Jacobian Norm Regularisation and Conditioning in Neural ODEs*

Shane Josias[1,2] and Willie Brink[1]

[1]Applied Mathematics, Stellenbosch University
[2]School for Data Science and Computational Thinking, Stellenbosch University
{josias,wbrink}@sun.ac.za

Abstract. A recent line of work regularises the dynamics of neural ordinary differential equations (neural ODEs), in order to reduce the number of function evaluations needed by a numerical ODE solver during training. For instance, in the context of continuous normalising flows, the Frobenius norm of Jacobian matrices are regularised under the hypothesis that complex dynamics relate to an ill-conditioned ODE and require more function evaluations from the solver. Regularising the Jacobian norm also relates to sensitivity analysis in the broader neural network literature, where it is believed that regularised models should be more robust to random and adversarial perturbations in their input. We investigate the conditioning of neural ODEs under different Jacobian regularisation strategies, in a binary classification setting. Regularising the Jacobian norm indeed reduces the number of function evaluations required, but at a cost to generalisation. Moreover, naively regularising the Jacobian norm can make the ODE system more ill-conditioned, contrary to what is believed in the literature. As an alternative, we regularise the condition number of the Jacobian and observe a lower number of function evaluations without a significant decrease in generalisation performance. We also find that Jacobian regularisation does not guarantee adversarial robustness, but it can lead to larger margin classifiers.

Keywords: Neural ODEs · Regularisation · Sensitivity.

1 Introduction

Neural ordinary differential equations (neural ODEs) [4] are a class of implicit deep learning layers, or continuous-depth models, in which the solution procedure (i.e. the forward pass) is separated from the definition of the layers (the neural network). This separation allows for flexibility in controlling the error of the solution procedure to trade-off against accuracy, and the use of adaptive solvers and correctors for ODE systems that are difficult to solve. Through the use of the adjoint method for gradient calculations, neural ODEs become memory efficient with a cost that grows constant in the number of layers, but sacrifices

* This work is based on research supported by the National Research Foundation of South Africa (grant number 138341).

on computational complexity since the forward and reverse trajectories for a given point must be re-evaluated [4]. Mitigating the computational complexity inherent in neural ODEs can make them more tractable for larger datasets and a wider variety of modelling problems [9].

Neural ODEs define a continuous transformation on the data where the forward pass through the ODE is represented by an integral. The computational complexity of a neural ODE is represented by the number of function evaluations (NFE) required by the ODE solver to determine the solution trajectory [7, 9, 15]. For a mini-batch of data considered during training, NFE describes the number of times points along a solution trajectory are passed through the neural network. During training, the dynamics of the data transformation process become increasingly complex such that the solver must take finer steps to determine a solution [7]. Figure 1 shows how the NFE increases during the training of an unregularised neural ODE on a dataset (details of this experiment will follow in Section 3). It is believed that the complexity of the dynamics is related to the conditioning of the ODE system [7, 9]. Simpler dynamics on the other hand, and a lower NFE, encourage faster convergence for neural ODEs and make them more practically feasible [9, 15]. However, it is not clear whether generalisation and robustness of the solution are maintained under simplified dynamics, or whether there is a trade-off.

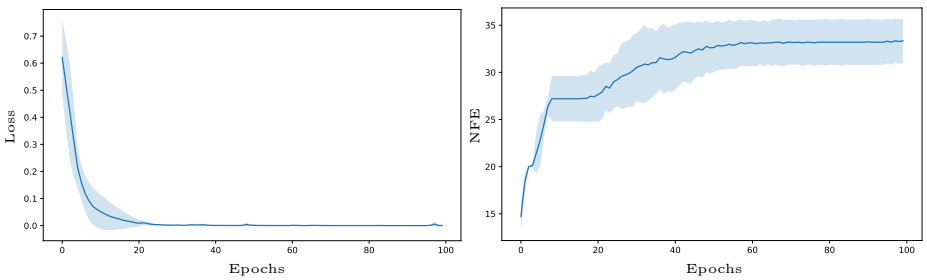

Fig. 1. As the loss is minimised during the training of an unregularised neural ODE on a 2-dimensional dataset (left), the number of function evaluations (NFE) required by the ODE solver increases (right). Here the solver requires on average about 34 function evaluations per batch. For continuous normalising flows on CIFAR-10, the NFE can go up to about 1400 [9].

Following Finlay et al. [9], we are interested in encouraging simple dynamics through regularising the Jacobian of a neural ODE, in classification problems instead of the normalising flows context. The Jacobian is evaluated at the input data, serving as initial conditions for the neural ODE. The focus on Jacobian regularisation provides a link to sensitivity analysis found in the conventional neural network literature [11, 13, 14, 23, 26] where both Frobenius and spectral norm regularisation have been employed. Conventional neural networks define an explicit data transformation, and the Jacobian represents the change in output

with respect to changes in the input data. The Jacobian of a neural ODE, on the other hand, gives an indication of how the vector field describing the data flow changes with respect to changes in initial conditions. In this paper we borrow sensitivity analysis tools from the neural network literature, such as perturbation analysis and distance to decision boundaries, in order to evaluate Jacobian regularisation strategies in the training of neural ODEs. Our contributions can be summarised as follows.

1. We empirically investigate the effect of simplified dynamics on generalisation, adversarial vulnerability, and conditioning through Jacobian regularisation, with a view towards making neural ODE training more practical without sacrificing performance.
2. We show that regularising the Jacobian norm can make the ODE system more ill-conditioned, contrary to what is suggested in the literature [9], and while it reduces the NFE, it does so at the cost of test accuracy.
3. We show that regularising the condition number of the Jacobian reduces the NFE without sacrificing test accuracy.
4. Finally, we show that Jacobian norm regularisation can increase distance to the decision boundary for correctly classified data points, but does not guarantee robustness against adversarial perturbations.

Section 2 provides background on neural ODEs and the regularisation methods we investigate. Section 3 describes experiment methodology, and Section 4 discusses results on a running example. Section 5 extends the experiments to additional datasets for verification. Section 6 relates existing literature to this study, and Section 7 provides conclusions and ideas for future work.

2 Background and definitions

Neural ODEs specify the dynamics of a continuous data transformation process by parameterising the ODE vector field with a neural network. Suppose that f_θ is a neural network with parameter set θ, and $\boldsymbol{h}(t)$ the transformed data at time t, for $t \in [t_0, t_1]$. The neural ODE is then defined as

$$\frac{d\boldsymbol{h}(t)}{dt} = f_\theta(\boldsymbol{h}(t), t), \tag{1}$$

with an initial condition $\boldsymbol{h}(t_0)$ represented by the input data. We are interested in a solution $\boldsymbol{h}(t_1)$ at some time t_1, with (t_0, t_1) normally chosen as $(0, 1)$ in the literature [4, 7, 12]. The solution itself is unique provided that the neural network is Lipschitz continuous, due to Picard's existence theorem [5]. Linear and convolutional layers, and nonlinearities such as ReLU and tanh satisfy this property. The forward pass of the data is determined by integration:

$$\boldsymbol{h}(t_1) = \boldsymbol{h}(t_0) + \int_{t_0}^{t_1} f_\theta(\boldsymbol{h}(t), t)\, dt. \tag{2}$$

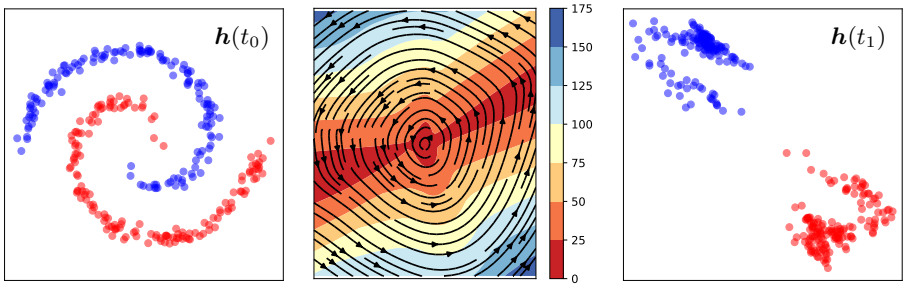

Fig. 2. The vector field defined by a neural ODE and the effect on a 2-dimensional dataset. The network in this neural ODE was trained to linearly separate the two classes shown on the left, leading to dynamics described by an unstable spiral fixed point in the vector field (middle). The colours in the vector field plot represent vector magnitude.

The initial condition $\boldsymbol{h}(t_0)$ and solution at t_1, $\boldsymbol{h}(t_1)$, define the input and output of the neural ODE, both of which having the same dimension as the input data. The network f_θ itself defines the gradient (vector field) of the solution trajectory at any given point in space. A loss function $\mathcal{L}$ to be optimised accepts the solution determined by a standard numerical ODE solver:

$$\mathcal{L}\left(\boldsymbol{h}(t_1)\right) = \mathcal{L}\left(\boldsymbol{h}(t_0) + \int_{t_0}^{t_1} f_\theta(\boldsymbol{h}(t), t)\, dt\right) = \mathcal{L}\left(\text{ODESolve}(\boldsymbol{h}(t_0), f_\theta, t_0, t_1)\right). \tag{3}$$

While it is possible to backpropagate through the operations of the ODE solver, such a strategy incurs a high memory cost and introduces additional numerical error [4] as the computational graph can become quite large. Instead, it is preferred to compute $\frac{\partial \mathcal{L}}{\partial \theta}$ by the adjoint sensitivity method, which considers $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}(t)}$, $\frac{\partial f_\theta}{\partial \theta}$ and $\frac{\partial f_\theta}{\partial \boldsymbol{h}(t)}$. These quantities can be computed efficiently and the integrals for constructing the forward and reverse trajectories can be found through standard ODE solvers. This approach scales more efficiently, has a low memory cost and explicitly controls numerical error [4].

Figure 2 provides an example of how a neural ODE trained jointly with a linear classifier at the end transforms input data at time t_0 to become linearly separable at time t_1.

NFE represents the number of times points along a solution trajectory are passed through the network f_θ that defines a neural ODE. An ODE solver that makes finer discretisations will require a higher NFE, and indicates more complex dynamics. In our experiments, NFEs will be averaged across the batches of an epoch, with batch size kept constant across regularisation methods. Investigating the impact of batch size on NFE is left for future work.

Jacobian regularisation is motivated by the fact that during training, NFE and the Frobenius norm of Jacobian matrices both increase [9]. The Jacobian $\boldsymbol{J} \in \mathbb{R}^{d \times d}$ we consider is that of the neural network defining the vector field, with respect to the input data:

$$\boldsymbol{J} = \nabla_{\boldsymbol{h}(t_0)} f_\theta(\boldsymbol{h}(t), t). \tag{4}$$

We will experiment with regularising the Frobenius norm $\|\boldsymbol{J}\|_F$, the spectral norm $\|\boldsymbol{J}\|_2$, and the condition number $\kappa(\boldsymbol{J})$. These are defined as

$$\|\boldsymbol{J}\|_F = \sqrt{\sum_{i=1}^{d}\sum_{j=1}^{d} |\boldsymbol{J}_{i,j}|^2}, \quad \|\boldsymbol{J}\|_2 = \sigma_{\max}(\boldsymbol{J}), \quad \kappa(\boldsymbol{J}) = \frac{\sigma_{\max}(\boldsymbol{J})}{\sigma_{\min}(\boldsymbol{J})}, \tag{5}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ refer to the largest and smallest singular values of a matrix. Regularising $\kappa(\boldsymbol{J})$ serves as a preliminary investigation into the claim that NFE increases due to ill-conditioning of the ODE [7, 9].

For the purposes of regularisation, we will evaluate the Jacobian at the initial conditions (the input data). Thus, we make an assumption that regularising dynamics at the initial conditions can lead to regularised dynamics across the entire trajectory. Regularising $\|\boldsymbol{J}\|_2$ controls the rate at which the input space is stretched along the first principal axis. Regularising $\|\boldsymbol{J}\|_F$ effectively scales the Jacobian matrix, and should induce the same effect as controlling $\|\boldsymbol{J}\|_2$, as is also apparent from the fact that $\|\boldsymbol{J}\|_2 \leq \|\boldsymbol{J}\|_F$. Exploring means of explicitly regularising dynamics across the entire solution trajectory is left for future work.

3 Methodology

We consider a supervised classification setting, where given a neural ODE defined by f_θ we aim to find parameters θ such that a linear classifier g_ϕ built on the solution of the ODE can associate an output $\boldsymbol{h}(t_1)$ with a one-hot encoded label y. Given a labelled training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with $\boldsymbol{x}_i \in \mathbb{R}^d$, and letting $\boldsymbol{h}_i(t_0) = \boldsymbol{x}_i$, we construct the classifier g_ϕ to accept solutions $\boldsymbol{h}_i(t_1)$ of a neural ODE system and to output confidence scores over labels. For a sample $(\boldsymbol{x}, y) \in \mathcal{D}$, we denote the cross-entropy loss of g_ϕ as $\mathcal{L}_{\text{base}}(g_\phi(\boldsymbol{x}), y)$ and construct additional loss functions with Jacobian regularisation terms as follows:

$$\mathcal{L}_*(\boldsymbol{x}, y) = \mathcal{L}_{\text{base}}(g_\phi(\boldsymbol{x}), y) + \lambda_n \|\boldsymbol{J}(\boldsymbol{x})\|_*, \tag{6}$$

$$\mathcal{L}_{\text{cond}}(\boldsymbol{x}, y) = \mathcal{L}_{\text{base}}(g_\phi(\boldsymbol{x}), y) + \lambda_c \|\kappa\left(\boldsymbol{J}(\boldsymbol{x})\right) - 1\|^2. \tag{7}$$

We set $\lambda_n = 1$ and $\lambda_c = \frac{1}{2}$ in later experiments, and add 10^{-6} to the denominator of $\kappa\left(\boldsymbol{J}\right)$ to avoid underflow in the early stages of training. The total loss to be minimised is then computed as the average over samples in a mini-batch.

The neural ODE is implemented using the torchdiffeq framework [3]. The neural ODE and linear classifier are trained jointly, end-to-end. The addition of a linear classifier encourages the neural ODE vector field to linearly separate the classes. Training is done for 300 epochs, and repeated for 10 random seeds to determine stability across runs.

In high dimensions, direct computation of the Jacobian becomes computationally expensive. For now, as a test-bed for our hypotheses, the 2-dimensional intertwining moons dataset shown in Figure 3 is used as a running example. Other datasets are considered in Section 5, and more efficient methods to scale to higher dimensions will be mentioned in Section 7.

We are interested in the relationships between NFE as defined in Section 2, generalisation performance, and the sensitivity of a trained model's output under perturbations. The remainder of this section discusses the metrics used in our experiments.

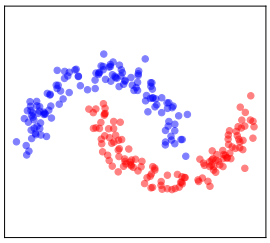

Fig. 3. The intertwining moons dataset (red and blue indicate class labels).

Generalisation and sensitivity are investigated by means of performance on a hold-out test set, as well as input perturbations. The input perturbations include varying levels of Gaussian noise, and adversarial perturbations created by the fast gradient sign method [10]:

$$\boldsymbol{x}_p = \boldsymbol{x} + \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, y)\right), \tag{8}$$

where ϵ is varied to increase the severity of the adversarial attack. This kind of perturbation moves a test sample $\boldsymbol{x}$ in a direction that increases the value of $\mathcal{L}(\boldsymbol{x}, y)$, such that it is more likely for a trained model to predict an incorrect class label. Standard test set performance corresponds to $\epsilon = 0$.

Jacobian norms and condition numbers are averaged over all data points for each epoch. Keeping track of the norms and condition numbers can aid in determining the effect of each regularisation strategy. For instance, we will show in Section 4 that regularising the Frobenius and spectral norm of the Jacobian can make the ODE system more ill-conditioned.

Distance to decision boundary is a common metric in neural network sensitivity analysis, with the hypothesis that larger distances correspond to a more robust model. To determine the distance to a decision boundary, we generate points on d-dimensional spheres uniformly at random. The spheres are centered at a data point, with increasing radii. We perform a linear search over the spheres to determine the largest radius for which points are still labelled consistently. This can be made more efficient using a binary search algorithm.

4 Results

4.1 Generalisation and sensitivity

Regularising $\|\boldsymbol{J}\|_F$ and $\|\boldsymbol{J}\|_2$ more than halves the NFE during training, as shown in Figure 4. However, Figure 5 shows a reduced test accuracy at $\epsilon = 0$

(no perturbations), revealing that complex dynamics might be a prerequisite for linearly separable representations of the input data. The steeper slope for Jacobian norm regularisation in Figure 5 suggests that norm regularisation leads to classifiers that are more vulnerable to adversarial perturbations than the other regularisation strategies, for small values of ϵ. Even though Jacobian norm regularisation does not lead to adversarially robust classifiers, they do provide models that are more stable across runs, as is evident from the lower standard deviations in Figures 4 and 5. On the contrary, Figure 6 shows a slower decline in performance under random (Gaussian) perturbations. This could relate to our observation that Jacobian norm regularisation leads to larger classification margins (as detailed in Section 4.3). The effect of random noise perturbation is mitigated by a larger margin, since those perturbations are less likely to cross decision boundaries. Adversarial perturbations specifically move the input data in a direction that will lead to a miss-classification.

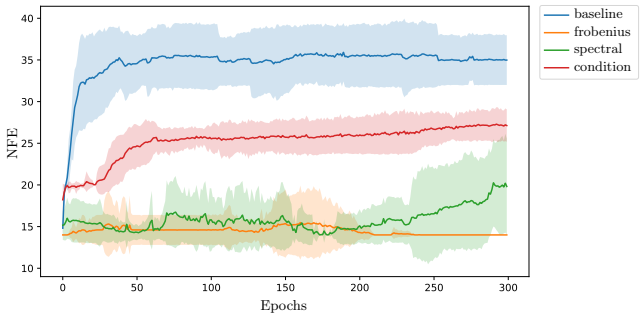

Fig. 4. Regularising the Jacobian norm (both Frobenius and spectral) and condition number controls the NFE required by the ODE solver during training. The baseline trains with no regularisation. Solid curves and shaded regions indicate mean and standard deviation over 10 runs.

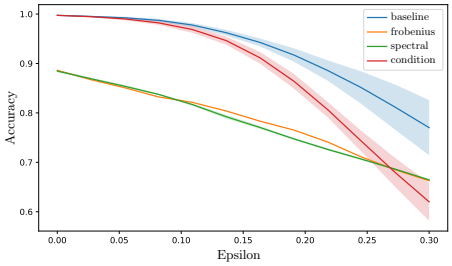

Fig. 5. Accuracy as a function of adversarial perturbations. $\epsilon = 0$ corresponds to standard test set performance. The baseline seems to be the most robust.

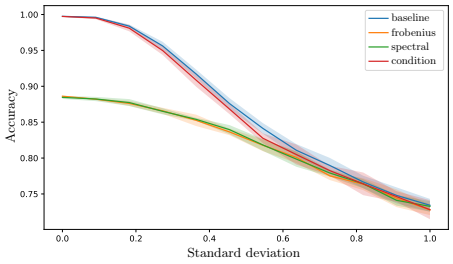

Fig. 6. Accuracy as a function of Gaussian perturbations. Standard deviation of 0 corresponds to standard test set performance.

Regularising the condition number to be closer to 1 also reduces the NFE, though not as significantly as Jacobian norm regularisation. On the other hand, condition number regularisation achieves a reduced NFE at virtually no cost to test accuracy. Training converges more stably when compared to the baseline (no regularisation). Interestingly, the baseline remains the least sensitive to adversarial perturbations. It is possible that the degree of complexity for unregularised dynamics in two dimensions is higher than the impact of adversarial perturbations, especially considering that the data coverage here is fairly dense.

4.2 Jacobian norms and condition numbers

Table 1 shows that while Jacobian norm regularisation reduces the NFE, the condition number of the Jacobian increases compared to both the baseline and Jacobian conditioning regularisation. That is, Jacobian norm regularisation can make the ODE more ill-conditioned. Moreover, the standard deviation on the condition number is significantly higher for Jacobian norm regularisation. The issue here is that both Jacobian norm regularisation strategies indiscriminately push $\sigma_{\min}$ to zero (Figure 7), driving the condition number to increase. In some cases, $\sigma_{\min}$ becomes so small that $\kappa(\boldsymbol{J})$ increases drastically. In fact, an extreme outlier for spectral norm regularisation on the order of 10^6 was omitted from the results in Table 1. As singular values tend to zero, the Jacobian matrix becomes closer to singular. It might be worth investigating the stiffness of neural ODEs. In some cases, the condition number is related to the stiffness index $S = \kappa(\boldsymbol{J})(t_1 - t_0)$. Stiff ODEs are numerically unstable and often require very small step sizes to solve, which may in turn increase the NFE, or even call for the use of implicit solvers.

Table 1. Measures of NFE, test accuracy and Jacobian condition number, for the different regularisation strategies investigated. The Jacobian condition number in the last column is an average over the training data after the final epoch of training.

	Intertwining moons		
Regularisation	NFE	Test accuracy	Condition number
None	34.98 ± 2.98	0.9975 ± 0.0008	5.31 ± 3.51
Frobenius	14.00 ± 0.00	0.8862 ± 0.0003	27.3 ± 34.1
Spectral	19.81 ± 5.76	0.8846 ± 0.0022	45.9 ± 70.6
Condition number	27.12 ± 1.94	0.9973 ± 0.0007	6.10 ± 5.22

Figure 7 shows that Jacobian condition number regularisation controls the spectral norm to some extent. The two graphs also indicate that Jacobian condition number regularisation effectively pushes $\kappa(\boldsymbol{J})$ closer to 1.

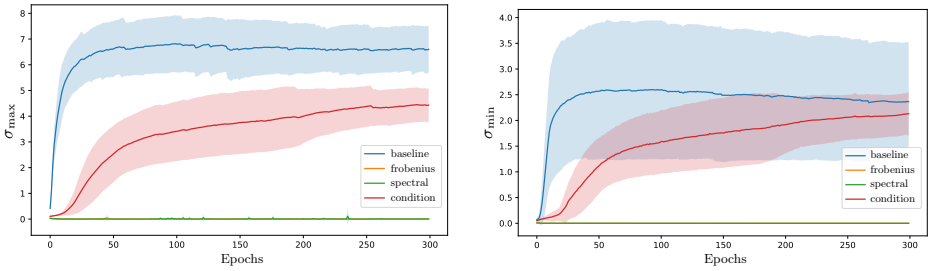

Fig. 7. Regularising the Jacobian condition number controls $\sigma_{\max}$ (left) and $\sigma_{\min}$ (right).

4.3 Distance to decision boundary

Figure 8 shows average distance to decision boundaries, for different regularisation strategies. Jacobian condition number regularisation results in little change from the baseline (no regularisation). Jacobian norm regularisation, on the other hand, does increase the average distance to the decision boundary for correctly classified points, and is consistent with prior work [14]. There seems to be a trade-off between having a classifier with a large margin and sufficiently complex dynamics for effective classification.

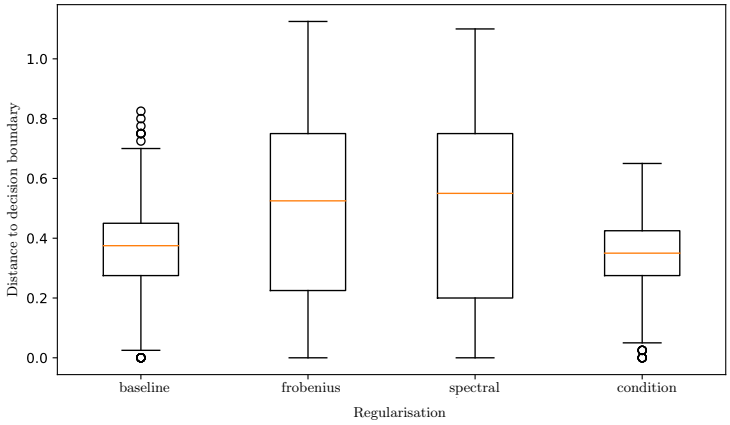

Fig. 8. Box-and-whisker plot of distance to decision boundary for different regularisation strategies, over the training data points. Jacobian norm regularisation increases distance to decision boundary on average, consistent with the literature.

5 Additional datasets

To verify the findings of the previous section, we consider additional datasets of varying complexity, as shown in Figure 9. We consider the spiral dataset as a more challenging version of the intertwining moons dataset. Then, as a known pathological case for neural ODEs, we consider the annulus dataset. For these data points to become linearly separable, the inner circle must pass through the outer circle. Since neural ODEs can only learn smooth homeomorphisms [7, 18], solution trajectories cannot cross. To lessen the challenge of the annulus, we also consider the broken annulus dataset where part of the outer circle has been removed, allowing a valid trajectory for points in the inner circle. We train neural ODE classifiers for each of these three datasets, over 100 epochs. Finally, we consider 5-dimensional latent representations of the 10-class MNIST dataset, from a trained autoencoder. Neural ODE classifiers for the MNIST dataset are trained over 25 epochs.

We again compare NFEs and test accuracy of an unregularised neural ODE and ones trained with Jacobian norm and condition number regularisation. Training of all models is repeated for 10 random seeds.

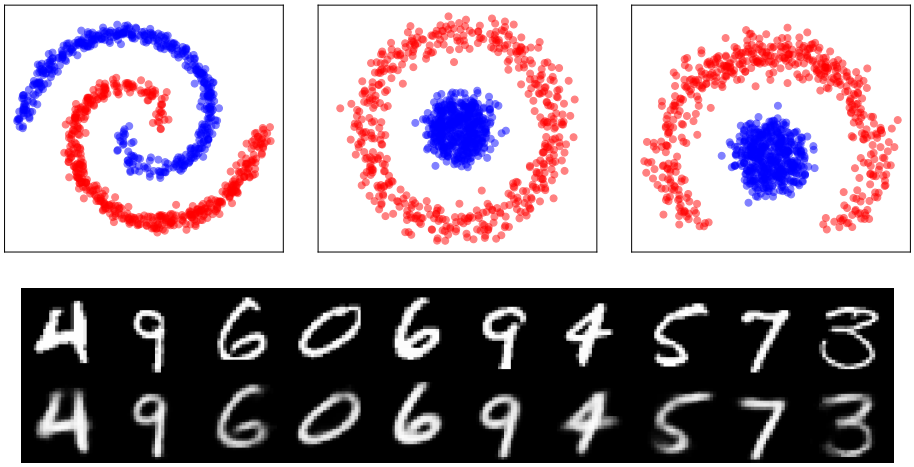

Fig. 9. Additional datasets used in our experiments: spiral (top left), annulus (top middle), broken annulus (top right), and 5-dimensional autoencoder representations of MNIST (here we show samples of original and reconstructed images in the bottom row).

NFE, accuracy and condition numbers are reported in Table 2. The finding that Jacobian condition number regularisation reduces NFE without a significant cost to generalisation performance, is consistent for all datasets. Surprisingly, Jacobian norm regularisation leads to test performance worse than random on

Table 2. NFEs, test accuracies and condition number for regularisation strategies investigated.

Spiral dataset			
Regularisation	NFE	Test accuracy	Condition number
None	40.27 ± 7.28	0.9978 ± 0.0030	20.6 ± 11.6
Frobenius	14.00 ± 0.00	0.7425 ± 0.0108	18.6 ± 13.7
Spectral	14.02 ± 0.06	0.7517 ± 0.0159	30.0 ± 20.0
Condition number	26.94 ± 11.3	0.8620 ± 0.1006	7.88 ± 5.88

Annulus			
Regularisation	NFE	Test accuracy	Condition number
None	38.08 ± 4.50	0.9528 ± 0.0251	23.2 ± 16.6
Frobenius	14.02 ± 0.06	0.6506 ± 0.0058	39.1 ± 38.7
Spectral	18.11 ± 10.8	0.6541 ± 0.0058	40.5 ± 58.8
Condition number	29.36 ± 3.18	0.9335 ± 0.0090	5.88 ± 5.53

Broken annulus			
Regularisation	NFE	Test accuracy	Condition number
None	34.40 ± 3.03	0.9965 ± 0.0032	11.7 ± 4.46
Frobenius	16.10 ± 4.66	0.4562 ± 0.1246	29.6 ± 43.7
Spectral	14.96 ± 1.90	0.3984 ± 0.1032	70.2 ± 61.9
Condition number	30.01 ± 3.96	0.9917 ± 0.0068	15.0 ± 29.7

5-dimensional MNIST			
Regularisation	NFE	Test accuracy	Condition number
None	29.59 ± 2.20	0.9522 ± 0.0012	2.47 ± 0.40
Condition number	19.19 ± 1.38	0.9364 ± 0.0011	1.54 ± 0.18

the broken annulus dataset. It is possible that Jacobian norm regularisation asserts too strong an inductive bias on trajectories learnt by the neural ODE, such that some types of dynamics are suppressed regardless of how simple a dataset may be. For the MNIST dataset, which is higher in dimension and has more classes, results remain consistent: Jacobian condition number regularisation can reduce NFE without a significant cost to test accuracy.

In Figure 10, we report adversarial and Gaussian robustness for the additional 2-dimensional datasets. Again, the baseline seems to be most robust to adversarial perturbations, except for the annulus and broken annulus datasets, where there is not much difference between the baseline and condition number regularisation. The slower decline in accuracy under Gaussian perturbations (spiral and annulus dataset) is again evident for Jacobian norm regularisation, especially for the spiral dataset.

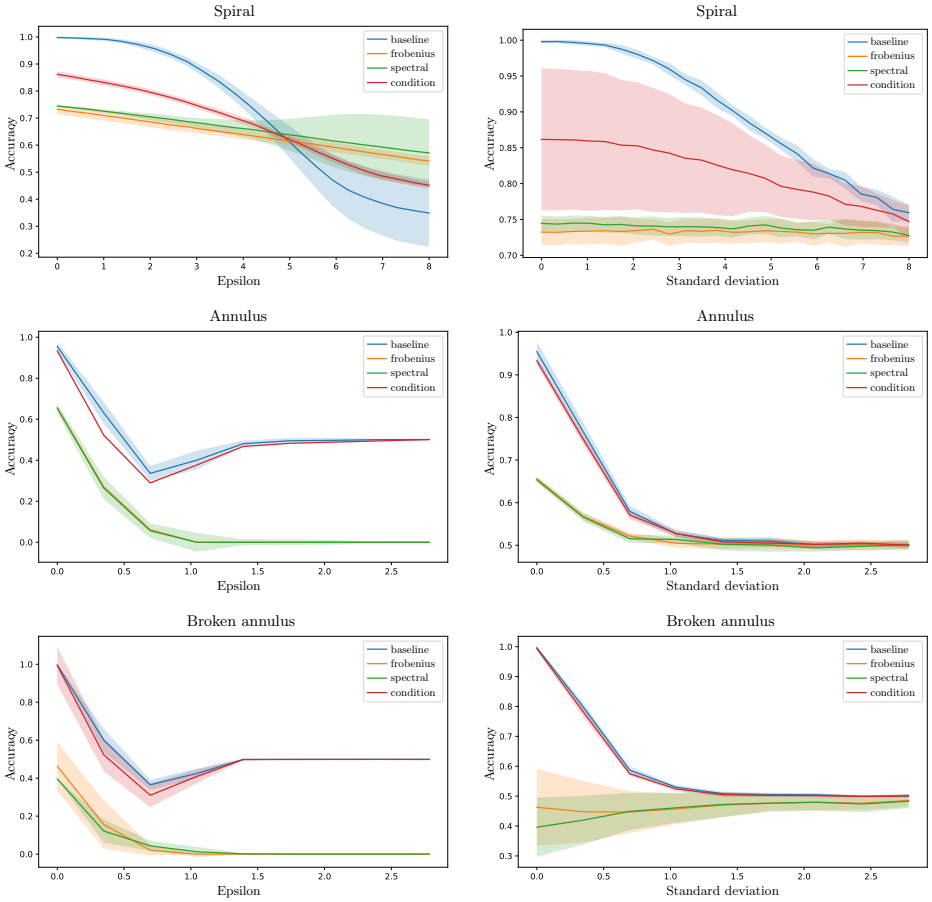

Fig. 10. Accuracy as a function of adversarial perturbations (left), and accuracy as a function of Gaussian perturbations (right), for the additional 2-dimensional datasets.

6 Related work

As mentioned earlier, neural ODEs can be regarded as continuous depth neural networks that are more memory efficient and separate the definition of the model from the forward pass. Neural ODEs may also be suitable for modelling continuous-time data from systems such as physical systems [20, 22, 25, 27], continuous-time time series [6, 16, 21] and continuous normalising flows [4, 8, 9, 17, 19]. Neural ODEs also appear in score-based generative modelling where a stochastic differential equation is used to model the data generation process, and have been shown to generate GAN-level quality samples without the need for adversarial training (which can suffer from issues such as mode collapse) [24]. Song et al. [24] convert the stochastic differential equation into an ordinary differential equation whose solution allows for the generation of samples.

It is important to acknowledge that there are modelling disadvantages to neural ODEs. In fact, neural ODEs are not universal function approximators: they can only learn smooth homeomorphisms [7, 18] due to the fact that solution trajectories cannot cross one another. Dupont et al. [7] circumvent this by augmenting the dimensions of the ODE, akin to the kernel trick in SVMs. Problems requiring complex dynamics and a high number of function evaluations then become simpler to solve with lower NFEs, much like the aim of our work. Yan et al. [12] suggest that the restriction of smooth homeomorphism can lead to more robust models, and perform perturbation vulnerability experiments by comparing neural ODEs with CNNs. Most related to our paper are the works of Finlay et al. [9] who regularise the Frobenius norm of the Jacobian in the context of normalising flows, and Kelly et al. [15] who encourage simpler dynamics by introducing a differentiable proxy for the time cost of a numerical solver. Bai et al. [2] observe increasing instability during the training of deep equilibrium models [1], a class of deep implicit layers where the solution procedure is defined by a fixed point iteration. This instability leads to an increased NFE and a higher likelihood of divergence from a fixed point, all of which are reduced by regularising the Frobenius norm of the Jacobian at the fixed point.

There also exists works in the broader machine learning literature that regularise the Frobenius and spectral norm of a neural network's weights or input-output Jacobian, towards improving generalisation and robustness [11, 13, 14, 23, 26]. Similar to our paper, these works operate under the hypothesis that norm regularisation can reduce sensitivity to perturbations in the input. Yoshida et al. [26] provide an upper bound to the spectral norm of a neural network's weight matrices that allows for more efficient regularisation. Johansson et al. [14] extend this by providing an exact method to compute the spectral norm of a neural network's input-output Jacobian, through a power iteration procedure. Our findings on an increased distance to decision boundary for Frobenius and spectral norm regularisation are consistent with those of Johansson et al. [14]. We further expect that their approach can be applied to neural ODEs, so that our experiments can be scaled to higher dimensional datasets. Hoffman et al. [13] provide an efficient computation for the Frobenius norm of a Jacobian through random projections. Their method, however, is not applicable to spectral norms or condition numbers, as those cannot be written in terms of a matrix trace operation.

7 Conclusion

We investigated the required number of function evaluations (NFE) and conditioning of a neural ODE system under different Jacobian regularisation strategies. Our experiments show that while Jacobian norm regularisation is effective at reducing the NFE of neural ODE solvers, they do so at the cost of test accuracy and may make the Jacobian matrix ill-conditioned. Jacobian condition number regularisation reduces NFE without a significant loss in accuracy, and controls the norm and condition number of the Jacobian. We acknowledge that

this study involved small datasets, and should be regarded as an initial exploration into neural ODE conditioning. Moreover, Jacobian condition number regularisation adds a significant computational overhead to the training process that might not be balanced by a reduced NFE. Future work will therefore investigate a more efficient computation of the condition number, similar to the power iteration already attempted for convolutional neural networks [14, 26]. This will allow experiments to be scaled to higher dimensional datasets, such as images, and may also allow for the regularisation of dynamics across the entire solution trajectory [15]. Higher dimensional datasets are important to consider given that the goal of reducing NFE through Jacobian regularisation is to make neural ODEs more practically relevant. Another possible direction would be to characterise the stiffness of the ODE at the solutions, as stiff ODEs often require higher NFEs or an implicit solver.

References

1. Bai, S., Kolter, J.Z., Koltun, V.: Deep equilibrium models. Advances in Neural Information Processing Systems. **32**, 690–701 (2019)
2. Bai, S., Koltun, V., Kolter, J.Z.: Stabilizing equilibrium models by Jacobian regularization. International Conference on Machine Learning. **139**, 554–565 (2021)
3. Chen, R.T.: torchdiffeq: PyTorch implementation of differentiable ODE solvers (2018), https://github.com/rtqichen/torchdiffeq
4. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. Advances in Neural Information Processing Systems. **31**, 6572–6583 (2018)
5. Coddington, E.A., Levinson, N.: Theory of ordinary differential equations. Tata McGraw-Hill Education (1955)
6. De Brouwer, E., Simm, J., Arany, A., Moreau, Y.: GRU-ODE-Bayes: continuous modeling of sporadically-observed time series. Advances in Neural Information Processing Systems. **32**, 7377–7388 (2019)
7. Dupont, E., Doucet, A., Teh, Y.W.: Augmented neural ODEs. Advances in Neural Information Processing Systems. **32**, 3134–3144 (2019)
8. Falorsi, L., Forré, P.: Neural ordinary differential equations on manifolds. arXiv:2006.06663 (2020)
9. Finlay, C., Jacobsen, J.H., Nurbekyan, L., Oberman, A.: How to train your neural ODE: the world of Jacobian and kinetic regularization. International Conference on Machine Learning. **119**, 3154–3164 (2020)
10. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. International Conference on Learning Representations (2015)
11. Gu, S., Rigazio, L.: Towards deep neural network architectures robust to adversarial examples. International Conference on Learning Representations, ICLR Workshop Track Proceedings (2015)
12. Hanshu, Y., Jiawei, D., Vincent, T., Jiashi, F.: On robustness of neural ordinary differential equations. International Conference on Learning Representations (2020)
13. Hoffman, J., Roberts, D.A., Yaida, S.: Robust learning with Jacobian regularization. arXiv:1908.02729 (2019)
14. Johansson, A., Strannegård, C., Engsner, N., Mostad, P.: Exact spectral norm regularization for neural networks. arXiv:2206.13581 (2022)

15. Kelly, J., Bettencourt, J., Johnson, M.J., Duvenaud, D.K.: Learning differential equations that are easy to solve. Advances in Neural Information Processing Systems. **33**, 4370–4380 (2020)
16. Kidger, P., Morrill, J., Foster, J., Lyons, T.: Neural controlled differential equations for irregular time series. Advances in Neural Information Processing Systems. **33**, 6696–6707 (2020)
17. Lou, A., Lim, D., Katsman, I., Huang, L., Jiang, Q., Lim, S.N., De Sa, C.M.: Neural manifold ordinary differential equations. Advances in Neural Information Processing Systems. **33**, 17548–17558 (2020)
18. Massaroli, S., Poli, M., Park, J., Yamashita, A., Asama, H.: Dissecting neural ODEs. Advances in Neural Information Processing Systems. **33**, 3952–3963 (2020)
19. Mathieu, E., Nickel, M.: Riemannian continuous normalizing flows. Advances in Neural Information Processing Systems. **33**, 2503–2515 (2020)
20. Miles, C., Sam, G., Stephan, H., Peter, B., David, S., Shirley, H.: Lagrangian neural networks. International Conference on Learning Representations, Workshop on Integration of Deep Neural Models and Differential Equations. (2020)
21. Rubanova, Y., Chen, R.T., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. Advances in Neural Information Processing Systems. **32**, 5321–5331 (2019)
22. Sanchez-Gonzalez, A., Bapst, V., Cranmer, K., Battaglia, P.: Hamiltonian graph networks with ODE integrators. arXiv:1909.12790 (2019)
23. Sokolić, J., Giryes, R., Sapiro, G., Rodrigues, M.R.: Robust large margin deep neural networks. IEEE Transactions on Signal Processing. **65**(16), 4265–4280 (2017)
24. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. International Conference on Learning Representations (2020)
25. Wang, W., Axelrod, S., Gómez-Bombarelli, R.: Differentiable molecular simulations for control and learning. International Conference on Learning Representations, Workshop on Integration of Deep Neural Models and Differential Equations. (2020)
26. Yoshida, Y., Miyato, T.: Spectral norm regularization for improving the generalizability of deep learning. arXiv:1705.10941 (2017)
27. Zhong, Y.D., Dey, B., Chakraborty, A.: Symplectic ODE-net: learning Hamiltonian dynamics with control. International Conference on Learning Representations (2019)