2017 IEEE-RAS 17th International Conference on
Humanoid Robotics (Humanoids)
Birmingham, UK, November 15-17, 2017

Learning Inverse Dynamics Models in O(n) time with LSTM networks

Elmar Rueckert[1] and Moritz Nakatenus[1] and Samuele Tosatto[1] and Jan Peters[1,2]

Abstract— **Inverse dynamics model learning is crucial for modern robots where analytic models cannot capture the complex dynamics of compliant actuators, elasticities, mechanical inaccuracies, frictional effects or sensor noise. However, such models are highly nonlinear and millions of samples are needed to encode a large number of motor skills. Thus, current state of the art model learning approaches like Gaussian Processes which scale exponentially with the data cannot be applied. In this work, we developed an inverse dynamics model learning approach based on a long-short-term-memory (LSTM) network with a time complexity of $O(n)$. We evaluated the approach on a KUKA robot arm that was used in object manipulation skills with various loads. In a comparison to Gaussian Processes we show that LSTM networks achieve better prediction performances and that they can be trained on large datasets with more than $100,000$ samples in a few seconds. Moreover, due to the small training batch size of for example 128 samples, the network can be continuously improved in life-long learning scenarios.**

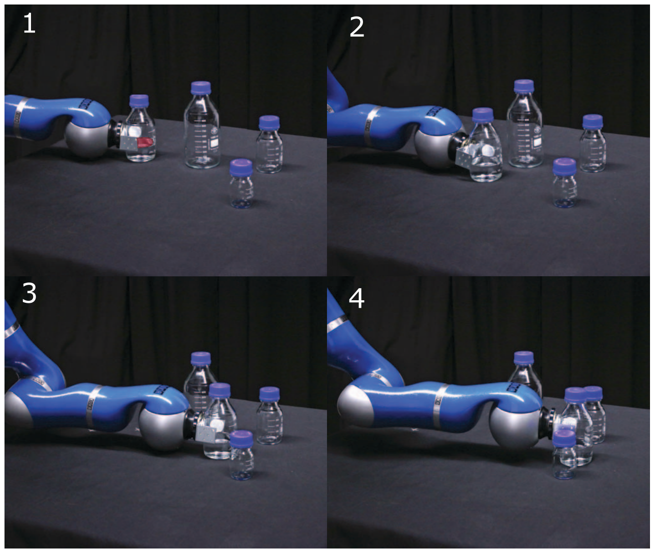

Fig. 1. Illustration of four snap shots of the investigated manipulation task. The robot had to push the flask that was filled with 200, 300, or 400 ml of liquid. The goal of this work is to predict the joint torques which depend on the fill level given joint angles, velocities and accelerations.

I. INTRODUCTION

In robotics, predictions based on dynamics models are essential for control, object manipulation or planning [14], [24]. However, for modern robots with its dozens of compliant actuators and thousands of noisy tactile or visual sensors, these models have to be learned because of unmodeled effects like manufacturing uncertainties, sensory noise or any form of unmodeled dynamics in presence of contacts [4]. The challenge is that even small model inaccuracies can result in catastrophic behavior [25] and for learning a rich repertoire of versatile skills millions of samples in high dimensional observational spaces need to be processed [27].

Gaussian Processes (GPs) [30], [21] are state of the art regression techniques to learn such model. They are widely used in machine learning because their hyper parameters can be optimized through maximizing the marginal likelihood and thus, it requires little effort to adapt the model to new datasets or problem domains. However, the main limitation of GPs is that the computational complexity scales exponentially in $O(n^3)$ due to the need to invert a $n \times n$ kernel matrix.

To reduce this computational demands local approximations based on sparse GPs [19], mixture of experts [28], [20], drifting GPs [13], or based on local partitions of the data [15] were proposed. However, also these local approximations scale quadratically with the samples, i.e., in $O(n^2)$ and are thus limited to few thousand samples [15].

Local approximations like the locally weighted projection regression (LWPR) approach [29] or even simpler lazy learning approaches like locally weighted regression [3] would scale linear with the number of samples in $O(n)$. However, tuning the often large number of hyper parameters, which are the Gaussian bandwidth parameters of many local models, can be as challenging as the original model learning task.

We developed in this work an inverse dynamics model learning approach based on a recurrent neural network with a time complexity of $O(N)$. To learn from long time series data we utilized the gating mechanism in Long-short-term-memory (LSTM) networks [9], [26] which do not suffer from vanishing gradients. Pairs of joint states and motor torques at a given time step were used as inputs and outputs, respectively. Network updates were applied after presenting batches of these pairs using backpropagation-through-time [31].

In a real robot manipulation experiment, where a KUKA arm pushes flasks, we demonstrate how LSTMs can learn the dynamics model of the torque controlled robot arm and compare to the state of the art approach GPs. The model-based LSTM controller achieves superior prediction performances and scales linearly with the number of data samples. These results can be exploited in neural model-based control approaches that can learn from high-dimensional visual and tactile data.

[1]Intelligent Autonomous Systems Lab, Technical University Darmstadt, Germany. {rueckert, nakatenus, tosatto}@ias.tu-darmstadt.de
[2]Robot Learning Group, Max-Planck Institute for Intelligent Systems, Tuebingen, Germany mail@jan-peters.net

II. METHODS

We start with a problem definition for learning inverse dynamics models and subsequently discuss Gaussian Processes (GPs) and Long-short-term-memory (LSTM) networks implementations.

A. Problem Definition

We denote the vectors for joint angles, for joint velocities and for joint accelerations by $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$, respectively. For a robot with d joints these vectors are of the dimension $\mathbb{R}^{d \times 1}$. Using these definitions the general inverse dynamics model of a robot is given by

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + h(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\epsilon}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \,, \qquad (1)$$

where $\boldsymbol{\tau}$ denotes the unknown joint torques which we want to learn. The inertia matrix is denoted by $M(\boldsymbol{q})$ and the term $h(\boldsymbol{q}, \dot{\boldsymbol{q}})$ combines effects of Coriolis and centripetal forces, friction and of gravitational forces, i.e.,

$$h(\boldsymbol{q}, \dot{\boldsymbol{q}}) = C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + f_r(\dot{\boldsymbol{q}}) + g(\boldsymbol{q}) \,.$$

All unmodeled dynamics such as elasticities in the mechanical designs, model parameter inaccuracies in the masses or inertiae, vibrational effects, Stribeck friction, couplings and sensor noise are modeled by the term $\boldsymbol{\epsilon}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ in Eq. 1.

The goal of this work is to learn such inverse dynamics models in Eq. 1 including all effects of the unmodeled dynamics. We formulate this learning problem as a standard regression task. Given some input vector $\boldsymbol{x}$ the goal is to learn the function

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\zeta} : \mathbb{R}^{3d \times 1} \mapsto \mathbb{R}^{d \times 1} \,, \qquad (2)$$

where $\boldsymbol{y} = \boldsymbol{\tau}$ and $\boldsymbol{x} = [\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T, \ddot{\boldsymbol{q}}^T]^T$. The variable $\boldsymbol{\zeta}$ denotes zero mean Gaussian noise with a standard deviation of σ_y.

Using a dataset $\mathfrak{D} = \langle \boldsymbol{x_t}, \boldsymbol{y_t} \rangle_{t=1,...,n}$ of n input-output pairs, we evaluate the trained models by computing the error

$$MSE = \frac{1}{d\,n} \sum_{j=1}^{d} \sum_{t=1}^{n} \left(\hat{y}_t^{[j]} - \tilde{y}_t^{[j]} \right)^2 \,, \qquad (3)$$

where $\hat{\boldsymbol{y}} = [\hat{y}^{[1]}, ..., \hat{y}^{[d]}]^T$ denotes the true label and $\tilde{\boldsymbol{y}}$ the model prediction.

In the following two subsections we will discuss how Gaussian Processes (GPs) and how Long-short-term-memory (LSTM) networks can be used to learn this mapping.

B. Inverse Dynamics Model Learning with GPs

Gaussian Processes (GPs) are state of the art model learning or regression approaches [30], [21] that were successfully used for learning inverse dynamics models in robotic applications [15], [4]. For comprehensive discussions we refer to [25], [14]. Here we briefly discuss them as we will use them for a comparison in our experiments.

GPs represent a distribution over inverse dynamics models in Eq. 2 of the form $f \sim \mathrm{GP}(m, k)$. This representation is fully defined by the mean m and the covariance k. We chose as covariance function a *Matérn kernel* [12]. It is a generalization of the *squared-exponential kernel* that has an additional parameter ν which controls the smoothness of the resulting function. The smoothing parameter can be beneficial for learning local models.We used *Matérn kernels* with $\nu = 5\,/\,2$ that are defined by

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma^2 \, \frac{1}{2^{\nu-1}\,\Gamma(\nu)} \, A^{\nu} \, \mathrm{H}_{\nu} \, A + \sigma_y^2 \, \delta_{pq} \,,$$

where Γ is the gamma function, $A = (2\sqrt{\nu}||\boldsymbol{x}_p - \boldsymbol{x}_q||)\,/l$ and H_{ν} is a modified *Bessel* function [2]. The length-scale parameter of the kernel is denoted by σ, the variance of the latent function is denoted by σ and δ_{pq} is the *Kronecker* delta function (which is one if $p = q$ and zero otherwise). Note that for $\nu = 1/2$ the *Matérn kernel* implements the *squared-exponential kernel*. In our experiments we optimized the hyper parameters $\boldsymbol{\theta} = [\sigma, l, \sigma_y]$ by maximizing the marginal likelihood [21].

Computing Predictions: Given a test sample $\boldsymbol{x}_*$ the predictive distribution is defined by

$$p(\tilde{\boldsymbol{y}}|\mathfrak{D}, \boldsymbol{x}_*, \boldsymbol{\theta}) = \mathcal{N}(\mu_{GP}, \sigma_{GP}) \,, \qquad (4)$$

with $\mu_{GP} = \boldsymbol{k}_*^T \, \boldsymbol{K}^{-1} \boldsymbol{y}$ and $\sigma_{GP} = k_{**} - \boldsymbol{k}_*^T \, \boldsymbol{K}^{-1} \, \boldsymbol{k}_*$. The matrix entries in $\boldsymbol{K}$ are $K_{pq} = k(\boldsymbol{x}_p, \boldsymbol{x}_q)$, the scalar $k_{**} = k(\boldsymbol{x}, \boldsymbol{x})$, and $\boldsymbol{k}_* = k(\boldsymbol{X}, \boldsymbol{x}_*)$ with $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$.

C. Inverse Dynamics Model Learning with LSTM Networks

Long-short-term-memory (LSTM) networks are popular recurrent neural networks for modeling long time series [10]. Special gating mechanisms were added to classical recurrent neural networks to avoid the vanishing gradient problem in back propagation through time [9], [26]. In this subsection, we will give a short introduction to LSTMs and discuss the model with respect to inverse dynamics model learning problems.

In the following, subscripts will be used like in the previous subsections to index samples like in the dataset $\mathfrak{D} = \langle \boldsymbol{x_t}, \boldsymbol{y_t} \rangle$ at time t. However, in addition superscripts are introduced to denote the layer $l = 1, ..., L$, where the output at the last layer $\boldsymbol{h}_t^L = \tilde{\boldsymbol{y}}_t$ is used to compute the prediction error, e.g. by evaluating Eq. 3. The input to the first layer is denoted by $\boldsymbol{a}_t^{l=1} = \boldsymbol{x}_t$ and the next layer's input is set to the prediction of the previous layer, i.e., $\boldsymbol{a}_t^{l+1} = \boldsymbol{h}_t^l$.

At layer l a LSTM network is fully defined by the equations

$$\begin{aligned}
\boldsymbol{i}_t^l &= \sigma(\boldsymbol{W}_{xi}^l \, \boldsymbol{a}_t^l + \boldsymbol{W}_{hi}^l \, \boldsymbol{h}_{t-1}^l + \boldsymbol{b}_i^l) \,, \\
\boldsymbol{f}_t^l &= \sigma(\boldsymbol{W}_{xf}^l \, \boldsymbol{a}_t^l + \boldsymbol{W}_{hf}^l \, \boldsymbol{h}_{t-1}^l + \boldsymbol{b}_f^l) \,, \\
\boldsymbol{c}_t^l &= \boldsymbol{f}_t^l \odot \boldsymbol{c}_{t-1}^l + \boldsymbol{i}_t^l \odot \tanh(\boldsymbol{W}_{xc}^l \boldsymbol{a}_t^l + \boldsymbol{W}_{hc}^l \boldsymbol{h}_{t-1}^l + \boldsymbol{b}_c^l) \,, \\
\boldsymbol{o}_t^l &= \sigma(\boldsymbol{W}_{xo}^l \, \boldsymbol{a}_t^l + \boldsymbol{W}_{ho}^l \, \boldsymbol{h}_{t-1}^l + \boldsymbol{b}_o^l) \,, \\
\boldsymbol{h}_t^l &= \boldsymbol{o}_t^l \odot \sigma(\boldsymbol{c}_t^l) \,,
\end{aligned}$$

where the vectors $\boldsymbol{i}, \boldsymbol{f}$ and $\boldsymbol{o}$ denote the input, the forget and the output gate, and $\boldsymbol{c}$ represents the memory cell of the network. The symbol $\sigma(\,.\,)$ denotes the sigmoid function and $\odot$ is the *Hadamard* or element-wise product.

We used a simple model implementation that neglects recurrent connections from the memory cell to the gates [16] or dropout implementations [8].

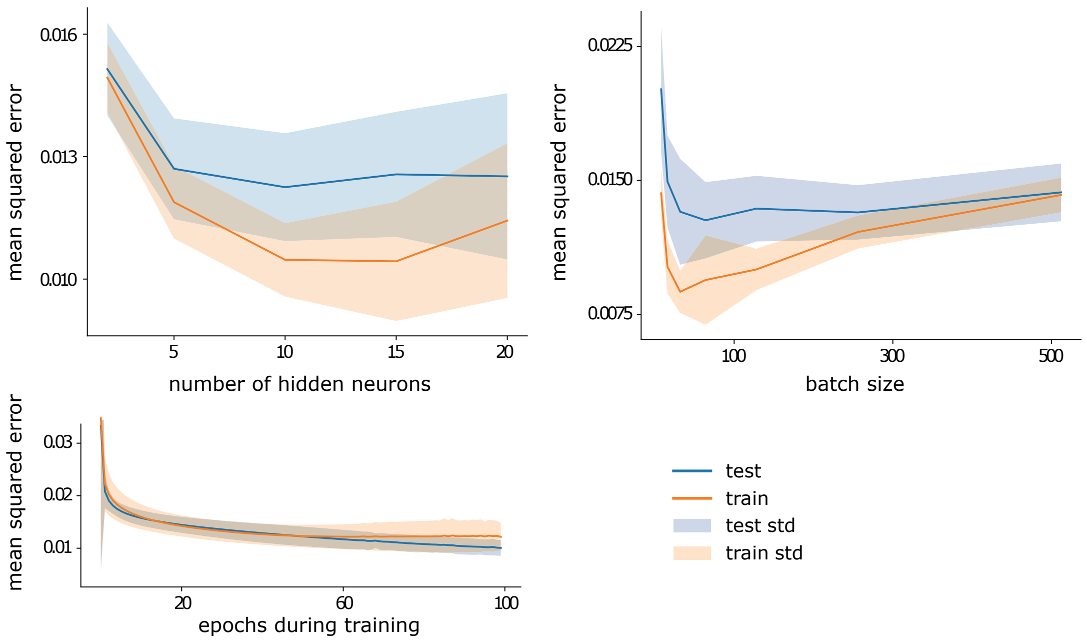

Fig. 2. Hyper parameter optimization results using LSTMs. Shown are the training and the test error for variations of the number of hidden neurons $\in \{2, 5, \mathbf{10}, 15, 20\}$, the batch size $\in \{8, 16, 32, 64, \mathbf{128}, 256, 512\}$ and the number of epochs $\in \{1, 10, 20, 50, \mathbf{100}\}$ used during training. Bold numbers mark the used parameter value. We also evaluated different numbers of layers, which is not shown as more than two layers did not lead to any improvement.

Computing Predictions: The hidden value or output of LSTMs for inverse dynamics model learning is given by

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{o}_t^L \odot \sigma(\boldsymbol{c}_t^L)\,,$$

where L denotes the number of layers in the LSTM network.

Training the LSTM network: Like for the GPs we assume a dataset $\mathfrak{D} = \langle \boldsymbol{x_t}, \boldsymbol{y_t} \rangle_{t=1,\dots,n}$ of n input-output pairs of joint states and motor torques, which are of the dimension $\mathbb{R}^{3d \times n} \mapsto \mathbb{R}^{d \times n}$. For training the LSTM network batches of u samples are constructed. For example the first set is defined by $\mathfrak{B} = \langle \boldsymbol{x_t}, \boldsymbol{y_t} \rangle_{t=1,\dots,u} \in \mathbb{R}^{3d \times u} \mapsto \mathbb{R}^{d \times u}$. We used backpropagation-through-time [31] with adaptive learning rates [6]. In our experiments and initial learning rate of 0.9 was found to perform best on the real robot dataset.

All experiments were performed using an efficient *tensorflow* [1] implementation of a two-layer LSTM network. We initialized the *forget gates* with values of 1 in order to improve the influence of past knowledge in early updates. The memory cells were initialized with values of zero and prior to our comparison to Gaussian Processes, we also optimized the batch size, the number of epochs and the number of neurons of each of the two layers using on the real robot dataset. These results are shown in Figure 2.

D. On the effect of noise on the prediction accuracy in a synthetic dataset

Prior to evaluations on real data which is discussed in the results section, we want to evaluate the effect of noise on the prediction performance. For that a synthetic dataset of one dimensional ($d = 1$) *sine*, *triangle* and *sawtooth* functions is used to explore the robustness to noise.

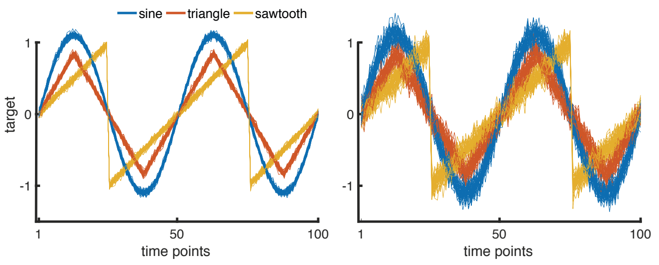

Fig. 3. Training data of a synthetic time series prediction task. The left panel shows the training data for the *low noise* condition with zero mean Gaussian noise with a standard deviation of 0.01. In the right, a standard deviation of 0.1 was used.

TABLE I

MEAN SQUARED ERRORS OF THE PREDICTIONS ON THE SYNTHETIC DATASET FOR ADDITIVE ZERO MEAN GAUSSIAN NOISE WITH A STANDARD DEVIATION OF 0.01 (LOW NOISE) AND 0.1 (LARGE NOISE CONDITION). THE $\pm$ SYMBOL DENOTES THE STANDARD DEVIATION OVER 20 EXPERIMENTS.

	Naive	**GPs**	**LSTMs**
low noise	0.264 ± 0.084	0.030 ± 0.011	**0.010** ± 0.004
large noise	0.310 ± 0.078	0.086 ± 0.016	**0.075** ± 0.014

We generated two synthetic datasets with additive zero mean Gaussian noise (with $\sigma = \{0.01{,}0.1\}$) to test the robustness of the regression techniques to noise. The data is illustrated in Figure 3 and shows an overlay of three functions, i.e., a *sine*, a *triangle* and *sawtooth* function.

With increasing noise level, the mean squared error of the model's predictions increased. In both conditions, LSTM networks outperform GPs. These results are shown in Table I. In Figure 4, we additionally illustrate some example predictions.

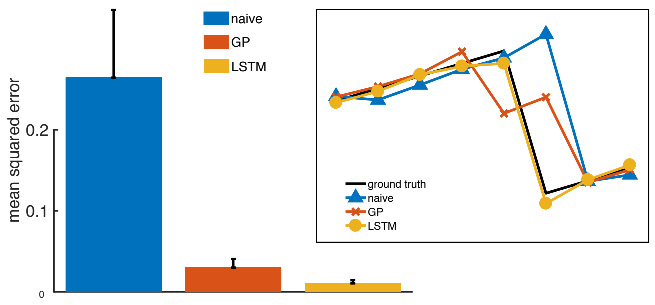

Fig. 4. Comparison of regression models on the synthetic time series prediction task. In the box, prediction results for eight consecutive time steps at phase $\pi/2$ are shown. The dark solid line denotes the ground truth target values.

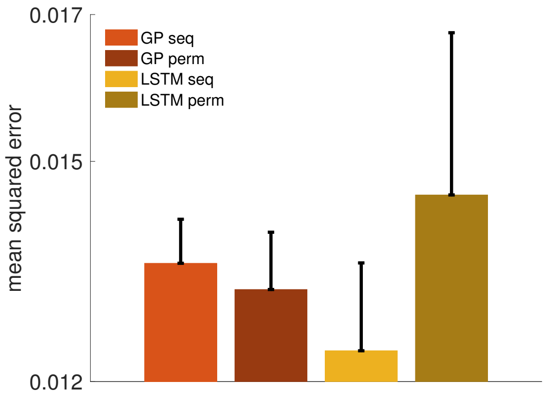

Fig. 5. Comparison of regression models on sequential and non-sequential **real-robot** data, where we eliminated all temporal correlations through random permutations. While there is no significant difference in the GP results, the LSTM network can exploit temporal correlations in the data, i.e, the robot joint angle, the velocity and the acceleration trajectories.

We also compared to a naive approach which simply computes the output based on the sum of the current state and the current velocity. This approach fails to compute accurate predictions at the turning points of the functions, which is shown in the inlay in Figure 4.

III. RESULTS

We evaluated the prediction performance and the computational times for training and for generating predictions in a *dynamics* model learning task using a torque controlled robot arm of KUKA. In our experiment, the task was to push a flask that was filled with liquid along a curved trajectory to a desired goal location. An example is shown in Figure 1. We recorded the joint angles, the joint velocities, the joint accelerations as well as the joint torques in a teleoperation experiment using a mirroring of commands approach in a bimanual setting. Note that through kinesthetic teaching the joint torques would be corrupted.

For training we used demonstrations of manipulation tasks with flask fill levels of 200 and 400 milliliters. For testing trials with a flask fill level of 300 milliliters were used which results in a challenging prediction problem. We evaluated different training set sizes ranging from 100 to $100,000$ samples, which had proportional number of test samples, ranging from 20 to $16,000$. The dataset is online available [18].

To evaluated the prediction performances and the required computational time for generating online predictions we trained GPs as well as our LSTM network from data of the first five joints ($d = 5$) of the arm. The remaining two wrist joints were hardly used in the experiment. The resulting dataset contained of a 15 dimensional input space, i.e., positions, velocities and accelerations and of five dimensional torque vectors as outputs or unknowns.

All results show average statistics over 20 cross validation experiments.

A. On the benefit of memory models for sequential data

An important question is if models with memory like LSTMs can capture and exploit temporal correlations in the inverse dynamics model learning data. To test that we compared the prediction performance of LSTMs trained on the **real robot** sequential data with models trained on data where we removed all temporal correlations through random permutations. These results and a comparison to GPs are shown in Figure 5. Obviously, GPs showed no significant difference in the prediction error. However, LSTMs could exploit the temporal correlations in the data.

These results suggest that the temporal correlations in our real robot dataset can be exploited to improve the model quality.

B. Dynamics model learning for torque control

The predicted joint torques are highly nonlinear signals which are shown for two arm reaching trials in Figure 6. Both motions have a time horizon of about 4.1 seconds, which is denoted by the vertical line.

The first panel in Figure 6 illustrates the mean squared error of the predictions of GPs and LSTMs. The remaining panels show predictions of GPs and LSTMs of all five joint torques in the KUKA robot. From these predictions one can see that already a small difference in the mean squared error, i.e., GPs: 0.014 ± 0.0006 and LSTMs: 0.005 ± 0.002, results in large deviations from the true joint torque signals.

For computing these results both the GPs and the LSTM networks were trained with 1000 samples. In Figure 7, we illustrate the effect of the sample size on the prediction performance. On small datasets with less than 1000 samples GPs achieve better results. However, with increasing samples LSTM networks outperform GPs which is shown in Table II. Note that the mean squared error shown in Figure 7 is in log scale and that no significant performance improvement or drop can be observed with more than $10,000$ samples in LSTMs. However, for GPs with more than 1000 samples the performance dropped as optimizing the length-scale parameters of the GPs became computationally intractable.

C. On the effect of dataset size on the computational time

Theoretically LSTM networks scale linearly with the number of samples in $O(n)$, while GPs scale exponentially with $O(n^3)$. To test these theoretical time complexities, we conducted a numerical evaluation using the real robot data.

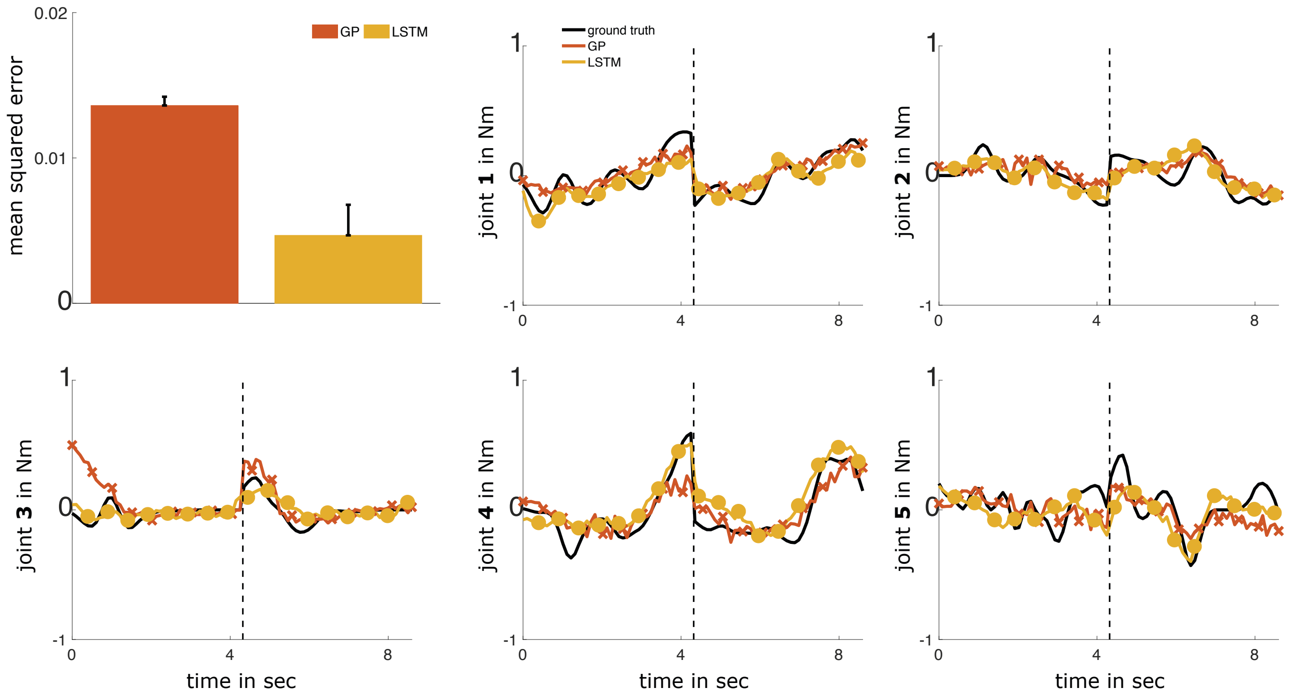

Fig. 6. Joint torque prediction results of GPs and LSTMs on a **real robot test dataset**. The first panel shows the mean squared error of both approaches for 1000 training samples. The remaining panels show the predicted joint torques of all five joints of the KUKA LWR arm. The dark line denotes the ground truth target values and the vertical line marks the end of one trial of robot arm reaching. Note that a precise match cannot be achieved because of the different flask fill levels used for training (200 & 400 ml) and for testing (300 ml) the models.

TABLE II

MEAN SQUARED ERROR IN 10^{-3} NM ON THE REAL ROBOT DATASET WITH $1,000$, $10,000$, $50,000$ AND $100,000$ TRAINING SAMPLES. THE $\pm$ SYMBOL DENOTES THE STANDARD DEVIATION OVER 10 EXPERIMENTS.

	1k	**10k**	**50k**	**100k**
GPs	13.6 ± 0.6	49.9 ± 22.0	—	—
LSTMs	12.6 ± 1.4	22.9 ± 3.7	24.5 ± 4.4	21.8 ± 3.9

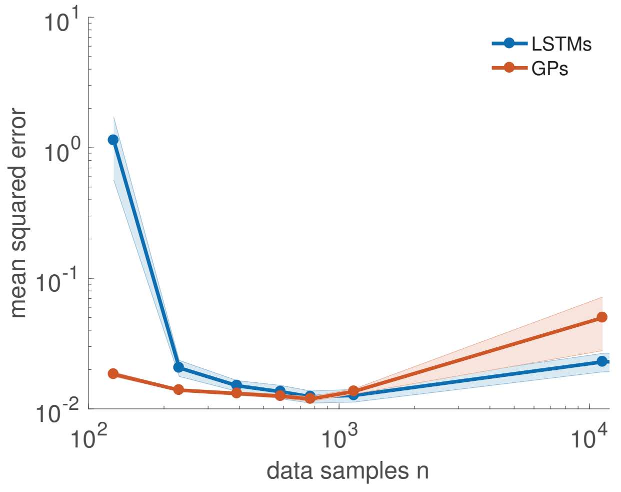

Fig. 7. Comparison of the mean squared error of GPs and LSTMs for an increasing number of data samples. The solid lines show the mean values and the shaded areas denote the standard deviation over 10 runs. Note that for 1000 samples a more detailed comparison is shown in Fig. 6.

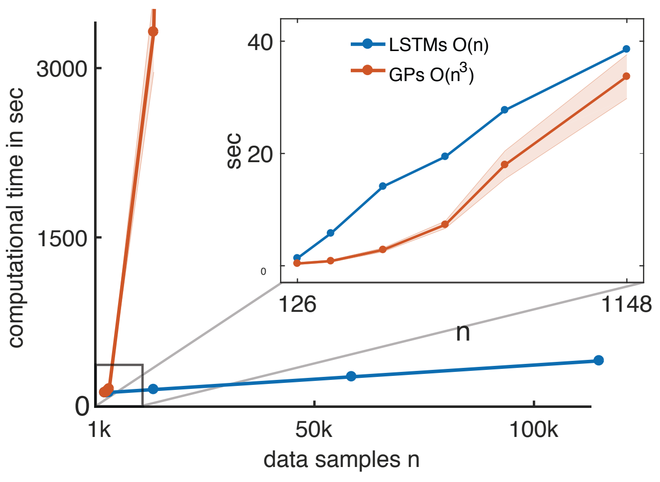

Fig. 8. Comparison of the computational time of GPs and LSTMs for an increasing number of data samples. The solid lines show the mean values and the shaded areas denote the standard deviation. The inlay shows results for less than 1148 samples.

TABLE III

COMPUTATIONAL TIMES IN SECONDS FOR AN INCREASING NUMBER OF SAMPLES.

	1k	**10k**	**50k**	**100k**
GPs	33.7 ± 3.9	3196 ± 355	$9\ 10^4$ (25h)	—
LSTMs	2.9 ± 0.2	28.1 ± 0.6	141.1 ± 1.2	282.4 ± 2.8

The results of this comparison of GPs and LSTMs is shown in Figure 8. For small datasets with up to 1000 samples both approaches need similar resources, however with more than 1000 samples GPs could not produce predictions in a reasonable amount of time, e.g., for $10,000$ samples, training the GPs took more than 24 hours as shown in Table III. LSTM networks in contrast could be trained in 28.1 seconds and for learning from $112,761$ samples, training took 282.4 seconds. Note that for all evaluations a standard PC with 8 cores operating at 3.4 GHz and 16 GB ram was used.

IV. CONCLUSIONS

We demonstrated in this work that complex inverse dynamics models of a real robot arm can be learned in linear time from large datasets with more than $100,000$ samples. The prediction error of the used Long-short-term-memory (LSTM) networks decreased exponentially with the number of samples and even for small datasets we outperformed the state of the art approach — Gaussian Processes (GPs).

A notable difference of the neural network predictions compared to the ones of GPs is the *nonexistence* of a variance or prediction uncertainty. In various approaches however, it was shown that such variance estimates can be used to improve a movement representation or to adjust the controller stiffness, see for example [17], [5], [11], [23]. Given the recent investigations aiming at learning a predictive variance in deep networks, e.g., by exploiting the stochasticity of the outputs when using *dropout* [7] or stochastic neurons [22], the LSTM model could be extended which is part of future work.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreements No 713010 (GOAL-Robots) and No 640554 (SKILLS4ROBOTS). Further financial support was received from the Bosch-Forschungsstiftung. The authors would like to also thank Alexandros Paraschos for his assistance in collecting data [18] used in this manuscript.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] M. Abromowitz and I. A. Stegun. Handbook of mathematical functions. *NBS (now NIST)*, 1965.

[3] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. In *Lazy learning*, pages 75–113. Springer, 1997.

[4] R. Calandra, S. Ivaldi, M. P. Deisenroth, E. Rueckert, and J. Peters. Learning inverse dynamics models with contacts. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3186–3191, 2015.

[5] S. Calinon, I. Sardellitti, and D. G. Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 249–254. IEEE, 2010.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[8] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[10] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[11] D. Lee and C. Ott. Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, 31(2):115–131, 2011.

[12] B. Matérn. *Spatial variation*, volume 36. Springer New York, 1960.

[13] F. Meier and S. Schaal. Drifting gaussian process regression for inverse dynamics learning. In *In IROS 2015 Workshop on Machine Learning in Planning and Control of Robot Motion*, 2015.

[14] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[15] D. Nguyen-Tuong, J. R. Peters, and M. Seeger. Local gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems*, pages 1193–1200, 2009.

[16] F. J. Ordóñez and D. Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[17] A. Paraschos, R. Lioutikov, J. Peters, and G. Neumann. Probabilistic prioritization of movement primitives. 2017.

[18] A. Paraschos, J. Peters, and E. Rueckert. Dataset to learning inverse dynamics models in o(n) time with lstm networks, 2017. "http://robotics.com.de/ds/DATASETHumanoids2017Rueckert.zip".

[19] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[20] C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in neural information processing systems*, pages 881–888, 2002.

[21] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[22] E. Rueckert, D. Kappel, D. Tanneberg, D. Pecevski, and J. Peters. Recurrent spiking networks solve planning tasks. *Nature Publishing Group: Scientific Reports*, 6(21142), 2016.

[23] E. Rueckert, J. Mundo, A. Paraschos, J. Peters, and G. Neumann. Extracting low-dimensional control variables for movement primitives. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015.

[24] E. Rueckert, G. Neumann, M. Toussaint, and W. Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Comp. Neuroscience*, 6(97), 2012.

[25] S. Schaal and C. G. Atkeson. Learning control in robotics. *IEEE Robotics & Automation Magazine*, 17(2):20–29, 2010.

[26] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[27] O. Sigaud, C. Salaün, and V. Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, 2011.

[28] V. Tresp. Mixtures of gaussian processes. In *Advances in neural information processing systems*, pages 654–660, 2001.

[29] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the International Conference on Machine Learning (ICML 2000)*, volume 1, pages 288–293, 2000.

[30] C. K. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in neural information processing systems*, pages 514–520, 1996.

[31] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Backpropagation: Theory, architectures, and applications*, 1:433–486, 1995.