

Closed-form continuous-time neural networks

Received: 23 March 2022

Accepted: 5 October 2022

Published online: 15 November 2022

 Check for updates

Ramin Hasani ^{1,5} , Mathias Lechner^{1,2,5}, Alexander Amini¹,
Lucas Liebenwein ¹, Aaron Ray¹, Max Tschaikowski³, Gerald Teschl ⁴ &
Daniela Rus¹

Continuous-time neural networks are a class of machine learning systems that can tackle representation learning on spatiotemporal decision-making tasks. These models are typically represented by continuous differential equations. However, their expressive power when they are deployed on computers is bottlenecked by numerical differential equation solvers. This limitation has notably slowed down the scaling and understanding of numerous natural physical phenomena such as the dynamics of nervous systems. Ideally, we would circumvent this bottleneck by solving the given dynamical system in closed form. This is known to be intractable in general. Here, we show that it is possible to closely approximate the interaction between neurons and synapses—the building blocks of natural and artificial neural networks—constructed by liquid time-constant networks efficiently in closed form. To this end, we compute a tightly bounded approximation of the solution of an integral appearing in liquid time-constant dynamics that has had no known closed-form solution so far. This closed-form solution impacts the design of continuous-time and continuous-depth neural models. For instance, since time appears explicitly in closed form, the formulation relaxes the need for complex numerical solvers. Consequently, we obtain models that are between one and five orders of magnitude faster in training and inference compared with differential equation-based counterparts. More importantly, in contrast to ordinary differential equation-based continuous networks, closed-form networks can scale remarkably well compared with other deep learning instances. Lastly, as these models are derived from liquid networks, they show good performance in time-series modelling compared with advanced recurrent neural network models.

Continuous neural network architectures built by ordinary differential equations (ODEs)² are expressive models useful in modelling data with complex dynamics. These models transform the depth dimension of static neural networks and the time dimension of recurrent neural networks (RNNs) into a continuous vector field, enabling parameter

sharing, adaptive computations and function approximation for non-uniformly sampled data.

These continuous-depth (time) models have shown promise in density estimation applications^{3–6}, as well as modelling sequential and irregularly sampled data^{1,7–9}.

¹Massachusetts Institute of Technology, Cambridge, MA, USA. ²Institute of Science and Technology Austria, Klosterneuburg, Austria. ³Aalborg University, Aalborg, Denmark. ⁴University of Vienna, Vienna, Austria. ⁵These authors contributed equally: Ramin Hasani, Mathias Lechner.  e-mail: rhasani@mit.edu

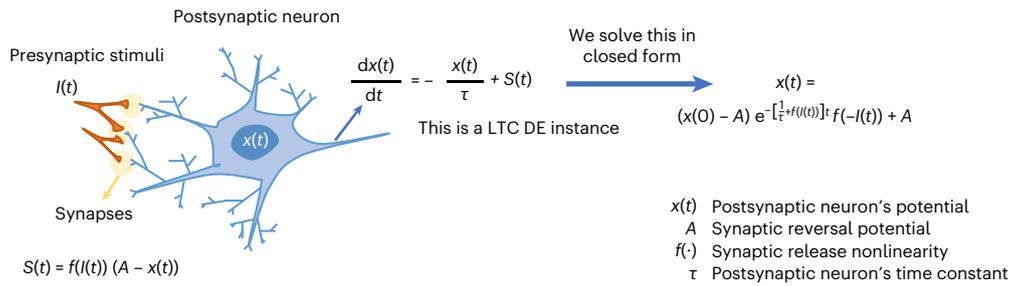


Fig. 1 | Neural and synapse dynamics. A postsynaptic neuron receives the stimuli $I(t)$ through a nonlinear conductance-based synapse model. Here, $S(t)$ stands for the synaptic current. The dynamics of the membrane potential of this postsynaptic neuron are given by the DE presented in the middle. This equation

is a fundamental building block of LTC networks¹, for which there is no known closed-form expression. Here, we provide an approximate solution for this equation which shows the interaction of nonlinear synapses with postsynaptic neurons in closed form.

While ODE-based neural networks with careful memory and gradient propagation design⁹ perform competitively with advanced discretized recurrent models on relatively small benchmarks, their training and inference are slow owing to the use of advanced numerical differential equation (DE) solvers¹⁰. This becomes even more troublesome as the complexity of the data, task and state space increases (that is, requiring more precision)¹¹, for instance, in open-world problems such as medical data processing, self-driving cars, financial time-series and physics simulations.

The research community has developed solutions for resolving this computational overhead and for facilitating the training of neural ODEs, for instance by relaxing the stiffness of a flow by state augmentation techniques^{4,12}, reformulating the forward pass as a root-finding problem¹³, using regularization schemes^{14–16} or improving the inference time of the network¹⁷.

Here, we derive a closed-form continuous-depth model that has the modelling capabilities of ODE-based models but does not require any solver to model data (Fig. 1).

Intuitively, in this work, we replace the integration (that is, solution) of a nonlinear DE describing the interaction of a neuron with its input nonlinear synaptic connections, with their corresponding nonlinear operators. This could be achieved in principle using functional Taylor expansions (in the spirit of the Volterra series)¹⁸. However, in the particular case of liquid time-constant (LTC) networks, we can leverage a closed-form expression for the system's response to input. This allows one to evaluate the system's response to exogenous input (I) and recurrent inputs from hidden states (x) as a function of time. One way of looking at this is to regard the closed-form solution as the application of a nonlinear forward operator to the inputs of each hidden state or neuron in the network, where the outputs of one neuron constitute the inputs for others. Effectively, this rests on approximating a conductance-based model with a neural mass model, of the kind used in the dynamic causal modelling of real neuronal networks¹⁹.

The proposed continuous neural networks yield considerably faster training and inference speeds while being as expressive as their ODE-based counterparts. We provide a derivation for the approximate closed-form solution to a class of continuous neural networks that explicitly models time. We demonstrate how this transformation can be formulated into a novel neural model and scaled to create flexible, performant and fast neural architectures on challenging sequential datasets.

Deriving an approximate closed-form solution for neural interactions

Two neurons interact with each other through synapses as shown in Fig. 1. There are three principal mechanisms for information propagation in natural brains that are abstracted away in the current building blocks of deep learning systems: (1) neural dynamics are typically continuous processes described by DEs (see the dynamics of $x(t)$ in Fig. 1), (2) synaptic release is much more than scalar weights, involving a

nonlinear transmission of neurotransmitters, the probability of activation of receptors and the concentration of available neurotransmitters, among other nonlinearities (see $S(t)$ in Fig. 1) and (3) the propagation of information between neurons is induced by feedback and memory apparatuses (see how $I(t)$ stimulates $x(t)$ through a nonlinear synapse $S(t)$ which also has a multiplicative difference of potential to the postsynaptic neuron accounting for a negative feedback mechanism). One could read $I(t)$ as a mixture of exogenous input to the (neural) network and presynaptic inputs from other neurons that result in a depolarization $x(t)$. This depolarization is mediated by the current $S(t)$ that depends upon depolarization and a reversal threshold A . LTC networks¹, which are expressive continuous-depth models obtained by a bilinear approximation²⁰ of a neural ODE formulation², are designed on the basis of these mechanisms. Correspondingly, we take their ODE semantics and approximate a closed-form solution for the scalar case of a postsynaptic neuron receiving an input stimulus from a presynaptic source through a nonlinear synapse.

To this end, we apply the theory of linear ODEs²¹ to analytically solve the dynamics of an LTC DE as shown in Fig. 1. We then simplify the solution to the point where there is one integral left to solve. This integral compartment, $\int_0^t f(I(s)) ds$ in which f is a positive, continuous, monotonically increasing and bounded nonlinearity, is challenging to solve in closed form since it has dependencies on an input signal $I(s)$ that is arbitrarily defined (such as real-world sensory readouts). To approach this problem, we discretize $I(s)$ into piecewise constant segments and obtain the discrete approximation of the integral in terms of the sum of piecewise constant compartments over intervals. This piecewise constant approximation inspired us to introduce an approximate closed-form solution for the integral $\int_0^t f(I(s)) ds$ that is provably tight when the integral appears as the exponent of an exponential decay, which is the case for LTCs. We theoretically justify how this closed-form solution represents LTCs' ODE semantics and is as expressive (Fig. 1).

Explicit time dependence

We then dissect the properties of the obtained closed-form solution and design a new class of neural network models we call closed-form continuous-depth networks (CfC). CfCs have an explicit time dependence in their formulation that does not require a numerical ODE solver to obtain their temporal rollouts. Thus, they maximize the trade-off between accuracy and efficiency of solvers. Formally, this property corresponds to obtaining lower time complexity for models without numerical instabilities and errors as illustrated in Table 1 (left). For example, Table 1 (left) shows that the complexity of a p th-order numerical ODE solver is $\mathcal{O}(Kp)$, where K is the number of ODE steps, while a CfC system (which has explicit time dependence) requires $\mathcal{O}(\bar{K})$, where \bar{K} is the exogenous input time steps, which are typically one to three orders of magnitude smaller than K . Moreover, the approximation error of a p th-order numerical ODE solver scales with $\mathcal{O}(e^{p+1})$, whereas CfCs are closed-form continuous-time systems, thus the notion of approximation error becomes irrelevant to them.

Table 1 | Computational complexity of models

Method	Time complexity		Sequence and time-step prediction complexity		
	Complexity	Local error	Model	Sequence prediction	Time-step prediction
<i>p</i> th-order solver	$\mathcal{O}(Kp)$	$\mathcal{O}(\epsilon^{p+1})$	RNN	$\mathcal{O}(nk)$	$\mathcal{O}(k)$
Adaptive-step solver	—	$\mathcal{O}(\bar{\epsilon}^{p+1})$	ODE-RNN	$\mathcal{O}(nkp)$	$\mathcal{O}(kp)$
Euler hypersolver	$\mathcal{O}(K)$	$\mathcal{O}(\delta\epsilon^2)$	Transformer	$\mathcal{O}(n^2k)$	$\mathcal{O}(nk)$
<i>p</i> th-order hypersolver	$\mathcal{O}(Kp)$	$\mathcal{O}(\delta\epsilon^{p+1})$	CfC	$\mathcal{O}(nk)$	$\mathcal{O}(k)$
CfC (current work)	$\mathcal{O}(\bar{K})$	Not relevant			

Left: The time complexity of the process to compute K solver steps. ϵ is step size. $\bar{\epsilon}$ is the maximum step size and $\delta \ll 0$. \bar{K} is the time steps for CfCs corresponding to the input time step, which is typically one to three orders of magnitude smaller than K . The left portion is reproduced with permission from ref. ¹⁷. Right: Sequence and time-step prediction complexity. n is the sequence length. k is the number of hidden units. p is the order of the ODE solver.

This explicit time dependence allows CfCs to perform computations at least one order of magnitude faster in terms of training and inference time compared with their ODE-based counterparts, without loss of accuracy.

Sequence and time-step prediction efficiency

In sequence modelling tasks, one can perform predictions based on an entire sequence of observations, or perform auto-regressive modelling where the model predicts the next time-step output given the current time-step input. Table 1 (right) depicts the time complexity of different neural network instances at inference, for a given sequence of length n and a neural network of k number of hidden units. We observe that the complexity of ODE-based networks and Transformer modules is at least an order of magnitude higher than that of discrete RNNs and CfCs in both sequence prediction and auto-regressive modelling (time-step prediction) frameworks.

This is desirable because not only do CfCs establish a continuous flow similar to ODE models¹ to achieve better expressivity in representation learning but they do so with the efficiency of discrete RNN models.

CfCs: flexible deep models for sequential tasks

Additionally, CfCs are equipped with novel time-dependent gating mechanisms that explicitly control their memory. CfCs are as expressive as their ODE-based peers and can be supplied with mixed memory architectures⁹ to avoid gradient issues in sequential data processing applications with long-range dependences. Beyond accuracy and performance metrics, our results indicate that, when considering accuracy per compute time, CfCs exhibit over 150 fold improvements over ODE-based compartments. We perform a diverse set of advanced time-series modelling experiments and present the performance and speed gain achievable by using CfCs in tasks with long-term dependences, irregular data and modelling physical dynamics, among others.

Deriving a closed-form solution

In this section, we derive an approximate closed-form solution for 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 CfC models. In this regard, we define the LTC semantics. We then state the main theorem that computes a closed-form approximation of a given LTC system for the scalar case. To prove the theorem, we first find the integral solution of the given LTC ODE system. We then compute a closed-form analytical solution for the integral solution for the case of piecewise constant inputs. Afterward, we generalize the closed-form solution of the piecewise constant inputs to the case of arbitrary inputs with our novel approximation and finally provide sharpness results (that is, measure the rate and accuracy of an approximation error) for the derived solution.

The hidden state of an LTC network is determined by the solution of the following initial value problem (IVP)¹:

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

where at a time step t , $\mathbf{x}^{(D \times 1)}(t)$ defines the hidden state of a LTC layer with D cells, and $\mathbf{I}^{(m \times 1)}(t)$ is an exogenous input to the system with m features. Here, $w_t^{(D \times 1)}$ is a time-constant parameter vector, $A^{(D \times 1)}$ is a bias vector, f is a neural network parametrized by θ and \odot is the Hadamard product. The dependence of $f(\cdot)$ on $\mathbf{x}(t)$ denotes the possibility of having recurrent connections.

The full proof of theorem 1 is given in Methods. The theorem formally demonstrates that the approximated closed-form solution for the given LTC system is given by equation (2) and that this approximation is tightly bounded with bounds given in the proof.

In the following, we show an illustrative example of this tightness result in practice. To do this, we first present an instantiation of LTC networks and their approximate closed-form expressions. Extended Data Fig. 1 shows a liquid network with two neurons and five synaptic connections. The network receives an input signal $I(t)$. Extended Data Fig. 1 further derives the DE expression for the network along with its closed-form approximate solution. In general, it is possible to compile an LTC network into its closed-form expression as illustrated in Extended Data Fig. 1. This compilation can be performed using Algorithm 1 provided in Methods.

Theorem 1

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

$$x(t) \approx (x_0 - A)e^{-[w_t + f(I(t), \theta)]t} f(-I(t), \theta) + A. \tag{2}$$

Tightness of the closed-form solution in practice

Figure 2 shows an LTC-based network trained for autonomous driving²². The figure further illustrates how close the proposed solution fits the actual dynamics exhibited from a single-neuron ODE given the same parametrization. The details of this experiment are given in Methods.

We next show how to design a novel neural network instance inspired by this closed-form solution that has well-behaved gradient properties and approximation capabilities.

Designing CfC models from the solution

Leveraging the scalar closed-form solution expressed by equation (2), we can now distil this model into a neural network model that can be trained at scale. The solution provides a grounded theoretical basis

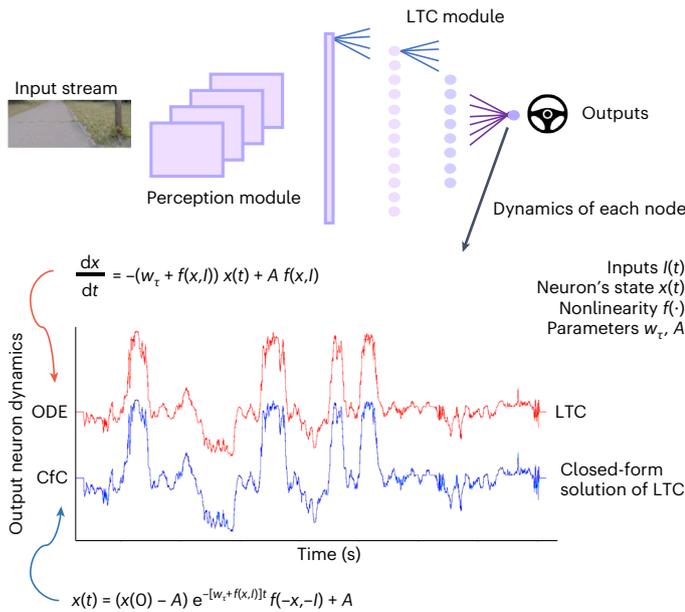


Fig. 2 | Tightness of the closed-form solution in practice. We approximate a closed-form solution for LTC networks¹ while largely preserving the trajectories of their equivalent ODE systems. We develop our solution into CfC models that are at least 100 fold faster than neural ODEs at both training and inference on complex time-series prediction tasks.

for solving scalar continuous-time dynamics, and it is important to translate this theory into a practical neural network model which can be integrated into larger representation learning systems equipped with gradient descent optimizers. 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, $\mathbf{x}(t)^{(D \times 1)}$ with D hidden units at each time step t , can be obtained explicitly as

$$\mathbf{x}(t) = B \odot e^{-[w_t + f(\mathbf{x}, \mathbf{I}; \theta)]t} \odot f(-\mathbf{x}, -\mathbf{I}; \theta) + A, \quad (3)$$

where $B^{(D)}$ collapses $(x_0 - A)$ of equation (2) into a parameter vector. $A^{(D)}$ and $w_t^{(D)}$ are system’s parameter vectors, while $\mathbf{I}(t)^{(m \times 1)}$ is an m -dimensional input at each time step t , f is a neural network parameterized by $\theta = \{W_{Lx}^{(m \times D)}, W_{xx}^{(D \times D)}, b_x^{(D)}\}$ and \odot is the Hadamard (element-wise) product. While the neural network presented in equation (3) can be proven to be a universal approximator as it is an approximation of an ODE system^{1,2}, in its current form, it has trainability issues which we point out and resolve shortly.

Resolving the gradient issues

The exponential term in equation (3) drives 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 descent²³. To reduce this 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 $t \rightarrow \infty$. However, unlike exponential decay, its transition happens much more smoothly, 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 $f(-\mathbf{x}, -\mathbf{I}; \theta)$ 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. To obtain a more general network architecture, we allow the nonlinearity $f(-\mathbf{x}, -\mathbf{I}; \theta)$ present in equation (3) to have both shared (backbone) and independent ($g(\cdot)$) network compartments.

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 $t \rightarrow -\infty$ and $t \rightarrow \infty$ of the ODE trajectory.

Backbone

Instead of learning all three neural network instances f , g and h separately, we have 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. More importantly, this architectural prior enables two simultaneous benefits: (1) Through the shared backbone, a coupling between the time constant of the system and its state nonlinearity is established that exploits causal representation learning evident in a liquid neural network^{1,24}. (2) through separate head network layers, the system has the ability to explore temporal and structural dependences independently of each other.

These modifications result in the CfC neural network model:

$$\mathbf{x}(t) = \underbrace{\sigma(-f(\mathbf{x}, \mathbf{I}; \theta_f)t)}_{\text{time-continuous gating}} \odot g(\mathbf{x}, \mathbf{I}; \theta_g) + \underbrace{[1 - \sigma(-f(\mathbf{x}, \mathbf{I}; \theta_f)t)]}_{\text{time-continuous gating}} \odot h(\mathbf{x}, \mathbf{I}; \theta_h). \quad (4)$$

The CfC architecture is illustrated in Extended Data Fig. 2. The neural network instances could be selected arbitrarily. The time complexity of the algorithm is equivalent to that of discretized recurrent networks²⁵, being at least one order of magnitude faster than ODE-based networks.

The procedure to account for the explicit time dependence

CfCs are continuous-depth models that can set their temporal behaviour based on the task under test. For time-variant datasets (for example, irregularly sampled time series, event-based data and sparse data), the 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 as many times as the batch length, with equidistant intervals within two hyperparameters a and b .

Experiments with CfCs

We now assess the performance of CfCs in a series of sequential data processing tasks compared with advanced, recurrent models. We first approach solving conventional sequential data modelling tasks (for example, bit-stream prediction, sentiment analysis on text data, medical time-series prediction, human activity recognition, sequential image processing and robot kinematics modelling), and compare CfC variants with an extensive set of advanced RNN baselines. We then evaluate how CfCs compare with LTC-based neural circuit policies (NCPs)²² in real-world autonomous lane-keeping tasks.

CfC network variants

To evaluate the proposed modifications we applied to the closed-form solution network described by equation (3), we test four variants of the CfC architecture: (1) the closed-form solution network (Cf-S) obtained by equation (3), (2) the CfC without the second gating mechanism (CfC-noGate), a variant that does not have the $1 - \sigma$ instance shown in Extended Data Fig. 2, (3) The CfC model (CfC) expressed by equation (4) and (4) the CfC wrapped inside a mixed memory architecture (that is, where the CfC defines the memory state of an RNN, for instance, a long short-term memory (LSTM)), a variant we call CfC-mmRNN.

Each of these four proposed variants leverages our proposed solution and thus is at least one order of magnitude faster than continuous-time ODE models.

To investigate their representation learning power, in the following we extensively evaluate CfCs on a series of sequence modelling tasks. The objective is to test the effectiveness of the CfCs in learning spatiotemporal dynamics, compared with a wide range of advanced models.

Baselines

We compare CfCs with a diverse set of advanced algorithms developed for sequence modelling by both discretized and continuous mechanisms. These baselines are given in full in Methods.

Human activity recognition

The human activity dataset⁷ contains 6,554 sequences of humans demonstrating activities such as walking, lying, sitting, etc. The input space is formed of 561-dimensional inertial sensor measurements per time step, recorded from the user's smartphone²⁶, being categorized into six group of activities (per time step) as output.

We set up our dataset split (training, validation and test) to carefully reflect the modifications made by Rubanova et al.⁷ on this task. The results of this experiment are reported in Table 2. We observe that not only do the CfC variants Cf-S, CfC-noGate and CfC-mmRNN outperform other models with a high margin, but they do so with a speed-up of more than 8,752% over the best-performing ODE-based instance (Latent-ODE-ODE). The reason for such a large speed difference is the complexity of the dataset dynamics that causes the ODE solvers of ODE-based models such as Latent-ODE-ODE to compute many steps upon stiff dynamics. This issue does not exist for closed-form models as they do not use any ODE solver to account for dynamics. The hyperparameter details of this experiment are provided in Extended Data Fig. 3.

Physical dynamics modelling

The Walker2D dataset consists of kinematic simulations of the MuJoCo physics engine²⁷ (see Methods for more details). As shown in Table 3, CfCs outperform the other baselines by a large margin, supporting 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. The hyperparameter details of this experiment are provided in Extended Data Fig. 3.

Event-based sequential image processing

We next assess the performance of CfCs on a challenging sequential image processing task. This task is generated from the sequential modified National Institute of Standards and Technology (MNIST) dataset following the steps described in Methods. Moreover, the hyperparameter details of this experiment are provided in Extended Data Fig. 4.

Table 4 summarizes the results on this event-based sequence classification task. We observe that models such as ODE-RNN, CT-RNN, GRU-ODE and LSTMs struggle to learn a good representation of the input data and therefore show poor performance. In contrast, RNNs endowed with explicit memory, such as bi-directional RNNs, GRU-D, Lipschitz RNN, coRNN, CT-LSTM and ODE-LSTM, perform well on this task. All CfC variants perform well on this task and establish the state-of-the-art on this task, with CfC-mmRNN achieving 98.09% and CfC-noGate achieving 96.99% accuracy in classifying irregularly sampled sequences. It is worth mentioning that they do so around 200–400% faster than ODE-based models such as GRU-ODE and ODE-RNN.

Regularly and irregularly sampled bit-stream XOR

The bit-stream XOR dataset⁹ considers the classification of bit streams by implementing an XOR function in time. That is, each item in the sequence contributes equally to the correct output. The details are given in Methods.

Table 2 | Human activity recognition, per time-step classification

Model	Accuracy (%)	Time per epoch (min)
*RNN-Impute ⁷	79.50±0.8	0.38
*RNN- Δt ⁷	79.50±0.8	0.45
*RNN-Decay ⁷	80.00±1.0	0.39
*GRU-D ⁵¹	80.60±0.7	0.15
*RNN-VAE ⁷	34.30±4.0	2.63
*Latent-ODE-RNN ⁷	83.50±1.0	7.71
*ODE-RNN ⁷	82.90±1.6	3.15
*Latent-ODE-ODE ⁷	84.60±1.3	8.49
Cf-S (current work)	87.04±0.47	0.097
CfC-noGate (current work)	85.57±0.34	0.093
CfC (current work)	84.87±0.42	0.084
CfC-mmRNN (current work)	85.97±0.25	0.128

Numbers represent mean±s.d. ($n=5$). The performance of the models marked by * is reported from ref. ⁷. Bold values indicate the highest accuracy and best time per epoch (min).

Extended Data Fig. 5 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 CfC, 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, CfC and CfC-mmRNNs. ODE-based RNNs cannot solve the event-based encoding tasks regardless of their choice of solvers, as they have vanishing/exploding gradient issues⁹. The hyperparameter details of this experiment are provided in Extended Data Fig. 4.

PhysioNet Challenge

The PhysioNet Challenge 2012 dataset considers the prediction of the mortality of 8,000 patients admitted to the intensive care unit. The features represent time series of medical measurements taken during the first 48 h after admission. The data are irregularly sampled in time and over features, that is, only a subset of the 37 possible features is given at each time point. We perform the same test–train split and preprocessing as in ref. ⁷, and report the area under the curve (AUC) on the test set as a metric in Extended Data Fig. 6. We observe that CfCs perform competitively to other baselines while performing 160 times faster in terms of training time compared with ODE-RNN and 220 times compared with continuous latent models. CfCs are also, on average, three times faster than advanced discretized gated recurrent models. The hyperparameter details of this experiment are provided in Extended Data Fig. 7.

Sentiment analysis using IMDB

The Internet Movie Database (IMDB) sentiment analysis dataset²⁸ consists of 25,000 training and 25,000 test sentences (see Methods for more details). Extended Data Fig. 8 shows how CfCs equipped with mixed memory instances outperform advanced RNN benchmarks. The hyperparameter details of this experiment are provided in Extended Data Fig. 7.

Performance of CfCs in autonomous driving

In this experiment, our objective is to evaluate how robustly CfCs learn to perform autonomous navigation in comparison with their ODE-based counterparts, LTC networks. The task is to map incoming

Table 3 | Per time-step regression

Model	Mean Squared Error (MSE)	Time per epoch (min)
†ODE-RNN ⁷	1.904±0.061	0.79
†CT-RNN ⁴⁸	1.198±0.004	0.91
†AugmentedLSTM ⁴⁴	1.065±0.006	0.10
†CT-GRU ⁴⁹	1.172±0.011	0.18
†RNN-Decay ⁷	1.406±0.005	0.16
†Bi-directional RNN ⁵³	1.071±0.009	0.39
†GRU-D ⁵¹	1.090±0.034	0.11
†PhasedLSTM ⁵²	1.063±0.010	0.25
†GRU-ODE ⁷	1.051±0.018	0.56
†CT-LSTM ⁵⁰	1.014±0.014	0.31
†ODE-LSTM ⁹	0.883±0.014	0.29
coRNN ⁵⁷	3.241±0.215	0.18
Lipschitz RNN ⁵⁸	1.781±0.013	0.17
LTC ¹	0.662±0.013	0.78
Transformer ³⁶	0.761±0.032	0.80
Cf-S (current work)	0.948±0.009	0.12
CfC-noGate (current work)	0.650±0.008	0.21
CfC (current work)	0.643±0.006	0.08
CfC-mmRNN (current work)	0.617±0.006	0.34

Modelling the physical dynamics of a walker agent in simulation. Numbers present mean±s.d. (n=5). The performance of the models marked by † is reported from ref. ⁹. Bold values indicate the lowest error and best time per epoch (min).

high-dimensional pixel observations to steering curvature commands. The details of this experiment are given in Methods.

We observe that CfCs similar to NCPs demonstrate a consistent attention pattern in each subtask while maintaining their attention profile under heavy noise as depicted in Extended Data Fig. 10c. This is while the attention profile of other networks such as CNNs and LSTMs is hindered by added input noise (Extended Data Fig. 10c).

This experiment empirically validates that CfCs possess similar robustness properties to their ODE counterparts, that is, LTC-based networks. Moreover, similar to NCPs, CfCs are parameter efficient. They performed the end-to-end autonomous lane-keeping task with around 4,000 trainable parameters in their RNN component (Extended Data Fig. 9).

Scope, discussion and conclusions

We introduce a closed-form continuous-time neural model built from an approximate closed-form solution of LTC networks that possess the strong modelling capabilities of ODE-based networks while being notably faster, more accurate, and stable. These closed-form continuous-time models achieve this by explicit time-dependent gating mechanisms and having a LTC modulated by neural networks. A discussion of related research on continuous-time models is given in Methods.

For large-scale time-series prediction tasks, and where closed-loop performance matters²⁴, CfCs can bring great value. This is because they capture the flexible, causal and continuous-time nature of ODE-based networks, such as LTC networks, while being more efficient. A discussion on how to use different variants of CfCs is provided in Methods. On the other hand, implicit ODE- and partial differential equation-based models^{17,29–31} can be beneficial in solving continuously defined physics problems and control tasks. Moreover, for generative modelling, continuous normalizing flows built by ODEs are the suitable choice of model as they ensure invertibility, unlike CfCs². This is because DEs

Table 4 | Event-based sequence classification on irregularly sequential MNIST

Model	Accuracy (%)	Time per epoch (min)
ODE-RNN ⁷	72.41±1.69	14.57
CT-RNN ⁴⁸	72.05±0.71	17.30
Augmented LSTM ⁴⁴	82.10±4.36	2.48
CT-GRU ⁴⁹	87.51±1.57	3.81
RNN-Decay ⁷	88.93±4.06	3.64
Bi-directional RNN ⁷	94.43±0.23	8.097
GRU-D ⁵¹	95.44±0.34	3.42
PhasedLSTM ⁵²	86.79±1.57	5.69
GRU-ODE ⁷	80.95±1.52	6.76
CT-LSTM ⁵⁰	94.84±0.17	3.84
coRNN ⁵⁷	94.44±0.24	3.90
Lipschitz RNN ⁵⁸	95.92±0.16	3.86
ODE-LSTM ⁹	95.73±0.24	6.35
Cf-S (current work)	95.23±0.16	2.73
CfC-noGate (current work)	96.99±0.30	3.36
CfC (current work)	95.42±0.21	3.62
CfC-mmRNN (current work)	98.09±0.18	5.50

Test accuracy shown as mean±s.d. (n=5). Bold values indicate the highest accuracy and best time per epoch (min).

guarantee invertibility (that is, under uniqueness conditions⁶, one can run them backwards in time). CfCs only approximate ODEs and therefore no longer necessarily form a bijection³².

What are the limitations of CfCs?

CfCs might express vanishing gradient problems. To avoid this, for tasks that require long-term dependences, it is better to use them together with mixed memory networks⁹ (as in the CfC variant CfC-mmRNN) or with proper parametrization of their transition matrices^{33,34}. Moreover, we speculate that inferring causality from ODE-based networks might be more straightforward than a closed-form solution²⁴. It would also be beneficial to assess whether verifying a continuous neural flow³⁵ is more tractable by using an ODE representation of the system or its closed form.

For problems such as language modelling where a large amount of sequential data and substantial computational resources are available, transformers³⁶ and their variants are great choices of models. CfCs could bring value when: (1) data have limitations and irregularities (for example, medical data, financial time series, robotics³⁷ and closed-loop control, and multi-agent autonomous systems in supervised and reinforcement learning schemes³⁸), (2) the training and inference efficiency of a model is important (for example, embedded applications^{39–41}) and (3) when interpretability matters⁴².

Ethics statement

All authors acknowledge the Global Research Code on the development, implementation and communication of this research. For the purpose of transparency, we have included this statement on inclusion and ethics. This work cites a comprehensive list of research from around the world on related topics.

Methods

Proof of theorem 1

Proof. In the single-dimensional case, the IVP in equation (1) becomes linear in x as follows:

$$\frac{d}{dt}x(t) = -[w_\tau + f(I(t))] \cdot x(t) + Af(I(t)). \tag{5}$$

Therefore, we can use the theory of linear ODEs to obtain an integral closed-form solution (section 1.10 in ref. ²¹) consisting of two nested integrals. The inner integral can be eliminated by means of integration by substitution⁴³. The remaining integral expression can then 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 arbitrary continuous input signal I and has a positive, bounded, continuous and monotonically increasing nonlinearity f :

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

Assumption. We assumed a second constant value w_τ in the above representation of a single LTC neuron. This is done to introduce symmetry in the structure of the ODE, yielding a simpler expression for the solution. The inclusion of this second constant may appear to profoundly alter the dynamics. However, as shown below, numerical experiments suggest that this simplifying assumption has a marginal effect on the ability to approximate LTC cell dynamics.

Using the variation of constants formula (section 1.10 in ref. ²¹), we obtain after some simplifications:

$$x(t) = (x(0) - A)e^{-w_\tau t - \int_0^t f(I(s))ds} + A. \tag{6}$$

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

$$0 = \tau_0 < \tau_1 < \tau_2 < \dots < \tau_{n-1} < \tau_n = \infty,$$

such that $\tau_1, \dots, \tau_{n-1} \in \mathbb{R}$ 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

$$\int_0^t f(I(s)) ds = f(\gamma_k)(t - \tau_k) + \sum_{i=0}^{k-1} f(\gamma_i)(\tau_{i+1} - \tau_i), \tag{7}$$

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

$$x(t) = (x(0) - A)e^{-w_\tau t} e^{-\sum_{i=0}^{k-1} f(\gamma_i)(\tau_{i+1} - \tau_i) - f(\gamma_k)(t - \tau_k)} + A, \tag{8}$$

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 input⁴³, a tight approximation may require a large number of discretization points τ_1, \dots, τ_n . We address this next.

Analytical LTC approximation for general inputs

Inspired by equations (7) and (8), the next result provides an analytical approximation of $x(t)$.

Lemma 1

For any Lipschitz continuous, positive, monotonically increasing and bounded f and continuous input signal $I(t)$, we approximate $x(t)$ in equation (6) as

$$\tilde{x}(t) = (x(0) - A)e^{-[w_\tau + f(I(t))]t} f(-I(t)) + A. \tag{9}$$

Then, $|x(t) - \tilde{x}(t)| \leq |x(0) - A|e^{-w_\tau t}$ 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 $\sup \left\{ \frac{1}{c} |x(t) - \tilde{x}(t)| : [0; t] \rightarrow \mathbb{R} \right\} = e^{-w_\tau t}$.*
2. *For any $t \geq 0$, we have $\inf \left\{ \frac{1}{c} |x(t) - \tilde{x}(t)| : [0; t] \rightarrow \mathbb{R} \right\} = e^{-w_\tau t} (e^{-t} - 1)$.*

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.

The full proof is given in the next section. Lemma 1 demonstrates that the integral solution we obtained and shown in equation (6) is tightly close to the approximate closed-form solution we proposed in equation (9). Note that, as w_τ is positively defined, the derived bound between equations (6) and (9) ensures an exponentially decaying error as time goes by. Therefore, we have the statement of the theorem. \square

Proof of lemma 1

We start by noting that

$$x(t) - \tilde{x}(t) = c e^{-w_\tau t} \left[e^{-\int_0^t f(I(s))ds} - e^{-f(I(t))t} f(-I(t)) \right].$$

Since $0 \leq f \leq 1$, we conclude that $e^{-\int_0^t f(I(s))ds} \in [0; 1]$ and $e^{-f(I(t))t} f(-I(t)) \in [0; 1]$. This shows that $|x(t) - \tilde{x}(t)| \leq |c|e^{-w_\tau t}$. 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

$$e^{-\int_0^t f(I(s))ds} - e^{-f(I(t))t} f(-I(t)) \geq e^{-\varepsilon t - \delta \cdot 1} - e^{-(1-\varepsilon)t} \varepsilon \rightarrow 1, \text{ when } \varepsilon, \delta \rightarrow 0$$

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

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

where L is the Lipschitz constant of f , and the last identity is due to the dominated convergence theorem⁴³. To see statement 2, we first note that the negation of the signal $-I$ provides us with

$$e^{-\int_0^t f(-I(s))ds} - e^{-f(-I(t))t} f(I(t)) \leq e^{-(1-\varepsilon)(t-\delta) - \delta \cdot 0} - e^{-\varepsilon t} (1 - \varepsilon) \rightarrow e^{-t} - 1,$$

if $\varepsilon, \delta \rightarrow 0$. 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 $I, I' : [0; t] \rightarrow \mathbb{R}$. \square

Compiling LTC architectures into their closed-form equivalent

In general, it is possible to compile the architecture of an LTC network into its closed-form version. This compilation allows us to speed up the training and inference time of ODE-based networks as the closed-form variant does not require complex ODE solvers to compute outputs. Algorithm 1 provides the instructions on how to transfer the architecture of an LTC network into its closed-form variant. Here, W_{adj} corresponds to the adjacency matrix that maps exogenous inputs to hidden states and the coupling among hidden states.

This adjacency matrix can have an arbitrary sparsity (that is, there is no need to use a directed acyclic graph for the coupling between neurons).

Algorithm 1. Translate the architecture of an LTC network into its closed-form variant

```

Inputs: LTC inputs  $\mathbf{I}^{(N \times T)}(t)$ , the activity  $\mathbf{x}^{(H \times T)}(t)$  and initial states  $\mathbf{x}^{(H \times 1)}(0)$  of LTC neurons and the adjacency matrix for synapses  $W_{Adj}^{(N+H) \times (N+H)}$ 
LTC ODE solver with step of  $\Delta t$ 
time-instance vectors of inputs,  $\mathbf{t}_{k(t)}^{(1 \times T)}$ 
time-instance of LTC neurons  $\mathbf{t}_{k(t)}$   $\nabla$  time might be sampled irregularly
LTC neuron parameter  $\tau^{(H \times 1)}$ 
LTC network synaptic parameters  $\{\sigma^{(N \times H)}, \mu^{(N \times H)}, A^{(N \times H)}\}$ 
Outputs: LTC closed-form approximation of hidden state neurons,  $\hat{\mathbf{x}}^{(N \times T)}(t)$ 
 $\mathbf{x}_{pre}(t) = W_{Adj} \times [I_0 \dots I_{N'}, x_0 \dots x_H]$   $\nabla$  all presynaptic signals to nodes
for  $i$ th neuron in neurons 1 to  $H$  do
  for  $j$  in Synapses to  $i$ th neuron do
     $\hat{x}_i += (x_0 - A_{ij}) e^{\left[ -\tau_{x(i)} \odot \left( \frac{1}{1 + e^{(-\sigma_{ij}(x_{pre_{ij}} - \mu_{ij})}}) \right) \right]} \odot \frac{1}{1 + e^{(\sigma_{ij}(x_{pre_{ij}} - \mu_{ij})}}) + A_{ij}$ 
  end for
end for
return  $\hat{\mathbf{x}}(t)$ 
    
```

Experimental details of the tightness experiment

We took a trained NCP²², which consists of a perception module and an LTC-based network¹ that possesses 19 neurons and 253 synapses. The network was trained to steer a self-driving vehicle autonomously. We used recorded real-world test runs of the vehicle for a lane-keeping task governed by this network. The records included the inputs, outputs and all the LTC neurons’ activities and parameters. To perform a numerical evaluation of our theory to determine whether our proposed closed-form solution for LTC neurons is good enough in practice as well, we inserted the parameters for individual neurons and synapses of the DEs into the closed-form solution (similar to the representations shown in Extended Data Fig. 1b,c) and emulated the structure of the ODE-based LTC networks. We then visualized the output neuron’s dynamics of the ODE (in blue) and of the closed-form solution (in red). As illustrated in Fig. 2, we observed that the behaviour of the ODE is captured by the closed-form solution with a mean squared error of 0.006. This experiment provides numerical evidence for the tightness results presented in our theory. Hence, the closed-form solution contains the main properties of liquid networks in approximating dynamics.

Baseline models

The example baseline models considered include some variations of classical auto-regressive RNNs, such as an RNN with concatenated Δt (RNN- Δt), a recurrent model with moving average on missing values (RNN-impute), RNN-Decay⁷, LSTMs⁴⁴ and gated recurrent units (GRUs)⁴⁵. 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 ref. 7.

Furthermore, we include models such as interpolation prediction networks (IP-Net)⁴⁶, set functions for time series (SeFT)⁴⁷, CT-RNN⁴⁸, CT-GRU⁴⁹, CT-LSTM⁵⁰, GRU-D⁵¹, PhasedLSTM⁵² and bi-directional RNNs⁵³. Finally, we benchmarked CfCs against competitive recent RNN architectures with the premise of tackling long-term dependencies, such as Legandre memory units⁵⁴, high-order polynomial projection operators (Hippo)⁵⁵, orthogonal recurrent models (expRNNs)⁵⁶, mixed memory RNNs such as ODE-LSTMs⁹, coupled oscillatory RNNs (coRNN)⁵⁷ and Lipschitz RNN⁵⁸.

Experimental details for the Walker2D dataset

This task is designed based on the Walker2d-v2 OpenAI gym⁵⁹ environment using data from four different stochastic policies. The objective is to predict the physics state in 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.

Description of the event-based MNIST experiment

We first sequentialize each image by transforming each 28×28 image into a long series of length 784. The objective is to predict the class corresponding to each image from the long input sequence. Advanced sequence modelling frameworks such as coRNN⁵⁷, Lipschitz RNN⁵⁸ and mixed memory ODE-LSTM⁹ can solve this task very well with accuracy of up to 99.0%. However, we aim to make the task even more challenging by sparsifying the input vectors with event-like irregularly sampled mechanisms. To this end, in each vector input (that is, flattened image), we transform each consecutive occurrence of values into one event. For instance, within the long binary vector of an image, the sequence 1, 1, 1, 1 is transformed to (1, $t = 4$) (ref. 9). This way, sequences of length 784 are condensed into event-based irregularly sampled sequences of length 256 that are far more challenging to handle than equidistance input signals. A recurrent model now has to learn to memorize input information of length 256 while keeping track of the time lags between the events.

Description of the event-based XOR encoding experiment

The bit streams are provided in densely sampled and event-based sampled formats. The densely sampled version simply represents an incoming bit as an input event. The event-based sampled version transmits only bit changes to the network, that is, multiple equal bits are 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.

Experimental details of the IMDB dataset experiment

Each sentence corresponds to either positive or negative sentiment. We tokenize the sentences in a word-by-word fashion with a vocabulary consisting of the 20,000 words occurring most frequently in the dataset. We map each token to a vector using trainable word embedding. The word embedding is initialized randomly. No pretraining of the network or word embedding is performed.

Setting of the driving experiment

It has been shown that models based on LTC networks are more robust when trained on offline demonstrations and tested online in closed loop with their environments, in many end-to-end robot control tasks such as mobile robots⁶⁰, autonomous ground vehicles²² and autonomous aerial vehicles^{24,61}. This robustness in decision-making (that is, their flexibility in learning and executing the task from demonstrations despite environmental or observational disturbances and distributional shifts) originates from their model semantics that formally reduces to dynamic causal models^{20,24}. Intuitively, LTC-based networks learn to extract a good representation of the task they are given (for example, their attention maps indicate what representation they have learned to focus on the road with more attention to the road’s horizon) and maintain this understanding under heavy distribution shifts. An example is illustrated in Extended Data Fig. 10.

In this experiment, we aim to investigate whether CfC models and their variants, such as CfC-mmRNN, possess this robustness characteristic (maintaining their attention map under distribution shifts and added noise), similar to their ODE counterparts (LTC-based networks called NCPs²²).

We start by training neural network architectures that possess a convolutional head stacked with the choice of RNN. The RNN

compartment of the networks is replaced by LSTM networks, NCPs²², Cf-S, CfC-NoGate and CfC-mmRNN. We also trained a fully convolutional neural network for the sake of proper comparison. Our training pipeline followed an imitation learning approach with paired pixel-control data from a 30 Hz BlackFly PGE-23S3C red–green–blue camera, collected by a human expert driver across a variety of rural driving environments, including different times of day, weather conditions and seasons of the year. The original 3 h data set was further augmented to include off-orientation recovery data using a privileged controller⁶² and a data-driven view synthesizer⁶³. The privileged controller enabled the training of all networks using guided policy learning⁶⁴. After training, all networks were transferred on-board our full-scale autonomous vehicle (Lexus RX450H, retrofitted with drive-by-wire capability). The vehicle was consistently started at the centre of the lane, initialized with each trained model and run to completion at the end of the road. If the model exited the bounds of the lane, a human safety driver intervened and restarted the model from the centre of the road at the intervention location. All models were tested with and without noise added to the sensory inputs to evaluate robustness.

The testing environment consisted of 1 km of private test road with unlabelled lane markers, and we observed that all trained networks were able to successfully complete the lane-keeping task at a constant velocity of 30 km h⁻¹. Extended Data Fig. 10 provides an insight into how these networks reach driving decisions. To this end, we computed the attention of each network while driving by using the VisualBackProp algorithm⁶⁵.

Related works on continuous-time models

Continuous-time models. Machine learning, control theory and dynamical systems merge at models with continuous-time dynamics^{60,66–69}. In a seminal work, Chen et al.^{2,7} revived the class of continuous-time neural networks^{48,70}, with neural ODEs. These continuous-depth models give rise to vector field representations and a set of functions that were not possible to generate before with discrete neural networks. These capabilities enabled flexible density estimation^{3–5,71,72} as well as performer modelling of sequential and irregularly sampled data^{1,7–9,58}. 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 modelling task increases, ODE-based networks demand a more advanced solver that largely impacts their efficiency¹⁷, stability^{13,15,73–75} and performance¹. A large body of research has studied how to improve the computational overhead of these solvers, for example, by designing hypersolvers¹⁷, deploying augmentation methods^{4,12}, pruning⁶ or regularizing the continuous flows^{14–16}. To enhance the performance of an ODE-based model, especially in time-series modelling tasks⁷⁶, solutions for stabilizing their gradient propagation have been provided^{9,58,77}. In this work, we showed that CfCs improve the scalability, efficiency and performance of continuous-depth neural models.

Which CfC variants to choose in different applications

Our extensive experimental results demonstrate that different CfC variants, namely Cf-S, CfC-noGate, vanilla CfC and CfC-mmRNN, achieve comparable results to each other while one comes on top depending on the nature of the data set. We suggest using CfC in most cases where the sequence length is up to a couple of hundred steps. To capture longer-range dependences, we recommend CfC-mmRNN. The Cf-S variant is effective when we aim to obtain the fastest inference time. CfC-noGate could be tested as a hyperparameter when using the vanilla CfC as the primary choice of model.

Description of hyperparameters

The hyperparameters used in our experimental results are as follows:

- clipnorm: the gradient clipping norm (that is, the global norm clipping threshold)
- optimizer: the weight update preconditioner (for example, Adam, Stochastic Gradient Descent with momentum, etc.)
- batch_size: the number of samples used to compute the gradients
- hidden_size: the number of RNN units
- epochs: the number of passes over the training dataset
- base_lr: the initial learning rate
- decay_lr: the factor by which the learning rate is multiplied after each epoch
- backbone_activation: the activation function of the backbone layers
- backbone_dr: the dropout rate of the backbone layers
- forget_bias: the forget gate bias (for mmRNN and LSTM)
- backbone_units: the number of hidden units per backbone layer
- backbone_layers: the number of backbone layers
- weight_decay: the L2 weight regularization factor
- τ_{data} : the constant factor by which the elapsed time input is multiplied (default value 1)
- init: the gain of the Xavier uniform distribution for the weight initialization (default value 1)

Data availability

All data and materials used in the analysis are openly available at <https://github.com/raminmh/CfC> under an Apache 2.0 license for the purposes of reproducing and extending the analysis.

Code availability

All code and materials used in the analysis are openly available at <https://github.com/raminmh/CfC> under an Apache 2.0 license for the purposes of reproducing and extending the analysis (<https://doi.org/10.5281/zenodo.7135472>).

References

1. Hasani, R., Lechner, M., Amini, A., Rus, D. & Grosu, R. Liquid time-constant networks. In *Proc. of AAAI Conference on Artificial Intelligence* 35(9), 7657–7666 (AAAI, 2021).
2. Chen, T. Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. K. Neural ordinary differential equations. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Bengio, S. et al.) 6571–6583 (NeurIPS, 2018).
3. Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I. & Duvenaud, D. Fjord: free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations* (2018). <https://openreview.net/forum?id=rJxgknCck7>
4. Dupont, E., Doucet, A. & Teh, Y. W. Augmented neural ODEs. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Wallach, H. et al.) 3134–3144 (NeurIPS, 2019).
5. Yang, G. et al. Pointflow: 3D point cloud generation with continuous normalizing flows. In *Proc. of the IEEE/CVF International Conference on Computer Vision* 4541–4550 (IEEE, 2019).
6. Liebenwein, L., Hasani, R., Amini, A. & Daniela, R. Sparse flows: pruning continuous-depth models. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Ranzato, M. et al.) 22628–22642 (NeurIPS, 2021).
7. Rubanova, Y., Chen, R. T. & Duvenaud, D. Latent Neural ODEs for irregularly-sampled time series. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Wallach, H. et al.) 32 (NeurIPS, 2019).

8. Gholami, A., Keutzer, K. & Biros, G. ANODE: unconditionally accurate memory-efficient gradients for neural ODEs. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* 730–736 (IJCAI, 2019).
9. Lechner, M. & Hasani, R. Learning long-term dependencies in irregularly-sampled time series. Preprint at <https://arxiv.org/abs/2006.04418> (2020).
10. Prince, P. J. & Dormand, J. R. High order embedded Runge–Kutta formulae. *J. Comput. Appl. Math.* **7**, 67–75 (1981).
11. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
12. Massaroli, S., Poli, M., Park, J., Yamashita, A. & Asma, H. Dissecting neural ODEs. In *Proc. of 33th Conference on Neural Information Processing Systems* (Eds. Larochelle, H. et al.) (NeurIPS, 2020).
13. Bai, S., Kolter, J. Z. & Koltun, V. Deep equilibrium models. *Adv. Neural Inform. Process. Syst.* **32**, 690–701 (2019).
14. Finlay, C., Jacobsen, J.-H., Nurbekyan, L. & Oberman, A. M. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *International Conference on Machine Learning* (Eds. Daumé III, H. & Singh, A.) 3154–3164 (PMLR, 2020).
15. Massaroli, S. et al. Stable Neural Flows. Preprint at <https://arxiv.org/abs/2003.08063> (2020).
16. Kidger, P., Chen, R. T. & Lyons, T. “Hey, that’s not an ODE”: Faster ODE Adjoints via Seminorms. In *Proceedings of the 38th International Conference on Machine Learning* (Eds. Meila, M. & Zhang, T.) 139 (PMLR, 2021).
17. Poli, M. et al. Hypersolvers: toward fast continuous-depth models. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Larochelle, H.) 21105–21117 (NeurIPS, 2020).
18. Schumacher, J., Haslinger, R. & Pipa, G. Statistical modeling approach for detecting generalized synchronization. *Phys. Rev. E* **85**, 056215 (2012).
19. Moran, R., Pinotsis, D. A. & Friston, K. Neural masses and fields in dynamic causal modeling. *Front. Comput. Neurosci.* **7**, 57 (2013).
20. Friston, K. J., Harrison, L. & Penny, W. Dynamic causal modelling. *Neuroimage* **19**, 1273–1302 (2003).
21. Perko, L. *Differential Equations and Dynamical Systems* (Springer-Verlag, 1991).
22. Lechner, M. et al. Neural circuit policies enabling auditable autonomy. *Nat. Mach. Intell.* **2**, 642–652 (2020).
23. Hochreiter, S. *Untersuchungen zu dynamischen neuronalen netzen*. Diploma, Technische Universität München 91 (1991).
24. Vorbach, C., Hasani, R., Amini, A., Lechner, M. & Rus, D. Causal navigation by continuous-time neural networks. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Ranzato, M. et al.) 12425–12440 (NeurIPS, 2021).
25. Hasani, R. et al. Response characterization for auditing cell dynamics in long short-term memory networks. In *Proc. of 2019 International Joint Conference on Neural Networks* 1–8 (IEEE, 2019).
26. Anguita, D., Ghio, A., Oneto, L., Parra Perez, X. & Reyes Ortiz, J. L. A public domain dataset for human activity recognition using smartphones. In *Proc. of the 21st International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* 437–442 (i6doc, 2013).
27. Todorov, E., Erez, T. & Tassa, Y. MuJoCo: a physics engine for model-based control. In *Proc. of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* 5026–5033 (IEEE, 2012).
28. Maas, A. et al. Learning word vectors for sentiment analysis. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* 142–150 (ACM, 2011).
29. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via deepnet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
30. Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
31. Wang, S., Wang, H. & Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deepnets. *Sci. Adv.* **7**, eabi8605 (2021).
32. Rezende, D. & Mohamed, S. Variational inference with normalizing flows. In *Proc. of International Conference on Machine Learning* (Eds. Bach, F. & Blei, D.) 1530–1538 (PMLR, 2015).
33. Gu, A., Goel, K. & Re, C. Efficiently modeling long sequences with structured state spaces. In *Proc. of International Conference on Learning Representations* (2022). <https://openreview.net/forum?id=uYLFoz1vIAC>
34. Hasani, R. et al. Liquid structural state-space models. Preprint at <https://arxiv.org/abs/2209.12951> (2022).
35. Grunbacher, S. et al. On the verification of neural ODEs with stochastic guarantees. *Proc. AAAI Conf. Artif. Intell.* **35**, 11525–11535 (2021).
36. Vaswani, A. et al. Attention is all you need. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Guyon, I. et al.) 5998–6008 (NeurIPS, 2017).
37. Lechner, M., Hasani, R., Grosu, R., Rus, D. & Henzinger, T. A. Adversarial training is not ready for robot learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* 4140–4147 (IEEE, 2021).
38. Brunnbauer, A. et al. Latent imagination facilitates zero-shot transfer in autonomous racing. In *2022 International Conference on Robotics and Automation (ICRA)* 7513–7520 (IEEE, 2021).
39. Hasani, R. M., Haerle, D. & Grosu, R. Efficient modeling of complex analog integrated circuits using neural networks. In *Proc. of 12th Conference on Ph.D. Research in Microelectronics and Electronics* 1–4 (IEEE, 2016).
40. Wang, G., Ledwoch, A., Hasani, R. M., Grosu, R. & Brintrup, A. A generative neural network model for the quality prediction of work in progress products. *Appl. Soft Comput.* **85**, 105683 (2019).
41. DelPreto, J. et al. Plug-and-play supervisory control using muscle and brain signals for real-time gesture and error detection. *Auton. Robots* **44**, 1303–1322 (2020).
42. Hasani, R. *Interpretable Recurrent Neural Networks in Continuous-Time Control Environments*. PhD dissertation, Technische Univ. Wien (2020).
43. Rudin, W. *Principles of Mathematical Analysis, 3rd edn.* (McGraw-Hill, 1976).
44. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
45. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. Preprint at <https://arxiv.org/abs/1412.3555> (2014).
46. Shukla, S. N. & Marlin, B. Interpolation–prediction networks for irregularly sampled time series. In *Proc. of International Conference on Learning Representations* (2018). <https://openreview.net/forum?id=r1efr3C9Ym>
47. Horn, M., Moor, M., Bock, C., Rieck, B. & Borgwardt, K. Set functions for time series. In *Proc. of International Conference on Machine Learning* (Eds. Daumé III, H. & Singh, A.) 4353–4363 (PMLR, 2020).
48. Funahashi, K.-i & Nakamura, Y. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Netw.* **6**, 801–806 (1993).
49. Mozer, M. C., Kazakov, D. & Lindsey, R. V. Discrete event, continuous time RNNs. Preprint at <https://arxiv.org/abs/1710.04110> (2017).

50. Mei, H. & Eisner, J. The neural Hawkes process: a neurally self-modulating multivariate point process. In *Proc. of 31st International Conference on Neural Information Processing Systems* (Eds. Guyon, I. et al.) 6757–6767 (Curran Associates Inc., 2017).
51. Che, Z., Purushotham, S., Cho, K., Sontag, D. & Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.* **8**, 1–12 (2018).
52. Neil, D., Pfeiffer, M. & Liu, S.-C. Phased LSTM: accelerating recurrent network training for long or event-based sequences. In *Proc. of 30th International Conference on Neural Information Processing Systems* (Eds. Lee, D. D. et al.) 3889–3897 (Curran Associates Inc., 2016).
53. Schuster, M. & Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **45**, 2673–2681 (1997).
54. Voelker, A. R., Kajić, I. & Eliasmith, C. Legendre memory units: continuous-time representation in recurrent neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Eds. Wallach, H. et al.) 15570–15579 (ACM, 2019).
55. Gu, A., Dao, T., Ermon, S., Rudra, A. & Ré, C. Hippo: recurrent memory with optimal polynomial projections. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Larochelle, H. et al.) 1474–1487 (NeurIPS, 2020).
56. Lezcano-Casado, M. & Martínez-Rubio, D. Cheap orthogonal constraints in neural networks: a simple parametrization of the orthogonal and unitary group. In *Proc. of International Conference on Machine Learning* (Eds. Chaudhuri, K. & Salakhutdinov, R.) 3794–3803 (PMLR, 2019).
57. Rusch, T. K. & Mishra, S. Coupled oscillatory recurrent neural network (coRNN): an accurate and (gradient) stable architecture for learning long time dependencies. In *Proc. of International Conference on Learning Representations* (2021). <https://openreview.net/forum?id=F3s69XzWOia>
58. Erichson, N. B., Azencot, O., Queiruga, A., Hodgkinson, L. & Mahoney, M. W. Lipschitz recurrent neural networks. In *Proc. of International Conference on Learning Representations* (2021). <https://openreview.net/forum?id=N7PBXqOUJZ>
59. Brockman, G. et al. OpenAI gym. Preprint at <https://arxiv.org/abs/1606.01540> (2016).
60. Lechner, M., Hasani, R., Zimmer, M., Henzinger, T. A. & Grosu, R. Designing worm-inspired neural networks for interpretable robotic control. In *Proc. of International Conference on Robotics and Automation* 87–94 (IEEE, 2019).
61. Tylkin, P. et al. Interpretable autonomous flight via compact visualizable neural circuit policies. *IEEE Robot. Autom. Lett.* **7**, 3265–3272 (2022).
62. Amini, A. et al. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)* 2419–2426 (IEEE, 2022).
63. Amini, A. et al. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robot. Autom. Lett.* **5**, 1143–1150 (2020).
64. Levine, S. & Koltun, V. Guided policy search. In *Proc. of International Conference on Machine Learning* (Eds. Dasgupta, S. & McAllester, D.) 1–9 (PMLR, 2013).
65. Bojarski, M. et al. VisualBackProp: efficient visualization of CNNs for autonomous driving. In *Proc. of IEEE International Conference on Robotics and Automation* 1–8 (IEEE, 2018).
66. Zhang, H., Wang, Z. & Liu, D. A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **25**, 1229–1262 (2014).
67. Weinan, E. A proposal on machine learning via dynamical systems. *Commun. Math. Stat.* **5**, 1–11 (2017).
68. Lu, Z., Pu, H., Wang, F., Hu, Z. & Wang, L. The expressive power of neural networks: a view from the width. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Guyon, I. et al.) 30 (Curran Associates, Inc 2017).
69. Li, Q., Chen, L., Tai, C. et al. Maximum principle based algorithms for deep learning. *J. Mach. Learn. Res.* **18**, 5998–6026 (2018).
70. Cohen, M. A. & Grossberg, S. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Trans. Syst. Man Cybern.* **5**, 815–826 (1983).
71. Mathieu, E. & Nickel, M. Riemannian continuous normalizing flows. In *Proc. of Advances in Neural Information Processing Systems* Vol. 33 (eds Larochelle et al.) 2503–2515 (Curran Associates, Inc., 2020).
72. Hodgkinson, L., van der Heide, C., Roosta, F. & Mahoney, M. W. Stochastic normalizing flows. In *Proc. of Advances in Neural Information Processing Systems* (Eds. Larochelle, H. et al.) 5933–5944 (NeurIPS, 2020).
73. Haber, E., Lensink, K., Treister, E. & Ruthotto, L. IMEXnet a forward stable deep neural network. In *Proc. of International Conference on Machine Learning* (Eds. Chaudhuri, K. & Salakhutdinov, R.) 2525–2534 (PMLR, 2019).
74. Chang, B., Chen, M., Haber, E. & Chi, E. H. AntisymmetricRNN: a dynamical system view on recurrent neural networks. In *International Conference on Learning Representations* (2018). <https://openreview.net/forum?id=ryxepo0cFX>
75. Lechner, M., Hasani, R., Rus, D. & Grosu, R. Gershgorin loss stabilizes the recurrent neural network compartment of an end-to-end robot learning scheme. In *Proc. of IEEE International Conference on Robotics and Automation* 5446–5452 (IEEE, 2020).
76. Gleeson, P., Lung, D., Grosu, R., Hasani, R. & Larson, S. D. c302: a multiscale framework for modelling the nervous system of *Caenorhabditis elegans*. *Philos. Trans. R. Soc. B* **373**, 20170379 (2018).
77. Li, X., Wong, T.-K. L., Chen, R. T. & Duvenaud, D. Scalable gradients for stochastic differential equations. In *Proc. of International Conference on Artificial Intelligence and Statistics* 3870–3882 (PMLR, 2020).
78. Shukla, S. N. & Marlin, B. M. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations* (2020). https://openreview.net/forum?id=4c0J6lwQ4_
79. Xiong, Y. et al. Nyströmformer: a Nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 35, No. 16, pp. 14138–14148 (AAAI, 2021).

Acknowledgements

This research was supported in part by the AI2050 program at Schmidt Futures (grant G-22-63172), the Boeing Company, and 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 Office of Naval Research grant N00014-18-1-2830. M.T. is supported by the Poul Due Jensen Foundation, grant 883901. M.L. was supported in part by the Austrian Science Fund under grant Z211-N23 (Wittgenstein Award). A.A. was supported by the National Science Foundation Graduate Research Fellowship Program. We thank T.-H. Wang, P. Kao, M. Chahine, W. Xiao, X. Li, L. Yin and Y. Ben for useful suggestions and for testing of CfC models to confirm the results across other domains.

Author contributions

R.H. and M.L. conceptualized, proved theory, designed, performed research and analysed data. A.A. contributed to designing research, data curation, research implementation, new analytical tools and analysed data. L.L. and A.R. contributed to the refinement of the theory and research implementation. M.T. and G.T. proved theory and analysed correctness. D.R. helped with the design of the research, and guided and supervised the work. All authors wrote the paper.

Competing interests

The authors declare no competing interest.

Additional information

Extended data is available for this paper at <https://doi.org/10.1038/s42256-022-00556-7>.

Correspondence and requests for materials should be addressed to Ramin Hasani.

Peer review information *Nature Machine Intelligence* thanks Karl Friston and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

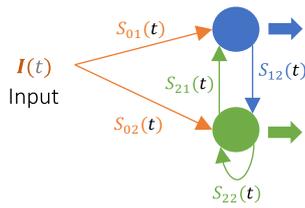
Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2022, corrected publication 2022

a. LTC network with 2 neurons



b. LTC differential equations

$$\frac{dx_1(t)}{dt} = -\frac{x_1(t)}{\tau_1} + f_{01}(I(t)) (A_{01} - x_1(t)) + f_{21}(x_2(t)) (A_{21} - x_1(t))$$

$$\frac{dx_2(t)}{dt} = -\frac{x_2(t)}{\tau_2} + f_{02}(I(t)) (A_{02} - x_2(t)) + f_{12}(x_1(t)) (A_{12} - x_2(t)) + f_{22}(x_2(t)) (A_{22} - x_2(t))$$

c. Approximate closed-form solution of LTCs

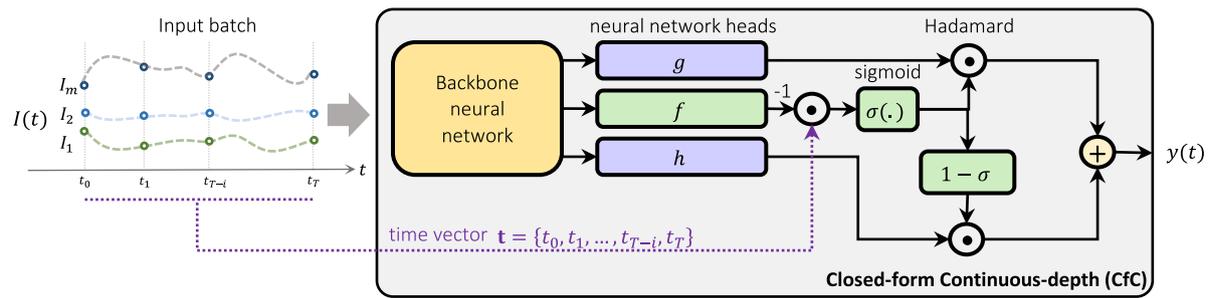
$$x_1(t) = (x_1(0) - A_{01}) e^{-\left[\frac{1}{\tau_1} + f_{01}(I(t))\right]t} f_{01}(-I(t)) + A_{01} + (x_1(0) - A_{21}) e^{-\left[\frac{1}{\tau_1} + f_{21}(x_2(t))\right]t} f_{21}(-x_2(t)) + A_{21}$$

$$x_2(t) = (x_2(0) - A_{02}) e^{-\left[\frac{1}{\tau_2} + f_{02}(I(t))\right]t} f_{02}(-I(t)) + A_{02} + (x_2(0) - A_{12}) e^{-\left[\frac{1}{\tau_2} + f_{12}(x_1(t))\right]t} f_{12}(-x_2(t)) + A_{12} + (x_2(0) - A_{22}) e^{-\left[\frac{1}{\tau_2} + f_{22}(x_2(t))\right]t} f_{22}(-x_2(t)) + A_{22}$$

Legend

- $x_i(t)$ potential of neuron i
- $S_{ij}(t)$ synapse between node i and j
- τ_i time-constant of neuron i
- A_{ij} synaptic reversal potential for nodes i and j
- f_{ij} nonlinearity of a synapse between i and j
- t time

Extended Data Fig. 1 | Instantiation of LTCs in ODE and closed-form representations. a) A sample LTC network with two nodes and five synapses. b) the ODE representation of this two-neuron system. c) the approximate closed-form representation of the network.



Extended Data Fig. 2 | Closed-form Continuous-depth neural architecture. A backbone neural network layer delivers the input signals into three head networks g , f and h . f acts as a *liquid* time-constant for the sigmoidal time-gates of the network. g and h construct the nonlinearities of the overall CfC network.

Parameter	Human Activity				Walker2D			
	Cf-S	CfC	CfC-noGate	CfC-mmRNN	Cf-S	CfC	CfC-noGate	CfC-mmRNN
optimizer	AdamW	AdamW	AdamW	AdamW	Adam	Adam	Adam	Adam
batch_size	64	64	64	64	128	256	128	128
Hidden size	64	448	64	128	256	64	256	128
epochs	100	100	100	100	50	50	50	50
base_lr	0.004	0.002	0.005	0.0005	0.006	0.02	0.008	0.005
decay_lr	0.97	0.97	0.97	0.99	0.95	0.95	0.95	0.95
backbone_activation	GELU	SiLU	SiLU	GELU	SiLU	SiLU	LeCun Tanh	LeCun Tanh
backbone_dr	0.4	0.0	0.2	0.5	0.0	0.1	0.1	0.2
forget_bias	-	-	-	1.0	5.0	1.6	2.8	2.1
backbone_units	256	64	192	128	192	256	128	128
backbone_layers	3	1	2	2	1	1	1	2
weight_decay	3e-05	1e-04	2-e04	4e-05	1e-06	1e-06	3e-05	6e-06
τ_{data}	0.1	10	0.5	10			not applicable	
init	0.67	0.84	0.78	1.35			not applicable	

Extended Data Fig. 3 | Hyperparameters for Human activity and Walker. List of hyperparameters used to obtain results in Human activity and Walker2D Experiments.

Parameter	Event-based sMNIST				Bit-Stream XOR			
	Cf-S	CfC	CfC-noGate	CfC-mmRNN	Cf-S	CfC	CfC-noGate	CfC-mmRNN
optimizer	AdamW	AdamW	AdamW	AdamW	Adam	RMSProp	RMSprop	RMSprop
batch_size	64	64	64	64	256	128	128	128
Hidden size	64	64	64	64	64	192	128	64
epochs	200	200	200	200	200	200	200	200
base_lr	0.0005	0.0005	0.0005	0.0005	0.005	0.05	0.005	0.005
decay_lr	lr decay not used				0.9	0.7	0.95	0.95
backbone_activation	GELU	GELU	GELU	GELU	SiLU	ReLU	SiLU	ReLU
backbone_dr	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0
forget_bias	-	-	-	3.0	1.2	1.2	4.7	0.6
backbone_units	128	128	128	128	64	128	192	128
backbone_layers	1	1	1	1	1	1	1	1
weight_decay	0	0	0	0	3e-05	3e-06	5e-06	2e-06
clipnorm	clipnorm not used				5	1	10	10

Extended Data Fig. 4 | Hyperparameters for ET-sMNIST and Bit-stream XOR. List of hyperparameters used to obtain results in Event-based MNIST and Bit-stream XOR Experiments.

Model	Equidistant encoding	Event-based (irregular) encoding	Time Per epoch (min)	ODE-based?
†Augmented LSTM ⁴⁴	100.00% ± 0.00	89.71% ± 3.48	0.62	No
†CT-GRU ⁴⁹	100.00% ± 0.00	61.36% ± 4.87	0.80	No
†RNN Decay ⁷	60.28% ± 19.87	75.53% ± 5.28	0.90	No
†Bi-directional RNN ⁵³	100.00% ± 0.00	90.17% ± 0.69	1.82	No
†GRU-D ⁵¹	100.00% ± 0.00	97.90% ± 1.71	0.58	No
†PhasedLSTM ⁵²	50.99% ± 0.76	80.29% ± 0.99	1.22	No
†CT-LSTM ⁵⁰	97.73% ± 0.08	95.09% ± 0.30	0.86	No
coRNN ⁵⁷	100.00% ± 0.00	52.89% ± 1.25	0.57	No
Lipschitz RNN ⁵⁸	100.00% ± 0.00	52.84% ± 3.25	0.63	No
†ODE-RNN ⁷	50.47% ± 0.06	51.21% ± 0.37	4.11	Yes
†CT-RNN ⁴⁸	50.42% ± 0.12	50.79% ± 0.34	4.83	Yes
†GRU-ODE ⁷	50.41% ± 0.40	52.52% ± 0.35	1.55	Yes
†ODE-LSTM ⁹	100.00% ± 0.00	98.89% ± 0.26	1.18	Yes
LTC ¹	100.00% ± 0.00	49.11% ± 0.00	2.67	Yes
Cf-S (ours)	100.00% ± 0.00	85.42% ± 2.84	0.36	No
CfC-noGate (ours)	100.00% ± 0.00	96.29% ± 1.61	0.78	No
CfC (ours)	100.00% ± 0.00	99.42% ± 0.42	0.75	No
CfC-mmRNN (ours)	100.00% ± 0.00	99.72% ± 0.08	1.26	No

Extended Data Fig. 5 | Bit-stream XOR sequence classification. The performance values (accuracy %) for all baseline models are reproduced from⁹. Numbers present mean ± standard deviations, (n=5). **Note:** The performance of models marked by † are reported from⁹. Bold declares highest accuracy and best time per epoch (min).

Model	AUC Score (%)	time per epoch (min)
†RNN-Impute ⁷	0.764 ± 0.016	0.5
†RNN-delta-t ⁷	0.787 ± 0.014	0.5
†RNN-Decay ⁷	0.807 ± 0.003	0.7
†GRU-D ⁵¹	0.818 ± 0.008	0.7
†Phased-LSTM ⁵²	0.836 ± 0.003	0.3
*IP-Nets ⁴⁶	0.819 ± 0.006	1.3
*SeFT ⁴⁷	0.795 ± 0.015	0.5
†RNN-VAE ⁷	0.515 ± 0.040	2.0
†ODE-RNN ⁷	0.833 ± 0.009	16.5
†Latent-ODE-RNN ⁷	0.781 ± 0.018	6.7
†Latent-ODE-ODE ⁷	0.829 ± 0.004	22.0
LTC ¹	0.6477 ± 0.010	0.5
Cf-S (ours)	0.643 ± 0.018	0.1
CfC-noGate (ours)	0.840 ± 0.003	0.1
CfC (ours)	0.839 ± 0.002	0.1
CfC-mmRNN (ours)	0.834 ± 0.006	0.2

Extended Data Fig. 6 | PhysioNet. AUC stands for area under curve. Numbers present mean ± standard deviations, (n=5). **Note:** The performance of the models marked by † are reported from⁷ and the ones with * from⁷⁸. Bold declares highest AUC score and best time per epoch (min).

Parameter	Physionet				IMDB			
	Cf-S	CfC	CfC-noGate	CfC-mmRNN	Cf-S	CfC	CfC-noGate	CfC-mmRNN
epochs	116	57	58	65	27	47	37	20
class_weight	18.25	11.69	7.73	5.91	not applicable			
clipnorm	0	0	0	0	1	10	5	10
Hidden size	64	256	64	64	320	192	224	64
base_lr	0.003	0.002	0.003	0.001	0.0005	0.0005	0.0005	0.0005
decay_lr	0.72	0.9	0.73	0.9	0.8	0.7	0.8	0.8
backbone_activation	Tanh	SiLU	ReLU	LeCun Tanh	Relu	SiLU	SiLU	LeCun Tanh
backbone_units	64	64	192	64	64	64	128	64
backbone_dr	0.1	0.2	0.0	0.3	0.0	0.0	0.1	0.0
backbone_layers	3	2	2	2	1	2	1	1
weight_decay	5e-05	4e-06	5e-05	4e-06	0.00048	3.6e-05	2.7e-05	0.00029
optimizer	AdamW	AdamW	AdamW	AdamW	Adam	RMSProp	RMSprop	RMSprop
init	0.53	0.50	0.55	0.6	not applicable			
batch_size	128	128	128	128	128	128	128	128
embed_dim	not applicable				64	192	192	32
embed_dr	not applicable				0.0	0.0	0.2	0.3

Extended Data Fig. 7 | Hyperparameters for Physionet and IMDB. List of hyperparameters used to obtain results in Physionet and IMDB sentiment classification experiments.

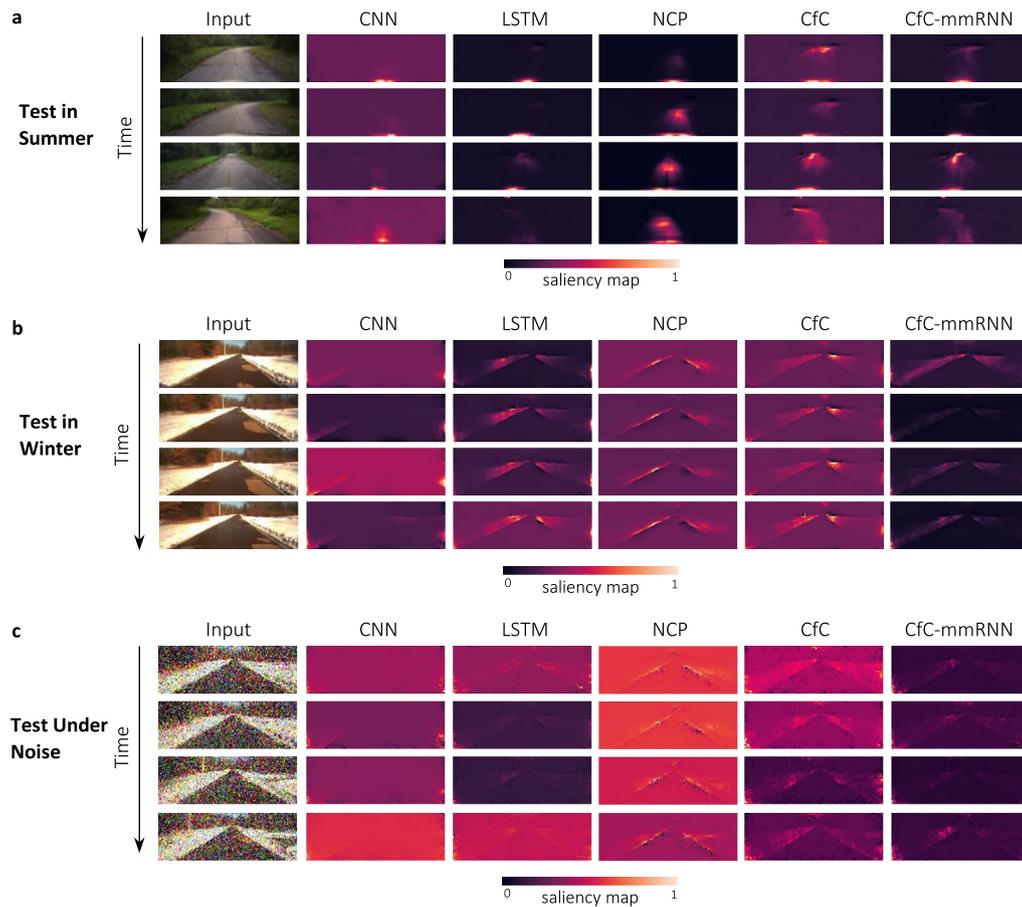
Model	Test accuracy (%)
†HiPPO-LagT ⁵⁵	88.0 ± 0.2
†HiPPO-LegS ⁵⁵	88.0 ± 0.2
†LMU ⁵⁴	87.7 ± 0.1
†LSTM ⁴⁴	87.3 ± 0.4
†GRU ⁴⁵	86.2 ± n/a
*ReLU GRU ²	84.8 ± n/a
*Skip LSTM ²	86.6 ± n/a
†expRNN ⁵⁶	84.3 ± 0.3
†Vanilla RNN ²	67.4 ± 7.7
*coRNN ⁵⁷	86.7 ± 0.3
LTC ¹	61.8 ± 6.1
Cf-S (ours)	81.7 ± 0.5
CfC-noGate (ours)	87.5 ± 0.1
CfC (ours)	85.9 ± 0.9
CfC-mmRNN (ours)	88.3 ± 0.1

Extended Data Fig. 8 | Results on the IMDB datasets. The experiment is performed without any pretraining or pretrained word-embeddings. Thus, we excluded advanced attention-based models^{78,79} such as Transformers³⁶ and RNN structures that use pretraining. Numbers present mean ± standard deviations,

(n=5). **Note:** The performance of the models marked by † are reported from⁵⁵, and * are reported from⁵⁷. The n/a standard deviation denotes that the original report of these experiments did not provide the statistics of their analysis. Bold declares highest accuracy and best time per epoch (min).

Modes	Total Parameter Count (CNN head + RNN)	RNN Parameter Count
CNN	2,124,901	-
LSTM	259,733	33089
NCP	233,139	6495
Cf-S	227,728	1084
CfC	230,828	4184
CfC-NoGate	230,828	4184
CfC-mmRNN	235,052	8408

Extended Data Fig. 9 | Lane-keeping models' parameter count. CfC and NCP networks perform lane-keeping in unseen scenarios with a compact representation.



Extended Data Fig. 10 | Attention Profile of networks. Trained networks receive unseen inputs (first column in each tab) and generate acceleration and steering commands. We use the Visual-Backprop algorithm⁶⁵ to compute the saliency maps of the convolutional part of each network. a) results for networks

tested on data collected in summer. b) results for networks tested on data collected in winter. c) results for inputs corrupted by a zero-mean Gaussian noise with variance, $\sigma^2 = 0.35$.