CS230: Lecture 5
Atacking Networks with Adversarial Examples
-
Generative Adversarial Networks

Kian Katanforoosh
Today’s outline

I. Attacking NNs with Adversarial Examples
II. Generative Adversarial Networks
I. Adversarial examples

*Discovery (2014)*: several machine learning models, including state-of-the-art neural networks, are vulnerable to adversarial examples

A. How to build adversarial examples and attack a network?

B. Examples

C. How to defend against adversarial examples?
I. A. How to build adversarial examples and attack a network?

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

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

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

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

---

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

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

Space of possible input images

Space of images classified as iguanas

Space of real images
### I. A. How to build adversarial examples and attack a network?

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

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

   And: $x = x_{\text{cat}}$

2. **Defining the loss function**
   
   $$L(\hat{y}, y) = \frac{1}{2} \left\| \hat{y}(W, b, x) - y_{\text{iguana}} \right\|_2^2$$

3. **Optimize the image**

   After many iterations
   
   $$x = x_{\text{cat}} + \lambda \| x - x_{\text{cat}} \|_2^2$$

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

---

[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri
I. A. How to build adversarial examples and attack a network?

92% Cat

94% Iguana
Adversarial Examples In The Physical World

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

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

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

Solution 1
• Train on correctly labelled adversarial examples

Solution 2
• Adversarial training
  \[ L_{\text{new}} = L(W, b, x, y) + \lambda L(W, b, x_{\text{adv}}, y) \]
• Adversarial logit pairing
  \[ L_{\text{new}} = L(W, b, x, y) + \lambda \| f(x; W, b) - f(x_{\text{adv}}; W, b) \|_2^2 \]

Do neural networks actually understand the data?

[Harini Kannan et al. (2018): Adversarial Logit Pairing]
II. Generative Adversarial Networks (GANs)

A. Motivation

B. G/D Game

C. Practical tips to train/evaluate GANs

D. Interesting results

[Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015): Explaining and harnessing adversarial examples]
II.A - Motivation

**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 $\ll$ amount of data

[The Gan Zoo (2017)]
II.A - Motivation

Probability distributions:

Samples from the “true data distribution”

[Images of various objects]

Samples from the “generated distribution”

[Images of generated images]

“true data distribution”

“generated distribution”

Goal

Matching distributions

[Andrej Karpathy et al. (2016): Generative Models, OpenAI blog]
II.B - G/D Game

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

[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

Generator “G”
(Neural Network)

(64,64,3)
generated image

Discriminator “D”
(Neural Network)

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

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

Binary classification

Gradients

Probability distributions

Image space

Real images
(database)

100-d random code
II.B - G/D Game

Discriminator “D” (Neural Network)

Binary classification

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

Generator “G” (Neural Network)

100-d random code

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

Real images (database)

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

(64,64,3) generated image

Gradients

Probability distribution

Image space

Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri
II.B - G/D Game

Training procedure, we want to minimize:

- The loss of the discriminator

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

  cross-entropy 1: “D should correctly label real data as 1”
  cross-entropy 2: “D should correctly label generated data as 0”

- The loss 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:
\[
\begin{align*}
  y_{\text{real}} & \text{ is always 1} \\
  y_{\text{gen}} & \text{ is always 0}
\end{align*}
\]
II.C - Training GANs

Saturating cost for the generator:

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

Non-saturating cost

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

Saturating cost

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

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

\text{cross-entropy 1:} \quad \text{"D should correctly label real data as 1"}

\text{cross-entropy 2:} \quad \text{"D should correctly label generated data as 0"}

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

\text{"G should try to fool D: by minimizing this"}

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
II.C - Training GANs

BatchNorm with GANs:

Generated images (batch 1)

Generated images (batch 2)
II.C - Training GANs

**BatchNorm with GANs:**

**Assume no batchnorm**

Code 1
\[
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix}
\]

Generator “G” (Neural Network)

(64,64,3) generated image

1

Code 2
\[
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix}
\]

Generator “G” (Neural Network)

(64,64,3) generated image

2

**Assume batchnorm**

Code 1
\[
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix}
\]

Generator “G” (Neural Network)

Code 2
\[
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix}
\]

(64,64,3) generated images

1

2
II.C - Training GANs

BatchNorm with GANs:

<table>
<thead>
<tr>
<th>BatchNorm</th>
<th>Reference BatchNorm</th>
<th>Virtual BatchNorm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z = {z^{(1)}, \ldots, z^{(m)}}$</td>
<td>$R = {r^{(1)}, \ldots, r^{(m)}}$</td>
<td>$R = {r^{(1)}, \ldots, r^{(m)}}$</td>
</tr>
<tr>
<td>$\mu_B = \frac{1}{m} \sum_{i=1}^{m} z^{(i)}$</td>
<td>$Z = {z^{(1)}, \ldots, z^{(m)}}$</td>
<td>$Z = {z^{(1)}, \ldots, z^{(m)}}$</td>
</tr>
<tr>
<td>$\sigma^2_B = \frac{1}{m} \sum_{i=1}^{m} (z^{(i)} - \mu_B)^2$</td>
<td>$\mu_B = \frac{1}{m} \sum_{i=1}^{m} r^{(i)}$</td>
<td>$\mu_B = \frac{1}{m+1} \left( z^{(k)} + \sum_{i=1}^{m} r^{(i)} \right)$</td>
</tr>
<tr>
<td>$z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu_B}{\sqrt{\sigma^2_B + \epsilon}}$</td>
<td>$\sigma^2_B = \frac{1}{m} \sum_{i=1}^{m} (r^{(i)} - \mu_B)^2$</td>
<td>$\sigma^2_B = \frac{1}{m+1} \left( (z^{(k)} - \mu_B)^2 + \sum_{i=1}^{m} (r^{(i)} - \mu_B)^2 \right)$</td>
</tr>
<tr>
<td>$z^{(i)} = \gamma z^{(i)}_{\text{norm}} + \beta$</td>
<td>$z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu_B}{\sqrt{\sigma^2_B + \epsilon}}$</td>
<td>$z^{(k)}_{\text{norm}} = \frac{z^{(k)} - \mu_B}{\sqrt{\sigma^2_B + \epsilon}}$</td>
</tr>
<tr>
<td>$z^{(k)} = \gamma z^{(k)}_{\text{norm}} + \beta$</td>
<td>$z^{(k)} = \gamma z^{(k)}_{\text{norm}} + \beta$</td>
<td>$z^{(k)} = \gamma z^{(k)}_{\text{norm}} + \beta$</td>
</tr>
</tbody>
</table>
Recap: GANs’ training tips

- Use the non-saturated 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

And a lot more, GANs are hard to train!

[Soumith et al. (2016): GanHacks]
II.D - Interesting results

Operation on codes

Code 1
\[
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix}
\]
Generator “G” (Neural Network)
\[
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix}
\]
generated image

Code 2
\[
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix}
\]
Generator “G” (Neural Network)
\[
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix}
\]
generated image

Code 3
\[
\begin{pmatrix}
0.42 \\
\vdots \\
0.07
\end{pmatrix}
\]
Generator “G” (Neural Network)
\[
\begin{pmatrix}
0.42 \\
\vdots \\
0.07
\end{pmatrix}
\]
generated image

Code 1
\[
\begin{pmatrix}
0.12 \\
\vdots \\
0.92
\end{pmatrix}
\]
- Code 2
\[
\begin{pmatrix}
0.47 \\
\vdots \\
0.19
\end{pmatrix}
\]
+ Code 3
\[
\begin{pmatrix}
0.42 \\
\vdots \\
0.07
\end{pmatrix}
\]
= Woman with glasses

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

Image Generation:

Samples from the "generated distribution"

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

Pix2Pix:

https://affinelayer.com/pixsrv/
II.D - Interesting results

Super-resolution image:


Kian Katanforoosh, Andrew Ng, Younes Bensouda Mourri
CycleGANs:

II.D - Interesting results

[Jun-Yan Zhu et al. (2017): Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]
II.D - Interesting results

CycleGANs:

https://hardikbansal.github.io/CycleGANBlog/

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

For Tuesday 05/01, 9am:

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)

For Friday 02/16, 9am:

• Hands-on session this Friday

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

Meet with your mentor (TA), you’ll receive a Calendly invite.