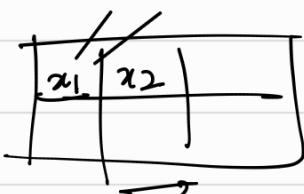
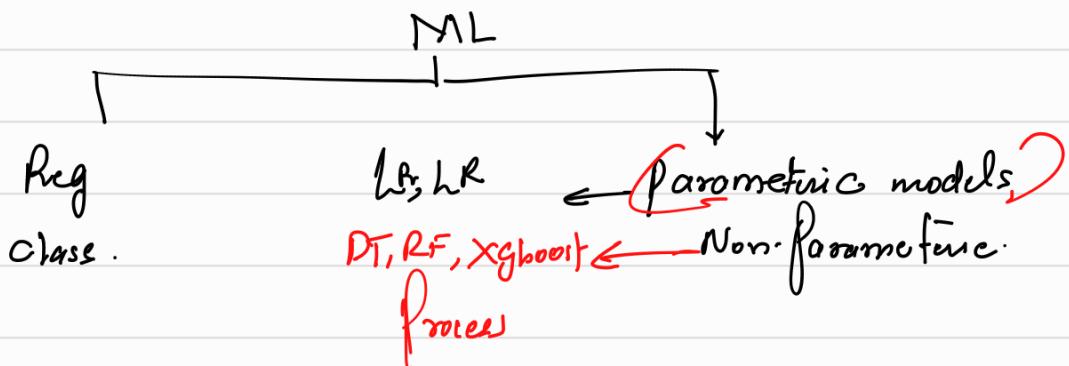


Deep learning

$$y = f(x) + \epsilon$$

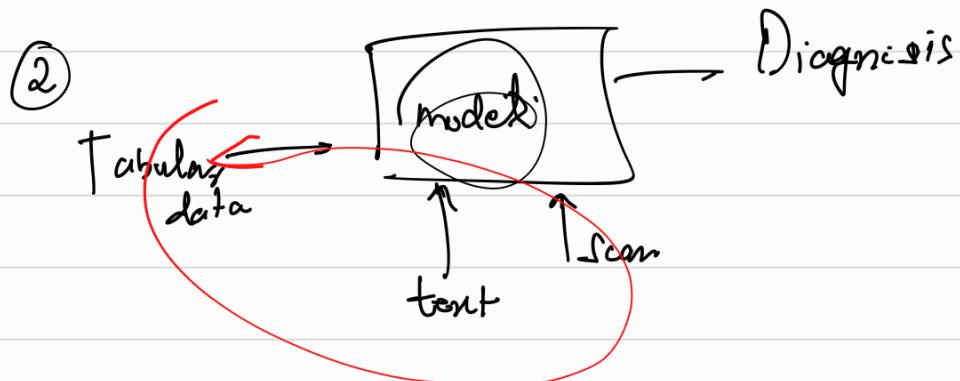


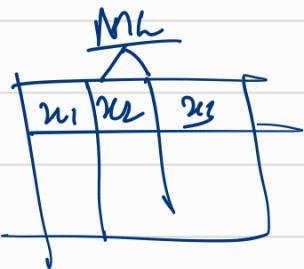
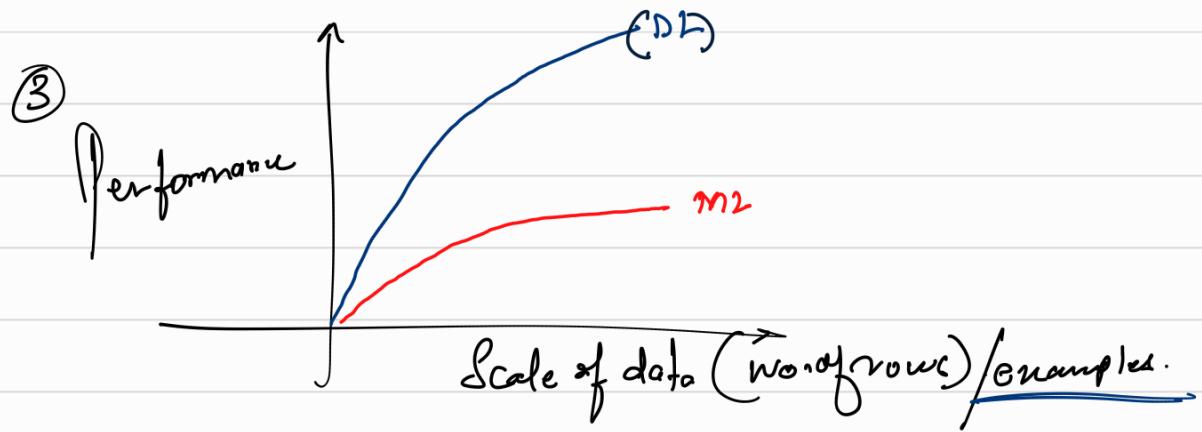
$$y \sim f(x)$$

①
architecture

$x \rightarrow$ (video, audio, text, tabular data, graph).

multimodality



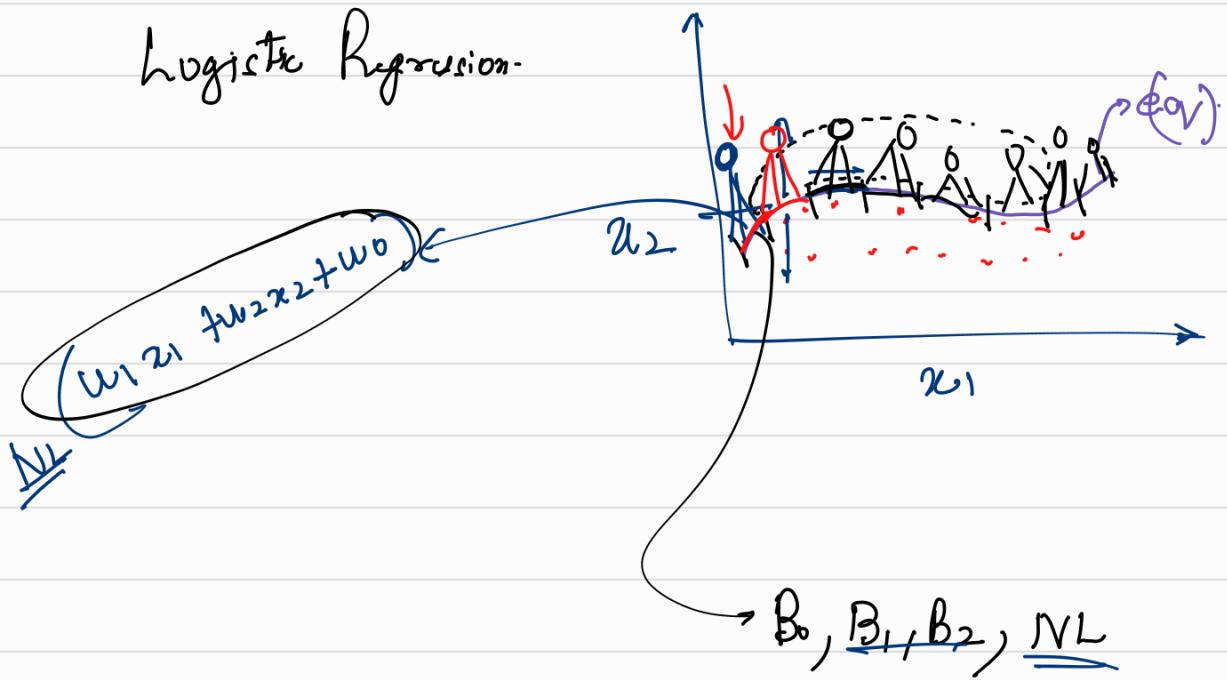


D_h → automated feature engineering

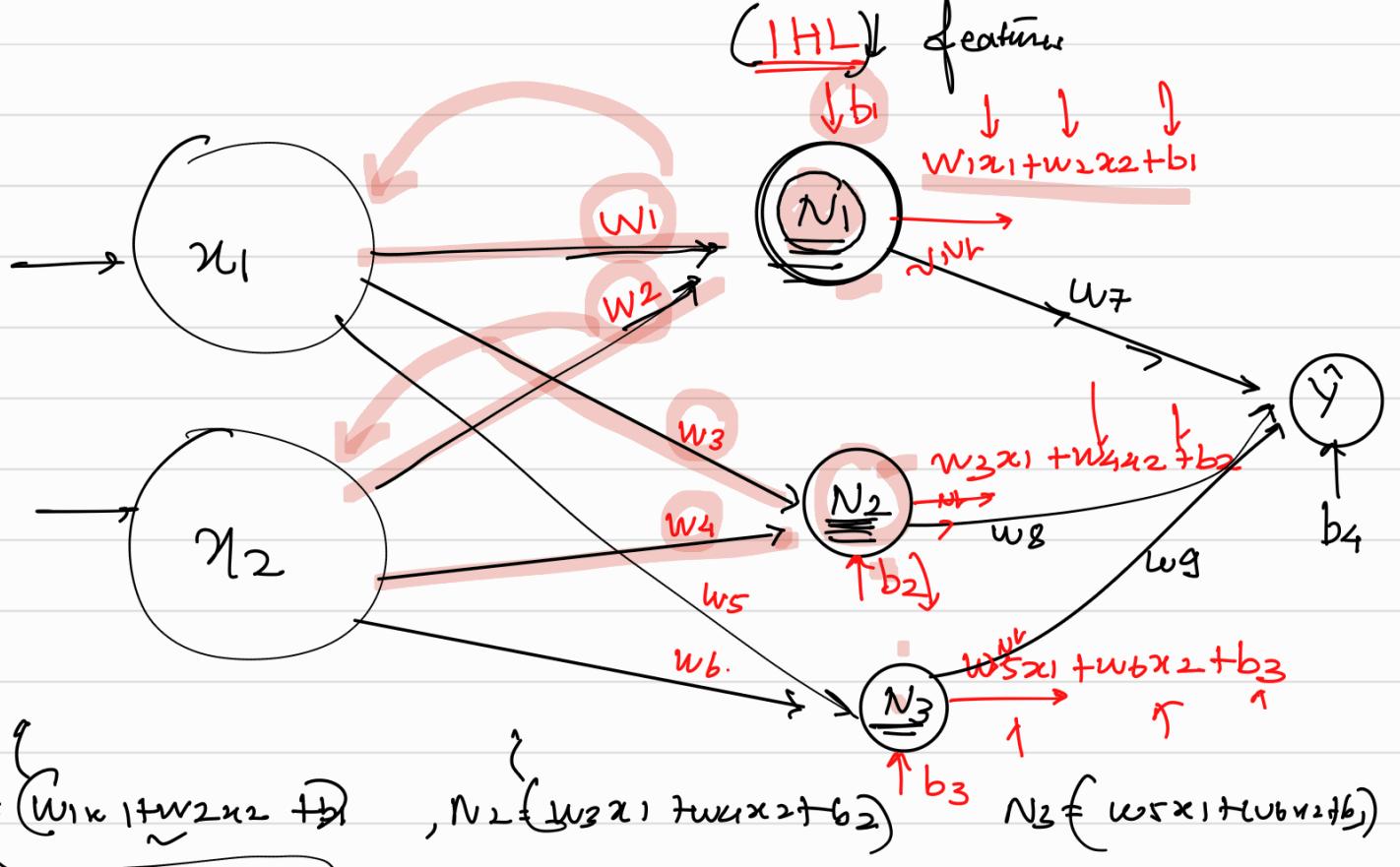


What is a Neuron

logistic Regression-



w_1	x_2	y

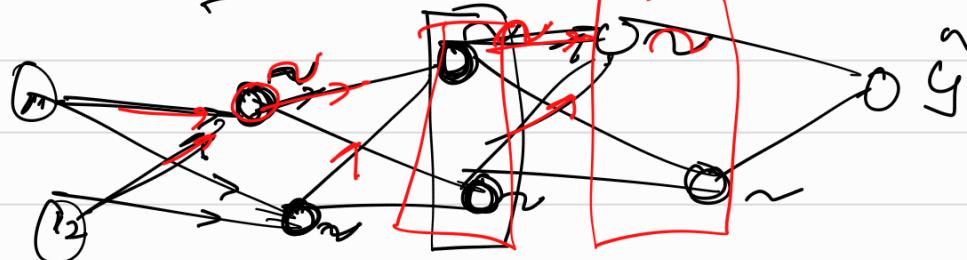


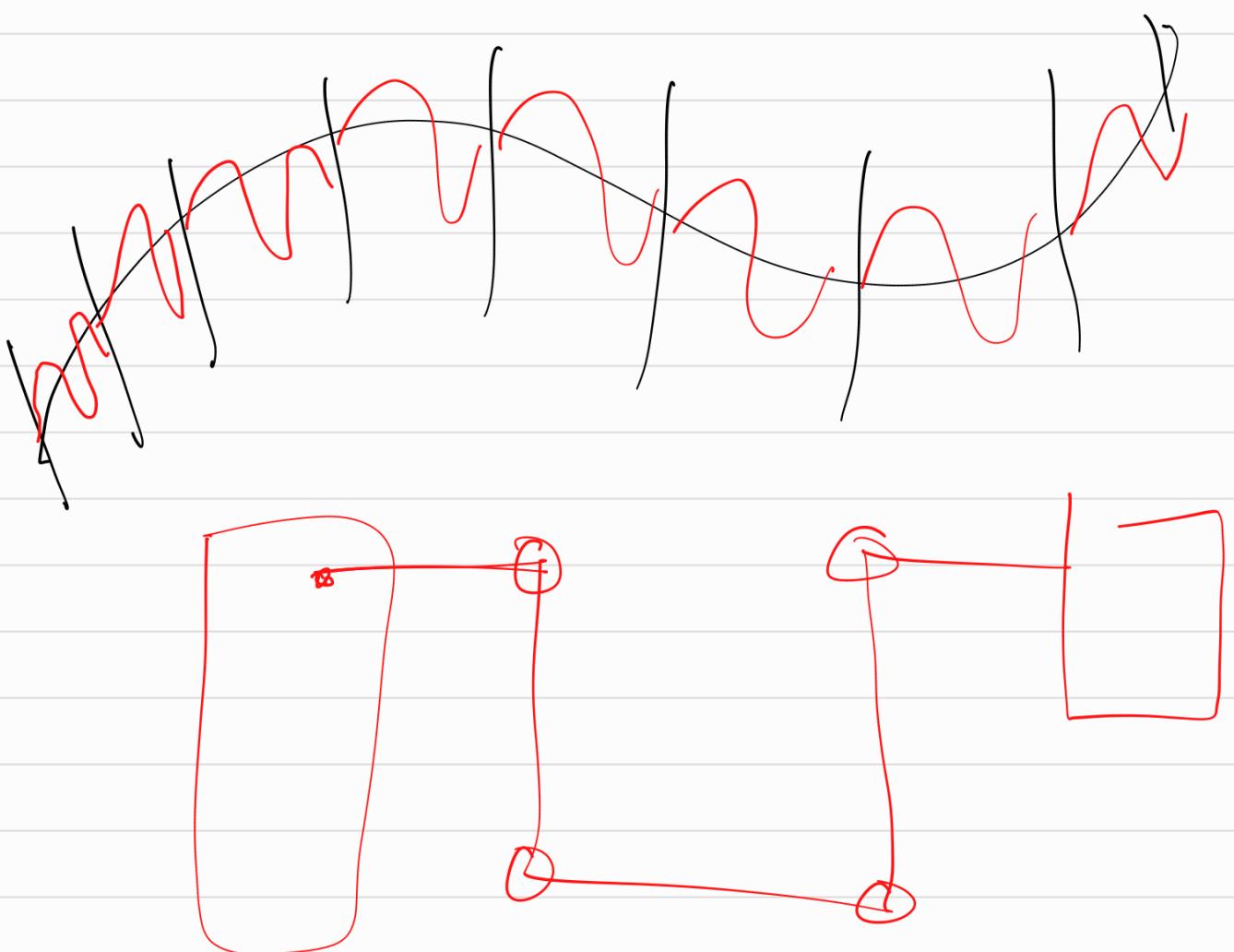
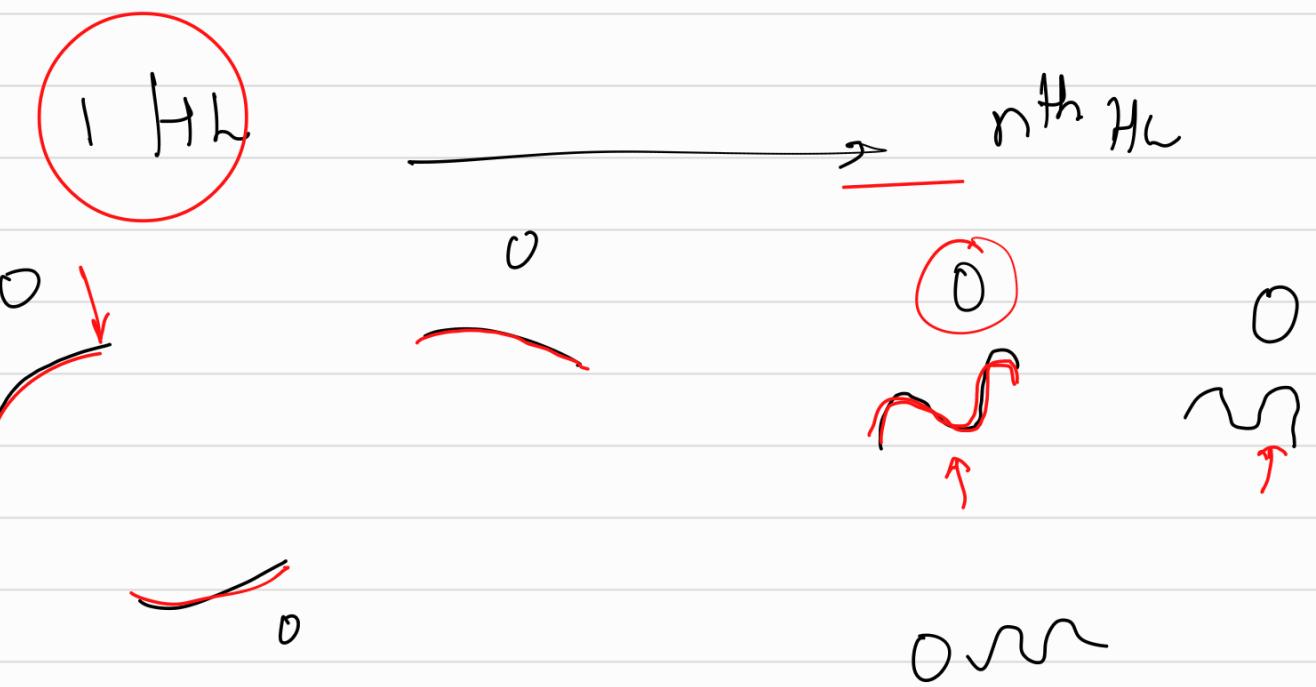
$$\Rightarrow \hat{y} = (\underline{w_7}) (\underline{w_1 x_1 + w_2 x_2 + b_1}) + \underline{w_8} (\underline{w_3 x_1 + w_4 x_2 + b_2}) + \underline{w_9} (\underline{w_5 x_1 + w_6 x_2 + b_3}) + b_4$$

$$\Rightarrow \hat{y} = x_1 (w_7 w_1 + w_8 w_3 + w_9 w_5) + x_2 (w_7 w_2 + w_8 w_4 + w_9 w_6)$$

$$+ w_7 b_1 + w_8 b_2 + w_9 b_3 + b_4$$

$$\Rightarrow \hat{y} = A x_1 + B x_2 + C$$



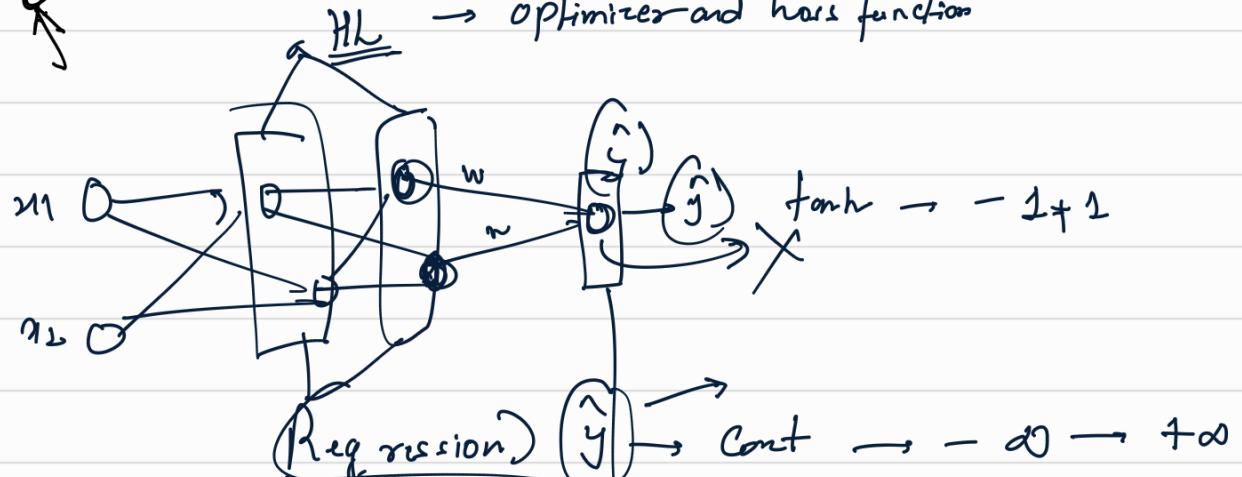
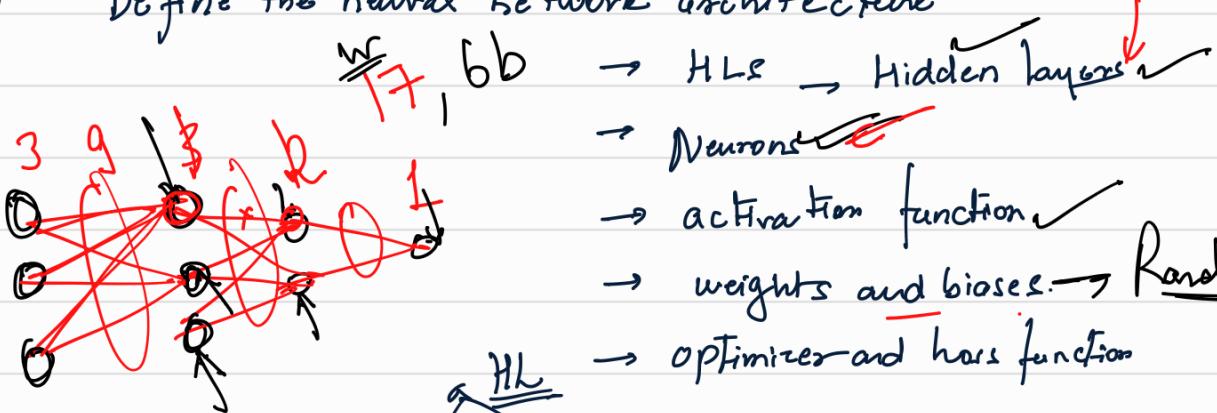


Gradient Descent

: minimise the loss by updating the W, B .
for all neurons.

RanCham

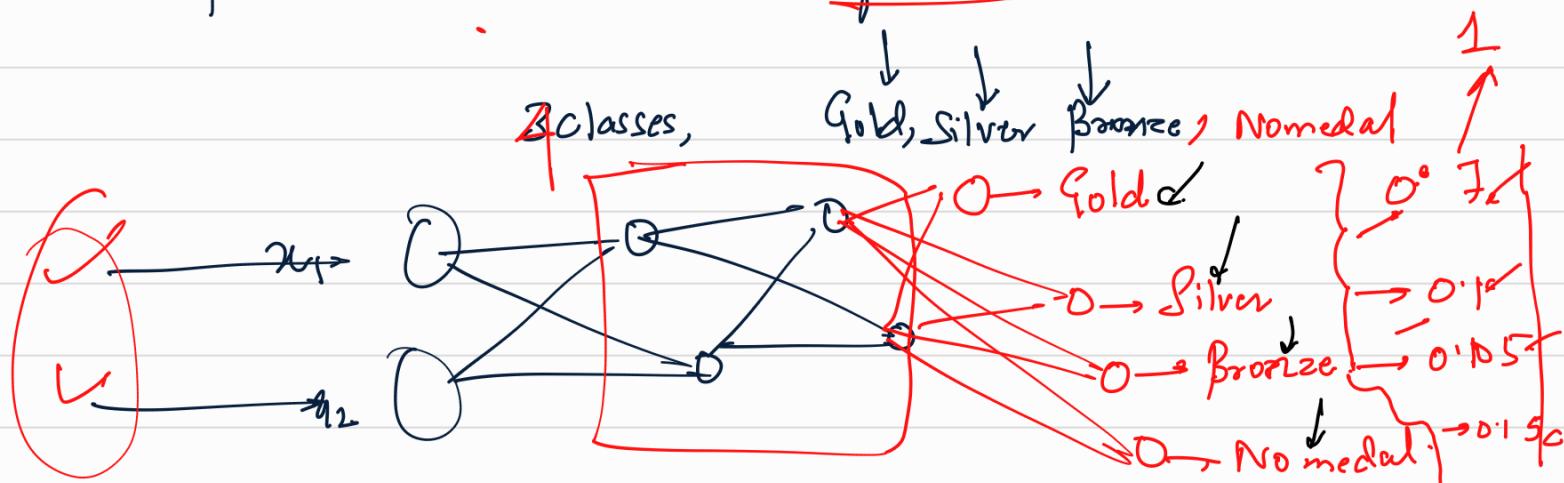
1) Define the neural network architecture

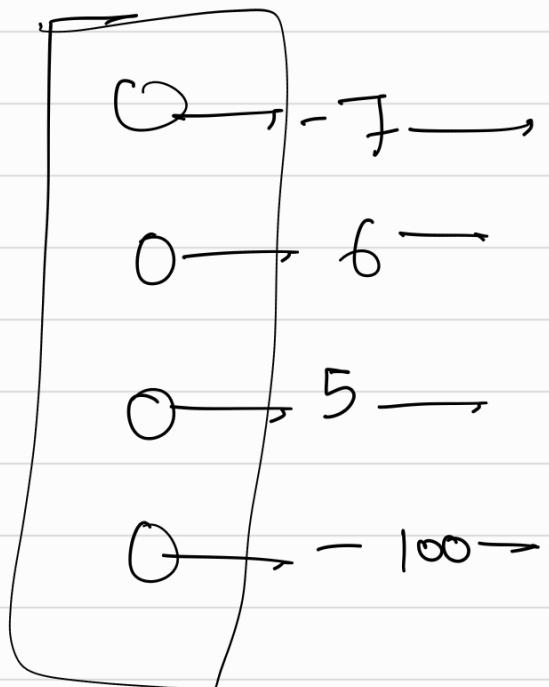
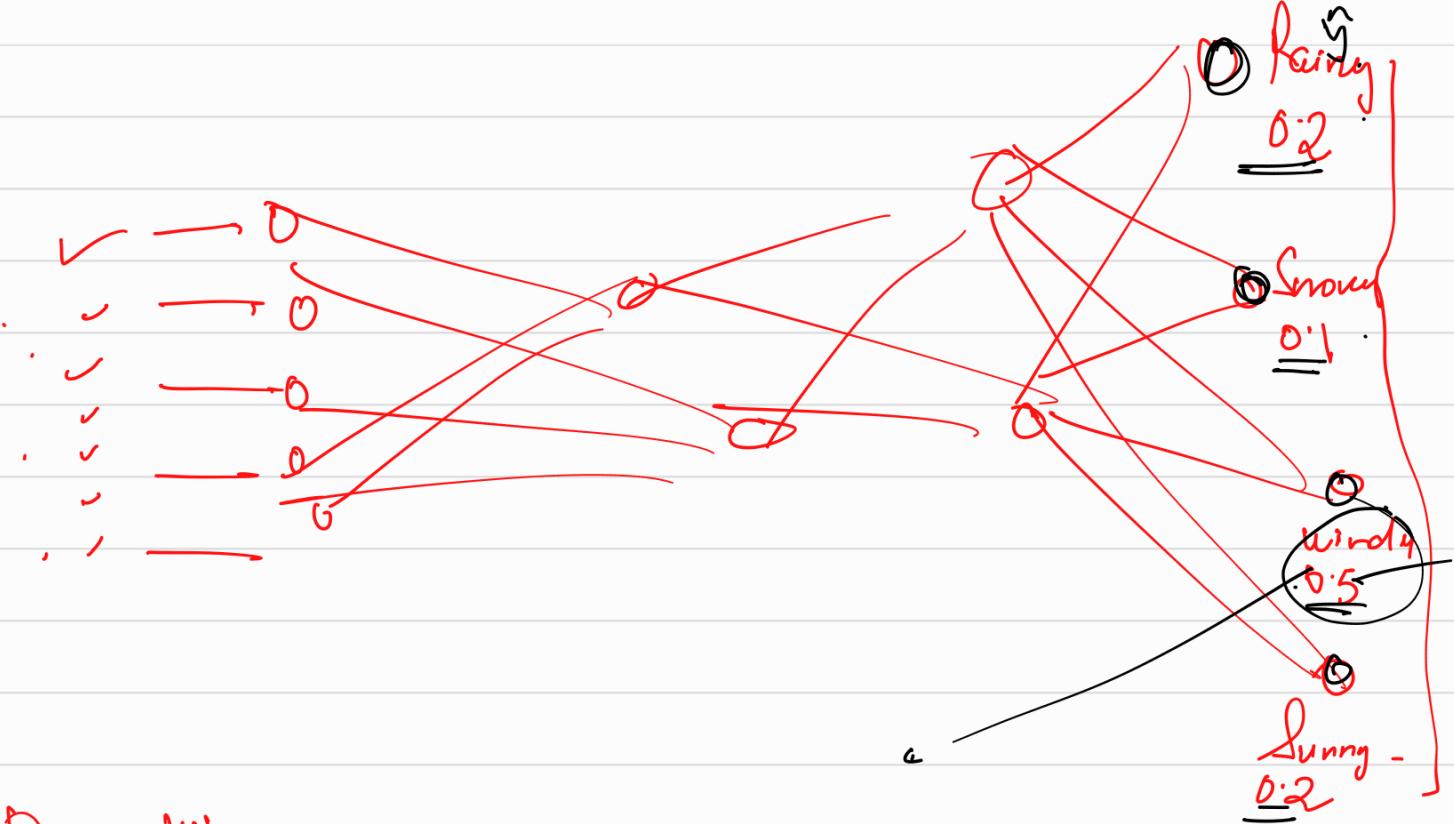


Classification (Binary). $\rightarrow \text{Sig}(\hat{y}) \rightarrow 0-1$

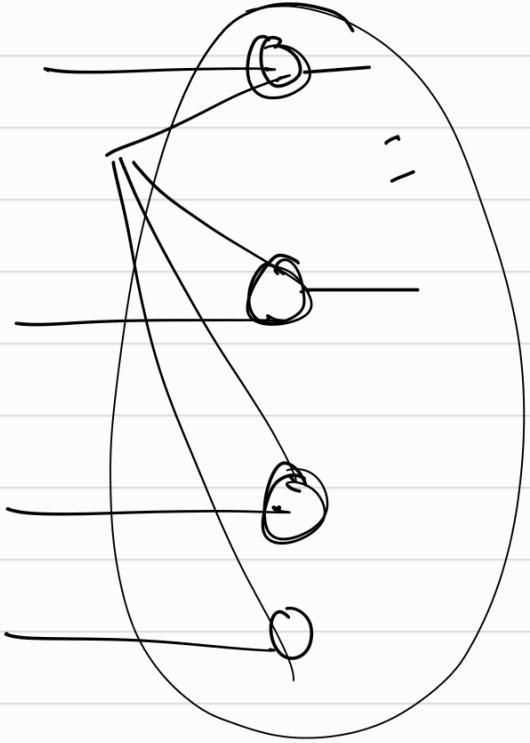
Classification (multiclass).

Softmax (\hat{y}):





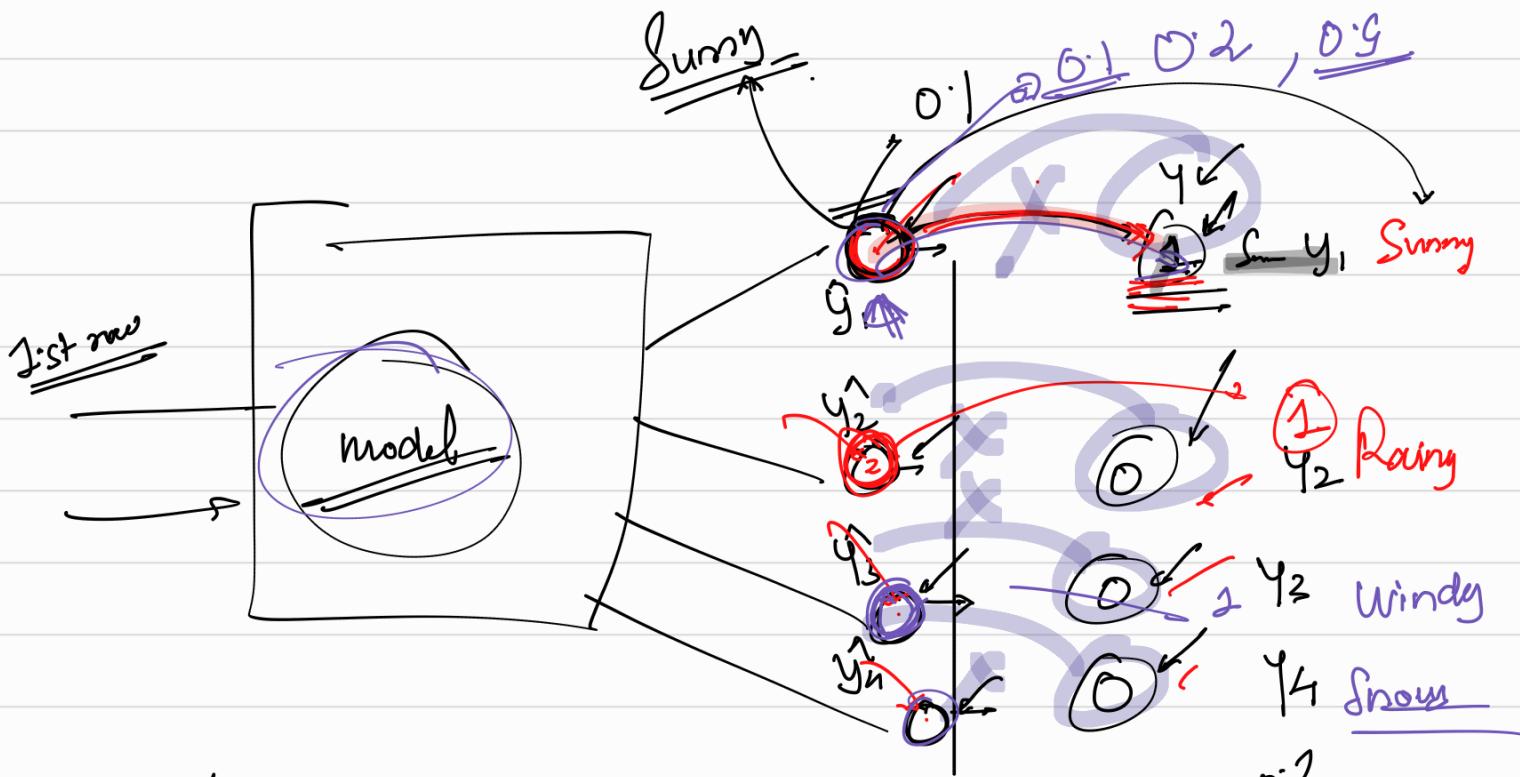
x_1	x_2	Weather (y)	y_{-Sw}	y_{-Ra}	y_{-Wi}	y_{-Snow}
		Sunny	1	0	0	0
		Rainy	0	1	0	0
		Windy	0	0	1	0
		Snowy	1	0	0	1
		Cloudy	0	0	0	1



0

0

0
0



log loss = for all rows of data sum - $(y_1 \log(\hat{y}_1) + y_2 \log(\hat{y}_2) + y_3 \log(\hat{y}_3) + y_4 \log(\hat{y}_4))$

$$- (\log 0.2)$$

$$= \underline{\underline{0.69}}$$

$$- (\log 0.2)$$

$$= \underline{\underline{0.09}}$$

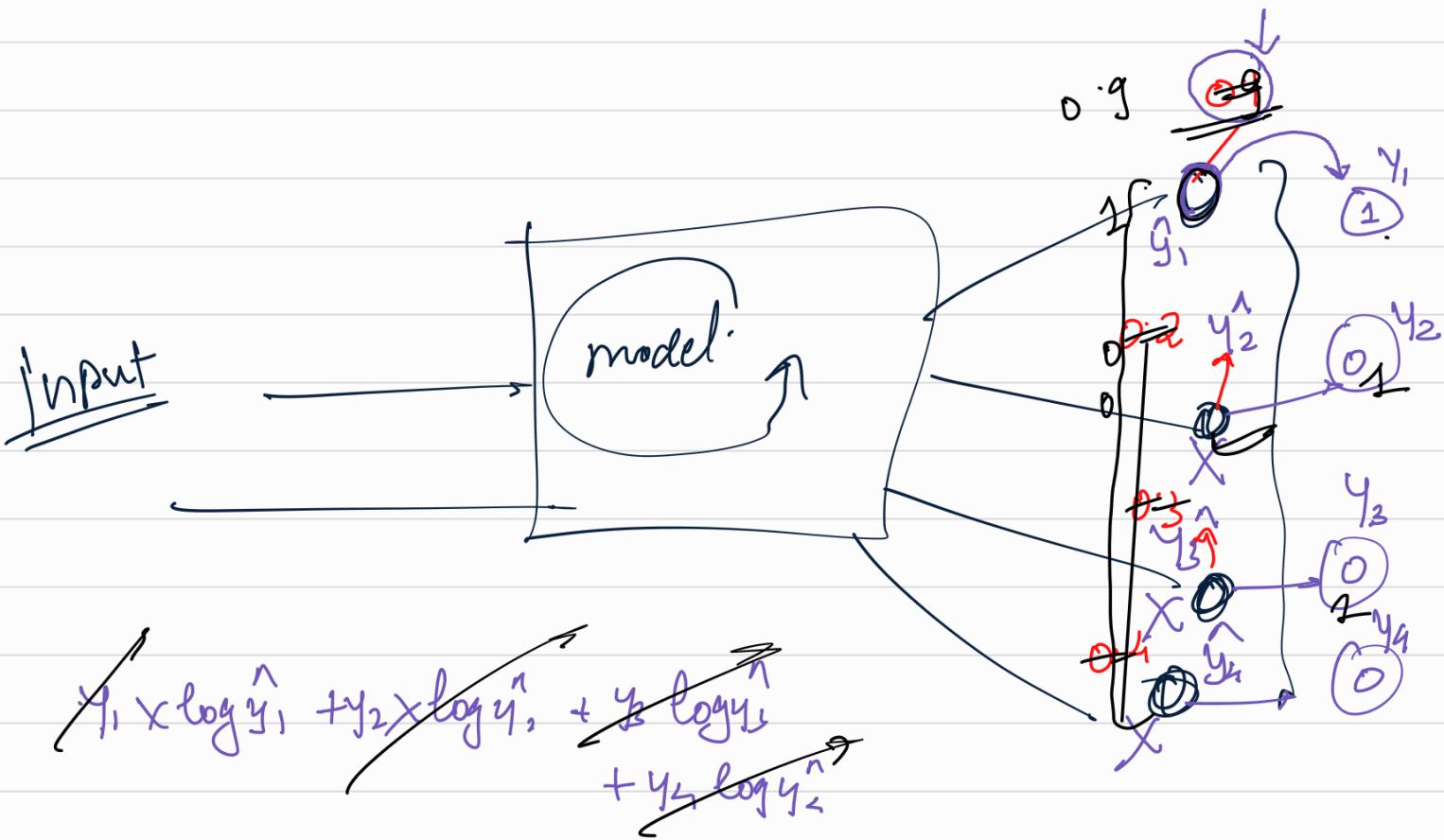
$$- \underline{\underline{\log(0.95)}} = 0.02$$

$$\log 1 = 0$$

$$y_1 \log y_1 +$$

$$y_2 \log y_2 + y_3 \log y_3 + y_n \log y_n$$

$$\log(0.1) = (-1)$$



$$-\left(\log \hat{y}_i\right)$$

$$\cancel{\log 1 = 0}$$

$$-\left(\log 0.1\right)$$

↓

$$-\left(1\right)$$

VS

VS

$$-\frac{\log 0.9}{0.045}$$

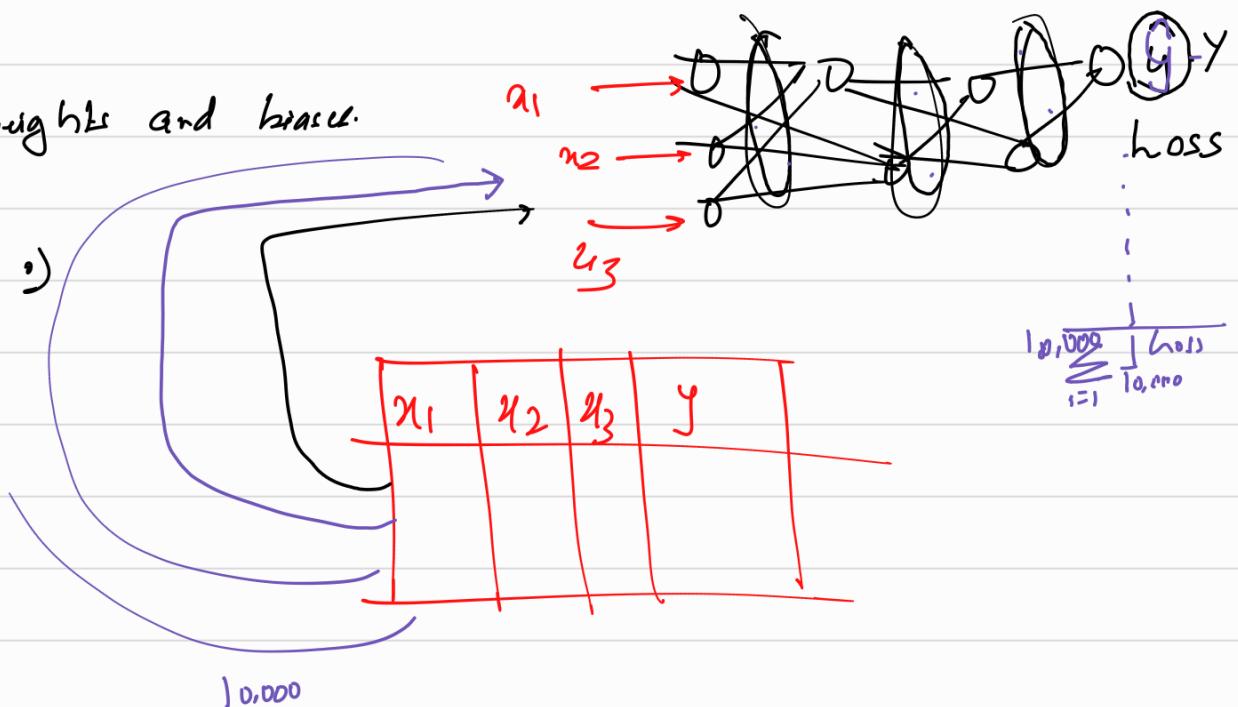
0
0
0

$$(0 \\ 0 \\ 0)$$

softmax, ohe

log loss

Random weights and biases.

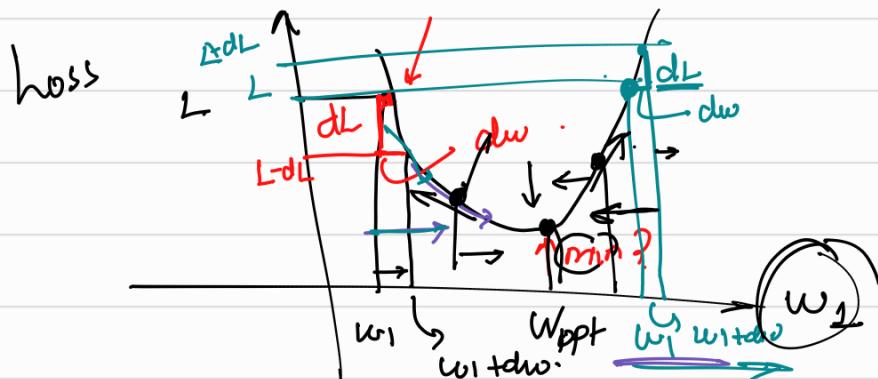
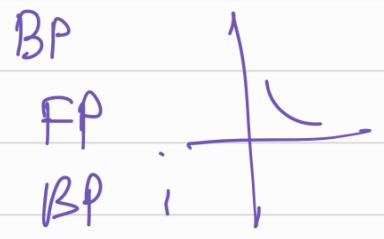




$$L = f(y_{\text{predicted}}, y_{\text{actual}})$$

$$L = f(\underline{w}, \underline{b}, \underline{z}), \underline{y_{\text{actual}}})$$

$$L = f(w, b)$$



$$\frac{dL}{dw} = -\text{ve.}$$

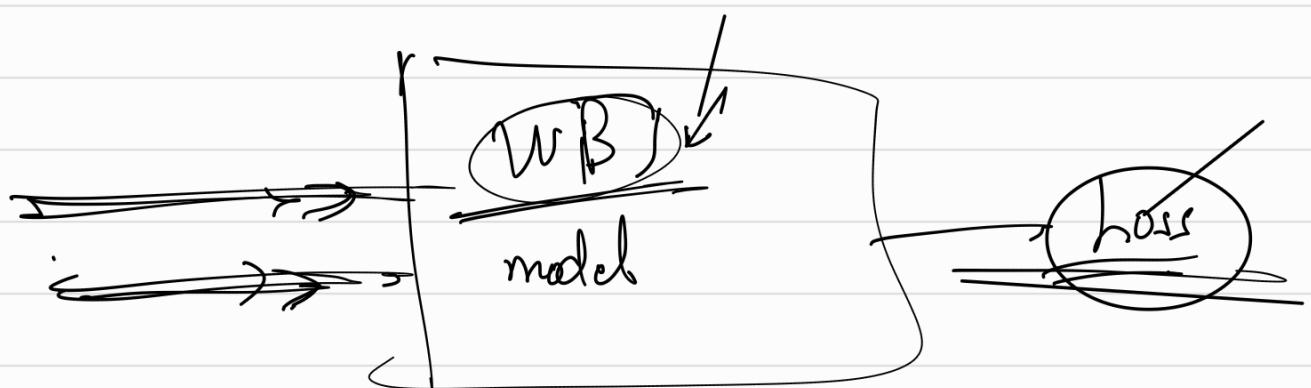
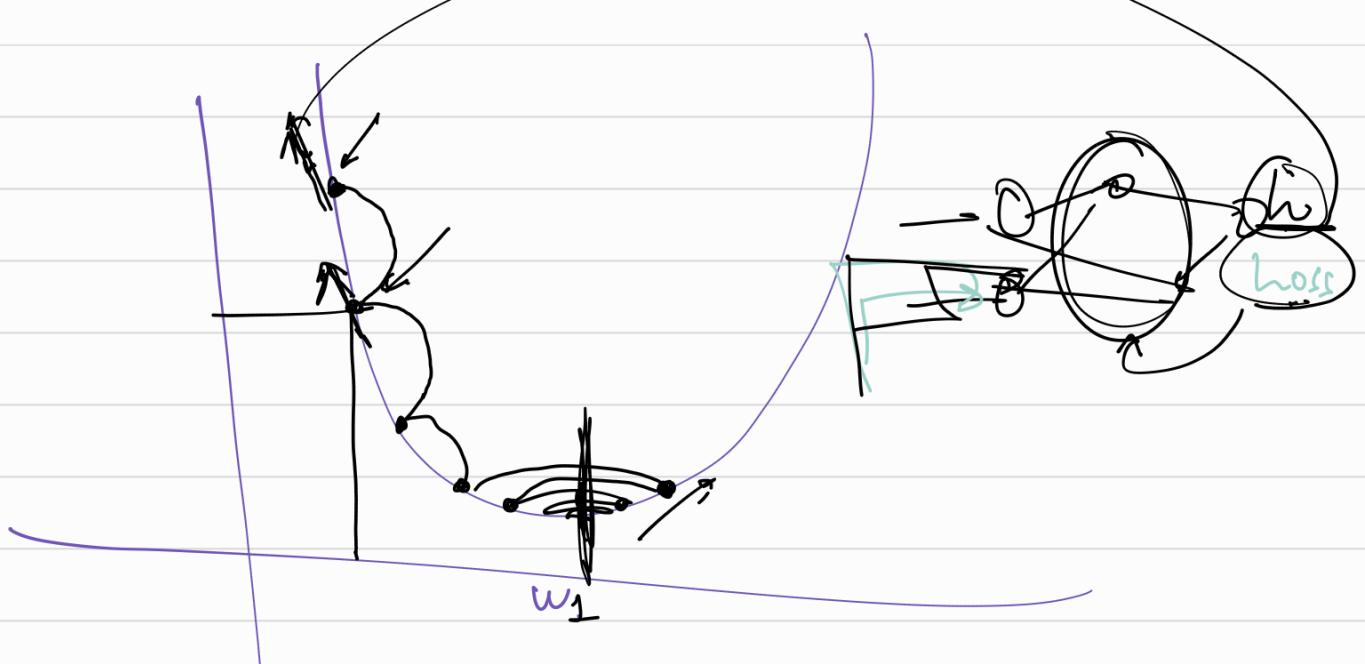
which side would +ve
move

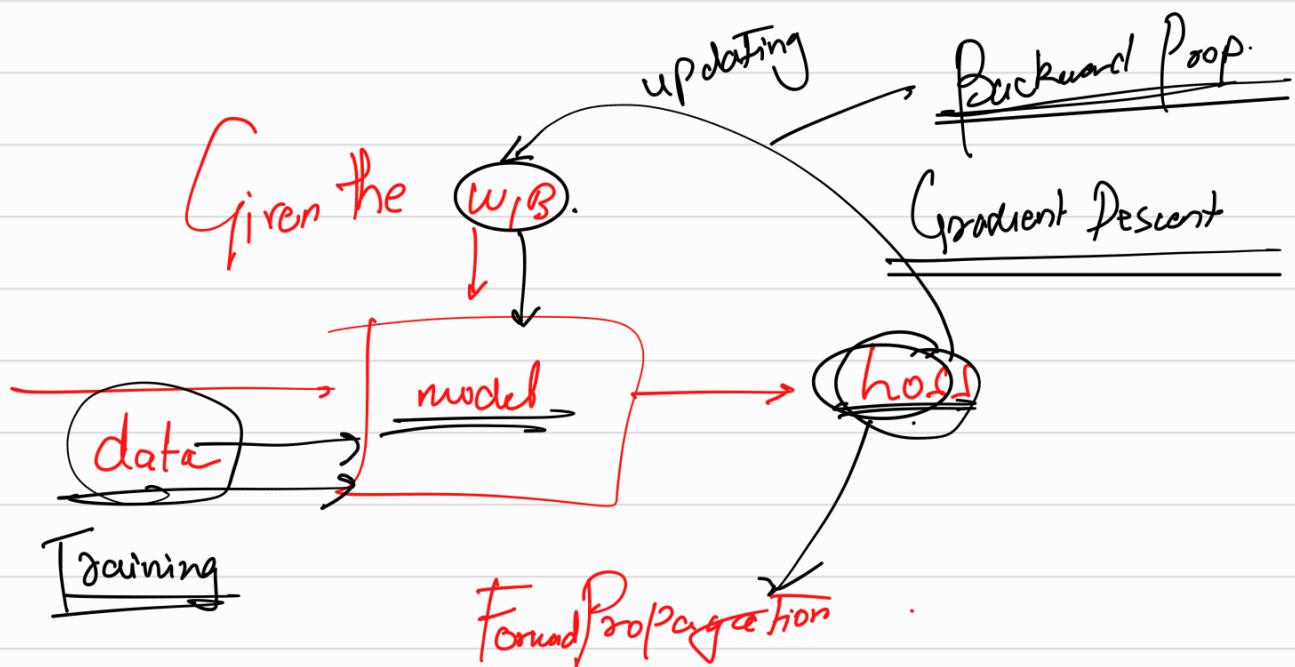
$$\frac{dL}{dw} = +\text{ve}$$

$$(w = -\text{ve.})$$

$$w_i(\text{new}) = w_i(\text{old}) - \frac{dL}{dw_i}$$

Applicable for all weights and biases.





$$(1FP + 1BP) \xrightarrow{\text{epoch}}$$

$$(1FP, 1BP) \xrightarrow{1\text{ epoch}} \text{epoch}$$

$$\boxed{1FP + 1BP}$$

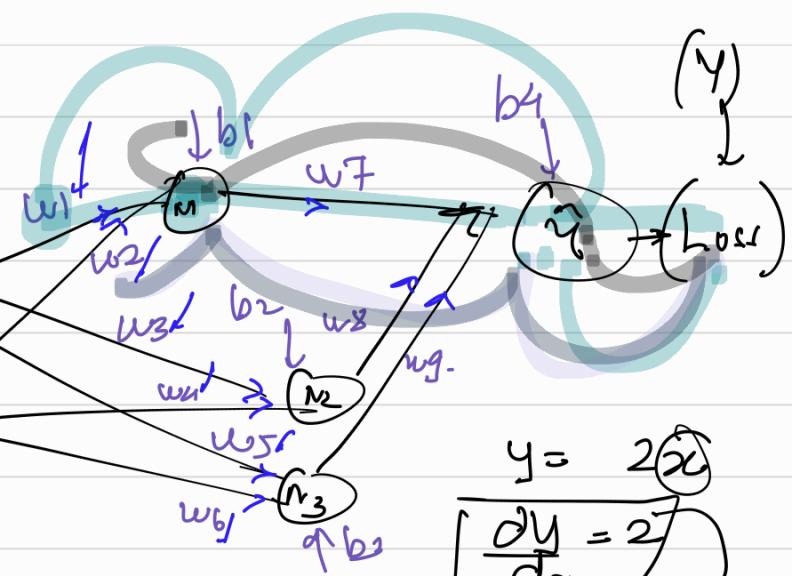
Lecture-3

$$N_1 = w_1 x_1 + w_2 x_2 + b_1 \quad (1)$$

$$N_2 = w_3 x_1 + w_4 x_2 + b_2$$

$$N_3 = w_5 x_1 + w_6 x_2 + b_3$$

$$\hat{y} = w_7 N_1 + w_8 N_2 + w_9 N_3 + b_4$$



$$y = 2x$$

$$\frac{\partial y}{\partial x} = 2$$

$$\frac{\partial L}{\partial \hat{y}} = \frac{1}{n} (y_i - \hat{y}_i)^2 \rightarrow \text{MSE}$$

$$\frac{\partial L}{\partial w_7} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_7} \Rightarrow \sum_{i=1}^n \frac{1}{n} \times 2 \times \frac{(y_i - \hat{y}_i)}{\partial \hat{y}} \times 1 \times N_1$$

$$\frac{\partial L}{\partial w_9}, \frac{\partial L}{\partial w_1}$$

6 mins

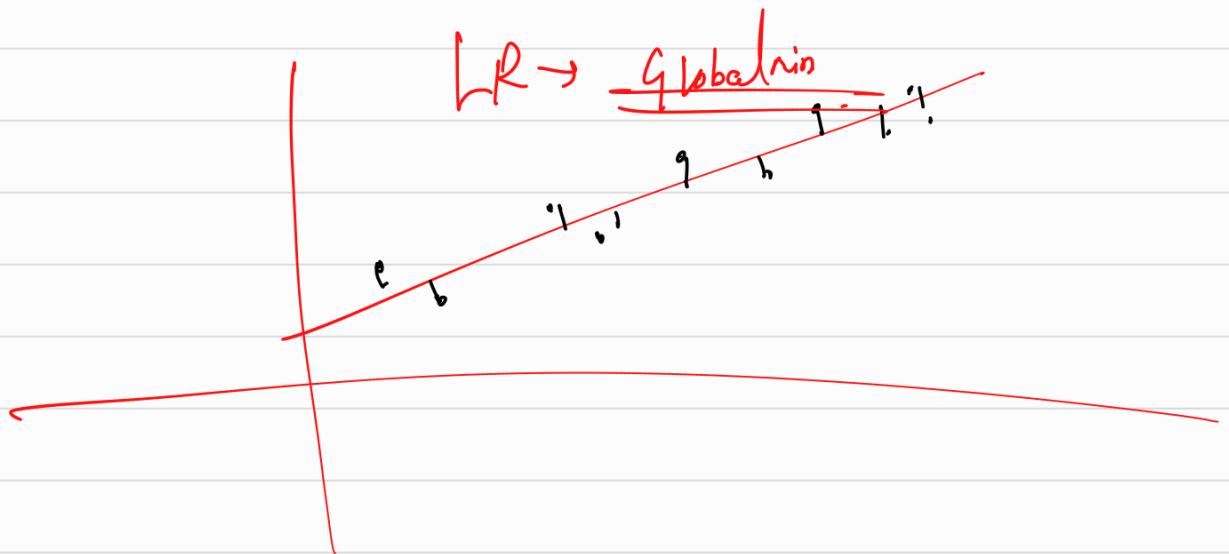
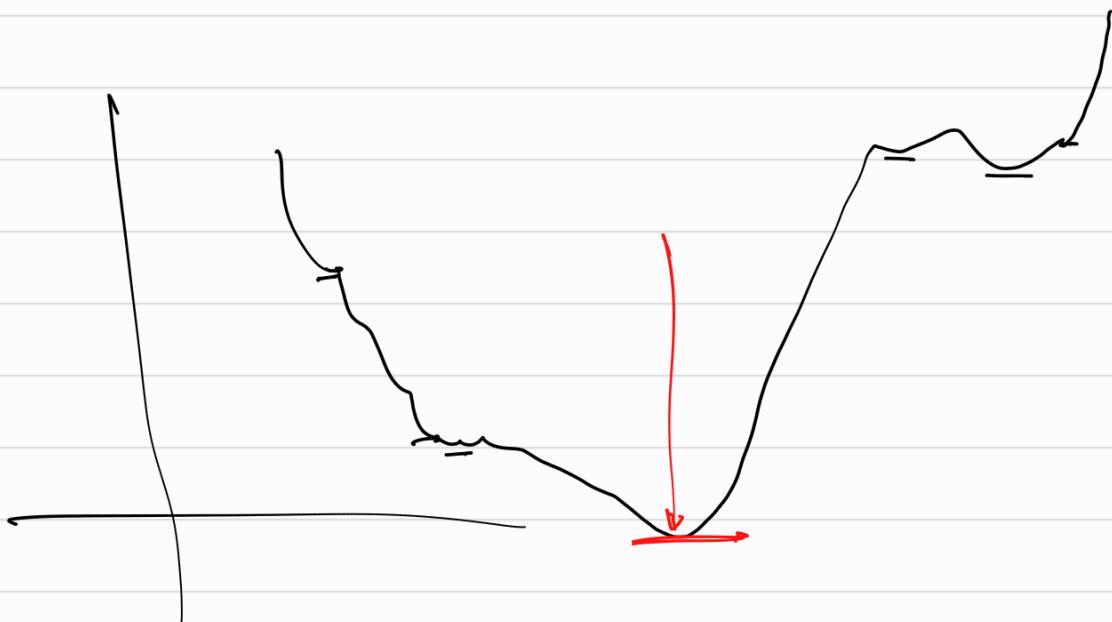
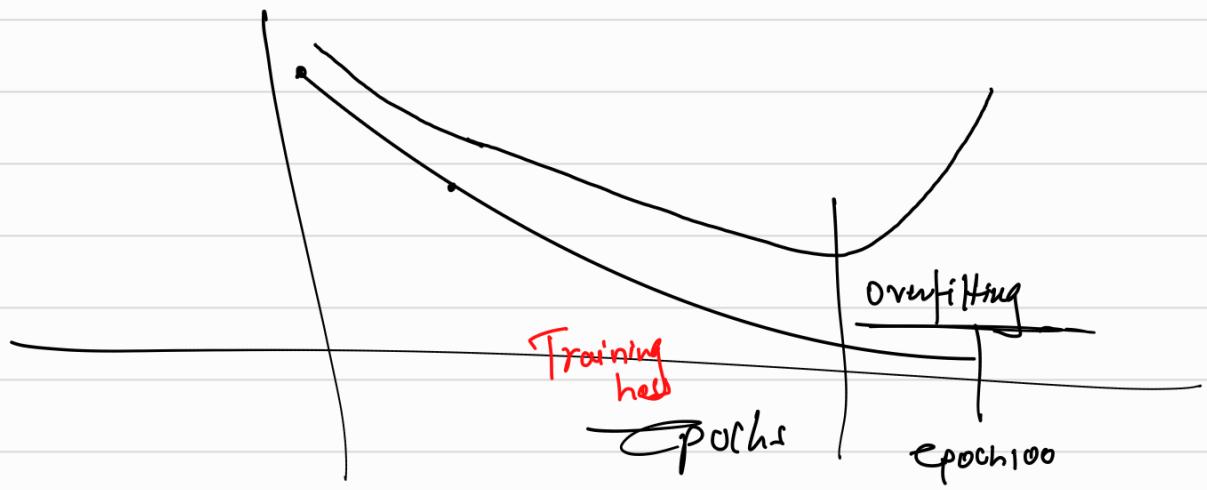
$$\frac{\partial L}{\partial w_9} = \left(\frac{\partial L}{\partial \hat{y}} \right) \times \frac{\partial \hat{y}}{\partial w_9} \rightarrow N_3$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial N_1} \times \frac{\partial N_1}{\partial w_1}$$

$$\hat{y} = w_7 N_1 + w_8 N_2 + w_9 N_3 + b_4$$

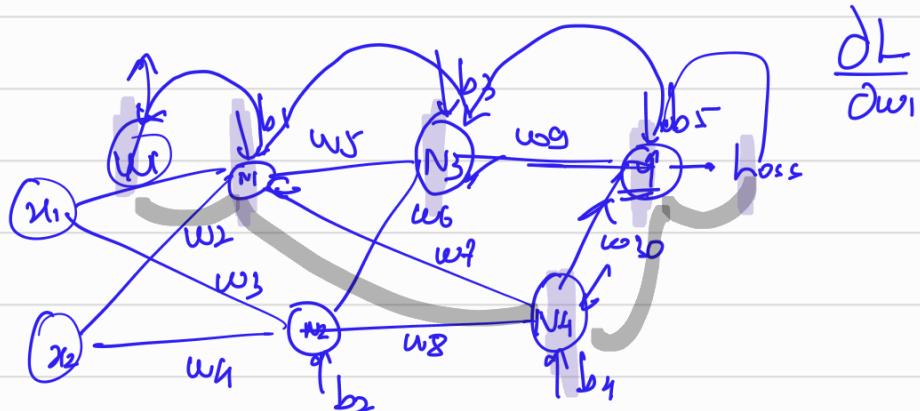
$$\frac{\partial \hat{y}}{\partial w_7} = \frac{\partial (\textcircled{w_7} \textcircled{N_1})}{\partial w_7} = N_1$$

$$\frac{\partial (x_1 x_2)}{\partial x} = 2$$



Activation functions

$$w_1x_1 + w_2x_2 + b_1$$



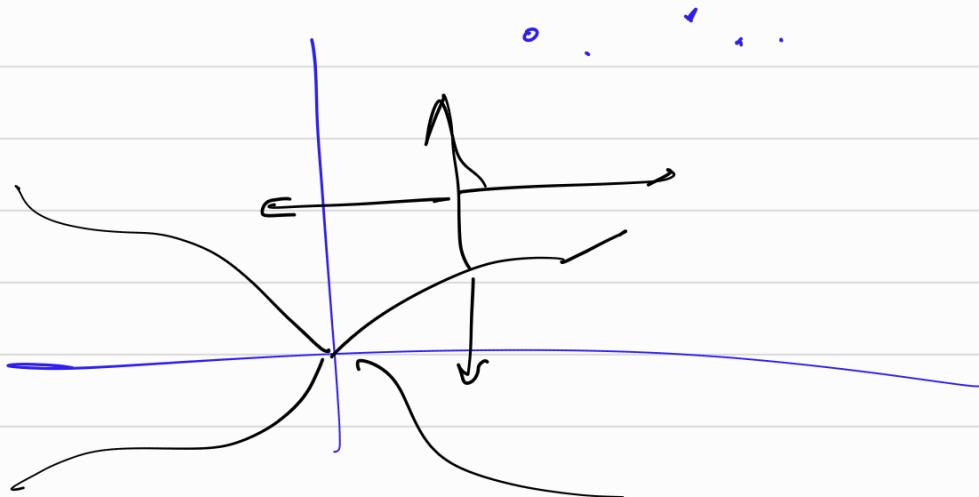
$$\frac{\partial L}{\partial w_1}$$

$\frac{\partial h}{\partial w_1} = \left(\frac{\partial L}{\partial y} \times \frac{\partial y}{\partial N_3} \times \frac{\partial N_3}{\partial w_1} \right) + \left(\frac{\partial L}{\partial y} \times \frac{\partial y}{\partial N_4} \times \frac{\partial N_4}{\partial w_1} \right)$

$\frac{\partial L}{\partial w_1} = 1000000$
~~Terms = $N(N_L)^2$~~

$$\frac{\partial h}{\partial w_1} = 0.00000^2$$

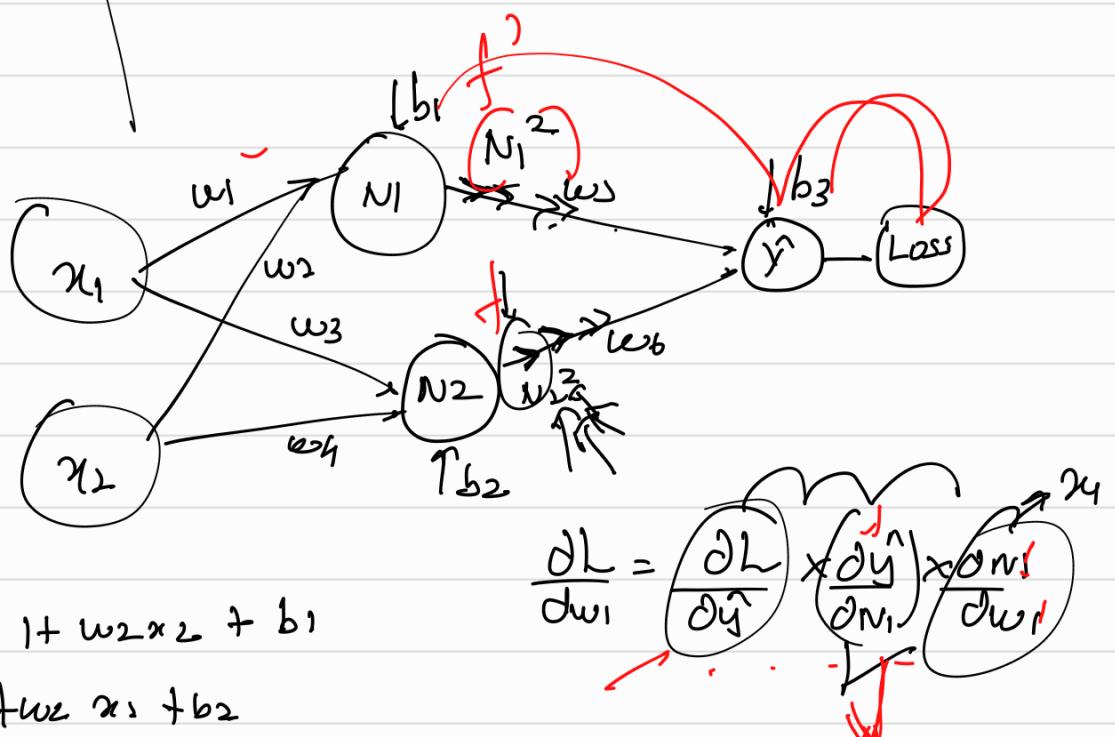
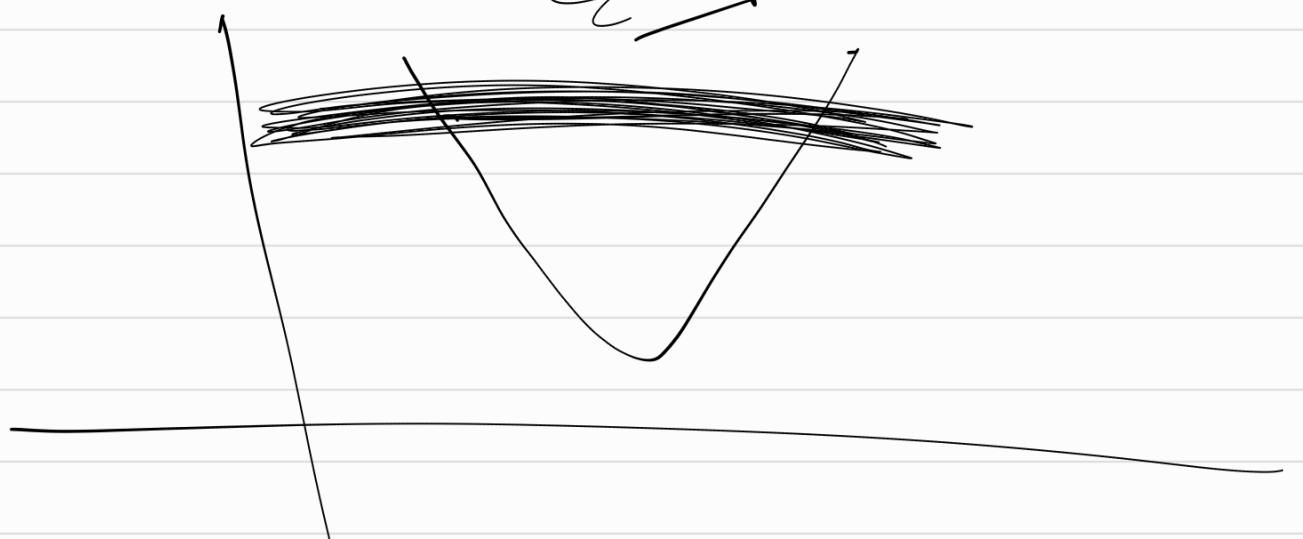
$w_1 -$



$$\left(\frac{0.99}{1.01}\right)^{365} \approx \underline{\downarrow} \underline{\downarrow} \downarrow$$

$$\left(\frac{1.01}{1.01}\right)^{365} = 1.99\underline{1.37}$$

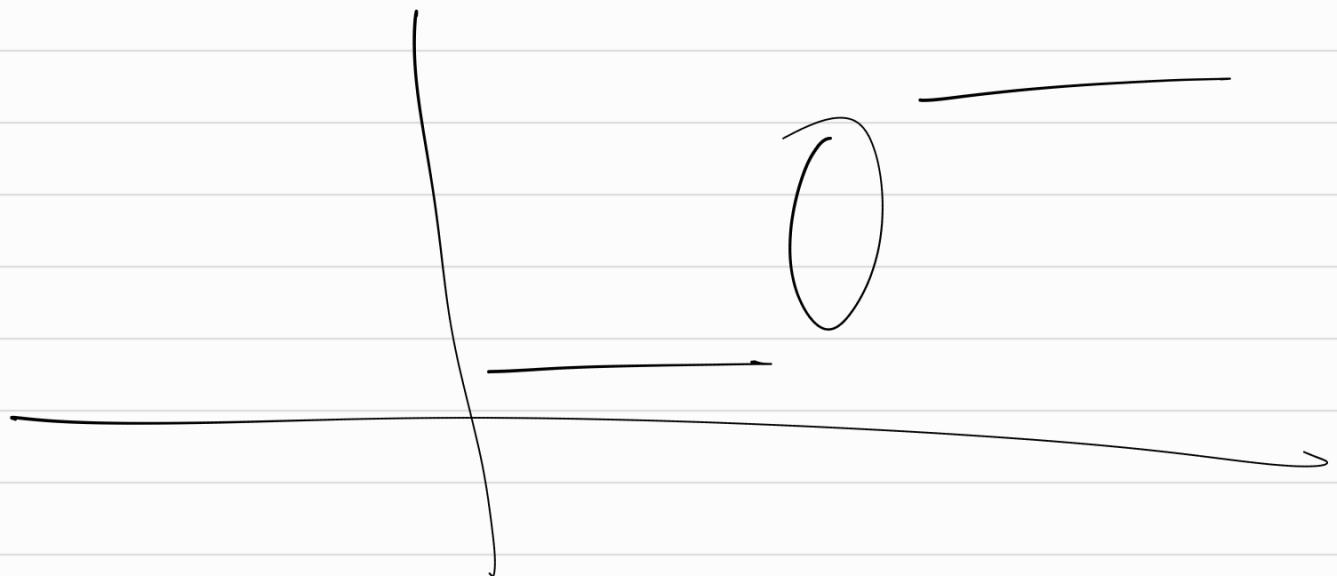
Exploding gradient



$$\hat{y} = w_5 \underline{N_1^3} + w_6 \underline{N_2^2} + b_3$$

$$\frac{\partial \hat{y}}{\partial w_1} = w_5 \times \underline{2 \times N_1}$$

~~(Activation function) \rightarrow Should be cont & diff~~



Activation functions are functions which should be continuous & diff that can be applied at the output of any hidden layers to create non-linear feature or at the 0th to create bounded outputs \rightarrow Probs

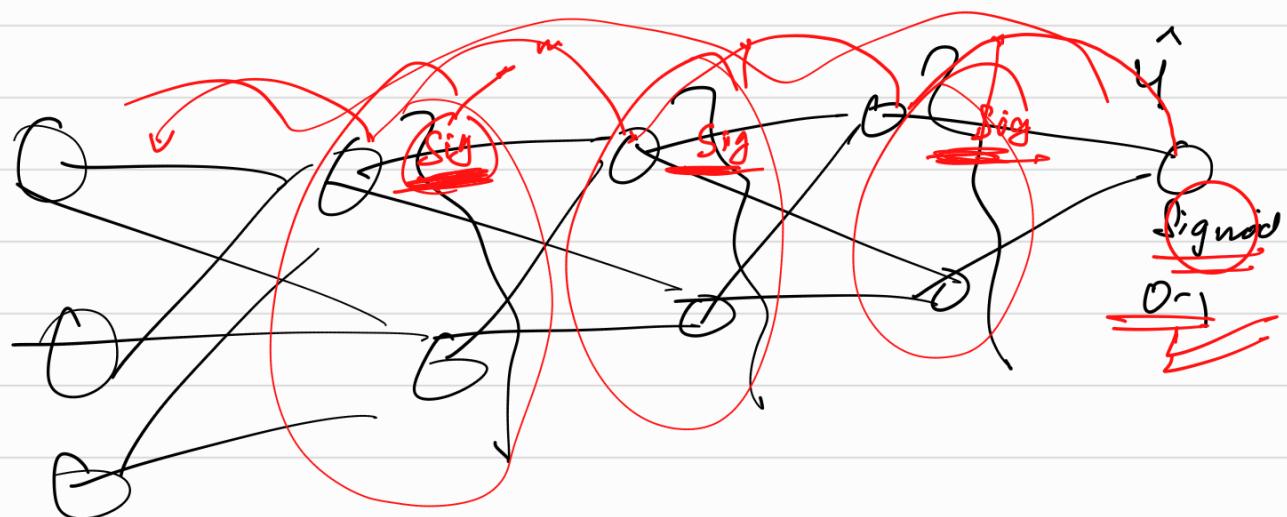
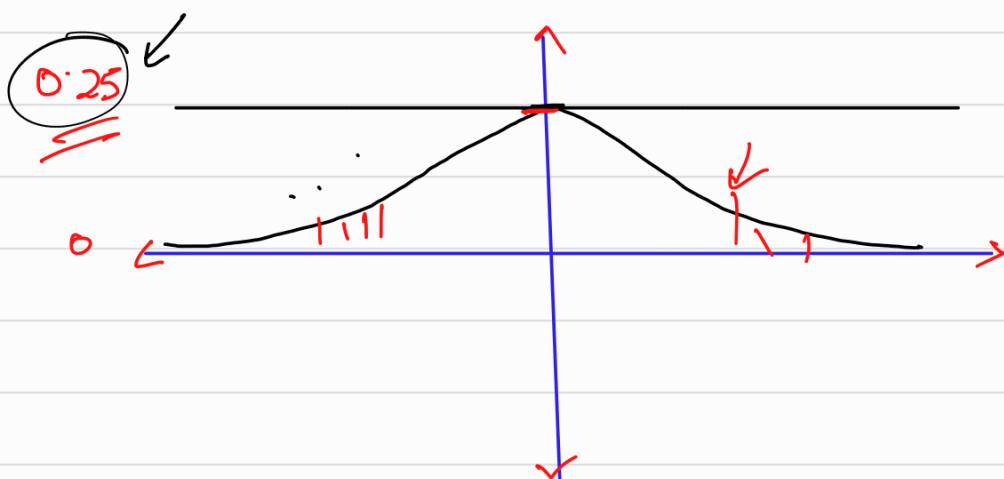
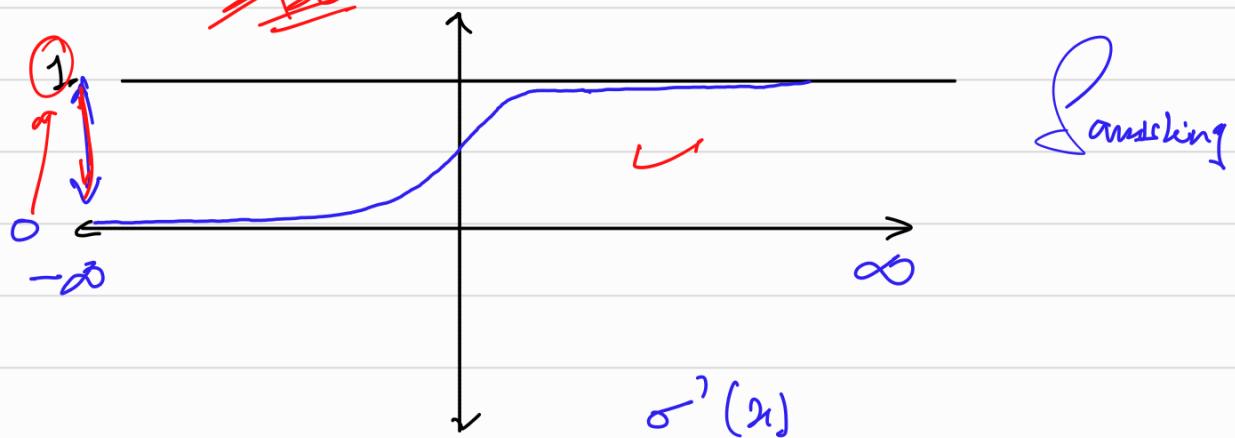
↓ output layer

→ Sigmoid

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

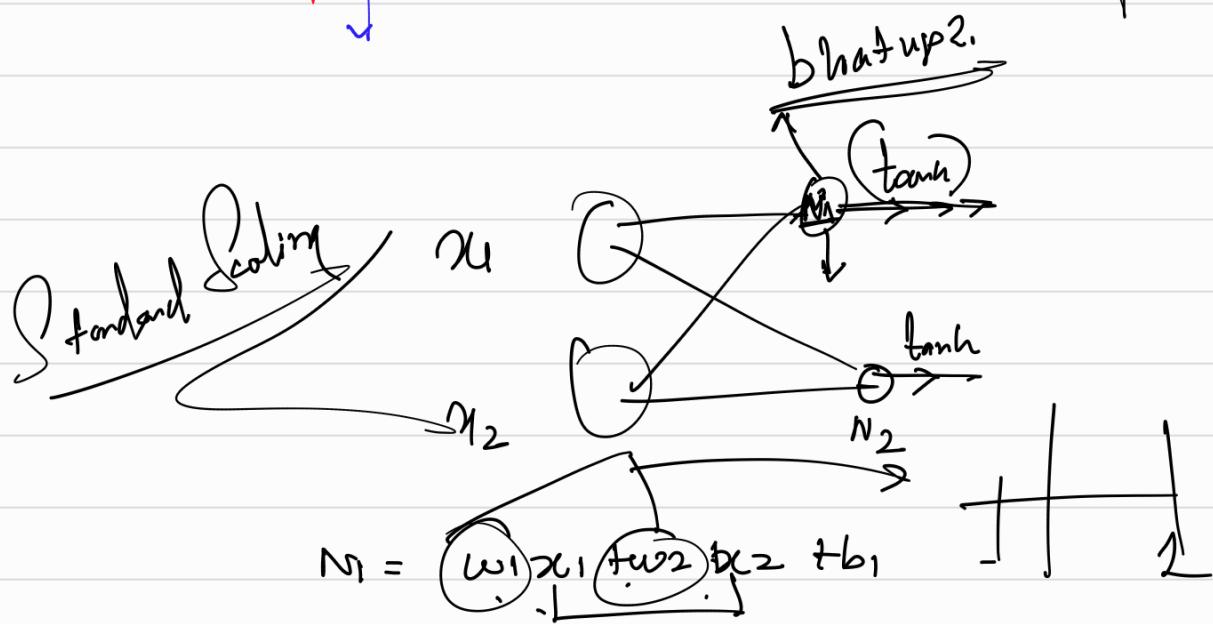
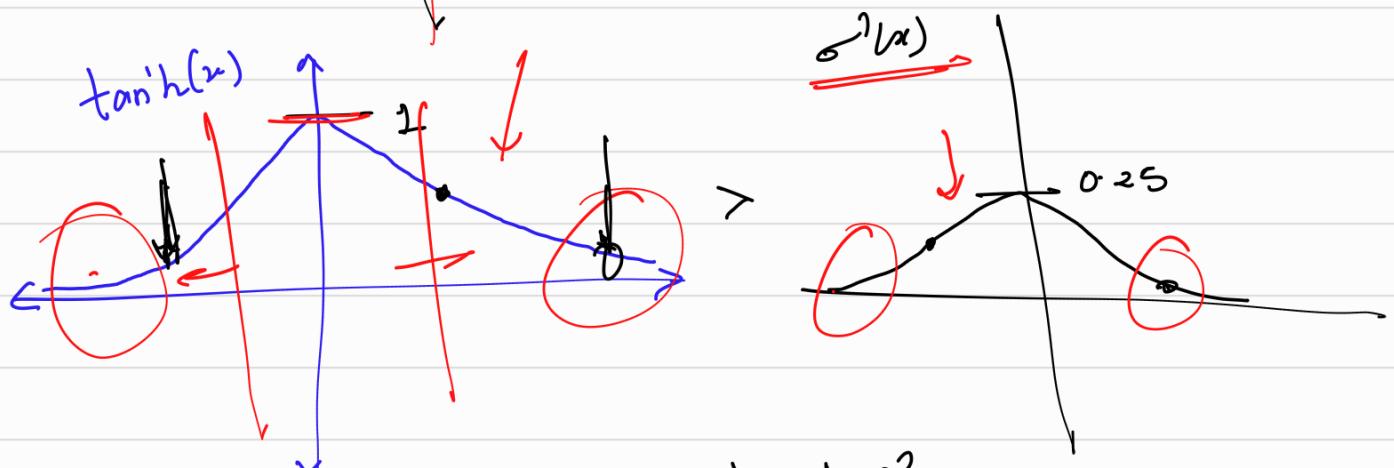
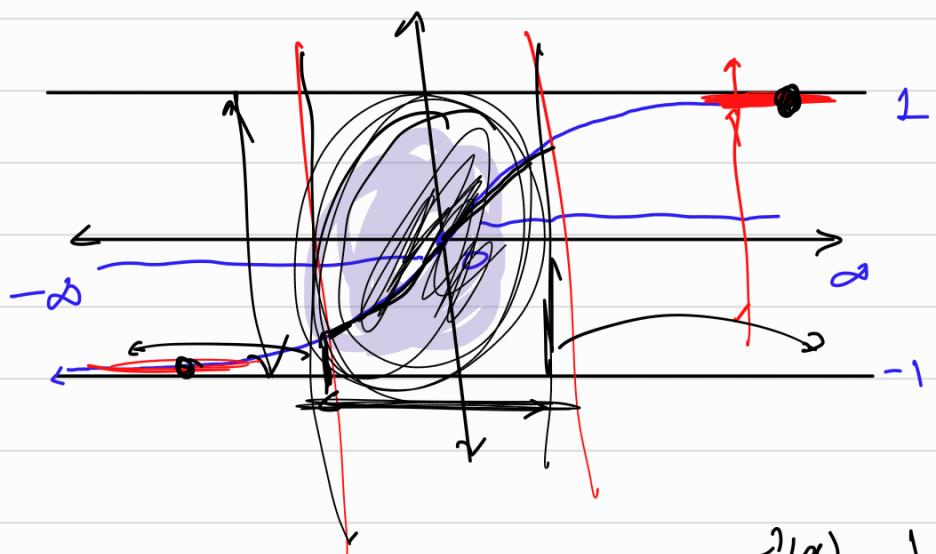
$-\infty < x < \infty$

H/LX \rightarrow $0 < \sigma(x) < 1$

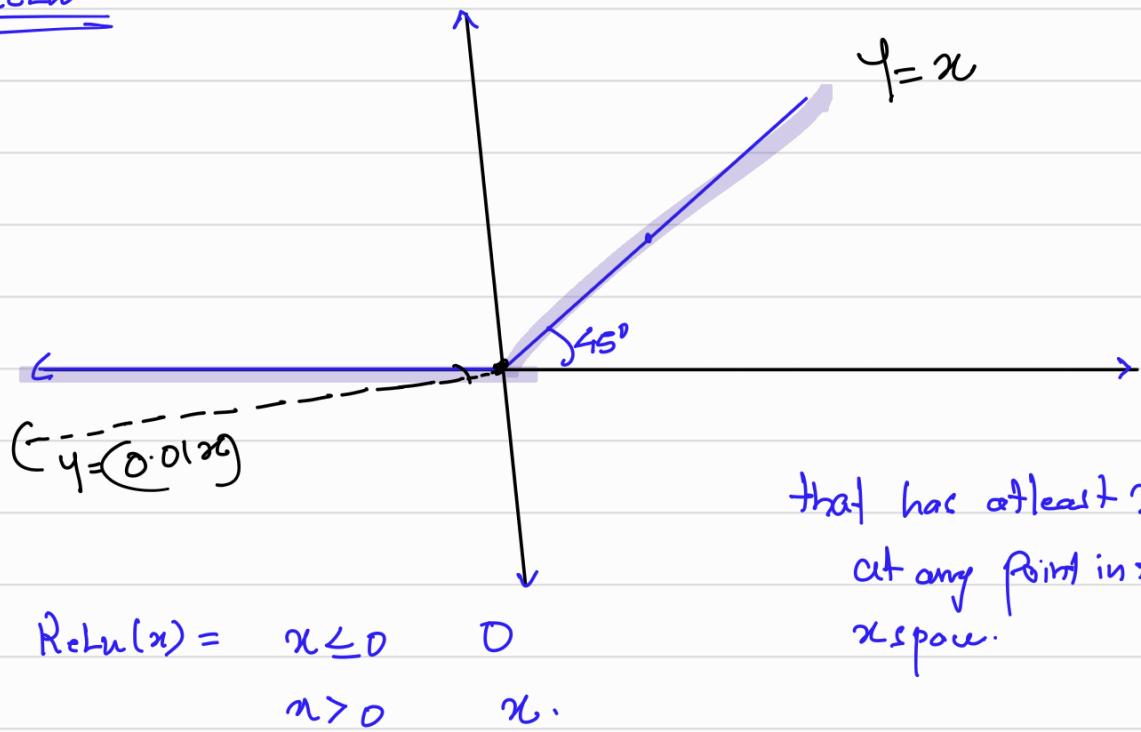


$\tanh \rightarrow$ hyperbolic tangent.

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



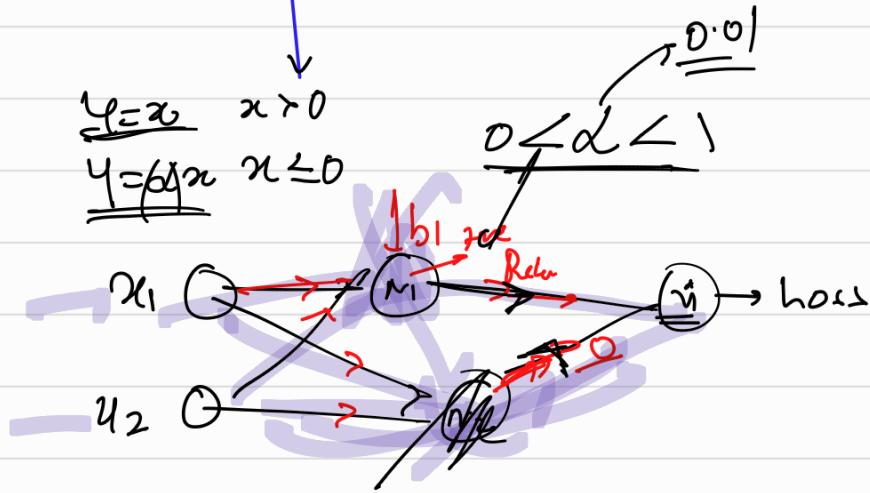
ReLU



that has at least 2 slopes
at any point in the
 x -space.

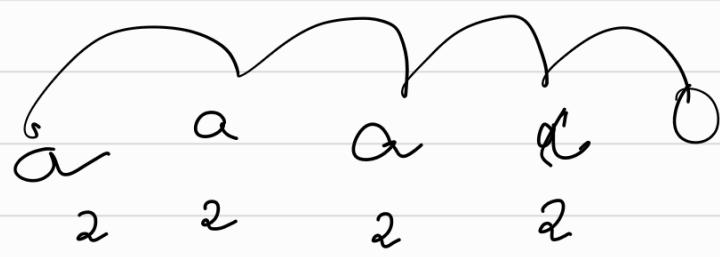
↳ $\text{ReLU}(x) \Rightarrow x \leq 0 \oplus \underline{0.01x}$
 $x > 0 \oplus 1x$

Parameteric Rule



Deep Neural network

$$(2)^{\overbrace{10}^{+10}} = 32$$

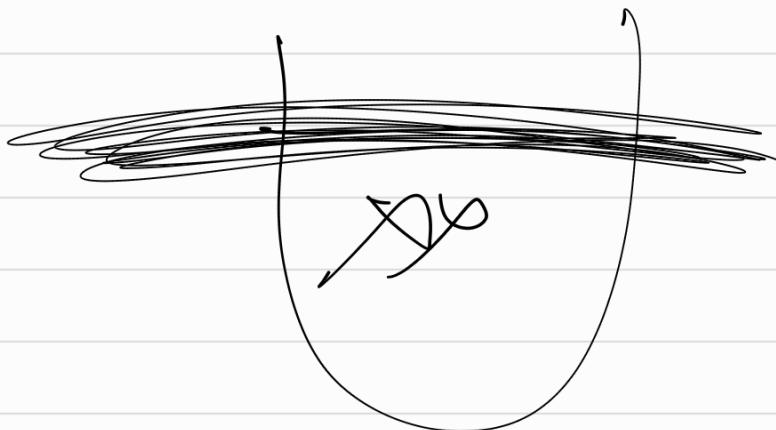


 a a a a

 2 2 2 2

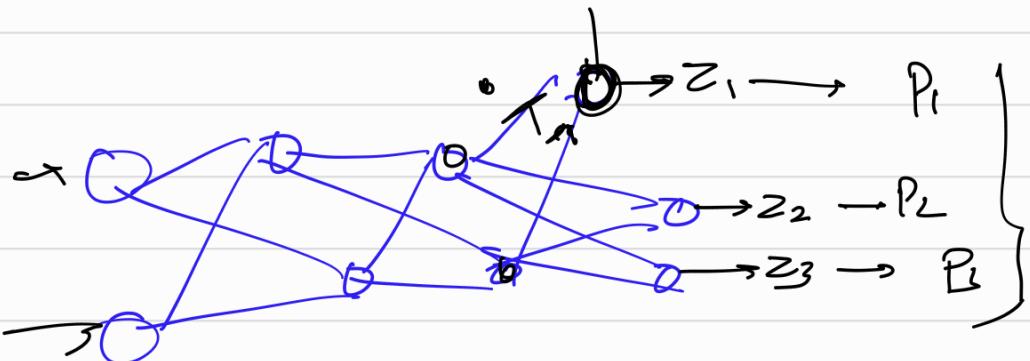
$$(2)^3 \quad \underline{(2)^{100}}$$

$$\frac{dh}{dx_1} = w_r - \alpha \left(\frac{1}{g_m} \right)$$



Softmax

: needs to be applied at the o/u of a multiclass Problem

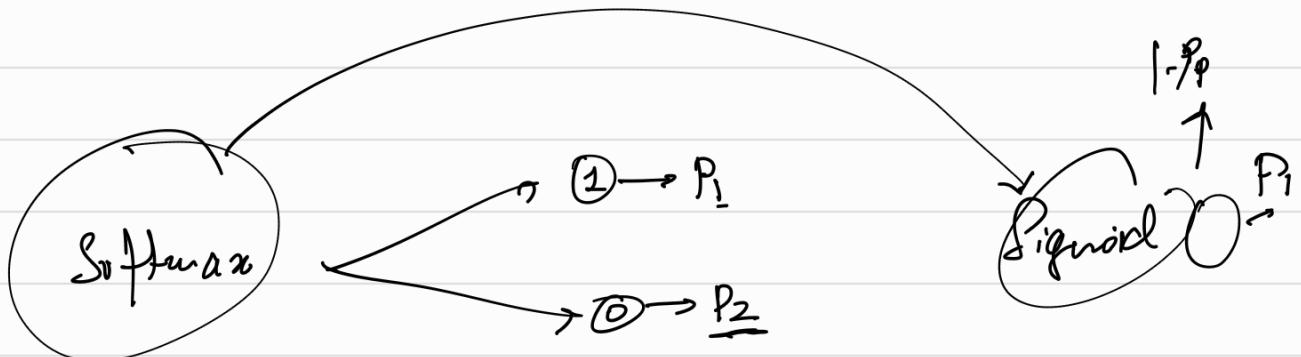


$$p_1 + p_2 + p_3 = 1$$

$$p_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$p_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$p_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$



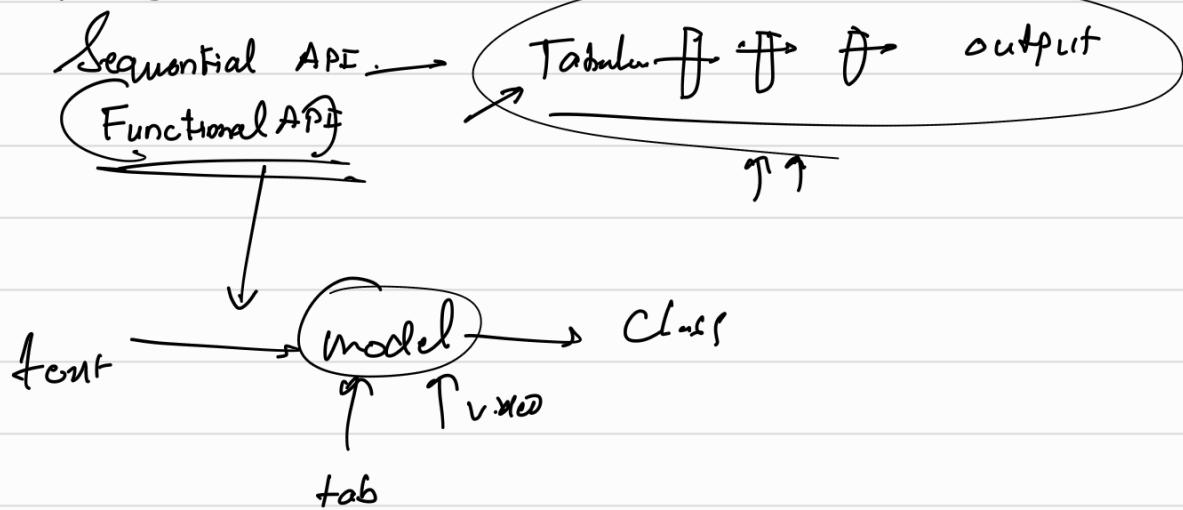
$$p_1 + p_2 = 1$$

$$p_1 = 1 - p_2$$

$$p_2 = 1 - p_1$$

Bocah

Lecture - 5:



Specce.

$$y \rightarrow LE(0, 1, 2)$$

$$(y=1)$$

$$\hat{y} = [0.1 \quad 0.3 \quad 0.6]$$

$$- \log(\hat{y}[y])$$

$$- \log 0.3$$

$$y \rightarrow DCE.$$

$$y \rightarrow [0, 1, 0].$$

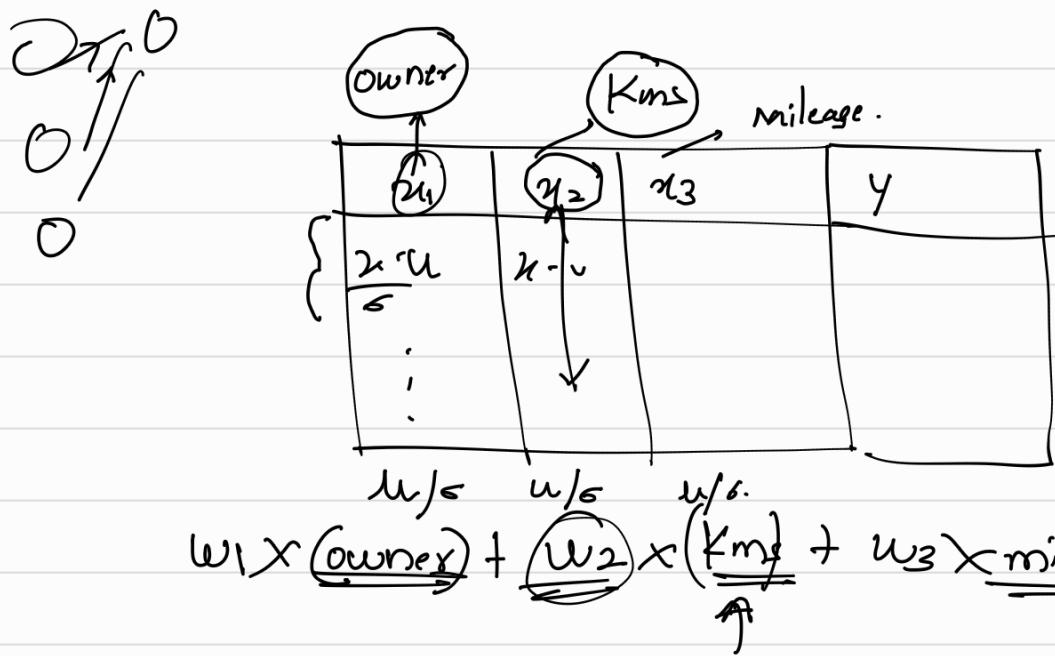
cce.

$$\hat{y} = [0.1 \quad 0.3 \quad 0.6]$$

$$y = [0 \quad 1 \quad 0]$$

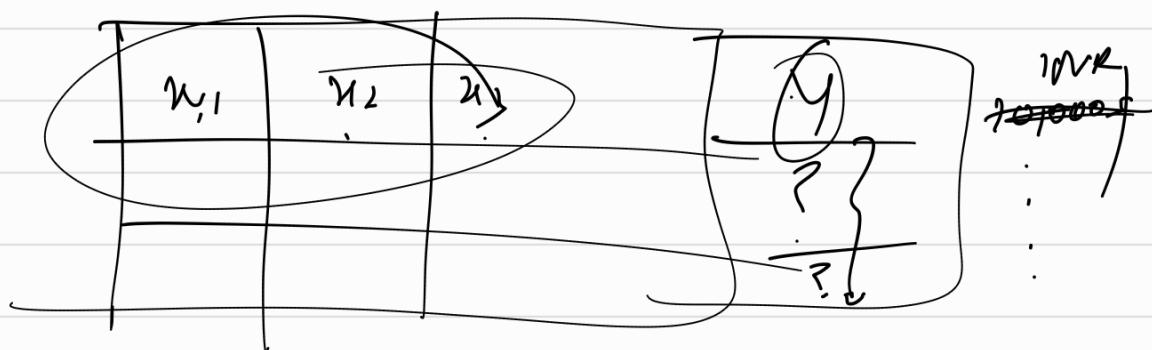
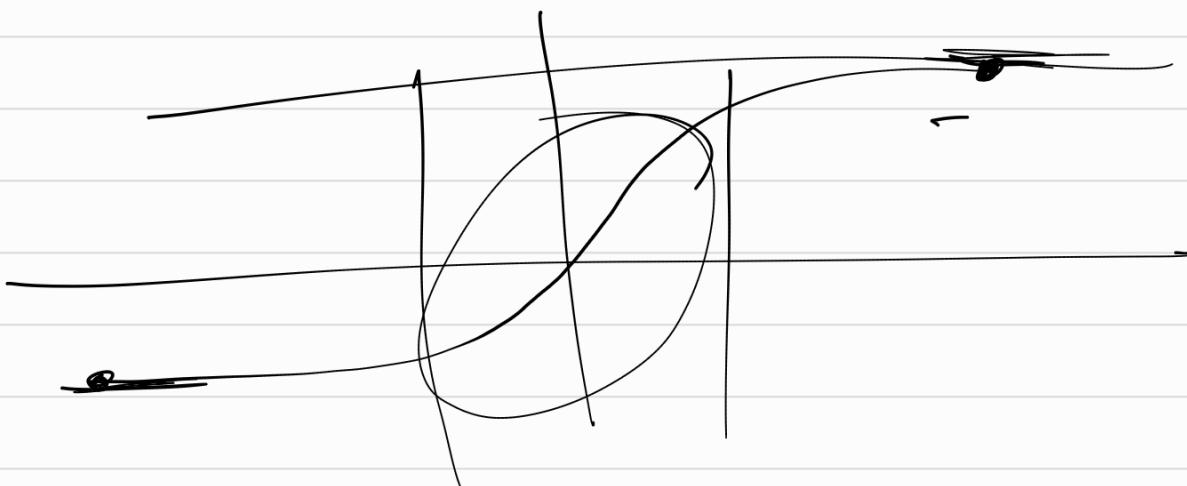
$$- [0 \times \log 0.1 + 1 \times \log 0.3 + 0 \times \log 0.6]$$

$$= - \log 0.3$$



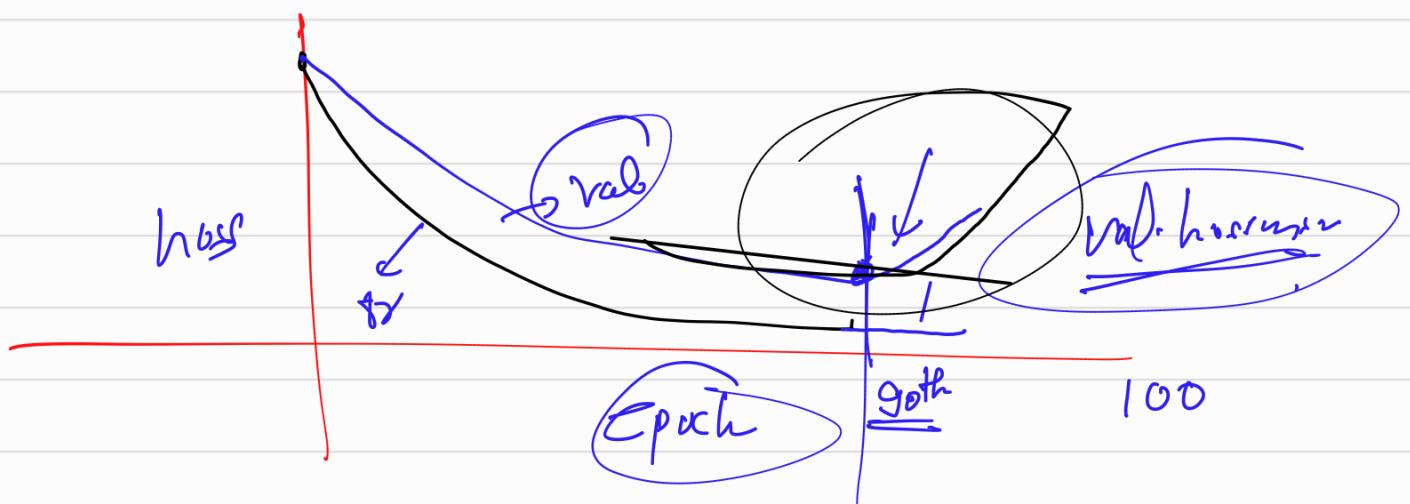
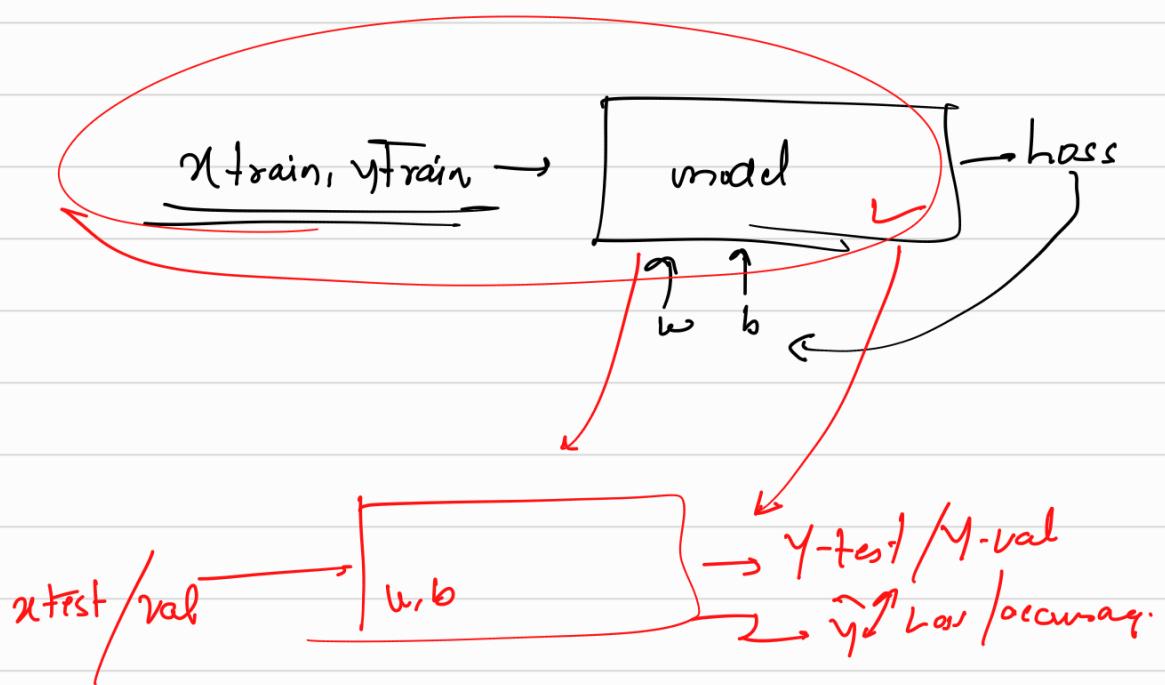
(1) Honda City
50,000 ↘

(2) ~~300~~ Honda City
40,000

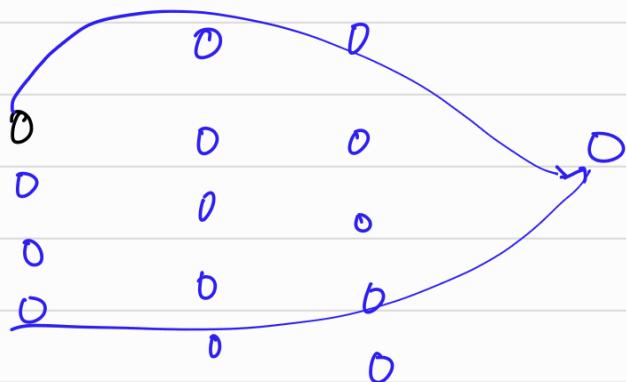


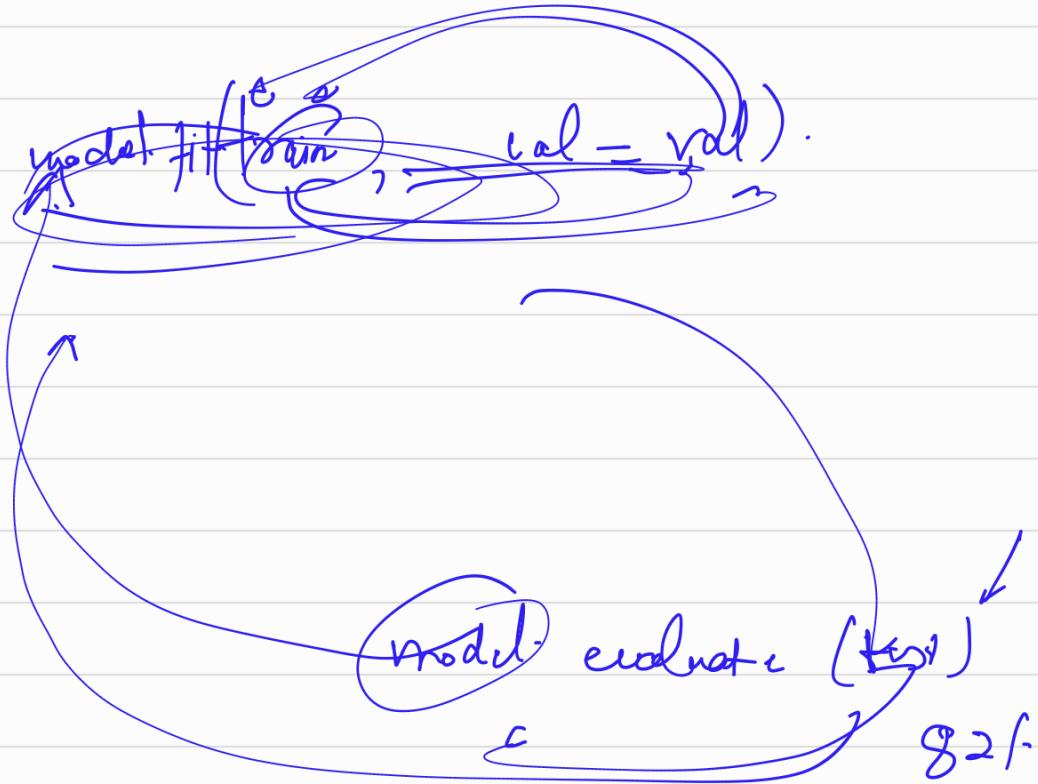
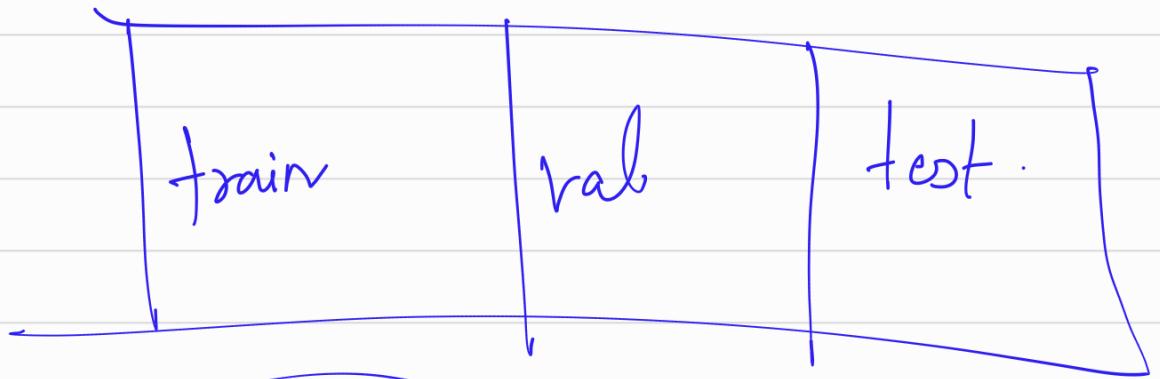
3,-3
($u_1 \sim -1, 1$)

3,-3 → 2-1 → 0 → 500,000
 $L = (\underline{500,000} - 0 \cdot 9)^2$



Callbacks

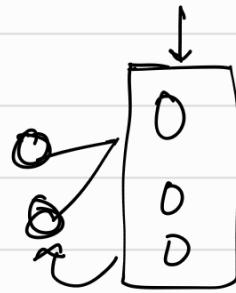




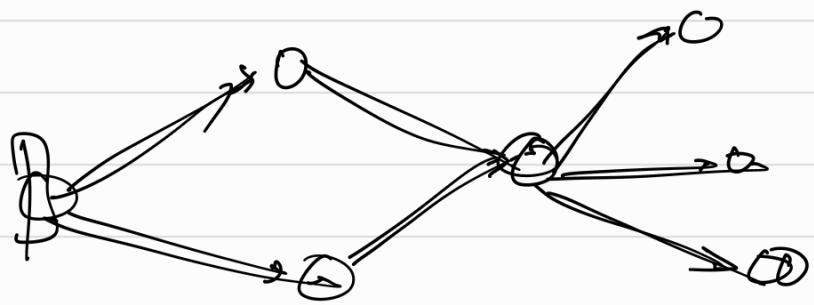
82f.

Functional API:

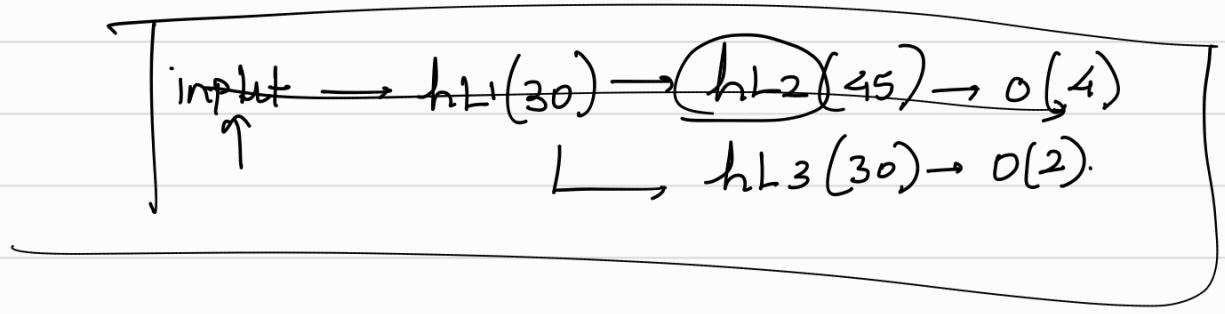
Graph Structure



any layer (ϵ)



$D \rightarrow D'$



Input $\xrightarrow{(4)} hL_1(30) \rightarrow hL_2(2)(40)$

Case 1: Input $\xrightarrow{(4)} hL_1(20) \rightarrow hL_2(2) \rightarrow 50$

Case 2:

inp1 = Input(4,)

inp2 = Input(4,)

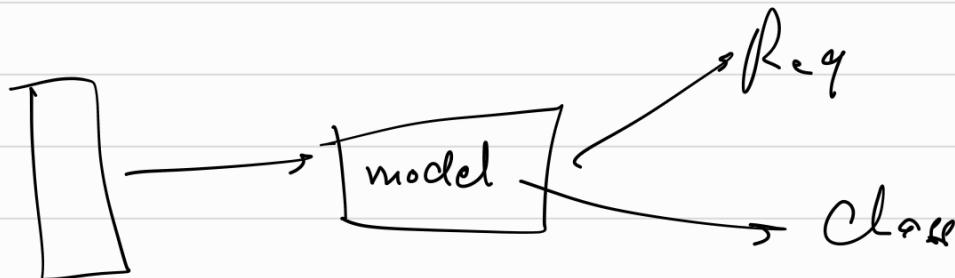
hL1 = Dense(30)(inp1)

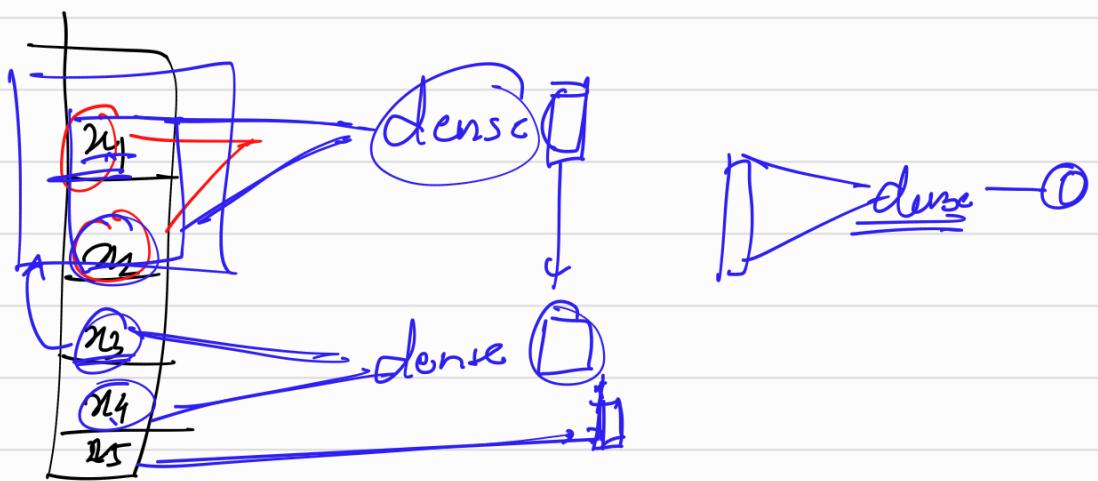
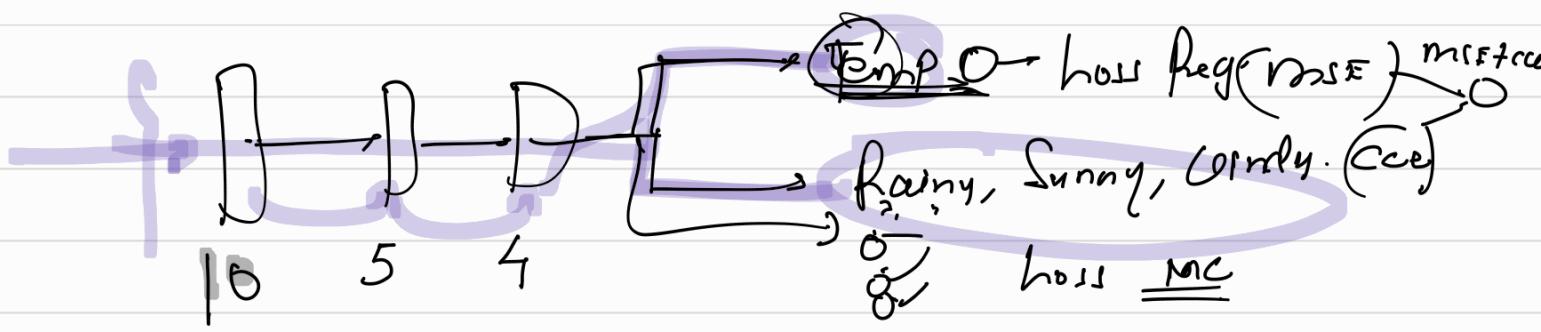
hL2 = Dense(40)(hL1)

hL3 = Dense(20)(inp2)

hL4 = Dense(50)(hL3)

O = Concatenate([hL2, hL4])

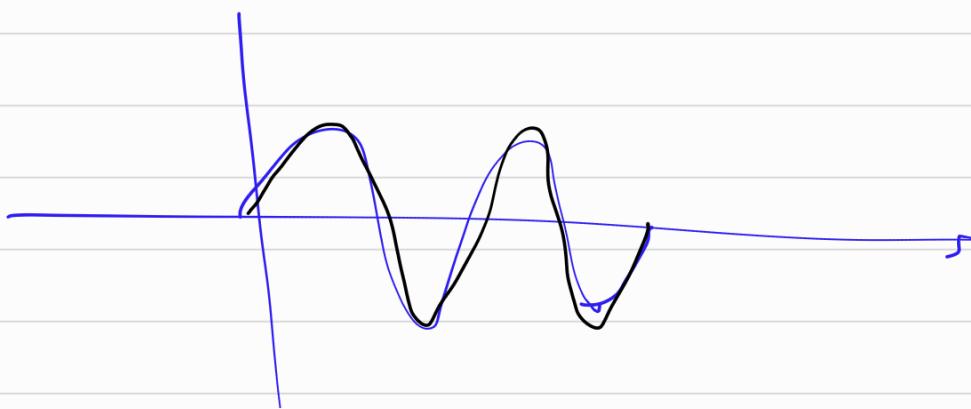
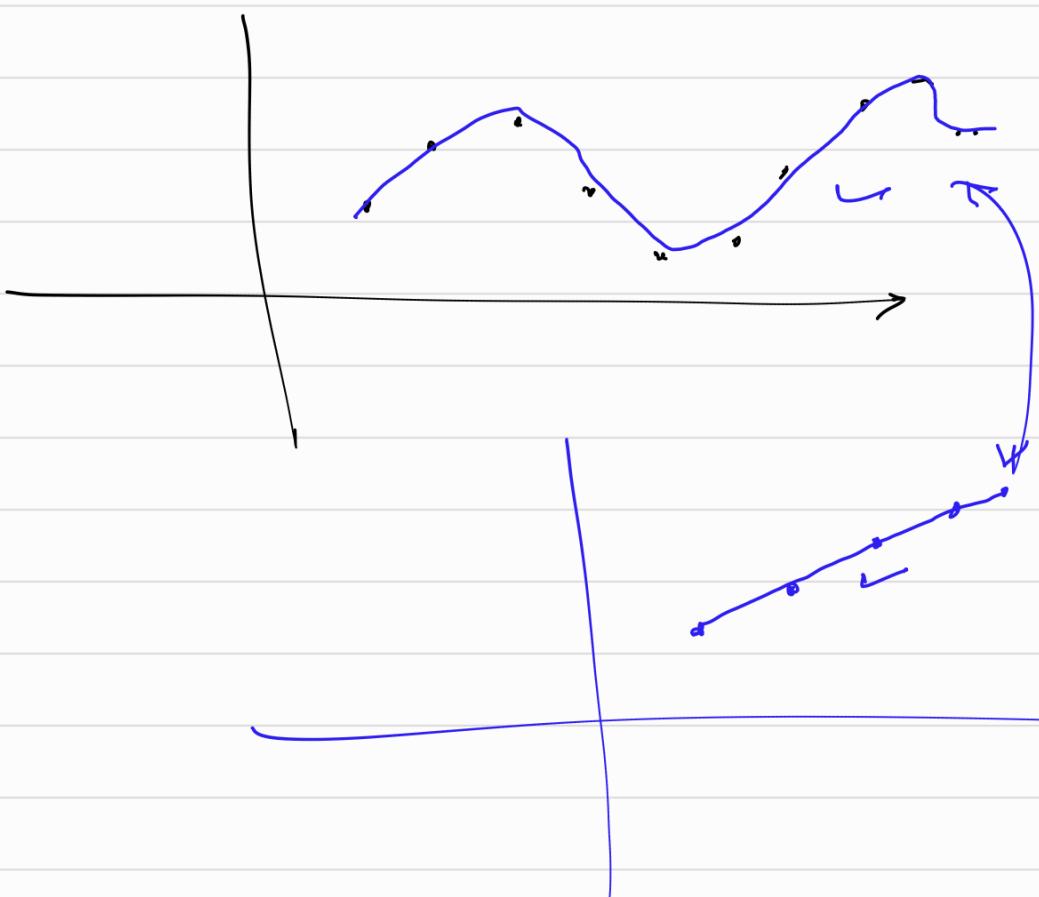
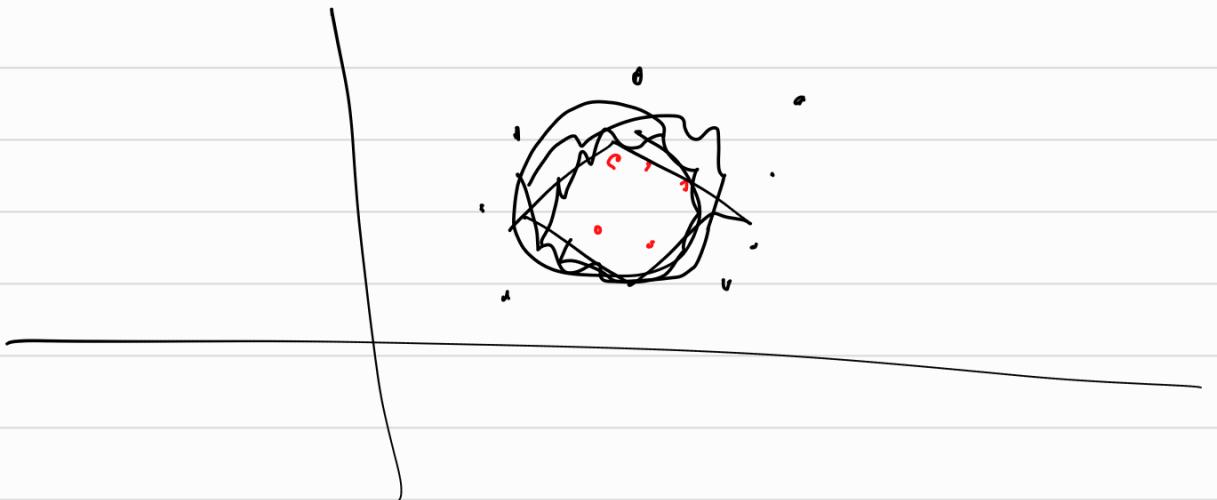




Optimizers

loss





gg. gggg).



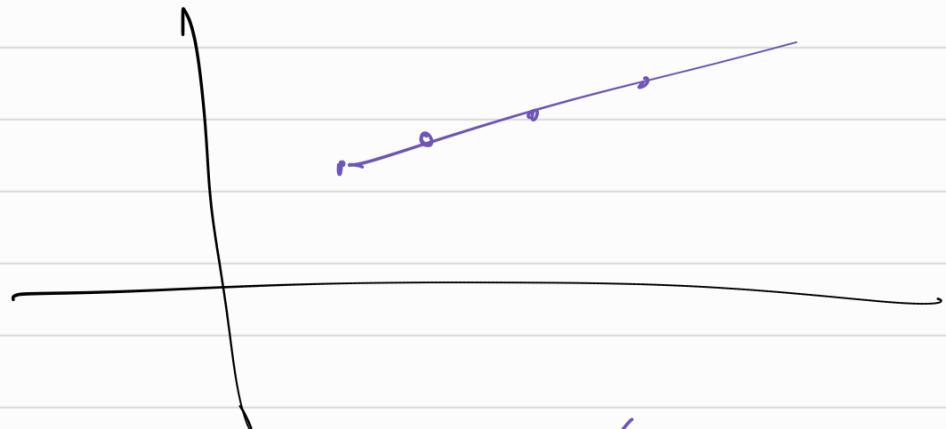
(Regularization.)

- Regularization
- Dropout
- Optimizers.
- Callbacks
- Batch Normalization
- weight initialization

Hyperparameter Tuning

Regularization:

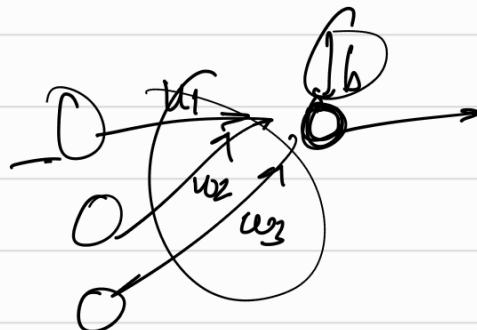
↓ ↓
(Ridge and Lasso.) ?



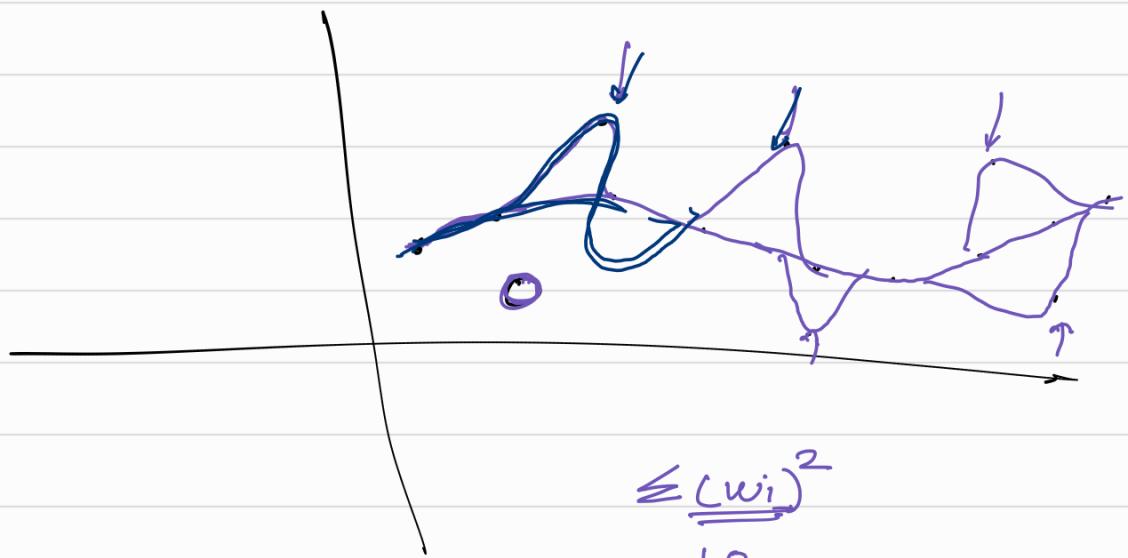
Overfit → has high variance (model is extremely sensitive to changes in n dict).

$$\textcircled{1} \quad y = \underline{100x_1} + \underline{500x_2} \quad y=1000$$

$$\textcircled{2} \quad y = 2x_1 + x_2 = \textcircled{2}$$



$$N1 = \underline{w_1 x_1} + \underline{w_2 x_2} + \underline{w_3 x_3} + b$$



$$\begin{aligned}
 TL & \quad mSE \\
 510 & = 500 \\
 425 & = 400 \\
 375 & = 320 \\
 355 & = 260 \\
 320 & = 200 \\
 305 & = 150 \\
 \hline
 345 & = 145
 \end{aligned}$$

$$\leq \frac{(w_i)^2}{10}$$

25

55

95

= 120

$$\begin{aligned}
 & = \frac{155}{200} \\
 & = \{ \}
 \end{aligned}$$

$$TL = \text{mse} + \boxed{\frac{1}{g}} \sum_{j=1}^p w_j^2$$

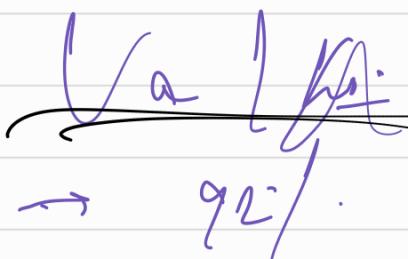
Lagrangean multiplier

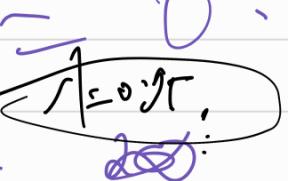
Hyperparameter 

~~Val-loss~~



 $\rightarrow 0.5$

 $\rightarrow 92\%$

$\lambda =$  $\rightarrow 0.9$

 95%

Reg.

$$T_L = \text{mse} + \lambda \sum w_j^2 \quad \text{Ridge } (L^2).$$

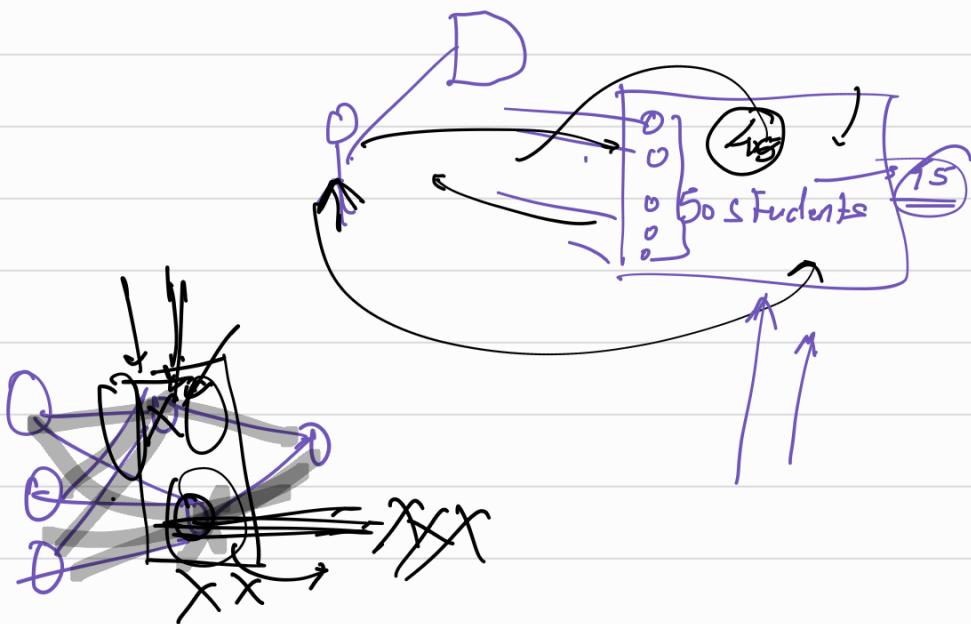
$$T_h = \text{mse} + \lambda \sum |w_j| \quad \text{Lasso } (L_1).$$

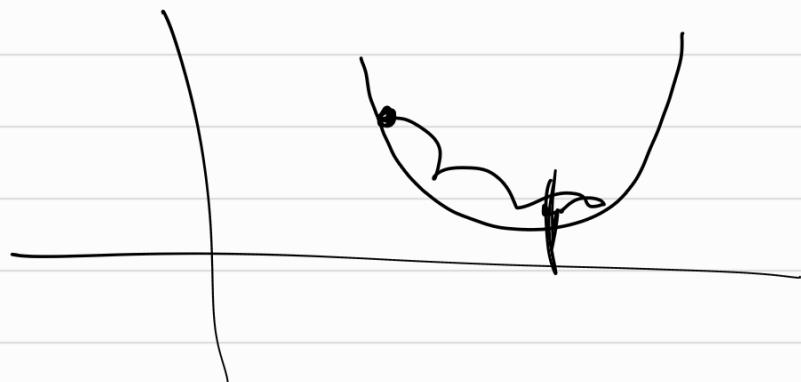
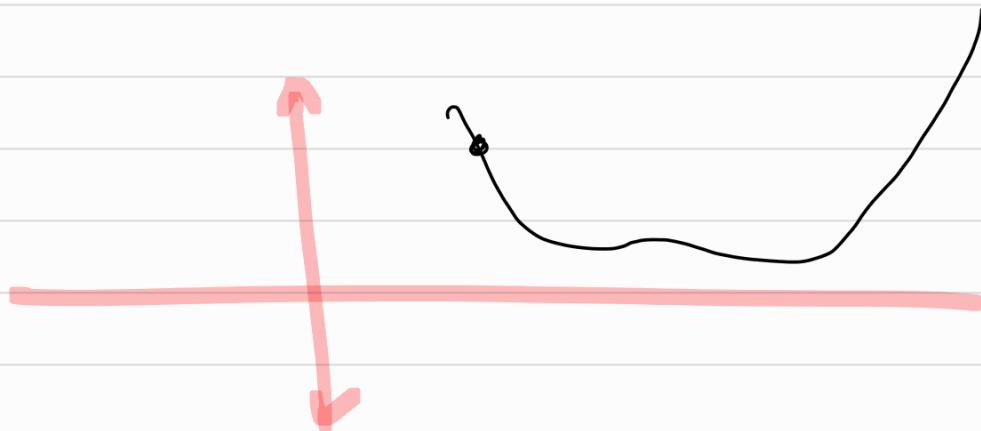
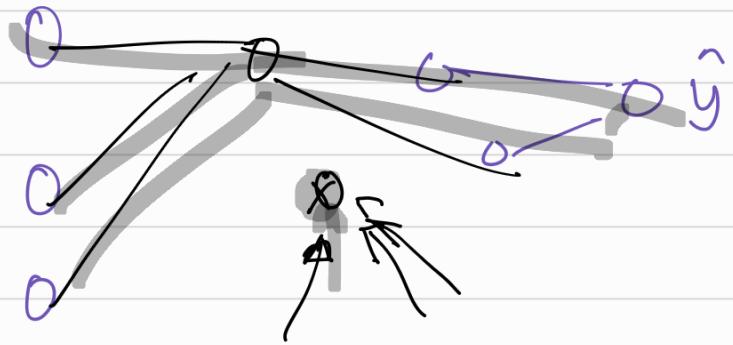
$$T_L = \text{mse} + \lambda_1 (\sum w_j^2) + \lambda_2 \sum |w_j| \quad L_1 L_2$$

Elastic Net

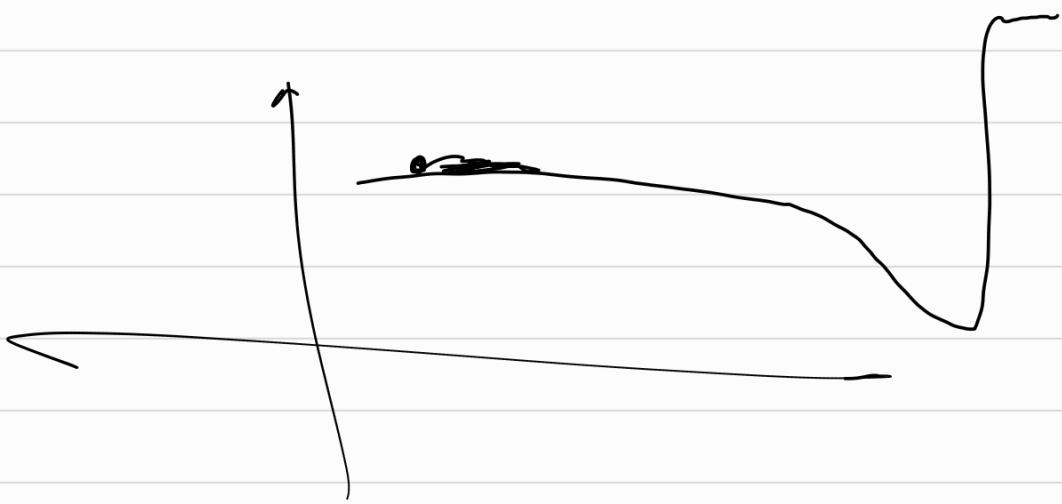
Break!

Dropout:



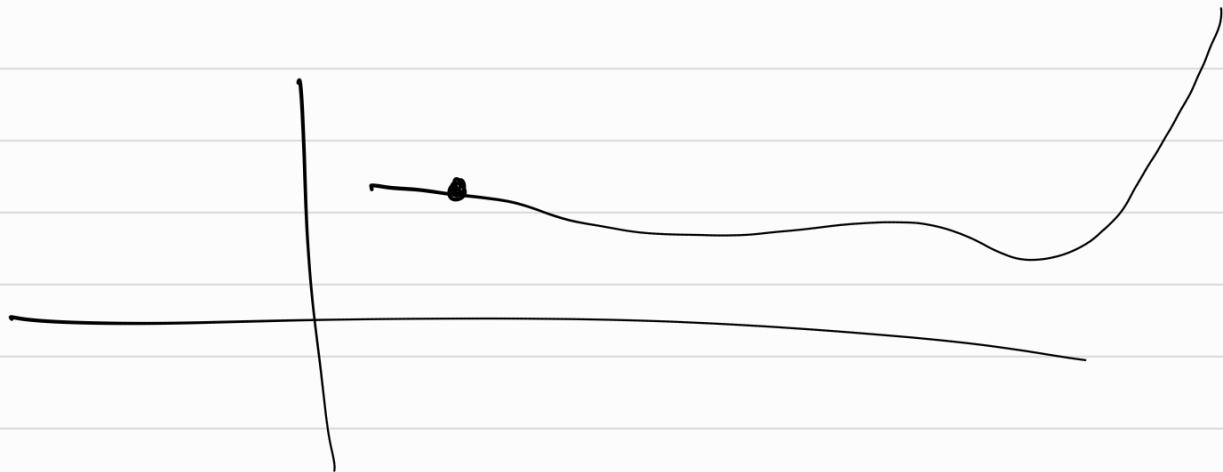
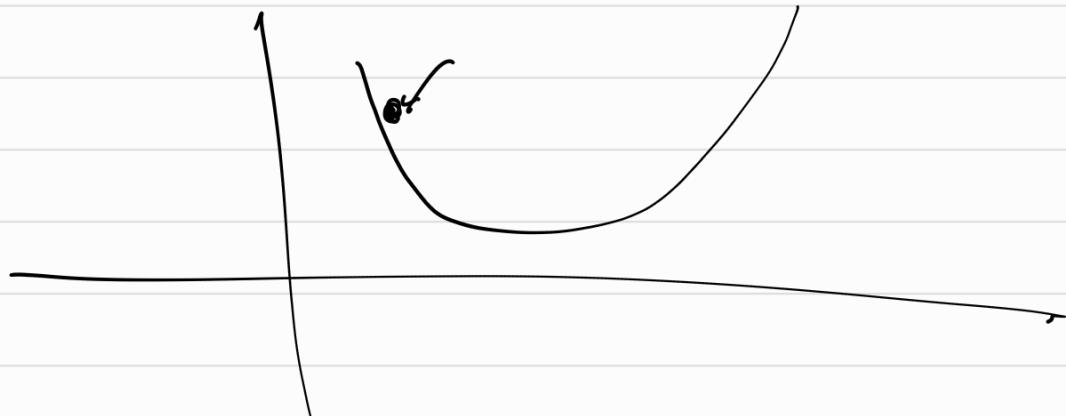
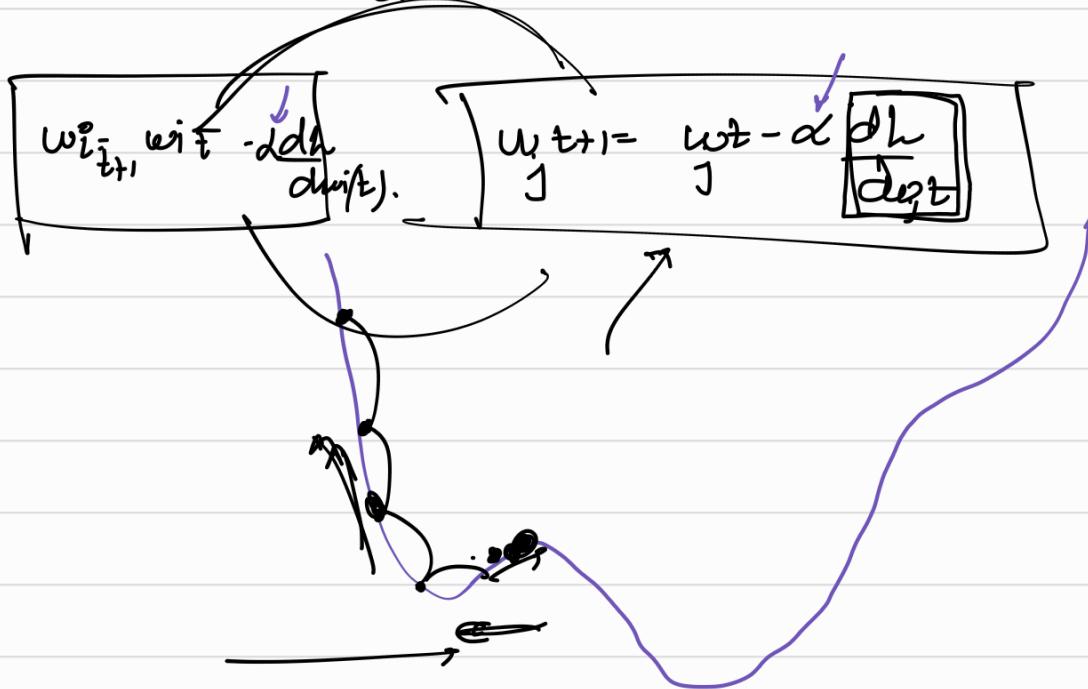


10

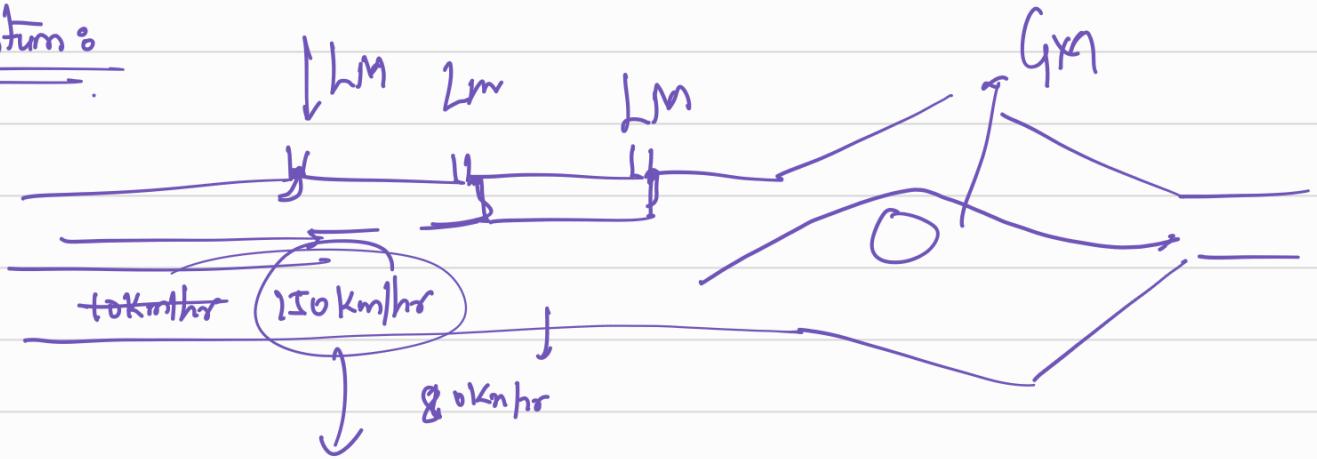


optimizers :

minimizing the loss by changing the weights and biases.



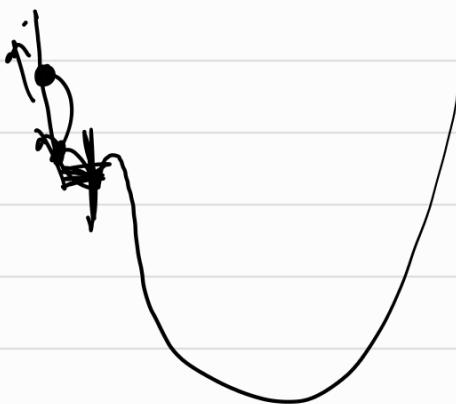
Momentum:



Momentum

GD

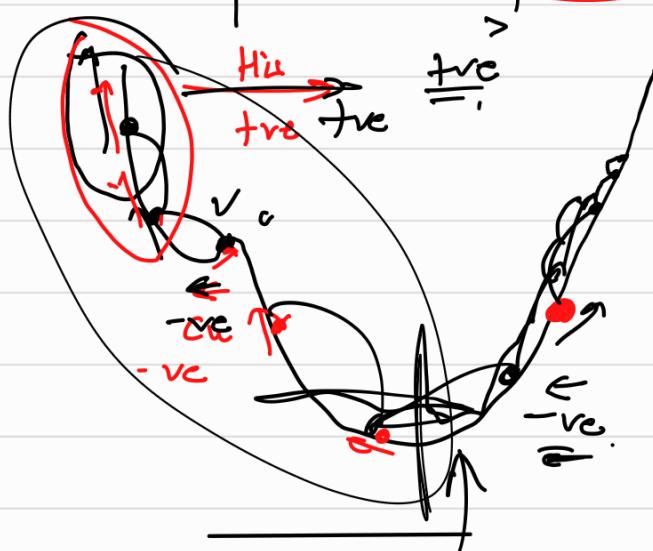
$$w_{t+1} = w_t - \alpha \frac{\partial h}{\partial w_t}$$



GD with momentum

$$w_{t+1} = w_t - \text{update}_t$$

$$\text{update}_t = \beta \text{update}_{t-1} + (1-\beta) \frac{\partial h}{\partial w_t}$$



$t=1$ $t=2$ $t=3$

$$\text{update}_1 = (1-\beta) \alpha \frac{\partial h}{\partial w_1}$$

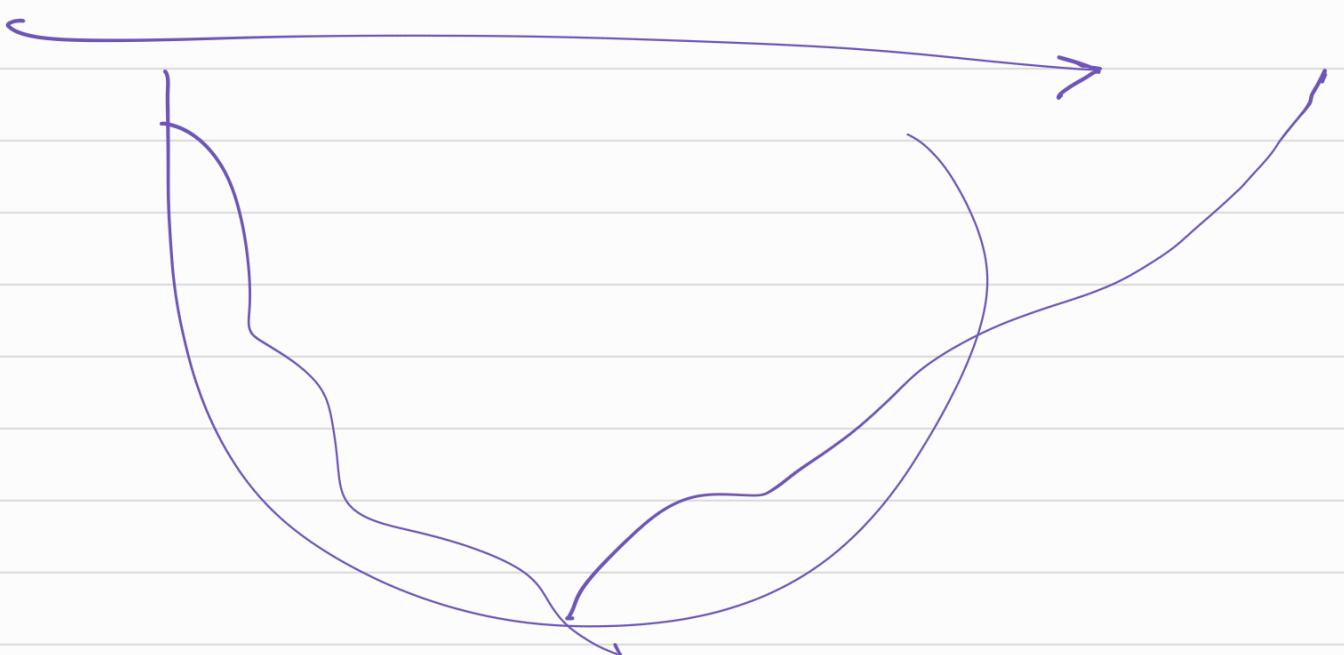
$$\text{update}_2 = \beta \text{update}_1 + (1-\beta) \alpha \frac{\partial h}{\partial w_2}$$

$$\text{update}_3 = \beta \left((1-\beta) \alpha \frac{\partial h}{\partial w_1} \right) + (1-\beta) \alpha \frac{\partial h}{\partial w_3}$$

$$\text{update}_3 = \alpha \beta \frac{\partial L}{\partial w_3} + \beta \text{update}_2 + (1-\beta) \alpha \frac{\partial h}{\partial w_3}$$

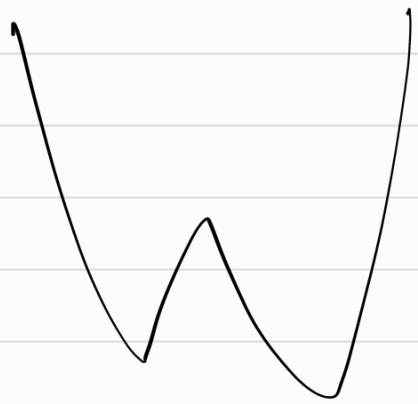
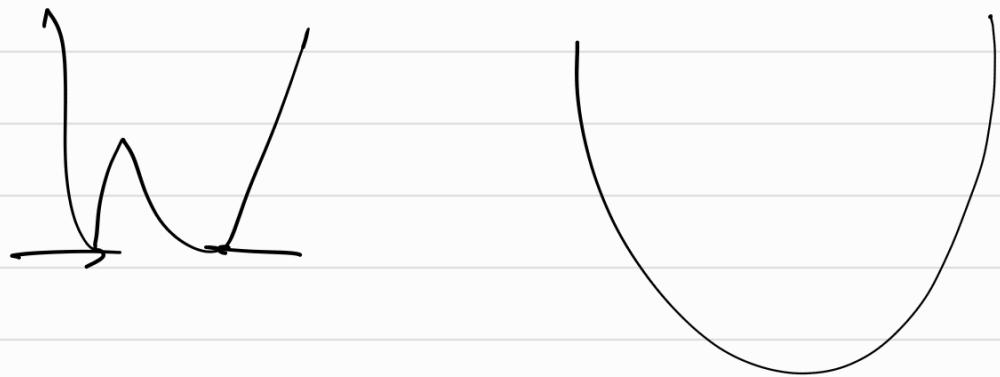
$$\text{update}_3 = \beta \left(\beta (1-\beta) \alpha \frac{\partial h}{\partial w_1} + (1-\beta) \alpha \frac{\partial h}{\partial w_2} \right) + (1-\beta) \alpha \frac{\partial h}{\partial w_3}$$

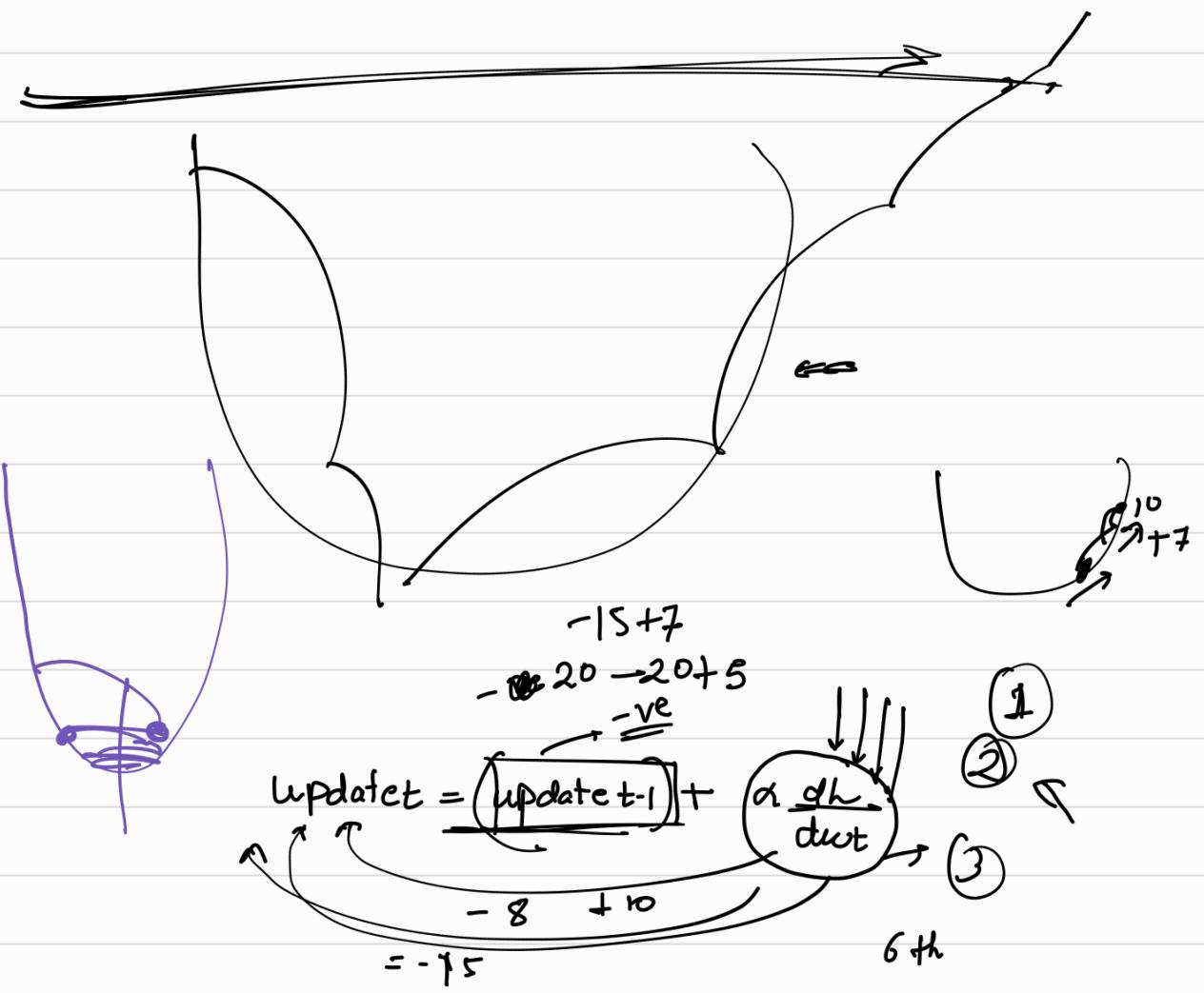
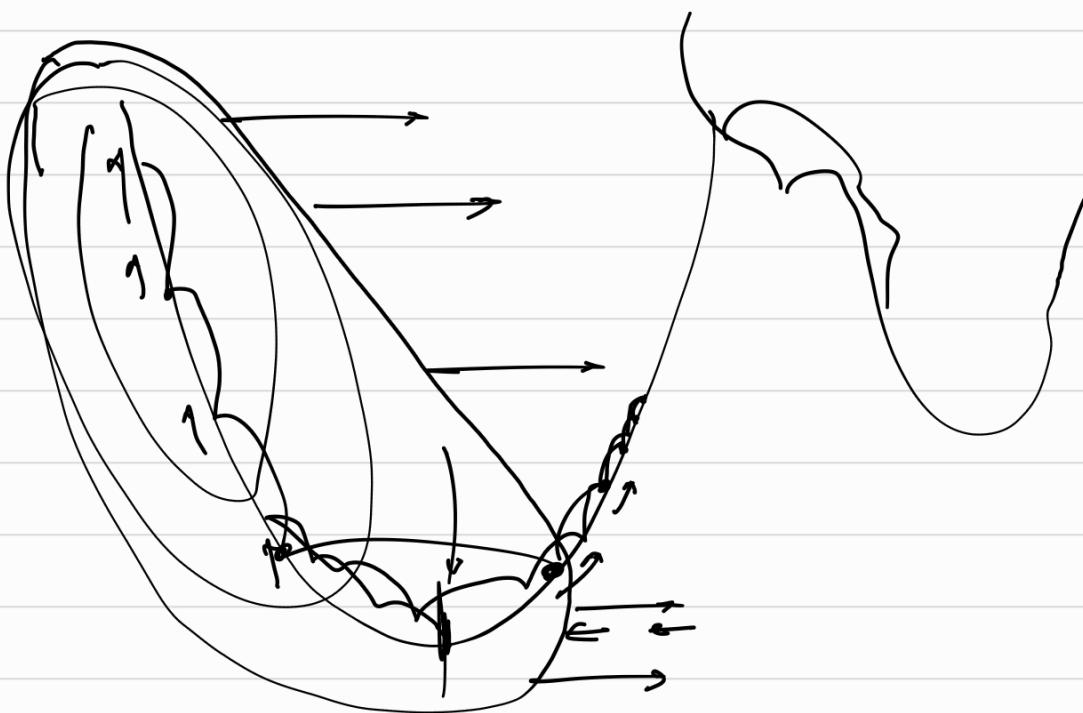
$\text{update}_3 = \beta \left(\beta (1-\beta) \alpha \frac{\partial h}{\partial w_1} + (1-\beta) \alpha \frac{\partial h}{\partial w_2} \right) + (1-\beta) \alpha \frac{\partial h}{\partial w_3}$

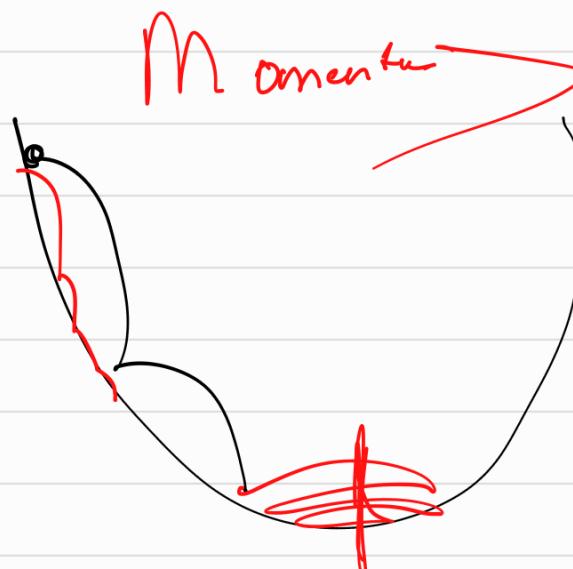
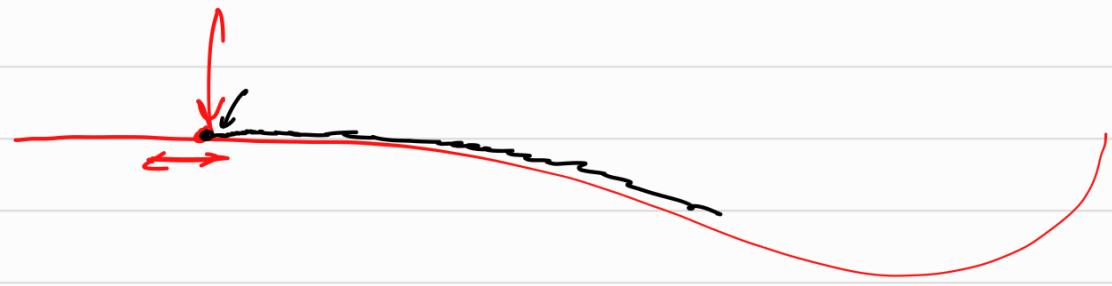
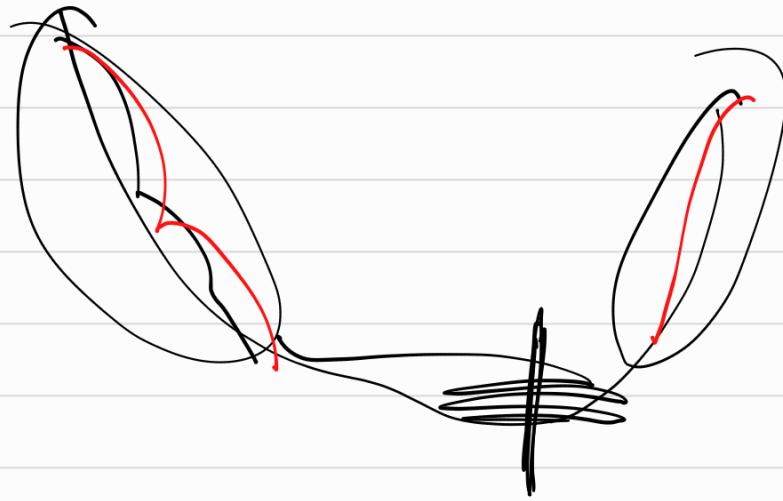


$i = i + \underline{\text{S}}$

25







$$\alpha_0 = 0$$

$$\alpha_t = \alpha_{t-1} + s$$

$$a_1 = 0 + s$$

$$a_1 = s$$

$$a_2 = a_1 + s$$

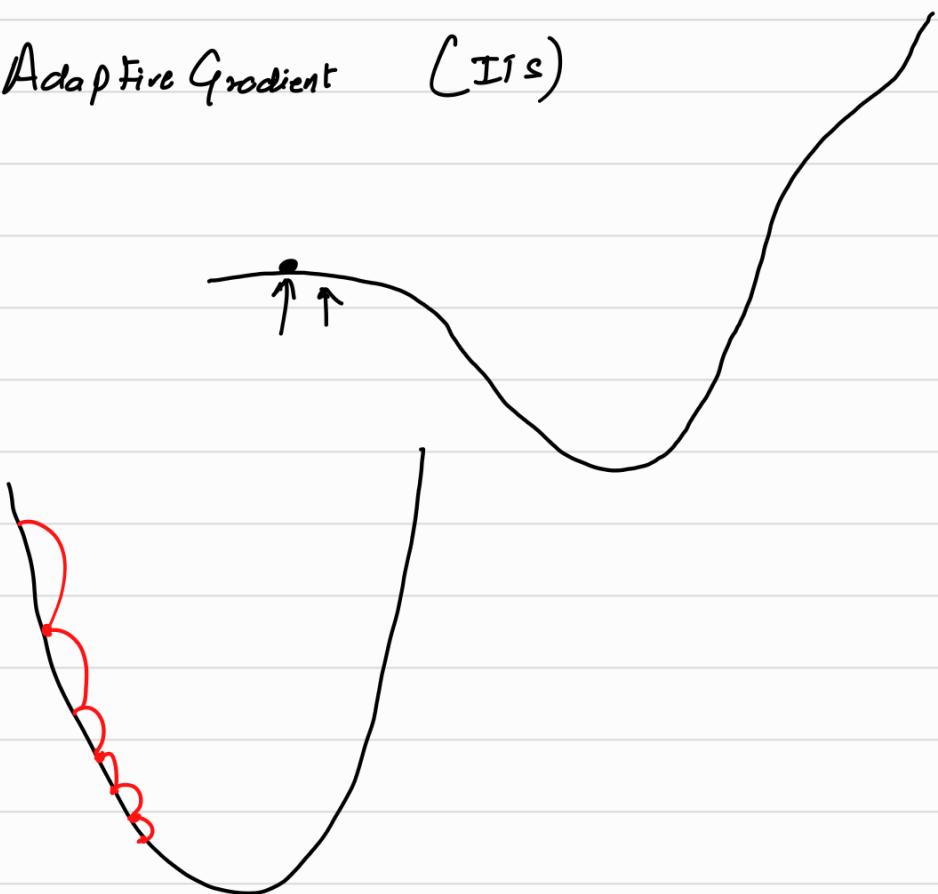
$$a_2 = 10$$

$$a_3 = a_2 + s$$

$$a_3 = (a_1 + s) + s$$

Break!

Adagrad: Adaptive Gradient (It's)



$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_{t+1}}} \frac{dh}{d w_t}$$
$$v_t = v_{t-1} + \left(\frac{dh}{d w_t} \right)^2$$

Rms Prop

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_{t+\epsilon}}} \frac{\partial h}{\partial w_t}$$

v_t $B v_{t-1} + \frac{(1-B)}{\tau} \left(\frac{\partial h}{\partial w_t} \right)^2$

Adam:

Adaptive moments

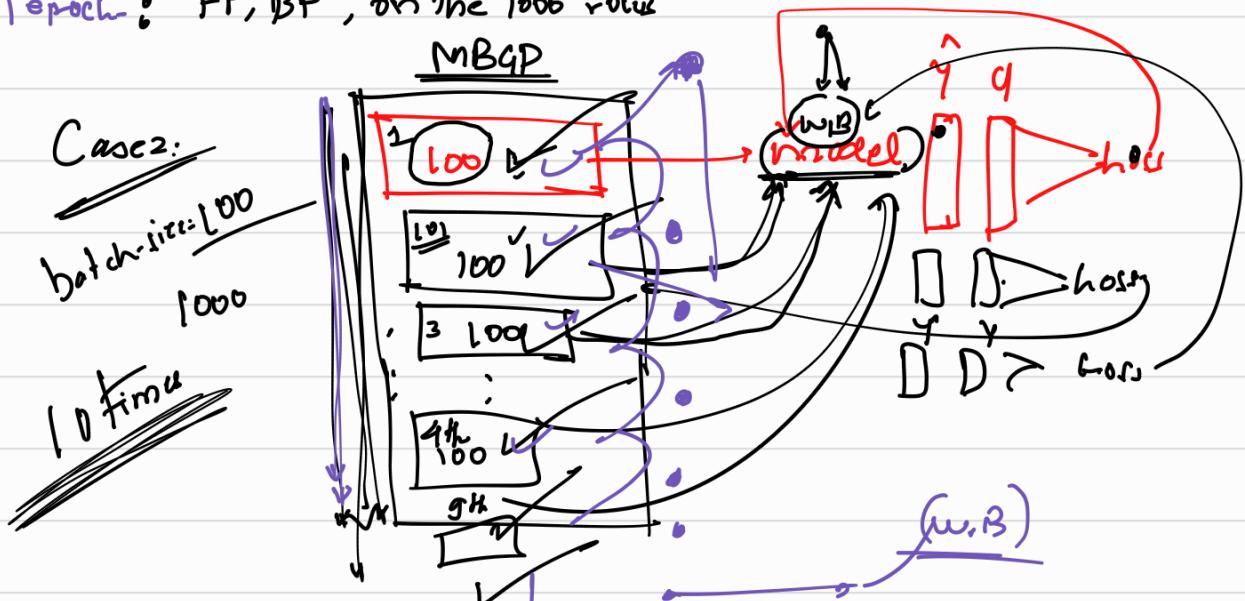
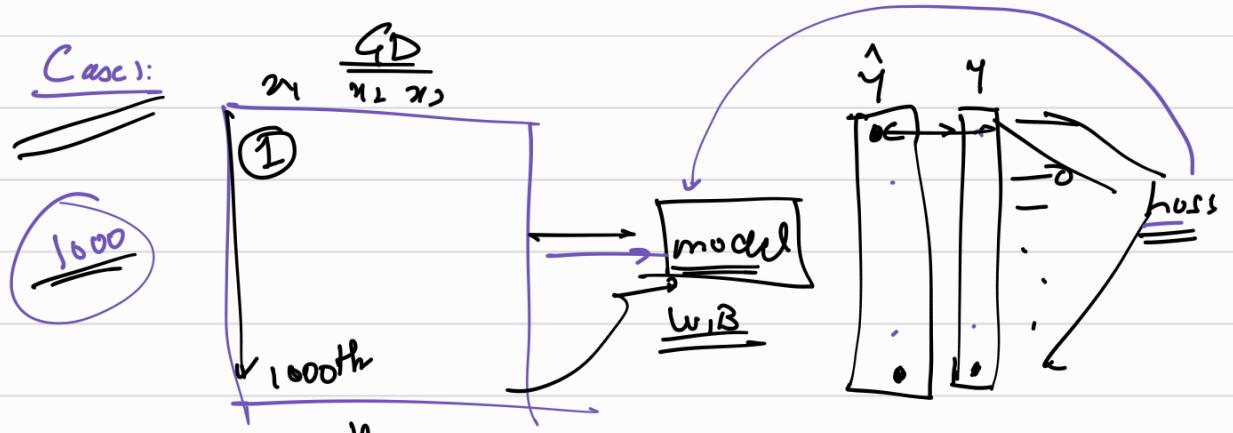
$$\hat{m}_t = \beta_1 m_{t-1} + (1-\beta_1) \frac{\partial h}{\partial w_t}$$

$$\hat{v}_t = \beta_2 (\hat{v}_{t-1}) + (1-\beta_2) \left(\frac{\partial h}{\partial w_t} \right)^2$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_{t+\epsilon}}} \times \hat{m}_t$$

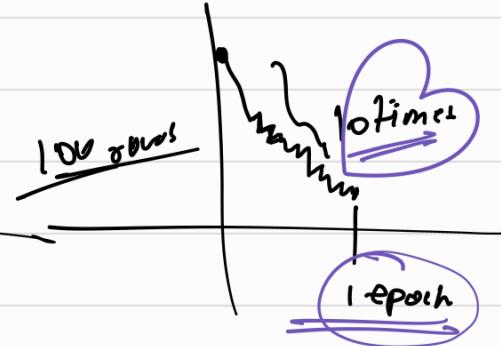
Mini Batch Gradient Descent \rightarrow Batch Normalization

Calculus



Case 1 randomly without replacement

Case 2



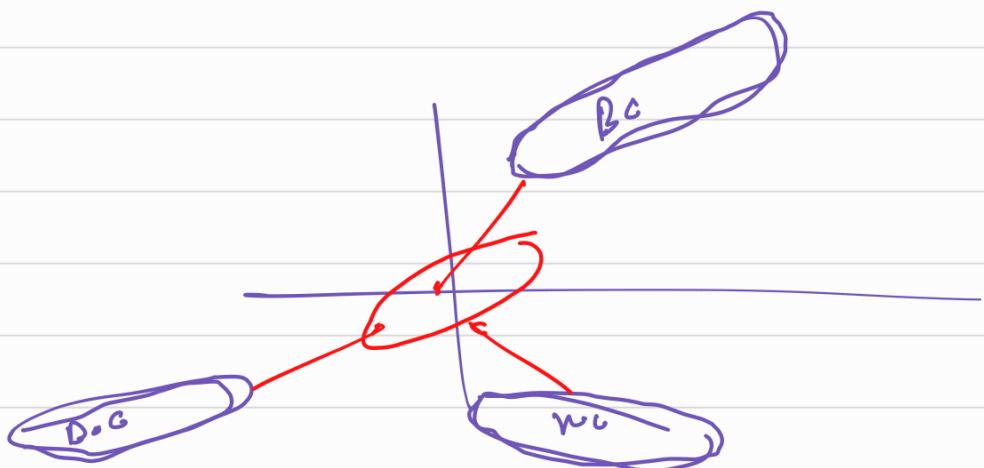
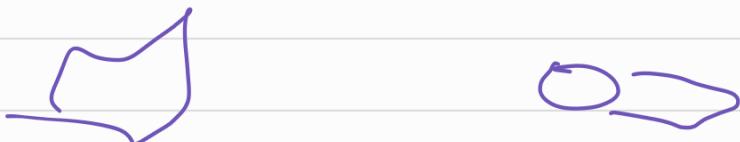
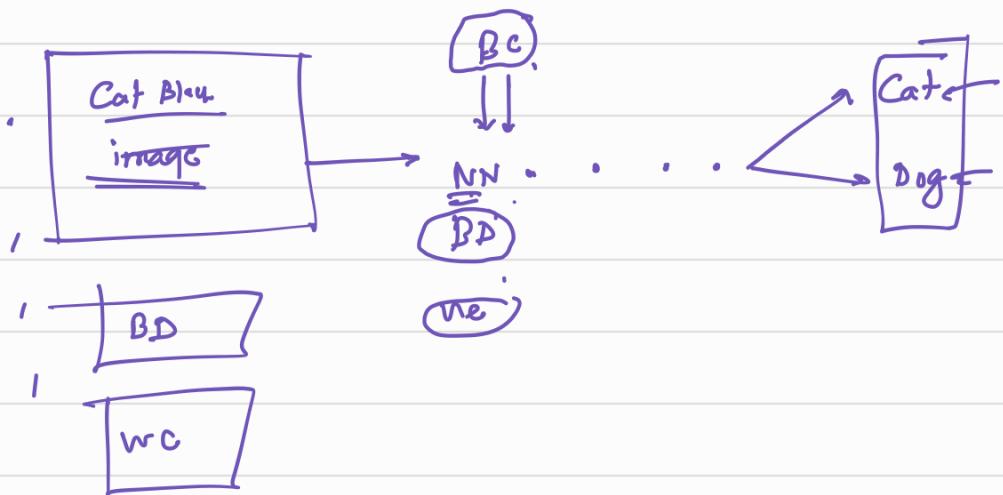
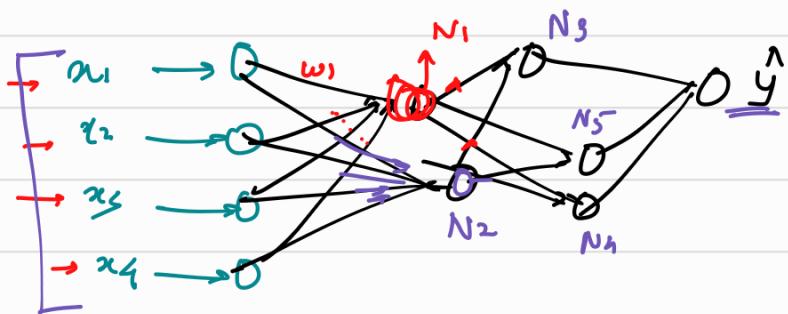
model-fit (∞)

model-fit (20)

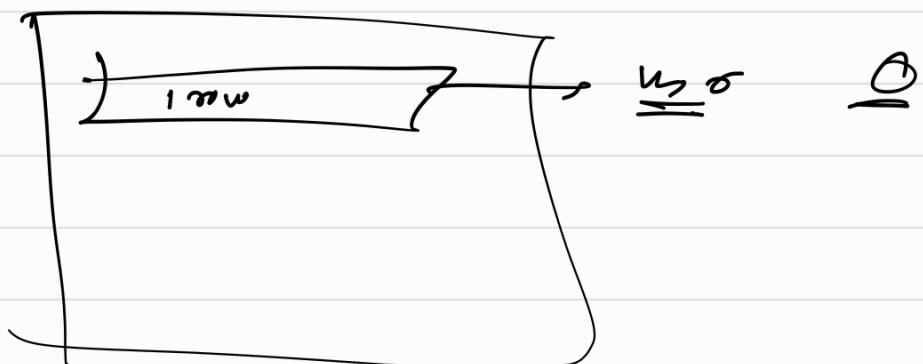
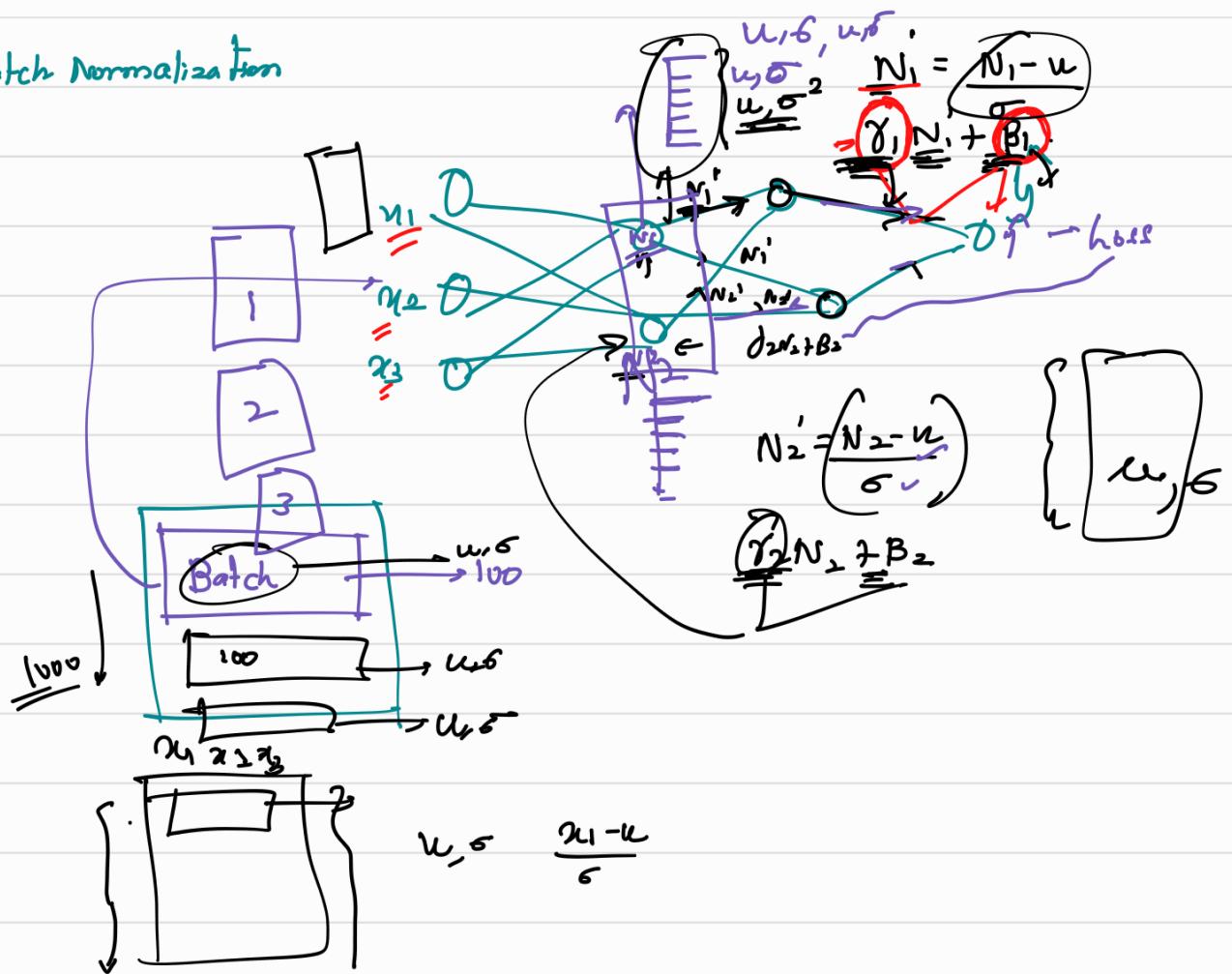
Batch Normalization ✓

Callbacks ←

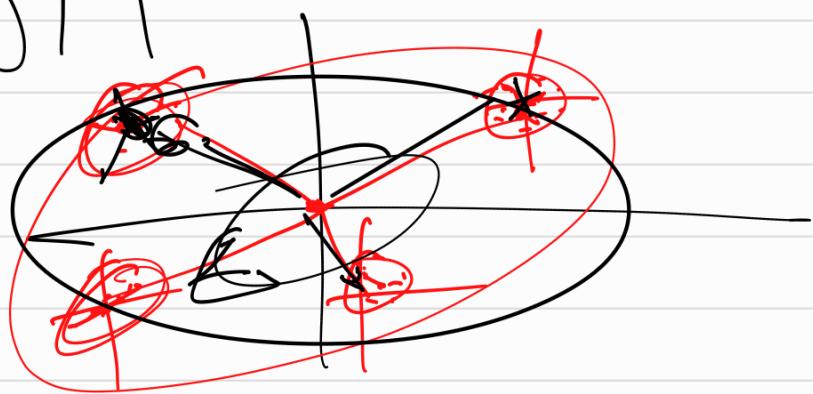
Batch Normalization

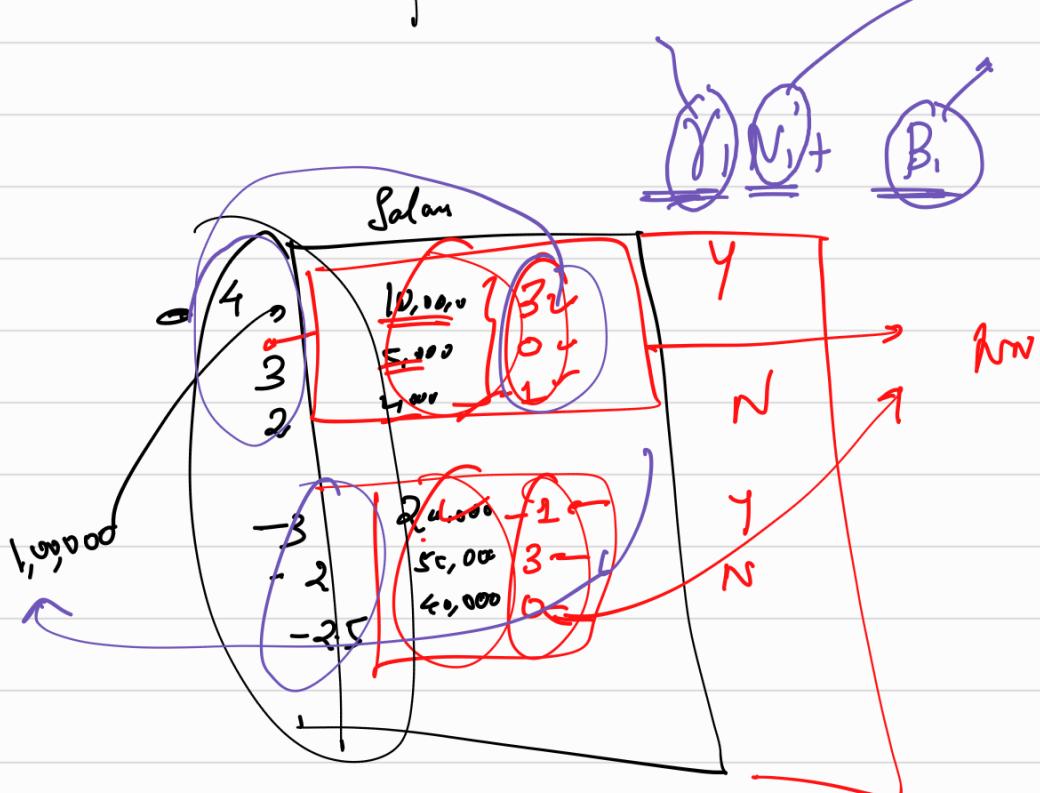
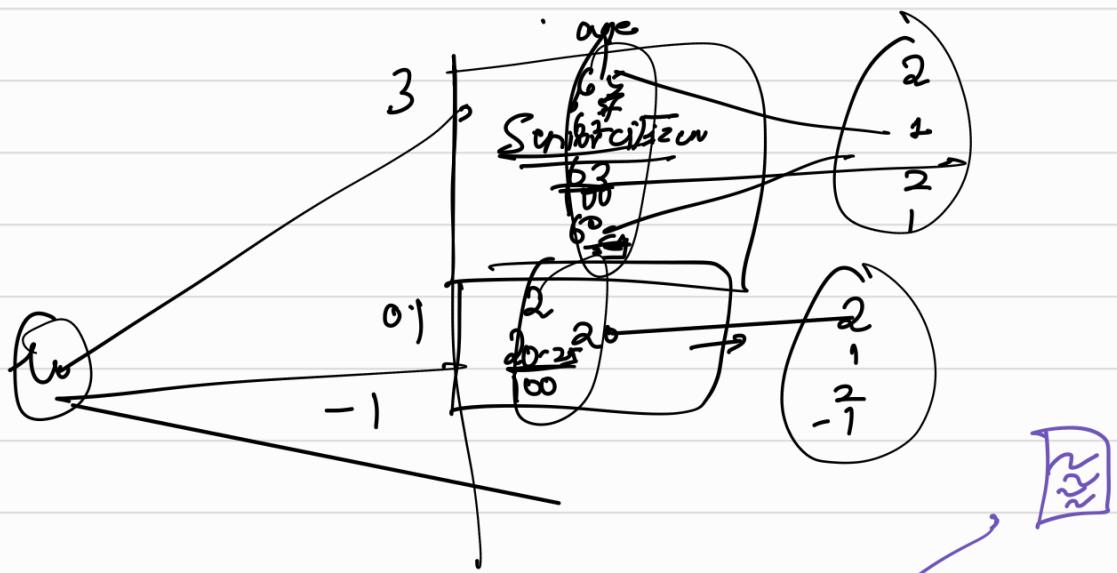
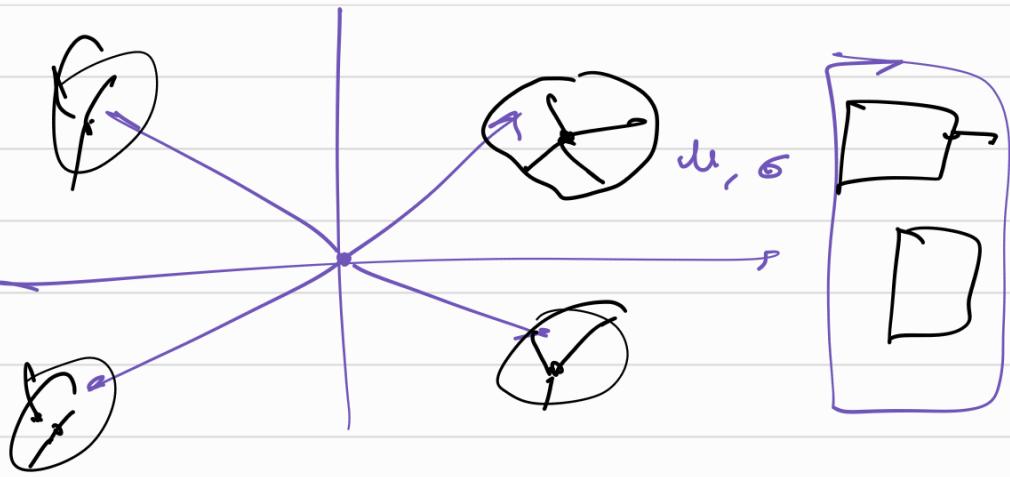


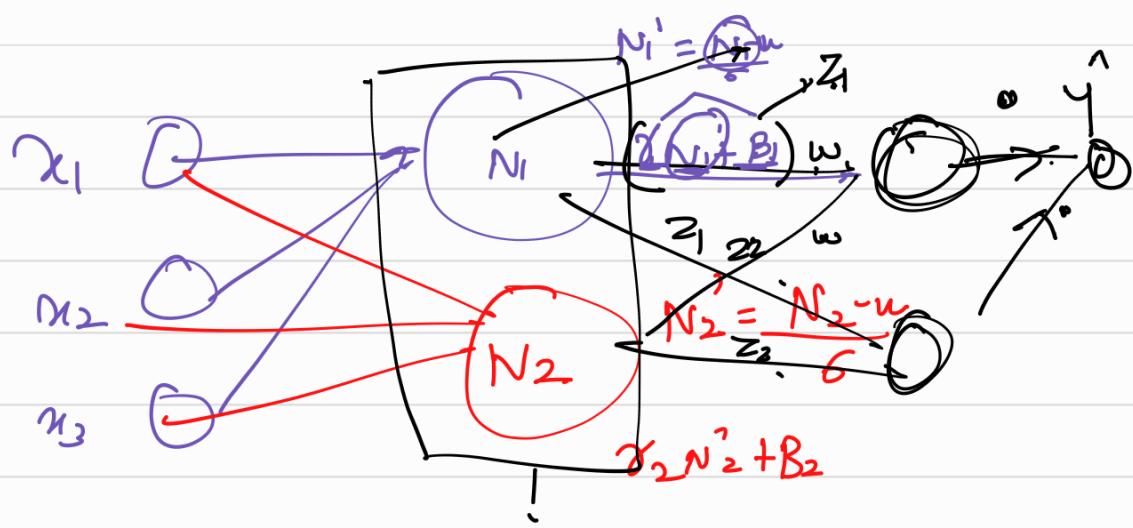
Batch Normalization



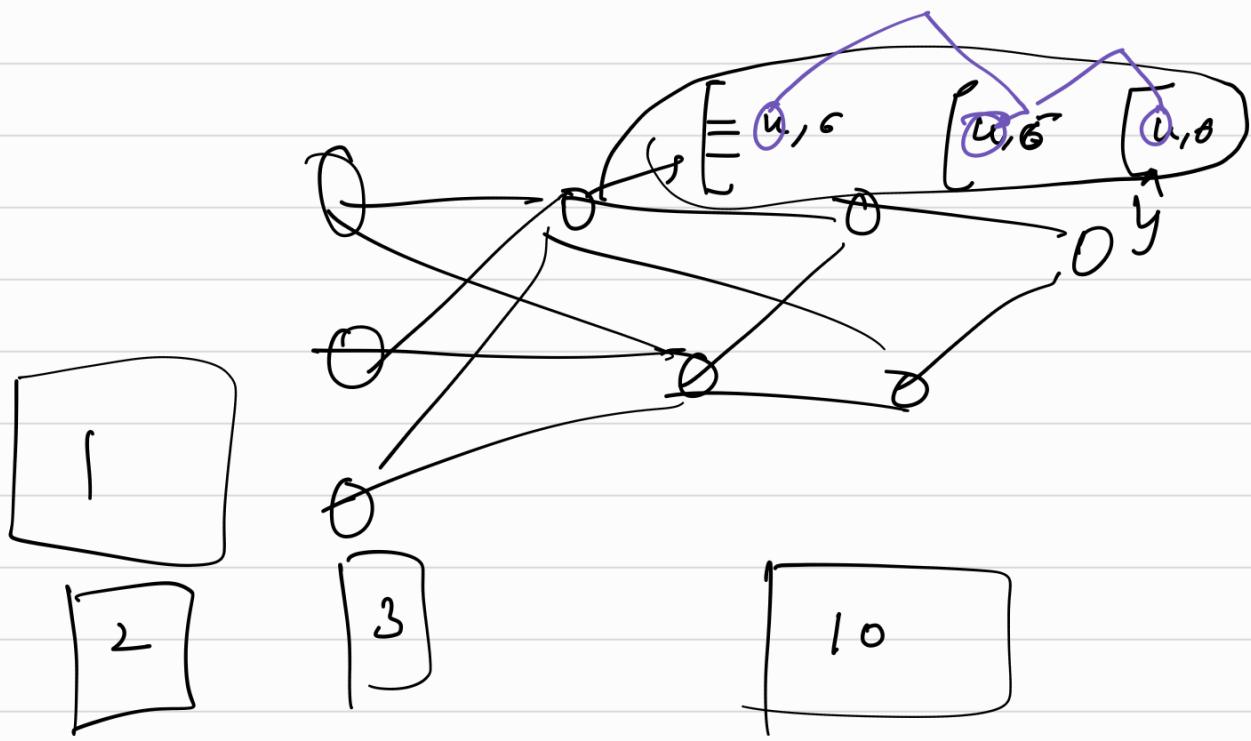
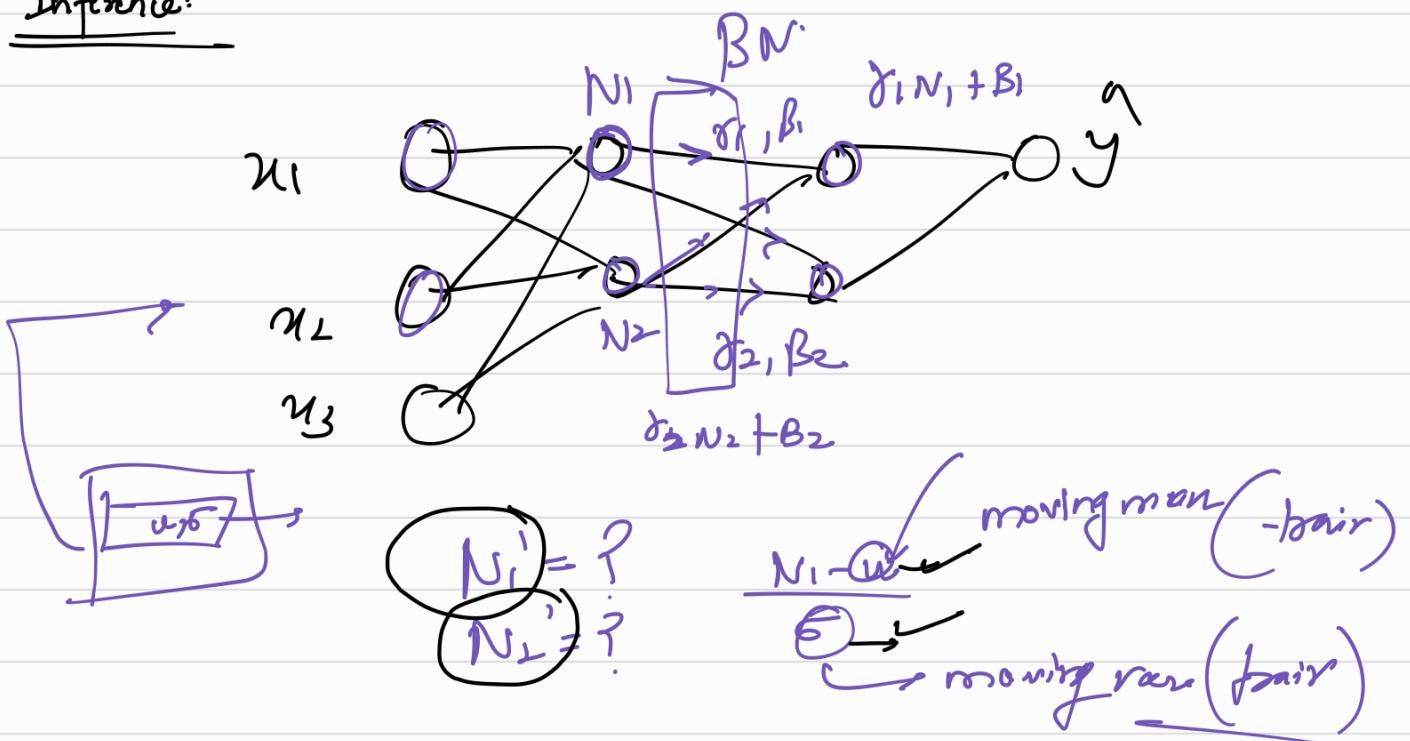
γ_1 , β_1







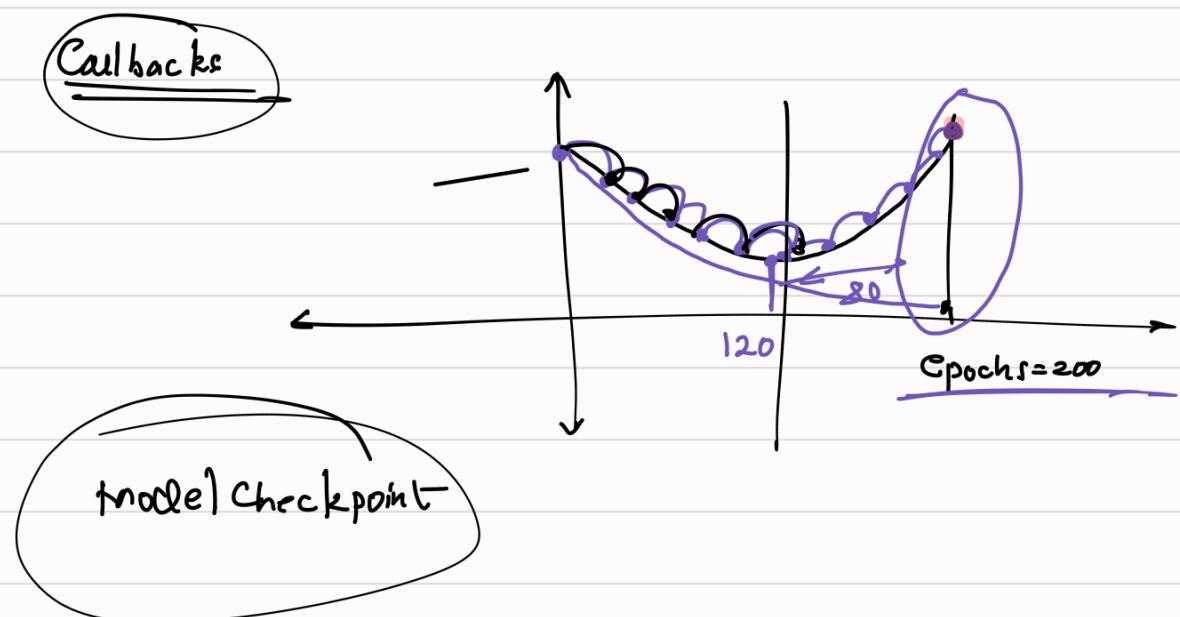
Inference:



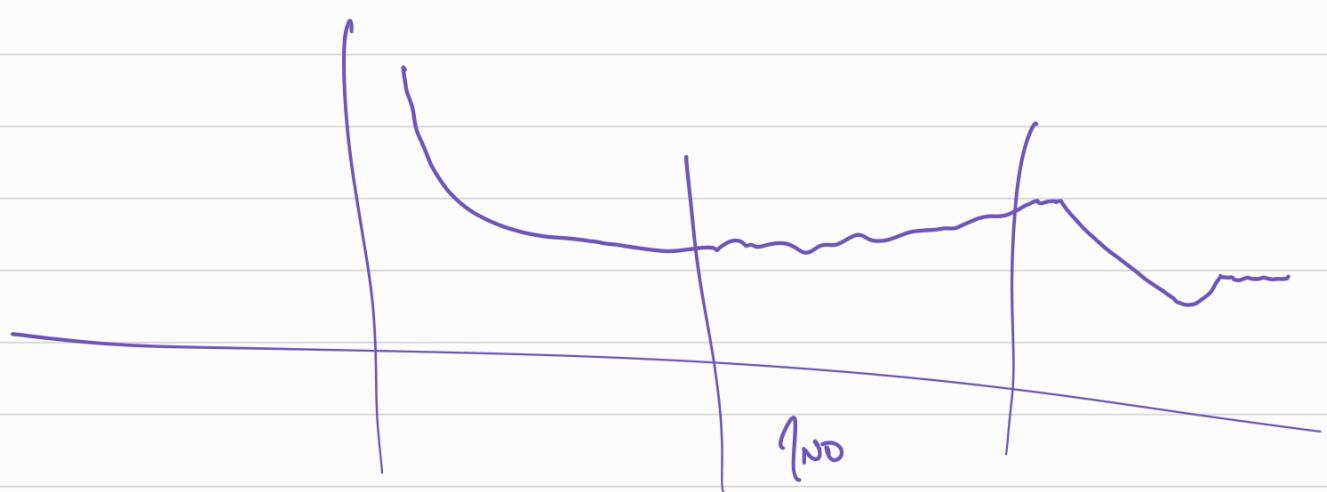
N1

moving mean $\rightarrow \beta \times \text{moving mean} + (1-\beta) \times \text{mean(batch)}$

moving var $\rightarrow \beta \times \text{moving var} + (1-\beta) \times \text{var(batch)}$



Early Stopping

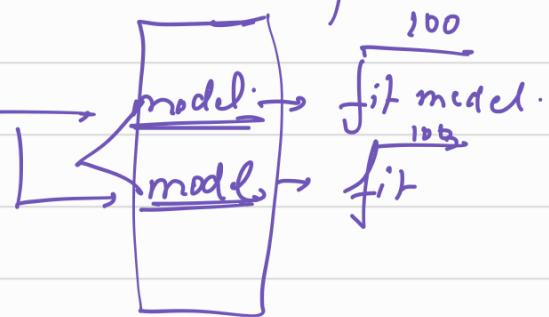




monotonic, = 3

$\{3, 32, \text{relu}, d=T, h=10^{-2}\}$

Execution parallel: 2



$[4, 32, \text{relu}, d=N, h=10^{-4}]$

$[4, 64, \tanh, d=T, h=10^{-3}]$

11:33 AM

Input layer (784) 1C

log sampling

Hidden layer (200, 400, activation (tanh, relu)).

Hidden layer (100, 200, activation (tanh, relu)).

2,4 { [Hidden (50, 100, activation (tanh, relu)) $\lambda = 0.1, 0.5$]

Batch Norm () \leftarrow maybe add / not

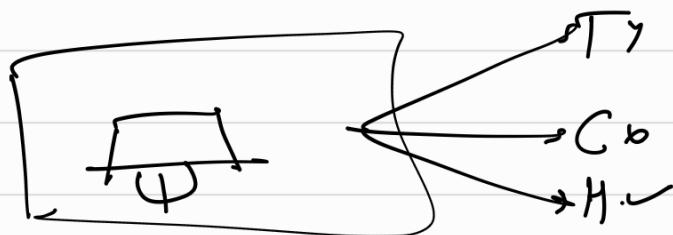
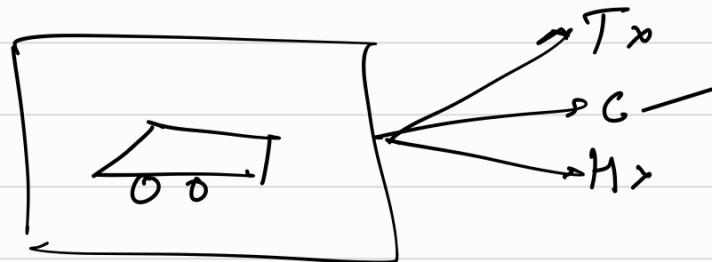
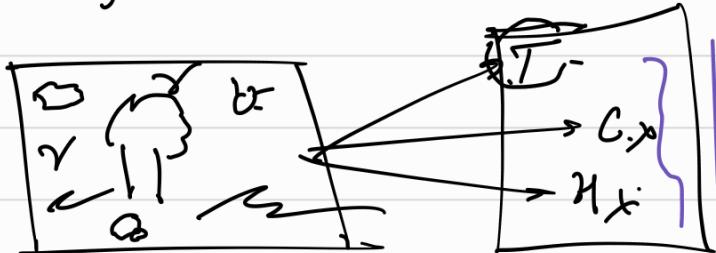
model. layers (Dense (10, 50) final).

Introduction to Computer Vision

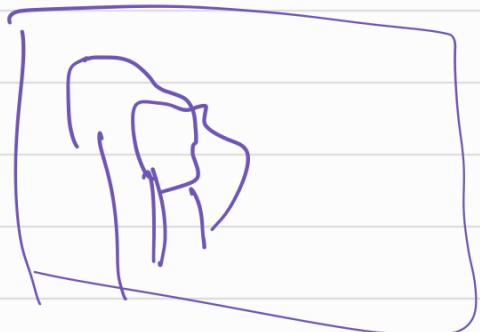
①

Image Classification

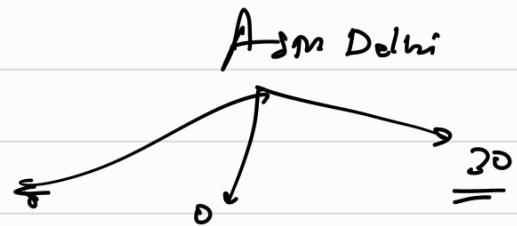
image



Object detection & localization



Bounding box



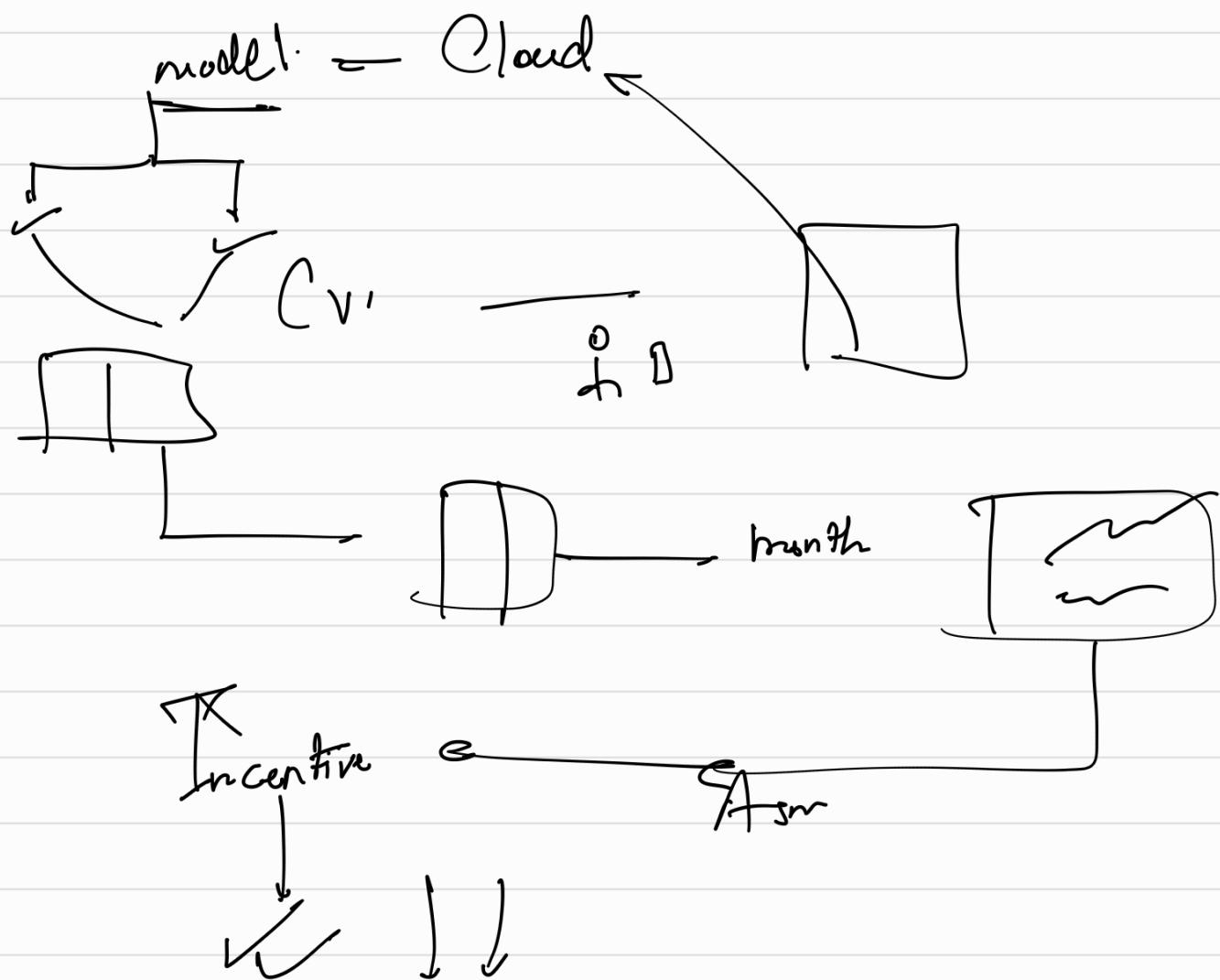
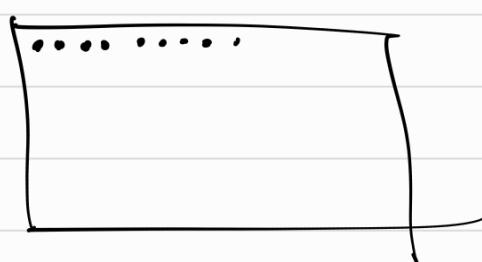
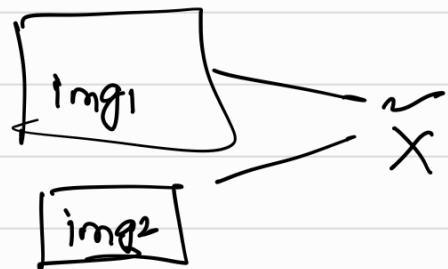


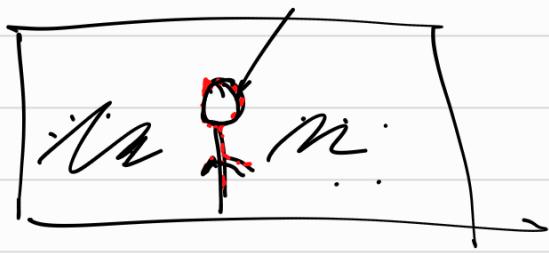
Image Segmentation



④ Game networks

FR



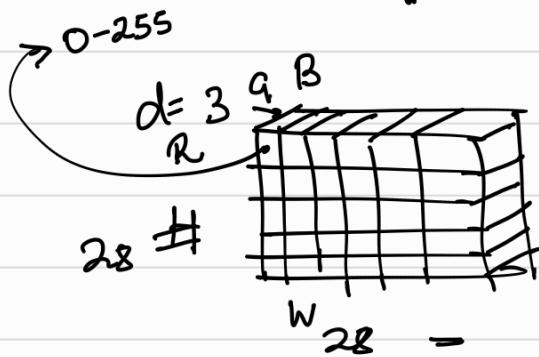


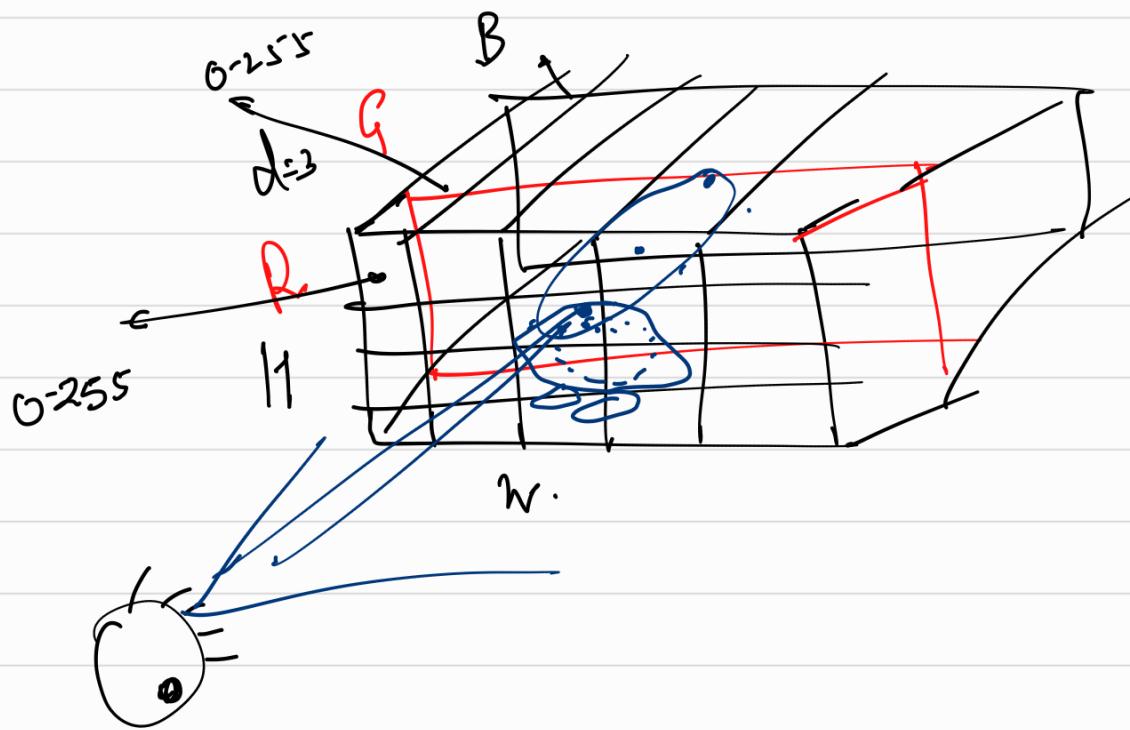
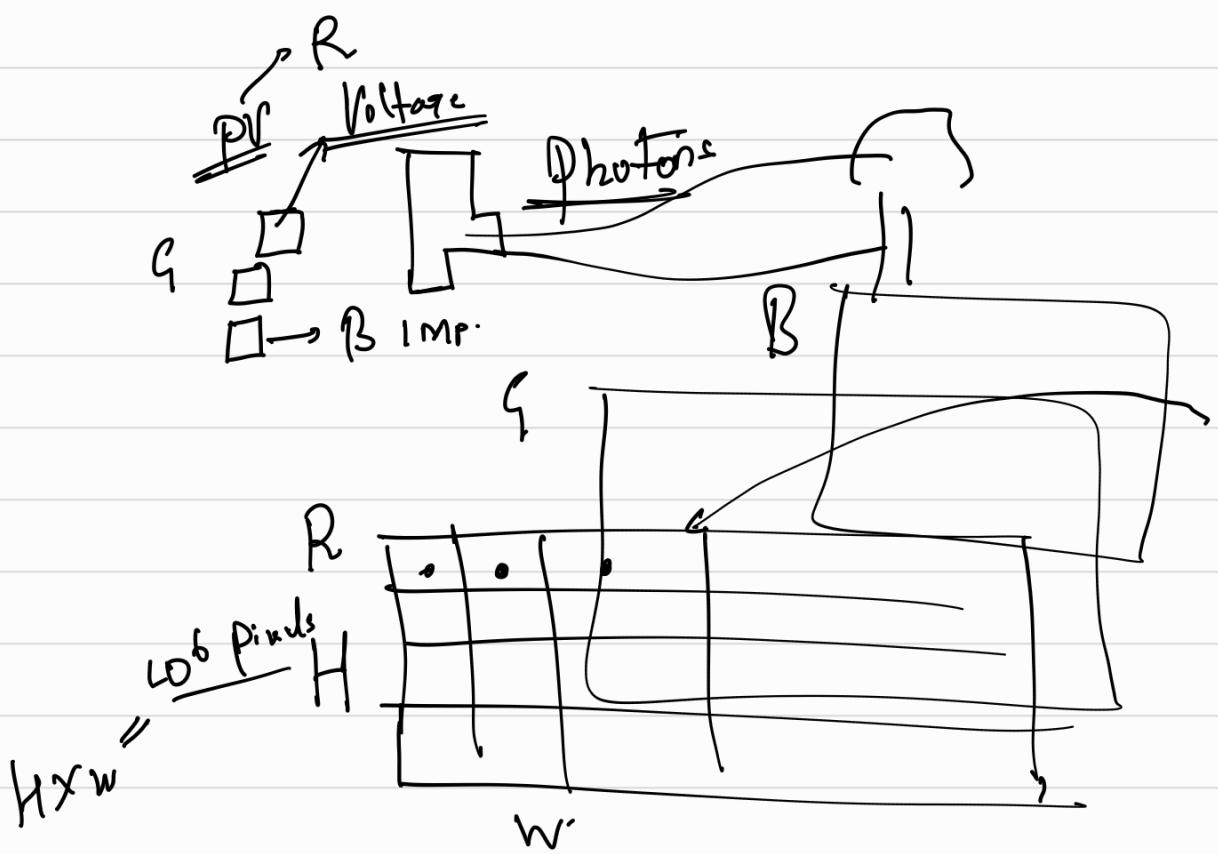
⑤ GAN → Gen Adver Networks.

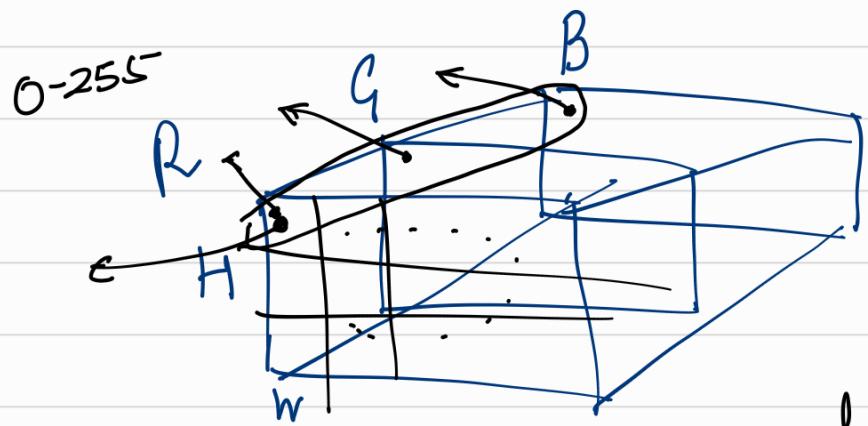


What is an image?

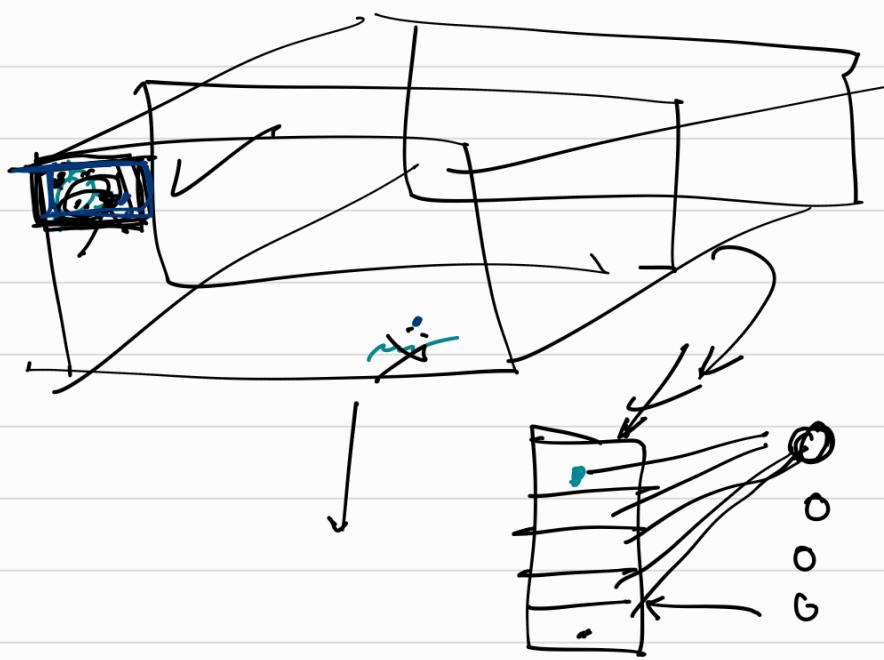
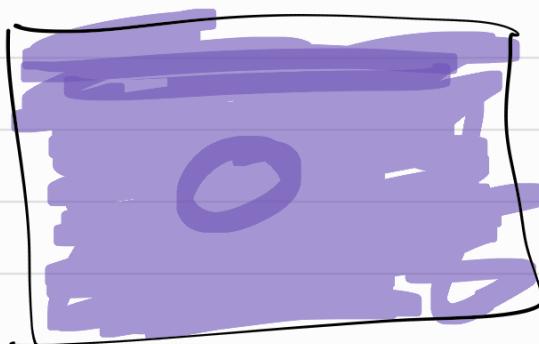
R, G, B

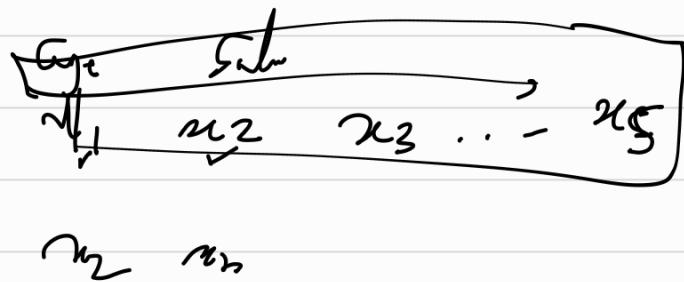






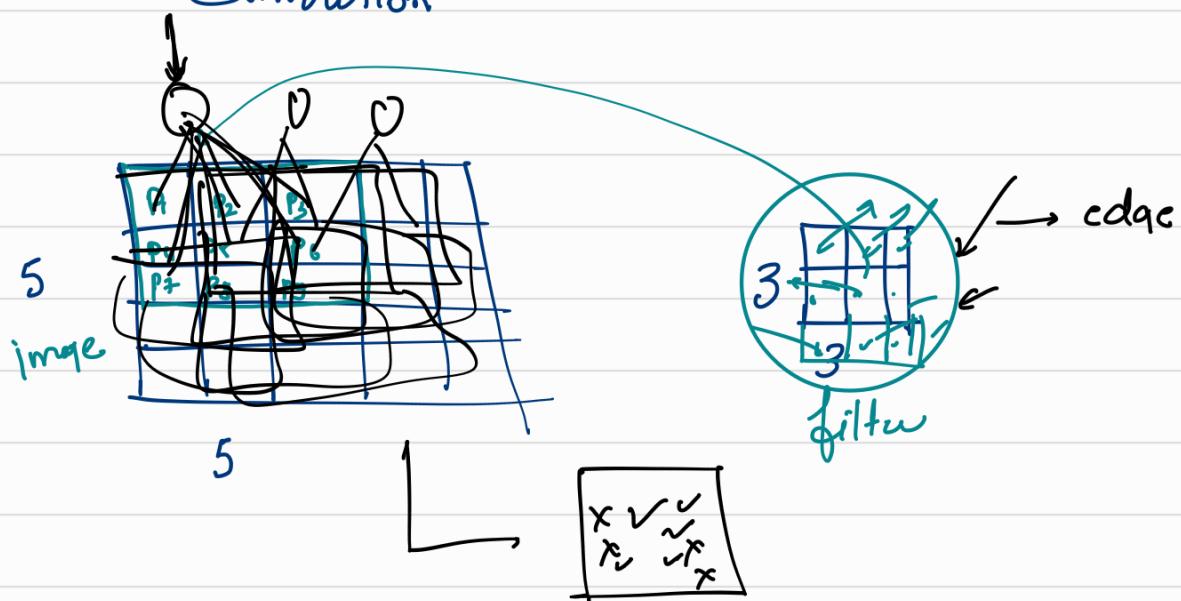
Information → Variation in the local intensity



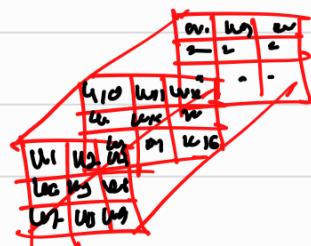
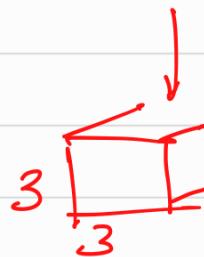
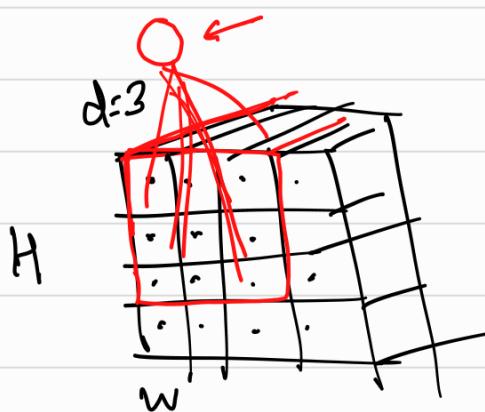


Filters

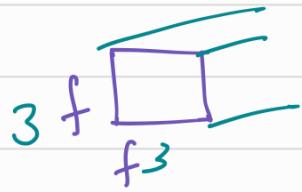
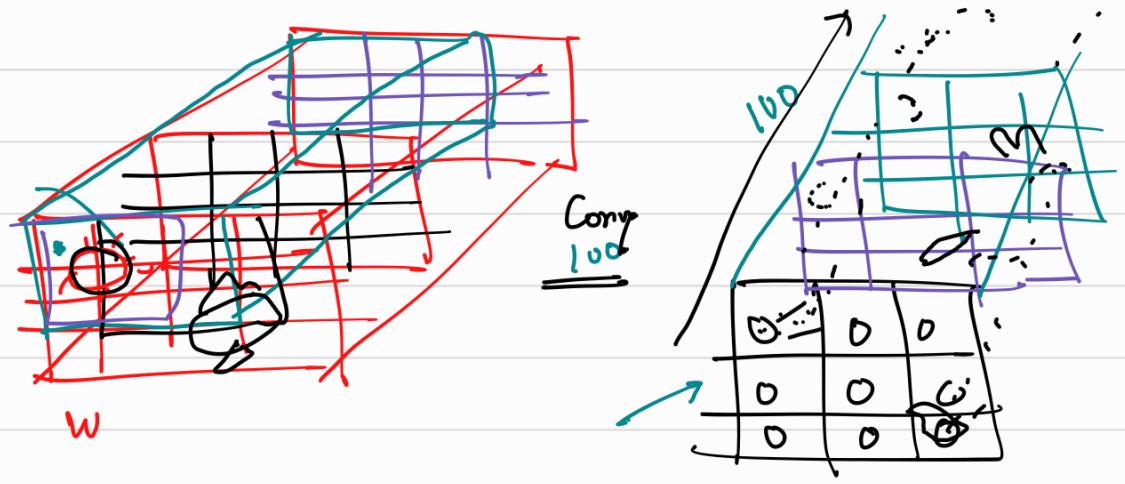
Convolution



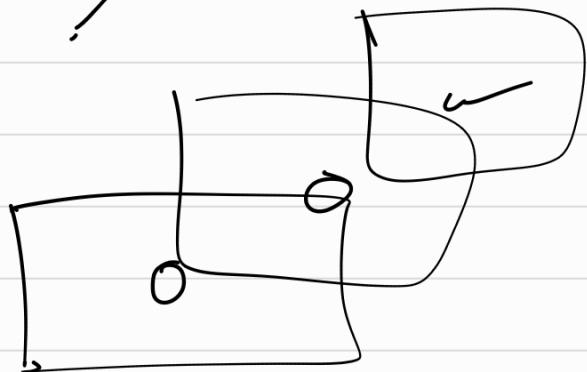
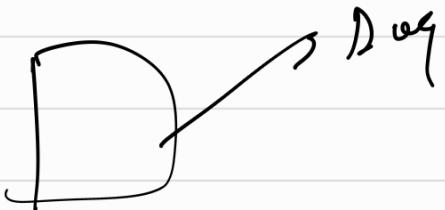
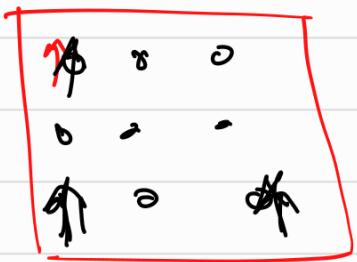
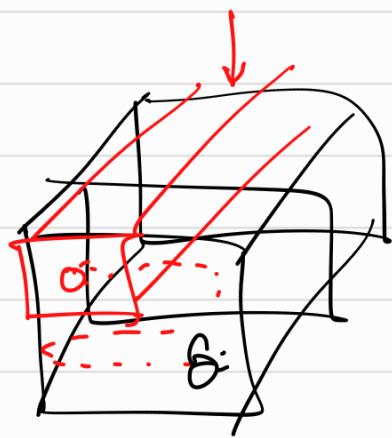
Convolutional Neural Network

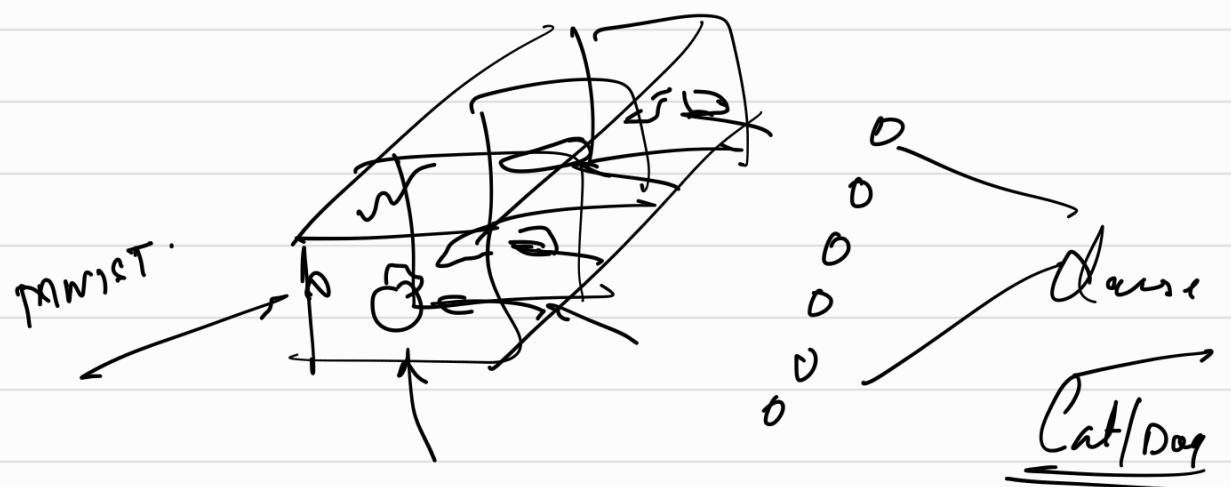
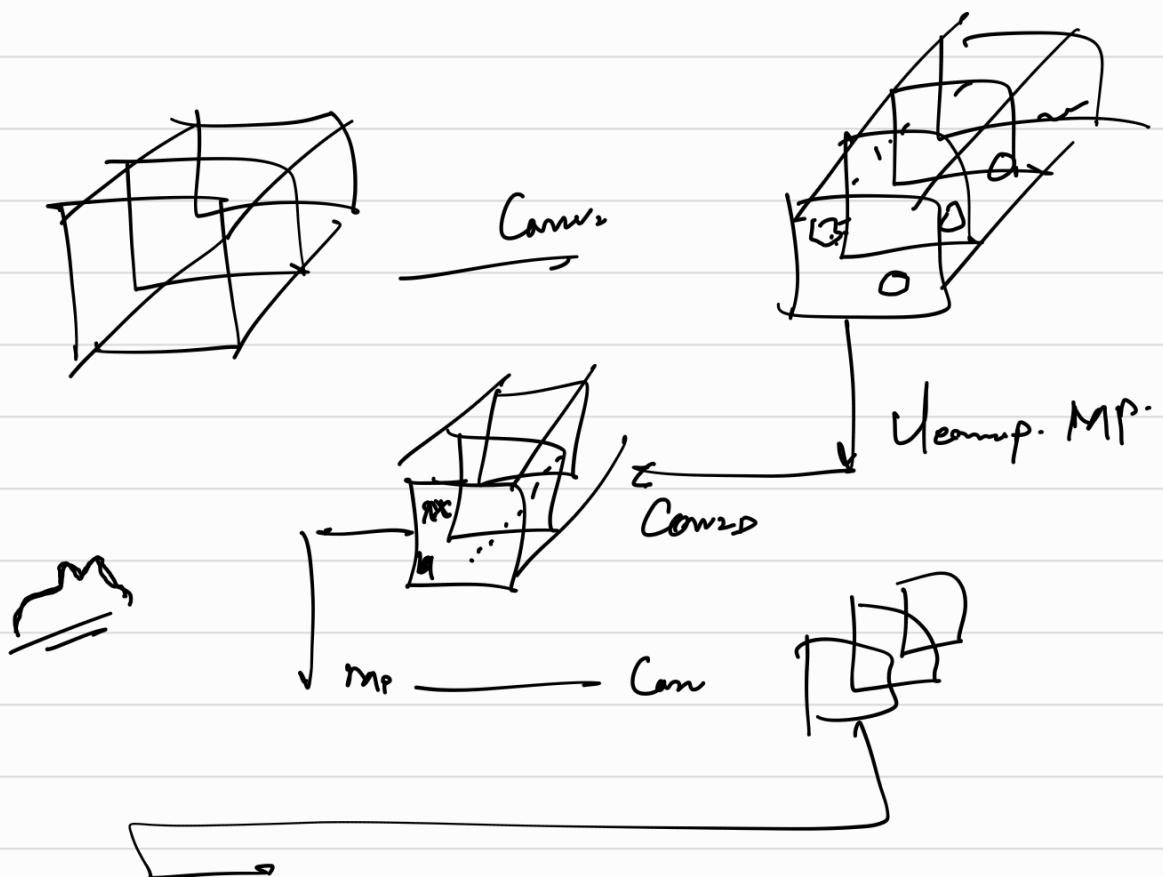


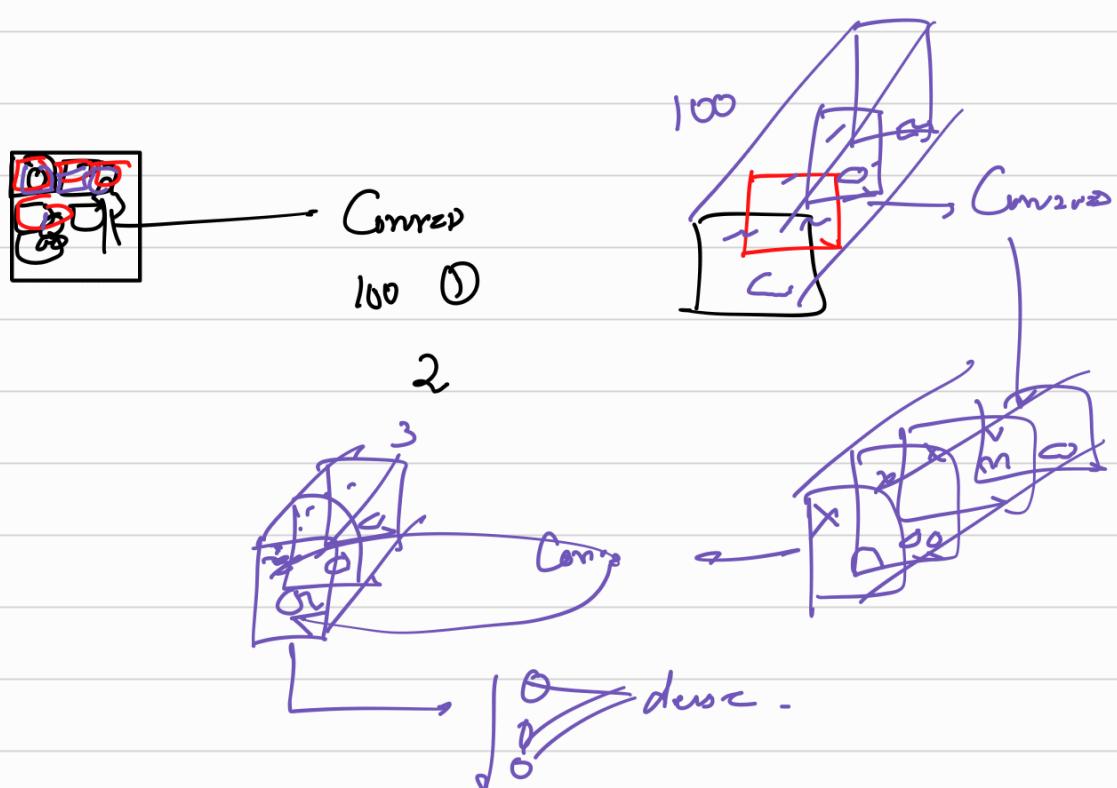
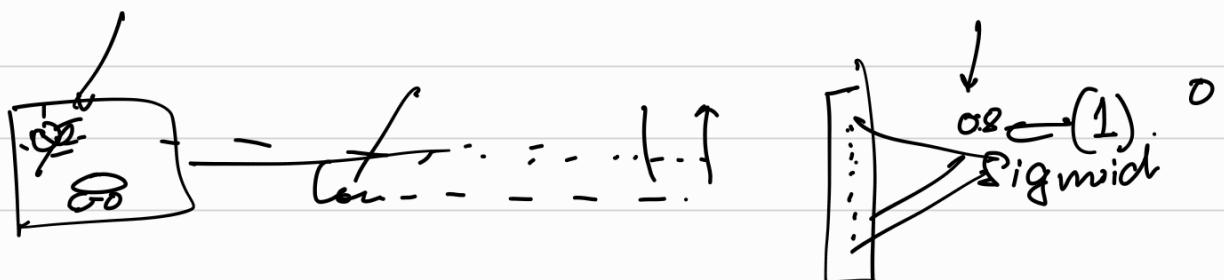
$$\text{tanh}(p_1 w_1 + p_2 w_2 + p_3 w_3 + \dots + p_{27} w_{27})$$

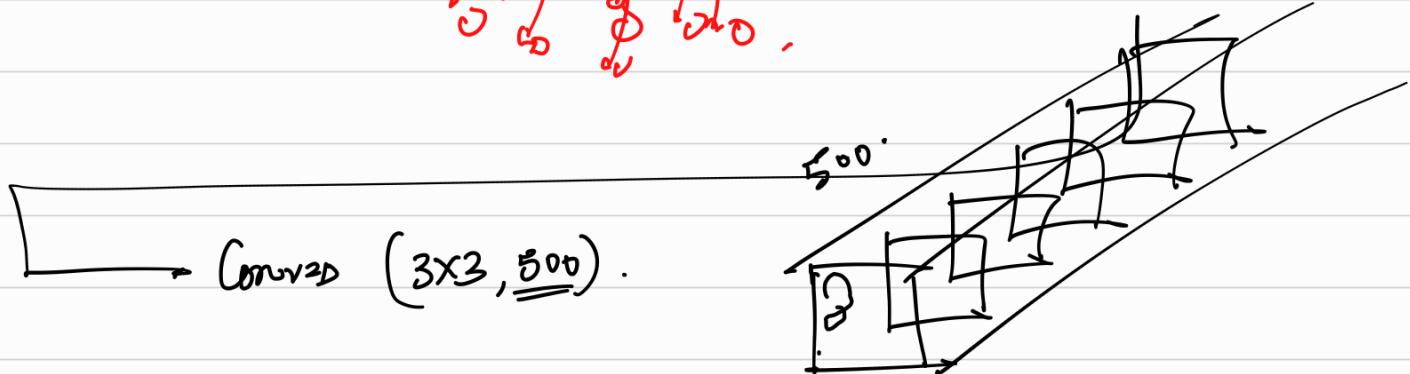
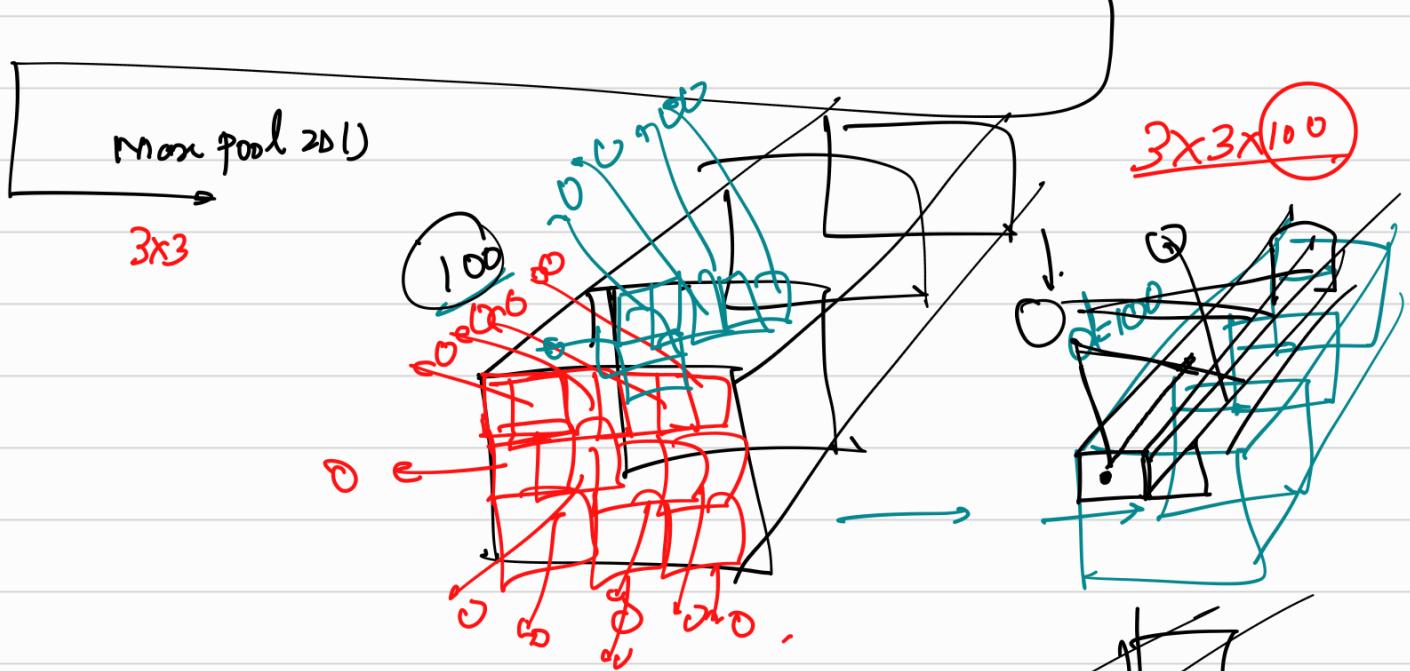
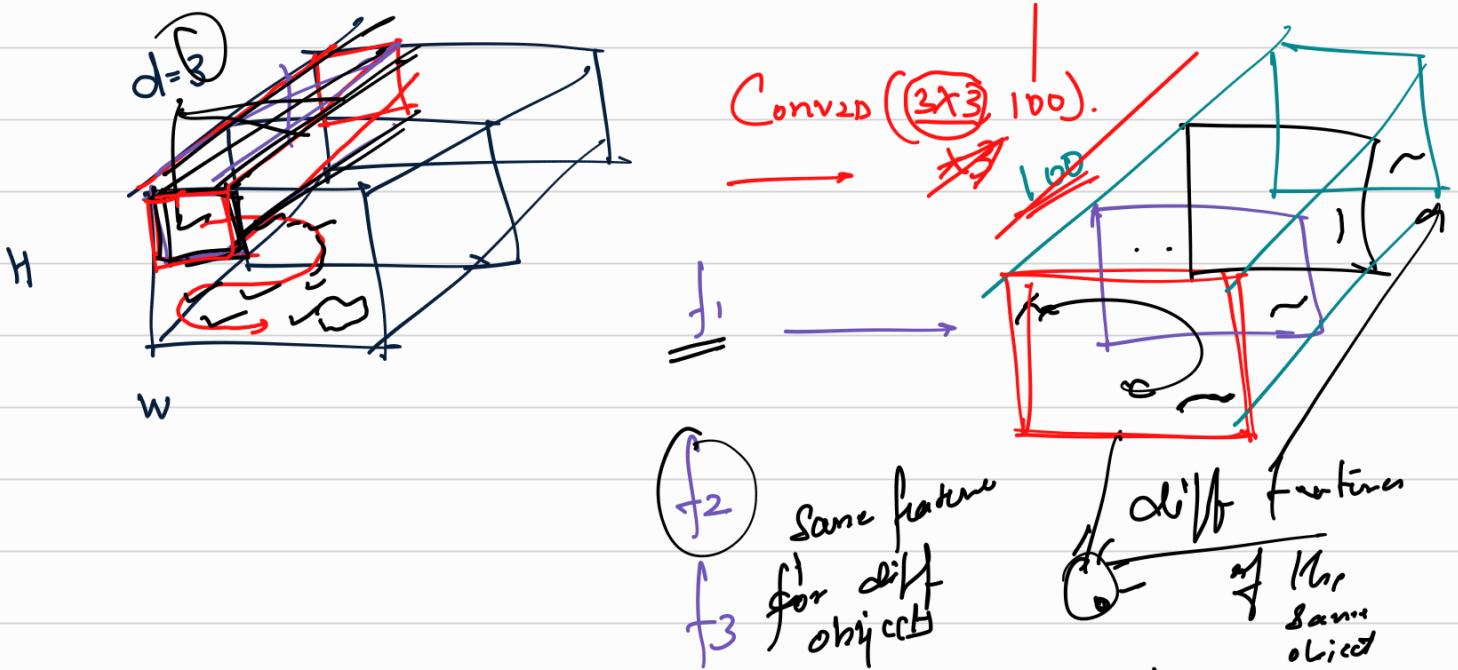


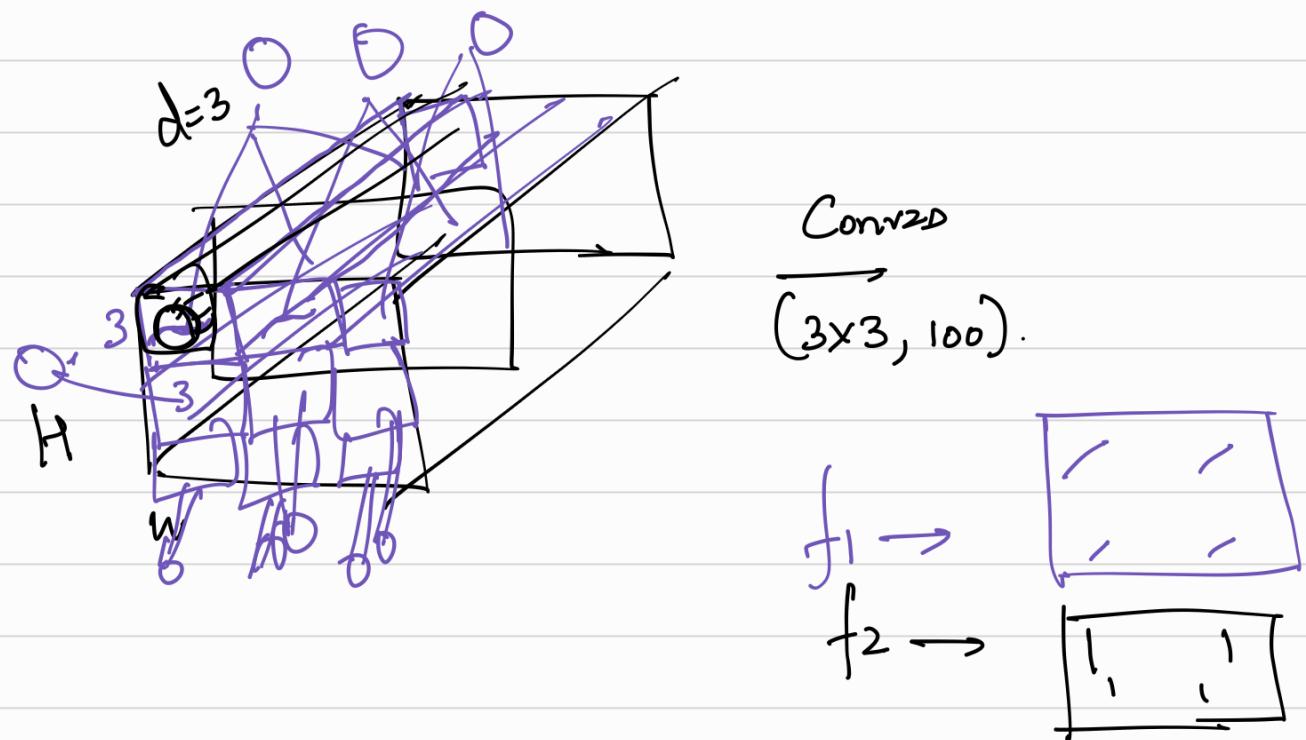
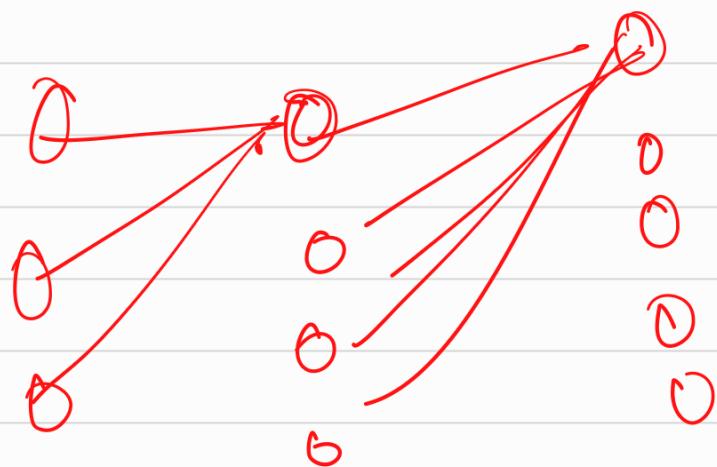
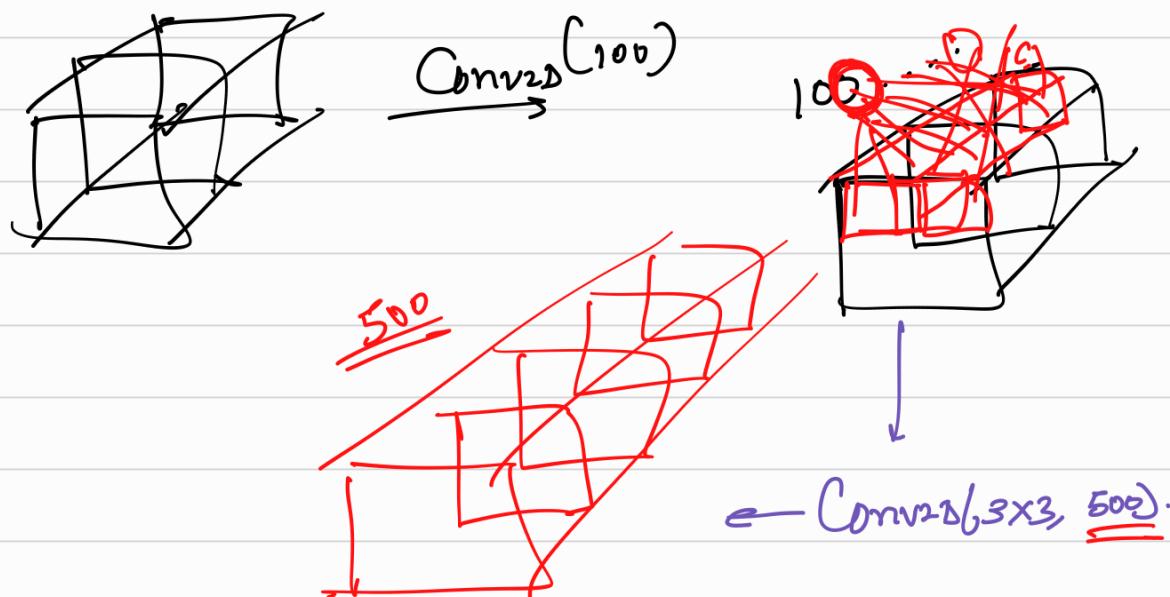
① If the filters (100) learnt the weights to extract the dog's features, then the feature maps will be containing mainly the features of the dog and very slight features of other objects like tree/bird/sun.

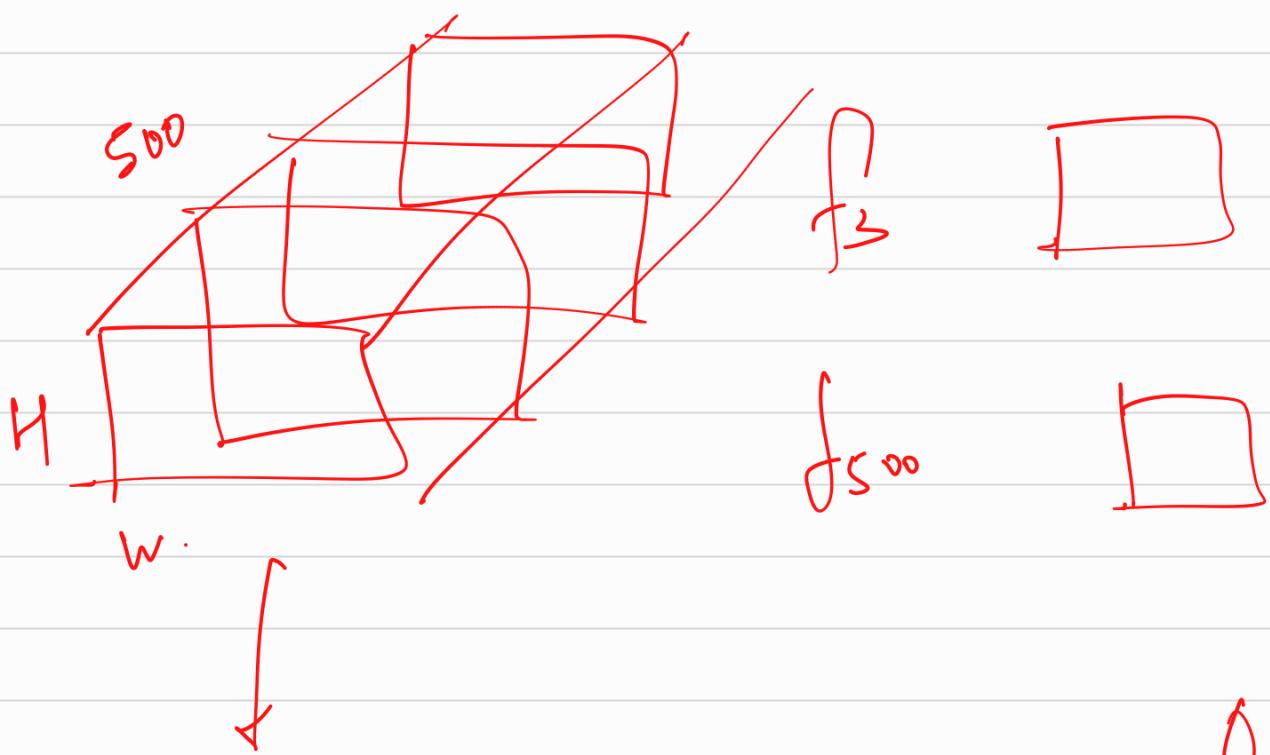
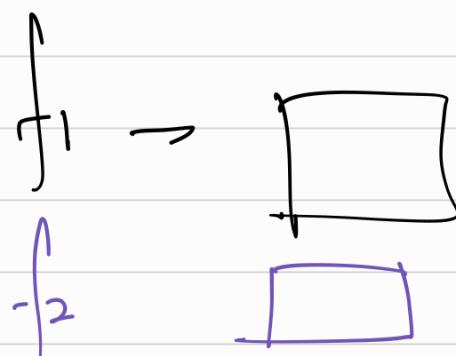
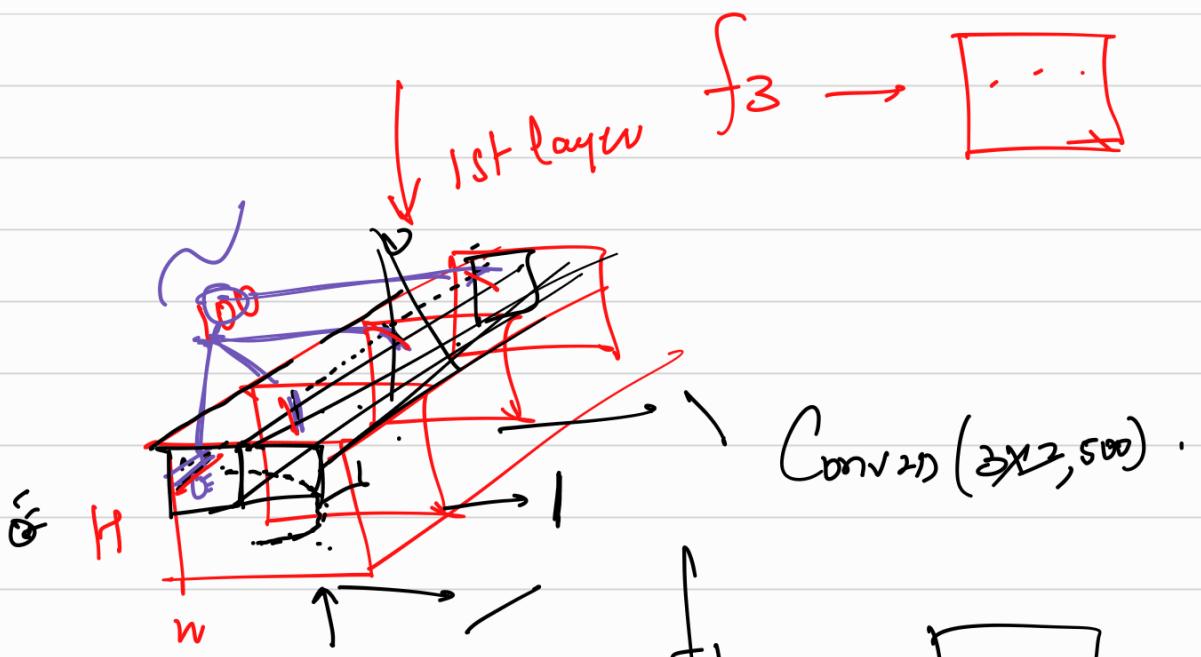






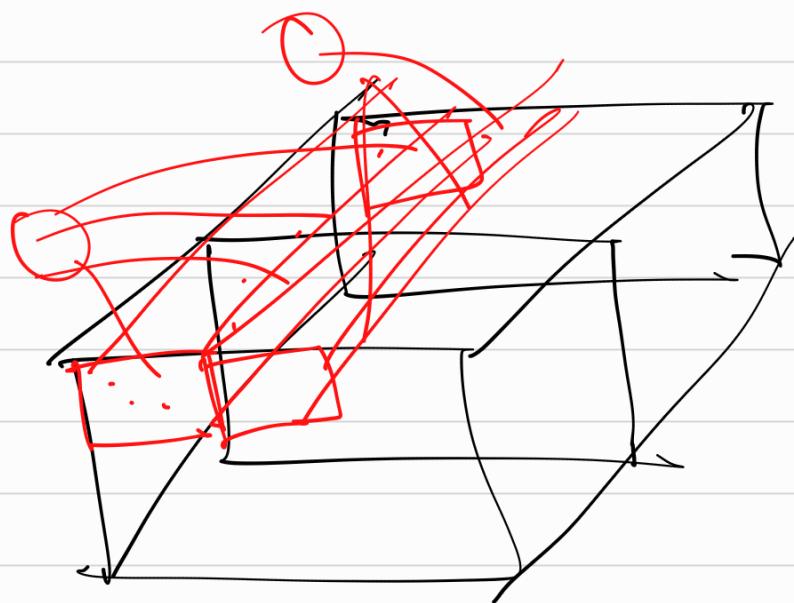
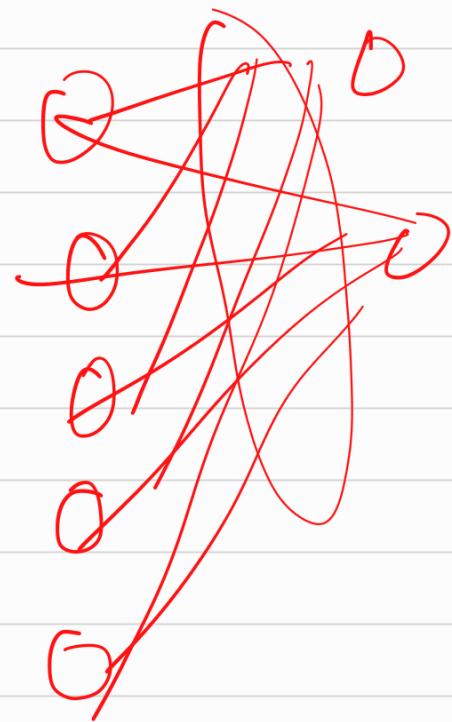






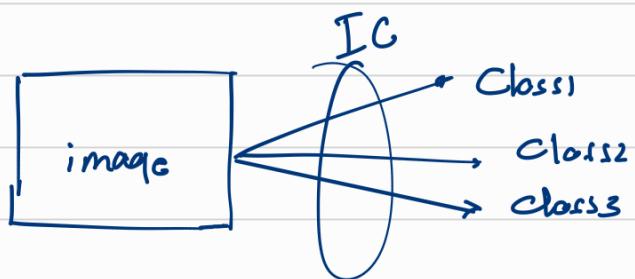
Conv2D



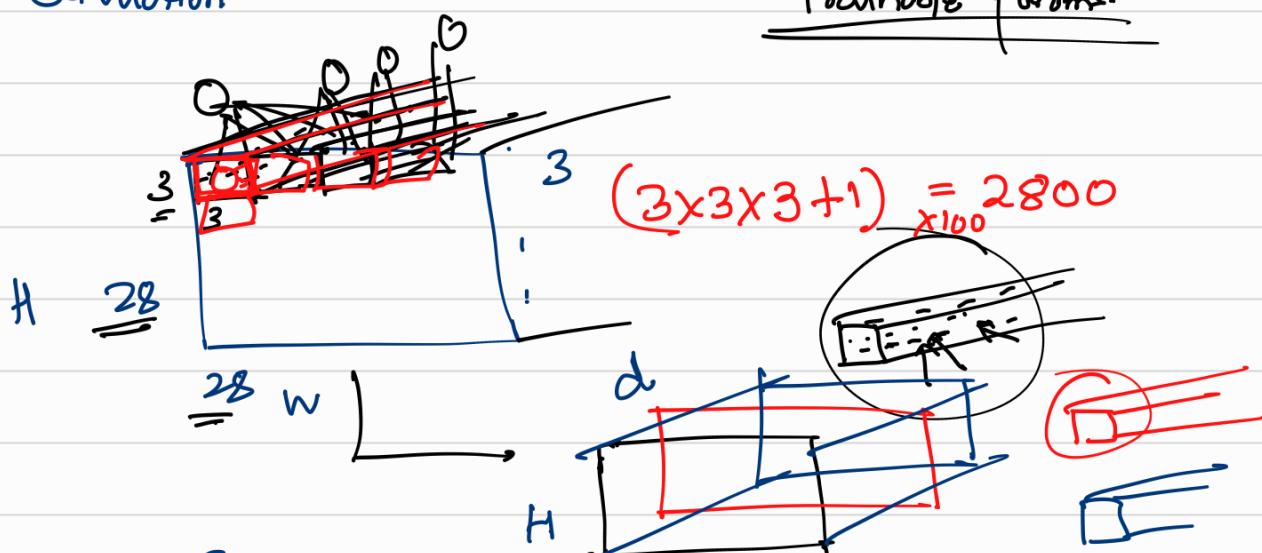


model.add (Conv2D(3x3, 100, s=1, p='valid')
(int))

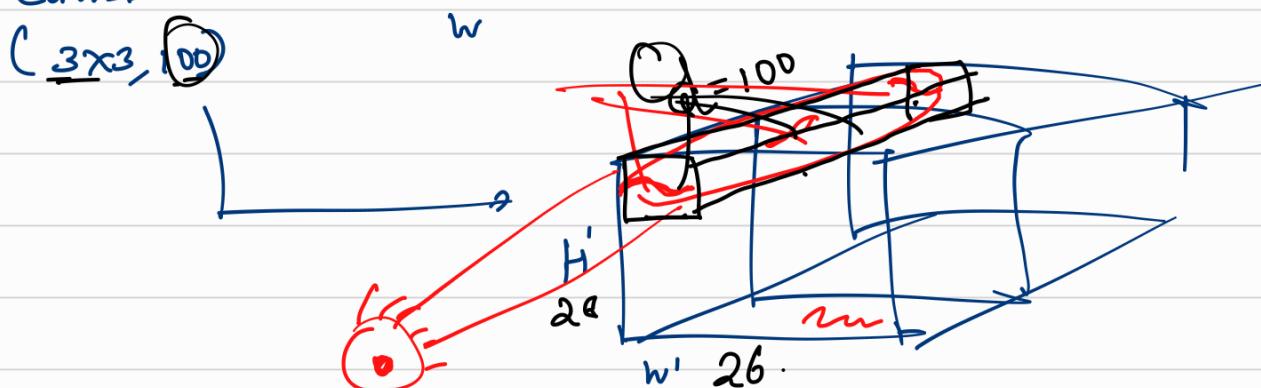
Convolution Neural Network



Convolution



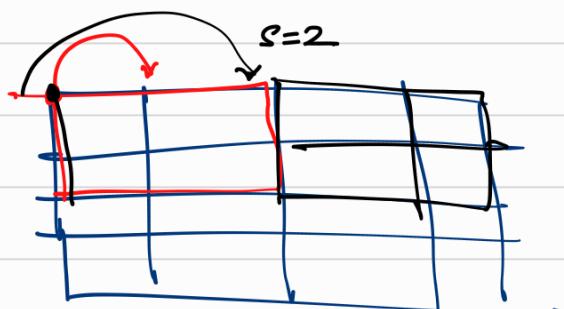
Conv2D



$$H' = \frac{H-f}{S} + 1$$

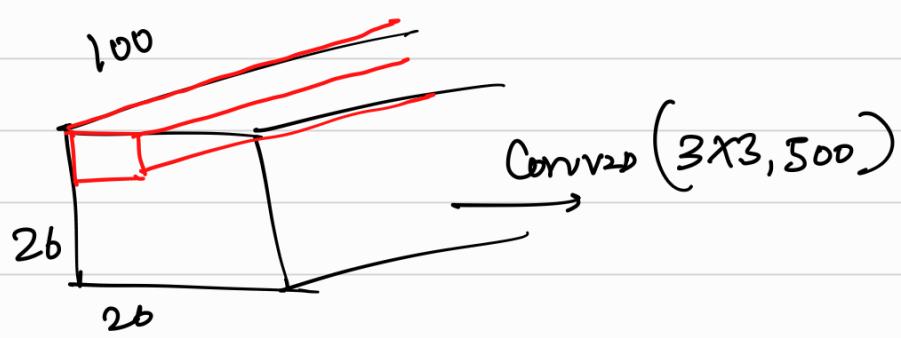
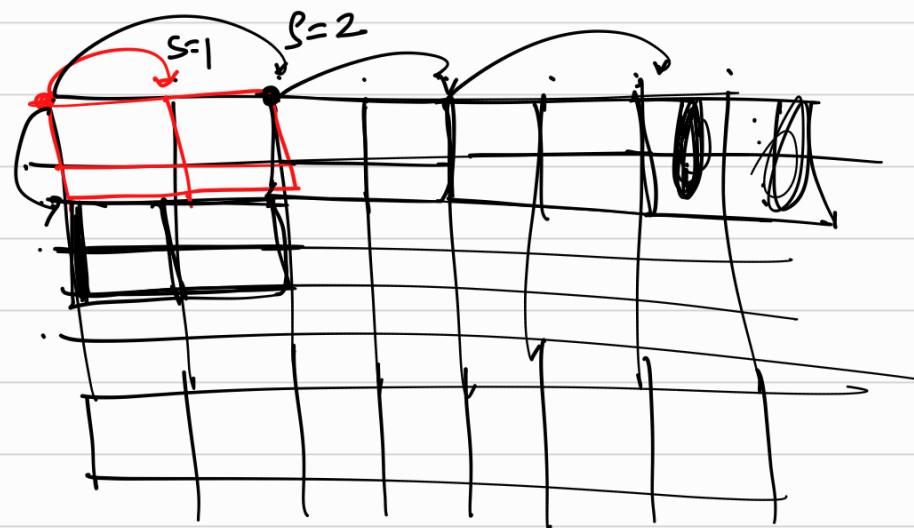
$$(S=1)$$

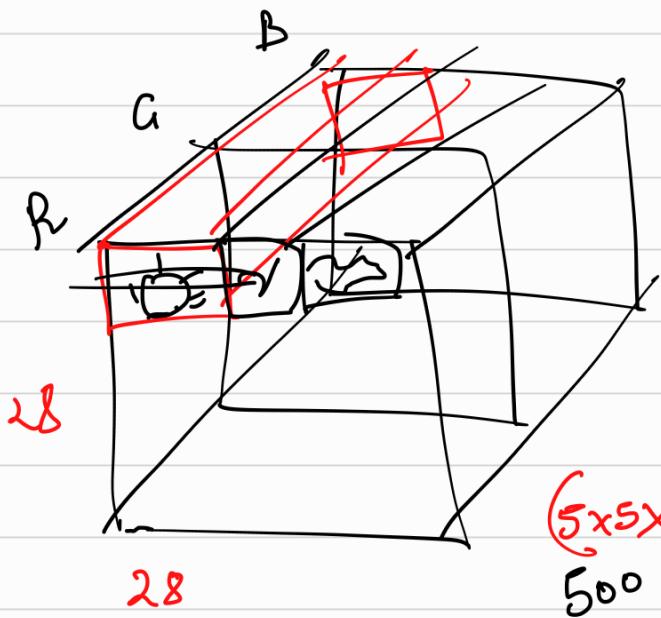
$$W' = \frac{W-f}{S} + 1$$



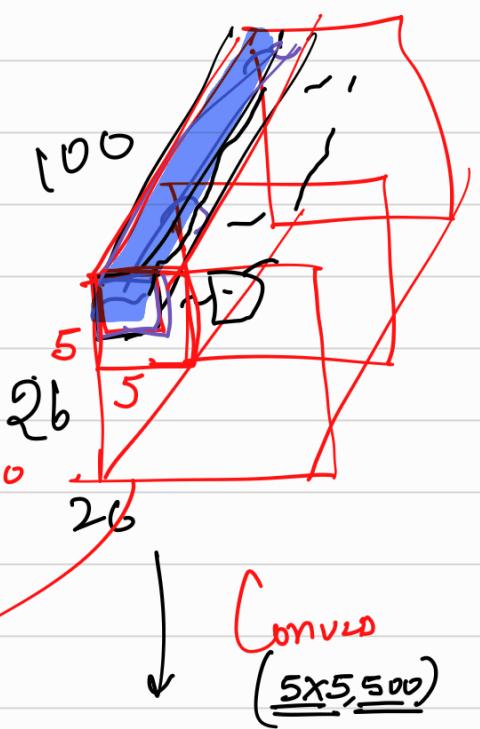
$$H' = \frac{28-3}{2} + 1$$

$$= 14 \quad 14$$

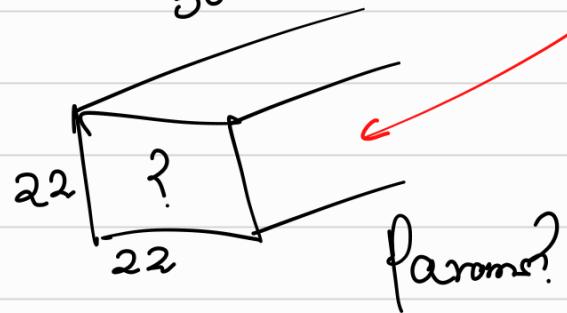




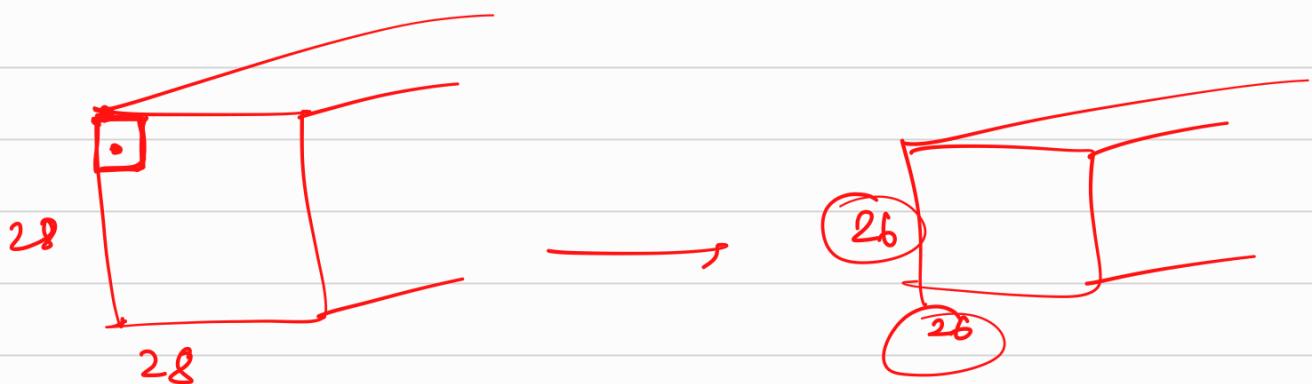
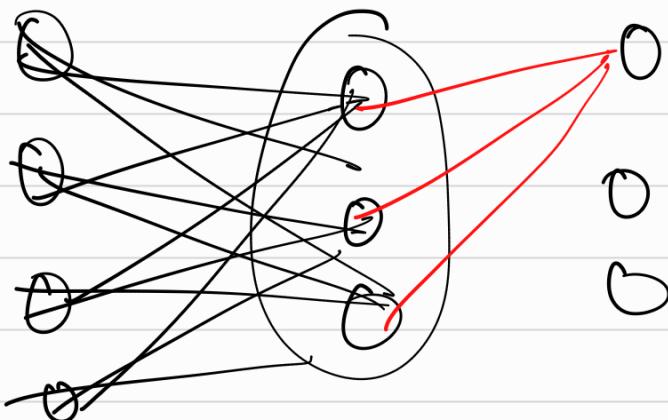
$$(5 \times 5 \times 100 + 1) \times 500$$

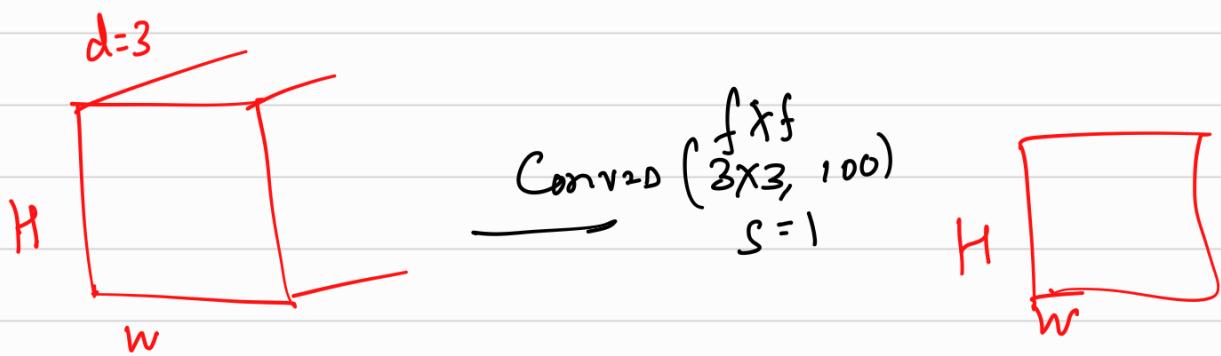
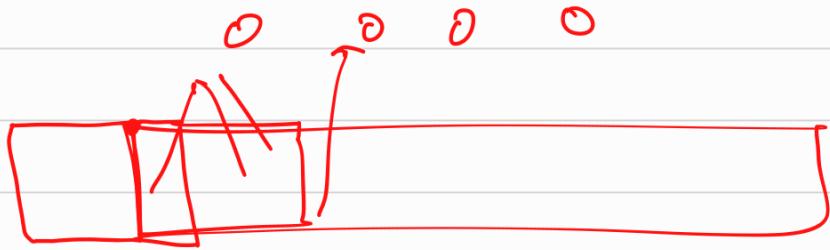


$$\frac{26-5}{1} + 1 = 22$$



param?





$$H' = \frac{H - f + 1}{s}$$

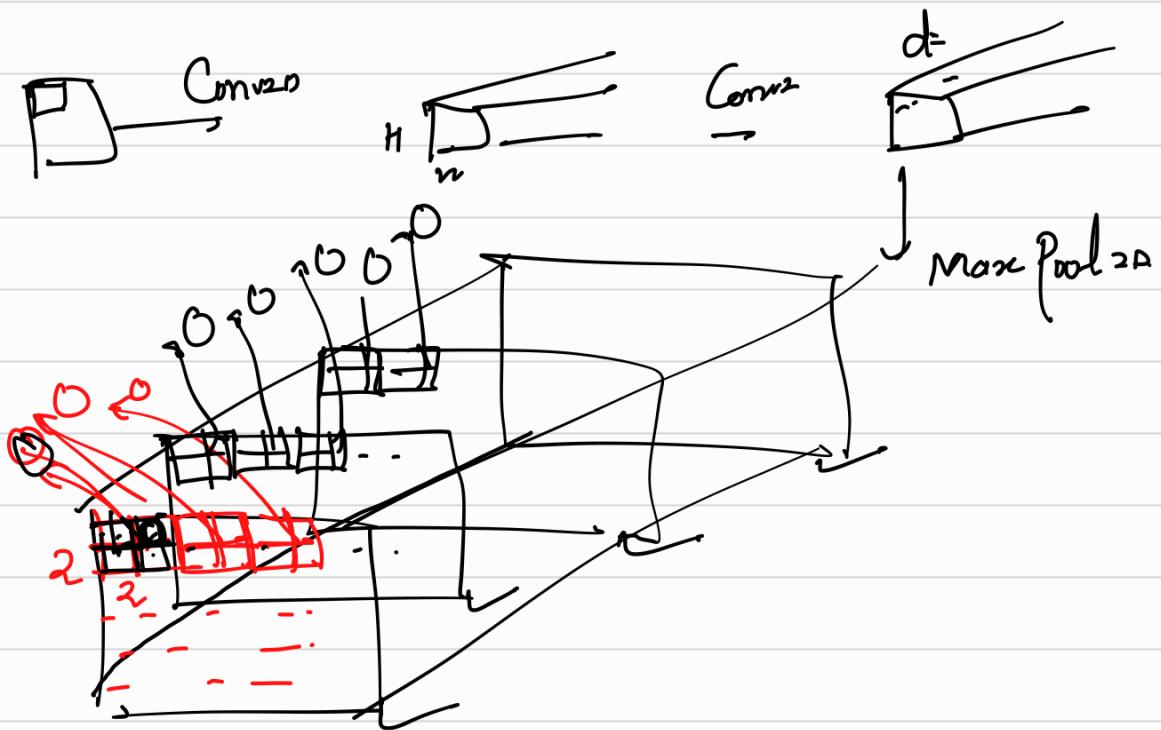
$$W' = \frac{W - f + 1}{s}$$

With Padding

$$\underline{H'} = \frac{\underline{H} + 2\underline{P} - \underline{f}}{s} + 1$$

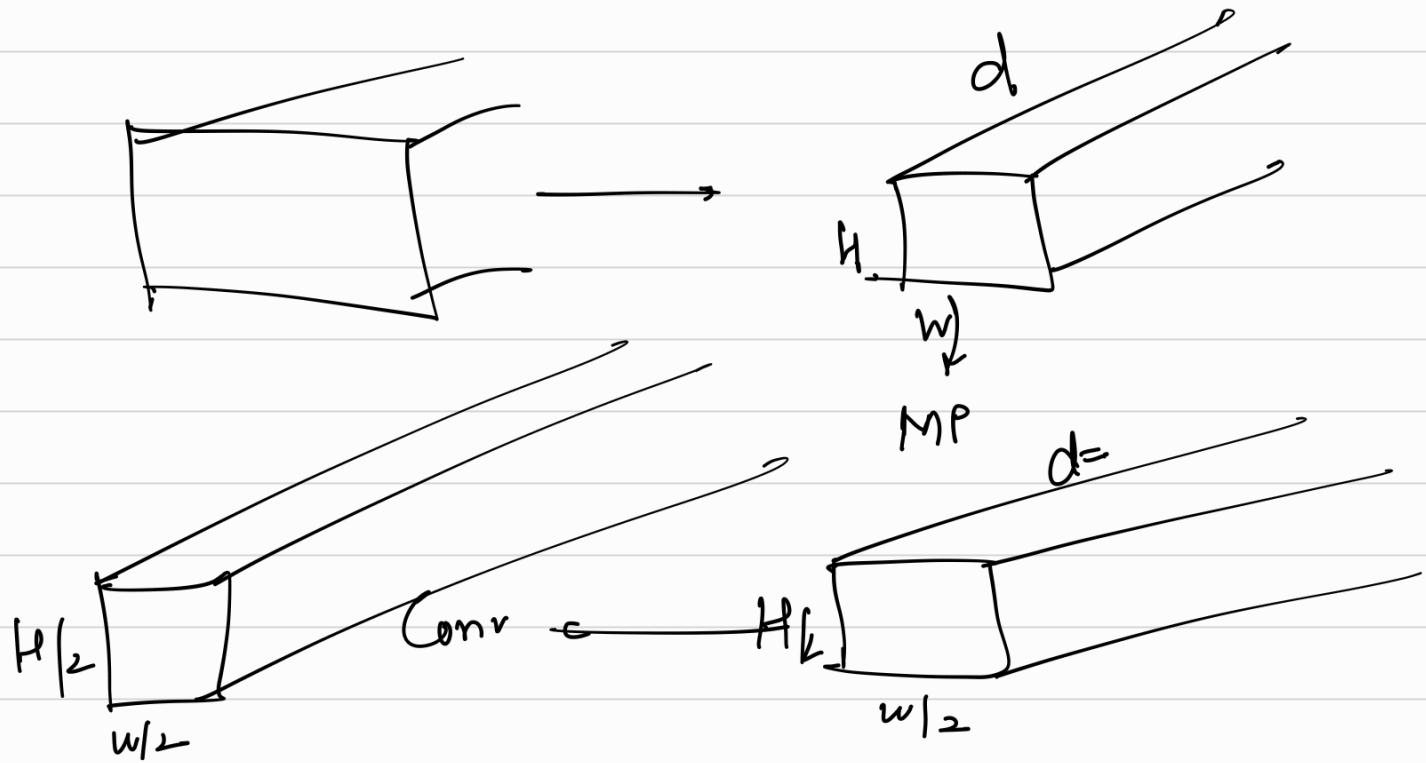
$$W' = \frac{W + 2P - f + 1}{s}$$

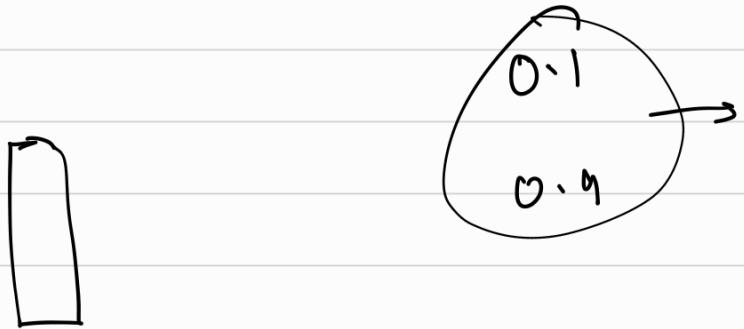
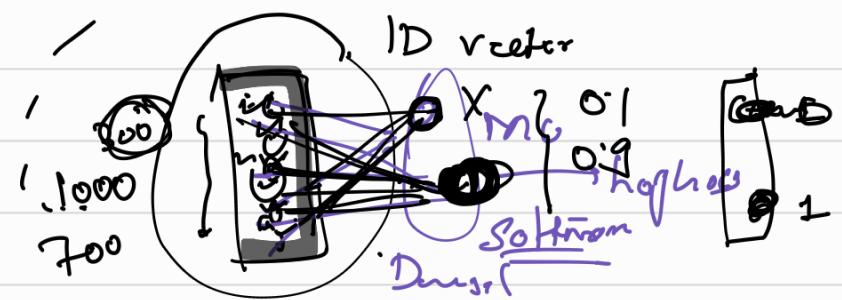
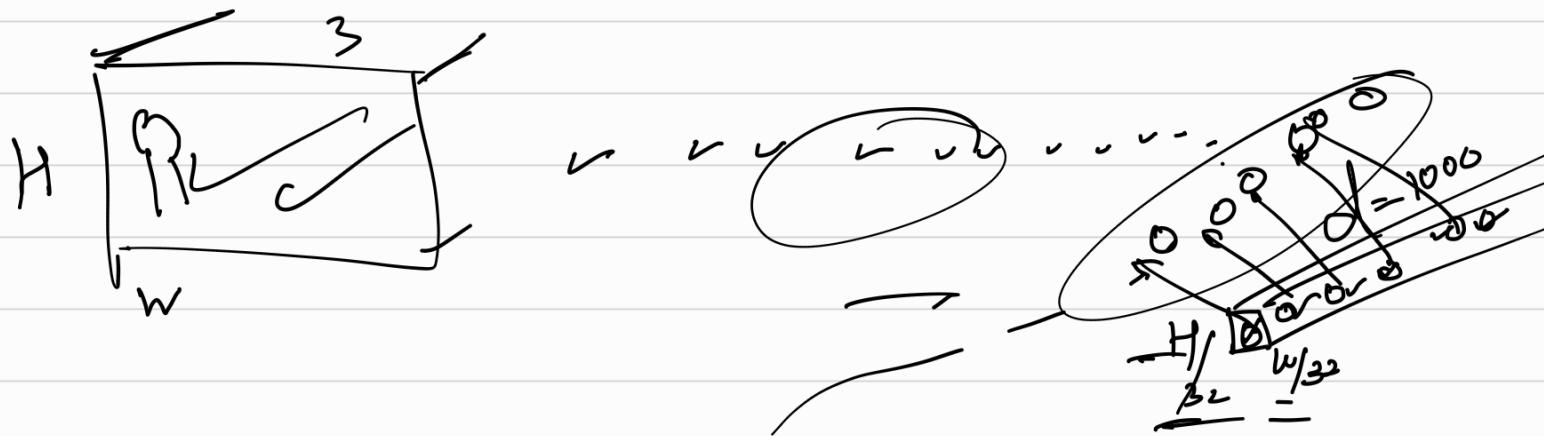
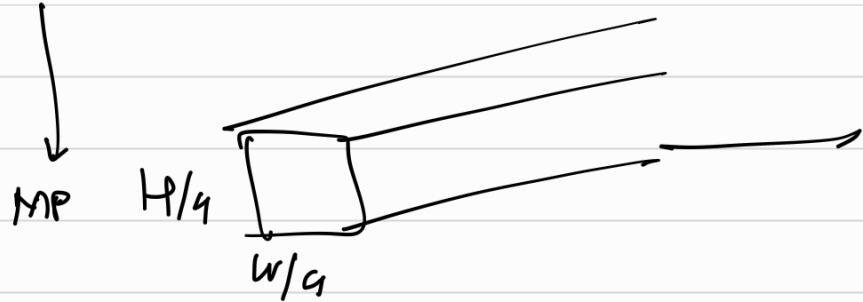
Conv2D (3x3, $s=1$, Padding = "Same")

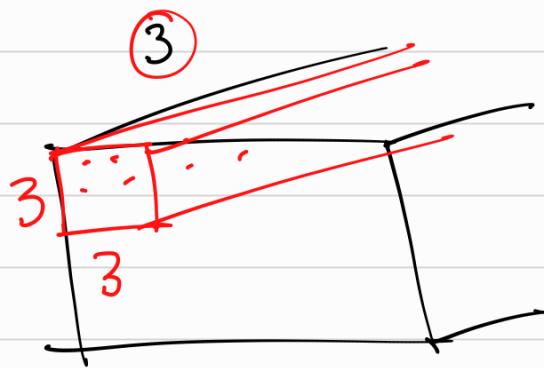


$$H' = \frac{H - f + 1}{s}$$

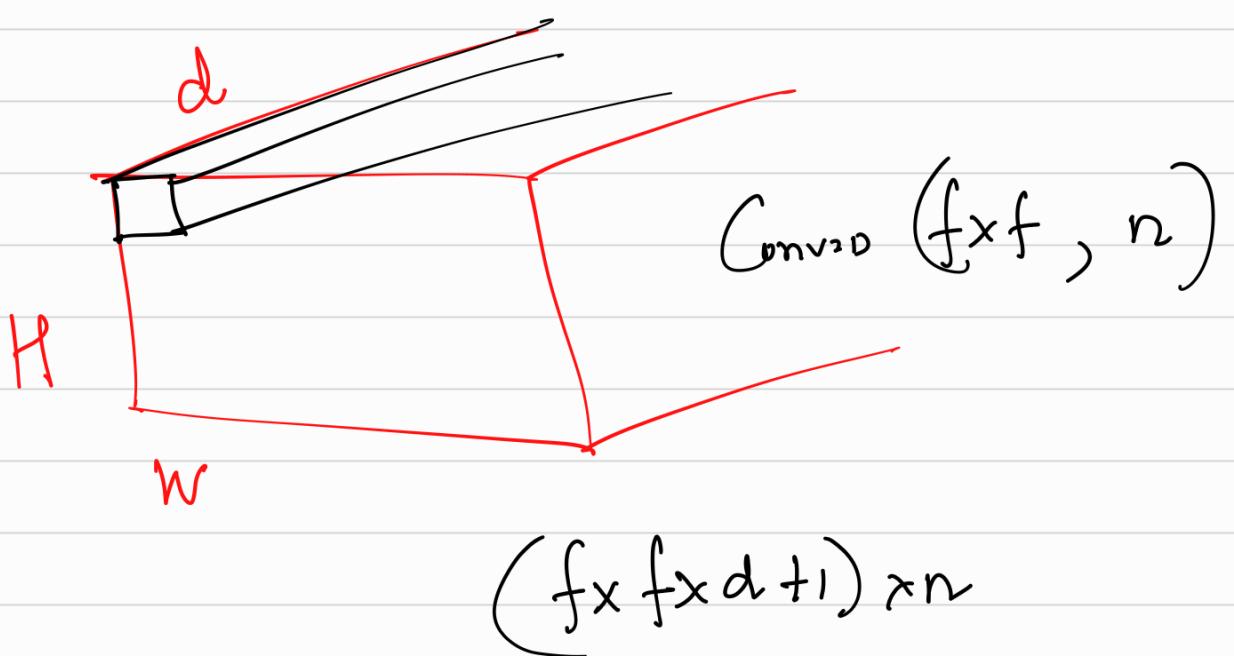
$$H' = \frac{H - 2f + 1}{s} + 1$$







$$(7+1) \times 32$$

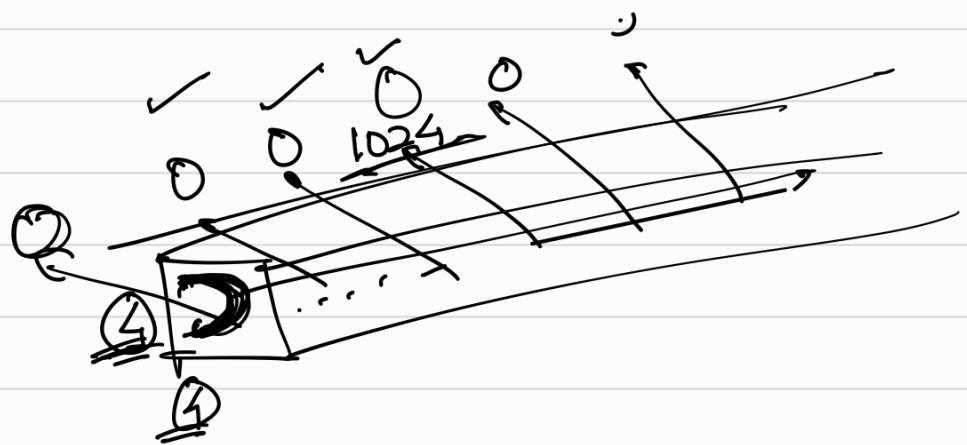
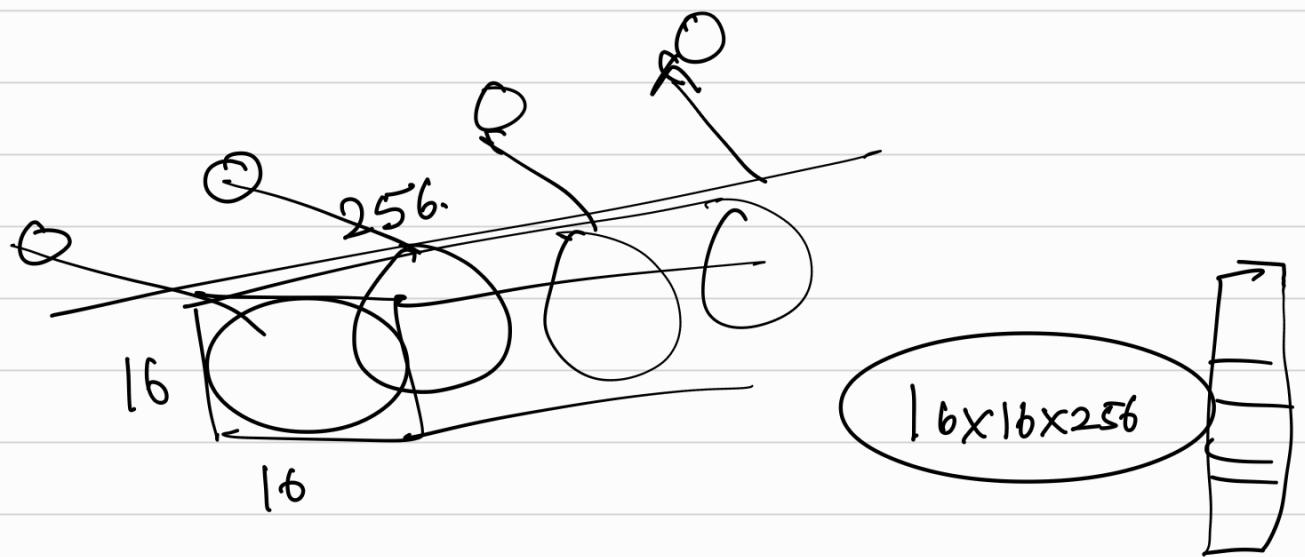


$$\frac{H-f}{s} + 1$$

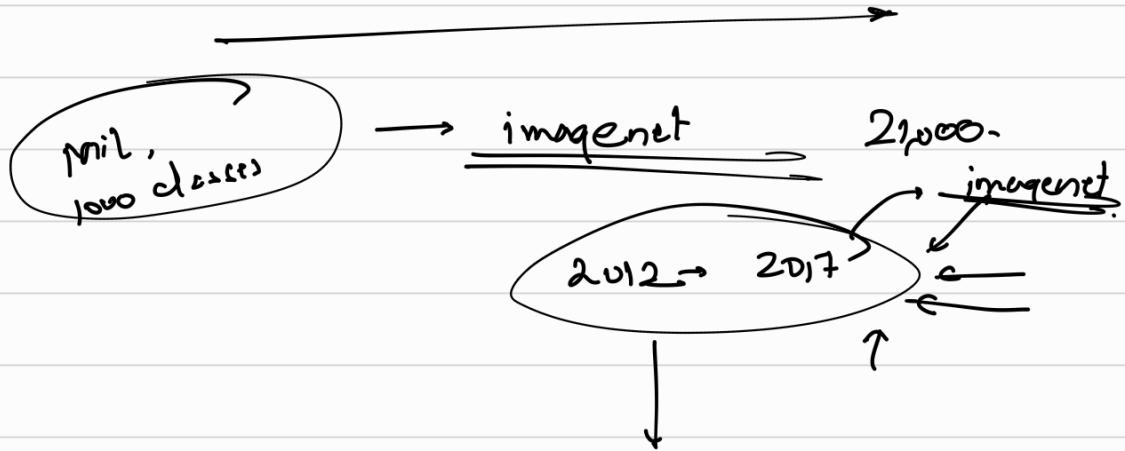
$$= \frac{256-2}{2} + 1$$

$$= 127 + 1$$

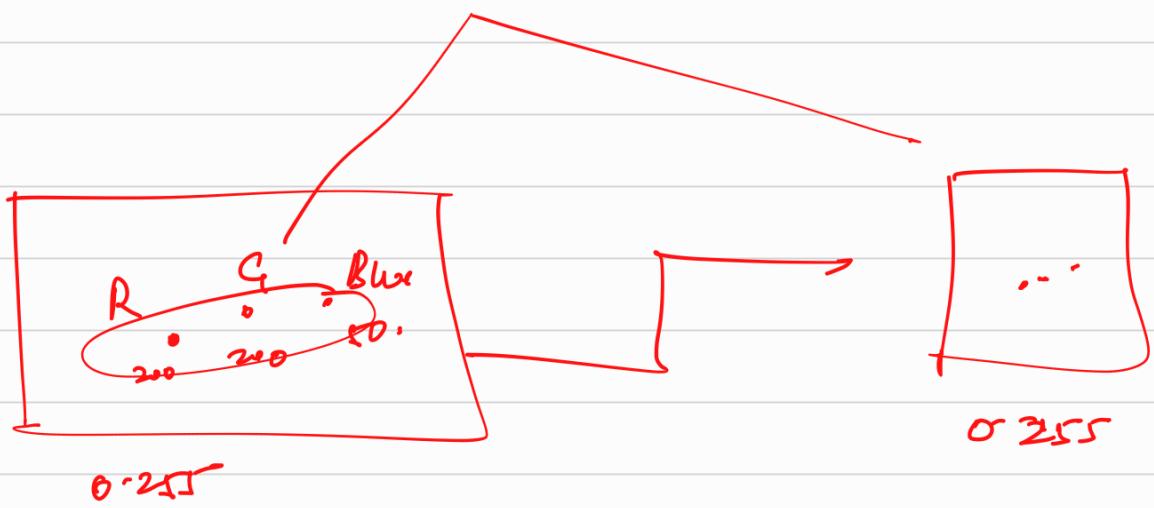
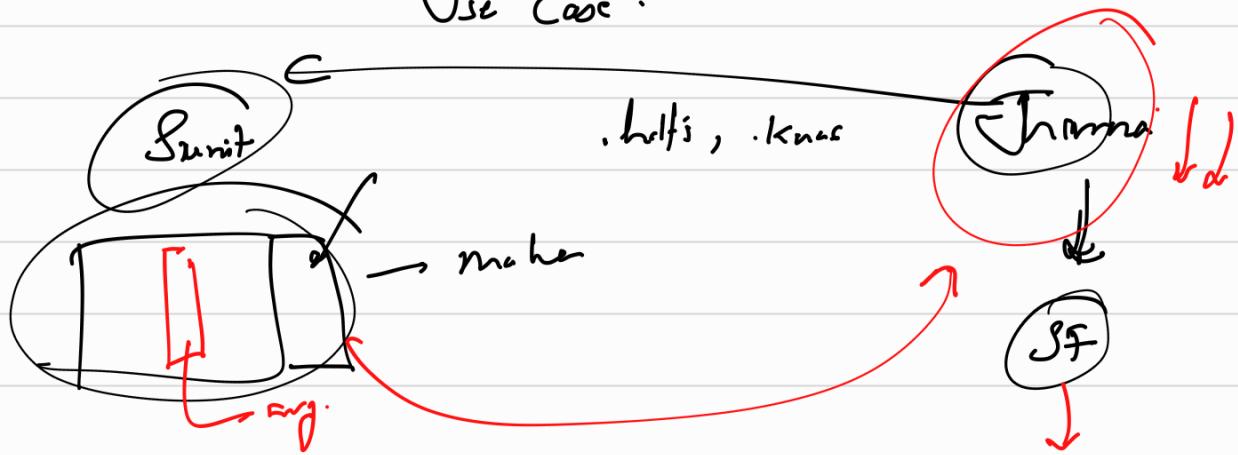
$$= 128$$

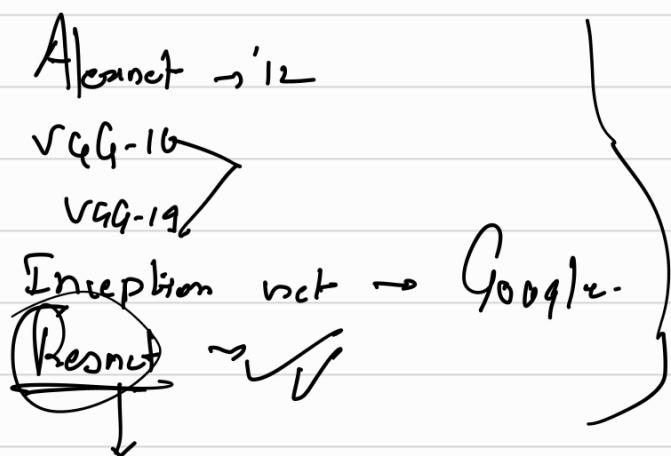
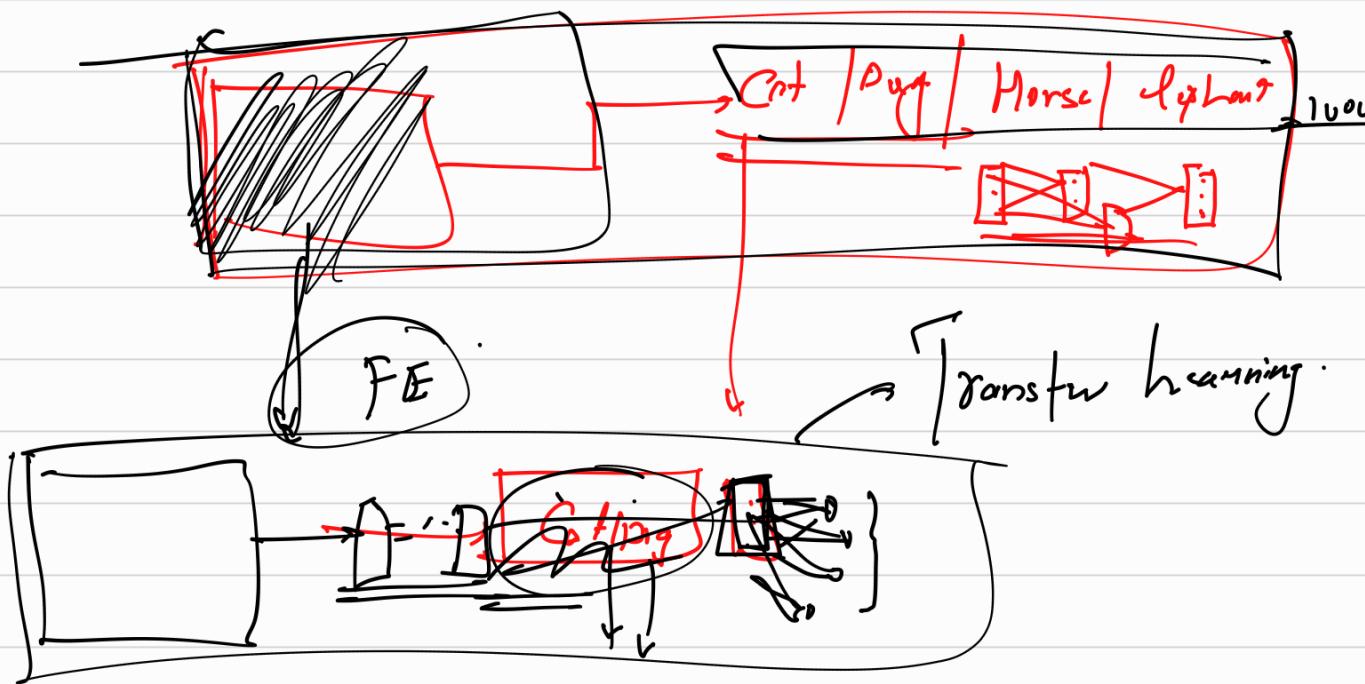


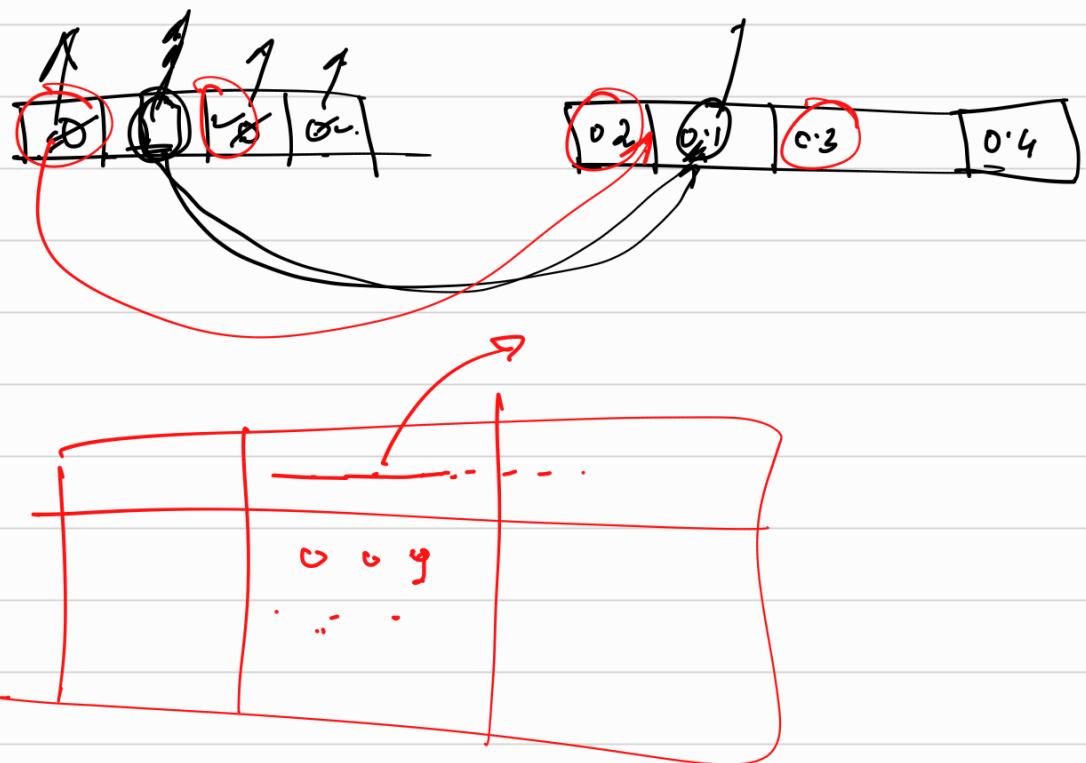
Transfer learning



Use Case :







\rightarrow TL (Imagenet dataset).

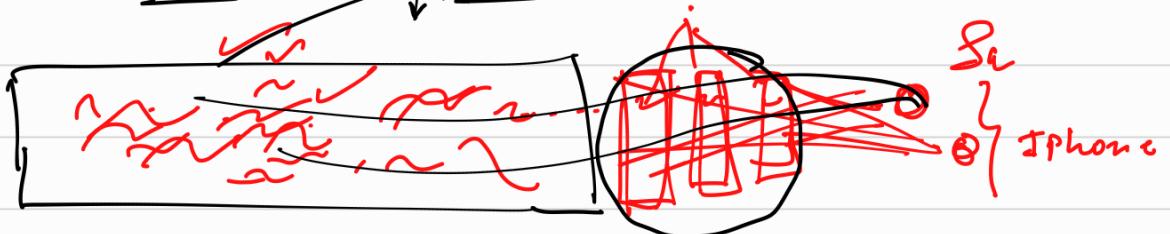


TL

F/C

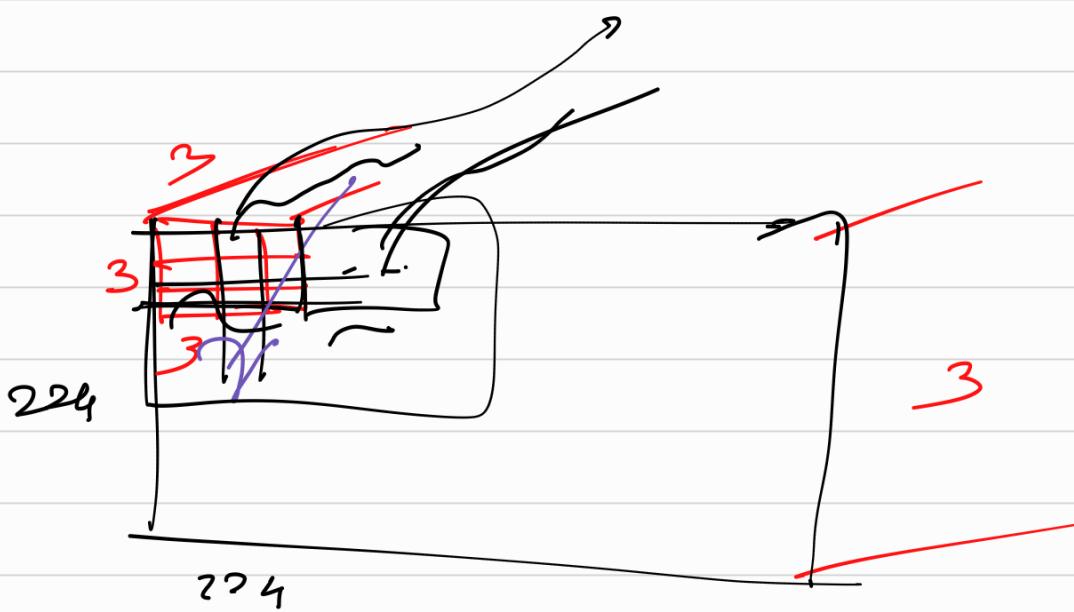
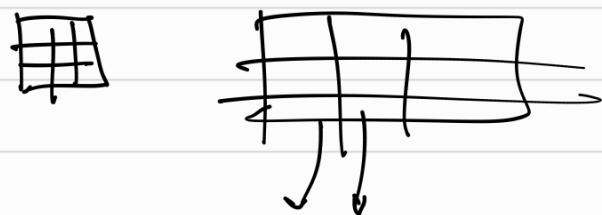
Frozen:

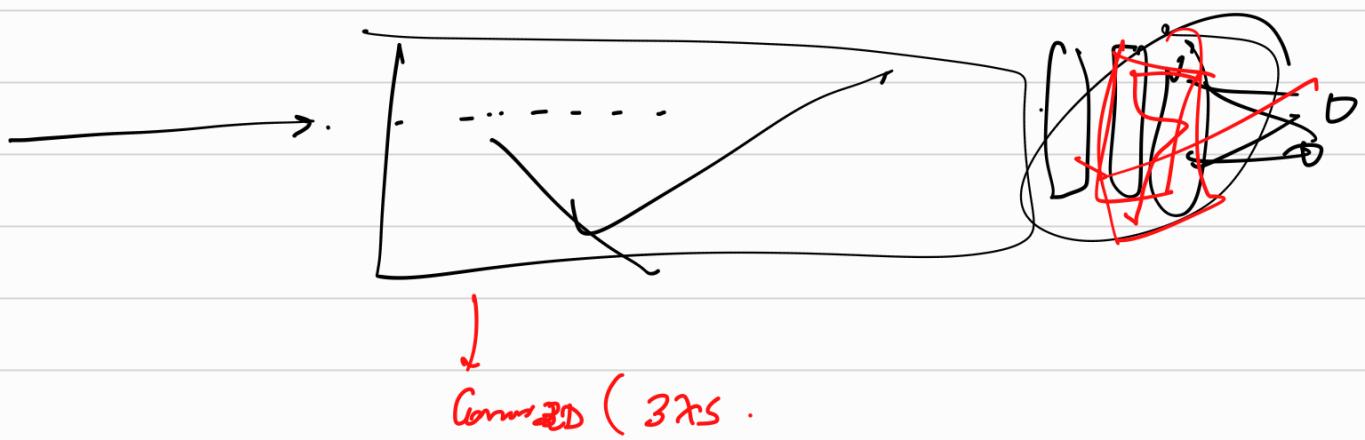
image



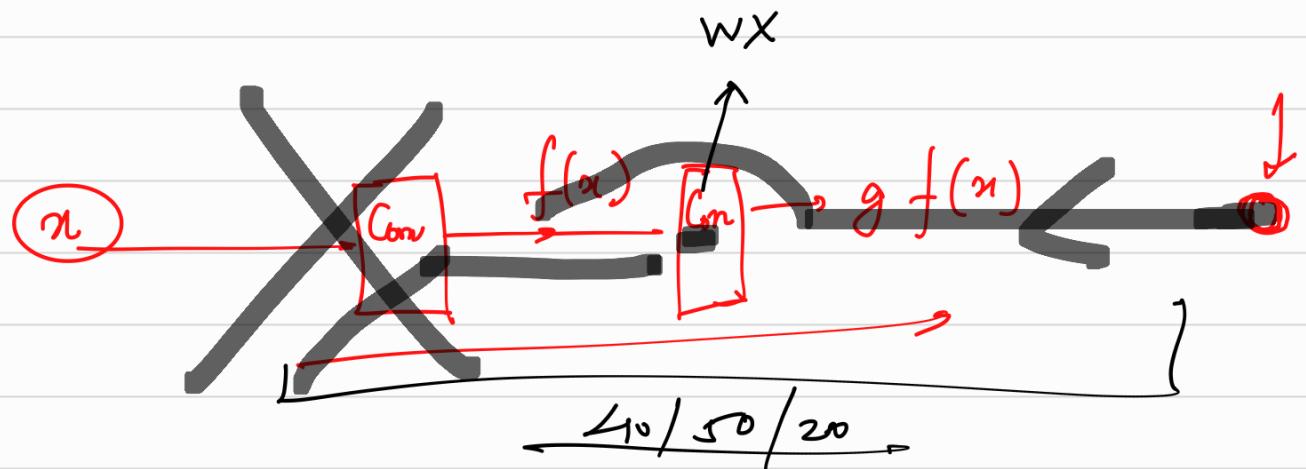
VGG-16 VGG-19

→ Conv conv MP 6x6 mp Conv Conv Conv MP
Conv Conv Conv MP → FC FC FC

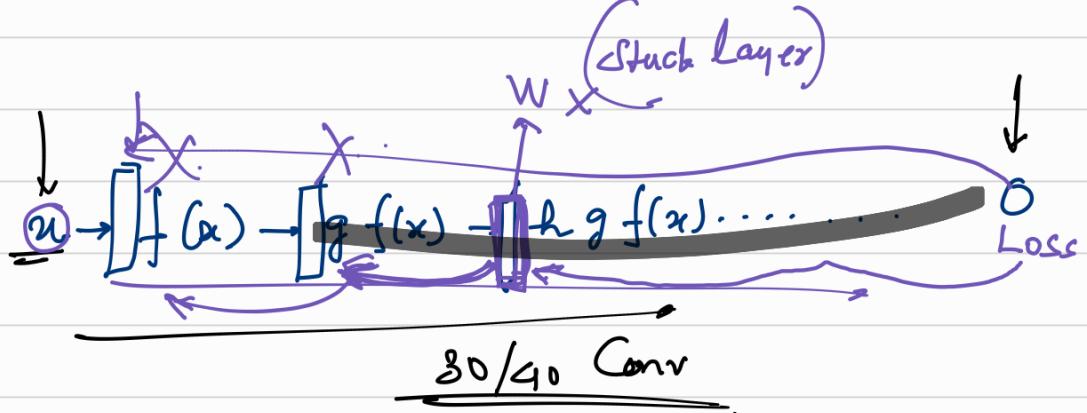




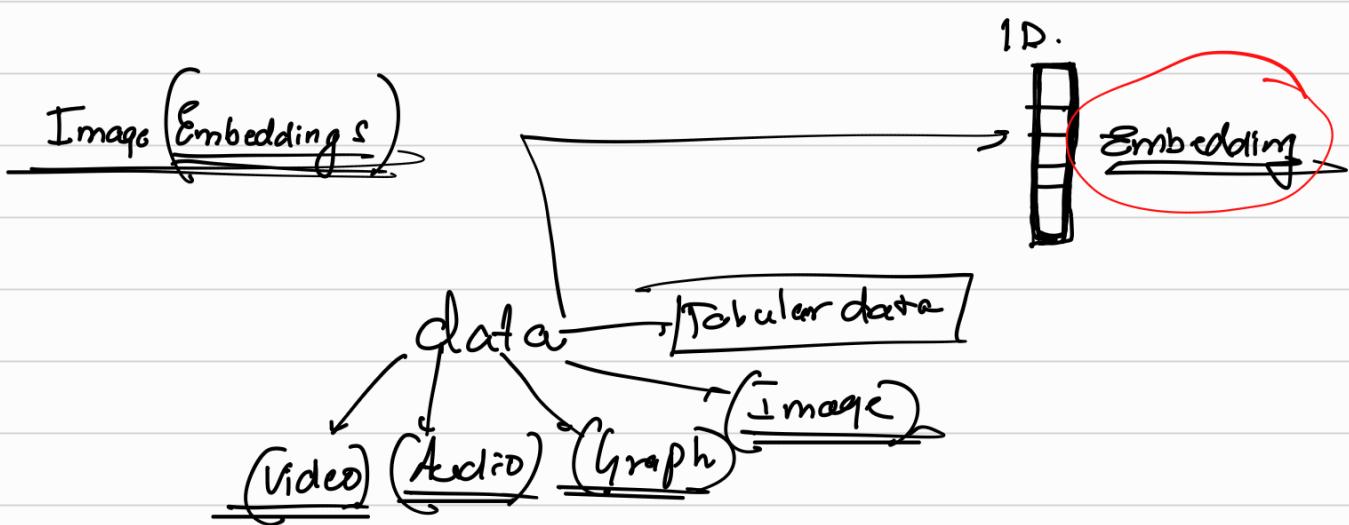
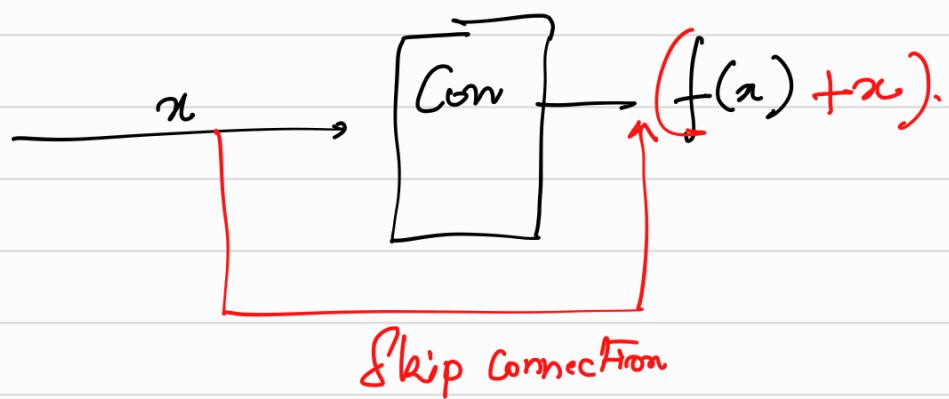
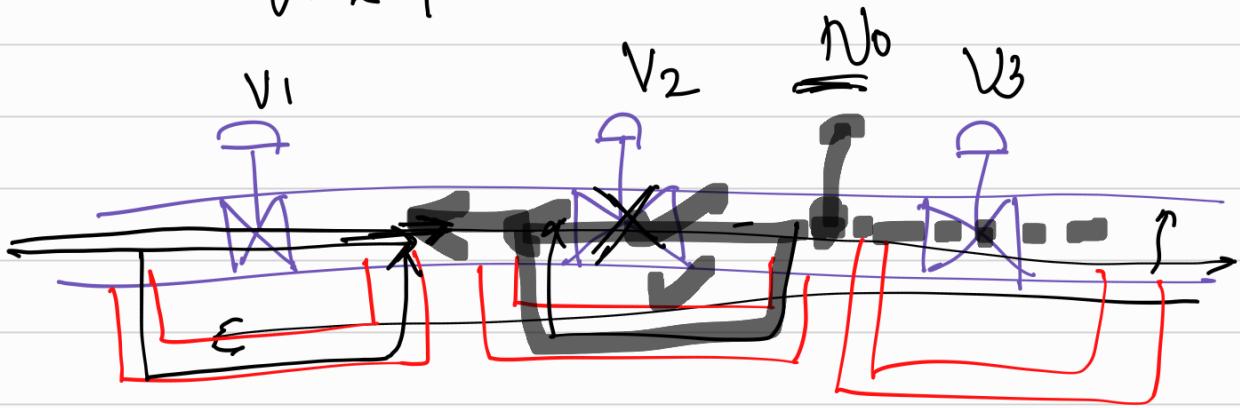
Resnet



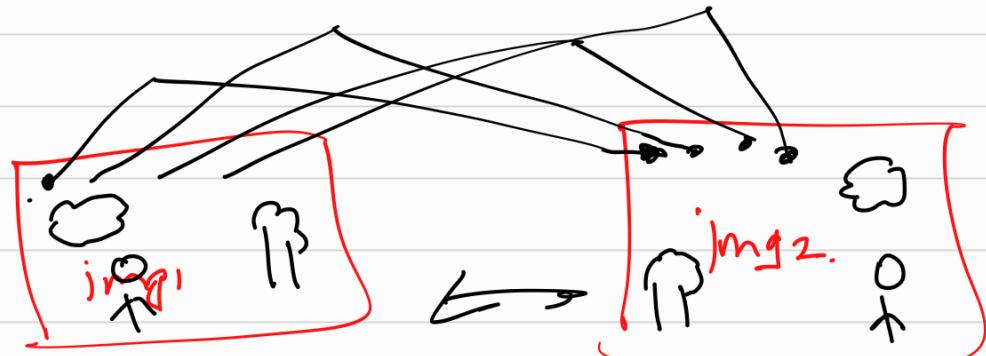
Resnet



$V = \text{Layers}$

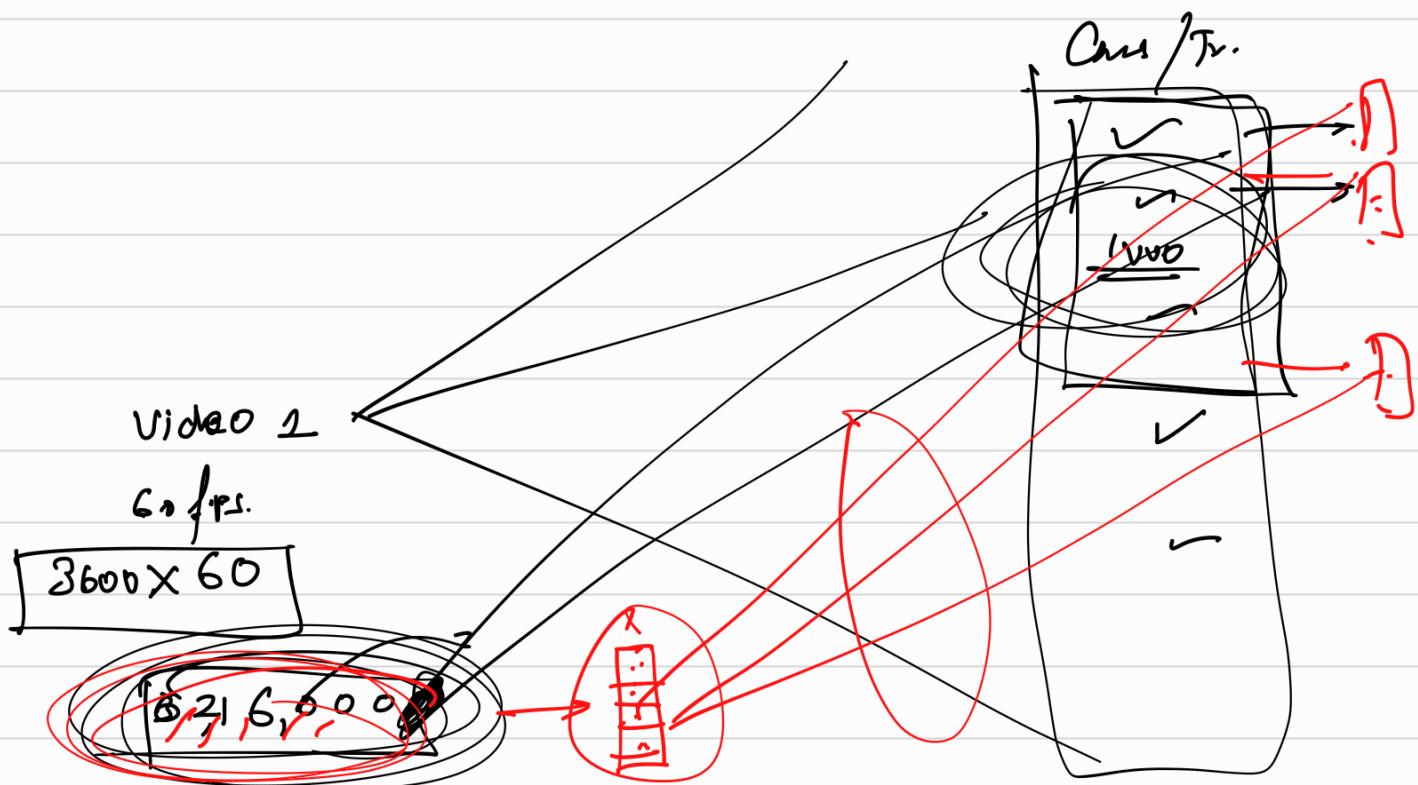


Youtube → Video → 500 MB → 20%



EU

Fribourg Norm.

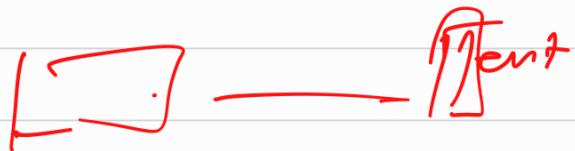


BMW
USA

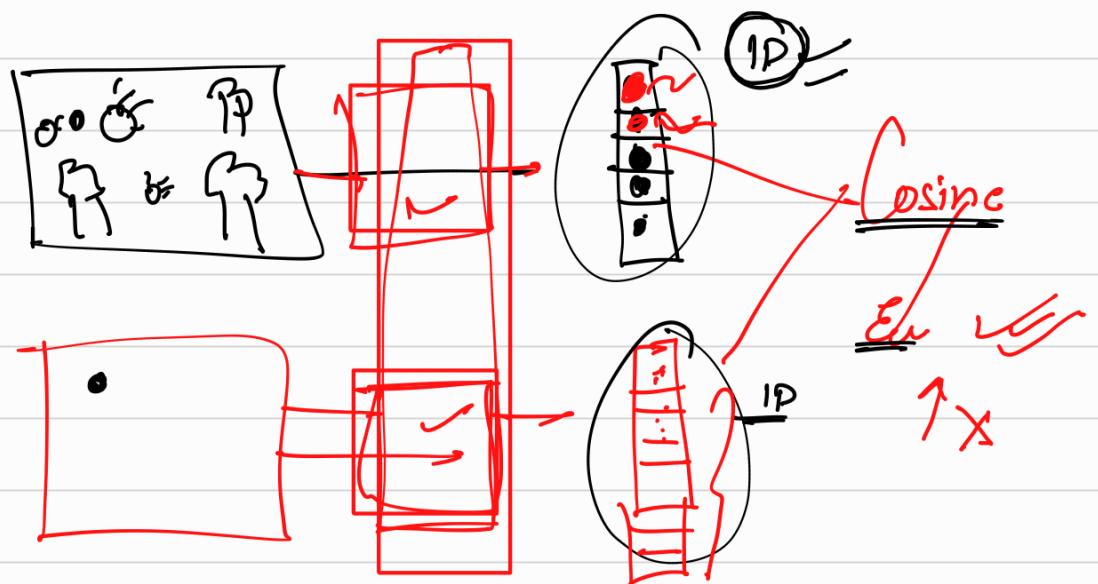
Review

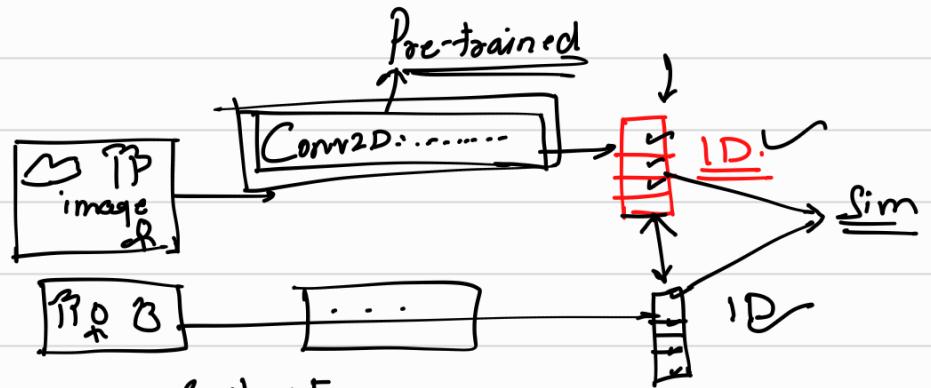
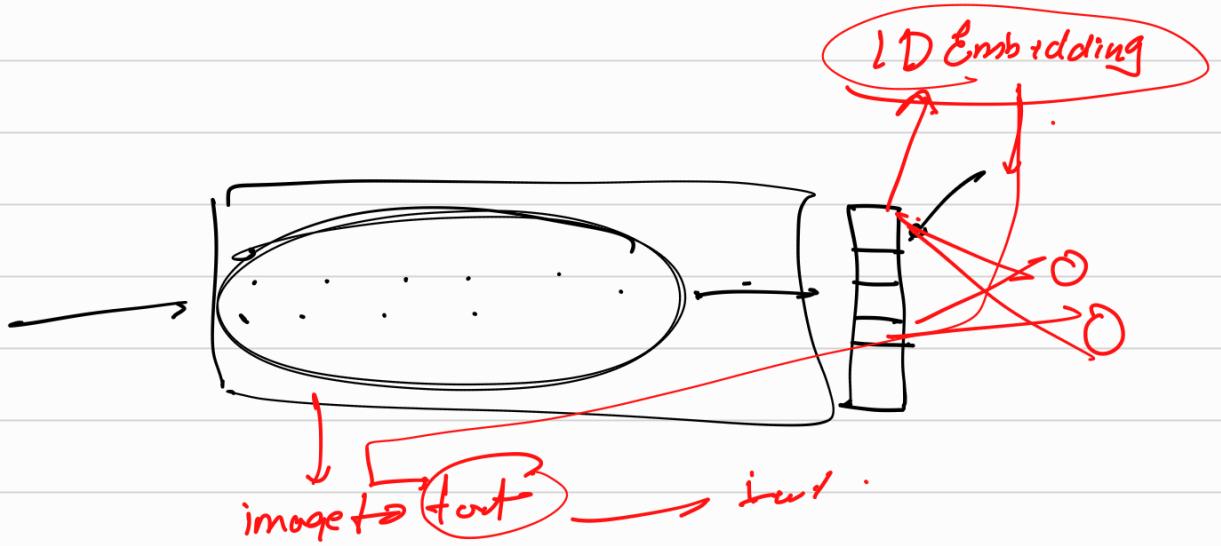
Muscle Review
Brace India

Tent → image



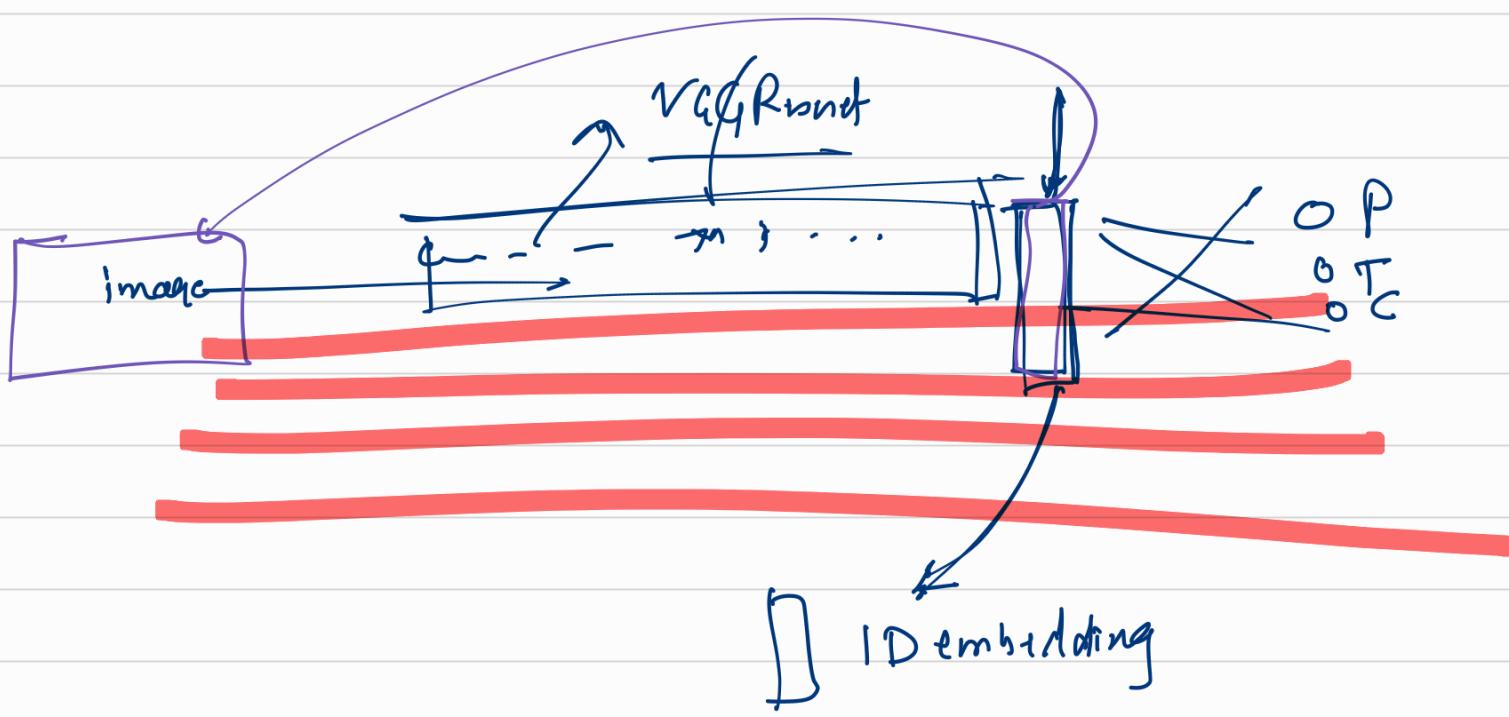
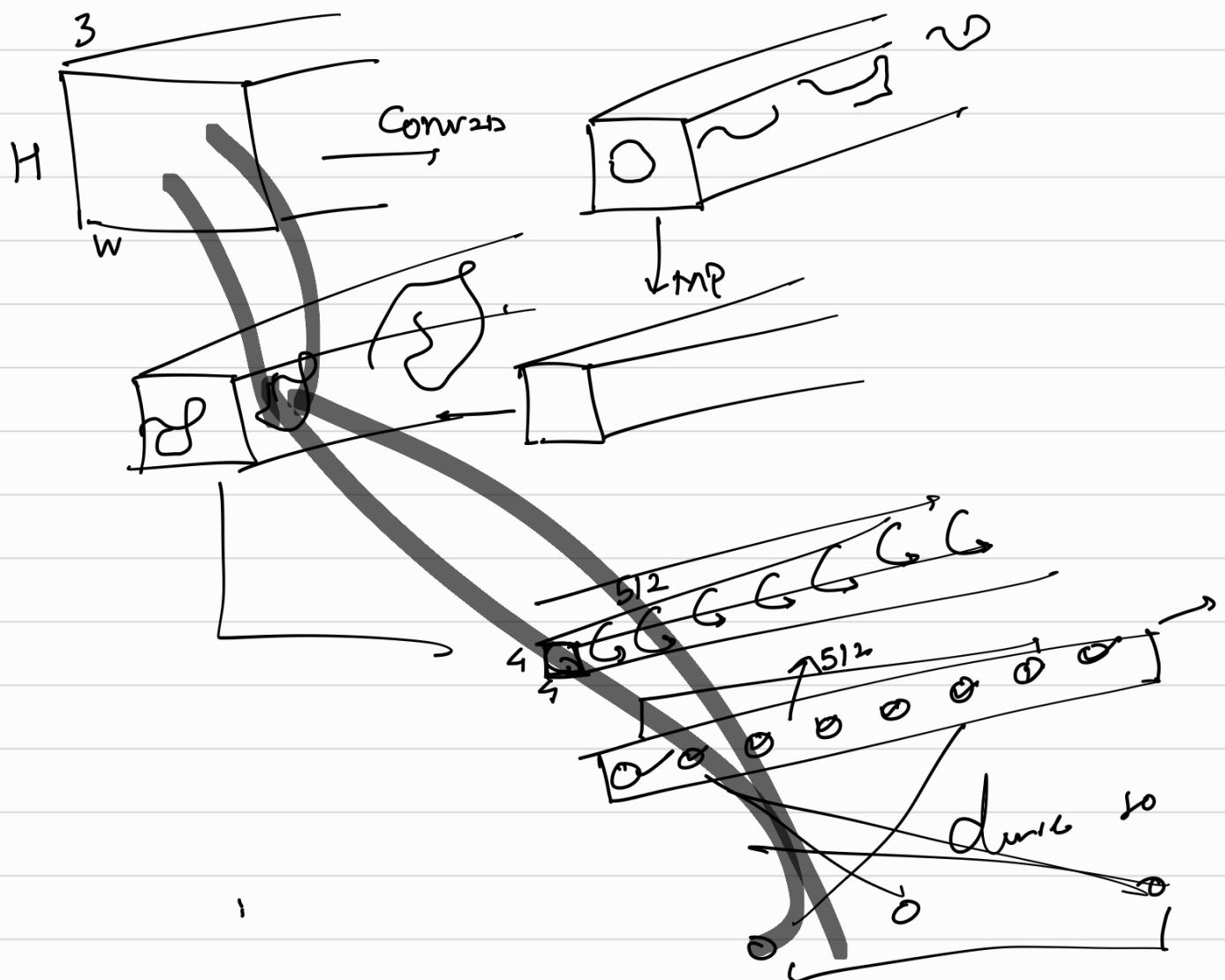
Tent → video.



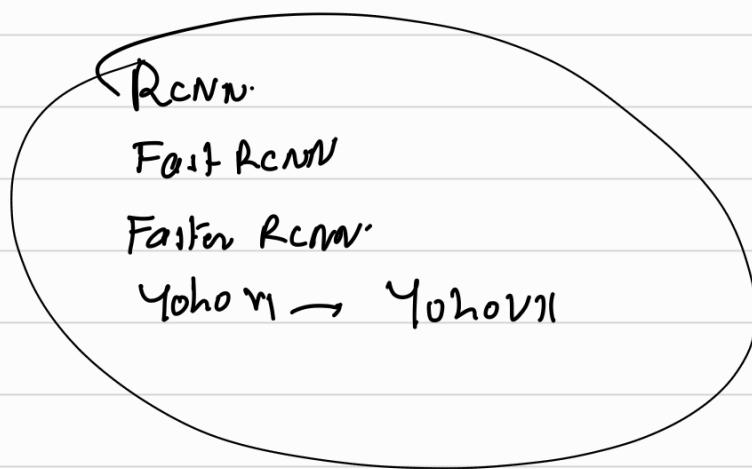
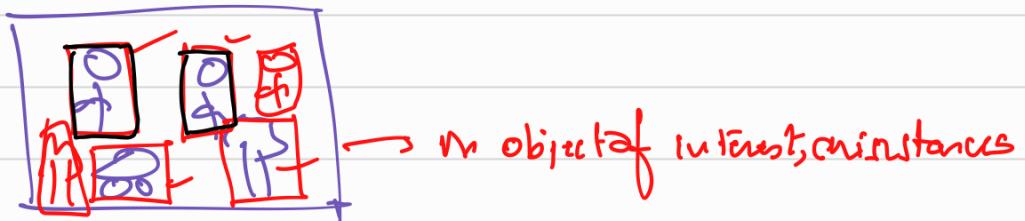
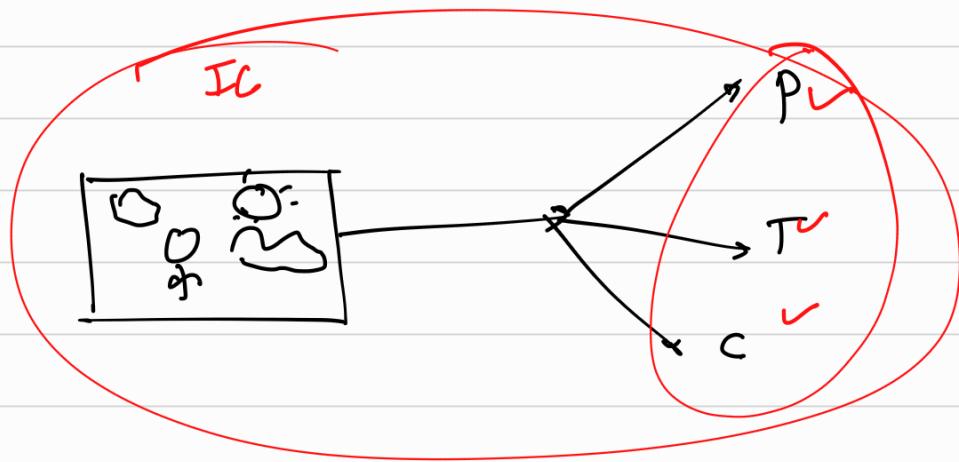


① Calculating image similarity.

- Recommendation systems ✓
- Clustering images ✓
- Facial Recognition Systems ✓
 - ↳ Siamese Network -
- image → text
- ↳ text → image



Object detection & localization :

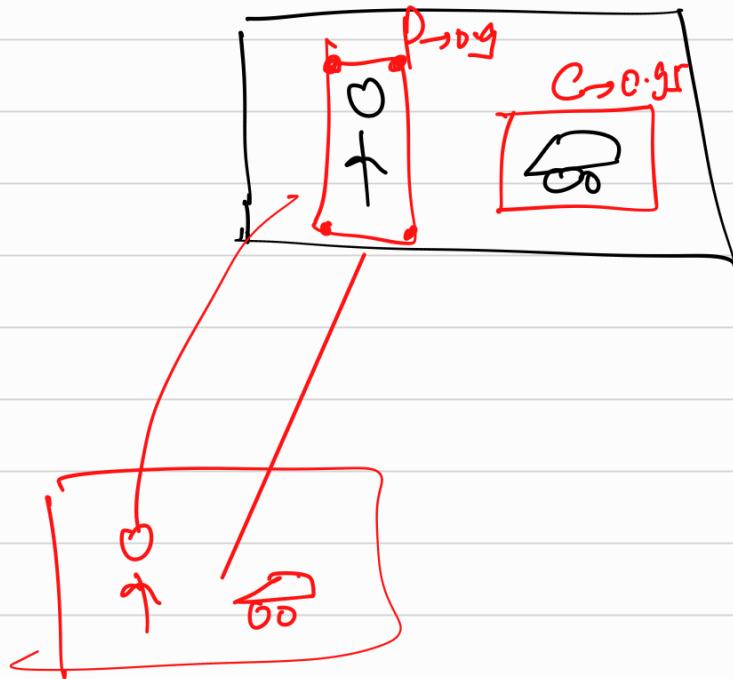


Annotation

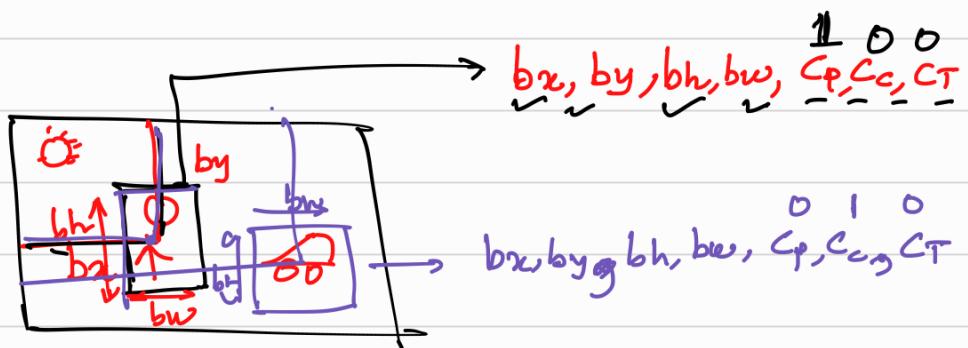
.

,

.



Annotation of bounding boxes to solve for the supervision of obj det and localization problem.



Annotation tools

Robotflow

