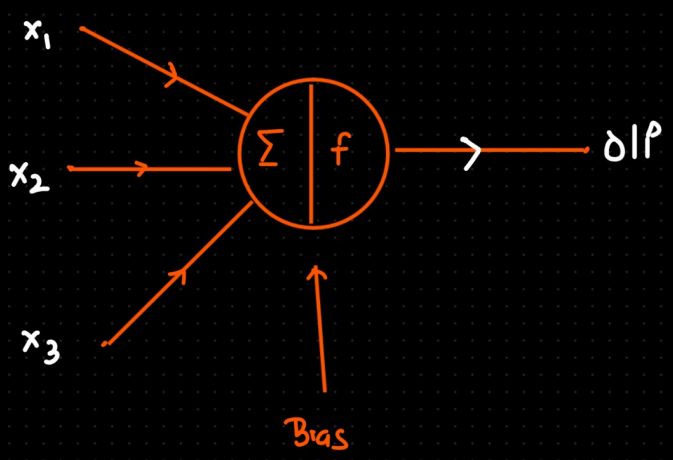
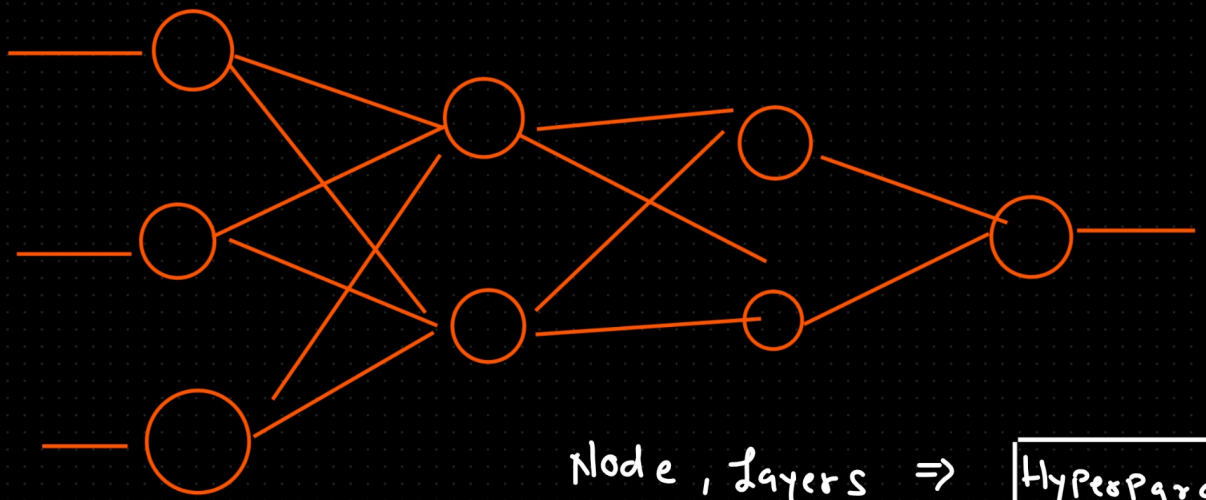
Optimizers

≥ Gradient Decent

Perceptron





Node, Layers ⇒ HyperParameter

= Compiled Neural Network

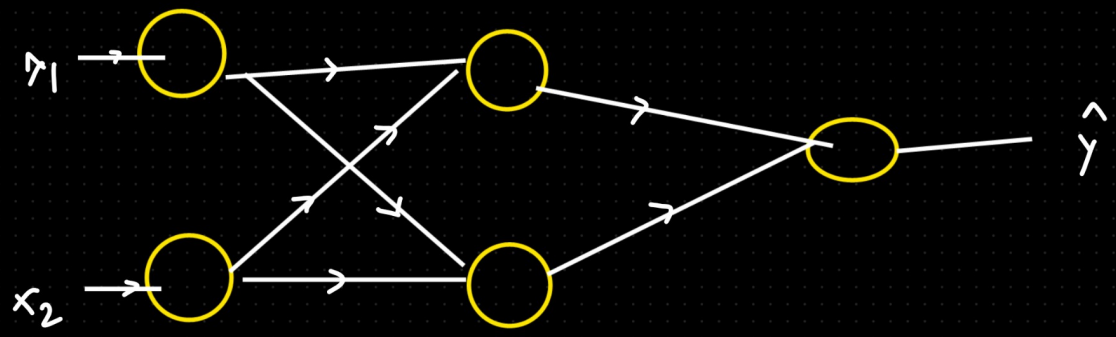
Performance of Neural Network might Decrease → ?

training time will be increased

1. Optimized weight. initialized.
2. Batch Normalization
3. Right selection of A/C fun
4. optimizer → Right optimizer

Performance of Deep NN

= Decrease the training time



$2 \times 2 + 2$ (6) + $2 \times 1 + 1$ (3) = 9

NN

= 6 weights and 3 biases

= Loss !!!

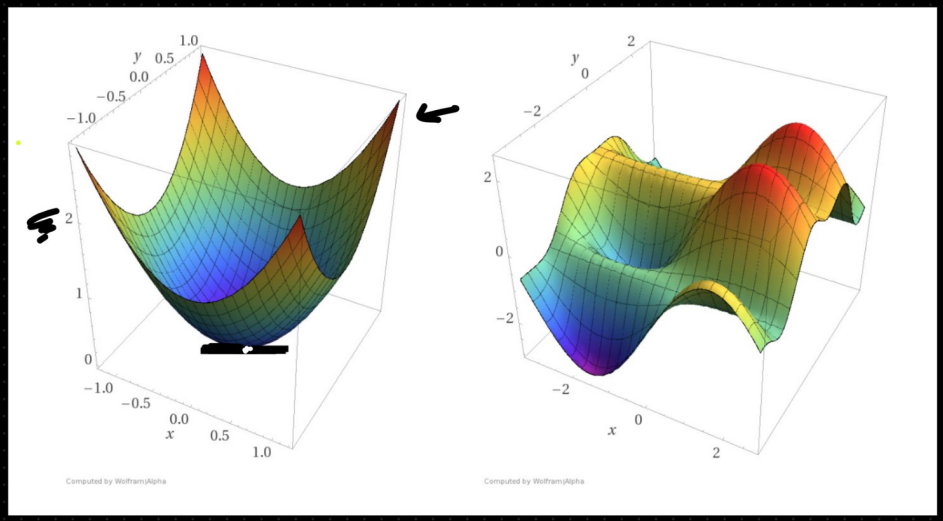
$$\mathcal{L}(\text{weights}, \text{biases})$$

$$(y - \hat{y})^2$$

$y = f(x)$

$y = x^2$

GD



NN ⇒ forward Propagation ⇒ Loss ⇒ minimise the loss ⇒ Back Propagation ⇒ Update the trainable Param

Optimizer = Gradient Descent

G.D

$$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b}$$

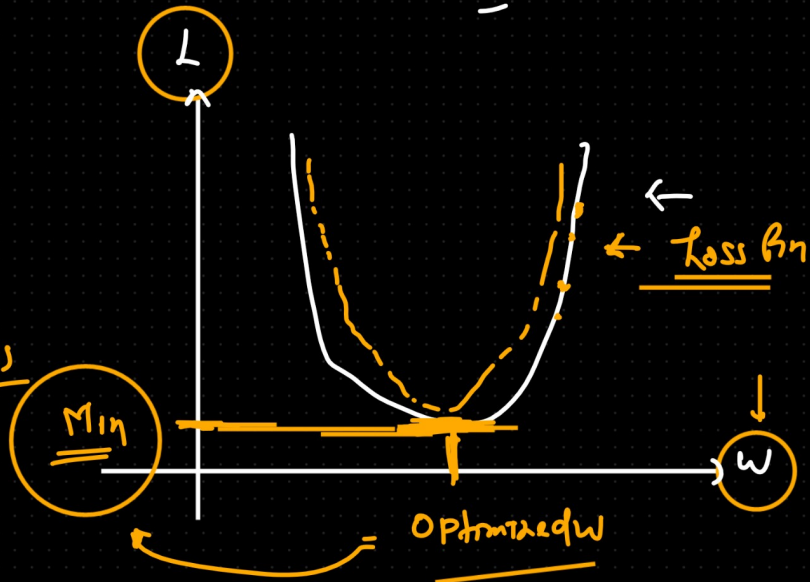
bias

G.D

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

W_{old} ← W_{new}

Loss fn ← optimizer ↑ optimized trainable parameter



Optimizer ⇒ to optimize the trainable parameter ⇒ minimum Loss

= (1) Gradient Decent ⇒ Vanila GD or batch GD.

→ Stochastic G.D.

→ Mini batch G.D.

Gradient Decent (Vanila G.D.)

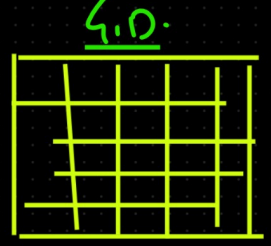
StochasticGD

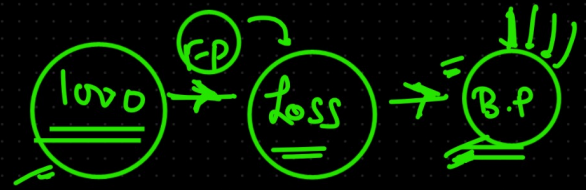
Mini batch G.D.

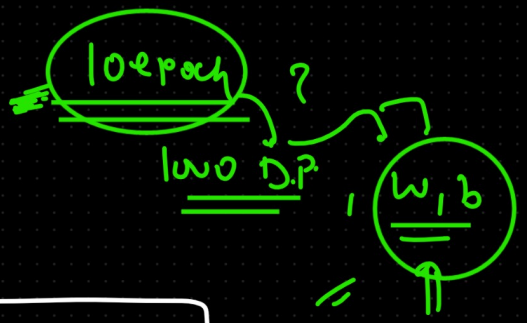
Data ⇒ NN ⇒ OIP ⇒ Loss(y-ŷ) → BP → FP

minimum.

→ N Number of Rows ⇒

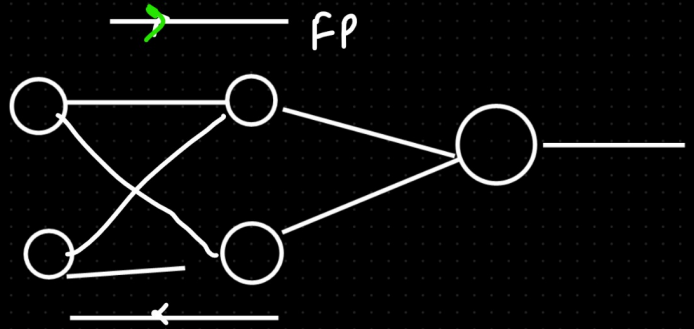






Epoch ⇒ 1 Forward Propogation + 1 Backward Propogation

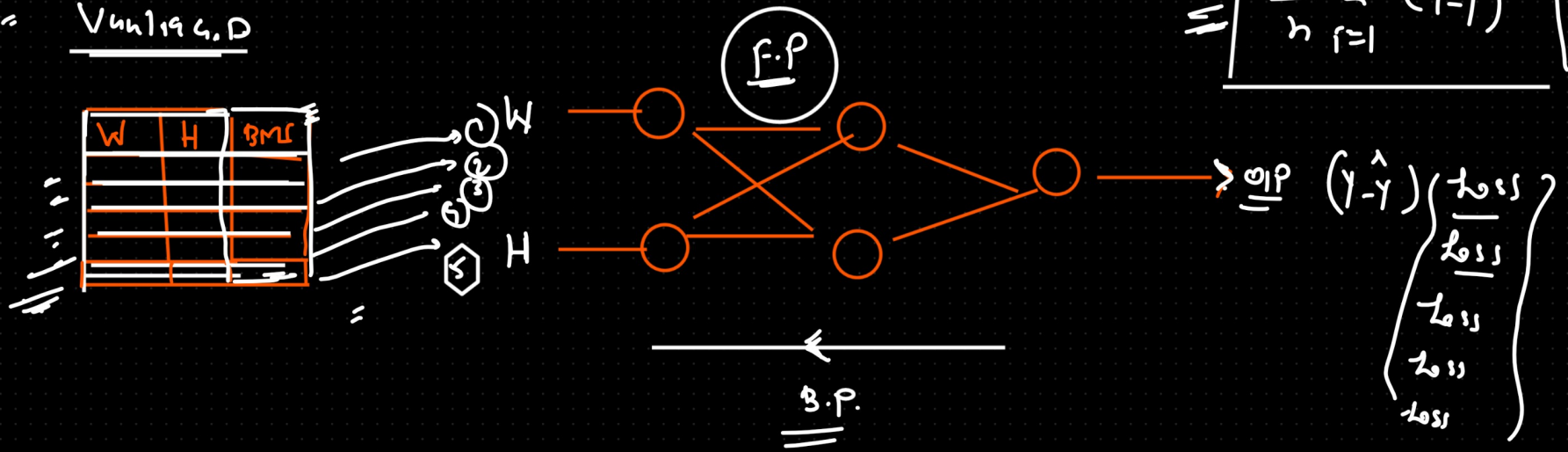
10 times



1 epoch

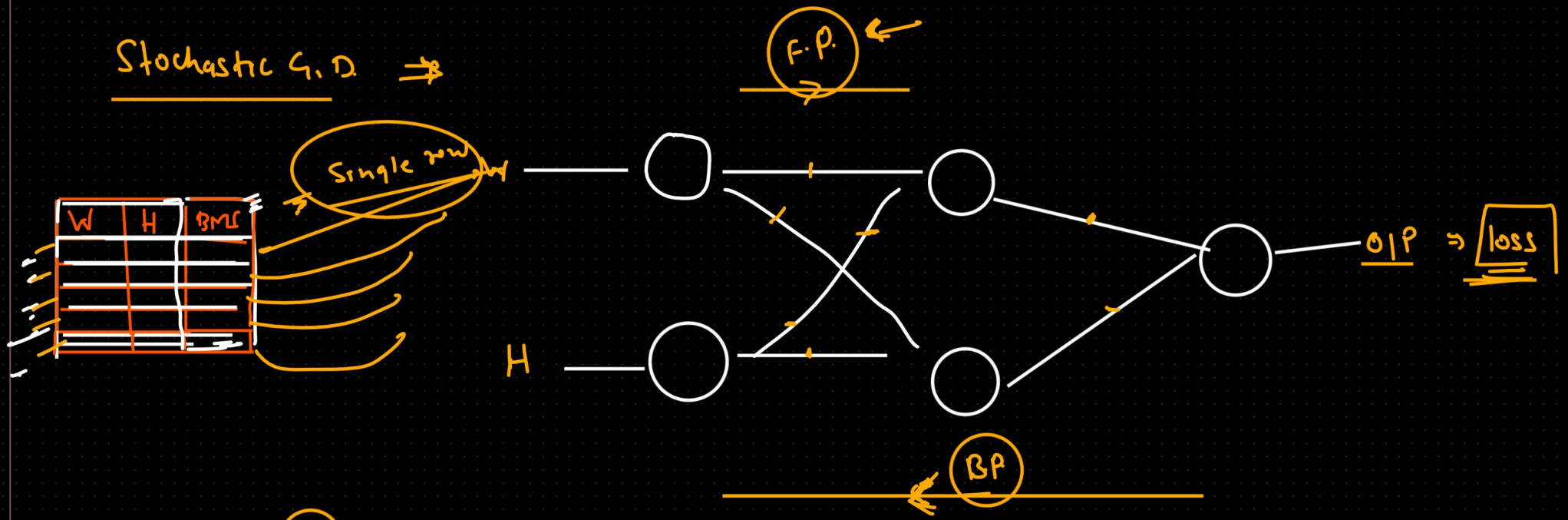
Vanilla G.D

$$\frac{1}{n}\sum_{i=1}^{n}(y-\hat{y})$$



1 = FP + BP ? = 5 epoch = 5 times

Stochastic G.D ⇒



Update the weight

1 Single epoch ⇒ 5 times

5 iteration

Single 1 epoch ⇒ F.P. + B.P

entire Data

5 iteration

: SGD ⇒ Single point

N ⇒ N iteration

1 epoch ⇒ F.P + B.P

Entire Data

5 epoch → 5 iteration ≠ 5 x 5 = 25

Summary → Vanila GD ⇒ Entire Data ⇒ NN

1 epoch = FP + BP

Entire Data

Single row

SGD ⇒ One point at a time

1 epoch ⇒ n iteration

↓

- n time weight will be updated

1 Which is faster ?

Vanilla G.D.

2 Memory consumption

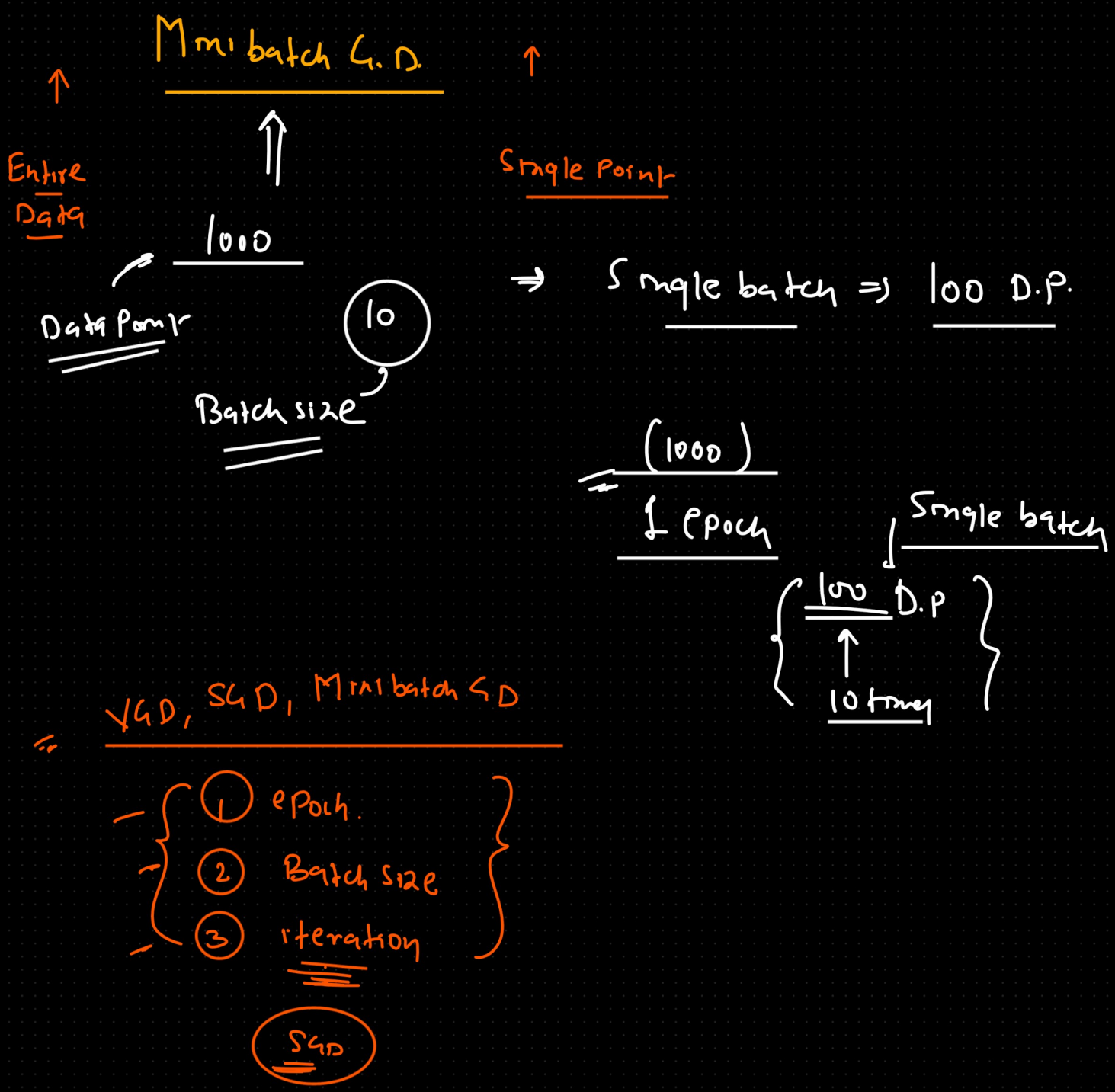
lcr ⇒ Vanilla G.D ↑ ⇒ high memory

RAM. ⇒ S.G.D. ↓ ⇒ less memory

one point at a time

3 Good accuracy

= SGD ⇒ 10 x n ⇒ n|0 ⇒ !!! weights ams to update


Mini batch G.D.
Entire Data
1000
Data Point
10
Batch size
Single Point
Single batch => 100 D.P.
(1000)
1 Epoch
Single batch
100 D.P
10 times
VGD, SGD, Mini batch GD
1 Epoch.
2 Batch size
3 Iteration
SGD

1000

VGD

1000 ⇒ NN

1 epoch ⇒ 1000 → FP
← BP

5 epoch ⇒ 5 times

Minibatch

~~Entire~~
~~Single~~

Batch

Batch

1000 ⇒ 10 ⇒ 100

(1000) / 1 epoch ⇒ 100, 100, 100, ..., 10

(1000) / 2 epoch ⇒ 100, 100, 100, ..., 10

3 epoch ⇒ 100, 100, 100

epoch
Batch size
iteration

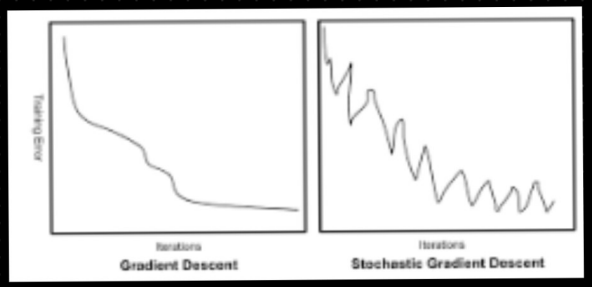
1000

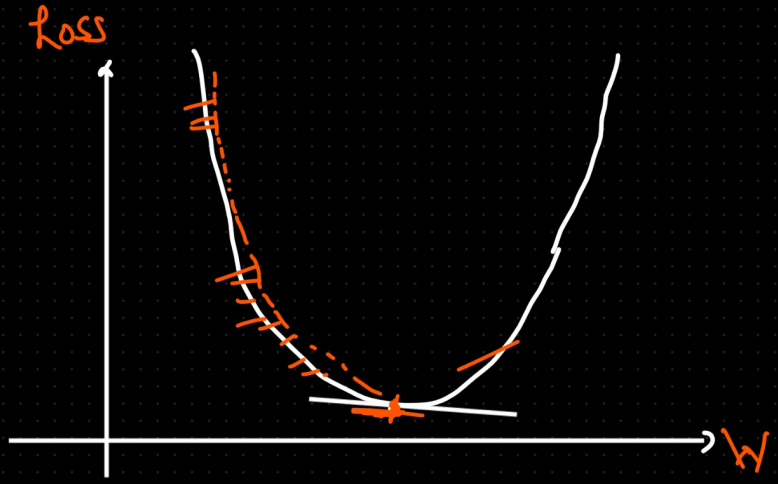
SGD

1 Data point

1 epoch ⇒ 1000

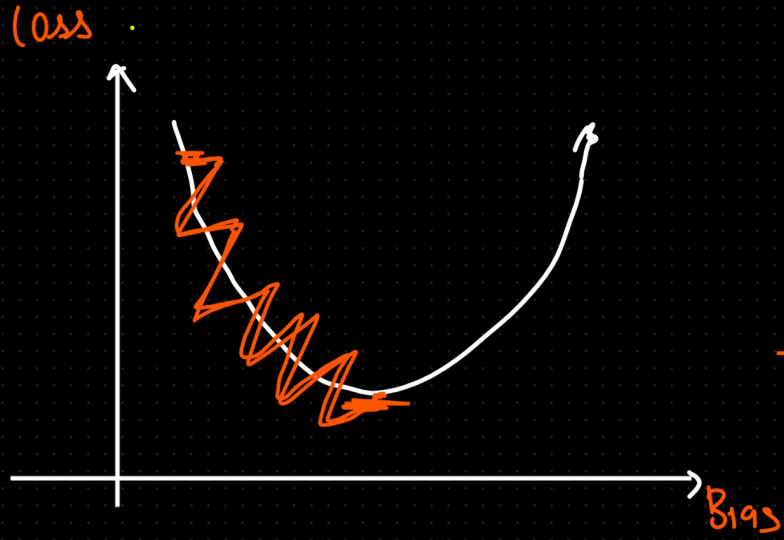
5 epoch ⇒ 5 × 1000 / 5000





100

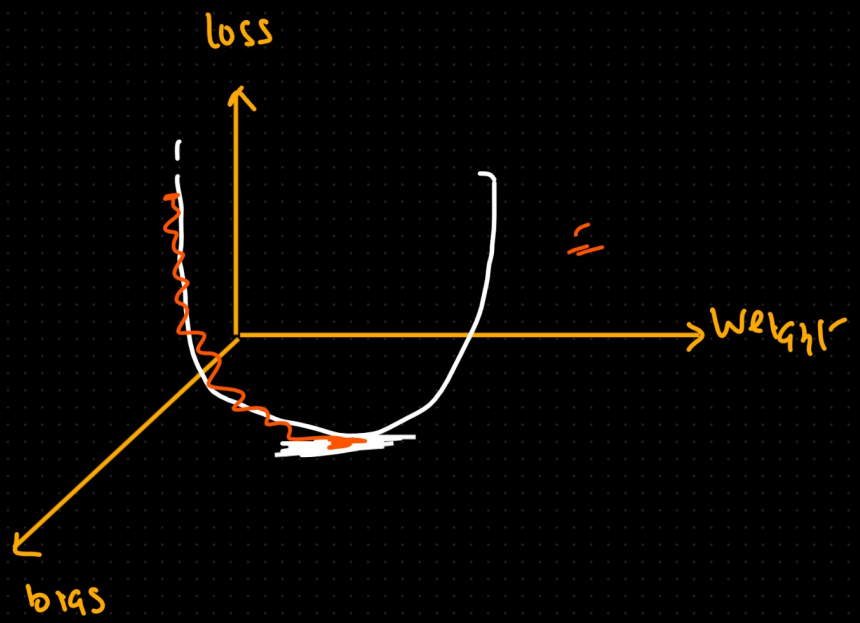
10

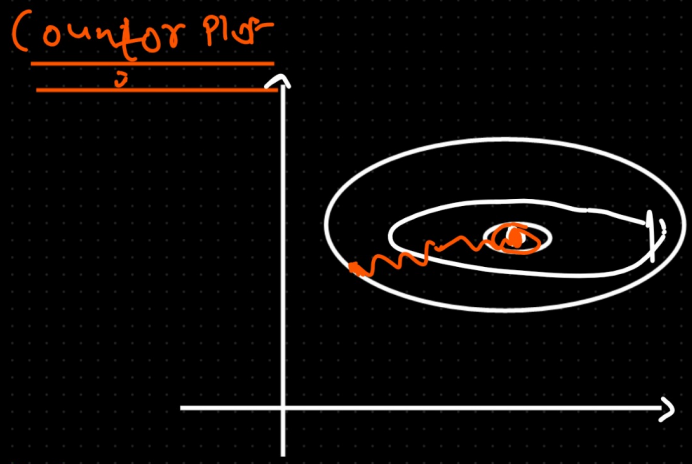


Single D.P Point

10 epoch

1 → 100

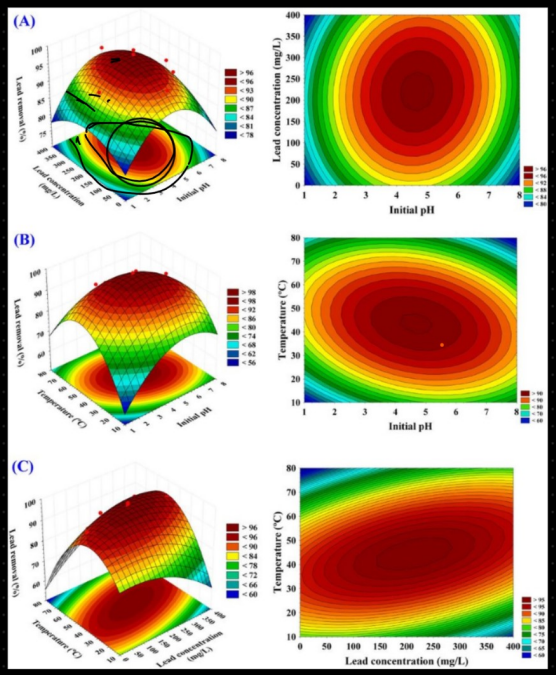


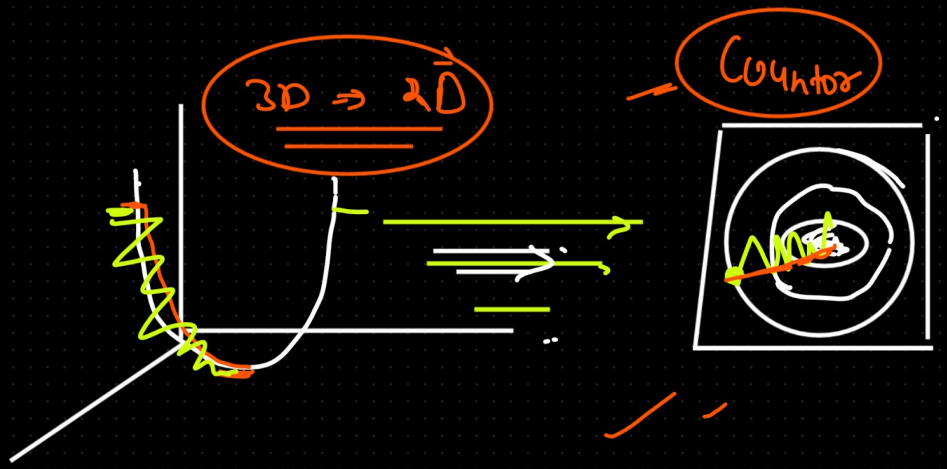


3-D graph ⇒ 2D

3-D graph

3 Params (3 axis)



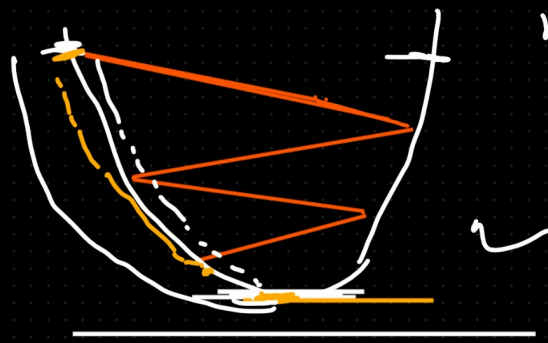


Challanges with G.D.

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

$\eta = 1, 2, 3 \ldots$

1. Learning rate
2. Local minima Prob.
3. Sadal Point

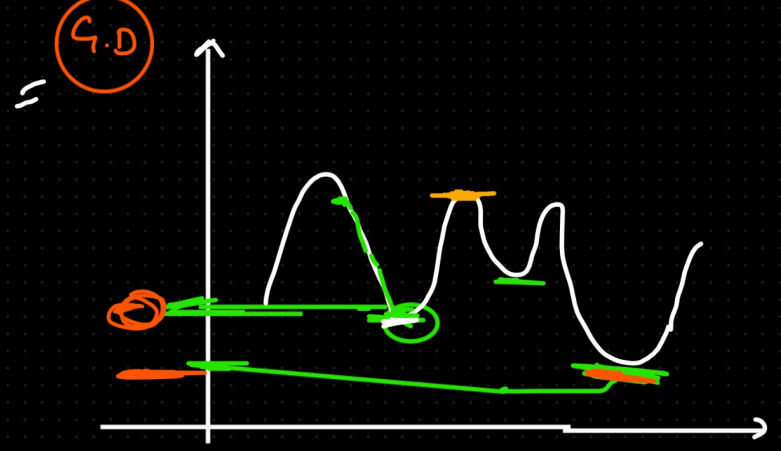


$\eta = 0.001$
0.01

$\eta = 0.1, 0.2$

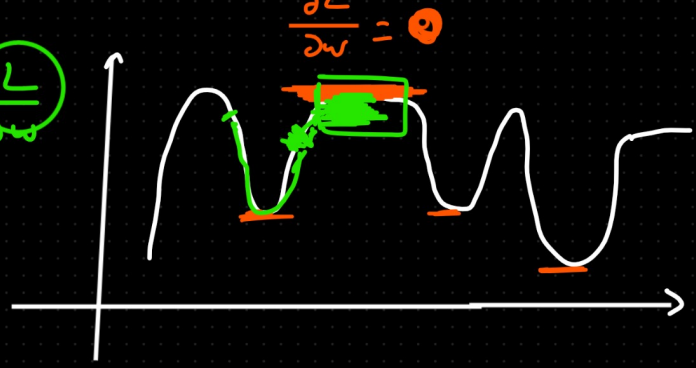
Changes G.D.

1. SGD with Momentum
2. Adagrade
3. NAG
4. RMS Prop
5. Adam



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

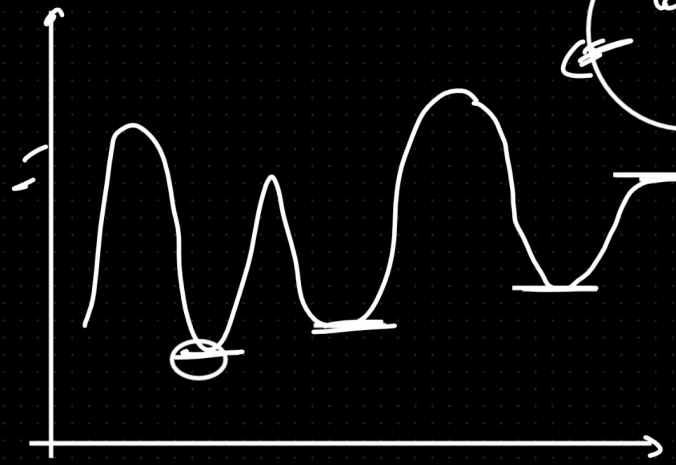
No weight updation



Non-convex curve

Optimizer ?

Loss

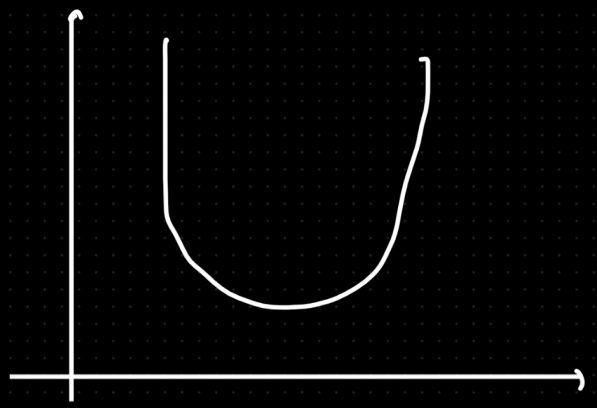


best ~~model~~

Optimizer

Convex

MSE



EWMA AND SMA

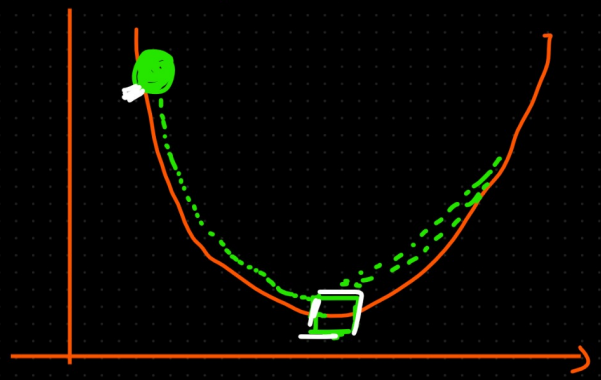
SGD with Momentum

?

Velocity

Mew

Increase or Decrease



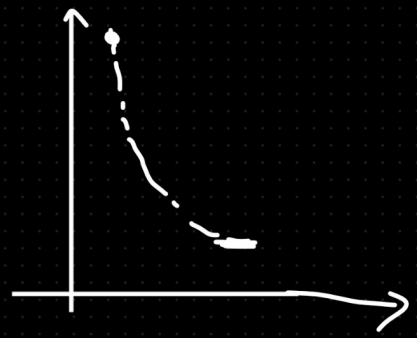
Contribution from Previous time stamp

Phy

M

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$$

$$w_{t+1} = w_t - \eta \frac{\partial L}{\partial w}$$



$$w_{t+1} = w_t - v_t$$

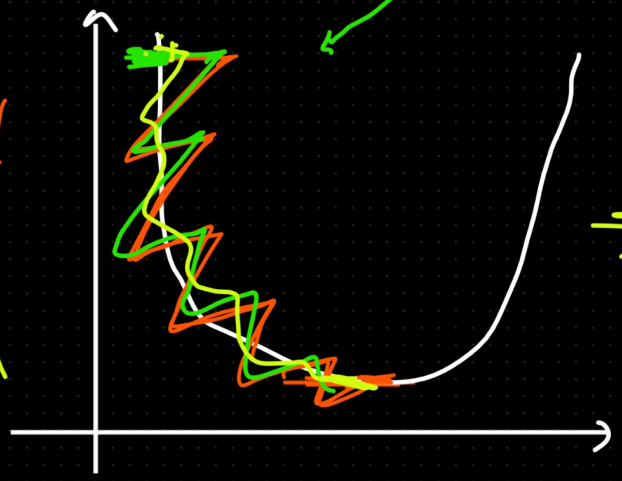
$$v_t = \beta \times v_{t-1} + \eta \frac{\partial L}{\partial w_t}$$

momentum

SGD | Mini Batch

momentum

Noise



SGD with momentum

considering past history

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$$

$\therefore$ Reducing the Noise

Thursday -> keras tuner, tensorflow 2x

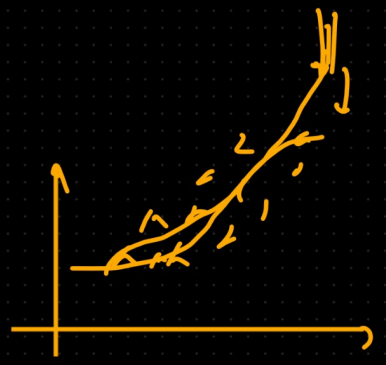
8-10

Sat, Sun -> weight init-

batch Norm, reg, dropout

CNN => pytorch

SMA



TS => Time depedt thing