Brainwashing Computers: nudging neural networks to make reckless decisions

SIG Research Meeting

Alexandru C. Serban[1,2]

[1]Digital Security
Radboud University, Nijmegen

[2]Research Team
Software Improvement Group, Amsterdam

20.07.2018

Contents

What?

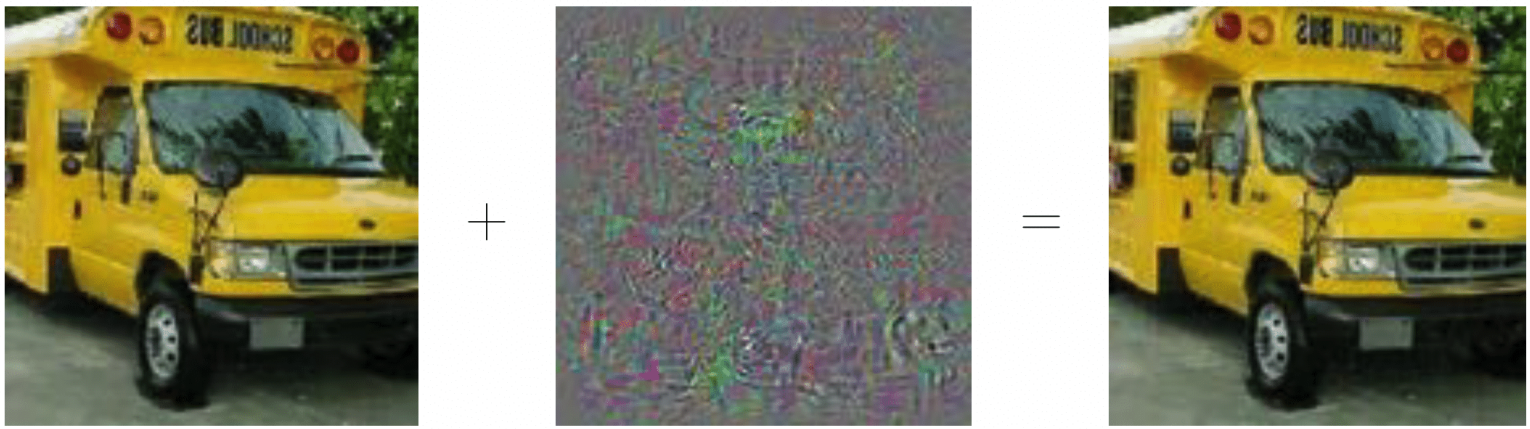


Figure: Adversarial Examples [11].

What?

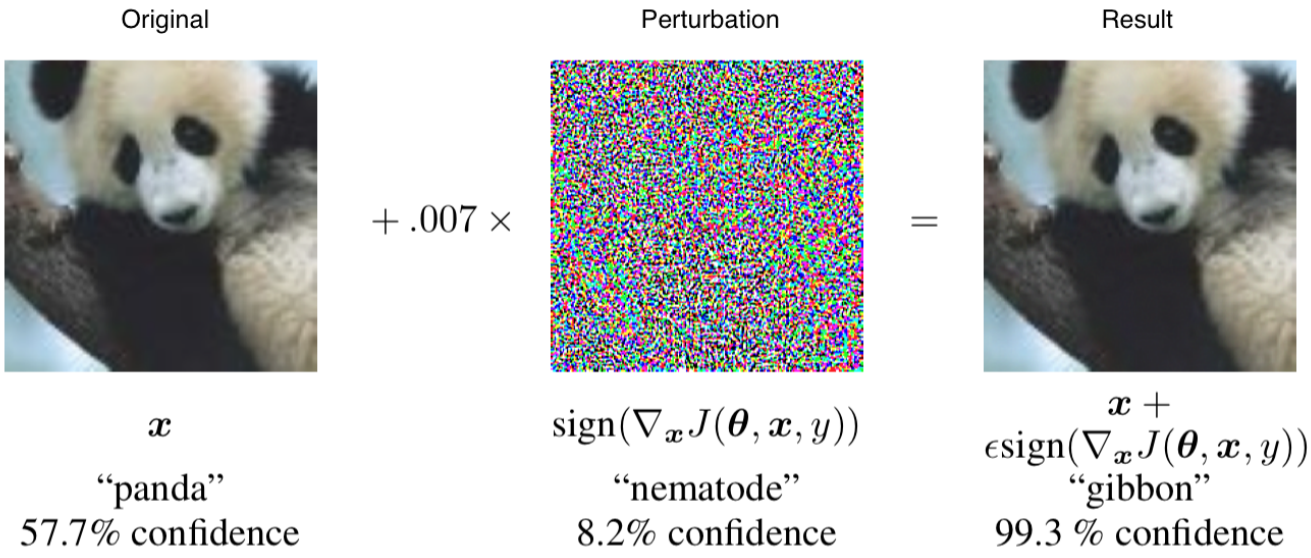


Figure: Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake [6].

Why?

Machine learning achieved high performance on a variety of tasks:

Figure: Left: Object Detection [4]. Middle: Malware Detection [3]. Right: Self-driving cars [2]

Why?

Some tasks are safety critical:

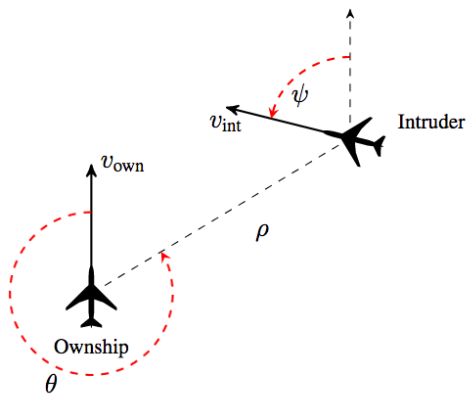


Figure: Left: Traffic Sign Detection [5]. Right: Airplane Collision Avoidance [7]

Deep Learning Recap [remember last talk?]

Recap: Machines Learning

Definition

Computer programs learn from *experience* E with respect to some class of *tasks* T and *performance measure* P, if its performance at tasks in T as measured by P *improves* with experience E [8].

Recap: Machines Learning

Definition

Computer programs learn from *experience* E with respect to some class of *tasks* T and *performance measure* P, if its performance at tasks in T as measured by P *improves* with experience E [8].

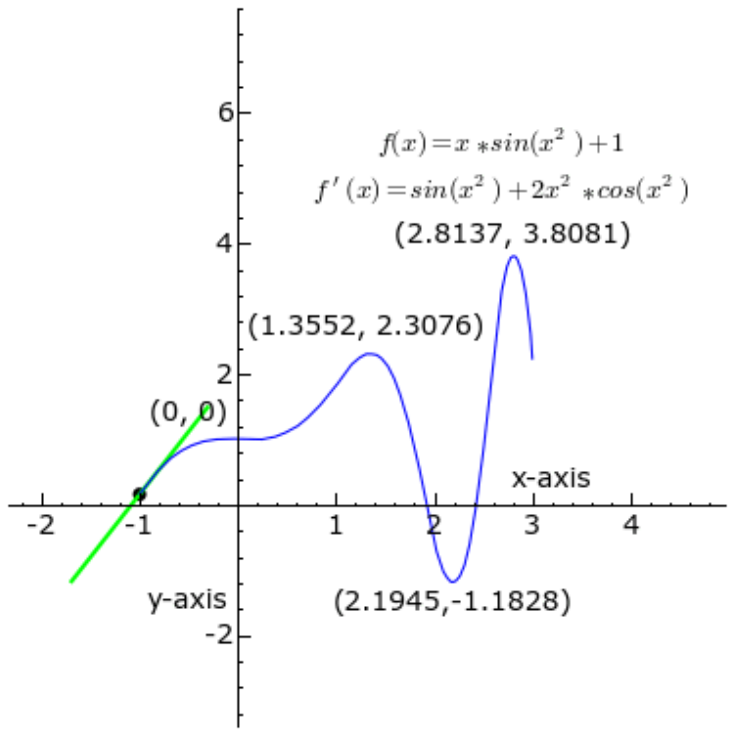
1. Task - understand images, move boxes, etc.
2. Experience - interpret *data*
3. Performance measure i.e. error/cost/objective functions - number of correct objects recognised, number of boxes moved, etc.

Learning is used analogously to *modeling* i.e. finding the best function $f(x, \theta)$ that can approximate a task.

Recap: Minimising Error

Definition

A function $f(x)$ is *increasing* on any interval in which $f'(x) > 0$ and it is *decreasing* on any interval in which $f'(x) < 0$.



Recap: Gradient

$$f : \mathbb{R}^n \to \mathbb{R}$$
$$y = f(x_1, x_2, \ldots x_n, \theta)$$
$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

Definition

The gradient is a multi-variable generalisation of the derivative.

Recap: The Jacobian Matrix

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

$$(y_1, y_2, \ldots y_m) = f(x_1, x_2 \ldots x_n, \theta)$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Definition

The Jacobian Matrix is the matrix of all *first-order* partial derivatives of a *vector-valued* function.

A simple example

Problem setting:

Task: Classify irises (flowers) in two categories.

Experience: Labeled
measurements: $(x, y) = ([\mathit{height}, \mathit{color}, \ldots], \mathit{Virginica})$

Performance measure?

Model $[f(x, \theta)]$?

Linear Models

Performance: Mean Squared Error (MSE)

Model: $\hat{y} = \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{x}$

Where:

$\boldsymbol{W} \in \mathbb{R}^n$ is a vector of parameters

$MSE = \frac{1}{m} \sum_i (\hat{y}^{(test)} - y^{(test)})_i^2$

Linear Models

Performance: Mean Squared Error (MSE)

Model: $\hat{y} = \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{x}$

Where:

$\boldsymbol{W} \in \mathbb{R}^n$ is a vector of parameters

$MSE = \frac{1}{m} \sum_i (\hat{y}^{(test)} - y^{(test)})_i^2$

Goal: Minimise the error

Linear Models

Performance: Mean Squared Error (MSE)

Model: $\hat{y} = \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{x}$

Where:

$\boldsymbol{W} \in \mathbb{R}^n$ is a vector of parameters

$MSE = \frac{1}{m} \sum_i (\hat{y}^{(test)} - y^{(test)})_i^2$

Goal: Minimise the error

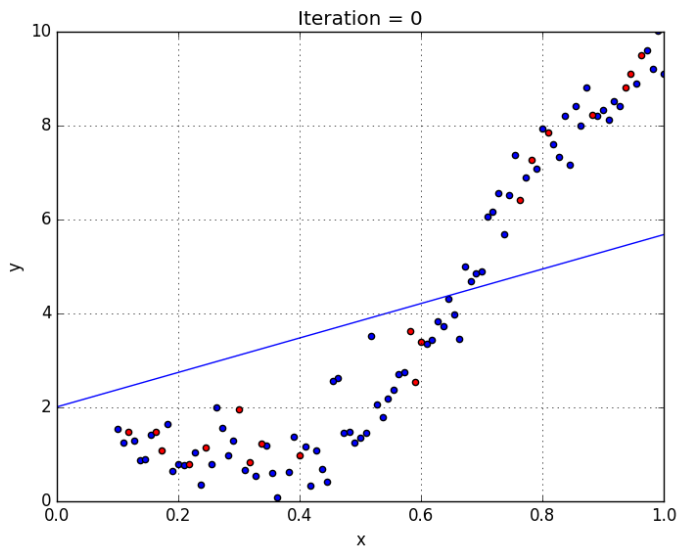
Solution: Compute derivatives!

Linear Models

Solution:

$$\nabla_w MSE_{train} = 0$$

$$\nabla_w \frac{1}{m} \sum_i (\hat{y}^{(test)} - y^{(test)})_i^2$$



In search for better approximations

Requirements:

- Represent highly nonlinear functions in large hyper-spaces
- Derivable
- Parallel scaling

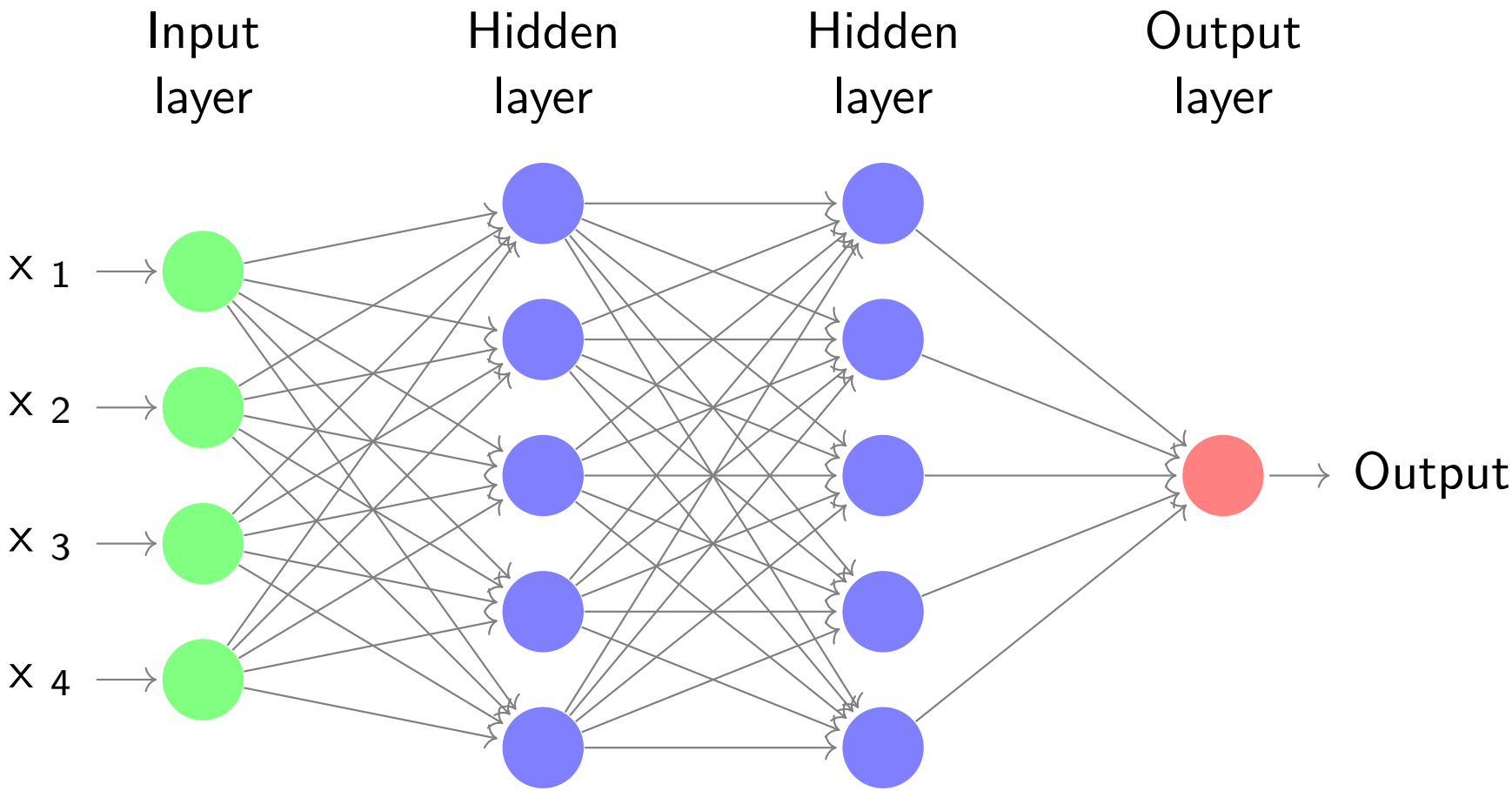
In search for better approximations

Requirements:

- Represent highly nonlinear functions in large hyper-spaces
- Derivable
- Parallel scaling

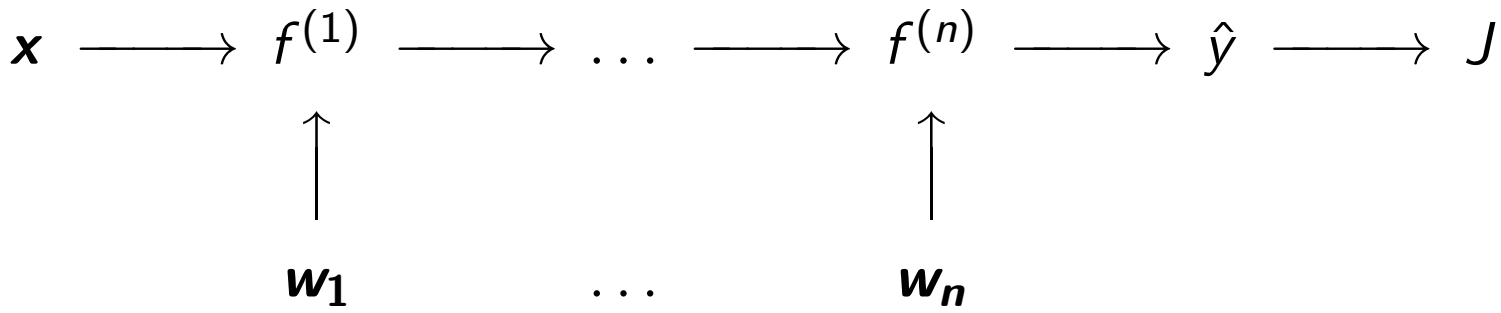
Solution? Functional *composition*!

Neural Networks



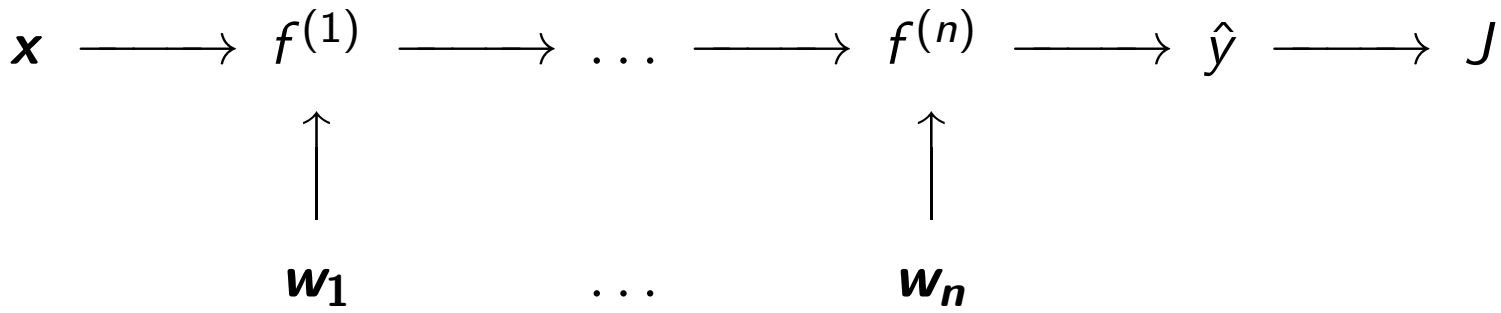
Deep Representations

- A *deep representation* is a composition of many functions

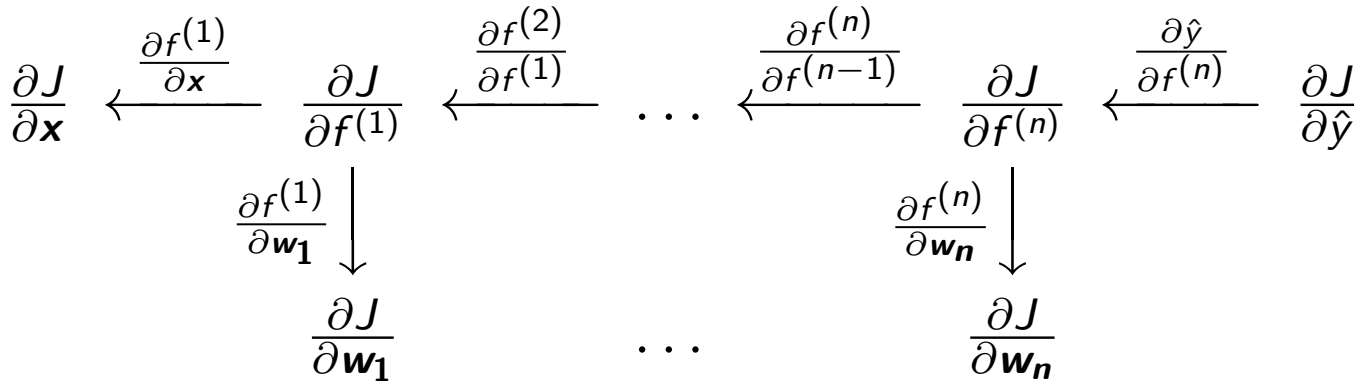


Deep Representations

- A *deep representation* is a composition of many functions



- Its gradient can be *back-propagated* by the chain rule



Projection in High Dimensional Space

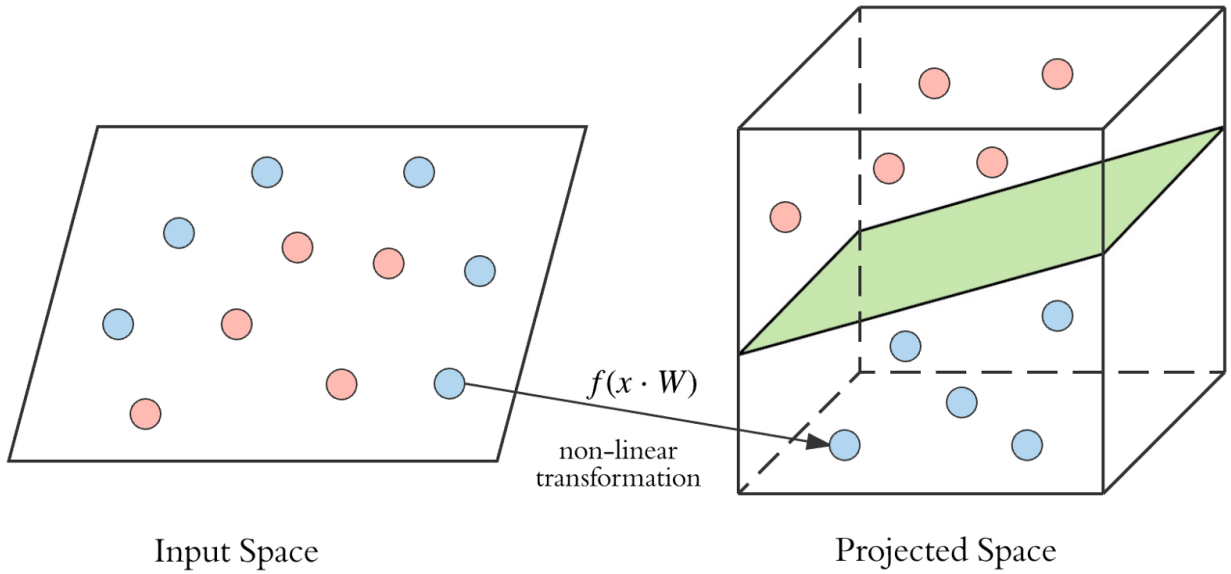


Figure: Example of Projection in R^3 [1].

Attacks [why? how?]

I.I.D Assumption

I.I.D Assumption

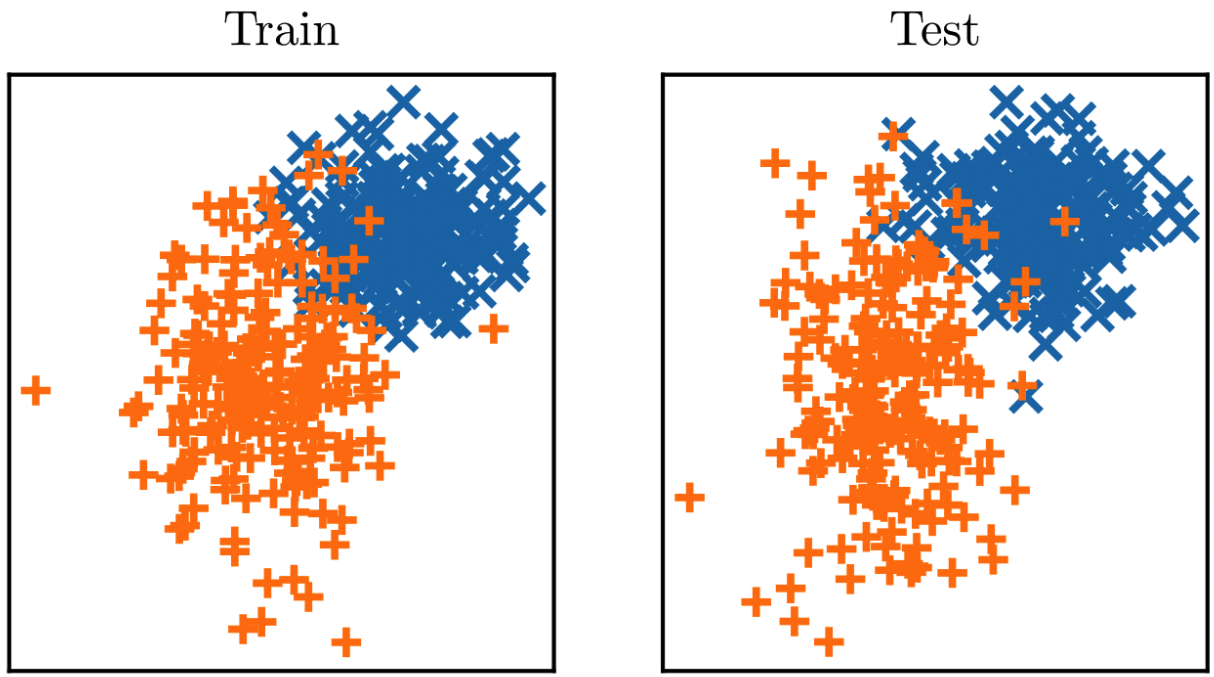


Figure: Machine learning algorithms assume all train and test examples are drawn from independently from the same distribution.

Security Requires Moving Beyond I.I.D

Figure: Physical perturbation experiments [5].

Security Requires Moving Beyond I.I.D

Figure: Physical perturbation experiments [5].

- attackers intentionally shift the distribution toward unusual inputs (at test time)
- attackers search for a single input that causes a mistake

Intentional Distribution Shift

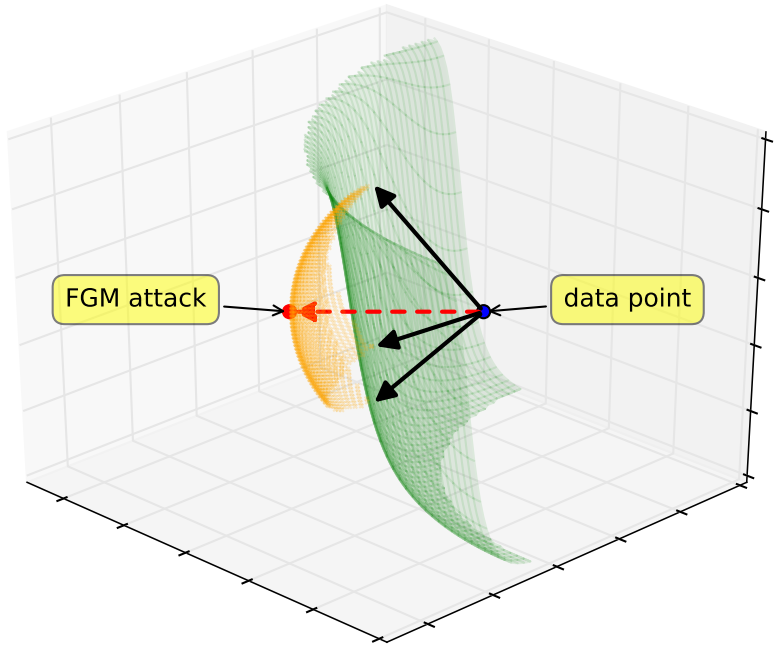


Figure: Designing an input to causes a miss-classification [12].

Examples of attacks

Design
by optimisation [11]:

$$\begin{aligned} \min_{\boldsymbol{x}'} \quad & \|\boldsymbol{x}' - \boldsymbol{x}\| \\ s.t. \quad & f(\boldsymbol{x}') = l' \\ & f(\boldsymbol{x}) = l \\ & l \neq l' \\ & \boldsymbol{x}' \in [0,1]^m \end{aligned}$$

Examples of attacks

Design by optimisation [11]:

$$\begin{aligned} \min_{\boldsymbol{x}'} \quad & \|\boldsymbol{x}' - \boldsymbol{x}\| \\ s.t. \quad & f(\boldsymbol{x}') = l' \\ & f(\boldsymbol{x}) = l \\ & l \neq l' \\ & \boldsymbol{x}' \in [0,1]^m \end{aligned}$$

Fast-Methods using Loss Gradient [6]:

$$\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$$

Examples of attacks

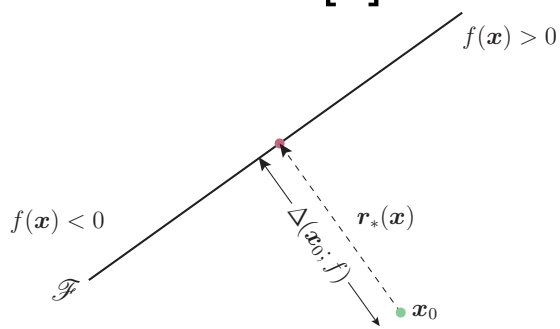
Design by optimisation [11]:

$$\begin{aligned} \min_{\boldsymbol{x}'} \quad & \|\boldsymbol{x}' - \boldsymbol{x}\| \\ s.t. \quad & f(\boldsymbol{x}') = l' \\ & f(\boldsymbol{x}) = l \\ & l \neq l' \\ & \boldsymbol{x}' \in [0,1]^m \end{aligned}$$

Fast-Methods using Loss Gradient [6]:

$$\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$$

Projections on discriminant surface [9]:



Results

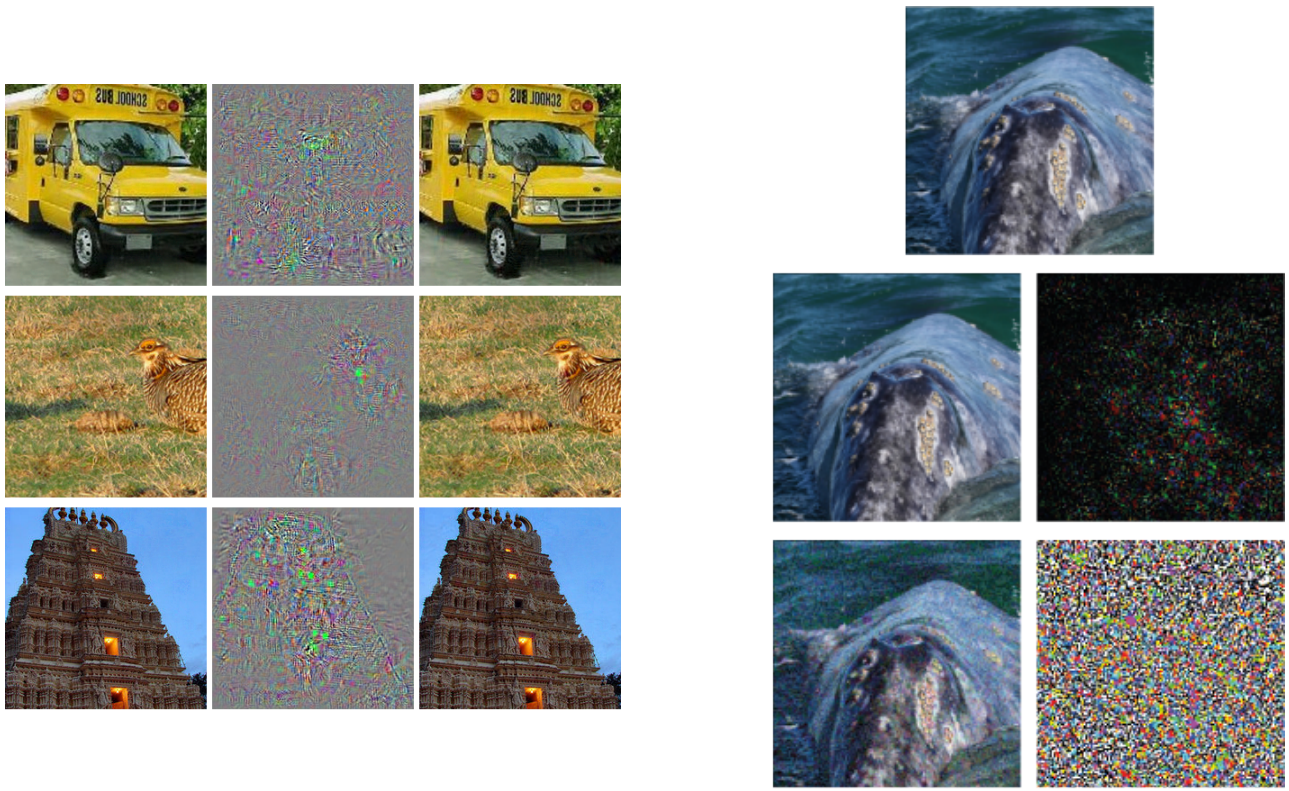


Figure: Adversarial Examples generated with L-BFGS [11] (left) and by projection on discriminant [9](right)

Generative Adversarial Attacks

Objective: Generate adversarial examples from noise.

Generative Adversarial Attacks

Objective: Generate adversarial examples from noise.

Steps:

1. Learn to generate normal examples from noise using generative models.
2. Learn to generate adversarial examples from (1).

Generative Adversarial Examples

Figure: Natural generative adversarial examples [13].

Defences [solutions?]

Proposed Defences

- Train with [weak] adversarial examples - does not generalise :(

Proposed Defences

- Train with [weak] adversarial examples - does not generalise :(
- Regularisation with weight decay - reduces accuracy on clean examples a lot :(

Proposed Defences

- Train with [weak] adversarial examples - does not generalise :(
- Regularisation with weight decay - reduces accuracy on clean examples a lot :(
- Transfer knowledge between models (distillation) - seems to generalise, but it's an illusion :(

Proposed Defences

- Train with [weak] adversarial examples - does not generalise :(
- Regularisation with weight decay - reduces accuracy on clean examples a lot :(
- Transfer knowledge between models (distillation) - seems to generalise, but it's an illusion :(
- Modify input image (crop, rotate, etc.) - does not affect adaptive attackers :(

Proposed Defences

- Train with [weak] adversarial examples - does not generalise :(
- Regularisation with weight decay - reduces accuracy on clean examples a lot :(
- Transfer knowledge between models (distillation) - seems to generalise, but it's an illusion :(
- Modify input image (crop, rotate, etc.) - does not affect adaptive attackers :(
- Increase capacity - hard to train :(

Proposed Defences

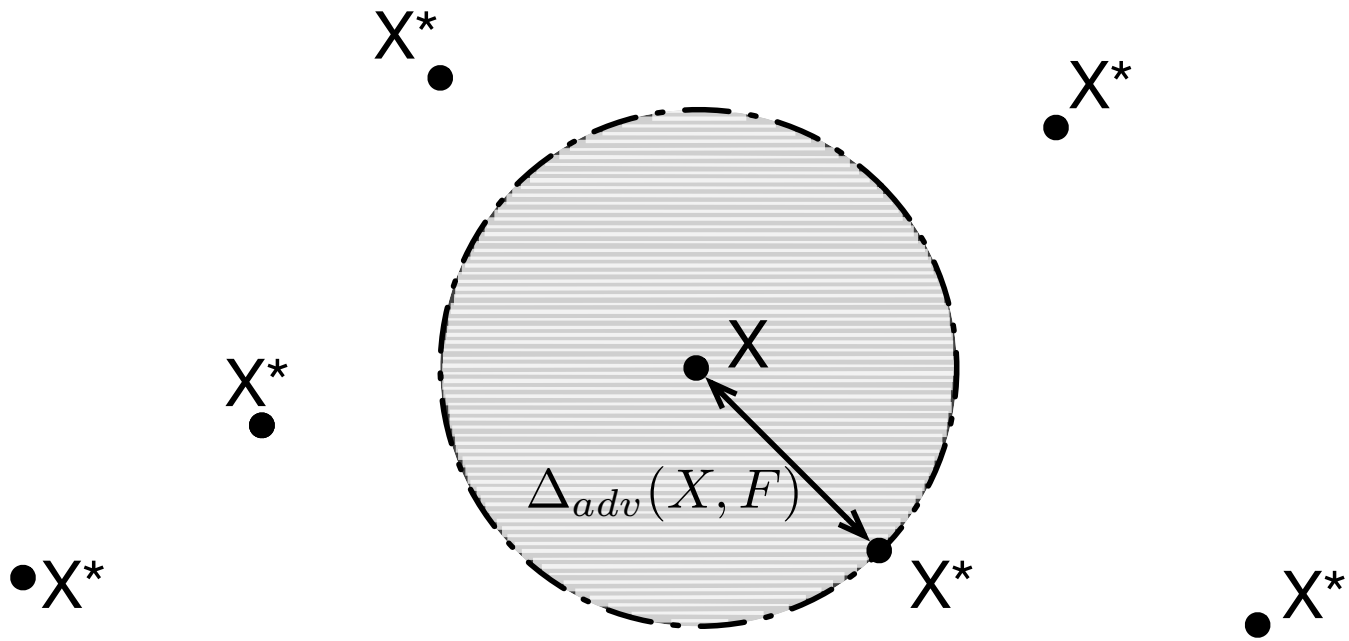
- Train with [weak] adversarial examples - does not generalise :(
- Regularisation with weight decay - reduces accuracy on clean examples a lot :(
- Transfer knowledge between models (distillation) - seems to generalise, but it's an illusion :(
- Modify input image (crop, rotate, etc.) - does not affect adaptive attackers :(
- Increase capacity - hard to train :(
- Increase non-linearity - hard to train :(
- Train only with maximum possible attacks (min loss -max attacks) - does not generalises over threat models :(

Proposed Defences

- Train with [weak] adversarial examples - does not generalise :(
- Regularisation with weight decay - reduces accuracy on clean examples a lot :(
- Transfer knowledge between models (distillation) - seems to generalise, but it's an illusion :(
- Modify input image (crop, rotate, etc.) - does not affect adaptive attackers :(
- Increase capacity - hard to train :(
- Increase non-linearity - hard to train :(
- Train only with maximum possible attacks (min loss -max attacks) - does not generalises over threat models :(
- Logit Pairing - state-of-the-art - only 50% accuracy :(

Research Directions [unsolved problems]

Robustness?



Transferability

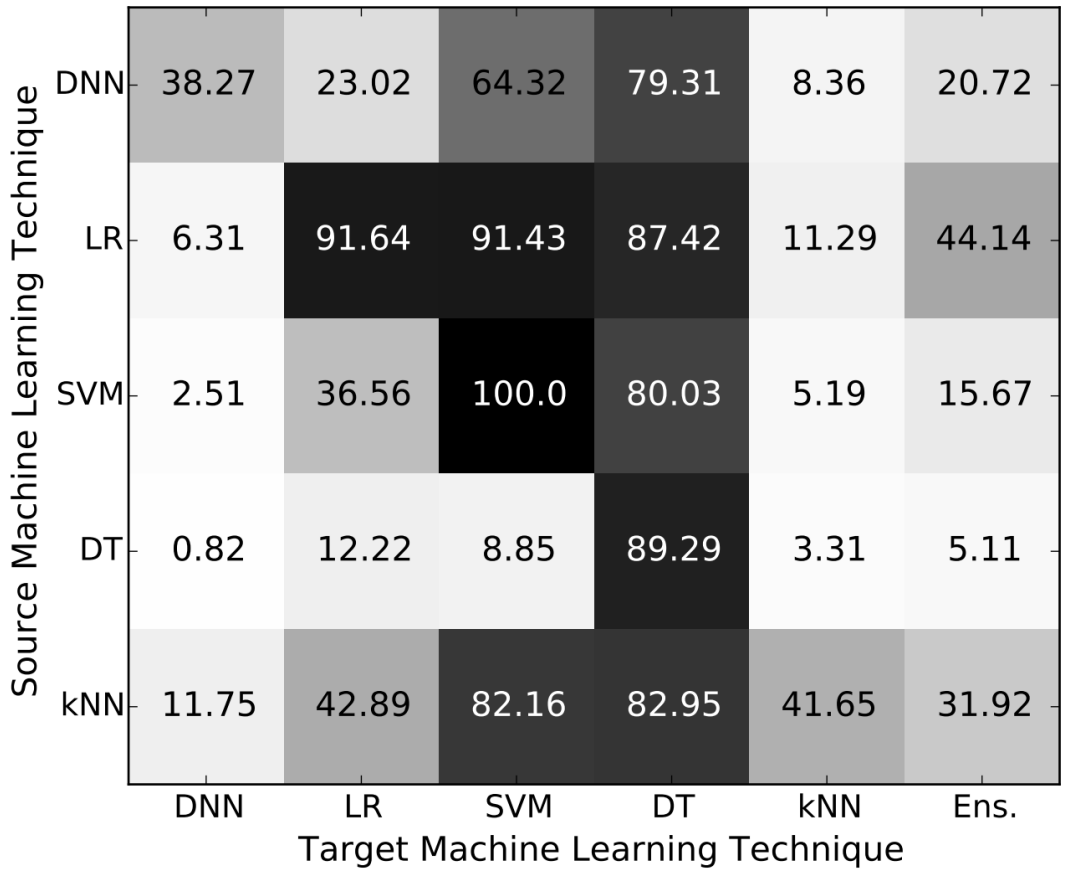


Figure: Cross-models transferability [10]

Who Watches The Watchers

- New attack models outside the norm ball

Who Watches The Watchers

- New attack models outside the norm ball
- New certification methods

Who Watches The Watchers

- New attack models outside the norm ball
- New certification methods
- Clear security requirements

Who Watches The Watchers

- New attack models outside the norm ball
- New certification methods
- Clear security requirements
- Semi-supervised learning, model-based optimisation, understanding the brain, etc.

Conclusions

Take aways

- Machine learning models assume training and test data are i.i.d

Take aways

- Machine learning models assume training and test data are i.i.d
- Because of this ML models are easily *fooled*

Take aways

- Machine learning models assume training and test data are i.i.d
- Because of this ML models are easily *fooled*
- Adversarial examples show that computers learn something different than we think (they might be less *intelligent*)

Take aways

- Machine learning models assume training and test data are i.i.d
- Because of this ML models are easily *fooled*
- Adversarial examples show that computers learn something different than we think (they might be less *intelligent*)
- There is no working defence against adversarial examples (no clear goals though)

References I

- https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6.
- Mariusz Bojarski et al. "End to end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316* (2016).
- George E Dahl et al. "Large-scale malware classification using random projections and neural networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3422–3426.
- Dumitru Erhan et al. "Scalable object detection using deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2147–2154.

References II

- et al Eykholt Kevin. "Robust Physical-World Attacks on Deep Learning Visual Classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
- Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).
- Kyle D Julian et al. "Policy compression for aircraft collision avoidance systems". In: *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE. 2016, pp. 1–10.
- Tom M Mitchell et al. "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

References III


Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. EPFL-CONF-218057. 2016.


Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples". In: *arXiv preprint arXiv:1605.07277* (2016).


Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).


Florian Tramèr et al. "The space of transferable adversarial examples". In: *arXiv preprint arXiv:1704.03453* (2017).

References IV

Zhengli Zhao, Dheeru Dua, and Sameer Singh. "Generating Natural Adversarial Examples". In: *CoRR* abs/1710.11342 (2017). arXiv: 1710.11342. URL: http://arxiv.org/abs/1710.11342.