Artificial Intelligence
Advanced Topics in AI & ML
ML Robustness and Adversarial Attacks

Aleksandr Petiushko

ML Research

Content

1. NN great success

Content

1. NN great success
2. NN lack of robustness

Content

1. NN great success
2. NN lack of robustness
3. ℓ_0-based adversaries

Content

1. NN great success
2. NN lack of robustness
3. ℓ_0-based adversaries
4. Adversarial examples in real world

Major approach

- Usual training:

$$\theta^{t+1} = \theta^{t} \color{red}{-} \eta \nabla_{\color{red}{\boldsymbol{\theta^t}}} L(f_{\theta^t}(x), y)$$

Major approach

- Usual training:

$$\theta^{t+1} = \theta^t \textcolor{red}{-} \eta \nabla_{\boldsymbol{\textcolor{red}{\theta^t}}} L(f_{\theta^t}(x), y)$$

- Adversarial perturbation:

$$x^{t+1} = x^t \textcolor{blue}{+} \eta \nabla_{\boldsymbol{\textcolor{blue}{x^t}}} L(f_\theta(x^t), y)$$

Robustness in Machine Learning

Robustness [informally]

Ability for a machine learning algorithm a to provide similar outputs on the similar data (i.e. having the same class or other invariant features)

Robustness in Machine Learning

Robustness [informally]

Ability for a machine learning algorithm a to provide similar outputs on the similar data (i.e. having the same class or other invariant features)

Two types of **Robustness** in ML:

Robustness in Machine Learning

Robustness [informally]

Ability for a machine learning algorithm a to provide similar outputs on the similar data (i.e. having the same class or other invariant features)

Two types of **Robustness** in ML:

Generalization

Dataset issue: algorithm needs to be robust if the dataset to evaluate it differs (sometimes significantly: we can treat it is a distribution shift) from the training dataset

Robustness in Machine Learning

Robustness [informally]

Ability for a machine learning algorithm a to provide similar outputs on the similar data (i.e. having the same class or other invariant features)

Two types of **Robustness** in ML:

Generalization

Dataset issue: algorithm needs to be robust if the dataset to evaluate it differs (sometimes significantly: we can treat it is a distribution shift) from the training dataset

Adversarial Robustness

Noise issue: algorithm needs to provide the similar output w.r.t. both clean and noisy images (where the model of noise is the topic to consider itself)

Robustness in Machine Learning

Robustness [informally]

Ability for a machine learning algorithm a to provide similar outputs on the similar data (i.e. having the same class or other invariant features)

Two types of **Robustness** in ML:

Generalization

Dataset issue: algorithm needs to be robust if the dataset to evaluate it differs (sometimes significantly: we can treat it is a distribution shift) from the training dataset

Adversarial Robustness

Noise issue: algorithm needs to provide the similar output w.r.t. both clean and noisy images (where the model of noise is the topic to consider itself)

For now we'll consider the **Adversarial Robustness**.

Questions to be answered

[1] Image credit: `https://spectrum.ieee.org`

Questions to be answered

Question1

How good is the human expert currently in comparison to Neural Nets for known datasets?

[1] Image credit: `https://spectrum.ieee.org`

Questions to be answered

Question1

How good is the human expert currently in comparison to Neural Nets for known datasets?

Question2

How stable are the current top neural net solutions subject to input data? How easy can they be fooled?

[1] Image credit: `https://spectrum.ieee.org`

Questions to be answered

Question1

How good is the human expert currently in comparison to Neural Nets for known datasets?

Question2

How stable are the current top neural net solutions subject to input data? How easy can they be fooled?

NN vs Human[1]

[1] Image credit: `https://spectrum.ieee.org`

Human expert VS NN

ImageNet[2] (1000-class image DB)

- Human expert top-5 error[3]: 5.1%
- NN top-5 error[4]: 0.98%

[2] http://www.image-net.org/
[3] Andrej Karpathy blog
[4] Yuan, Lu, et al. "Florence: A new foundation model for computer vision." 2021
[5] http://vis-www.cs.umass.edu/lfw/
[6] Kumar N. et al. "Attribute and simile classifiers for face verification." 2009
[7] Deng J. et al. "Arcface: Additive angular margin loss for deep face recognition." 2018

Human expert VS NN

ImageNet[2] (1000-class image DB)

- Human expert top-5 error[3]: 5.1%
- NN top-5 error[4]: 0.98%

Labeled Faces in the Wild[5] (famous faces DB)

- Human expert verification error[6]: 2.47%
- NN verification error[7]: 0.17%

---

[2] http://www.image-net.org/
[3] Andrej Karpathy blog
[4] Yuan, Lu, et al. "Florence: A new foundation model for computer vision." 2021
[5] http://vis-www.cs.umass.edu/lfw/
[6] Kumar N. et al. “Attribute and simile classifiers for face verification.” 2009
[7] Deng J. et al. “Arcface: Additive angular margin loss for deep face recognition.” 2018

Human expert VS NN

ImageNet[2] (1000-class image DB)

- Human expert top-5 error[3]: 5.1%
- NN top-5 error[4]: 0.98%

Labeled Faces in the Wild[5] (famous faces DB)

- Human expert verification error[6]: 2.47%
- NN verification error[7]: 0.17%

LFW

[2] http://www.image-net.org/
[3] Andrej Karpathy blog
[4] Yuan, Lu, et al. "Florence: A new foundation model for computer vision." 2021
[5] http://vis-www.cs.umass.edu/lfw/
[6] Kumar N. et al. “Attribute and simile classifiers for face verification.” 2009
[7] Deng J. et al. “Arcface: Additive angular margin loss for deep face recognition.” 2018

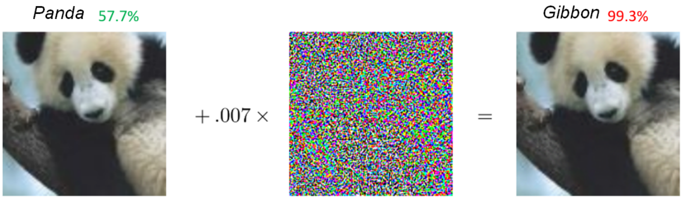
NN instability

- It turned out that one can add to the input almost invisible to the human eye perturbation in such a way that this perturbation completely changes the NN output

[8] Image credit: https://arxiv.org/pdf/1412.6572.pdf

NN instability

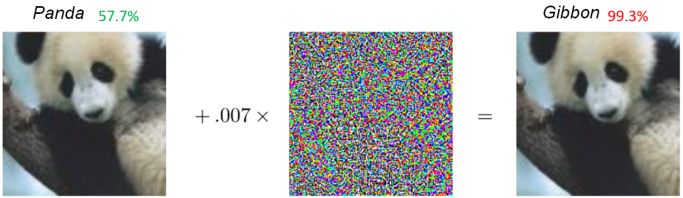
- It turned out that one can add to the input almost invisible to the human eye perturbation in such a way that this perturbation completely changes the NN output
- E.g. classification result from "Panda" changes to "Gibbon"[8]



[8] Image credit: `https://arxiv.org/pdf/1412.6572.pdf`

NN instability

- It turned out that one can add to the input almost invisible to the human eye perturbation in such a way that this perturbation completely changes the NN output
- E.g. classification result from "Panda" changes to "Gibbon"[8]

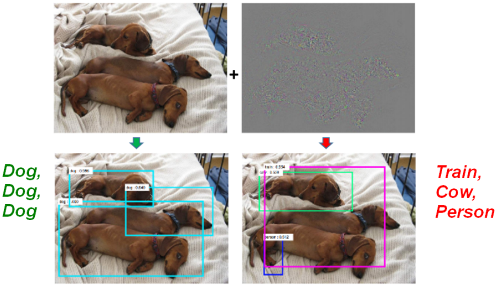


Such almost invisible perturbations leading to changing of the NN output are called **adversarial examples** / **perturbations** (or **adversarial attacks** on NN)

[8] Image credit: `https://arxiv.org/pdf/1412.6572.pdf`

Different types of NN to attack

Detection and segmentation[9] NNs:

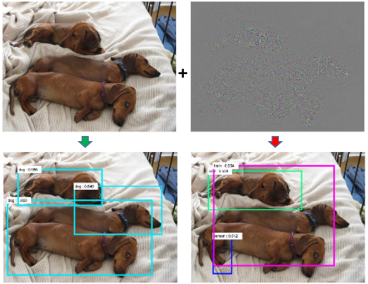


[9] Xie C. et al. "Adversarial examples for semantic segmentation and object detection." 2017
[10] Jia R. et al. "Adversarial examples for evaluating reading comprehension systems." 2017

Different types of NN to attack

Detection and segmentation[9] NNs:

Dog,
Dog,
Dog



Train,
Cow,
Person

And even NN for text processing (QA, question answering systems)[10]:

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

[9] Xie C. et al. "Adversarial examples for semantic segmentation and object detection." 2017
[10] Jia R. et al. "Adversarial examples for evaluating reading comprehension systems." 2017

One of the main sources of adversarial examples to exist

- One of the main reasons for this neural nets behavior on similar inputs: **lack of NN robustness**

[11] Image credit: `https://secml.github.io/`

[12] Fawzi, Alhussein, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Robustness of classifiers: from adversarial to random noise." 2016

One of the main sources of adversarial examples to exist

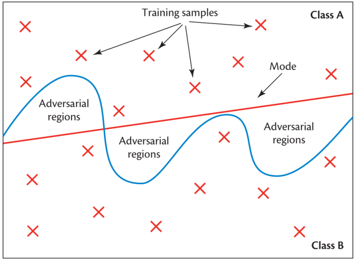
- One of the main reasons for this neural nets behavior on similar inputs: **lack of NN robustness**
- Namely, the separating classification boundaries often are very close to the training examples, and it is very easy to "go abroad"[11,12]

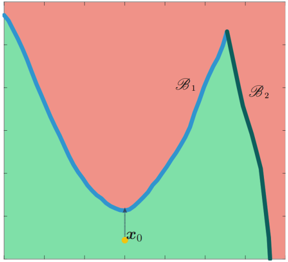
[11] Image credit: `https://secml.github.io/`

[12] Fawzi, Alhussein, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Robustness of classifiers: from adversarial to random noise." 2016

One of the main sources of adversarial examples to exist

- One of the main reasons for this neural nets behavior on similar inputs: **lack of NN robustness**
- Namely, the separating classification boundaries often are very close to the training examples, and it is very easy to "go abroad"[11,12]





[11]Image credit: `https://secml.github.io/`

[12]Fawzi, Alhussein, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Robustness of classifiers: from adversarial to random noise." 2016

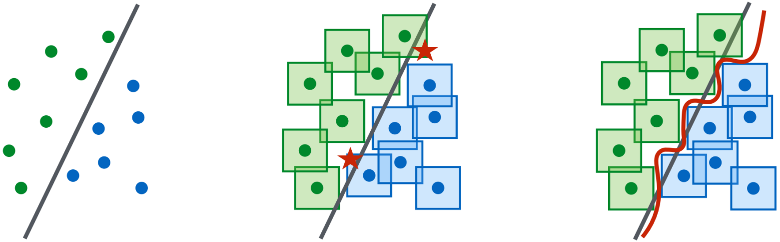
Easy defense method

- We can change the output of classification neural net by a subtle per-pixel perturbation $\Rightarrow$ why not to add for any training example its whole per-pixel vicinity (based on some norm — e.g., ℓ_∞) during the training process[13]

[13] Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." 2017

Easy defense method

- We can change the output of classification neural net by a subtle per-pixel perturbation $\Rightarrow$ why not to add for any training example its whole per-pixel vicinity (based on some norm — e.g., ℓ_∞) during the training process[13]



[13] Madry, Aleksander, et al. “Towards deep learning models resistant to adversarial attacks.” 2017

Easy, but not realistic defense method

- Suppose the input image size is 100×100 pixels, 3 RGB colors (8-bit color depth)

Easy, but not realistic defense method

- Suppose the input image size is 100×100 pixels, 3 RGB colors (8-bit color depth)
- Suppose that human eye cannot distinguish the changes in color of 2 units out of 256: at every pixel we can allow for ± 1 perturbation for every color channel

Easy, but not realistic defense method

- Suppose the input image size is 100×100 pixels, 3 RGB colors (8-bit color depth)
- Suppose that human eye cannot distinguish the changes in color of 2 units out of 256: at every pixel we can allow for ± 1 perturbation for every color channel
- Then for every training example we need to add into the training dataset the following number of its "pixel neighbors"

$$2^{3\times 100\times 100} = 2^{30000} = (2^{10})^{3000} \approx (10^3)^{3000} = 10^{9000}$$

Easy, but not realistic defense method

- Suppose the input image size is 100×100 pixels, 3 RGB colors (8-bit color depth)
- Suppose that human eye cannot distinguish the changes in color of 2 units out of 256: at every pixel we can allow for ± 1 perturbation for every color channel
- Then for every training example we need to add into the training dataset the following number of its "pixel neighbors"

$$2^{3 \times 100 \times 100} = 2^{30000} = (2^{10})^{3000} \approx (10^3)^{3000} = 10^{9000}$$

- It is much more the number of atoms in the observable part of the Universe (10^{80})!

Easy, but not realistic defense method

- Suppose the input image size is 100×100 pixels, 3 RGB colors (8-bit color depth)
- Suppose that human eye cannot distinguish the changes in color of 2 units out of 256: at every pixel we can allow for ± 1 perturbation for every color channel
- Then for every training example we need to add into the training dataset the following number of its "pixel neighbors"

$$2^{3\times 100\times 100} = 2^{30000} = (2^{10})^{3000} \approx (10^3)^{3000} = 10^{9000}$$

- It is much more the number of atoms in the observable part of the Universe (10^{80})!
- As a consequence, not very realistic

Realistic defense method

- Let us not iterate over the entire vicinity of the training example, but to use only those points from example vicinity, which are the closest to the separating surface

[14] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." 2014

Realistic defense method

- Let us not iterate over the entire vicinity of the training example, but to use only those points from example vicinity, which are the closest to the separating surface
- This training method is called **A**dversarial **T**raining (AT)[14]

[14] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples.” 2014

Realistic defense method

- Let us not iterate over the entire vicinity of the training example, but to use only those points from example vicinity, which are the closest to the separating surface
- This training method is called **A**dversarial **T**raining (AT)[14]

Adversarial training: pros

- No need in iterating over the vicinity of huge cardinality
- In general, protects against the method of adversarial example generation

[14] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." 2014

Realistic defense method

- Let us not iterate over the entire vicinity of the training example, but to use only those points from example vicinity, which are the closest to the separating surface
- This training method is called **A**dversarial **T**raining (AT)[14]

Adversarial training: pros

- No need in iterating over the vicinity of huge cardinality
- In general, protects against the method of adversarial example generation

Adversarial training: cons

- Procedure of *good* adversarial examples generation usually works slowly (significantly slower than 1 backprop iteration)
- Protects **only** against the method of generating adversarial examples used for adversarial training

[14] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." 2014

Definitions

- $x \in B = [0,1]^{C \times M \times N}$ — input image $C \times M \times N$, where C — number of color channels (1 for grayscale, 3 for RGB)
- y — correct class label for input x
- θ — parameters of NN-classifier
- $L(\theta, x, y)$ — loss function
- $f(x)$ — output of classifier (recognized class), and we are trying to make $f(x) = y$ when training

Definitions

- $x \in B = [0,1]^{C\times M\times N}$ — input image $C \times M \times N$, where C — number of color channels (1 for grayscale, 3 for RGB)
- y — correct class label for input x
- θ — parameters of NN-classifier
- $L(\theta, x, y)$ — loss function
- $f(x)$ — output of classifier (recognized class), and we are trying to make $f(x) = y$ when training
- $r \in B = [0,1]^{C\times M\times N}$ — the additive perturbation for the input x

Definition of adversarial example and robustness

Goal of adversarial attack

To change the output of the classifier f from the correct class label to the incorrect one by means of minimal in terms of some norm ℓ_p perturbation r:

1. $||r||_p \to \min$ so as:

Definition of adversarial example and robustness

Goal of adversarial attack

To change the output of the classifier f from the correct class label to the incorrect one by means of minimal in terms of some norm ℓ_p perturbation r:

1. $||r||_p \to \min$ so as:
2. $f(x) = y$ (initially the output is correct)
3. $f(x + r) \neq y$ ("break" the NN output with perturbation r)
4. $x + r \in B$ (still in the space of correct images)

Definition of adversarial example and robustness

Goal of adversarial attack

To change the output of the classifier f from the correct class label to the incorrect one by means of minimal in terms of some norm ℓ_p perturbation r:

1. $||r||_p \to \min$ so as:
2. $f(x) = y$ (initially the output is correct)
3. $f(x+r) \neq y$ ("break" the NN output with perturbation r)
4. $x + r \in B$ (still in the space of correct images)

Classifier robustness

To find the perturbation class $S(x, f) \subseteq B$ so as the classifier will not change its output:

$$f(x+r) = f(x) = y \quad \forall r \in S(x, f)$$

ℓ_p norms

Let us remind the most used ℓ_p norms for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$:

[15] **Exercise**: Prove it

ℓ_p norms

Let us remind the most used ℓ_p norms for $x = (x_1, \dots, x_n) \in \mathbb{R}^n$:

- ℓ_2: $||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$

[15] **Exercise**: Prove it

ℓ_p norms

Let us remind the most used ℓ_p norms for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$:

- ℓ_2: $||x||_2 = \sqrt{\sum_{i=1}^n x_i^2}$
- ℓ_1: $||x||_1 = \sum_{i=1}^n |x_i|$

[15] **Exercise**: Prove it

ℓ_p norms

Let us remind the most used ℓ_p norms for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$:

- ℓ_2: $||x||_2 = \sqrt{\sum_{i=1}^n x_i^2}$
- ℓ_1: $||x||_1 = \sum_{i=1}^n |x_i|$
- ℓ_∞: $||x||_\infty = \max_i |x_i|$

[15] **Exercise**: Prove it

ℓ_p norms

Let us remind the most used ℓ_p norms for $x = (x_1, \dots, x_n) \in \mathbb{R}^n$:

- ℓ_2: $||x||_2 = \sqrt{\sum_{i=1}^n x_i^2}$
- ℓ_1: $||x||_1 = \sum_{i=1}^n |x_i|$
- ℓ_∞: $||x||_\infty = \max_i |x_i|$
- ℓ_0: $||x||_0 = \sum_{i=1}^n \mathbf{1}_{x_i \neq 0}$

[15] **Exercise**: Prove it

ℓ_p norms

Let us remind the most used ℓ_p norms for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$:

- ℓ_2: $||x||_2 = \sqrt{\sum_{i=1}^n x_i^2}$
- ℓ_1: $||x||_1 = \sum_{i=1}^n |x_i|$
- ℓ_∞: $||x||_\infty = \max_i |x_i|$
- ℓ_0: $||x||_0 = \sum_{i=1}^n \mathbf{1}_{x_i \neq 0}$

Remark. For $0 < p < 1$ the functional ℓ_p such as $||x||_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$ is not a norm[15]

[15] **Exercise**: Prove it

Adversarial examples taxonomy (1)

Target

- **Untargeted**: only need to change the classifier's output
- **Targeted**: need to change the classifier's output to the specific class y_t

Adversarial examples taxonomy (1)

Target

- **Untargeted**: only need to change the classifier's output
- **Targeted**: need to change the classifier's output to the specific class y_t

Awareness

- **White-box**: adversary knows everything about the classifier (architecture and its weights)
- **Black-box**: adversary knows only partial details about the classifier (usually only classifier's output)

Adversarial examples taxonomy (1)

Target

- **Untargeted**: only need to change the classifier's output
- **Targeted**: need to change the classifier's output to the specific class y_t

Awareness

- **White-box**: adversary knows everything about the classifier (architecture and its weights)
- **Black-box**: adversary knows only partial details about the classifier (usually only classifier's output)

Application

- **Digital**: for digital record application only (e.g., photo of an object)
- **Real-world**: to be applied in the real environment (e.g., the object itself)

Adversarial examples taxonomy (2)

Universality

- **Input-aware**: the perturbation r depends on the input x
- **Universal**: the perturbation r is to be applied for any input x

Adversarial examples taxonomy (2)

Universality

- **Input-aware**: the perturbation r depends on the input x
- **Universal**: the perturbation r is to be applied for any input x

Transfer

- **Non-transferable**: successfully working only for a small range of classifiers
- **Transferable**: successfully working for a wide range of classifiers (but at the same time can be non-universal ones)

- The hardest case — targeted black-box real-world universal transferable adversarial examples
- This part will be devoted to white-box adversarial examples

Adversarial examples: success criteria

Let us introduce the success criteria $S(A, Z)$ of an algorithm A for generating adversarial examples $r_A(x)$ using the set $Z \ni (x^i, y^i)$:

Adversarial examples: success criteria

Let us introduce the success criteria $S(A, Z)$ of an algorithm A for generating adversarial examples $r_A(x)$ using the set $Z \ni (x^i, y^i)$:

- For untargeted adversarial examples:

$$S(A, Z) = \frac{\sum_i \mathbf{1}\{f(x^i) = y^i\} \cdot \mathbf{1}\{f(x^i + r_A(x^i)) \neq y^i\}}{\sum_i \mathbf{1}\{f(x^i) = y^i\}}$$

Adversarial examples: success criteria

Let us introduce the success criteria $S(A, Z)$ of an algorithm A for generating adversarial examples $r_A(x)$ using the set $Z \ni (x^i, y^i)$:

- For untargeted adversarial examples:

$$S(A, Z) = \frac{\sum_i \mathbf{1}\{f(x^i) = y^i\} \cdot \mathbf{1}\{f(x^i + r_A(x^i)) \neq y^i\}}{\sum_i \mathbf{1}\{f(x^i) = y^i\}}$$

- For adversarial examples targeted towards the class y_t:

$$S(A, Z, y_t) = \frac{\sum_i \mathbf{1}\{f(x^i) = y^i\} \cdot \mathbf{1}\{f(x^i + r_A(x^i)) = y_t\}}{\sum_i \mathbf{1}\{f(x^i) = y^i\}}$$

Adversarial examples: success criteria

Let us introduce the success criteria $S(A, Z)$ of an algorithm A for generating adversarial examples $r_A(x)$ using the set $Z \ni (x^i, y^i)$:

- For untargeted adversarial examples:

$$S(A, Z) = \frac{\sum_i \mathbf{1}\{f(x^i) = y^i\} \cdot \mathbf{1}\{f(x^i + r_A(x^i)) \neq y^i\}}{\sum_i \mathbf{1}\{f(x^i) = y^i\}}$$

- For adversarial examples targeted towards the class y_t:

$$S(A, Z, y_t) = \frac{\sum_i \mathbf{1}\{f(x^i) = y^i\} \cdot \mathbf{1}\{f(x^i + r_A(x^i)) = y_t\}}{\sum_i \mathbf{1}\{f(x^i) = y^i\}}$$

Remark. Apparently $S(A, Z, y_t) \leq S(A, Z)$

Adversarial examples on images: forerunner

- Initially CNN robustness was studied towards how adequate the output of CNN using different inputs

[16] Nguyen, Anh, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images.” 2014

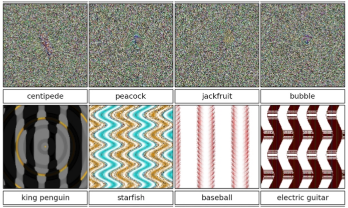
Adversarial examples on images: forerunner

- Initially CNN robustness was studied towards how adequate the output of CNN using different inputs
- It was turned out that some examples (structured ot not) exist that produce as the output of CNN any class with any probability

[16] Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." 2014

Adversarial examples on images: forerunner

- Initially CNN robustness was studied towards how adequate the output of CNN using different inputs
- It was turned out that some examples (structured ot not) exist that produce as the output of CNN any class with any probability
- Such examples were called "fooling images"[16] and were generated by evolutionary algorithms



[16] Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." 2014

Generation method: FGSM

- **Proposition**: to use only the linear part of loss function in the vicinity of x and walk by its gradient — FGSM[17] (**F**ast **G**radient **S**ign **M**ethod):

$$r = \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y_t))$$

where $0 < \epsilon < 1$ — some constant

[17] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." 2014

Generation method: FGSM

- **Proposition**: to use only the linear part of loss function in the vicinity of x and walk by its gradient — FGSM[17] (**F**ast **G**radient **S**ign **M**ethod):

$$r = \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y_t))$$

where $0 < \epsilon < 1$ — some constant

- **Reminder**: for NN weights optimization the backpropagation method is used where the gradient is taken w.r.t. NN weights, i.e. $\nabla_\theta L(\theta, x, y)$

[17] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." 2014

Generation method: FGSM

- **Proposition**: to use only the linear part of loss function in the vicinity of x and walk by its gradient — FGSM[17] (**F**ast **G**radient **S**ign **M**ethod):

$$r = \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y_t))$$

  where $0 < \epsilon < 1$ — some constant
- **Reminder**: for NN weights optimization the backpropagation method is used where the gradient is taken w.r.t. NN weights, i.e. $\nabla_\theta L(\theta, x, y)$
- For most CNNs the norm ℓ_∞ is used because it is correlated with the process of how a human eye perceive the visual information

[17] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." 2014

Generation method: One pixel

- One pixel adversarial example[18] — extreme case of ℓ_0-based generation method

[18] Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks." 2017

[19] Storn, Rainer, and Kenneth Price. "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces." 1997

Generation method: One pixel

- One pixel adversarial example[18] — extreme case of ℓ_0-based generation method
- **Idea**: to apply the evolutionary algorithm (differential evolution[19])

[18] Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. “One pixel attack for fooling deep neural networks.” 2017

[19] Storn, Rainer, and Kenneth Price. “Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces.” 1997

Generation method: One pixel

- One pixel adversarial example[18] — extreme case of ℓ_0-based generation method
- **Idea**: to apply the evolutionary algorithm (differential evolution[19])
- Population consists of 400 examples each defined by 5 numbers: 2 coordinates and 3 color channels

[18] Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks." 2017

[19] Storn, Rainer, and Kenneth Price. "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces." 1997

Generation method: One pixel

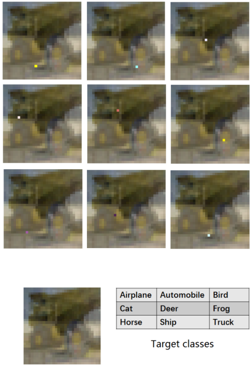
- One pixel adversarial example[18] — extreme case of ℓ_0-based generation method
- **Idea**: to apply the evolutionary algorithm (differential evolution[19])
- Population consists of 400 examples each defined by 5 numbers: 2 coordinates and 3 color channels
- Offspring generation — the linear combination of 3 random parents

[18] Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks." 2017

[19] Storn, Rainer, and Kenneth Price. "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces." 1997

Generation method: One pixel

- One pixel adversarial example[18] — extreme case of ℓ_0-based generation method
- **Idea**: to apply the evolutionary algorithm (differential evolution[19])
- Population consists of 400 examples each defined by 5 numbers: 2 coordinates and 3 color channels
- Offspring generation — the linear combination of 3 random parents



Airplane	Automobile	Bird
Cat	Deer	Frog
Horse	Ship	Truck

Target classes

Original image (dog)

[18] Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks." 2017

[19] Storn, Rainer, and Kenneth Price. "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces." 1997

Universal adversarial examples

- Until now all adversarial examples have been constructed as a function of input x

[20] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." 2016

Universal adversarial examples

- Until now all adversarial examples have been constructed as a function of input x
- It turns out that it is possible to build so called "universal" example[20] which is the function of the whole training dataset X

[20] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." 2016

Universal adversarial examples

- Until now all adversarial examples have been constructed as a function of input x
- It turns out that it is possible to build so called "universal" example[20] which is the function of the whole training dataset X
- **Idea**: to find perturbation r approximately equally far away from all classes from X

[20] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." 2016

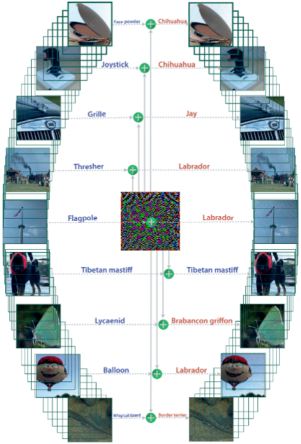
Universal adversarial examples

- Until now all adversarial examples have been constructed as a function of input x
- It turns out that it is possible to build so called "universal" example[20] which is the function of the whole training dataset X
- **Idea**: to find perturbation r approximately equally far away from all classes from X
- To the right is the universal perturbation for every input changing the output of the classifier

[20] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." 2016

Universal adversarial examples

- Until now all adversarial examples have been constructed as a function of input x
- It turns out that it is possible to build so called "universal" example[20] which is the function of the whole training dataset X
- **Idea**: to find perturbation r approximately equally far away from all classes from X
- To the right is the universal perturbation for every input changing the output of the classifier



[20] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." 2016

Taxonomy of generation methods

In general, the adversarial examples generation methods (in Computer Vision) can be divided into the following types:

Taxonomy of generation methods

In general, the adversarial examples generation methods (in Computer Vision) can be divided into the following types:

- Using ℓ_2-based norm (including geometric ones): most convenient for classical optimization

Taxonomy of generation methods

In general, the adversarial examples generation methods (in Computer Vision) can be divided into the following types:

- Using ℓ_2-based norm (including geometric ones): most convenient for classical optimization
- Using ℓ_∞-based norm: correspond to perception process by the human eye of any visual information

Taxonomy of generation methods

In general, the adversarial examples generation methods (in Computer Vision) can be divided into the following types:

- Using ℓ_2-based norm (including geometric ones): most convenient for classical optimization
- Using ℓ_∞-based norm: correspond to perception process by the human eye of any visual information
- Using ℓ_0-based norm: the area of perturbation is minimized, but not the delta value per pixel

Anyway it is not enough for generation of physically plausible adversarial examples.

Real-world adversarial examples (1)

- All adversarial examples until now were designed to work in digital domain: e.g. to change the image on a pixel level

[21] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. “Adversarial examples in the physical world.” 2016

Real-world adversarial examples (1)

- All adversarial examples until now were designed to work in digital domain: e.g. to change the image on a pixel level
- If there is no possibility to change the image before feeding it into NN, then the digital method is useless

[21] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. “Adversarial examples in the physical world.” 2016

Real-world adversarial examples (1)

- All adversarial examples until now were designed to work in digital domain: e.g. to change the image on a pixel level
- If there is no possibility to change the image before feeding it into NN, then the digital method is useless
- That's why **real-world** (or physical) adversarial examples are the most universal ones

[21] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." 2016

Real-world adversarial examples (1)

- All adversarial examples until now were designed to work in digital domain: e.g. to change the image on a pixel level
- If there is no possibility to change the image before feeding it into NN, then the digital method is useless
- That's why **real-world** (or physical) adversarial examples are the most universal ones
- The first try of real-world adversarial examples[21] — generation of an image in the digital domain, then printing it out on the physical carrier (paper sheet), then photo by digital camera and finally NN recognition

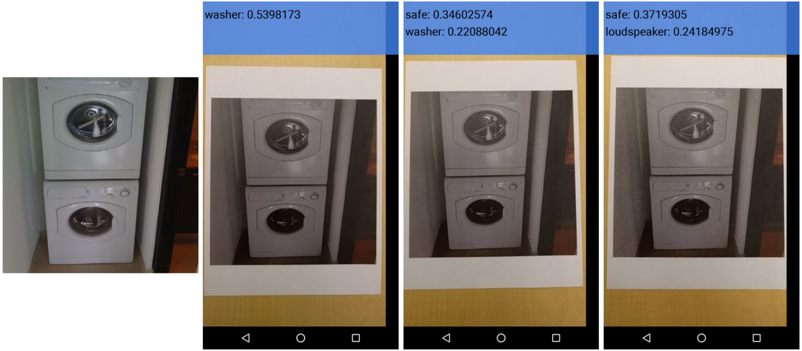
[21] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." 2016

Real-world adversarial examples (1)

- All adversarial examples until now were designed to work in digital domain: e.g. to change the image on a pixel level
- If there is no possibility to change the image before feeding it into NN, then the digital method is useless
- That's why **real-world** (or physical) adversarial examples are the most universal ones
- The first try of real-world adversarial examples[21] — generation of an image in the digital domain, then printing it out on the physical carrier (paper sheet), then photo by digital camera and finally NN recognition
- No any specific technology to generate the real-world adversarial examples was proposed: only its existence was shown

[21] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." 2016

Real-world adversarial examples (2)



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Adversarial examples in real world: EOT

- Don't have the control on the image pixels after the photo $\Rightarrow$ the only option is to change the object appearance itself

[22] Athalye A. et al. "Synthesizing robust adversarial examples." 2017

Adversarial examples in real world: EOT

- Don't have the control on the image pixels after the photo $\Rightarrow$ the only option is to change the object appearance itself
- **E**xpectation **O**ver **T**ransformation (EOT)[22] to the rescue — takes into account the transformations of objects in the real world, e.g.:
  - Different scaling factors
  - Random translation and rotation
  - Luminosity / contrast variation, noise etc

[22] Athalye A. et al. "Synthesizing robust adversarial examples." 2017

Adversarial examples in real world: EOT

- Don't have the control on the image pixels after the photo $\Rightarrow$ the only option is to change the object appearance itself
- **E**xpectation **O**ver **T**ransformation (EOT)[22] to the rescue — takes into account the transformations of objects in the real world, e.g.:
  - Different scaling factors
  - Random translation and rotation
  - Luminosity / contrast variation, noise etc
- So for the object x in the real world the task is to find the adversarial perturbation r taking into account transformation $g \in T$:

EOT

Find $\arg\min_r \mathbb{E}_{g\sim T}[P(y|g(x+r))]$ s.t.:

1. $\mathbb{E}_{g\sim T}[d(g(x+r), g(x))] < \epsilon$, where $d(a,b)$ – some distance function (e.g. $d(a,b) = ||a-b||_p$)
2. $x + r \in B$

[22] Athalye A. et al. "Synthesizing robust adversarial examples." 2017

Physical adversarial examples: key ingredients

- ℓ_0-optimization (mask-based) + EOT: <u>the must</u>

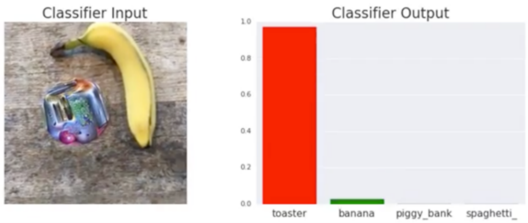
Physical adversarial examples: key ingredients

- ℓ_0-optimization (mask-based) + EOT: the must
- **T**otal **V**ariation (TV) loss — penalty for the perturbation to be non-smooth (in the real world there is no distinct pixel gradients):

$$TV(x) = \sum_{i,j} \sqrt{(x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2}$$

Examples of physical adversarial examples

Attack on ImageNet objects[23]:

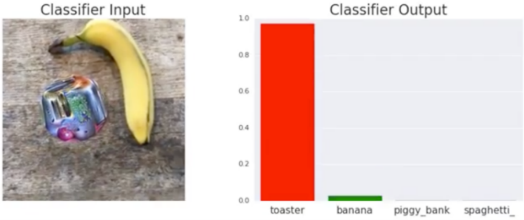


[23] Brown T. et al. "Adversarial patch." 2017

[24] Eykholt K. et al. "Robust physical-world attacks on deep learning models." 2017

Examples of physical adversarial examples

Attack on ImageNet objects[23]:



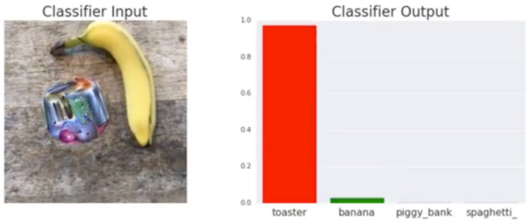
Attack on road signs[24]:

[23] Brown T. et al. “Adversarial patch.” 2017

[24] Eykholt K. et al. “Robust physical-world attacks on deep learning models.” 2017

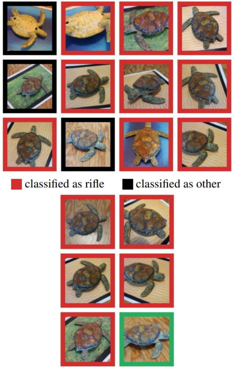
Examples of physical adversarial examples

Attack on ImageNet objects[23]:



Attack on road signs[24]:

3D adversarial objects:



[23] Brown T. et al. “Adversarial patch.” 2017

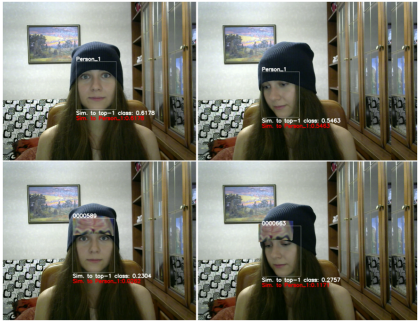
[24] Eykholt K. et al. “Robust physical-world attacks on deep learning models.” 2017

AdvHat[25] — invisibility hat

Due to the better projection procedure and richer color information, the attack is robust to rotations and brightness variation

Frontal face (advhat: no)
Similarity to origin: **0.61**

Frontal face (advhat: yes)
Similarity to origin: **0.02**
Similarity to other: **0.23**



Rotated face (advhat: no)
Similarity to origin: **0.54**

Rotated face (advhat: yes)
Similarity to origin: **0.11**
Similarity to other: **0.27**

[25] Komkov S. et al. "Advhat: Real-world adversarial attack on arcface face id system." 2019

Takeaway notes

1. NNs for now are much better than human expert in controlled conditions

Takeaway notes

1. NNs for now are much better than human expert in controlled conditions
2. NNs are unstable w.r.t. its input

Takeaway notes

1. NNs for now are much better than human expert in controlled conditions
2. NNs are unstable w.r.t. its input
3. Digital $\rightarrow$ physical domain attack translation is hard

Takeaway notes

1. NNs for now are much better than human expert in controlled conditions
2. NNs are unstable w.r.t. its input
3. Digital $\rightarrow$ physical domain attack translation is hard
4. But even the most successful face ID systems can be fooled by a simple grayscale patch from common printer

Takeaway notes

1. NNs for now are much better than human expert in controlled conditions
2. NNs are unstable w.r.t. its input
3. Digital $\rightarrow$ physical domain attack translation is hard
4. But even the most successful face ID systems can be fooled by a simple grayscale patch from common printer
5. ℓ_0-based local attack + TV loss + EOT are the must

Thank you!