# Adversarial Deformations for Neural Ordinary Differential Equations

Shpresim Sadiku

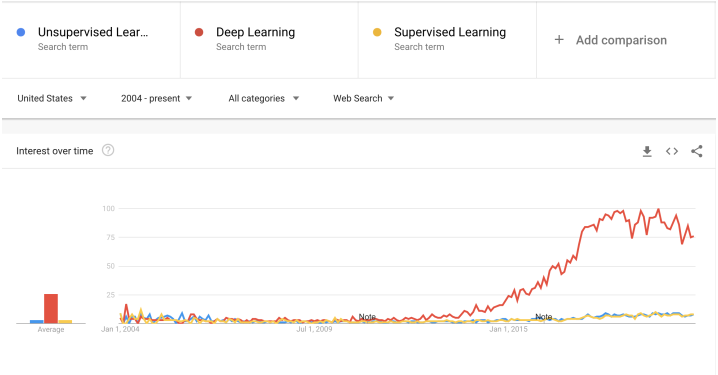
Technische Universität München
Department of Mathematics
Chair of Mathematical Physics

May 11, 2020

# Outline

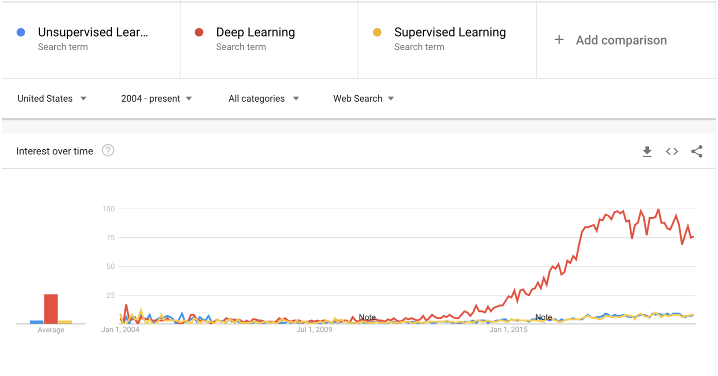
# Outline

# Machine Learning trends




Major breakthrough: (Krizhevsky et al., 2012) win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a large margin using Deep Convolutional Neural Networks (DCNNs) – AlexNet

# Machine Learning trends




Major breakthrough: (Krizhevsky et al., 2012) win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a large margin using Deep Convolutional Neural Networks (DCNNs) – AlexNet

# Limitations of Neural Networks

- Notoriously opaque inner workings
- Only few theoretical results explain their success in practice
- In image classification, imperceptibly perturbed input images (adversarial examples) are often classified very differently than the original image

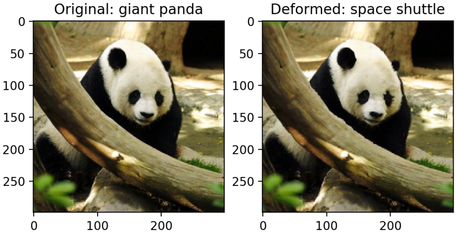



Figure 1: An adversarial example for a pre-trained Inception-v3 model (Szegedy et al., 2016) produced by ADef (Alaifari et al., 2018).

# Limitations of Neural Networks

- Notoriously opaque inner workings
- Only few theoretical results explain their success in practice
- In image classification, imperceptibly perturbed input images (adversarial examples) are often classified very differently than the original image

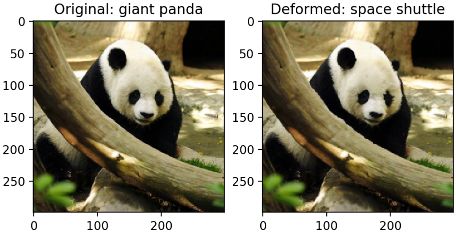



Figure 1: An adversarial example for a pre-trained Inception-v3 model (Szegedy et al., 2016) produced by ADef (Alaifari et al., 2018).

# Limitations of Neural Networks

- Notoriously opaque inner workings
- Only few theoretical results explain their success in practice
- In image classification, imperceptibly perturbed input images (adversarial examples) are often classified very differently than the original image

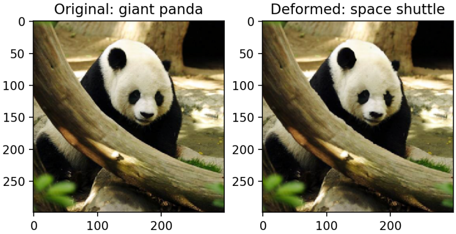



Figure 1: An adversarial example for a pre-trained Inception-v3 model (Szegedy et al., 2016) produced by ADef (Alaifari et al., 2018).

# Limitations of Neural Networks

- Notoriously opaque inner workings
- Only few theoretical results explain their success in practice
- In image classification, imperceptibly perturbed input images (adversarial examples) are often classified very differently than the original image

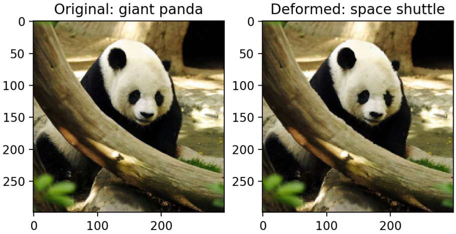



Figure 1: An adversarial example for a pre-trained Inception-v3 model (Szegedy et al., 2016) produced by ADef (Alaifari et al., 2018).

# Central Question

Should we expect rigorous mathematical analysis of neural networks?

- Focus on the interplay of three areas
  1. Expressivity of the Network Design
     ($\hookrightarrow$ Approximation Theory, Applied Harmonic Analysis,...)
  2. Learning via Optimal Control
     ($\hookrightarrow$ Optimization, Optimal Control,...)
  3. Generalization
     ($\hookrightarrow$ Statistics, Learning Theory, Stochastics,...)

The three problems cannot be studied in isolation!

# Central Question

Should we expect rigorous mathematical analysis of neural networks?

- Focus on the interplay of three areas
  1. Expressivity of the Network Design
     ($\hookrightarrow$ Approximation Theory, Applied Harmonic Analysis,...)
  2. Learning via Optimal Control
     ($\hookrightarrow$ Optimization, Optimal Control,...)
  3. Generalization
     ($\hookrightarrow$ Statistics, Learning Theory, Stochastics,...)

The three problems cannot be studied in isolation!

# Central Question

Should we expect rigorous mathematical analysis of neural networks?

- Focus on the interplay of three areas
  1. Expressivity of the Network Design
     ($\hookrightarrow$ Approximation Theory, Applied Harmonic Analysis,...)
  2. Learning via Optimal Control
     ($\hookrightarrow$ Optimization, Optimal Control,...)
  3. Generalization
     ($\hookrightarrow$ Statistics, Learning Theory, Stochastics,...)

The three problems cannot be studied in isolation!

# Central Question

Should we expect rigorous mathematical analysis of neural networks?

- Focus on the interplay of three areas
  1. Expressivity of the Network Design
     ($\hookrightarrow$ Approximation Theory, Applied Harmonic Analysis,...)
  2. Learning via Optimal Control
     ($\hookrightarrow$ Optimization, Optimal Control,...)
  3. Generalization
     ($\hookrightarrow$ Statistics, Learning Theory, Stochastics,...)

The three problems cannot be studied in isolation!

# Central Question

Should we expect rigorous mathematical analysis of neural networks?

- Focus on the interplay of three areas
  1. Expressivity of the Network Design
     ($\hookrightarrow$ Approximation Theory, Applied Harmonic Analysis,...)
  2. Learning via Optimal Control
     ($\hookrightarrow$ Optimization, Optimal Control,...)
  3. Generalization
     ($\hookrightarrow$ Statistics, Learning Theory, Stochastics,...)

The three problems cannot be studied in isolation!

# Central Question

Should we expect rigorous mathematical analysis of neural networks?

- Focus on the interplay of three areas
  1. Expressivity of the Network Design
     ($\hookrightarrow$ Approximation Theory, Applied Harmonic Analysis,...)
  2. Learning via Optimal Control
     ($\hookrightarrow$ Optimization, Optimal Control,...)
  3. Generalization
     ($\hookrightarrow$ Statistics, Learning Theory, Stochastics,...)

The three problems cannot be studied in isolation!

# Outline

# Density in $C(K)$

Consider density questions associated with the single hidden layer perceptron model

$$\Sigma(\sigma) = \text{span}\{\sigma(w \cdot x - \theta) : \theta \in \mathbb{R}, w \in \mathbb{R}^n\}$$

with activation function $\sigma : \mathbb{R} \to \mathbb{R}$, weights $w \in \mathbb{R}^n$ and bias $\theta \in \mathbb{R}$

- Find conditions under which $\Sigma(\sigma)$ is dense in $C(K)$ for any compact set $K \subset \mathbb{R}^n$

**Theorem 2.1 (Leshno et al., 1993)**

$\Sigma(\sigma)$ is dense in $C(\mathbb{R}^n)$ iff $\sigma \in L^\infty_{loc}(\mathbb{R})$ is not a polynomial (a.e.) and the closure of its points of discontinuity is of zero Lebesgue measure.

# Density in $C(K)$

Consider density questions associated with the single hidden layer perceptron model

$$\Sigma(\sigma) = \operatorname{span}\{\sigma(w \cdot x - \theta) : \theta \in \mathbb{R}, w \in \mathbb{R}^n\}$$

with activation function $\sigma : \mathbb{R} \to \mathbb{R}$, weights $w \in \mathbb{R}^n$ and bias $\theta \in \mathbb{R}$

- Find conditions under which $\Sigma(\sigma)$ is dense in $C(K)$ for any compact set $K \subset \mathbb{R}^n$

**Theorem 2.1 (Leshno et al., 1993)**

$\Sigma(\sigma)$ is dense in $C(\mathbb{R}^n)$ iff $\sigma \in L^\infty_{loc}(\mathbb{R})$ is not a polynomial (a.e.) and the closure of its points of discontinuity is of zero Lebesgue measure.

# Density in $C(K)$

Consider density questions associated with the single hidden layer perceptron model

$$\Sigma(\sigma) = \operatorname{span}\{\sigma(w \cdot x - \theta) : \theta \in \mathbb{R}, w \in \mathbb{R}^n\}$$

with activation function $\sigma : \mathbb{R} \to \mathbb{R}$, weights $w \in \mathbb{R}^n$ and bias $\theta \in \mathbb{R}$

- Find conditions under which $\Sigma(\sigma)$ is dense in $C(K)$ for any compact set $K \subset \mathbb{R}^n$

**Theorem 2.1 (Leshno et al., 1993)**

$\Sigma(\sigma)$ is dense in $C(\mathbb{R}^n)$ iff $\sigma \in L^\infty_{loc}(\mathbb{R})$ is not a polynomial (a.e.) and the closure of its points of discontinuity is of zero Lebesgue measure.

# Density in $C(K)$

Consider density questions associated with the single hidden layer perceptron model

$$\Sigma(\sigma) = \operatorname{span}\{\sigma(w \cdot x - \theta) : \theta \in \mathbb{R}, w \in \mathbb{R}^n\}$$

with activation function $\sigma : \mathbb{R} \to \mathbb{R}$, weights $w \in \mathbb{R}^n$ and bias $\theta \in \mathbb{R}$

- Find conditions under which $\Sigma(\sigma)$ is dense in $C(K)$ for any compact set $K \subset \mathbb{R}^n$

**Theorem 2.1 (Leshno et al., 1993)**

$\Sigma(\sigma)$ is dense in $C(\mathbb{R}^n)$ iff $\sigma \in L^\infty_{loc}(\mathbb{R})$ is not a polynomial (a.e.) and the closure of its points of discontinuity is of zero Lebesgue measure.

# Exponential Benefits of Deep Neural Networks

Denote by $\mathcal{F}(m,l) \subseteq \mathbb{R}^{\mathbb{R}}$ feed-forward neural networks with $l$ layers each with at most $m$ units, with ReLU activation functions everywhere but the output

- Binarize for classification problems: for each $f \in \mathcal{F}(m,l)$ define $\tilde{f} := \mathbb{1}_{f(x)\geq 1/2}$ and $\hat{R}(f) := \frac{1}{|S|}\sum_{(x,y)\in S} \mathbb{1}_{\tilde{f}(x)\neq y}$

**Theorem 2.2 (Telgarsky, 2015)**

Let $k \in \mathbb{N}, n = 2^k$ and $S := ((x_i, y_i))_{i=0}^{n-1}$ with $x_i = \frac{i}{n}, y_i = i \bmod 2$

- There is a $f \in \mathcal{F}(2, 2k)$ such that $\hat{R}(f) = 0$.
- If $m, l \in \mathbb{N}$ and $m < 2^{\frac{k-3}{l}-1}$ ($m$ is exponentially large) then $\hat{R}(h) \geq \frac{1}{6}, \forall h \in \mathcal{F}(m,l)$.

# Exponential Benefits of Deep Neural Networks

Denote by $\mathcal{F}(m,l) \subseteq \mathbb{R}^{\mathbb{R}}$ feed-forward neural networks with $l$ layers each with at most $m$ units, with ReLU activation functions everywhere but the output

- Binarize for classification problems: for each $f \in \mathcal{F}(m,l)$ define $\tilde{f} := \mathbb{1}_{f(x) \geq 1/2}$ and $\hat{R}(f) := \frac{1}{|S|} \sum_{(x,y) \in S} \mathbb{1}_{\tilde{f}(x) \neq y}$

**Theorem 2.2 (Telgarsky, 2015)**

Let $k \in \mathbb{N}, n = 2^k$ and $S := ((x_i, y_i))_{i=0}^{n-1}$ with $x_i = \frac{i}{n}, y_i = i \bmod 2$

- There is a $f \in \mathcal{F}(2, 2k)$ such that $\hat{R}(f) = 0$.
- If $m, l \in \mathbb{N}$ and $m < 2^{\frac{k-3}{l} - 1}$ ($m$ is exponentially large) then $\hat{R}(h) \geq \frac{1}{6}, \forall h \in \mathcal{F}(m,l)$.

# Exponential Benefits of Deep Neural Networks

Denote by $\mathcal{F}(m,l) \subseteq \mathbb{R}^{\mathbb{R}}$ feed-forward neural networks with $l$ layers each with at most $m$ units, with ReLU activation functions everywhere but the output

- Binarize for classification problems: for each $f \in \mathcal{F}(m,l)$ define $\tilde{f} := \mathbb{1}_{f(x) \geq 1/2}$ and $\hat{R}(f) := \frac{1}{|S|} \sum_{(x,y) \in S} \mathbb{1}_{\tilde{f}(x) \neq y}$

**Theorem 2.2 (Telgarsky, 2015)**

Let $k \in \mathbb{N}, n = 2^k$ and $S := ((x_i, y_i))_{i=0}^{n-1}$ with $x_i = \frac{i}{n}, y_i = i \bmod 2$

- There is a $f \in \mathcal{F}(2, 2k)$ such that $\hat{R}(f) = 0$.
- If $m, l \in \mathbb{N}$ and $m < 2^{\frac{k-3}{l} - 1}$ ($m$ is exponentially large) then $\hat{R}(h) \geq \frac{1}{6}, \forall h \in \mathcal{F}(m,l)$.

# Exponential Benefits of Deep Neural Networks

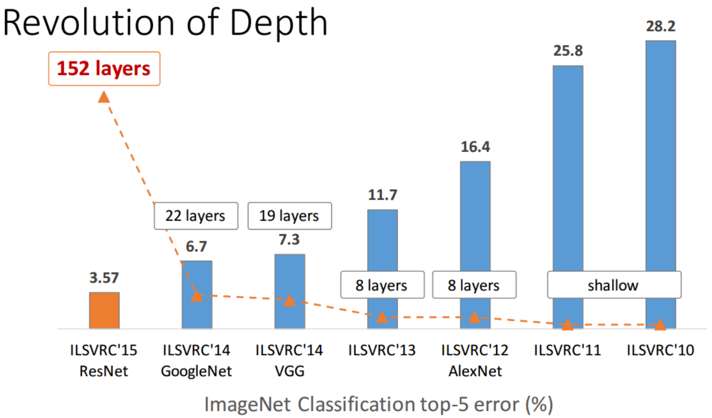
Denote by $\mathcal{F}(m,l) \subseteq \mathbb{R}^{\mathbb{R}}$ feed-forward neural networks with $l$ layers each with at most $m$ units, with ReLU activation functions everywhere but the output

- Binarize for classification problems: for each $f \in \mathcal{F}(m,l)$ define $\tilde{f} := \mathbb{1}_{f(x)\geq 1/2}$ and $\hat{R}(f) := \frac{1}{|S|}\sum_{(x,y)\in S}\mathbb{1}_{\tilde{f}(x)\neq y}$

**Theorem 2.2 (Telgarsky, 2015)**

Let $k \in \mathbb{N}, n = 2^k$ and $S := ((x_i, y_i))_{i=0}^{n-1}$ with $x_i = \frac{i}{n}, y_i = i \bmod 2$

- There is a $f \in \mathcal{F}(2, 2k)$ such that $\hat{R}(f) = 0$.
- If $m, l \in \mathbb{N}$ and $m < 2^{\frac{k-3}{l}-1}$ ($m$ is exponentially large) then $\hat{R}(h) \geq \frac{1}{6}, \forall h \in \mathcal{F}(m,l)$.


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

# Outline

# Neural Ordinary Differential Equations (Neural ODEs)

- (Weinan E, 2017) considers the continuous dynamical systems approach to deep learning
- Residual Networks (ResNets) updates

$$x_{t+1} = x_t + f(x_t, \theta_t)$$

  can be seen as an Euler discretization of a continuous transformation.
  Adding more layers and taking smaller steps, in the limit, the continuous dynamics of hidden units can be parameterized using an ODE specified by a neural network

$$\dot{x}(t) = f(x(t), \theta, t) \tag{1}$$

1. Given input $x_0$, solve (1) at time $t_N$, get output $x(t_N)$
2. Image classification task: apply a linear map $\mathcal{L} : \mathbb{R}^n \to \mathcal{Y}$ to $x(t_N)$

# Neural Ordinary Differential Equations (Neural ODEs)

- (Weinan E, 2017) considers the continuous dynamical systems approach to deep learning
- Residual Networks (ResNets) updates

$$x_{t+1} = x_t + f(x_t, \theta_t)$$

can be seen as an Euler discretization of a continuous transformation.
Adding more layers and taking smaller steps, in the limit, the continuous dynamics of hidden units can be parameterized using an ODE specified by a neural network

$$\dot{x}(t) = f(x(t), \theta, t) \tag{1}$$

1. Given input $x_0$, solve (1) at time $t_N$, get output $x(t_N)$
2. Image classification task: apply a linear map $\mathcal{L} : \mathbb{R}^n \to \mathcal{Y}$ to $x(t_N)$

# Neural Ordinary Differential Equations (Neural ODEs)

- (Weinan E, 2017) considers the continuous dynamical systems approach to deep learning
- Residual Networks (ResNets) updates

$$x_{t+1} = x_t + f(x_t, \theta_t)$$

can be seen as an Euler discretization of a continuous transformation.
Adding more layers and taking smaller steps, in the limit, the continuous dynamics of hidden units can be parameterized using an ODE specified by a neural network

$$\dot{x}(t) = f(x(t), \theta, t) \tag{1}$$

1. Given input $x_0$, solve (1) at time $t_N$, get output $x(t_N)$
2. Image classification task: apply a linear map $\mathcal{L} : \mathbb{R}^n \to \mathcal{Y}$ to $x(t_N)$

# How to train Neural ODEs?

- Find the frameworks and links with mathematics

$$\begin{aligned} \text{Deep Network} &\longleftrightarrow \text{Differential Equations (DE)} \\ \text{Network Architecture} &\longleftrightarrow \text{Numerical DE} \\ \text{Network Training} &\longleftrightarrow \text{Optimal Control} \end{aligned}$$

- Define a loss function $L$, $\mathcal{L}$ is fixed, and consider full-batch training. Optimization problem for training Neural ODEs

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(x(t_N)) \\ &\dot{x}(t) = f(x(t), \theta, t), \quad x(t_0) = x_0, \quad t_0 \le t \le t_N \end{aligned} \tag{2}$$

# How to train Neural ODEs?

- Find the frameworks and links with mathematics

Deep Network $\longleftrightarrow$ Differential Equations (DE)
Network Architecture $\longleftrightarrow$ Numerical DE
Network Training $\longleftrightarrow$ Optimal Control

- Define a loss function $L$, $\mathcal{L}$ is fixed, and consider full-batch training. Optimization problem for training Neural ODEs

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(x(t_N)) \\ &\dot{x}(t) = f(x(t), \theta, t), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_N \end{aligned} \tag{2}$$

# How to train Neural ODEs?

- Find the frameworks and links with mathematics

| | | |
|---|---|---|
| Deep Network | $\longleftrightarrow$ | Differential Equations (DE) |
| Network Architecture | $\longleftrightarrow$ | Numerical DE |
| Network Training | $\longleftrightarrow$ | Optimal Control |

- Define a loss function $L$, $\mathcal{L}$ is fixed, and consider full-batch training. Optimization problem for training Neural ODEs

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(x(t_N)) \\ &\dot{x}(t) = f(x(t), \theta, t), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_N \end{aligned} \tag{2}$$

# How to train Neural ODEs?

- Find the frameworks and links with mathematics

Deep Network $\longleftrightarrow$ Differential Equations (DE)
Network Architecture $\longleftrightarrow$ Numerical DE
Network Training $\longleftrightarrow$ Optimal Control

- Define a loss function $L$, $\mathcal{L}$ is fixed, and consider full-batch training. Optimization problem for training Neural ODEs

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(x(t_N)) \\ &\dot{x}(t) = f(x(t), \theta, t), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_N \end{aligned} \tag{2}$$

# How to train Neural ODEs?

- Find the frameworks and links with mathematics

$$\begin{aligned} \text{Deep Network} &\longleftrightarrow \text{Differential Equations (DE)} \\ \text{Network Architecture} &\longleftrightarrow \text{Numerical DE} \\ \text{Network Training} &\longleftrightarrow \text{Optimal Control} \end{aligned}$$

- Define a loss function $L$, $\mathcal{L}$ is fixed, and consider full-batch training. Optimization problem for training Neural ODEs

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(x(t_N)) \\ &\dot{x}(t) = f(x(t), \theta, t), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_N \end{aligned} \tag{2}$$

# How to train Neural ODEs?

- Find the frameworks and links with mathematics

| | | |
|---|---|---|
| Deep Network | $\longleftrightarrow$ | Differential Equations (DE) |
| Network Architecture | $\longleftrightarrow$ | Numerical DE |
| Network Training | $\longleftrightarrow$ | Optimal Control |

- Define a loss function $L$, $\mathcal{L}$ is fixed, and consider full-batch training. Optimization problem for training Neural ODEs

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(x(t_N)) \\ &\dot{x}(t) = f(x(t), \theta, t), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_N \end{aligned} \tag{2}$$

# Optimal Control Theory

In optimal control theory the following general control problem is considered

$$\min_{\theta \in \mathcal{U}} L(x(t_N), t_N) + \int_{t_0}^{t_N} R(x(t), \theta(t), t)dt \tag{3}$$
$$\dot{x}(t) = f(x(t), \theta(t), t), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_N$$

Defining the Hamiltonian $H(x, p, \theta, t) = p \cdot f(x, \theta, t) - R(x, \theta, t)$ for a costate process $p$ then the Pontryagin's Maximum Principle (PMP) gives the <u>necessary</u> conditions for optimal solutions of problem (3).

# Optimal Control Theory

In optimal control theory the following general control problem is considered

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(x(t_N), t_N) + \int_{t_0}^{t_N} R(x(t), \theta(t), t) dt \\ &\dot{x}(t) = f(x(t), \theta(t), t), \quad x(t_0) = x_0, \quad t_0 \le t \le t_N \end{aligned} \tag{3}$$

Defining the Hamiltonian $H(x, p, \theta, t) = p \cdot f(x, \theta, t) - R(x, \theta, t)$ for a costate process $p$ then the Pontryagin's Maximum Principle (PMP) gives the necessary conditions for optimal solutions of problem (3).

# Optimal Control Theory

In optimal control theory the following general control problem is considered

$$\begin{aligned}&\min_{\theta\in\mathcal{U}} L(x(t_N), t_N) + \int_{t_0}^{t_N} R(x(t), \theta(t), t)dt \\ &\dot{x}(t) = f(x(t), \theta(t), t), \quad x(t_0) = x_0, \quad t_0 \leq t \leq t_N\end{aligned} \tag{3}$$

Defining the Hamiltonian $H(x, p, \theta, t) = p \cdot f(x, \theta, t) - R(x, \theta, t)$ for a costate process $p$ then the Pontryagin's Maximum Principle (PMP) gives the necessary conditions for optimal solutions of problem (3).

# Pontryagin's Maximum Principle (Athans et al., 1966)

## Theorem 3.1

Let $\theta^*(t)$ be a bounded piecewise continuous function. Then, there exists a costate process $p^* : [t_0, t_N] \to \mathbb{R}^n$ such that the Hamilton's equations

$$\dot{x}^*(t) = \frac{\partial H}{\partial p}(x^*(t), p^*(t), \theta^*(t), t), \qquad x^*(t_0) = x_0$$

$$\dot{p}^*(t) = -\frac{\partial H}{\partial x}(x^*(t), p^*(t), \theta^*(t), t), \qquad p^*(t_N) = -\frac{\partial L}{\partial x}(x^*(t_N))$$

are satisfied. Moreover, for each $t \in [t_0, t_N]$, we have the Hamiltonian maximization condition

$$H(x^*(t), p^*(t), \theta^*(t), t) \geq H(x^*(t), p^*(t), \theta, t)$$

for all $\theta \in \Theta$.

# Pontryagin's Maximum Principle (Athans et al., 1966)

## Theorem 3.1

Let $\theta^*(t)$ be a bounded piecewise continuous function. Then, there exists a costate process $p^* : [t_0, t_N] \to \mathbb{R}^n$ such that the Hamilton's equations

$$\dot{x}^*(t) = \tfrac{\partial H}{\partial p}(x^*(t), p^*(t), \theta^*(t), t), \qquad x^*(t_0) = x_0$$
$$\dot{p}^*(t) = -\tfrac{\partial H}{\partial x}(x^*(t), p^*(t), \theta^*(t), t), \qquad p^*(t_N) = -\frac{\partial L}{\partial x}(x^*(t_N))$$

are satisfied. Moreover, for each $t \in [t_0, t_N]$, we have the Hamiltonian maximization condition

$$H(x^*(t), p^*(t), \theta^*(t), t) \geq H(x^*(t), p^*(t), \theta, t)$$

for all $\theta \in \Theta$.

# Reverse-mode derivative of an ODE IVP

- Problem (2) is a special case of (3), no regularization term $R$

$$H(x, p, \theta, t) = p \cdot f(x, \theta, t)$$

- (Chen et al., 2018) give the gradients of the loss w.r.t. all possible inputs to an ODE solver

$$\frac{\partial L}{\partial x(t_0)} = p(t_N) - \int_{t_N}^{t_0} \left( \frac{\partial f}{\partial x}(x(t), \theta, t) \right)' p(t) dt$$

$$\frac{\partial L}{\partial \theta} = - \int_{t_N}^{t_0} \left( \frac{\partial f}{\partial \theta}(x(t), \theta, t) \right)' p(t) dt$$

$$\frac{\partial L}{\partial t_N} = f(x(t_N), \theta, t_N)' p(t_N)$$

$$\frac{\partial L}{\partial t_0} = \frac{\partial L}{\partial t_N} - \int_{t_N}^{t_0} \left( \frac{\partial f}{\partial t}(x(t), \theta, t) \right)' p(t) dt$$

# Reverse-mode derivative of an ODE IVP

- Problem (2) is a special case of (3), no regularization term $R$

$$H(x, p, \theta, t) = p \cdot f(x, \theta, t)$$

- (Chen et al., 2018) give the gradients of the loss w.r.t. all possible inputs to an ODE solver

$$\frac{\partial L}{\partial x(t_0)} = p(t_N) - \int_{t_N}^{t_0} \left(\frac{\partial f}{\partial x}(x(t), \theta, t)\right)' p(t)dt$$

$$\frac{\partial L}{\partial \theta} = -\int_{t_N}^{t_0} \left(\frac{\partial f}{\partial \theta}(x(t), \theta, t)\right)' p(t)dt$$

$$\frac{\partial L}{\partial t_N} = f(x(t_N), \theta, t_N)' p(t_N)$$

$$\frac{\partial L}{\partial t_0} = \frac{\partial L}{\partial t_N} - \int_{t_N}^{t_0} \left(\frac{\partial f}{\partial t}(x(t), \theta, t)\right)' p(t)dt$$

# Reverse-mode derivative of an ODE IVP

- Problem (2) is a special case of (3), no regularization term $R$

$$H(x, p, \theta, t) = p \cdot f(x, \theta, t)$$

- (Chen et al., 2018) give the gradients of the loss w.r.t. all possible inputs to an ODE solver

$$\frac{\partial L}{\partial x(t_0)} = p(t_N) - \int_{t_N}^{t_0} \left( \frac{\partial f}{\partial x}(x(t), \theta, t) \right)' p(t) dt$$

$$\frac{\partial L}{\partial \theta} = - \int_{t_N}^{t_0} \left( \frac{\partial f}{\partial \theta}(x(t), \theta, t) \right)' p(t) dt$$

$$\frac{\partial L}{\partial t_N} = f(x(t_N), \theta, t_N)' p(t_N)$$

$$\frac{\partial L}{\partial t_0} = \frac{\partial L}{\partial t_N} - \int_{t_N}^{t_0} \left( \frac{\partial f}{\partial t}(x(t), \theta, t) \right)' p(t) dt$$

# Robustness of Neural ODEs

- Expose Neural ODEs to inputs of various types of adversarial attacks, measure the sensitivity of the corresponding outputs
- Adversarial perturbations (Szegedy et al., 2013) add Gaussian noise to inputs

$$\min \|r\|_2$$
$$\mathcal{K}(x+r) = l$$
$$x + r \in [0,1]^n$$

- Fast Gradient Sign Method (Goodfellow et al., 2014) maximize the network loss

$$r = \arg\max_{\|r\|_\infty \leq \epsilon} J(\theta, x + r, t)$$

- DeepFool (Moosavi-Dezfooli et al., 2016), assuming linear separation, finds minimal perturbations in the $\ell_p$ norm

# Robustness of Neural ODEs

- Expose Neural ODEs to inputs of various types of adversarial attacks, measure the sensitivity of the corresponding outputs
- Adversarial perturbations (Szegedy et al., 2013) add Gaussian noise to inputs

$$\min \|r\|_2$$
$$\mathcal{K}(x+r) = l$$
$$x + r \in [0,1]^n$$

- Fast Gradient Sign Method (Goodfellow et al., 2014) maximize the network loss

$$r = \arg\max_{\|r\|_\infty \leq \epsilon} J(\theta, x+r, t)$$

- DeepFool (Moosavi-Dezfooli et al., 2016), assuming linear separation, finds minimal perturbations in the $\ell_p$ norm

# Robustness of Neural ODEs

- Expose Neural ODEs to inputs of various types of adversarial attacks, measure the sensitivity of the corresponding outputs
- Adversarial perturbations (Szegedy et al., 2013) add Gaussian noise to inputs

$$\begin{gathered}\min \|r\|_2 \\ \mathcal{K}(x+r) = l \\ x + r \in [0,1]^n\end{gathered}$$

- Fast Gradient Sign Method (Goodfellow et al., 2014) maximize the network loss

$$r = \underset{\|r\|_\infty \leq \epsilon}{\arg\max}\, J(\theta, x+r, t)$$

- DeepFool (Moosavi-Dezfooli et al., 2016), assuming linear separation, finds minimal perturbations in the $\ell_p$ norm

# Robustness of Neural ODEs

- Expose Neural ODEs to inputs of various types of adversarial attacks, measure the sensitivity of the corresponding outputs
- Adversarial perturbations (Szegedy et al., 2013) add Gaussian noise to inputs

$$\min \|r\|_2$$
$$\mathcal{K}(x+r) = l$$
$$x + r \in [0,1]^n$$

- Fast Gradient Sign Method (Goodfellow et al., 2014) maximize the network loss

$$r = \underset{\|r\|_\infty \le \epsilon}{\arg\max}\, J(\theta, x+r, t)$$

- DeepFool (Moosavi-Dezfooli et al., 2016), assuming linear separation, finds minimal perturbations in the $\ell_p$ norm

# Robustness of Neural ODEs

- Expose Neural ODEs to inputs of various types of adversarial attacks, measure the sensitivity of the corresponding outputs
- Adversarial perturbations (Szegedy et al., 2013) add Gaussian noise to inputs

$$\min \|r\|_2$$
$$\mathcal{K}(x+r) = l$$
$$x + r \in [0,1]^n$$

- Fast Gradient Sign Method (Goodfellow et al., 2014) maximize the network loss

$$r = \underset{\|r\|_\infty \leq \epsilon}{\arg\max}\, J(\theta, x+r, t)$$

- DeepFool (Moosavi-Dezfooli et al., 2016), assuming linear separation, finds minimal perturbations in the $\ell_p$ norm

# Adversarial Deformations

- Adversarial deformations - ADef (Alaifari et al., 2018) deform inputs w.r.t. vector field $\tau : [0,1]^2 \to \mathbb{R}^2$

$$x^\tau(u) = x(u + \tau(u)), \quad \forall u \in [0,1]^2$$

- In general, $r = x - x^\tau$ is unbounded in $\ell_p$ norm even for indistinguishable transformations
- Size of the deformation is calculated as

$$\|\tau\|_T := \max_{i,j \in W} \|\tau(i,j)\|_2$$

# Adversarial Deformations

- Adversarial deformations - ADef (Alaifari et al., 2018) deform inputs w.r.t. vector field $\tau : [0,1]^2 \to \mathbb{R}^2$

$$x^\tau(u) = x(u + \tau(u)), \quad \forall u \in [0,1]^2$$

- In general, $r = x - x^\tau$ is unbounded in $\ell_p$ norm even for indistinguishable transformations
- Size of the deformation is calculated as

$$\|\tau\|_T := \max_{i,j \in W} \|\tau(i,j)\|_2$$

# Adversarial Deformations

- Adversarial deformations - ADef (Alaifari et al., 2018) deform inputs w.r.t. vector field $\tau : [0,1]^2 \to \mathbb{R}^2$

$$x^\tau(u) = x(u + \tau(u)), \quad \forall u \in [0,1]^2$$

- In general, $r = x - x^\tau$ is unbounded in $\ell_p$ norm even for indistinguishable transformations
- Size of the deformation is calculated as

$$\|\tau\|_T := \max_{i,j \in W} \|\tau(i,j)\|_2$$

# Adversarial Deformations

- Adversarial deformations - ADef (Alaifari et al., 2018) deform inputs w.r.t. vector field $\tau : [0,1]^2 \to \mathbb{R}^2$

$$x^\tau(u) = x(u + \tau(u)), \quad \forall u \in [0,1]^2$$

- In general, $r = x - x^\tau$ is unbounded in $\ell_p$ norm even for indistinguishable transformations
- Size of the deformation is calculated as

$$\|\tau\|_T := \max_{i,j \in W} \|\tau(i,j)\|_2$$

# Tests

- Superior stability of Neural ODEs over convolutional neural networks w.r.t. adversarial perturbations and deformations
- Intrinsic regularization in Neural ODEs due to non-intersecting ODE trajectories

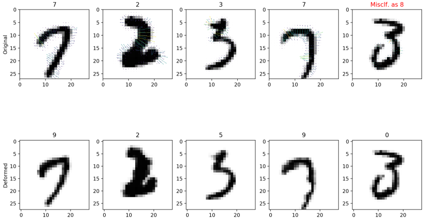



Figure 2: Adversarial deformations for Neural ODEs. First row: Original images from the MNIST test set. Second row: The deformed images.

# Tests

- Superior stability of Neural ODEs over convolutional neural networks w.r.t. adversarial perturbations and deformations
- Intrinsic regularization in Neural ODEs due to non-intersecting ODE trajectories

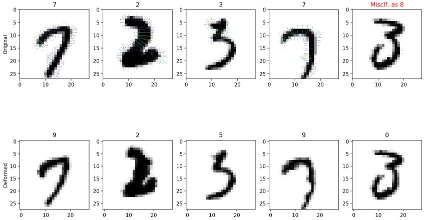



Figure 2: Adversarial deformations for Neural ODEs. First row: Original images from the MNIST test set. Second row: The deformed images.

# Tests

- Superior stability of Neural ODEs over convolutional neural networks w.r.t. adversarial perturbations and deformations
- Intrinsic regularization in Neural ODEs due to non-intersecting ODE trajectories

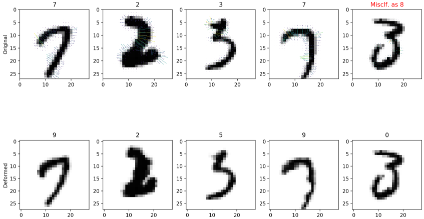



Figure 2: Adversarial deformations for Neural ODEs. First row: Original images from the MNIST test set. Second row: The deformed images.

# Tests

- Superior stability of Neural ODEs over convolutional neural networks w.r.t. adversarial perturbations and deformations
- Intrinsic regularization in Neural ODEs due to non-intersecting ODE trajectories

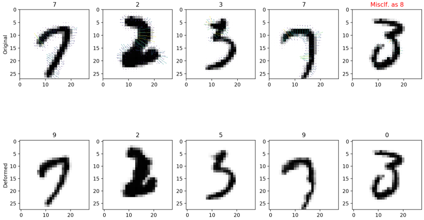



Figure 2: Adversarial deformations for Neural ODEs. First row: Original images from the MNIST test set. Second row: The deformed images.

# Outline

# Outcomes

- Universality of neural networks within the space of continuous functions under weak assumptions on the activation function (i.e., non-polynomiality and local essential boundedness)
- Exponential efficiency of deep neural networks over shallow neural networks
- Optimal Control Theory to exploit the specific structure and train continuous-depth models of constant memory cost
- Stability results of Neural ODEs along with formal verification promise possible usage in safety and security critical applications

# Outcomes

- Universality of neural networks within the space of continuous functions under weak assumptions on the activation function (i.e., non-polynomiality and local essential boundedness)
- Exponential efficiency of deep neural networks over shallow neural networks
- Optimal Control Theory to exploit the specific structure and train continuous-depth models of constant memory cost
- Stability results of Neural ODEs along with formal verification promise possible usage in safety and security critical applications

# Outcomes

- Universality of neural networks within the space of continuous functions under weak assumptions on the activation function (i.e., non-polynomiality and local essential boundedness)
- Exponential efficiency of deep neural networks over shallow neural networks
- Optimal Control Theory to exploit the specific structure and train continuous-depth models of constant memory cost
- Stability results of Neural ODEs along with formal verification promise possible usage in safety and security critical applications

# Outcomes

- Universality of neural networks within the space of continuous functions under weak assumptions on the activation function (i.e., non-polynomiality and local essential boundedness)
- Exponential efficiency of deep neural networks over shallow neural networks
- Optimal Control Theory to exploit the specific structure and train continuous-depth models of constant memory cost
- Stability results of Neural ODEs along with formal verification promise possible usage in safety and security critical applications