Second-Order Neural ODE Optimizer

Guan-Horng Liu, Tianrong Chen, Evangelos A. Theodorou

Georgia Institute of Technology

{ghliu, tianrong.chen, evangelos.theodorou}@gatech.edu

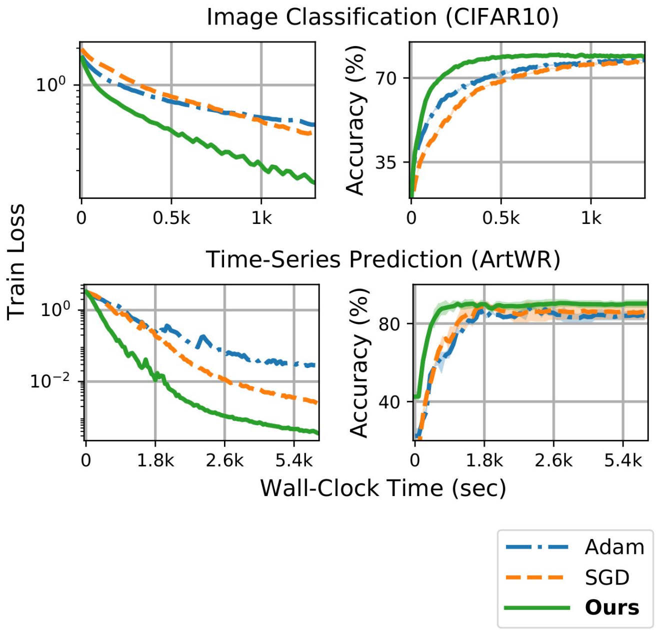
NeurIPS 2021 (Spotlight)

Second-Order Neural ODE Optimizer

A new optimizer for deep continuous-time models (*e.g.*, Neural ODEs) that enjoys

- Strong empirical results
  - superior convergence & test-time performance
  - hyper-parameter robustness
  - architecture optimization

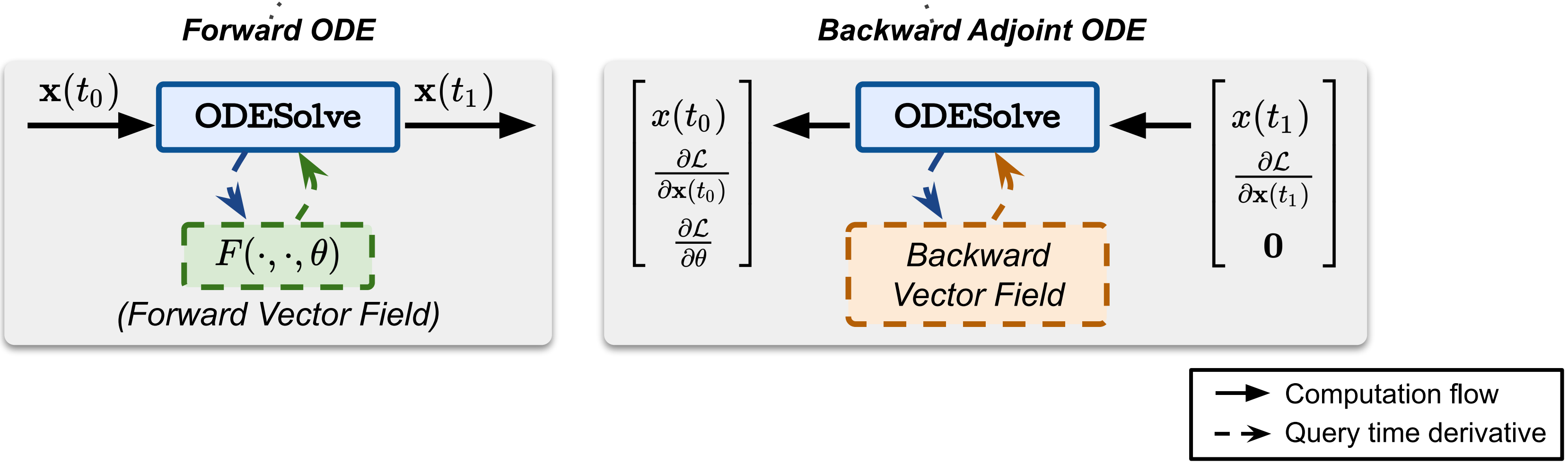
- Solid theoretical analysis
  - continuous-time optimal control theory
  - generalization of first-order adjoint method to higher-order at the *same* ***O(1)*** memory (in depth)



Training Process of Neural ODEs

Adjoint-based optimization

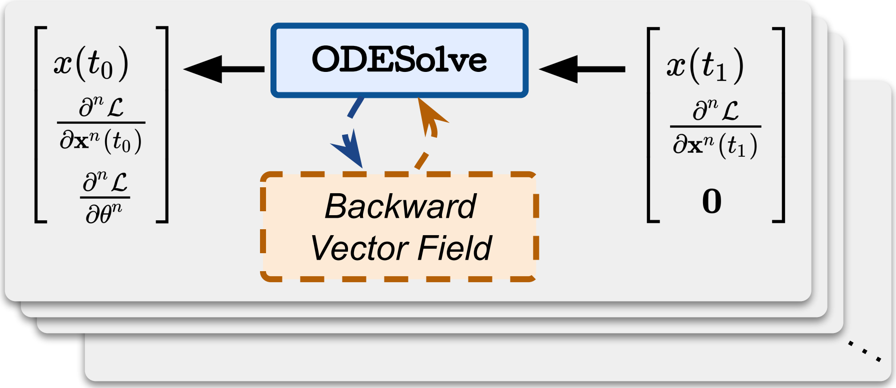
$$\min_{\theta} \mathcal{L}(\mathbf{x}(t_1)), \quad \text{s.t.} \quad \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \underbrace{F(t, \mathbf{x}(t), \theta)}_{\textit{Neural ODE}}, \quad \mathbf{x}(t_0) = \boldsymbol{x}_{t_0}$$



Training Process of Neural ODEs

Application to higher-order optimization (prior attempts)

Backward Recursive Adjoint ODE



$$\frac{\partial^n \mathcal{L}}{\partial \theta^n} = \texttt{grad}\left(\frac{\partial^{n-1} \mathcal{L}}{\partial \theta^{n-1}}, \theta\right)$$

☹️ linear runtime dependency

☹️ accumulated integration errors

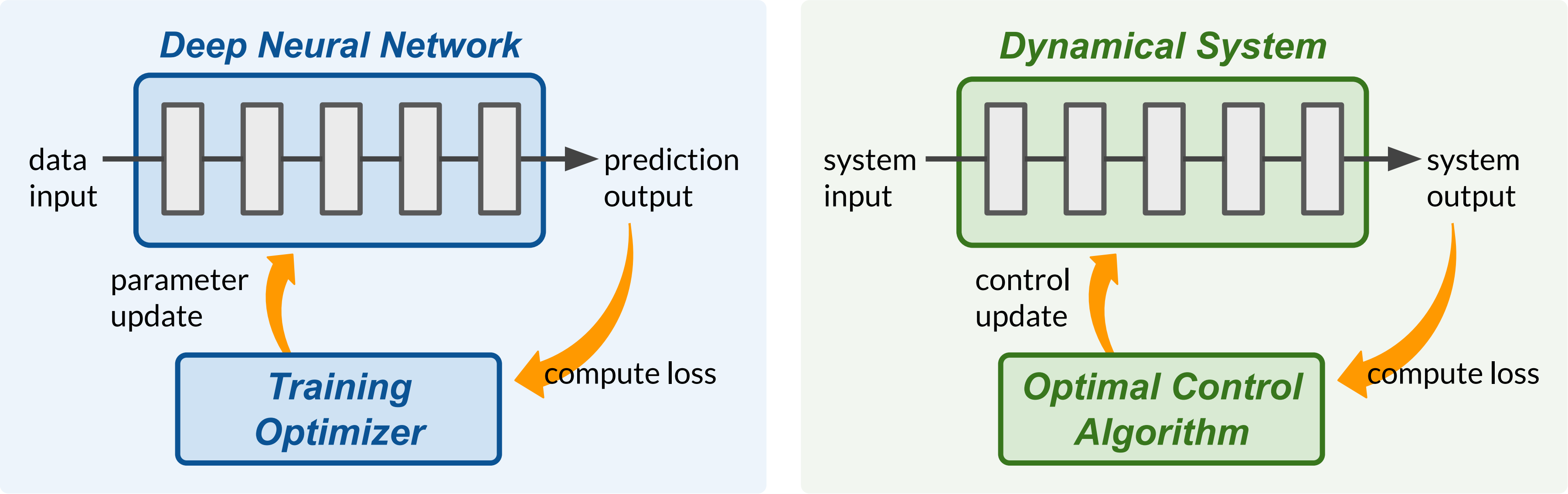
This work (our contribution)

Higher-order computational framework from an optimization viewpoint.

→ Computation flow

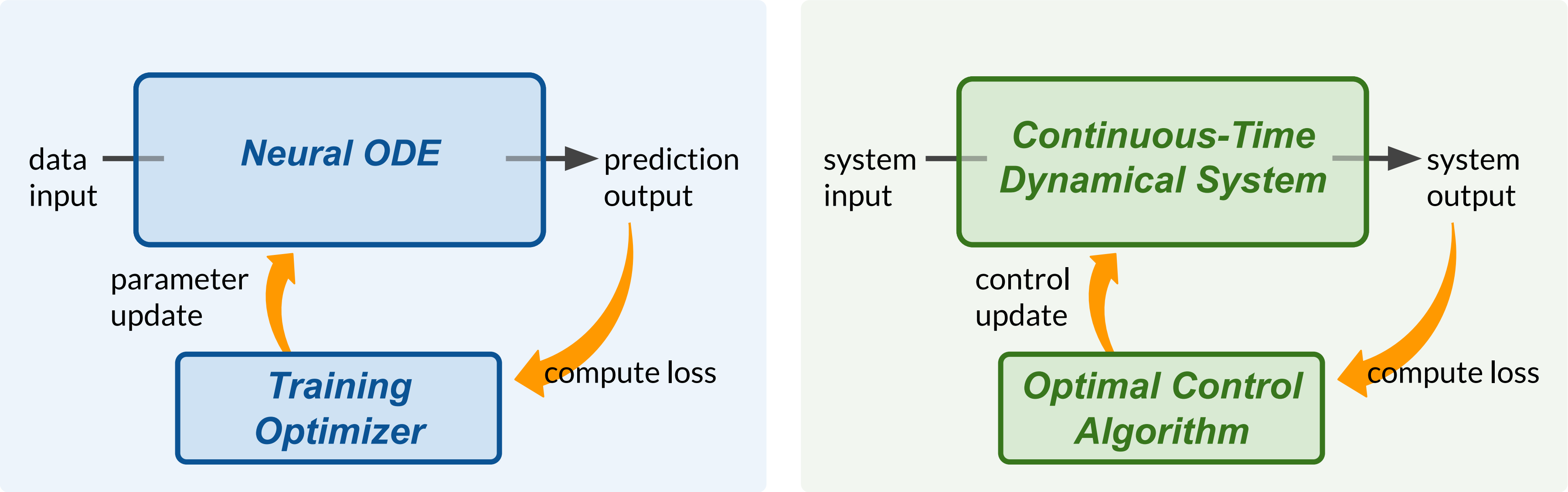
- → Query time derivative

Optimal Control Perspective (OC)



- Treat the *propagation of each layer* as a distinct *time step of a nonlinear dynamical system*.
- Interpret *layer parameter* as the *time-varying control* (*Weinan et al., 2018*; *Liu & Theodorou, 2019*).
- New optimization theory and OC-inspired training methods (*Liu et al., 2021a,b*).

Optimal Control Perspective (OC)



- *Neural ODEs*, by construction, aim to represent *continuous-time dynamical systems*.
- *Backward adjoint ODE* originates from optimality conditions in *Optimal Control* (*Pontryagin et al., 1962*).

Training Neural ODE by Solving OCP

Original Training Process

$$\min_{\theta} \mathcal{L}(\mathbf{x}(t_1)),$$

$$\text{s.t.} \quad \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \underbrace{F(t, \mathbf{x}(t), \theta)}_{\text{Neural ODE}}, \quad \mathbf{x}(t_0) = \boldsymbol{x}_{t_0},$$

$$(\mathcal{L}, 0) := (\Phi, \ell)$$

Optimal Control Programming (OCP)

$$\min_{\theta} \left[\Phi(\mathbf{x}_{t_1}) + \int_{t_0}^{t_1} \ell(t, \mathbf{x}_t, \mathbf{u}_t) \mathrm{d}t \right],$$

$$\text{s.t.} \begin{cases} \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \underline{\underline{F(t, \mathbf{x}(t), \mathbf{u}(t))}}, & \mathbf{x}(t_0) = \boldsymbol{x}_{t_0} \\ \frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t} = \mathbf{0}, & \mathbf{u}(t_0) = \theta \end{cases}$$

ODE with time-invariant control

Define accumulated loss:

$$Q(t, \mathbf{x}_t, \mathbf{u}_t) := \Phi(\mathbf{x}_{t_1}) + \int_t^{t_1} \ell(\tau, \mathbf{x}_\tau, \mathbf{u}_\tau) \mathrm{d}\tau$$

Adjoint-based derivatives

$$\frac{\partial \mathcal{L}}{\partial \theta}$$

$$\frac{\partial^2 \mathcal{L}}{\partial \theta \partial \theta}$$

$$(\mathcal{L}, \theta) := (Q(t_0), \mathbf{u}_{t_0})$$

OCP-based derivatives

$$\frac{\partial Q(t_0, \mathbf{x}_{t_0}, \mathbf{u}_{t_0})}{\partial \mathbf{u}_{t_0}} \equiv Q_{\boldsymbol{u}}(t_0)$$

$$\frac{\partial^2 Q(t_0, \mathbf{x}_{t_0}, \mathbf{u}_{t_0})}{\partial \mathbf{u}_{t_0} \partial \mathbf{u}_{t_0}} \equiv Q_{\boldsymbol{uu}}(t_0)$$

Generalization of Adjoint Process

Our goal is to solve derivatives of $Q(t, \mathbf{x}_t, \mathbf{u}_t)$ w.r.t. the control $\mathbf{u}$ at t_0 :

Theorem 1 (Second-Order Differential Programming)

The derivatives of Q expanded along a solution path $(\boldsymbol{x}_t, \boldsymbol{u}_t)$, which solves the forward ODE, obey the following set of couple backward ODEs:

$$\left.\begin{aligned} -\frac{\mathrm{d}Q_{\boldsymbol{x}}}{\mathrm{d}t} &= \ell_{\boldsymbol{x}} + F_{\boldsymbol{x}}^{\mathsf{T}} Q_{\boldsymbol{x}} & \quad -\frac{\mathrm{d}Q_{\boldsymbol{u}}}{\mathrm{d}t} &= \ell_{\boldsymbol{u}} + F_{\boldsymbol{u}}^{\mathsf{T}} Q_{\boldsymbol{x}} \end{aligned}\right\} \text{original Adjoint}$$

$$\left.\begin{aligned} -\frac{\mathrm{d}Q_{\boldsymbol{xx}}}{\mathrm{d}t} &= \ell_{\boldsymbol{xx}} + F_{\boldsymbol{x}}^{\mathsf{T}} Q_{\boldsymbol{xx}} + Q_{\boldsymbol{xx}} F_{\boldsymbol{x}} & \quad -\frac{\mathrm{d}Q_{\boldsymbol{xu}}}{\mathrm{d}t} &= \ell_{\boldsymbol{xu}} + F_{\boldsymbol{x}}^{\mathsf{T}} Q_{\boldsymbol{xu}} + Q_{\boldsymbol{xx}} F_{\boldsymbol{u}} \\ -\frac{\mathrm{d}Q_{\boldsymbol{uu}}}{\mathrm{d}t} &= \ell_{\boldsymbol{uu}} + F_{\boldsymbol{u}}^{\mathsf{T}} Q_{\boldsymbol{xu}} + Q_{\boldsymbol{ux}} F_{\boldsymbol{u}} & \quad -\frac{\mathrm{d}Q_{\boldsymbol{ux}}}{\mathrm{d}t} &= \ell_{\boldsymbol{ux}} + F_{\boldsymbol{u}}^{\mathsf{T}} Q_{\boldsymbol{xx}} + Q_{\boldsymbol{ux}} F_{\boldsymbol{x}} \end{aligned}\right\} \text{generalization to second-order}$$

- 😀 no recursive dependency! (everything's solved in *one single backward pass*)
- 😀 can be extended to computing higher-order tensors

Memory Complexity

- Memory complexity of backward pass ($\mathbf{x}_t \in \mathbb{R}^m, \mathbf{u}_t \in \mathbb{R}^n$, rank R)

	1st-order Adjoint	*Theorem 1*	*Proposed method (2nd-order)*
w.r.t. depth (NFE)	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
w.r.t m, n	$\mathcal{O}(n+m)$	$\mathcal{O}((n+m)^2)$	$\mathcal{O}(2n+Rm)$

  - Practical implementation: adopt *low-rank approximation* & *Kronecker factorization*.
  - 10-40% additional *constant* memory compared to 1st-order Adjoint.
    (less than 1GB on all experiments)

NFE: number of function evaluation

Per-Iteration Runtime

- Per-iteration runtime (seconds) w.r.t. Adam

	Image Classification			Time-series Prediction			Continuous NF		
	MNIST	SVHN	CIFAR10	SpoAD	ArtWR	CharT	Circle	Gas	Minib.
$\frac{\text{Ours}}{\text{Adam}}$	1.00	0.87	1.16	0.99	1.01	1.01	2.75	1.93	1.60

😀 run nearly as fast as Adam!
😲 may run *faster* (*e.g.*, SVHN)!

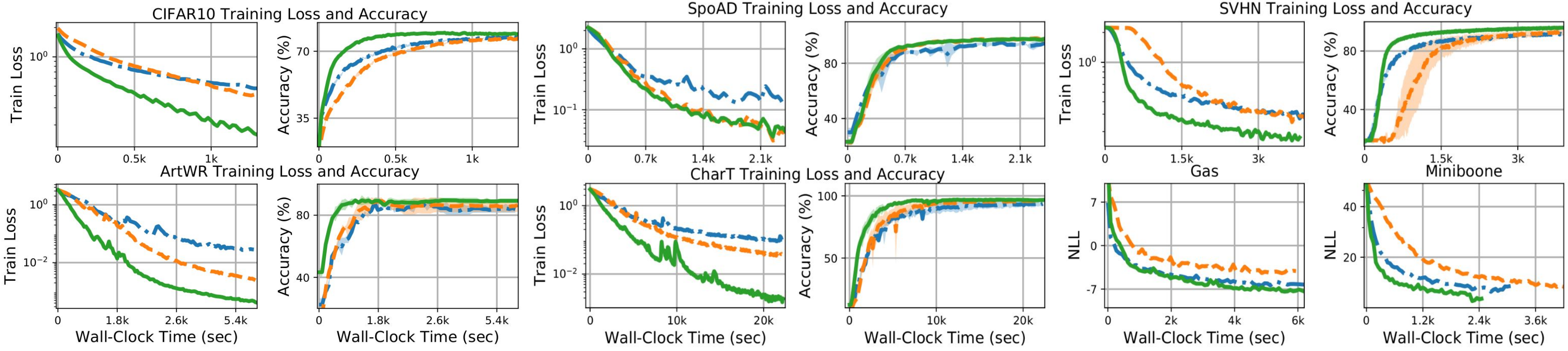
😐 1.5~3x slower on CNF datasets
😀 still faster convergence (next slide)

- 2~5x faster than recursive adjoint baseline
- Preconditioned updates may lead to implicit regularization.

	# of function evaluation (NFE)		*Regularization* (*Finlay et al., 2020*)
	forward pass	*backward pass*	($\int\|\|\nabla_{\mathbf{x}}F\|\|^2 + \int\|\|F\|\|^2$)
Adam	*32.0*	*42.1*	*323.9 (100%)*
Ours	***26.0***	***32.6***	***199.1 (61.5%)***

Empirical Performance

- Superior convergence in wall-clock time (Adam, SGD, Ours)



- Improves test-time performance (accuracy for Image/Time-series, NLL for CNF)

	Image Classification			Time-series Prediction			Continuous NF		
	MNIST	SVHN	CIFAR10	SpoAD	ArtWR	CharT	Circle	Gas	Minib.
Adam	98.83	91.92	77.41	94.64	84.14	93.29	0.90	-6.42	13.10
SGD	98.68	93.34	76.42	**97.70**	85.82	95.93	0.94	-4.58	13.75
Ours	**98.99**	**95.77**	**79.11**	97.41	**90.23**	**96.63**	**0.86**	**-7.55**	**12.50**

Application to Architecture Optimization

Consider an extension of $Q(t, \mathbf{x}_t, \mathbf{u}_t)$ that includes the terminal horizon t_1.

$$Q(t, \mathbf{x}_t, \mathbf{u}_t) := \Phi(\mathbf{x}_{t_1}) + \int_t^{t_1} \ell(\tau, \mathbf{x}_\tau, \mathbf{u}_\tau) \mathrm{d}\tau$$

$\widetilde{\Phi} := \Phi +$ penalty on T

$$\widetilde{Q}(t, \mathbf{x}_t, \mathbf{u}_t, \mathrm{T}) := \widetilde{\Phi}(\mathrm{T}, \mathbf{x}_\mathrm{T}) + \int_t^{\mathrm{T}} \ell(\tau, \mathbf{x}_\tau, \mathbf{u}_\tau) \mathrm{d}\tau$$

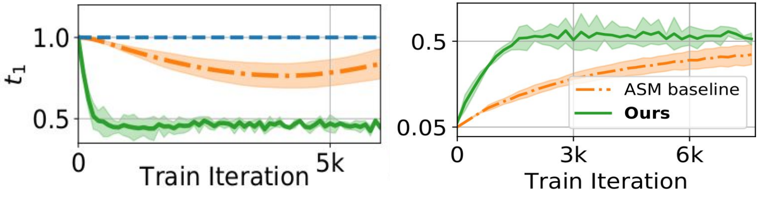
iterative *feedback* update rule

$\mathrm{T} \leftarrow T - \delta\mathrm{T}(\delta\theta)$, where

$$\delta\mathrm{T}(\delta\theta) = [\widetilde{Q}_{TT}(t_0)]^{-1} \left(\widetilde{Q}_T(t_0) + \widetilde{Q}_{T\mathbf{u}}(t_0)\delta\theta\right)$$

self-adaptive policy accounting for parameter update

Terminal Horizon (t_1) Optimization on CIFAR10



😀 fast convergence to desired t_1 without divergence (as in baseline)

😀 reduce runtime by 20% without hindering test-time accuracy

Conclusion

A new second-order optimizer for training Neural ODEs that

- grounded on optimal control theory
- generalizes first-order adjoint method while retaining the same constant ***O(1)*** memory (in depth)
- achieves strong empirical results (*e.g.*, convergence & test-time performance)
- opens up new applications and questions (*e.g.*, architecture optimization, implicit regularization)

Paper **Poster**

Reference

Weinan et al., 2018, “A mean-field optimal control formulation of deep learning.”
Liu & Theodorou, 2019, “Deep learning theory review: An optimal control and dynamical systems perspective.”
Liu et al., 2021a, “DDPNOpt: Differential dynamic programming neural optimizer.”
Liu et al., 2021b, “Dynamic game-theoretic neural optimizer.”
Pontryagin et al., 1962, "The mathematical theory of optimal processes."
Finlay et al., 2020, "How to train your neural ode: the world of jacobian and kinetic regularization."