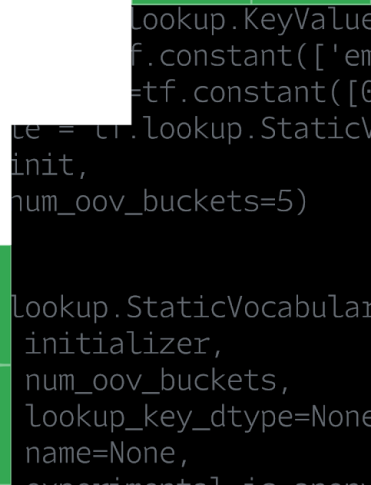
Google Developer Student Clubs

ML/DL Study W02

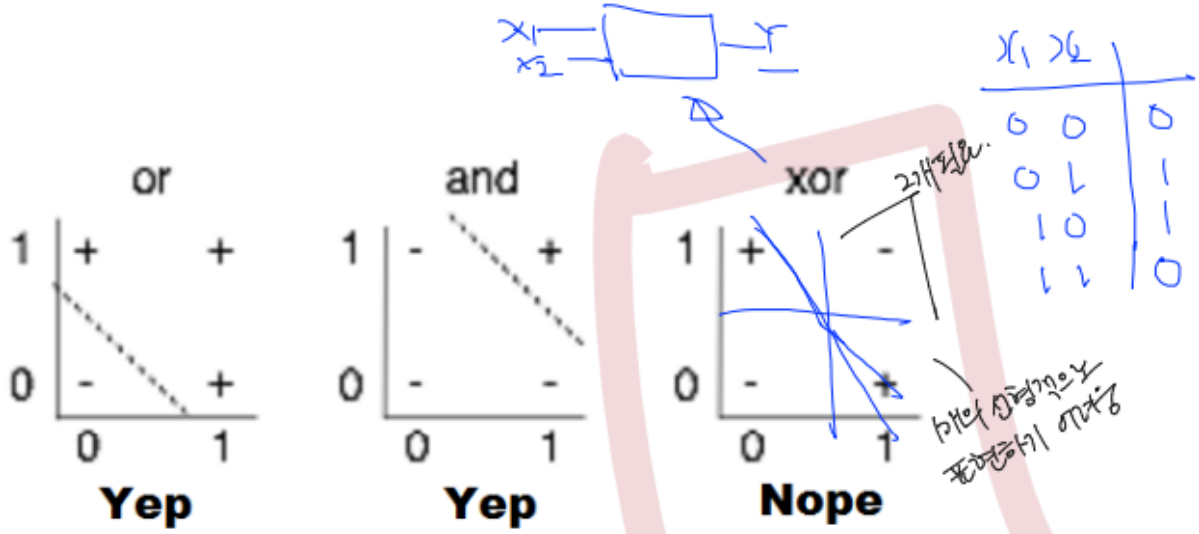
- XOR – Neural Networks
- Forward/Back Propagation
- Sigmoid VS Relu
- Xavier VS He
- Drop out
- Batch Normalization



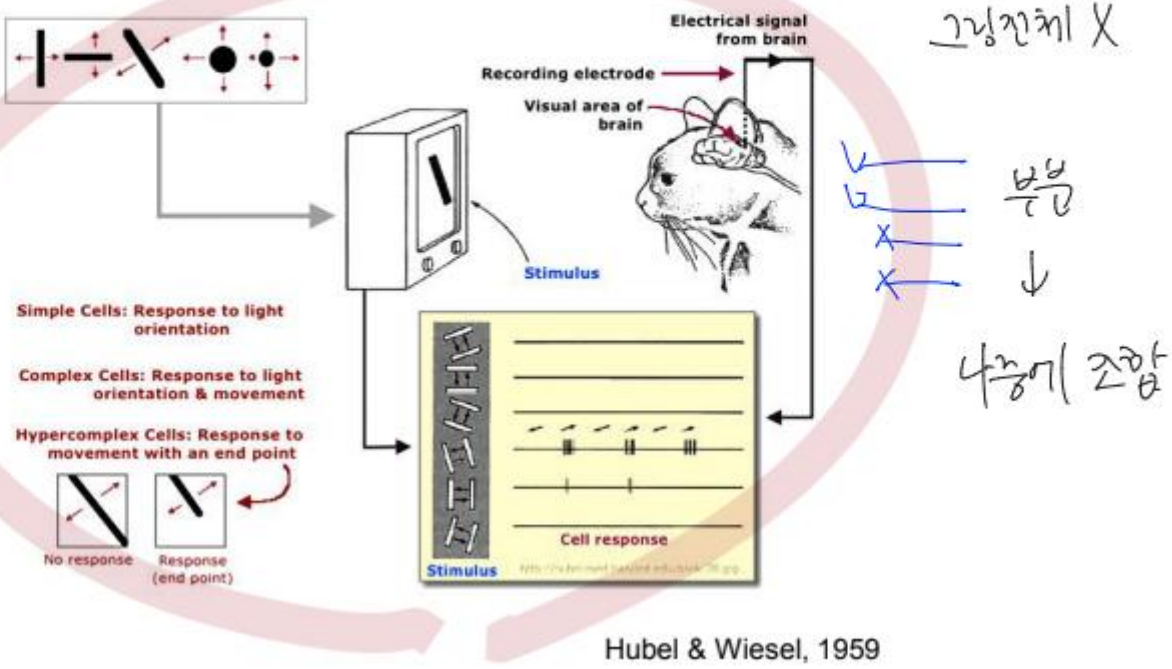
XOR – NN

XOR

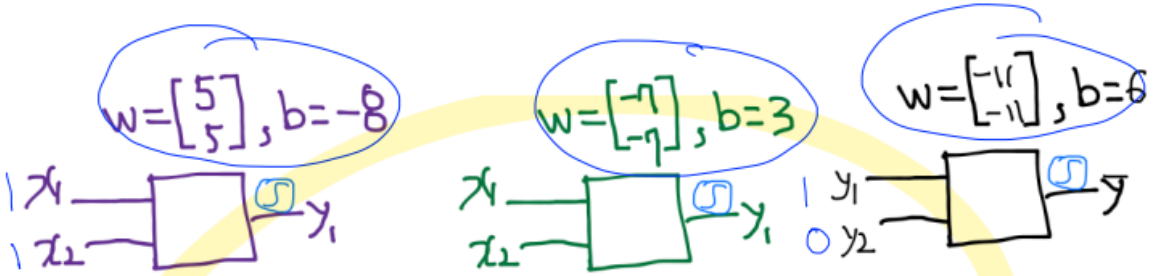
(Simple) XOR problem: linearly separable?



Convolutional Neural Networks



Hubel & Wiesel, 1959



$[1\ 1]\begin{bmatrix}5\\5\end{bmatrix}-8=5+5-8=2$, Sigmoid(2) = 1

$[1\ 1]\begin{bmatrix}-7\\-7\end{bmatrix}+3=-7+-7+3=-11$, Sigmoid(-11) = 0

$[1\ 0]\begin{bmatrix}-11\\-11\end{bmatrix}+6=-11+0+6=-5$

Sigmoid(-5) = 0

x_1	x_2	y_1	y_2	y	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0

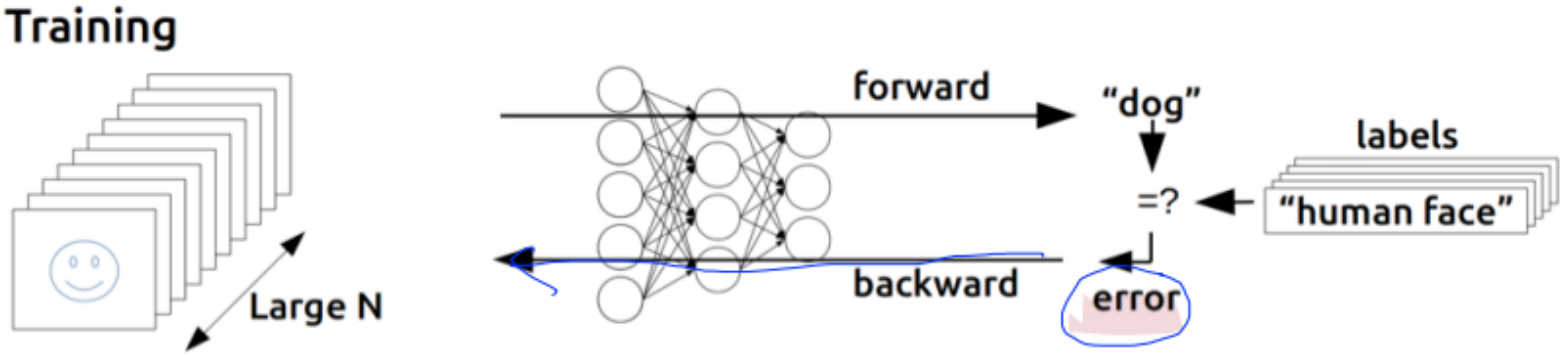
3개 layer learning

```
K = tf.sigmoid(tf.matmul(X, W1) + b1)
Hypothesis = tf.sigmoid(tf.matmul(K, W2) + b2)
```

Forward/Back Propagation

Forward/Back Propagation

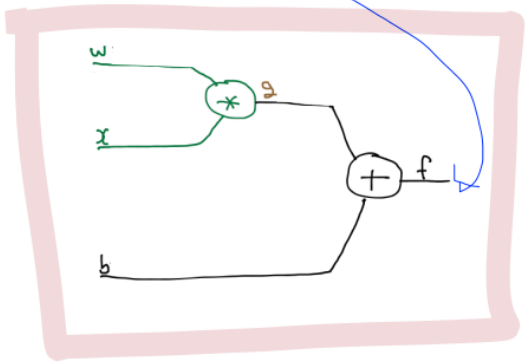
Backpropagation
(1974, 1982 by Paul Werbos, 1986 by Hinton)



Forward/Back Propagation

Back propagation (chain rule)

$f = wx + b,\ g = wx,\ f = g + b$

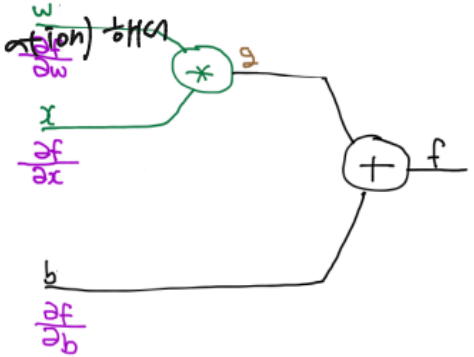


Back propagation (chain rule)

$f = wx + b,\ g = wx,\ f = g + b$

역전파.

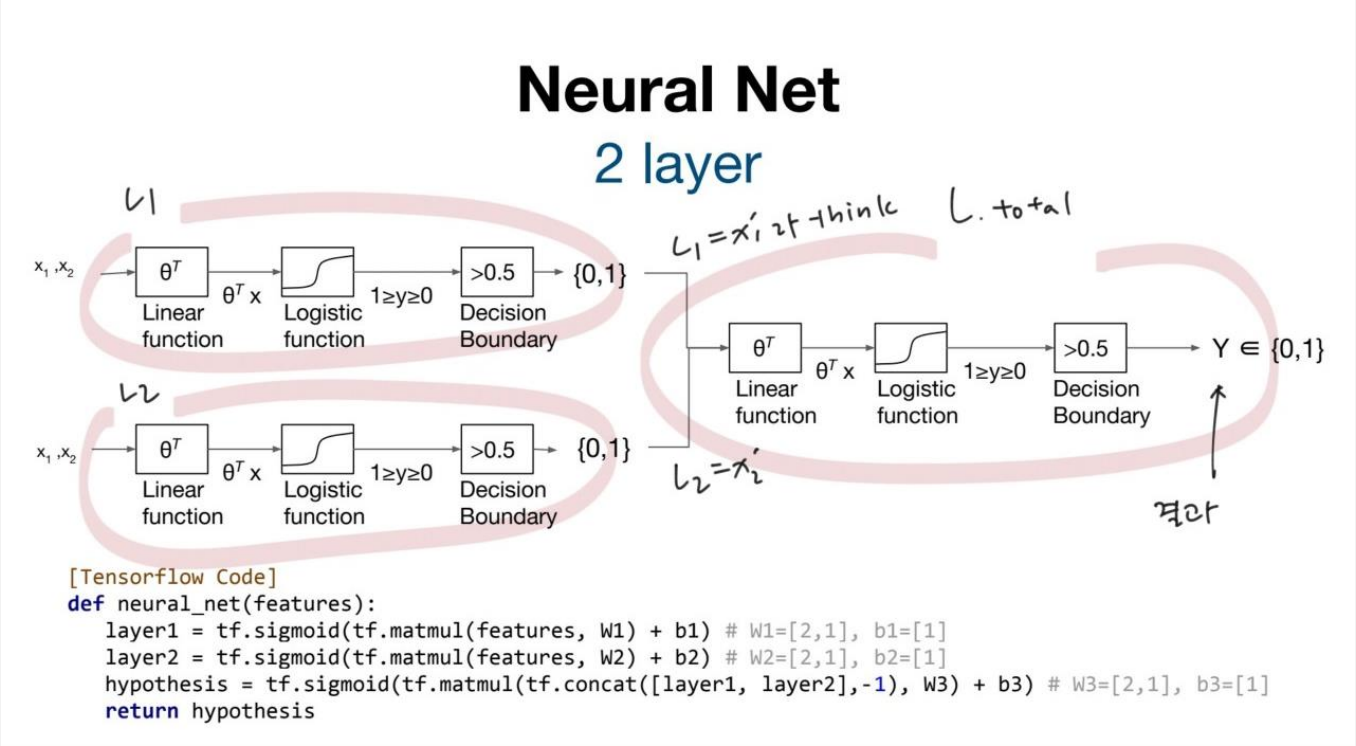
내 target과 실제 model이 계산한 output 사이
cost를 구하고 이것을
뒤로(back) 전파(propagation)해서
각 node가 갖는
변수를 갱신해내는 AI.



http://cs231n.

Sigmoid VS Relu

Sigmoid

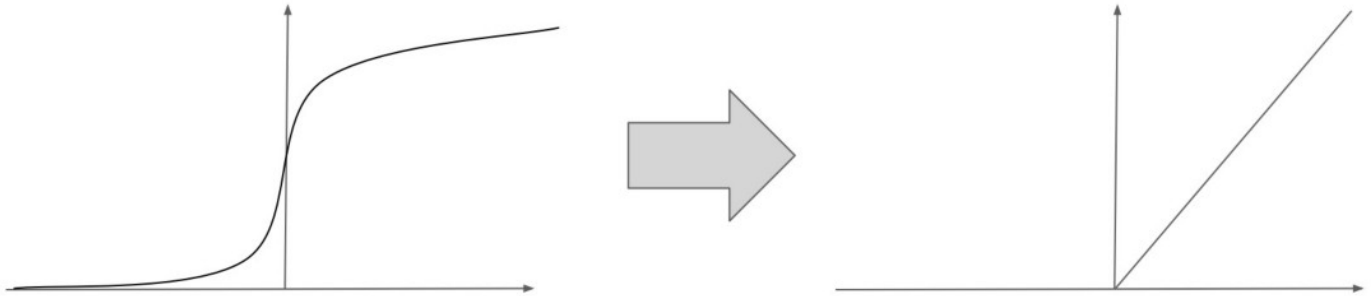



```
[Tensorflow Code]
def neural_net(features):
    layer1 = tf.sigmoid(tf.matmul(features, W1) + b1) # W1=[2,1], b1=[1]
    layer2 = tf.sigmoid(tf.matmul(features, W2) + b2) # W2=[2,1], b2=[1]
    hypothesis = tf.sigmoid(tf.matmul(tf.concat([layer1, layer2],-1), W3) + b3) # W3=[2,1], b3=[1]
    return hypothesis
```

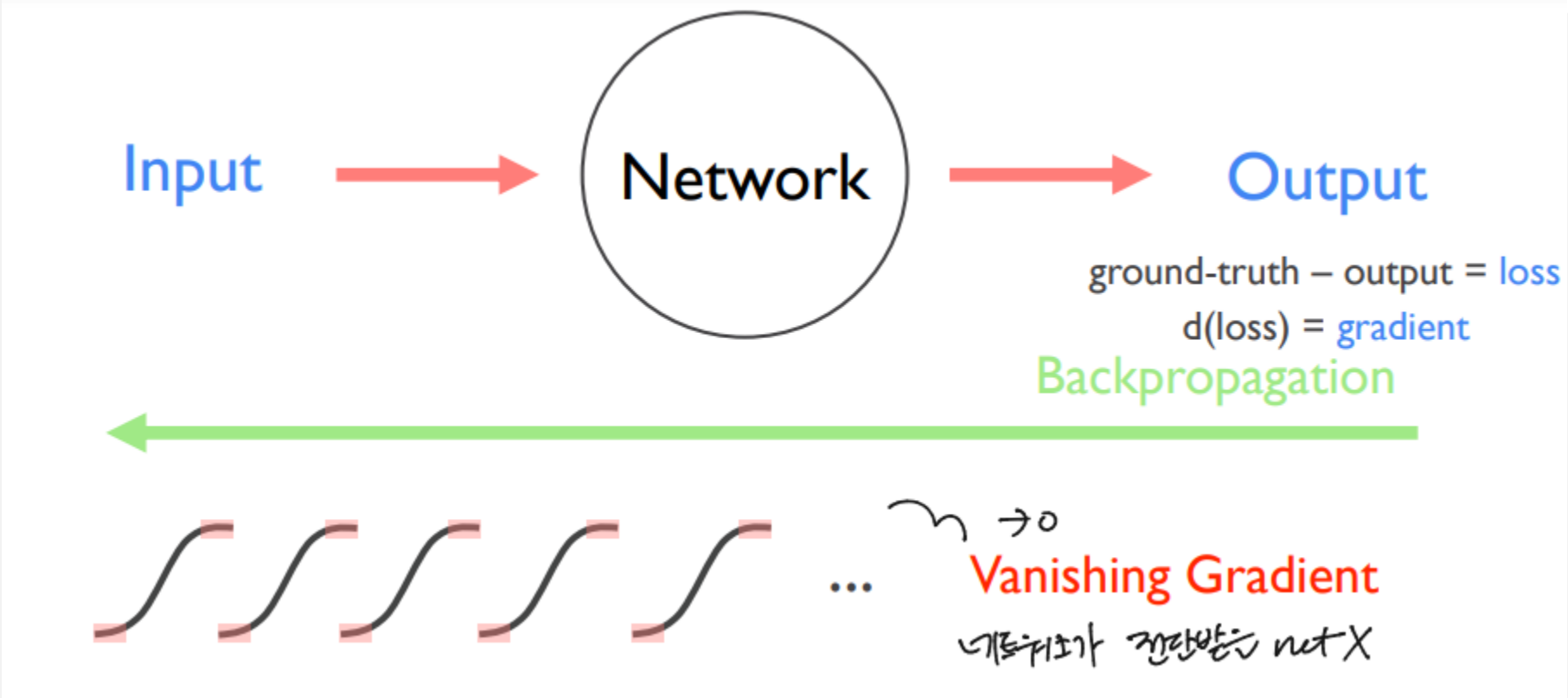
Sigmoid

What's Next?

Sigmoid -> Relu



Problem of Sigmoid

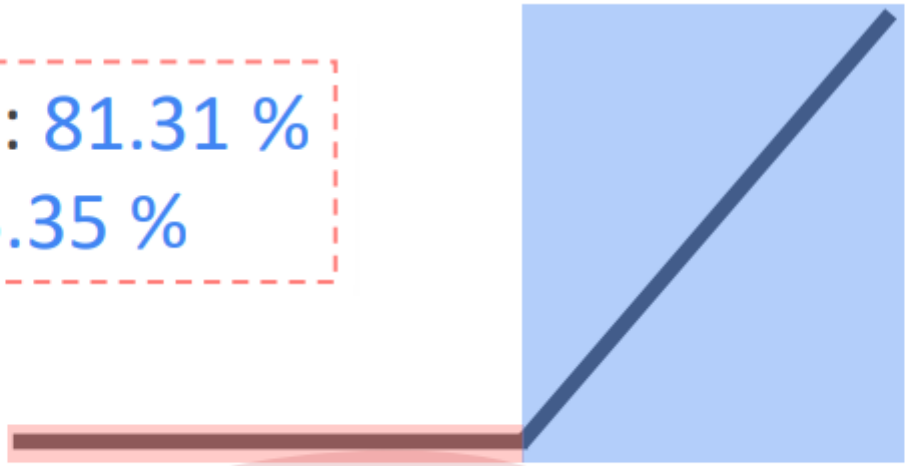


Why Relu?

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

Sigmoid : 81.31 %

Relu : 85.35 %

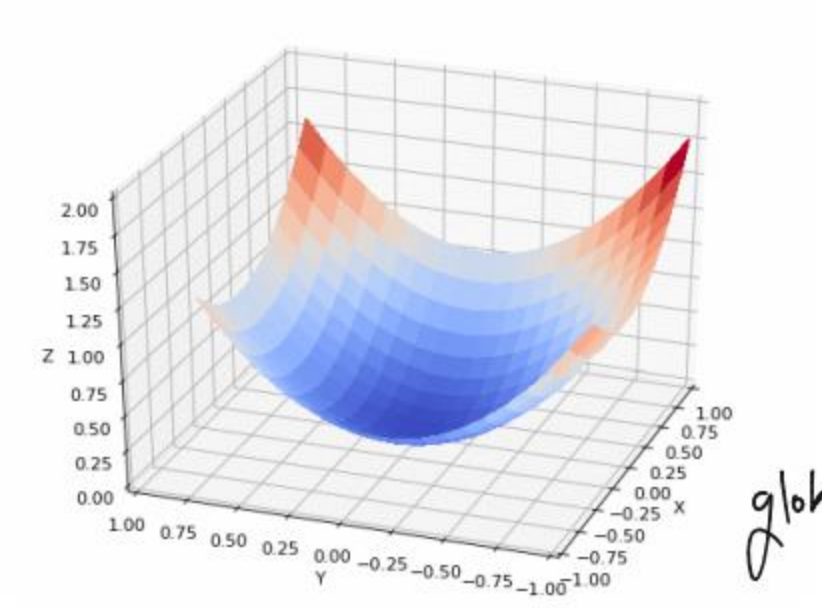


tf.keras.activations → sigmoid, tanh
relu, elu, selu

functional 에서 사용하면 좋습니다

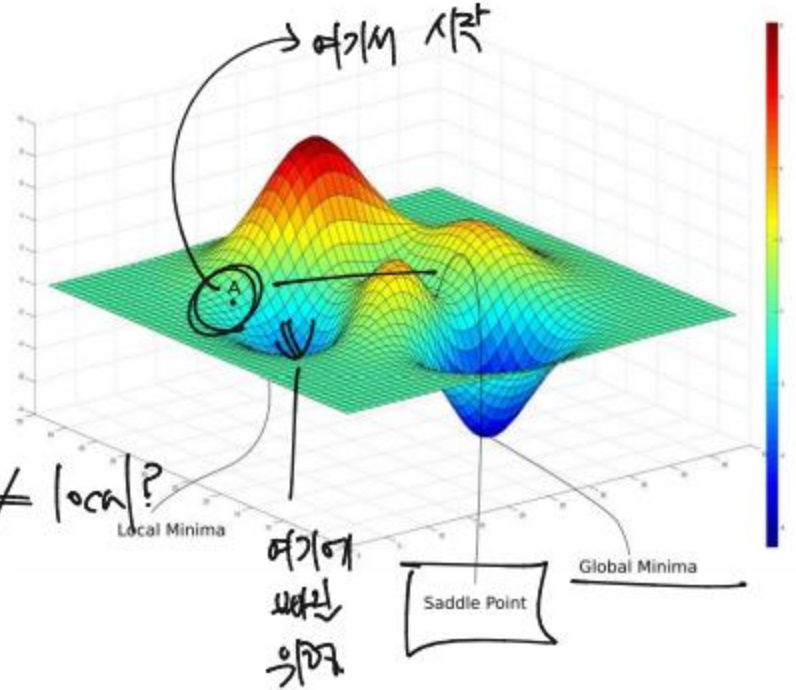
Weight Initialization

Xavier VS He



global min.

global ≠ local?



Xavier VS He

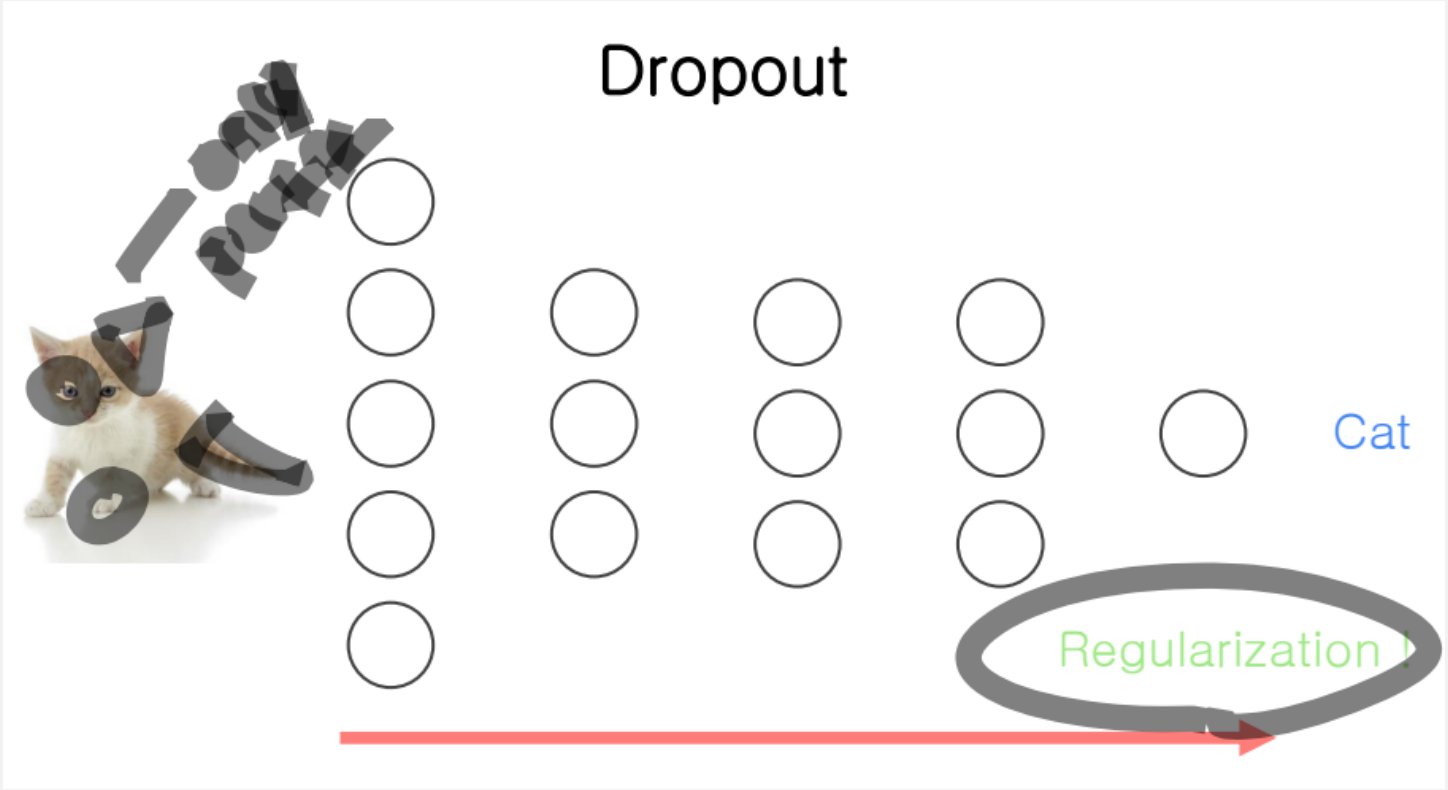
```
# Create network

weight_init = tf.keras.initializaers.RandomNormal()

weight_init = tf.keras.initializaers.glorot_uniform()

weight_init = tf.keras.initializaers.he_uniform()
```

Random : 85.35 %

Xavier : 96.50 %

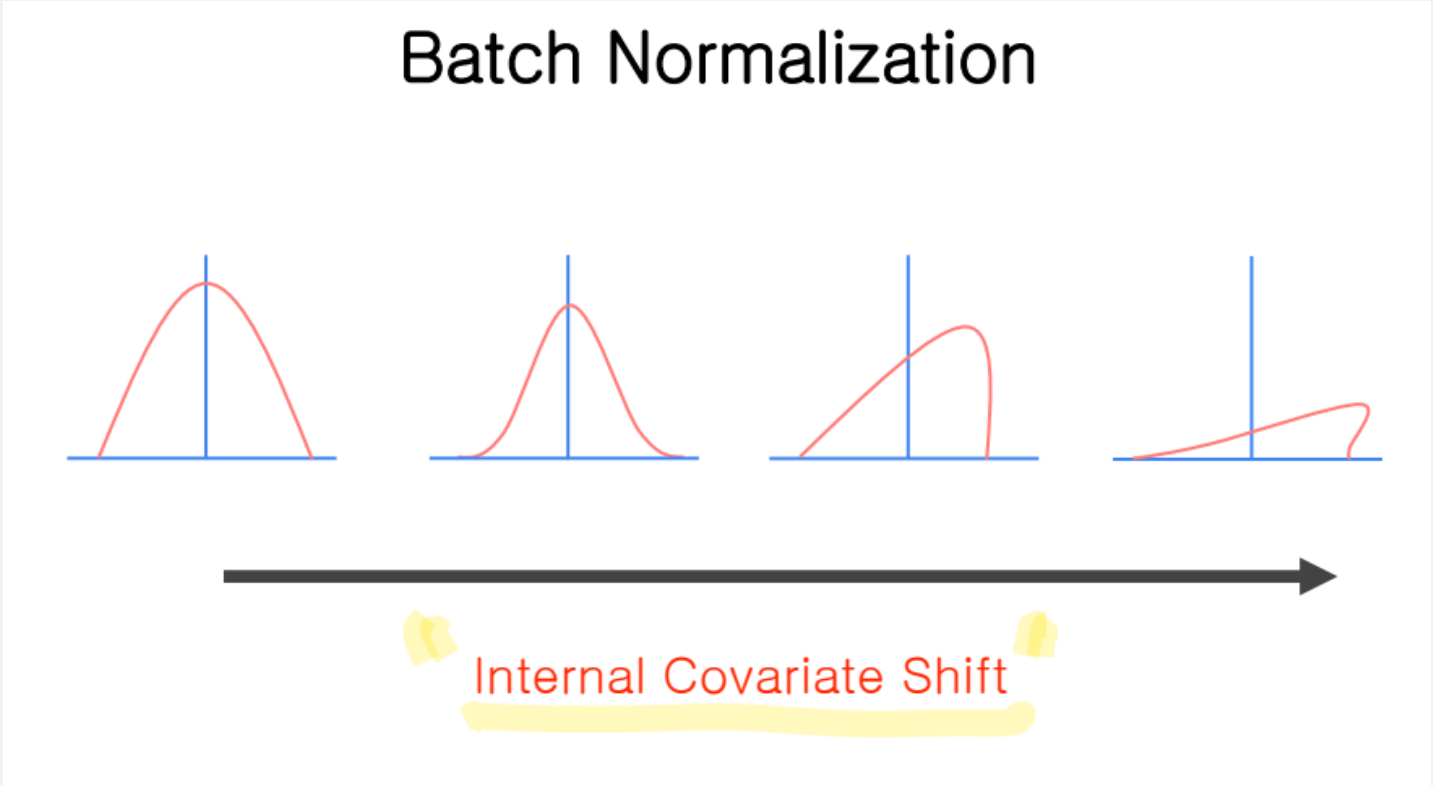
Drop out

Drop out



Batch Normalization

Batch Normalization



Batch Normalization