Liquid Resistance Liquid Capacitance Networks

Mónika Farsang
Technische Universität Wien (TU Wien)
Vienna, Austria
monika.farsang@tuwien.ac.at

Sophie A. Neubauer
DatenVorsprung GmbH
Vienna, Austria
sophie@datenvorsprung.at

Radu Grosu
Technische Universität Wien (TU Wien)
Vienna, Austria
radu.grosu@tuwien.ac.at

Abstract

We introduce liquid-resistance liquid-capacitance neural networks (LRCs), a neural-ODE model which considerably improve the generalization, accuracy, and biological plausibility of electrical equivalent circuits (EECs), liquid time-constant networks (LTCs), and saturated liquid time-constant networks (STCs), respectively. We also introduce LRC units (LRCUs), as a very efficient and accurate gated RNN-model, which results from solving LRCs with an explicit Euler scheme using just one unfolding. We empirically show and formally prove that the liquid capacitance of LRCs considerably dampens the oscillations of LTCs and STCs, while at the same time dramatically increasing accuracy even for cheap solvers. We experimentally demonstrate that LRCs are a highly competitive alternative to popular neural ODEs and gated RNNs in terms of accuracy, efficiency, and interpretability, on classic time-series benchmarks and an autonomous-driving lane-keeping task.

1 Introduction

Electrical equivalent circuits (EECs) are the foremost mathematical model used in neuroscience for capturing the dynamic behavior of biological neurons [Kandel et al., 2000, Wicks et al., 1996]. EECs associate the membrane of a neuron with a capacitor, whose potential varies according to the sum of the stimulus, leaking, and synaptic currents, passing through the membrane, respectively. As a consequence, EECs are formulated as a set of capacitor ordinary differential equations (ODEs).

EECs however, received little attention in the ML community, due to their ODE nature, except for continuous-time recurrent neural networks (CT-RNNs) [Funahashi and Nakamura, 1993], which are arguably EECs with electrical synapses [Farsang et al., 2024]. This changed with the advent of Neural ODEs [Chen et al., 2018a, Rubanova et al., 2019, Dupont et al., 2019], and EECs with chemical synapses were called liquid time-constant networks (LTCs) in Lechner et al. [2019], Hasani et al. [2020], Lechner et al. [2020], as their time constant depends on both the state and the input.

The main promise of LTCs and Neural ODEs is to better capture physical reality and bridge the gap between natural sciences and ML. They were shown to be more interpretable [Lechner et al., 2020, Farsang et al., 2024] and able to recover missing data in irregularly-sampled time series [Chen et al., 2018a, Rubanova et al., 2019, Dupont et al., 2019, Brouwer et al., 2019, Kidger et al., 2020, li et al., 2023]. An important obstacle in the use of Neural ODEs and especially LTCs however, is their stiff oscillatory nature, which requires the use of expensive ODE solvers [Biloš et al., 2021]. A best solver moreover, cannot fully remove oscillations and improve accuracy, as shown in Figures 1 and 2.

1^{st} Workshop on NeuroAI @ 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

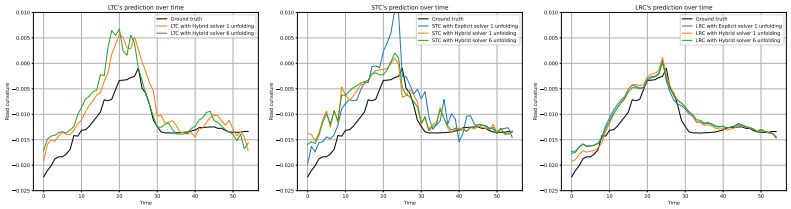
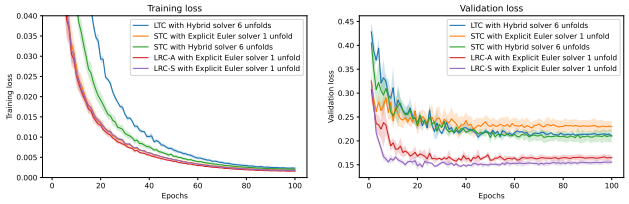


Figure 1: Dynamic behavior of the output neuron in the Lane-Keeping Task [Lechner et al., 2022]. Left: In LTCs, this neuron has a very stiff oscillatory behavior, with a relatively large validation loss. Moreover, LTCs fail to converge with an explicit Euler integration scheme of order one and just one unfolding. Middle: STCs in contrast, convergence with an explicit Euler integration scheme of order one and just one unfolding. However, the output neuron still exhibits stiff oscillatory behavior. Right: In LRCs, this neuron has a very gentle varying behavior, with a very small validation loss, even when using an explicit Euler scheme of order one and just one unfolding (no overhead).



Model	Time Per Epoch	Best Epoch	Lipschitz Constant
LTC H6	10.06 min	52 ± 06	$90 \pm 23 \times 10^{-4}$
STC E1	03.97 min	50 ± 22	$101 \pm 25 \times 10^{-4}$
STC H6	10.13 min	69 ± 10	$89 \pm 23 \times 10^{-4}$
LRC-A E1	04.05 min	43 ± 08	$47 \pm 08 \times 10^{-4}$
LRC-S E1	04.16 min	22 ± 03	$43 \pm 11 \times 10^{-4}$

Figure 2: Training loss in the first and validation loss in the second figure of LTCs, STCs, and LRCs for the Lane-Keeping Task. LTCs and STCs use a constant, unit elastance, whereas LRCs use the liquid elastance of Equations (4-5). The results for LRCs solved with an explicit Euler integration scheme with one unfolding (called LRCUs) are substantially better, both in terms of convergence speed and accuracy. Per epoch and curves, the training took around 10 minutes for 6 unfoldings per step and 4 minutes for 1 unfolding per step, which is 2.5 times faster.

In this paper, we show that the stiff dynamic behavior of LTCs results from abstraction. For simplicity, EECs and LTCs first ignore the saturation aspects of membrane's ionic channels [Alberts et al., 2008], as discussed in Farsang et al. [2024]. To take such aspects into account, they introduce saturated LTCs (called STCs). Second, and more importantly, EECs, LTCs and STCs also assume that membranal capacitance is constant, which is disproved by recent results demonstrating a nonlinear dependence on the neural state and input [Howell et al., 2015, Severin et al., 2022, Kumar et al., 2023].

Adding a liquid membrane capacitance to STCs, considerably reduces their variation wrt. inputs, while also enhancing their accuracy, as shown in Figures 1,2 and proved in Theorems A.2. We call the resulting model a liquid-resistance, liquid-capacitance neural network (LRC), as the LRC's liquid time-constant is now factored this way. Due to the gentle varying LRC behavior, an explicit Euler integration scheme with one unfolding, is often enough to obtain a very small error. We call LRCs solved with this scheme, LRC units (LRCUs). They closely resemble and competitive on the performance of gated recurrent units [Hochreiter and Schmidhuber, 1997, Cho et al., 2014, Zhou et al., 2016].

In summary, the main results of our paper are as follows:

- We introduce Liquid Resistance Liquid Capacitance Networks (LRCs), which considerably improve the generalization, accuracy, and biological plausibility of EEC, LTC, and STC models, respectively. We introduce LRC Units (LRCUs), as a very efficient and accurate gated RNN-model, which results from solving LRCs with an explicit Euler scheme of order one and one unfolding, only.

- We prove that the liquid capacitance of LRCs, leads to a more gentle varying behavior than in LTCs and STCs, while at the same time dramatically increasing their accuracy even for cheap solvers. We experimentally show that LRCs are competitive in accuracy, efficiency, and interpretability to gated RNNs and Neural ODEs on classic benchmarks, and on a complex Lane-Keeping Task.

2 Background

2.1 Liquid Time-Constant Networks (LTCs)

Definition. EECs are a simplified electrical model, defining the dynamic behavior of the membrane-potential (MP) of a post-synaptic neuron, as a function of the MP of its pre-synaptic neurons, for electrical or chemical synapses [Kandel et al., 2000, Wicks et al., 1996]. EECs with chemical synapses only, are also known in the ML community as LTCs [Lechner et al., 2019, Hasani et al., 2020, Lechner et al., 2020]. An LTC with m neurons and n inputs is defined as follows:

$$\begin{aligned} \dot{h}_i &= -f_i\, h_i + u_i\, e_{li} \\ f_i &= \textstyle\sum_{j=1}^{m+n} g_{ji}\sigma(a_{ji}y_j + b_{ji}) + g_{li} \\ u_i &= \textstyle\sum_{j=1}^{m+n} k_{ji}\sigma(a_{ji}y_j + b_{ji}) + g_{li} \end{aligned} \tag{1}$$

It states that the rate of change of the MP h_i of neuron i, is the sum of its forget current $-f_i\, h_i$ and its update current $u_i\, e_{li}$. The forget conductance f_i is the liquid time-constant reciprocal of h_i. It depends on $y = [h, x]$, the MPs h of presynaptic neurons and inputs x. Here, g_{ji} is the maximum conductance of the synaptic channels, a_{ji}, b_{ji} are parameters controlling the sigmoidal probability of these channels to be open, and g_{li}, e_{li} are the membrane's leaking (resting) conductance and potential, respectively. In the update conductance u_i, parameters $k_{ji} = g_{ji}e_{ji}/e_{li}$ take the sign of e_{ji}/e_{li}, where e_{ji} is the synaptic channel's reversal potential, that is, the MP at which there is no net ionic flow.

2.2 Saturated Liquid Time-Constant Networks (STCs)

Definition. For simplicity, LTCs ignore saturation in synaptic channels [Alberts et al., 2008]. One can take this aspect into account, by constraining the forget and signed-update conductances f_i and u_i with a sigmoid and a hyperbolic tangent, to range within [0,1] and [-1,1], respectively. Letting f_i and u_i be as above, this leads to the definition of saturated LTCs (STCs) of Farsang et al. [2024]:

$$\dot{h}_i = -\sigma(f_i)\, h_i + \tau(u_i)\, e_{li} \tag{2}$$

Accuracy. The dynamic behavior of the STC output neuron in the Lane-Keeping Task is shown in Figure 1 (middle). It is more gently varying than in LTCs, and it succeeds to converge with an explicit Euler integration with one unfolding. However, its behavior still oscillates around the ground truth, and its validation loss, while better than the one of LTCs, remains relatively large, as shown in Figure 2.

3 Liquid Resistance Liquid Capacitance Networks (LRCs)

Definition. For simplicity, LTCs and STCs assume that the membrane capacitance is constant and equal to one. This is however not the case in biological neurons. They have a nonlinear dependence on the MPs and the input [Howell et al., 2015, Severin et al., 2022, Kumar et al., 2023]. LRCs take this aspect into account, by adding a liquid elastance (reciprocal of capacitance) $\epsilon(w_i)$, in Equation (2), where the weights matrix o and the bias vector p play the same role as the ones in an artificial neuron:

$$\begin{aligned} \dot{h}_i &= \epsilon(w_i)\,(-\sigma(f_i)\, h_i + \tau(u_i)\, e_{li}) \\ w_i &= \textstyle\sum_{j=1}^{m+n} o_{ji}y_j + p_i \end{aligned} \tag{3}$$

Thus, the time constant in LRCs factors into a liquid resistance and a liquid capacitance. The second, acts as an additional control of the time constant, by increasing it when the desired variation of the function is gentle, and leaving it unchanged when the desired behavior is stiff. For convenience, we provide the definition of both an asymmetric and a symmetric form of elastance in Equations (4-5), respectively. Here, the parameters vector k of the symmetric version is non-negative in each of its components:

$$(A) \quad \epsilon(w_i) = \sigma(w_i) \tag{4}$$

$$(S) \quad \epsilon(w_i) = \sigma(w_i + k_i) - \sigma(w_i - k_i) \tag{5}$$

The shape of the two elastances is shown in Figure 3. The first is asymmetric with respect to y, while the second is symmetric with respect to y. They both scale the elastance to range within [0,1], thus acting to possibly decrease the variation of h. The linear output layer of the LRC may compensate this, by possibly having values larger than one. The architecture of LRCs is given in Figure 6 (left).

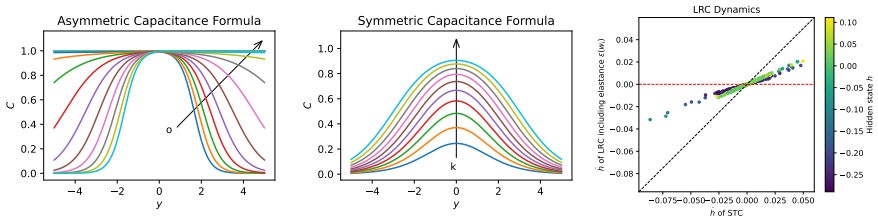


Figure 3: Left, Middle: The asymmetric and symmetric shape of the membrane's elastance $\epsilon(w_i)$ proposed for LRCs, using Formulas (4) and (5), respectively. Right: Experiment showing that the LRC's elastance dampens the dynamic behavior of STCs in an input and state-dependent way. It can fully reduce it to zero, or keep it as it is, indicated by the red and black dashed lines, respectively.

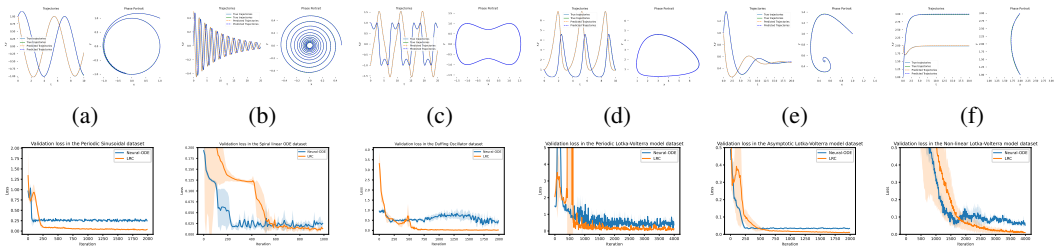


Figure 4: Top: Neural-ODE experiments with (x, y) state variables with plotted trajectory and phase diagram, solved by LRCs. (a) Periodic sinusoidal system, (b) Spiral towards the origin, (c) Non-linear Duffing oscillatory system, and Lotka-Volterra models with (d) Periodic and (e) Asymptotically stable dynamics and (f) Non-linear version. Bottom: LRCs converged to a smaller loss, indicating their ability to fit these dynamics better than the vanilla Neural-ODE.

4 Experimental Results

In the following subsections, we present our experimental results for LRCs, on different topics of interest in the ML community. First, we investigate the accuracy and speed of LRCs in solving classic Neural-ODE tasks. Second we explore the interpretability of the learned LRCUs (and thus LRCs) for the Lane-Keeping Task. More experiments can be found in the Appendix A.4.

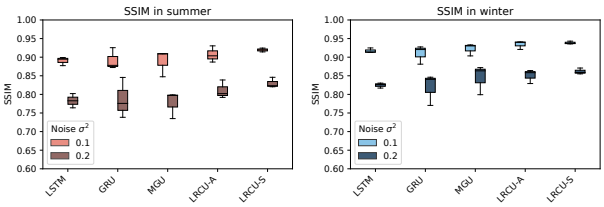
4.1 Accuracy and Speed of LRCs on Neural ODEs Benchmarks

LRCs are competitive in accuracy and convergence speed on popular Neural-ODE benchmarks. As examples, we consider a Periodic Sinusoidal, a Spiral towards the origin Chen et al. [2018b], the non-linear Duffing oscillator [Riyazia et al., 2023, Sholokhov et al., 2023, Constante-Amores et al., 2024] and three versions of the Lotka-Volterra (LV) predator-prey model [Bhattacharya and Martcheva, 2010, ten Klooster, 2021] well-known from evolutionary biology and ecology.

In our experiments we use an LRC with 16 hidden states. LRCs are highly suitable for such tasks as they are ODEs. We also use two extra layers for mapping. The former maps the state of these systems to the dimensionality of the hidden state, allowing us to learn more expressive 16-dimensional dynamics. The latter remaps these dynamics back to the original system state dimensionality. We compare the performance of the LRCs with a 3-layer Neural-ODE, containing 32, 32, and 2 neurons, respectively. These three layers match the number of trainable parameters, with the ones of the LRCs. Neural-ODEs are solved by Runge-Kutta 4(5) of Dormand-Prince, and LRCs by explicit Euler of order 1. As one can see in Figure 4, we obtain for all tasks an arguably perfect fit. The results of our experiments, averaged over 3 seeds, are shown in Figure 4 and in Table 2.

4.2 Interpretability of LRCUs on the Autonomous-Driving Lane-Keeping Task

To show that LRCUs outperform LSTMs, GRUs, and MGUs in terms of interpretability, we conduct Lane-Keeping Task experiments, and compute the following interpretability metrics: the network attention and its robustness by the structural similarity index, the absolute correlation of neural activity with road trajectory, and the activity of neurons.



	Summer	Winter
LSTM	0.400 ± 0.299	0.450 ± 0.262
GRU	0.363 ± 0.256	0.389 ± 0.248
MGU	0.317 ± 0.227	0.308 ± 0.232
LRCU-A	0.500 ± 0.314	0.533 ± 0.292
LRCU-S	$\mathbf{0.645 \pm 0.321}$	$\mathbf{0.766 \pm 0.243}$

Figure 5: Left and middle: Robustness of the attention, measured by the Structural Similarity Index (SSIM) Wang et al. [2004] of the models, in summer (left) and in winter (middle). The LRCU models maintain the most similar focus of their attention in the presence of noise, indicated by the boxplots closer to 1. Lighter color refers to additional Gaussian noise of 0-mean and $\sigma^2 = 0.1$ variance, and darker to $\sigma^2 = 0.2$ variance. Right: Absolute correlation between the neural activity and the trajectory of the road. Values closer to 1 indicate stronger correlation. Averaged over 3 runs.

Understanding neural-network attention during decision-making is a crucial element in increasing trust in the network. By using the VisualBackprop [Bojarski et al., 2016] method, we visualize which pixels in the input image have the most impact at the given timestep. Even though the same structure of CNN heads is used in the training pipeline, the different recurrent parts of the decision-making policy, have a big influence on the learned features in the convolutional part through backpropagation. The attention in summer and winter season is given in the Appendix in Figure 11. By injecting a zero mean Gaussian noise with a variance of 0.1 and 0.2 into the tests, respectively, we can measure how robust their attention is. Quantitatively, this is measured by the Structural Similarity Index, displayed in Figure 5.

As another interpretability metric, we assess how the neural activity within a Lane-Keeping RNN-policy changes during deployment in the closed-loop simulation. Specifically, we are interested in identifying neurons exhibiting an activity matching the geometry of the trajectory. We conduct tests for a 1 km long drive, in both summer and winter. In the Appendix in Figure 12, we illustrate the activity of States h_1-h_2, corresponding to Cells 1-2 in the RNN policy, respectively. Our results reveal that the LSTM, GRU, MGU cells, have a scattered, hard-to-interpret activity, respectively. In contrast, LRCU-S cells demonstrate a very gentle varying activity, that aligns very well with the road's trajectory. This is most likely due to their double-liquid resistance and capacitance time constant. As shown in Table 5, we also compute the absolute-value cross-correlation metric, between the final prediction sequence of the policy (which aligns with the road's trajectory), and the sequence of activities of all individual neurons. By considering the absolute value of the cross-correlations, we ensure that equal importance is given to both positive (that is excitatory) and negative (that is inhibitory) behaviors. This method yields values within the range of $[0, 1]$, where values close to zero indicate little to no correlation, and values close to one mean a very high correlation. The results presented in Table 5 reinforce the observations from Figure 12, which indicate that LRCU-based models, exhibit higher absolute-correlation values with the road's trajectory compared to the other models.

5 Discussion, Scope and Conclusion

We introduced Liquid-Resistance Liquid-Capacitance networks (LRCs), a Neural ODE model which is more accurate, more efficient, and biologically more plausible, compared to neuroscience's Electric Equivalent Circuits of chemical synapses, and Liquid Time-Constant networks (STCs and LTCs).

Due to their gentle varying behavior LRCs are easy to integrate and applicable to a wide range of scenarios, including continuous processes and datasets sampled regularly or irregularly. LRCs thus represent a promising advancement in the field of bio-inspired deep learning. LRCs integrated with explicit Euler of order 1, and a time-step of 1 (LRCUs), have a very similar structure to gated RNNs. They not only demonstrate comparable performance to LSTMs, GRUs, and MGUs on several benchmarks, but also exhibit significantly higher interpretability, demonstrated in the autonomous Lane-Keeping task. Despite the limitation of the benchmarks used, which we acknowledge could be expanded in subsequent work, our focus on analyzing the properties of our advanced bio-inspired model and its performance across diverse scenarios demonstrates its potential and lays a solid foundation for future exploration. Our results suggest that incorporating additional concepts from

neuroscience can advance machine learning models, offering opportunities for future research to explore and further enhance model performance.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101034277.

References

B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J.D. Watson. *Molecular Biology of the Cell*. W. W. Norton & Company, 7th edition, 2008.

Souvik Bhattacharya and Maia Martcheva. Oscillations in a size-structured prey-predator model. *Mathematical Biosciences*, 228(1):31–44, 2010. ISSN 0025-5564. doi: https://doi.org/10.1016/j.mbs.2010.08.005. URL https://www.sciencedirect.com/science/article/pii/S0025556410001252.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural odes. In *Advances in Neural Information Processing Systems*, volume 34, pages 21325–21337. Curran Associates, Inc., 2021.

Mariusz Bojarski, Anna Choromańska, Krzysztof Choromanski, Bernhard Firner, Lawrence D. Jackel, Urs Muller, and Karol Zieba. Visualbackprop: visualizing cnns for autonomous driving. *ArXiv*, abs/1611.05418, 2016. URL https://api.semanticscholar.org/CorpusID:16384108.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems*, volume 32, pages 7379–7390. Curran Associates, Inc., 2019.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018a.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018b. URL https://api.semanticscholar.org/CorpusID:49310446.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International conference on machine learning*, pages 854–863. PMLR, 2017.

C. Ricardo Constante-Amores, Alec J. Linot, and Michael D. Graham. Enhancing predictive capabilities in data-driven dynamical modeling with automatic differentiation: Koopman and neural ode approaches, 2024. URL https://arxiv.org/abs/2310.06790.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Mónika Farsang, Mathias Lechner, David Lung, Ramin Hasani, Daniela Rus, and Radu Grosu. Learning with chemical versus electrical synapses does it make a difference? In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15106–15112, 2024. doi: 10.1109/ICRA57147.2024.10611016.

Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993.

Ramin M. Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *AAAI Conference on Artificial Intelligence*, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

B. Howell, L.E. Medina, and W.M. Grill. Effects of frequency-dependent membrane capacitance on neural excitability. *Neural Engineering*, 12(5):56015–56015, October 2015.

Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, Sarah Mack, et al. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, volume 33, pages 6696–6707. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4a5876b450b45371f6cfe5047ac8cd45-Paper.pdf.

Jitender Kumar, Patrick Das Gupta, and Subhendu Ghosh. Effects of nonlinear membrane capacitance in the hodgkin-huxley model of action potential on the spike train patterns of a single neuron. *Europhysics Letters*, 142(6):67002, jun 2023. doi: 10.1209/0295-5075/acd80c.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Mathias Lechner and Ramin Hasani. Learning long-term dependencies in irregularly-sampled time series. *arXiv preprint arXiv:2006.04418*, 2020.

Mathias Lechner, Ramin Hasani, Manuel Zimmer, Thomas A. Henzinger, and Radu Grosu. Designing worm-inspired neural networks for interpretable robotic control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 87–94, 2019. doi: 10.1109/ICRA.2019.8793840.

Mathias Lechner, Ramin M. Hasani, Alexander Amini, Thomas A. Henzinger, Daniela Rus, and Radu Grosu. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2: 642–652, 2020.

Mathias Lechner, Ramin Hasani, Alexander Amini, Tsun-Hsuan Wang, Thomas A Henzinger, and Daniela Rus. Are all vision models created equal? a study of the open-loop to closed-loop causality gap. *arXiv preprint arXiv:2210.04303*, 2022.

ting li, Jianguo Li, and Zhanxing Zhu. Neural lad: A neural latent dynamics framework for times series modeling. In *Advances in Neural Information Processing Systems*, volume 36, pages 17345–17356. Curran Associates, Inc., 2023.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

Yassin Riyazia, NavidReza Ghanbaria, and Arash Bahramib. Leveraging koopman operator and deep neural net-works for parameter estimation and future prediction of duffing oscillators. *ISAV*, 2023.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Daniel Severin, Sofia Shirley, Alfredo Kirkwood, and Jorge Golowasch. Daily and cell type-specific membrane capacitance changes in mouse cortical neurons. *bioRxiv*, 2022. doi: 10.1101/2022.12.09.519806.

Aleksei Sholokhov, Yuying Liu, Hassan Mansour, and Saleh Nabi. Physics-informed neural ode (pinode): embedding physics into models using collocation points. *Scientific Reports*, 13(1):10166, 2023.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

L.R. ten Klooster. Approximating differential equations using neural odes., July 2021. URL `http://essay.utwente.nl/87568/`.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). *arXiv preprint arXiv:1805.12152*, 2(3), 2018.

Vedrana Vidulin, Mitja Lustrek, Bostjan Kaluza, Rok Piltaver, and Jana Krivec. Localization Data for Person Activity. UCI Machine Learning Repository, 2010. DOI: https://doi.org/10.24432/C57G8X.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Stephen R Wicks, Chris J Roehrig, and Catharine H Rankin. A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, 16(12):4017–4031, 1996.

Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3):226–234, 2016.

A Appendix

This appendix explores the properties of LRCs, provides the proofs of the two theorems stated, discusses the RNN form of LRCUs and their connection to GRUs and other gated recurrent units, and gives additional experimental evaluation and details, demonstrating the accuracy, efficiency, and interpretability of LRCs.

A.1 Properties of LRCs

Lipschitz constant. The dynamic behavior of the LRC output neuron in the Lane-Keeping Task is shown in Figure 1(right). It has a very gently varying behavior, closely following the ground truth. It was demonstrated that a lower Lipschitz constant enhances robustness [Cisse et al., 2017], generalization [Sokolić et al., 2017], and interpretability [Tsipras et al., 2018].

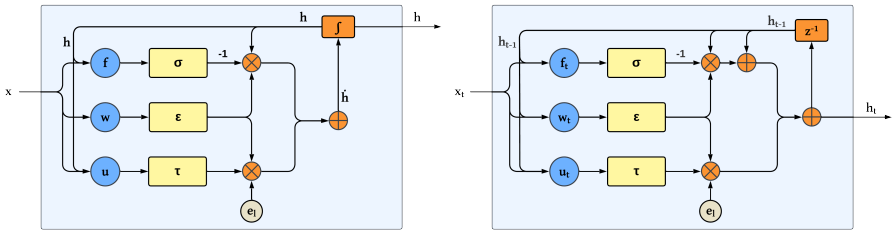


Figure 6: The architecture of LRCs (left) and of LRCUs (right, explicit 1st order Euler discretization of LRCs with one unfolding). Both LRCs and LRCUs have a sigmoidal forget gate $\sigma(f_t)$, a hyperbolic-tangent update gate $\tau(u_t)$, and a sigmoidal symmetric or asymmetric smoothen gate $\epsilon(w_t)$.

Due to its very gentle varying behavior, an explicit Euler integration with one unfolding, is often sufficient for obtaining an acceptable validation loss. This dramatically speeds up computation, as the solver introduces no overhead. This is particularly useful for time series with regularly sampled data, lacking timing information. Here, a time step of one, leads to a network closely resembling gated recurrent neural networks [Hochreiter and Schmidhuber, 1997, Cho et al., 2014, Zhou et al., 2016]. Its architecture is shown in Figure 6 (right). We call this network, an LRC unit (LRCU).

Accuracy. The addition of a liquid elastance in LRCs does not only reduce oscillations, but it also dramatically improves accuracy. This is illustrated for the Lane-Keeping Task in Figure 2, where the validation loss of the LRC's output neuron, is considerably smaller compared to the one of LTCs and STCs. While the above theorem proves a smaller Lipschitz constant, the following theorem proves accuracy.

Efficiency. Solving LTCs, STCs, and LRCs with popular ODE-integration techniques, is computationally expensive. A hybrid forward-backward Euler method [Lechner et al., 2020] for example, takes multiple unfolding rounds for a given input, to achieve a good approximation of the solution. In Figure 2 we show the validation loss of LTCs, STCs, and LRCs, for the Lane-Keeping Task [Lechner et al., 2022], where the state of the neurons is computed with a fixed integration step Δ. Diving this into 6 equidistant time steps $\Delta/6$ within one update step, results in a considerably better model, but at a higher cost of 10min versus 4min per epoch. This results in a $2.5\times$ longer computation time.

In Figure 2, we also compare the training and validation loss of the LRCUs and LRCs, for the Lane-Keeping Task, where we compute the elastance term $\epsilon(w_i)$ in two different ways, by using Equations (4-5), and learning the parameters o, p, and k. Using these proposed liquid elastances leads to a considerably faster convergence of validation loss, to a much smaller loss. In particular, the symmetric elastance achieves the best results, by essentially converging in around 20 epochs, to a loss of about 0.15. The LTCs and STCs instead, converge in about 60 epochs, at a loss of 0.22.

Interpretability. In Lechner et al. [2020], it was shown that trained LTCs are interpretable, when used in a neural-circuit-policy (NCP) architecture. Later on in Farsang et al. [2024], it was shown that interpretability naturally occurs in STCs, even when using a single, dense, all-to-all hidden layer. This is also true for LRCs. As we show in Section 4 for the Lane-Keeping Task, LRCUs lead to very interpretable behavior of neurons in the hidden layer.

A.2 Theorems and Proofs

In this section we provide the theorems and proofs. The LRC Lipschitz Theorem first derives the local Lipschitz constant of LRCs and then shows that it is smaller than the one of STCs. The LRC Generalization Theorem, first computes the error of LRCs on the validation time series, and then shows that it is less than the one of the STCs, considering the validation set being drawn from in-distribution.

A.2.1 Lipschitz constant Theorem and Proof

Theorem 1 (LRC Lipschitz)**.** *Let the Lipschitz constant λ^s bound the sensitivity of an STC-model instance of Equation* (2)*, with respect to its inputs and hidden states. Then, there exists an associated LRC-model instance of Equation* (3)*, with a Lipschitz constant λ^r, such that $\lambda^r < \lambda^s$.*

Intuitively, for an LRC having the parameter values for its forget and update conductances fixed as in an STC, one can set the parameter values of its elastance, such that the Lipschitz constant of the LRC is smaller than the one of the STC. This leads to LRC being a function which does not change rapidly when the input changes.

Proof. Let T be the time horizon of the prediction, K be the dimension of the output, m the dimension of the states, n the dimension of the input and $Q \in \mathbb{R}^{K \times m}$ define the linear output layer. Let the output of the STC be defined by $o^s = Q\,h^s$, with $\dot{h}^s$ satisfying Equations (2), hidden states h^s, inputs x, and linear output Qh^s, and with trained synaptic parameters for Q and $\dot{h}^s$. Let an associated LRC be defined by $o^r = Q\,h^r$, with $\dot{h}^r$ satisfying Equations (3). If we keep in the LRC all the parameters as they were trained for the STC, and only adapt the elastance $\epsilon(w)$, then it holds that $\dot{h}^r = \epsilon(w) \odot \dot{h}^s$.

The Lipschitz constant λ of the output $o(y)$ computed by the LRC and STC networks, is defined such that $\|o(y_1) - o(y_2)\|_2 \leq \lambda \|y_1 - y_2\|_2$ for all y_1, y_2 in the input space of these networks. From the Mean Value Theorem it holds that for every y_1, y_2 it exists a c such that $\|o(y_1) - o(y_2)\|_2 \leq \|\nabla_y o(c)\|_2 \|y_1 - y_2\|_2$. Thus, if a maximum exists, then $\max_c \|\nabla_y o(c)\|_2$ is a Lipschitz constant.

We now want to prove that there exists a choice of parameters for $\epsilon(w)$, such that the Lipschitz constant of the output o^r of the LCR with respect to its hidden states and input $y = [h, x]$, is less than the one of o^s in the STC. So the goal is to state conditions on the parameters of $\epsilon(w)$ such that:

$$\max_y \|\nabla_y o^s\|_2 \leq \lambda^s \;\wedge \max_y \|\nabla_y o^r\|_2 \leq \lambda^r \quad \Rightarrow \quad \lambda^r < \lambda^s \tag{6}$$

Let us use o, h when stating an equation which holds for STCs as well as LRCs:

$$o = Qh, \qquad o_k = \sum_{i=1}^{m} q_{ki} h_i \tag{7}$$

$$\frac{\partial o_k}{\partial y_j} = \sum_{i=1}^{m} q_{ki} \frac{\partial h_i}{\partial y_j} \tag{8}$$

With Δ_t the time difference between two outputs, and by using Leibniz integral rule it holds that:

$$\left| \frac{\partial h_t}{\partial y_{t,j}} \right| = \left| \frac{\partial h_{t-\Delta t}}{\partial y_{t,j}} + \frac{\partial}{\partial y_{t,j}} \int_{t-\Delta t}^{t} \dot{h} dt \right| \tag{9}$$

$$= \left| \underbrace{\frac{\partial h_{t-\Delta t}}{\partial y_{t,j}}}_{\leq 1} + \int_{t-\Delta t}^{t} \frac{\partial \dot{h}}{\partial y_{t,j}} dt \right| \tag{10}$$

$$\leq 1 + \left| \int_{t-\Delta t}^{t} \max_y \frac{\partial \dot{h}}{\partial y_{t,j}} dt \right| \tag{11}$$

As $\dot{h}$ is time-independent, $\max_y \partial_{y_j} \dot{h}$ is the same for all t, thus:

$$\left| \frac{\partial h_t}{\partial y_{t,j}} \right| \leq 1 + \left| \Delta t \cdot \max_y \frac{\partial \dot{h}}{\partial y_{t,j}} \right| \tag{12}$$

$$\overset{(8)}{\Rightarrow} \left|\frac{\partial o_k}{\partial y_j}\right| \leq \sum_{i=1}^{m} |q_{ki}| \left(1 + \Delta t \cdot \max_y \left|\frac{\partial \dot{h}_i}{\partial y_j}\right|\right) = v_{kj} \tag{13}$$

$$\max_y \|\nabla_y o\|_2 \leq \|V\|_2 = \lambda \tag{14}$$

Here $V \in \mathbb{R}^{K \times m}$ contains the values v_{kj} from Equation (13). When bounding V later on in the proof, we will need the following known upper bounds of derivatives:

$$\sigma'(x) \leq 0.25, \quad \tau'(x) \leq 1 \tag{15}$$

Using these bounds, for the symmetric version of elastance in Equation (5) it holds that:

$$\epsilon'(w_i) = \sigma'(w_i + k_i) - \sigma'(w_i - k_i) \leq \max \sigma'(w_i + k_i) - \min \sigma'(w_i - k_i) \leq 0.25 \tag{16}$$

Further, we will need later on, that for $\epsilon(w_i) \in (0, 1)$ it holds that:

$$\frac{\epsilon'(w_i)}{1 - \epsilon(w_i)} \leq \epsilon(w_i) \tag{17}$$

For asymmetric ϵ this is straightforward as the derivivative of a sigmoid is $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$. For symmetric ϵ we need to reformulate the equations using that $\sigma(w_i + k_i) \geq \sigma(w_i - k_i)$ for $k_i \geq 0$:

$$\frac{\epsilon'(w_i)}{1 - \epsilon(w_i)} = \frac{\sigma(w_i + k_i)(1 - \sigma(w_i + k_i)) - \sigma(w_i - k_i)(1 - \sigma(w_i - k_i))}{1 - \sigma(w_i + k_i) + \sigma(w_i - k_i)} \tag{18}$$

$$\leq \frac{\sigma(w_i + k_i)(1 - \sigma(w_i + k_i)) - \sigma(w_i - k_i)(1 - \sigma(w_i + k_i))}{1 - \sigma(w_i + k_i)} \tag{19}$$

$$= \epsilon(w_i) \tag{20}$$

Now we can derive an upper bound for the sensitivity of $\dot{h}_i^s$ to the input or hidden state y_j:

$$\left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right| \leq \left|\frac{\partial \sigma(f_i)}{\partial y_j} h_i^s\right| + \underbrace{\left|\sigma(f_i) \frac{\partial h_i^s}{\partial y_j}\right|}_{\leq 1} + \left|\frac{\partial \tau(u_i)}{\partial y_j} e_{li}\right| \tag{21}$$

$$\leq |\sigma'(f_i) g_{ji} \sigma'(a_{ji} y_j + b_{ji}) a_{ji} h_i^s| + |\tau'(u_i) k_{ji} \sigma'(a_{ji} y_j + b_{ji}) a_{ji} e_{li}| + 1 \tag{22}$$

$$\leq 0.0625 |g_{ji} a_{ji} h_i^s| + 0.25 |k_{ji} a_{ji} e_{li}| + 1 = \max_y \left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right| \tag{23}$$

For the sensitivity of the LRC it holds that:

$$\dot{h_i^r} = \epsilon(w_i) \cdot \dot{h_i^s} \tag{24}$$

$$\left|\frac{\partial \epsilon(w_i)}{\partial y_j}\right| = |\epsilon'(w_i) \cdot o_{ji}| \tag{25}$$

$$\left|\frac{\partial \dot{h_i^r}}{\partial y_j}\right| = \left|\epsilon(w_i) \cdot \frac{\partial \dot{h}_i^s}{\partial y_j} + \frac{\partial \epsilon(w_i)}{\partial y_j} \cdot \dot{h}_i^s\right|, \tag{26}$$

Here o_{ji} are the trainable parameters of w_i. As $\frac{\partial \epsilon(w_i)}{\partial y_j} = \epsilon(w_i)(1 - \epsilon(w_i)) o_{ji}$, for $\epsilon(w_i) \in \{0, 1\}$ it follows from Equation (26) that $|\partial_{y_j} \dot{h_i^r}| = |\partial_{y_j} \dot{h}_i^s|$. For $\epsilon(w_i) \in (0, 1)$:

$$\left|\frac{\partial \dot{h}_i^r}{\partial y_j}\right| = \left|\epsilon(w_i) \cdot \frac{\partial \dot{h}_i^s}{\partial y_j} + \frac{\partial \epsilon(w_i)}{\partial y_j} \cdot \dot{h}_i^s\right| \tag{27}$$

$$\leq \epsilon(w_i) \cdot \left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right| + \epsilon'(w_i) \cdot |o_{ji}| \cdot |\dot{h}_i^s| \tag{28}$$

Now let us choose the trainable parameters o_{ji} in such a way, that:

$$|o_{ji}| \leq \min\{0.0625 |g_{ji} a_{ji}|, 0.25 |k_{ji} a_{ji}|\} = o_{ji}^* \tag{29}$$

Then for this choice, and using Equation (17) it holds that:

$$\frac{\epsilon'(w_i)}{1-\epsilon(w_i)}|o_{ji}||\dot{h}_i^s| \overset{(17)}{\leq} \epsilon(w_i)|o_{ji}||\dot{h}_i^s| \tag{30}$$

$$\overset{(2),(15)}{\leq} \epsilon(w_i)|o_{ji}|(|h_i^s| + |e_{li}|) \tag{31}$$

$$\leq |o_{ji}||h_i^s| + |o_{ji}||e_{li}| \tag{32}$$

$$\overset{(29)}{\leq} 0.0625|g_{ji}a_{ji}h_i^s| + 0.25|k_{ji}a_{ji}e_{li}| + 1 \overset{(23)}{=} \max_y \left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right|, \tag{33}$$

which is equivalent to:

$$\epsilon'(w_i) \cdot |o_{ji}| \cdot |\dot{h}_i^s| \leq (1-\epsilon(w_i)) \max_y \left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right| \tag{34}$$

$$\Leftrightarrow$$

$$\epsilon(w_i) \cdot \max_y \left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right| + \epsilon'(w_i) \cdot |o_{ji}| \cdot |\dot{h}_i^s| \leq \max_y \left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right| \tag{35}$$

$$\overset{(28)}{\Rightarrow}$$

$$\max_y \left|\frac{\partial \dot{h}_i^r}{\partial y_j}\right| \leq \max_y \left|\frac{\partial \dot{h}_i^s}{\partial y_j}\right| \tag{36}$$

If for all i, j Equation (29) holds, and if there exists at least one pair $\bar{i}, \bar{j}$ such that $|o_{\bar{j}\bar{i}}| < o^*_{\bar{j}\bar{i}}$, then also Equation (36) holds with inequality for these indices ($\max_y \partial_{y_{\bar{j}}} \dot{h}_{\bar{i}}^r < \max_y \partial_{y_{\bar{j}}} \dot{h}_{\bar{i}}^s$) and thus by using Equations (13)- (14) it follows that:

$$v_{kj}^r \leq v_{kj}^s \quad \forall j = \{1, \ldots, m+n\} \tag{37}$$

$$v_{k\bar{j}}^r < v_{k\bar{j}}^s \tag{38}$$

$$\Rightarrow \lambda^r < \lambda^s \tag{39}$$

This proves that the Lipschitz constant of LRCs is smaller than the Lipschitz constant of STCs. □

A.2.2 Generalization Theorem and Proof

Theorem 2 (LRC Generalization)**.** *Let h^s be an STC-model instance of Equation* (2) *with h^r being an associated LRC-model instance of Equation* (3) *with $\lambda^r < \lambda^s$ (Theorem. A.2.1). Let o_t^T be training labels and $\hat{o}_t^{T,r}$ and $\hat{o}_t^{T,s}$ be the predictions of the models h^r and h^s respectively. Let both models have a small training loss $\|o_t^T - \hat{o}_t^T\|_2 < \epsilon_T$ with small $\epsilon_T > 0$. Let y_t^T and y_t^V be respectively training and validation input. Let the validation set be similar to the training set (drawn in-distribution) in a sense that for every time t it holds that $\|y_t^T - y_t^V\|_2 < \epsilon_y$ and $\|o_t^T - o_t^V\|_2 < \epsilon_o$ with small $\epsilon_y, \epsilon_o > 0$. Then it holds that the upper bound of the validation loss of the LRC-model is smaller than the one of the STC-model.*

Intuitively, if the validation set is similar to the training set modulo some small perturbation, are drawn from the same distribution, one can use a Taylor series approximation of the validation loss $o(y^V) \approx o(y^T) + (y^V - y^T)^T \nabla_y o(y^V)$ and get an upper bound on the validation loss by using Theorem A.2.1. As the Lipschitz constant of the LRC is smaller than the one of STC, also the upper bound of the validation loss of the LRC-model is smaller than the one of the STC-model.

Proof. Let $o_1^T, \ldots, o_T^T$ be the labels of the training set, $o_1^V, \ldots, o_T^V$ be the ones of the validation set and $\hat{o}_t^T$ and $\hat{o}_t^V$ be respectively the output of the models of the training and the validation set. The STC or LRC model will be denoted in the superscript as s or r. When not using one of these superscripts, the statement holds for both models. Let T be the time horizon of the prediction.

Let the loss function for the STC and LRC networks be defined as usual, as follows:

$$L(o, \hat{o}) = \frac{1}{T} \sum_{t=1}^{T} \|o_t - \hat{o}_t\|_2, \tag{40}$$

Then for the validation loss it holds that:

$$\begin{aligned}
L(o^V, \hat{o}^V) &= \frac{1}{T} \sum_{t=1}^{T} \|o_t^V - \hat{o}_t^V\|_2 && (41)\\
&= \frac{1}{T} \sum_{t=1}^{T} \|o_t^V - o_t^T + o_t^T - \hat{o}_t^T + \hat{o}_t^T - \hat{o}_t^V\|_2 && (42)\\
&\leq \frac{1}{T} \sum_{t=1}^{T} \|o_t^V - o_t^T\|_2 + \|o_t^T - \hat{o}_t^T\|_2 + \|\hat{o}_t^T - \hat{o}_t^V\|_2 && (43)\\
&\leq \frac{1}{T} \sum_{t=1}^{T} \epsilon_o + \epsilon_T + \|\hat{o}_t^T - \hat{o}_t^V\|_2 && (44)
\end{aligned}$$

As $\hat{o}_t^V$ is a function of the input y_t^V, we will switch to the notation: $\hat{o}_t^T = \hat{o}(y_t^V)$. Moreover, as we assumed that $\|y_t^T - y_t^V\|_2 < \epsilon_y$, we can use a Taylor-series approximation:

$$\begin{aligned}
\hat{o}(y_t^V) &\approx \hat{o}(y_t^T) + (y_t^V - y_t^T)^T \nabla_y \hat{o}(y_t^V) && (45)\\
&\Leftrightarrow \\
\hat{o}(y_t^V) - \hat{o}(y_t^T) &\approx (y_t^V - y_t^T)^T \nabla_y \hat{o}(y_t^V) && (46)\\
&\Rightarrow \\
\|\hat{o}(y_t^V) - \hat{o}(y_t^T)\|_2 &\approx \|(y_t^V - y_t^T)^T \nabla_y \hat{o}(y_t^V)\|_2 && (47)\\
&\leq \|(y_t^V - y_t^T)\|_2 \cdot \|\nabla_y \hat{o}(y_t^V)\|_2 && (48)\\
&\leq \epsilon_y \cdot \lambda, && (49)
\end{aligned}$$

Here, λ is the Lipschitz constant of the model. Putting this upper bound into Equation (44), we can set the upper bound of the validation loss $\bar{L}(o^V, \hat{o}^V)$ as follows:

$$L(o^V, \hat{o}^V) \leq \epsilon_o + \epsilon_T + \epsilon_y \cdot \lambda = \bar{L}(o^V, \hat{o}^V) \tag{50}$$

Finally, by using Theorem A.2.1, it holds that:

$$\bar{L}(o^V, \hat{o}^{V,r}) = \epsilon_o + \epsilon_T + \epsilon_y \cdot \lambda^r < \epsilon_o + \epsilon_T + \epsilon_y \cdot \lambda^s = \bar{L}(o^V, \hat{o}^{V,s}) \tag{51}$$

This proves that the validation loss of LRCs is smaller than the validation loss of STCs. □

A.3 Liquid Resistance Liquid Capacitance Units (LRCU)

In this section we first introduce the RNN formulation of LRCUs, which as discussed in the paper, are a very accurate and efficient Euler integration of order one of LRCs, with a time step of one. Since LRCUs turn out to be a new form of gated recurrent units, we explore their relation to GRUs.

A.3.1 LRCUs as an RNN

Starting from the Neural-ODEs model of saturated biological neurons with chemical synapses (the saturated EECs) of Equation (2), we have shown that first considering a liquid capacitance (elastance) as in Equation (3) and then discretizing the ODEs leads to a very accurate and efficient gated RNN, which we called an LRCU. The formulation of this discrete unit is the following:

$$\begin{aligned}
h_{i,t} &= (1 - \epsilon(w_{i,t})\, \sigma(f_{i,t}))\, h_{i,t-1} + \epsilon(w_{i,t})\, \tau(u_{i,t})\, e_{li} \\
f_{i,t} &= \textstyle\sum_{j=1}^{m+n} g_{ji} \sigma(a_{ji} y_{j,t} + b_{ji}) + g_{li} \\
u_{i,t} &= \textstyle\sum_{j=1}^{m+n} k_{ji} \sigma(a_{ji} y_{j,t} + b_{ji}) + g_{li} \\
w_{i,t} &= \textstyle\sum_{j=1}^{m+n} o_{ji} y_{j,t} + p_i
\end{aligned} \tag{52}$$

Here, one can use either an asymmetric or a symmetric form of elastance ϵ in Equations (4-5), respectively. We will denote the first choice as LRCU-A, and the second choice as LRCU-S.

Next, we show how can one re-formulate GRUs and make connections to LRCUs. MGUs and LSTMs could be related to LRCUs, with minor modifications, in a similar manner.

A.3.2 LRCUs versus GRUs

In this section we explore the connections between LRCUs and GRUs. The general form of a gated recurrent units (GRU) is according to Cho et al. [2014], an RNN of the following form:

$$\begin{aligned} h_{i,t} &= (1-\sigma(f_{i,t}))\, h_{i,t-1} + \sigma(f_{i,t})\, \tau(u_{i,t}) \\ f_{i,t} &= \textstyle\sum_{j=1}^{m+n} a^f_{ji}\, y_{j,t} + b^f_j \\ u_{i,t} &= \textstyle\sum_{j=1}^{m+n} a^u_{ji}\, y'_{j,t} + b^u_j \\ r_{i,t} &= \textstyle\sum_{j=1}^{m+n} a^r_{ji}\, y_{j,t} + b^r_j \end{aligned} \tag{53}$$

Here, vector y_t occurring in functions $f_{i,t}$ and $r_{i,t}$, is defined as before, as $y_t = [h_{t-1}, x_t]$. However, vector y'_t occurring in $u_{i,t}$ is defined as $y'_t = [\sigma(r_t) \odot h_{t-1}, x_t]$. Thus, previous state h_{t-1} is pointwise scaled in the update part $\tau(u_{i,t})$ of the GRU, with a nonlinear state-and-input dependent function $\sigma(r_{i,t})$, whose parameters are to be learned. This function is called in GRUs a Reset Gate (RG). Moreover, the state-and-input dependent function $\sigma(f_{i,t})$ is called in GRUs an Update Gate (UG).

The RG determines how previous state h_{t-1} is used in the update $\tau(u_{i,t})$. The UG $\sigma(f_{i,t})$ controls the amount $(1-\sigma(f_{i,t}))$ of the previous state $h_{i,t-1}$, to be remembered in the next state. However, this UG also controls the amount of the update $\tau(u_{i,t})$ to be considered in the next state, by using it to multiply the update. One can identify the GRU's UG with the LRCU's smoothen (elastance) gate.

Given the above discussion, GRUs can also be understood as the ordinary difference equations associated with the Neural ODEs below. From this, one can get back to the original form of Equations (53), using an explicit Euler integration scheme of order one with a unit time difference:

$$\begin{aligned} \dot{h}_i &= \sigma(f_i)(-h_i + \tau(u_i)) \\ f_i &= \textstyle\sum_{j=1}^{m+n} a^f_{ji}\, y_j + b^f_j \\ u_i &= \textstyle\sum_{j=1}^{m+n} a^u_{ji}\, y'_j + b^u_j \\ r_i &= \textstyle\sum_{j=1}^{m+n} a^r_{ji}\, y_j + b^r_j \end{aligned} \tag{54}$$

Here the vector $y = [h, x]$ is defined as before, and vector $y' = [\sigma(r) \odot h, x]$. The RG occurring in y' determines how state h is used in the update part $\tau(u_i)$ of the ODE. The LRC's time constant (TC) thus consists of a liquid resistance liquid capacitance, while the GRU's TC consists of a liquid capacitance, only. As a consequence, GRUs have a less expressive TC but more expressive update.

A.4 Experiments for LRC(U)s

In this section, we provide additional experiments and implementation details about our experimental evaluation of LRC in ODE modeling and LRCUs on the popular benchmarks for gated recurrent units. The benchmarks considered were the Localization Data for Person Activity, the IMDB Movie Sentiment Classification, the Permuted Sequential MNIST, and a relatively complex autonoumous-driving in the Lane-Keeping Task.

In general, LRC(u)s have more trainable parameters per cell. Therefore, we are training the other models with more cells to ensure a fair comparison between them.

The experiments are run on an Nvidia Tesla T4.

A.4.1 Neural ODE Experiments

For ODE modeling tasks, we used a 3-layer (units of 32, 32 and 2) Neural ODE and the LRC (units of 16) with additional input and output mapping, which hold 1k trainable parameters. The hyperparameters are presented in Table 1. We use sequences of 16 points during training and test the

models on the whole sequences of 1000 data points giving them only the initial state information of (x_0, y_0). As some ODE modeling tasks are more challenging, we allocated longer training iteration time for them. The training time is 0.4-0.5 seconds per iteration for both models.

Table 1: Hyperparameters of the Neural ODE experiments.

Variable	Value
Learning rate	10^{-3}
Batch size	16
Training sequence length	16
Iterations	$1000/2000/4000$

- Periodic Sinusoidal: $dx/dt = x \cdot (1 - \sqrt{x^2 + y^2}) - y$, $dy/dt = x + y \cdot (1 - \sqrt{x^2 + y^2})$.
- Spiral: $dx/dt = Ax$, where $A = [[-0.1, 3], [-3, -0.1]$.
- Duffing Oscillation: $dx/dt = y$, $dy/dt = x - x^3$.
- Periodic Lotka-Volterra: $dx/dt = a \cdot x - b \cdot x \cdot y$, $dy/dt = -c \cdot y + d \cdot x \cdot y$ with $a = 1.5$, $b = 1$, $c = 3$ and $d = 1$.
- Asymptotic Lotka-Volterra: $dx/dt = x \cdot (1 - x) - x \cdot y$, $dy/dt = -y + d \cdot x \cdot y$ with $d = 2$.
- Non-linear Lotka-Volterra: $dx/dt = x \cdot (1 - x) - a \cdot x \cdot y$, $dy/dt = y \cdot (1 - y) + x \cdot y$ with $a = 0.33$.

These examples originate from the TensorFlow implementation of Chen et al. [2018b], tfdiffeq.

A.4.2 Accuracy and Speed of LRCUs on RNNs Benchmarks

In this section we ask and positively answer, if LRCUs are competitive with respect to accuracy and convergence speed, compared to the popular gated RNNs. To this end, we conduct experiments on a wide range of time-series modeling applications, including: Classification of activities based on irregularly sampled localization data; IMDB movie reviews; and Permuted sequential MNIST tasks.

To achieve a somewhat similar number of parameters, we use 64 cells for LRCUs, and 100 cells for the others. The total number of parameters in a model also depends on the number of inputs and outputs of the tasks considered. Finally, we also conduct experiments in a relatively complex and high-dimensional, Image-based Regression Task, for lane keeping in autonomous vehicles.

All experimental details, including the hyperparameters and the number of trainable parameters, are given in Appendix A.4. In Table 3 we provide the accuracy of the various models compared. Moreover, in Figure 7 we give the validation accuracy convergence curves for the compared models.

Localization Data for Person Activity. The localization data for the Person Activity dataset given in Vidulin et al. [2010], captures the recordings of five individuals, engaging in various activities. Each wore four sensors at the left and the right ankle, at the chest, and at the belt, while repeating the same activity five times. The associated task, is to classify their activity based on the irregularly sampled time-series. This task is a classification problem, adapted from Lechner and Hasani [2020].

Table 2: Test loss of various ODE tasks, with LRCs showing good performance. Results are averaged over 3 seeds.

Task	Neural-ODE	LRC
Sinusoid	0.140 ± 0.033	$\mathbf{0.019 \pm 0.005}$
Spiral	0.012 ± 0.005	$\mathbf{0.009 \pm 0.004}$
Duffing Oscill.	0.142 ± 0.117	$\mathbf{0.003 \pm 0.001}$
Periodic LV	0.025 ± 0.006	$\mathbf{0.005 \pm 0.001}$
Asymptotic LV	0.030 ± 0.0002	$\mathbf{0.009 \pm 0.001}$
Non-linear LV	0.033 ± 0.010	$\mathbf{0.008 \pm 0.004}$

Table 3: Classification accuracy in percentage. For similar number of trainable parameters, we used 64 cells in LRCUs and 100 cells in the other models. We repeated the experiments $5\times$ for the irregularly sampled localization of person activity, and $3\times$ for IMDB and permuted sequential MNIST classification tasks, respectively.

Task	LSTM	GRU	MGU	LRCU-A	LRCU-S
Localization	82.90 ± 0.31	82.76 ± 0.41	83.35 ± 0.30	83.90 ± 0.34	$\mathbf{84.21 \pm 0.26}$
IMDB	86.56 ± 0.49	86.32 ± 0.51	85.18 ± 0.85	$\mathbf{87.00 \pm 0.53}$	85.73 ± 0.41
psMNIST	91.20 ± 0.10	90.20 ± 0.44	87.78 ± 0.82	91.31 ± 0.33	$\mathbf{91.74 \pm 0.41}$

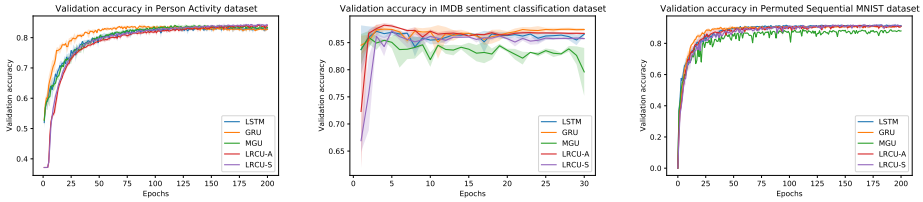


Figure 7: Accuracy convergence. Left: Irregularly sampled person activity dataset. As one can see, LRCUs converge a bit slower, yet they achieve the best accuracy after 100 epochs. Middle: IMDB sentiment classification problem. LRCU-A achieves the highest accuracy after only 5 epochs. The accuracy though, decreases a bit over time. Right: Permuted sequential MNIST dataset. LRCUs demonstrate a sligtly better validation accuracy when compared to GRUs and LSTMs.

In Table 3, we present the experimental results for the accuracy of the classification for LSTMs, GRUs, MGUs, and LRCUs. The experiments for the LRCUs were done with both the asymmetric (LRCU-A) and the symmetric (LRCU-S) elastance, respectively. As one can see, both LRCUs considerably outperformed the traditional gated RNNs. In particular, the LRCU-S performed best, achieving an accuracy of 84.99%, slightly better than the LRCU-A. In Figure 7 we compare the validation accuracy. LRCUs converge slower, but they both surpass the other models after 100 epochs.

Considering that this is an irregularly sampled dataset, including separate extra timestep information per input, traditionally used gated units need further modification when dealing with this task.

Table 4: Number of trainable parameters in the Localization Data for Person Activity dataset. To have at least the same number of trainable parameters, we employed 64 cells for LRCUs and 100 cells in the other models.

	LSTM	GRU	MGU	LRCU-A	LRCU-S
Parameters	40k	30k	20k	20k	20k

For LSTMs, GRUs, and MGUs we concatenate the time-step information (difference in time between to inputs) directly with the input features. This time step is used as the Δ_t value in LRCUs. LSTM, GRU and MGU networks contain 100 units, while LRCUs have 100 units.

In Table 3 we provide the total number of trainable parameters used by each model. Moreover, in Table 4 we provide information about the hyperparameters used by all models to solve this task.

IMDB Movie Sentiment Classification. The IMDB Movie-Review Sentiment Classification dataset, also known as the Large Movie Review Dataset [Maas et al., 2011], is designed for binary sentiment classification. This data set includes 25,000 movie reviews, for both training and testing. Each review was labeled with either a positive or a negative sentiment.

Table 3 presents our experimental results for this task, too. While the LRCU-S achieved a performance which was comparable to the one of the traditional RNNs, the LRCU-A had the best performance, by achieving an accuracy of 87%. Moreover, LRCU-A achieves the highest accuracy after only 5 epochs, even though this decreases a bit over time, as can be seen in the second image of Figure 7.

In the IMDB review dataset we keep the 20,000 most frequent words and truncate the sequences up to 256 characters. Token embeddings of size 64 are used. LSTM, GRU and MGU have 100 units, while LRCU variants have 64 cells in the networks.

Table 5: Hyperparameters of the Localization Data for Person Activity experiment.

Variable	Value
Learning rate	10^{-3}
Batch size	128
Training sequence length	32
Epochs	100

Table 6: Total number of trainable parameters in the IMDB Movie Sentiment Classification task. We used 64 cells in the LRCUs and 100 cells in the other models.

	LSTM	GRU	MGU	LRCU-A	LRCU-S
Parameters	70k	50k	35k	40k	40k

In Table 6 we provide the total number of trainable parameters used by each model to solve this task. Moreover, in Table 7 we provide information about the hyperparameters used to solve this task.

Permuted Sequential MNIST. The Permuted Sequential MNIST dataset is a variant of the MNIST Digits Classification data set, designed to evaluate recurrent neural networks, adapted from Le et al. [2015]. In this task, the 784 pixels (originally images of size 28×28) of digits are presented sequentially to the network. The main challenge lies in predicting the digit category, only after all pixels are observed. This task tests the network's ability to handle long-range dependencies. To make the task even more complex, a fixed random permutation of the pixels is first applied.

As in the other classification tasks, traditionally used gated networks, such as LSTMs, GRUs and MGUs, have 100 units, and the proposed LRCUs contain 64 units. In Table 7 we provide details about the hyperparameters used in this task. Moreover, in Table 8, we provide the total number of trainable parameters for each model used.

During the training of this task, the computed loss values of MGUs became NaNs in 2 out of the 3 runs in the middle of the experiments. Thus, for plotting the validation accuracy in Figure 7, we included only the one fully successful run for them.

in Table 8 we provide the total number of trainable parameters used by each model to solve this task. Moreover, in Table 9 we provide information about the hyperparameters used to solve this task.

A.4.3 Lane-Keeping Task

In the Lane-Keeping Task, the control agent is provided with the front-camera input, consisting of RGB images of size 48×160 pixels. It is required to autonomously navigate and maintain its position within the road, by properly predicting its curvature. The predicted road curvature corresponds to the steering action necessary for lane-keeping, and holds the advantage of being vehicle-independent, as the actual steering angle depends on the type of car used.

The dataset for the Lane-Keeping Task task is obtained from human-driving recordings captured under various weather and illumination conditions [Lechner et al., 2022], as shown in Figure 8. The network architecture contains a CNN-head for extracting the main features from the camera input. They are fed into the gated recurrent models, for the sequential-regression prediction. This network architecture is illustrated on the right in Figure 8. This setup is adapted from Farsang et al. [2024].

In Figure 9, we report the experimental losses of the models considered on the Lane-Keeping task. As it was the case before, both LRCU-A and LRCU-S obtain comparable validation losses, with LRCU-A achieving the best weighted-validation loss (loss weighted by road curvature).

We test the models in the closed-loop setting to evaluate their performance of robust lane keeping. By injecting extra Gaussian noise with variance 0.1 into the input images, we find that LRCUs can handle it without any crashes, and show better performance than other models in even higher Gaussian noise with variance 0.2 too, as reported in Table 13.

In this section we provide details about the Lane-Keeping-Task RNN-policy architecture. In Table 10 below we describe the convolutional head used by the policy.

Table 7: Hyperparameters of the IMDB Movie Sentiment Classification task.

Variable	Value
Learning rate	10^{-3}
Batch size	64
Training sequence length	256
Epochs	30

Table 8: Total number of parameters in the Permuted Sequential MNIST task. LRCUs use 64 neurons, whereas the other 100 neurons, each. All gated-RNN models were run for 3 seeds.

	LSTM	GRU	MGU	LRCU-A	LRCU-S
Parameters	40k	30k	20k	20k	20k

For the Lane-Keeping task, the total number of parameters of the recurrent part is 8k, where the number of neurons used for LSTMs, GRUs, MGUs, LRCU-A, and LRCU-S are 23, 28, 38, 19, and 19 respectively. This choice ensures a similar number of trained parameters.

In Table 11 we provide a description of the hyperparameters used in the Lane-Keeping Experiment. They had the same value for all the RNN models compared.

A.5 Interpretability Experiments for LRCUs

In this section we provide additional details for the interpretability experiments for the Lane-Keeping Task.

Figure 10 shows our results in the summer season, where lighter-highlighted regions indicate the attention of the network. We found that the LSTM takes into account irrelevant parts of the image, during its decision-making. The rest of the models have most of the attention on the road, and LRCU networks especially, focus on the horizon. In Figure 11, we provide the attention maps for the winter season.

In Figure 11, we provide the attention maps for the winter season.

In Figure 13 we provide the neural activity of two cells of the lane-keeping policy for the winter season. As one can clearly see, the LRCU-S has a very smooth behavior closely matching the road geometry: Cell 1 is responsible for turning left, and Cell 2 is responsible for turning right.

Table 9: Hyperparameters of the Permuted Sequential MNIST experiment.

Variable	Value
Learning rate	10^{-3}
Batch size	64
Training sequence length	784
Epochs	200

Figure 8: Lane-Keeping task. The first two figures show summer and winter conditions, respectively. The red rectangle indicates the input and the blue line the predicted steering angle of the network. The third figure illustrates the overall network architecture for this task. The CNN head extracts the input features of a video stream. They are passed to the recurrent policy, responsible for steering.

Table 10: The shape and size of the layers in the convolutional head of the RNN-policy used to solve the Lane-Keeping Task. Settings are adapted from Farsang et al. [2024].

Layer Type	Settings
Input	Input shape: (48, 160, 3)
Image-Norm.	Mean: 0, Variance: 1
Conv2D	Filters: 24, Kernel size: 5, Stride: 2, Activ.: ReLU
Conv2D	Filters: 36, Kernel size: 5, Stride: 1, Activ.: ReLU
MaxPool2D	Pool size: 2, Stride: 2
Conv2D	Filters: 48, Kernel size: 3, Stride: 1, Activ.: ReLU
MaxPool2D	Pool size: 2, Stride: 2
Conv2D	Filters: 64, Kernel size: 3, Stride: 1, Activ.: ReLU
MaxPool2D	Pool size: 2, Stride: 2
Conv2D	Filters: 64, Kernel size: 3, Stride: 1, Activ.: ReLU
Flatten	-
Dense	Units: 64

Table 11: Hyperparameters of the Lane-Keeping experiment.

Variable	Value
Learning rate	cosine decay, $5 \cdot 10^{-4}$
Weight decay	10^{-6}
Batch size	32
Training sequence length	32
Epochs	100

Table 12: Training time per epoch.

Model	Time/epoch
LSTM	2.57 min
GRU	2.50 min
MGU	3.08 min
LTC	10.06 min
STC	10.13 min
LRCU-A	4.05 min
LRCU-S	4.16 min

Model	Validation Loss	W-Validation Loss
LSTM	$\mathbf{0.139 \pm 0.008}$	0.013 ± 0.0010
GRU	0.145 ± 0.006	0.013 ± 0.0010
MGU	$\mathbf{0.139 \pm 0.007}$	0.012 ± 0.0010
LTC	0.204 ± 0.004	$\mathbf{0.011 \pm 0.0004}$
STC	0.201 ± 0.006	$\mathbf{0.011 \pm 0.0010}$
LRCU-A	0.154 ± 0.005	$\mathbf{0.010 \pm 0.0003}$
LRCU-S	0.142 ± 0.004	$\mathbf{0.011 \pm 0.0010}$

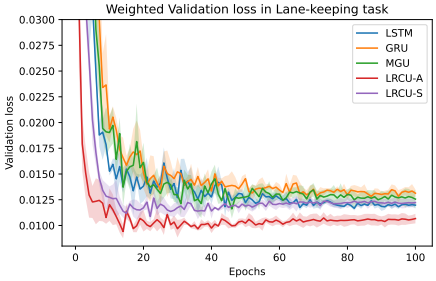


Figure 9: Mean squared loss results for the Lane-Keeping task, averaged over 3 seeds.

Table 13: Crash likelihood in closed-loop simulation with additional Gaussian noise in the Lane-Keeping task. Values closer to zero indicate reliable, crash-free behavior.

Model	Noise $\sigma^2 = 0.1$	Noise $\sigma^2 = 0.2$
LSTM	$\mathbf{0.000 \pm 0.000}$	0.483 ± 0.254
GRU	0.133 ± 0.189	0.767 ± 0.047
MGU	0.267 ± 0.221	0.817 ± 0.037
LRCU-A	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.200 \pm 0.231}$
LRCU-S	$\mathbf{0.000 \pm 0.000}$	0.300 ± 0.216

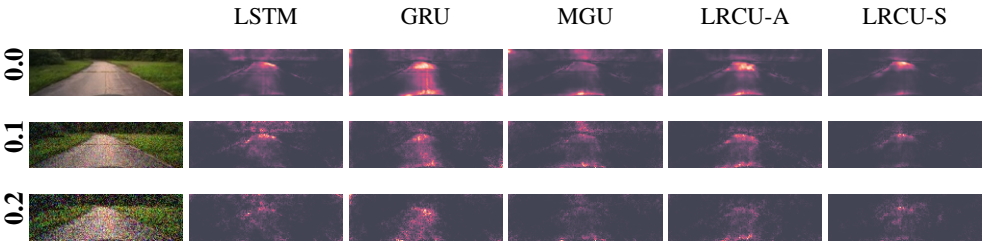


Figure 10: Interpretability. Attention in summer. Column 1 shows the input of the network: Row 1, without additional noise, Rows 2-3, with Gaussian noise of variance $\sigma^2 = 0.1$ and $\sigma^2 = 0.2$, respectively. Remaining columns correspond to the networks, showing their attention to the same input image. One can observe how much the focused areas get distorted in the presence of noise.

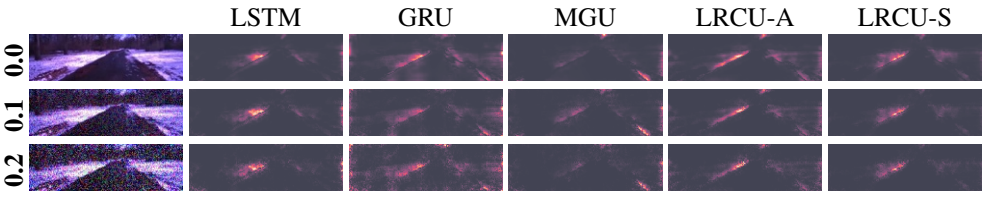


Figure 11: Attention maps in the winter season. In general, the focus shifts from the road to the side of the road compared to summer. Unimportant regions on the off-road are attended by LSTM, GRU and MGU. A winter input image and its saliency maps of the analyzed models are displayed, with increasing noise of Gaussian noise of $\sigma^2 = 0.1$ and $\sigma^2 = 0.2$ variances, in the second and third rows.

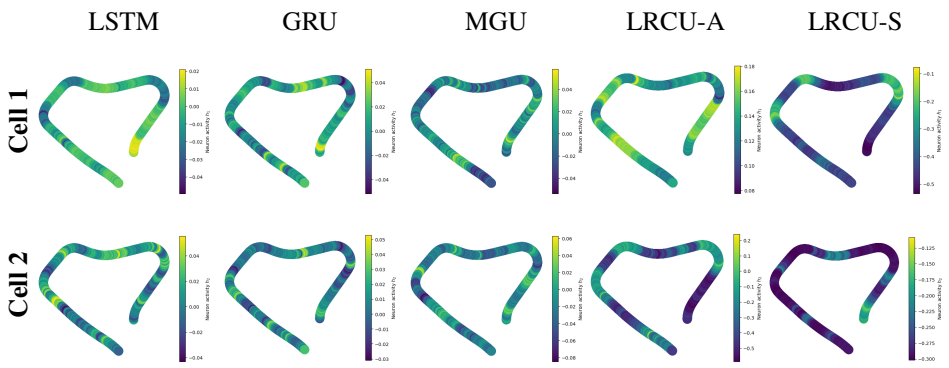


Figure 12: Interpretability. The neural activity of two command cells in the learned policy for the Lane-Keeping Task, projected over time on the 1km road driven in summer, for three popular gated recurrent units (LSTMs, GRUs, and MGUs) and for LRC units (LRCU-A, and LRCU-S). It is very hard to visually match the neural activity of LSTM, GRU, and MGU cells to the traversed road geometry, respectively. However, LRCU-S cells especially, display a very smooth and identifiable pattern during driving: Cell 1 is responsible for turning right, and Cell 2 for turning left.

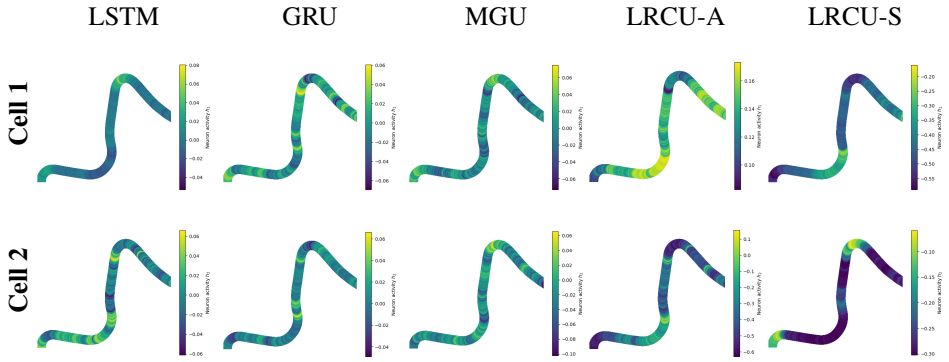


Figure 13: Interpretability. The neural activity of two cells in the learned policy for the Lane-Keeping Task, projected over time on the 1km road driven in winter, for three popular gated recurrent units (LSTMs, GRUs, and MGUs) and for LRC units (LRCU-A, and LRCU-S). It is very hard to visually match the neural activity of LSTM, GRU, and MGU cells to the traversed road geometry, respectively. However, LRCU-S cells especially, display a very smooth and identifiable pattern during driving: Cell 1 is responsible for turning right, and Cell 2 for turning left.