INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Luckeciano Carvalho Melo

DEEP REINFORCEMENT LEARNING FOR HUMANOID KICK MOTION

Final Paper
2018

Course of Computer Engineering

CDU 621.38

Luckeciano Carvalho Melo

DEEP REINFORCEMENT LEARNING FOR HUMANOID KICK MOTION

Advisor

Prof. Dr. Adilson Marques da Cunha (ITA)

Co-advisor

Prof. Dr. Marcos R. O. de A. Máximo (ITA)

COMPUTER ENGINEERING

São José dos Campos
Instituto Tecnológico de Aeronáutica

2018

Cataloging-in Publication Data
Documentation and Information Division

Carvalho Melo, Luckeciano
Deep Reinforcement Learning for Humanoid Kick Motion / Luckeciano Carvalho Melo.
São José dos Campos, 2018.
30f.

Final paper (Undergraduation study) – Course of Computer Engineering– Instituto Tecnológico de Aeronáutica, 2018. Advisor: Prof. Dr. Adilson Marques da Cunha. Co-advisor: Prof. Dr. Marcos R. O. de A. Máximo.

1. Deep Reinforcement Learning. 2. Robotics. 3. Artificial Intelligence. I. Instituto Tecnológico de Aeronáutica. II. Deep Reinforcement Learning for Humanoid Kick Motion.

BIBLIOGRAPHIC REFERENCE

CARVALHO MELO, Luckeciano. **Deep Reinforcement Learning for Humanoid Kick Motion**. 2018. 30f. Final paper (Undergraduation study) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

Luckeciano Carvalho Melo
H8A St., 113
12.228-460 – São José dos Campos–SP

DEEP REINFORCEMENT LEARNING FOR HUMANOID KICK MOTION

This publication was accepted like Final Work of Undergraduation Study

---

Luckeciano Carvalho Melo

Author

Adilson Marques da Cunha (ITA)

Advisor

Marcos R. O. de A. Máximo (ITA)

Co-advisor

Prof[a].Dr[a]. Cecília César
Course Coordinator of Computer Engineering

São José dos Campos: JUNE 12, 2018.

Abstract

Controlling a high degrees of freedom humanoid robot is acknowledged as one of the hardest problems in Robotics. Due to the lack of mathematical models, an approach frequently employed is to rely on human intuition to design keyframe movements by hand, usually aided by graphical tools. In this preliminary work, we propose a learning framework based on neural networks in order to mimic humanoid robot movements. The developed technique does not make any assumption about the underlying implementation of the movement, therefore both keyframe and model-based motions may be learned. The framework was applied in the RoboCup 3D Soccer Simulation domain and promising results were obtained using the same network architecture for several motions, even when copying motions from another teams.

List of Figures

FIGURE 1.1 – AlphaGo Zero, learning model that beat the best players of Go, Chess and Shogi, learning to play without previous human knowledge (SILVER *et al.*, 2017) 9

FIGURE 1.2 – Locomotion of Agent via Deep Reinforcement Learning (HEESS *et al.*, 2017) 10

FIGURE 1.3 – RoboCup Soccer 3D Simulation League 10

FIGURE 3.1 – Artificial neuron and feed forward artificial neural network (TANIKIC; DESPOTOVIC, 2012). 14

FIGURE 4.1 – Architecture of the neural network designed to learn motions. 18

FIGURE 5.1 – The plots of mean squared error and mean absolute error during training. 20

FIGURE 5.2 – Kick motion. The first row of figures shows the original kick motion. The second row shows the learned kick motion. The motions are visually indistinguishable. 21

FIGURE 5.3 – Joint values for comparing original and learned kicks. The neural network was able to fit the joint trajectories with small errors. . . . 22

FIGURE 5.4 – Joints positions during a period of the walking motion for the original walk, the learned walk and the joints positions effectively attained during the learned walking motion. 23

FIGURE 5.5 – Walking motions comparison. The figure (a) shows our agent in its regular walk. The figure (b) shows the same agent mimicking UT Austin Villa walk. The figure (c) is the UT Austin Villa agent itself, performing his own walking motion. 24

List of Tables

TABLE 4.1 – Network Summary 18

TABLE 5.1 – Kick Comparison 22

TABLE 5.2 – Walk Comparison - Forward Walk 24

Contents

1 Introduction 9

1.1 Motivation 9

1.2 Contextualization 10

1.3 Objective 11

1.4 Scope 11

1.5 Organization of this work 11

2 RoboCup Soccer3D Simulation League techniques . . . 12

2.1 Domain Description and Kick Motion 12

2.2 Keyframe Movements 13

3 Deep Learning 14

3.1 Neural Networks 14

4 Methodology 17

4.1 Dataset 17

4.2 Neural Network Architecture and Hyperparameters 17

4.3 Training Procedure 18

4.4 Deployment in Soccer 3D Environment 19

5 Results and Discussion 20

5.1 Training Results 20

5.2 The Learned Kick Motion 21

5.3 The Learned Walk Motion 23

5.4 Other motions 24

6 Conclusions and Future Work 26

6.1 Preliminary Conclusions and Future Work 26

6.2 Activities Plan 26

Bibliography 28

1 Introduction

1.1 Motivation

Robotics is understood as an important area of research within Engineering and Computer Science, optimizing and automating various areas of industry. One of the ways in which research is developed in this area is in robot soccer, since it comprises challenges of machine perception, environment modeling, planning and reasoning, control, and multiagent strategy.

Over the years, several techniques have been developed to address each of the problems related to robot soccer, based on the theory of Signal Processing, Control, Trajectory Planning and classical Artificial Intelligence. These techniques proved to be functional for maturing the challenge. However, they still lacked when compared to human performance in these activities.

In recent years, however, with the development of processing and memory architectures, Machine Learning techniques have been able to achieve or have surpassed human performance in machine perception activities (Computer Vision (LU; TANG, 2014) and Speech Recognition (XIONG *et al.*, 2016)), planning and reasoning (SILVER *et al.*, 2017) (Figure 1.1) and controlling agent locomotion (HEESS *et al.*, 2017) (Figure 1.2). In this way, the learning field, combining techniques of Deep Learning and Reinforcement Learning, appears as a great candidate in search of General Artificial Intelligence.

FIGURE 1.1 – AlphaGo Zero, learning model that beat the best players of Go, Chess and Shogi, learning to play without previous human knowledge (SILVER *et al.*, 2017)

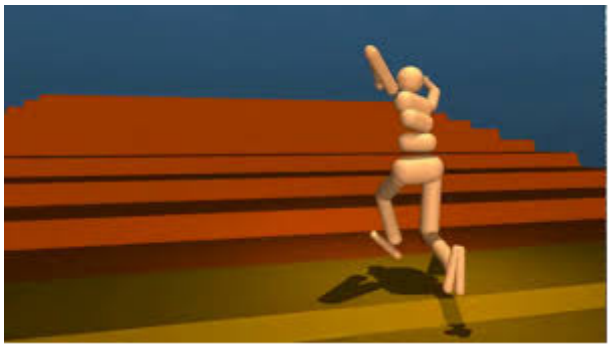


FIGURE 1.2 – Locomotion of Agent via Deep Reinforcement Learning (HEESS *et al.*, 2017)

1.2 Contextualization

RoboCup is an international scientific community aiming to advance the state of the art of intelligent robots. Its mission is that a team of robots will be able to beat the human team champion of the World Cup until the year 2050 (KITANO *et al.*, 1998). Since this is a goal with a range of different challenges to be solved, there are several categories of competition within the community

In Robocup Soccer 3D Simulation League (Figure 1.3), there is a robot soccer competition in which each team has eleven Nao robots in a physical simulation environment called Simspark (OBST; ROLLMAN, 2005). The purpose in this case is to create and improve algorithms and physical models for the perception, locomotion, and strategy of the humanoid robot before actually shipping them into hardware for real-world evaluation.

FIGURE 1.3 – RoboCup Soccer 3D Simulation League

Over the last few years, team efforts have been seen in three areas: the first, more basic, construction of the agent that can model the environment and interact with it – involving both the construction of a solid software architecture and the application of traditional techniques of localization and control of humanoid robot (MACALPINE *et al.*, 2011). The second layer, explored at the highest level, involves creating behaviors to perform actions within the game, given the modeled environment – from the creation of models and heuristics for navigation to the creation of multiagent strategies for positioning and marking (MACALPINE; STONE, 2016).

The third strand, to be approached in this work, is the creation and optimization of

models based on learning for activities such as robot walking and kicking. Simulated categories have great value for learning tests because they provide a benchmark for comparison and do not involve physical robots. Historically, the teams with the best performance in these two issues are usually the best positioned in the category competitions, due to the fact that they are able to maintain greater possession of the ball and are more offensive to the opposing goal.

1.3 Objective

Inspired in the context of RoboCup 3D Simulation League and based on the results of the recent techniques of Deep and Reinforcement Learning, we aim to develop and evaluate new learning models, based on Deep Reinforcement Learning, for the task of making a humanoid robot kick the soccer ball inside the environment simulated category.

1.4 Scope

In this work, Reinforcement Learning algorithms applied to models based on Deep Neural Networks will be approached in order to find, through gradient-based optimization techniques, optimal policies for walking and kick control of the humanoid robot. These will be contrasted with classic control techniques coupled with evolutionary optimization strategies, widely used in the context of RoboCup Soccer 3D Simulation League.

1.5 Organization of this work

The bachelor's thesis is organized as follows: Chapter 2 will describe RoboCup Soccer 3D Simulation League and the traditional methods used for the kick motion. Chapter 3 will cover the theory behind Deep Learning used in this work. Chapter 4 will explain all the methods and tooling used in experimentation. Chapter 5 describes the experimentation itself, detailing problem modeling, test scenarios and results. Finally, Chapter 6 will share conclusions and future work.

2 RoboCup Soccer3D Simulation League techniques

2.1 Domain Description and Kick Motion

RoboCup Soccer 3D Simulation League (Soccer 3D) is a particularly interesting challenge concerning humanoid robot soccer. It consists of a simulation environment of a soccer match with two teams, each one composed by up to 11 simulated NAO robots (GOUAILLIER *et al.*, 2009), the official robot used for RoboCup Standard Platform League since 2008. Soccer 3D is interesting for robotics research since it involves high level multi-agent cooperative decision making while providing a physically realistic environment which requires control and signal processing techniques for robust low level skills.

In the current level of evolution of Soccer 3D, motion control is a key factor in team's performance. Indeed, controlling a high degrees of freedom humanoid robot is acknowledged as one of the hardest problems in Robotics. Much effort has been devised to humanoid robot walking, where researchers have been very successful in designing control algorithms which reason about reduced order mathematical models based on the Zero Moment Point (ZMP) concept, such as the linear inverted pendulum model (KAJITA *et al.*, 2001). Nevertheless, these techniques restrict the robot to operate under a small region of its dynamics, where the assumptions of the simplified models are still valid (COLLINS *et al.*, 2005; MUNIZ *et al.*, 2016).

Therefore, model-based techniques are hard to use for designing highly dynamic movements, such as a long distance kick and a goalkeeper's dive to defend the goal from a fast moving ball. In the robot soccer domain, a common approach for these movements is to employ keyframe movements, where the motion is composed by a sequence of robot postures. In this case, the movement is designed off-line and executed in a open-loop fashion in execution time.

Due to the lack of mathematical models, an approach frequently employed is to rely on human intuition to design keyframe movements by hand, usually aided by graphical tools. However, this process is difficulty, time consuming, and is often unable to obtain high

performance motions given the high dimensionality of the search space. Other possible solution is to use motion capture data from humans (SHON *et al.*, 2005), which has its own challenges due to the fact that the kinematic and dynamic properties of a humanoid robot differs greatly from those of a human.

2.2 Keyframe Movements

Definition 1 *A* keyframe $\mathbf{k} = [j_1, j_2, \ldots, j_n]^T \in K \subseteq \mathbb{R}^n$ *is an ordered set of joint angular positions, where* K *and* n *are the joint space and the number of degrees of freedom of the robot, respectively.*

Definition 2 *A* keyframe step *is an ordered pair* $\mathbf{s} = (\mathbf{k}, t) \in S = K \times \mathbb{R}$*, where* $\mathbf{k}$ *is a keyframe and* t *is the time when the keyframe must be achieved with respect to the beginning of the movement, respectively.*

Definition 3 *A* keyframe movement*, or simply a movement, is defined as* $\mathbf{m} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_\gamma, r) \in M = S^\gamma \times \mathbb{R}$*, where* γ *and* r *are the number of keyframe steps and the speed rate of the movement, respectively. In this representation, we assume the movement starts at time 0 and the first keyframe step represents the robot posture at the beginning of the movement. Therefore,* $t_1 = 0$ *and each time* $t_i, \forall i \geq 2$ *is a time since the beginning of the movement.*

Keyframe movements are executed in an open-loop fashion, where joint positions are computed through interpolation of keyframe steps based on the current time. If the interface to the robot joints is not position-based, local controllers may be used to track the position references issued by the keyframe. For example, in the Simspark simulator, the simulated NAO has speed-controlled joints, therefore we use simple proportional controllers for each joint to track the desired joint positions. To obtain smooth joint trajectories, we interpolate keyframe steps using cubic splines (BARTELS *et al.*, 1987), which are functions of class $\mathcal{C}^2$.

Finally, when a keyframe movement is requested, the joint positions are often far away from the movement's initial joint positions. Hence, simply executing the keyframe movement in this case would result in high joints accelerations, which would probably make the robot fall. To avoid this from happening, a transition movement based on linear interpolation is first employed to bring the joint positions to the initial joint positions required by the keyframe movement.

3 Deep Learning

3.1 Neural Networks

Neural Networks are a learning representation whose goal is to approximate some function f^*. The data collected from an environment encodes an underlying function $\mathbf{y} = f^*(\mathbf{x})$ that maps an input $\mathbf{x}$ to an output $\mathbf{y}$, which may be a category from a classifier or a continue value in regression problems. The neural network defines an approximate mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ by learning the values of the parameters $\boldsymbol{\theta}$ which result in the best function approximation. The Figure 3.1 shows a neural network and an artificial neuron in detail.

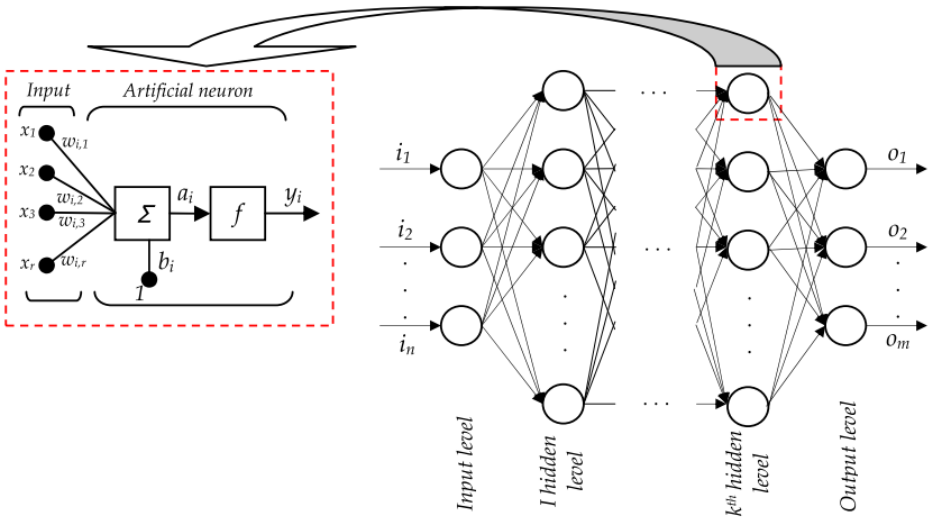


FIGURE 3.1 – Artificial neuron and feed forward artificial neural network (TANIKIC; DESPOTOVIC, 2012).

These networks are typically represented by composing together many different functions, which are associated with a directed acyclic graph describing a computational model. For example, we might have three layers (each of them representing a function $f^{(1)}, f^{(2)}$, and $f^{(3)}$ respectively), connecting in a chain, resulting in a final representation $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$.

During neural network training, the objective is to adjust $f(\mathbf{x})$ to match $f^*(\mathbf{x})$ using

the training dataset, which provides noisy examples of $f^*(\mathbf{x})$ evaluated in different points. The training examples specify directly what the output layer must do at each point $\mathbf{x}$, but the learning algorithm must decide how to use all layers to produce this desired output (GOODFELLOW *et al.*, 2016).

Additionally, we must also choose a learning algorithm to tune this function approximation. In the context of neural networks, gradient-based algorithms are broadly used, especially those based on the backpropagation idea (RUMELHART *et al.*, 1988). The purpose of these algorithms are to propagate the gradient of a cost function through the whole network, in order to minimize the cost function. Most modern neural networks perform this optimization strategy using maximum likelihood, i.e. the cross-entropy between the training data and the model distribution:

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{data}} \log p_{model}(\mathbf{y}|\mathbf{x}) \tag{3.1}$$

In this work, we used the mean squared error loss function in order to fit the dataset. Indeed, we may show that both cost functions are closely related. Let us consider normally distributed errors:

$$p_{model}(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}; f(\mathbf{x};\boldsymbol{\theta}), \sigma^2\mathbf{I}) \tag{3.2}$$

where $f(\mathbf{x};\boldsymbol{\theta})$ and $\sigma^2\mathbf{I}$ are the mean and covariance of this distribution, respectively. Substituting Eq. (3.2) in Eq. (3.1):

$$J(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{data}}\|\mathbf{y} - f(\mathbf{x};\boldsymbol{\theta})\|^2 + const \tag{3.3}$$

The constant term does not depend on $\boldsymbol{\theta}$ and may be dropped. By explicitly evaluating the expectation in Eq. (3.3), we arrive at the mean squared error cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i}^{m}\|y_i - f(\mathbf{x};\boldsymbol{\theta})\|^2 \tag{3.4}$$

Lastly, the gradient of the loss function is taken and propagated through the hidden layers by the chain rule. For example, given $\mathbf{Y} = g(\mathbf{X})$ and $z = f(\mathbf{Y})$, then the chain rule states:

$$\nabla_{\mathbf{X}} z = \sum_j (\nabla_{\mathbf{X}} Y_j)\frac{\partial z}{\partial Y_j} \tag{3.5}$$

This equation is taken recursively until the gradient is propagated to all layers of the

neural network.

4 Methodology

4.1 Dataset

In order to use supervised learning for learning keyframe motions using neural networks, we first need to construct a dataset. A dataset consists of samples of keyframe steps. The samples were collected within Soccer 3D environment with a frequency of 50 Hz. We acquired these samples in two different ways.

In the first one, we commanded an agent of our team to execute specific motions and sampled the reference joint positions computed by our code. In this case, we sampled the kick and get up keyframe motions (MUNIZ *et al.*, 2016). Notice that for this approach to be successful, one needs access to the source code.

The second approach involved changing the Soccer 3D server source code to provide current joint positions of a given robot, in a similar way as described in (MACALPINE *et al.*, 2013). This allowed us to acquire motion datasets from other teams, without any knowledge of how these movements are implemented. In this case, we collected two types of kicks based on keyframes and sampled joint values of the walking engine (MACALPINE *et al.*, 2012).

4.2 Neural Network Architecture and Hyperparameters

The neural network has to be able to learn how to interpolate between samples, which actually happens. The architecture that performed best – in terms of mean absolute error minimization and simplicity – is shown in Figure 4.1. A deep neural network with 2 hidden, fully connected layers of 75 and 50 neurons was used. The output layer has 23 regression neurons, which represent the 22 joint angles and a neuron whose output indicates if the motion has ended or not. The neurons in each hidden layer use the LeakyReLU activation function (XU *et al.*, 2015):

$$f(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

where α is a small constant. This activation function was used to improve the representation capacity of the neural network, adding support for non-linear functions.

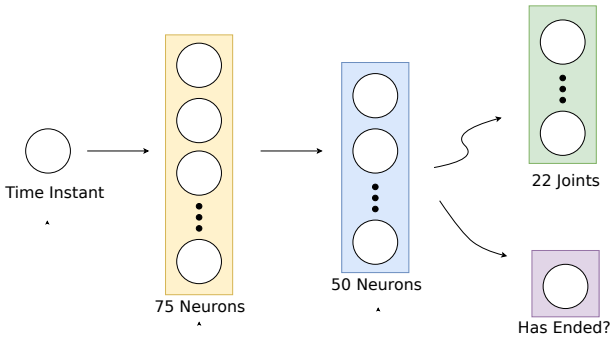


FIGURE 4.1 – Architecture of the neural network designed to learn motions.

This architecture resulted in thousand of parameters to optimize, as exposed in Table 4.1. A very high number when compared to more traditional optimization approaches (MACALPINE *et al.*, 2012). Notice that increasing the number of parameters usually allows representing better movements.

TABLE 4.1 – Network Summary

Layer	**Neurons**	**Activation**	**Parameters**
Dense	75	LeakyReLU	130
Dense	50	LeakyReLU	3800
Dense	23	Linear	1173

Total Parameters	5123

4.3 Training Procedure

Since keyframe motions are executed in an open-loop fashion, the sequence of joint positions are always the same for different repetitions, independently of the robot's state. Therefore, adding samples of multiple executions of the same motion would not make our dataset richer, so we decided to use only one repetition for each movement for faster training. In the case of the walking motion, we collected samples within one walking period.

During training, we used 50 thousands epochs divided in 5 training phases, where the learning rate was decreased between phases in order to achieve better performance. First,

we executed 30000 epochs using learning rate of 0.001. The other phases had 5000 epochs each, and we decreased the learning rate by 0.0002 in each phase.

Furthermore, we used Adam optimization (KINGMA; BA, 2014) during the whole training. The loss function used was the mean squared error, as explained in Subsec. 3.1. We decided this loss function is adequate for this problem because it strongly penalizes large errors, which can collapse the whole motion.

4.4 Deployment in Soccer 3D Environment

In order to perform network design and the training procedure, we used the Keras (CHOLLET *et al.*, 2015) framework coupled with Tensorflow (AL, 2015) as backend. After training, the weights were freezed and converted to a specific format which is readable using the Tensorflow C++ API integrated within the agent's code. Hence, the training is performed outside the environment, but the agent actually computes network inferences during simulation execution.

5 Results and Discussion

5.1 Training Results

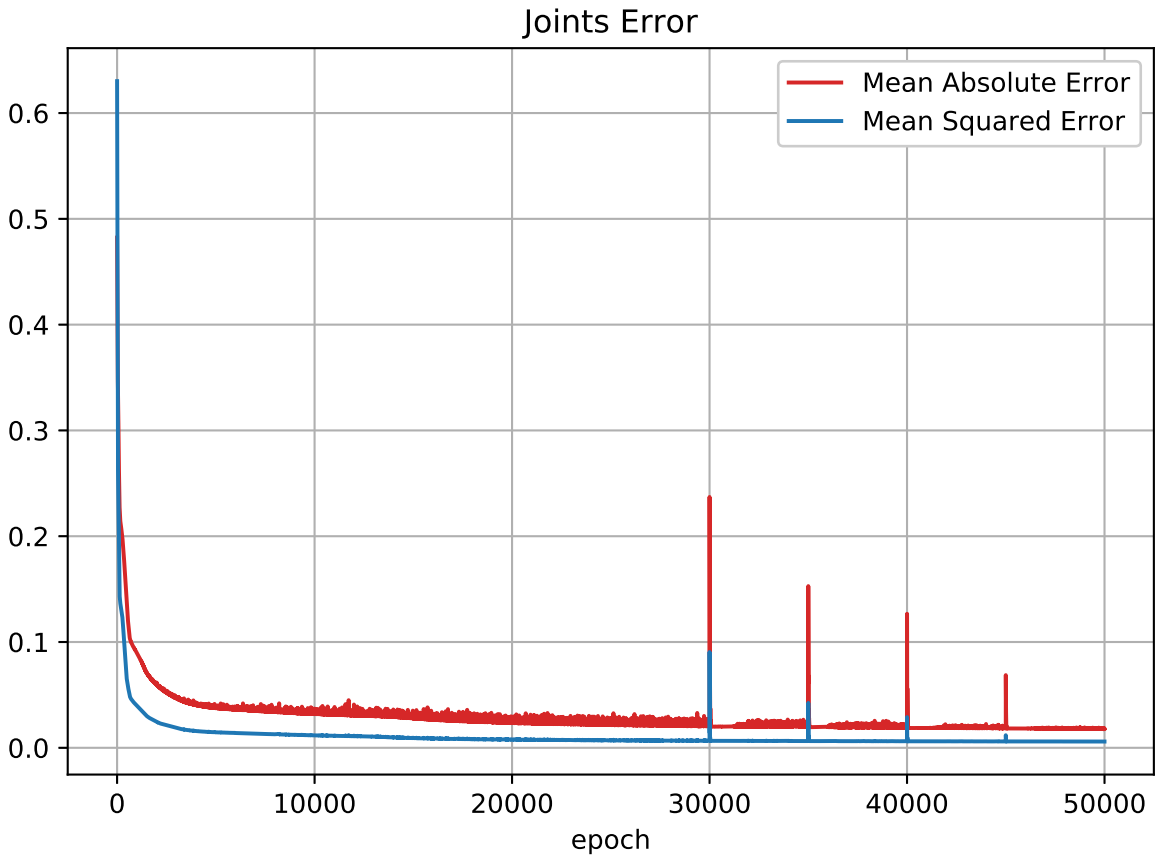


FIGURE 5.1 – The plots of mean squared error and mean absolute error during training.

The initial results come from the training procedure, outside the simulation environment. Figure 5.1 presents training curves for the kick keyframe dataset. In this case, the plots show mean squared error and mean absolute error metrics, respectively. In both metrics, the value decreases drastically in the first epochs. This same behavior was present in other training procedures as well. However, only after thousands of epochs the network achieved a low error that reproduced the motion successfully, which shows how sensible to small joint errors keyframes are, given that they are open-loop motions. The

peaks during the training happen at the learning rate transition instants, but they do not hurt the training procedure itself.

5.2 The Learned Kick Motion

The final mean absolute error is **0.018** radians and the motion is visually indistinguishable from the original one, as can be seen in Figure 5.2. In this figure, snapshots from both motions were taken. The Figure 5.3 shows several plots of joint angles comparing the original and learned kick motions. As we may see, the learned motion has fitted the movement with minor errors[1].

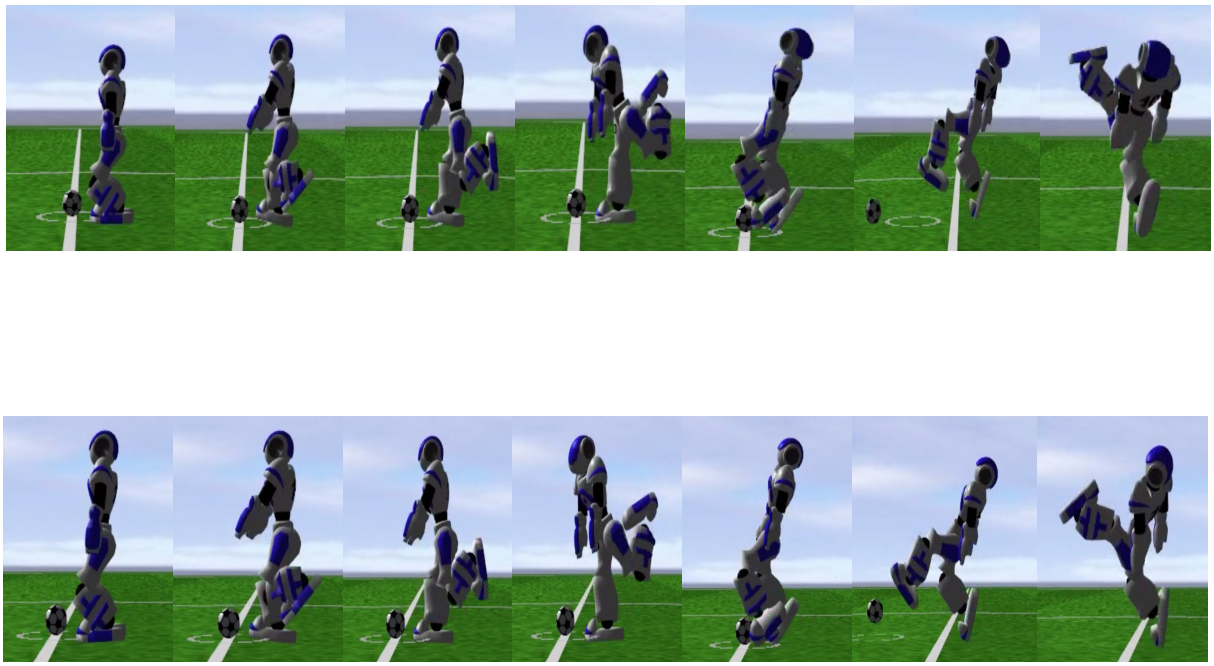


FIGURE 5.2 – Kick motion. The first row of figures shows the original kick motion. The second row shows the learned kick motion. The motions are visually indistinguishable.

In order to evaluate the learned kick motion in the RoboCup Soccer 3D domain, we created a statistical test. Inside the test scenario, the ball was placed initially in the center of the field with an agent near to it. The only action of the agent is to kick the ball in the goal direction. After the kick, the agent runs until reaching the ball and kicks it again, repeating this process till scoring a goal. When the goal occurs, this same scenario is repeated. The whole test was conducted during thirty minutes in clock time and the following data was collected: total number of kicks, number of successful kicks, mean distance that the ball has traveled and the standard deviation of this measure. The results from the original and learned kicks is shown in Table 5.1.

[1] Kick results video: https://youtu.be/UAbqQLUnvDo

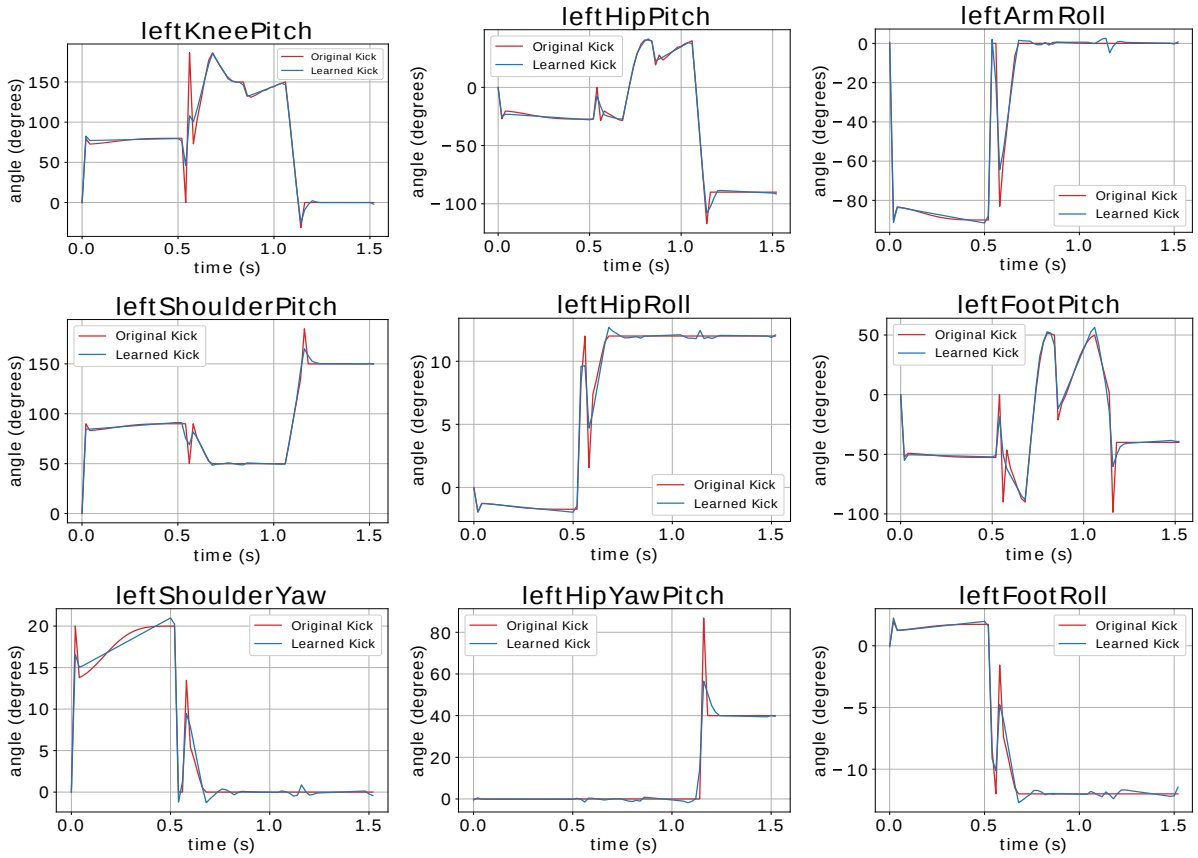


FIGURE 5.3 – Joint values for comparing original and learned kicks. The neural network was able to fit the joint trajectories with small errors.

TABLE 5.1 – Kick Comparison

Kick Type	**Statistics**		
	Accuracy (%)	**Distance** (m)	
		Mean	***Std***
Original Kick	64.5	8.92	3.82
Neural Kick	52.6	7.16	4.06

Although both kicks have similar results, the original kick is slightly better in this scenario. Confronting Figure 5.3, we can conclude that even with an almost equal representation, the kick lose part of its efficiency and this fact show us how sensible are movements based on keyframe data.

5.3 The Learned Walk Motion

Using the modified server described in Subsec. 4.1, a dataset with samples of the UT Austin Villa's walking motion (MACALPINE *et al.*, 2013) was acquired. This team is the current champion of RoboCup Soccer 3D competition (MACALPINE; STONE, 2018).

The objective is to mimic the walk motion as a keyframe and use that in our agent. The previously described framework used for learning our own kick motion was used in this training, including the neural network architecture and its hyperparameters.

The results from this training are shown in Figure 5.4. Similarly to Figure 5.3, it shows the joint angles throughout the walking motion period for the original and learned walk. Additionally, it shows the real joints values from the movement in the server. These joints were chosen because they are the most dynamic in the walk motion and therefore the hardest to learn.

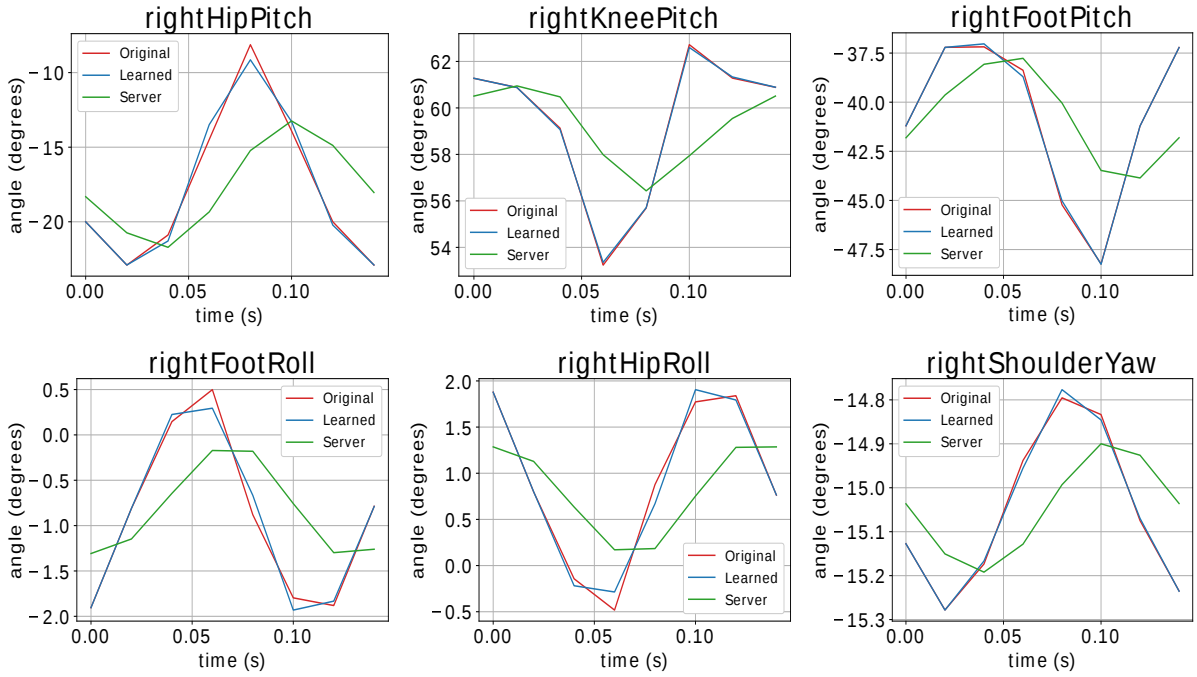


FIGURE 5.4 – Joints positions during a period of the walking motion for the original walk, the learned walk and the joints positions effectively attained during the learned walking motion.

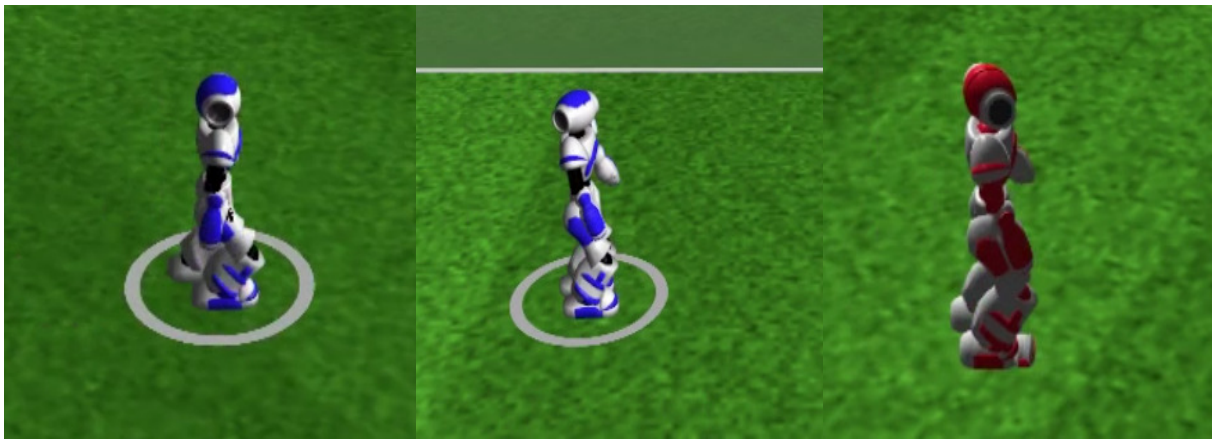
The learned motion has fitted the dataset very well. However, these values are just

desired joints. In fact, these values are used as references to joint controllers and are attenuated due to joint dynamics. Furthermore, this motion is operated in a open-loop fashion, so the agent is not able to correct its own trajectory, and this walks gets biased in the simple task of walking straight forward.

Despite the facts above, the motion works well in a non-competitive scenario[2], which is shown in the metrics collected in Forward Walk test scenario – agent walking forward from the goal post until the center line of field – in Table 5.2 and the visual representation in Figure 5.5.

TABLE 5.2 – Walk Comparison - Forward Walk

Walk Type	**Statistics**			
	Velocity (m/s)		**Y Error** (m)	
	Mean	***Std***	***Mean***	***Std***
Original Walk	0.87	0.01	-	-
Learned Walk	0.23	0.01	0.96	2.63



(a) (b) (c)

FIGURE 5.5 – Walking motions comparison. The figure (a) shows our agent in its regular walk. The figure (b) shows the same agent mimicking UT Austin Villa walk. The figure (c) is the UT Austin Villa agent itself, performing his own walking motion.

5.4 Other motions

This same framework was used to learn other keyframe motions originated by our agent itself, such as the get up motion. As the cases previously described, the resultant

[2] Walk results video: https://youtu.be/-pHxTrxllyY

neural network was capable of mimicking the keyframe including its interpolation. Hence, all of our keyframe motions can be replaced by neural motions with similar performance.

However, the huge improvement of this method is about mimicking other teams motions. In Soccer 3D environment, movements like kick and walking have giant impact in the team's performance. With this learning framework, our agent is able to mimic multiples movements from several teams.

As example, we collected data from UT Austin Villa kick, which was originally optimized using Deep Reinforcement Learning techniques (MACALPINE; STONE, 2017). Our agent learned this kick without any additional optimization strategy: we just used samples collected from the modified server.

6 Conclusions and Future Work

6.1 Preliminary Conclusions and Future Work

In this work, we presented a method for learning humanoid robot movements using datasets composed of joint values at each time instant. The learning framework provided was capable to learn several types of motion, including walk and kick, without any change in network architecture or hyperparameters. Moreover, the learned motions have similar performance to the original ones. Furthermore, this framework was able to learn other teams motions, without any knowledge about the underlying implementation – only using the joints values provided by a modified version of the server. This is a huge improvement in terms of getting improved motions, as our agent can mimic other teams motions using this machine learning technique.

As future work, we plan to apply Deep Reinforcement Learning algorithms to obtain faster and more robust kicks, using as "seed" the neural networks obtained in this work. Another track to be followed is to transfer the learning of this network to a new network that represents the motion policy itself (i.e a network which has as inputs the current state of the robot, including joint and link states, besides the current time instant), and optimize this motion policy in order to get a closed-loop walking and kick motion that can correct itself. As a long term goal, we intend to create model-free kick and walking engines.

6.2 Activities Plan

As next steps of this work, we plan to:

1. Import our supervised policy model into Deep Reinforcement Learning algorithms – Expect to finish until the end of **June, 2018**;

2. Iterate over objective function construction and optimization, in order to improve the kick motion – Expect to finish until mid **August, 2018**;

3. Execute the same test scenarios previously applied to compare results – Expect to finish until the end of **August, 2018**;

4. Test in 11x11 game to compare team performance – Expect to finish until the end of **August, 2018**; and

5. Complement the thesis with the new background, methodology, results and conclusions – Expect to finish until **November, 2018**.

Bibliography

AL, M. A. et. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.

BARTELS, R. H.; BEATTY, J. C.; BARSKY, B. A. **An Introduction to Splines for Use in Computer Graphics &Amp; Geometric Modeling**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. ISBN 0-934613-27-3.

CHOLLET, F. *et al.* **Keras**. 2015. Https://keras.io.

COLLINS, S.; RUINA, A.; TEDRAKE, R.; WISSE, M. Efficient Bipedal Robots Based on Passive Dynamic Walkers. **Science Magazine**, v. 307, p. 1082–1085, February 2005.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. `http://www.deeplearningbook.org`: MIT Press, 2016.

GOUAILLIER, D.; HUGEL, V.; BLAZEVIC, P.; KILNER, C.; MONCEAUX, J.; LAFOURCADE, P.; MARNIER, B.; SERRE, J.; MAISONNIER, B. Mechatronic design of nao humanoid. In: **2009 IEEE International Conference on Robotics and Automation**. Kobe, Japan: IEEE, 2009. p. 769–774. ISSN 1050-4729.

HEESS, N.; TB, D.; SRIRAM, S.; LEMMON, J.; MEREL, J.; WAYNE, G.; TASSA, Y.; EREZ, T.; WANG, Z.; ESLAMI, S. M. A.; RIEDMILLER, M. A.; SILVER, D. Emergence of locomotion behaviours in rich environments. **CoRR**, abs/1707.02286, 2017. Disponível em: <http://arxiv.org/abs/1707.02286>.

KAJITA, S.; KANEHIRO, F.; KANEKO, K.; YOKOI, K.; HIRUKAWA, H. The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation. In: **In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems**. Hawaii, USA: IEEE, 2001.

KINGMA, D. P.; BA, J. Computer science > learning adam: A method for stochastic optimization. **arXiv**, Dec 2014.

KITANO, H.; ASADA, M.; KUNIYOSHI, Y.; NODA, I.; OSAWAI, E.; MATSUBARA, H. Robocup: A challenge problem for ai and robotics. In: KITANO, H. (Ed.). **RoboCup-97: Robot Soccer World Cup I**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 1–19. ISBN 978-3-540-69789-3.

LU, C.; TANG, X. Surpassing human-level face verification performance on LFW with gaussianface. **CoRR**, abs/1404.3840, 2014. Disponível em: <http://arxiv.org/abs/1404.3840>.

MACALPINE, P.; BARRETT, S.; URIELI, D.; VU, V.; STONE, P. Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition. In: **Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)**. Toronto, Ontario, Canada: AAAI, 2012.

MACALPINE, P.; COLLINS, N.; LOPEZ-MOBILIA, A.; STONE, P. Ut austin villa: Robocup 2012 3d simulation league champion. In: CHEN, X.; STONE, P.; SUCAR, L. E.; ZANT, T. van der (Ed.). **RoboCup 2012: Robot Soccer World Cup XVI**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 77–88. ISBN 978-3-642-39250-4.

MACALPINE, P.; STONE, P. Prioritized role assignment for marking. In: BEHNKE, S.; LEE, D. D.; SARIEL, S.; SHEH, R. (Ed.). **RoboCup 2016: Robot Soccer World Cup XX**. Berlin: Springer Verlag, 2016, (Lecture Notes in Artificial Intelligence). p. 306–18.

MACALPINE, P.; STONE, P. Ut austin villa 3d simulation soccer team 2017. RoboCup Symposium, 2017.

MACALPINE, P.; STONE, P. UT Austin Villa: RoboCup 2017 3D simulation league competition and technical challenges champions. In: SAMMUT, C.; OBST, O.; TONIDANDEL, F.; AKYAMA, H. (Ed.). **RoboCup 2017: Robot Soccer World Cup XXI**. Japan: Springer, 2018, (Lecture Notes in Artificial Intelligence).

MACALPINE, P.; URIELI, D.; BARRETT, S.; KALYANAKRISHNAN, S.; BARRERA, F.; LOPEZ-MOBILIA, A.; STIURCA, N.; VU, V.; STONE, P. **UT Austin Villa 2011 3D Simulation Team Report**. Austin, Texas, December 2011.

MUNIZ, F.; MAXIMO, M. R. O. A.; RIBEIRO, C. H. C. Keyframe movement optimization for simulated humanoid robot using a parallel optimization framework. In: **2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)**. Recife, Brazil: IEEE, 2016. p. 79–84.

OBST, O.; ROLLMAN, M. Spark – a generic simulator for physical multiagent simulation. **Computer Systems Science and Engineering**, v. 5, p. 347–356, 2005.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 533, 1988.

SHON, A. P.; GROCHOW, K.; RAO, R. P. N. Robotic imitation from human motion capture using gaussian processes. In: **In Proceedings of the IEEE/RAS International Conference on Humanoid Robots (Humanoids**. Tsukuba, Japan: [s.n.], 2005.

SILVER, D.; HUBERT, T.; SCHRITTWIESER, J.; ANTONOGLOU, I.; LAI, M.; GUEZ, A.; LANCTOT, M.; SIFRE, L.; KUMARAN, D.; GRAEPEL, T.; LILLICRAP, T. P.; SIMONYAN, K.; HASSABIS, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. **CoRR**, abs/1712.01815, 2017. Disponível em: <http://arxiv.org/abs/1712.01815>.

TANIKIC, D.; DESPOTOVIC, V. Artificial intelligence techniques for modelling of temperature in the metal cutting process. IntechOpen, 2012.

XIONG, W.; DROPPO, J.; HUANG, X.; SEIDE, F.; SELTZER, M.; STOLCKE, A.; YU, D.; ZWEIG, G. Achieving human parity in conversational speech recognition. **CoRR**, abs/1610.05256, 2016. Disponível em: <http://arxiv.org/abs/1610.05256>.

XU, B.; WANG, N.; CHEN, T.; LI, M. Empirical evaluation of rectified activations in convolutional network. **arXiv**, Nov 2015.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO N	4. N DE PÁGINAS
TC	June 12th, 2018	DCTA/ITA/DM-018/2015	30

5. TÍTULO E SUBTÍTULO:

Deep Reinforcement Learning for Humanoid Kick Motion

6. AUTOR(ES):

Luckeciano Carvalho Melo

7. INSTITUIÇÃO(ÕES)/ÓRGAO(S) INTERNO(S)/DIVISÃO(ÕES):

Aeronautics Institute of Technology – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Deep Reinforcement Learning; Robotics; Artificial Intelligence

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Deep Reinforcement Learning; Robotics; Artificial Intelligence

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**

ITA, São José dos Campos, 2018. Trabalho de Graduação.

11. RESUMO:

Controlling a high degrees of freedom humanoid robot is acknowledged as one of the hardest problems in Robotics. Due to the lack of mathematical models, an approach frequently employed is to rely on human intuition to design keyframe movements by hand, usually aided by graphical tools. In this paper, we propose a learning framework based on neural networks in order to mimic humanoid robot movements. The developed technique does not make any assumption about the underlying implementation of the movement, therefore both keyframe and model-based motions may be learned. The framework was applied in the RoboCup 3D Soccer Simulation domain and promising results were obtained using the same network architecture for several motions, even when copying motions from another teams.

12. GRAU DE SIGILO:

(**X**) **OSTENSIVO** ( ) **RESERVADO** ( ) **SECRETO**