Adversarial Attack

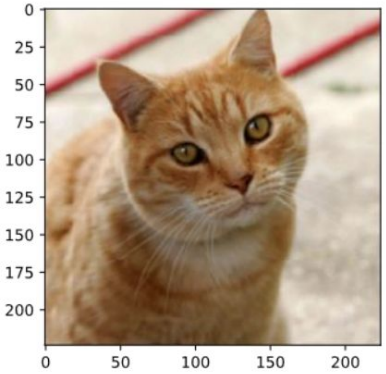
助教：b05902121 黃冠博
老師：李宏毅 教授

Adversarial Attack Outline

- Attacks on Images
  - one pixel attack
    - differential evolution
- Attacks on Audio
  - Attacks on ASR
  - Attacks on ASV
  - Hidden Voice Attack

Attacks on Images

Original Image



x^0

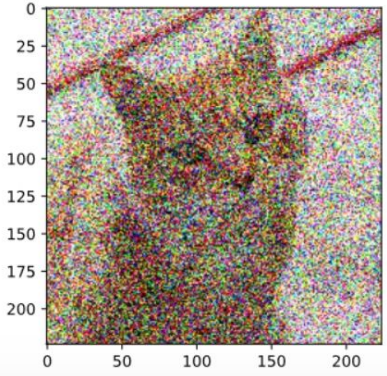
Network

Something Else

~~Tiger Cat~~

0.64

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} + \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \vdots \end{bmatrix}$$

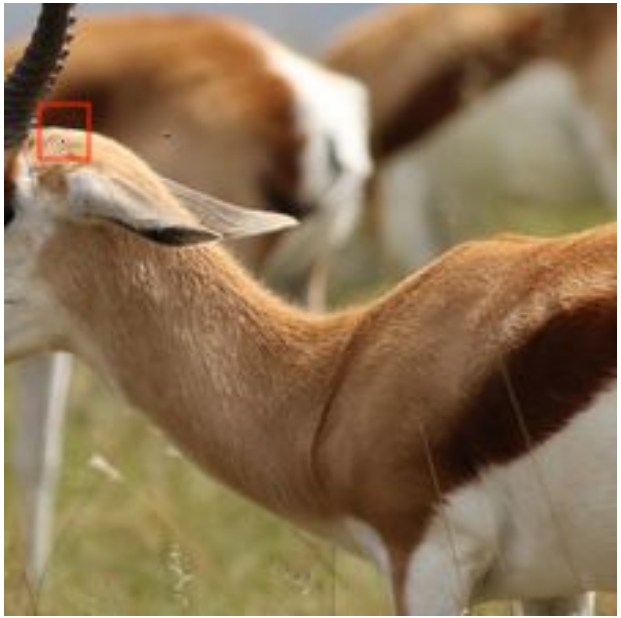


Attacked Image

$$x' = x^0 + \Delta x$$

http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/Attack%20(v8).pdf

One Pixel Attack



One Pixel Attack vs. 一般的Attack

一般的 attack

0	42	30	32
12	12	11	0
32	0	3	3
9	23	22	0

$$\underset{e(\mathbf{x})^*}{\text{maximize}} \quad f_{adv}(\mathbf{x} + e(\mathbf{x}))$$

$$\text{subject to} \quad \|e(\mathbf{x})\| \leq L$$

or

L-infinity

one pixel attack

0	0	0	0
0	0	50	0
0	0	0	0
0	0	0	0

$$\underset{e(\mathbf{x})^*}{\text{maximize}} \quad f_{adv}(\mathbf{x} + e(\mathbf{x}))$$

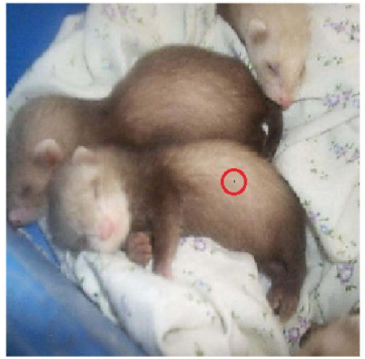
$$\text{subject to} \quad \|e(\mathbf{x})\|_0 \leq d,$$

d = 1

Cup(16.48%)
Soup Bowl(16.74%)

Bassinet(16.59%)
Paper Towel(16.21%)

Teapot(24.99%)
Joystick(37.39%)



Hamster(35.79%)
Nipple(42.36%)

https://arxiv.org/pdf/1710.08864.pdf

One Pixel Attack

- x: n-dimensional inputs, $x = (x_1, .., x_n)$
- f: image classifier (model的output過softmax)
- $f_t(x)$: 給定input x，model認為x是class t 的機率
- $e(x)$: additive adversarial perturbation according to x

untargeted attack:

$\arg\min_{e(x)^*} \ f_t(x + e(x))$

where t is the original class of x.

target attack:

$\arg\max_{e(x)^*} \ f_{adv}(x + e(x))$

where adv is the targeted class.

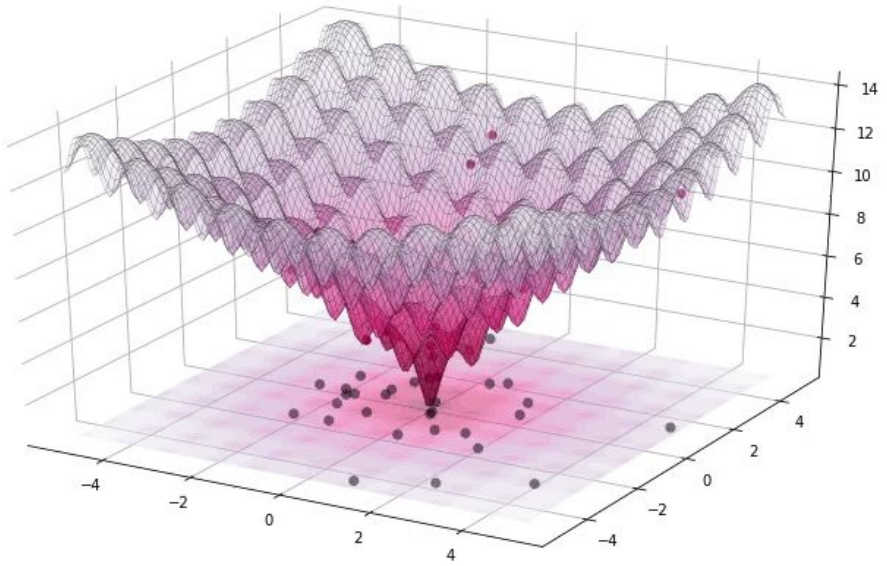
Both of the attack are subject to $||e(x)||_0 = 1$.

How do we find the exact pixel and value ?

- Brute Force
  - 224 * 224 = 50176, takes very long time

Do we really need the best perturbation ?

Differential Evolution



By Pablormier - https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=62208829

Differential Evolution

During each iteration another set of candidate solutions (children) is generated according to the current population (parents). Then the children are compared with their corresponding parents, surviving if they are more fitted (possess higher fitness value) than their parents. In such a way, only comparing the parent and his child, the goal of keeping diversity and improving fitness values can be simultaneously achieved.

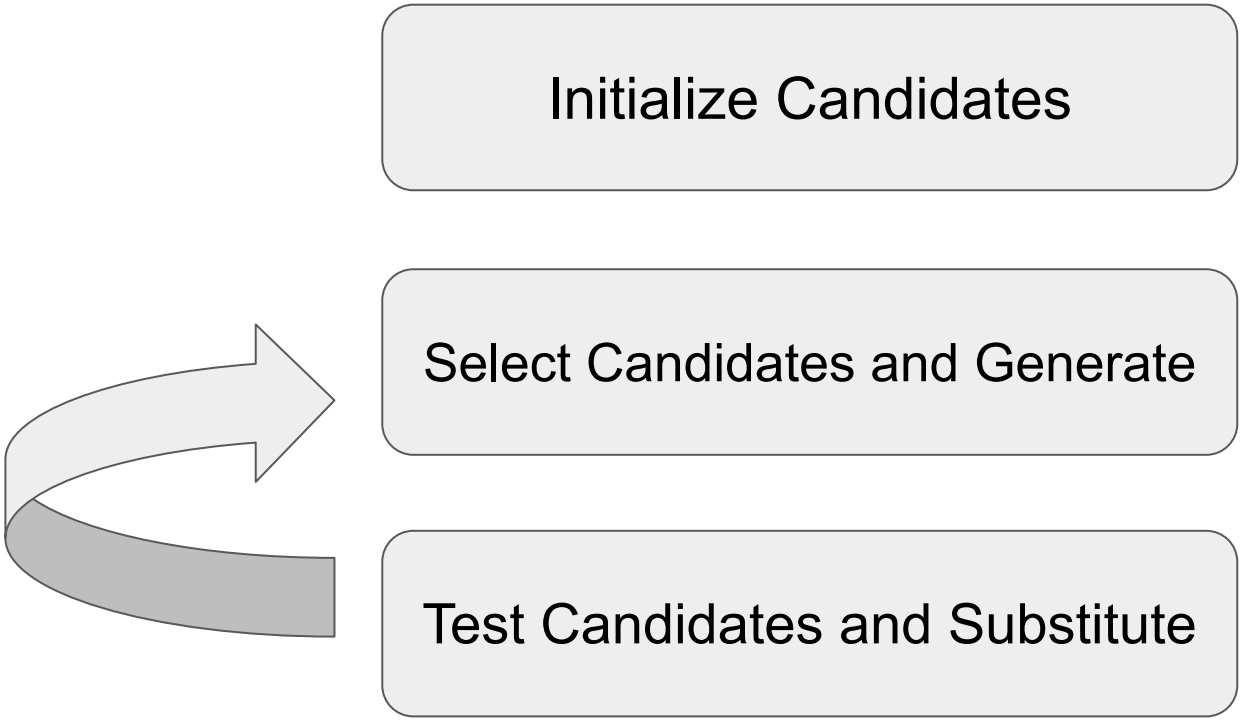
https://arxiv.org/pdf/1710.08864.pdf

Differential Evolution

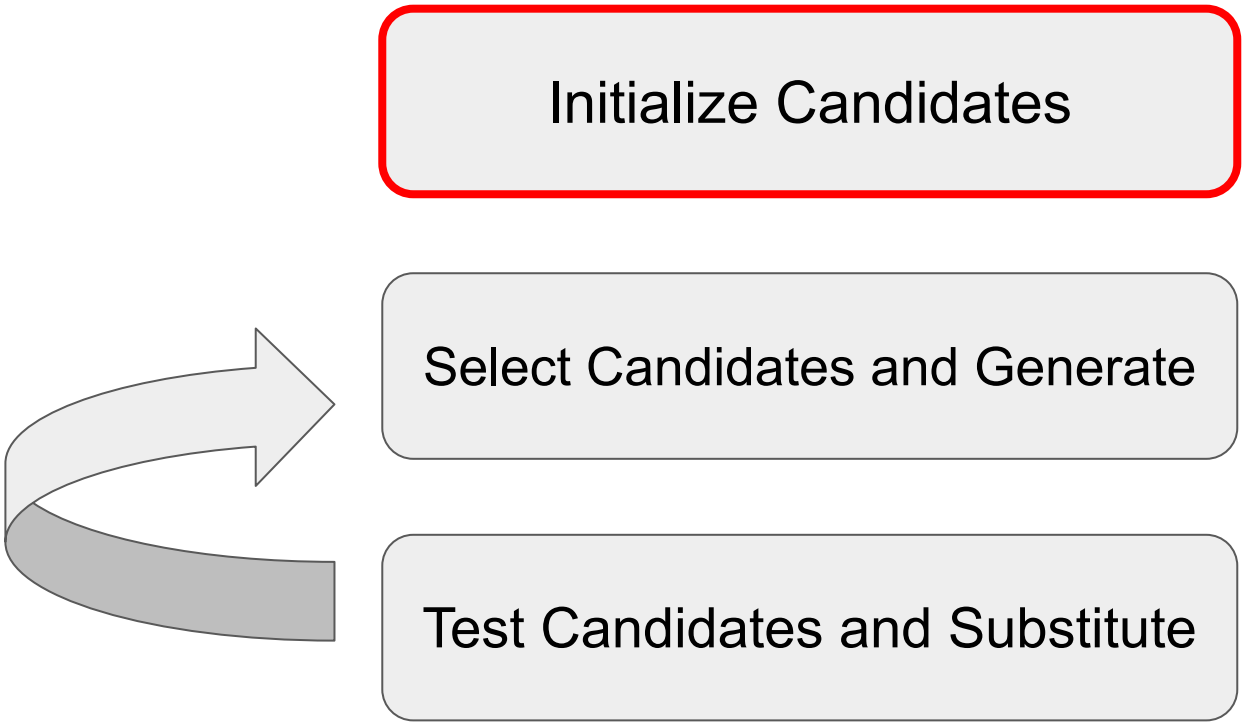
- Higher probability of Finding Global Optima
  - due to diversity keeping mechanisms and the use of a set of candidate solutions
- Require Less Information from Target System
  - 相比FGSM, DE不需要算gradient, 因此不需要攻擊對象model太多的細節
  - independent of the classifier used

https://arxiv.org/pdf/1710.08864.pdf

Differential Evolution



Differential Evolution



Differential Evolution

$$f(x) = \frac{1}{4}\Sigma x_i^2$$

$$x = \{x_1, x_2, x_3, x_4\}$$

$$-5 \leq x_i \leq 5$$

```
fobj = lambda x: sum(x**2)/len(x)
```

https://arxiv.org/pdf/1710.08864.pdf

Differential Evolution

```
>>> pop = np.random.rand(popsize, dimensions)
>>> pop

#        x[0]   x[1]   x[2]   x[3]
array([[ 0.09,  0.01,  0.4 ,  0.21],
       [ 0.04,  0.87,  0.52,  0.  ],
       [ 0.96,  0.78,  0.65,  0.17],
       [ 0.22,  0.83,  0.23,  0.84],
       [ 0.8 ,  0.43,  0.06,  0.44],
       [ 0.95,  0.99,  0.93,  0.39],
       [ 0.64,  0.97,  0.82,  0.06],
       [ 0.41,  0.03,  0.89,  0.24],
       [ 0.88,  0.29,  0.15,  0.09],
       [ 0.13,  0.19,  0.17,  0.19]])
```

https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/

Differential Evolution

$$f(x) = \sum x_i^2/4$$

$$x = \{x_1, x_2, x_3, x_4\}$$

$$-5 \leq x_i \leq 5$$

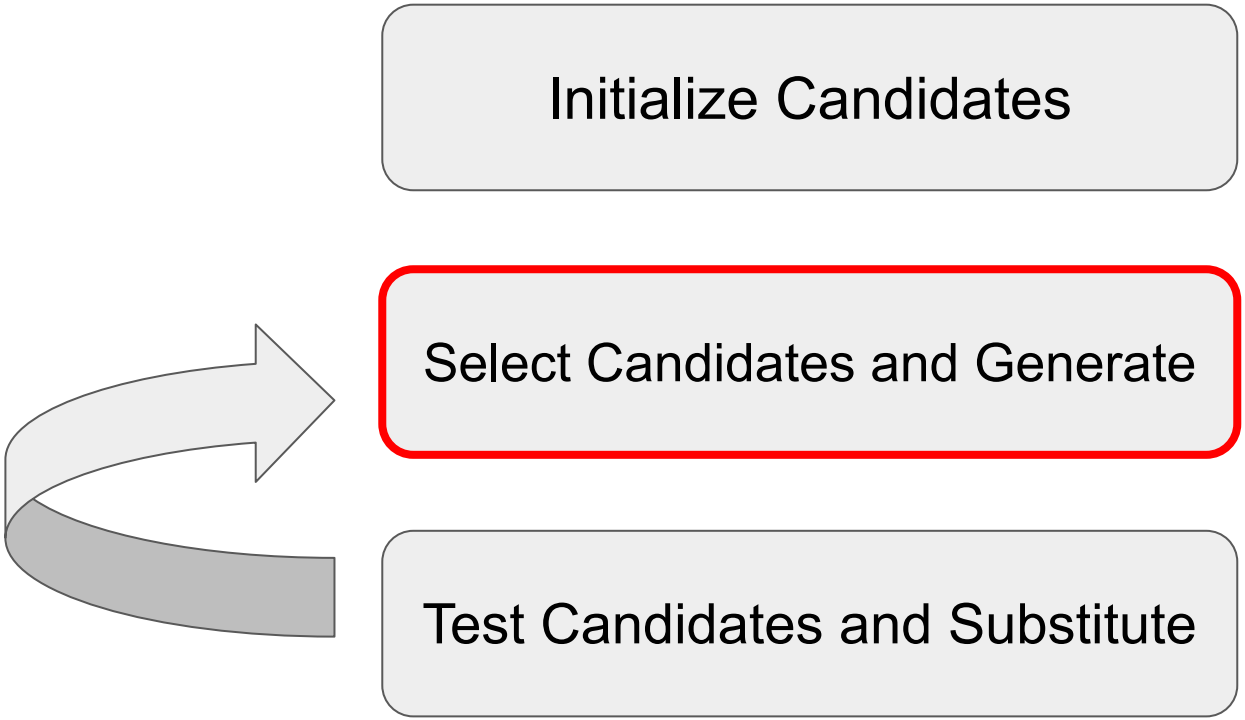
```
>>> pop_denorm = min_b + pop * diff
>>> pop_denorm
```

```
array([[-4.06, -4.89, -1.  , -2.87],
       [-4.57,  3.69,  0.19, -4.95],
       [ 4.58,  2.78,  1.51, -3.31],
       [-2.83,  3.27, -2.72,  3.43],
       [ 3.  , -0.68, -4.43, -0.57],
       [ 4.47,  4.92,  4.27, -1.05],
       [ 1.36,  4.74,  3.19, -4.37],
       [-0.89, -4.67,  3.85, -2.61],
       [ 3.76, -2.14, -3.53, -4.06],
       [-3.67, -3.14, -3.34, -3.06]])
```

```
12.3984504837
14.7767132816
10.4889711137
9.44800715266
7.34888318457
15.8691538075
13.4024093959
11.0571791104
11.9129095178
10.9544056745
```

https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/

Differential Evolution



Differential Evolution

```
>>> pop = np.random.rand(popsize, dimensions)
>>> pop

#        x[0]   x[1]   x[2]   x[3]
array([[ 0.09,  0.01,  0.4 ,  0.21],
       [ 0.04,  0.87,  0.52,  0.  ],
       [ 0.96,  0.78,  0.65,  0.17],
       [ 0.22,  0.83,  0.23,  0.84],
       [ 0.8 ,  0.43,  0.06,  0.44],
       [ 0.95,  0.99,  0.93,  0.39],
       [ 0.64,  0.97,  0.82,  0.06],
       [ 0.41,  0.03,  0.89,  0.24],
       [ 0.88,  0.29,  0.15,  0.09],
       [ 0.13,  0.19,  0.17,  0.19]])
```

https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/

```
>>> selected = np.random.choice(idxs, 3, replace=False)
>>> selected
array([[ 0.04,  0.87,  0.52,  0.  ],  # a
       [ 0.8 ,  0.43,  0.06,  0.44],  # b
       [ 0.41,  0.03,  0.89,  0.24]]) # c
```

```
# mut = 0.8
>>> mutant = a + mut * (b - c)

array([ 0.35,  1.19, -0.14,  0.17])
```

```
>>> np.clip(mutant, 0, 1)

array([ 0.35,  1.  ,  0.  ,  0.17])
```

target

```
[ 0.09,  0.01,  0.4 ,  0.21]
```

https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/

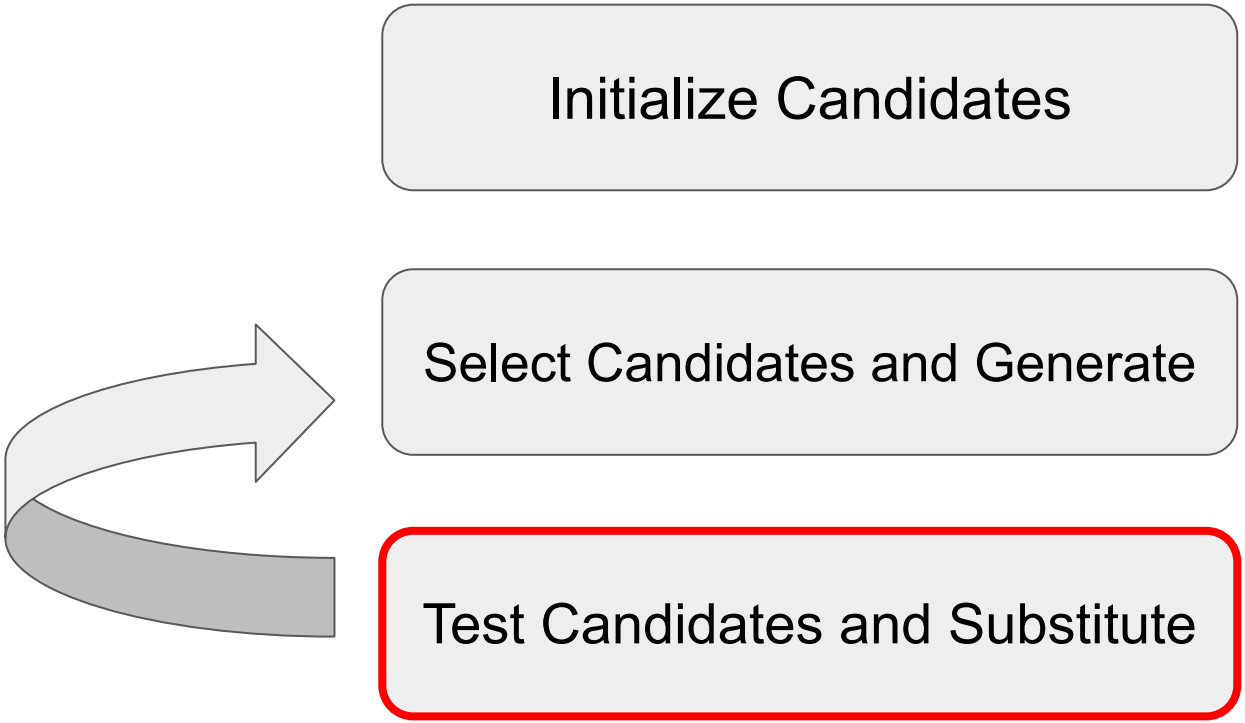
Differential Evolution

```
>>> cross_points = np.random.rand(dimensions) < crossp
>>> cross_points

array([False,  True, False,  True], dtype=bool)
```

```
>>> trial = np.where(cross_points, mutant, pop[j]) # j = 0
>>> trial

array([ 0.09,  1.  ,  0.4 ,  0.17])
```

https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/

Differential Evolution



Differential Evolution

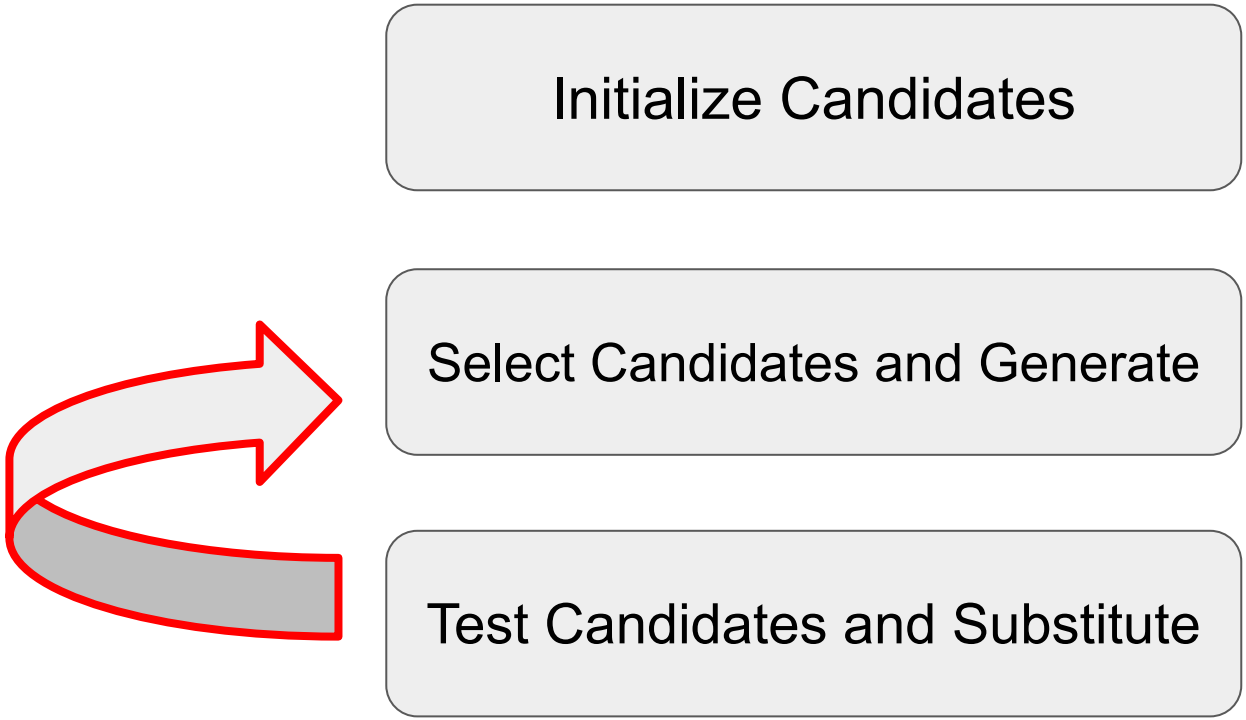
```
>>> trial_denorm = min_b + trial * diff
>>> trial_denorm

array([-4.1,  5. , -1. , -3.3])
```

```
>>> fobj(mutant_denorm)

13.425000000000001
```

the trial vector is worse than the target vector (13.425 > 12.398), so the target vector is preserved and the trial vector is discarded

https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/

Differential Evolution



One Pixel Attack Recap

- x: n-dimensional inputs, $x = (x_1, .., x_n)$
- f: image classifier (model的output過softmax)
- $f_t(x)$: 給定input x，model認為x是class t 的機率
- $e(x)$: additive adversarial perturbation according to x

untargeted attack:

$\arg\min_{e(x)^*} \ f_t(x + e(x))$

where t is the original class of x.

target attack:

$\arg\max_{e(x)^*} \ f_{adv}(x + e(x))$

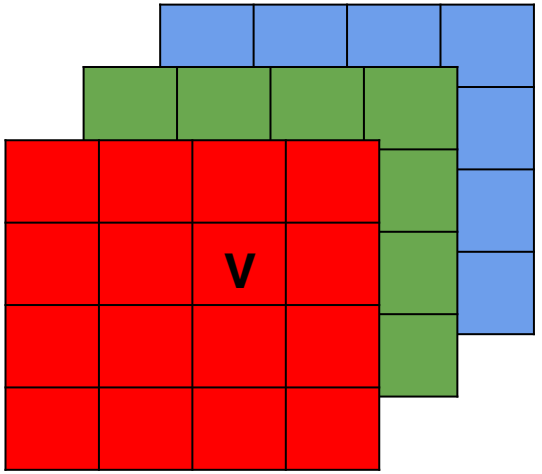
where adv is the targeted class.

Both of the attack are subject to $||e(x)||_0 = 1$.

How is DE used on one pixel attack ?

RGB image:

(channel=3, img_size, img_size)

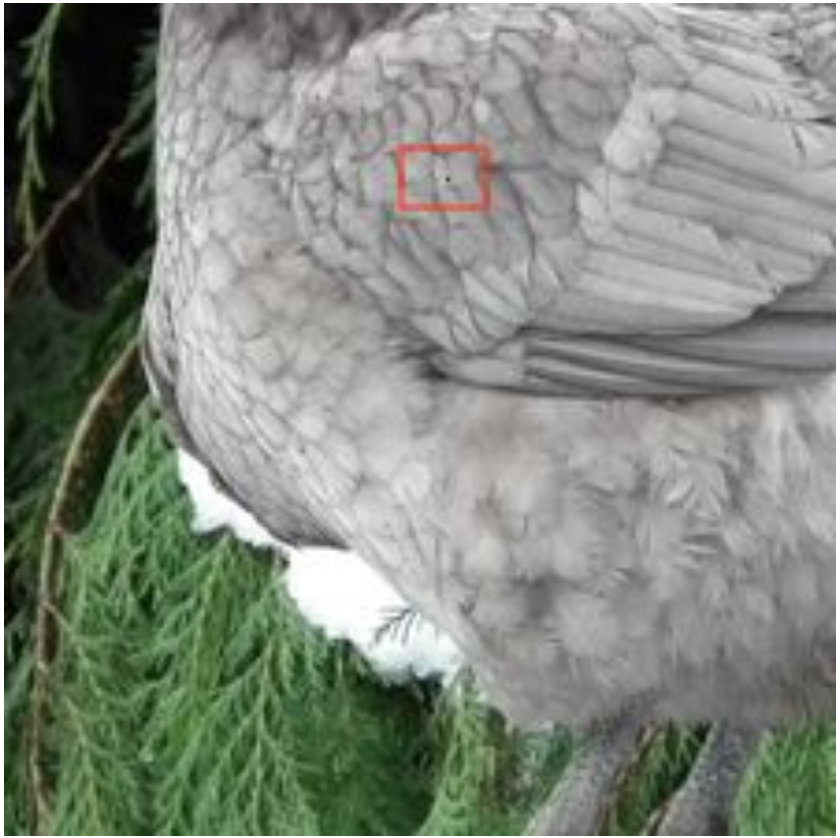


x, y coordinates:

the x and y coordinates of the pixel to attack

(x, y, R,G,B)

Some other examples



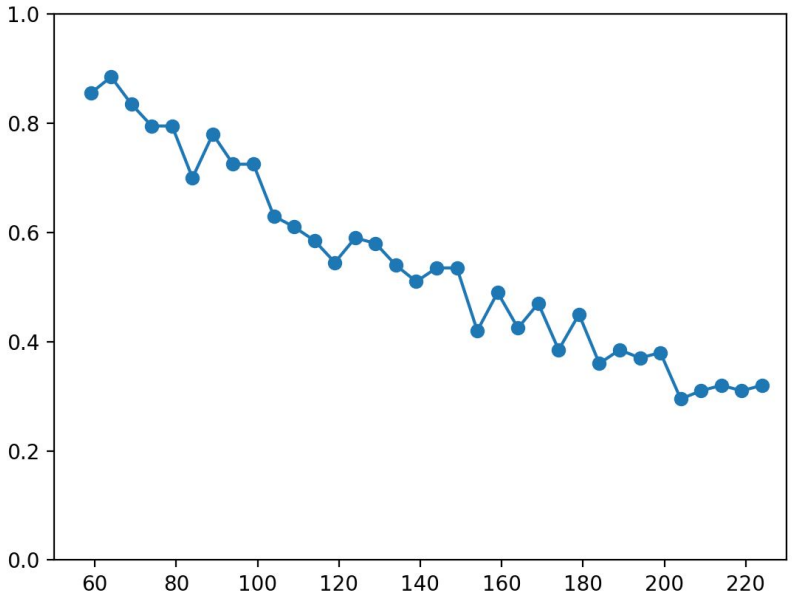
Discussion

- 成功率與圖片大小的關係

iteration = 50
candidate num = 100

Discussion

- 成功率與圖片大小的關係



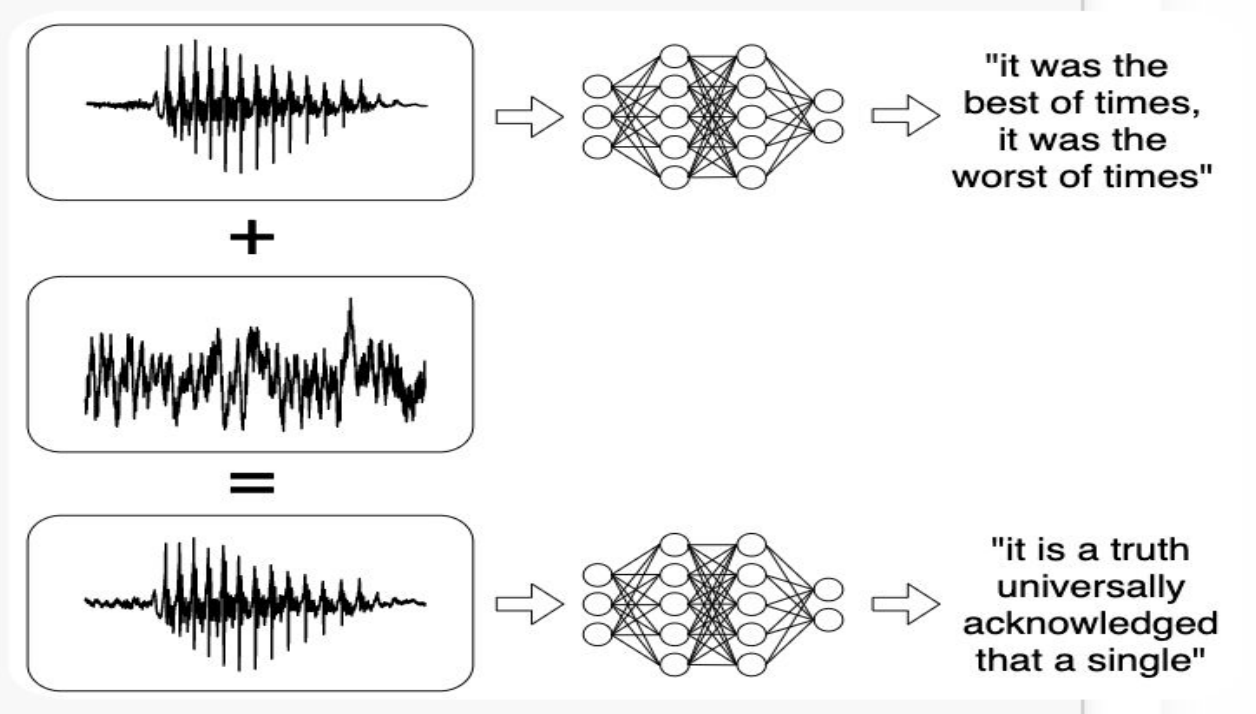
iteration = 50
candidate num = 100

Attacks on Audio

Adversarial Attack Outline

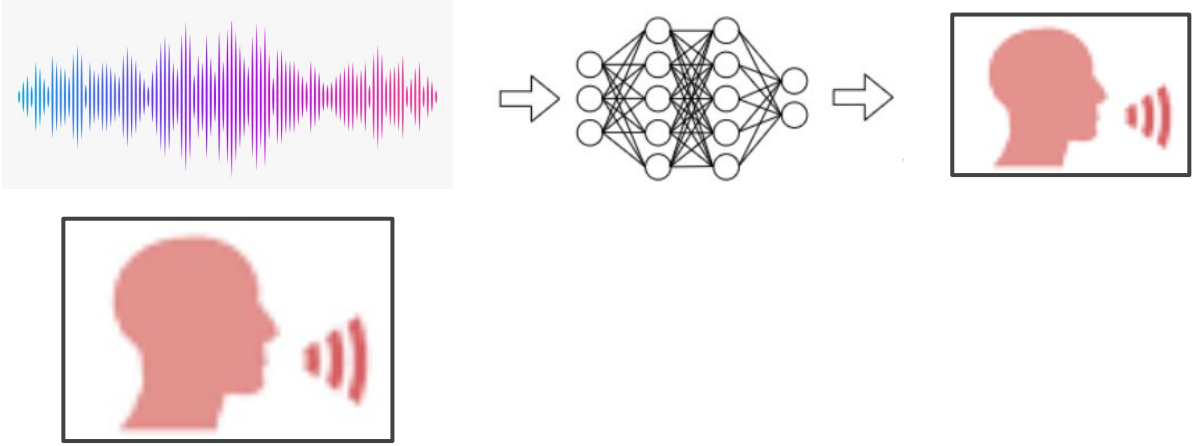
- Attacks on Images
  - one pixel attack
    - differential evolution
- Attacks on Audio
  - Attacks on ASR
  - Attacks on ASV
  - Hidden Voice Attack

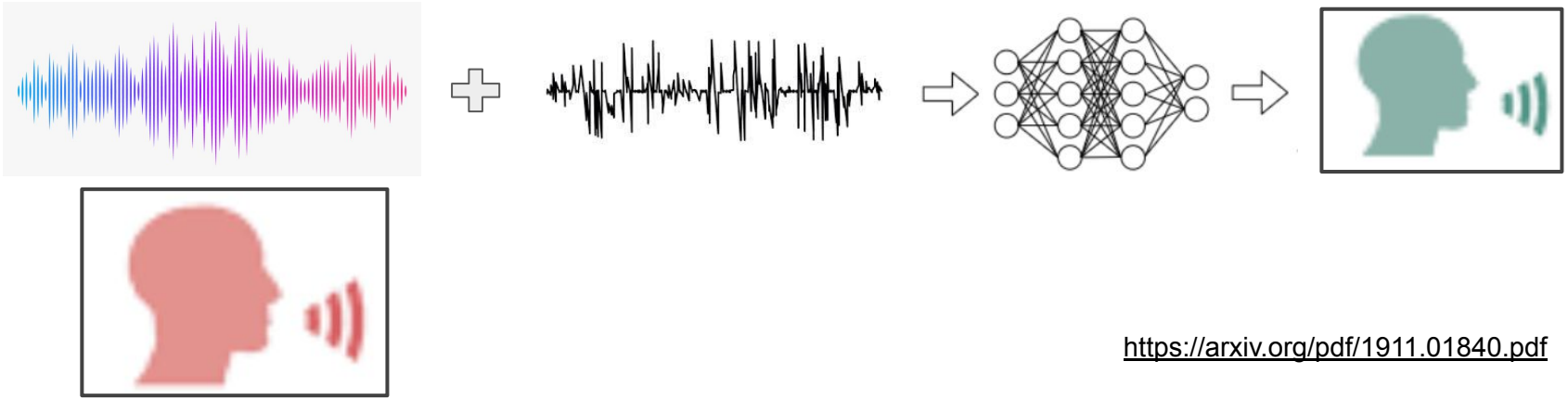
Attacks on ASR



https://nicholas.carlini.com/code/audio_adversarial_examples/

Attacks on ASV





https://arxiv.org/pdf/1911.01840.pdf

Wake Up Words

google home

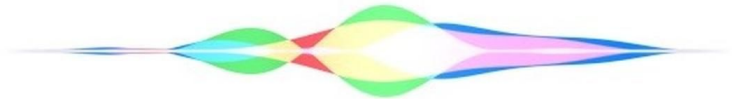
http://speech.ee.ntu.edu.tw/~tlkagk/courses/DLHLP20/Introduction%20(v9).pdf

Hidden Voice Attack

- Hey Siri ! -> @#%&^%$#!

~~"Hey, Siri"~~

@#%&^%$#!

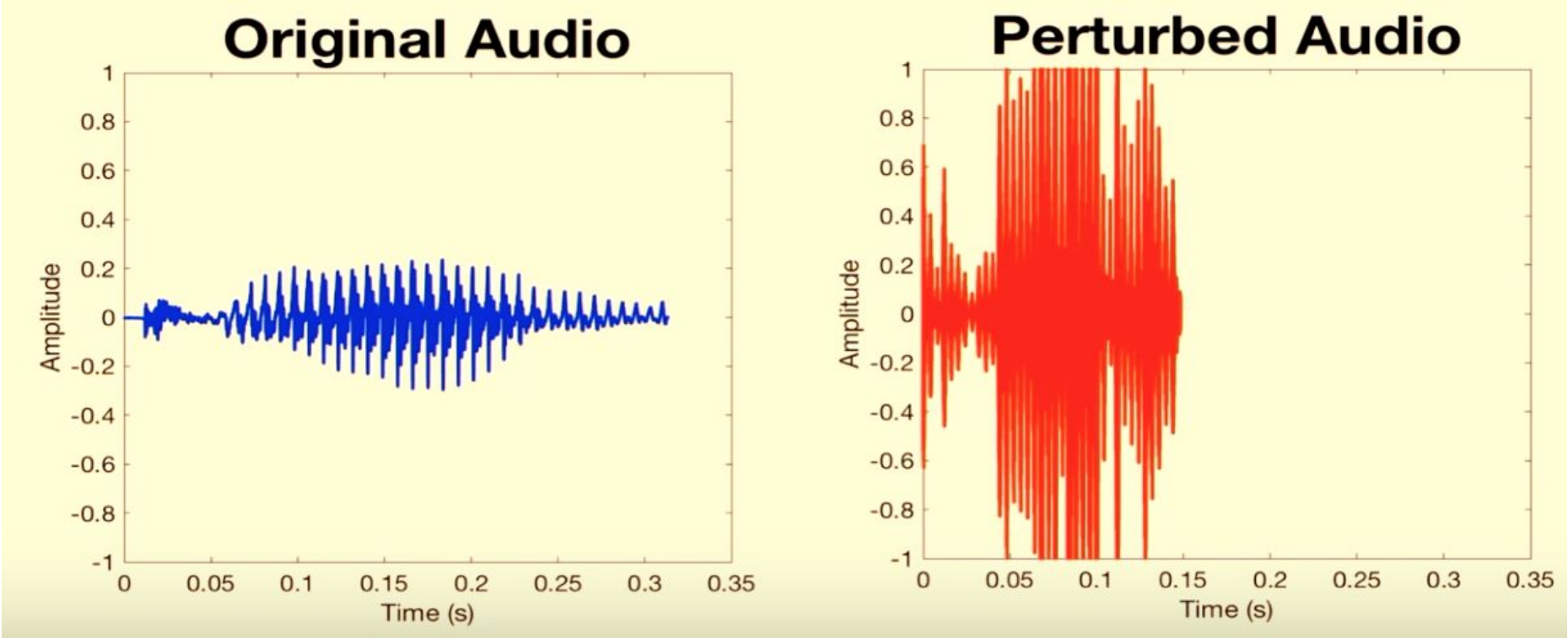


turn on the computer

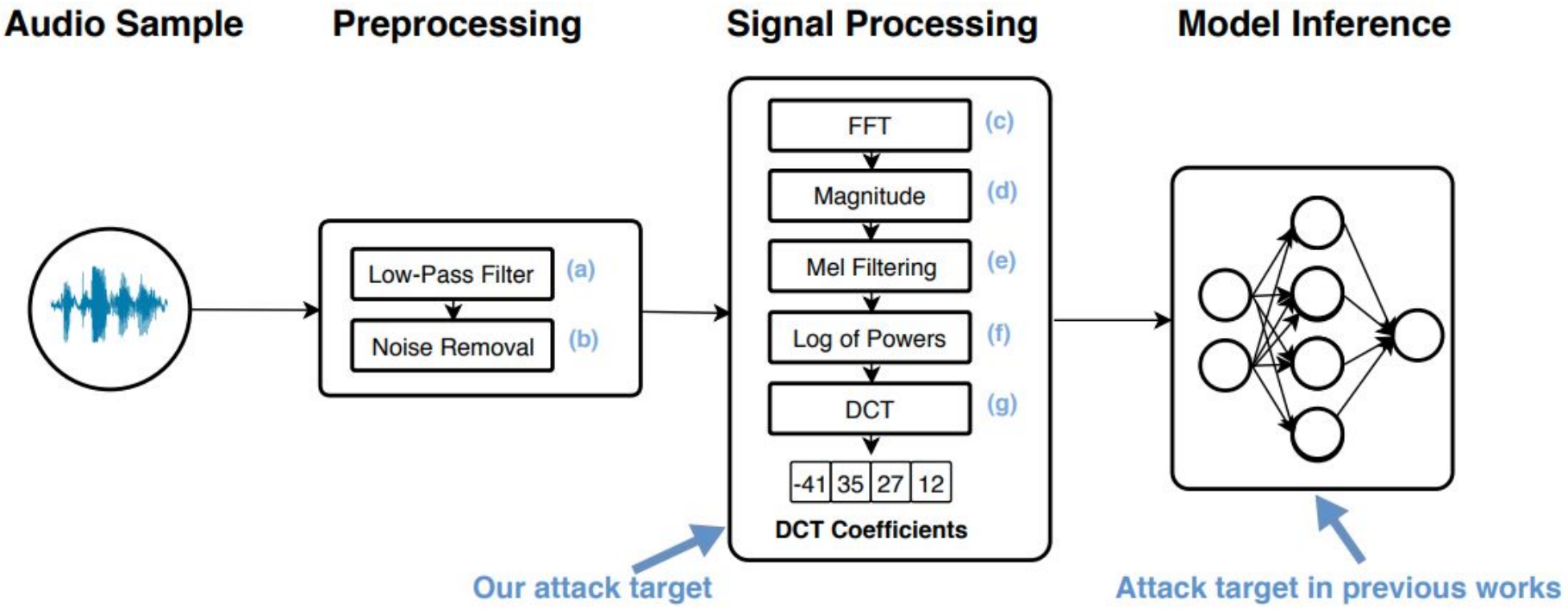
Psychoacoustics

- 研究人對於聲音的感知程度跟反應
- 感知極限：人只能聽到 20 Hz (0.02 kHz) 到 20,000 Hz (20 kHz) 範圍的波段
- 聲音定位：人藉由雙耳所聽到的聲音的時間差或是響度差來辨識出聲音的方位
  - 電競選手玩槍戰戴耳機就能聽出敵人的位置

https://en.wikipedia.org/wiki/Psychoacoustics


Original Audio
Amplitude
Time (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
Perturbed Audio
Amplitude
Time (s)

Signal Preprocessing

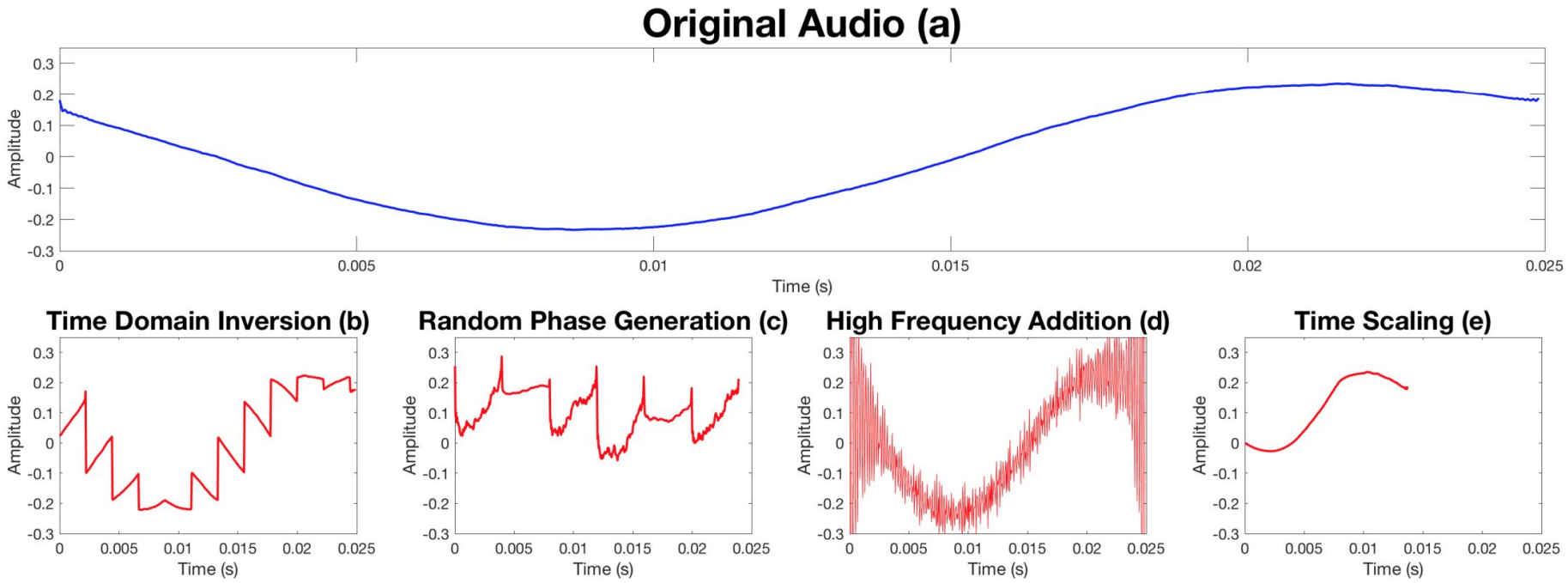


https://arxiv.org/pdf/1904.05734.pdf

Perturbation

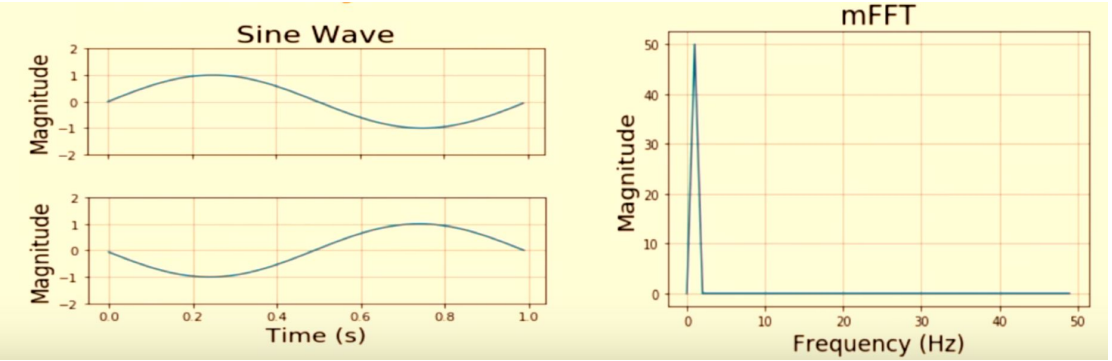
- Time Domain Inversion(TDI)
- Random Phase Generation(RPG)
- High Frequency Addition(HFA)
- Time Scaling(TS)

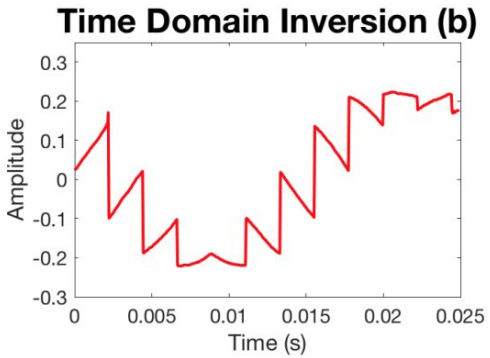
Perturbation



Time Domain Inversion (TDI)

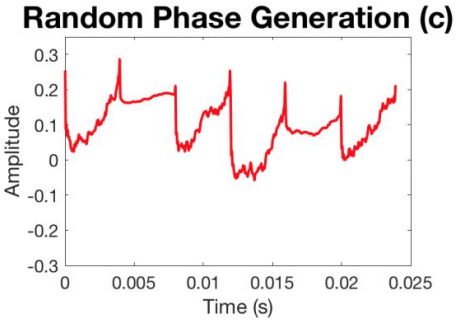
- 利用mFFT(magnitude fft)多對一的性質
- two completely different signals in the time domain can have similar spectra
- modifying the audio in the time domain while preserving its spectrum, by inverting the windowed signal
- inverting small windows across the entire signal removes the smoothness





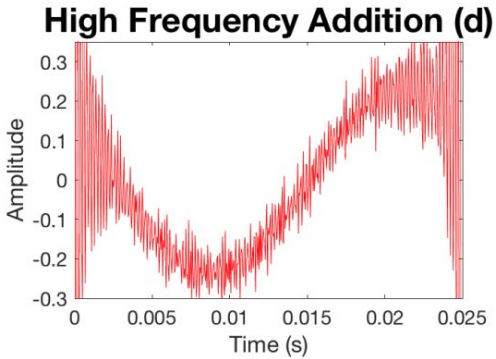
Random Phase Generation

- For each frequency in the spectrum, the FFT returns the value in complex form $a + bi$, where a and b define the phase of a signal.
- $magnitude = \sqrt{a^2 + b^2}$
- 調整a跟b，使得magnitude仍然相同，phase跟原本不同(聽起來就不會是原本的聲音)，但是magnitude spectrum仍會相同



High Frequency Addition (HFA)

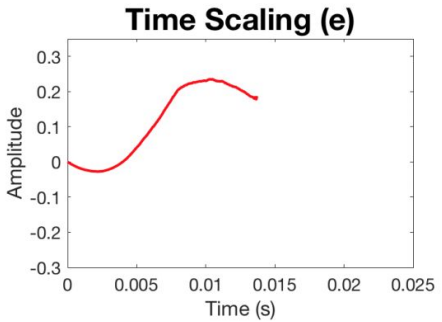
- Signal processing的過程當中，low-pass filter會把相較於人聲高很多的頻段濾掉以增加VPS(Voice Processing System)的準確率
- add high frequencies to the audio that are filtered out during the preprocessing stage
- create high frequency sine waves and add it to the real audio
- If the sine waves have enough intensity, it has the potential to mask the underlying audio command to the human ear.

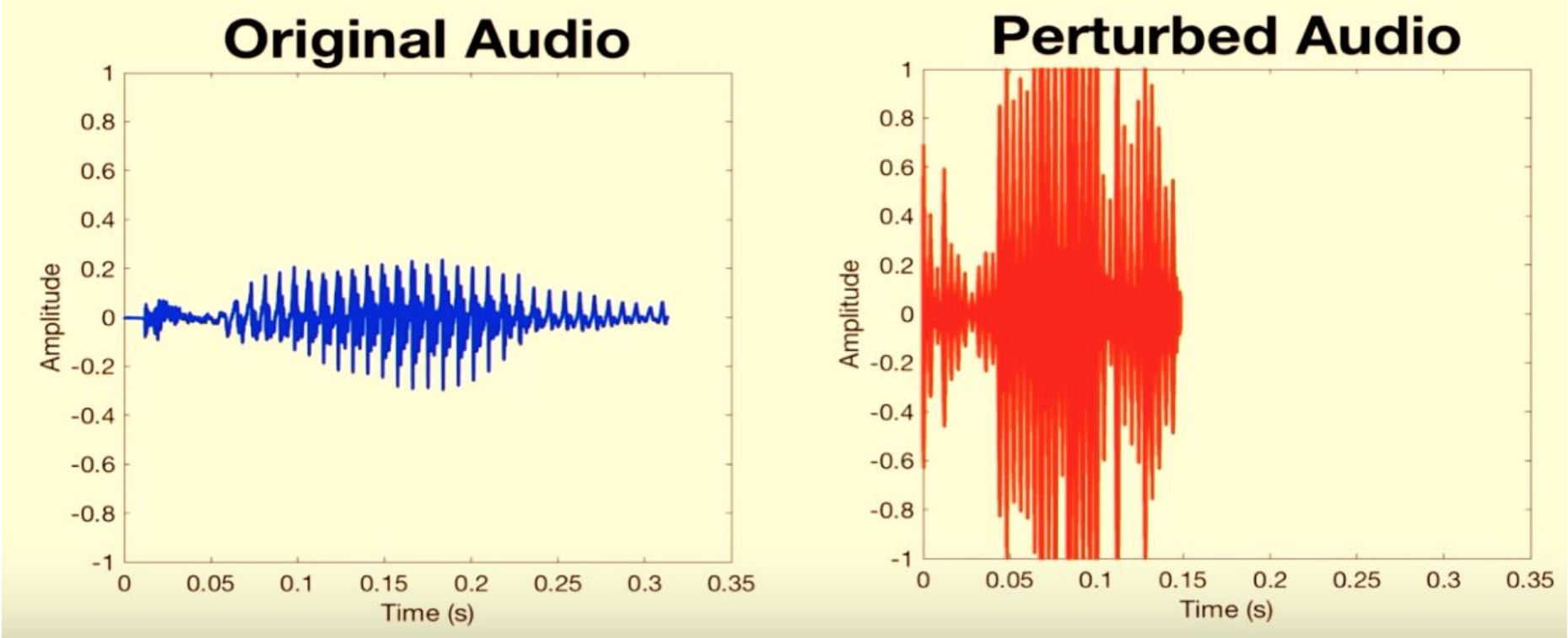


Time Scaling (TS)

- 將音訊快轉到model能正確辨識但是人又聽不太懂在說什麼
- compress the audio in the time domain by discarding unnecessary samples and maintain the same sample rate
- the audio is shorter in time, but retains the same spectrum as the original

什麼是sample rate？
波是由好幾的點組成,
sample rate就是每秒有幾個 data point




Original Audio
Amplitude
Time (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
Perturbed Audio
Amplitude
Time (s)

Adversarial Attack Recap

- Attacks on Images
  - one pixel attack
    - differential evolution
- Attacks on Audio
  - Attacks on ASR
  - Attacks on ASV
  - Hidden Voice Attack

Reference

- A Tutorial on Differential Evolution with Python
- NDSS 2019 - Practical Hidden Voice Attacks against Speech and Speaker Recognition Systems(Youtube)