Noisy Learning for Neural ODEs Acts as a Robustness Locus Widening

Martin Gonzalez [1] **Hatem Hajri** [1] **Loic Cantat** [1] **Mihaly Petreczky** [2]

Abstract

We investigate the problems and challenges of evaluating the robustness of Differential Equation-based (DE) networks against synthetic distribution shifts. We propose a novel and simple accuracy metric which can be used to evaluate intrinsic robustness and to validate dataset corruption simulators. We also propose methodology recommendations, destined for evaluating the many faces of neural DEs' robustness and for comparing them with their discrete counterparts rigorously. We then use this criteria to evaluate a cheap data augmentation technique as a reliable way for demonstrating the natural robustness of neural ODEs against simulated image corruptions across multiple datasets.

1. Introduction

Neural Ordinary Differential Equations (NODEs) (Chen et al., 2019), conjoining dynamical systems (DS) and machine learning (ML), have come to be a popular source of interest, in particular for tackling generative problems and continuous-time modeling, and seem to have a bright future among the ML community. Nevertheless, many questions regarding their robustness have been raised, creating a debate on whether these networks benefit from natural robustness properties or if the latter are overestimated.

Likewise, the importance of *reliability* in real-world applications with AI-driven decision-making in safety-critical systems have brought a lot of attention to studying a model's behavior under *distribution shifts*. Understanding the latter implies focusing on how feasible is a chosen model's domain generalization against the kinds of shifts that may occur in real-world scenarios. The past few years have seen an emerging industry proposing new and relevant shifted datasets for different actors and purposes. Numerous benchmarks (Hendrycks & Dietterich, 2019; Mu & Gilmer, 2019; Koh et al., 2021; Salehi et al., 2021) addressing different aspects of distribution shifts have come to light and the rigorous analysis and evaluation of both models and benchmarks have become increasingly important. Although they transfer poorly to real-world shifted images, synthetic distribution shifts are a good starting point for experimenting a new model's accuracy and robustness. For instance, in (Gilmer et al., 2019) it is hypothesized that methods that incur into vanishing gradients also show no improvement in Gaussian noise, a phenomenon which they relate in a rigorous way to adversarial attacks. Corruption robustness can be then seen as a *sanity check* to ensure that a proposed adversarial defense method doesn't present gradient masking. Nevertheless, it is important to separate accuracy improvements from robustness improvements when interpreting the results and different metrics have been proposed for doing so (Hendrycks et al., 2021; Taori et al., 2020). For common corruptions (Hendrycks & Dietterich, 2019), the (un-normalized, unaveraged) relative Corruption Error (rCE) is the difference[1] of the model's corrupted and clean errors. As the very notion of a corruption is always relative to a clean counterpart, we find that this metric has a particular weakness for simulated corruptions as it doesn't take into account the following structural principle underlying such corruptions: *miss-classified clean images should result in miss-classified simulated corruptions*. As such, the rCE answers questions like "how much does the model decline under corruption inputs" but it doesn't detect the corruption error contributions coming from clean miss-classifications.

This brief account aims to lay initial ground on theoretical and application-driven aspects, problems and methodology perspectives for evaluating robustness of NODEs against synthetic distribution shifts. For this purpose, we

1. assess and highlight several properties of NODEs in connection to different robustness criteria, link them to specific aspects of real-world data features they may capture and determine general guidelines on when and how they can be compared to static networks or between them;

2. introduce an intrinsic robustness metric $\mathcal{A}_c^{\text{rel}}$, well-

[1]Institut de Recherche Technologique SystemX, Palaiseau, France [2]Centre de Recherche en Informatique, Signal et Automatique de Lille, France. Correspondence to: Martin Gonzalez <martin.gonzalez@irt-systemx.fr>.

Presented at ICML Workshop on Principles of Distribution Shift 2022,

[1]Precise definitions are recalled in (2), Appendix A.

suited for evaluating well-posedness of dataset corruption simulators, and capable of measuring a model's corruption accuracy more subtly than the rCE (Hendrycks & Dietterich, 2019);

3. evaluate an easy-to-implement robustifying method for NODEs against corrupted images, leading us to conclude that NODEs are naturally more robust to several synthetic distribution shifts than their discrete counterparts and that noisy learning for NODEs acts, as expected, as a robustness locus widening.

We aim to propose a baseline upon which to build step-wise incremental implementations of robustifying methods for NODEs under such corruptions. It is our hope that our proposed metric and methodology recommendations will be helpful both when studying implicit nets robustness and when designing new and more diverse corruption simulation algorithms and datasets.

2. On evaluating common robustness for neural ODEs

Evaluating robustness of NODEs is particularly challenging: their output is computed via iterative optimization schemes and such test-time optimization has shown to prevent the proper evaluation of established robustness methods designed for static networks like AUTOATTACK in the adversarial context (Croce et al., 2022). Additionally, comparing NODEs to chosen static analogs has shown to disregard implicit assumptions (adaptive step-size solvers, inexact backward pass computation) which pose methodological problems preventing to formally compare them and ultimately incurs into falsifying the results of many conducted experiments. We will concentrate on classification tasks in this report.

Neural ODEs meet dynamical systems: Denote h_x a feature extractor (FE) and h_y a fully-connected classifier (FCC). The inference of a NODE model is carried out by solving,for $\dot{z}$ denoting the time-derivative:

$$\begin{cases} \dot{z}(t) = f(t, z(t), \theta(t), x) \\ z(0) = h_x(x) \\ \hat{y}(T_x) = h_y(z(T_x)) \end{cases} \qquad t \in \mathcal{T}_x = [0, T_x] \qquad (1)$$

Contrary to static architectures, f formalizes the dynamics controlling a *continuous-in-depth* model, $t \in \mathcal{T}_x$ being its depth variable and the components in (1) traduce the following features: the dependence on h_x for $z(0)$ is traduced by input layer augmentation; the dependence on t for f (resp. θ) is traduced by depth-dependence (resp. depth-variance[2]) and is taken in practice as an augmentation component (Dupont et al., 2019); the dependence on x for f (resp. T_x) is traduced as data-control (resp. depth-adaptation) and can traduce recurrent architectures. We refer to (Massaroli et al., 2020) for details on these features and to (Kidger, 2022) for a clear comprehensive introduction to neural DEs.

Several overlaps occur between ML and DS modeling techniques and approaches which we now try to articulate to shed light on their singular benefits. These distinctions will be the basis of our methodology guidelines for evaluating and comparing general Neural Differential Equation (NDE) models.

DS-inspired neural DEs: These consist in manufacturing constraints on a loss function or on the weight matrices inside the dynamics that would enhance their robustness from a stability analysis point of view. Another way of stating DS-inspired NDEs is to say that the ML focus comes **post-hoc** the DS focus: the trained architecture is supposed to have benefit of theoretical properties at training or inference. We highlight the fact that such approaches can serve different purposes: (Pal et al., 2022; Ivan et al., 2022; Djeumou et al., 2022) address mainly speed problems while (Kang et al., 2021; Yan et al., 2020; Huang et al., 2022) addresses stability training considerations (Li et al., 2019) for NODEs e.g. using steady-states, Lyapunov equilibrium points. This usually is an idealized analysis made upon an idealized ML architecture. A common problem would be to neglect the numerical errors that come to hand while training, which were absent from classical discrete neural networks. First, in (Ott et al., 2021) it is shown that there exists a critical step size only beyond which the training yields a valid ODE vector field. Thus, for instance, the system theoretic formulation of the Picard-Lindelöf theorem, used for ensuring non intersecting trajectories, effectively applies only if such condition is met. *Methodology point*: ensure that the discretization resulting from the numerical solver's execution preserves formalized DS properties. A first characterization of robustness for NODEs is then met.

DS-based neural DEs: These consist in formalizing NDE architectures as analogs of system theoretic paradigms such as a full use of the components of (1) but also formalizing neural CDEs, SDEs and PDEs (Kidger, 2022; Fermanian et al., 2021; Xu et al., 2022; Li et al., 2021). Here, the ML focus comes **ante-hoc** the DS focus: the formalized architecture is supposed to be endowed with structural characteristics both at training and inference. This approach is more involved than the previous one as it needs to have at hand simultaneously an adapted, analytically proven, adjoint method analog or generalization, and a non empty choice of adapted numerical solvers. For instance, Kidger (Kidger, 2022) adapted the analytic adjoint method for neural CDEs and neural SDEs while developing for the latter an algebraically reversible *Heun method* SDE solver. While

[2]When θ is a constant function, we still use the term depth-dependence.

NODEs have served as inspiration for constructing many discrete neural architectures by formalizing in continuous-time an ODE and discretizing it, the difficulty has been to create continuous time analogs for discrete neural components. For instance, crafting a stateful batch normalization (BN) layer has been recently reflected in the NODE formulation (Queiruga et al., 2021) as a generalized ODE. On the contrary, in (Huang et al., 2022) ResNets have BN layers, NODEs have group normalization (GN) layers and the length of the skip connection does not coincide between the compared architectures and in (Xu et al., 2022), although testing NODEs against corruptions, a mix between deterministic and stochastic methods may weaken their claims. *Methodology point*: ensure that the chosen NDE architecture identifies in a clear manner all arguments of the function passed to the DE solver, determine if the latter is an exact or an approximate solver; distinguish stochastic and deterministic architectures; comparing NDEs and discrete architectures should be mathematically justified by an explicit end-to-end discretization scheme, taking into account the nature of the h_x and h_y layers and identifying, for instance, NODE blocks and weight-tied residual blocks. This constitutes a second robustness characterization.

DS-destined neural DEs: These consist in manufacturing NDEs that incorporate known modeling physical constraints. Here, the ML focus is **ad-hoc** to the DS focus: the proposed architecture is supposed to capture *intrinsically* the dynamics (e.g. Lagrangians, Hamiltonians) of the studied phenomenon and NDEs *specify* DEs (Zhong et al., 2020). This is different, though somehow related, to *physics-informed* NNs (Karniadakis et al., 2021), which aim to obtain solutions through NNs to *pre-specified* DEs for which traditional solvers are computationally expensive. Well-defined NDEs may not effectively capture continuous-time inductive biases if the used numerical ODE methods have too low order of convergence while high-order methods need for fast and exact gradient computations (Matsubara et al., 2021; Djeumou et al., 2022). *Methodology point*: conduct NDE-oriented numerical *convergence* tests, presenting a non-trivial difference between the ML-based (Bottou & Bousquet, 2007) and the round-off (Chaitin-Chatelin & Frayssé, 1996) numerical errors, such as proposed in (Krishnapriyan et al., 2022) to check if the implemented model successfully learned meaningful continuous dynamics. We then find a third robustness characterization for such networks.

Leveraging well-studied mathematical approaches for stability, robustness and resilience[3] into the continuous-time ML community can prove to be very advantageous and the above robustness properties, while already being studied jointly in the cited works, call for clear distinctions between such notions. Their formal analysis will be the subject of an extended version of the present report, complementary to the system-theoretic approach which is being conducted in parallel (Gonzalez et al., 2022).

Intrinsic robustness metrics: We now define the metric $\mathcal{A}_c^{\text{rel}}$ mentioned in the introduction of this report. Let C be a set of simulated corruptions $c = (\tilde{c}, s)$, where $\tilde{c}$ is a corruption label and s is a severity level. We make the assumption that for each $c \in C$ one can generate (at least) one corruption simulation x_c of a clean image x. We denote y the true label of x, y_{cl} the model's prediction on x and y_c the model's prediction on x_c. Let N be the size of the dataset. Define $M = \sum_{i=1}^{N} \mathbb{I}_{\{y_{\text{cl}}^i = y^i\}}$, $\mathcal{A}_{\text{cl}} = M/N$, and for $c \in C$,

$$\mathcal{A}_c = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{\{y_c^i = y^i\}}, \quad \mathcal{A}_c^{\text{rel}} = \frac{1}{M}\sum_{i=1}^{N} \mathbb{I}_{\{y_c^i = y^i \,\&\, y_{\text{cl}}^i = y^i\}}$$

where $\mathbb{I}_{\{y_{\text{cl}}=y\}} = 1$ if $y_{\text{cl}} = y$ and 0 else. *Methodology point*: the clean accuracy $\mathcal{A}_{\text{cl}}$ is used to save the model's parameters during training; the absolute corruption accuracy $\mathcal{A}_c$ gives the model's accuracy for a corruption c; the relative corruption accuracy $\mathcal{A}_c^{\text{rel}}$ computes how many corrupted simulations x_c were correctly classified among the correctly classified clean counterparts; the positiveness of the rCE remains a relevant sanity check for debugging and verifying that the above hypothesis is preserved by the corruption simulator. By leveraging the above-mentioned structural principle for simulated corruptions, this metric addresses more accurately (in the statistical analysis sense) questions like "how do corruptions intrinsically behave for this model".

Our experiments show that $\mathcal{A}_c^{\text{rel}}$ doesn't overestimate the model's robustness and abnormal behavior[4] implies that a selected corruption simulation is ill-posed. On the other hand, the Relative mCE increasingly underestimates it, as highly accurate models will see a greater proportion of their miss-classified corruptions to come from miss-classified clean samples, making $\mathcal{A}_c$ to decrease while clean accuracy increases, as shown in Fig. 3 of (Hendrycks & Dietterich, 2019). This phenomenon is confirmed in Figures 5–8 of (Mu & Gilmer, 2019) where miss-classified clean images represent 12% of the shown examples and none of them incur into well-classified associated corruptions.

3. Experiments

We propose a minimalist, yet precise, comparative analysis on the robustness of a simple NODE (ODENet) and its discrete counterpart (ResNet) against simulated image corruptions. The chosen[5] Models are trained on MNIST with

[3] For instance, NODEs with depth-adaptation should be evaluate their recovery rate in time *after* an input perturbation.

[4] Such as unexpectedly observing $\mathcal{A}_c > \mathcal{A}_c^{\text{rel}}$.

[5] Results of the remaining corrupted datasets and specifics on the chosen architectures are available in Appendix A.

Table 1. Mean $\mathcal{A}_c^{\text{rel}}$ (%) on corrupted MNIST. A $> 5\%$ difference of model's performance is colored in orange. The last block computes the improvement on $\mathcal{A}_c^{\text{rel}}$ for each model induced by noisy training w.r.t. clean training. The listed corruptions are 1: gaussian, 2: shot, 3: impulse, 4: defocus, 5: glass, 6: motion, 7: zoom, 8: snow, 9: frost, 10: fog, 11: brightness, 12: contrast, 13: elastic_transform, 14: pixelate, 15: jpeg_compression.

Model	Training	$\mathcal{A}_{\text{cl}}$	$\mathcal{A}_c^{\text{rel}}$ on Noise			$\mathcal{A}_c^{\text{rel}}$ on Common Corruptions (severity 1)														
			Gaussian (σ)			Noise			Blur				Weather				Digital			
			50	75	100	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ResNet	Clean	99.45	98.1	88.9	66.3	99.7	99.8	98.8	79.9	35.9	96.9	99.4	98.1	97.9	58.9	99.7	97.5	67.2	98.8	99.8
ODENet		99.59	**99.2**	**89.2**	**74.1**	**99.9**	**99.9**	**99.4**	**82.7**	**57.3**	**98.1**	**99.8**	**99.5**	**99.3**	**93.4**	**99.9**	**99.5**	**77.9**	**98.8**	**99.9**
ResNet	Noisy	99.44	-	-	98.7	99.9	99.2	99.8	81.7	71.8	93.0	99.6	99.4	99.6	85.3	99.8	97.8	91.3	99.5	99.9
ODENet		99.59	-	-	**99.3**	**99.9**	**99.9**	**99.9**	**91.2**	**87.1**	**98.6**	**99.8**	**99.7**	**99.9**	**95.2**	**99.9**	**99.6**	**95.9**	**99.7**	**99.9**
ResNet	Noisy $\mathcal{A}_c^{\text{rel}}$- clean $\mathcal{A}_c^{\text{rel}}$				**32.4**	**0.2**	-0.6	**1**	1.8	**35.9**	-3.9	**0.2**	**0.7**	**1.7**	**26.4**	**0.1**	**0.3**	**24.1**	0.7	**0.1**
ODENet					25.2	0	**0**	0.5	**8.5**	29.8	**0.8**	0	0.2	0.6	1.8	0	0.1	18	**0.9**	0

Table 2. Mean $\mathcal{A}_c^{\text{rel}}$ (%) at changes in corruption severity for MNIST. At fixed $(c, s) \in C$, each block (in green) contains results for cleanly trained ResNet (upper-left), ODENet (lower-left) and their noisy counterparts (upper-right, lower-right) The listed corruptions are as in Table 1. Performance shifts are colored in red. Corruptions where noisy training is not beneficial are colored in blue.

Sev.	1		2		3		4		5		6		7		8		9		10		11		12		13		14		15	
1	99.7	99.9	99.8	99.2	98.8	99.8	79.9	81.7	35.9	71.8	96.9	93.0	99.4	99.6	98.1	99.4	97.9	99.6	58.9	85.3	99.7	99.8	97.5	97.8	67.2	91.3	98.8	99.5	99.8	99.9
	99.9	**99.9**	**99.9**	**99.9**	**99.4**	**99.9**	**82.7**	**91.2**	**57.3**	**87.1**	**98.1**	**98.6**	**99.8**	**99.8**	**99.5**	**99.7**	**99.3**	**99.9**	**93.4**	**95.2**	**99.9**	**99.9**	**99.5**	**99.6**	**77.9**	**95.9**	**98.8**	**99.7**	**99.9**	**99.9**
2	99.4	99.9	99.6	99.9	96.3	99.6	**54.0**	51.4	33.1	68.8	83.0	75.5	99.1	99.6	91.5	98.3	84.0	98.8	45.8	74.4	99.4	99.7	91.7	93.3	37.4	70.7	98.6	99.5	99.7	99.9
	99.8	**99.9**	**99.8**	**99.9**	**98.1**	**99.7**	41.5	**58.9**	**50.3**	**84.5**	**88.3**	**92.1**	**99.7**	**99.7**	**98.0**	**99.5**	**94.6**	**99.7**	**86.6**	**85.5**	**99.8**	**99.9**	**99.0**	**99.3**	**50.7**	**82.9**	**99.1**	**99.6**	**99.9**	**99.9**
3	98.6	99.7	99.4	99.8	91.4	99.5	**16.1**	**18.4**	13.0	23.6	47.6	49.2	98.7	99.4	82.8	88.9	69.7	96.7	34.7	55.3	97.9	99.5	77.7	83.0	21.6	41.1	91.6	97.6	99.7	99.9
	99.5	**99.9**	**99.6**	**99.8**	**94.0**	**99.7**	7.6	11.1	**16.4**	**30.4**	**61.3**	**65.4**	**99.5**	**99.7**	**93.9**	**96.1**	**89.4**	**99.6**	**75.8**	**64.4**	**99.4**	**99.9**	**95.7**	**98.3**	**29.0**	**56.1**	**95.8**	**98.7**	**99.8**	**99.8**
4	93.7	99.6	98.1	99.4	68.1	98.7	**11.1**	**13.7**	12.6	20.7	24.7	29.7	98.0	99.2	71.3	79.2	68.5	96.8	30.6	50.2	93.0	99.2	**50.9**	59.7	16.9	26.9	64.2	84.6	99.6	99.8
	93.8	**99.7**	**98.6**	**99.6**	**77.1**	**99.4**	8.9	3.5	**15.2**	**25.4**	**35.0**	**41.5**	**99.3**	**99.5**	**98.5**	**91.1**	**89.4**	**99.5**	**69.7**	**58.8**	**97.4**	**99.8**	44.8	**95.0**	**21.2**	**37.0**	**79.2**	**91.6**	**99.7**	**99.9**
5	67.3	98.9	94.9	98.7	39.1	95.9	**9.9**	**11.2**	12.3	18.3	19.6	24.2	96.4	98.8	67.0	84.1	60.6	94.4	24.4	36.9	74.1	98.7	**25.9**	33.7	14.4	17.1	61.9	78.1	99.5	99.6
	76.2	**99.2**	**96.4**	**99.3**	**52.8**	**97.7**	9.6	4.9	**14.2**	**19.5**	**25.0**	**34.6**	**99.0**	**99.2**	**87.5**	**94.3**	**86.7**	**99.4**	**52.2**	**43.3**	**93.2**	**99.7**	15.2	**86.6**	**16.9**	**23.8**	**72.1**	**84.0**	**99.7**	**99.8**

two methods: *clean training* is done on clean-only images; *noisy training* is conducted on a random combination of 50% of clean images and 50% of images added Gaussian noise with randomly chosen $\sigma \in \{50, 75, 100\}$. We train each model on three different random seeds, each trained model is then tested on 3 runs of corruption simulations and only report the mean of the resulting 9 tests in Tables 1 & 2. We report the mean of the 3 models clean accuracy $\mathcal{A}_{\text{cl}}$ used to save each model's parameters at which the rest of the tests are conducted.

Results: Table 1 shows that ODENet is consistently more robust than ResNet and that cleanly trained ODENet has less necessity of data augmentation to achieve good performances than ResNet do, as seen in the last lines of Table 1, make us conclude that they are naturally more robust than ResNet. This experimental result is compatible with those appearing in the test-time adaptive models literature (Sun et al., 2020; Wang et al., 2021). In light of the study in (Gilmer et al., 2019) relating adversarial attacks as naturally appearing in the scope of common corruptions, this result can also be thought as a sanity check for determining that adversarial robustness for NODEs, contrary to what is hypothesized in (Huang et al., 2022), might *not* come from obfuscated gradients. This fact seems to be further confirmed in (Chu et al., 2022) although we have some reserves on their argument: increasing the time horizon of a NODE should, in our opinion, rather be linked to the model's resilience, roughly seen as the speed of convergence *after* an input perturbation, while robustness criteria usually focuses on the distances and positions of inputs incurring on invariance of a model's prediction. At increasing severity, as shown in Table 2, notice that, while ODENet is more robust than ResNet on most corruptions, their decay of robustness is bigger, shifting on some of the corruptions to ResNets as the best model. Nonetheless, the same shift occur at higher severity levels with noisy trained ODENets. Namely, for defocus_blur, on clean train mode, the shift was done at level 3 while at noisy train mode it was only done at level 4. Analogously, for contrast corruption, the shift at severity 4 on clean train mode was never reached on noisy train mode. This sheds evidence to the fact that noisy training for ODENets acts as a *robustness locus widening* i.e. that the robustness neighborhood of data points x become bigger with data augmentation. Notice that these robustness neighborhoods are *threat-model free*: they do not depend on the choice of a norm ball as is commonly considered on gradient-based defenses. Finally, noisy training made ResNet more vulnerable to corruptions 2 and 6 at severity 1

but this vulnerability got corrected at severity 3. This may suggest that partial information on the trade-off between accuracy and robustness may be captured by a notion of model's deterioration *resilience* whose rigorous study will be included in our upcoming extended study.

Acknowledgements

The authors thank to each other for the fruitful conversations that led to the project for which this paper consists on a preliminary work. We would like to thank the reviewers for their comments. This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute.

References

Bottou, L. and Bousquet, O. The Tradeoffs of Large Scale Learning. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/file/0d3180d672e08b4c5312dcdafdf6ef36-Paper.pdf.

Chaitin-Chatelin, F. and Frayssé, V. *Lectures on finite precision computations*. SIAM, 1996.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. In *Neural Information Processing Systems (NeurIPS)*, 2019. URL http://arxiv.org/abs/1806.07366.

Chu, H., Wei, S., Lu, Q., and Zhao, Y. Improving Neural ODEs via Knowledge Distillation. *arXiv preprint arXiv:2203.05103*, 2022.

Croce, F., Gowal, S., Brunner, T., Shelhamer, E., Hein, M., and Cemgil, T. Evaluating the Adversarial Robustness of Adaptive Test-time Defenses. *arXiv preprint arXiv:2202.13711*, 2022.

Djeumou, F., Neary, C., Goubault, E., Putot, S., and Topcu, U. Taylor-Lagrange Neural Ordinary Differential Equations: Toward Fast Training and Evaluation of Neural ODEs. *arXiv preprint arXiv:2201.05715*, 2022.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented Neural ODEs, 2019.

Fermanian, A., Marion, P., Vert, J.-P., and Biau, G. Framing RNN as a Kernel Method: A Neural ODE Approach. *Advances in Neural Information Processing Systems*, 34, 2021.

Gilmer, J., Ford, N., Carlini, N., and Cubuk, E. Adversarial Examples are a Natural Consequence of Test Error in Noise. In *International Conference on Machine Learning*, pp. 2280–2289. PMLR, 2019.

Gonzalez, M., Defourneau, T., Hajri, H., and Petreczky, M. Realization Theory Of Recurrent Neural ODEs Using Polynomial System Embeddings. *arXiv preprint arXiv:2205.11989*, 2022.

Hendrycks, D. and Dietterich, T. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. The Many Faces Of Robustness: A Critical Analysis Of Out-Of-Distribution Generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021.

Huang, Y., Yu, Y., Zhang, H., Ma, Y., and Yao, Y. Adversarial Robustness of Stabilized Neural ODE Might be from Obfuscated Gradients. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145 of *Proceedings of Machine Learning Research*, pp. 497–515. PMLR, 2022.

Ivan, Aaron, and Yue, Y. LyaNet: A Lyapunov framework for training neural ODEs. *arXiv pre-print server*, 2022. doi: Nonearxiv:2202.02526.

Kang, Q., Song, Y., Ding, Q., and Tay, W. P. Stable Neural ODE with Lyapunov-Stable Equilibrium Points for Defending Against Adversarial Attacks. *Advances in Neural Information Processing Systems*, 34, 2021.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kidger, P. *On Neural Differential Equations*. PhD thesis, Oxford University, 2022.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B., Haque, I., Beery, S. M., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. WILDS: A Benchmark of in-the-Wild Distribution Shifts. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5637–5664. PMLR, 18–24 Jul 2021.

Krishnapriyan, A. S., Queiruga, A. F., Erichson, N. B., and Mahoney, M. W. Learning Continuous Models for Continuous Physics. *arXiv preprint arXiv:2202.08494*, 2022.

Li, P., Yi, J., Zhou, B., and Zhang, L. Improving the Robustness of Deep Neural Networks via Adversarial Training with Triplet Loss. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2909–2915. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/403. URL https://doi.org/10.24963/ijcai.2019/403.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021.

Massaroli, S., Poli, M., Park, J., Yamashita, A., and Asama, H. Dissecting Neural ODEs. *arXiv preprint arXiv:2002.08071*, 2020.

Matsubara, T., Miyatake, Y., and Yaguchi, T. Symplectic Adjoint Method for Exact Gradient of Neural ODE with Minimal Memory. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=46J_l-cpc1W.

Mu, N. and Gilmer, J. MNIST-C: A Robustness Benchmark for Computer Vision. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

Ott, K., Katiyar, P., Hennig, P., and Tiemann, M. ResNet After All: Neural ODEs and Their Numerical Solution. In *International Conference on Learning Representations*, 2021.

Pal, A., Edelman, A., and Rackauckas, C. Mixing Implicit and Explicit Deep Learning with Skip DEQs and Infinite Time Neural ODEs (Continuous DEQs). *arXiv preprint arXiv:2201.12240*, 2022.

Queiruga, A. F., Erichson, N. B., Hodgkinson, L., and Mahoney, M. W. Stateful ODE-Nets using Basis Function Expansions. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021.

Salehi, M., Mirzaei, H., Hendrycks, D., Li, Y., Rohban, M. H., and Sabokrou, M. A Unified Survey on Anomaly, Novelty, Open-Set, and Out-of-Distribution Detection: Solutions and Future Challenges. *arXiv preprint arXiv:2110.14051*, 2021.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 13–18 Jul 2020.

Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., and Schmidt, L. Measuring Robustness to Natural Distribution Shifts in Image Classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Xu, W., Chen, R. T., Li, X., and Duvenaud, D. Infinitely Deep Bayesian Neural Networks with Stochastic Differential Equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 721–738. PMLR, 2022.

Yan, H., Du, J., Tan, V., and Feng, J. On Robustness of Neural Ordinary Differential Equations. In *International Conference on Learning Representations*, 2020.

Zhong, Y. D., Dey, B., and Chakraborty, A. Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxmb1rKDS.

A. Detailed experiment results

In this Appendix we give further details on the application of our methodology to the presented experiments: the chosen corrupt simulation algorithm is shown to be non-trivial along different tested datasets (no miss-classified clean images incur into well-classified corrupted counterparts); we do not include corruptions in our train or validation sets, networks share the same h_x and h_y modules; weight-tied ResNet blocks correspond to discretized NODE blocks. Since our model is not DS-destined, we do not conduct a numerical convergence test for the chosen Euler method.

A.1. Model specifications

All our models share the same FE and FCC modules and the RM modules consist on the same layers to which one either applies a residual connection (for ResNet) or the *odeint* function (for ODENet). In order to favor our ability to compare ResNets and ODENets, we fix the Euler method as our ODE numerical solver at time range $[0, 1]$ with 0.1 time steps which corresponds to ten weight-tied residual blocks. Finally, we use Group Normalization (GN) instead of Batch Normalization (BN) to ensure that the dynamics of the RM module truly correspond to an autonomous NODE.

Table 3. The FE h_x and FCC h_y modules are identical for all our ResNet and ODENet models for MNIST, SVHN and CIFAR. The arguments of Conv2d are in order: the input channel, output channel, kernel size, stride and padding. Conv2dTime ensures time-dependence of the convolution component. The two arguments of the Linear layer represents the input dimension and the output dimension of this fully-connected layer.

Common Modules	Sequenced Layers
h_x	Conv2d(1, 64, 3, 1) + GN + ReLU Conv2d(64, 64, 4, 2) + GN + ReLU
h_y	AdaptiveAvgPool2d + Linear(64,10)
RM	Internal Layers (input x)
ResNet	out:=(Conv2d(64, 64, 3, 1,1) + GN + ReLU Conv2d(64, 64, 3, 1,1) + GN + ReLU) out +x
ODENet	f=(Conv2dTime(64+1, 64, 3, 1,1) + GN + ReLU +Conv2dTime(64+1, 64, 3, 1,1) + GN + ReLU) odeint(f, x, [0,1], Δt=0.1, Euler)

We train all our models for 100 epochs, learning rate 0.001, milestones [30, 60, 90]; decay 0.0005, L_2-penalty 0.2. Both models have around 142k parameters.

While, in (1), f formalizes a single layer's dynamics, it usually is taken in practice to be a composition of explicit functions that we pass to the ODE solver (a block). Using BN as a block component holds mini-batch information, which not only cannot be formalized as an autonomous ODE but may lead to gradient explosion at back-propagation. When comparing ODENet and ResNet blocks (without BN), one must ensure that each composite function for a NODE block is both stateful (has an implicit dependence on the integration time) and input-autonomous (does not present dependence or has leaked information of the rest of the samples) either at training, validation or testing. We use the ReLU function for practical purposes (increased performance) after checking that models trained with fully differentiable functions present the same behavior. We avoid taking into account in our analysis neither neural SDEs, which can genuinely be seen as continuous-time analog of noise injection robustifying methods, but whose inference is not deterministic and which do not take into account stateful BN layers, as they induce depth-varying architectures which do not have a clear discrete counterpart upon which one could establish a comparative analysis. Choosing the good basis function method for the latter to achieve competitive results (Queiruga et al., 2021) was a long effort, according to the authors, and hasn't yet come close to state-of-the-art clean accuracy performances. In addition, BN which has been shown to be crucial to achieve state-of-the-art robustness performances, has also been shown to be a source of adversarial vulnerability and it is unclear if their stateful counterpart from (Queiruga et al., 2021) will present or not this same behavior. Finally, our discretized NODE block is formulated in terms of weight-tied ResNet blocks and matches the function to which the residual skip connection is added with the one sent to the ODE solver and does not match to the total length of convolution blocks present in the architecture. For instance, appending 10 independent identical residual convolutional blocks does not correspond to passing a single convolutional block through a ODE solved with fixed 10 Euler steps as the state space of the NODE is controlled by only one set of convolutional block parameters.

A.2. Further Simulation Corrupted Dataset Experiments

We use several datasets and their synthetic corruptions by using the same simulation algorithm and selecting corruptions that make sense on all datasets. Noisy training is done with randomly added noise with $\sigma \in \{10, 15, 25\}$ for the SVHN dataset, and with $\sigma \in \{10, 15, 20\}$ for the CIFAR10 dataset. Our experiment's relative perturbed accuracy is given in Tables 4 and 5 for SVHN[6].

Table 4. Mean $\mathcal{A}_c^{\text{rel}}$ (%) on corrupted SVHN images for ResNet and ODENet. The listed corruptions are as in Table 1. The last block computes the improvement on performance for each model induced by noisy training w.r.t. clean training. Corruptions where noisy training is not beneficial are colored in blue and a $> 5\%$ difference of model's performance is colored in orange.

Model	Training	$\mathcal{A}_{\text{cl}}$	$\mathcal{A}_c^{\text{rel}}$ on Noise			$\mathcal{A}_c^{\text{rel}}$ on Common Corruptions (severity 1)															
			Gaussian (σ)			Noise			Blur				Weather				Digital			
			10	15	25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ResNet	Clean	94.3	47.2	29.3	20.1	84.7	83.2	81.9	93.7	93.8	95.3	98.9	77.4	88.3	50.7	99.2	97.5	66.6	97.7	97.2
ODENet		96.3	**63.7**	**44.0**	**31.7**	**92.3**	**91.4**	**91.1**	**95.3**	**96.3**	**96.8**	**99.3**	**83.4**	**91.8**	**59.3**	**99.6**	**98.8**	**84.7**	**98.9**	**98.1**
ResNet	Noisy	94.2	-	-	93.2	95.2	95.2	93.9	93.4	94.8	95.1	98.9	81.9	91.3	45.8	99.1	97.4	87.5	98.3	97.9
ODENet		96.2	-	-	96.2	**97.5**	**97.3**	**97.0**	**94.6**	**96.4**	**96.5**	**99.3**	**86.7**	**94.6**	**53.2**	**99.6**	**98.1**	**93.0**	**99.1**	**98.8**
ResNet	Noisy $\mathcal{A}_c^{\text{rel}}$- clean $\mathcal{A}_c^{\text{rel}}$				**73.1**	**10.5**	**12**	**12**	**-0.3**	**1**	**-0.2**	**0**	**4.5**	**3**	**-4.9**	**0.1**	**-0.1**	**20.9**	**0.6**	0.7
ODENet					64.5	5.2	5.9	5.9	-0.7	0.1	-0.3	0	3.3	2.8	-6.1	0	-0.7	8.3	0.2	0.7

Table 5. Mean $\mathcal{A}_c^{\text{rel}}$ (%) at changes in severity. The listed corruptions are as in Table 1. Red color means a shift of best model's accuracy w.r.t. previous severity value. Corruptions where noisy training is not beneficial are colored in blue.

Model	Training	Sev.	Noise			Blur				Weather				Digital			
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ResNet	Clean	1	84.7	83.2	81.9	93.7	93.8	95.3	98.9	77.4	88.3	50.7	99.2	97.5	66.6	97.7	97.2
ODENet			**92.3**	**91.4**	**91.1**	**95.3**	**96.3**	**96.8**	**99.3**	**83.4**	**91.8**	**59.3**	**99.6**	**98.8**	**84.7**	**98.9**	**98.1**
ResNet		2	69.3	64.8	64.8	84.1	81.2	87.0	98.7	57.3	73.6	39.8	97.8	95.9	42.6	97.4	95.9
ODENet			**83.0**	**80.8**	**80.2**	**86.9**	**87.1**	**90.6**	**99.1**	**68.1**	**80.5**	**48.5**	**98.9**	**98.0**	**63.0**	**98.9**	**97.2**
ResNet		3	48.6	46.1	51.9	**46.2**	38.2	70.2	98.5	64.1	63.2	26.5	95.5	91.9	24.2	89.1	94.7
ODENet			**67.1**	**65.8**	**70.8**	45.6	**45.7**	**76.7**	**99.0**	**71.0**	**71.9**	**33.2**	**97.4**	**96.4**	**35.8**	**93.9**	**96.3**
ResNet		4	32.2	26.6	31.7	**28.0**	30.4	54.6	98.3	54.4	60.9	19.9	91.9	75.5	18.7	68.0	90.5
ODENet			**50.1**	**44.5**	**51.2**	21.2	**35.6**	**61.3**	**98.7**	**62.4**	**70.1**	**25.1**	**94.8**	**90.7**	**24.7**	**75.0**	**93.3**
ResNet		5	18.9	18.7	19.6	**21.3**	24.7	47.3	97.7	52.2	55.0	15.0	85.3	49.2	15.4	59.3	83.5
ODENet			**33.2**	**33.4**	**35.1**	13.6	**24.9**	**53.8**	**98.3**	**61.5**	**64.8**	**18.8**	**89.9**	**78.9**	**17.7**	**65.7**	**87.9**
ResNet	Noisy	1	95.2	95.2	93.9	93.4	94.8	95.1	98.9	81.9	91.3	45.8	99.1	97.4	87.5	98.3	97.9
ODENet			**97.5**	**97.3**	**97.0**	**94.6**	**96.4**	**96.5**	**99.3**	**86.7**	**94.6**	**53.2**	**99.6**	**98.1**	**93.0**	**99.1**	**98.8**
ResNet		2	90.1	89.1	88.0	83.7	85.3	86.4	98.7	64.5	79.9	36.0	98.0	96.0	74.2	98.3	97.0
ODENet			**94.8**	**94.0**	**93.7**	**85.4**	**90.0**	**89.8**	**99.1**	**73.8**	**86.1**	**42.4**	**99.0**	**96.7**	**84.8**	**99.0**	**98.1**
ResNet		3	80.3	79.6	82.0	44.4	44.2	70.2	98.6	66.5	71.6	24.4	95.9	92.3	51.4	94.3	96.2
ODENet			**88.7**	**88.6**	**90.2**	**45.6**	**54.4**	**76.1**	**98.9**	**72.6**	**80.2**	**29.6**	**97.5**	**93.6**	**66.7**	**96.7**	**97.4**
ResNet		4	66.0	61.7	66.4	**28.3**	34.4	54.0	98.3	56.6	70.2	19.3	92.4	76.4	37.8	82.6	93.2
ODENet			**78.9**	**75.5**	**79.9**	26.6	**43.2**	**60.0**	**98.7**	**63.0**	**78.9**	**25.0**	**94.8**	**83.5**	**51.0**	**88.0**	**95.1**
ResNet		5	47.6	48.9	50.2	**21.3**	25.4	47.3	97.7	56.9	65.0	15.3	86.3	47.3	26.5	74.8	87.7
ODENet			**63.1**	**65.0**	**66.4**	17.4	**30.0**	**52.0**	**98.3**	**65.1**	**73.9**	**19.3**	**89.9**	**66.2**	**34.0**	**80.7**	**90.9**

Results: ODENet is consistently more robust than ResNet and the latter benefited more than ODENet from noisy training. This confirms our MNIST conclusions on the natural robustness of ODENet. Notice that noisy training makes the models slightly more vulnerable for some corruptions. At severity changes, noisy training acts here again as a robustness locus widening. Interestingly, for those corruptions where noisy training made the model more vulnerable at severity 1, noisy trained models see their vulnerability remain only on half of those corruptions at severity 5. Finally, as shown in orange, it seems that noisy training makes ODENet more resilient to corruption deterioration than it does to ResNet.

[6]The results for CIFAR showing no novel or different behaviour than the one conducted for SVHN other than an overall drop in accuracy, we do not present them in this preliminary report.

A.3. Further thoughts on corruption simulators and their metrics

Baseline Normalization: Usual corruption metrics such as the mCE^f and the relative mCE^f for a model f are computed with respect to a baseline (e.g. AlexNet) error in order to normalize it over those corruptions that are known to be particularly challenging. For $(c, s) \in C$, they are the average over all 15 corruptions labels c of the clean and corrupted top-1 error rates (averaged first across all 5 severity levels):

$$\mathrm{CE}_c^f = \left(\sum_{s=1}^{5} E_{s,c}^f\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\text{AlexNet}}\right), \qquad \mathrm{rCE}_c^f = \left(\sum_{s=1}^{5} E_{s,c}^f - E_{\text{clean}}^f\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\text{AlexNet}} - E_{\text{clean}}^{\text{AlexNet}}\right) \tag{2}$$

We do not normalize our metrics for two reasons. First, the mCE^f was proposed as an attempt to unify robustness benchmarking under a unique number across many models, which we do not do here. But most importantly, it has been shown that feature aggregating networks and deeper nets markedly enhance robustness. Thus, as NODEs consist on a paradigm shift of the notion of depth, which becomes adaptive even while testing, we find that testing such network against a baseline fixed depth network such as AlexNet could be harmful for our comparative analysis. This is somewhat concordant with the last recommendation in (Croce et al., 2022) for generating adversarial attacks for adaptive test-time defenses. We do not average our metrics across severity levels to analyse their behaviour at increasing severity.

Corruption simulators: Instead of testing our models on static corrupted datasets (MNIST-C, CIFAR10-C, ImageNet-C), we run the exact same corruption simulator[7] on the datasets of all our experiments (and which is the same one used by the authors that proposed the above-mentioned static corrupted datasets). For simplicity, corruptions that were specially tailored for MNIST (such as zig-zag or canny edges) that only make sense to be conducted on that dataset will be left out from our comparative study, as well as fully formalizable corruptions (such as rotations, translations..) that one can use as auxiliary data augmentation techniques such as adversarial and S&P noise augmentations. This should be taken into account as a partial reproducibility issue: while the performance of the considered models should decrease when tested on compressed JPEG corrupted datasets such as ImageNet-C, the comparative results conducted in this work do not show any qualitative distinction (although the overall model's accuracies decrease). Also, our objective is not to propose an architecture capable of achieving state-of-the-art robust performances in either of the mentioned datasets. Our choice to fix a common corruption simulator for different datasets is somehow a model-driven choice. It has been shown that classification error patterns between robust models and those coming from human perception are fundamentally different. As such, focusing on understanding a model's behavior around simulated corruptions can be improved by fixing one simulator and generating corruptions among different datasets. This allows to release some part of the randomness of a generated simulation and prevents different data augmentation techniques to guess a corruption simulator's parameters, which in a sense can be seen as information leakage. By using the corruption simulator as a white-box component of our generated corruption dataset we hope this will promote better model's transferability to some degree.

[7] Available at https://github.com/bethgelab/imagecorruptions.