TÉCNICO LISBOA

# SENSORIMOTOR GRAPH: Action-Conditioned Relational Forward Model Learning of Robotic Soft Hand Dynamics

**João Damião Mano Lopes Marques de Almeida**

Thesis to obtain the Master of Science Degree in

## Computer and Electrical Engineering

Supervisor: Prof. José Alberto Rosado dos Santos Victor

### Examinatiom Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. José Alberto Rosado dos Santos Victor
Member of the Committe: Prof. Pedro Manuel Urbano de Almeida Lima

**February 2021**

# Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

***Scientific knowledge is in perpetual evolution;***
***it finds itself changed from one day to the next.***

Jean Piaget

# Acknowledgments

First, I would like to sincerely thank my supervisor, Prof. José Santos-Victor, for his guidance and vision. You have granted me a unique, immersive, research experience that would not be possible without your utmost dedication and mastery at engaging me with so many brilliant agents in a common goal.

Also, I would like to express my warm deepest gratitude to Paul Schydlo, my mentor in the first day of University and in the last semester of graduation. Your tireless commitment and insightful orientations helped shaping this research project and gave me lessons for life.

Futhermore, I would like to thank Atabak Dehban who, with his experience and incisive feedback gave an invaluable contribution to this work.

My gratitude extends to all other people who helped me with this thesis - with valuable feedback or advice - but also to all people who brought me where I now stand.

To my partner, Raquel, whose support and laughter made 2020 a wonderful year; to my colleagues at Instituto Superior Técnico, who shared this journey of graduation with me; to my lifelong friends who always supported and shared my dreams: thank you all, deeply.

Finally, this master thesis is dedicated to my family. To my parents, grandparents and sisters, for all the love and support.

# Abstract

Soft robotics is a thriving branch of robotics which takes inspiration from nature and uses affordable flexible materials to design adaptable non-rigid robots. However, their flexible behaviour makes these robots hard to model, which is essential for a precise actuation and for optimal control. The problem of modelling a robotic soft hand is analogous to the processes of sensorimotor self-discovery and limb control that occurs in early stages of human life.

Moreover, when trying to model a system, its structured nature is not often accounted for, as there is not always an easy representation. Nonetheless, learning a system's connectivity is a valuable asset to understanding its kinematics and even predicting future dynamics. This issue has been addressed with Graph Neural Networks, which take advantage of system compositionality and order-invariance to combine artificial neural networks with graph-based representations.

With the goal of finding a modelling strategy for soft systems in robotics, and inspired by sensorimotor learning and recent work on Graph Neural Networks, we propose a self-calibration mechanism that learns the action-conditioned relational forward model of the non-rigid kinematic chain of a robotic soft gripper. We denote our model as the "Sensorimotor Graph" and we benchmark it against non-structured baselines to asses the model performance and robustness to increasingly adverse conditions.

We demonstrate that our model outperforms the studied baselines in basic scenarios while not being significantly affected by configurational variations, tracking errors or node failures, extending the state-of-the-art in directions that are key to perfecting soft robotics modelling and actuation.

# Keywords

Soft Robotics, Relational Forward Model, Graph Neural Networks, Sensorimotor Learning

# Resumo

A robótica *soft* é um ramo fluorescente da robótica, inspirado na natureza e que utiliza materiais acessíveis para conceber robôs não-rígidos versáteis. Contudo, o comportamento flexível destes robôs torna-os difíceis de modelar - essencial para serem atuados com precisão e para um controlo ótimo. O problema de modelar uma mão robótica mole é análogo no reino animal aos processos de auto-descoberta sensorimotora que ocorrem nas primeiras fases da vida humana.

Na modelação de um sistema, a sua natureza estrutural não é frequentemente contabilizada, já que nem sempre tem uma representação fácil, mas constitutui uma vantagem valiosa na compreensão da sua cinemática e na previsão de dinâmicas futuras. Esta questão foi abordada com as Redes Neuronais em Grafo (RNGs), que tiram partido da composicionalidade do sistema e da permutabilidade na ordem dos elementos para combinar redes neuronais artificiais com representações baseadas em grafos.

Para encontrar uma estratégia de modelação de sistemas robóticos não-rígidos e inspirados pela aprendizagem sensorimotora e pelo recente trabalho em RNGs, propomos um mecanismo de auto-calibração que aprende o modelo dinâmico, relacional e condicionado-por-ações, da cadeia cinemática não-rígida de uma garra robótica flexível. Denominamos o nosso modelo como 'Grafo Sensorimotor' e comparamos este modelo com modelos-de-base para avaliar o seu desempenho e robustez em condições crescentemente adversas.

Mostramos que a nossa solução supera os modelos-de-base estudados em cenários básicos e não é significativamente afetada por variações configuracionais, erros de seguimento ou falhas pontuais, estendendo o estado-da-arte em direções fundamentais para aperfeiçoar a modelação e actuação da robótica *soft*.

# Palavras Chave

Robótica Mole, Modelo Dinâmico Relacional, Redes Neuronais em Grafo, Aprendizagem Sensorimotora

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**SMG** Sensorimotor Graph

**NRI** Neural Relational Inference

**SOFA** Simulation Open Framework Architecture

**GT** Graph Theory

**MSE** Mean Square Error

**SR** Soft Robotics

**MPC** Model Predictive Control

**BL/s** Body Length per second

**ANN** Artificial Neural Network

**CNN** Convolutional Neural Network

**RNN** Recurrent Neural Network

**RvNN** Recursive Neural Network

**SVM** Support Vector Machine

**MC** Markov Chain

**GNN** Graph Neural Network

**ELBO** Evidence Lower Bound

**VAE** Variational Autoencoder

**LSTM** Long Short-Term Memory

**g.t.** ground truth

**MLP** Multi-Layer Perceptron

# List of Symbols

| | |
|---|---|
| $\Delta$ | Dynamics inference module |
| $\delta_a$ | Cable-pull actuation signal |
| $\Gamma$ | Connectivity inference module |
| $\mathbf{a_i^t}$ | actuation signal on object $i$ at time $t$ |
| $\mathbf{e}$ | Set of edges of a graph |
| $\mathbf{p_i^t}$ | position of object $i$ at time $t$ |
| $\mathbf{v}$ | Set of vertices of a graph |
| $\mathbf{x_N^T}$ | feature vector of N objects for T time steps |
| $\mathbf{z}$ | Connectivity graph of a system |
| $d$ | Number of edge types in the connectivity graph |
| $E$ | Young Modulus |
| $M_{burnin}$ | Burn-in temporal window |
| $p_\theta$ | Decoding function of the variational autoencoder |
| $ps$ | Temporal window prediction step |
| $q$ | Encoding function of the variational autoencoder |
| $v$ | Poisson ratio |

# 1

# Introduction

## Contents

## 1.1 Motivation

Following the trend of recent years, it is expected that we will continue to see a growing number of autonomous systems in the decades to come [1]. With an increasing coexistence of humans and sophisticated robots, soft robotics will play a major role: they are simple, affordable and adaptable [2].

Soft grippers, for instance, are hand-inspired end-effectors made of soft materials that promote smooth continuous movements, tight contact and shape adaptability. They have outperformed their rigid counterparts for many applications [3] and will continue to achieve breakthroughs in sectors as industry, agriculture and healthcare. However, they can be hard to model, and hence to control precisely.

In this thesis, we develop a simple bio-inspired self-calibration process for a soft robotic gripper that captures the relational dynamics inherent in the system of inter-connected parts. We propose a self-supervised framework that, by actuating on its flexible fingers (action-conditioned) and observing their trajectory response (using robot vision) is able to learn the kinematics of the hand and predict future dynamics. The self-supervised nature of such framework allows this to be applicable to any robotic soft system without the need for specific configurations or external supervision. We firmly believe that improving our modelling of the relational dynamics of robotic soft systems is the first step towards a better deployment and control of these end-effectors, and hence the normalization of their benefits: in autonomous locomotion, in industrial assembly lines, in bionic prosthetics and in medical surgeries.

## 1.2 Literature overview

This thesis uses graph-based networks and bio-inspired sensorimotor babbling to address the problem of learning an action-conditioned relational forward model of the non-rigid kinematics of a robotic soft hand. In order to reach a complete formulation of our problem statement and the strategy used to tackle it, we first need to understand its key concepts.

In this section, we provide a concise overview of the topics that constitute the background literature to this work and that we are going to expand throughout this thesis.

### 1.2.1 Sensorimotor learning

The father of developmental psychology, Jean Piaget, has dedicated his life to the study of cognitive development in children and provided many answers to the human learning process[4].

The day we are born (and even before), we start learning. One of the first things we learn, the simplest of all, is the control of our own body. And we master it before even knowing what physics or anatomy are. The period of building implicit body models in our brain that allow us to move our limbs in a controlled way and interact with the environment was baptized as *Sensorimotor Stage* [5]. The process occurs through the association of motor actuations and sensorial inputs.

For instance, in order to learn how to move his pointing finger, a baby repeatedly performs "random" short movements with the hand muscles to observe its effects and disentangle the complex

anatomy [6]. If we have knee surgery and spend a few months without walking, we may need to use similar strategies to "refresh" the subconscious sensorimotor models that allow us to efficiently use our legs.

Being a simple yet effective strategy learning process to understand complex interactions without any idea of the explicit equations that rule the system, this biological process can serve as an inspiration to learn the actuation for other intricate structures.

### 1.2.2 Soft Robotics

The word robot (*robota* - Czech for 'servant' or 'forced labour') was first mentioned in a fiction novel by Karel Čapek [7] and, since then, much science fiction has come true. The greatest contribution of robots has probably been to the manufacturing industry with a complete revolution of the secondary sector [3]. However, in the present day, robots are deployed in many more applications - from agriculture to health care, from defense to domestic assistance - and the trend seems to be growing for years to come [1].

Soft robotics is a sub-field of robotics which uses inexpensive soft materials to design robots that can safely interact with and adapt to their immediate environment better than the robots made of hard components. These robots take inspiration from biological soft systems to boost flexibility and adaptability. While rigid actuators brought speed and accuracy to automated processes, in order to manipulate delicate or irregular-shaped objects (like fresh fruit or a glass), a soft gripper would be the ideal choice.

In this thesis, we consider the non-rigid kinematic chain of a robotic soft gripper: it contributes to many tasks (manipulation, locomotion, etc) in numerous fields and it is complex and structured enough to be representative of soft systems.

### 1.2.3 Forward Models and Artificial Neural Networks

A model is a representation of a system - its behaviour or structure. Modelling is an essential step towards understanding a given system and it is necessary for most (model-based) control strategies [8].

One classical way of describing a system's forward model is to write the equations that rule that system. For example, to model a box being pushed, we know the applied force relates proportionally with the mass and the desired acceleration. However, as systems get more and more complex, explicit modelling becomes challenging or even unfeasible. Soft robots, making use of flexible elastic materials with infinite degrees of freedom, are an example of such systems.

Artificial Neural Networks have been used in machine learning to reverse this process: instead of explicitly describing a system and use those rules for modelling, they observe the system behaviour to extract inherent rules. This brings a promising solution to the limitations of classical approaches [9]. In this work, we propose one such framework.

### 1.2.4 Structured representations and graph neural networks

All systems, from simple geometries to living organisms, have an inherent structure [10]. If systems are interactions of different elements, the way these elements are organized is called the structure and it deeply affects their relations and behaviours.

When trying to model a given robotic system - to learn its kinematics or dynamics - the relations between the parts are often diminished or even ignored as there is no easy representation. However, since structure is a key part of any system, including relational knowledge in the learning process could help to achieve better modelling of our robot [11].

Graph theory [12] provides a robust representation for many types of structures - like grids, hierarchies, or meshes, - so the search for combining graphs with artificial neural networks has been a long-lasting quest.

More recently, with the debut of Graph Neural Networks [13–15], graph-structured representations were combined with neural networks with promising results in many fields. This constitutes the last building block to our claim: learning the relational forward model of the non-rigid kinematic chain of a robotic soft hand using Graph Neural Networks and inspired in human sensorimotor learning.

## 1.3 Objectives

With the purpose of learning a graph-based action-conditioned relational forward model of a robotic soft hand, we propose a novel self-supervised framework, the Sensorimotor Graph (SMG). In order to implement this strategy, we extend previous work on Graph Neural Networks for neural relational inference and adapt it to our framework.

The choice in this thesis to address the non-rigid kinematic chain of a soft gripper had three main reasons:

- Soft robotics is a prominent field with emerging applications and a lot of potential.
- Soft systems are a particularly relevant domain given the importance of understanding the complexity of their structure and dynamics for control;
- New developments in Graph Neural Networks let us explore its relational nature.

### 1.3.1 Original contributions

In this thesis, we propose the SMG model for learning the action-conditioned relational dynamics of a robotic soft gripper. Moreover, we seek to find if this graph-based framework increases the performance and robustness of the system modelling when compared to different non-structured models. In this respect, the main contributions of this thesis can be compacted as:

1. Extend existing graph-based relational inference networks to **integrate external actions** in the dynamics model;

2. Applying the action-conditioned graph-based forward model to the domain of a non-rigid **robotic soft hand** kinematic chain with varying configurations;

3. Benchmarking the proposed structure-based model against different non-structured baselines to **compare performance and robustness** to conditions with increasing adversity.

### 1.3.2 Thesis structure

This dissertation includes five chapters, organized in three main parts.

1. Motivation

   **Chapter 2: Sensorimotor Learning** - introduces the theory of cognitive development, soft robotics and forward models.

   **Chapter 3: Relational Systems** - presents the concept of structured system representations (with particular focus on Graph Theory) and Graph Neural Networks

2. Proposed solution

   **Chapter 4: Sensorimotor Graph** - describes our proposed framework and the implementation it takes inspiration from, the NRI model

3. Experiments

   **Chapter 5: Experimental Setup** - details how the data was generated, collected and organized, the structured models that constitute our baseline and the different experiments

   **Chapter 6: Results** - introduces the performance metrics used and expands on the performance and robustness of the SMG and the baselines in the different experimental tests

The first part focuses on providing the reader with a broad perspective and motivation concerning the different topics that are related to our work. Chapters 2 and 3 introduce the problem statement, the strategies that have been used in the past to solve it and the necessary building blocks to a new solution. The bottom-up structure of this part is depicted in Figure 1.1.

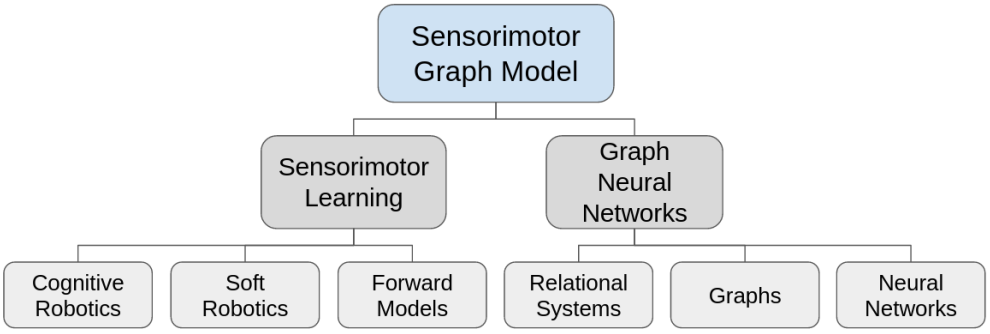



**Figure 1.1:** Motivational background as building blocks of the proposed solution

The second part focuses on the solution proposed in this thesis to tackle the challenge motivated in Part 1. After reading Chapter 4, one should have a comprehensive understanding of our proposed solution.

Finally, the last part consists of testing the proposed solution. In Chapters 5 and 6, we establish the experimental procedure, define the metrics and show the results that qualify and quantify the success of our solution across various dimensions. Moreover, in this part, we make some considerations and draw conclusions in light of the results.

# 2

# Sensorimotor learning

## Contents

We are what we learn. Like eternal apprentices, we constantly absorb knowledge, experiences and values that define us as conscious actors. As a central aspect of intelligent behaviour, learning has been in the spotlight for many years now: first in psychology [4], then in artificial intelligence and robotics [16]. Out of the many open questions that build up the debate of 'How do we learn?', a particularly significant topic that is extensible to robotics concerns cognitive development: how do we learn to understand the physical world?

In this work, we propose a framework to independently learn the structure of a soft robot end-effector, inspired in how newborns acquire sophisticated motor skills. In the same way as an infant learns to use his limbs by observation and experimentation and without any explicit knowledge of world physics or human anatomy, a sensorimotor-learning-based self-calibrating process could be applied in robotics to easily empower these with a latent understanding of their structure and dynamics. A robotic system that, by observing its own hand and without the need for external supervision, creates the appropriate mapping between actions and fingers and the appropriate model of the flexible elastic fingers' dynamics.

Self-calibration is important in the field of robotics, particularly for soft robots. To begin with, robotic systems, just like humans, are required to "have" a good understanding of their inherent kinematics to act on the exterior world. Robots are equipped with sensors and motors which can contribute to sensorimotor learning. Although soft robotics has not yet received as much attention as its rigid counterpart, a sophisticated self-calibration dynamics-learning strategy is certainly much necessary in soft systems, for three main reasons: 1) the flexible and elastic nature of the soft materials leads to a more unconstrained motion of the kinematic chain, which is more difficult to model; 2) the elastic contact of soft gippers with the manipulated objects alters their shape and kinematics over time; 3) inherent imperfections that some manufacturing techniques carry and the mechanical and abrasive wear of the rubbery materials call for self-calibration [17, 18].

The subject of our bio-inspired structure learning framework will be a non-rigid robotic soft hand kinematic chain. Soft grippers constitute relevant ubiquitous end-effectors and are representative systems of soft robotics.

This chapter is organized into three sections. We will start by our inspiration: understanding how newborns learn to control their body parts and what such strategies have been replicated in robotics. Then, we will focus on the domain of soft robotics (with its recent achievements and challenges), and the particular case of soft grippers. Finally, we shall look at the different modelling strategies that have been applied to robotic system dynamics.

## 2.1 Cognitive Robotics

From the moment we are born, we never stop collecting lessons, data and experience. We start by learning about ourselves - our body, our senses, our consciousness, - we learn about the world around us, we learn communication, we learn values, we learn social interactions and, by the age we start school, we are extremely complex knowledgeable beings. Of course, we don't learn from

scratch. We are born ready to learn. Millions of years of evolution climaxed in brain structures predisposed for cognitive development [19]. The importance of our innate ability to learn versus the relevancy of a stimulating environment is an age-old philosophical debate.

The way humans collect knowledge about the world and themselves is a much-studied open question, whose answers start to cast light onto many relevant applications in psychology, medicine and engineering. In the case of artificial learning algorithms and robotics, looking for inspiration in human biology and cognition seems a promising path for replicating these processes [20].

In this section, we will introduce some relevant topics that served as inspiration for this thesis. We will start by presenting Piaget's theory of cognitive development, in particular the sensorimotor stage. Next, we will introduce a much-used form of sensorimotor learning in infants: motor babbling. Finally, we will discuss how sensorimotor learning, and particularly motor babbling, can and has been used in different engineering applications.

### 2.1.1 Sensorimotor Stage of Cognitive Development

The Swiss psychologist Jean Piaget dedicated most of his life to understanding the nature and development of human knowledge. In his theoretical work on cognitive development, he expands his belief that children continuously alternate between building their perception of the environment and adjusting that construction to the discrepancies that they find through senses. [5]

In his books (e.g. [21–23]), which became a reference in psychology and neurosciences, Piaget proposed that humans progress through four developmental stages from birth to adolescence [4]. These four stages are described in Table 2.1.

| **Cognitive development four stages** | |
|---|---|
| **Stage** | **Description** |
| *Sensorimotor (birth - 2yo)* | Where infants develop a permanent sense of self and objects. Includes six stages that involve the use of reflexive behaviours, habits, hand-eye coordination, intentionality and curiosity. |
| *Pre-operational (2yo - 7yo)* | Includes language development, magical thinking and the use of symbols. Still lacks concrete logic, perspective and some mental operations. Includes a symbolic function sub-stage (when children are able to understand, represent and remember objects in their minds) and the intuitive thought sub-stage (when children want to understand everything and ask many questions). |
| *Concrete operational (7yo-12yo)* | Characterized by the acquisition of logical thinking to solve problems. Children are capable of inductive reasoning but deduction is still troublesome. They start understanding ethics and social matters, being able to view things from other's perspectives in concrete events. They still struggle with abstract reasoning or hypothetical situations. |
| *Formal operational (from 12yo)* | Intelligence is demonstrated through the logical use of symbols related to abstract concepts. The individual is capable of hypothetical and deductive reasoning, as well as metacognition. |

**Table 2.1:** Four stages of cognitive development according to Piaget

Particularly relevant to our case is the first stage of cognitive development: the sensorimotor stage. It lasts approximately 2 years and is a period of rapid cognitive growth where infants develop an understanding of the world and themselves through trial and error. It is called sensorimotor stage

since it is through his innate senses - sight, hearing, smell, taste and touch - combined with the motor abilities developed - touching, grasping, biting - that infants gain a basic self-awareness and an understanding of the world around them.

In a first sub-stage of the sensorimotor phase, the learning process is based on reflex acts. It is "the coordination of sensation and action through reflexive behaviours" [24], such as sucking objects touching the mouth, grasping objects that contact the hand palm, blinking, smiling or following moving objects with the eyes. It is the most preliminary phase of learning, a primary understanding of one's own body, how it perceives and acts on the world. Moreover, it is an essential phase for what follows: a controlled intentional action of the body parts and the adequate processing of the information that it captures.

### 2.1.2 Motor babbling

As we have seen before, the primary stage of learning is sensorimotor acquisition and its simplest form involves reflexive actions. This includes both unintended responses to external stimuli, like blinking or smiling, and exploratory actuations on the unknown environment, such as motor babbling [5, 25].

**Definition: Motor babbling**

The process of repeatedly performing an "arbitrary" motor command for a short duration to observe its effect on an unknown system.

Motor babbling is of paramount relevance for newborns in the context of cognition development during the period of acquiring their own body models as sensorimotor relationships. For the more experienced reader, this is analogous to, in signal processing, using the impulse response of a linear system for system identification. Although it might sometimes not be efficient (or even sufficient) to model high dimensional systems, it disentangles complex interactions and fits many applications.

In order to illustrate what motor babbling would feel like for the reader, let us imagine the absurd situation of being bitten by a supernatural cat and waking up the next day with a long flexible tail [26]. During the first hours or days, being a totally unknown body part to us, we would not know how to control it and would clumsily drop several objects around us to our passage. With only a few new muscles to control an elongated flexible sequence of caudal vertebrae, we could try motor babbling to help us understand its functioning: the quick contraction of muscle A would curl the tail, a relaxation of muscle B wold make this new appendage lower horizontally, and so on. Performing random short motor commands on single muscles would help us understand its effect on the tail, disentangle the overlap of multiple muscle actions and help to progress towards a controlled usage of the tail.

The use of this caricatured situation is intended to capture the reader's interest but should not suggest that motor babbling is something distant or unnatural. Not only have we all been newborns, adapting to our unknown complex body, but it can also apply in later ages [27] in the case of physiatry, stroke recovery, spinal cord injury rehabilitation, and many other situations where a childlike exploration is needed to reteach the brain how to command movement.

### 2.1.3 Artificial learning

Sensorimotor learning is a fundamental phase for our brain to create the necessary models of our body and its relation to the physical world. This is mostly done in the first months of our existence but continues happening throughout life: as our bodies change, our sensorimotor models need adjustment; when we spend a lot of time without using a certain limb, these internal models need "refreshment" during recovery. But the human brain might need this sensorimotor phase in another circumstance: to adapt to artificial biomechatronic body parts.

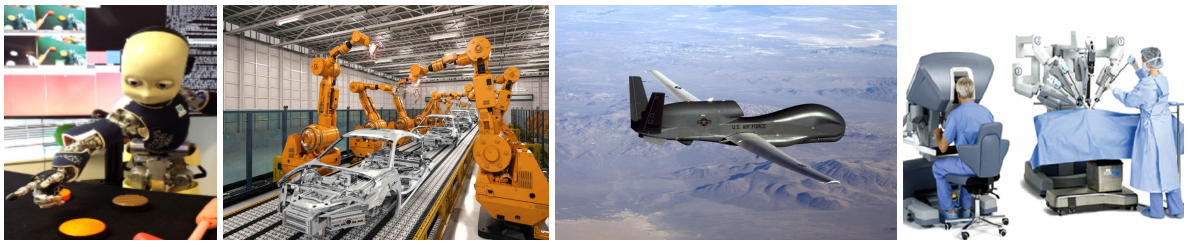
With the advances in medicine and engineering, we are assisting to a *cyborgization* of the human body [28] - the inclusion (or combination) of mechanical or electronic devices in organic systems. Whether to enhance human abilities or to replace missing body functions, the embodiment of biomechatronic parts is increasingly widespread. While some of these are technologies the human body adjusts quickly to (like pacemakers), some others might involve longer adaptation periods. It is the case of prosthesis, for which, as controllable limbs, some sensorimotor adaptation phase is required. Just like before, an exploratory phase is necessary for the brain to sculpt the physics of the new body part. Another example is cochlear implants on deaf individuals [29] who will go through prelexical (instead of motor) babbling to learn how to speak.

Moreover, as we seek to teach artificial systems to learn, we often seek inspiration in our own cognitive experiences. It is the case of robot learning, where motor babbling carries promising results in varied real-life applications [30]. With the deployment of this bio-inspired technique, robotic systems can autonomously develop an internal model of its body and environment, as in [6, 31, 32]. For instance, for learning sensorimotor control for robotic eye saccades or hand-eye coordination, motor babbling might constitute the first stage, followed by planned movements [33]. A recent study [34] showed exciting results at teaching a legged robot to walk by using only a few minutes of motor babbling to create an initial inverse map and using it for the task of locomotion.

In particular, bio-inspired motor babbling has been addressed for robotic grasping applications. The model introduced by Demiris [35] learns multiple forward models without any prior information by means of motor babbling and a Bayesian belief network. Oztop et al's motor babbling based models [36, 37] are capable of grasping without visual feedback (only sensory stimuli) using Hebbian learning and inverse kinematics. Takahashi [38] studies tool-body assimilation using motor babbling with robots grasping tools in different ways to learn its body schema and tool functions for each grasping position.

## 2.2 Soft Robotics

Although *automata* have a millennial history [39], electronic autonomous robots came into existence in the second half of the twentieth century as an intersection of computer science and engineering. Robots have been mainly designed to cooperate with humans - side by side or with remote control - or to replace them - in monotonous or dangerous tasks. Nowadays, robots are indispensable in industry or agriculture and show superhuman results in healthcare or defense (see Figure 2.1).



**(a)** humanoid robot (source: VisLab ISR)

**(b)** automated assembly line (source: Expense Reduction Analysts)

**(c)** unmanned aerial vehicle (source: Military.com)

**(d)** automated surgery (source: [40])

**Figure 2.1:** Traditional hard robotics in different fields

Robotics combines mechanical and electrical engineering to create the components responsible for sensing, actuation, locomotion or manipulation that constitute the cybernetic agent. Quite recently, there has been a trend of moving towards more adaptable robots, bio-inspired shapes and flexible materials. This led to the birth of what has was baptized as Soft Robotics (SR).

**Definition: Soft robotics**

A sub-field of robotics focused on endowing robots with soft materials and bio-inspired motor capabilities that allow adaptive, flexible interactions with unpredictable environments.

After an overview of cognitive learning and its recent promising deployments in growing fields, in this chapter we will focus on describing the concrete application domain of our work. In the following sections, we shall understand: 1) what motivated this new wave of robotics; 2) its main applications; and 3) the limitations or challenges for the future.

### 2.2.1 Seeking adaptability

For many decades, manufacturing robots have been designed to be stiff so that they can perform fast, precise and repetitive position control tasks in assembly lines. For this purpose, common actuators are composed of rigid electromagnetic components or metallic internal combustion engines [41].

However, we notice that in the natural living world, soft materials prevail. Most animals are soft-bodied (like worms or cephalopods) and many others live part of their lives as such before developing exoskeletons. In total, invertebrates may constitute more than ninety-five per cent of all life forms [42]. Even the animals with vertebral column and endoskeletons, like ourselves, are mainly constituted of soft tissue and liquids. In fact, of our different task-specific body systems, all but the skeletal system are purely soft-composed.

This trend is no accident in evolution. This soft structure helps animals adapt to complex ever-changing environments and provides numerous advantages. First, they can conform to different surfaces, distribute stress over large areas, increase contact time and lower impact force. It is the case of the soft palms of mammalian runners that damp the force of impact when their legs strike the ground

- Figure 2.2(a) - or the soft finger pads of arboreal mammals that contribute to climbing adhesion. Second, their flexible deformable bodies enable them to fit small apertures and different shapes. It is the case of many sea species when they need to find shelter or hide - Figure 2.2(b). Finally, a soft structure provides adherence and fluidity in body movements that help in locomotion or grasping. For example, a snake's elongated flexible body not only helps it moving easily in different surfaces and inclinations but also allows this suborder to trap and hunt large mammals - Figure 2.2(c).

**(a)** tiger cub paw (Source: Pixabay [43])

**(b)** octopus hiding (Source: Pixabay [43])

**(c)** resting snake (Source: Pixabay [43])

**Figure 2.2:** Advantage of soft stuctures in the animal kingdom

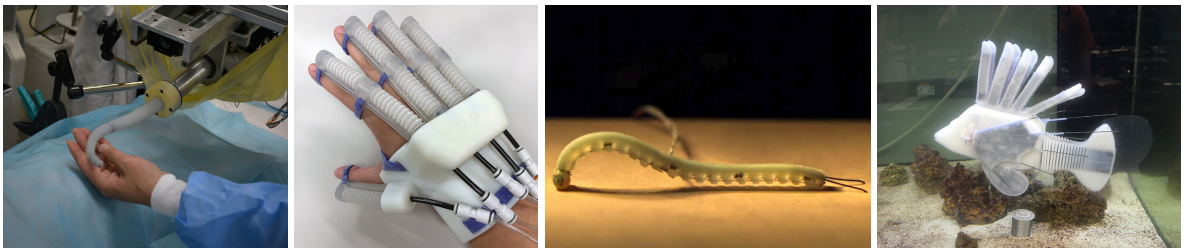
Also in every day robotic applications there are many situations that require similar attributes - adaptability, flexibility, adherence, efficient contact - and that require similar tasks - locomotion, manipulation. For this reason, the robotics scientific community has recently started seeking inspiration in the valuable lessons that invertebrate life may offer.

Although the history of soft robots can be traced back more than half a century, it was only in the last fifteen years that this field started drawing attention. Initially referring to rigid robots with compliant joints and variable stiffness, SR is now a multidisciplinary growing field [2] involving soft material-based structures with multiple horizons for applications.

### 2.2.2 Grasping new horizons

In recent years, SR has established itself as one of the fastest-growing topics in the robotic community [44], with the potential to revolutionize the role of robotics in society [1, 45]. Just like 3D printing has done for industry, the ubiquity and lower cost of soft robots raw materials (silicone or rubber instead of metallic alloys) may easily be a step forward in bringing robotics to every household. While that may be too soon to tell, the fact is that its adaptability and organicity is already bringing promising improvements to many fields - Figure 2.3.

In industry and agriculture, larger hard machinery is not always the best choice. For some jobs in manufacturing that involve more uncertainty or additional care, SR can be more thorough than bulkier rigid actuators [3]. Also, SR can include more durable materials and withstand heat, heavier objects and water, that electrical equipment could not resist [50]. Since food items are often one type



**(a)** surgery system [46] **(b)** hand rehab [47] **(c)** robotic caterpiller [48] **(d)** robotic fish [49]

**Figure 2.3:** Soft robots in different fields

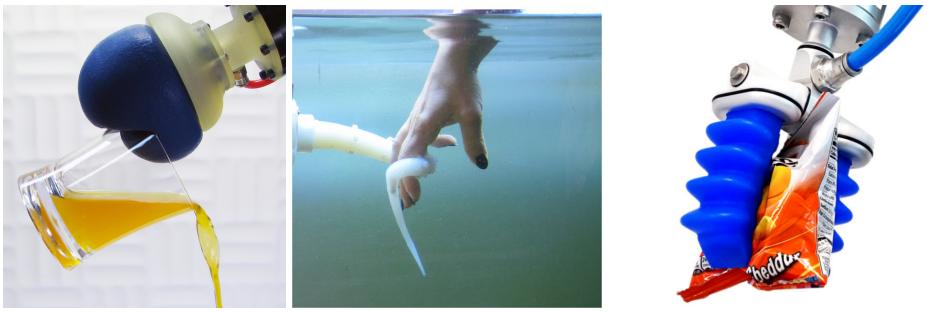
of products that requires careful handling, the agricultural sector has increasingly been using SR to manage crops from cultivation to harvest [51, 52].

In healthcare, soft robots have been assigned to diverse tasks. From the surgical perspective, minimally invasive surgery (MIS) has been prone to adopt SR techniques given their compliance with natural tissues [53–56]. Regarding wearable robots for rehabilitation, biocompatibility is vital and the use of elastic soft materials with similar mechanical properties to living organisms has a strong claim over conventional rigid cybersystems. This includes soft exoskeletons for limbs [57–59], posture [60], oral intervention [61] and even for animal rehabilitation [62]. Moreover, SR techniques have been valuable in robotic prosthetics to provide delicateness, precision and organic feeling to the artificial body parts.

New forms of robotic locomotion bring interesting potentials in fields like construction, exploration or defense. One of the most known examples is GoQBot [48], a robot that mimics caterpillar rolling for locomotion. Worm-inspired robots may use abdominal contraction [63] or peristaltic movements [64] for locomotion and reveal external shock absorption. A pneumatic untethered mobile soft robot is proposed in [65], which is capable of carrying heavy payloads and endure different adverse conditions. Crawling and jumping over rough terrains is also addressed in [65–67]. There have been also many approaches for water locomotion: propelling elastic shell-based models [68]; shape variating oscillators for aquatic propulsion [69]; hydraulically powered fish-like soft robots with three-dimensional swimming [49, 70]; octopus-inspired robots that travel through small apertures and unstructured surfaces [71, 72].

It is worth emphasizing a relevant application both to the field of SR and to this thesis: grasp - Figure 2.4. Above, we have mentioned some applications that involved soft grippers (dangerous procedures in manufacturing, manipulating food items in agriculture, hand exoskeletons or prosthetics). However, gripping technologies no longer work just as end effectors but as enablers of other key functions of autonomous robots like locomotion or body shape control [73]. Indeed, many SR developments are grasping-oriented, leading to even more applications that involve gripping and object manipulation. For instance, fast mobility or climbing [74, 75] (where soft grippers facilitate adherence); human-robot interaction (HRI) [76] (where the organic feel is more natural and compliant); remote tasks [77] (where SR allow much more sensitivity when being remotely controlled).

We have addressed the advantages of SR, including its flexibility, adaptability, adherence and



**(a)** vacuum gripper [41] **(b)** cephalopod limb [41] **(c)** two-finger gripper [78]

**Figure 2.4:** Different soft gripping applications

efficiency. But invertebrate robots, in particular soft grippers and soft limbs, bring us another major advantage, simplicity. If we think about a rigid robotic hand, there may be as many control signals as finger joints, or even more. Similarly, if we imagine a human-like leg, for proper locomotion we may need to actuate on the leg-hip connection, on the knee, on the ankle rotation and maybe on the foot position. This makes actuation unnecessarily complex for many applications. Most soft limbs - whether they are fingers in soft grippers or an octopus arm in a squid-like vehicle - may only use one actuation signal. As these limbs can only contract or distend, it usually takes only one value to characterize the limbs desired state. This can be imposed by a wire, air pressure, an electric field, light or temperature and it is more than enough to make the soft gripper delicately grab an object, or for an egg-sized 'cheetah' to achieve 2.5 BL/s [79].

### 2.2.3 Challenges

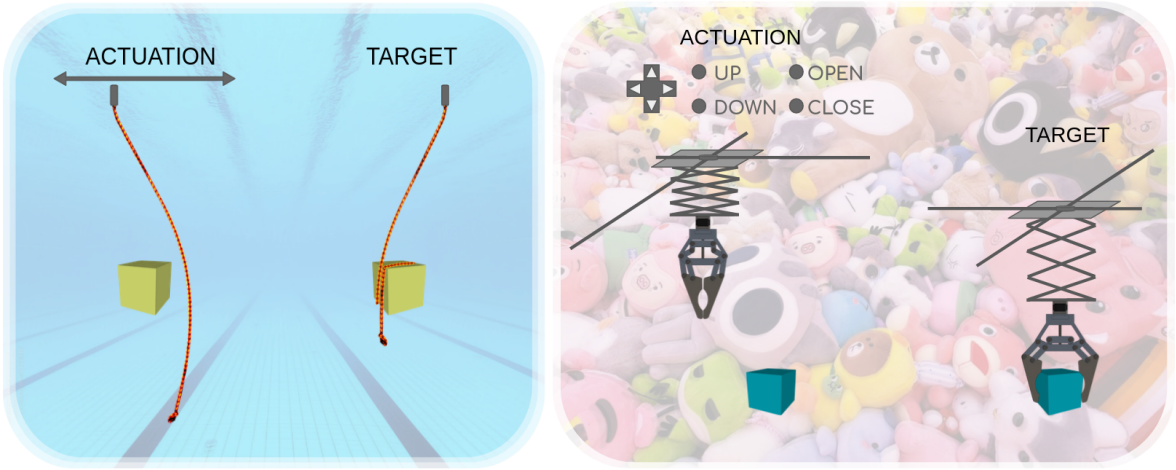
The fresh field of SR has caught many attentions and outperformed its rigid counterpart for specific applications. Its simplicity and adaptability, alongside the cheaper production materials, put SR on the verge of success. However, there are also some limitations of these robots and challenges for the next decades.

One limitation of soft robots concerns elasticity fatigue. Depending on the characteristics of the flexible materials and the usage near its ultimate strain, elastic materials may lose some properties or become irreversibly damaged [17], which motivates the need for self-calibrating mechanisms. Elastomers have been a polymer of choice given the wide range of commercially available formulations, low stiffness and high strain at rupture [73]. However, the use of elastomers has generally meant a lack of lifelong robustness, though effective solutions have been demonstrated and commercialized [78, 80].

Moreover, some types of soft end-effectors seem to be conditioned on the gripping method and its properties. Since there are various techniques of grabbing objects with soft grippers - vacuum, adhesion, electroadhesion, dry adhesion, etc - and each process is task- and environment-specific, a soft gripper designed for certain applications might turn unusable if some conditions change. For instance, electroadhesion and dry adhesion work well in a vacuum but are not effective in liquid

environments. Actuators relying on temperature can have different time constants if placed in liquids or in air. Pneumatic systems are particularly adaptable to different external pressures and cable-actuation - which will be used in this work - is even more robust.

Soft robots carry one challenge that is particularly relevant to our work. Hard robotics kinematics are based on joints and rigid components, providing a discrete topology with finite degrees of freedom. In SR, we notice the opposite: its soft, flexible, stretchable materials contribute to the continuum topology with infinite degrees of freedom; but the user only has very few actuation dimensions. The reason why it is possible for the control to be more constrained but the end movement being more flexible is because most of the extra dimensionality comes from the soft nature of the materials. But with the positive behavioural versatility of soft robots, comes the not so positive unpredictability of their movements.




**(a)** Soft rope dipped in water

**(b)** Rigid claw

**Figure 2.5:** Grabbing a target: discrete and continuous topologies with finite and infinite degrees of freedom, respectively

We can think of this problem - the one of actuating an elongated soft limb, like the octopus arm in Figure 2.4(b) - as holding an elastic rope by its top extremity in a swimming pool - Figure 2.5(a). We can only slide the tip of the dipped rope left or right (one degree of freedom) and we want the rope to grab a suspended object by wrapping around this target. The flexibility and stretchability of the rope, alongside the friction of the environment (in this case water) and our single dimension actuation, will make the task non-trivial. If we compare this to operating a multiple-command fully functioning claw machine to grab a specific target - Figure 2.5(b), - we can understand that dealing with soft materials and infinite degrees of freedom makes learning much more difficult.

The general problem of modelling the underlying dynamics of such complex soft systems is essential to actuate the soft robot properly. A fair model of a soft gripper dynamics will lead to significantly more precise manipulation. The problem has been addressed in many ways, some of which we will see in the next section. Moreover, manufacturing inherent imperfections and elasticity fatigue call for constant update of this models. The elastic contact with the manipulated object might in-

volve a change in kinematics that should be accounted for. By this time, we hope it is clear: 1) the non-trivial problem of modelling a soft system dynamics; 2) the relevance of a self-calibrating dynamics-modelling framework in an effective use of soft robots.

| **Soft Robotics** | **Traditional Hard Robotics** |
|---|---|
| Adaptive and tolerant | Limited adaptability to operate in unknown environments |
| High level of behavioral diversity | Low level of behavioral diversity |
| Organically inspired and compatible | Low level of bio-inspiration |
| Abundant inexpensive materials | High production costs |
| Continuum topology with infinite degrees of freedom | Discrete topology with finite degrees of freedom |
| Strenuous accuracy and lower speed | High accuracy and speed |

**Table 2.2:** Characteristics of soft and hard robotics

In order to summarize the benefits and drawbacks of SR, compared to traditional rigid robotic parts, the most relevant differences are compiled in Table 2.2, similar to [17].

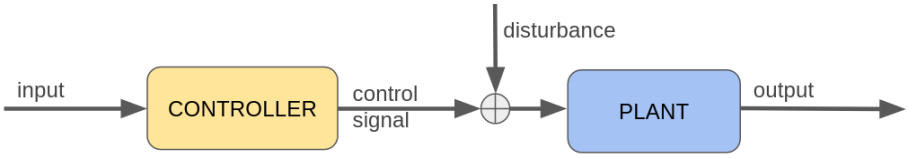
## 2.3 Forward Models

Now that we have seen our motivation problem in section 2.1 and our field of application in section 2.2, the challenge addressed in this thesis of learning to model, actuate and control unknown structured systems like robotic soft actuators should be more clear. But before wrapping up our problem statement, we ought to understand what actuation and control strategies exist and their benefits and limitations.

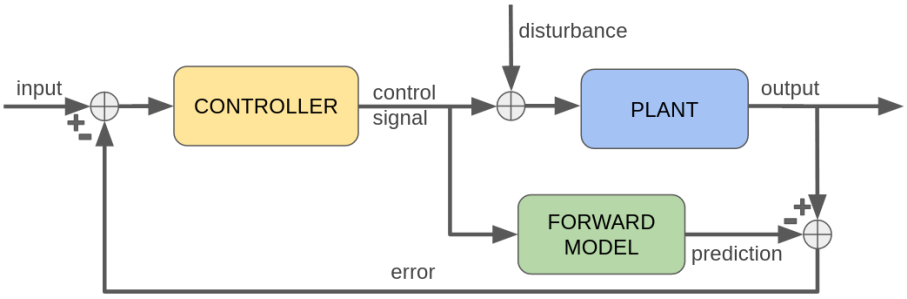
Pure model-free techniques have demonstrated state-of-the-art results in many applications by directly mapping observations to actions. Despite that, model-free methods (like model-free reinforcement learning) have many drawbacks and limitations, including a lack of interpretability, poor generalization, and a high sample complexity [81]. Alternatively, model-based approaches rely on learning a forward model directly from interactions with the real system and then incorporating the learned model into the control policy.

A forward model is an essential part of control theory [8]. It describes a system and is used to make predictions: 'Given a certain state and input, where is this system going in the next time steps?' which can also turn into 'If I contract a muscle of my new superhuman tail, how is it going to react?' or even 'If I act on a flexible quite-non-linear robotic soft gripper, how will it behave?'. Forward models describe a system as close to reality as possible to produce the best predictions. Sadly, the more intricate the system, the harder the task.

In the most basic scenario, with a forward model we can already engage in open-loop control, meaning that our actuation does not depend on the system output and therefore any disturbances can neither be detected nor corrected. It is the case of a heater controlled by a timer. This process can be improved with feedback control, where the output from the forward model (predicted body position) is compared to the output from the plant (body position) and this error is used by the controller to adjust the next actuations in order to smoothly converge to the desired input reference. There are many




**(a)** Open-loop control: no feedback




**(b)** Closed-loop control: input is updated by forward model's prediction feedback

**Figure 2.6:** Open- and closed-loop control

distinct ways to implement this simple strategy with different levels of complexity, depending on the controlled system and application.

Although the control of our system is out of the scope of this thesis, it felt not only relevant but also necessary to include this section with a short overview of forward models and control frameworks. This thesis proposes a model that learns the dynamics of a robotic soft hand by sensorimotor babbling. As the end goal of this thesis is to couple this model with a control framework to provide a smooth actuation of soft end-effectors, the reasons to include here a discussion of this second part were two:

- Contextualizing and justifying some strategic or operational choices in this thesis;
- Motivating some future work by explaining the natural next step for the contribution in this thesis.

This section will be divided into three parts: we shall start by briefly inspecting different types of forward models, from simple strategies to adaptive control and learning-based approaches; then, we will detail why do we use action-conditioned forward model learning; and finally move on to looking at distinct state-based representations, as well as their (dis)advantages.

### 2.3.1 Classical and learning-based approaches

In classical hand-designed approaches, the system model and parameters are assumed to be known in advance, and the controller design makes use of this information [82]. If the system kinematics can be accurately described and if they are constant in time, they can be used to model the system. However, there are several limitations to this. When the complexity of the system increases, it becomes very difficult to explicitly write precise and detailed model equations, thus hampering the model-based control system design. An example of a simple but highly non-linear system is a pair of

wheels moving in the snow. Moreover, it assumes the future parameters will be much like the present, which does not always apply: the control action ruling an aircraft is dependant on its weight but this changes as fuel is consumed.

Adaptive control emerged as a partial solution to this problem [83]. By updating the controller, it allows parameters of the model to be estimated - like friction or the weight of objects, - adapting to some levels of uncertainty. However, this approach is limited to a fixed structure of the model, which can limit its application to complex systems.

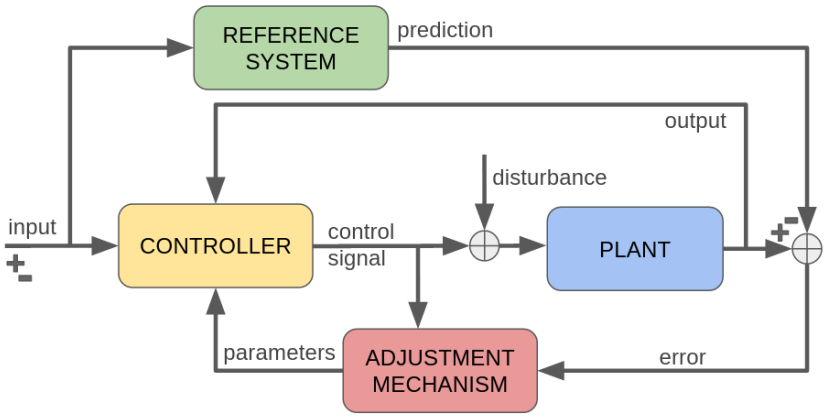



**Figure 2.7:** Adaptative control representation: an adjustment mechanism updates the controller

This limitation has lead the community to consider learning-based approaches to tackle the effects of under-modeling [84, 85]. Reinforcement learning [86] is one such method which learns to model and control a system by learning to map a system state to a control signal, trained by optimizing a reward signal. The system learns to control implicit system dynamics. Compared to adaptive control, this trial and error approach resembles more how humans learn some movement strategies. It is the case of a learning agent interacting with an environment, executing actions and transitioning between states in order to maximize its reward. This can be the situation of a computer game player trying to explore and win a game or a dog learning to 'sit' or 'roll over' and getting a cookie every time the trick is performed right.

Notwithstanding, reinforcement learning assumes the availability of interaction samples and a reward signal, which can be costly to collect. Besides the curse of dimensionality, reinforcement learning in (soft) robotics suffers from the curse of real-world samples [86]: expensive materials call for safe exploration; the change in the soft robot dynamics and external factors might make the learning process to never fully converge; slow convergence and high variance improves the risk of material damage and elasticity fatigue; time discretization might generate undesirable artefacts.

Model-based control methods, on the other hand, decouple learning of the system dynamics and control of the system, reducing the need for costly interaction samples. They use information about the dynamics of the system's structure and its behaviour in time to obtain a better control result regarding the stability and performance of the controlled system. Optimal control formulations based on differential system dynamic model [81, 87] optimize a cost function conditioned on the system dynamics, reference state and actuation signals. Model Predictive Control (MPC), for instance, solves,

for each time step, the optimization problem:

$$\underset{x_{1:t}\in\mathcal{X},u_{1:T}\in\mathcal{U}}{argmin}\sum_{t=1}^{T}C_t(x_t,u_t) \quad \text{subject to} \quad x_{t+1}=f(x_t,u_t), x_1=x_{init} \tag{2.1}$$

where $x_t$, $u_t$ are, respectively, the system state and control at time $t$, $x_{init}$ is the initial state, $\mathcal{X}$ and $\mathcal{U}$ are, respectively, constraints on valid states and controls, $C_t : \mathcal{X}$ x $\mathcal{U} \rightarrow \mathbb{R}$ is a (possibly time-varying) cost function and $f : \mathcal{X}$ x $\mathcal{U} \rightarrow \mathcal{X}$ is a dynamics model. A differentiable model is a necessary condition to be coupled with the MPC cost function and to then run optimization algorithms. Although a convex model is ideal, in practice modern optimization algorithms handle non-convex formulations, as in [81].

### 2.3.2 Action-conditioned learning

In model-based control approaches, the system dynamics learning and the system control are separate tasks, but they are not independent. In order to apply control to a system (using control actions to stabilize a system or reach a desired target state), a comprehensive understanding of the dynamics of such system is much valuable [88].

For this first part, the task of learning the dynamics of our robotic soft hand, in the form of forward models, we not only have vision-based data (showing how the several hand parts are moving in time when responding to muscle/wire contractions) but we also know *a priori* the quantified values of those input contractions. As we want to create a framework that capacitates a robot to learn the dynamics of his hand by sensorimotor babbling and observation, it is logical that the robot should have access to the actuation signals being fed to the end-effectors. The problem of learning a forward model without action-conditioning information would be less trivial and correspond to an infant who learns to use his own hands only by observing other infants hand movements.

In this thesis, we use the model-based control overview to propose learning the model of the dynamic soft hand. The natural following step - the control of the hand - is out of the scope of this thesis but its discussion here was justified for reasons that were mentioned in the introduction of this section and could be achieved using an MPC framework, as briefly described and as used in previous work on soft robots [89, 90]. For model learning, we adopt a self-supervised setting, where the own data provides supervision for the task, without the need for rewards or labels.

### 2.3.3 State-based representations

When trying to learn the kinematics of a robotic system using computer vision, learning-based approaches are generally based on two kinds of representations: structure-based, (e.g. two- or three-dimensional coordinates), or directly on the image space.

Image space representations take as input a sequence of images to predict future frames, modelling the dynamics of the environment in pixel space. This kind of model acts either by encoding the sequence into a compressed representation of the information contained in the frames - the so-called latent space [85] - and then learning dynamics over this space or directly warping past pixels into the

future. However, this type of representation is not invariant to distribution shifts such as lighting or camera pose changes.

Methods based on explicit structured representations on the other hand are lighting and camera pose invariant. These approaches are based on either manually annotating the data, including additional instrumentation in the form of markers, or learning to extract and identify key-point representations in an unsupervised manner [87, 91, 92]. More structured representations have gained attention with the recent growing adoption of Graph Neural Networks, which enable learning-based methods to exploit the structure inherent in the problem domain [87, 93] (see next chapter).

In this thesis, we focus on the question of how to best model a non-rigid kinematic chain, that is, a dynamical system of interconnected parts. We will assume explicit three-dimensional point position representations, which allow us to limit the scope of this work to the problem of modelling the non-rigid kinematic chain of a robotic soft hand. Additionally, this representation is invariant to lighting geometry and specularity and camera pose changes [94]. In a real system, the points can be obtained either by placing markers on the different fingers or by learning to segment an RGB-D camera stream into different finger segments.

## 2.4 Conclusion

In this chapter, we have laid the foundations to support our problem statement with three interconnected topics in three complementary sections. We motivated our problem in:

- Section 2.1: when we described the challenge of cognitive development in the early stages of human life and how bio-inspired learning approaches seem to be promising in artificial systems.
- Section 2.2: when we motivated our chosen domain - the promising field of soft robotics - and how soft-structured systems are difficult to model and have such thrive for self-calibration strategies.
- Section 2.3: when we reviewed some most relevant control strategies and confirmed the need for a self-supervised action-conditioned strategy with no rewards or labels and a distribution-shift-invariant representation.

The concepts defined in this chapter and their relations are illustrated in Figure 2.8 for a better understanding.

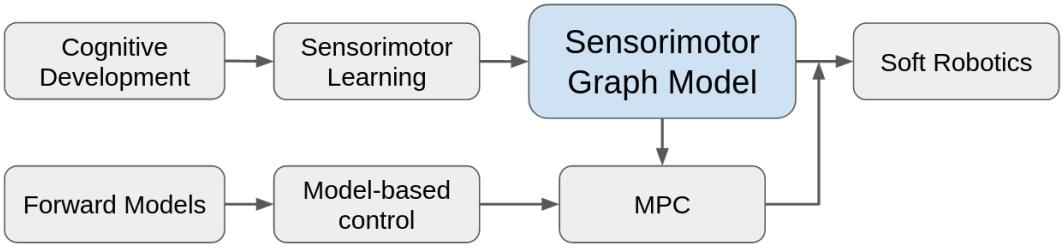



**Figure 2.8:** Chapter 2: A schematic representation of its main concepts

In this sense, we are now ready to reformulate our problem domain, in the light of what we learned about cognitive robotics, soft robotics and forward models:

**Domain**

In this work, we propose to learn a structured differentiable action-conditioned dynamics model of a non-rigid robotic (soft) hand inspired by how newborns acquire controlled motor capabilities.

# 3

# Relational Systems

## Contents

Aristotle wrote in his *Metaphysics* that "the whole is something besides the parts". The *Iliad* is not just the sum of the thousands of letters in it and a soup tastes different from the ingredients all together. This was used to justify emergence but it motivates probably the first definition ever of a system.

Two thousand years later, Bertalanffi [10] proposed a simple yet muscular definition of a system as "a complex of interacting elements". This (too) broad formulation - that includes atoms, galaxies, a rock or a cell - reveals the ubiquity of system theory. In an action-centered perspective, Perez [95] characterized a system as a set of congruent actions, which are combined to obtain a pre-established result and which are generated through a complex of five indispensable parts: energy, components, organization, structure and process. Systems may vary in their nature, application or relation with its surroundings, creating different types of classifications, some of which are presented in Table 3.1.

**Table 3.1:** Types of systems according to different criteria

| Criteria | System types | Definition | Examples |
|---|---|---|---|
| **semantics** [96] | **conceptual** | basic units of the system are words or symbols | a book or a mathematical model |
| | **concrete** | basic units are nonrandom accumulation of objects in physical space-time | object-centric systems |
| | **abstracted** | basic units of analysis are relations | social or biological systems with roles and associations |
| **nature** | **natural** | systems we find in nature | biological, cosmological or organic systems |
| | **technological** | systems created by human innovation | mechanical, electrical or information systems |
| | **socioeconomic** | systems that concern society and intelligent-agent mechanisms | social, economic or cultural systems |
| **boundary** | **bounded** | system that is fully containable in a bounded time and space | a cell or the United States highway system |
| | **unbounded** | system with no bound, that can expand indefinitely | social network or Wikipedia |

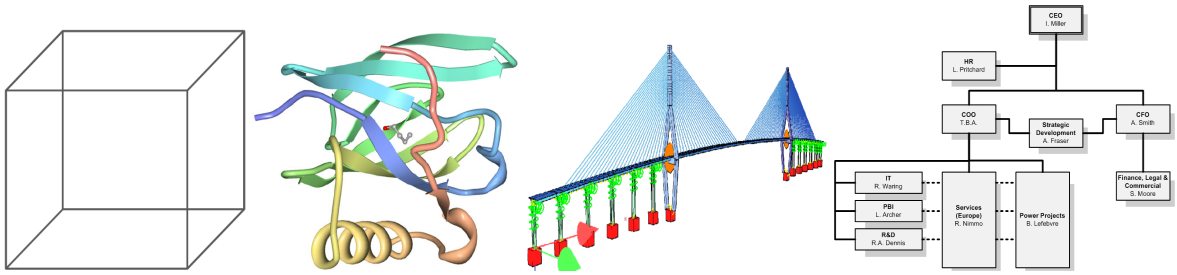
For all types of systems and definitions, we notice a common idea: it is not the sum of independent components but a set of elements that relate, that interact, that share actions, that have common processes or contribute to a common result. In the previous chapter, we have focused on dynamical systems. We have talked, often implicitly, about systems that evolve in time, their properties, some examples and how to model them. However, ignoring their relational nature is neglecting a fundamental part of their behaviours. Acknowledging a system's structure and relational bias - the assumptions that impose constraints on relationships and interactions among entities - will be a major part of understanding and modelling that system.

In this chapter, we shall focus on the relational nature of systems. In the first section, we will expand on structured systems, their ubiquity and representations. We will briefly talk about graph theory and the triumph of some structure-based methods over non-structured approaches. Next, we

shall introduce the Graph Neural Network, a recent class of neural networks that combines graph representations by using message passage mechanisms between nodes and edges and reach new application horizons. Similarly to what has been done before, we will describe for different fields how some approaches that use Graph Neural Networks have outperformed strategies that neglect the relational essence of the system or that use less powerful representations.

## 3.1 Structure Representations

Structure is a key part of any system. If a system is a group of interrelated entities, the relation and organization between the elements are called the structure. Just like in system theory, structure can be found everywhere (Figure 3.1): in geometry (3.1(a)), in a protein (3.1(b)), in a bridge (3.1(c)), in a company (3.1(d)). Maybe in the form of a hierarchy, a network or a lattice, structure is the embodiment of any system.




**(a)** Cube

**(b)** Porcine Odorant Binding Protein [97]

**(c)** Long span cable-stayed bridge [98]

**(d)** Organizational chart of Doosan Babcock Energy Ltd. [99]

**Figure 3.1:** The ubiquity of structure in everyday systems

Structure defines the relations and actions between the several components and therefore defines the system itself: a cube is only a cube while it keeps its three-dimensional isometric structure; a protein will stop working and a bridge will fall if their structure is broken down; changing the organizational structure of a company will change its essence and methods completely.

The fact that structured systems can be found everywhere is not just relevant *per se*, but it becomes particularly interesting when we notice structure is the driver of generalization [100, 101]. Object relations are the enablers of the learned knowledge transfer across peripheral dissimilarities: the predator-prey relation between a cheetah and a gazelle is similarly useful knowledge when applied to a golden eagle and a rabbit, even if their characteristics are substantially distinct.

Relational reasoning is also a central part of intelligent behaviour [11]: when we search for the shortest path on a map, pairwise distances between different crossroads must be inferred. Human ability (from a young age) to find unknown underlying structures is still unpaired for most systems.

In robotic systems, structure is fundamental and it often takes the form of kinematic chains: an assembly of rigid bodies connected by joints to provide constrained motion.

In this section, we shall focus on the importance of structure representations. We will start by a

general insight into structured systems and how this structure can be capitalized to achieve a better understanding of such system; then, we will introduce one of the most robust system representation fields, Graph Theory; after that, we will take a close look at the case of robotics, how structure is usually expressed and how it can be combined with graphs; and finally how graph representations have been used in artificial neural networks.

### 3.1.1 Capitalizing on structure

If we believe structure is an important part of a system, then we ought to use it to our advantage. From everyday tasks to complex applications, when one seeks to understand a system (identify it, use it effectively, predict its future) it may help capitalize on its structure.

As we said before, relational behaviour is the driver of generalization. If we are exceptional at dealing with unfamiliar elements of the physical world is because our brain models - the ones we developed in the early stage of cognitive development - exploit physical structures to deal with uncertainty and generalization better. When we arrive at a new restaurant and sit at the table to eat, we are able to dexterously interact in a complicated environment with dozens of different-sized objects even though we never laid eye on any of them before. We recognize some of them, we know what a dish is, that it is rigid and weighs around 250 grams. But then we complement our prior knowledge with observation, relational inference and generalization. When someone passes us an opaque jar of water, we mentally break down the system into a rigid container with an unknown amount of fluid that we can guess by observing how it is lifted and moved. If a weird-looking dessert arrives, we can infer its different parts and their consistencies just by observing the plate being passed around.

This unintentional disentanglement of reality for prediction purposes does not happen only for objects. When we are walking in crowded environments and want to predict the movement of the mass of people in front of us, understanding its structure - which individuals are connected in groups, which groups are walking towards each other, etc - helps significantly. But the process of exploiting a system's structure (its different components and their relations) to better understand its behaviour and properties is not just something we do unintentionally. It can also be a more or less planned strategy to tackle different problems.

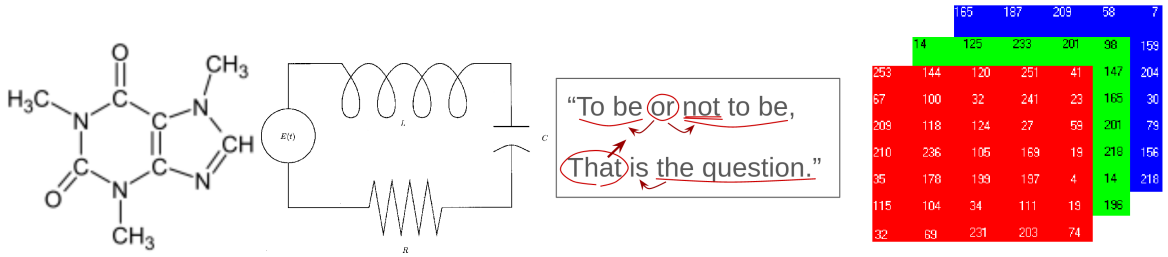
Let us imagine for one moment we select and open a book by chance, read a random paragraph and do the exercise of trying to guess the author. At first, we might look for hints on specific words: names of characters, places or dates. However, that is not enough. Hence, we turn to the structure to obtain more information about our system - the paragraph read. This would be the way words are connected, how sentences are punctuated, the literary style, the meaning of the written text. Although we would still need some talent and luck to correctly guess the author after this, I believe the reader can agree that this is a winning strategy compared to plain word search. In fact, this is a similar approach to what some text processing systems use, like author identification [102], plagiarism detectors [103] and novel clustering [104].

There are many other fields where the use of structure-based approaches have outperformed non-structured baselines. It is the case of chemistry, pharmacy and biology where biochemical information

in living systems or compounds is often standardly organized and which has been used for example in drug design [105, 106] or phylogenetic classification [107]. In pedagogy or psychology, a strongly structured essence can be found in many theories or techniques. In computer science, concepts like structural programming or structure-based testing reveal the importance of organized analysis. Relational behaviour is further found being used in algebra, market analysis or geography.

### 3.1.2 Graph Theory

As we have seen in the introduction of this section, there are many representations for structure (Figure 3.2): chemistry uses a very particular convention to represent molecular compounds (3.2(a)); in engineering, mechanical and electrical circuits have their own representation (3.2(b)); linguistics follow grammatical rules to create meaningful structures (3.2(c)); in computer science, data usually take the form of arrays of numbers (3.2(d)). Structure can even be represented through charts, through meaningful elements, through realistic renderings, etc. However, some representations gained particular attention and relevance given their robustness and generality.




**(a)** Chemical structure of caffeine molecule [108]

**(b)** Electrical circuit [109]

**(c)** Grammatical structure of famous quote [110]

**(d)** 3D matrix of RGB image [111]

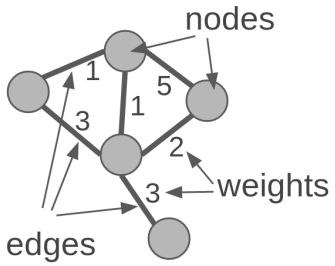
**Figure 3.2:** The ubiquity of structure in everyday systems

Graph Theory (GT) [12] is a field of discrete mathematics that focuses its study on graphs and its properties. A graph is a mathematical structure used to model pairwise relations between objects. Being applicable to a myriad of real-world systems, a graph is very close to a universal discrete modeller. GT can be traced back to the eighteenth century and it has helped to solve problems in different domains and to create ground-breaking axioms and dozens of present-day open problems. The key to the success of this field resides in the simplicity of its basic units and formulation, which makes it applicable to almost any system and scalable to increasing complexity. One of the most known theorems of GT that represents well its uncomplicated and universal nature is the four colour theorem and it remained an open problem for more than a century (until its computer-aided proof): "Given any map, the different regions can be coloured using at most four colours so that no two adjacent regions have the same colour" [112].

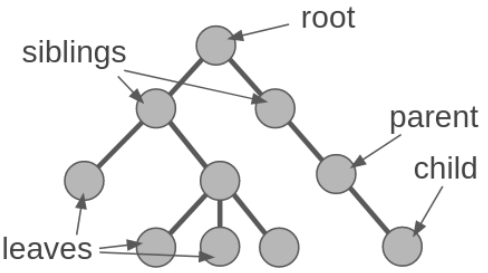
**Definition: Graph**

A graph is a network representation in the form of vertices (also called nodes or points) connected by edges (also called links or lines) which may have different weights.

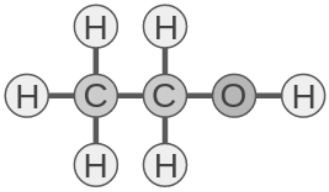
Different graph representations can be used depending on the underlying system. For example, graphs can be divided into directed or undirected graphs, depending on whether the edges have an orientation - from node A to node B - or are a mutual link - between A and B. For instance, a molecule - Figure 3.3(c) - is an undirected graph, while a piece of text - Figure 3.3(d) - could be directed as certain words must be followed by others. They could also be classified as sparse or dense graphs, depending on the number (actually, the density) of edges. Graphs can be cyclic or acyclic whether it is or not possible to move along the graph from node A to itself. Another relevant distinction is between weighted and unweighted graphs, the first type having distinct edge weights that represent a cost of moving along that edge (Figure 3.3(a)).



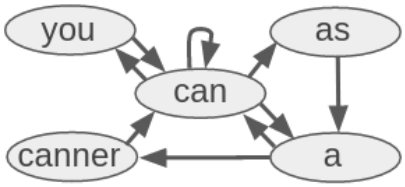

**(a)** Weighted graph




**(b)** Acyclic undirected graph (tree)




**(c)** Graph representation of ethanol molecule




**(d)** Graph representation of the tongue twister 'Can you can a can as a canner can can a can?'

**Figure 3.3:** Graph representations

A connected acyclic graph is also called a tree and can be used to represent genetic heredity, for example (Figure 3.3(b)). In this case, some relevant concepts emerge: *parent-child* relationship between sequential nodes in the hierarchy; *leaves* for the final nodes; *root* for the orphan nodes; *siblings* for same-level nodes. Table 3.2 summarizes the most important graph classifications.

More formally, we can formulate a simple undirected graph as an ordered pair $\mathbf{z} = (\mathbf{v}, \mathbf{e})$ where $\mathbf{v}$ is the set of vertices and $\mathbf{e} \subseteq \{\{i, j\} \mid i, j \in \mathbf{v} \text{ and } i \neq j\}$ the set of edges, which are unordered pairs of vertices. Similar formulations could be elaborated for directed or weighted graphs.

A graph encodes a problem we are interested in. GT is a formal metalanguage to transcribe and operate discrete structured systems. It was a stepping stone in structure-based approaches and was

**Table 3.2:** Examples of graphs according to different criteria

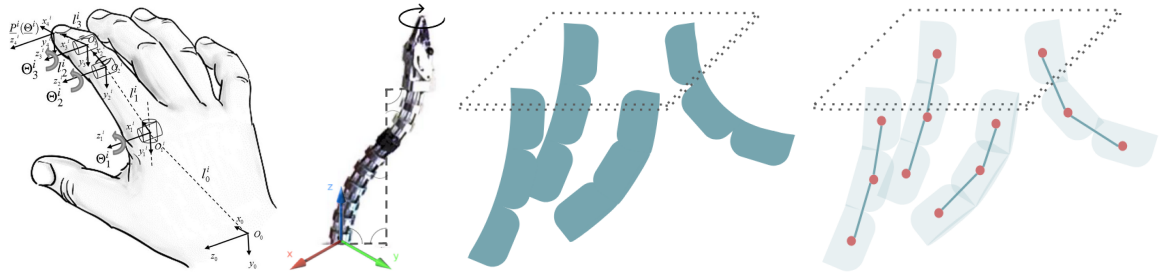
| Criteria | Graph Types | Examples |
|---|---|---|
| **edge directions** | **directed** | Instagram; Wikipedia |
| | **undirected** | Facebook; subway system |
| **edge density** | **dense** | European capital city airports flight connections |
| | **sparse** | Worldwide airports flight connections |
| **loops** | **cyclic** | Lisbon subway system; the internet |
| | **acyclic** | genealogical tree; corporate hierarchical structure |
| **edge weight** | **unweighted** | American highway system if time is not an issue |
| | **weighted** | American highway system if time is an issue |

accountable for advances in countless fields. In our domain, graphs can be used to discretize soft robotic systems and to express their non-rigid kinematics.

### 3.1.3 Robot kinematics

After an overall perspective of the importance of structure in characterizing a system and the robustness of graph theory in representing them, we will now turn to our domain - robotics - to understand how is connectivity usually expressed. Robot kinematics is a crucial concept for this thesis (and one we have mentioned a few times before) whose meaning and importance should be clarified.

Kinematics studies the geometric properties of the motion of a set of points [113]. In robotics, a kinematic model is a mathematical description of the robot structure, its functional dimensions and degrees of freedom. If a set of points has the property that the distances between any two of them is constant, it is called a rigid body. A chain is constructed by connecting rigid bodies together with joints that constrain their relative movement and it is the structural essence of most robotic arms, legs or hand fingers in manufacturing or outside of industry [113].



**(a)** Hand kinematics [114] **(b)** Soft robot kinematics **(c)** Soft gripper kinematic chain **(d)** Soft gripper kinematic graph

**Figure 3.4:** Kinematic chains

In one of its forms, one end of the kinematic chain is attached to a rigid base, and it is called an open chain. It is the case of our fingers, where each one represents a three-rigid-phalanges chain attached to the hand palm - Figure 3.4(a). It is also the case of most robotic hand-inspired grippers.

In the case of soft robots, many soft limbs also work in a similar way, although instead of rigid body chains, each segment is flexible and elastic - Figure 3.4(b). It is the case of the non-rigid robotic systems in Figures 2.3(a)-2.3(c) or the soft grippers in Figures 2.4(b) and 2.4(c). In this thesis, as stated before, we ought to learn the forward model of a non-rigid kinematic chain. Being a soft hand-inspired end-effector, for gripping and manipulation purposes, this kinematic chain will also include multiple fingers, each one with three connected soft "phalanges" and with the base of each finger attached to the same rigid plane, as in Figure 3.4(c).

Also here we can use graph representations to express our system: each non-rigid segment of the chain is a graph node and their sequence is represented by links (or vertices), as in Figure 3.4(d).

Since, throughout this thesis, some different (sometimes similar) concepts will be used when talking about a soft system kinematics, Table 3.3 helps clarifying the definition of each.

| **Nomenclature** | | |
|---|---|---|
| **Concept** | **Definition** | **Example** |
| **connectivity** | the (physical) link between different elements of a system | points A and B are connected, C is not |
| **structure** | the connectivity and spatial arrangement of different elements of a system | points A and B are connected, C is equidistant to A and B |
| **kinematics** | description of the motions in all possible configurations of a system, subject to constraints | points A and B are connected and still; C moves freely along the plane that is perpendicular to $\overline{AB}$ |
| **dynamics** | time varying phenomena that involves motion of points under the action of forces | points A and B are still; point C is pushed away from the others with decreasing speed |

**Table 3.3:** Definition of kinematics and other related nomenclature

### 3.1.4 Graph representations in artificial neural networks

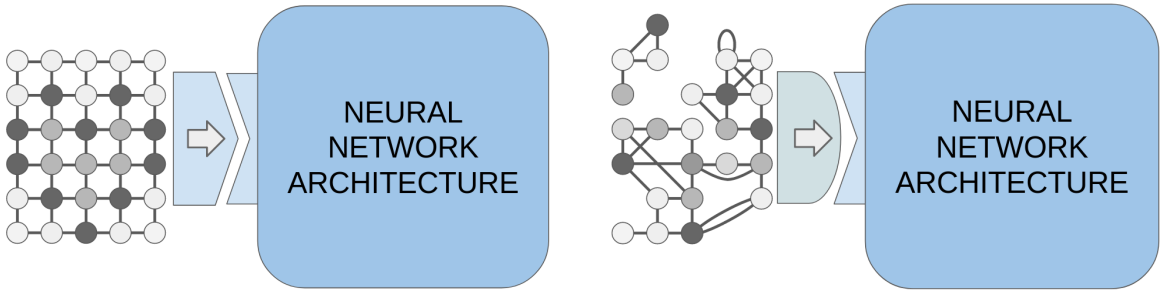
Artificial Neural Networks (ANNs) are computing systems from the second half of the twentieth century, vaguely inspired by the biological neural networks that constitute human brains [9, 115]. Using learning processes, they are trained to extract meaningful patterns from large amounts of data. Over the last decades, many architectural variations have been proposed and applied to tasks like classification, prediction or clustering in fields as computer vision, finance or robotics.

**Definition: Artificial neural network**

Biologically inspired computational network of simple processing units which learn to acquire knowledge from data and store this knowledge in its connections.

By the start of this millennium, Convolutional Neural Networks (CNNs) had been making significant advances in computer vision as simple yet powerful deep learning architectures but their scope, although wide, was limited. By that time, Recurrent Neural Networks (RNNs) were also achieving ground-breaking results in natural language processing with their ability to use internal memory to process variable-length sequences of input. However, in both cases, as in most relevant ANN architectures, the implicit data structures were relatively regular, with images being represented by pixels

in a grid and sentences following a sequential structure of limited characters. None of these methods would effectively operate on non-Euclidean data structures, like meshes or networks (see Figure 3.5).




**(a)** Euclidean structured data in the form of a matrix of image pixels fits desired input of ANN architectures

**(b)** Non-euclidean graph representation needs some adaptation to fit desired input of ANN architectures

**Figure 3.5:** Data structure as input for neural network architectures

As we have seen before, graphs are powerful representations for a myriad of relational systems. They can capture both simple Euclidean relations and more complex ones. So, by this time, it was relevant to adapt existing methods to combine graph representations with ANN architectures.

A first strategy from standard machine learning approaches was to map graphs into simpler representations, like vectors or sequences of real numbers in a preliminary *pre-processing phase*. However, this procedure was quite problem-dependent, time-consuming, order-variant and could involve (topological) information losses. More recently, several approaches [116, 117] have tried to preserve the graph-structured nature of the data for as long as required before the pre-processing phase [13]. Recursive neural networks [116, 118] and Markov chains [117, 119, 120] are main examples of this set of techniques and have been applied both to graph- and node-focused scenarios.

Recursive Neural Networks (RvNNs) operate on graphs by estimating the parameters of a mapping function (from a graph to a vector of reals) and have been applied to various fields [121]: logical term classification [122], chemical compound classification [123], logo recognition [124, 125], web page scoring [126], and face localization [127]. However, some limitations of RvNNs usually include being limited to acyclic graphs and the state of each node depending only on its children instead of its neighbours. Also, for node-focused applications [127] (as well as for certain cyclic graphs [128]) the input graph must undergo a specific pre-processing phase. RvNNs are also related to Support Vector Machines (SVMs) [129–131], which can adopt special kernels to operate on graph-structured data. They both automatically encode the input graph into an internal representation but this internal encoding is learned from examples in RvNNs, while in SVMs it is hand-designed by the user [132].

Graphs are also the representation *par excellence* for Markov Chain (MC) models [133]. The MC is a stochastic model that sequences a series of events in which an event's (or edge's) probability (weight) depends only on the state (node) attained from the previous event. In this sense, MC models can emulate processes where the causal connections among events are represented by graphs and be extensive to different scenarios in many fields. For instance, random walk theory (a particular class of MC models) has been successfully applied to the realization of web page ranking algorithms

in [134, 135]. Nonetheless, we must remember that the Markov assumption (or memorylessness property) is sometimes restrictive as not all processes can be modelled as markovian. More generally, several other statistical methods assume that the data set consists of patterns and relationships between patterns: random fields [136], Bayesian networks [137], statistical relational learning [138], transductive learning [139], and semi-supervised approaches for graph processing [140].

At this point, the need for a framework that combined neural networks with non-Euclidean order-invariant graph representations was not fully satisfied. More recently, a new architecture was proposed to tackle this exact issue: Graph Neural Networks.

## 3.2 Graph Neural Networks

Graph Neural Networks (GNNs) [13] were proposed in 2009 as an extension of "existing neural network methods for processing the data represented in graph domain" and, since its publication, many improvements derived from it and for countless applications [141].

**Definition: Graph Neural Network**
Recent class of neural networks that uses local message passing to operate directly on graph-structured data.

The key motivations for this framework were two:

1. **graph representation power**: analyzing graphs with machine learning has been receiving more and more attention as graphs allow non-Euclidean representations, graph embeddings and can be used to characterize a large number of systems across various areas.

2. **order invariance**: standard neural networks like CNNs and RNNs struggle handling (order-invariant) graph input properly as they stack the feature of nodes in a specific sequence.

In this section, we will concentrate on providing an overview of this framework. We will start by formulating and describing this architecture and how it works, followed by the applicability of GNNs in varied fields for the past decade and finally a summary of their variations and deployments.

### 3.2.1 Formulation and description

The GNN can be summarized as a sequence of node-to-edge and edge-to-node message passing. Each node is a representation of a variable with a set of features. It can represent an image, a word, an embedding, etc. Edges also include a set of features, for instance the type of relation between nodes. Similarly to the notation in [15], for a given graph $\mathbf{z} = (\mathbf{v}, \mathbf{e})$ with node features $x_i$, $i \in \mathbf{v}$, and edge features $e_{ij} \in \mathbf{e}$, one single node-to-node message passing can be defined as:

$$v \to e : h^t_{(i,j)} = f^t_e(h^t_i, h^t_j, e_{ij}) \tag{3.1}$$

$$e \to v : h^{t+1}_i = f^t_v(\sum_{j \in \mathcal{N}_i} h^t_{i,j}, h^t_i) \tag{3.2}$$

where the hidden states $h_{i,j}^t$ and $h_i^t$ of the edge $(i,j) \in \mathbf{e}$ and node i, respectively, are updated. $\mathcal{N}_i$ denotes the set of indices of the neighbouring nodes of i. The update functions $f_e^t$ and $f_v^t$ are node- and edge-specific neural networks, respectively.

On a first step, each edge will carry a message, which is computed taking into account both nodes and the type of edge. This will happen for all edges connected to node $i$. Then, all the neighbouring messages to $i$ are aggregated to update the state of this node $i^{t+1}$. To fully understand this two-step mechanism we must keep in mind that this is not happening just for $i$ but for all nodes at the same time. The two equations - (3.1) and (3.2) - repeat for as many rounds as how far we want the messages to propagate. In the beginning, each node knows only about itself; after one round, each node knows about its 1-neighbourhood; after $k$ rounds of message passing, each node includes knowledge about its k-neighbouring nodes. Figure 3.6 shows the message passing phase for two steps, where the node states and message passing are represented by different shades of grey.

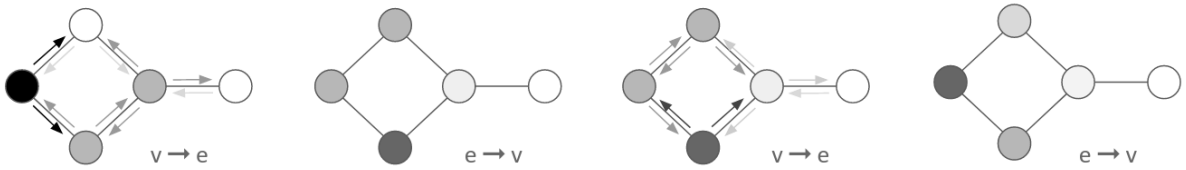



**Figure 3.6:** Message passing mechanism: two rounds of node-to-edge and edge-to-node updates

Although $\mathcal{G}$ is assumed to be an undirected graph, it is trivial to extend the formalism to directed graphs by explicitly assigning two directed edges in opposite directions for each undirected edge. Even in directed graphs, creating (fake) backwards relations is also possible and sometimes needed for *centrifugal* nodes that need to learn how they belong in the graph.

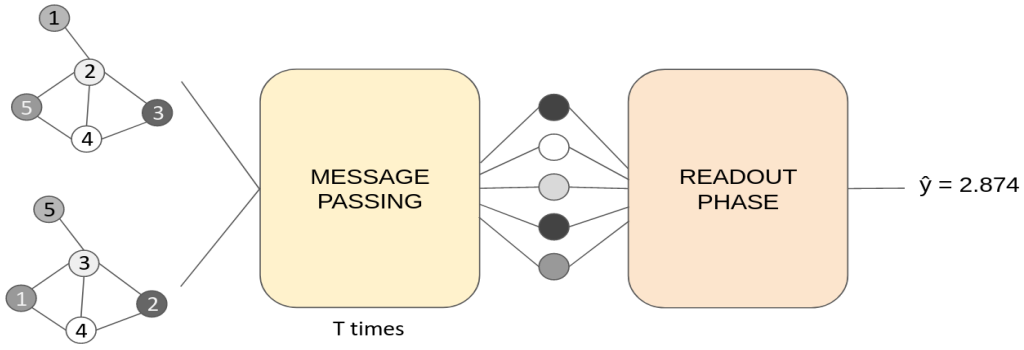



**Figure 3.7:** Graph neural networks structure

At this point, we reached a new set of nodes whose updated features include knowledge about the graph, with direct applications in node selection or classification, for example. From this point, GNNs can also be used for graph classification tasks, by including a readout phase, where a feature vector for the whole graph is computed using a (learned differentiable) readout function R, according to:

$$\hat{y} = R(\{h_i^T | i \in \mathbf{v}\}) \tag{3.3}$$

where R operates on the set of node states and must be invariant to permutations of this set in order for the GNN to be invariant to graph isomorphism. This means that the order of the nodes in the input

graph is redundant for the calculation of the final node and graph feature vectors. Figure 3.7 shows a representation of the structure of the GNN mechanism described before.

Dozens of different variations to the GNN structure have been proposed [141]. Some of the most prominent baselines combine CNNs, LSTMs or attention mechanisms. In the next section, we describe some of its applications to different domains.

### 3.2.2 GNNs are everywhere

After describing the general framework that is common to all GNN baselines, we now include a compact literature review for some of the most relevant applications of GNN architectures in different fields and tasks. This should represent a general yet compact state-of-the-art review; for further understanding of any of the undermentioned applications, please refer to the cited documentation.

The GNN architecture came to unify graph representations with neural networks with promising possibilities in many applications. For instance, variants of GNNs have been shown to outdistance non-structured approaches in relational reasoning tasks [142, 143] or modelling interacting or multi-agent systems [14, 144, 145]. Furthermore, GNNs have proven to be highly effective at graph [146–148] or node [149, 150] classification for large graphs. Node importance estimation has been studied by Amazon in [151], applied to knowledge graphs, so important in item recommendation and resource allocation. GNNs were also the subject of more theoretical studies in [152–154].

Due to the ubiquitous nature of structure and to the simplicity and generality of graph representations, GNN-based models have been applied in varied fields where relational links between elements are meaningful. From chemistry [155, 156] to cosmology [157, 158], from crowd analysis [159, 160] to combinatorial optimization [161], combining neural networks with graph-based approaches has displayed more-than-decent performance increase over non-structured based techniques. Other scenarios include: pharmacological studies [162] of polypharmacy side effects (drug interactions); financial stock market [163] where knowledge graph datasets and relational representations of the market are useful in making stock predictions across different markets and longer time horizons; social recommendations [164] where GNNs improve user and item representations; and many others.

In computer vision, Lee et al. [165] also used structured knowledge graphs, this time for visual reasoning, showing state-of-the-art results in multi-label classification and ML-ZSL tasks. A similar work by Marino et al. [166] takes inspiration from the way humans learn about the relational visual world and uses knowledge graphs for image multi-label classification. GNNs were also used to learn interactions between humans and objects from images and videos in [167].

Furthermore, GNNs have been widely used to model physical systems [14, 168] - such as springs, rope or n-body-systems - allegedly inspired in how the human brain reasons about these entities and the outer physical world, in general. Some approaches have tackled the similar issue of inferring structure and dynamics but using raw visual data [169, 170]. Also, some recent work [87, 93] on particle-based physics simulators use GNN-based approaches to achieve state-of-the-art realism with fluids, deformable objects and granular materials.

Little work on GNNs has been targetted at robotics end-effectors sensorimotor skills. The Tact-

ileGCN [171] used tactile sensors and a known graph structure to predict grasping stability while TactileSGNet [172] combined GNNs with spiking neural networks for event-based tactile object recognition. There has been even fewer GNN-related research in the case of soft robotics.

For a deeper understanding of GNN strategies in different fields, recent literature surveys about their methods and applications can be found in [141] and [173].

### 3.2.3 GNN variations

We have described the structure and functioning of GNNs and mentioned some of the different architectural adaptations. In the previous segment, we talked about the ubiquity of GNNs and how they are used in many scientific fields and machine learning tasks. We shall now display a compact review of distinct GNN-based models, algorithms and applications.

**Table 3.4:** GNN-based models in different fields

| Field | Application | Algorithm | Deep Learning Model | References |
|---|---|---|---|---|
| Text | Text classification | GCN | Graph Convolutional Network | [147, 149, 150] |
| | Relational reasoning | RRN | Recurrent Neural Netowrk | [143] |
| | | IN | Graph Neural Network | [14] |
| Image | Image classification | GGNN | Gated Graph Neural Network | [165] |
| | | GSNN | Gated Graph Neural Network | [166] |
| | Interaction detection | GPNN | Graph Neural Network | [167] |
| Science | Molecular Fingerprints | NGF | Graph Convolutional Network | [155] |
| | | GCN | Graph Convolutional Network | [156] |
| | Physics Systems | IN | Graph Neural Network | [14] |
| | | VIN | Graph Neural Network | [169] |
| | Cosmological patterns | GCN | Graph Convolutional Network | [158] |
| | Polypharmacy effects | Decagon | Graph Convolutional Network | [162] |
| Finance | Stock Prediction | GCN | Graph Convolutional Network | [163] |
| Crowd Prediction | Group interaction | AG-GCN | Graph Convolutional Network | [159] |
| Knowledge Graph | Node importance estimation | GENI | Graph Attention Networks | [151] |
| Combinatorial Optimization | | GCN | Graph Convolutional Network | [161] |

We have seen that the GNN is a recent class of neural networks that uses node-to-edge and edge-to-node message passing to operate directly on non-Euclidean order-invariant graphs. Different variations in architecture were proposed since then, some of the most influential being:

- Gated GNNs [174]: use Gated Recurrent Units [175], unrolling the recurrence for a fixed number of steps and using backpropagation through time in order to compute gradients.
- Graph Convolutional Networks [149]: result from the application of convolutional neural networks on graphs. Stack multiple graph convolutional layers to extract high-level representations.

- Graph Attention Networks [176]: Adopt attention mechanisms to learn the relative weights between two connected nodes.

Moreover, there were also some relevant algorithms that inspired further work such as Interaction Networks [14] or Relation Networks [142]. In this work, our focus will not be comparing different GNN strategies but to compare structured and non-structured approaches; *ergo*, we will not concentrate on further expanding the many variations.

Table 3.4 depicts a summary of different GNN applications, the algorithms used and the respective deep learning model, similarly to what has been done in [141].

## 3.3 Conclusion

In this chapter, we were introduced to structure representations: from the importance of structure in relational systems and the prominent Theory of Graphs; to the challenges of combining neural networks with graph representations and the recent solution of GNNs.

We started by looking at the ubiquity of structure in everyday systems and how this connectivity can be capitalized to better learn complex systems. After that, we introduced the graph as a simple yet powerful form of expressing a system as a network of points and links. We concentrated on our domain of (soft) robotics to describe how kinematic chains are structural units of robotic end-effectors and how they can be expressed by graphs. We moved on to showing how neural networks are often unprepared to deal with graph representations and the urgency for such a combination. Finally, we introduced GNNs as the ultimate neural network framework to operate on such robust structures and we described its operation and applications.

A schematics representation of the concepts in Chapters 2 and 3 is depicted in Figure 3.8. It can be seen as a rearrangement of the chart in Figure 1.1.

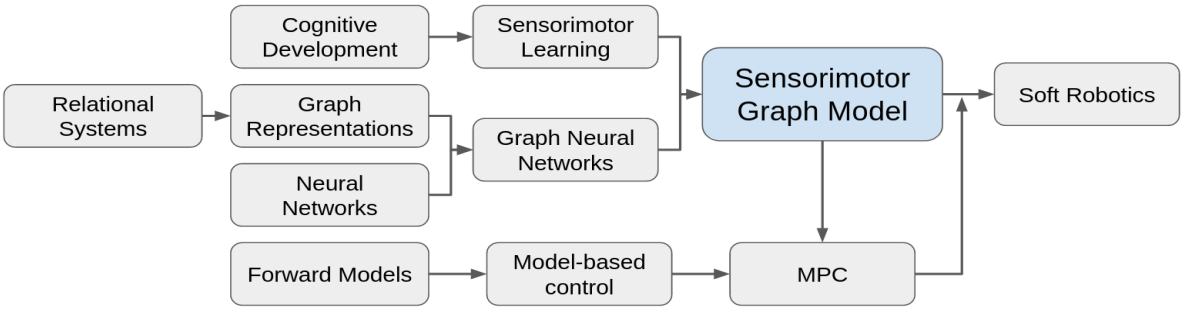



**Figure 3.8:** Chapter 3: A summary of the concepts so far and their relation

These new concepts clarified the need to combine the modelling power of artificial neural networks with graph-based representations and allow us to formulate our problem statement as:

**Problem statement**

In this work, we propose to learn a structured differentiable action-conditioned dynamics model of the non-rigid robotic (soft) hand kinematic chain inspired by how newborns acquire controlled motor capabilities and using Graph-based Neural Networks.

# 4

# Sensorimotor Graph

## Contents

Over the past two chapters, a thorough literature review has shedded light on some challenges in modern soft robotics as well as on different strategies that have been used to model relational systems. In concrete, we can summarize the main conclusions drawn in Chapters 2 and 3 as follows:

1. Accurate modelling and control of complex systems is essential for planning regulated agile interactions with the environment, particularly in the field of robotics.
2. Soft robotics is a recent flourishing branch of robotics that uses flexible organic materials to increase adaptability and efficiency. However, these soft materials promote more degrees of freedom in the non-rigid kinematic chain that are much harder to model and control.
3. In the first months of life, humans rely on sensorimotor mechanisms to develop brain models of their body parts and the physical world that allows them controlled action of their limbs. A similar self-calibration strategy could be used in soft robotics.
4. In order to create these relational models, we shall use the modelling power of artificial neural networks - specifically deep learning.
5. A robust representation for the non-rigid robotic hand is using graphs as they can efficiently express non-Euclidean networks
6. Recently proposed Graph Neural Networks combine the modelling power of neural networks with graph-based representations. As robust order-invariant networks, they have been used to learn the dynamical model of many relational systems and could be ideal to model our robotic soft kinematic chain.

After portraying the challenge addressed in this thesis - how it is motivated, how it is composed, what strategies have been tried, their limitations and what elements are necessary to tackle this problem, - this chapter will be introducing our proposed strategy, the Sensorimotor Graph model.

## 4.1 Sensorimotor Graph Model

The Sensorimotor Graph (SMG) takes its name from the sensorimotor stage of cognitive learning, where infants develop an understanding of the world through trial and error and using their senses and motor actions. This model is a graph-based approach that picks up on recent work on Graph Neural Networks and applies it to soft robotic limbs.

In this section, we describe this proposed model. We will start by revisiting the context to this approach and how it is inspired by biological mechanisms. Then, a brief motivation will be provided. Finally, we detail how the model is formulated.

### 4.1.1 Context and Inspiration

At a very early age, newborns learn by themselves how to control and work their limbs. Without any knowledge of the physical equations that rule the world, infants learn the required force to push

against rigid or soft surfaces, to grab objects of different shapes and weights, and later to crawl, to stand, to walk, to run. As grown adults, even without an exact understanding of Newton laws of motion or gravitational constants, we manage to adapt to unseen circumstances, from manipulating Oobleck [177] to walking on low gravitation scenarios [178]. This is accomplished by a subconscious motor self-calibration faculty we acquire during the sensorimotor stage of cognitive development [4]. It includes motor babbling - repeatedly performing simple motor commands for a short duration, - during a period in life of rapid cognitive growth and where we develop an understanding of the world through sensorimotor trial and error.

Inspired by sensorimotor development in newborns, just like expanded in Section 2.1, it is the same sort of self-calibration process that this work soughts to mimic. Applied to a robotic soft hand's non-rigid kinematic chain, we propose to learn a structured differentiable action-conditioned dynamics model, based on an explicit joint connectivity graph, the SensoriMotor Graph-based model, or SMG model.

This bio-inspired calibration process is described and applied for hand kinematics but it is general and transfers to arms, legs, face and all other sensorimuscular parts of the body. Sensorimotor babbling is not only essential for understanding the body dynamics, but also a required step for the *controlled* use of the muscles.

### 4.1.2 Motivation

The importance of modelling and control in robotics, combined with the added complexity of soft systems leads to the necessity of an effortless self-calibration mechanism. A way of learning both dynamics and connectivity without the need to write arduous equations, rewards or labels. Moreover, a framework that learns the relational behaviour of the system in an unsupervised way.

On the one hand, the non-rigid nature of soft materials originates complex dynamics that are hard to capture by forward models. On the other hand, adaptive control is capable of learning the model parameters but is alien to the connectivity of the kinematic chain. In this section, we introduce the SMG, which addresses the aforementioned shortcomings in two steps: first, it infers the underlying system connectivity by observing sequences of joint positions and actuation signals and then, to model the system dynamics, conditions its prediction model on the explicit connectivity graph structure.

Previously, we mentioned how sensorimotor learning is crucial for the understanding of body dynamics and for a *controlled* use of the muscles. Similarly, with soft robotics getting more and more attention, may it be with cables or air compression acting as muscles, the dexterous efficient use of limbs that consequences from the SMG model may be paramount for tasks in locomotion and manipulation, among others.

### 4.1.3 Model Formulation

A simplified design of the SMG model is depicted in Figure 4.1. On a first stage, the past states $\boldsymbol{x}^{\boldsymbol{T}_\Gamma}$, $T_\Gamma$ being a temporal context, are fed to the connectivity inference module $\Gamma(\boldsymbol{x}^{\boldsymbol{T}_\Gamma})$ that observes

that sequence and infers the explicit connectivity graph $\mathbf{z}$. We formalize our input as trajectories of N objects. We denote by $x_i^t$ the feature vector (with position and finger constraint) of object $i$, at time t. With this notation, $x^t = x_1^t, ..., x_N^t$ is the set of features of all N objects at time t, and $x_i = (x_i^1, ..., x_i^T)$ the trajectory of object $i$, for the total context of $T_\Gamma$ time steps. These input states include the past trajectories $\boldsymbol{p}_i^t$ for each element (or node) $n \in N$, as well as the past actuation signals $\boldsymbol{a}_i^t$ that are used to actuate on those nodes, such that $\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{a})$ being $(:,:)$ a concatenation. Furthermore, we assume that the dynamics can be modelled by a GNN given an unknown graph $\mathbf{z}$ where $z_{ij}$ represents the discrete edge between objects i and j, that can be directed or not.

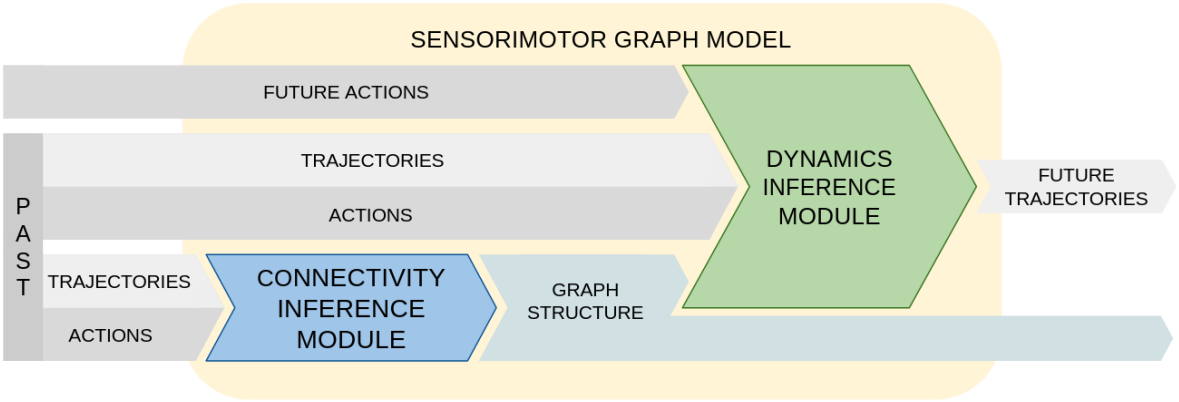



**Figure 4.1:** Simplified representation of the SMG model sequential components

In the example of motor babbling, if the infant flexes the index finger tendon and notices that this finger contracts, the first module $\Gamma(\boldsymbol{x})$ will correspond to learning that this particular tendon is connected to the index finger and that all points along the finger are connected and move together in a 3-joint fixed-base structure.

Then, the past states $\boldsymbol{x}^{T_\Delta}$ and the connectivity graph $\boldsymbol{z}$ will be used by the dynamics inference module $\Delta(\boldsymbol{x}^{T_\Delta}, \boldsymbol{z}, \boldsymbol{a}^{T_w})$, together with the future actuation signals $\boldsymbol{a}^{T_w}$, to predict the future trajectories. Here, $T_w$ is the prediction window ($T_w = \{0, ..., t_w\}$) and $T_\Delta$ is the new temporal context ($T_\Delta = \{-t_c', ..., 0\}$), where we only use the current state (in our case $T_\Delta = \{t_0\}$). The input actuation signals and the output predicted trajectories are for $T_w$ future time steps. For our infant motor babbler parallel, this second part corresponds to, now understanding the hand structure, the toddler moving some finger tendons and learning to predict how each finger is going to react.

For the purpose of learning the structured differentiable action-conditioned dynamics model of a soft hand gripper, we need an algorithm that uses order-invariant GNNs to infer explicit connectivity in an unsupervised manner. For this task, we chose the Neural Relational Inference Model [91], a recent unsupervised approach, in the form of a variational autoencoder. The next section describes this model in detail and its relation to the SMG.

## 4.2 Neural Relational Inference Model

In the previous section, we have seen our proposed framework for modelling a structured robotic non-rigid kinematic chain, the Sensorimotor Graph. This architecture allows for the self-calibration of soft robotic end-effectors through a process similar to motor babbling, inspired by the learning process. This graph-based architecture makes use of the structure of the soft parts for a better understanding of their dynamics, shaping more efficient and robust learning when compared to non-structure-based approaches. Since graph-based neural networks have led to significant improvements in modelling such different interacting systems using varied architectures, we now seek to find the most adequate model to adapt to our context of soft robotics end-effectors.

Inspired by previous work on GNNs, Kipf *et al* have proposed a new model for inferring an explicit interaction structure while simultaneously learning the dynamics of the interacting system, the Neural Relational Inference model, or NRI [149]. The NRI takes the form of a variational autoencoder to, in an unsupervised way, infer an *explicit* interaction structure while simultaneously learning the system dynamical model. This architecture further allows us to incorporate prior beliefs about the graph structure, such as different types of edges, in a principled manner.

We can highlight three different reasons for which the NRI was chosen to test our SMG:

1. Its **explicit connectivity**: the NRI model infers an explicit structure, allowing us to use a GNN-based approach.

2. Its **unsupervised training**: the NRI model learns connectivity and dynamics jointly, in an unsupervised manner.

3. Its **results**: the NRI model could accurately recover ground-truth connectivity and predict complex dynamics in different environments with different types of interactions.

The proposed motivational example formulation mentions a basketball match (Figure 4.2). If we were to watch the movement of the ten isolated points, corresponding to the ten players on the field (five of each team), we would notice that the movements were not random nor independent. Maybe after a while, we would recognize some implicit constraints, like certain frequent disposition corresponding to the offense tactics or some pairwise links corresponding to defender-defended relations. These constraints would help us infer underlying relations between the ten elements and this relational bias would help us understand their dynamics. After watching a few minutes of this ten-player "dance", we would suddenly close our eyes and ask ourselves how that offense would go from then on. And, at that time, maybe the movement we would picture in our head would not be so different from the one happening in front of our shut eyes.

The challenge of inferring the inherent structure of an observed system, is a universal problem statement, applicable to a basketball court, the Solar system or a silicon soft gripper. NRI solves this problem by learning the dynamics (future states) and explicit connectivity (graph structure) at the same time and without the need for any labels. This is done by training two parts jointly: an encoder that predicts the interactions given the trajectories (corresponding to the *connectivity inference module* in

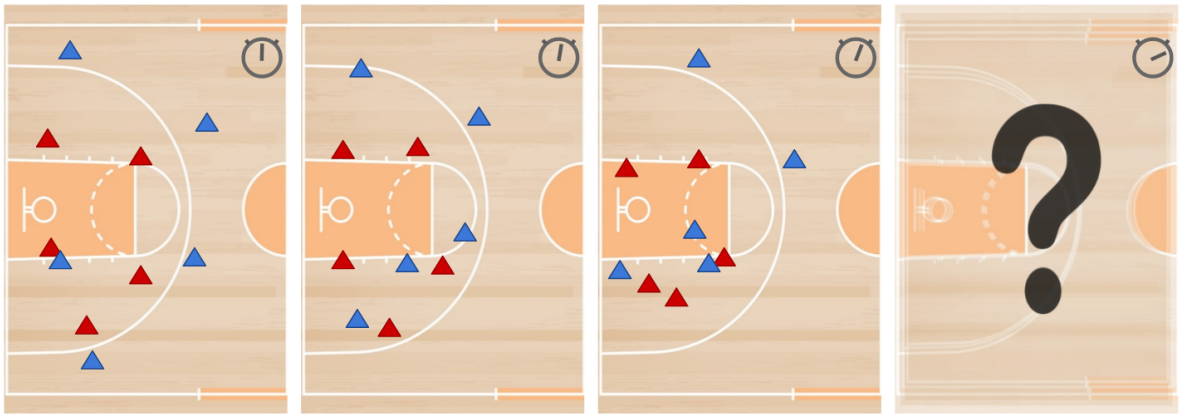


**Figure 4.2:** Relational system: basketball players don't move independently

Figure 4.1) and a decoder that learns the dynamical model given the interaction graph (the *dynamics inference module* in Figure 4.1).

Similarly to what is described in Section 4.1, Kipf et al. formalize the feature vector (input) of object $i \in N$, at time $t \in T$ and the edge vector as $z_{ij}$ between nodes $i$ and $j$. The NRI model is then formalized as a Variational Autoencoder (VAE) ([179, 180]) that maximizes the Evidence Lower Bound (ELBO):

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[log(p_\theta(\mathbf{x}|\mathbf{z}))] - KL[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(z)] \tag{4.1}$$

In the above equation, we can see two distinct terms, the first one representing an expected value ($\mathbb{E}_{\alpha(x)}(X) = \int_x x \cdot \alpha(x) \cdot dx$) and the latter the Kullback-Leibler divergence ( $KL(A \parallel B)$ between probability distribution A and B, simply put, measures how different the distributions are). We can rewrite Eq. (4.1) as

$$\mathcal{L} = \int_z log(p_\theta(\mathbf{x}|\mathbf{z})) \cdot q_\phi(\mathbf{x}|\mathbf{z}) \cdot dz - KL[q_\phi(\mathbf{z}|\mathbf{x}) \parallel N(z)] \tag{4.2}$$

The expected value can be represented by an integral, as in Eq. (4.2), and expresses an average of each sequence of states for the respective graph, weighted by the probability of that graph structure. In other words, it is trying to maximize the probability of a trajectory given a certain structure (weighted by the probability of that structure, given the dynamics). This will correspond to the dynamics inference module, or the decoder. The $KL$ divergence can be seen as being minimized to approximate the encoder to a normal distribution.

### 4.2.1 Encoder

The encoder is the first of two components of the NRI model and its goal is to infer pairwise interactions $\mathbf{z}_{ij}$ between object $i$ and $j$, given observed trajectories $(\mathbf{x}^1, ..., \mathbf{x}^T)$. Since we do not know the underlying graph, we can use an order invariant GNN on the fully-connected graph to predict the latent graph structure. This corresponds to the *connectivity inference module* in Figure 4.1.

More formally, the encoder is modeled as $q_\phi(\mathbf{z}_{i,j}|\mathbf{x}) = softmax(f_{enc,\phi}(\mathbf{x})_{ij,1:K})$, where $f_{enc,\phi}(\mathbf{x})$ is a GNN acting on the fully-connected graph (with no self-loops). The encoder computes iterative

node-to-edge and edge-to-node message passing operations to simulate node interactions and then calculates the edge type. Given input trajectories $\mathbf{x}_1, ..., \mathbf{x}_K$, the sequential node and edge updates follow:

$$\mathbf{h}_j^1 = f_{emb}(\mathbf{x}_j) \tag{4.3}$$

$$v \to e : \mathbf{h}_{(i,j)}^1 = f_e^1([\mathbf{h}_i^1, \mathbf{h}_j^1]) \tag{4.4}$$

$$\vdots$$

$$e \to v : \mathbf{h}_j^n = f_v^{n-1}(\sum_{i \neq j} \mathbf{h}_{(i,j)}^{n-1}) \tag{4.5}$$

$$v \to e : \mathbf{h}_{(i,j)}^n = f_e^n([\mathbf{h}_i^n, \mathbf{h}_j^n]) \tag{4.6}$$

The use of multi-step effect message passing enables the propagation of signals beyond pairwise interactions and the entanglement of multiple interactions. More specifically, in the first pass - Equations (4.3) and (4.4) - the node-to-edge messages are computed using an edge-specific embedding function $f_e$. Embedded node values from nodes $i$ and $j$ are then used to update the edge $e_{ij}$ between them. After updating the edge state, another embedding function, $f_v$, takes the edge values of all edges connected to node $i$, uses each one to compute an edge-to-node message and aggregates them all - Equation (4.5). Then, a final node-to-edge message passing operation infers the final edge value - Equation (4.6) from which the edge existence probability is calculated. These last two steps use information from the whole graph and are repeated for $n$ passes ($n \in \mathbb{N}$).

Given the distribution, it is possible to either train the model in a supervised setting, by providing ground truth labels for the edges, or training the model end-to-end by sampling edges from the inferred distribution and using these as graph structure for the subsequent trajectory prediction step. The functions $f_v$ and $f_e$ are neural networks that map between the node and edge representations. For this purpose, the NRI model uses two distinct possible implementations: fully-connected MLPs or one-dimensional CNNs with attentive pooling, similar to [181]. We generally adopt the first implementation unless otherwise mentioned.

After sampling from $q_\phi(\mathbf{z}_{i,j}|\mathbf{x})$, since the latent variables are discrete and therefore the reparametrization trick cannot be used to backpropagate through the sampling, the NRI model bypasses this difficulty by sampling from a continuous approximation of the discrete distribution and then using the reparametrization trick to get the (biased) gradients. The samples were drawn from the concrete distribution [182]:

$$\mathbf{z}_{ij} = softmax((\mathbf{h}_{(i,j)}^2 + g)/\tau) \tag{4.7}$$

where $g \in \mathbb{R}^K$ is a vector of i.i.d. samples drawn from a $Gumbel(0, 1)$ distribution and $\tau$ is the softmax temperature parameter that controls the 'smoothness' of the samples (distribution converges to one-hot samples as $\tau \to 0$).

### 4.2.2 Decoder

After the encoder producing a connectivity graph $\mathbf{z}$, the goal of the decoder $p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}^{t+1}|x^t, ..., x^1, z)$ is to predict the future sequence of the interacting system's dynamics using a GNN. The prior $p_\theta(\mathbf{z}) = \prod_{i \neq j} p_\theta(\mathbf{z}_{ij})$ is a factorized uniform distribution over edges types. This corresponds to the *dynamics inference module* in Figure 4.1.

For physics simulations, the dynamics follow the memorylessness Markov property [183], meaning that to predict future states only the present state is needed, or $p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z}) = p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, \mathbf{z})$. With this in mind, Kipf et al [91] use a GNN similar to interaction networks [14], although employing a separate neural network for each edge type. Just like for the encoder, the trajectory predictions are based on a sequence of node and edge message passing which formally transcribes as:

$$v \rightarrow e : \tilde{\mathbf{h}}_{(i,j)}^t = \sum z_{ij} \tilde{f}_e([\mathbf{x}_i^t, \mathbf{x}_j^t]) \tag{4.8}$$

$$e \rightarrow v : \boldsymbol{\mu}_j^{t+1} = x_j^t + \tilde{f}_v\Big(\sum_{i \neq j} \tilde{\mathbf{h}}_{(i,j)}^t\Big) \tag{4.9}$$

$$p(\mathbf{x}_j^{t+1}|\mathbf{x}^t, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_j^{t+1}, \sigma^2 \mathbf{I}) \tag{4.10}$$

where $\sigma^2$ is a fixed variance. An initial accumulation function updates the edge value by weighting adjacent nodes by the edge existence probability - Equation (4.8). Subsequently, in Equation (4.9), edges are aggregated in the nodes through an order invariant summation and transformed by an embedding function, $\tilde{f}_v$. Since the current node state $x_j^t$ is added to the weighted sum, the model only learns the change in state, $\Delta x_j^t$. The next state $x_j^{t+1}$ is sampled from the normal distribution - Equation (4.10) - with mean value of the new position, $\boldsymbol{\mu}_j^{t+1}$. The neural networks used for the embedding functions are, by default, MLPs.

Looking at the ELBO formulation in Equation (4.1), we notice that the reconstruction loss term has the form $\sum_{t=1}^{T} log[p(\mathbf{x}^t|\mathbf{x}^{t-1}, z)]$ which involves only single step predictions. When optimizing this objective, one issue that might arise is that the connectivity and interactions ought to have a small effect on short-term dynamics. In other words, the dynamics of a physical system moving for a short period of time can be fairly modelled by a simple approximation, for example a linear model, and will lead to a sub-optimal decoder that ignores the latent edges completely and achieves only a marginally worse reconstruction loss. On the original NRI work, this issue is addressed in two ways:

- The prediction window should be large enough to promote non-linearities in the motion and highlight the insufficiency of the "degenerate" model.
- The inclusion of different types of edges should make the dependency on the edge type more explicit and harder to be ignored by the model.

The first strategy - multiple step prediction - is implemented by iteratively feeding the predicted mean $\mu^t$ for P steps (P=10 in the original experiments) to new state calculation in Equation 4.9. Since the errors accumulate for the P steps, the degenerate decoder will be highly suboptimal. The solution to the second obstacle consists of having separate MLPs for each edge type, with Equation 4.8 being

the sum of the $d$ (types) products instead, $\sum_d z_{ij,d} \tilde{f}_e^d([\mathbf{x}_i^t, \mathbf{x}_j^t])$, where $z_{ij,d}$ denotes the d-th edge type of the vector $\mathbf{z}_{ij}$.

Since, in many applications, the Markovian assumption does not hold, a recurrent decoder that can model $p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z})$ is used. In particular, this recurrent decoder (RNN), adds a GRU [175] unit to the GNN message passing operation in 4.9. In this case, since an internal state initialization is required and multi-step prediction must be handled with care, instead of alternating between sequences of P predicted steps and ground truth values, a *burn-in* phase is applied where the correct input is provided for the first $M_{burnin}$ time steps and the predicted mean is used only for the last $T - M_{burnin}$ steps (Figure 4.3).

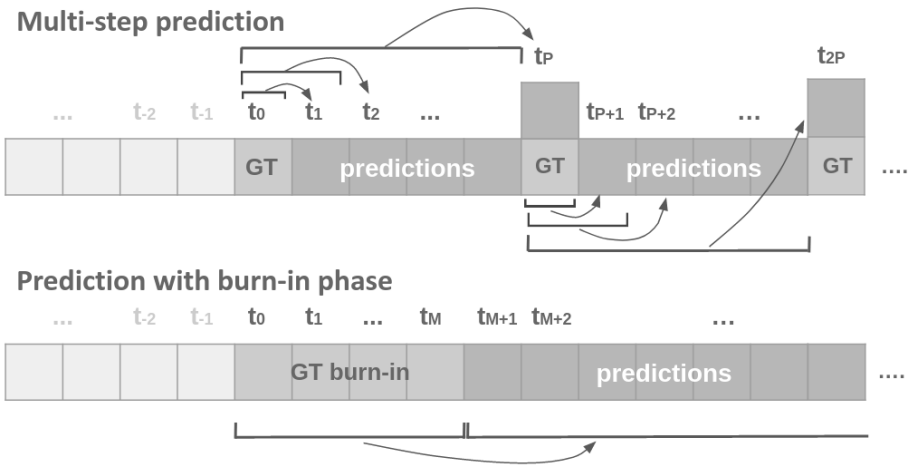



**Figure 4.3:** Difference between multi-step prediction of MLP and burn-in intialization of RNN

### 4.2.3 Extensions to the NRI model

After describing the general structure of the NRI model and its variants, we now explain the changes introduced in the original model. The two main architectural differences to our framework and the main adaptations regarded the inclusion of action inputs in the system state and the fluctuation in the number of nodes, as described below:

- Whereas the original NRI system state included a two-dimensional position and a two-dimensional velocity, our application required three-dimensional positions and the input (finger constraint) actions. Since each point state is described by its unique position in space $(p_x, p_y, p_z)$ and also by all the actions being fed to the soft hand $(a_1, ..., a_F)$, $F$ being the number of fingers, each state is not 4- but (3+F)-dimensional. The $F$ values (the same for all points in the same hand configuration at the same time) are not predicted like the trajectories, but their real values are included (concatenated) in the prediction. This simulates the real-life application, where finger movement orders are being given to the robotic hand in real-time and it is the final configuration (positions only) we want to control. However, each finger actuation is not explicitly associated with the respective finger; instead, the (3+F)-dimensional input is fed to the model which should map these links between action and effect.

- Unlike the NRI experiments, using constant-number-of-nodes systems, we want to test the model's robustness to configurational changes in hand kinematics and to tracking errors or node failures. This corresponds to having different number of fingers or different number of points per finger, which involves having a varying number of discretized hand points at the end of the day. In this work, we tackle this issue by zero-padding (filling with zeros) the states that are not necessary for each configuration. We have this simplification in mind when further analysing the results for the different modelling strategies.

Besides these architectural changes, further additions were made to the code:

- **metrics**: different metrics were relevant to this new model and were included in the code. For example, the calculation of the travelled distance, displacement, MSE normalized to travelled distance and displacement and the number of parameters of the neural network.
- **training**: regarding training procedures, we implemented early stopping and included burn-in phases or teacher forcing when needed, to balance opposing modelling strategies.
- **hyperparameters**: many hyperparameters were fine-tuned for the new conditions, such as the number of epochs, batch size, learning rate, learning rate decay, number of hidden layers, prediction step and patience. Some of the most relevant hyperparameters are further described in Section 4.3.

## 4.3 Hyperparameters

As we have seen before in Section 3.1.4, machine learning is an application of artificial intelligence where a model is trained to fit some observed data and improve from experience in order to make predictions or generalizations without being explicitly programmed to do so. The Sensorimotor Graph uses machine learning and artificial neural networks to infer a system's relational dynamics by "observation", similar to how we develop our brain models as curious infants.

In machine learning algorithms, there are usually two types of configurations, the model parameters and the model hyperparameters. The first are the configuration variables that are internal to the model and whose values it tries to learn from historical training data. They are key to a machine learning algorithm as they are its subject of estimation. In this section, we will focus on the latter.

A model hyperparameters are external configurations, whose values cannot be estimated or changed from data. They help estimate model parameters and are usually specified (and fine-tuned) by the practitioner. They might relate to the dimension of the network or the duration of the training, for example, and, in most cases, they aim to produce a good data fit.

### 4.3.1 Fitting the data

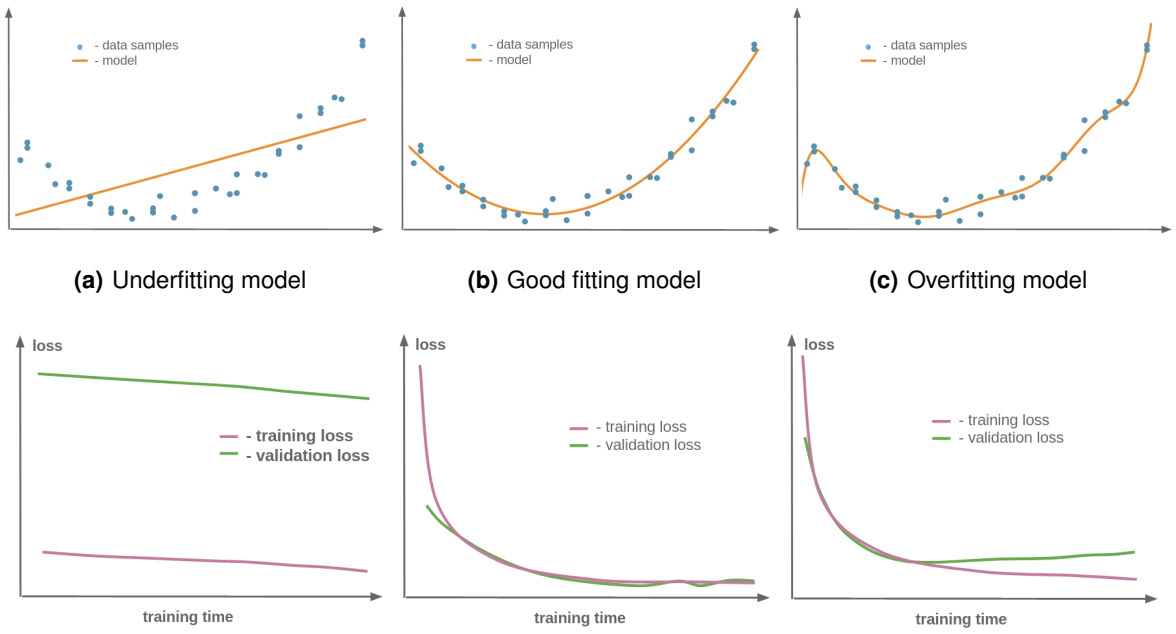
When trying to model data via machine learning algorithms, we usually use two distinct data sets: the training data set, for our model to use as examples when fine-tuning its internal parameters;

and the validation set, with new examples to see if the resulting model is actually satisfactory and generalizes well to unseen samples. We say that the model fits the data well if it fits both the training data and the validation data.

In order to understand if a model is "fitting the data well" - meaning it approximates a target function that rules the data distribution - a performance metric is necessary. The function used to evaluate a candidate solution (that is, a set of model internal weights) is referred to as the objective function. As we seek to minimize the error of our solution, the objective function is usually called the cost function or the loss function. The loss function for the NRI model is depicted in Equation 4.1.

We say that the model fits the data well if both the validation loss and the training loss are small and comparable, as this means the error of our approximate function is low for the data it learned from and for new data. However, this is not always the case. As such, it becomes relevant to track the loss function in both data sets, as the relation between the two might indicate under or overfit.

- **Underfit**: refers to a model that is not suitable or capable of fitting the data. This will be evident in a poor fit of both the training and the validation data.
- **Overfit**: refers to a model that fits the training data too well. Overfitting happens when the level of detail that the model learns from the training data starts negatively impacting its fit on new data. This will result in high training loss and low validation loss.




**(a)** Underfitting model **(b)** Good fitting model **(c)** Overfitting model

**(d)** Underfitting loss curves **(e)** Good fitting loss curves **(f)** Overfitting loss curves

**Figure 4.4:** Data fit

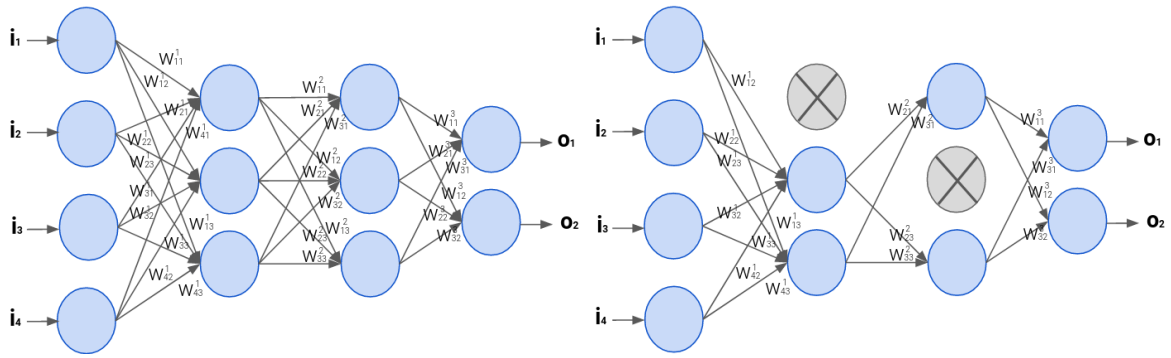
These two concepts become particularly meaningful when visually depicted. In Figures 4.4(a) to 4.4(c) we can see the harmful effect of under and over modelling: the first one will be too simplistic to capture the model's essence while the latter is too sensitive to noise or unimportant details. The

consequence of these limitations on the loss curves is also noticeable - Figures 4.4(d) to 4.4(f). Underfitted loss plots might take different aspects, whether with training and validation curves being stabilized at high values or still decreasing (meaning that the model hasn't stopped fine-tuning yet). A good model fit means the too curves stabilize at low values, close together or with a certain margin difference, the generalization gap. Overfit modelling will usually be similar to the ideal case until a point where the parameters keep fitting the training data (training loss decreases) harming their performance on the validation set.

As we will see next, the ability of a model to be more or less expressive - meaning to represent simpler or more complex patterns in data - is deeply linked with the size of the artificial neural network.

### 4.3.2 Network

When trying to fit some data, a model will fine-tune the many weights that constitute the networks hidden layers (Figure 4.5(a)). Given a certain input, it is the variation of those weights, through backpropagation, that will converge to the desired output.



**(a)** Fully connected artificial neural network **(b)** Artificial neural network with dropout layers

**Figure 4.5:** Dropout regularization

If a model expressiveness power can be reduced to the number of weights and how they are structured, then the size of the network positively affects its modelling potential. Similarly to how human brains can process more intricate relations than other species with fewer neurons, a linear regression will never be able to accurately model a quadratic data distribution - Figure 4.4(a).

In this sense, when designing a neural network, it is relevant to account for the **number of parameters** of the network, meaning the size of the network or the number of adjustable weights. For all contrasting modelling strategies that may come (in this work as in general), it is often pertinent to insure that the opposing candidates have similar magnitude order in their number of parameters, as a necessary condition for a fair comparison.

One of the hyperparameters that contributes to the size of the network and that is adjustable in our implementation is the number of **neurons per hidden layer**: large hidden layers are greatly expressive being prone to overfit and the opposite goes for small-sized hidden layers. Another technique to avoid overfit is to include **drop-out layers** to the network. This process consists of ignoring each node

of the layer with a probability $p$ during training, resulting in a sparser network (Figure 4.5). This makes the model more robust as it holds the active nodes more accountable and less prone to be making mistakes that are masked by following layers [184]. Moreover, dropout encourages the network to actually learn a sparse representation.

### 4.3.3 Training

After adjusting our neural network, the training (and validation) procedure also includes important decisions. The **number of epochs**, for example, can make the difference between a good fit and a poor one. Each epoch represents one full cycle where the learning model goes over the entire training set. As one epoch is (usually) not enough for the model to optimize its randomly initialized internal weights, multiple iterations follow with consecutive parameter update. As the training loss (usually) keeps going down, the same doesn't happen for the validation curve and the training should stop while there is small error - Figure 4.4(e) - and before it overfits - Figure 4.4(f).

**Early-stopping** is a strategy that can help fine-tuning the training period as it can be programmed to stop when the validation loss curve starts going up, instead of making the model train for a fixed pre-set number of epochs. With the necessary data for good generalization increasing, there is often a trade-off between data fitting (training for a few hundred epochs) and training time (each epoch might take some minutes). **Batch size** regulation helps in such scenarios: batch size defines the number of samples to work through before updating the internal model parameters. Larger batch sizes allow computational speedups but too large of a batch size may lead to poor generalization [185].

Finally, the **learning rate** is a training hyperparameter of utmost importance and it should be adjusted in accordance with the other settings (for instance, some optimization literature has shown that increasing the learning rate can compensate for larger batch sizes [186]). It plays an essential role in backpropagation as it defines the pace of the weights update and therefore the speed of the learning process. Too large learning rates can lead to instability or cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck in local minima [187].

### 4.3.4 Prediction

We define prediction hyperparameters as those that directly affect the connectivity or dynamics inference process and they are probably the most relevant parameters for the experiments that follow.

Regarding connectivity prediction, we recall that this module uses a sequence of states to infer a connectivity graph in an unsupervised manner. Nonetheless, the NRI model can also work as a supervised framework: in this variation, the NRI decoder ignores the output graph of the encoder and replaces its weights by the ground truth edges. This **supervision** selection will be further addressed. Also, in order to infer the structure graph, the **number of edge types** must be decided. Since graphs can represent diverse structures, it might be meaningful in some representations to have different types of edges (different relations). For instance, in order to characterize the connectivity of the discretized point structure of a snail, at least two types of relationships would be required: one for the

elastic (gooey) soft body and another for the rigid body structure of the shell. In the case of a robotic soft gripper, the relational topology seems common to all connections but can also be divided into two types, one for each orientation of the directed graph.

There are other relevant hyperparameters regarding dynamics inference. As we have seen before in 4.2.2, some implementations of the NRI model will use multi-step prediction while others should use a single prediction preceded by a burn-in phase. In both cases, the **prediction step** - meaning the time window of the requested prediction - is of paramount interest. Since we are trying to understand how well structure-based models predict ahead in time, it is certainly relevant to decide how many time steps do we want it to predict. The end-goal of the prediction is also relevant: in this case, the dynamics predictor temporal step might be useful and appropriate for online control. Moreover, this parameter becomes particularly critical when comparing different approaches: some models might be better for short-time predictions while others may have worse initial forecasts but be more resilient to deterioration. For the models that need an internal state initialization, like the RNN, the **burn-in phase** will determine the time context window that the model has access to before the prediction: it should be large enough not to compromise the model internal state and its performance but should not provide the model with too much context as this will be advantageous for the following predictions.

## 4.4 Conclusion

In this chapter, we introduced the Sensorimotor Graph (see Figure 4.6). After setting the background in Chapters 2 and 3, we were able to define a framework that combines Graph Neural Networks and Sensorimotor Learning to model the action-conditioned dynamics of a robotics soft gripper.

We described our framework as divided into two sequential modules - one for connectivity inference and the other for dynamics inference - that together constitute a meaningful modeller of relational dynamics and explicit structure. To implement this architecture, we selected the Neural Relational Inference model, which learns explicit connectivity and infers future dynamics jointly and in an unsupervised manner, with results in multiple relational systems. After describing this strategy and the adaptations required to our context, we concluded this chapter by expanding on the different hyperparameters that regulate the learning process, their impact and significance.

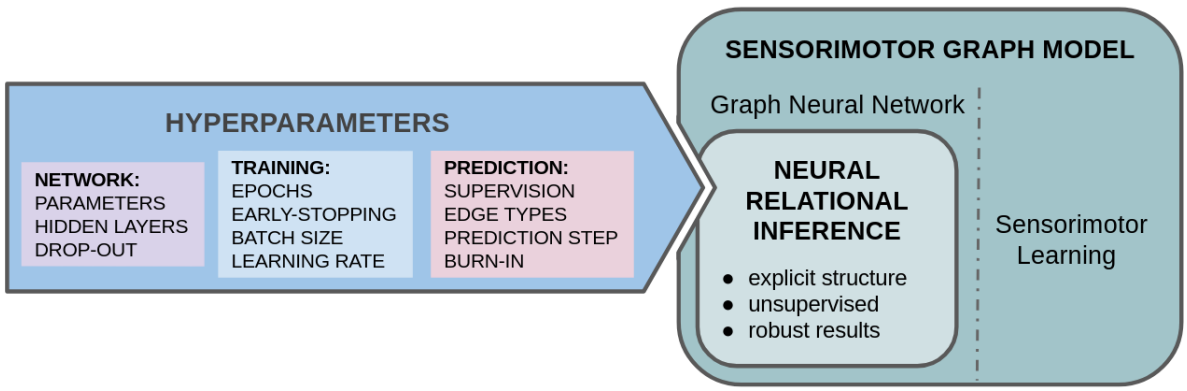



**Figure 4.6:** Chapter 4: main concepts and their relation

# 5

# Experimental Setup

## Contents

Experimentation plays an integral part in the scientific method. It can be divided into three stages [188]: we put forth conjectures and hypotheses based on the current state of knowledge, then we experiment on collected data to address unknown aspects of the problem and finally, we analyse the experimental results which may lead to one hypothesis being favoured over others or to new questions and investigations, so that the process is repeated, with the accumulation of additional knowledge about the scientific process under investigation. In the scientific method, an experiment is an empirical procedure that arbitrates competing models or hypotheses [189]. It is a rigorous procedure covering natural sciences, engineering or social sciences.

Even on a daily basis, we do experimentation constantly and without realizing: when we are cooking, when we try on new clothes, when we seek to find the fastest route to get somewhere. Even as little kids, we learn to socialize by analysing the response of our parents and friends to different stimuli because we learn from an early age that one good way to learn from everyday systems is to disturb them and then observe them.

Although early civilizations like the ancient Egyptians, Babylonians or classical Greeks have laid the foundations to what may be considered the scientific methodology, it was later, with the Islamic cultures - mainly Arab and Persian - that the inductive experimental method emerged. Experimentation and quantification began in the medieval Islamic world (personified by the Arab physicist Ibn al-Haytham [190]) since there was a greater emphasis on combining theory with practice and it was common for those studying natural sciences to be artisans as well - which would be considered an aberration in the classical world.

The process of scientific experimentation can be broken down into two basic components: data and method. 'Data' (pl. *Datum* - latin for "given") is the information, in the form of numbers, documents, images, etc, that can be used to analyse, measure, interpret and extract meaningful conclusions while the 'Method' (*Methodos* - greek for "pursuit of path/knowledge") refers to the algorithms used to process that data and extract the valuable information. In the particular case of computer science, these methods are powerful learning algorithms and the data plays a particularly important role as the success of these algorithms relies deeply on the quantity and quality of the data. This is guaranteed by meticulous processes of data generation and preparation before using the final data sets.

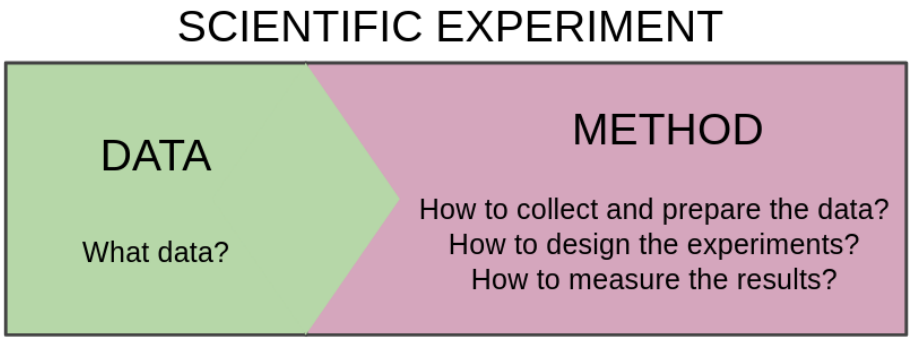



**Figure 5.1:** The two elements of scientific experimentation: data and method

In this chapter, we describe the experimental methodology used in this work. We start by describing the simplest and most important aspect of experimentation: the data used. We will describe its

structure, how we collected it and how we organized the data. Then, we will move on to describe the baselines we will use for our evaluation. In specific, we will describe the four distinct baseline models that will be compared to our hypothesis - the SMG. Finally, we shall specify the different experiments we will perform to test our proposed model against the baselines. We shall thoroughly describe the different experiments so that they can be replicated and for a better interpretation of the results.

## 5.1 Data Preparation

As we have discussed in the introduction of this section, data is a key part of scientific experimentation, as data can be seen as raw information and information can be seen as raw knowledge. In deep learning, data is the essential fuel of all algorithms. It is often collected as raw (numerical) symbols that must be prepared and organized before being valuable to learning models.

In this section, we will focus on the description of our data. We will start by our model domain, an abstract description of our subject system. Then, we will relax some assumptions to transit from the abstract system to a physics simulator and outline how the data is collected. At last, we will describe how the data is finally organized to be fuelled to the experiments.

### 5.1.1 Domain

We have described in detail the Sensorimotor Graph, its biological inspiration and the motivation for its application in soft robotics. As one of the most common deployments of soft materials in robotics still is for manipulating tasks, we decided to focus our experiments around a simple yet representative end-effector: the soft gripper.

Robotic grippers are (usually) hand-inspired impactive end-effectors that use force closure mechanisms to grab objects. The differences to soft grippers are two: the non-rigid materials that build the fingers and whose flexibility widens gripping applications; and the simplified actuation (single or few constraints).

The domain of our experiments will then be a **robotic three-joint soft gripper**. In order to include some variance in the configuration and parameters of the subject of our work, we shall allow variations in the number of fingers (three or four), the configuration of the fingers in the hand (they might be next to each other, equidistant, etc) and the elasticity of the soft material. The possible number of fingers were selected given the widespread usage of three- and four-finger grippers and being the minimum number of fingers for a precise control of the orientation of the handled object and for a precise manipulation of spherical objects [191, 192].

Furthermore, it is important to stress the goal of this experiment, the input and desired output. The purpose of this experiment is to learn the dynamics of the soft gripper by "observation": we know the control actions being applied to this hand and we can see how these actions are affecting the motion of the fingers; we use this knowledge to infer a explicit connectivity, the kinematic chain of our robotic system; finally, we use the actions, trajectories and the connectivity to model the dynamics.

Now we must move from the abstract domain of our experiment subject and end-goal to the prac-

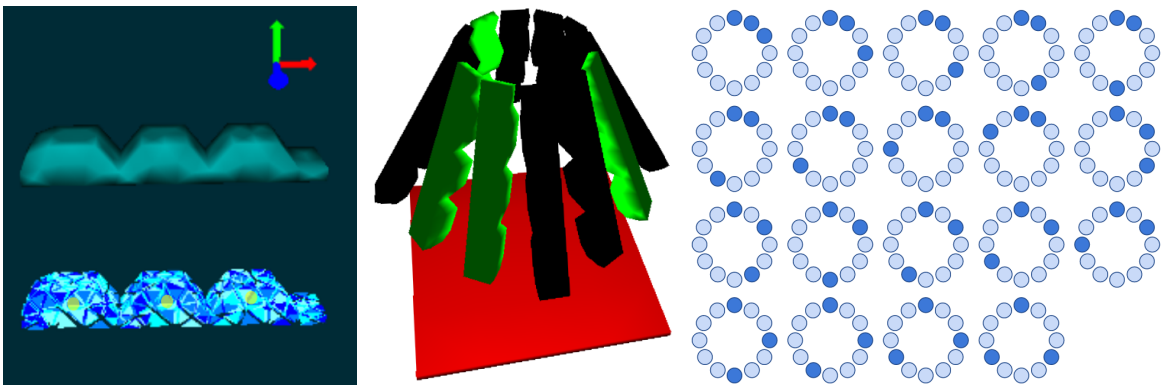
tical procedure to represent it and carry it out. To do this, we wish to represent the dynamics of this hand in a simple but accurate manner, hence we use a physics simulator to create our soft end-effector and to act on it. When deploying this framework in real end-effectors, the observational data coming from the simulator should be obtained, online, whether by placing markers on different fingers or by learning to extract and identify key-point representations in an unsupervised manner [87, 92].

When representing our soft hand in the simulator, we can discretize this gripper and extract only a few spatial points (one per joint is sufficient). The practical procedure to represent and collect this data is explained next. From the physics simulator, we can extract the three-dimensional position of our discretized hand, but we can also actuate this hand using finger-contraction orders. When infering the kinematic and dynamical model, this input actions (one for each finger) are not associated with the finger they are intended to actuate and this association must be learned by our model. This simulates real-life applications, where we often have easy access to what control actions are being fed to the system, but specifying which order concerns each motor might be challenging, or tedious.

### 5.1.2 Data Collection

In order to collect diverse data from a soft hand gripper in motion, the Simulation Open Framework Architecture (SOFA) [193] was used, alongside with the Soft Robotics Toolkit [194]. This Toolkit includes many shared resources to support the design, fabrication, modelling, characterization, and control of soft robotic devices.

The soft hand configuration is based on one of the cable-actuated hands in the showcase of Soft Robotics Toolkit, where three-joint fingers are actuated by a non-extensible pulling cable - Figure 5.2(a). The finger base is fixed and the variation in the cable pull actuation signal ($\delta_a(x) \in [\delta_{min}, \delta_{max}]$) allows the soft finger to experience a wide range of motions.



**(a)** 3D mesh of finger and actuation cable with three sampled points

**(b)** SOFA environment with 12 overlapped finger positions - only three or four are usually active

**(c)** Nineteen circular combinations of 3 positions out of 12.

**Figure 5.2:** Data generation on the SOFA simulator environment

Variability is added to the scene by different sources. First, structural diversity is created by in-

cluding fingers in different number and configurations around a dodecagon - Figure 5.2(b). Since the relevance of the different configurations is the variation in their *relative* position, the number of possible configurations for $n$ fingers is limited to the number of circular combinations. Circular combinations can be obtained by Polya Enumeration Theorem (a detailed explanation of the following steps can be found in [195]). For our case of 12 elements $x_{1:12}$, we obtain the polynomial known as the cycle index:

$$P_{C_{12}}(x_1, x_2, ..., x_{12}) = \frac{x_1^{12}}{12} + \frac{x_2^6}{12} + \frac{x_3^4}{6} + \frac{x_4^3}{6} + \frac{x_6^2}{6} + \frac{x_{12}}{3} \tag{5.1}$$

Since we try to find the number of configurations using 2 colours (dark/light blue; finger/void), it would make sense to substitute $x_i = d^i + l^i$. However, since the 2 colours are dependant and one can be deducted from the other, we write $x_i = d^i + 1$ and get the generating function

$$P_{C_{12}}(1+d, 1+d^2, ..., 1+d^{12}) = d^{12} + d^{11} + 6d^{10} + 19d^9 + 43d^8 + 66d^7 + 80d^6 + 66d^5 + 43d^4 + 19d^3 + 6d^2 + d + 1 \tag{5.2}$$

from which we can obtain the number of configurations with $k$ nodes from the coefficient on $b^k$. Figure 5.2(c) shows the 19 possible configurations for three-finger grippers. Also, the actuation signal for each sample is one of a set of predefined waves that were built from trigonometric and random functions. For each sample, the movement between fingers varies only in amplitude, frequency and phase. Finally, the gripper material elasticity is adjusted through the variation of the Young modulus $E$ and the Poisson ratio $\nu$, obeying

$$constrained\ modulus = E\frac{1-\nu}{(1+\nu)(1-2\nu)} \tag{5.3}$$

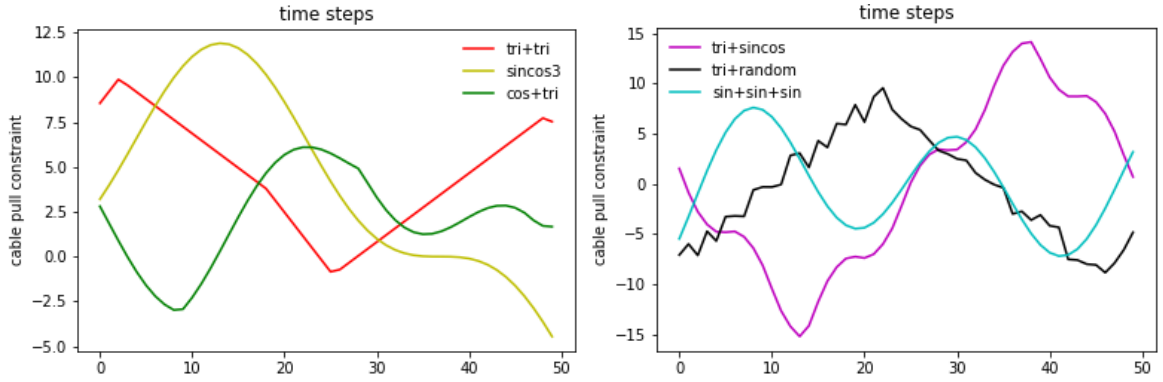
Each run in the SOFA environment creates a sequence of 100 sampled time steps. Generating hundreds of distinct configurations might take several hours. For each finger, three points are sampled, one for (the center) of each joint, as in Figure 5.2(a). The data we generated is available for future use (see Section 7.2).

### 5.1.3 Data Sets

A total of six different data sets are extracted from the soft gripper simulation. The training set for all tests includes only four fingers in a total of thirty different configurations (relative positions) and with four different values of elasticity. We shall reference this set as *Trainset*. For each sample, the type of motion applied to the cable pull actuation signal for every finger is the same, although the characteristics of the motion - amplitude, frequency and phase - are different and randomized. Three different types of motion are used, all resulting from operations between trigonometric functions - Figure 5.3(a). Our first set, *Testset 0*, includes novel configurations, some combinations of motion and elasticity unseen at training time, as well as a different cable pull actuation signal. We will use *Testset 0* for a (first) baseline test. Then, we created four sets to test the model in different validation sets with varying conditions. The validation sets are set up to understand:

- *Testset 1* - How well the model generalizes to previously unseen motions.
- *Testset 2* - Effect of system configuration (relative position of joints) deviating from the training configurations, either due to sensor measurement errors or due to versatility in the end-effector kinematics.
- *Testset 3* - Adaptation to changes in the number of points, either due to mechanical failure of one of the fingers or sensor measurements failing to provide the position for any of the points.
- *Testset 4* - Robustness to changes in the order of the joint measurements, either due to tracking errors or wrong correspondences between detected parts.

More specifically, the four validation sets are used to test the model generalization to motion, relative position, number of fingers and order of the points, respectively. The first test set, *Testset 1*, uses three different motions and includes random noise - Figure 5.3(b). The second test set, *Testset 2*, uses the same motions and elasticities as the training set but for thirteen unseen configurations. The third test set, *Testset 3*, uses only three-finger configurations with the same motions and elasticities whereas the fourth, *Testset 4*, and last validation set, shuffles the order of the twelve points for each sample.




**(a)** Training **(b)** Validation

**Figure 5.3:** Plot of different types of motions in *Testset 1*

| Ablation Study | |
|---|---|
| **Test sets** | **Description** |
| *tri+tri* | $A_1 * [asin(sin(t/f_1 + p_1) + asin(sin(t/f_2 + p_1))]$ |
| *sincoscube* | $A_2 * [sin(t/f_3 + p_2) + cos(t/f_3 + p_2)]^3$ |
| *cos+tri* | $A_3 * cos(t/f_4 + p_3) + 0.4 * asin(sin(t/f_5 + p_3))$ |
| *tri+sincos* | $A_4 * [asin(sin(t/f_6 + p_4) + 2 * sin(cos(t/f_7 + p_4))$ |
| *tri+random* | $A_5 * asin(sin(t/f_8 + p_5) + 0.15 * rand[-1, 1]$ |
| *sin+sin+sin* | $A_6 * [sin(t/f_9 + p_6) + sin(t/f_{10} + p_7) + sin(t/f_0 + p_8)$ |

**Table 5.1:** Motions description in *Testset 1*

## 5.2 Baseline Models

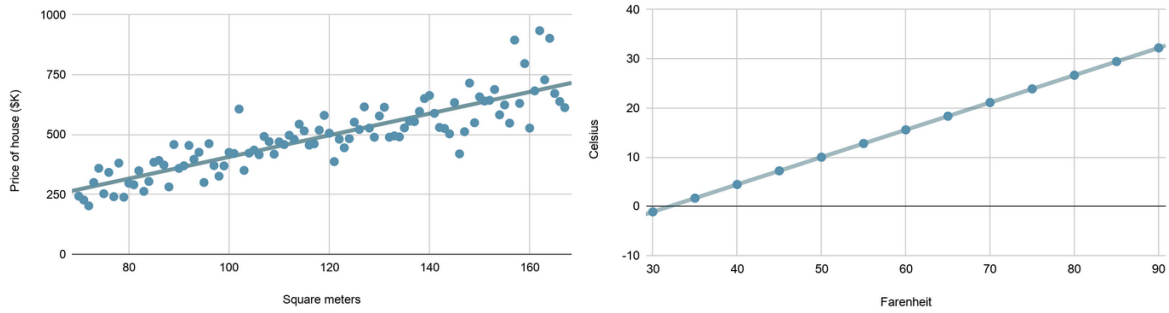
In order to prove our claim that structure is key to model a robotic soft hand's non-rigid kinematic chain and that our Sensorimotor Graph model should rely on GNN-based strategies, we must find other baseline approaches to compare with. In particular, we must compare the performance and robustness of our selected NRI model, qualitative and quantitatively, in tasks of dynamics prediction, with opposing non-structural approaches.

For this purpose, four different baseline models were selected to assess, by comparison, different aspects of the unsupervised NRI model. A brief explanation of each model, its characteristics and relevance for this work is provided below.

### 5.2.1 LINEAR

The linear model is (one of) the simplest models and yet a powerful tool with still many applications. This method goes back to the beginning of the $XIX^{th}$ century when Legendre and Gauss used least squares linear regression to predict planetary movement [196], being the first type of regression analysis to be studied rigorously and to be used extensively in practical applications [197].

As the name indicates, it is an ideal model for variables that behave linearly or in a similar way, for example, the relation between Celsius and Farenheit or the price of a house in central Lisbon and its floor area. In Figure 5.4 we can see an example of this type of regression for these two examples.




**(a)** Price of square meter in central Lisbon **(b)** Conversion between Celsius and Farenheit

**Figure 5.4:** Linearly dependant variable

The general (multivariate) linear regression model can be compactly written as in 5.4, where Y is a matrix with series of multivariate measurements, X is a matrix of observations of independent variables that might be a design matrix, B is a matrix containing parameters to be estimated and U is a matrix containing errors (noise). The errors are usually assumed to be uncorrelated across measurements, and follow a multivariate normal distribution.

$$\mathbf{Y} = \mathbf{XB} + \mathbf{U} \tag{5.4}$$

One of the most common approaches to solving linear regressions is the least squares method, which finds the optimal parameter values by minimizing the sum of squared residuals, $S = \sum_{i=1}^{n}(y_i -$

$f(x_i, \beta))^2$, where the model function $f(x_i, \beta)$ uses the vector $\beta$ with $m$ adjustable parameters.

Let us now imagine a body moving from point A to point B in a short time. In this case, a fixed velocity assumption may result in a fair approximation for its movement. The linear model simplicity makes it computationally inexpensive but also prone to underfit - Figure 4.4(a). It is, however, a relevant baseline for our case since, just like for the previous example, constant velocity assumptions are good approximations in physics simulations for short time periods. It is, therefore, an important sanity test to check if our soft hand kinematics are sufficiently non-linear as we suspect and to assure that our prediction window is large enough, as warned before in 4.2.2.

### 5.2.2 MLP

By interleaving linear with non-linear layers, the Multi-Layer Perceptron (MLP) exponentiates its representational power. MLPs are popular robust universal approximators and have been used for a different number of applications including classification, regression and time series prediction. These feed-forward neural networks are also the base unit for the NRI encoder and decoder.

Also called feed forward neural networks, the MLP is based on a simpler unit, the perceptron [198] - Figure 5.5(a). The perceptron consists of a weighted sum $(z)$ of inputs $(x_1,\ x_2,\ ...\ ,\ x_n)$ whose signal (positive or negative) determines the sign of the unitary output $\phi(z)$. When there is a mismatch between the true and predicted labels, its weight is updated with $w = w + yx$. This update rule is enough to empower the perceptron with the ability to find the best N-plane division in a N+1 dimensional space - Figure 5.5(c).

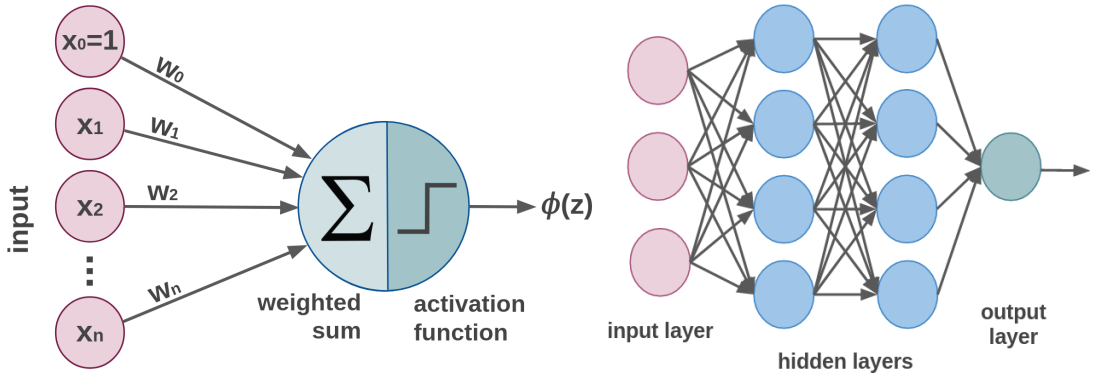
The MLP is a set of perceptrons that are stacked in several layers to solve more complex problems. In specific, single perceptrons can only learn linear separations of space while MLPs are capable of non-linear regression - Figure 5.5(d). This is achieved by the internal (hidden) layers that can be as many as the complexity of our problem aks for. Also, the activation function that maps the weighted sum of all neurons to the output has a major impact in the model's convergence and efficiency. In Figure 5.5(b) we can see a fully connected 2-layer MLP where the parameters of each unit are independent of the rest of the units in the layer, meaning each unit possesses a unique set of weights.

Our MLP implementation is of a fully connected network with 2 (non-linear) ReLu hidden layers. Just like the linear model, it receives a context window and predicts the next position - which will be incorporated in the context window for the following prediction - for $p_s > 0$ prediction steps.

### 5.2.3 LSTM

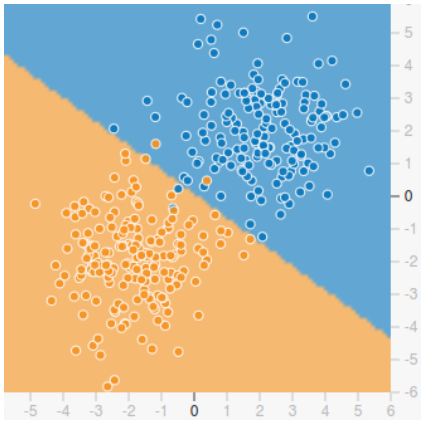
The third baseline model and last non-structure based approach is the Long Short-Term Memory (LSTM). This is a type of RNN, able to learn order dependency in sequence prediction problems. Unlike feed forward neural networks, the Long Short-Term Memory has feedback connections and is able to process entire sequences of data, being applicable to speech or image recognition, for example.

The LSTM keeps an internal state that can represent context information about past inputs for an amount of time that is not fixed *a priori* but rather depends on its weights and on the input data

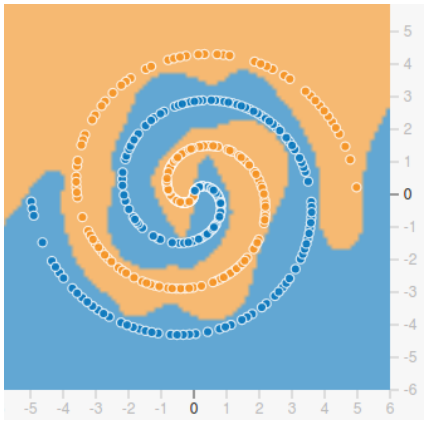



**(a)** Perceptron

**(b)** MLP



**(c)** Perceptron classifier (source: Tensorflow playground [199])



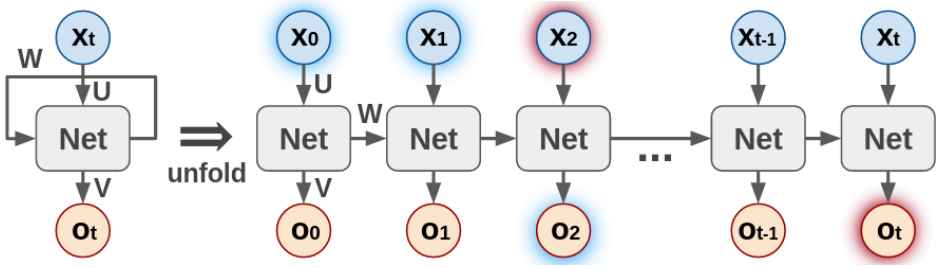
**(d)** MLP classifier (source: Tensorflow playground [199])

**Figure 5.5:** Perceptron and MLP

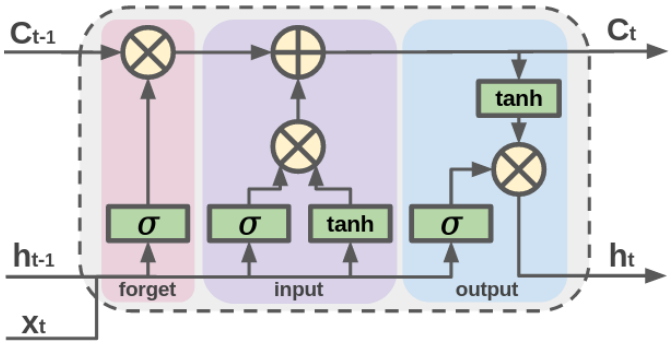
[200]. This constitutes an addition in complexity that brings the promise of new behaviours that the traditional methods cannot achieve.

The LSTM model was created to answer the gradient vanishing and exploding problems. Although the RNNs already have internal memory and can process larger sequences of input, they still struggle with long-term dependencies. This issue relates to the back-propagation of gradients all the way from deeper to the initial layer. As these gradients (smaller or greater than 1) go through continuous matrix multiplications, they will shrink or grow exponentially, making the learning infeasible. In Figure 5.6(a) we can see how an RNN can be equivalent to an unrolled sequence of copies of the neural network and how long-term dependencies (red) can be more difficult to learn than short-term dependencies (blue). In Figure 5.6(a), $x_t$ and $o_t$ are the input vector and the output at time $t$; U, V and W are the weights of the hidden layer, the output layer and the hidden state, respectively.

The LSTM network is a modified version of the RNN, which tries to make it easier to remember past data in memory - Figure 5.6(b). In short, this architecture usually includes a cell state (the memory of the LSTM unit, keeping track of the dependencies between the elements in the input sequence) and three gates which regulate the flow of information inside the LSTM unit: an input gate (controlling the extent to which a new value flows into the cell), a forget gate (controlling the extent to which a value




**(a)** Recurrent Neural Network




**(b)** Long Short-Term Memory

**Figure 5.6:** Artificial recurrent neural network architectures

remains in the cell) and an output gate (controlling the extent to which the value in the cell is used to compute the output activation of the LSTM unit). In Figure 5.6(b), $C_t$ is the cell state, $h_t$ the hidden state, $x_t$ is the input vector and $\sigma$ and $tanh$ are activation functions.

The implementation used was the same as the proposed baseline in [91]: a two-layer LSTM with shared parameters and 256 hidden units that models each trajectory individually, preceded and followed by a two-layer MLP (256 hidden units and ReLU activations). The implementation included the adaptations mentioned in 4.2.3 and a burn-in phase of 50 (fifty) time steps to initialize its internal state.

### 5.2.4 Supervised NRI

Although this already constitutes a version of the GNN-based NRI model, we include its supervised version here for it also sets a comparative baseline to its unsupervised implementation.

We saw in section 4.2 that the NRI model takes the form of a variational autoencoder to jointly infer an explicit connectivity graph and the system dynamics. This includes an encoder, that learns connectivity, followed by a decoder that uses this graph to predict motion forward in time. As mentioned before, in 4.3.4, in this supervised baseline, the decoder will ignore the output graph of the encoder and replaces its weights by the ground truth edges, as we can see represented in Figure 5.7.

Since this version uses supervised training on the encoder and focuses on evaluating the performance of the *dynamics inference module* for a given (accurate) connectivity graph, this baseline is ideal to test the importance of the *right* structure in dynamics inference.

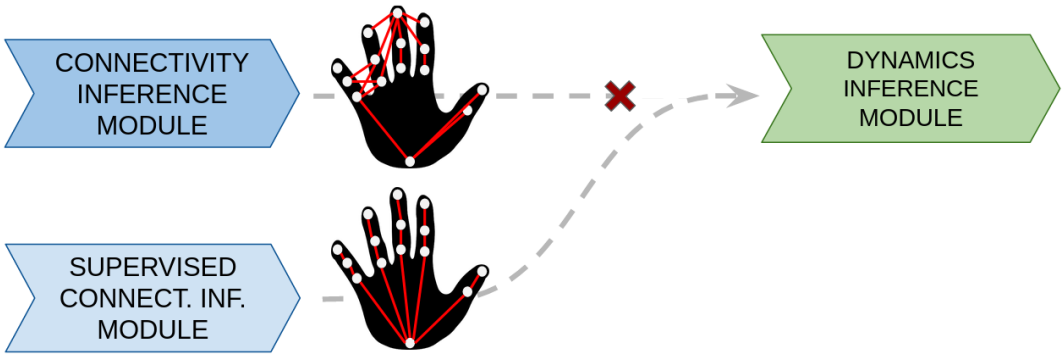



**Figure 5.7:** Supervised NRI baseline: the output of the encoder is replaced by ground truth graph connectivity

## 5.3 Experiments

At this time, we resume our explanation at the beginning of this chapter - Figure 5.1 - about the two key aspects of experimentation: data and method. They work very closely together - and often interdependently - to produce a valuable experiment and are both necessary (and sufficient?) to draw meaningful conclusions. The importance of method is about the importance of asking the right questions. Data is about asking them to the right subject.

In Section 5.1 we focused on how the data was collected and organized. This already answered some of the questions regarding the data preparation methodology. After that, in section 5.2 we took a step back to understand what are the targets of our experimentation; what our proposed model SMG will be tested against; what different approaches will go through this experimental process. Now, in this next section, we will look at the second major aspect of experimentation: the method(ology). In specific, we shall systematically write down the design of the different experiments to assure reliable and replicable results.

### 5.3.1 Baseline Prediction

In the first experiment, we shall assess a model's ability to generalize to relatively new scenarios and, for this reason, we shall refer to this experiment as the *Baseline Prediction*.

During training, a model will look at different sequences of a four-finger robotic soft hand movement. The observed trajectory of this gripper will be different every time since there shall be different configurations (relative position of different fingers), motions (function and parameters) and finger elasticities, in a total of 300 combined variations. For each sample, the type of motion applied to the cable pull actuation signal is the same for every finger, although the characteristics of the motion - amplitude, frequency and phase - are different and randomized. Three different types of motion are used, depicted in Figure 5.3(a), all resulting from operations between trigonometric functions.

In the *Baseline Prediction*, the trained model will then face some changes at test time. The validation set for this experiment includes novel configurations, some combinations of motion and elasticity unseen at training time, as well as a different cable pull actuation signal.

In order to assess the alleged robustness of the SMG model, we will test the NRI and all four baseline models. With this experiment, we want to evaluate the error of each model predicting unseen

scenarios for 10 steps ahead in time and obtain a qualitative and quantitative hierarchy of the different model predictions.

This task and the variety of modelling architectures (linear, MLP, LSTM and NRI with four variants) had two practical challenges that are worth mentioning: 1) adjusting the hyperparameters of each network to optimize data fit and reduce under or overfitting; 2) high computational time (and also space), particularly with some NRI variants and the LSTM.

### 5.3.2 Ablation Study

After a broad approach to this generalization problem, with the test set including mixed small variations to similar scenarios, we now want to disentangle the multiple sources of variability. With this purpose, an ablation study is carried out, where all aspects are kept the same as in training time, except for a single varying ingredient.

When using the SOFA framework to simulate soft hand movements, different parameters can be adjusted from sample to sample. For this ablation study, we select four of these parameters we consider the most relevant. Each of them uses one of the *Testsets 1* to *4*, described in 5.1.3 and the training set for all four experiments is the same as before, *Trainset*. A summary of the ablation study data sets and the issues they address can be found in 5.2.

| Ablation Study | |
|---|---|
| **Test sets** | **Description** |
| *Testset 1* | Generalization to unseen motions |
| *Testset 2* | Generalization to unseen configurations |
| *Testset 3* | Generalization to different number of points |
| *Testset 4* | Generalization to different order of points |

**Table 5.2:** Four experiments in the ablation study

Most often, a robotic soft body does not change its characteristics but is required to perform a plethora of limb contractions or extensions (to grab an object, to move in the water, etc). Sometimes, it might even find some external or internal disturbances that make this soft body limbs perform a sudden unexpected movement. For all these reasons, the model we use to predict and control its limbs trajectories will never see every type of motion sequence at training time and will have to deal with new accelerations or new shifts. The first experiment of the ablation study uses three different motions - Figure 5.3(b) - at test time and includes random noise in a total of 300 different samples.

Also, the effect of the system configuration (relative position of joints or limbs) might deviate from the training configurations. This can be either due to sensor measurement errors or due to changes in the end-effector. In the second condition of our ablation study, *Testset 2* uses the same motions and elasticities as the training set but for thirteen unseen configurations.

Furthermore, we consider the situation where there is an unexpected number of discretized system points (nodes). Either due to mechanical failure of one of the limbs or to sensor measurements failing to provide the position for any of the joints, it may happen some nodes of the soft body provide no value (or retrieve an absurd one). We assess this situation by zero padding three sequential joints,

simulating moving from a four-finger to a three-finger hand system. *Testset 3* uses only three-finger configurations with the same motions and elasticities as before.

Finally, we want to assess the model robustness to changes in the order of the joint measurements. An altered order in the desired sequence of points (nodes) might happen either due to tracking errors or wrong correspondences between detected parts. Moreover, it might happen that, for a certain system, it is desirable to have a mutable order of points for simplicity or flexibility. For all this, we create a fourth and final ablation experiment where we keep the exact same data as in *Trainset* but we shuffle the order of the (twelve) points.

For this experiment, we still use a prediction step of 10. We expect the resulting hierarchy from the baseline prediction already allows us to select the best performing NRI model and the best performing baselines to further study in this ablation work.

### 5.3.3 Connectivity

After qualitatively assessing the *dynamics inference module* (Figure 4.1) of our structure-based architecture, we shall now look inside this process to evaluate the output graph of the *connectivity module* and how this unit works under different conditions.

In this section, we shall look at the NRI model only, since it is the only approach that estimates structure. When training the full model (encoder+decoder) in the supervised approach, the graph that outputs the encoder is switched by its ground truth (g.t.) weights - Figure 5.7. We can do this since we know that the encoder can be trained separately, in a supervised fashion, and with almost perfect accuracy. Here, we train the supervised encoder to see if it can in fact recover g.t. connectivity and compare it to the structure graph in the unsupervised approach. It is relevant to see how distinct the two are to understand what justifies the difference in the predictions of both strategies. In the supervised approach, we will include dropout rate of 0.5 and early stopping but keep the same parameters as before: encoder hidden layers with 256 units, learning rate of 5x$10^{-4}$ and batch size 32.

Next, we will test the effect of one hyperparameter mentioned in Section 4.3.4. The number of edge types affects how many different relations must be considered and weighted in the node-edge message passing mechanism. As explained in Section 4.2.2, literature [149] suggests increasing the number of edge types should make the dependency on the edge type more explicit and harder to be ignored, leading to a greater responsibility of the encoder in the process and therefore better results. Although one may claim that it does not make much sense in our system to have different edge types, we believe our soft gripper can be considered to have two edge types (connectivity between two adjacent joints or no connectivity) or three (one relation type when there is no connectivity at all and one for each orientation of the directed graph). Although only two relation types are considered for the other experiments (and not three), here we shall compare both strategies. With that purpose, the training should be similar to before, although this time we increase the size of the hidden layers of the encoder from 256 to 400 and reduce early-stopping patience to avoid overfitting.

Here we consider the same conditions as in the first experiment - *Trainset* and *Testset 0* for a prediction window of 10 time steps.

### 5.3.4 Prediction Step

At last, we want to see how another relevant hyperparameter affects the results of our structure-based model, the prediction step. This variable defines how many time steps the model will learn to predict and also on how many steps its accuracy will be tested.

The prediction step ($ps$) is of utmost practical importance. On the previous experiments, we used $ps = 10$ as it seemed a reasonable time window for control applications and as it already allowed for some variation in the data. However, for some applications, different prediction windows could be required and it is relevant to test if the increase and decrease of the number of predicted steps leads to under or overfit, respectively.

In this experiment, we shall vary the prediction window (with $ps = 5, 10, 15, 20, 25$) of the NRI and, for all values, plot the evolution of the prediction error for 25 time steps. The parameters of the model shall not be changed (same network dimension) and the training is going to last 200 epochs, although with varying patience to allow early stopping. For this experiment, we can use only the best performing variant of the unsupervised NRI.

## 5.4 Conclusion

In this chapter, we moved from the abstract plane of inspiration and motivation that governed Chapters 2 and 3 and from the general perspective of our framework in Chapter 4 to dive in the most material chapter so far where many concepts aforementioned take practical shape to create a picture of how experimentation is going to be - Figure 5.8.

We started by learning our practical domain, the non-rigid gripper kinematic chain and its representation for data collection purposes. Next, we focused on data organization and how the different sets vary among them. We moved on to briefly detail the baseline models as they will serve as non-structure-based or supervised references to compare with our structure-based SMG. Finally, we dived into the experiments themselves, listing the different answers we want to find and why they are relevant to our study.

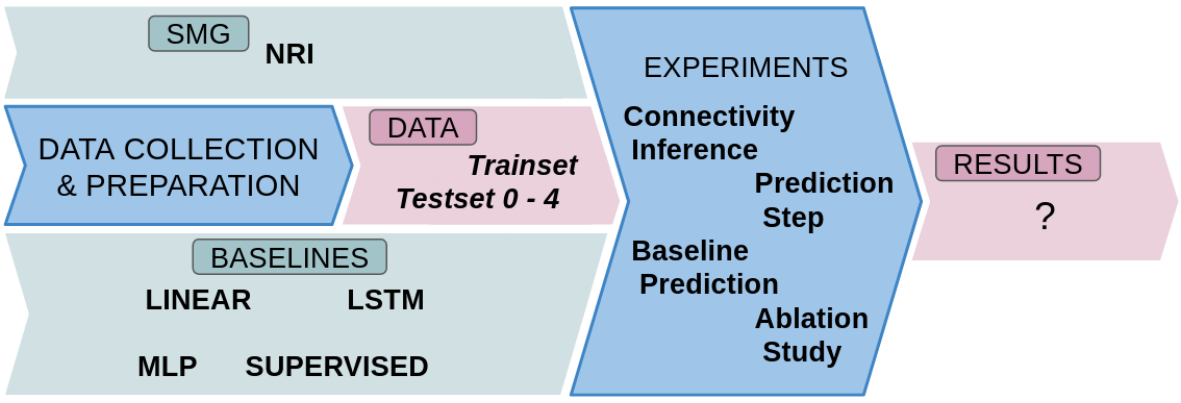



**Figure 5.8:** Chapter 5: data and method creating meaningful results

# 6

# Results

## Contents

For the reader who has been following us from the start, it has been a long (and hopefully rewarding) journey. We now reach what is probably the most relevant chapter of all, the outcome of the narrative we have been telling so far, where all of the previous chapters climax.

As mentioned in the introduction, if we were to divide the work in this thesis into three parts, the first would be the motivational background contained in Chapters 2 and 3, the second would focus on our proposed solution (Chapter 4) and the last part would concentrate on testing this solution, in Chapters 5 and 6. Being the second half of this last part, after describing how is the experimentation going to happen, this chapter will focus on the results of such experimentation.

Lord Kelvin once wrote (what would be contracted as) 'To measure is to know' [201]. I would go further and add that 'To measure is to know, *if we know how to measure*.'. The first part of this chapter will focus on the performance metrics used, since knowing how to measure our results is key to generating valid conclusions. Then, we will go through the results of the experiments described in 5.3, using the performance metrics, reviewing the experimental results and interpreting them.

## 6.1 Performance Metrics

We start by describing the evaluation metrics we shall use for this chapter. Metrics are a set of measurements that help one evaluate his results. These are used to measure the quality of a process, in this case, a deep learning model.

Many systemic divisions could be created when characterizing different types of measurements of which we highlight only one: the difference between quantitative and qualitative measuring:

- **quantitative measurement** - performance is measured by numeric metrics or statistics.
- **qualitative measurement** - performance is evaluated by intangible non-numerical information.

The importance of metrics is indisputable in scientific research and has been the subject of many studies [202, 203]. The metrics we use will shape the information we retrieve from the experiment. Moreover, the way we choose to evaluate the performance of alternative methods in an experiment defines the conclusions we draw. With this in mind, our metrics should be complete, relevant and meaningful.

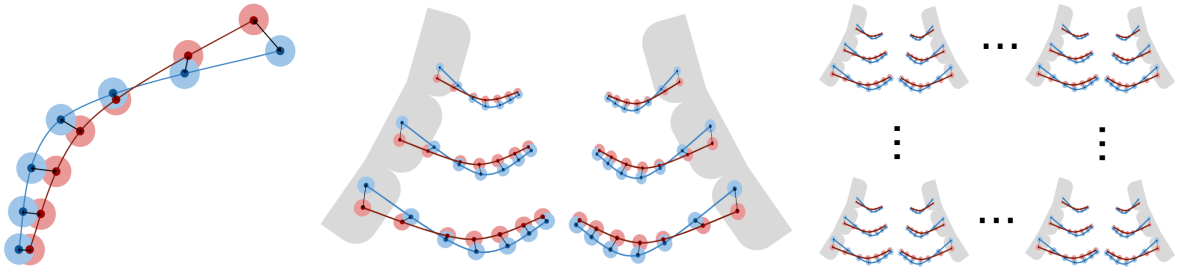
For evaluating a deep learning model, different metrics are commonly used. In this section, we describe three metrics we will be using, why is each one relevant and why they together form a complete qualitative and quantitative assessment of our experiments.

### 6.1.1 Cumulative Mean Squared Error

**Mean Square Error (MSE)** of an estimator measures the average of the square of the errors. The error is the difference between a prediction and a real value. Squaring the errors is important to remove negative values and to give more weight to larger differences. More formally, if $X_i$ is a vector of n observed values, we have that:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_{i,real} - X_{i,model})^2 \tag{6.1}$$

Figure 6.1 shows a representation of this metric for the prediction of the motion of a point, represented by a blue circle, in a similar system to ours.. As we can see, for each position prediction (in red), there is an associated error. For a certain trajectory - Figure 6.1(a), - MSE is calculated as we have seen before, using the average of the squared difference between ground truth and predicted values. As the calculation involves a multi-node system - Figure 6.1(b) - and then several variants of this system - Figure 6.1(c), - the same process applies.



**(a)** MSE over single point trajectory

**(b)** MSE over multiple points trajectories

**(c)** MSE over multiple configurations

**Figure 6.1:** MSE calculation for compositional systems

The **cumulative MSE** at time $t$ is the sum of the errors from time $0$ to $t$. This is a non-decreasing function, often convex since the predictions tend to deviate (with the error tending to grow). In the case of a soft gripper, since the movement is bounded (and sometimes periodic), the convexity may not hold. As an integration of the normal MSE, the cumulative error may offer a distinct view on the evolution of the prediction inaccuracy.

### 6.1.2 Mean Squared Error normalized to travelled distance

One of the problems with MSE is that it often does not translate into an intuitive result. If the fact that it results from several operations (average, sum and exponentiation) already makes it difficult to keep track of the original meaning, sometimes the original meaning is not so clear either. For example if, when predicting a trajectory, the resulting MSE is 1.5, would the reader find this a good or poor result? What about if it were 0.2?

The fact is that the error becomes more meaningful when compared with something else, for instance in the form of a percentage or ratio. Another metric we use besides the regular MSE is a normalized version, adjusted to the travelled distance.

The **MSE normalized to the travelled distance** or $\text{MSE}_{dist}$ can be interpreted as the average squared error of the prediction of the movement of an object compared to the travelled distance of that object. More formally,

$$MSE_{dist} = \frac{MSE}{\sum_{i=2}^{n}(X_{i,real} - X_{i-1,real})} \tag{6.2}$$

With this metric, we hope to have a better intuition of the quality of the results: $MSE_{dist}$=1 means the average squared error of the prediction is as large as the travelled distance, and therefore the prediction has deviated considerably. In our implementation, an MSE normalized to displacement was also included, although for periodic movements this result must be interpreted with caution and will not be included in the shown results.

### 6.1.3 F1 Score

We have seen how MSE metrics can be used to assess the performance of the model in terms of predicting three-dimensional trajectories. However, we are still missing some metric to evaluate the model in terms of connectivity prediction.

The problem of finding the structure of a system is translated into finding the edges of a directed graph. For this issue, each edge can only exist or not exist and the model will probabilistically classify them into these two groups. Hence, when evaluating its classification performance, a few metrics come to mind.

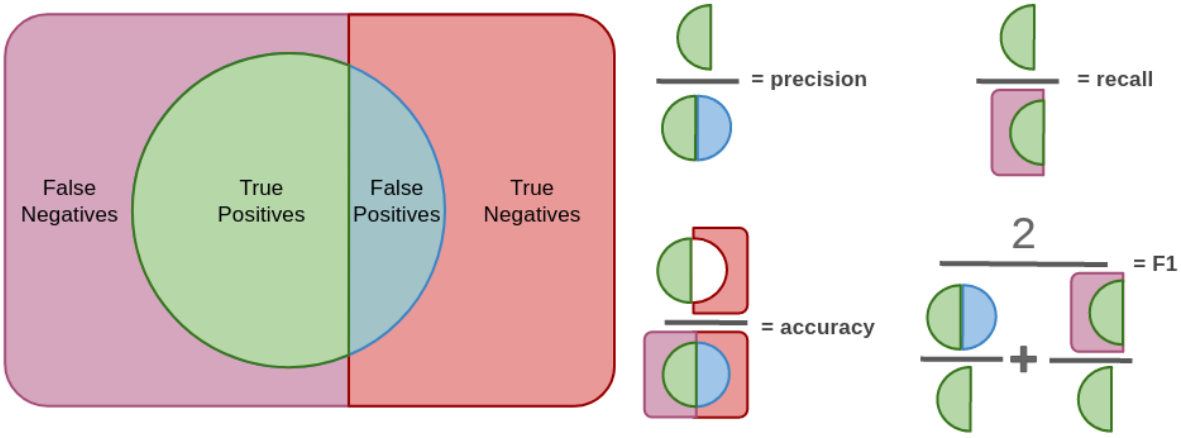



**Figure 6.2:** Representation of performance metrics regarding true and false positives and negatives

**Accuracy** measures the fraction of correctly classified elements and it is a good baseline for evaluating a classification method. However, accuracy suffers from class imbalance sensitivity. Let's imagine a model predicting the love relations in a ten people office. In reality, Sarah and Jack are in love with each other (they are happily married with two kids and three dogs) and Sebastian is in love with Courtney (with no success), making a total of three (directed) love relations out of ninety possible passions. If a model were to predict these relations (positive edges in a fully connected circular graph) and concluded there was no one in love at all, it would score more than 96% of accuracy. However, this model was probably being affected by the sparsity of the graph to always favour false negatives and would perform much worse in an American dating reality show.

**F1-score** solves this issue by weighting precision and recall (Fig. 6.2). It uses a harmonic mean to penalize larger deviations and can be written as

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} = 2 * \left( \frac{1}{\frac{TruePostives}{TruePositives+FalsePositives}} + \frac{1}{\frac{TruePostives}{TruePositives+FalseNegatives}} \right)^{-1} \quad (6.3)$$

Since, like for the office relationship example above, our connectivity graph is sparse (two positives out of each thirty-three edges), accuracy should be complemented with the F1-score and we shall use both.

## 6.2 Performance and robustness

In this work, we want to find if structure-based models - that explicitly infer system connectivity to help them predict future dynamics - *perform* and *generalize* better when applied to a robotic limb's non-rigid kinematic chain. Our proposed graph-based approach is the Sensorimotor Graph model and uses an adaptation of the unsupervised Neural Relational Inference model.

Following the experimental procedures detailed in Chapter 5 and using some of the metrics described in Section 6.1 the next subsections will focus on answering, respectively:

1. Do structure-based models outperform non-structure-based approaches for soft robotics dynamics prediction?
2. Are structure-based approaches more robust to generalization than non-structure-based approaches for this application?
3. While modelling the soft system dynamics, do structure-based models accurately recover ground-truth connectivity of the robotic non-rigid kinematic chain?

### 6.2.1 Dynamics modelling

In order to answer the first question in Section 6.2, each model was trained to observe sequences of a soft gripper motion and asked to predict further in time.

For this first experiment, the structure-based approach (NRI) is compared to four different baselines: Linear, MLP, LSTM and supervised NRI. The NRI model has two model variations on the decoder part of the model as it can either be based on a MLP (NRI-mlp) or an RNN structure (NRI-rnn). In this experiment, we consider both variants for both the unsupervised and the supervised approach. The supervised encoder receives ground truth edges for the connections between the finger nodes whereas the unsupervised learns the connections in an end-to-end manner while learning to predict future trajectories.

**Table 6.1:** Mean Squared Error for the NRI model and baselines, with *Testset 0*

| MODEL | | | $MSE\ 5$ | $MSE\ 10$ | $MSE_{dist}\ 10$ |
|---|---|---|---|---|---|
| baselines | non-strucutured | LINEAR | 7.987x10$^{-4}$ | 1.268x10$^{-3}$ | 1.839x10$^{-2}$ |
| | | MLP | 7.878x10$^{-4}$ | 1.117x10$^{-3}$ | 1.698x10$^{-2}$ |
| | | LSTM | 6.157x10$^{-4}$ | 1.354x10$^{-3}$ | 1.964x10$^{-2}$ |
| | supervised | NRI-mlp | 4.753x10$^{-4}$ | 1.281x10$^{-3}$ | 1.846x10$^{-2}$ |
| | | NRI-rnn | **2.108x10**$^{-4}$ | **7.457x10**$^{-4}$ | **1.063x10**$^{-2}$ |
| SMG | unsupervised | NRI-mlp | 4.962x10$^{-4}$ | 1.336x10$^{-3}$ | 1.926x10$^{-2}$ |
| | | NRI-rnn | **2.200x10**$^{-4}$ | **8.013x10**$^{-4}$ | **1.142x10**$^{-2}$ |

In this experiment, all models are trained for 10 prediction steps. In the variations of the NRI with no ground truth edge information, a non-overlapping set of 50 time steps is provided to the encoder to estimate the connectivity graph. For all the baselines and for the NRI-rnn, there is an initialization or *burn-in* phase where the models have access to the 5 time steps before the sequence that is considered for the evaluation. The LSTM *burn-in* phase, similarly to the NRI, has 50 steps before the 10 step prediction window. This allows the LSTM to initialize its internal state.

Table 6.1 shows that the proposed model outperforms the different baseline models in the first test set, *Testset 0*, which has a diverse set of motions, configurations and elasticities. The values of the $MSE_{dist}$ show all baselines reach the 10th step with a prediction deviation that is, on average, smaller than 2% of the travelled distance, which gives some confidence about the performance of every strategy. However, we can see that the proposed model (NRI) achieves lower average error compared to the baseline models in *Testset 0* and when the prediction step equals the training step. Within the two variations of the NRI model, we find that the best combination is the unsupervised NRI-decoder variant (NRI-rnn) only (slightly) outperformed by the supervised implementation (supervised NRI-rnn). In the case of the NRI-mlp, this architectural variation seems to outperform non-structured baselines for the first time steps, but it quickly degenerates and reaches the 10th step with similar performance. We will look more closely at how this unsupervised NRI-rnn model performs under different conditions in a subsequent experiment.

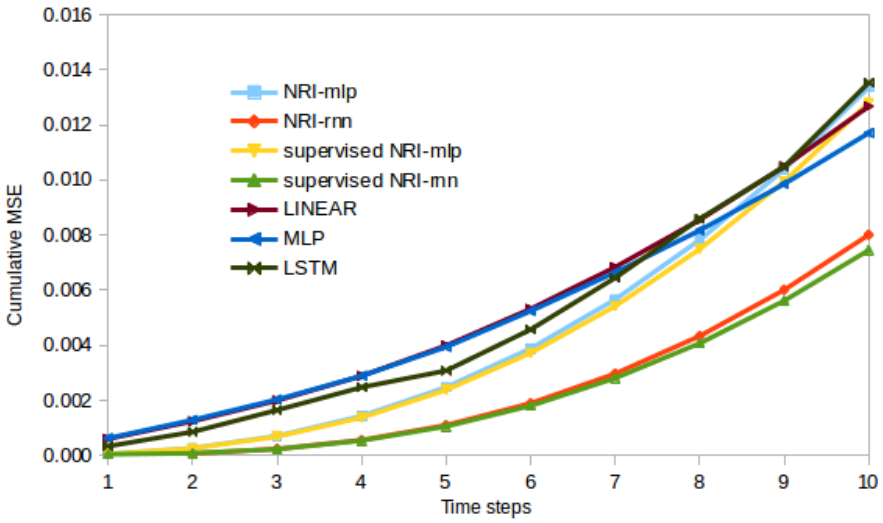



**Figure 6.3:** Cumulative mean squared error (MSE) over 10 time steps for different models

Figure 6.3 shows how the models compare when predicting for 10 steps ahead, the same number they were trained on. This validation set includes small variations in motion, elasticity and configurations. We can see that the NRI model using an RNN decoder outperforms the other variations considerably. When compared to the NRI-mlp poor performance, this difference may be due to: 1) differences in training: in this case, the absent burn-in phase in the NRI-mlp places this model at a disadvantage 2) structural differences between the two models in which the one with a Markovian assumption behaves worse. We notice that while the NRI-rnn has better performance for the first steps, particularly relevant in planning, the accumulated error starts deteriorating fast as we predict

beyond the training distribution, further into the future. We will look at this more closely in the following experiments.

### 6.2.2 Generalization

After evaluating how well the different approaches model the non-rigid kinematic chain and predict future dynamics, we now wish to disentangle the multiple sources of variability and establish how the model generalizes for different input distribution shifts. Moreover, with this ablation study, we propose to answer the second question in Section 6.2 of whether graph-based approaches are more robust to generalization than the non-structured baselines.

In order to test the models under a diverse set of conditions, we shall consider *Testset 0* under a prediction step of 25 to test how well the model reacts to auto-regressively predicting more steps ahead than it was trained on and the four *Testsets* introduced in the experimental setup (refer to Table 5.2 for an overview of the different *Testsets*). This enables us to study the effects of the different sources of prediction error separately. For this ablation study, we will consider the best performing NRI model and two relevant baselines, LSTM and MLP.

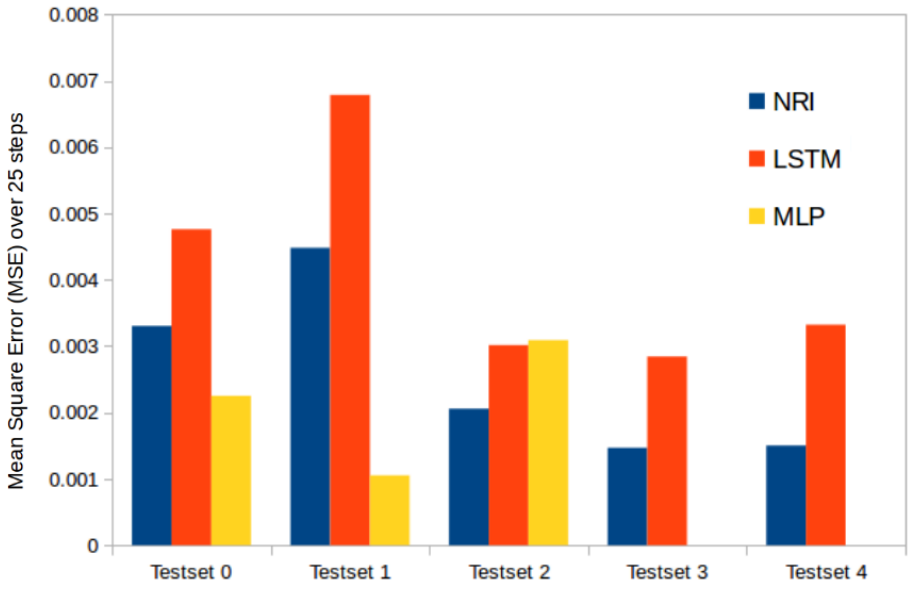



**Figure 6.4:** Mean squared error (MSE) after the 25 time steps for different models and validation sets

From the four ablation studies (Figure 6.4), different conclusions can be drawn but the robustness of the NRI model becomes evident. The first validation set shows that the NRI performs worse than the baseline models when generalizing to the distribution shift introduced by predicting longer sequences. In *Testset 0* and *Testset 1* , the MLP seems to be capable of predicting longer sequences with a lower mean error and proves to have good generalization to new motions, respectively. This advantage of the MLP compared to the structure-based approaches can be justified with the fact that it is able to exploit the relations between the nodes in adjacent positions of the input vector. The NRI, due to its order invariant graph-based structure, only has access to the nodes without explicit order - like in a vector-based representation of the nodes.

This, however, makes the MLP quite fragile to changes or disturbances in the data, as we see from *Testsets 2-4*. Testing the same model in a situation with new relative positions of the fingers, the MLP's prediction error increases considerably while the NRI model outperforms the baseline models. When changing the number of fingers, the MLP cannot deal with the zero-padding, is not able to generalize and has a significantly higher error than the other models, hence it is not considered in this validation set. Once more, we can see the NRI outperforming the baseline models in this condition. The last *Testset* considers the situation where points are shuffled and the points at validation are in a different order than the configurations it was trained on. This models a tracking or re-identification error. Here, once again, we notice that, due to its order invariance properties, the NRI model outperforms the baseline models.

### 6.2.3 Connectivity Inference

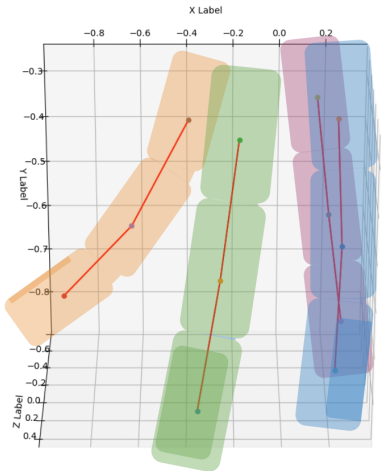
Finally, we answer the third and last question of this section: while modelling the soft system dynamics, do structure-based models accurately recover ground truth (g.t.) connectivity of the robotic non-rigid kinematic chain? In order to do this, we now focus on the performance of our proposed model at inferring the connectivity graph. Since this structure will be used by the decoder to predict the dynamical behaviour, we expect it to have great impact on the model's overall performance we have seen before.

In order to evaluate the connectivity inference process, we select the best performing NRI strategy, the NRI-rnn and compare its supervised and unsupervised strategies in terms of their F1-score at predicting the system relational graph. We use the same setup as in our baseline experiment: *Testset0* at test time and prediction step of 10. The accuracy, F1-score and MSE for both situations are depicted in Table 6.2 as well as three examples of the three-dimensional space connectivity graph for those approaches at training time - Figure 6.5.

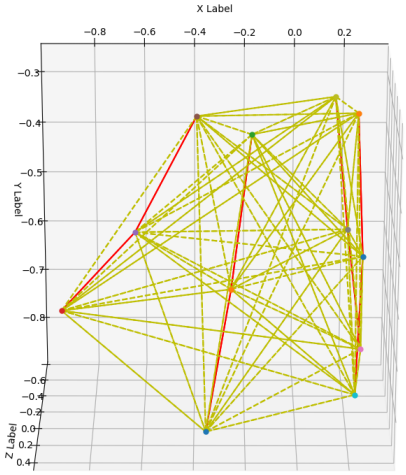
**Table 6.2:** Connectivity inference performance

| **NRI-rnn** | | | |
|---|---|---|---|
| **training** | **MSE10** | **accuracy** | **F1score** |
| supervised | $7.499\text{x}10^{-4}$ | 0.998 | 0.991 |
| unsupervised | $8.062\text{x}10^{-4}$ | 0.486 | 0.176 |

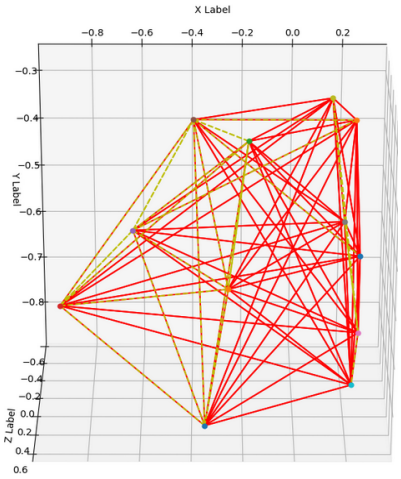
Simply put, we can immediately state that: 1) connectivity inference contributes positively to the dynamics prediction; 2) supervised training of the encoder allows to recover g.t. connectivity; 3) unsupervised training does not seem to recover g.t. connectivity. The predicted graph helps to increase the performance of the NRI model, with the encoder trying to optimize the graph structure to help the decoder predicting future dynamics. Table 6.2 shows that, by looking at the dynamics prediction error, the unsupervised encoder effort to achieve an expressive connectivity graph is not as effective as the supervised approach. Moreover, the supervised encoder can in fact recover g.t. connectivity with almost perfect accuracy and F1-score. We notice that the same does not happen in the unsupervised approach, where the accuracy is usually around 50% and the F1score is even lower. These differences in the results of supervised and unsupervised training can be explained by two phenomena.



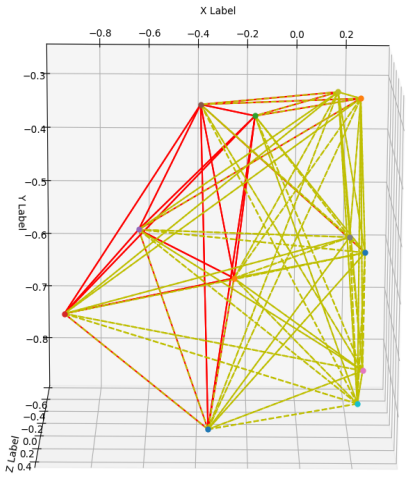

**(a)** Ground truth configuration of the 4 fingers




**(b)** Supervised training (ground truth connectivity)




**(c)** Unsupervised training (case 1)




**(d)** Unsupervised training (case 2)

**Figure 6.5:** Connectivity graph (12 nodes with positive (red) or null (yellow) links) for supervised and unsupervised training

A first interesting aspect shown in Figure 6.5 is that the unsupervised approach does not see any labels for the edge types and might arrive at a similar, but opposite, correspondence. In Figure 6.5(b) we can see binary edge types: red for an existing connection and yellow for a non-existing physical connection. In Figure 6.5(c) we can see the opposite: most of the edges are marked as red (positive), while a few edges near the four fingers are yellow (null). Although the accuracy and F1-score of such prediction will yield very poor results - in this case, the unsupervised encoder is actually retrieving very "similar" results to the supervised approach.

The second interesting aspect is that the graph connectivity that the model arrives to is often surprisingly different from the supervised g.t. structure (see Figure 6.5(d) compared to 6.5(c)). In fact, for other experiments with different parameters, it was even the case of similar apparently-arbitrary structures leading to better performances than the supervised version. Although finger-to-finger connections may sound counter-intuitive and difer from the "natural" physical structure, this phenomenon

was already pointed out in [149] and it is comparable to the results found in leading approaches for modelling motion capture data that also include hand-to-hand interactions [204].

In conclusion, this model's end-goal is to learn the dynamics of a system and predict its near-future evolution for control applications which, by the results in Sections 6.2.1 and 6.2.2, it accomplishes successfully. For this, it creates an internal connectivity representation to help inferring the kinematics of such system and hence inferring its dynamics. Even though this connectivity may vary from the true physical structure, this does not affect its performance at mapping actions to trajectories and would not harm the inversion of this map for control application. If, for a certain deployment, it is necessary that the model infers g.t. connectivity, some additional relational bias must be provided to the network or it can be trained in a supervised way. However, since we usually only care about an accurate dynamics prediction, we optimize the inferred connectivity to help reaching this goal - whether it is "equal but opposite" or it includes finger-to-finger links.

## 6.3 Hyperparameters

Finally, we shall test the effect of some hyperparameters (described in Section 4.3) in the performance our model. In particular, the effect of the prediction step ($ps$) and of the number of edge types ($d$) in the dynamics and connectivity inference, respectively.

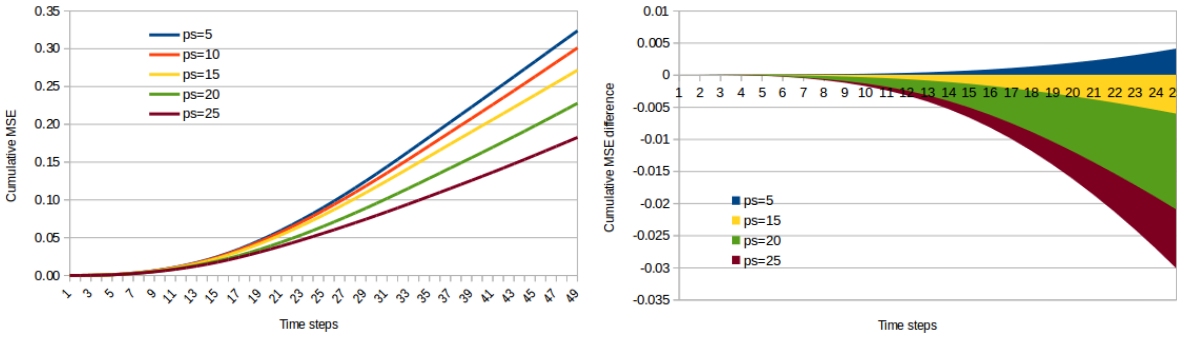
### 6.3.1 Prediction step

One of the parameters with the greatest meaning and influence in the model is the prediction window. We have seen in Section 6.2 that the NRI model, when trained to predict 10 time steps, had its performance outracing the other baselines for the first time steps but suffered quick deterioration after transgressing the training prediction window.

Since, depending on the deployment, shorter or longer time windows can be required, we now test the winning unsupervised structured approach, the NRI-rnn, to see how its performance is affected when training for by a different prediction step. The conditions are kept the same, with the model being trained on *Trainset* for different prediction steps (ps=5,10,15,25), as described in 6.3.1, and tested on *Testset 0*. The results for the variation of this hyperparameter can be seen in Figure 6.6.

As expected, training the model for larger time sequences leads to a better performance predicting those. After 25 time steps, the model that was trained with ps=25 has the smallest accumulated error and the one that was trained with ps=5 has the highest. Furthermore, it is also noticeable that increasing the prediction step does not compromise initially accurate results: larger windows benefit predictions right from the start.

However, some considerations should be added when talking about training the model for larger prediction steps. Not only is this more time- and computationally-consuming, it should require more epochs and network hidden layers to properly capture the increasingly more complex dynamics and not to underfit. Moreover, for a short prediction window, the advantage of large-prediction-step over short-prediction-step-training should be insignificant for most applications and might not pay off the




**(a)** Cumulative MSE for different prediction steps

**(b)** Variation of prediction step compared to the baseline value of 10

**Figure 6.6:** Effect of the prediction step in the performance of the NRI-rnn model

trouble. It is the case, for example, of optimal control applications - which usually don't require too large prediction windows.

### 6.3.2 Types of Edges

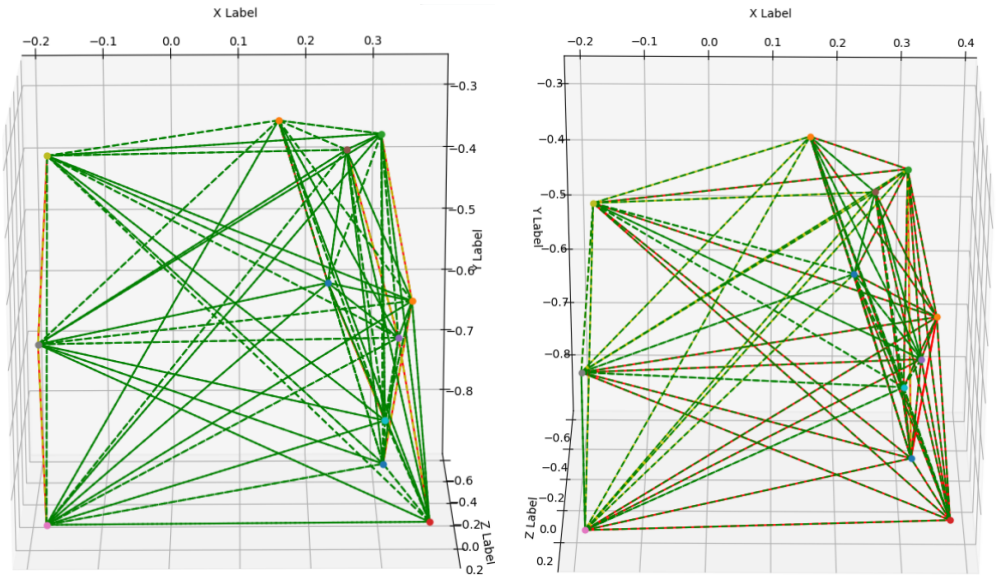
Finally, we try varying the number of edge types $d$ in the encoder connectivity inference to see its effect on the dynamics prediction. As explained in 5.3.3, in the previous experiments we have assumed binary relations between nodes - positive if the nodes were connected and adjacent, negative if not. However, we could assume more types of edges, different ways in which the different nodes relate with each other. In this case, we compare the outputs of the encoder and decoder, for $d$=2 and $d$=3. Table 6.3 depicts the results.

**Table 6.3:** Effect of varying the number of types of edges in the performance of the decoder

| NRI-rnn | | |
|---|---|---|
| **d** | **training** | $\mathbf{MSE_{10}}$ |
| 2 | supervised | 8.045x10$^{-4}$ |
| | unsupervised | 8.206x10$^{-4}$ |
| 3 | supervised | 7.588x10$^{-4}$ |
| | unsupervised | 8.083x10$^{-4}$ |

We explained in 4.2.2 and 5.3.3 how increasing the number of edges (each type with a separate MLP) should lead to a more explicit dependency on the edge type. Moreover, this should enrich the model ability to encode valuable relational information in the connectivity graph that is used by the decoder to infer future dynamics. For these reasons, it was expected that increasing edge diversity would, in theory, improve the consequent dynamics prediction results.

By these results, we notice that increasing the number of edge types $d$ from 2 to 3, is enough to have a positive effect in the unsupervised prediction, with the MSE of the first 10 steps (same as training prediction step) with $d$=3 being slightly smaller (around 1.5%) than with $d$=2. Although this difference is not much significant, we can see that this gap widens for supervised approaches. When providing the decoder with the g.t. connectivity graph, we notice that, for $d$=3, the MSE is also smaller (around 5.7%) than for $d$=2.




**(a)** Supervised training **(b)** Unsupervised training

**Figure 6.7:** Connectivity graph for d=3

For both values of $d$, the supervised variant outperforms the unsupervised. In Figure 6.7 we can see the contrast between the supervised and the unsupervised version for $d$=3: in supervised training, the encoder output is forced to the g.t. connectivity (red for adjacent descending joints, yellow for adjacent ascending joints, green for non-adjacent nodes), while in the unsupervised variant, the connectivity graph includes green, red and yellow finger-to-finger connections.

## 6.4 Conclusion

In this chapter, we focused on detailing and interpreting the results of the experiments formerly described. Our research questions and respective conclusions can be divided into five moments:

- Structure-based models do outperform non-structure-based approaches for soft robotics dynamics modelling and for predicting time windows of pre-defined length.
- Moreover, structure-based approaches proved more robust to generalization than non-structure-based approaches regarding configurational variations or tracking errors.
- Structure-based models can be trained in both supervised (with ground truth connectivity) and unsupervised (without ground truth connectivity) fashions. With unsupervised training, the inferred internal connectivity graph might vary from the ground-truth kinematic chain but still achieves similar dynamics prediction results.
- Training the structure-based model for longer (reasonable) predictions improves its prediction performance for all time in that interval
- Increasing the number of expected edge types in the connectivity graph from 2 to 3 improves the graph expressiveness and therefore the prediction results

# 7

# Conclusions

## Contents

In this work, we proposed to learn the graph-based action-conditioned relational forward model of the non-rigid kinematic chain of a robotic soft gripper. More specifically: we proposed to adapt a graph-based relational inference network (NRI) to integrate constraint actions; we generated a simulation environment to generate data regarding the robotic soft hand with varying configurations; and we benchmarked the proposed structure-based model against different implementations of non-structured baselines to compare performance and robustness.

As the work in this thesis lies in the intersection of different topics, we started by providing an overview of the necessary pieces to formulate the problem statement and our solution. We peeked into the psychological perspective of cognitive development and sensorimotor learning, after which we dived into the specific domain of soft robotics and the end goal of optimal control integration. Next, we emerged on the exciting potentials of graph representations in relational systems and the recent breakthroughs of Graph Neural Networks.

After outlining the challenge that motivates this work, we proposed the Sensorimotor Graph model to tackle these issues. It was introduced as a sensorimotor-based framework which inferred relational knowledge of the system (in the form of GNNs) to achieve better results at modelling and predicting dynamics. We adapted recent work on graph-based relational inference [149] to build our model and prepared different non-structured baseline models to assess our claim.

Finally, after generating simulation motions of a robotic soft gripper with varying configurations, we trained our unsupervised model and compared it with the different baselines to assess performance and robustness to increasing adversity.

We showed that the Sensorimotor Graph, even if detecting unusual latent connectivity, not only outperform non-structured strategies when predicting future dynamics of our non-rigid kinematic chain but also reveal significant robustness to configurational variations, tracking errors or node failures.

In other words, we showed that Graph Neural Networks, more specifically the Neural Relational Inference model, can be used to robustly model a system of interacting soft material parts in the form of a differentiable dynamics model, which opens up promising avenues for combining the learnt model with a non-convex optimal control framework, using the differentiable dynamics model as systems constraints.

Our approach extends the state-of-the-art in directions that are key to improving soft robotics modelling and actuation, and hence to contributing to soft robotics universalization.

## 7.1 Future Work

We believe that this work showed encouraging results and generalization capabilities of our proposed Sensorimotor Graph model, paving the way to the ability to acquire the structure and dynamics of complex systems, such as soft robotic limbs and will be of utmost importance in future work:

1. adapting the model to always recover ground-truth connectivity with unsupervised training. This can be done by including relational biases like a sparsity prior, or by including motor synergies during sensorimotor learning to progressively unlock degrees of freedom;

2. combining the learnt model with a non-convex optimal control framework, using the differentiable dynamics model as systems;

3. deployment in complex, non-rigid kinematic chains and articulated end-effectors such as soft hands for advanced manipulation and online control, using additional instrumentation in the form of markers, or learning to extract and identify key-point representations in an unsupervised manner;

4. extend this model to handle contact with external objects. Apart from the observation data and the actuation signal, the external forces coming from contact or pressure sensors could be included in the model input to help modelling not only the end-effector body schema but also its relation with the object being manipulated.

## 7.2 Material Contributions

The work in this thesis has expanded the understanding of GNNs by applying them to soft robotics (kinematics chains), making different important contributions in both fields. We highlight two key material contributions that have sprung from the development of this research thesis:

- **ICRA submission**: A research paper with the summarized work of this thesis was submitted to the International Conference on Robotics and Automation 2020 under the name "SENSORIMOTOR GRAPH: Action-Conditioned Relational Forward Model Learning of Robotic Soft Hand Dynamics"

- **code repository**: The necessary code for this work is available in [205] for future use. This includes the data extraction from SOFA, the data preparation file, the modified NRI model (adapted from [206]) and all four baseline models.

# Bibliography


[1] C. Majidi, “Soft robotics: a perspective - current trends and prospects for the future,” *Soft Robot*, vol. 1, pp. 5–11, 2014.

[2] G. Bao, H. Fang, L. Chen, Y. Wan, F. Xu, Q. Yang, and L. Zhang, “Soft robotics: Academic insights and perspectives through bibliometric analysis,” *Soft Robotics*, pp. 229–241, 05 2018.

[3] Y. Ansari, T. Hassan, M. Manti, E. Falotico, M. Cianchetti, and C. Laschi, “Soft robotic technologies for industrial applications,” pp. 35–64, 01 2015.

[4] W. Huitt and J. Hummel, “Piaget’s theory of cognitive development,” *Educational Psychology Interactive*, 2003.

[5] J. Piaget, “Cognitive development in children: development and learning,” *journal of research in science teaching*, no. 2, pp. 176–186, 1964.

[6] R. Saegusa, G. Metta, G. Sandini, and S. Sakka, “Active motor babbling for sensory-motor learning,” *2008 ROBIO IEEE International Conference on Robotics and Biomimetics*, pp. 794 – 799, 03 2009.

[7] K. Capek, *R.U.R. (Rossum’s Universal Robots)*. New York: Penguin Books, 2004.

[8] R. Miall and D. Wolpert, “Forward models for physiological motor control,” *Neural Networks*, vol. 9, no. 8, pp. 1265 – 1279, 1996, four Major Hypotheses in Neuroscience.

[9] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed. Cambridge, MA, USA: MIT Press, 1995.

[10] L. Bertalanffy, *General System Theory: Foundations, Development, Applications*. New York: George Braziller, 1968.

[11] C. Kemp and J. B. Tenenbaum, “The discovery of structural form,” *Proceedings of the National Academy of Sciences*, vol. 105, no. 31, pp. 10 687–10 692, 2008.

[12] R. J. Wilson, *Introduction to Graph Theory*. USA: John Wiley  Sons, Inc., 1986.

[13] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, Jan 2009a.

[14] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, “Interaction networks for learning about objects, relations and physics,” *Advances in Neural Information Processing Systems*, pp. 4502–4510, 2016.

[15] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” *ICML*, 2017.

[16] R. de Kleijn, G. Kachergis, and B. Hommel, *Cognitive Robotics*. CRC Press, 2015, vol. 1.

[17] G. Alici, “Softer is harder: What differentiates soft robotics from hard robotics?” *MRS Advances*, vol. 3, pp. 1–12, 02 2018.

[18] F. Schmitt, O. Piccin, L. Barbe, and B. Bayle, “Soft robots manufacturing: A review,” *Frontiers Robotics AI*, vol. 5, 07 2018.

[19] D. M. Buss, *Evolutionary Psychology: The New Science of the Mind*. Taylor and Francis ltd, 2015.

[20] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, “Developmental robotics: A survey,” *Connect. Sci.*, vol. 15, pp. 151–190, 12 2003.

[21] J. Piaget, *The origins of intelligence in children*, I. U. Press, Ed., New York, 1952.

[22] J. Piaget, *Epistemology and psychology of functions*, Dordrecht, Netherlands: D. Reidel Publishing Company, 1977.

[23] J. Piaget, H. E. Gruber, and J. J. Voneche, *The essential Piaget*, New York, NY: Basic Books, 1977.

[24] J. W. Santrock, *A Topical Approach to Life-Span Development*, New York, NY: McGraw-Hill, 2008.

[25] C. von Hofsten, “Eye-hand coordination in newborns,” *Developmental Psycology*, no. 18, p. 450â461, 1982.

[26] N. Wada, A. Nakata, T. Koga, and M. Tokuriki, “Anatomical structure and action of the tail muscles in the cat,” *The Journal of veterinary medical science*, vol. 56, no. 6, p. 1107â1112, December 1994.

[27] Z. Schonbrun, *The performance cortex: How neuroscience is redefining athletic genius*, New York: Dutton Books, 2018.

[28] A. Clark, “Natural-born cyborgs?” in *Cognitive Technology: Instruments of Mind*, M. Beynon, C. L. Nehaniv, and K. Dautenhahn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 17–24.

[29] K. Schauwers, S. Gillis, and P. J. Govaerts, “The characteristics of prelexical babbling after cochlear implantation between 5 and 20 months of age,” *Ear Hear*, pp. 627–37, 08 2008.

[30] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. MIT Press, 01 2015.

[31] M. Lopes and J. Santos-Victor, "A developmental roadmap for learning by imitation in robots," *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 37, pp. 308–21, 05 2007.

[32] atsuya Aoki, T. Nakamura, and T. Nagai, "Learning of motor control from motor babbling," *IFAC-PapersOnLine*, vol. 49, no. 19, pp. 154 – 158, 2016, 13th IFAC Symposium on Analysis, Design, and Evaluation ofHuman-Machine Systems HMS 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896316320687

[33] D. McFarland, K. Stenning, and M. McGonigle, *The Complex Mind: An Interdisciplinary Approach*, Palgrave Macmillan, London, 2012.

[34] A. Marjaninejad, D. Urbina-Melendez, B. A. Cohn, and F. J. Valero-Cuevas, "Autonomous functional movements in a tendon-driven limb via limited experience," *Nature Machine Intelligence*, vol. 1, pp. 144–154, 03 2019.

[35] Y. Demiris and A. Dearden, "From motor babbling to hierarchical learning by imitation: a robot developmental pathway," *In Proc. of the 5th International Workshop on Epigenetic Robotics*, 2005.

[36] E. Oztop and M. Arbib, "A biologically inspired learning to grasp system," *Medicine, E. and Society, B.*, 2001.

[37] E. Oztop, N. S. Bradley, and M. Arbib, "Infant grasp learning: a computational model," *Exp Brain Res*, 2004.

[38] K. Takahashi, K. Kim, T. Ogata, and S. Sugano, "Tool-body assimilation model considering grasping motion through deep learning," *Robotics and Autonomous Systems*, vol. 91, pp. 115 – 127, 2017.

[39] M. A. Goodrich and A. C. Schultz, *Human-robot Interaction: A survey*. now Publicher Inc., 2007.

[40] A. Ramos, E. Bastos, and K. Kim, *Perspectives of Robotic Bariatric Surgery*, 08 2015.

[41] S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: A bioinspired evolution in robotics," *Trends in biotechnology*, vol. 31, 04 2013.

[42] R. M. May, "How many species are there on earth?" *Science*, vol. 247, pp. 1441–49, 1988.

[43] *Pixabay - free image bank*, (accessed December, 2020). [Online]. Available: https://pixabay.com/

[44] C. Laschi and M. Cianchetti, "Soft robotics: New perspectives for robot bodyware and control," *Front Bioeng Biotechnol.*, 01 2014.

[45] C. Lee, M. Kim, Y. Kim, N. Hong, S. Ryu, and S. Kim, “Soft robot review,” *International Journal of Control, Automation and Systems*, vol. 15, 01 2017.

[46] H. Wang, R. Zhang, W. Chen, X. Wang, and R. Pfeifer, “A cable-driven soft robot surgical system for cardiothoracic endoscopic surgery: preclinical tests in animals,” *Surgical Endoscopy*, vol. 31, pp. 3152–3158, 2016.

[47] H. HL, “Robotic glove with soft-elastic composite actuators for assisting activities of daily living,” *Soft Robotics*, 2019.

[48] H.-T. Lin, G. Leisk, and B. Trimmer, “Goqbot: A caterpillar-inspired soft-bodied rolling robot,” *Bioinspiration biomimetics*, vol. 6, p. 026007, 06 2011.

[49] C. Aubin, S. Choudhury, R. Jerch, L. Archer, J. Pikul, and R. Shepherd, “Electrolytic vascular systems for energy-dense robots,” *Nature*, vol. 571, 07 2019.

[50] R. Martinez, A. Glavan, C. Keplinger, A. Oyetibo, and G. Whitesides, “Soft actuators and robots that are resistant to mechanical damage,” *Advanced Functional Materials*, 12 2013.

[51] T. Duckett, S. Pearson, S. Blackmore, and B. Grieve, “Agricultural robotics: The future of robotic agriculture,” *CoRR*, vol. abs/1806.06762, 2018.

[52] G. Chowdhary, M. Gazzola, G. Krishnan, and S. Soman, C.; Lovell, “Soft robotics as an enabling technology for agroforestry practice and research,” *Sustainability*, 11 2019.

[53] A. Jiang, E. Secco, H. Wurdemann, T. Nanayakkara, P. Dasgupta, and K. Athoefer, “Stiffness-controllable octopus-like robot arm for minimally invasive surgery,” *Workshop on New Technologies for Computer/Robot Assisted Surgery*, 2013.

[54] M. Cianchetti, T. Ranzani, G. Gerboni, T. Nanayakkara, K. Althoefer, P. Dasgupta, and A. Menciassi, “Soft robotics technologies to address shortcomings in todayâs minimally invasive surgery: the stiff-flop approach,” *Soft Robotics*, vol. 1, pp. 122–131, 2014.

[55] S. Sareh, A. Jiang, A. Faragasso, Y. Noh, T. Nanayakkara, P. Dasgupta, L. D. Seneviratne, H. A. Wurdemann, and K. Althoefer, “Bio-inspired tactile sensor sleeve for surgical soft manipulators,” 2014, pp. 1454–1459.

[56] T. Deng, H. Wang, W. Chen, X. Wang, and R. Pfeifer, “Development of a new cable-driven soft robot for cardiac ablation,” *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 728–733, 2013.

[57] P. Maeder-York, T. Clites, E. Boggs, R. Neff, P. Polygerinos, D. Holland, L. Stirling, K. Galloway, C. Wee, and C. Walsh, “Biologically inspired soft robot for thumb reha- bilitation,” *Journal of Medical Devices*, vol. 8, 2014.

[58] P. Polygerinos, K. C. Galloway, E. Savage, M. Herman, K. O. Donnell, and C. J. Walsh, “Soft robotic glove for hand rehabilitation and task specific training,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2913–2919, 2015.

[59] H. K. Yap, L. J. Hoon, F. Nasrallah, J. C. H. Goh, and R. C. H. Yeow, “A soft exoskeleton for hand assistive and rehabilitation application using pneumatic actuators with variable stiffness,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4967–4972, 2015.

[60] A. T. Asbeck, R. J. Dyer, A. F. Larusson, and C. J. Walsh, “Biologically-inspired soft exosuit,” *IEEE International Conference on Rehabilitation Robotics (ICORR)*, pp. 1–8, 2013.

[61] J. A. Gallego, E. Rocon, J. Ib, J. L. Dideriksen, A. D. Koutsou, R. Paradiso, M. B. Popovic, J. M. Belda-Lois, F. Gianfelici, D. Farina, D. B. Popovic, M. Manto, T. D. Alessio, and J. L. Pons, “A soft wearable robot for tremor assessment and suppression,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2249–2254, 2011.

[62] Y. S. Song, Y. Sun, R. van den Brand, J. Zitzewitz, S. Micera, G. Courtine, and J. Paik, “Soft robot for gait rehabilitation of spinalized rodents,” 11 2013, pp. 971–976.

[63] W. Wei, L. Jang-Yeob, R. Hugo, S. Sung-Hyuk, C. Won-Shik, and A. Sung-Hoon, “Locomotion of inchworm-inspired robot made of smart soft composite (ssc),” *Bioinspiration  Biomimetics*, vol. 9, 2014.

[64] S. Seok, C. D. Onal, K. J. Cho, D. R. R. J. Wood, and S. Kim, “Meshworm: A peristaltic soft robot with an- tagonistic nickel titanium coil actuators,” *IEEE/ASME Transactions on Mechatronics*, vol. 18, pp. 1485–1497, 2013.

[65] M. T. Tolley, R. F. Shepherd, M. Karpelson, N. W. Bartlett, K. C. Galloway, M. Wehner, R. Nunes, G. M. Whitesides, and R. J. Wood, “An untethered jumping soft robot,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 561–566, 2014.

[66] R. F. Shepherd, A. A. Stokes, J. Freake, J. Barber, P. W. Snyder, A. D. Mazzeo, L. Cademartiri, S. A. Morin, and G. M. Whitesides, “Using explosions to power a soft robot,” *Angewandte Chemie International Edition*, vol. 52, pp. 2892–2896, 2013.

[67] Y. Sugiyama and S. Hirai, “Crawling and jumping by a deformable robot,” *The International Journal of Robotics Research*, vol. 25, pp. 603–620, 2006.

[68] F. Giorgio Serchi, F. Renda, C. Laschi, and F. Boyer, “Locomotion and elastodynamics model of an underwater shell-like soft robot,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, 05 2015.

[69] F. Giorgio Serchi and G. Weymouth, “Underwater soft robotics, the benefit of body-shape variations in aquatic propulsion,” *Biosystems and Biorobotics*, vol. 17, pp. 37–46, 09 2017.

[70] R. Katzschmann, A. Marchese, and D. Rus, “Hydraulic autonomous soft robotic fish for 3d swimming,” *Experimental Robotics: Springer Tracts in Advanced Robotics*, vol. 109, 2016.

[71] M. Calisti, A. Arienti, F. Renda, G. Levy, B. Hochner, B. Mazzolai, P. Dario, and C. Laschi, “Design and develop- ment of a soft robot with crawling and grasping capabili- ties,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4950–4955, 2012.

[72] M. Cianchetti, M. Calisti, L. Margheri, M. Kuba, and C. Laschi, “Bioinspired locomotion and grasping in wa- ter: the soft eight-arm octopus robot,” *Bioinspiration Biomimetics*, vol. 10, 2015.

[73] J. Shintake, V. Cacucciolo, D. Floreano, and H. Shea, “Soft robotic grippers,” *Advanced Materials*, vol. 30, p. 1707035, 05 2018.

[74] G. Gu, J. Zou, R. Zhao, X. Zhao, and X. Zhu, “Soft wall-climbing robots,” *Science Robotics*, vol. 3, p. eaat2874, 12 2018.

[75] C. Li, G. C. Lau, H. Yuan, A. Aggarwal, V. L. Dominguez, S. Liu, H. Sai, L. C. Palmer, N. A. Sather, T. J. Pearson, D. E. Freedman, P. K. Amiri, M. O. de la Cruz, and S. I. Stupp, “Fast and programmable locomotion of hydrogel-metal hybrids under light and magnetic fields,” *Science Robotics*, vol. 5, no. 49, 2020.

[76] T. Arnold and M. Scheutz, “The tactile ethics of soft robotics: Designing wisely for humanârobot interaction,” *Soft Robotics*, vol. 4, 05 2017.

[77] B. Phillips, K. Becker, and S. e. a. Kurumaya, “A dexterous, glove-based teleoperable low-power soft robotic arm for delicate deep-sea biological exploration,” *Sci Rep*, vol. 8, 2018.

[78] *Soft Robotics Inc.*, Accessed: December 2020. [Online]. Available: https://www.softroboticsinc.com/

[79] Y. Tang, Y. Chi, J. Sun, T.-H. Huang, O. H. Maghsoudi, A. Spence, J. Zhao, H. Su, and J. Yin, “Leveraging elastic instabilities for amplified performance: Spine-inspired high-speed and high-force soft robots,” *Science Advances*, vol. 6, 05 2020.

[80] J. Amend, N. Cheng, S. Fakhouri, and B. Culley, “Soft robotics commercialization: Jamming grippers from research to product,” *Soft Robot*, pp. 213–222, 12 2016.

[81] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, “Differentiable mpc for end-to-end planning and control,” in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 8289–8300. [Online]. Available: http://papers.nips.cc/paper/8050-differentiable-mpc-for-end-to-end-planning-and-control.pdf

[82] J. L. Pons, R. Ceres, and F. Pfeiffer, “Multifingered dextrous robotics hand design and control: A review,” *Robotica*, vol. 17, no. 6, p. 661â674, Nov. 1999.

[83] T. Hsia, “Adaptive control of robot manipulators - a review,” in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, 1986, pp. 183–189.

[84] F. Ebert, C. Finn, A. X. Lee, and S. Levine, “Self-supervised visual planning with temporal skip connections,” in *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, ser. Proceedings of Machine Learning Research, vol. 78. PMLR, 2017, pp. 344–356.

[85] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, “Learning latent dynamics for planning from pixels,” in *ICML*, 2019.

[86] J. Kober, J. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 09 2013.

[87] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, “Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids,” in *ICLR*, 2019.

[88] S. Privara, J. Cigler, Z. Vana, F. Oldewurtel, C. Sagerschnig, and E. Zacekova, “Building modeling as a crucial part for building predictive control,” *Energy and Buildings*, vol. 56, p. 8â22, 01 2013.

[89] P. Hyatt, D. Wingate, and M. Killpack, “Model-based control of soft actuators using learned non-linear discrete-time models,” *Frontiers in Robotics and AI*, vol. 6, 04 2019.

[90] D. Bruder, B. Gillespie, C. Remy, and R. Vasudevan, “Modeling and control of soft robots using the koopman operator and model predictive control,” *2019 Robotics: Science and Systems*, 02 2019.

[91] Kipf, T. and Fetaya, E. and Wang, K. C. and Welling, M. and Zemel, R., “Neural relational inference for interacting systems,” *International Conference on Machine Learning*, 2018.

[92] P. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gülcehre, H. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, “Relational inductive biases, deep learning, and graph networks,” *CoRR*, 06 2018.

[93] Alvaro Sanchez-Gonzalez and Jonathan Godwin and Tobias Pfaff and Rex Ying and Jure Leskovec and Peter W. Battaglia, “Learning to simulate complex physics with graph networks,” *International Conference on Machine Learning*, 02 2020.

[94] G. Finlayson, E. Tj, and M. Drew, “4-sensor camera calibration for image representation invariant to shading, shadows, lighting, and specularities,” *International Conference on COmputer Vision*, 07 2001.

[95] G. Perez, “On the essence and ontology of systems,” *Open Science Journal*, vol. 5, 08 2020.

[96] J. Miller, *Living Systems*. McGraw Hill Inc., 1978.

[97] N. Cennamo, S. Di Giovanni, A. Varriale, M. Staiano, F. Di Pietrantonio, A. Notargiacomo, L. Zeni, and S. D'Auria, "Easy to use plastic optical fiber-based biosensor for detection of butanal," *PloS one*, vol. 10, p. e0116770, 03 2015.

[98] M. Del Sorbo, M. Ibarzabal, G. Lee, M. Chen, and C. Plancarte, "Doosan project final version (doosan babcock energy)," 04 2007.

[99] J. Jungwirth, J. Casper, and A. Baumhauer, "Particular design features for a long span cable-stayed bridge over the harbour of port louis, mauritius," *Multi-Span Large Bridges*, 07 2015.

[100] J. Tenenbaum, C. Kemp, T. R. Griffiths, and N. D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," *Science*, vol. 331, pp. 1279 – 1285, 2011.

[101] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, vol. 40, p. e253, 2017.

[102] E. Stamatatos, "Plagiarism detection based on structural information," *ACM Conference on Information and Knowledge Management*, pp. 1221–1230, 10 2011.

[103] S. Ferilli, "A sentence structure-based approach to unsupervised author identification," *Journal of Intelligent Information Systems*, vol. 46, 12 2014.

[104] M. Coll Ardanuy and C. Sporleder, "Structure-based clustering of novels," in *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*. Gothenburg, Sweden: Association for Computational Linguistics, apr 2014, pp. 31–39. [Online]. Available: https://www.aclweb.org/anthology/W14-0905

[105] A. C. Anderson, "The process of structure-based drug design," *Chemistry Biology*, vol. 10, no. 9, pp. 787 – 797, 2003.

[106] E. H. B. Maia, "Structure-based virtual screening: From classical to artificial intelligence," *Frontiers in chemistry*, vol. 8, no. 3430, pp. 787 – 797", 04 2020.

[107] W. M. Ng, A. J. Stelfox, and T. A. Bowden, "Unraveling virus relationships by structure-based phylogenetic classification," *Virus Evolution*, vol. 6, no. 1, 02 2020.

[108] T.-S. Chen, "Anodic degradation of caffeine under different operating conditions," *Fresenius Environmental Bulletin*, vol. 24, p. 800, 02 2015.

[109] S. Devi and G. Kandasamy, "Modelling electric circuit problem with fuzzy differential equations," *Journal of Physics: Conference Series*, vol. 1377, p. 012024, 11 2019.

[110] W. Shakespeare, *Hamlet. In T. J. Spencer (Ed.)*, London, England: Penguin Books, 1996.

[111] J. Courtney, "Application of digital image processing to marker-free analysis of human gait," Ph.D. dissertation, 01 2001.

[112] L. Xiang, "A formal proof of the four color theorem," *CoRR*, vol. abs/0905.3713, 2009.

[113] J. M. McCarthy, *Introduction to Theoretical Kinematics*. Cambridge, MA, USA: MIT Press, 1990.

[114] R. Conti, E. Meli, and A. Ridolfi, “A novel kinematic architecture for portable hand exoskeletons,” *Mechatronics*, vol. 35, 04 2016.

[115] E. Guresen and G. Kayakutlu, “Definition of artificial neural networks with comparison to other networks,” *World Conference on Information Technology*, vol. 3, pp. 426 – 433, 2011.

[116] A. Sperduti and A. Starita, “Supervised neural networks for the classification of structures,” *IEEE Trans. Neural Netw.*, vol. 8, p. 429â459, 3 1997.

[117] A.A.Markov, “Extension of the law of large numbers to dependent quantities,” *Izvestiia Fiz. - Matem. Obsch. Kazan Univ.*, pp. 135–156, 1906.

[118] P. Frasconi, M. Gori, and A. Sperduti, “A general framework for adap-tive processing of data structures,” *IEEE Trans. Neural Netw.*, vol. 9, p. 768â786, 9 1998.

[119] A.A.Markov, “Investigation of a notable instance of dependent trials,” *Izvestiia Akad. Nauk (St.Petersburg)*, pp. 61–80, 9 1907.

[120] P. A. Gagniuc, *Markov Chains: From Theory to Implementation and Experimentation*. USA, NJ: John Wiley Sons, 2017.

[121] V. Di Massa, G. Monfardini, L. Sarti, F. Scarselli, M. Maggini, and M. Gori, “A comparison between recursive neural networks and graph neural networks,” *IEEE International Conference on Neural Networks - Conference Proceedings*, pp. 778–785, 01 2006.

[122] A. Kuchler and C. Goller, “Inductive learning in symbolic domainsusing structure-driven recurrent neural networks,” *Lecture Notes in Computer Science*, vol. 1137, p. 1464â1470, 09 1996.

[123] T. Schmitt and C. Goller, “Relating chemical structure to activity: An application of the neural folding architecture,” *Workshop Fuzzy-Neuro Syst./ Conf. Eng. Appl. Neural Netw.*, p. 170â177, 1998.

[124] E. Francesconi, P. Frasconi, M. Gori, S. Marinai, J. Sheng, G. Soda, and A. Sperduti, “Logo recognition by recursive neural networks,” *Lecture Notes in Computer Science â Graphics Recognition*, 1997.

[125] M. Hagenbuchner and A. C. Tsoi, “A supervised training algorithmfor self-organizing maps for structures,” *Pattern Recognit. Letters*, vol. 26, p. 1874â1884, 2005.

[126] M. Gori, M. Maggini, E. Martinelli, and F. Scarselli, “Learning user profiles in nautilus,” *Int. Conf. Adaptive Hypermedia Adaptive Web-Based Syst., Trento, Italy*, p. 323â326, 08 2000.

[127] M. Bianchini, P. Mazzoni, L. Sarti, and F. Scarselli, “Face spotting incolor images using recursive neural networks,” *1st Int. Work-shop Artif. Neural Netw. Pattern Recognit. Florence, Italy*, p. 76â81, 09 2003.

[128] M. Bianchini, M. Gori, and F. Scarselli, “Processing directed acyclic graphs with recursive neural networks,” *IEEE Trans. Neural Netw.*, vol. 12, p. 1464â1470, 11 2001.

[129] T. Gartner, “Kernel-based learning in multi-relational data mining,” *ACM SIGKDD Explorations*, vol. 5, pp. 49–58, 2003.

[130] P. Mahe, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert, “Extensions of marginalized graph kernels,” *Proc. 21th International Conference on Machine Learning (ICML2004)*, p. 70, 01 2004.

[131] H. I. J. Suzuki and E. Maeda, “Convolution kernels with feature selection for natural language processing tasks,” *ACL*, pp. 119–126, 2004.

[132] B. Hammer and B. Jain, “Neural methods for non-standard data,” *Proc. 12th Eur. Symp. Artif. Neural Netw.*, 03 2004.

[133] O. Cappé, E. Moulines, and T. Rydén, *Inference in hidden markov models*, 2005.

[134] S. Brin and L. Page, “The anatomy of a large-scale hypertextual websearch engine,” *7th World Wide Web Conf.*, pp. 107–117, 04 1998.

[135] J. Kleinberg, “Authoritative sources in a hyperlinked environment,” vol. 46, p. 604â632, 1999.

[136] J. Lafferty, A. McCallum, , and F. Pereira, “Conditional random fields:probabilistic models for segmenting and labeling sequence data,” *18th Int. Conf. Mach. Learn.*, p. 282â289, 2001.

[137] F. V. Jensen, *Introduction to Bayesian Networks*. New York: Springer-Verlag, 1996.

[138] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning*. Cambridge, MA: MIT Press, 2007.

[139] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[140] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[141] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, “Graph neural networks: A review of methods and applications,” *CoRR*, 12 2018.

[142] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, “A simple neural network module for relational reasoning,” *Advances in Neural Information Processing Systems*, 2017.

[143] R. Palm, U. Paquet, and O. Winther, “Advances in neural information processing systems,” *Neural Information Processing Systems*, vol. 31, pp. 3368–3378, 2018.

[144] S. Sukhbaatar, A. Szlam, and R. Fergus, “Learning multiagent communication with backpropagation,” *Advances in Neural Information Processing Systems*, 2016.

[145] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, “Learning decentralized controllers for robot swarms with graph neural networks,” *Proceedings of Machine Learning Research*, 03 2019.

[146] H. Dai, B. Dai, and L. Song, “Discriminative embeddings of latent variable models for structured data,” *International Conference on Machine Learning*, 2016.

[147] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” *International Conference on Neural Information Processing*, 2016.

[148] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning convolutional neural networks for graphs,” *In Proceedings of Machine Learning Research*, 2016.

[149] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *ICLR*, 2017.

[150] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *International Conference on Neural Information Processing*, vol. abs/1706.02216, 2017.

[151] N. Park, A. Kan, X. L. Dong, T. Zhao, and C. Faloutsos, “Estimating node importance in knowledge graphs using graph neural networks,” *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery  Data Mining*, Jul 2019.

[152] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardinil, “Computational capabilities of graph neural networks,” *IEEE Transactions on Neural Networks*, Feb 2009b.

[153] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, “Deep sets,” *International Conference on Neural Information Processing*, pp. 3391–3401, 2017.

[154] R. Herzig, M. Raboh, G. Chechik, J. Berant, and A. Globerson, “Mapping images to scene graphs with permutation-invariant structured prediction,” *International Conference on Neural Information Processing*, 2018.

[155] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gomez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints,” *Advances in Neural Information Processing Systems*, 2015.

[156] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, “Molecular graph convolutions: moving beyond fingerprints,” *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, p. 595â608, Aug 2016.

[157] R. Beck, Y. Glaser, P. Sadowski, and I. Szapudi, “Refined redshift regression in cosmology with graph convolution networks,” *NeurIPS*, 2019.

[158] T. Bister, M. Erdmann, J. Glombitza, N. Langner, J. Schulte, and M. Wirtz, “Identification of patterns in cosmic-ray arrival directions using dynamic graph convolutional neural networks,” *Astroparticle Physics*, 2020.

[159] F. Yang, W. Yin, T. Inamura, M. Bjorkman, and C. Peters, “Group behavior recognition using attention-and graph-based neural networks,” *European Conference on Artificial Intelligence (ECAI)*, 02 2020.

[160] A. Luo, F. Yang, X. Li, D. Nie, Z. Jiao, S. Zhou, and H. Cheng, “Hybrid graph neural networks for crowd counting,” *AAAI Conference on Artificial Intelligence*, 2020.

[161] Z. Li, Q. Chen, and V. Koltun, “Combinatorial optimization with graph convolutional networks and guided tree search,” *International Conference on Neural Information Processing*, 2018.

[162] M. Zitnik, M. Agrawal, and J. Leskovec, “Modeling polypharmacy side effects with graph convolutional networks,” *Bioinformatics*, vol. 34, no. 13, p. 457â466, 2018.

[163] D. Matsunaga, T. Suzumura, and T. Takahashi, “Exploring graph neural networks for stock market predictions with rolling window analysis,” *NeurIPS 2019 Workshop on Robust AI in Financial Services*, 2019.

[164] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin, “A graph neural network framework for social recommendations,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[165] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. F. Wang, “Multi-label zero-shot learning with structured knowledge graphs,” *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[166] K. Marino, R. Salakhutdinov, and A. Gupta, “The more you know: Using knowledge graphs for image classification,” *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[167] S. Qi, W. Wang, B. Jia, J. Shen, and S.-C. Zhu, “Learning human-object interactions by graph parsing neural networks,” *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[168] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake, “Propagation networks for model-based control under partial observation,” *ICRA*, 2019.

[169] N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, and D. Zoran, “Visual interaction networks,” *Advances in Neural Information Processing Systems*, 2017.

[170] S. van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber, “Relational neural expectation maximization: Unsupervised discovery of objects and their interactions,” *ICLR*, 2018.

[171] A. Garcia-Garcia, B. Zapata-Impata, S. Orts, P. Gil, and J. Rodriguez, “Tactilegcn: A graph convolutional network for predicting grasp stability with tactile sensors,” *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 07 2019.

[172] F. Gu, W. Sng, T. Taunyazov, and H. Soh, “Tactilesgnet: A spiking graph neural network for event-based tactile object recognition,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 07 2020.

[173] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[174] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” *International Conference on Learning Representations (ICLR)*, 2017.

[175] Cho, K. and Van Merri Ìenboer, B.and Gulcehre, C. and Bahdanau, D. and Bougares, F. and Schwenk, H. and and Bengio, Y. , “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *Conference on Empiri-cal Methods in Natural Language Processing (EMNLP)*, 2014.

[176] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” *International Conference on Learning Representations*, 2018.

[177] T. Hoover, “Oobleck: Is it a solid or a liquid?” *Science Scope*, vol. 41, pp. 14–16, 07 2018.

[178] F. Karmali, T. K. Clark, A. D. Artiles, D. P. Sherwood, R. G. Garza, and L. R. Young, “Development of a countermeasure to enhance sensorimotor adaptation to altered gravity levels,” *IEEE Aerospace Conference*, pp. 1–7, 2016.

[179] Diederik P. Kingma and Max Welling, “Auto-encoding variational bayes,” *Proceedings of the Interna-tional, Conference on Learning Representations (ICLR)*, 2014.

[180] Rezende, D. J. and Mohamed, S. and Wierstra, D., “Stochastic backpropagation and approximate inference in deep generative models,” *International Conference on MachineLearning (ICML)*, 2014.

[181] Lin, Z. and Feng, M. and Nogueira dos Santos, C. and Yu, M. and Xiang,B. and Zhou, B. and and Bengio, Y. , “A structured self-attentive sentence embedding,” *International Conference on Learning Representations (ICLR)*, 2017.

[182] Maddison, C. J. and Mnih, A. and Teh, Y. W., “The concretedistribution: A continuous relaxation of discrete ran-dom variables,” *International Conference on LearningRepresentations (ICLR)*, 2017.

[183] Y. Zhang and Q. Zhang and R. Yu, “Markov property of markov chains and its test,” *2010 International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 1864–1867, 2010.

[184] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[185] T. Takase, S. Oyama, and M. Kurihara, “Why does large batch training result in poor generalization? a comprehensive explanation and a better strategy from the viewpoint of stochastic optimization,” *Neural Computation*, vol. 30, pp. 1–19, 04 2018.

[186] S. L. Smith, P.-J. Kindermans, and Q. V. Le, “Don’t decay the learning rate, increase the batch size,” *International Conference on Learning Representations*, 2018.

[187] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[188] M. Clyde, “Experimental design: Bayesian designs,” in *International Encyclopedia of the Social Behavioral Sciences*, N. J. Smelser and P. B. Baltes, Eds. Oxford: Pergamon, 2001, pp. 5075 – 5081.

[189] F. F. Cooperstock, *General relativistic dynamics extending Einstein’s legacy throughout the universe*. Singapore: World Scientific, 2009.

[190] B. Steffens, *Ibn Al-Haytham: First Scientist (Profiles in Science)*. Morgan Reynolds Publishing, 2006.

[191] G. Monkman, S. Hesse, R. Steinmann, and H. Schunk, *Robot Grippers*. Wiley, 2007.

[192] J. Khan, “Some aspects of multi fingered grippers,” *National Seminar on Solar Robotics*, 10 2015.

[193] Faure, François and Duriez, Christian and Delingette, Hervé and Allard, Jérémie and Gilles, Benjamin and Marchesseau, Stéphanie and Talbot, Hugo and Courtecuisse, Hadrien and Bousquet, Guillaume and Peterlik, Igor and Cotin, Stéphane, “SOFA: A Multi-Model Framework for Interactive Physical Simulation,” in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, ser. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Y. Payan, Ed. Springer, Jun. 2012, vol. 11, pp. 283–321.

[194] Holland, D. and E. J. Park and P. Polygerinos and G. J. Bennett and C. J. Walsh, “The soft robotics toolkit: Shared resources for research and design,” *Soft Robotics*, vol. 1, no. 3, pp. 224–230, 2014.

[195] M. T. Keller and W. T. Trotter, *Applied Combinatorics*, USA, 2017.

[196] S. M. Stigler, *The History of Statistics: The Measurement of Uncertainty before 1900*. Cambridge: Harvard: Harvard UNiversity Press, 1986.

[197] X. Yan, *Linear Regression Analysis: Theory and Computing*. World Scientific, 2009.

[198] F. Rosenblatt, “The perceptron - a perceiving and recognizing automaton,” *Cornell Aeronautical Laboratory*, 1957.

[199] D. Smilkov, S. Carter, D. Sculley, F. Viégas, and M. Wattenberg, “Direct-manipulation visualization of deep networks,” *International Conference on MachineLearning*, 08 2017.

[200] Bengio, P. Simard and P. Frasconi., “Learning long-term dependen-cies with gradient descent is difficult,” *IEEE Transactions on NeuralNetworks*, vol. 5, pp. 157–166, 1994.

[201] L. Kelvin, *Electrical Units of Measurement*. Popular Lectures and Addresses, 1883, vol. 1.

[202] M. Hossin and S. M.N, “A review on evaluation metrics for data classification evaluations,” *International Journal of Data Mining Knowledge Management Process*, vol. 5, pp. 01–11, 03 2015.

[203] G. S. Handelman, H. K. Kok, R. V. Chandra, A. H. Razavi, S. Huang, M. Brooks, M. J. Lee, and H. Asadi, “Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods,” *American Journal of Roentgenology*, pp. 38–43, 2019.

[204] A. Jain, A. Zamir, S. Savarese, and A. Saxena, “Structural-rnn: Deep learning on spatio-temporal graphs,” *CVPR*, pp. 5308–5317, 06 2016.

[205] J. D. Almeida, *Sensorimotor Graph: Action-Conditioned Relational Forward Model Learning of Robotic Soft Hand Dynamics*, 2020 (accessed December, 2020). [Online]. Available: https://github.com/joaodalmeida/SMG

[206] E. Fetaya, *Neural relational inference for interacting systems*, 2020 (accessed November, 2020). [Online]. Available: https://github.com/ethanfetaya/NRI