Improving Neural ODE Training with Temporal Adaptive Batch Normalization

Su Zheng[†1], Zhengqi Gao[‡1], Fan-Keng Sun[‡], Duane S. Boning[‡], Bei Yu[†], Martin Wong[†]

[†] The Chinese University of Hong Kong
[‡] Massachusetts Institute of Technology

Dec. 12, 2024

[1]The first two authors contribute equally.

NEURAL INFORMATION PROCESSING SYSTEMS

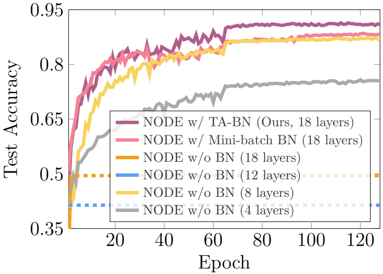
Neural ODE

It models the continuous dynamics of hidden states with a learnable ODE system:

$$\frac{d\mathbf{h}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{h}(t), t), \quad \mathbf{h}(T) = \mathbf{h}(0) + \int_0^T \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{h}(t), t)dt. \tag{1}$$

Problem in Neural ODE

Without special treatment, merely stacking additional layers in the temporal derivatives does not necessarily enhance Neural ODE performance.



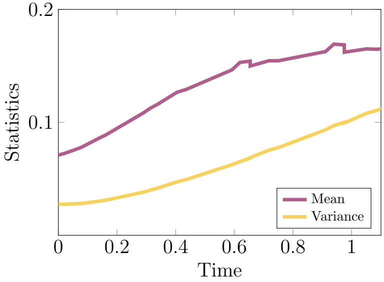
Batch Normalization

BN performs a re-centering and a re-scaling operation on the given input by subtracting the mean and dividing by the standard deviation:

$$\mathrm{BN}(x_i) = \mathrm{BN}_{\gamma,\alpha}(x_i) = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}\gamma + \alpha. \tag{2}$$

Problem in Neural ODE + Batch Normalization

BN uses a single pair of μ and σ^2 for normalization, while the output statistics from Neural ODE are time-dependent. Thus, BN can not correctly normalize Neural ODE's output.



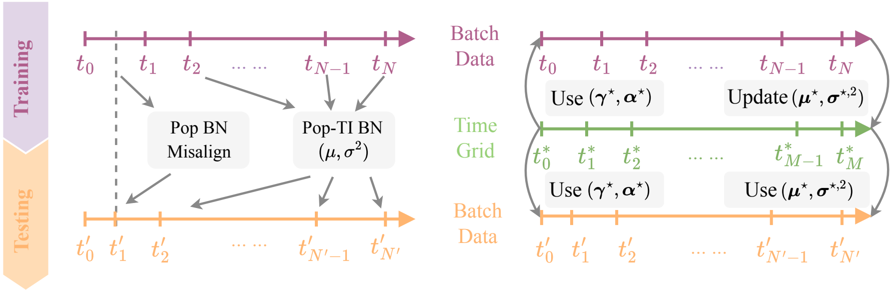
NEURAL INFORMATION PROCESSING SYSTEMS

What if we use μ_t and σ_t^2 for each time t?

Due to the adoption of adaptive ODE solver, the population statistics associated with the time point $t'_j \in \mathcal{T}'$, required by the temporal discretization during inference, might not be available if the time value t'_j is never encountered during training.

Temporal Adaptive Batch Normalization

We associate the time grid $t^\star_m$ with population mean $\mu^\star_m$ and population variance $\sigma^{\star,2}_m$, as well as learnable parameters $\gamma^\star_m$ and $\alpha^\star_m$ for every $m = 0, 1, 2, \cdots, M$. Given any time t, get $(\mu_t, \sigma_t^2, \gamma_t, \alpha_t)$ by interpolating $(\mu^\star_m, \sigma^{\star,2}_m, \gamma^\star_m, \alpha^\star_m)$ over time.

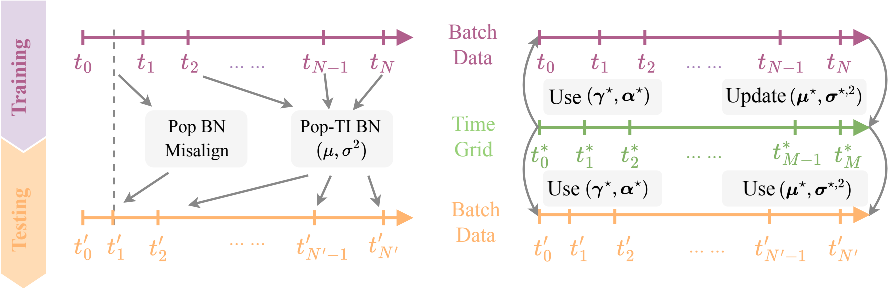


Temporal Adaptive Batch Normalization

$$\mathrm{TABN}_{\boldsymbol{\gamma}^\star,\boldsymbol{\alpha}^\star}(x_{i,j}) = \frac{x_{i,j}-\mu_j}{\sqrt{\sigma_j^2+\epsilon}}\gamma_j + \alpha_j, \text{ where } x_{i,j} = w \cdot h_i(t_j) + b, \tag{3}$$

$$\mu_j = G(t_j, \boldsymbol{\mu}^\star, \mathcal{T}^\star),\ \sigma_j^2 = G(t_j, \boldsymbol{\sigma}^{\star,2}, \mathcal{T}^\star),\ \gamma_j = G(t_j, \boldsymbol{\gamma}^\star, \mathcal{T}^\star),\ \alpha_j = G(t_j, \boldsymbol{\alpha}^\star, \mathcal{T}^\star), \tag{4}$$

$$G(t, \mathbf{a}, \mathcal{T}) = \frac{t_{l+1}-t}{t_{l+1}-t_l}a_l + \frac{t-t_l}{t_{l+1}-t_l}a_{l+1}. \tag{5}$$



Scalable Neural ODE

When the layer count exceeds 10, vanilla Neural ODEs fails due to numerical instability. In contrast, the incorporation of TA-BN enables deeper layers within Neural ODE as the learnable derivatives, scaling up the model size and enhancing accuracy.

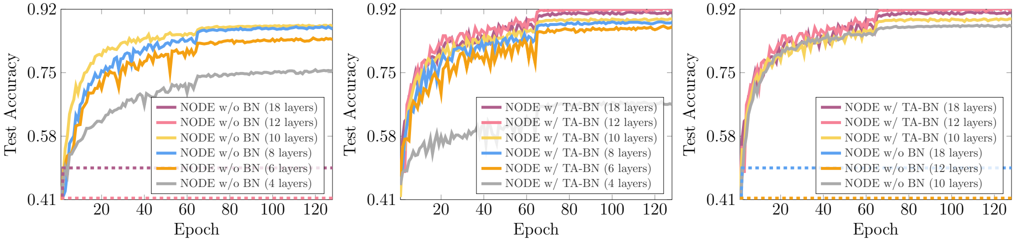


Figure: CIFAR-10 accuracies with increasing sizes of the backbones for learnable derivatives. These figures illustrate the scaling up of Neural ODEs without BN (left) and Neural ODEs with TA-BN (middle). We also compare the accuracies of these two settings in one figure (right).

NEURAL INFORMATION PROCESSING SYSTEMS

Efficient Neural ODE

Neural ODEs with TA-BN achieves better accuracies and parameter efficiency than existing Neural ODEs.

Model	MNIST		CIFAR10		SVHN		CIFAR100		Tiny-Imagenet	
	Accuracy	#Params	Accuracy	#Params	Accuracy	#Params	Accuracy	#Params	Accuracy	#Params
IL-NODE	0.991	21k	0.734	36k	-	-	-	-	-	-
2nd-Ord[2]	**0.992**	20k	0.728	35k	-	-	-	-	-	-
HBNODE	0.983	86k	0.622	173k	-	-	-	-	-	-
GHBNODE[3]	0.987	85k	0.605	173k	-	-	-	-	-	-
Aug-NODE[4]	0.982	84k	0.606	172k	0.835	172k	N/A	N/A	N/A	366k
STEER[5]	0.986	84k	0.621	172k	0.841	172k	N/A	N/A	N/A	N/A
w/o BN	0.989 ± 0.001	37k	0.517 ± 0.049	2.2M	0.096 ± 0.025	2.2M	0.246 ± 0.084	2.2M	-	2.2M
w/ Pop-TI BN	0.973 ± 0.011	37k	0.548 ± 0.087	2.2M	0.241 ± 0.123	2.2M	0.251 ± 0.112	2.2M	0.044 ± 0.007	2.2M
w/ Mini-batch BN	0.962 ± 0.013	37k	0.822 ± 0.095	2.2M	0.906 ± 0.031	2.2M	0.492 ± 0.176	2.2M	0.200 ± 0.006	2.2M
w/ TA-BN	0.988 ± 0.001	37k	0.748 ± 0.059	70k	0.953 ± 0.002	220k	0.576 ± 0.016	220k	0.436 ± 0.013	220k
(ours)	0.988 ± 0.001	220k	$\mathbf{0.910\pm0.010}$	2.2M	$\mathbf{0.958\pm0.004}$	2.2M	$\mathbf{0.664\pm0.025}$	2.2M	$\mathbf{0.512\pm0.008}$	2.2M

[2] Stefano Massaroli et al. (2020). “Dissecting neural odes”. In: *Advances in Neural Information Processing Systems* 33, pp. 3952–3963.

[3] Hedi Xia et al. (2021). “Heavy ball neural ordinary differential equations”. In: *Advances in Neural Information Processing Systems* 34, pp. 18646–18659.

[4] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh (2019). “Augmented neural odes”. In: *Advances in neural information processing systems* 32.

[5] Arnab Ghosh et al. (2020). “STEER: Simple temporal regularization for neural ode”. In: *Advances in Neural Information Processing Systems* 33, pp. 14831–14843.

THANK YOU!