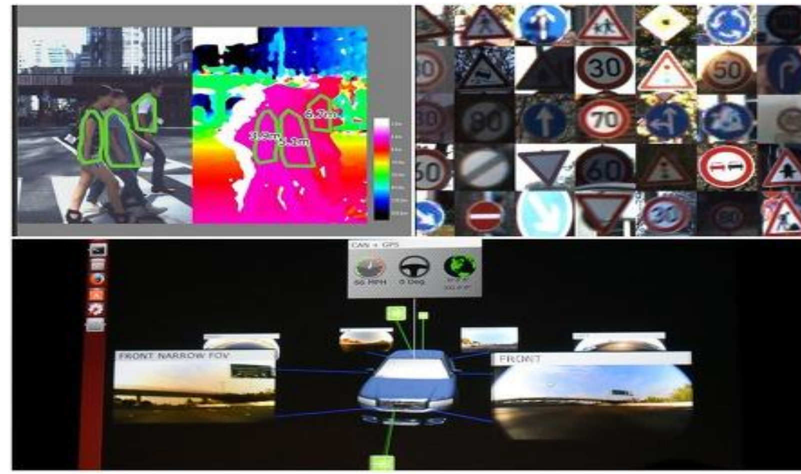
Adversarial AI

Gyu-Young Lee
Lee-Ahn Professional Engineer Office

Motivation – Deep Learning

- Deep Learning will change everything.



Motivation – Adversarial Attack

- Deep learning is vulnerable to adversarial samples.
- Autonomous vehicles can be crashed :
  - ① Slightly altering “STOP” signs
  - ② DNNs misclassify them
  - ③ The car would not stop
  - **④ Devastating consequences**

Driverless AI Car

STOP sign

Original Image

YIELD sign

Human

STOP sign

Crafted Image

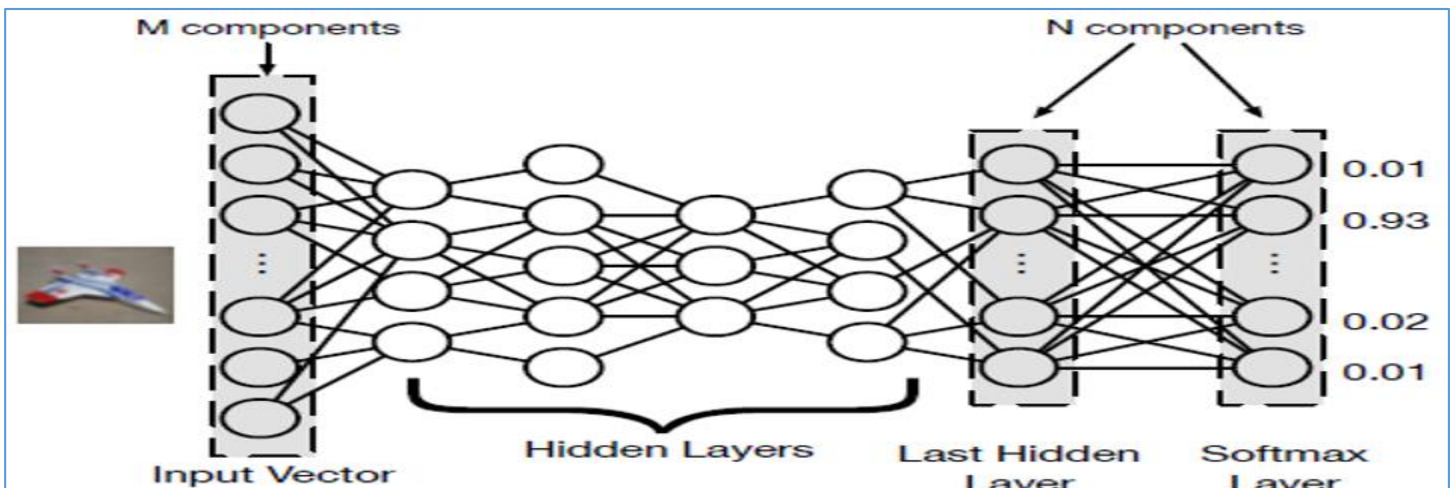
STOP sign

Project Summary

- TITLE : Adversarial AI
- SCOPE
  1) Training DNN with MNIST dataset
  2) Making adversarial samples
  3) To improve making adversarial samples
  4) To analyze defensive mechanism

Training DNN (approach)

- Constructing DNN environment
  - Machine Learning Library : **Tensor Flow**
  - Language : Python
  - Tool : Docker Toolbox, Jupyter open source
- Training DNN with MNIST database
  - **MNIST** dataset consists of 28×28 pixel images of handwritten digits having 50000 training images and 10000 test images

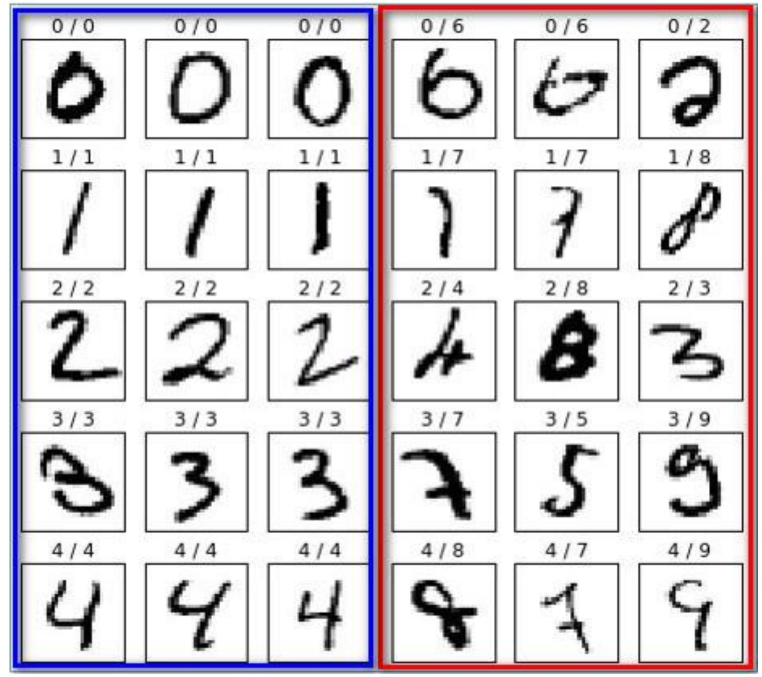


Training DNN (result)

[Accuracy performance of the DNN]

```
Step: 100, Loss: 3136.286377, Accuracy: 0.906700
Step: 200, Loss: 2440.697021, Accuracy: 0.928000
Step: 300, Loss: 1919.005249, Accuracy: 0.941900
Step: 400, Loss: 1982.860718, Accuracy: 0.939400
Step: 500, Loss: 1734.469971, Accuracy: 0.945500
Step: 600, Loss: 1377.535767, Accuracy: 0.956100
Step: 700, Loss: 1332.846313, Accuracy: 0.960600
Step: 800, Loss: 1184.055786, Accuracy: 0.963600
Step: 900, Loss: 1134.486084, Accuracy: 0.964700
Step: 1000, Loss: 1236.647095, Accuracy: 0.961900
Step: 1100, Loss: 1116.422852, Accuracy: 0.965500
Step: 1200, Loss: 1125.365234, Accuracy: 0.964700
Step: 1300, Loss: 1193.366577, Accuracy: 0.961900
Step: 1400, Loss: 1101.243652, Accuracy: 0.966800
Step: 1500, Loss: 1062.339966, Accuracy: 0.969400
Step: 1600, Loss: 1112.656494, Accuracy: 0.966600
Step: 1700, Loss: 953.149780, Accuracy: 0.972200
Step: 1800, Loss: 960.959900, Accuracy: 0.970900
Step: 1900, Loss: 1035.524414, Accuracy: 0.967900
Step: 2000, Loss: 990.451965, Accuracy: 0.970600
```

[Result of prediction test on the DNN]



[Left 3 columns] **Correct : Over 97%**

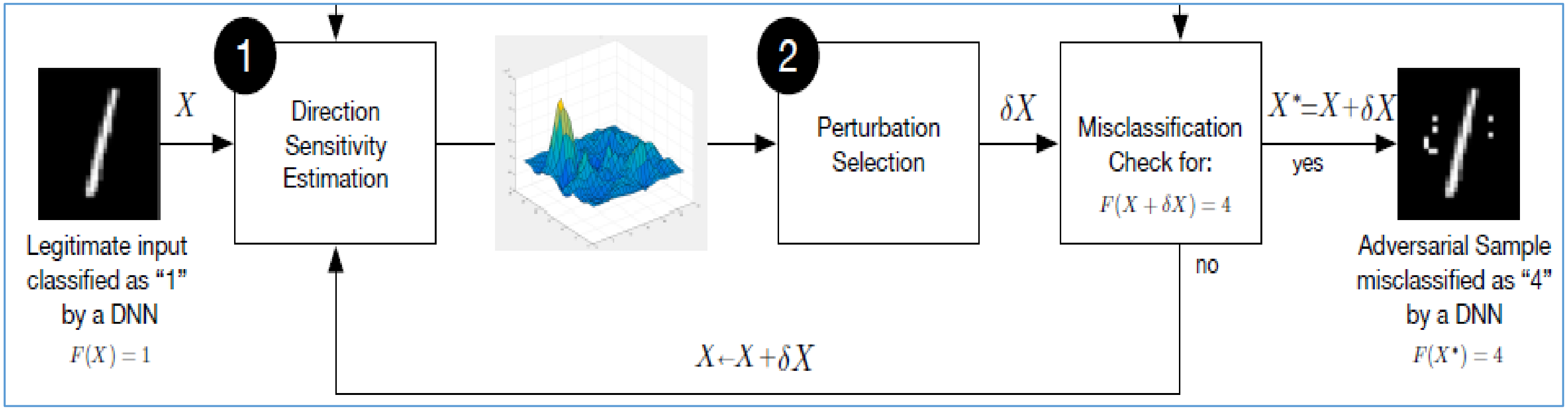
[Right 3 columns] **Incorrect : Below 3%**

[Notation] prediction/actual

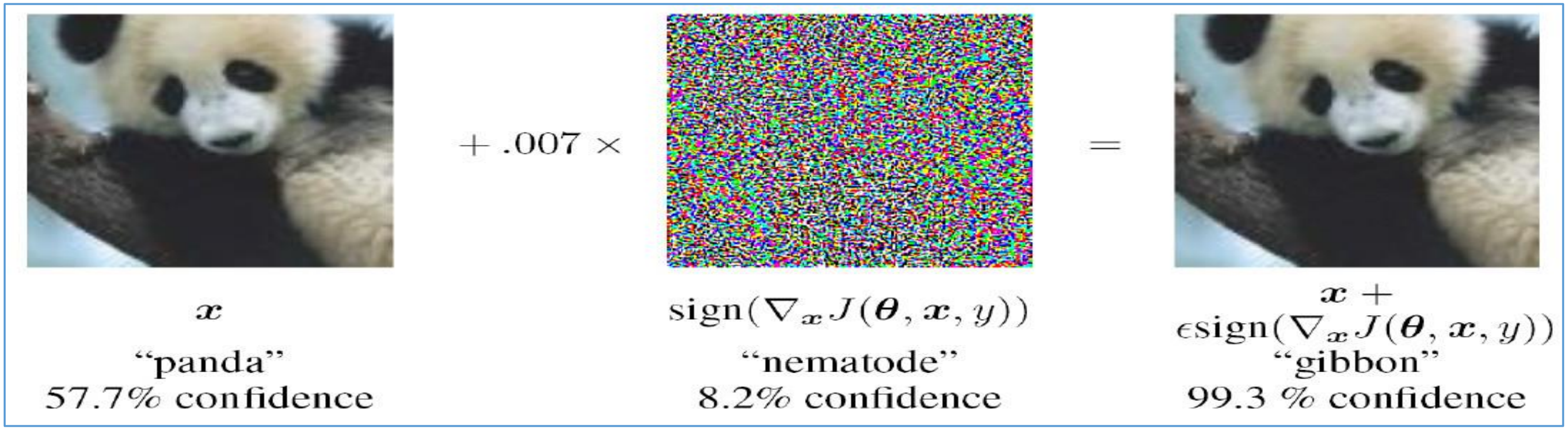
☞ Accuracy rate can be nearly over 99% If applying convolution filters.

Making adversarial samples (Approach)

- Adversarial crafting framework

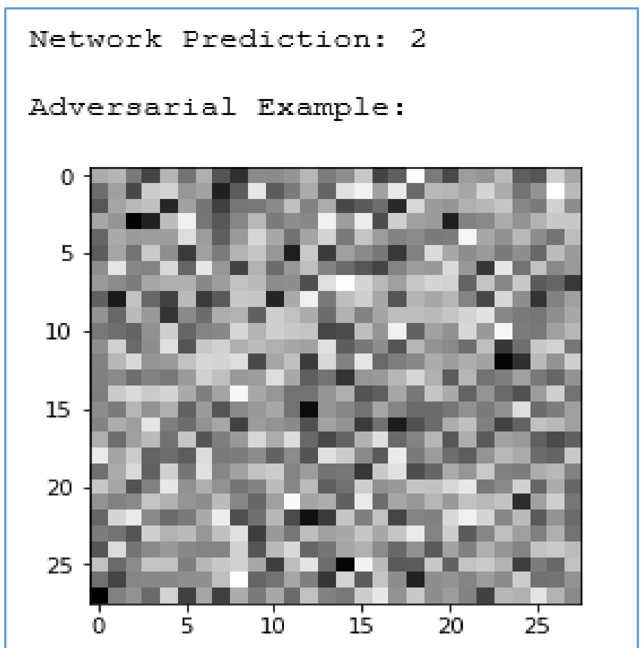


- Fast Gradient Sign Method



Making adversarial samples (Analysis)

- Non-Targeted Attack
  - Cost Function $C = \frac{1}{2}\left\| y_{goal} - \hat{y}(\vec{x}) \right\|_2^2$
  - Choosing an $\vec{x}$ input that minimizes the cost instead of weights and biases that minimize the cost.



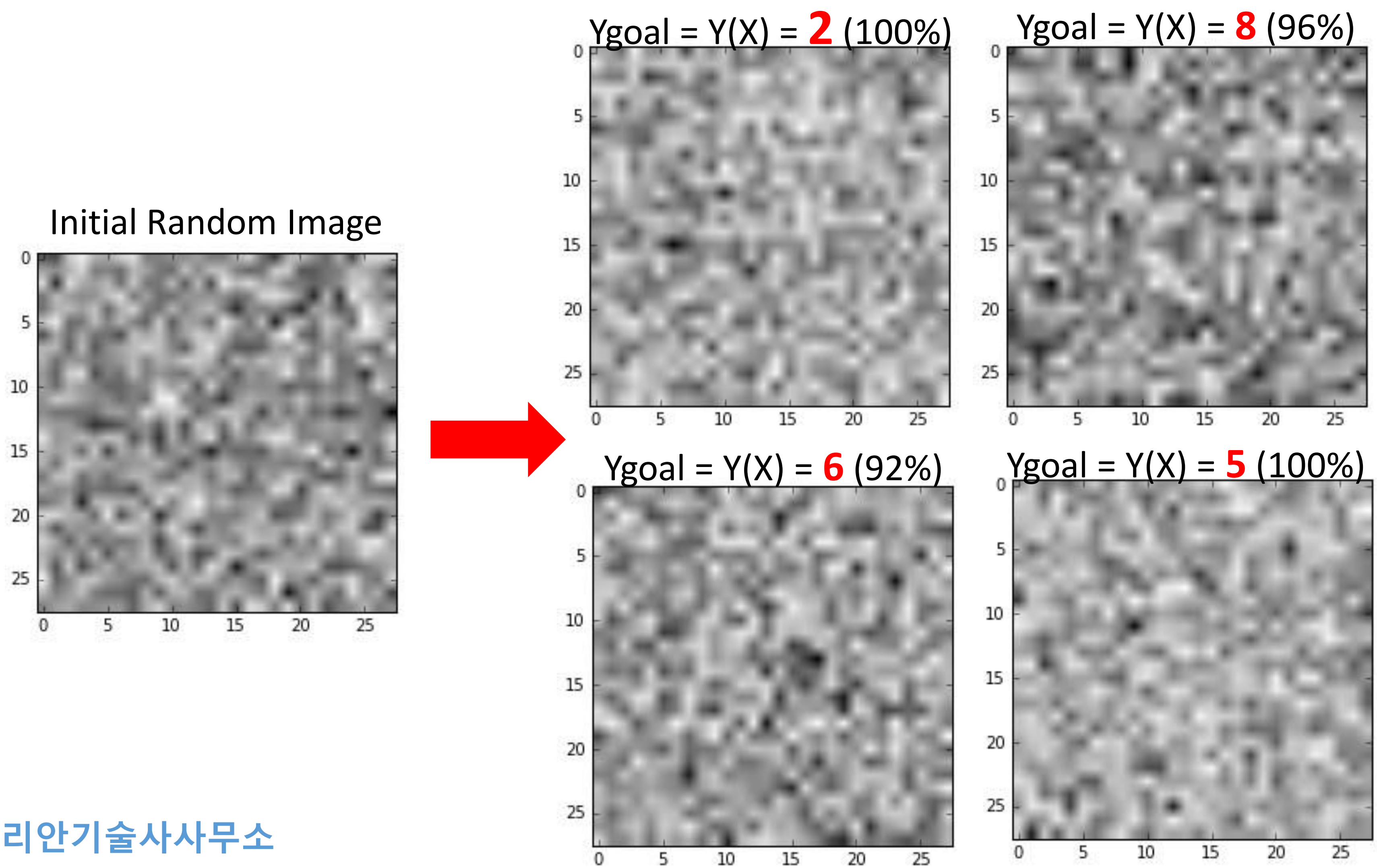
Human ☞ Perfect Noise

Deep Learning ☞ Number “2”

1) Finding the derivatives of the cost function with respect to the input, $\nabla_x C$ using backpropagation.
2) Using the gradient descent update to find the best $\vec{x}$ that minimizes the cost.

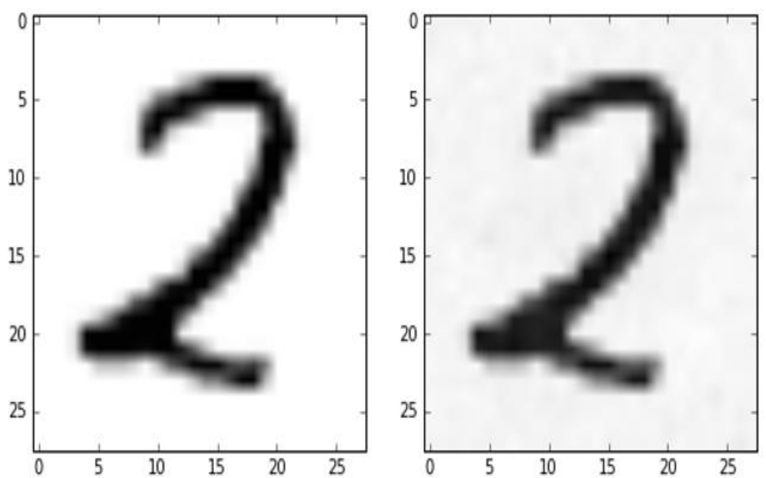
Making adversarial samples (Result)

- Experiment - Non-Targeted Attack (Step=500)



Making adversarial samples (Analysis)

- Targeted Attack
  - Cost Function $C = \frac{1}{2}\left\|y_{goal} - \hat{y}(\vec{x})\right\|_2^2 + \lambda\left\|\vec{x} - x_{target}\right\|_2^2$
  - Picture.(b) : Human ☞ number “2”
  - Picture.(b) : Deep Learning ☞ number “8” with 99.99%



(a) Actual Image (Prediction: 2) (b) Adversarial Image (Prediction: 8)

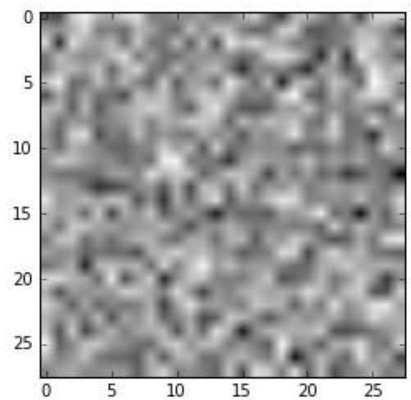
Minimizing the left term $\left\|y_{goal} - \hat{y}(\vec{x})\right\|_2^2$ will make the DNN output y_{goal} when given $\vec{x}$.

Minimizing the second term $\lambda\left\|\vec{x} - x_{target}\right\|_2^2$ will try to force our adversarial image x to be as close as possible to x_{target}.

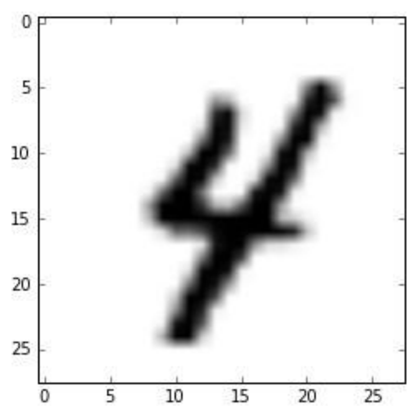
Making adversarial samples (Result)

- Experiment - Targeted Attack (Step=500, λ=0.5)

Initial Random Image

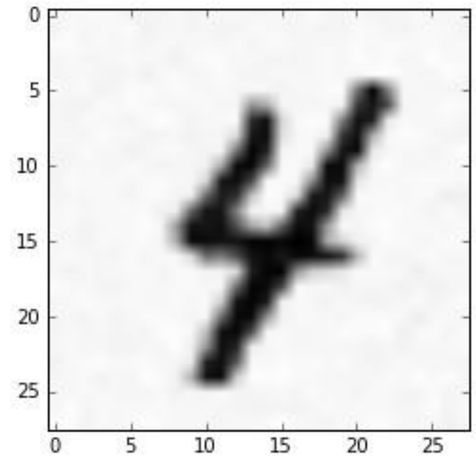


Xtarget=4

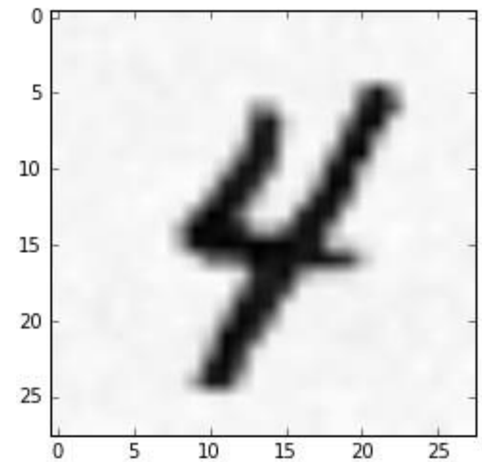


Success

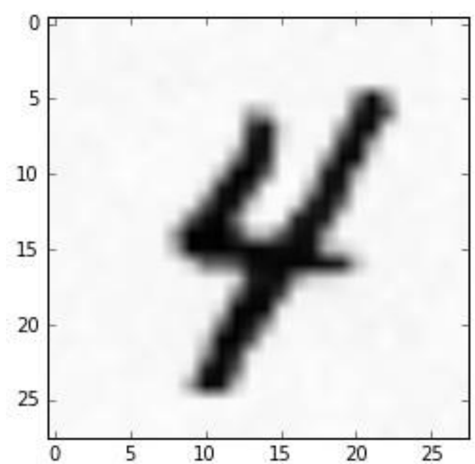
Ygoal = Y(X) = **2** (96%)



Ygoal = Y(X) = **3** (97%)



Ygoal = Y(X) = **6** (96%)



Ygoal = Y(X) = **8** (100%)