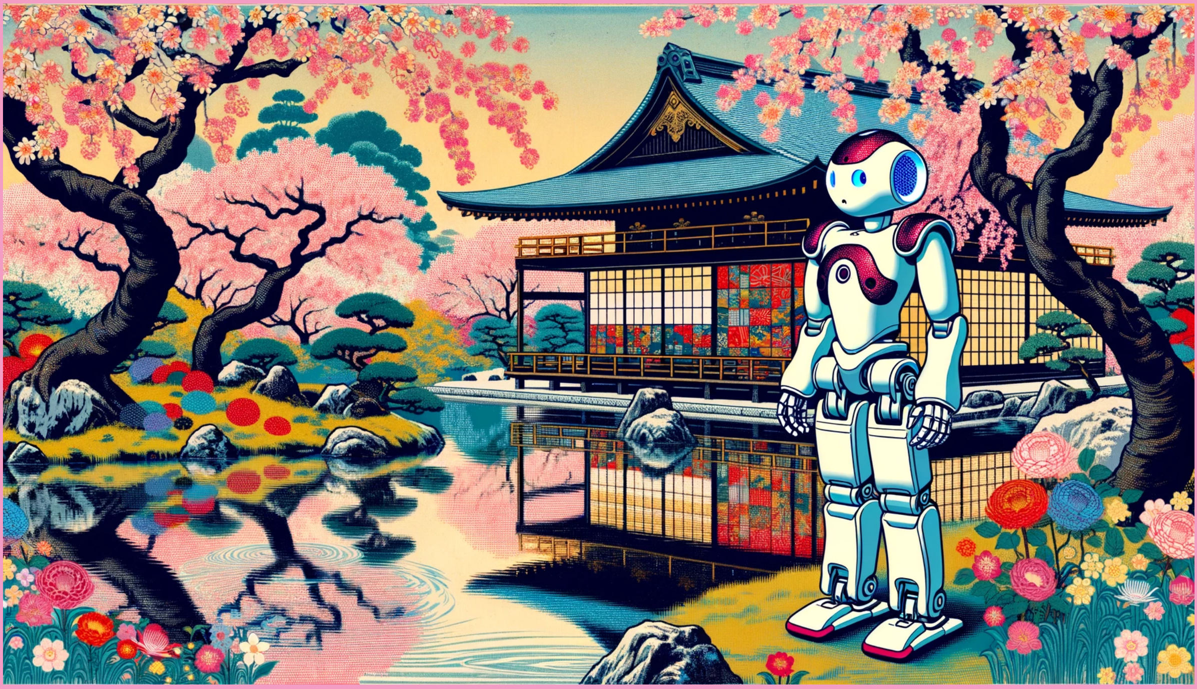
Performance of LTCs in Adapting to Joint Wear



Julian S. Blaauboer

Layout: typeset by the author using LaTeX.
Cover illustration: GPT-4

Performance of LTCs in Adapting to Joint Wear

Julian S. Blaauboer
13056131

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 900
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 900
1098 XH Amsterdam

Semester 1, 2023-2024

Abstract

The RoboCup is an international initiative with the goal to have a full team of robots beat the winner of the FIFA World Cup in the year 2050. One of the leagues hosted by RoboCup, the Standard Platform League, requires all participants to use the NAO 6 humanoid robot. In order to precisely control this robot and with modifying hardware being forbidden by the rules of the league, teams have sought to predict the sensor measurements needed for walking. However, due to joint wear the dynamics of the robot change over time, reducing the effectiveness of such a prediction model. This thesis focuses on liquid time-constant networks, a novel addition to the class of neural ODEs, and their potential to adapt to the joint wear condition of the robots. It demonstrates that although LTCs are relatively better at adapting to joint wear, their absolute performance is significantly worse than previous methods in terms of both inference time and MAE.

Contents

Chapter 1

Introduction

The RoboCup is an international competition and initiative with the goal to have a full team of robots beat the winner of the FIFA World Cup in the year 2050. RoboCup hosts multiple leagues, each of which has a different set of rules. For the Standard Platform League (SPL), all participants use identical robots, this is currently the humanoid NAO 6 developed by Aldebaran. The different teams are allowed to upload their own code to the robots before the match, but the robots must operate autonomously during the match.

Figure 1.1: Two NAO 6 robots photographed during a RoboCup SPL match.

In order to control the NAO 6 precisely and reactively, fast feedback is required. Because the hardware is fixed under the rules of the SPL, it is not possible to bring down the latency between joint angle commands and measurements by optimizing it. Marginal improvements could always be made in software, but the software overhead is insignificant in comparison.

As it is not possible to reduce this latency, another approach is to more optimally use the available information. Although it is not possible to receive the measurements faster than is physically possible, some degree of predictability can be exploited. One such approach then is to train a model that predicts these measurements n time-steps in the future. These predictions can then be fed into subsequent models and systems that ultimately control the robot. If the model is accurate, the effect will be *as if* the latency has been reduced.

For this to work, the dynamics of the NAO 6 needs to be captured by this model. To analyse these dynamics, we must discern between the interior and the exterior. The internal state is partially observable through the measurements of the joint angles, but also consists of another part which can only be indirectly observed (e.g., the joint wear). Although this hidden state might be reconstructable, it is only necessary for its effect to be *capturable* by the model (i.e., it can be a black box). The exterior can then be defined as the boundary to the interior in so far as what is behind it should not be included, but still influences it. In order for the model to be robust and properly constrained to the dynamics of the NAO 6, the dynamics behind the external factors (e.g., the control system) must not be learned. Rather, the external factors themselves (e.g., the control signal) must be taken directly.

The goal then, is to train a model that takes as input previous observations of its own state alongside any external factors and outputs the next set of observations. These external factors consist for the most part of the control commands sent to the platform. If this was not the case, the robot would barely be controllable. Of course, the robot may be held or restricted, but these circumstances are outside of its operating environment and do not need to be accounted for.

B-Human, a team from the University of Bremen, has already implemented such a model in their framework (Fiedler and Laue 2023; Reichenberg and Röfer 2023). However, as the joint wear significantly changes the dynamics of the robot, it is worth investigating if other neural network architectures can outperform the LSTM presented by Fiedler and Laue. For this purpose, this thesis is an investigation into the performance and adaptability of liquid time-constant networks (LTCs) with regards to this task.

1.1 Liquid Time-Constant Networks

Liquid time-constant networks are a type of time-continuous neural networks that belong to the broader class of neural ordinary differential equations (neural ODEs) (Hasani et al. 2021). The state of a neural ODE $\mathbf{x}(t)$ is determined by the differential equation (Chen et al. 2018):

$$\frac{d\mathbf{x}}{dt}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t; \theta) \tag{1.1}$$

Where f is a neural network parameterized by θ. To solve such an ODE requires the usage of numerical integration methods such as those belonging to the family of Runge-Kutta methods. This equation is reminiscent of the equation that defines continuous-time recurrent neural networks (CT-RNNs), which exhibit more stable behavior (Funahashi and Nakamura 1993):

$$\frac{d\mathbf{x}}{dt}(t) = -\frac{\mathbf{x}(t)}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t; \theta) \tag{1.2}$$

Liquid time-constant networks multiply the second term by $A - \mathbf{x}(t)$, yielding:

$$\begin{aligned} \frac{d\mathbf{x}}{dt}(t) &= -\frac{\mathbf{x}(t)}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t; \theta)(A - \mathbf{x}(t)) && (1.3) \\ &= -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t; \theta)\right] \mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t; \theta)A && (1.4) \end{aligned}$$

Such a network gains its name from the fact that its time-constant τ_{sys} is parameterized by the neural network f embedded in it, which can be seen by rewriting the preceding equation in the form $\tau_{sys}\frac{d\mathbf{x}}{dt}(t) + \mathbf{x}(t)$:

$$\frac{1}{1/\tau + f(\mathbf{x}(t), \mathbf{I}(t), t; \theta)} \frac{d\mathbf{x}}{dt}(t) + \mathbf{x}(t) = \frac{Af(\mathbf{x}(t), \mathbf{I}(t), t; \theta)}{1/\tau + f(\mathbf{x}(t), \mathbf{I}(t), t; \theta)} \tag{1.5}$$

This adaptation of CT-RNNs, in part inspired by the dynamics of neurons in small species, has been proven to show a marked improvement on similar methods in modelling time-series, measured in terms of expressivity, stability and performance.

It is expressable as a differentiable computational graph, and can therefore be trained using gradient-based methods (Hasani et al. 2021).

Alongside this state equation, Hasani et al. also introduce a novel ODE solver which can be characterised as a fusion between implicit and explicit Euler methods. The reason for this is that LTCs realize a system of stiff equations that would require an exponential number of steps when using a more common Runge-Kutta based algorithm.

In particular, the update step for an LTC using this method is:

$$\mathbf{x}(t+\Delta t) = \frac{\mathbf{x}(t) + \Delta t f(\mathbf{x}(t), \mathbf{I}(t), t; \theta) A}{1 + \Delta t(1/\tau + f(\mathbf{x}(t), \mathbf{I}(t), t; \theta))} \tag{1.6}$$

For this reason, LTCs are hypothesized to be effective in joint angle prediction. In this thesis we explore whether or not LTCs are effective in modelling the dynamics of the NAO 6 and its adaptability to different wear conditions it was not trained on.

Chapter 2

Method

2.1 The NAO 6 Platform

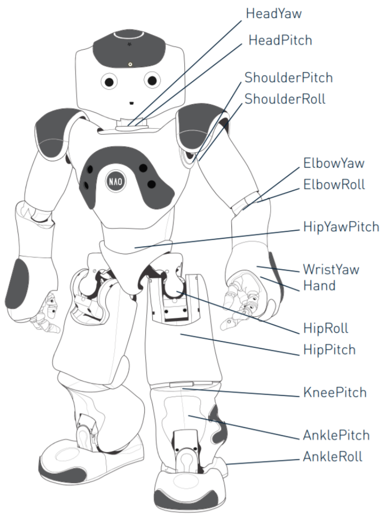
The NAO 6 has 25 distinct motor positions, of which 2 are in its head, 8 are in its arms, 4 are in its hands, and 11 are in its legs. For locomotion, only the 11 joint angles in its legs are critical. Although controlling the head, arms and hands might shift the center of mass to a more optimal position, the walking engine does not require as low a latency for these motors. These are therefore excluded from the prediction model, in order to reduce the model complexity. The joint angles in the legs are at its hip, ankles and knees and can be controlled individually. These are mirrored on each side, with the hip and ankles having both a roll and pitch component whereas the knee just has a pitch component. Alongside the left and right hip joints, there is also a central yaw-pitch control for the hip.

In order to interact with the hardware, the NAO 6 libraries include a real-time process called LoLA (Low Level Abstraction). LoLA runs at 83Hz (12ms) and communicates each cycle with the hardware through the HAL (Hardware Abstraction Layer). This means that software frameworks can simultaneously receive measurements are send commands to the platform every 12ms.

For the controller, the NAO 6 houses an Intel ATOM E3845 running at 1.91 GHz with 4 GB of DDR3 RAM, alongside 32 GB of eMMC flash memory (Franco 2022).

left	hip	roll
		pitch
	knee	pitch
	ankle	roll
		pitch
center	hip	yaw-pitch
right	hip	roll
		pitch
	knee	pitch
	ankle	roll
		pitch

(a) The 11 joint angles of interest.



(b) Positions of the motors on the NAO 6.

Figure 2.1: Overview of the controllable joint angles.

2.2 B-Human Dataset

For training the models, I have opted to use the datasets made publicly available by B-Human (B-Human 2023b). Dutch NAO Team, the RoboCup team at the University of Amsterdam, did not a have a way to collect the data before the start of this research project. Although it would have been possible to start collecting this data, B-Human has a collection spanning multiple events. Crucially, their data set starts at the 2019 German Open, which is the first year the NAO 6 was introduced. This means that at the time, all the robots were in good condition and are useful as a baseline for comparing the wear in the subsequent years. Because the data was collected during real matches, there is no risk of training on data that does not accurately reflect its operating environment.

The dataset is formatted as a comma-seperated list of values (CSV) with header. The B-Human dataset consists of one CSV per match per robot, or 266 CSVs in total. The smallest CSV consists of just 2170 rows, the largest has 94273 rows. As we know that every row corresponds to a LoLA cycle, this works out to a recording time between about 26 seconds and 19 minutes.

All CSVs follow the same column order: first 11 outputs, then 11 measurements (alongside an index column). These are named “request” and “sensor” angles respectively, and these names will from now on be used to refer to them. The request and sensor angles use the same angle order.

Robot	2019 GO	2019 RC	2022 GORE	2022 RC	Total
DasKaenguru	293270	283246	101864	295509	973889
DerPinguin	434366	303977	398843	375052	1512238
ElseKling	0	505379	0	0	505379
FriedrichWilhelm	0	0	440246	347038	787284
Gott	0	0	453895	295654	749549
Herta	343231	536729	70869	393544	1344373
JuliaMueller	232933	404322	0	271322	908577
Krapotke	280547	513428	374497	252404	1420876
MarcUwe	492172	430532	0	403282	1325986
OttoVon	208429	327846	168344	377008	1081627
Sarah	337739	522151	394362	248123	1502375

Figure 2.2: Row count per robot, per event for the 2019 German Open, the 2019 RoboCup, the 2022 GORE and the 2022 RoboCup.

request	left	hip	roll
			pitch
		knee	pitch
		ankle	pitch
			roll
	center	hip	yaw-pitch
	right	hip	roll
			pitch
		knee	pitch
		ankle	pitch
			roll

sensor	left	hip	roll
			pitch
		knee	pitch
		ankle	pitch
			roll
	center	hip	yaw-pitch
	right	hip	roll
			pitch
		knee	pitch
		ankle	pitch
			roll

Figure 2.3: Order and grouping of the fields in the datasets.

2.3 Measuring Latency

In order to measure the base latency, we need a method of correlating request angles with sensor angles. Using Mean Squared Error, we expect the offset with the smallest average MSE between the request and the sensor angles over all datasets to correspond to the latency. These results are given in Figure 2.4.

latency	offset	MSE
0 ms	0	0.001483
12 ms	-1	0.000712
24 ms	-2	0.000320
36 ms	-3	0.000339
48 ms	-4	0.000766
60 ms	-5	0.001555

Figure 2.4: MSE between requests and sensors, offset relatively to each other. 24 ms is the closest match, although the real latency is likely between 24 and 36 ms.

2.4 B-Human LSTM and Input-Output Structure

To run their LSTM within their framework on the NAO 6, B-Human has compiled their deployed model into an ONNX model, which is then loaded by ONNX Runtime (B-Human 2023a). The model deployed in their codebase, `lstm_i03l5u48.onnx`, consists of 5 layers and 48 units per layer. For their models, they have opted to use 3 frames of input, with an offset of -1 between requests and sensors. Although, as they have noted, a larger window does show a slightly higher precision, for the sake of direct comparison and faster training and inference, this experiment will use the same format.

right	ankle	roll
		pitch
	knee	pitch
	hip	pitch
		roll
left	knee	pitch
	ankle	roll
		pitch
	hip	pitch
		roll
center	hip	yaw-pitch

(a) Order and grouping of the fields in the model, note that this order differs from the order presented in Figure 2.3.

request	sensor
$i-3$	$i-3$
$i-2$	$i-2$
$i-1$	$i-1$
i	i
$i+1$	$i+1$
$i+2$	$i+2$

(b) The window taken as input (light gray) and the output (dark gray).

Figure 2.5: Structure of the inputs and outputs of B-Human's model.

2.5 Experimental Setup

2.5.1 Training

The experimental setup involves the training of six LTC models on the 2019 German Open dataset with a varying number of units and layers. While LTCs in their original paper were using learning rates ranging from 0.01 to 0.02, this yielded inferior results during preliminary training runs. Through experimentation it became evident that a lower learning rate of 0.001 yielded more desirable convergence characteristics. The reason for this is unknown, but could possibly be due to the nature of the data and in combination with other hyperparameters.

Compared to the LSTM used by B-Human, the number of units and layers has been kept relatively small, due to the higher computational complexity and the fact that the expressivity demonstrated by LTCs means that a smaller number of units and layers should nevertheless be comparable to larger models with simpler internals.

For numerical integration, the LTCs use the semi-implicit Euler method introduced by Hasani et al., with 6 as the fixed number of steps.

Moreover, a batch size of 128 was deemed a good compromise between computational efficiency and model performance. Higher batch sizes, while significantly decreasing training time, did underperform, while lower batch sizes did not yield significantly better results to justify the increased training time.

The optimization process was guided by the Adam algorithm ($\beta_1 = 0.99, \beta_2 = 0.999, \hat{\epsilon} = 10^{-7}$). For both the training and validation loss, the Mean Absolute Error (MAE) was used. The rationale for this is that possible anomalies in the dataset are hard to identify, as we only have the joint angles to go off, and so while the Mean Squared Error (MSE) would be the more common approach for this kind of regression model, it gives a much heavier weight to these potential anomalies.

Alongside these LTCs, an LSTM with the same parameters as the one used by B-Human was trained. While it would have been possible to use their compiled model, this would leave differences in the training data and other hyperparameters uncontrolled for.

The dataset was partitioned using the typical scheme of 60% training data, 20% validation data, and 20% test data. As the dataset consists of multiple series of

Name	Units per layer	Layers
`lstm-48-5`	48	5
`ltc-16-1`	16	1
`ltc-32-1`	32	1
`ltc-64-1`	64	1
`ltc-16-2`	16	2
`ltc-32-2`	32	2
`ltc-64-2`	64	2

Figure 2.6: Overview of the evaluated models.

data that differ in length, the partitioning of this data takes the form of a subset sum problem. A stochastic method was used to solve this problem, the combined row counts of each partition are laid out in Figure 2.7.

Partition	Split	Target	Actual	Error
Training	60%	1573612.2	1573614	+1.8
Validation	20%	524537.4	524542	+4.6
Test	20%	524537.4	524531	−6.4
Total	100%	2622687	2622687	

Figure 2.7: The partition sizes of the dataset in rows.

2.5.2 Evaluation

To evaluate the trained models, they were run on the test partition after convergence. As this partition is part of the 2019 German Open dataset, it is a measure of the models' overall performance. Again, the MAE is used for the same reasons as outlined in the previous subsection. Because of the limited data available in this partition per robot, it is insufficient to evaluate the models' performance per robot from this partition alone. Rather, the entire dataset was used, including training data, to serve as a baseline measurement. This is possible, because the final analysis looks at their relative performance, and the per robot baseline provides a normalized index for the subsequent events.

Afterwards, each model was run on each robot in each subsequent event. These results were normalized as a fraction of the aforementioned baseline. This way, the results can be compared to each other as an insight into the models' ability to adapt to the accumulated joint wear. The robots that were absent during the 2019 German Open have been omitted from the results.

Lastly, while there are too many variables to conclusively say whether or not the models can run on the NAO 6, as this depends on the load that the rest of the framework takes up, we can measure the inference time relative to the LSTM as a hint of the models' performance.

Chapter 3

Results

Figure 3.1a shows that all models have converged after 75 epochs. For the 16 unit-variants of the LTC, the single-layer LTC initially converged faster than its dual-layer complement. This is true for the other LTCs as well, although the effect becomes less pronounced as the number of units increases.

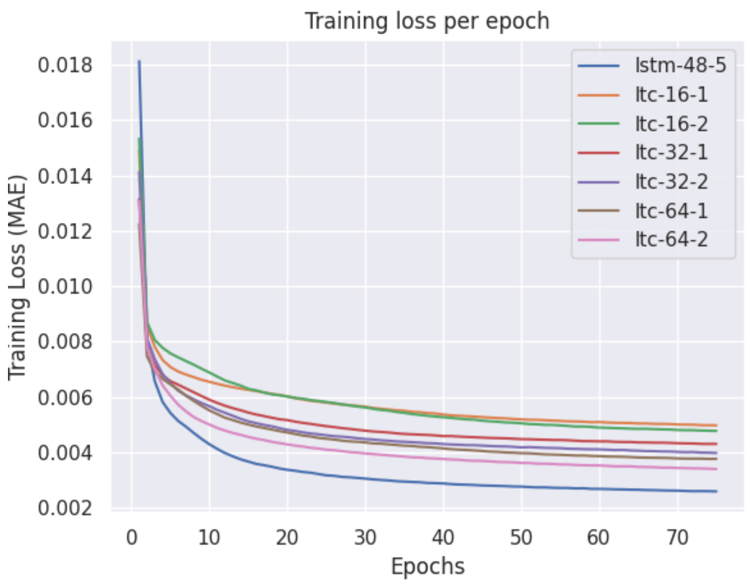
The validation loss as shown in Figure 3.1b is a lot noisier, as the validation partition is only one third the size of the training partition, but nevertheless shows the same ordering of the different models and roughly the same loss as Figure 3.1a

As can be seen in Figure 3.2, the overall performance of all LTCs is worse than the LSTM in terms of the MAE. For every number of units, the dual-layer LTC performs better than its single-layer equivalent. The same is true for the average absolute MAE as presented in Figure 3.4, where the LSTM still outperforms the LTCs at every event.

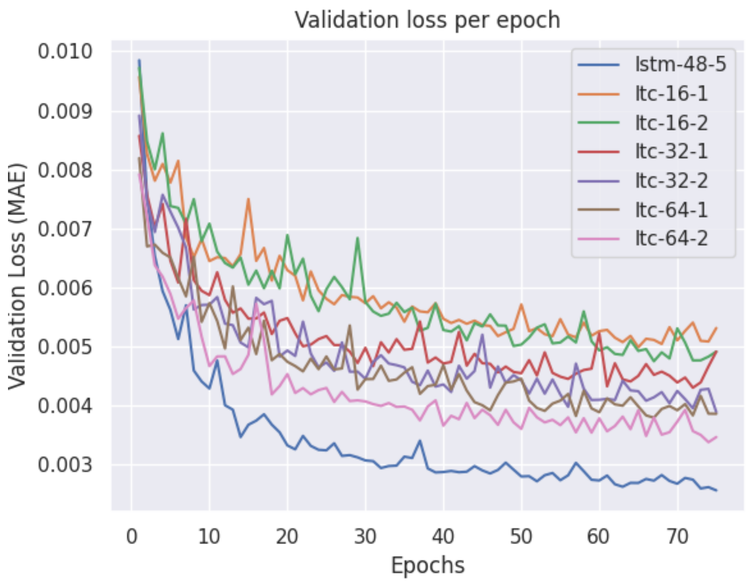
However, the results from Figure 3.5 do show that all LTCs relatively outperform the LSTM in adapting to the different wear conditions in subsequent events. The relative MAE of the 2019 RoboCup is very close to 1 relative to the other events. This is to be expected, as wear between the 2019 German Open and the 2019 RoboCup is negligable compared to the three year gap between 2019 RoboCup and the 2022 GORE. Especially the `ltc-32-1` performs well in this regard, with both the smaller and larger models performing worse. It is important to note that this LTC has the second-worst MAE on the test partition. Interestingly, every dual-layer LTC performs worse than its single-layer equivalent.

Looking at Figure 3.3, it is clear that running any LTC is much more CPU intensive than the LSTM, even though the LSTM has many more layers, with the largest LTC being more than two orders of magnitude slower. Even the fastest LTC, the `ltc-16-1`, still takes 3.5 times the amount of time to run. The dual-layer LTCs take roughly twice the amount to run as their single-layer counterparts. A doubling of the number of units corresponds to a 3.5 to 4.5 times increase in inference time.

3.1 Performance on the Test Partition



(a) Training



(b) Validation

Figure 3.1: Loss (MAE) per epoch, although (b) is very noisy compared to (a) (due to less data), the order is the same.

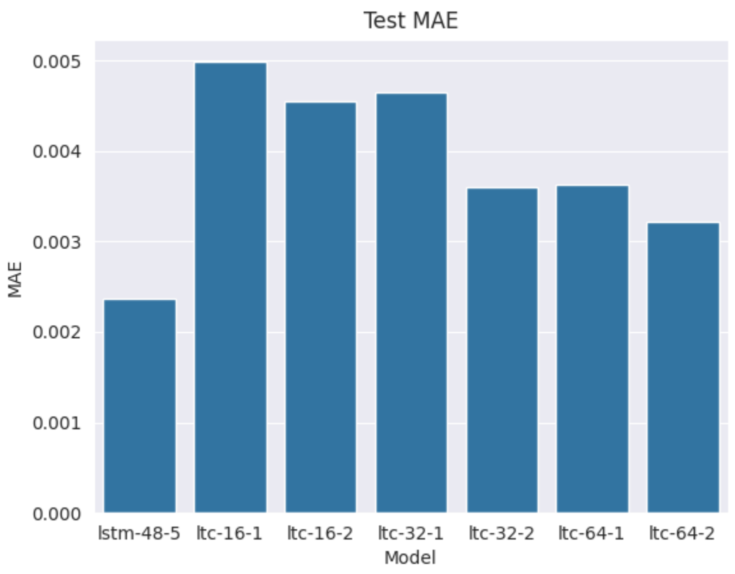


Figure 3.2: MAE on test partition of the 2019 German Open dataset.

3.2 Performance in Terms of Inference Time

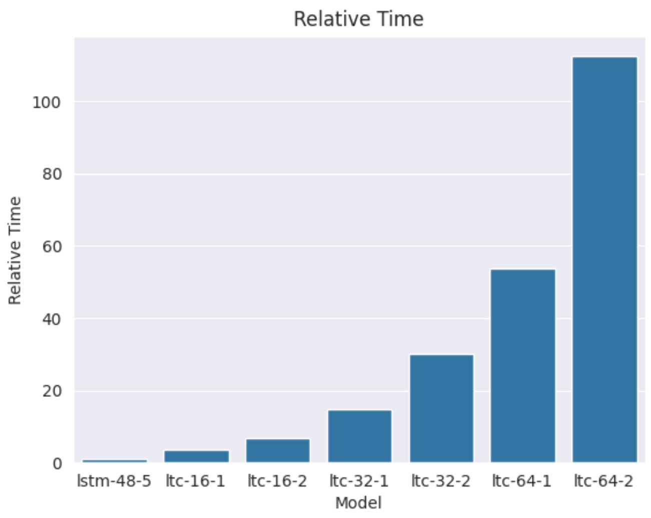


Figure 3.3: Inference time relative to `lstm-48-5` (running on an Intel i9-13900KF).

3.3 Performance on Subsequent Events

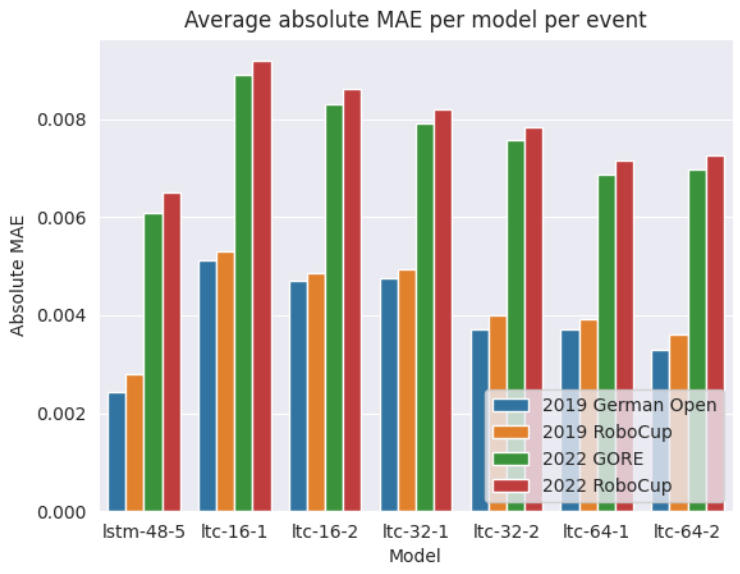


Figure 3.4: Absolute MAE per model per event averaged over all robots.

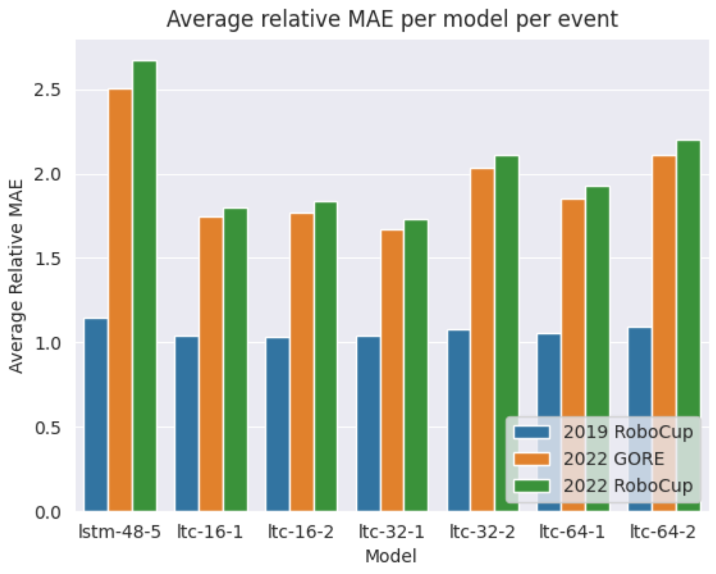


Figure 3.5: Relative MAE per model per event averaged over all robots.

Chapter 4

Discussion

Although the LTCs do seem to adapt relatively better to joint wear than the LSTM, the additional computational complexity and especially the worst absolute performance as presented in Figures 3.3, 3.2 and 3.4 make them an infeasible alternative to the LSTM.

While the inference time of `ltc-16-1` might still be acceptable, it is significantly outperformed by every other model. The closest contender to the LSTM is `ltc-64-2`, but the inference time of the latter is more than two orders of magnitude larger than the former, which forecloses the possibility of training and deploying an even larger LTC model.

It might be possible to reduce the number of integration steps performed by the LTC to bring down the inference time, but this would be detrimental to its performance. There does not seem to be an obvious way to alter the model to bring both the MAE and inference time down at the same time.

As such, a better approach would simply be to periodically re-train an LSTM on new data, which can be passively collected during use. As the wear condition might differ from robot to robot, this would ideally mean training a separate model for each robot. Another approach could be to perform a calibration run where certain pre-defined movements are performed to assess the current joint wear of the robot. From this calibration, the robots could be grouped based on similarity, and the data collected by them could be merged into a single dataset per group.

4.1 Conclusion

In conclusion, although LTCs have generally shown good performance on time-series prediction tasks in the relevant literature, they are unsuitable for adapting to joint wear compared to previous methods. In particular, this is due to the worse absolute performance and the significant increase in inference time. Still, they did show some relative improvement over the LSTM when comparing their performance between the event they were trained on and the subsequent events.

My recommendation for dealing with joint wear would be to periodically re-train an LSTM if a robot has sustained significant wear compared to the last data it was trained on. Additionally, a calibration test could be performed to group the robots by wear level if the data collected during runs is insufficient to adequately train a model on.

4.2 Future Research

Even though the LTCs did not perform better than the LSTMs, it might be possible to combine to combine the architectures into a single model (e.g., an LTC layer followed by an LSTM layer or vice versa). Since they did do relatively better at adapting to joint wear, other architectures in or proximate to the family of neural ODEs might also be worth investigating.

With regards to the broader context in which this model is deployed, rather than have a model architecturally completely separate from the walking engine, a more closely coupled solution which might leverage information about the walking engine could outperform current frameworks. However, all the drawbacks of a closer coupling apply, as it would make development and tweaking of either system much harder.

Bibliography

B-Human (2023a). *B-Human Code Release.* URL: `https://github.com/bhuman/BHumanCodeRelease` (visited on 02/16/2024).

— (2023b). *B-Human Joint Angles Public Dataset.* URL: `https://b-human.informatik.uni-bremen.de/public/datasets/joint_angles` (visited on 02/16/2024).

Chen, Ricky T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In: *Advances in Neural Information Processing Systems.* Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc. URL: `https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf`.

Fiedler, Jan and Tim Laue (2023). “Neural Network-based Joint Angle Prediction for the NAO Robot”. In: *RoboCup Symposium 2023.*

Franco, Herve (2022). *NAO - Technical Specifications.* URL: `https://support.unitedrobotics.group/en/support/solutions/articles/80000959718-nao-technical-specifications` (visited on 02/16/2024).

Funahashi, Ken-ichi and Yuichi Nakamura (1993). “Approximation of dynamical systems by continuous time recurrent neural networks”. In: *Neural Networks* 6.6, pp. 801–806. ISSN: 0893-6080. DOI: `https://doi.org/10.1016/S0893-6080(05)80125-X`. URL: `https://www.sciencedirect.com/science/article/pii/S089360800580125X`.

Hasani, Ramin et al. (May 2021). “Liquid Time-constant Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.9, pp. 7657–7666. DOI: `10.1609/aaai.v35i9.16936`. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/16936`.

Reichenberg, Philip and Thomas Röfer (2023). “Dynamic Joint Control For A Humanoid Walk”. In: *RoboCup Symposium 2023.*

Appendix A

Relative MAE per Model per Event per Robot

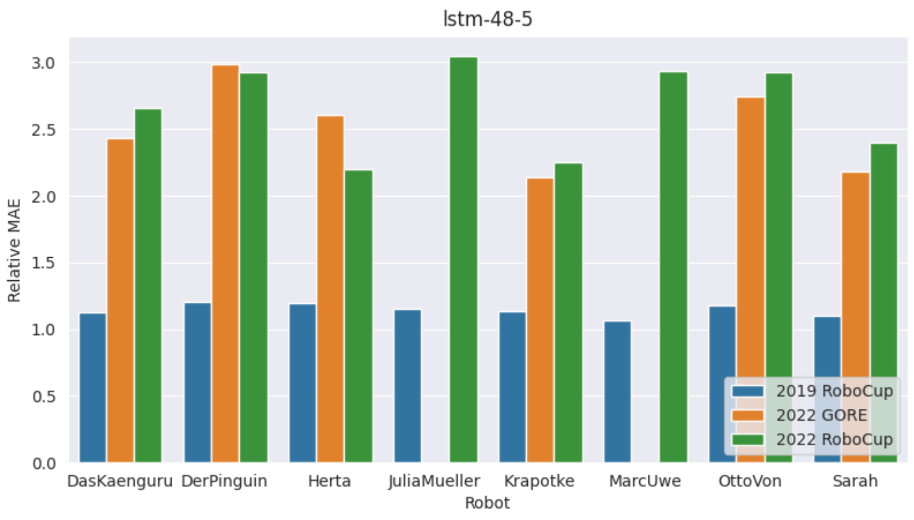


Figure A.1: Relative MAE on subsequent events for `lstm-48-5`.

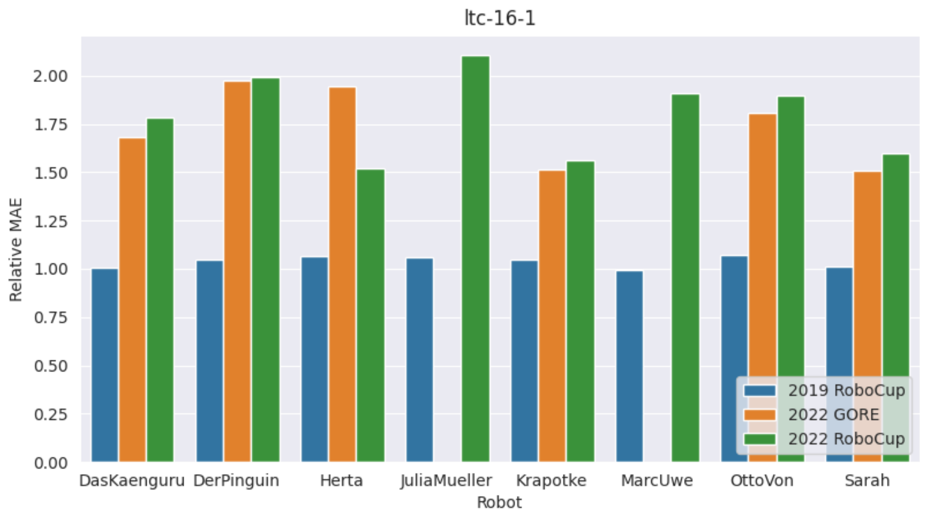


Figure A.2: Relative MAE on subsequent events for `ltc-16-1`.

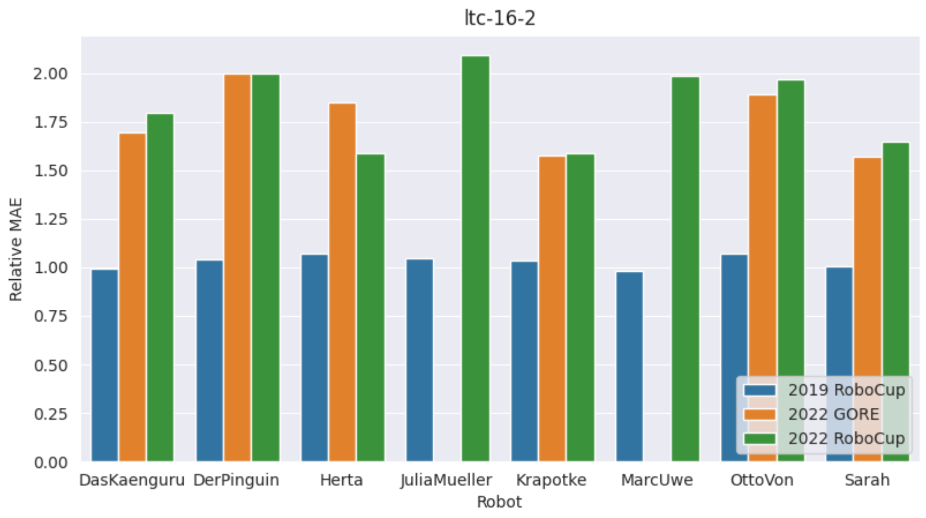


Figure A.3: Relative MAE on subsequent events for `ltc-16-2`.

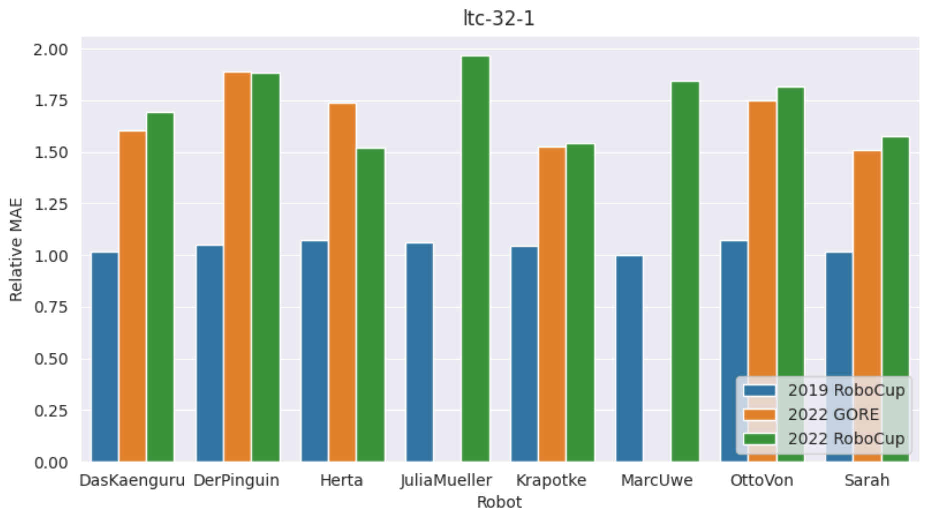


Figure A.4: Relative MAE on subsequent events for `ltc-32-1`.

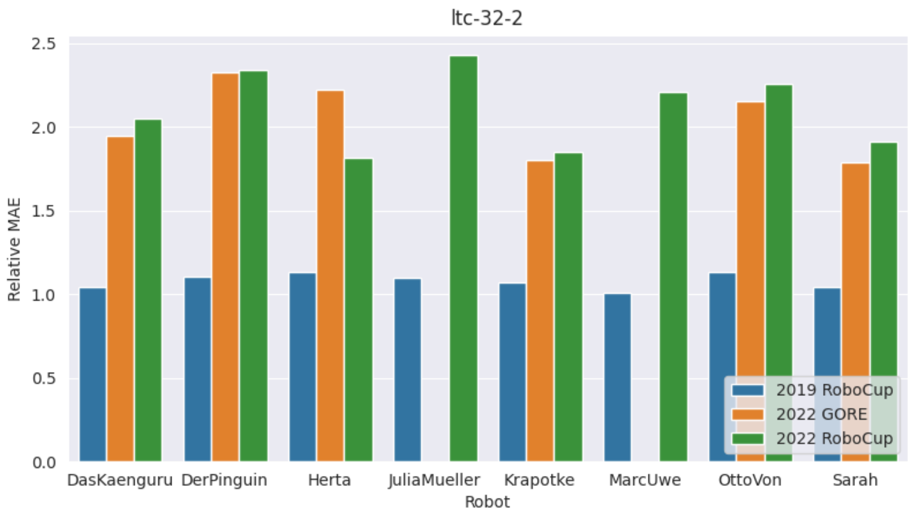


Figure A.5: Relative MAE on subsequent events for `ltc-32-2`.

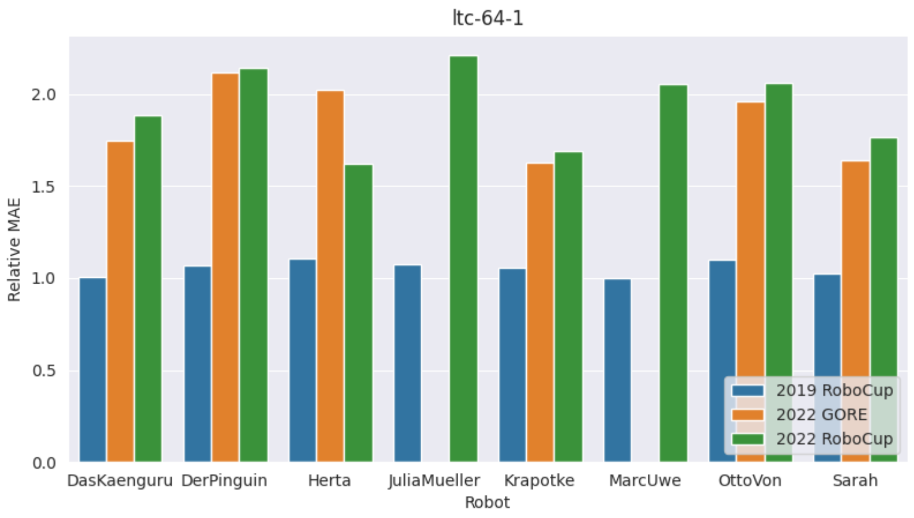


Figure A.6: Relative MAE on subsequent events for `ltc-64-1`.

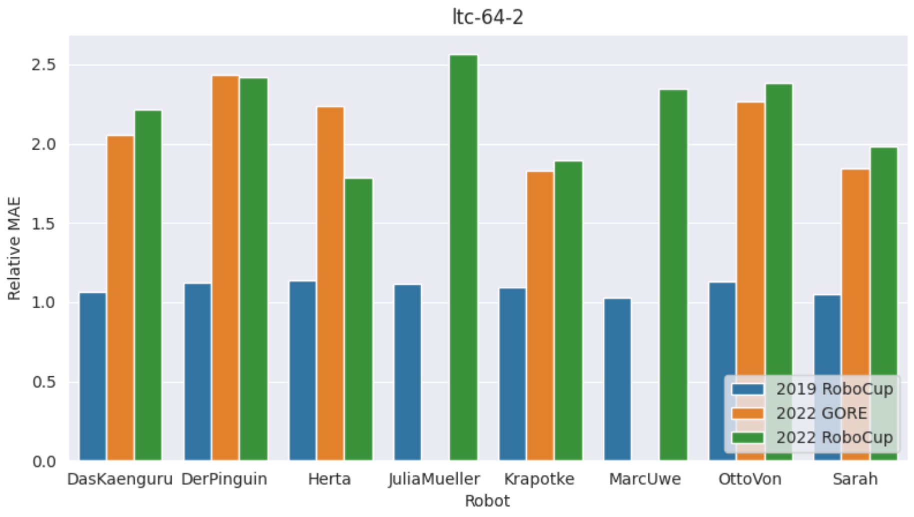


Figure A.7: Relative MAE on subsequent events for `ltc-64-2`.