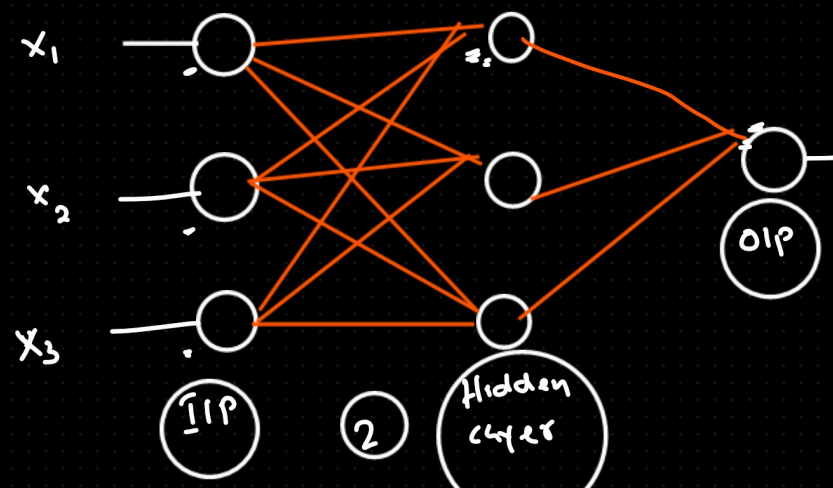
1. Vanishing grad | Exploding grad.
2. Data Scaling → Batch Norm.
3. Drop out
4. Regularization
5. Weight initialization

1. Batch norm
2. Weight init
3. Vanishing grad
4. Pytorch
5. hyperparameter tuning (Keras tunar)

Vanishing grad Prob

1. Forward Propagation

Epoch = train



$\hat{y} \rightarrow loss(y, \hat{y})$

$(\Sigma + Act)$

Sigmoid, tanh, Relu

Softmax

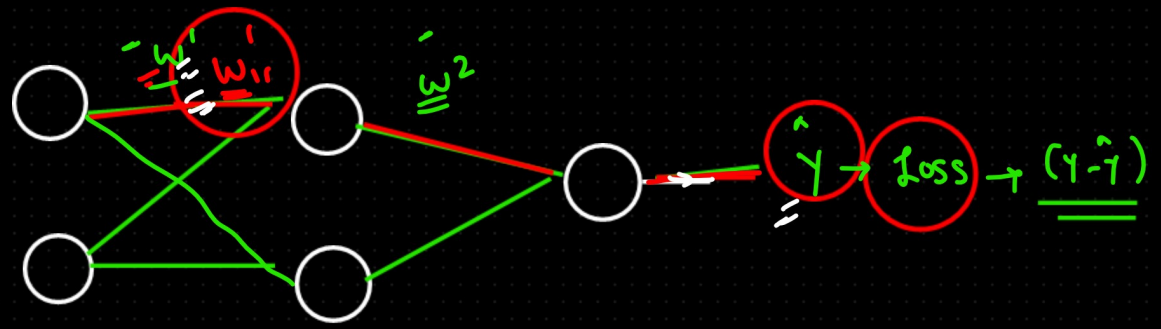
Backward Propagation

Gradient will be very very small → there won't be any sort of a weight update

Vanishing

= Vanishing gradient

Neural Network ⇒ Layers Nodes ⇒ Multiple ⇒ Deep Neural Network



$$W_n = W_o - \eta \frac{\partial L}{\partial w}$$

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{Y}} \times \frac{\partial \hat{Y}}{\partial I} \times \frac{\partial I}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{11}^1}$$

= Calculation (I/P x weights) = O/P ⇒ Back Propagation

(Derivation / Partial Derivation)

[0-1]

Sigmoid → [0-0.25]

1

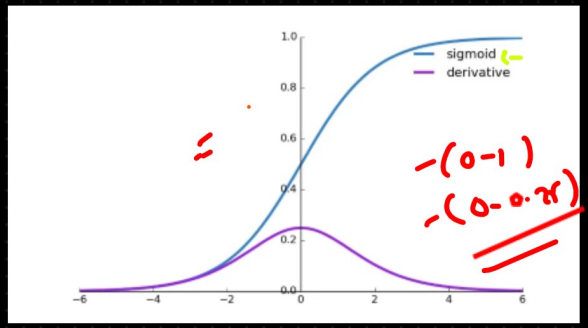
Neural Net → Deep Neural Net ⇒ Vanishing grad. ⇐ wrong Activation

2

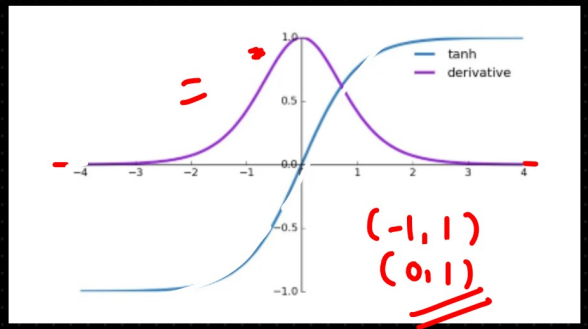
1. sigmoid
2. tanh

there is no update in the weight

BP



-(0-1)
-(0-0.25)



(-1, 1)
(0, 1)

Sigmoid

[0, 1]

B.P. derivation -[0, 0.25]

tanh.

(0, 1)

$0.1 \times 0.1 \times 0.1 \times 0.1 = 0.0001$

= Deep Neural Network ⇒ Training ⇒ Forward Propagation ⇒ Loss

Sigmoid / tanh

Update the weights

$W_n = W_o - \eta \frac{\partial L}{\partial W_o}$

Chain rule ← (0 - 0.25)

0.01, 0.1, 0.001, 0.2, 0.15

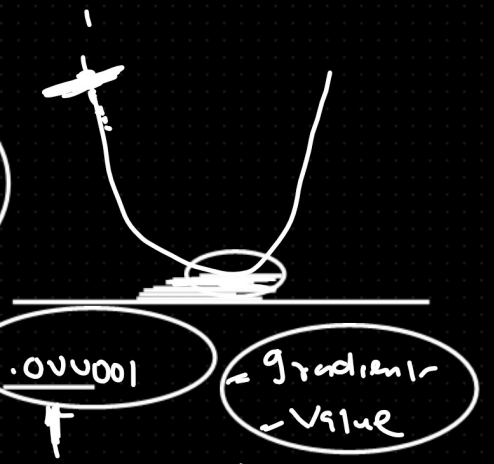
Small × Small × Small ... = very small value

$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W}$

1 − 0.0001 = 0.9999

negligible

$\frac{\partial L}{\partial W}$



which is going to be vanish

Vanishing grad. → Deep Neural Net

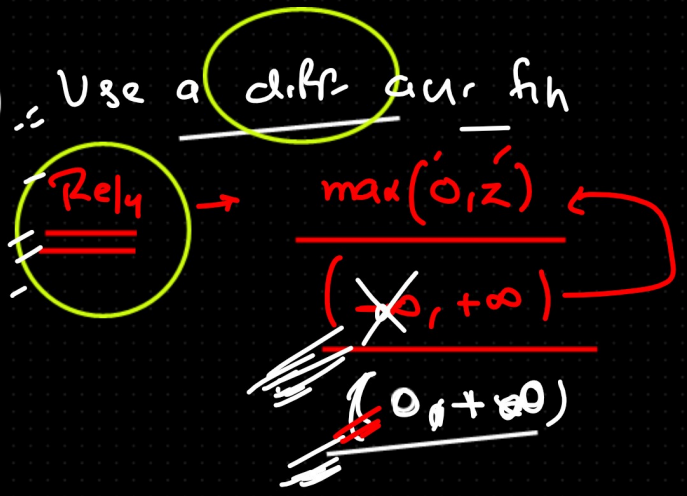
1. Loss will not change
2. Weight will not change

Vanishing grad (How you gonna resolve it)

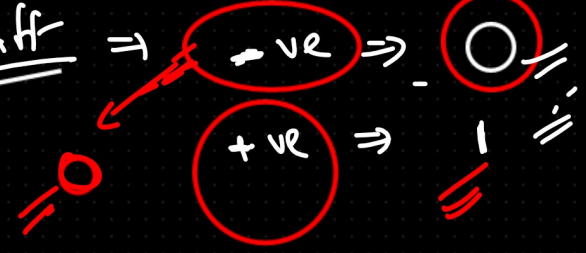
1. Reduce the Complexity of your NN:
   - ↳ Reduce the Nb. of layer
   - ↳ Reduce the No. of Nodes

   (Shallow Neural Net)

2. Use a diff act fn



It will remove negtive value from your data



Dying relu ← 0 or 1

(chain rule $\frac{\partial L}{\partial w} = (\ldots\ldots\ldots) \leftarrow 0$

Leaky relu

$\max(0.1x, x)$

= L-Relu

$= \max(0, x) \leftarrow (-\infty, +\infty)$

3 Proper weight initialization

1 Xavier

2 global

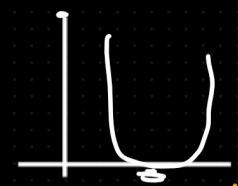
Derivation

0, 1

chain rule

$w_n = w_0 - \eta \frac{\partial}{\partial}$

4 Batch Normalization ←



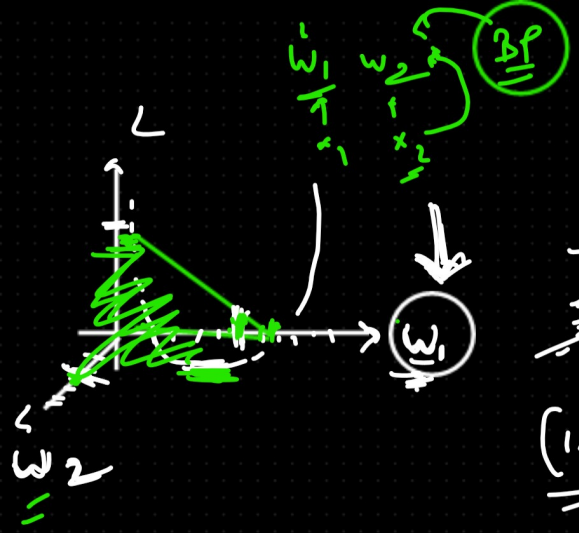
weight rule

Exploding → $(2 \times 0.5 \times 1 \times 3 \times 5 =)$

Relu

How to improve a Performance of Neural Network ?

1. Handle or TC vanishing gradient
   - Activation fn
   - Correct weight initialization
2. Overfitting
   - Early stopping
   - Dropout
   - Regularization
   - Reduce complexity (Layer, Node)
   - Increase data
3. Normalization (fast training)
   - Normalization / Std.
   - Batch Normalization
4. Optimizer (exploding grd.)
   - Momentum
   - Adagrad
   - Adadelta / RMSProp
   - Adam
5. HyperParameter
   - No. of hidden layer
   - Node / Neuron
   - Batch size
   - Learning rate
6. Activation
   - sigmoid
   - tanh
   - type Relu
   - Softmax
   - Linear / Step



Data Scaling

Age

Salary

[1-95] ⟷ [1 - 10000000]

(1-95) | (1-100)

↓

Keep the data in d same scale

= Input x weight + bias

= Age x w_1 + Sal x w_2 + bias

so | 00000

↓

Contour

1 Std

$$\frac{X_i - \mu}{\sigma}$$

[-3, +3] ztable SND

Salary

2 Normalization

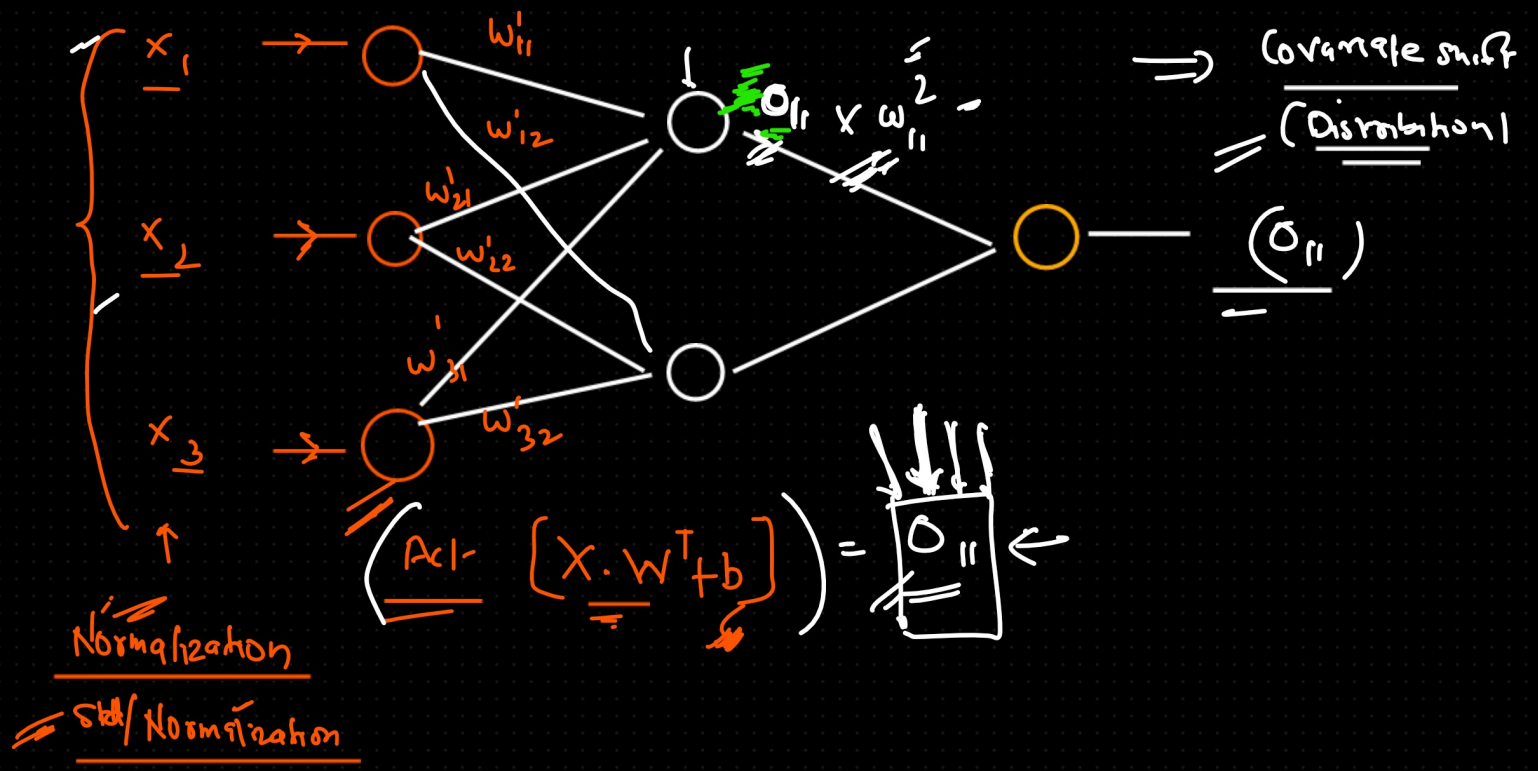
$$\frac{X_i - X_{min}}{X_{max} - X_{min}}$$

[0-1]

CGPA

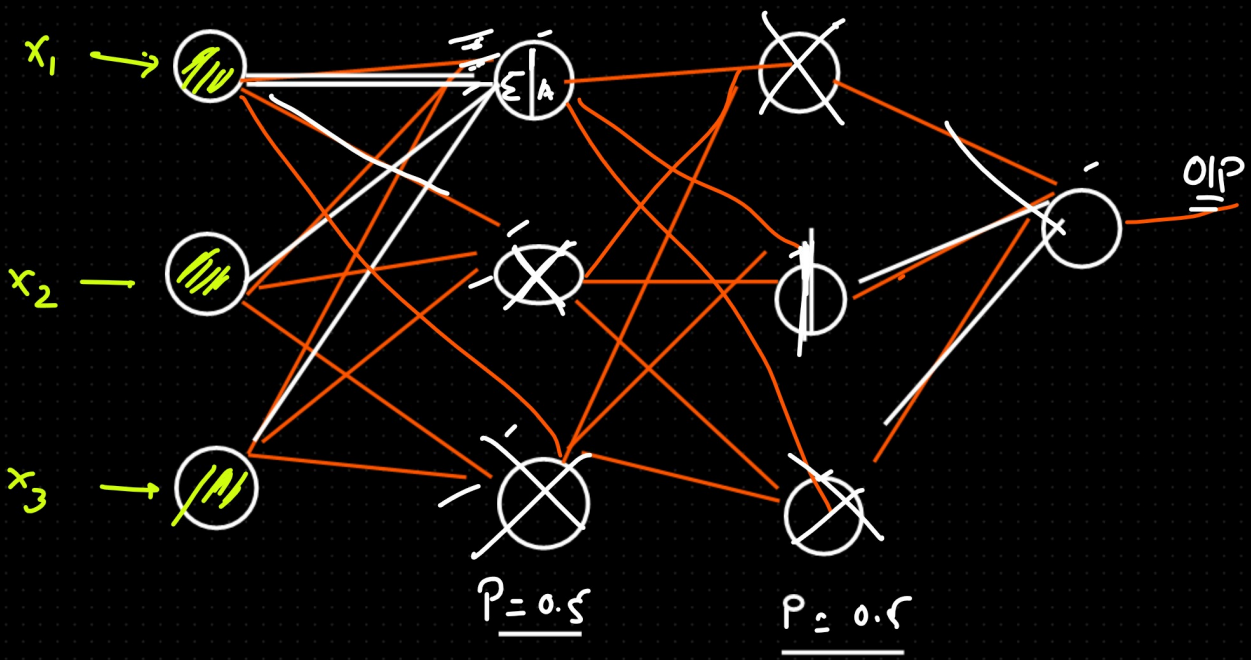
(0-10)

Zig Zag

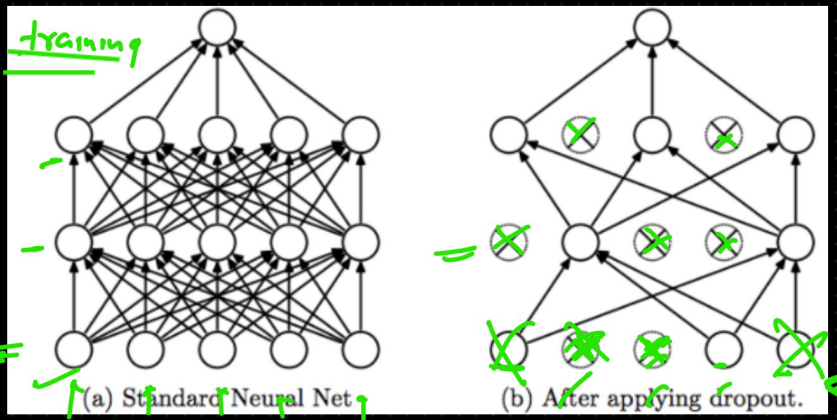

x_1
x_2
x_3
w'_{11}
w'_{12}
w'_{21}
w'_{22}
w'_{31}
w'_{32}
$O_{11} \times w^2_{11}$
O_{11}
Act- $[X \cdot W^T + b]$ = O_{11}
Normalization
Std/Normalization
Covariate shift
(Distribution)

Drop out-

Training (1 epoch)



5 epoch



Dropout → 0.5 → 50%

→ 0.30 = 30%

→ 0.25 = 25%

→ 0.5

Shutdown

Randomization

ff-

Regularization ⇒

- L_1 → Lasso ← $w \to 0$ ⇒ Sparsity
- L_2 → Ridge → toward to 0 (not completly zero)
- $L_1 + L_2$ → elastic

Overfitting

$$\text{Loss} + \text{Penality}$$

↳ $L_1 / L_2 / L_1 + L_2$

$$L_1 \rightarrow \text{Loss} + \lambda |w|$$

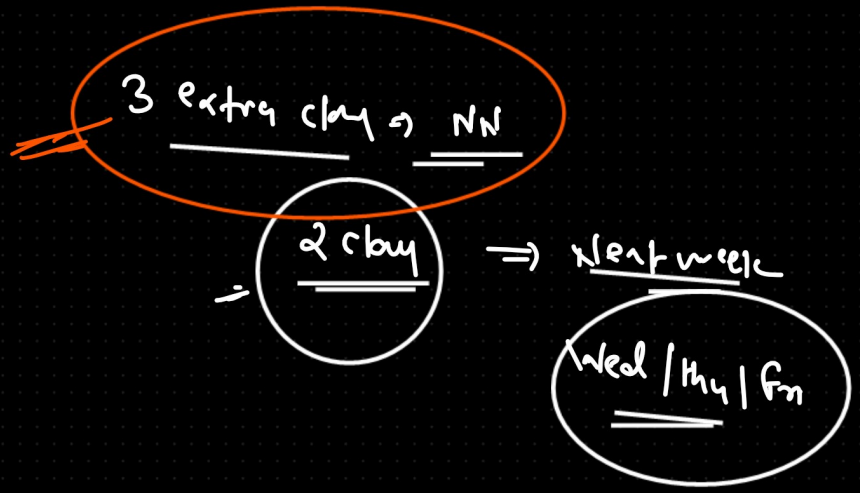
$$L_2 \rightarrow \text{Loss} + \lambda |w|^2$$

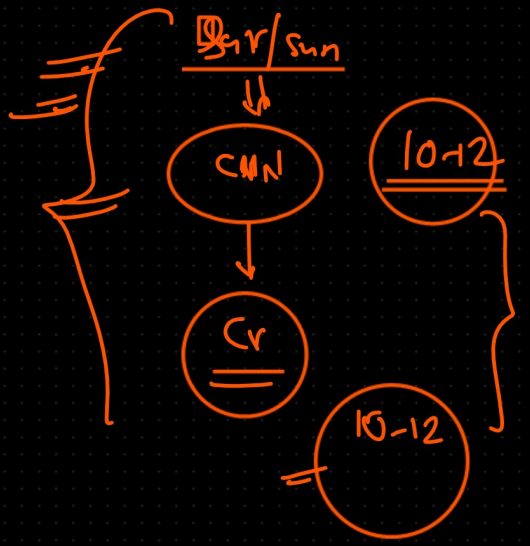
$$L_1 + L_2 \rightarrow \text{Loss} + \lambda(|w1|) + \lambda(w2)$$

} Single value

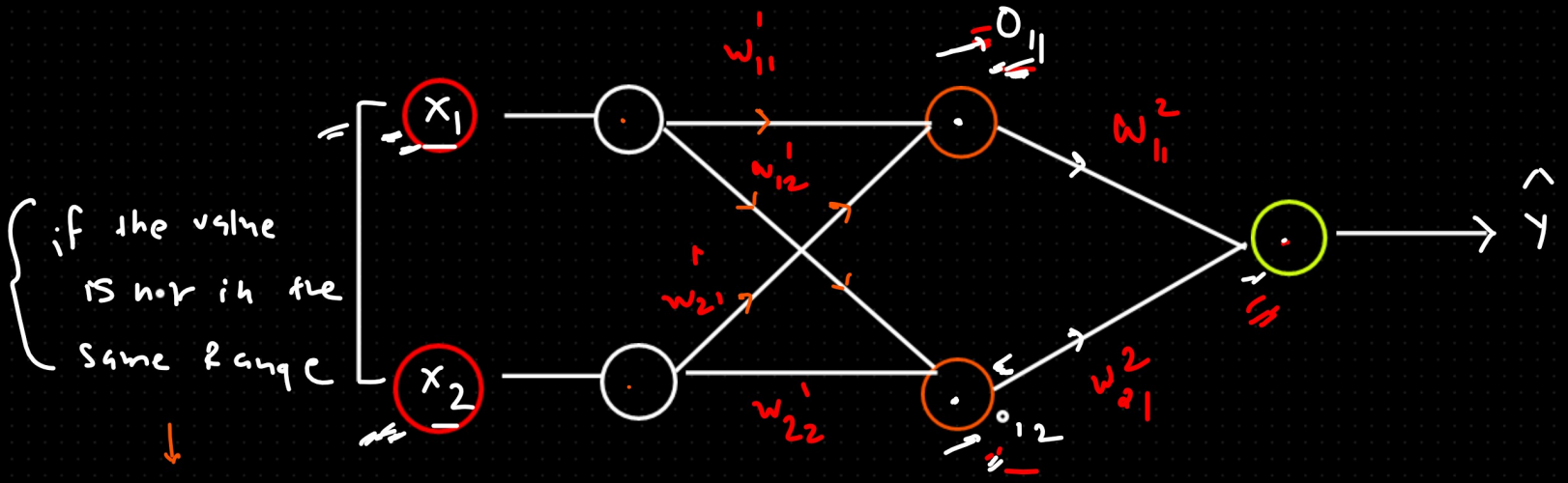
Cost

$$\text{Loss} + \lambda \sum_{n=1}^{n} |w|^2$$


3 extra classes NN
2 days
Next week
Wed / Thu / Fri


Sat/Sun
CMN
10-12
Cr
10-12

Batch normalization



if the value is not in the same range

↓

Scaling

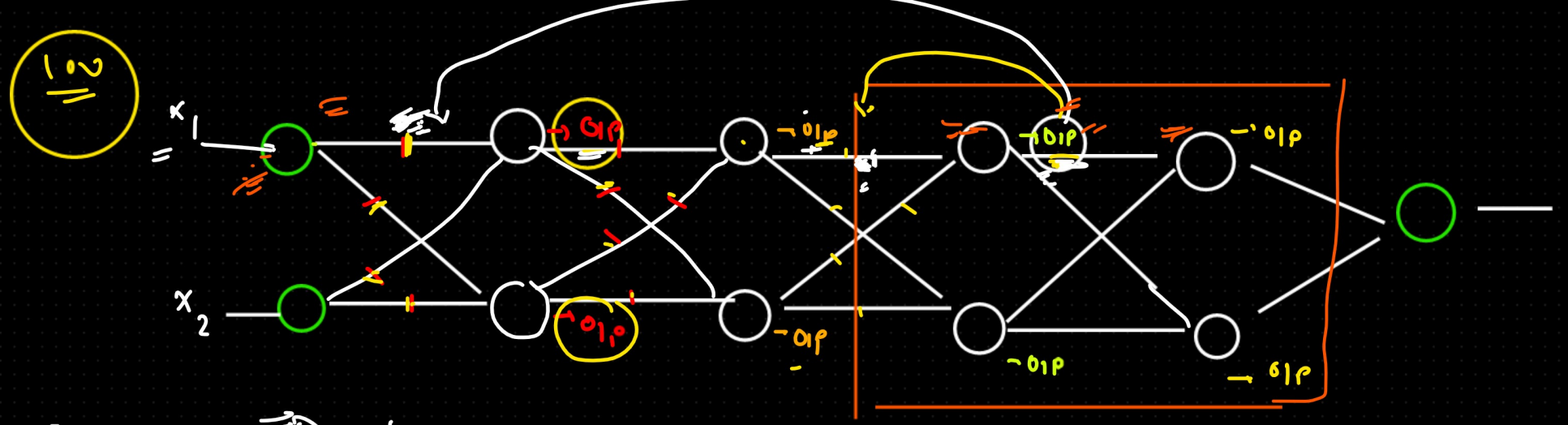
Std Normalization

$$z = (x_1 * w'_{11} + x_2 * w'_{21} + b_{11})$$

$$o_{11} = Act(z)$$

≠ Batch normalization → Batch gradient decent

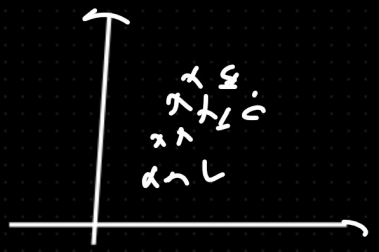
o_{11}, o_{12} → Next layer

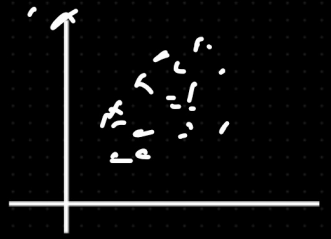


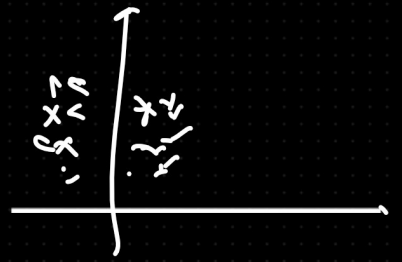
Deep layer

Batch Normalization

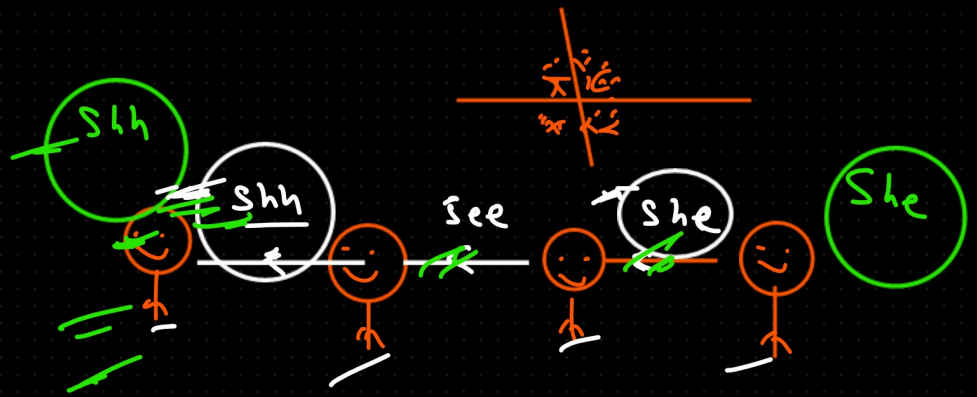
Scientist



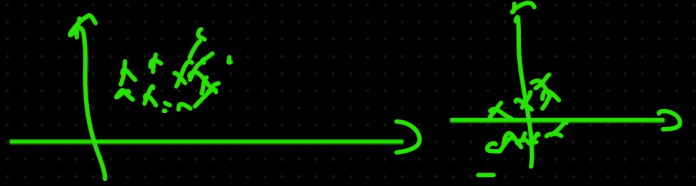




Deep NN



Internal Covariate Shift



Normalization

↓

Batch Normalization

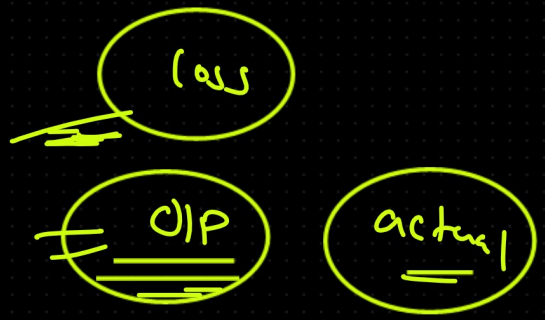
1. fast conversion, fast training
2. Internal cov. shift (DNN)

(Batch + Normalization)

↓

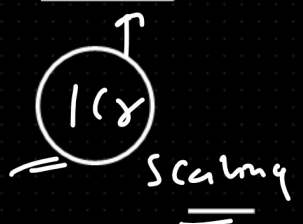
o/p from each layer

Hidden



time

Data → 95%.



i/p

65%, 70%

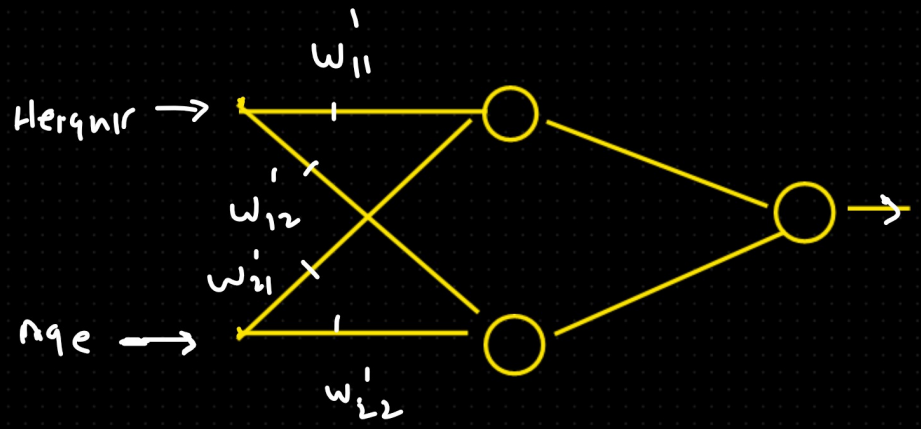
6 hr

Scaling

Batch

3.5 hr 72%.

Increase → Batch → 5 hr. → 85%.



(1) $z_{11} = w'_{11} \times \text{height} + \text{Age} \times w'_{12}$

$\text{Act}(z_{11}) = o_{11}$

Act

$$\left\{ \frac{\text{Std}}{\frac{o_{11} - \mu}{\sigma}} \right\}$

$\left\{ \frac{x - x_{min}}{x_{max} - x_{min}} \right\}$

1 $z_{11} \rightarrow Actr(z_{11}) \rightarrow o_{1_1} \leftarrow o_{11}^{N}$

2 $z_{1_1} \rightarrow z_{11}^{h} \rightarrow Actr(z_{11}^{h}) \rightarrow o_{11}$

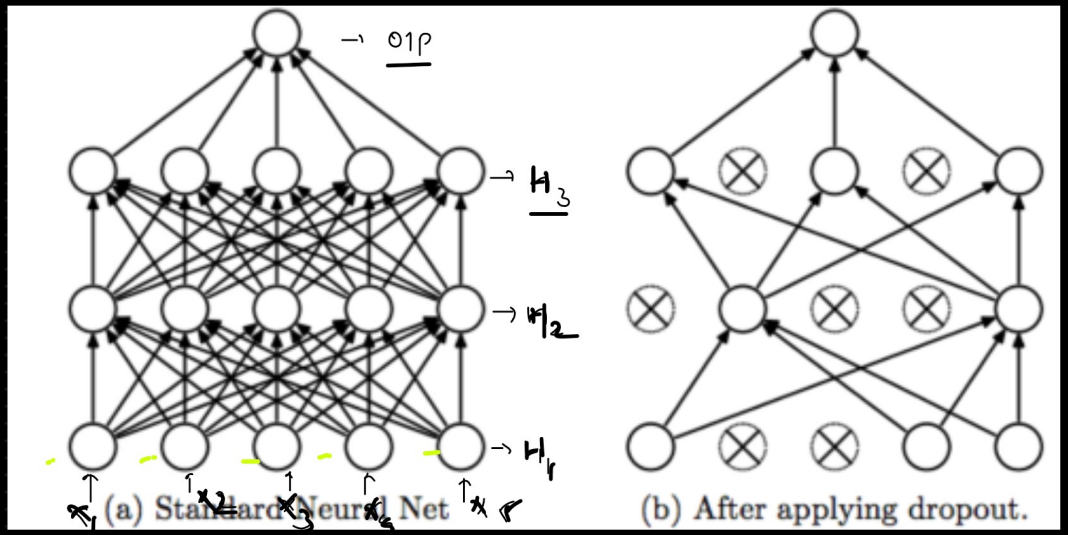
research

Batch Normalization

Dropout ⇒ Recent research in the field of NN

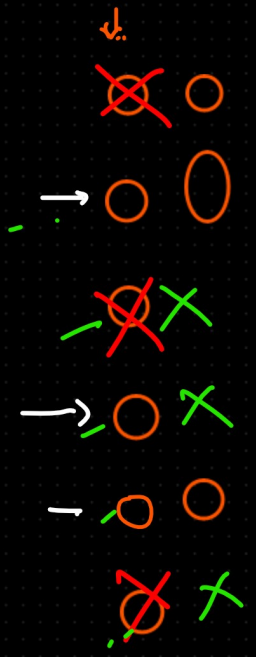
(2014)

- Concept



1 epoch ⇒ Forward (o/p) + Backward (Parameter)

Shortcut, Shortcut (Neuron) = Randomly



Epoch

Epoch

ShutDown → training

(deactivation)

testing → Neuron will be activated

Dropout ?

P = 0.5

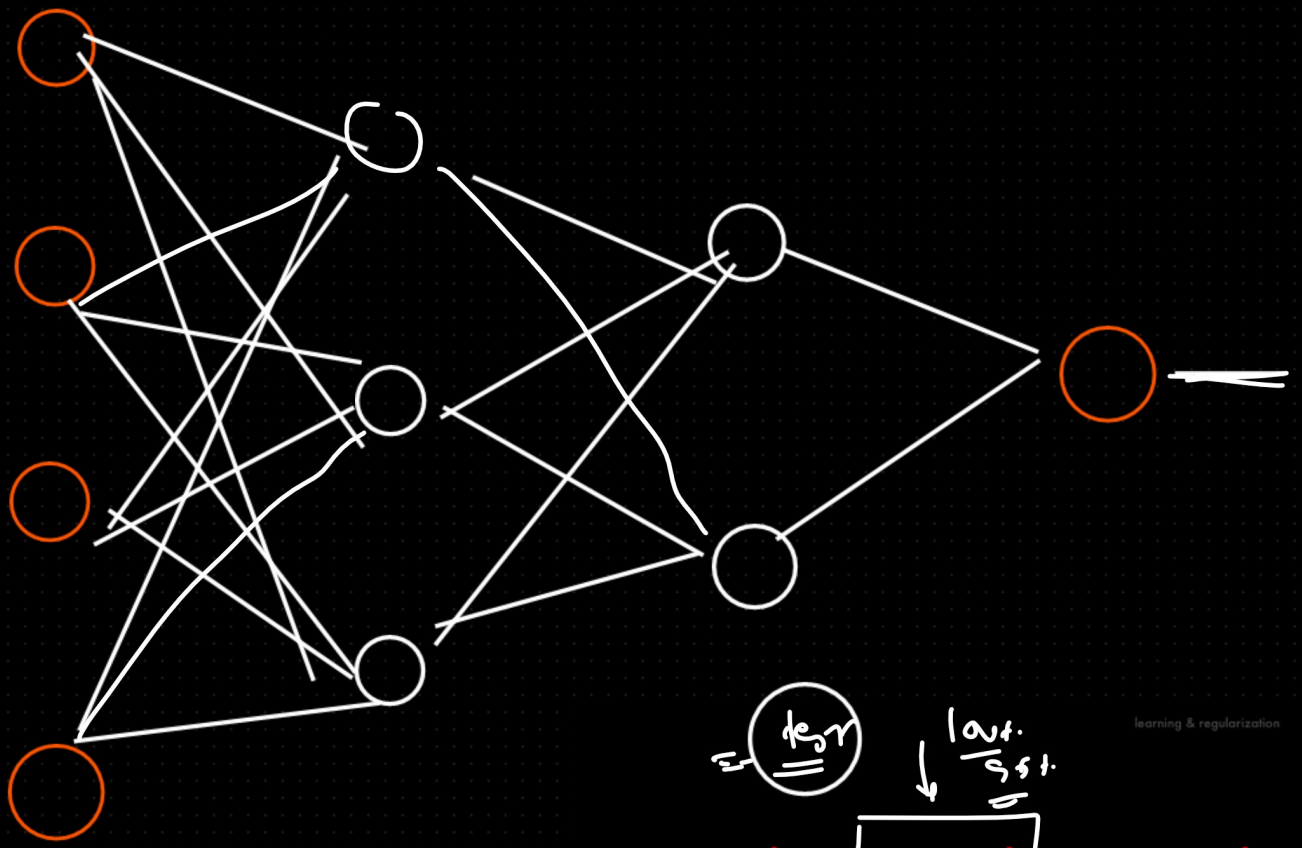
Shut Down

Sol.

Deep =

fully connected

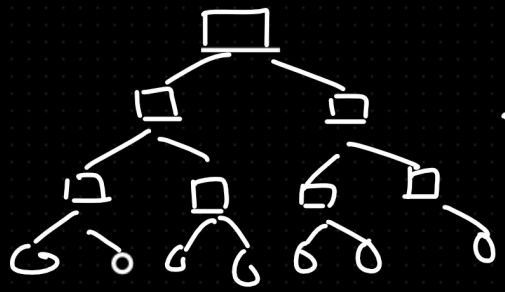
Precise



Decision tree → Random forest

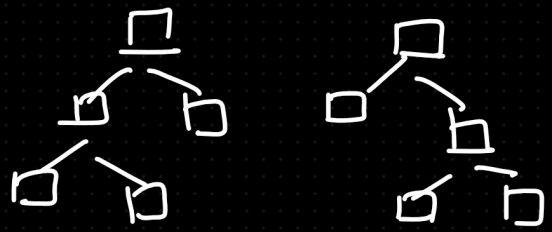
(Bootstrap)

– (R+C)



→ overfitting

= Shalini

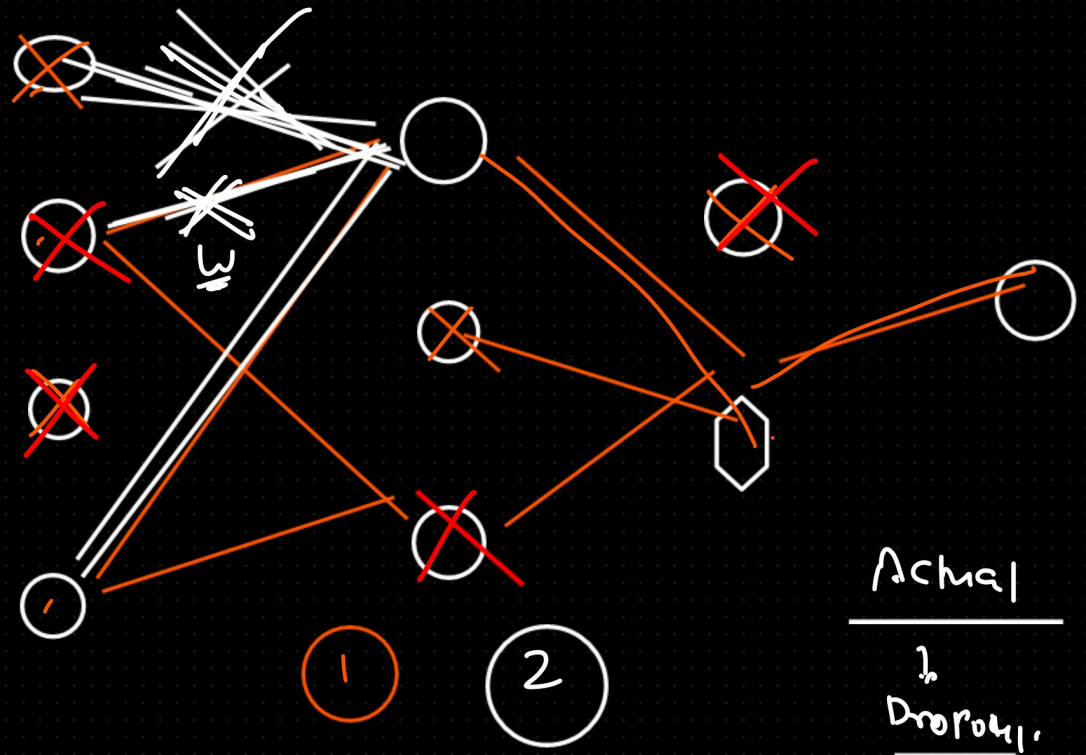


Randomization –

Voting – – General training testing

ANN

General



1 epoch

2 epoch

Actual

1. Proposal

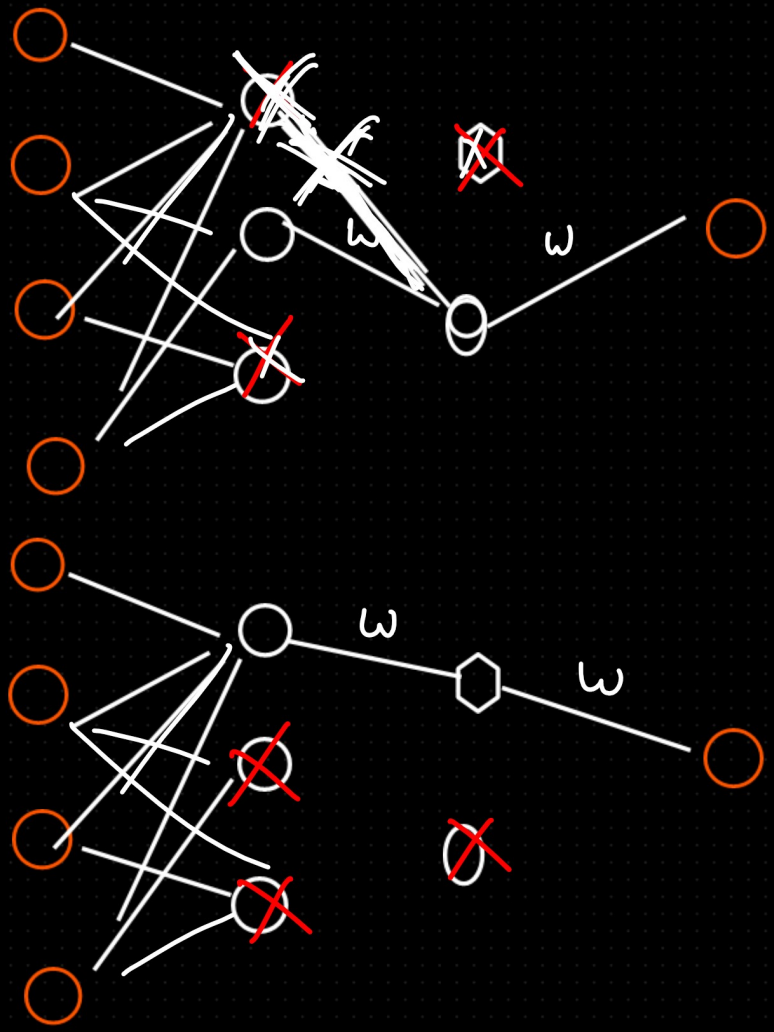
P

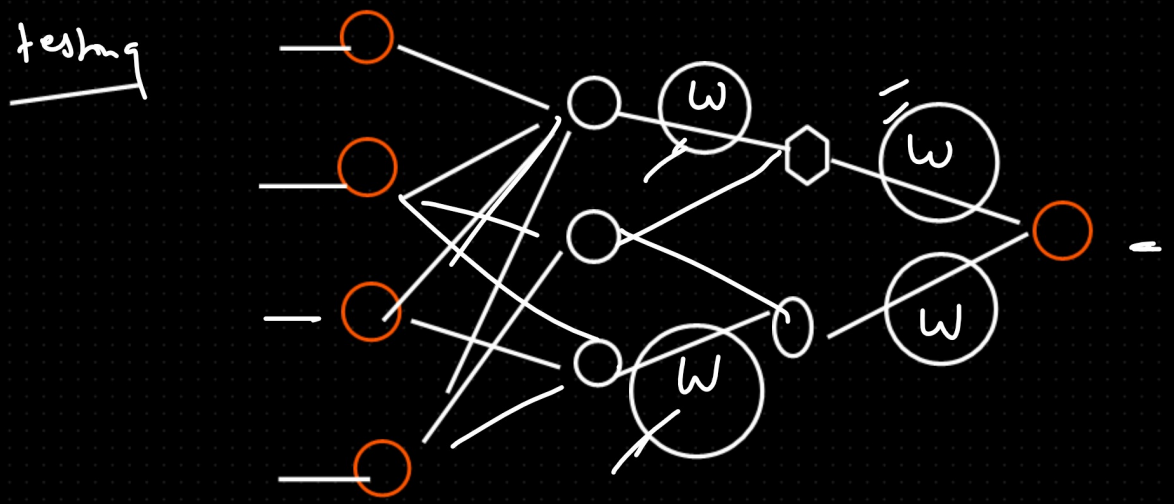
diff pattern

= Randomization

P = 0.25 — deactivation

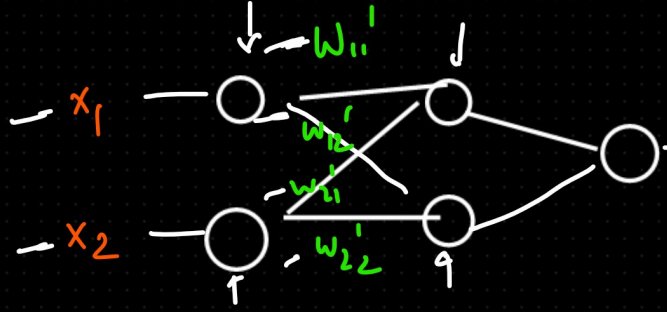
1-P = 0.75 — activation




testing
w
w
w
w

Weight initialization

1. Zero initialization
2. initialization with Constant value
3. Random initialization
4. Xavier/global init
5. he init



$z_{11} = x_1 \times W_{11}' + x_2 \times W_{21}' = 0$?

$\text{sigm}(z_{11}) = 0.5$

Cons: W = equal → after training

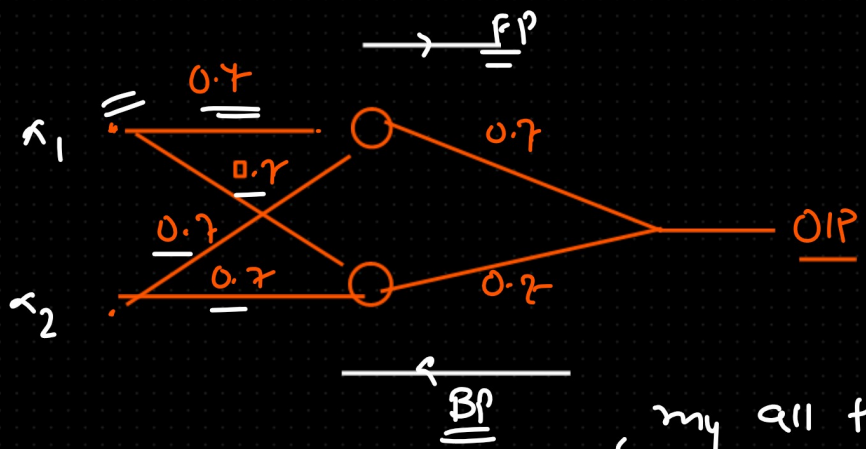
$W = 0$ zero-initialization

1. tanh → 0
2. Sigmoid
3. Relu → 0

No training, No weight updation

2 Constant value

$W = 0.7$ FP



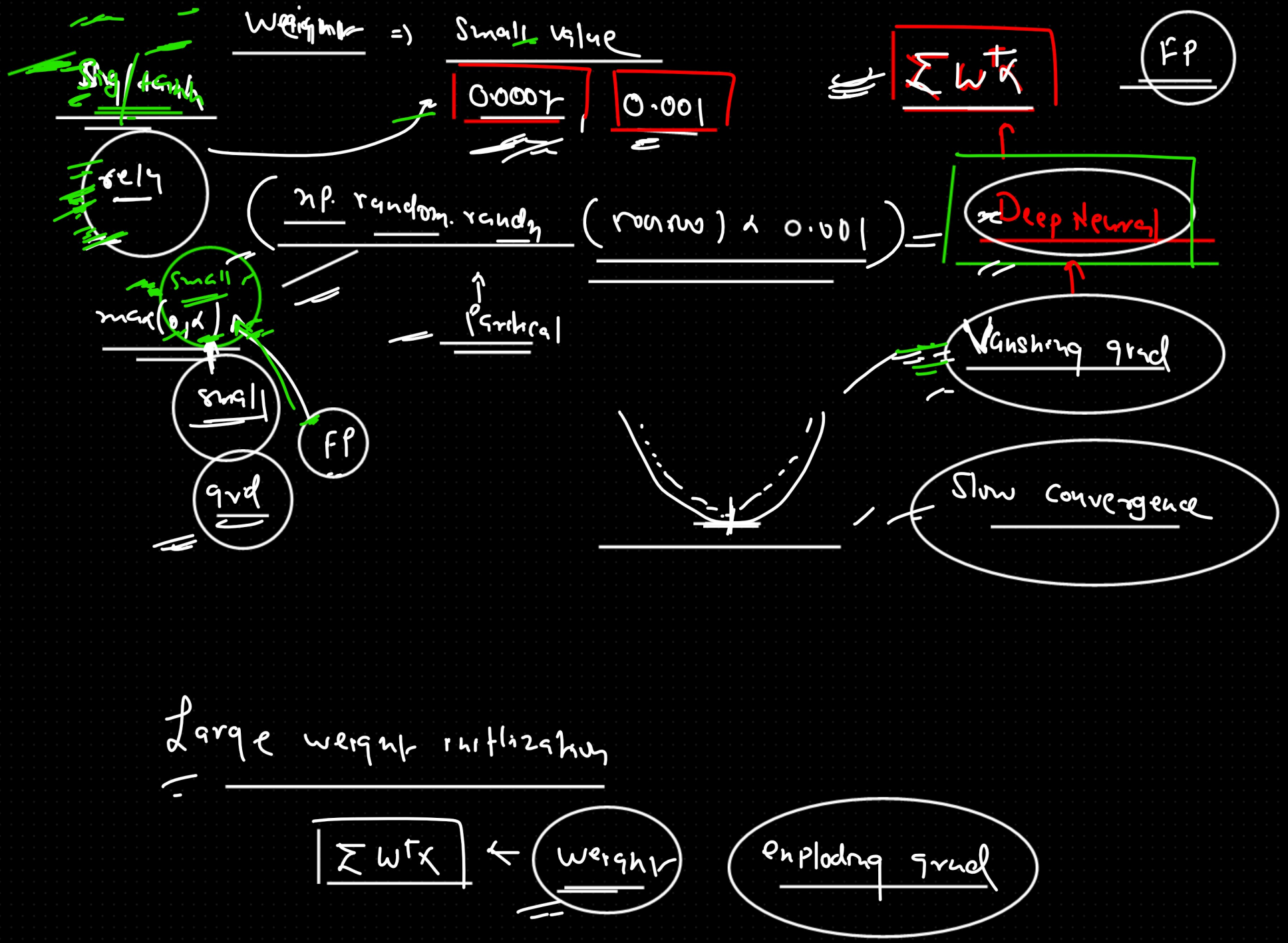
∴ my all the weight will same

3 Random initialization

↳ ① initialization with small value

② initialization with larger value

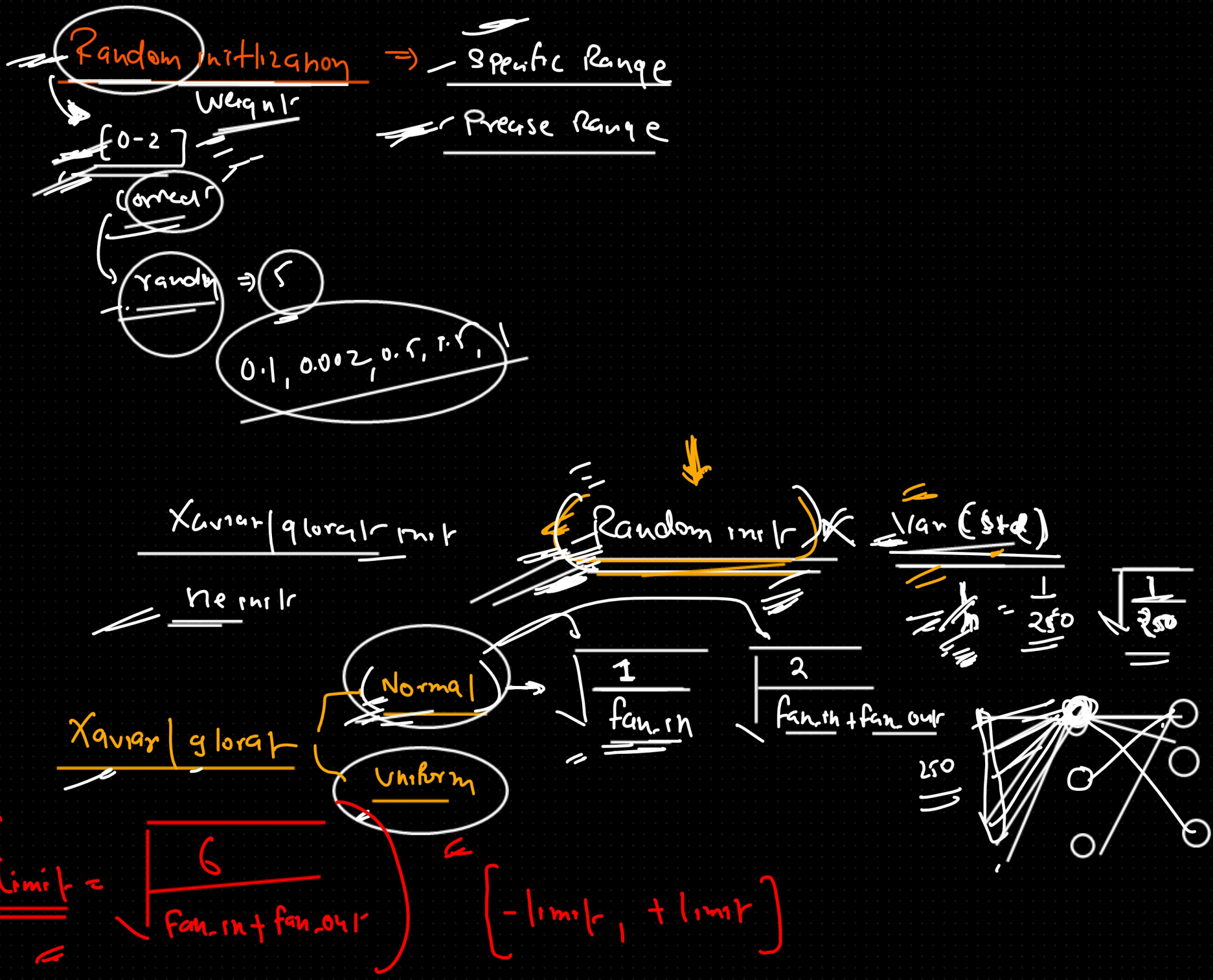
tanh, sigmoid, relu



Large weight initialization

ΣwTx ← Weight

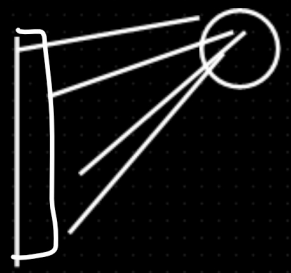
Exploding grad


Random initialization
Specific Range
Precise Range
Weight
[0-2]
Normal
randn
0.1, 0.002, 0.5, 1.5, 1
Xavier/global init
He init
Random init
Var (std)
1/250
√(1/250)
Normal
Uniform
1/fan_in
2/(fan_in + fan_out)
250
Xavier/global
limit = √(6/(fan_in + fan_out))
[-limit, +limit]

He init → Uniform → $[-limit, +limit] \rightarrow \sqrt{\frac{6}{fan\_in}}$

↘ Normal → Random init $\times \sqrt{\frac{2}{fan\_in}}$

np.random



1. Zero initialization
2. Initialization with constant
3. Random initialization → small / larger } $\times$ (we don't the specific range)

?

my weight can misslead to me

4. Xavier/Glorot init
   - ↳ Normal (NP.random) $\times \sqrt{\frac{1}{f\_in}}$ or $\sqrt{\frac{2}{f\_in + f\_out}}$
   - ↳ Uniform $[-limit, +limit] \Rightarrow limit \left[\sqrt{\frac{6}{f\_in + f\_out}}, + \sqrt{\frac{6}{f\_in + f\_out}} \right]$
5. He init
   - ↳ Normal (NP.random) $\times \sqrt{\frac{2}{f\_in}}$
   - ↳ Uniform $[-limit, +limit] \rightarrow \left[-\sqrt{\frac{6}{f\_in}}, + \sqrt{\frac{6}{f\_in}} \right]$

Random initialization * $\left(\frac{1}{n}\right)$ or $\sqrt{\frac{1}{n}}$

(var) (std)

Normal $\propto \frac{1}{\text{var}}$

n = input to the Node