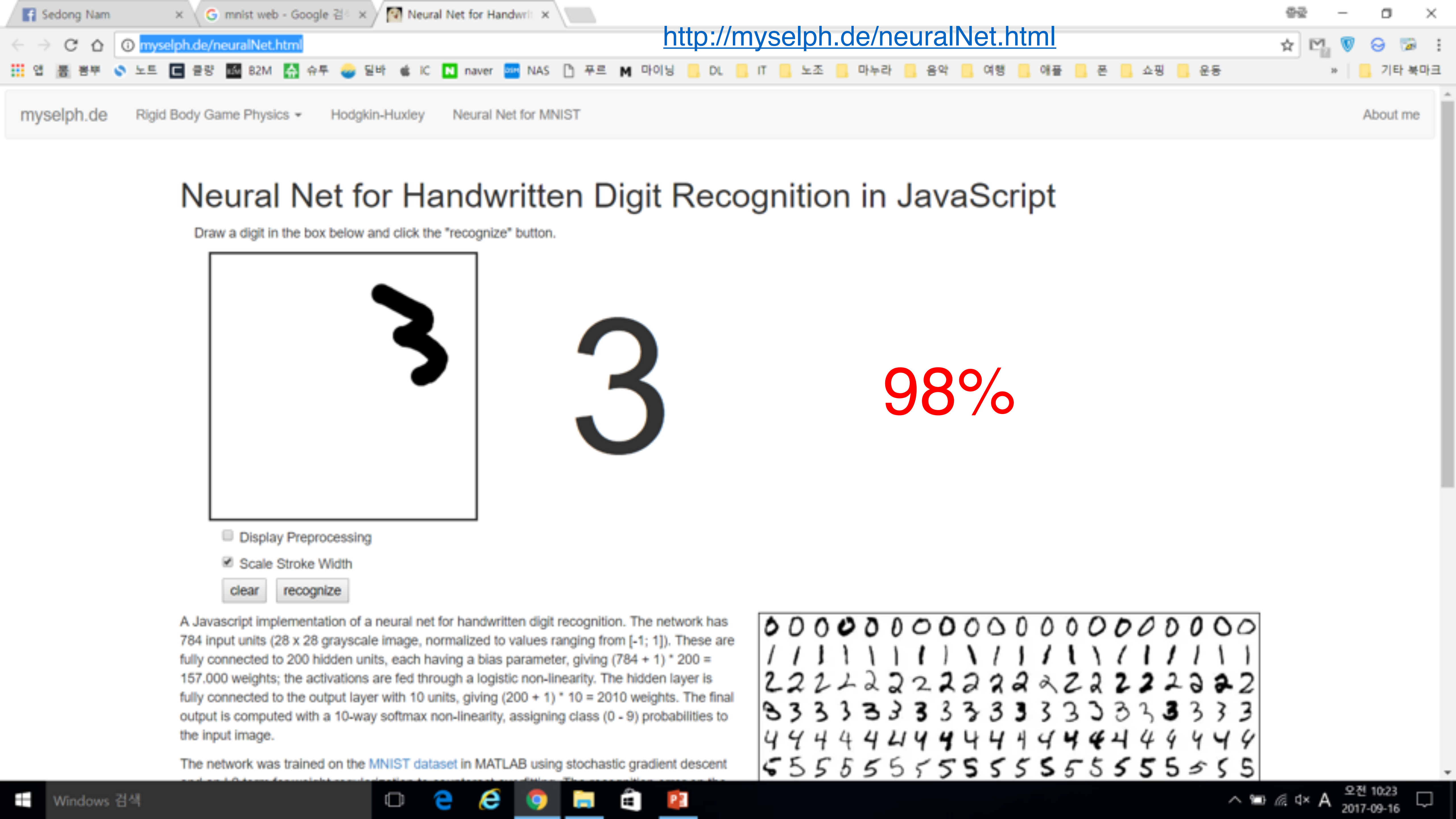
초심자를 위한 숫자 인식 99% 인공지능 만들기

유 용 균


http://myselph.de/neuralNet.html
myselph.de
Rigid Body Game Physics
Hodgkin-Huxley
Neural Net for MNIST
About me
Neural Net for Handwritten Digit Recognition in JavaScript
Draw a digit in the box below and click the "recognize" button.
3
98%
Display Preprocessing
Scale Stroke Width
clear
recognize
A Javascript implementation of a neural net for handwritten digit recognition. The network has 784 input units (28 x 28 grayscale image, normalized to values ranging from [-1; 1]). These are fully connected to 200 hidden units, each having a bias parameter, giving (784 + 1) * 200 = 157.000 weights; the activations are fed through a logistic non-linearity. The hidden layer is fully connected to the output layer with 10 units, giving (200 + 1) * 10 = 2010 weights. The final output is computed with a 10-way softmax non-linearity, assigning class (0 - 9) probabilities to the input image.
The network was trained on the MNIST dataset in MATLAB using stochastic gradient descent
2017-09-16

MNIST 데이터베이스

위키백과, 우리 모두의 백과사전.

MNIST 데이터베이스 (Modified National Institute of Standards and Technology database)는 손으로 쓴 숫자들로 이루어진 대형 데이터베이스이며, 다양한 화상 처리 시스템을 트레이닝하기 위해 일반적으로 사용된다.[1][2] 이 데이터베이스는 또한 기계 학습 분야의 트레이닝 및 테스트에 널리 사용된다.[3][4] NIST의 오리지널 데이터셋의 샘플을 재혼합하여 만들어졌다. 개발자들은 NIST의 트레이닝 데이터셋이 미국의 인구조사국 직원들로부터 취합한 이후로 테스팅 데이터셋이 미국의 중등학교 학생들로부터 취합되는 중에 기계 학습 실험에 딱 적합하지는 않은 것을 느꼈다.[5] 게다가 NIST의 흑백 그림들은 28x28 픽셀의 바운딩 박스와 앤티엘리어싱 처리되어 그레이스케일 레벨이 들어가 있도록 평준화되었다.[5]

MNIST 데이터베이스는 60,000개의 트레이닝 이미지와 10,000개의 테스트 이미지를 포함한다.[6] 트레이닝 세트의 절반과 테스트 세트의 절반은 NIST의 트레이닝 데이터셋에서 취합하였으며, 그 밖의 트레이닝 세트의 절반과 테스트 세트의 절반은 NIST의 테스트 데이터셋으로부터 취합되었다.[7]

MINST 테스트 데이터셋의 샘플 이미지.

목차 [숨기기]

코드 먼저 돌려봐요

- https://github.com/yoyogo96

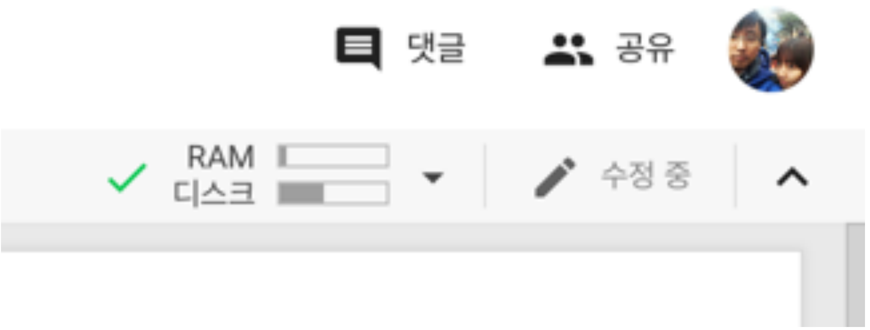
- 웹에서 손글씨 인식
- http://myselph.de/neuralNet.html

- Fully connected Neural Network
- https://colab.research.google.com/drive/1Py8Eme5lPx3yZ7LvgKbT-3WmGmDSW4CS
- https://colab.research.google.com/drive/1iA_Mkt2aSxC_tJU6RrStuugTLsBet_i4

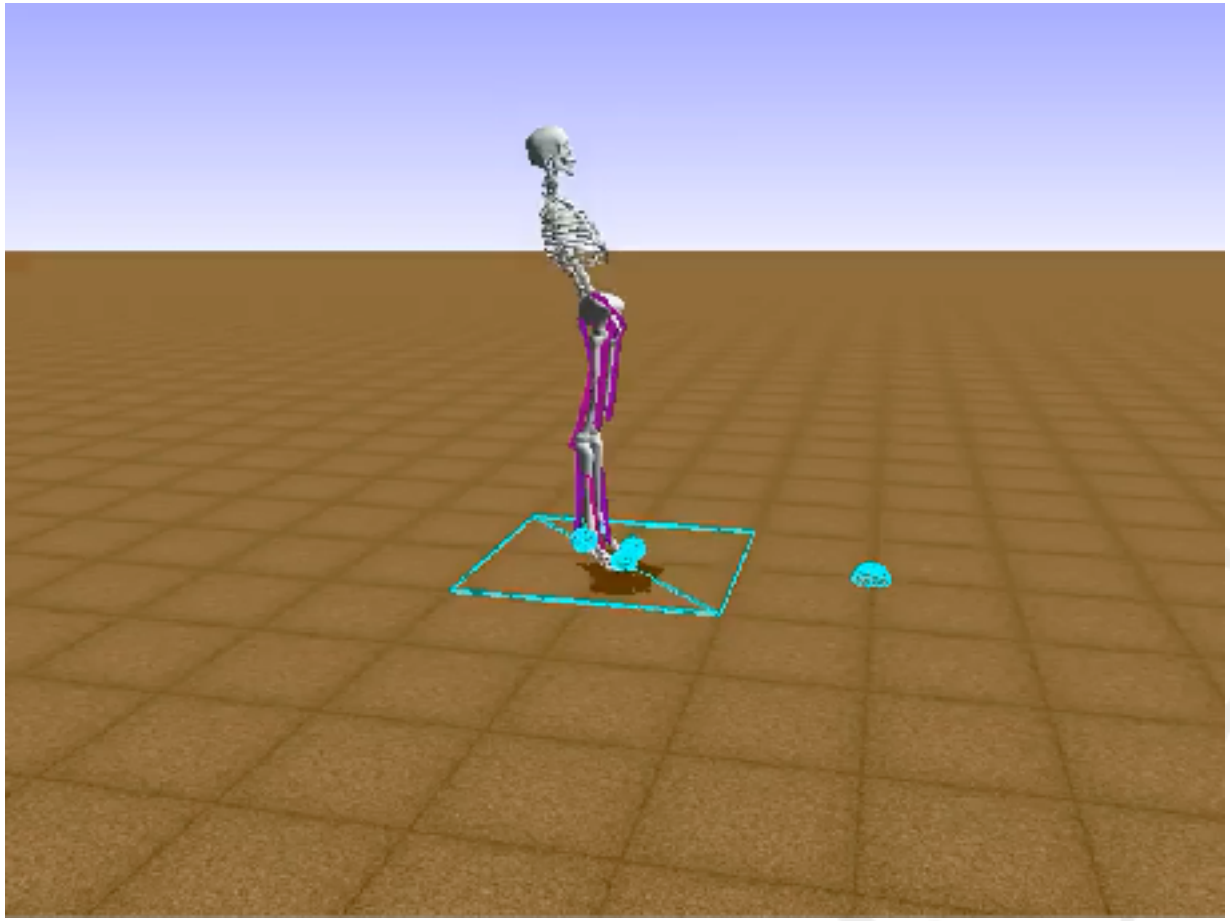
- Convolutional Neural Network
- https://colab.research.google.com/drive/1nWoT-jVuEs1AJBHAGKa5V4u0noLA0OrW

코드 먼저 돌려봐요

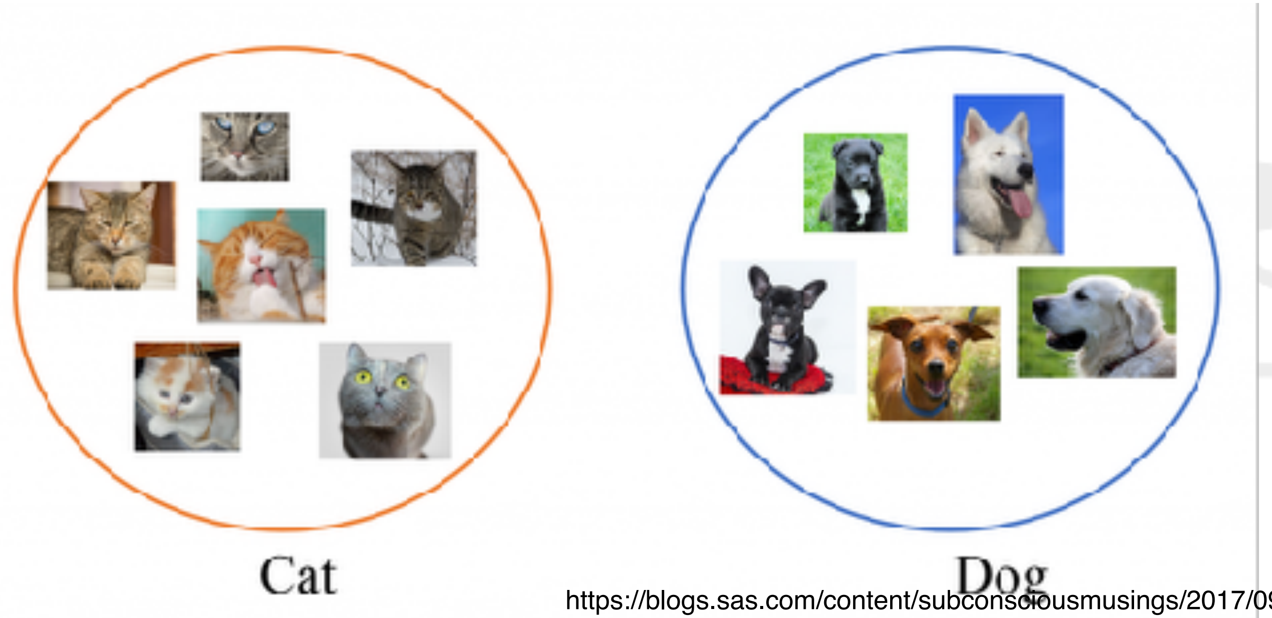
- 파일 > 드라이브에 사본저장
- 연결
- 런타임 > 런타임 유형변경
  > None (GPU off)



기계 (컴퓨터)도 인간이 배우는 방법을 모방해 보자!

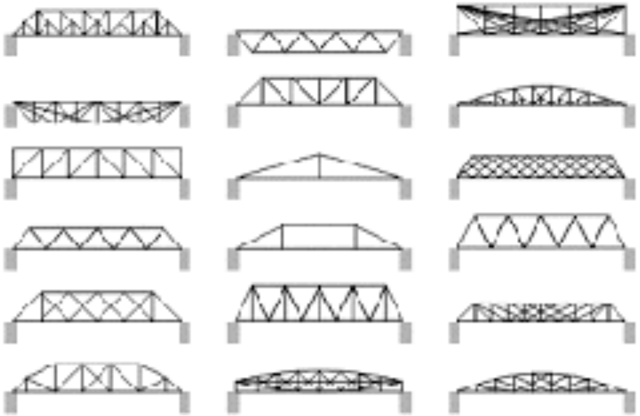


사람은 개 고양이를 구분하는 방법을 어떻게 배우는 가?



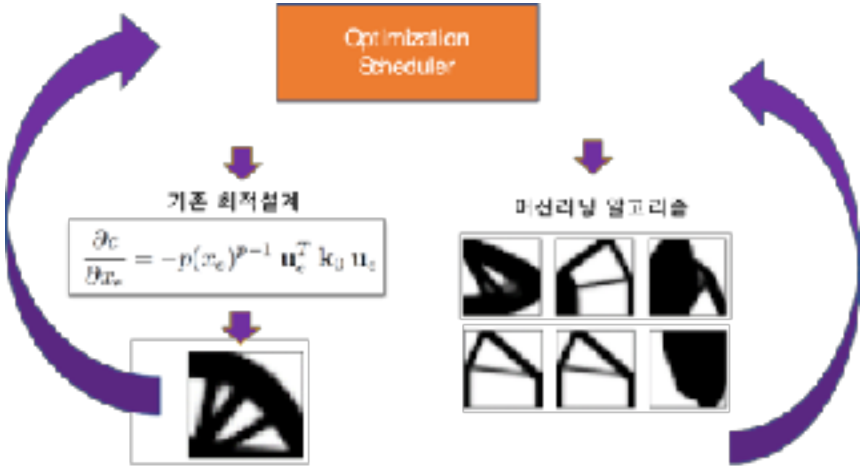
https://blogs.sas.com/content/subconsciousmusings/2017/09/25/machine-learning-concepts-styles-machine-learning/

기존 설계로부터 설계의 원리를 배울 수 있지 않을까?



인공지능을 활용한 최적설계

최적설계의 효율화



- 머신러닝 기술을 적용한 기존 최적설계 및 수치해석 방법론의 효율성 향상
- 설계자를 위한 빠른 해석 툴

감성의 메타모델

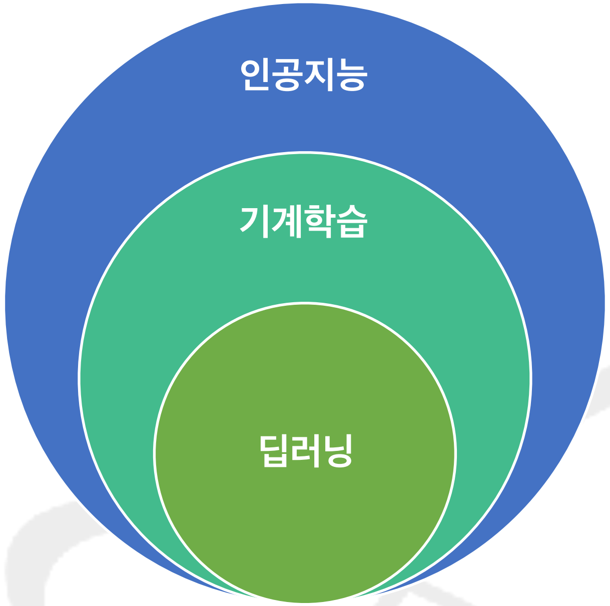
- 기존에 공학적으로 정의하기 힘들었던 것 (개인의 취향, 제작성, 심미성)을 고려한 최적설계

기계학습(Machine Learning)

특정한 과제에 대해서

경험을 통해

성능을 향상시키는 것



경험을 통해 데이터를 모아서
패턴을 분석해서 성능을 향상시키는 것

Quiz

X = 1, Y = 2
X = 2 , Y = 4
X = 3 , Y = ???

X = 1 , Y = 1
X = 2 , Y = 1
X = 3, Y = 1
X= -1, Y = 0
X= -2, Y = 0
X= -3, Y = ??

여러분은 기계.. 아니 인간 학습을 하셨습니다.

X = 1, Y = 2
X = 2 , Y = 4

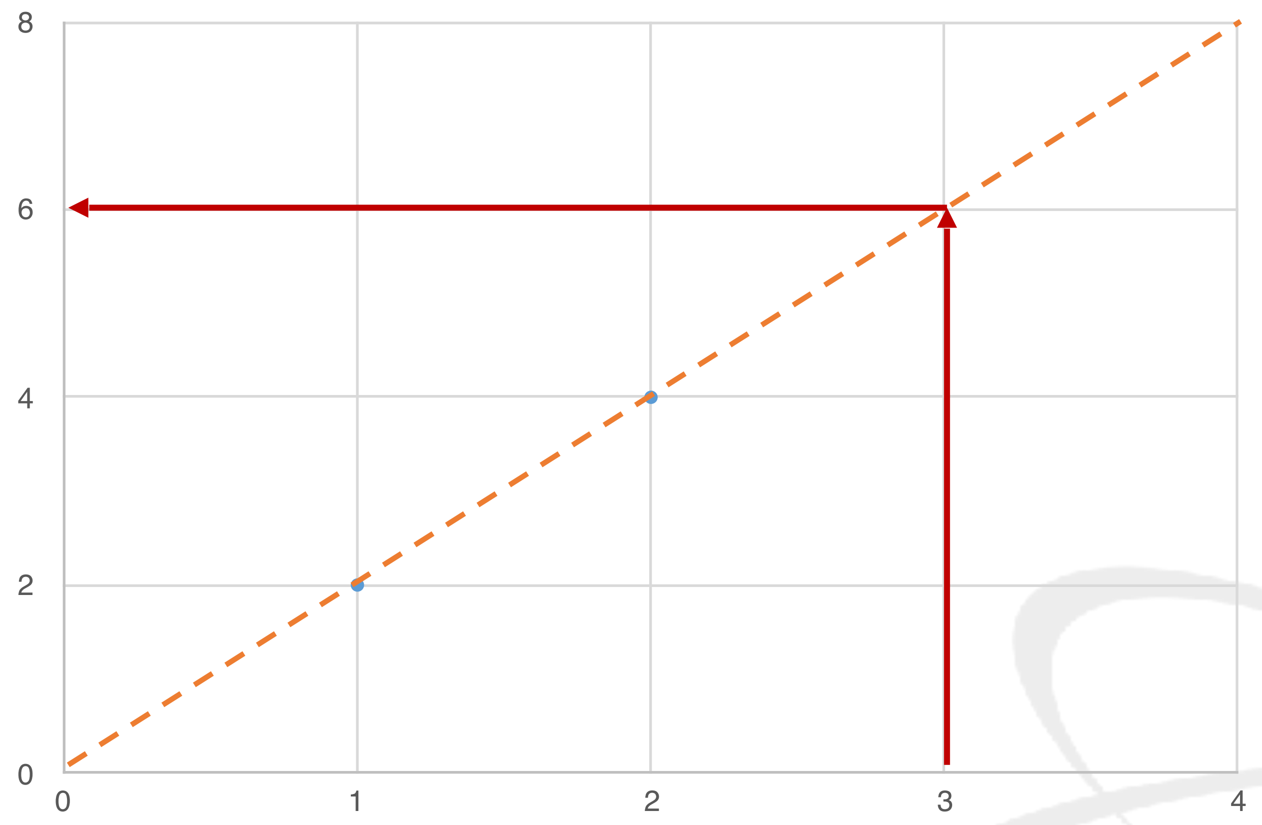
X = 3 , Y = ???

경험을 통해

성능을 향상
(미래를 예측)

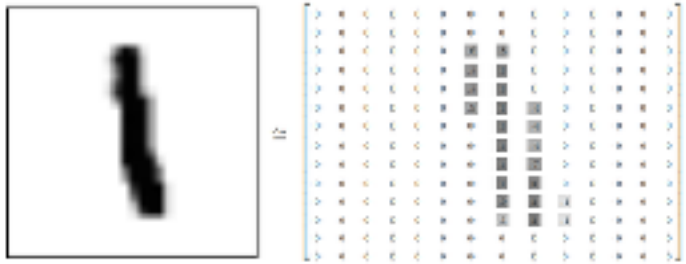
경험을 통해 데이터를 모아서
패턴을 분석해서 성능을 향상시키는 것

컴퓨터는 어떻게 배울까요?



패턴을 찾는다 = 선을 긋는다 = 함수를 찾는다.

손글씨 인식



1

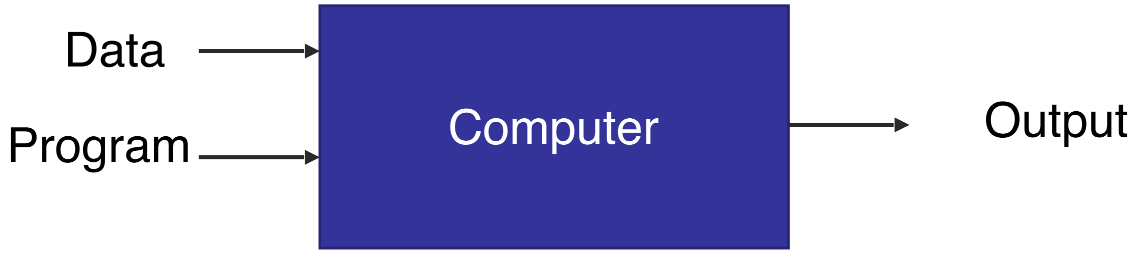
패턴(함수)를 찾는다.

수학 + CS(Computer Science)

수학은 대학교 2학년이면 85%는 이해할 수 있다고..
CS쪽은 라이브러리가 너무 잘 되어 있어서..

Machine Learning

Traditional Programming



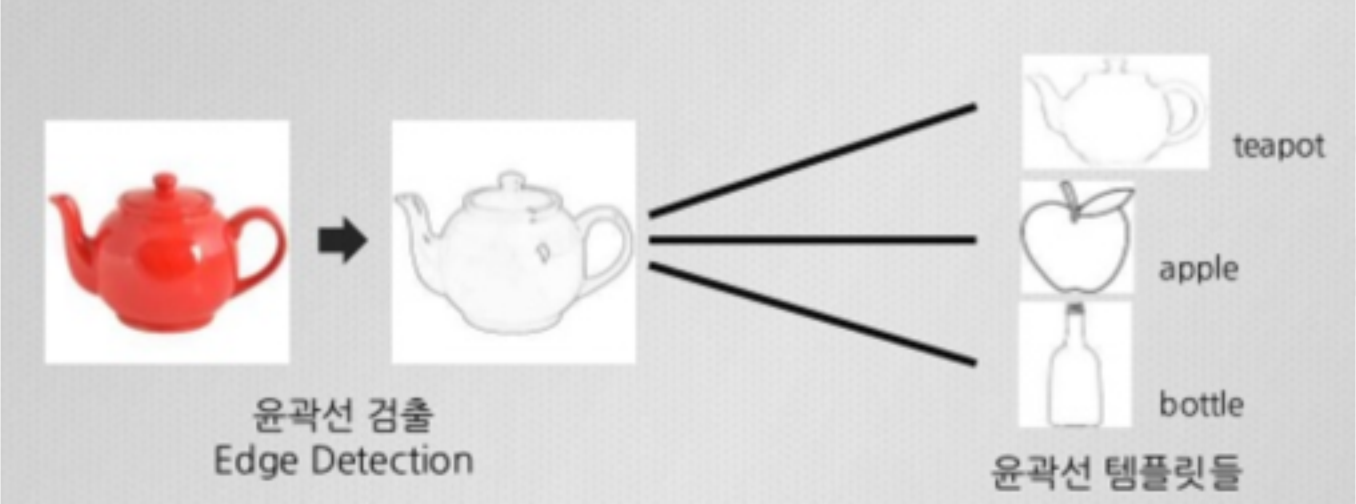
Machine Learning

남세동님 자료에서?

Machine Learning

Traditional Programming

Data → Computer → Output
Program → Computer



https://www.slideshare.net/yonghakim900/ss-60252533

Machine Learning

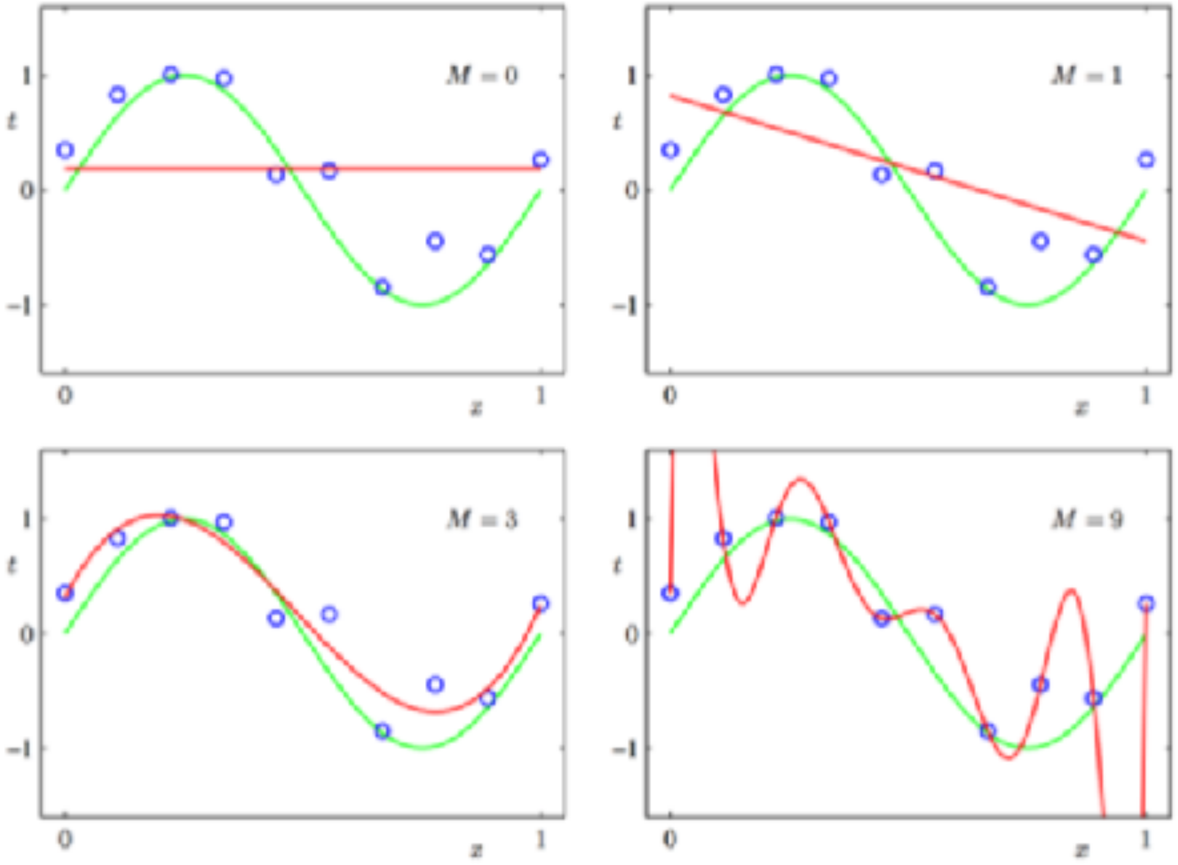
Machine Learning

→ CAT

고전적으로 컴퓨터가 고전했던
고도의 인식 문제를
컴퓨터가 계산할 수 있는 계산문제로 치환
>> CS + 통계학

Data →
Output → **Computer** → Program

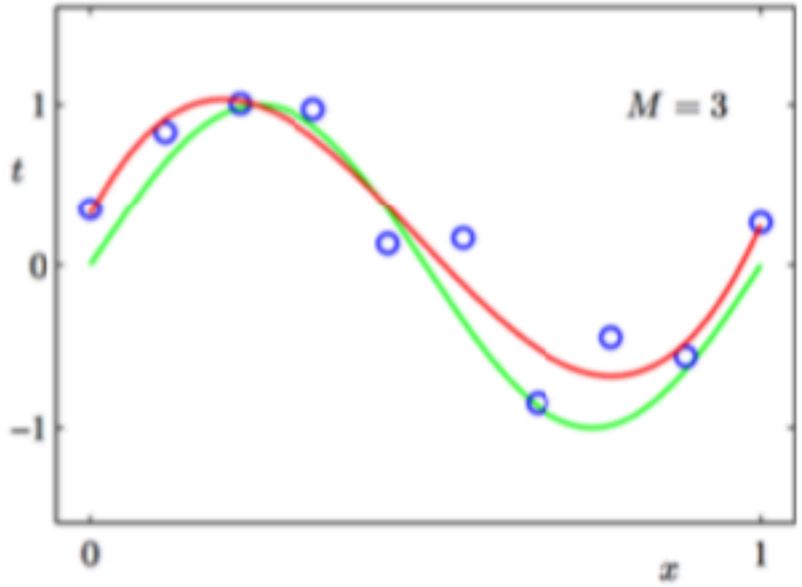
무엇이 최선일까요?



(Bishop)

https://www.sciencedirect.com/science/article/pii/S0895717713001775

커브 피팅을 하려면 무엇을 결정해야 하는가?



어떤 함수로 Fitting 할 것인가?
- 한꺼번에 다항식으로?
- 군데군데 잘라서? (spline?)

다항식의 경우몇차로 Fitting 할 것인가?

오차는 어떻게 정의할 것인가?

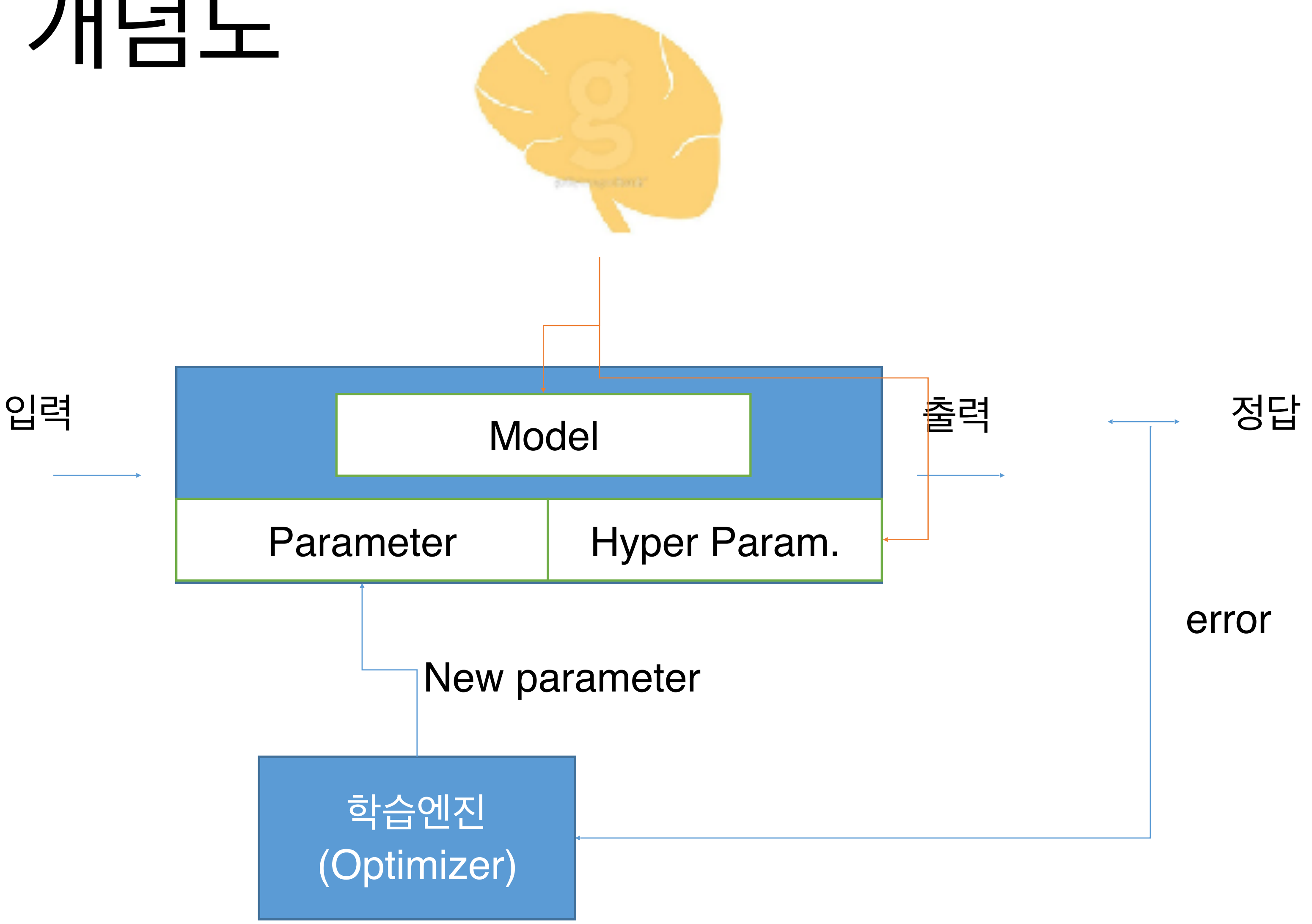
오차는 어떻게 줄일 것인가?

F(x)

$$f(x) = e^{ax} + b???$$

a, b, c, d = ??

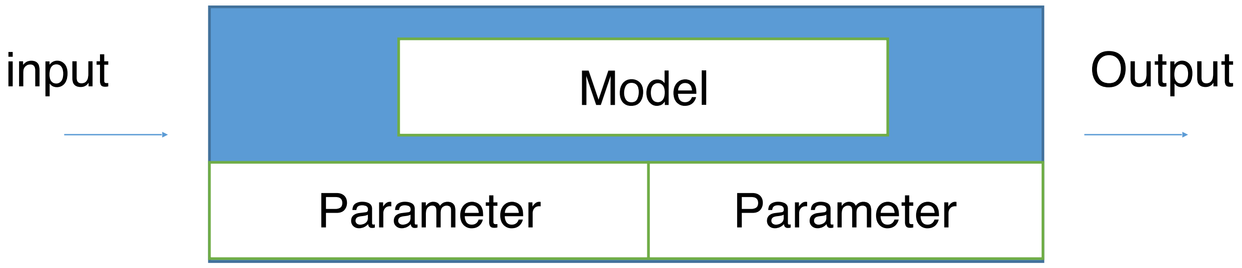
머신러닝 개념도



입력
Model
출력
정답
Parameter
Hyper Param.
error
New parameter
학습엔진
(Optimizer)

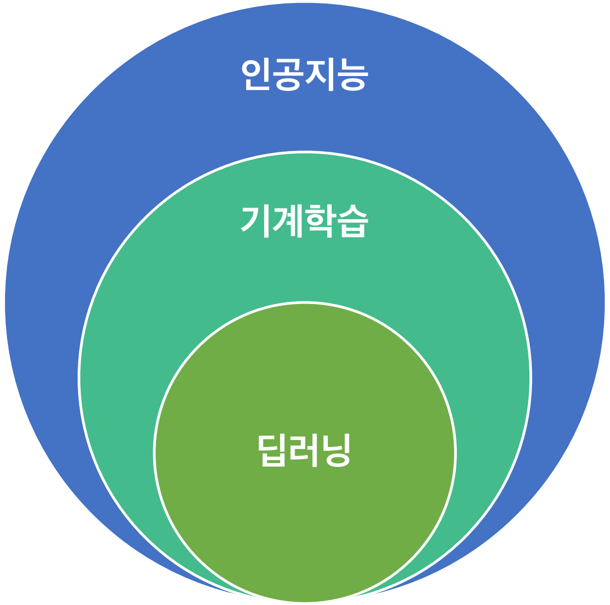
머신러닝의 요소

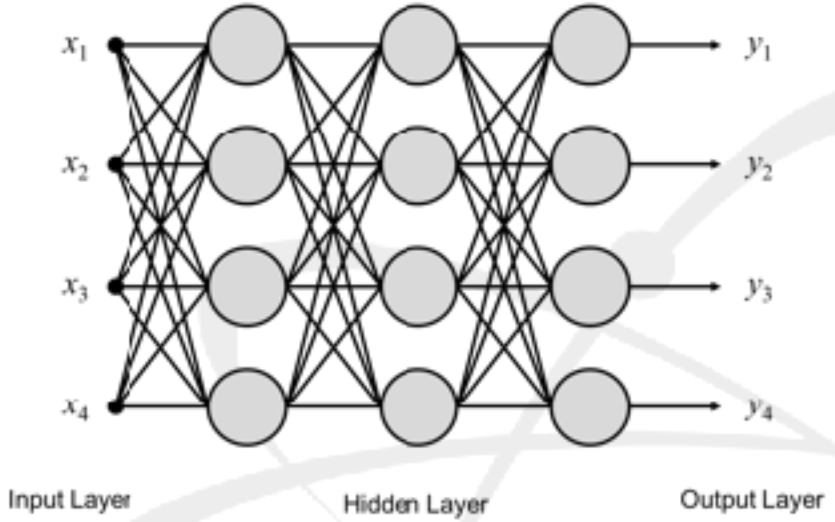
- 데이터
  - 데이터를 어떻게 습득할 것인가?
  - 데이터에 Lable은 존재하는 가?
  - Feature Engineering..
- 데이터를 Fitting 할 방법
  - SVM, Random forest, Neural network ……
- 변수
  - Parameter
  - Hyperparameter (학습을 하지 않고 사용자가 결정하는 튜닝 파라미터)
- 오차 정의 (Loss function)
- 학습 방법 (최적화 방법)



Deep Learning (Deep Neural Network)

$$Y = W(X)$$

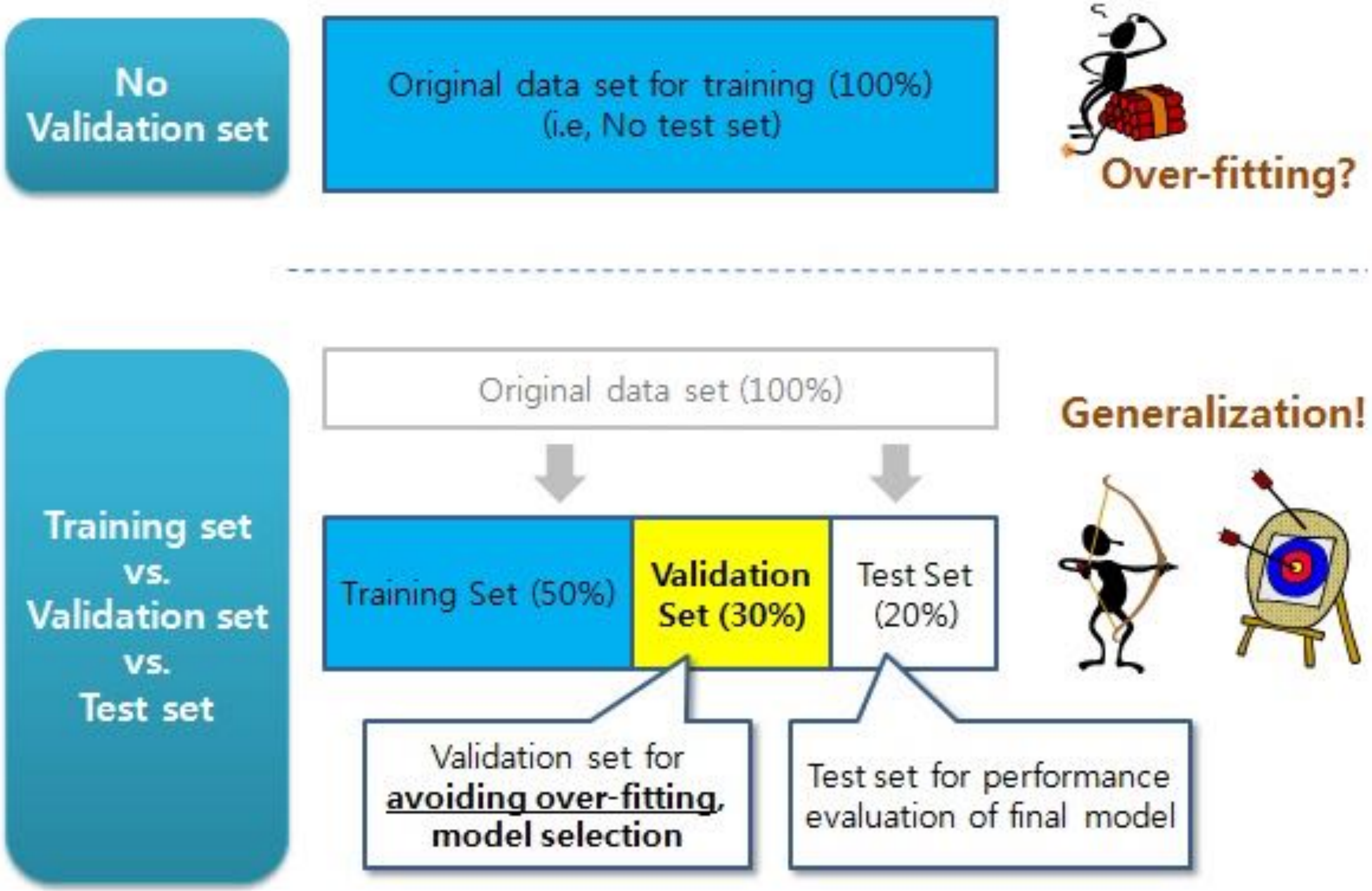




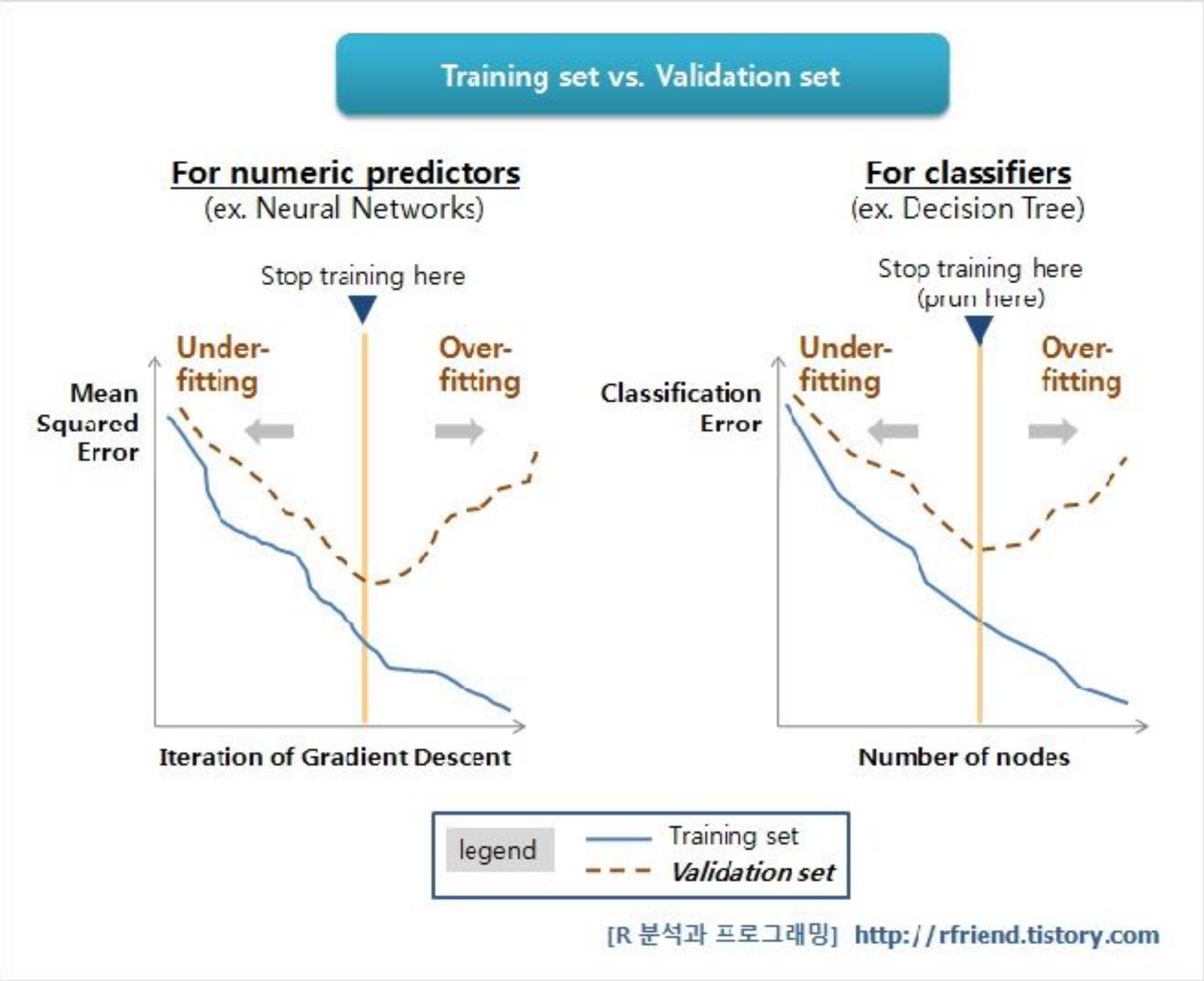
용어 정리

- Epoch
- Batch

- Train set
- Validation set
- Test set

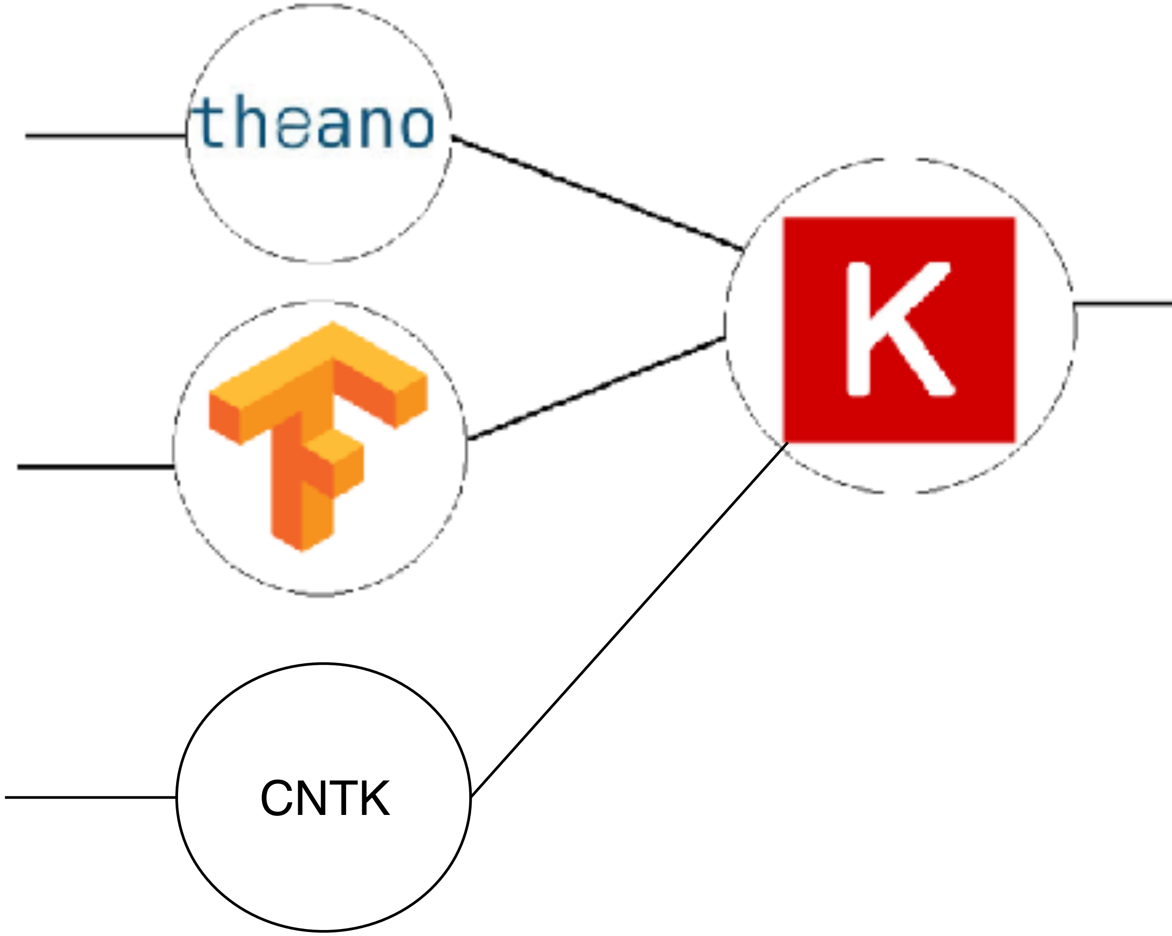


[R 분석과 프로그래밍] http://rfriend.tistory.com


Training set vs. Validation set
For numeric predictors
(ex. Neural Networks)
Stop training here
Under-fitting
Over-fitting
Mean Squared Error
Iteration of Gradient Descent
For classifiers
(ex. Decision Tree)
Stop training here
(prun here)
Under-fitting
Over-fitting
Classification Error
Number of nodes
legend
Training set
Validation set
[R 분석과 프로그래밍] http://rfriend.tistory.com

KERAS (Keras.io)

- Google 엔지니어 Francois Chollet 창시
- Torch에서 영감을 얻은 직관적 API
- Theano, TensorFlow, CNTK 백엔드
- 빠르게 성장하고 있는 프레임워크
- 신경망의 표준 Python API이 될 가능성이 큼



- Why Keras?
    - 짧다. 진입장벽이 낮다
    - 추상화가 잘되어 있어 코드 가독성이 높다
    - Keras를 이해하면 다른 API도 쉽게 활용 가능.
    - Theano, TensorFlow, CNTK를 골라서 사용가능(Backend)

Tensorflow vs. Keras Code

```
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# dropout (keep_prob) rate  0.7 on training, but should be 1 for testing
keep_prob = tf.placeholder(tf.float32)

# weights & bias for nn layers
# http://stackoverflow.com/questions/33640581/how-to-do-xavier-initializa
W1 = tf.get_variable("W1", shape=[784, 512],
                     initializer=tf.contrib.layers.xavier_initializer())
b1 = tf.Variable(tf.random_normal([512]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)

W2 = tf.get_variable("W2", shape=[512, 512],
                     initializer=tf.contrib.layers.xavier_initializer())
b2 = tf.Variable(tf.random_normal([512]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)
L2 = tf.nn.dropout(L2, keep_prob=keep_prob)

W3 = tf.get_variable("W3", shape=[512, 512],
                     initializer=tf.contrib.layers.xavier_initializer())
b3 = tf.Variable(tf.random_normal([512]))
L3 = tf.nn.relu(tf.matmul(L2, W3) + b3)
L3 = tf.nn.dropout(L3, keep_prob=keep_prob)

W4 = tf.get_variable("W4", shape=[512, 512],
                     initializer=tf.contrib.layers.xavier_initializer())
b4 = tf.Variable(tf.random_normal([512]))
L4 = tf.nn.relu(tf.matmul(L3, W4) + b4)
L4 = tf.nn.dropout(L4, keep_prob=keep_prob)

W5 = tf.get_variable("W5", shape=[512, 10],
                     initializer=tf.contrib.layers.xavier_initializer())
```

```
                     initializer=tf.contrib.layers.xavier_initializer())
b5 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L4, W5) + b5

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys, keep_prob: 0.7}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')

# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Accuracy:', sess.run(accuracy, feed_dict={
      X: mnist.test.images, Y: mnist.test.labels, keep_prob: 1}))
```

```
model = Sequential()

model.add(Dense(256, input_dim=784,
                kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(256, kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(256, kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(256, kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_split=0.2)
```

Tensorflow Code

Keras Code

Tensorflow vs. Keras Code (2)

```
W1 = tf.get_variable("W1", shape=[784, 512],
                     initializer=tf.contrib.layers.xavier_initializer())
b1 = tf.Variable(tf.random_normal([512]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)
```

```
model.add(Dense(256, input_dim=784,
          kernel_initializer='glorot_uniform', activation='relu'))
model.add(Dropout(0.3))
```

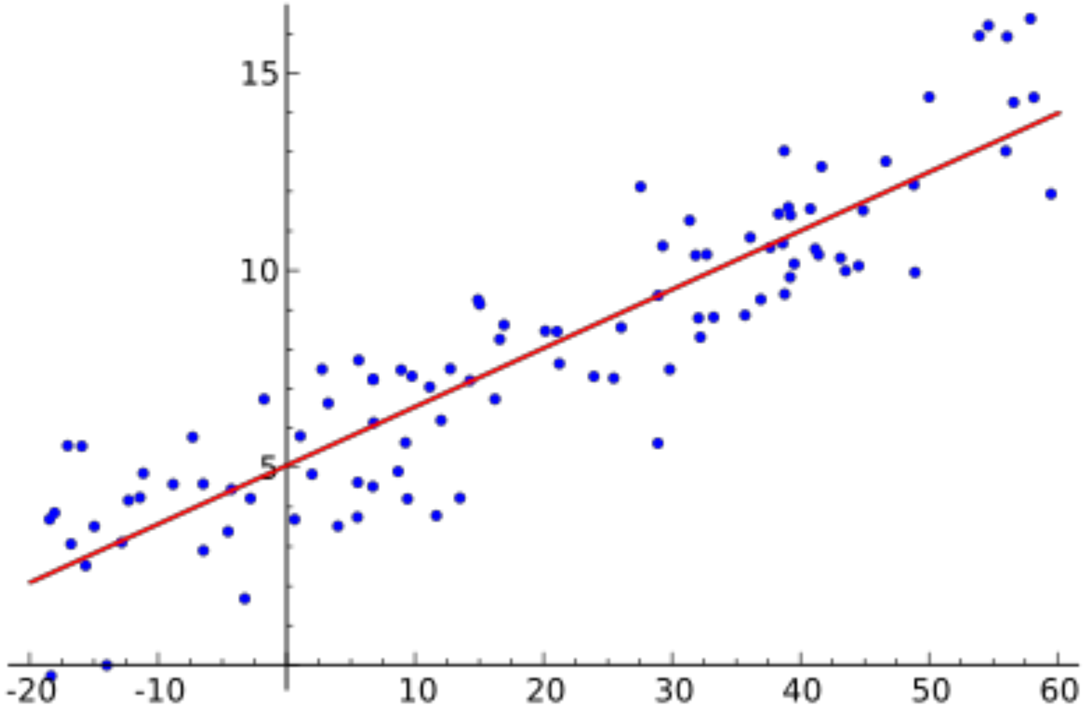
```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys, keep_prob: 0.7}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))
```

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=[illegible],
                    validation_split=[illegible])
```

Tensorflow Code

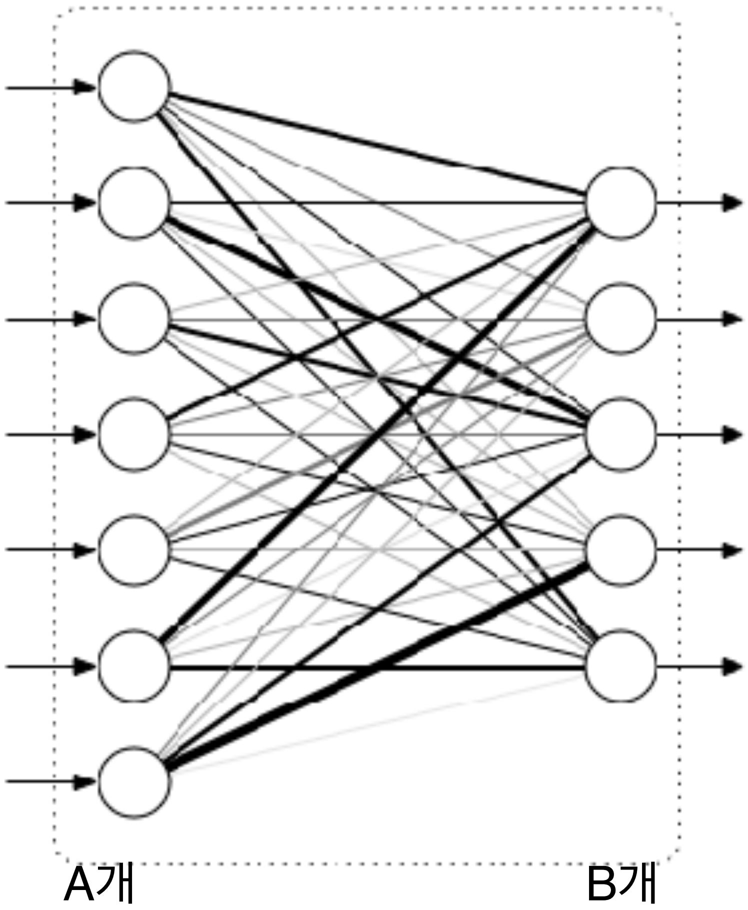
Keras Code

실습 0. Linear Regression




```
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]

model = Sequential()
model.add(Dense(1, input_dim=1))

sgd = optimizers.SGD(lr=0.1)
model.compile(loss='mse', optimizer=sgd)

# prints summary of the model to the terminal
model.summary()

model.fit(x_train, y_train, epochs=200)

y_predict = model.predict(np.array([5]))
print(y_predict)
```

Fully Connected Network



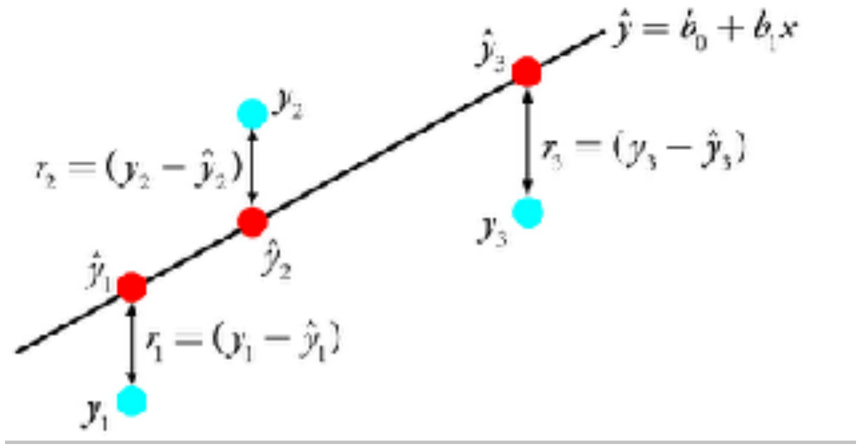
Y’=AX + B

전체 Connection 수:

A X B

>> DENSE(A,B)

모델 컴파일




```
sgd = optimizers.SGD(lr=0.1)
model.compile(loss='mse', optimizer=sgd)
```

$$J(\theta) = \frac{1}{2}\sum_{i=0}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Loss를 계산하는 방법
(Mean Square Error)

Optimizer 종류
(Stochastic Gradient decent)

Loss = L(Y’, Y)

모델 학습 & 평가

```
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]
```

```
model.fit(x_train, y_train, epochs=200)
```

모델 학습

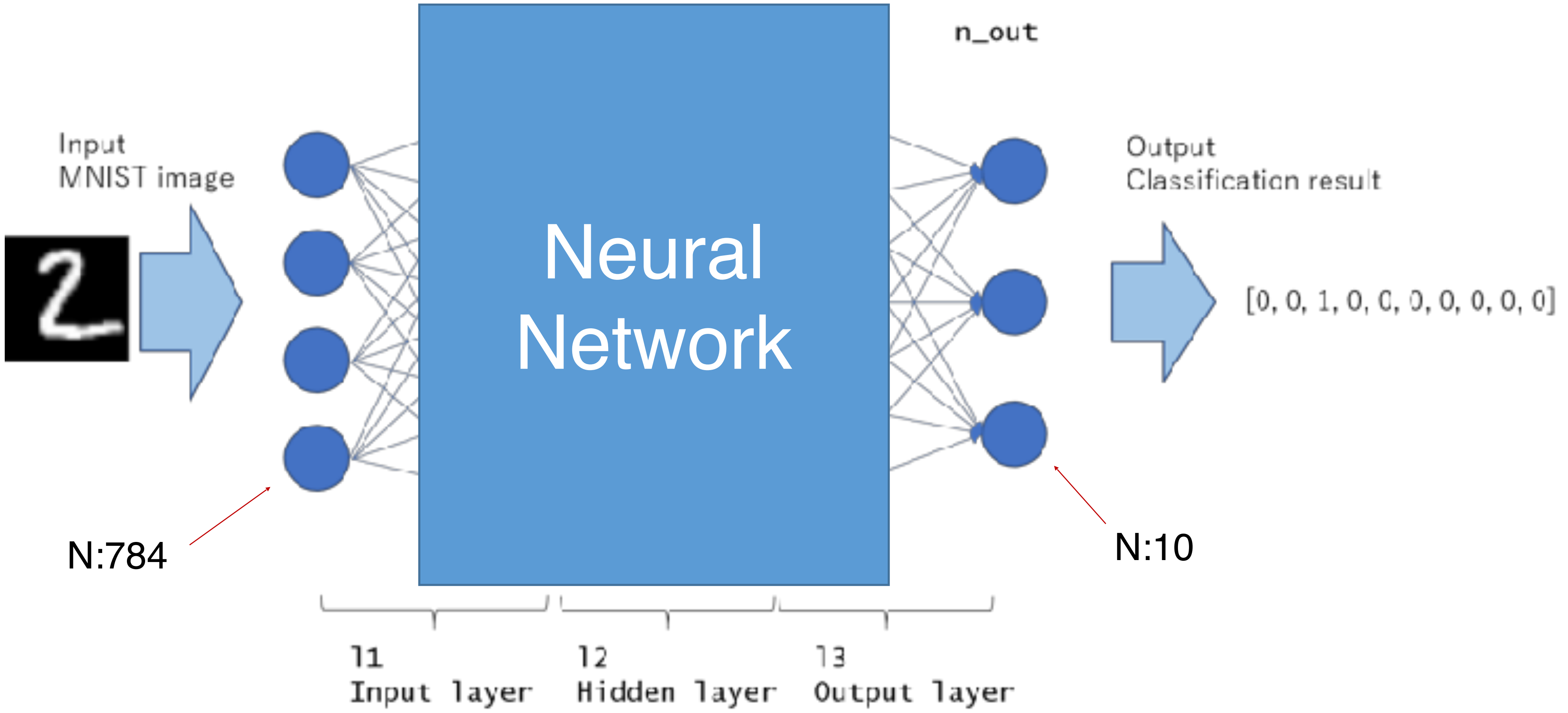
```
y_predict = model.predict(np.array([5]))
print(y_predict)
```

모델 평가

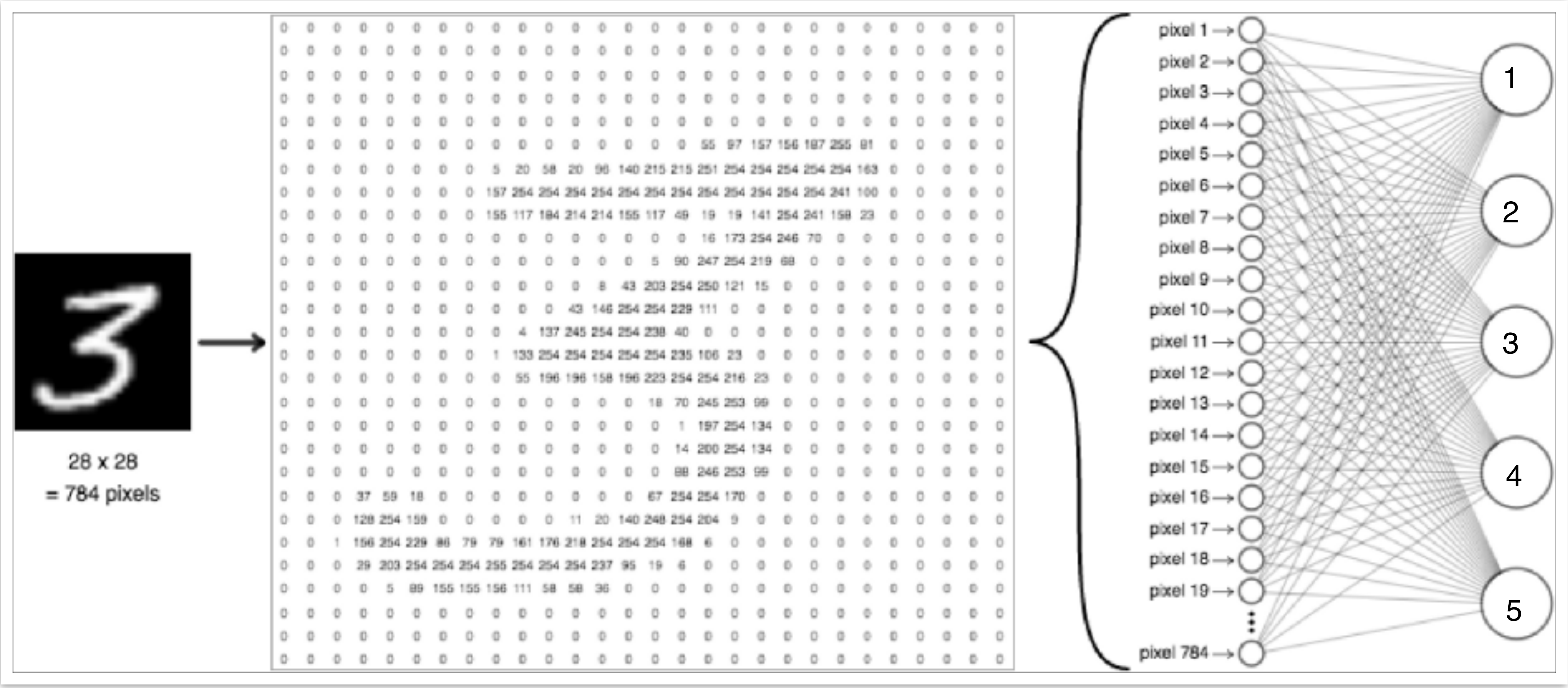
실습 1. Logistic regression

- MNIST (손글씨 숫자 데이터셋) 소개
- Softmax
- Cross-Entropy
- Batch & Epoch
- Train & Validation & Test Data

우리의 목표 = 정확도 99% 도전!



No Hidden Layer



784 x 10 = 7840 Weights

Lab 1 코드 설명(1)

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# for Using TensorFlow backend.
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1] * x_train.shape[2])
x_train = x_train.astype('float32') / 255
# one_hot
y_train = np_utils.to_categorical(y_train, nb_classes)
```

60000

28*28=784

(60000,28,28) -> (60000,784)

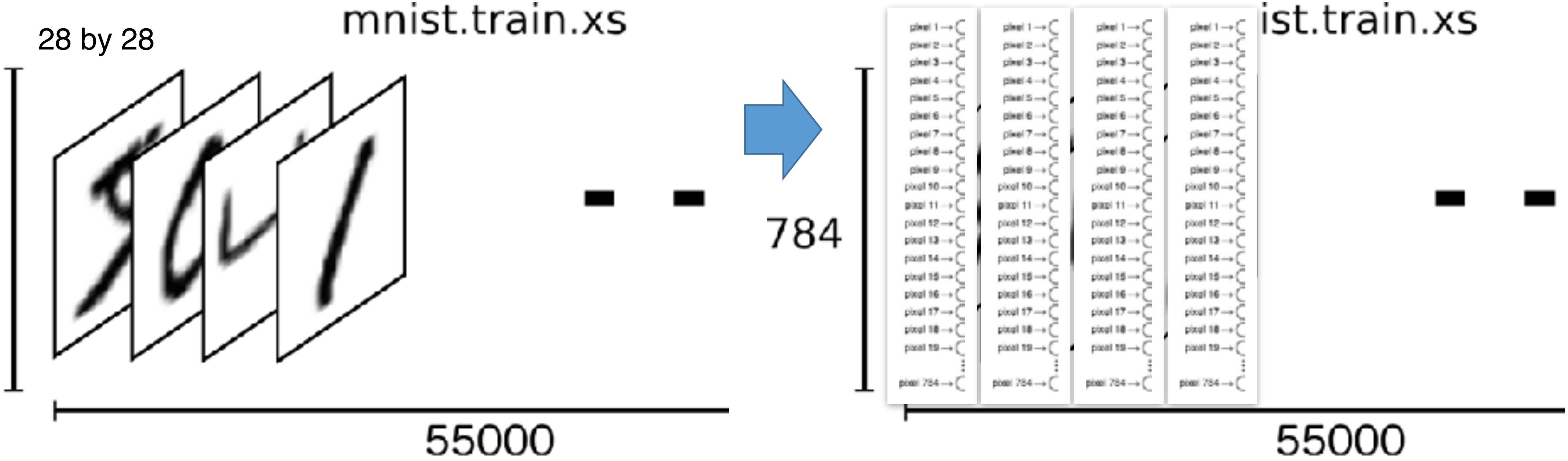
Array 펴주기

0~255를 0~1로 다시 스케일링

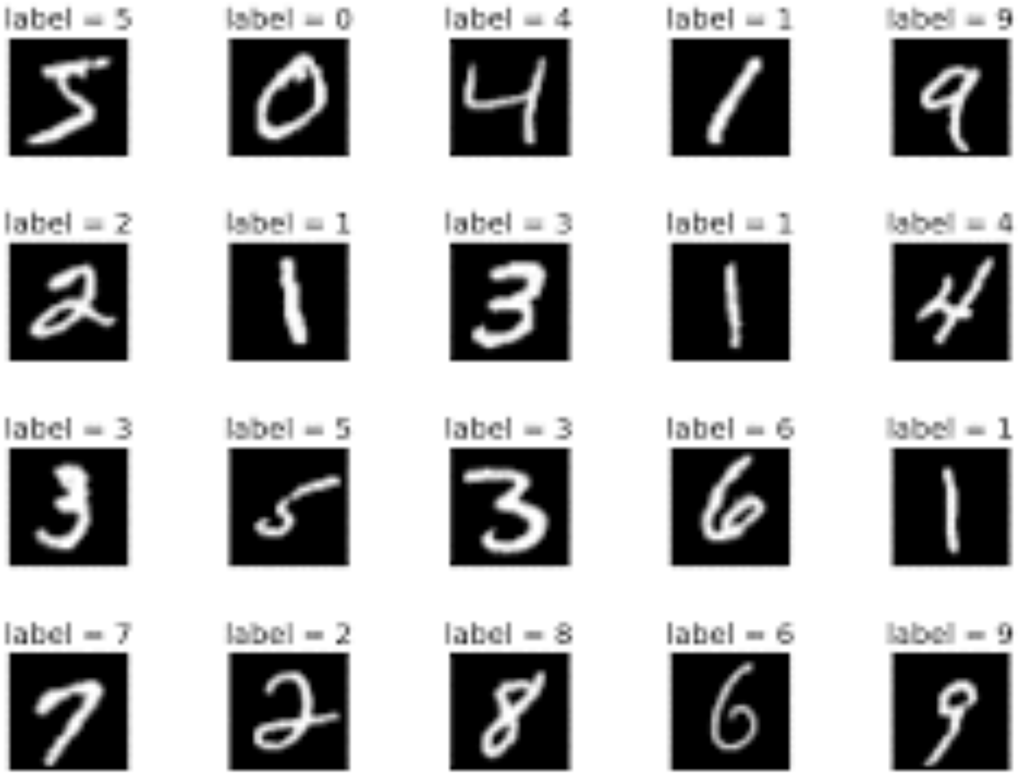
1,2,3,4,5,6 값을 one hot encoding

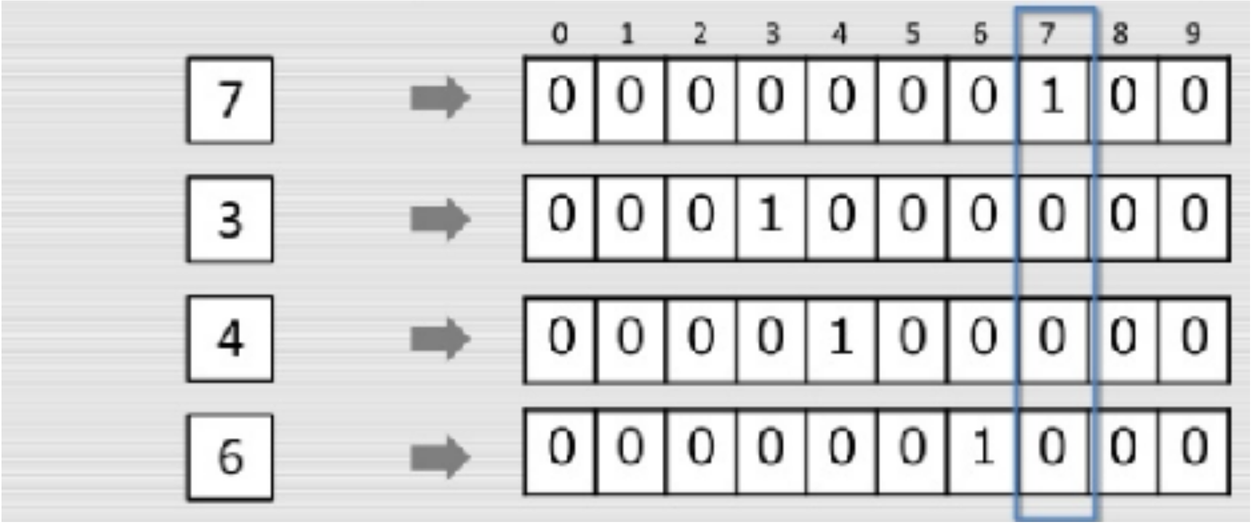
몰라도 됩니다....

X_train.reshape()



One-Hot Encoding



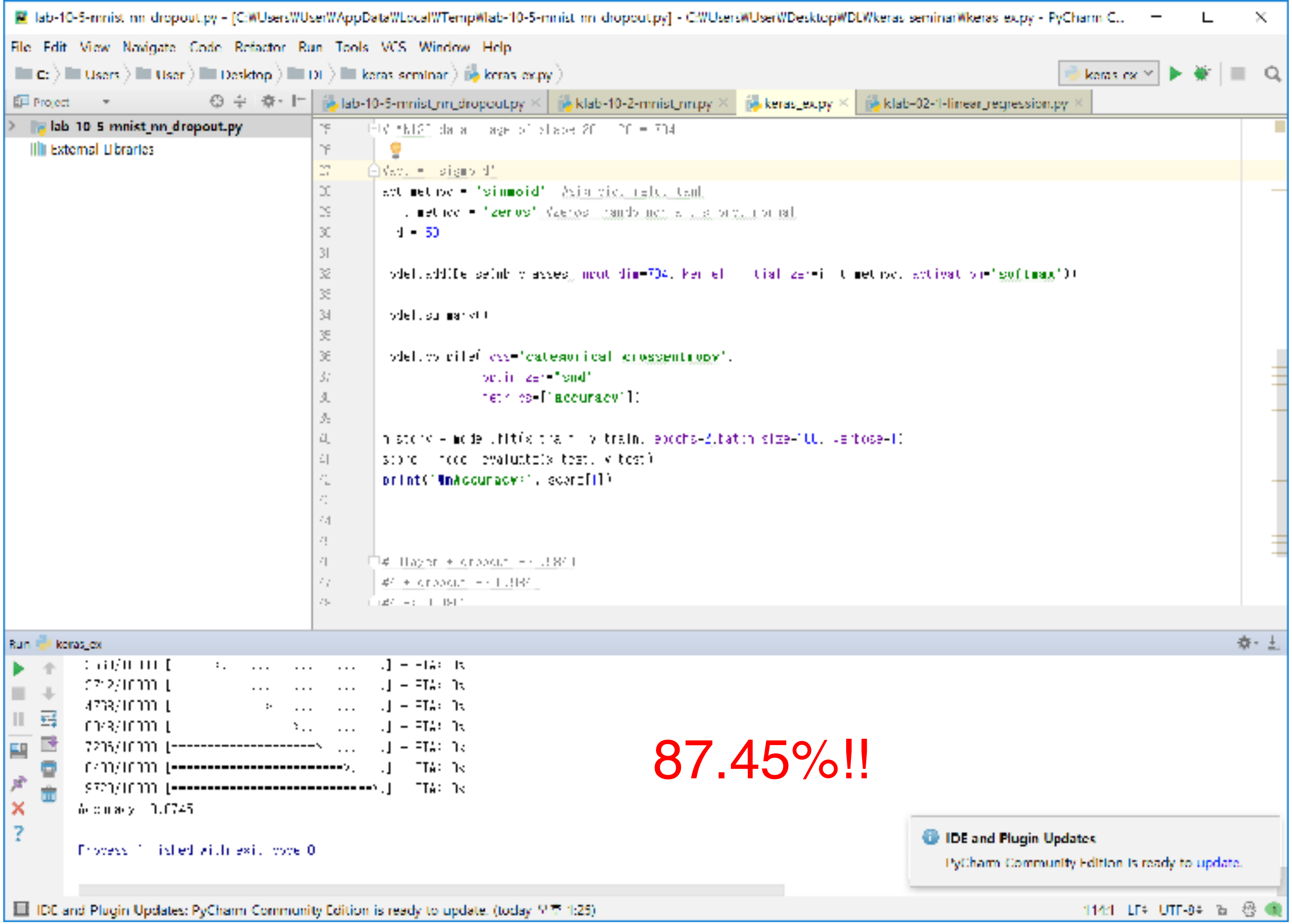



```
y_train = np_utils.to_categorical(y_train, nb_classes)
```

Lab 1 코드 설명

10

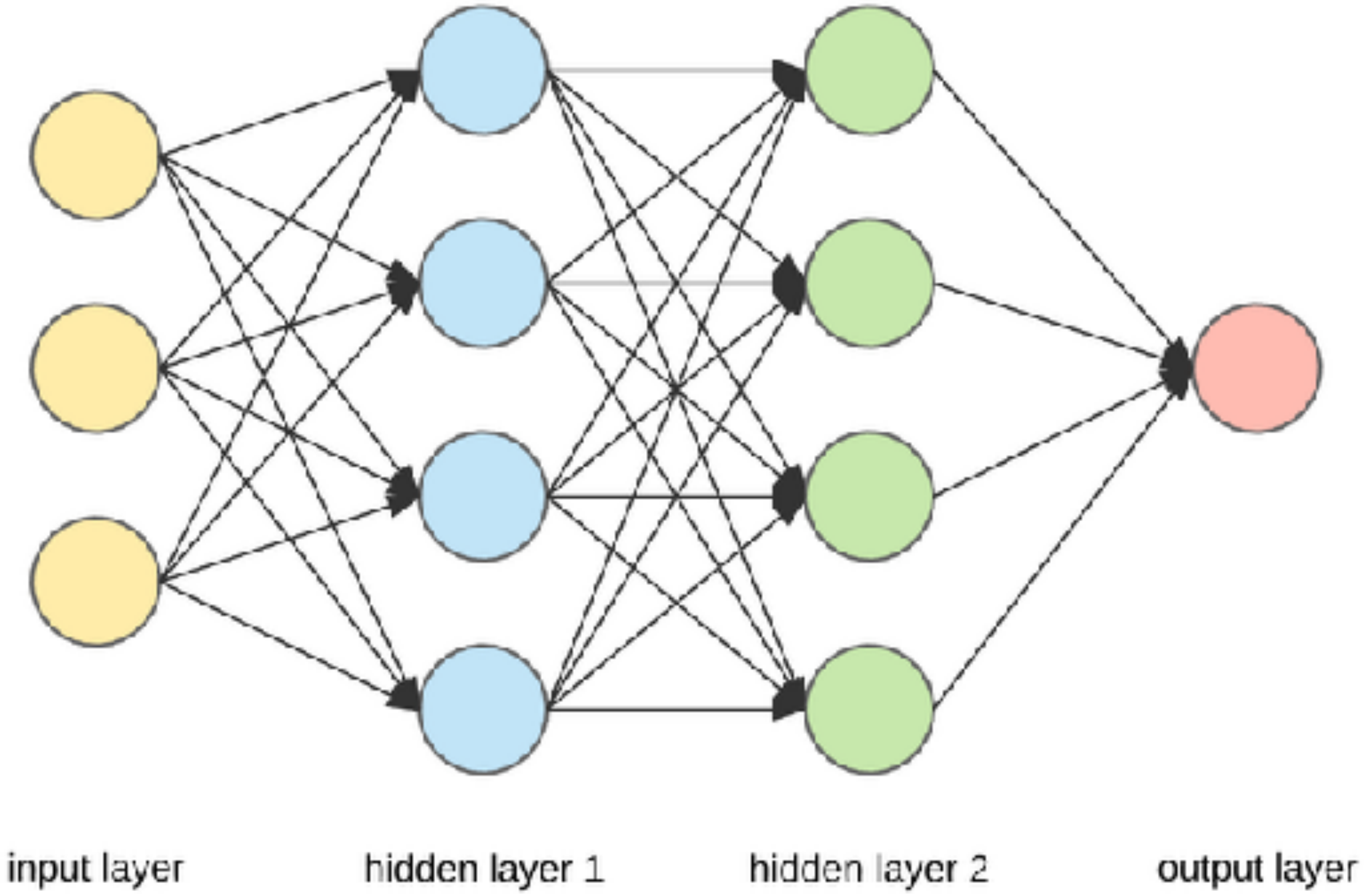
```
model.add(Dense(nb_classes,input_dim=784, kernel_initializer=init_method, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=2,batch_size=100, verbose=1)
score = model.evaluate(x_test, y_test)
print('₩nAccuracy:', score[1])
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab-10-5-mnist_nn_dropout.py
klab-10-2-mnist_nn.py
keras_ex.py
klab-02-1-linear_regression.py
External Libraries
IDE and Plugin Updates
PyCharm Community Edition is ready to update.
87.45%!!

Lab 2. Deep Neural Network

- Weight Initialization
- Activation Functions
- Optimization Methods

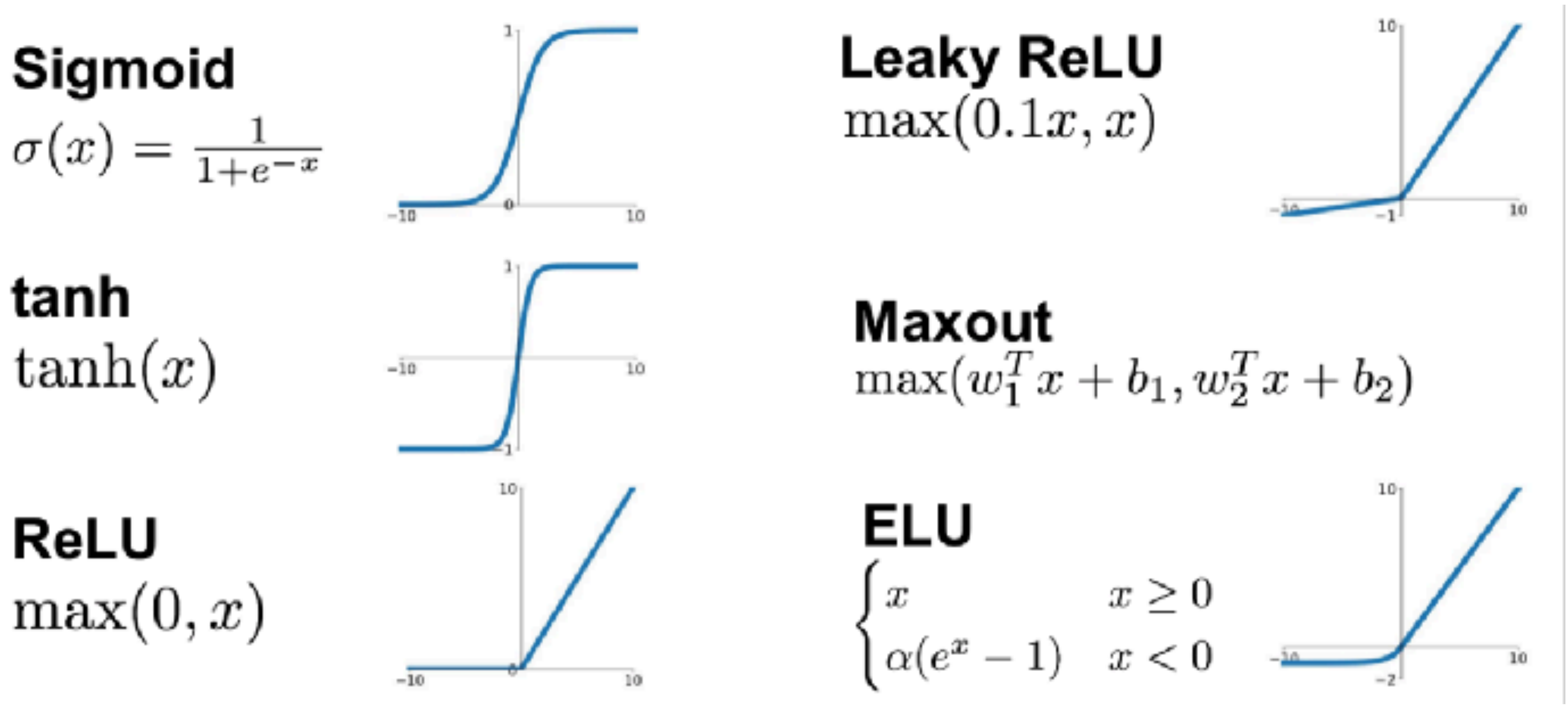
Neural Network



Graph model +
Nonlinear Function (Activation Function)

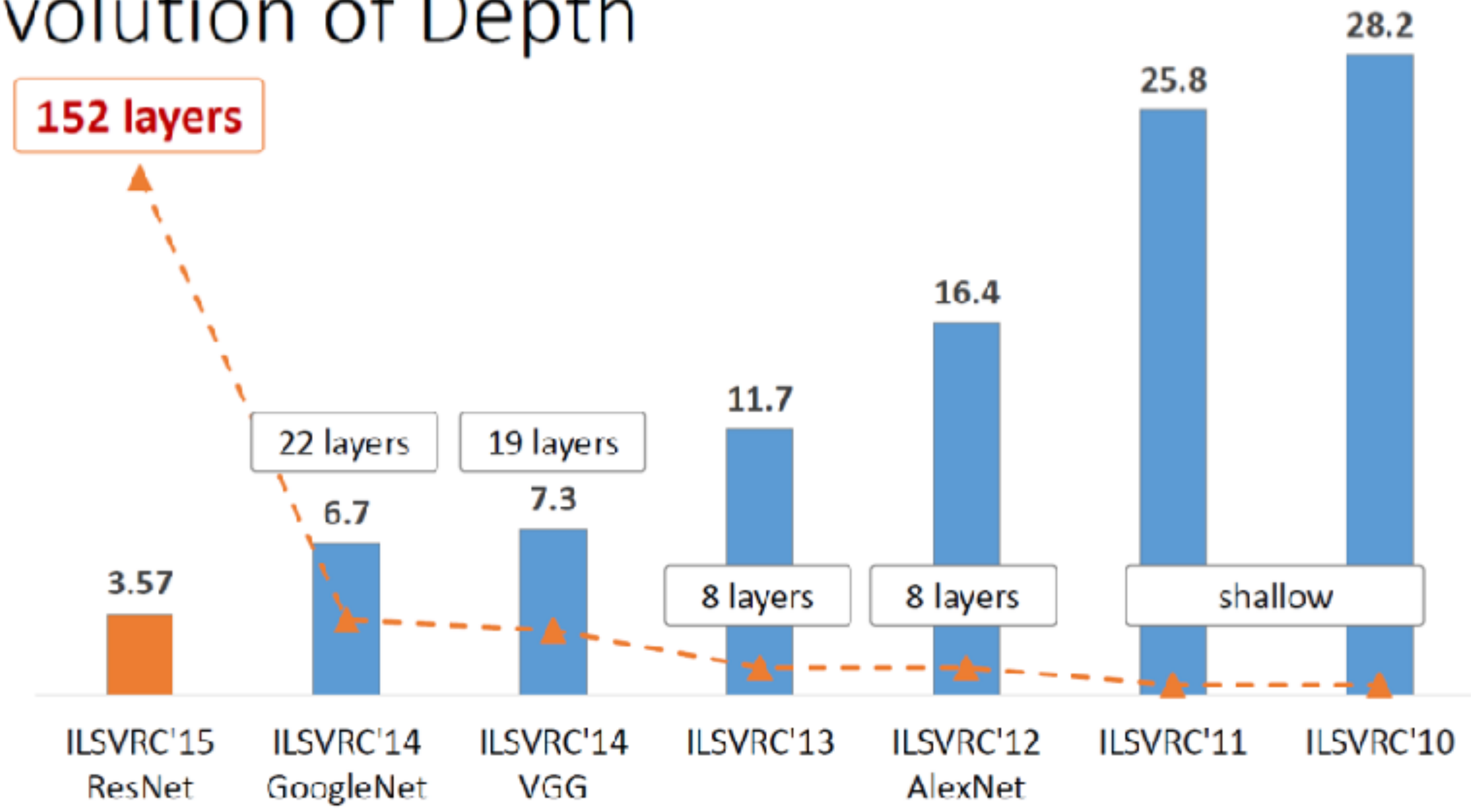
https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Activation Function



https://medium.com/@krishnakalyan3/introduction-to-exponential-linear-unit-d3e2904b366c

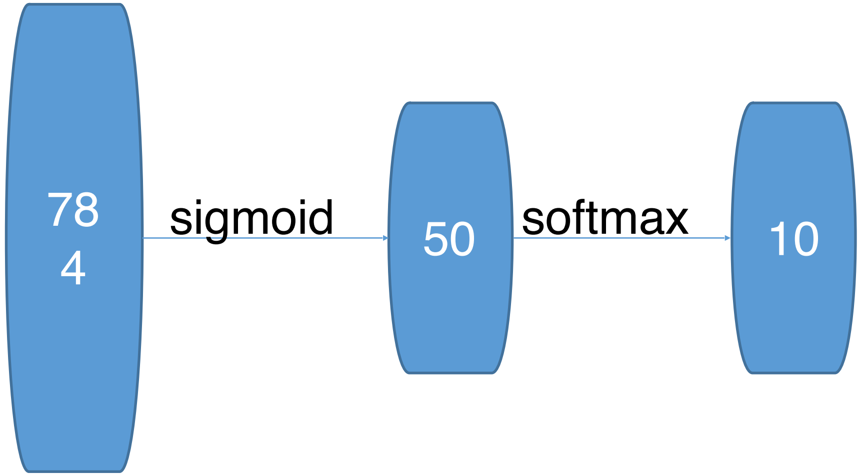
Revolution of Depth



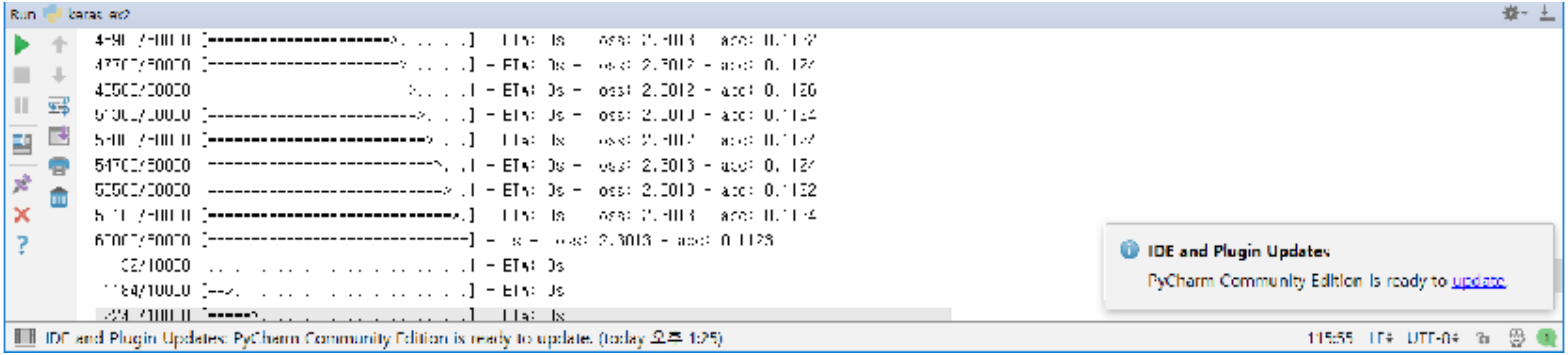
ImageNet Classification top-5 error (%)

Lab 2 코드

```
mid = 50


model.add(Dense(mid, input_dim=784,kernel_initializer=init_method))
model.add(Activation(act_method))
model.add(Dense(nb_classes, kernel_initializer=init_method, activation='softmax'))
```



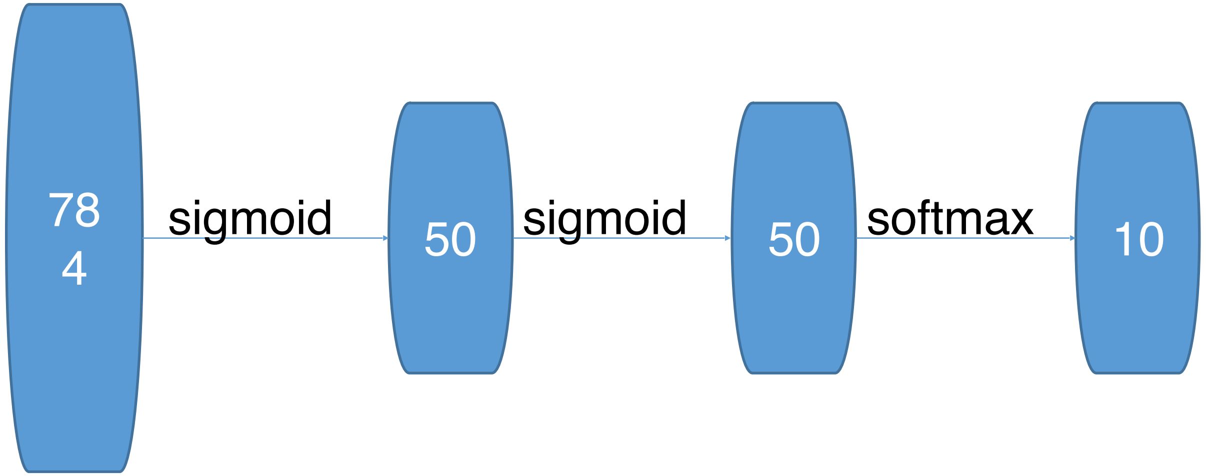
Lab 2 실행 결과



11.23% ?????????

Lab 2-1. Add 2 Hidden Layers

```
model.add(Dense(mid, input_dim=784,kernel_initializer=init_method))
model.add(Activation(act_method))

model.add(Dense(mid,kernel_initializer=init_method))
model.add(Activation(act_method))

model.add(Dense(nb_classes, kernel_initializer=init_method, activation='softmax'))
```



Hidden Layer 추가 후 결과

```
50200/60000 [========================>.....] - ETA: 0s - loss: 2.2660 - acc: 0.1780
51800/60000 [========================>.....] - ETA: 0s - loss: 2.2655 - acc: 0.1789
53500/60000 [=========================>....] - ETA: 0s - loss: 2.2650 - acc: 0.1799
55100/60000 [==========================>...] - ETA: 0s - loss: 2.2646 - acc: 0.1809
56800/60000 [===========================>..] - ETA: 0s - loss: 2.2641 - acc: 0.1828
58300/60000 [============================>.] - ETA: 0s - loss: 2.2637 - acc: 0.1849
59900/60000 [============================>.] - ETA: 0s - loss: 2.2632 - acc: 0.1876
60000/60000 [==============================] - 1s - loss: 2.2632 - acc: 0.1877
```

Activation Function을 Relu로 수정

```
act_method = 'relu'  #sigmoid, relu, tanh
init_method = 'glorot_normal' #zeros, random_normal, glorot_normal
```

```
48100/60000 [========================>......] - ETA: 0s - loss: 0.5644 - acc: 0.8526
49900/60000 [========================>......] - ETA: 0s - loss: 0.5613 - acc: 0.8533
51500/60000 [=========================>.....] - ETA: 0s - loss: 0.5585 - acc: 0.8538
53300/60000 [==========================>....] - ETA: 0s - loss: 0.5548 - acc: 0.8548
55000/60000 [===========================>...] - ETA: 0s - loss: 0.5523 - acc: 0.8554
56600/60000 [============================>..] - ETA: 0s - loss: 0.5499 - acc: 0.8560
58300/60000 [=============================>.] - ETA: 0s - loss: 0.5460 - acc: 0.8568
60000/60000 [==============================] - 1s - loss: 0.5427 - acc: 0.8574
```

실습: SGD -> ADAM으로 수정

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
54900/60000 [==========================>...] - ETA: 0s - loss: 0.1881 - acc: 0.9446
56200/60000 [===========================>..] - ETA: 0s - loss: 0.1877 - acc: 0.9449
57600/60000 [===========================>..] - ETA: 0s - loss: 0.1868 - acc: 0.9452
59000/60000 [============================>.] - ETA: 0s - loss: 0.1857 - acc: 0.9456
60000/60000 [==============================] - 2s - loss: 0.1852 - acc: 0.9456
```

?!

Lab 2. Summary

- Deep Neural Network
- Activation Functions (Sigmoid -> Relu)
- Weight Initialization (Xavier/He Initialization)
- Optimization Method (SGD -> ADAM)

- 95%!!!

Lab 3. Deeeep Network

- Over-fitting
- Drop out
- Batch normalization

Lab 3. Go Deep & Wide (실행먼저!)

```
mid = 500

model.add(Dense(mid, input_dim=784,kernel_initializer=init_method))
model.add(Activation(act_method))

model.add(Dense(mid,kernel_initializer=init_method))
model.add(Activation(act_method))

model.add(Dense(mid,kernel_initializer=init_method))
model.add(Activation(act_method))

model.add(Dense(mid,kernel_initializer=init_method))
model.add(Activation(act_method))

model.add(Dense(nb_classes, kernel_initializer=init_method, activation='softmax'))
```

4 Hidden Layers

```
history = model.fit(x_train, y_train, epochs=15,batch_size=100, verbose=2,validation_data=(x_test, y_test))
```

출력을 1 epoch마다 출력 epoch마다 Validation

```
3s - loss: 0.0260 - acc: 0.9917 - val_loss: 0.0823 - val_acc:
0.9805
Epoch 9/15
3s - loss: 0.0246 - acc: 0.9925 - val_loss: 0.0903 - val_acc:
0.9797
Epoch 10/15
3s - loss: 0.0197 - acc: 0.9943 - val_loss: 0.0856 - val_acc:
0.9802
Epoch 11/15
3s - loss: 0.0195 - acc: 0.9943 - val_loss: 0.0866 - val_acc:
0.9781
Epoch 12/15
3s - loss: 0.0200 - acc: 0.9940 - val_loss: 0.0858 - val_acc:
0.9821
Epoch 13/15
3s - loss: 0.0181 - acc: 0.9945 - val_loss: 0.0874 - val_acc:
0.9813
Epoch 14/15
3s - loss: 0.0153 - acc: 0.9956 - val_loss: 0.0845 - val_acc:
0.9826
Epoch 15/15
3s - loss: 0.0156 - acc: 0.9955 - val_loss: 0.1149 - val_acc:
```

Total params: 1,149,010
Train Data: 60,000

Over-Fitting!

(실습) Dropout 추가

```
model.add(Dense(mid, input_dim=784,kernel_initializer=init_method))
model.add(Dropout(0.25))
model.add(Activation(act_method))

model.add(Dense(mid,kernel_initializer=init_method))
model.add(Dropout(0.25))
model.add(Activation(act_method))

model.add(Dense(mid,kernel_initializer=init_method))
model.add(Dropout(0.25))
model.add(Activation(act_method))

model.add(Dense(mid,kernel_initializer=init_method))
model.add(Dropout(0.25))
model.add(Activation(act_method))
```

랜덤하게 25% node는 죽이겠습니다!

0.75^4 = 0.316

Training 결과

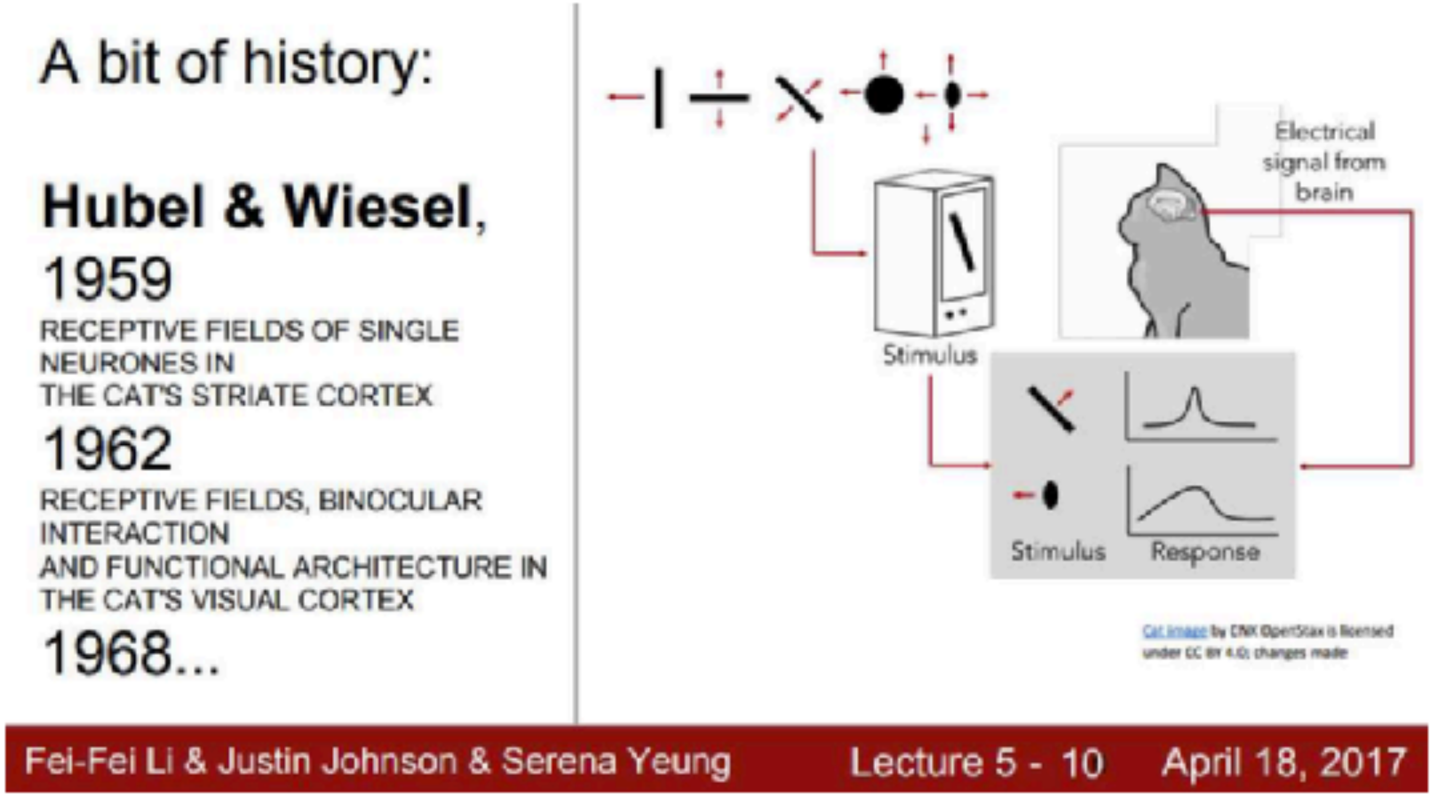
```
15s - loss: 0.0615 - acc: 0.9818 - val_loss: 0.0696 - val_acc: 0.9805
Epoch 8/15
15s - loss: 0.0581 - acc: 0.9829 - val_loss: 0.0730 - val_acc: 0.9808
Epoch 9/15
16s - loss: 0.0506 - acc: 0.9856 - val_loss: 0.0863 - val_acc: 0.9784
Epoch 10/15
17s - loss: 0.0472 - acc: 0.9860 - val_loss: 0.0774 - val_acc: 0.9792
Epoch 11/15
16s - loss: 0.0451 - acc: 0.9865 - val_loss: 0.0862 - val_acc: 0.9783
Epoch 12/15
19s - loss: 0.0402 - acc: 0.9880 - val_loss: 0.0787 - val_acc: 0.9812
Epoch 13/15
16s - loss: 0.0417 - acc: 0.9886 - val_loss: 0.0844 - val_acc: 0.9814
Epoch 14/15
16s - loss: 0.0386 - acc: 0.9886 - val_loss: 0.0750 - val_acc: 0.9812
Epoch 15/15
18s - loss: 0.0351 - acc: 0.9897 - val_loss: 0.0919 - val_acc: 0.9823
```

Hidden 추가 (5 Hidden layer)

```
17s - loss: 0.0563 - acc: 0.9840 - val_loss: 0.0804 - val_acc: 0.9796
Epoch 10/15
17s - loss: 0.0580 - acc: 0.9842 - val_loss: 0.0831 - val_acc: 0.9813
Epoch 11/15
17s - loss: 0.0545 - acc: 0.9848 - val_loss: 0.0768 - val_acc: 0.9805
Epoch 12/15
17s - loss: 0.0498 - acc: 0.9858 - val_loss: 0.0770 - val_acc: 0.9826
Epoch 13/15
17s - loss: 0.0464 - acc: 0.9880 - val_loss: 0.0847 - val_acc: 0.9818
Epoch 14/15
17s - loss: 0.0482 - acc: 0.9870 - val_loss: 0.0970 - val_acc: 0.9815
Epoch 15/15
17s - loss: 0.0432 - acc: 0.9881 - val_loss: 0.0919 - val_acc: 0.9804
```

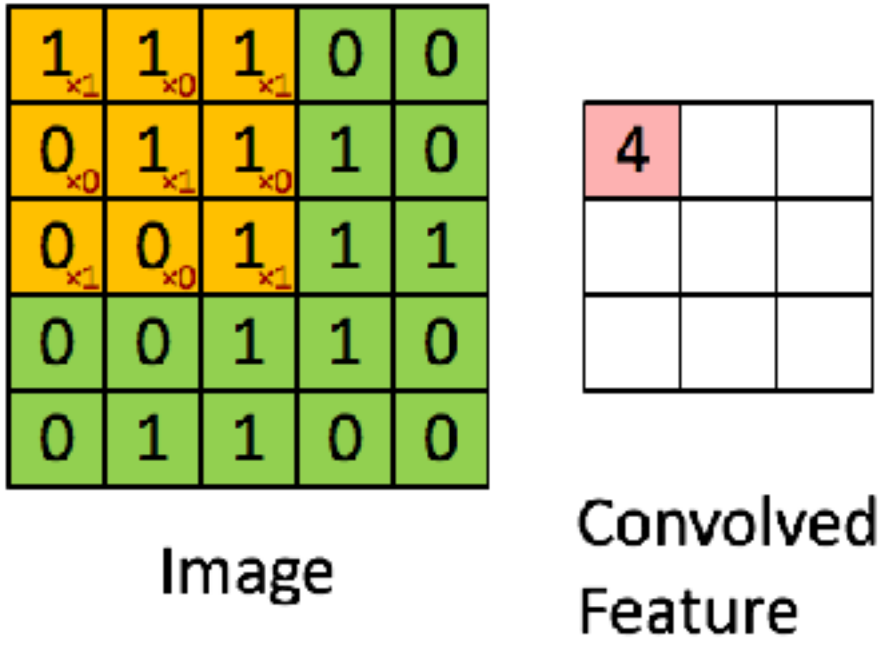
Lab 4. CNN
(Convolutional Neural Network)

- motivation



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolution Layer

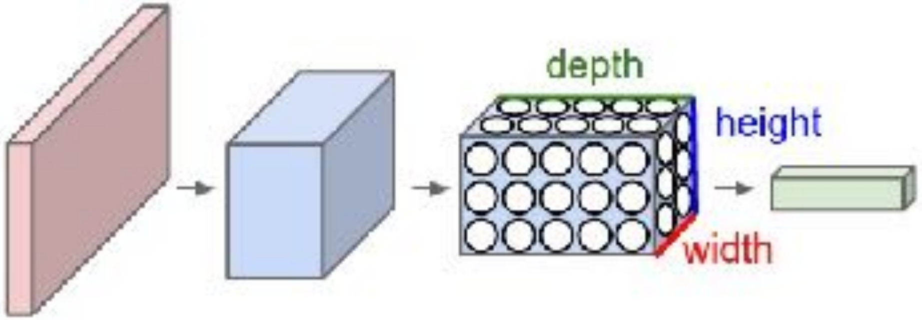


변수:

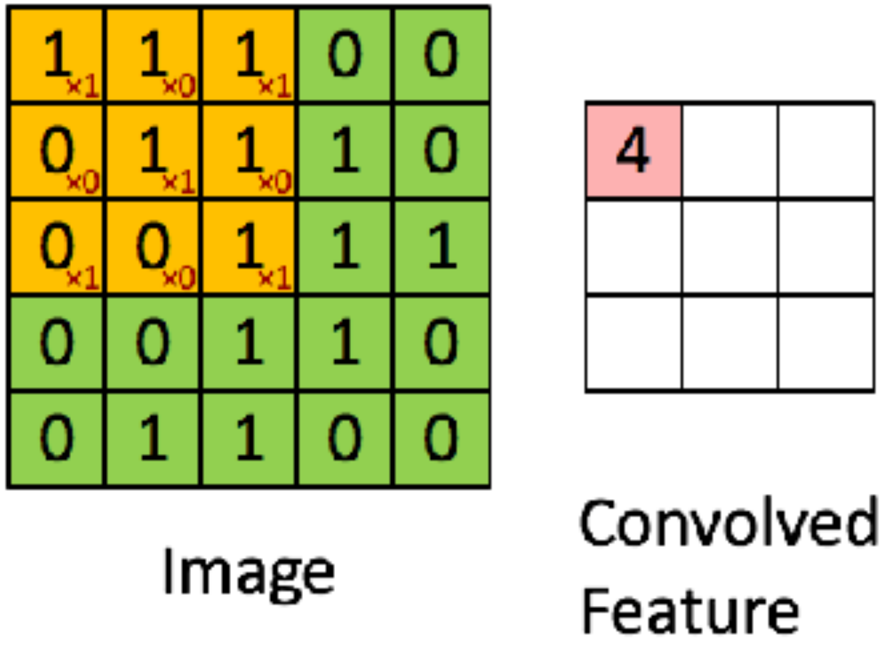
-Filter Size: (3,3)
-Stride : (1,1)

Height, width

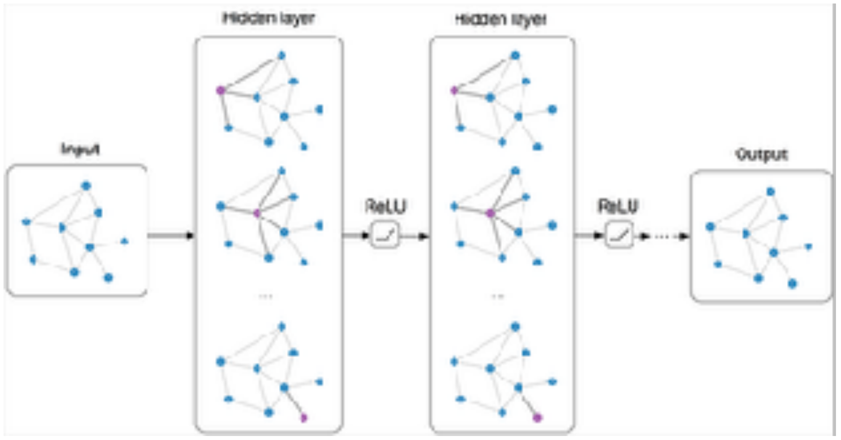
-Filter 수 -> depth



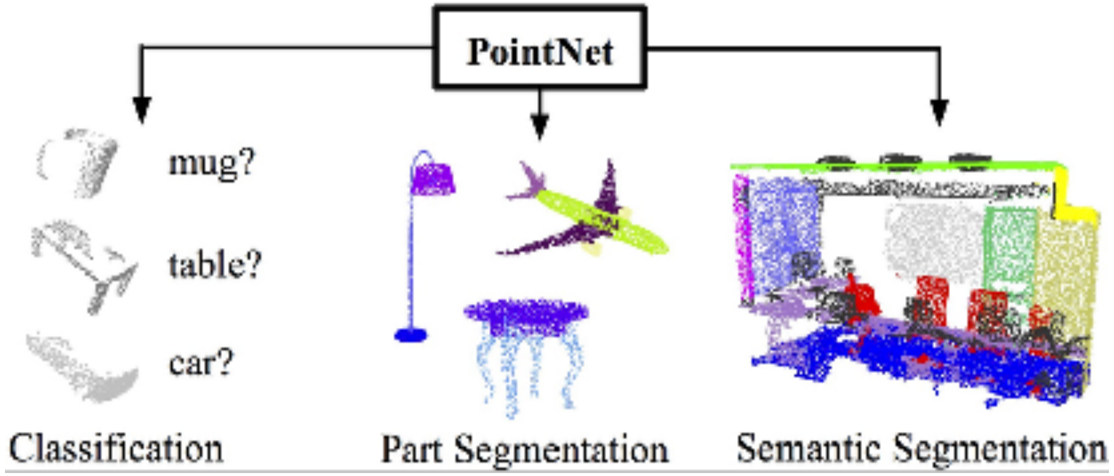
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution



Convolutional Neural Network

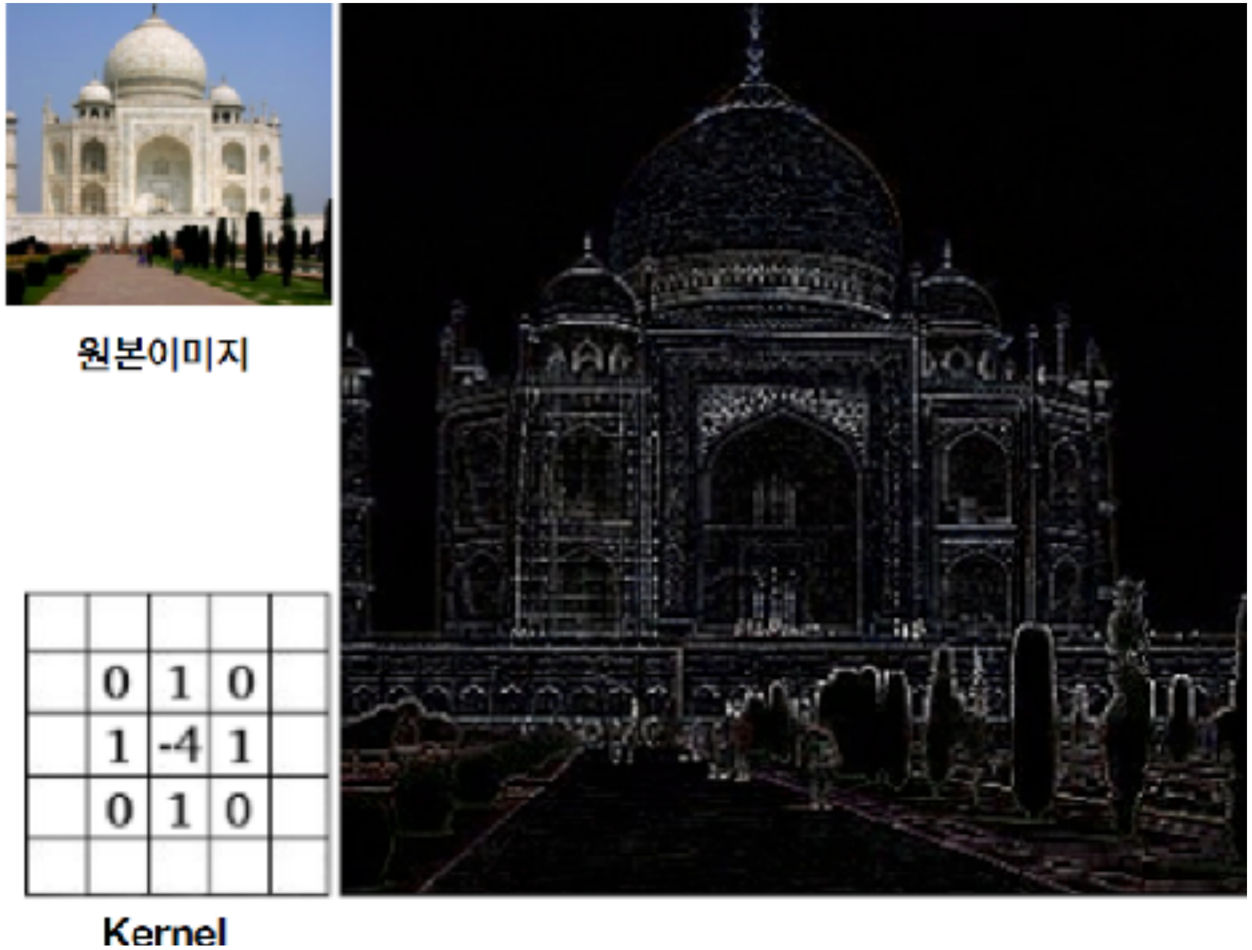


Graph Neural Network



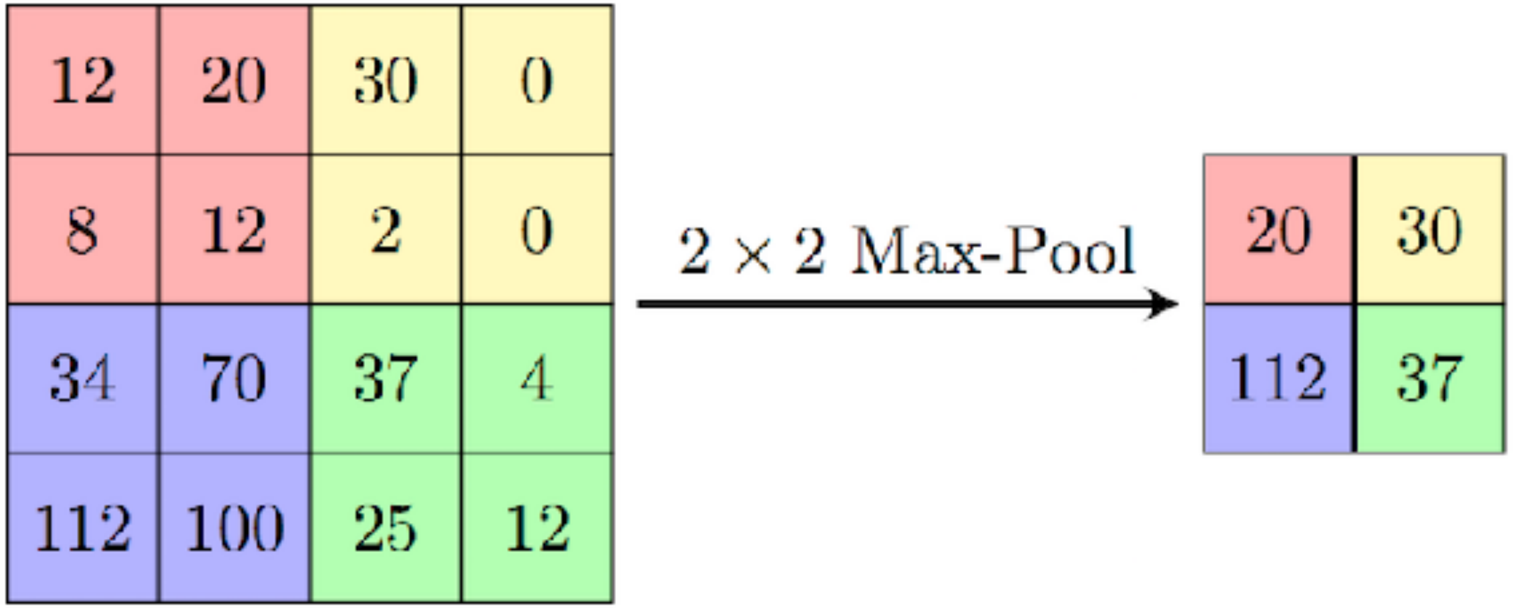
Point Net

Convolution Operation

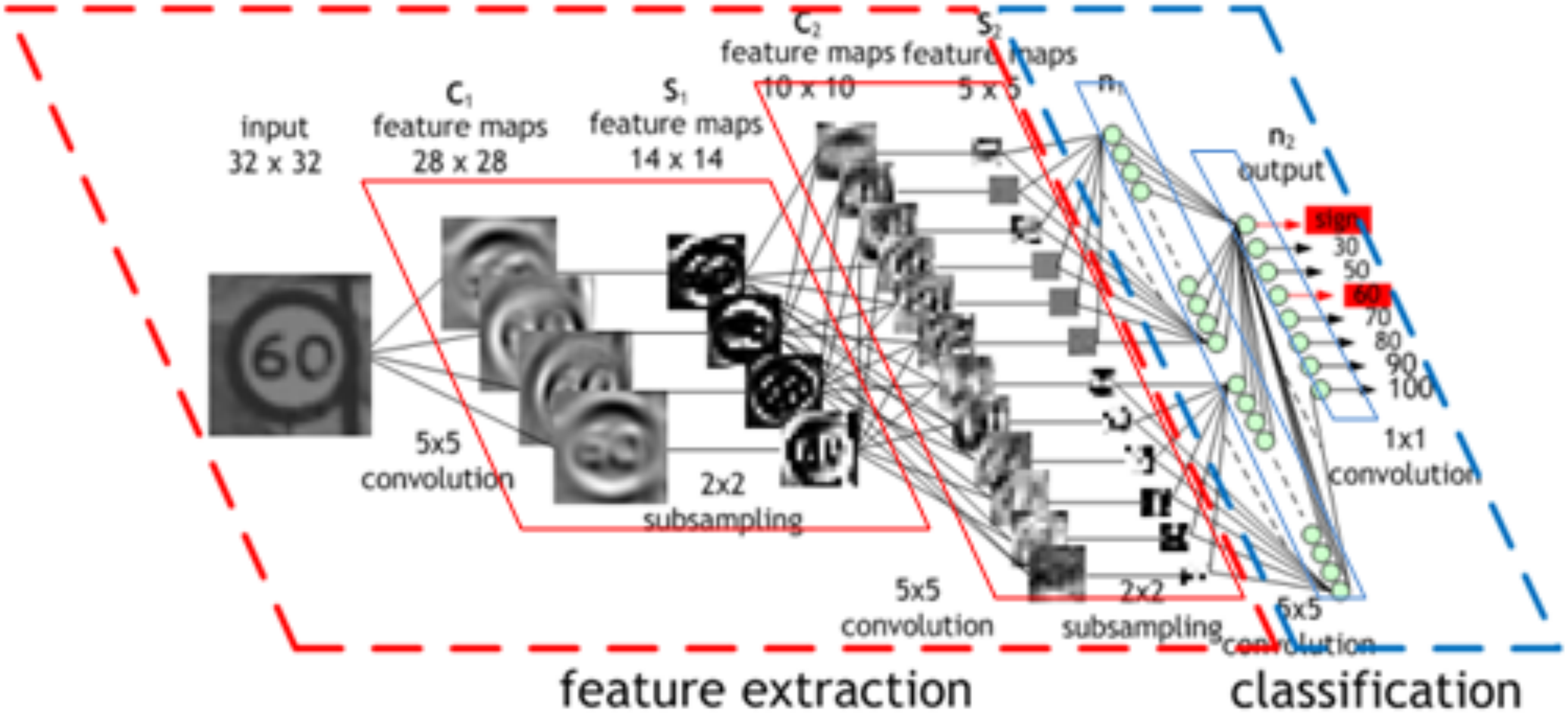


https://docs.gimp.org/2.8/en/plug-in-convmatrix.html

Pooling



http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/



차원 변화

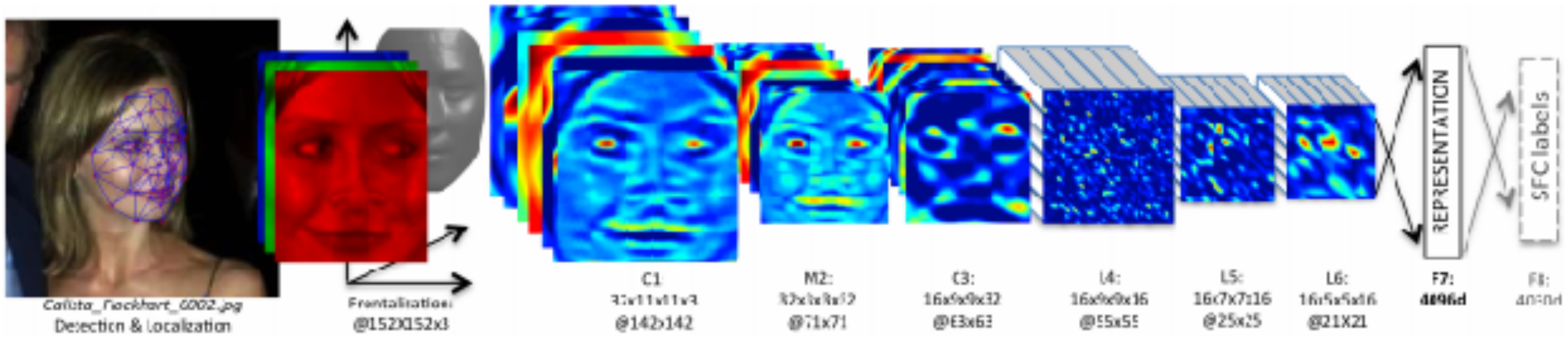


Figure 2. **Outline of the *DeepFace* architecture.** A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

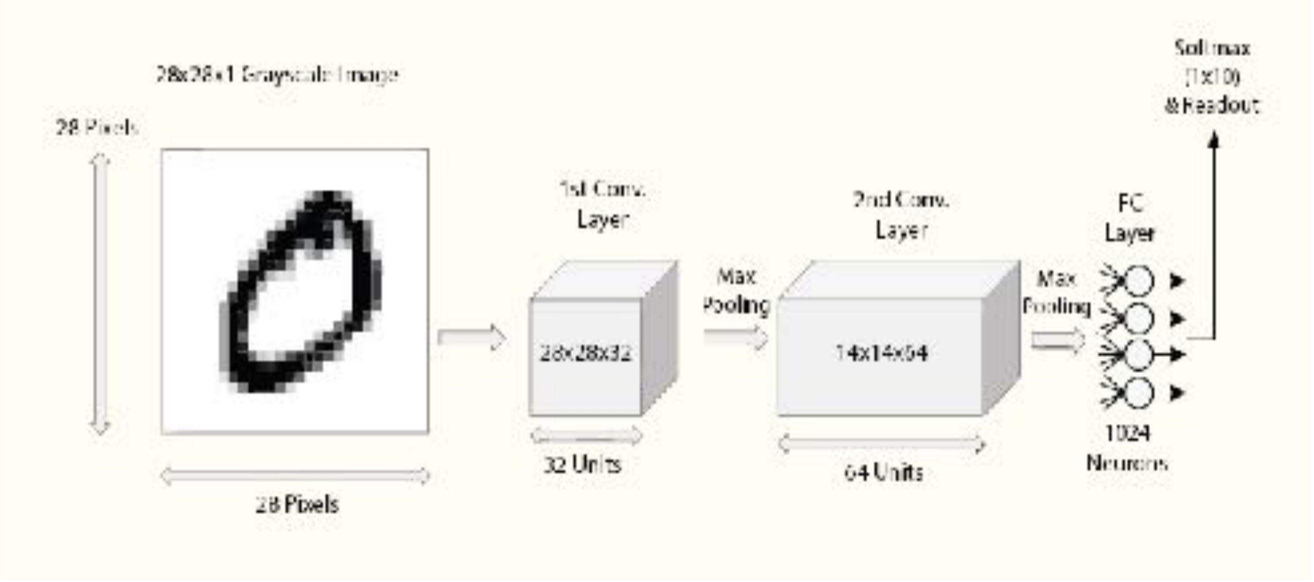
실습 4: 먼저 실행~~!

필터 갯수　　필터 크기　　Stride는 지정하지 않으면 1

```
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(pooling.MaxPooling2D(pool_size=(2, 2)))
#model.add(Dropout(0.25))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(128))
#model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Activation('relu'))

model.add(Dense(10, activation='softmax'))
```

1D로 쫘악 펴주기

26 by 26
24 by 24
12 by 12



실행 결과

- 5s - loss: 0.0296 - acc: 0.9905 - val_loss: 0.0376 - val_acc: 0.9895
- Epoch 7/12
- 5s - loss: 0.0259 - acc: 0.9916 - val_loss: 0.0359 - val_acc: 0.9906
- Epoch 8/12
- 5s - loss: 0.0234 - acc: 0.9922 - val_loss: 0.0313 - val_acc: 0.9912
- Epoch 9/12
- 5s - loss: 0.0230 - acc: 0.9928 - val_loss: 0.0361 - val_acc: 0.9897
- Epoch 10/12
- 5s - loss: 0.0198 - acc: 0.9938 - val_loss: 0.0412 - val_acc: 0.9905
- Epoch 11/12
- 5s - loss: 0.0192 - acc: 0.9936 - val_loss: 0.0362 - val_acc: 0.9925
- Epoch 12/12
- 5s - loss: 0.0163 - acc: 0.9948 - val_loss: 0.0354 - val_acc: 0.9905

How far can we go with MNIST??

A collection of implementations for 'how far can we go with MNIST' challenge, which has been held in TF-KR at April 2017.

https://github.com/hwalsuklee/how-far-can-we-go-with-MNIST

List of Implementations

Kyung Mo Kweon

99.7% 99.6% 99.6%

- Test error : 0.20%
- Features : keras, esemble of 3 models (small VGG, small Resnet, very small VGG)
- https://github.com/kkweon/mnist-competition

Junbum Cha

- Test error : 0.24%
- Features : tensorflow, ensemble of 3 models (VGG-like with batch size 64/128, resnet 32layers), best accuracy with a single model is 99.74%, data augmentation (rotation, shift, zoom)
- https://github.com/khanrc/mnist

Jehoon Shin

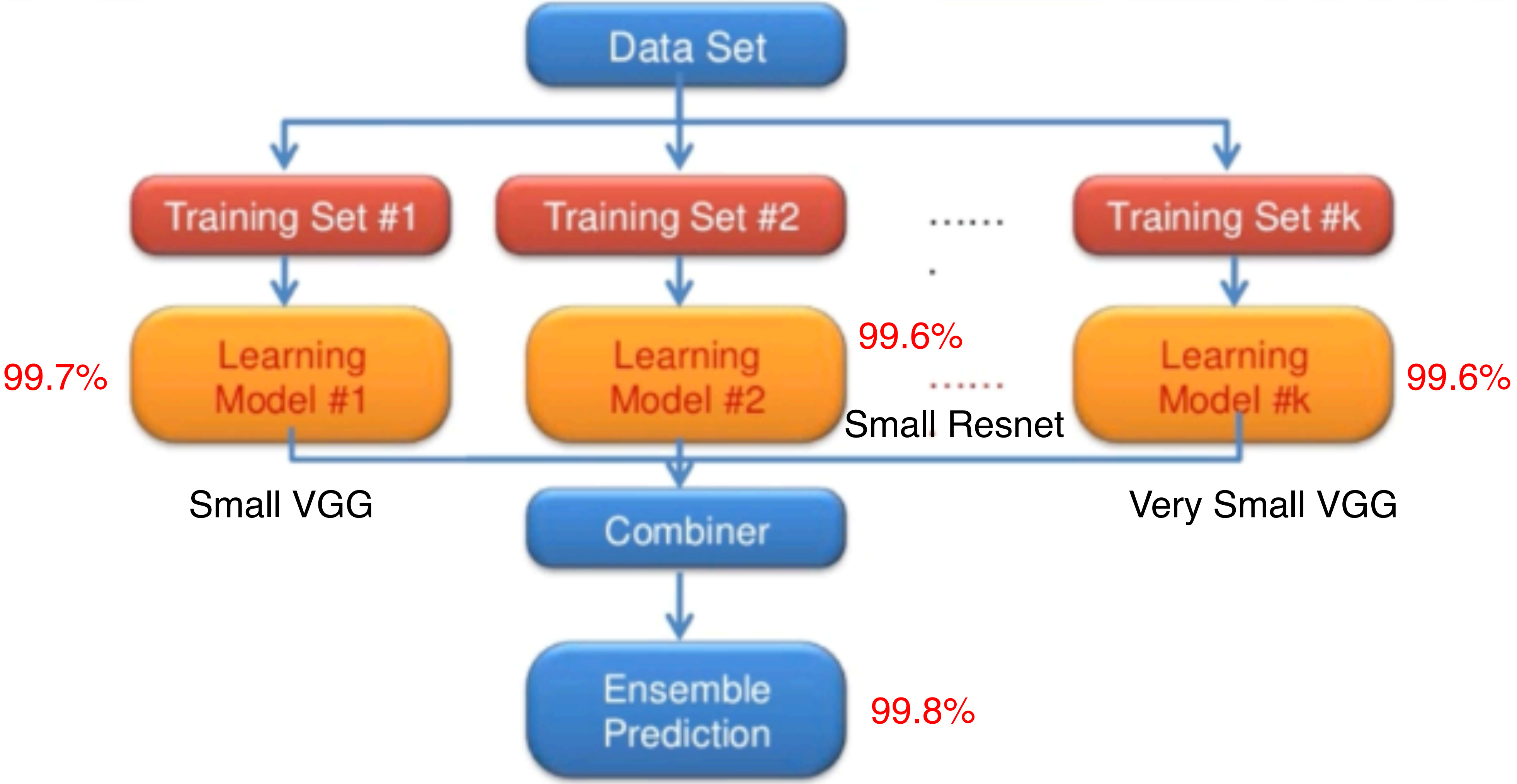
- Test error : 0.26%
- Features : tensorflow, ensemble of 5 models obtained with different hyper-params and same architecture (4 conv-layers, 1 fc-layer), best accuracy with a single model is 0.9968
- https://github.com/zeran4/mnist_trial_and_error/blob/master/lab-11-5-1-mnist_cnn_ensemble_layers_tensorflow-kr.py

Owen Song

- Test error : 0.28%
- Features : keras (theano base), ensemble of 5 models obtained with different hyper-params and same architecture (6

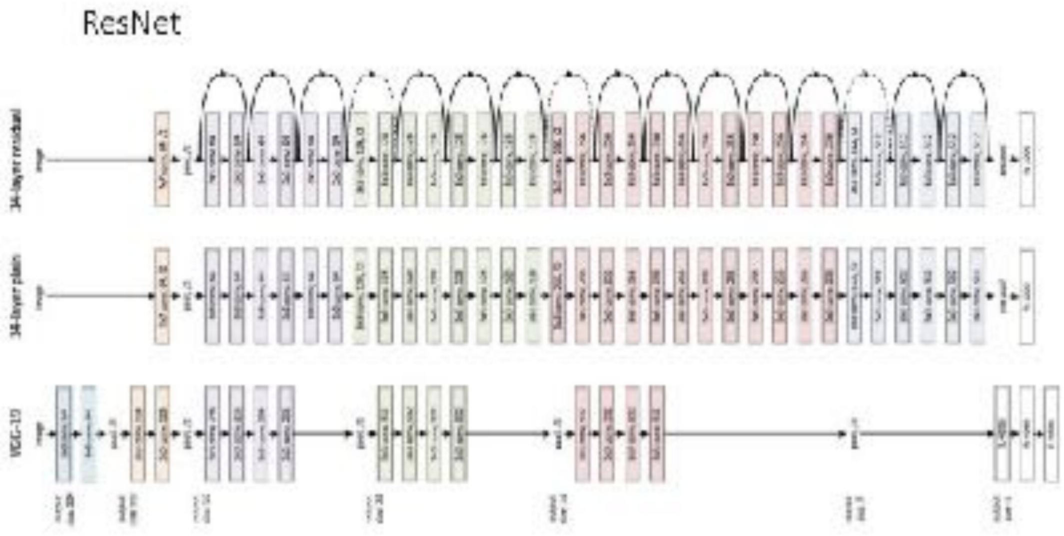
https://github.com/hwalsuklee/how-far-can-we-go-with-MNIST

What is Ensemble?

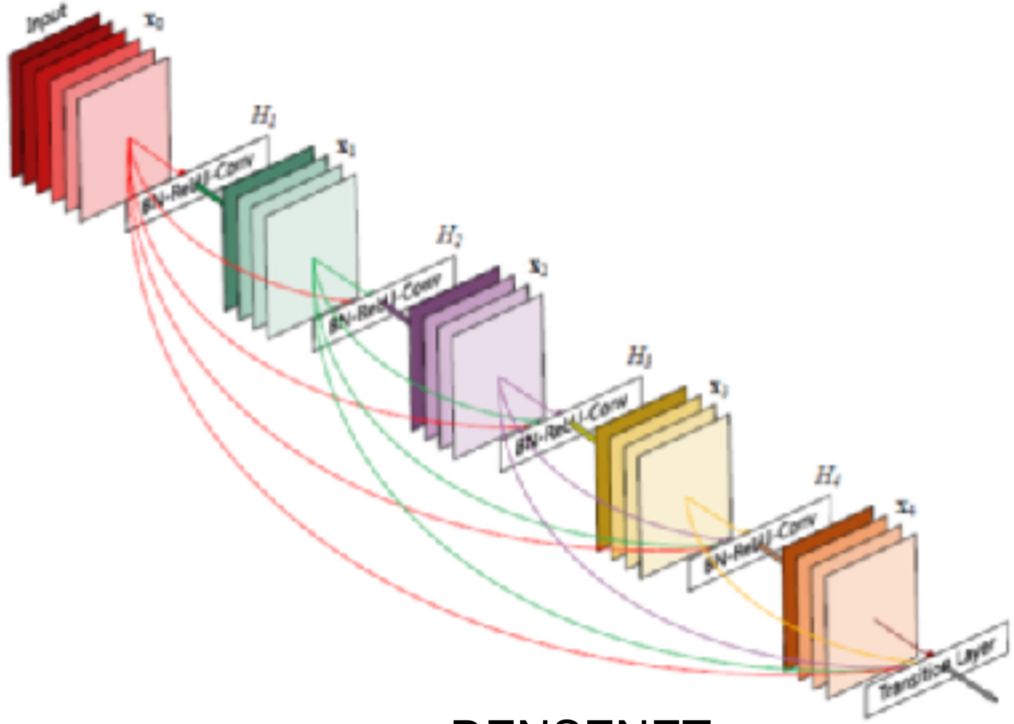


http://www.slideshare.net/sasasiapacific/ipb-improving-the-models-predictive-power-with-ensemble-approaches

Fast Forward
(Shortcut, Highway Network)

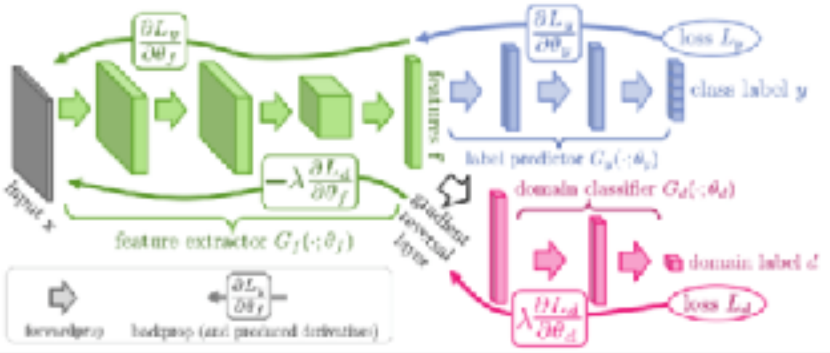


RESNET, winner of IMAGENET 2015

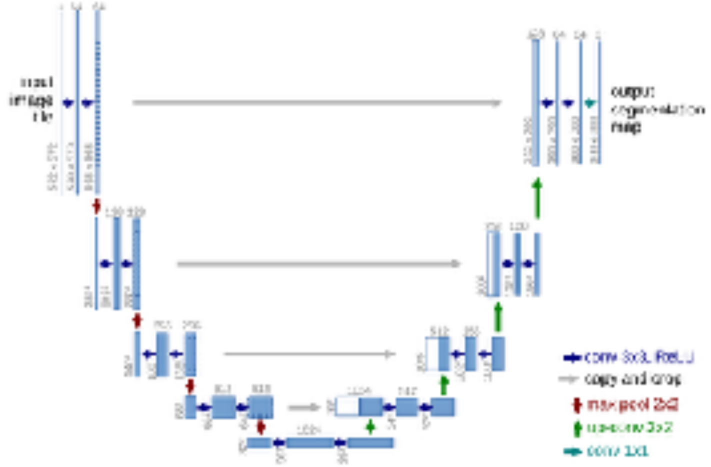


DENSENET

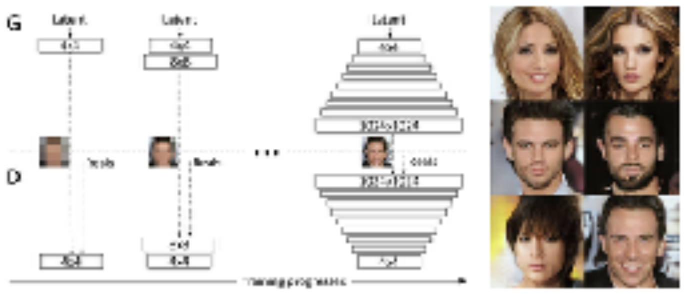
Variants…



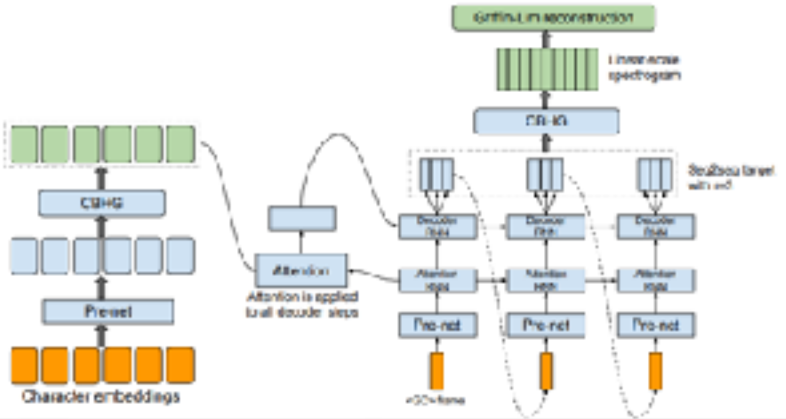
DANN https://arxiv.org/abs/1505.07818



UNET https://arxiv.org/abs/1505.04597



PG-GAN https://arxiv.org/pdf/1710.10196.pdf



Tacotron https://arxiv.org/pdf/1703.10135.pdf

앞으로 무엇을 할까요?

- 캐글 시스템 이용법 (다음주, 이유한)
- ETRI BeeAI 사용방법 (다다음주, 김귀훈)

- 숫자인식 알고리즘을 임베디드 장비에 심기
- 휴대폰 앱 만들기

- 학습데이터와 현장이 다른 경우 어떻게 극복할 것인가?
  - Domain adaptation ??

주요 참고자료

- 김성훈 교수, 모두를 위한 딥러닝 강의
  - 예제 코드는 해당 강의 KERAS 코드를 세미나 내용과 offline환경에 맞게 수정)
- 하용호, 백날 자습해도 이해안가던 딥러닝 머릿속에 인스톨해 드립니다.
- 김진호, Deep learning Short Course, ICEC 2017