EVALUATING DOMAIN-SHIFT GENERALIZATION OF LIQUID NEURAL NETWORKS IN AUTONOMOUS DRIVING

Mihaela-Larisa Clement*
TU Wien, AIT Austrian Institute of Technology

Mónika Farsang*
TU Wien

Mihai-Teodor Stănușoiu
AIT Austrian Institute of Technology

Daniela Rus, Ramin Hasani
MIT CSAIL, Liquid AI

Radu Grosu, Ezio Bartocci
TU Wien

ABSTRACT

Specialized small models are gaining increasing interest for autonomous driving subtasks such as steering control, where efficient learning and strong generalization are essential. Liquid neural networks have demonstrated promising performance in continuous control, yet their task-learning behavior and cross-domain generalization remain underexplored. In this work, we compare bio-inspired liquid recurrent architectures with gated recurrent networks by training them on an indoor small-scale driving dataset and evaluating their transfer to an outdoor, full-scale driving environment. Liquid models exhibit substantially stronger zero-shot transfer, whereas gated recurrent networks often fail to complete driving episodes without large deviations or crashes. To better understand these differences, we analyze internal representations using saliency-based and manifold learning techniques. Our results show that liquid models learn more task-aligned representations that remain stable across domains, indicating stronger task abstraction capabilities.

1 INTRODUCTION

Autonomous control systems increasingly rely on learning-based models that map raw sensory inputs directly to actions (Singh, 2023; Chen et al., 2024). Although these approaches have achieved impressive performance in structured and well-controlled environments, their reliability under distributional shift remains a fundamental challenge (Wang et al., 2022). In real-world deployment, changes in scale, lighting, textures, and dynamics are inevitable, and analyzing model behavior under such shifts is difficult (Taori et al., 2020; Hendrycks et al., 2021; Sanchez et al., 2023). This has motivated a growing interest in models that are not only performant, but also compact, adaptive, and interpretable (Rudin, 2019; Lechner et al., 2020; Kumar, 2025), allowing principled reasoning about their behavior both within and beyond the training distribution.

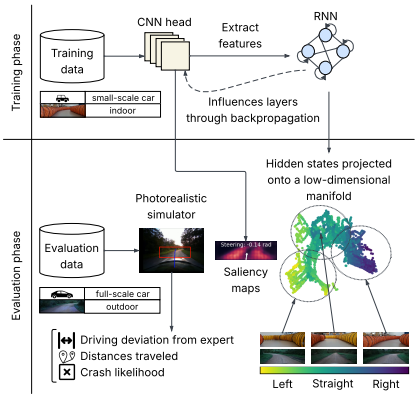


Figure 1: Overview of our training and evaluation phases.

Recurrent neural networks (RNNs) play a central role in control, as they integrate temporal information to support stable decision-making in partially observable environments (Ai et al., 2022; Chahine

*denotes equal contribution
Corr. author: monika.farsang@tuwien.ac.at

et al., 2023; Liu & Diao, 2024). However, commonly used gated architectures, such as LSTMs and GRUs, rely on discrete-time updates and hand-designed gating mechanisms to regulate information flow, implicitly assuming fixed temporal dynamics. In contrast, bio-inspired recurrent architectures, such as liquid neural networks (LNNs), offer a fundamentally different modeling paradigm characterized by continuous-time dynamics with adaptive time-constants (Hasani et al., 2020; 2022). These properties suggest that LNNs may be particularly well-suited for adaptive control under domain shift, yet their transfer behavior in realistic vision-based driving tasks remains largely unexplored. This raises the question of whether recurrent architectures with adaptive, continuous-time dynamics offer a more robust inductive bias for control systems under significant domain shift.

In this paper, we explore how compact, physics-agnostic visual control models trained in structured indoor environments can adapt to considerably more complex outdoor settings. Specifically, we ask whether differences in recurrent dynamics (with an identical initial perception and training protocol) can explain the variations in zero-shot transfer performance, internal-representation structure, and attended visual features under a strong domain shift.

Our models receive RGB input frames and predict continuous steering angles, without an explicit knowledge of the vehicle dynamics or physics. Despite their simplicity, we investigate the ability of these models to generalize across significant environmental shifts from the scale and lighting of indoor tracks to full-scale, physically realistic outdoor driving simulations. Figure 1 shows our setup.

During training, each model (CNN head with RNN) receives indoor driving data, learning to attend to task-relevant cues such as lane surface, boundaries and wall edges. These learned spatial biases shape both the convolutional features and control policies. The trained models are then subjected to zero-shot evaluation in an outdoor simulator that introduces substantial domain shifts, including realistic lighting, new textures, and complex vehicle dynamics.

This setting enables a principled and targeted investigation into how the learned visual features (via the CNN head) and control policy (via the RNNs) generalize to out-of-distribution inputs. We assess whether the models' attention remains focused on semantically meaningful regions and whether control predictions remain coherent and interpretable by analyzing their hidden state dynamics.

To the best of our knowledge, this is the first controlled comparison, evaluating domain transfer capabilities of LNNs compared to gated RNNs in vision-based driving tasks. We empirically demonstrate that the LNN models offer better generalization under domain shift when compared to gated RNNs. These findings highlight the potential of LNNs for the use in adaptive and interpretable control systems.

Our main contributions are as follows:

- Performing the first empirical comparison between liquid neural networks and gated neural networks under domain shift in an autonomous lane-keeping scenario.
- Demonstrating the main advantages of liquid neural networks in terms of interpretability, attention consistency, and generalization under domain shift.
- Integrating methods ranging from control theory and neuroscience to machine learning in order to study interpretability and adaptation under domain shift.

2 BACKGROUND

First, we give a brief overview of liquid neural networks, followed by gated recurrent neural networks.

2.1 LIQUID NEURAL NETWORKS

Throughout this paper, we use the term *liquid neural networks (LNNs)* to denote broad class of models, instantiated either as Liquid Time-Constant networks (LTCs), Closed-form Continuous-time networks (CfCs), or Liquid-Resistance Liquid-Capacitance networks (LRCs). LTCs and LRCs are defined through ordinary differential equations, and CfCs correspond to a closed-form version of LTCs.

Liquid Time-Constant Neural Networks (LTCs) were introduced in (Hasani et al., 2020), to capture the electrical equivalent circuit (EECs) of biological neurons (Wicks et al., 1996), (Kandel et al., 2000). The term *liquid* here refers to their input-and-state-dependent resistive time-constant, which

allows the neuron to adapt its temporal response at each time step, based on its current state and the current input. LTCs are defined as ordinary differential equation (ODE)-based models, with the following form:

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\mathbf{A}, \tag{1}$$

where $\mathbf{x}(t)$ denotes the hidden state, $\mathbf{I}(t)$ is the external input, τ is a fixed intrinsic and resistive time constant, and $f\big(\mathbf{x}(t), \mathbf{I}(t), t, \theta\big)$ governs the resistive liquid-time-constant dynamics. The parameter $\mathbf{A}$ represents the leaking potential. In practice, the parameter set θ includes multiple learnable biologically relevant quantities, including synaptic conductances and chemical channel parameters.

Closed-form Continuous-time Neural Networks (CfCs) introduced in (Hasani et al., 2022), offer a closed-form version of LTCs, with the following equation:

$$\mathbf{x}(t) = \sigma\left(-f(\mathbf{x}, \mathbf{I}; \theta_f)\mathbf{t}\right) \odot g(\mathbf{x}, \mathbf{I}; \theta_g) + \left(1 - \sigma\left(-f(\mathbf{x}, \mathbf{I}; \theta_f)t\right)\right) \odot h(\mathbf{x}, \mathbf{I}; \theta_h), \tag{2}$$

where $\mathbf{x}$ is the hidden state size, $\mathbf{I}(t)$ is the external input as before. The sigmoidal function σ forms $\sigma\left(-f(\mathbf{x}, \mathbf{I}; \theta_f)t\right)$ time-continuous gating terms. Three neural network heads f, g and h share the same backbone, which are parameterized by θ_f, θ_g, and θ_h, respectively. As a consequence of parameterizing the dynamics through learnable parameters θ, these networks partially reduce the explicit biological meaning that was present in the earlier LTC formulation. On the other hand, the CfC form offers a significant advantage over traditional ODE-based models by making time an explicit component of the formulation. As a result, the proposed models is able to achieve speed-ups from one up to five orders of magnitude (Hasani et al., 2022).

Liquid-Resistance Liquid-Capacitance Networks (LRCs) introduced in (Farsang et al., 2024b; 2025), extend the LTCs time constant with an input-and-state-dependent capacitance inspired by neuroscience, instead of a constant capacitance (Severin et al.; Kumar et al., 2023). As a result, each neuron effectively has a double liquid time constant $\frac{1}{RC}$: the original liquid-resistance term from LTCs and a newly introduced liquid-capacitance term that dynamically modulates the neuron's temporal response. This allows the network to adapt both its resistance and capacitance based on the current hidden state $\mathbf{x}(t)$ and input $\mathbf{I}(t)$, enhancing the flexibility and expressivity of the temporal dynamics.

$$c(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\frac{d\mathbf{x}(t)}{dt} = -\sigma\Big[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\Big]\mathbf{x}(t) + \tanh(f(\mathbf{x}(t), \mathbf{I}(t), t, \theta))\mathbf{A} \tag{3}$$

Here, the term $c(\mathbf{x}(t), \mathbf{I}(t), t, \theta)$ represents the *liquid-capacitance* and $f(\cdot)$ the liquid resistance, (as in LTCs). This form preserves the ODE formulation, but it can be solved by cheaper ODE-solvers, compared to LTCs.

2.2 Gated Recurrent Networks

Gated recurrent networks are well-known variants of RNNs that address the vanishing-gradient problem by introducing gated hidden states, which let the model selectively remember, forget, and update information over long sequences. Among the most widely used gated architectures are LSTMs (Hochreiter & Schmidhuber, 1997), GRUs (Cho et al., 2014), and the more lightweight MGUs (Zhou et al., 2016). Details on their model equations are in Appendix A.1. In summary, for LSTMs, the forget gate f_t controls how much information from the previous state is preserved, the input gate i_t regulates the incorporation of new information, and the output gate o_t determines the hidden state. For GRUs, the update gate z_t balances the contribution of the previous and candidate hidden states, while the reset gate r_t controls the influence of the previous hidden state when computing the candidate state. MGUs use a single forget gate f_t to jointly regulate both forgetting and updating.

3 Related Work

Existing work on liquid neural networks and recurrent control has primarily evaluated performance within a fixed domain or under limited perturbations. In contrast, our focus is on isolating how recurrent dynamics themselves (with a jointly trained perception) affect generalization, interpretability and internal representation structure under domain shift.

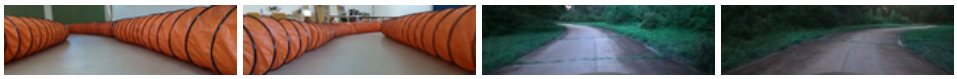


Figure 2: Samples from the indoor dataset (left) and the outdoor datasets (right). While the underlying control task for lane-keeping remains identical, the visual domain differs significantly.

Imitation Learning (IL) trains agents to reproduce the behavior of an expert by learning the mapping between sensory inputs and corresponding actions (Schaal, 1999). Rather than handcrafting explicit reward functions, IL leverages demonstration data to capture task-specific strategies and control patterns implicitly. This paradigm has proven highly effective across a wide range of robotic domains (Fang et al., 2019; Chen et al., 2024; Correia & Alexandre, 2024). Training liquid networks via IL has been demonstrated in autonomous driving (Lechner et al., 2020) and in aerial applications (Vorbach et al., 2021; Chahine et al., 2023). Beyond its broad applicability, IL provides a foundation for developing agents generalizing from limited demonstrations, making it especially valuable in settings where direct reinforcement signals are sparse or costly.

Autonomous Lane-Keeping has been addressed with a range of learned perception–control pipelines. Early approaches employed fully end-to-end convolutional neural networks that map raw camera images directly to steering commands (Bojarski et al., 2016b; Xu et al., 2016). Subsequent work extended this paradigm by augmenting convolutional feature extractors with recurrent layers, enabling the controller to integrate temporal context and smooth decisions over time (Ai et al., 2022). Using liquid networks in this way was explored in (Lechner et al., 2020; Farsang et al., 2024a). In this work, we adopt the latter class of architectures: a convolutional front-end processes the raw visual observations into compact feature representations, which are then passed to a recurrent module that performs history-aware control. This design matches our setting, where robust lane-keeping requires both rich visual understanding and the ability to exploit past observations during decision making.

Cross-Domain Evaluation of LTCs in a sparse model architecture has been shown to capture causal features in drone navigation tasks (Vorbach et al., 2021), where the same environment was used for offline imitation learning and evaluation. Their tests incorporated several environmental perturbations. However, the setup remained largely unchanged in terms of domain (e.g., indoor vs. outdoor) or scale (e.g., different drone sizes). Similarly, (Chahine et al., 2023) evaluated LTCs in fly-to-target tasks that included variations such as target rotations, sensor noise, and range changes. One of their subtasks involved a strong out-of-distribution (OOD) scenario with a modified environment, but that evaluation only reported success rates without deeper analysis. In contrast, our work focuses exclusively on strong OOD settings with changed environment and agent size and provides a more detailed analysis.

4 METHODS

In this section, we describe our experimental setup in indoor and outdoor environments, followed by the evaluation of the approaches we have used with the various models.

4.1 EXPERIMENTAL SETUP

Models. For each model, we used the same CNN head, whose configuration is described in more detail in Appendix A.2. The RNNs use the visual features extracted by the CNN to learn the steering control. During training, the CNN head is optimised jointly with the RNN, meaning that the recurrent component influences the learned convolutional features through gradient propagation. To ensure a fair comparison across models, we used a consistent training setup, which we detailed in Appendix A.2.

Indoor. We formulate an open-loop imitation learning problem using the indoor small-scale car lane-keeping recordings of (Clement et al., 2025), see Fig. 2, left. The dataset consists of input-output trajectories $\{(x_t, y_t) \mid t \in [0..T]\}$, where x_t denotes the front camera image at time t and y_t is the corresponding steering angle. At each time step, the network processes the current frame x_t and predicts a steering angle prediction $\hat{y}_t$. The parameters are optimised by minimising the mean squared error (MSE) between the predicted steering angle $\hat{y}_t$ and the ground-truth label y_t. Training is performed in an offline, open-loop fashion, meaning that the model's predictions do not affect the subsequent inputs, which remain fixed images from the original trajectory.

Translating car dynamics. Before transferring the models trained in the indoor setup to small-scale vehicles, we convert the predicted steering angles into road curvature. While steering angles are

vehicle-dependent, road curvature is a vehicle-invariant quantity, enabling transfer across different platforms. The curvature is computed from the steering angle using the following relation: $y_t = \frac{\tan(\frac{\alpha_t}{S_v})}{L_v}$, where y_t denotes the road curvature, α_t is the steering angle at timestep t, S_v is the steering-ratio, and L_v is the wheelbase of the vehicle.

Outdoor. After training, we evaluate the models in the data-driven VISTA simulation platform (Amini et al., 2022). VISTA can synthesise novel driving trajectories based on real human driving data from an outdoor lane-keeping dataset of (Lechner et al., 2022), see Fig. 2, right. This setup enables evaluation of the models' generalization performance in a closed-loop environment, where predictions at the current time step influence the camera observations at subsequent time steps. At the start of an episode rollout in the simulator, the model hidden states and world are reset. Moreover, the reference trajectories are fixed across models.

4.2 Evaluation Approaches

Driving deviation. To assess stability, we measure lateral deviation between the ego vehicle and the dataset reference (human) trajectory using the simulator's relative pose $(\Delta_t^{lat}, \Delta_t^{lon}, \Delta_t^{\psi})$, where Δ_t^{lat} is lateral displacement in the reference frame. At each timestep, deviation is $d_t = |\Delta_t^{lat}|$, and episode mean deviation is $d = \frac{1}{T}\sum_{t=1}^{T} d_t$. We report deviation both on a truncated horizon (to handle early crashes) and over the full horizon for completed episodes. This closed-loop metric reflects prediction bias, dynamics, and environment feedback.

Crash likelihood. For each episode, a crash is the occurrence of a terminal state violation (out-of-lane or exceeding rotation). The episode-level crash likelihood is the fraction of episodes in which a crash occurred for a controller. This gives a measure of the probability of crashing in a random episode and serves as a critical indicator of reachability. Another view is the per-episode distance distribution, which summarizes how far the ego vehicle travels in each episode under a given controller. For each model, we look at the total path length over the episode and visualize the resulting set of episode distances as distributions. Models that reliably avoid failure score near the maximum attainable distance.

Manifold learning. Manifold learning aims to uncover the low-dimensional structure underlying high-dimensional data. In neuroscience, it provides a way to visualize and analyze the trajectory of neural population activity over time, revealing how complex patterns of activity in the motor cortex or other brain regions evolve to generate behavior. By projecting neural dynamics onto a low-dimensional manifold, researchers can identify latent variables and simplify the interpretation of high-dimensional neural computations.

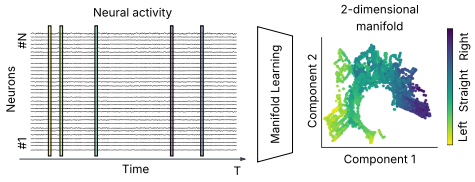
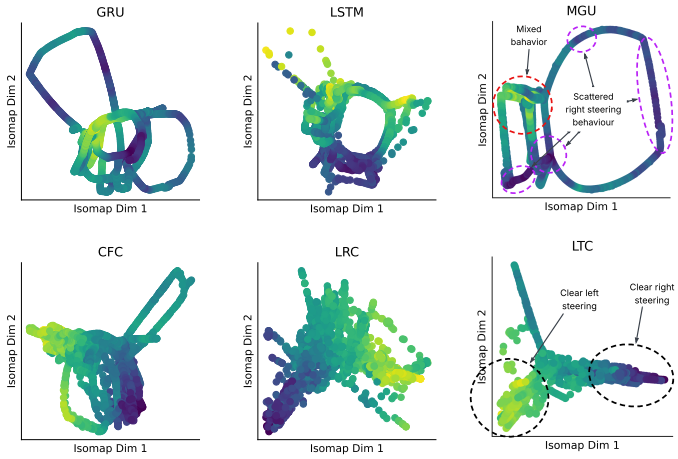


Figure 3: Illustration of manifold learning. The neural population activity $\mathbf{X}$, consisting of recordings from N neurons ($\mathbf{x} \in \mathbb{R}^D$) over T timesteps is projected into a lower-dimensional space with $d << D$. In the illustration $d = 2$ and $D = N$.

The relevance of manifold learning extends to artificial intelligence. Many learned tasks in neural networks involve high-dimensional representations that, in practice, lie on low-dimensional manifolds. Identifying these manifolds allows us to understand how the network organizes its information, how it compresses redundant features, and how it generalizes across inputs. In this sense, *low-dimensional representations* correspond to the essential degrees of freedom that a model uses to *solve a task efficiently*.

Isometric Feature Mapping (Isomap) (Tenenbaum et al., 2000) is a widely used non-linear manifold learning algorithm that uncovers the intrinsic geometry of high-dimensional data. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T] \in \mathbb{R}^{DxT}$ denote the observed neural population activity over time. The goal of Isomap is to compute a low-dimensional embedding $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T] \in \mathbb{R}^{dxT}$ that preserves the intrinsic geometry of the underlying neural manifold $\mathcal{M}$. The full algorithm description is in Appendix A.4. Specifically, in our setup, we used k-nearest neighbors for constructing the neighborhood graph G. For the shortest path, we let the algorithm choose automatically between Floyd-Warshall and Dijkstra's algorithm. To assess how well the high-dimensional $\mathbf{x_i} \in \mathbb{R}^D$ can be embedded in the low-dimensional $\mathbf{y_i} \in \mathbb{R}^d$ manifold, the reconstruction error of an Isomap



Model	Reconstruction Error (↓)
GRU	806.302 ± 403.827
LSTM	265.135 ± 125.523
MGU	643.730 ± 143.630
CFC	99.829 ± 38.199
LRC	57.614 ± 33.581
LTC	$\mathbf{33.241 \pm 23.727}$

Figure 4: Left: Two-dimensional manifold projections of hidden state trajectories $\mathbf{x}$ over 3,000 time steps, color-coded by steering output (yellow: left, green: straight, dark blue: right). Extra labels on the MGU plot show multiple not-interpretable features, while LTC plot labels highlight the desired behavior. Liquid networks form more clearly separated steering regions, whereas gated networks exhibit stronger mixing between turning directions on their manifolds. Right: Corresponding manifold reconstruction errors, with lower errors for liquid networks, indicating better compressibility into more compact and task-relevant latent representations.

embedding can be calculated with the following formula:

$$E = \frac{||K(\mathbf{D_G}) - K(\mathbf{D_Y})||_F}{T}, \tag{4}$$

where $||\cdot||_F$ denotes the Frobenius norm, $\mathbf{D_G}$ the matrix of distances for the input data $\mathbf{X}$, $\mathbf{D_Y}$ the matrix of distances for the output embedding $\mathbf{Y}$, K the Isomap kernel $K(\mathbf{D}) = -0.5 * \mathbf{H} * \mathbf{D}^2 * \mathbf{H}$ converting the distances to inner products (Tenenbaum et al., 2000; Pedregosa et al., 2011). $\mathbf{H} = (\mathbf{I} - 1/T)$ is the centering matrix, and T the number of samples (sequence length for time series).

5 Results

5.1 Zero-shot Performance

In the zero-shot transfer evaluation, we observe substantial differences in generalization capacity across architectures (Table 1). Liquid-based models (CfC, LTC) achieve the strongest performance: CfC yields the lowest steering deviation from the expert trajectory before any model crashed (0.373) and the longest average distance traveled (745.3), with only a 10% crash likelihood. Considering episodes without crashes, LTCs demonstrated the smallest steering deviation from the human trajectory (0.470).

Table 1: Zero-shot performance of the models trained with early stopping, averaged over 20 episodes. The same metrics for all models trained without early stopping are reported in Table 4.

Model	Steering Deviation		Distances Traveled (↑)	Crash Likelihood (↓)
	Until first crash(↓)	Finished episodes(↓)		
GRU	0.856 ± 0.283	N/A	96.4 ± 92.1	100%
LSTM	0.422 ± 0.153	0.544 ± 0.219	102.6 ± 68.4	90%
MGU	0.487 ± 0.230	0.584 ± 0.137	681.4 ± 349.4	35%
CfC	$\mathbf{0.373 \pm 0.233}$	0.592 ± 0.167	$\mathbf{745.3 \pm 339.2}$	**10%**
LRC	0.639 ± 0.237	0.549 ± 0.011	247.6 ± 212.1	90%
LTC	0.419 ± 0.326	$\mathbf{0.470 \pm 0.267}$	435.7 ± 360.9	55%

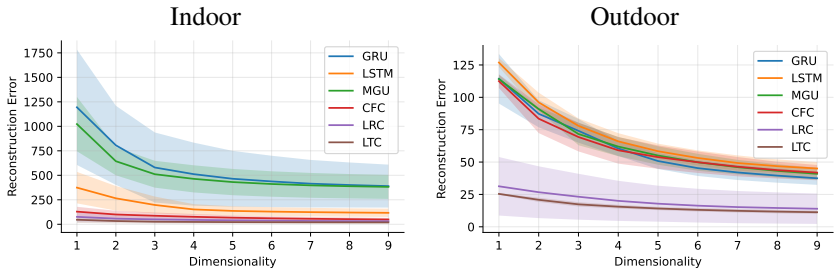


Figure 5: Reconstruction error of RNN hidden-state trajectories over 3,000 test steps as a function of manifold dimensionality, averaged over three seeds (lower is better). **Indoor:** Liquid networks achieve consistently lower reconstruction error across low-dimensional embeddings (1D–9D) compared to gated networks, indicating more compact and task-relevant latent representations. **Outdoor:** This behavior transfers to the outdoor setting, where all liquid models yield lower errors in low-dimensional embeddings (1D–4D). In particular, LRC and LTC achieve approximately 2-4x lower reconstruction error across all manifold dimensions (1D–9D).

5.2 Manifold Learning

With manifold learning, we aim to uncover how models shape their task-specific spaces of hidden states. Ideally, manifolds should be smooth and interpretable, that is, nearby points of the manifold should correspond to similar inputs and behaviors (e.g., camera stream input of a left turn and the corresponding left steering output), with the possibility to interpolate between points without sharp jumps. This means that the steering control output should separate the manifold into distinct regions (Meilă & Zhang, 2024; Freeborn, 2025). Besides these, low reconstruction error also indicates that the model has learned generalizable, meaningful features of the task (Schuster & Krogh, 2021).

The left panel of Fig. 4 shows 2-dimensional manifold projections of the hidden state trajectories $\mathbf{x}$ over 3,000 time steps for different network architectures, color-coded by the corresponding steering output (yellow: left turns, green: driving straight, dark blue: right turns). We observe that liquid networks exhibit more clearly separated clusters in the reduced space according to steering commands, whereas gated networks show a greater degree of mixing, with less distinct separation between turning directions. This qualitative observation is supported by the quantitative results shown on the right, where the low-dimensional manifold reconstruction error is lower for liquid networks, indicating that their hidden states capture task-relevant structure more effectively. Overall, these results suggest that liquid networks learn hidden representations that are more structured and more interpretable in relation to the steering task compared to gated networks.

Another key property of the low-dimensional manifold, beyond visual interpretability, is reconstruction error, which measures how accurately points map back to the original space. Figure 5 shows the reconstruction error of RNN hidden-state trajectories over 3,000 test steps versus manifold dimensionality, averaged over three seeds (lower is better). In both indoor and outdoor settings, liquid networks achieve lower error than gated architecture, especially in low-dimensional embeddings, suggesting more compact, task-relevant latent representations.

The emergence of low-dimensional manifolds is a desirable property for sequential decision-making systems, as it supports interpretability and reduces representational redundancy by capturing the essential task dynamics with a small number of latent factors. Importantly, this property is preserved under domain shift by the liquid models: the low reconstruction errors observed in the indoor environment transfer to the outdoor setting, where they again yield lower errors at low manifold dimensionalities, with LRC and LTC achieving up to a fourfold reduction in reconstruction error across all embeddings compared to gated networks.

More results on manifold learning with different neighborhood setups, and t-SNE (Maaten & Hinton, 2008) and UMAP (McInnes et al., 2018a) projection approaches are in the Appendix A.4.3 and A.4.4, respectively.

5.3 SALIENCY MAPS

Saliency maps are a useful tool for analyzing the learning behavior of models. During training, the CNN head is influenced by the recurrent component through gradient propagation, shaping its features under the influence of the recurrent model. Starting from the same CNN head (using the same seed across models), we investigate how the saliency maps (attention) develop and which regions in these maps have the greatest impact on the predictions. Its methodology and results are discussed in Appendix A.5.

6 DISCUSSION

In this empirical study, we found that liquid neural networks (LNNs) exhibit several advantages over gated recurrent architectures, particularly under domain shift.

In closed-loop control, liquid networks (CfC, LTC) showed the lowest steering deviation from a human driver and successfully completed long trajectories when trained with early stopping, indicating robust cross-domain generalization. When trained without early stopping, LRCs demonstrated strong performance. Liquid networks also learned more consistent cross-domain features: saliency was stable, and LTC focused primarily on the road surface, matching human driving and improving interpretability.

Hidden-state analysis shows liquid networks learn more interpretable representations than gated models. Manifold learning reveals well-separated latent regions aligned with steering commands, with dynamics that are more structured and low-dimensional, especially for LTCs and LRCs, and that remain so under domain shift. This indicates they preserve compact, semantically meaningful internal states even in unseen scenarios.

Among gated models, MGUs achieved competitive closed-loop distance but showed noisy, off-road saliency and less compressible, mixed latent dynamics. GRUs and LSTMs performed poorly across all metrics, with unstable control and inconsistent representations. Overall, gated architectures were less robust and interpretable than liquid networks, which retained structured manifolds and stable attention supporting cross-domain generalization and reliable reasoning.

These differences likely stem from how the architectures model temporal dynamics. Gated RNNs (LSTMs/GRUs/MGUs) use learned gates with fixed update dynamics, so adaptation is implicit in the weights. Liquid neural networks instead use adaptive time constants that directly modulate state evolution, providing a more flexible, task-dependent temporal bias.

Although our study demonstrates the advantages of liquid neural networks, some aspects need further exploration. Our experiments focused exclusively on vision-based steering control, which was intentional to isolate task-specific behaviors and ensure that observed representations and dynamics were not influenced by other tasks or modalities. The architectures were evaluated with particular hyperparameters and training procedures, which means that implementation and tuning choices could further improve performance. We provide a further analysis of the generalization and interpretability results in relation to indoor training performance in Appendix A.3.

Finally, for saliency map computation, VisualBackprop is a simple and computationally efficient method, whereas other gradient-based approaches are known to have limitations. We complement this method with a detailed manifold analysis of the hidden states. This analysis provides a fresh and informative insight into the internal representations. Future work could extend this approach by more rigorously quantifying interpretability and directly linking it to control performance. These considerations provide promising directions for building on our findings and further advance the design of robust, task-adaptive recurrent networks.

7 CONCLUSION

We presented a controlled study of domain-shift generalization in vision-based steering control. Models are first trained in a structured indoor environment and then evaluated in a physically realistic outdoor simulator to assess zero-shot transfer under substantial distribution shift. Our results show that LNNs outperform gated RNNs in closed-loop control under these conditions. Beyond raw

performance, LNNs pay more attention to visual clues and organize their internal states into low-dimensional, task-aligned manifolds. In contrast, gated RNNs develop less robust visual features and less interpretable and compressible internal representations in their manifolds. From a reliability perspective, these findings indicate that representation-level monitoring, such as tracking manifold structure or saliency stability, may offer a practical signal for detecting performance degradation under domain drift. More broadly, our findings indicate that adaptive *liquid dynamics* constitutes a strong inductive bias for compact control policies under distributional mismatch, positioning them as a promising future foundation for interpretable autonomous control systems capable of operating reliably under evolving data distributions.

ACKNOWLEDGEMENTS

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/DOC1345324. M.F. has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101034277. Experiments were performed on the dataLab cluster at TU Wien.

REFERENCES

Ese6150: Roboracer autonomous racing cars. URL `https://roboracer.ai/course`.

Bo Ai, Wei Gao, David Hsu, et al. Deep visual navigation under partial observability. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9439–9446. IEEE, 2022.

Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2419–2426. IEEE, 2022.

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, 2016a.

Mariusz Bojarski, David W. del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *ArXiv*, abs/1604.07316, 2016b.

Makram Chahine, Ramin Hasani, Patrick Kao, Aaron Ray, Ryan Shubert, Mathias Lechner, Alexander Amini, and Daniela Rus. Robust flight navigation out of distribution with liquid neural networks. *Science Robotics*, 8(77):eadc8892, 2023. doi: 10.1126/scirobotics.adc8892.

Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

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.

Mihaela-Larisa Clement, Mónika Farsang, Felix Resch, Mihai-Teodor Stanusoiu, and Radu Grosu. Depth matters: Multimodal rgb-d perception for robust autonomous agents. *arXiv preprint arXiv:2503.16711*, 2025.

Andre Correia and Luis A Alexandre. A survey of demonstration learning. *Robotics and Autonomous Systems*, 182:104812, 2024.

Bin Fang, Shi-Dong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362 – 369, 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)*, pp. 15106–15112. IEEE, 2024a.

Mónika Farsang, Sophie A Neubauer, and Radu Grosu. Liquid resistance liquid capacitance networks. In *The First Workshop on NeuroAI@ NeurIPS2024*, 2024b.

Mónika Farsang, Ramin Hasani, Daniela Rus, and Radu Grosu. Scaling up liquid-resistance liquid-capacitance networks for efficient sequence modeling. *arXiv preprint arXiv:2505.21717*, 2025.

David Peter Wallis Freeborn. Effective theory building and manifold learning. *Synthese*, 205(1):23, 2025.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 4(11):992–1003, November 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00556-7.

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

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8340–8349, 2021.

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.

Haiyang Huang, Yingfan Wang, Cynthia Rudin, and Edward P Browne. Towards a comprehensive evaluation of dimension reduction methods for transcriptomic data visualization. *Communications biology*, 5(1):719, 2022.

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.

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, 2023.

Yash Kumar. Understanding generalization, robustness, and interpretability in low-capacity neural networks. *arXiv preprint arXiv:2507.16278*, 2025.

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.

Yuchen Liu and Shuzhen Diao. An automatic driving trajectory planning approach in complex traffic scenarios based on integrated driver style inference and deep reinforcement learning. *PLoS one*, 19(1):e0297192, 2024.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018a.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018b.

Marina Meilă and Hanyu Zhang. Manifold learning: What, how, and why. *Annual Review of Statistics and Its Application*, 11(Volume 11, 2024):393–417, 2024. ISSN 2326-831X. doi: https://doi.org/10.1146/annurev-statistics-040522-115238.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Jules Sanchez, Jean-Emmanuel Deschaud, and François Goulette. Domain generalization of 3d semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 18077–18087, 2023.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3: 233–242, 1999.

Viktoria Schuster and Anders Krogh. A manifold learning perspective on representation learning: learning decoder and representations without an encoder. *Entropy*, 23(11):1403, 2021.

Daniel Severin, Sofia Shirley, Alfredo Kirkwood, Jorge Golowasch, Jorge Golowasch, and Alfredo Kirkwood. Daily and cell type-specific membrane capacitance changes in mouse cortical neurons.

Apoorv Singh. End-to-end autonomous driving using deep learning: A systematic review. *arXiv preprint arXiv:2311.18636*, 2023.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Charles J. Vorbach, Ramin M. Hasani, Alexander Amini, Mathias Lechner, and Daniela Rus. Causal navigation by continuous-time neural networks. In *Neural Information Processing Systems*, 2021.

Yunkai Wang, Dongkun Zhang, Yuxiang Cui, Zexi Chen, Wei Jing, Junbo Chen, Rong Xiong, and Yue Wang. Domain generalization for vision-based driving trajectory generation. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8950–8956. IEEE, 2022.

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.

Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3530–3538, 2016.

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

A.1 Gated Recurrent Networks

Notation. At each time step t, $x_t \in \mathbb{R}^{d_x}$ denotes the input vector and $h_t \in \mathbb{R}^{d_h}$ the hidden state. For LSTMs, $c_t \in \mathbb{R}^{d_h}$ denotes the cell state. We define the concatenated state $\xi_t = [x_t; h_{t-1}] \in \mathbb{R}^{d_x+d_h}$. Gating variables f_t, i_t, o_t, z_t, and r_t take values in $[0, 1]$ and are computed via affine transformations of ξ_t. Weight matrices $W_\cdot \in \mathbb{R}^{d_h \times (d_x+d_h)}$ and bias terms $b_\cdot \in \mathbb{R}^{d_h}$ are learned parameters. The functions $\sigma(\cdot)$ and $\tanh(\cdot)$ denote the sigmoid and hyperbolic tangent activations, respectively, and $\odot$ denotes element-wise multiplication. For GRUs and MGUs, the candidate hidden state additionally applies the reset or forget gate directly to the previous hidden state prior to concatenation.

LSTM

$$f_t = \sigma(W_f \xi_t + b_f) \tag{5}$$
$$i_t = \sigma(W_i \xi_t + b_i) \tag{6}$$
$$o_t = \sigma(W_o \xi_t + b_o) \tag{7}$$
$$\tilde{c}_t = \tanh(W_c \xi_t + b_c) \tag{8}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{9}$$
$$h_t = o_t \odot \tanh(c_t) \tag{10}$$

GRU

$$z_t = \sigma(W_z \xi_t + b_z) \tag{11}$$
$$r_t = \sigma(W_r \xi_t + b_r) \tag{12}$$
$$\tilde{h}_t = \tanh(W_h [x_t; r_t \odot h_{t-1}] + b_h) \tag{13}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{14}$$

MGU

$$f_t = \sigma(W_f \xi_t + b_f) \tag{15}$$
$$\tilde{h}_t = \tanh(W_h [x_t; f_t \odot h_{t-1}] + b_h) \tag{16}$$
$$h_t = f_t \odot h_{t-1} + (1 - f_t) \odot \tilde{h}_t \tag{17}$$

A.2 Training Details

To ensure a systematic and fair experimental setup, each model shared the same CNN-head setup, which is shown in Table 2. All models closely match the same number of trainable parameters. All recurrent models were configured with 32 units to match the same capacity of their latent representation.

We trained all models on the indoor dataset of Clement et al. (2025) for 30 epochs with batch size of 20 and performed hyperparameter tuning of the learning rates in $\{10^{-3}, 10^{-4}, 10^{-5}\}$ with a decaying schedule and the Adam optimizer. Early stopping was used to prevent overfitting. All models performed best with 10^{-4}. ODE-based models were used with a hybrid solver and 6 unfolds. The validation losses are reported in Table 3.

Experiments were run for training and closed-loop testing on A100 GPUs and for analysis on A40 GPUs. Model training took around 2-3 hours per model per seed, based on early stopping.

A.3 Validation Loss and Its Relation to Generalization and Interpretability

Overfitting of the models to the training data was mitigated during training through the use of early stopping. With respect to underfitting, we observe that validation loss alone is not predictive of downstream performance or representational quality. Models achieving higher validation loss

Table 2: Layer configuration of the convolutional head of the RNN-policy. Settings are adapted from Farsang et al. (2024a).

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
Conv2D	Filters: 48, Kernel size: 3, Stride: 1, Activ.: ReLU
Conv2D	Filters: 64, Kernel size: 3, Stride: 1, Activ.: ReLU
Conv2D	Filters: 64, Kernel size: 3, Stride: 1, Activ.: ReLU
Flatten	-
Dense	Units: 64

can nevertheless exhibit good closed-loop stability and more interpretable internal representations, as observed for LTCs. Conversely, models with lower validation loss may perform poorly when evaluated beyond open-loop prediction, as in the case of GRUs. At the same time, low validation loss can be associated with good closed-loop performance in terms of travelled distance; however, it does not guarantee favourable representational properties, as indicated by our saliency and manifold analyses for MGUs.

Table 3: Mean-squared error (MSE), scaled by $\cdot 10^{-3}$, of the evaluated models with early stopping on the indoor dataset. All models were able to be trained to a comparable error range, indicating stable optimization across architectures.

Model	GRU	LSTM	MGU	CfC	LRC	LTC
Validation MSE $\cdot 10^{-3}$ (↓)	4.7 ± 0.6	6.2 ± 0.9	4.2 ± 0.3	5.7 ± 1.1	8.9 ± 1.2	6.5 ± 0.8

We also evaluated the models without early stopping to examine how well they generalize when trained for longer (30 epochs). This was motivated by research investigating whether extended training can improve generalization (Power et al., 2022). The results are summarized in Table 4. We found that longer training helps reduce steering deviation from the human trajectory but leads to a higher crash likelihood for the gated models. Interestingly, for LRCs, which showed only moderate closed-loop performance with early stopping, longer training improved performance.

Table 4: Zero-shot performance of the models trained without early stopping, averaged over 20 episodes. Compared to the early-stopped models, all models have smaller steering deviation until the first crash and LSTM, LRC, and LTC complete longer distances after being trained longer in the indoor dataset. LRCs achieve this without leaving the lane.

Model	Steering Deviation		Distances Traveled (↑)	Crash Likelihood (↓)
	Until first crash(↓)	Finished episodes(↓)		
GRU	0.532 ± 0.325	N/A	58.9 ± 57.7	100%
LSTM	0.160 ± 0.120	N/A	312.0 ± 210.5	100%
MGU	0.331 ± 0.288	N/A	15.2 ± 21.7	100%
CfC	0.157 ± 0.118	0.499 ± 0.463	528.9 ± 394.1	65%
LRC	$\mathbf{0.093 \pm 0.068}$	$\mathbf{0.453 \pm 0.339}$	$\mathbf{841.7 \pm 250.6}$	**0%**
LTC	0.155 ± 0.104	0.570 ± 0.362	529.0 ± 335.3	70%

We also evaluated all models without early stopping to assess their manifold learning capabilities. We found that extended training has a much larger influence on closed-loop performance than on the internal representations of the hidden states. As shown in Figure 6, the models exhibit trends similar to those observed with early stopping (Figure 5).

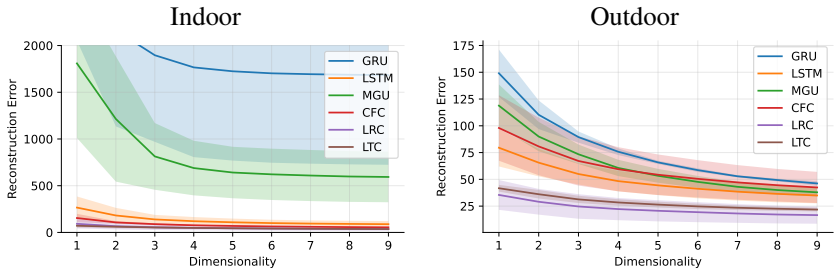


Figure 6: Reconstruction error of RNN hidden-state trajectories over 3,000 test steps as a function of manifold dimensionality, averaged over three seeds (lower is better), for models with longer training on the indoor dataset. Very similar internal representation can be observed with models trained with early stopping in Figure 5, indicating that training has limited influence on their internal representation.

A.4 Manifold Learning

A.4.1 Algorithmic Details

Isomap (Tenenbaum et al., 2000) extends classical MDS by replacing Euclidean distances with geodesic distances along the manifold, making it particularly effective at unfolding non-linear structures. The algorithm produces a low-dimensional representation $\mathbf{Y}$ that faithfully preserves the intrinsic geometry of the high-dimensional data $\mathbf{X}$.

Step 1: Construct the Neighborhood Graph First, local neighborhoods are defined for each data point using either the k-nearest neighbors method or an ε-radius criterion to limit the distance of neighbors:

$$\mathcal{N}_i = \begin{cases} \{\mathbf{x}_j \mid \|\mathbf{x}_i - \mathbf{x}_j\| < \varepsilon\}, & \varepsilon\text{-Isomap} \\ \text{k nearest neighbors of } \mathbf{x}_i, & \text{K-Isomap} \end{cases}$$

These neighborhood relations are encoded in a weighted graph $G = (V, E)$, where nodes correspond to data points, and edges connect neighboring points with weights equal to their pairwise distances:

$$d_G(i,j) = \begin{cases} d_X(i,j) = \|\mathbf{x}_i - \mathbf{x}_j\|, & \mathbf{x}_j \in \mathcal{N}_i \\ \infty, & \text{otherwise} \end{cases}$$

Step 2: Estimate Geodesic Distances Isomap approximates the geodesic distance $d_{\mathcal{M}}(i,j)$ between points $\mathbf{x}_i$ and $\mathbf{x}_j$ as the shortest path distance in the neighborhood graph G:

$$d_{\mathcal{M}}(i,j) \approx d_G(i,j) = \min_{\text{paths } i \to j} \sum_{(u,v) \in \text{path}} d_X(u,v)$$

All-pairs shortest paths can be computed using algorithms such as Dijkstra's or Floyd–Warshall. The resulting distance matrix is denoted as

$$\mathbf{D_G} = \{d_G(i,j)\}_{i,j=1}^{T}.$$

Step 3: Low-Dimensional Embedding via MDS Finally, classical multidimensional scaling (MDS) is applied to $\mathbf{D_G}$ to embed the data into a d-dimensional Euclidean space $\mathbf{Y}$ while preserving the estimated geodesic distances. The embedding minimizes the cost function:

$$\mathbf{y_i} = \arg\min_{\mathbf{y_i} \in \mathbb{R}^d} \|K(\mathbf{D_G}) - \mathbf{y_i}\mathbf{y_i}^\top\|_{L_2}^2, \tag{18}$$

where $K(\mathbf{D_G})$ is the Isomap kernel that converts distances to inner products:

$$K(\mathbf{D_G}) = -\frac{1}{2}\mathbf{H}\mathbf{D_G}^2\mathbf{H}, \quad \mathbf{H} = \mathbf{I} - \frac{1}{T}. \tag{19}$$

Here, $\mathbf{H}$ is the centering matrix and $\mathbf{I}$ is the identity. The solution is obtained by taking the top d eigenvectors of $K(\mathbf{D_G})$ corresponding to the largest eigenvalues:

$$\mathbf{Y} = [\sqrt{\lambda_1}\mathbf{v}_1, \ldots, \sqrt{\lambda_d}\mathbf{v}_d], \tag{20}$$

where λ_p and $\mathbf{v}_p$ are the p-th eigenvalue and eigenvector, respectively.

A.4.2 Our setup

As manifold learning methods rely on nearest-neighbor search, we ensured that all features are scaled the same. For this, we standardized the hidden state trajectories using StandardScaler from scikit-learn following their tips on practical use. We then applied K-Isomap with 5-nearest neighbours and let the algorithm to select the optimal method for both computing the nearest neighbours and calculating the shortest path between them, reducing sensitivity to the choice of these parameters.

A.4.3 More Results on Manifold Learning with Isomap

We evaluated the models across $k = \{5, 10, 15\}$, with $k = 5$ (the default setting) shown in Fig. 4. Result for $k = 10$ and $k = 15$ manifolds are presented in Fig. 7. Across all settings, gated networks consistently exhibited less continuous manifolds, with mixed regions corresponding to different steering directions, whereas liquid networks preserved smoothly varying and more interpretable steering regions.

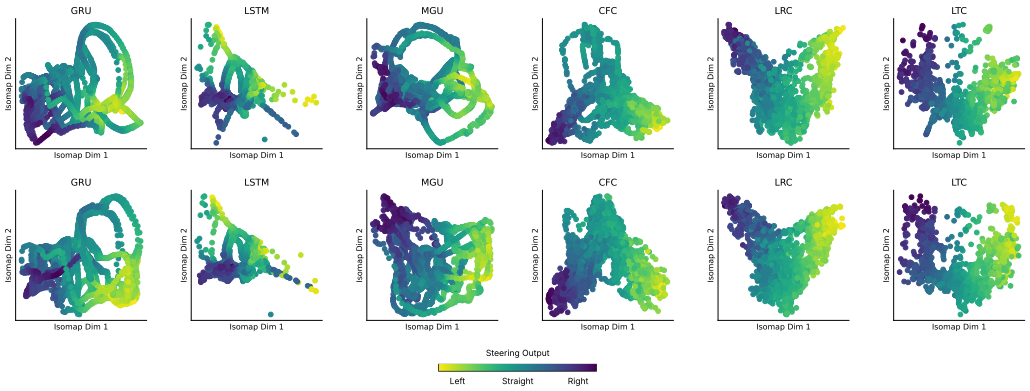
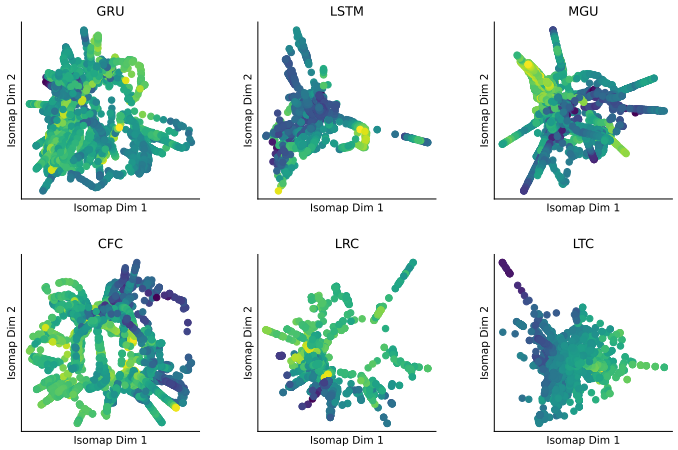


Figure 7: Isomap results with $k = 10$ nearest neighbors are shown on the top row, and results with $k = 15$ nearest neighbors on the bottom row. We found that further increasing the neighborhood size does not lead to substantially different manifolds. For Fig. 4, we kept the default setting of $k = 5$.

Figure 8 shows low-dimensional (2D) manifold projections of the models in the outdoor environment over 3,000 test steps, in the same style as previously presented for the indoor environment. These visualizations illustrate how effectively each model captures the underlying dynamics of the new environment. Liquid networks, particularly LRC and LTC, map their adapted dynamics onto the low-dimensional manifold more accurately than gated models, as reflected in their lower error values.

While the absolute error values differ between the indoor and outdoor environments, this difference arises from dataset characteristics rather than changes in model behavior, as the indoor dataset consists of long, continuous episodes, whereas the outdoor dataset contains shorter trajectories where we reset the hidden states, from which many sequences were sampled to match the same evaluation

length (3,000). Despite this shift in data distribution and temporal structure, liquid networks maintain compact low-dimensional representations, underscoring their suitability for dynamically varying environments.



Model	Reconstruction Error (↓)
GRU	87.285 ± 10.470
LSTM	96.077 ± 7.522
MGU	90.933 ± 1.076
CfC	83.486 ± 11.093
LRC	26.717 ± 20.008
LTC	$\mathbf{20.840 \pm 1.619}$

Figure 8: Low-dimensional (2D) manifold projections of the models for the outdoor environment for 3,000 test steps, similar to Fig. 4. Liquid networks, especially LRC and LTC project their adapted dynamics into the low-dimensional manifold better than the gated models, shown by the reconstruction error, averaged over 3 seeds.

A.4.4 Manifold Learning with Other Methods

Isomap is not the only method for projecting neuron population activity onto a lower-dimensional manifold. To provide a more comprehensive overview, we also applied T-distributed Stochastic Neighbor Embedding (t-SNE) (Maaten & Hinton, 2008; Pedregosa et al., 2011) and Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018a;b) to the analyzed hidden state trajectories. The resulting manifolds display more spread-out clusters of points, which is an inherent property of these techniques and not specific to the hidden state data we are analyzing (McInnes et al., 2018b; Huang et al., 2022).

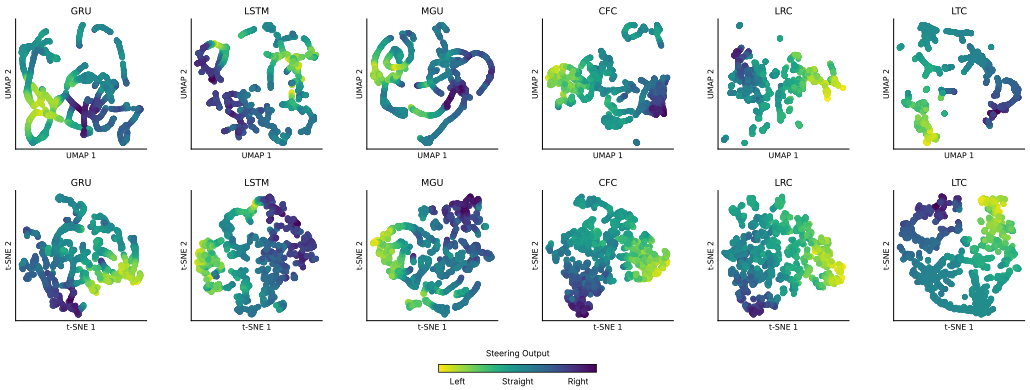


Figure 9: Manifold visualizations using different manifold learning techniques, namely UMAP and t-SNE. Consistent with previous observations using Isomap, embeddings of gated networks exhibit more scattered and less continuous clusters across regions corresponding to different steering angles, whereas liquid networks form more coherent and smoothly changing steering manifolds.

A.5 Saliency Maps

Saliency Maps Understanding how a neural network makes its decisions is crucial for end-to-end autonomous systems. An effective approach is to analyze where the network focuses its attention during decision-making, namely by identifying the regions of the input image that most strongly

influence the output. Despite using the same convolutional head architecture, it is often the case, that distinct attention patterns emerge across models during training.

To visualize network attention, we apply the VisualBackProp algorithm (Bojarski et al., 2016a) to the convolutional heads. This method is based on the fact that task-relevant information is encoded in the deepest feature maps. These feature maps can be then traced back to the input through the network layers.

Algorithm 1 VisualBackprop (Bojarski et al., 2016a)

Require: $A = (A_0, A_1, ..., A_L)$, A_i: feature maps of the i-th layer of the CNN: L: number of CNN layers
Ensure: Saliency map

function VISUALBACKPROP
  $a \leftarrow \text{mean}(A_L)$
  for $l \leftarrow L-1$ to 0 **do**
    $m \leftarrow \text{mean}(A_l)$
    $s \leftarrow \text{conv\_transpose}(a)$
    $a \leftarrow m \cdot s$
  end for
  return a
end function

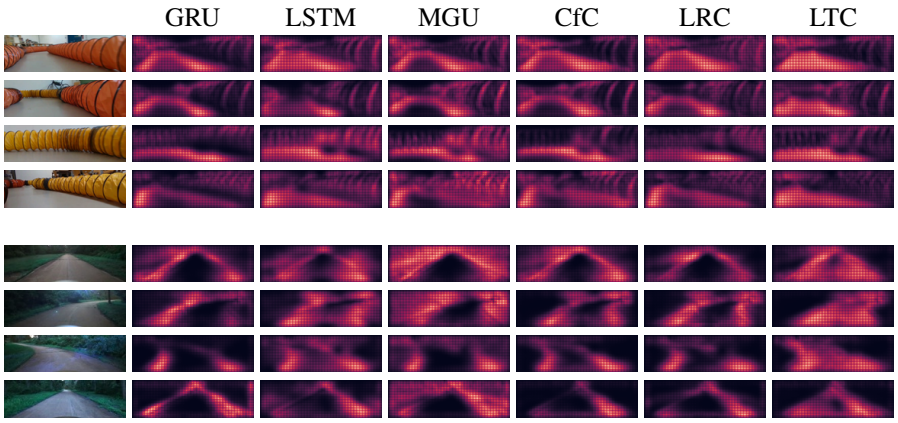


Figure 10: Saliency maps computed via VisualBackprop (Bojarski et al., 2016a) for indoor and outdoor environments. LSTM- and MGU-based models focus more on track infrastructure than the road, resulting in noisier attention in both settings, while LTC consistently concentrates on the road, indicating robust task-relevant feature selection.

Saliency maps for indoor and outdoor environments are presented in Fig. 10. The first four rows analyze indoor features, while the last four rows show outdoor features. In each row, the first frame depicts the input received by the models, and the subsequent columns display the saliency maps for each model. All models were trained exclusively on indoor data and were evaluated both indoors and outdoors to assess how well task-relevant visual features generalize across domains. In the indoor environment, models based on LSTMs and MGUs exhibit increased attention to the indoor track infrastructure (e.g., pipe texture, especially in Row 3-4), rather than focusing primarily on the road boundaries or the drivable surface itself. This behavior carries over to the outdoor setting, where these models display noisier attention patterns and attend to the green side-areas outside of the drivable region, compared to the other models (GRU, CfC, LRC, and LTC). GRUs were able to extract valuable features, but not entirely. The right-side of the image in Row 8 is highlighted, although there are no visual clues. CfCs and LRCs show consistent and similar attention patterns in the two domains. LTCs are the only model that consistently concentrates its attention on the road, suggesting superior feature selectivity and robustness under domain shift.

A.6 Details of Indoor and Outdoor Setups

In the indoor setup (Clement et al., 2025), the vehicle controlled here is a 1/10th scale (RoboRacer (f1t) platform) vehicle measuring a wheelbase of 0.322m, width of 0.28m, driving a constant 0.9 m/s, and a wheelbase. The average track width is 0.546m, with an average length of 33.32m. The commanded Ackermann steering angle represents the front-axle steering angle. We can assume a steering ratio of 1, as the commanded steering and the tire angle are 1:1.

In the outdoor setup (Lechner et al., 2020; Amini et al., 2022), the used car configuration has a wheelbase of 2.912m, width of 1.874m, with different acceleration profiles, measuring an average speed of 11m/s. The available onboard camera details are reported in Table 5. We resized the camera frames in both setups to a width of 160px and a height of 48px to be used for the model training and evaluation. In the simulator, the road width is 6m, and the maximum travelling distance is set to 1,000m. The car dynamics are modelled using a simple continuous kinematic model of a rear-wheel driven vehicle.

Parameter	**Outdoor (camera_front)**	**Indoor Realsense**
Image width (px)	960	848
Image height (px)	600	480
FOV (H × V)	$123^\circ \times 98^\circ$	$69^\circ \times 42^\circ$
f_x (px)	262.242074	423.86032104492188
f_y (px)	261.700119	423.86032104492188
c_x (px)	498.437823	427.46524047851563
c_y (px)	285.741420	239.06629943847656
Intrinsic matrix K	$\begin{bmatrix} 262.242074 & 0 & 498.437823 \\ 0 & 261.700119 & 285.741420 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 423.8603210449 & 0 & 427.4652404785 \\ 0 & 423.8603210449 & 239.0662994385 \\ 0 & 0 & 1 \end{bmatrix}$

Table 5: Comparison of camera parameters for the outdoor and indoor experimental setups. Both camera frames were resized to a width of 160px and a height of 48px to be used for the model training and evaluation.

The mapping described in Section 4.1 normalizes vehicle-specific steering geometries, converting into a vehicle-invariant road path curvature. However, this does not normalize speed-dependent dynamics, tire-road interaction, actuation limits, or perceptual geometry differences such as camera geometry and visual scale, all of which still change across domains.

Using the steering angle to curvature formula from Section 4.1, we present in Table 6 the statistics of the indoor dataset commanded steering angle, in comparison to the outdoor data distribution. This table shows that steering-angle distributions in radians are one order of magnitude smaller outdoors, and converting them to curvature reveals bigger differences due to wheelbase and steering-ratio scaling between the vehicles.

Statistic	**Indoor (1/10 scale)**	**Outdoor (full scale)**
Wheelbase L_v (m)	0.322	2.912
Steering ratio S_v	1.0	14.8
Mean steering μ_α (rad)	-1.23×10^{-2}	9.76×10^{-4}
Std steering σ_α (rad)	1.85×10^{-1}	4.26×10^{-2}
Min steering (rad)	-5.50×10^{-1}	-8.43×10^{-2}
Max steering (rad)	5.50×10^{-1}	8.56×10^{-2}
Mean curvature μ_κ (m^{-1})	-3.82×10^{-2}	2.26×10^{-5}
Std curvature σ_κ (m^{-1})	5.75×10^{-1}	9.88×10^{-4}
Min curvature (m^{-1})	-1.71	-1.95×10^{-3}
Max curvature (m^{-1})	1.71	1.98×10^{-3}

Table 6: Comparison of steering-angle statistics (radians) and the corresponding road-curvature statistics computed with $y_t = \frac{\tan(\frac{\alpha_t}{S_v})}{L_v}$. Indoor values assume direct tire-angle commands ($S_v = 1$); outdoor values use steering ratio $S_v = 14.8$.