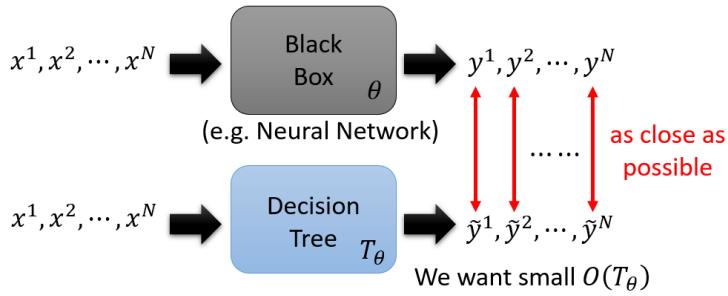


# Decision Tree

$O(T_\theta)$  : how complex  $T_\theta$  is

e.g. average depth of  $T_\theta$

- Using an interpretable model to mimic the behavior of an uninterpretable model.



**Problem:** We don't want the tree to be too large.

Decision Tree足够复杂时，理论上完全可以模仿黑盒子的结果，但是这样Decision Tree会非常非常复杂，那么Decision Tree本身又变得没法解释了，因此我们不想Decision Tree太过复杂。这里我们用 $O(T_\theta)$ 表示决策树的复杂度，定义方式可以自选，比如树的平均深度。

要考虑决策树的复杂度，因此要在损失函数中加一个正则项： $\theta = \operatorname{argmin}_\theta L(\theta) + \lambda O(T_\theta)$

## Decision Tree – Tree regularization

<https://arxiv.org/pdf/1711.06178.pdf>

- Train a network that is easy to be interpreted by decision tree.  
 $T_\theta$  : tree mimicking network with parameters  $\theta$   
 $O(T_\theta)$  : how complex  $T_\theta$  is
- $$\theta^* = \arg \min_{\theta} L(\theta) + \lambda O(T_\theta)$$
- Original loss function for training network      Preference for network parameters
- Tree Regularization

Is the objective function with tree regularization differentiable? No! Check the reference for solution.

但是这个正则项没法做偏导，所以没有办法做GD。

解决方法：<https://arxiv.org/pdf/1711.06178.pdf>

train一个神奇的神经网络，我们给它一个神经网络的参数，它就可以输出一个数值，来表示输入的参数转成Decision Tree后Decision Tree的平均深度。

中心思想是用一个随机初始化的结构简单的Neural Network，找一些NN转成DT，算出平均深度，就有了训练data，训练后可以模拟出决策树的平均深度，然后用Neural Network替换上面的正则项，Neural Network是可以微分的，然后就可以Gradient Descent了。

# Adversarial Attack

## Adversarial Attack

### Motivation

We seek to deploy machine learning classifiers not only in the labs, but also in real world.

The classifiers that are robust to noises and work "most of the time" is not sufficient. We want the classifiers that are robust the inputs that are built to fool the classifier.

Especially useful for spam classification, malware detection, network intrusion detection, etc.

光是强还不够，还要应对人类的恶意攻击。

攻击是比较容易的，多数machine learning的model其实相当容易被各式各样的方法攻破，防御目前仍然是比较困难的。

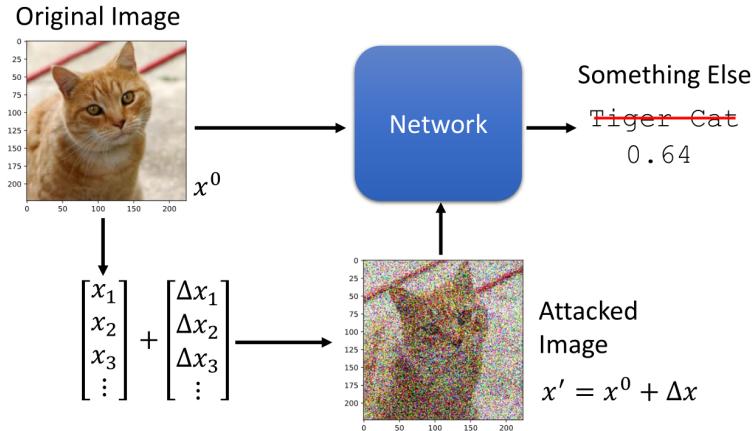
## Attack

### What do we want to do?

给当前的machine input一个图片，输出Tiger Cat的信息分为0.64，你不会说它是一个很差的model，你会说它是一个做的还不错的model。

我们现在想要做的是：往图片上面加上一些杂讯，这些杂讯不是从gaussian distribution中随机生成的（随机生成的杂讯，没有办法真的对Network造成太大的伤害），这是种很特殊的讯号，把这些很特殊的讯号加到一张图片上以后，得到稍微有点不一样的图片。将这张稍微有点不一样的图片丢到Network中，Network会得到非常不一样的结果。

本来的图片叫做 $x^0$ ，而现在是在原来的 $x^0$ 上加上一个很特别的 $\Delta x$ ，得到一张新的图片 $x'$  ( $x' = x^0 + \Delta x$ )。然后将 $x'$ 丢到Network中，原来看到 $x^0$ 时，Network会输出是一张Tiger Cat，但是这个Network看到时输出一个截然不同的答案 (Attacked image, Else)，那么这就是所谓攻击所要做的事情。



### Loss Function for Attack

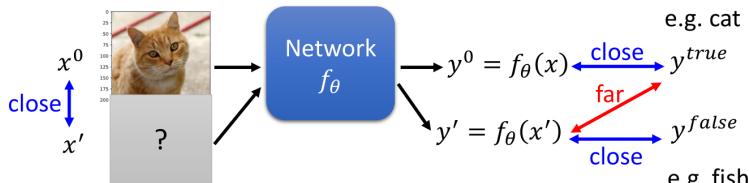
即找一张图片，使得loss(cross-entropy loss)越大越好，此时网络的参数训练完了，不能改变，而是只改变输入，使我们找到这样一张图片，能够让结果“越错越好”，离正确答案越远越好。

普通的训练模型， $x^0$ 不变，希望找到一个 $\theta$ ，使cross-entropy越小越好

Non-targeted Attack， $\theta$ 不变，希望找到一个 $x'$ ，使得cross-entropy越大越好，cross-entropy越大Loss越小

Targeted Attack，希望找到一个 $x'$ ，使得cross-entropy越大越好，同时与目标（错误）标签越近越好

输入不能改变太大， $x^0$ 和 $x'$ 越接近越好



- **Training:**  $L_{train}(\theta) = C(y^0, y^{true})$   $x$  fixed
- **Non-targeted Attack:**  $L(x') = -C(y', y^{true})$   $\theta$  fixed
- **Targeted Attack:**  

$$L(x') = -C(y', y^{true}) + C(y', y^{false})$$
- **Constraint:**  $d(x^0, x') \leq \varepsilon$  不要被發現

### Constraint

$distance(x^0, x')$ 由任务的不同而不同，主要取决于对人来说什么样的文字、语音、图像信号是相似的。

L-infinity稍微更适合用于影像的攻击上面

Constraint     $d(x^0, x') \leq \varepsilon$

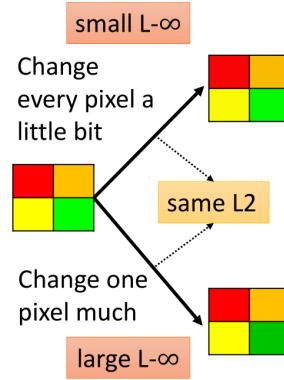
$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ \vdots \\ x' \end{bmatrix} - \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x^0 \end{bmatrix} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \vdots \\ \Delta x \end{bmatrix}$$

- L2-norm

$$\begin{aligned} d(x^0, x') &= \|x^0 - x'\|_2 \\ &= (\Delta x_1)^2 + (\Delta x_2)^2 + (\Delta x_3)^2 \dots \end{aligned}$$

- L-infinity

$$\begin{aligned} d(x^0, x') &= \|x^0 - x'\|_\infty \\ &= \max\{\Delta x_1, \Delta x_2, \Delta x_3, \dots\} \end{aligned}$$



## How to Attack

损失函数的梯度为 (此时变量为  $x$ )

$$\nabla L(x) = \begin{bmatrix} \frac{\partial L(x)}{\partial x_1} \\ \frac{\partial L(x)}{\partial x_2} \\ \frac{\partial L(x)}{\partial x_3} \\ \vdots \end{bmatrix}_{gradient}$$

Gradient Descent, 但是加上限制, 判断  $x^t$  是否符合限制, 如果不符合, 要修正  $x^t$

修正方法: 穷举  $x^0$  附近的所有  $x$ , 用距离  $x^t$  最近的点来取代  $x^t$

## How to Attack

Just like training a neural network,  
but network parameter  $\theta$  is  
replaced with input  $x'$

$$x^* = \arg \min_{d(x^0, x') \leq \varepsilon} L(x')$$

- Gradient Descent (Modified Version)

```
Start from original image  $x^0$ 
For t = 1 to T
     $x^t \leftarrow x^{t-1} - \eta \nabla L(x^{t-1})$ 
    If  $d(x^0, x^t) > \varepsilon$ 
         $x^t \leftarrow fix(x^t)$ 
```

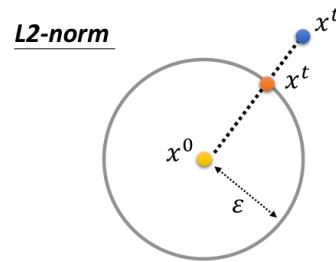
```
def fix( $x^t$ )
    For all  $x$  fulfill
         $d(x^0, x) \leq \varepsilon$ 
    Return the one
    closest to  $x^t$ 
```

# How to Attack

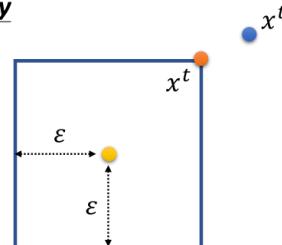
```
def fix(xt)
```

For all  $x$  fulfill  
 $d(x^0, x) \leq \varepsilon$

Return the one  
 closest to  $x^t$



L-infinity



## Example

使用一个猫，可以让它输出海星

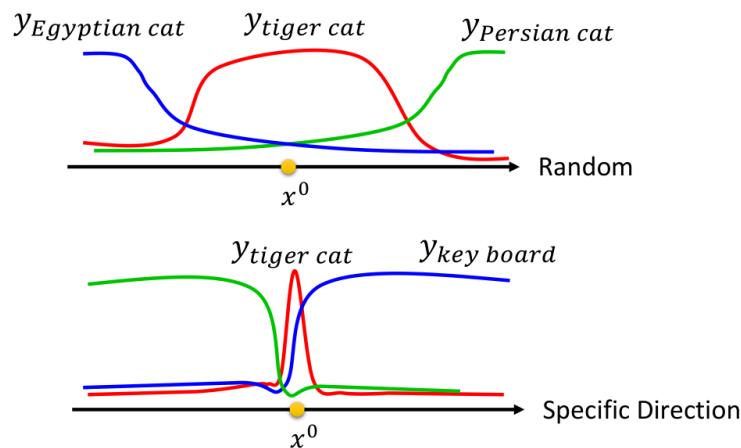
两张图片的不同很小，要相减后乘以50倍才能看出

是否是因为网络很弱呢？随机加噪声，但是对结果的影响不大；加的噪声逐渐增大以后会有影响

## What happened?

高维空间中，随机方向与特定方向的y表现不同，因此一般的杂讯没有效果，特定的杂讯会有效果。

你可以想象  $x^0$  是在非常高维空间中的一个点，你把这个点随机的移动，你会发现多数情况下在这个点的附近很大一个范围内都是能正确辨识的空间，当你把这个点推到的很远的时候才会让机器识别成类似的事物，当你把它推的非常远时才会被辨识成不相关的事物。但是，有一些神奇维度，在这些神奇的维度上  $x^0$  的正确区域只有非常狭窄的一点点，我们只要将  $x^0$  推离原值一点点，就会让模型输出产生很大的变化。



这种现象不止存在于Deep Model中，一般的Model也会被攻击。

## Attack Approaches

- [FGSM](#)
- [Basic iterative method](#)
- [L-BFGS](#)
- [Deepfool](#)
- [JSMA](#)
- [C&W](#)
- [Elastic net attack](#)

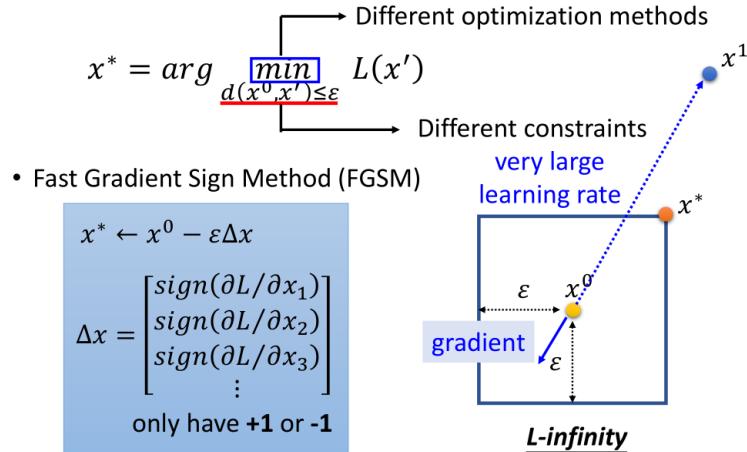
- [Spatially Transformed](#)
- [One Pixel Attack](#)
- ..... only list a few

### Fast Gradient Sign Method (FGSM)

攻击方法的主要区别在于Different optimization methods和Different constraints

FGSM是一种简单的model attack方法，梯度下降的时候计算的梯度，如果是负的，则直接为-1，如果是正的，则直接为+1。所以 $x^0$ 的更新要么为 $-\varepsilon$ ，要么为 $+\varepsilon$ ，只更新一次就结束了。

这个算法的思想就是只攻击一次就好（减去或者加上 $\varepsilon$ ）。多攻击几次确实会更好，所以FGSM有一个进阶方法Iterative fast gradient sign method (I-FGSM)



FGSM只在意gradient的方向，不在意它的大小。假设constrain用的是L-infinity，FGSM相当于设置了一个很大的learning rate，导致可以马上跳出范围，再拉回来。

### White Box v.s. Black Box

In the previous attack, we fix network parameters  $\theta$  to find optimal  $x'$ .

To attack, we need to know network parameters  $\theta$

- This is called **White Box Attack**.

Are we safe if we do not release model?

- You cannot obtain model parameters in most on-line API.

No, because **Black Box Attack** is possible.

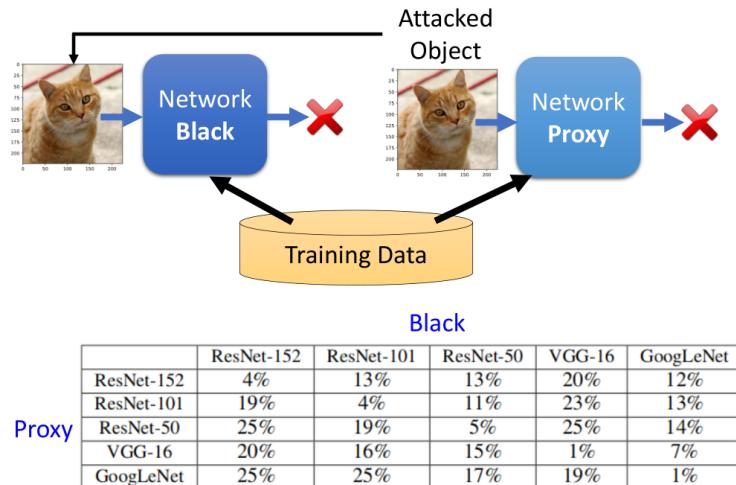
### Black Box Attack

If you have the **training data** of the target network

- Train a **proxy network** yourself
- Using the proxy network to generate attacked objects

Otherwise, obtaining input-output pairs from target network

如图中描述的是模型辨识正确的概率，也就是攻击失败的概率。上述五种神经网络的架构是不一样的，但是我们可以看到即使是不同架构的模型攻击成功的概率也是非常高的，而相同的架构的模型攻击成功率则明显是更高的。



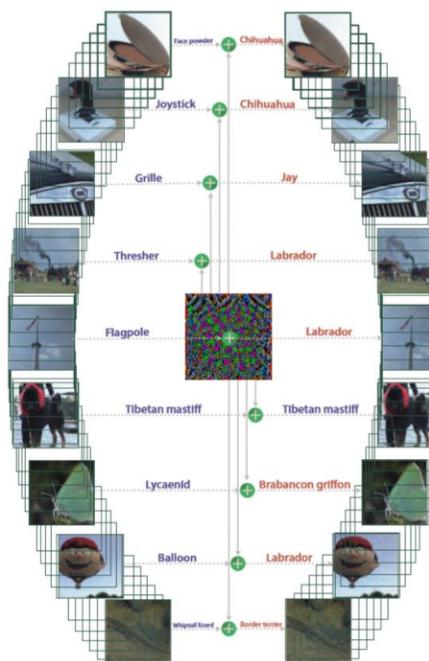
<https://arxiv.org/pdf/1611.02770.pdf>

## Universal Adversarial Attack

核心精神是找一个通用的攻击向量，将其叠加到任意样本上都会让模型辨识出错。

<https://arxiv.org/abs/1610.08401>

这件事做成以后，你可以想象，只要在做辨识任务的摄像机前面贴一张噪声照片，就可以让所有结果都出错。另外，这个通用攻击甚至也可以做上述的黑盒攻击。

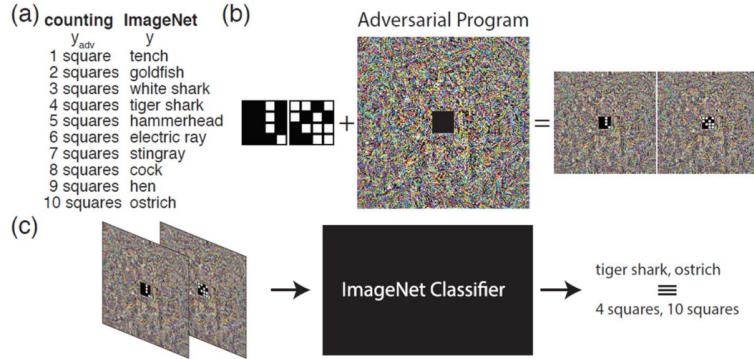


Black Box Attack is also possible!

## Adversarial Reprogramming

这个攻击的核心精神是：通过找一些噪声，让机器的行为发生改变，达到重编程实现其他功能的效果。改变原来network想要做的事情，例如从辨识动物变成数方块

当我们把图片中间贴上上图中那种方块图，机器就会帮我们数出途中方块的个数，如果有一个方块会输出tench，有两个方块就输出goldfish... 这件事还挺神奇的，因为我们并没有改变机器的任何参数，我们只是用了和前述相同的方法，找了一个噪声图片，然后把要数方块的图贴在噪声上，输入模型就会让模型帮我们数出方块的个数。具体方法细节参考引用文章。



Gamaleldin F. Elsayed, Ian Goodfellow, Jascha Sohl-Dickstein, "Adversarial Reprogramming of Neural Networks", ICLR, 2019

## Attack in the Real World

我们想知道上述的攻击方法是否能应用在现实生活中，上述的所有方法中加入的噪声其实都非常的小，在数字世界中这些噪声会对模型的判断造成很大影响似乎是很合理的，但是在真实的世界中，机器是通过一个小相机看世界的，这样的小噪声通过相机以后可能就没有了。通过镜头是否能识别到那些微小的杂讯呢？实验把攻击的图像打印出来，然后用摄像头识别。证明这种攻击时可以在现实生活中做到的。

对人脸识别上进行攻击，可以把噪声变成实体眼镜，带着眼镜就可以实现攻击。

需要确保：多角度都是成功的；噪声不要有非常大的变化（比如只有1像素与其他不同），要以色块方式呈现，这样方便摄像头能看清；不用打印机打不出来的颜色。

1. An attacker would need to find perturbations that generalize beyond a single image.
2. Extreme differences between adjacent pixels in the perturbation are unlikely to be accurately captured by cameras.
3. It is desirable to craft perturbations that are comprised mostly of colors reproducible by the printer

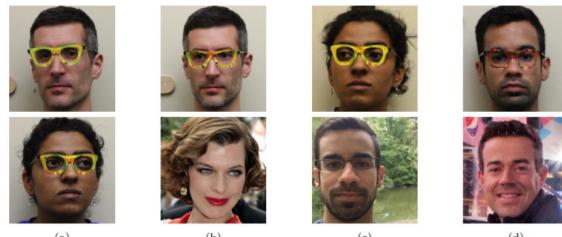


Figure 4: Examples of successful impersonation and dodging attacks. Fig. (a) shows  $S_A$  (top) and  $S_B$  (bottom) dodging against  $DNN_B$ . Fig. (b)-(d) show impersonations. Impersonators carrying out the attack are shown in the top row and corresponding impersonation targets in the bottom row. Fig. (b) shows  $S_A$  impersonating Milla Jovovich (by Georges Biard / CC BY-SA / cropped from <https://goo.gl/GlsWIC>); (c)  $S_B$  impersonating  $S_C$ ; and (d)  $S_C$  impersonating Carson Daly (by Anthony Quintano / CC BY / cropped from <https://goo.gl/VfnDct>).

在不同角度变成限速45的标志

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5' 0°					
5' 15°					
10' 0°					
<a href="https://arxiv.org/abs/1707.08945">https://arxiv.org/abs/1707.08945</a>					
10' 30°					
40' 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

## Beyond Images

### You can attack audio

- [https://nicholas.carlini.com/code/audio\\_adversarial\\_examples/](https://nicholas.carlini.com/code/audio_adversarial_examples/)
- <https://adversarial-attacks.net>

### You can attack text

<https://arxiv.org/pdf/1707.07328.pdf>

**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

## Defense

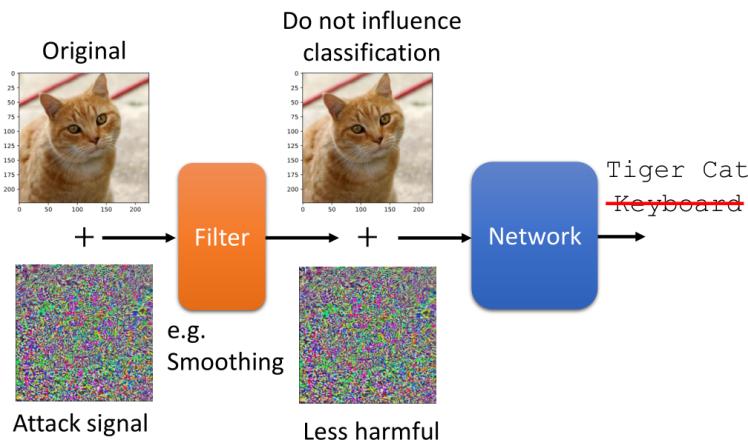
Adversarial Attack **cannot** be defended by weight regularization, dropout and model ensemble.

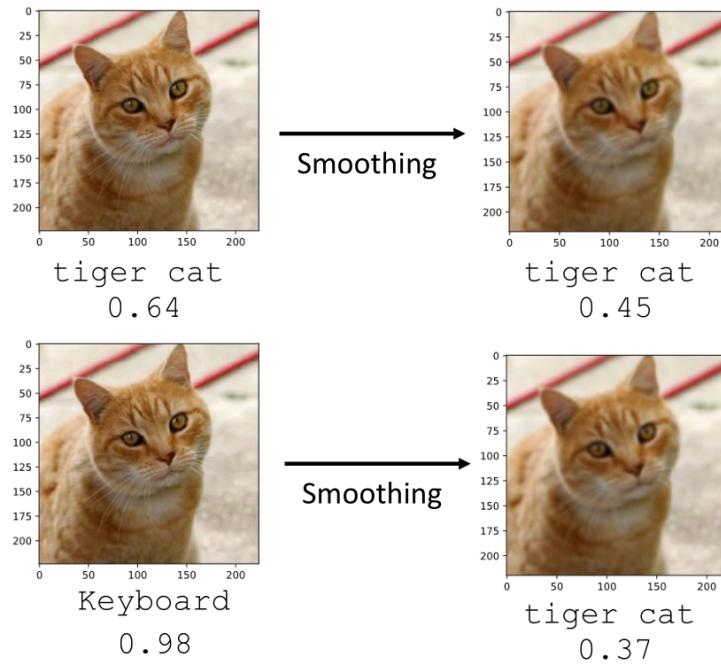
Two types of defense

- Passive defense: Finding the attached image without modifying the model
  - Special case of Anomaly Detection, 找出加入攻击的图片, 不修改网络
- Proactive defense: Training a model that is robust to adversarial attack

### Passive Defense

加入filter, 比如smoothing。

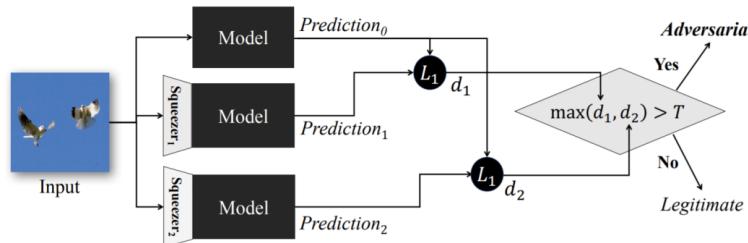




解释是，找攻击信号时，实际上只有某一种或者某几种攻击信号能够让攻击成功。虽然这种攻击信号能够让model失效，但是只有某几个方向上的攻击信号才能work。一旦加上一个filter，讯号改变了，攻击便失效。

### Feature Squeeze

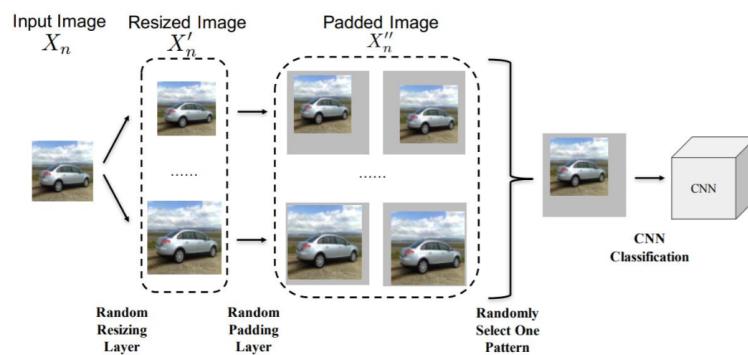
先用正常的model做一个prediction，再用model之前加上squeeze模块的model做一个prediction，如果这两个prediction差距很大，说明这个image被攻击过。



<https://arxiv.org/abs/1704.01155>

### Randomization at Inference Phase

还可以在原图基础上做一点小小的缩放，一些小padding，让攻击杂讯与原来不太一样，让攻击杂讯失效。



<https://arxiv.org/abs/1711.01991>

但问题是这样在model之前加“盾牌”的方法，有一个隐患是，如果，“盾牌”的机制被泄露出去，那么攻击仍然很有可能成功。（把filter想象成network的第一个layer，再去训练攻击杂讯即可）