디지털 영상처리 개요

강사: 홍영택 박사

1. 영상 처리의 개요

1. 영상 처리의 개요

- 영상 처리 (image processing) 란?
  - 영상을 대상으로 하는 신호 처리(signal processing)의 한 분야
  - 영상의 화질 향상, 소실된 정보의 복원, 데이터의 압축, 영상의 인식 등을 포함

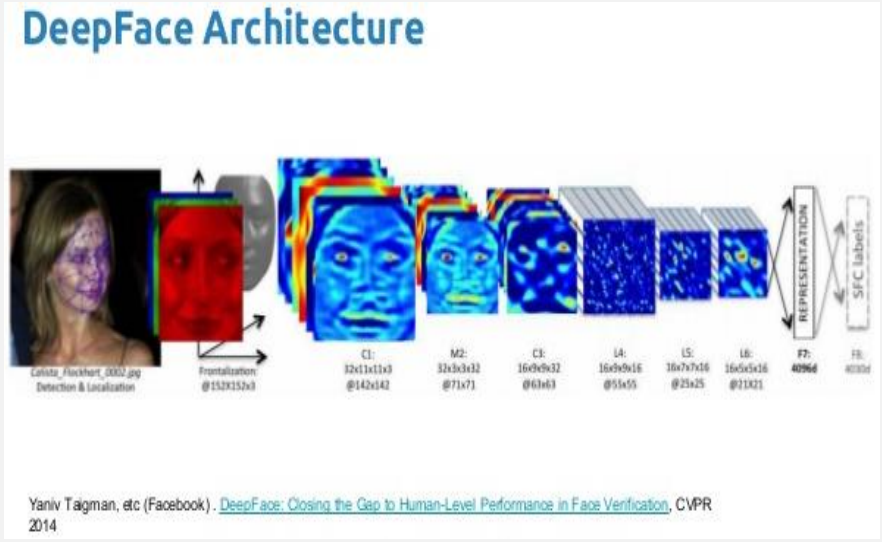
- 영상 처리의 역사
  - 20세기 중반까지 아날로그(analog) 방식
  - 1960년대 미국에서 위성으로 부터 전송 받은 달 표면 사진의 화질을 복원시키는 방법에 대한 연구가 디지털 영상 처리의 시초

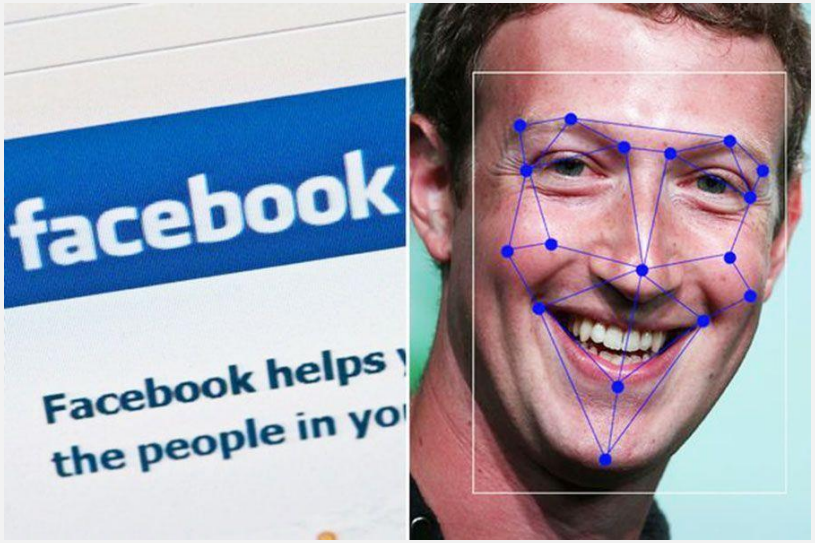
1. 영상 처리의 개요

- 영상 처리의 분야
  - 영상의 기하학적 변형 (geometric transform)
  - 영상의 화질 개선(enhancement)
  - 영상의 복원(restoration)
  - 영상 데이터 압축(compression)
  - 객체 인식(object recognition)

1. 영상 처리의 개요

- 얼굴 검색 및 인식
  - 영상 내에 존재하는 얼굴의 위치를 찾고, 그 얼굴에 해당하는 사람을 인식하는 기술





1. 영상 처리의 개요

- 영상 화질 개선
  - 저화질의 영상(이미지 또는 비디오)를 고화질로 개선하는 방법
  - 디지털 카메라, HDTV

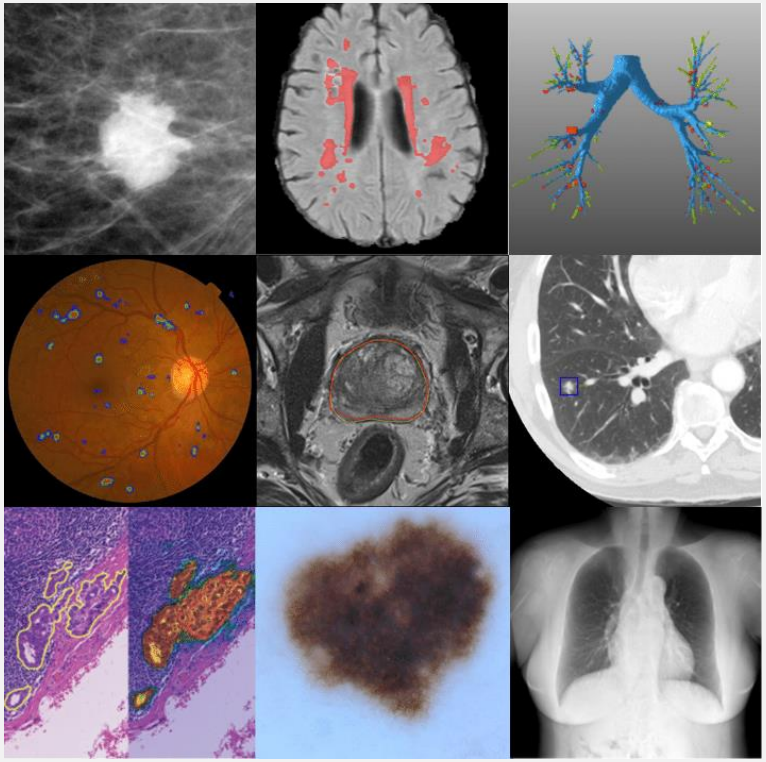
Deep Generative Adversarial Networks:

- Photo-Realistic Single Image Super-Resolution using GAN (SRGAN)

Fig: Results comparison between bicubic, SRResNet optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image for 4x upscaling.[3]

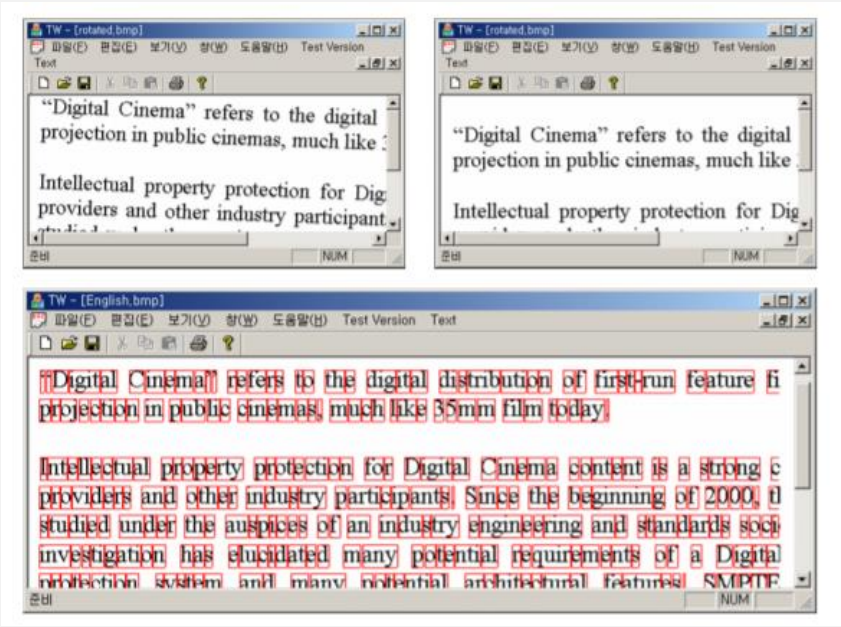
1. 영상 처리의 개요

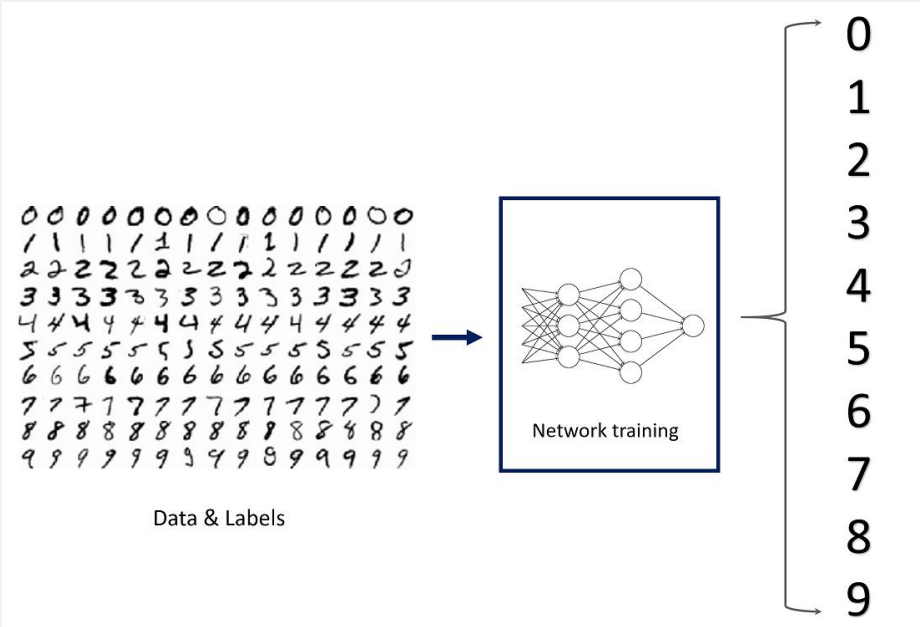
- 의료 영상 처리
  - 정교한 분석이 필요, 정량화 Quantification, 모델링 modeling,
  - 분류 classification



1. 영상 처리의 개요

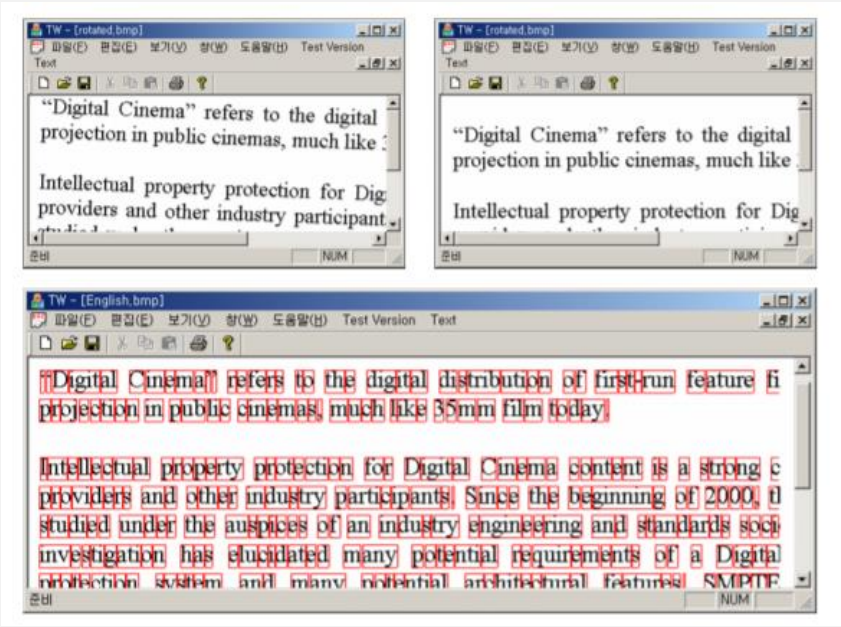
- 문서 처리
  - OCR(Optical Character Recognition)
  - 필기체 인식

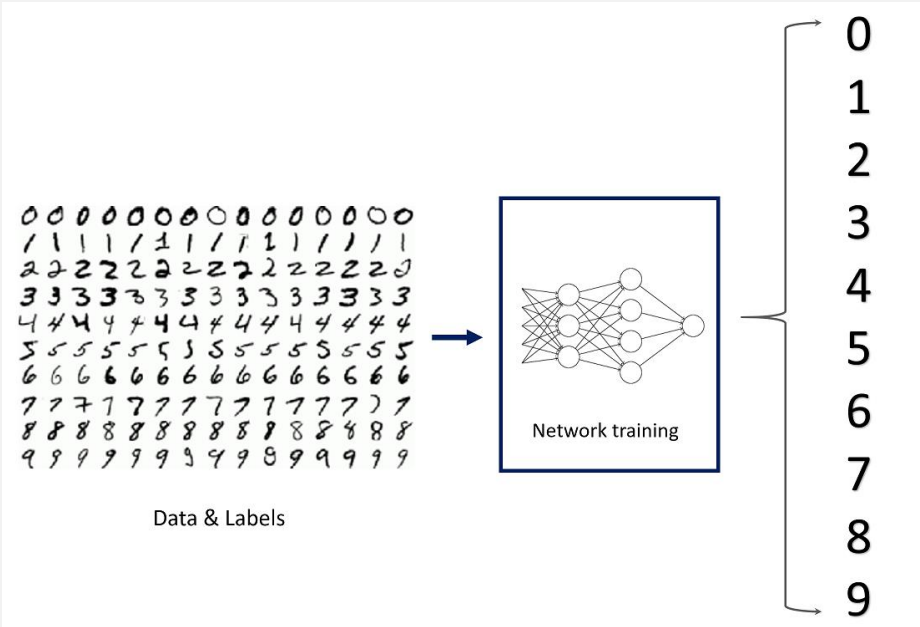




1. 영상 처리의 개요

- 문서 처리
  - OCR(Optical Character Recognition)
  - 필기체 인식





1. 영상 처리의 개요

- Image Captioning
  - 이미지 캡션은 이미지의 텍스트 설명을 생성하는 프로세스
  - 자연어 처리와 컴퓨터 비전을 함께 사용

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

Graphics / Image Data Types

- Terminology
  - Pixel / Pel (畵素): Picture elements in digital images
    - 영상의 기본 단위
  - Bitmap: A two dimensional array of pixel values
  - Image Resolution: The number of pixels in a digital image (1600x1200, 640x480, etc)
  - Aspect Ratio: 한 이미지의 가로/세로 크기 비.
    - 일반적으로 4:3 (황금비) 비가 가장 사람에게 아름답게 보인다 하여 (황금비) 4:3의 비율이 많이 쓰였다.
    - 영화에서와 같은 와이드 스크린이 인기를 얻으며, 16:9의 비를 갖는 와이드 스크린 TV 제품들이 많이 나오고 있으며, HDTV에서도 16:9의 aspect ratio를 채택하였다
  - Half-tone printing: Analog process that uses smaller or larger filled circles of black ink to represent shading. (Widely used in newspaper printing.)

Graphics / Image Data Types

- 1-Bit Images
  - Each pixel is stored as a single bit (1 or 0)
  - Also called binary image / 1-bit monochrome image
- 8-Bit Gray-Level Images
  - 색상 정보가 없이 오직 밝기 정보만으로 구성된 영상
  - Each pixel has a gray value between 0 and 255 (0: Black, 255: White)
  - Each pixel is represented by a single byte

Graphics / Image Data Types

- 24-Bit Color Images
  - Each pixel is represented by three bytes, representing RGB.
  - 256x256x256 colors (16,777,216 colors)
  - A 640x480 24-bit color image would require 921.6 Kbytes w/o compression
  - 24-bit color images are actually stored as 32-bit images with 8 bit alpha channel values.

Graphics / Image Data Types

- 8-Bit Color Images
  - Also called 256 color images
  - Color information is stored as a set of bytes representing index of the color lookup table whose elements are 3-Byte color information.
  - Color Lookup Tables can be stored in the image file header.

1.2. 영상 처리 프로그램의 기초

- 그레이스케일 값의 범위
  - 그레이스케일 영상에서 하나의 픽셀은 0부터 255 사이의 정수 값을 가짐.
  - C/C++ 에서 unsigned char 로 표현 (1 byte)
  - 0 : 가장 어두운 밝기(검정색)
  - 255 : 가장 밝은 밝기(흰색)

1.2. 영상 처리 프로그래밍의 기초

- 그레이스케일 영상에서 픽셀 값 분포의 예

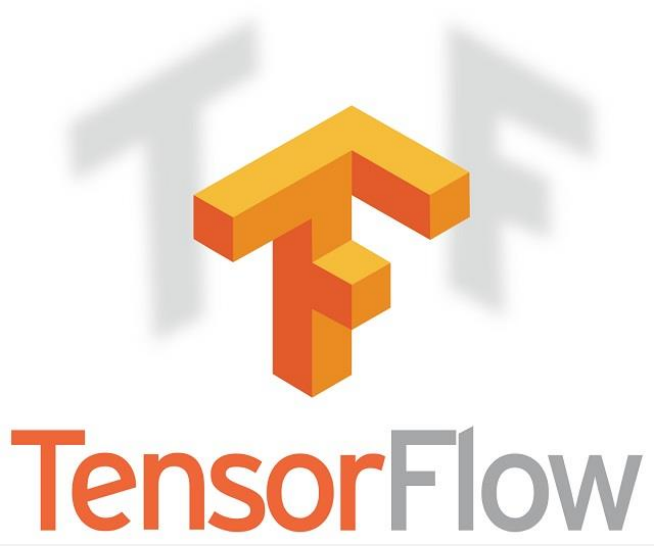
187	187	187	194	197	173	77	25	19	19
190	187	190	191	158	37	15	14	20	20
187	182	180	127	32	16	13	16	14	12
184	186	172	100	20	13	15	18	13	18
186	190	187	127	18	14	15	14	12	10
189	192	192	148	16	15	11	10	10	9
192	195	181	37	13	10	10	10	10	10
189	194	54	14	11	10	10	10	9	8
189	194	19	16	11	11	10	10	9	9
192	88	12	11	11	10	10	10	9	9

2. Tensorflow 구성 및 동작

1. Tensorflow 소개

- Tensorflow는?
  - ✓구글에서 개발하여 공개한 딥러닝/머신러닝을 위한 오픈소스 라이브러리

- 지원언어
  - ✓Python, C++, JAVA, Go (Python 환경에 최적화가 되어 있음)



1. Tensorflow 소개

- 유사 라이브러리
  - ✓토치(Torch) : 페이스북에서 주도적으로 개발한 Lua 언어 기반의 라이브러리
  - ✓파이토치(PyTorch) : 토치의 파이썬 버전
  - ✓CNTK(CogNitive ToolKit) : MS에서 공개

1. Tensorflow 소개

- Tensorflow의 장점
  - ✓전세계적으로 활발한 커뮤니티
  - ✓TensorBoard를 이용한 편리한 시각화
  - ✓단일 데스크톱, 대량의 서버 클러스터, 모바일 등의 광범위한 이식성
  - ✓Keras, TF-Slim, Sonnet 등 다양한 추상화 라이브러리와 혼용해서 사용 가능
  - ✓구글, 딥마인드, 우버, 스냅챗 등 글로벌 기업들이 활발히 사용 중

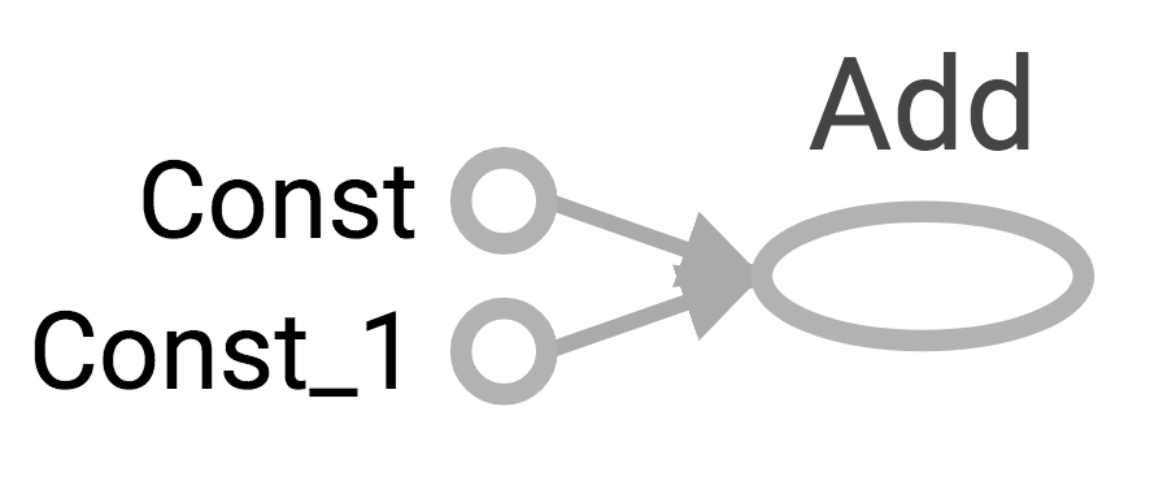
1. Tensorflow 소개

- Tensorflow 응용 분야
  - ✓컴퓨터 비전 : 이미지 분류, 물체 검출 등 (CNN)
  - ✓자연어처리
  - ✓음성인식
  - ✓게임
  - ✓생성모델 (GAN)

Tensorflow 구성 및 동작

- Tensorflow 동작
  - ✓Tensor를 흘려보내면서(flow) 데이터를 처리

- Tensorflow 기본구조
  - ✓Graph : Node와 Edge의 조합
  - ✓Node : Operator, Variable, Constant
  - ✓Edge : 노드간의 연결



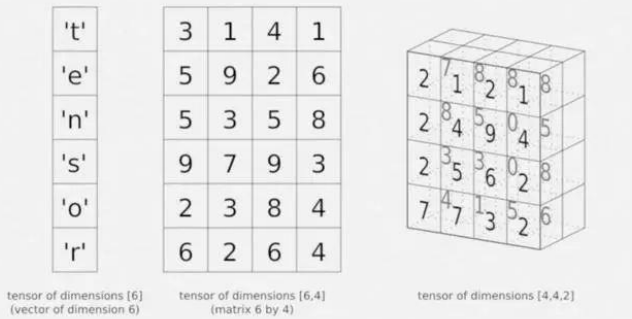
Tensorflow 구성 및 동작

- Tensor란?
  - ✓임의의 차원을 갖는 배열

- Tensor Rank
  - ✓Tensor의 차원

텐서(Tensor)의 Rank

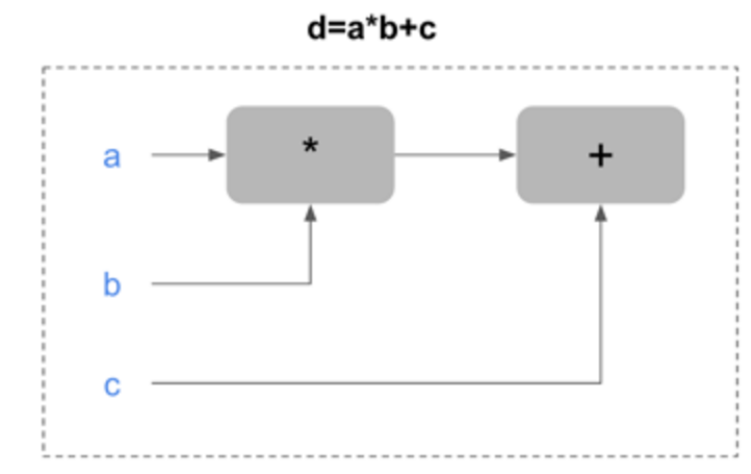
Rank	Math Entity
0	Scalar(magnitude only)
1	Vector(magnitude and direction)
2	Matrix(table of numbers)
3	3-**Tensor**(cube of numbers)



Tensorflow 구성 및 동작

- Tensorflow 실행 과정
  - ✓그래프 생성
  - ✓그래프 실행

- 그래프 구현 예시




```
# 그래프 생성
a = tf.constant(3.0)
b = tf.constant(4.0)
c = tf.constant(5.0)
d = a * b + c
print (d)

# 그래프 실행
sess = tf.Session()
result = sess.run(d)
print (result)
```

출력 : Tensor("add:0", shape=(), dtype=float32)

출력 : 17.0

Tensorflow 구성 및 동작

- 데이터 자료구조
  - ① Constant (상수형)
    - ✓ 변하지 않는 값
  - ② Placeholder
    - ✓ 데이터를 담는 그릇
    - ✓ 학습용 데이터를 위한 타입 (input feeding)
  - ③ Variable (변수형)
    - ✓ 학습을 통해 구해야 하는 값
    - ✓ Weight 등의 파라미터를 위한 타입

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.
DT_UINT8	tf.uint8	8 bits unsigned integer.
DT_UINT16	tf.uint16	16 bits unsigned integer.
DT_STRING	tf.string	Variable length byte arrays. Each element of a Tensor is a byte array.
DT_BOOL	tf.bool	Boolean.
DT_COMPLEX64	tf.complex64	Complex number made of two 32 bits floating points: real and imaginary parts.
DT_COMPLEX128	tf.complex128	Complex number made of two 64 bits floating points: real and imaginary parts.
DT_QINT8	tf.qint8	8 bits signed integer used in quantized Ops.
DT_QINT32	tf.qint32	32 bits signed integer used in quantized Ops.
DT_QUINT8	tf.quint8	8 bits unsigned integer used in quantized Ops.

Tensorflow 구성 및 동작

- Constant (상수형)
  - ✓API 및 파라미터

```
tf.constant(
    value,
    dtype = None,
    shape = None,
    name = 'Const'
    verify_shape=False
)
```

파라미터	역할
value	상수값이며 직접 지정하고나 shape 형태로 채울 값을 지정
dtype	데이터 타입 (예 : tf.float32, tf.int32, tf.bool)
shape	상수 데이터의 형태
name	텐서의 이름 (Optional)
verify_shape	shape를 체크

  - ✓API 사용 예시

```
# Constant 1-D Tensor populated with value list.
tensor = tf.constant([1, 2, 3, 4, 5, 6, 7]) => [1 2 3 4 5 6 7]

# Constant 2-D tensor populated with scalar value -1.
tensor = tf.constant(-1.0, shape=[2, 3]) => [[-1. -1. -1.]
                                             [-1. -1. -1.]]
```

Tensorflow 구성 및 동작

- Placeholder (플레이스 홀더)
  - ✓API 및 파라미터

```
tf.placeholder(
    dtype,
    shape = None,
    name = None
)
```

파라미터	역할
dtype	Feed할 데이터 타입
shape	Feed할 데이터의 형태 (None이면 임의의 차원)
name	텐서의 이름 (Optional)

  - ✓API 사용 예시

```
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
```

Tensorflow 구성 및 동작

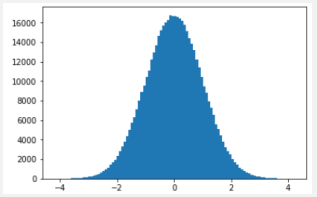
- Variable (변수형)
  - ✓API 및 파라미터

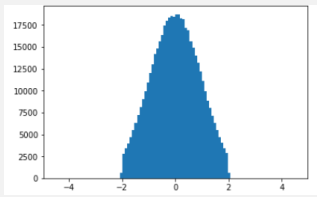
```
tf.Variable(
    initial_value = None,
    trainable = True,
    name = None,
)
```

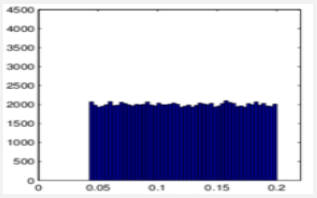
파라미터	역할
initial_value	변수의 초기값이며 변수의 shape를 포함한 상태로 지정
trainable	트레이닝 가능 여부 (False면 파라미터가 업데이트 안됨)
name	텐서의 이름 (Optional)

  - ✓API 사용 예시

```
W = tf.Variable(tf.random_normal(shape=[1], name='w'))
b = tf.Variable(tf.random_normal(shape=[1], name='b'))
```





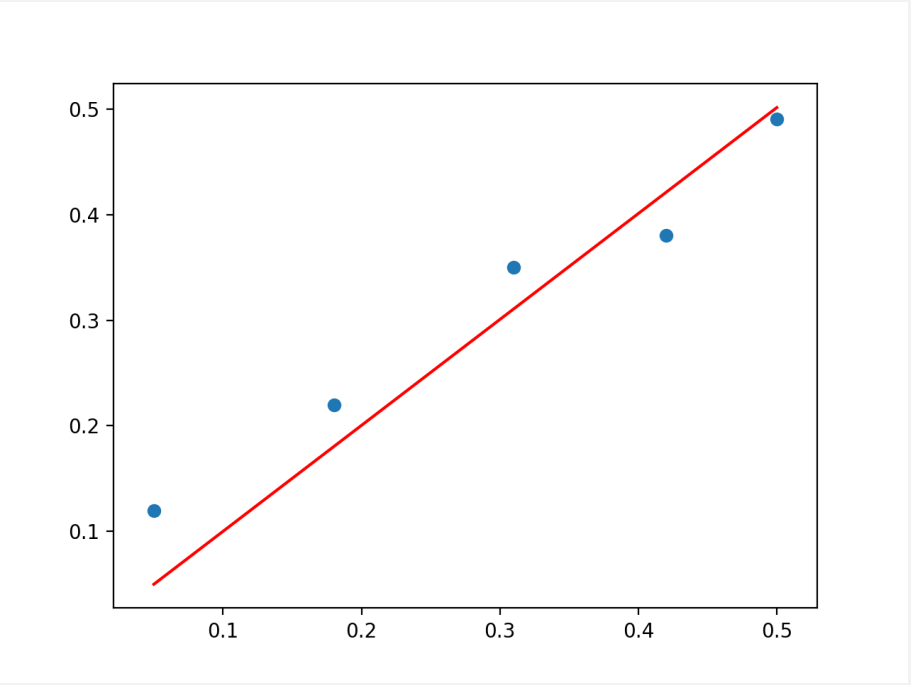


  - ✓초기화 종류

종류	설명
tf.random_normal	가우시안 분포(정규 분포)에서 임의의 값을 추출
tf.truncated_normal	Truncated normal 분표(끝 부분이 잘린 정규분포 모양)에서 임의의 값 추출
tf.random_uniform	균등 분포에서 임의의 값을 추출
tf.constant	특정한 상수값으로 지정한 행렬로 채움
tf.zeros	모두 0으로 채움
tf.ones	모두 1로 채움

3. 예제: Linear Regression

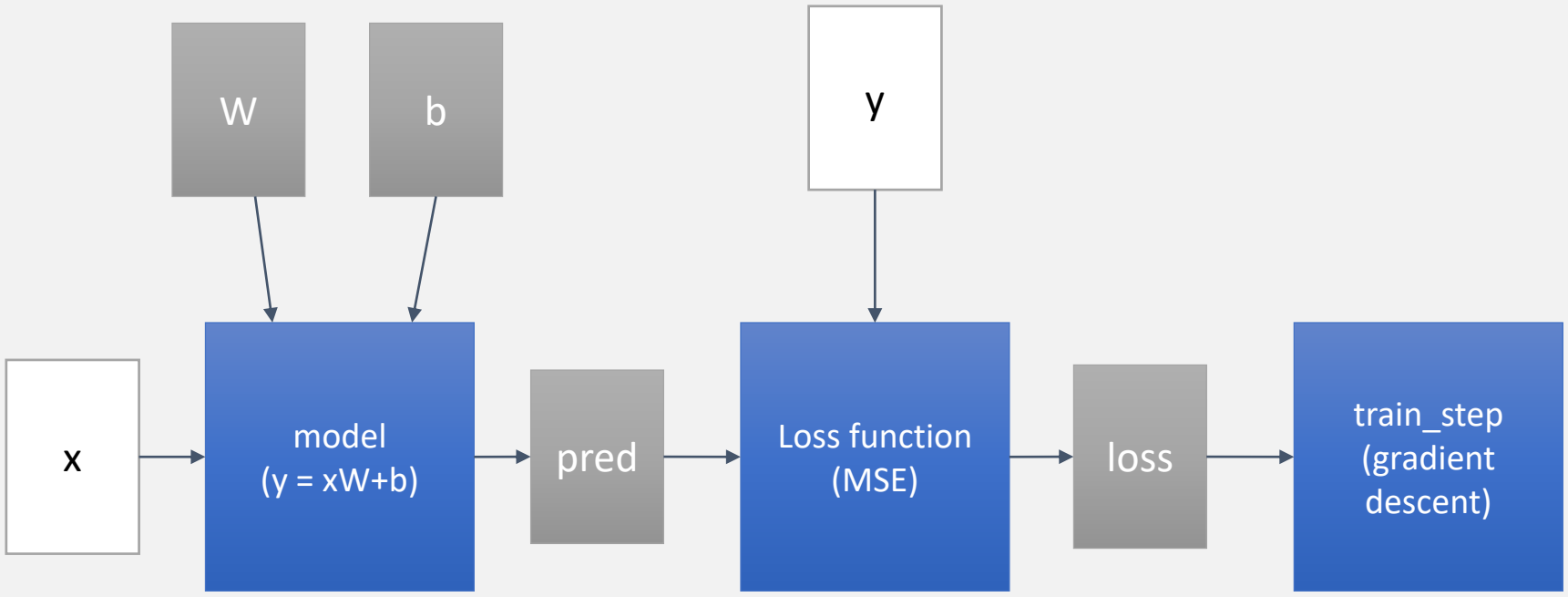
예제: Linear Regression



- Linear Regression (선형 회귀)란?
  - ✓ 데이터들의 1차 함수(직선) 상관관계를 모델링
- 학습 목표 및 방법
  - ✓ 데이터들을 통해 y = xW+b의 W와 b를 찾는 것이 목표
  - ✓ Optimizer : Gradient Descent
  - ✓ Loss Function : MSE (Mean Squared Error)

예제: Linear Regression

- 데이터 연산의 흐름

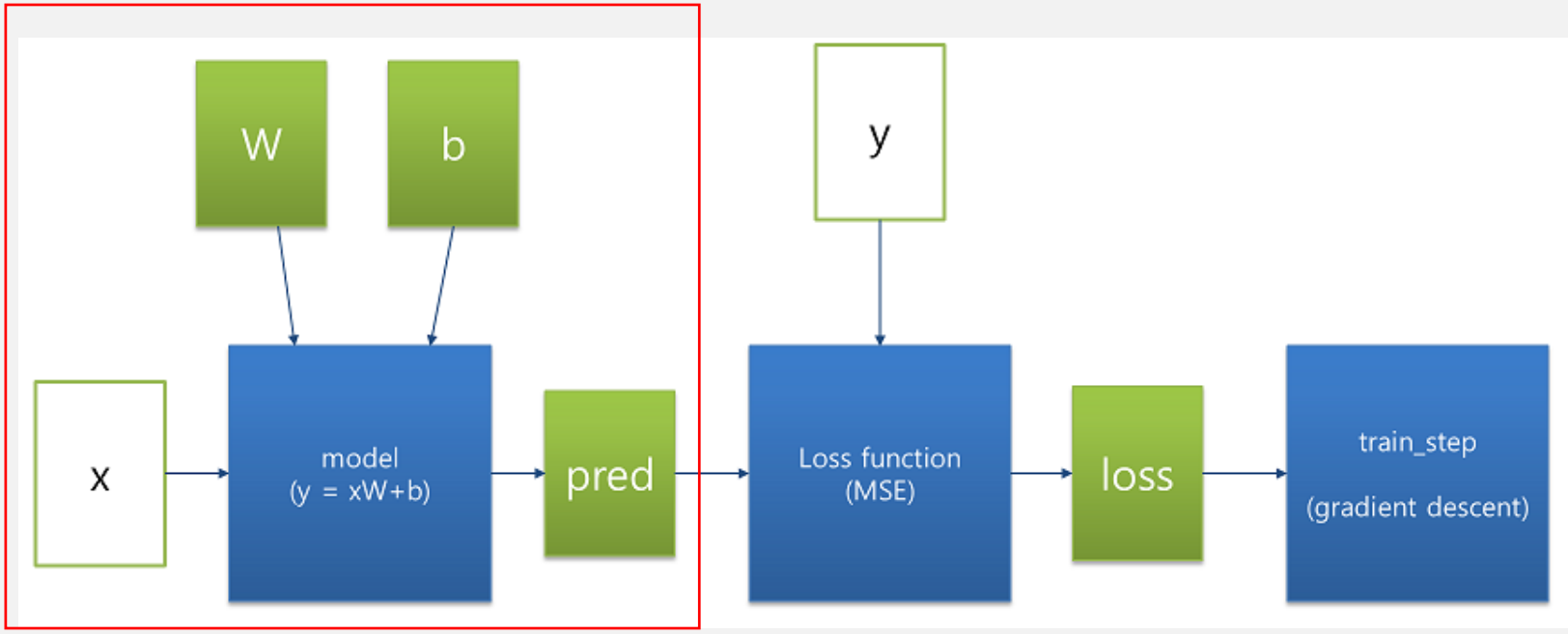


예제: Linear Regression

- 입력 및 모델 정의

```
W = tf.Variable(tf.random_normal(shape=[1], name='w'))
b = tf.Variable(tf.random_normal(shape=[1], name='b'))
x = tf.placeholder(tf.float32)

pred = W*x + b
```



예제: Linear Regression

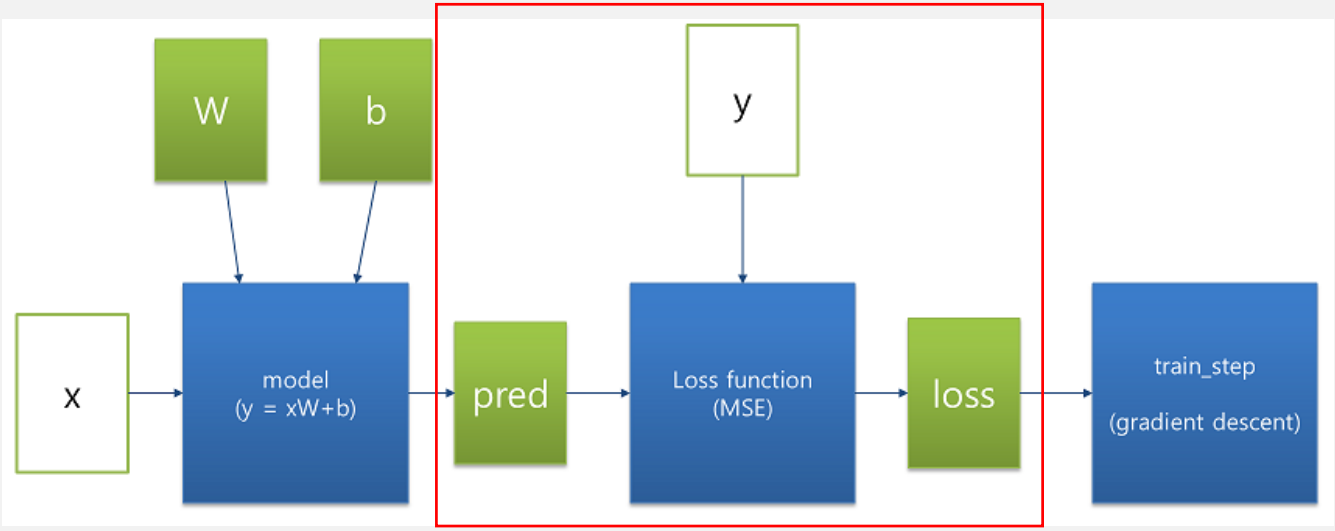
- 정답과 Cost Function 정의

```
y = tf.placeholder(tf.float32)

loss = tf.reduce_mean(tf.square(pred - y))
```

✓ Cost Function = MSE (Mean Squared Error)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$



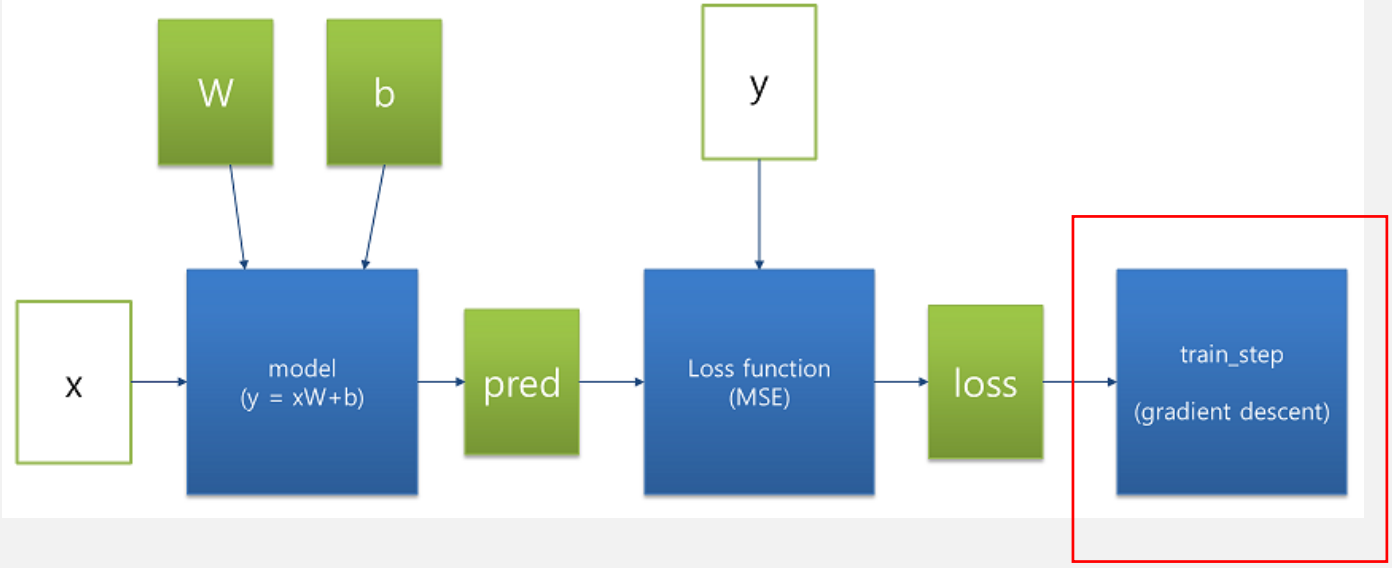
예제: Linear Regression

- Optimizer 정의

```
optimizer = tf.train.GradientDescentOptimizer(0.01)

train_step = optimizer.minimize(loss)
```

  - ✓ 학습 방법 : Gradient Descent
  - ✓ Learning Rate : 0.01

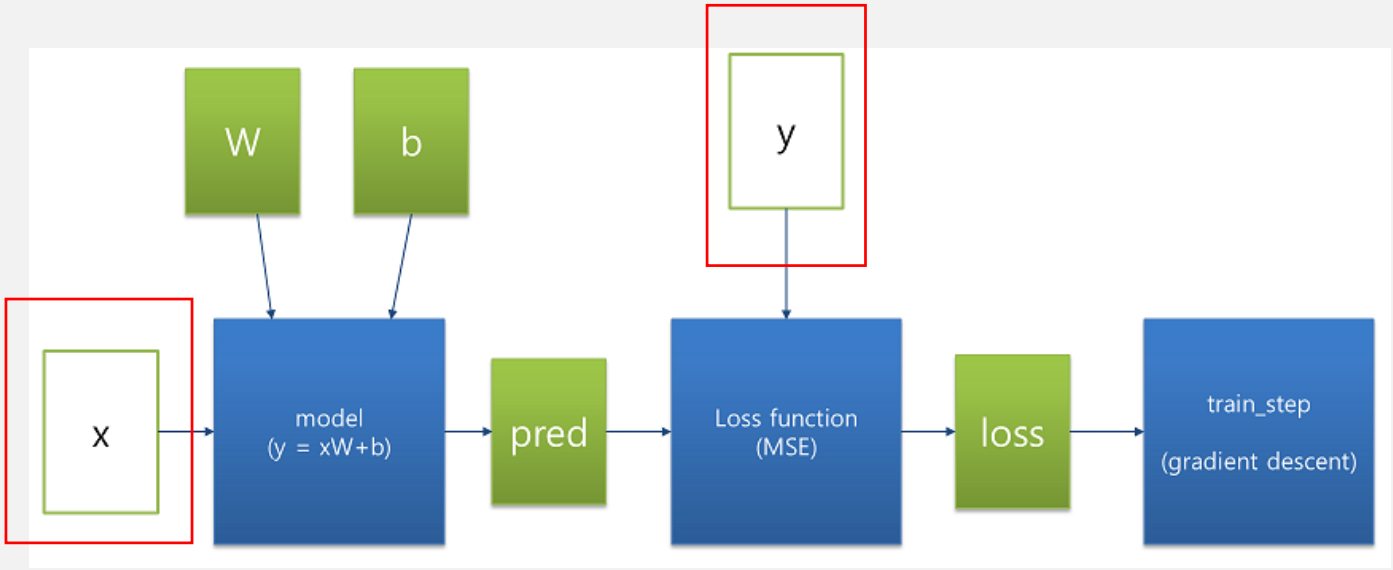


예제: Linear Regression

- 학습셋 구성

```
# y = 2x + 1
x_train = [1,2,3,4]
y_train = [3,5,7,9]
```

✓ 목표 모델 : y = 2x + 1

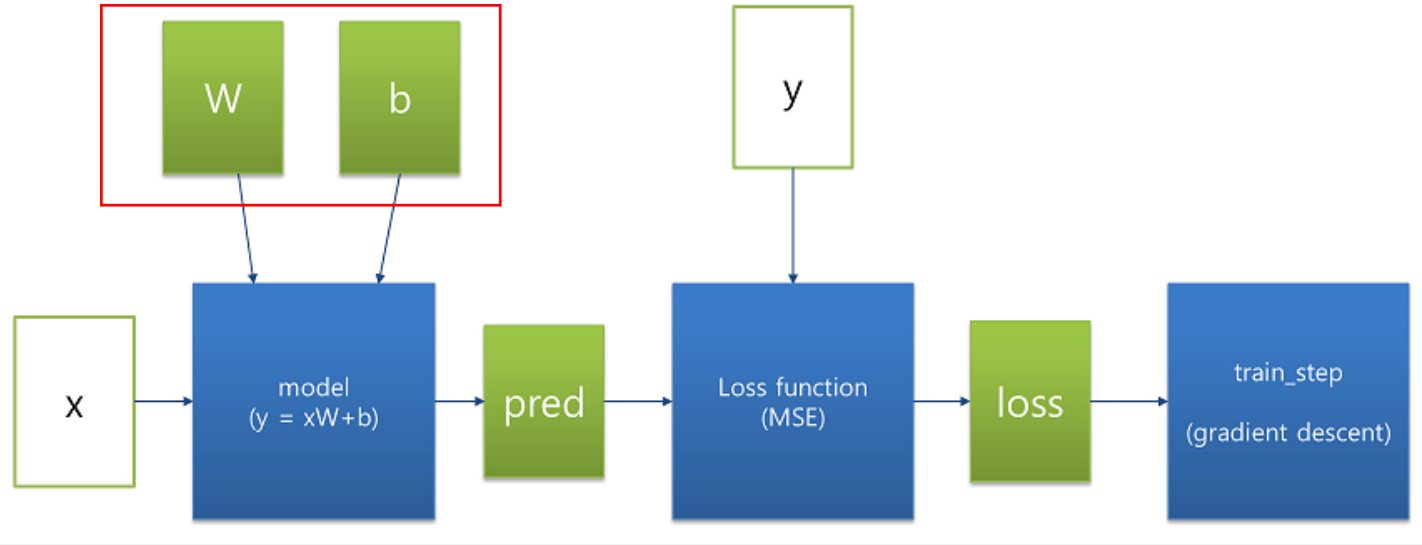


예제: Linear Regression

- 변수 초기화

```
sess = tf.Session()
result = sess.run(tf.global_variables_initializer())
```

✓ 파라미터 초기화 : w, b



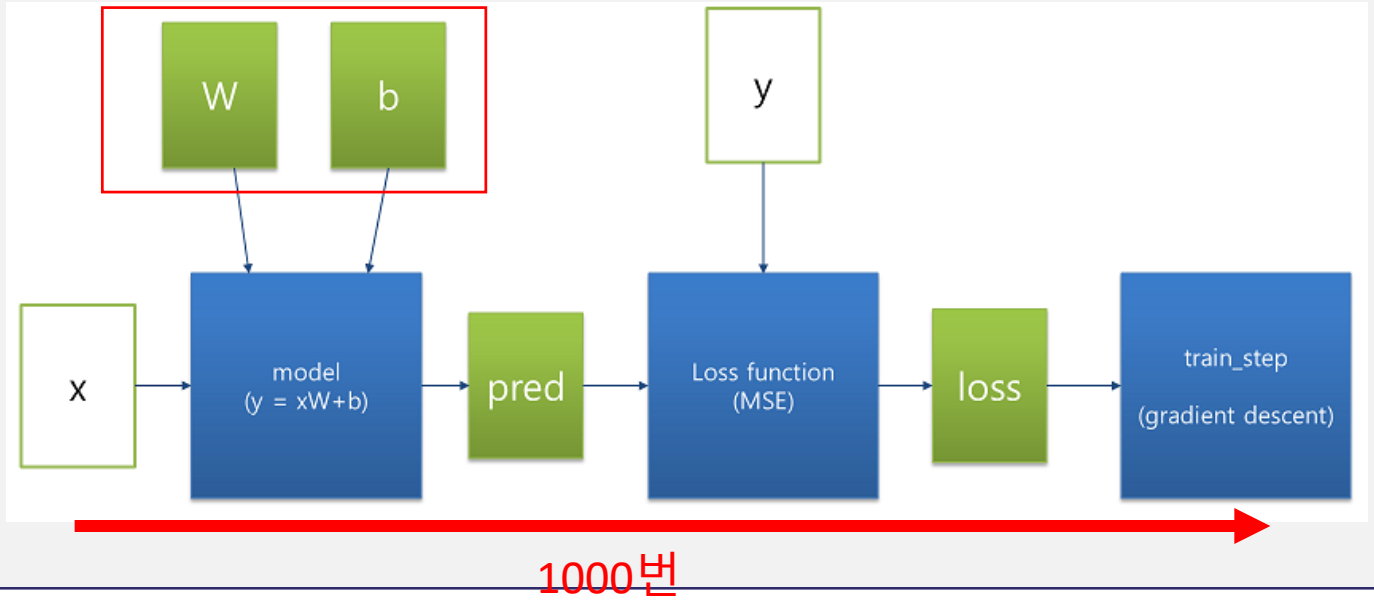
예제: Linear Regression

- 학습 진행

```
for i in range(1000):
    sess.run(train_step, feed_dict={x: x_train, y: y_train})
```

✓ 학습 진행

✓ Epoch : 1000



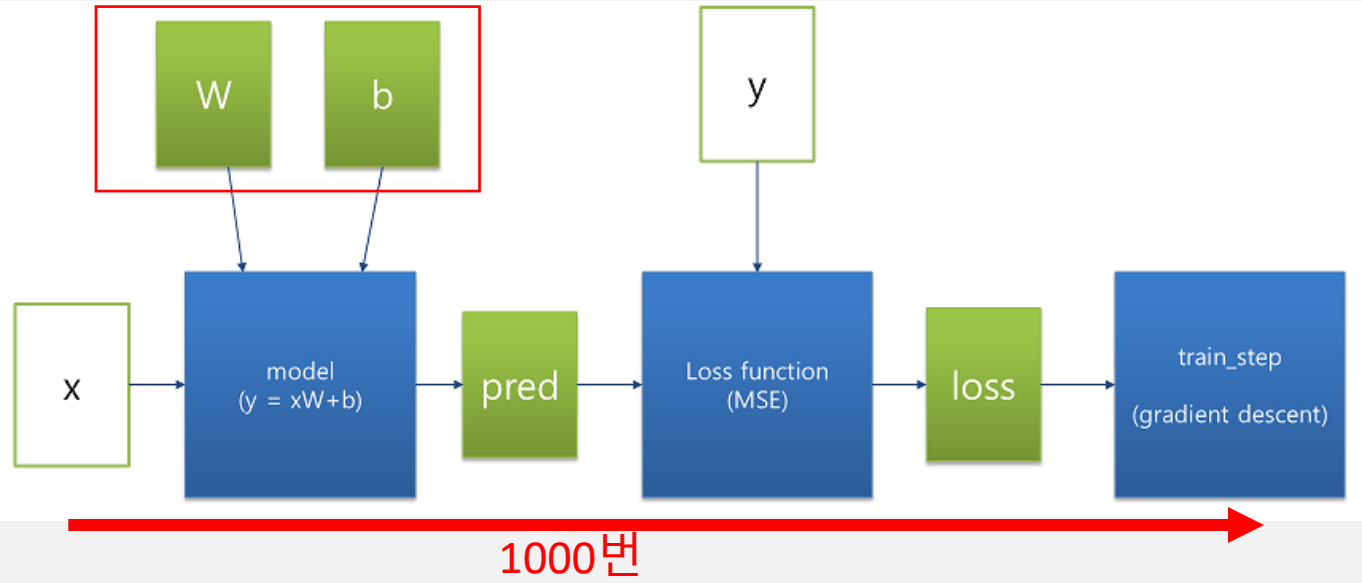
예제: Linear Regression

- 학습 진행

```
for i in range(1000):
    sess.run(train_step, feed_dict={x: x_train, y: y_train})
```

✓ 학습 진행

✓ Epoch : 1000



예제: Linear Regression

- 모델 테스트

```
x_test = [3, 5, 5, 6]

print (sess.run(pred, feed_dict={x: x_test}))
```

- 결과

```
2019-03-27 17:16:13.734138: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:893] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2019-03-27 17:16:13.734411: I tensorflow/core/common_runtime/gpu/gpu_device.cc:955] Found device 0 with properties:
name: GeForce GTX 980
major: 5 minor: 2 memoryClockRate (GHz) 1.329
pciBusID 0000:01:00.0
Total memory: 3.94GiB
Free memory: 3.87GiB
2019-03-27 17:16:13.734435: I tensorflow/core/common_runtime/gpu/gpu_device.cc:976] DMA: 0
2019-03-27 17:16:13.734439: I tensorflow/core/common_runtime/gpu/gpu_device.cc:986] 0:   Y
2019-03-27 17:16:13.734443: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1045] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX 980, pci bus id: 0000:01:00.0)
[ 7.000265  11.0091095 11.0091095 13.013533 ]

Process finished with exit code 0
```