On the Robustness to Adversarial Examples of Neural ODE Image Classifiers

Fabio Carrara
ISTI CNR, Pisa, Italy
fabio.carrara@isti.cnr.it

Roberto Caldelli
CNIT, Florence, Italy
roberto.caldelli@unifi.it

Fabrizio Falchi
ISTI CNR, Pisa, Italy
fabrizio.falchi@isti.cnr.it

Giuseppe Amato
ISTI CNR, Pisa, Italy
giuseppe.amato@isti.cnr.it

***Abstract*—The vulnerability of deep neural networks to adversarial attacks currently represents one of the most challenging open problems in the deep learning field. The NeurIPS 2018 work that obtained the best paper award proposed a new paradigm for defining deep neural networks with continuous internal activations. In this kind of networks, dubbed Neural ODE Networks, a continuous hidden state can be defined via parametric ordinary differential equations, and its dynamics can be adjusted to build representations for a given task, such as image classification. In this paper, we analyze the robustness of image classifiers implemented as ODE Nets to adversarial attacks and compare it to standard deep models. We show that Neural ODE are natively more robust to adversarial attacks with respect to state-of-the-art residual networks, and some of their intrinsic properties, such as adaptive computation cost, open new directions to further increase the robustness of deep-learned models. Moreover, thanks to the continuity of the hidden state, we are able to follow the perturbation injected by manipulated inputs and pinpoint the part of the internal dynamics that is most responsible for the misclassification.**

I. Introduction

In a broad set of perceptual tasks, state-of-the-art deep-learned models are usually comprised of a set of discrete transformations, known as layers, whose parameters are optimized in an end-to-end fashion via gradient-based methods. Thanks to the hierarchical architecture, the modelling stage is divided among a series of transformations each extracting features from its input and providing higher-level abstractions downstream. When trained with enough data, their ability to model complex mappings is remarkable and produced new state-of-the-art results in multiple fields, such as automatic guided vehicles, speech recognition, text classification, anomaly detection, decision making and even human-like face/people generation in images and videos.

However, the success of deep-learnt models is accompanied by a major downside, that is the vulnerability to *adversarial examples* — maliciously manipulated inputs that appear legit but fool the model to produce a wrong output [1], [2], [3]. Due to the complexity of data and models (often required to learn complex input-output mapping,) adversarial examples can be usually found for most of neural-network-based models in efficient ways [4], [5], [6], and despite studies in this field [7], [8], a universal approach to produce adversarial-robust deep models is still missing.

A complementary way pursued by the research community to solve the adversarial problem is the investigation of the effects of adversarial examples on trained models [9], [10]. The characterization of these effects often gives enough insight to detect and distinguish adversarial examples from authentic inputs and thus led to several proposals in adversarial detection [11], [12], [13], [14], [15].

In this paper, our goal is to get insights into a novel recently proposed deep-learning architecture based on ordinary differential equations (ODEs) — the *Neural ODE Network* [16] — when undergone adversarial attacks. Differently from standard neural networks, ODE-defined networks are not comprised of a set of discrete transformations, but instead define their processing pipeline as a continuous transformation of the input towards the output whose derivative is parameterized and trainable with standard gradient-descent methods. The continuous transformation is computed by an adaptive ODE solver which gives the model additional useful properties, such as reversibility, O(1)-memory cost, sample-wise adaptive computation, and a trade-off between computational speed and accuracy tunable at inference time. This novel formulation (and all the benefits that derive from it) makes ODE nets an interesting and promising new tool in the deep learning field: thanks to their generality, they can potentially replace current models (e.g. residual networks in image processing) and enable new models in novel applications such as continuous-time modelling.

However, Neural ODE networks are still differentiable models, and thus, the same adversarial attacks used in standard neural networks are implementable and applicable. This allows us to apply the same efficient adversarial samples crafting algorithm used in standard neural networks and study the robustness of ODE nets to adversarial attacks in the context of image classification.

The main contributions of the present work are the following:

- we analyzed the behavior of ODE Nets against adversarial examples (that we believe it has not been tested yet) in the context of image classification; we compared three architectures — a standard residual network and two ODE-based models — in terms of robustness to adversarial attacks on the MNIST and CIFAR-10 datasets;

Preprint submitted to WIFS'2019.
Final version to appear on IEEE.

- we studied how the robustness to adversarial attacks varies with specific properties of ODE networks, such as the precision-cost trade-off offered by adaptive ODE solvers via a tunable tolerance parameter; our findings showed that lowering the solver precision leads to a substantial decrease in the attack success rate while degrading only marginally the classification performance;
- thanks to the peculiarity of ODE Nets (i.e. their continuity), we observed and measured the effect of attacks on the evolution of the internal state of the network as the adversarial perturbation propagates through the continuous input-output mapping.

After this introductory section, the paper is organized as follows: Section II presents some related work, Section III briefly describes the working of ODE-Nets, while Section IV describes the whole operation context in terms of tested networks and attacks. Section V provides details on the experimental set-up[1], Section VI discusses achieved results, and Section VII draws main conclusions proposing, at the same time, some possible future directions.

II. Related Work

Adversarial examples represent one of the major challenges applicability and technological transfer of deep learning-based techniques. Thus, after the seminal work of Szegedy et al. [2] exploring adversarial examples for convolutional neural networks, the research community focused on this problem publishing several analyses [4], [9], [17] and organizing public challenges [18] about this phenomenon.

Several works provided efficient crafting algorithms for adversarial examples including the Fast Gradient Sign Method (FGSM) [4], Projected Gradient Descend (PGD) [19], and the one proposed by Carlini and Wagner [6]. Kurakin et al. [7] and Athalye et al. [20] showed that adversarial attacks to computer vision systems are possible and effective also in the physical world using respectively 2D or 3D objects with malicious textures.

Defensive actions against adversarial attacks have also been proposed. In this regard, the literature can be roughly divided in robustness improvement and adversarial detection techniques. The former aims at changing the model to be more robust and include techniques such as adversarial training [21] and model distillation [8]. The latter aims at detecting adversarial examples and thus discarding malicious predictions. Detection methods include statistical tests on the inputs [10], [11], adversarial perturbation removal [22], and auxiliary detection models [12], [13], [23].

Due to its peculiarity, ODE nets may differ from standard deep neural networks in their interaction with adversarial examples. To the best of our knowledge, we are the first analyzing the robustness of ODE nets to adversarial attacks.

[1]Code and resources to reproduce the experiments presented here are available at https://github.com/fabiocarrara/neural-ode-features

III. Neural ODE Networks: background

In this section, we provide the reader with a brief description of ODE Nets and their properties; for a full detailed description, see [16].

We refer as ODE Net to a parametric model including an *ODE block* — a computation block defined a parametric ordinary differential equation (ODE) whose solution provides the output result. Let $\mathbf{h}_0$ the block's input coinciding with the initial state at time t_0 of the following initial-state ODE

$$\begin{cases} \frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = f(\mathbf{h}(t), t, \theta) \\ \mathbf{h}(t_0) = \mathbf{h}_0 \end{cases} . \tag{1}$$

The function f, parameterized by θ, defines the continuous dynamic of the state $\mathbf{h}(t)$. In the context we are interested in (image classification), f is often implemented as a small convolutional neural network. The output of the block is the value $\mathbf{h}(t_1)$ of the state at a time $t_1 > t_0$ that can be computed by integrating the ODE

$$\mathbf{h}(t_1) = \mathbf{h}(t_0) + \int_{t_0}^{t_1} \frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t}\mathrm{d}t = \mathbf{h}(t_0) + \int_{t_0}^{t_1} f(\mathbf{h}(t), t, \theta)\mathrm{d}t\,. \tag{2}$$

The above integral can be computed with standard ODE solvers, such as Runge-Kutta or Multi-step methods [24]. Thus, the computation performed by the ODE block can be formalized as a call to a generic ODE solver

$$\mathbf{h}(t_1) = \mathrm{ODESolver}(f, \mathbf{h}(t_0), t_0, t_1, \theta)\,. \tag{3}$$

During the training phase, the gradients of the output $\mathbf{h}(t_1)$ with respect to the input $\mathbf{h}(t_0)$ and the parameter θ can be obtained using the adjoint sensitivity method [25]. This consists of solving an additional ODE backward in time and thus invoking again the ODE solver in the backward pass. Once the gradient is obtained, standard gradient-based optimization can be applied.

ODE Nets benefit from several properties derived by their formulation or inherited from ODE solvers, including a) *reversibility*, as the evolution can be computed in forward or backward in time, b) *O(1)-memory cost*, as intermediate states do not need to be stored when solving ODEs, c) *adaptive computation*, as adaptive ODE solvers can adjust the integration step size for each input, d) *accuracy-efficiency trade-off tunable at inference time*, that can be controlled by tuning the tolerance of adaptive ODE solvers.

IV. Robustness of ODE Nets

We are interested in implementing image classifiers with ODE Nets and compare their robustness to adversarial attacks with respect to standard neural networks. We tested a total of three architectures: a standard residual network model (*RES*) used as a baseline, a mixed model (*MIX*) comprised of some residual layers and an ODE block, and an ODE-dominated model (*ODE-Only Net, OON*) whose computation is mostly comprised of a single ODE block. We train all models on public datasets for image classification, perform adversarial attacks on respective test sets, and measure the attack success rate.

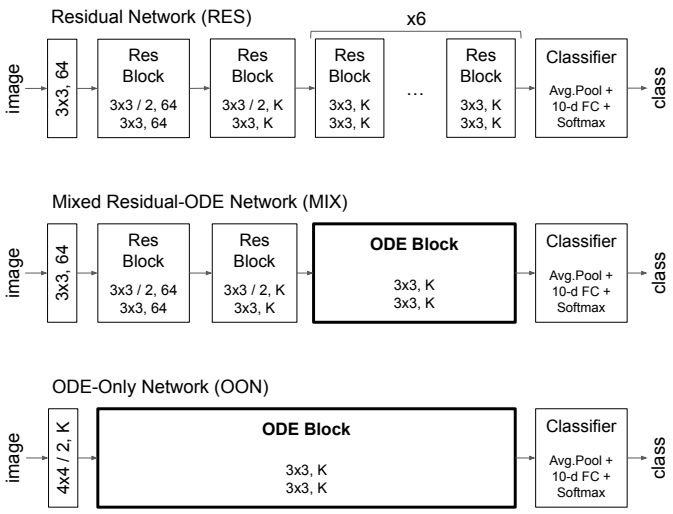

Fig. 1: Architectures of the tested models. Convolutional layers are written in the format *kernel width* $\times$ *kernel height [/ stride], n. filters*; padding is always set to 1. For MNIST, $K = 64$, and for CIFAR-10, $K = 256$.

A. Tested Architectures

In this section, we provide details about the tested architectures (also depicted in Figure 1).

a) Residual Network (RES)*:* this is the convolutional residual neural network image classifier defined and used as a baseline by Chen et al. [16] in their comparisons with ODE-Nets. It is comprised of a 64-filter 3x3 convolutional layer and 8 residual blocks. Each residual block follows the standard formulation defined in [26], with the only difference that Group Normalization [27] is used instead of batch normalization. The sequence of layers comprising a residual block is GN-ReLU-Conv-GN-ReLU-Conv-GN where GN stands for a Group Normalization with 32 groups, and Conv is a 3x3 convolutional layer. The first two blocks downsample their input by a factor of 2 using a stride of 2 (also employed in 1x1 convolutions in the shortcut connections), while the subsequent blocks maintain the input dimensionality. The first block employs 64-filters convolutions while subsequent blocks employ K-filter convolutions where K varies with the specific dataset. The final classification step is implemented with a global average-pooling operation followed by a single fully-connected layer with softmax activation.

b) Mixed ResNet-ODE Network (MIX)*:* this is the ODE Net architecture defined by Chen et al. [16]. The model is comprised of the same first convolutional layer and the first two residual blocks of the above-mentioned residual network plus an ODE block. The ODE function f defining the dynamics of the internal state is implemented by a residual block defined as above, with the difference that the current evolution time t is also considered as input and concatenated as a constant feature map to the inputs of the convolutions in the block. The input and the output of the ODE block are arbitrarily mapped respectively to time $t = 0$ and $t = 1$, i.e. $\mathbf{h}(0)$ is the input coming from the residual blocks, and $\mathbf{h}(1)$ the output of the ODE block. The output of the ODE block is average-pooled and followed by a fully-connected layer with softmax activation as in the residual net.

c) ODE-Only Network (OON)*:* in the previously described mixed architecture, it is not clear how the feature extraction process is distributed among standard and ODE blocks. To ensure that most of the image processing and feature extraction process happens in the ODE block, we define and test also an architecture mostly comprised of a single ODE block. In this model, the input is fed to a minimal pre-processing stage comprised of a single K-filter 4x4 convolutional layer with no activation function that linearly maps the image in an adequate state space. The output of this step is then fed to a single ODE block which is responsible for the whole feature extraction chain. Finally, the same classification stage as above-mentioned architectures follows.

B. Measuring the Robustness to Attacks

In this section, we describe the methodology used to perform attacks and measure the robustness of the trained models. Among the available adversarial attacks, we consider the untargeted Projected Gradient Descent (PGD) algorithm (also known as Basic Iterative Method or iterative-FGSM) [7], [19], a strong and widely-used gradient-based multi-step attack. Starting from an original sample, the PGD algorithm iteratively searches for an adversarial sample by taking small steps in the direction of the gradient of the loss function with respect to the input. Let $\mathbf{x}$ a sample in the input space, $\mathcal{L}(\mathbf{x})$ the classifier loss function, $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x})$ its gradient with respect to the input, $d(\mathbf{x}_1, \mathbf{x}_2)$ a distance function in the sample space, η the step size, and ε the maximum magnitude of the adversarial perturbation. Formally, PGD performs

$$\mathbf{x}^0_{\text{adv}} = \mathbf{x}, \quad \mathbf{x}^i_{\text{adv}} = \text{Proj}_\varepsilon(\mathbf{x}^{i-1}_{\text{adv}} + \eta\, \text{Norm}(\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^{i-1}_{\text{adv}}))\,. \quad (4)$$

$\text{Proj}_\varepsilon(\cdot)$ is a function that projects a sample $\tilde{\mathbf{x}}$ having a distance $d(\tilde{\mathbf{x}}, \mathbf{x}) > \varepsilon$ on the L_p ball centered on the original sample $\mathbf{x}$ with radius ε (common choices for d are the L_2 or L_∞ norms). The $\text{Norm}(\cdot)$ function ensures the gradient has unitary norm: it is implemented as $\text{sign}(\cdot)$ when using $d = L_\infty$ and as L_2 normalization ($\text{Norm}(\mathbf{x}) = \frac{\mathbf{x}}{|\mathbf{x}|_2}$) when $d = L_2$ is used.

We quantify the overall robustness to adversarial examples of a classifier by measuring the success rate of the attack under different configurations of the attack parameters. We consider an attack successful if the PGD algorithm is able to find an adversarial sample leading to a misclassification within the available budget, i.e. without exceeding the maximum number of iterations or the maximum perturbation norm ε). For seek of simplicity, we fixed some parameters dataset-wise (η and the maximum number of PGD iterations) while exploring configurations of other parameters (i.e. ε and d).

V. Experimental Setup

The following experimental set-up has been considered to evaluate the robustness to adversarial examples of the proposed three kinds of neural networks.

TABLE I: Performance (classification error % on MNIST and CIFAR-10 test sets) and complexity (number of trainable parameters) of the tested architectures: ResNet (RES), Mixed (MIX), and ODE-Only Net (OON). For architectures with ODE blocks, we show how the classification error varies with the ODE solver tolerance τ.

	MNIST			CIFAR-10		
τ	RES	MIX	OON	RES	MIX	OON
10^{-4}	0.4%	0.5%	0.5%	7.3%	7.8%	9.1%
10^{-3}		0.5%	0.5%		7.8%	9.2%
10^{-2}		0.5%	0.6%		7.9%	9.3%
10^{-1}		0.5%	0.8%		7.9%	10.6%
10^{0}		0.5%	1.2%		7.9%	11.3%
10^{1}		0.5%	1.5%		7.8%	11.5%
params	0.60M	0.22M	0.08M	7.92M	2.02M	1.20M

A. Datasets

We trained the models under analysis on two standard low-resolution image classification datasets: MNIST [28] and CIFAR-10 [29]. MNIST consists of 60,000 28x28 grayscale images of hand-written digits subdivided in train (50,000) and test (10,000) sets; it is the de facto standard baseline for novel machine learning algorithms and is the only dataset used in most research concerning ODE nets.

In addition to MNIST, we extended our analysis using also CIFAR-10 — a 10-class image classification dataset comprised of 60,000 32x32 RGB images of common objects subdivided in train (50,000) and test (10,000) sets.

B. Training Details

The number of filters K of convolutional networks in internal blocks is set to 64 for MNIST and 256 for CIFAR-10. For all models, we adopted the following hyperparameters and training procedures: dropout with 0.5 drop probability applied before the fully-connected classifier, the SGD optimizer with momentum of 0.9, weight decay of 10^{-4}, batch size of 128, and learning rate of 0.1 reduced by a factor 10 every time the error plateaus. For models containing an ODE block, we adopted the Dormand–Prince variant of the fifth-order Runge–Kutta ODE solver [30] in which the step size is adaptive and can controlled by a tolerance parameter τ (set to 10^{-3} in our experiments during the training phase). The specific value of τ indicates the maximum absolute and relative error (estimated using the difference between the fourth-order and the fifth-order solution) accepted when performing a step of integration; if the step error is higher than τ, the integration step is rejected and the step size decreased.

All the models obtained a classification performance comparable with current state of the art on those datasets. In Table I, we report the classification error (as percentage of misclassification) and the model complexity (as the number of trainable parameters in convolutions and fully-connected layers) for each model and dataset. For models with ODE blocks, Table I also shows how the classification performance changes with the tolerance of the ODE solver. A higher value of τ corresponds to a lower precision of the integration carried out by the ODE solver. We observed that this loss in precision has only marginal effects on the overall accuracy of the classifier (the worst degradation happens for the ODE-Only Network, that exhibit a 1-2% drop in accuracy) and also slightly decreases the computational cost of the forward pass.

C. Adversarial Attack Details

We employed the Foolbox toolkit [31] to perform adversarial attacks on PyTorch models. We attacked each model with iterative PGD using the test sets as source of original samples to perturb. We discarded from the analysis the images that are naturally misclassified by the models. For MNIST, we fixed the step size $\eta = 0.05$ and performed attacks with maximum perturbation $\varepsilon = 0.05$, 0.1, and 0.3; concerning the measure of the perturbation magnitude, we experimented with $d = L_2$ and L_∞. The same considerations apply for CIFAR-10, with the difference that we lowered the maximum perturbation allowed (and thus the step size) in order to capture the diversities, in terms of robustness, among the models. Thus, for CIFAR-10 we set $\eta = 0.01$ and $\varepsilon = 0.01$, 0.03, and 0.05 (attacks with higher values of ε always yield an adversarial example in all models). In all experiments, we set the maximum number of PGD iterations to 10; if an adversarial is found before the last iteration, we early stop the attack as soon as the misclassification happens.

VI. Results

Tables II and III report the attack success rate respectively on the MNIST and CIFAR-10 datasets for all the models and configuration tested.

In our experiments, ODE-based models (*MIX* and *OON*) consistently have a lower or equal attack success rate with respect to standard residual networks in the same attack context. This is particularly true for the ODE-only model when attacked with smaller perturbations: the attack success rate is roughly halved with respect to *RES* or *MIX*.

In both *OON* and *MIX* models, lowering the constraints on integration precision (i.e. increasing the solver tolerance) results in a decreasing probability of attack success. Increasing the solver tolerance τ to 1 decreases the attack success rate by roughly 40% in the best cases while increasing the classification error at most by roughly 2% in the worst case. For ease of comparison, we report in Table IV the classification error and attack success rate of the best configurations — the ones obtaining the best trade-off between the two quantities — for different values of τ. This phenomenon is probably due to the fact that allowing greater approximations in the feature extraction and classification process results in blurring the decision boundaries and thus attenuating the effects of malicious perturbations during the ODE integration.

To explore this hypothesis, we analyzed the continuous internal states of the ODE-Only Net when classifying a pristine or perturbed image. Specifically, let $\mathbf{h}(t)$ the trajectory of the continuous internal state caused by a pristine sample and $\mathbf{h}_{adv}(t)$ the one caused by the same but adversarially perturbed

TABLE II: Attack success rate on MNIST test set of *iterative PGD* (step size $\eta = 0.05$) applied to the tested architectures: ResNet (RES), Mixed (MIX), and ODE-Only Net (OON). ε indicates the maximum perturbation allowed and d the distance used to measure the perturbation magnitude.

	$\varepsilon = .05$, $d = L_2$			$\varepsilon = .1$, $d = L_2$			$\varepsilon = .3$, $d = L_2$		
tol	RES	MIX	OON	RES	MIX	OON	RES	MIX	OON
10^{-4}	.52	.51	.27	.99	.98	.87	1.	1.	1.
10^{-3}		.51	.19		.98	.75		1.	.99
10^{-2}		.50	.09		.98	.57		1.	.94
10^{-1}		.44	.08		.95	.54		1.	.92
10^{0}		.35	.06		.91	.46		1.	.96
	$\varepsilon = .05$, $d = L_\infty$			$\varepsilon = .1$, $d = L_\infty$			$\varepsilon = .3$, $d = L_\infty$		
tol	RES	MIX	OON	RES	MIX	OON	RES	MIX	OON
10^{-4}	.04	.04	.02	.32	.31	.12	1.	1.	.96
10^{-3}		.04	.02		.31	.08		1.	.87
10^{-2}		.03	.01		.30	.04		.99	.66
10^{-1}		.03	.01		.24	.05		.98	.64
10^{0}		.02	.02		.17	.04		.95	.65

TABLE III: Attack success rate on CIFAR-10 test set of *iterative PGD* (step size $\eta = 0.01$) applied to the tested architectures: ResNet (RES), Mixed (MIX), and ODE Net (OON). ε indicates the maximum perturbation allowed and d the distance used to measure the perturbation magnitude.

	$\varepsilon = .01$, $d = L_2$			$\varepsilon = .03$, $d = L_2$			$\varepsilon = .05$, $d = L_2$		
tol	RES	MIX	OON	RES	MIX	OON	RES	MIX	OON
10^{-4}	.97	.96	.96	1.	1.	1.	1.	1.	1.
10^{-3}		.96	.95		1.	1.		1.	1.
10^{-2}		.95	.86		1.	1.		1.	1.
10^{-1}		.95	.61		1.	.98		1.	1.
10^{0}		.87	.52		1.	.96		1.	1.
	$\varepsilon = .01$, $d = L_\infty$			$\varepsilon = .03$, $d = L_\infty$			$\varepsilon = .05$, $d = L_\infty$		
tol	RES	MIX	OON	RES	MIX	OON	RES	MIX	OON
10^{-4}	.81	.79	.73	1.	1.	1.	1.	1.	1.
10^{-3}		.79	.68		1.	1.		1.	1.
10^{-2}		.77	.50		1.	.98		1.	1.
10^{-1}		.76	.35		1.	.89		1.	.99
10^{0}		.62	.28		1.	.81		1.	.97

sample. In Figure 2, we reported the difference $\Delta\mathbf{h}(t)$ (see Equation (5)) between those two trajectories measured as the L_2 distance between each point in time

$$\Delta\mathbf{h}(t) = |\mathbf{h}(t) - \mathbf{h}_{\text{adv}}(t)|_2 \tag{5}$$

and averaged over all the samples in the test set. Each line is computed using adversarial examples crafted with the same tolerance τ used when measuring $\Delta\mathbf{h}(t)$. We can confirm that increasing values of τ causes the adversarial perturbation to be attenuated during the evolution of the internal state, even if the adversarial attack is performed setting a higher tolerance.

VII. Conclusions

In this paper, we presented an analysis of the robustness to adversarial examples of ODE-Nets — a recently introduced neural network architecture with continuous hidden states defined by ordinary differential equations. We compared three architectures (a residual, an ODE-based, and a mixed architecture) using the MNIST and CIFAR-10 datasets and observed that ODE-Nets provide superior robustness against PGD — a strong multi-step adversarial attack. Furthermore, by investigating the evolution of the internal states of ODE-nets in response to pristine and corresponding adversarial examples, we observed that the error tolerance parameter of the ODE solver does not substantially affect classification performances with respect to pristine samples while it significantly improve the robustness to adversarial examples: the higher the tolerance, the higher the resilience to adversarial inputs.

As future work, we plan to extend our analysis to higher-resolution and/or larger-scale datasets and to additional state-of-the-art adversarial attacks, such as Carlini and Wagner [6]. In addition, the presented findings open new interesting research directions in the field: as an example and possible future work, a novel detection method for adversarial examples could rely on the analysis of the outputs (or internal states) of ODE nets evaluated with multiple values of the tolerance parameter.

TABLE IV: Best configurations in terms of the trade-off between classification error and attack success rate.

Model	Tolerance of ODE Solver (τ)				
	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}
	Classification Error (MNIST)				
RES	.004	–	–	–	–
MIX	.005	.005	.005	.005	.005
OON	.005	.005	.006	.008	.012
	Attack Success Rate (MNIST, $\varepsilon = .1$, $d = L_2$)				
RES	.99	–	–	–	–
MIX	.98	.98	.98	.95	.91
OON	.87	.75	.57	.54	.46
	Classification Error (CIFAR-10)				
RES	.073	–	–	–	–
MIX	.078	.078	.079	.079	.079
OON	.091	.092	.093	.106	.113
	Attack Success Rate (CIFAR-10, $\varepsilon = .01$, $d = L_\infty$)				
RES	.81	–	–	–	–
MIX	.79	.79	.77	.76	.62
OON	.73	.68	.50	.35	.28

Acknowledgments

This work was partially supported by the ADA project, funded by Regione Toscana (CUP CIPE D55F17000290009), by the AI4EU EC-H2020 project (Contract n. 825619) and by the SMARTACCS project, funded by Fondazione CR Firenze (Contract n. 2018.0896). The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp and Tesla K40 GPUs used for this research.

References

[1] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

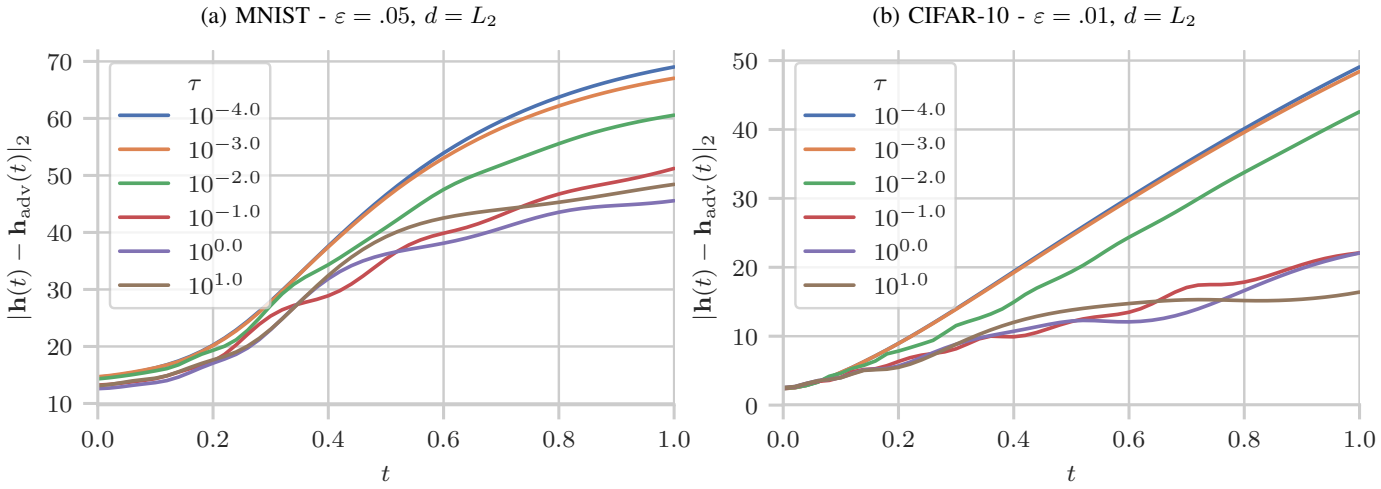

Fig. 2: Discrepancy of internal representations of the *OON* model caused by successful adversarial attacks. The y-axis shows the mean L_2 distance $\Delta\mathbf{h}(t)$ between pristine and attacked image representations during its continuous trajectory (time/depth is in the x-axis) for different values of the tolerance τ of the adaptive ODE solver.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[3] M. Barni, M. C. Stamm, and B. Tondi, "Adversarial multimedia forensics: Overview and challenges ahead," in *2018 26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 962–966.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[7] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *ICLR Workshops*, 2017. [Online]. Available: https://openreview.net/forum?id=HJGU3Rodl

[8] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy, SP 2016*, 2016, pp. 582–597. [Online]. Available: https://doi.org/10.1109/SP.2016.41

[9] P. Tabacof and E. Valle, "Exploring the space of adversarial images," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 426–433.

[10] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel, "On the (statistical) detection of adversarial examples," *CoRR*, vol. abs/1702.06280, 2017. [Online]. Available: http://arxiv.org/abs/1702.06280

[11] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *CoRR*, vol. abs/1703.00410, 2017. [Online]. Available: http://arxiv.org/abs/1703.00410

[12] Z. Gong, W. Wang, and W. Ku, "Adversarial and clean data are not twins," *CoRR*, vol. abs/1704.04960, 2017. [Online]. Available: http://arxiv.org/abs/1704.04960

[13] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *ICLR*, 2017. [Online]. Available: https://openreview.net/forum?id=SJzCSf9xg

[14] F. Carrara, F. Falchi, R. Caldelli, G. Amato, and R. Becarelli, "Adversarial image detection in deep neural networks," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 2815–2835, 2019.

[15] R. Caldelli, R. Becarelli, F. Carrara, F. Falchi, and G. Amato, "Exploiting CNN layer activations to improve adversarial image classification," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 2289–2293.

[16] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, 2018, pp. 6572–6583.

[17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[18] A. Kurakin, I. J. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. L. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J. Long, Y. Berdibekov, T. Akiba, S. Tokui, and M. Abe, "Adversarial attacks and defences competition," *CoRR*, vol. abs/1804.00097, 2018. [Online]. Available: http://arxiv.org/abs/1804.00097

[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[20] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *ICML*, 2018, pp. 284–293.

[21] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *ICLR*, 2017. [Online]. Available: https://openreview.net/forum?id=BJm4T4Kgx

[22] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics." in *ICCV*, 2017, pp. 5775–5783.

[23] F. Carrara, R. Becarelli, R. Caldelli, F. Falchi, and G. Amato, "Adversarial examples detection in features distance spaces," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[24] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations. 1, Nonstiff problems*. Springer-Vlg, 1991.

[25] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[27] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[30] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[31] J. Rauber, W. Brendel, and M. Bethge, "Foolbox v0.8.0: A python toolbox to benchmark the robustness of machine learning models," *CoRR*, vol. abs/1707.04131, 2017. [Online]. Available: http://arxiv.org/abs/1707.04131