Deep learning

$y = f(x) + \varepsilon$

ML

Reg

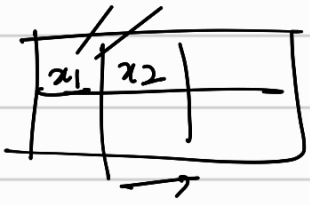
Class.

LR, LR ← Parametric models

DT, RF, Xgboost ← Non-parametric

Process

$y \sim f(x)$

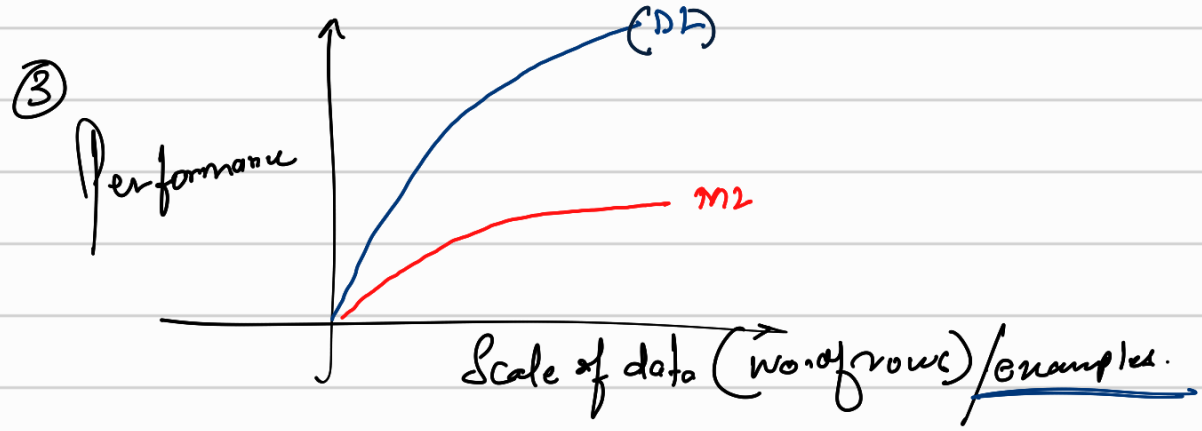


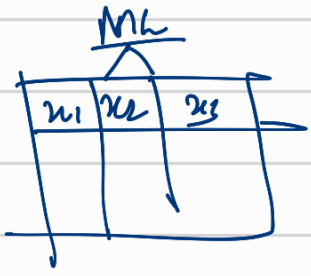
① $x \rightarrow$ (image, video, audio, text, tabular data, graph).

architecture

multimodality

② Tabular data, text, Scan → model → Diagnosis



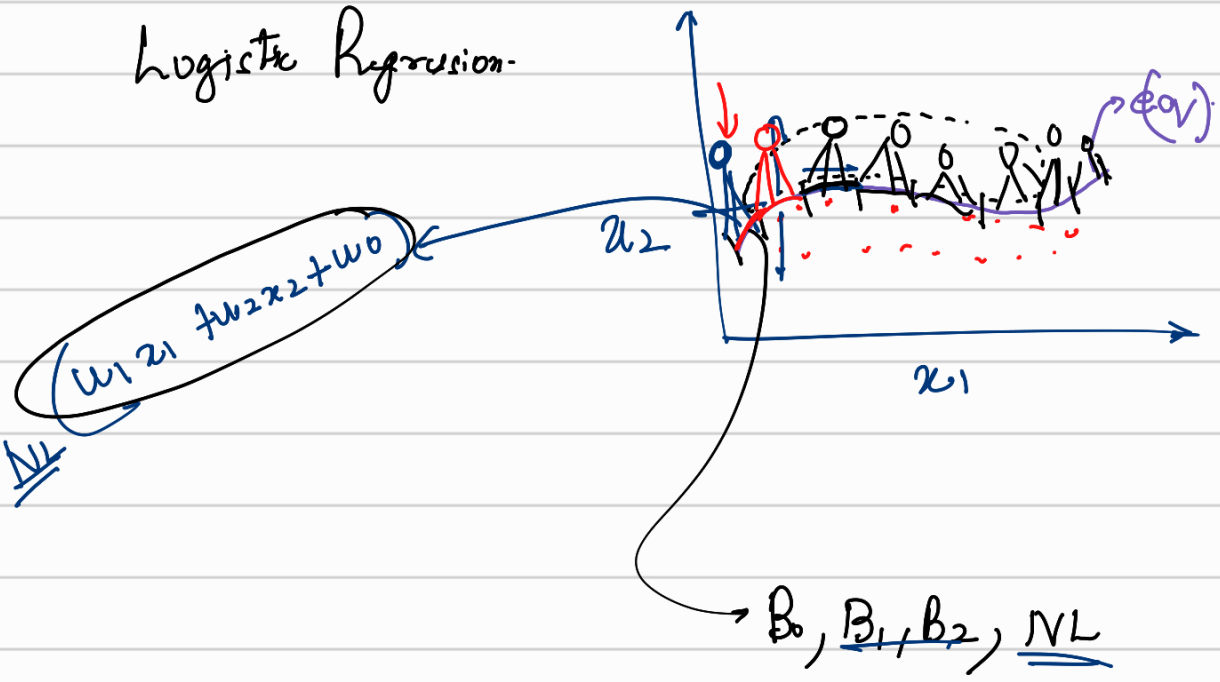


DL → automated feature engineering

DL

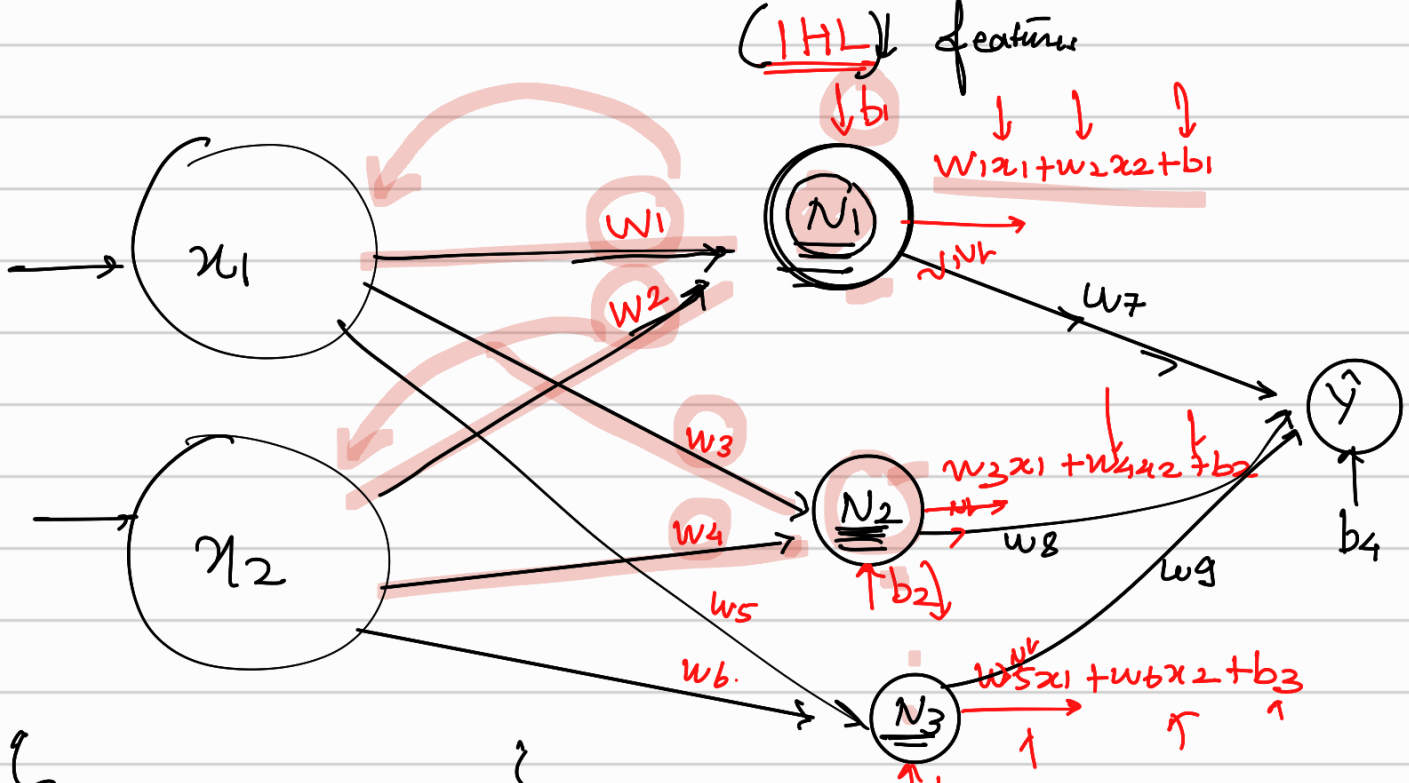
What is a Neuron

Logistic Regression.



B_0, B_1, B_2, ML

x_1	x_2	y



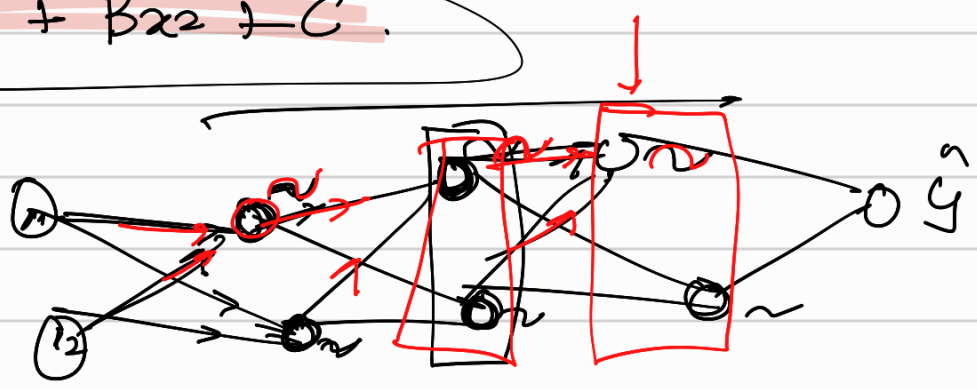
$N_1 = (w_1x_1 + w_2x_2 + b_1)$, $N_2 = (w_3x_1 + w_4x_2 + b_2)$ $N_3 = (w_5x_1 + w_6x_2 + b_3)$

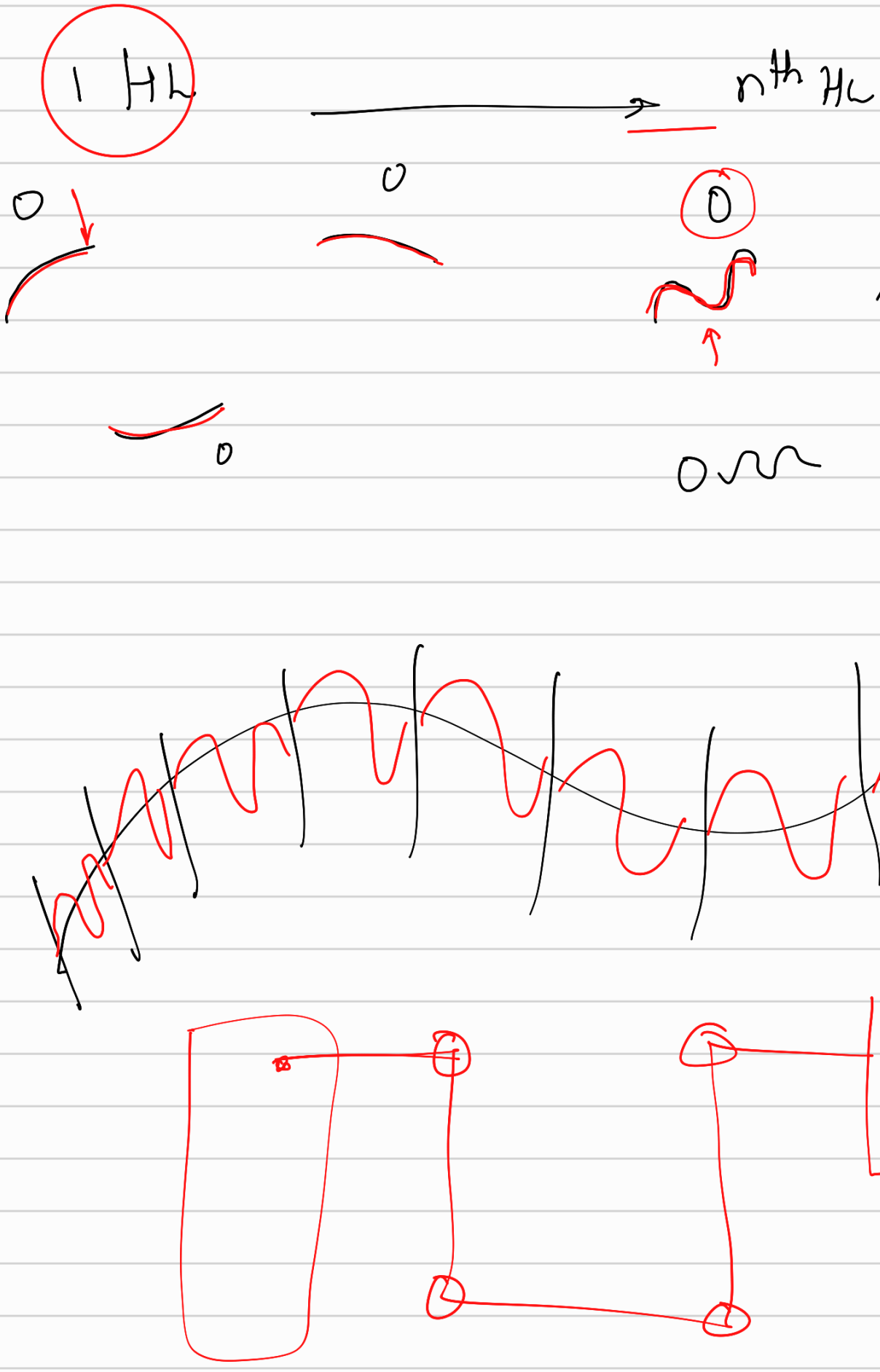
$$\hat{y} = w_7N_1 + w_8N_2 + w_9N_3 + b_4$$

$$\Rightarrow \hat{y} = w_7(w_1x_1 + w_2x_2 + b_1) + w_8(w_3x_1 + w_4x_2 + b_2) + w_9(w_5x_1 + w_6x_2 + b_3) + b_4$$

$$\Rightarrow \hat{y} = x_1(w_7w_1 + w_8w_3 + w_9w_5) + x_2(w_7w_2 + w_8w_4 + w_9w_6) + w_7b_1 + w_8b_2 + w_9b_3 + b_4$$

$$\Rightarrow \hat{y} = Ax_1 + Bx_2 + C$$



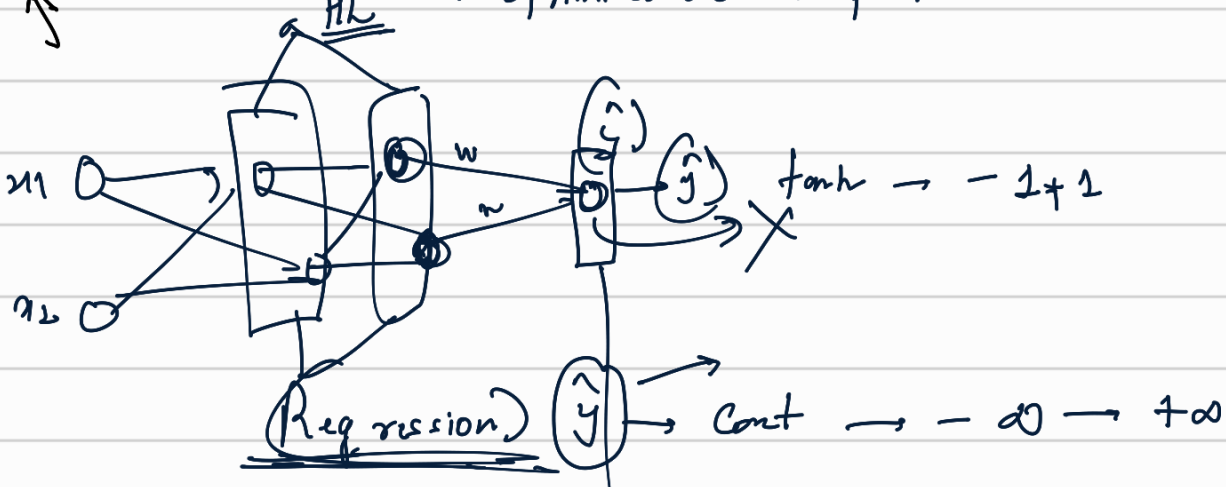

1 HL
nth HL

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

Random

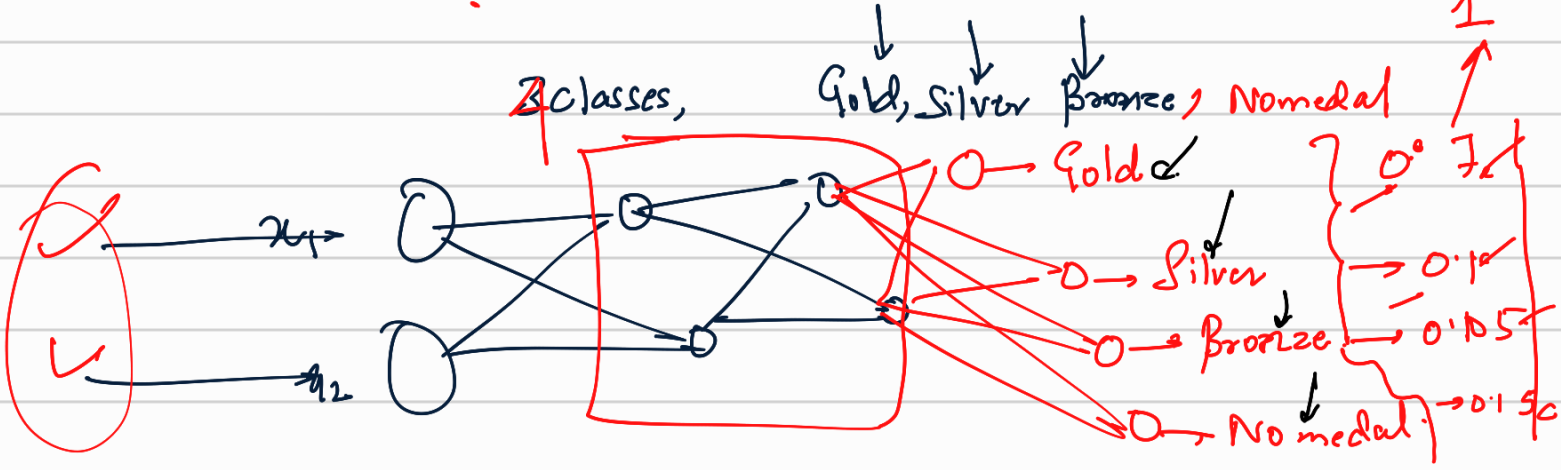
1) Define the neural network architecture

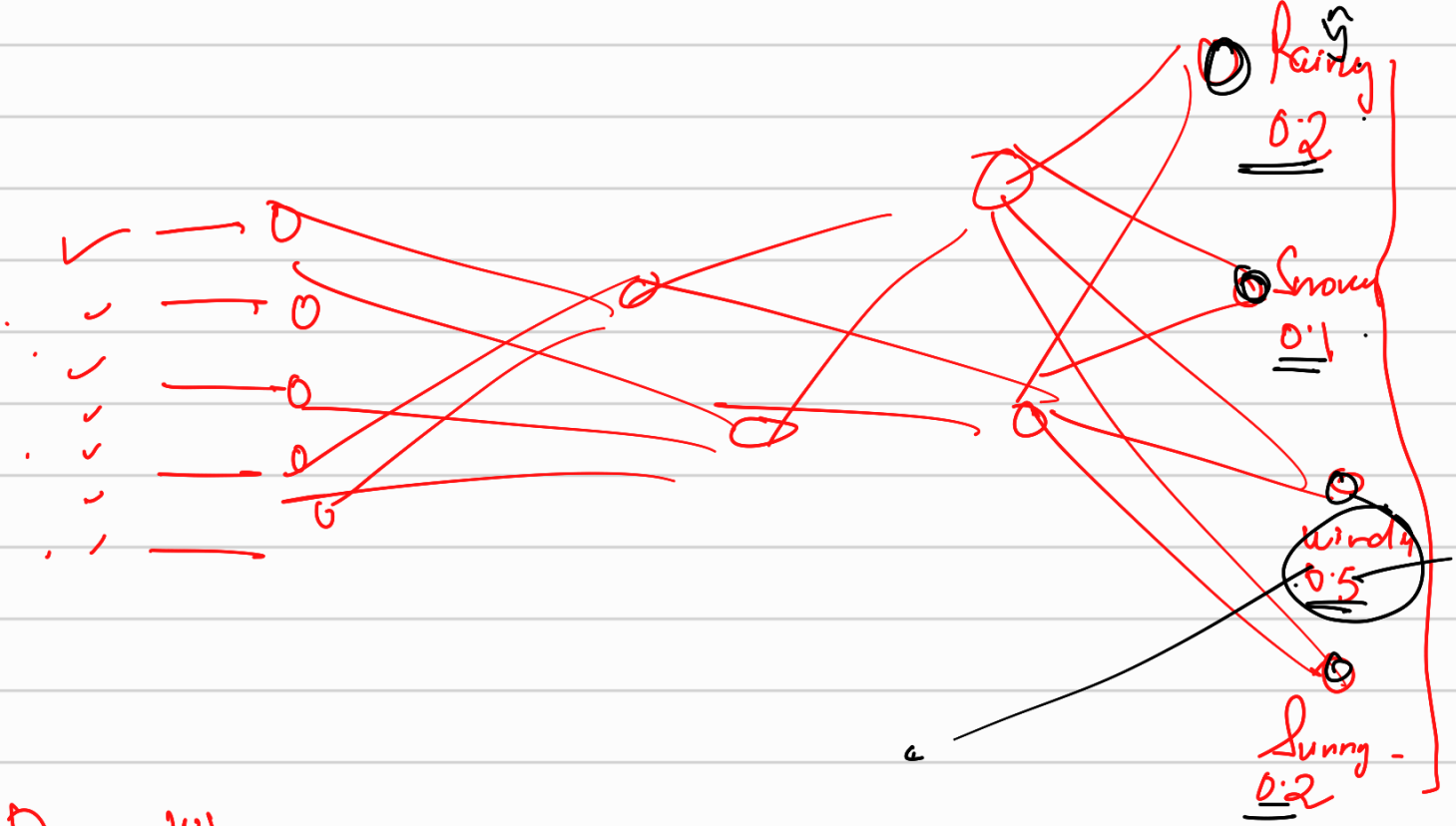
W, b → HLs → Hidden layers
→ Neurons
→ activation function
→ weights and biases → Random
→ optimizer and loss function



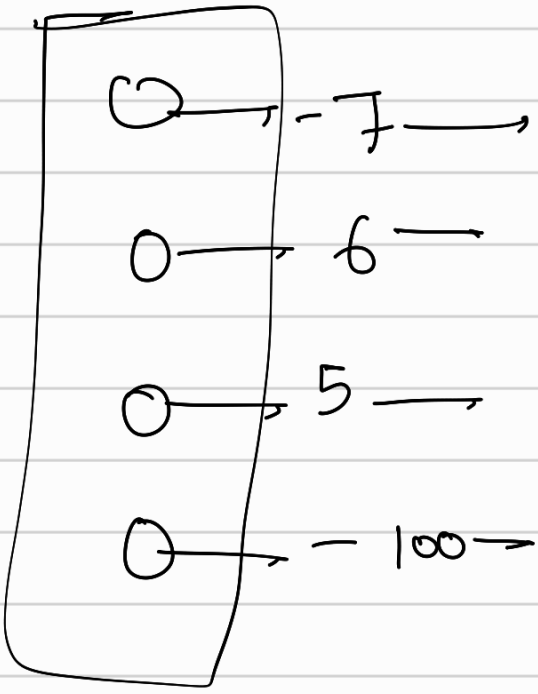
Classification (Binary). → sig($\hat{y}$) → 0 - 1

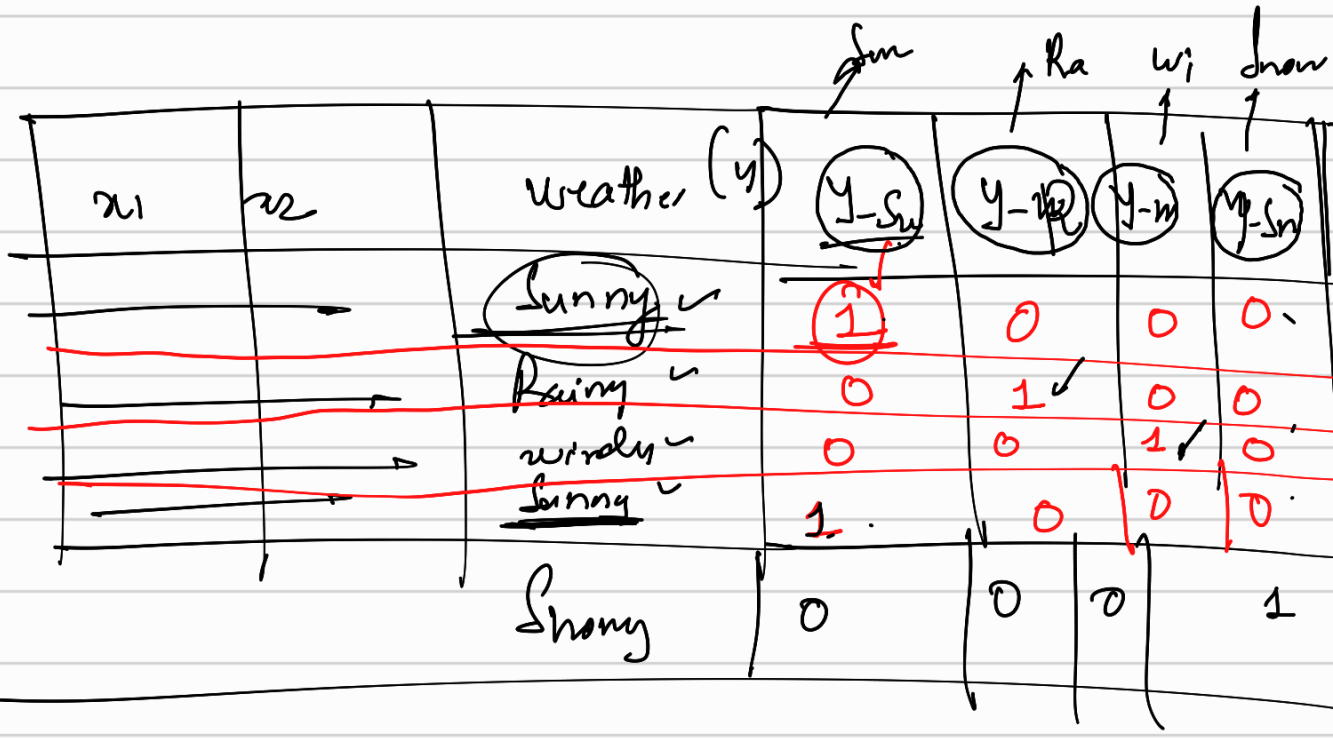
Classification (multiclass). Softmax ($\hat{y}$).

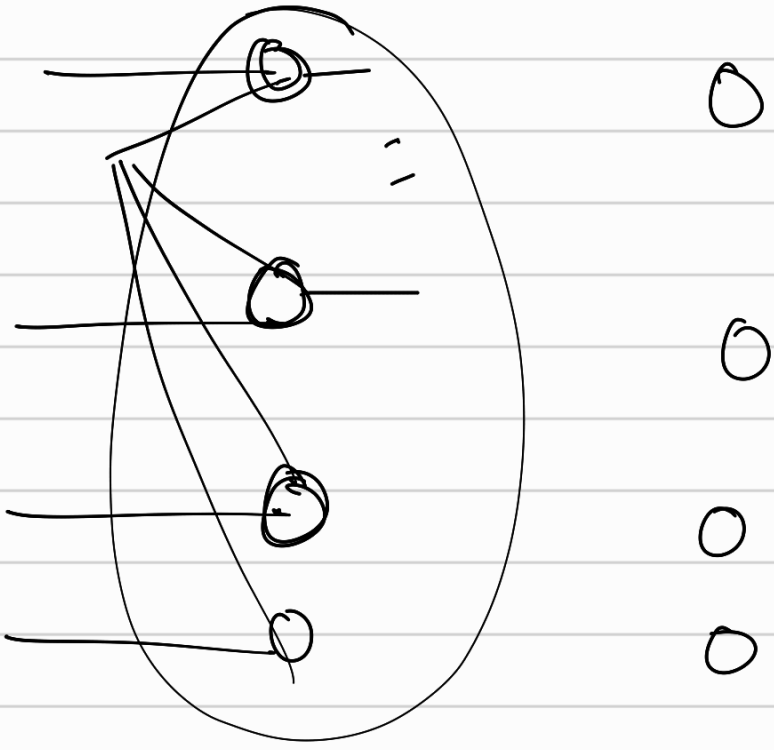


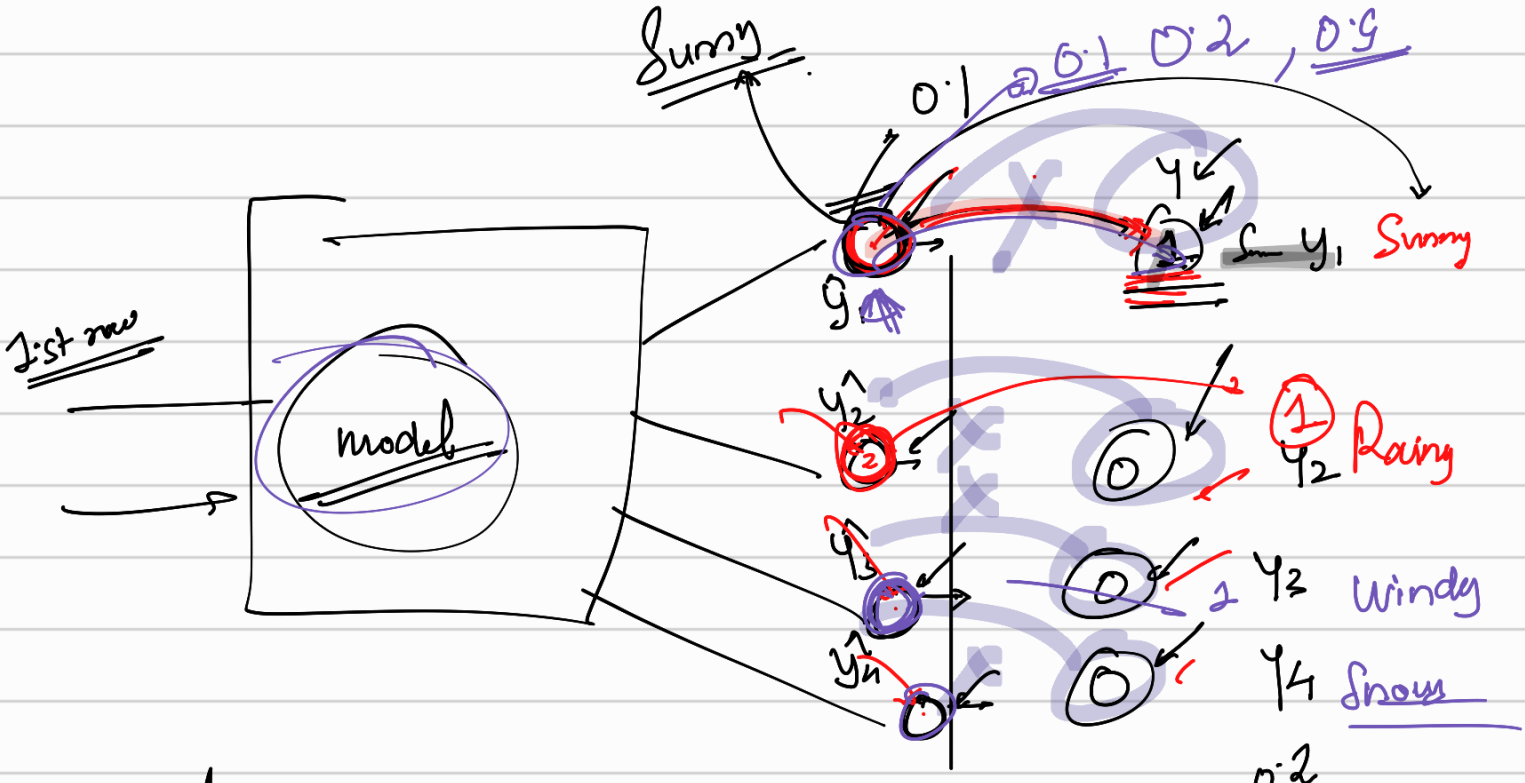

Rainy
0.2
Snowy
0.1
windy
0.5
Sunny -
0.2


x
w1
x


7
6
5
100


Sun
Ra
Wi
Snow
x1
x2
weather (y)
y-Su
y-Ra
y-Wi
y-Sn
Sunny
Rainy
windy
Sunny
Snowy
1
0
0
0
0
1
0
0
0
1
0
0
1
0
0
0
0
1





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

$\hat{y}_1 = 0.2$, $\hat{y}_2 = 0.8$

$- (\log 0.2)$

$= 0.69$

$- (\log 0.8)$

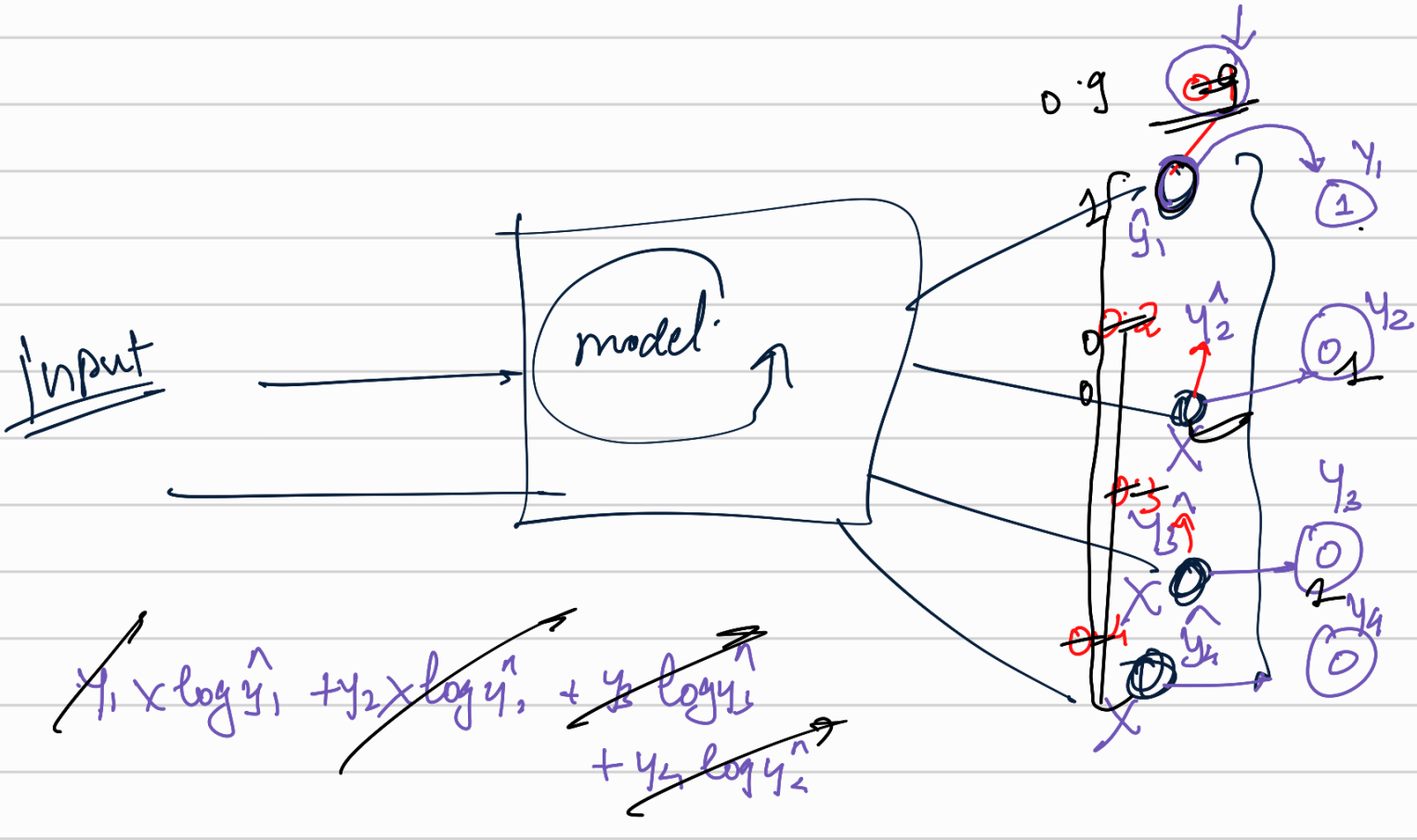
$= 0.09$

$- \log(0.95) = 0.02$

$\log 1 = 0$

$$y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + y_3 \log \hat{y}_3 + y_4 \log \hat{y}_4$$

$$\log(0.1) = 1$$



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

$\log 1 = 0$

①

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

vs

$-\log(0.9) = 0.045$

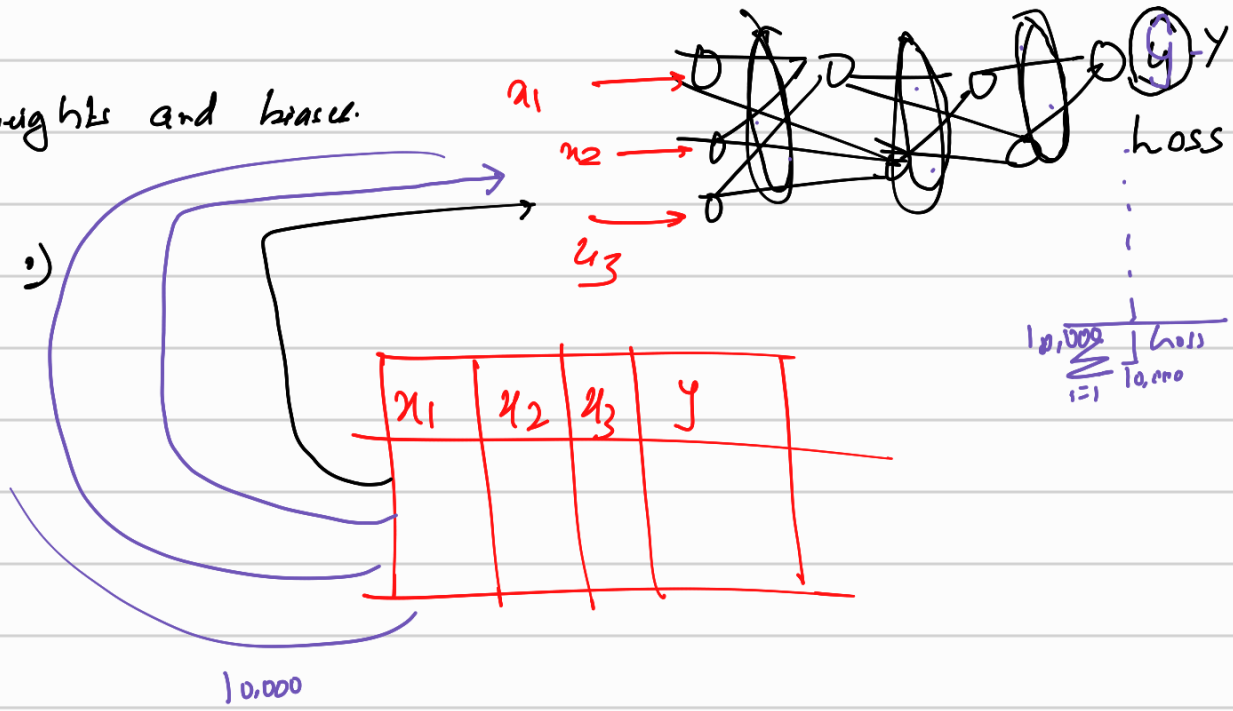
0
0
0

0
0
0

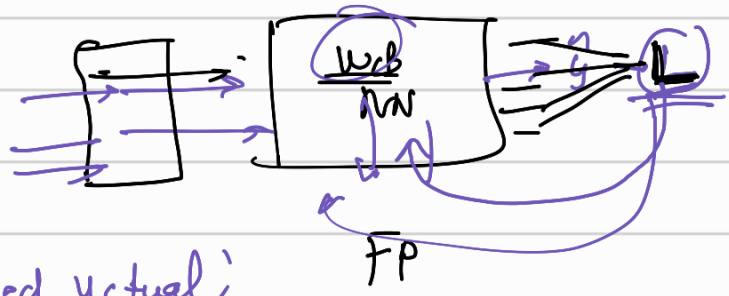
→ Softmax, ohe

log Loss

Random weights and biases.



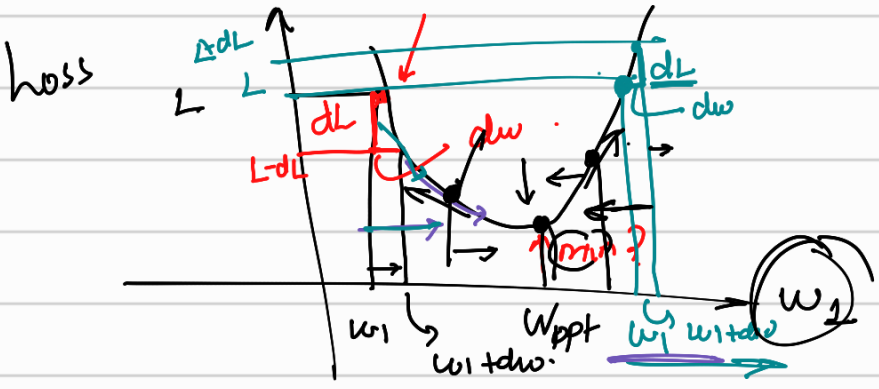
To minimise the loss



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

$L = f((w, B, x), y_{actual})$

$L = f(w, B)$



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

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

which side we should move

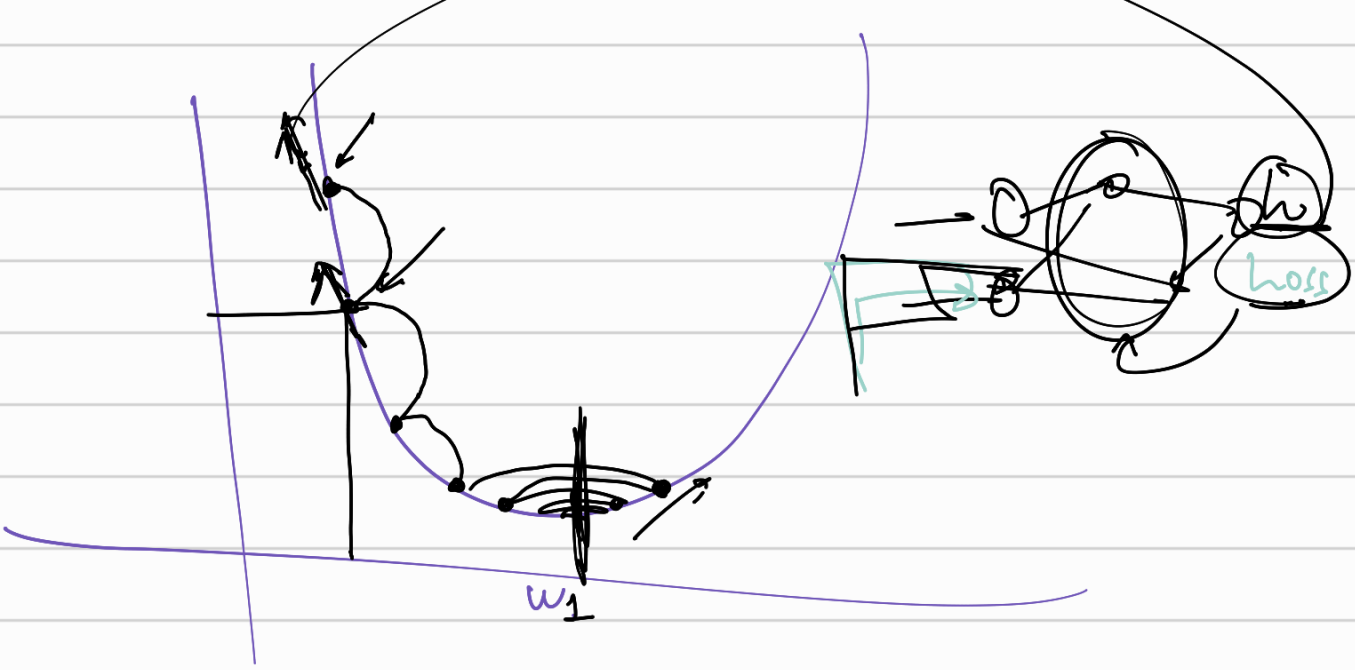
$w = -ve$.

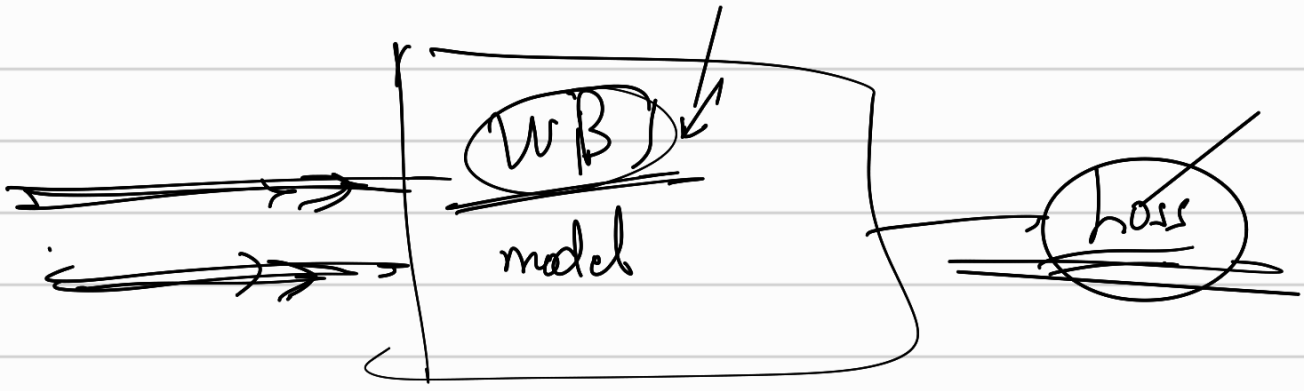
$$w_1(new) = w_1(old) - \alpha\left(\frac{dL}{dw_1}\right)$$

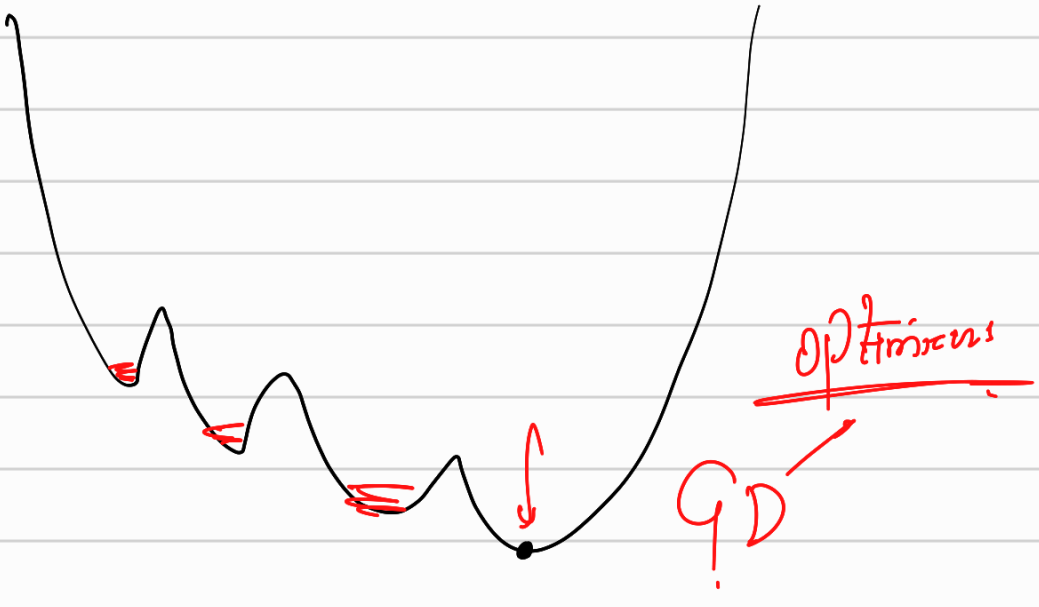
Hyp

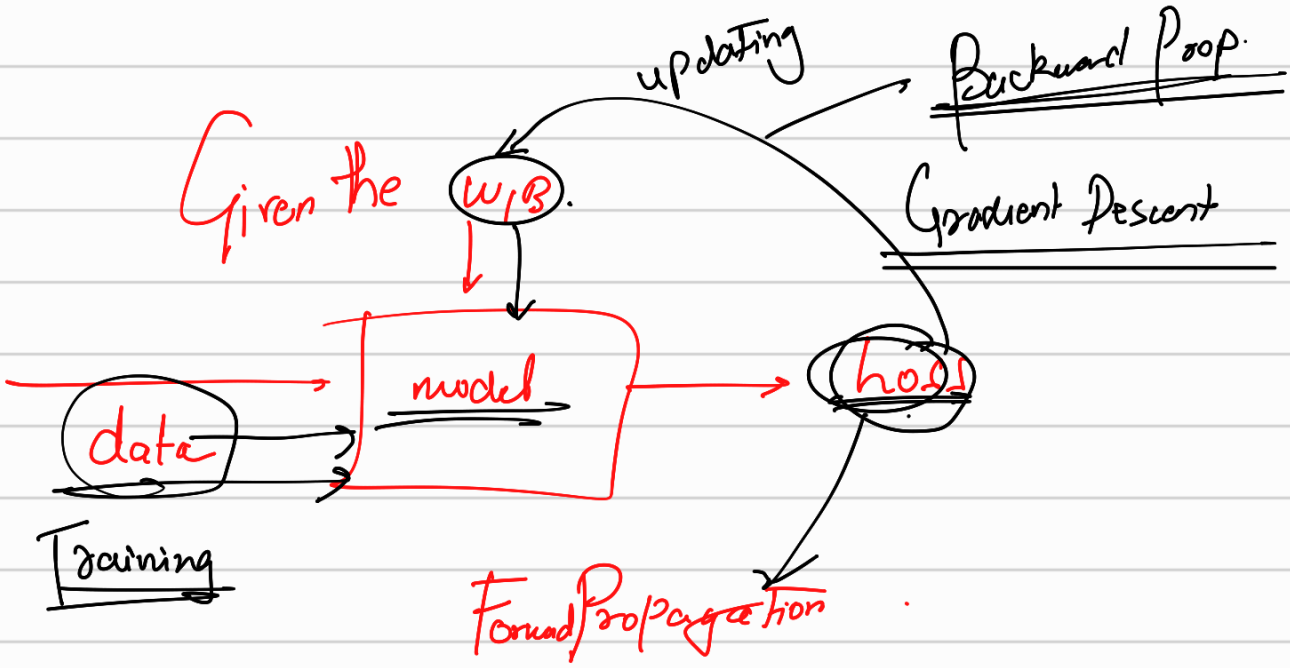
Partial

applicable for all weights and biases.


loss
w_1


WB
model
loss


optimum
GD



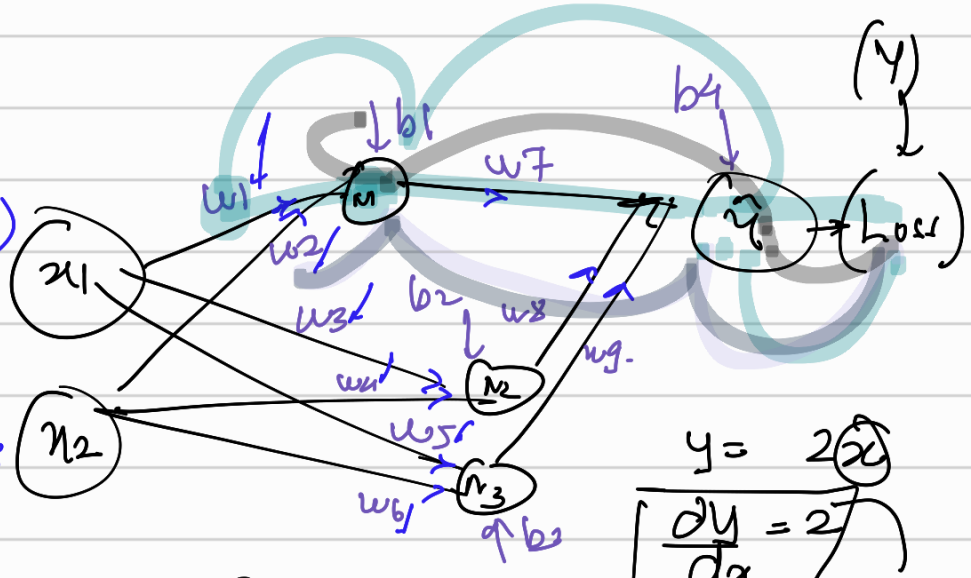
(1FP + 1BP) → epoch

(1FP, 1BP) → 1 epoch.

188 epoch

1FP + 1BP

lecture-3



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

$N_2 = w_3 x_1 + w_4 x_2 + b_2$

$N_3 = w_5 x_1 + w_6 x_2 + b_3$

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

$y = 2x$

$\frac{dy}{dx} = 2$

$\hat{y} = w_7 N_1$

$\frac{d\hat{y}}{dw_7} = N_1$

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

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

$\frac{\partial L}{\partial w_9}$, $\frac{\partial L}{\partial w_1}$ 6mins

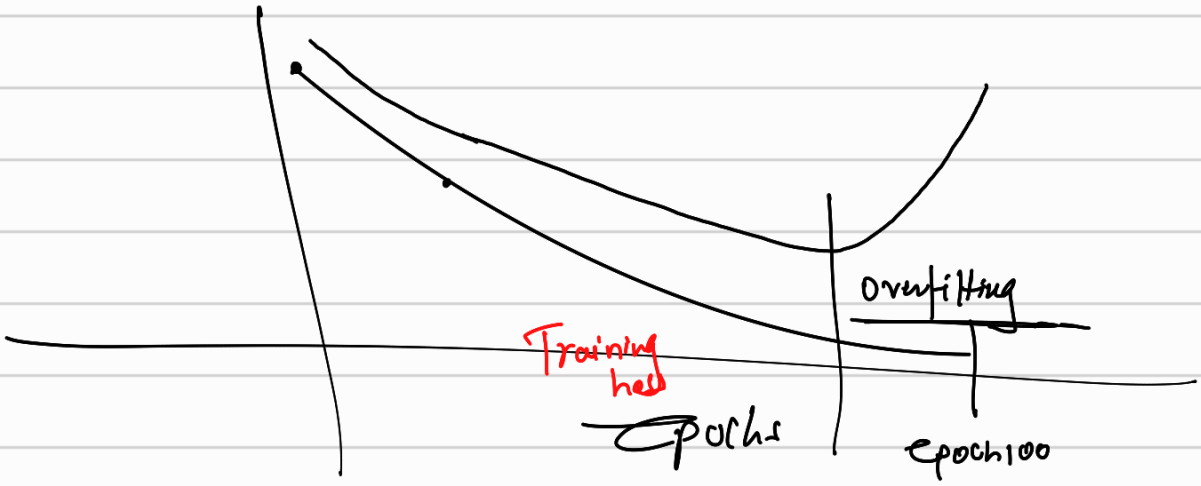
$\frac{\partial L}{\partial w_9} = \frac{\partial L}{\partial \hat{y}} \times \frac{d\hat{y}}{dw_9} \rightarrow N_3$

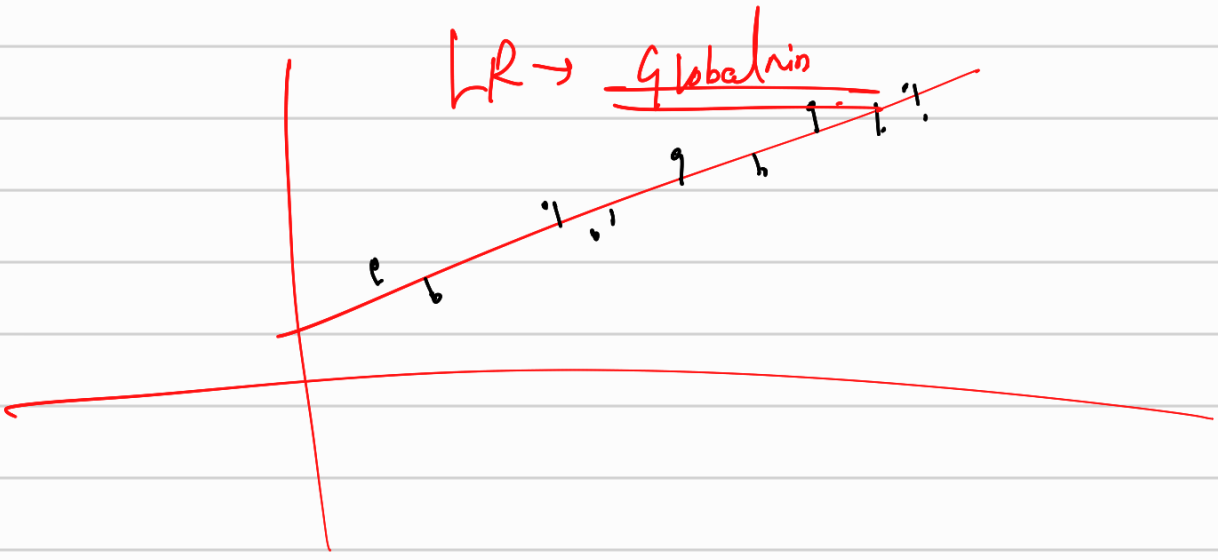
$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \times \frac{d\hat{y}}{dN_1} \times \frac{dN_1}{dw_1}$

$\hat{y} = w_7 N_1 + w_8 N_2 + w_9 N_3 + b_4$ (terms $w_8 N_2$, $w_9 N_3$, b_4 $\rightarrow 0$)

$\frac{d\hat{y}}{dw_7} = \frac{\partial (w_7 N_1)}{\partial w_7} = N_1$

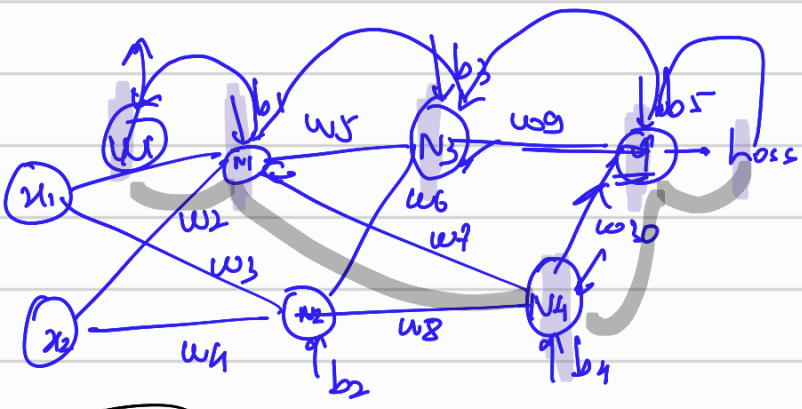
$\frac{d(x \times 2)}{dx} = 2.$


Overfitting
Training
Epochs
epoch100


LR → Global min

Activation functions

$w_1 x_1 + w_2 x_2 + b_1$



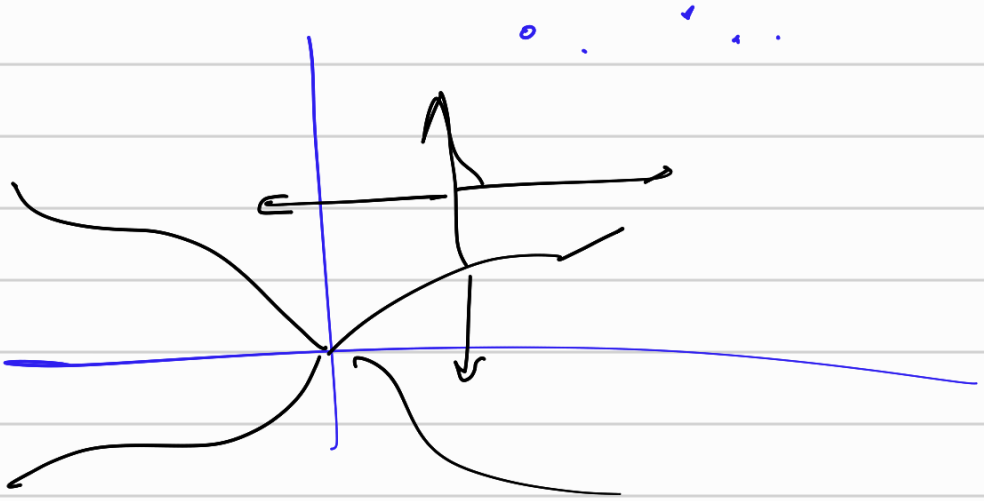
$\frac{\partial L}{\partial w_1}$

$$\frac{\partial L}{\partial w_1} = \left(\frac{dL}{d\hat{y}} \times \frac{\partial \hat{y}}{\partial N_3} \times \frac{\partial N_3}{\partial N_1} \times \frac{\partial N_1}{\partial w_1}\right) + \left(\frac{dL}{d\hat{y}} \times \frac{\partial \hat{y}}{\partial N_4} \times \frac{\partial N_4}{\partial N_1} \times \frac{\partial N_1}{\partial w_1}\right).$$

$\frac{dL}{dw} = 100000$ Terms $= N(HL) + 2$

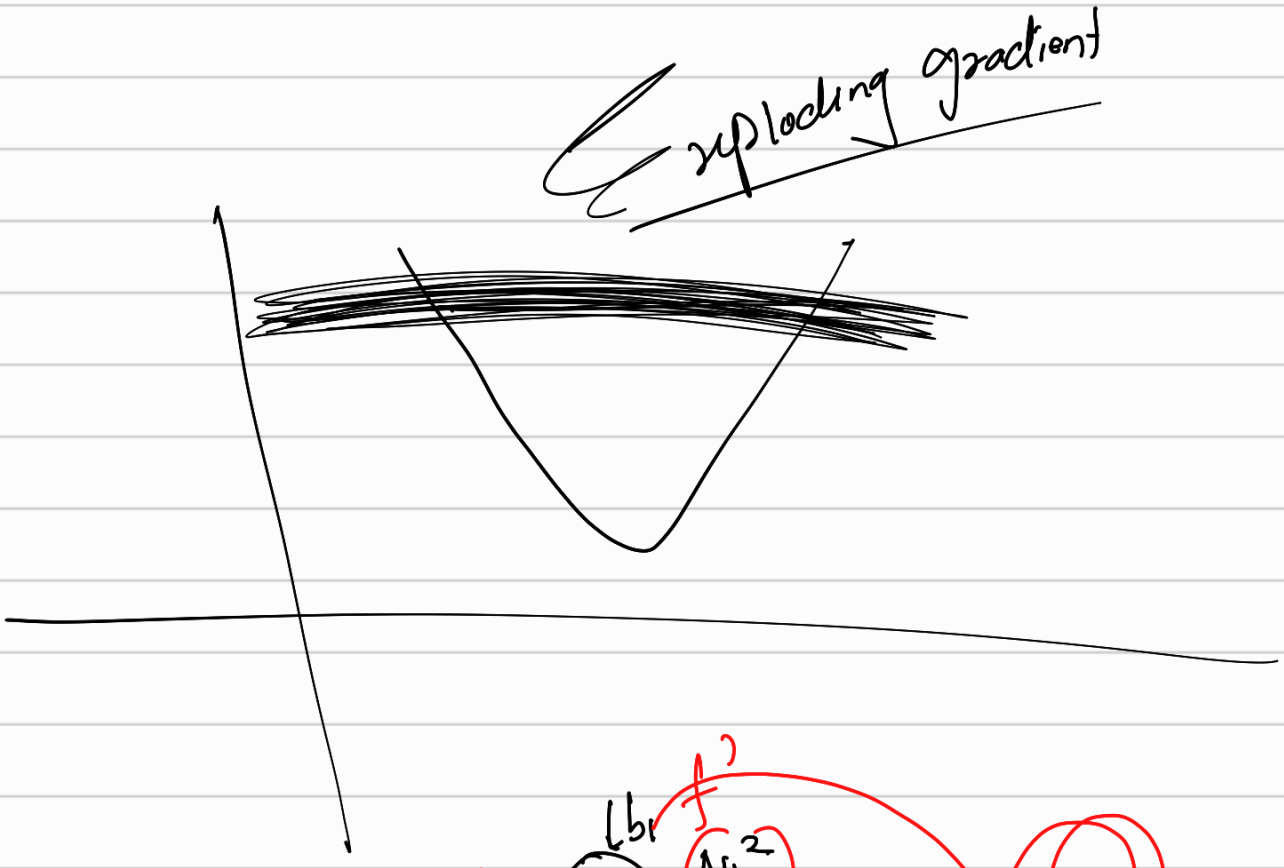
$\frac{\partial L}{\partial w_1} = 0.000002$

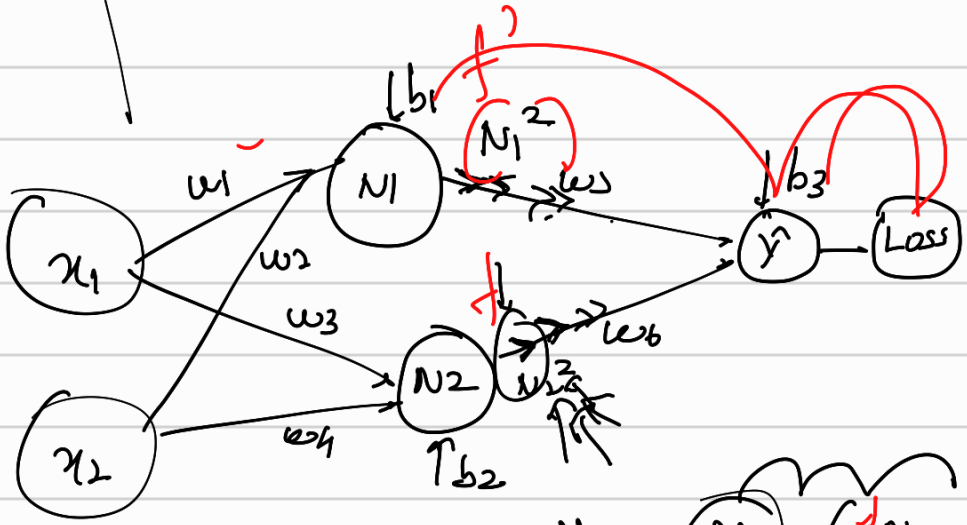
$w_1 -$



$(0.99)^{365} \approx$ 0

$(1.01)^{365} =$ 37.78





$N_1 = w_1 x_1 + w_2 x_2 + b_1$

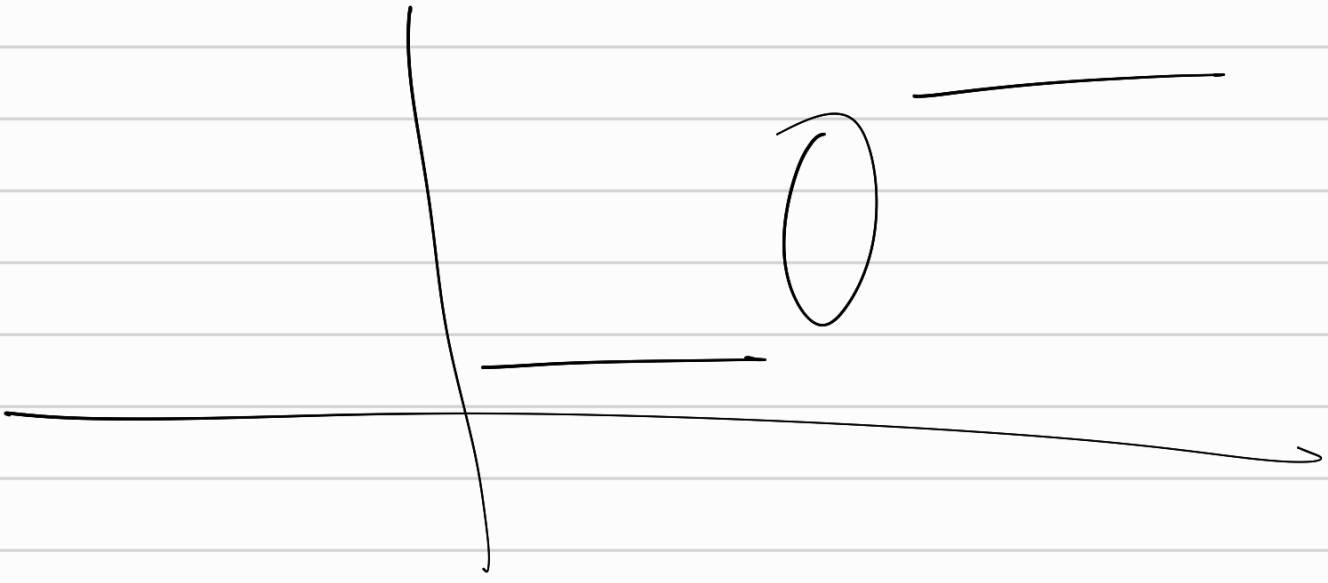
$N_2 = w_3 x_1 + w_4 x_2 + b_2$

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

$$\hat{y} = w_5 (N_1)^2 + w_6 N_2^2 + b_3$$

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

Activation functions → Should be cont & diff

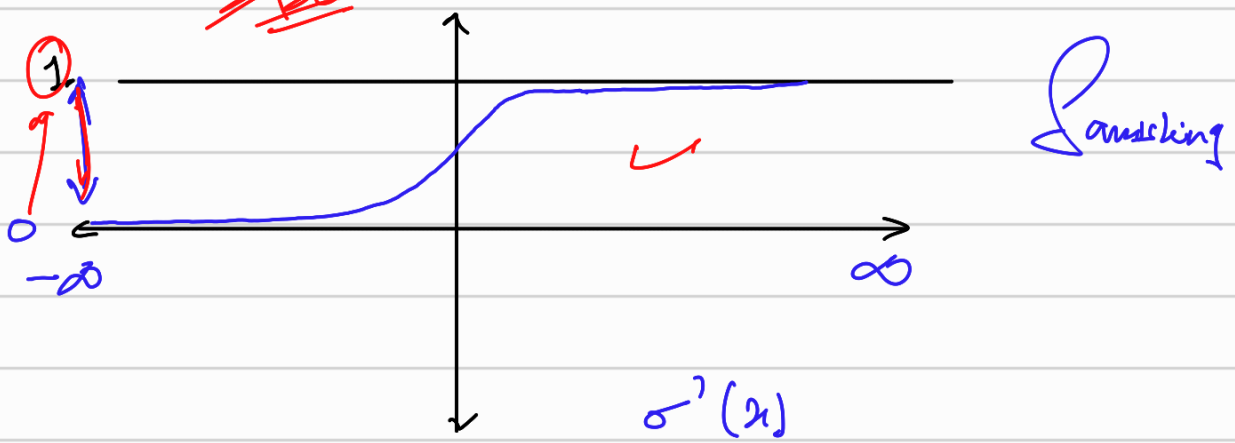


Activation functions are functions which should be continous & diff that can be applied at the output of any hidden layers to create non-linear feature or at the o/p to create bounded outputs → Probs

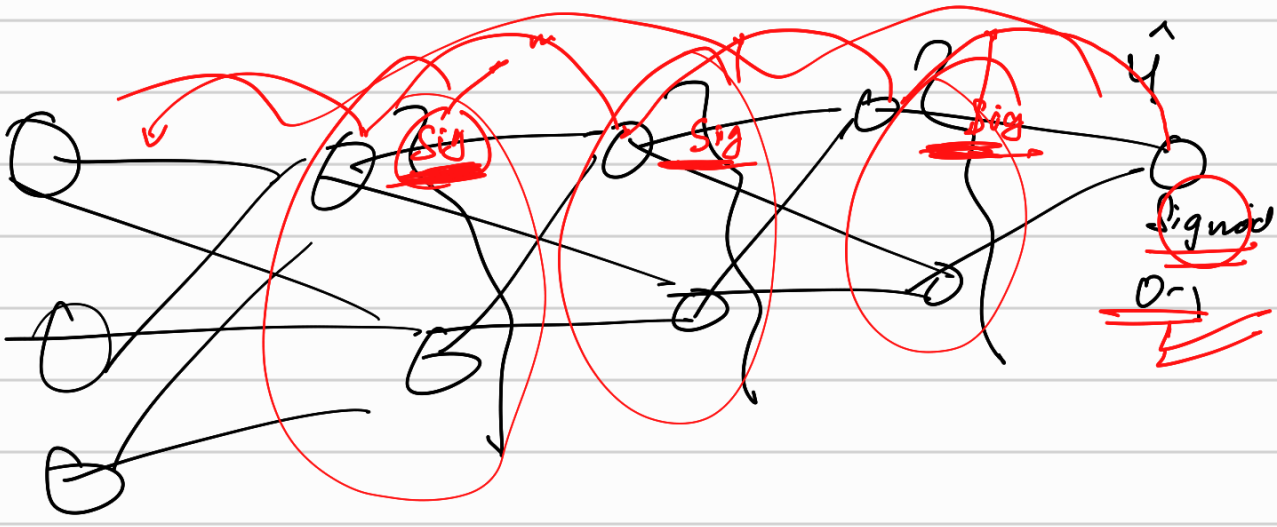
↳ output layer

→ Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ $-\infty < x < \infty$

HLX 0 or 1 Probs $0 < \sigma(x) < 1$

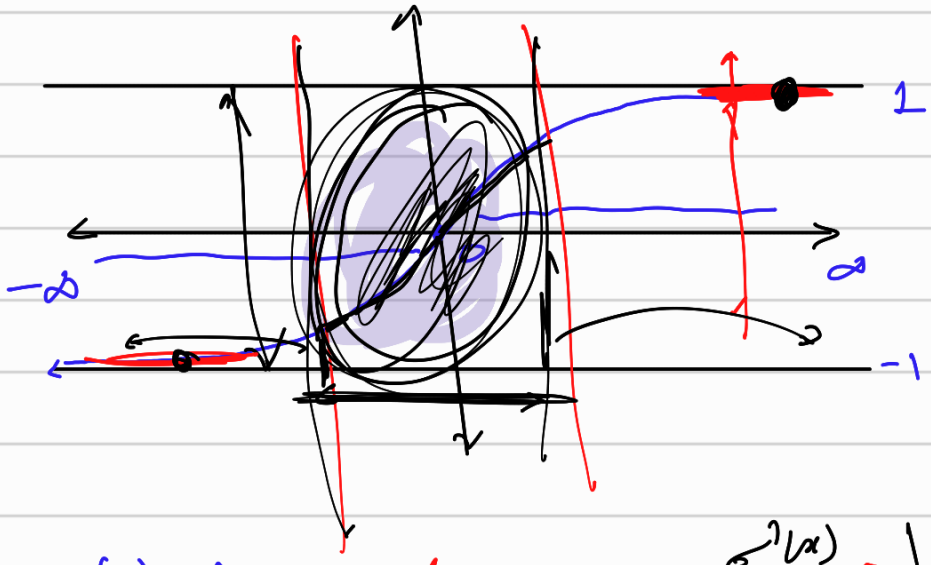


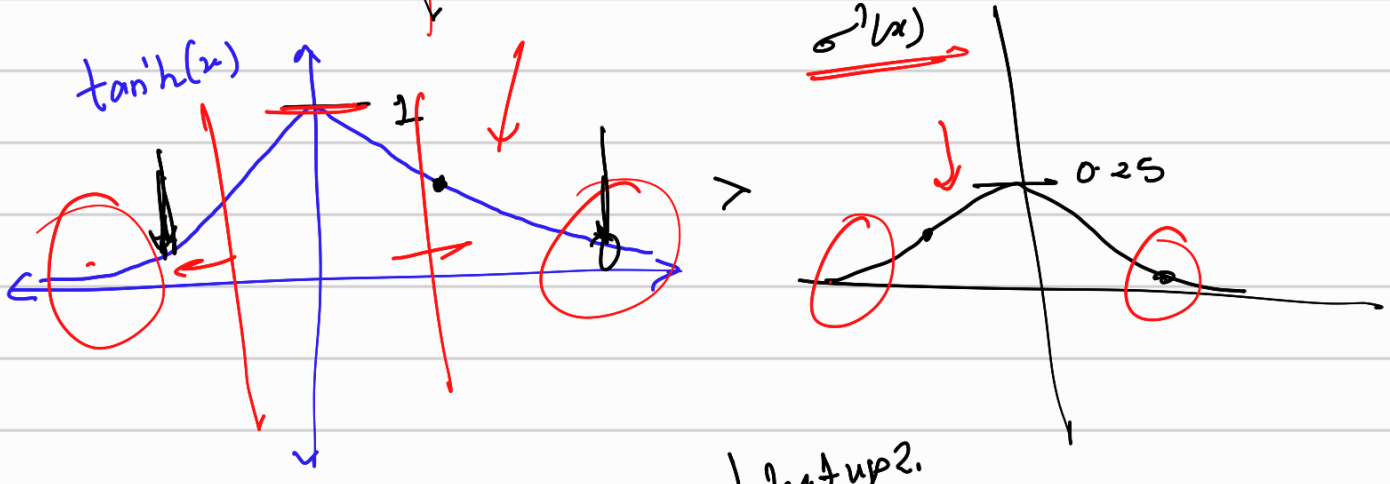
$\sigma'(x)$

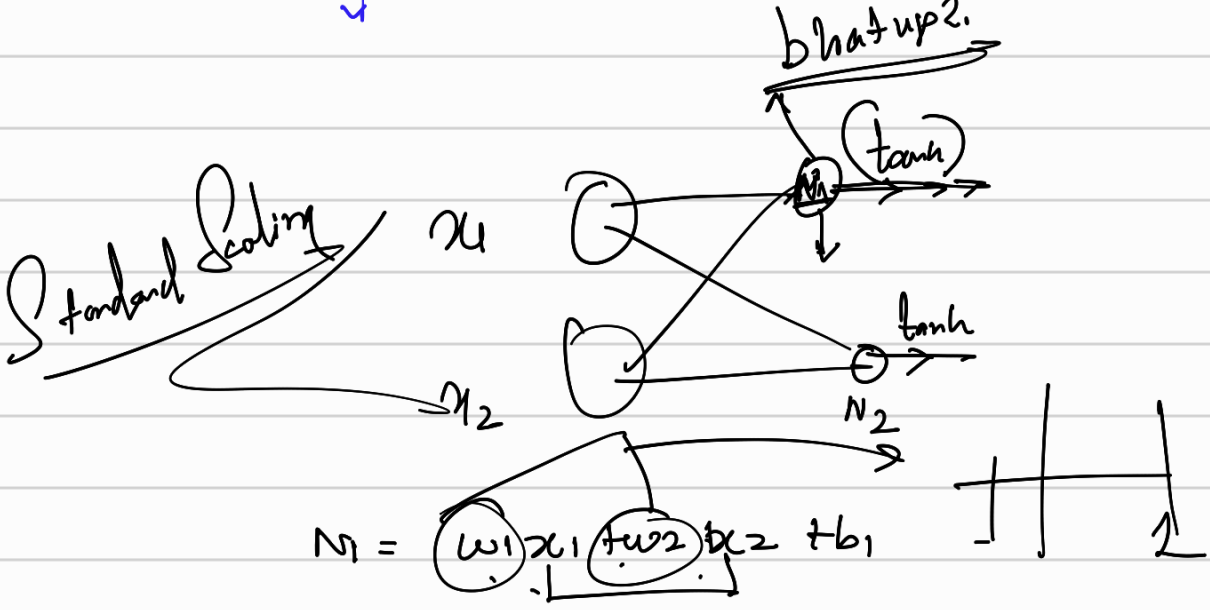


tanh → hyperbolic tangent

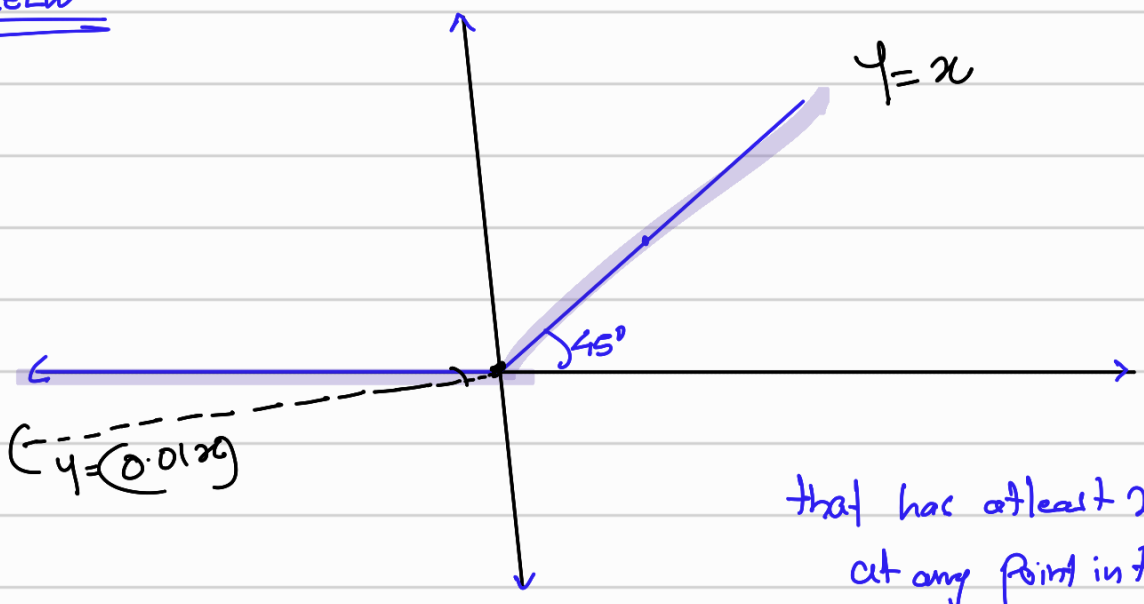
$$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$







ReLu



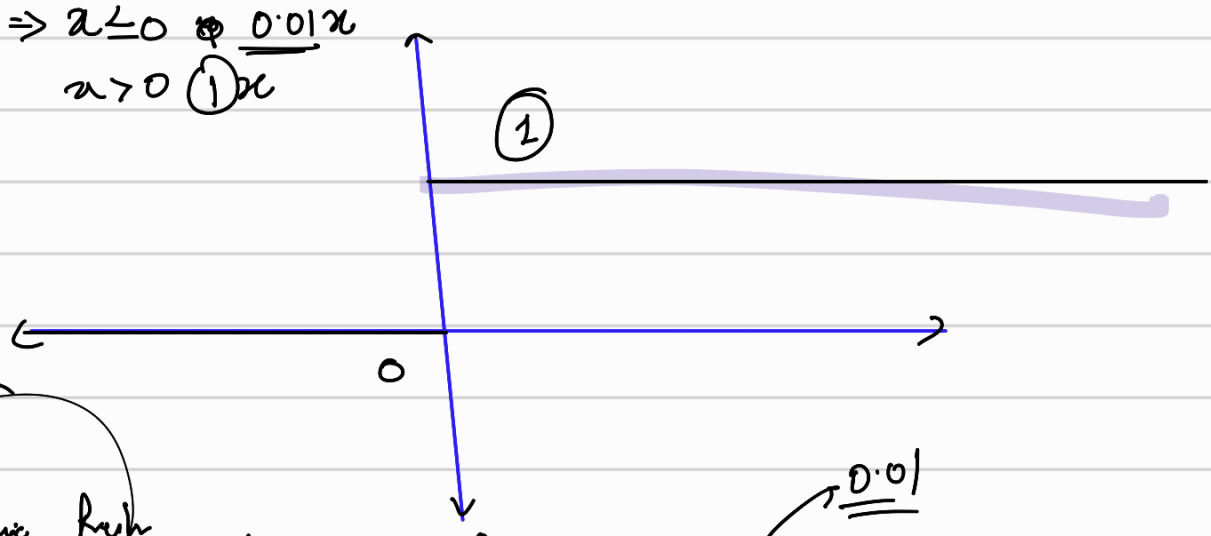
that has atleast 2-slopes at any point in the x space.

ReLu(x) = $x \le 0$ → 0

$x > 0$ → x.

Leaky ReL(x) ⇒ $x \le 0$ → $0.01x$

$x > 0$ → (1)x

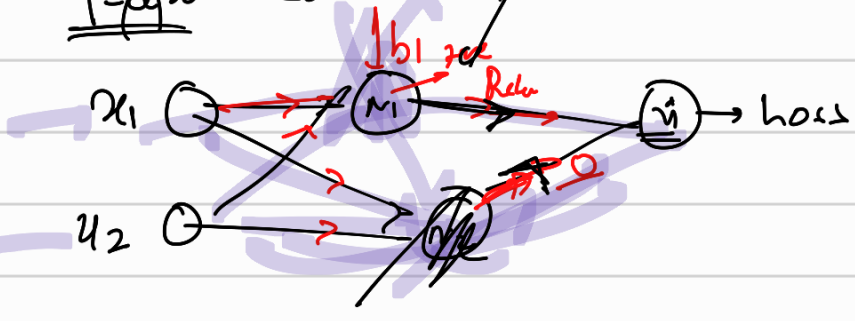


Parametric ReLu

$y = x$ $x > 0$

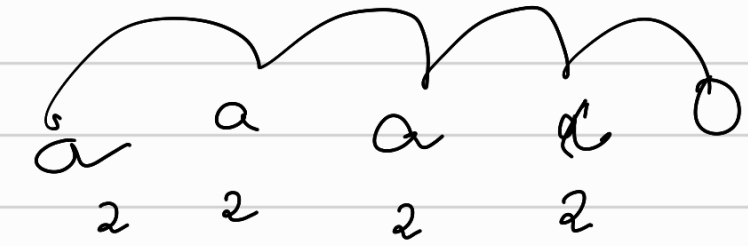
$y = \alpha x$ $x \le 0$

$0 < \alpha < 1$ → 0.01



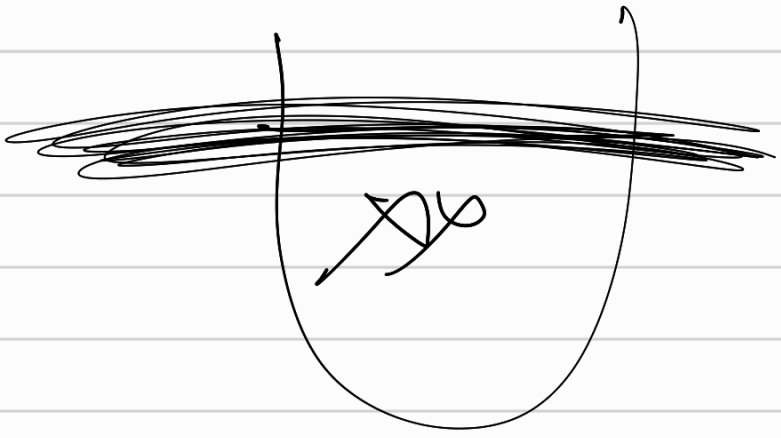
Deep Neural network

$(2)^{0+1+2+\cdots 50} = 32$

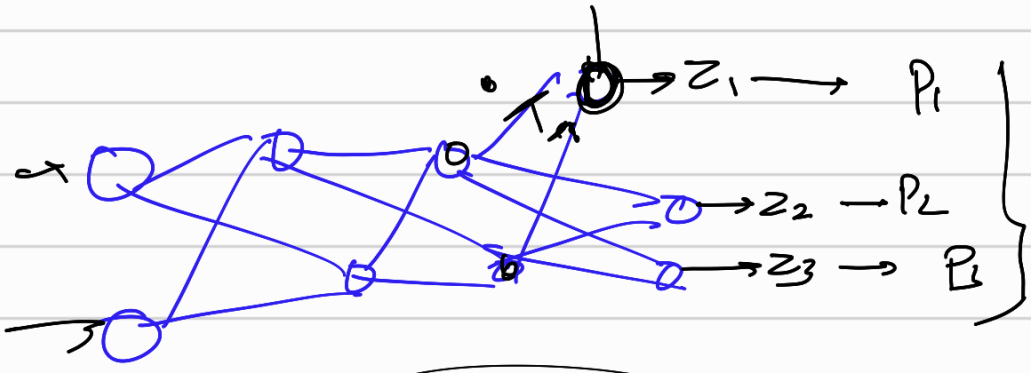


$(2)^3$ $(2)^{100}$

$$\frac{dh}{dw_1} = w_1 - \alpha(\quad)$$

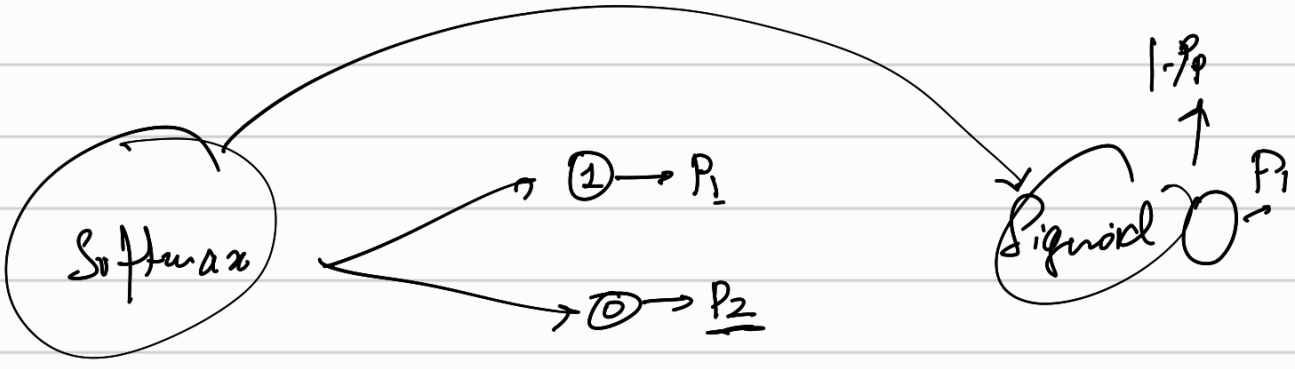


Softmax : needs to be applied at the o/p of a multiclass problem



$p_1 + p_2 + p_3 = 1$

$$p_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \qquad p_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}} \qquad p_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$



$p_1 + p_2 = 1$

$p_1 = 1 - p_2$

$p_2 = 1 - p_1$

Break

Lecture -5:

Sequential API → Tabular → output

Functional API

text → model → Class

↑ Tweet

↑ tab

Sparse cce.

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

$(y = 1)$

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

0 1 2

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

$-\log 0.3$

$y \rightarrow$ OHE.

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

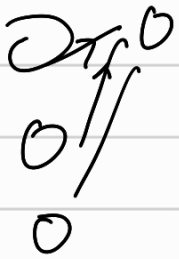
cce.

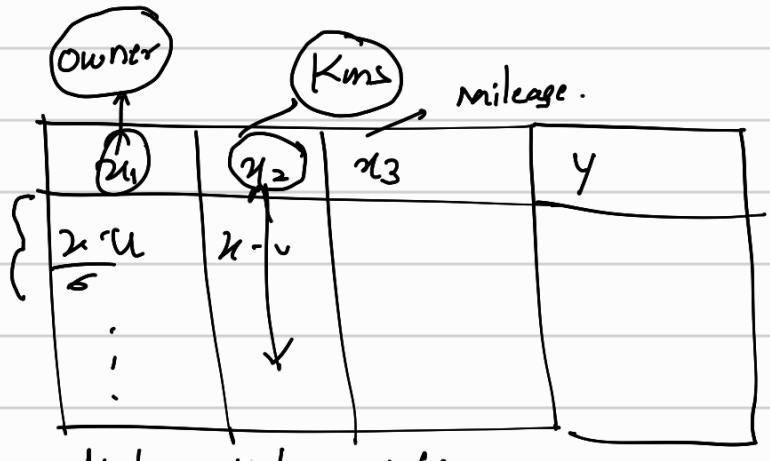
$\hat{y} = [0.1 \quad 0.3 \quad 0.6]$.

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

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

$= -\log 0.3$





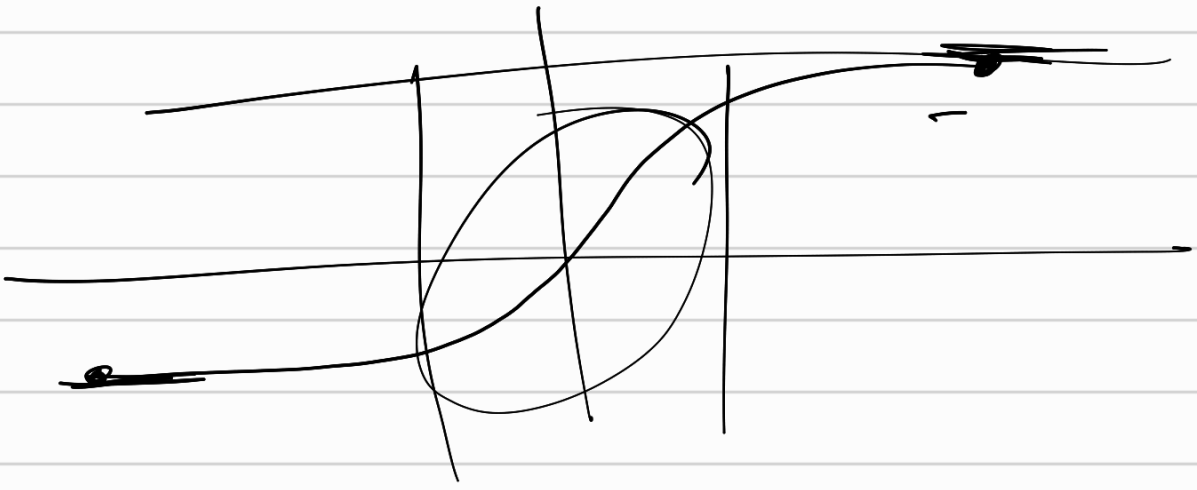
$w_1 \times \text{owner} + w_2 \times \text{kms} + w_3 \times \text{mileage}$

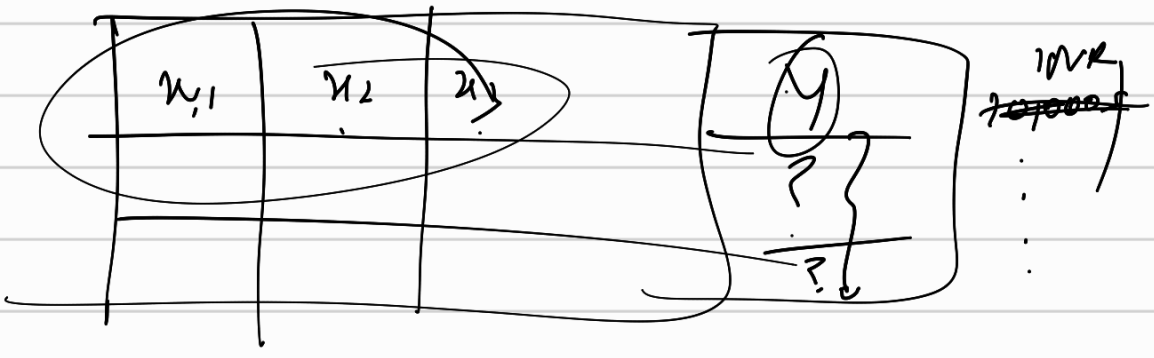
(1) Honda City

50,000 ↑

(2) ~~2nd~~ Honda city

40,000



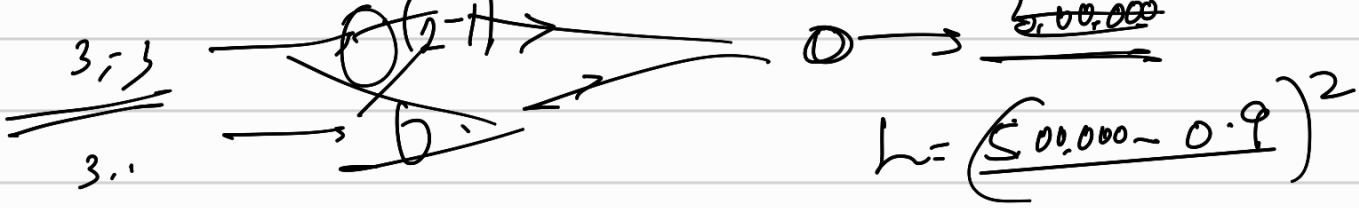


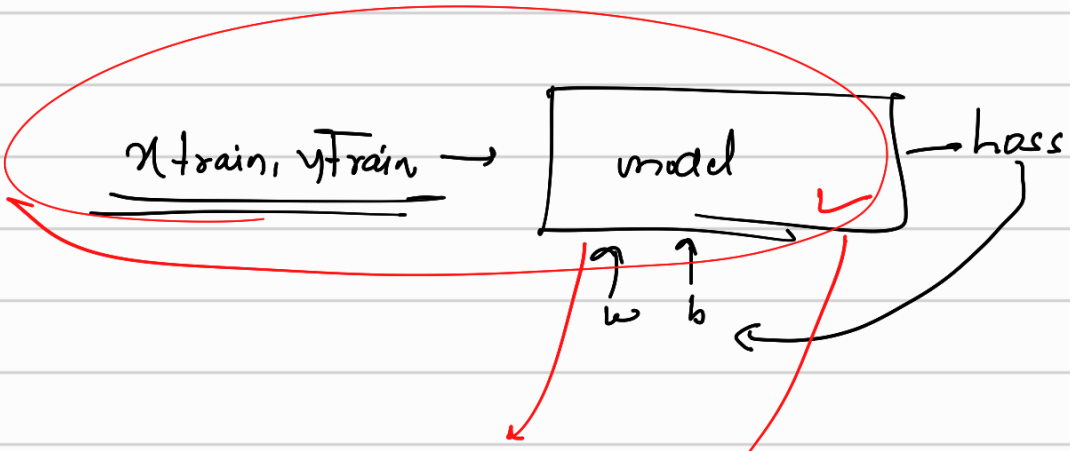
3.-3

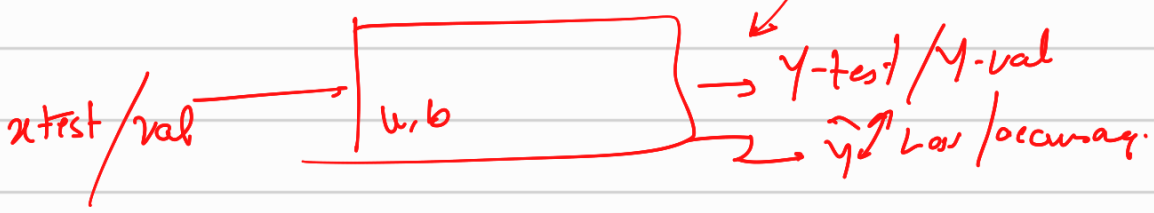
$(w_i \sim -1, 1)$

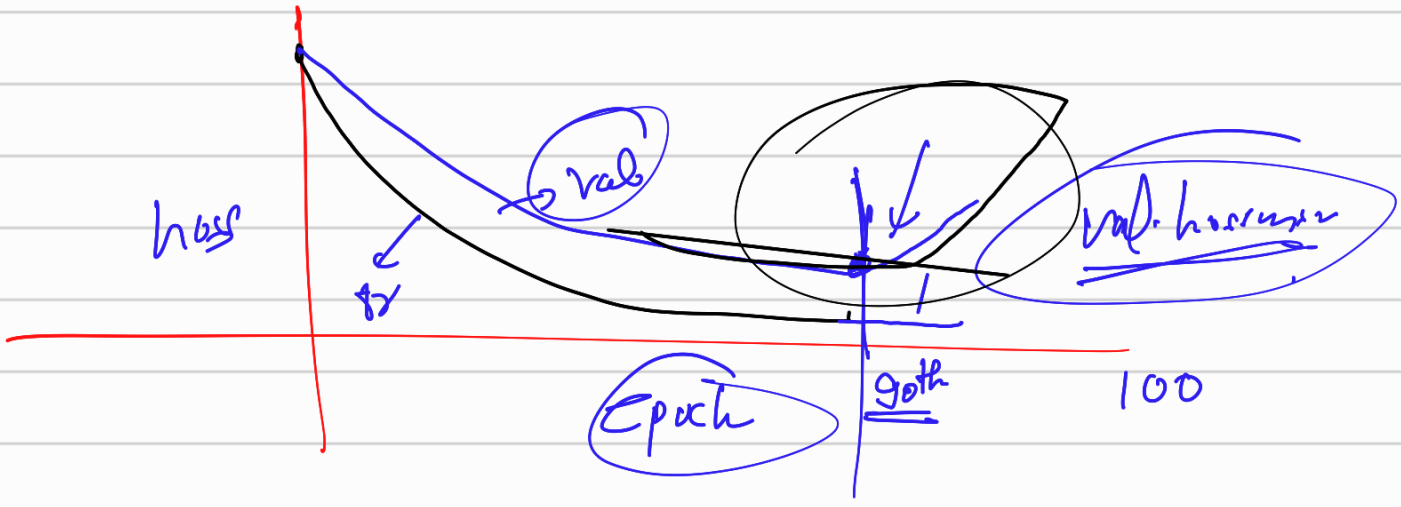
3;3

3.1

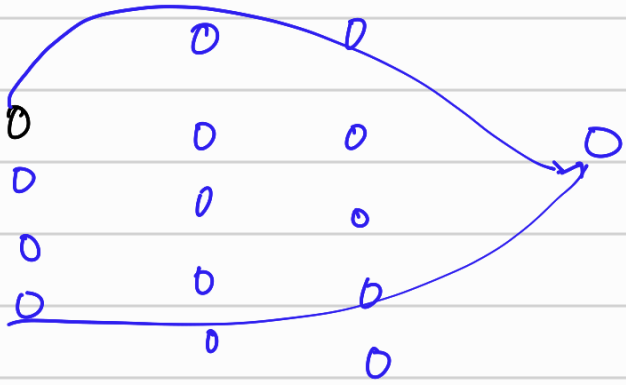


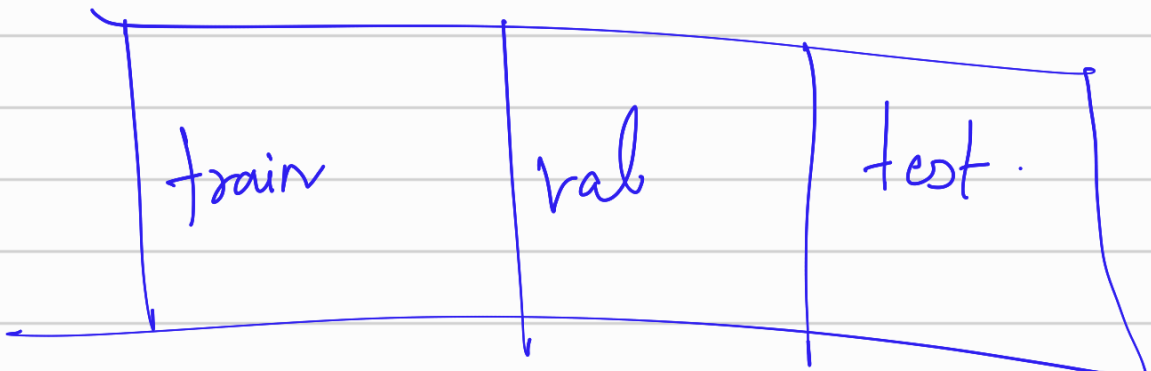

Xtrain, ytrain
model
loss
w
b


xtest/val
w, b
y-test/y-val
ŷ
Loss/accuracy


loss
val
val. loss
Epoch
90th
100

Callbacks





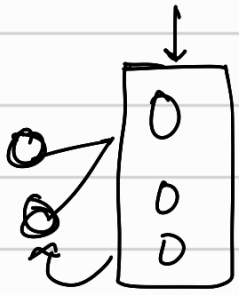
model.fit(train, val = val).

model.evaluate(test)

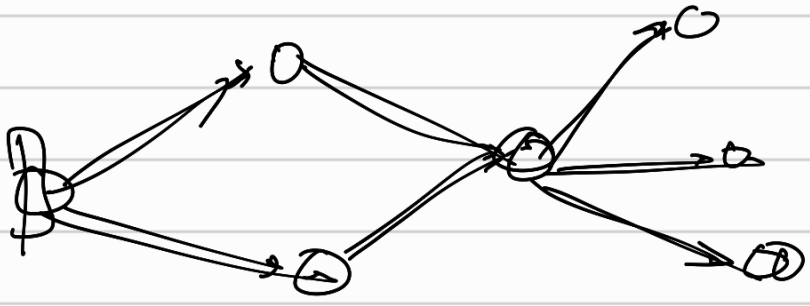
82%

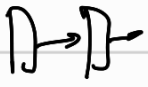
Functional API:

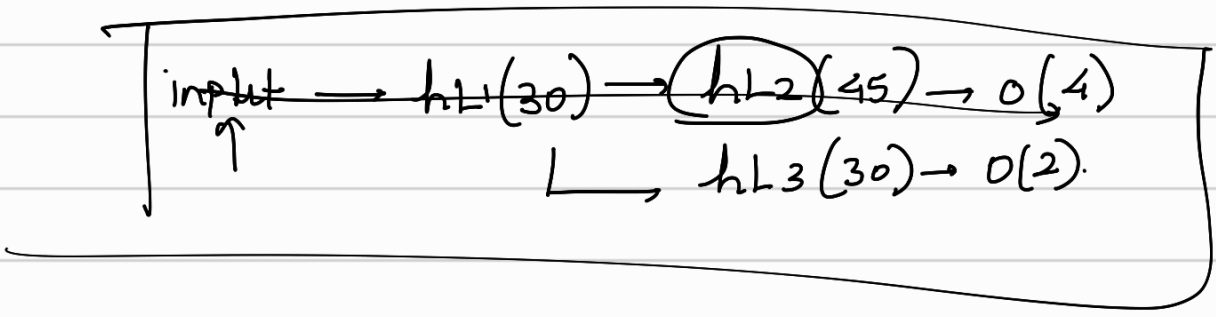
Graph Structure

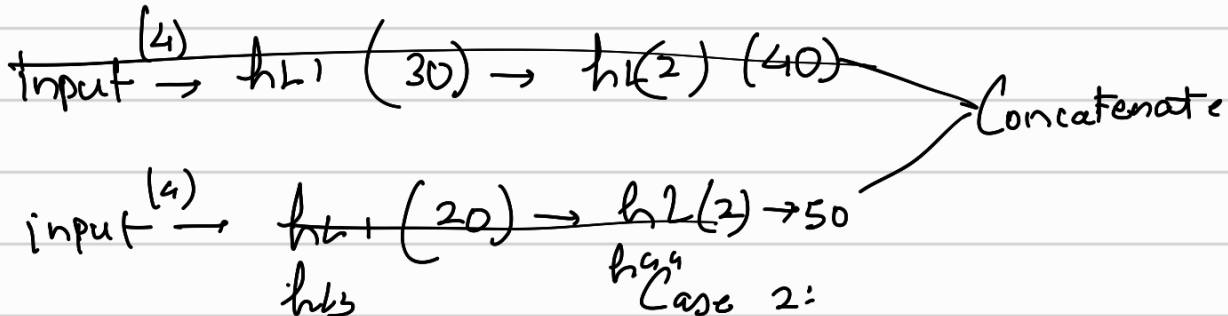


any layer (✓)







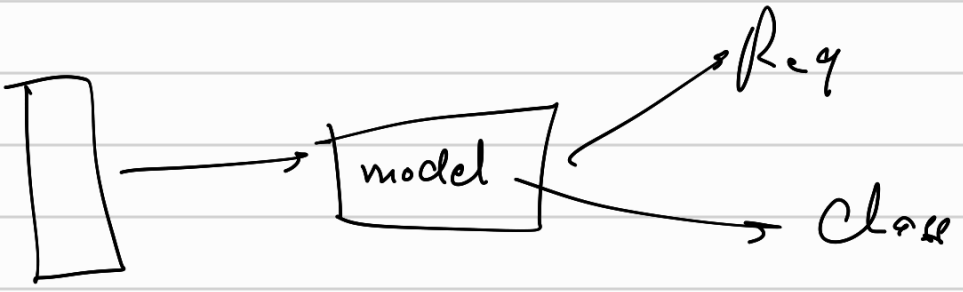


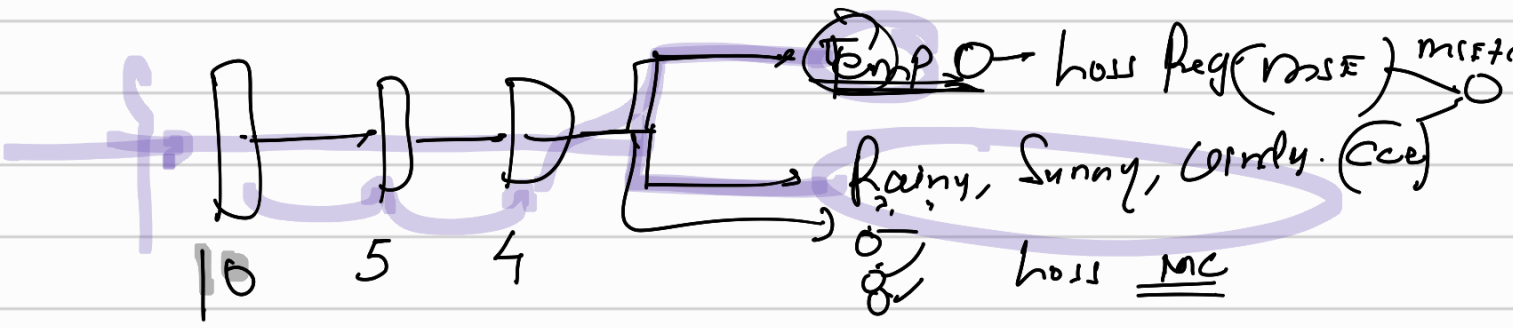
Case 1:

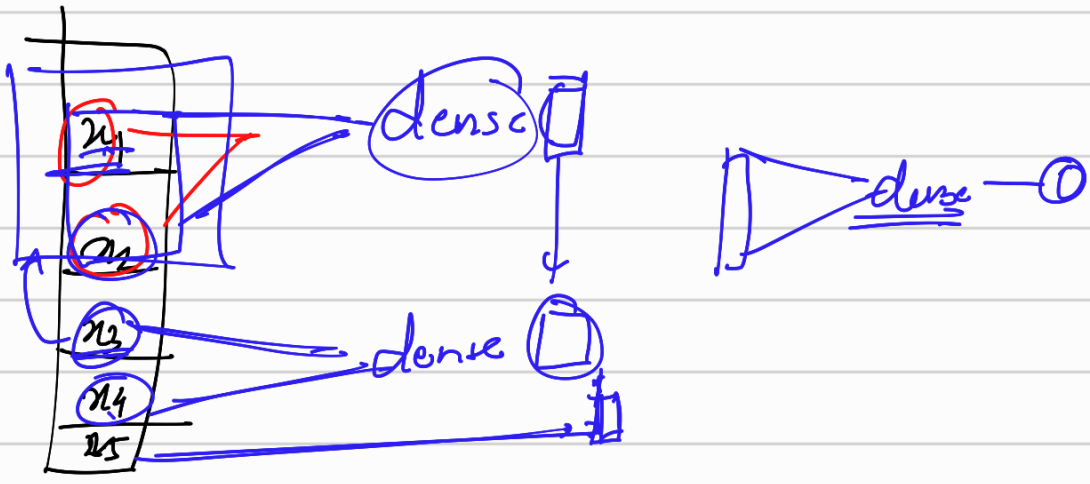
```
inputs = Input (4,)
hL1 = Dense(30) (inputs)
hL2 = Dense (45, activation = 'relu') (hL1)
O1 = Dense (4) (hL2).
hL3 = Dense(30) (hL1)
O2 = Dense (2) (hL3).
```

Case 2:

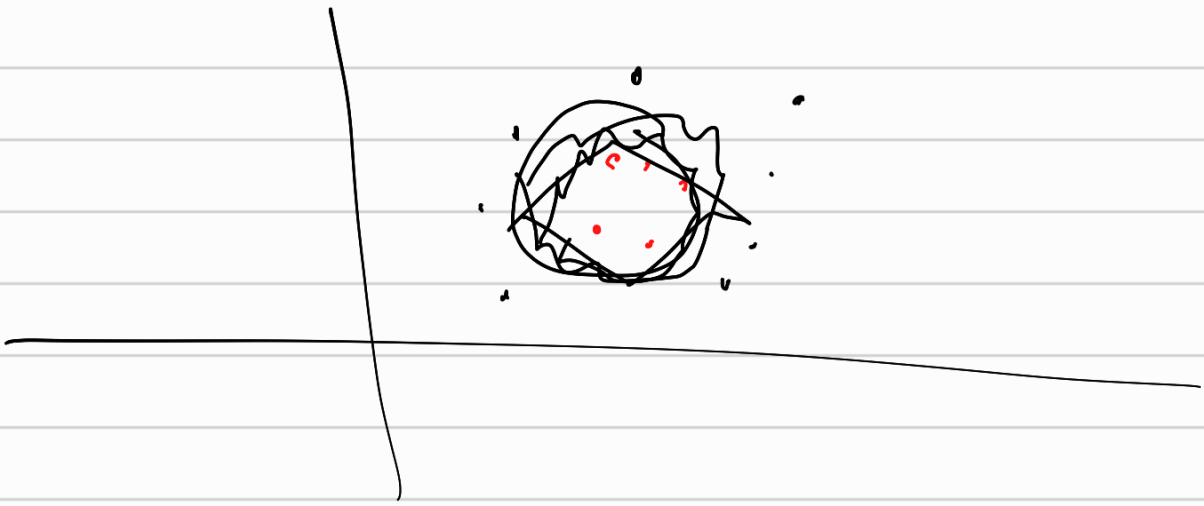
```
inp1 = Input (4,)
inp2 = Input (4,).
hL1 = Dense(30) (inp1).
hL2 = Dense (40)(hL1).
hL3 = Dense (20) (inp2).
hL4 = Dense (50) (hL3)
O = Concatenate() ([hL2, hL4])
```

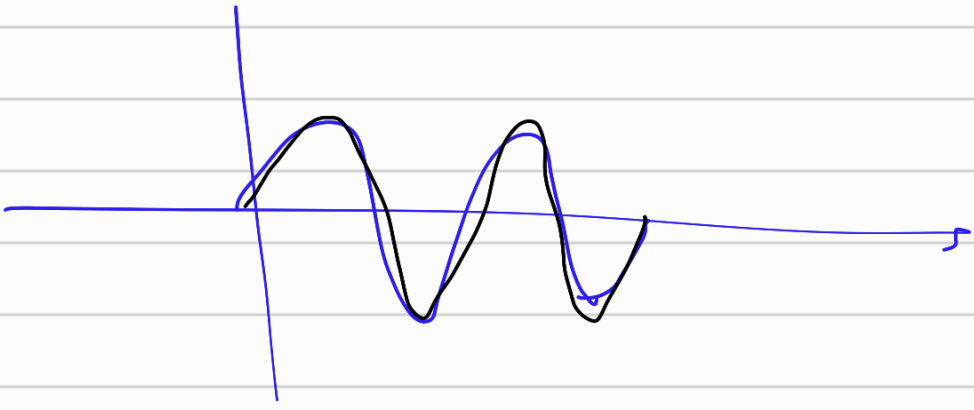



10
5
4
Temp
loss Reg(MSE)
Rainy, Sunny, Windy. (CCE)
loss MC

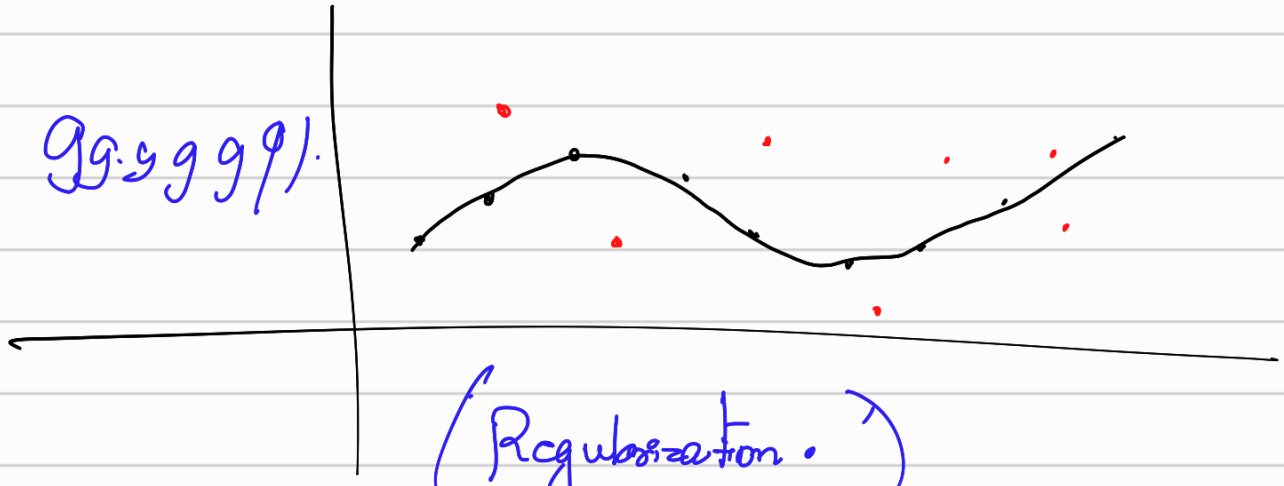

dense
dense
dense


Optimizers
loss
w_1





99.9999%.



(Regularization.)

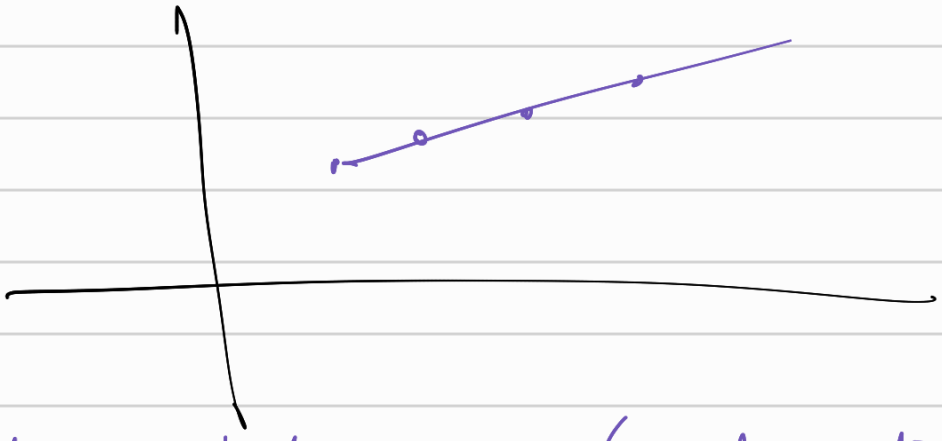
- Regularization
  - Dropout.
  - optimizers.
  - Callbacks
  - Batch Normalization
  - weight initialization

→ Hyperparameter Tuning.

Regularization:

↓↓

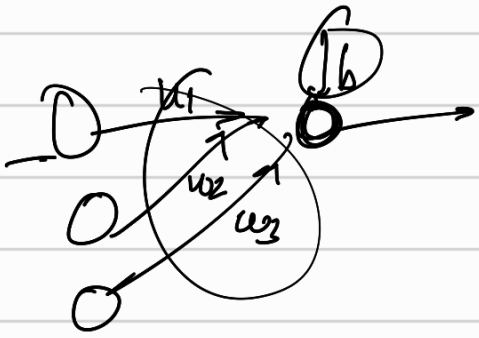
(Ridge and Lasso.) ?



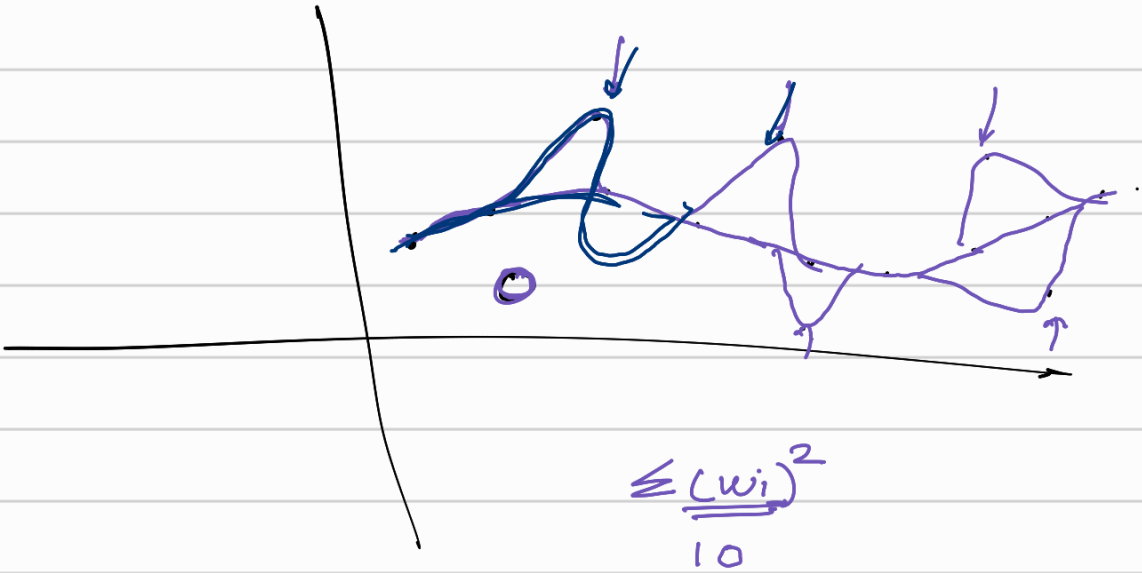
Overfit → has high variance (model is extremely sensitive to changes in x dist).

① $y = 1000x_1 + 500x_2$ $\quad y = 1000$

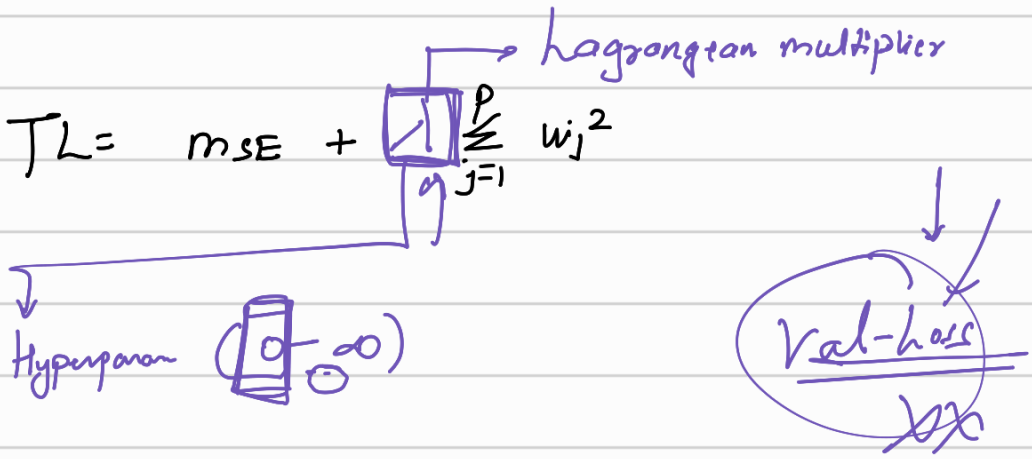
② $y = 2x_1 + x_2 \quad = ②$

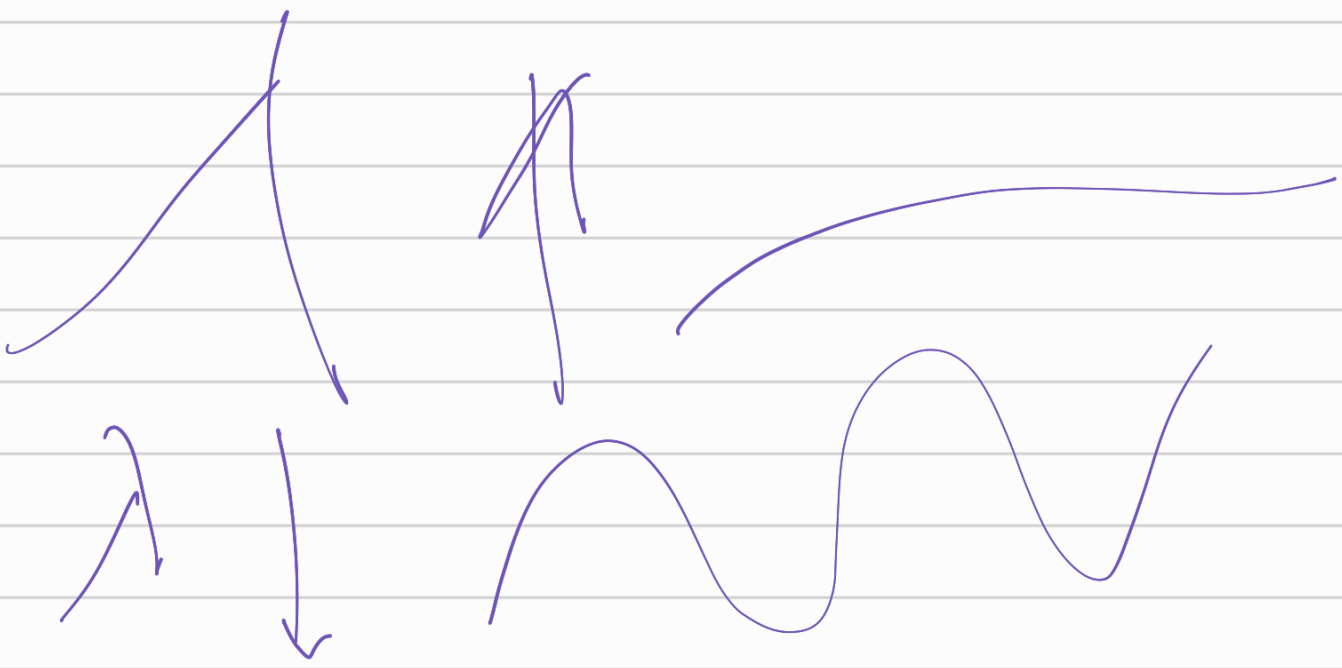


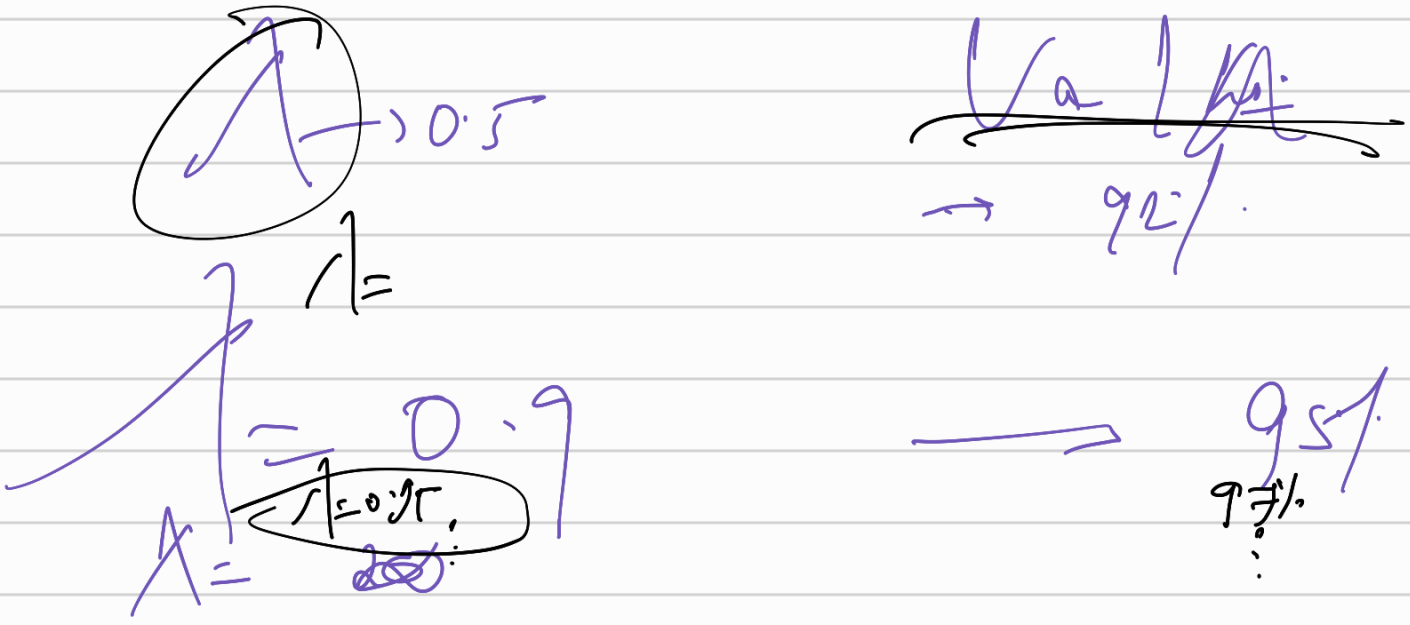
$N_1 = w_1 x_1 + w_2 x_2 + w_3 x_3 + b_1$



TL	MSE	$\sum (w_i)^2$
510 =	500	10
421 =	400	25
375 =	320	55
355 =	260	95
320 =	200	= 120
305 =	150	= 155
345 =	145	= 200


Lagrangian multiplier
TL = mse + $\sum_{j=1}^{p} w_j^2$
Hyperparam (0 ~ ∞)
Val-loss




0.5
λ =
= 0.9
λ = 0.7
Val loss
→ 92%
95%
97%

Reg.

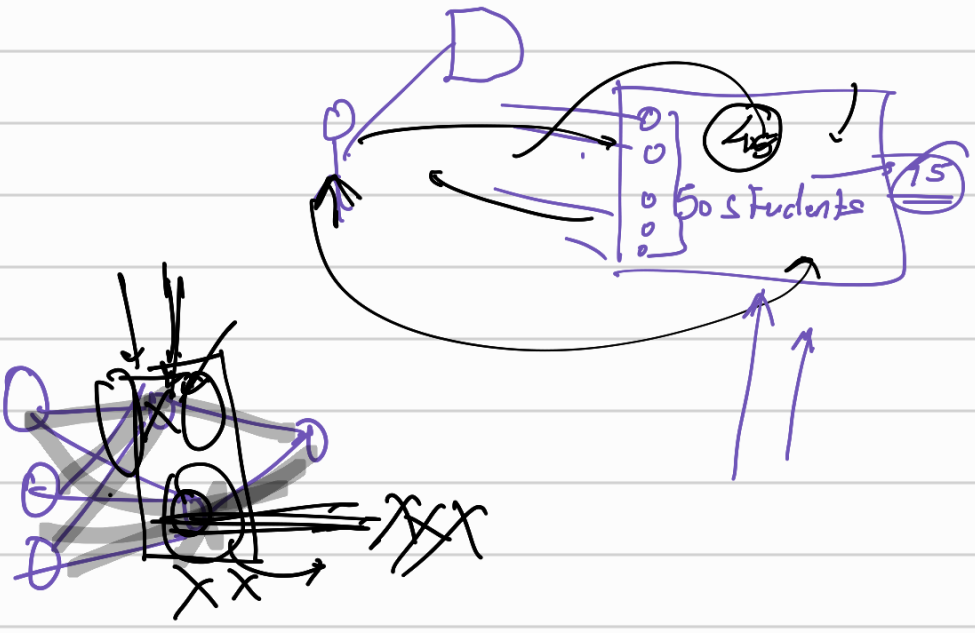
$Th = mse + \lambda \Sigma w_j^2$ Ridge (L^2).

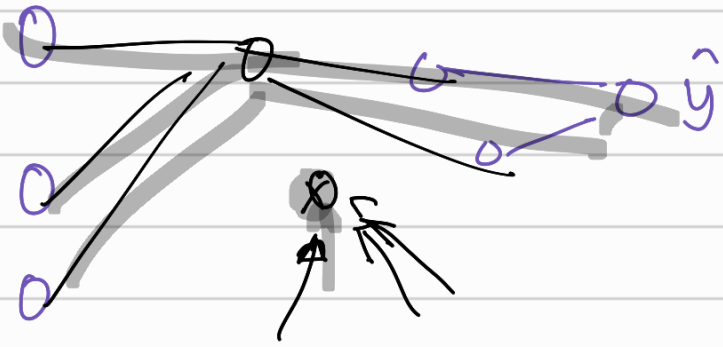
$Th = mse + \lambda \Sigma |w_j|$ Lasso (L_1).

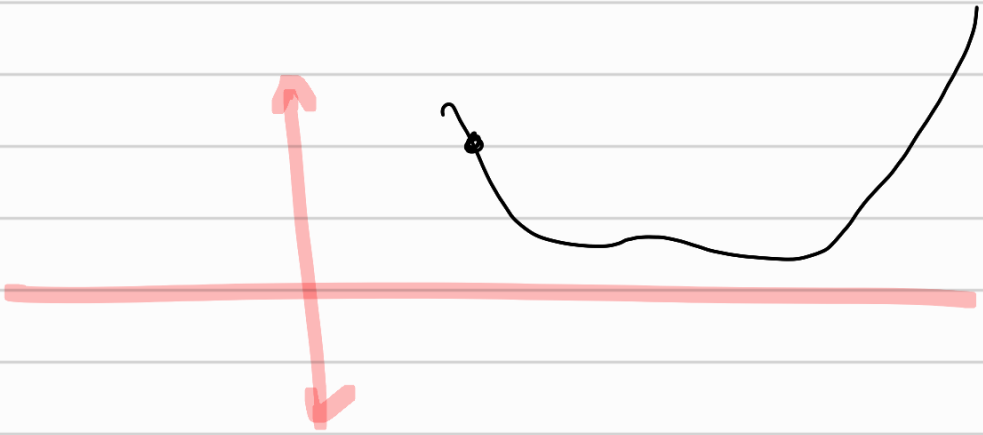
$Th = mse + \lambda_1 (\Sigma w_j^2) + \lambda_2 \Sigma |w_j|$ $L_1 L_2$ Elastic Net

Break!

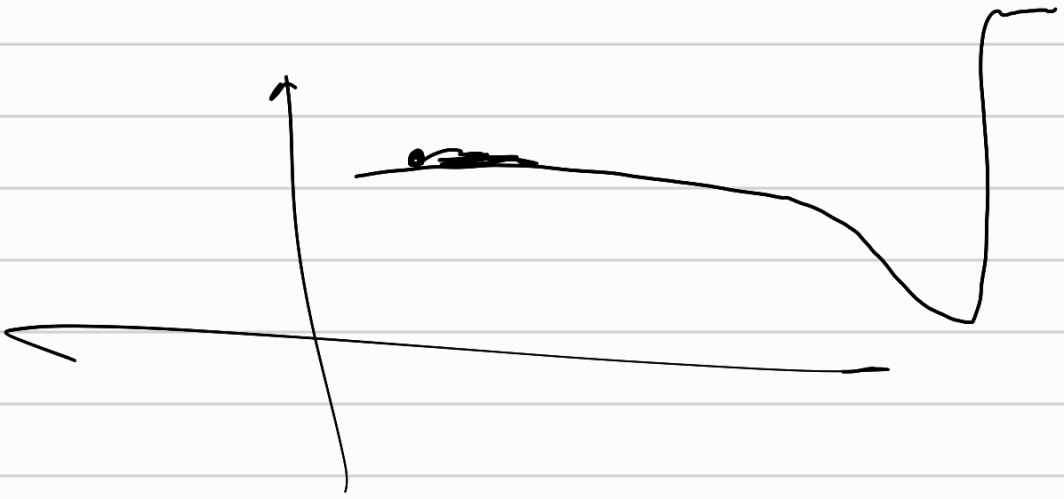
Dropout:



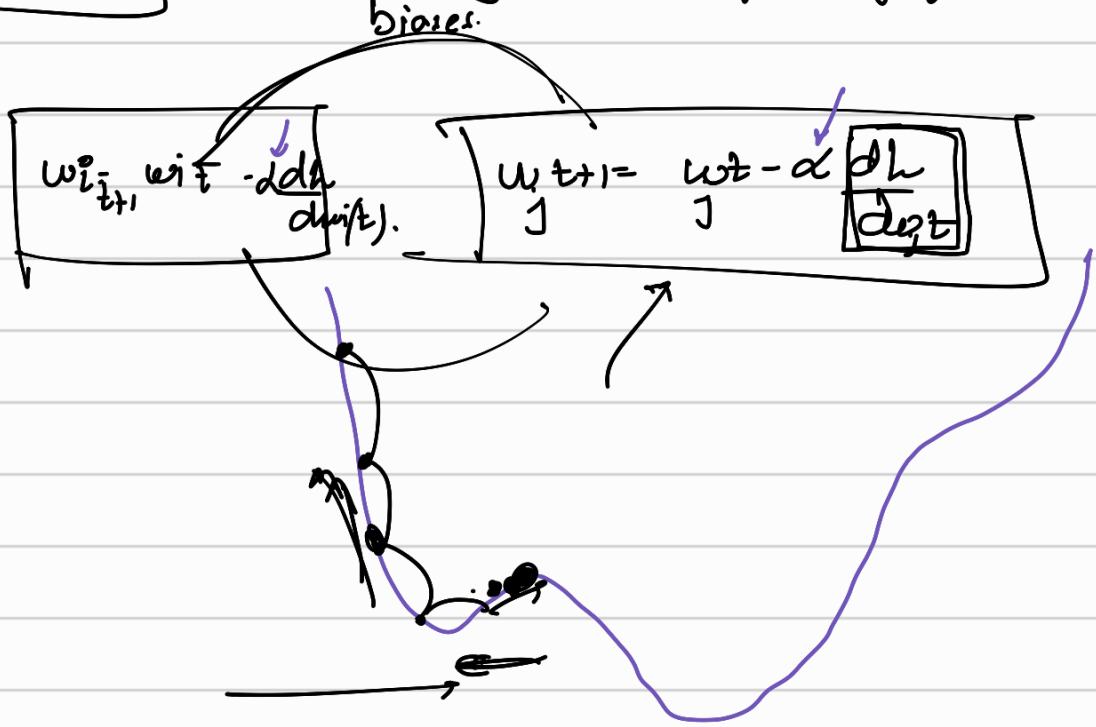


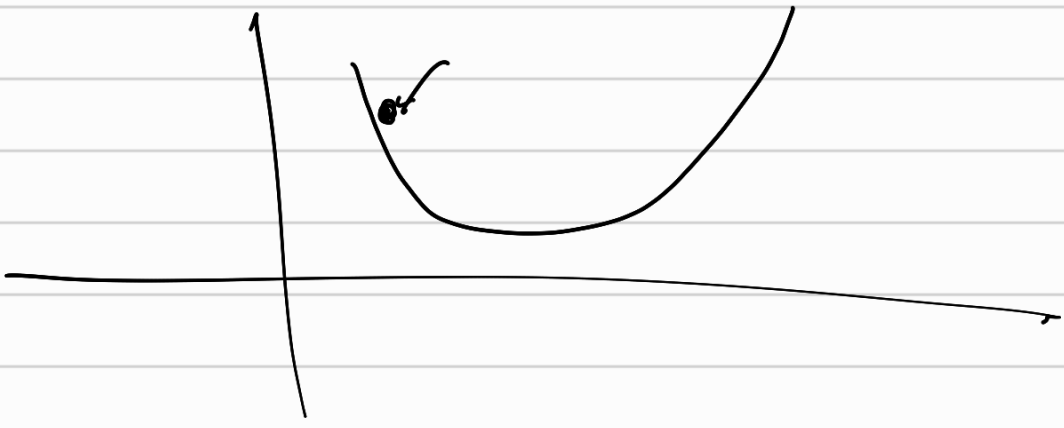


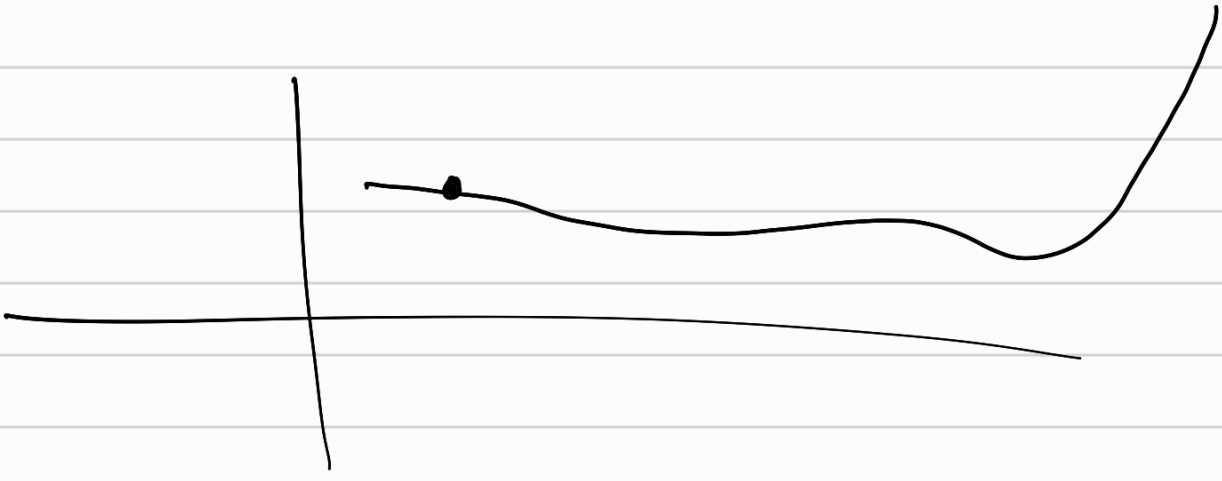
10



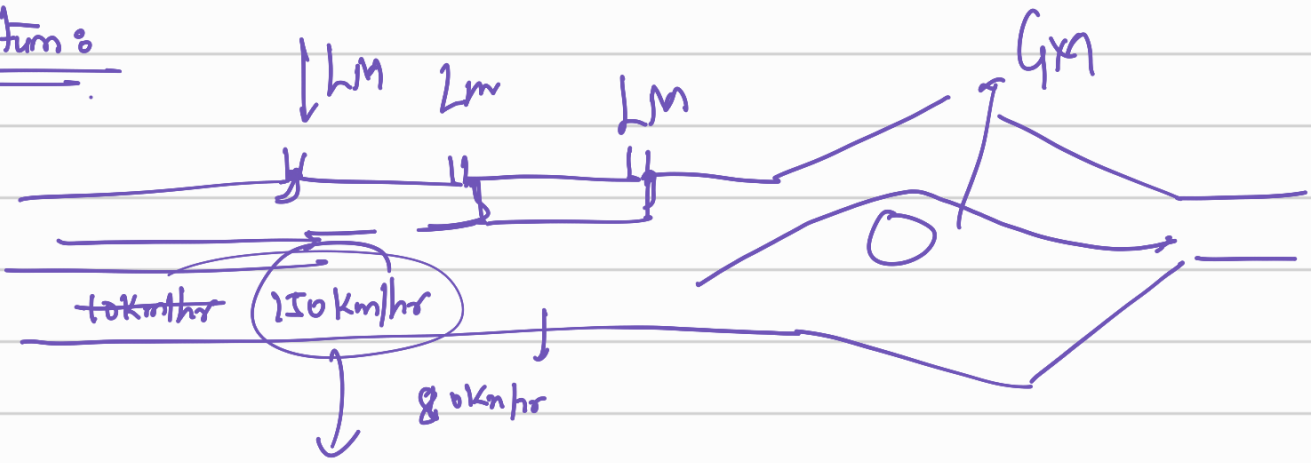
optimizers : minimizing the loss by changing the weights and biases.







Momentum :



momentum

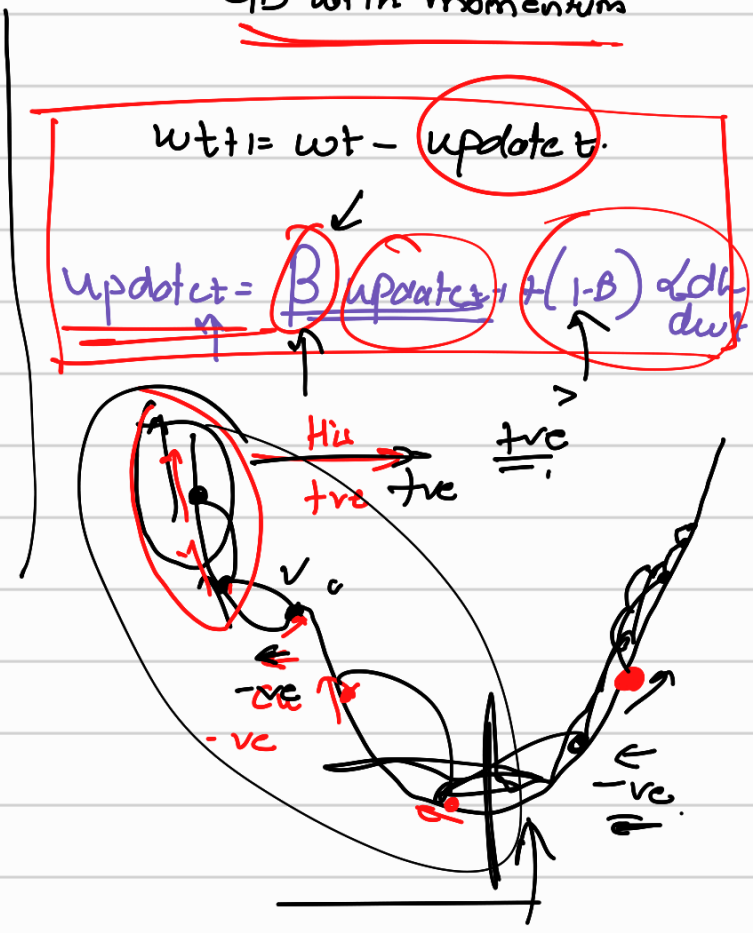
GD

$$w_{t+1} = w_t - \alpha \frac{dh}{dw_t}$$

GD with momentum

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

$$\text{update}_t = \beta\, \text{update}_{t-1} + (1-\beta)\, \alpha \frac{dh}{dw_t}$$



$t \Rightarrow t=1 \;\; t=2. \;\; t=3$

$$\text{update}_1 = (1-\beta)\, \alpha \frac{dh}{dw_2}$$

$$\text{update}_2 = \beta\, \text{update}_1 + (1-\beta)\, \alpha \frac{dh}{dw_2}$$

$$\text{update}_2 = \beta\,(1-\beta)\, \alpha \frac{dh}{dw_1} + (1-\beta)\, \alpha \frac{dh}{dw_2}$$

$\alpha, \beta < 1$

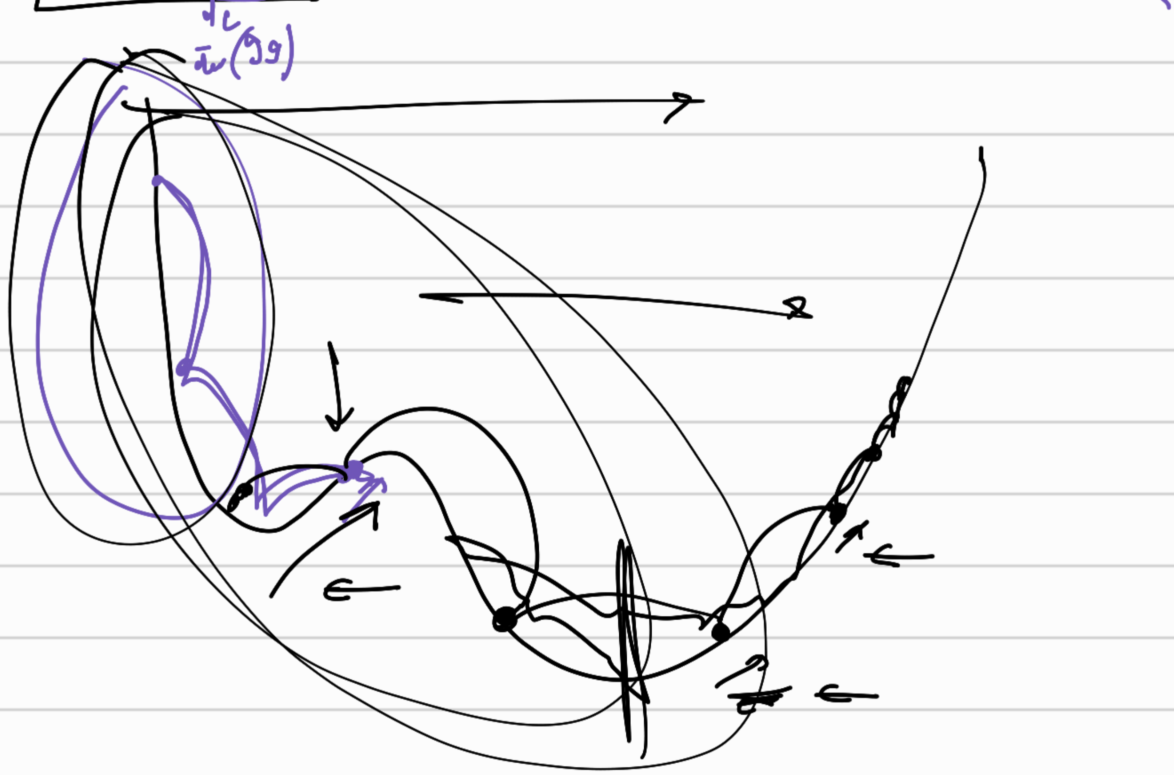
$$update_3 = \beta \, update_2 + (1-\beta)\,\alpha \frac{\partial L}{\partial w_3}$$

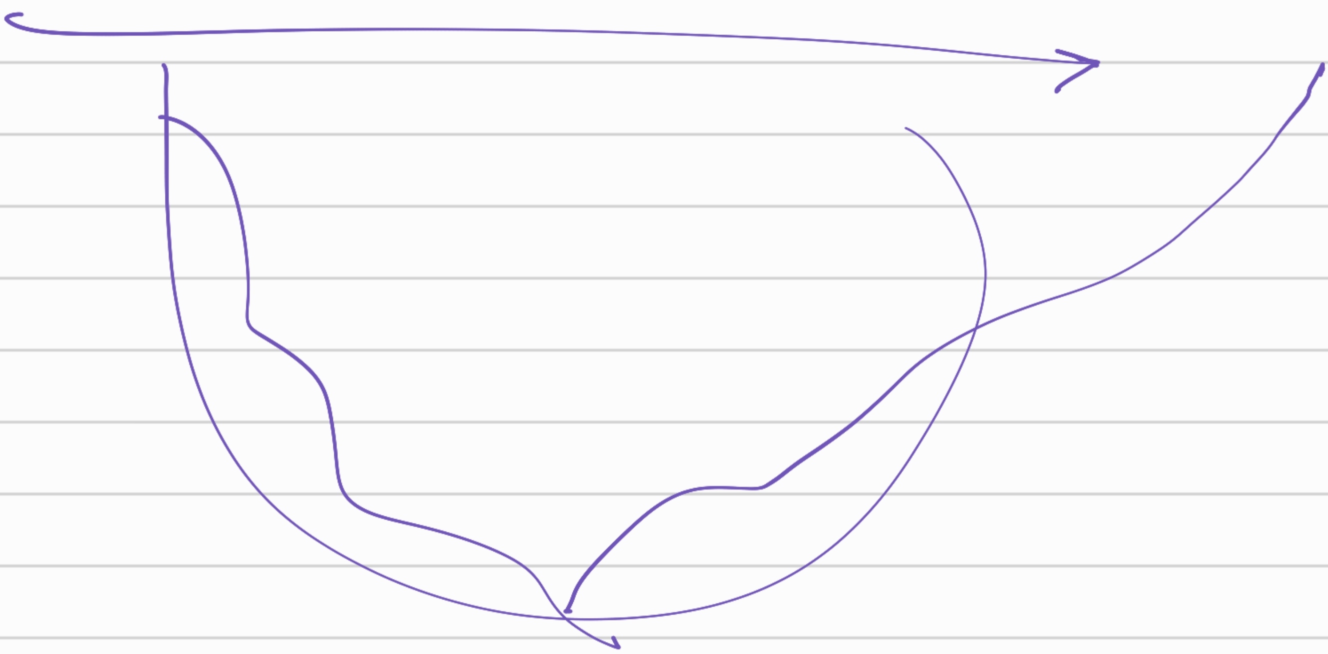
$$update_3 = \beta\left(\beta(1-\beta)\alpha\frac{\partial L}{\partial w_1} + (1-\beta)\alpha\frac{\partial L}{\partial w_2}\right) + (1-\beta)\alpha\frac{\partial L}{\partial w_3}$$

β^{100}

$$update_3 = \beta^2(1-\beta)\alpha\frac{\partial L}{\partial w_1} + \beta(1-\beta)\alpha\frac{\partial L}{\partial w_2} + (1-\beta)\alpha\frac{\partial L}{\partial w_3}$$

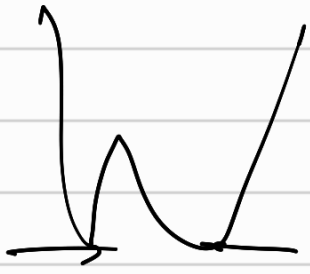
$B < 1$

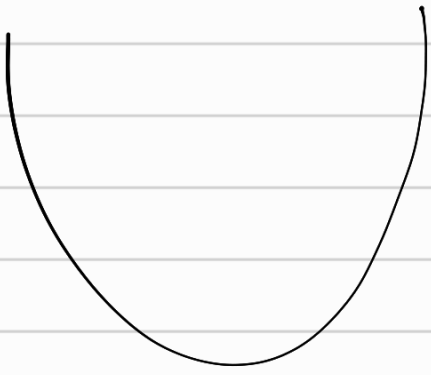


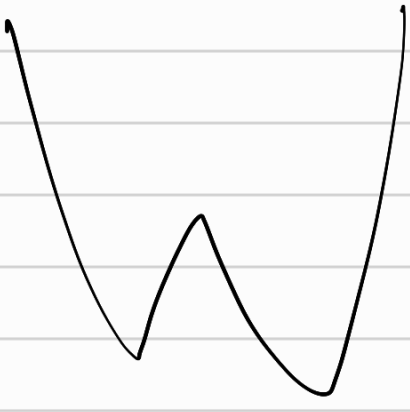


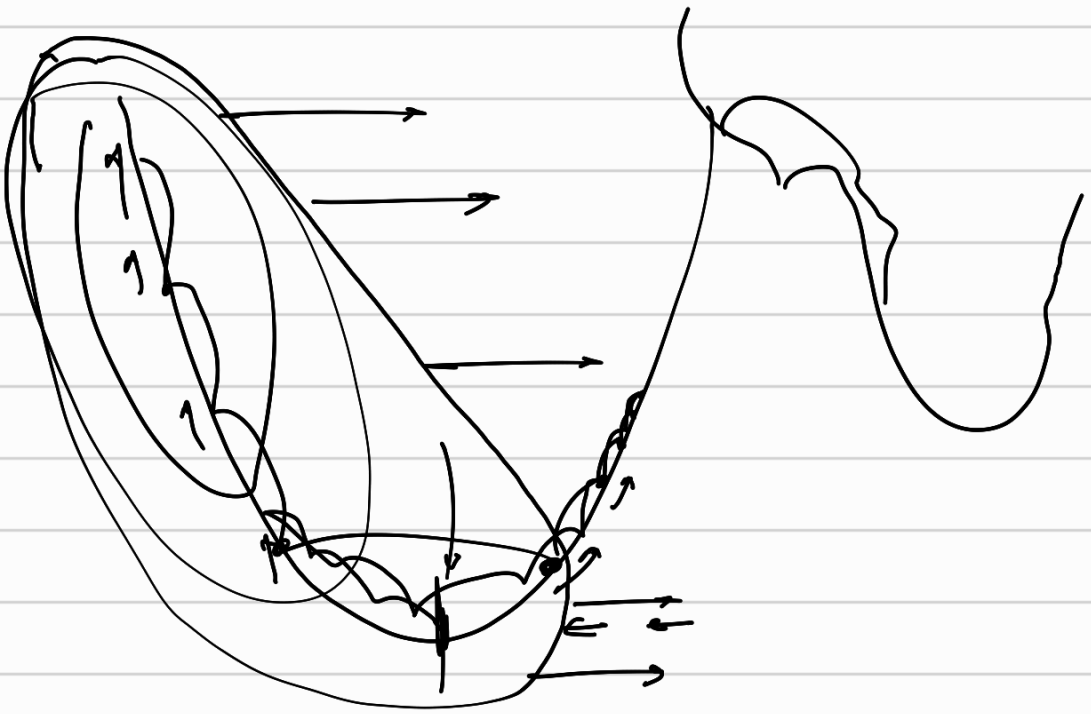
$i = i + 1$

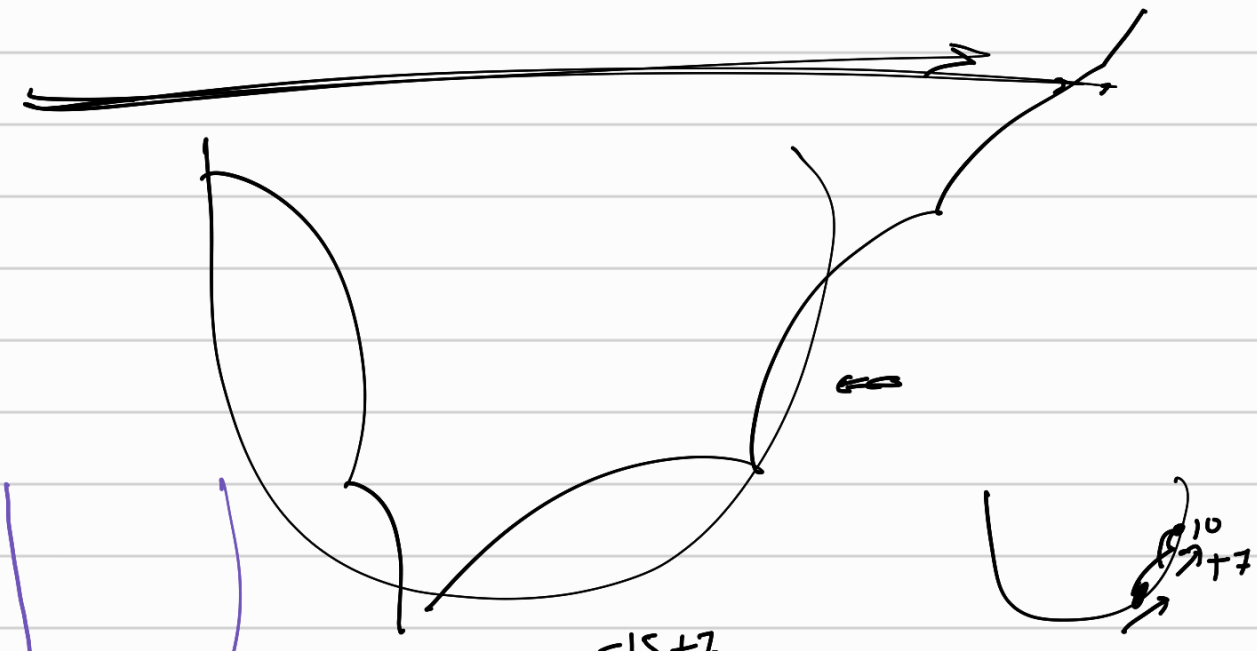
끝











$-15+7$

$-20 \rightarrow 20+5$

$-ve$

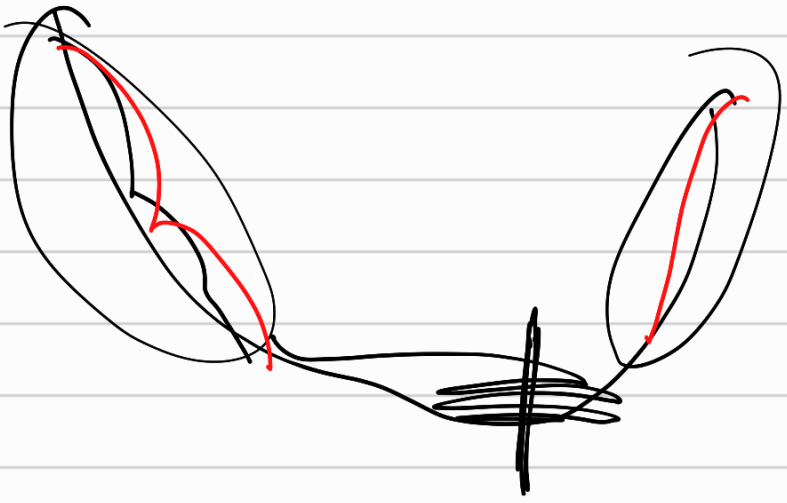
$$\text{update}_t = \text{update}_{t-1} + \alpha \frac{dh}{dwt}$$

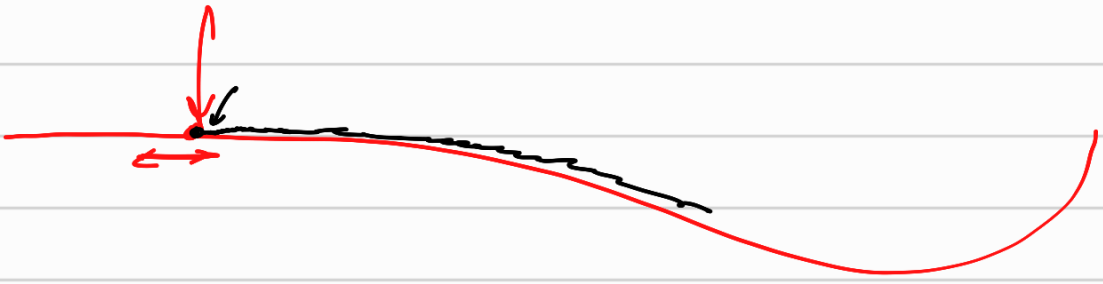
1
2
3

$-8 \quad +10$

$= -15$

6th





$a_0 = 0$

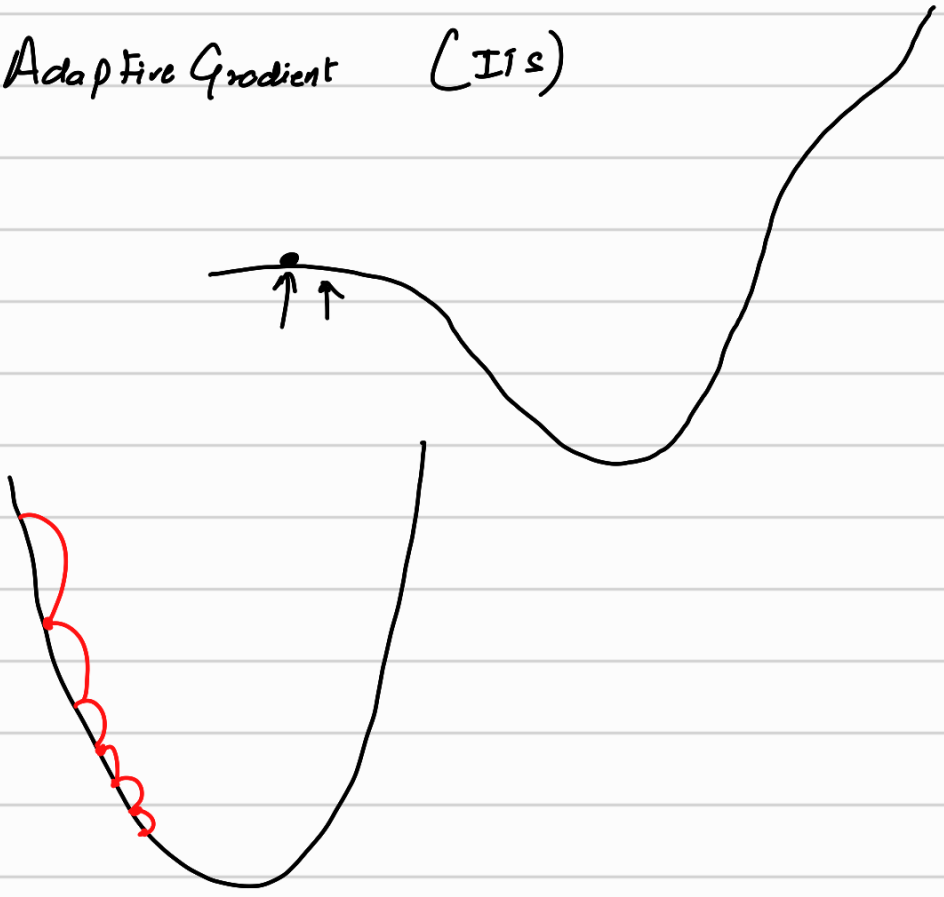
$a_t = a_{t-1} + 5$

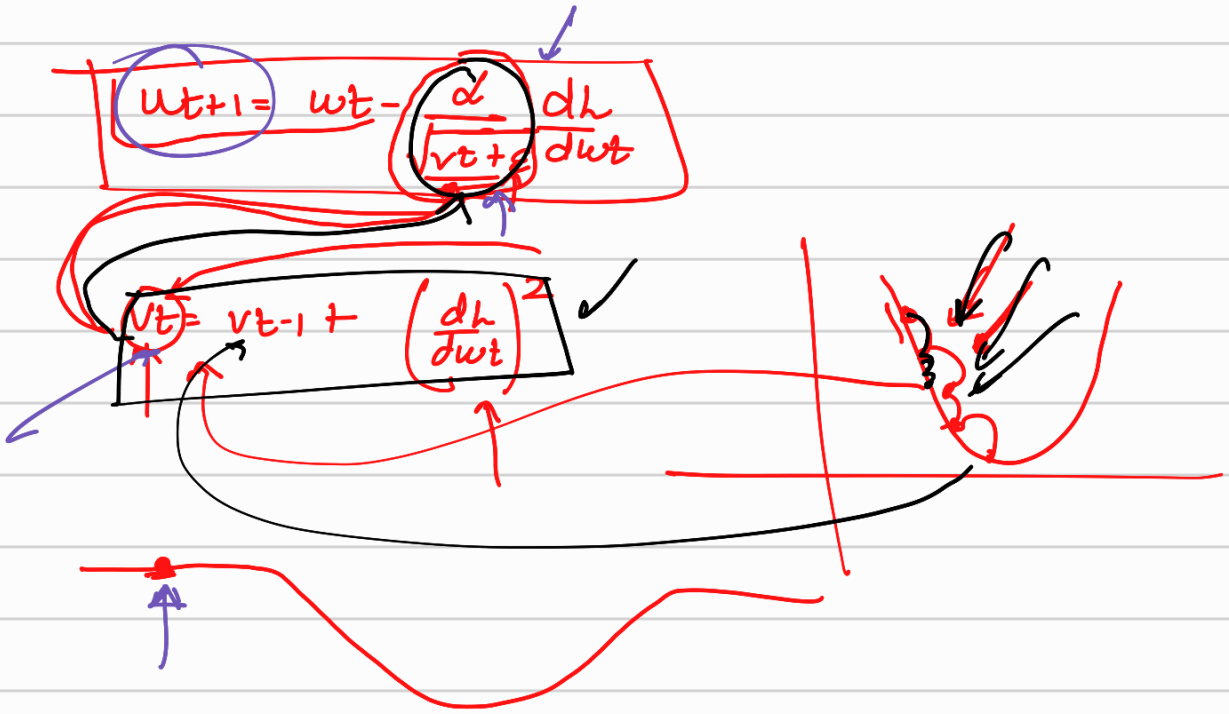
$a_1 = 0 + 5$ $\quad a_2 = a_1 + 5$ $\quad a_3 = a_2 + 5$

$a_1 = 5$ $\quad a_2 = 10$ $\quad a_3 = (a_1 + 5) + 5$

Break!

Adagrad: Adaptive Gradient (IIs)





Rms Prop

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

$$v_t = \beta \, v_{t-1} + (1-\beta) \times \left(\frac{\partial h}{\partial w_t}\right)^2$$

$\beta = 0.99$, $(1-\beta) = 0.01$

Adam:

Adaptive moments →

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

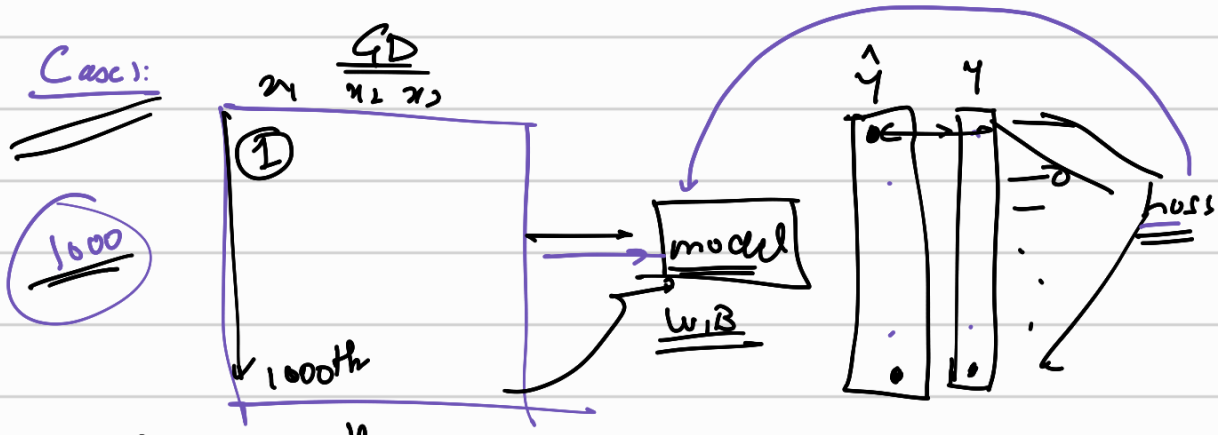
$$v_t = \beta_2 (v_{t-1}) + (1-\beta_2) \times \left(\frac{\partial h}{\partial w_t}\right)^2$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \times m_t$$

Mini Batch Gradient Decent → Batch Normalization

Callbacks

Case1:



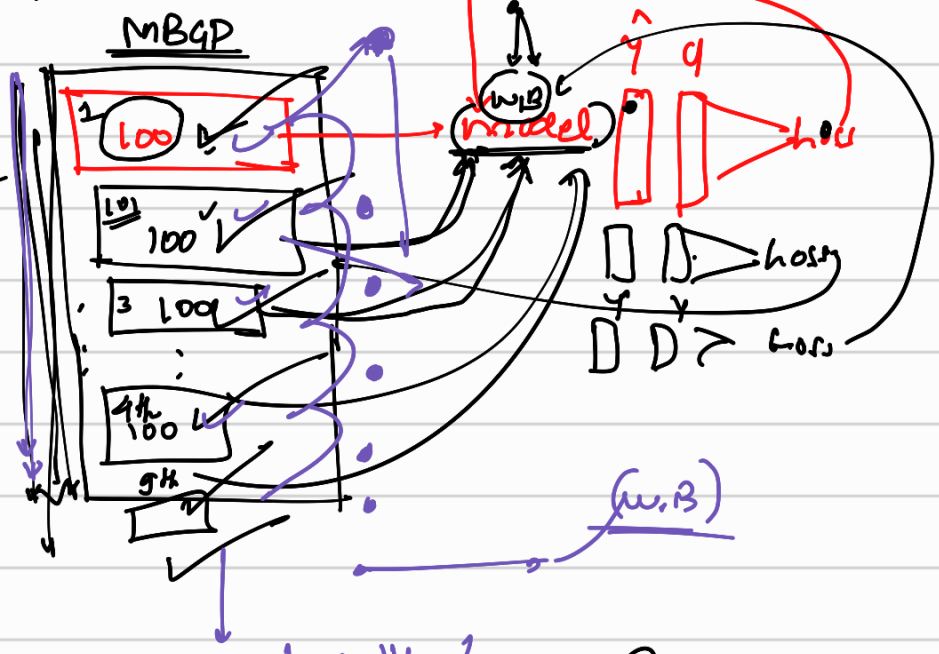
1 epoch: FP, BP, on the 1000 rows

Case2:

batch-size: 100

1000

10 times

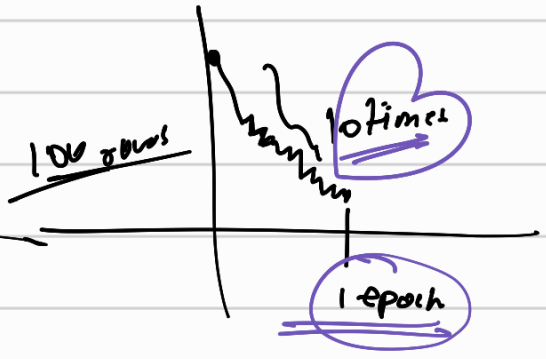


(W,B)

randomly without replacement

Case1

Case2



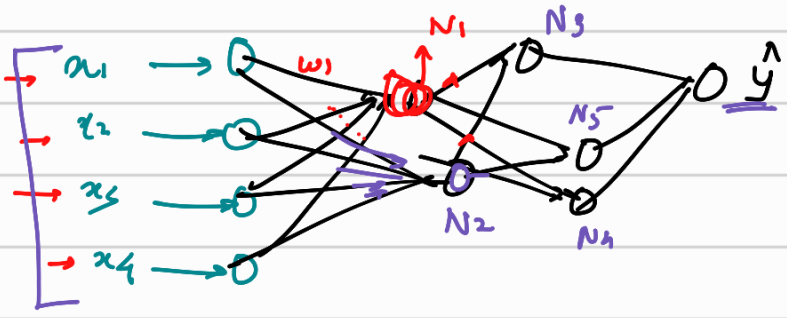
model.fit (10)

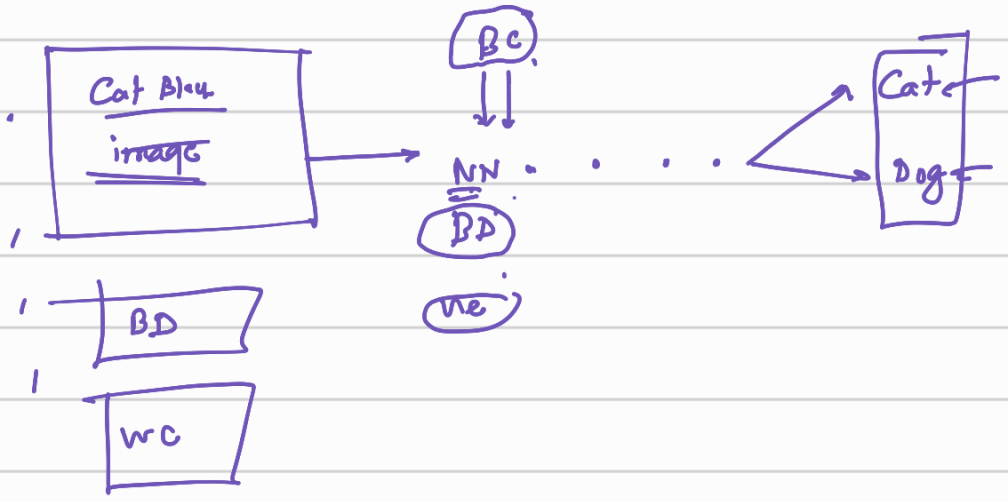
model.fit (20)

Batch Normalization

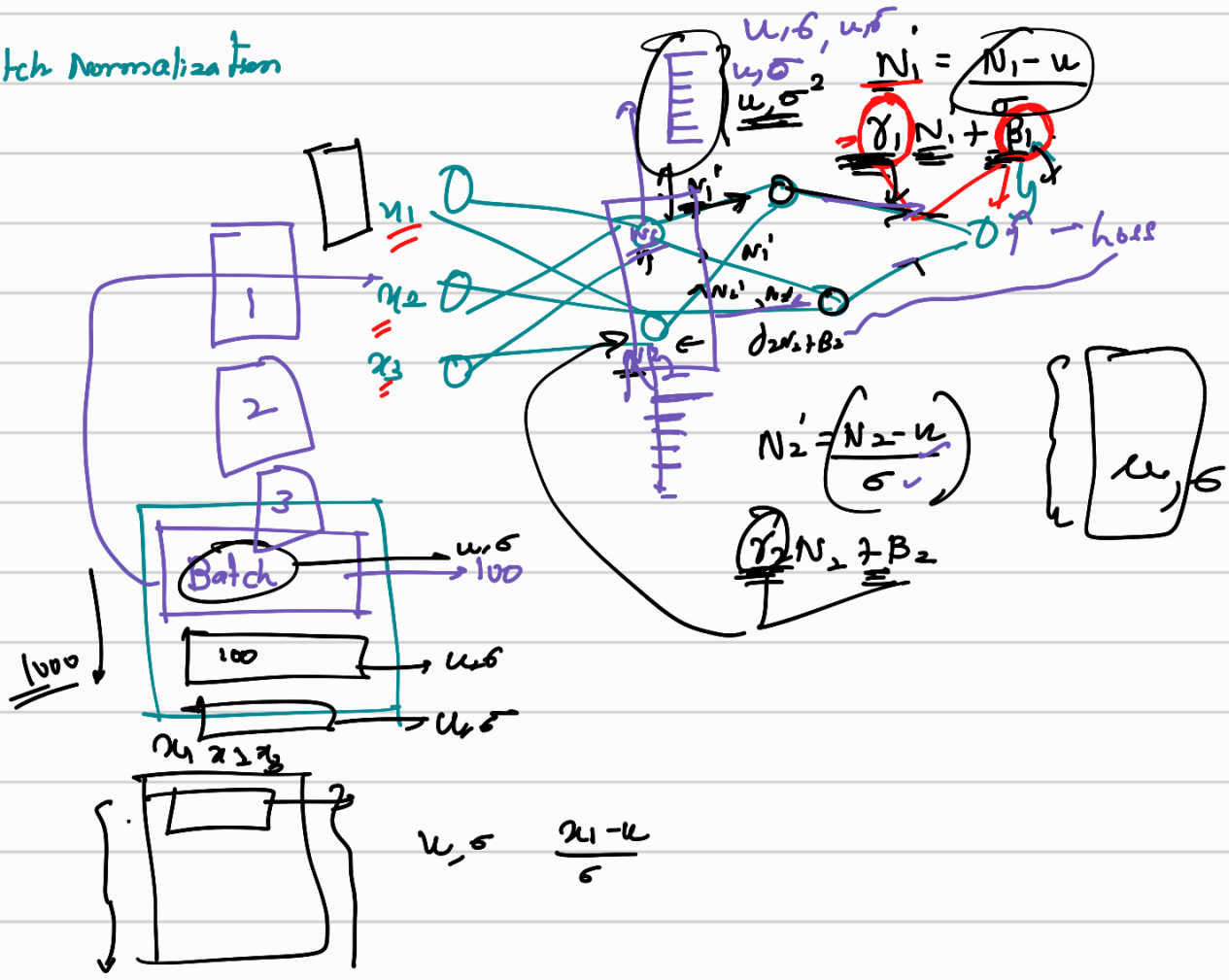
Callbacks

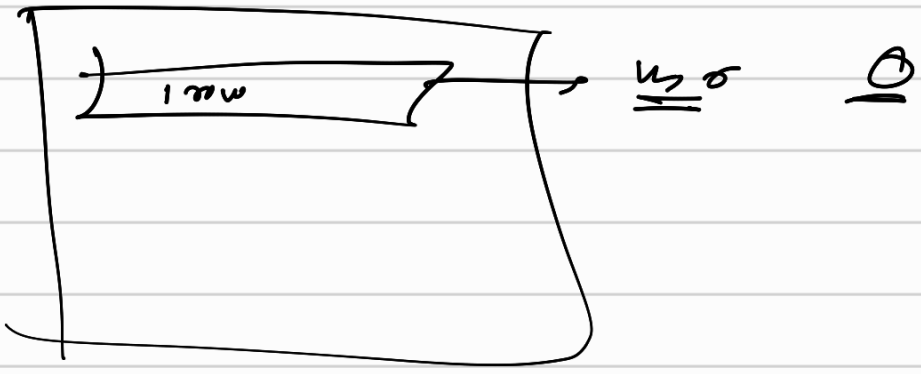
Batch Normalization

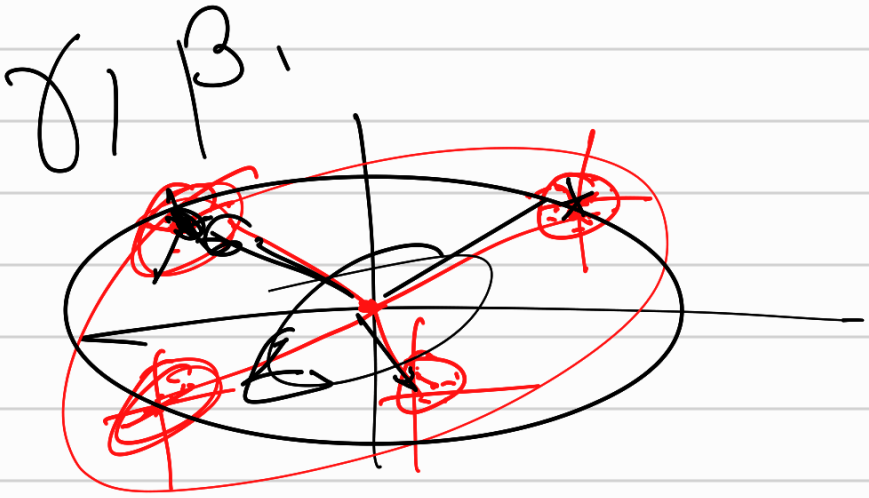


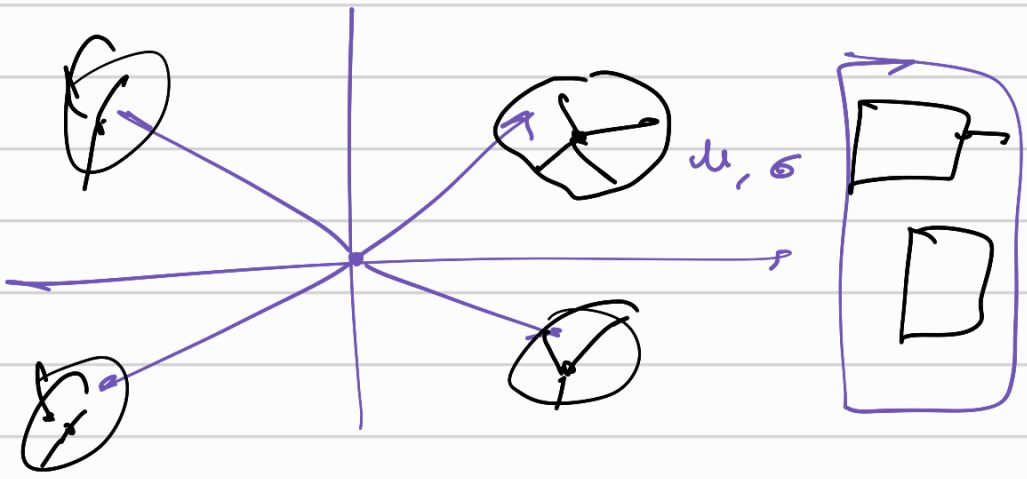


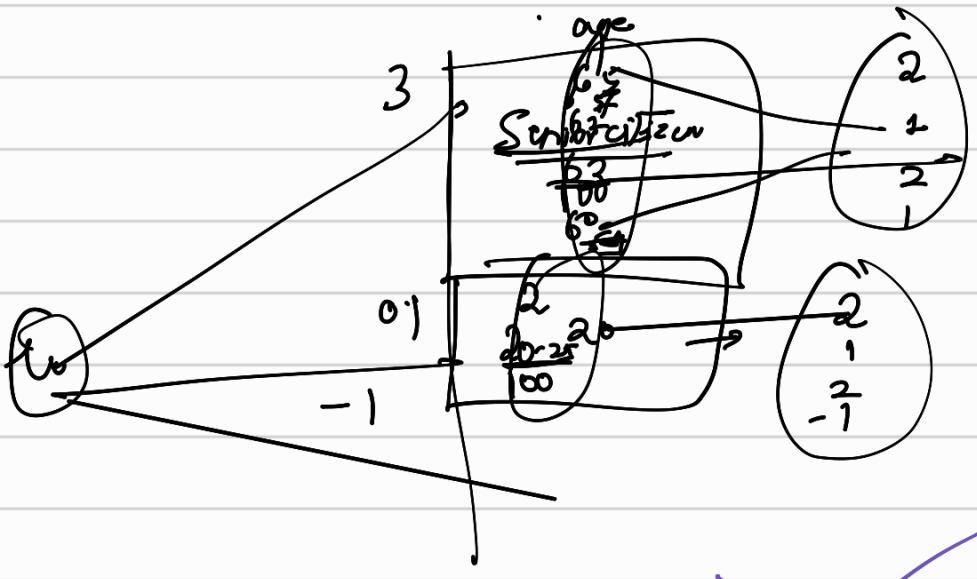
Batch Normalization

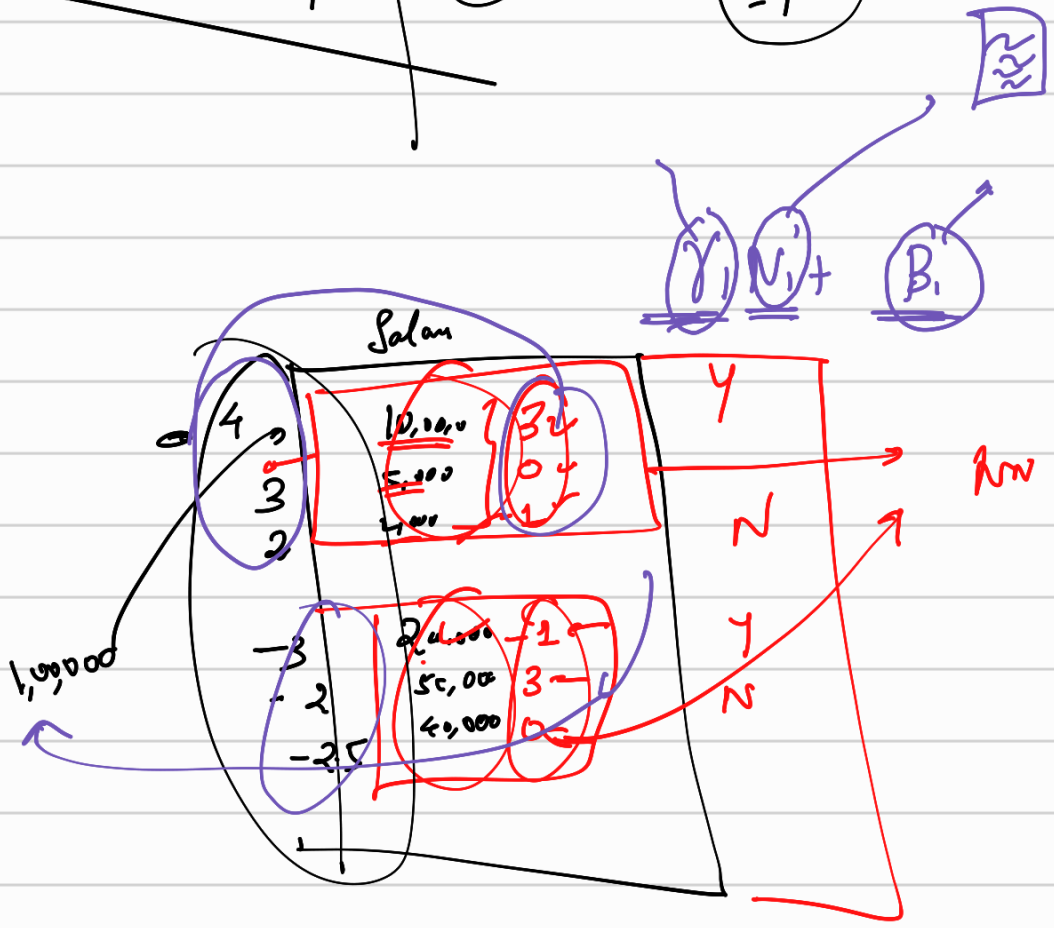


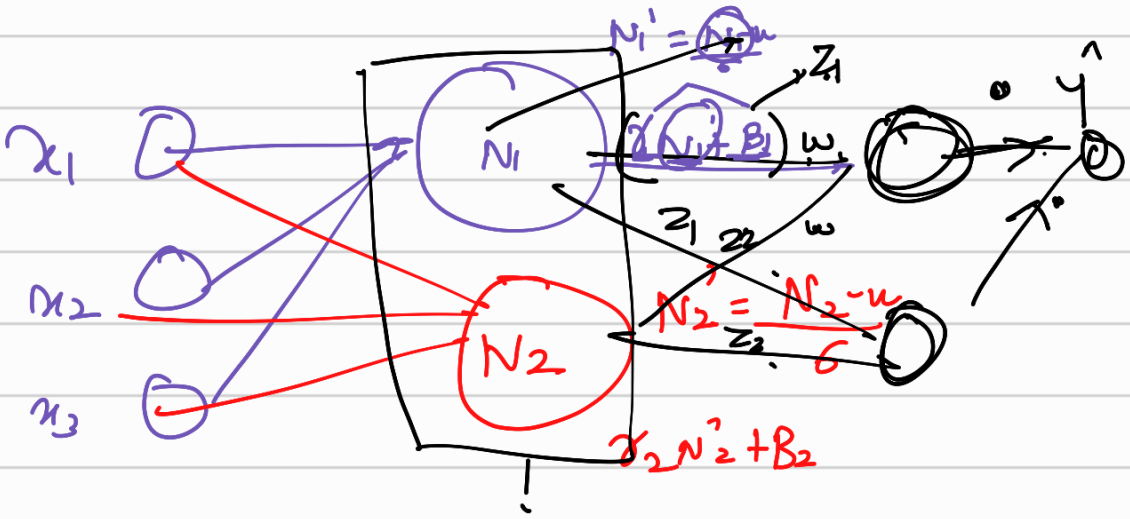




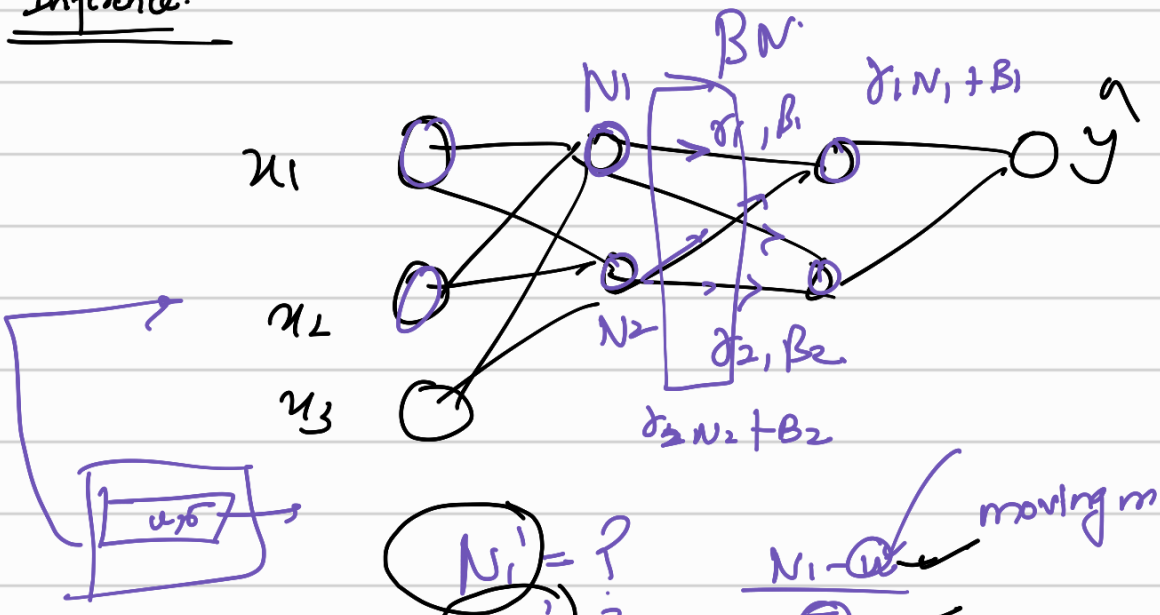

μ, σ

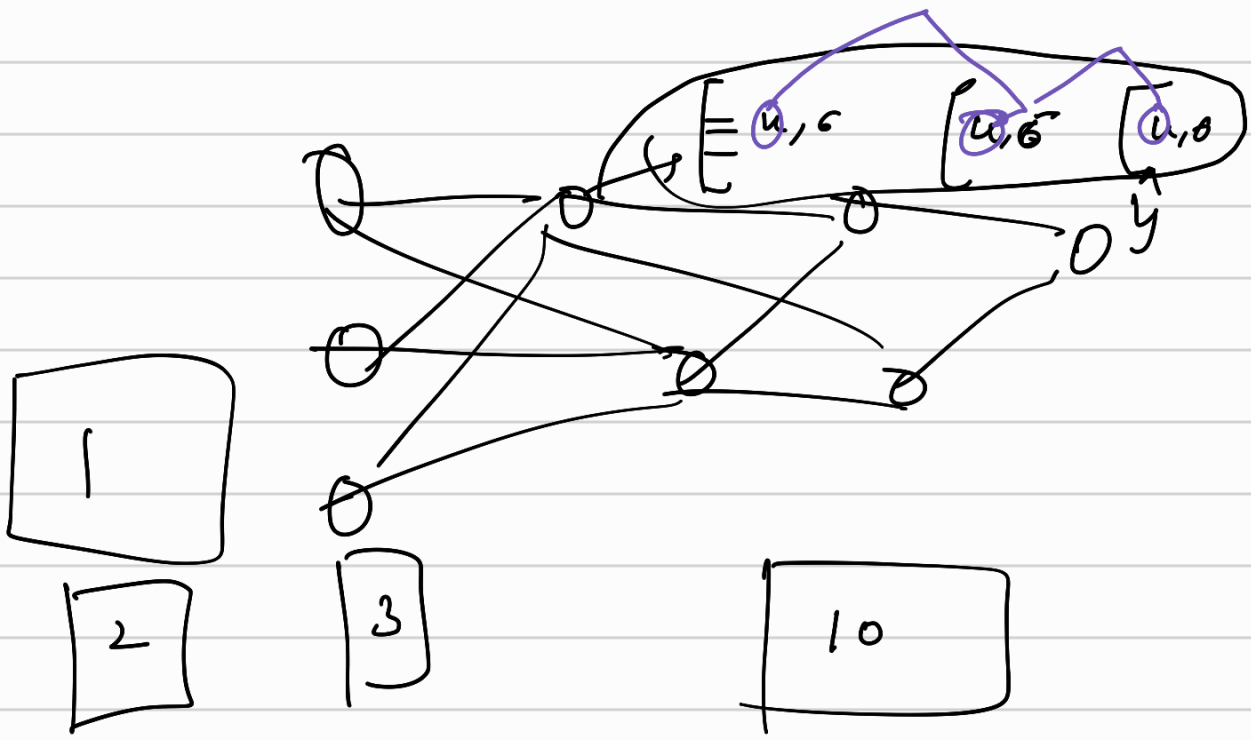

age
3
Senior citizen
2
1
2
1
0.1
2
20
2
1
2
-1
-1


Salary
4
3
2
-3
-2
-2.5
1,00,000
Y
N
Y
N
Any


x_1
x_2
x_3
N_1
N_2
$N_1' = \frac{N_1 - \mu}{\sigma}$
$\gamma_1 N_1' + B_1$
$N_2' = \frac{N_2 - \mu}{\sigma}$
$\gamma_2 N_2' + B_2$
z_1
z_2
z_3
ω
y

Inference:


BN
N_1
N_2
γ_1, B_1
γ_2, B_2
$\gamma_1 N_1 + B_1$
$\gamma_2 N_2 + B_2$
x_1
x_2
x_3
y
μ, σ
$N_1' = ?$
$N_2' = ?$
$\frac{N_1 - \mu}{\sigma}$
moving mean (-bair)
moving var (bair)

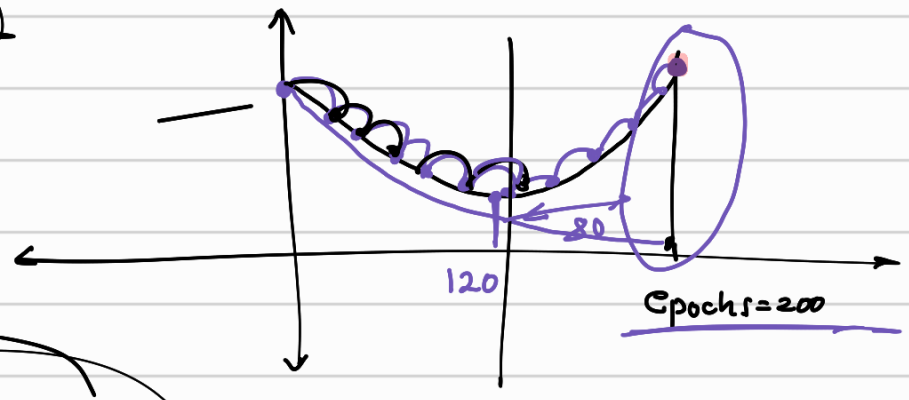

μ, σ
μ, σ
$\mu, 0$
y
1
2
3
10

N1

moving mean → $\beta \times$ movingmean + $(1-\beta) \times$ mean(batch)

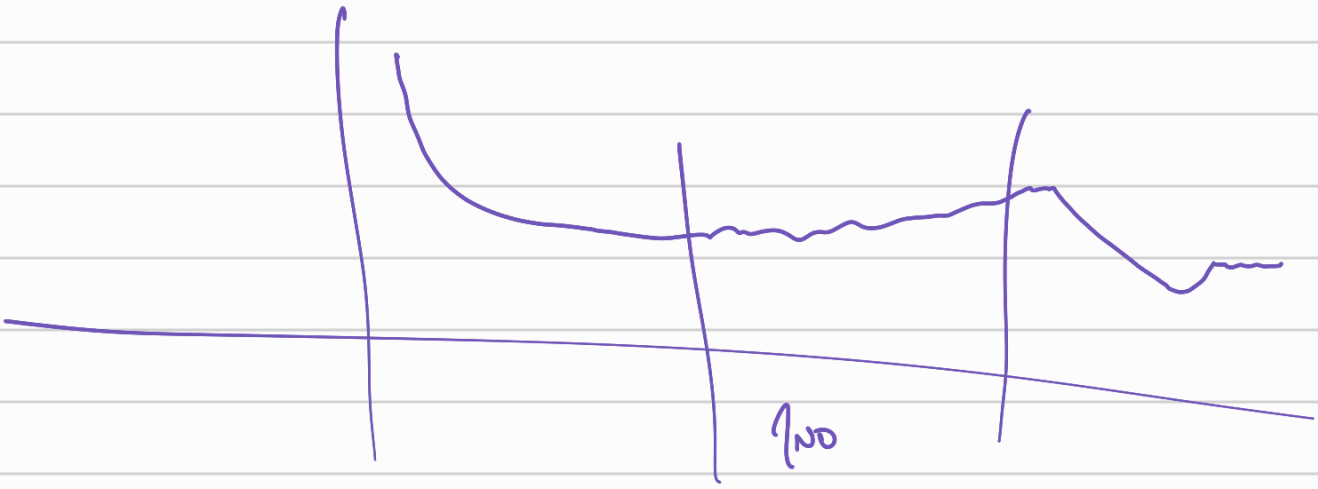
N2 movingvar = $\beta \times$ movingvar + $(1-\beta)$ var(batch)

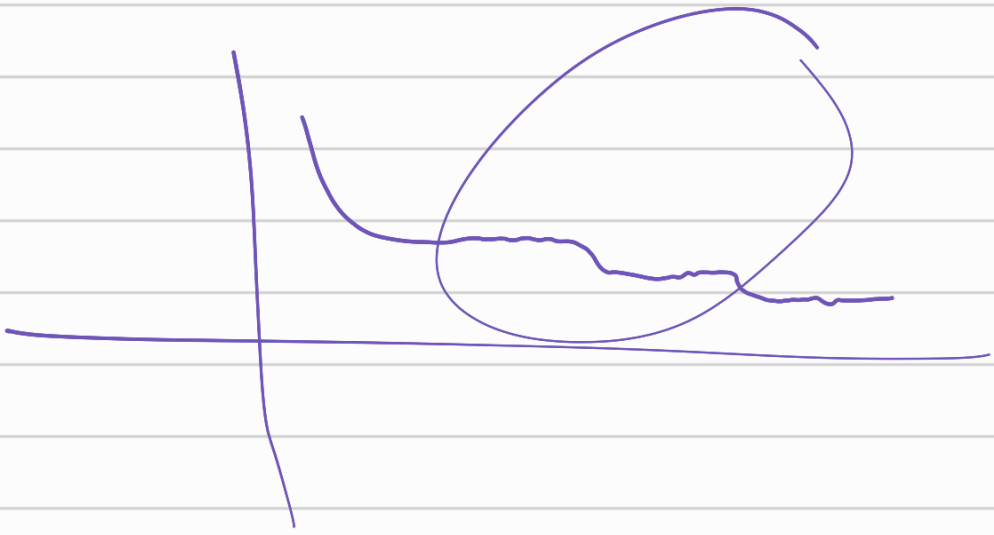
Callbacks



model Checkpoint

Early Stopping





max_trials = 3

Executions_per_trial: 2

[3, 32, relu, d=T, lr=10^{-2}]

[4, 32, relu, d=N, lr=10^{-4}]

[4, 64, tanh, d=T, lr=10^{-3}]

model → fit model. 100

model → fit 100

11:33 pm

Input layer (784)

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

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

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

Batch Norm()

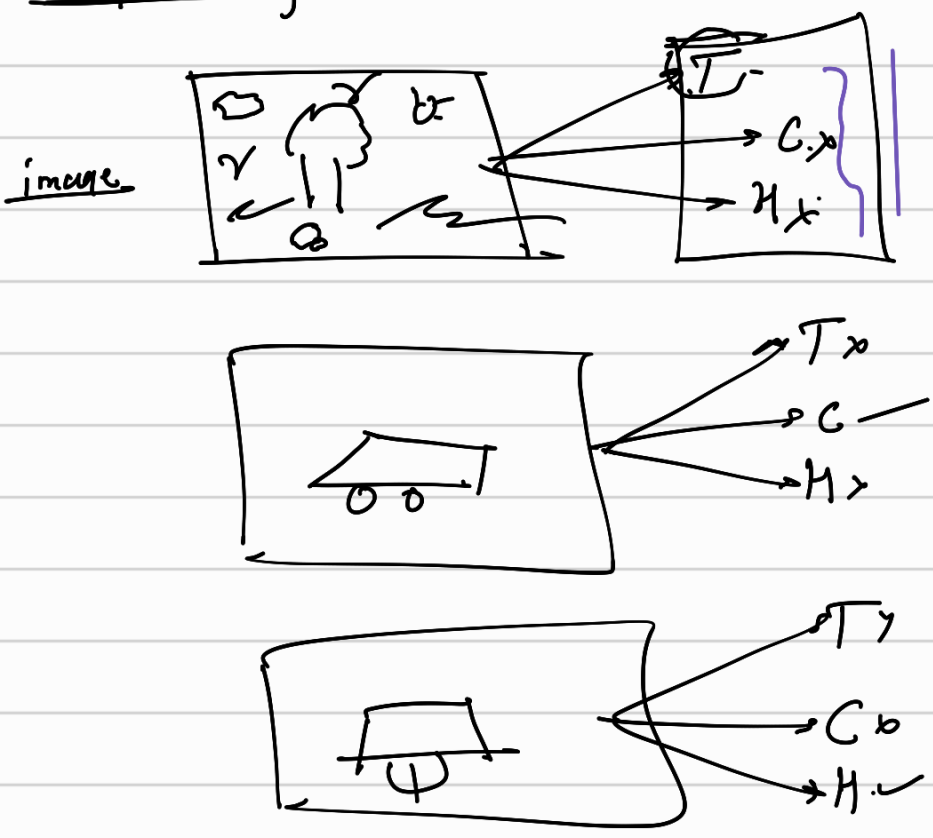
Batch Norm() ← maybe add / not

model. layers (Dense (10, softmax).

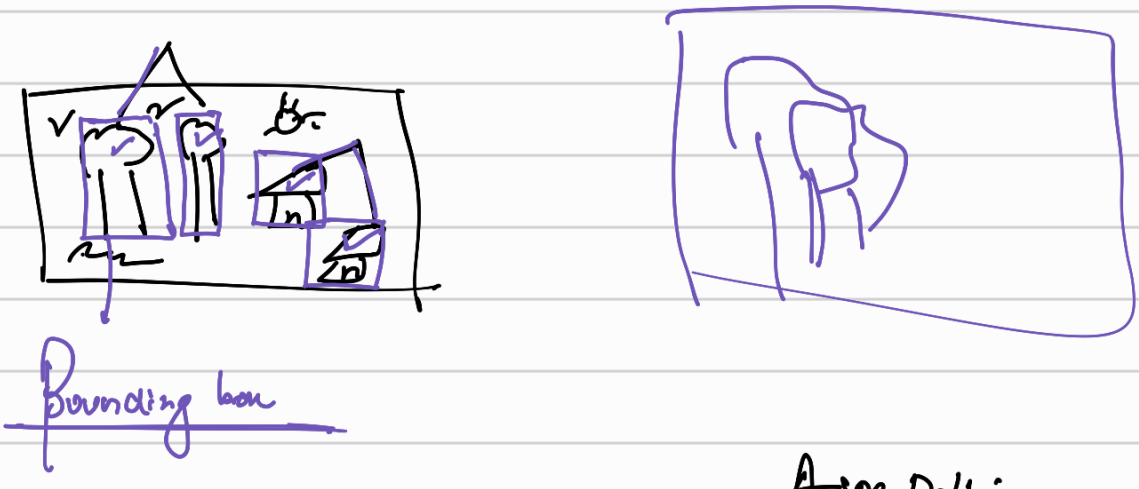
log sampling

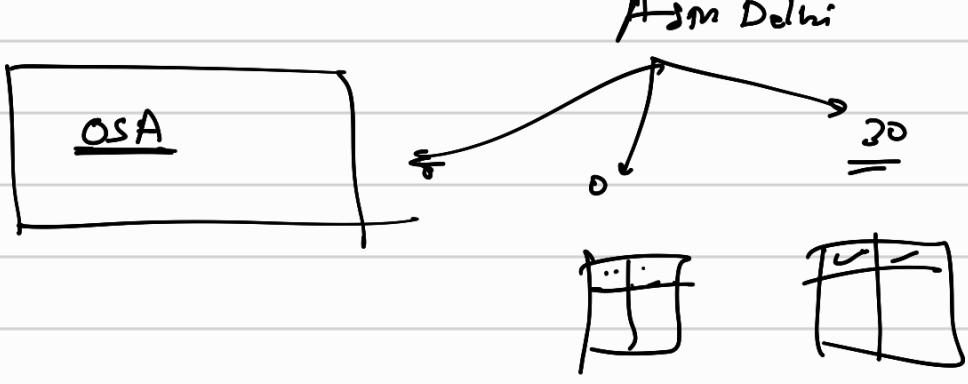
Introduction to Computer Vision

① Image Classification



Object detection & localization





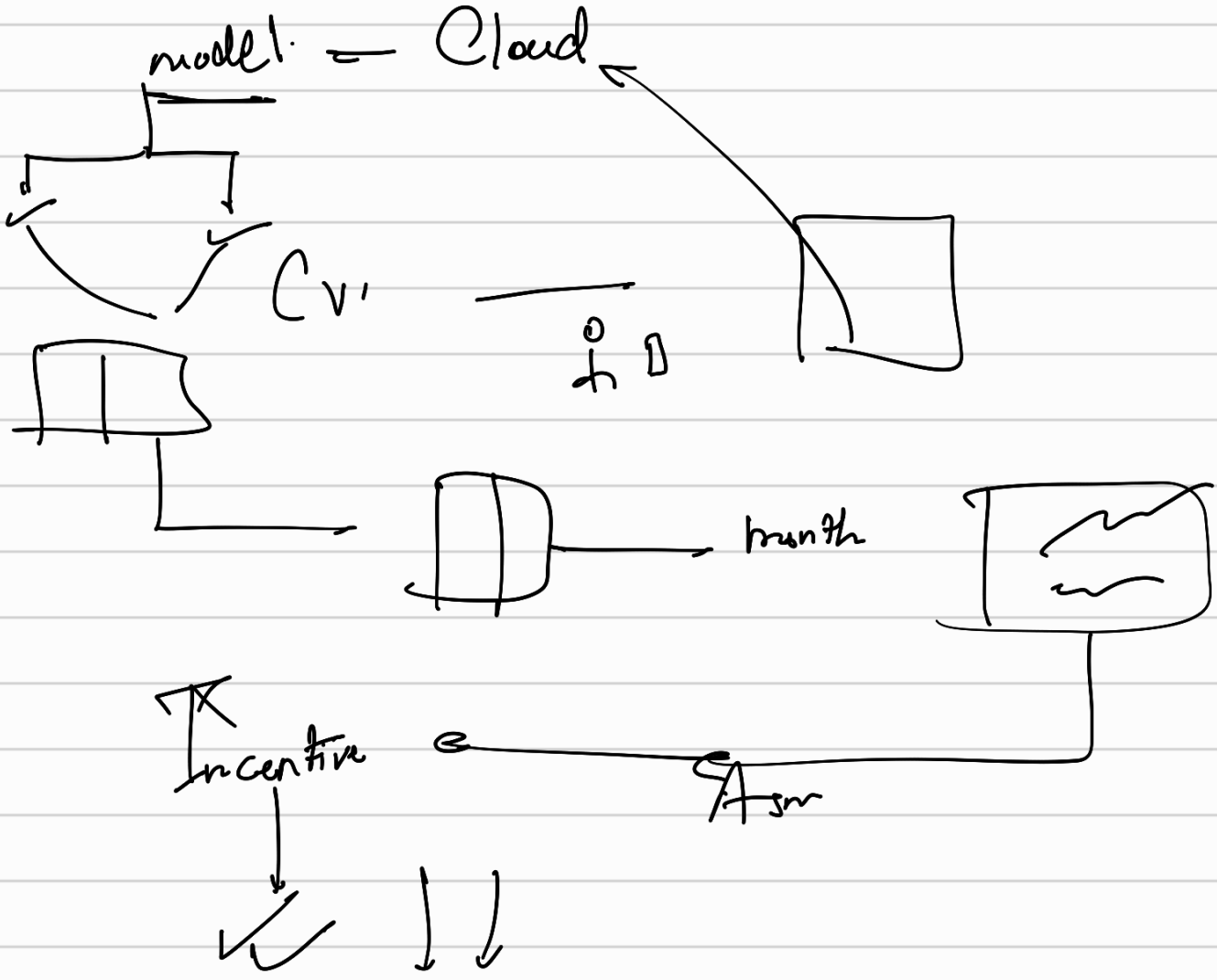
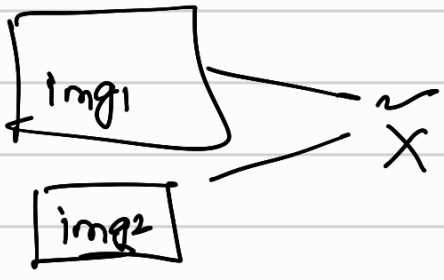


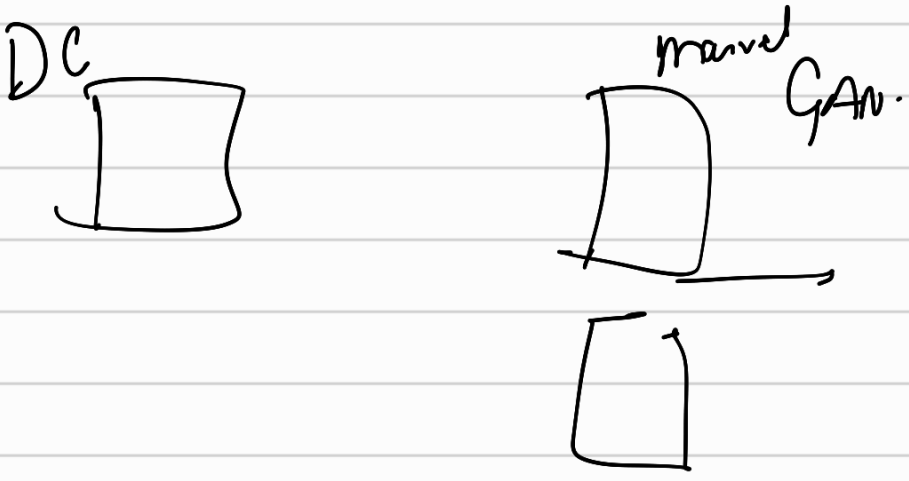
Image Segmentation

④ Framework

FR



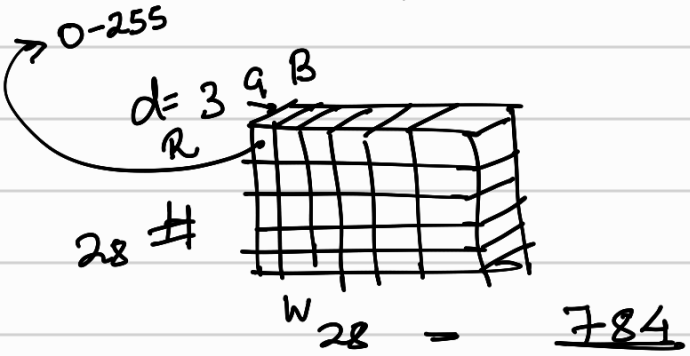
⑤ GAN → Gen Adv Networks.

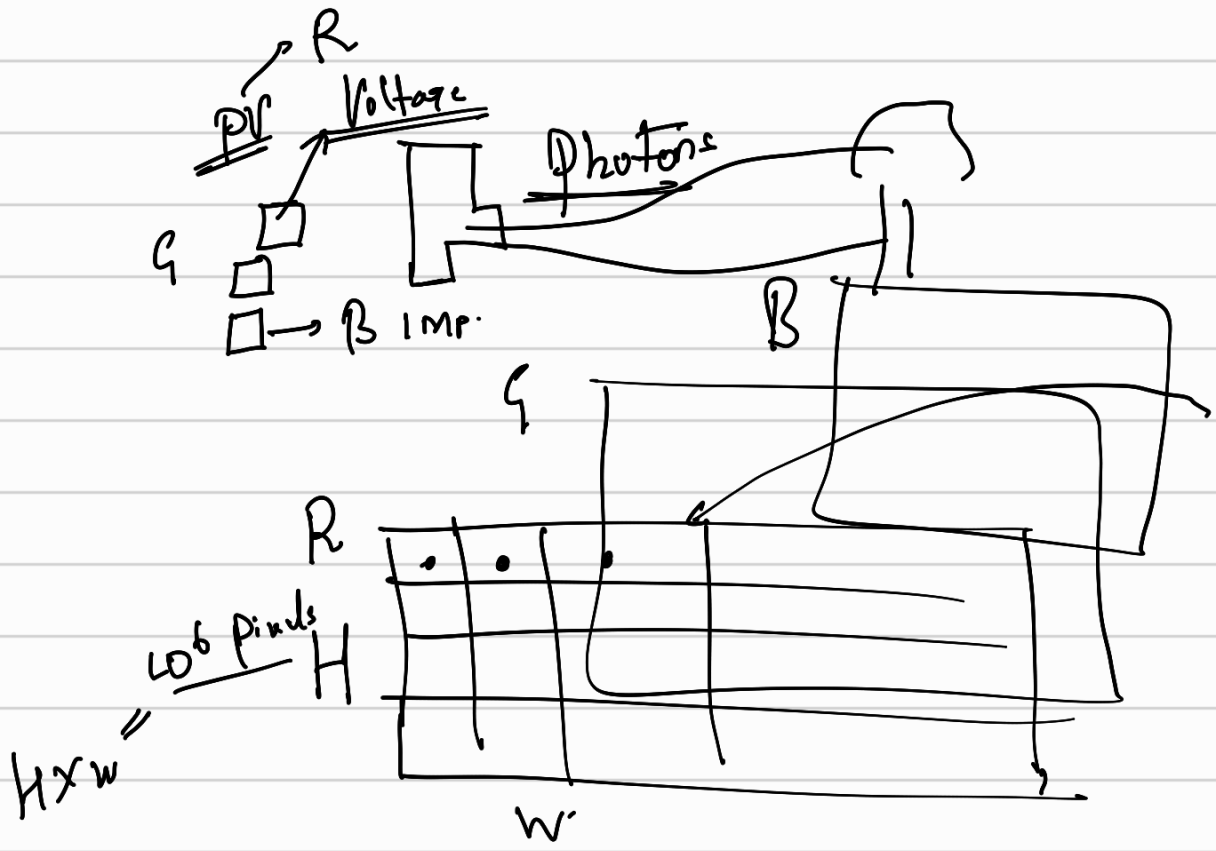


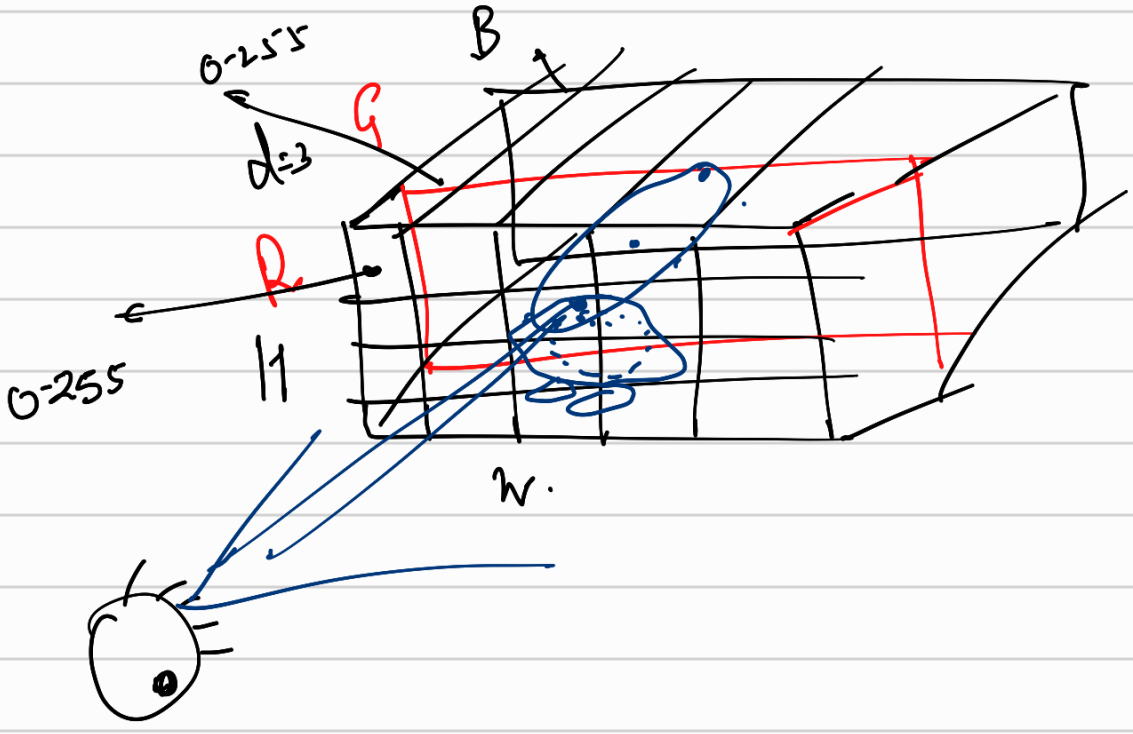
CNN

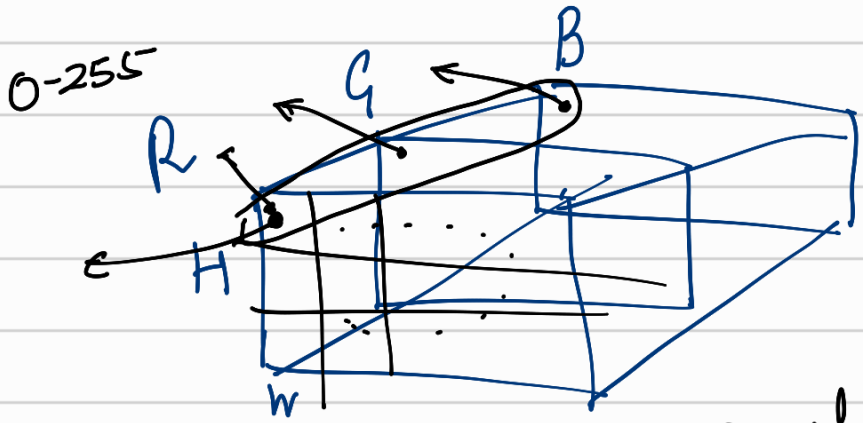
What is an image:

R, G, B

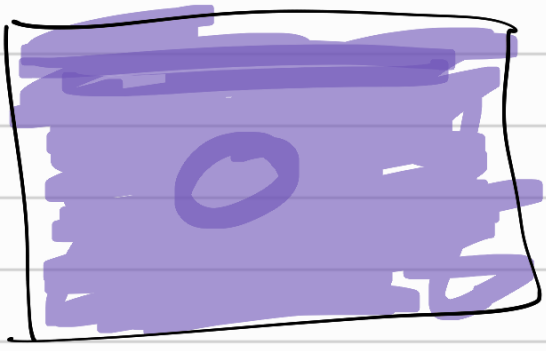


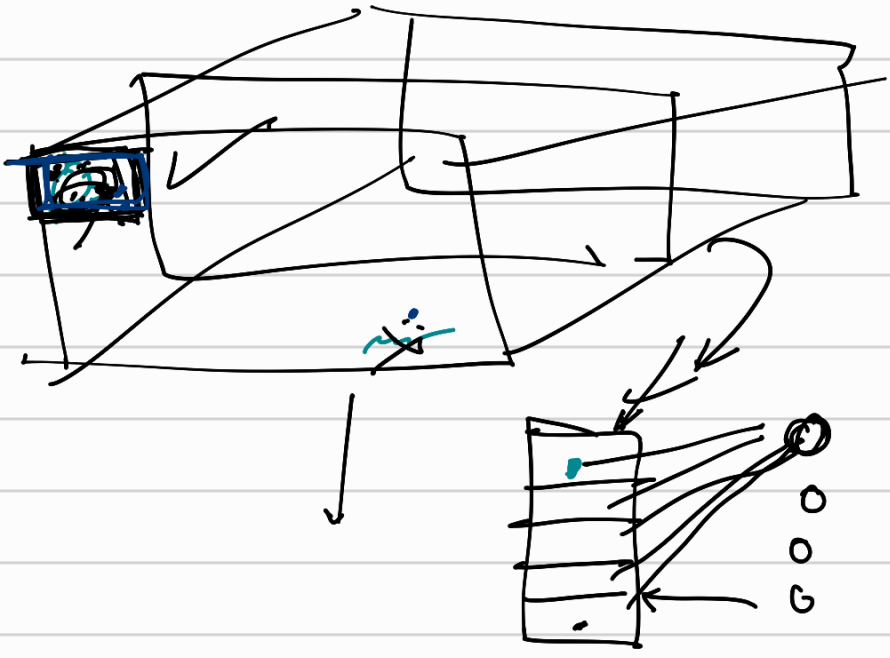

R
PV
Voltage
Photons
G
B Imp.
B
G
R
10^6 Pixels
H
H x W
W

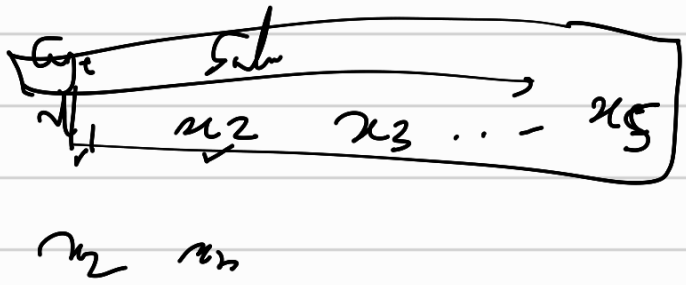

0-255
B
G
d=3
R
0-255
H
W

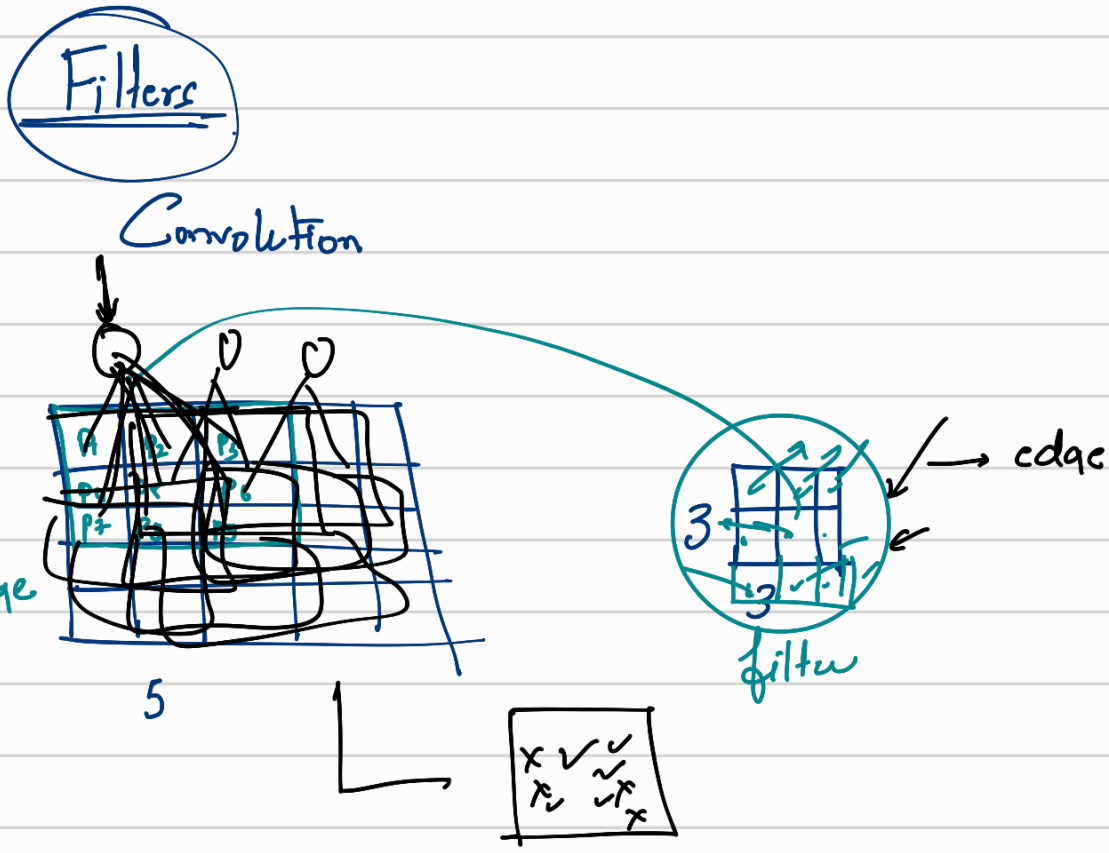


Intensity — Variation in the pixel intensity

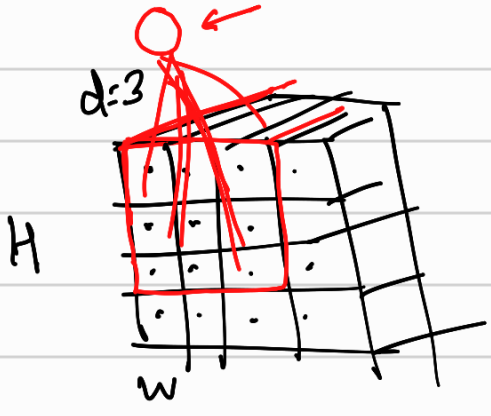


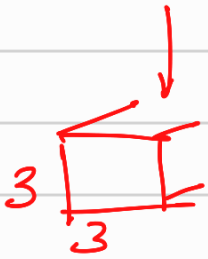


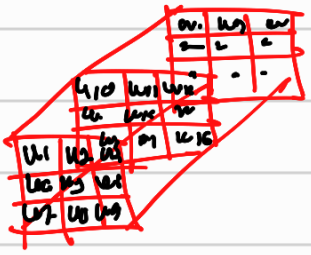




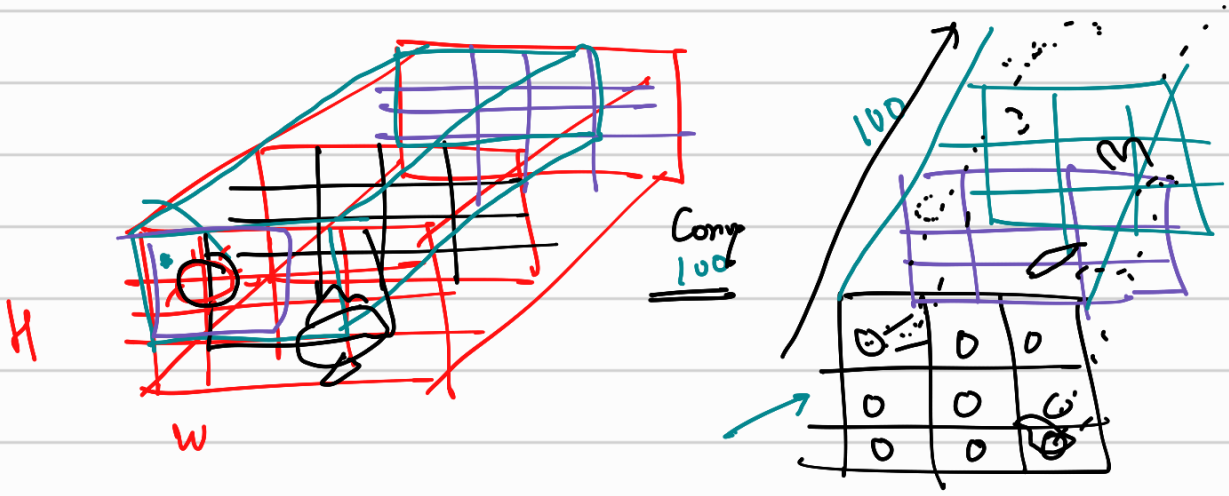
Convolutional Neural Network

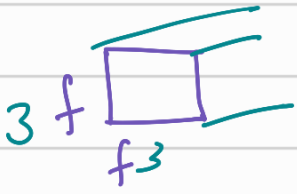




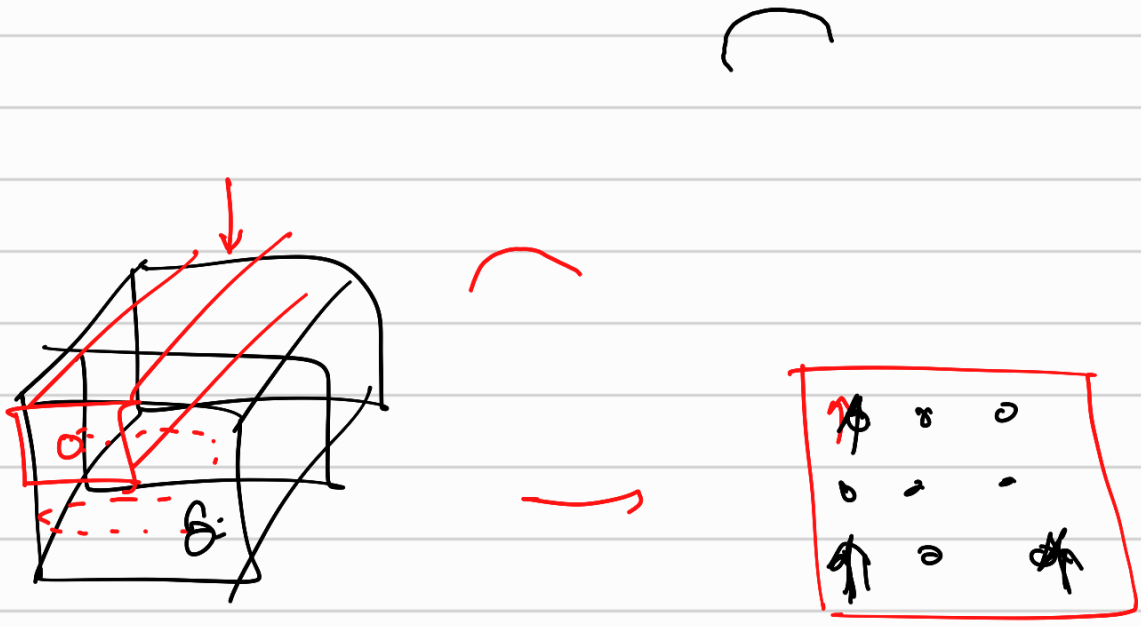


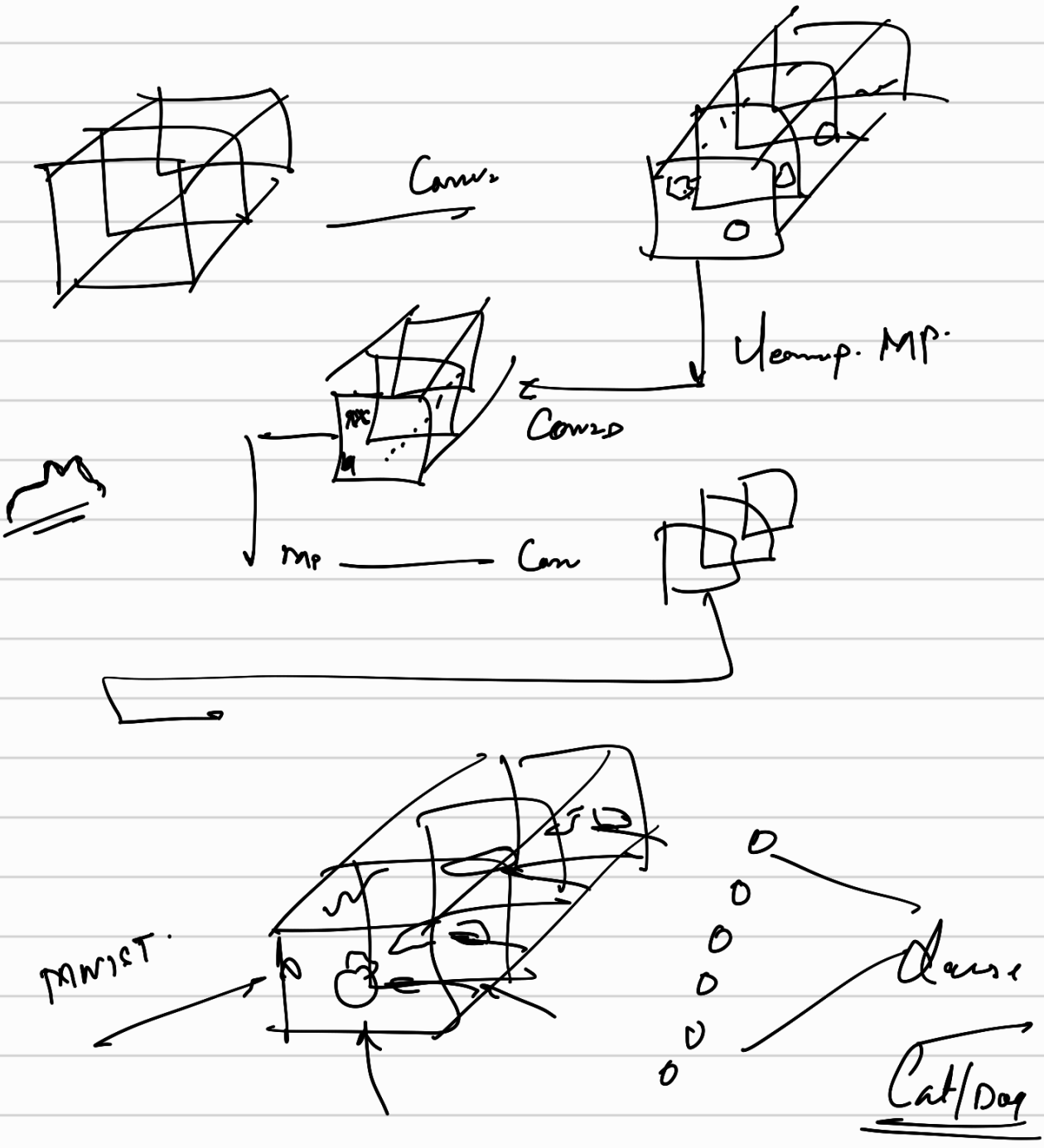
$\tanh(p_1 w_1 + p_2 w_2 + p_3 w_3 \dots p_{27} w_{27})$

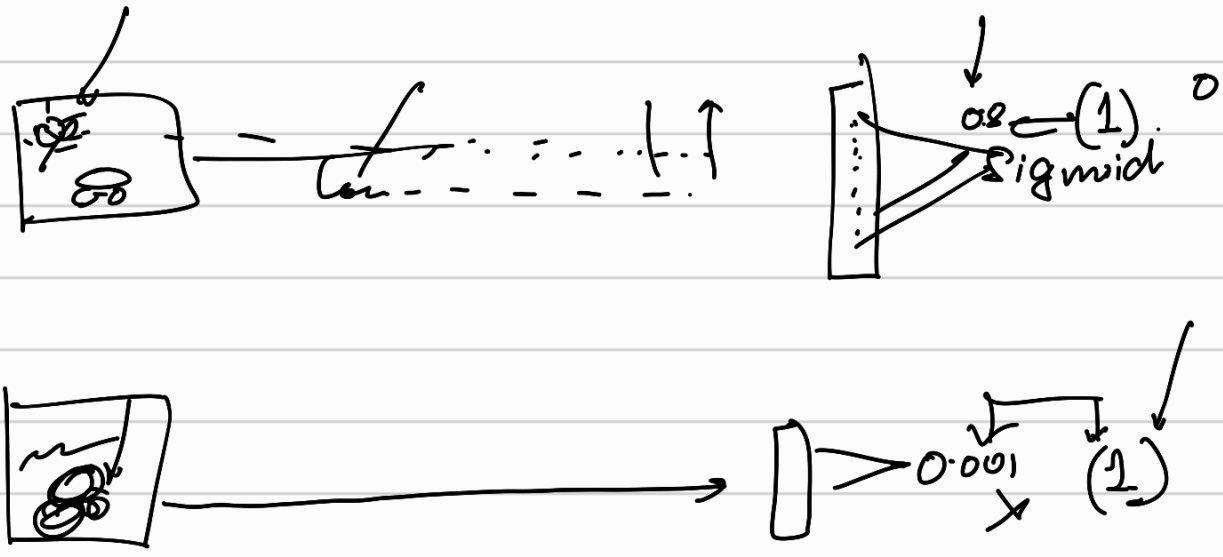


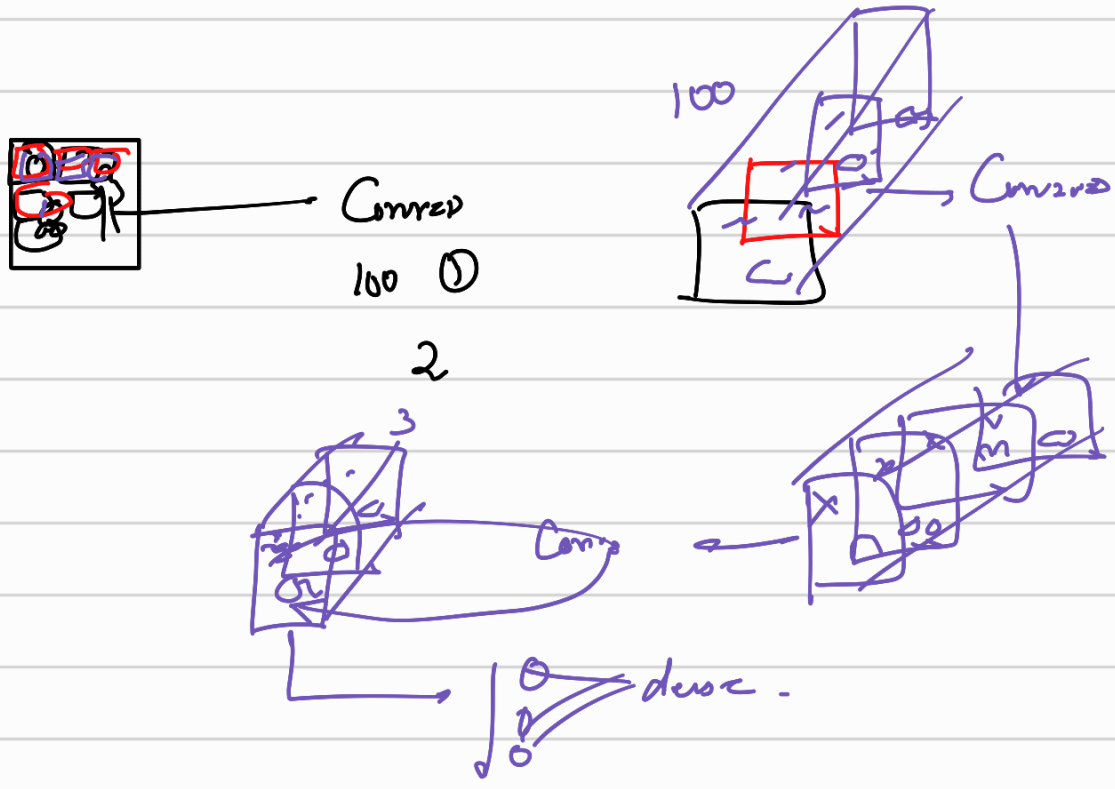


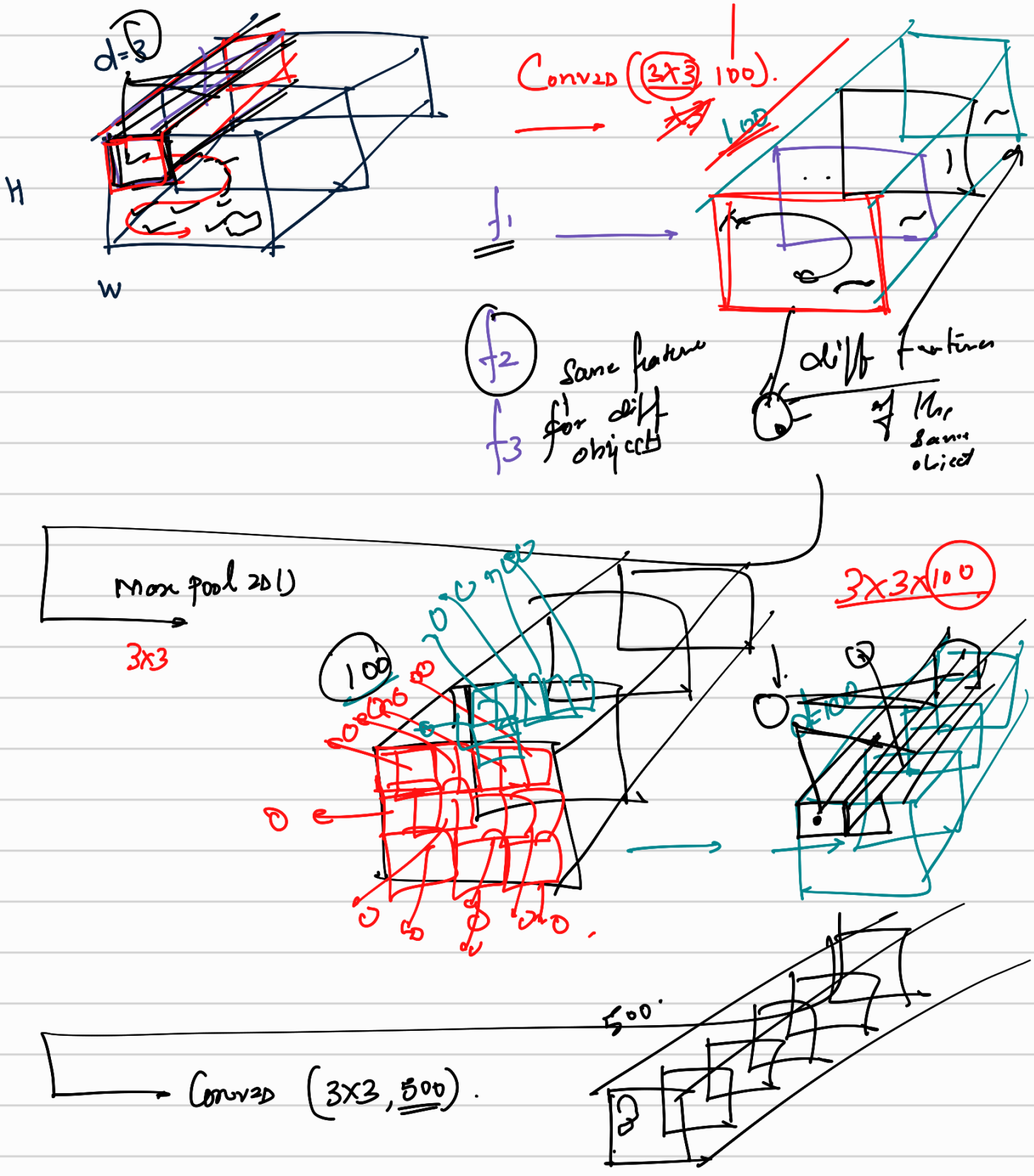
① If the filters (100) learnt the weights to extract the dog's features, then the feature maps will be containing majorly the features of the dog and very slight features of other objects like tree / bird / sun.

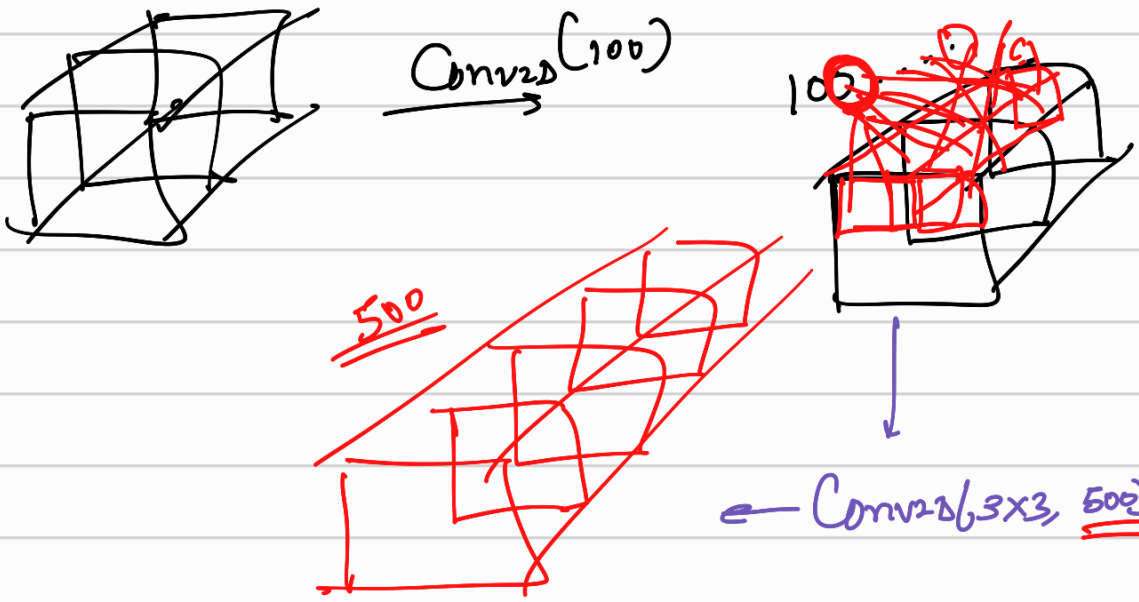

Dog
Cat

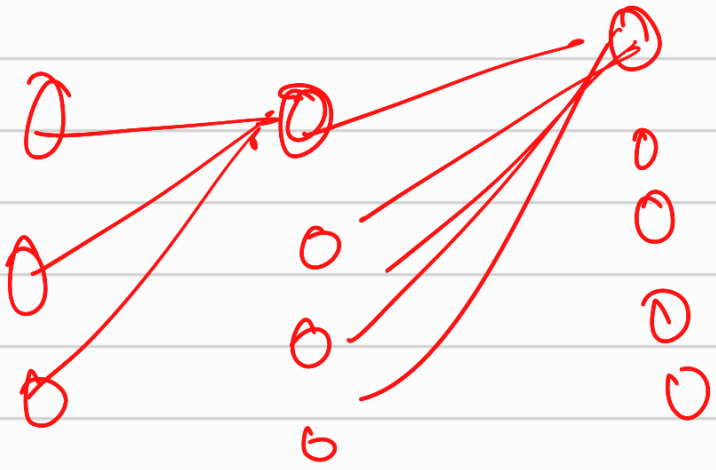

Conv
Maxp. MP.
Conv2D
MP
Conv
MNIST.
Dense
Cat/Dog

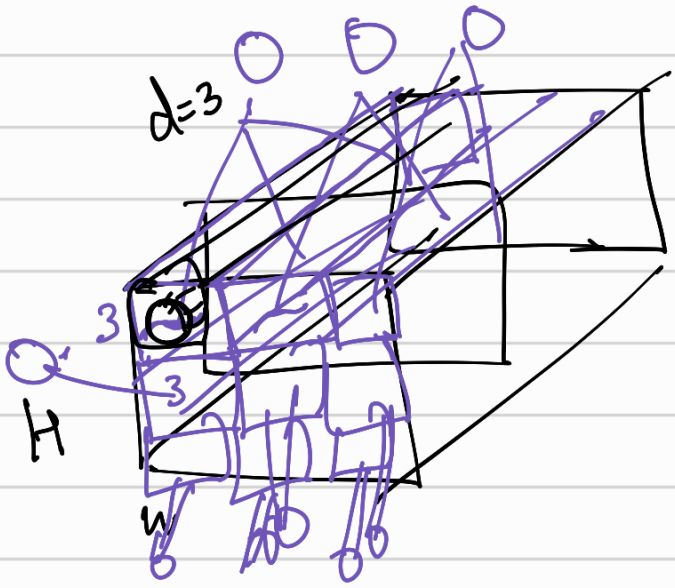

(1)
Sigmoid
0.001
(1)


100
100
2
3
dense.

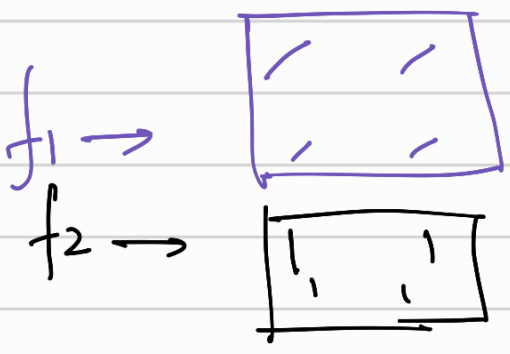

d=3
H
W
Conv2D (3x3, 100).
100
f1
f2
f3
Same feature for diff objects
diff features of the same object
Max pool 2D
3x3
100
3x3x100
500.
Conv2D (3x3, 500).

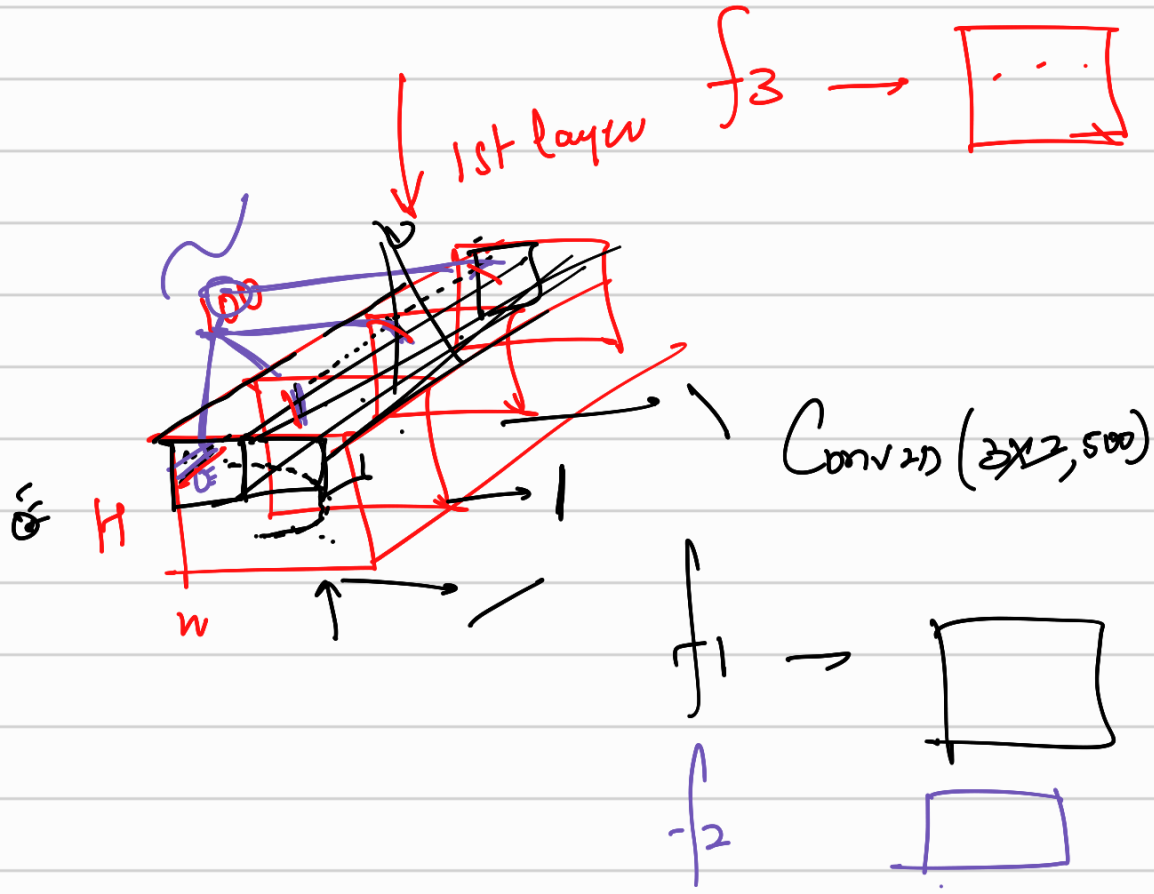


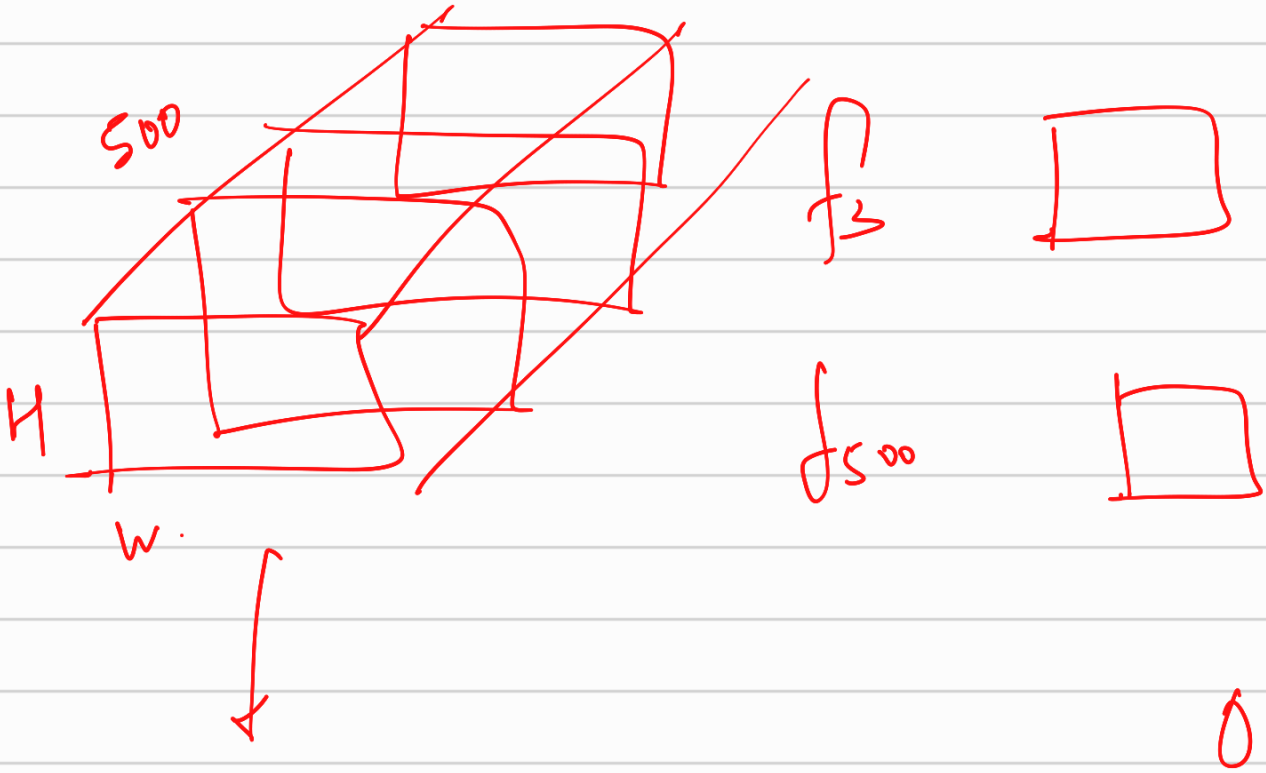




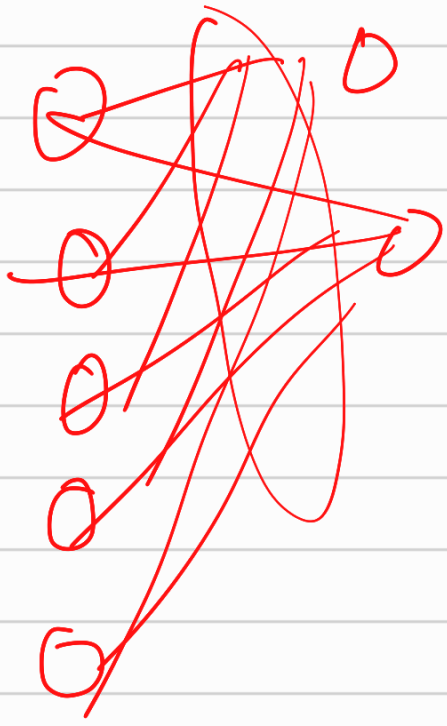
Conv2D
(3x3, 100).

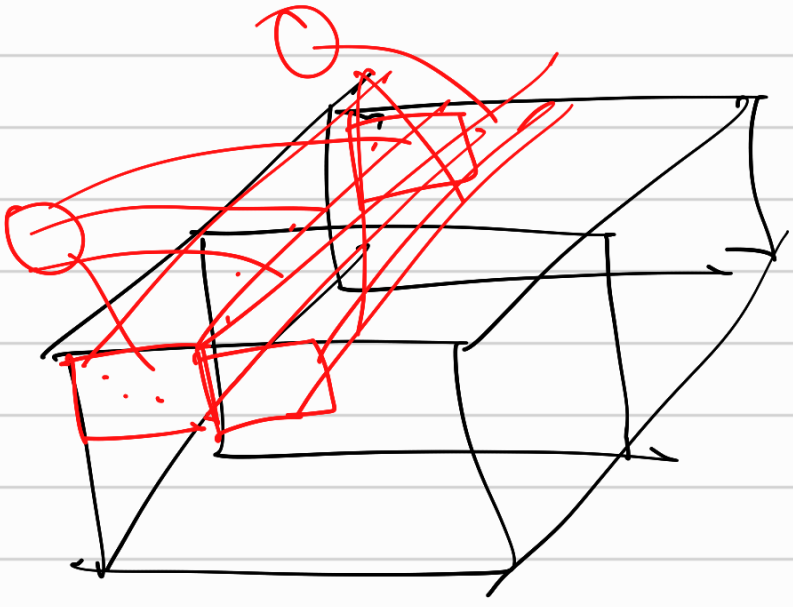






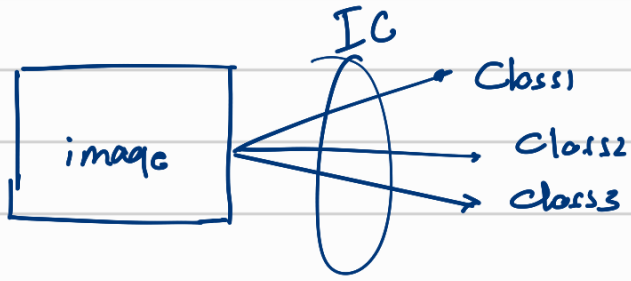
Conv2D.





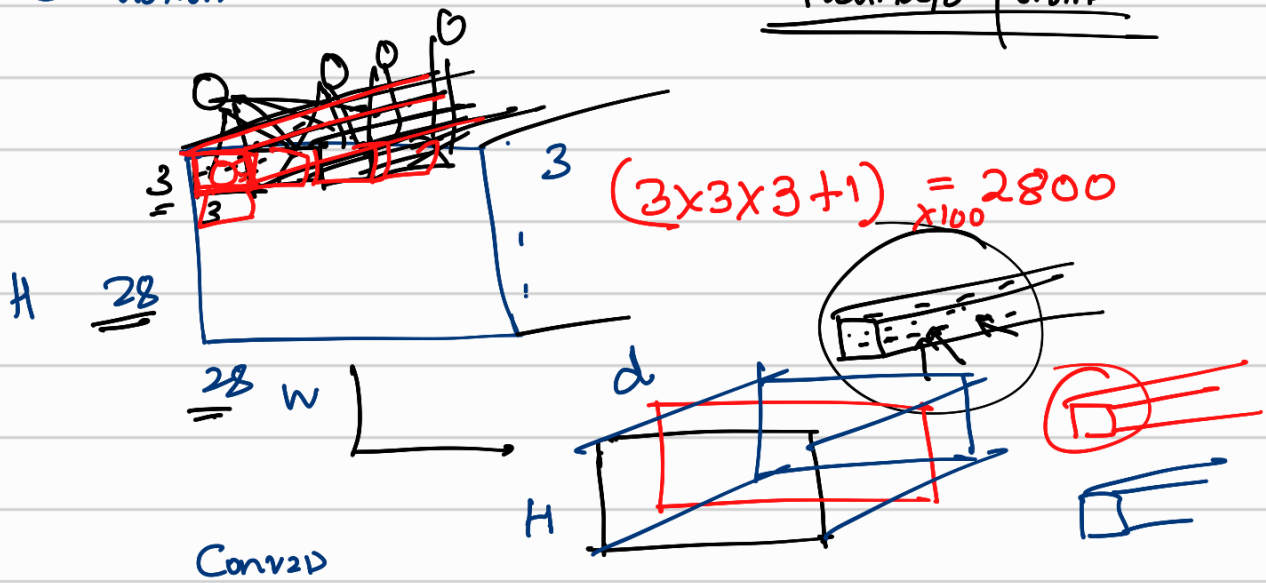
model.add (Conv2D(3x3, 100, s=1, p=~~, (input)

Convolution Neural Network

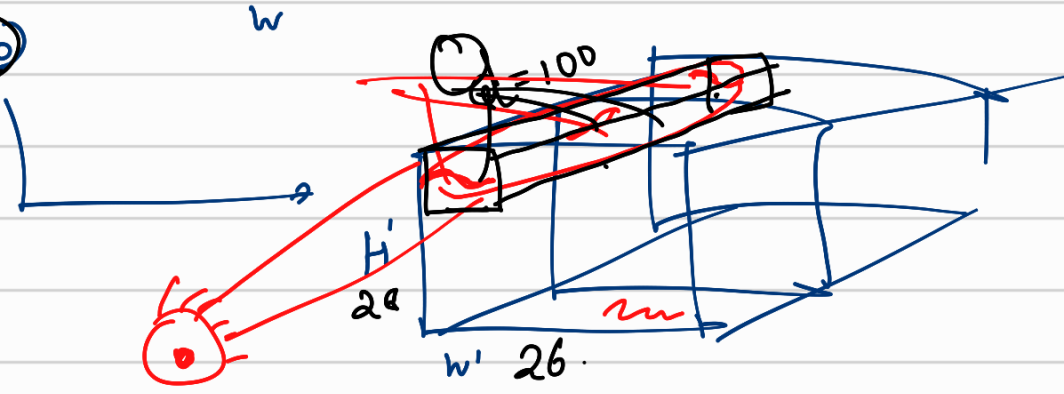


Convolution

Trainable Params:

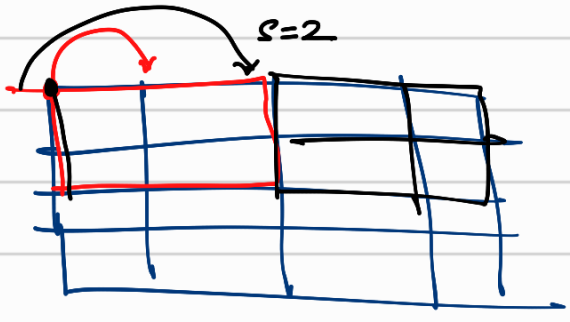


Conv2D
(3x3, 100)



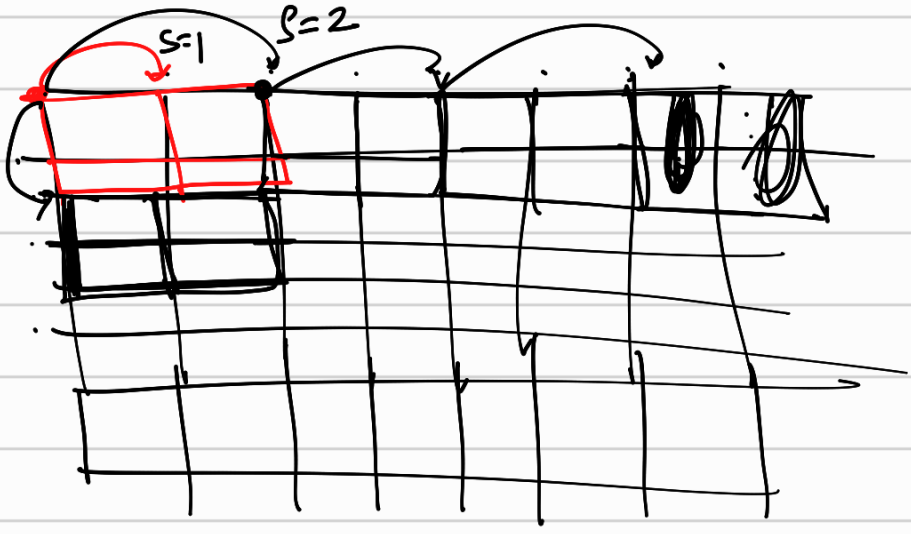
$$H' = \frac{H - f}{S} + 1 \qquad S = 1$$

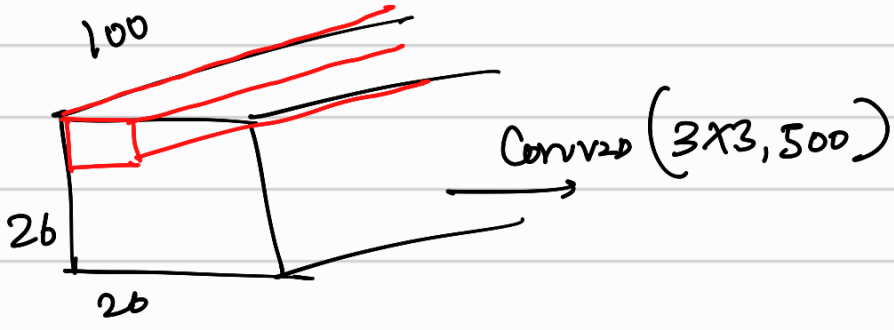
$$W' = \frac{W - f}{S} + 1$$

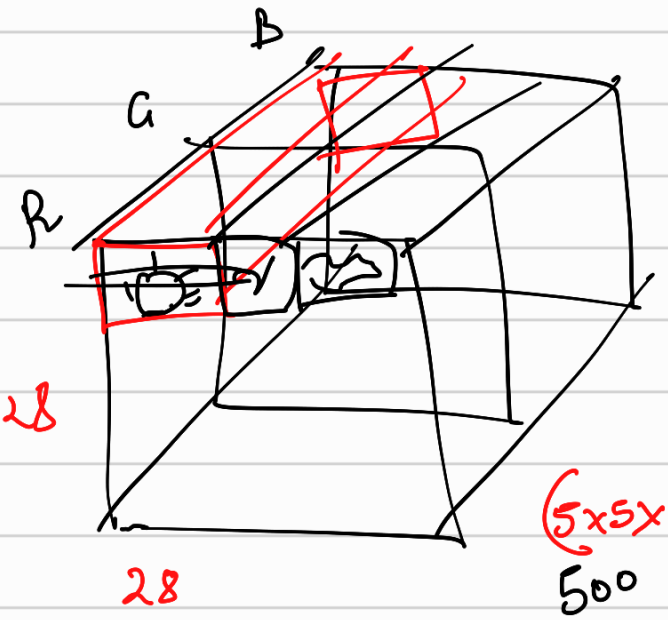


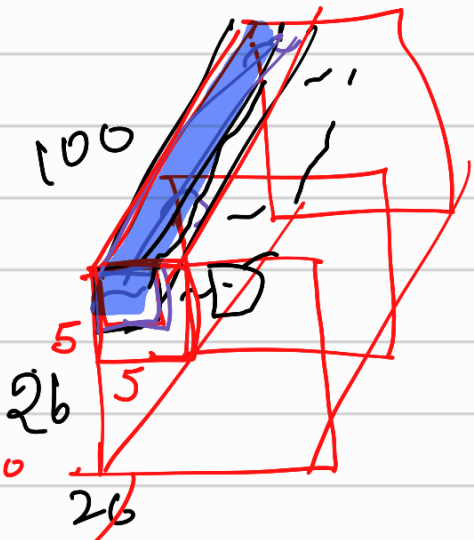
$$H' = \frac{28 - 3}{1} + 1$$

$$= 26 \qquad 26$$

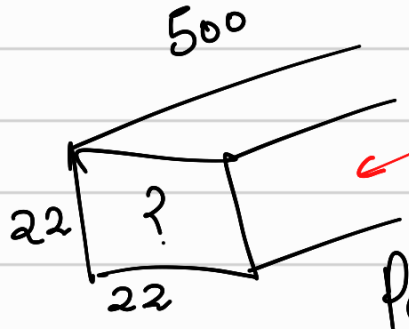

S=1
S=2


100
26
26
Conv2D (3X3, 500)





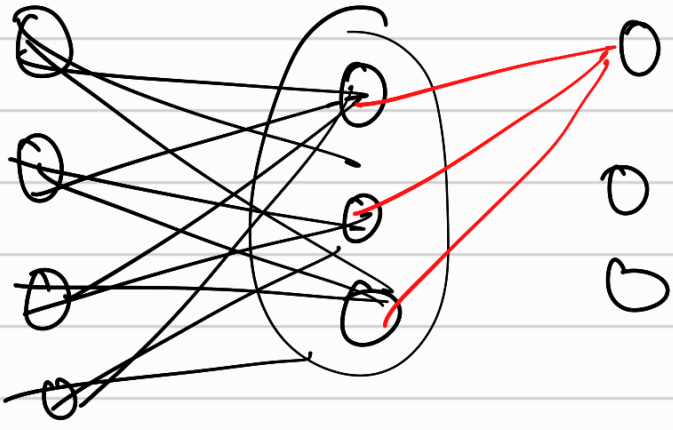
$(5 \times 5 \times 100 + 1) \times 500$

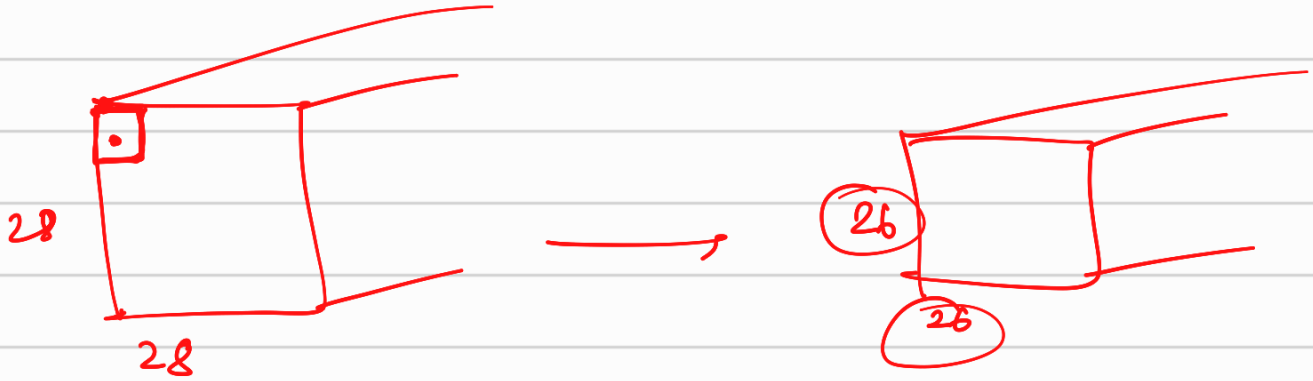


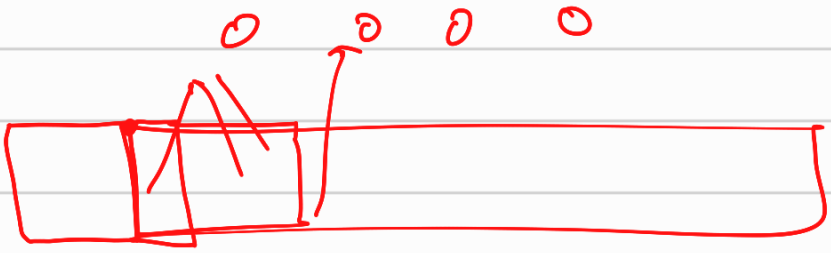
Convolve $(5 \times 5, 500)$

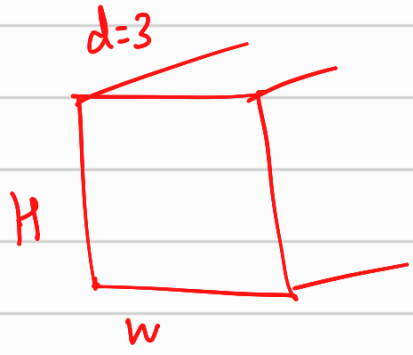
$$\frac{26-5}{1} + 1 = 22$$

Params?

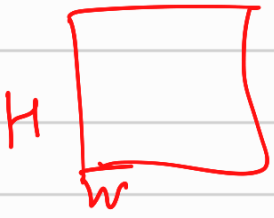








Conv2D ($f \times f$, 3X3, 100, $S=1$)



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

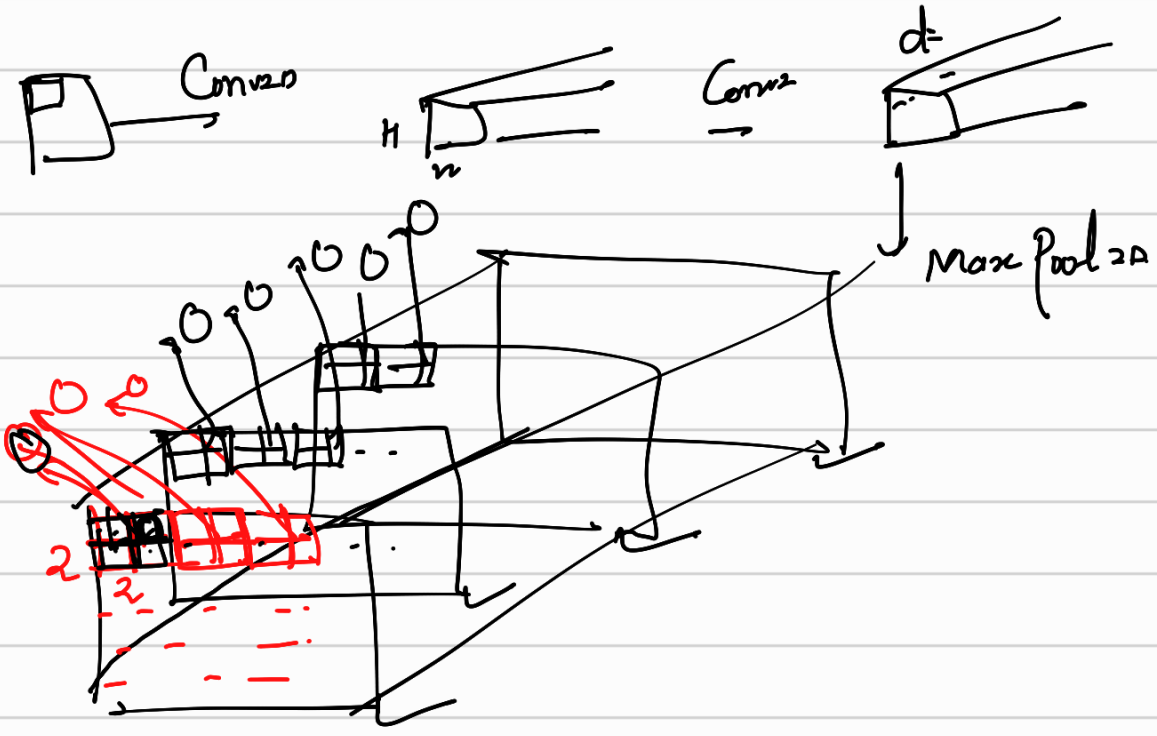
$$W' = \frac{W - f}{S} + 1$$

with Padding

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

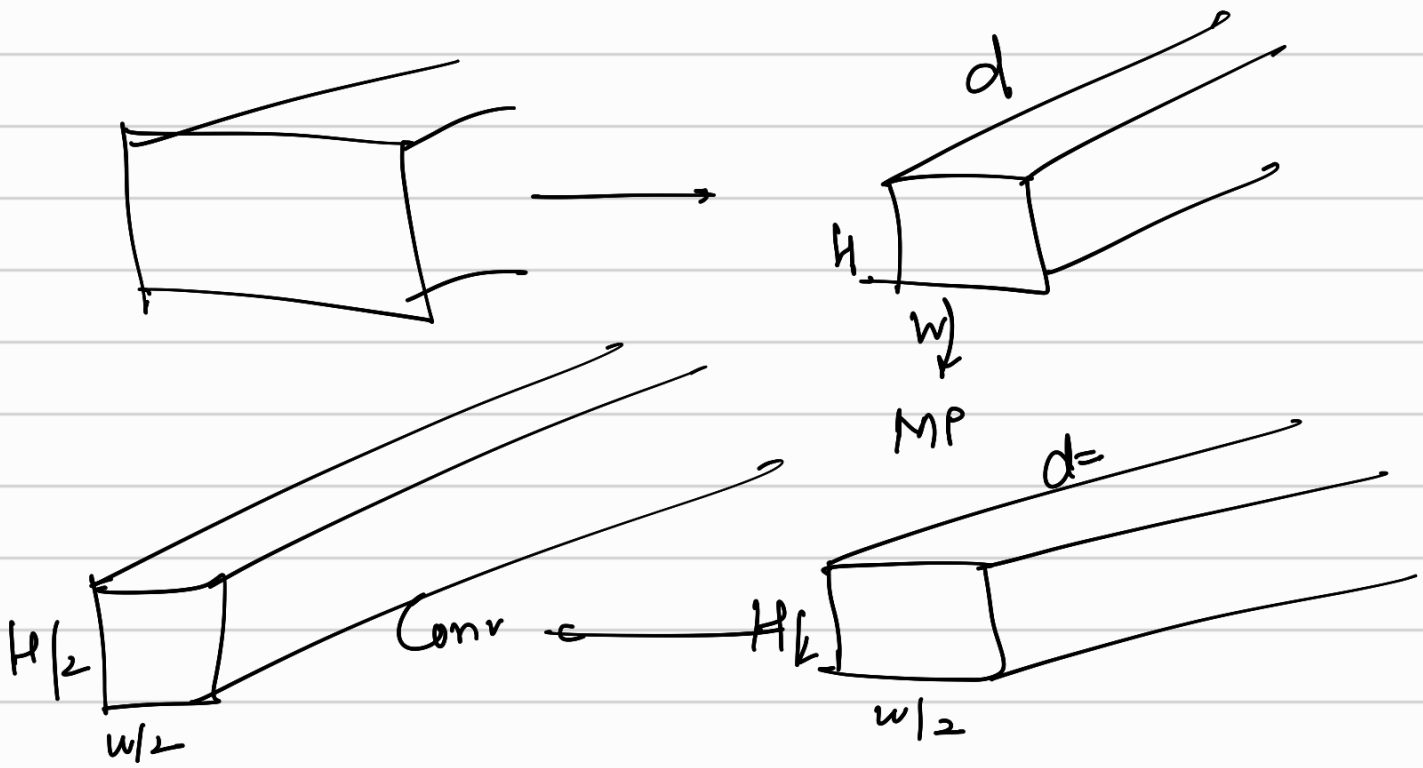
$$W' = \frac{W + 2P - f}{S} + 1$$

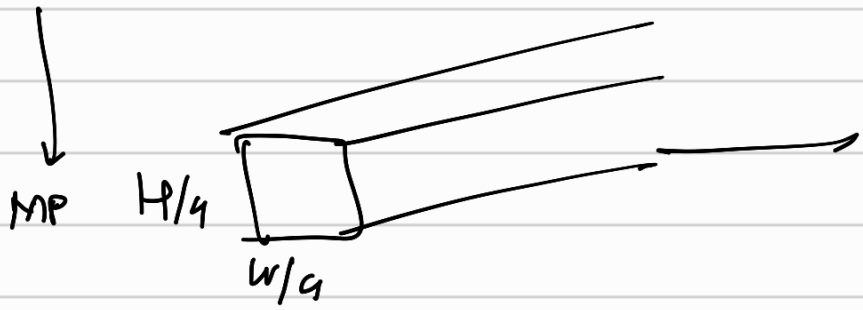
Conv2D (3X3, $S=1$, Padding="same")

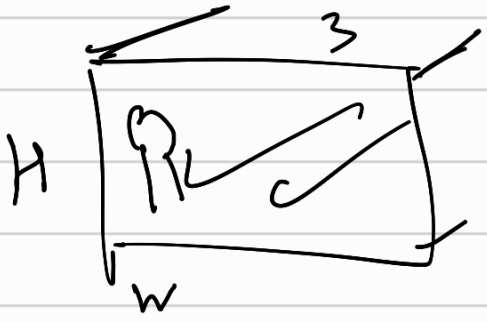


$$H' = \frac{H - f}{s} + 1$$

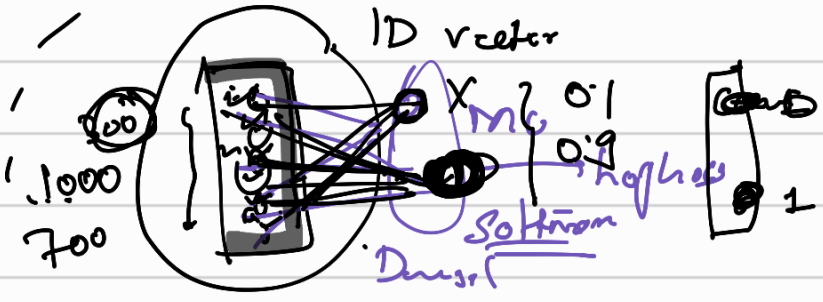
$$H' \quad \frac{H-2}{2} + 1$$

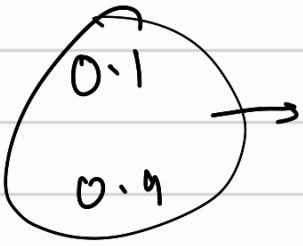


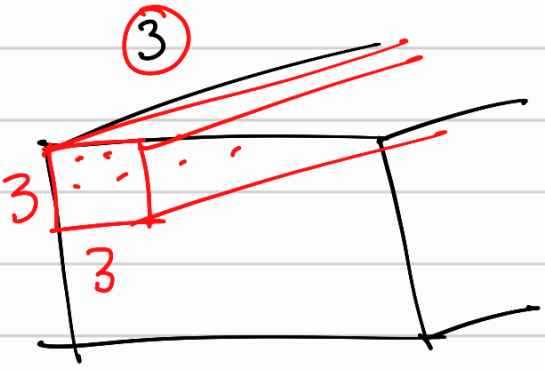

MP
H/4
W/4


3
H
W

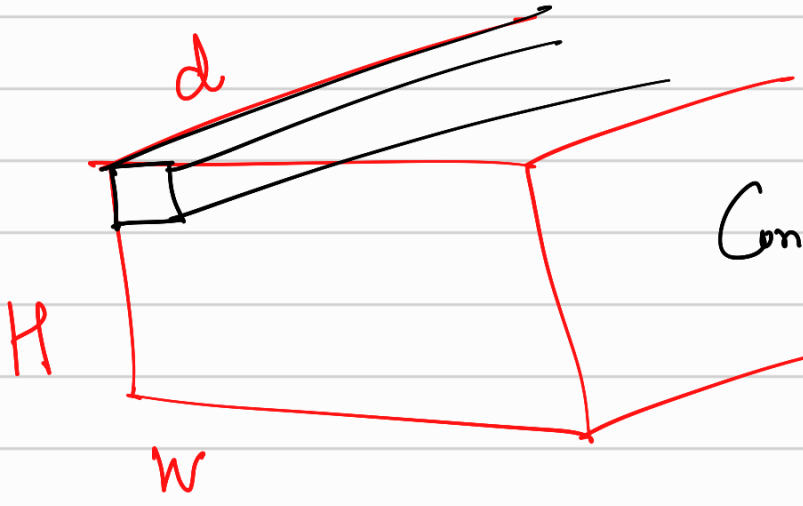

1000
H/32
W/32


1D vector
X
0.1
0.9
1,000
700
1


0.1
0.9



(7+1) X 32



Conv2D (f x f, n)

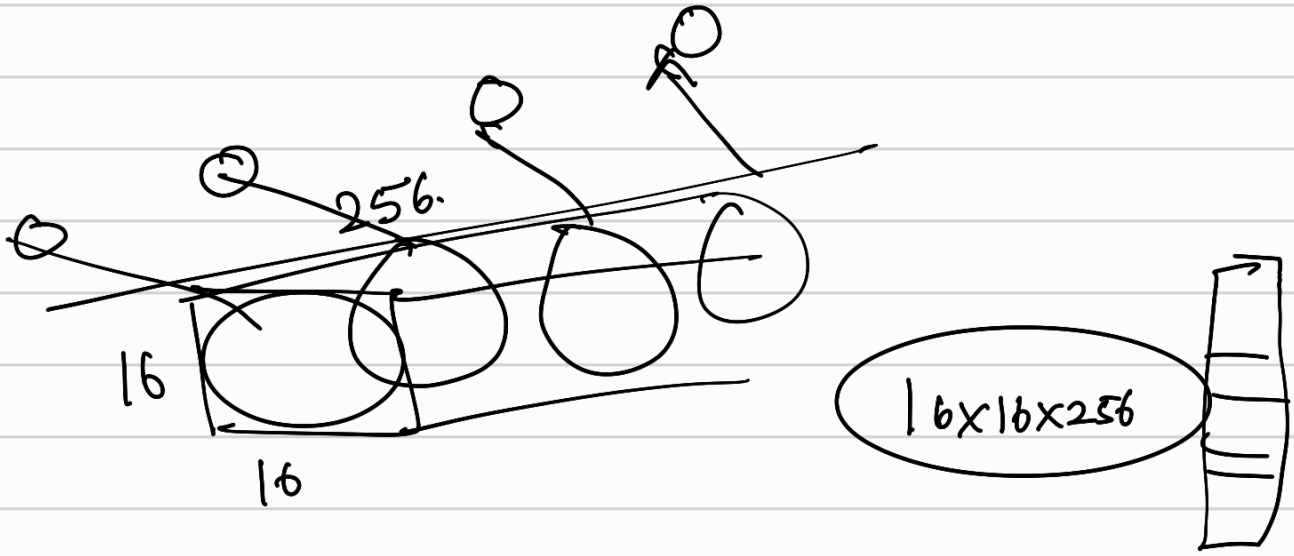
(f x f x d + 1) x n

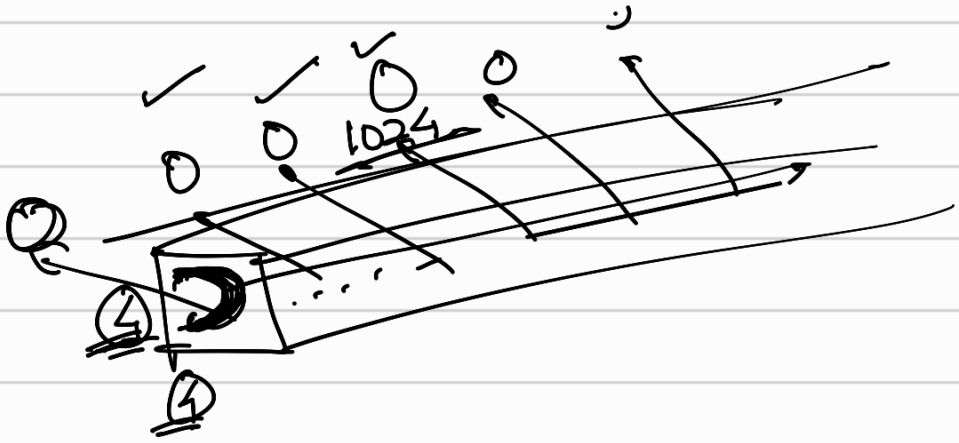
$$\frac{H-f}{s} + 1$$

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

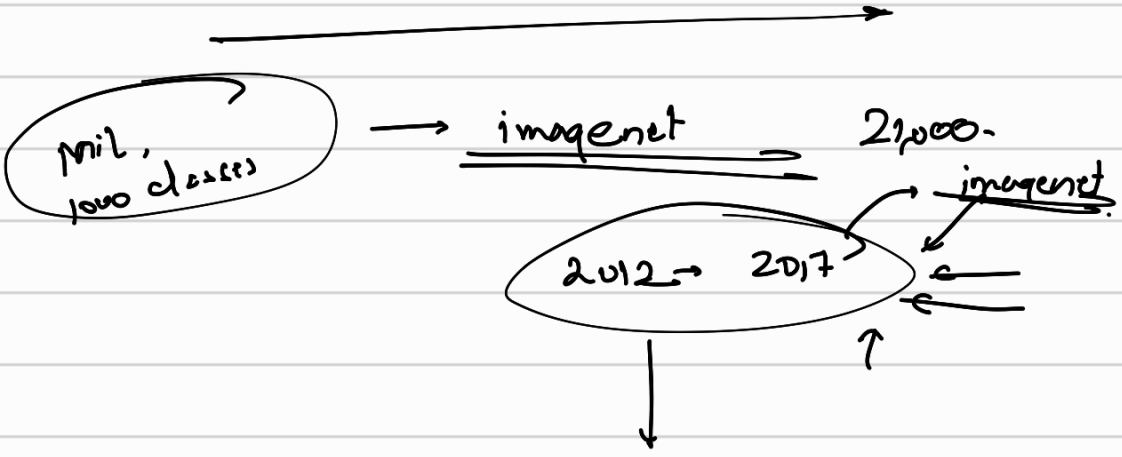
$$= 127 + 1$$

$$= 128$$

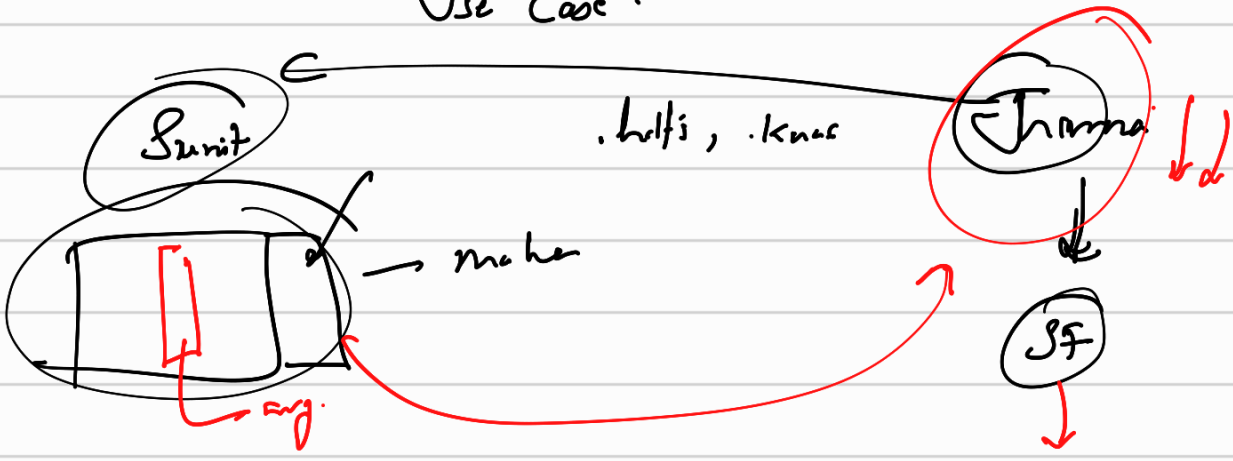

256
16
16
16x16x256

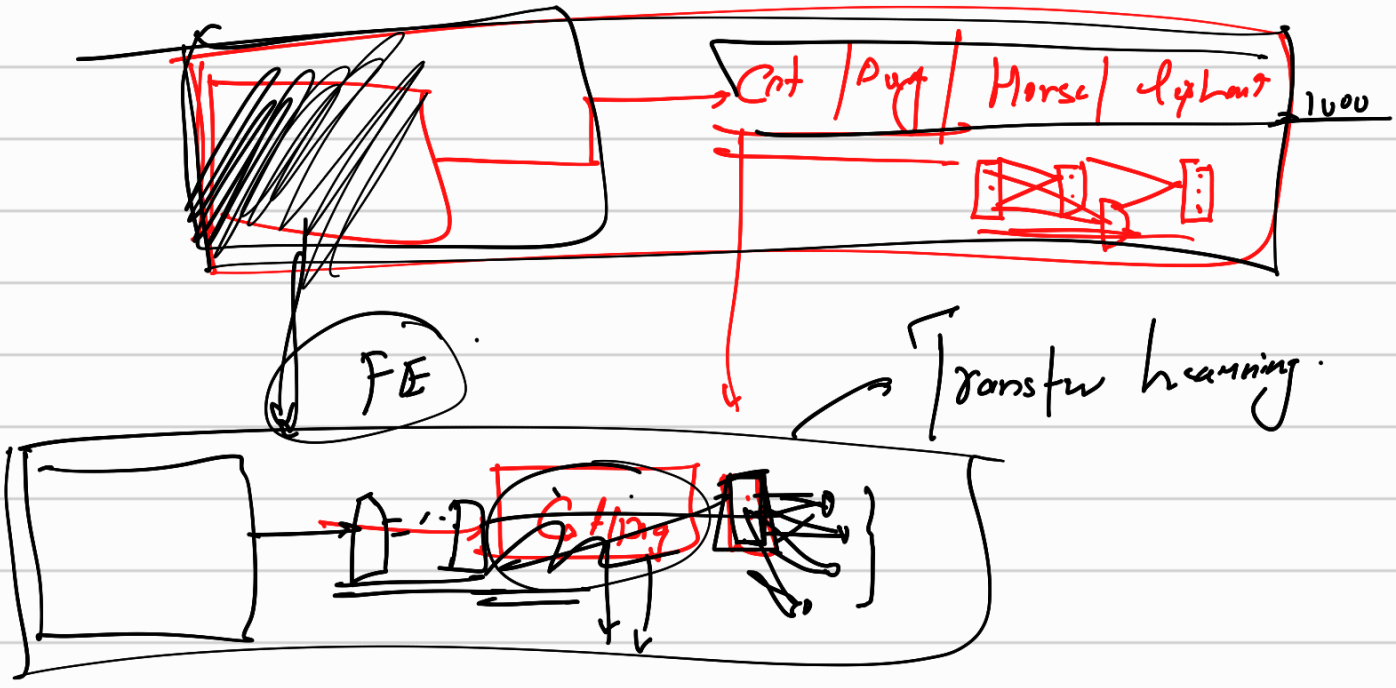

1024

Transfer learning →



Use Case.





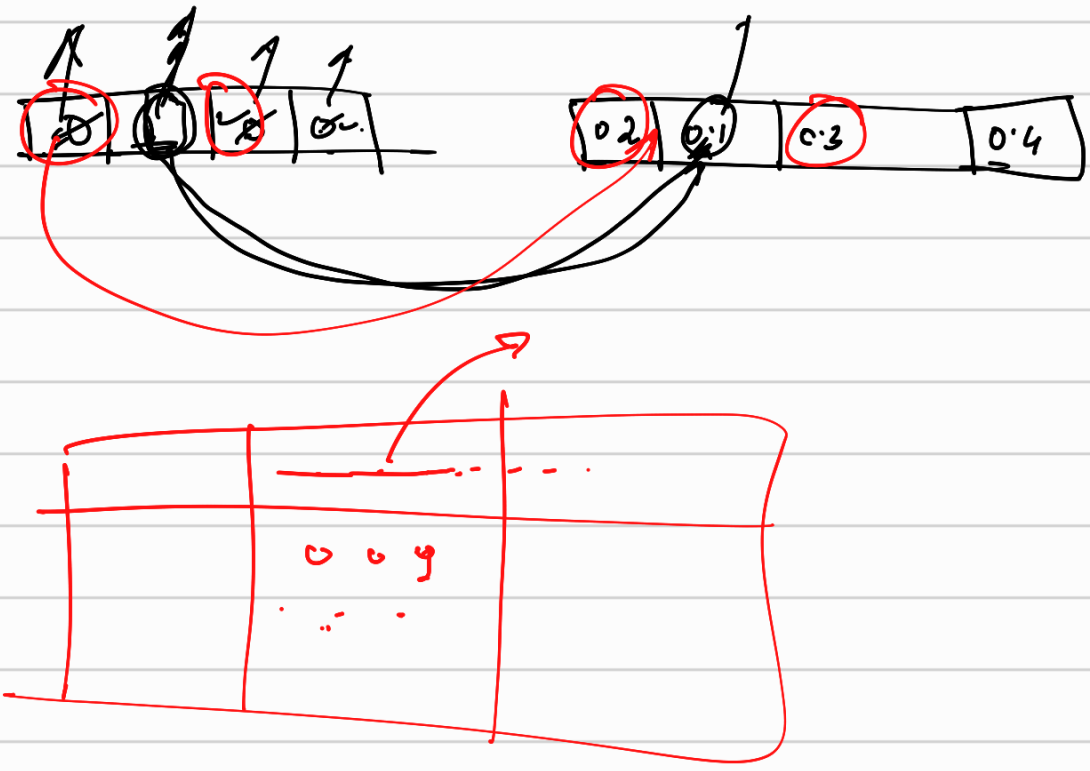
Alexnet → '12

VGG-16

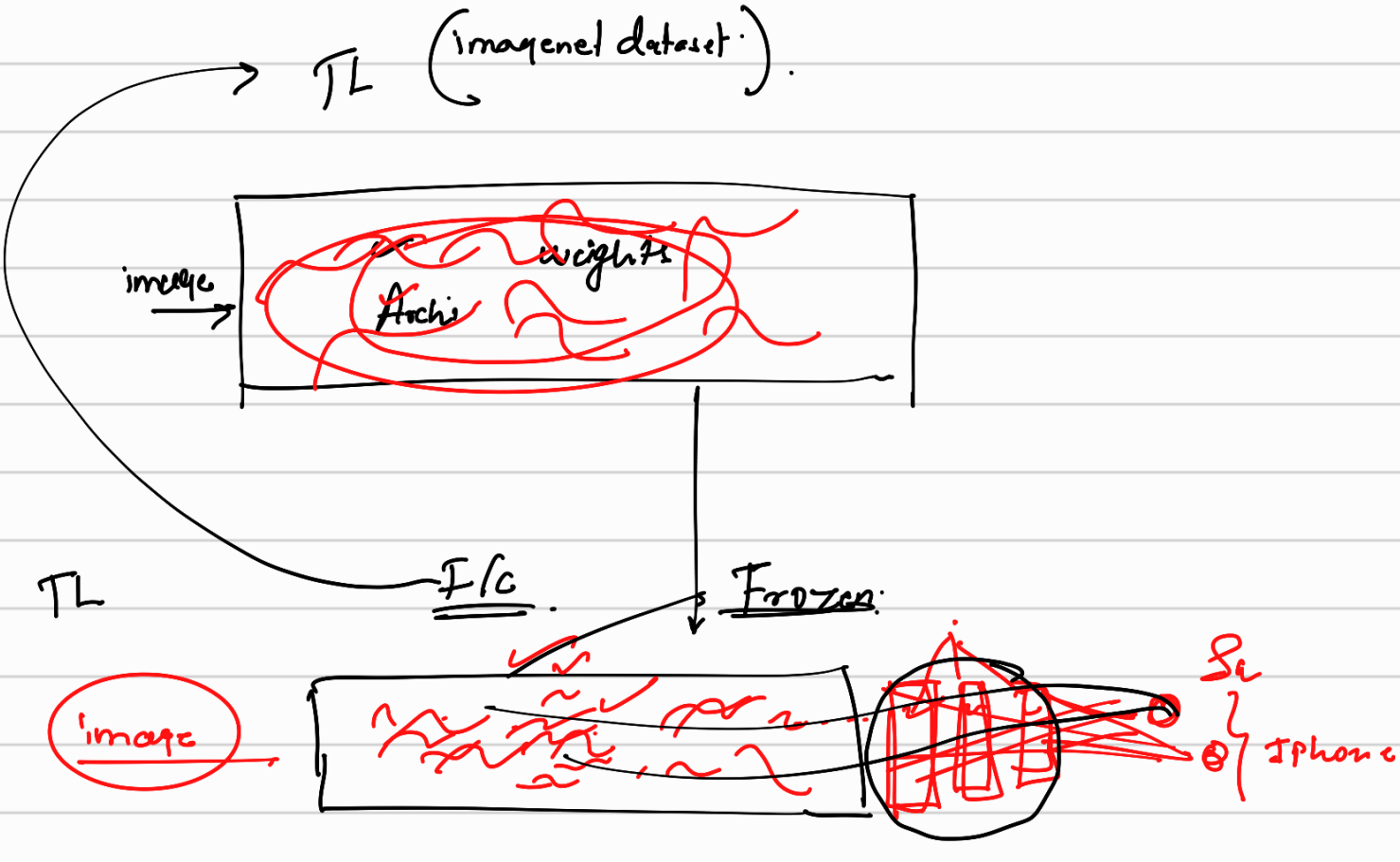
VGG-19

Inception net → Google.

Resnet → ✓✓



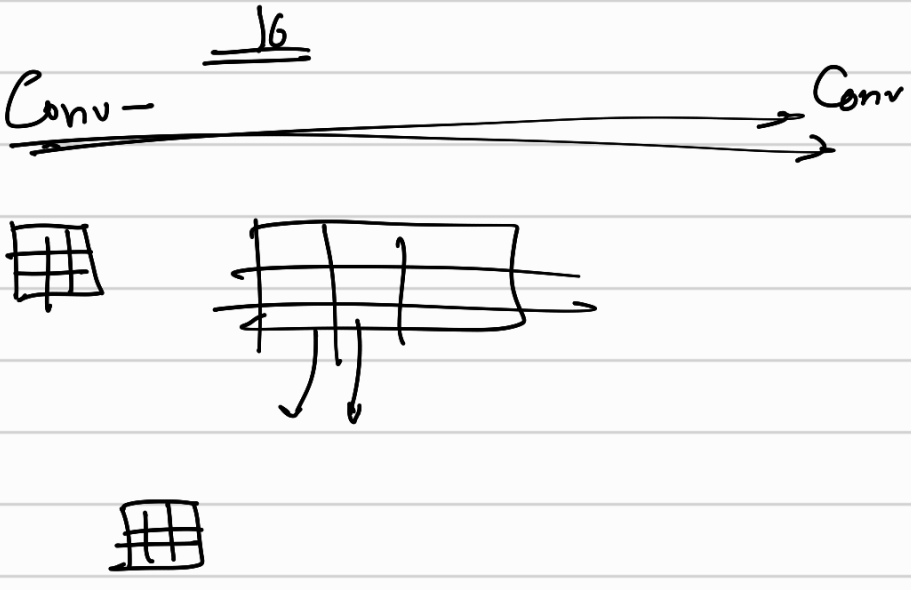
TL (imagenet dataset).

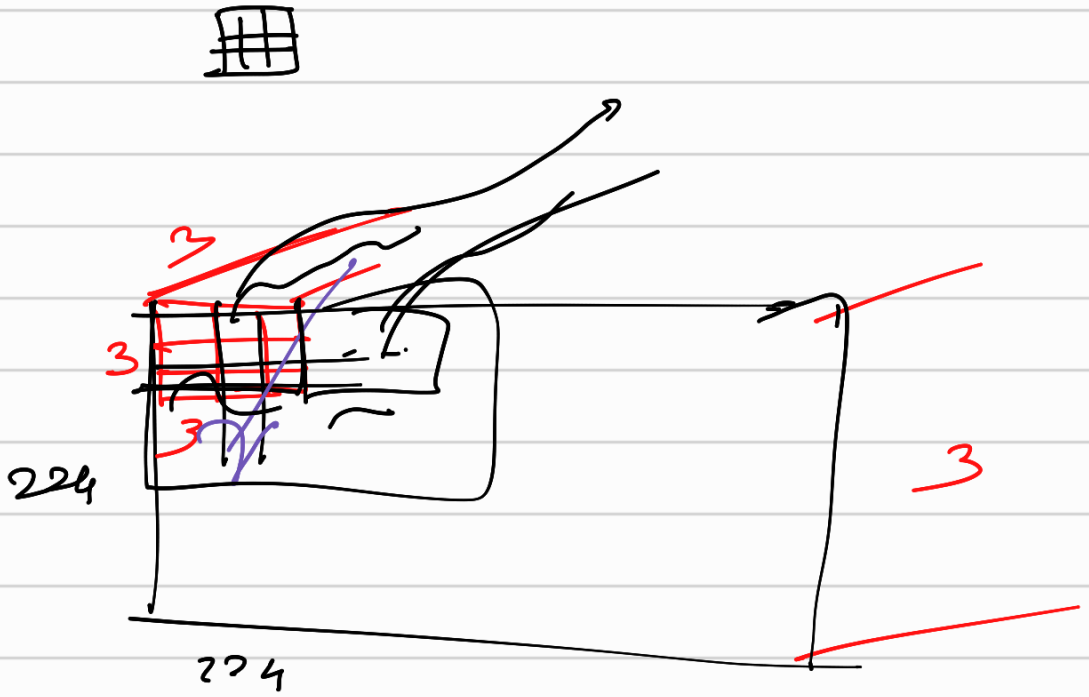


VGG-16 / VGG-19

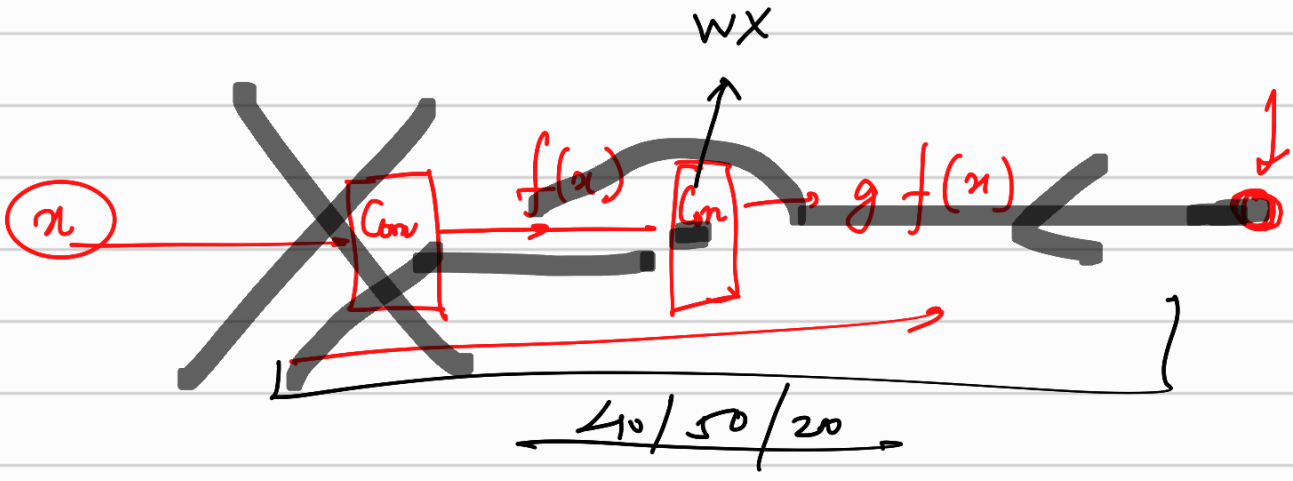
→ Conv conv MP Conv conv MP Conv conv conv MP → FC FC FC

Conv conv conv MP →

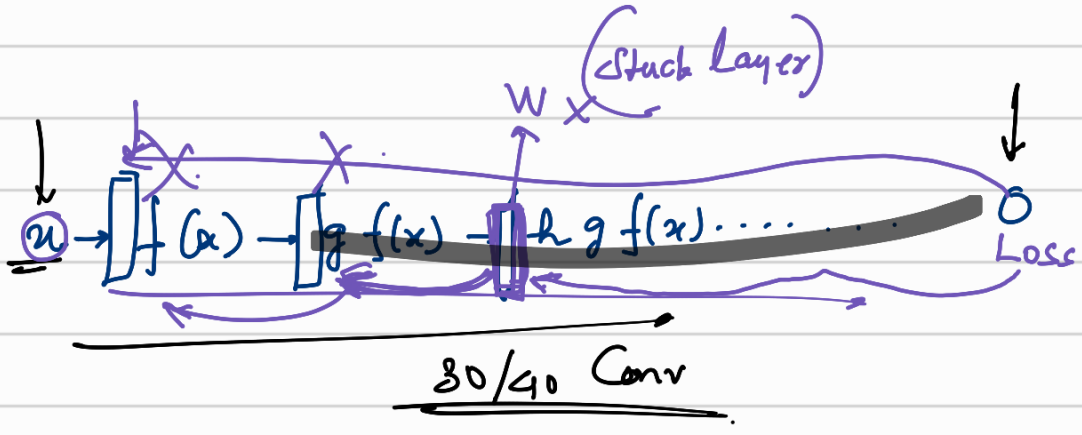




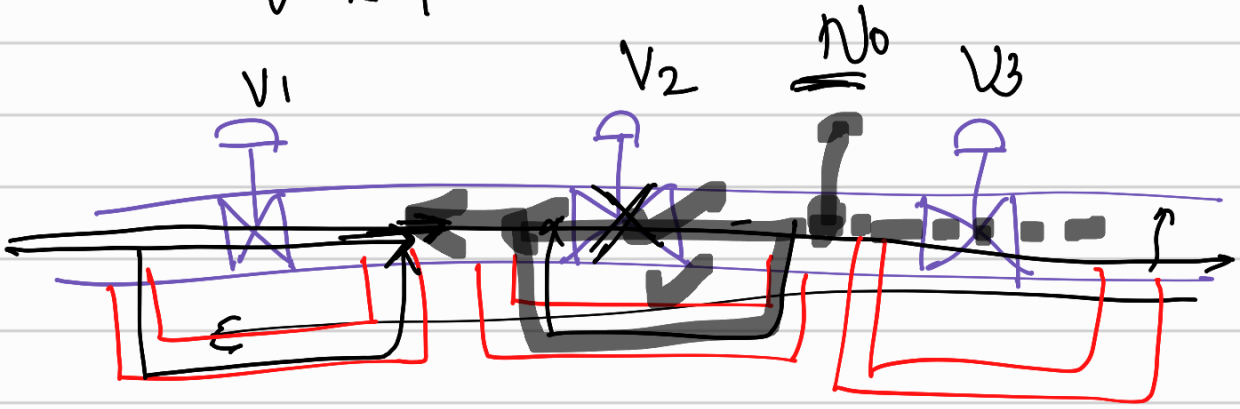
Resnet

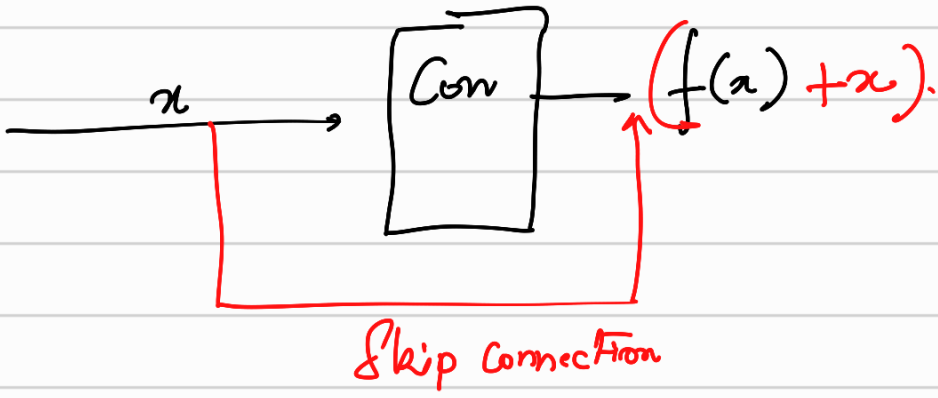


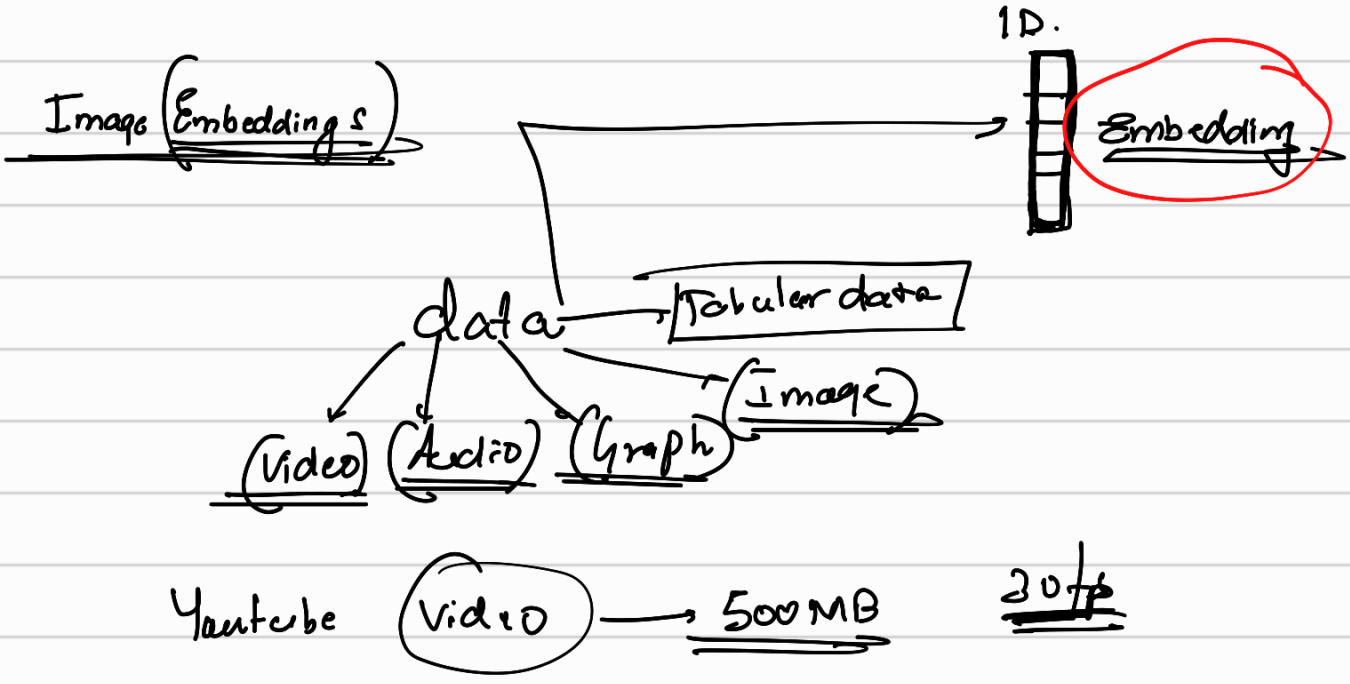
Resnet

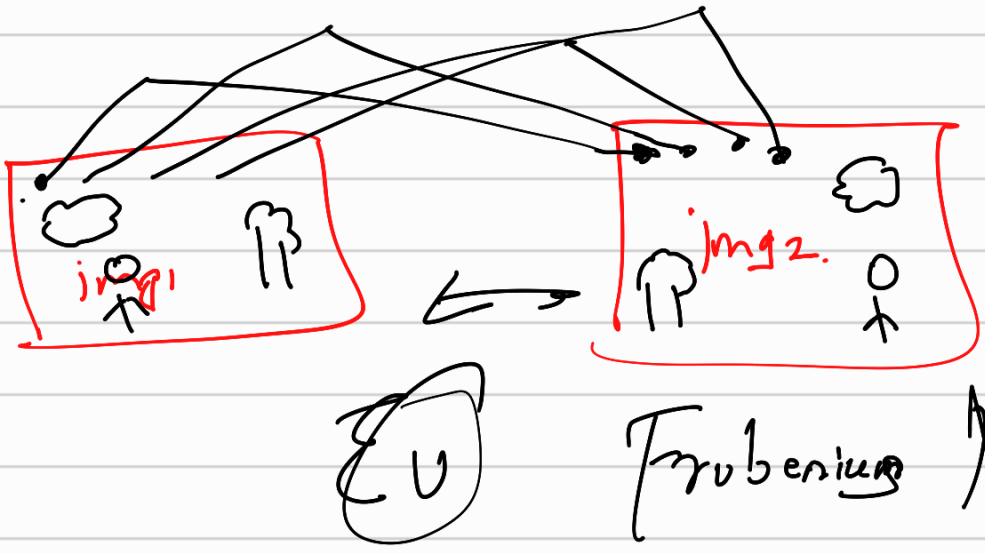


V = Layers

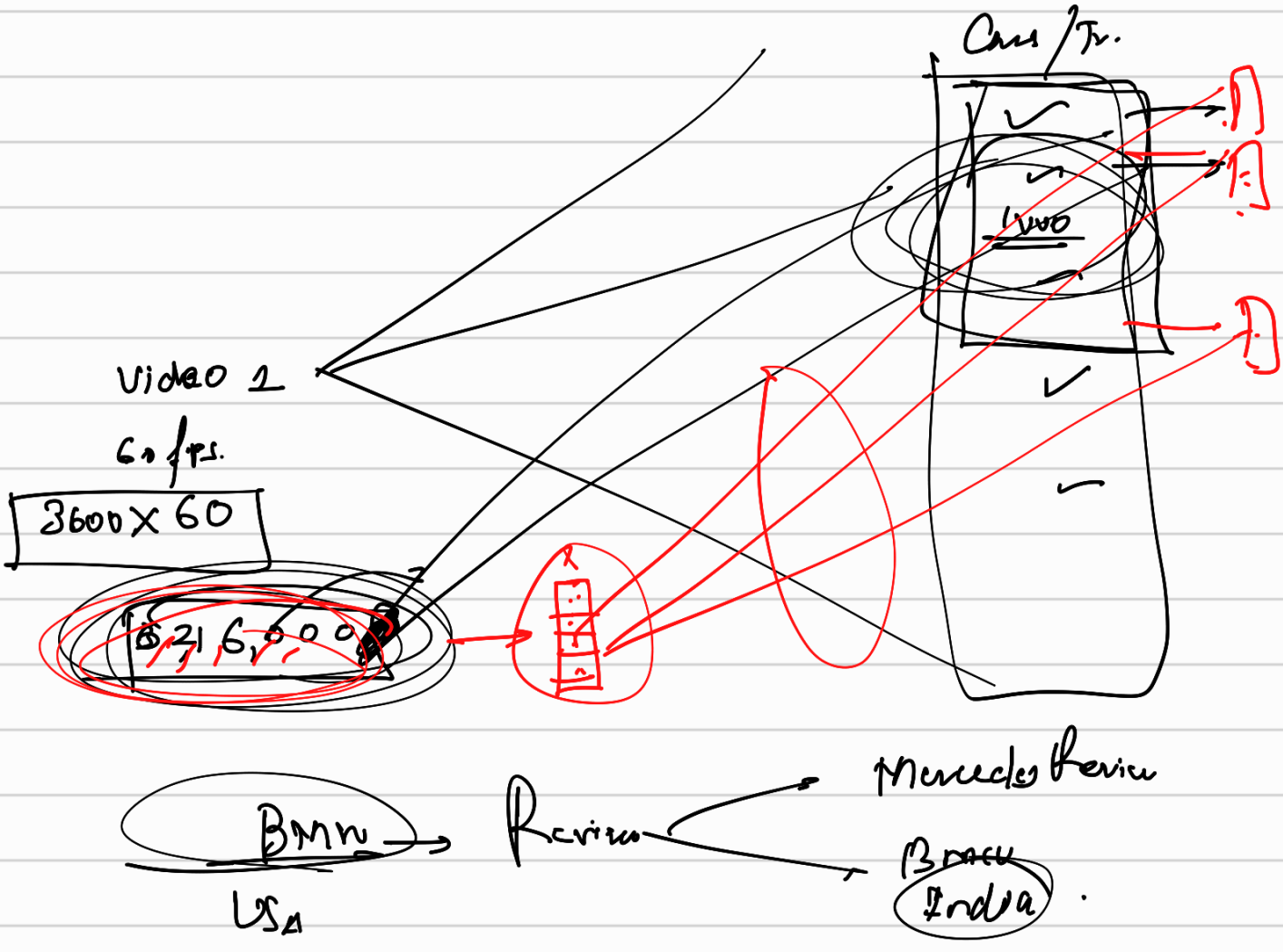

V1
V2
No
V3


x
Conv
f(x) + x
Skip connection

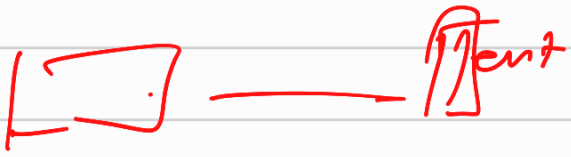

1D.
Image Embeddings
Embedding
data
Tabular data
Image
Video
Audio
Graph
Youtube
Video
500MB
20%



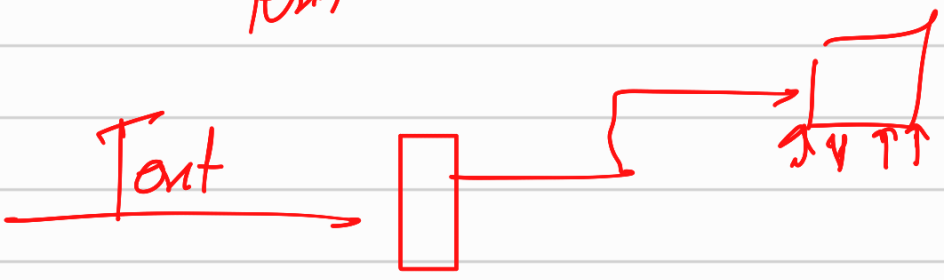
EU Trubenius Norm.

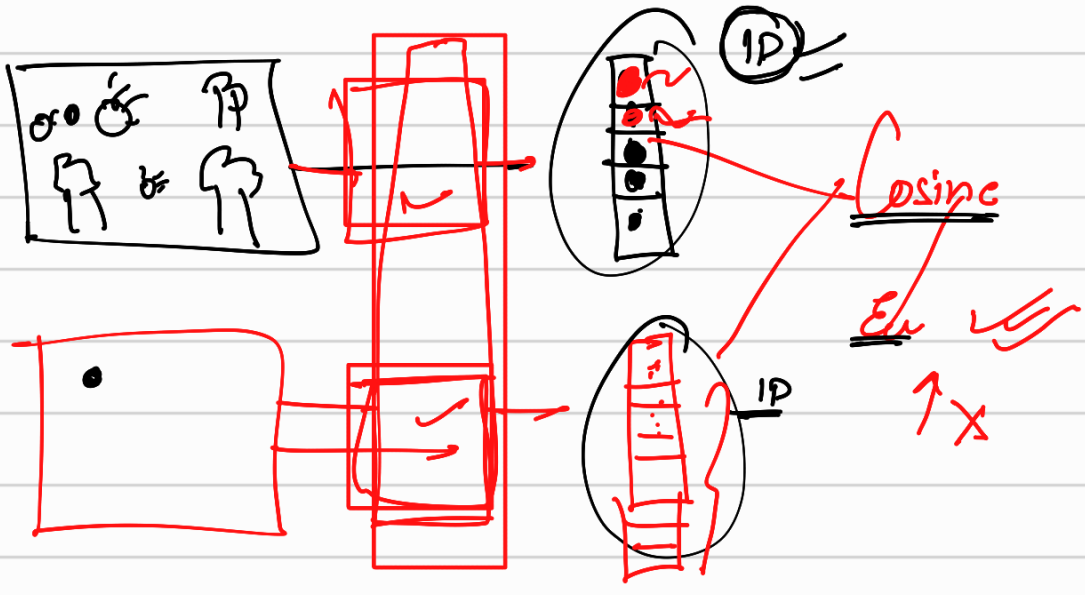


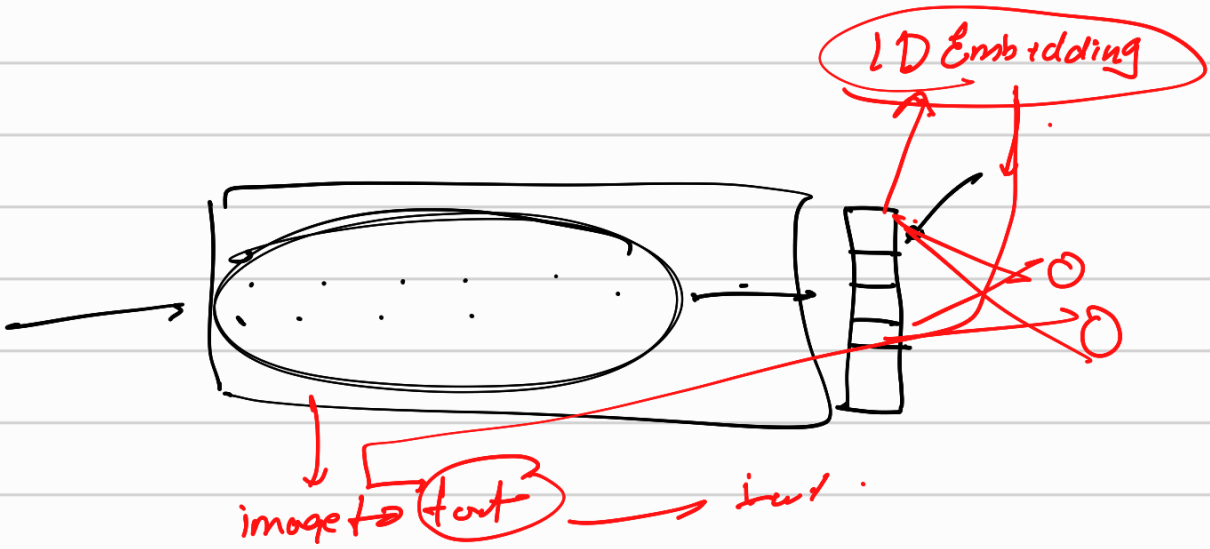
Text ---. → image

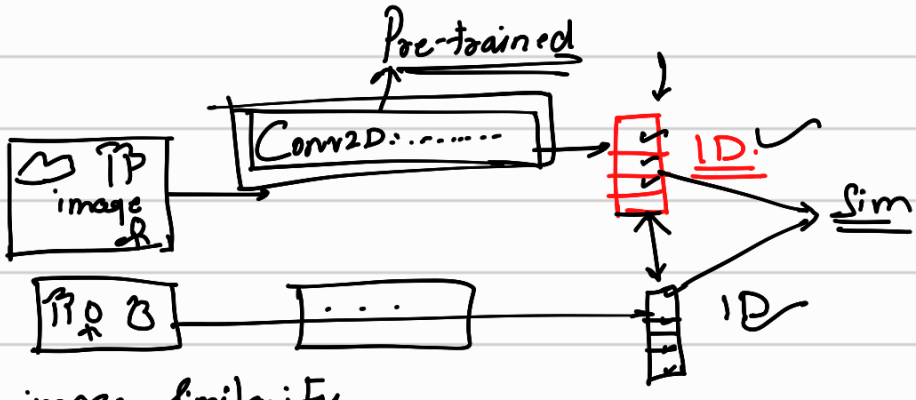


Text → video.



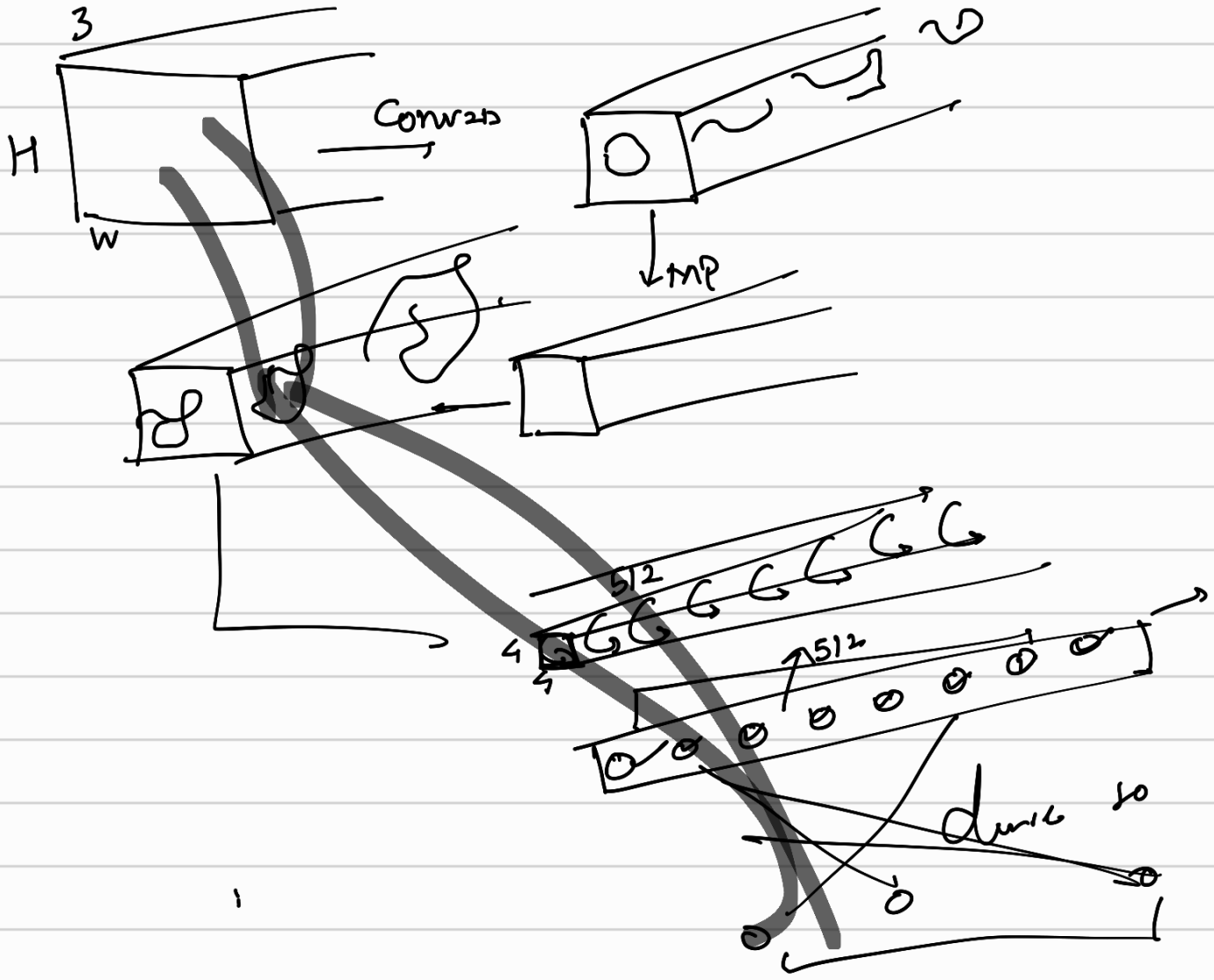


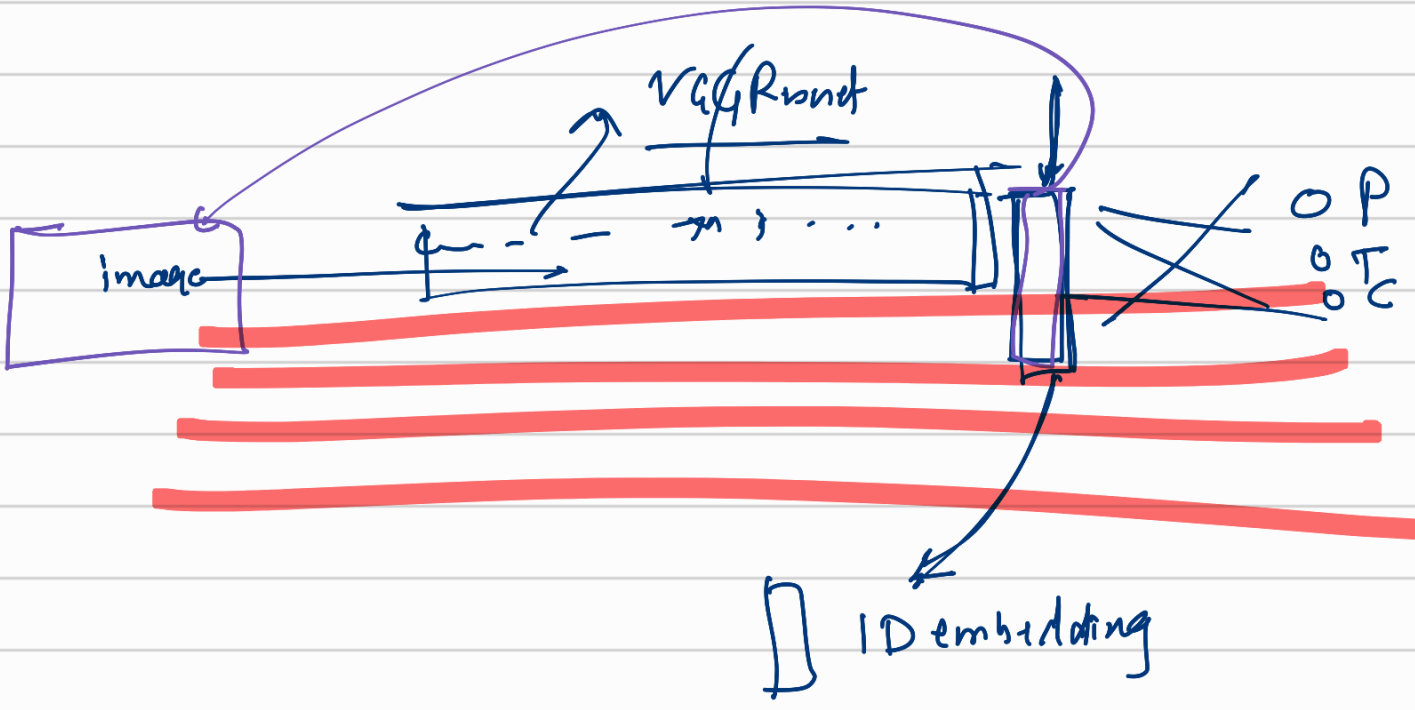




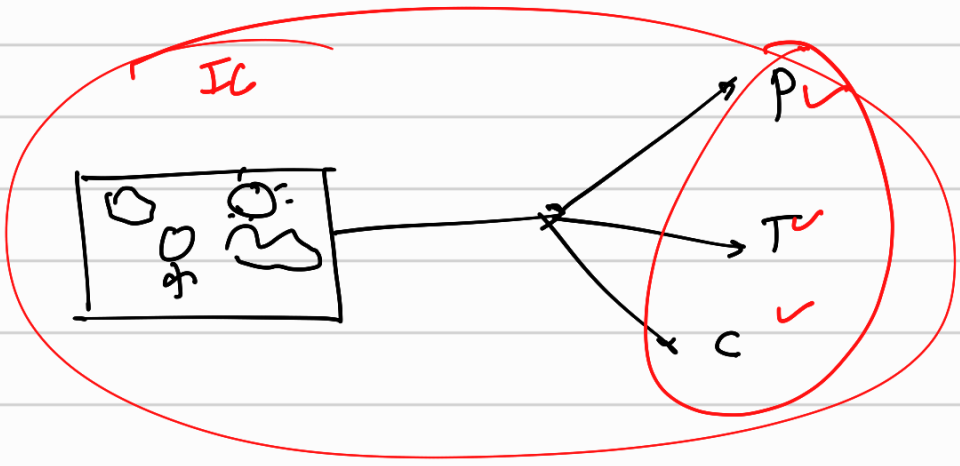
① Calculating image similarity.

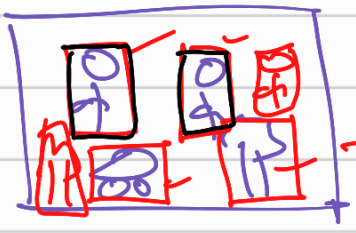
- Recommendation systems.
- Clustering images
- Facial Recognition systems.
  - Siamese Network.
- image → text
- text → image


3
H
W
Conv2d
MP
512
512
4


VGGRoot
image
P
T
C
ID embedding

Object detection & localization :





→ n objects of interest, n instances

RCNN.

Fast RCNN

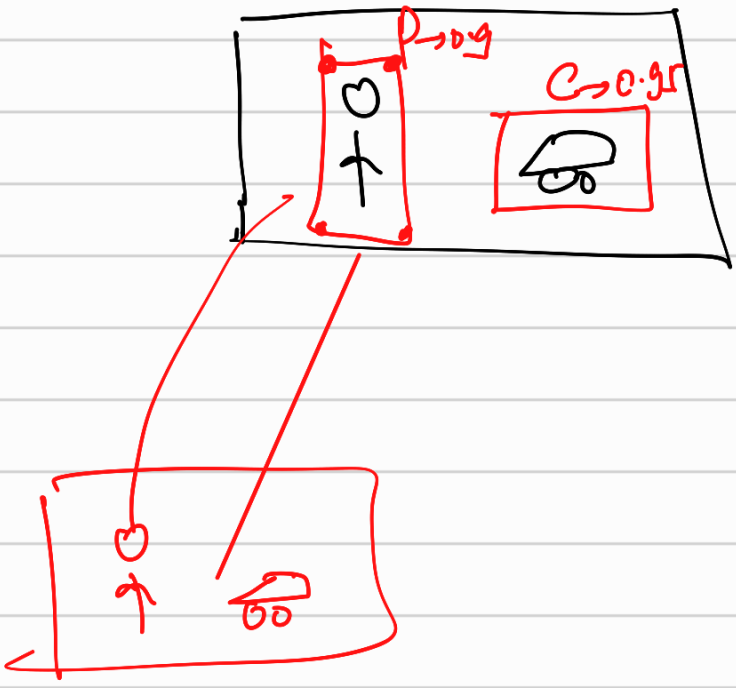
Faster RCNN.

Yoho v1 → YoloV11

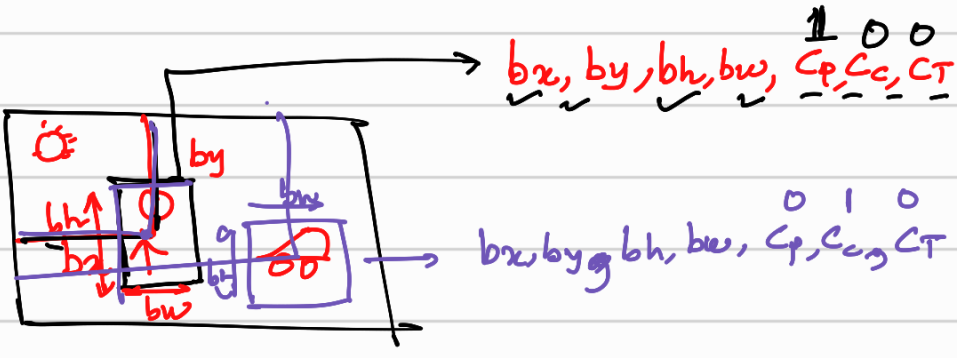
Supervised con.

Annotations

.
.
.



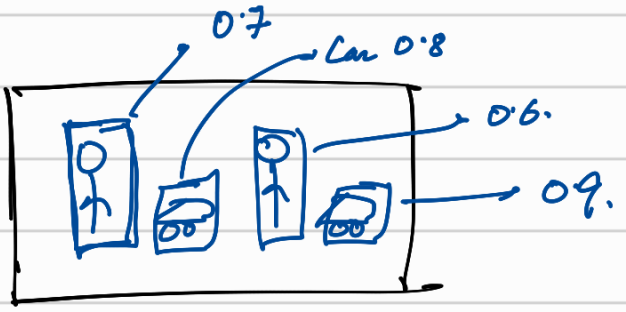
Annotation of bounding boxes to solve for the supervision of obj det and localization problem.

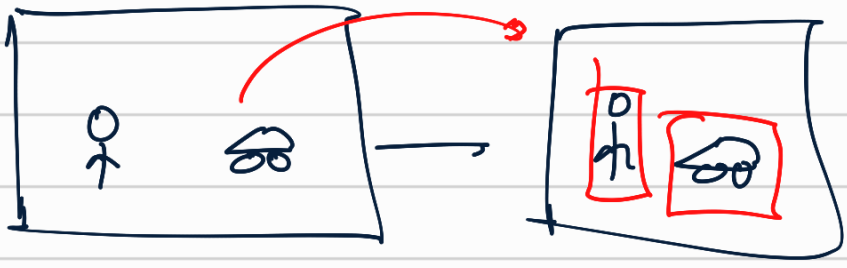


Annotation tools

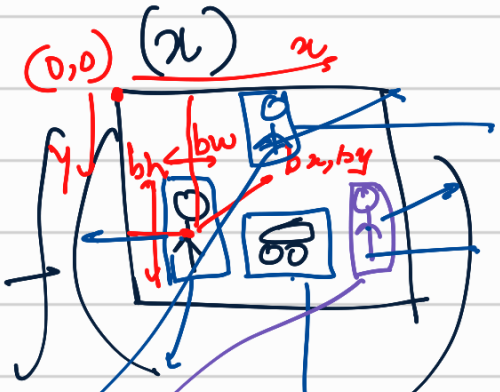
Roboflow

Obj Detn and Loc





Training time:



$\longrightarrow$ y

bbox 1: PTC

$[b_x, b_y, b_h, b_w, 1, 0, 0]$.

Roboflow:

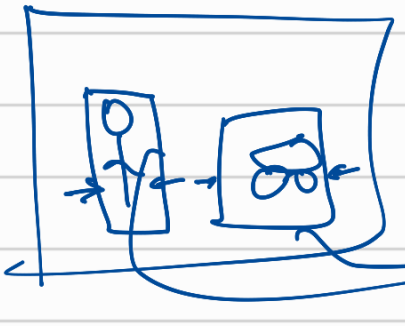
bbox 2:

$[b_x, b_y, b_h, b_w, 0, 0, 1]$

bbox 3 :

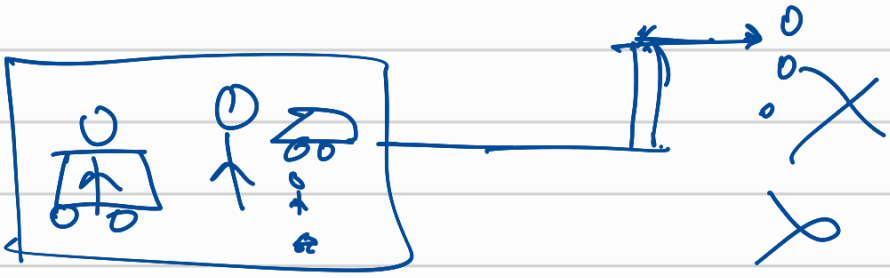
$[b_x, b_y, b_h, b_w, 1, 0, 0]$.

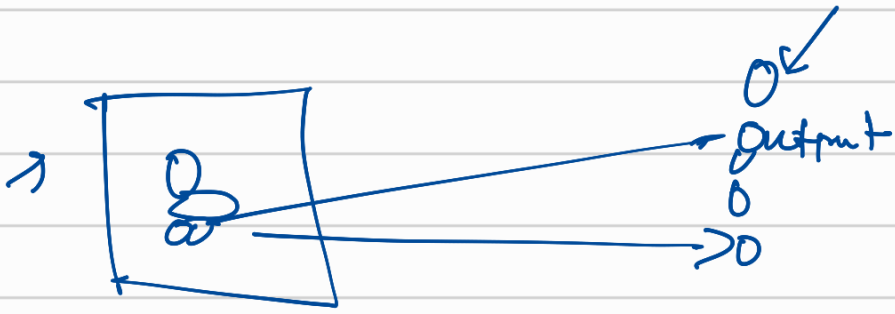
bbox 4 $[b_x, b_y, b_h, b_w, 1, 0, 0]$.



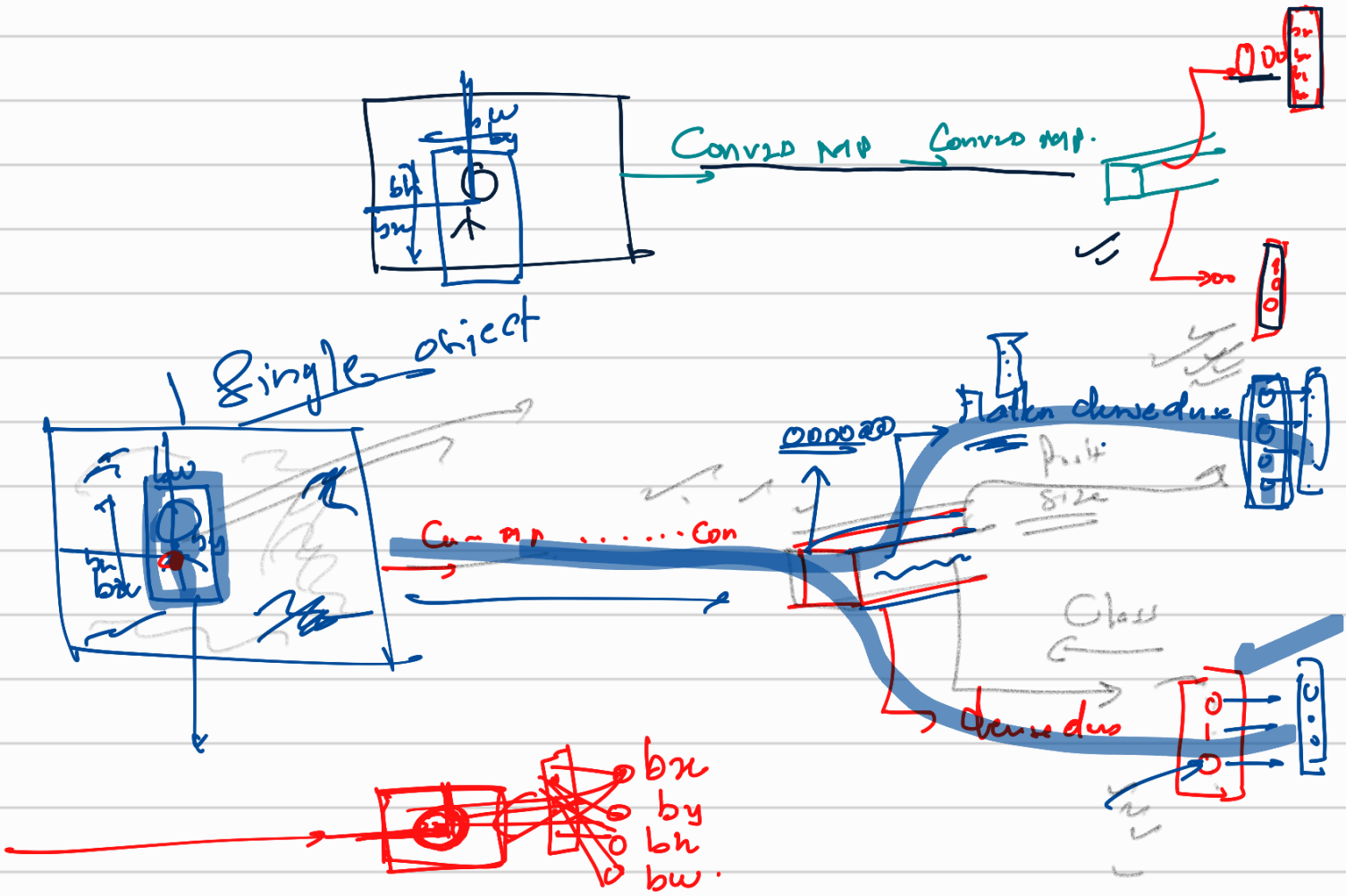
Roboflow.

]





Lets solve the easier problems:



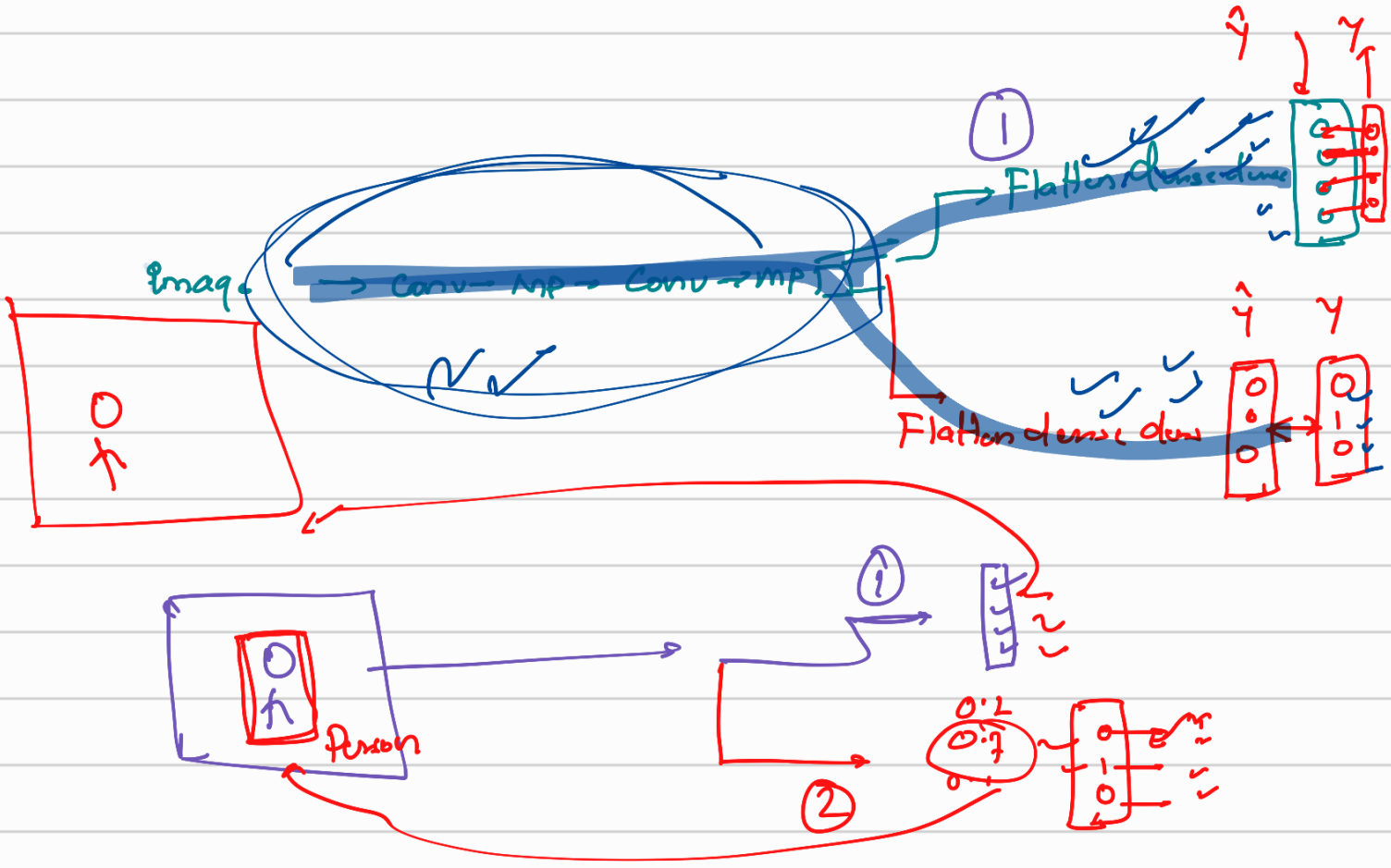
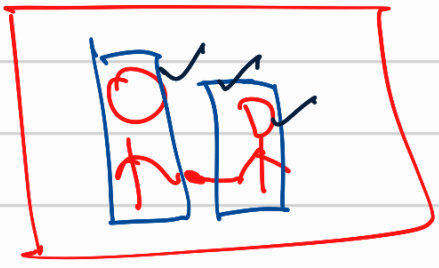
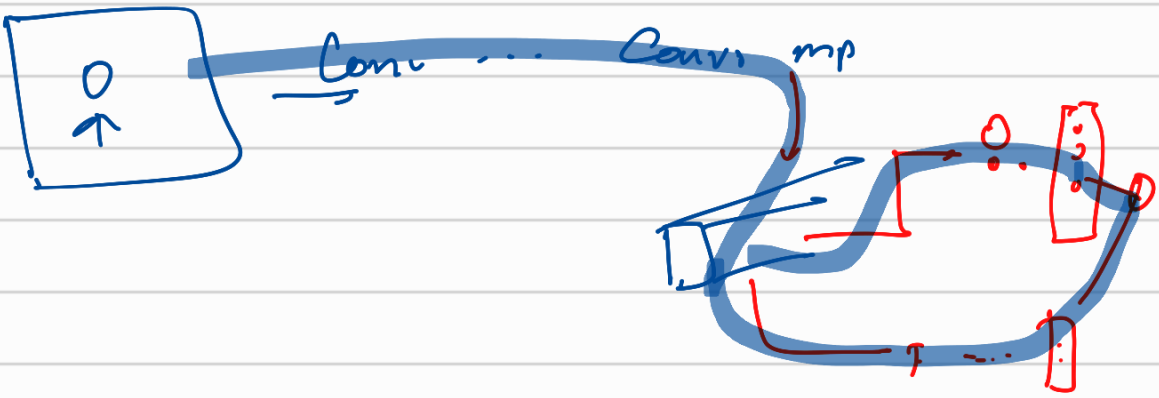
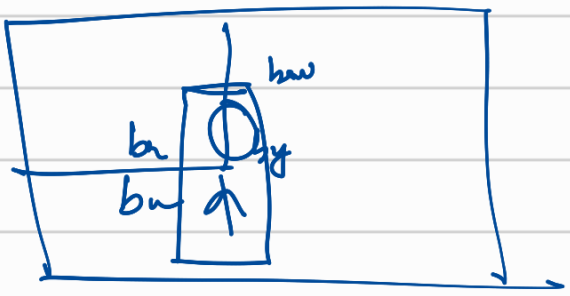


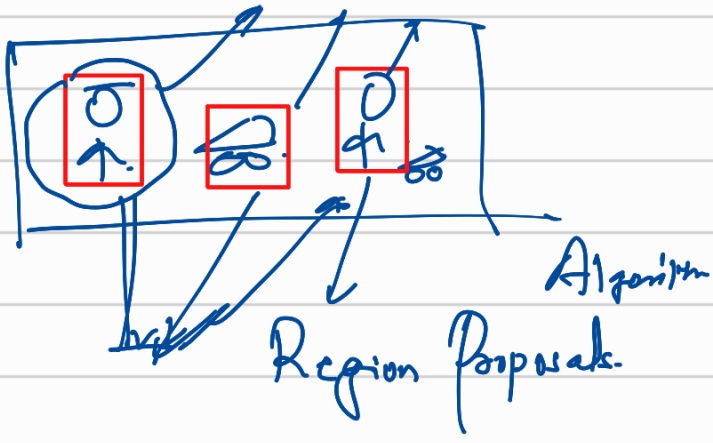
Image
Conv
mp
Conv
mp
1
Flatten
dense
dense
ŷ
y
ŷ
y
Flatten
dense
dense
1
Person
2
0.2
0.7




Conv
Conv
mp

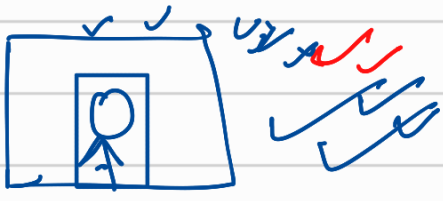


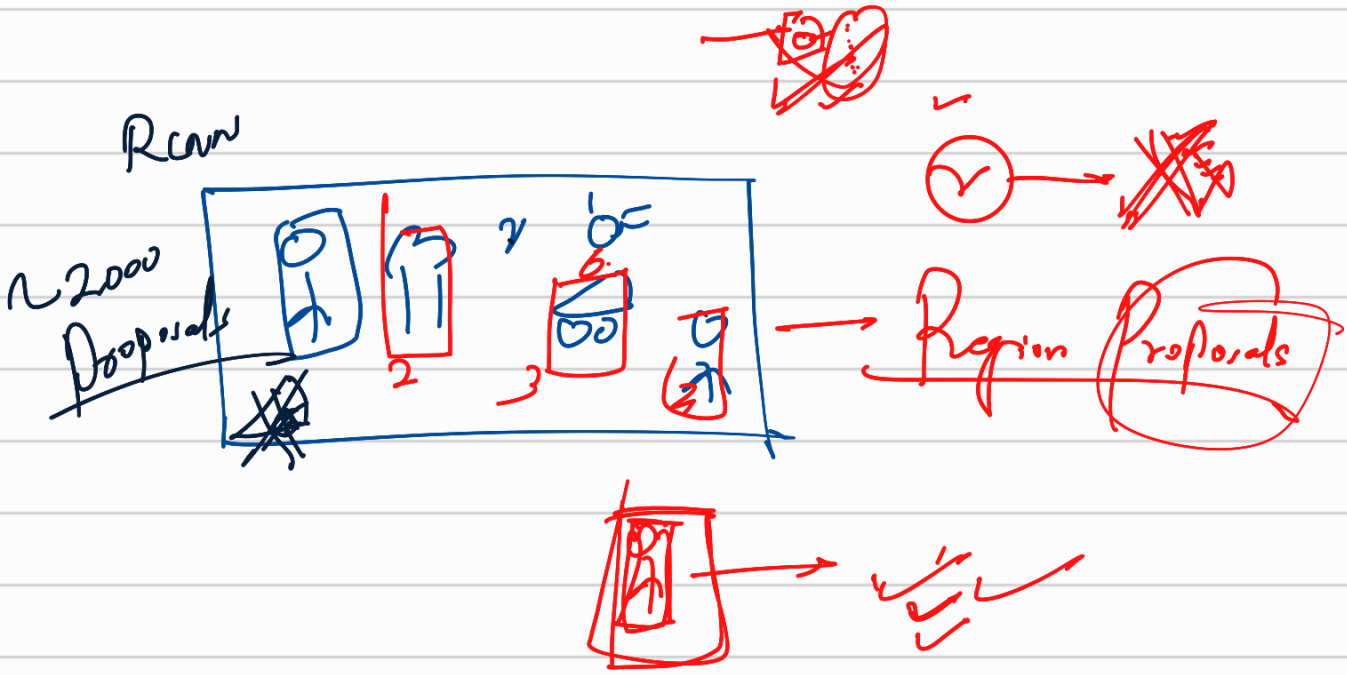
Architecture



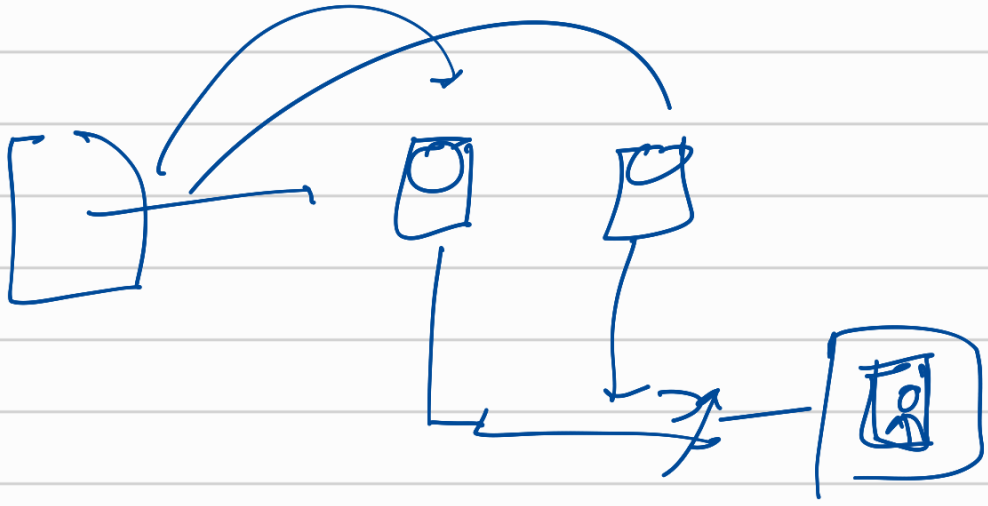
Algorithm

Region Proposals





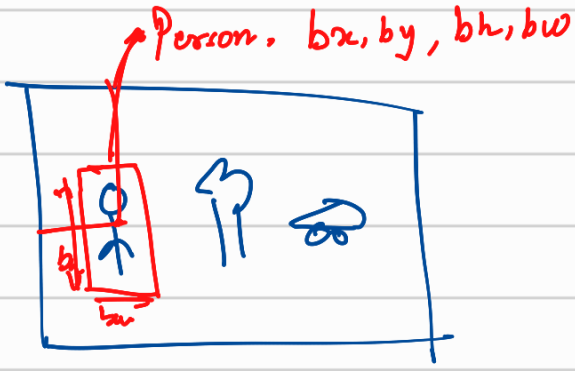
Os 2 Stage

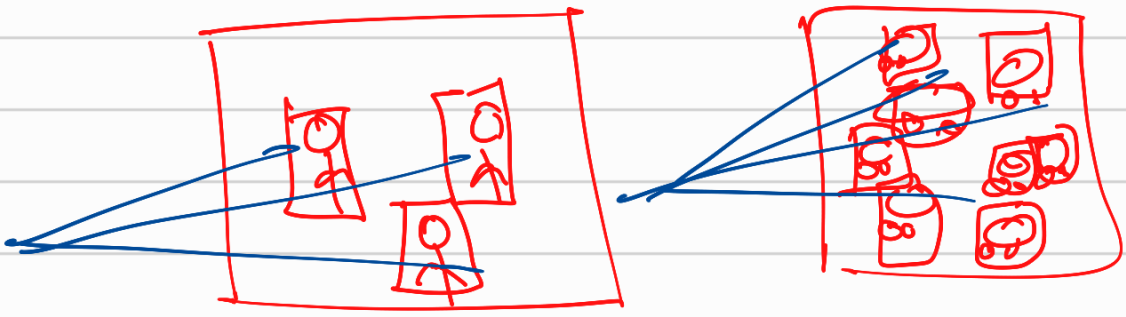


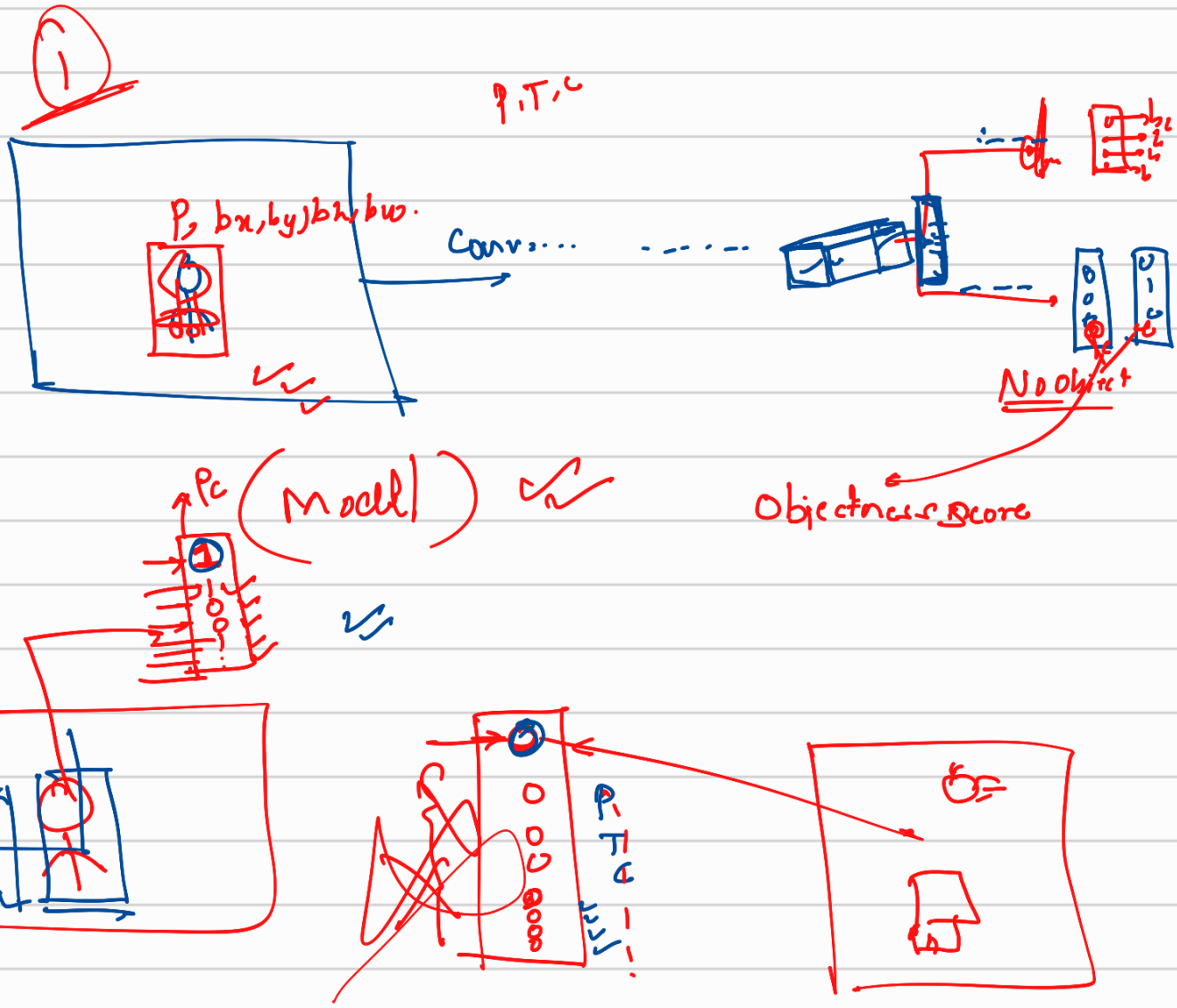
2000

R CNN, Fast R-CNN, Faster R-CNN

YOLO — 1 Stage


Person, bx, by, bh, bw



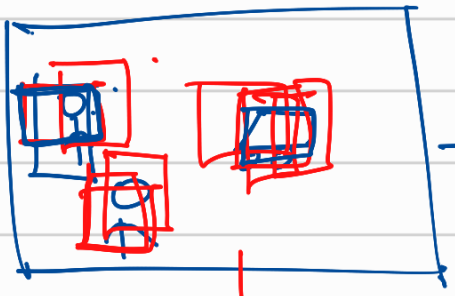

1
P, bx, by, bh, bw.
Conv....
P, T, C
No Object
Objectness Score
Pc
Model
2

Two stage Detection Algorithm

① Region Proposals — Fast RCNN

↓

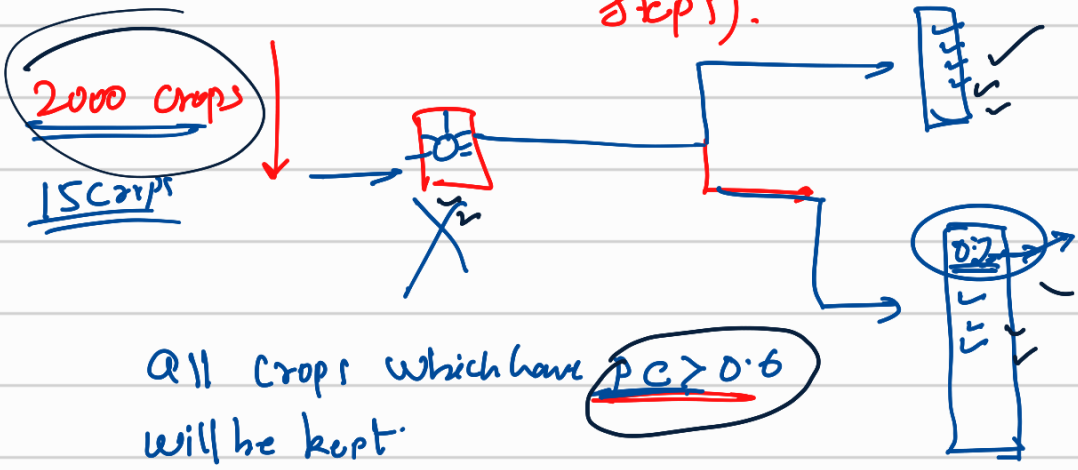
Selective Search



→ Suggest "crops" wherein you may find an object

~500 / 2000 crops

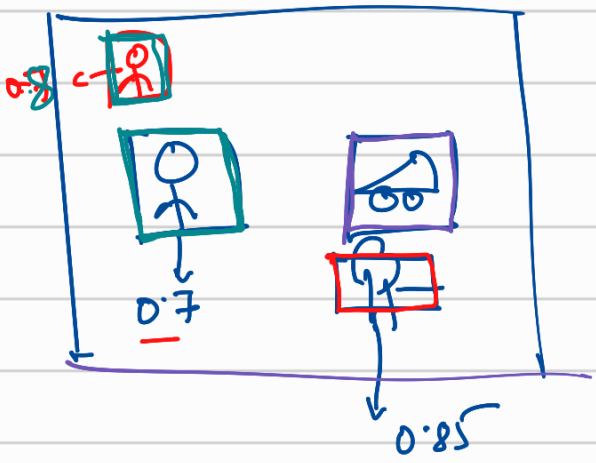
model (which has been trained by Step 1).



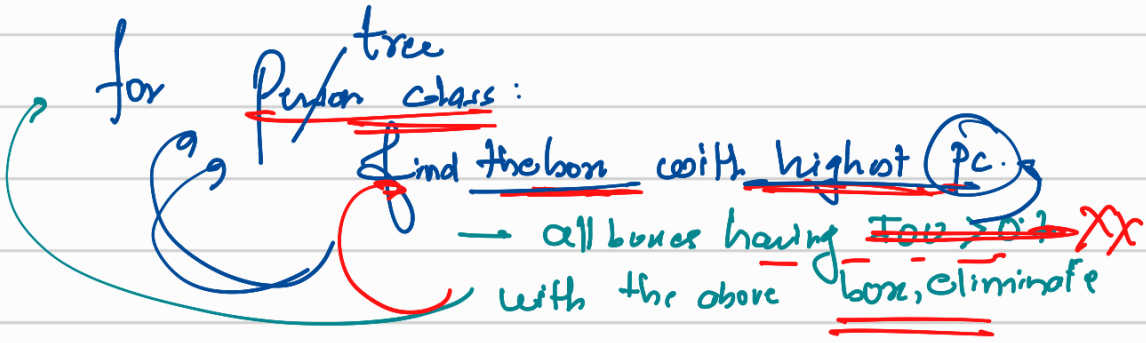
All crops which have $pc \geq 0.6$ will be kept.

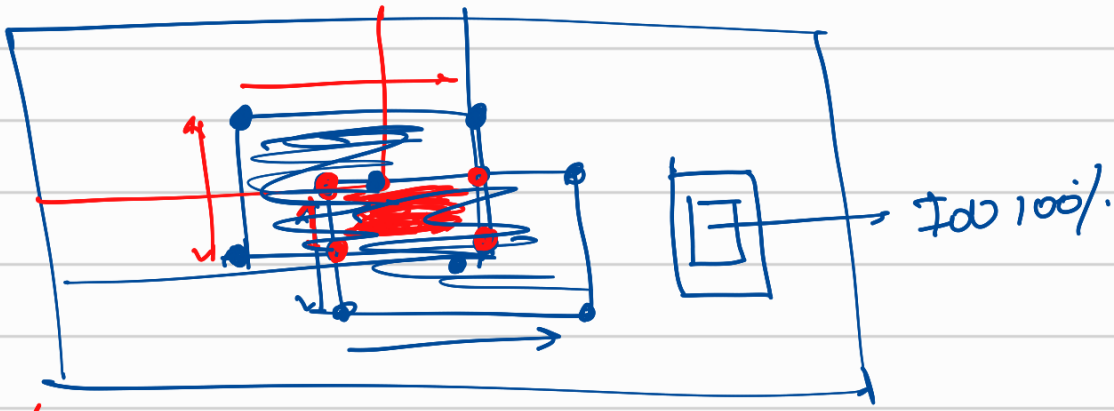
Post Processing : Non-max suppression

IOU → Intersection over Union

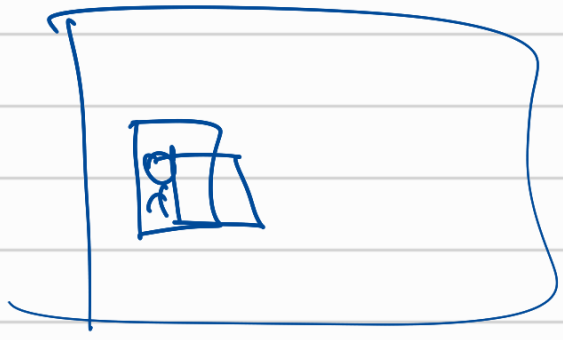

0.8
0.7
0.85

NMS:


for Person class in tree:
find the box with highest pc.
→ all boxes having IoU > 0.5
with the above box, eliminate

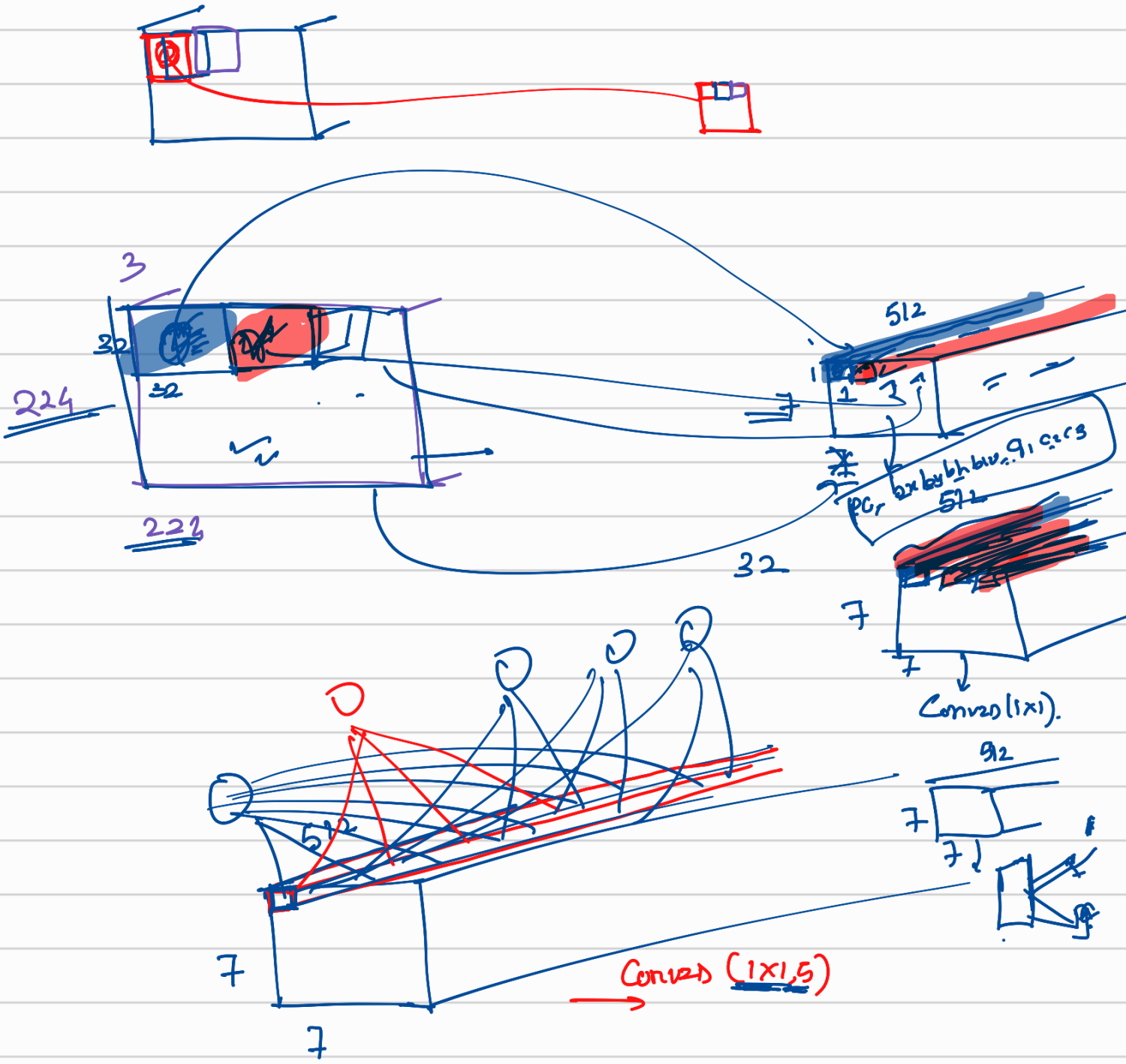

A
A
A
Ideal

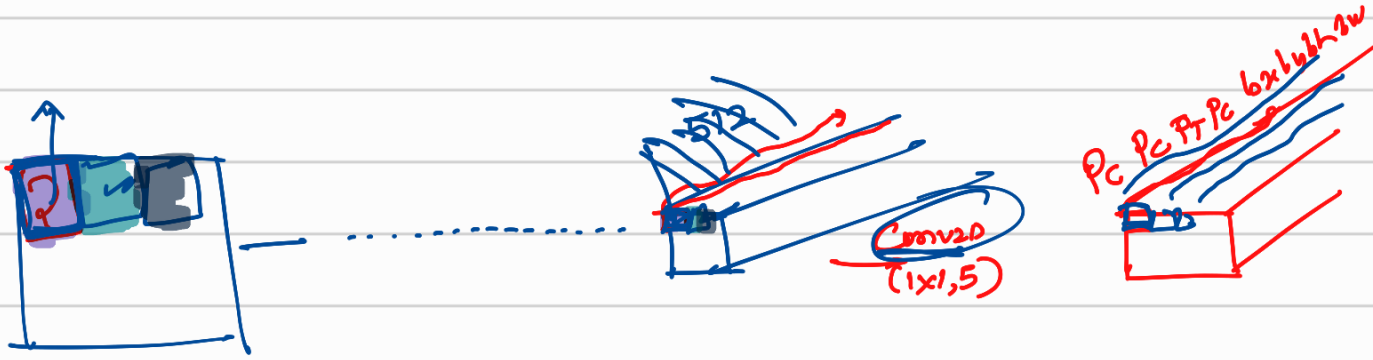
= = / =

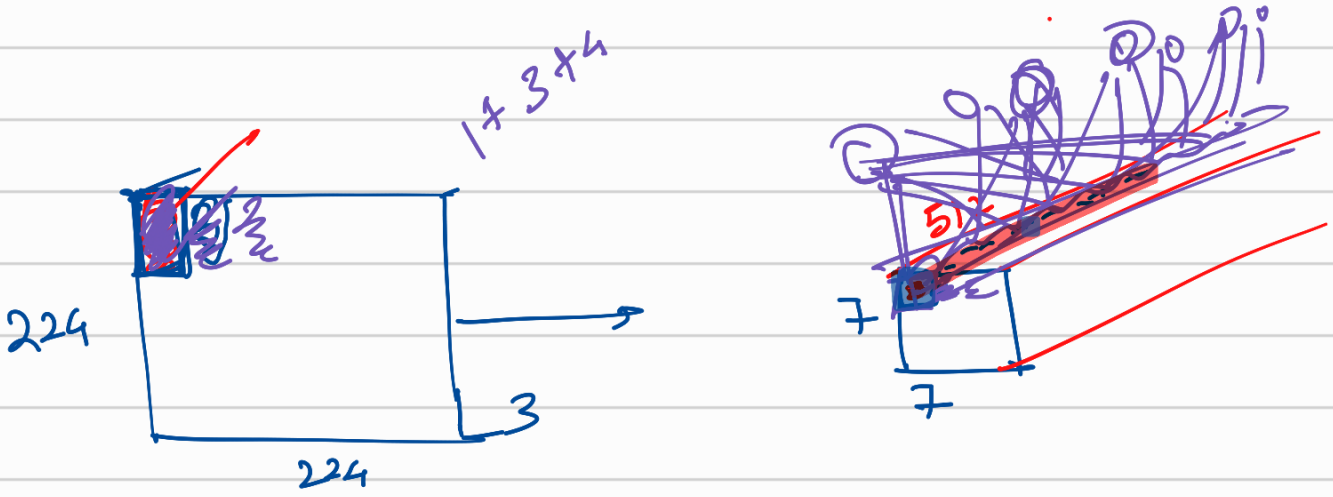


FASTER RCNN

Convolution implementation of Sliding window



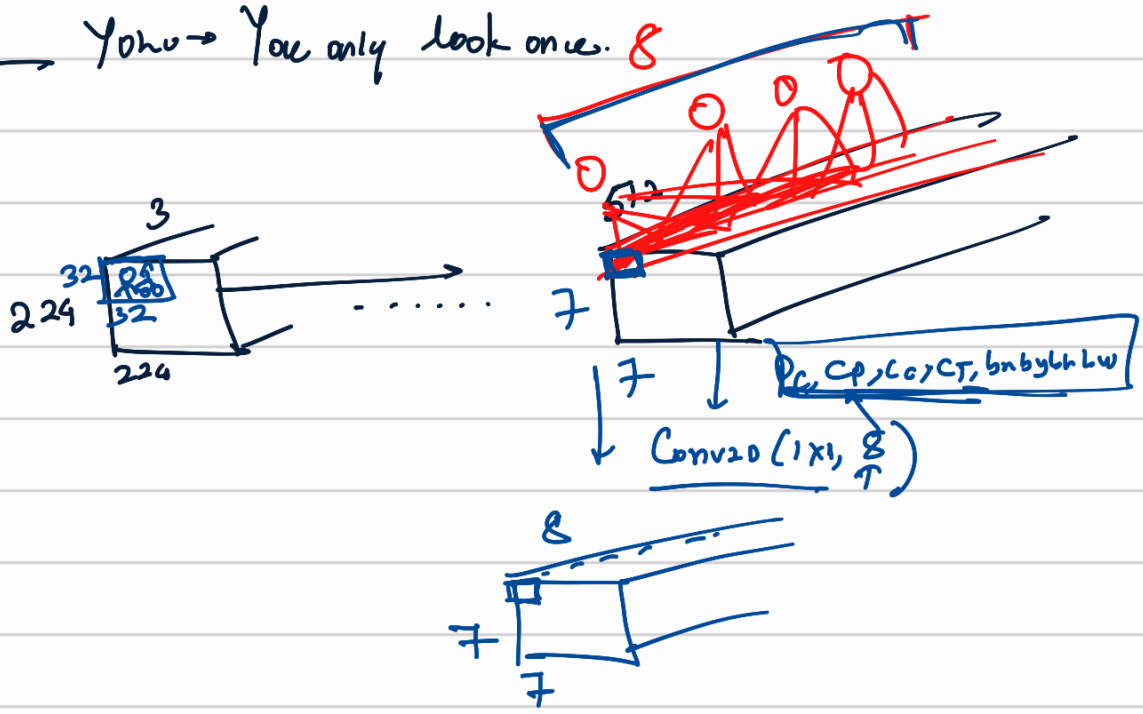

Conv20
(1x1,5)


1+3x4
224
224
3
7
7

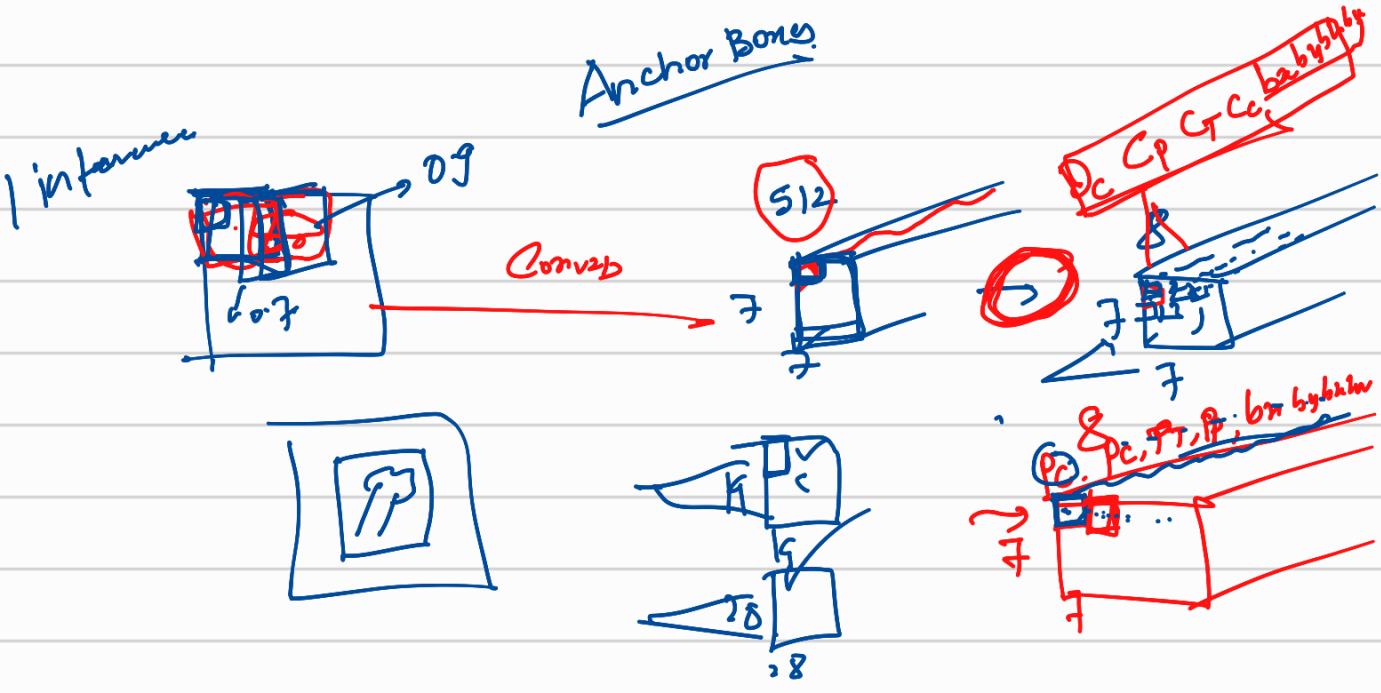
Faster RCNN

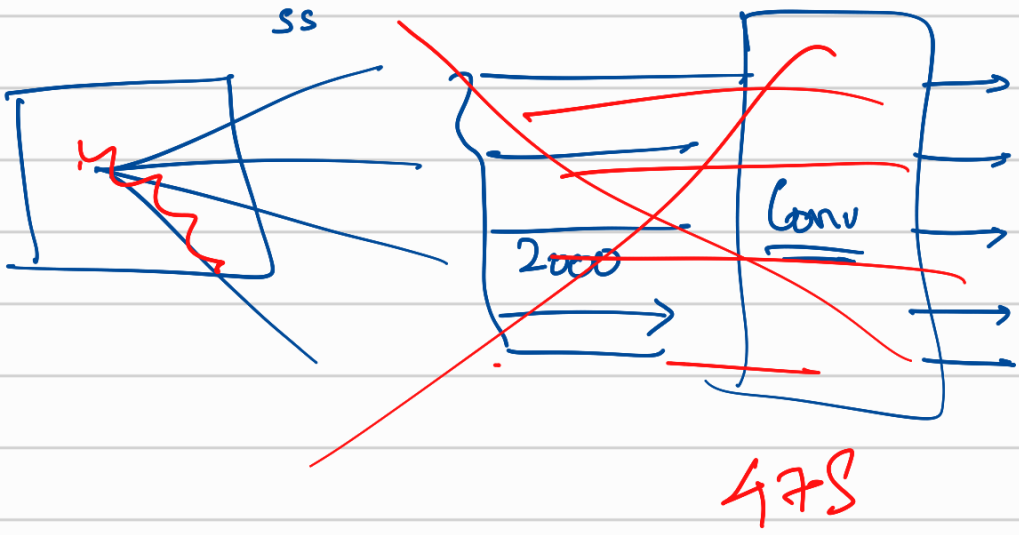
YOLO — Single Stage

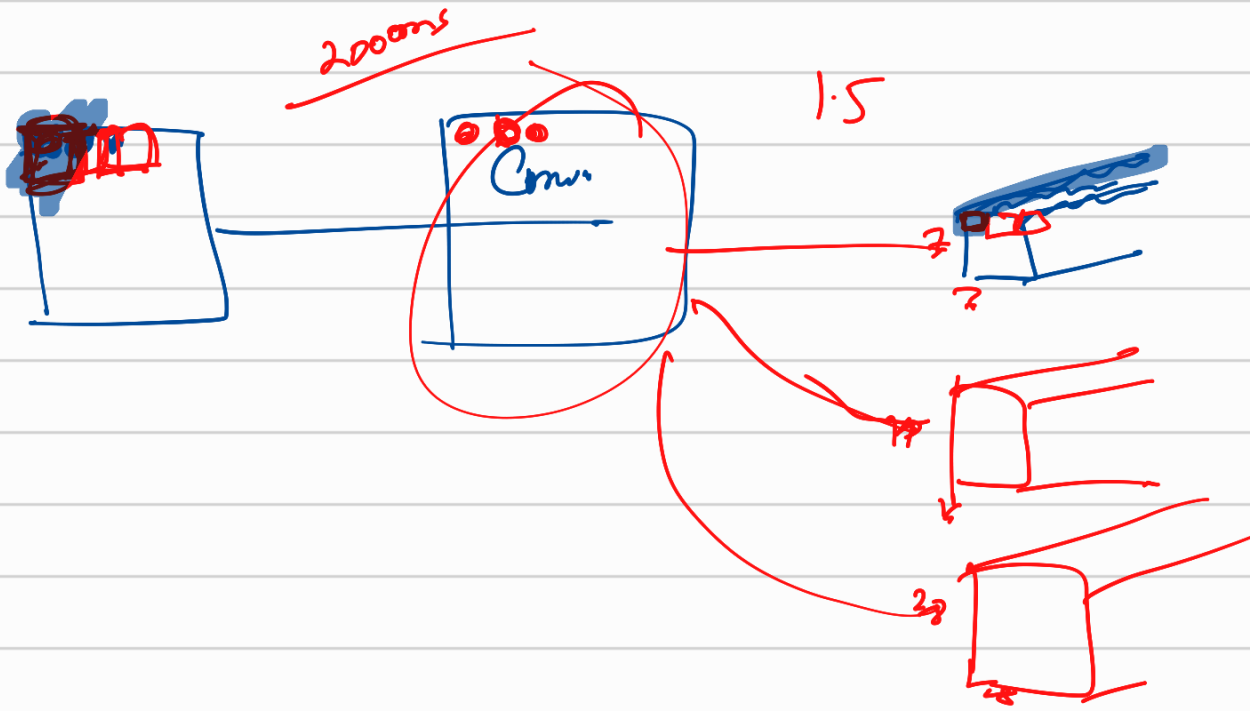
↳ Yolo → You only look once.

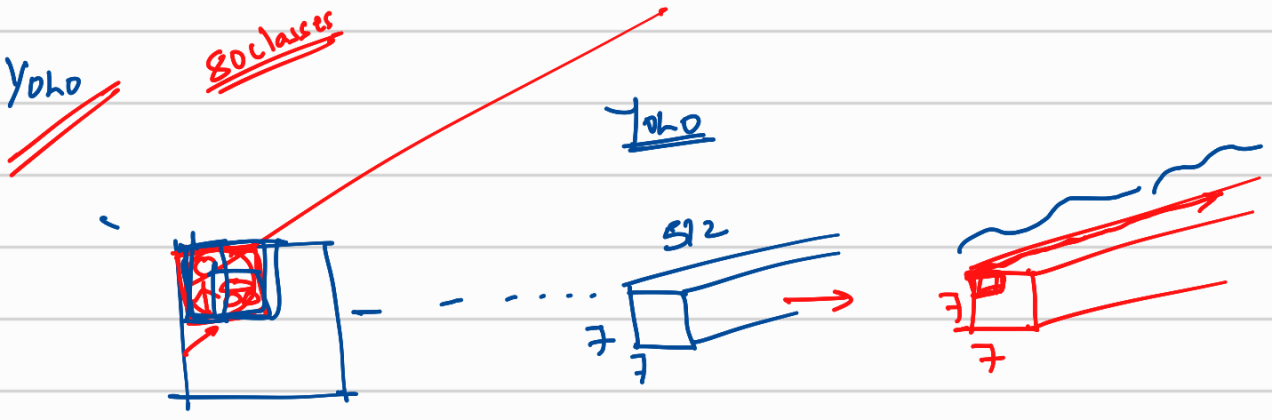


Anchor Boxes



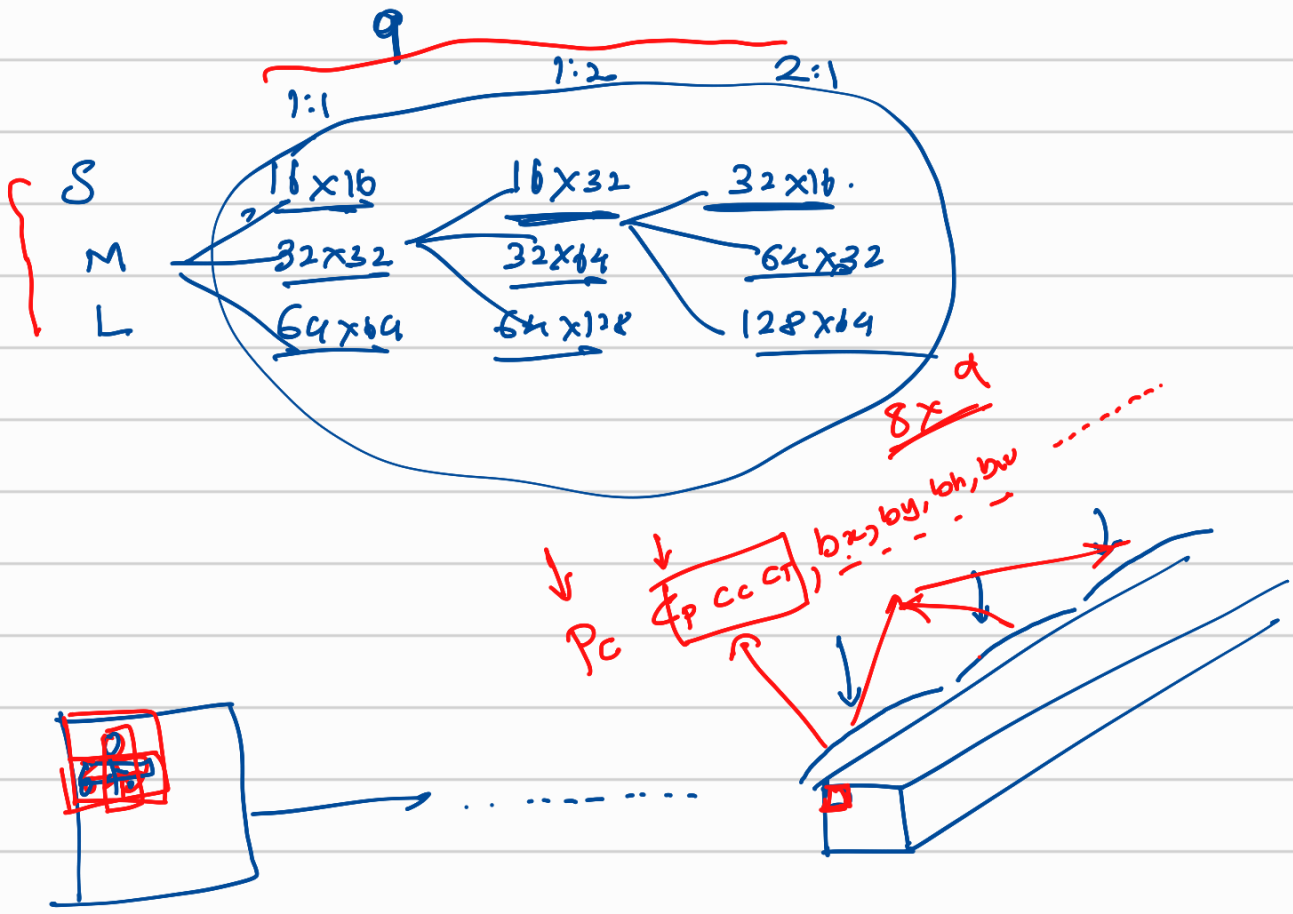

SS
Conv
2000
475

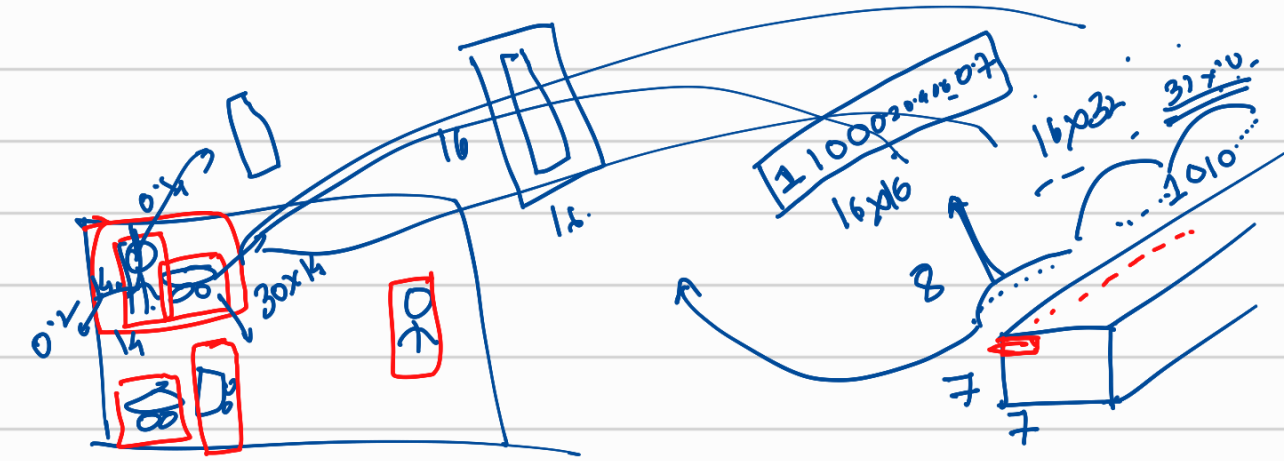

2000ms
1.5
Conv
7
7


Yolo
80 classes
Yolo
512
7
7
7
7

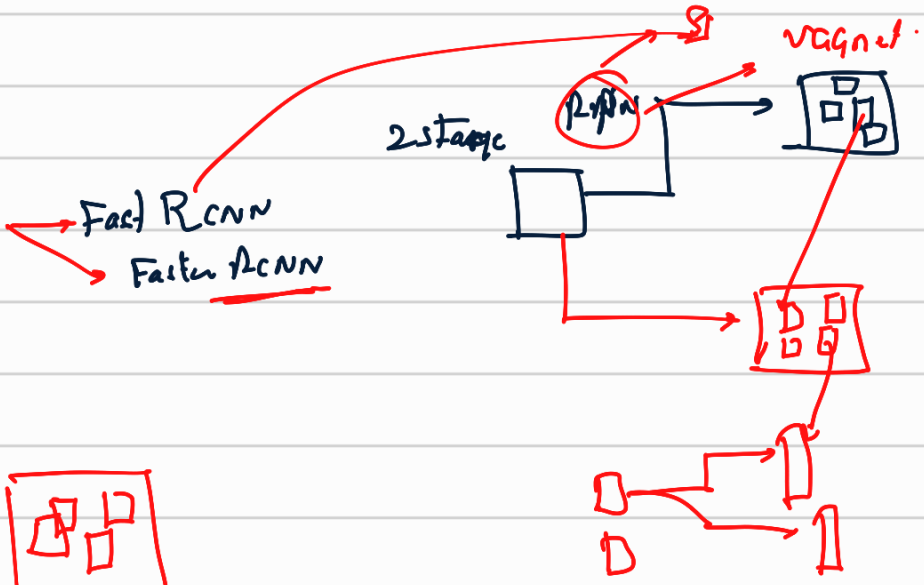
GT

Cluster

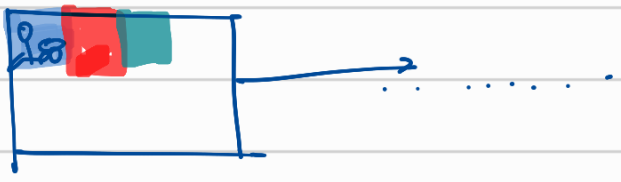

9
1:1
1:2
2:1
S
M
L
16x16
16x32
32x16
32x32
32x64
64x32
64x64
64x128
128x64
8x
Pc

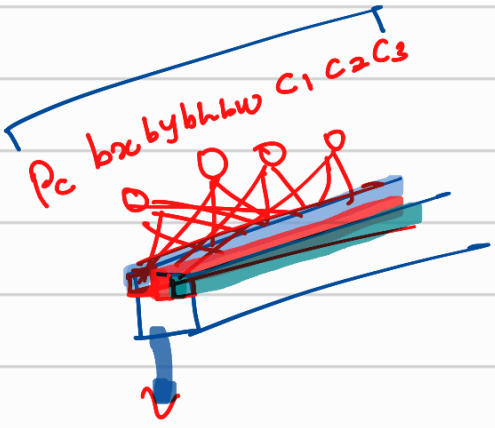

16
16x32
16x16
8
7
7

YOLO



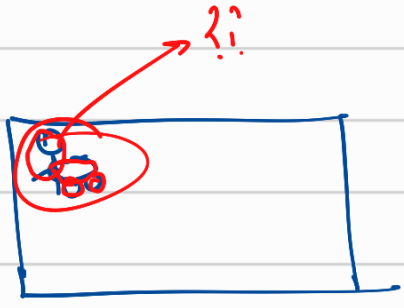
Yolo (Single Stage)

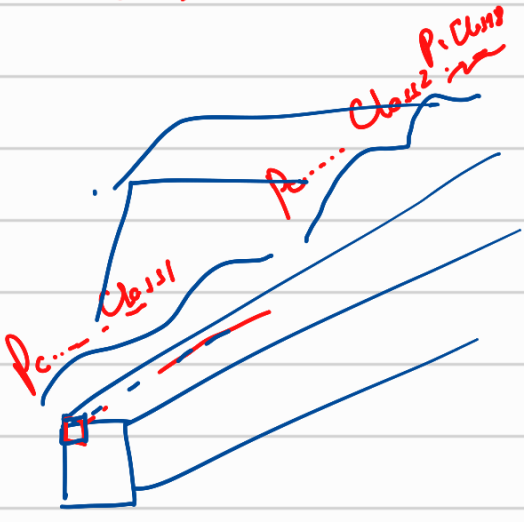


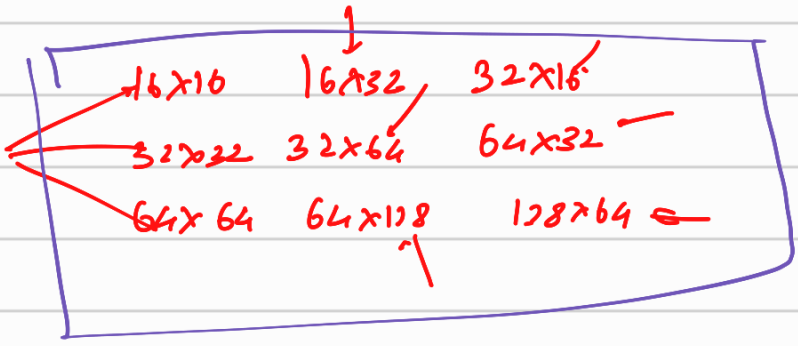


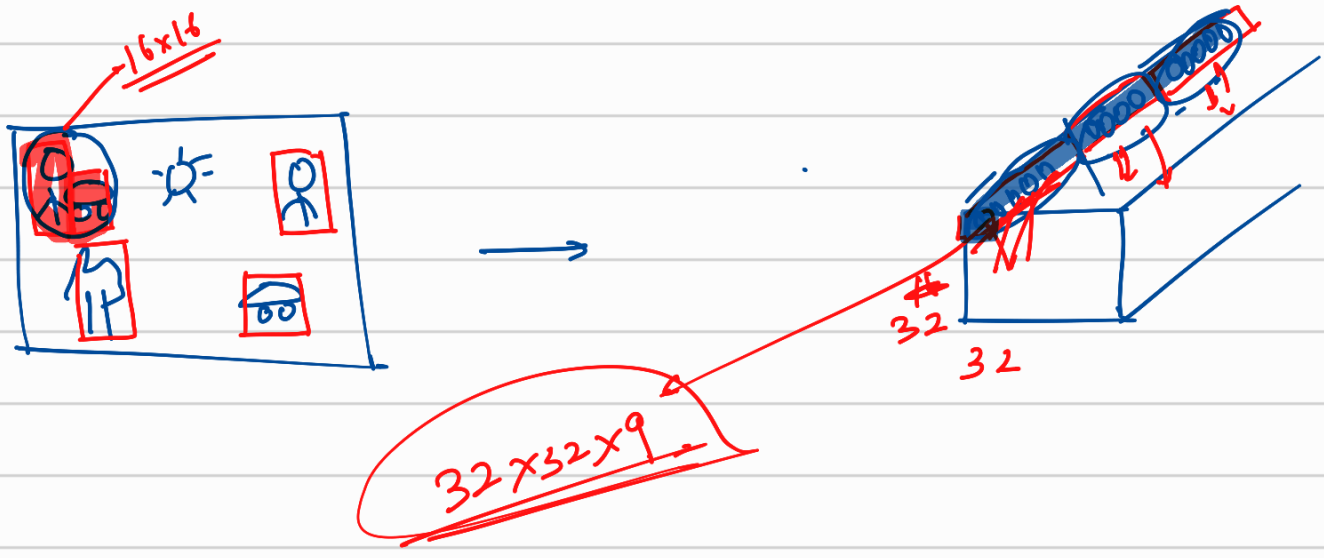
Conv2D (1×1)

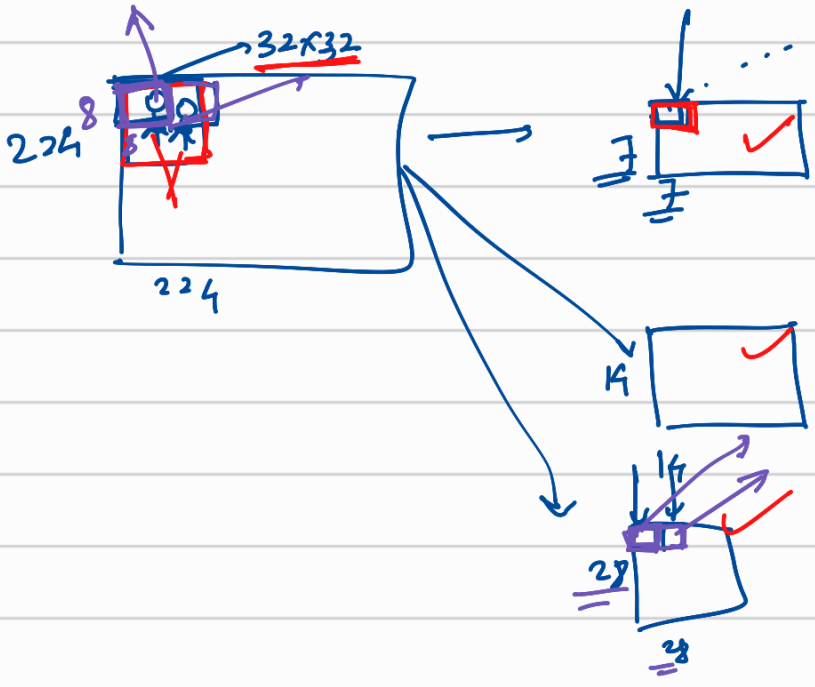
Anchor Boxes

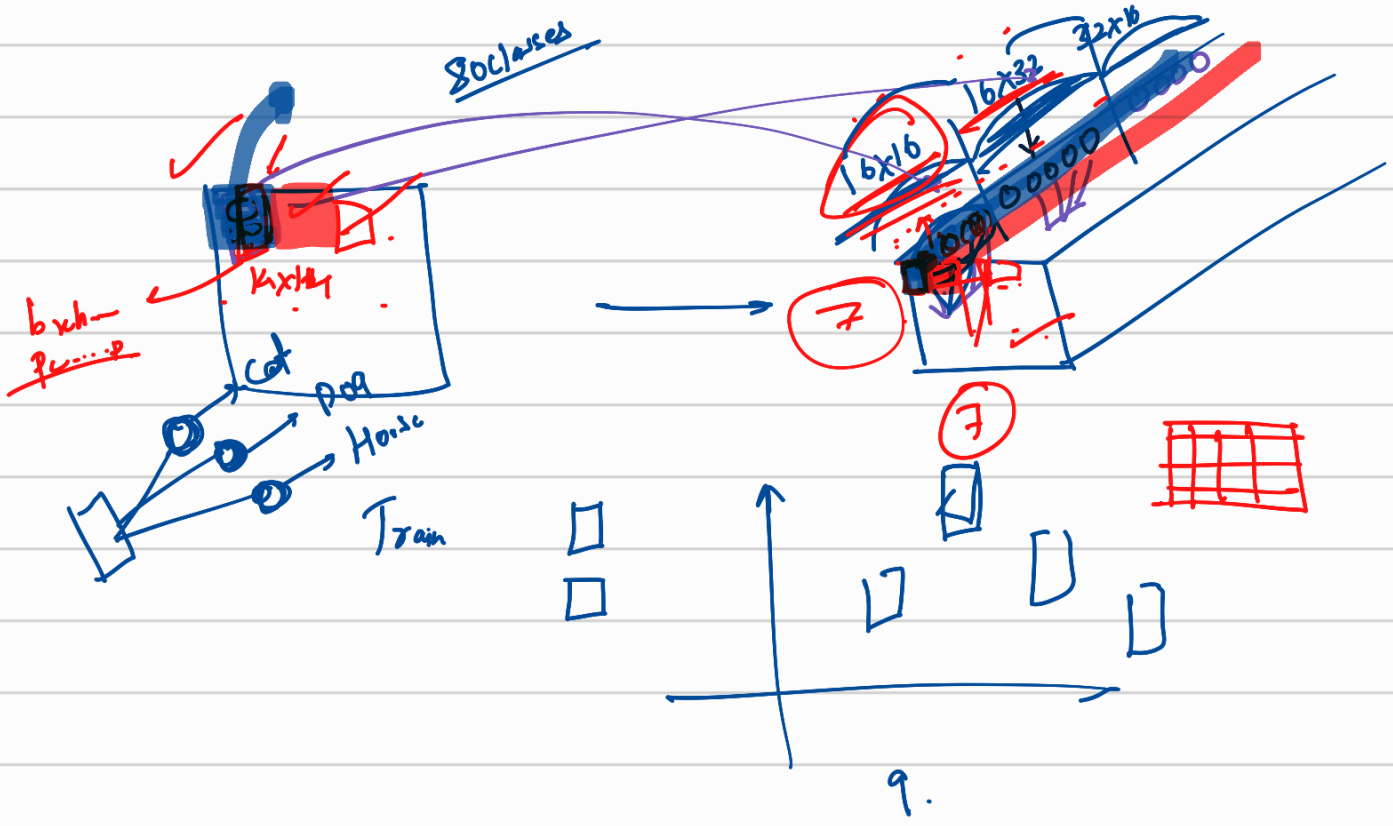





16x16
16x32
32x16
32x32
32x64
64x32
64x64
64x128
128x64


16x16
32
32
32x32x9

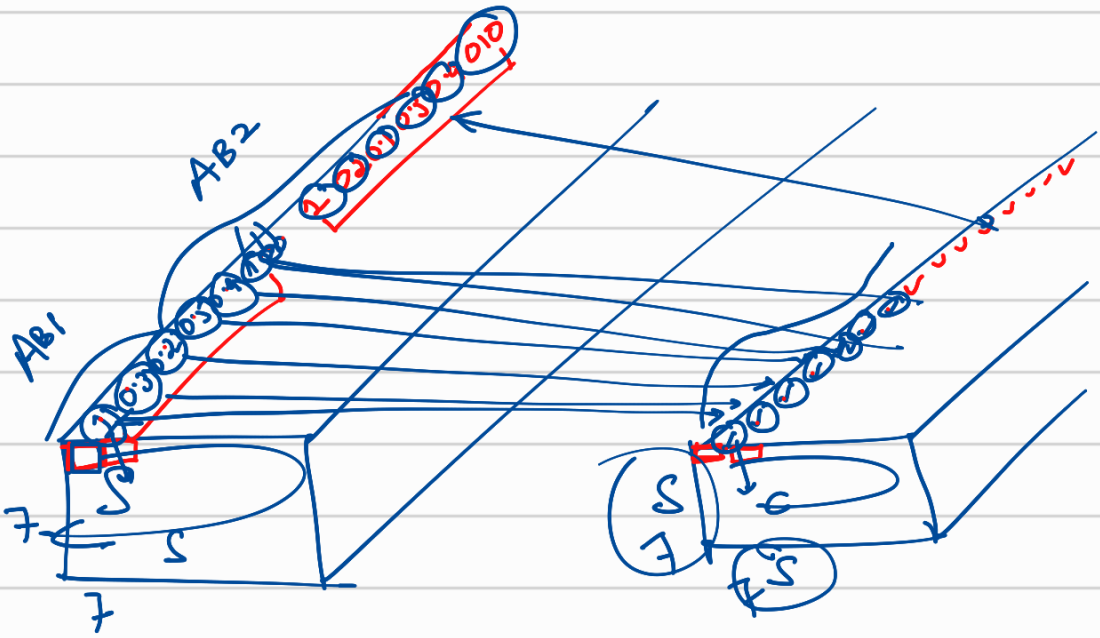

32x32
224
224
14
28
28

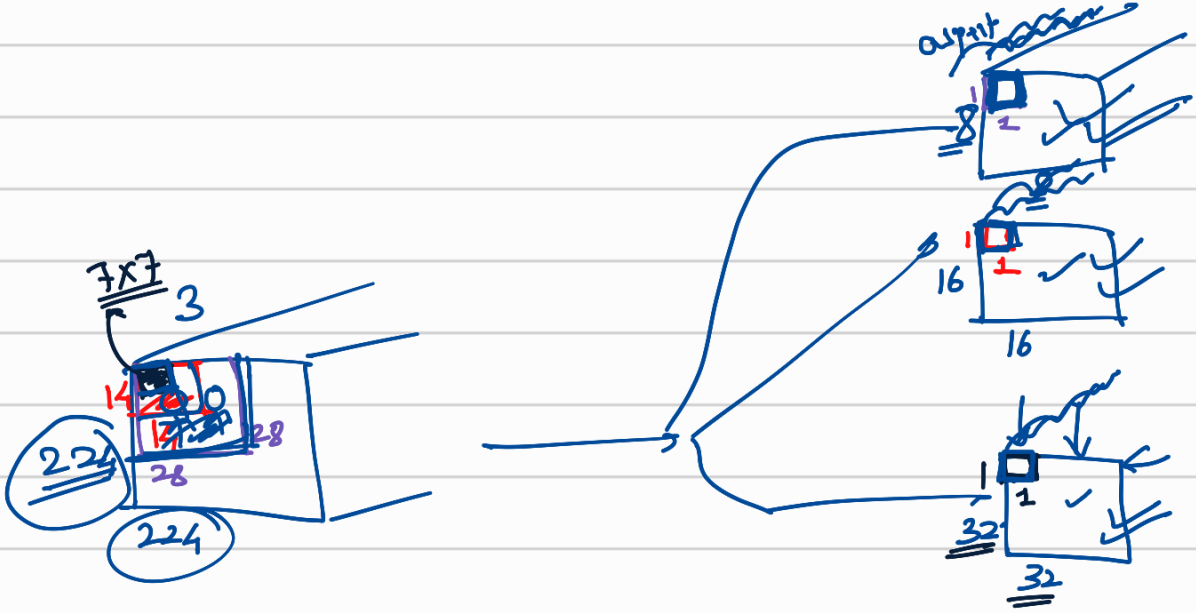


Pc bx by bh hw C_1 C_2 C_3 C_4

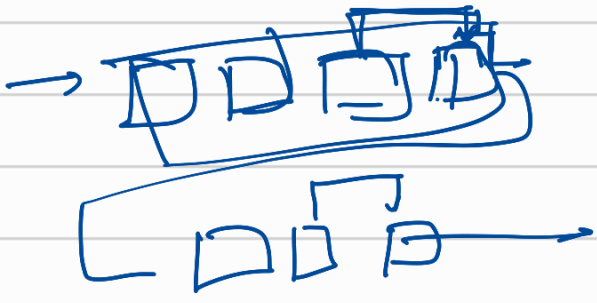
Break!

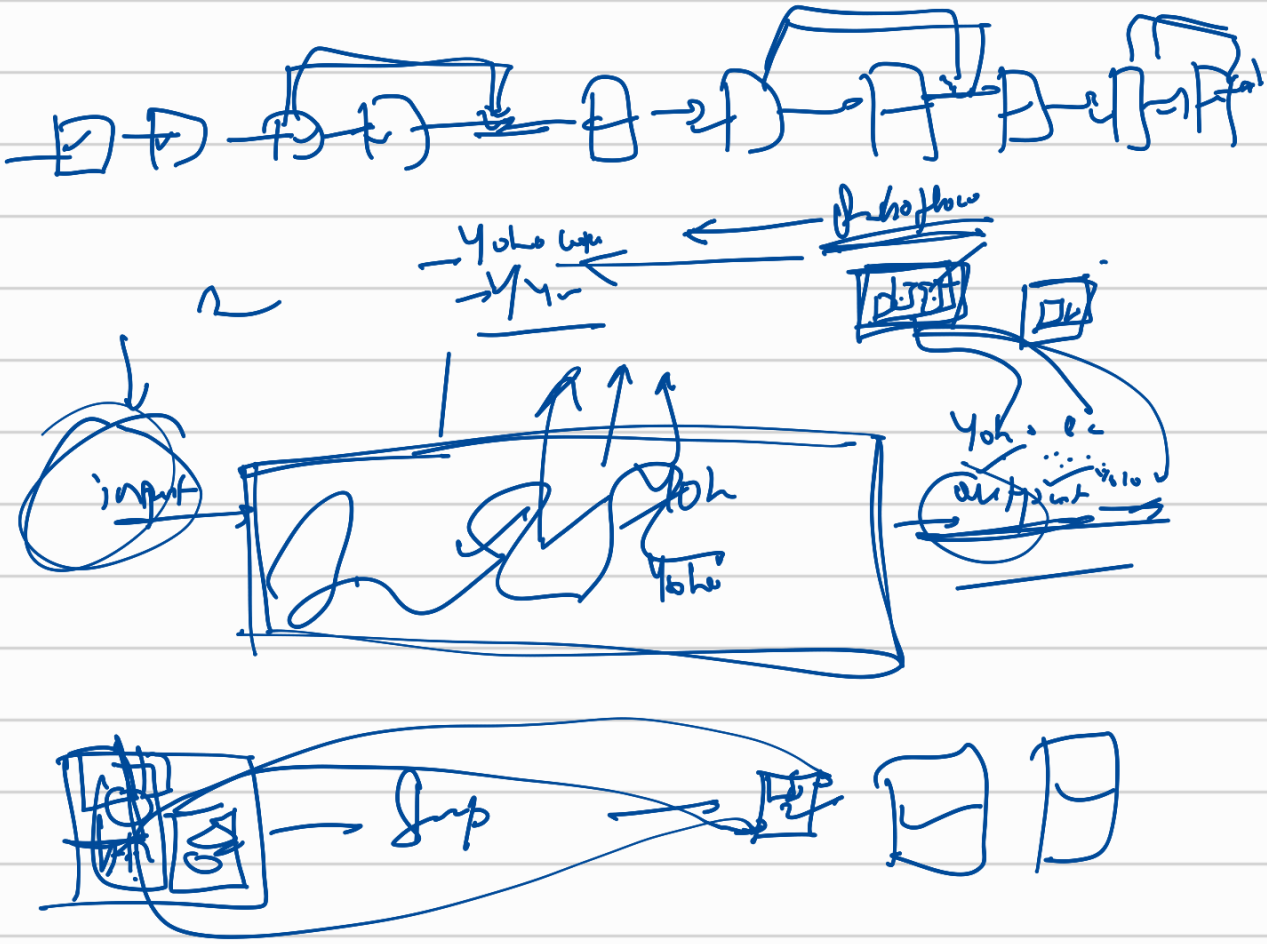
YOLO V3 Loss

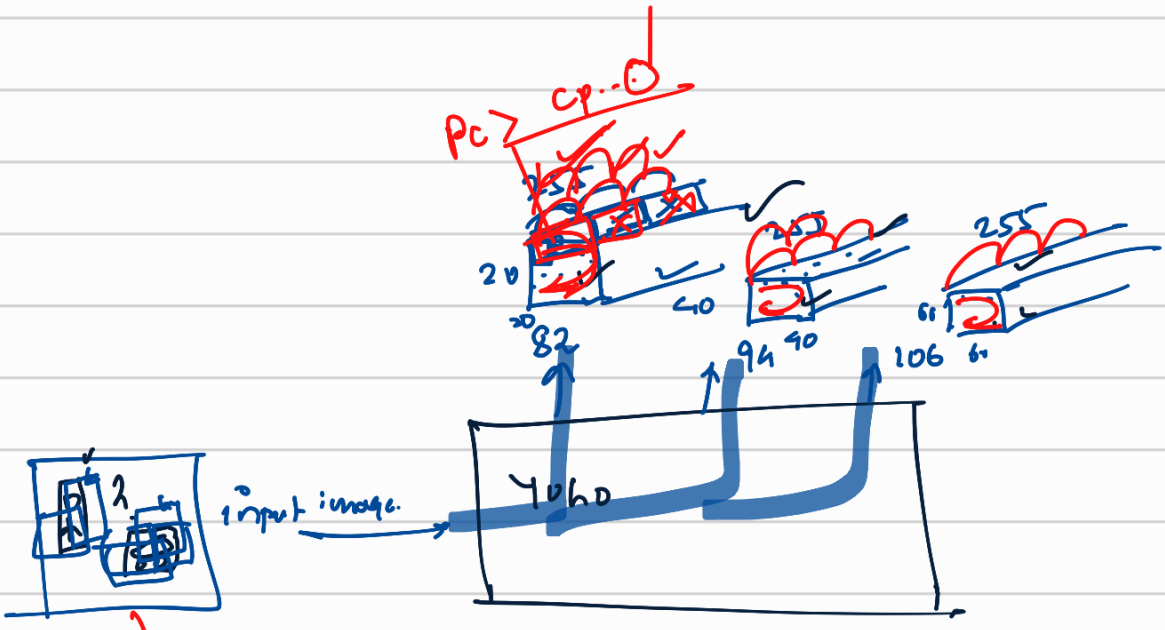




Yolo







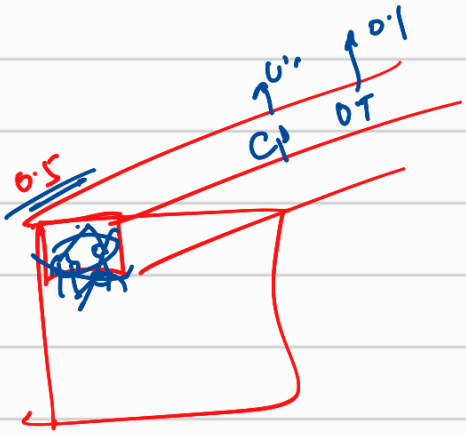
① $P_c > 0.6$ ✓

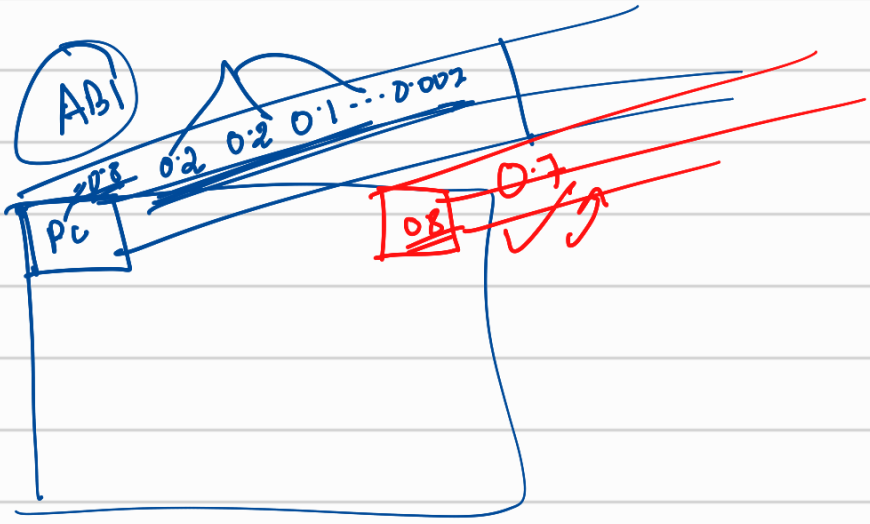
② np.argmax(Class) → Person

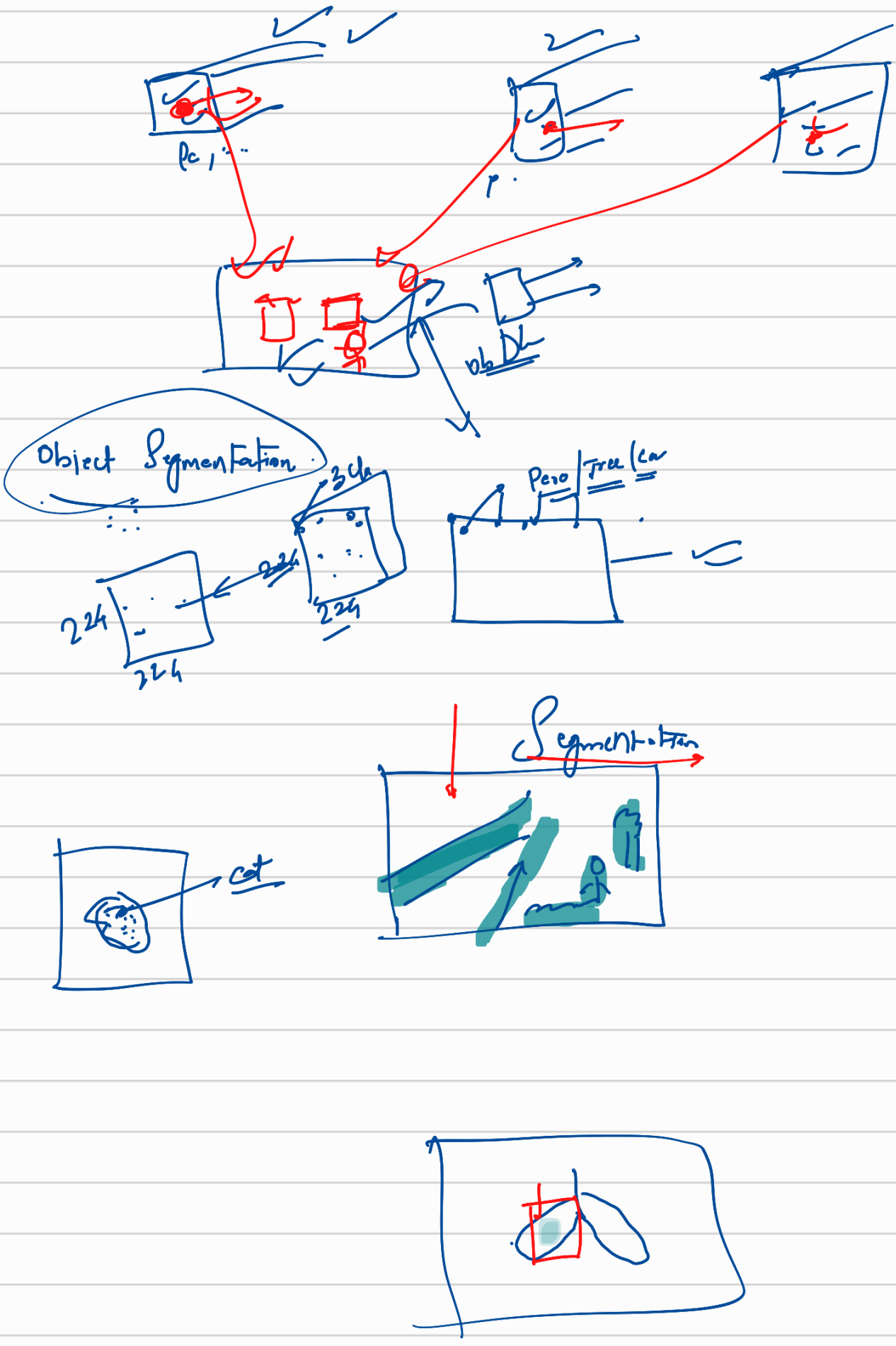
③ $C_p > 0.1$ ✓ AB

④ Bbox activated

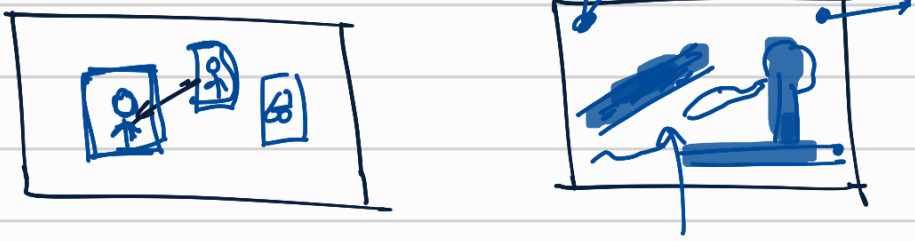
→ Nms

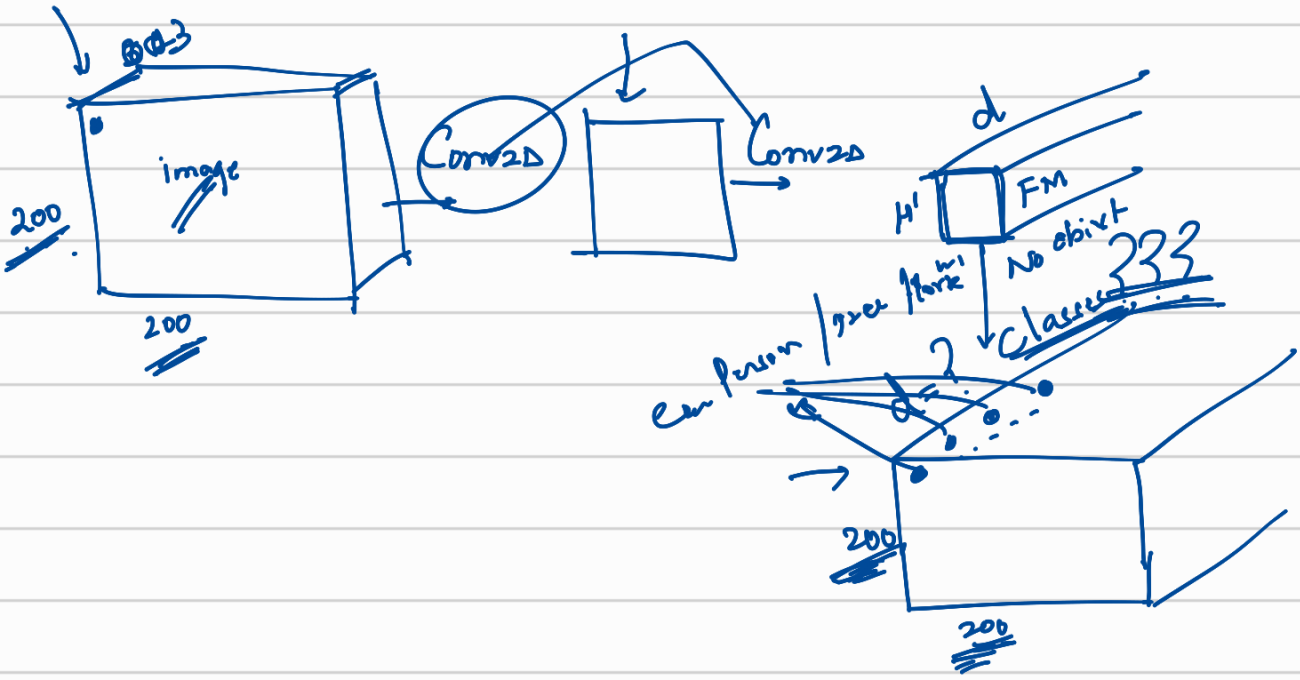





Object Segmentation
224
224
224
Pero
cat
Segmentation

Object Segmentation

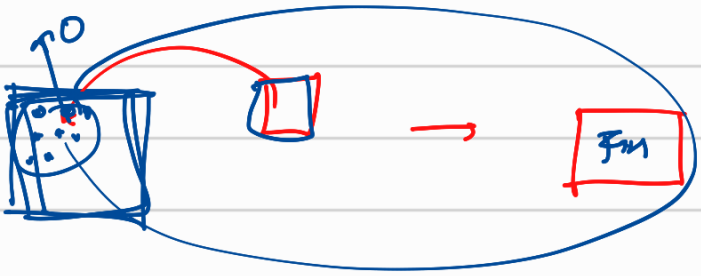


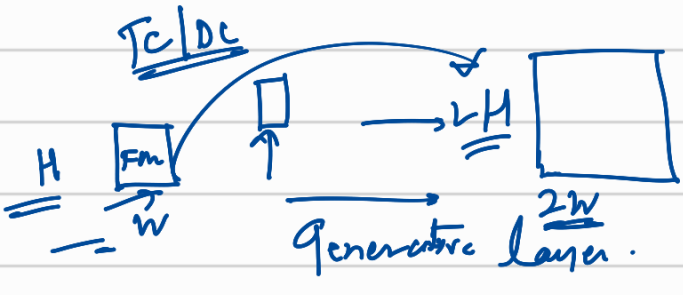


FCN — Fully convolution nets

Unet

Deconvolution / Transpose Convolution.

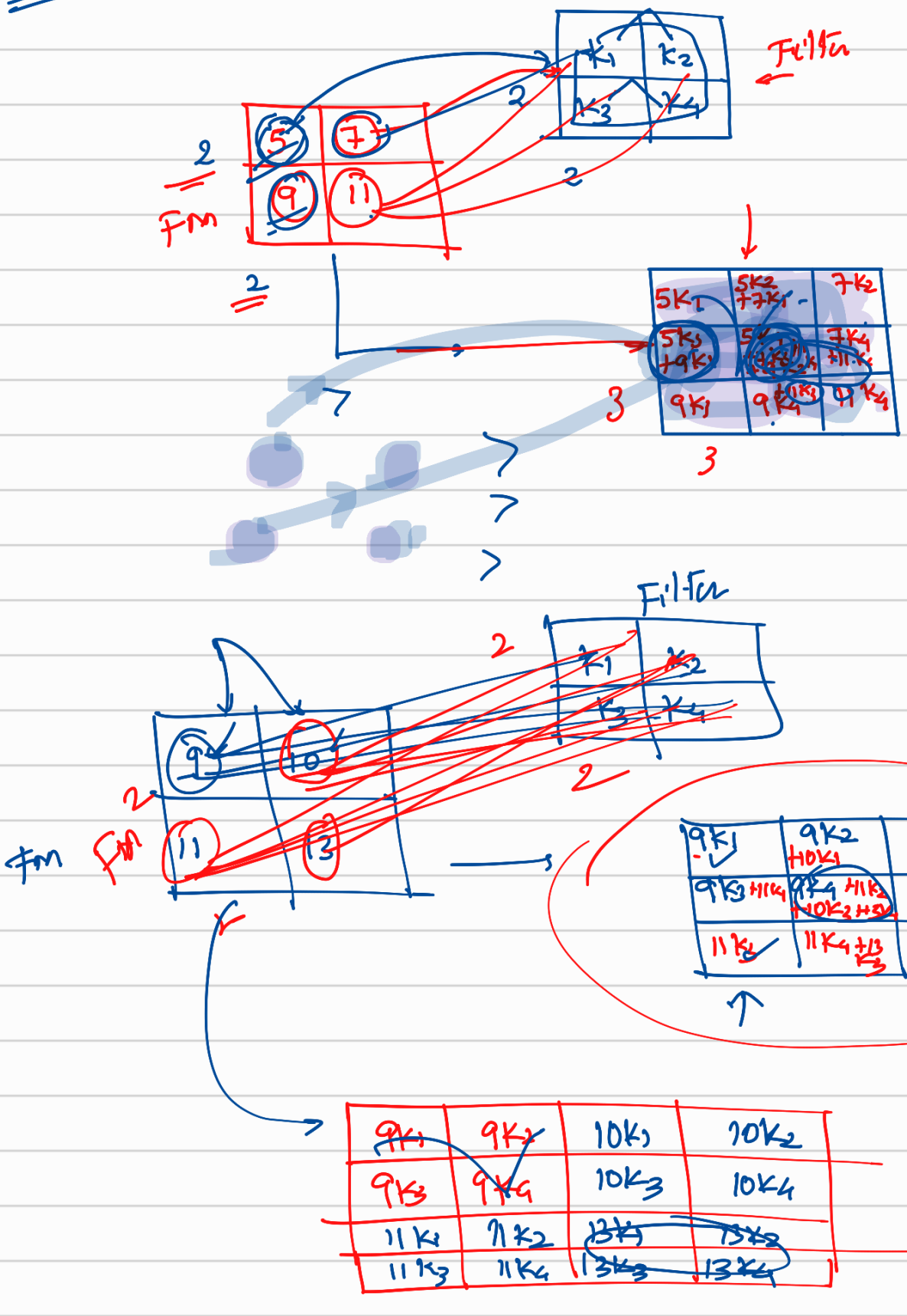




IC Transpose convolution

① Upsampling 2D.

IC

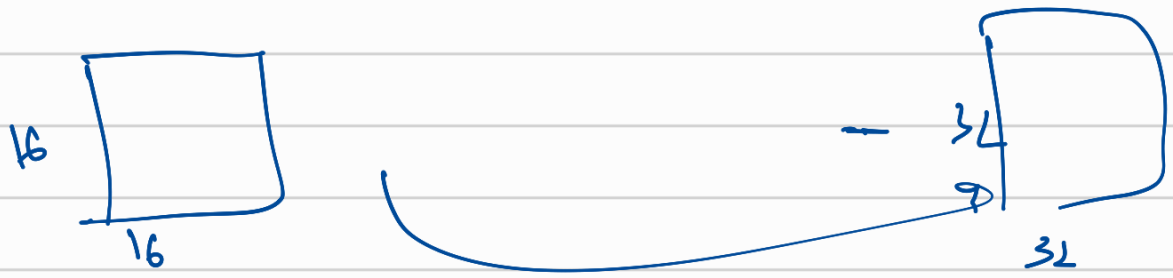


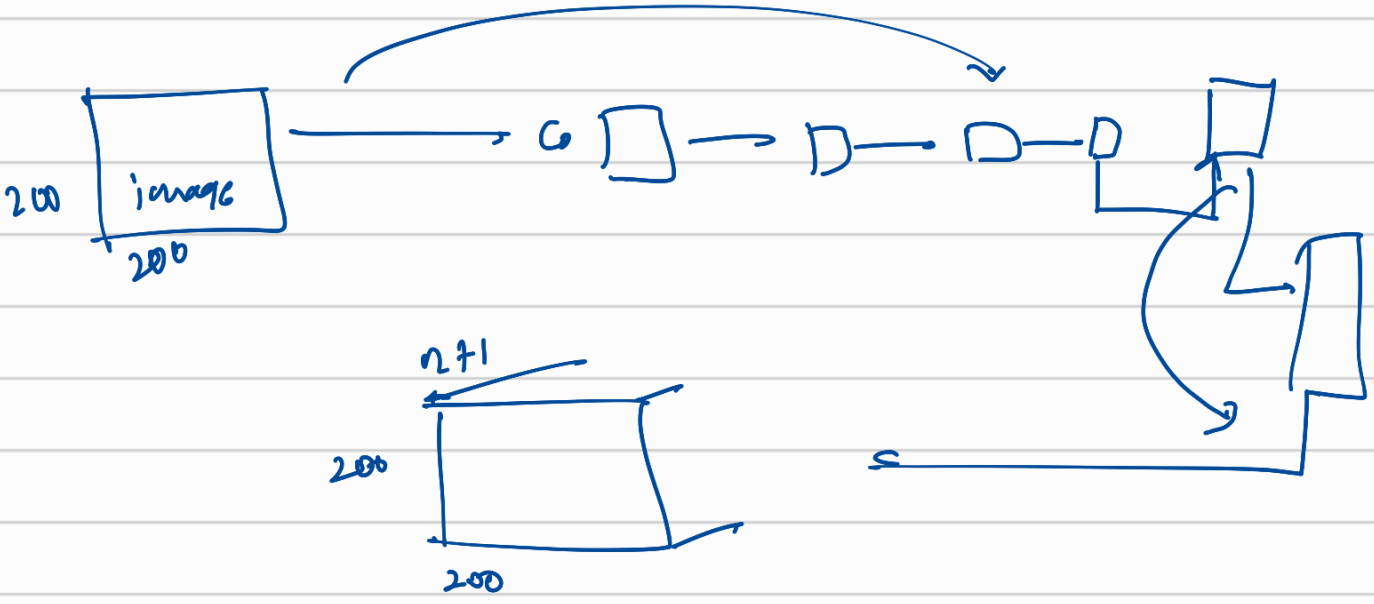
$9K_1$	$9K_2$	$10K_1$	$10K_2$
$9K_3$	$9K_4$	$10K_3$	$10K_4$
$11K_1$	$11K_2$	$13K_1$	$13K_2$
$11K_3$	$11K_4$	$13K_3$	$13K_4$

4x4

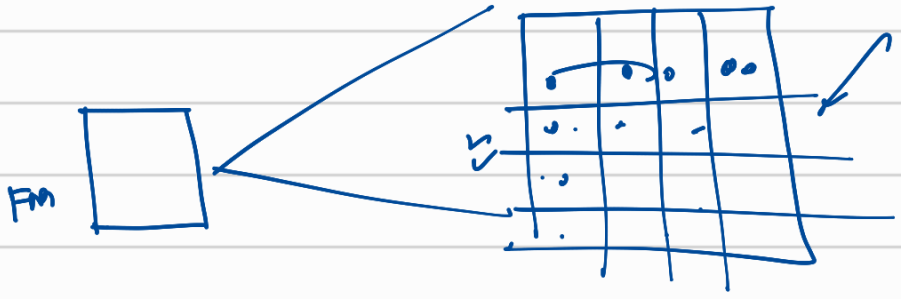
F=2x2

S=2x2

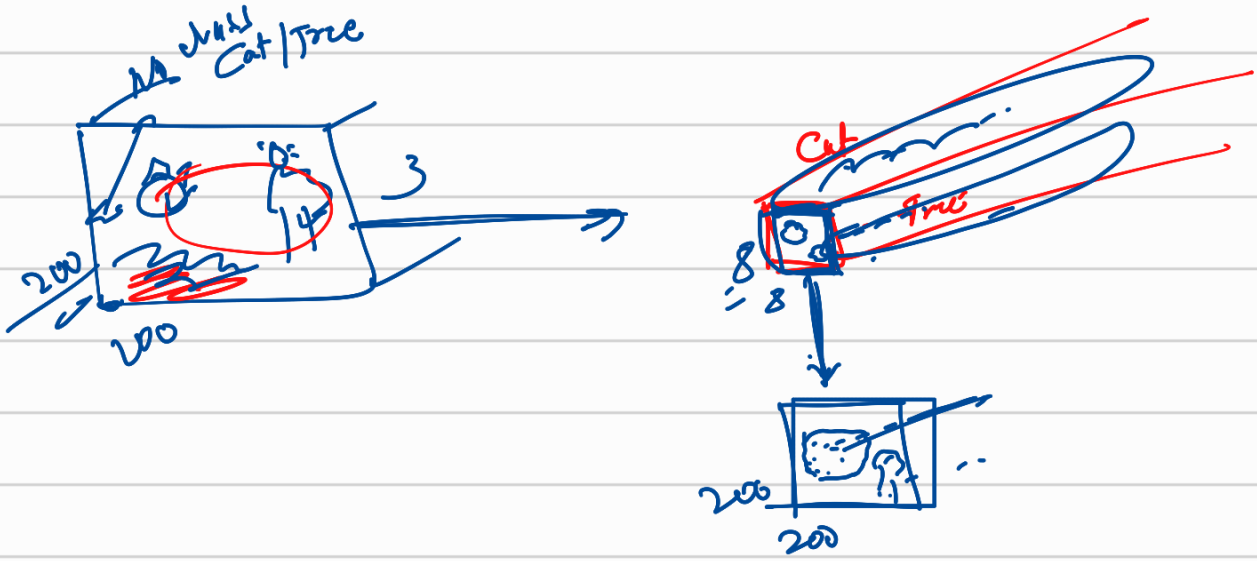


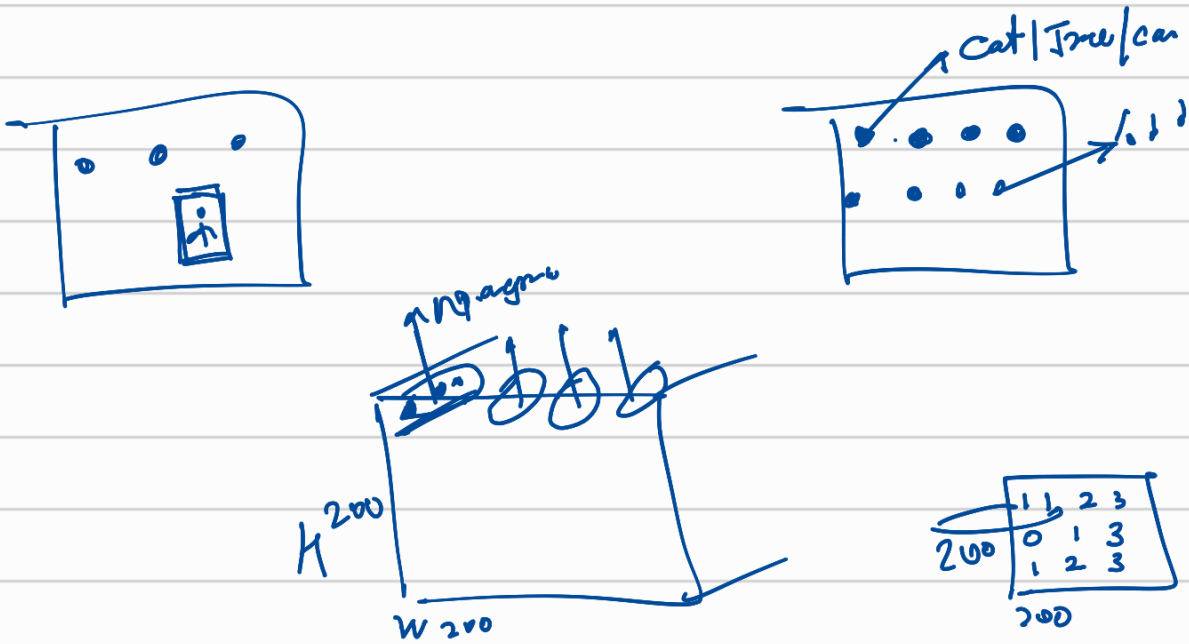


Upsampling2D



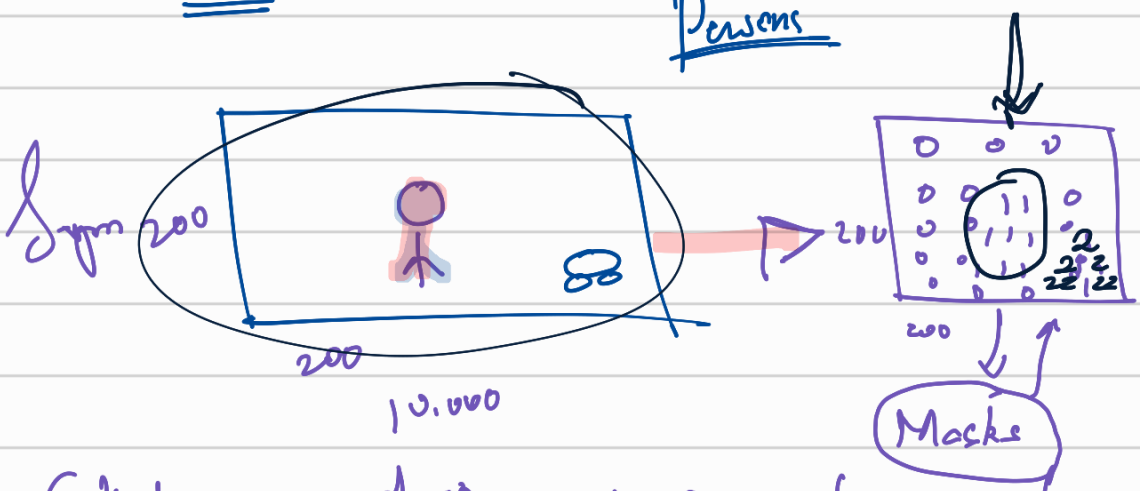
Unet architecture





Task

Persons

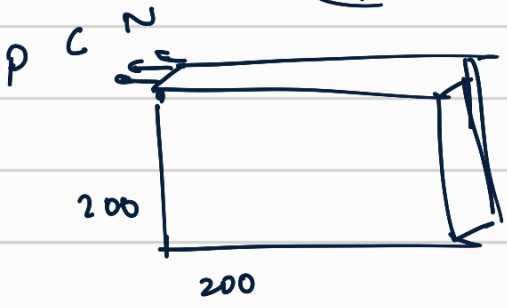


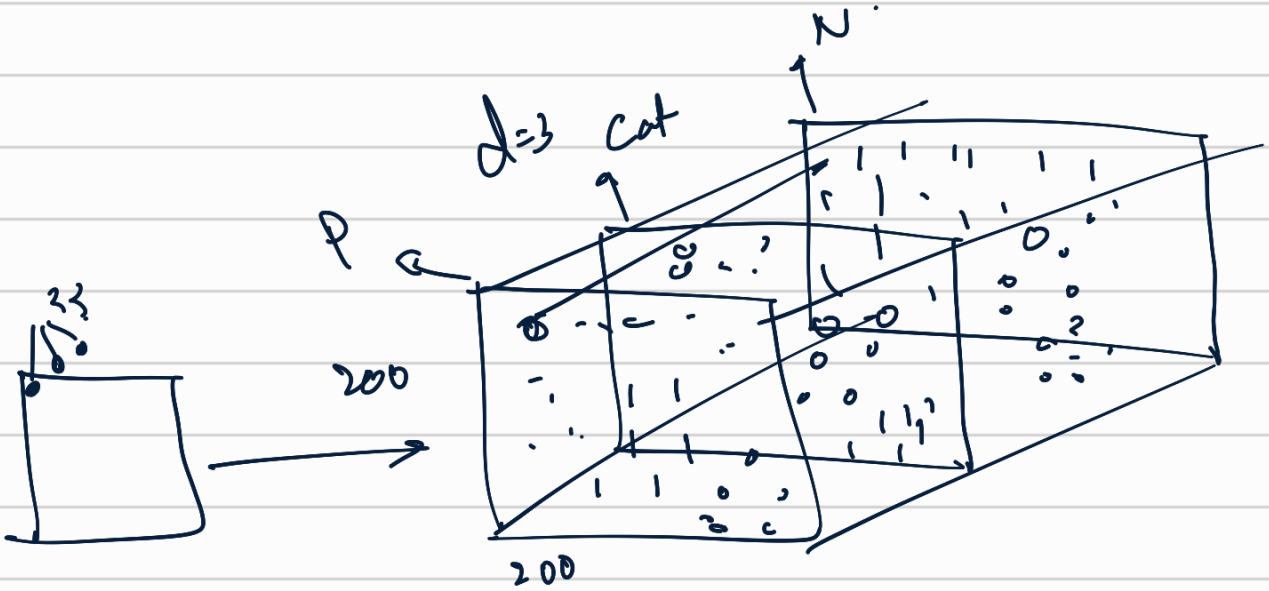
→ Collect images of persons in same background.

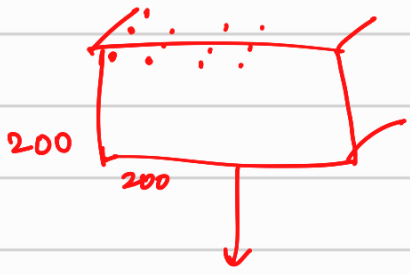
→ Annotation tool to draw the contour of persons

Unet.fit(x_train, masks).

Conv2D(3, activation='softmax').





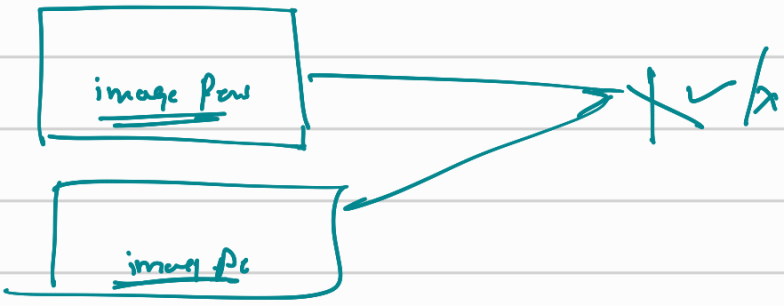


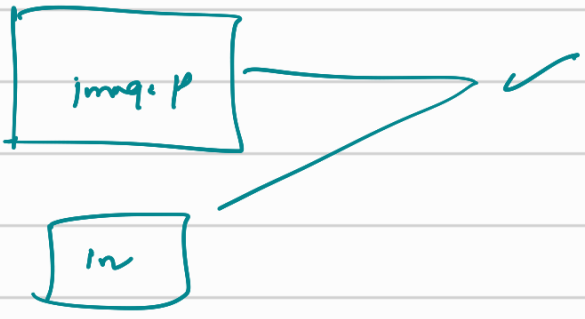
MC Pixel level Cross Entropy

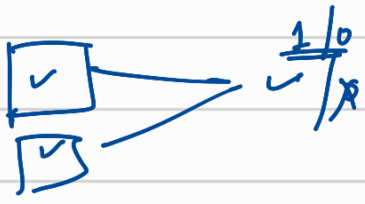
Siamese Networks

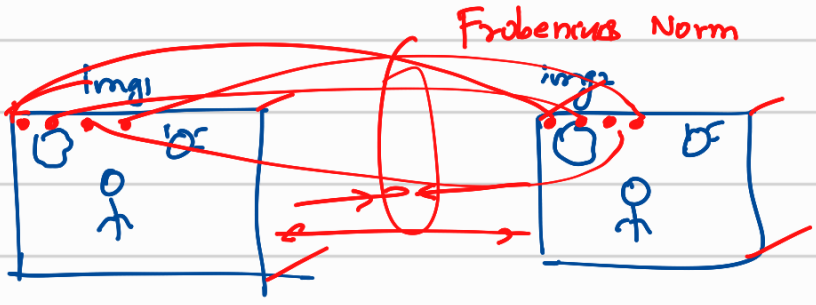
Siamese Network →

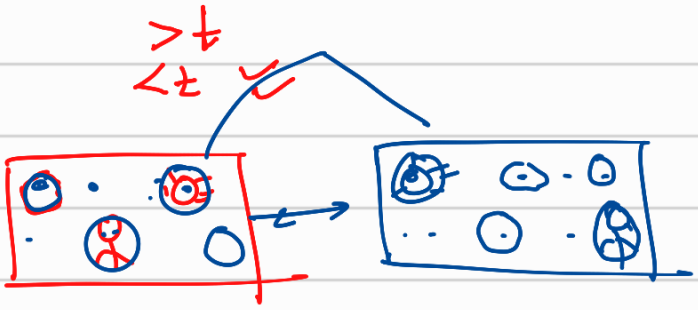
Compare two images

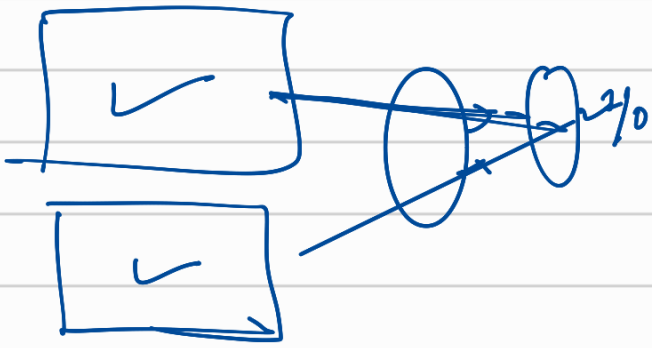


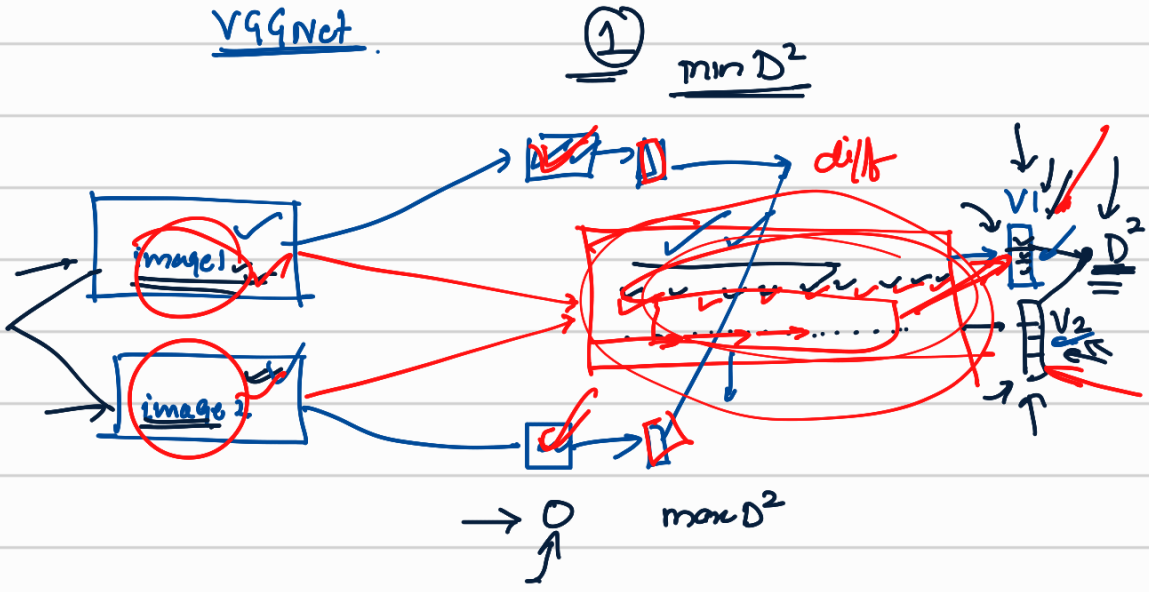


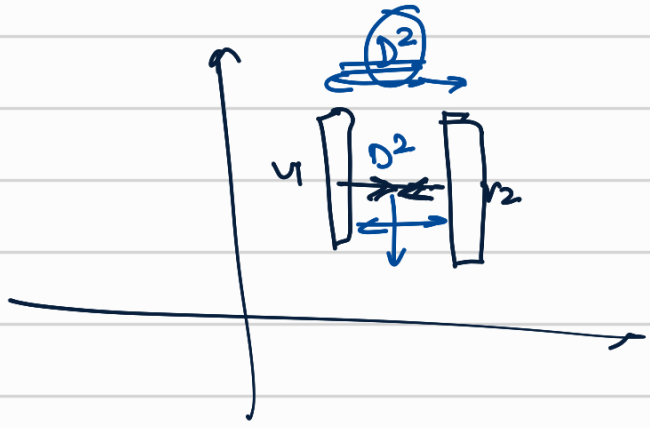


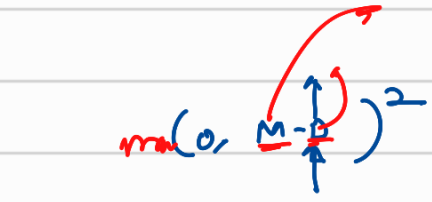


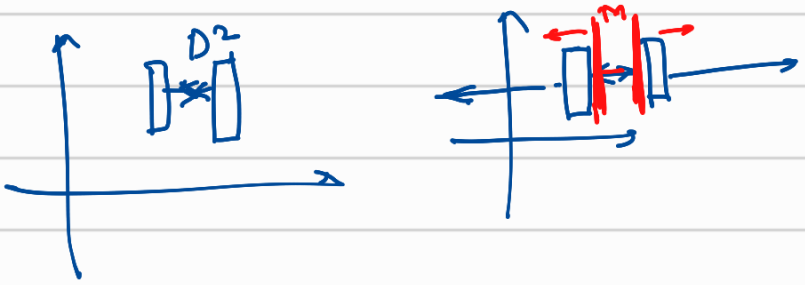



L
L
1/0


VGGNet
①
min D²
image1
image2
diff
V1
V2
D²
max D²

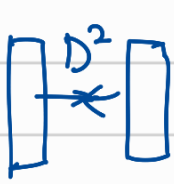

D²
V1
D²
V2

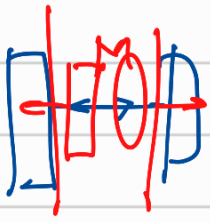

max(0, M - D²)²


D²
M

$$loss = \min\left(y_i D^2 + (1-y_i)\left(\max(0, m-D)\right)^2 \right).$$

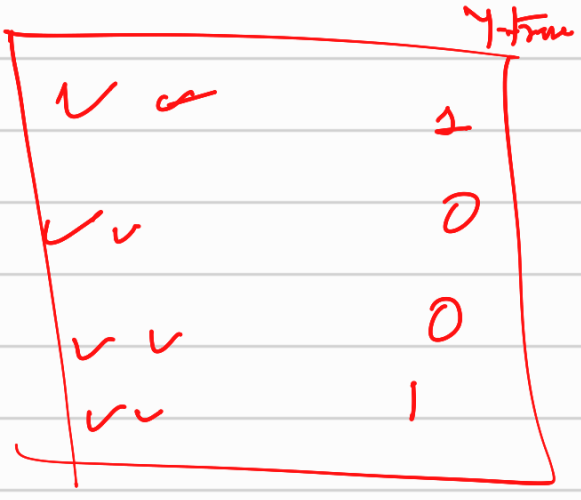
m



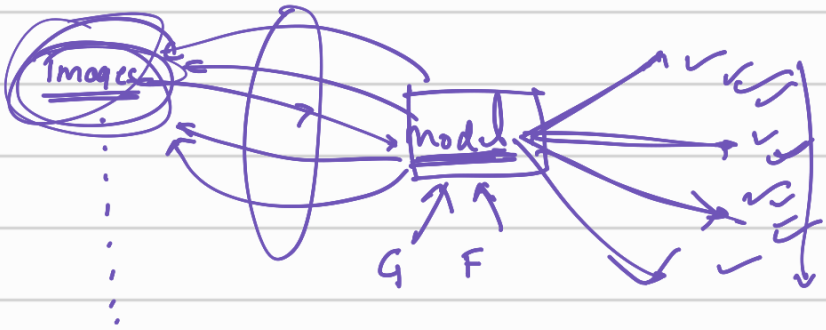


Anchor Loss

Brahs

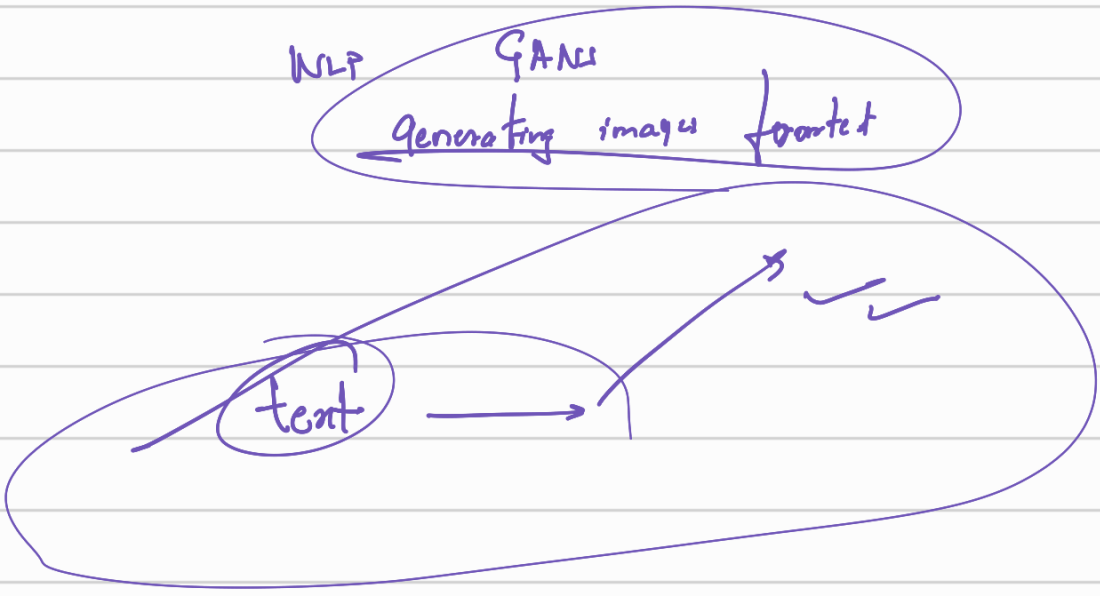


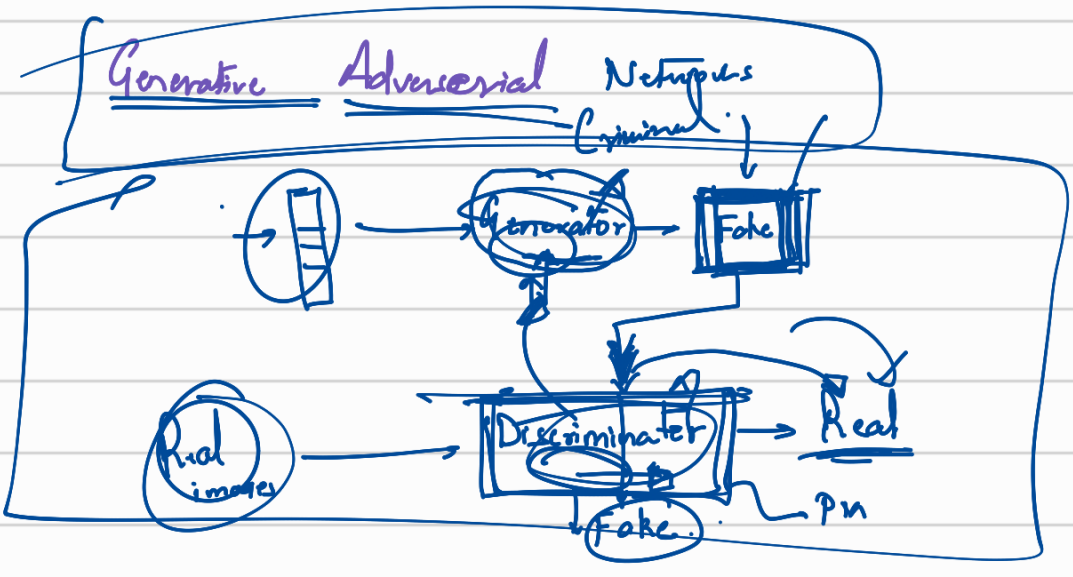
GANs → Generative Adversarial Networks

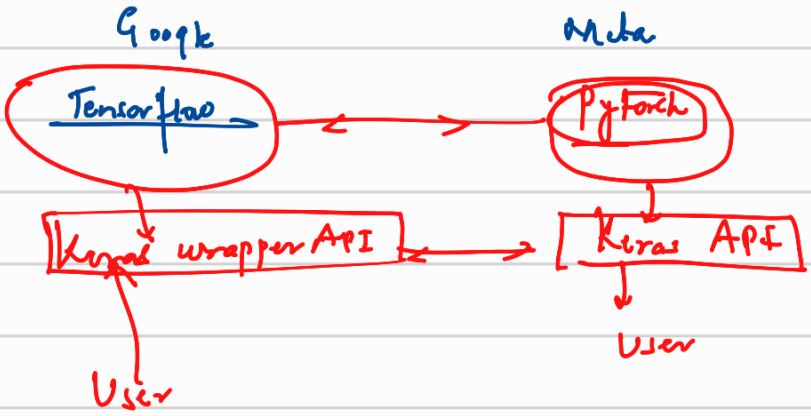


→ Increasing the data size when we have less data

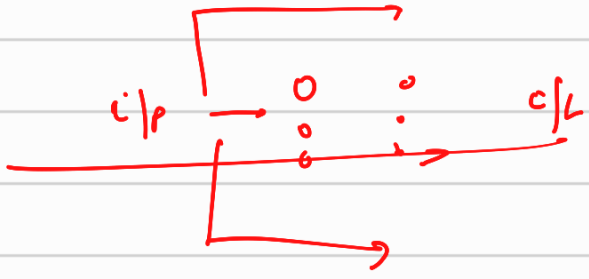
→ Neural Style Transfer






Google
Meta
TensorFlow
PyTorch
Keras wrapper API
Keras API
User
User


i/p
o/p


i/p
o/p