"Hello, world!"

Lecture 6.
Training Neural Networks I

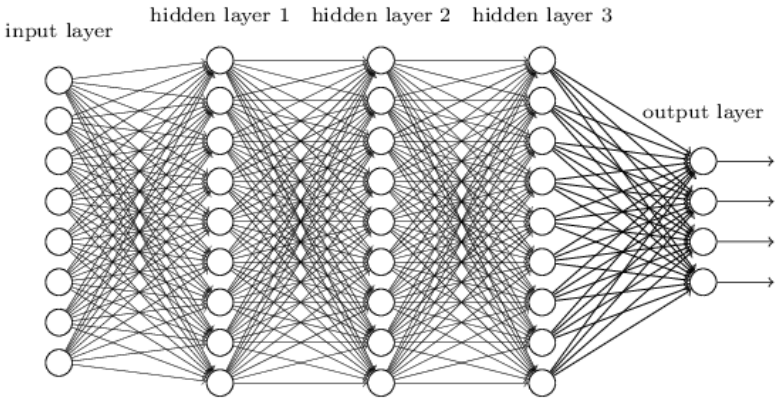
Review

1. Back to Image Classification

2. Convolutional Neural Network

Review

MLP



10%

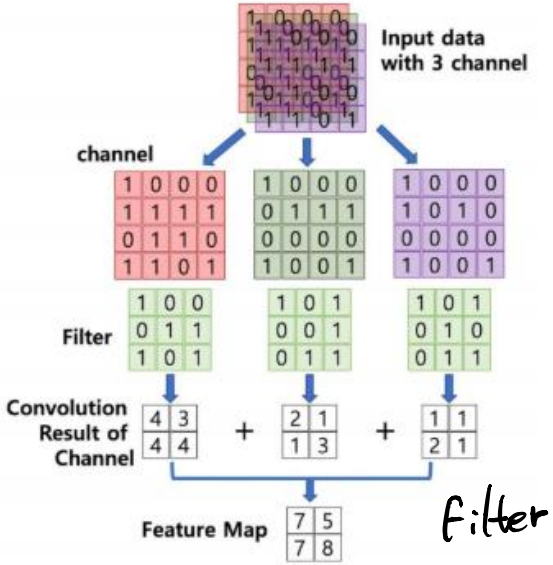
Review

CNN

$$O = \frac{I + 2Pa - F}{S} + 1$$

$O \leq I$

$O = Output\ size, \quad I = Input\ size, \quad Pa = Padding,$
$F = Filter\ size, \quad S = Stride$



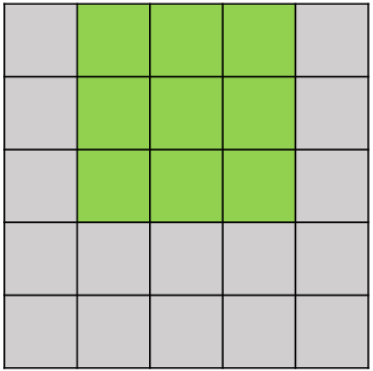
0	0	0	0	0	0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

Padding

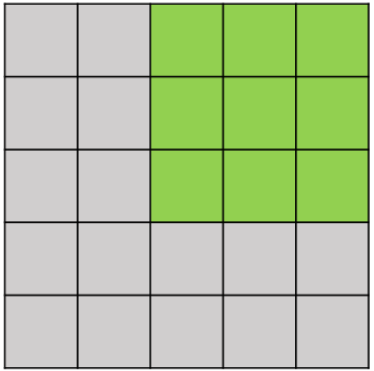
Review

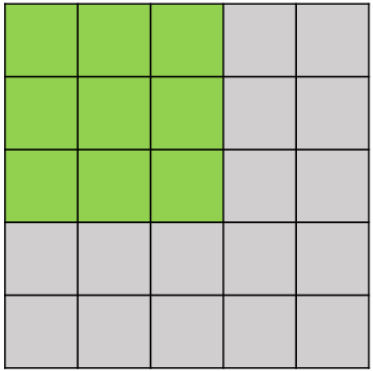
How much does a filter move per slide? **Stride**

Stride 1

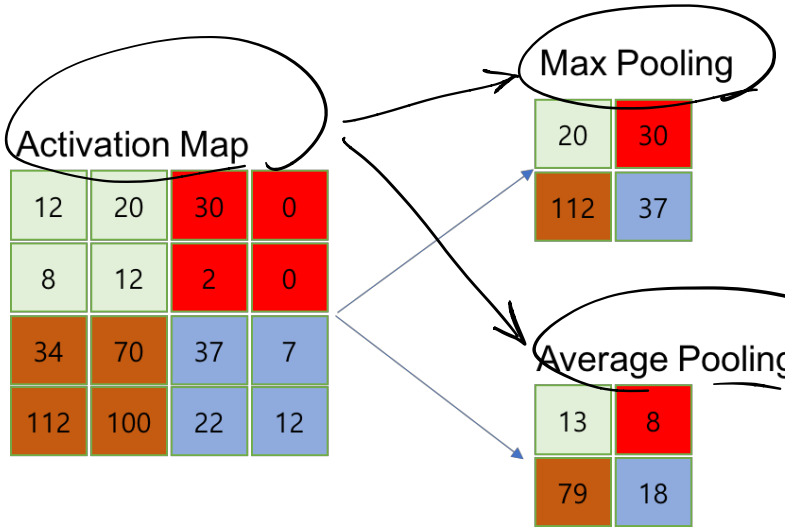


Stride 2





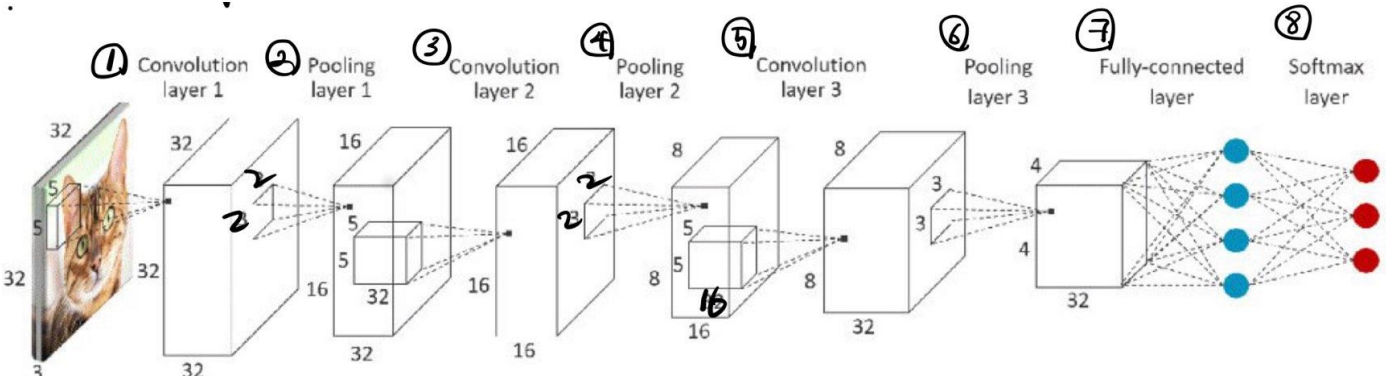
Review



$$O = \frac{I}{Po}$$

$O = Output\ size,$ $I = Input\ size,$
$Po = Pooling$

Review


① Convolution layer 1
② Pooling layer 1
③ Convolution layer 2
④ Pooling layer 2
⑤ Convolution layer 3
⑥ Pooling layer 3
⑦ Fully-connected layer
⑧ Softmax layer

Today's Content

1. Basic Things for Training
   - GPU (+TPU)
   - Activation functions
   - Weight Initialization
   - Data Normalization
   - Batch Normalization
   - Hyperparameter Search

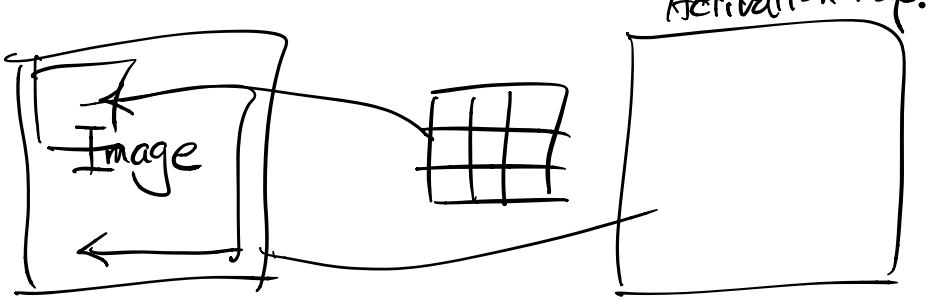
2. Change Optimization Process
   - Gradient based method
   - Regularization

GPU란? ⟷ CPU. Central Process Unit

Graphic Process Unit의 약자

4352

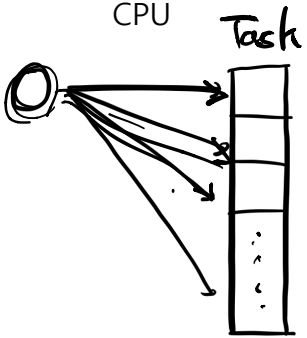
GPU란?



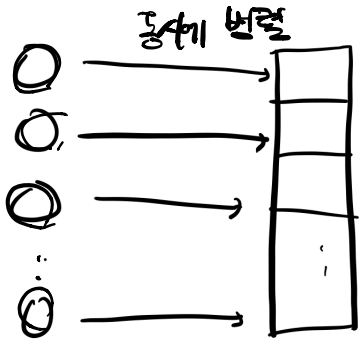
GPU는 병렬 계산에 특화 되어있음

2,4,8

CPU



GPU



1000s.
4000s.

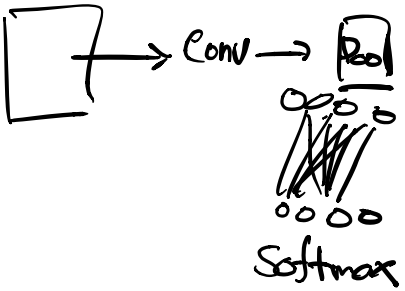
3000s → 1h.

GPU란?

100번 Epoch

XL Net

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 64)        832
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 10)                163850
=================================================================
Total params: 164,682
Trainable params: 164,682
Non-trainable params: 0
_________________________________________________________________
```



CPU

```
[9]  1 model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
     2              loss='categorical_crossentropy',
     3              metrics=['accuracy'])
     4
     5 hist = model.fit(train_data, train_labels, epochs=1, batch_size=32, validation_data=(val_data, val_labels))
```

```
Train on 40000 samples, validate on 10000 samples
40000/40000 [==============================] - 47s 1ms/sample - loss: 1.6668 - acc: 0.4205 - val_loss: 1.6619 - val_acc: 0.4252
```

GPU

```
[11] 1 model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
     2              loss='categorical_crossentropy',
     3              metrics=['accuracy'])
     4
     5 hist = model.fit(train_data, train_labels, epochs=1, batch_size=32, validation_data=(val_data, val_labels))
```

```
Train on 40000 samples, validate on 10000 samples
40000/40000 [==============================] - 10s 240us/sample - loss: 1.9463 - acc: 0.3175 - val_loss: 1.8910 - val_acc: 0.3235
```

47s.

10s

None

Colaboratory ← Platform

* TPU란? Tensorflow ← Software

TPU ← Hardware.

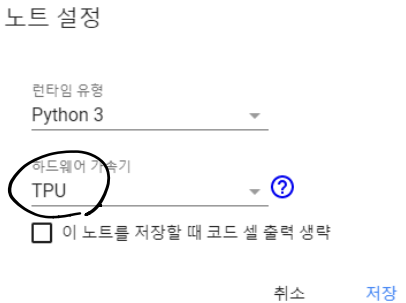
알파고

구글에서 개발한 Tensor Process Unit

뉴럴 네트워크 연산에 대해 GPU보다 15~30배 빠르다..!

클라우드로 TPU를 제공한다.

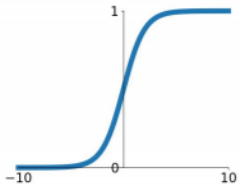
런타임 유형 변경



Activation Function

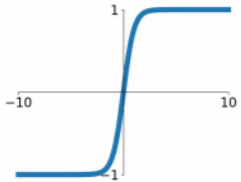
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



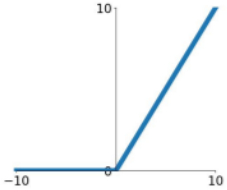
tanh

$$\tanh(x)$$



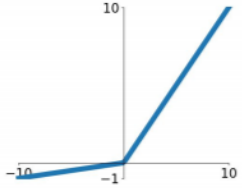
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

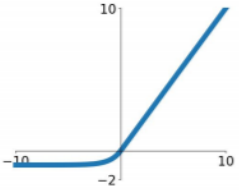


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation Function

[0, 1]

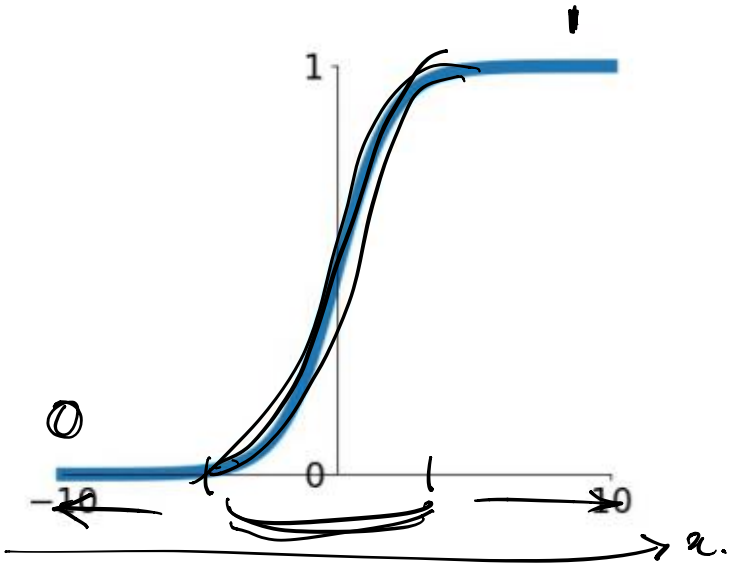
Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

초기 머신러닝에 적용한 활성화 함수

문제점 3가지

1. Kill gradient
2. Not zero-centered
3. exp() is compute expensive

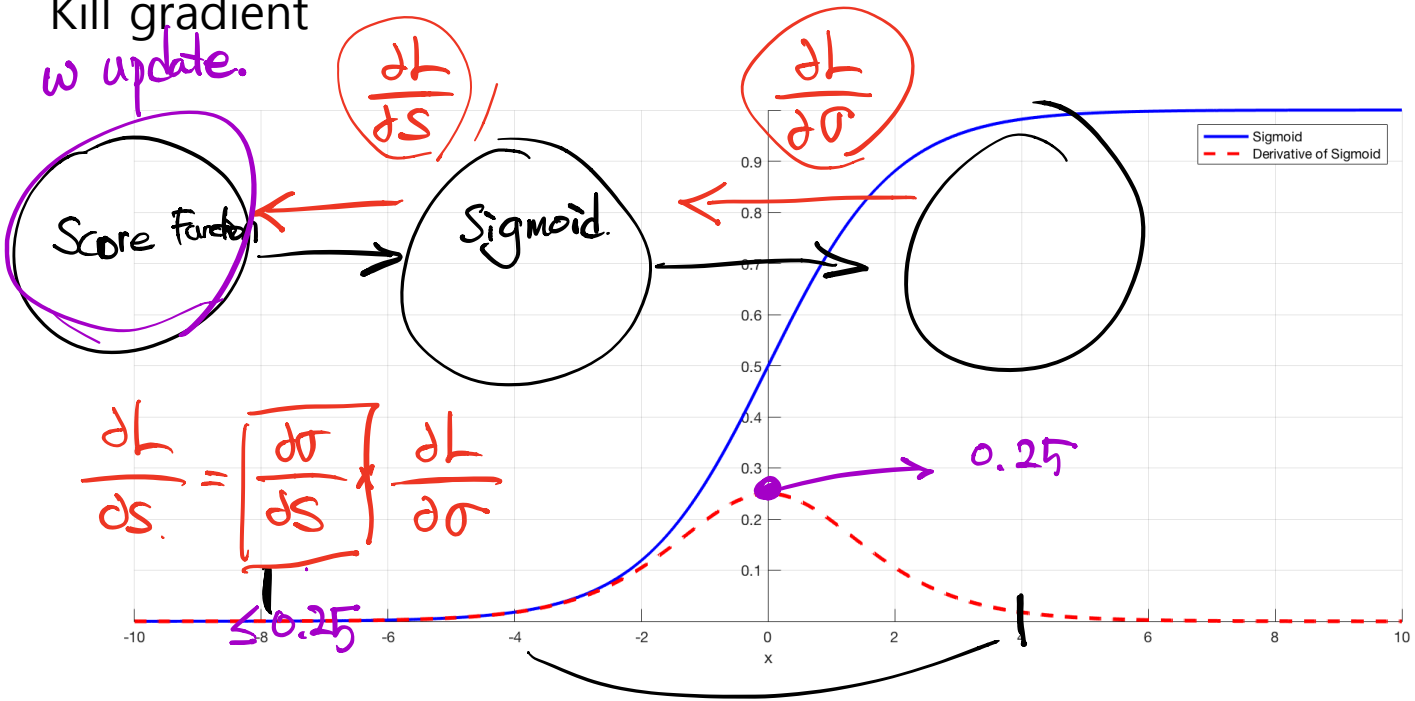


$S(z) = Wz + b$

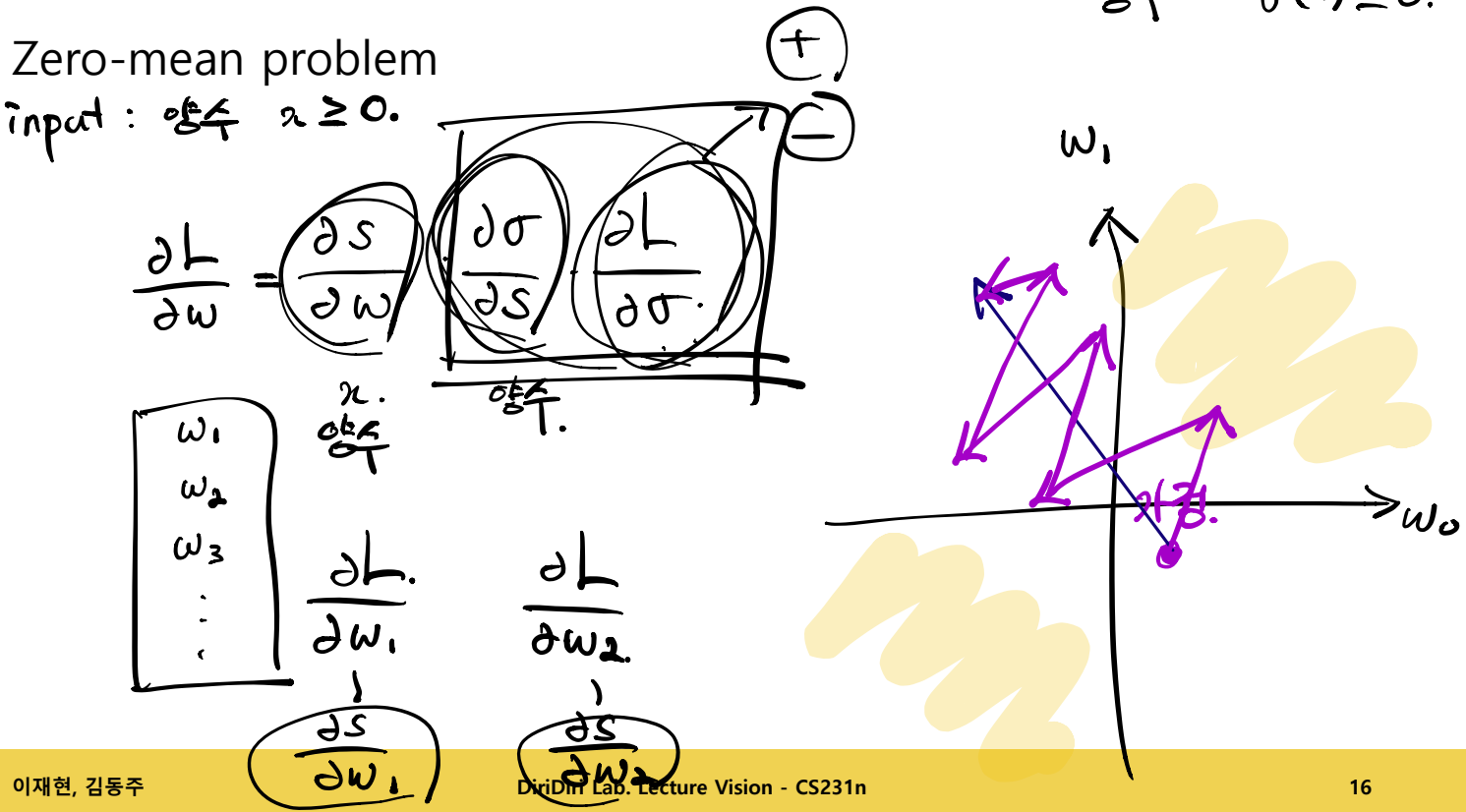
$\sigma(z)$

Kill gradient

w update.



Zero-mean problem



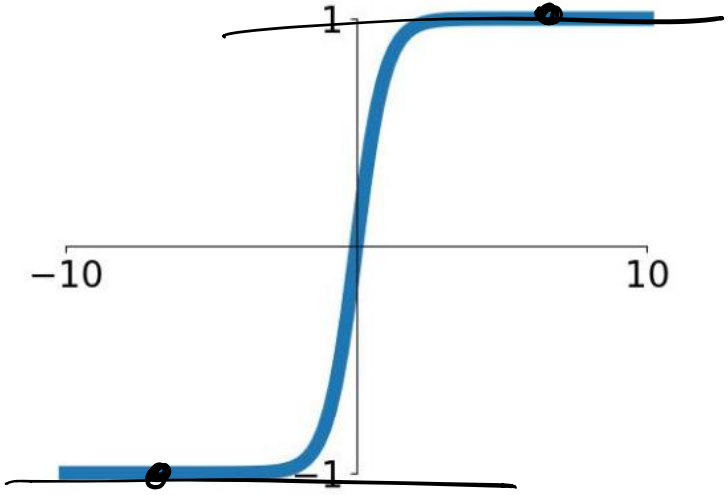
[0,1] [-1,1]

Activation Function

tanh

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- -1부터 1사이
- Zero-centered!
- Still kill gradient..



Activation Function

ReLU

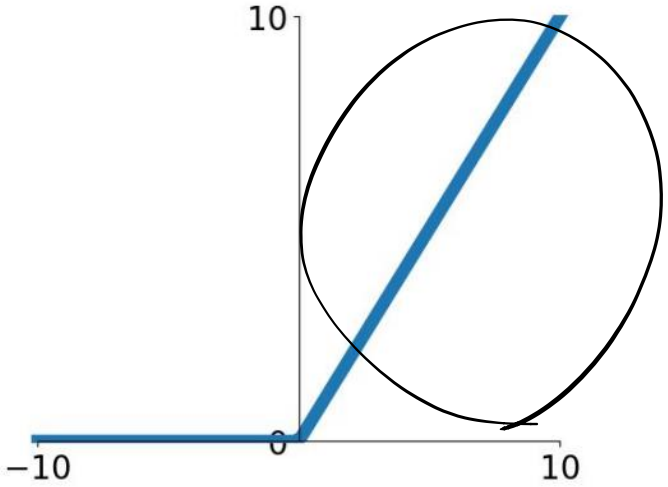
2012. AlexNet.

$$f(x) = max(0, x)$$

가장 널리 사용되는 활성화함수

장점 3가지

1. Not kill gradient (in +region)
2. 계산이 편하고 빠름
3. 훨씬 빠르게 수렴 (6배)



Activation Function

$f(x) \geq 0$

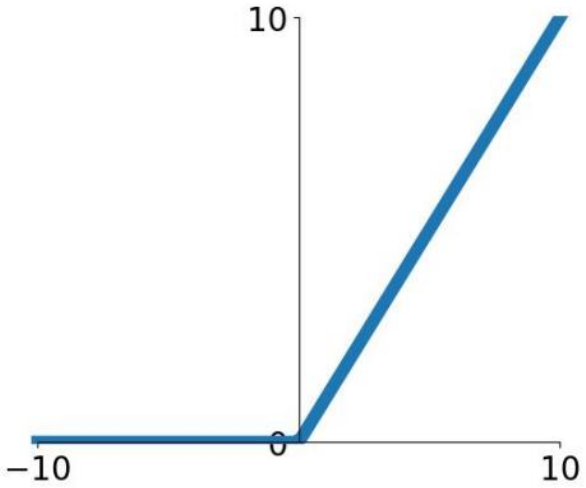
Data → Zero-Centered

ReLU

$$f(x) = max(0, x)$$

단점 2가지

1. Not zero-centered
2. Dead ReLU problem



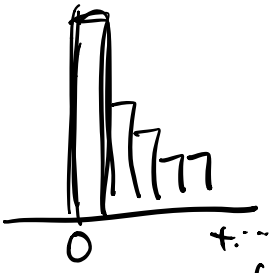
f: ReLU, (s): score function
↳ wx+b.

'Dead' ReLU problem

$$\frac{\partial L}{\partial w} = \frac{\partial s}{\partial w} \cdot \frac{\partial f}{\partial s} \cdot \frac{\partial L}{\partial f} = 0$$

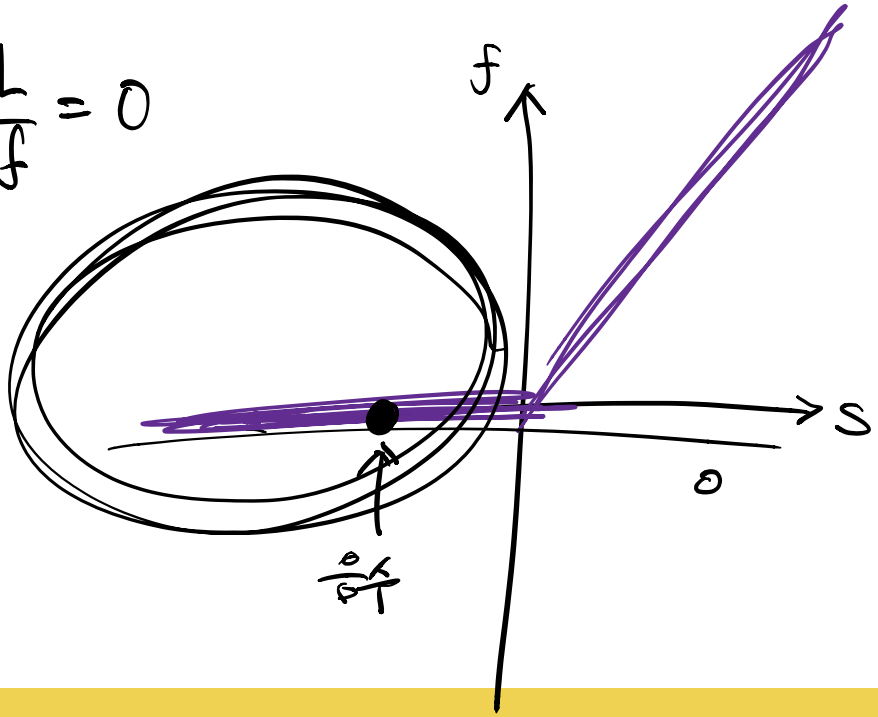
$\frac{\partial s}{\partial w}$ → x.

$\frac{\partial f}{\partial s}$ → 0



w는 고정.

x



Activation Function

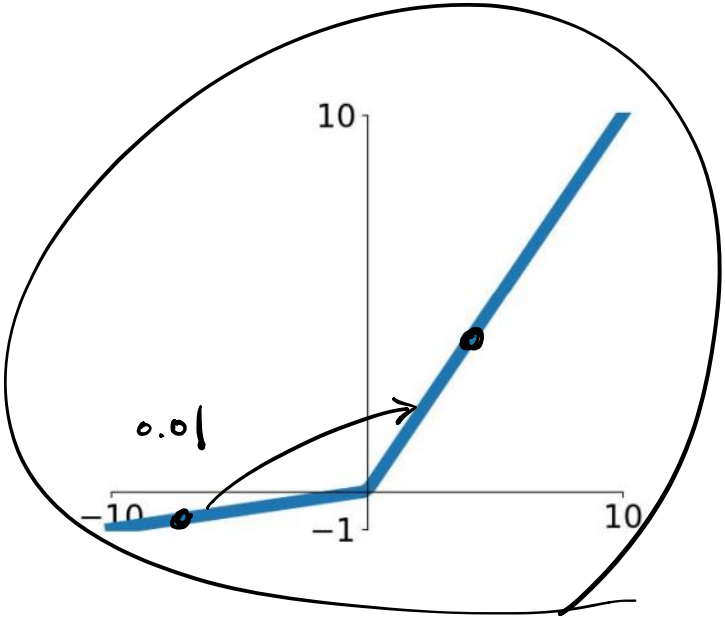
Leaky ReLU

$$f(x) = max(0.01x, x)$$

- ReLU의 변형
- Not die!

PReLU

$$f(x) = max(\alpha x, x)$$

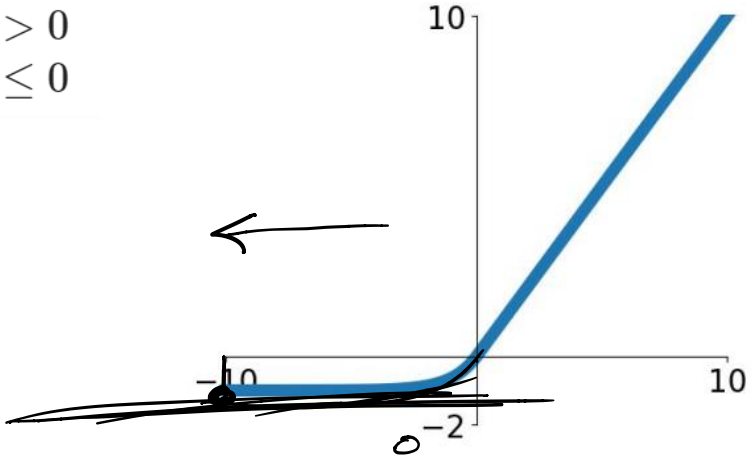


Activation Function

ELU

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{if } x \leq 0 \end{cases}$$

- Robustness to noise
- Compute expensive
- ReLU 와 Leaky ReLU 의 중간

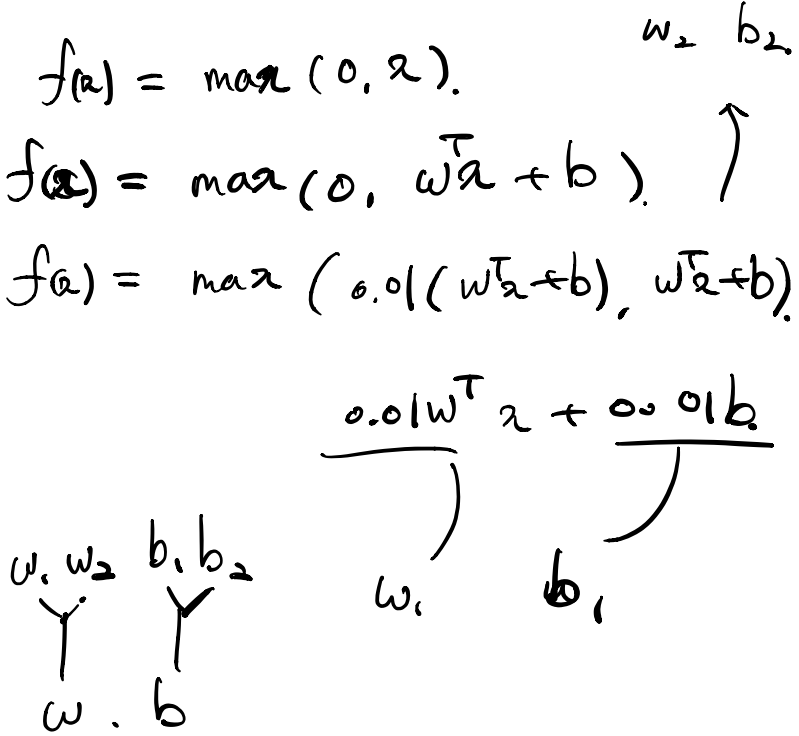


Activation Function

Maxout

$$max(w_1^T x + b_1, w_2^T x + b_2)$$

- "Neuron"
- Nonlinearity by max
- Generalize ReLUs
- Double # of parameters : **Big Problem**



Summary of activation function

그래서 뭐 쓸까요?

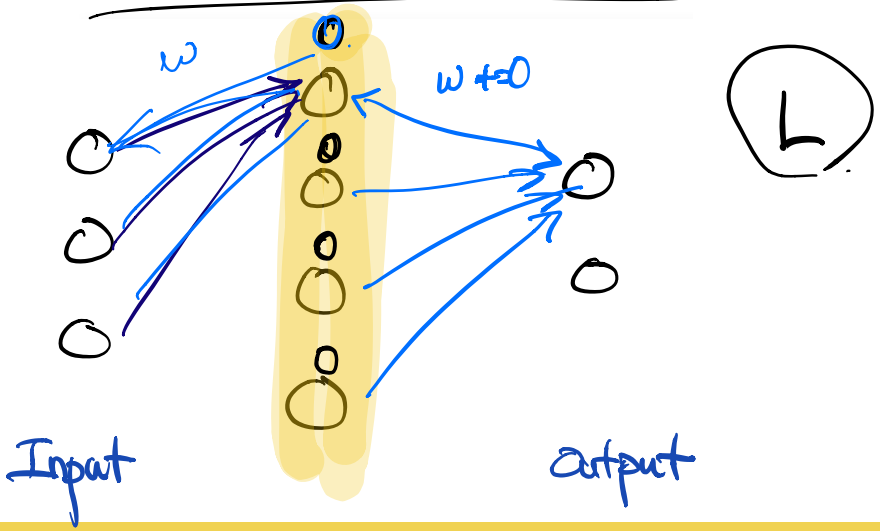
1. 일단 ReLU learning rate 주의!
2. Leaky ReLU / ELU / PReLU / Maxout 은 실험정신으로 사용
3. tanh도 시도해볼만 하지만 성능향상을 기대하긴 어려움
4. Sigmoid는 pass

Weight Initialization

Weight initialzation은 어떻게 할까?

Idea 1. weight = 0

$$np.zeros(size = (D, H))$$



mean = 0

stddev = 0.01

Idea 2. weight = small random distribution

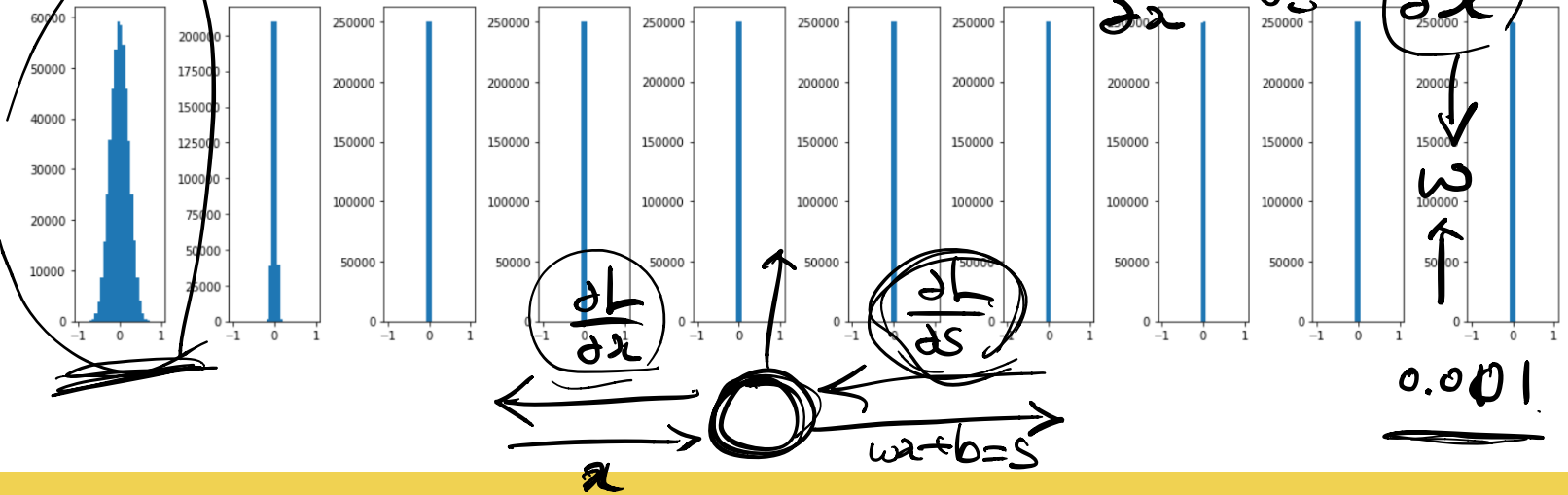
$$0.01 * np.random.randn(size = (D, H))$$

```
W = np.random.randn(fan_in, fan_out) * 0.01
```

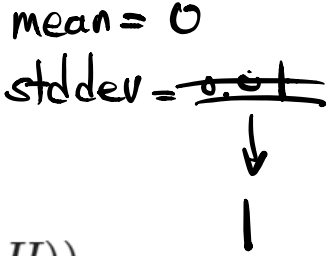
Input

Output

Idea 2. weight = small random distribution



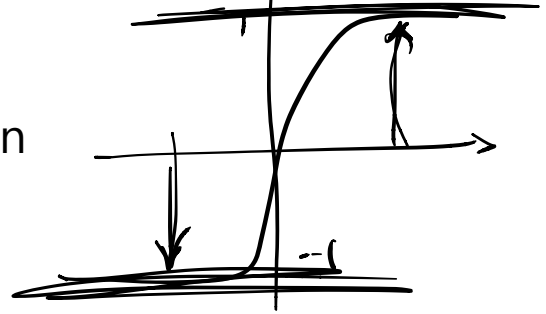
Idea 3. weight = big random distribution



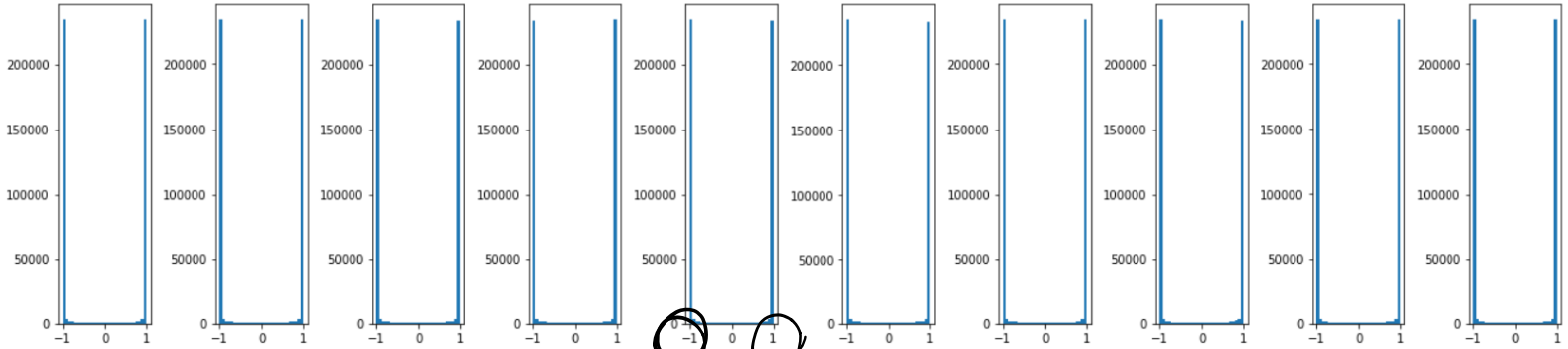
$$1 * np.random.randn(size = (D, H))$$

```
W = np.random.randn(fan_in, fan_out) * 1.0
```

tanh



Idea 3. weight = big random distribution

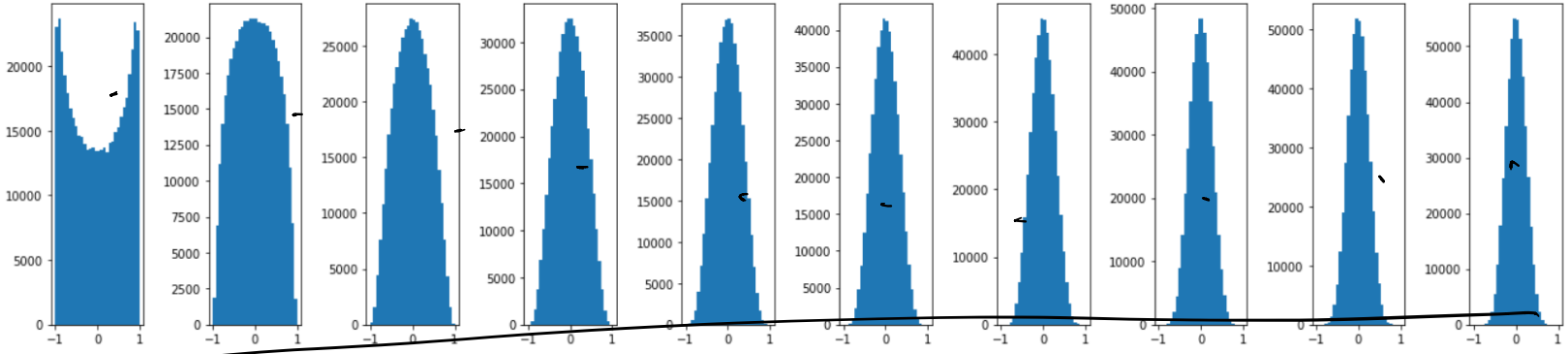


-1 or 1

Xavier initialize

$$np.random.randn(size = (D, H))/np.sqrt(D)$$

```
W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in)
```



Preview on Next Class(es)

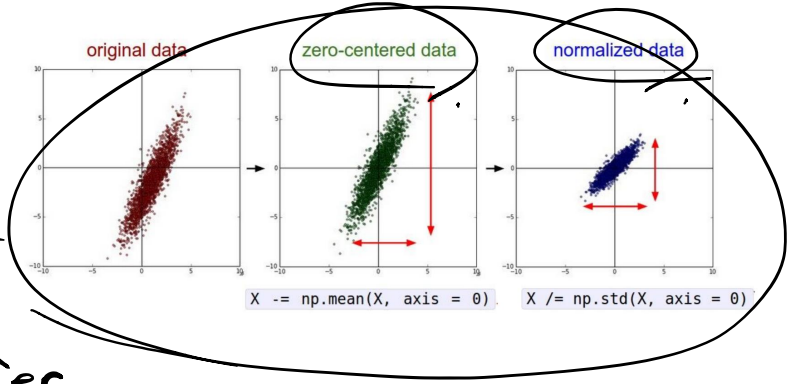
CPU < GPU ≤ TPU

~~Sigmoid~~ tanh ReLU

Not zero-Centered

1. Basic Things for Training
   - ~~GPU (+ TPU)~~
   - ~~Activation functions~~
   - ~~Weight Initialization~~
   - Data Normalization
   - Batch Normalization
   - Hyperparameter Search

0 에 가까운



2. Change Optimization Process
   - Gradient based method
   - Regularization

Xavier