A security study of Neural ODEs

Julien Girard, Guillaume Charpiat, Zakaria Chihani, Marc Schoenauer

4 juin 2019
Outline

1. Adversarial examples: a challenge to tackle
2. Neural Ordinary Differentials Equations
3. Case study
   - Methodology
   - Attacks
4. Results
What are adversarial examples?

A small video to begin with
What are adversarial examples?

A small video to begin with
Formal definition

For an input $x$, a classification function $f$, an adversarial perturbation $\delta$:

\[
\text{maximize} \quad \text{classifier misclassification}
\]

\[
\text{such that} \quad \text{perturbation stays below a certain threshold}
\]
Formal definition

For an input $x$, a classification function $f$, an adversarial perturbation $\delta$:

maximize $f(x) \neq f(x + \delta)$

such that $\|\delta\|_p \leq \varepsilon$
Why are adversarial examples important?

Adversarial examples:

- are transferable (Papernot et al., 2016, Transferability in Machine Learning..., Carlini et al. papers) ⇒ make ML more robust
Why are adversarial examples important?

Adversarial examples:
- are transferable (Papernot et al., 2016, Transferability in Machine Learning... , Carlini et al. papers) ⇒ make ML more robust

\[ \delta \]
Why are adversarial examples important?

Adversarial examples:

- are transferable (Papernot et al., 2016, Transferability in Machine Learning... Carlini et al. papers) ⇒ make ML more robust

- not well understood (Goodfellow et al. 2018, Adversarial Spheres, Madry et al., 2018, Adversarial Examples are not bugs...) ⇒ design better ML algorithms
Why are adversarial examples important?

Adversarial examples:

- are transferable (Papernot et al., 2016, Transferability in Machine Learning..., Carlini et al. papers) ⇒ make ML more robust

- not well understood (Goodfellow et al. 2018, Adversarial Spheres, Madry et al., 2018, Adversarial Examples are not bugs...) ⇒ design better ML algorithms

- provide us a *specification* to verify against ⇒ formal methods (later on my thesis, tomorrow at ForMaL)
Why do we bother about new architectures?

- evaluation of new architecture designs robustness $\Rightarrow$ test state of the art attacks
Why do we bother about new architectures?

- evaluation of new architecture designs robustness $\Rightarrow$ test state of the art attacks
Why do we bother about new architectures?

- evaluation of new architecture designs robustness ⇒ test state of the art attacks
- new vision on neural network computation ⇒ better intrinsic robustness properties?
Why do we bother about new architectures?

- Evaluation of new architecture designs robustness $\Rightarrow$ test state of the art attacks
- New vision on neural network computation $\Rightarrow$ better intrinsic robustness properties?
- New design could inspire us to invent new attacks and defenses $\Rightarrow$ invariants as stability?
What are ODE Nets?
What are ODEs?
Small recap on ODEs

Let \( y : x \in \mathbb{R}^d \rightarrow y(x) \in \mathbb{R}^p \), differentiable, \( t \) time

An ordinary differential equation (ODE) is \( F \) such that:

\[
F(x, y, y^1, y^2, \ldots, y^{(n)}, t) = 0
\]
Neural Ordinary Differentials Equations

Small recap on ODEs

Let \( y : x \in \mathbb{R}^d \rightarrow y(x) \in \mathbb{R}^p \), differentiable, \( t \) time

An ordinary differential equation (ODE) is \( F \) such that:

\[
F(x, y, y^1, y^2, \ldots, y^{(n)}, t) = 0
\]

\[
\begin{bmatrix}
\dot{\theta} \\
\ddot{\theta}
\end{bmatrix} = \begin{bmatrix}
\dot{\theta} \\
-\frac{g}{L} \sin(\theta(t))
\end{bmatrix}
\]

\[
\ddot{\theta} - \frac{g}{L} \sin(\theta(t)) = 0
\]

\[
F(\theta, \ddot{\theta}, t) : \ddot{\theta} - \frac{g}{L} \sin(\theta(t)) = 0
\]
How do we solve them

\[ \ddot{y} - \epsilon \cdot w \cdot (1 - y^2) \cdot \dot{y} + w^2 \cdot y = 0 \]

Van der Pol oscillator

No analytical solution in the general case \( \Rightarrow \) numeric approximations
How do we solve them - continued

A simple numerical method: Euler method

\[
\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \approx F(x_i, y_i)
\]

\[
y_{i+1} \approx F(x_i, y_i)
\]
How do we solve them - continued

A simple numerical method: Euler method

\[ \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \approx F(x_i, y_i) \]

\[ \frac{y_{i+2} - y_{i+1}}{x_{i+2} - x_{i+1}} \approx F(x_{i+1}, y_{i+1}) \]

Parameters:
- timesteps: accuracy vs speed
- for other solvers: multiple evaluations for increased stability
- error control
How do we solve them - continued

A simple numerical method: Euler method

\[
\begin{align*}
  \frac{y_{i+1} - y_i}{x_{i+1} - x_i} & \approx F(x_i, y_i) \\
  y_i & \approx y(t)
\end{align*}
\]

Parameters:
- timesteps: accuracy vs speed
- for other solvers: multiple evaluations for increased stability
- error control

A security study of Neural ODEs
4 juin 2019
Neural Ordinary Differentials Equations

**ODEs in neural networks?**

\[ h(x_i) = \mathcal{F}(x_i) + x_i : \text{state at a given point } i \]

A skip connection (He et al., 2015, Residual Deep Learning...)

\[ F(x) \]

\[ F(x) + x \]

\[ \text{weight layer} \]

\[ \text{relu} \]

\[ \text{identity} \]
ODEs in neural networks?

\[ h(x_i) = F(x_i) + x_i : \text{state at a given point } i \]

\[ x_i \text{ is the result of previous computations } h(x_{i-1}) \]

A skip connection (He et al., 2015, Residual Deep Learning...)
Neural Ordinary Differentials Equations

ODEs in neural networks?

\[ \mathbf{h}(\mathbf{x}_i) = \mathcal{F}(\mathbf{x}_i) + \mathbf{h}(\mathbf{x}_{i-1}) : \text{state at a given point } i \]

\[ \mathbf{x}_i \text{ is the result of previous computations } \mathbf{h}(\mathbf{x}_{i-1}) \]

\[
\frac{\mathbf{h}(\mathbf{x}_i) - \mathbf{h}(\mathbf{x}_{i-1})}{i+1-i} = \mathcal{F}(\mathbf{x}_i)
\]
ODEs in neural networks?

\[ h(x_i) = \mathcal{F}(x_i) + h(x_{i-1}) : \text{state at a given point } i \]
\[ x_i \text{ is the result of previous computations } h(x_{i-1}) \]

\[
\frac{h(x_i) - h(x_{i-1})}{i+1-i} = \mathcal{F}(x_i)
\]
\[
\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \approx \mathbf{F}(x_i, y_i)
\]

Euler method!

A skip connection (He et al., 2015, Residual Deep Learning...)
Neural Ordinary Differential Equations

ODE Net pipeline

Classical pipeline

Interest in image classification: lower parameter footprint

Chen et al., 2018, Neural Ordinary Differential Equations
Threat model

Goal: assert the attack and defense perimeter
Threat model

Goal : assert the attack and defense perimeter

<table>
<thead>
<tr>
<th></th>
<th>White box</th>
<th>Black box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access to the model’s</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access to the model’s</td>
<td>✓</td>
<td>limited</td>
</tr>
<tr>
<td>output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access to the gradient</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Knowledge of the defense</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Perturbation characteristics</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Assumptions on attacker’s capabilities
The Big Questions

What makes a good attack?

What makes a good defense?
The Big Questions

What makes a good attack?

Break robustness within the given threat model

What makes a good defense?
The Big Questions

What makes a good attack?

Break robustness within the given threat model

What makes a good defense?

1. Provably increase robustness within the given threat model
2. Limits the attack surface
FGSM (Goodfellow et al., 2014)

\[ x' = x + \varepsilon \, \text{sign}(\nabla_x L(\theta, x, y)) \]

Idea: make a step towards the direction maximizing the loss
Projected Gradient Descent $l_\infty$ (Madry et al., 2017)

\[ \min_\theta (\rho(\theta)), \text{ where } \rho(\theta) = E_{(x,y) \in D} \left[ \max_{\delta \in D} (L(\theta, x + \delta, y)) \right] \]

\[ x_{t+1} = \Pi(x + \delta)(x_t + \alpha \text{ sign}(\nabla_x L(\theta, x, y)) \left[ \delta \right] \]

Multiples iterations, works because of the geometric landscape
Carlini-Wagner $l_2$ (Carlini et al., 2016)

maximize classifier misclassification

such that perturbation stays below a certain threshold

$$
\min (c \star \| \delta \|_p + J(x + \delta, l))
$$

w.r.t.

$$x + \delta \in X
$$

$$J(x, l) = \max \left( \max_{i \neq t} \left( \text{logits}(x)_i - \text{logits}(x)_t \right), 0 \right)$$
Carlini-Wagner $l_2$ (Carlini et al., 2016)

maximize $f(x) \neq f(x + \delta)$

such that $\|\delta\|_p \leq \varepsilon$

$$\min(c \times \|\delta\|_p + J(x + \delta, l))$$

w.r.t.

$x + \delta \in X$

$$J(x, l) = \max((\max_{i \neq t}(\text{logits}(x)_i) - \text{logits}(x)_t), 0)$$
# Results

**ODE Nets are more vulnerable**

<table>
<thead>
<tr>
<th></th>
<th>FGSM ($\varepsilon = 0.3$)</th>
<th>C&amp;W</th>
<th>PGD ($\varepsilon = 0.3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>classical mnist</td>
<td>6.23/5.19%</td>
<td>2.39/0.21%</td>
<td>5.62/4.17%</td>
</tr>
<tr>
<td>ODE mnist</td>
<td>6.02/8.56%</td>
<td>0.82/0.01%</td>
<td>4.54/4.34%</td>
</tr>
<tr>
<td>classical cifar10</td>
<td>6.53/9.96%</td>
<td>1.07/0.0%</td>
<td>5.30/0.1%</td>
</tr>
<tr>
<td>ODE cifar10</td>
<td>7.22/0.14%</td>
<td>1.06/0.0%</td>
<td>6.48/0.03%</td>
</tr>
</tbody>
</table>

A security study of Neural ODEs

4 juin 2019

Inria

Inventors for the digital world
Visual clues

PGD (classical)

CW (classical)

FGSM (classical)

FGSM (ODE)

PGD (ODE)

CW (ODE)
Adversarial training is still efficient

<table>
<thead>
<tr>
<th></th>
<th>Natural</th>
<th>FGSM ($\varepsilon = 0.3$)</th>
<th>C&amp;W</th>
<th>PGD ($\varepsilon = 0.3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet5</td>
<td>−/98.65%</td>
<td>5.76/95.81%</td>
<td>0.57/79.43%</td>
<td>5.09/97.5%</td>
</tr>
<tr>
<td>ODE</td>
<td>−/99.4%</td>
<td>6.03/96.79%</td>
<td>2.51/22.24%</td>
<td>5.48/98.52%</td>
</tr>
</tbody>
</table>
ODE integration time : a potential key towards robustness?

<table>
<thead>
<tr>
<th>Network</th>
<th>Training end time</th>
<th>t=1</th>
<th>t=10</th>
<th>t=100</th>
<th>t=500</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODE-net small</td>
<td>10</td>
<td>49.45 / 0</td>
<td>98.64 / 9.34</td>
<td>26.35 / 12.83</td>
<td>9.94 / 8.09</td>
</tr>
<tr>
<td>ODE-net small</td>
<td>10-100</td>
<td>61.54 / 0</td>
<td>98.46 / 0.52</td>
<td>98.31 / 23.64</td>
<td>94.35 / 11.67</td>
</tr>
<tr>
<td>ODE-net small</td>
<td>100</td>
<td>37.58 / 0</td>
<td>66.43 / 0</td>
<td>98.52 / 13.25</td>
<td>72.16 / 14.16</td>
</tr>
<tr>
<td>ODE-net large</td>
<td>10</td>
<td>97.06 / 0.18</td>
<td>98.93 / 30.76</td>
<td>91.43 / 28.20</td>
<td>9.35 / 8.57</td>
</tr>
<tr>
<td>ODE-net large</td>
<td>10-100</td>
<td>72.84 / 0.15</td>
<td>99.08 / 70.67</td>
<td>99.11 / 85.98</td>
<td>94.66 / 62.29</td>
</tr>
<tr>
<td>ODE-net large</td>
<td>100</td>
<td>78.88 / 0.59</td>
<td>98.85 / 83.13</td>
<td>99.01 / 92.62</td>
<td>96.68 / 78.60</td>
</tr>
</tbody>
</table>

Courtesy of https://rajatvd.github.io/Neural-ODE-Adversarial/
Summary

1. ODE Nets are less robust than naturals comparable models
   - less parameters
   - perturbating the derivative is easier

2. Adversarial training is still efficient

3. Possible way to improve robustness
   - integration time
   - numerical stability (more robust numerical schemes, Lyapunov invariants, etc.)
Questions?

Shoot your questions :)