CS230: Lecture 4
Attacking Networks with Adversarial Examples
- Generative Adversarial Networks
  Kian Katanforoosh
I. Attacking NNs with Adversarial Examples
II. Generative Adversarial Networks
I. Adversarial examples

Discovery (2013): several machine learning models, including state-of-the-art neural networks, are vulnerable to adversarial examples

A. Attacking a network with adversarial examples

B. Defenses against adversarial examples

C. Why are neural networks vulnerable to adversarial examples?
I. A. Attacking a network with adversarial examples

**Goal**: Given a network pretrained on ImageNet, find an input image that is not an iguana but will be classified as an iguana.

1. **Rephrasing what we want:**
   
   Find $x$ such that: $\hat{y}(x) = y_{iguana} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

2. **Defining the loss function**
   
   $L(\hat{y}, y) = \frac{1}{2}\|\hat{y}(W, b, x) - y_{iguana}\|^2$

3. **Optimize the image**
   
   After many iterations:
   
   $x = x - \alpha \frac{\partial L}{\partial x}$

---

[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]
Question: Will the forged image $x$ look like an iguana?

$256^{32 \times 32 \times 3} \approx 10^{7400}$

Space of possible input images

Space of real images

Space of images classified as iguanas
I. A. Attacking a network with adversarial examples

**Goal**: Given a network pretrained on ImageNet, find an input image that is a cat but will be classify as an iguana.

1. Rephrasing what we want:
   
   Find $x$ such that: $\hat{y}(x) = y_{\text{iguana}} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$
   
   And: $x = x_{\text{cat}}$

2. Defining the loss function

   $L(\hat{y}, y) = \frac{1}{2} \left\| \hat{y}(W, b, x) - y_{\text{iguana}} \right\|^2_2 + \lambda \left\| x - x_{\text{cat}} \right\|^2_2$

3. Optimize the image

   After many iterations

   $x = x - \alpha \frac{\partial L}{\partial x}$

[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]
I. A. Attacking a network with adversarial examples

92% Cat

94% Iguana
I. A. Attacking a network with adversarial examples

$256^{32 \times 32 \times 3} \approx 10^{7400}$

Space of possible input images

Space of images classified as iguanas

Space of real images

Space of images that look real to humans
Adversarial Examples In The Physical World

[Alexey Kurakin, Ian J. Goodfellow, Samy Bengio (2017): Adversarial examples in the physical world]
I. B. Defenses against adversarial examples

Types of attacks:
- Non-targeted attacks
- Targeted attacks

Knowledge of the attacker:
- White-box
- Black-box

Solution 1
- Create a SafetyNet

Solution 2
- Train on correctly labelled adversarial examples

Solution 3
- Adversarial training
  \[ L_{new} = L(W,b,x,y) + \lambda L(W,b,x_{adv},y) \]
- Adversarial logit pairing
  \[ L_{new} = L(W,b,x,y) + \lambda \left\| f(x;W,b) - f(x_{adv};W,b) \right\|^2_2 \]
I. C. Why are neural networks vulnerable to adversarial examples?

See board.

Do neural networks actually understand the data?

[Yuan et al. (2017): Adversarial Examples: Attacks and Defenses for Deep Learning]
II. Generative Adversarial Networks (GANs)

A. Motivation

B. G/D Game

C. Training GANs

D. Nice results

E. Evaluating GANs

[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]
**Motivation**: endowing computers with an understanding of our world.

**Goal**: collect a lot of data, use it to train a model to generate similar data from scratch.

**Intuition**: number of parameters of the model $<<$ amount of data
II.A - Motivation

**Probability distributions:**

Samples from the “real data distribution”

Samples from the “generated distribution”

[Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas (2017): StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks]
II.B - G/D Game

How can we train G to generate images from the true data distributions?

100-d random code

\[
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix}
\]

Generator “G” (Neural Network)

(64,64,3) generated image

≠

[Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas (2017): StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks]
II.B - G/D Game

Run Adam simultaneously on two minibatches (true data / generated data)

Gradients

Binary classification

\[
\begin{cases}
    y = 0 \text{ if } x = G(z) \\
    y = 1 \text{ otherwise}
\end{cases}
\]

Probability distributions

Image space

Real images (database)

(64,64,3) generated image

100-d random code

\[
\begin{pmatrix}
0.47 \\
n \\
0.19
\end{pmatrix}
\]

Generator “G” (Neural Network)

Discriminator “D” (Neural Network)
II.B - G/D Game

End goal: G is outputting images that are indistinguishable from real images for D.

\[
\begin{align*}
    &\text{Generator “G” (Neural Network)} \\
    &\begin{pmatrix} 0.47 \\ \vdots \\ 0.19 \end{pmatrix} \\
    &z \\
    &\text{100-d random code} \\
    &\text{(64,64,3) generated image} \\
    &\text{Discriminator “D” (Neural Network)} \\
    &\begin{cases} 
        y = 0 & \text{if } x = G(z) \\
        y = 1 & \text{otherwise}
    \end{cases}
\end{align*}
\]
II.B - G/D Game

Training procedure, we want to minimize:

- **The cost of the discriminator**

\[
J^{(D)} = - \frac{1}{m_{\text{real}}} \sum_{i=1}^{m_{\text{real}}} y_{\text{real}}^{(i)} \log(D(x^{(i)})) - \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} (1 - y_{\text{gen}}^{(i)}) \log(1 - D(G(z^{(i)})))
\]

- **Cross-entropy 1:**
  "D should correctly label real data as 1"

- **Cross-entropy 2:**
  "D should correctly label generated data as 0"

- **The cost of the generator**

\[
J^{(G)} = -J^{(D)} = \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(1 - D(G(z^{(i)})))
\]

"G should try to fool D: by minimizing the opposite of what D is trying to minimize"

Labels:
\[
\begin{cases}
  y_{\text{real}} & \text{is always 1} \\
  y_{\text{gen}} & \text{is always 0}
\end{cases}
\]
II.C - Training GANs

Saturating cost for the generator:

\[
\min \left[ \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(1 - D(G(z^{(i)}))) \right] \Leftrightarrow \max \left[ \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D(G(z^{(i)}))) \right] \Leftrightarrow \min \left[ -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D(G(z^{(i)}))) \right]
\]

Non-saturating cost

\[
J^{(G)} = - \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D(G(z^{(i)})))
\]

Saturating cost

\[
J^{(G)} = \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(1 - D(G(z^{(i)})))
\]

[Ian Goodfellow (2014): NIPS Tutorial: GANs]
II.C - Training GANs

Note that: \[
\min \left[ \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(1 - D(G(z(i)))) \right] \Leftrightarrow \max \left[ \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D(G(z(i)))) \right] \Leftrightarrow \min \left[ -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D(G(z(i)))) \right]
\]

New training procedure, we want to minimize:

\[
J^{(D)} = -\frac{1}{m_{\text{real}}} \sum_{i=1}^{m_{\text{real}}} y_{\text{real}}^{(i)} \log(D(x^{(i)})) - \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} (1 - y_{\text{gen}}^{(i)}) \log(1 - D(G(z^{(i)})))
\]

Cross-entropy 1: “D should correctly label real data as 1”

Cross-entropy 2: “D should correctly label generated data as 0”

\[
J^{(G)} = -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D(G(z^{(i)})))
\]

“G should try to fool D: by minimizing this”
Table 1: Generator and discriminator loss functions. The main difference whether the discriminator outputs a probability (MM GAN, NS GAN, DRAGAN) or its output is unbounded (WGAN, WGAN GP, LS GAN, BEGAN), whether the gradient penalty is present (WGAN GP, DRAGAN) and where is it evaluated. We chose those models based on their popularity.

<table>
<thead>
<tr>
<th>GAN</th>
<th>Discriminator Loss</th>
<th>Generator Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM GAN</td>
<td>$L_{D}^{GAN} = -E_{x \sim p_d}[\log(D(x))] - E_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$</td>
<td>$L_{G}^{GAN} = E_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$</td>
</tr>
<tr>
<td>NS GAN</td>
<td>$L_{D}^{NSGAN} = -E_{x \sim p_d}[\log(D(x))] - E_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$</td>
<td>$L_{G}^{NSGAN} = -E_{\hat{x} \sim p_g}[\log(D(\hat{x}))]$</td>
</tr>
<tr>
<td>WGAN</td>
<td>$L_{D}^{WGAN} = -E_{x \sim p_d}[D(x)] + E_{\hat{x} \sim p_g}[D(\hat{x})]$</td>
<td>$L_{G}^{WGAN} = -E_{\hat{x} \sim p_g}[D(\hat{x})]$</td>
</tr>
<tr>
<td>WGAN GP</td>
<td>$L_{D}^{WANGP} = L_{D}^{WGAN} + \lambda E_{\hat{x} \sim p_g}[(</td>
<td></td>
</tr>
<tr>
<td>LS GAN</td>
<td>$L_{D}^{LSGAN} = -E_{x \sim p_d}[(D(x) - 1)^2] + E_{\hat{x} \sim p_g}[D(\hat{x})^2]$</td>
<td>$L_{G}^{LSGAN} = -E_{\hat{x} \sim p_g}[(D(\hat{x} - 1)^2]$</td>
</tr>
<tr>
<td>DRAGAN</td>
<td>$L_{D}^{DRAGAN} = L_{D}^{GAN} + \lambda E_{\hat{x} \sim p_d+N(0,\sigma)}[(</td>
<td></td>
</tr>
<tr>
<td>BEGAN</td>
<td>$L_{D}^{BEGAN} = E_{x \sim p_d}[</td>
<td></td>
</tr>
</tbody>
</table>

[Lucic, Kurach et al. (2018): Are GANs Created Equal? A Large-Scale Study]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri
Simultaneously training G/D?

\[ J^{(G)} = -\frac{1}{m_g} \sum_{i=1}^{m_g} \log(D(G(z^{(i)}))) \]

Non-saturating cost

\[ J^{(G)} = \frac{1}{m_g} \sum_{i=1}^{m_g} \log(1 - D(G(z^{(i)}))) \]

Saturating cost

II.C - Training GANs

for num_iterations:
for k iterations:
update D
update G

[Ian Goodfellow (2014): NIPS Tutorial: GANs]
II.C - Training GANs

BatchNorm with GANs:

Generated images (batch 1)

Generated images (batch 2)

[Ian Goodfellow (2014): NIPS Tutorial: GANs]
BatchNorm with GANs:

Assume no batchnorm

\[
\begin{align*}
\text{Generator } "G" \quad &\text{(Neural Network)} \\
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix} &\rightarrow \\
\text{generated image} &\rightarrow \\
1 &\rightarrow \\
\text{Code 1}
\end{align*}
\]

\[
\begin{align*}
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix} &\rightarrow \\
\text{Generator } "G" \quad &\text{(Neural Network)} \\
\text{generated image} &\rightarrow \\
2 &\rightarrow \\
\text{Code 2}
\end{align*}
\]

Assume batchnorm

\[
\begin{align*}
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix} &\rightarrow \\
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix} &\rightarrow \\
\text{Generator } "G" \quad &\text{(Neural Network)} \\
\text{generated images} &\rightarrow \\
1 &\rightarrow \\
2 &\rightarrow \\
\text{Code 1} &\text{Code 2}
\end{align*}
\]
### II.C - Training GANs

**BatchNorm with GANs:**

**BatchNorm**

\[
Z = \{z^{(1)}, ..., z^{(m)}\}
\]

\[
\mu_B = \frac{1}{m} \sum_{i=1}^{m} z^{(i)}
\]

\[
\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (z^{(i)} - \mu_B)^2
\]

\[
z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}
\]

\[
z^{(i)} = \gamma z^{(i)}_{\text{norm}} + \beta
\]

**Reference BatchNorm**

\[
R = \{r^{(1)}, ..., r^{(m)}\}
\]

\[
Z = \{z^{(1)}, ..., z^{(m)}\}
\]

\[
\mu_B = \frac{1}{m} \sum_{i=1}^{m} r^{(i)}
\]

\[
\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (r^{(i)} - \mu_B)^2
\]

\[
r^{(i)}_{\text{norm}} = \frac{r^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}
\]

\[
r^{(i)} = \gamma r^{(i)}_{\text{norm}} + \beta
\]

**Virtual BatchNorm**

\[
R = \{r^{(1)}, ..., r^{(m)}\}
\]

\[
Z = \{z^{(1)}, ..., z^{(m)}\}
\]

For \(k = 1, ..., m\)

\[
\mu_B = \frac{1}{m+1} \left( z^{(k)} + \sum_{i=1}^{m} r^{(i)} \right)
\]

\[
\sigma_B^2 = \frac{1}{m+1} \left( (z^{(k)} - \mu_B)^2 + \sum_{i=1}^{m} (r^{(i)} - \mu_B)^2 \right)
\]

\[
z^{(k)}_{\text{norm}} = \frac{z^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}
\]

\[
z^{(k)} = \gamma z^{(k)}_{\text{norm}} + \beta
\]
Recap: GANs’ training tips

- Modification of the cost function
- Keep D up-to-date with respect to G (k update for D / 1 update for G)
- Use Virtual Batchnorm
- (not presented but important) One-sided label smoothing

II.C - Training GANs

And a lot more, GANs are hard to train!

[Soumith et al. (2016): GanHacks]
[Lucic, Kurach et al. (2018): Are GANs Created Equal? A Large-Scale Study]
II.D - Nice results

**Operation on codes**

Code 1

\[
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix}
\]

Generator “G” (Neural Network)

(generated image)

Code 2

\[
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix}
\]

Generator “G” (Neural Network)

(generated image)

Code 3

\[
\begin{pmatrix}
0.42 \\
\vdots \\
0.07
\end{pmatrix}
\]

Generator “G” (Neural Network)

(generated image)

\[
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix} - \begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix} + \begin{pmatrix}
0.42 \\
\vdots \\
0.07
\end{pmatrix}
\]

Generator “G” (Neural Network)

(generated image)

Man with glasses - man + woman = woman with glasses
II.D - Nice results

Image Generation:

Samples from the “generated distribution”

[Zhang et al. (2017): StackGAN++]
II.D - Nice results

Pix2Pix:

https://affinelayer.com/pixsrv/ by Christopher Hesse.
II.D - Nice results

Super-resolution image:

II.D - Nice results

CycleGANs:

[Zhu, Park et al. (2017): Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]
II.D - Nice results

**Goal:** Convert horses to zebras on images, and vice-versa.

<table>
<thead>
<tr>
<th>Data?</th>
<th>Architecture?</th>
<th>Cost?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpaired images</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Horse images</td>
<td>Zebra images</td>
<td></td>
</tr>
</tbody>
</table>

[Zhu, Park et al. (2017): Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]
Architecture?

\[
\begin{align*}
\text{Generator1 (H2Z)} & \quad \text{Discriminator1} & \\
G1(H) & \quad & \\
\text{H} & \quad \text{G2(G1(H))} & \\
\text{Generator2 (Z2H)} & \quad & \\
G2(Z) & \quad \text{Discriminator2} & \\
\text{Z} & \quad \text{G1(G2(Z))} & \\
\end{align*}
\]
Loss to minimize?

\[
J^{(D1)} = -\frac{1}{m_{\text{real}}} \sum_{i=1}^{m_{\text{real}}} \log(D1(h^{(i)})) - \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(1 - D1(G1(H^{(i)})))
\]

\[
J^{(G1)} = -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D1(G1(H^{(i)})))
\]

\[
J^{(D2)} = -\frac{1}{m_{\text{real}}} \sum_{i=1}^{m_{\text{real}}} \log(D2(z^{(i)})) - \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(1 - D2(G2(Z^{(i)})))
\]

\[
J^{(G2)} = -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log(D2(G2(Z^{(i)})))
\]

\[
J_{\text{cycle}} = -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \| G2(G1(H^{(i)}) - H^{(i)} \|_1 - \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \| G1(G2(Z^{(i)}) - Z^{(i)} \|_1
\]

\[
J = J^{(D1)} + J^{(G1)} + J^{(D2)} + J^{(G2)} + \lambda J_{\text{cycle}}
\]
II.D - Nice results

**CycleGANs:**

[Zhu, Park et al. (2017): Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]

Face2ramen
Other applications:

• Beaulieu-Jones et al., Privacy-preserving generative deep neural networks support clinical data sharing.

• Hwang et al., Learning Beyond Human Expertise with Generative Models for Dental Restorations.

• Gomez et al., Unsupervised cipher cracking using discrete GANs.

• Many more…
II.E - Evaluating GANs

How to evaluate GANs?

Human annotators

<table>
<thead>
<tr>
<th>web-application</th>
<th>Indicate if image is fake or real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image 1</td>
<td>Image 2</td>
</tr>
<tr>
<td>Image 3</td>
<td>Image 4</td>
</tr>
<tr>
<td>Image 5</td>
<td>Image 6</td>
</tr>
</tbody>
</table>

Inception Score (IS)

\[
IS(G) = \exp\left( E_{x \sim p_g} \left[ KL(p(y | x) \| p(y)) \right] \right)
\]

Measure of image quality

\[
IS(G) = \exp\left( \frac{1}{m_b} \sum_{i=1}^{m_b} \sum_{j=1}^{n_y} \hat{y}^{(i)}_j \left( \log \hat{y}^{(i)}_j - \log\left( \frac{1}{m_b} \sum_{k=1}^{m_b} \hat{y}^{(k)}_j \right) \right) \right)
\]

Measure of image diversity

Also: Fréchet Inception Distance (FID)

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri
For Wednesday 10/24, 11am:

C2M3
- Quiz: Hyperparameter tuning, Batch Normalization, Programming Frameworks
- Programming assignment: Tensorflow

C3M1 and C3M2
- Quiz: Bird recognition in the city of Peacetopia (case study)
- Quiz: Autonomous driving (case study)

This Friday 10/19:
- Hands-on session this Friday

Check out the project example code!
(cs230-stanford.github.io)

Meet with your mentor TA, to go over project proposal feedback (attendance required.)