Closed-form Continuous-Depth Models

Ramin Hasani 1,*, Mathias Lechner 2,*, Alexander Amini 1, Lucas Liebenwein 1, Max Tschaikowski 3, Gerald Teschl 4, Daniela Rus 1

Abstract

Continuous-depth neural models, where the derivative of the model’s hidden state is defined by a neural network, have enabled strong sequential data processing capabilities. However, these models rely on advanced numerical differential equation (DE) solvers resulting in a significant overhead both in terms of computational cost and model complexity. In this paper, we present a new family of models, termed Closed-form Continuous-depth (CfC) networks, that are simple to describe and at least one order of magnitude faster while exhibiting equally strong modeling abilities compared to their ODE-based counterparts. The models are hereby derived from the analytical closed-form solution of an expressive subset of time-continuous models, thus alleviating the need for complex DE solvers all together. In our experimental evaluations, we demonstrate that CfC networks outperform advanced, recurrent models over a diverse set of time-series prediction tasks, including those with long-term dependencies and irregularly sampled data. We believe our findings open new opportunities to train and deploy rich, continuous neural models in resource-constrained settings, which demand both performance and efficiency.

1 Introduction

Continuous neural network architectures built by ordinary differential equations (ODEs) (Chen et al., 2018) opened a new paradigm for obtaining expressive and performant neural models. These continuous-depth models have shown promise in density estimation applications (Dupont et al., 2019, Grathwohl et al., 2018, Yang et al., 2019), as well as modeling sequential and irregularly-sampled data (Gholami et al., 2019, Hasani et al., 2021, Lechner and Hasani, 2020, Rubanova et al., 2019).

While ODE-based neural networks with careful memory and gradient propagation design (Lechner and Hasani, 2020) perform competitively well compared to advanced discretized recurrent models on relatively smaller benchmarks, their training and inference are slow due to the use of advanced numerical DE solvers (Prince and Dormand, 1981). This becomes even more troublesome as the complexity of the task increases (i.e., requiring more precision) (Raissi et al., 2019), for instance, in open-world problems such as medical data processing, self-driving cars, financial time-series, and physics simulations.

The community has worked out solutions for resolving this computational overhead and facilitating the training of neural ODEs, for instance, by relaxing the stiffness of a flow by state augmentation techniques (Dupont et al., 2019, Massaroli et al., 2020b), reformulating the forward-pass as a root-finding problem (Bai et al., 2019), using regularization schemes (Finlay et al., 2020, Kidger et al., 2020, Massaroli et al., 2020a), or improving the inference time of the network (Poli et al., 2020).

In this paper, we take a step back and propose a fundamental solution: we derive a closed-form continuous-depth model that is not only endowed with rich modeling capabilities of ODE-based models but simultaneously does not require any solver to model data. The proposed continuous

*Equal Contributions. 1CSAIL MIT, 2IST Austria, 3Aalborg University, 4University of Vienna. Correspondence to: rhasani@mit.edu Code and data are available at: https://github.com/raminmh/CfC

Preprint. Under review.
neural networks yield significantly faster training and inference speeds while being as expressive as their ODE-based counterparts. Our paper provides a derivation for a closed-form solution to a class of continuous neural networks that implicitly models time to achieve significant speedups. We demonstrate how this solution can be formulated into a neuron model and scaled to create flexible, highly performant architectures on challenging sequential datasets.

**Deriving a Closed-form Solution.** To this end, we take the ODE semantics of liquid time-constant (LTC) networks (Hasani et al., 2021), which are expressive continuous-depth models obtained by a bilinear approximation (Friston et al., 2003) of neural ODE formulation (Chen et al., 2018), and approximate a closed-form solution for the scalar case. Background on the LTC model is provided in Section 2. Given a set of mild assumptions, we theoretically justify how this closed-form solution represents LTCs’ ODE semantics and is as expressive. Figure 1 for instance, shows an LTC-based network trained for autonomous driving (Lechner et al., 2020a). The figure further illustrates how close the proposed solution fits the actual dynamics exhibited from a single neuron ODE given the same parametrization.

**Implicit Time Dependency.** We then dissect the properties of the obtained closed-form solution and accordingly design a new class of neural network models we call Closed-form Continuous-depth (CfC) networks. CfCs have an implicit time dependency in their formulation that does not require an ODE solver to obtain their temporal rollouts. Thus, they maximize the trade-off between accuracy and efficiency of solvers (See Table 1). CfCs perform *local-error-free computations* at least one order of magnitude faster training and inference time compared to their ODE-based counterparts, without loss of accuracy.

**Table 1:** Time Complexity of the process to compute $K$ solver’s steps. *epsilon* is step-size, $\tilde{\epsilon}$ is the max step-size and $\delta << 0$. $\tilde{K}$ is time steps for CfCs which is equivalent to $K$. Table is reproduced and taken from Poli et al. (2020).

| Method                  | Complexity | Local Error  |
|-------------------------|------------|--------------|
| $p$-th order solver     | $O(K \cdot p)$ | $O(\epsilon^{p+1})$ |
| adaptive–step solver    | $-$        | $O(\tilde{\epsilon}^{p+1})$ |
| Euler hypersolver       | $O(K)$     | $O(\delta \epsilon^2)$ |
| $p$-th order hypersolver| $O(K \cdot p)$ | $O(\delta \epsilon^{p+1})$ |
| CfC (Ours)              | $O(\tilde{K})$ | 0 |

Figure 1: We approximate a closed-form solution for LTCs (Hasani et al., 2021) while largely preserving the trajectories of their equivalent ODE systems. We develop our solution into closed-form continuous-depth (CfC) models that are at least 100x faster than neural ODEs at both training and inference on complex time-series prediction tasks.
Sequence and Time-step Prediction Efficiency. CFCS perform per-time-step and per-sequence modeling by establishing a continuous flow similar to ODE-based models. However, as they do not require ODE-solvers, their complexity is at least one order of magnitude less than ODE-based models. Think of having a performant gated recurrent model (Hochreiter and Schmidhuber, 1997) with the abilities to create expressive continuous flows (Chen et al., 2018) and adaptable dynamics (Hasani et al., 2021). Table 2 compares the time complexity of CFCS to that of standard RNNs, ODE-RNNs and Transformers.

CfCs: Flexible Deep Models for Sequential Tasks. CFCS are equipped with novel gating mechanisms that explicitly control their memory. CFCS are at least as expressive as their ODE-based peers and can be supplied with mixed memory architectures (Lechner and Hasani, 2020) to avoid gradient issues in sequential data processing applications. Beyond solely accuracy and performance metrics, our results indicate that when considering accuracy-per-compute time, CFCS exhibit over 150× improvement. We perform a diverse set of advanced time series modeling experiments and present the performance and speed gain achievable by using CFCS in tasks with long-term dependencies, irregular data, and modeling physical dynamics, among others.

2 Deriving a Closed-form Solution

In this section, we derive an approximate closed-form solution for liquid time-constant (LTC) networks, an expressive subclass of time-continuous models. We discuss how the scalar closed-form expression derived from a small LTC system can inspire the design of CFCS models.

The hidden state of an LTC network is determined by the solution of the initial-value problem (IVP) given below (Hasani et al., 2021):

\[
\frac{d x}{dt} = -(w_\tau + f(x, I, \theta)) x(t) + Af(x, I, \theta),
\]

where \( x(t) \) defines the hidden states, \( I(t) \) is the inputs to the system, \( w_\tau \) is a time-constant parameter vector, \( A \) is a bias vector, and \( f \) is a neural network parametrized by \( \theta \).

**Theorem 1.** Given an LTC system determined by the IVP (1), constructed by one cell, receiving a single dimensional time-series input \( I \) with no self connections, the following expression is an approximation of its closed-form solution:

\[
x(t) = (x_0 - A)e^{-[w_\tau + f(I(t), \theta)]t} f(-I(t), \theta) + A
\]

**Proof.** In the single-dimensional case, the IVP (1) becomes linear; and therefore, we can use the theory of linear ODEs to obtain an integral closed-form solution (Perko, 1991, Section 1.10) consisting of two nested integrals. The inner integral can be eliminated by means of integration by substitution (Rudin, 1976). With this, the remaining integral expression can be solved in the case of piecewise constant inputs and approximated in the case of general inputs. The three steps of the proof are outlined below.

**Integral closed-form solution of LTC.** We consider the ODE semantics of a single neuron that receives some continuous input signal \( I \) and relies on a sigmoid function \( f \):

\[
\frac{d x}{dt} = -[w_\tau + f(I(t))] \cdot x(t) + A \cdot [w_\tau + f(I(t))]
\]

By applying linear ODE systems theory (Perko, 1991, Section 1.10), we obtain:

\[
x(t) = e^{-\int_0^t [w_\tau + f(I(s))] ds} \cdot x(0) + \int_0^t e^{-\int_s^t [w_\tau + f(I(v))] dv} \cdot A \cdot (w_\tau + f(I(s))) ds
\]

To resolve the double integral in the equation above, we define

\[
u(s) := \int_s^t [w_\tau + f(I(v))] dv,
\]

| Model          | Sequence prediction | Time-step prediction |
|----------------|---------------------|----------------------|
| RNN            | \( O(nk) \)         | \( O(k) \)           |
| ODE-RNN        | \( O(nkp) \)        | \( O(kp) \)          |
| Transformer    | \( O(n^2k) \)       | \( O(nk) \)          |
| CFCS           | \( O(nk) \)         | \( O(k) \)           |

Table 2: Sequence and time-step prediction complexity. \( n \) is the sequence length, \( k \) the number of hidden units, and \( p \) order of the ODE-solver.
and observe that \( \frac{d}{ds} u(s) = -(w_r + f(I(s))) \). Hence, integration by substitution allows us to rewrite \( (3) \) into:

\[
x(t) = e^{-\int_0^t [w_r + f(I(s))]ds} \cdot x(0) - A \int_{u(0)}^{u(t)} e^{-u} du
\]

\[
= x(0)e^{-\int_0^t [w_r + f(I(s))]ds} + A[e^{-u}|_{u(0)}^{u(t)}]
\]

\[
= x(0)e^{-\int_0^t [w_r + f(I(s))]ds} + A(1 - e^{-\int_0^t [w_r + f(I(s))]ds})
\]

\[
= (x(0) - A)e^{-w_r t} e^{-\int_0^t f(I(s))ds} + A
\]

\[ (4) \]

**Analytical LTC solution for piecewise constant inputs.** The derivation of a *useful* closed-form expression of \( x \) requires us to solve the integral expression \( \int_0^t f(I(s))ds \) for any \( t \geq 0 \). Fortunately, the integral \( \int_0^t f(I(s))ds \) enjoys a simple closed-form expression for piecewise constant inputs \( I \). Specifically, assume that we are given a sequence of time points:

\[
0 = \tau_0 < \tau_1 < \tau_2 < \ldots < \tau_{n-1} < \tau_n = \infty,
\]

such that \( \tau_1, \ldots, \tau_{n-1} \in \mathbb{R} \) and \( I(t) = \gamma_i \) for all \( t \in [\tau_i; \tau_{i+1}) \) with \( 0 \leq i \leq n - 1 \). Then, it holds that

\[
\int_0^t f(I(s))ds = f(\gamma_k)(t - \tau_k) + \sum_{i=0}^{k-1} f(\gamma_i)(\tau_{i+1} - \tau_i),
\]

when \( \tau_k \leq t < \tau_{k+1} \) for some \( 0 \leq k \leq n - 1 \) (as usual, one defines \( \sum_{i=0}^{-1} := 0 \)). With this, we have:

\[
x(t) = (x(0) - A)e^{-w_r t} e^{-\int_\tau_k^{\tau_{k+1}} f(\gamma_i)(t - \tau_k) - \sum_{i=0}^{k-1} f(\gamma_i)(\tau_{i+1} - \tau_i)} + A
\]

\[ (6) \]

when \( \tau_k \leq t < \tau_{k+1} \) for some \( 0 \leq k \leq n - 1 \). While any continuous input can be approximated arbitrarily well by a piecewise constant input \( \text{Rudin} \) (1976), a tight approximation may require a large number of discretization points \( \tau_1, \ldots, \tau_n \). We address this next.

**Analytical LTC approximation for general inputs.** Inspired by Eq. 5, the next result provides an analytical approximation of \( x(t) \).

**Lemma 1.** For any Lipschitz continuous sigmoid \( f \) and continuous input signal \( I \), we approximate \( x(t) \) in \( (4) \) as follows:

\[
\hat{x}(t) = (x(0) - A)e^{-w_r t} e^{-f(I(t))} f(-I(t)) + A
\]

\[ (7) \]

Then, \( |x(t) - \hat{x}(t)| \leq |x(0) - A| e^{-w_r t} \) for all \( t \geq 0 \). Writing \( c = x(0) - A \) for convenience, we can obtain the following sharpness results, additionally:

1. For any \( t \geq 0 \), we have \( \sup \{ \frac{1}{c} |x(t) - \hat{x}(t)| \mid I : [0; t] \rightarrow \mathbb{R} \} = e^{-w_r t} \).

2. For any \( t \geq 0 \), we have \( \inf \{ \frac{1}{c} |x(t) - \hat{x}(t)| \mid I : [0; t] \rightarrow \mathbb{R} \} = e^{-w_r t}(e^{-t} - 1) \).

Above, the supremum and infimum are meant to be taken across all continuous input signals. These statements settle the question about the worst-case errors of the approximation. The first statement implies in particular that our bound is sharp. Note that as \( w_r \) is positively defined, the bound ensures an exponentially decaying error as time goes by.

**Proof.** We start by noting that

\[
x(t) - \hat{x}(t) = c[e^{-w_r t} - \int_0^t f(I(s))ds - e^{-w_r t} f(-I(t))] f(-I(t))
\]

\[
= ce^{-w_r t} [e^{-\int_0^t f(I(s))ds} - e^{-f(I(t))} f(-I(t))]
\]

Since \( 0 \leq f \leq 1 \), we conclude \( e^{-\int_0^t f(I(s))ds} \in [0; 1] \) and \( e^{-f(I(t))} f(-I(t)) \in [0; 1] \). This shows that \( |x(t) - \hat{x}(t)| \leq |c| e^{-w_r t} \). To see the sharpness results, pick some arbitrary small \( \varepsilon > 0 \) and a sufficiently large \( C > 0 \) such that \( f(-C) \leq \varepsilon \) and \( 1 - \varepsilon \leq f(C) \). With this, for any \( 0 < \delta < t \), we
consider the piecewise constant input signal $I$ such that $I(s) = -C$ for $s \in [0; t - \delta]$ and $I(s) = C$ for $s \in (t - \delta; t]$. Then, it can be noted that

$$e^{-\int_0^t f(I(s))ds} - e^{-f(I(t))t} f(-I(t)) \geq e^{-e^{-\delta} t} - e^{-(1-\epsilon) t} \epsilon \to 1, \quad \text{when } \epsilon, \delta \to 0$$

Statement 1) follows by noting that there exists a family of continuous signals $I_n : [0; t) \to \mathbb{R}$ such that $|I_n(t)| \leq C$ for all $n \geq 1$ and $I_n \to I$ pointwise as $n \to \infty$. This is because

$$\lim_{n \to \infty} \left| \int_0^t f(I(s))ds - \int_0^t f(I_n(s))ds \right| = \lim_{n \to \infty} \int_0^t |f(I(s)) - f(I_n(s))|ds \leq \lim_{n \to \infty} L \int_0^t |I(s) - I_n(s)|ds = 0$$

where $L$ is the Lipschitz constant of $f$ and the last identity is due to dominated convergence theorem Rudin (1976). To see 2), we first note that the negation of the signal $-I$ provides us with

$$e^{-\int_0^t f(-I(s))ds} - e^{-f(-I(t))t} f(I(t)) \leq e^{-(1-\epsilon)(t-\delta)} - e^{-\epsilon t}(1-\epsilon) \to e^{-t} - 1,$$

if $\epsilon, \delta \to 0$. The fact that the left-hand side of the last inequality must be at least $e^{-t} - 1$ follows by observing that $e^{-t} \leq e^{-\int_0^t f(I(s))ds}$ and $e^{-f(I''(t))t} f(-I''(t)) \leq 1$ for any $I', I'' : [0; t) \to \mathbb{R}$. □

Therefore, we have the statement of the theorem. □

3 Design a Closed-form Continuous-depth Model Inspired by the Solution

Leveraging the scalar closed-form solution expressed by Eq. (2), we can now distill this model into a neural network that can be trained at scale. Despite the solution providing a grounded theoretical basis for solving scalar continuous-time dynamics, it is critically important to translate this theory into a practical neural network model which can be integrated into larger representation learning systems. Doing so requires careful attention to potential gradient and expressivity issues that can arise during optimization, which we will outline in this section.

Formally, the hidden states, $x(t) (D \times 1)$ with $D$ hidden units at each time step $t$, can be implicitly obtained by:

$$x(t) = B \odot e^{-[w_r + f(x, \theta)]t} \odot (-x, -I; \theta) + A,$$

where $B^{(D)}$, $A^{(D)}$, and $w_r^{(D)}$ are system’s parameter vectors, $I(t) (m \times 1)$ is an $m$-dimensional input at each time step $t$, $f$ is a neural network parametrized by $\theta = \{W^{(m \times D)}, W^{(D \times D)}, b^{(D)}\}$, and $\odot$ is the Hadamard (element-wise) product. While the neural network presented in 8 can be proven to be a universal approximator as it is an approximation of an ODE system Chen et al. (2018), Hasani et al. (2021), in its current form, it has trainability issues which we point out and resolve shortly:

Resolving the gradient issues. The exponential term in Eq. 8 derives the system’s first part exponentially fast to 0 and the entire hidden state to $A$. This issue becomes more apparent when there are recurrent connections and causes vanishing gradient factors when trained by gradient descent.
Hochreiter (1991). To reduce the effect, we replace the exponential decay term with a reversed sigmoidal nonlinearity $\sigma(\cdot)$. This nonlinearity is approximately 1 at $t = 0$ and approaches 0 in the limit $t \to \infty$. However, unlike the exponential decay, its transition happens much smoother, yielding a better condition on the loss surface.

**Replacing biases by learnable instances.** Next, we consider the bias parameter $B$ to be part of the trainable parameters of the neural network $f(-x, -I; \theta)$ and choose to use a new network instance instead of $f$ (presented in the exponential decay factor). We also replace $A$ with another neural network instance, $h(\cdot)$ to enhance the flexibility of the model.

**Gating balance.** The time-decaying sigmoidal term can play a gating role if we additionally multiply $h(\cdot)$, with $(1 - \sigma(\cdot))$. This way, the time-decaying sigmoid function stands for a gating mechanism that interpolates between the two limits of $t \to -\infty$ and $t \to \infty$ of the ODE trajectory.

**Backbone.** Instead of learning all three neural networks $f, g$ and $h$ separately, we make them share the first few layers in the form of a backbone that branches out into these three functions. As a result, the backbone allows our model to learn shared representations, thereby speeding up and stabilizing the learning process.

These modifications result in the closed-form continuous-depth (CfC) neural network model:

$$x(t) = \sigma(-f(x, I; \theta_f) t) \odot g(x, I; \theta_g) + [1 - \sigma(-[f(x, I; \theta_f)] t)] \odot h(x, I; \theta_h).$$  \hspace{1cm} (9)

The CfC architecture is illustrated in Figure 2. The neural network instances could be selected arbitrarily. The time complexity of the algorithm is equivalent to that of discretized recurrent networks (Hasani et al., 2019a), which is at least one order of magnitude faster than ODE-based networks.

**How do you deal with time, $t$?** CfCs are continuous-depth models that can set their temporal behavior based on the task-under-test. For time-variant datasets (e.g., irregularly sampled time series, event-based data, and sparse data), $t$ for each incoming sample is set based on its time-stamp or order. For sequential applications where the time of the occurrence of a sample does not matter, $t$ is sampled batch-length-times with equidistant intervals within two hyperparameters $a$ and $b$.

## 4 Experiments with CfCs

We now extensively assess the performance of CfCs in a series of sequential data processing tasks compared to advanced, recurrent models. In each experiment, we ablate the full CfC architecture into four different variants, compare them to an extensive set of baselines described below and report results.

**CfC Network Variants.** To evaluate how the proposed modifications we applied to the closed-form solution network described by Eq. 8, we test four variants of the CfC architecture: 1) Closed-form solution network (Cf-S) obtained by Eq. 8, 2) CfC without the second gating mechanism (CfC-noGate). This variant does not have the $1 - \sigma$ instance shown in Figure 2. 3) Closed-form Continuous-depth model (CfC) expressed by Eq. 9. 4) CfC wrapped inside a mixed-memory architecture (i.e., CfC defines the memory state of an RNN for instance an LSTM). We call this variant CfC-mmRNN. Each of these four proposed variants leverage our proposed solution, and thus are at least one order of magnitude faster than continuous-time ODE models.

**Baselines.** We compare CfCs to a diverse set of advanced algorithms developed for sequence modeling by both discretized and continuous mechanisms. Examples include some variations of classical autoregressive RNNs, such as an RNN with concatenated $\Delta t$ (RNN-$\Delta t$), a recurrent model with moving average on missing values (RNN-impute), RNN Decay (Rubanova et al., 2019), long short-term memory (LSTMs) Hochreiter and Schmidhuber (1997), and gated recurrent units (GRUs) Chung et al. (2014). We also report results for a variety of encoder-decoder ODE-RNN based models, such as RNN-VAE, Latent variable models with RNNs, and with ODEs, all from Rubanova et al. (2019).

Furthermore, we include models such as interpolation prediction networks (IP-Net) (Shukla and Marlin, 2018), Set functions for time-series (SeFT) (Horn et al., 2020), CT-RNNs Funahashi and
Table 3: Bit-stream XOR sequence classification. The performance values for all baseline models are reproduced from Lechner and Hasani (2020), n=5

| Model                        | Equidistant encoding | Event-based (irregular) encoding |
|------------------------------|----------------------|----------------------------------|
| ODE-RNN (Rubanova et al., 2019) | 50.47% ± 0.06        | 51.21% ± 0.37                    |
| CT-RNN (Funahashi and Nakamura, 1993) | 50.42% ± 0.12        | 50.79% ± 0.34                    |
| Augmented LSTM (Hochreiter and Schmidhuber, 1997) | **100.00% ± 0.00**   | 89.71% ± 3.48                    |
| CT-GRU (Mozer et al., 2017) | **100.00% ± 0.00**   | 61.36% ± 4.87                    |
| RNN Decay (Rubanova et al., 2019) | 60.28% ± 19.87       | 75.53% ± 5.28                    |
| Bi-directional RNN (Schuster and Paliwal, 1997) | **100.00% ± 0.00**   | 90.17% ± 0.69                    |
| GRU-D (Che et al., 2018)    | **100.00% ± 0.00**   | **97.90% ± 1.71**                |
| PhasedLSTM (Neil et al., 2016) | 50.99% ± 0.76        | 80.29% ± 0.99                    |
| GRU-ODE (Rubanova et al., 2019) | 50.41% ± 0.40        | 52.52% ± 0.35                    |
| CT-LSTM Mei and Eisner (2017) | 97.73% ± 0.08        | 95.09% ± 0.30                    |
| ODE-LSTM (Lechner and Hasani, 2020) | **100.00% ± 0.00**   | **98.89% ± 0.26**                |
| coRNN (Rusch and Mishra, 2021) | 100.00% ± 0.00       | 52.89% ± 1.25                    |
| Lipschitz RNN (Erichson et al., 2021) | 100.00% ± 0.00       | 52.84% ± 3.25                    |
| LTC (Hasani et al., 2021)   | **100.00% ± 0.00**   | **49.11% ± 0.00**                |
| Cf-S (ours)                  | **100.00% ± 0.00**   | 85.42% ± 2.84                    |
| CIC-noGate (ours)            | **100.00% ± 0.00**   | 96.29% ± 1.61                    |
| CIC (ours)                   | **100.00% ± 0.00**   | **99.42% ± 0.42**                |
| CIC-mmRNN (ours)             | **100.00% ± 0.00**   | **99.72% ± 0.08**                |

Note: The performance of models marked by † are reported from (Lechner and Hasani, 2020).

Nakamura (1993), CT-GRU (Mozer et al., 2017), CT-LSTM (Mei and Eisner, 2017), GRU-D (Che et al., 2018), Phased-LSTM (Neil et al., 2016), bi-directional RNNs (Schuster and Paliwal, 1997). Finally, we benchmarked CICs against competitive recent RNN architectures with the premise of tackling long-term dependencies, such as Legandre Memory Units (LMU) (Voelker et al., 2019), high-order polynomial projection operators (Hippo) (Gu et al., 2020), orthogonal recurrent models (expRNNs) Lezcano-Casado and Martinez-Rubio (2019), mixed memory RNNs such as (ODE-LSTMs) (Lechner and Hasani, 2020), coupled oscillatory RNNs (coRNN) (Rusch and Mishra, 2021), and Lipschitz RNNs (Erichson et al., 2021).

4.1 Regularly and Irregularly-Sampled Bit-Stream XOR

The bit-stream XOR dataset (Lechner and Hasani, 2020) considers classifying bit-streams implementing an XOR function in time, i.e., each item in the sequence contributes equally to the correct output. The bit-streams are provided in densely sampled and event-based sampled format. The densely sampled version simply represents an incoming bit as an input event. The event sampled version transmits only bit-changes to the network, i.e., multiple equal bit is packed into a single input event. Consequently, the densely sampled variant is a regular sequence classification problem, whereas the event-based encoding variant represents an irregularly sampled sequence classification problem.

Table 3 compares the performance of many RNN baselines. Many architectures such as Augmented LSTM, CT-GRU, GRU-D, ODE-LSTM, coRNN, and Lipschitz RNN, and all variants of CIC can successfully solve the task with 100% accuracy when the bit-stream samples are equidistant from each other. However, when the bit-stream samples arrive at non-uniform distances, only architectures that are immune to the vanishing gradient in irregularly sampled data can solve the task. These include GRU-D, ODE-LSTM and CICs, and CIC-mmRNNs. ODE-based RNNs cannot solve the event-based encoding tasks regardless of their choice of solvers, as they have vanishing/exploding gradient issues (Lechner and Hasani, 2020).

4.2 PhysioNet Challenge

The PhysioNet Challenge 2012 dataset considers the prediction of the mortality of 8000 patients admitted to the intensive care unit (ICU). The features represent time series of medical measurements of the first 48 hours after admission. The data is irregularly sampled in time, and over features, i.e., only a subset of the 37 possible features is given at each time point. We perform the same test-train
Table 4: PhysioNet. **without any pretraining or pretrained word-embeddings:** Thus, we excluded advanced attention-based models (Shukla and Marlin, 2021, Xiong et al., 2021) such as Transformers (Vaswani et al., 2017) and RNN structures that use pretraining. n=5

| Model                              | AUC Score (%) | time per epoch (min) |
|------------------------------------|---------------|----------------------|
| †RNN-Impute (Rubanova et al., 2019) | 0.764 ± 0.016  | 0.5                  |
| †RNN-delta-t (Rubanova et al., 2019) | 0.787 ± 0.014  | 0.5                  |
| †RNN-Decay (Rubanova et al., 2019) | 0.807 ± 0.003  | 0.7                  |
| †GRU-D (Che et al., 2018) | 0.818 ± 0.008  | 0.7                  |
| †Phased-LSTM (Neil et al., 2016) | **0.836 ± 0.003**  | 0.3                  |
| †IP-Nets (Shukla and Marlin, 2018) | 0.819 ± 0.006  | 1.3                  |
| †SeFT (Horn et al., 2020) | 0.795 ± 0.015  | 0.5                  |
| †RNN-VAE (Rubanova et al., 2019) | 0.515 ± 0.040  | 2.0                  |
| †ODE-RNN (Rubanova et al., 2019) | **0.833 ± 0.009**  | 16.5                 |
| †Latent-ODE-RNN (Rubanova et al., 2019) | 0.781 ± 0.018  | 6.7                  |
| †Latent-ODE-ODE (Rubanova et al., 2019) | 0.829 ± 0.004  | 22.0                 |
| LTC (Hasani et al., 2021) | 0.6477 ± 0.010  | 0.5                  |

| CF-S (ours) | 0.643 ± 0.018 | 0.1 |
| CF-C-noGate (ours) | **0.840 ± 0.003**  | 0.1 |
| CF-C (ours) | **0.839 ± 0.002**  | 0.1 |
| CF-C-mmRNN (ours) | 0.834 +/- 0.006  | 0.2 |

Note: The performance of the models marked by † are reported from (Rubanova et al., 2019) and the ones with * from (Shukla and Marlin, 2021).

split and preprocesing as (Rubanova et al., 2019), and report the area under the curve (AUC) on the test set as metric in Table 4. We observe that CfCs perform competitively to other baselines while performing 160 times faster training time compared to ODE-RNNs and 220 times compared to continuous latent models. CfCs are also, on average, three times faster than advanced discretized gated recurrent models.

### 4.3 Sentiment Analysis - IMDB

The IMDB sentiment analysis dataset (Maas et al., 2011) consists of 25,000 training and 25,000 test sentences. Each sentence corresponds to either positive or negative sentiment. We tokenize the sentences in a word-by-word fashion with a vocabulary consisting of 20,000 most frequently occurring words in the dataset. We map each token to a vector using a trainable word embedding. The word embedding is initialized randomly. No pretraining of the network or the word embedding is performed. Table 5 represents how CfCs equipped with mixed memory instances outperform advanced RNN benchmarks.

### 4.4 Physical Dynamics Modeling

The Walker2D dataset consists of kinematic simulations of the MuJoCo physics engine (Todorov et al., 2012) on the Walker2d-v2 OpenAI gym (Brockman et al., 2016) environment using four different stochastic policies. The objective is to predict the physics state of the next time step. The training and testing sequences are provided at irregularly-sampled intervals. We report the squared error on the test set as a metric. As shown in Table 6, CfCs outperform the other baselines by a large margin rooting for their strong capability to model irregularly sampled physical dynamics with missing phases. It is worth mentioning that on this task, CfCs even outperform Transformers by a considerable 18% margin.

### 5 Related Works

**Continuous-Depth Models.** Machine learning, control theory and dynamical systems merge at models with continuous-time dynamics (Lechner et al., 2019, Li et al., 2017, Lu et al., 2017, Weinan, 2017, Zhang et al., 2014). In a seminal work, Chen et al. (2018) revived the class of continuous-time neural networks (Cohen and Grossberg, 1983, Funahashi and Nakamura, 1993), with neural ODEs.
Thus, we excluded advanced attention-based models (Shukla and Marlin, 2021, Xiong et al., 2021) such as Transformers (Vaswani et al., 2017) and RNN structures that use pretraining.

| Model                          | Test accuracy (%)          |
|--------------------------------|---------------------------|
| HiPPO-LagT (Gu et al., 2020)  | 88.0 ± 0.2                |
| HiPPO-LegS (Gu et al., 2020)  | 88.0 ± 0.2                |
| LMU (Voelker et al., 2019)   | 87.7 ± 0.1                |
| LSTM (Hochreiter and Schmidhuber, 1997) | 87.3 ± 0.4            |
| GRU (Chung et al., 2014)     | 86.2 ± n/a                |
| ReLU GRU (Dey and Salem, 2017) | 84.8 ± n/a               |
| Skip LSTM (Campos et al., 2017) | 86.6 ± n/a               |
| expRNN (Lezcano-Casado and Martinez-Rubio, 2019) | 84.3 ± 0.3           |
| Vanilla RNN (Campos et al., 2017) | 67.4 ± 7.7            |
| coRNN (Rusch and Mishra, 2021) | 86.7 ± 0.3              |
| LTC (Hasani et al., 2021)    | 61.8 ± 6.1                |
| Cf-S (ours)                  | 81.7 ± 0.5                |
| CfC-noGate (ours)            | 87.5 ± 0.1                |
| CfC (ours)                   | 85.9 ± 0.9                |
| CfC-mmRNN (ours)             | 88.3 ± 0.1                |

Note: The performance of the models marked by † are reported from (Gu et al., 2020), and * are reported from (Rusch and Mishra, 2021).

Table 6: Per time-step regression. Walker2d kinematic dataset. (mean ± std, N = 5)

| Model                                     | Square-error |
|-------------------------------------------|--------------|
| †ODE-RNN (Rubanova et al., 2019)         | 1.904 ± 0.061|
| †CT-RNN (Funahashi and Nakamura, 1993)   | 1.198 ± 0.004|
| †Augmented LSTM (Hochreiter and Schmidhuber, 1997) | 1.065 ± 0.006|
| †CT-GRU (Mozer et al., 2017)             | 1.172 ± 0.011|
| †RNN-Decay (Rubanova et al., 2019)       | 1.406 ± 0.005|
| †Bi-directional RNN (Schuster and Paliwal, 1997) | 1.071 ± 0.009|
| †GRU-D (Che et al., 2018)                | 1.090 ± 0.034|
| †PhasedLSTM (Neil et al., 2016)          | 1.063 ± 0.010|
| †GRU-ODE (Rubanova et al., 2019)         | 1.051 ± 0.018|
| †CT-LSTM (Mei and Eisner, 2017)          | 1.014 ± 0.014|
| †ODE-LSTM (Lechner and Hasani, 2020)     | 0.883 ± 0.014|
| coRNN (Rusch and Mishra, 2021)           | 3.241 ± 0.215|
| Lipschitz RNN (Erichson et al., 2021)    | 1.781 ± 0.013|
| LTC (Hasani et al., 2021)                | 0.662 ± 0.013|
| Transformer (Vaswani et al., 2017)        | 0.761 ± 0.032|
| Cf-S (ours)                               | 0.948 ± 0.009|
| CfC-noGate (ours)                         | 0.650 ± 0.008|
| CfC (ours)                                | 0.643 ± 0.006|
| CfC-mmRNN (ours)                          | 0.617 ± 0.006|

Note: The performance of the models marked by † are reported from Lechner and Hasani (2020).

These continuous-depth models give rise to vector field representations and a set of functions which were not possible to generate before. These capabilities enabled flexible density estimation (Dupont et al., 2019, Grathwohl et al., 2018, Hodgkinson et al., 2020, Mathieu and Nickel, 2020, Yang et al., 2019), as well as performant modeling of sequential and irregularly-sampled data (Erichson et al., 2021, Gholami et al., 2019, Hasani et al., 2021, Lechner and Hasani, 2020, Rubanova et al., 2019). In this paper, we showed how to relax the need for an ODE-solver to realize an expressive continuous-time neural network model for challenging time-series problems.
**Improving Neural ODEs.** ODE-based neural networks are as good as their ODE-solvers. As the complexity or the dimensionality of the modeling task increases, ODE-based networks demand a more advanced solver that significantly impacts their efficiency (Poli et al., 2020), stability (Bai et al., 2019, Chang et al., 2019, Haber et al., 2019, Lechner et al., 2020b, Massaroli et al., 2020a) and performance (Hasani et al., 2021). A large body of research went into improving the computational overhead of these solvers, for example, by designing hypersolvers (Poli et al., 2020), deploying augmentation methods (Dupont et al., 2019, Massaroli et al., 2020b), pruning (Liebenwein et al., 2021) and by regularizing the continuous flows (Finlay et al., 2020, Kidger et al., 2020, Massaroli et al., 2020a). To enhance the performance of an ODE-based model, especially in time series modeling tasks (Gleeson et al., 2018), solutions provided for stabilizing their gradient propagation (Erichson et al., 2021, Lechner and Hasani, 2020, Li et al., 2020). In this work, we showed that CfCs improve the scalability, efficiency, and performance of continuous-depth neural models.

### 6 Scope, Discussions and Conclusions

We introduced a closed-form continuous-time neural model that possesses the strong modeling capabilities of ODE-based networks while being significantly faster, more accurate, and stable. These closed-form continuous-depth models achieve this by implicit time-dependent gating mechanisms and having a liquid time-constant modulated by neural networks.

**Now that we have a closed-form system, where does it make sense to use Neural ODEs?** ODE-based models can still be significantly beneficial in solving continuously defined physics problems and control tasks. Though, for large-scale time-series prediction tasks, CfCs are the right way to go. Moreover, for generative modeling, continuous normalizing flows built by ODEs are the suitable choice of model as they ensure invertibility unlike CfCs (Chen et al., 2018).

**What are the limitations of CfCs?** CfCs might express vanishing gradient problems. To avoid this, for tasks that require long-term dependencies, it is better to use them together with mixed memory networks (Sec CfC-mmRNN). Moreover, we speculate that inferring causality from ODE-based networks might be more straightforward than a closed-form solution (Vorbach et al., 2021). It would also be beneficial to assess if verifying a continuous neural flow (Grunbacher et al., 2021) is more tractable by an ODE representation of the system or their closed form.

**When shall we use Transformers, when CfCs?** For problems such as language modeling where a significant amount of sequential data and substantial compute resources are available, Transformers are the right choice. In contrast, we use CfCs when: 1) data has limitations and irregularities (e.g., medical data, financial time-series, robotics (Lechner et al., 2021) and automation (Hasani et al., 2019b, 2017), and multi-agent autonomous systems in supervised and reinforcement learning schemes (Brunnbauer et al., 2021)), 2) training and inference efficiency of a model is important (e.g., embedded applications (DelPreto et al., 2020, Hasani et al., 2016, Wang et al., 2019)), and 3) when interpretability matters (Hasani, 2020).

### Acknowledgments

R.H. and D.R. are partially supported by Boeing and MIT. M.L. is supported in part by the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award). A.A. is supported by the National Science Foundation (NSF) Graduate Research Fellowship Program. M.T. is supported by the Poul Due Jensen Foundation, grant 883901. This research was partially sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. This work was further supported by The Boeing Company and the Office of Naval Research (ONR) Grant N00014-18-1-2830.
References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32:690–701, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Axel Brunnbauer, Luigi Berducci, Andreas Brandstätter, Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. Model-based versus model-free deep reinforcement learning for autonomous racing cars. *arXiv preprint arXiv:2103.04909*, 2021.

Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017.

Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Michael A Cohen and Stephen Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE transactions on systems, man, and cybernetics*, pages 815–826, 1983.

Joseph DelPreto, Andres F Salazar-Gomez, Stephanie Gil, Ramin Hasani, Frank H Guenther, and Daniela Rus. Plug-and-play supervisory control using muscle and brain signals for real-time gesture and error detection. *Autonomous Robots*, 44(7):1303–1322, 2020.

Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In *Advances in Neural Information Processing Systems*, pages 3134–3144, 2019.

N. Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W. Mahoney. Lipschitz recurrent neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-N7PBXqOUIZ.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. *arXiv preprint arXiv:2002.02798*, 2020.

Karl J Friston, Lee Harrison, and Will Penny. Dynamic causal modelling. *Neuroimage*, 19(4):1273–1302, 2003.

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

Amir Gholami, Kurt Keutzer, and George Biros. Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298*, 2019.

Padraig Gleeson, David Lung, Radu Grosu, Ramin Hasani, and Stephen D Larson. c302: a multiscale framework for modelling the nervous system of caenorhabditis elegans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1758):20170379, 2018.
Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Sophie Grunbacher, Ramin Hasani, Mathias Lechner, Jacek Cyranka, Scott A. Smolka, and Radu Grosu. On the verification of neural odes with stochastic guarantees. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11525–11535, May 2021.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *arXiv preprint arXiv:2008.07669*, 2020.

Eldad Haber, Keegan Lensink, Eran Treister, and Lars Ruthotto. Imexnet a forward stable deep neural network. In *International Conference on Machine Learning*, pages 2525–2534. PMLR, 2019.

Ramin Hasani. *Interpretable Recurrent Neural Networks in Continuous-time Control Environments*. PhD dissertation, Technische Universität Wien, 05 2020.

Ramin Hasani, Alexander Amini, Mathias Lechner, Felix Naser, Radu Grosu, and Daniela Rus. Response characterization for auditing cell dynamics in long short-term memory networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019a.

Ramin Hasani, Guodong Wang, and Radu Grosu. A machine learning suite for machine components’ health-monitoring. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9472–9477, 2019b.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7657–7666, May 2021.

Ramin M Hasani, Dieter Haerle, and Radu Grosu. Efficient modeling of complex analog integrated circuits using neural networks. In *2016 12th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*, pages 1–4. IEEE, 2016.

Ramin M Hasani, Dieter Haerle, Christian F Baumgartner, Alessio R Lomuscio, and Radu Grosu. Compositional neural-network modeling of complex analog circuits. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2235–2242. IEEE, 2017.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Liam Hodgkinson, Chris van der Heide, Fred Roosta, and Michael W Mahoney. Stochastic normalizing flows. *arXiv preprint arXiv:2002.09547*, 2020.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In *International Conference on Machine Learning*, pages 4353–4363. PMLR, 2020.

Patrick Kidger, Ricky TQ Chen, and Terry Lyons. "hey, that’s not an ode": Faster ode adjoints with 12 lines of code. *arXiv preprint arXiv:2009.09457*, 2020.

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

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

Mathias Lechner, Ramin Hasani, Alexander Amini, Thomas A Henzinger, Daniela Rus, and Radu Grosu. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2(10):642–652, 2020a.
Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. Gershgorin loss stabilizes the recurrent neural network compartment of an end-to-end robot learning scheme. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 5446–5452. IEEE, 2020b.

Mathias Lechner, Ramin Hasani, Radu Grosu, Daniela Rus, and Thomas A Henzinger. Adversarial training is not ready for robot learning. arXiv preprint arXiv:2103.08187, 2021.

Mario Lezcano-Casado and David Martinez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In International Conference on Machine Learning, pages 3794–3803. PMLR, 2019.

Qianxiao Li, Long Chen, Cheng Tai, et al. Maximum principle based algorithms for deep learning. arXiv preprint arXiv:1710.09513, 2017.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In International Conference on Artificial Intelligence and Statistics, pages 3870–3882. PMLR, 2020.

Lucas Liebenwein, Ramin Hasani, Alexander Amini, and Rus Daniela. Sparse flows: Pruning continuous-depth models. arXiv preprint arXiv:2106.12718, 2021.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiquiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. arXiv preprint arXiv:1709.02540, 2017.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies, pages 142–150, 2011.

Stefano Massaroli, Michael Poli, Michelangelo Bin, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Stable neural flows. arXiv preprint arXiv:2003.08063, 2020a.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asma. Dissecting neural odes. In 34th Conference on Neural Information Processing Systems, NeurIPS 2020. The Neural Information Processing Systems, 2020b.

Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 2503–2515. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1aa3d9c6ce672447e1e5d0f1b5207e85-Paper.pdf.

Hongyuan Mei and Jason Eisner. The neural hawkes process: a neurally self-modulating multivariate point process. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 6757–6767, 2017.

Michael C Mozer, Denis Kazakov, and Robert V Lindsey. Discrete event, continuous time rnns. arXiv preprint arXiv:1710.04110, 2017.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: accelerating recurrent network training for long or event-based sequences. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 3889–3897, 2016.

Lawrence Perko. Differential Equations and Dynamical Systems. Springer-Verlag, Berlin, Heidelberg, 1991. ISBN 0387974431.

Michael Poli, Stefano Massaroli, Atsushi Yamashita, Hajime Asama, Jinkyoo Park, et al. Hypersolvers: Toward fast continuous-depth models. Advances in Neural Information Processing Systems, 33, 2020.

Peter J Prince and John R Dormand. High order embedded runge-kutta formulae. Journal of computational and applied mathematics, 7(1):67–75, 1981.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019.
Yulia Rubanova, Ricky TQ Chen, and David Duvenaud. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907*, 2019.

Walter Rudin. *Principles of mathematical analysis*. McGraw-Hill New York, 3d ed. edition, 1976.

T. Konstantin Rusch and Siddhartha Mishra. Coupled oscillatory recurrent neural network (co{rnn}): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=F3s69XzW0ia.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Satya Narayan Shukla and Benjamin Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2018.

Satya Narayan Shukla and Benjamin M Marlin. Multi-time attention networks for irregularly sampled time series. *arXiv preprint arXiv:2101.10318*, 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In 2012 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Aaron R V oelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. *NeurIPS Reproducability Challenge*, 2019.

Charles Vorbach, Ramin Hasani, Alexander Amini, Mathias Lechner, and Daniela Rus. Causal navigation by continuous-time neural networks. *arXiv preprint arXiv:2106.08314*, 2021.

Guodong Wang, Anna Ledwoch, Ramin M Hasani, Radu Grosu, and Alexandra Brintrup. A generative neural network model for the quality prediction of work in progress products. *Applied Soft Computing*, 85:105683, 2019.

E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *CoRR*, abs/2102.03902, 2021.

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019.

Huaguang Zhang, Zhanshan Wang, and Derong Liu. A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(7):1229–1262, 2014.
S1 Experimental details of Figure 1.

We took a trained Neural Circuit Policy (NCP) (Lechner et al., 2020a), which consisted of a perception module and a liquid time-constant (LTC) based network Hasani et al. (2021). The network was trained to autonomously steer an self-driving vehicle. We used a couple of recorded real-world test-runs of the vehicle for a lane-keeping task. The records included the inputs, outputs as well as all LTC neurons’ activities and parameters. To perform a sanity check whether our proposed closed-form solution for LTC neurons is good enough, we plugged in the parameters of neurons and synapses to the closed-form solution and emulated the structure of the ODE-based LTC networks. We then visualized the output neuron’s dynamics taken from the ODE (in blue) and from the closed-form solution (in red). As illustrated in Figure 1, we observed that the behavior of the ODE is captured considerably well by the closed-form solution. We included the data and code for reproducing this experiment as a supplementary material (folder: figure_1_experiments).

S2 Hyperparameters

| Parameter               | IMDB experiments | Physionet experiments |
|-------------------------|------------------|-----------------------|
| **Table S1**: IMDB experiments. Hyperparameters | **Table S2**: Physionet experiments. Hyperparameters |
| Parameter               | Value            | Value                |
| clipnorm                | 1                | 1                    |
| optimizer               | Adam             | AdamW                |
| batch_size              | 128              | 128                  |
| Hidden size             | 320              | 64                   |
| embed_dim               | 64               | 64                   |
| embed_dr                | 0.0              | 0.0                  |
| epochs                  | 27               | 27                   |
| base_lr                 | 0.0005           | 0.003                |
| decay_lr                | 0.8              | 0.7                  |
| backbone_activation     | ReLu             | Tanh                 |
| backbone_dr             | 0.0              | 0.1                  |
| backbone_units          | 64               | 64                   |
| backbone_layers         | 1                | 2                    |
| weight_decay            | 0.00048          | 3.6e-05              |

| Parameter               | Value            | Value                |
|-------------------------|------------------|----------------------|
| epochs                  | 116              | 116                  |
| class_weight            | 18.25            | 11.69                |
| clipnorm                | 0                | 0                    |
| Hidden size             | 64               | 64                   |
| base_lr                 | 0.003            | 0.002                |
| decay_lr                | 0.72             | 0.9                  |
| backbone_activation     | Tanh             | SiLU                 |
| backbone_units          | 64               | 64                   |
| backbone_dr             | 0.1              | 0.2                  |
| backbone_layers         | 3                | 2                    |
| weight_decay            | 5e-05            | 4e-06                |
| optimizer               | AdamW            | AdamW                |
| init                    | 0.53             | 0.50                 |
| batch_size              | 128              | 128                  |
Table S3: **Bit-Stream XOR experiments.** Hyperparameters

| Parameter              | Cf-S | CfC | CfC-noGate | CfC-mmRNN |
|------------------------|------|-----|------------|-----------|
| clipnorm               | 5    | 1   | 10         | 10        |
| optimizer              | Adam | RMSProp | RMSprop | RMSprop |
| batch_size             | 256  | 128 | 128        | 128       |
| Hidden size            | 64   | 192 | 128        | 64        |
| epochs                 | 200  | 200 | 200        | 200       |
| base_lr                | 0.005| 0.05| 0.005      | 0.005     |
| decay_lr               | 0.9  | 0.7 | 0.95       | 0.95      |
| backbone_activation    | SiLU | ReLU | SiLU       | ReLU      |
| backbone_dr            | 0.0  | 0.0 | 0.3        | 0.0       |
| forget_bias            | 1.2  | 1.2 | 4.7        | 0.6       |
| backbone_units         | 64   | 128 | 192        | 128       |
| backbone_layers        | 1    | 1   | 1          | 1         |
| weight_decay           | 3e-05| 3e-06| 5e-06      | 2e-06     |

Table S4: **Walker2D experiments.** Hyperparameters

| Parameter              | Cf-S | CfC | CfC-noGate | CfC-mmRNN |
|------------------------|------|-----|------------|-----------|
| clipnorm               | 10   | 1   | 1          | 10        |
| optimizer              | Adam | Adam| Adam       | Adam      |
| batch_size             | 128  | 256 | 128        | 128       |
| Hidden size            | 256  | 64  | 256        | 128       |
| epochs                 | 50   | 50  | 50         | 50        |
| base_lr                | 0.006| 0.02| 0.008      | 0.005     |
| decay_lr               | 0.95 | 0.95| 0.95       | 0.95      |
| backbone_activation    | SiLU | SiLU| LeCun Tanh| LeCun Tanh|
| backbone_dr            | 0.0  | 0.1 | 0.1        | 0.2       |
| forget_bias            | 5.0  | 1.6 | 2.8        | 2.1       |
| backbone_units         | 192  | 256 | 128        | 128       |
| backbone_layers        | 1    | 1   | 1          | 2         |
| weight_decay           | 1e-06| 1e-06| 3e-05     | 6e-06     |
S3 Reproducibility Matters

All code and data are provided in https://github.com/raminmh/CfC.