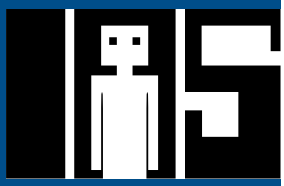
Learning a library of Physical Interactions for Social Robots

Bachelor thesis by Martina Gassen
Date of submission: November 15, 2021

1. Review: Prof. Dr. Jan Peters
2. Review: Vignesh Prasad M.Sc
3. Review: Dr. Ing Dorothea Koert
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT



Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Martina Gassen, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, November 15, 2021

Martina Gassen

Abstract

Social robots can be used to assist in various social environments such as customer services, education, caring facilities and more. Physical interaction with humans is a crucial part of all of those possible scenarios. With a vast field of application and possible contact to many people with varying social habits, a generalized system that can differentiate and react to a set of different interactions would be required. All of this should feel natural to human participants and should happen in a timely manner. In this work, we propose the Interaction Recognition and Generation (ReGen) Pipeline, a three-step pipeline composed of a Multi-task learning (MTL) Neural Network (NN) for classification and dimensionality reduction, whose output is used to gate and to efficiently and timely condition a set of Interaction Movement Primitives (MPs). We argue that MTL is a useful addition for improving the quality of both tasks as they are closely related and thus can profit from shared information. Additionally, we explore the applicability of segmenting complex movements into simple segments that can be reused in novel interactions, to ensure modularity. We evaluate this approach in a generalized scenario for humanoid robots that can be teleoperated using human joints.
Our results show that segmenting is a useful preprocessing step on modular movements. Furthermore, we re-establish the timely benefits of using a lower number of degrees of freedom (DOFs) for conditioning Bayesian Interaction Primitives (BIPs) and discuss drawbacks to keep and mind and points that can still be improved when encoding movements for MPs.

Zusammenfassung

Soziale Roboter können als Assistenz im Kundenservice, in Bildungs- und Pflegeeinrichtungen und vielen mehr verwendet werden. Die physikalische Interaktion mit Menschen ist ein wichtiger Bestandteil in allen diesen Szenarios. Mit einem breitem Anwendungsgebiet und der Möglichkeit zum Kontakt mit vielen Menschen mit variierenden sozialen Gewohnheiten, wäre ein generalisierbares System nötig, das in der Lage ist, verschiedene Interaktionen zu erkennen und entsprechend zu reagieren. All das sollte sich für die betreffenden Personen natürlich anfühlen und in Echtzeit umgesetzt werden. In dieser Arbeit schlagen wir die Anwendung der Interaction Recognition and Generation (ReGen) Pipeline; einer drei-stufigen Pipeline bestehend aus einem Multi-task learning (MTL) Neural Network (NN) zur Klassifizierung und zur Reduzierung der Dimensionalität der Eingabe, dessen Ausgabe genutzt wird, um ein Set aus Interaction Movement Primitives (MPs) zu steuern und zeitlich effizient zu konditionieren. Wir argumentieren, dass MTL eine sinnvolle Ergänzung ist, um die Qualität beider Aufgaben zu steigern, da diese eng miteinander verbunden sind und so von geteilten Informationen profitieren können. Zusätzlich untersuchen wir die Anwendbarkeit davon, komplexe Bewegungsabläufe in einfachere Segmente zu unterteilen, die in neuen Interaktionen wiederverwendet werden können und so Modularität gewähren sollen. Wir evaluieren diesen Ansatz in einem allgemeinen Szenario für humanoide Roboter, die anhand von menschlichen Gelenken ferngesteuert werden können.
Unsere Ergebnisse zeigen, dass Segmentierung ein sinnvoller Vorverarbeitungsschritt für modulare Bewegungen ist. Darüber hinaus zeigen wir noch einmal die zeitlichen Vorteile von eine reduzierten Menge an degrees of freedom (DOFs) für die Konditionierung von Bayesian Interaction Primitives (BIPs) und diskutieren Nachteile die bei der Anwendung von Kodierungstaktiken beachtet werden sollten und Punkte die dabei immer noch verbesserungswürdig sind.

Contents

1. Introduction

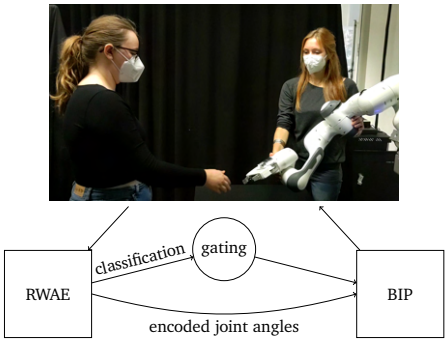
1.1. Motivation

While every interaction between two or more social actors is bound to entail some form of communication – a lot of which is influenced by nonverbal cues such as gestures, body language or physical touch [24] – the social factor of interactions can vary strongly. There are interactions that are goal-oriented and practical in nature and on the flip-side there are interactions, such as greeting customs (e.g. handshakes) or celebratory actions (e.g. high fives) that are purely social. Such social rituals are suspected to serve multiple functions including: signifying group affiliation, demonstrating commitment to a group – which can increase trust in cooperative scenarios – and increasing group cohesion [26, 30]. These can be crucial factors in establishing trust and in making the experience of a social exchange feel good and natural to people and therefore is something to pay attention to when designing social robots for assisting in social roles such as customer services, education or caring facilities (See Figure 1.1a). However, adapting these social customs in Human-Robot-Interaction (HRI)-scenarios can be hard for several reasons.

Firstly, robot movements can amplify a feeling of uncanniness in comparison with static robots [20], which is undesirable and a reason why their performance is especially important here. This performance is affected by several aspects, such as adaptability to the partners' movements and timely reactions, which we focus on in this work.
Secondly, there is not just one go-to interaction that can be used in every social scenario, as social rituals are highly dependent on context and culture [27]. Furthermore every human interaction is subject to change as time goes by. Therefore, having a library of interactions to choose from, which can be adapted to the need of the current situation, is crucial for robots that are used in different situations over long periods of time.

All of this should be applicable in realistic scenarios, so the robot has to recognize the initiated interaction only from observing its partner, without further knowledge about its

(a)



(b)

Figure 1.1.: (a) shows an exemplary image [1] of a PEPPER Robot interacting with an elderly person. (b) shows the rough pipeline of our proposed framework. We first learn to encode and to classify the observations of the human using a RWAE and then use this information to choose the correct BIP which is in turn conditioned with the encoded observation, resulting in the controls for the robot.

surroundings. Additionally, it should be possible to make the aforementioned observations easily and without extensive preparation by the interacting human.

With these preliminaries in mind, we propose the Interaction Recognition and Generation (ReGen) Pipeline (as outlined in Figure 1.1b); a pipelined approach that is composed of an MTL Neural Network (NN), for classification and dimensionality reduction whose output is used to gate and to condition a set of Interaction MPs. Additionally, we explore a modular view of interactions, where each interaction is a sequence of simple segments that can be used in multiple interactions. This is done to ensure stable behavior in variations of known interactions and to be able to react to novel interactions that are composed of known parts.

[1] Image taken from `https://uebermorgen.haz.de/2020/03/roboter-im-pflegeeinsatz/`. Last accessed on October 31, 2021

1.2. Outline

In **Chapter 2** we first summarize the main concepts used in this work – namely Long Short-Term Memory (LSTM), Variational Auto-Encoders (VAEs), Variational Recurrent Neural Networks (VRNNs) and Bayesian Interaction Primitives (BIPs) – and discuss some approaches used in previous works.
We then provide a formal problem statement and elaborate on the structure of our proposed Interaction ReGen Pipeline in **Chapter 3**. Specifically, we will elaborate on design choices and adaptations of the classification and encoding done by the MTL NN, the gating unit and the BIPs.
In **Chapter 4** we outline the experimental setups the ReGen Pipeline is tested on and present the associated results.
Finally, in **Chapter 5**, we discuss those results and outline possible starting points for future work.
The **Appendix A** contains additional detail on the hyper-parameter optimization process that was done in preparation to the experimental evaluation.

2. Foundations and Related Work

In the following chapter, we briefly introduce and formalize the main concepts used in this work and we further discuss related work regarding the different points that this work focuses on.

2.1. Foundations

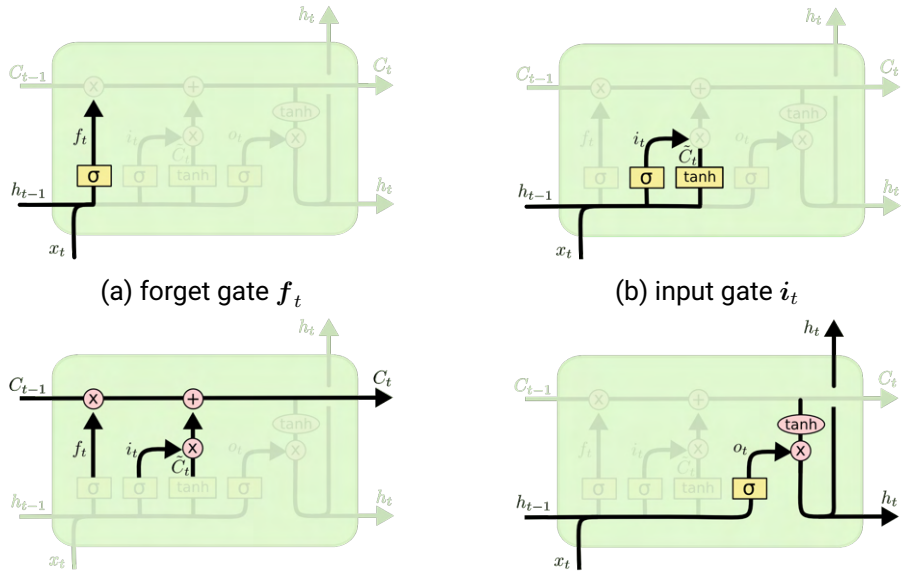
We now introduce and formalize the concepts of LSTMs [13], probabilistic latent models [14, 28], sequential modeling of latent space [8, 12], and Bayesian Interaction Primitives (BIPs) [4].

2.1.1. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) [13] networks are special forms of Recurrent Neural Networks (RNNs) that can learn long time dependencies on temporal data. They extend the standard RNN-structure – where a state vector $\boldsymbol{h}_t$ that encodes information of previous timesteps is an additional input to a NN that iteratively computes $\boldsymbol{h}_t$ and other task specific outputs for each timestep t using the according input $\boldsymbol{x}_t$. In LSTMs additional state vectors and gating units that control how much of the new information $\boldsymbol{x}_t$ and the previous information $\boldsymbol{h}_{t-1}$ gets carried through the network are introduced. They include a cell state $\boldsymbol{c}_t$, an input gate $\boldsymbol{i}_t$, which manages how much of the new information $\tilde{\boldsymbol{c}}_t$ flows into $\boldsymbol{c}_t$, an output gate $\boldsymbol{o}_t$, which controls the extent of which the cell state $\boldsymbol{c}_t$ influences the state $\boldsymbol{h}_t$, and a forget gate $\boldsymbol{f}_t$, which controls how much of the previous information $\boldsymbol{c}_{t-1}$ flows into $\boldsymbol{c}_t$. They are computed using the following equations:

$$
\begin{aligned}
\boldsymbol{f}_t &= \boldsymbol{sig}(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \in (0,1)^h, \\
\boldsymbol{i}_t &= \boldsymbol{sig}(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \in (0,1)^h, \\
\boldsymbol{o}_t &= \boldsymbol{sig}(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \in (0,1)^h, \\
\tilde{\boldsymbol{c}}_t &= \boldsymbol{tanh}(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c c) \in (-1,1)^h, \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tilde{\boldsymbol{c}}_t \in \mathbb{R}^h, \\
\boldsymbol{h_t} &= \boldsymbol{o}_t \circ \boldsymbol{tanh}(\boldsymbol{c}_t) \in (-1,1)^h,
\end{aligned}
$$

with $\boldsymbol{c}_0 = \boldsymbol{0}, \boldsymbol{h}_0 = \boldsymbol{0}$ and $\circ$ denoting the element-wise product. The Structure of a LSTM-module is shown in a step-by-step view in Figure 2.1 [21].



(a) forget gate $\boldsymbol{f}_t$

(b) input gate $\boldsymbol{i}_t$

(c) cell state $\boldsymbol{c}_t$

(d) output gate $\boldsymbol{o}_t$ and state $\boldsymbol{h}_t$

Figure 2.1.: Step-by-step view[1] of an LSTM: (a) computation of the forget gate; (b) computation of the input gate; (c) computation of the cell state (d) computation and usage of the output gate.

[1] Image taken from `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Last accessed on November 3, 2021

2.1.2. Probabilistic Latent Models

Probabilistic Latent Models are a family of generative models derived from Auto-encoder (AE)-models, which are encoder-decoder models that aim to reconstruct their input $\boldsymbol{x}$ from an encoded lower-dimensional representation. Instead of directly computing the latent vector $\boldsymbol{z}$, the encoder in a Probabilistic Latent Model computes the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ of a normal-distribution from which a latent vector can be sampled (e.g. $\boldsymbol{z} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) = q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$). As stochastic nodes cannot be optimized trough back-propagation, a reparameterization trick is used to generate samples. A valid reparameterization for $\boldsymbol{z}$ is $\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ [14]. Since $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are now deterministic nodes, back-propagation can be applied to them as usual.
The loss-function of probabilistic latent models is typically composed of a reconstruction term and a regularization term. The former ensures that the results resemble the inputs and the latter aims to make the latent space continuous and complete by forcing the latent distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to be close to a fixed prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ which is typically chosen to be the standard normal distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.

Variational Auto-Encoders (VAEs)

The original VAE [14] achieves this by using the Kullback-Leibler divergence (KL-divergence) in order to reward $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ being close to the prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$. This results in the following definition of the loss-function for a single input $\boldsymbol{x}_i$:

$$\mathcal{L}_{\text{VAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_i) = \underbrace{-D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)||p_{\boldsymbol{\theta}}(\boldsymbol{z}))}_{\text{regularization term}} + \underbrace{\frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z}_{i;l}))}_{\text{reconstruction loss}} \quad .$$

Here $\boldsymbol{z}_{i,l}$ are reparameterized samples $\boldsymbol{z}_{i;l} = g_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{\epsilon}_{i;l}), \boldsymbol{\epsilon}_{i;l} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ [14]. Without a regularization term, latent representations could be arbitrarily far apart. Thus, when decoding a random point that is not close to any latent point seen during training, the results would not be meaningful. As the regularization term condenses the latent space around the prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, one can sample a latent vector from $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ and yield good results.

Wasserstein Autoencoders (WAEs)

In addition to the original VAEs probabilistic latent models encompass other variations such as Wasserstein Autoencoders (WAEs) [28], which suggests the usage of the Wasserstein-distance to construct the loss-function. One of the two proposed regularizers utilizes empirical Maximum Mean Discrepancy (MMD) between M samples $\hat{\boldsymbol{z}}_i$ of the prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{z}}_i)$ and the encoded $\boldsymbol{z}_i$ which doesn't force each $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to be close to $p_{\boldsymbol{\theta}}$ but instead forces the continuous mixture $q_{\boldsymbol{z}} := \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\ dp_{\boldsymbol{\theta}}$ to be close to $p_{\boldsymbol{\theta}}$. This allows the different examples to stay somewhat far away from another while still being gathered around the prior and thus reducing overlap between the latent representation by that and improving the reconstruction [28]. This results in the following loss function for an input batch $\boldsymbol{X} \in \mathbb{R}^{N \times D_{\boldsymbol{x}}}$:

$$
\begin{aligned}
\mathcal{L}_{\text{WAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}) = & \overbrace{\frac{1}{N} \sum_{i=1}^{N} c(\boldsymbol{x}_i, \mathbf{Dec}(\boldsymbol{z}_i))}^{\text{reconstruction loss}} + \overbrace{\lambda \cdot \widehat{MMD}(Q_{\boldsymbol{z}}, P_{\boldsymbol{z}})}^{\text{regularization term}} \\
= & \frac{1}{N} \sum_{i=1}^{n} c(\boldsymbol{x}_i, \mathbf{Dec}(\boldsymbol{z}_i)) + \frac{\lambda}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} k(\boldsymbol{z}_i, \boldsymbol{z}_j) + \frac{\lambda}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} k(\hat{\boldsymbol{z}}_i, \hat{\boldsymbol{z}}_j) \\
& + \frac{\lambda}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} k(\boldsymbol{z}_i, \hat{\boldsymbol{z}}_j),
\end{aligned}
$$

where c is the 2-Wasserstein distance $c(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$, k is a kernel (e.g. the IMQ-kernel $k(\boldsymbol{x}, \boldsymbol{y}) = C/(C + \|\boldsymbol{x} - \boldsymbol{y}\|_2^2)$) or a mixture of kernels, $\lambda > 0$ is a hyper-parameter and **Dec** is the decoder function [28]. The difference between VAEs and WAEs is illustrated in Figure 2.2 [28].

2.1.3. Sequential Modeling of Latent Space

Probabilistic latent models themselves are not fit for sequential data, as they lack a way to include the context from previous inputs. Combining them with recurrent units and modifying the prior in such a way that it is no longer fixed but flexible with respect to the input data has shown promising results on structured sequences [8, 12]. We'll further explore two approaches on doing so.

[2]Figure excerpted from Ilya Tolstikhin et al. "Wasserstein Auto-encoders" (2019).

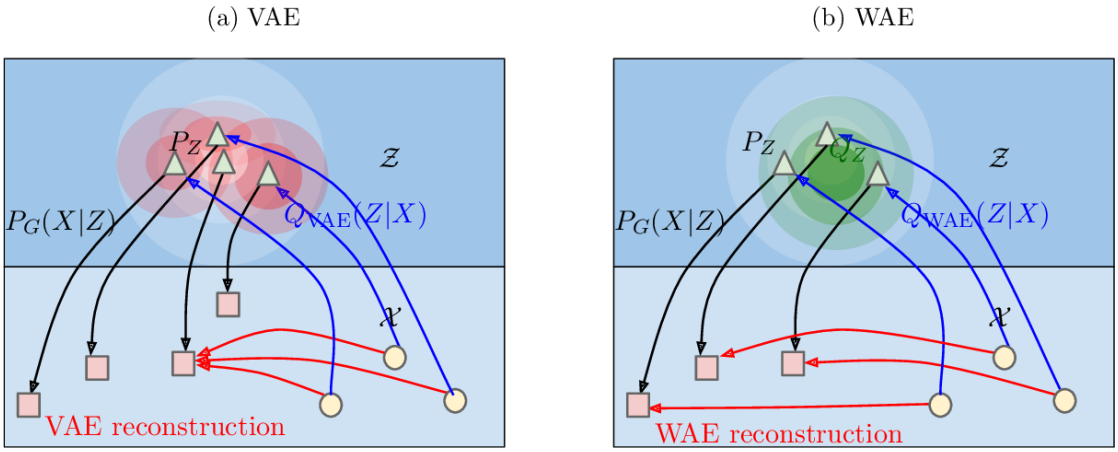


Figure 2.2.: Examplary depiction[2]of the difference between VAEs and WAEs. Both minimize the reconstruction loss in combination with the regularization term. The regularization term of the VAE's loss function forces the each $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ (depicted as the red areas in picture (a)) to be close to $p_{\boldsymbol{\theta}}$ (depicted as the white areas in both pictures). The red areas are intersecting which can lead to problems when reconstructing. The regularization term of the WAE's loss function on the other hand forces the continuous mixture $q_{\boldsymbol{z}} := \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\, dp_{\boldsymbol{\theta}}$ (depicted as the green areas in picture (b)) to be close to $p_{\boldsymbol{\theta}}$. Here the latent vectors have a better chance of staying away from each other and thus reducing overlap.

Variational Recurrent Neural Networks (VRNNs)

Variational Recurrent Neural Networks (VRNNs) [8] are a combination of RNNs and VAEs. The input data $\boldsymbol{x}_t$ passes through a VAE at each timestep, however, in order to take the temporal structure of the input into account, the input of both the encoder Enc and the decoder Dec will be concatenated with the state variable $\boldsymbol{h}_{t-1}$ of a recurrent unit which can be an RNN like in the original proposition of Chung et al. 2016, but it can also be a more complex recurrent unit like a Gated Recurrent Unit (GRU) or an LSTM[8].

$$\begin{aligned} \textbf{Enc} : \mathbb{R}^{D_{\boldsymbol{h}}+D_{\boldsymbol{x}}} \to \mathbb{R}^{D_{\boldsymbol{z}}}, \qquad \textbf{Dec} : \mathbb{R}^{D_{\boldsymbol{h}}+D_{\boldsymbol{z}}} \to \mathbb{R}^{D_{\boldsymbol{x}}}, \\ \boldsymbol{h_t} \in \mathbb{R}^{D_{\boldsymbol{h}}}, \qquad \boldsymbol{x_t} \in \mathbb{R}^{D_{\boldsymbol{x}}}, \qquad \boldsymbol{z_t} \in \mathbb{R}^{D_{\boldsymbol{z}}} \qquad . \end{aligned}$$

The hidden state $\boldsymbol{h}_t$ of the RNN is updated in dependence on the previous hidden state $\boldsymbol{h}_{t-1}$, current input $\boldsymbol{x}_t$ and its latent representation $\boldsymbol{z}_t$.

$$\textbf{RNN} : \mathbb{R}^{D_{\boldsymbol{h}}+D_{\boldsymbol{x}}+D_{\boldsymbol{z}}} \to \mathbb{R}^{D_{\boldsymbol{h}}},$$

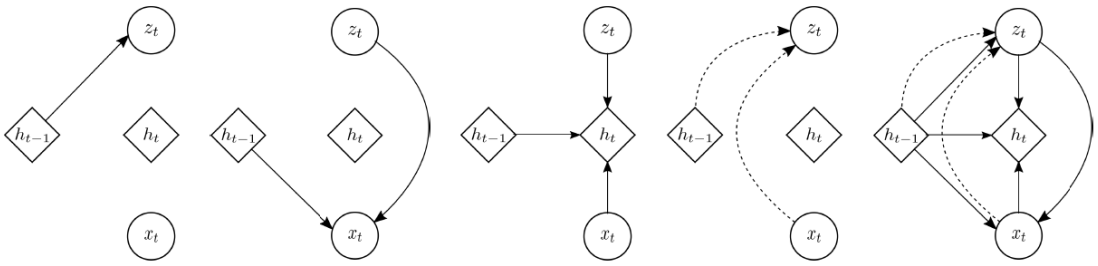
Additionally, the prior $p_{\boldsymbol{\theta}}$ is no longer a static distribution, instead it is parameterized by another function – e.g. another neural network – with $\boldsymbol{h}_{t-1}$ as its input: $[\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t] = \textbf{Prior}(\boldsymbol{h}_{t-a})$ [8]. This moving prior allows $\boldsymbol{z}$ to adapt as needed while still being constrained to ensure continuity in the latent space.

The respective loss function for a single input $\boldsymbol{x} = [\boldsymbol{x}_t]_{t=1..T}$ is composed of the timestep-wise losses of all $\boldsymbol{x}_t$:

$$\mathcal{L}_{\text{VRNN}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T}\left[-D_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{x}_{<t})||p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{<t}, \boldsymbol{z}_{<t})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{\leq t}\boldsymbol{z}_{\leq t;l}))\right],$$

with, again, $\boldsymbol{z}_{\leq t;l}$ being reparameterized samples $\boldsymbol{z}_{\leq t;l} = g_{\boldsymbol{\phi}}(\boldsymbol{x}_{\leq t}, \boldsymbol{\epsilon}_{\leq t;l}), \boldsymbol{\epsilon}_{\leq t;l} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ [8]. A schematic visualization of the proposed VRNN is shown in figure 2.3 [8].

[3] Figure excerpted from Junyoung Chung et al. "A Recurrent Latent Variable Model for Sequential Data" (2016).



(a) Prior (b) Generation (c) Recurrence (d) Inference (e) Overall

Figure 2.3.: Schematic view[3] of each segment of the VRNN: (a) computation of the moving **Prior** function; (b) decoding function **Dec**; (c) **RNN** state update (d) encoding function **Enc**; (e) complete scheme of the VRNN.

Recurrent Wasserstein Autoencoders (RWAEs)

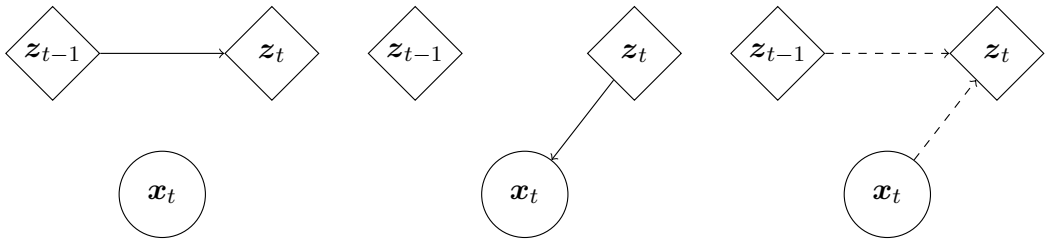
Note that in the following we will describe a simplified version of the RWAE [12] that omits the originally proposed disentanglement of $\boldsymbol{z}_t$ into a static part $\boldsymbol{z}^{\mathrm{c}}$ and a dynamic part $\boldsymbol{z}^{\mathrm{m}}$ and the proposed weak supervision.
Another proposal for probabilistic latent models are Recurrent Wasserstein Autoencoders (RWAEs) [12] which adapt WAEs to sequential data. Like in the aforementioned VRNNs, in RWAEs the data passes through a WAE at each time-step. Again, the information of all previous timesteps is forwarded via a recurrent unit, however, instead of having a separated recurrent state variable $\boldsymbol{h}_t$ the latent vector $\boldsymbol{z}_t$ is used as the recurrent state.

$$\textbf{Enc} : \mathbb{R}^{D_{\boldsymbol{z}}+D_{\boldsymbol{x}}} \to \mathbb{R}^{D_{\boldsymbol{z}}}, \qquad \textbf{Dec} : \mathbb{R}^{2\cdot D_{\boldsymbol{z}}} \to \mathbb{R}^{D_{\boldsymbol{x}}}, \qquad \textbf{Prior} : \mathbb{R}^{D_{\boldsymbol{z}}} \to \mathbb{R}^{D_{\boldsymbol{x}}}$$

Like above the respective loss for an input batch $\boldsymbol{X} \in \mathbb{R}^{N\times T\times D_{\boldsymbol{x}}}$ containing N sequences is composed of the timestep-wise losses, assuming that all sequences have the same length:

$$\mathcal{L}_{\text{RWAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}) = \sum_{t=1}^{T} \left[\frac{1}{N} \sum_{i=1}^{N} c(\boldsymbol{x}_{it}, Dec(\boldsymbol{z}_{it})) + \lambda \cdot \widehat{MMD}(Q_{\boldsymbol{z}_{it}}, P_{\boldsymbol{z}_{it}}) \right] \tag{2.1}$$

For the computation of the Prior we use the batches encodings as the input for the **Prior** function. A schematic visualization of the proposed RWAE is shown in Figure 2.4.



(a) Prior (b) Generation (c) Inference

Figure 2.4.: Schematic view of each segment of a simplified RWAE [12] (a) computation of the moving **Prior** function; (b) decoding function **Dec**; (c) recurrent encoding function **Enc**

2.1.4. Bayesian Interaction Primitives (BIPs)

Bayesian Interaction Primitives (BIPs) [4] are a framework for learning probability distributions over sets of interactive trajectories. The main difference between BIPs and other MPs is that BIPs perform inference and phase estimation in a single combined step by combining the concept of interaction primitives with Extended Kalman Filter (EKF) Simultaneous Localization and Mapping (SLAM) [4].
A single trajectory is a sequence $\boldsymbol{\tau} = [\boldsymbol{y}_t]_{t=0..T}$ of T observations $\boldsymbol{y}_t = [\boldsymbol{y}_t^{a_1}, \dots, \boldsymbol{y}_t^{a_m}]^T \in \mathbb{R}^D$ of D joint positions, angles, velocities, etc. for all M involved agents ($D = \sum_{i=0..M} D_{a_i}$). When working with two agents we'll refer to them as either the controlled agent c or the observed agent o which results in $\boldsymbol{y}_t$ being $\boldsymbol{y}_t = [y_{1,t}^{\mathrm{o}}, \dots, y_{D_\mathrm{o},t}^{\mathrm{o}}, y_{1,t}^{\mathrm{c}}, \dots, y_{D_\mathrm{c},t}^{\mathrm{c}}]$.
BIPs are learned from demonstrations and model the time-varying mean and variance of trajectories. This is achieved by representing each observation as a linear combination of B time-dependent basis functions. Therefore $y_{i,t}^a$ – the i$^{\text{th}}$ degree of freedom of agent a at timestep t – is represented as:

$$y_{i,t}^a = \boldsymbol{\Phi}_t^T \boldsymbol{w}_i^a + \epsilon_y, \qquad \boldsymbol{w}_i^a \in \mathbb{R}^{B\times 1}, \qquad \boldsymbol{\Phi}_t \in \mathbb{R}^{B\times 1}, \qquad \epsilon_y \sim \mathcal{N}(0, \sigma_y) \quad .$$

Let $\boldsymbol{\Psi} \in \mathbb{R}^{B\times D\cdot B}$ be a block-diagonal matrix with $\boldsymbol{\Phi}_t^T$ along the diagonal, $\boldsymbol{\Sigma_y}$ be a diagonal matrix of σ_y and $\boldsymbol{w}$ be a weight vector, which is the concatenation of all $\boldsymbol{w}_i^a$: $\boldsymbol{w} = [\boldsymbol{w}_1^{\mathrm{o}T}, \dots, \boldsymbol{w}_B^{\mathrm{o}\,T}, \boldsymbol{w}_1^{\mathrm{c}T}, \dots, \boldsymbol{w}_B^{\mathrm{c}\,T}]^T \in \mathbb{R}^{D\cdot B\times 1}$. The likelihood of observation $\boldsymbol{y}_t$ given

the underlying weight vector $\boldsymbol{w}$ is thus given as:

$$\begin{aligned} p(\boldsymbol{y}_t|\boldsymbol{w}) &= p(\boldsymbol{y}_t|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\Psi}_t^T\boldsymbol{w}, \boldsymbol{\Sigma}_y) \\ &= \mathcal{N}\left(\boldsymbol{y}_t|\begin{pmatrix} \boldsymbol{\Phi}_t^T & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\Phi}_t^T \end{pmatrix}\boldsymbol{w}, \boldsymbol{\Sigma}_y\right) \quad . \end{aligned}$$

Assuming that the weight vector $\boldsymbol{w}$ is drawn from a Gaussian distribution with $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w})$, $\boldsymbol{\mu_w}$ and $\boldsymbol{\Sigma_w}$ can be computed using Maximum Likelihood Estimation (MLE) for a set of training trajectories [4].

In order to adapt to different execution speeds, a phase variable $\delta \in [0, 1]$ is introduced. At the beginning of the trajectory, it is set to zero $\delta_0 = 0$ and at the end, it is set to one $\delta_T = 1$, with everything in between being a monotonic function such as n linear interpolation of δ_0 and δ_t. It replaces t, so that the basis functions Φ are now no longer dependent on absolute time but on the progress of the trajectory instead [4]. When considering incomplete trajectories $\boldsymbol{\tau}^* = [\boldsymbol{y}_t]_{t=0..K}, K \leq T$ an approximation of the length of the complete trajectory is required to estimate the phase. This would usually be accomplished by using time alignment algorithms such as Dynamic Time Warping (DTW), however BIPs use EKF SLAM in coupling with the concept of interaction primitives in order to perform inference and phase estimation in the same step [4].

In interaction MPs, a common use case scenario is observing the partial trajectory $\boldsymbol{\tau}^*$ of one agent and to use this information to infer the associated movement for the controlled agent. This is done by conditioning $\boldsymbol{w}$ to $\boldsymbol{\tau}^*$ and generating a response trajectory from the conditioned $\boldsymbol{w}$.

$$p(\boldsymbol{w}|\boldsymbol{y}^*) \propto p(\boldsymbol{y}^*|\boldsymbol{w})p(\boldsymbol{w})$$

In this case, all controlled agents degrees of freedom (DOFs) are replaced with zero while keeping the DOFs of the observed agent, resulting in the observation $\boldsymbol{z}$. Additionally, the measurement noise values in $\boldsymbol{\Sigma}_y$ are replaced with an arbitrary but suitably high value resulting in the new matrix $\boldsymbol{\Sigma}_z$ [4].

When combining those prerequisites with SLAM, the phase δ, its velocity $\dot{\delta}$ and the weight vector $\boldsymbol{w}$ get concatenated into the state vector $\boldsymbol{s}_t = [\delta, \dot{\delta}, \boldsymbol{w}^T]^T \in \mathbb{R}^{D \cdot B + 2 \times 1}$ which is conditioned instead of $\boldsymbol{w}$ alone. This changes the goal from determining $p(\boldsymbol{w}|\boldsymbol{y}^*)$ to determining $p(\boldsymbol{s_t}|\boldsymbol{z}_{1:t}) = \mathcal{N}(\boldsymbol{s_t}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma_t})$, which has time dependent parameters. That can be determined iteratively:

$$\boldsymbol{\mu}_0 = [0, \beta, {\boldsymbol{\mu}_{\boldsymbol{w}}}^T]^T,$$
$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{F}^T[1,1]^T + \mathcal{N}(\boldsymbol{0}, \boldsymbol{F}^T\boldsymbol{Q}\boldsymbol{F}),$$
$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t & \dots & 0 \\ 0 & 1 & \dots & 0 \end{bmatrix}, \qquad \boldsymbol{Q}_t = \begin{bmatrix} \sigma_{\delta,\delta} & \sigma_{\delta,\dot{\delta}} \\ \sigma_{\dot{\delta},\delta} & \sigma_{\dot{\delta},\dot{\delta}} \end{bmatrix},$$

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} \boldsymbol{\Sigma}_{\delta,\delta} & \boldsymbol{\Sigma}_{\delta,\delta} \\ \boldsymbol{\Sigma}_{\delta,\delta} & \boldsymbol{\Sigma}_{\delta,\delta} \end{bmatrix},$$
$$\boldsymbol{\Sigma}_t = \boldsymbol{G}_t\boldsymbol{\Sigma}_{t-1}\boldsymbol{G}_t^T + \boldsymbol{R}_t, \qquad \boldsymbol{G}_t = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Similar to the standard interaction primitives, the conditioning of $\boldsymbol{s}_t$ for $\boldsymbol{\tau}^*$ is done via Bayesian filtering on $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$:

$$\boldsymbol{H}_t = \begin{bmatrix} \frac{\partial {\boldsymbol{\Phi}_\delta}^T \boldsymbol{w}_1^{\mathrm{o}}}{\partial \delta} & \boldsymbol{0} & {\boldsymbol{\Phi}_\delta}^T & \dots & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial {\boldsymbol{\Phi}_\delta}^T \boldsymbol{w}_{D_{\mathrm{o}}}^{\mathrm{o}}}{\partial \delta} & \boldsymbol{0} & \boldsymbol{0} & \dots & {\boldsymbol{\Phi}_\delta}^T & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \frac{\partial {\boldsymbol{\Phi}_\delta}^T \boldsymbol{w}_1^{c}}{\partial \delta} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial {\boldsymbol{\Phi}_\delta}^T \boldsymbol{w}_{D_c}^{c}}{\partial \delta} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{D\cdot B \times B+2},$$
$$\boldsymbol{K} = \boldsymbol{\Sigma}_t\boldsymbol{H}_t^T(\boldsymbol{H}_t\boldsymbol{\Sigma}_t\boldsymbol{H}_t^T + \boldsymbol{Q}_t)^{-1},$$
$$\boldsymbol{\mu}_t^{[\text{new}]} = \boldsymbol{\mu}_t + \boldsymbol{K}_t(\boldsymbol{z}_t - h(\boldsymbol{\mu}_t)),$$
$$\boldsymbol{\Sigma}_t^{[\text{new}]} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{\Sigma}_t \tag{2.2}$$

With the conditioned $\boldsymbol{\mu}_t^{[\text{new}]}$ and $\boldsymbol{\Sigma}_t^{[\text{new}]}$ for all t, a suitable trajectory can be sampled from the associated normal distributions $\mathcal{N}(\boldsymbol{\mu}_t^{[\text{new}]}, \boldsymbol{\Sigma}_t^{[\text{new}]})$ [4].

2.2. Related Work

We will now discuss related work regarding the generation of movement in response to the human agent, interaction classification and HHI-supported robot control.

2.2.1. Responsive Movement Generation

HRI has been the focus of many works in robotics and thus there are many approaches regarding responsive movement generation that have already been explored. In the following, we will discuss some of those approaches in detail.

For movements that revolve around hand reaching, such as handshakes or fist bumps, for example, some works suggest predicting the final location of the hand continuously using an LSTM[29, 25]. The robot is then made to reach for a location which is a time-depended non-linear combination of the predicted location and current hand position, that favors the actual hand position more as time passes. This can be achieved by linear interpolations [29] or by employing MPs conditioned on the predicted position [25] to give some examples. While this approach yields high accuracy, it has its limitation as not every form of HRIs requires reaching in the same ways or sometimes at all. For example, in times of the COVID-19 Pandemic, many people have opted for an elbow bump or waving instead of the usually more common hand-shake. Including other interactions that are not centered around reaching, requires a lot of task-wise adaptation and does not allow for effortless online learning especially for more involved interactions, while some types of interactions are simply not realizable.

Common strategies also include the usage of Machine Learning (ML) that are trained to predict the whole-body movements of an agent [2] given its own and the other agent's previous movements. This prediction in turn can be used to control the robot later on. The ML can for example be done through a recurrent AE such as an VRNN [2] or a Deep Reinforcement Learning (DRL) algorithm [8]. This approach builds on top of non-collaborative human motion prediction strategies [16, 3]. An advantage of this approach is that the model can be extended to include multiple types of interactions.

Interaction MPs of different sorts have been proposed [1, 19, 4] and explored [10, 5, 15] in many recent works regarding HRI, where they have shown promising results. Unfortunately, the conditioning of MPs is heavily reliant on matrix multiplication. Since the

sizes of these matrices are dependent on how many DOFs are involved, the conditioning becomes computationally inefficient with a growing number of DOFs. This inefficiency can be problematic when expecting the robot to react in time, but Dermy et al. have shown, that it can be addressed by replacing the actual input joints with a lower-dimensional encoding of it [9].

2.2.2. Interaction Classification

Determining the the ongoing interaction can be helpful in the previously mentioned ML-based approaches by affecting the reward function in DRL [7] or as additional information available in a Neural Network (NN) [2]. It can even be necessary, for example, when choosing the right MP from a set of possible options.

Predicting the category in a MTL-setting while also predicting future movement has shown to improve the accuracy of the movement prediction in a non-collaborative scenario [16]. Additionally, the classification itself seems to be able to keep up with models that direct all of their resources towards classification [3]. Thus MTL could be a reasonable extension to NN-based approaches in HRI.

On the other hand approaches that rely in MPs, specifically those that represent trajectories in a probabilistic manner (namely Probabilistic Movement Primitives (ProMPs) [22, 23] and BIPs [4]), can quite simply make use of probabilistic classification [19] and/or clustering with Gaussian Mixture Models (GMMs) [10, 18, 15]. This can also be done with adaptive GMMs which in turn allows for online-learning [15]. The works utilizing this assume that the entire trajectory can be observed before the robot needs to react.

2.2.3. HHI-supported Robot Control

One hurdle in controlling robots is that different robots need to be controlled differently. Consequently, each robot needs a separately trained model with according training data. The procedure of collecting HRI-data can be cumbersome thus, doing so for every variation of robots is not feasible. As a result, many works have addressed this problem and tried to alleviate the need for robot-specific data with HHI-data [25, 2]. One approach is to treat a humanoid robot as if it were a person when inferring its trajectory and to control it as if it were teleoperated [11, 25]. Others suggest learning an encoding for human

motions and robot motions. In either scenario – HHI or HRI – the encoding is affected by a task dynamic among the participating agents. With that, the training done on the HHI-data lays the foundation from which each robot can be trained [2]. This approach still requires robot-specific data but should reduce the amount of data needed to properly learn interactions.

3. Interaction Recognition and Generation (ReGen) Pipeline

In this chapter, we'll firstly formalize the problem statement for this work. Secondly, we'll explain the structure of our proposed Interaction Recognition and Generation (ReGen) Pipeline and discuss the properties and design choices of its different parts in detail.

3.1. Problem Statement

The goal of this work is to learn a library of interactions such that social robots can recognize the type of interaction that is being initiated by an observed human agent and that they are able to perform the associated counter movement in real-time. Formally, the robot observes a partial trajectory $\boldsymbol{\tau}^* = [\boldsymbol{x}_t^{\mathrm{o}}]_{t=0..T^*}$ of observations $\boldsymbol{x}_t^{\mathrm{o}}$ of human joint positions at time-step T^*. The robot should be able to classify the current interaction C_t^* from a set $C = \{C_i\}_{i=0..K}$ of K possible interactions and to generate a corresponding observation $\hat{\boldsymbol{x}}_{T^*}^{\mathrm{c}}$ in response, which can be used to control the robot accordingly. This should happen in a timely manner, i.e. the computation time $c_{\text{computation}}$ should not exceed $\frac{1}{f}$, with f being the observation frequency.

3.2. Proposed Approach

This section describes the structure of the approach of this work and explains the modeling choices. Mainly, there are two separate processes at play; The training-data pre-processing and the pipeline that generates the robot's motion.

For the pre-processing of the training data (Section 3.2.1) we focused on segmenting an

interaction into less complex sub-interactions to improve distinguishability and reconstruction of different interactions.

The movement generation pipeline is inspired by AE-ProMPs proposed by Dermy et al. [9]. They employ AEs in order to reduce the dimensionality of the MP's input. This is done because the inference of trajectories is computationally inefficient with higher dimensional inputs. Since we also want to perform action recognition, we extend the AE with a classification task and add a gating unit (Section 3.2.3) that manages which MPs to use. Unlike AE-ProMPs, we chose to encode and decode each timestep with the previous ones in mind and opted to use variational auto-encoding techniques, (Section 3.2.2) instead of standard AEs as they tend to perform better. Specifically we used a MTL RWAE. Moreover, we chose BIPs (Section 3.2.4) over ProMPs as they are more efficient because they don't need additional DTW and thus allows either for a higher computation frequency or for a higher dimensional latent space at the same frequency. This results in the pipeline depicted in Figure 3.1.

3.2.1. Interaction Segmenting

Segmenting complex movements can be beneficial when trying to classify a movement from a set of possible movements when these have similarities at some points of the trajectory. Especially, when the classification is done without observing the whole trajectory. We argue that – at least when looking at common western social interactions – many of them do have such similarities. A lot of them are centered around hand movement and thus many start with a reaching movement. While humans are mostly able to infer what interaction another person is initiating, robots may lack some information, for example when their observations do not include positioning of fingers or they lack the necessary context to correctly classify the interaction.

Additionally, the responses could be much more stable when dealing with variations, whether that entails reacting to different numbers of repetitions in cyclical parts of interactions or even the creation of new interactions that are sequenced from the set of known segments.

We propose segmenting the training trajectories in such a way that similarities across interactions will be mapped to the same primitive. This could be achieved in different ways but one common way to do so is by applying heuristics. We propose to create a

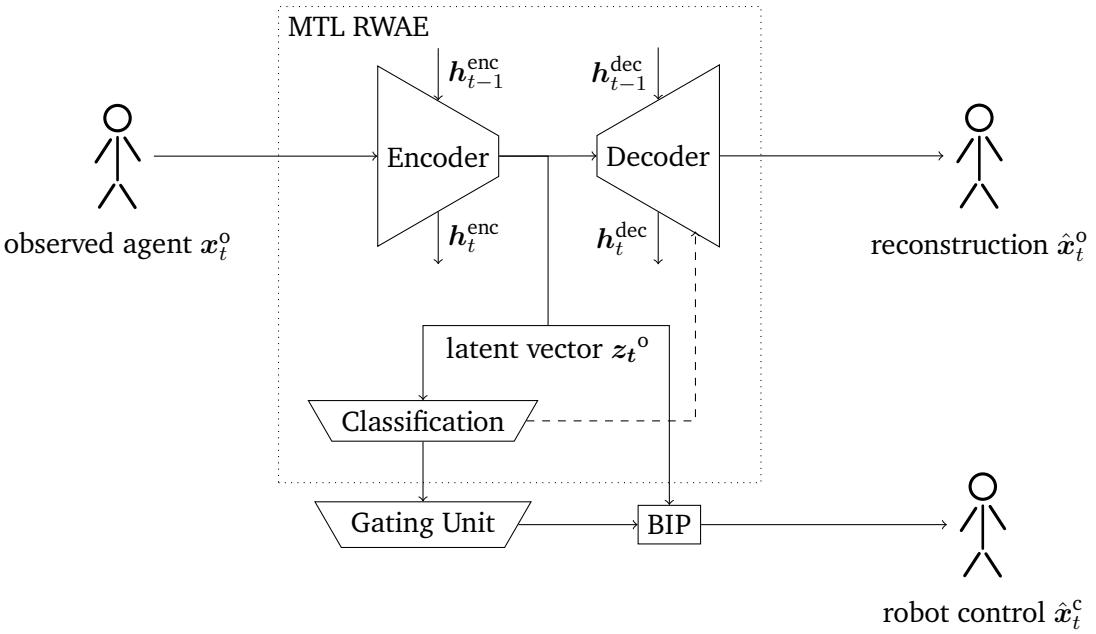


Figure 3.1.: Graphical depiction of the proposed Interaction ReGen Pipleline. At each timestep, we compute a latent representation z_t^o of the observed agents joints x_t^o and use this to classify the current action. Both the classification and the latent representation are used to infer the response $\hat{x}_t^c$ of the controlled agent using BIPs.

segment every time a velocity of zero for all joints can be observed (Zero Crossing velocity (ZCV)) or in other words every time a moment stops or changes direction. Since we are working with discrete observations we need to approximate the velocity at each timestep by observing how much the joint positions differ from timestep t to $t+1$. It is unlikely to observe velocities with an exact value of zero under these circumstances, thus we leverage the constraint by introducing a threshold that the absolute velocity be beneath of, to create a segment. More precisely, to avoid over-segmentation in periods of little movement a segment is only created when the absolute velocity exceeds the threshold after previously undershooting it. The segmentation of a synthetic trajectory with one degree of freedom is shown in Figure 3.2. In order to gain meaningful information from these automatically generated segments, they need to be reviewed, over-segmentation needs to be resolved and the segments need to be labeled with respect to what part of the interactions they represent. We also suggest labeling the resulting segments on whether

they are cyclical or not.

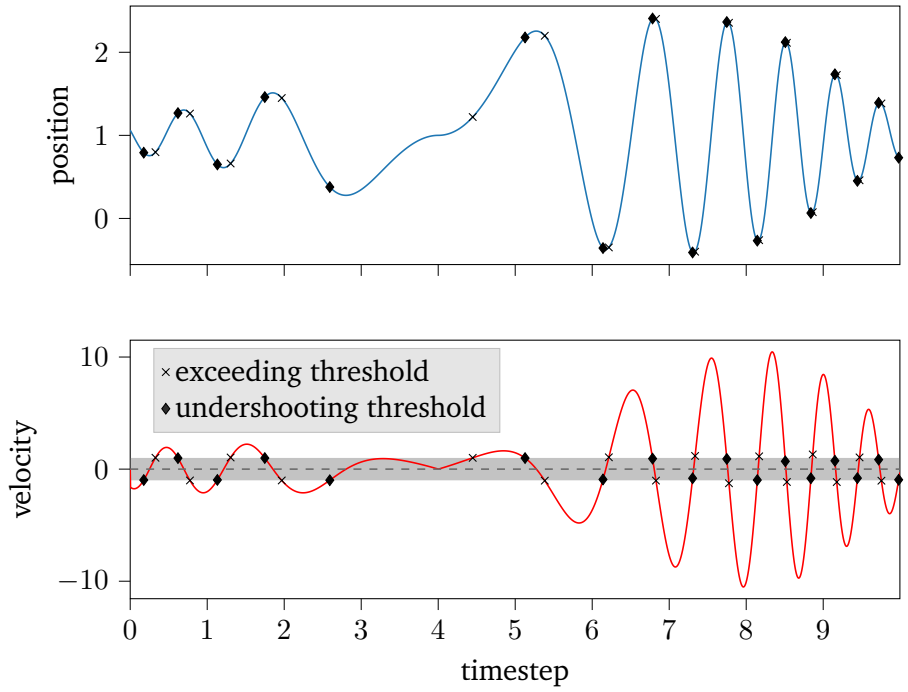


Figure 3.2.: Possible segmentation of a synthetic trajectory with a single DOF. The position of the DOF is plotted in blue in the upper graph. Below, the respective approximate velocity is plotted in red. The timesteps at which the absolute value of the approximate velocity enters and exits the range encompassed by the threshold are marked in both graphs.

3.2.2. Motion Auto-Encoding and Classification using an RWAE

The RWAE part of the pipeline is – as previously mentioned – responsible for two sub-tasks; Dimensionality Reduction and Interaction Classification. As the input data is sequential, each task will be continuously performed, taking the representations of earlier time-steps into account. In the following, we'll describe and formalize the two tasks and elaborate on how they are combined using MTL.

Dimensionality Reduction

One of the main tasks of the network is to act as a mean of dimensionality reduction. This is done to counter BIPs' computational inefficiency on high dimensional data. As seen in Equation 2.1.4, the conditioning of MPs is heavily reliant on matrix multiplication. Since the sizes of these matrices depend on the number of DOFs, the conditioning becomes computationally inefficient the higher that number.
The dimensionality of the data could be reduced by manually choosing a subset of joints that are deemed important for a specific interaction. For example, one might reduce the input's dimensionality by specifically observing the joints of the upper body (See e.g. [25]) or even only the joints of the right arm, as they are the main actors in performing many hand-centered interactions such as a handshake. However, the important joints for different interactions may vary. For instance, a handshake uses different joints as bowing does. This results in an overhead of manual labor for identifying and implementing the best choice for said subsets, in order to include a multitude of tasks. The optimal subset might also not always be clear without extensive cross-validation, especially for more involved interactions. Furthermore, it might be hard to actually choose a sufficiently small subset of joints while still fully representing the important parts of an interaction when many joints play a significant role in performing it.
AE-models could offer a way to automate this process across multiple interactions. The learned encoding might even be more efficient than the ones done by hand since the movement of multiple joints could be represented in a single degree of freedom. Variational versions of AEs, such as VAEs [8] and WAE [12], typically perform better in interpolating the latent space as they enforce continuity and cover more of the latent space during training because of the sampling step during encoding. Because we're using sequential data it seems reasonable to employ some form of RNN to represent time-dependencies. For this, we used a simplified RWAE as described in Section 2.1.3 of the RWAE proposed by Han et al. [12]. The major simplification is that we don't disentangle the encoding into a static and a dynamic encoding because this would require observing the whole movement before performing the associated robot motion, which contradicts this work's goal of interacting in real-time.

Interaction Classification

The NN's second task is to classify the interaction that is currently being performed to choose the matching BIP for the robot to perform further down the pipeline. The output

of This task is a vector $\boldsymbol{o} \in \mathbb{R}^K$ with K being the number of known interactions. The sum of all values is normalized to 1 by applying the softmax-function to $\boldsymbol{o}$:

$$\hat{y}_i = \text{softmax}(\boldsymbol{o})_i = \frac{e^{o_i}}{\sum_{j=1}^{K} e^{o_j}} \quad .$$

That way $\hat{\boldsymbol{y}} = [\hat{y}_i]_{i=1..K}$ represents the probability of each interaction. The loss function is chosen to be the categorical Cross-Entropy-Loss (CEL)

$$\mathcal{L}_{\text{classification}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{K} y_i \cdot \log \hat{y}_i, \tag{3.1}$$

where $\boldsymbol{y}$ is the target classification as a one-hot vector. Alternatively, one could use weighted CEL

$$\mathcal{L}_{\text{weighted classification};\boldsymbol{w}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{K} w_i \cdot y_i \cdot \log \hat{y}_i, \tag{3.2}$$

in order to counter class imbalances. Here $\boldsymbol{w}$ should be a weight vector whose elements are in inverse proportion to the according classes' portions in the training set. With $\boldsymbol{w} = \mathbf{1}$ $\mathcal{L}_{\text{weighted classification};\boldsymbol{w}}$ becomes equivalent to $\mathcal{L}_{\text{classification}}$.

Multi-task learning (MTL)

Previous work has already shown that including an additional classification class improves the quality of movement prediction [16] with the classification's quality staying at the same level [3]. While movement prediction is not the same as auto-encoding, we argue that the tasks are nevertheless related enough to potentially perform similarly. So, since the classification is a needed step anyway, doing so in a MTL-setting seems like a sound decision.
Similarly to a semi-supervised variation of the originally proposed RWAE [12] we extend our structure with the classification of the given sequence. This classification uses the encoded representation $\boldsymbol{z}$ of the data as its input ($\mathbf{CL} : \mathbb{R}^{D_{\boldsymbol{z}}} \to \mathbb{R}^K, \hat{\boldsymbol{y}} = \mathbf{CL}(\boldsymbol{z})$). The classification is also used as an additional input to the decoder. Unlike Han et al. [12] we don't compute one classification for the entire input sequence as this, again, would require observing the whole human trajectory. Instead, we classify the interaction at each

observed timestep. This results in the final structure shown within the dotted rectangle shown in Figure 3.1.

In order to compute the combined loss of these tasks we add the tasks losses with respect to a batch of sequences $\boldsymbol{X} \in \mathbb{R}^{N \times T \times D_{\boldsymbol{x}}}$ and target classifications $\boldsymbol{Y} \in \mathbb{R}^{N \times T \times K}$ as defined in Equation 2.1 and Equation 3.2.

$$\begin{aligned} \mathcal{L}_{\text{MTL RWAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}, \boldsymbol{Y}) &= \mathcal{L}_{\text{RWAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}) \\ + \sum_{t=1}^{T} \left[\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{weighted classification};\boldsymbol{w}}(\boldsymbol{y}_{it}, \mathbf{CL}(\boldsymbol{x}_{it})) \right] & \quad . \end{aligned}$$

Here we use of weighted CEL, but by simply choosing $\boldsymbol{w} = \mathbf{1}$ this equation becomes equivalent to non-weighted CEL. This composition of the loss function requires same-length trajectories, however, the NN should be able to deal with trajectories of varying length. Therefore, we suggest padding each sequence $\boldsymbol{X}_i$ and its target classification $\boldsymbol{Y}_i$ to the same length with fixed vectors $\boldsymbol{x}_{\text{padding}}, \boldsymbol{y}_{\text{padding}}$ that are distinct from the actual data and to introduce a function

$$\delta(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \boldsymbol{x} = \boldsymbol{x}_{\text{padding}} \\ 1, & \text{otherwise} \end{cases}$$

that is used to exclude the padded values from affecting the loss in an altered loss function.

$$\begin{aligned} \hat{\mathcal{L}}_{\text{MTL RWAE}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \delta(\boldsymbol{x}_{it}) \cdot \Big[& c(\boldsymbol{x}_{it}, Dec(\boldsymbol{z}_{it})) + \lambda \cdot \widehat{MMD}(Q_{\boldsymbol{z}_{it}}, P_{\boldsymbol{z}_{it}}) \\ & + \mathcal{L}_{\text{weighted classification};\boldsymbol{w}}(\boldsymbol{y}_{it}, \mathbf{CL}(\boldsymbol{x}_{it}))) \Big] \quad . \end{aligned}$$

This MTL approach could also potentially allow to initially train the model on unlabeled data by only optimizing the WAE's loss, alleviating the need for labeled training data slightly.

3.2.3. Gating Unit

Since the interaction classifications' certainty may vary, it may not always be optimal to begin moving the controlled agent. Because of that, we introduce a gating unit that decides if and how the respective movement primitives should be initiated.

To do so, we inspect the classifications' probabilities $\hat{y}_{it}, i = 1..K$ of the current timestep t. In case the likelihood of the top interaction $m_t = \arg\max_i \hat{y}_i$ is sufficiently high ($\hat{y}_m \geq y_{\text{threshold}}$), the controlled agent's trajectory will be inferred from the associated primitive and executed. Otherwise, we check whether we inferred a trajectory during the last timestep $t-1$. If we did and either the segment used during the previous inference is marked as cyclical or the estimated phase has not passed a certain threshold close to 1 yet ($\delta < \delta_{\text{threshold}}$), we keep inferring using the last certain MP. If none of this holds, we don't do anything for the current timestep t. The resulting gating unit is described by Algorithm 1.

Algorithm 1 Gating Unit

Input: Classification $\hat{\boldsymbol{y}}$
Find the most likely interaction $m_t := \arg\max_i \hat{y}_i$
if $\hat{y}_m \geq y_{\text{threshold}}$ **then**
    Infer the controlled agents trajectory from primitive m_t
    Execute the inferred trajectory
else if an inference was performed during the last timestep **and** (interaction m_{t-1} is cyclical **or** phase $\delta < \delta_{\text{threshold}}$) **then**
    Infer the controlled agents trajectory from primitive m_{t-1}
    Execute the inferred trajectory
    $m_t := m_{t-1}$
end if

3.2.4. Motion Generation using BIPs

We propose using a set of BIPs for the movement generation, one for each unique segment identified during the segmentation process. We train them on the the training trajectories, with one trajectory $\boldsymbol{\tau} = [\boldsymbol{x}_t]_{t=0..T}$ being composed of observations $\boldsymbol{x}_t$ that are formed from the concatenations of the encoded joint angles $\boldsymbol{z}_t^{\text{o}}$ of the observed agent and an

representation of the controlled agent's joints[1] $\boldsymbol{x}_t^{\mathrm{c}}$. When inferring a trajectory the corresponding trajectory chosen by the gating unit is conditioned in the same way as proposed by Campbell and Amor et al. [4]. Furthermore, we suggest using cyclical filtering in a similar fashion as done with the INTPRIM[2] library to allow the phase to roll back to 0 once it reaches 1 and thus to allow for any number of repetitions for segments that were marked as cyclical during the segmenting pre-processing without explicitly resetting the phase.

[1] The representation can either be the robot's joints angles in specialized scenarios or the encoding of human joint angles for generalized teleoperation scenarios.

[2] Available at `https://github.com/ir-lab/intprim`

4. Experimental Evaluation

In our experiments we tested the Interaction ReGen Pipeline on HHI data. We trained the pipeline on the HHI-part of the dataset constructed for Bütepage et al. [2]. The predicted human movements can be transfered to robot movement by teleoperaing a humanoid robot [11]. In the following chapter, we'll describe the experimental setups as well as the employed training procedures.

4.1. Dataset

In our experiments, we used the dataset that was constructed for the Bütepage et al. paper [2]. It contains high-frequency motion capture data of both human-robot and human-human interaction. This data was captured using Rokoko[1] motion capture suits that record the 3D positions of the human joints in Cartesian space at 40 Hz, which we sampled further down to 5Hz in order to have a less harsh time constraint.

The dataset contains four different types of interactions: hand wave, handshake, parachute fist-bump, and rocket fist-bump. Each interaction follows a similar structure of: initiating, n repetitions of the main movement, and returning to the starting position. This structure was considered during the segmenting resulting in seven different types of segments, each marked on whether they are cyclical or not. While the act of returning to the starting position is considered to be universal across all interactions, the segments in the initiation are partially shared and the main action phase is unique for all four interactions. The possible sequencing of these is depicted in Figure 4.1 and some frames of a sample recordings of the interactions alongside corresponding extracted joint angles are depicted in Figure 4.3, 4.4, 4.5 and 4.6. Additionally a description of each segment can be found

[1] `https://www.rokoko.com/`

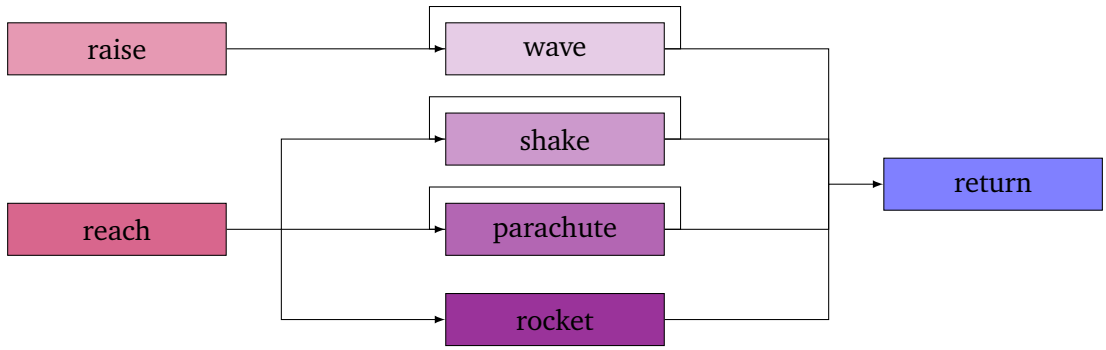


Figure 4.1.: A flowchart depicting the possible interaction transitions based on the manual segmenting done on the Bütepage et al. dataset [2]. Since the dataset doesn't contain hand posture, the reaching can be considered to be the same movement for the handshake and both fistbumps. For data including hand posture, it might be necessary to split this segment further.

in Table 4.1 and the supplementary material [2] of Bütepage et al. [2] provides a video containing all different base types of interactions. It needs to be noted that the dataset included variations in the segment transitions. So the waving, for example, would not always start with moving the arm to the leading actors left and it would not always end with moving the arm to the leading actors right, in contrast to the segment description found in Table 4.1. Those segments were marked as incomplete.

The distribution of the interactions and the segments in the dataset are shown in Figure 4.2. While the interactions themselves encompass roughly the same share of timesteps, the class imbalances are much more prominent in the segmented data and may thus be something to address further down the line.

Since all of the interactions happen to be based around movement of the upper body we excluded all joints that are below the hips, to alleviate computation times slightly.

The four interactions can be categorized into symmetrical and asymmetrical interactions. In symmetrical interactions, both agents do roughly the same thing while in asymmetrical interactions both actors have different roles. By that definition, the handshake and the hand wave fall under symmetrical interactions and both fist-bumps fall under asymmetrical interactions because both have a leading and a following agent. Since the robot should

[2] `https://www.frontiersin.org/articles/10.3389/frobt.2020.00047/full#supplementary-material`

Segment name	Segment description
raise	Starting from an idle position, the actors raise their right arms above their heads.
reach	Starting from an idle position, both actors stretch their arms forward to meet each other's hands between both of them. The height of the meeting point may vary.
wave	With their arms raised above their heads, the actors swing their arms from left to right in an oscillatory motion, starting the movement towards the left of the leading agent.
shake	With their hands meeting between them, the actors grasp each other's hands and shake them up and down in an oscillatory motion.
parachute	With their hands meeting between the two actors, the leading actor moves their hand above the following actor's hand. Both start moving their hands downwards while simultaneously swinging their hand from side to side, starting the movement towards the left of the leading agent. The following actor keeps their hand sightly below the leading actor's hand following their oscillatory motion. The movement stops when the hands are approximately at hip height.
rocket	With their hands meeting between the two actors, the leading actor moves their hand above the following actor's hand. Both and start moving their hands upwards. The following actor keeps their hand sightly below the leading actor's hand following the upwards motion. The movement stops when the hands are approximately at shoulder height.
return	The actors return to an idle position.

Table 4.1.: A table describing the manually identified segments for the dataset created by Bütepage et al. [2].

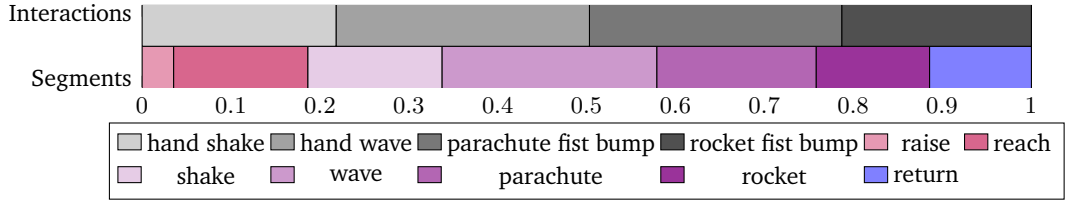


Figure 4.2.: The class distribution of the interactions (first row) and the corresponding segments (second row) with regards to the number of timesteps associated with them.

always react to the human, we determined that the robot should always fill the role of the following agent and the human should always fill the role of the leading agent.

4.2. Experimental Setups

For our experiments, we only use the human-human part of the Bütepage et al. dataset [2] because the human-robot data is specific to the robot used and because the human-human dataset itself is more extensive. In order to present the results of the experiments on an actual robot we teleoperation techniques can be employed using the generated human data [11]. To assess the success of the pipeline and to differentiate the contribution of the different parts we perform an ablation study using the configuration outlined in Table 4.2.

Each configuration was trained on 80% of the Bütepage et al. dataset [2] using the processes described in Section 4.2.1 and Section 4.2.2 and evaluated in the way described in Section 4.3 on the remaining 20% of the dataset and on the self recorded dataset.
The thresholds of the gating unit were chosen to allow fairly uncertain classifications with $y_{\text{threshold}} = 0.6$ and $\delta_{\text{threshold}} = 0.95$.

4.2.1. Implementation and Training of the RWAE

All Neural Networks were implemented using Python and PyTorch[3]. For each configuration mentioned above, we performed k-fold cross-validation with random hyper-parameters. The specific hyper-parameters that were optimized that way and the final

[3] `https://pytorch.org/`

Configuration name	Configuration description
MTL RWAE (Seg)	Employing MTL and dimensionality reduction on segmented interactions.
MTL RWAE (SegW)	Employing MTL and dimensionality reduction on segmented interactions using weighted CEL.
MTL RWAE (No Seg)	Employing MTL and dimensionality reduction on unsegmented interactions.
RWAE + CL (Seg)	Using two separate NNs for the classification and the dimensionality reduction on segmented interactions.
RWAE + CL (SegW)	Using two separate NNs for the classification and the dimensionality reduction on segmented interactions using weighted CEL.
RWAE + CL (No Seg)	Using two separate NNs for the classification and the dimensionality reduction on unsegmented interactions.
CL (Seg)	Using only classification on segmented interactions.
CL (SegW)	Using only classification on segmented interactions using weighted CEL.
CL (No Seg)	Using only classification on unsegmented interactions.

Table 4.2.: A table naming and describing all evaluated configurations.

choices for them are listed in Appendix A. Moreover the specific architecture of each components is listed in Figure B.1 in Appendix B. We kept the NNs relatively shallow because of the relatively small dataset used.

The computation of the loss of the RWAE, seen in equation 2.1, requires the use of a kernel. The proposal of the original RWAE [12] suggests the usage of a mixture of RBF-kernels $k : \sigma(\boldsymbol{x}, \boldsymbol{y}) = \exp(\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 / \sigma_k^2)$, however the proposal of the original WAE argues that inverse multiquadratics (IMQ) kernels $k_C(\boldsymbol{x}, \boldsymbol{y}) = C/(C + \|\boldsymbol{x} - \boldsymbol{y}\|_2^2)$ are a better choice [28], which is why we also opted to use a mixture of IMQ kernels

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{C_i \in C} \frac{C_i}{(C_i + \|\boldsymbol{x} - \boldsymbol{y}\|_2^2},$$

with $C = \{0.1, 0.2, 0.5, 1, 2, 5, 10\}$.

The training of the RWAE configurations was done on the data of both agents. This was done to ensure proper en- and decoding for both agents, which would not be needed when reconstructing robot specific joints for the second agent. We normalized the data using the standard score $\tilde{x} = \frac{x-\mu}{\sigma}$ for each DOF, as this yielded better results in some sample runs done before the final cross-validation and training.
To counter overfitting the model to the training data, we also introduced dropout layers in all of the components. The dropout probability p_{dropout} was the same across all dropout layers in a model and was determined during the hyper-parameter-optimization mentioned above.

4.2.2. Implementation and Training of the Bayesian Interaction Primitives

The BIPs were implemented using the INTPRIM library[4] [4, 6]. The basis spaces were optimized separately for each BIP by performing grid-search over a set of basis functions. The basis space was chosen based on the Bayesian Information Criterion. Since the incomplete cyclical segments were always missing one half of the cycle we adjusted their phase in such a way that either the phase of the first frame was 0.5 in case the first half was missing or the phase of the last frame was 0.5 in cast the second half was missing, before using them for training. With that adjustment, we were able to utilize these segments during the training of the BIPs despite their incompleteness.

4.3. Evaluation

We evaluated all configurations using multiple metrics. The RWAE component itself was evaluated on the average non-weighted CEL and the average accuracy of its classification per timestep and on the Mean-Squared-Error (MSE) loss of its reconstruction (See Table 4.3). Since accuracy is not always a good metric when dealing with unbalanced classes we also created confusion matrices for the RWAE's classification (See Figure 4.7). The BIP

[4] Available at `https://github.com/ir-lab/intprim`

were tested an optimal scenario – using the target classification instead of the RWAE's – and in the context of the full ReGen Pipeline. Furthermore we animated and inspected the testing interactions. We now further discuss the observations we made while doing so in relation to the metric evaluation.

		CEL	Accuracy	MSE
MTL RWAE	Seg	1.23716	0.67665	0.68161
	SegW	1.37827	0.85184	0.67447
	No Seg	0.78304	**0.97063**	0.99362
Non-MTL RWAE	Seg	0.52788	0.85067	**0.55059**
	SegW	**0.47736**	0.86280	**0.55059**
	No Seg	1.64068	0.75380	**0.55059**

Table 4.3.: This table shows the NN-specific metrics for all NN configurations. The CL-only configurations use the same classification NNs as the RWAE + CL ones and thus the performance of both is encompassed by Non-MTL RWAE performances listed above. Since the different configurations in these models don't affect the encoding part, the MSE is the same across all of them. The best result of each metric is printed boldly, each metric is rounded to five decimal places. CEL denotes the average non-weighted CEL per timestep, the accuracy denotes the average accuracy per timestep, and the MSE denotes the mean squared error.

In contrast to our assumptions the MSE of the reconstruction does not improve by incorporation MTL in comparison with the non-MTL models, as portrayed in Table 4.3. However, it is not significantly worse either. When comparing the MTL models with each other we can see a slight improvement when incorporating segmenting, suggesting that it benefits the encoding. When inspecting the confusion-matrices (See Figure 4.7) it is notable that all models trained on the segmented data mainly miss-classify the initiating movements and the return movement of the interactions. Some models seem to struggle with differentiating the two initiating movements, which may be due to them beginning in a similar way. Furthermore, the initiating movements and the return movement seem to get confused with the main actions and vice versa. This most likely happens when the segments transition from one to another which may not necessarily impact the movement prediction as severely as other forms of miss-classification.

When inspecting animations of the generated trajectories, the difference in errors shown in Table 4.3 is reflected in the quality of the animations. The trajectories that were inferred

		optimal classification		NN classification	
		Avg. time [sec]	MSE	Avg. time [sec]	MSE
MTL RWAE	Seg	0.32656	4.20235	0.33674	4.37260
	SegW	0.28471	3.49618	0.29840	3.49618
	No Seg	0.37678	3.53024	0.36201	3.75011
RWAE + CL	Seg	0.23800	4.39630	0.24174	4.35912
	SegW	0.21480	4.23854	0.22030	5.22607
	No Seg	**0.15540**	3.64157	**0.15879**	3.82488
CL	Seg	0.45979	**2.35523**	0.512701	2.82804
	SegW	0.45979	**2.35523**	0.463222	**2.54350**
	No Seg	0.45170	2.46183	0.41619	2.89722

Table 4.4.: MSE for the inference and manual scoring of each model using the target classification and the NN classification, respectively. The best results of each metric among a group are printed boldly, each metric is rounded to five decimal places. The MSE denotes the mean squared error of the controlled agent as compared to the training data, the avg. time denotes the average computation time of the reconditioning of the BIPs in seconds.

using encoded inputs all more or less have a bias towards keeping the hand in a strechted out position, without visibly adapting much to the observed agent's movements. As this is not the case with the BIPs trained on the original data, we conclude that the current RWAE-setup is not able to properly encode time-dependent variations in the movement.

The loss is the lowest when employing segmenting in combination with unencoded inputs. This low loss is, again, reflected by the animations inspected. While the initiating and the returning segments are always pretty reliable with and without segmenting, the main movements, specifically those that contain arbitrary numbers of repetitions, are not predicted properly without segmentation. This is not the case for the segmented BIPs.

In terms of timeliness, only the unsegmented non-MTL model was able to perform the average conditioning faster than the 0.2 seconds that would be the maximum required for real-time application using a conditioning frequency of 5 Hz. Nonetheless, we are always able to lower the computation time below 0.4 seconds with encoded trajectories. Final models for application should be adapted to match their possible conditioning rate determined during previous experimentation but in the scope of this study we don't alter the models any further.

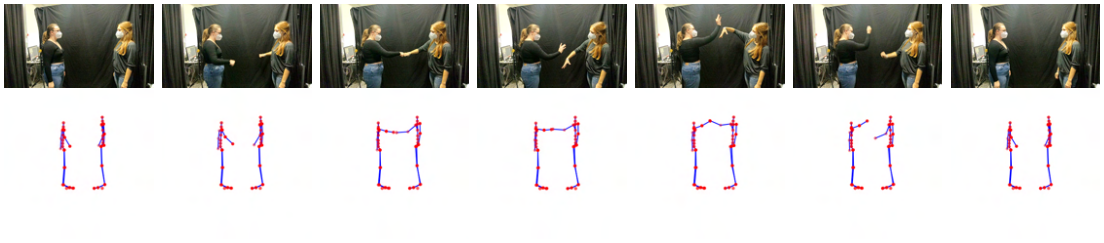


Figure 4.3.: Frames excerpted from a recording of the rocket-fist-pump (above) and corresponding joint angles (below).

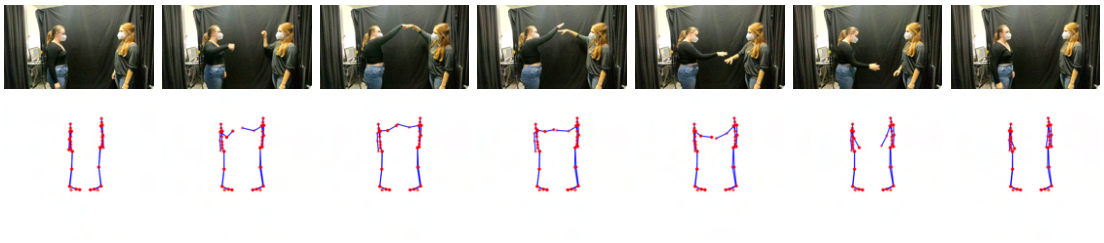


Figure 4.4.: Frames excerpted from a recording of the parachute-fist-pump (above) and corresponding joint angles (below).

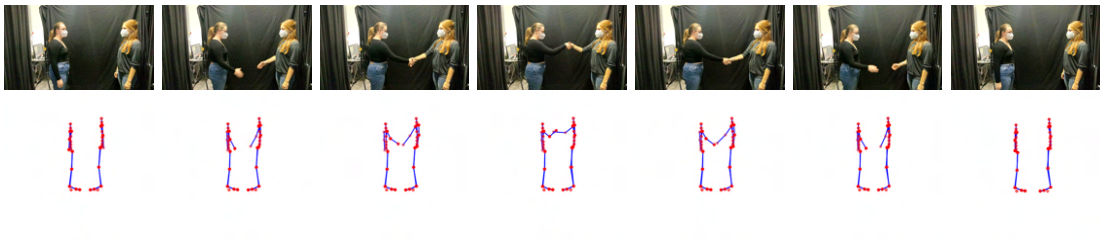


Figure 4.5.: Frames excerpted from a recording of the handshake (above) and corresponding joint angles (below).

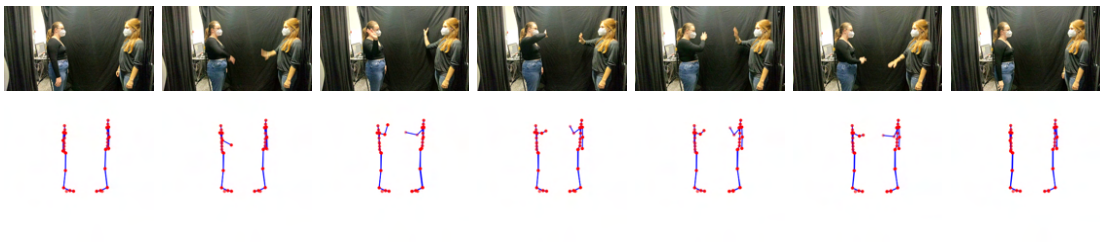


Figure 4.6.: Frames excerpted from a recording of the waving (above) and corresponding joint angles (below).

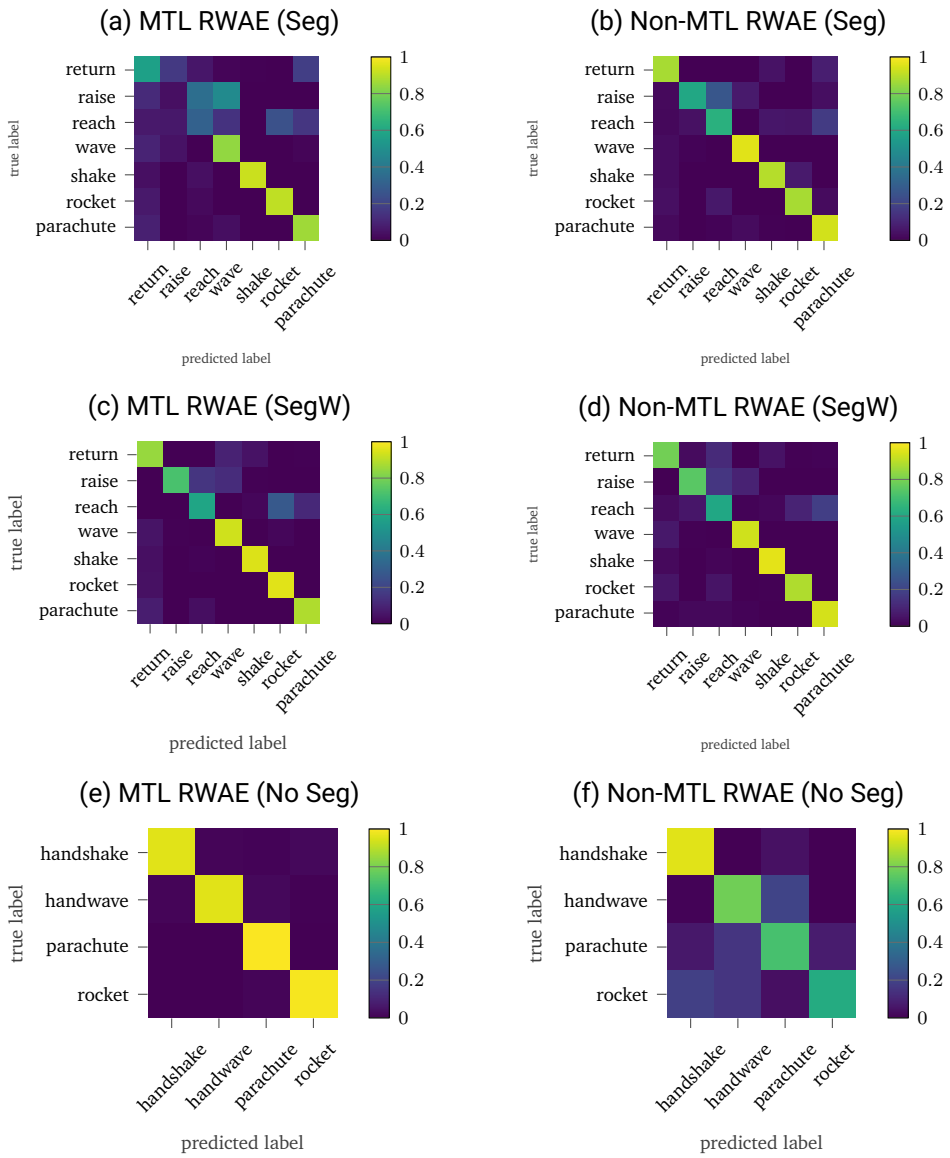


Figure 4.7.: Confusion matrices for each configuration with the rows (true labels) normalized.

5. Discussion

In this chapter, we summarize this thesis' contributions and discuss the results of chapter 4 with regards to the different pipeline stages outlined. Furthermore, we acknowledge any conscious shortcomings and explore possible space for further work.

5.1. Summary

In this thesis, we proposed the Interaction ReGen Pipeline for learning a library of physical interactions for social robots in combination with interaction sequencing. The pipeline encompasses an MTL NN for classification and dimensionality reduction of the input data, which is used to gate and to condition a set of BIP with a lower computation time than BIPs trained on the original input data. We discussed all parts of the proposed pipeline in detail and tested it's using HHI-data. Our experiments show that segmenting improves the inference of complex movements notably. Furthermore, we establish that BIPs are not able to infer movements based on large numbers of DOFs in a timely manner. Our proposed way of encoding input data in order to offset this improved the temporal behavior of the BIPs but is still lacking the in the ability to properly encode the time-varying aspects of movements.

5.2. Shortcomings and Future Work

While we hoped the segmenting would improve the accuracy of both the prediction and the inference it did only partially achieve that goal. The classifications slightly worsened after introducing the segmentation with MTL and only slightly improve without MTL. Some reason for the superior accuracy may be the increased amount of possible classes which makes the training process harder. In the long run, they could thus still perform

better when given more resources but this should be done in the scope of a more specified study as done in this thesis. Furthermore, the confusion matrices shown in Figure 4.7 suggest that a lot of miss-classifications happen when the types of segments change, which may not be as problematic as other kinds of miss-classification. Lastly, the lower accuracy may also be at least partially due to the class imbalance that the segmentation introduced. Unfortunately, this problem cannot be solved easily, as interactions are oftentimes just naturally imbalanced when some parts of them are either repeated more often than others or they are just longer, to begin with.
While weighing the classes in the MTL setting improved the classifications' accuracy again, the accuracy of the models employing weighted segmenting is still lower than the accuracy of the associated models that are trained on unsegmented data.
An untapped approach for countering the imbalance could be to treat each segment as a separate trajectory and to under- or oversample the dataset. We did not consider this method as this would firstly decrease the amount of training data at hand when undersampling large classes and secondly would deprive the RWAE of possibly important context that could be gained from knowledge of the previous segments.
It needs to be acknowledged that the Bütepage dataset [2] is optimal in several ways and does not reflect real-world circumstances perfectly. First of all, the actors wore motion tracking suits, which eliminates any problems that would occur with data generated by computer vision as it's more precise and there are no challenges such as occlusion. Furthermore, motions were executed at moderate speed with slight pauses between the different segments of the motion. In this scenario ZCV can be considered a good fit for segmenting. However in other cases, whenever movements flow into each other more seamlessly, ZCV may not be able to properly segment the trajectories. Exploring other heuristics to apply instead of or in combination with ZCV might therefore be necessary when working with different data.
Another shortcoming in that part of the pipeline is the need for manually labeling the segments. Because of that, much manual work is needed in order to extend the dataset. Further work could experiment with using unsupervised segmenting, e.g. implementing the approach proposed by Lioutikov et al.[17], to offset this workload. This could potentially get rid of any need for manual labeling.

The proposed setup of the pipeline employs normalization of the input data. This is beneficial in the sense that the data can be collected using different methods without needing to adapt the Pipeline. One downside of this, however, is that the position of the human relative to the robot is not taken into account when generating the robot's trajectory. Since the relative position alter movements where physical touch is required the lack if that information could lead to problems when the interacting person stands

in a different spot than during training. Avoiding these would require the interacting humans to stand in the same spot relative to the robot every time they want to interact with the robot, which is not feasible in uncontrolled environments. Subsequently, this problem needs to be addressed in the future.

During the training of the RWAE we opted to not differentiate both agents. This can be feasible when all interactions in the library are symmetrical or as long as the asymmetrical actions are similar enough, which we determined to be the case for the four interactions of the used dataset. However, when the acts of both agents differ strongly this needs to be acknowledged.

In any scenario, extending the library requires complete retraining so exploring online learning techniques might also be an improvement that could be considered in future work.

With our conclusion that the RWAE does not properly encode time-varying factors of the movement, but that some sort of dimensionality reduction is necessary to deal with growing amounts of DOFs, we suggest doing further work on improving the encoding with respect to the aforementioned time-variations. We assume that having the NN learn to predict the phase value or including the phase in the computation of the prior may already be simple additions that could improve the encoding and thus might be worth exploring in the future.

Bibliography

[1] Heni Ben Amor, Gerhard Neumann, Sanket Kamthe, and Jan Peters Oliver Kroemer and. “Interaction Primitives for Human-Robot Cooperation Tasks”. In: 2014.

[2] Judith Bütepage, Ali Ghadirzadeh, Mårten Björkman Özge Öztimur Karadağ and, and Danica Kragic. “Imitating by Generating: Deep Generative Models for Imitation of Interactive Tasks”. In: (2020).

[3] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. “Classify, predict, detect, anticipate and synthesize: Hierarchical recurrent latent variable models for human activity modeling”. In: (2018).

[4] Joseph Campbell and Heni Ben Amor. “Bayesian Interaction Primitives: A SLAM Approach to Human-Robot Interaction”. In: (2017).

[5] Joseph Campbell, Arne Hitzmann, Simon Stepputtis, Koh Hosoda Shuhei Ikemoot and, and Heni Ben Amor. “Learning Interactive Behaviors for Musculoskeletal Robots Using Bayesian Interaction Primitives”. In: 2019.

[6] Joseph Campbell, Simon Stepputtis, and Heni Ben Amor. “Probabilistic Multimodal Modeling for Human-Robot Interaction Tasks”. In: (2019).

[7] Sammy Christen, Stefan Stevšic, and Otmar Hilliges. “Demonstration-Guided Deep Reinforcement Learning of Control Policies for Dexterous Human-Robot Interaction”. In: (2019).

[8] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. “A Recurrent Latent Variable Model for Sequential Data”. In: (2016).

[9] Oriane Dermy, Maxime Chaveroche, Francis Colas, François Charpillet, and Serena Ivaldi. “Prediction of Human Whole-Body Movements with AE-ProMPs”. In: 2018.

[10] Marco Ewerton, Gerhard Neumann, Rudolf Lioutikov, Heni Ben Amor, Jan Peters, and Guilherme Maeda1. “Learning Multiple Collaborative Tasks with a Mixture of Interaction Primitives”. In: 2015.

[11] Lars Fritsche, Felix Unverzagt, Jan Peters, and Roberto Calandra. “First-Person Tele-Operation of a Humanoid Robot”. In: (2015).

[12] Jun Han, Martin Renqiang Min, Ligong Han, Li Erran Li, and Xuan Zhang. “Disentagled Recurrent Wasserstein Autoencoder”. In: (2021).

[13] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: (1997).

[14] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: (2013).

[15] Dorothea Koert, Susanne Trick, Marco Ewerton, Michael Lutter, and Jan Peters. “Online Learning of an Open-Ended Skill Library for Collaborative Tasks”. In: 2018.

[16] Bin Li, Jian Tian, Zhongfei Zhang, Hailin Fengand, and Xi Li. “Multitask Non-Autoregressive Model for Human Motion Prediction”. In: (2020).

[17] Rudolf Lioutikov, Gerhard Neumann, Guilherme Maeda, and Jan Peters. “Learning Movement Primitive Libraries through Probabilistic Segmentation”. In: (2017).

[18] Rudolf Lioutikov, Gerhard Neumann, Guilherme Maeda, and Jan Peters. “Probabilistic Segmentation Applied to an Assembly Task”. In: (2015).

[19] Guilherme Maeda, Marco Ewerton, Rudolf Lioutikov, Heni Ben Amor, and Jan Peters abd Gerhard Neumann. “Learning Interaction for Collaborative Tasks with ProbabilisticMovement Primitives”. In: (2014).

[20] Masahiro Mori. “The Uncanny Valley”. In: (2022).

[21] Christopher Olah. “Understanding LSTM Networks”. In: (2015). URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[22] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. “Probabilistic Movement Primitives”. In: (2013).

[23] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. “Using Probabilistic Movement Primitives in Robotics”. In: (2018).

[24] Deepika Phutela. “The Importance of Non-Verbal Communication”. In: (2016).

[25] Vignesh Prasad, Ruth Stock-Homburg, and Jan Peters. “Learning Human-like Hand Reaching for Human-Robot Handshaking”. In: 2021.

[26] Celene Reynolds and Emily Erikson. “Agency, Identity, and the Emergence of Ritual Experience”. In: (2017).

[27] Sudar. “Non-Verbal Greeting Of Different Cultures Used in Global Society”. In: (2018).

[28] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gell, and Bernhard Schölkopf. “Wasserstein Auto-encoders”. In: (2019).

[29] Phongtharin Vinayavekhin, Michiaki Tatsubori, Daiki Kimura1, Yifan Huang, Giovanni De Magistris, Asim Munawar, and Ryuki Tachibana. “Human-Like Hand Reaching by Motion Prediction Using Long Short-Term Memory”. In: (2017).

[30] Rachel E. Watson-Jones and Cristine H. Legare. “The Social Functions of Group Rituals”. In: (2016).

A. Hyper-Parameter Optimization

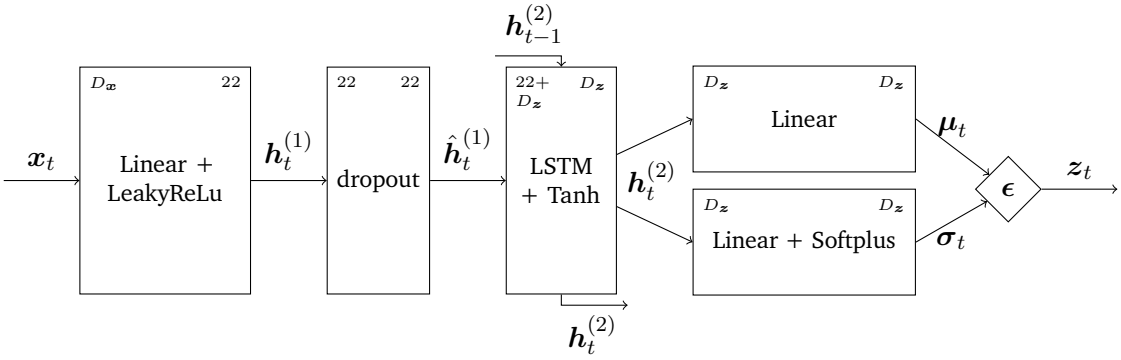
For each configuration, we performed 3-fold cross-validation on 5 models with randomly selected hyper-parameters. Each model was trained for 750 epochs and a fixed batch size of 40. The set of randomly chosen hyper-parameters was the same for all configurations. For the non-MTL configurations, hyperparameters affecting the RWAE could be chosen from different runs as hyperparameters affecting the classification. Subsequently, we allowed two choices for hyperparameters affecting both of them. We always chose the models with the lowest combined loss. In Table A.1 you find a list of all hyper-parameters including the possible range they could be selected from and the final selection for each configuration.

	domain	MTL RWAE			RWAE	Non-MTL Classification		
		SegW	Seg	No Seg		SegW	Seg	No Seg*
$D_{\boldsymbol{z}}$	$\{5, 6..., 20\}$	19	16	16	16	-	-	-
λ	$[0, 10]$	5.4558	7.9778	7.9778	7.9778	-	-	-
α	$[10^{-4}, 10^{-1}]$	0.0352	0.0467	0.0467	0.0467	0.0467	0.0680	0.0680
p_{dropout}	$[0, 0.75]$	0.1383	0.2937	0.2937	0.2937	0.2937	0.0595	0.0595

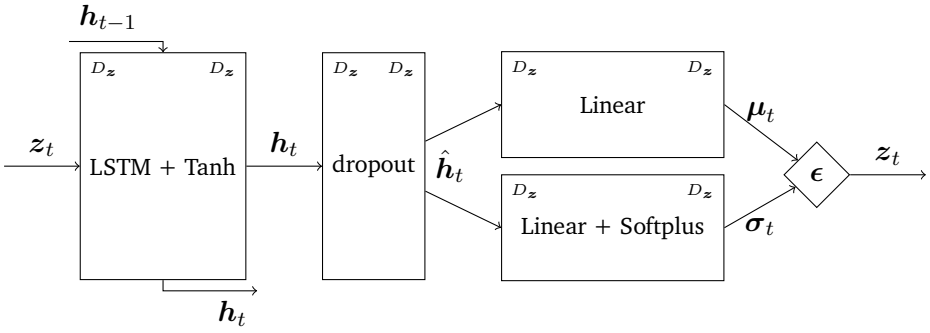
Table A.1.: A tabular view of all hyper-parameters that were subject to hyper-parameter optimization including their possible range and the final choices for them. Here α denotes the learning rate, $D_{\boldsymbol{z}}$ denotes the dimesionality of the encoding $\boldsymbol{z} \in \mathbb{R}^{D_{\boldsymbol{z}}}$, λ denotes the influence of the regularization term as seen in Equation 2.1 and p_{dropout} denotes the dropout probability. The hyperparameters for non-MTL are split into hyperparameters for RWAE model and hyperparameters for the classification model. Models marked with * have stopped their training early at the 500 epoch mark due to performance drops in-between checkpoints for the final models.

B. Model Structure

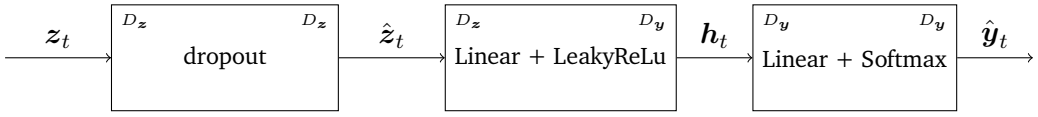
In the following, we describe the specific structure of the models' components used for our experiments. Figure B.1 illustrates the structure of the different components visually.



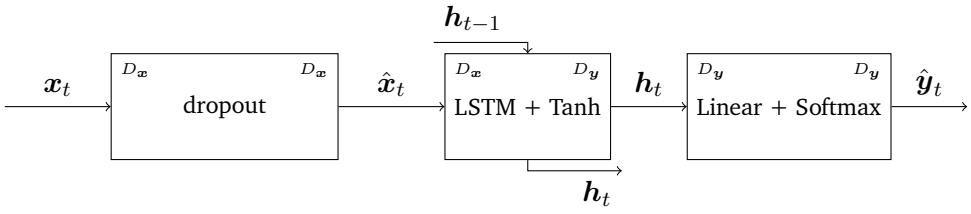
(a) encoder



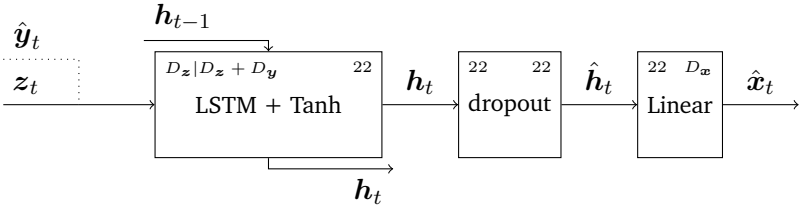
(b) prior



(c) classification (MTL)



(d) classification (No MTL)



(e) decoder (No MTL | MTL)

Figure B.1.: Structure of the different components of the RWAE in both the MTL and non-MTL setting. Each rectangle represents a corresponding layer with the in- and output dimensions marked in the upper corners. The ϵ-labeled diamond represents the sampling of z_t using reparameterization.

List of Abbreviations

AE	Auto-encoder
BIP	Bayesian Interaction Primitive
CEL	Cross-Entropy-Loss
DOF	degree of freedom
DRL	Deep Reinforcement Learning
DTW	Dynamic Time Warping
EKF	Extended Kalman Filter
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HHI	Human-Human-Interaction
HRI	Human-Robot-Interaction
IMQ	inverse multiquadratics
KL-divergence	Kullback-Leibler divergence
LSTM	Long Short-Term Memory
MP	Movement Primitive
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MMD	Maximum Mean Discrepancy
MSE	Mean-Squared-Error
MTL	Multi-task learning
NN	Neural Network
ProMP	Probabilistic Movement Primitive
SLAM	Simultaneous Localization and Mapping
VAE	Variational Auto-Encoder
VRNN	Variational Recurrent Neural Network
ReGen	Recognition and Generation
RNN	Recurrent Neural Network
RWAE	Recurrent Wasserstein Autoencoder

WAE	Wasserstein Autoencoder
ZCV	Zero Crossing velocity