Hack Object Detector is just Like Training Neural Networks

Jay Xiong, Wang Yang, Liu Yan, Hao Xin, Wei Tao

Presented at HITB GSEC 2019
About me

· Jay Xiong,

· Security Researcher

· League of Legends/Dead by Daylight Player
Agenda

- Background on object detector deception (ODD)
- Object Detector Deception Analysis
- Use Trick to Enhance ODD
- One More Thing
Some concepts

- What is object detector?
- What is adversarial example?
- What is object detector deception?
What is object detector?

Faster YOLO v2 model architecture

Cited from github user gliese581gg

https://github.com/gliese581gg/YOLO_tensorflow/blob/master/YOLO_tiny_tf.py
What is object detector?

Terminator 1, 1984,
https://www.youtube.com/watch?v=9MeaaCwBW28
What is adversarial example?

Ian J. Goodfellow, etc, EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES, ICLR 2015,
https://arxiv.org/abs/1412.6572
Object detector deception (ODD)

Object detector combined with adversarial example.

Figure 4: Output of the extended RP₂ algorithm to attack YOLO v2 using poster and sticker attacks.

Kevin Eykholt, etc, Physical Adversarial Examples for Object Detectors, USENIX WOOT 2018
Adversarial Threat to DNN

- A real threat in deep learning application scenario

Physical attack against YOLO-V3 Baidu X-lab demonstrated on Blackhat Euro 2018
Related work on object detector deception (ODD)

CVPR Workshop 2019, [*], obtained from attacking test batch distribution
https://www.youtube.com/watch?v=MIbFvK2S9g8

Generated from attacking prediction score of a single image
https://github.com/advboxes/AdvBox

[*] Simen Thys, Wiebe Van Ranst, Toon Goedemé, Fooling automated surveillance cameras: adversarial patches to attack person detection. CVPR Workshop 2019,
Get down to what’s happened

· The neural network training process.

The neural network training process is to search for a set of weights, which minimize the difference between model prediction and training/test label distribution. It can be summarized as follows:

Training Loss is $J(y_t = h_\theta(x), y_s), y_s \sim D_s$, find $\theta, y = h(\theta, x)$, st. $$(x, y) \sim D_s$$

· The ODD process

The ODD process is an "odd" process compared to how we usually a neural network, instead of trying to find a set of weights to make model prediction more precise, the ODD is a reversed process.

With model weights remain unchanged, we try to find a robust input $\delta$ to drift model prediction far from label distribution. It can be summarized as follows:

Attack loss is $J(y_t = h_\theta(x'), y), find a or a set of $\delta', y' = h_\theta(x + \delta'), st. (x', y') \sim D_{target}$ we would like $\delta$ to be as small as possible.
Object Detector Deception Modeling

Let’s say:
convolution layer: \( \text{conv}_l(x_{l-1}) = w_{l-1}x_{l-1} + b_{l-1} \)
leaky relu activation: \( h_l(x_{l-1}) = \text{relu}_\text{leaky}(x_{l-1}) \)
maxpooling layer: \( m_l(x_{l-1}) = \text{maxpool}(x_{l-1}) \)
we can obtain their gradients by:

\[
\frac{\partial \text{conv}_l(x_{l-1})}{\partial x_{l-1}} = w_{l-1}^T \\
\frac{\partial h_l(x_{l-1})}{\partial x_{l-1}} = \text{np.where}(x_{l-1} > 0, 1, 0.1x_{l-1}) = h_l \\
\frac{\partial m_l(x_{l-1})}{\partial x_{l-1}} = \text{np.where}(x_{l-1} > 0, 1, 0) = m_l
\]

let’s name \( y_t \) is the score to bend, we have the inference from the last conv layer.
\( y_t = w_l x_l + b_l \)

\[
d y_t = d \text{conv}_{l+1}(x_l) = d(w_l x_l + b_l) \\
= w_l dx_l = w_l dm_l(x_{l-1}) = w_l m_l \odot dx_{l-1} \\
= w_l m_l \odot d(h_{l-1}(x_{l-2})) = w_l m_l \odot h_{l-1} \odot dx_{l-2} \\
= w_l m_l \odot h_{l-1} \odot d \text{conv}_{l-2}(x_{l-3}) \\
\vdots \\
= w_l m_l \odot h_{l-1} \odot w_{l-3} m_{l-3} \odot h_{l-4} \odot w_{l-6} \cdots dx_{\text{img}} \\
= r_l r_{l-1} r_{l-2} \cdots r_1 dx_{\text{img}} \\
= \text{tr}(r_l r_{l-1} r_{l-2} \cdots r_1 dx_{\text{img}})
\]

\[ \Rightarrow \Delta y = \text{tr}(r_l r_{l-1} \cdots r_l \Delta x_{\text{img}}) \]
\[ \Rightarrow \frac{\partial y_t}{\partial x_{\text{img}}} = (r_l r_{l-1} \cdots r_1)^T \]

similarly:
\[ \frac{\partial x_{\text{img}}}{\partial y_t} = (R_1 R_2 \cdots R_l)^T, R_i = r_i^{-1} \]
Object Detector Deception Modeling

\[ \Delta y \]

\[ \Delta x_{img} \]

\[
\frac{\partial y_t}{\partial x_{img}} = (r_l r_{l-1} \ldots r_1)^T
\]

\[
\frac{\partial x_{img}}{\partial y_t} = (R_1 R_2 \ldots R_l)^T
\]
Object Detector Deception Modeling

Adversarial Sticker = Perturbation = \( x' = x + \delta' \)

- Expectation over Testing Batch (eotb)
  \[ \Delta y = tr(n_i n_{i-1} \ldots n_1 \Delta x_{img}) \]

- Expectation of a single target case
  \[ \Delta y = tr(n_i n_{i-1} \ldots n_1 \Delta x_{img}) \]
Why not use tricks in neural networks training for ODD?

Use proper weights initialization for better regression and local optimal?

Use proper pixel value initialization for better physical deception robustness over environmental interference?
The trick: Adversarial Initialization

Why does it work:

when training: \( \uparrow \Delta y = tr(\eta_{l-1} \ldots r_l \Delta \bar{x}_{img}) \Rightarrow \uparrow \Delta x_{img} \)

when attacking: \( \uparrow \Delta x_{img} \Rightarrow \uparrow \Delta y = tr(\eta_{l} \eta_{l-1} \ldots r_l \Delta \bar{x}_{img}) \)

note: \( \Delta x_{img} \) (perturbation) is the accumulation of model gradients
Use Adversarial Initialization to Enhance ODD

Experiment Setting:

```python
def build_graph(self, inputs, mode):
    assert mode == 'init_model' or mode == 'reuse model'

    self.conv1 = self.conv_layer(inputs, 16, (3, 3), 'Variable:0', mode=mode)
    self.conv2 = self.conv_layer(self.conv1, 32, (3, 3), 'Variable:0', mode=mode)
    self.pool2 = self.pooling_layer(self.conv2, 2, 'Variable:0', mode=mode)
    self.conv3 = self.conv_layer(self.pool2, 64, (3, 3), 'Variable:0', mode=mode)
    self.conv4 = self.conv_layer(self.conv3, 128, (3, 3), 'Variable:0', mode=mode)
    self.pool4 = self.pooling_layer(self.conv4, 2, 'Variable:0', mode=mode)
    self.conv5 = self.conv_layer(self.pool4, 256, (3, 3), 'Variable:0', mode=mode)
    self.conv6 = self.conv_layer(self.conv5, 512, (3, 3), 'Variable:0', mode=mode)
    self.conv7 = self.conv_layer(self.conv6, 1024, (3, 3), 'Variable:0', mode=mode)
    self.conv8 = self.conv_layer(self.conv7, 1024, (3, 3), 'Variable:0', mode=mode)
    self.pool8 = self.pooling_layer(self.conv8, 2, 'Variable:0', mode=mode)
    self.conv9 = self.conv_layer(self.pool8, 1024, (3, 3), 'Variable:0', mode=mode)
    self.conv10 = self.conv_layer(self.conv9, 1024, (3, 3), 'Variable:0', mode=mode)
    self.conv11 = self.conv_layer(self.conv10, 1024, (3, 3), 'Variable:0', mode=mode)
    self.conv12 = self.conv_layer(self.conv11, 1024, (3, 3), 'Variable:0', mode=mode)
    self.conv13 = self.conv_layer(self.conv12, 1024, (3, 3), 'Variable:0', mode=mode)
    self.conv14 = self.conv_layer(self.conv13, 1024, (3, 3), 'Variable:0', mode=mode)
    self.conv15 = self.conv_layer(self.conv14, 1024, (3, 3), 'Variable:0', mode=mode)
    self.fc16 = self.fc_layer(self.conv15, 10, 'Variable:0', mode=mode)
    self.fc17 = self.fc_layer(self.fc16, 10, 'Variable:0', mode=mode)
    self.fc18 = self.fc_layer(self.fc17, 10, 'Variable:0', mode=mode)
    self.fc19 = self.fc_layer(self.fc18, 10, 'Variable:0', mode=mode)
    self.fc20 = self.fc_layer(self.fc19, 10, 'Variable:0', mode=mode)

    return self.CTarge
```

Cited from github user gliese581gg

https://github.com/gliese581gg/YOLO_tensorflow/blob/master/YOLO_tiny_tf.py
Use Adversarial Initialization to Enhance ODD

Experiment Setting:

· End to end differentiable object detector faster yolo v2
· 1200+ images containing people
· Adam optimizer lr = 1e-2
· Loss function $\text{Loss} = C_{people} + \lambda_1 \|\delta\|_2^2 + \lambda_2 \|d\delta\|_2$

```python
# original
init_inter = tf.constant_initializer(0.001*np.random.random([1,448,448,3]))

# improved
# think of it, we want ad sticker starts at somewhere easy
init_inter = tf.constant_initializer(0.7*np.random.normal(scale=0.8,size=[1,448,448,3]))
```
Compare Original & Adversarial Initialization

The physical adversarial robustness is not free.

- the cost of adversarial initialization: less optimal minimum
- at every iteration, larger $y$ than original
- result in more color-saturated adversarial sticker
Compare Original & Adversarial Initialization

Time-lapse photography:

Original

Improved
Compare Original & Adversarial Initialization

Effectiveness in physical world:

The experiment:

![Diagram showing input image, fast yolo v2, and output confidence]

- input image
- faster yolo v2
- output confidence

showing or blocking the sticker
Compare Original & Adversarial Initialization

Effectiveness in physical world:
One More thing

How to craft your own adversarial sticker on your own object detector?

https://github.com/advboxes/AdvBox
THANKS