

Table 10: The architecture of the network f of ODE-Net for MNIST image classification

Layer	Design	Input Dim.	Output Dim.
1	ReLU(GroupNorm)	$6 \times 6 \times 64$	$6 \times 6 \times 64$
2	ReLU(GroupNorm(Conv2d(filter size 3x3, stride 1, padding 1)))	$6 \times 6 \times 64$	$6 \times 6 \times 64$
3	GroupNorm(Conv2d(filter size 3x3, stride 1, padding 1))	$6 \times 6 \times 64$	$6 \times 6 \times 64$

A THE STEP SIZE CONTROL IN DOPRI

The formula to calculate the step size at i -th step of DOPRI is as follows:

$$s_i = s_{i-1} \cdot \left(\frac{1}{err_i} \right)^{\frac{1}{q+1}}, \quad (6)$$

where s_{i-1} is a step size in the previous step, $q > 0$ is a hyperparameter, and err_i is the estimated error at i -th step calculated by the difference between the fifth-order Runge–Kutta method and the fourth-order Runge–Kutta method at the step.

B NFE IN INTEGRATORS

As noted earlier, different integrators have different NFE values: DOPRI has an NFE of 6 in a step, RK4 has an NFE of 4 in a step, and the Euler method has an NFE of 1 in a step. For instance, the Euler method can be written as follows for a step:

$$\mathbf{h}(t_i) = \mathbf{h}(t_{i-1}) + s \cdot f(\mathbf{h}(t_{i-1}), t_{i-1}; \boldsymbol{\theta}), \quad (7)$$

where s is a fixed step size of the Euler method. Note that the function f is evaluated once in a step, which is different in other advanced integrators. RK4 and DOPRI evaluate f multiple times in a step to calculate reliable gradients.

To illustrate a more concrete example, suppose that we want to calculate $\mathbf{h}(1)$ from $\mathbf{h}(0)$. The best case happens when the step size is 1 for both of DOPRI and the Euler method, in which case DOPRI requires a total NFE of 6 and the Euler method requires a total NFE of 1. Therefore, DOPRI has a limitation in its design when it comes to decreasing the total NFE value.

C NEURAL ODE FOR MNIST IMAGE CLASSIFICATION

We use ODE-Net for this experiment which was also used in (Chen et al., 2018). Table 10 shows the detailed architecture of the network $f(\mathbf{h}(t), t; \boldsymbol{\theta})$ of ODE-Net. The list of hyperparameters that we had considered to train $\boldsymbol{\theta}$ is as follows:

1. The learning rate is $\{0.1, 0.01, 0.001\}$,
2. The size of batch is 128 for training,
3. The method to extract $\mathbf{h}(0)$, i.e., feature extractor or downsampling method, is convolution,
4. The dimensionality of ODE state, i.e., $\dim(\mathbf{h}(t))$, is $6 \times 6 \times 64$.

D NEURAL ODE FOR PHYSIONET MORTALITY RATE FORECASTING

We use Latent-ODE which consists of an RNN encoder and an ODE decoder (Rubanova et al., 2019). In Table 11, we summarize the architecture of the network f of the decoder. The list of hyperparameters that we had considered to train $\boldsymbol{\theta}$ is as follows:

1. The learning rate is 0.01 with a decay factor of 0.999 every epoch,
2. The size of batch is 3×50 ,
3. The dimensionality of RNN hidden vector is 40,
4. The dimensionality of ODE state, i.e., $\dim(\mathbf{h}(t))$, is 20.

Table 11: The architecture of the network f of the ODE decoder in Latent-ODE for PhysioNet mortality rate forecasting

Layer	Design	Input Dim.	Output Dim.
1	Tanh(FC)	20×1	50×1
2	Tanh(FC)	50×1	50×1
3	Tanh(FC)	50×1	50×1
4	Tanh(FC)	50×1	50×1
5	FC	50×1	20×1

Table 12: The architecture of the network f of ODE-Net in Continuous Normalizing Flow

Layer	Design	Input Dim.	Output Dim.
1	Tanh(FC)	32×2	32×1
2	Tanh(FC)	32×1	32×1
3	FC	32×1	488×1
4	FC	488×1	32×2

E NEURAL ODE FOR CONTINUOUS NORMALIZING FLOW

We use the ODE network in (Chen et al., 2018) for continuous normalizing flows. Table 12 shows the detailed architecture of the network $f(\mathbf{h}(t), t; \theta)$ for this experiment. The list of hyperparameters that we had considered to train θ is as follows:

1. The learning rate is $\{0.01, 0.001, 0.0001, 0.00001\}$
2. The dimensionality of ODE state, i.e., $\dim(\mathbf{h}(t))$, is 32×2 .

F ADDITIONAL EXPERIMENTAL RESULTS

We introduce additional experimental results in this section. First, we test other integrators as well. In particular, we are interested in low-order adaptive integrators, e.g., Bosh3. Second, we introduce the results of when we randomly choose integrators either i) following a uniform distribution or ii) following the distributions reported in Tables 3, 6, and 9. In Tables 13 to 15, we summarize the additional results. In the table, zero standard deviations mean small standard deviations that can be neglected. For MNIST, we could observe many zero standard deviations because the MNIST classification task is a rather easier task than others. For MNIST, all integrators produce similar accuracy. However, the Euler method has a relatively low accuracy. In other datasets, however, we could observe meaningful accuracy differences among integrators.

G FORWARD AND BACKWARD NFE VS. EPOCHS

In Figure 4, we report the time in seconds needed for the forward and backward-pass processing of the entire batch of MNIST. When there are no regularization terms, forward/backward-pass time increases gradually and saturates around an epoch of 25. When we use our regularization, however, they drastically decrease around an epoch of 30. After being quickly stabilized, they maintain low values, which shows the efficacy of our regularization method.

Table 13: MNIST classification results (1 NFE \approx 0.0007 seconds for a test batch of 100 images)

Model	Accuracy	NFE
No reg.	0.99602 ± 0.0002	26 ± 0
Kinetic energy reg.	0.99653 ± 0.0006	20 ± 0
L^1 reg.	0.99564 ± 0.0004	20 ± 0
L^2 reg.	0.99614 ± 0.0004	20 ± 0
Our reg.	0.99640 ± 0.0004	14 ± 0
No reg. & DISE	0.99584 ± 0.0002	11.36 ± 0
Our reg. & DISE	0.99626 ± 0.0004	5.79 ± 0
Our reg. (Euler)	0.99316 ± 0.0016	1 ± 0
Our reg. (RK4)	0.99632 ± 0.0005	4 ± 0
Our reg. (Bosh3)	0.99640 ± 0.0004	14 ± 0
Our reg. & Random	0.99602 ± 0.0002	6.62 ± 0.733
Our reg. & Random (with the DISE's distribution)	0.99558 ± 0.0002	5.79 ± 0
No reg. & Random	0.99434 ± 0.0006	10.844 ± 1.439
No reg. & Random (with the DISE's distribution)	0.99440 ± 0.0007	11.36 ± 0

Table 14: PhysioNet mortality prediction results (1 NFE \approx 0.013 seconds for a test batch of 60 records)

Model	AUC	NFE
No reg.	0.7190 ± 0.027	74 ± 0
Kinetic energy reg.	0.7581 ± 0.0401	63.5 ± 7.55
L^1 reg.	0.7630 ± 0.0285	68 ± 20.8
L^2 reg.	0.7450 ± 0.0304	59 ± 3.4641
Our reg.	0.7509 ± 0.0254	39.71 ± 2.93
No reg. & DISE	0.7513 ± 0.002	57.57 ± 1.378
Our reg. & DISE	0.7604 ± 0.004	34.1 ± 0.369
Our reg. (Euler)	0.7589 ± 0.0008	10 ± 0
Our reg. (RK4)	0.7584 ± 0.0017	40 ± 0
Our reg. (Bosh3)	0.7584 ± 0.0011	29 ± 0
No reg. & Random	0.7494 ± 0.0014	57.68 ± 6.234
No reg. & Random (with the DISE's distribution)	0.7491 ± 0.0008	56.875 ± 0
Our reg. & Random	0.7582 ± 0.002	34.5 ± 0.904
Our reg. & Random (with the DISE's distribution)	0.7578 ± 0.001	34.5 ± 0

Table 15: Continuous Normalizing Flow results (1 NFE \approx 0.008 seconds for a test batch of 32 samples)

Model	NLP	NFE
No reg.	0.89113 ± 0.0555	2297 ± 51.718
Kinetic energy reg.	0.89148 ± 0.0555	2286 ± 49.249
L^1 reg.	0.89011 ± 0.0556	2259 ± 23.520
L^2 reg.	0.87607 ± 0.0493	2904 ± 23.597
Our reg.	0.88418 ± 0.0536	2166 ± 85.163
No reg. & DISE	0.88837 ± 0.0318	2104 ± 94.728
Our reg. & DISE	0.87420 ± 0.0262	1984 ± 152.450
Our reg. (Euler)	0.93941 ± 0.0550	640 ± 0
Our reg. (RK4)	0.88418 ± 0.0536	2560 ± 0
Our reg. (Bosh3)	0.88425 ± 0.0536	5309.5 ± 45.593
Our reg. & Random	0.84360 ± 0.0009	1887.6 ± 135.554
Our reg. & Random (with the DISE's distribution)	0.83344 ± 0.0067	1888.8 ± 24.519
No reg. & Random	0.85292 ± 0.0083	1975.2 ± 154.277
No reg. & Random (with the DISE's distribution)	0.84470 ± 0.0054	2108.4 ± 5.367

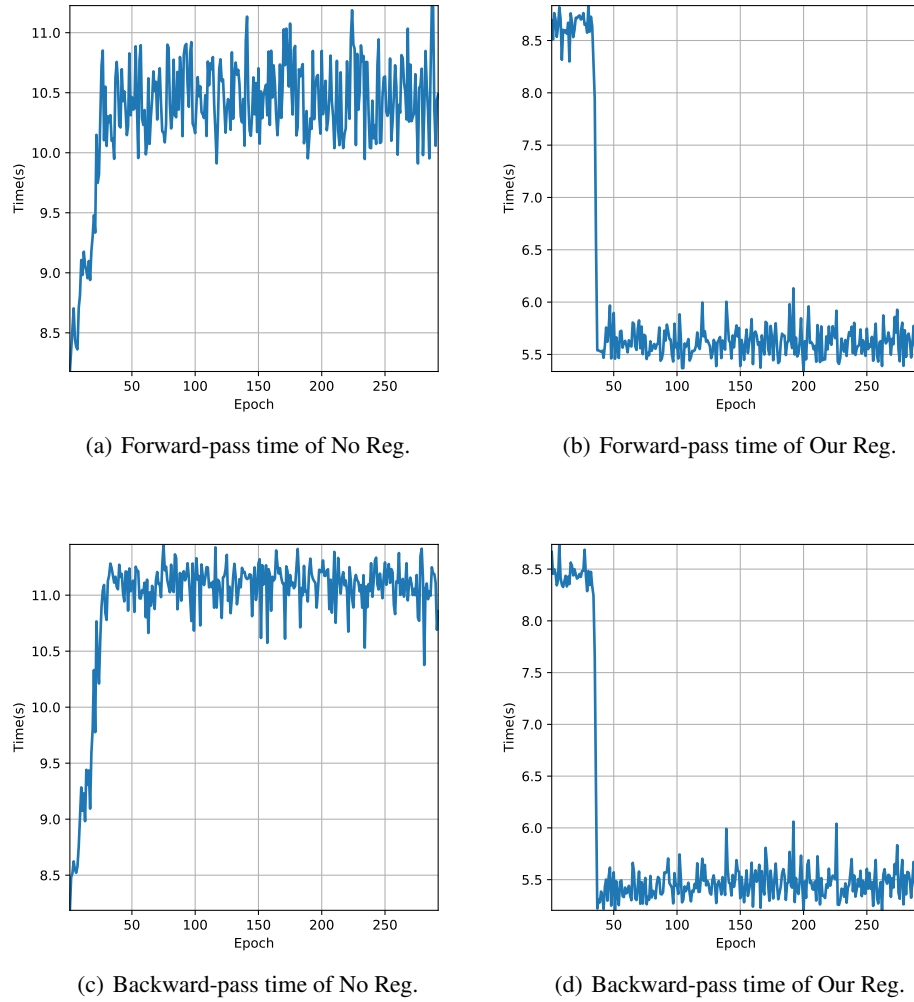


Figure 4: The forward/backward-pass time (in seconds) vs. epochs. We report the time needed to process the entire batch (not a mini-batch).