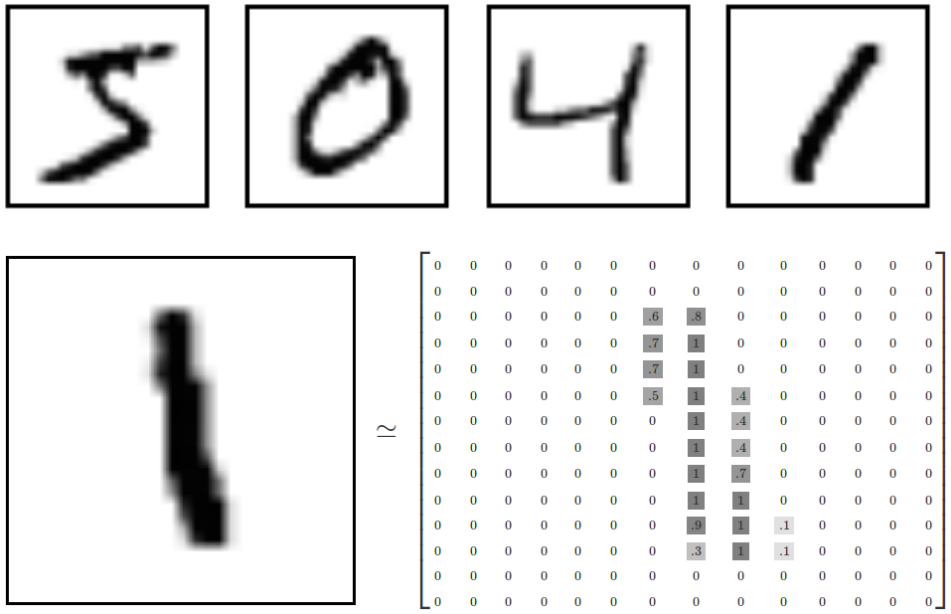
(Artificial) Neural Networks with Scikit-learn

Industrial AI Lab.

Prof. Seungchul Lee

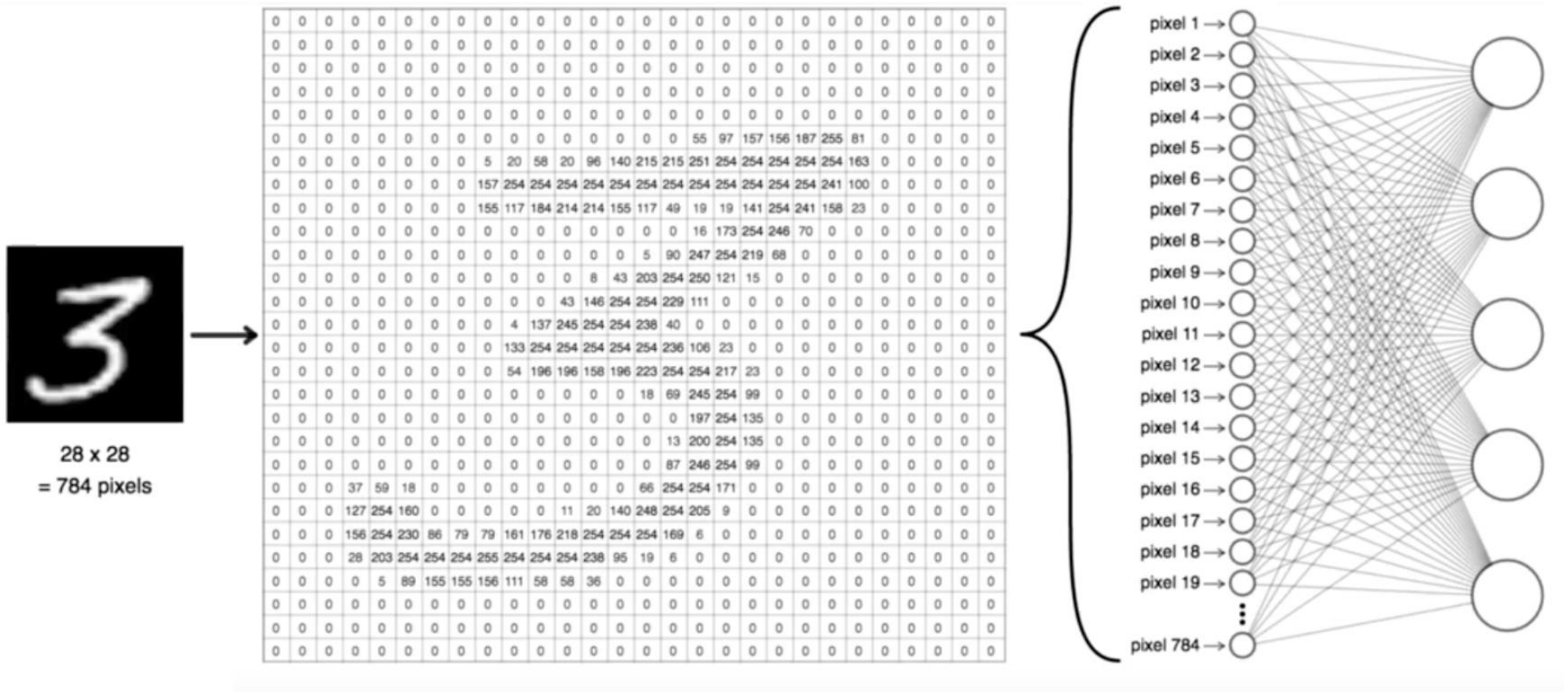
ANN with MNIST

- MNIST database
  - Mixed National Institute of Standards and Technology database
  - Handwritten digit database
  - 28×28 gray scaled image
  - Flattened matrix into a vector of 28×28=784

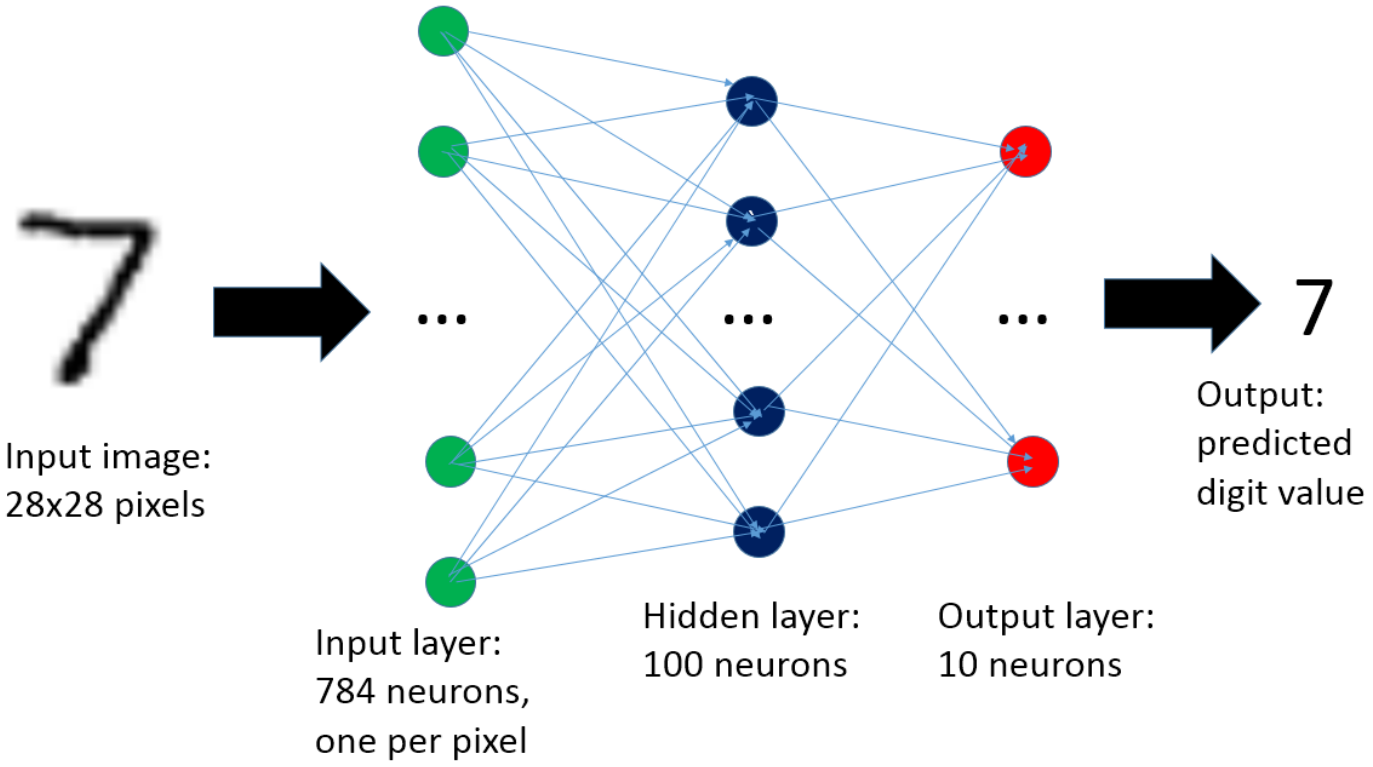


ANN

- Feed a gray image to ANN



Our Network Model

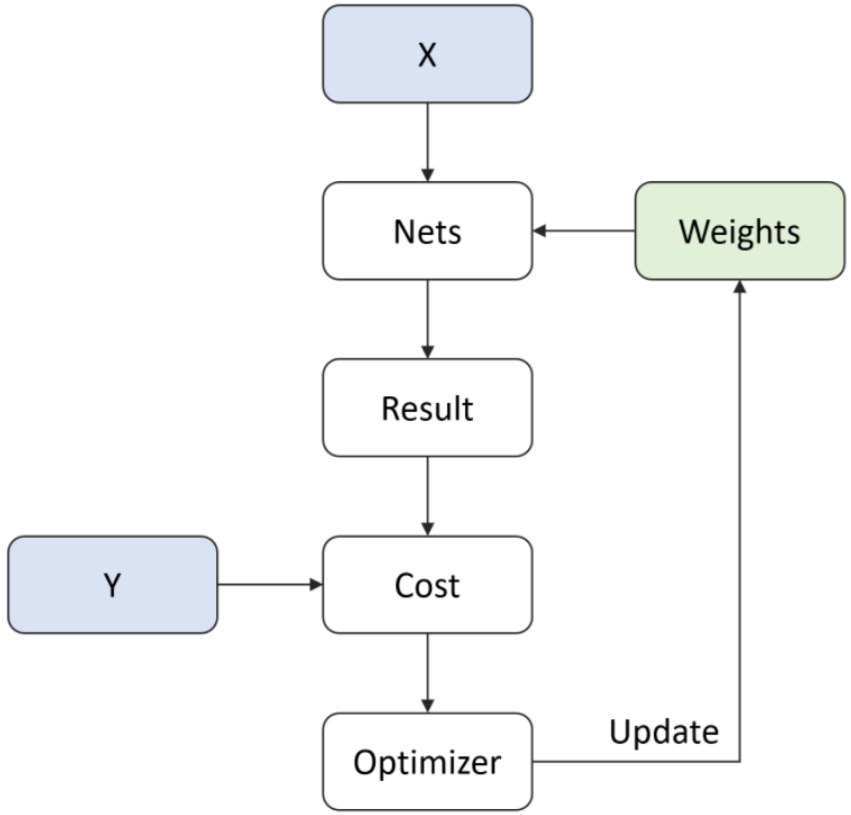


Iterative Optimization

- We will use
  - Mini-batch gradient descent
  - Adam optimizer

$$\begin{aligned} \min_{\theta} \quad & f(\theta) \\ \text{subject to} \quad & g_i(\theta) \leq 0 \end{aligned}$$

$$\theta := \theta - \alpha \nabla_\theta \left(h_\theta \left(x^{(i)} \right), y^{(i)} \right)$$



ANN with Scikit-learn

- Import Library

```
# Import Library
import numpy as np
import matplotlib.pyplot as plt

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

- Load MNIST Data
  - Download MNIST data

```
train_x = np.load('./data_files/mnist_train_images.npy')
train_y = np.load('./data_files/mnist_train_labels.npy')
test_x = np.load('./data_files/mnist_test_images.npy')
test_y = np.load('./data_files/mnist_test_labels.npy')
```

One Hot Encoding

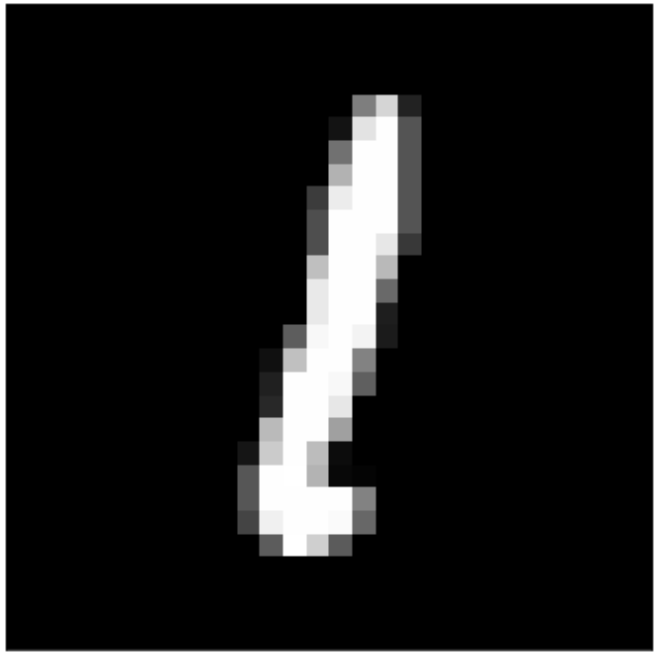
- One hot encoding

```
mnist_train_labels[7]
```

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

```
np.argmax(mnist_train_labels[7])
```

```
0
```

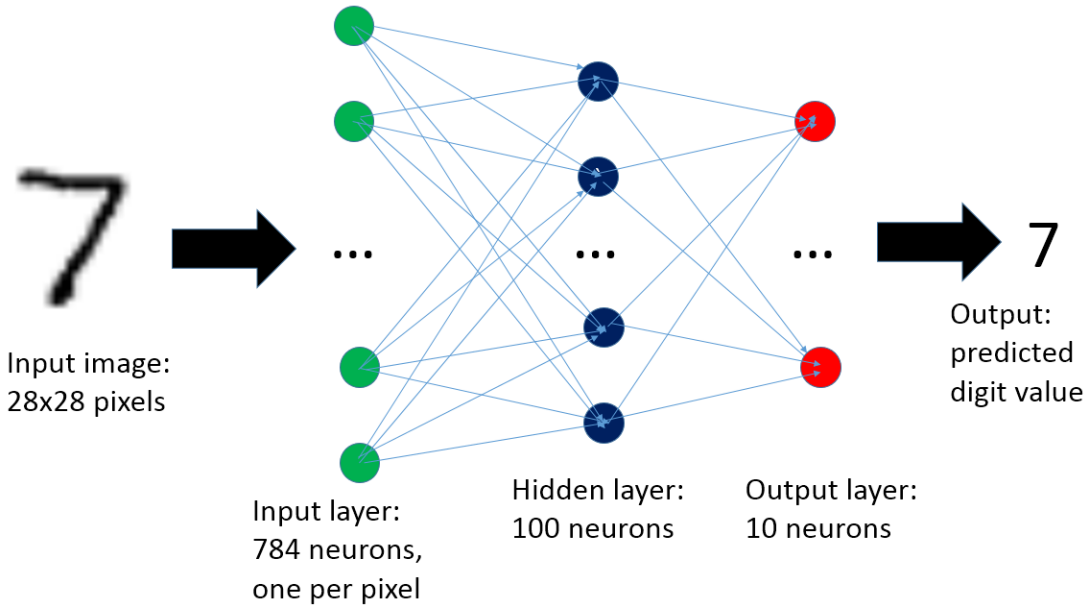


Training

```
clf = MLPClassifier(hidden_layer_sizes = (100,),
                    activation = 'logistic',
                    solver = 'sgd',
                    learning_rate_init = 0.0001,
                    batch_size = 50,
                    max_iter = 100,
                    verbose = True)
```

```
clf.fit(train_x, train_y)
```

```
Iteration 1, loss = 3.57824966
Iteration 2, loss = 3.18791200
Iteration 3, loss = 3.13809828
Iteration 4, loss = 3.08519212
Iteration 5, loss = 3.02767935
Iteration 6, loss = 2.96478203
Iteration 7, loss = 2.89614439
Iteration 8, loss = 2.82202289
Iteration 9, loss = 2.74309025
Iteration 10, loss = 2.66058488
```



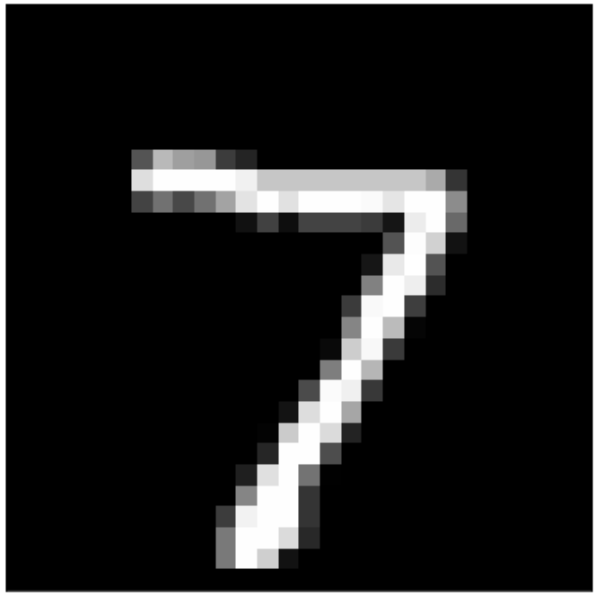
Test or Evaluation

```
pred = clf.predict(test_x)
print("Accuracy : {}%".format(accuracy_score(test_y, pred)*100))
```

```
Accuracy : 96.0%
```

```
logits = clf.predict_proba(test_x[:1])
predict = clf.predict(test_x[:1])

plt.figure(figsize = (6,6))
plt.imshow(test_x[:1].reshape(28,28), 'gray')
plt.xticks([])
plt.yticks([])
plt.show()
```



```
Prediction : 7
Probability : [ 0.02  0.    0.01  0.03  0.01  0.02  0.    0.93  0.01  0.12]
```