Gershgorin Loss Stabilizes the Recurrent Neural Network Compartment of an End-to-end Robot Learning Scheme

Mathias Lechner[1*], Ramin Hasani[2*], Daniela Rus[3], and Radu Grosu[2]

***Abstract*— Traditional robotic control suits require profound task-specific knowledge for designing, building and testing control software. The rise of Deep Learning has enabled end-to-end solutions to be learned entirely from data, requiring minimal knowledge about the application area. We design a learning scheme to train end-to-end linear dynamical systems (LDS)s by gradient descent in imitation learning robotic domains. We introduce a new regularization loss component together with a learning algorithm that improves the stability of the learned autonomous system, by forcing the eigenvalues of the internal state updates of an LDS to be negative reals. We evaluate our approach on a series of real-life and simulated robotic experiments, in comparison to linear and nonlinear Recurrent Neural Network (RNN) architectures. Our results show that our stabilizing method significantly improves test performance of LDS, enabling such linear models to match the performance of contemporary nonlinear RNN architectures. A video of the obstacle avoidance performance of our method on a mobile robot, in unseen environments, compared to other methods can be viewed at `https://youtu.be/mhEsCoNao5E`.**

I. Introduction

In order to process spatiotemporal data, a memory mechanism is required to take into account past-time dependencies into an agent's current decision. Recurrent neural networks (RNN)s and their variants (e.g. long short-term memory (LSTM) [1]) are of nonlinear sequential models that have shown great success in modeling sequences in a broad range of application domains, specifically in robotics learning tasks such as maximum likelihood estimation of dynamical systems [2], continuous control [3], [4], [5] and simulation to real-world end-to-end reinforcement learning [6], [7]. Despite their empirically represented effectiveness, their nonlinear dynamical properties are yet to be discovered. It has been recently shown that linear dynamical systems can be learned through gradient descent with polynomial sample complexity [8], in contrast to the prior works [9], which suggested an exponential complexity. To take a step forward towards the understanding of RNNs in continuous-time spaces, in the present study, we remove the nonlinearity of an RNN's internal state and express its dynamics by the state transition of a time-invariant linear dynamical system (LDS).

The Backpropagation-Through-Time (BPTT) algorithm [10] used for training RNNs does not scale well with increasing sequence length, due to the sequential workload that cannot be parallelized [11]. In order to utilize parallel computing hardware effectively, training sequences are usually split into fixed-length sub-sequences. Though this technique significantly improves training efficiency, it creates a training-testing discrepancy when learned RNNs are deployed on arbitrary-length environments. An example of such issue observed in practice is the explosion of the RNN's internal memory, caused by test episodes that are much longer than the training sequences. Contemporary nonlinear RNN architectures tackle this problem by contracting the RNN state, e.g. the LSTM implementation of TensorFlow has an optional clipping operation applied to the memory variables[1].

In the context of LDS, such nonlinear state-contractors are inapplicable, as they would interfere with the linearity of the system. In this work, we illustrate that careful stability considerations have to be taken into account when

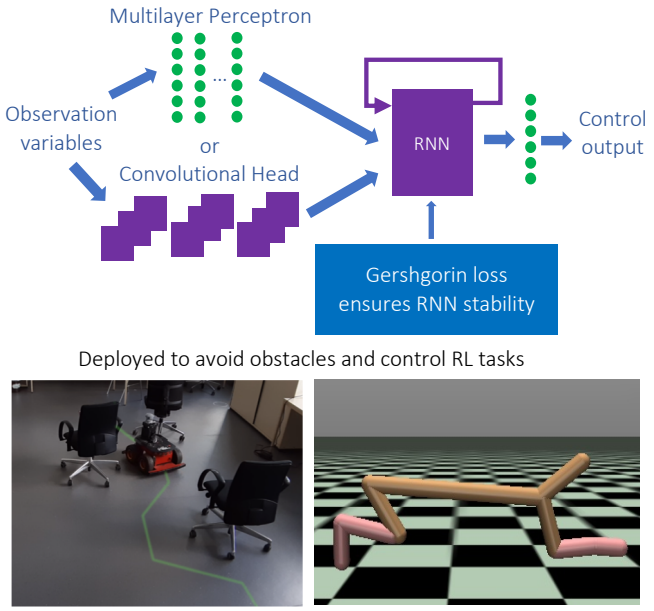

Fig. 1: Top: Network architecture supplied with a linear stable (through our Gershgorian regularization method) RNN compartment to be trained end-to-end by gradient descent. A video of the performance of our algorithm compared to others in the obstacle avoidance experiment can be viewed at `https://youtu.be/mhEsCoNao5E` and on the Half-Cheetah experiment at `https://youtu.be/MIUGkGPxCdY`.

*Equal Contributions
[1]Institute of Science and Technology Austria (IST Austria)
[2]Technische Universität Wien (TU Wien), 1040 Vienna, Austria
[3]Massachusetts Institute of Technology (MIT), USA

[1]`https://www.tensorflow.org/api_docs/python/tf/nn/rnn_cell/LSTMCell`

an LDS model is learned over fixed-length sequences. For instance, we show that an LDS trained to make a mobile robot avoid obstacles from short episodes of imitation can easily go unstable. To avoid such divergent behavior, we equip a gradient descent-based learning platform with a new regularization loss component-driven from the *Gershgorin circle theorm* [12]. We prove that the resulting loss function ensures the stability of the autonomous LDS by pushing all its eigenvalues to be negative real numbers.

Our learning scheme enables end-to-end training of stacked convolutional neural networks (CNN)s or multi-layer perceptron (MLP)s kernels, together with the LDS, in simulated and real-life robotic control environments. Our experimental evaluations demonstrate that an LDS learned by our method can be effectively deployed on sequences of arbitrary length. Additionally, we empirically show that our LDS can match the performance of modern nonlinear sequential models such as Continuous-time RNN and LSTM.

The main contributions of this paper can be summarized as follows:

1) Introducing a novel regularization loss for learning linear dynamical systems, that guarantees the stability of the autonomous LDS.
2) Development of a new end-to-end learning algorithm to train stable linear dynamical systems stacked with additional Deep Learning layers.
3) Real-life robotics and simulated experiments demonstrating that our approach improves stability, and enables LDS to match the performance of modern nonlinear RNN architectures.

II. Related Works

Learning from Demonstration is a method in which robots learn and generalize well from a set of observations of represented tasks [13], [14]. Let $f(x) : \mathbb{R}^N \rightarrow \mathbb{R}^N$, be a mapping function from the observations to actions, and $\dot{x} = f(x)$ with $x \in \mathbb{R}^N$ be the state variable of the robotic system, $f(x)$ can be estimated from data formulated as a regression problem [15], [16]. Several machine learning methods have been introduced for the approximation of $f(x)$ such as Gaussian Mixture Regression [17], Gaussian Processes [18], Bayesian Non-Parametric Mixture Models [15], and Neural Networks [19]. While these approaches, employing Learning from Demonstrations, often learn only a single component of the entire control stack, i.e. usually the trajectory controller [20], [21], [22], the successes of Deep Learning has made it possible to learn the complete control suit in an end-to-end fashion [4], [23], [24], [8]. Here, we extend the existing end-to-end imitation learning scheme to the context of linear dynamical systems.

Learning stable Dynamical Systems - Learning a system that is provable stable is a desired property for most control environments. Naive approaches for learning a stable system formulate the learning task as a *Constrained Optimization* problem, where a stability condition is added as a constraint to the main objective. For instance, [20], [21] proposed to stabilize a trajectory controller realized by a Gaussian mixture regression, by introducing a stability condition based on a Lyapunov function. The concept of Lyapunov functions was also employed by [19], [25] to learn stable dynamics by Neural Network models. [26], [22] derived a stability condition from contraction theory, which is then used as a constraint.

In order for such approaches to work, the stability constraint must imply the stability of the system. However, this implication does not necessarily hold in the other direction; in order to employ a gradient-based optimization, the stability condition must be continuous and differentiable and is therefore often a bound to the true stability condition of the system. Our learning algorithm distinguishes between the differentiable stability condition and the true stability property of the autonomous LDS. The constraint is only optimized when the system is non-stable.

An approach closely related to ours was introduced by [27], which also leverages linear system theory to learn a stable system. One key distinction to our approach is, that [27] employs fixed nonlinear *basis functions* to enhance the expressiveness of the learned controller, whereas we stack our LDS with more flexible Deep Learning layers such as convolutional and fully-connected layers.

Is the closed control-loop system stable? Note that, our approach does not mathematically proof the stability of the closed control-loop system. We aim to improve the stability of a deployed LDS that is learned in an end-to-end fashion on fixed-length sequences. Rigorously proofing the stability of any closed control-loop system requires detailed knowledge and strong assumptions about the dynamics of the control environment. This prerequisite is antagonistic to the idea of end-to-end learning, which demands only minimal prior knowledge and assumptions.

Recurrent Neural Networks for Modeling Dynamical Systems - RNNs presented great performance in robotic control environments; examples include the maximum likelihood estimation of a dynamical system [2], [28], continuous control [3], [4] and simulation to real-world reinforcement learning [6]. Fully-connected LSTM networks have been used for learning of unsupervised video representations [29]. [30] proposed the combination of data-driven and model-based learning to predict the behavior of dynamical systems; Authors stacked a convolutional LSTM [31], an architecture which performs the convolution operation as the input-to-state transitions instead of dense connections, to a Cellular Neural Networks [32], an algorithm for solving partial differential equations (PDE) computationally efficient, and achieved state-of-the-art performance on two dynamical systems test-beds. Here, we propose a novel learning scheme to learn structures by CNNs or MLPs and to learn the temporal dependencies by continuous-time dynamical systems (RNNs) in an end-to-end fashion.

III. Learning Stable Linear Dynamical Systems

The hidden state transition $x(t)$, and the output dynamics $y(t)$ of a time-invariant *linear dynamical system* (LDS) can

be determined as follows:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \\ x(0) &= x_0, \end{aligned} \tag{1}$$

where $u(t)$ is the input, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times k}$ and $C \in \mathbb{R}^{m\times n}$ are linear transformation matrices which are denoted as the parameters of the LDS to be learned.

Properties of LDS systems of the form Eq. (1), specifically their stability, have been studied for decades within the Control Theory community [8]. For instance, the closed form solution of the autonomous sub-system $\dot{x}(t) = Ax(t)$ is a linear combination of complex exponential functions [33]. The stability of such autonomous sub-system, i.e. $\lim_{x\to\infty} x(t) < \infty$, can be identified by computing the eigenvalues of A. The system is considered stable if all eigenvalues of A are negative [33].

In this section, we aim at modeling the input and output of a recurrent neural network with standard nonlinear functions, but simplifying the state-transition representation (sequential dependencies as a result of the recurrent connections), as a linear dynamical system. The motivation is to discover unknown dynamical system properties of recurrent neural networks which might be identifiable by the simplification.

In order to efficiently apply gradient-descent to the ordinary differential equation (ODE) solution, we discretize the ODE using Euler's explicit method [34]:

$$x(t+\Delta) = x(t) + \Delta\dot{x}(t), \tag{2}$$

which essentially translates the LDS into a recurrent neural network (RNN).

Many RNN architecture suffer from the *vanishing and exploding gradient* problems [35], [36], [37]. Learning long-term dependencies becomes challenging with the vanishing gradient effect. In this case, the RNN can solely learn to correlate events that happen close in time. On the other hand, the explosion of the gradient results in an unstable learning process. Truncated back-propagation through time [38], [37] is a commonly used method to ease the exploding gradient problem. Despite the choice of the ODE solver, in case of the Euler discretization of the LDS, the described challenges of the gradient computations become a stability issue of the linear dynamical system. The learned dynamical system, post-training, operates in a continuous loop, therefore assuring its stability is vital. Accordingly, we introduce the *Gershgorin circle loss* as follows [12]:

$$\mathscr{L}_{gc}(A) := \sum_{i=1}^{n} \max\{0, A_{i,i} + \sum_{j\neq i} |A_{i,j}| + \varepsilon\}, \qquad \varepsilon > 0 \tag{3}$$

Where A is the state transition matrix of the LDS. In the following, we prove that by minimizing the Gershgorin circle loss, eigenvalues of the matrix A are forced to be negative real numbers, Therefore, if the Gershgorin circle loss is zero, all eigenvalues must have a negative real part.

Lemma 1 (Gershgorin circle theorem)**.** *Every eigenvalue of A lies within at least one of the Gershgorin discs $D(A_{ii}, R_i)$, with $R_i = \sum_{j\neq i} |A_{i,j}|$ and $D(a,b) := \{x \in \mathbb{C} \mid |x-a| \leq b\}$*

Proof: See [39].

Theorem 1 (Gershgorin circle loss ensures stability)**.** *Let $A \in \mathbb{R}^{n\times n}$ and*

$$\mathscr{L}_{gc}(A) := \sum_{i=1}^{n} \max\{0, A_{i,i} + R_i + \varepsilon\}, \tag{4}$$

with $R_i = \sum_{j\neq i} |A_{i,j}|$ and $\varepsilon > 0$. If $\mathscr{L}_{gc}(A) \leq 0$ then all eigenvalues of A have negative real part.

Proof: Given the definition of D, for every $x \in D(A_{ii}, R_i) \subseteq \mathbb{C}$ it holds that $Re(x) < A_{i,i} + R_i + \varepsilon$ for arbitrary $i = 1, \ldots n$. We assumed $\mathscr{L}_{gc}(A) \leq 0$, ergo $A_{i,i} + R_i + \varepsilon \leq 0$ for every $i = 1, \ldots n$. Using the triangular inequality, it follows that for every $x \in D(A_{ii}, R_i)$ $Re(x) \leq -\varepsilon$ for every Gershgorin disc $D(A_{ii}, R_i)$. According to Lemma 1, every eigenvalue of A must lie within at least one Gershgorin disc; therefore, every eigenvalue must have a negative real part.

Note that a learned linear dynamical system can be stable even if the Gershgorin circle loss is greater than zero. Consequently, it does not make sense to use the Gershgorin circle loss as regularizer during every optimization step. Therefore, we introduce Algorithm 1, which checks if at least one eigenvalue has a non-negative real part and only then perform a gradient update step with respect to the Gershgorin circle loss. Algorithm 1 declares a stable learning process for LDS equipped with the Gershgorin loss.

Algorithm 1 Training algorithm for linear dynamical systems where all eigenvalues are guaranteed to be negative

Input Maximum number of training epochs N, Training loss $\mathscr{L}_{train}$, Validation loss $\mathscr{L}_{valid}$, Gershgorin circle loss $\mathscr{L}_{gc}$, parameter θ with $A \in \theta$, learning rate α
while at least one eigenvalue of A has non-negative real part **do**
  $\theta \leftarrow \theta - \alpha \frac{\partial \mathscr{L}_{gc}(A)}{\partial A}$
end while
$\theta_{best} \leftarrow \theta$
$v_{best} \leftarrow \mathscr{L}_{valid}(\theta)$
for $1 \ldots N$ **do**
  $\theta \leftarrow \theta - \alpha \frac{\partial \mathscr{L}_{train}(\theta)}{\partial \theta}$
  while at least one eigenvalue of A has positive real part **do**
    $\theta \leftarrow \theta - \alpha \frac{\partial \mathscr{L}_{gc}(A)}{\partial A}$
  end while
  $v \leftarrow \mathscr{L}_{valid}(\theta)$
  if $v < v_{best}$ **then**
    $v_{best} \leftarrow v$
    $\theta_{best} \leftarrow \theta$
  end if
end for
return θ_{best}

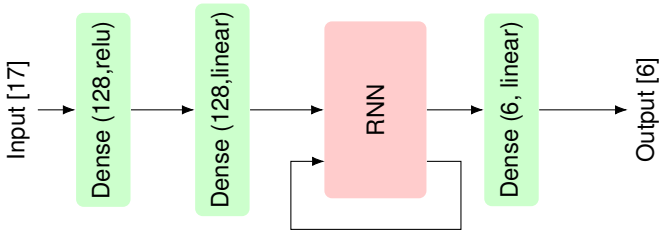

Fig. 2: Network architecture used for the "HalfCheetah-v2" imitation learning task, RNN state size is 32 for all models

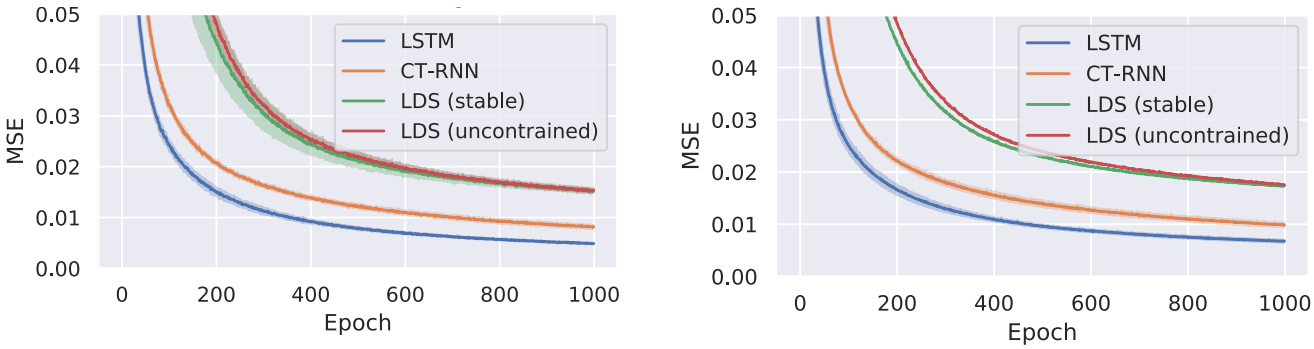

Fig. 3: Learning curves on the "HalfCheetah-v2" imitation learning task. Left: Training loss. Right: validation loss. Mean (thick) and standard deviation (semi-transparent), N=5.

A. *LDS as a Continuous-Time Recurrent Neural Networks*

The representation of the described discretization of ODEs as recurrent neural networks highly resembles the state-space dynamics of continuous-time recurrent neural networks (CT-RNN)s [40], [41], [42]. More specifically, a CT-RNN can be formulated as the Euler simulation of an ODE of the form

$$\dot{x} = f_\theta(x,u) - x \tag{5}$$

where $f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^n$ is a neural net parametrized by θ. The major difference between the state-space variable dynamics of a linear dynamical system and a CT-RNN's state representation is in the nonlinearity introduced by the neural network f in equation (5). This difference makes CT-RNNs arguably more expressive, but simultaneously increases the complexity of the system and correspondingly reduces provability of a system's characteristics such as stability and the closed-form solution.

Note that the transition state-stability of any recurrent model can be feasibly enforced by clipping the RNN state to a bounded range (e.g. between -10 and 10), after every update. However, this approach introduces nonlinearity into the feedback operation, which is non-permissible for linear dynamical systems. By proposing the Gershgorin circle regularization, we aim at exploring the performance of the stable LDS compared to that of nonlinear RNN architectures. In particular, in the next section, we perform robotic control experiments to observe how LDS with and without Gershgorin circle regularization compare to LSTM and CT-RNNs with the clipped state.

IV. Experimental Results

In this section, we evaluate the capabilities of our proposed learning approach on two imitation learning task, as illustrated in Figure 1. The first task is to clone the behavior of an existing recurrent neural network controller for the "HalfCheetah-v2" OpenAI Gym [43] environment.

In the second experiment, we perform real-world learning from demonstrations of a small set of obstacle avoidance scenarios performed by a mobile robot (Pioneer 3-AT). The objective is to detect obstacles using LIDAR input data and execute avoidance motions. To compare performance with existing RNN models, we carry out the same experiments with Long short-term memory networks (LSTM), standard continuous-time RNN, an unconstrained LDS and our proposed LDS with Gershgorin circle loss regularization. The code and training data are provided in the supplementary materials.

A. *OpenAI Gym HalfCheetah-v2*

The objective of this task is to imitate the behavior of a reference RNN policy trained on the "HalfCheetah-v2" OpenAI Gym [43] environment. The reference policy achieves

TABLE I: Loss on the imitation learning task and final performance on the gym "HalfCheetah-v2" environment. Mean and standard deviation, N=10

Model	Train	Validation	Return
LSTM	0.005	0.007	2706.32±82.97
CT-RNN	0.008	0.010	2473.15±242.67
LDS (ours)	0.015	0.017	**2721.03±94.51**
LDS (unconstrained)	0.015	0.017	2636.52±201.14

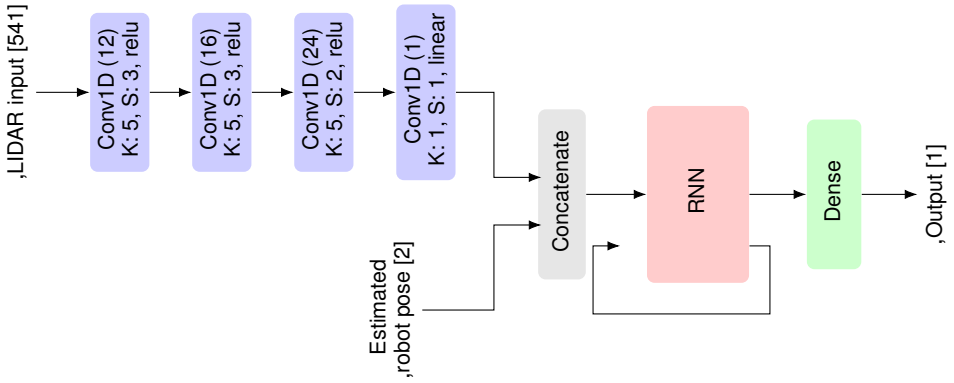

Fig. 4: Network architecture used for the robot navigation task, K indicates kernel size, S indicates size of stride, RNN state size is 32 for all models

a return value of 2635.11 with a standard deviation of 416.28 estimated in 30 episodes. The training set consists of recordings of 25 full episodes and the validation set of 5 episodes of length 1000 time step each, i.e. 25000 time-steps training and 5000 time steps validation samples.

Each tested model consists of an RNN preceded by a fully-connected ReLU and a linear layer, as illustrated in Figure 2 a). We train all models with truncated back-propagation through time [38] by splitting each sequence into multiple sub-sequences of length 32. The continuous-time models, i.e. CT-RNN and both LDS, are simulated using Euler's explicit method in equation (2) with $\Delta = 0.166$ timesteps. As the final evaluation metric, we execute the imitated policy on the gym environment ten times and report the mean return. For each model, we repeat this experiment 5 times and report the mean and standard deviation in Table I.

1) Discussions on the performance of the HalfCeetah experiments: When we evaluate the networks trained over the sequences of length 32 time-steps and impose no environmental feedback during training, the performance of the LDS with and without Gershgorin circle regularizer is almost identical as shown in the learning curves in Figure 3. The learning curves depict the superior performance of the nonlinear models (LSTM and the standard CT-RNN) over the linearized models. This result is intuitive, and roots for the existing trade-off between a model's expressivity and having stability guarantees.

When executing the model in a closed feedback loop on the gym environment with sequences up to full episode length, i.e. 1000 time steps, LDS supplied with the Gershgorin circle regularizer significantly outperforms the LDS without the regularizer as depicted in Table I. The regularized LDS surpasses the performance of the CT-RNN and matches that of LSTM policy (see Table I).

B. Obstacle detection and avoidance

The objective of this task is to navigate a Pioneer 3-AT mobile robot safely through a course of obstacles. In particular, the agent is provided with the input stream from a 270 degree 2D LiDAR scanner that is mounted on the rover. The robot estimates pose (angle and lateral displacement from initial pose) as a time series sampled at 10Hz. The agent is expected to output a decision in the form of a target angular velocity, which should maneuver the robot safely around the obstacles.

We collected 20 training, five validation, and five test traces by navigating the robot around the obstacles via joystick teleoperation. Each evaluated model consists of an RNN preceded by a set of convolutional layers, as illustrated in Figure 4 b). We train all models with end-to-end truncated back-propagation through time [38] by splitting each training and validation sequence into multiple sub-sequences of length 32, i.e. corresponding to 3.2 real-time seconds. The continuous-time models, i.e. CT-RNN and both LDS, are simulated using Euler's explicit method in equation (2) with $\Delta = 16.6$ milliseconds.

To evaluate the models, we perform two-fold testing, I: a simulated test loop and II: deploying the agent on the real robot. Our simulated test loop evaluates the model's mean-square error on the five full-length test sequences. This process can be done computationally fast; therefore, we repeat the simulated experiment for each model 10 times with different weight initialization for the networks. Though this way, we can draw a significant number of samples, no feedback from the environment would affect the model's ability to maneuver the robot safely around the obstacles.

Consequently, we deploy the best performing model out of the ten initializations on the real robot to navigate five real-world obstacle course scenarios. As for the performance metric, we count the number of avoided obstacles in each of these five real-world scenarios.

These real-world scenarios differ from the training, validation, and test-set as they include more obstacles with different shapes (i.e. the training, validation, and test-set contain recordings of only up to 2 obstacles during an episode, whereas in the real-life testing scenarios there are more obstacles configured in various settings).

1) Discussions on the performance of the Obstacle avoidance experiments: Similar to the HalfCheetah-v2 experiment, the performance of the LDS with and without Gersh-

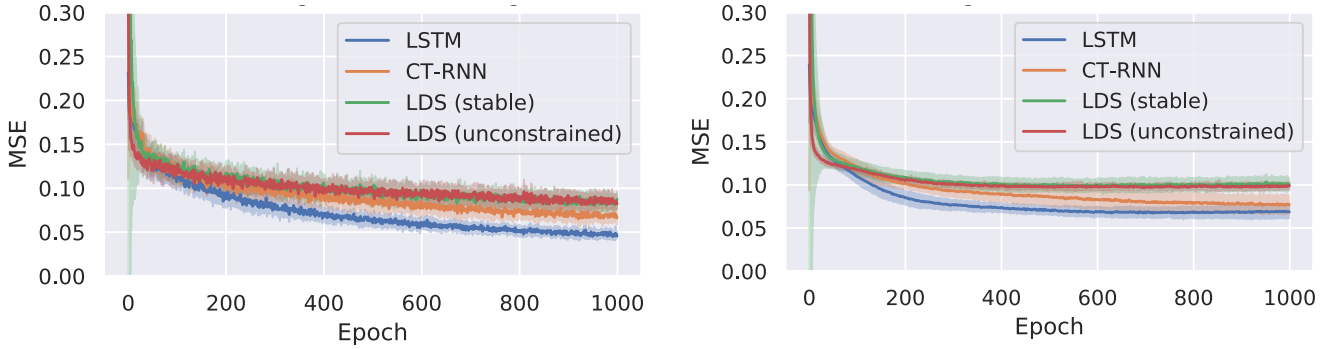

Fig. 5: Learning curves on the robot navigation task imitation learning task. Left: Training loss. Right: validation loss.

TABLE II: Training results on the robotic imitation task. N=10.

RNN Scheme	Model	Train loss	Validation loss	Test loss
Nonlinear	LSTM	0.057 ± 0.007	0.065 ± 0.006	0.125 ± 0.014
	CT-RNN	0.070 ± 0.008	0.074 ± 0.010	**0.119 ± 0.010**
Linear	**LDS (ours)**	0.089 ± 0.007	0.095 ± 0.007	**0.388 ± 0.431**
	LDS (unconstrained)	0.090 ± 0.010	0.094 ± 0.003	3.005e6 ± 5.958e6

gorin circle regularization are almost identical on the training and validation sub-sequences as shown in the learning curves in Figure 5 and Table II.

When testing the models on longer sequences the internal state of the unconstrained LDS explodes at some point resulting in an unstable behavior and extremely high mean-squared error. Note that all of the trained unconstrained LDS for the HalfCheetah-v2 as well as the obstacle avoidance task had at least one eigenvalue with a positive real part. However, no catastrophic state explosion of the unconstrained LDS occurred in the HalfCheetah-v2 task, suggesting that model stability also depends on the data-set dependent properties.

As discussed previously about the trade-off between a model's expressiveness vs. its auditability, in terms of validation and simulated test performance, both linear RNNs are outperformed by the nonlinear RNNs, as shown in Table II.

However, when deploying the models in a feedback loop on the real robot, the gap in the performance decreases and even in some scenarios the regularized LDS outperforms the other models as it can be viewed at `https://youtu.be/mhEsCoNao5E`, and is shown in Table III, Scenario 5. The LDS with Gershgorin circle regularization matches the performance of the LSTM model but marginally falls behind the CT-RNN. As expected, the unconstrained LDS also suffers from instability on all tested real scenarios. A video recording of all experiments performed on the robot can be viewed at `https://youtu.be/mhEsCoNao5E`.

TABLE III: Results of our live experiment consisting of 5 test scenarios which the agent has to navigate. The number of obstacles passed by each agent. A video recording of all experiments performed on the robot can be viewed at `https://youtu.be/mhEsCoNao5E`.

Scenario	LSTM	CT-RNN	**LDS (ours)**	LDS (uncontrained)
#1 (2 obstacles)	2	2	2	0
#2 (3 obstacles)	2	3	2	0
#3 (4 obstacles)	4	4	4	0
#4 (4 obstacles)	4	4	3	0
#5 (4 obstacles)	3	3	**4**	0
Total (17 obstacles)	15	**16**	15	0

V. Conclusion

We introduced a learning scheme for training stable linear dynamical systems (LDS)s by gradient descent. We leveraged the Gershgorin circle theorem to define a regularization loss, which ensures the stability of the learned autonomous LDS.

This work builds on the recently proposed idea by [8], to study the learning characteristics of a linear dynamical system to take a step forward towards better understanding the dynamics of recurrent neural networks.

We showed on two real-life and simulated imitation learning robotic problems that our approach enables LDS to generalize well from fixed-length training sequences to arbitrary length sequences which occur naturally when deploying the agent in a continuous control loop.

We illustrated that our approach could be stacked with complex architectures such as multilayer perceptrons and convolutional neural networks. We showed that LDS could match and even surpass the generalization ability of the existing nonlinear RNN models.

Acknowledgment

M.L. is supported in parts by the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award). R.H., and R.G. are partially supported by the Horizon-2020 ECSEL Project grant No. 783163 (iDev40), and the Austrian Research Promotion Agency (FFG), Project No. 860424. R.H. and D.R. is partially supported by the Boeing Company.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, "Learning deep neural network policies with continuous memory states," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 520–527.

[5] M. Lechner, R. Hasani, M. Zimmer, T. A. Henzinger, and R. Grosu, "Designing worm-inspired neural networks for interpretable robotic control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 87–94.

[6] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *arXiv preprint arXiv:1610.04286*, 2016.

[7] R. M. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Re-purposing compact neuronal circuit policies to govern reinforcement learning tasks," *arXiv preprint arXiv:1809.04423*, 2018.

[8] M. Hardt, T. Ma, and B. Recht, "Gradient descent learns linear dynamical systems," *Journal of Machine Learning Research*, vol. 19, no. 29, pp. 1–44, 2018.

[9] M. Vidyasagar and R. L. Karandikar, "A learning theory approach to system identification and stochastic adaptive control," in *Probabilistic and randomized methods for design under uncertainty*. Springer, 2006, pp. 265–302.

[10] P. J. Werbos *et al.*, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[11] D. P. Rodgers, "Improvements in multiprocessor system design," *ACM SIGARCH Computer Architecture News*, vol. 13, no. 3, pp. 225–231, 1985.

[12] G. H. Golub and C. F. Van Loan, "Matrix computations, johns hopkins u," *Math. Sci., Johns Hopkins University Press, Baltimore, MD*, 1996.

[13] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

[14] G. Biggs and B. MacDonald, "A survey of robot programming systems," in *Proceedings of the Australasian conference on robotics and automation*, 2003, pp. 1–3.

[15] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," in *Conference on Robot Learning*, 2018, pp. 927–946.

[16] H. C. Ravichandar, I. Salehi, and A. P. Dani, "Learning partially contracting dynamical systems from demonstrations." in *CoRL*, 2017, pp. 369–378.

[17] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[18] A. P. Shon, K. Grochow, and R. P. Rao, "Robotic imitation from human motion capture using gaussian processes," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* IEEE, 2005, pp. 129–134.

[19] A. Lemme, K. Neumann, R. F. Reinhart, and J. J. Steil, "Neural learning of vector fields for encoding stable dynamical systems," *Neurocomputing*, vol. 141, pp. 3–14, 2014.

[20] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[21] ——, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[22] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5883–5890.

[23] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," *arXiv preprint arXiv:1709.04905*, 2017.

[24] K.-H. Zeng, W. B. Shen, D.-A. Huang, M. Sun, and J. Carlos Niebles, "Visual forecasting by imitating dynamics in natural sequences," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2999–3008.

[25] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamic systems," *arXiv preprint arXiv:1808.00924*, 2018.

[26] C. Blocher, M. Saveriano, and D. Lee, "Learning stable dynamical systems using contraction theory," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2017, pp. 124–129.

[27] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[28] R. Hasani, A. Amini, M. Lechner, F. Naser, R. Grosu, and D. Rus, "Response characterization for auditing cell dynamics in long short-term memory networks," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[29] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, 2015, pp. 843–852.

[30] Y. Long, X. She, and S. Mukhopadhyay, "Hybridnet: integrating model-based and data-driven learning to predict evolution of dynamical systems," *arXiv preprint arXiv:1806.07439*, 2018.

[31] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[32] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transactions on circuits and systems*, vol. 35, no. 10, pp. 1257–1272, 1988.

[33] G. Teschl, *Ordinary differential equations and dynamical systems*. American Mathematical Soc., 2012, vol. 140.

[34] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[35] Y. Bengio, P. Simard, P. Frasconi, *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[36] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[37] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[38] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[39] S. A. Gershgorin, "Uber die abgrenzung der eigenwerte einer matrix," . , no. 6, pp. 749–754, 1931.

[40] K.-i. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural networks*, vol. 6, no. 6, pp. 801–806, 1993.

[41] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in neural information processing systems*, 2018, pp. 6571–6583.

[42] R. M. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid time-constant recurrent neural networks as universal approximators," *arXiv preprint arXiv:1811.00321*, 2018.

[43] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.