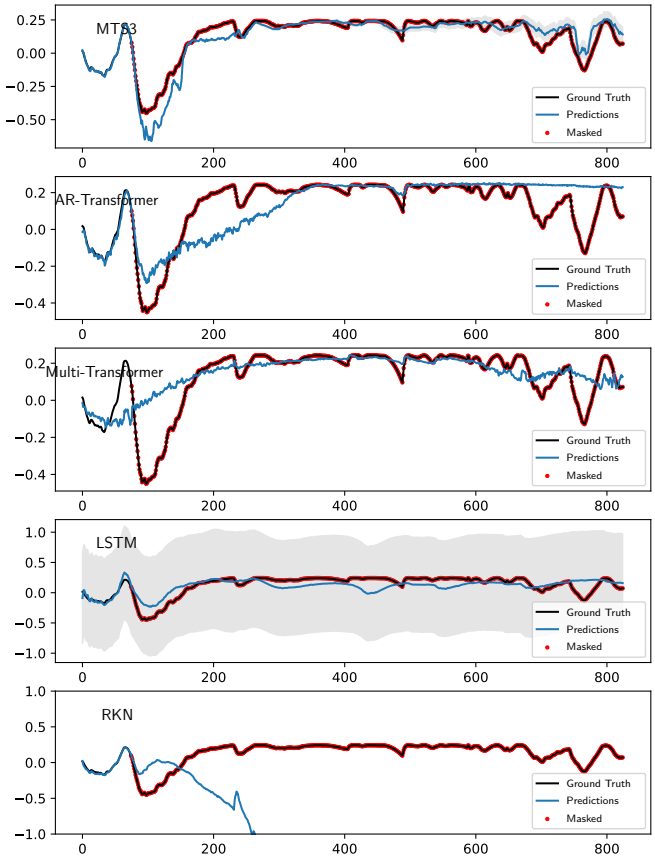


Figure C.5.: Multi-step ahead mean and variance predictions for a particular joint (joint 7) of Mobile Robot Dataset. The multi-step ahead prediction starts from the first red dot, which indicates masked observations. MTS3 gives the most accurate mean and variance estimates among all algorithms.

D. Appendix: Robots and Dataset Details

D.1. Robots and Data Used In Chapter 4 on Ac-SSM

The experiments are performed on data from four different robots. The details of robots, data, and data preprocessing are explained below:

D.1.1. Hydraulic Brokk 40 Robot Arm

Observation and Data Set: The data was obtained from a HYDROLEK–7W 6 degree–of–freedom manipulator with a continuous (360 degree) jaw rotation mechanism. We actuate the joints via hydraulic pistons, which are powered via an auxiliary output from the hydraulic pump. Thus learning the forward model is difficult due to inherent hysteresis associated with hydraulic control. For this robot, only one joint is moved at a time, so we have independent time series per joint. The joint data consists of measured joint positions and the input current to the controller of the joint sampled at 100Hz.

Training Procedure: During training, we work with sequences of length 500. For the first 300 time steps those sequences consist of the full observation, i.e., the joint position and current. We give only the current signals in the remaining 200 time steps. The models have to impute the missing joint positions in an uninformed fashion, i.e., we only indicate the absence of a position by unrealistically high values.

Musculoskeletal Robot Arm

Observation and Data Set: For this soft robot we have 4 dimensional observation inputs(joint angles) and 8 dimensional action inputs(pressures). We collected the data of a four DoF robot actuated by Pneumatic Artificial Muscles (PAMs). The robot arm has eight PAMs in total with each DoF actuated by an antagonistic pair. The robot arm reaches high joint angle accelerations of up to 28, 000deg/s2 while avoiding dangerous joint limits thanks to the antagonistic actuation and limits on the air pressure ranges. The data consists of trajectories collected while training with a model-free reinforcement learning algorithm to hit balls while playing table tennis. We sampled the data at 100Hz. The hysteresis associated with the pneumatic actuators used in this robot is challenging

to model and is relevant to the soft robotics in general.

Training Procedure: During training, we randomly removed three-quarters of the states from the sequences and tasked the models with imputing those missing states, only based on the knowledge of available actions/control commands, i.e., we train the models to perform action conditional future predictions to impute missing states. The imputation employs the model for multi-step ahead predictions in a convenient way. One could instead go for a dedicated loss function as in approaches like Finn, Goodfellow, and Levine 2016, Oh et al. 2015 for long term predictions.

Franka Emika Panda Robot Arm

Observation and Data Set: We collected the data from a 7 DoF Franka Emika Panda manipulator during free motion. We chose this task since the robot exhibits different dynamics behaviour due to electric actuators and high frequencies(1kHz). The raw joint positions, velocities and torques were recorded using Franka Interfaces while the joint accelerations were computed by finite differences on filtered velocity data (obtained using a zero-phase 8th-order digital Butterworth filter with a cut-off frequency of 5Hz). The observations for the forward model consist of the seven joint angles in radians, and the corresponding actions were joint Torques in Nm. While the inverse model use both joint angles and velocities as observations. The data was divided into train and test sets in the ratio 4:1. We divide the data into sequences of length 300 while training the recurrent models for forward dynamics and use sequences of length 50 for inverse dynamics.

Training Procedure Forward Dynamics: Similar to the multi-step ahead training procedure in D.1.1, during training we randomly removed three-quarters of the observations(joint angles) from the sequences and tasked the models with imputing those missing observations, only based on the knowledge of available actions/control commands.

Training Procedure Inverse Dynamics: The recurrent models (LSTM, ac-RKN) uses a similar architecture, as shown in Figure 3 of the main paper, except for the recurrent module. The hyperparameters including learning rate, latent state and observation dimensions, learning rate, control model architecture, action decoder architecture and regularization parameter for the joint forward-inverse dynamics loss function are searched via GpyOptauthors 2016 and is mentioned in Appendix D. The observation encoder and decoder architecture is chosen to be of the same size across the models being compared. For all models, we use the joint positions and velocities as the observation input and differences to the next state as desired observation. The FFNN gets the current observation and desired observation as input and is tasked to predict the joint Torques directly(unlike differences in recurrent models) as in previous regression approachesNguyen-Tuong and Peters 2010.

Barrett WAM Robot Arm

Observation and Data Set: The Barett task is based on a publicly available dataset comprising joint positions, velocities, acceleration and torques of a seven degrees-of-freedom real Barett WAM robot. The original training dataset (12, 000 data points) is split into sequences of length 98. Twenty-four out of the total 119 episodes are utilized for testing, whereas the other 95 are used for training. The direct cable drives which drive this robot produce high torques, generating fast and dexterous movements but yield complex dynamics. Rigid-body dynamics cannot be model this complex dynamics due to the variable stiffness and lengths of the cables.

Training Procedure Inverse Dynamics: The training procedure is repeated as in D.1.1

D.2. Robots and Datasets Used In Chapter 5 on HiP-SSM

D.2.1. Franka Emika Panda Robot Arm

Observation and Data Set: We collected the data from a 7 DoF Franka Emika Panda manipulator during free motion and while manipulating loads with weights 0kg (free motion), 0.5 kg, 1 kg, 1.5 kg, 2 kg and 2.5 kg. Data is sampled at high frequencies (1kHz). The training trajectories were motions with loads 0kg(free motion), 1kg, 1.5kg, 2.5 kgs, while the testing trajectories contained motions with loads of 0.5kg and 2 kgs. The observations for the forward model consist of the seven joint angles in radians, and the corresponding actions were joint Torques in Nm. We divide the data into sequences of length 600 while training the recurrent models for forward dynamics, with 300 time-steps (corresponding to 300 milli-seconds) used as context set and rest 300 is used for the recurrent time-series modelling.

Training Procedure: For the fully observable case, we trained HiP-RSSM for one-step ahead prediction using an RMSE loss. Similar to the training procedure for partially observable case as in D.1.1, during training we randomly removed half of the observations(joint angles) from the sequences and tasked the models with imputing those missing observations, only based on the knowledge of available actions/control commands. Other recurrent baselines (RKN, LSTM, GRU) are trained in a similar fashion except that, we dont maintain a context set of interaction histories during training/inference.

D.2.2. Wheeled Mobile Robot

Observation and Data Set: We collected 50 random trajectories from a Pybullet simulator a wheeled mobile robot traversing terrain with sinusoidal slopes. Data is sampled at high frequencies (500Hz). 40 out of the 50 trajectories were used for training and the rest 10 for testing. The observations consists of parameters which completely describe its location and orientation of the robot. The observation of the robot at any time instance t consists of the following features:.

$$o_t = [x, y, z, \cos(\alpha), \sin(\alpha), \cos(\beta) \\ \sin(\beta), \cos(\gamma), \sin(\gamma)]$$

where, x, y, z - denote the global position of the Center of Mass of he robot, $\alpha, \beta, \gamma-$ Roll, pitch and yaw angles of the robot respectively, in the global frame of reference (Sonker and Dutta 2020). We divide the data into sequences of length 300 while training the recurrent models for forward dynamics, with 150 time-steps (corresponding to 300 milli-seconds) used as context set and rest 150 is used for the recurrent time-series modelling.

Training Procedure: For the fully observable case, we trained HiP-RSSM for one-step ahead prediction using an RMSE loss. Similar to the training procedure for partially observable case as in D.1.1, during training we randomly removed half of the observations from the sequences and tasked the models with imputing those missing observations, only based on the knowledge of available actions/control commands.
Other recurrent baselines (RKN, LSTM, GRU) are trained in a similar fashion except that, we dont maintain a context set of interaction histories during training/inference.

D.3. Robots and Data used in Chapter 6 on Multi Time Scale SSM

In all datasets, we only use information about agent/object positions and we mask out velocities to create a partially observable setting. All datasets are subjected to a mean zero, unit variance normalization during training. During testing, they are denormalized after predictions. The details of the different datasets used are explained below:

D.3.1. D4RL Datasets

Details: We use a set of 3 different environments/agents from D4RL dataset Fu, Kumar, et al. 2020, which includes the HalfCheetah, Franka Kitchen and Maze2D (medium) environment. **(a) HalfCheetah:** We used 1000 suboptimal trajectories collected from a policy trained to approximately 1/3 the performance of the expert. The observation space consists of 8 joint positions and the action space consists of 6 joint torques collected

at 50 Hz frequency. 800 trajectories were used for training and 200 for testing. For the long horizon task, we used 1.2 seconds (60 timesteps) as context and tasked the model to predict 6 seconds (300 timesteps) into the future. **(b) Franka Kitchen:** The goal of the Franka Kitchen environment is to interact with the various objects to reach a desired state configuration. The objects you can interact with include the position of the kettle, flipping the light switch, opening and closing the microwave and cabinet doors, or sliding the other cabinet door. We used the "complete" version of the dataset and collected 1000 trajectories where all four tasks are performed in order. The observation space consists of 30 dimensions (9 joint positions of the robot and 21 object positions). The action space consists of 9 joint velocities clipped between -1 and 1 rad/s. The data was collected at a 50 Hz frequency. 800 trajectories were used for training and 200 for testing. For the long horizon task, we used 0.6 seconds (30 timesteps) as context and tasked the model to predict 2.7 seconds (135 timesteps) into the future. The dataset is complex due to multi-task, multi-object interactions in a single trajectory. **(c) Medium Maze:** We used 20000 trajectories from a 2D Maze environment, where each trajectory consists of a force-actuated ball (along the X and Y axis) moving to a fixed target location. The observation consists of as the (x, y) locations and a 2D action space. The data is collected at 100 Hz frequency. 16000 trajectories were used for training and 4000 for testing. For the long horizon task, we used 0.6 seconds (60 timesteps) as context and tasked the model to predict 3.9 seconds (390 timesteps) into the future. Rendering of the three environments is shown in Figure 6.11.

D.3.2. Hydraulic Excavator

Details: We collected the data from a wheeled excavator JCB Hydradig 110W show in Figure 6.11. The data was collected by actuating the boom and arm of the excavator using Multisine and Amplitude-Modulated Pseudo-Random Binary Sequence (APRBS) joystick signals with safety mechanisms in place. A total of 150 mins of data was collected at a frequency of 100 Hz. of which was used as a training dataset and the rest as testing. The observation space consists of the boom and arm positions, while the joystick signals are chosen as actions. For the long horizon task we used 1.5 seconds (150 timesteps) as context and tasked the model to predict 12 seconds (1200 timesteps) into the future.

D.3.3. Panda Robot With Varying Payloads

Details: We collected the data from a 7 DoF Franka Emika Panda manipulator during free motion and while manipulating loads with weights 0kg (free motion), 0.5 kg, 1 kg, 1.5 kg, 2 kg and 2.5 kg. The robot used is shown in Figure 6.11. Data is sampled at a frequency of 100 Hz. The training trajectories were motions with loads of 0kg(free motion), 1kg, 1.5kg, and 2.5 kgs, while the testing trajectories contained motions with loads of 0.5 kg and 2 kg. The observations for the forward model consist of the seven

joint angles in radians, and the corresponding actions were joint Torques in Nm. For the long horizon task we used 0.6 seconds (60 timesteps) as context and tasked the model to predict 1.8 seconds (180 timesteps) into the future.

D.3.4. Wheeled Mobile Robot

Observation and Data Set: We collected 50 random trajectories from a Pybullet simulator a wheeled mobile robot traversing terrain with slopes generated by a mix of sine waves as opposed to the sine wave terrain for experiments used in Chapter 5, making this more challenging. Data is sampled at high frequencies (500Hz). 40 out of the 50 trajectories were used for training and the rest 10 for testing. The observations consist of parameters which completely describe the location and orientation of the robot. The observation of the robot at any time instance t consists of the following features:

$$o_t = [x, y, z, \cos(\alpha), \sin(\alpha), \cos(\beta) \\ \sin(\beta), \cos(\gamma), \sin(\gamma)]$$

where, x, y, z - denote the global position of the Center of Mass of the robot, $\alpha, \beta, \gamma-$ Roll, pitch and yaw angles of the robot respectively, in the global frame of reference (Sonker and Dutta 2020). For the long horizon task we used 0.6 seconds (150 timesteps) as context and tasked the model to predict 3 seconds (750 timesteps) into the future.

E. Appendix: Hyperparameters

E.1. Hyperparameters: Chapter 4

Pneumatic Musculoskeltal Robot Arm

Table E.1.: Forward Dynamics Hyperparameters For Pneumatic Musculoskeletal Robot.

Hyperparameter	**ac-RKN**	**RKN**	**LSTM**
Learning Rate	3.1e-3	1.9e-3	6.6e-3
Latent Observation Dimension	60	60	60
Latent State Dimension	120	120	120

Encoder (ac-RKN,RKN,LSTM): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 120, ReLU

Observation Decoder (ac-RKN,RKN,LSTM): 1 fully connected + linear output:

- Fully Connected 1: 120, ReLU

Transition Model (ac-RKN,RKN): bandwidth: 3, number of basis: 15

- $\alpha(\vec{z}_t)$: No hidden layers - softmax output

Control Model (ac-RKN): 3 fully connected + linear output

- Fully Connected 1: 120, ReLU
- Fully Connected 2: 120, ReLU
- Fully Connected 3: 120, ReLU

Architecture For FFNN Baseline 2 fully connected + linear output

- Fully Connected 1: 6000, ReLU
- Fully Connected 2: 3000, ReLU

Dropout Regularization - 0.512
Learning Rate - 1.39e-2
Optimizer Used: Adam Optimizer

Hydraulic Brokk-40 Robot Arm

Table E.2.: Forward Dynamics Hyperparameters For Pneumatic Musculoskeletal Robot.

Hyperparameter	**ac-RKN**	**RKN**	**LSTM**	**GRU**
Learning Rate	5e-4	5e-4	9.1e-4	2.1e-3
Latent Observation Dimension	30	30	30	30
Latent State Dimension	60	60	60	60

Encoder (ac-RKN,RKN,LSTM,GRU): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 30, ReLU

Observation Decoder (ac-RKN,RKN,LSTM,GRU): 1 fully connected + linear output:

- Fully Connected 1: 30, ReLU

Transition Model (ac-RKN,RKN): bandwidth: 3, number of basis: 32

- $\alpha(\vec{z}_t)$: No hidden layers - softmax output

Control Model (ac-RKN): 1 fully connected + linear output

- Fully Connected 1: 120, ReLU

Franka Emika Panda - Forward Dynamics Learning

Table E.3.: Forward Dynamics Learning Hyperparameters For Panda.

Hyperparameter	**ac-RKN**	**RKN**	**LSTM**	**GRU**
Learning Rate	3.1e-3	1.7e-3	6.6e-3	8.72e-3
Latent Observation Dimension	45	30	30	45
Latent State Dimension	90	60	60	90

Encoder (ac-RKN,RKN,LSTM,GRU): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 120, ReLU

Observation Decoder (ac-RKN,RKN,LSTM,GRU): 1 fully connected + linear output:

- Fully Connected 1: 240, ReLU

Transition Model (ac-RKN,RKN): bandwidth: 3, number of basis: 15

- $\alpha(\vec{z}_t)$: No hidden layers - softmax output

Control Model (ac-RKN): 3 fully connected + linear output

- Fully Connected 1: 30, ReLU

- Fully Connected 2: 30, ReLU
- Fully Connected 3: 30, ReLU

Architecture For FFNN Baseline - Forward Dynamics 3 fully connected + linear output

- Fully Connected 1: 1000, ReLU
- Fully Connected 2: 1000, ReLU
- Fully Connected 3: 1000, ReLU

Dropout Regularization - 0.1147
Learning Rate - 8.39e-3
Optimizer Used: SGD Optimizer

Franka Emika Panda - Inverse Dynamics Learning

Table E.4.: Inverse Dynamics Learning Hyperparameters For Panda.

Hyperparameter	**ac-RKN**	**RKN (No Action Feedback)**	**LSTM**
Learning Rate	7.62e-3	3.5e-3	9.89e-3
Latent Observation Dimension	15	30	30
Latent State Dimension	30	60	60
Regularization Factor (λ)	0.158	0.179	0.196

Encoder (ac-RKN,RKN,LSTM): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 120, ReLU

Observation Decoder (ac-RKN,RKN,LSTM): 1 fully connected + linear output:

- Fully Connected 1: 240, ReLU

Action Decoder (ac-RKN,RKN,LSTM): 1 fully connected + linear output:

- Fully Connected 1: 512, ReLU

Transition Model (ac-RKN,RKN): bandwidth: 3, number of basis: 15

- $\alpha(\vec{z}_t)$: No hidden layers - softmax output

Control Model (ac-RKN): 1 fully connected + linear output

- Fully Connected 1: 45, ReLU

Architecture For FFNN Baseline - Inverse Dynamics 3 fully connected + linear output

- Fully Connected 1: 500, ReLU
- Fully Connected 2: 500, ReLU

- Fully Connected 3: 500, ReLU

Dropout Regularization - 0.563
Learning Rate - 1.39e-2
Optimizer Used: SGD Optimizer

Barrett WAM - Inverse Dynamics Learning

Table E.5.: Inverse Dynamics Learning Hyperparameters Barrett WAM.

Hyperparameter	**ac-RKN**	**RKN (No Action Feedback)**	**LSTM**
Learning Rate	7.7e-3	1.7e-3	9.33e-3
Latent Observation Dimension	15	30	45
Latent State Dimension	30	60	90
Regularization Factor (λ)	0.176	0	3.42e-3

Encoder (ac-RKN,RKN,LSTM): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 120, ReLU

Observation Decoder (ac-RKN,RKN,LSTM): 1 fully connected + linear output:

- Fully Connected 1: 240, ReLU

Action Decoder (ac-RKN): 2 fully connected + linear output:

- Fully Connected 1: 256, ReLU
- Fully Connected 1: 256, ReLU

Action Decoder (RKN,LSTM): 1 fully connected + linear output:

- Fully Connected 1: 512, ReLU

Transition Model (ac-RKN,RKN): bandwidth: 3, number of basis: 15

- $\alpha(\vec{z}_t)$: No hidden layers - softmax output

Control Model (ac-RKN): 1 fully connected + linear output

- Fully Connected 1: 45, ReLU

Architecture For FFNN Baseline 3 fully connected + linear output

- Fully Connected 1: 500, ReLU
- Fully Connected 2: 500, ReLU
- Fully Connected 3: 500, ReLU

Dropout Regularization - 0.563
Learning Rate - 1e-5
Optimizer Used: SGD Optimizer

E.2. Hyperparameters: Chapter 5

E.2.1. Pneumatic Musculoskeletal Robot Arm

Recurrent Models

Table E.6.: Forward Dynamics Learning Hyperparameters For Panda.

Hyperparameter	**HiP-RSSM**	**RKN**	**LSTM**	**GRU**
Learning Rate	8e-4	8e-4	1e-3	1e-3
Latent Observation Dimension	15	15	15	15
Latent State Dimension	30	30	75	75
Latent Task Dimension	30	-	-	-

Context Encoder (HiP-RSSM): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 240, ReLU

Observation Encoder (HiP-RSSM,RKN,LSTM,GRU): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 120, ReLU

Observation Decoder (HiP-RSSM,RKN,LSTM): 1 fully connected + linear output:

- Fully Connected 1: 120, ReLU

Transition Model (HiP-RSSM,RKN): number of basis: 15

- $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Control Model (HiP-RSSM,RKN): 3 fully connected + linear output

- Fully Connected 1: 120, ReLU
- Fully Connected 2: 120, ReLU
- Fully Connected 3: 120, ReLU

E.2.2. Franka Robot Arm With Varying Loads

Recurrent Models

Encoder (HiP-RSSM,RKN,LSTM,GRU): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 30, ReLU

Observation Decoder (HiP-RSSM,RKN,LSTM,GRU): 1 fully connected + linear output:

Table E.7.: Forward Dynamics Learning Hyperparameters For Franka.

Hyperparameter	**HiP-RSSM**	**RKN**	**LSTM**	**GRU**
Learning Rate	1e-3	1e-3	3e-3	3e-3
Latent Observation Dimension	15	15	15	15
Latent State Dimension	30	30	75	75
Latent Task Dimension	30	-	-	-

- Fully Connected 1: 30, ReLU

Transition Model (HiP-RSSM,RKN): number of basis: 32

- $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Control Model (HiP-RSSM, RKN): 1 fully connected + linear output

- Fully Connected 1: 120, ReLU

E.2.3. Wheeled Robot Traversing Slopes Of Different Height

Recurrent Models

Table E.8.: Forward Dynamics Learning Hyperparameters For Wheeled Robot.

Hyperparameter	**HiP-RSSM**	**RKN**	**LSTM**	**GRU**
Learning Rate	9e-4	9e-4	1e-2	1e-2
Latent Observation Dimension	30	30	15	15
Latent State Dimension	60	60	75	75
Latent Task Dimension	60	-	-	-

Encoder (HiP-RSSM,RKN,LSTM,GRU): 1 fully connected + linear output (elu + 1)

- Fully Connected 1: 120, ReLU

Observation Decoder (HiP-RSSM,RKN,LSTM,GRU): 1 fully connected + linear output:

- Fully Connected 1: 240, ReLU

Transition Model (HiP-RSSM,RKN): number of basis: 15

- $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Control Model (HiP-RSSM, RKN): 3 fully connected + linear output

- Fully Connected 1: 120, ReLU

E.3. Hyperparameters: Chapter 6

Compute Resources For training MTS3, LSTM, GRU and Transformer models we used compute nodes with (i) Nvidia 3090 and (ii) Nvidia 2080 RTX GPUs. For training more computationally expensive locally linear models like RKN, HiP-RSSM we used compute nodes with NVIDIA A100-40 GPUs.

Hyperparameters Hyperparameters were selected via grid search. In general, the performance of MTS3 is not very sensitive to hyperparameters. Among all the baselines, Transformer models were most sensitive to hyperparameters (see Appendix E.5 for details of Transformer architecture).

Discretization Step: For MTS3, the discretization step for the slow time scale SSM as discussed in Section 3.1 for all datasets was fixed as $\mathbf{H} \cdot \Delta t = 0.3$ seconds. In our experiments, we found that discretization values between $0.2 \leq \mathbf{H} \cdot \Delta t \leq 0.5$ seconds give similar performance.

Rule Of thumb for choosing discretization step in MTS3: For any N-level MTS3 as defined in Section 3.4, we recommend searching for discretization factor H_i as a hyperparameter. However, as a general rule of thumb, it can be chosen as $H_i = (\sqrt[N]{T})^i$, where T is the maximum prediction horizon required / episode length. This ensures that very long recurrences are divided between smaller equal-length task-reconfigurable local SSM windows (of length $\sqrt[N]{T}$) spread across several hierarchies.

Encoder Decoder Architecture: For all recurrent models (MTS3, HiP-RSSM, RKN, LSTM and GRU) we use a similar encoder-decoder architecture across datasets. Small variations from these encoder-decoder architecture hyperparameters can still lead to similar prediction performance as reported in the paper.

Observation Set Encoder (MTS3): 1 fully connected + linear output:

- Fully Connected 1: 240, ReLU

Action Set Encoder (MTS3): 1 fully connected + linear output:

- Fully Connected 1: 240, ReLU

Observation Encoder (MTS3, HiP-RSSM, RKN, LSTM, GRU): 1 fully connected + linear output:

- Fully Connected 1: 120, ReLU

Observation Decoder (MTS3, HiP-RSSM, RKN, LSTM, GRU): 1 fully connected + linear output:

- Fully Connected 1: 120, ReLU

Control Model (Primitive Action Encoder) (MTS3, HiP-RSSM, RKN): 1 fully connected + linear output:

- Fully Connected 1: 120, ReLU

The rest of the hyperparameters are described below:

E.3.1. D4RL Datasets

E.3.1.1. Half Cheetah

Recurrent Models

Hyperparameters	MTS3	HiP-RSSM	RKN	LSTM	GRU
Learning Rate	3e-3	1e-3	1e-3	1e-3	1e-3
Latent Observation Dimension	15	15	15	15	15
Observation Set Latent Dimension (sts-SSM)	15	-	-	-	-
Latent State Dimension	30	30	30	45	45
Latent Task Dimension	30	30	-	-	-
Latent Abstract Action Dimension (sts-SSM)	30	-	-	-	-

Transition Model (HiP-RSSM, RKN): number of basis: 32

- $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Autoregressive Transformer Baseline
Learning Rate: 1e-5
Optimizer Used: Adam Optimizer
Embedding size: 96
Number of Decoder Layers: 4
Number Of Attention Heads: 4

Multistep Transformer Baseline
Learning Rate: 1e-5
Optimizer Used: Adam Optimizer
Embedding size: 128
Number Of Encoder Layers: 2
Number of Decoder Layers: 1
Number Of Attention Heads: 4

Hyperparameters	MTS3	HiP-RSSM	RKN	LSTM	GRU
Learning Rate	3e-3	9e-4	9e-4	1e-3	1e-3
Latent Observation Dimension	30	30	30	30	30
Observation Set Latent Dimension (sts-SSM)	30	-	-	-	-
Latent State Dimension	60	60	60	90	90
Latent Task Dimension	60	60	-	-	-
Latent Abstract Action Dimension (sts-SSM)	60	-	-	-	-

E.3.1.2. Franka Kitchen

Recurrent Models

Transition Model (HiP-RSSM, RKN): number of basis: 15

- $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Autoregressive Transformer Baseline
Learning Rate: 5e-5
Optimizer Used: Adam Optimizer
Embedding size: 64
Number of Decoder Layers: 4
Number Of Attention Heads: 4

Multistep Transformer Baseline
Learning Rate: 1e-5
Optimizer Used: Adam Optimizer
Embedding size: 64
Number Of Encoder Layers: 2
Number of Decoder Layers: 1
Number Of Attention Heads: 4

E.3.1.3. Maze 2D

Recurrent Models

Hyperparameters	MTS3	HiP-RSSM	RKN	LSTM	GRU
Learning Rate	3e-3	9e-4	9e-4	1e-3	1e-3
Latent Observation Dimension	30	30	30	30	30
Observation Set Latent Dimension (sts-SSM)	30	-	-	-	-
Latent State Dimension	60	60	60	90	90
Latent Task Dimension	60	60	-	-	-
Latent Abstract Action Dimension (sts-SSM)	60	-	-	-	-

Transition Model (HiP-RSSM, RKN): number of basis: 15

- $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Autoregressive Transformer Baseline
Learning Rate: 5e-5
Optimizer Used: Adam Optimizer
Embedding size: 96
Number of Decoder Layers: 4
Number Of Attention Heads: 4

Multistep Transformer Baseline
Learning Rate: 1e-5
Optimizer Used: Adam Optimizer
Embedding size: 128
Number Of Encoder Layers: 2
Number of Decoder Layers: 1
Number Of Attention Heads: 4

E.3.2. Franka Robot Arm With Varying Loads

Recurrent Models

Hyperparameters	MTS3	HiP-RSSM	RKN	LSTM	GRU
Learning Rate	3e-3	9e-4	9e-4	3e-3	3e-3
Latent Observation Dimension	15	15	15	15	15
Observation Set Latent Dimension (sts-SSM)	15	-	-	-	-
Latent State Dimension	30	30	30	45	45
Latent Task Dimension	30	30	-	-	-
Latent Abstract Action Dimension (sts-SSM)	30	-	-	-	-

Transition Model (HiP-RSSM,RKN): number of basis: 32

- $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Autoregressive Transformer Baseline
Learning Rate: 5e-5
Optimizer Used: Adam Optimizer
Embedding size: 64
Number of Decoder Layers: 4
Number Of Attention Heads: 4

Multistep Transformer Baseline
Learning Rate: 2e-5
Optimizer Used: Adam Optimizer
Embedding size: 64
Number Of Encoder Layers: 2
Number of Decoder Layers: 1
Number Of Attention Heads: 4

E.3.3. Hydraulic Excavator

Hyperparameters	MTS3	HiP-RSSM	RKN	LSTM	GRU
Learning Rate	3e-3	8e-4	8e-4	1e-3	1e-3
Latent Observation Dimension	15	15	15	15	15
Observation Set Latent Dimension (sts-SSM)	15	-	-	-	-
Latent State Dimension	30	30	30	45	45
Latent Task Dimension	30	30	-	-	-
Latent Abstract Action Dimension (sts-SSM)	30	-	-	-	-

Transition Model (HiP-RSSM,RKN): number of basis: 15

- coefficient net $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Autoregressive Transformer Baseline
Learning Rate: 1e-5
Optimizer Used: Adam Optimizer
Embedding size: 96
Number of Decoder Layers: 4
Number Of Attention Heads: 4

Multistep Transformer Baseline
Learning Rate: 5e-5
Optimizer Used: Adam Optimizer
Embedding size: 64
Number Of Encoder Layers: 2
Number of Decoder Layers: 1
Number Of Attention Heads: 4

E.3.4. Wheeled Robot Traversing Uneven Terrain

Hyperparameters	MTS3	HiP-RSSM	RKN	LSTM	GRU
Learning Rate	3e-3	8e-4	8e-4	1e-3	1e-3
Latent Observation Dimension	30	30	30	30	30
Observation Set Latent Dimension (sts-SSM)	30	-	-	-	-
Latent State Dimension	60	60	60	90	90
Latent Task Dimension	60	60	-	-	-
Latent Abstract Action Dimension (sts-SSM)	60	-	-	-	-

Transition Model (HiP-RSSM,RKN): number of basis: 15

- coefficient net $\alpha(\boldsymbol{z}_t)$: No hidden layers - softmax output

Autoregressive Transformer Baseline
Learning Rate: 5e-5
Optimizer Used: Adam Optimizer
Embedding size: 128

Number of Decoder Layers: 4
Number Of Attention Heads: 4

Multistep Transformer Baseline
Learning Rate: 5e-5
Optimizer Used: Adam Optimizer
Embedding size: 64
Number Of Encoder Layers: 4
Number of Decoder Layers: 2
Number Of Attention Heads: 4

E.3.5. Transformer Architecture Details

For the AR-Transformer Baseline, we use a GPT-like autoregressive version of transformers except that for the autoregressive input we also concatenate the actions to make action conditional predictions.

For Multi-Transformer we use the same direct multistep prediction and loss as in recent Transformer time-series forecasting literature Zhou et al. 2021; Liu et al. 2022; Nie et al. 2023; Zeng et al. 2022. A description of the action conditional direct multi-step version of the transformer is given in Algorithm 3.

Require: Input past observations $\mathbf{o_{inp}} \in \mathbb{R}^{S\times C}$; Input Past Actions $\mathbf{a_{inp}} \in \mathbb{R}^{S\times A}$; Future Actions $\mathbf{a_{pred}} \in \mathbb{R}^{O\times A}$;Input Length S; Predict length O; Observation Dimension C; Action Dimension A; Feature dimension d_k; Encoder layers number N; Decoder layers number M.

$\mathbf{o_{inp}} \in \mathbb{R}^{S\times C}, \mathbf{a}_{inp} \in \mathbb{R}^{S\times A}, \mathbf{a}_{pred} \in \mathbb{R}^{O\times A}$

$\mathbf{X}_{inp} = \text{ConCatFeatureWise}\left(\mathbf{o}_{inp}, \mathbf{a}_{inp}\right)$ ▷$\mathbf{X}_{inp} \in \mathbb{R}^{S\times(C+A)}$

$\mathbf{X}_{pred} = \text{ConCatFeatureWise}\left(Zeros(O, C), \mathbf{a}_{pred}\right)$ ▷$\mathbf{X}_{pred} \in \mathbb{R}^{O\times(C+A)}$

$\mathbf{X}_{\text{enc}}, \mathbf{X}_{\text{dec}} = \mathbf{X}_{\text{inp}}, ConCat\left(\mathbf{X}_{inp}, \mathbf{X}_{pred}\right)$ ▷ $\mathbf{X}_{\text{enc}} \in \mathbb{R}^{S\times(C+A)}, \mathbf{X}_{\text{dec}} \in \mathbb{R}^{(S+O)\times(C+A)}$

$\mathbf{X}^0_{\text{enc}} = Embed\left(\mathrm{X}_{\text{enc}}\right)$ ▷$\mathbf{X}^0_{\text{enc}} \in \mathbb{R}^{S\times d_k}$

for l *in* $\{1, \cdots, N\}$ **do**

  $\mathbf{X}^{l-1}_{\text{enc}} = LayerNorm\left(\mathbf{X}^{l-1}_{\text{enc}} + Attn\left(\mathbf{X}^{l-1}_{\text{enc}}\right)\right)$ ▷$\mathbf{X}^{l-1}_{\text{enc}} \in \mathbb{R}^{S\times d_k}$

  $\mathbf{X}^{l}_{enc} = LayerNorm\left(\mathbf{X}^{l-1}_{enc} + FFN\left(\mathbf{X}^{l-1}_{enc}\right)\right)$ ▷$\mathbf{X}^{l}_{\text{enc}} \in \mathbb{R}^{S\times d_k}$

end

$\mathbf{X}^0_{\text{dec}} = Embed\left(\mathbf{X}_{\text{dec}}\right)$ ▷$\mathbf{X}^0_{\text{dec}} \in \mathbb{R}^{(S+O)\times d_k}$

for ***for*** l *in* $\{1, \cdots, M\}$ **do**

  $\mathbf{X}^{l-1}_{dec} = LayerNorm\left(\mathbf{X}^{l-1}_{dec} + Attn\left(\mathbf{X}^{l-1}_{dec}\right)\right)$ ▷ Decoder

  $\mathbf{X}^{l-1}_{\text{dec}} = LayerNorm\left(\mathbf{X}^{l-1}_{\text{dec}} + Attn\left(\mathbf{X}^{l-1\prime}_{\text{dec}}, \mathbf{X}^{N}_{\text{enc}}\right)\right)$ ▷$\mathbf{X}^{l-1}_{\text{dec}} \in \mathbb{R}^{(S+O)\times d_k}$

  $\mathbf{X}^{l}_{\text{dec}} = LayerNorm\left(\mathbf{X}^{l-1}_{\text{dec}} + FFN\left(\mathbf{X}^{l-1}_{\text{dec}}\right)\right)$ ▷$\mathbf{X}^{l}_{\text{dec}} \in \mathbb{R}^{(S+O)\times d_k}$

end

$\mathbf{y} = MLP\left(\mathbf{X}^{M}_{\text{dec}}\right)$ ▷$\mathbf{y} \in \mathbb{R}^{(S+O)\times C}$

Return y ▷ Return the prediction results

Algorithm 3: MultiStep Transformer

F. List Of Publications

List Of All Publications

Publications that are relevant to the thesis

Vaisakh Shaj, Philipp Becker, Dieter Buchler, Harit Pandya, Niels van Duijkeren, C James Taylor, Marc Hanheide, and Gerhard Neumann (2020). "Action-Conditional Recurrent Kalman Networks For Forward and Inverse Dynamics Learning". In: *Conference On Robot Learning*.

Vaisakh Shaj, Dieter Buchler, Rohit Sonker, Philipp Becker, and Gerhard Neumann (2022). "Hidden Parameter Recurrent State Space Models For Changing Dynamics Scenarios". In: *International Conference On Learning Representations*.

Vaisakh Shaj, Saleh Gholam Zadeh, Ozan Demir, Luiz Douat, and Gerhard Neumann (2023). "Multi Time Scale World Models". In: *Advances in Neural Information Processing Systems* 36. **(Spotlight $<$ 3%)**.

Other Publications

Gholam Zadeh, Saleh, **Vaisakh, Shaj**, Patrick Jahnke, Gerhard Neumann, and Tim Breitenbach (2024). "Towards Measuring Predictability: To which extent data-driven approaches can extract deterministic relations from data exemplified with time series prediction and classification". In: *Transactions on Machine Learning Research (TMLR)*.

Gospodinov, Emiliyan, **Vaisakh Shaj**, Philipp Becker, Stefan Geyer, and Gerhard Neumann (2024). "Adaptive World Models: Learning Behaviors by Latent Imagination Under Non-Stationarity". In: *NeurIPS 2024 Adaptive Foundation Models Workshop*.

Reuss, Moritz, Niels van Duijkeren, Robert Krug, Philipp Becker, **Vaisakh,Shaj**, and Gerhard Neumann (2022). "End-to-End Learning of Hybrid Inverse Dynamics Models for Precise and Compliant Impedance Control". In: *Proceedings of Robotics: Science and Systems*.

Vaisakh, Shaj*, Konda Reddy Mopuri*, and R. Venkatesh Babu (2020). "Adversarial Fooling Beyond "Flipping the Label"". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. ***Equal contribution**.

Vaisakh, Shaj*, Gaurav Kumar Nayak*, Konda Reddy Mopuri*, R Venkatesh Babu, and Anirban Chakraborty (2019). "Zero-Shot Knowledge Distillation in Deep Networks". In: *International Conference on Machine Learning*. ***Equal contribution**.

Vaisakh,Shaj and Puranjoy Bhattacharya (2017). "Learning Sparse Adversarial Dictionaries for Multi-Class Audio Classification". In: *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*. IEEE, pp. 315–320.

Under Review/Preprints

Jacob, Ruben, Vaisakh Shaj, Philipp Becker, and Gerhard Neumann (2023). "Episode Transformer: Model-based Episodic Reinforcement Learning". In: *Openreview preprint*.

Patents

Mathews, Sherin M., **Vaisakh, Shaj**, Sriranga Seetharamaiah, Carl D. Woodward, and Kantheti VVSMB Kumar (May 2021). "Methods, systems, and media for detecting anomalous network activity". US11005868B2.