Advanced ML course

Lecture 9: Adversarial Examples

Neychev Radoslav
ML Instructor (MIPT, HSE, Harbour.Space, BigData Team)
Research Scientist, MIPT

28.11.2019, Moscow, Russia

Outline

- Previous lectures recap
- Adversarial examples
    - Historical overview
    - Overfitting vs underfitting
    - Fast gradient sign method
    - Investigating the decision boundaries
- Attacking black-box models
- Potential applications
- Practice and Q&A

References

These slides are inspired a lot by Ian Goodfellow Lecture 16 in CS231n 2017 course at Stanford and paper “Explaining and harnessing adversarial examples” by Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy, ICLR 2015

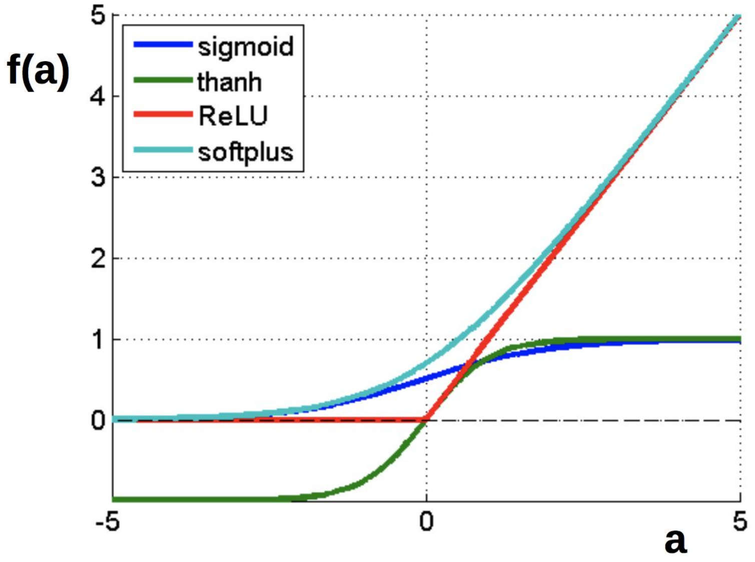
Activation functions

$$f(a) = \frac{1}{1 + e^a}$$

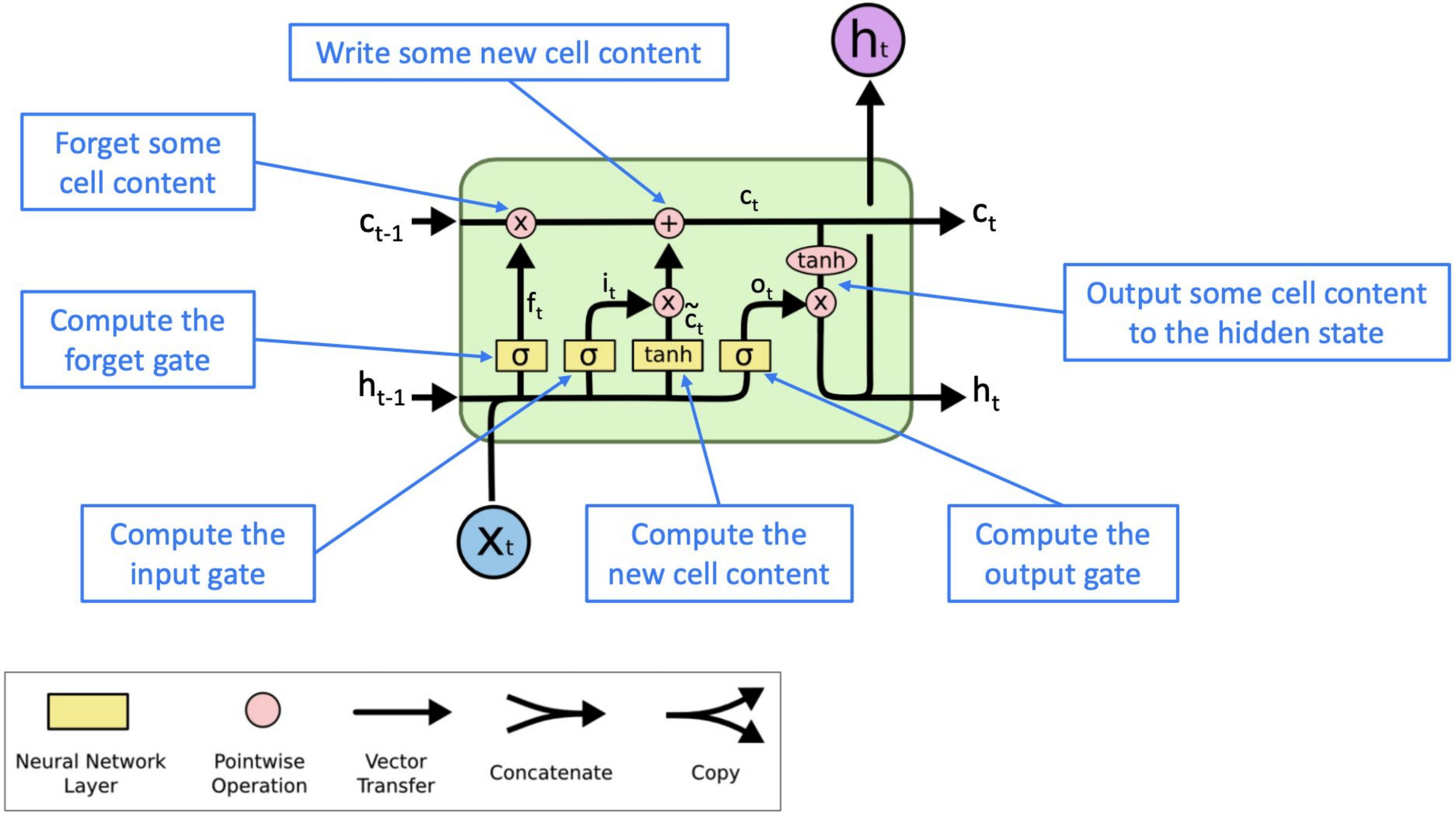
$$f(a) = \tanh(a)$$

$$f(a) = \max(0, a)$$

$$f(a) = \log(1 + e^a)$$



LSTM cell



Based on: Lecture by Abigail See, CS224n Lecture 7

LSTM operations

Sigmoid function: all gate values are between 0 and 1

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

$$\boldsymbol{f}^{(t)} = \boxed{\sigma}\left(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{i}^{(t)} = \boxed{\sigma}\left(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{o}^{(t)} = \boxed{\sigma}\left(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

New cell content: this is the new content to be written to the cell

Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content

Hidden state: read ("output") some content from the cell

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}$$

All these are vectors of same length *n*

Gates are applied using element-wise product

Adversarial examples timeline

- 2004
  - fooling the spam filter
    - e.g. “Adversarial classification” Dalvi et al 2004
- 2013
  - fooling neural nets
    - e.g. “Evasion Attacks Against Machine Learning at Test Time” Biggio 2013
- 2014
  - cheap closed form attack
    - e.g. Goodfellow et al 2014
- 2015+
  - Self-supervised learning, many kinds of different attacks etc.

Adversarial examples

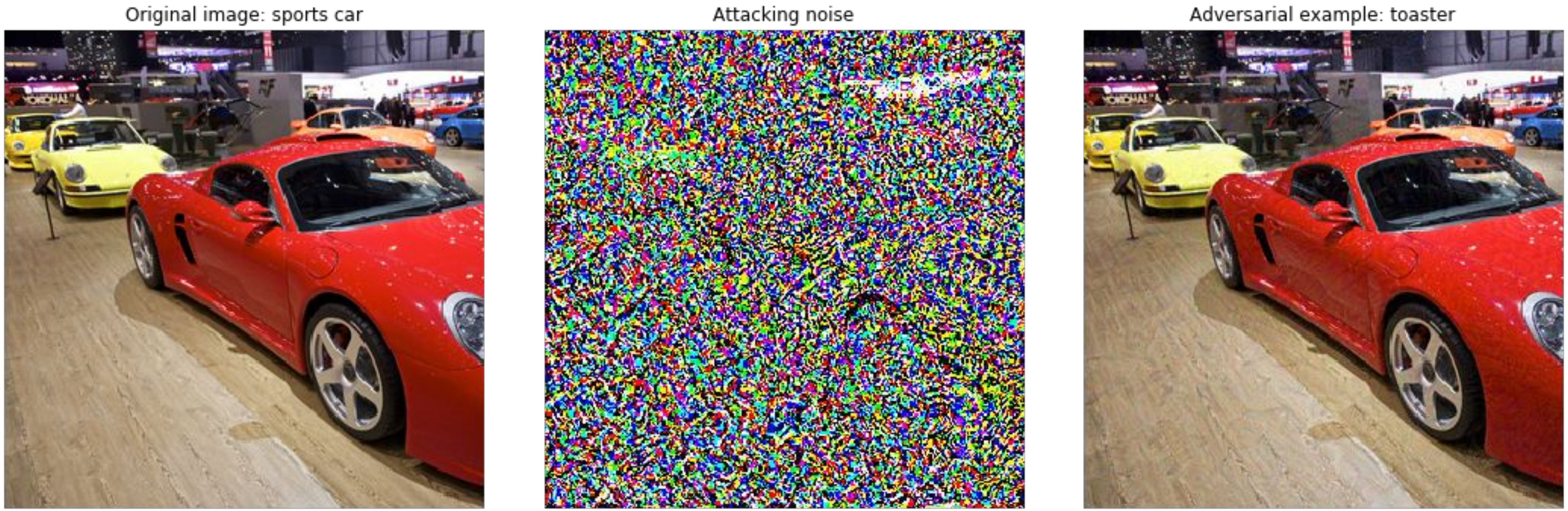


Image source: Y combinator blog post on Adversarial attacks

Adversarial examples

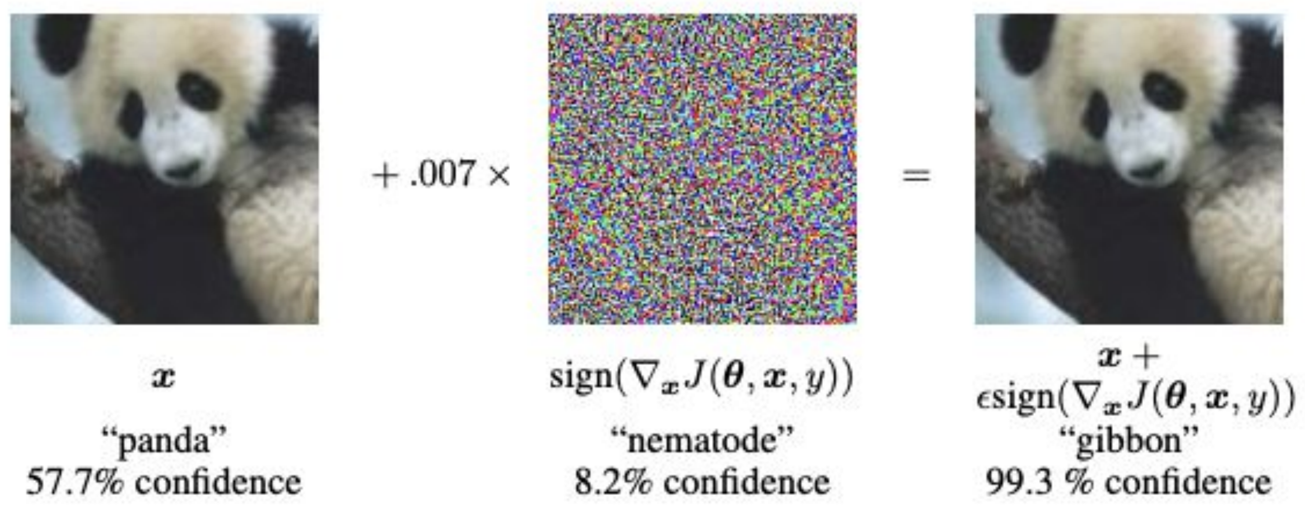

$\boldsymbol{x}$
"panda"
57.7% confidence
$+ .007 \times$
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$=$
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Image source: "Explaining and harnessing adversarial examples" by Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy

How does it work?

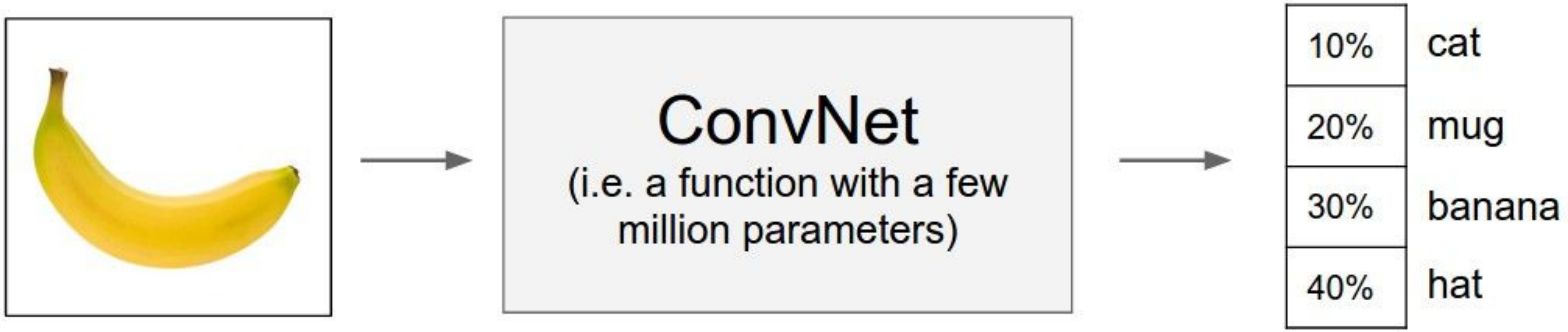


Image source: [Breaking Linear Classifiers on ImageNet blogpost by Andrej Karpathy]

Linear explanation of adversarial examples

Assume adversarial example $\vec{x}_{\mathrm{adv}} = \vec{x} + \vec{\eta}$ where $\|\vec{\eta}\|_\infty = \varepsilon$

Linear model is defined as $f(\vec{x}) = \vec{w}^T \vec{x}$

$$f(\vec{x}_{\mathrm{adv}}) = \vec{w}^T \vec{x} + \vec{w}^T \vec{\eta}$$

While input is changed only by ε feature-wise, the general activation

is changed by $\varepsilon n m$ if there are n features and their average magnitude is m.

Adversarial examples

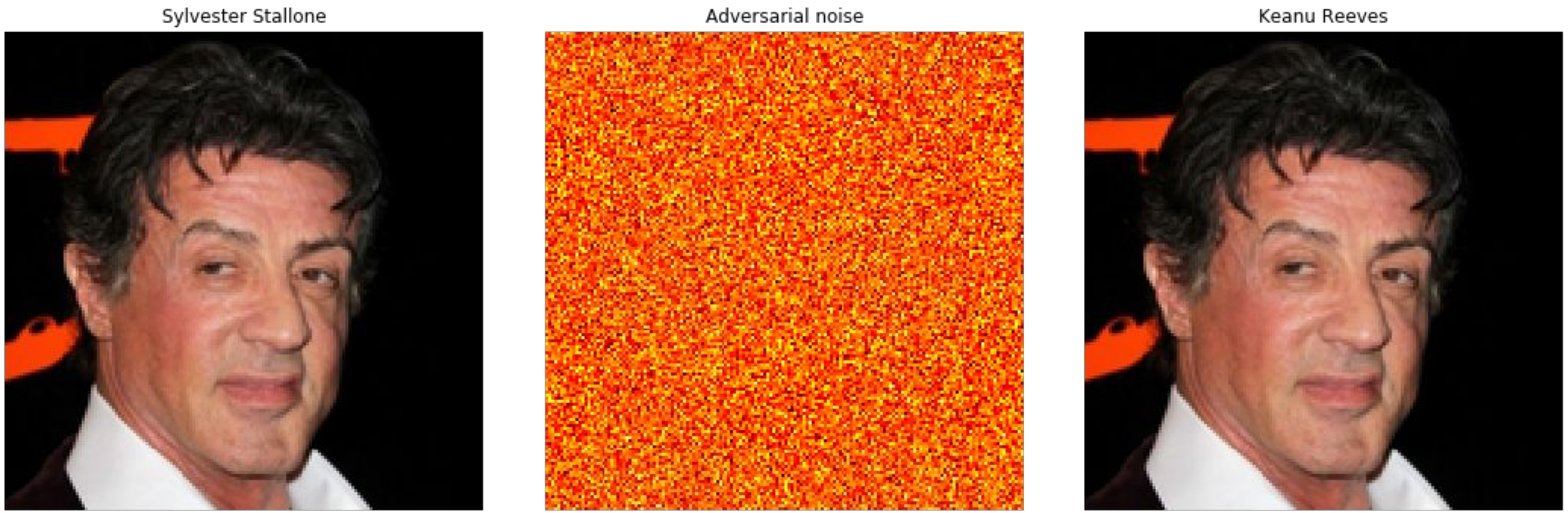


Image source: [Y combinator blog post on Adversarial attacks]()

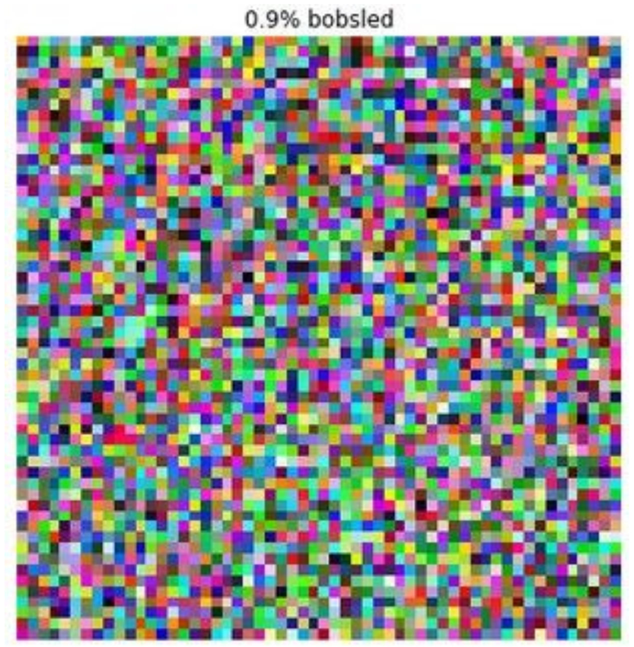
Non-targeted attack

$$\hat{y}(\vec{x}) = f_{\text{trained}}(\vec{x})$$

$$L = \frac{1}{2} \| \vec{y}_{goal} - \hat{y}(\vec{x}) \|_2^2$$

$$\vec{x}_{\text{adv}} = \arg\min_{\vec{x}} L(\vec{x})$$

No need in “signal” at all



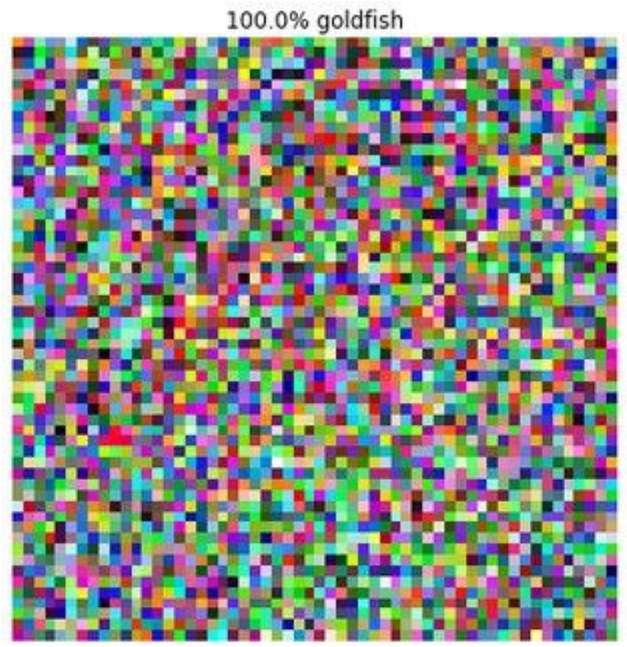


Image source: Breaking Linear Classifiers on ImageNet blogpost by Andrej Karpathy

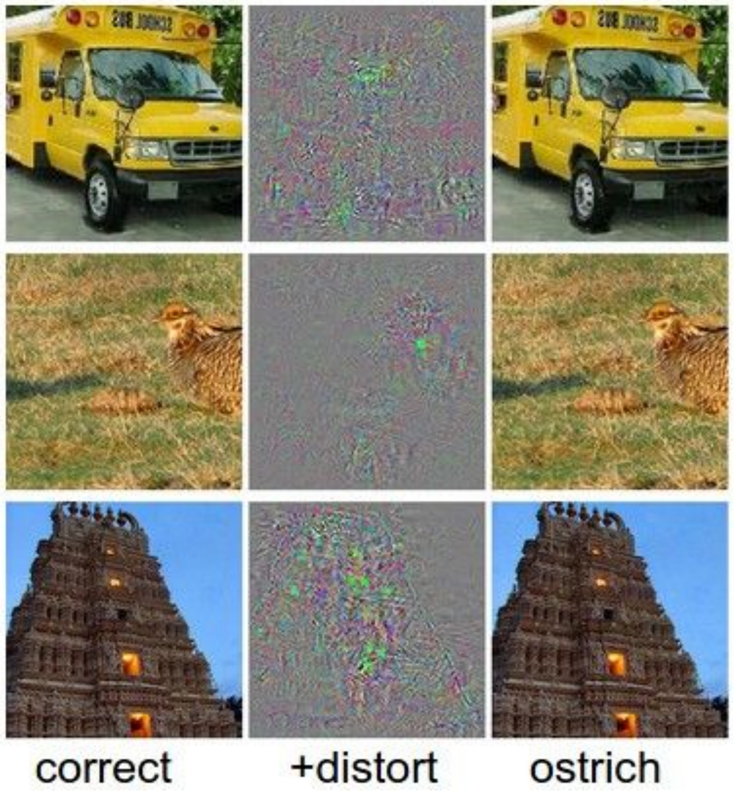
Targeted attack

$$\hat{y}(\vec{x}) = f_{\text{trained}}(\vec{x})$$

$$L = \|\vec{y}_{goal} - \hat{y}(\vec{x})\|_2^2 + \lambda\|\vec{x} - \vec{x}_{target}\|_2^2$$

$$\vec{x}_{\text{adv}} = \arg\min_{\vec{x}} L(\vec{x})$$

Targeted attack examples



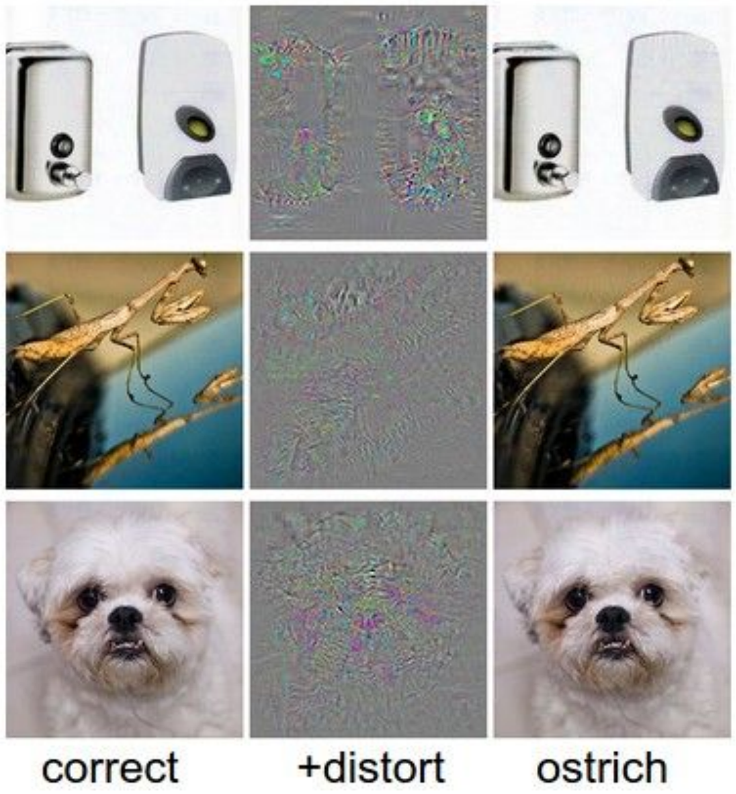


Image source: [Breaking Linear Classifiers on ImageNet blogpost by Andrej Karpathy]

Fast Gradient Sign Method

Let’s fix the model parameters $L(\vec{x}) \hat{=} L(\vec{x}, f, y_{\text{true}})$

If adversarial example is close to original $\vec{x}_{\text{adv}} = \vec{x} + \vec{\eta}$,

where $||\vec{\eta}||_\infty = \varepsilon$.

$$L(\vec{x}_{\text{adv}}) \approx L(\vec{x}) + \vec{\eta}^T \nabla_{\vec{x}} L(\vec{x})$$

Loss minimization delivers

$$\vec{\eta}_{\text{opt}} = \varepsilon \text{sign}(\nabla_{\vec{x}} L(\vec{x}))$$

FGSM applied to Logistic Regression

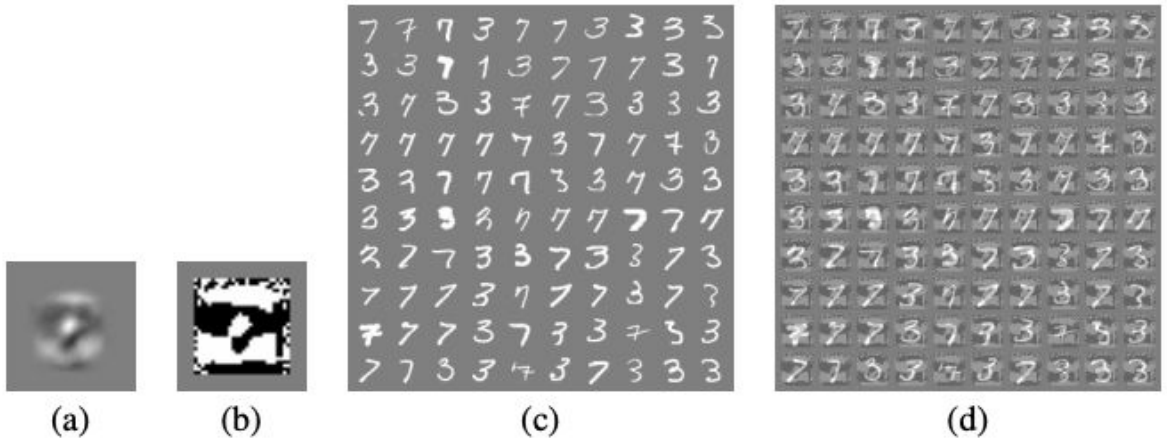
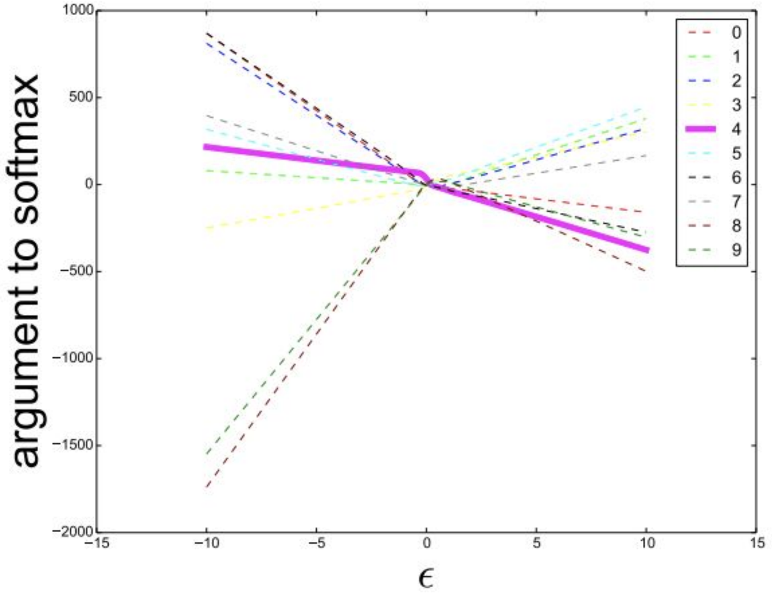


Figure 2: The fast gradient sign method applied to logistic regression (where it is not an approximation, but truly the most damaging adversarial example in the max norm box). a) The weights of a logistic regression model trained on MNIST. b) The sign of the weights of a logistic regression model trained on MNIST. This is the optimal perturbation. Even though the model has low capacity and is fit well, this perturbation is not readily recognizable to a human observer as having anything to do with the relationship between 3s and 7s. c) MNIST 3s and 7s. The logistic regression model has a 1.6% error rate on the 3 versus 7 discrimination task on these examples. d) Fast gradient sign adversarial examples for the logistic regression model with $\epsilon = .25$. The logistic regression model has an error rate of 99% on these examples.

Image source: [“Explaining and harnessing adversarial examples” by Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy]

Linear behaviour of the model



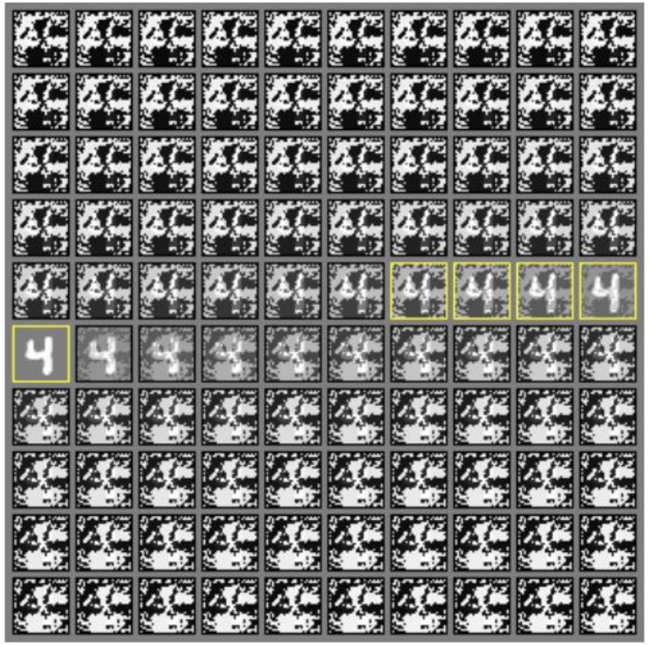
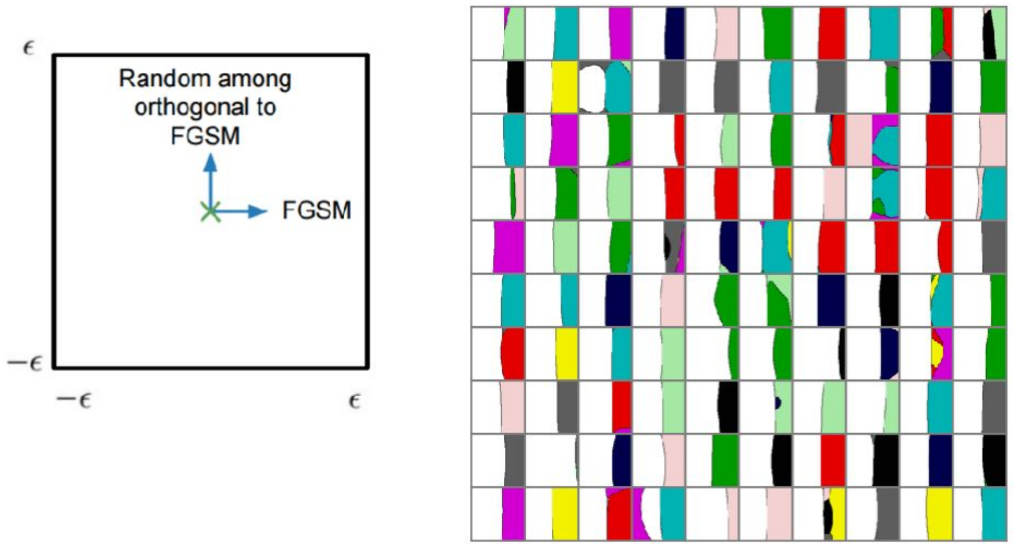


Image source: "Explaining and harnessing adversarial examples" by Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy

Linear behaviour of the model

Maps of Adversarial and Random Cross-Sections

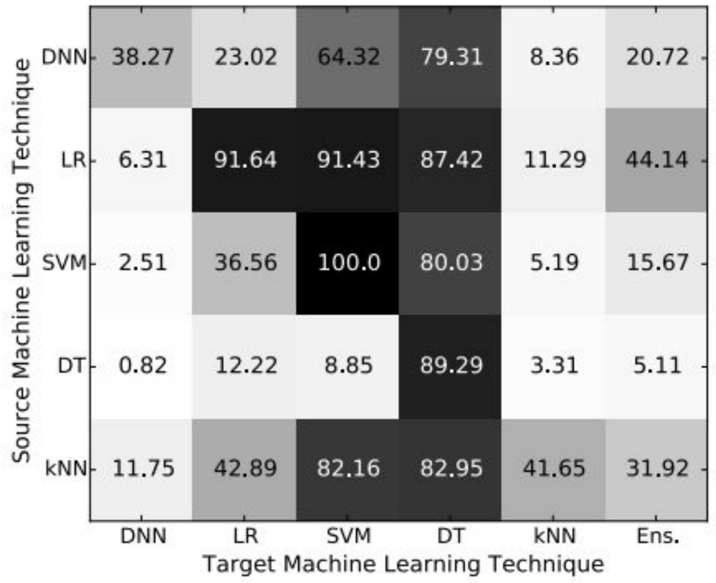


(collaboration with David Warde-Farley and Nicolas Papernot)

(Goodfellow 2016)

Image source: Y combinator blog post on Adversarial attacks, original authors Ian Goodfellow, David Warde-Farley & Nicolas Papernot

Cross-technique transferability



Source \ Target	DNN	LR	SVM	DT	kNN	Ens.
DNN	38.27	23.02	64.32	79.31	8.36	20.72
LR	6.31	91.64	91.43	87.42	11.29	44.14
SVM	2.51	36.56	100.0	80.03	5.19	15.67
DT	0.82	12.22	8.85	89.29	3.31	5.11
kNN	11.75	42.89	82.16	82.95	41.65	31.92

Figure 3: cross-technique Transferability matrix: cell (i, j) is the percentage of adversarial samples crafted to mislead a classifier learned using machine learning technique i that are misclassified by a classifier trained with technique j.

Image source: Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples, Nicolas Papernot, Partic McDanien & Ian Goodfellow

Attacking the black-box model

If one has access only to the model predictions, it still can be attacked.

1. Train the *mimicking model* on the predictions of *target model* (or even on the same dataset)

2. Attack the *mimicking model* with adversarial examples.

3. Deploy the adversarial examples against the *target model.*

Adversarial examples in human visual systems

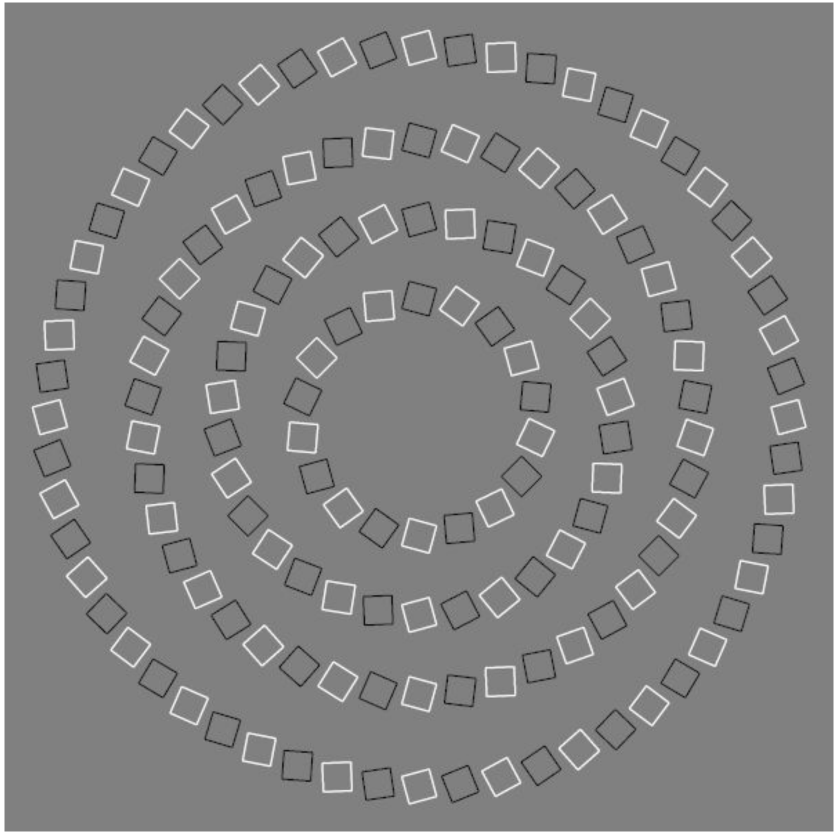


Image source: [Pinna, B., Gregory, R.L. (2002). "Shifts of Edges and Deformations of Patterns". Perception 31: 1503-1508]

Potential applications

- Virtual adversarial training to enhance models (~regularization)
- Model behaviour investigation (w.r.t. specific features if needed)
- Defences of adversarial attacks (training on both real and adversarial data)

Q & A

That’s all. Feel free to ask any questions.

Time for some practice.