Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

Cognitively-inspired learning of causal relations in a simulated robotic environment

Bachelor Thesis

2024
Miroslav Cibula

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Cognitively-inspired learning of causal relations in a simulated robotic environment

Bachelor Thesis

Study Programme:	Applied Informatics
Field of Study:	Computer Science
Department:	Department of Applied Informatics
Supervisor:	prof. Ing. Igor Farkaš, Dr.
Consultant:	Mgr. Michal Vavrečka, PhD.

Bratislava, 2024
Miroslav Cibula

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Miroslav Cibula
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Cognitively-inspired learning of causal relations in a simulated robotic environment
Kognitívne inšpirované učenie sa kauzálnych vzťahov v simulovanom robotickom prostredí

Anotácia: Pozorovanie a učenie sa kauzálnych vzťahov v danom prostredí je dôležitý prvok kognície ľudí a vyšších živočíchov. Prítomnosť kauzálneho modelu sveta umožňuje agentovi predikovať efekt jeho akcií na prostredie. Tieto informácie dokáže následne v kombinácii s mentálnou simuláciou využiť na kreatívne a flexibilné plánovanie a univerzálne riešenie úloh v známom prostredí.

Cieľ: 1. Navrhnite systém umožňujúci robotickému ramenu (agentovi) učiť sa kauzalitu interakciou s objektmi, reprezentovať a uchovávať tieto informácie, a aplikovať ich pri riešení jednoduchých manipulačných úloh.
2. Implementujte simulované randomizovateľné prostredie pre tréning a inferenciu, a otestujte komponenty systému.
3. Analyzujte experimentálne výsledky.

Literatúra: Hellström, T. (2021). The relevance of causation in robotics: A review, categorization, and analysis. Paladyn, Journal of Behavioral Robotics, 12(1).
Lee, T.E. et al. (2021). Causal reasoning in simulation for structure and transfer learning of robot manipulation policies. IEEE ICRA.
Vavrečka, M., Sokovnin, N., Mejdrechová, M., & Šejnová, G. (2021). MyGym: Modular Toolkit for Visuomotor Robotic tasks. IEEE 33rd Int. Conf. on Tools with AI (ICTAI).

Vedúci: prof. Ing. Igor Farkaš, Dr.
Konzultant: Mgr. Michal Vavrečka, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 05.10.2023

Dátum schválenia: 10.10.2023 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

52373788

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Miroslav Cibula
Study programme:	Applied Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Cognitively-inspired learning of causal relations in a simulated robotic environment

Annotation: Observing and learning causal relations in a given environment is an essential element of cognition in humans and high animals. A causal model of the world allows an agent to predict the effect of its actions on the environment. When combined with mental simulation, such information can be utilized for creative, flexible, and universal task-solving in a known environment.

Aim:
1. Design a system enabling a robotic arm (agent) to learn causality by interacting with objects, to represent and store this information, and to apply it in solving simple manipulation tasks.
2. Implement a simulated randomizable environment for the system training and inference and test the system components.
3. Analyze experimental results using available computational methods.

Literature:
Hellström, T. (2021). The relevance of causation in robotics: A review, categorization, and analysis. Paladyn, Journal of Behavioral Robotics, 12(1).
Lee, T.E. et al. (2021). Causal reasoning in simulation for structure and transfer learning of robot manipulation policies. IEEE ICRA.
Vavrečka, M., Sokovnin, N., Mejdrechová, M., & Šejnová, G. (2021). MyGym: Modular Toolkit for Visuomotor Robotic tasks. IEEE 33rd Int. Conf. on Tools with AI (ICTAI).

Supervisor:	prof. Ing. Igor Farkaš, Dr.
Consultant:	Mgr. Michal Vavrečka, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	05.10.2023

Approved: 10.10.2023 doc. RNDr. Damas Gruska, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgments: First and foremost, I would like to express my gratitude to my supervisor, prof. Ing. Igor Farkaš, Dr., for his constant guidance and support throughout this project and for his valuable advice and ideas. I want to thank my consultant, Mgr. Michal Vavrečka, PhD., for his ideas and feedback on the concepts presented in this work, as well as for his technical help with robotic simulations. My sincere thanks also goes to RNDr. Kristína Malinovská, PhD. for her extensive feedback and advice on various aspects of this research. I would also like to thank Mgr. Iveta Bečková and Mgr. Štefan Pócoš for their valuable suggestions, mainly regarding neural network analysis and custom loss computation methods.

Abstrakt

Učenie sa kauzálnych vzťahov umožňuje ľuďom predikovať následky ich akcií v známom prostredí a použiť túto znalosť na plánovanie komplexnejších úkonov. Znalosť kauzálnych vzťahov taktiež zachytáva správanie sa prostredia, čo je možné využiť na jeho analýzu ako aj na hľadanie dôvodov za takýmto správaním. V tejto práci skúmame schopnosť učenia sa kauzálnych vzťahov pozorovaných v simulovanom robotickom prostredí pomocou dopredného a inverzného modelu. Inšpirujeme sa pri tom mechanizmami prítomnými v ľudskej kognícii a navrhujeme niekoľko prístupov ku konštrukcii takýchto modelov. Ďalej skúmame, ako tieto modely natrénované na syntetických dátach vygenerovaných v simulácii vieme analyzovať, a ako z nich vieme extrahovať naučené nízkoúrovňové kauzálne vzťahy, ktoré môžu poslúžiť ako podklad k dimenzionálnej redukcii stavových reprezentácii daného prostredia. V poslednom bode navrhujeme prístup k plánovaniu akcií za cieľom riešenia jednoduchých robotických manipulačných úloh. Na tento účel využívame niektoré vyvinuté metódy a koncepty uvedené v tejto práci. Na vyhodnotenie navrhovaných metód vykonávame niekoľko experimentov so simulovaným robotickým ramenom, ktoré sa učí kauzálne vzťahy v postupne náročnejších úlohách.

Kľúčové slová: kauzálne učenie, dopredný model, inverzný model, interpretabilita, plánovanie

Abstract

Learning causal relations allows humans to predict the effect of their actions on the known environment and use this knowledge to plan the execution of more complex actions. Such knowledge also captures the behaviour of the environment and can be used for its analysis and the reasoning behind the behaviour. In this thesis, we explore learning causal relations observed in a simulated robotic environment using the forward and inverse models. Inspired by mechanisms of human cognition, we propose multiple approaches to constructing such models. Further, we investigate how these models trained on synthetic data generated in a simulation can be analyzed to extract learned low-level causal relationships, which could be then used as a basis for dimensional reduction of the environment's state representations as well as for the explainability of the environment's behaviour at higher levels. Finally, we propose an approach for planning actions for simple robotic manipulation task-solving using some of the developed methods and concepts presented in this work. In order to evaluate the proposed methods, we conduct several experiments concerning a simulated robotic arm learning causal relations in tasks of increasing difficulty.

Keywords: causal learning, forward model, inverse model, interpretability, planning

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

BPTT backpropagation through time . . . 10
DoF degrees of freedom . . . 29
FM forward model . . . 6
GRU gated recurrent unit . . . 10
HIM hindsight information matching . . . 12
IM inverse model . . . 6
KAN Kolmogorov-Arnold network . . . 7
LSTM long short-term memory . . . 10
MAE mean absolute error . . . 30
MDP Markov decision process . . . 9
ML machine learning . . . 4
MLP multilayer perceptron . . . 6
MSE mean squared error . . . 18
PDP partial dependence plot . . . 21
RCSL return-conditioned supervised learning . . . 12
RL reinforcement learning . . . 9
RNN recurrent neural network . . . 10
RvS reinforcement learning via supervised learning . . . 12
SMCs sensorimotor contingencies . . . 4
TM trajectory model . . . 23
UCM uncontrolled manifold . . . 19
XAI explainable artificial intelligence . . . 8

List of Symbols

General

x	scalar
$\boldsymbol{v}$	vector
$\boldsymbol{M}$	matrix
$\boldsymbol{u} \subseteq \boldsymbol{v}$	subvector of the vector $\boldsymbol{v}$
$\boldsymbol{u} \subset \boldsymbol{v}$	proper subvector of vector $\boldsymbol{v}$
$\boldsymbol{u} \cup \boldsymbol{v}$	vector concatenation
$\boldsymbol{u} \setminus \boldsymbol{v}$	vector $\boldsymbol{u}$ without its subvector $\boldsymbol{v}$
$\left\|\boldsymbol{u}\right\|_p$	L^p norm of vector $\boldsymbol{u}$
$\dim(\boldsymbol{u})$	dimensionality of vector $\boldsymbol{u}$
$\mathcal{P}(A)$	power set of set A
$\lvert A \rvert$	cardinality of set A / length of sequence A
$\Pr(A)$	probability
$\Pr(B \mid A)$	conditional probability
$\mathbb{E}[X]$	expected value

Environment

t	timestep
T	maximal timestep
$\mathcal{S}$	state space
$\mathcal{A}$	action space
$\mathcal{E}$	environment
$\boldsymbol{s}(t)$	state vector at time t
$\boldsymbol{s}'(t)$	state vector excluding joint configuration at time t
$\hat{\boldsymbol{s}}(t)$	predicted state vector at time t
$\tilde{\boldsymbol{s}}(t)$	rectified state vector at time t
$\boldsymbol{\theta}(t)$	joint configuration vector at time t
$\boldsymbol{ef}(t)$	Cartesian end-effector position vector at time t
$\boldsymbol{a}(t)$	action vector at time t
π	policy

τ	trajectory
τ_s	state trajectory
τ_a	action sequence

Models

$\hat{\boldsymbol{y}}$	prediction
$\mathrm{FM}(\cdot)$	forward model
$\mathrm{IM}(\cdot)$	inverse model
$\mathrm{TM}(\cdot)$	trajectory model
$\mathrm{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	mean squared error
$\mathrm{MAE}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	mean absolute error
$\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	loss function
$\mathcal{L}_{\mathrm{FM}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	forward model loss function
$\mathcal{L}_{\mathrm{TM}}^{(\mathrm{I})}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	element-wise trajectory model loss function
$\mathcal{L}_{\mathrm{TM}}^{(\mathrm{II})}(\hat{\boldsymbol{y}}, \boldsymbol{y})$	FM/IM-aided trajectory model loss function
η	initial learning rate
λ	initial weight decay

Introduction

Observing and learning causal relations in a given environment is an essential element of cognition in humans and other higher animals. Thanks to this ability, agents can assemble their intuitive knowledge (such as intuitive physics and psychology) about the world in which they operate from multiple observations and use them to further predict the environment's behaviour, mainly in response to their actions. Such ability is principal to common sense understanding – a concept mastered even by young children while having proven to be highly inapprehensible for artificial intelligence.

In this thesis, we were inspired by causal learning and other mechanisms observed in human cognition, leveraging them in constructing system planning procedures to be executed to solve simple robotic manipulation tasks in a simulated environment. Specifically, we use forward and inverse models to learn the effects of actions performed by an agent (a simulated robotic arm). These actions are a product of simple motor babbling or other strategies allowing the agent to interact with the environment and observe its behaviour.

Further, as a by-product of this approach, we hypothesize that these models trained on a sufficient amount of observations contain knowledge about the environment and the task the agent was performing. We argue that this knowledge is similar to intuitive theories assembled by humans from causal experience collected since an early age. As such knowledge can be helpful in the analysis of the environment, the task, and their properties, we explore methods for extracting this information by analyzing the trained forward model using explainable artificial intelligence methods.

Finally, we propose a model generating trajectories for solving simple robotic manipulation tasks. For this purpose, we use sequence modelling using recurrent neural networks and leverage the trained forward and inverse models in post-processing generated trajectories. We were inspired by imitation learning, utilizing it for the sequence modelling optimization within the supervised learning paradigm instead of in robotics, the more commonly used reinforcement learning approach.

In Chapter 1, we provide an essential overview of concepts used in this thesis – specifically, causal learning, neural forward and inverse models, reinforcement learning and sequence modelling using recurrent neural networks. Chapter 2 lists related research and alternative solutions to some methods and problems presented in this work.

In Chapter 3, we define and describe the basic terms used throughout this thesis and elaborate on the aims of our research, briefly presented in this section. In Chapter 4, we propose the principal methods of this research and describe them in a general manner. Finally, the application of these methods to the designed experiments is described in Chapter 5. The chapter describes the experiments and testing procedures, the methods' concrete implementation details, and provides the evaluation results, as well as the relevant discussion and analyses.

Chapter 1

Preliminaries

In this chapter, serving as a theoretical overview, we summarize methods, approaches and technologies used in the thesis. Specifically, we provide an overview of causality, causal relationships, and causal learning from both human cognition and machine intelligence point of view as we use causal learning as a central concept in our proposed methods (Chapter 4).

We further describe the biological and robotic backgrounds of forward and inverse models used as facilitators for causal learning as well as artificial neural networks used for their implementation. In addition, as we leverage the models' analysis, we summarize the principles behind the family of analysis methods used.

Lastly, we describe the principal components of sequence modelling and reinforcement learning used for the proposed planning method.

1.1 Causal Learning

Causal learning refers to capturing and learning causal relationships from observations of the behaviour of an environment in which the agent (e.g., human or robot) operates. This ability allows agents to form intuitive theories and use them to predict the environment's behaviour in response to their actions (Gerstenberg & Tenenbaum, 2017), establishing common sense understanding, including the knowledge of intuitive physics and psychology (Lake et al., 2016).

Human Causal Cognition

Causal cognition has been studied extensively on both human and machine intelligence levels. Regarding human cognition, Gärdenfors and Lombard (2018; 2017) propose a causal cognition evolution model categorizing levels of causal understanding varying in complexity. This model's grades range from understanding the perceived effects of the agent's motor actions to understanding interactions between entities of

the environment and the ability to extrapolate from this knowledge.

Causality in Robotics

Regarding machine intelligence, Lake et al. (2014; 2016) argue that causality might be one of three central "ingredients" needed to replicate rapid learning akin to human learning.[1] The argument supports the current effort to transfer causal cognition to robotics, involving embodied agents interacting with the world. Analogically to the model of the evolution of human causal cognition mentioned above, Hellström (2021) proposes a categorization of robot causal cognition ranging in difficulty from simple sensorimotor learning to the ability to plan and beyond. These grades are divided into three groups: learning causal relations, inferring the causes related to an interacting human, and robot deciding how to act (Fig. 1.1).

In this work, we focus on low-level causality regarding two categories: sensorimotor self-learning (C1) and learning the consequences of an agent's own actions on objects in the environment (C2). Causal relationships adhering to these two categories – especially those following our definition (Chapter 3), may be conceptually equivalent to sensorimotor contingencies (SMCs) (O'Regan & Noë, 2001; Noë, 2004) defined as "law-like relations between actions and contingent changes in the sensory signals" (Maye & Engel, 2012) with relation to contingency in psychology generally defined as "a correspondence of one's behavior to another's behavior" (Yamaoka et al., 2007).

Causality in Machine Learning

Besides applications in robotics, causality is also studied as part of machine learning (ML) research. Zhang et al. (2017) and Zhu et al. (2020) argue that causality understanding can be beneficial toward building more robust models with common sense. In practice, one of the most common ML applications of causal learning is causal model construction via Bayesian modelling. This approach concerns building symbolic representations by identifying and extracting concepts and relations between them based on the observations of the world in which the model operates. Lake et al. (2016) argue that causal model construction as a primary learning mechanism could be an alternative to classical pattern recognition methods currently dominating ML research.

Specific examples of causal model construction include human-like concept learning through Bayesian program learning (Lake et al., 2015) and other probabilistic modelling methods (Tenenbaum, 1998, 1999; Lake, 2014), and human-like learning of abstract theory of causality (Goodman et al., 2011) using Bayesian networks (Pearl, 1985).

[1] The other two are compositionality (i.e., the ability to construct new representations from more primitive elements) and learning-to-learn (i.e., the ability to reuse knowledge of related concepts or tasks to accelerate learning of a new concept or task).

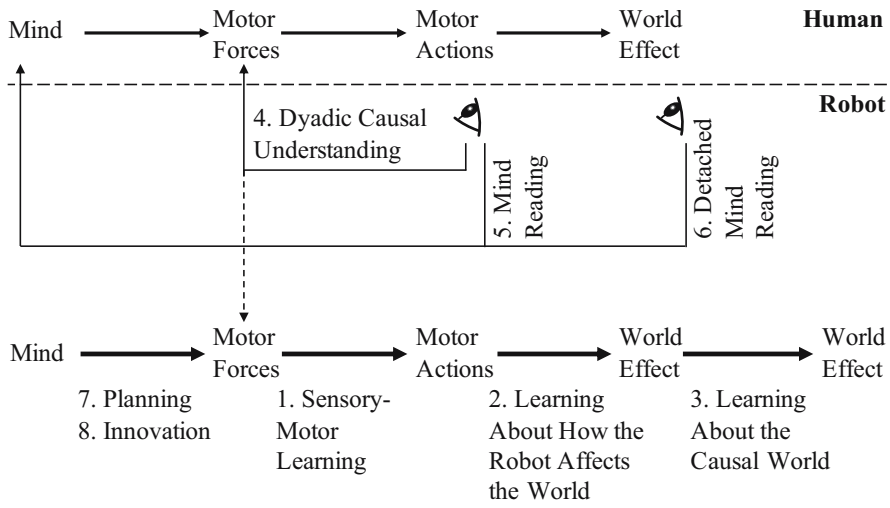

Figure 1.1: Diagram of robot causal cognition categorization. Bold arrows refer to learning causal relations, while thin solid arrows refer to "inference of causes related to an interacting human". Adjacent categories are required to facilitate the agent's understanding of respective relationships (Hellström, 2021).

As demonstrated by the mentioned research, causal model construction or, more broadly, causal learning is predominantly part of the symbolic paradigm (Kotseruba & Tsotsos, 2018) as it mainly operates with symbolic representations on different levels (Schölkopf, 2022). This contrasts with the sub-symbolic models and systems (Rosenblatt, 1958) generally following the parallel distributed processing paradigm (McClelland et al., 1987, 1988) inspired by low-level brain mechanisms and neural structures.

For a further comprehensive overview of causal cognition in humans, in robots, and causality research in ML, see Gärdenfors and Lombard (2018), Hellström (2021), and Schölkopf (2022) and Zhang et al. (2017), respectively.

1.2 Forward and Inverse Models

As de Lange et al. (2018) note, humans and similar higher animals have been labelled "anticipatory systems" (Rosen, 2012) due to the fact that they can construct predictive models (Clark, 2013) of themselves and their environment and use them to "quickly and robustly make sense of incoming data". Such models are representations of the knowledge acquired by the agent from observations of the environment, allowing it to infer expectations about the world's behaviour.

More concretely, causal and especially sensorimotor knowledge produced by causal learning performed by a robotic system, solving C1 and C2 tasks in our case, can be represented by a pair of complementary internal models: the forward model (FM) and the inverse model (IM) (Wolpert & Kawato, 1998).

While the FM (Dearden & Demiris, 2005) unambiguously predicts perceivable consequences of the agent's actions, the IM predicts actions needed to reach the desired state from the initial state. In contrast to the FM, the IM is mathematically ill-defined in general, as the IM also models inverse kinematics, which is ill-posed in redundant robots (Nguyen-Tuong & Peters, 2011).

Forward and inverse models in robotics take inspiration from the internal model principle of control theory (Francis & Wonham, 1976) modelling physiological internal models (Wolpert & Flanagan, 2001; Miall & Wolpert, 1996; Sperry, 1950; von Holst & Mittelstaedt, 1950). It is generally acknowledged (Dogge et al., 2019) that humans use a forward internal model to predict the outcomes of their motor actions. Dogge et al. (2019) describe "[physiological] forward models [...] as simulations of the motor system that use a copy of the motor command, known as an efference copy [...], to predict the sensory consequences of the action in question (known as corollary discharge)".

It should be noted that while in this thesis we use forward models to predict environment-related outcomes beyond body-related outcomes, Dogge et al. (2019) argue that involvement of biological motor-based forward models to such extent is "limited and hitherto unjustified".

Artificial Neural Networks

As both FM and IM represent functions, modelling (learning) can be performed by artificial neural networks as universal function approximators (Hornik et al., 1989). For the forward and inverse modelling in this work, we specifically use multilayer perceptrons (MLP) (Rosenblatt, 1958, 1962; Minsky & Papert, 2017) as universal regressors.

The topology of MLP models consists of L layers of neural units with layers $l = 1$, $l = L$ and $1 < l < L$ defined as an input layer, output layer and hidden layers, respectively. Each layer is composed of d_l neural units, with $d_1 = \dim(\boldsymbol{x})$ and $d_L = \dim(\boldsymbol{y})$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are real input and output vectors of a function f being approximated. MLP is fully connected, meaning that each unit i of each layer l except the output layer is connected with every neuron j of the subsequent layer $l+1$ using oriented synapse with assigned weight $w_{ij}^{(l)} \in \mathbb{R}$. Then, activation of i-th neuron in l-th layer can be computed as

$$h_i^{(l)} = \varphi_l \left(\sum_{j=1}^{d_{l-1}+1} w_{ij}^{(l)} h_j^{(l-1)} \right), \tag{1.1}$$

where φ_l denotes activation function of the l-th layer. Additionally, $\boldsymbol{h}^{(L)} \equiv \hat{\boldsymbol{y}}$ where $\hat{\boldsymbol{y}}$ is a predicted output. To reformulate, considering layer a function

$$h^{(l)}(\boldsymbol{v}) \triangleq \varphi_l \left(\boldsymbol{W}^{(l)} \boldsymbol{v} \right) \tag{1.2}$$

where $\boldsymbol{W}^{(l)}$ is the weight matrix of the l-th layer, the approximation of sought function f can be defined as

$$\hat{f}(\boldsymbol{x}) \triangleq \left(h^{(L)} \circ h^{(L-1)} \circ \cdots \circ h^{(2)} \circ h^{(1)} \right) (\boldsymbol{x}). \tag{1.3}$$

In order to train the regressor, the model's weights are commonly optimized in a supervised learning scheme where the error of generated predictions $\hat{\boldsymbol{y}}$ is computed against the ground-truth targets $\boldsymbol{y}$ using an error function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$. The computed error is further backpropagated (Rumelhart et al., 1986) through the whole network, with new weights being calculated as

$$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) + \Delta w_{ij}^{(l)}, \tag{1.4}$$

where $\Delta w_{ij}^{(l)}$ denotes weight adjustment defined as

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} \tag{1.5}$$

with η designating learning rate constant.

Recently, as an alternative to MLPs, Liu et al. (2024) have proposed Kolmogorov-Arnold networks (KAN) as an application of Kolmogorov-Arnold representation theorem (Kolmogorov, 1961). The authors claim that KANs have a series of properties such as "comparable or better accuracy than larger MLPs in function fitting tasks" or interpretability. The interpretability results from KANs' essential ability to decompose multivariate functions being learned into simple univariate functions capturing relationships between input and output features of these networks, thus providing an alternative to explainability methods described in Section 1.3. Although we do not use KANs in this thesis, they could be helpful in our future research (for more details, see Discussion in Section 5.2).

1.3 Model Analysis

In this thesis, we analyze neural models using explainable AI (XAI) methods. Specifically, we study feature importance, evaluating the significance of a specific input feature on the prediction of a specific output feature.

Shapley Values

Feature importance is commonly computed using Shapley values (Shapley, 1953) while interpreting the task of a single output feature prediction for a single data point $\boldsymbol{x}$ as a cooperative game. Input features are interpreted as players belonging to possible coalitions $S \in \mathcal{P}(F)$, where F is the set of all features. Then, interpreting a model f trained on a set of features as a value function evaluating the worth of coalition, Shapley value ϕ_i defines the marginal contribution of feature i for the input $\boldsymbol{x}$ for the model f:

$$\phi_i(\boldsymbol{x}) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} \Delta_i(S, \boldsymbol{x}), \tag{1.6}$$

where $|S|$ and $|F|$ denote the number of features in the coalition and the total number of feature, respectively. $\Delta_i(S, \boldsymbol{x})$ denotes the marginal contribution of feature i to coalition S defined as

$$\Delta_i(S, \boldsymbol{x}) = f_{S \cup \{i\}}\left(\boldsymbol{x}_{S \cup \{i\}}\right) - f_S\left(\boldsymbol{x}_S\right) \tag{1.7}$$

with f_S and $f_{S \cup \{i\}}$ denoting trained models on the feature subset S and S including the feature i, respectively, and x_S denoting the values of input features from S.

SHAP Methods

As computing Shapley values is generally NP-hard (Matsui & Matsui, 2001), various methods for estimating them have been developed, with the most popular being SHAP (Lundberg & Lee, 2017), which unifies different additive feature attribution methods. Here, we were experimenting with two variants: KernelSHAP and DeepSHAP.

KernelSHAP is a model-agnostic kernel-based method utilizing the idea of local surrogate models (Ribeiro et al., 2016) to estimate Shapley values. However, since it does not make any assumptions about the analyzed model, it is generally slower than model-specific methods on account of its combinatorial nature. DeepSHAP, on the other hand, is applicable only to neural models as it uses attribution rules of DeepLIFT method (Shrikumar et al., 2017) to propagate SHAP values from the output layer back to the input layer.

SHAP methods are local, providing an explanation for one prediction. However, thanks to their properties, local explanations can be aggregated across the set of instances, providing global feature importance within the analyzed model.

For further comprehensive review of XAI methods, see (Zhang et al., 2021).

1.4 Reinforcement Learning

Reinforcement learning (RL) is an ML paradigm of algorithms learning by interacting with a given environment. The principal objective of agents controlled by these algorithms is to perform actions maximizing the cumulative reward. Although methods proposed in this work operate beyond the RL paradigm, we reuse some of its principles and nomenclature.

Problems solvable by RL methods are most commonly modelled as Markov decision processes (MDP) (Bellman, 1957). Discrete-time MDP can be defined as a tuple

$$\text{MDP}\colon\ (\mathcal{S}, \mathcal{A}, T, R, \gamma)\,, \tag{1.8}$$

where $\mathcal{S}$ and $\mathcal{A}$ denote discrete or continuous state and action spaces, respectively. Then, in a discrete timestep t, an action $\boldsymbol{a}(t) \in \mathcal{A}$ transitions the state $\boldsymbol{s}(t)$ to $\boldsymbol{s}(t+1)$, with $\boldsymbol{s}(t), \boldsymbol{s}(t+1) \in \mathcal{S}$, with probability

$$\Pr\left[\boldsymbol{s}(t+1) \mid \boldsymbol{s}(t), \boldsymbol{a}(t)\right] \equiv T\left[\boldsymbol{s}(t), \boldsymbol{a}(t), \boldsymbol{s}(t+1)\right], \tag{1.9}$$

where T denotes a state transition function. Transitioning from the state $\boldsymbol{s}(t)$ to $\boldsymbol{s}(t+1)$ with the action $\boldsymbol{a}(t)$ is evaluated by reward function $r(t) = R\left[\boldsymbol{s}(t), \boldsymbol{a}(t), \boldsymbol{s}(t+1)\right]$.

The process of taking action in a particular state produces a trajectory

$$\tau = [\boldsymbol{s}(0), \boldsymbol{a}(0), r(0), \boldsymbol{s}(1), \boldsymbol{a}(1), r(1), \ldots] \tag{1.10}$$

which can be evaluated by computing its discounted cumulative return

$$G = \sum_{t=0}^{\infty} \gamma^t r(t), \tag{1.11}$$

where $0 \leq \gamma \leq 1$ is a discount factor. The goal of RL algorithms is to learn optimal policy π^* such that

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}\left[\boldsymbol{s}(0)\right], \tag{1.12}$$

where $V^{\pi}\left[\boldsymbol{s}(0)\right]$ denotes a state-value function defined as

$$V^{\pi}\left[\boldsymbol{s}(0)\right] \triangleq \mathbb{E}\left[G \mid \boldsymbol{s}(0), \pi\right] \tag{1.13}$$

providing the expected cumulative return of a trajectory produced by an agent starting in state $\boldsymbol{s}(0)$ and taking action $\boldsymbol{a}(t) \sim \pi\left[\boldsymbol{s}(t)\right]$ with stochastic policy $\pi\left[\boldsymbol{s}(t)\right] \equiv \Pr\left[\boldsymbol{a}(t) \mid \boldsymbol{s}(t)\right]$.

The generation of the whole or partial future trajectory based on a model or policy prior to the trajectory's execution (i.e., offline) can be, in this context, understood as planning. While the offline planning problem is commonly solved using RL methods, recent advances demonstrate that such a problem can also be solved using supervised learning methods (Section 2.2).

1.5 Sequence Modelling

Sequence modelling in its autoregressive form (Goodfellow et al., 2016; Jurafsky & Martin, 2024) is a problem considering modelling

$$\Pr(x_1, x_2, \ldots, x_T) = \Pr(x_1) \prod_{t=2}^{T} \Pr(x_t \mid x_{t-1}, x_{t-2}, \ldots, x_1). \tag{1.14}$$

An autoregressive model performing such modelling can be a neural network classifier or regressor, depending on whether the modelled variables are discrete or continuous. Most of the modern architectures used in the solving of this problem include either gated recurrent neural networks (RNN) or more recent Transformers (Vaswani et al., 2017). In this thesis, however, we use only RNNs.

RNNs are sequential models generally using hidden state $\boldsymbol{h}^{(t-1)}$ from the previous timestep to condition the prediction in the current timestep:

$$\Pr(x_t \mid x_{t-1}, x_{t-2}, \ldots, x_1) \approx \Pr\left(x_t \mid \boldsymbol{h}^{(t-1)}\right) \tag{1.15}$$

Due to their sequential nature, RNNs are commonly optimized using backpropagation through time (BPTT) (Werbos, 1988). BPTT algorithm unfolds an RNN through T timesteps into separate pseudo-networks retaining identical copies of parameters. Each pseudo-network is then optimized using a standard backpropagation algorithm (Rumelhart et al., 1986), and weight updates across all timesteps are summed together. A more efficient and contemporary version of the BPTT algorithm is known as Truncated BPTT (Williams & Peng, 1990). Truncated BPTT runs BPTT in intervals on small segments of the input sequence and thus only estimates the gradient used to update the model's parameters. This approach contrasts with the classical BPTT, which performs a backward pass through the whole sequence and fully computes the gradient. Such operation is very time-expensive, especially for longer sequences.

Gated RNNs, in contrast with early simple recurrent models, support gating of the hidden state, meaning they employ dedicated mechanisms for updating and resetting it. Gated RNNs involve mainly long short-term memories (LSTM) (Hochreiter & Schmidhuber, 1997) proposed initially for making the long-term dependency learning (Bengio et al., 1994; Hochreiter et al., 2001) feasible, or gated recurrent units (GRU) (Cho et al., 2014) providing more simple LSTM-inspired memory cell architecture consequently providing computation time reduction while maintaining performance comparable with LSTMs (Chung et al., 2014).

Chapter 2

Related Work

In this chapter, we provide an overview of existing full or partial solutions alternative to the main contribution points of this thesis: learning causal relations in robotics (Section 2.1) and planning using sequence modelling aided by causal models (Section 2.2).

2.1 Causal Learning in Robotic Applications

Albeit causal learning (for an overview, see Section 1.1) and causality-based approaches in the context of robotics are presently deemed under-explored (Stocking et al., 2022; Lee et al., 2023), it has been demonstrated that they can be helpful for multiple applications. In many robotics and RL applications, studying and learning causal relations can reveal relationships between state and action variables regarding a given environment or task. This information is often used to reduce the complexity of either space or identify relevant or important variables.

Our work, especially the knowledge extraction part (Section 4.3), was inspired by CREST (Lee et al., 2021), where authors used causal reasoning in simulation to learn the relevant state space variables for a robot manipulation policy. In their approach, they conduct causal interventions[2] to elicit the relationships between action and state variables. This allows them to reduce the complexity of neural network policies using only state variables relevant to the task being solved.

Leveraging the research on CREST, SCALE approach (Lee et al., 2023) for discovering and learning diverse robot skills has been proposed. SCALE uses CREST in a pipeline to identify sets of relevant variables related to individual skills.

A method proposed by Diehl and Ramirez-Amaro (2023) concerns a causal Bayesian network (Pearl, 1985) learning causal relationships between task executions and their consequences. They then utilize this model to allow a robot to "conjecture" whether

[2]This scheme is similar to the randomized controlled trials (Fisher, 1925).

and why the action executed in its current state will succeed or fail.

Furthermore, Sontakke et al. (2021) introduce causal curiosity, an intrinsic reward allowing the agent to discover latent causal factors in the dynamics of the environment it operates in. Wang et al. (2022) use the causal dynamics model to remove unnecessary dependencies between the state and action variables and subsequently use the dynamics model to yield state abstractions. Sonar et al. (2021) utilize causality to learn invariant policies.

2.2 Planning as a Sequence Modelling Problem

Planning is predominantly a reinforcement learning problem. However, as recently demonstrated by Chen et al. (2021) and Janner et al. (2021), planning can also be achieved beyond the RL paradigm. The mentioned approaches use sequence modelling (for an overview, see Section 1.5) in combination with imitation learning to generate trajectories needed to complete a given task. Such recent developments demonstrate, that the problem of online RL is being shifted to the domain of supervised learning. Specifically, RL components are often entirely replaced with offline behavioural cloning (Furuta et al., 2021).

Inspired by the prior research, Wen et al. (2022) propose their own Transformer architecture for solving cooperative multi-agent RL problems. Furuta et al. (2021) further demonstrate that these approaches perform hindsight information matching (HIM). They define HIM as a method concerning "training policies that can output the rest of trajectory that matches some statistics of future state information" and propose a Generalized Decision Transformer capable of solving any HIM problem.

The paradigm covering these approaches has been coined as return-conditioned supervised learning (RCSL), whose central idea "is to learn the return-conditional distribution of actions in each state, and then define a policy by sampling from the distribution of actions that receive high return" (Brandfonbrener et al., 2022). A related broader concept has been referred to as reinforcement learning via supervised learning (RvS) (Emmons et al., 2021).

Planning leveraging the trajectory modelling in a supervised learning scheme inherently requires a training dataset. For this reason, most approaches mentioned above employ imitation learning (Zare et al., 2023; Mandlekar et al., 2021), a process in which an expert demonstrates a desired behaviour and an agent learns by imitation from the collected observations of expert demonstrations.

As an alternative to imitation learning, Oh et al. (2018) propose self-imitation learning during which the agent imitates its own past good experiences. In a similar fashion, Ghosh et al. (2019) propose a hybrid algorithm combining reinforcement and

supervised learning in which "an agent continually relabels and imitates the trajectories it generates to progressively learn goal-reaching behaviors from scratch". In contrast with self-imitation learning by Oh et al. (2018), this approach reuses and imitates every generated trajectory, not only a small subset. This approach is also one of the applications of the RvS paradigm mentioned above.

Chapter 3

Aims and Task Formulation

In this chapter, we formulate the aims of our work, introduce and define concepts and nomenclature we use throughout the thesis, and state simplifying assumptions and delimitations applied in this work.

The central concept of this thesis is learning causal relationships. In the context of this thesis, we operate with a concept of low-level, mechanistic causal relationships, which we understand as a discrete-time transition function

$$\mathcal{T}_{\mathcal{E}}\colon\ [\boldsymbol{s}(t), \boldsymbol{a}(t)] \mapsto \boldsymbol{s}(t+1) \tag{3.1}$$

where $\boldsymbol{s}(t), \boldsymbol{s}(t+1) \in \mathcal{S}$ denote the initial (pre-action) and the next (post-action) state of the environment, respectively, from a state space $\mathcal{S}$; $\boldsymbol{a}(t) \in \mathcal{A}$ denotes an action from an action space $\mathcal{A}$ executed at time t in the state $\boldsymbol{s}(t)$. Hence, we regard the state $\boldsymbol{s}(t+1)$ as a consequence of the action $\boldsymbol{a}(t)$ and we refer to $\Delta\boldsymbol{s}_{\boldsymbol{a}}(t+1) = \boldsymbol{s}(t+1) - \boldsymbol{s}(t)$ as an effect of action $\boldsymbol{a}$ reflected in changes of some features of the environment state. It should be noted that relationships characterized by $\mathcal{T}_{\mathcal{E}}$ are entirely dependent on the discrete-time environment $\mathcal{E} \equiv (\mathcal{S}, \mathcal{A})$ in which they were observed.

By this definition, we are concerned only with short-term relationships since $\mathcal{T}_{\mathcal{E}}$ satisfies Markov property as opposed to naturally perceived causal relationships that can commonly depend on event episodes spanning for a longer duration. Additionally, we study only causal relations produced by egocentric learning (Woodward, 2011; Gärdenfors, 2006; Gärdenfors & Lombard, 2018) – i.e., this concerns only relationships observed as a product of agent's actions, not as a product of actions performed by other agents or different environmental entities. This delimitation also corresponds with our focus on two most elementary categories (C1 and C2) of robot causal learning according to Hellström (2021) stated in Section 1.1: sensorimotor self-learning and learning the consequences of an agent's own actions on objects in the environment.

Causal relationships following definitions above and exhibiting the presented properties can be deemed as sensorimotor contingencies (O'Regan & Noë, 2001; Noë, 2004) (for a brief overview of SMCs, see Section 1.1). Furthermore, from the conceptual point

of view, we understand $\mathcal{T}_\mathcal{E}$ as a low-level intuitive theory (Gerstenberg & Tenenbaum, 2017) encapsulating the accumulated knowledge of causal relationships observed in the environment $\mathcal{E}$.

Our research has three components. As stated in Introduction, in this thesis, we aim to explore possibilities of implementation of learning such defined causal relationships observed in a simulated robotic environment using forward and inverse models in a supervised learning scheme. We test and study these models in a few experiments considering a robotic arm involved in two sensorimotor tasks corresponding with C1 and C2 causal learning categories.

Moreover, we aim to investigate the possibility of extracting relationships between state and action features in a given environment $\mathcal{E}$ using SHAP methods applied to the trained forward model. Using this approach, we intend to provide the basis for the dimensional reduction of the state representations and the higher-level intuitive explainability of the environment's behaviour.

Our last goal is to combine information about the environment's behaviour from the forward and inverse models with imitation learning. In this part, we explore the usage of supervised sequence modelling using RNNs for trajectory generation and attempt to integrate learned causal relationship knowledge into it. Here, we also propose a semi-supervised training scheme leveraging the forward and inverse models for loss computation. This proposition, however, is not experimentally tested in this thesis.

For all presented experiments and analyses, we only use data obtained by observations and interactions with the simulated robotic environment. As such, we assume that robot perception is reliable and errorless and use it as ground truth for model learning and other applications. It should be acknowledged, however, that this assumption may not hold well in real-world conditions when the perception may be inaccurate or fail. Such cases are beyond the scope of this research.

Chapter 4

Methods

In this chapter, we introduce the main contribution points of our research in the form of general methods to be applied and tested in the specific experiments. For the exact implementation details regarding these methods, refer to Chapter 5.

4.1 Synthetic Data Generation

To facilitate causal learning from observations, we collected sensorimotor data in an automated manner using a simulated robotic environment using myGym toolkit (Vavrečka et al., 2021). For this purpose, for each experiment, we design a simple routine executed by an agent during the simulation, which forces the agent to explore the state-action space to the fullest extent and thus substitutes a more complex system providing artificial motivation, which could, however, be used instead as well. A routine is randomized to diversify generated observations efficiently and can be formally defined as a policy π.

Routines can have multiple forms depending on the difficulty of causal learning we try to facilitate. In the case of sensorimotor learning (C1), the robotic arm performs motor babbling and records its joint configuration $\boldsymbol{\theta} \subset \boldsymbol{s}$ and Cartesian effector position $\boldsymbol{ef} \subset \boldsymbol{s}$ before and after a random joint action. We hypothesize that from this data, the forward and inverse models should be able to learn the relationship between joint action and change in the effector position. The specific implementation of this approach can be seen in the experiment in Section 5.1.

In the case of C2 causal learning focused on learning how the agent can affect entities in the environment, we add an object into the environment and introduce a routine allowing the arm to interact with it and observe how its joint actions affect various attributes of the object. Related experiment is described in Section 5.2.

The general procedure for the synthetic data generation can be seen in Algorithm 4.1.

Algorithm 4.1: Synthetic observation data generation.

Input: Environment $\mathcal{E} \equiv (\mathcal{S}, \mathcal{A})$, routine policy π, and start state $\boldsymbol{s}(0)$

Output: Dataset of observed relations $\mathcal{O}$

for $i = 1, \ldots, N$ **do**
- Read the current state $\boldsymbol{s}(t)$
- Perform action $\boldsymbol{a}(t) \sim \pi\left[\boldsymbol{s}(t)\right]$
- Read the state $\boldsymbol{s}(t+1)$
- $\mathcal{O}_i \leftarrow [\boldsymbol{s}(t), \boldsymbol{a}(t), \boldsymbol{s}(t+1)]$

4.2 Forward and Inverse Models

We use the generated data to learn the forward and inverse models offline.

Forward Model

The FM is implemented by a feed-forward MLP (Fig. 4.1)[3] that learns the mapping

$$\text{FM}\colon\ [\boldsymbol{s}(t), \boldsymbol{a}(t)] \mapsto \hat{\boldsymbol{s}}(t+1) \tag{4.1}$$

and thus directly models causal relations defined in Eq. 3.1. Hence, as stated in Chapter 3, trained FM as well as state feature vectors $\boldsymbol{s}(t), \hat{\boldsymbol{s}}(t+1) \in \mathcal{S}$ and action vector $\boldsymbol{a}(t) \in \mathcal{A}$ are completely dependent on an environment $\mathcal{E}$ in which the relations where observed and a task during which they were observed.

Specifically, in our experiments, $\boldsymbol{\theta}(t), \boldsymbol{ef}(t) \subset \boldsymbol{s}(t)$, where $\boldsymbol{\theta}(t)$ denotes the joint configuration of the robotic arm (agent) and $\boldsymbol{ef}(t)$ is its end-effector position in 3D Cartesian space. We also represent the action vector as $\boldsymbol{a}(t) = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1)$, making our approach biologically plausible since the action is relative to a current state. This approach contrasts with common alternative action representation as an absolute target joint vector.

As state features can be diverse, the architecture of the FM contains separate output heads for different state subvectors $\hat{\boldsymbol{y}}_i \subseteq \hat{\boldsymbol{s}}(t+1)$. For each output head, the loss during the training is computed separately, and the model is optimized according to the overall loss

$$\mathcal{L}_{\text{FM}}\left[\hat{\boldsymbol{s}}(t+1), \boldsymbol{s}(t+1)\right] \triangleq \frac{1}{N} \sum_{\substack{\hat{\boldsymbol{y}}_i \subseteq \hat{\boldsymbol{s}}(t+1) \\ \boldsymbol{y}_i \subseteq \boldsymbol{s}(t+1)}} \text{MSE}\left(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i\right), \tag{4.2}$$

where N is a total number of output heads or defined state subvectors, MSE denotes the mean squared error and $\boldsymbol{y}_i$ is a ground-truth subvector corresponding to its prediction $\hat{\boldsymbol{y}}_i$.

[3] It should be noted that the dimensionalities of hidden layers are task-specific based on the dimensionalities of input and output layers. Thus, we do not list them in this general description.

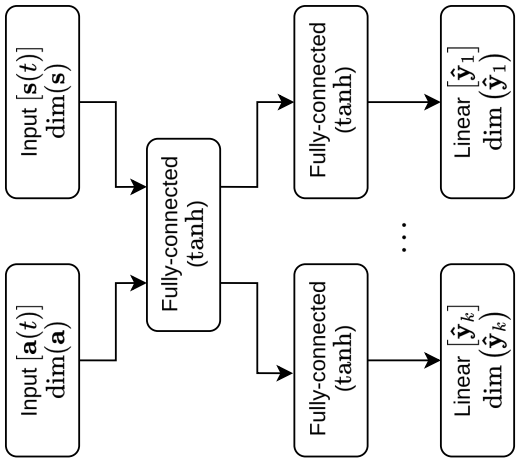

Figure 4.1: General forward model architecture.

Inverse Model

Similarly, the inverse model is implemented by a feed-forward MLP that learns the mapping

$$\text{IM}\colon\ [\boldsymbol{s}(t), \boldsymbol{s}(t+1)] \mapsto \hat{\boldsymbol{a}}(t). \tag{4.3}$$

Albeit we can assume the availability of joint configuration subvector $\boldsymbol{\theta}(t+1) \subset \boldsymbol{s}(t+1)$ during the offline training from the generated dataset, we cannot during the inference as $\boldsymbol{\theta}(t+1)$ and output $\hat{\boldsymbol{a}}(t)$ to be estimated are circular dependent.

For this reason, we propose two approaches to the inverse model construction. The monolithic architecture (Fig. 4.2) consists of an MLP taking $\boldsymbol{s}(t)$ and $\boldsymbol{s}'(t+1) = \boldsymbol{s}(t+1) \setminus \boldsymbol{\theta}(t+1)$ (i.e., the original state vector without $\boldsymbol{\theta}$ subvector) as input during both training and inference and thus completely ignores $\boldsymbol{\theta}(t+1)$ information.

However, since learning the IM mapping yields better results with $\boldsymbol{\theta}(t+1)$ available, we were looking for approaches that would leverage this component during the training from the dataset but leave it free to vary unrestricted and conform to other variables during the inference. This requirement may weakly resemble the uncontrolled manifold (UCM) hypothesis (Scholz & Schöner, 1999), which we took inspiration from. According to the UCM hypothesis, a controller (central nervous system) controls only certain variables relevant to the motor task being executed and includes others in UCM to leave them uncontrolled while trying to preserve the stability of the motor action execution.

Hence, the second approach (Fig. 4.3) relies on the composition of the base model learning the original IM mapping (Eq. 4.3) and the pre-network learning the mapping $[\boldsymbol{s}(t), \boldsymbol{s}'(t+1)] \mapsto \hat{\boldsymbol{\theta}}(t+1)$ on the generated data and pre-computing the approximate value of $\boldsymbol{\theta}(t+1)$ during the inference. This output is concatenated with the rest of the

inputs and fed into the base model.

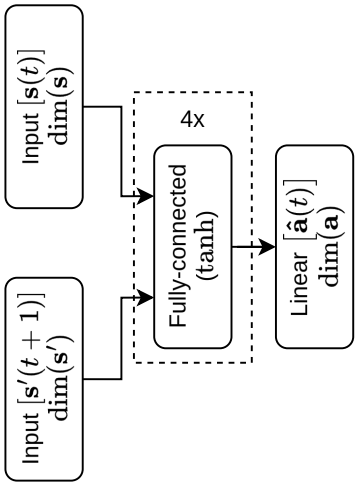

Figure 4.2: General monolithic inverse model architecture.

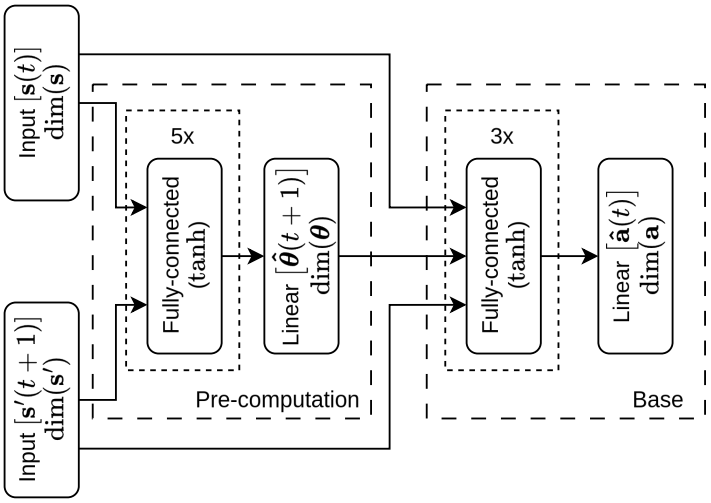

Figure 4.3: Inverse model architecture with $\boldsymbol{\theta}(t+1)$ pre-computation pre-network.

4.3 Knowledge Extraction

We mentioned in Chapter 3 that one of our aims is to explore the possibility of extracting relationships between state and action features in a given environment and a learning session. We can obtain such information by analyzing trained FM. Our primary focus is on analyzing feature importance for the model, which allows us to highlight state features that the agent's actions cannot manipulate. Such features could be then removed, hence reducing the dimensionality of the state space for the specific task and environment.

Contrasted with the similar approach by Lee et al. (2021) (for its brief overview, see Section 2.1), which we take inspiration from, we do not study causal relationships by direct systematic interaction with an environment but by using trained FM as a proxy preserving this information.

Using the learned FM, we can determine the relevance of state features in relation to action features by analyzing their importance. For this purpose, we use SHAP methods (Lundberg & Lee, 2017) (for an overview, see Section 1.3) – specifically, in the experiments, we use the DeepSHAP method only as, in our case, it is significantly faster than the model-agnostic KernelSHAP method while yielding the same results.

When analyzing the FM using the DeepSHAP method, the method measures the contribution of every input feature $x \in \boldsymbol{s}(t) \cup \boldsymbol{a}(t)$ to the prediction of every output feature $\hat{y} \in \hat{\boldsymbol{s}}(t+1)$. We can discard results for input features $x \in \boldsymbol{s}(t)$, leaving only contributions of action features to the state features $\hat{y} \in \hat{\boldsymbol{s}}(t+1)$.

Provided contributions are local, meaning they pertain to a single data instance and the output of the FM computed from it. However, these local results can be aggregated across a sample of N instances, forming a distribution of action feature contributions to the state features. Such distribution is visualizable using partial dependence plots (PDP) (Friedman, 2001) (e.g., Fig. 5.5).

From PDPs, we can determine whether there is any relationship between the selected action and the state features and properties of this relationship. A high correlation between the action feature value and its contribution to the state feature indicates a strong impact of the action feature on the state feature. However, such a relationship should not always be regarded as causal (Dillon et al., 2021).

Alternatively, compressed visualization in the form of a heat map can be created using mean absolute contribution values from the distribution (e.g., Figs. 5.2 and 5.4).

4.4 Planning

Our last contribution focuses on leveraging trained FM and IM for planning. Planning concerns the generation of a trajectory prior to its execution, and it is necessary when an agent needs to perform more complex actions consisting of multiple steps. Hence, it is required to solve more complex motor tasks.

Mental Simulation

As per the definitions of the forward and inverse models (Eqs. 4.1 and 4.3), they operate only with time-adjacent states and thus cannot be directly used for generating long-term trajectories. However, considering an initial state $\boldsymbol{s}(0)$ and a sequence of actions

$$\tau_a = [\boldsymbol{a}(0), \boldsymbol{a}(1), \ldots, \boldsymbol{a}(T-1), \boldsymbol{a}(T)]\,, \tag{4.4}$$

it is possible to perform a chained inference on the FM by recurrently generating a successor state from the previous state:

$$\hat{\boldsymbol{s}}(t) = \mathrm{FM}\left[\hat{\boldsymbol{s}}(t-1), \boldsymbol{a}(t-1)\right], \tag{4.5}$$

where $\hat{\boldsymbol{s}}(t-1)$ is a predicted state from the previous timestep and $\boldsymbol{a}(t-1) \in \tau_a$. This approach allows us to generate trajectory T steps ahead as long as the action sequence τ_a is provided. The approach is analogous to a mental simulation humans perform when planning and problem solving (Klein & Crandall, 1995; Jug et al., 2018).

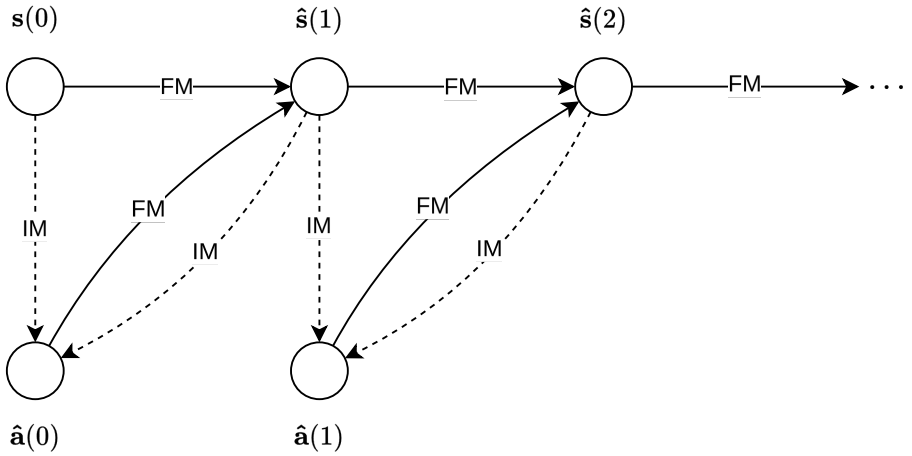

Figure 4.4: Relation between inputs and outputs of the forward (FM) and inverse (IM) model during the mental simulation. The FM processes $\boldsymbol{s}(t)$ and $\hat{\boldsymbol{a}}(t)$ (solid line) and outputs a new state $\hat{\boldsymbol{s}}(t+1)$. The IM processes two time-adjacent states $\hat{\boldsymbol{s}}(t)$ and $\hat{\boldsymbol{s}}(t+1)$ (dashed line) and outputs an action $\hat{\boldsymbol{a}}(t)$. Note the circular dependency between $\hat{\boldsymbol{a}}(t)$ and $\hat{\boldsymbol{s}}(t+1)$.

Trajectory Model

As seen above, τ_a is crucial for the trajectory generation and cannot be computed using the IM since the FM and IM outputs are circular dependent (Fig. 4.4). Hence, τ_a must be generated externally, but once it is available, it can be optimized using the FM and IM.

Our proposed approach to this problem is to generate trajectories using sequence modelling (for overview and related research, see Sections 1.5 and 2.2, respectively). Here, the agent would learn from the expert demonstrating more complex motor actions. In our case, the expert is the agent itself performing a programmed routine (similar to the approach in Section 4.1) to reduce the technical difficulty of this solution. Thus, the agent performs self-imitation learning (Oh et al., 2018) by observing itself being "guided" by the routine towards completing a task.[4] Trajectories

$$\tau^{(i)} = [\boldsymbol{s}(0), \boldsymbol{a}(0), \boldsymbol{s}(1), \boldsymbol{a}(1), \ldots, \boldsymbol{s}(T-1), \boldsymbol{a}(T-1), \boldsymbol{s}(T)], \tag{4.6}$$

where $\boldsymbol{s}(T)$ is the goal state in which the task is completed, generated this way are collected in the dataset.

The dataset of trajectories is further used for the training of the trajectory model (TM) defined as

$$\text{TM}\colon\ [\boldsymbol{s}(0), \boldsymbol{s}(T)] \mapsto \hat{\tau}_s, \tag{4.7}$$

where $\boldsymbol{s}(0)$ is the initial state and $\boldsymbol{s}(T)$ denotes the goal state in which the task is complete. The TM outputs trajectory of intermediate states

$$\hat{\tau}_s = [\hat{\boldsymbol{s}}(1), \hat{\boldsymbol{s}}(2), \ldots, \hat{\boldsymbol{s}}(T-1)] \tag{4.8}$$

that are required to transition from $\boldsymbol{s}(0)$ to $\boldsymbol{s}(T)$. Such trajectory can be subsequently converted to an action sequence using the IM:

$$\hat{\tau}_a = \big[\,\text{IM}\,[\hat{\boldsymbol{s}}(t), \hat{\boldsymbol{s}}(t+1)] \mid 0 \leq t \leq T-1\big], \tag{4.9}$$

where $\hat{\boldsymbol{s}}(t), \hat{\boldsymbol{s}}(t+1) \in \hat{\tau}_s$, and $\hat{\boldsymbol{s}}(0) \equiv \boldsymbol{s}(0)$, $\hat{\boldsymbol{s}}(T) \equiv \boldsymbol{s}(T)$.

As the TM performs sequence modelling, we implemented it as a recurrent decoder architecture (Cho et al., 2014) (Fig. 4.5) using a single or stacked RNNs – specifically, in our case, we used LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Cho et al., 2014) layers. RNN layers feed the time-distributed FM generating output state $\hat{\boldsymbol{s}}(t)$ for each timestep t.

The ground-truth and generated trajectories can have variable lengths; however, as the TM outputs constant-length trajectories, we had to ensure the same length for all

[4] However, conventional imitation learning using a secondary agent as an expert is possible as well.

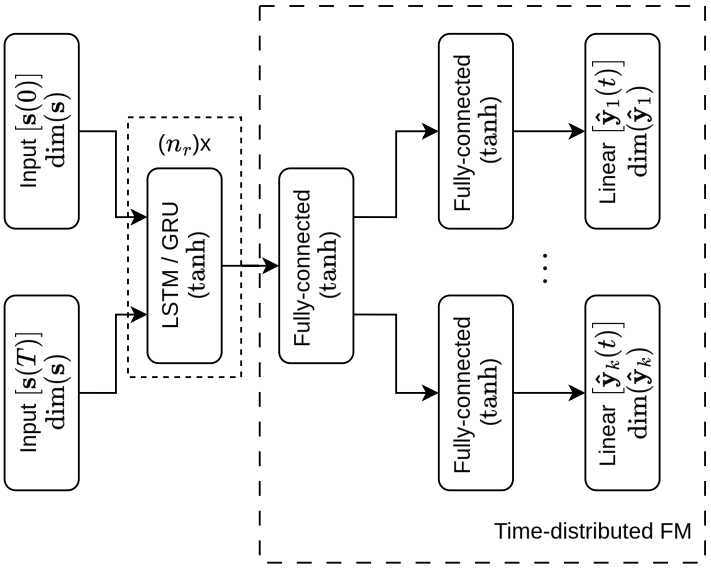

Figure 4.5: General trajectory model architecture. n_r denotes the number of recurrent layers.

trajectories. For this purpose, we edge-pad[5] the training ground-truth trajectories τ_s to a fixed length. Edge padding secures that all trajectories have the same length, and the last element of every padded trajectory τ_s is the last state of the original trajectory. Consequently, the TM will output $\hat{\tau}_s$ with $|\hat{\tau}_s| = |\tau_s|$. The generated trajectory $\hat{\tau}_s$ can then be fully converted to action sequence $\hat{\tau}_a$ using the process from Eq. 4.9. For an action $\hat{\boldsymbol{\tau}}_{\boldsymbol{a}}(i)$ inferred by the optimal TM that extends the necessary length of the successful trajectory (i.e., the action transition states of $\hat{\tau}_s$ that the TM output as padding values), it should hold that

$$\left\|\hat{\boldsymbol{\tau}}_{\boldsymbol{a}}(i)\right\|_2 \approx 0, \tag{4.10}$$

with $\left\|\hat{\boldsymbol{\tau}}_{\boldsymbol{a}}(i)\right\|_2$ denoting magnitude of the action vector.

Due to its sequential nature, the TM is trained by Truncated BPTT (Williams & Peng, 1990). The basic TM is optimized according to the element-wise error of the generated trajectory against the ground-truth trajectory using the loss function of the FM (Eq. 4.2):

$$\mathcal{L}_{\mathrm{TM}}^{(\mathrm{I})}\left(\hat{\tau}_s, \tau_s\right) \triangleq \frac{1}{|\hat{\tau}_s|} \sum_{i=1}^{|\hat{\tau}_s|} \mathcal{L}_{\mathrm{FM}}\left[\hat{\boldsymbol{\tau}}_{\boldsymbol{s}}(i), \boldsymbol{\tau}_{\boldsymbol{s}}(i)\right], \tag{4.11}$$

where $|\hat{\tau}_s|$ is the length of the generated trajectory $\hat{\tau}_s$ (identical to $|\tau_s|$ thanks to the padding, as mentioned above) and $\mathcal{L}_{\mathrm{FM}}\left[\hat{\boldsymbol{\tau}}_{\boldsymbol{s}}(i), \boldsymbol{\tau}_{\boldsymbol{s}}(i)\right]$ denotes the error of the i-th predicted state in the trajectory $\hat{\boldsymbol{\tau}}_{\boldsymbol{s}}(i)$ against the i-th ground-truth state $\boldsymbol{\tau}_{\boldsymbol{s}}(i)$ according to the FM loss function.

[5]Edge padding consists of repeatedly appending the last element of a sequence until the sequence has a desired length.

FM/IM-aided Training

Although the TM can be optimized based on the element-wise error (Eq. 4.11), this approach does not evaluate possible trajectories based on their capacity to reach goal states. For this reason, we propose a semi-supervised training scheme leveraging the FM and IM. The idea is that we can let the TM infer the preliminary trajectory $\hat{\tau}_s$ and using ground-truth $\boldsymbol{s}(0)$ and the optimal FM and IM as reference points, we can compute the error of each generated state $\hat{\boldsymbol{s}}(t) \in \hat{\tau}_s$ and the error expected goal state of such trajectory.

Note that the approach present in this subsection, as stated in Chapter 3, is a mere proposition, and as such, it has not been experimentally tested. However, we present it here as a potential point of continuation of this research, and it should be subjected to testing as part of future work. The analysis of this approach is provided in Discussion of Section 5.3.

During the first epoch of the TM training, we propose employing the element-wise loss function $\mathcal{L}_{\text{TM}}^{(\text{I})}$ (Eq. 4.11) facilitating the supervised learning. The purpose of calculating the loss this way only in the first epoch is to initialize the TM to output trajectories of states from the similar distribution as was the IM training distribution.[6] Since, in the next step of the scheme, the IM consumes pairs of states to output actions, states produced by TM with randomly initialized parameters would cause the IM to perform out-of-distribution predictions resulting in inaccuracies.

From the second epoch onward, the error of each generated trajectory is computed by traversing the trajectory in the time direction and evaluating each predicted state $\hat{\boldsymbol{s}}(t) \in \hat{\tau}_s$ in multiple steps (Fig. 4.6). Here, we no longer use provided ground-truth trajectories τ_s, and instead, the correcting signal is computed by the scheme itself; thus, the learning is self-supervised (or, in a sense, unsupervised).

First, the first intermediate state $\hat{\boldsymbol{s}}(1)$ is paired with the ground-truth initial state $\boldsymbol{s}(0)$. The state pair is input to the IM inferring action $\hat{\boldsymbol{a}}(0)$ responsible for the transition between $\boldsymbol{s}(0)$ and $\hat{\boldsymbol{s}}(1)$. Using the FM, we then predict the consequence of the execution of $\hat{\boldsymbol{a}}(0)$ in $\boldsymbol{s}(0)$. This step has the purpose of producing a rectified state $\tilde{\boldsymbol{s}}(1)$ with respect to the ground-truth initial state $\boldsymbol{s}(0)$. Assuming optimal IM and FM, the FM in this step essentially verifies whether the predicted $\hat{\boldsymbol{s}}(1)$ is achievable by the action $\hat{\boldsymbol{a}}(0)$ estimated on the basis of $\boldsymbol{s}(0)$. If $\hat{\boldsymbol{s}}(1)$ is predicted accurately, discrepancy of $\tilde{\boldsymbol{s}}(1)$ against it should be minimal. Thus, we can measure the prediction error of $\hat{\boldsymbol{s}}(1)$ using the FM loss function (Eq. 4.2) as $\mathcal{L}_{\text{FM}}\left[\hat{\boldsymbol{s}}(1), \tilde{\boldsymbol{s}}(1)\right]$.

For the next state $\hat{\boldsymbol{s}}(t) \in \hat{\tau}_s$, the process is similar; however, instead of using $\boldsymbol{s}(0)$ as the IM and FM input, we propose using the previous rectified state $\tilde{\boldsymbol{s}}(t-1)$. This way,

[6] This should hold as the training dataset for the TM is generated under the same conditions as the dataset for the IM (and FM).

we can propagate the "correcting signal" in the form of more accurate rectified states originating in the ground-truth $\boldsymbol{s}(0)$ throughout the whole $\hat{\tau}_s$. The last generated state $\tilde{\boldsymbol{s}}(T)$ should then correspond to the ground-truth goal state $\boldsymbol{s}(T)$. The rectified state trajectory generation procedure is formally described in Algorithm 4.2.

With the generated rectified trajectory $\tilde{\tau}_s$, we can now compute the prediction error of $\hat{\tau}_s$ as

$$\mathcal{L}_{\text{TM}}^{(\text{II})}\left(\hat{\tau}_s, \tilde{\tau}_s\right) \triangleq \frac{1}{|\hat{\tau}_s| + 1}\left(\sum_{i=1}^{|\hat{\tau}_s|} \mathcal{L}_{\text{FM}}\left[\hat{\boldsymbol{\tau}}_{\boldsymbol{s}}(i), \tilde{\boldsymbol{\tau}}_{\boldsymbol{s}}(i)\right] + \mathcal{L}_{\text{FM}}\left[\tilde{\boldsymbol{s}}(T), \boldsymbol{s}(T)\right]\right). \tag{4.12}$$

$\mathcal{L}_{\text{TM}}^{(\text{II})}$ measures the average magnitude of discrepancy of predicted trajectory state $\hat{\boldsymbol{\tau}}_{\boldsymbol{s}}(i)$ against the hypothetically more accurate corresponding rectified state $\tilde{\boldsymbol{\tau}}_{\boldsymbol{s}}(i)$ and also the magnitude of discrepancy between the last rectified state $\tilde{\boldsymbol{s}}(T)$ and the corresponding ground-truth goal state $\boldsymbol{s}(T)$.

We suppose the discrepancy measured here is conceptually analogous to the one humans experience when their observation of reality is not aligned with the expectation produced by their predictive model (Clark, 2013). As den Ouden et al. (2012) remark, such error, among other purposes, is vital to improving humans' internal representations as "the brain's primary objective is to infer the causes of its sensory input by reducing surprise, in order to allow it to successfully predict and interact with the world".

Algorithm 4.2: Generation of the rectified state trajectory $\tilde{\tau}_s$. $\hat{\boldsymbol{s}}'(t)$ denotes $\hat{\boldsymbol{s}}(t)$ without its joint configuration subvector $\hat{\boldsymbol{\theta}}(t)$ as per the implementation of the IM described in Section 4.2.

Input: Ground-truth initial state $\boldsymbol{s}(0)$ and goal state $\boldsymbol{s}(T)$

Output: Trajectory of rectified intermediate states $\tilde{\tau}_s$ and the last rectified state $\tilde{\boldsymbol{s}}(T)$

$\hat{\tau}_s \leftarrow \text{TM}\left[\boldsymbol{s}(0), \boldsymbol{s}(T)\right]$
$\tilde{\boldsymbol{s}}(0) \leftarrow \boldsymbol{s}(0)$
for $t = 1, \ldots, T$ **do**
  $\hat{\boldsymbol{s}}(t) \leftarrow \hat{\tau}_s(t)$
  $\hat{\boldsymbol{s}}'(t) \leftarrow \hat{\boldsymbol{s}}(t) \setminus \hat{\boldsymbol{\theta}}(t),\ \hat{\boldsymbol{\theta}}(t) \subset \hat{\boldsymbol{s}}(t)$
  $\hat{\boldsymbol{a}}(t-1) \leftarrow \text{IM}\left[\tilde{\boldsymbol{s}}(t-1), \hat{\boldsymbol{s}}'(t)\right]$
  $\tilde{\boldsymbol{s}}(t) \leftarrow \text{FM}\left[\tilde{\boldsymbol{s}}(t-1), \hat{\boldsymbol{a}}(t-1)\right])$
  if $t < T$ **then**
    $\tilde{\boldsymbol{\tau}}_{\boldsymbol{s}}(t) \leftarrow \tilde{\boldsymbol{s}}(t)$

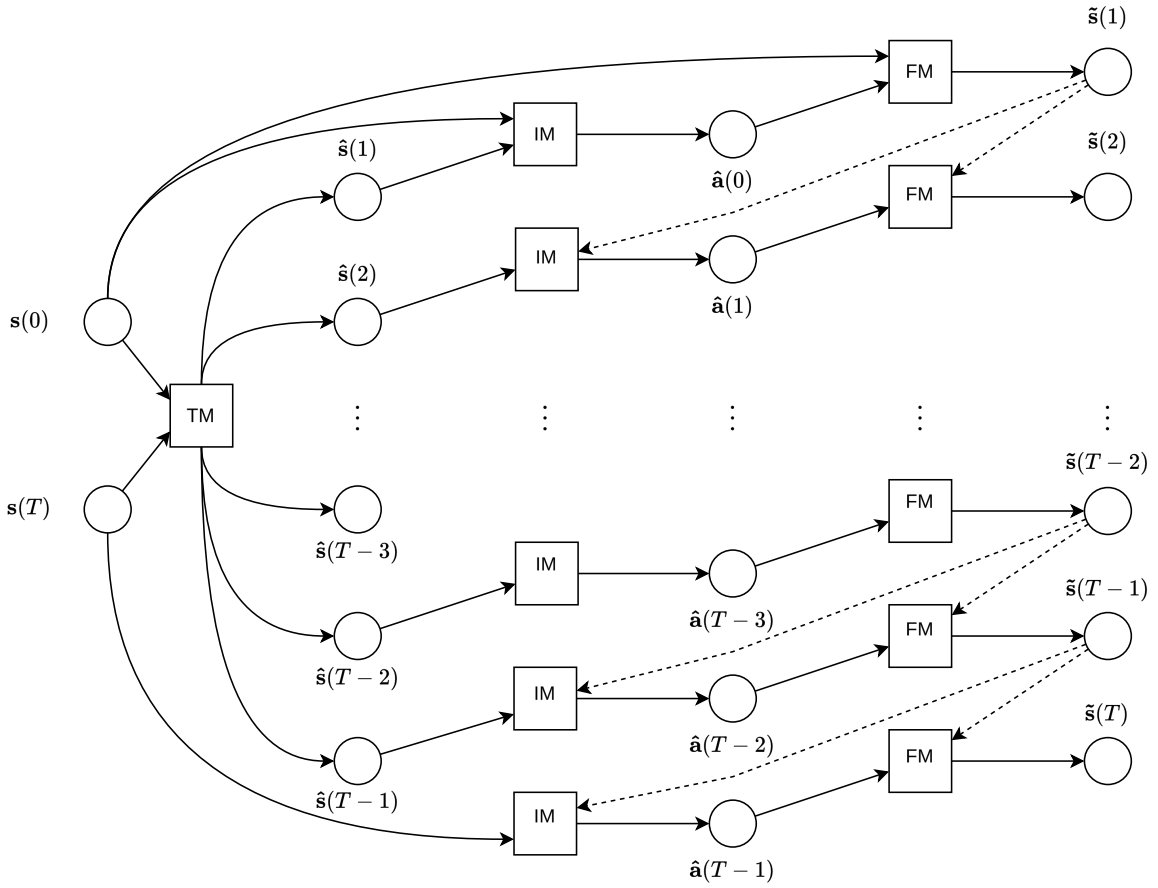

Figure 4.6: TM/IM-aided generation of the rectified intermediate state trajectory. Dashed lines designate the flow of the "correcting signal" from the previous timesteps.

Chapter 5

Experiments and Results

We applied the methods described in Chapter 4 in three experiments. The first two (Sections 5.1 and 5.2) are related to categories C1 and C2 of learning causal relationships (Hellström, 2021). The structure of these experiments is similar. The first step consists of data generation in a simulated robotic environment (Section 4.1) provided by myGym toolkit (Vavrečka et al., 2021). The generated data is subsequently used for training the FM and IM (Section 4.2). These models are further analyzed in the second experiment to reveal causal relations. In Experiment 3 (Section 5.3), we experiment with the planning methods applied to a pick-and-place task with multiple objects in an environment.

Each experiment has a discussion section that analyzes the obtained results and proposes related future work. The discussion of Experiment 3 also analyzes the proposed FM/IM-aided training scheme for the TM that could not be experimentally tested in this research.

5.1 Learning Kinematics

This experiment is focused on sensorimotor learning (C1). Here, we used the Franka Emika Panda robotic arm with a gripper and 7 degrees of freedom (DoF) that performed motor babbling to observe relationships between joint actions and changes in the position of the arm's end-effector.

Environment

The simulation ran in 500,000 steps. In each step t, a joint configuration $\boldsymbol{\theta}(t) \in \mathbb{R}^8$ is sampled from the normal distribution with limits according to Table 5.1. The gripper (θ_7)[7] is fixed open during the simulation as it is not a part of the experiment. A motor command is executed in 10 substeps before proceeding to the next simulation step,

[7]The joints are numbered from zero.

allowing a longer execution time, which results in the actual action being more similar to the planned one. After the action execution, only the resulting configuration and 3D Cartesian effector position $\boldsymbol{ef}(t) = \left[ef_x, ef_y, ef_z\right]$ are recorded, both composing the state vector $\boldsymbol{s}(t) = [\boldsymbol{\theta}(t), \boldsymbol{ef}(t)]$.

Table 5.1: Joint motion range in Panda arm used for Experiment 1. $q_{\min}$ and $q_{\max}$ denote minimal and maximal angle for each joint, respectively.

	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
$q_{\min}$ [rad]	-2.967	-1.833	-2.967	-3.142	-2.967	-0.087	-2.967	0.0
$q_{\max}$ [rad]	2.967	1.833	2.967	0.0	2.967	3.822	2.967	0.0

Models

The FM for this experiment uses two separate output heads – one for joint configuration $\hat{\boldsymbol{\theta}}(t+1)$ prediction and the other for the prediction of the effector position $\hat{\boldsymbol{ef}}(t+1)$. Each head computes a separate mean squared error used as a loss. The FM is trained according to the overall loss $\mathcal{L}_{\text{FM}}$ as per Eq. 4.2. For training the FM, we used Adam optimizer (Kingma & Ba, 2014) with an initial learning rate $\eta = 10^{-3}$ for 60 epochs. The model was evaluated using 5-fold cross-validation with average mean absolute error (MAE) for the effector position and joint configuration outputs of 2 mm and 1.2×10^{-3} rad, respectively.

For the IM, we tried both architectural approaches proposed in Section 4.2. All model variants of both approaches use MSE as a loss function per the IM's description. For the monolithic approach, we trained a base IM using AdamW optimizer (Loshchilov & Hutter, 2017). We experiment with two variants of this model, differing in the unit's activation function at the hidden layer: hyperbolic tangent (tanh) or ReLU. However, the differences in resulting performance were negligible.

We also tested the base IM by inputting $\boldsymbol{\theta}(t+1) = \mathbf{0}$ or sampling $\theta_i(t+1)$ from the dataset as an alternative to approximating $\boldsymbol{\theta}(t+1)$ by the pre-network. The sampling method and inputting zero vector resulted in an MAE of 0.208 rad and 0.457 rad, respectively.

Finally, we experimented with the pre-computation approach, which uses a base IM trained first using Adam optimizer and the feature-generating pre-network trained separately. After the training, the pre-network was put in front of the base model, forming the assembly used for inference. Both approaches' training hyperparameters and final results can be seen in Table 5.2.

Table 5.2: Hyperparameters and resulting MAE of approaches to inverse model construction for kinematics data. Results were obtained using 5-fold cross-validation. η and λ denote the initial learning rate and initial weight decay of Adam and AdamW optimizers, respectively.

Approach	Model	Epochs	Optimizer	MAE [rad]
Monolithic	Base	1,000	AdamW($\eta = 10^{-3}$, $\lambda = 0.004$)	0.0120
Pre-computation	Base	100	Adam($\eta = 10^{-3}$)	9.594×10^{-4}
	Pre-network	4,000	AdamW($\eta = 10^{-3}$, $\lambda = 0.004$)	0.0126
	Assembly	N/A	N/A	0.0139

Mental Simulation

We subjected the trained FM to chained inference (mental simulation) as described in Section 4.4. First, we generated 30,000 ground-truth trajectories by motor babbling in the simulation. In each such trajectory

$$\tau^{(i)} = [\boldsymbol{s}(0), \boldsymbol{a}(0), \boldsymbol{s}(1), \boldsymbol{a}(1), \ldots, \boldsymbol{s}(9), \boldsymbol{a}(10), \boldsymbol{s}(10)] \tag{5.1}$$

there is an initial state $\boldsymbol{s}(0)$ and ten actions, each applied on the respective previous state. We let the FM generate a trajectory $\hat{\tau}^{(i)}$ by repeatedly querying the model on the previously generated state and the respective action $\boldsymbol{a}(t)$ from the ground-truth trajectory $\tau^{(i)}$ without the model being given ground-truth state reference. The first prediction uses ground-truth state $\boldsymbol{s}(0)$.

For k-th step ahead with $1 \leq k \leq 10$, its error in the mental simulation is computed as

$$E(k) \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathrm{MAE}\left[\hat{\boldsymbol{y}}^{(i)}(k), \boldsymbol{y}^{(i)}(k)\right] \tag{5.2}$$

with $N = 30,000$ being the number of testing samples, and either $\hat{\boldsymbol{y}}^{(i)}(k) \equiv \hat{\boldsymbol{\theta}}^{(i)}(k) \subset \hat{\boldsymbol{\tau}}^{(i)}(k)$ or $\hat{\boldsymbol{y}}^{(i)}(k) \equiv \hat{\boldsymbol{ef}}^{(i)}(k) \subset \hat{\boldsymbol{\tau}}^{(i)}(k)$ as we measure the error for joint configuration and effector position predictions separately. The same goes for the ground-truth output component $\boldsymbol{y}^{(i)}(k)$. For the results, see Fig. 5.1.

Discussion

The results show that the FM and IM can learn the relation between the joint configuration and the effector position with a high degree of accuracy. Using this method, the agent is able to learn its embodiment in a more organic and flexible way, as opposed to the classical methods of modelling kinematics. Additionally, we suppose

that this method could be applied to a range of different robots with more DoF and even beyond robotic arms; however, to support this claim, additional study on the scalability of this method may be needed.

Regarding the IM, both the monolithic and pre-computation approaches demonstrate similar performance, with the monolithic IM having a slightly lower error. Thus, it is not evidenced that the pre-computation approach provides significant improvement. However, this may not be the case for other experiments with more complex action and state spaces where simple monolithic architecture may not be sufficient.

As shown in the results of the mental simulation experiment (Fig. 5.1), the joint configuration prediction error exhibits linear growth while the effector position error grows in an approximate semi-logarithmic log-linear trend. This result is an improvement over our expectation of both errors growing exponentially.

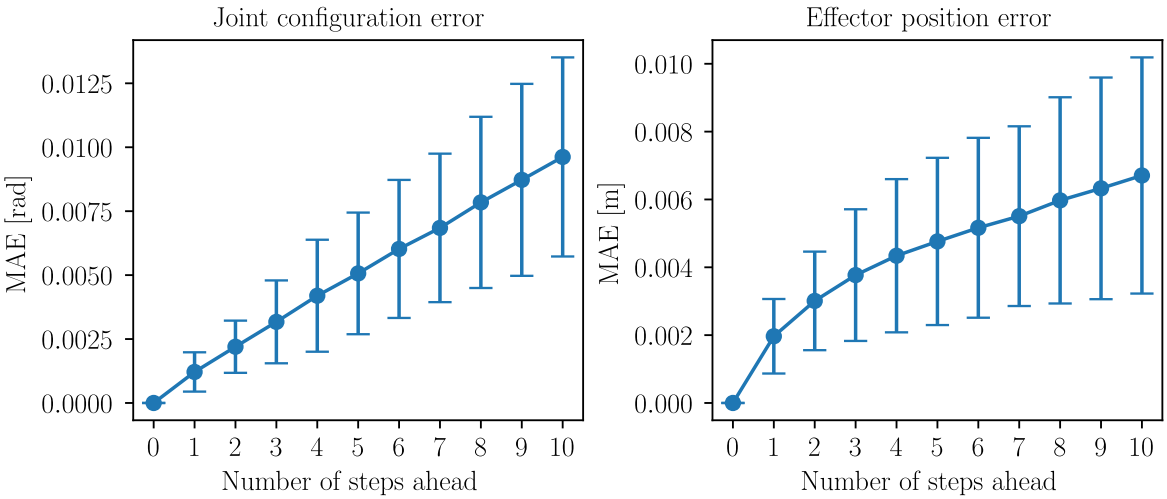

Figure 5.1: Average mean absolute error and its standard deviation of joint configuration and effector position prediction by the forward model during mental simulation 10 steps ahead.

5.2 Simple Intuitive Physics

This experiment aims to evaluate the proposed methods on the C2 causal learning task proposed by Hellström (2021), described as "Learning about how the robot affects the world". In this experiment, we test how well the FM and IM can learn relationships in the environment with an object the robot interacts with and what information can be extracted from the trained FM using the knowledge extraction method proposed in Section 4.3.

Environment

The experiment utilizes the KUKA LBR iiwa robotic arm with 7 DoF and a togglable magnetic endpoint as an agent. The task consisted of the arm randomly switching the magnet on and off. If it was not holding anything, the arm navigated to the magnetized cube lying on the table, picked it up, and randomly manoeuvred with it in the space for a random duration. After that, the magnet was turned off, the cube was released, and the arm performed motor babbling empty-handed for a random duration. The setting of this experiment can be seen in Fig. 5.3.

The aim of this experiment was to let the agent learn the simple physics of the cube in the environment as well as the agent's own kinematics. Knowledge gained by the agent in this experiment can be thus understood as a superset of knowledge from the previous kinematics experiment (Section 5.1). Furthermore, we wanted to verify whether the model architectures specified in Section 4.2 can efficiently work with state spaces of higher dimensionality.

The data-generating simulation ran in 4,000 episodes, lasting 500 iterations each. After each iteration t, we recorded the final joint configuration $\boldsymbol{\theta}(t) \in \mathbb{R}^7$, the end-effector position and rotation as a 6D pose $\boldsymbol{ef}(t) = \left[ef_x, ef_y, ef_z, ef_{rx}, ef_{ry}, ef_{rz}\right]$, object information (its position, rotation and RGB colour) $\boldsymbol{o}(t) = [o_x, o_y, o_z, o_{rx}, o_{ry}, o_{rz}, o_R, o_G, o_B]$, and the magnet state $mgt(t) \in \{0, 1\}$. As we do not use the seventh joint of the arm, $\theta_6 = 0.0$ throughout the experiment. Object colour features were added as control variables, randomized at the start of each episode and not manipulated during it.

Models

As in the previous experiment, we trained an FM and a monolithic IM on the generated data. The FM uses separate output heads to predict object position, object rotation, colour, joint configuration, and effector position and rotation. Each head computes a separate MSE, which is used as loss $\mathcal{L}_{\text{FM}}$ (Eq. 4.2). The FM was trained for 100 epochs using Adam optimizer (Kingma & Ba, 2014) with the initial learning rate $\eta = 10^{-3}$. For the final results of the FM obtained by the evaluation using 5-fold cross-validation, see Table 5.3.

We applied only a monolithic approach to the inverse model construction in this experiment. The model is optimized according to the average of separate MSEs for joint $\hat{a}_i$ and magnet action $\hat{a}_{mgt} \in \{-1, +1\}$ prediction. The optimization was facilitated by AdamW optimizer (Loshchilov & Hutter, 2017) with the initial learning rate $\eta = 10^{-3}$ and initial weight decay $\lambda = 0.004$ for 1,000 epochs with the final MAE of joint action prediction 0.0077 rad and of magnet action prediction 4.56×10^{-4}.

Table 5.3: Errors of respective output heads of the forward model for intuitive physics data. Results were obtained using 5-fold cross-validation.

Output head	MAE	Output head	MAE
Object position	0.0089 m	Effector position	0.008 m
Object rotation	0.0721 rad	Effector rotation	0.0625 rad
Object colour	0.004	Joint configuration	0.0084 rad
		Magnet state	1.3×10^{-4}

Knowledge Extraction

The trained FM is further analyzed using methods proposed in Section 4.3. Here, we experimented primarily with the DeepSHAP method (Lundberg & Lee, 2017); however, we tried to compare it with the slower model-agnostic KernelSHAP method. Both methods analyzed the trained FM on a sample of 200 observations from the generated dataset. The analysis with KernelSHAP took approximately 40 minutes on an AMD Ryzen 9 5900X CPU with Nvidia RTX 3060 GPU performing the FM inference. The analysis by DeepSHAP method with the same setup took ca. 22 seconds. Both methods yielded the same result, so we continued further with DeepSHAP only.

The resulting full global contribution heat map output by the DeepSHAP method for this experiment is shown in Fig. 5.4. The y-axis denotes state and action features in time t (input of the FM), and the x-axis contains state features in time $t+1$ (output of the FM). The colour of each square corresponds to the magnitude of contributions of the input features to the output features averaged across the selected sample of 200 observations.

Since our focus is mainly on determining how action features affect the state features in the next timestep, we cropped the full contribution heat map to the action portion and rescaled the colour map to correspond with values in the segment. This heat map can be seen in Fig. 5.2.

Feature importance and dependencies can also be studied using feature contribution distributions visualized as partial dependence plots. Fig. 5.5 shows a selected sample of PDPs generated from feature contribution data. Averaged absolute contributions across the distributions correspond to the respective values in Fig. 5.2.

Discussion

Overall, the FM and IM were able to learn the kinematics of the agent with the same-order error as in Experiment 1. Comparing the results of the FM, the prediction error is four times higher for the effector position and seven times higher for the joint configuration. Apart from differences in the training distributions for this experiment

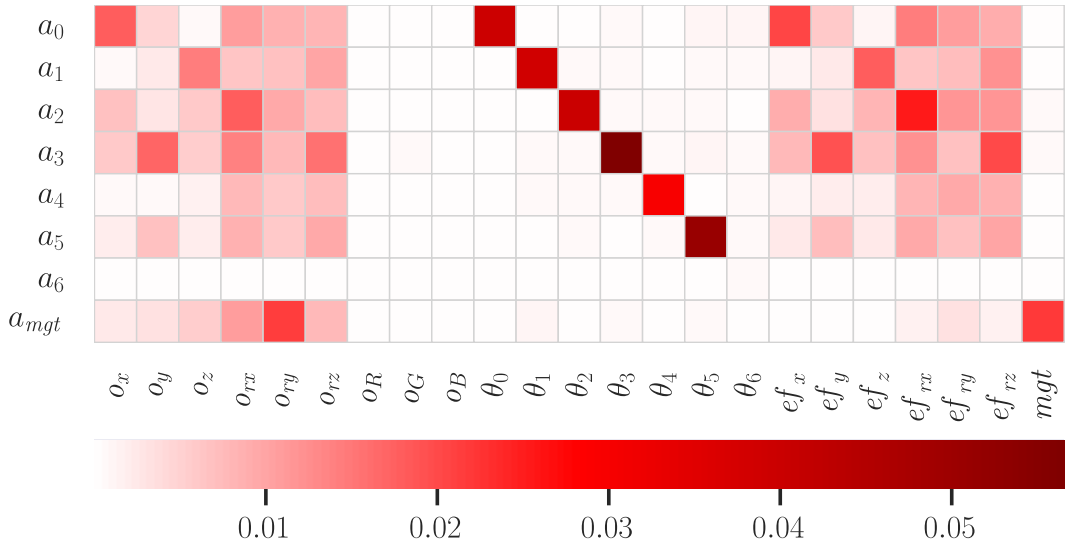

Figure 5.2: Action portion of the contribution heat map generated by DeepSHAP method on the forward model showing magnitude of contribution of specific actions to output features. This heat map is a cropped version of Fig. 5.4 with a rescaled color map.

and Experiment 1, the higher prediction error might also be caused by the optimizer's inability to further optimize the more complex architecture of the FM in this experiment. The general solution for this problem might be performing an exhaustive grid search to find the best combination of hyperparameters suitable for the specific task the FM tries to learn. For the IM, the overall joint error is smaller than in Experiment 1 (0.0077 rad against 0.012 rad). In summary, we verified that the FM and IM can learn environment-related relations beyond those body-related (i.e., kinematics of the agent) with a relatively small error (Table 5.3).

Regarding knowledge extraction, several observations can be made from the analysis results. The Fig. 5.2 shows, for instance, that joint 6 is not used in the sampled observation data as no action feature correlates with the feature θ_6. In addition, the object's colour is irrelevant in this experiment as no action can affect it and thus could be removed (or ignored) from the state space. On the other hand, all action features (except a_6) affect most object features. This low-level knowledge could be useful for causal analysis at higher levels.

Furthermore, in the presented sample[8] of PDPs in Fig. 5.5, it can be observed that action feature a_0 (movement of joint 0) has a substantial impact on state features θ_0 (state of joint 0) and o_x (object position on the global x-axis). These observations intuitively correspond with the fact that the change of the state of joint 0 occurs in the observations always when there is a movement of that joint, and as joint 0 is a base joint rotating the arm around its vertical axis, the object's x-axis position commonly

[8] All the generated PDPs can be found in Appendix A

changes if the cube is attached to the magnetic endpoint. The second relationship is illustrated in Fig. 5.3.

Moreover, from the bottom-left PDP, we can see that action feature a_2 does not significantly correlate with the contribution to state feature o_B (a blue colour component of the object) with contribution centred around 0.0 indicating a_2 does not profoundly impact o_B. Finally, the action of the magnetic endpoint a_{mgt} is prevalently null as the state of the magnet does not often change between iterations. However, when it does, it significantly contributes to o_z (object position on the z-axis) since turning the magnet on ($a_{mgt} = +1$) or off ($a_{mgt} = -1$) in this experiment is followed by lifting the object in the air or dropping it on the table.

It should be acknowledged that these results stem from the correlations between the features observed in the data, and although they may capture causal relationships, they should not always be treated as such.

These findings are obviously trivial as the experiment is relatively simple. However, as the FM and IM are scalable to larger state spaces, this method should be capable of revealing relationships even in more complex cases where the relations are not easily detectable.

In relation to this experiment, it could be interesting to implement the FM and IM using Kolmogorov-Arnold networks (Liu et al., 2024) instead of conventional MLPs. As mentioned in Section 1.2, the authors of KANs claim they possess several interesting properties, including interpretability. This property could thus simplify knowledge extraction as we could remove the currently used SHAP method from the pipeline, and we would be able to perform the analysis using the FM alone.

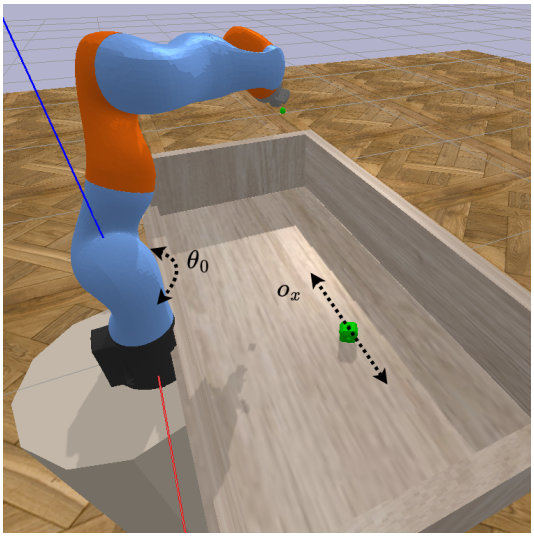

Figure 5.3: Setting of Experiment 2 with illustrated relationship between the change of the rotation of the joint 0 (θ_0) and movement of the object along the global x-axis (o_x)

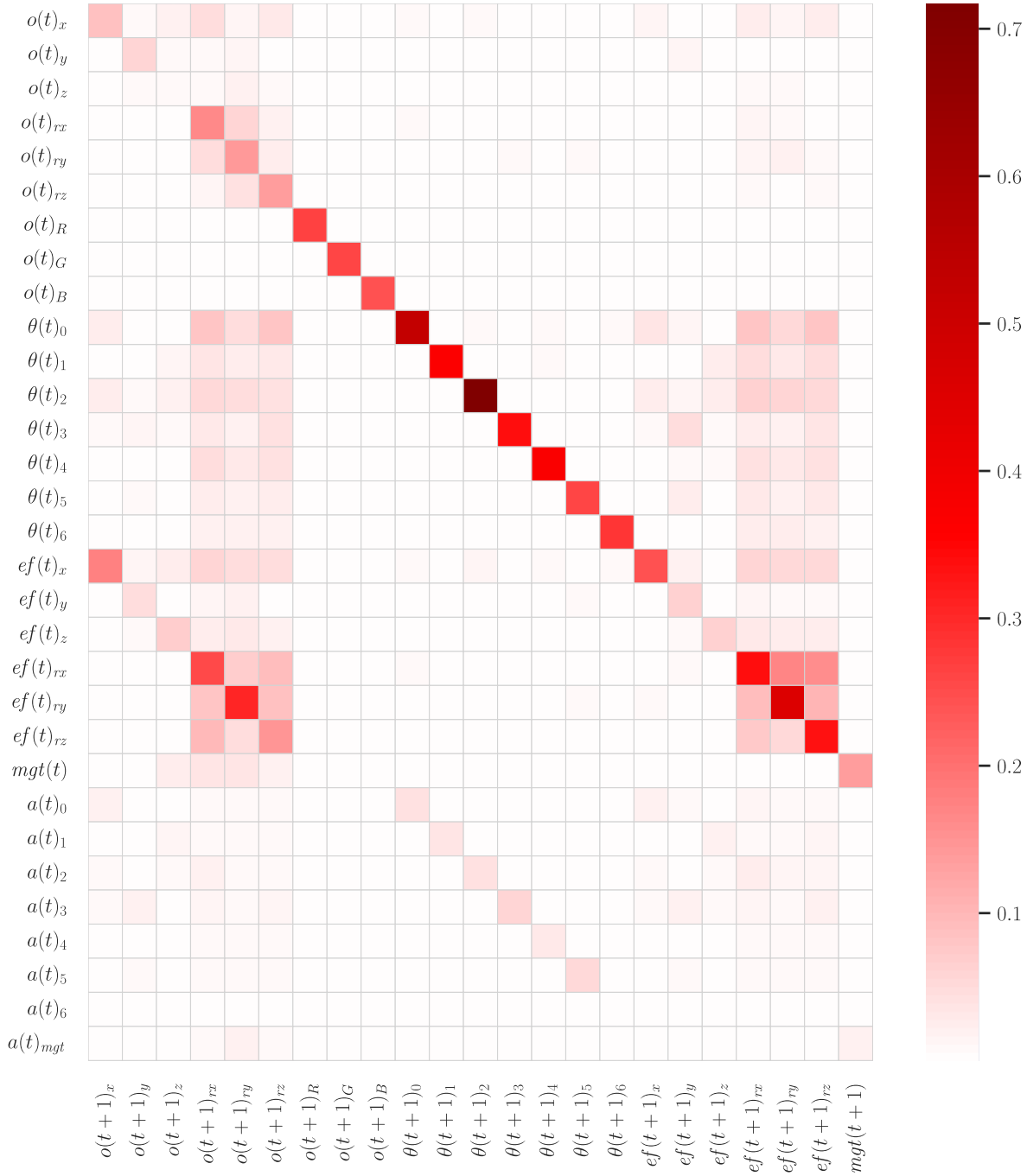

Figure 5.4: Full contribution heat map generated by DeepSHAP method on the forward model showing contribution magnitude of every state and action feature in time t (y-axis) to the features of output state $\boldsymbol{s}(t+1)$ (x-axis).

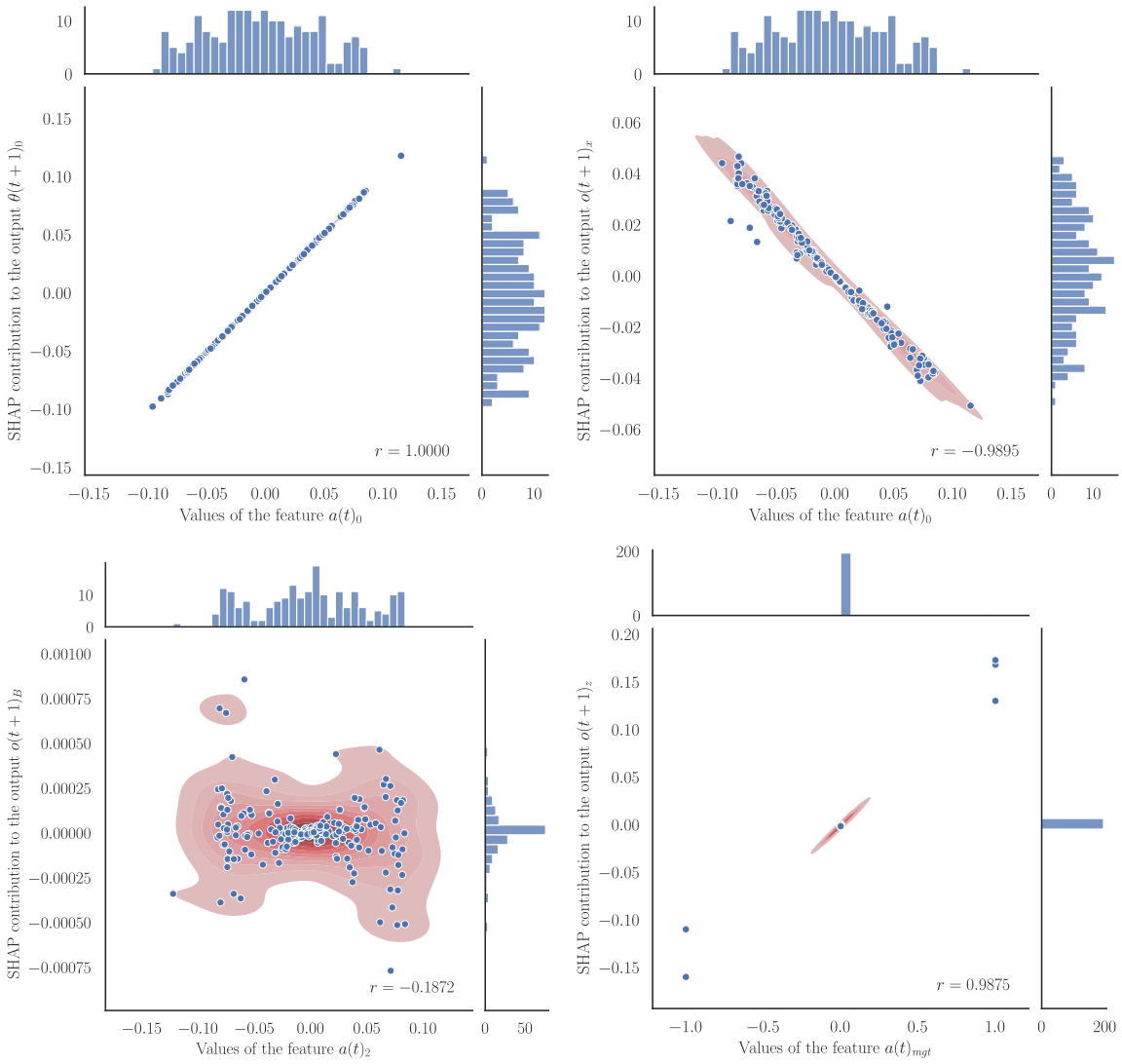

Figure 5.5: A sample of partial dependence plots generated by DeepSHAP method applied to the forward model showing correlation between a value of a specific action component and its contribution to an output state feature. All the generated PDPs can be found in Appendix A.

5.3 Planning

This experiment aims to test approaches for trajectory generation presented in Section 4.4. Specifically, we evaluate the trajectory model optimized using supervised learning on trajectories generated by self-imitation learning. As already mentioned, we do not evaluate the proposed FM/IM-aided training scheme, but we analyze it in Discussion.

Environment

For this experiment, we use a similar setup as in Experiment 2 (Section 5.2). We used the same KUKA LBR iiwa robotic arm with a magnetic endpoint. The environment contained two cubes lying on the table, and the arm was performing a pick-and-place task consisting of picking one cube and either placing it in a random position on the table or on top of the other cube. The purpose of this was to generate trajectories of picking and placing an object (as part of the self-imitation learning) for training the TM and to generate observations of causal relationships as was done in the previous experiments. Apart from kinematics and the behaviour of an object in the environment, these observations also captured interactions between the cubes, such as the top cube falling over when placed improperly.

The simulation generating data ran in 12,000 episodes of at least 13 and at most 30 iterations. Each episode began with the empty-handed arm approaching a cube and ended when the cube was placed at its designated position. After the end of each episode, a new environment was constructed with randomly placed objects and a randomly selected target position.

After each iteration t, same as in Experiment 2, we recorded the final joint configuration $\boldsymbol{\theta}(t)$, the end-effector 6D pose $\boldsymbol{ef}(t)$ and the magnet state $mgt(t)$. The object information vector was recorded for each cube and, in this experiment, consisted only of the object's 6D pose $\boldsymbol{o}_i(t) = [o_x, o_y, o_z, o_{rx}, o_{ry}, o_{rz}]$. The state vector was assembled per each timestep as $\boldsymbol{s}(t) = [\boldsymbol{o}_1(t), \boldsymbol{o}_2(t), \boldsymbol{\theta}(t), \boldsymbol{ef}(t), mgt(t)]$ and each episode's trajectory was constructed as a sequence of such states and action vectors in between.

Models

First, the FM and the monolithic IM were trained, the same as in Experiment 2. While we do not directly use these models in this experiment, they could be used for additional post-processing of the generated trajectory (e.g., to generate an action sequence from the state trajectory). Thus, we wanted to test how well these models perform for a task such as this.

The FM used separate output heads to compute both objects' positions, rotations,

the arm's joint configuration, and effector position and rotation. The FM was trained for 60 epochs using Adam optimizer (Kingma & Ba, 2014) with the initial learning rate $\eta = 10^{-3}$ according to the standard FM error $\mathcal{L}_{\text{FM}}$ (Eq. 4.2). The final results of the FM training for this experiment evaluated using 5-fold cross-validation are listed in Table 5.4.

The monolithic IM was trained for 100 epochs using AdamW optimizer (Loshchilov & Hutter, 2017) with the initial learning rate $\eta = 10^{-3}$ and the initial weight decay $\lambda = 0.004$. The IM was optimized according to separate MSE for joint and magnet action prediction as in the previous experiment. The final MAE of joint action prediction was 0.0018 rad and of magnet action prediction 3.15×10^{-5}.

Table 5.4: Errors of respective output heads of the forward model for planning experiment. Results were obtained using 5-fold cross-validation.

Output head	MAE	Output head	MAE
Object 1 position	0.0035 m	Effector position	0.0023 m
Object 1 rotation	0.0111 rad	Effector rotation	0.0089 rad
Object 2 position	0.0036 m	Joint configuration	0.0037 rad
Object 2 rotation	0.0138 rad	Magnet state	0.0045

Next, the TM was trained on the ground-truth trajectories generated in the simulation. The dataset of 12,000 trajectories was split into training and testing subsets of 9,600 and 2,400 trajectories, respectively (80:20 size ratio).

Considering the TM's general architecture (Fig. 4.5), as part of a small comparative study, we experimented with the number of recurrent layers of the decoder $n_r \in \{1, 2\}$, number of units in each recurrent layer and the immediately connected fully-connected layer $d_r \in \{100, 200\}$, and the type of the recurrent units (LSTM (Hochreiter & Schmidhuber, 1997) or GRU (Cho et al., 2014)). Each variant was trained on the training subset for 1,000 epochs using Adam optimizer with $\eta = 10^{-3}$ according to the $\mathcal{L}_{\text{TM}}^{(\text{I})}$ loss (Eq. 4.11) based on the errors of separate output heads of the appended time-distributed FM. The output heads are the same as in the case of the FM for this experiment.

All the TM variants were tested on the testing subset. Table 5.5 lists the variants with their respective hyperparameters and the resulting error according to $\mathcal{L}_{\text{TM}}^{(\text{I})}$. The mean absolute error per output head of each tested variant measured during the same testing procedure can be seen in Table 5.6.

Table 5.5: Comparison of the trajectory model variants. n_r denotes the number of recurrent layers in the decoder, and d_r is the number of units in them. Overall $\mathcal{L}_{\text{TM}}^{(\text{I})}$ denotes the error of the respective variant on the testing subset computed using Eq. 4.11.

Variant	Recurrent layer	n_r	d_r	Overall $\mathcal{L}_{\text{TM}}^{(\text{I})}$
#1	LSTM	1	100	0.0135
#2	LSTM	1	200	**0.0112**
#3	LSTM	2	100	0.013
#4	LSTM	2	200	0.0118
#5	GRU	1	100	0.017
#6	GRU	1	200	0.0204
#7	GRU	2	100	0.0137
#8	GRU	2	200	0.0118

Table 5.6: Comparison of the errors of respective output heads for the trajectory model variants listed in Table 5.5.

	MAE per variant							
Output head	#1	#2	#3	#4	#5	#6	#7	#8
Obj. 1 position [m]	0.0077	**0.0071**	0.0082	0.0085	0.0086	0.0095	0.0074	0.0072
Obj. 1 rotation [rad]	0.0151	0.013	0.0137	**0.0117**	0.0161	0.0255	0.0143	0.013
Obj. 2 position [m]	0.0079	0.0076	0.0083	0.0083	0.0083	0.0131	0.0083	**0.0073**
Obj. 2 rotation [rad]	0.0152	0.0112	0.0148	0.0121	0.0149	0.0116	0.0155	**0.0105**
Eff. position [m]	0.0072	0.0068	0.008	0.0089	0.0078	0.0082	0.0072	**0.0063**
Eff. rotation [rad]	0.0106	0.0094	0.0108	0.0089	0.0118	0.012	0.0102	**0.0082**
Joint config. [rad]	0.0097	0.008	0.01	0.0089	0.0101	0.0099	0.0096	**0.0075**
Magnet state	0.0156	0.0156	0.016	**0.0131**	0.0215	0.0181	0.0194	0.0155

Discussion

The FM and IM achieved significantly better performance (Table 5.4) than in Experiment 2 and comparable performance as in Experiment 1. This result is probably caused by using differently structured data for the training of the FM and IM. While in the previous two experiments, training data were fully or partially generated by the motor babbling, in this case, the observations originated only from the arm performing the pick-and-place task without any action perturbations. Hence, the observations in the dataset are limited, and the data contains simpler and straightforward patterns, whose learning is generally easier. As both models were evaluated by cross-validation,

the testing data is from the same distribution and thus does not pose such a challenge. We hypothesize that evaluating these models on a broader, more general dataset would produce inferior results.

This is an example of the differences between specialized and general FM and IM. Specialized models trained on a dataset with a more limited domain may perform better for a dedicated task than general, more robust models. However, applying the specialized models to more diverse tasks may naturally result in poor performance.

Regarding the TM, we can conclude that the model was able to learn to generate accurate state trajectories for solving the posed pick-and-place task. As mentioned in Section 4.4, such state trajectory can be input to the trained IM, generating an action sequence. Considering the accuracy of the IM in this experiment and the accuracy of generated state trajectories, we assume that the generated action sequences should be accurate as well, especially considering the contained actions are small-scale and, thus, there is not ample space for error. We did not, however, qualitatively test the planning deployed in the environment and focused instead on quantitative evaluation of the approach, so we left the qualitative study for future work. Additionally, the planning approach could be evaluated on multiple diverse tasks with varying average trajectory lengths, as this single experiment does not investigate this aspect.

The performed small-scale hyperparameter search for the best architectural variant of the TM did not produce a significantly superior model. In Table 5.6, the variant #8 consisting of the 2-layer 200-unit GRU decoder dominates in performance for most of the output heads. However, the differences in the error are relatively insignificant when compared with other model variants. As these experiments primarily serve as a proof of concept, we did not perform extensive hyperparameter search and optimization of the models. Nevertheless, from the differences in performance between the present variants, we assume further scaling of the model will not result in a notable improvement.

In Section 4.4, we proposed a training scheme for the TM consisting of the IM and FM rectifying each state of the generated trajectory based on the previous rectified or ground-truth state and computing the prediction error as an average magnitude of discrepancy between the generated and rectified state. As stated in Chapter 3 and Section 4.4, we did not experimentally test this approach as part of this thesis, and we leave it as a significant point for future work. Despite that, we are going to provide a brief analysis of this approach's advantages and potential problems.

The presented training scheme is semi-supervised, as only the first optimization epoch is supervised based on the ground-truth trajectories. Thus, it could require less data as it is only used to initialize the model. If this approach works well, as a prospective future work, the data efficiency of this method could be investigated, as could the possibility of a few-shot learning.

As mentioned in the description of the scheme, the primary purpose of it is to

facilitate trajectory generation learning only with respect to a goal state a generated trajectory should achieve. This should, hypothetically, give space to generate trajectories that are out of the training data distribution and dissimilar to those observed by the agent during the imitation learning process yet still valid as they achieve a desired goal state. One problem that may arise is that due to the conventional supervised learning in the first epoch, the TM is biased in generating trajectories from the distribution, which is similar to the training one. This is not necessarily undesired, as generated trajectories cannot be completely random (as explained in the scheme description). However, the introduced bias should not be profound as it would give very little space for the optimization by this scheme in the second step. The amount of data used during the first supervised step of the scheme needed for the balanced bias should be studied in an empirical study.

One of the obvious problems of this approach is the disproportionality of the amount of input data against the output. While this method aims at correcting an output trajectory and providing feedback on the quality of the prediction, it heavily relies on the ground-truth initial state $\boldsymbol{s}(0)$ used for initializing the correcting signal, which should be then propagated throughout the whole trajectory, contributing to the rectification of each of its states. Nonetheless, however this method may be efficient and the IM and FM in the pipeline accurate, it is safe to assume that such signal quality would eventually decay with a growing length of the processed trajectory. We deem this is a similar problem as in the case of learning long-term dependencies in RNNs (Bengio et al., 1994) or, more generally, vanishing gradient problem (Pascanu et al., 2013), where the magnitude of a gradient forming an error signal gradually decreases as it is backpropagated through the neural network. In future work, the performance of the proposed method could be studied as a function of generated trajectory length to establish how feasible this method is for optimizing the TM for tasks requiring trajectories of various lengths.

Another problem we identify is that this training scheme is relatively time-expensive as it requires performing the inference on two neural models (FM and IM) for each state in the generated trajectory as opposed to a simple element-wise loss (e.g., Eq. 4.11).

Conclusion

This thesis explored learning causal relationships observed in a simulated robotic environment using the forward and inverse models. In the methods, we proposed several approaches to implementing the FM and IM, taking inspiration from mechanisms observed in human cognition. Further, we explored the means of analysis of these models to extract useful information about the environment and the task during which the causal relations were observed. In our third contribution point, we reused some of the introduced methods in constructing a system, performing planning to solve simple manipulation tasks.

Most of the proposed methods were tested in a series of experiments with a simulated robotic arm acting as an agent. The first experiment concerned learning the agent's kinematics by observing the causal relationship between the movement of its joints and the position of its end-effector. Here, we confirmed that the FM and IM can learn such a relation with high accuracy, and thus, these artificial models are capable of sensorimotor learning.

In the second experiment concerning learning simple intuitive physics by interaction with an object, we demonstrated that these models could further learn environment-related relationships beyond those body-related ones. Moreover, we showed that we can visualise the learned relations using the proposed knowledge extraction methods. We proposed that information obtained this way can be utilized to determine relevant state features, serving as a basis for dimensional reduction of the state space. Our method can be applied to scenarios that are much more complex and much harder for humans to understand, and thus, it can be an important tool for extracting causal knowledge.

In the final third experiment, we tested how well the proposed trajectory model can learn to generate trajectories guiding the agent towards completing a pick-and-place task. Here, we quantitatively evaluated different configurations of the TM and showed that the proposed approach can accurately generate trajectories based on only the initial and goal states. Related to this, in the methods, we propose an alternative semi-supervised training scheme for the TM, which could be, hypothetically, able to optimize the model based on the initial and goal state alone without a large set of ground-truth trajectories provided. However, the verification of the viability of this

method is a subject of future work.

Overall, in the future, apart from testing the training scheme mentioned, we would like to evaluate the proposed methods using a more complex and diverse set of experiments to confirm that these methods are applicable in general. Furthermore, we would like to modify these approaches for learning higher-level causal relationships beyond category 2 of the classification by Hellström (2021). For more detailed future work propositions, refer to the discussion sections of each experiment in Chapter 5.

To conclude, we believe we provided a relatively large set of methods primarily leveraging learning causal relationships applicable to different tasks in robotics and that, in the future, these methods could contribute to the development of commonsense understanding in artificial intelligence and robotics.

Bibliography

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, *6*(5), 679–684. https://doi.org/10.1512/iumj.1957.6.56038

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., & Bruna, J. (2022). When does return-conditioned supervised learning work for offline reinforcement learning? *Advances in Neural Information Processing Systems*, *35*, 1542–1553.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., & Mordatch, I. (2021). Decision Transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, *34*, 15084–15097.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. https://doi.org/10.3115/v1/D14-1179

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. https://doi.org/10.48550/arxiv.1412.3555

Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, *36*(3), 181–204. https://doi.org/10.1017/s0140525x12000477

de Lange, F. P., Heilbron, M., & Kok, P. (2018). How do expectations shape perception? *Trends in Cognitive Sciences*, *22*(9), 764–779. https://doi.org/10.1016/j.tics.2018.06.002

Dearden, A. M., & Demiris, Y. (2005). Learning forward models for robots. *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 1440–1445.

den Ouden, H. E. M., Kok, P., & de Lange, F. P. (2012). How prediction errors shape perception, attention, and motivation. *Frontiers in Psychology*, *3*. https://doi.org/10.3389/fpsyg.2012.00548

Diehl, M., & Ramirez-Amaro, K. (2023). A causal-based approach to explain, predict and prevent failures in robotic tasks. *Robotics and Autonomous Systems*, *162*, 104376. https://doi.org/10.1016/j.robot.2023.104376

Dillon, E., LaRiviere, J., Lundberg, S., Roth, J., & Syrgkanis, V. (2021). *Be Careful When Interpreting Predictive Models in Search of Causal Insights*. Retrieved March 30, 2024, from https://towardsdatascience.com/be-careful-when-interpreting-predictive-models-in-search-of-causal-insights-e68626e664b6

Dogge, M., Custers, R., & Aarts, H. (2019). Moving forward: On the limits of motor-based forward models. *Trends in Cognitive Sciences*, *23*(9), 743–753. https://doi.org/10.1016/j.tics.2019.06.008

Emmons, S., Eysenbach, B., Kostrikov, I., & Levine, S. (2021). RvS: What is essential for offline RL via supervised learning? https://doi.org/10.48550/arxiv.2112.10751

Fisher, R. A. (1925). *Statistical Methods for Research Workers*. Oliver and Boyd.

Francis, B. A., & Wonham, W. M. (1976). The internal model principle of control theory. *Automatica*, *12*(5), 457–465. https://doi.org/10.1016/0005-1098(76)90006-6

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Furuta, H., Matsuo, Y., & Gu, S. S. (2021). Generalized decision transformer for offline hindsight information matching. https://doi.org/10.48550/arxiv.2111.10364

Gärdenfors, P. (2006). *How Homo Became Sapiens: On the Evolution of Thinking*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198528517.001.0001

Gärdenfors, P., & Lombard, M. (2018). Causal cognition, force dynamics and early hunting technologies. *Frontiers in Psychology*, *9*. https://doi.org/10.3389/fpsyg.2018.00087

Gerstenberg, T., & Tenenbaum, J. B. (2017). Intuitive theories. In M. R. Waldmann (Ed.), *The Oxford Handbook of Causal Reasoning* (pp. 515–548). Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199399550.013.28

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., & Levine, S. (2019). Learning to reach goals via iterated supervised learning. https://doi.org/10.48550/arxiv.1912.06088

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/

Goodman, N. D., Ullman, T. D., & Tenenbaum, J. B. (2011). Learning a theory of causality. *Psychological Review*, *118*(1), 110–119. https://doi.org/10.1037/a0021336

Hellström, T. (2021). The relevance of causation in robotics: A review, categorization, and analysis. *Paladyn, Journal of Behavioral Robotics*, *12*(1), 238–255. https://doi.org/10.1515/pjbr-2021-0017

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In J. F. Kolen & S. C. Kremer (Eds.), *A Field Guide to Dynamical Recurrent Networks*. IEEE Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Janner, M., Li, Q., & Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems*, *34*, 1273–1286.

Jug, J., Kolenik, T., Ofner, A., & Farkaš, I. (2018). Computational model of enactive visuospatial mental imagery using saccadic perceptual actions. *Cognitive Systems Research*, *49*, 157–177. https://doi.org/10.1016/j.cogsys.2018.01.005

Jurafsky, D., & Martin, J. H. (2024, February). *Speech and Language Processing* (3rd draft ed.). https://web.stanford.edu/~jurafsky/slp3/

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. https://doi.org/10.48550/arxiv.1412.6980

Klein, G., & Crandall, B. W. (1995). The role of mental simulation in problem solving and decision making. In P. A. Hancock, J. M. Flach, J. Caird, & K. J. Vicente (Eds.), *Local Applications of the Ecological Approach to Human-Machine Systems* (1st ed., pp. 324–358). CRC Press. https://doi.org/10.1201/9780203748749-11

Kolmogorov, A. N. (1961). On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *American Mathematical Society Translations: Series 2*, *17*, 369–373. https://doi.org/10.1090/trans2/017/12

Kotseruba, I., & Tsotsos, J. K. (2018). 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, *53*(1), 17–94. https://doi.org/10.1007/s10462-018-9646-y

Lake, B. M. (2014). *Towards More Human-like Concept Learning in Machines: Compositionality, Causality, and Learning-to-learn* [PhD thesis]. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/95856

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338. https://doi.org/10.1126/science.aab3050

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2016). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*. https://doi.org/10.1017/s0140525x16001837

Lee, T. E., Zhao, J. A., Sawhney, A. S., Girdhar, S., & Kroemer, O. (2021). Causal reasoning in simulation for structure and transfer learning of robot manipulation policies. *2021 IEEE International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/icra48506.2021.9561439

Lee, T. E., Vats, S., Girdhar, S., & Kroemer, O. (2023). SCALE: Causal learning and discovery of robot manipulation skills using simulation. *Proceedings of the 7th Conference on Robot Learning*, *229*, 2229–2256.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., & Tegmark, M. (2024). KAN: Kolmogorov-Arnold networks. https://doi.org/10.48550/arxiv.2404.19756

Lombard, M., & Gärdenfors, P. (2017). Tracking the evolution of causal cognition in humans. *Journal of Anthropological Sciences*, *95*, 219–234. https://doi.org/10.4436/JASS.95006

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. https://doi.org/10.48550/arxiv.1711.05101

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, *30*, 4768–4777.

Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., & Martín-Martín, R. (2021). What matters in learning from offline human demonstrations for robot manipulation. https://doi.org/10.48550/arxiv.2108.03298

Matsui, Y., & Matsui, T. (2001). NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, *263*(1–2), 305–310. https://doi.org/10.1016/s0304-3975(00)00251-6

Maye, A., & Engel, A. K. (2012). Using sensorimotor contingencies for prediction and action planning. *From Animals to Animats 12*, 106–116. https://doi.org/10.1007/978-3-642-33093-3_11

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1988). The appeal of parallel distributed processing. In *Readings in Cognitive Science* (pp. 52–72). Elsevier. https://doi.org/10.1016/b978-1-4832-1446-7.50010-8

McClelland, J. L., Rumelhart, D. E., & the PDP Research Group. (1987). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Vol. 2: Psychological and Biological Models). The MIT Press.

Miall, R. C., & Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural Networks*, *9*(8), 1265–1279. https://doi.org/10.1016/s0893-6080(96)00035-4

Minsky, M., & Papert, S. A. (2017). *Perceptrons: An Introduction to Computational Geometry* (S. A. Papert & L. Bottou, Eds.; 2017 ed.). The MIT Press. https://doi.org/10.7551/mitpress/11301.001.0001

Nguyen-Tuong, D., & Peters, J. (2011). Model learning for robot control: A survey. *Cognitive Processing*, *12*(4), 319–340. https://doi.org/10.1007/s10339-011-0404-1

Noë, A. (2004). *Action in Perception* (1st ed.). MIT Press.

Oh, J., Guo, Y., Singh, S., & Lee, H. (2018). Self-imitation learning. *Proceedings of the 35th International Conference on Machine Learning*, *80*, 3878–3887.

O'Regan, J. K., & Noë, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, *24*(5), 939–973. https://doi.org/10.1017/s0140525x01000115

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*, *28*(3), 1310–1318.

Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. *Proceedings of the 7th Conference of the Cognitive Science Society*, 15–17.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144. https://doi.org/10.1145/2939672.2939778

Rosen, R. (2012). Anticipatory systems. In *Anticipatory Systems: Philosophical, Mathematical, and Methodological Foundations* (2nd ed., pp. 313–370, Vol. 1). Springer New York. https://doi.org/10.1007/978-1-4614-1269-4_6

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Schölkopf, B. (2022). Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl* (1st ed., pp. 765–804). Association for Computing Machinery. https://doi.org/10.1145/3501714.3501755

Scholz, J. P., & Schöner, G. (1999). The uncontrolled manifold concept: Identifying control variables for a functional task. *Experimental Brain Research*, *126*(3), 289–306. https://doi.org/10.1007/s002210050738

Shapley, L. S. (1953). A value for n-person games. In H. W. Kuhn & A. W. Tucker (Eds.), *Contributions to the Theory of Games II* (pp. 307–317). Princeton University Press. https://doi.org/10.1515/9781400881970-018

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. *Proceedings of the 34th International Conference on Machine Learning*, *70*, 3145–3153.

Sonar, A., Pacelli, V., & Majumdar, A. (2021). Invariant policy optimization: Towards stronger generalization in reinforcement learning. *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, *144*, 21–33.

Sontakke, S. A., Mehrjou, A., Itti, L., & Schölkopf, B. (2021). Causal curiosity: RL agents discovering self-supervised experiments for causal representation learning. *Proceedings of the 38th International Conference on Machine Learning*, *139*, 9848–9858.

Sperry, R. W. (1950). Neural basis of the spontaneous optokinetic response produced by visual inversion. *Journal of Comparative and Physiological Psychology*, *43*(6), 482–489. https://doi.org/10.1037/h0055479

Stocking, K. C., Gopnik, A., & Tomlin, C. (2022). From robot learning to robot understanding: Leveraging causal graphical models for robotics. *Proceedings of the 5th Conference on Robot Learning*, *164*, 1776–1781.

Tenenbaum, J. B. (1998). Bayesian modeling of human concept learning. *Advances in Neural Information Processing Systems*, *11*.

Tenenbaum, J. B. (1999). *A Bayesian Framework for Concept Learning* [PhD thesis]. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/16714

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Vavrečka, M., Sokovnin, N., Mejdrechová, M., & Šejnová, G. (2021). MyGym: Modular toolkit for visuomotor robotic tasks. *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 279–283. https://doi.org/10.1109/ictai52525.2021.00046

von Holst, E., & Mittelstaedt, H. (1950). Das Reafferenzprinzip: Wechselwirkungen zwischen Zentralnervensystem und Peripherie. *Naturwissenschaften*, *37*(20), 464–476. https://doi.org/10.1007/bf00622503

Wang, Z., Xiao, X., Xu, Z., Zhu, Y., & Stone, P. (2022). Causal dynamics learning for task-independent state abstraction. *Proceedings of the 39th International Conference on Machine Learning*, *162*, 23151–23180.

Wen, M., Kuba, J., Lin, R., Zhang, W., Wen, Y., Wang, J., & Yang, Y. (2022). Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, *35*, 16509–16521.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, *1*(4), 339–356. https://doi.org/10.1016/0893-6080(88)90007-x

Williams, R. J., & Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, *2*(4), 490–501. https://doi.org/10.1162/neco.1990.2.4.490

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*(7–8), 1317–1329. https://doi.org/10.1016/s0893-6080(98)00066-5

Wolpert, D. M., & Flanagan, J. R. (2001). Motor prediction. *Current Biology*, *11*(18), R729–R732. https://doi.org/10.1016/s0960-9822(01)00432-8

Woodward, J. (2011). A philosopher looks at tool use and causal understanding. In T. McCormack, C. Hoerl, & S. Butterfill (Eds.), *Tool Use and Causal Cognition* (pp. 18–50). Oxford University Press.

Yamaoka, F., Kanda, T., Ishiguro, H., & Hagita, N. (2007). How contingent should a lifelike robot be? The relationship between contingency and complexity. *Connection Science*, *19*(2), 143–162. https://doi.org/10.1080/09540090701371519

Zare, M., Kebria, P. M., Khosravi, A., & Nahavandi, S. (2023). A survey of imitation learning: Algorithms, recent developments, and challenges. https://doi.org/10.48550/arxiv.2309.02473

Zhang, K., Schölkopf, B., Spirtes, P., & Glymour, C. (2017). Learning causality and causality-related learning: Some recent progress. *National Science Review*, *5*(1), 26–29. https://doi.org/10.1093/nsr/nwx137

Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(5), 726–742. https://doi.org/10.1109/tetci.2021.3100641

Zhu, Y., Gao, T., Fan, L., Huang, S., Edmonds, M., Liu, H., Gao, F., Zhang, C., Qi, S., Wu, Y. N., Tenenbaum, J. B., & Zhu, S.-C. (2020). Dark, beyond deep: A paradigm shift to cognitive AI with humanlike common sense. *Engineering*, *6*(3), 310–345. https://doi.org/10.1016/j.eng.2020.01.011

Appendix A

Contents of the Electronic Attachment

The electronic attachment of this thesis contains the source codes of the implementations of the proposed methods and their evaluation in the experiments. It also contains additional plots generated from the experiments and other results. For a detailed overview of the structure of this attachment, refer to `README.md`.