# Evaluating Liquid Neural Networks on a Behavioral Cloning Quadrotor Control Problem

Tudor M. Avarvarei[1,2,*], Nicolas Drougard[2], Aurélien Plyer[3] and Mario Cassaro[1]

***Abstract*— This paper evaluates Liquid Neural Networks (LNNs) on an energy-optimal quadrotor control benchmark formulated as a behavioral cloning problem. Several LNN variants, including ordinary differential equation (ODE)-based formulations and closed-form continuous-time (CfC) models, are compared against classical recurrent architectures and a reference multilayer perceptron (MLP). Results show that ODE-based LNNs achieve the strongest closed-loop performance among recurrent models, demonstrating robustness to discretization effects and partial observability while maintaining stable behavior across network sizes. In contrast, CfC variants provide competitive performance under nominal conditions but degrade more significantly under scaling, revealing limitations of its closed-form approximation. While the MLP baseline achieves superior steady-state accuracy and lowest inference latency, it is less reliable under degraded observability and numerical perturbations. Overall, the results emphasize complementary trade-offs between continuous-time recurrent models and feedforward architectures, suggesting that LNNs are particularly well-suited for control tasks involving temporal uncertainty or limited state information.**

## I. INTRODUCTION

The recent emergence of liquid neural networks (LNNs) has opened a new direction in neural architectures, highlighting enhanced adaptability and robustness in dynamic environments [1]. LNNs draw inspiration from neuromorphic computing via brain-inspired leaky integrate-and-fire neurons while operating on continuous-valued signals rather than spikes. Concretely, they extend continuous-time recurrent neural networks (CT-RNNs) by introducing input-dependent time constants [2]. By embedding continuous-time state dynamics, LNNs model time series with stable, bounded behavior and real-time adaptability, while refining representations online [3]. Its authors[1] claim that these networks improve pattern recognition and generalization under non-stationary and noisy data. In practice, these properties yield faster, more robust responses to abrupt changes which is desirable in safety-critical guidance and control [2], [4].

LNNs have already shown promising applications in video recognition and control, as reported in recent studies. They have been predominantly to control applications such as quadrotor flight [5], [6], [7], self-driving vehicles [8], [9] but also other time series prediction problems [3], [4], [10]. However, prior work has primarily focused on application aspects, without providing a comprehensive analysis of the underlying claims or a systematic comparison of LNN variants. To better understand this class of networks, the present work applies to the well-known task of behavioral cloning of a full state feedback flight control system for a quadrotor model. This choice reduces visual perception driven ambiguities and enables a more focused examination of the LNN architectures, clarifying the contribution of each network component to the closed-loop behavior.

From a machine-learning perspective, autonomous quadrotor flight control from multivariate state time series through behavioral cloning has been widely studied and solved with classical neural architectures. Many approaches rely on multi-layer perceptrons (MLPs), emphasizing energy-efficient designs and high-rate control [11], [12], [13]. However, MLP-based policies discard temporal dependencies, which can limit performance in dynamic or partially observable environments. Comparatively fewer works employ time-dependent models. One example is a recurrent neural network (RNN) targeting position and attitude stabilization [14]. As noted, LNNs offer a potential alternative for guidance and control by retaining temporal context within the processing pipeline, potentially improving robustness to disturbances extending the existing algorithms to partial observability problem formulations. Among existing contributions, the work of Ferede et al. [11] stands out for achieving state-of-the-art results with a simple MLP trained on data generated by an energy-optimal control strategy. Their framework serves as a benchmark in the current study.

Therefore, this paper aims to critically evaluate the recently developed LNNs on the autonomous quadrotor control behavioral cloning problem employing an energy-optimal strategy, and providing a comprehensive analysis of both their performance and internal operating mechanisms. In particular, a minimal LNN architecture will be identified that delivers robust, high-performance quadrotor control. The resulting network will be compared against state-of-the-art MLP and RNN approaches to assess the advantages and limitations for state-to-motor quadrotor control.

## II. METHODOLOGY

### A. Dataset Generation

The training dataset was generated as in [11], using an energy-optimal control strategy defined as follows: given a state space $X$ and control space $U$, the objective is to find a control trajectory $\mathbf{u} : [0,T] \to U$ that drives the quadrotor from an initial state $x_0$ to a target state $x_T \in X$ within time $T$, while minimizing the quadratic control effort:

$$E(\mathbf{u},T) = \int_0^T \|\mathbf{u}(t)\|^2 dt. \tag{1}$$

[1]DTIS, ONERA, Université de Toulouse, 31000, Toulouse, France
[2]ISAE-SUPAERO, 31055 Toulouse, France
[3]DTIS, ONERA, Université Paris-Saclay, 91120, Palaiseau, France
*tudor_mihai.avarvarei@onera.fr

The optimization jointly considers the motor commands $\mathbf{u}$ and the total duration $T$, subject to the system dynamics ($f$) and boundary conditions ($x_0$ and $x_T$):

$$\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u}), \quad \mathbf{x}(0) = x_0, \quad \mathbf{x}(T) = x_T. \tag{2}$$

Thus, the dataset consists of optimal single state–control trajectories minimizing a trade-off between control effort and maneuver duration. The state and control vectors are:

$$\mathbf{x} = [p, v, \lambda, \Omega, \omega]^T, \quad \mathbf{u} = [u_1, u_2, u_3, u_4]^T \tag{3}$$

where $p = [x, y, z]$ and $v = [v_x, v_y, v_z]$ are position and velocity while $\Omega = [p, q, r]$ and $\lambda = [\phi, \theta, \psi]$ describe orientation representing the angular velocity and Euler angles, respectively The vector $\omega = [\omega_1, \omega_2, \omega_3, \omega_4]$ collects propeller angular velocities and $M_{ext} = [M_{ext,x}, M_{ext,y}, M_{ext,z}]$ represents disturbance moments capturing salient unmodeled dynamics to increase guidance and control robustness as reported by Ferede et al. [11]. These were sampled uniformly from $[-0.04, 0.04]$ for $M_{ext,x}, M_{ext,y}$ and $[-0.01, 0.01]$ for $M_{ext,z}$. The control inputs $\mathbf{u}$ correspond to target revolutions per minute (RPM) commands, normalized to $[0, 1]$. The internal drone dynamics as well as the equivalent dynamical model constants for the Parrot Bebop 1 quadrotor were adopted from the works of Ferede et al.[11].

The dataset comprises optimal trajectories with randomized initial states $x_0$ and terminal hover states $x_T$ under the conditions: $p(T), v(T), \lambda(T), \Omega(T), \dot{v}(T), \dot{\Omega}(T), \dot{\omega}(T) = 0$. Trajectories $\mathbf{x}(t), \mathbf{u}(t)$ were discretized with timestep $\Delta t = T/N$ where $N = 200$. As time of the simulation $T$ is optimized, the final value differs in between trajectories, leading also to timestep $\Delta t$ to vary inside dataset. The initial states were sampled uniformly over:

$$\begin{cases} x \in [-5,5], \;\; y \in [-5,5], \;\; z \in [-1,1], \\ v_x \in [-0.5,0.5], \;\; v_y \in [-0.5,0.5], \;\; v_z \in [-0.5,0.5], \\ \phi \in \left[\frac{-2\pi}{9}, \frac{2\pi}{9}\right], \;\; \theta \in \left[\frac{-2\pi}{9}, \frac{2\pi}{9}\right], \;\; \psi \in [-\pi, \pi], \\ p \in [-1,1], \;\; q \in [-1,1], \;\; r \in [-1,1], \\ \omega \in [\omega_{min}, \omega_{max}]^4. \end{cases} \tag{4}$$

### B. Liquid Neural Network

The LNNs [1], [15], [2] extend CT-RNNs with biologically inspired, input-adaptive time constants, closely related to a membrane integrator enhancing thus temporal modeling and adaptation. Its neuron dynamics are:

$$\frac{dv(t)}{dt} = -\left[\frac{1}{\tau} + f\big(v(t), I(t), t, \theta\big)\right] v(t) + f\big(v(t), I(t), t, \theta\big) A \tag{5}$$

where $v(t)$ represents the neuron state, $\mathbf{I}(t)$ is the input, $A$ is a learnable parameter, $\tau$ is a time constant and the function $f(\cdot)$ is a neural function parametrized by $\theta$. The resulting system time constant $\tau_{sys}$, characterizing the speed and coupling sensitivity of the ODE, becomes input-dependent $\tau_{sys} = \frac{\tau}{1+\tau f(v(t), I(t), t, \theta)}$, enabling the hidden state to identify dynamics tailored for input features arriving at each time point. This adaptable timescale underpins the real-time versatility discussed in the introduction and is particularly relevant for closed-loop control under time-varying disturbances.

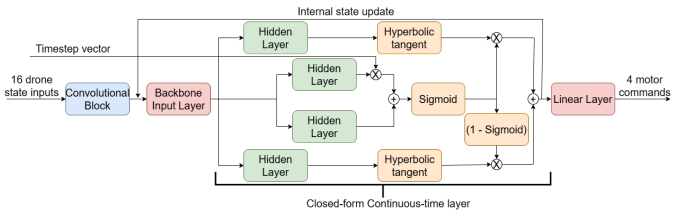


Fig. 1. The schematic describing the used gated CfC network composed of convolutional block, input and output mapping layers, including the CfC layer dynamics[15].

Solving the internal neuron dynamics requires an ordinary differential equation (ODE) step. For this reason, training LNN leverages a vanilla Backpropagation Through Time (BPTT) scheme that directly backpropagates through the chosen ODE solver outputs (the sequence of neural states), effectively yielding an RNN unroll for several times. The preset value for the number of unrolls suggested by the authors [1] is of 6. This network formulation will be referred as ODE-based LNN.

However, numerically integrating LNNs with input-adaptive ODEs remains computationally demanding and prone to numerical challenges. To mitigate this, Hasani et al. [15] derive a specialized closed-form update by exploiting analytic solutions for specific nonlinearities. In the case of a single neuron driven by a one-dimensional time series input and without self-connections, solving the LNN dynamics yields a closed-form expression for the neuron state defined by the corresponding initial value problem where $v_0$ represents the initial neuron state:

$$v(t) = (v_0 - A) \cdot e^{-[\frac{1}{\tau} + f(I(t),\theta)]\cdot t} \cdot f(-I(t), \theta) + A. \tag{6}$$

This derivation led to the closed-form continuous-time (CfC) network (this network formulation will be referred as analytical CfC). Building on the scalar closed-form solution above, Hasani et al. [15] extended the model to a trainable, scalable neural architecture that mitigates gradient-instability and expressivity limits encountered during optimization. The resulting architecture is shown in Fig. 1 (this network formulation will be referred as gated CfC).

Moreover, CfCs can operate under sparse connectivity, enabling closer emulation of biological synaptic wiring as in Neural Circuit Policies (NCPs) [8]. Unlike standard dense networks, the connectivity between neurons follows a biologically inspired wiring scheme, often hand-crafted or evolved, rather than being freely learned. They claim such sparsity enhances interpretability and should reduce computational load. Inspired by Lechner et al. [8], the NCP used in current work comprises three layers, each instantiated as a closed-form continuous-time layer as shown in Fig. 1.

For all four architectures, a linear input projection maps the raw features to the network width (number of neurons) and a linear readout maps hidden states to outputs (motor commands). Among the LNN variants considered, the analytical CfC, gated CfC, and NCP models process the inputs differently. They fuse the raw features with the liquid

block's internal hidden state via a so-called backbone layer (formed of 128 units unless stated otherwise). Upstream, a 1D convolutional block extracts features from the input space to improve conditioning and sample efficiency. This follows Lechner et al. [8], who employ a convolutional front end for image processing while keeping the core unchanged. This technique aligns with Jana et al. [10] who used an MLP autoencoder for preprocessing the input time series. However, a 1D convolutional block was adopted, as it consistently provided stable performance due to its efficient extraction capabilities of channel-local patterns from the ordered input while using fewer parameters than an equivalent MLP.

### C. Experiments

For learning, 100,000 trajectories of length $N = 200$ timesteps were generated, each consisting of state-motor command pairs. The dataset was split into 90%, 9% and 1% respectively for training, validation and testing. Inputs and outputs were normalized min-max to $[0,1]$ before being passed to the network. Optimization used the ADAM algorithm with a learning rate of $1 \times 10^{-3}$. An $\ell_2$ regularization term (weight decay of $1 \times 10^{-6}$) was included to improve generalization [16]. To promote robustness, mild stochastic perturbations were injected during training: a subset of trajectories received zero-mean noise on randomly selected model parameters. Noise levels were bounded to remain realistic (standard deviation of $1 \times 10^{-3}$) and were applied independently per timestep using a combination of normal, Laplace, Poisson and uniform distributions. All these procedural steps were necessary to enable reliable evaluation and avoid overfitting.[1]

The objective function was the mean squared error (MSE):

$$\text{MSE} = \frac{1}{4}\sum_{i=1}^{4}(u_i - \hat{u}_i)^2 \tag{7}$$

where $u_i$ denotes the target motor command and $\hat{u}_i$ the network prediction. Energy consumption was estimated by discrete integration of the motor commands over time. Simulations run for $T = 6s$ and used explicit Euler integration with timestep $\Delta t = 0.01s$, close to the dataset's mean interval. A run is considered a failure if, at termination, the quadrotor exceeds any of the following thresholds relative to the target point as stated in Section II-A: Euclidean distance of $0.25m$, Euclidean velocity of $0.25m/s$ and attitude error, based on pitch and roll angle, of $0.3rad$. Failure metrics are reported using the **Randomized Start Test**, which represents the most challenging evaluation setting. To assess performance, 3 types of simulation experiments were conducted:

- **Prediction Accuracy Test:** Direct comparison of predicted outputs with ground-truth motor commands was done, evaluating how closely the network reproduces the reference signals. Two metrics are reported: (i) the mean MSE over the test set (**MSE Loss [−]**), and (ii) the mean of the neural network inference runtime obtained across 200 timesteps over all test trajectories (**Inference [$\mu s$]**). Inference was measured on an Intel Core i5-1345U (13th Gen), 10 cores, 12 threads, max clock 1.6 GHz, with 16.8 GB RAM.
- **Direct Dataset Comparison Test:** Simulations were initialized with dataset-based initial drone state conditions. An initial warm-up phase was introduced in this test to ensure stability and to mitigate transients arising from random initial hidden states in recurrent networks. However, subsequent experiments revealed that the drone was capable of stable flight without this phase. Furthermore, it was observed that the inclusion of the warm-up phase negatively impacted energy performance. This degradation is primarily attributed to the transition between dataset-based control and neural network control, which consistently introduced abrupt changes in the generated commands. These discontinuities led to disturbances in the drone's dynamics and overall control behavior. The absolute difference in energy consumption between ideal and predicted trajectories was then computed over the test set. The mean of these values is reported as "Mean Energy Difference" (**MED [−]**). This test ensures a fair comparison between recurrent networks and MLPs and serves as a stability check before the randomized start test. In practice, it could be emulated by a classical controller or an MLP that can reproduce the first few ideal timesteps.
- **Randomized Start Test:** Agents were initialized from pseudo-random states drawn from the same intervals as in (4). Initial conditions were held identical across models, enabling computation of mean and standard deviation over repeated trials. From each initial state, the neural network agent navigated directly to the target. Energy consumption was evaluated over trajectories from 1,000 start points. The mean across these trials is reported as "Mean Energy Consumption" (**MEC [−]**).

## III. RESULTS AND DISCUSSION

### A. Liquid Neural Network Model Refinement

This section analyses only liquid neural networks and their variants (gated CfC, analytical CfC, ODE-based LNN and NCP) under identical training and evaluation protocols. Unless stated otherwise, the liquid core width in all the models is formed of 64 units and all models had no failures during simulation tests. For all the network variants, the convolutional front end was essential for extracting features from the multivariate input. A stable convolutional block, that achieved constant stability throughout the tests, was formed of 4 consecutive layers (of 64, 128, 128 and 256 out channels respectively) and of 2 batch normalization layers after the second and fourth layers. Performance depended strongly on input ordering as presented in Equation 3: structuring channels by axis or semantic group enabled the convolutions to capture local correlations before cross-group fusion, which simplified feature extraction and improved downstream LNN behavior. The performance of the variants is reported in

[1] https://github.com/TudorAvarvarei/LNN_behavioural_cloning_quadrotor

TABLE I

COMPARISON OF BENCHMARK MODELS; VALUE IS PRESENTED AS "MEAN(STANDARD DEVIATION)"

| Network | Gated CfC | Analytical CfC | ODE LNN | NCP | RNN | CT-RNN | GRU | LSTM | MLP |
|---|---|---|---|---|---|---|---|---|---|
| **MSE Loss** [$\times 10^{-4}$] | 1.542(1.931) | 1.979(1.245) | 1.813(2.344) | 1.344(1.647) | 3.984(2.368) | 1.657(2.028) | 0.882(0.888) | **0.847**(0.983) | 2.508(3.257) |
| **MED** [−] | 0.902(0.666) | 1.043(0.709) | 0.712(0.756) | 0.968(0.712) | 1.174(0.624) | 0.774(0.690) | 1.642(1.032) | 1.366(1.144) | **0.232**(0.264) |
| **MEC** [−] | 5.071(1.103) | 5.237(1.089) | 4.770(1.087) | 5.086(1.138) | 5.332(1.056) | 4.836(1.031) | 6.101(1.327) | 5.575(1.427) | **4.570**(0.894) |
| **Inference** [$\mu s$] | 559 | 508 | 1036 | 604 | 439 | 493 | 639 | 710 | **121** |

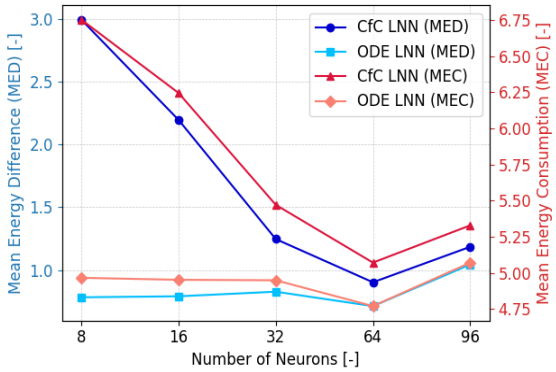


Fig. 2. Effect of neuron counts in liquid layer on performance showcasing the optimum number of neurons is around 64.

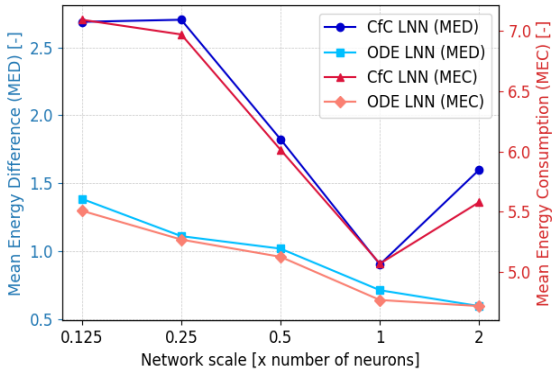


Fig. 3. Effect of network scale factor on performance indicating the difference between gated and ODE-based LNNs performance

Table I along with a visual simulation of the LNN controller[2].

Across liquid variants, lowest energy metrics (MED, MEC) are obtained by the ODE-based LNN at the cost of largest inference time. This behavior is expected, as ODE unfolds require multiple evaluations of the core dynamics per step but appear to favor closed-loop performance. With its high inference time, the LNN may be challenging to deploy for real-time onboard quadrotor control. However, as shown later, its performance remains stable under substantial reductions in network size and ODE unrolls, which can significantly reduce runtime and enable real-time hardware deployment. Among the remaining models, the simplest architecture (analytical CfC) achieved the fastest inference yet performed worst in simulation under unseen conditions, consistent with its reduced expressivity. The NCP configuration used throughout the analysis followed Lechner et al. [8]: three layers with 36, 24, and 4 neurons, using approximately 37% and 80% synaptic connectivity between the 3 layers, respectively. Comparing gated CfC to NCP (that share similar computational logic but arranged as a single-layer versus a hierarchical three-layer topology) yields similar simulation performance, suggesting that the number of overall internal synapses drives performance more than the architectural arrangement itself. This overhead stems from the execution environment, which still treats cut synapses as computational variables. Hardware that naturally exploits sparsity, such as neuromorphic accelerators, may offer a means to reduce inference [17]. Thus, stacking multiple gated CfC layers did not offer clear benefits, suggesting that increased depth does not translate into improved closed-loop behavior for this task. For the remainder of the study, only the two top-performing architectures, namely the gated CfC and the ODE-based LNN, are retained for detailed evaluation.

Fig. 2 shows gated CfC performs best with a single 64 neurons layer. Wider configurations increase MED and MEC, suggesting overfitting, whereas narrower ones ($< 32$ neurons) degrade significantly all metrics. In contrast, the ODE-based LNN exhibits minimal sensitivity to the number of neurons, maintaining nearly constant performance with only a slight degradation at 96 units. This finding is significant for control applications, as it demonstrates that stable quadrotor control can be achieved with a compact eight-neuron model, thereby reinforcing the interpretability advantage of LNNs, as fewer units facilitate clearer attribution of functional roles. Notably, under these conditions, the gated CfC does not represent a strong approximation of the ODE-based LNN.

Each network, including the feature extractor as well as the mapping layers, was uniformly width-scaled while maintaining the relative layer-width ratios. As illustrated in Fig. 3, the gated CfC displays a sharp performance degradation under moderate uniform scaling ($\times 0.5/\times 2$), whereas further compression yields comparative penalties. At one eighth of the network, the failure rate reaches 40.8% which still remains substantially lower than under neuron-count reduction (see performance of 8 neurons in Fig 2) which achieves a failure rate of 72.9%. Hence, for large model compressions, uniform whole-network scaling is preferable, while for small adjustments where performance preservation is critical, targeted neuron-count tuning is more effective. For the ODE-based LNN, downscaling the network width leads to limited degradation, even under substantial scaling (e.g., to one-eighth of the original size). The controller remains

[2]https://youtube.com/shorts/YC2ngGuIwJQ?feature=share

TABLE II

ANALYSIS OF THE NUMBER OF ODE UNROLLS EFFECT ON ODE-BASED LNN PERFORMANCE; VALUE IS PRESENTED AS "MEAN(STANDARD DEVIATION)"

| ODE Unrolls | 1 | 3 | 6 | 10 |
|---|---|---|---|---|
| **MSE Loss** [$\times 10^{-4}$] | 1.680(2.191) | 1.946(2.233) | 1.813(2.344) | 1.908(2.048) |
| **MED** [$-$] | 0.698(0.596) | 0.842(0.677) | 0.712(0.756) | 0.849(0.644) |
| **MEC** [$-$] | 4.949(1.130) | 4.883(1.130) | 4.770(1.087) | 4.998(1.115) |
| **Inference** [$\mu s$] | 528 | 727 | 1036 | 1444 |

functional, with a failure rate of only 11.7%, exhibiting greater sensitivity to global width scaling compared to the gated CfC. Conversely, increasing the network width yields only marginal improvements, with performance saturating close to the baseline configuration. The behavior stems from its highly parameterized neuron structure and continuous-time latent dynamics. Increasing the width expands both the dimensionality of the latent dynamical system and the number of internal synaptic interactions, allowing temporal information to be distributed across richer dynamical states. This makes the ODE-LNN less sensitive to width changes than CfC. Moreover, the ODE-LNN encodes temporal information directly through continuous hidden-state evolution, whereas CfC relies on a gated closed-form update, giving the ODE-LNN greater expressive benefit from additional hidden units.

To assess the effect of integration depth, we vary the number of ODE unrolls used during training of the LNN model. The results, summarized in Table II, illustrate that the performance remains relatively stable across different numbers of unrolls, indicating that this hyperparameter is not critical for the present task. This limited sensitivity may be attributed to the relatively large training dataset, which reduces the need for deeper integration during training. Overall, while some configurations slightly favor lower MSE (e.g., 1 unroll), others achieve better energy consistency (e.g., 6 unrolls), suggesting a balance between fitting the data and capturing the underlying dynamics. In particular, 6 unrolls provide a good compromise across metrics, in line with recommendations from the authors of LNN [1]. However, this parameter could become more relevant in real-time deployment, where inference cost is critical thus smaller values being preferable for onboard control applications.

### *B. Benchmark Comparison with Literature*

This section compares control performance across alternative network architectures, followed up by a similar analysis against the baseline MLP. To enable a fair assessment against the best LNN variants, all models were trained with 64 neurons (hidden neurons) including a similar convolutional block as a feature extractor (with the exception of the baseline MLP model).

- **Recurrent Neural Networks (RNNs)**
- **Continuous-Time Recurrent Neural Networks (CT-RNNs, [18])**
- **Long Short-Term Memory (LSTM, [19]) networks**
- **Gated Recurrent Units (GRUs, [20])**
- **Multi-Layer Perceptron (MLP) networks:** Baseline model with three fully connected layers of 120 units, optimized for the current task by Ferede et al. [11].

According to Table I, the ODE-based LNN still achieves the best overall performance among recurrent architectures. CT-RNNs followed closely, supporting the initial claims that input-adaptive time constants provide an advantage for handling time-varying dynamics. Moreover, relative to other architectures, the networks' continuous-time processing appears to be the main driver of performance. The gated CfC, which blends GRU-like gating mechanisms with CT-RNN architecture in a closed-form manner, ranked next. Architectures emphasizing long-term memory (GRU, LSTM) underperformed on the current problem, suggesting that persistent memory is not the limiting factor for the present guidance and control setting. However, despite their great accuracy, LNN-inspired models exhibited the largest inference runtimes, which may constrain real-time deployment; nevertheless, the method can still be effectively scaled.

A dedicated robustness study was conducted to assess the behavior of the trained policies under measurement and initialization biases, as well as under varying numerical integration schemes and step sizes. As expected, both networks exhibit gradual performance degradation as perturbations increased, ultimately failing beyond a threshold. The ODE-based LNN consistently demonstrated lower robustness than the MLP, failing at smaller perturbation thresholds. For example, under a bias in pitch rate $p$, the ODE-based LNN begins to exhibit failures with a bias value of 0.015 whereas the MLP remains failure-free up to a bias of 0.042. When bias terms are included in the ODE-based LNN, systematic sensor biases can be integrated into the neurons' internal hidden-state dynamics. Because these biases are persistently injected into the hidden-state evolution, the latent trajectory may gradually shift, causing biased control actions to accumulate over time. By contrast, the MLP processes each observation independently, so measurement biases affect each action locally rather than being propagated through recurrent dynamics.

Under initialization bias, the networks were initialized outside the range encountered during training. Thus the simulations were started in the region $x, y \in [-8,8] \setminus [-6,6]$ and $z \in [-3,3] \setminus [-1,1]$. The MLP proves more resilient, failing in only 3 out of 1000 missions compared with 46 out of 1000 for the ODE-based LNN. In contrast, the ODE-based LNN demonstrates superior robustness, against numerical integration step and solver variation. For example, under Euler explicit integration, LNN managed to keep stable operation up to $\Delta t = 0.044s$, whereas the MLP became unstable beyond $\Delta t = 0.031s$.

Additionally, an ablation study was conducted to evaluate the partial observability claims of LNNs. The results shown in Fig. 4 indicate that ODE-based LNN, CT-RNN and CfC generally outperform the MLP baseline, particularly when rate information (velocity and angular velocity) is not

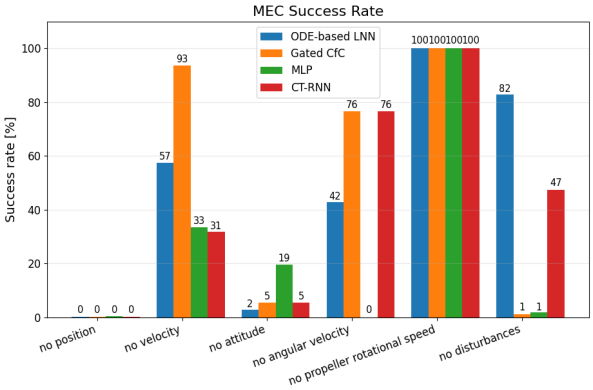


Fig. 4. Ablation study performed on the MLP, CfC default and ODE-based LNN networks with respect to the groups of parameters.

available. This suggests that the recurrent models internally capture action history through their state dynamics, implicitly learning to estimate velocities from position and attitude history, while the MLP relies more directly on instantaneous state-to-action mappings. On the other hand, position and attitude are essential for all models, as their removal significantly reduces performance across both metrics. Interestingly, propeller rotational speed has negligible impact. Finally, in the absence of disturbance information, the ODE-based formulation provides improved robustness, resulting in better adaptability to unseen conditions relative to other recurrent architectures.

## IV. CONCLUSION

This work evaluated LNNs on an energy-optimal quadrotor control problem within a behavioral cloning framework, with a focus on their behavior under varying architectural choices and operating conditions. Among the recurrent models, the ODE-based LNN, together with CT-RNN, consistently achieved the best closed-loop results, combining favorable performance with robustness to time discretization and partial observability. These properties support the view that continuous-time formulations provide a tangible advantage in handling time-varying inputs. Within the LNN family, the ODE-based formulation showed the highest robustness and stability across network scales, remaining effective even in compact configurations down to eight neurons. By contrast, closed-form variants, exhibited sharper degradation under width scaling, highlighting limitations in their approximation of continuous dynamics, although they remain attractive when computational resources are limited.

Compared to recurrent approaches, the MLP baseline achieved superior steady-state accuracy and computational efficiency and was more resilient to initialization and measurement biases. However, its performance degraded more rapidly under partial observability and increased numerical integration timesteps, whereas the ODE-based LNN remained more robust due to its continuous-time recurrent formulation. Overall, the results indicate that MLPs are well suited to the behavioral cloning problem when the full drone state is available, while LNNs become more advantageous under hindered observability and more uncertain dynamic conditions. These complementary characteristics highlight LNNs as a promising direction for safety-critical control applications.

## ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support provided by the Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France, through a dedicated PhD fellowship.

## REFERENCES

[1] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, 'Liquid time-constant networks,' in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 35, pp. 7657–7666, 2021.

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

[3] M. Bidollahkhany, F. Atasoy, and H. Abdellatef, 'LTC-SE: Expanding the potential of liquid time-constant neural networks for scalable AI and embedded systems,' *arXiv preprint* arXiv:2304.08691, 2023.

[4] O. Ayoub *et al.*, 'Liquid neural network-based adaptive learning vs. incremental learning for link load prediction amid concept drift due to network failures,' *arXiv preprint* arXiv:2404.05304, 2024.

[5] C. J. Vorbach, R. Hasani, A. Amini, M. Lechner, and D. Rus, 'Causal navigation by continuous-time neural networks,' in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021.

[6] M. Chahine *et al.*, 'Robust flight navigation out of distribution with liquid neural networks,' *Sci. Robot.*, vol. 8, no. 77, Art. no. eadc7892, 2023.

[7] A. Quach, M. Chahine, A. Amini, R. Hasani, and D. Rus, 'Gaussian splatting to real-world flight navigation transfer with liquid networks,' *arXiv preprint* arXiv:2406.15149, 2024.

[8] M. Lechner *et al.*, 'Neural circuit policies enabling auditable autonomy,' *Nat. Mach. Intell.*, vol. 2, pp. 642–652, 2020.

[9] V. P. C. R. Udumula, R. Ancha, M. K. Ramayanam, P. V. S. M. Teja, and V. Rachpudi, 'An enhanced real-time object detection method using liquid neural network and echo state network architecture,' in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Lalitpur, Nepal, pp. 669–677, 2024.

[10] S. Jana, M. S. Sinha, and T. Dasgupta, 'StayLTC: A cost-effective multimodal framework for hospital length of stay forecasting,' *arXiv preprint* arXiv:2504.05691, 2025.

[11] R. Ferede, G. C. H. E. de Croon, C. De Wagter, and D. Izzo, 'End-to-end neural network based quadcopter control,' *Robot. Auton. Syst.*, vol. 172, 2024.

[12] R. Ferede, C. De Wagter, D. Izzo, and G. C. H. E. de Croon, 'End-to-end reinforcement learning for time-optimal quadcopter flight,' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Yokohama, Japan, pp. 6172–6177, 2024.

[13] P. Varshney, G. Nagar, and I. Saha, 'DeepControl: Energy-efficient control of a quadrotor using a deep neural network,' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Macau, China, pp. 43–50, 2019.

[14] S. C. Yogi, V. K. Tripathi, and L. Behera, 'Adaptive integral sliding mode control using fully connected recurrent neural network for position and attitude control of quadrotor,' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5595–5609, Dec. 2021.

[15] R. Hasani *et al.*, 'Closed-form continuous-time neural networks,' *Nat. Mach. Intell.*, vol. 4, no. 11, pp. 992–1003, 2022.

[16] S. Scardapane, 'Alice's adventures in a differentiable wonderland–Volume I, a tour of the land,' *arXiv preprint* arXiv:2404.17625, 2025.

[17] W. A. Pawlak, M. Isik, D. Le, and I. C. Dikmen, 'Exploring liquid neural networks on Loihi-2,' *arXiv preprint* arXiv:2407.20590, 2024.

[18] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, 'Neural ordinary differential equations,' *arXiv preprint* arXiv:1806.07366, 2019.

[19] S. Hochreiter and J. Schmidhuber, 'Long short-term memory,' *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, 'Empirical evaluation of gated recurrent neural networks on sequence modeling,' *arXiv preprint* arXiv:1412.3555, 2014.