COSINE TOLERANCE ANNEALING FOR NEURAL ODE TRAINING: A STUDY OF COST AND ACCURACY

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Ordinary Differential Equations (Neural ODEs) are usually trained with adaptive solvers whose tolerances stay constant for the whole run. Loose tolerances keep the number of function evaluations (NFE) low but give less accurate solutions; tight tolerances are accurate but expensive at every iteration. We study *Cosine Tolerance Annealing* (CTA), which tightens the relative and absolute tolerances from loose initial values to tight final values along a cosine curve over the first 60% of training, and compare it with constant tolerances and other schedules. On a synthetic cubic system with the Dormand–Prince solver and ten seeds, CTA has on every seed a lower error, averaged over the last ten checkpoints, than each of the four constant tolerances we tested; relative to the tight final tolerance (Fixed-Final), it lowers this error by 62% and the training NFE by 21%. Across five synthetic settings, CTA needs 13–40% fewer training NFE than Fixed-Final, with an error that is lower in two settings and not significantly different in the other three; a linear schedule is not significantly different from the cosine one in four of them. On MNIST (three seeds) and Jena Climate forecasting (ten seeds), CTA reaches an accuracy similar to that of Fixed-Final with 22.8% and 17.8% fewer function evaluations per training iteration, but a constant tolerance between the initial and final values (Fixed-Geometric) reaches a similar accuracy at a lower average cost. On the two synthetic systems, however, this constant tolerance is significantly less accurate than Fixed-Final, and CTA has a lower error than it in four of the five settings, so CTA is useful when it is not known in advance whether a loose constant tolerance is accurate enough.

1 INTRODUCTION

Neural Ordinary Differential Equations (Neural ODEs; Chen et al., 2018) model hidden representations as continuous-time dynamical systems solved by adaptive ODE solvers, so the solver configuration affects both numerical accuracy and computational cost. Common Neural ODE implementations (Chen et al., 2018; Grathwohl et al., 2019; Dupont et al., 2019) keep the solver tolerances constant throughout training. A loose tolerance lowers the cost of every iteration, but whether it is accurate enough for a task, that is, as accurate as a tight one, is usually not known before training. Ott et al. (2021) adapt the step size or tolerance during training with feedback from a more accurate solver, and InfoCNF (Nguyen et al., 2019) learns the tolerances with gating networks. We study a simpler alternative that needs neither feedback nor learned parameters: *Cosine Tolerance Annealing* (CTA) tightens the relative ($rtol$) and absolute ($atol$) tolerances along a cosine, following a schedule that is set before training and applied unchanged to every task. We ask whether such a schedule lowers the training cost without raising the task error. CTA lowered the training cost relative to the tight final tolerance (Fixed-Final) in every setting we tested, with no significant loss of accuracy in any ten-seed setting. Fixed-Geometric, a constant tolerance at the geometric midpoint of the initial and final values ($rtol = 10^{-3}$), was cheaper still at a similar accuracy on MNIST and Jena Climate but significantly less accurate than Fixed-Final on the two synthetic systems, so CTA is useful when it is not known in advance whether a loose constant tolerance is accurate enough.

The main contributions of this work are:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- We state tolerance scheduling as a cost-constrained problem and use a known bound for gradient descent with inexact gradients to motivate schedules that are loose early and tight late (Section 3.2).
- CTA lowers the training cost relative to Fixed-Final in every setting we tested, by 13–40% (training NFE) in five synthetic settings and by 22.8% and 17.8% (NFE per training iteration) on MNIST and Jena Climate (Tables 2–4), with no significant loss of accuracy in any ten-seed setting and a similar accuracy on MNIST (three seeds).
- In the five synthetic settings, CTA has a lower error than Fixed-Geometric in four and a significantly lower error than Fixed-Final in two (by 62.2% on the cubic system with Dopri5; Section 4.3.2), and a linear schedule is not significantly different from CTA in four, so the data do not favor the cosine shape.

2 Related Work

Neural ODEs and their cost. Neural ODEs (Chen et al., 2018; Kidger, 2022) have been extended to free-form continuous normalizing flows (Grathwohl et al., 2019), augmented state spaces (Dupont et al., 2019), irregularly sampled time series (Rubanova et al., 2019; Kidger et al., 2020) and other continuous-depth models (Massaroli et al., 2020). The number of function evaluations (NFE) of an adaptive solver often grows as training proceeds (Finlay et al., 2020; Kelly et al., 2020), and several regularizers make the learned dynamics cheaper to solve. Finlay et al. (2020) penalize the kinetic energy of the flow and the Frobenius norm of its Jacobian, and Kelly et al. (2020) penalize higher-order time derivatives of the solution. Pal et al. (2021) penalize the local error and stiffness estimates that the solver already computes. Ghosh et al. (2020) sample the end time of the integration at random during training, and hypersolvers (Poli et al., 2020) add a learned correction to a low-order fixed-step solver to make inference cheaper. CTA only sets the tolerances, so it could be combined with these regularizers, but we have not tested this.

Solver accuracy during training. In adaptive solvers such as Dormand–Prince (Dopri5; Dormand & Prince, 1980; Shampine, 1986), $rtol$ and $atol$ bound the estimated local error of each step and so set both the accuracy and the NFE (Hairer et al., 1993; Butcher, 2016). The works closest to ours are Ott et al. (2021) and InfoCNF (Nguyen et al., 2019). Ott et al. (2021) show that a model trained with too large a step size or too loose a tolerance loses accuracy when it is tested with a more accurate solver, so it no longer behaves as an ODE flow. Their algorithm periodically compares the accuracy under the training solver with that under a more accurate solver, decreases the step size or tolerance when the two differ by more than a threshold and increases it otherwise, so that the model stays a valid ODE flow at a small NFE. InfoCNF instead attaches learned gating networks to the solvers of a conditional continuous normalizing flow to choose their error tolerances, and reports fewer function evaluations on CIFAR10. Unlike both methods, CTA uses no feedback from the model and no learned component: its tolerances follow a schedule set in advance and only move from loose to tight. Sections 4.3.2–4.5 check whether the models trained with CTA are tied to their training tolerance in the sense of Ott et al. (2021).

Gradient computation. The adjoint method (Chen et al., 2018) solves an adjoint ODE backward in time with a memory cost that does not grow with the number of solver steps. ANODE (Gholami et al., 2019) and the Adaptive Checkpoint Adjoint method (Zhuang et al., 2020) use checkpointing to obtain more accurate gradients, Onken & Ruthotto (2020) compare discretize-then-optimize with optimize-then-discretize training, and Kidger et al. (2021) lower the backward NFE of the adjoint method with a seminorm that leaves the parameter-gradient components out of the error estimate. These methods change how the gradient is computed or how the backward solve controls its error, whereas CTA only sets the tolerances; in our adjoint runs (MNIST, Jena Climate) the backward solve uses the same scheduled tolerances as the forward solve.

Schedules during training. Learning rates are often varied with cosine (Loshchilov & Hutter, 2017) or cyclical (Smith, 2017) schedules, and curriculum learning orders the examples from easy to hard (Bengio et al., 2009). CPT (Fu et al., 2021) cycles the numerical precision of low-precision training between two bit widths. Dynamic sample-size methods (Byrd et al., 2012; Friedlander & Schmidt, 2012) compute early gradients on small samples and enlarge the sample later. Adaptive

computation time (Graves, 2016) lets a recurrent network learn how many steps to spend on each input. In a Neural ODE the adaptive solver already sets the number of steps of each solve (here, of each mini-batch) through its error control, and CTA changes only the tolerances that the solver targets. Like the sample-size methods, CTA uses cheap and inexact computation early and more accurate computation later. Unlike CPT, it does not cycle: it tightens the tolerances over a fixed number of iterations and then holds them at their final values.

3 METHODOLOGY

3.1 NEURAL ODES WITH ADAPTIVE ERROR CONTROL

Neural ODEs model hidden-state evolution as a continuous-time dynamical system parameterized by a neural network:

$$\frac{dh(t)}{dt} = f(h(t), t; \theta), \tag{1}$$

where $h(t) \in \mathbb{R}^d$ is the hidden state, $f(\cdot)$ is a neural network parameterized by θ, and $t \in [t_0, t_1]$ denotes the continuous-depth interval. The final state $h(t_1) = h(t_0) + \int_{t_0}^{t_1} f(h(t), t; \theta)\, dt$ is approximated by an adaptive ODE solver, so the cost of a Neural ODE depends on the integration and is measured by the NFE.

Adaptive solvers adjust their integration step sizes according to local error estimates controlled by the relative and absolute tolerances, $rtol$ and $atol$ (either is denoted tol):

$$\mathrm{err}_n \leq atol + rtol \cdot \max(|y_n|, |y_{n+1}|). \tag{2}$$

For an adaptive Runge–Kutta pair whose lower-order solution has order p ($p = 4$ for Dopri5 and Tsit5; Dormand & Prince, 1980; Tsitouras, 2011), the step size Δt approximately follows (Hairer et al., 1993; Butcher, 2016)

$$\Delta t \propto tol^{\frac{1}{p+1}}. \tag{3}$$

3.2 OPTIMIZATION FORMULATION

Decision variable. A tolerance schedule $S = \{(rtol_i, atol_i)\}_{i=1}^{N_{\text{train}}}$ sets the tolerances of the forward solve, and of the adjoint solve where one is used, in each of the N_{train} training iterations. We choose S; the parameters θ are trained by the optimizer of each task, and $\theta^*(S)$ denotes the parameters obtained by training under S.

Problem. Let $\mathcal{L}_{\text{val}}(\theta^*(S))$ be the task error of $\theta^*(S)$ (Section 4.2), measured on the validation set for Jena Climate, on the test set for MNIST, on which no schedule is selected, and, for the synthetic systems, on the trajectory from which the training subsequences are drawn. Let $C(S) = \sum_i \mathrm{NFE}_i(S)$ be the training cost, where $\mathrm{NFE}_i(S)$ is the NFE of the solves in iteration i and the sum runs over the iterations trained; both are averaged over random seeds. We formulate tolerance scheduling as

$$\min_S \ \mathcal{L}_{\text{val}}(\theta^*(S)) \qquad \text{s.t.} \qquad C(S) \leq B \tag{4}$$

for a computational budget B. By Eq. (3), a solve over $[t_0, t_1]$ takes $(t_1 - t_0)/\Delta t \propto tol^{-1/(p+1)}$ steps, so $C(S) \approx \sum_i \kappa_i\, rtol_i^{-1/(p+1)}$ ($atol_i = rtol_i/10$ in all our schedules), where κ_i depends on the learned dynamics, and hence on S, and changes during training. On the error side, the solves of iteration i give the gradient of the training loss ℓ with an error e_i that depends on tol_i. For gradient descent with step size $1/L$ on an L-smooth, μ-strongly convex ℓ with minimum ℓ^*, Friedlander & Schmidt (2012, Lemma 2.1) show that $\ell(\theta_i) - \ell^* \leq (1 - \mu/L)\,[\ell(\theta_{i-1}) - \ell^*] + \|e_i\|^2/(2L)$; applying this bound for $i = 1, \ldots, N_{\text{train}}$ gives

$$\ell(\theta_{N_{\text{train}}}) - \ell^* \leq \left(1 - \frac{\mu}{L}\right)^{N_{\text{train}}} \left[\ell(\theta_0) - \ell^*\right] + \frac{1}{2L} \sum_{i=1}^{N_{\text{train}}} \left(1 - \frac{\mu}{L}\right)^{N_{\text{train}} - i} \|e_i\|^2. \tag{5}$$

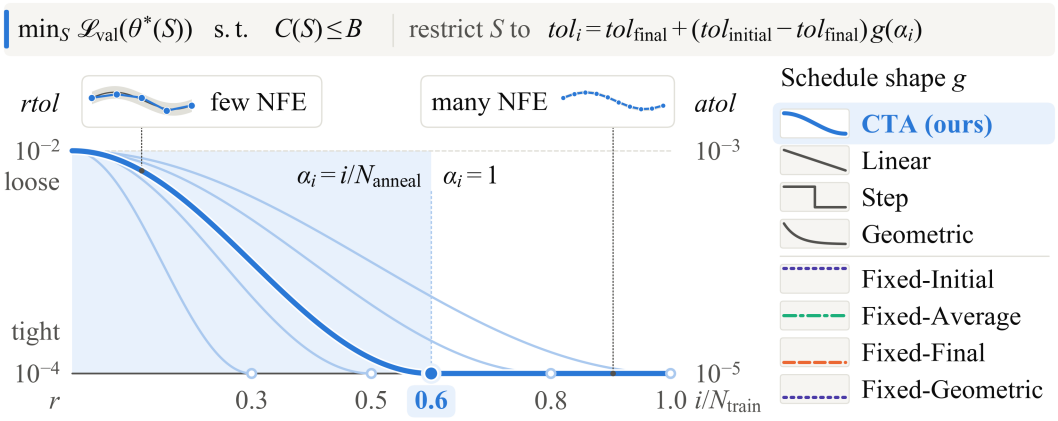

Figure 1: Optimization formulation for solver-tolerance scheduling. Left: $rtol$ and $atol$ over training on linear axes; with annealing ratio r, the tolerances reach their final values at iteration rN_{train} (shaded for CTA). Bold: CTA with $r = 0.6$, used in our experiments; light: the other ratios. Right: the evaluated shapes g over $\alpha \in [0, 1]$; the geometric schedule interpolates in $\log tol$, and Fixed-Geometric keeps its midpoint value. Insets: schematic solver steps inside a tolerance band (gray).

The error of iteration i thus enters with a weight that decays geometrically in the number of later iterations, so a schedule that is loose early and tight late saves cost in the iterations whose error has the smallest weight in Eq. (5). Our training problems are not convex and use Adam or SGD with momentum, and κ_i and e_i have no closed form, so we use this argument only to choose the search space, and we compare schedules in that space empirically under Eq. (4).

Search space and optimization. Each evaluation of $\mathcal{L}_{\text{val}}(\theta^*(S))$ requires training the model with every seed, so we restrict S to schedules of the form

$$tol_i = tol_{\text{final}} + (tol_{\text{initial}} - tol_{\text{final}})\, g(\alpha_i), \qquad \alpha_i = \min\left(1, \frac{i}{N_{\text{anneal}}}\right), \tag{6}$$

where the shape g maps $[0, 1]$ into $[0, 1]$, and the annealing horizon $N_{\text{anneal}} = rN_{\text{train}}$ is set by the annealing ratio r. The annealed shapes (cosine, linear, geometric, step) decrease from $g(0) = 1$ to $g(1) = 0$, and the constant shapes give the baselines Fixed-Initial ($g \equiv 1$), Fixed-Average ($g \equiv \frac{1}{2}$), Fixed-Final ($g \equiv 0$) and Fixed-Geometric ($tol_i = \sqrt{tol_{\text{initial}}\, tol_{\text{final}}}$) (Figure 1; formulas in Appendix B.6). With the initial and final tolerances fixed in each experiment, the decision variables are g and r. On the cubic system of Eq. (8) with ten seeds, we evaluate $r \in \{0.3, 0.5, 0.6, 0.8, 1.0\}$ for the cosine shape (Section 4.3.1) and the eight shapes at $r = 0.6$ (Section 4.3.2). For a budget B, the search selects the evaluated schedule with the lowest mean task error among those with $C(S) \le B$. CTA, the schedule with the cosine shape and $r = 0.6$ (Section 3.3), was chosen before these runs, which check this choice and give the reported cubic-system results of CTA; four more synthetic settings (Table 2), MNIST and Jena Climate test CTA without further tuning.

3.3 Cosine Tolerance Annealing (CTA)

CTA uses the shape of cosine learning-rate annealing (Loshchilov & Hutter, 2017) with $r = 0.6$:

$$tol_i = tol_{\text{final}} + \frac{1}{2}\,(tol_{\text{initial}} - tol_{\text{final}})\,(1 + \cos(\pi\alpha_i))\,, \tag{7}$$

with α_i from Eq. (6), applied independently to $rtol$ and $atol$ with their respective initial and final values. The tolerances are loose early in training and are held at their final values after N_{anneal} iterations. Computing the schedule adds negligible cost, and the tolerances can be passed to an unmodified adaptive solver (Appendix B.6); Algorithm 1 in Appendix A gives the procedure.

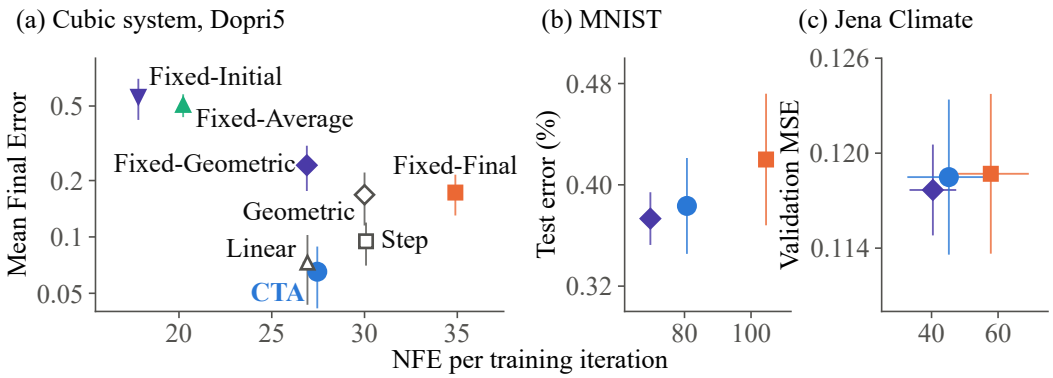

Figure 2: Error against training cost. Markers show the mean over seeds (ten for the cubic system and Jena Climate, three for MNIST), and bars show ± 1 standard deviation; (b) and (c) use the markers of (a). The NFE per training iteration counts forward evaluations for the cubic system, which backpropagates through the solver, and forward plus backward evaluations for MNIST and Jena Climate (adjoint method; for Jena Climate, up to the best checkpoint).

4 Experiments

4.1 Experimental Setup

We evaluate CTA on synthetic dynamical systems, MNIST image classification (LeCun et al., 1998) and Jena Climate long-horizon forecasting (Attri et al., 2020). The synthetic systems are a cubic system from the examples of `torchdiffeq` (Chen, 2018) and, in Table 2 and Appendix B.7, a Lotka–Volterra system (Lotka, 1925; Volterra, 1926). On every task, we compare CTA with Fixed-Final, whose accuracy it should match, and with Fixed-Geometric, a loose constant tolerance (Figure 2).

The synthetic and Jena Climate experiments use ten seeds (123, 3407, 42 and 1–7); MNIST and the three-seed synthetic runs (Appendices B.3–B.5) use the first three. Appendices B–D give the model architectures, solver configurations, tolerance settings and training hyperparameters.

4.2 Evaluation Metrics

Task error. For the synthetic systems, we use *Mean Final Error*, the mean prediction error (a mean squared error; Appendix B.1) over the last 10 evaluation checkpoints (iterations 4100–5000). For MNIST, we use the test error and report classification accuracy. For Jena Climate, the task error is the validation mean squared error (MSE) of the best checkpoint saved under the early-stopping criterion (minimum improvement 10^{-5}, Appendix D).

Computational efficiency. NFE counts evaluations of the ODE function during numerical integration; unless otherwise specified, it counts forward-pass evaluations. For the synthetic experiments, the training NFE (Train NFE in the tables) is the NFE per training iteration, averaged over all iterations (that is, $C(S)/N_{\text{train}}$), and Mean Eval NFE is the NFE of an evaluation pass over the 1,000 time points of the trajectory, averaged over the evaluation checkpoints (one every 100 training iterations); each evaluation pass uses the tolerances in effect at that iteration. For MNIST, we also report backward-pass evaluations, the total NFE of training and the time per training iteration. For Jena Climate, we report the NFE per training iteration (forward plus backward) and the total NFE of training, each up to the best checkpoint and over the whole run.

4.3 Synthetic Dynamical System Experiments

4.3.1 Solver and Annealing Ratio

In the three-seed runs (Appendices B.3–B.5), CTA had its lowest error with Dopri5 among Adaptive Heun (Hairer et al., 1993), Dopri5 and Tsit5 (Tsitouras, 2011), so we use Dopri5; this choice favors CTA, whose error reduction relative to Fixed-Final is not significant with Tsit5 (Table 2). With ten

Table 1: Ten-seed comparison of tolerance schedules on the cubic system with Dopri5 (Appendix B.6). Mean Final Error is the mean ± standard deviation over seeds; Train NFE is the forward NFE per training iteration, averaged over all iterations and seeds. The last two columns compare CTA with each schedule on the same seeds (two-sided Wilcoxon signed-rank test).

Schedule	Mean Final Error	Mean Eval NFE	Train NFE	CTA lower	p
Cosine (CTA)	0.0652 ± 0.0236	425.7	27.45	–	–
Linear	0.0729 ± 0.0295	419.2	26.92	6/10	0.492
Step	0.0948 ± 0.0245	474.1	30.08	8/10	0.020
Geometric	0.1679 ± 0.0528	454.2	30.00	10/10	0.002
Fixed-Final	0.1725 ± 0.0424	538.9	34.90	10/10	0.002
Fixed-Geometric	0.2414 ± 0.0655	370.4	26.89	10/10	0.002
Fixed-Average	0.5069 ± 0.0708	330.7	20.23	10/10	0.002
Fixed-Initial	0.5601 ± 0.1386	250.2	17.81	10/10	0.002

seeds (Table 11), $r = 0.6$ has the lowest Mean Final Error among the five ratios (0.065; $r = 0.8$: 0.078), $r = 0.5$, 0.8 and 1.0 are also lower than Fixed-Final on at least 9 of 10 seeds, and $r = 0.3$ is not distinguishable from Fixed-Final (6 of 10 seeds, $p = 0.38$).

4.3.2 Constant Tolerances, Other Schedules and Other Settings

We compare CTA over ten seeds with the four constant tolerances (Fixed-Geometric: $rtol = 10^{-3}$, $atol = 10^{-4}$) and, to separate the effect of the cosine shape from that of annealing itself, with the linear, geometric and step schedules of Section 3.2, with the same initial and final tolerances and $r = 0.6$ (the step schedule switches at $0.3N_{\text{train}}$). These runs use the unmodified `torchdiffeq` solver on a CPU (Appendix B.6), so their values differ from those of the three-seed runs (Appendix A).

Table 1 and Figure 2(a) show that CTA has a lower Mean Final Error than each of the four constant tolerances on all ten seeds ($p = 0.002$, or $p = 0.014$ after a Holm correction (Holm, 1979) for the seven comparisons), including Fixed-Geometric, which uses about 2% fewer training NFE (26.89 vs. 27.45 per iteration) but ends with 3.7 times the error. Relative to Fixed-Final, CTA lowers the Mean Final Error by 62.2% (95% bootstrap interval 53–69%; Efron & Tibshirani, 1994) and the Mean Eval NFE by 21.0%, in line with the three-seed runs (Table 8). The NFE reduction comes from the annealing phase: over the last ten checkpoints, where both use the final tolerances, the evaluation NFE is 584 for CTA and 580 for Fixed-Final, so CTA lowers the cost of training but not that of evaluating the trained model. On seeds 1–7, which the three-seed runs did not use, CTA is also lower than each constant tolerance on all seven ($p = 0.016$ for each comparison).

In the search of Section 3.2 (Table 1 and the bottom block of Table 11), no schedule has a lower Mean Final Error than CTA, so CTA has the lowest one at any budget B at or above its cost, but the linear schedule and the ratios $r = 0.8$ and 1.0, which need 2–20% fewer training NFE, are not significantly different from CTA ($p = 0.49$, 0.13 and 0.06). Annealing alone does not give a significantly lower error: the geometric schedule, which also tightens gradually but already reaches $rtol = 10^{-3}$ at $0.3N_{\text{train}}$, is not significantly more accurate than Fixed-Final (lower on 6 of 10 seeds, $p = 0.85$), and the step schedule lies between CTA and the geometric schedule. Re-evaluating the last ten checkpoints with a tighter tolerance ($rtol = 10^{-7}$, $atol = 10^{-9}$) changes the ten-seed Mean Final Error of the annealed schedules and Fixed-Final by less than 0.001, lowers that of Fixed-Initial and Fixed-Average to 0.486 and 0.441, and raises that of Fixed-Geometric to 0.255, so the ranking does not change under this evaluation tolerance. Over the whole run, the largest change of the seed-averaged prediction error between consecutive checkpoints is similar for CTA and Fixed-Final (0.215 vs. 0.231). From iteration 3000 on, where both use the final tolerances, it is 0.113 vs. 0.133, and the largest change within a seed is smaller for CTA on 9 of 10 seeds (mean 0.35 vs. 0.58, $p = 0.004$). With the constant learning rate of Adam, the error of every schedule oscillates between checkpoints: at a single one of the last ten checkpoints, CTA is lower than Fixed-Final on 4–8 of the 10 seeds (5 at iteration 5000), and the lowest of these ten errors is lower for Fixed-Final than for CTA on 8 seeds. The gain lies in the level of the late error: the seed-averaged error of CTA is lower at each of the last ten checkpoints, and averaging over the last 5, 15 or 20 checkpoints instead of 10 gives reductions of 48–64% (CTA lower on 9 or 10 of 10 seeds).

Table 2: CTA and Fixed-Final in five synthetic settings, ten seeds each (Appendix B.7). "Cubic" is the system of Eq. (8). The tolerance pair gives $rtol$ from initial to final ($atol$ is ten times smaller); Fixed-Final uses the final value. The last two columns give the NFE reduction of CTA.

Setting	Mean Final Error CTA	Mean Final Error Fixed-Final	CTA lower	p	NFE reduction (%) Eval	NFE reduction (%) Train
Cubic, Dopri5, $10^{-2} \to 10^{-4}$	0.0652	0.173	10/10	0.002	21.0	21.3
Cubic, Dopri5, $10^{-2} \to 10^{-6}$	0.0756	0.166	10/10	0.002	39.9	39.6
Cubic, Dopri5, $10^{-3} \to 10^{-5}$	0.178	0.166	4/10	0.375	25.5	22.1
Cubic, Tsit5, $10^{-2} \to 10^{-4}$	0.149	0.151	4/10	0.922	18.6	15.6
Lotka–Volterra, Dopri5, $10^{-2} \to 10^{-4}$	1.474	1.080	6/10	1.000	26.3	13.4

Table 2 repeats the comparison with Fixed-Final in four more settings: the tolerance pairs $10^{-3} \to 10^{-5}$ and $10^{-2} \to 10^{-6}$, the Tsit5 solver, and a Lotka–Volterra system (Appendix B.7, Tables 10 and 11). In all five settings CTA needs 13–40% fewer training NFE and 18–40% fewer evaluation NFE than Fixed-Final. Its error is lower in the two Dopri5 settings that start from $rtol = 10^{-2}$ on the cubic system and not significantly different in the other three, where its mean is higher in two cases (0.178 vs. 0.166 with the pair $10^{-3} \to 10^{-5}$, 1.474 vs. 1.080 on Lotka–Volterra). On Lotka–Volterra, the error of every schedule oscillates strongly during training, and the mean of CTA is driven by one seed (8.57; the medians are 0.77 for CTA and 0.73 for Fixed-Final). The step schedule has the lowest Mean Final Error (0.654) and is lower than CTA on all ten seeds ($p = 0.002$), so on this system the cosine shape is not the most accurate one. In the other four settings, the linear schedule is not significantly different from CTA, and in all settings except the pair $10^{-3} \to 10^{-5}$ ($p = 1.000$), CTA has a lower error than Fixed-Geometric ($p \leq 0.01$). In the three settings where Fixed-Geometric trains at $rtol = 10^{-3}$, its Mean Final Error is also 1.4–2.8 times that of Fixed-Final (higher on 8–9 of 10 seeds, $p \leq 0.01$; Appendix B.7).

4.4 MNIST Image Classification Experiments

We next test whether the lower cost carries over to image classification. An ODE-Net (Chen et al., 2018) is trained on MNIST for 160 epochs with Dopri5 and the adjoint method, with three seeds per method. Besides Fixed-Final, we include Fixed-Geometric ($rtol = 10^{-3}$, $atol = 10^{-4}$). All methods use the same training script and differ only in the tolerances (Appendix C).

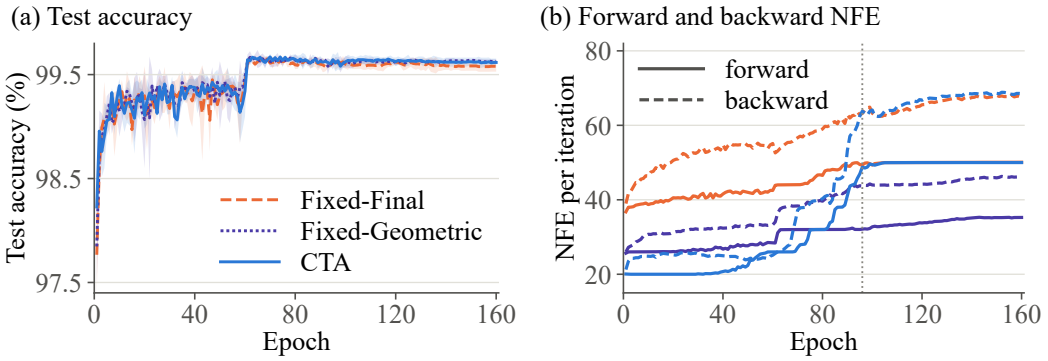

Figure 3: MNIST, mean over three seeds. (a) Test accuracy; the bands show ± 1 standard deviation. (b) Forward (solid) and backward (dashed) NFE per training iteration, averaged over the iterations of each epoch, with colors as in (a); the dotted line marks the end of annealing (epoch 96).

The test accuracies of the three methods track each other throughout training (Figure 3(a)). After epoch 1, where CTA is 0.46 percentage points above Fixed-Final, the seed means of the two differ by at most 0.35 percentage points. The final accuracies are 99.62% for CTA, 99.58% for Fixed-Final and 99.63% for Fixed-Geometric (Table 3). Evaluating the final models with a tighter tolerance ($rtol = 10^{-6}$, $atol = 10^{-7}$) changes the test accuracy of each seed by at most 0.01 percentage points, so no final model is tied to its training tolerance in the sense of Ott et al. (2021).

Table 3: Fixed-Final, Fixed-Geometric and CTA on MNIST (mean $\pm$ standard deviation over three seeds). The forward and backward NFE rows are per training iteration, averaged over all iterations; the total training NFE is their sum over all $74{,}880$ training iterations. The time per iteration is measured on saved checkpoints while no other run of ours uses the GPU (Appendix C).

Metric	Fixed-Final	Fixed-Geometric	CTA
Final test accuracy (%)	99.58 ± 0.05	99.63 ± 0.02	99.62 ± 0.04
Forward NFE	45.71 ± 0.56	30.91 ± 0.69	35.42 ± 0.42
Backward NFE	58.81 ± 1.17	38.83 ± 0.97	45.30 ± 0.95
Total training NFE (10^6)	7.83 ± 0.13	5.22 ± 0.07	6.04 ± 0.10
Time per iteration (ms)	132.9 ± 2.2	92.7 ± 1.7	104.8 ± 1.3

Until the end of annealing, the seed mean of the forward NFE of CTA is lower than that of Fixed-Final at every epoch, and that of the backward NFE at every epoch up to epoch 95 (Figure 3(b)); over epochs 1–96, its forward plus backward NFE per iteration is 41.5% lower (56.7 vs. 96.9). From epoch 97 on, both use the final tolerances, and their NFE per iteration are nearly equal (116.7 vs. 115.9). Over the whole run, CTA uses 22.8% fewer function evaluations than Fixed-Final, fewer on each of the three seeds, and its training iterations take 21.1% less time. With the final tolerances, the final models of all three methods need 50 forward evaluations per test batch, so relative to Fixed-Final, CTA lowers the cost of training but not that of inference.

Fixed-Geometric is cheaper than CTA (Figure 2(b)): it uses 13.6% fewer function evaluations, fewer on each seed, and its iterations take 11.5% less time, at a similar final accuracy. CTA needs fewer evaluations than Fixed-Geometric in the first 75 epochs, but after annealing it solves with the final tolerances, whereas Fixed-Geometric stays at $rtol = 10^{-3}$ (116.7 vs. 79.1 NFE per iteration from epoch 97 on). In our three seeds, training at $rtol = 10^{-3}$ (the default $rtol$ of the `odenet_mnist` example, Appendix C) instead of 10^{-4} did not lower the mean final accuracy on MNIST, so this constant tolerance lowers the training cost more than CTA, whose saving over Fixed-Final matters only when 10^{-3} is not known to be accurate enough. This differs from the cubic system, where Fixed-Geometric ends with 3.7 times the Mean Final Error of CTA (Table 1).

4.5 Jena Climate Forecasting Experiments

We then test whether the MNIST results also hold for multivariate long-horizon forecasting with the Jena Climate dataset, using an LSTM encoder followed by a Neural ODE block that is trained with the adjoint method (Appendix D). Training stops when the validation MSE has not improved for five epochs, at the latest after 50 epochs, and the annealing horizon is 60% of the 50 epochs, so annealing ends at epoch 30. As on MNIST, we compare CTA with Fixed-Final and Fixed-Geometric; all methods use the same training script and differ only in the tolerances. Because the epoch of the best checkpoint varies strongly between seeds, we use ten seeds.

Table 4: Fixed-Final, Fixed-Geometric and CTA on Jena Climate over ten seeds (mean $\pm$ standard deviation). NFE counts forward and backward evaluations. The NFE per iteration is averaged over the epochs up to the best checkpoint or over all epochs trained; the total NFE is summed over the training iterations up to the best checkpoint or over the whole run. Table 14 gives the values of each seed.

Metric	Fixed-Final	Fixed-Geometric	CTA
Best validation MSE	0.1187 ± 0.0050	0.1177 ± 0.0029	0.1185 ± 0.0049
Epoch of best checkpoint	32.9 ± 9.6	32.2 ± 7.0	31.6 ± 9.1
NFE per iteration, up to best	57.8 ± 11.4	40.4 ± 7.0	45.2 ± 12.5
NFE per iteration, whole run	62.3 ± 11.7	43.4 ± 7.0	51.2 ± 13.3
Total NFE up to best (10^6)	2.33 ± 1.07	1.57 ± 0.62	1.78 ± 0.94
Total NFE of whole run (10^6)	2.85 ± 1.14	1.91 ± 0.59	2.31 ± 1.07

The best validation MSE of the three methods is not significantly different (Table 4): CTA is lower than Fixed-Final on 4 of 10 seeds ($p = 0.846$; 95% bootstrap interval of the change in mean MSE -1.6% to $+1.3\%$, computed as in Appendix B.6) and lower than Fixed-Geometric on 4 of 10 seeds ($p = 0.492$). On 5 of 10 seeds, the best checkpoint of CTA comes before the end of annealing (epochs 15–29), when its tolerances are still looser than those of Fixed-Final ($rtol$ between 1.3×10^{-4} and 5.1×10^{-3}); on seed 42, early stopping ends training at epoch 20, before annealing is complete. Evaluating the best checkpoints again with the final tolerances, or with $rtol = 10^{-6}$ and $atol = 10^{-7}$, changes the validation MSE by at most 3×10^{-6} for CTA and 2×10^{-5} for Fixed-Geometric, so this comparison is unchanged under these evaluation tolerances.

Up to the best checkpoint, CTA needs 21.8% fewer function evaluations per training iteration than Fixed-Final, fewer on all ten seeds ($p = 0.002$), and Fixed-Geometric (Figure 2(c)) needs 10.6% fewer than CTA (lower on 7 of 10 seeds, $p = 0.160$). Because the best checkpoint of CTA falls in its cheaper annealing phase on five seeds, we also average over all epochs trained (Table 4): CTA then needs 17.8% fewer evaluations per iteration than Fixed-Final (all ten seeds, $p = 0.002$), and Fixed-Geometric needs 30.3% fewer than Fixed-Final and 15.2% fewer than CTA (lower on 8 of 10 seeds, $p = 0.027$). As on MNIST, training at $rtol = 10^{-3}$ did not raise the mean validation MSE on this task, so Fixed-Geometric gave the lowest cost of the three methods.

5 Conclusion

CTA, with $r = 0.6$ and no per-task tuning, lowered the training cost relative to Fixed-Final in every setting we tested, with no significant loss of accuracy in any ten-seed setting. Fixed-Geometric ($rtol = 10^{-3}$) was cheaper still at a similar accuracy on MNIST and Jena Climate but significantly less accurate than Fixed-Final on the two synthetic systems, so CTA is useful when it is not known in advance whether a loose constant tolerance is accurate enough; when it is known to be, as on MNIST and Jena Climate, the loose constant tolerance is the cheaper choice. On the cubic system with Dopri5, CTA reduced the Mean Final Error by 62.2% and the training NFE by 21.3% relative to Fixed-Final over ten seeds, and its Mean Final Error was lower than that of each of the four constant tolerances on every seed. Across five synthetic settings, CTA needed 13–40% fewer training NFE than Fixed-Final, with an error that was lower in two settings and not significantly different in the other three, and a linear schedule was not significantly different from the cosine one in four of them. In four of the five synthetic settings, CTA had a lower error than Fixed-Geometric ($p \leq 0.01$); with the pair $10^{-3} \rightarrow 10^{-5}$ the two did not differ significantly ($p = 1.000$). On MNIST and Jena Climate, CTA reached an accuracy similar to that of Fixed-Final with 22.8% and 17.8% fewer function evaluations per training iteration, and Fixed-Geometric reached a similar accuracy with 13.6% and 15.2% fewer function evaluations per iteration than CTA. The synthetic and real tasks differ in gradient computation (backpropagation through the solver vs. adjoint), optimizer, learning-rate schedule and model size, and our experiments do not separate these factors, so we have not identified what predicts whether a loose constant tolerance is accurate enough.

The study is limited to monotone schedules and small models. The synthetic systems are two-dimensional and their error is measured on the training trajectory; MNIST accuracy is near its ceiling, with three seeds; the ODE block of our Jena Climate model is small, and our Jena Climate split, like that of the Keras example, has no test part, so early stopping and the reported MSE use the same validation set, which makes the MSE optimistic for all methods; and with early stopping, half of the CTA runs on Jena Climate select a checkpoint before annealing ends. On the real tasks we tested only $r = 0.6$ and no schedule ending at 10^{-3}, and we did not test continuous normalizing flows, on which InfoCNF and Finlay et al. (2020) report. We did not run the feedback algorithm of Ott et al. (2021); its check compares the accuracy under two solvers, so among our tasks it applies directly only to MNIST, where the constant $rtol = 10^{-3}$ already gave a model that is not tied to its training tolerance; there, such feedback might also find a tolerance cheaper than that of CTA.

AI use statement

In this work, we used generative AI tools, including a coding assistant based on a large language model, to design the ten-seed experiments in Sections 4.3.1–4.3.2 and Appendices B.6 and B.7 (the linear, geometric and step schedules, Fixed-Geometric, the Tsit5 solver, the Lotka–Volterra system, and the other tolerance pairs and annealing ratios) and the reruns of the MNIST and Jena Climate

experiments, including the questions these experiments test; to implement and run them, including a reimplementation of the schedules outside the solver, the code that generates the synthetic trajectories of these runs, the training scripts and the statistical tests; and to interpret their results. We have not used generative AI tools to propose the CTA method or to formulate or prove mathematical claims, and the remaining tasks that require disclosure under the ICLR 2027 AI Policy for Authors are not applicable to this work. Additionally, we used generative AI tools to draft and edit parts of the manuscript; to search for related work and draft the descriptions and bibliography entries of the cited papers; to cross-check the reported numbers against the experiment logs and training scripts, which led to corrections of table entries and to reruns of the MNIST experiment and of the three-seed synthetic runs, the latter also implemented and run with the coding assistant, including the change to the training script described in Appendix A; and to design and draft Figure 1 and write the plotting scripts for the other figures. We have reviewed all AI-assisted work: the authors checked the descriptions of the cited papers against the original papers and reviewed the experiment code, the analyses and the text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work uses synthetic trajectories of two dynamical systems and two public benchmark datasets (MNIST and Jena Climate). It involves no human subjects and no personally identifiable information, and we are not aware of ethical concerns arising from it.

Reproducibility statement

The evaluation metrics are defined in Section 4.2, and Algorithm 1 in Appendix A gives the schedule. Model architectures, solver settings, tolerance values, random seeds and training hyperparameters for the three tasks are listed in Appendices B, C and D. The modified `torchdiffeq` code of the three-seed synthetic runs and the training, timing, table and plotting scripts are included in the supplementary material and will be released upon publication, together with the training logs and timing measurements behind every results table and figure.

References

Prabhanshu Attri, Yashika Sharma, Kristi Takach, and Falak Shah. Timeseries forecasting for weather prediction. Keras code examples, 2020. URL `https://keras.io/examples/timeseries/timeseries_weather_forecasting/`.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009.

John Charles Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, third edition, 2016.

Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, 2012. doi: 10.1007/s10107-012-0572-5.

Ricky T. Q. Chen. torchdiffeq, 2018. URL `https://github.com/rtqichen/torchdiffeq`.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

J. R. Dormand and P. J. Prince. A family of embedded Runge–Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural ODEs. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1994.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3154–3164. PMLR, 2020.

Michael P. Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012. doi: 10.1137/110830629.

Yonggan Fu, Han Guo, Meng Li, Xin Yang, Yining Ding, Vikas Chandra, and Yingyan Lin. CPT: Efficient deep neural network training via cyclic precision. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=87ZwsaQNHPZ.

Amir Gholami, Kurt Keutzer, and George Biros. ANODE: Unconditionally accurate memory-efficient gradients for neural ODEs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 730–736, 2019.

Arnab Ghosh, Harkirat Singh Behl, Emilien Dupont, Philip H. S. Torr, and Vinay Namboodiri. STEER: Simple temporal regularization for neural ODE. In *Advances in Neural Information Processing Systems*, volume 33, pp. 14831–14843. Curran Associates, Inc., 2020.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

Alex Graves. Adaptive computation time for recurrent neural networks. arXiv preprint arXiv:1603.08983, 2016.

Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Berlin Heidelberg, second edition, 1993.

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

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Jacob Kelly, Jesse Bettencourt, Matthew J. Johnson, and David K. Duvenaud. Learning differential equations that are easy to solve. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4370–4380. Curran Associates, Inc., 2020.

Patrick Kidger. On neural differential equations. arXiv:2202.02435, 2022. PhD thesis, University of Oxford.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6696–6707. Curran Associates, Inc., 2020.

Patrick Kidger, Ricky T. Q. Chen, and Terry J. Lyons. "Hey, that's not an ODE": Faster ODE adjoints via seminorms. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5443–5452, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Alfred J. Lotka. *Elements of Physical Biology*. Williams & Wilkins, 1925.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural ODEs. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Tan M. Nguyen, Animesh Garg, Richard G. Baraniuk, and Anima Anandkumar. InfoCNF: An efficient conditional continuous normalizing flow with adaptive solvers. arXiv preprint arXiv:1912.03978, 2019.

Derek Onken and Lars Ruthotto. Discretize-optimize vs. optimize-discretize for time-series regression and continuous normalizing flows. arXiv preprint arXiv:2005.13420, 2020.

Katharina Ott, Prateek Katiyar, Philipp Hennig, and Michael Tiemann. ResNet after all? Neural ODEs and their numerical solution. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=HxzSxSxLOJZ.

Avik Pal, Yingbo Ma, Viral Shah, and Christopher V. Rackauckas. Opening the blackbox: Accelerating neural differential equations by regularizing internal solver heuristics. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8325–8335. PMLR, 2021.

Michael Poli, Stefano Massaroli, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Hypersolvers: Toward fast continuous-depth models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21105–21117. Curran Associates, Inc., 2020.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Lawrence F. Shampine. Some practical Runge–Kutta formulas. *Mathematics of Computation*, 46 (173):135–150, 1986.

Leslie N. Smith. Cyclical learning rates for training neural networks. In *IEEE Winter Conference on Applications of Computer Vision*, pp. 464–472, 2017.

Ch. Tsitouras. Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62(2):770–775, 2011.

Vito Volterra. Fluctuations in the abundance of a species considered mathematically. *Nature*, 118 (2972):558–560, 1926.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Juntang Zhuang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan. Adaptive checkpoint adjoint method for gradient estimation in neural ODE. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11639–11649. PMLR, 2020.

A Cosine Tolerance Annealing: Algorithm and Implementation

The following algorithm summarizes the implementation of Cosine Tolerance Annealing (CTA). At each training iteration, the tolerances are computed from Eqs. (6)–(7) and passed to the adaptive solver, which controls the local error as in Eq. (2).

Algorithm 1 Cosine Tolerance Annealing (CTA)

Require: Initial tolerances $(rtol_{\text{initial}}, atol_{\text{initial}})$; final tolerances $(rtol_{\text{final}}, atol_{\text{final}})$; annealing horizon N_{anneal}; total training iterations N_{train}
Ensure: Neural ODE model trained with the CTA tolerance schedule
1: **for** $i = 1, \ldots, N_{\text{train}}$ **do**
2: Compute the normalized annealing progress

$$\alpha_i \leftarrow \min\left(1, \frac{i}{N_{\text{anneal}}}\right)$$

3: Update the relative tolerance using Eq. (7)

$$rtol_i \leftarrow rtol_{\text{final}} + \frac{1}{2}(rtol_{\text{initial}} - rtol_{\text{final}})(1 + \cos(\pi\alpha_i))$$

4: Update the absolute tolerance using Eq. (7)

$$atol_i \leftarrow atol_{\text{final}} + \frac{1}{2}(atol_{\text{initial}} - atol_{\text{final}})(1 + \cos(\pi\alpha_i))$$

5: Solve the forward Neural ODE using $(rtol_i, atol_i)$, with adaptive step-size control according to Eq. (2)
6: Compute the training loss
7: Compute the gradient, by backpropagation through the solver or by the adjoint method with $(rtol_i, atol_i)$, and update the parameters
8: **end for**

The runs in Tables 7, 8 and 9 and Figures 4 and 5 use a modified copy of the `torchdiffeq` library (Chen, 2018) that computes the CTA schedule inside the solver and selects the initial step with the library default tolerances ($rtol = 10^{-7}$, $atol = 10^{-9}$). Before submission, we repeated all runs of these tables with the original training script (for the constant tolerances, with a small change that passes them to both solver calls as the `rtol` and `atol` arguments instead of through the solver options). The repetitions reproduced the reported per-checkpoint mean errors and NFE exactly, except for $r = 0.8$. The original log of that run is not a valid record: at several checkpoints its across-seed standard deviation exceeds what three non-negative errors with its mean allow (e.g. mean 0.0070, standard deviation 0.240 at iteration 2900), and it gave a Mean Final Error of 0.0831 instead of 0.0645. We therefore report the repetition in Table 9; with it, $r = 0.8$ has a slightly lower three-seed error than $r = 0.6$, and the choice of $r = 0.6$ is checked on ten seeds (Table 11). The ten-seed runs in Tables 1, 2, 10 and 11 compute every schedule outside the solver and pass it to the unmodified library as `rtol` and `atol` (Appendix B.6), and so do the MNIST and Jena Climate runs (Appendices C and D).

B Synthetic Dynamical System: Setup and Additional Results

B.1 Experimental Configuration

The synthetic trajectories are generated from the following nonlinear continuous dynamical system:

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{y}(t)^{\odot 3}\mathbf{A}, \tag{8}$$

where $\mathbf{y}^{\odot 3}$ denotes the element-wise cubic transformation, $\mathbf{A}$ is the coefficient matrix, and $\mathbf{y}(0)$ is the initial condition, with

$$\mathbf{A} = \begin{bmatrix} -0.1 & 2.0 \\ -2.0 & -0.1 \end{bmatrix}, \qquad \mathbf{y}(0) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

We refer to this system, taken with its initial condition and time grid from the `ode_demo` example of `torchdiffeq` (Chen, 2018), as the cubic system. Ground-truth trajectories are generated over $t \in$

$[0, 25]$ using the Dopri5 adaptive solver with 1,000 uniformly spaced time points and no observation noise. At each evaluation checkpoint, the prediction error is the mean squared error, over the 1,000 time points and both state components, between the trajectory that the model predicts from $\mathbf{y}(0)$ with the tolerances in effect at that iteration and the ground-truth trajectory; the training loss is the mean squared error on the sampled subsequences.

During training, each batch contains 20 subsequences of 20 time steps. The Neural ODE dynamics take the element-wise cubic transformation $\mathbf{y}^{\odot 3}$ as input. Table 5 summarizes the experimental configuration.

Table 5: Experimental configuration for the synthetic dynamical system experiments. Seeds and device are those of the three-seed runs; Appendix B.6 gives those of the ten-seed runs.

Component	Configuration
Dataset	Synthetic nonlinear continuous dynamical system
Gradient Computation	Backpropagation through the solver (`odeint`)
Ground-truth ODE	Eq. (8)
Time Interval	$[0, 25]$
Trajectory Points	1,000
Observation Noise	None
Ground-truth Solver	Dopri5
Ground-truth $rtol$	10^{-7}
Ground-truth $atol$	10^{-9}
Model	Neural ODE with multilayer perceptron dynamics
Dynamics Network Input	$\mathbf{y}^{\odot 3}$
Hidden Dimension	64
Activation Function	Tanh
Optimizer	Adam (Kingma & Ba, 2015)
Learning Rate	1×10^{-3} (constant)
Batch Size	20 subsequences
Subsequence Length	20 time steps
Training Iterations	5000
Adaptive Solvers	Dopri5, Tsit5, Adaptive Heun
Initial $(rtol, atol)$	$(10^{-2}, 10^{-3})$
Final $(rtol, atol)$	$(10^{-4}, 10^{-5})$
Annealing Ratio	0.6
Random Seeds	$\{123, 3407, 42\}$
Device	NVIDIA RTX GPU

B.2 Tolerance Configurations

Following the solver comparison in Appendix B.3, Dopri5 is used for the constant-tolerance baselines and CTA. Table 6 summarizes the tolerance configurations used by the compared methods.

Table 6: Tolerance configurations of the compared methods in the synthetic dynamical system experiments.

Method	Relative Tolerance ($rtol$)	Absolute Tolerance ($atol$)
Fixed-Initial	10^{-2}	10^{-3}
Fixed-Average	5.05×10^{-3}	5.05×10^{-4}
Fixed-Final	10^{-4}	10^{-5}
CTA	Scheduled by Eqs. (6)–(7)	Scheduled by Eqs. (6)–(7)

All methods use the same model architecture, optimizer, batch size, subsequence length, training iterations and random seeds, and differ only in the tolerances.

B.3 Solver Comparison over Three Seeds

Besides the metrics of Section 4.2, Tables 7–9 report Final Error Std, the standard deviation of the prediction error across seeds at iteration 5000; Table 7 also reports Peak Eval NFE, the largest seed-averaged NFE over the evaluation checkpoints, and Table 9 Final Eval NFE, the seed-averaged NFE at iteration 5000. Standard deviations are population values for the three-seed synthetic runs and sample values ($n - 1$ in the denominator) elsewhere.

We first run CTA with three adaptive solvers, Adaptive Heun (Hairer et al., 1993), Dopri5 and Tsit5 (Tsitouras, 2011), under identical settings (Appendix B.1) and with the modified solver of Appendix A; CTA uses $r = 0.6$ except in Appendix B.5. Dopri5 has the lowest Mean Final Error (0.0668) and Final Error Std (0.0033) (Table 7). Tsit5 has a 2% lower Mean Eval NFE (408.12 vs. 417.76) but a 74% higher Mean Final Error (0.1164), and Adaptive Heun has both a higher cost (Mean Eval NFE 1232.25) and a higher error (0.1680), so we use Dopri5 in the remaining experiments; Table 10 (Appendix B.7) repeats the ten-seed comparison of Table 1 with Tsit5.

Table 7: Adaptive solvers under CTA scheduling on the cubic system, three seeds. Lower is better.

Solver	Mean Final Error	Final Error Std	Mean Eval NFE	Peak Eval NFE
Adaptive Heun	0.1680	0.1588	1232.25	2316.33
Dopri5	0.0668	0.0033	417.76	596
Tsit5	0.1164	0.0139	408.12	570

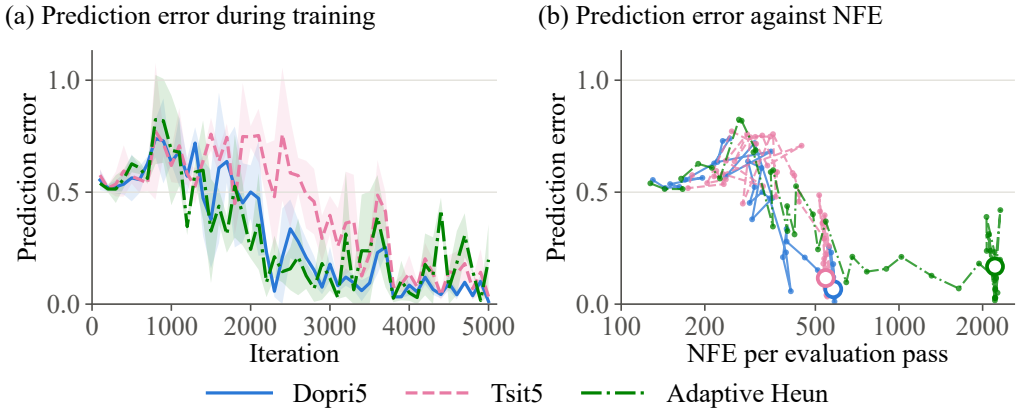

Figure 4: CTA with three adaptive solvers on the cubic system. Lines show the mean over three seeds at each evaluation checkpoint (every 100 iterations); the bands in (a) show ± 1 standard deviation, clipped at zero. In (b), checkpoints are connected in training order, and the open circles mark the mean over the last 10 checkpoints, whose error is the Mean Final Error in Table 7; their NFE (580, 545 and 2216) is likewise a mean over these 10 checkpoints, whereas Mean Eval NFE in Table 7 averages all 50 checkpoints. The NFE axis is logarithmic.

In Figure 4(a), the error of Dopri5 changes the least between consecutive checkpoints (mean absolute change 0.078, compared with 0.105 for Tsit5 and 0.126 for Adaptive Heun); the error of Tsit5 is the highest at 18 of the 21 checkpoints between iterations 1500 and 3500, and that of Adaptive Heun fluctuates the most late in training. In Figure 4(b), Adaptive Heun has about three times the Mean Eval NFE of Dopri5 (1232.25 vs. 417.76) and ends with a higher error.

B.4 Constant-Tolerance Comparison over Three Seeds

With Dopri5, we then compare CTA with Fixed-Initial, Fixed-Average and Fixed-Final (Table 6). Over three seeds (Table 8), CTA reduces the Mean Final Error of Fixed-Final, the most accurate of these baselines, by 66.0% and its Mean Eval NFE by 22.0%. Fixed-Initial and Fixed-Average have a lower Mean Eval NFE (250.80 and 324.24 vs. 417.76), but their Mean Final Error is 8.6 and 6.6

times that of CTA. CTA also has the smallest change of the error between consecutive checkpoints and the smallest spread across seeds; Section 4.3.2 revisits these comparisons with ten seeds.

Table 8 also reports $\max|\Delta\mathrm{Err}|$, the largest absolute change of the seed-averaged prediction error between two consecutive evaluation checkpoints, which are 100 iterations apart. Each evaluation pass uses the tolerances in effect at that iteration, so the errors of Fixed-Initial and Fixed-Average, and those of CTA before the end of annealing (iteration 3000), include the larger discretization error of looser tolerances. From iteration 3000 on, CTA and Fixed-Final are evaluated with the same final tolerances; over checkpoints 3000–5000, $\max|\Delta\mathrm{Err}|$ is 0.221 for CTA and 0.376 for Fixed-Final.

Table 8: Comparison of constant-tolerance baselines and CTA with Dopri5 over three seeds.

Method	Mean Final Error	Mean Eval NFE	Max $\lvert\Delta\mathrm{Err}\rvert$	Final Error Std
Fixed-Initial	0.5745	250.80	3.513	0.2159
Fixed-Average	0.4424	324.24	0.510	0.1904
Fixed-Final	0.1964	535.36	0.376	0.1700
CTA	0.0668	417.76	0.246	0.0033

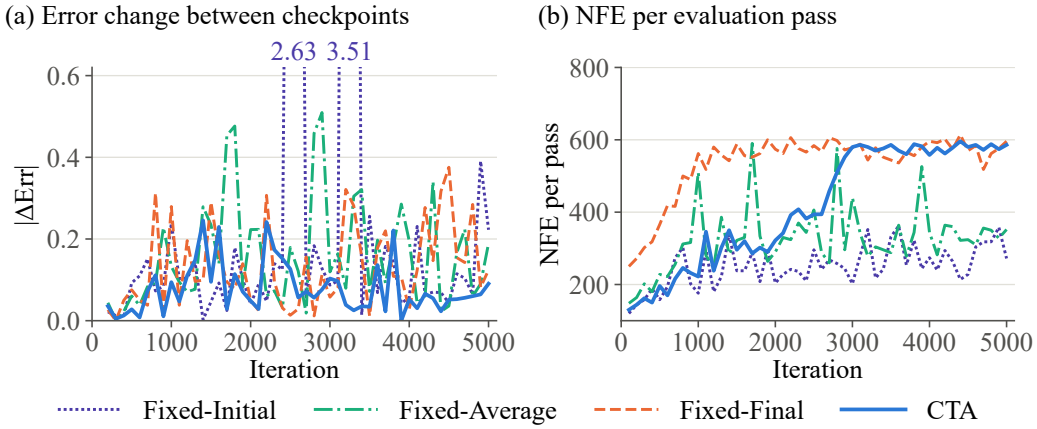

Figure 5: Constant-tolerance baselines and CTA with Dopri5, averaged over three seeds. (a) Absolute change of the prediction error between consecutive checkpoints; its maximum is $\max|\Delta\mathrm{Err}|$ in Table 8. The vertical axis is cut at 0.62; Fixed-Initial exceeds it at iterations 2500–2600 and 3200–3300, with maxima 2.63 and 3.51. (b) NFE of each evaluation pass. Line styles are the same in both panels.

Figure 5(a) shows that Fixed-Initial fluctuates far more than the other methods ($\max|\Delta\mathrm{Err}|$ of 3.513 vs. at most 0.510). In Figure 5(b), the NFE of CTA rises as the tolerances are annealed, whereas the NFE of the constant-tolerance baselines levels off after about 1,000 iterations, apart from isolated peaks of Fixed-Average with a large spread across seeds.

B.5 Annealing Ratio over Three Seeds

We evaluate $r \in \{0.3, 0.5, 0.6, 0.8, 1.0\}$ with three seeds and all other settings unchanged (Table 9). The shortest horizon ($r = 0.3$) has the highest Mean Final Error (0.1737) and Mean Eval NFE (486.80). Longer horizons lower the NFE; $r = 0.6$ and $r = 0.8$ have the lowest Mean Final Error (0.0668 and 0.0645), whereas $r = 1.0$ has the lowest Mean Eval NFE but ends with a higher error (0.1020). The $r = 0.8$ row comes from the repetition described in Appendix A; with ten seeds, $r = 0.6$ has the lowest Mean Final Error (Table 11).

Table 9: CTA with different annealing ratios r on the cubic system with Dopri5, three seeds. Lower is better.

Ratio	Mean Final Error	Final Error Std	Mean Eval NFE	Final Eval NFE
0.3	0.1737	0.0452	486.80	606
0.5	0.1340	0.0091	441.44	586
0.6	0.0668	0.0033	417.76	586
0.8	0.0645	0.0055	376.68	600
1.0	0.1020	0.0260	335.80	588

B.6 Ten-Seed Schedule Comparison

The runs in Table 1 use the seeds $\{123, 3407, 42, 1, 2, 3, 4, 5, 6, 7\}$, the model, data, optimizer and evaluation of Appendix B.1, and the unmodified `torchdiffeq` Dopri5 solver. At iteration i the tolerances are computed from the schedule and passed to the solver as `rtol` and `atol`, which the solver also uses to select its initial step; the three-seed runs compute the schedule inside a modified solver (Appendix A). With $\alpha_i = \min(1, i/N_{\text{anneal}})$ and $N_{\text{anneal}} = 0.6N_{\text{train}}$, the annealed schedules are cosine (Eq. 7), linear $tol_i = tol_{\text{final}} + (tol_{\text{initial}} - tol_{\text{final}})(1 - \alpha_i)$, geometric $tol_i = tol_{\text{initial}}^{1-\alpha_i}\, tol_{\text{final}}^{\alpha_i}$, and step, which uses tol_{initial} for $\alpha_i < 0.5$ and tol_{final} otherwise. Fixed-Geometric uses $rtol = 10^{-3}$ and $atol = 10^{-4}$. Train NFE is the number of ODE function evaluations in the forward solve of a training iteration, averaged over all $N_{\text{train}} = 5000$ iterations; gradients are obtained by backpropagating through the solver. Each run uses one CPU thread instead of the GPU listed in Table 5. Standard deviations use $n - 1$ in the denominator. The p-values come from a two-sided Wilcoxon signed-rank test (Wilcoxon, 1945) on the per-seed Mean Final Error; when CTA is lower on all ten seeds, the test gives $p = 2/2^{10} \approx 0.002$. These p-values are not adjusted for multiple comparisons; with a Holm correction (Holm, 1979) over the seven comparisons they become 0.014 for the geometric schedule and the four constant tolerances, 0.039 for the step schedule and 0.49 for the linear schedule. The bootstrap intervals in Sections 4.3.2 and 4.5 resample seeds with replacement (10,000 resamples, percentile interval; Efron & Tibshirani, 1994).

B.7 Other Solver, System, Tolerance Pairs and Annealing Ratios

All runs in this section use ten seeds and the setup of Appendix B.6. The Tsit5 runs change only the solver. The Lotka–Volterra runs replace the cubic system by $\dot{x} = 1.5x - xy$, $\dot{y} = xy - 3y$ (Lotka, 1925; Volterra, 1926) with $(x_0, y_0) = (1, 1)$ on $[0, 15]$ (1,000 time points) and feed the state to the network without the cubic map; all other settings are unchanged. The tolerance-pair runs change $(rtol_{\text{initial}}, rtol_{\text{final}})$ to $(10^{-3}, 10^{-5})$ or $(10^{-2}, 10^{-6})$, with $atol$ ten times smaller, and Fixed-Final and Fixed-Geometric use the corresponding final and geometric-midpoint values. The annealing-ratio runs change only r. Compared with Fixed-Final by the test of Appendix B.6, Fixed-Geometric ($rtol = 10^{-3}$) has a Mean Final Error 1.4, 1.5 and 2.8 times as high with Dopri5 and Tsit5 on the cubic system and with Dopri5 on Lotka–Volterra, and a higher error on 9, 8 and 9 of the 10 seeds ($p = 0.004$, 0.010 and 0.004).

C MNIST Experimental Setup

C.1 Experimental Configuration

The experiments use the standard MNIST dataset (LeCun et al., 1998). The model, the optimizer and the learning-rate schedule follow the `odenet_mnist` example of `torchdiffeq` (Chen et al., 2018; Chen, 2018), run with its adjoint option, except that the two convolutions of the ODE function have no bias and a weight decay of 10^{-4} is added; the example uses $rtol = atol = 10^{-3}$ by default. Training images are augmented using random cropping with four-pixel padding, while evaluation images are converted directly to tensors.

Table 12 summarizes the experimental configuration.

Table 10: Ten-seed comparison with Tsit5 on the cubic system (top) and with Dopri5 on the Lotka–Volterra system (bottom). Columns as in Table 1.

Schedule	Mean Final Error	Mean Eval NFE	Train NFE	CTA lower	p
Cubic system, Tsit5					
Cosine (CTA)	0.149 ± 0.039	409.3	30.49	–	–
Linear	0.161 ± 0.058	401.5	30.03	6/10	0.557
Step	0.167 ± 0.044	446.0	32.76	7/10	0.375
Geometric	0.165 ± 0.038	427.0	31.81	6/10	0.557
Fixed-Final	0.151 ± 0.038	503.1	36.11	4/10	0.922
Fixed-Geometric	0.233 ± 0.067	373.6	27.54	9/10	0.010
Fixed-Average	0.544 ± 0.117	318.1	25.81	10/10	0.002
Fixed-Initial	0.698 ± 0.109	321.8	24.48	10/10	0.002
Lotka–Volterra system, Dopri5					
Cosine (CTA)	1.474 ± 2.523	373.4	18.29	–	–
Linear	1.889 ± 2.904	364.3	17.90	7/10	0.027
Step	0.654 ± 0.890	419.0	20.14	0/10	0.002
Geometric	1.309 ± 0.827	418.6	18.87	6/10	0.322
Fixed-Final	1.080 ± 0.842	506.9	21.12	6/10	1.000
Fixed-Geometric	3.010 ± 3.474	326.7	16.26	10/10	0.002
Fixed-Average	3.090 ± 1.564	266.8	14.81	9/10	0.049
Fixed-Initial	4.925 ± 2.855	223.1	15.12	10/10	0.002

Table 11: Ten-seed comparison with other tolerance pairs (top) and other annealing ratios (bottom) on the cubic system with Dopri5. The Fixed-Geometric rows equal Fixed-Final in Table 1 because both pairs have the geometric midpoint $rtol = 10^{-4}$. In the bottom block the last two columns compare CTA with each ratio against Fixed-Final ($rtol = 10^{-4}$).

Schedule	Mean Final Error	Mean Eval NFE	Train NFE	CTA lower	p
rtol: $10^{-3} \to 10^{-5}$					
Cosine (CTA)	0.178 ± 0.038	641.8	39.26	–	–
Linear	0.169 ± 0.040	624.1	38.38	5/10	0.922
Fixed-Final	0.166 ± 0.042	861.1	50.42	4/10	0.375
Fixed-Geometric	0.173 ± 0.042	538.9	34.90	6/10	1.000
rtol: $10^{-2} \to 10^{-6}$					
Cosine (CTA)	0.0756 ± 0.0178	794.4	45.14	–	–
Linear	0.0640 ± 0.0218	774.0	43.83	3/10	0.193
Fixed-Final	0.166 ± 0.042	1321.7	74.67	10/10	0.002
Fixed-Geometric	0.173 ± 0.042	538.9	34.90	10/10	0.002
Annealing ratio r (CTA)					
$r = 0.3$	0.150 ± 0.046	489.8	31.51	6/10	0.375
$r = 0.5$	0.0936 ± 0.0236	442.5	28.78	9/10	0.004
$r = 0.6$	0.0652 ± 0.0236	425.7	27.45	10/10	0.002
$r = 0.8$	0.0779 ± 0.0266	383.3	24.73	10/10	0.002
$r = 1.0$	0.0913 ± 0.0315	337.8	21.96	9/10	0.004

All three methods use the hyperparameters in Table 12 and are trained with one script. At every training iteration it computes the tolerances from Eqs. (6)–(7), or keeps them constant, and passes them to the unmodified `torchdiffeq` Dopri5 solver; the adjoint solve of the backward pass uses the same tolerances. The NFE counter of the ODE function is read after the forward pass and after the backward pass of every iteration, and it is reset after each read and after each evaluation pass. For each epoch we record the mean forward and backward NFE over its 468 iterations and, at the end of the epoch, the test accuracy with the tolerances in effect at that time. Table 3 averages the per-epoch means over the 160 epochs and the three seeds, and the total training NFE sums the forward and backward NFE of all $74{,}880$ iterations. The final test accuracy is measured after epoch 160 with the tolerances in effect at that time, which are the final tolerances for CTA; the final models are also evaluated with $(rtol, atol) = (10^{-4}, 10^{-5})$ and $(10^{-6}, 10^{-7})$. Fixed-Geometric needs 32–38 forward evaluations per test batch at its own tolerance and 50 at the final tolerances.

Table 12: Experimental configuration for the MNIST image classification experiments.

Component	Configuration
Dataset	MNIST
Training Images	60,000
Test Images	10,000
Gradient Computation	Adjoint (`odeint_adjoint`)
Training Preprocessing	Random crop (padding = 4), ToTensor
Evaluation Preprocessing	ToTensor
Model	ODE-Net with continuous-depth architecture
ODE Function	Two-layer convolutional ODE function
Hidden Channels	64
Normalization	GroupNorm (Wu & He, 2018)
Activation Function	ReLU
Optimizer	SGD with momentum
Momentum	0.9
Weight Decay	1×10^{-4}
Initial Learning Rate	0.1
Learning-Rate Schedule	$\times 0.1$ at epochs 60, 100, and 140
Batch Size	128
Batches per Epoch	468
Training Epochs	160
Adaptive Solver	Dopri5
Initial $(rtol, atol)$	$(10^{-2}, 10^{-3})$
Final $(rtol, atol)$	$(10^{-4}, 10^{-5})$
Fixed-Final $(rtol, atol)$	$(10^{-4}, 10^{-5})$
Fixed-Geometric $(rtol, atol)$	$(10^{-3}, 10^{-4})$
Annealing Ratio	0.6
Random Seeds	{123, 3407, 42}
Solver Library	Unmodified `torchdiffeq`, tolerances set per iteration
Device	NVIDIA L40S GPU

C.2 Time per Iteration

The runs shared one GPU with each other and with our Jena Climate runs, so the wall-clock time logged during training depends on the load of the GPU. We therefore saved every model at epochs 1, 20, 40, . . . , 160 and, after training, timed training iterations from each saved model while all other runs of ours were paused. A measurement loads the saved model with the tolerances and learning rate of its epoch and runs 5 warm-up and 20 timed iterations on the same 25 training batches held in GPU memory; an iteration consists of the forward pass, the backward pass and the parameter update, followed by CUDA synchronization. Every checkpoint is measured in 5 rounds, and each round visits the 81 checkpoints in a random order, so that a period in which other jobs use the GPU affects all methods alike. We use the median of the 100 timed iterations of each checkpoint, and the time per iteration in Table 3 is the average over epochs 1–160 of the piecewise-linear interpolation between these medians. At epoch 1, an iteration takes 58–65 ms for CTA, 70–78 ms for Fixed-Geometric and 100–109 ms for Fixed-Final; from epoch 120 on, it takes 141–153 ms for CTA and Fixed-Final and 100–111 ms for Fixed-Geometric. An earlier measurement of the CTA and Fixed-Final checkpoints alone, with 100 consecutive iterations per checkpoint and processed in the same way, gave 104.2 and 129.7 ms (19.7% less for CTA); a repetition of it was slower on some checkpoints by up to 52% and not on others, which is why we use the interleaved measurement. For reference, the six CTA and Fixed-Final runs were trained at the same time, and the time per iteration logged during training, which also includes moving each batch to the GPU, averages 0.697 s for Fixed-Final and 0.599 s for CTA (14.1% less); a whole run, including the evaluation passes, took 14.6–14.9 hours for Fixed-Final and 12.5–12.9 hours for CTA. The Fixed-Geometric runs were trained later, alongside other runs, so their logged times are not comparable with these.

D JENA CLIMATE FORECASTING EXPERIMENTAL SETUP

We use the Jena Climate dataset, recorded by the Max Planck Institute for Biogeochemistry and distributed with a Keras example (Attri et al., 2020), for multivariate long-horizon temperature forecasting. The input features are seven climate variables, standardized using training-set statistics, and the data are split chronologically into training and validation sets.

Each sample uses 720 historical time steps sampled every six measurements, corresponding to 120 observations, to predict air temperature 72 time steps ahead. The feature selection, standardization, split and windowing follow the Keras example (Attri et al., 2020); unlike that example, we keep a sample only if its input and target lie in the same part and use all such samples (299,901 for training and 119,066 for validation). The model consists of an LSTM encoder (Hochreiter & Schmidhuber, 1997), a latent Neural ODE block, and a linear prediction head. The initial ODE state is $\mathbf{z}_0 = \tanh(\mathbf{h})$, and the latent dynamics are scaled by 0.5 and integrated over $t \in [0, 1]$. The NFE per iteration recorded for an epoch is the number of ODE-function evaluations in the forward and backward passes of a training iteration, averaged over the 1172 training batches of that epoch.

Table 13 summarizes the experimental configuration.

Table 13: Experimental configuration for Jena Climate forecasting.

Component	Configuration
Dataset	Jena Climate
Gradient Computation	Adjoint (`odeint_adjoint`)
Input Features	7 climate variables
Target	Air temperature
Train / Validation Split	71.5% / 28.5% (chronological)
Normalization	Z-score using training statistics
Input Window	720 time steps, sampled every 6 measurements
Effective Sequence Length	120
Prediction Horizon	72 time steps
Model	LSTM–Neural ODE
LSTM	Single-layer, hidden size 32
ODE Function	Two-layer MLP with Tanh
ODE Initial State	$\mathbf{z}_0 = \tanh(\mathbf{h})$
ODE Dynamics	$0.5 f_\theta(\mathbf{z})$
ODE Interval	$[0, 1]$
Prediction Head	Linear
Optimizer	Adam (Kingma & Ba, 2015)
Learning Rate	10^{-3}
Loss	MSE
Batch Size	256
Maximum Epochs	50
Gradient Clipping	1.0
Data Shuffling	None
Early Stopping	Patience 5, minimum improvement 10^{-5}
Solver	Dopri5
Maximum Solver Steps	200
Initial $(rtol, atol)$	$(10^{-2}, 10^{-3})$
Final $(rtol, atol)$	$(10^{-4}, 10^{-5})$
Fixed-Final $(rtol, atol)$	$(10^{-4}, 10^{-5})$
Fixed-Geometric $(rtol, atol)$	$(10^{-3}, 10^{-4})$
Annealing Ratio	0.6
Random Seeds	$\{123, 3407, 42, 1, 2, \ldots, 7\}$
Solver Library	Unmodified `torchdiffeq`, tolerances set per iteration
Device	NVIDIA L40S GPU

CTA applies the cosine tolerance schedule defined in Eqs. (6)–(7) with N_{train} equal to 50 epochs of 1172 iterations, whereas Fixed-Final and Fixed-Geometric use the constant tolerances $(rtol, atol) = (10^{-4}, 10^{-5})$ and $(10^{-3}, 10^{-4})$. All other optimization settings are identical across methods, and all methods use the same training script, which computes the tolerances at every iteration, passes them to the unmodified `torchdiffeq` Dopri5 solver, and uses the same tolerances in the adjoint solve. The forward and backward NFE of every iteration are counted separately. The validation MSE of each epoch is computed with the tolerances in effect at the end of that epoch, and early stopping uses this value. The final version of the script also keeps the best checkpoint and, after training, evaluates it again on the validation set with $(rtol, atol) = (10^{-4}, 10^{-5})$ and $(10^{-6}, 10^{-7})$.

The first runs of seeds 123, 3407, 42 and 2 were made with an earlier version that trains identically but lacks this evaluation, so we trained CTA a second time on these four seeds with the final script. Training is deterministic: the second runs reproduced the training loss, validation MSE and NFE of every epoch exactly, and they provide the re-evaluated validation MSE of CTA for these seeds. The Fixed-Geometric runs were made with the final script. Fixed-Final is trained and evaluated with the final tolerances, so its best validation MSE is already computed with them. Table 14 lists the results of each seed.

Table 14: Jena Climate results of each seed. NFE counts forward and backward evaluations and is taken up to the best checkpoint or over the whole run.

		Best	Best	NFE per iteration		Total NFE (10^6)	
Seed	Method	validation MSE	epoch	up to best	whole run	up to best	whole run
123	Fixed-Final	0.1253	26	43.0	46.3	1.31	1.68
	Fixed-Geometric	0.1216	27	32.1	34.1	1.02	1.28
	CTA	0.1216	29	32.5	38.0	1.10	1.51
3407	Fixed-Final	0.1120	47	74.6	76.8	4.11	4.50
	Fixed-Geometric	0.1125	49	53.4	53.8	3.06	3.15
	CTA	0.1121	45	64.3	68.8	3.39	4.03
42	Fixed-Final	0.1288	19	42.0	43.9	0.94	1.24
	Fixed-Geometric	0.1211	28	32.7	34.8	1.07	1.35
	CTA	0.1301	15	28.0	28.7	0.49	0.67
1	Fixed-Final	0.1169	25	47.7	52.1	1.40	1.83
	Fixed-Geometric	0.1179	25	33.5	36.0	0.98	1.26
	CTA	0.1184	25	31.0	36.5	0.91	1.28
2	Fixed-Final	0.1138	45	69.3	74.0	3.66	4.34
	Fixed-Geometric	0.1174	31	39.2	42.3	1.42	1.79
	CTA	0.1191	28	41.4	49.3	1.36	1.91
3	Fixed-Final	0.1170	28	59.1	65.4	1.94	2.53
	Fixed-Geometric	0.1178	32	46.2	50.1	1.73	2.17
	CTA	0.1148	27	40.2	49.0	1.27	1.84
4	Fixed-Final	0.1191	38	59.8	65.4	2.67	3.29
	Fixed-Geometric	0.1153	34	39.6	43.5	1.58	1.99
	CTA	0.1170	42	57.1	62.7	2.81	3.45
5	Fixed-Final	0.1191	25	51.9	58.6	1.52	2.06
	Fixed-Geometric	0.1183	26	37.7	42.0	1.15	1.53
	CTA	0.1196	33	48.1	55.5	1.86	2.47
6	Fixed-Final	0.1158	36	69.4	74.3	2.93	3.57
	Fixed-Geometric	0.1149	33	48.0	51.6	1.85	2.30
	CTA	0.1172	31	52.7	60.9	1.92	2.57
7	Fixed-Final	0.1192	40	61.2	66.0	2.87	3.48
	Fixed-Geometric	0.1200	37	41.7	45.7	1.81	2.25
	CTA	0.1151	41	56.7	62.5	2.73	3.37