Final Year Project Interim Report

Inverse Dynamic Model Identification of a Bipedal Robot Using Recurrent Neural Networks

Temirlan Nurgazy, Ramazan Bolatbek

Supervisor: Michele Folgheraiter

Moderator: Timur Umurzakov

Department of Robotics and Mechatronics
School of Science and Technology
Nazarbayev University

May 3, 2024

Table of Contents

Abstract

The principal aim is to contribute to the identification of the inverse dynamics of the bipedal robot by training several Artificial Neural Network (ANN) models. These include Feedforward Neural Networks (FFNN), Long Short-Term Memory(LSTM), and Recurrent Neural Networks (RNN). The project will compare these models to determine the best performer in terms of learning the inverse dynamics of the bipedal robot, considering both performance and computational complexity. The main tools for the achievement of the task are CoppeliaSim simulation and Python programming language

Acknowledgments

Thanks to Professor Michele Folgheraiter for his mentorship and supervision, which greatly contributed to the project's success. Also, acknowledgments are due to Timur Umurzakov for his invaluable guidance and support during the project.

Chapter 1: **Introduction**

A bipedal robot is a robot that walks on two legs, similar to humans. A Good example of these robots' implementation is to handle tasks that involve replacing humans in places where it is dangerous for people to work. This includes walking over rough ground or going upstairs. While performing such tasks reliably, they need to walk properly without falling. Robots incorporate special systems to control their movements and maintain balance. However, controlling a robot that moves smoothly and handles unexpected obstacles is tough. Inverse Dynamics is a popular way to control bipedal robots. However, it involves advanced calculations of movements which is time-consuming and this method can not calculate all factors affecting the robot. Therefore, we're considering using Neural Networks to improve the bipedal robot's working process. Neural networks are a smart technology that helps solve problems in various areas, including robotics. They learn from examples and can help robots figure out how to move more naturally. Our main goal is to use neural networks to improve how these robots control the force in joints. This approach is torque control. By doing this, we hope to make bipedal robots move more accurately, predicting and adjusting how their bodies need to move. This could make these robots more useful for different tasks. Adding to the idea of using neural networks for improving bipedal robots, it's also crucial to understand that not all neural networks are the same. They can be very different in how accurate and efficient they are. This means that some neural networks might be good at helping robots move smoothly, while others might not be as helpful. Because of these differences, it's important to compare various neural networks to see which works best for predicting inverse dynamics prediction thus improving the movements of bipedal robots. By testing and comparing different neural networks, we can find the one that makes robots move most accurately. This step is crucial for ensuring we implement the best technology available to make our robots' movements as efficient as possible.

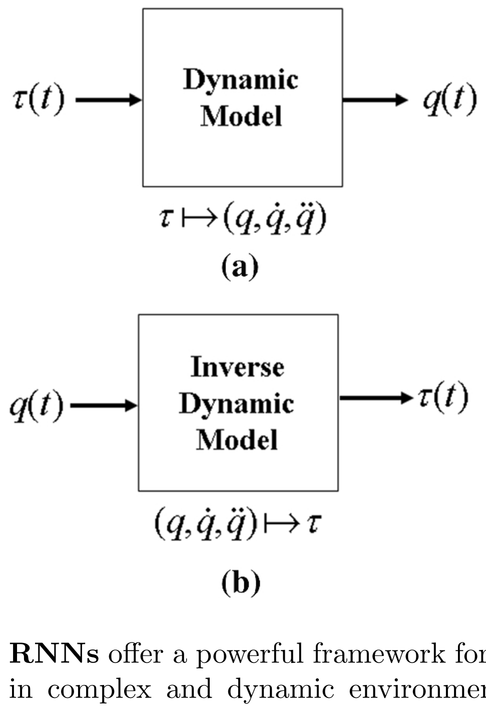
Chapter 2: **Literature review**

The research focuses on the real-time neural network control of a biped walking robot, specifically employing Cerebellar Model Articulation Controllers (CMAC) neural networks to address the challenges of dynamic balance in bipedal locomotion. This project sets out to refine biped control strategies without relying on detailed kinematic or dynamic models, instead utilizing a hierarchy of simple gait oscillators, PID controllers, and neural network learning[1]. The document highlights the progress in teaching the experimental biped robot to achieve side-to-side weight shifting, quasi-static balance, and dynamic balance necessary for walking. The biped has learned to march in place and take short steps through real-time control experiments, marking a significant stride towards efficient walking.

Bipedal robots are impressive machines with diverse applications, but achieving stable movement remains challenging. Recent advancements in recurrent neural networks (RNNs), offer promising solutions. This literature review explores how RNNs, particularly Long-short term memory (LSTM), can enhance the inverse dynamic model identification of bipedal robots. We aim to assess the effectiveness of RNN-based approaches, identify inverse dynamics principles, and research gaps, and suggest future directions for improving bipedal robot control.

Inverse dynamics for bipedal robots is a fundamental concept in robotics that enables precise control of movement by determining the forces and torques needed at each joint to achieve desired motions. It forms the basis for developing control strategies that ensure stability, agility, and efficiency in bipedal robot locomotion.

Inverse dynamics involves solving the dynamic equations in reverse. Instead of predicting the motion of the robot given the forces and torques, as in forward dynamics (figure 1a), inverse dynamics (figure 1b) aims to determine the forces and torques required to produce a desired motion. This is achieved by rearranging the dynamic equations to solve for the forces and torques.


$\tau(t)$
Dynamic Model
$q(t)$
$\tau \mapsto (q,\dot{q},\ddot{q})$
(a)
$q(t)$
Inverse Dynamic Model
$\tau(t)$
$(q,\dot{q},\ddot{q}) \mapsto \tau$
(b)

RNNs offer a powerful framework for learning and predicting the behavior of bipedal robots in complex and dynamic environments. By leveraging sequential data from sensors and incorporating recurrent connections, RNN-based approaches enable robots to perform tasks more autonomously, adaptively, and robustly. Wu Yolei's[2] article introduces a robust control strategy for bipedal robots using Recurrent Neural Networks (RNNs). The approach aims to enhance the robot's stability and adaptability by leveraging the capabilities of RNNs to learn and predict dynamic behaviors in real-time. By employing robust RNN-based controllers, the article contributes to advancing the field of bipedal robotics, offering potential solutions for improved locomotion control in challenging environments.

LSTMs excel at capturing long-term dependencies in sequential data by effectively managing information flow through their memory cells and gates. The paper of Yu Wang [3], introduces a novel approach to dynamic system identification using Long Short-Term Memory (LSTM) neural networks, a type of recurrent neural network (RNN). LSTM networks are known for their ability to model sequential data effectively. This concept offers a promising method for accurately capturing and predicting the behavior of dynamic systems[4]. By leveraging LSTM networks, the paper aims to enhance the accuracy and efficiency of dynamic system identification, potentially leading to advancements in various fields such as control systems, robotics, and time-series prediction.

LSTMs have memory cells that can store information over time. These cells are equipped with three gates: the input gate, the forget gate, and the output gate[5]. First, the "input gate" regulates the flow of new information into the memory cell. It decides which values from the input and the previous cell state to update. Next comes the "forget gate" controls what information should be discarded from the memory cell. It decides which parts of the previous cell state are irrelevant to the current task. Last is the output gate determines which parts of the memory cell state should be outputted as the final prediction. It filters the information stored in the memory cell before passing it to the next time step or output layer. Combining those gates we get that, at each time step, the LSTM network computes the new cell state by combining the information from the input, the previous cell state, and the output from the gates. This allows LSTMs to selectively remember or forget information over time.

A **Feedforward Neural Network** (FFNN), also known as a Multilayer Perceptron (MLP), is a type of artificial neural network in which connections between nodes do not form cycles. Feedforward Neural Networks are not typically the first choice for controlling bipedal robots due to several limitations such as lack of temporal information, limited adaptability, difficulty in handling High-Dimensional data and limited feedback mechanisms. Instead, recurrent neural networks or other architectures with recurrent connections, such as Long Short-Term Memory networks are more suitable for controlling bipedal robots. This model can capture temporal dependencies, handle sequential data, and incorporate feedback mechanisms, making them better equipped to deal with the dynamic and time-varying nature of bipedal locomotion.

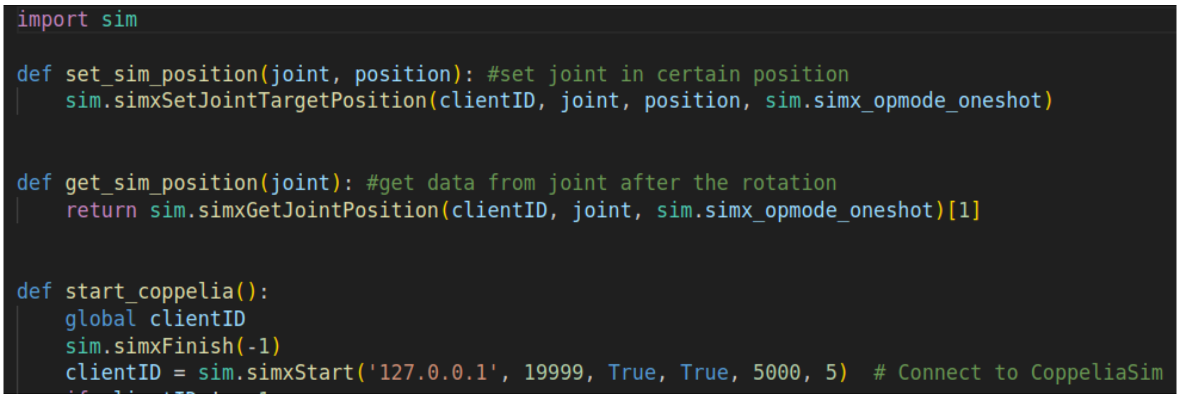
This research aligns with the broader goal of achieving more natural, human-like locomotion in robots, fitting neatly into the context of advancing robotic autonomy and adaptability. By focusing on dynamic balance and employing neural networks for real-time learning, this project contributes significantly to the field. It not only addresses the inherent limitations of static balance models but also showcases the potential of neural networks to revolutionize how robots learn and execute complex tasks like walking. This approach offers a promising pathway to robots that can navigate real-world environments with the grace and efficiency akin to biological organisms, marking a critical step forward in robotics research.

Chapter 3: **Methodology**

In the inverse dynamics system calculate the torque for the motor using Position, velocity, and acceleration. The goal of Neural Networks is to replace the calculation of torque. Thus, to use Neural networks for the prediction of inverse dynamics, we needed to collect a dataset.

To gather the necessary data, such as position, velocity, and more, we based on a position control system that dictates the robot's movements. This system precisely directs the robot's motion—its destination, speed, and rotation, then the algorithm calculates the force required for each action[6]. Both in a computer simulation and the physical robot, data collection this system was a role model.

In our data collection efforts, CoppeliaSim emerged as a crucial tool. This simulation software enabled us to replicate the robot's actions and gather valuable data. By integrating a Python library called 'sim,' we established communication with CoppeliaSim, allowing us to command the robot's movements and record pertinent information, such as position and velocity.



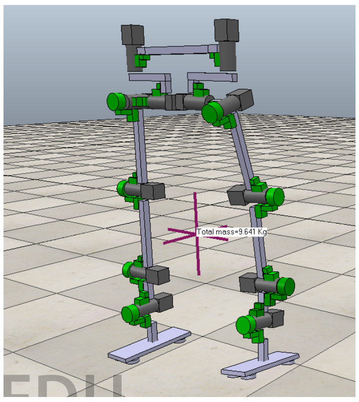
```python
import sim

def set_sim_position(joint, position): #set joint in certain position
    sim.simxSetJointTargetPosition(clientID, joint, position, sim.simx_opmode_oneshot)


def get_sim_position(joint): #get data from joint after the rotation
    return sim.simxGetJointPosition(clientID, joint, sim.simx_opmode_oneshot)[1]


def start_coppelia():
    global clientID
    sim.simxFinish(-1)
    clientID = sim.simxStart('127.0.0.1', 19999, True, True, 5000, 5)  # Connect to CoppeliaSim
```

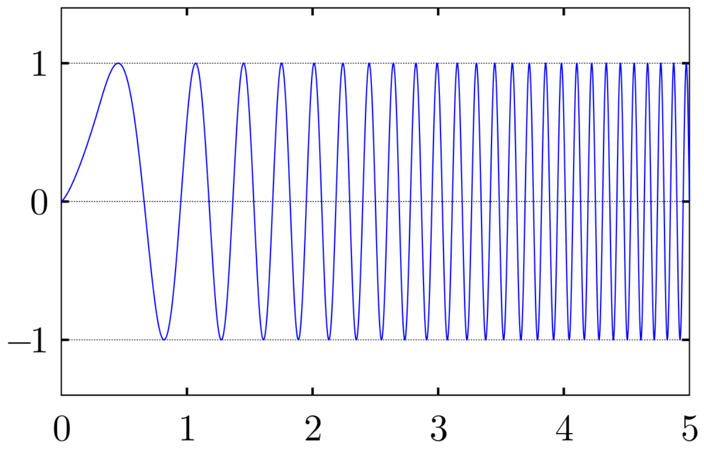
For the best result of NN, we took the approach of simultaneously moving all joints with different frequencies for each joint. Thus, our dataset will cover the most data for the training of NN.



Apart from CoppeliaSim, we also had a bipedal robot prototype in the laboratory. Using the support of a supervisor and script, by providing rotation angles for joints we also could collect data from prototype

As we mentioned, we had two ways of dataset collection and testing of NN. These are the CoppeliaSim model and physical prototype. We decided to focus on the left leg of bipedal robots. Thus, we will be able to do the research faster as if we did for both legs we would have to always change the code second leg and consider one leg hitting another. The left leg has 6 joints. Therefore, we will gather data and train NN for the supply of torque for these 6 joints

For dataset collection to determine the angles for controlling the robot's joint movements within the simulation, we employed a method known as the chirp function. This function provided smooth transitions in the robot's motion, starting with gradual movements and gradually accelerating. By implementing this technique, we obtained comprehensive data on the robot's behavior across various speeds.



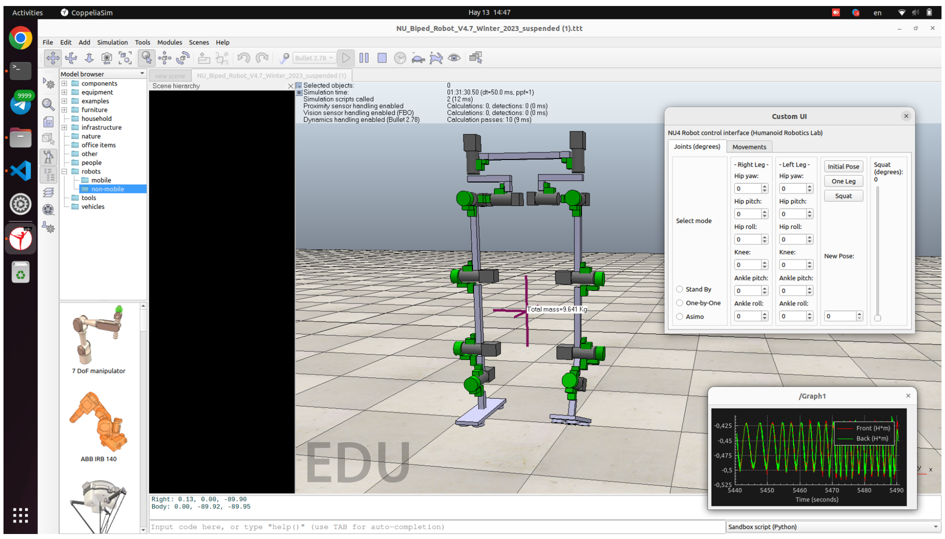
After dataset collection, for the NN training, using the TensorFlow library in Python we wrote a script from scratch for training different Neural Networks. These include RNN, LSTM, and FFNN. We trained these NNs with the same dataset and discovered the best among them. Apart from NNs, the right settings can increase the accuracy of NNs. These are dropouts, the number of layers, and activation functions. We will try different combinations with these for maximization of accuracy. As an identification of good training, MSE is a key variable that represents the accuracy of the Neural Network.

Chapter 4: **Achieved Results**

Our project began with a comprehensive literature review, setting the foundation for our subsequent progress. We started by focusing on research and case studies related to neural networks, bipedal robotics, and the specific challenges associated with simulating and controlling such systems. This initial phase was crucial for understanding the current state of the art, identifying gaps in existing methodologies, and pinpointing potential areas where our project could contribute to advancing the field. Through academic journals, online repositories, and industry publications, we gathered insights into the latest developments, tools, and best practices in machine learning and robotics simulation.

Following the literature review, we moved on to learning with TensorFlow, a leading framework for developing and training neural networks. Our engagement with TensorFlow began with practical exercises, including training models for stock predictions and text generation, guided by tutorials and resources found online. This hands-on experience was instrumental in familiarizing ourselves with the nuances of neural network architecture, optimization, and deployment within the context of our project goals.

We also explored CoppeliaSim, a versatile simulation platform that became integral to our project for creating realistic virtual environments for our bipedal robot.



```
Selected objects:                          0
Simulation time:                           01:31:30.50 (dt=50.0 ms, ppf=1)
Simulation scripts called                  2 (12 ms)
Proximity sensor handling enabled          Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)       Calculations: 0, detections: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78)    Calculation passes: 10 (9 ms)
```

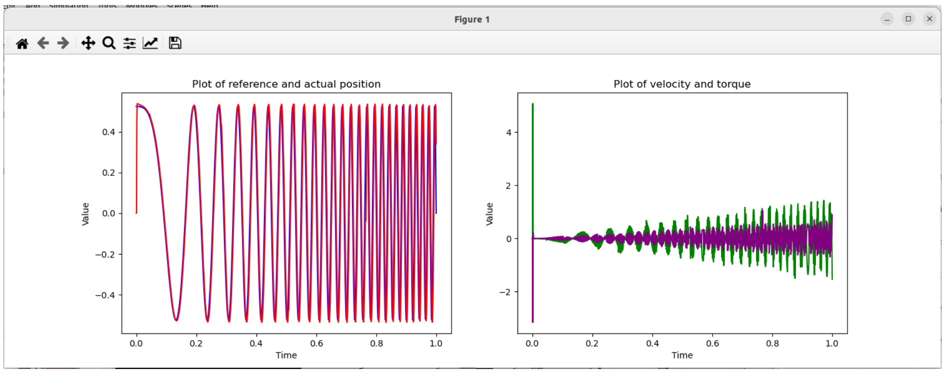
NU4 Robot control interface (Humanoid Robotics Lab)

- Right Leg -		- Left Leg -	
Hip yaw:	0	Hip yaw:	0
Hip pitch:	0	Hip pitch:	0
Hip roll:	0	Hip roll:	0
Knee:	0	Knee:	0
Ankle pitch:	0	Ankle pitch:	0
Ankle roll:	0	Ankle roll:	0

New Pose: 0

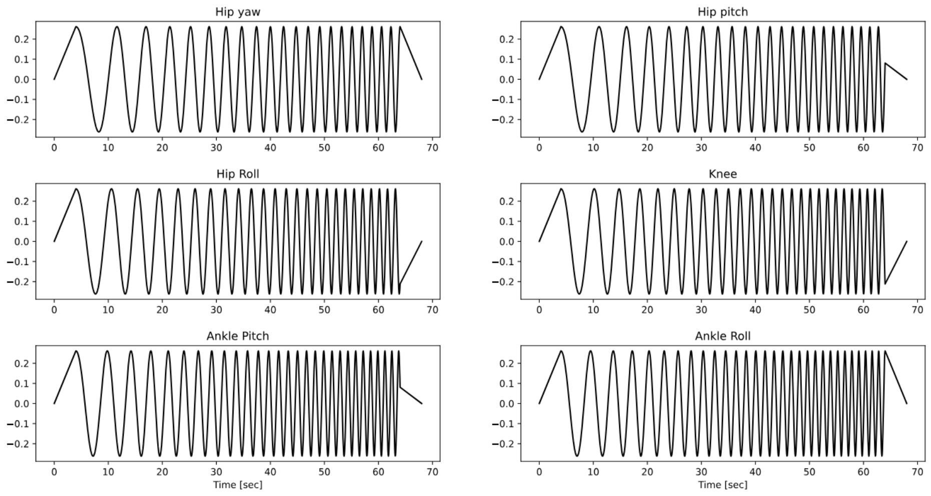
```
Right: 0.13, 0.00, -89.90
Body: 0.00, -89.92, -89.95
Input code here, or type "help()" (use TAB for auto-completion)
```

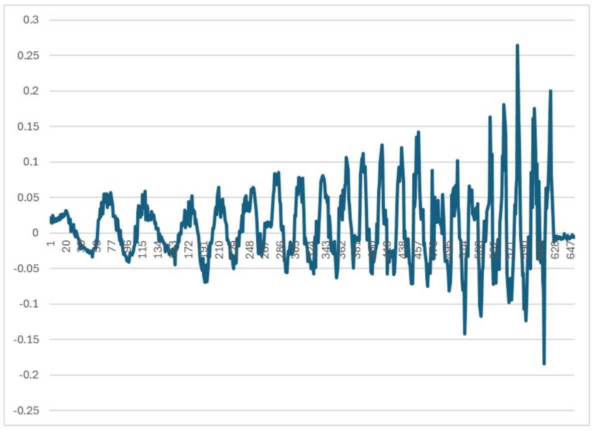
Initially, we collected data from one joint at a time which resulted in the disappointing accuracy of neural networks. Therefore, we advanced our approach by utilizing the chirp function to move all joints simultaneously. This new method significantly enhanced the complexity and richness of our dataset by simulating more dynamic and interconnected movements of the robot. Here you can see the result of one first joint in simulation:



On the left, there are reference positions and actual positions. On the right, there are the velocity (green line) and torque (purple line) of the joint. The fact that velocity and torque are increasing over time makes sense as that is what the chirp function is for. It allows us to accelerate the rotation through time

Here you can see reference positions through the time graphs created using chirp functions and the obtained torque graph of one joint of the prototype robot:

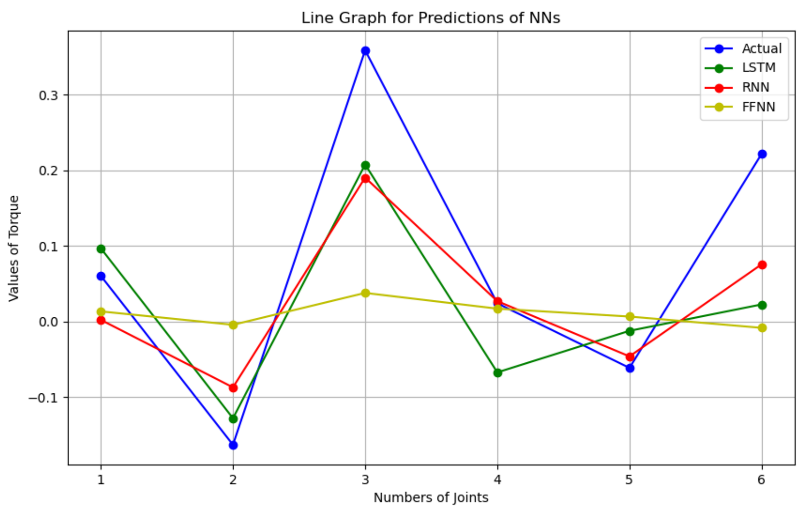




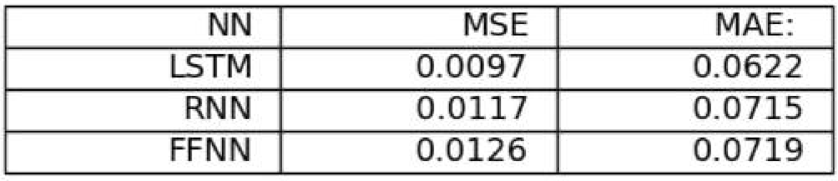
In comparison with the torque graph obtained in the simulation, the prototype torque graph seems less accurate. However, real-life data is expected to be not smooth as unlike simulation there are a lot more affecting factors.

For the training of NNs, in the early stages, we used a traditional regression neural network to predict and control the robot's movements, but the results were underwhelming with a Mean Square Error (MSE) of about 30.5 which is unacceptable for NN. This prompted a pivotal shift to employ other NNs such as Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNN), and Feed-Forward Neural Networks. These NNs proved to be more accurate as we achieved an MSE of nearly 0.01. Our comprehensive dataset, encompassing reference positions, actual positions, velocities, and torques of the robot's joints, laid the groundwork for this phase of the project.

For the training of the neural network, we saved data collected from the simulation and prototype in a CSV file. The prototype issue was that in many cases two rows of data were identical. These could be due to the low speed of motors or lags in encoders in robots Therefore, we had to delete repeating rows which shortened our dataset. Then, we had to set a model for 18 inputs and 6 outputs where 18 inputs are the reference position of 6 joints, the reference velocity of 6 joints, and the reference acceleration of 6 joints. Though we were not provided with acceleration from the simulation or prototype, we wrote a special script for the calculation of acceleration using velocity and time intervals. We divided data with a ratio of 80 to 20 where 80 percent is training data and 20 percent is for testing. For training we chose 100 epochs and 10 batch sizes as these were recommended for regression tasks As a result of training, here is the example prediction of torque using the best model LSTM:

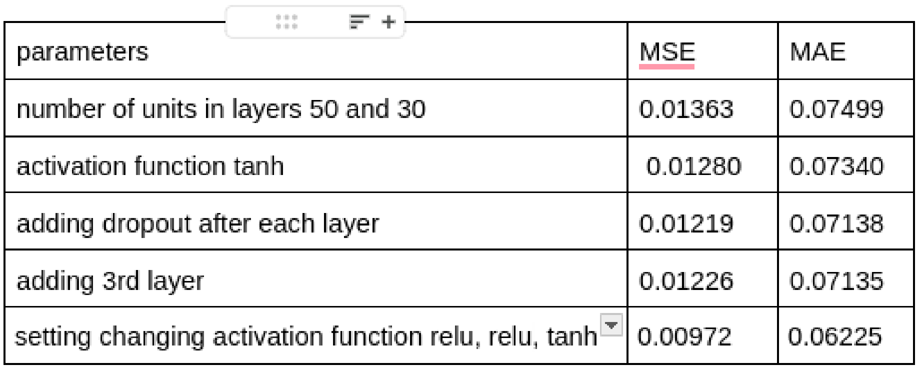


As we can see in some cases predictions are almost similar to actual data and in some cases, there is a noticeable difference. However, all NNs follow a trend of actual output, and in most cases, the predicted output is similar to the actual. Following is a comparison of the accuracy of NNs:



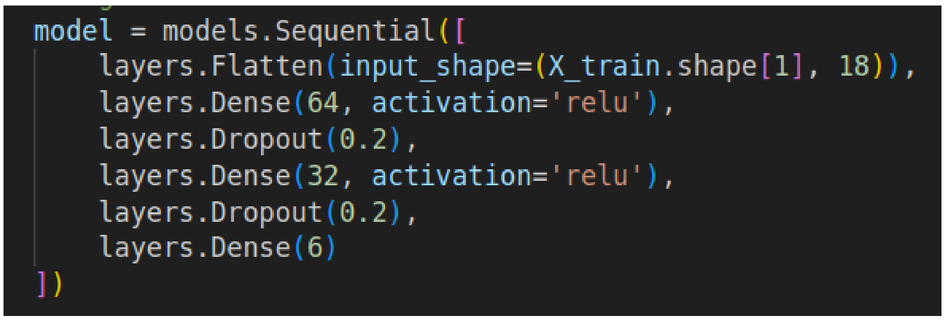
NN	MSE	MAE:
LSTM	0.0097	0.0622
RNN	0.0117	0.0715
FFNN	0.0126	0.0719

According to this data, we can see that LSTM is the best solution in terms of accuracy. However, to maximize the accuracy of LSTM we picked the best settings for its training. For this we did testing with MSE and MAE of different settings. We tried many different combinations but here are the parameters that improved the results:



parameters	MSE	MAE
number of units in layers 50 and 30	0.01363	0.07499
activation function tanh	0.01280	0.07340
adding dropout after each layer	0.01219	0.07138
adding 3rd layer	0.01226	0.07135
setting changing activation function relu, relu, tanh	0.00972	0.06225

Using these parameters we defined a neural network model using the Keras Sequential API. Here you can see the code of this part:



```
model = models.Sequential([
    layers.Flatten(input_shape=(X_train.shape[1], 18)),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(32, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(6)
])
```

Thus, we discovered that with an MSE of 0.0092, LSTM is the best neural network for the prediction of inverse dynamics for bipedal robots. The insights and methodologies we've developed offer promising directions for future research and applications in bipedal robotic systems.

Chapter 5: **Challenges and Solutions**

At the beginning of our project, we ran into a few tough spots. The first big challenge came when we were looking at different research papers for information. We found that even though these papers were trying to achieve the same thing, they went about it in very different ways, which was confusing. Our supervisor helped us see that there are several methods to control a motor, like controlling its position, speed, or torque. This helped us pick the right papers to look at and understand better what we were reading.

Then, we had a tricky time trying to get our Python script to work with the CoppeliaSim simulation for our bipedal robot. We had to set up a special connection using a certain IP address, organize specific files in one folder, and install a program that worked with our computer's operating system. I was using Ubuntu 22.04, but the latest version of the program was for Ubuntu 20.04. Thankfully, we found some help on StackOverflow and got the connection working.

Coding was another big part of the project where we spent a lot of time. We ran into many bugs or errors in our code. Luckily, we knew a bit about coding in Python, so even though we had a lot of problems, we could fix them pretty quickly.

Furthermore, using the chirp function was hard because there wasn't much information out there on how to use it. We had to try changing its settings one by one and see what happened. In the end, we figured it out.

Upon new data provided earlier, we also faced the challenge of altering our neural network training script from a simple regression model to more complicated NNs like LSTM. This transition was necessary to handle the dynamics of our more complex dataset, but it required a shift from a 2-dimensional to a 3-dimensional data normalization format. Adjusting all subsequent parts of our script to accommodate this change proved to be particularly difficult. It was a significant technical hurdle, demanding a thorough reworking of our existing codebase to effectively utilize LSTM networks[7].

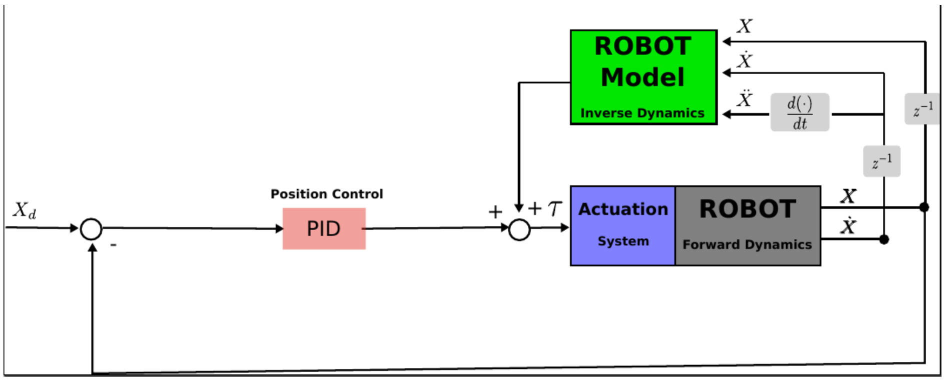
Even though we faced these challenges, they taught us a lot. We learned more about how to solve problems and got better at understanding and working on our project.

Chapter 6: **Future work**

While collecting the dataset we limited the joints' angle of rotation to 15 degrees [6]. This was done to ensure the prototype's safety as if the maximum angle was set to a big number there would be the possibility of collision of legs with each other. Also, the prototype is lifted on a special construction. When the joint rotates fast at a big angle. There is a risk that the balance will be shattered. Therefore, the dataset could be collected with a bigger rotation angle at joints

Looking ahead, our next step is to test our neural networks on a real motor and an actual bipedal robot. This will be a crucial step in transitioning from theoretical models and simulations to practical applications. Testing on real-world hardware will not only validate our models but also provide invaluable insights into their performance and potential improvements.

Unfortunately, at this time robot works based on position control. Our torque control approach is not compatible with the current algorithm on the prototype bipedal robot. Therefore, our system requires writing a new script based on torque control. This approach would need a specific control system. The following system has a good potential for this:



While our current model for torque prediction in the bipedal robot has shown promising results, it is essential to acknowledge its limitations. One significant constraint is that the model assumes the robot to be in a lifted position during operation. This assumption may not always hold in real-world scenarios where the robot may encounter various terrain conditions or unexpected disturbances.

To address this limitation and enhance the adaptability of our model, future work should focus on implementing an online learning mechanism. Specifically, we propose the integration of a Reservoir-based Recurrent Neural Network (RNN) combined with the Recursive Least Squares (RLS) algorithm. This approach enables real-time learning and adaptation to changing dynamics, allowing the robot to continuously improve its torque prediction accuracy as it operates.

By incorporating such an online learning mechanism into our torque prediction model, we anticipate significant improvements in its robustness and performance across various operational scenarios. Additionally, this advancement will pave the way for the development of more adaptive and resilient bipedal robots capable of navigating complex environments with greater efficiency and stability.

In summary, our future work involves a strategic focus on training, testing, and refining neural networks, with an emphasis on practical application and evaluation. Through this process, we hope to conclude our project with a clear understanding of the best neural network model for controlling bipedal robots, contributing valuable knowledge to the field and setting the stage for further advancements.

Chapter 7: **Conclusion**

In conclusion, our journey through this project has been both challenging and enriching, offering us invaluable insights into the complexities of neural network applications in robotics. Starting with a comprehensive literature review, we navigated through diverse methodologies and conflicting data to build a solid foundation for our work. The hands-on experience we gained with TensorFlow and CoppeliaSim not only enhanced our technical skills but also prepared us for the intricate process of data collection and simulation.

The challenges we encountered, from establishing a connection between Python scripts and CoppeliaSim to implementing the chirp function, tested our problem-solving abilities and pushed us to explore creative solutions. Our progress in coding and debugging further exemplified the iterative nature of research and development, highlighting the importance of persistence and technical acumen.

The prospect of testing our models on actual hardware brings us closer to bridging the gap between theoretical research and practical application, a crucial step for validating our findings and contributing to the field of robotics.

Our project stands as a testament to the interdisciplinary nature of robotics and machine learning, illustrating the synergy between software simulation and hardware implementation. As we move forward, we remain dedicated to refining our models, enhancing our dataset, and ultimately contributing to the ongoing evolution of bipedal robotic control. Through our efforts, we aspire to not only achieve our project goals but also to inspire further research and innovation in this dynamic and impactful field.

Bibliography

[1]W. T. Miller, ”Real-time neural network control of a biped walking robot,” in IEEE Control Systems Magazine, vol. 14, no. 1, pp. 41-48, Feb. 1994, doi: 10.1109/37.257893.

[2]Wu, Y., Song, Q. & Yang, X. Robust Recurrent Neural Network Control of Biped Robot. *J Intell Robot Syst* **49**, 151–169 (2007). https://doi.org/10.1007/s10846-007-9133-1

[3] Yu Wang, ”A new concept using LSTM Neural Networks for dynamic system identification,” 2017 American Control Conference (ACC), Seattle, WA, USA, 2017, pp. 5324-5329, doi: 10.23919/ACC.2017.7963782.

[4]M. Folgheraiter, A. Yskak, and S. Yessirkepov, “One-shot bipedal robot dynamics identification with a reservoir-based RNN,” *IEEE Access*, vol. 11, pp. 50180–50194, 2023. doi:10.1109/access.2023.3277977

[5]F. A. Gers, J. Schmidhuber and F. Cummins, ”Learning to forget: continual prediction with LSTM,” 1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470), Edinburgh, UK, 1999, pp. 850-855 vol.2, doi: 10.1049/cp:19991218.

[6]] T. Umurzakov, S. Yessirkepov and M. Folgheraiter, ”Fast Prototyping and Testing of a New Full Scale Bipedal Robot,” 2022 7th International Conference on Robotics and Automation Engineering (ICRAE), Singapore, 2022, pp. 215-221, doi: 10.1109/ICRAE56463.2022.10056190.

[7]S. K. Challa, A. Kumar, V. B. Semwal and N. Dua, ”An Optimized-LSTM and RGB-D Sensor-Based Human Gait Trajectory Generator for Bipedal Robot Walking,” in IEEE Sensors Journal, vol. 22, no. 24, pp. 24352-24363, 15 Dec.15, 2022, doi: 10.1109/JSEN.2022.3222412.

,