Learning Neural Event Functions for ODEs

Ricky T. Q. Chen[1], Brandon Amos[2], Maximilian Nickel[2]

[1]University of Toronto; Vector Institute. [2]Facebook AI Research.

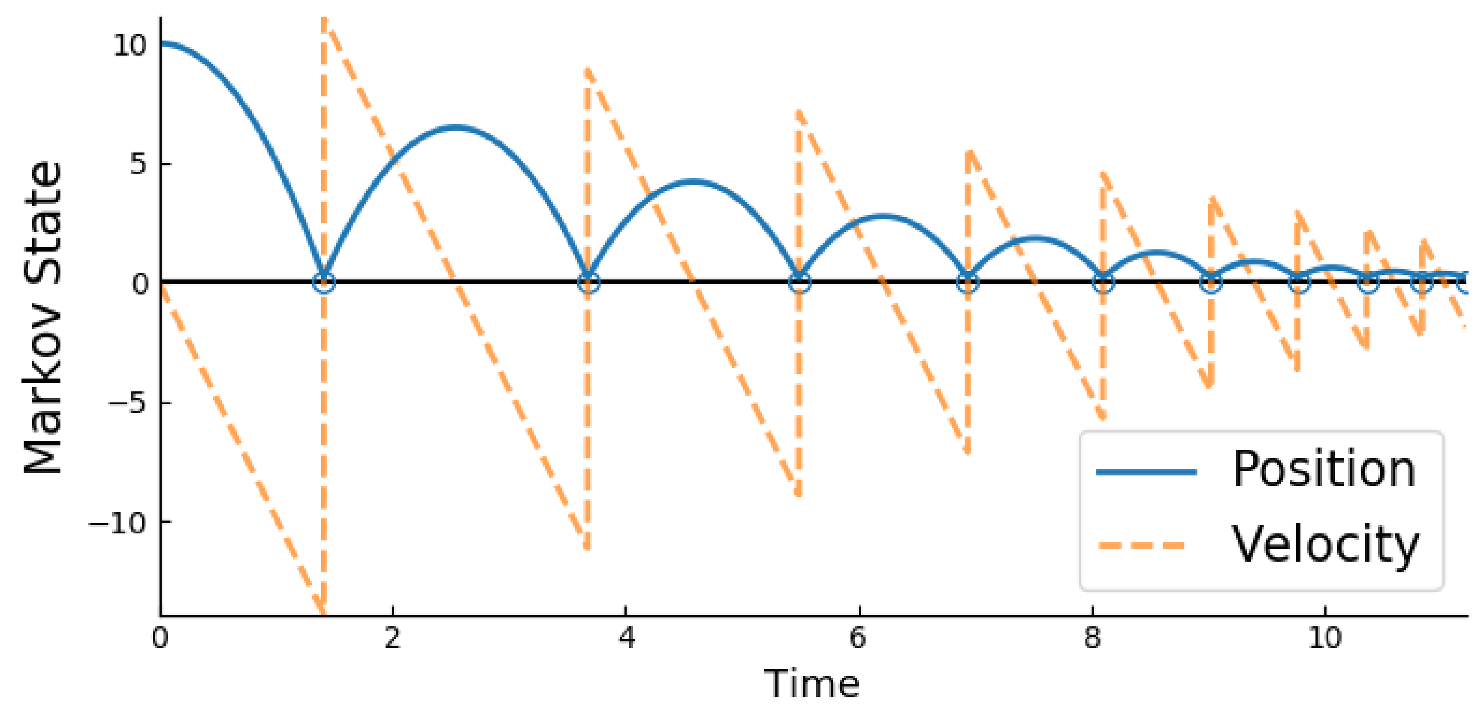
UNIVERSITY OF TORONTO

FACEBOOK AI

Extending Neural ODEs with Discontinuities

Solutions of ODEs are smooth trajectories. ODEs lack a mechanism for modeling **instantaneous interventions**.

Example (simulation of a bouncing ball):



Velocity is discontinuous. Position has discontinuous derivative. A Neural ODE fails to model this.

We implement *differentiable event handling* to build models that learn **when** and **how** to apply instantaneous interventions.

Differentiable Event Handling

Define event function $g(z, t)$. Event defined as when trajectories cross the root, *i.e.* t^* s.t. $g(z(t^*), t^*) = 0$.

$$t^*, z(t^*) = \texttt{ODESolveEvent}(z_0, f, g, t_0, \theta, \phi) \qquad (1)$$

Solves $\frac{dz}{dt} = f(z, t, \theta)$ with initial state (z_0, t_0) until event t^*.

Gradient of `ODESolveEvent` can be derived by combining *implicit function theorem* and *adjoint method* (see paper for details).

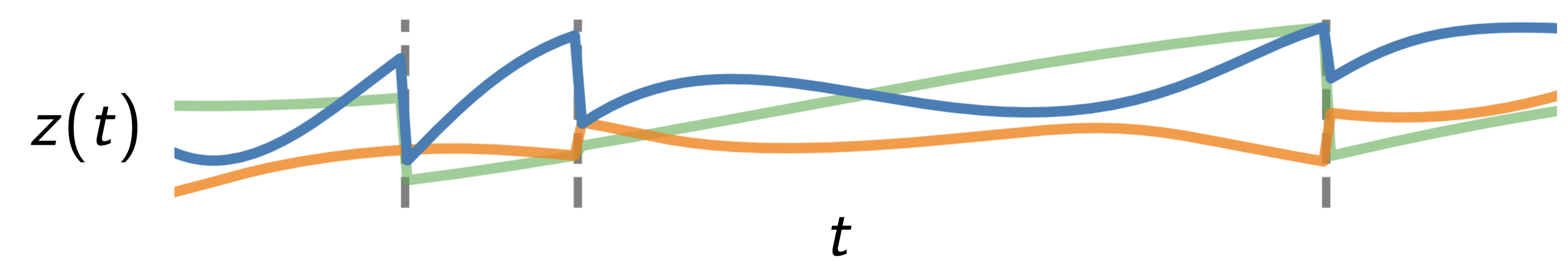
Efficient $O(1)$-memory gradient implemented in github.com/rtqichen/torchdiffeq.

Neural Event ODE

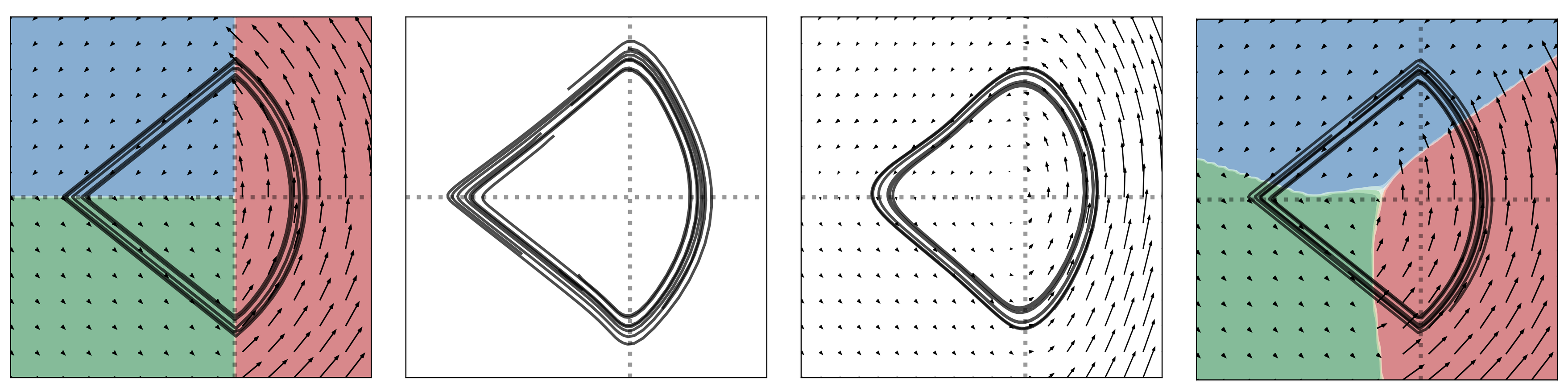
Repeat: (i) solve until event, (ii) update state instantaneously.

while $t_i < T$ do
  (i) $t_{i+1}, z'_{i+1} = \texttt{ODESolveEvent}(z_i, f, g, t_i)$
  (ii) $z_{i+1} = h(t_{i+1}, z'_{t+1})$
end while

Capable of modeling **variable** number of discontinuities.



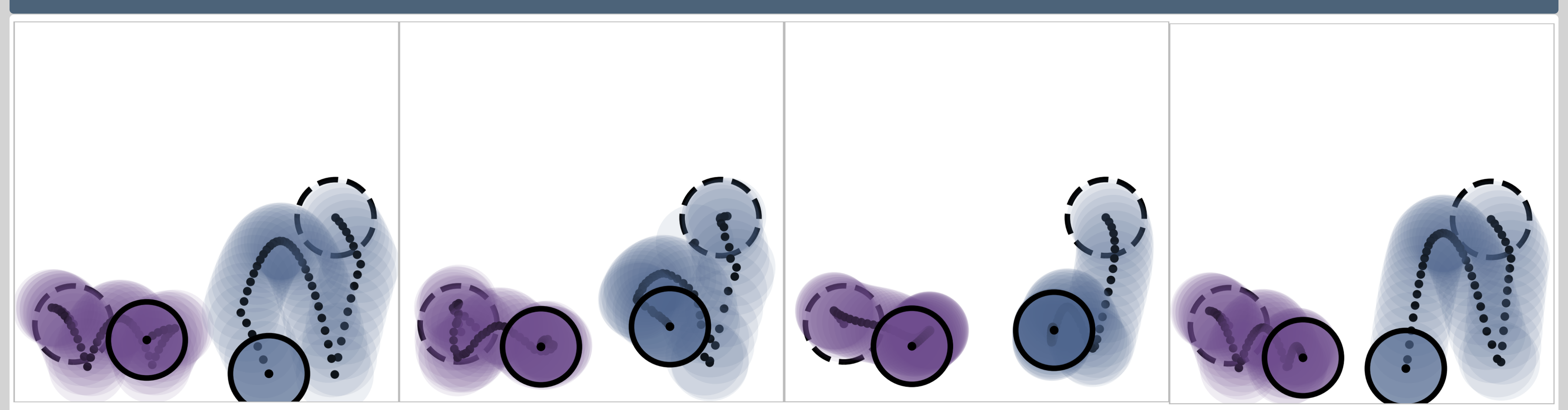
Switching Linear Dynamical System



(a) Ground truth (b) RNN (LSTM) (c) Neural ODE (d) Neural Event ODE

Figure: A Neural Event ODE **jumps** between a set of linear dynamics using *only gradients*. (Colors denote which linear system, not event boundaries.)

Modeling Physics with Collision



(a) Ground truth (b) RNN (LSTM) (c) Neural ODE (d) Neural Event ODE

Moreover, Neural ODE baseline approximates collisions with stiff trajectories, requiring $10\times$ more compute to solve.

Threshold-based Event Functions

Threshold-based event occurs when an integral over a positive function reaches a predetermined threshold.

$$t^* \text{ such that } s = \int_{t_0}^{t^*} \lambda(t)\, dt \qquad (2)$$

where $\lambda(t) > 0$. Implemented by tracking $\Lambda(t) \triangleq \int_{t_0}^{t} \lambda(s) ds$ as part of the ODE state and using $g(t, z(t)) = s - \Lambda(t)$.

Allows **exact gradients** for integrate-and-fire spiking neural nets, inverse sampling, temporal point processes (TPP), etc.

Differentiable Sampling for TPPs

Sampling from a temporal point process (repeat):
  (i) sample $s_i \sim \text{Exp}(1)$
  (ii) solve for t_i such that $s_i = \int_{t_{i-1}}^{t_i} \lambda(t) dt$

Differentiable event handling through step (ii) provides the **reparameterization gradient** for temporal point processes.

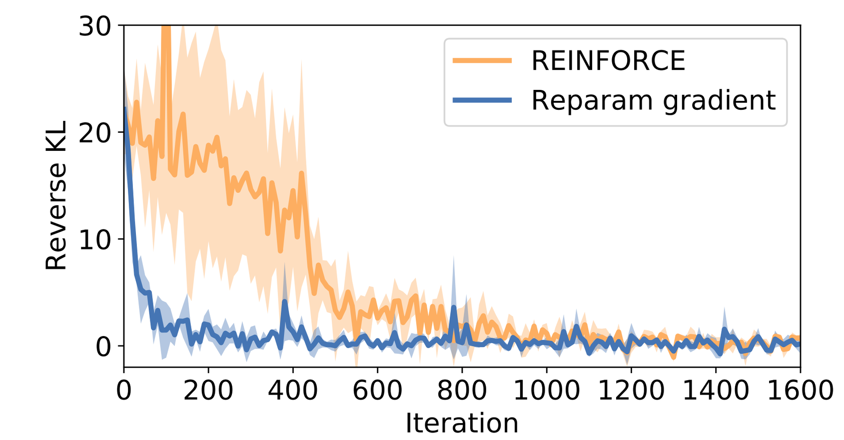


Figure: Reverse-KL Training

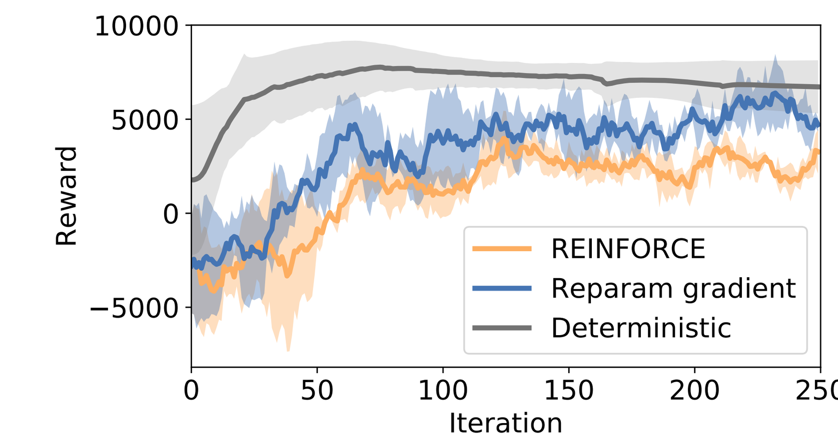


Figure: Discrete-valued Control

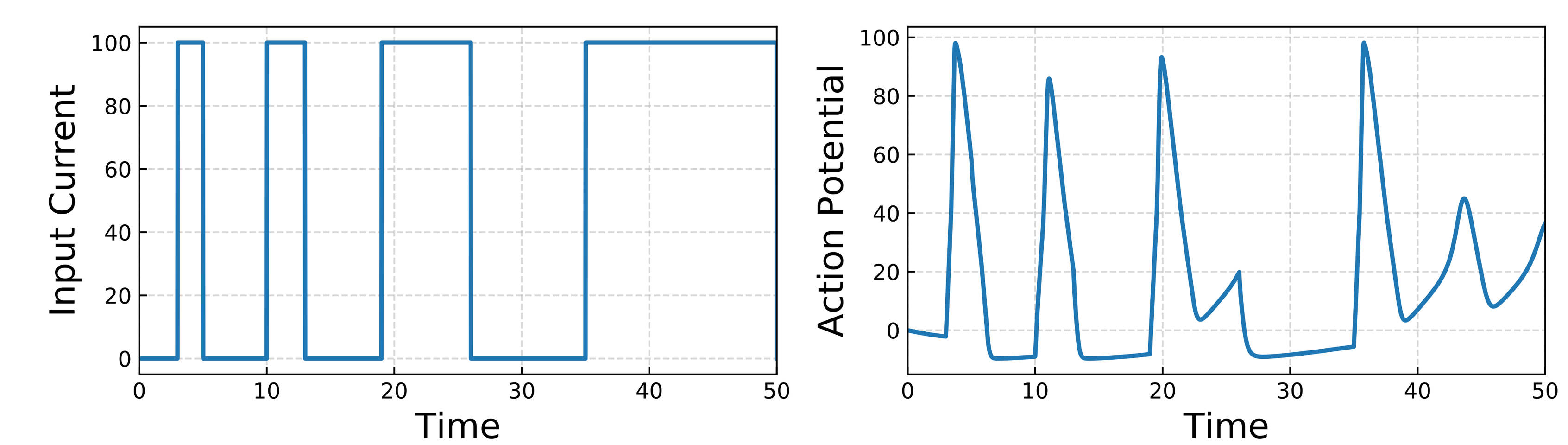


Figure: Visualization of a discrete-valued control (*left*) in a continuous-time environment (*right*), using the Hodgkin-Huxley model of neuronal dynamics.