PHD

Learning What To Say And What To Do: A Model For Grounding Language And Actions

Antunes, Alexandre

Award date:
2020

Awarding institution:
University of Plymouth

UNIVERSITY OF
PLYMOUTH

Learning What To Say And What To Do: A Model For Grounding Language And Actions

by

Alexandre Antunes

A thesis submitted to University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Electronics and Mathematics

July, 2020

Acknowledgements

All of this started with who would turn out to be my future supervisor asking me if I had not applied for this project on purpose. For this opportunity, the opportunity to do research with other wonderful researchers, experiencing working in other labs across the world, and for the well structured feedback which remained a constant even when I was working elsewhere, for all that, I will always be thankful to Angelo Cangelosi. Without him none of this would have been possible.

I would like to thank my second supervisor, Alban Laflaquière, for the huge help and feedback provided when I moved to Paris and faced a deadlock in my research. His knowledge and willingness to help provided a huge boost to my research when I most needed it. Also in the same line of thought I would like to thank Michael Garcia-Ortiz, who despite not being in any way related to my PhD, somehow found some time to give some insights. And also to tease me about using LSTMs for everything. I feel like my productivity increased a lot with the support from both, and I will always be thankful to them for that.

I would like to thank Ogata-sensei for receiving me in his lab in Waseda, Japan, and guiding me through my research into these models, and for supporting me since then. Also to Vadim Tikhanoff, who supervised me in my stay at IIT, providing a lot of technical (and not only...) help.

I would like to thank my first family, Rui, Ju and Joana, for supporting me in whatever way they could during this process, and to my second family, Edgar, Sofia, Manuel, Raquel and Zé for the support they constantly provide. I see you all less often now, but you are not any further from the heart.

I would like to thank my colleagues in both Plymouth University and SoftBank Robotics Europe; I always felt at ease with all of you, and couldn't wish for a better group.

A particular thank you for the friends that supported me through the whole thing, independently of how near or far they are: Debora, Emmanuel, Samuele, Luca, Oksana; I hope we continue meeting whenever possible, and complain about life as always!

A special thank you for the usual group, Gui, Wei, Jonny, Gonçalo, Ralfe, Simão, Rita, Jen and Sara. This group has held solid since immemorial times, I truly wish it remains so, no matter where we all end up in the meantime.

A no less special thank you to the "Portugal group" (and less "Portugal" every year...): Inês, Banha, Bernardo, Lisboa, Manito, and of course Mónica and Carlos - I hope I keep having the opportunity to see you all every now and again, and may our diaspora lead to more visits to different places!

And for last I leave my borrowed sister, Bahar. We have supported each other this far, in everything - may this never change, no matter how far.

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee. Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

This work has been carried out by Alexandre Antunes under the supervision of Prof. Dr. Angelo Cangelosi, Dr. Alban Laflaquière, and Dr. Torbjorn Dahl. The work was funded by the H2020 Marie Skłodowska-Curie Innovative Training Networks through the APRIL (H2020-MSCA-ITN-2015-674868) project.

Relevant scientific seminars and conferences were regularly attended at which work was often presented:

Publications submitted as part of the thesis:

Antunes, A., Pizzuto, G., & Cangelosi, A. (2017a). Communication with speech and gestures: applications of recurrent neural networks to robot language learning. In *Proceedings of GLU 2017 International Workshop on Grounding Language Understanding*, (pp. 4–7)

DOI: 10.21437/GLU.2017-1

Antunes, A., Laflaquière, A., & Cangelosi, A. (2018). Solving bidirectional tasks using mtrnn. In *2018 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, (pp. 19–25). IEEE

DOI: 10.1109/DEVLRN.2018.8761012

Antunes, A., Laflaquière, A., Ogata, T., & Cangelosi, A. (2019). A bidirectional multiple timescales lstm model for grounding of actions and verbs. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ

DOI: 10.1109/IROS40897.2019.8967799

Publications submitted as parallel work:

Antunes, A., Saponaro, G., Morse, A., Jamone, L., Santos-Victor, J., & Cangelosi, A. (2017b). Learn, plan, remember: A developmental robot architecture for task solving. In *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, (pp. 283–289). IEEE

DOI: 10.1109/DEVLRN.2017.8329819

Presentations and conferences attended:

2016 IEEE International Conference on Robotics and Automation (ICRA)

2017 International Workshop on Grounding Language Understanding (GLU)

2017 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)

2018 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)

2019 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)

2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Word count for the main body of this thesis: **31469**

Signed: Alexandre Antunes

Date: 25 July, 2020

Abstract

LEARNING WHAT TO SAY AND WHAT TO DO: A MODEL FOR GROUNDING OF LANGUAGE AND ACTIONS

Alexandre Gomes Pereira Antunes

Automation is becoming increasingly important in nowadays society, with robots performing a lot of repetitive tasks in industry and even entering our households in the form of vacuum cleaners and lawn mowers. When considering regular tasks outside of the controlled environments of industry, robots tend to perform poorly. In particular, in situations where robots have to interact with humans, a problem arises: how can a robot understand what the human means? While a lot of work has been made in the past towards visual perception and classification of objects, but understanding what action a verb translates into has still been an unexplored area. In solving this challenge, we would enable robots to execute commands given in natural language, and also to verbalise what actions they are performing when prompted.

This work studies how a robot can learn the meaning behind the sentences humans use, how it translates into its perception and the real world, but also how to translate its actions into sentences humans understand. To achieve this we propose a novel Bidirectional machine learning model, along with a data collection module that can be used by non-technical users. The main idea behind this model is the ability to generalise to novel concepts, being able to compose new sentences and actions from what it learned previously. Humans show this ability to generalise from a young age, and it is a desirable feature for this model. By using humans natural teaching instincts to teach the robot together with this generalisation ability we hope to obtain a model that allows people everywhere to teach the robot to perform the actions we desire. We validate the model in a number of tasks, using an iCub and Pepper robots physically interacting with objects in order to complete a natural language command. We test different actions, including motor actions and emotional displays, while using both transitive and intransitive verbs in the natural language commands.

The main contribution of this thesis is the development of a Bidirectional Learning Algorithm, applied to a Multiple Timescale Recurrent Neural Network enabling these models to link action and language in a bidirectional way. A second contribution sees the extension of Multiple Timescale architectures to Long Short-Term Memory models, increasing the capabilities of these models. Finally the third contribution is in the form of data collection modules, with the development of an easy-to-use module based on physical interaction and speech to provide the iCub and Pepper robots with the data to be learned.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots are becoming increasingly present in nowadays society. From automated vacuum cleaners to guiding and receptionist robots, from voice-commanded smart phones to autonomous vehicles, these machines are appearing everywhere from common areas to our households. So far, they have been built for specific tasks and are incapable of performing other different tasks in our world, but the future points towards robots that are able to perform most tasks a human can, interact with us in our own language, and recognise and react to our emotions. All this requires robots to interact with an unstructured environment, learn from it and from humans, understand its own surroundings and acting appropriately.

Nowadays, it is very common to find systems that recognise and reply in a reasonable way to voice commands, like Siri (Apple) or Alexa (Amazon), but these systems usually live in their own semantic bubbles, being incapable of recognising and interacting physically with the real world. Robots that do interact with the real world usually use pre-programmed actions that depend on very little feedback and/or recognition. To deal with more unstructured environments robots, much like humans, have to rely on their senses (sensors) to acquire information. This information, much like the environment it refers to, is unstructured, and for a robot to make use of it, to *understand* it, it has to *ground* this sensory information in some form of representation. Humans ground their knowledge in words and verbs, creating a grammar structure to connect and make sense of different types of information about the world, and thus it comes naturally that robots, too, could be taught to do this to interact in a natural way with humans, in the future. One way to do this is to teach robots to interact with the world and humans the same way

we teach children. While children learn to interact physically with the world before they learn to speak and understand what is being said to them, there is a strong connection between physical actions and the language representations, a connection suggested by research in neuroscience (Pulvermüller et al. (2005), Pulvermüller (2005), Pulvermüller & Fadiga (2010)).

By emulating these behaviours in a robot we enable it to learn how to deal with novel situations, with easy interaction with non tech-savvy humans, and to execute instructions requested by them. This research leads to developments in Machine Learning, through the use of Artificial Neural Network (ANN), and Developmental Robotics, with the links between neuroscience, psychology and robotics.

1.1 The Scope

The idea that looking into the "mind of the child" is the path to make machines that think can be traced back to Alan Turing, in the 1950s (Turing (1950)). The field of robotics, virtually nonexistent in the times of Turing, has recently exploded, and with it the emergence of the field of Developmental Robotics. One important pillar of Developmental Robotics is embodied artificial intelligence, where it is assumed that only through physical interaction with the world, and learning from those experiences, can an agent acquire intelligence (Brooks et al. (1998); Pfeifer & Scheier (2001); Pfeifer & Bongard (2006); Sandini et al. (1997)). To guide the implementation of embodied Artificial Intelligence (AI) we rely on studies from neuroscience and psychology, revealing how the "mind of the child" works, and providing insights into how to design the models (Pulvermüller & Fadiga (2010); Samuelson et al. (2011)). This connection between the human mind and the physical embodiment of agents led to the creation of different humanoid robotic platforms that could emulate the human behaviour, like the iCub (Metta et al. (2010)), NAO (SoftBank Robotics Europe), Pepper (SoftBank Robotics Europe), ASIMO (Hirai et al. (1998)), the underlying assumption being that if we have an agent powered by models based on human behaviour and brain structure, using a physical representation that is itself close to human appearance, then it should be able to learn in a human like way. Such an agent could then be taught by any person, much like a child, in order to participate actively in human tasks and with human participants.

While the idea of Developmental Robotics is promising, we are still very far from under-

standing the underlying mechanisms of the human mind. While this provides another window of research in the field of developmental robotics, in implementing and testing emergent theories about the human mind in more test-friendly agents, it has the drawback of the complexity and uncertainty in such models. We will thus list a number of areas the current work focuses on, inside the general domain of Developmental robotics, and how they fit the scope of the thesis.

1.1.1 Frame

The main task of the current work is to endow a robot with the mechanisms necessary to understand and execute instructions in natural spoken language. This objective can be split into two separate areas: i) the robot needs to be able to understand what is being said, and; ii) the robot needs to be able to execute a certain action in the real world.

In terms of understanding what the command means, the robot needs to be able to understand what certain words mean, what they relate to in the world. This task of connecting words with their sensorimotor representations is called *grounding*. This problem was first introduced by Harnad (1990), where it was defined the following way: **"how can the meaning be obtained for a certain symbol without grounding it on other symbols?"**. One possible solution, as suggested by Harnad himself, is to ground elementary symbols on sensory projections, which in turn reflect the world surrounding the agent. In this thesis, we follow this solution, using sensorimotor input and visual perception to ground verbs and objects. While this solution provides an efficient way to learn this grounding, the agent should also be able to compose the meaning for sentences that were not trained yet. This is suggested in the work of Lupyan & Bergen (2016), in that humans actually exhibit *programmability* of the mind, that is, we can understand the meaning of new sentences by connecting the meanings learned previously. Using the authors example regarding monkeys, they have to be trained several times to press a button to achieve a certain task, while humans can simply be told to do it (presuming, of course, that the human has seen a button and pressed something before...). This is also called *compositionality*, and it is explained by Brighton et al. (2005), where the meaning of a sentence would be a function of the meaning of each of its components. Such a definition allows for the creation of novel meanings and sentences by creating unseen combinations of words. It is thus the focus of this thesis to enable the robot not only to ground the sentences, but also to understand

commands that are not explicitly trained.

In terms of motor actions execution, the robot needs to connect the meaning of the command to the object it will interact with, if any, and to the motor action it should execute. The robot will then need to react to the position of the object and calculate the trajectory needed. Children typically learn how to interact with the environment before they generate any sentences; however, works by Pulvermüller & Fadiga (2010) found that the same areas were activated when executing an action or talking about that same action. This seems to suggest a common neural pathway between actions and language, where the meaning for a sentence would correspond to the meaning of the action. Under this hypothesis, children who have yet to learn how to speak are still able to infer some meaning from their actions - they are just unable to put what they mean into words yet. Given the compositional nature of language discussed previously, transferred which is transferred to the level of meaning, it makes sense to verify this same compositionality in actions: a novel action should be possible to compose by using an unseen combination of previously learned meanings.

Finally, in the implementation of this model, several assumptions are considered: the robots that will learn this language and action match will be used in public spaces, as interactive robots, that will execute tasks. In order for them to be trainable, the training methods have to be simple enough for the non-tech-savvy human, and therefore should rely on the simplest methods: spoken language and imitation. The resulting question for this thesis is thus: "Can a robot be taught by a human how to recognise and execute Natural Language commands, in a child-like way?". Any person should then be able to train their personal robot to execute the specific tasks they require. This thesis will therefore focus on the necessary structure, algorithms and learning strategies to implement this model, enabling the robot to learn this link between actions and language.

1.1.2 Developmental Psychology

Homo Sapiens are the most developed species on the Earth, the one that can acquire the most skills and dominate its environment like no other. We can make tools, use these to make bigger tools, shape the environment to suit us, domesticate animals, communicate and store knowledge, and many other skills. This is due, in great part, to the particularly strong ability of learning that we possess, and our ability to formulate concepts into

language.

Human babies are amongst the least skilled newborns amongst all animals. While many animals can walk a few hours after being born, human babies have to learn the skills to survive: at 4 months old they are capable of holding their heads steady and reaching for objects, at 9 they can sit and crawl, and they take their first steps around 12 months old. From then on, they start learning how to speak, and at 18 months they go through a vocabulary explosion, learning many words very fast (Bloom (2013); Mayor & Plunkett (2010)). This language acquisition allows for different sources of knowledge to be transferred: mathematics, history, sciences, and so on.

The processes children go through are explored in the field of Developmental Psychology, which provides insights into the mechanisms that drive this growth, the information children are exposed to, and how they learn the regularities in it. Understanding these mechanisms provides valuable insight towards implementing a model that is able to learn language like children do.

1.1.3 Neuroscience

The human body is the most sophisticated machine we know of, and is therefore a valuable source of inspiration for developing robots. In particular for the field of robotics, the ability of the human being to coordinate its limbs, react to changes in the world very quickly, all this while executing tasks that are described in natural language, is of special importance.

Studying the human brain is hard since we can't rely on invasive techniques, and studies on animals like macaques (Rizzolatti et al. (2002)) is limited by how similar their brain is to the human brain and how much we can compare their behaviour to the human behaviour, not to mention the lack of language. Non invasive techniques like Functional Magnetic Resonance Imaging (fMRI) have been used to track activations of areas of the human brain (Rizzolatti (2005)), providing insight into which areas are responsible for language processing, image processing and motor coordination.

Humans have exhibited a strong connection between actions and language. We use gestures to tell stories and to make references to instances in the world, we go through the motions when describing certain actions, and learning language is easier for children when matching actions to words. This link has been analysed by neuroscientific studies which found a connection in the human brain, with the same areas activating whether we

are executing an action or talking about that action. The neurons present in these areas have been described as "mirror neurons" (Keysers & Fadiga (2008); Rizzolatti et al. (2002)) , and they are of particular interest towards the development of learning models for both actions and language.

1.1.4 Developmental Robotics

Humans are one of the most adaptable living beings in the world. We adapt to the surrounding world, to each other, to different languages, to different concepts, to different environments and even to changes in our own bodies. This is specially visible in the case of children, as they learn to walk, talk, use tools, express themselves, learn new concepts, meet new people, etc.

In order to have successful robots that share our world we can then take inspiration on human beings, as was suggested by Turing (1950). Merging the fields of AI, psychology and neuroscience, along with robotics, we can develop algorithms that model human behaviours on platforms that resemble humans in some way, like humanoid robots (Cangelosi & Schlesinger (2015); Asada et al. (2001)). This allows for two different approaches for science: we can study how robots behave in these circumstances, and compare them to the studies on humans to fine-tune our models, or we can try to simulate theories from neuroscience and psychology on these same robots to provide more evidence for these studies.

The developmental robotics approach usually begins with the human, by analysing studies from neuroscience and psychology, trying to find more information regarding a particular behaviour we intend to model. This model is often designed with an AI approach, making use of machine learning techniques, and implemented on a physical robot, usually representing the human physical abilities, either with just a part of its body (arms, legs, head) or full body (humanoid robots).

In the current state of the art, machines are still far from human performance in the domain of interaction with the physical world, and using developmental robotics approaches in these tasks is very challenging, but a lot of progress has been done recently in the area of computer vision and language processing, allowing for some exploration in the domain of language learning and grounding of visual inputs. It is important to note that most animals coordinate the different sources of input in order to achieve a certain goal, and it

is therefore important to consider this coordination in robots, particularly between visual, motor and touch inputs.

1.1.5 Machine Learning

Teaching machines how to perform tasks is not a trivial issue. While the basic theories of AI have existed for 30 years, it was not until recently that hardware was developed enough for these algorithms to be used in an efficient way. With this extra computational power, in the last 15 years, a lot of advances have been made in algorithms and neural networks.

Two particular areas have been extremely successful in using machine learning methods to solve the task: computer vision and machine translation.

In computer vision, the different combinations of convolutional neural networks have allowed machines to classify, segment, and focus on sections of images, enabling a multitude of applications from face recognition to image searching, to image generation, detection of fakes, etc., and these models are used in numerous places from surveillance cameras to smart phones, to robots.

Machine translation has also seen a lot of progress as a result of recent advances in the field of AI. The development of models like LSTM, developed by Hochreiter & Schmidhuber (1997) and perfected by Gers et al. (1999), has allowed for the learning of large texts, enabling fast training of such models to perform translations between different languages, analyse sentiments of texts or short sentences, and even to make summaries of research papers and other texts.

Other areas have benefited from developments in this field. Analysis of events that happen over time, like music or videos and actions, is now being studied by Recurrent Neural Network (RNN) like the LSTM mentioned previously. In terms of robotics, these models are being used in perception of world states over time, enabling agents to perform actions based on certain goals and their previous actions and experiences.

It is clear that research into machine learning is vital for the advances in robotics, enabling these machines to learn, adapt to the world and even communicate with humans.

1.2 Thesis

The main thesis proposed in this document is the following:

> **A robot can learn to recognise natural language commands and execute them, and to verbalise the action it is doing, being taught like a child, without any explicit prior knowledge.**

Having a robot that can recognise vocal commands and use the context of the world that surrounds it in order to complete these requests would allow for more integration of these machines in peoples homes. Teaching of the robot would require physical demonstrations where the human would guide the robot in the action, and speak the corresponding command, but would otherwise not require any technical expertise, enabling any person to teach the robot the commands it wants.

In order to achieve this result, several research questions have to be answered, both technical and behaviour linked. During this research all four research questions mentioned below were addressed.

Two questions show up as a technical side of the proposal of the thesis. These questions will address problems regarding the current algorithms used in most machine learning tasks, their ability to solve the task at hand, and the general architecture used in this thesis. These research questions provide insight into the behaviours of some algorithms, and leads directly to the implementation of the model:

RQ1: **What neural models would allow for the grounding between actions and language?**

In order to be able to execute instructions in the real world, and to verbalise the actions it is executing, a robot has to understand what the commands mean in the world, and how to express some of its actions and perception in words. The link between these two sources of knowledge is called grounding. This research question will address the issue of grounding with ANN, namely RNN, and whether the existing models can create this mapping between actions and language.

RQ2: **Can existing neural models be trained in a bidirectional way, generating a map-**

ping from language to actions and simultaneously from actions to language?

A successful grounding of actions and language should allow the model to execute in the world a task described in natural language, but also to translate into words what it is seeing and executing in the world. This bidirectionality is addressed by this research questions, studying how RNN perform when trained in such a way, and whether this translates into increased performance in these tasks.

In the case of research questions 3 and 4 a more exploitation oriented side will be approached. These questions are linked mainly with behaviour of the model after training, and are directly linked to the thesis proposal:

RQ3: **Can this model generalise its learned skills to new data, and how does it compare with children?**

The generalisation of the learned skills is an important skill in itself for robots to have. It allows the robot to interact in many different situations, with different objects, by learning only a limited set of actions. This research question will investigate how good the model will be at generalising to this new data, and how it compares to human performance in this matter.

RQ4: **Can the robot be taught to verbalise what action it is doing?**

One important aspect that should be present in interactive robots is the ability to explain what it is doing. This provides more interaction with the humans, and allows the human to understand what the robot is trying to achieve, and whether it needs to be corrected. This research question will study the ability of the same model to verbalise the action it is doing, without affecting the original intention of executing a certain verbal instruction.

1.3 Research Approach

1.3.1 State of the Art Review

The first step into the thesis is to analyse the state of the art in a number of fields, most importantly in psychology, neuroscience, robotics and machine learning, all in connection

with language and action grounding. Psychology and neuroscience will be the main drive for the model, inspiring the architecture and providing data for the results we hope to achieve, in comparing our model to the human performance. Research into robotics and machine learning will provide the technical background for the implementation of the model, in answering questions like what are the best algorithms for modelling actions or language, which type of architectures work best for sequential data, what visual perception models to use and integrate on the grounding module. Research in the state of the art of these disciplines should narrow down our options and return either a successful model we could expand on, or a gap in knowledge that could be filled up by a new model.

1.3.2 Model

To solve the task proposed by the thesis we propose a single ANN architecture that will connect language to actions, generating a motor output for a certain language input. In addressing RQ3, we opt for a Multiple Timescales (MT) type of architecture, where different levels of the architecture react to different hierarchies of behaviours. In addition to RQ3, these types of models are also used in grounding, addressing RQ1. To answer RQ2 and RQ4 we implement a bidirectional training algorithm on the model, in order to train the model to work not only in the original direction, from language to actions, but also the reverse, outputting the sentence that corresponds to the action it is executing. This architecture is the fundamental piece of this thesis, and this work details the research and development steps that led to the final version of the model.

1.3.3 Study Design

In order to develop and evaluate the proposed model, we designed four different experiments, with increasing complexity, which will be described in Chapters 3 through 6. Each of these experiments aims at evaluating different aspects of the model, from the training algorithm to the final learning behaviour of the full model. The first two experiments studied different models and their learning behaviours, namely in terms of bidirectionality and generalisation, thus addressing RQ1 and RQ2. The last two experiments were directed towards exploring and exploiting the model, obtaining statistical data on its behaviour, and comparing to human performance. In all experiments we used data from both language and robot motors, the main data sources for the model. The considered setup can

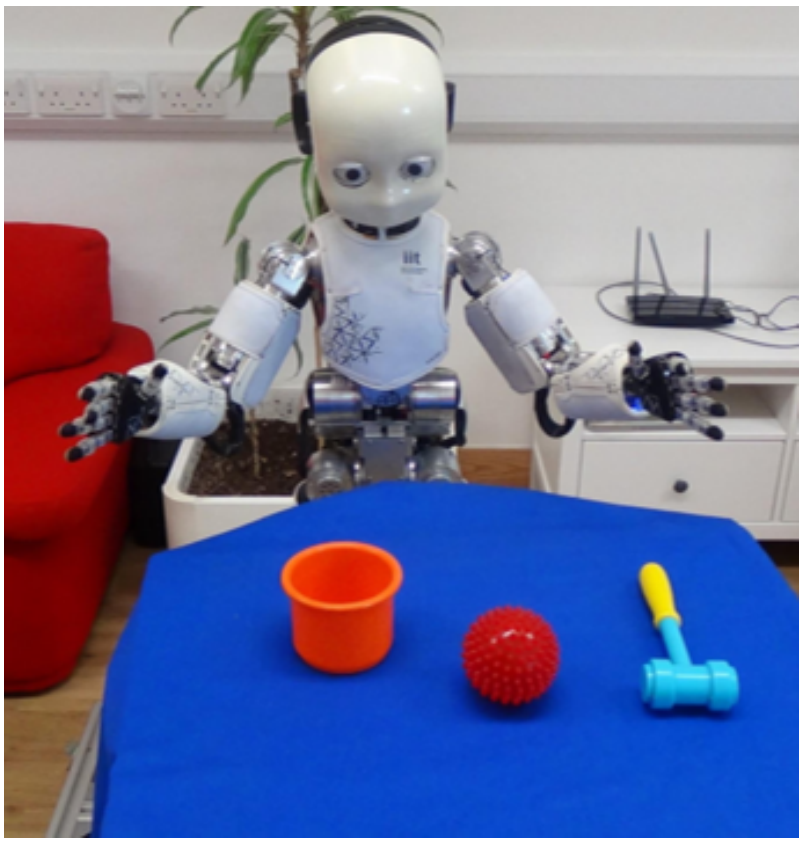


Figure 1.1: Example of the setup for an experiment. The iCub robot is placed in front of a table with the object on the table. During the experiment only a single object can be on the table.

be seen on Fig. 1.1, where the object should be in a position reachable to the robot.

On a first experiment we evaluated the ability of recurrent models to perform bidirectional training. While this is not the main point of the research, it is a critical element for RQ2 and RQ4, which would not be trivial to solve in later stages, and at the same time narrows the choices for RQ1. At a second stage, the models studied on the first experiment were extended to address RQ3, and we tested the model with an iCub robot to execute verbal commands and verbalise the actions it performs. In extending the models studied in the first experiment, we answered RQ1 by choosing a model that is able to perform this task.

1.3.4 Terminology

Throughout this thesis there are some terms used that need to be explained in further detail. The first one of these terms is the term *meaning*, which we use to denominate the encoding that connects language to actions and vice-versa. As such, when the *meaning layer* is mentioned, we imply that this layer is responsible for this mapping.

A second term is the term *grounding*. Grounding corresponds to the association of a

sensory input to an abstract concept. As an example, we can consider the word "red" to be grounded if it is associated with the sensory input of a red-coloured image. we will explore the definition of *grounding* in further detail on Sec. 2.1.2.

Finally, a third term should be explained in further detail: *bidirectionality*. In many works with RNN, this term refers to the direction of recurrence, meaning a cell at a certain time-step receives inputs not just from the previous time-step (normal recurrence), but also from the next time-step (bidirectional recurrence). This is only possible when the networks have access to the full data from the start (i.e.: in the case of text processing works). In this thesis we make use of the term bidirectionality to describe a different behaviour, namely the ability of a network to generate outputs in two different directions. Throughout this thesis the term "bidirectional" or "bidirectionality" will always refer to this second definition.

1.4 Contributions

This research work led to several contributions towards the general field of robotics. In this section we evaluate the contributions both in scientific contributions and in technical contributions.

1.4.1 Scientific Contributions

- A first contribution saw the study of **Bidirectional Training** in MTRNN, highlighting their compositional and generalising behaviours when trained and used in both directions. (Chapter 3, Antunes et al. (2018))
- A second contribution presented the first attempt merging language and actions in a single contiguous neural network, making use of **Bidirectional training in a MTLSTM** structure. This study showed that the network kept all three of its desired behaviours, namely compositionality, generalisation, and bidirectional training. We validate the model using data from an iCub robot. (Chapter 4, Antunes et al. (2019))
- We implemented a data collection modules that enables anyone to train an iCub and Pepper robots by means of speech and physical interaction. We use these modules to collect data that is then used to train the MTLSTM model, showing the robustness of the model to increased amount and different sequences, as well as data from

different robots. We train the model also with transitive and intransitive verbs, further exploring the capabilities of the MTLSTM model. We present experimental results with two robots, iCub and Pepper. (Chapter 5)

- The model was tested for its meaning understanding, by monitoring the activations at the connecting layer, labelled "meaning layer". It was further tested for its ability to deal with synonyms, by testing how it connects similar actions to different verbs. The tests were performed on a smaller version of the iCub dataset. (Chapter 6)

1.4.2 Technical Contributions

- Development and implementation of a Bidirectional Training algorithm in a MTRNN model in Tensorflow, with the code available online (https://github.com/AlexAntn/MTRNN_experiments).
- Development of a novel MTLSTM model in Tensorflow, together with the bidirectional training algorithm, with all code available online (https://github.com/AlexAntn/MTLSTM).
- Implementation of data collection modules for Pepper robot, able to collect images, natural language command and the motion sequence for the model.
- Implementation of a data collection module for the iCub robot completely controlled by verbal commands and direct interaction with a human trainer. The module stores verbal command, image and motion sequence, with motions including arms, hands, head, and torso.

1.4.3 Other Contributions

- Integration of object learning modules from University of Plymouth, UK, with planning and reasoning modules from Instituto de Sistemas e Robótica, Lisbon, for the POETICON++ project, that enables an iCub robot to decide which objects he needs to learn to solve a certain task, and enabling the robot to react to new unknown objects during execution. This contribution was a minor extension of the POETICON++ project, developed in parallel to APRIL project. (Annex A, Antunes et al. (2017b))

1.5 Structure

Below we describe the structure of this thesis together with a brief description of each chapter.

- This chapter presents the general goal for this work and how it fits in the different fields of research, the research questions addressed in its development and the main contributions, technical and scientific.
- Chapter 2 describes the different fields related to this research, from neuroscience and psychology to machine learning and robotics. A strong emphasis is given to the machine learning models given the nature of this work. This chapter also explains the concepts of Grounding and Generalisation.
- Chapter 3 introduces the first experiment in this work, in using MTRNN models to learn grounding between actions and language by means of CS. This experiment showcases the ability of these models to successfully learn this link.
- Chapter 4 presents the first fully connected model able to link actions and language without any intermediate CS. We also present evaluation data showcasing its ability to generalise to new concepts.
- Chapter 5 presents the data collection module, describing its implementation and use, and evaluates the data collected by training the fully connected model on this new dataset, both larger and more complex.
- Chapter 6 addresses the properties of language compositionality and the emergence of meaning, by exploring the behaviour of the model when exposed to synonyms and studying the representations in the model.
- Chapter 7 compares the findings of the previous chapters to the initial research questions. It discusses also future extensions for this field of research and for the model.
- Chapter 8 concludes the thesis with a summary of the findings and final status of the model.

Chapter 2

Background

One of the long term dreams of the field of robotics is to have robots that are capable of understanding humans, interact with humans, and act alongside humans. To achieve this robots would have to share concepts, motor capabilities and physical spaces, and thus should be able to learn how to interact in these domains. With the increasing tendency of moving robots outside of controlled environments into more public areas where they interact physically and through spoken language with humans, it becomes more critical to develop solutions that enable robots to understand humans, their surroundings and their capabilities. Several projects have brought up these questions in a multitude of scenarios, from collaboration in an industrial environment in CoLLaboratE[1], focusing on teaching robots to share the physical environment with humans, to companion robots in COGNIRON (The Cognitive Robot Companion) [2] and RoboCom++ (Rethinking Robotics for the Robot Companion of the future), to more generic tasks involving different scenarios in SECURE (Safety Enables Cooperation in Uncertain Robotic Environments) [3] and the current project itself, APRIL (Applications of Personal Robotics for Interaction and Learning) [4]. It is clear that future robots will have to perform in many different environments and will be increasingly exposed to humans, and an increasing number of research projects[5] are tackling these challenges and presenting new tools to increase the capabilities of these robots.

In a project dedicated to create a robot that will interact with and learn from its users, it is

[1] https://collaborate-project.eu/
[2] http://www.cogniron.org/final/Home.php
[3] http://secure-robots.eu/
[4] https://www.fose1.plymouth.ac.uk/socem/crns/april/
[5] In H2020 insights it is reported that there are about 20 projects for collaborative robotics alone, ranging from entertainment to industrial collaboration. Source: https://amires.eu/increasing-productivity-with-collaborative-robots-h2020-insights/

fundamental that such a robot is able to understand and reply in natural, spoken language. Language learning and grounding in developmental robotics requires analysing different fields, from the psychology of language learning in children to neuroscience studies on the link between actions and language in the human brain, to different modelling methods to be implemented in robots. In order to understand each of these items, we will review each of them separately, indicating the main points to be taken from each field; and at a second step, we describe the link between the different fields and the main points to be considered in this project.

2.1 Overview of Language and Action Grounding in Developmental Robotics

2.1.1 Inclusion Criteria

Developmental Robotics is an area that concerns itself with methods that develop on previous knowledge, in a way similar to human beings, usually taking inspiration in psychology and/or neuroscience (Asada et al. (2001, 2009); Cangelosi & Schlesinger (2015)). Language and action grounding is focused on the description of perceptual information in natural language, effectively linking language and actions/perception. As such, the following fields should be included in this overview:

- Psychological experiments aimed at explaining a learning behaviour in humans, usually focused on children. These works should be focused on knowledge acquisition.
- Neuroscientific experiments into knowledge localisation in the brain, providing insight into links between different knowledge sources like language and motor actions.
- Computer Science machine learning models. In particular, for this type of research, we are looking into models that link sequences from different information sources, in a hierarchical way.

2.1.2 Definition of Grounding

One of the main distinctions between humans and other animals is the ability to talk, that is, to produce language. With language we are able to synthesise our perceptions into words, which we can then share with other humans. This, however, is only possible because we know what the words mean, that is, we know what they translate into in our own perception. This issue was best illustrated by the Chinese Room argument (Searle (1982); Harnad (1990); Cangelosi et al. (2002)). In this argument, a subject has to answer some questions in Chinese, having no previous experience with the language. They have access to a Chinese-Chinese dictionary, and they could search out the symbols corresponding to the question, and try to come up with an answer, but this effort would be in vain, since any symbol would only be explained by other symbols, leading to a never-ending loop. In fact, without any way to translate from Chinese language to either their mother tongue, or to real-world instances of those symbols, the subject would have no way to understand what they mean in those questions, even if they get a good understanding of the symbol system.

The Chinese Room argument showcases the importance of this mapping between language and perception: If a symbolic system, no matter how complex, is not mapped into a real-world instance, it is impossible to understand what the meaning behind those symbols is. This mapping between the real-world and the symbols is defined as *grounding*. Understanding the mechanisms of language grounding is particularly useful for robotics, since it allows for the design of models that enable a robot to understand what language concepts relate to in its perception of the real world (Cangelosi et al. (2002); Taniguchi et al. (2016)).

2.1.3 Language Learning in Psychology

Language learning in children has been a subject of intense studies. While this field was dominated by the Universal Grammar proposed by Chomsky (1993), more recent theories have presented a more learning-oriented approach to language learning. In the studies of Saffran (2003), evidence was found suggesting children can learn language structure from statistical proprieties of language itself. This work further argues that the similarities found between different languages arise from the same learning mechanisms being used, other than a universal grammar innate to all human beings as suggested by Chomsky.

The theory supported by this work, usually called "constructivist theory for language learning", defends that children have powerful learning mechanisms that allow them to have a burst of language learning from a very early age. Constructivist theories are defended by Tomasello (Tomasello (2000), Tomasello (2009)), who suggests humans don't have any innate language structure, but rather adapt our brains while learning language. Tomasello goes further, defending that children initially form "verb-islands", learning each verb independently before being able to generalise into a grammar structure. This theory, commonly referred to as "verb-island theory", has been challenged by Ninio (2003), who suggests, still on the constructivist point of view, that children start generalising as soon as they learn their first words, but that this generalisation is only successful when enough vocabulary is learned. In the present work, we will consider the constructivist approach in that grammar structure should come from learning and not from any innate system. The learning in our model will be studied and compared to both Tomasello and Ninio theories, checking for generalisation capabilities in the model and at what stage they appear.

2.1.4 Language and Actions in Neuroscience

A lot of research has been made into the neurological pathways for learning actions and language, and how these are related in the brain. A strong suggestion from scientists in the early stages of this field pointed to a common link between actions and language (Iverson & Thelen (1999)), pioneered by Piaget (Piaget (1954)). Later studies have found similar evidence (Pulvermüller et al. (2005), Pulvermüller (2005), Pulvermüller & Fadiga (2010)), presenting a shared compositional nature between actions and language.

We posit that it is from this link between actions, perception and language that *meaning* emerges, effectively grounding the different sources of knowledge. In this work we try to model this link, and expect to see the emergence of meaning from the connection of the different sources.

2.1.5 Language and Action Learning Models

For robots to be useful in interacting physically with humans they have to be able to understand and reply to natural language interaction with human beings. Most systems, commercial and otherwise, make use of an extensive vocabulary to construct the reply, and

are prone to make grammar mistakes. A different route consists of learning the language and its grammar. These systems usually make use of statistical methods and machine learning to learn the structure of a language from interaction with humans. Typically, these systems consist of RNN, which are able to learn from a sequence of either phonemes or letters. Currently, two predominant types of RNN are used in language learning: LSTM and MTRNN.

In the works of Zhong et al. (2014) a RNN with Parametric Bias (RNNPB) was used to ground visual features in a non-symbolic way, relating the visual features of an object and its parameters in a self-organised way. In a different project (Zhong et al. (2016)), Zhong used an MTRNN to ground a set of actions and verbs together. This work was a first attempt at merging language and actions directly, but the language input consisted on a set of pre-established verbs and actions, not being scale-able to new verbs and motor sequences outside the presented dataset.

Some similar works have been done in the past, using many different methods. In the work of Hinaut et al. (2014), the authors use an Echo State Network (ESN), a type of recurrent neural network with a very sparsely connected hidden layer, that is capable of reproducing temporal patterns. This work was capable of learning two different types of words, Semantic (context dependant) and Grammatical (connectors), how to distinguish them, and how to use them in sentences. It enabled commands to be given in a non-predefined way and still be understood by the model as having the same meaning. Although the system enjoyed a lot of flexibility in its use, it was restricted to sentences comprising action, agent and object. Additionally, the actions the robot can use are pre-set and not learn-able, with the focus being on language learning.

Many different works have been done by Ogata et al. (Ogata & Okuno (2013), Hinoshita et al. (2011), Nishide et al. (2009), Yamashita & Tani (2008), Murata et al. (2014b)), with different types of networks being used. In work by Murata et al. (2014b) an MTRNN was used to model reflexive behaviour in cooperative tasks. In Ogata & Okuno (2013) an architecture very similar to the one used in this project was used, with a motor branch of MTRNN connected to a language branch at a higher level. This connection, however, was done by parametric bias, that is, through a control sequence of neurons with a pre-defined value for each action. In our approach, we implement this connection with an extra layer in the network, allowing the control sequences of each branch to be different, but still

connected to each other. In Hinoshita et al. (2011) it was found that MTRNN were able to replicate the hierarchical structure of language, learning this structure from characters and without any pre-trained words or sentences. This work provided a lot of inspiration to design the proposed architecture in the current implementation.

In the works of Peniak (Peniak et al. (2011a), Peniak et al. (2011b), Peniak et al. (2013)) an MTRNN was used to learn the hierarchy in actions instead of language. Motor sequences from an iCub robot were used as input for pre-trained Self Organising Maps (SOMs), that would in turn be the input to an MTRNN. the MTRNN was able to learn and replicate the actions on the dataset, consisting on 9 different actions with 9 different objects in 6 different positions. This dataset was used to train the first versions of the model but it has no visual input (position of an object is given by the direction the head is looking at).

Heinrich et al. (2015) used an MTRNN to learn language, connecting the network with a perception pipeline at the control sequence layer. The input was in the form of phonemes instead of characters, and the network was successfully able to connect language input to the visual scene. This linking of perception at the higher level of an MTRNN also provided some inspiration for our model. In a previous work (Heinrich & Wermter (2014)), Heinrich proposed an extended architecture with many similar concepts to the one proposed in this project: it consisted in 3 MTRNN connected at the control sequence level (Context slow layer). Each branch was responsible for a different aspect, namely language, vision and actions. This work focused on learning language representations for the actions executed by a robot and the objects it was seeing, and did not try to tackle bi-directional learning.

MTRNN have been used extensively in other works by Tani et al. (Park et al. (2015), Jeong et al. (2014), where MTRNN were used to address other problems like planning and generation of behaviours from visual input. In Tani (2014) Tani addresses the compositionality in human brains, of not only language but actions, images, and so on.

Yu et al. (2013) modified an MTRNN in order to make supervised classification of human actions, enabling the network to classify its inputs without setting initial states in parametric biases.

A different type of network commonly used for language learning, although not common for action learning, is the LSTM. These networks have been used extensively, and even been shown to present better results at learning a language.

In the works of Sundermeyer et al. (2012) an LSTM was compared to regular RNN and

proven to perform better at the task of language learning.

In Bowman et al. (2015) a large corpus was created for making Natural Language Inference, called Stanford Natural Language Inference corpus. This corpus was tested with an LSTM, which obtained the best results of any Neural Network model.

LSTM have also been used for sequence-to-sequence learning, a task similar to the one the current project addresses. In Sutskever et al. (2014) an LSTM was used to learn sequences with very few assumptions on the structure of the sequences. LSTM were proven to be successful at the task of translating from English to French, achieving a score close to state-of-the-art when reevaluating statistic machine translation system outputs.

In a different type of sequence to sequence learning, an LSTM was used in Venugopalan et al. (2014) to translate videos into natural language sequences, in order to ground language with a visual scene. The LSTM was partially successful, but was unable to use temporal information on the videos.

More recently the focus has been shifting from RNN to transformers and attention-based networks (Vaswani et al. (2017)) for language learning tasks. These models do not require the sequences to be processed in the correct order, thus allowing for parallel processing of the data, opening these models to be trained with much more data.

2.2 Axes of Analysis

Humans have many adaptive behaviours when learning language and actions. Upon learning a new object, we easily adapt previous concepts to this new object (e.g.: we don't need to relearn the action of hammering when we find another long object, we adapt our previously learned action of hammering to this new object). Less obviously, when we learn a new verb, we often try to generalise it to the grammar structure we have learned, often resulting in mistakes for irregular verbs (e.g.: "cutted") and intransitive verbs (e.g.: "smile the ball"). Ideally, a robot that shares the same space as us, that interacts with other humans in a very human-like way through human-like speech, has to be able to learn and generalise concepts much like we humans do. It is fundamental, then, to test the ability of our model to not only learn but generalise concepts.

Three different types of generalisation should be possible for the robot: i) on a physical domain, the robot has to be able to generalise the actions to different objects and positions;

ii) on the language domain, the robot has to be able to generalise verbs to different objects that it never explicitly learned; and iii) generalise the grammar structure to learn new verbs faster. If the robot is able to perform these three levels of generalisation, it should be possible for it to learn new concepts and adapt them to the concepts it already knows.

2.2.1 Generalisation in the Physical Domain

Generalisation in the physical domain has been achieved in many different ways in the field of robotics, in the past. Cartesian solvers (Pattacini (2011), Pattacini et al. (2010)) could generalise pre-defined actions to different positions, by calculating the necessary position of an arm in order to get the end-effector, usually centred on the palm of a hand, above the object. These models could be trained, adjusting for the height of the end-effector above the object, slight deviations, but learning of a new action required changing the model. In Tikhanoff et al. (2010) Neural Networks were used to train the motion of reaching and grasping an object, but due to the nature of networks used (feed-forward and Jordan neural network) generalisation was not always possible, and the number of actions tried was rather limited (only reaching and grasping). In Pastor et al. (2009) generalisation is achieved by teaching a robot a certain movement, through the use of a master arm controlled by a human, and then tweaking a goal and start parameters. The model used in this work is a Dynamic Movement Primitive, represented by a differential equation, that was adapted to fix some undesirable behaviours. This method generalised to spaces close to the original goal, for which the arm was trained, but its error increased the further from this original position the desired goal got.

Recently, generalisation of motor behaviours has been performed using Recurrent Neural Networks. In the works of Peniak (Peniak et al. (2013), Peniak et al. (2011b), Peniak et al. (2011a)) and Zhong (Zhong et al. (2016), Zhong et al. (2014)) an MTRNN was used to learn and generalise motor behaviours. In their experiments, they would typically exclude some particular combinations of actions and certain positions. They would train the model with this dataset, and test the network for the excluded cases. These models presented very good generalisation capabilities, being able to adapt, even if not perfectly, to situations where objects were put in places they were not trained to handle.

For robots to be successful in real world scenarios they have to be able to deal with the unstructured and ever-changing environments we humans usually live in, which in turn

requires these robots to be very adaptable and capable of generalising their behaviours to many different situations. It would not be helpful to have to teach a robot every single different possible position a cup could be in the house, in order for the robot to be able to perform the action of grasping the cup in any situation. It is, then, very important that, upon teaching a robot a certain action, the robot is able to generalise this action to different places and contexts, with minimal human intervention.

2.2.2 Generalisation of Verbs to Different Objects

Adult humans are extremely good at generalising language, even though it is hard to point exactly when this generalisation capability starts in children. We are capable of constructing sentences with new words, with minimal exposure, generalising from the grammatical structure we have learned previously. These capabilities allow us to share new concepts and experiences, and are of crucial importance in learning.

While robots can be endowed with a sophisticated language system, allowing them to interact with humans in a very natural way (like, in the present, Alexa or Siri), they would not be adaptable to the different people and environments if they do not possess this ability to learn and generalise language concepts. It becomes important, then, to implement a model that allows the robot to learn new concepts in language and is capable of generalising immediately to the concepts it already knows without explicit training.

A model that is able to make such a generalisation was developed by Heinrich et al. (2015), using an MTRNN to learn language and linking it to a feed-forward network for visual recognition. Heinrich proved that an MTRNN was able to generalise to new visual scenes, building sentences to describe them. It is this capability that makes MTRNN an attractive solution to develop a language learning model.

2.2.3 Generalisation of the Grammar Structure

The human capability of generalising new verbs to the grammar structure that was learned previously leads to a very fast learning of new concepts. Such a mechanism would be very useful in robotics, since it would allow a robot to easily adapt to its owners language and surroundings. It is yet unclear if children are born with such a mechanism, generalising from the start (Ninio (2003)), or if it happens only after a certain threshold of knowledge is achieved (Tomasello (2000), Tomasello (2009)), and as such it is also interesting to test

which theory might apply to the model being considered.

In a review by Cambria & White (2014) a brief review on the story of Natural Language Processing (NLP) is made, along with the current trends and future directions. While old NLP systems were very static systems, often designed by experts and not very adaptable, based on syntax only, modern NLP systems rely more on semantics, learning and adaptation, making use of Neural Networks and other systems in order to learn. The authors argue that, while these systems perform very well in a conversation (e.g.: Alexa, Siri, etc.), these agents have no understanding of what is being said - they merely reply according to what they learned. Additionally, with the advent of internet, mass data provided by users, and the general misinformation and so-called "trolling" found online, these systems face a new challenge in learning: distinguishing between valuable information and useless or detrimental information. The analysis of this data requires a context understanding and reaction, which the authors suggest to be the next jump in NLP: Pragmatics. Here, NLP should be aware of its context, but also intent-driven and able to work in an open domain.

In the syntax field, three different types of algorithms have been developed: i) Keyword searching, which relies on keywords on certain texts in order to classify them into some categories; ii) Lexical Affinity, an algorithm that assigns words the probability of "affinity" to a certain category, and iii) Statistical NLP, usually based in machine-learning algorithms, and the most successful of the three types of algorithms.

Graphical models have been used rather successfully in statistical NLP. These models generally work by learning the conditional distribution of a word given its priors, usually comprised of a certain number of preceding words. In the works of Mnih & Hinton (2007) we can see three different graphical models used for this purpose: a Factored Restricted Boltzmann Machine (RBM), a temporal version of this same RBM, and a Log-Bilinear model. While these models are able to learn and generalise language relatively well, they are not developmental models, and learning a new word requires rearranging and re-learning the entire dataset.

A different approach to NLP, like stated before, is the semantics approach. This approach focuses on the meaning of sentences rather than their syntax, and is able to, as such, avoid the mistakes usually associated with syntax-only approaches. This approach can be split into three main fields, i) endogenous NLP, focused only on the text without any prior or external knowledge; ii) taxonomic NLP which usually aims at developing universal

taxonomies relating concepts, instances, attributes and values, along with the relationships to other such concepts; and iii) noetic NLP, which creates a knowledge-base by collecting information on individual objects and creating connections between them, which in turn allows for reasoning on these concepts, creating new concepts. This method, however, uses a construction-based semantic parser in order to structure its knowledge, which in turn is not easily extensible and adaptable to natural language communication.

An example of Noetic NLP is PRAXICON (Pastra et al. (2010)) which uses such a knowledge-base network in order to reason on requests by humans (parsed in the correct structure). It is able to reason solely on the semantics of the problem, translating a single instruction into a sequence of smaller instructions using other available concepts (e.g.: "knife" is associated with "cutting"). While this system has been successfully implemented in other systems and robots (Antunes et al. (2016)), it does not generalise grammar or sentences in a "human-like" and developmental way.

Finally, as Cambria & White (2014) suggest, the next step in NLP is Pragmatic. This approach focuses mostly on "story-telling" and construction of narratives, which is out of focus for the current research; however, for pragmatic approaches to work correctly, it becomes fundamental that an agent is able to understand and generalise grammar, reinforcing the need for such algorithms in NLP.

2.3 Summary and Work Plan

In this literature review we followed the developmental cognitive robotics approach: first, we reviewed existing and predominant theories for language learning and grounding in both psychology and neuroscience, then we investigated how present algorithms performed, trying to find algorithms that could perform this task. We found several algorithms that performed well at either language learning or action learning, but little to no work was found in actual language grounding with actions. Indeed, some works explored the possibility of merging this learning (Zhong et al. (2016)), but one or both "sides", language or action, would be considered discrete, that is, with a limited number of possible actions or words. To learn different words and/or actions, these models would have to be changed and retrained.

One key point found in literature has to do with the simultaneous learning of both actions

and language. The emergence of concepts in the human brain arises from the interaction between action learning and language learning, suggesting a bidirectional link between them. This bidirectional learning is not explored in detail in the field of robotics, and it is therefore critical to study its influence in the grounding of actions and language, leading to the first experiment implementing a bidirectional learning algorithm in Chapter 3. The model chosen for this experiment is a MTRNN, which has been used before in both action and language learning, but in a single direction.

While the experiments in Chapter 3 present the bidirectional learning algorithm, enabling a robot to connect a symbol to its physical perception in a bidirectional way, they do not fully connect language and actions, making use of predefined symbols to do so. In the human brain these symbols emerge from the learning itself, behoving the creation of a model connecting language and actions, in a bidirectional way, learning these symbols in the process. An experiment showcasing this behaviour is described in Chapter 4, using the bidirectional learning algorithm applied to a novel MTLSTM model to connect actions and language directly, in a single model, and without any predefined symbols.

An important key to grounding is the link to the perception of the real world. While a human with a disability might be unable to grasp an object in the same way as a human with no disabilities, they are still capable of attempting the same action in their own way, grounding it to the same verb - grasp. An accurate model of language and action grounding should account for this, and should therefore be able to perform under different motor capabilities. In order to study the MTLSTM model when exposed to different motor capabilities we develop a data acquisition module for two robots, iCub and Pepper, and train the model with this data, connecting language to motor actions performed with the two different robotic platforms. The modules were designed to collect data as raw as possible to simulate the way children learn, being lead by a human instructor to perform the actions in a real world environment. These experiments are presented in Chapter 5.

Finally, a key point in grounding is the connection of two different verbal representations for the same motor action, described as synonyms. Humans often use synonyms to describe situations and actions, and while these synonyms might be very different when described by language, they relate to the same motor action. We test the model for its ability to deal with such synonyms, and explore the symbols generated after learning in Chapter 6.

Chapter 3

Bidirectional MTRNN for Robot Action and Language Learning Tasks

Key points:

- Multiple Timescale Recurrent Neural Network (MTRNN)
  - Continuous Timescale Recurrent Neural Network (CTRNN) - building block of MTRNN
  - Description of an MTRNN model
  - Generalisation and Compositionality behaviour on MTRNN
- Bidirectional training experiments
  - Language learning
  - Motor action learning
- Full integrated model extension
  - Model description
  - Challenges of big MTRNN models - Vanishing Gradient

Current research in AI has been dominated by the use of different neural networks to process different structures of data. When it comes to sequential data, the standard is to use RNN to learn them.

While short sequences are well modelled by even the simplest RNN, longer sequences

have been hard to manage. RNN typically learn to predict the next element in a sequence based on the past elements, which leads to errors when the number of elements in past becomes too large. Increasing the size of the RNN models is often counterproductive since it might lead to problems like Vanishing Gradient, which will be described in Section 3.2.7.

Research into these issues has provided an answer to some of the major problems, in the shape of LSTM, which avoids the problem of Vanishing Gradient. There is, however, another issue with increasing the complexity of data to be learned, namely the compositional nature of the data. In data that exhibits compositional structure, past sequences can be extremely similar while the future can be completely different. An example of such data would be the reaching motion in humans, which can lead to a grasp, or to a push, or other actions that involve touching the object. Further input is needed for the model to be able to learn when to generate one sequence and not another with a similar past. Such an input could be understood as a Control Sequence (CS), that is, a sort of command that would tell the model which output it should be generating.

LSTM have been shown to have difficulties in learning the compositional nature of data, even if being more accurate at generating the individual sequences. In order to learn this data, a different type of model was created, called Multiple Timescale Recurrent Neural Network (MTRNN). This model is structured in a way that sequences with different periods are recorded in different areas of the model, which allows it to compose an output based on these individual sequences, thus exhibiting compositionality. This model allowed research into complex structures such as grammar, action learning and generation, music, and others. In this chapter we will study MTRNN in detail, highlighting the desirable properties of the model, and presenting some experiments performed with these models when trained in a bidirectional way. Finally, we discuss a full integrated model and the challenges that arise from larger MTRNN models.

3.1 Multiple Timescale Recurrent Neural Network

The simplest representation of a MT Architecture can be seen in the form of a MTRNN model. This model is based on the concept of CTRNN, which was developed as a model of the human neurons. An MTRNN is, in essence, a stack of CTRNN with different timescale factors.

3.1.1 Continuous Timescale Recurrent Neural Network

CTRNN emerged as an attempt to model human neurons using already studied normal RNN. They were first developed by Funahashi and Nakamura Funahashi & Nakamura (1993).

The main principle that drives this model is the equation that models a human neuron:

$$\dot{u} = \frac{1}{\tau}(-u + w\sigma(u + \theta) + I) \tag{3.1}$$

where u is the state of the neuron, τ is the timescale factor, w represents the weights applied to the incoming connections, σ corresponds to the activation function, θ is a bias related to the neuron, and I is an external constant input.

When applying this model to ANN we assume discrete time intervals. The equation that defines the change in the internal state of a neuron for a certain time interval comes:

$$u_{i,t} = (1 - \tau_i^{-1})u_{i,t-1} + \tau_i^{-1} \sum_j w_{i,j} x_{j,t} \tag{3.2}$$

where $u_{i,t}$ is the internal state of the neuron i at time step t, τ_i corresponds to the timescale factor for the neuron i, $w_{i,j}$ are the weights connecting neuron j to i, and $x_{j,t}$ is the activation of the j neuron at time step t. A more visual representation of the neuron can be seen in Fig. 3.1.

Equation 3.2 describes a leaky integrator type of behaviour with rate defined by τ. CTRNN thus emerge as a group of such neurons, all with the same timescale factor, being capable of displaying complex temporal behaviour. These networks have been used in many different time-dependent scenarios, from balance and pose in robots (Hénaff et al. (2011)) to temporal pattern detection (Murata et al. (2014a)), to music generation (Bown & Lexer (2006)).

3.1.2 The MTRNN architecture

As was mentioned in the introduction to this section, MTRNN are essentially a stack of CTRNN. The typical MTRNN consists on three instances of CTRNN, organised in a hierarchical way: the bottom CTRNN layer, named I/O, is only connected to the middle

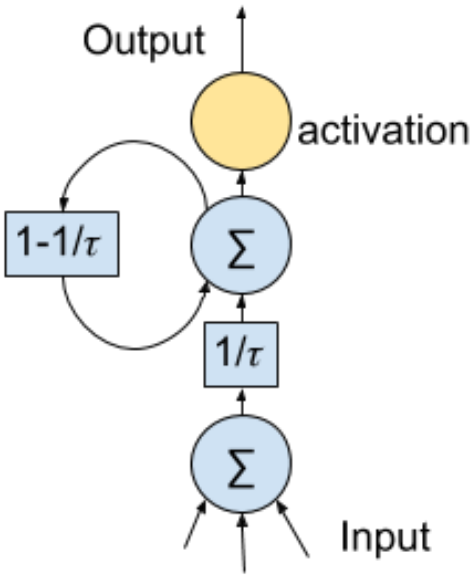


Figure 3.1: Representation of a CTRNN neuron.

CTRNN layer, named Fast Context (FC), which is then connected to the top CTRNN layer, SC. There are no connections between the SC and I/O layers, forcing a sort of pipeline of information flow. In addition to this structure, there is also an increasing timescale factor over the network: typically the I/O layer has a very small timescale factor τ, while the SC layer has a larger timescale factor, generally one order of magnitude higher. This structure allows the network to encode some interesting complex behaviours, with the SC layer usually encoding the slower frequency events, reacting more slowly to inputs, and the I/O layer reacting faster to inputs and thus encoding fast frequency events. Evidence of this behaviour has been presented in works where the network learns motor primitives at the FC layer (Yamashita & Tani (2008), Peniak Peniak et al. (2011a), and Zhong Zhong et al. (2016)). An example of an MTRNN can be seen in Fig 3.2.

MTRNN have two main modes of operation: top-down and bottom-up. For the top-down mode the network is fed an input at the SC layer, generally called Control Sequence CS. The CS usually consists on a vector input to a sub-set of neurons on the top layer, and it can have different natures: it can be a single time-step vector input, permanent input during the whole sequence, or even originate with a different network and changing over time. This input triggers behaviours on the other layers, finally generating an output at the I/O layer. The I/O layer can depend only on the input from the FC layer, or have a feedback loop from the real world (i.e.: it can have the actual position of the robot as feedback, while generating the desired position of the robot).

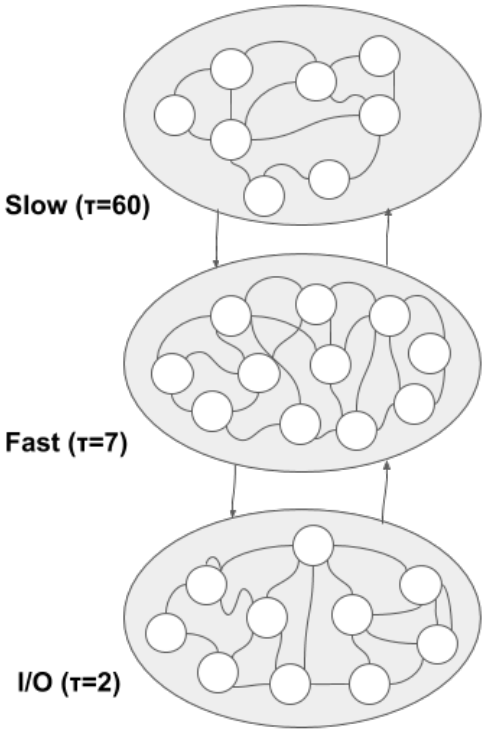


Figure 3.2: Simple MTRNN consisting on three layers, with an example of timescale factors.

A second mode of operation, bottom-up, works in an inverse mode: A sequence is given to the I/O layer, which is then processed through the network, generating a CS at the SC layer. In this mode, the network encodes the input sequences into a smaller key sequence, behaving as a classifier.

To train these models the Back-Propagation Through Time (BPTT) algorithm is often used. This algorithm unfolds a RNN over time, with a mirror of the network created for each time-step and connected to the previous and next time-steps, representing the RNN as a big feed-forward ANN. It calculates the gradients for the weights in this network, with a caveat: the weights for each mirror are the same, and the gradients for these weights are added up before updating these weights.

3.1.3 Generalisation and Compositionality

The main strength of MTRNN when compared with other RNN is their ability to generalise to new sequences, being able to compose them from the sets of primitives that it learned. This behaviour is analogous to how humans behave: An adult doesn't have to learn every single combination of words to be able to produce novel sentences. The compositional behaviour of these networks is linked to their hierarchical structure, where the primitives

learned in the FC layer are combined into novel sequences.

This behaviour is well-studied in literature. Hinoshita et al. (2011) demonstrated such compositionality in language, where the I/O layer would encode characters, the FC layer encoded words and SC layer encoded full sentences. In a similar way, Yamashita & Tani (2008) showed that MTRNN models for motor action learning would encode motor primitives on the FC layer, again exhibiting compositional behaviour. In both cases the compositionality of the model was attributed to its hierarchical structure.

3.2 Bidirectional training experiments

The presence of mirror neurons, and the fact they activate both when performing an action, observing an action or talking about an action suggest multiple pathways of information to those neurons. While these connections can be modelled by fully connected ANN models, the multiple flow of information has not been tested. One way to represent such a flow is to have a model trained both to categorise and to generate an action.

In the works of Heinrich et al. (2013) it was shown that MTRNN models usually have extremely weak backward connections, suggesting the networks are not being used to their full potential. This makes MTRNN a prime candidate for bidirectional training, exploiting these backward connections to propagate information back through the model.

Following on the idea of MTRNN, we intend to train the model in order to solve both tasks: generate an output in the I/O layer given a certain CS, but also to generate the CS given a certain input in the I/O layer. In addition to this, we expect the model to display its compositional behaviour also when generating CS.

In this section we will discuss two experiments conducted with MTRNN models: a language learning experiment and a motor action learning experiment. These experiments were published in Antunes et al. (2018).

3.2.1 Data representation

Language Data

The language dataset we use is comprised of combinations of 9 verbs with 9 objects, for a total of 81 sequences. The verbs and objects used can be visualised on Table 3.1.

Table 3.1: List of verbs and objects present in the dataset.

Verbs	Objects
slide left	tractor
slide right	hammer
touch	ball
reach	bus
push	modi
pull	car
point	cup
grasp	cubes
lift	spiky

Each sequence is 30 time-steps long, and each time-step corresponds to a single letter activation, in a one-hot encoding. the initial 4 time-steps correspond to the empty space character to allow the information to propagate through the network. The one-hot vector is 28 elements long, with 26 elements dedicated to the letters in the English alphabet, with 2 additional elements for empty space and full stop. A visual representation of this data can be seen in Fig. 3.3.

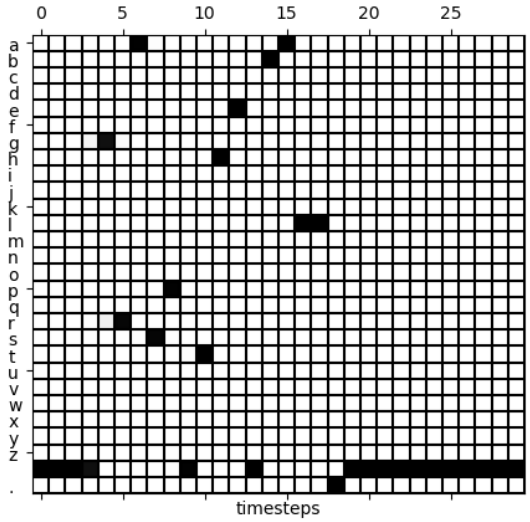


Figure 3.3: Representation of the input/output for the language learning scenario. Each column represents a distinct time-step, and each line the respective letter. Active letters are indicated in a darker colour.

Motor Action Data

The motor action dataset used in this experiment was collected by Peniak et al. (2011a) and it is composed of 432 sequences. These sequences represent 9 different actions performed on 9 different objects in 6 different positions, making a total of 486 sequences excluding

54 sequences intentionally excluded corresponding to the combinations of 3 actions on 3 objects over all 6 positions. The included and excluded combinations can be visualised on Table 3.2.

Table 3.2: Combinations of verbs and objects present in the dataset.

	tractor	hammer	ball	bus	modi	car	cup	cubes	spiky
slide left	✔	✔	✔	✔	✔	✔	✔	✔	✔
slide right	✔	✔	✔	✔	✔	✔	✔	✔	✔
touch	✔	✔	✔	✔	✔	✔	✔	✔	✔
reach	✔	✔	✔	✔	✔	✔	✔	✔	✔
push	✔	✔	✔	✔	✔	✔	✔	✔	✔
pull	✔	✔	✔	✔	✔	✔	✔	✔	✔
point	✘	✘	✘	✔	✔	✔	✔	✔	✔
grasp	✔	✔	✔	✘	✘	✘	✔	✔	✔
lift	✔	✔	✔	✔	✔	✔	✘	✘	✘

Each sequence in this dataset is comprised of 43-element vectors over 100 time-steps. The 43 elements correspond to the state of the robot, the encoding of the verb and the encoding of the object. The robot used was an iCub robot, and we use 41 elements to encode its joint positions: 3 for the torso, 3 for the neck, 3 for the eyes and 16 for each arm. The position of the object is given by the head and eyes joints, since the robot follows the object. The encoding of the verb and object was done in windows of 0.1, where a value between 0.1 and 0.2 for the verb would correspond to "slide right" and the same value for the object would translate into "hammer". The full list of this encoding can be seen on Table 3.3.

Table 3.3: Encoding of verbs and objects in the dataset. One element is used for verbs and one for objects, and they can range between 0 and 1. The objects/verbs are encoded in intervals of 0.1 in these elements.

Element Value	**Verbs**	**Objects**
$[0.0 - 0.1[$	slide left	tractor
$[0.1 - 0.2[$	slide right	hammer
$[0.2 - 0.3[$	touch	ball
$[0.3 - 0.4[$	reach	bus
$[0.4 - 0.5[$	push	modi
$[0.5 - 0.6[$	pull	car
$[0.6 - 0.7[$	point	cup
$[0.7 - 0.8[$	grasp	cubes
$[0.8 - 0.9[$	lift	spiky

The vectors of the robot joints are normalised to the maximum values of each individual joint. This is both a safety precaution as well as an implementation aide, since it prevents the robot from trying to achieve positions that would break its joints while also allowing the model to converge faster. The values in the vector are continuous between 0 and 1.

A plot of some joint positions over time can be seen in Fig. 3.4. A picture of the iCub performing an action from the dataset can be seen in Figure 3.5.

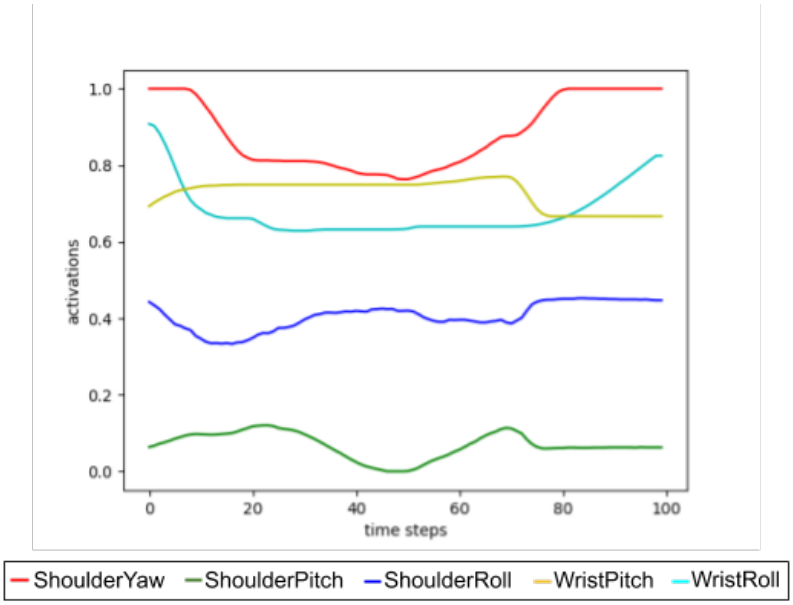


Figure 3.4: Examples of the joint positions over time for an action. These plots correspond to the yaw, pitch and roll of the right shoulder, as well as the pitch and roll of the right wrist. The action represented is "slide left the cup", when the cup is on the 4th position, slightly to the right (sequence number 40).

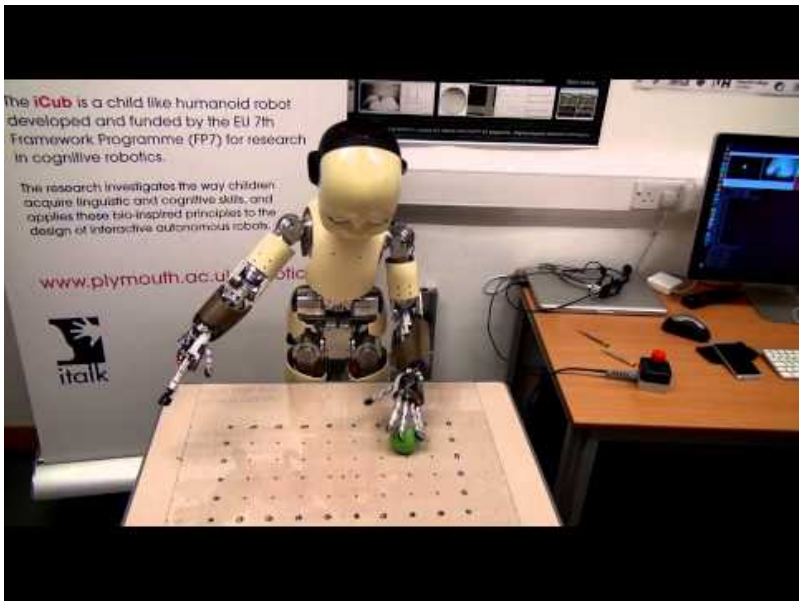


Figure 3.5: Example of iCub robot touching a ball, as used by Peniak et al. (2013).

Control Sequences

Both sources of data mentioned previously are linked to the CS used to represent them. These CS are defined as 8-element vectors, split into 2 segments of 4 elements: the first segment encodes the verb and the second segment the object. The encoding was done in

a binary way, e.g. [0,0,1,1] would correspond to the verb "touch" while [0,1,0,0] would encode the verb "reach". The encoding is arbitrary, with the only requirement being that each verb and object is linked to its particular encoded CS. For this experiment, we used the encoding presented on Table 3.4.

Table 3.4: CS encoding for the verbs and objects. The 8 element vector is split into two 4-element vectors which encode the respective verbs/objects as shown.

4-element segment	Verbs	Objects
[0,0,0,1]	slide left	tractor
[0,0,1,0]	slide right	hammer
[0,0,1,1]	touch	ball
[0,1,0,0]	reach	bus
[0,1,0,1]	push	modi
[0,1,1,0]	pull	car
[0,1,1,1]	point	cup
[1,0,0,0]	grasp	cubes
[1,0,0,1]	lift	spiky

An example of a full CS for the sentence "pull the cubes." would then come as [0,1,1,0,1,0,0,0].

3.2.2 Model description

The model used for this experiment is a traditional MTRNN with 3 layers. The SC layer has 45 neurons and a timescale factor τ of 60 and the FC layer has 160 neurons and a τ of 5. The I/O layer will vary depending on the type of input: for the language learning experiment it has 40 neurons while for motor action learning it has 140, in both cases it has a τ of 2. The I/O layer for the motor action case was chosen to be bigger in order to deal with the more complex data. The timescale factors τ were chosen from the literature (Yamashita & Tani (2008); Peniak et al. (2011a); Zhong et al. (2016); Heinrich et al. (2013); Zhong et al. (2014, 2016)) and tested within a small range of variations, with these values being found acceptable for the task. The model can be visualised on Fig. 3.6.

In this project we decided to use character inputs instead of word encoding. While the later would reduce the complexity of the model, possibly making it easier to train, it has the drawback that in order to teach new words to the model it would require changing the model itself; instead, we opt to use characters, which allows the model to learn any possible word limited, in this case, to the English alphabet. The output of the network in the language scenario is put through a softmax layer that calculates the probability distribution of the letters, with the most likely letter being chosen for each time-step. In

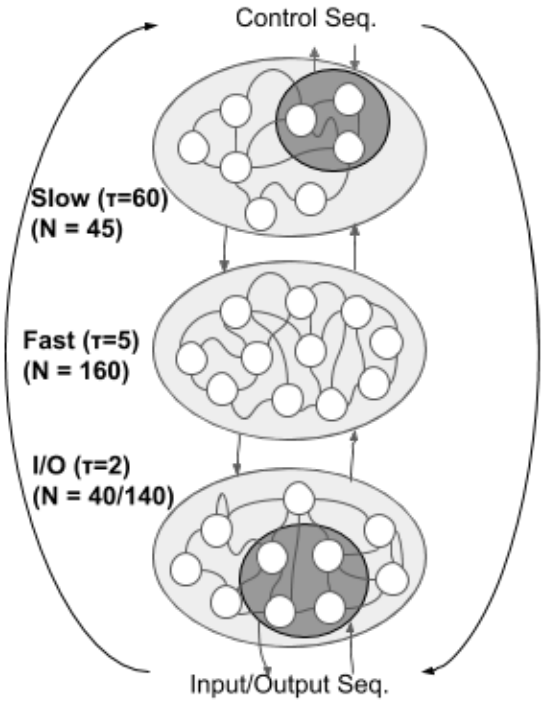


Figure 3.6: MTRNN model used for the bidirectional training experiments, with indication of timescale factors τ and number of neurons **N**. The I/O layer has a different value for the language learning scenario (N = 40) and for the motor action learning scenario (N = 140).

the motor action scenario we obtain the output vector directly from the network, since the values are already normalised by the model.

Finally, in the present model there is still a limitation due to the CS binary encoding, which will be addressed in the full integrated model in Section 3.2.7 and Chapter 4.

3.2.3 Training

In traditional settings MTRNN are trained in a single direction, that is, they are usually provided with CS inputs that generate an output at the I/O layer, which is then compared to the desired output. The resulting error is used to calculate the gradients that are then propagated through the network using BPTT. This means, in short, that we only have to optimise for a single loss function. In the case of bidirectional training, however, the same model will be trained to generate the CS and the output (sentences or motor actions), meaning optimising two different loss functions.

In this experiment we deal with different natures of data: a sentence, translated into single letter activations spread over time; a motor action, with a vector of normalised values representing the robot joints at different time steps; and a CS, corresponding to a single output at the very final time step. To account for this, we create two different training

modes, one for the outputs and one for CS, and we alternate them during training. In order to speed up the training we focus the direction of training that returns the highest error; if the calculated loss for the direct flow of information becomes lower than the loss for the inverse flow, within a fixed range, we switch the training modes to focus on the inverse flow instead. Additionally, if one of the training modes hits the stopping threshold, the training algorithm will focus on the opposite direction until both losses are below the corresponding thresholds. The fixed range between the losses was set to be 1.5 times the loss of the current direction, and was tuned experimentally. The language loss function is defined as the cross-entropy between the output of the softmax and the one-hot vector. For the generation of CS or motor actions we use the mean-squared error to compute the loss.

The loss landscape of these models, particularly when trained in a bidirectional way, is extremely irregular and sometimes leads to exploding and/or vanishing gradients. In order to account for the exploding gradients we clip the gradients in an interval of $[-7.0, 7.0]$, not only preventing the problem of exploding gradients but also preventing too strong oscillations when changing the direction of training. For the optimiser we used AMSGrad from Reddi et al. (2018), with an initial learning rate of 0.0005.

3.2.4 Language Learning

In this first experiment we trained and tested the MTRNN model to generate a CS from a certain sentence and vice-versa. We set a maximum number of iterations at 80000, with an early stopping threshold loss of 0.012 for language generation and 0.0001 for CS generation. The network attains these values between 35000 and 45000 iterations.

Testing on training data

The first test we performed on the model was the ability to reproduce correctly all the training data. We tested sentence generation by using CS inputs and testing the output. The model was able to reproduce all 81 sentences successfully.

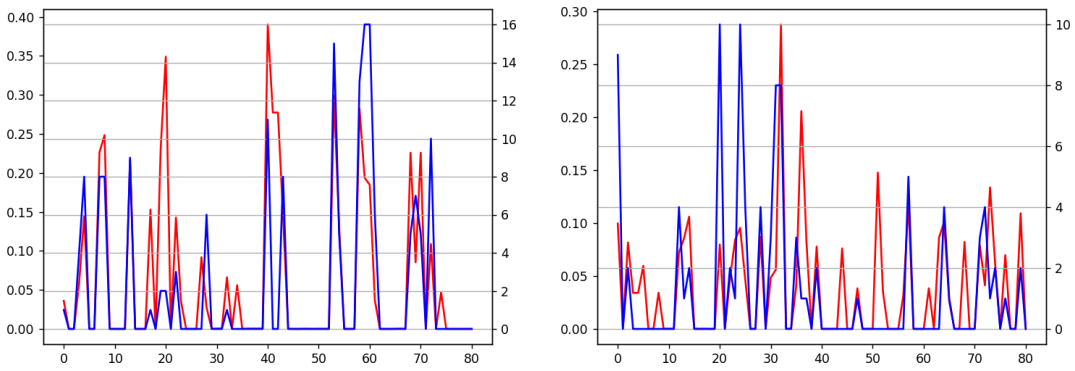
In the opposite direction, when generating CS, the model is able to correctly generate them with an error on the order of magnitude of 10^{-3}.

Table 3.5: Examples of the expected (left) and generated (right) sentences for the generalisation tests. Erroneous letters are shown in bold and underlined. Most errors happen on the object spelling.

lift the hammer.	lift the **caae.**
push the cubes.	push the **thhcccee.**
grasp the bus.	gra__ _**h**h **muu**..
grasp the modi.	gras_ the **o**o**i**i.
point at the hammer.	point at th**hbb**a**a.**

Testing for generalisation ability

In a second step we tested the model for its generalisation ability, by running a 9-fold cross-validation test. In each test 9 sentences were removed at random from the dataset, and the tests were ran 9 times, covering the entire dataset.



(a) (b)

Figure 3.7: Error plots for the 9-fold cross-validation tests. In 3.7a we have the plot for the first experiment (included in Antunes et al. (2018)) and in 3.7b we have the new test run. Red represents the error on the generation of CS (left axis), for each sentence. Blue represents the number of erroneous letters (right axis) for each sentence.

The model was able to generalise for 57 out of 81 cases without any mistakes when generating sentences. Mistakes on the remaining cases were usually on the object side, with the verb being correct, as can be seen on Table 3.5. For the CS, the errors are similar, with good results for the same 57 sentences. These results are illustrated in Fig. 3.7a. It evaluates the error on sentence generation by checking the number of erroneous letters on the output, while for control sequence generation it computes the average error over all 8 cells of the output vector.

To better understand the results we ran the test once more, and printed the sentences and CS generated for all removed sentences. These results can be seen on Table 3.6. This test reported 53 out of 81 sentences correctly reproduced, and again most mistakes were

located on the object side of the sentence, sometimes spreading to the articles ("the" and "at"). We can see the activations for one such mistaken sentence corresponding to "slide right the cubes." in Fig. 3.8.

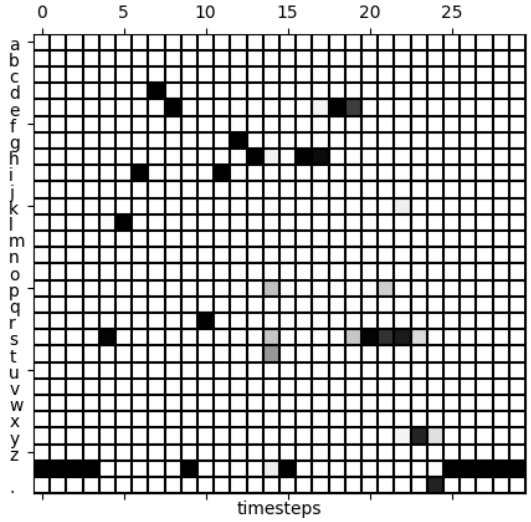


Figure 3.8: Matrix of the letter activations for the case "slide right the cubes.". We can see that the activations are very well defined for the verb, but not so for the object, where the activation is spread over different letters (in grey colors).

On the CS side, we saw again that in general the network was able to reproduce correctly the CS for the new sequences (see Fig.3.7b), and the higher errors on CS generation correspond to the same errors in sentence generation, with the highest error happening on the object encoding side of the CS. An example of this can be seen in the sequence with the highest error for CS generation shown on Table 3.7, corresponding to the sentence "slide right the cubes." with the output "slide right hheesssy.".

3.2.5 Motor Action training

On this second experiment we trained the MTRNN model on motor action data, again testing if it could generate a motion from a certain CS and vice-versa. Due to the dynamics of the model, we allow 4 time-steps in the beginning of each sequence, in addition to the 100 time-steps that describe the motion, for a total of 104 steps. These extra steps allow the information of the initial position of the robot to propagate through the model. Only the 41 cells corresponding to the robot joints and the 1 cell corresponding to the object encoding are used for this model; the verb encoding is discarded.

Instead of training the network with the full set of sequences we created batches of 32

Table 3.6: List of sentences generated during the k-fold test, covering the entire dataset. In bold and underlined are the mistakes in the output sentences compared to the ground truth.

Ground truth	Output		Ground Truth	Output
slide left the tractor.	slide left the tractor.		push the modi.	push the modi.
slide left the hammer.	slide left the hammer.		push the car.	push the car.
slide left the ball.	slide left the ball.		push the cup.	push the cu**u**.
slide left the bus.	slide left the bus.		push the cubes.	push the cube**e.**.
slide left the modi.	slide left the modi.		push the spiky.	push the spiky.
slide left the car.	slide left the ca**a**.		pull the tractor.	pull the **bolitor**.
slide left the cup.	slide left the**e**cup.		pull the hammer.	pull the hammer.
slide left the cubes.	slide left the cubes.		pull the ball.	pull the **tams**.
slide left the spiky.	slide left th**hesssiiiyy.**		pull the bus.	pull the bus.
slide right the tractor.	slide right the tractor.		pull the modi.	pull the modi.
slide right the hammer.	slide right the hammer.		pull the car.	pull the car.
slide right the ball.	slide right the ball.		pull the cup.	pull the cup.
slide right the bus.	slide right the bu**u**.		pull the cubes.	pull the cubes.
slide right the modi.	slide right the modi.		pull the spiky.	pull the spiky.
slide right the car.	slide right the **haro**		point at the tractor.	point at the tractor.
slide right the cup.	slide right the cup.		point at the hammer.	point at the hammer.
slide right the cubes.	slide right **hheesssy**.		point at the ball.	point at the ball.
slide right the spiky.	slide right the spiky.		point at the bus.	point at th_ bu**ll**.
touch the tractor.	touch the tractor.		point at the modi.	point at the modi.
touch the hammer.	touch the hammer.		point at the car.	point at the car.
touch the ball.	touch the **mamll**.		point at the cup.	point **th uu.**
touch the bus.	touch the bu**l.**.		point at the cubes.	point at the cubes.
touch the modi.	touch the**foui.**.		point at the spiky.	point at the spiky.
touch the car.	touch the **ba.e**.		grasp the tractor.	grasp the tractor.
touch the cup.	touch the **b**up.		grasp the hammer.	grasp the hammer.
touch the cubes.	touch the **aallbll.**		grasp the ball.	grasp the ball.
touch the spiky.	touch the spiky.		grasp the bus.	grasp the bus.
reach the tractor.	reach the **_ood.**.		grasp the modi.	grasp the **sooi**.
reach the hammer.	reach the**ehalle.**.		grasp the car.	grasp the car.
reach the ball.	reach the ba**am**.		grasp the cup.	grasp the cu**r**.
reach the bus.	reach the**m**bus.		grasp the cubes.	grasp the **_ussss**
reach the modi.	reach the modi.		grasp the spiky.	grasp the **co**iky.
reach the car.	reach the car.		lift the tractor.	lift the tractor.
reach the cup.	reach the cup.		lift the hammer.	lift the hammer.
reach the cubes.	reach the cu**u**es.		lift the ball.	lift the ball.
reach the spiky.	reach the spiky.		lift the bus.	lift the bus.
push the tractor.	push the tractor.		lift the modi.	lift the modi.
push the hammer.	push the hammer.		lift the car.	lift the car.
push the ball.	push the **hams**.		lift the cup.	lift the cup.
push the bus.	push the bus.		lift the cubes.	lift the cubes.
			lift the spiky.	lift the spiky.

Table 3.7: Example of CS generation corresponding to the highest error. We can see that the error is higher for the object encoding segment while it remains relatively low for the verb.

Verb				
Ground Truth	0	0	1	0
Output	0.00118441	-0.02124905	1.0120379	-0.00754238
Object				
Ground Truth	1	0	0	0
Output	0.3218979	0.7065416	0.32312515	0.55215436

sequences. At each iteration we feed a random batch to the model.

For this training we set a maximum number of iterations of 250000, higher than before due to the increased number of sequences and complexity of data. We set the early stopping thresholds at 0.05 for motor action generation and 0.0001 for CS generation. The model never crossed the threshold for motor actions, achieving the lowest loss of approximately 0.7 at iteration 226180. This model was used for the experiment on Antunes et al. (2018).

Testing on training data

As for the language scenario, we tested the model for the correct generation of the output. To test this we calculated the difference between the desired output and the actual output at each time-step, for each joint and for each action. We then plotted the error corresponding to the combination action-position that exhibited the highest error. These plots can be seen in Fig. 3.11.

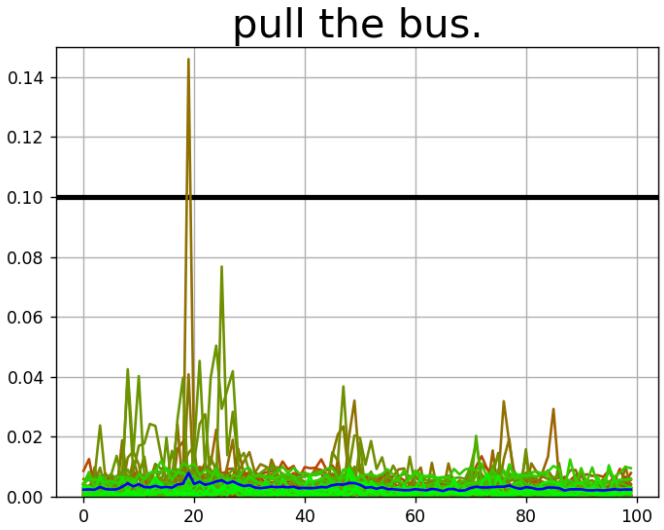


Figure 3.9: Plot of the highest-error action "pull the bus". Each curve corresponds to the error of the 41 sized output vector. We can see that very few neurons have an error higher than 0.1 (thick black line), and then only for very few timesteps. The mean error (blue line) is below 0.01. Best seen in color.

The errors for each individual joint was kept low (<0.03) with sporadic spikes in the error (<0.2) lasting a single time-step. The highest error obtained was in the action "pull", shown in Fig. 3.9. As before, the error has short-term spikes occurring very sporadically. In order to study these errors we explore the trajectories of the joints with spikes. We can see such a case in Fig. 3.10. We can easily verify that the error is due to a small delay around time-step 20, where the output has some difficulty following the sudden increase

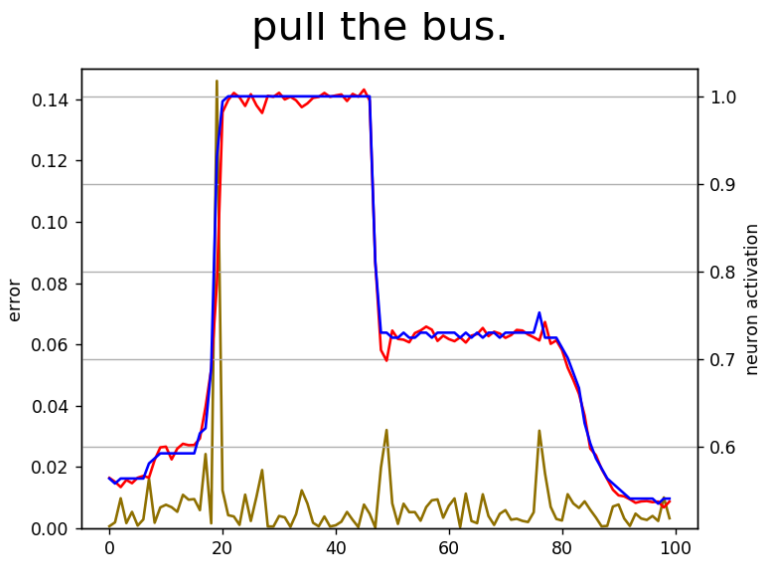


Figure 3.10: Plot of the trajectory corresponding to the highest error value, neuron 35. Blue corresponds to the target trajectory, red to the output of the network. Gold indicates the error for each time-step. Notice the spike before step 20, and the actual effect on the trajectory.

in the joint position, but is otherwise capable of following the correct trajectory.

As in Section 3.2.4, we test also the CS generation. The network was able to generate correct CS with a error of the order of magnitude of 10^{-3}. An example of such a CS can be seen on Table 3.8.

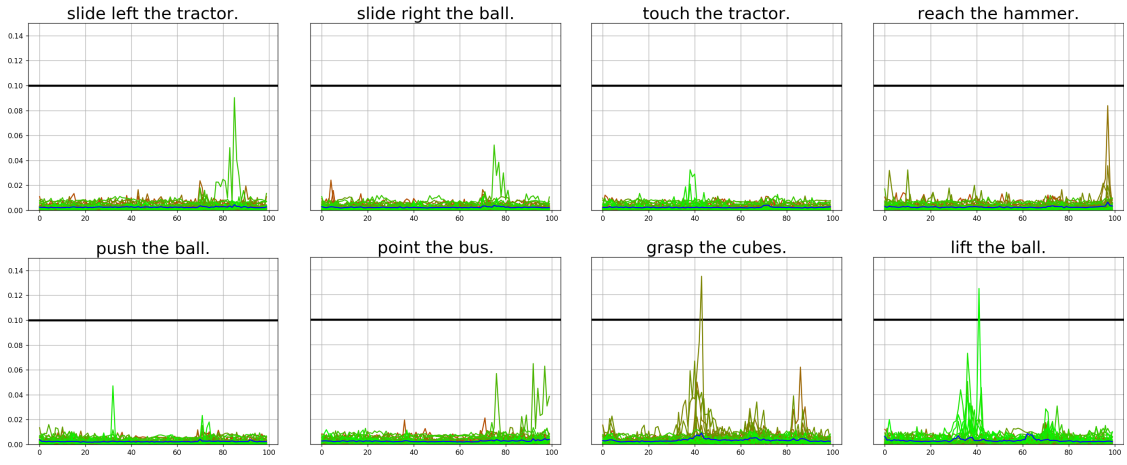


Figure 3.11: Error plots for sequences corresponding to the highest error (for each action). The action "pull" is shown in Fig. 3.9. Each curve corresponds to the error of the 41 sized output vector. Vertical axis indicates the error, with the black line pointing to an error of 0.1. The mean error is represented by the blue line, and is always below 0.01.

Table 3.8: CS generated by the motor action input, with target CS for comparison. The error is very small (<0.006). The action vector [0,0,0,1] corresponds to the verb "slide left" while the object vector [0,0,0,1] corresponds to the object "tractor"

Action				
Ground Truth	0	0	0	1
Output	5.37800378e-03	2.43033865e-04	1.05508996e-04	1.00041854
Object				
Ground Truth	0	0	0	1
Output	-9.21239727e-04	4.72593513e-03	4.09128651e-03	1.00172555

3.2.6 Testing of preliminary integrated architecture

The main motivation behind this work is the creation of a system that links actions and language, being able to generate a motor action given a certain sentence and vice-versa. While the experiments mentioned on Sections 3.2.4 and 3.2.5 provided evidence that these models are able to learn such bidirectional tasks, they were still trained in motor actions and language separately.

As a first test of the system we tried linking the two models at the CS level, creating a single pipeline. Since the CS are predefined, there are still many limitations to the model due to external influences (e.g. a limit to the number of sentences it can learn due to the encoding in the CS). Even so, it is important to test the effect of such a link before integrating the full model.

To perform this test we used the iCub simulator Tikhanoff et al. (2008) to execute the motor action and do a qualitative evaluation on its performance. We provided a sentence to the language branch which then generated the corresponding CS. This CS was then used as input to the motor action branch, thus generating the corresponding motor action. The robot state inputs fed back to the model were the actual joint values taken from the simulator, thus also supporting the robustness of the system when used with an actual robot. The integrated model was able to generate the correct motor action for the sentence, completing the link between language and motor actions. The video of this experiment can be seen on the link: https://youtu.be/p-In7eaCZmA.

As a final test, we ran this model on the real robot. Due to the constraints of the robot controllers, we couldn't specify a specific time value for the time-steps; instead, the model has to wait for the controllers of the robot to finish the movement to the previous joint position, get the current position, feed it to the model, get the output for the next time-step,

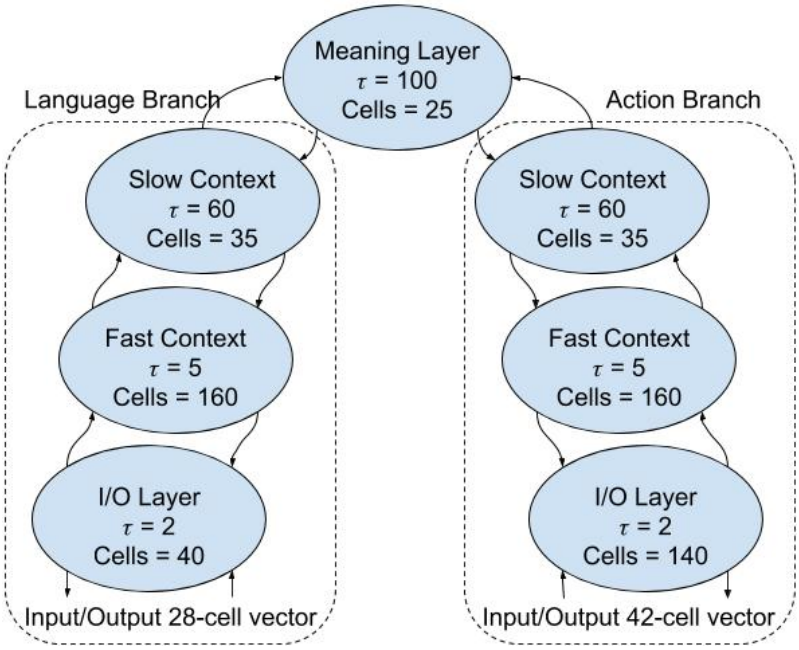


Figure 3.12: Illustration of the full integrated MTRNN model for language and motor action learning.

and repeat this process. Unfortunately this leads to a very rough movement which is undesirable - however, the robot was still capable of performing the requested action.

3.2.7 Full Integrated Model

With the satisfactory results of the experiments performed in 3.2.4 and 3.2.5, and the results of a integrated model on a real robot, we proceed to integrate everything under a single model, still using the MT structure and the bidirectional training. To connect the motor action and language branches we use another CTRNN layer, defined in this thesis as *meaning layer*, that will link to the SC layer on both branches, and have a higher timescale factor. An illustration of the full model can be seen in Fig. 3.12.

With the meaning layer we remove the last source of external limitations: the encoding of the CS. Without a specific encoding of the CS, the model is theoretically able to learn any number of combinations, limited only to its internal parameters.

Training of the full model

In the experiments with a single branch, when testing an integrated model, we considered the following chain of events: i) feed language input to the language model; ii) generate the corresponding CS; iii) feed the generated CS to the motor action model; and iv) generate

the corresponding motor action. This chain was trivial to implement with the two separate branches, since the dynamics of one would not affect the other. In a full integrated model, however, this is not the case: while processing the sentence input, it is not immediately clear that this will not have an influence on the motor action branch, and vice-versa.

In order to account for this, special care must be taken with how the data is fed to the model during training. When training motor action generation from a sentence, it must first input the sentence to the model, and only after it is completed should it start generating the motor action; this ensures that the model has time to encode the *meaning* of the sentence (or motor action) on the meaning layer before starting to generate a motor action.

We have adapted our training to achieve this: instead of running an experiment for 30 or 100 time-steps, it instead considers the full 130 time-steps. The way to feed the inputs and compare to the desired outputs depends on the source of data: when generating a motor action, it feeds the language input on the first 30 time-steps; at time-step 30 it starts comparing the output on the motor branch with the motor action desired output. In the case of sentence generation, the reverse happens: it first feeds the motor action for 100 time-steps, and then compares the last 30 steps to the desired sentence. During the initial time-steps corresponding to the inputs the outputs are discarded.

Outcome

The model did not converge on a particular solution, with the loss never decreasing below an order of magnitude of 10^2. Reducing the size of the data, both in time-steps and in size per time-step did not influence the training.

An analysis of the gradients during training provided insight into the issue: vanishing gradient. Given the depth of the model (7 layers), the calculated gradients were too small to have an effect on training.

Several trials and changes had no effect - only a reduction in either depth or size of the layers had any effect on the gradient. This was, unfortunately, followed by poor accuracy or inability to learn the data.

3.2.8 Conclusion

The experiments in 3.2.4 and 3.2.5 have shown that the MTRNN models work for both language and motor action learning, being able to reproduce correctly all the data they were trained with, and to generalise to most tested sentences. The models were trained in a bidirectional way, learning to generate the corresponding output (language or motor action) from a CS and vice-versa.

We ran an preliminary integrated model on the simulated and real robot, showing that these models solve the problem of linking language and motor actions - however, this was achieved with the use of CS which were arbitrary and which encoding was predefined, thus limiting the model in terms of number of sequences it can learn.

Finally we designed a full integrated model and tested it on the data, connecting directly language and motor actions. Unfortunately the model was too deep and faced the common issue of vanishing gradient, preventing the successful training of the model. In previous works using regular Recurrent Neural Networks this problem was avoided by using LSTM, whose internal neuron structure avoids the issue. A comparison with these models, along with a solution that avoids the vanishing gradient, is shown in Chapter 4.

Chapter 4

Bidirectional Multiple Timescales LSTM Model for Grounding Language and Robot Actions

Key points:

- Multiple Timescale Long Short-Term Memory (MTLSTM)
  - Continuous Timescale LSTM (CTLSTM) cells
- Bidirectional training experiments
  - Language learning
  - Motor action learning
  - Comparison with baselines - LSTM and auto-encoder-like architectures

In the experiments described on Section 3.2.7 we have implemented a model that fully connects language and motor actions. The attempts at training this model, however, faced the issue of vanishing gradients, thus being unable to learn correctly how to make the link. While MTRNN are incapable of dealing with vanishing gradient due to the nature of its neurons, its architecture could be made to work provided a new set of neurons is used.

From the literature we can see that a particular model was developed specifically to deal with vanishing gradients: LSTM (Hochreiter & Schmidhuber (1997)). LSTM keep an internal memory cycle that is not affected by the weights of the model, therefore simplifying the transfer of information in very long sequences. The drawback of these models is the number of parameters in the cells, where a LSTM cells has 4 parameters for

a regular RNN neuron single parameter. A simplified version, called Gated Recurrent Unit (GRU, Cho et al. (2014)), which has a reduced number of parameters when compared to the LSTM cell.

As was discussed previously, however, LSTM have presented some difficulty when dealing with the compositional nature of data. One way to possibly deal with these issues is to merge the idea of Multiple Timescales with LSTM. MTLSTM are virtually unexplored, with a first instance developed by Liu et al. (2015) and based on clockwork RNN (Koutnik et al. (2014)), and a second implementation by Yu et al. (2017) with a more continuous, MTRNN-like behaviour. An implementation with GRU was developed by Kim et al. (2016) that was also proven to perform better than simple LSTM. The works from Yu et al. (2017) and Kim et al. (2016) show the importance of a hierarchical structure in these RNN, while using LSTM cells.

In this chapter we describe an implementation of MTLSTM, adapting LSTM to include a timescale factor as in MTRNN. We will present some experiments showcasing its ability to learn the training data in a bidirectional way and generalise to novel sequences.

4.1 Multiple Timescale Long Short-Term Memory

A MTLSTM is structured much like MTRNN, with the exception that its neurons are no longer vanilla recurrent neurons with a timescale factor τ but rather an adapted LSTM cell, which we will call Continuous Timescale Long Short-Term Memory (CTLSTM) cell. This is a novel model, with a different design of the CTLSTM cell than the work of Yu et al. (2017), and a completely different implementation from Liu et al. (2015) where the "frame-rate" of the inputs to the layers was changed, rather than a continuous timescale factor implementation. In addition, the model presented in this chapter is more complex than the models presented in Yu et al. (2017) and Liu et al. (2015), with 7 layers compared to 2 and 3, respectively.

4.1.1 Continuous Timescale Long Short-Term Memory cells

CTLSTM cells are the basic unit for MTLSTM. While inspired in the neurons from CTRNN, the inclusion of LSTM into the model means moving away from its original bio-inspiration. LSTM cells (Fig. 4.1) consist on 4 different information pathways: i) the input, responsible

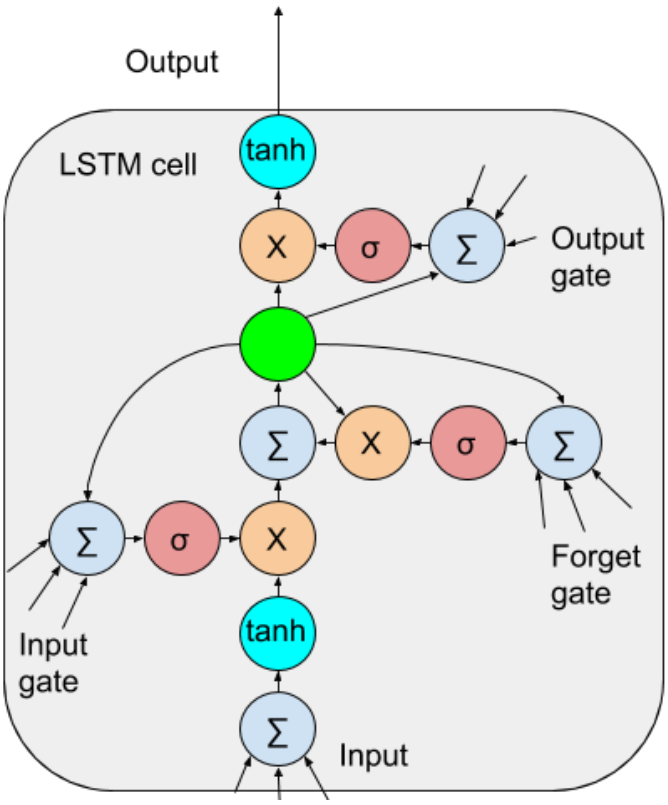


Figure 4.1: Illustration of a basic LSTM with peepholes. Each gate corresponds to a sum (light blue) followed by a sigmoid function (light red) and the multiplication (dark yellow). Tanh is the activation function for the inputs/outputs. The central cell (green) is the memory cell.

for adding up the different inputs to the cell; ii) the input gate, controlling how much of the input should go into the cell ; iii) the forget gate, controlling the amount of information that should come from memory; and iv) the output gate, calculating the output of the cell. The LSTM cells used as basis for the CTLSTM are named "peepholes" LSTM due to the connections from the memory cell to the inputs to the different gates.

One of the important aspects of CTRNN neurons and the main reason for their bio-inspiration is the timescale factor. While it is trivial to implement the RNN as seen in 3.1.2, the same cannot be said when adapting an LSTM. One way of keeping the timescale behaviour is to apply the timescale factor to the cell internal memory, thus enforcing the "leaky" aspect to the model.

In the MTRNN, we considered the output of the neuron pre-activation function to be the memory of the neuron. In keeping with that implementation we use the output as the memory for this cell, using it in the LSTM. An illustration of this can be found in Fig. 4.2. As in vanilla LSTM, the cell has the power to choose how much of this internal memory should be used for generating the output of the cell, based on its inputs; however, the

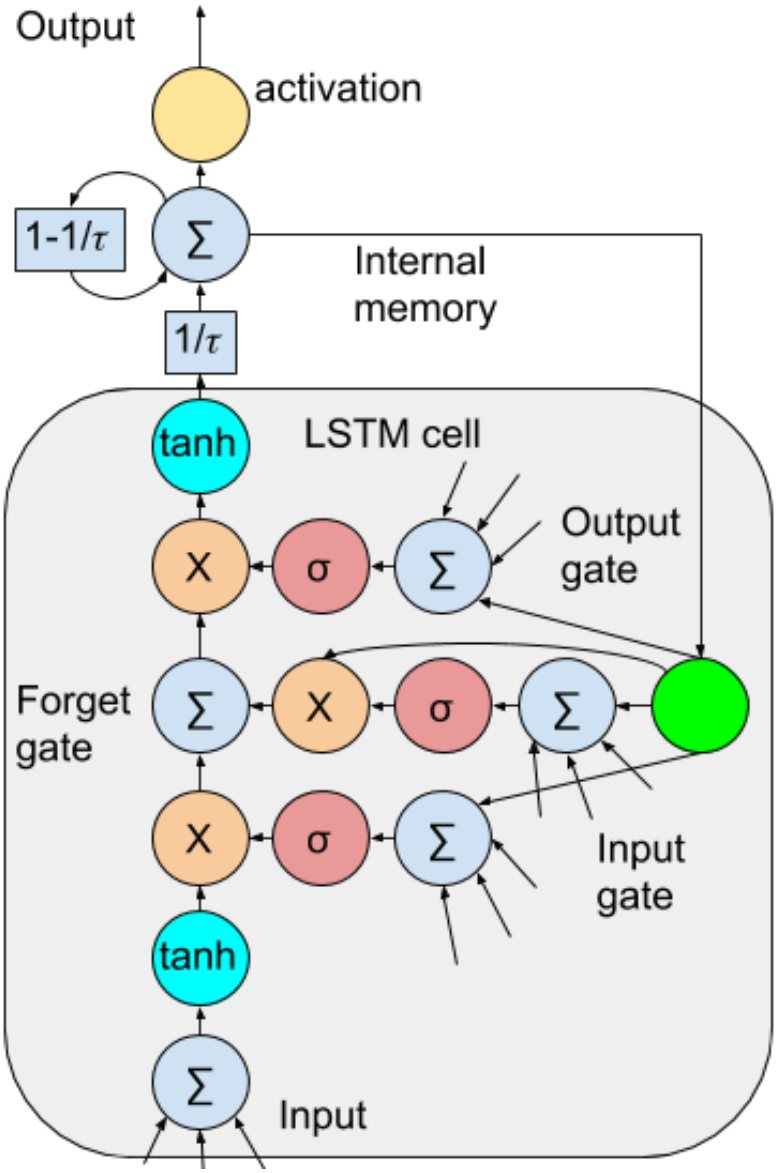


Figure 4.2: MTLSTM implementation in this work. Notice the feedback connection into the LSTM cell memory (green).

output of the entire CTLSTM always considers the timescale factor, changing the internal memory slowly if the timescale factor is large, or faster if it is small. This implementation allows the cell to control its behaviour more than CTRNN neurons, since the LSTM cell can completely inhibit any inputs or, on the contrary, keep the memory of the cell unchanged. The handling of the memory of the cell is the main difference to the model presented by Yu et al. (2017), where one cell could be described as a LSTM followed by a timescale RNN neuron.

The following equations describe the forward propagation on a CTLSTM cell:

$$y_l^t = \sigma(\sum_i^I w_{il} x_i^t + \sum_c^C w_{cl} s_c^{t-1}) \tag{4.1}$$

$$y_f^t = \sigma(\sum_i^I w_{if} x_i^t + \sum_c^C w_{cf} s_c^{t-1}) \tag{4.2}$$

$$y_o^t = \sigma(\sum_i^I w_{io} x_i^t + \sum_c^C w_{co} s_c^{t-1}) \tag{4.3}$$

$$u_i^t = \tanh(\sum_i^I w_i x_i^t) \tag{4.4}$$

$$y_c^t = (u_i^t y_l^t + s_c^{t-1} y_f^t) \tag{4.5}$$

$$y_{lstm}^t = \tanh(y_c^t y_o^t) \tag{4.6}$$

$$s_c^t = (1 - \frac{1}{\tau}) s_c^{t-1} + \frac{1}{\tau} y_{lstm}^t \tag{4.7}$$

$$y_c^t = g(s_c^t) \tag{4.8}$$

Equation 4.1 defines the behaviour of the input gate, with y_l^t corresponding to the output of the input gate at time-step t, w_{il} are the weights on the inputs, w_{cl} are the weights on the cell previous state, x_i^t corresponds to the inputs to the cell, including recurrent connections, and finally s_c^{t-1} is the cell state from the previous time-step, with σ reflecting the sigmoid activation on the gate. Equation 4.2 defines the behaviour of the forget gate with w_{if} being the weights on the inputs and w_{cf} the weights on the cell state. Similarly, equation 4.3 describes the behaviour of the output gate, with w_{io} representing the weights on the inputs and w_{co} the weights on the cell state. The new input itself is defined in equation 4.4, with w_i^t representing the weights on the inputs. The output of the LSTM proper is then calculated with equations 4.5 and 4.6. The output of the LSTM then goes through the continuous timescale factor τ, as described in equation 4.7, where the new internal cell state s_c^t is calculated. The output of the CTLSTM cell is then given by an

activation function g, represented by equation 4.8. For the current project we have used the activation function suggested by LeCun et al. (2012):

$$g(s_c^t) = 1.7159 \tanh(\frac{2}{3} s_c^t) \tag{4.9}$$

4.1.2 The MTLSTM architecture

The success of MTRNN relies not only on the smart design of its neurons, but also on the hierarchical architecture. In the experiments in 3.2.7 we tried extending a simple, 3-layer MTRNN to a full integrated model connection actions and language, but faced issues with vanishing gradient. Since LSTM should not suffer from vanishing gradient, we implemented the full model for these experiments, using MTLSTM instead of MTRNN.

MTLSTM models are, from the outside, similar in every way to MTRNN. They are fully connected at each individual layer, they connect only to "adjacent" layers, and they have different timescales for each of the layers. As such, we assume their modes of operation to be similar: they would be able to operate on either direction, generating an output given a CS or the opposite. In addition to this, we expect the layers to behave in a similar way: the FC layer would encode primitives for either language or motor actions, with the SC layer generating a representation for the whole sequence.

Finally, and most importantly, the model will function in a bidirectional way, generating motor actions from a given sentence and vice-versa. We use BPTT to train the network, as described in Section 3.1.2.

For the work presented in this chapter we will extend the simple 3-layer model into a full 7-layer model, with two branches corresponding to a language MTLSTM and a motor action MTLSTM, connected at the SC layer by a new CTLSTM layer called "meaning" layer. This model will be described in more detail in Section 4.2.2.

4.1.3 Generalisation and Compositionality

One of the key aspects and main reasons for choosing the MTRNN model initially is due to its ability to generalise and compose novel sequences. It is desirable to maintain these traits in the MTLSTM model, as described in 2.2.

The main hypothesis towards compositionality is that the model learns a compositional

representation of the motor action or sentence at the meaning layer, which can then be translated into the corresponding sentence or motor action, respectively. The behaviour in this case should be similar to the results obtained in 3.2.4.

4.2 Bidirectional training experiments

In 3.2 we presented some experiments with MTRNN showcasing their ability to generate an output and its corresponding CS using the same model, training it in both directions. This ability is the key point for this thesis, and therefor it is of extreme importance that this model be able to learn this bidirectional behaviour.

When comparing to the experiment in 3.2 we have additional complexity, not only due to the size of the network (7 layers vs 3 layers) but also due to the nature of the data, since no intermediate CS will be used. This model will generate a motor action directly from a sentence, and vice-versa, without any explicit prior knowledge outside of the structure of the model. These experiments will also show whether the model maintains its compositionality and generalisation behaviours on both motor actions and language.

In this section we will present some experiments conducted with the MTLSTM model, trained in a bidirectional way to connect motor actions and language. These experiments were published in Antunes et al. (2019).

4.2.1 Data Representation

The dataset used for these experiments will be the same used in 3.2.1, obtained by Peniak et al. (2011a), with a caveat: in the experiments presented on that chapter the language and motor actions were trained separately, linked to a CS. In the experiments presented here there are no CS linked to either branch, and therefore the data needs more careful manipulation to train the model.

As presented in 3.2.1 the dataset consists of combinations of 9 verbs with 9 objects in 6 different positions, for a total of 486 sequences. Not all combinations are present however, with 54 sequences, corresponding to the combination of 3 actions with 3 objects in 6 different positions, being excluded from the dataset for testing purposes. The table, shown in 3.2.1, can also be seen in 4.1 for the reader convenience.

For each sequence there are two data sources: a language sequence and a motor sequence.

Table 4.1: Combinations of verbs and objects present in the dataset.

	tractor	hammer	ball	bus	modi	car	cup	cubes	spiky
slide left	✔	✔	✔	✔	✔	✔	✔	✔	✔
slide right	✔	✔	✔	✔	✔	✔	✔	✔	✔
touch	✔	✔	✔	✔	✔	✔	✔	✔	✔
reach	✔	✔	✔	✔	✔	✔	✔	✔	✔
push	✔	✔	✔	✔	✔	✔	✔	✔	✔
pull	✔	✔	✔	✔	✔	✔	✔	✔	✔
point	✘	✘	✘	✔	✔	✔	✔	✔	✔
grasp	✔	✔	✔	✘	✘	✘	✔	✔	✔
lift	✔	✔	✔	✔	✔	✔	✘	✘	✘

On the language side, the input is a sequence of 28-cell one-hot vectors corresponding to the current active letter. Each sequence is 30 time-steps long, with the initial 4 time-steps appearing as the empty space character, in order to allow the information to propagate through the model. Each character is active for the duration of a single time-step.

On the motor action side the dataset has 43-cell vectors over 100 time-steps. These correspond to the 41 iCub robot joints, plus 1 value for verb encoding and 1 value for object encoding. For these experiments we will discard the verb encoding, but keep the object encoding. The 41 robot joints positions come from the torso (3 joints), neck (3 joints), eyes (3 joints) and arms (16 joints in each arm). The values of these joints are normalised to the extremes for each individual joint. Finally, the encoding for the objects and verbs is made on a decimal interval basis, and can be seen in Table 4.2, as in Section 3.2.1 and copied here for the convenience of the reader. The position of the object is given by the eyes and head joints, since the robot follows the object.

Table 4.2: Encoding of verbs and objects in the dataset. One element is used for verbs and one for objects, and they can range between 0 and 1. The objects/verbs are encoded in intervals of 0.1 in these elements.

Element Value	**Verbs**	**Objects**
$[0.0 - 0.1[$	slide left	tractor
$[0.1 - 0.2[$	slide right	hammer
$[0.2 - 0.3[$	touch	ball
$[0.3 - 0.4[$	reach	bus
$[0.4 - 0.5[$	push	modi
$[0.5 - 0.6[$	pull	car
$[0.6 - 0.7[$	point	cup
$[0.7 - 0.8[$	grasp	cubes
$[0.8 - 0.9[$	lift	spiky

4.2.2 Model Descritpion

For these experiments the MTLSTM model was extended to 7-layers, connecting motor actions and language. For this model we consider the two MTLSTM 3-layer branches, one for language and one for motor actions, connected at the top by a meaning layer. On the language side, the I/O layer has 40 cells and τ of 2; the FC layer has 160 cells and τ of 5; and the SC layer has 35 cells and a τ of 60. On the motor action side the I/O layer has 140 cells and τ of 2; FC layer has 160 cells and τ of 5; and the SC layer has 35 cells and τ of 60. These values were kept from the experiments performed in Chapter 3, and were inspired by previous works with MTRNN (Yamashita & Tani (2008); Peniak et al. (2011a); Zhong et al. (2016); Heinrich et al. (2013); Zhong et al. (2014, 2016)). Finally, the meaning layer connecting language and motor action branches has 25 cells and τ of 100. The τ value was chosen in order to be large enough to encode the most abstract level of knowledge, but should not be too high so the layer can still converge to a meaningful representation of this knowledge.

The language input is done character by character, as described in Section 3.2.2, with the same argument: while this increases the complexity of the model, it removes yet one more level of explicit prior knowledge, when compared to word encoding. The output is computed with a softmax layer after the I/O layer, calculating the probability distribution of the letters at each time-step. For the motor action input and output we use directly the 42-cell vector corresponding to the 41 iCub robot joints and 1 object encoding cell.

In a final comparison to the model described in 3.2.2, in the model presented in this chapter there are no CS to be learned; instead, the model learns its own internal representations of the data in the meaning layer, implicitly making the link we forced in the experiment presented in Section 3.2.7. This removes yet another source of external prior knowledge, and frees the model from the limitation imposed by the size of the CS. The 7-layer model can be seen in Figure 4.3.

4.2.3 Training

LSTM are traditionally trained in a single direction, generating a certain output sequence, optimising for a single loss function. As was proven by the experiments in Chapter 3 MTRNN can be trained in a bidirectional way, which is a critical aspect of this thesis, and

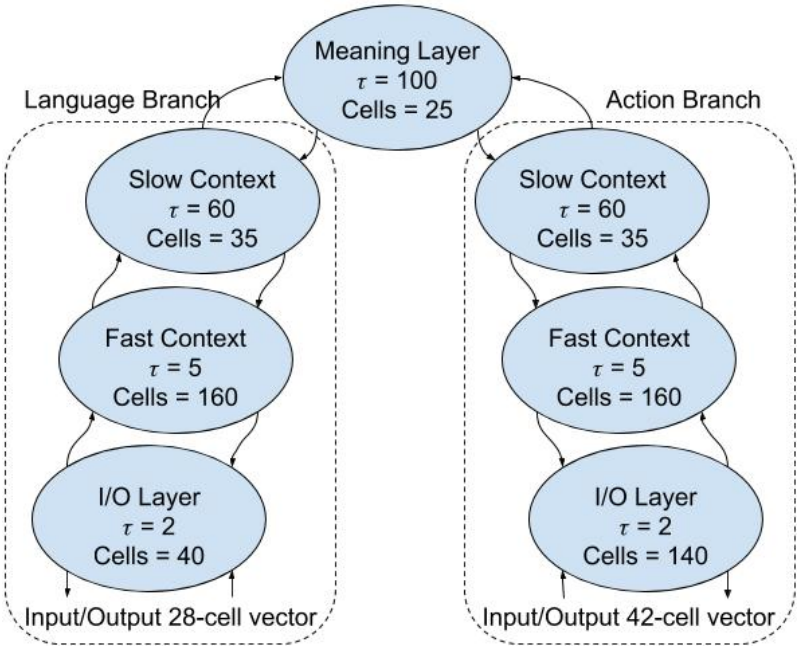


Figure 4.3: Illustration of the full integrated MTLSTM model for language and motor action learning. The structure is the same as in Section 3.2.7, with the difference being only in the implementation of the cells/neurons.

thus a particular point for MTLSTM.

While in Chapter 3 the network learned to generate a CS from a sentence or motor action, and vice-versa, in this chapter we train the model to link directly language and motor actions, with no intermediate CS. This adds an extra layer of complexity, since instead of comparing a single time-step when generating CS, we are learning sequence to sequence mapping in both directions.

The training algorithm for this model was refined when compared to 3.2.3. The complex algorithm that changed the direction of training was discarded, since it did not provide any obvious benefit; instead we always train both directions at every iteration, reducing the oscillations in training loss due to direction switch. As in 3.2.3 we kept the gradients clipped in an interval of $[-7.0, 7.0]$. For the optimiser we kept the AMSGrad optimiser from Reddi et al. (2018), with the same learning rate of 0.0005.

The biggest difference in the training algorithm is in the batch creation and iterations. In 3.2.5 the batches were created randomly at every iteration, with each iteration corresponding to an epoch, which increased the randomness and oscillations during learning. For the MTLSTM model we opt instead to create a number of batches covering the entire dataset. These batches are all fed to the network during an epoch, with the only random factor being the order they are fed. The algorithm for the training can be seen in algorithm 1.

Algorithm 1 Description of the basic training loop for MTLSTM training.

```
Batches - list of all batches
b - batch
numBatch - number of batches
NEpc - number of epochs
MaxEpc, thresh_l, thresh_m - thresholds (epochs, language loss, motor loss)
avg_l, avg_m - average loss over all batches (language, motor)
loss_l, loss_m - loss per batch (language, motor)
while NEpc < MaxEpc and (avg_l > thresh_l or avg_m > thresh_m) do
  for all b in Batches do
    Train motor action
    Train language
  end for
  avg_l = loss_l / numBatch
  avg_m = loss_m / numBatch
  NEpc++
end while
```

We set 2 different conditions for stopping the training iterations: either the model reached a maximum number of epochs, or the values of the loss for both motor action generation and sentence generation reached respective thresholds. For the experiments in this chapter the maximum number of epochs was set at 40000 while the thresholds were set at 0.005 and 0.1 for language and motor actions respectively.

4.2.4 Experimental Results

the aim of these experiments is to test the model for both its data reproduction and its generalisation abilities. To test for data reproduction we use the training set to generate sentences from motor actions and vice-versa. For the generalisation ability we perform two tests: we remove 16 random sequences from the training data during training and use them to test the model, and we test the model on the missing 54 sequences and try to generate the corresponding 9 missing combinations.

The model was trained 5 times with the best one chosen for these tests. This model achieved a loss value of 0.43465 at epoch 40000.

Testing on training data

The first test we performed was on the training set. We tested the performance of the model when generating the training sequences. To compute the error on the motor action generation we calculated the euclidean distance between the output of the model and the

ground truth for the motor action. On the sentence generation we focused on obtaining the correct characters, with the error being the number of mistaken characters.

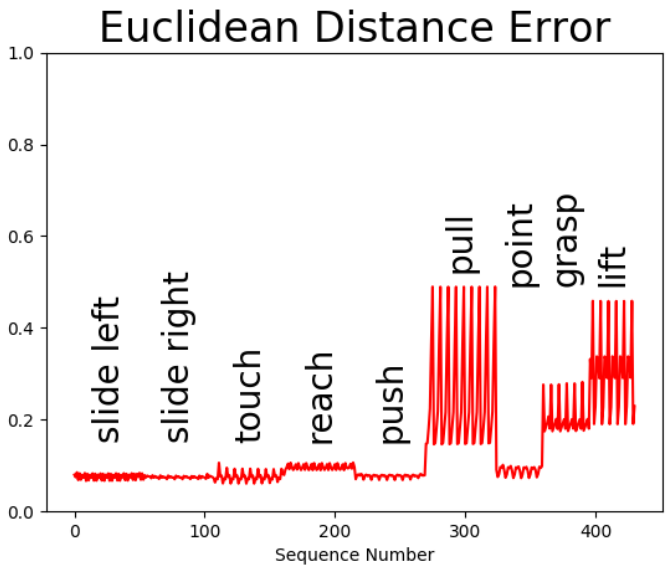


Figure 4.4: Plot with the error for each verb. The error was calculated by computing the euclidean distance between the target output and the effective output over all time steps. For each verb there are combinations of 9 objects and 6 positions.

The error for action generation was kept generally low, with the trajectories for each joint being followed rather precisely. The error per action can be seen in Figure 4.4. We can also see that the error changed a lot depending on the position, but not so much depending on the objects, leading to this "periodic" appearance on the plot.

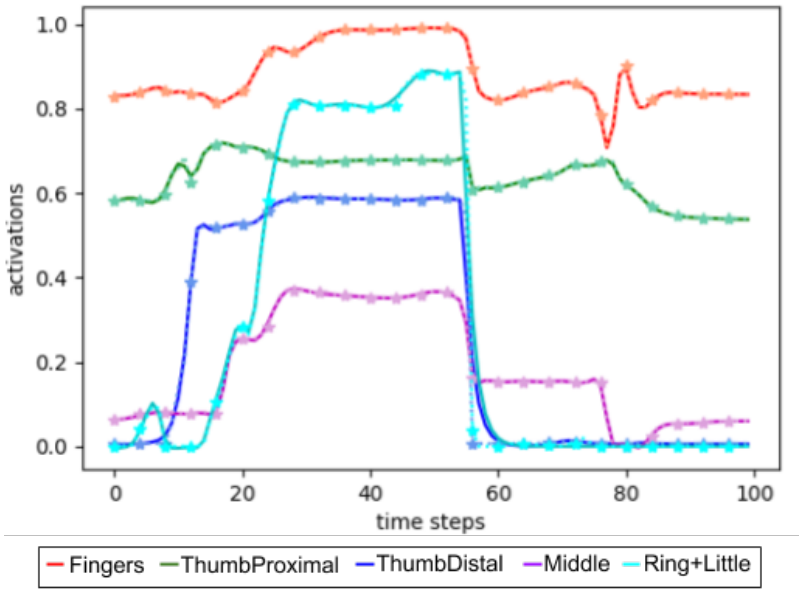


Figure 4.5: Plots for hand encoders during action execution where we compare the output of the model (full lines) with the ideal trajectory (dotted lines and markers).

The highest error was recorded for action "pull the spiky". To analyse this error we plotted

the trajectories for each joint, verifying that the highest error was recorded for the hand joints. The plot of these trajectories can be seen in Figure 4.5.

In the plot we can see that most trajectories (full lines) are very close to the ground truth (dotted lines). The main source of error arises from the deviation near time-step 55, where the thumb distal and ring+little fingers have an abrupt change in the joint value in the ground truth, while the output of the network reacts slowly, leading to a small discrepancy.

In the sentence generation mode, the model was able to reproduce all sentences correctly, without a single mistaken character.

Testing for generalisation ability

In order to test generalisation we have conducted two different tests: we excluded 16 sentences from the training set, testing them with the ground truth input; and we test the 54 sequences that were excluded. For these 54 sequences we have no ground truth knowledge to compare to, so instead we try comparing with similar actions in the dataset.

The motor action generation error for the 16 sentences is included in the plot in Figure 4.5, and is indistinguishable from the remaining actions, showing that the network is able to correctly generalise the motor action.

On the sentence generation side the model was able to generate correctly 12 out of the 16 removed sequences. Of the 4 sequences with mistakes, 2 have minor mistakes, while 2 other are completely wrong. The output for the 16 sequences can be seen in Table 4.3.

We can see the model has some difficulty distinguishing between touch and reach, with the two "reasonable" mistakes hinting at this mistake, when trying to generate "touch the tractor" and "touch the spiky". More critically, however, are the errors when trying to generate "pull the tractor" and "grasp the tractor". In these cases, the network generates the verb "point", with a garbled object resembling "tractor", but the sentence generated is the same for both cases. Interestingly, 3 of the 4 errors happened for the object tractor.

In a second test for generalisation we recreated the 54 missing sequences. In the motor generation task, we provided the corresponding sentences as input, assuming an initial position similar to other sequences, and compared the output to similar actions present in the dataset. The resulting euclidean distance can be seen in Figure 4.6. We can see that the error is comparable to the plots in Figure 4.4, showing that the network can generate

target	output
slide left the bus.	slide left the bus.
slide left the car.	slide left the car.
slide left the cup.	slide left the cup.
slide right the ball.	slide right the ball.
touch the tractor.	**rra**ch the tractor.
touch the ball.	touch the ball.
touch the ball.	touch the ball.
touch the spiky.	t**ea**ch the spiky.
push the car.	push the car.
pull the tractor.	**point thettrattrr.**
pull the bus.	pull the bus.
pull the cup.	pull the cup.
point the bus.	point the bus.
point the modi.	point the modi.
grasp the tractor.	**point hhettrattrr.**
grasp the spiky.	grasp the spiky.

Table 4.3: Table presenting the results of sentence production for the 16 excluded sequences. Mistakes are highlighted in bold.

motor actions even for sequences not explicitly learned.

This result proves that the model is able to learn to distinguish between verb and object in the sentence, and is able to generate the corresponding motion independently of the object.

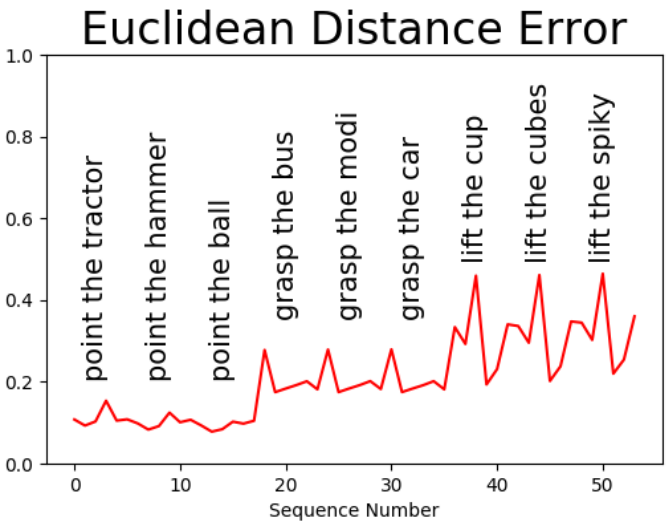


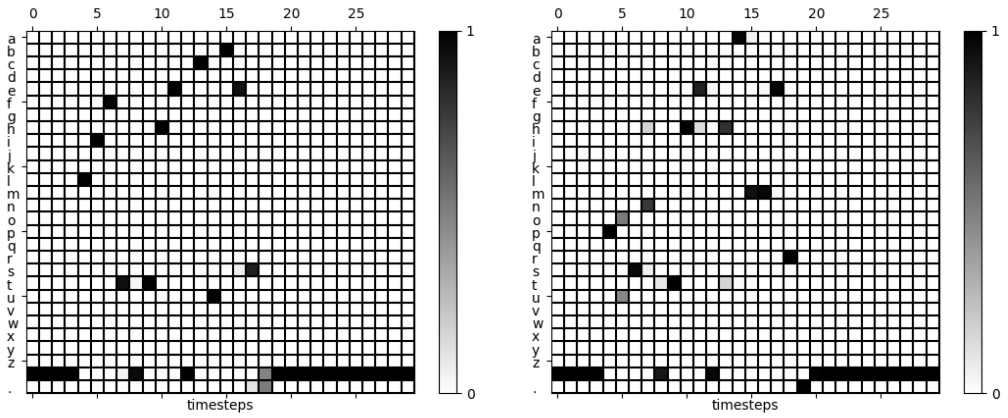
Figure 4.6: Plot with the error for each of the unseen combinations. The error is comparable to the the error in the rest of the dataset, proving that the model can generalise the trajectory to different objects.

On the sentence generation side, the model was able to generate 6 out of the 9 sentences missing from the dataset. The outputs of the network can be seen in Table 4.4. The model had some trouble generating the verb "point", making several mistakes both in the verb itself and in the object.

target	output
point the tractor.	posn the **hammer**.
point the hammer.	posn th**h**e **alle**.
point the ball.	point **h**he bal**.**.
grasp the bus.	grasp the bus.
grasp the modi.	grasp the modi.
grasp the car.	grasp the car.
lift the cup.	lift the cup.
lift the cubes.	lift the cubes.
lift the spiky.	lift the spiky.

Table 4.4: Table presenting the results of the generalisation test for the language output using novel combinations. Mistakes are highlighted in bold.

To investigate further the errors on the sentence generation we have checked the probability distribution of the characters in the model output. An example of such graphs can be seen in Figure 4.7, for a good output (Fig. 4.7a) and one with an error (Fig. 4.7b). We can see that while the probability is concentrated on the correct character in Figure 4.7a, the same is not true for Figure 4.7b, where there is a confusion between "push" and "point". Interestingly, despite outputting the wrong object for the sequence, it does so with a lot of certainty, with the probability for the corresponding characters being quite high.



(a) (b)

Figure 4.7: Activation of the language output for each time step for the cases “lift the cubes” and “point the tractor”. X axis corresponds to the time steps while Y axis corresponds to the different letters.

4.2.5 Comparison with Baselines

One important aspect of the MTLSTM model is how it performs when compared to other methods already available in the literature. To test this we defined a set of baselines, with combinations of bidirectional training, presence of timescale factors and hierarchical architecture. In this section we will present the different baselines and how they compare

to the MTLSTM model.

Each of these models will be trained 5 times, with the best model being used for the tests.

Dual MTLSTM

The dual MTLSTM model tries to learn how to generate motor actions and sentences without the bidirectional training. To achieve this, each MTLSTM will be trained on a single direction, learning only to generate one of the outputs.

The best model hit the language threshold at epoch 7924, while it never reached the threshold for motor actions, achieving a best loss of 0.60978 at epoch 40000.

This model allows to see the effect and utility of bidirectional training compared to common single-direction architectures.

Mini Dual MTLSTM

Similar to the model described above, but tuned so the number of parameters to learn are the same as for the bidirectional MTLSTM. We calculated 691900 parameters, corresponding to all the weights applied to the inputs/outputs of the cells in the model. In order to obtain a similar amount of parameters, it was calculated that the mini dual MTLSTM should have the following structure: in the language branch, the I/O layer has 28 cells, the FC layer has 112 and the SC layer has 25; on the motor action branch, the I/O layer has 98 cells, the FC layer has 112 and the SC layer has 25; finally, the meaning layer has 18 cells. The timescale factors were not changed.

This model hit the language learning threshold at epoch 18465, and it never hit the motor action threshold, with the minimum loss for motor actions being 1.16157 at epoch 40000.

bi-LSTM

This model tests the bidirectional training but using only simple LSTM organised in a hierarchical architecture, but with no timescale factors. The only difference between this model and the MTLSTM model is the replacement of CTLSTM cells for regular LSTM cells. This way we intend to test the effect of timescale factors in the model.

The model achieved a best loss of 1.48914 for sentence generation and 42193.04807 for

motor action generation, being incapable of learning how to generate either of the outputs.

Dual Hierarchical LSTM

This model represents the implementation closest to the what is traditionally used in literature, with a stack of LSTM working in a single direction. This model uses no timescale factors and is trained in single direction, generating motor actions in one directions and sentences in the opposite direction, with a parallel model.

The best loss achieved by this model in sentence generation was 1.97266 at epoch 34652 and motor loss of 497.99 at epoch 34701. As in the bi-LSTM model, this model was also unable to learn to reproduce the data.

Analysis

The results of the training of these baseline models can be visualised in Table 4.5. From these results we can see that the only model performing better than the MTLSTM trained in a bidirectional way is the dual MTLSTM, achieving a slightly better result in motor action generation. This could be due to the effectively higher number of parameters, and when we adjust for this, in the mini dual MTLSTM, we can see that the results for motor action generation are not as good as for our MTLSTM model. Both LSTM implementations faced serious issues with learning this data, revealing how critical the timescale factors are for the model.

Model	lang. error	action error	epoch
MTLSTM	**0.00253**	0.86677	40000
2 MTLSTM	0.00499	**0.60978**	7924 / 40000
2 mini-MTLSTM	0.00429	1.16157	18465 / 40000
bi-LSTM	1.48914	42193.04807	40000
2 hierarch LSTM	1.97266	497.99	34652 / 34701

Table 4.5: Table indicating the lowest error and the epoch it was obtained for each of the models.

With this analysis we acquired some more insight into the model, revealing the importance of timescale factors and showing how bidirectional training can help the model function in both directions with fewer parameters.

4.3 Conclusion

In this Chapter we introduced a novel MT model, based on LSTM models, to connect actions and language in a single model, and trained this model in a bidirectional way, allowing the model to generate actions from sentences and vice-versa. This model had no explicit prior knowledge in the form of control sequences, relying entirely on joint angles for motor input and one-hot letter inputs for sentences.

We validated the model using a dataset from an iCub robot, verifying the ability of the model to generate all trained data, generalise to unlearned sequences, and randomly removed sequences. The model was able to generalise to most sequences, both when generating a motor action and a sentence. We studied also motor action generation and verified that the model was able to correctly follow the trajectories for each joint.

Finally, we compared this model to a set of baselines involving combinations of the different aspects of the MTLSTM model, showcasing the importance of each for the correct learning. The model outperformed all baselines presented, with a minimally better performance obtained only when two models with the same size are used in single directions, in parallel. We verified that training a MTLSTM model in a bidirectional way allows for a reduction in the number of parameters for minimal change in performance.

Finally, by successfully testing a model that connects languages and actions directly, we have implemented the first successful model that answers RQ1 and RQ2: we now have a model that grounds actions and language, trained in a bidirectional way, capable of generating motor actions from sentences and vice-versa.

Chapter 5

Language and Action Learning Through Robot Interaction

Key points:

- Data acquisition module
  - Pepper Robot
  - iCub Robot
- Extension to different robots and data
  - Training with a bigger dataset
  - Using Pepper motor data
  - Comparison between Pepper and iCub training
- Three-branch MTLSTM model
  - separating language, motor and visual inputs
  - iCub large dataset training

In Chapter 4 we arrived at a model that is capable of linking language to motor actions directly, achieving good results. For those experiments, a dataset collected previously with the iCub robot by Peniak et al. (2011a) was used. While this worked well for designing and testing the model, further testing requires exposing the model to more and different data.

A first experiment involved using a different robot, in this case Pepper (Softbank Robotics). Pepper is significantly different from the iCub robot, particularly when it comes to the hand joints, and thus would expose the model to slightly different robot and data.

In order to acquire more data for testing, a collection module was developed. The data collected by the module represents very low-level data collected by the robot sensors for its motor actions. On the language side, the module records the sentence, requiring some speech-to-text conversion when using voice inputs.

One particular aspect of the previous dataset was that it relied entirely on transitive verbs. This meant that the network would easily learn to expect an object after a verb. In reality, grammar also has intransitive verbs, and therefore it would be of interest to test the model when it has to learn both types of verbs.

Finally, while a visual pipeline in itself is not the goal of this project, we aim to test a 3-branch MTLSTM, connecting language input, visual input and motor input at the meaning layer. We extend our MTLSTM model to include object label input in the meaning layer instead of the motor I/O layer. The position of the object will be input in the motor I/O layer, obtained by 3D visual processing with the iCub robot modules. More details on this extension will be presented in Section 5.1.3.

5.1 Dataset Acquisition with Pepper

The Pepper robot was developed by Softbank Robotics as an interactive robot, capable of understanding spoken language, replying and interacting with the world in a limited way. It is also capable of locomotion by way of wheels at its base, and includes a touch-screen in its chest for more interactive capabilities. It has 6 joints for each arm (shoulder pitch and roll, elbow yaw and roll, wrist yaw and hand open/close), 2 for the head (yaw and pitch), and 3 for lower body (hip roll and pitch, and knee pitch).

The scenario to be considered is one where the Pepper robot is in front of a table with a single object. The data to be retrieved is a written sentence, the joint positions over time, and a snap shot of the initial position of the object.

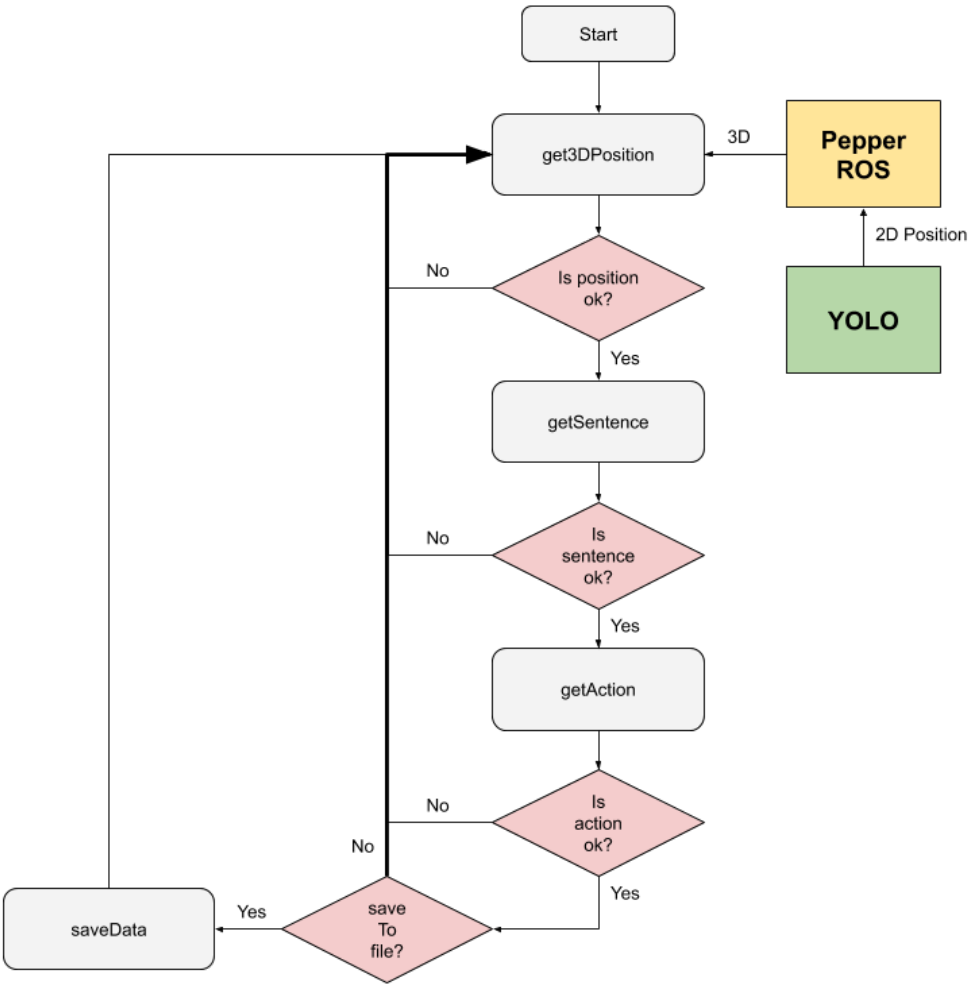


Figure 5.1: Flow chart for the Pepper data acquisition module. YOLO (green) and Pepper ROS (cream) correspond to interaction with the respective external modules.

5.1.1 Architecture

The data acquisition module was developed with the objective of being as user-friendly as possible, in order to enable non-tech-savvy users to be able to interact with the robot. It first records the sentence, followed by a snap-shot of the initial conditions, recording also the 3D position of the object, and finally recording the joint positions over time, while the user physically moves the robot.

In order to acquire speech, we recorded spoken language using Choregraphe, an application developed by Softbank Robotics that allows for easy programming and interaction with the Pepper robot. This allowed easy interaction with the robot, however it sometimes proved too noisy for easy data collection, and therefore a terminal-input option was left for recording the written sentence.

In order to store the labels and obtaining the 3D position of the objects we developed a module interacting with YOLO (You Only Look Once, developed by Redmon et al. (2016)) and Pepper ROS (Robot Operative System) modules. We trained the YOLO model to recognise the objects used for our experiments, and used the model to locate the objects in the snap-shot image of the initial position. Using ROS transforms, we were able to transform the 2D image position into the 3D position relative to the centre of the robot.

Finally, the user is prompted to perform the motion with the robot. Each action takes 16 seconds, divided in 100 steps, with the joint position being recorded each 0.16 seconds. During the action, the robot is set to compliant mode in order to be moved by the user. Once the action is complete the robot return to an initial position.

The interaction between modules can be visualised in Fig. 5.1. Interaction with the robot is performed via touch in its head, with a forward touch corresponding to a "yes" and a back touch to a "no". The modules were implemented in Python programming language.

5.1.2 Data Structure

The data collected is saved in 3 different formats: i) a .cfg file that links all information; ii) a .data file that stores all action sequences; and iii) the image files corresponding to the different actions. The .cfg file stores the language component (sentence), together with the object label, the corresponding action sequence number, and the name of the image file. An example of an entry in this file follows:

Entry in the .cfg file
ELEMENT 6 Image: image-6.png Sentence: slide left the whale Label vector: 0 0 0 0 1 0 0 0 0

The .data file stores all the action sequences recorded. Each sequence has its own number, corresponding to the number on the .cfg file. The sequences are 100 time-steps long, and store all 17 joints of the Pepper robot, and the 3D position of the object, for a total of 20 cells per time-step.

Finally, the images are stored directly from the robot cameras as .png files. While the current model does not use the images to generate any output, they are stored to account for future extensions of the model and/or future research in this area. An example of a visual entry can be seen in image 5.2.

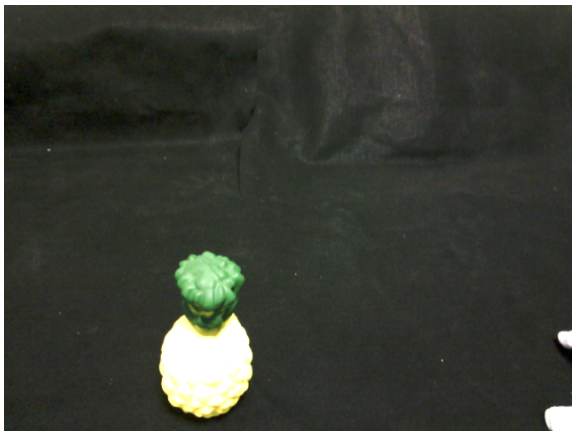


Figure 5.2: Example of visual data stored in the dataset for Pepper robot.

5.1.3 MTLSTM training

Dataset

The main purpose of the data collection modules for the Pepper robot was to collect data to test the MTLSTM model under different conditions. Using this robot, we collected a new dataset comprising 9 different actions ("slide left", "slide right", "touch", "reach", "grasp", "lift", "push", "pull", "point at"), 9 objects ("duck", "whale", "frog", "funnel", "marker", "ball", "hipo", "pineapple", "octopus") in 4 different positions, totalling 324 sequences. The main difference of this dataset compared to the one used in Sec. 4.2 and 3.2 is the size of the motor sequences, with 17 joints for the Pepper robot compared to 41 for the iCub robot. The objects were chosen to be common objects and toys, picked from the objects available

in the lab.

adapted MTLSTM

In the MTLSTM model used in Sec. 4.2 the object encoding was an input to the motor branch, which meant the motor and visual pipelines in the model occupied the same areas. For future model adaptations, however, we would consider three different pipelines for the three different sources of information: language, visual and motor. To account for this, we have adapted the MTLSTM model to take the object labels as input to the meaning layer, thus merging all three different pipelines only at this layer. The object position is still considered as an input to the motor branch, however, given its relevance to the action execution. The schema for this model can be seen in Figure 5.3.

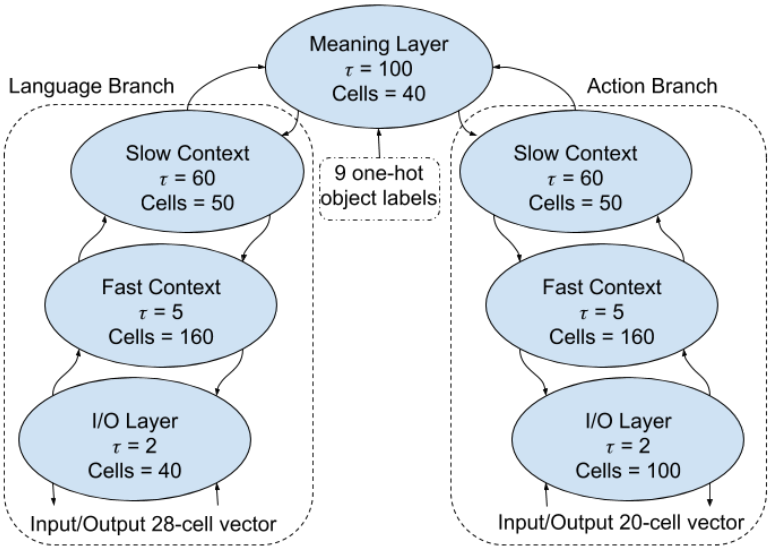


Figure 5.3: Illustration of the adapted MTLSTM model. Object labels are now input at the meaning layer level.

In addition to the new input for the meaning layer, the size of the model was also adapted: the I/O layer size was reduced, given that the input size is now considerably smaller, and the size of both the meaning layer and the SC layers for both branches was increased, in order to account for the new label input.

training

We trained the MTLSTM model in the same way as in Sec. 4.2.3, using batches of 32 sequences, bidirectional training and early stopping thresholds of 0.005 and 0.1 for language and actions respectively. The model was trained 5 times with the best iteration being

chosen for testing.

The objective of the training was to test if the model could still train correctly, and as such the entire dataset was used for training. The best model achieved the best average loss of 0.437647 at epoch 23090.

Experimental Results

In this experiment we aimed to test the ability of the model to learn how to reproduce the training data, given the 3 different information flows: language, vision and motor. We tested the model for generating both sentences and motor actions, and evaluated its performance by analysing the number of wrong letters and the average error for the motor action.

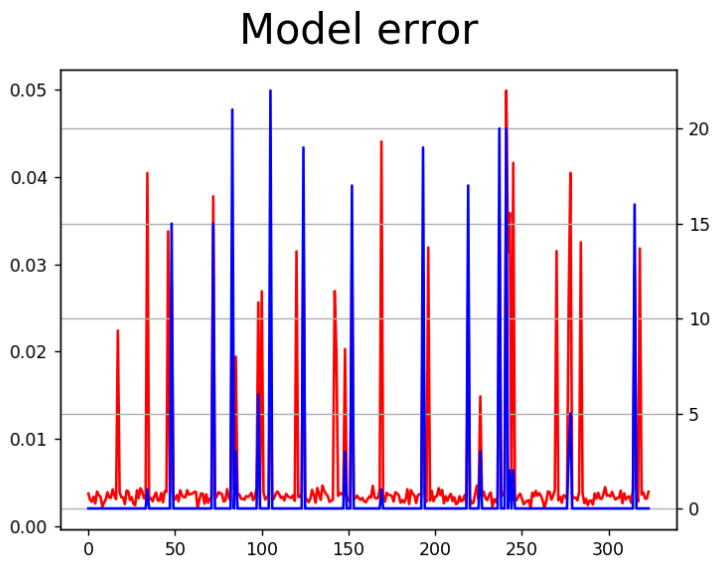


Figure 5.4: Plot of the error for sentence generation (blue, right axis) and motor action generation (red, left axis) when learning the Pepper dataset.

The model was able to reproduce most of the sentences, making mistakes in 21 sentences out of the entire dataset. This is a rather big contrast to the success reported in Sec. 4.2.3, where 100% of the training data was correctly generated. The error for sentence and motor action generation can be visualised in Figure 5.4. To further study the problems with the model, we explored the wrong sentences, which can be seen in Table 5.1. The generation of the object name is correct in most instances, while the verb is often mistaken, even if spelled correctly.

On the motor action generation task the model also had few actions with a very high error.

target	output
grasp the marker.	**lift** the marker.
grasp the funnel.	**lift** the funnel.
grasp the hipo.	**lift** the hipo.
grasp the pineapple.	**lift** the pineapple.
slide right the funnel.	slide **liftt** the **u**unn**l**l.
slide right the octopus.	**grasp** the octopus**..**
slide right the pineapple.	slide **left** the pineapple.
slide left the funnel.	slide **i**eft the **uunel**.
slide left the duck.	slide l**i**ft the duck.
slide left the duck.	slide left the **u**uck.
lift the hipo.	**slide left the iip**.
lift the marker.	**slide left** the marker.
pull the frog.	**grasp** the frog.
pull the pineapple.	**geash** the pine**ea**ple..
push the hipo.	**slide lift thhepiip**.
push the pineapple.	push _ he pi**i**eapple.
reach the marker.	**touch** the marker.
reach the ball.	**touch** the ball.
touch the whale.	t**ra**ch the whale.
touch the hipo.	**reach** the hipo.
touch the frog.	**reach** the frog.

Table 5.1: Table presenting the mistaken sentences when trying to generate the training set sentences. Mistakes are highlighted in bold.

We plotted the trajectory for the action with the highest error in order to understand what is leading to these errors. This plot can be seen in Figure 5.5.

The trajectories corresponding to these error spikes translate into trajectories that are very different from the target. It is, however, interesting to note that this trajectory corresponds to a sentence that was also erroneously generated, mistaking the verb "slide left" for "lift".

We extended the training of the model to investigate if the errors found were due to lack of training. We let the model train until the 120000 epoch, where it achieved a best average loss of 0.3697358 at epoch 109920. The results for this training can be visualised in Figure 5.6.

Despite training the model further, the number of erroneous sentences increased to 23, revealing that no improvement was achieved.

A comparison between this experiment and iCub experiments is made in Section 5.3, analysing in more detail the possible sources of error in this model.

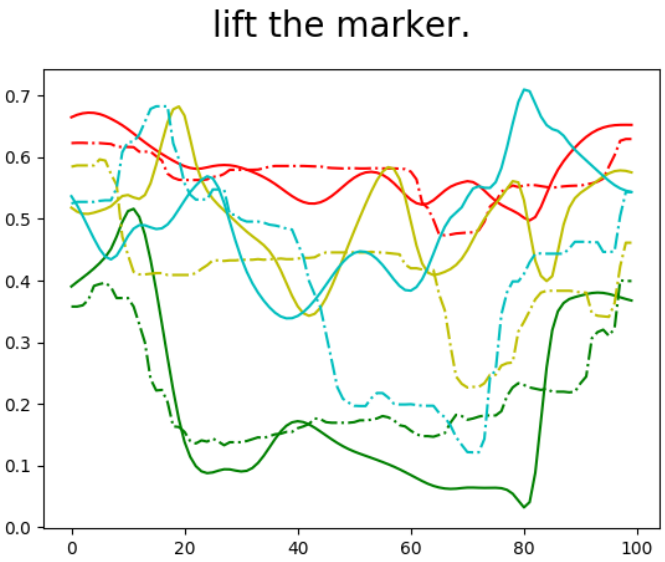


Figure 5.5: Plot of the trajectories for the action with the highest error, "lift the marker". Red is the left shoulder pitch, green is the left shoulder roll, yellow corresponds to left elbow roll and cyan corresponds to left wrist yaw. Target trajectories are indicated in dotted lines, output of the model in full lines.

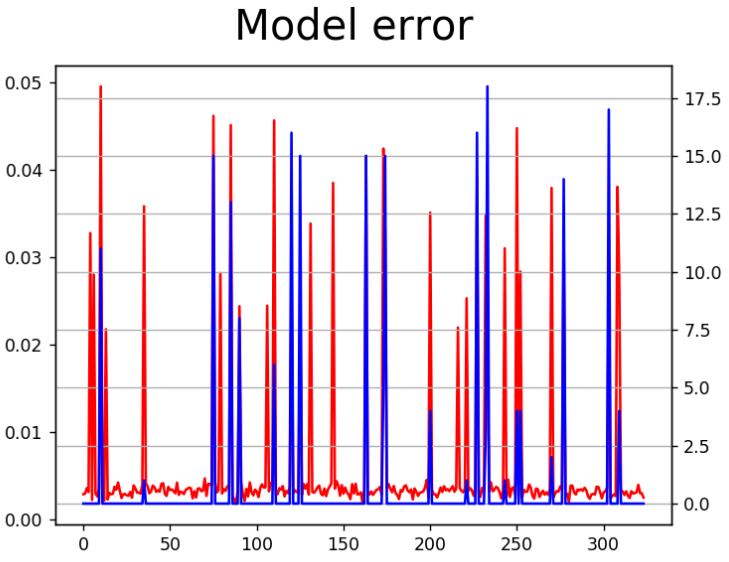


Figure 5.6: Plot of the error for sentence generation (blue, right axis) and motor action generation (red, left axis) when learning the Pepper dataset with 120000 epochs.

5.2 Dataset Acquisition with iCub

In order to develop the dataset acquisition module for the iCub robot, some constraints were considered similar to Section 5.1.1: i) 3 different sources of data should be collected, namely language (sentences), robot joints, and images of visual input; ii) motor actions should be performed by a human moving the robot arms during the action; and iii) the system should work with only a human participant interacting with the robot, with no external help. This last constraint allows any person, even if not very tech-savvy, to teach the robot how to perform an action, since only speech and physical interaction are needed.

5.2.1 Architecture

Due to the different sources of data, several modules need to work in unison to collect all the necessary data. This module was developed within the YARP (Yet Another Robot Platform, Metta et al. (2006)) framework, in C++ programming language, that introduces many useful tools for connecting and interacting with the iCub robot.

The system is focused on the central module, responsible for collecting data from the different sources, recognising the inputs from the user, and storing all the data in the corresponding file. The module works as a simple state machine, and a flow chart of its execution can be seen in Figure 5.7.

All interaction with the robot is done via speech, and physical interaction while teaching an action. The visual pipeline comprises a series of modules present in YARP, and are responsible for both recognising the object and obtaining its 3D position. During the recording of an action, the user will have to issue simple commands to the robot, e.g. "close left hand", which are actions that are difficult to demonstrate. These actions are encoded in the "Actions Rendering Engine" (ARE), also from YARP, and responsible for executing these actions. The recording of the joint positions is done directly using the robot controllers, by the main module.

5.2.2 Data structure

The structure of the data for the dataset collected with this module is similar to the one presented in Section 5.1.2. It is split into 3 different formats, the .cfg file, the .data file, and

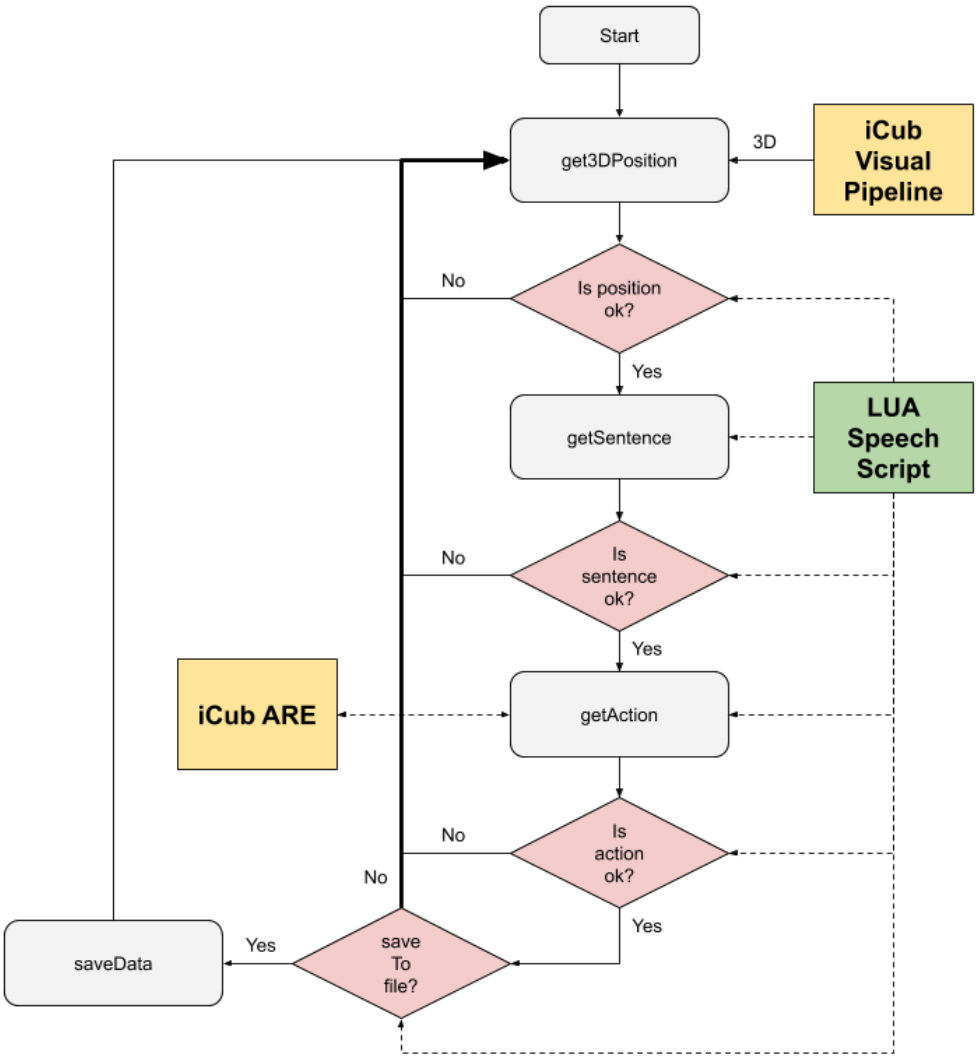


Figure 5.7: Flow chart of the iCub architecture. All interaction with the modules are done by speech, processed by a LUA script.

the image files, which are stored as .jpg files instead of .png. The second difference has to do with the number of objects in the object label, where 18 objects are now considered. An example can be seen in the following entry:

Entry in the .cfg file
ELEMENT 13 Image: image-13.jpg Sentence: slide right the cube Label vector: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

As in Section 5.1.2 the action sequences were recorded on the .data file. Instead of storing 20 elements per time-step, however, this module stores a total of 46 elements: 41 elements for the 41 joints of the iCub robot (16 for each arm, 3 for the torso, 3 for the head, 3 for the eyes), 3 for the 3D position of the object, and finally 2 more elements to encode the emotional displays of the iCub robot. The last two elements are useful for encoding actions like "smile" or "laugh" which have no motion components, but are shown through the iCub facial expressions LEDs. Each action has a length of 100 time-steps, for a time duration of 16 seconds.

Images are stored directly from iCub cameras. An example of such an image can be seen in Image 5.8.

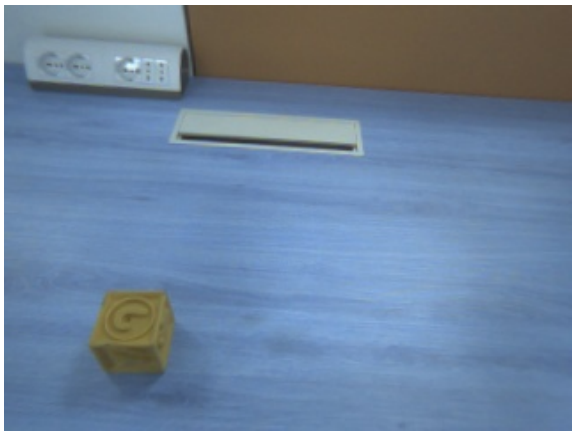


Figure 5.8: Example of visual data stored in the dataset for the iCub robot.

The data collected is saved in 3 different formats: i) a .cfg file that links all information; ii) a .data file that stores all action sequences; and iii) the image files corresponding to the different actions. The .cfg file stores the language component (sentence), together with the object label, the corresponding action sequence number, and the name of the image file. An example of an entry in this file follows:

5.2.3 MTLSTM training

Dataset

In the dataset collected for this training we opted to extend the number of actions and objects, and also the nature of the actions, by introducing intransitive verbs and actions like "smile", "nod", etc. The objects were chosen to be common objects and toys, picked from the objects available in the lab. The full list of actions and objects can be seen on Table 5.2.

Table 5.2: Table of the actions and objects used in this dataset

Verbs	**Objects**
push	cube
pull	box
slide left	ball
slide right	car
point at	bottle
grasp	dog
smile	carrots
frown	bear
nod	fish
laugh	tool
cry	rake
show	fruit
shake the head	cat
touch	cereals
take	turtle
punch	lego
give	octopus
poke	spoon

A special note should be given to the action "shake the head". This action does not take any other object, and thus is considered as a singular action.

All combinations of objects and actions are present in the dataset. All actions are taught with the object in 4 different positions, except for intransitive verbs ("smile", "frown", "nod", "laugh", "cry", "shake the head") which are taught with only 2 positions (left and right). The total number of sequences recorded is 1080.

adapted MTLSTM

The model used for these experiments is a similar model to the one described in 5.1.3, with the following changes: the size of the label input was increased (18 instead of 9 labels) and

the size of the I/O layer for the motor branch was again increased to 140 cells (from 100). This is the same size used in Chapter 4, which had good results when learning iCub data. The model with the updated size and cells can be seen in Figure 5.9

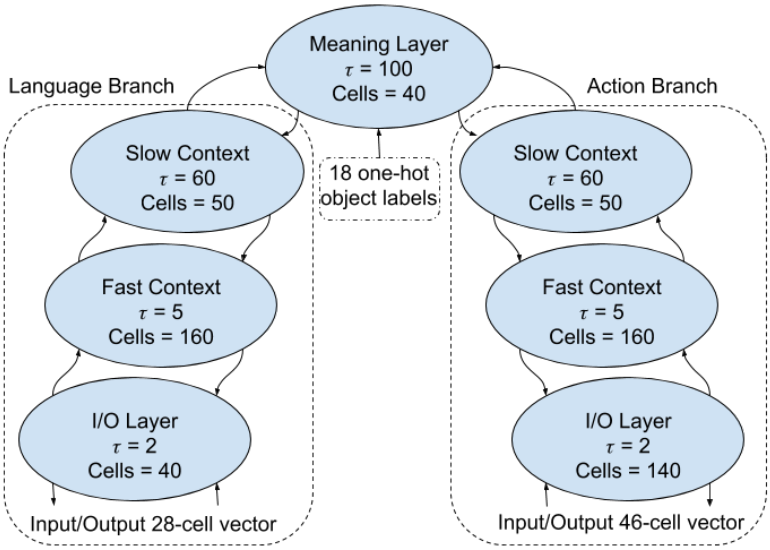


Figure 5.9: Illustration of the adapted MTLSTM model used for the iCub experiments. Note the different input sizes for the labels and motor actions, and the increased size of the I/O layer on the motor branch

training

To train this model with this larger dataset we increased the size of the batches to 64. We have kept the early stopping thresholds for both motor actions and language, but increased the maximum number of epochs to 90000 epochs. As in 5.1.3 we trained the model 5 times and picked the best for testing.

The best model achieved an average loss of 3.734095 at epoch 90000.

Experimental Results

This experiment, as in 5.1.3, aims at testing the ability to reproduce all the training data. We tested the model to generate both sentences and motor actions, and plotted the average error in Figure 5.10.

The model was able to reproduce all sentences correctly, making one single mistake: "tave the fruit." instead of "take the fruit". On the motor action generation the error was also very small, never exceeding the average of 0.01. We plotted the joints with the worst error, corresponding to the action "point at the lego", which can be visualised in Figure 5.11.

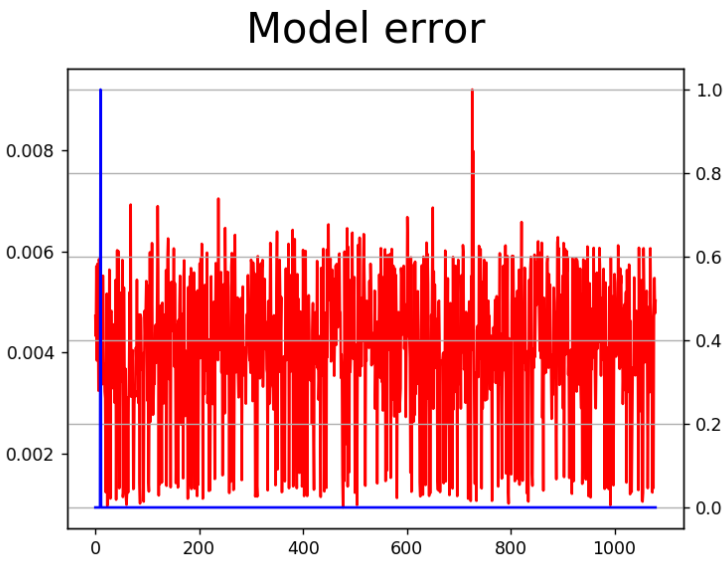


Figure 5.10: Plot of the error for sentence generation (blue, right axis) and motor action generation (red, left axis) when learning the iCub dataset.

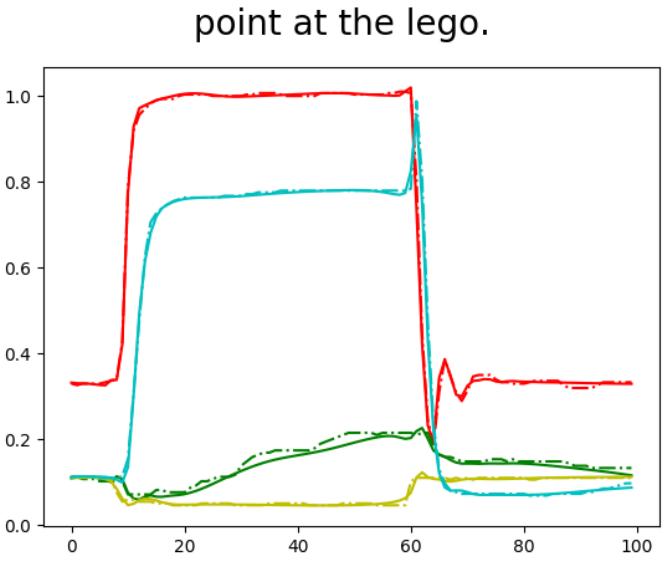


Figure 5.11: Plot of the trajectories for the action with the highest error, "point at the lego.". Red is the left hand finger abduction, green is the left thumb proximal joint, yellow corresponds to left index proximal joint and cyan corresponds to left middle proximal joint. Target trajectories are indicated in dotted lines, output of the model in full lines.

We can see in the plot that the model follows the trajectories reasonably well, even in this worst case scenario. The extended model was able to correctly generate both sentences and motor actions, considering new sources of input to the meaning layer, a larger dataset with larger inputs.

5.3 Comparison between Pepper and iCub

While experiments with the iCub robot described in Section 5.2 proved successful, the same model had some difficulties in correctly learning the dataset acquired with Pepper robot, described in Section 5.1.

The iCub robot has more than double the degrees of freedom of Pepper robot, providing a lot more data to be used in classifying actions. This is particularly true when considering actions that involve using fingers, like pointing or grasping, where Pepper considers a single joint while iCub has different values for the different fingers. This could lead to problems when trying to analyse such actions performed by the Pepper robot. To investigate this, we verify whether the model has similar issues when using the dataset collected with the iCub robot, but using only the index or middle fingers instead of the whole hand, simulating the effect of this single joint in Pepper robot.

5.3.1 Tests with constrained data

We performed two experiments with constrained data: i) using iCub dataset but disregarding all finger joints except for the middle finger, and ii) iCub dataset, but using only the index finger. We trained the model with this data in the same way described in Section 5.2.3. The best model achieved an average loss of 1.35379 at epoch 89922 for the middle finger case, and an average loss of 0.629259 at epoch 88905 for the index finger.

The middle finger model correctly reproduced almost the entire dataset, failing only in 8 sentences, while the index finger model failed only in 5 cases. The plots for both model can be seen in Figure 5.12.

We can see that even in the cases where there were errors in sentence generation, these errors were usually restricted to one or two letters for the middle finger model. In the index finger model there were less erroneous cases, but the errors were worse in that there were more wrong letters (up to 5) in the generated sentences.

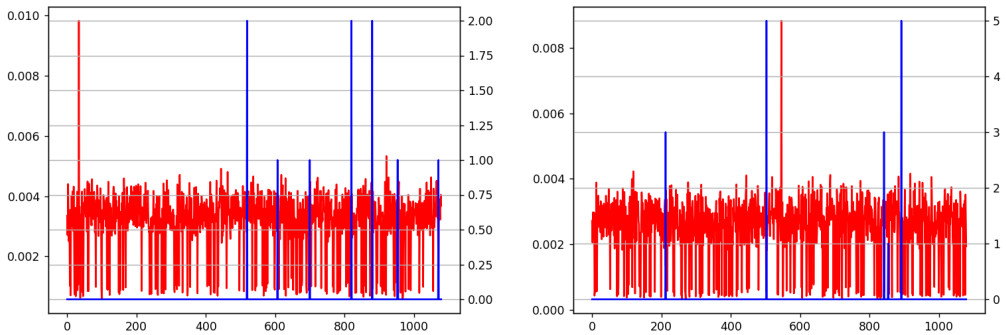


Figure 5.12: Plots of the error for sentence generation (blue, right axis) and motor action generation (ref, left axis) for the models trained with only the middle finger (5.12a) and index finger (5.12b).

In both situations the error in motor action generation was kept very low, particularly when compared to the results obtained in Section 5.1.

5.3.2 Discussion

While the MTLSTM model was able to reproduce the entire iCub dataset, it had some difficulties in the case of the Pepper dataset. The hypothesis that more joints in a robot provide more data, allowing the model to learn more accurately, was verified with the experiments with constrained data using the iCub dataset, where using a single finger proved detrimental for the successful training of the model, leading to errors on sentence generation. This, however, does not completely explain the problems faced when using the Pepper dataset, since the number of errors when reproducing the dataset was an order of magnitude higher, particularly considering the sizes of the datasets (324 sequences for Pepper versus 1080 for the iCub).

Most of the errors when generating sentences in the Pepper dataset case were substitution errors, where an incorrect verb was generated, even if spelled correctly. In the tests performed with constrained data with the iCub dataset this was not the case, with most errors being only one or two letters. This could indicate that the actions performed with Pepper robot are not different enough for the model to be able to distinguish them in some cases. The training time of these models is very high, taking up to two weeks for training the Pepper dataset and 3 to 4 weeks for the iCub dataset. These models were tested in i7 CPUs (Intel Core i7-6800K CPU @ 3.40GHz x 12) with 16 GB of memory, using all 12 CPUs. Testing in GPUs (GeForce GTX 1070) did not lead to a decrease in training time,

given the recurrent nature of the model.

5.4 Conclusion

In this chapter we described data acquisition modules for two different robots, Pepper and iCub, and an extension of the MTLSTM model, considering a third information pipeline in the form of object labels, connected to the meaning layer.

We validated the datasets acquired and the extended model by training it with the newly acquired data, showing the ability of the model to generate the trained data.

We compared the model trained with the data acquired with Pepper robot with the model trained with iCub data, including some tests with constrained data for a more balanced analysis, showing that the morphology of the iCub robot contributes for better learning due to the richer data provided by the increased degrees of freedom.

Chapter 6

Emergence of Meaning

Key points:

- Analysis of meaning in MTLSTM
  - PCA analysis of different layers
  - Understanding meaning
- Synonyms
  - How MTLSTM reacts to synonyms

The models presented in the thesis have been heavily inspired in neuroscience and psychology, leading to the implementation of Bidirectional training and the full connected model presented in Chapter 5. While the model was successful in connecting actions to language, the mechanisms underlying this connection are not entirely clear.

One of the key points for the introduction of a "meaning layer" was the emergence of "meaning" at higher-level layers, that would successfully connect a motor action branch with a language branch. While this layer was critical in the model, it is not clear whether a true "meaning" emerges, and it behoves the analysis of this and other layers in order to find if, and where, meaning is actually learned.

In finding meaning, however, a different question should be answered: how is meaning defined? One could argue that the action of pushing a ball and the sentence "push the ball" have the same underlying meaning, but would two different sentences with the same verb, e.g. "push the ball" and "push the car" have the same meaning? Or, in a parallel

reasoning, would synonyms, e.g. "pull the car" and "drag the car" have the same meaning? In the latter case, the different verbs would relate to the same motor action, while the in the first situation the same verb would relate to slightly different motor actions. Where, then, does meaning emerge?

In this chapter we will explore the inner workings of the model, trying to find and describe meaning and its emergence, particularly in higher-level layers. We will study the similarities within verbs and actions, and also the effect of synonyms in the model, and how they can help in finding where meaning truly emerges, and how it should look like in the model.

6.1 Definition of Meaning

The definition of meaning is anything but clear. In LEXICO [1] the definition comes as "What is meant by a word, text, concept, or action". Merriam-Webster [2] has several definitions:

1. The thing one intends to convey, or is conveyed, particularly through language;
2. Something meant or intended;
3. Significant quality;
4. The logical connotation of a word or phrase; the logical denotation or extension of word or phrase

For this thesis we are particularly interested in definitions 1 and 2. It is interesting, however, how these definitions fall into the Chinese Room problem (Searle (1982); Harnad (1990); Cangelosi et al. (2002)) - in order to define meaning, these definitions fall back on the verb "to mean", which itself is defined as "to convey meaning". There is no real world grounding for meaning, but one possible definition for the word could be "the internal representation of concepts in the human brain". Oishi (2003) defines two different types of meaning, semantic and pragmatic, where semantic is connected to the content of a sentence, derived from the items in the sentence, while pragmatic depends on the context in which the sentence is uttered. According to this definition, meaning itself is not purely

[1] https://www.lexico.com/
[2] https://www.merriam-webster.com/dictionary/meaning

connected to perception, nor is it completely "in our heads"; in fact, meaning arises from the interaction between these components.

The link between grounding and meaning is evident: grounding a concept can be seen as giving it a meaning, and therefore the emergence of meaning derives directly from grounding itself (Cangelosi et al. (2002); Cangelosi (2006)). Such link fits directly with the definitions mentioned above: grounding is not connected solely with perception, or entirely in our heads, rather it arises from the attempts at creating a representation of our perceptual inputs in our brains. Such representations need not be fully connected to physical entities: abstract concepts can be grounded on other concepts which, themselves, are grounded in physical representations. In this way, meaning can be linked to a physical entity (e.g.: what we mean by "that ball") or be a more abstract entity (e.g.: what we mean by "studying").

In the present analysis we will consider "meaning" as a "symbol" that links action to language. This ensures the following definitions:

1. The meaning inferred from a certain sentence can be used to trigger the corresponding action;
2. The meaning inferred from a certain action can be used to trigger the corresponding sentence;
3. Synonyms will have similar meanings;
4. Similar actions will have similar meanings;

Using these definitions, we can define meaning, in the model, in the following way: meaning will be found in the layer that generates similar trajectories for similar actions and/or sentences.

6.2 Analysing MTLSTM layers

A first analysis of the model will try to find where meaning emerges. To do so we will perform a Principal Component Analysis (PCA) on the inner states of the cells of the different layers, in order to reduce the dimensionality of these layer into a 2D plane, expecting to find some structures that represent shared concepts or meanings.

6.2.1 Analysis on adapted MTLSTM

The first analysis was performed on the adapted model described on Section 5.1.3. We used the same model trained in Section 5.2.3, and ran a PCA on the states of the cells in the two SC layers and meaning layer. We hypothesise that if meaning is to emerge in the model, it would be in one of these connecting layers where the abstraction level is higher.

There are three different axes for comparison: comparing within an action, comparing to different actions, and comparing in different directions.

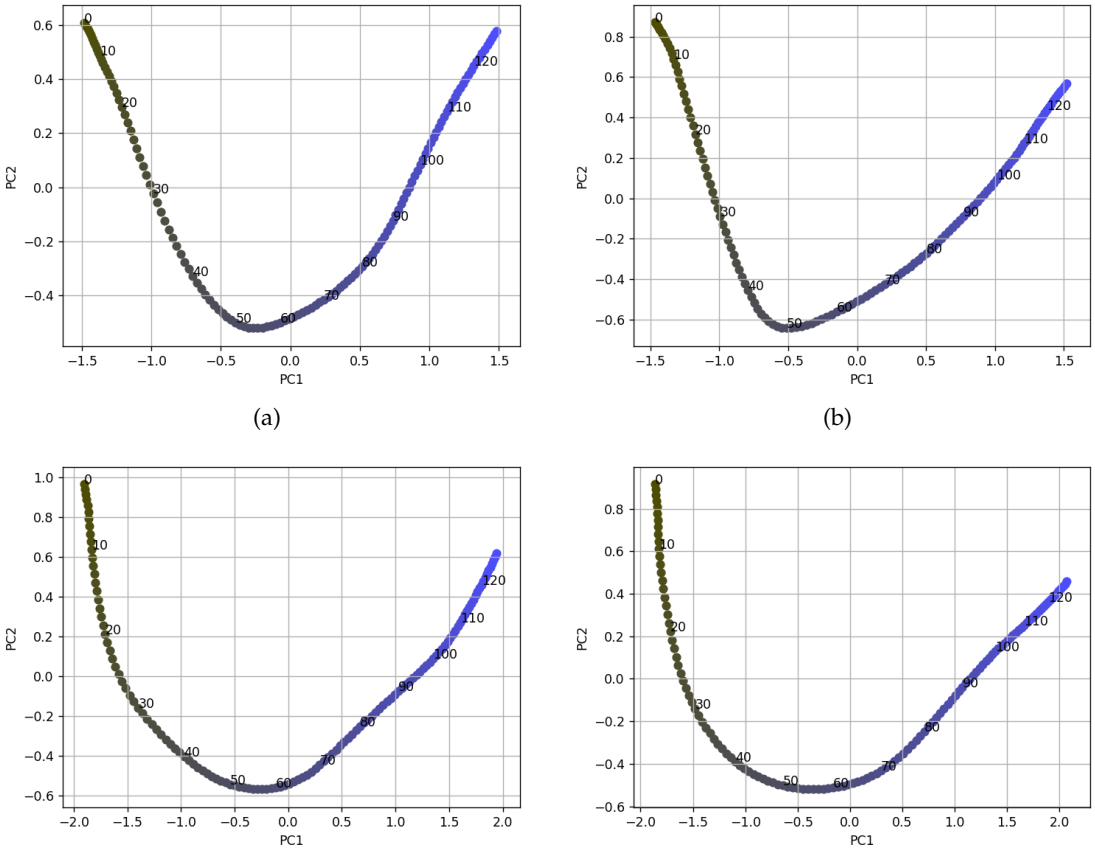


Figure 6.1: PCA scatter plot for the meaning layer for the action "give the cereals", performed in 4 different positions: far right (6.1a), slightly to the right (6.1b), slightly to the left (6.1c), and far to the left (6.1d). Dark brown corresponds to earlier timesteps, light blue to later timesteps. Stamps of the corresponding timesteps are indicated in the graph for every 10 timesteps. These trajectories were generated when the model was generating a sentence from a motor action.

Comparison with same action

The first comparison we run is within an action, comparing the meaning for different positions, when generating a sentence from a certain motor action. We can see such a

comparison in Figure 6.1.

From this figure we can see that the trajectories are not similar, changing with the position of the object, from the far right in Figure 6.1a to the far left in Figure 6.1d. This comparison seems to suggest that either the model generates a different meaning for the same action performed on different positions, or the meaning is not emerging on the meaning layer, but rather somewhere else in the model.

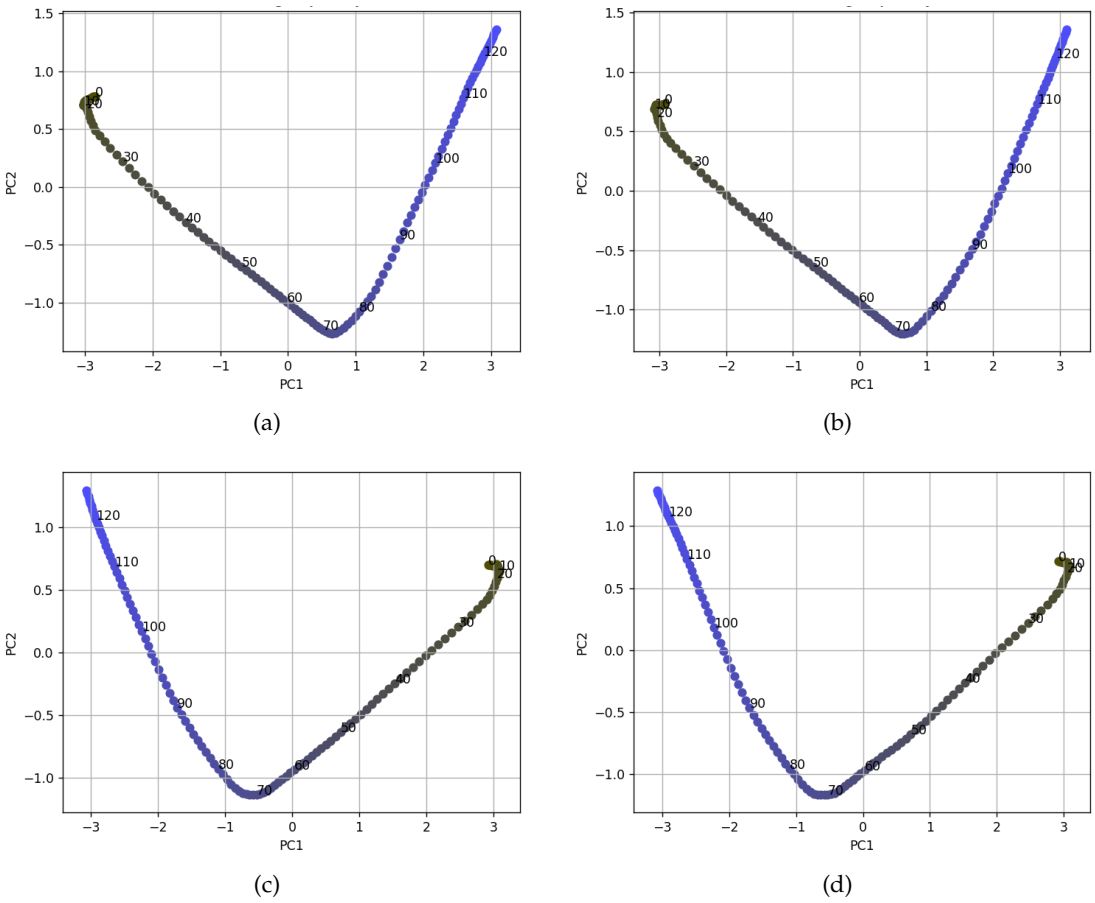


Figure 6.2: PCA scatter plot on the SC layer for the language branch, for the action "give the cereals", performed in 4 different positions: far right (6.2a), slightly to the right (6.2b), slightly to the left (6.2c), and far to the left (6.2d). Dark brown corresponds to earlier timesteps, light blue to later timesteps. Stamps of the corresponding timesteps are indicated in the graph for every 10 timesteps. These trajectories were generated when the model was generating a sentence from a motor action.

To exclude the second option we explore the subsequent layers, namely the language SC layer and the motor action SC layer. Starting with the motor action SC layer, we plotted the trajectories for that layer and compared them. These trajectories, much like the trajectories in the meaning layer, were very different from each other, and thus provided no further insight into the hypothesis.

We proceeded to analyse the SC layer for the language branch, and again plotted the

trajectories for this layer. These trajectories can be seen in Figure 6.2.

In the plots of the SC layer for the language branch we can see that the trajectories are very similar between each other. This layer is, thus, a strong candidate for the emergence of meaning. To fully verify this, however, two more points need to be checked: comparison between directions, and comparison between two different actions.

Comparison between directions

For the comparison between directions we ran the PCA on the SC layer for the language branch, but this time the model was generating a motor action from a certain sentence. The trajectories can be seen in Figure 6.3.

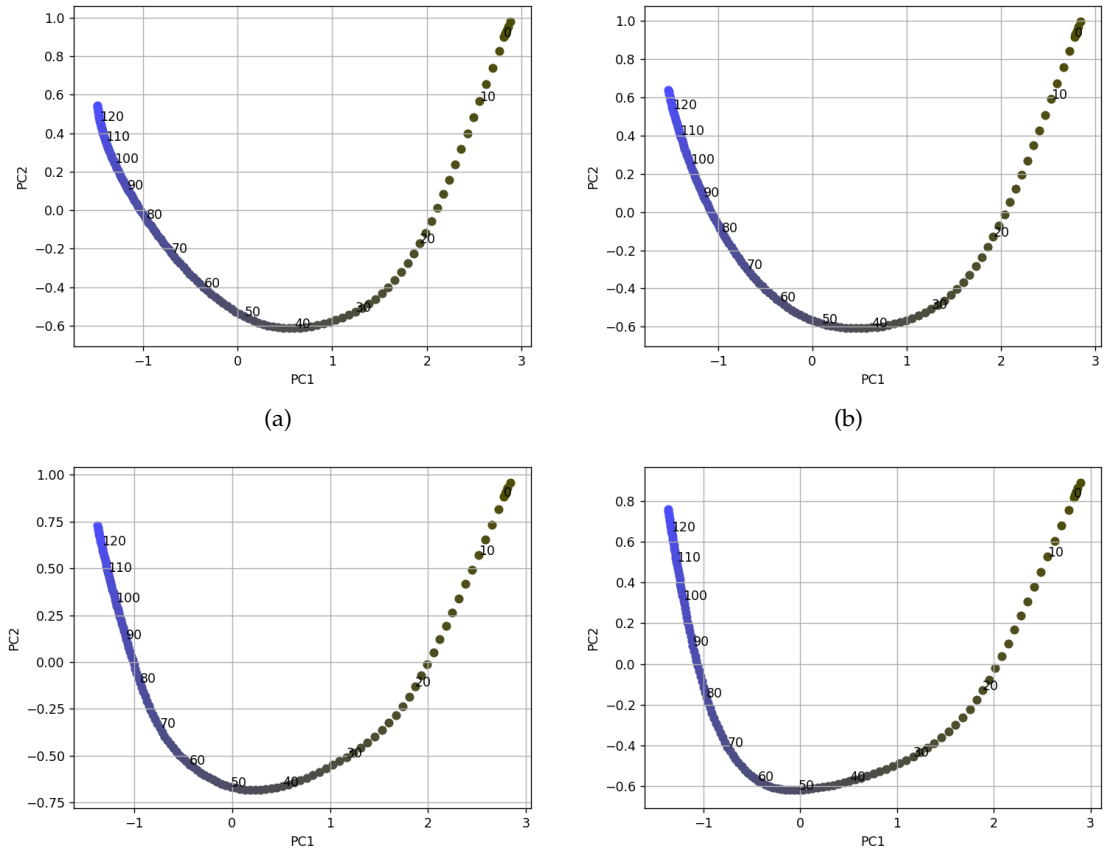


Figure 6.3: PCA scatter plot on the SC layer for the language branch, for the action "give the cereals", performed in 4 different positions: far right (6.3a), slightly to the right (6.3b), slightly to the left (6.3c), and far to the left (6.3d). Dark brown corresponds to earlier timesteps, light blue to later timesteps. Stamps of the corresponding timesteps are indicated in the graph for every 10 timesteps. These trajectories were generated when the model was generating a motor action from a sentence.

From these plots we can see that, while they are still somewhat similar, there are some differences that start standing out: while in 6.3a we see the trajectory ending with a value

around 0.55 for PCA 2, in 6.3d we can see the same point with a value close to 0.8. Another important point is where, in time, the meaning exists: the sentence input ends at timestep 30, at which point the motor action starts. The meaning has to be known by timestep 30, then, for the correct action to be generated. In the plots we can see that timestep 30, and the points just before it (indicated by a darker brown colour), keep the same overall trajectory and values on PC1 and PC2; it is after timestep 30 that bigger changes appear, which would correspond to the duration of the motor action.

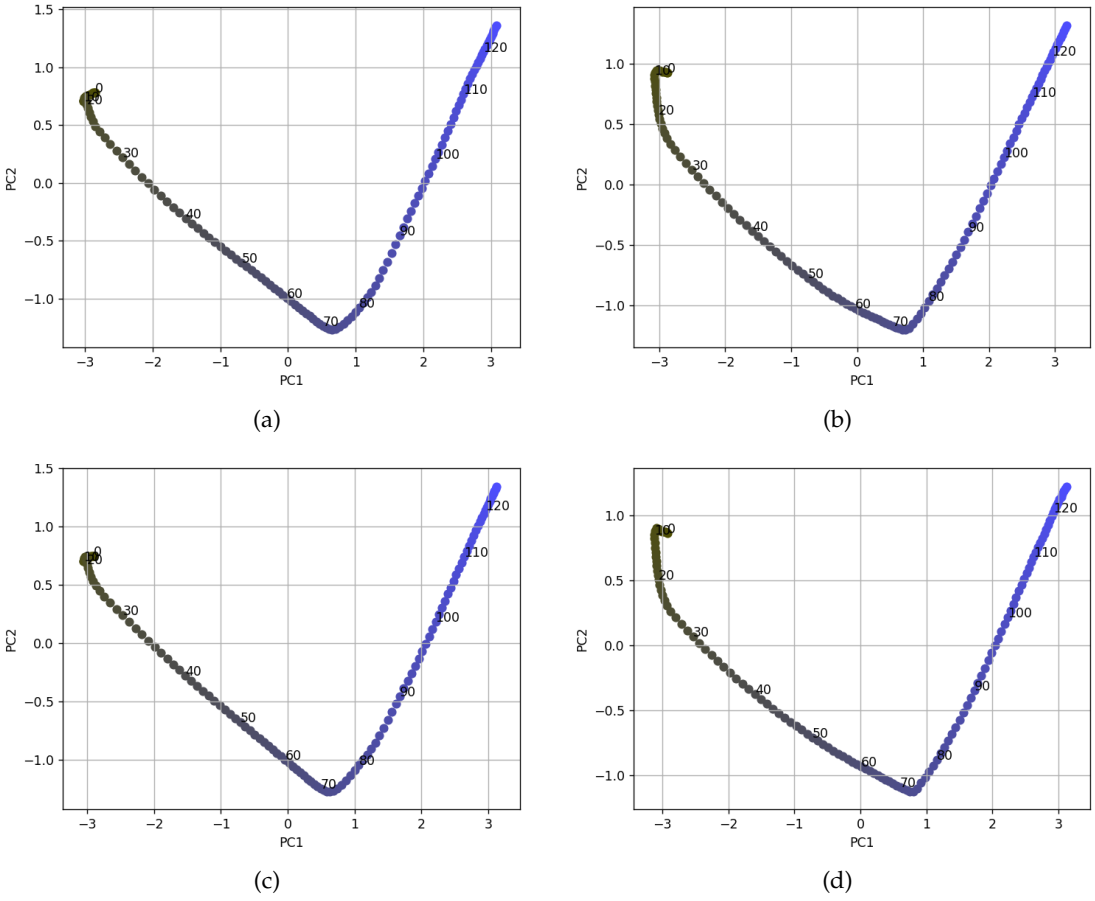


Figure 6.4: PCA scatter plot on the SC layer for the language branch, for the actions "give the cereals" (6.4a), "give the cube" (6.4b), "show the cereals" (6.4c), and "show the cube" (6.4d). Dark brown corresponds to earlier timesteps, light blue to later timesteps. Stamps of the corresponding timesteps are indicated in the graph for every 10 timesteps. These trajectories were generated when the model was generating a sentence from a motor action.

Comparison between actions

Finally, we need to compare the trajectories between actions. For this we choose the actions "give the cereals", "give the cube", "show the cereals" and "show the cube". By making this 4 way comparison we hope to see not only changes due to the actions, but also changes

due to the objects. The plots of these trajectories can be seen in Figure 6.4

In these plots we can see that the object has a very strong influence, with strong variations in the early timesteps between 6.4a and 6.4b. The variations due to the action, which should be visible when comparing 6.4a and 6.4c are much more subtle, to the point where they are indistinguishable. The difference is slightly more visible between 6.4b and 6.4d, where we can see a higher value on PC2 during early timesteps for the action "give the cube" than for "show the cube", but it is still a rather small difference.

The small difference presented here due to changing actions is evident across the whole dataset, with variations due to actions being very subtle.

6.2.2 Testing with synonyms

One other test on the language compositionality and emergence of meaning to be performed in the model has to do with the effect of synonyms.

Synonyms are defined as a set of words that represent the same meaning. In the case of this experiment, we will treat synonyms as verbs that are spelled differently, but translate into the same motor action.

To test this we created a synonym for the verb "pull", which we named "drag", connected to the same motor sequence. We trained the model with a simplified dataset, comprising 10 actions ("push", "pull", "drag", "slide left", "slide right", "point at", "grasp", "smile", "frown" and "show") on 9 objects ("cube", "box", "ball", "car", "bottle", "dog", "carrots", "bear", "fish"). A single motor action is linked to each combination, with the same motor action being connected to both "drag" and "pull".

As in Chapter 5 we trained this model 5 times, picking the best of the trained models. The best model reached an average loss of 0.452614 at epoch 69993.

We tested the ability of the model to reproduce the training data, with the result of 13 wrong sentences with errors of up to 4 letters. The error on the motor action side was negligible. These results can be visualised in Figure 6.5.

While 13 wrong sentences out of a total of 90 is a large percentage, we should keep in mind that at the very best case the model can pick the wrong verb when trying to decide between the synonyms, which we consider a desirable error. The worst case scenario would see the network trying to average the output on the language side, leading to

Figure 6.5: Plot of the error for sentence generation (blue, right axis) and motor action generation (red, left axis) when trained with the synonym dataset.

incomprehensible verbs, which we consider undesirable errors. In order to verify which hypothesis is confirmed, we printed the wrong sentences, and present them in Table 6.1.

target	output
pull the car.	**d**ull the car.
pull the box.	**d**ull the box.
pull the ball.	**d**ull the ball.
pull the bottle.	**d**ull the bottle.
drag the fish.	**d**ull the fish.
pull the dog.	**drag** the dog.
drag the ball.	**d**ull the ball.
drag the car.	**d**ull the car.
drag the box.	**d**ull the box.
drag the bear.	**pull** the bear.
drag the carrots.	**pull** the carrots.
pull the fish.	**d**ull the fish.
drag the bottle.	**d**ull the bottle.

Table 6.1: Table presenting the results of sentence production for the 13 erroneous sequences. Mistakes are highlighted in bold.

In the table we can see that out of the 13 erroneous sentences, 10 were due to an "averaging", where the network confuses "drag" with "pull", creating a mixed "dull" verb. This is undesirable as the model is unable to pick one or the other when generating a sentence. We have 3 replacement errors, where the network picked the wrong verb, but spelled it correctly. We consider these last errors as desirable, since it shows the network is able to reproduce a correct synonym, and thus would still be understandable to a human.

Interestingly, the averaging errors seem to be connected to the objects: when the objects

"car", "box", "ball", "bottle" or "fish" are the target of the action, the model averages the verb to "dull". The same is not verified for the replacement errors. In addition, the averaged verb seems to be closer to the verb "pull" than the verb "drag", suggesting the model could eventually converge to that verb with further training.

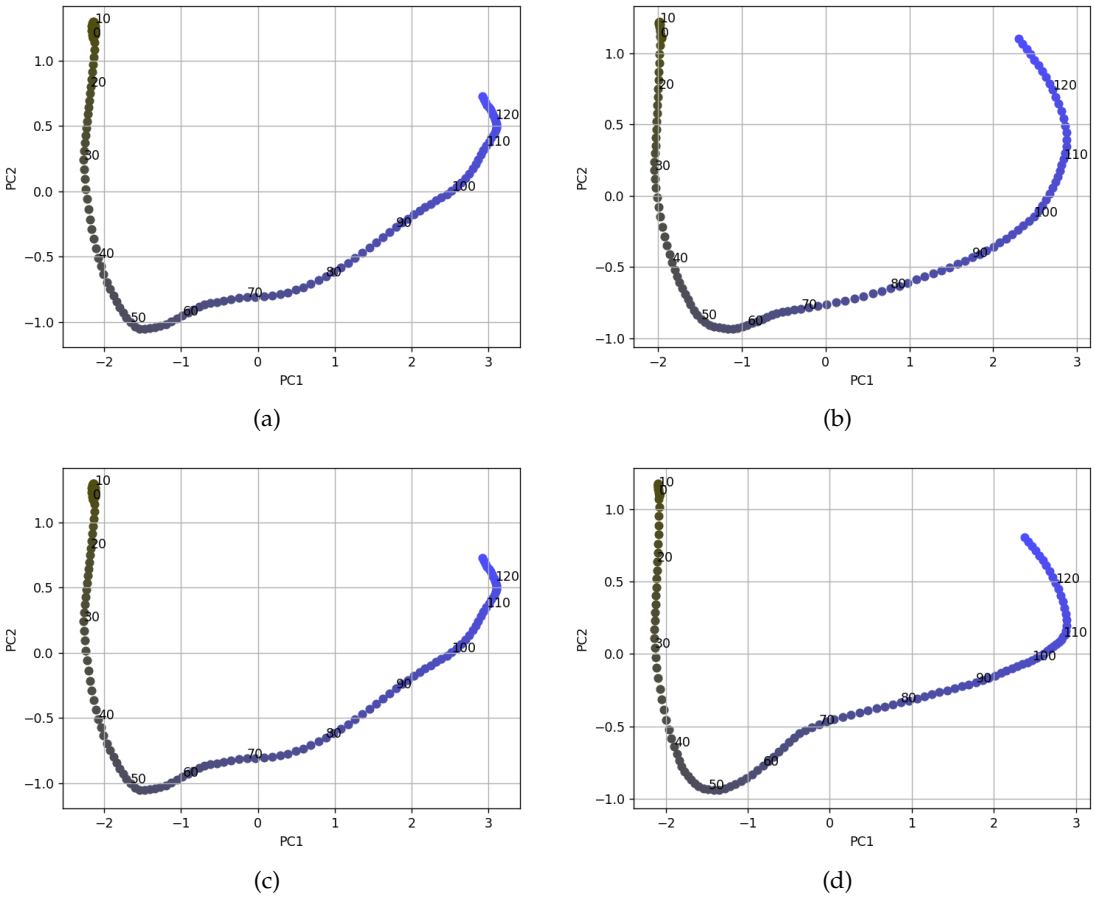


Figure 6.6: PCA scatter plot on the SC layer for the language branch, for the actions "drag the bottle" (6.6a), "drag the cube" (6.6b), "pull the bottle" (6.6c), and "pull the cube" (6.6d). Dark brown corresponds to earlier timesteps, light blue to later timesteps. Stamps of the corresponding timesteps are indicated in the graph for every 10 timesteps. These trajectories were generated when the model was generating a sentence from a motor action.

analysis of meaning

To further investigate the behaviour with synonyms we analysed the meaning in the higher-level layers of the model. In Section 6.2 we found that the best candidate for the layer where meaning emerges is the SC layer on the language branch, and thus we explore the trajectories in this layer. We investigated the trajectories for the actions "drag the bottle", "drag the cube", "pull the bottle" and "pull the cube", with two of them being generated correctly while two others had the same "averaged" verb. The PCA trajectories

for these sequences can be seen in Figure 6.6.

The trajectories suggest that the correct generation of a sentence happens only when the meaning is distinct. Indeed the model seems to create a third meaning for the averaged verb, common to both cases 6.6a and 6.6c.

In the opposite direction, when generating a motor action from a certain sentence, it should be easier for the model to learn the common meaning to the two verbs. We plot the trajectories for these cases, as can be seen in Figure 6.7.

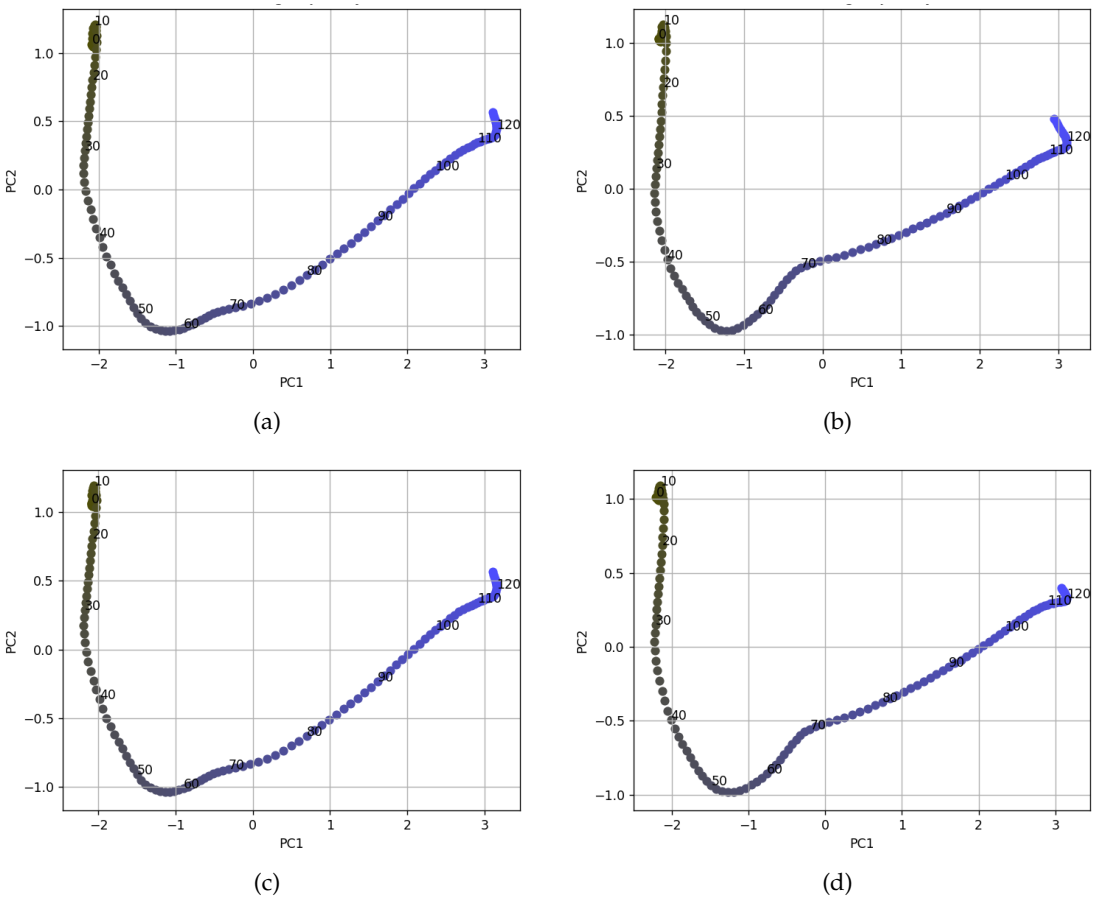


Figure 6.7: PCA scatter plot on the SC layer for the language branch, for the actions "drag the bottle" (6.7a), "drag the cube" (6.7b), "pull the bottle" (6.7c), and "pull the cube" (6.7d). Dark brown corresponds to earlier timesteps, light blue to later timesteps. Stamps of the corresponding timesteps are indicated in the graph for every 10 timesteps. These trajectories were generated when the model was generating a sentence from a motor action.

As expected, the plots for "drag the bottle" (6.7a) and "pull the bottle" (6.7c) are extremely similar, as in image 6.6. However, we can see that also the plots for "drag the cube" (6.7b) and "pull the cube" (6.7d) are extremely similar, particularly in the initial timesteps, while the sentence is still being processed. While this is not unexpected, given that the error for action generation was very low, it reinforces the idea that should meaning emerge, it

should be in this layer.

6.3 Discussion

In this chapter we have analysed the inner workings of the final MTLSTM model that connects language, objects and motor actions, and tried to find a possible source of "meaning" that connects the different information sources. While some strong suggestions of the presence of such meaning structures are present, it is definitely not certain that it emerges in the model, or that the structures being investigated are the correct representations of such meaning.

For meaning to emerge in the model it should be at the link between the different sources of information, and should be deeply connected with the compositional behaviour of the model. Only by separating the meaning of "object" and "verb" would the model be able to compose new combinations, and generate the respective action.

This hypothesis led to the natural candidate for the emergence of meaning to be the layer defined as "meaning layer", at the "physical" intersection of the three different branches. The analysis of this layer, however, presented no evidence of such meaning emerging, with each combination generating its own, unique trajectory, distinct from any other.

Further analysis into the SC layers of both motor action and language branches provided some more insight, with the SC layer in the language branch emerging as a candidate. The analysis of this layer revealed a strong influence from the different objects, suggesting that the meaning of the object is formed at this level; the meaning for the verb was less visible, however.

Testing the model for its behaviour when learning synonyms had mixed results. On one hand, the model proved somewhat capable of learning such synonyms, and there are strong hints for better performance with more training and/or data; on the other hand, analysing the emergence of meaning in this model did not provide more insight into where to look, or how such meaning should look like.

The results obtained with these models suggest that such a link, and therefore such "meaning", exists in the model, behoving further analysis of these models. One possibility is that meaning is not located in any particular layer, but is rather a combination of the activations of cells of different layers, particularly the SC layers and meaning layer, over

time. Such a structure would be extremely hard to find and visualise; however, it is important to take such considerations when searching for meaning, independently of the models being tested.

A different source of information that might clear up the search for meaning could be further research in neuroscience and psychology. The mechanisms through which humans create symbols to represent their perception, generating a meaning for these symbols and their derivatives, are not fully understood, and further insight into the structure of such symbolic structures could help the search for meaning in these AI models.

6.4 Conclusion

In this chapter we have analysed in more detail the MTLSTM models developed during the thesis. We have searched for structures in the internal states of the model that showed evidence of the emergence of meaning, with some useful conclusions.

We have tested the model for its ability to learn and reproduce synonyms, further analysing the emergence of meaning in such cases, and linking the errors in sentence generation to these ambiguous meaning representations. The model is able to link synonyms to the correct motor action, however it is still a challenging task to generate the correct sentences for the verb, and the nature of this difficulty might be in the learning of the meaning itself.

The questions raised by these experiments, namely how is meaning represented in these models, and how does meaning affect synonyms and vice-versa, are important questions for future research in grounding. While we have shown some experiments with some useful insights, in this chapter, it is not yet clear where meaning emerges, or how it is represented.

Chapter 7

Discussion

In Chapter 2 we discussed the fields to be investigated for future robotics, namely in language grounding, and discussed an approach to research in this field through developmental robotics, coupled with research in neuroscience and psychology. Using a developmental robotics approach, in Chapter 3 we implemented and tested a bidirectional training algorithm that allows a single model to learn to categorise and generate a motor action or a sentence. This algorithm was tested with MTRNN, providing a first model for the grounding of actions and language, separately. In Chapter 4 we expanded on the MT architecture, implemented a MTLSTM model that is able to link actions to language directly. We trained this model in a bidirectional way, getting the first model that truly connects motor actions with language. In Chapter 5 we implement the data collection modules that allow for the collection of any raw data with two robots, Pepper and iCub, and demonstrate that the MTLSTM model can learn this data from either robot. Finally, in Chapter 6 we analyse the model for the emergence of meaning, presenting some results that indicate the emergence of some structures linked to meaning. Through the work developed in these chapters we gather support for the main proposal of this thesis:

> A robot can learn to recognise natural language commands and execute them, and to verbalise the action it is doing, being taught like a child, without any explicit prior knowledge.

In Chapter 3 we design and implement the bidirectional training module that allows the same model to link in a bidirectional way two different data sources. We perform a first test using MTRNN to link actions and language to a CS and vice-versa, performing the first basic grounding experiment. We demonstrate the power of this method by linking a

motor action MTRNN model with a language MTRNN model, by means of shared CS, and generating a motor action by writing up a sentence.

In Chapter 4 we introduce the first functioning full MT model, using MTLSTM to link motor actions and language without the need for explicit CS. This chapter demonstrates the power of bidirectional training together with MT models to successfully link the two information sources without any need for any intermediate connections. This experiment represents the first truly self-learned grounding of actions and language.

Chapter 5 steps slightly away from model implementation and focuses instead on data acquisition. The requirement of no explicit prior knowledge means very little data processing should be performed before training the model, and being taught like a child requires that any person, through speech and physical interaction, should be able to teach the robot. We present modules that enable this data collection, validating the acquired data on the MTLSTM models, not just using different robots, Pepper and iCub, but also using different dataset sizes. The data was collected in a relatively unstructured way, simply by presenting objects to the robot in relative locations and talking to the robot, allowing any person, tech-savvy or not, to teach the robots.

Finally, in Chapter 6, as a corollary to the behaviour of the model, we explore the emergence of meaning, establishing the link between grounding and meaning, and searching for such structures in the model. While finding meaning in the model proved challenging, starting from the not-at-all-clear definition of what meaning should look like, we did find common structures between actions and objects, suggesting that emergence of meaning does happen.

In this Chapter we link the results presented in previous chapters, providing an overall perspective to the work done in this thesis. Section 7.1 discusses how the research questions presented in Chapter 1 were answered by the different experiments. Section 7.2 discusses the main limitations of these modules, along with the possible solutions present in current literature. Section 7.3 discusses the main contributions of this thesis towards the different fields, along with the important questions raised by these experiments. Finally, Section 7.4 discusses future paths for further studies in this area, both in the presented models and in the general area of grounding.

7.1 Research Questions

In this section we will review the research questions presented in Section 1.2 and link the work developed in this thesis to them.

RQ1: **What neural models would allow for the grounding between actions and language?**

In investigating how children learn language and actions, and how to create models that learn like children, two key points emerge: the model should exhibit compositionality in generating new sentences from what it learns; and the model should be able to generalise to new motor actions. The literature review offered some options for such models, namely in the form of MTRNN. Furthermore, these models have been used previously in both language tasks and motor action tasks. Using these models as a base, we tested the bidirectional algorithm, proving they can also be trained in a bidirectional way. We further expanded on the MT concept by developing a novel model, MTLSTM, that retains the compositional and generalisation behaviour, while being more scalable and not suffering from vanishing gradient. This model was trained to link actions and language directly, providing the first model to fully connect the two sources of information, and grounding them. We demonstrate the importance of bidirectional training, multiple timescales, and hierarchical architecture of RNN in models that ground language and actions, showcasing the efficiency of this model, particularly compared to baseline simple LSTM models which are unable to effectively learn this data.

RQ2: **Can existing neural models be trained in a bidirectional way, generating a mapping from language to actions and simultaneously from actions to language?**

The presence of mirror neurons and their activation not only when performing an action, but also when discussing or visualising an action, strongly suggest that grounding emerges from bi-directional links between language, visual perception and motor perception. Using this suggestion as a basis, we implemented the Bidirectional training algorithm which is one of the key points of this thesis. In testing the Bidirectional training algorithm in MTRNN models, in Chapter 3, we answered the research question for the basic MTRNN tasks, connecting CS to motor actions and language. In Chapter 4 we prove that a MTLSTM model can indeed be trained

in a bidirectional way to directly map language to actions and vice-versa, validating the model and training by generating both training and test data correctly. In successfully completing this experiment we answer this research question with an unequivocal confirmation.

RQ3 : **Can this model generalise its learned skills to new data, and how does it compare with children?**

In Section 2.2 we separate the concept of "generalisation" into three different domains: physical, language and grammar. In the physical domain, the model should generalise motor actions to different positions and objects; in the language domain the model should be able to generalise learned verbs to new objects, exhibiting compositionality; and finally, on the grammar domain, generalise the structure of grammar, learning verbs faster.

In generalising in the physical domain, the model was capable of generalising to all test sequences as described in Chapter 4, comprising new combinations with objects in positions not trained before. In the language generalisation domain, we have tested the model to generate the corresponding sentences when performing an action-object combination that it had not been trained with, successfully generating the corresponding sentence; finally, by exhibiting a good separation between object and verb in Chapter 4, the model exhibits good grammar separation, being able to link the correct segments in the sentence to the motor action, and therefore to the verbs. Furthermore, tests in Chapter 5 with intransitive verbs show that the model is able to learn not to include an object when verbalising an intransitive verb, even if an object is being seen.

It is unclear if the model is able to learn like a child. While the model was successful when learning datasets of different sizes and complexity, the model didn't always converge in the 5 trained models used in our experiments. This indicates that when trained with increasingly larger datasets, the model will have a tendency to eventually diverge. While this can be mitigated with multiple iterations of training in the same model every time the dataset is increased, this is a technical solution, and therefore it should not be stated that the model learns like a child. In addition to this, constructivist theories of language learning in children are divided between "verb-island" theory (Tomasello (2009)) and the theory that children generalise from the

go, not needing any critical amount of vocabulary (Ninio (2003)). In comparing with these theories, the model fits Ninios description, exhibiting generalisation behaviour independently of the size of the dataset, or vocabulary, it is trained on. Increasing the size of the dataset could increase the generalisation ability of the model, but the behaviour is present independently of the amount of data it is trained with.

RQ4: **Can the robot be taught to verbalise what action it is doing?**

This research question is deeply linked with RQ2, in that if a model is able to generate such mapping, it should be able to verbalise the action it is doing. In answering RQ2, then, we indirectly answer RQ4, and the experiments in Chapters 4 and 5 show that the robot is able to verbalise the action it is performing, accurately, and even generalising for combinations of verbs and objects that it did not explicitly learn. When learning synonyms, as demonstrated in Chapter 6, the model tries to verbalise the action with either verb, sometimes using an "averaged" verb which is a mixture of the synonyms. This experiment suggest, however, that such "averaged" verb could eventually converge to either of the synonyms, since the output verb was a single letter away from one of the synonyms. With these experiments, we can say that the robot can be taught to verbalise the action it is performing.

7.2 Limitations of the model

During our experiments, an initial MTRNN model was used to demonstrate the capabilities of the Bidirectional training. This model was then upgraded to a MTLSTM, based on a similar Multiple Timescales architecture but using LSTM instead of regular RNN neurons. Finally, the MTLSTM model was expanded to consider three different inputs. We will discuss the limitations of each of these models, individually, along with the dataset acquisition modules.

7.2.1 MTRNN

MTRNN have been used extensively in the literature, often used to process sequences of data that exhibits compositional structures, such as music, language, or motor actions. These models suffer from vanishing gradient when scaled up, however, with gradients during training nearing 0, which leads to a model that is not learning. This restricts these

models to learn short sequences, and makes it very hard to scale the model to consider inputs from both language sequences and motor action sequences.

In the first experiment with a full model, we linked two MTRNN branches by means of CS, allowing such a model to generate motor actions from sentences and vice-versa. This model, however, required explicit CS, thus limiting the training data to the size of the CS. One alternative to allow such a model to learn its own CS representation would be to train both branches simultaneously, while allowing the training of the control sequence. This would be an incredibly complex system, however, where not only are the weights of each individual branch being trained, but also the CS at the output of the SC layer, and in addition these learned CS have to be matched between motor action and language branches.

7.2.2 MTLSTM

This novel model, created for this thesis, takes the multiple timescales concept and applies it to LSTM. Using this model, as shown in Chapter 4, we were able to circumvent the vanishing gradient problem, and thus implement the full MTLSTM model to connect actions and language. This came at a cost, however, as a simple CTLSTM cell has 4 different parameters for each parameter in a CTRNN neuron, thus increasing the computation time required to train such a model, along with the amount of data required.

While the simple MTRNN models used in Chapter 3 took up to 3 days to train, the MTLSTM model used in Chapter 4 could take up to two weeks to complete the training of the model. This computation time is prohibitive when running multiple tests on the model, since each different model requires an additional 2 weeks training.

The main problem of this model, and indeed of most RNN models, is the parallel computation problem. Due to the sequential nature of the data and model, it is very hard to train the model using parallel computing in GPUs, and we thus have to rely on CPUs to train the model, which severely reduces the computational power when compared to other, parallelised AI models. Recent attention-based models (Vaswani et al. (2017)), used particularly in language learning, use parallel computing by allowing the model to look into the future, disregarding the sequential structure of sentences, and thus being able to parallelise the training.

7.2.3 adapted MTLSTM

The model considered here is the model presented in Chapter 5, with three different input sources: motor actions, object labels, and sentences. This model was a first step towards a model that consider three different inputs: visual, motor, and language. As an adaptation from the MTLSTM model, this model suffers from the same limitations described in 7.2.2. During the experiments performed in Chapter 5, particularly with the large dataset collected with the iCub robot, the issue of computation time was exacerbated, preventing thorough testing of the model beyond the experiments presented in this thesis.

There are two main sources for the sequential nature of data: the language input is done via letters, and the motor action sequences. On the language side, one alternative would be to consider word encoding. This would allow for one level of complexity less, and possibly allowing for parallelisation, at the cost of introducing explicit knowledge in the form of existing words in the model. On the motor side, however, it is unclear how to remove the sequential nature of the data - the next timestep in a certain motor action has to depend on the previous one, given the state of the robot and the position of its joints, and the object.

7.2.4 Data collection modules

Data collection modules are, by design, robot specific. In this thesis we developed two different modules, for the Pepper and iCub robots, and therefore only these robots are covered by the data collection.

These modules are very human-friendly, requiring only speech and physical interaction to operate. The data is collected in files, and its processing requires some manipulation in a computer, however, which not every person is capable of. In addition, different robots have different joint spaces, which requires slightly adapting the MTLSTM model to account for this, which again is not within the capabilities of everyone. As such, it is advisable that two people are present during collection, one being tech-savvy enough to handle data manipulation, restarting of the modules and dealing with any other occurrences during collection.

For the experiments in this thesis, object labels were collected based on the experiment being run. No visual processing is yet performed, and therefore a limit on the number of

labels that can be learned was enforced, being 18 for the larger dataset. This is a current limitation of this system - however, visual input data was collected, allowing for future developments, namely in the presence of visual processing pipelines, to circumvent this requirement.

7.3 Impact

7.3.1 Pushing the State of the Art

This thesis describes a novel training algorithm, together with a novel MTLSTM model, to successfully connect actions and language. At the time of publication few other methods were applied to the task of grounding between actions and language, particularly when considering the developmental robotics approach used in this thesis.

In the field of language processing a lot of progress has been done, namely by new models based on attention, but this progress has been focused on text processing, and always using word encoding. When it comes to lower level linguistics, not many other works have explored the issue, with some works exploring the emergence of multi-word utterances, but not connected to the actions of the agent (Pizzuto et al. (2019); Pizzuto & Cangelosi (2019)).

With Chapter 3 we introduce a Bidirectional training algorithm that allows a model to connect two different sources of data. This algorithm is critical for the thesis, and its effect was demonstrated in Section 4.2.5. This algorithm could prove particularly useful in bigger models that deal with several sources of data, and it is an essential component in the work towards grounding in this thesis.

With Chapter 4 and 5 we introduce and expand a novel MTLSTM model, capable of scaling up to 7 layers without suffering from vanishing gradient, and connecting language and actions. This model is a novelty in the literature, with the single different implementation being proposed by Liu et al. (2015). With this model we merge the ability of the LSTM to encode long term dependencies while not suffering from vanishing gradient, with the compositional behaviour of MTRNN. With the introduction of this model we hope to provide more strength towards hierarchical recurrent architectures, and for Multiple Timescales architectures in particular, since they prove critical when learning complex sequential data.

Finally, with the introduction of the data collection modules in Chapter 5, we aim at opening up the possibilities for collecting data with both Pepper and iCub robots, possibly leading to more research in the are of language, vision and action grounding by simplifying data collection. Current data collection is extremely individualised when it comes to robot experiments, with individual labs coming up with their own solutions. By developing a generic data collection module, we hope to streamline this process, making it easier to compare datasets collected across institutions, and thus leading to easier comparison between trained models.

7.3.2 Discussing the nature of Meaning

One particular point of research into grounding, particularly in this thesis, has to do with the emergence of meaning, and its nature.

As was discussed in Section 6.1, the definition of meaning is anything but straightforward. Current hypothesis put meaning at the level of grounding, as something that emerges from the link between symbols, or between symbols and the perception of physical entities. With this definition, meaning can have several natures: when describing an object, meaning is a certain representation, static in time. For an action, or goal, this might not be the case, with meaning linked to different points in the action - our internal representation of meaning of an action could potentially shift in time as we execute that action. It is also not clear how and where humans create these representations of meaning, and no insight can be obtained there when answering this question.

One way to define meaning in a technical way for developmental robotics is a structured abstract representation emerging in a AI model that connects two different sources of data. The key points in this definition are "structured abstract representation" - it is hard to justify the emergence of meaning if concepts that we would consider as having the same meaning, e.g.: the generic action of pushing, has different representations depending, for example, on the position of the target object. Therefore, in searching for meaning, we should aim to find such representations that are invariant to some external properties (e.g.: position) but change based on other input (e.g.: verbs).

We finish this section by stressing the importance of finding a definition, linguistic, philosophical and technical, for meaning, in order to further progress in this area.

7.3.3 Streamlining research

The work developed for this thesis was designed in order to streamline research in this area as much as possible. All code is available publicly, and was programmed in order to be as modular as possible, in order to be used in later experiments.

The MTRNN and MTLSTM models have their own modules that can be called from the training module, and can be changed easily, be that in the number of layers, their sizes, or the timescale factors to be used. The bidirectional training algorithm is also available; however, it is less modular due to the training itself being more domain-specific.

In terms of data collection, the software was designed to be as modular as possible, particularly in the case of the iCub robot, using the YARP framework. Speech input modules and visual data modules are separate and can be changed for any other such pipeline. As with the MTRNN and MTLSTM models, the dataset acquisition modules are also publicly available. As was discussed in 7.3.1, our intentions is towards streamlining the dataset acquisition methods, in order to make it easier to collect data for research into language grounding in the future, along with more easier comparison between models.

7.4 Future Work

This thesis proposed a Bidirectional training algorithm and a MTLSTM model to ground actions and language. Research in this field could be improved with different models, along with improvements in the fields of neuroscience, psychology, and linguistics.

In this section we will explore future research paths split into three different categories: technical, supporting research, and definitions.

7.4.1 Technical Improvements

This thesis introduced three novel technical components: a bidirectional learning algorithm, a MTLSTM model, and data collection modules. The first two are relatively unexplored, being new introductions in the field, while the last could be improved in a variety of ways.

Bidirectional Learning Algorithm

The underlying principle of the Bidirectional learning algorithm is rather simple: by training a model in two different directions, we can exploit to the fullest the bidirectional connections present in fully-connected RNN models. Structures that are common to both data sources can potentially be learned faster, and the model will be more efficient, as described in Section 4.2.5.

While this algorithm proved crucial to the work developed in this thesis, it could be expanded in a variety of ways. Currently, when the model is trained in either direction, the model is updated. This could result in the training in the opposite direction, in the next direction, undoing the update in the previous instance. One way to deal with this is to consider a more integrated bidirectional learning algorithm, where the gradients are fused before performing a single update for both directions. This could potentially lead to less oscillations during training, and a more stable learning process.

During the development of the different models, two different input options were considered: in Chapter 3 batches were created randomly at each iteration, with no guarantee that the entire dataset would be covered by any number of batches. This approach is closer to the random input entries often used when trained machine learning models. A second approach, presented in Chapter 4, created the batches before training, and fed each batch a single time per epoch, guaranteeing that each epoch covered the entire dataset. The first method was more likely to converge to a solution, while the second method provided more stable learning curve; however, should the model be presented with conflicting data, it could potentially be harder to converge using the entire dataset, on the second option, than random batches. Further study in this area could highlight the best option, or potentially find different methods to use with the bidirectional learning algorithm.

Finally, we tested the algorithm only with Multiple Timescale architectures. It would be interesting to explore the effects of bi-directional training in other architectures.

MTLSTM model

The MTLSTM model arose as an attempt to merge the resilience to vanishing gradient from LSTM with the compositional behaviours of MTRNN. The literature provided only a single case of any other such attempt, designed by Liu et al. (2015). In this thesis we sought

to test the model for all the attributes of MTRNN, from compositionality to generalisation behaviour, and extended the model to consider 3 different sources of input: language, labels and motor actions.

The model proved very successful at connection actions to language, and was able to tackle a significantly larger dataset in Chapter 5 with minimal increase in size. Analysing the model for its learning ability, after a reduced accuracy in the Pepper dataset, showed that the model learns better from more rich dataset, such as the one provided by the iCub due to its increased motor capabilities. Such an effect is rather interesting, and could be studied further to verify the relation between language and motor capabilities. Further research in this model could be in the form of a third pipeline, introducing visual processing into the model. It should be noted, however, that the model is already very computationally costly, and further extensions should tackle this problem before further increasing the size of the model.

In this work we have presented the model when tackling the problem of language and action grounding. This model could potentially be used for other tasks, analysing sequences that have both long-term dependencies and compositional behaviours. The current focus is shifting towards attention-based models in the field; however, given the sequential nature of a lot of events in the world, RNN such as MTLSTM will still have a place in research.

Finally, the biggest change in the MTLSTM model was in adapting the LSTM to include the timescale factors. The method described here is different from the one described by Liu et al. (2015), and no comparison exists between these models, or other such models that could potentially link LSTM and timescale factors in different ways. Such a study could provide more insight into the effect of timescales, or even lead to the discovery of a new gated model that increases the performance of RNN models.

7.4.2 Supporting Research

Research in the field of Developmental Robotics is based on previous research in neuroscience, psychology and cognitive science; it comes, therefore, that research in these fields could boost this work.

One possible path for such research could be in the modelling of mirror neurons, that could then be used to create more accurate AI models to learn grounding.

In the field of psychology, and in particular in language learning in children, further study could pinpoint the type of learning in early stages of childhood. This is still an open topic in the field, and future developments could lead to better models of such learning.

In the field of cognitive science, more research into the physical nature of grounding could open up ways to represent it in AI models. In particular, research into the representation systems in the human brain could lead to a better analysis of AI models in search for representations of meaning and grounding.

7.4.3 Definitions

One particular challenge when analysing the model for the emergence of meaning had to do with its definition. The definition, and therefore the physical representation of meaning, is not clear.

In this thesis we raised a possible definition of meaning, deeply linked to how it should be represented in the model. This definition is certainly not all inclusive, and a deeper exploration into this particular issue could lead to a more concise definition, which in turn could guide research in language and action grounding in the future.

7.5 Summary

In this Chapter we brought up the research questions presented in Chapter 1, linking the questions to the work developed in this thesis, and explaining how this work addresses and answers these questions. We have discussed the limitations of the models and algorithms used in this work, how it pushed the state of the art, particularly in the fields of machine learning and grounding of actions and language, and discussed improvements on these models, and possible areas for future research that could improve this work.

Chapter 8

Conclusion

8.1 Summary

In Chapter 1 we presented the thesis of this work:

> **A robot can learn to recognise natural language commands and execute them, and to verbalise the action it is doing, being taught like a child, without any explicit prior knowledge.**

In Chapter 2 we explored the main requirements for a model to answer the main thesis and linked research questions: i) the model has to be able to learn both actions and language; ii) the model has to function in a bidirectional way; iii) the data has to be collected in a human-friendly way; and iv) the model has to learn from raw data, disregarding as much explicit prior knowledge as possible. Based on these requirements we concluded hierarchical recurrent architectures were the most indicated to address this issue, with MTRNN coming up as a promising candidate, having been previously used for both language learning and motor action learning, separately. In addition, MTRNN models had been used for language learning from a very low level, down to phonemes in Heinrich et al. (2013). In Chapter 3, based on this analysis, we opted to start with an MTRNN model, learning language at the letter level and motor actions using robot joints as inputs, using our bidirectional training algorithm to train the model to generate both actions and language, and their corresponding CS. This model successfully learned to link actions and sentences to their respective CS, and further analysis demonstrated its capability to generalise to unlearned combinations. The Bidirectional behaviour was further described

when we linked the motor action model to the language model, connecting them at the CS level, and showcasing an example of a sentence generating a CS, which is then used to generate a motor action.

While the model described in Chapter 3 answered the requirements in very basic terms, an extension should be considered, particularly to obey requirement iv): the model should learn its own internal representation, not relying on explicit CS established previously by a human. To satisfy this requirement we attempted to extend the model, connecting the two branches by an extra layer, as described in Section 3.2.7. This model proved unsuccessful, suffering from vanishing gradient and thus being unable to learn. To overcome this challenge, in Chapter 4, we implemented a novel model, MTLSTM, merging the structure of MTRNN with the cells and behaviour of LSTM, which are resilient to vanishing gradient. We trained the full MTLSTM model with the same data from Chapter 3, this time successfully, and demonstrate how this model exhibits the same compositional and generalisation behaviour of MTRNN. In this Chapter we arrived, thus, at the first model that fully connects motor actions with language, without any explicit prior knowledge.

At this point we have a model that is able to learn both language and motor actions, is trained and behaves in a bidirectional way, and is trained with raw data. This leaves one final requirement: iii) the data should be collected in a human-friendly way. This meant that the interaction with the robot should be done through user-friendly means, namely through physical interaction and speech. In order to achieve this, in Chapter 5, we developed data acquisition modules for both Pepper and iCub robots. Both of these modules operate using solely physical interaction and speech, enabling any person capable of moving the robot and talking to operate and teach the robot any action. We validated the data collected in this way by training the MTLSTM model, demonstrating its ability to learn complex data, learned in a very unstructured way, with different levels of complexity arising from the different motor abilities of the robots. In learning this data, we got more insight into the inner workings of the model, where it learned more accurately data provided by the iCub robot than Pepper robot. Further analysis into this showed that the higher complexity of the iCub robot actually provides richer data for the model to learn. This link between motor capabilities and the ability to learn both language and actions is quite interesting, and could be further investigated.

The inspiration in neuroscience and psychology behoves the research into the emergence of meaning in the model. To study this, in Chapter 6 we analyse the hidden states of the model during its operation, searching for structures that could indicate such a meaning has emerged. While some structure is indeed found, particularly for objects, it is rather challenging when it comes to actions, raising the issue of the definition of meaning. Further analysis into the effect of synonyms shows that different verbs are linked to different meanings, even if related to the same motor action, and that the model converges to a third additional "averaged" meaning, generating a verb not present in the dataset that is a mixture of the two synonyms, in some cases.

With the final model introduced in Chapter 5, trained in a bidirectional way, with data collected with the modules also introduced in Chapter 5, we achieve the main thesis of this work, verifying all requirements.

8.2 Contributions

As was described in Chapter 1, this work led to the following contributions:

- **Implementation of Bidirectional Training algorithm.** One of the main contributions of this work, presented in Chapter 3, saw the first implementation of the bidirectional training algorithm, showcasing the bidirectional behaviour of the model when trained in this way. This is one of the key components of the thesis, allowing the subsequent models to be trained to generate an action from a sentence and vice-versa. Code available online (https://github.com/AlexAntn/MTRNN_experiments)

- **Design and implementation of Multiple Timescales Long-Short Term Memory (MTLSTM).** Implemented in Chapter 4 and extended in Chapter 5, this is the model that learns to ground actions and language. Trained in a bidirectional way, this model allows a single pipeline from sentences to action and vice-versa, without any explicit prior knowledge in the model. This model is the second key component of the thesis, and is the model upon which all subsequent experiments were ran. Code available online (https://github.com/AlexAntn/MTLSTM).

- **User-friendly data collection modules.** In order to easily collect data from both iCub and Pepper robots, we implemented in Chapter 5 two data collection modules that function solely based on speech and physical interaction, allowing any user,

even if not tech-savvy, to teach the robot actions. The data collected with this module is used in the two last experiments, and combined also intransitive verbs and emotions.

- **The search for meaning.** We analysed the emergence of meaning in the model in Chapter 6, analysing the higher-level layers to find structures that could indicate the presence of meaning. While the behaviour of the model indicates such information is present, and the visualisation of trajectories corresponding to the objects, the structures related to the verbs and actions were more challenging to spot, sparking the discussion on the definition, and structure, of meaning itself. We tested also the behaviour of the model when learning synonyms and the effect on the meaning representations from such verbs.

8.3 Conclusion

To support the thesis "A robot can learn to recognise natural language commands and execute them, and to verbalise the action it is doing, being taught like a child, without any explicit prior knowledge." I propose the MTLSTM model, trained in a bidirectional way. The MTLSTM model successfully linked actions and language, trained in a bidirectional way to generate sentences from a motor action and vice-versa. In developing this model, the bidirectional algorithm to train it, and studying the model for its generalisation and compositionality behaviours, we expanded on the field of research in grounding of language and actions, along with the introduction of the novel model in the field of machine learning. In studying the inner workings of the model, we attempted to find signs of the emergence of meaning, finding signs of such meaning for the objects, but facing a bigger challenge in identifying such structures for verbs and actions.

The introduction of data collection modules, relying entirely on speech and physical interaction, allows any person to contribute towards teaching the robot, opening the module to non-tech-savvy people, and leading to the possibility of richer datasets in the future.

By studying and demonstrating the ability of the MTLSTM model to learn the data, generalise to new sequences, and compose new sentences, I confirm the applicability of this model and the bidirectional training for the grounding of actions and language.

Acronyms

AI Artificial Intelligence. 2, 6, 7, 27, 104, 107, 110, 111

ANN Artificial Neural Network. 2, 8, 10, 29, 31, 32

CS Control Sequence. xi, xv, 14, 28, 30–32, 35–45, 47, 54, 55, 57, 58, 99–101, 104, 113, 114

CTLSTM Continuous Timescale Long Short-Term Memory. 50–54, 64, 104

CTRNN Continuous Timescale Recurrent Neural Network. xi, 28–30, 45, 50–52, 104

FC Fast Context. 30, 32, 36, 54, 57, 64

I/O Input/Output. xi, xiii, 29–32, 36, 37, 57, 64, 68, 72, 80

LSTM Long Short-Term Memory. xii, 7, 19–21, 28, 47, 49–54, 57, 64–66, 101, 103, 104, 106, 109, 110, 114

MT Multiple Timescales. 10, 28, 45, 66, 99–101

MTLSTM Multiple Timescale Long Short-Term Memory. xii, xiii, 12, 13, 26, 49, 50, 52, 54, 55, 57–59, 63–68, 71, 72, 79, 80, 83–85, 96, 97, 99–101, 103–106, 108–110, 114–116

MTRNN Multiple Timescales Recurrent Neural Network. xi, xii, 12–14, 19, 20, 23, 26–32, 36–38, 40, 45, 47, 49–51, 54, 55, 57, 99–101, 103, 104, 106, 108–110, 113, 114

RNN Recurrent Neural Network. 7–9, 12, 19–21, 27–29, 31, 50, 51, 101, 103, 104, 109, 110

SC Slow Context. xiii, xiv, 30–32, 36, 45, 54, 57, 64, 72, 88–91, 94–96, 104

Bibliography

Antunes, A., Jamone, L., Saponaro, G., Bernardino, A., & Ventura, R. (2016). From human instructions to robot actions: Formulation of goals, affordances and probabilistic planning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 5449–5454). IEEE.

Antunes, A., Laflaquière, A., & Cangelosi, A. (2018). Solving bidirectional tasks using mtrnn. In *2018 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, (pp. 19–25). IEEE.

Antunes, A., Laflaquière, A., Ogata, T., & Cangelosi, A. (2019). A bidirectional multiple timescales lstm model for grounding of actions and verbs. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ.

Antunes, A., Pizzuto, G., & Cangelosi, A. (2017a). Communication with speech and gestures: applications of recurrent neural networks to robot language learning. In *Proceedings of GLU 2017 International Workshop on Grounding Language Understanding*, (pp. 4–7).

Antunes, A., Saponaro, G., Morse, A., Jamone, L., Santos-Victor, J., & Cangelosi, A. (2017b). Learn, plan, remember: A developmental robot architecture for task solving. In *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, (pp. 283–289). IEEE.

Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., & Yoshida, C. (2009). Cognitive developmental robotics: A survey. *IEEE transactions on autonomous mental development*, *1*(1), 12–34.

Asada, M., MacDorman, K. F., Ishiguro, H., & Kuniyoshi, Y. (2001). Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous systems*, *37*(2-3), 185–193.

Bloom, L. (2013). *One word at a time: The use of single word utterances before syntax*, vol. 154. Walter de Gruyter.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Bown, O., & Lexer, S. (2006). Continuous-time recurrent neural networks for generative and interactive musical performance. In *Workshops on Applications of Evolutionary Computation*, (pp. 652–663). Springer.

Brighton, H., Smith, K., & Kirby, S. (2005). Language as an evolutionary system. *Physics of Life Reviews*, *2*(3), 177–226.

Brooks, R. A., Breazeal, C., Irie, R., Kemp, C. C., Marjanovic, M., Scassellati, B., & Williamson, M. M. (1998). Alternative essences of intelligence. *AAAI/IAAI*, *1998*, 961–968.

Cambria, E., & White, B. (2014). Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, *9*(2), 48–57.

Cangelosi, A. (2006). The grounding and sharing of symbols. *Pragmatics & Cognition*, *14*(2), 275–285.

Cangelosi, A., Greco, A., & Harnad, S. (2002). Symbol grounding and the symbolic theft hypothesis. In *Simulating the evolution of language*, (pp. 191–210). Springer.

Cangelosi, A., & Schlesinger, M. (2015). *Developmental robotics: From babies to robots*. MIT Press.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chomsky, N. (1993). *Lectures on government and binding: The Pisa lectures*. 9. Walter de Gruyter.

Funahashi, K.-i., & Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, *6*(6), 801–806.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, *42*(1-3), 335–346.

Heinrich, S., Magg, S., & Wermter, S. (2015). Analysing the multiple timescale recurrent neural network for embodied language understanding. In *Artificial neural networks*, (pp. 149–174). Springer.

Heinrich, S., Weber, C., & Wermter, S. (2013). Embodied language understanding with a multiple timescale recurrent neural network. In *International Conference on Artificial Neural Networks*, (pp. 216–223). Springer.

Heinrich, S., & Wermter, S. (2014). Interactive language understanding with multiple timescale recurrent neural networks. In *International Conference on Artificial Neural Networks*, (pp. 193–200). Springer.

Hénaff, P., Scesa, V., Ouezdou, F. B., & Bruneau, O. (2011). Real time implementation of ctrnn and bptt algorithm to learn on-line biped robot balance: Experiments on the standing posture. *Control engineering practice*, *19*(1), 89–99.

Hinaut, X., Petit, M., Pointeau, G., & Dominey, P. F. (2014). Exploring the acquisition and production of grammatical constructions through human-robot interaction with echo state networks. *Frontiers in neurorobotics*, *8*, 16.

Hinoshita, W., Arie, H., Tani, J., Okuno, H. G., & Ogata, T. (2011). Emergence of hierarchical structure mirroring linguistic composition in a recurrent neural network. *Neural Networks*, *24*(4), 311–320.

Hirai, K., Hirose, M., Haikawa, Y., & Takenaka, T. (1998). The development of honda humanoid robot. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 2, (pp. 1321–1326). IEEE.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Iverson, J. M., & Thelen, E. (1999). Hand, mouth and brain. the dynamic emergence of speech and gesture. *Journal of Consciousness Studies*, *6*(11-12), 19–40.

Jeong, S., Park, Y., Mallipeddi, R., Tani, J., & Lee, M. (2014). Goal-oriented behavior sequence generation based on semantic commands using multiple timescales recurrent neural network with initial state correction. *Neurocomputing*, *129*, 67–77.

Keysers, C., & Fadiga, L. (2008). The mirror neuron system: new frontiers. *Social Neuroscience*, *3*(3-4), 193–198.

Kim, M., Singh, M. D., & Lee, M. (2016). Towards abstraction from extraction: Multiple timescale gated recurrent unit for summarization. *arXiv preprint arXiv:1607.00718*.

Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). A clockwork rnn. In *International Conference on Machine Learning*, (pp. 1863–1871).

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, (pp. 9–48). Springer.

Liu, P., Qiu, X., Chen, X., Wu, S., & Huang, X. (2015). Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, (pp. 2326–2335).

Lupyan, G., & Bergen, B. (2016). How language programs the mind. *Topics in cognitive science*, *8*(2), 408–424.

Mayor, J., & Plunkett, K. (2010). Vocabulary spurt; are infants full of zipf? In *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 32.

Metta, G., Fitzpatrick, P., & Natale, L. (2006). Yarp: yet another robot platform. *International Journal of Advanced Robotic Systems*, *3*(1), 8.

Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A., & Montesano, L. (2010). The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, *23*(8), 1125–1134.

Mnih, A., & Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, (pp. 641–648). ACM.

Murata, S., Arie, H., Ogata, T., Tani, J., & Sugano, S. (2014a). Learning and recognition of multiple fluctuating temporal patterns using s-ctrnn. In *International Conference on Artificial Neural Networks*, (pp. 9–16). Springer.

Murata, S., Yamashita, Y., Arie, H., Ogata, T., Tani, J., & Sugano, S. (2014b). Generation of sensory reflex behavior versus intentional proactive behavior in robot learning of cooperative interactions with others. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, (pp. 242–248). IEEE.

Ninio, A. (2003). No verb is an island: Negative evidence on the verb island hypothesis. *Psychology of language and communication*, *7*(1), 3–21.

Nishide, S., Nakagawa, T., Ogata, T., Tani, J., Takahashi, T., & Okuno, H. G. (2009). Modeling tool-body assimilation using second-order recurrent neural network. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 5376–5381). IEEE.

Ogata, T., & Okuno, H. G. (2013). Integration of behaviors and languages with a hierarchal structure self-organized in a neuro-dynamical model. In *2013 IEEE Workshop on Robotic Intelligence in Informationally Structured Space (RiiSS)*, (pp. 89–95). IEEE.

Oishi, E. (2003). Semantic meaning and four types of speech act. *PRAGMATICS AND BEYOND NEW SERIES*, (pp. 135–148).

Park, M., Lee, S., & Han, W. (2015). Development of steering control system for autonomous vehicle using geometry-based path tracking algorithm. *Etri Journal*, *37*(3), 617–625.

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, (pp. 763–768). IEEE.

Pastra, K., Dimitrakis, P., Balta, E., & Karakatsiotis, G. (2010). Praxicon and its language-related modules. In *Proceedings of companion volume of the 6th Hellenic conference on artificial intelligence (SETN)*, (pp. 27–32).

Pattacini, U. (2011). Modular cartesian controllers for humanoid robots: Design and implementation on the icub. *Dizertacná práca, Istituto Italiano di Tecnologia*.

Pattacini, U., Nori, F., Natale, L., Metta, G., & Sandini, G. (2010). An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, (pp. 1668–1674). IEEE.

Peniak, M., Marocco, D., Tani, J., Yamashita, Y., Fischer, K., & Cangelosi, A. (2011a). Multiple time scales recurrent neural network for complex action acquisition. *Proceedings of ICDL-Epirob*.

Peniak, M., Morse, A., & Cangelosi, A. (2013). Aquila 2.0 software architecture for cognitive robotics. In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, (pp. 1–6). IEEE.

Peniak, M., Morse, A., Larcombe, C., Ramirez-Contla, S., & Cangelosi, A. (2011b). Aquila: An open-source gpu-accelerated toolkit for cognitive and neuro-robotics research. In *The 2011 International Joint Conference on Neural Networks*, (pp. 1753–1760). IEEE.

Pfeifer, R., & Bongard, J. (2006). *How the body shapes the way we think: a new view of intelligence*. MIT press.

Pfeifer, R., & Scheier, C. (2001). *Understanding intelligence*. MIT press.

Piaget, J. (1954). The construction of reality in the child (m. cook, trans.). new york, ny, us.

Pizzuto, G., & Cangelosi, A. (2019). Exploring deep models for comprehension of deictic gesture-word combinations in cognitive robotics. In *2019 International Joint Conference on Neural Networks (IJCNN)*, (pp. 1–7). IEEE.

Pizzuto, G., Hospedales, T. M., Capirci, O., & Cangelosi, A. (2019). Modelling the single word to multi-word transition using matrix completion. In *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, (pp. 284–289). IEEE.

Pulvermüller, F. (2005). Brain mechanisms linking language and action. *Nature Reviews Neuroscience*, *6*(7), 576.

Pulvermüller, F., & Fadiga, L. (2010). Active perception: sensorimotor circuits as a cortical basis for language. *Nature Reviews Neuroscience*, *11*(5), 351–360.
URL http://www.nature.com/nrn/journal/v11/n5/abs/nrn2811.html

Pulvermüller, F., Hauk, O., Nikulin, V. V., & Ilmoniemi, R. J. (2005). Functional links between motor and language systems. *European Journal of Neuroscience*, *21*(3), 793–797.

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 779–788).

Rizzolatti, G. (2005). The mirror neuron system and its function in humans. *Anatomy and embryology*, *210*(5-6), 419–421.

Rizzolatti, G., Fadiga, L., Fogassi, L., & Gallese, V. (2002). 14 from mirror neurons to imitation: facts and speculations. *The imitative mind: Development, evolution, and brain bases*, *6*, 247–266.

Saffran, J. R. (2003). Statistical language learning: Mechanisms and constraints. *Current directions in psychological science*, *12*(4), 110–114.

Samuelson, L. K., Smith, L. B., Perry, L. K., & Spencer, J. P. (2011). Grounding word learning in space. *PloS one*, *6*(12), e28095.

Sandini, G., Metta, G., & Konczak, J. (1997). Human sensori-motor development and artificial systems. In *International Symposium on Artificial Intelligence, Robotics, and Intellectual Human Activity Support for Nuclear Applications (AIR&IHAS 97), Japan*. Citeseer.

Searle, J. R. (1982). The chinese room revisited. *Behavioral and brain sciences*, *5*(2), 345–348.

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, (pp. 3104–3112).

Tani, J. (2014). Self-organization and compositionality in cognitive brains: A neurorobotics study. *Proceedings of the IEEE*, *102*(4), 586–605.

Taniguchi, T., Nagai, T., Nakamura, T., Iwahashi, N., Ogata, T., & Asoh, H. (2016). Symbol emergence in robotics: a survey. *Advanced Robotics*, *30*(11-12), 706–728.

Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L., & Nori, F. (2008). An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator. In *Proceedings of the 8th workshop on performance metrics for intelligent systems*, (pp. 57–61). ACM.

Tikhanoff, V., Cangelosi, A., & Metta, G. (2010). Integration of speech and action in humanoid robots: icub simulation experiments. *IEEE Transactions on Autonomous Mental Development*, *3*(1), 17–29.

Tomasello, M. (2000). The item-based nature of children's early syntactic development. *Trends in Cognitive Sciences*, *4*(4), 156–163.
URL http://view.ncbi.nlm.nih.gov/pubmed/10740280

Tomasello, M. (2009). *Constructing a language*. Harvard University Press.

Turing, A. (1950). Mind. *Mind*, *59*(236), 433–460.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, (pp. 5998–6008).

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., & Saenko, K. (2014). Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*.

Yamashita, Y., & Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS computational biology*, *4*(11), e1000220.

Yu, Z., Mallipeddi, R., & Lee, M. (2013). Supervised multiple timescale recurrent neuron network model for human action classification. In *International Conference on Neural Information Processing*, (pp. 196–203). Springer.

Yu, Z., Moirangthem, D. S., & Lee, M. (2017). Continuous timescale long-short term memory neural network for human intent understanding. *Frontiers in neurorobotics*, *11*, 42.

Zhong, J., Cangelosi, A., & Wermter, S. (2014). Toward a self-organizing pre-symbolic neural model representing sensorimotor primitives. *Frontiers in behavioral neuroscience*, *8*.

Zhong, J., Peniak, M., Tani, J., Ogata, T., & Cangelosi, A. (2016). Sensorimotor input as a language generalisation tool: A neurorobotics model for generation and generalisation of noun-verb combinations with sensorimotor inputs. *arXiv preprint arXiv:1605.03261*.

Appendices

Appendix A

Learn, plan, remember: A developmental robot architecture for task solving

In parallel to this thesis we worked on an experiment developed in the POETICON++ project in partnership with ISR, Lisbon. This collaboration aimed at integrating work done previously in the project to create an architecture that could choose which objects to learn, and then plan a sequence of actions to execute a task given by a human.

My task in this work was the implementation of the link between the object learning modules developed at Plymouth University and the planning architecture developed in ISR, Lisbon. I implemented a set of functions for communication between the modules, along with the planning rules necessary for the planning architecture to choose which objects to learn. The iCub robot could choose to interact with the object, change the perspective to try and recognise it, or ask a human for the name of the object, activating the learning model from Plymouth University. As a result of this collaboration we published one paper (Antunes et al. (2017b)).