Agenda (ANN):

1/2/3
- Why DL / What is DL, What is ANN.
- What is a Neuron, Why ANN / DL
- What is MLP
- Forward / Backward Propagation.
- Training steps of a ANN.

4/5/6
- implement NN in code. Tensorflow / Keras
- Dropout, Regularization, Batch Normalization, losses, Callbacks. etc., optimizers

7/8/9
- Code implementation, hyperparameter tuning, optimization in ANN.
- Explainability / Autoencoding (Denoising).

What is Deep learning, Why DL, vs ML.

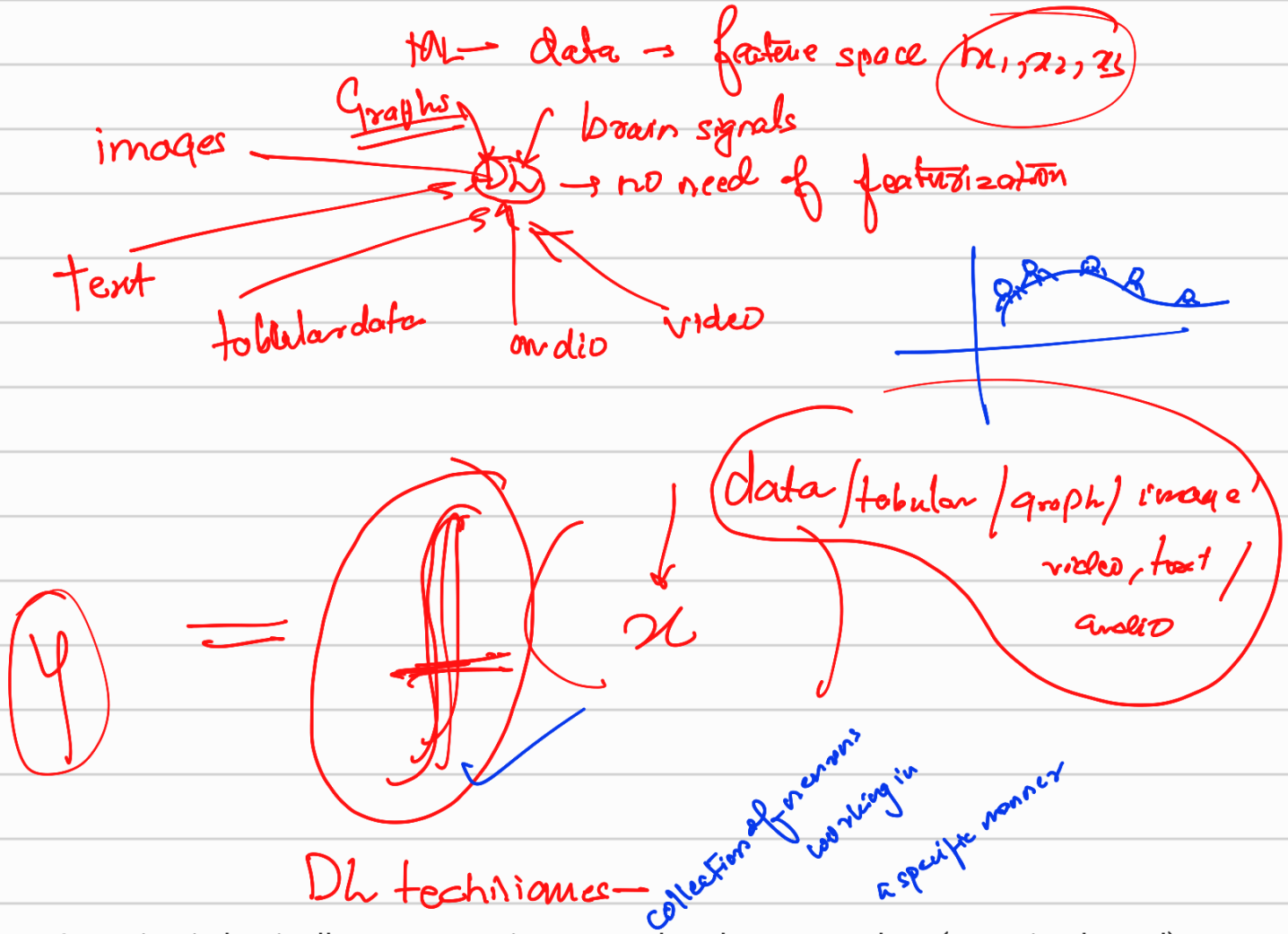
DL

E

DL

Deep learning is basically a field of study where in we use something known as neurons for training the data.

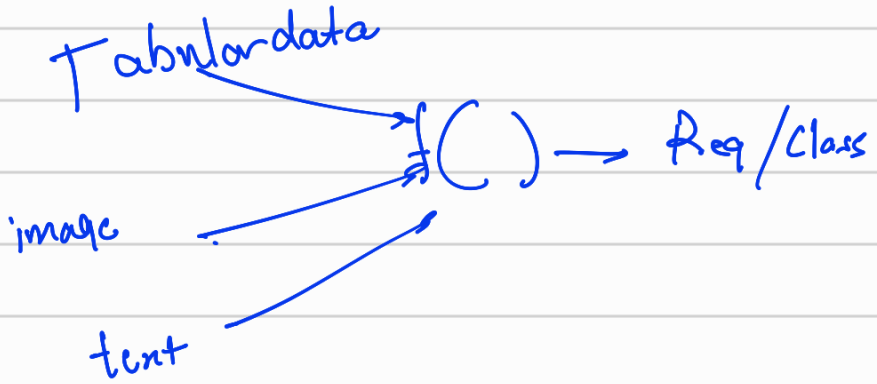
Automated feature engineering.

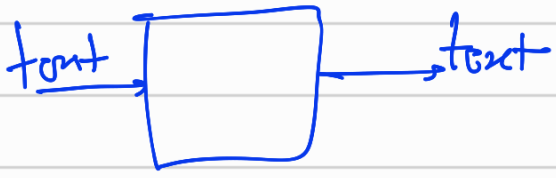


Deep Learning is basically a parametric approach to learn your data (equation based)

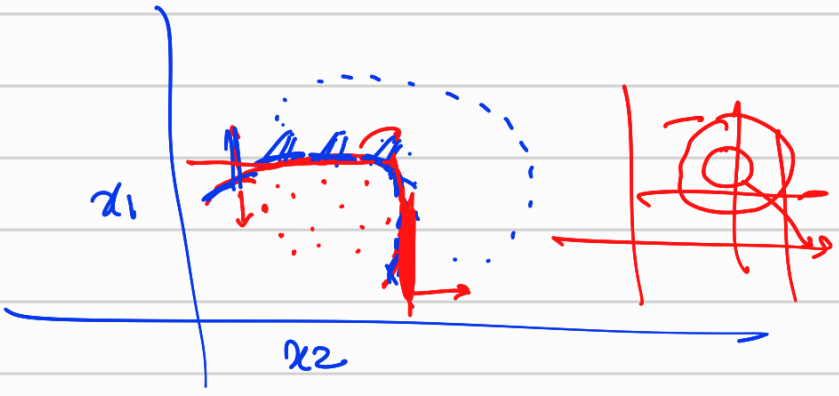
- Artificial Neural Networks- NN is designed to etract information from tabular data
- Convolution Neural Networks- NN is designed to extract information from image data
- Recurrent Neural Networks (LSTM/GRU?Transformers)- NN is designed to extract information from sequence data (like text, audio, videos)
- Graph Neural Network for Graphs (molecule)

Multi Modality:

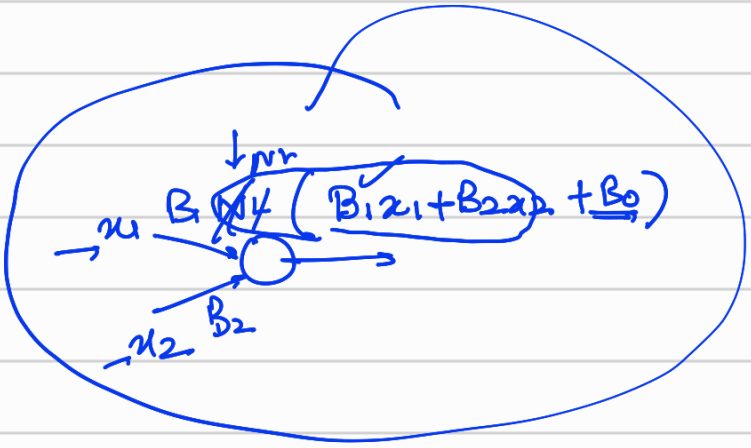


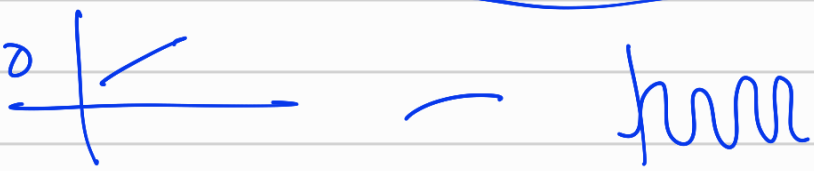


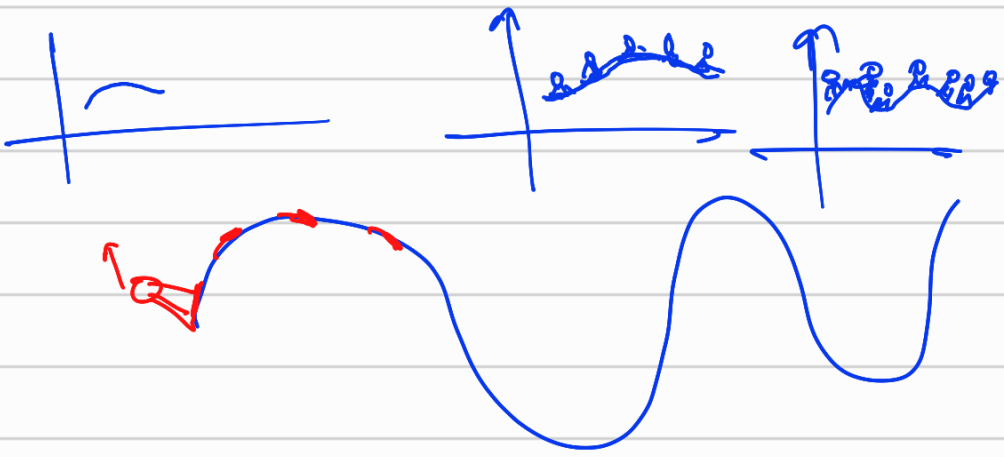
what is Neural network:



Can i fit a decision boundary along the red curve which can be thought of multiple connected curves, and each curve has the independence to move up or down and has the liberty to change o the amount of curvature







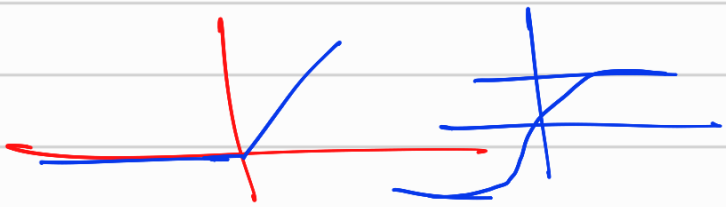
22:55 pm Break

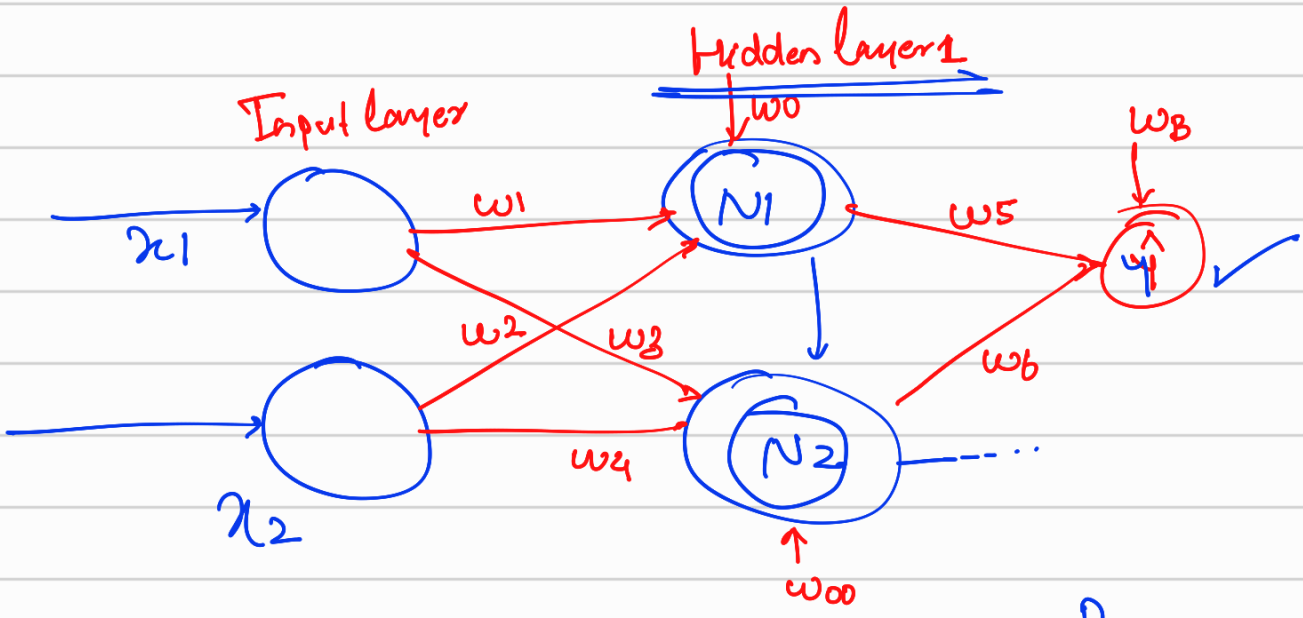
15 - 20 mins Proving why non-linearity is important

① Why the non-linearity is absolutely vital ✓

② Gradient descent - underlying algo for training the NN.

Input layer





$$N_1 = \tanh(w_1 x_1 + w_2 x_2 + w_0)$$ ← Biases

$$N_2 = \tanh(w_3 x_1 + w_4 x_2 + w_{00})$$ ← Biases

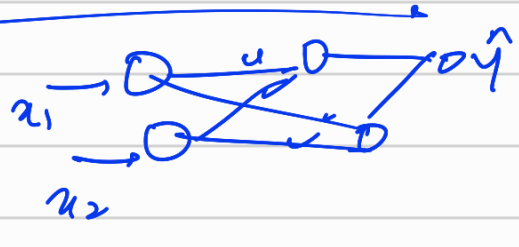
$$\hat{y} = w_5 N_1 + w_6 N_2 + w_B$$

$$\hat{y} = w_5(w_1 x_1 + w_2 x_2 + w_0) + w_6(w_3 x_1 + w_4 x_2 + w_{00}) + w_B$$

$$\hat{y} = w_5 w_1 x_1 + w_5 w_2 x_2 + w_5 w_0 + w_6 w_3 x_1 + w_6 w_4 x_2 + w_6 w_{00} + w_B$$

$$\hat{y} = x_1(w_5 w_1 + w_6 w_3) + x_2(w_5 w_2 + w_6 w_4) + w_5 w_0 + w_6 w_{00} + w_B$$

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



How do we train such a network of neurons.

1. Initialise the architecure (define the number of hidden layers, neurons etc.)
2. Randomly initialise the weights
3. Given the weights calculate the loss..--> Forward Propogation
4. Given the loss, update the weights- Gradient Descent Algorithm +Chain Rule --> BP
5. using the updated weights go to step 3 and keep on doing it till you stabilise your training loss to the min possible (min Val loss)

glorot uniform, glorot normal

activation

Backward Propogation

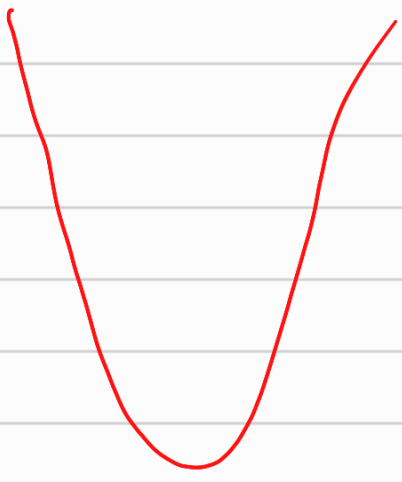
$$loss = \sum_{i=1}^{n} \frac{1}{n} (y_i - \hat{y}_i)^2$$

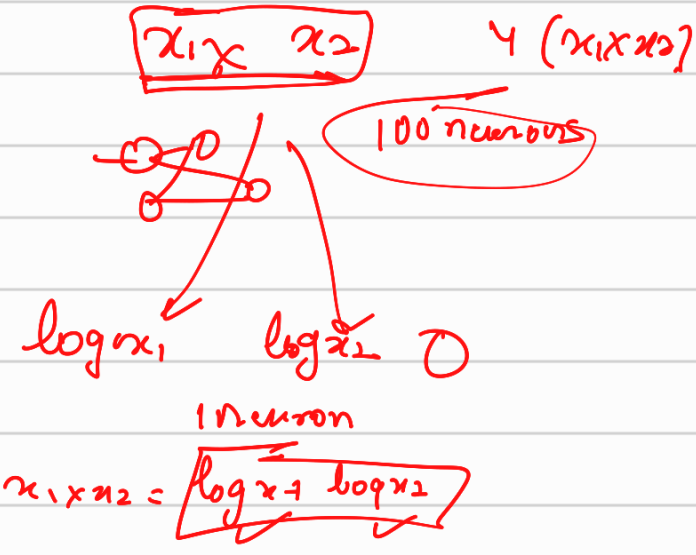
$$loss = f(???)$$

(biases, weights), label, input(x)

minimise the loss

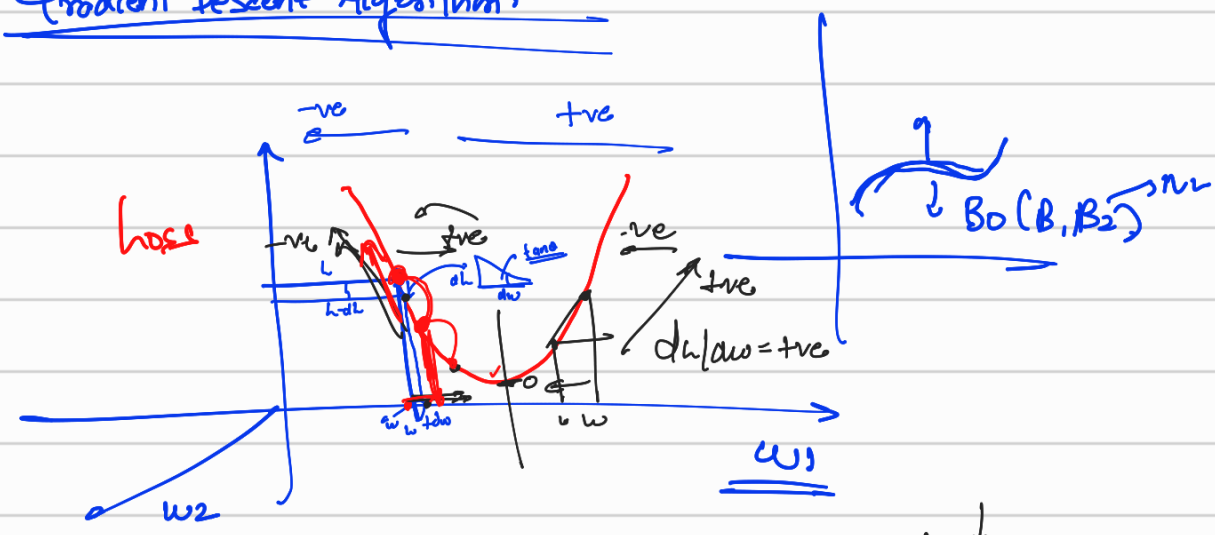
Gradient Descent Algorithm and Chain Rule:





13/12/2022

Gradient Descent Algorithm:



$dL/dw_1 = -ve$

$y = f(x)$

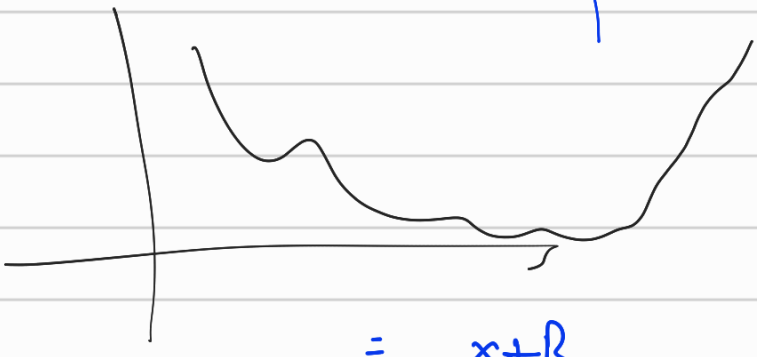
$y = f(x) + \varepsilon$, irreducible error

update rule for my GD for weight w_1

for all w_1 is independently and b_1 is

$$\Rightarrow \boxed{w_1(t) = w_1(t-1) - \alpha\left(\frac{dL}{dw_1(t-1)}\right)}$$

(α: learning rate; $\frac{dL}{dw_1(t-1)}$: Partial derivative)

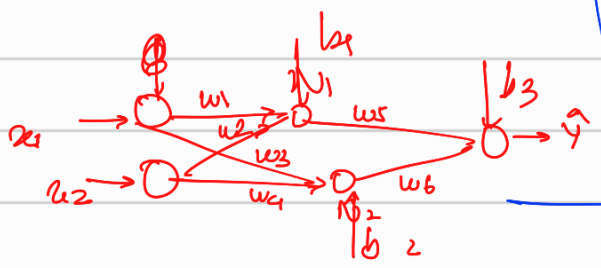


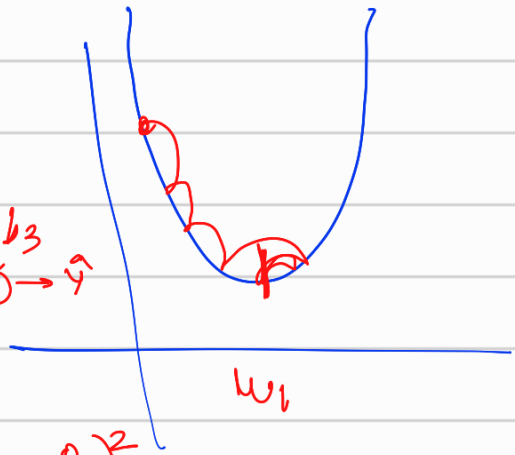
$= x + B$

$\frac{dL}{dw}$

Gradient ascent

$= (w_1 + \alpha\, dL/dw)$





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

$$\frac{\partial L}{\partial w_1} \qquad L = \sum_{i=1}^{n} \frac{1}{n} \left(y_i - (w_5 N_1 + w_6 N_2 + b_3)\right)^2$$

$$\frac{\partial L}{\partial w_2} \qquad dL/dw_5 \text{ , } dL/dw_6.$$

$$dL/dw_3 \ldots dL/dw_6$$

$(w_1 x_1 + w_3 x_2 + b_1)$

$$L = \sum_{i=1}^{n} \frac{1}{n} (y_i - \tanh(w_5 N_1 + w_6 N_2 + b_3))$$

Chain Rule

$y = 5z$ → $y \sim f(z)$ → $dy/dz = 5$

$z = 5x$ → $z \sim f(x)$ → $dz/dx = 5$

$x = 5m$ → $dx/dm = 5 \sim f(m)$

$$\frac{dy}{dx} = \frac{dy}{dz} \times \frac{dz}{dx}$$ → Chain Rule

$y \sim$ implicit $f(x)$

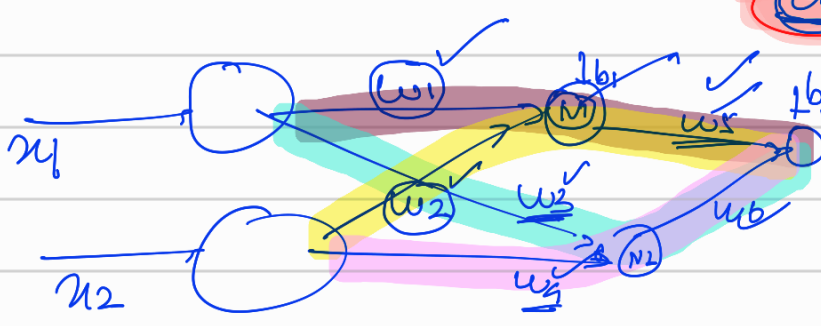
$y = 5 \times 5x$

$y = 25x$

$dy/dx = 25$

Replacement

$$\frac{dL}{dw_1} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dN_1} \times \frac{dN_1}{dw_1}$$



$$\frac{dL}{dw_1} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dN_1} \times \frac{dN_1}{dw_1}$$

$$\sum_{i=1}^{n} \frac{1}{n} (y - \hat{y})^2 = L$$

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

$$\frac{dL}{dw_5} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dw_5}$$

$$\frac{dL}{dw_1} \cdots \frac{dL}{dw_5} \frac{dL}{dw_6}$$

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

$$\frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dw_5}$$

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

$$\frac{dL}{d\hat{y}} = \sum_{i=1}^{n} -\frac{1}{n} 2 (y_i - \hat{y}_i) \times (1)$$

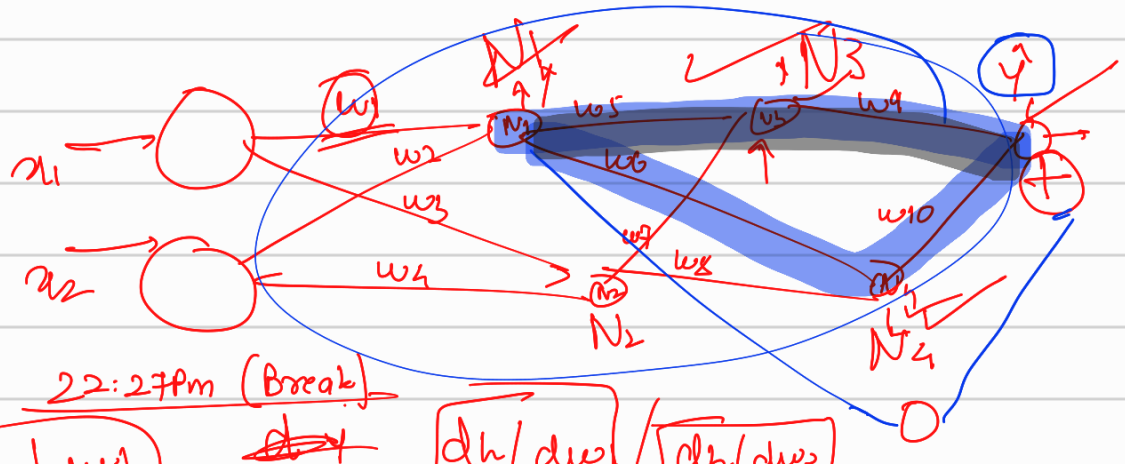
$$\hat{y} = (w_5 N_1 + w_6 N_2 + b_3)$$

$$\frac{d\hat{y}}{dw_5} = N_1 i \times$$

$$dL/dw_5 = \sum_{i=1}^{n} -\frac{1}{n} 2 (y_i - \hat{y}_i) \times N_1 i$$

$$dL/dw_6 = \sum_{i=1}^{n} -\frac{1}{n} \times 2 (y_i - \hat{y}_i) \times N_2 i$$

$$dL/dw_1 =$$



22:27pm (Break)

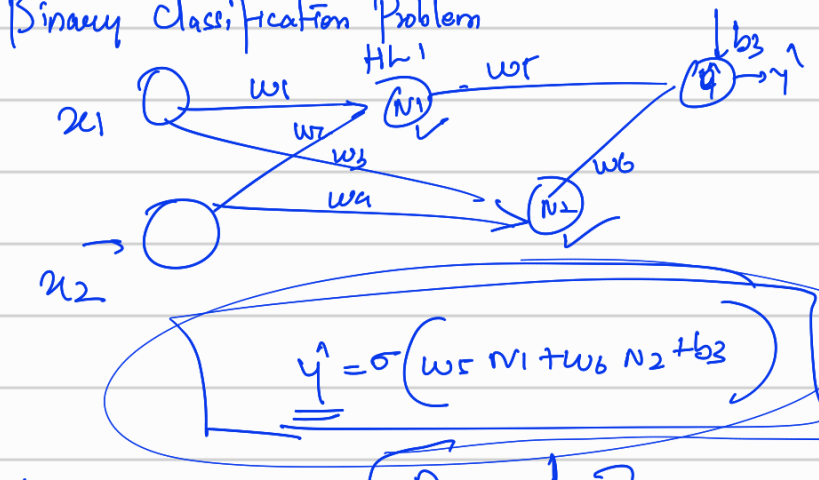
dL/dw_1 ~~dL/dw_1~~ dL/dw_1 / dL/dw_3

$dL/dw_1 =$ $(dL/d\hat{y} \times d\hat{y}/dN_3 \times dN_3/dN_1 \times dN_1/dw_1)$

$$+ \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dN_4} \times \frac{dN_4}{dN_1} \times \frac{dN_1}{dw_1}$$

$$\frac{dL}{dw_1} = \;\swarrow$$

Binary classification Problem



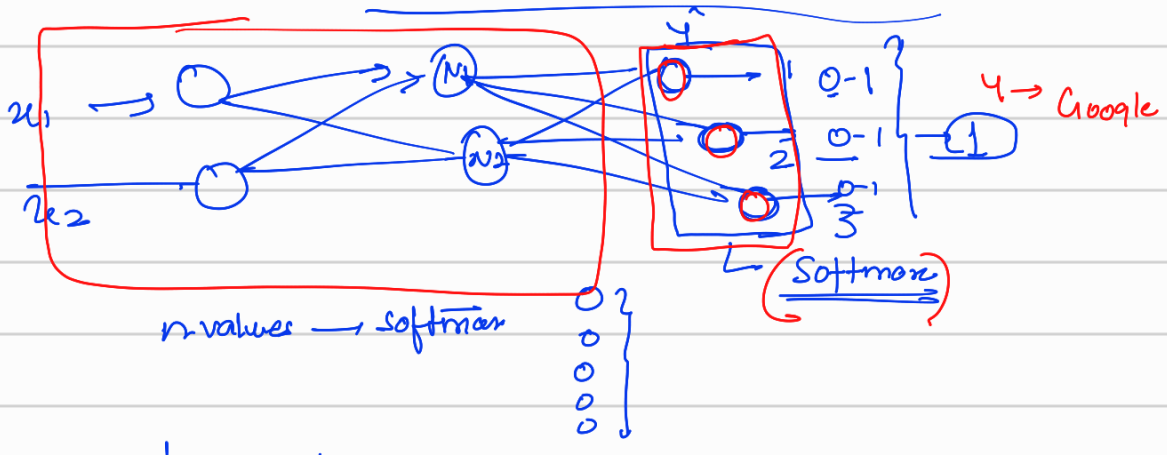
loss = ?

loghoss

$$= \sum_{i=1}^{n} -\left[y \times \log \hat{y} + (1-y) \times \log(1-\hat{y}) \right]$$

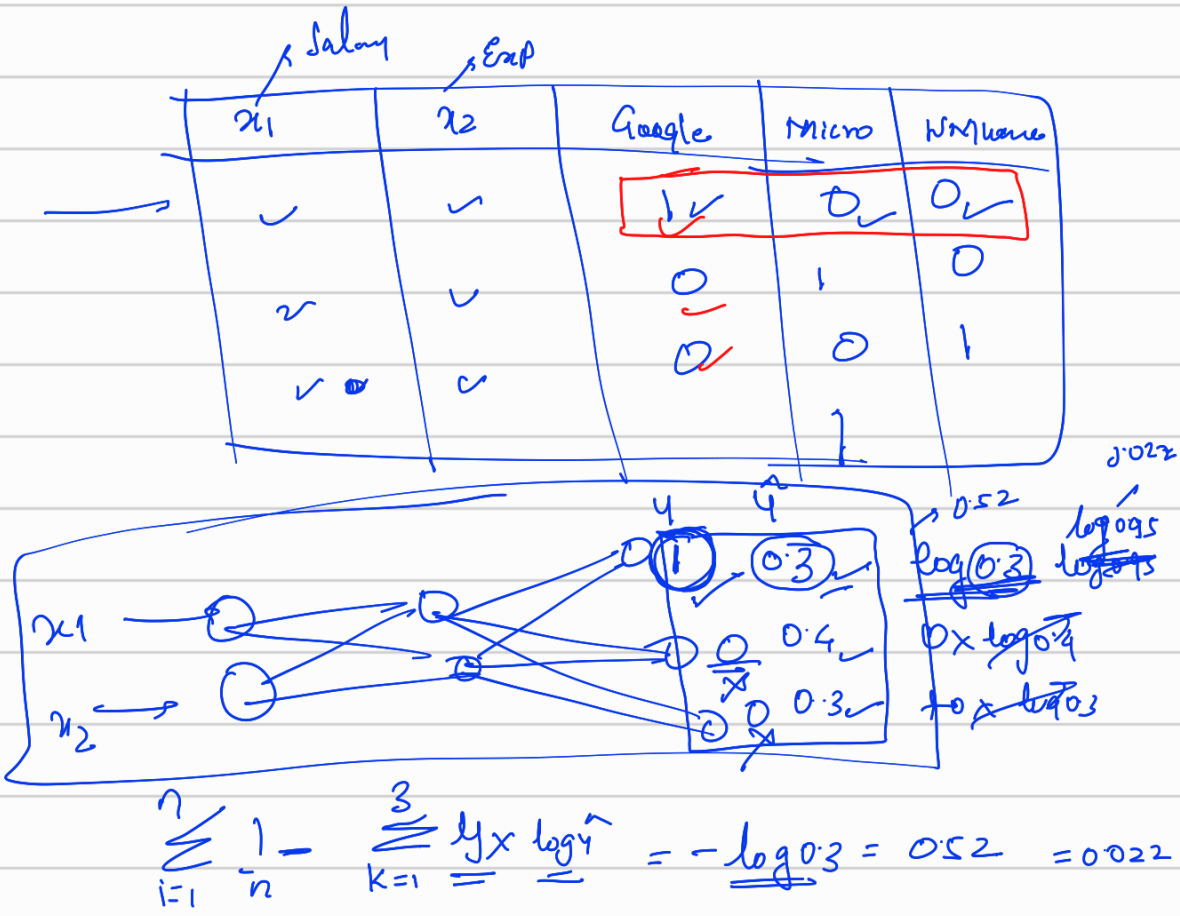
$\log_0 \hat{y}$: $0 \to -\infty$

Classification → Multi Problems (Three classes)

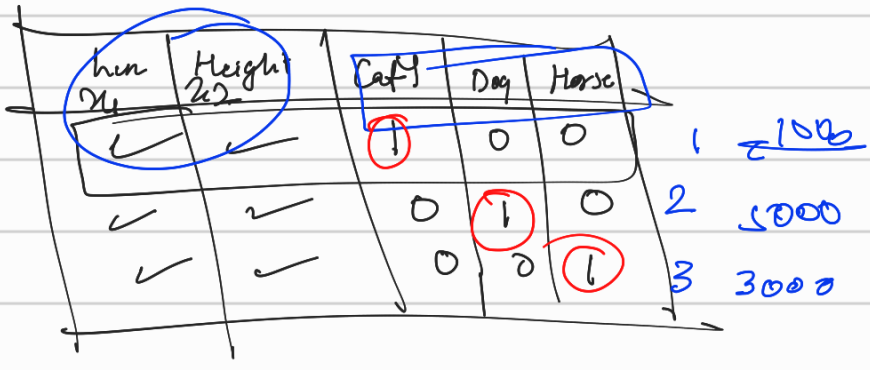


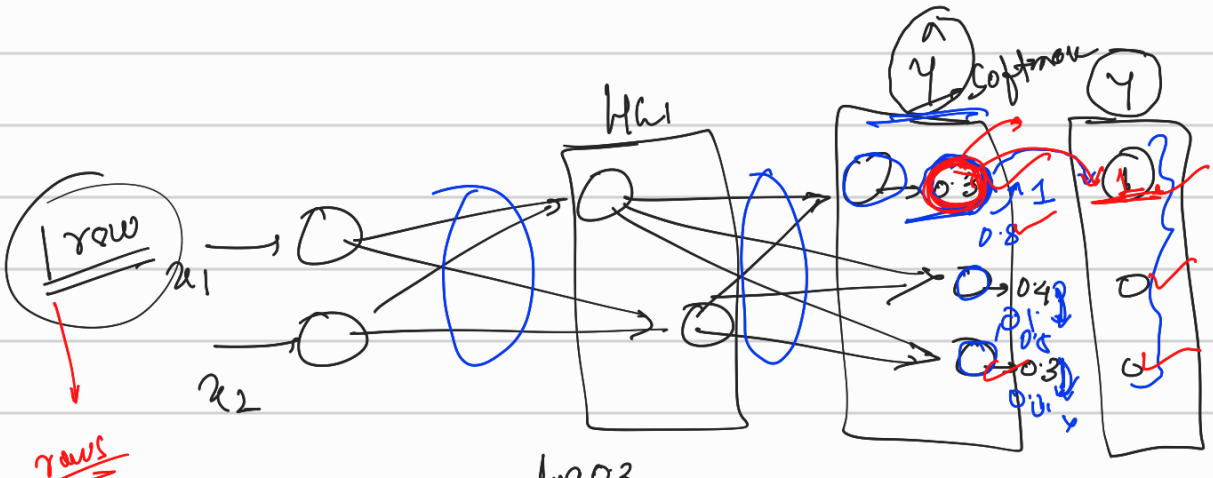
loss = loghoss

$$= \sum_{i=1}^{n} \frac{1}{n} \sum_{j=1}^{K} y_j \log \hat{y}_i$$



$$-\sum_{i=1}^{n} 1 - \sum_{k=1}^{3} y \times \log \hat{y} = -\log 0.3 = 0.52 = 0.022$$





$$-\left[1 \times \log 0.3 + 0 \times \log \hat{y} + 0 \times \log \hat{y} + 0 \times \log \hat{y}\right]$$

Cat Dog Horse

$-\log 0.3 = 0.52$

FHL

Reg

Output layer (→ activation) loss = MSE

BC

Output layer (sigmoid) loss = binary log loss.

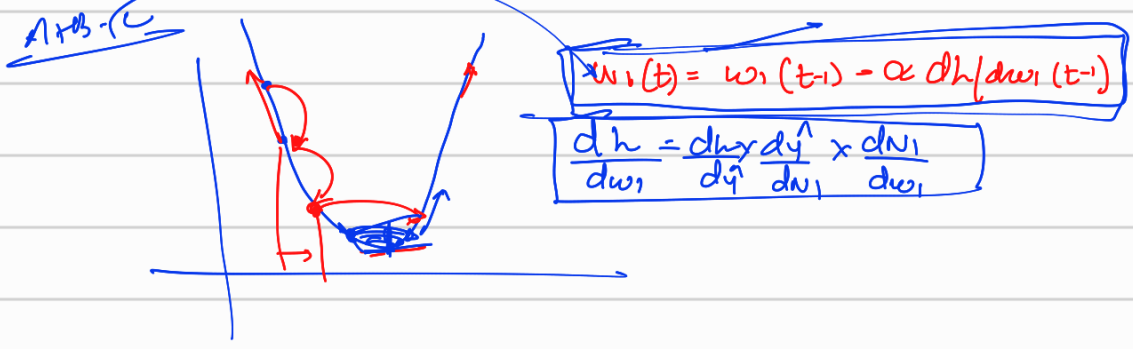
MC output layer (softmax) - loss (categorical loss) $y = [1\ 2\ 0]$

Summary of Previous Class

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

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

$\frac{dL}{dw_1}$



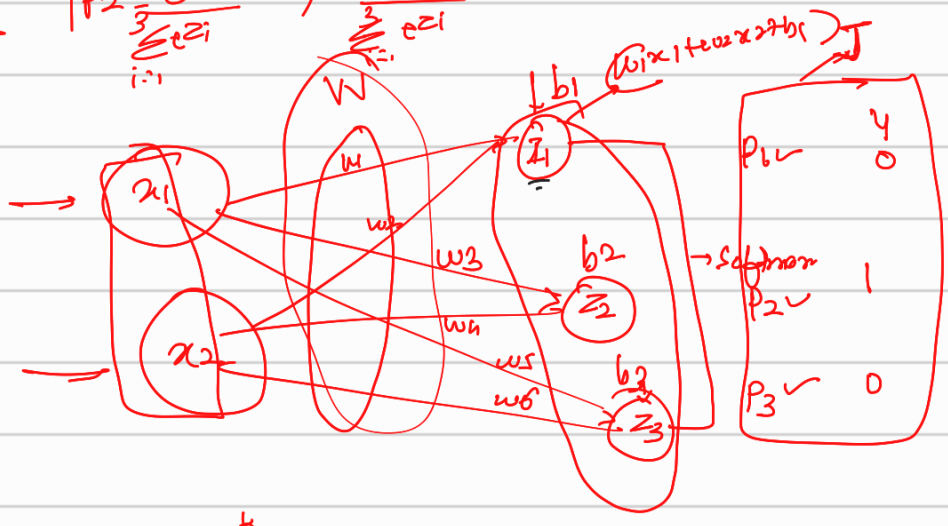
⊙ How do we create a 1 Layer of neural network by basic Python functions.

FP, BP

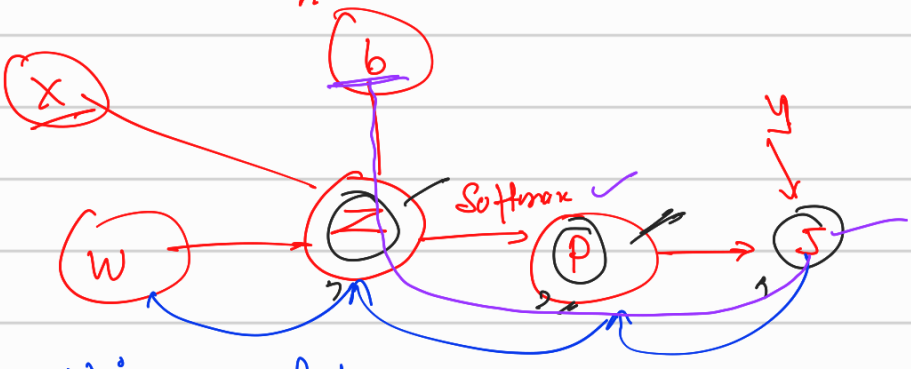
⊙ how do we code our first NN from keras.

⊙ Activation functions

$$P_1 = \frac{e^{z_1}}{\sum_{i=1}^{3} e^{z_i}}, \quad P_2 = \frac{e^{z_2}}{\sum_{i=1}^{3} e^{z_i}}, \quad P_3 = \frac{e^{z_3}}{\sum_{i=1}^{3} e^{z_i}}$$



$$\text{Loss}(J) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{i=1}^{k} y_k \log P_k$$



Using chain Rule

$$\frac{dJ}{dw} = \frac{dJ}{dP} \times \frac{dP}{dz} \times \frac{dz}{dw}$$

$\frac{dJ}{dz}$

Denoting $\frac{dJ}{dw} = dw$

Denoting $\frac{dJ}{dz} = dz$

$$\frac{dJ}{dw} = \frac{dJ}{dz}\cdot\frac{dz}{dw}$$

$\frac{dz}{dw} = x$

$z = wx + b$

$\frac{dz}{dw} = x$

Probs $-$

$$dz = \frac{dJ}{dz} = [P_1, P_2, P_3] - [0\ 1\ 0]$$

$z = wx + b$

$\frac{dz}{dB} = 1$

$$dw = (P - y) \times x$$

$$\frac{dJ}{dB} = \frac{dJ}{dP} \times \frac{dP}{dz} \times \frac{dz}{dB}$$

$$w = w - \alpha\, dw$$

$$\frac{dJ}{dB} = dz \times 1$$

Probs $-$

$$[P_1, P_2, P_3] - [0\ 1\ 0] = 1$$

0
1
2

$=$

$P_1, P_2 = 1, P_3 = 0$

Probs $[\ , y] = \text{Probs}[\ , y) - 1$

dz Probs

dz[range(m), y] = dz[range(m), y] - 1

dz = [0.2 0.3 0.5] - [0 0 1]

y=2 [0 0 1]

y=1 [0 1 0]

y=0 [1 0 0]

Probs[2] = 1

Probs = [0.2 0.3 0.5] → [1 0 0]

[0.2-1 0.3 0.5]

= -0.8 0.3 0.5

[0.7 0.2 0.1] → [0 1 0]

[0.7 0.2-1 0.1]

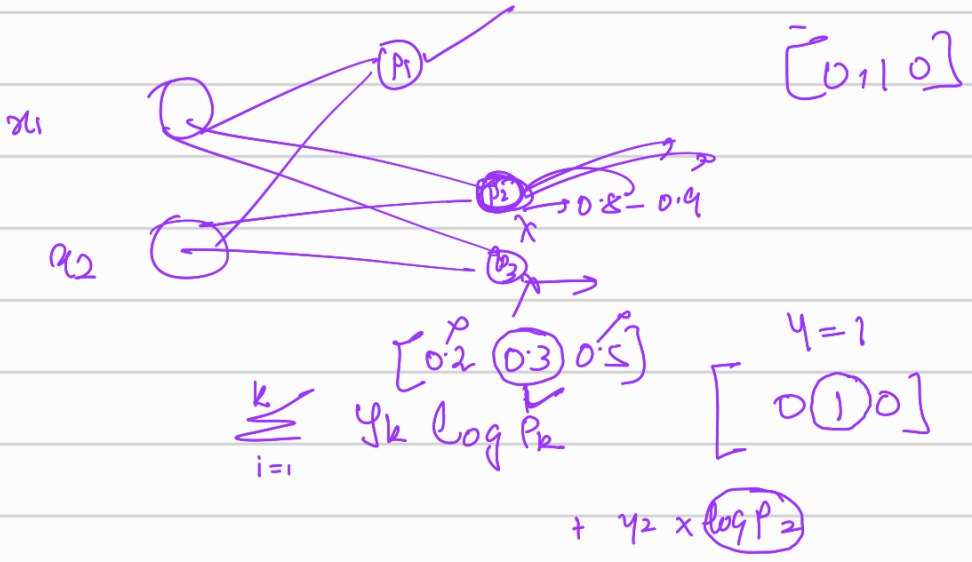
dz = [0.2 0.3 0.5] - [0 1 0]

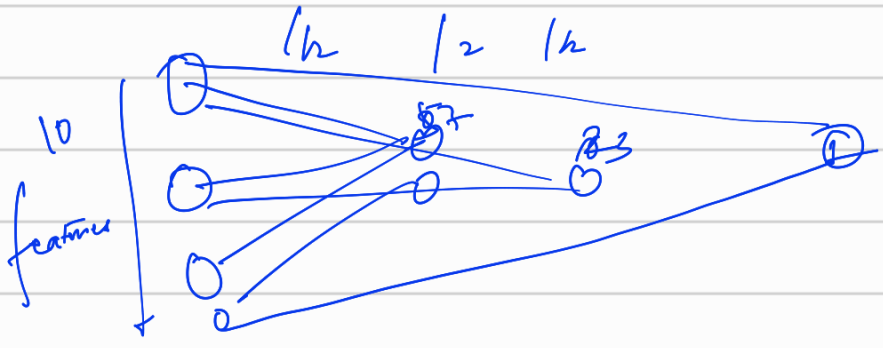
-1

22:22 → 22:32 pm

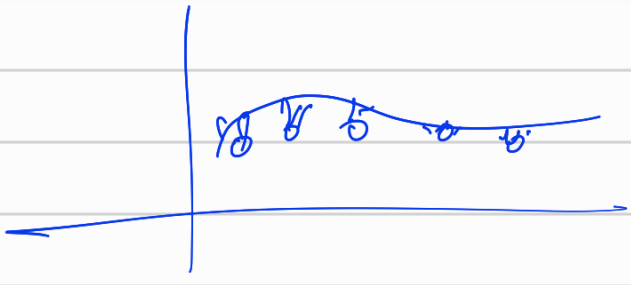
BC → 0/1

MC → Cat, Dog, Horse

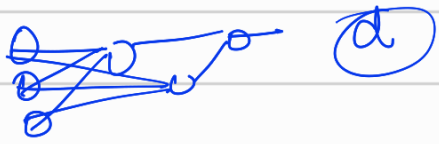




autoencoders



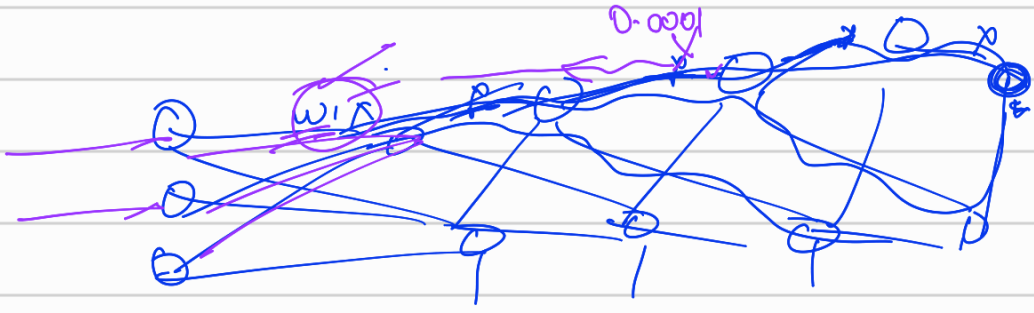
Activation function: what would be a choice of a good activation function.

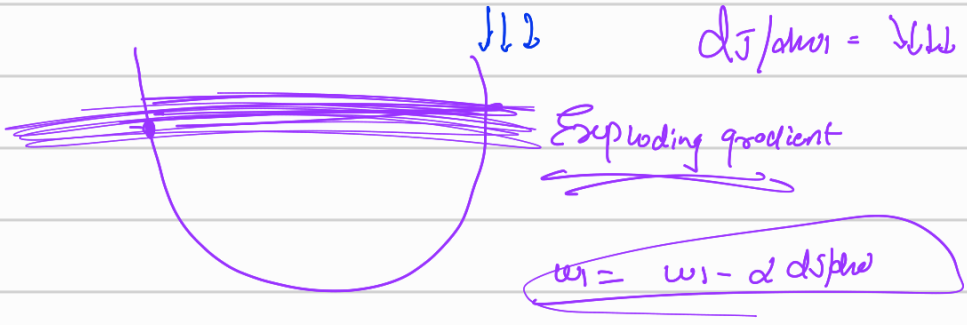


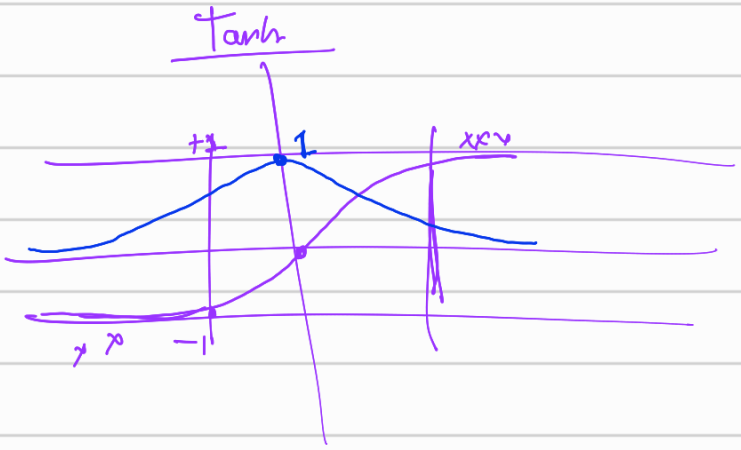
Or $\tilde{o}$

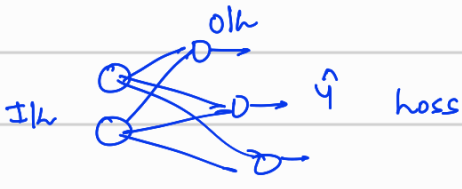
Chain Rule x x x x

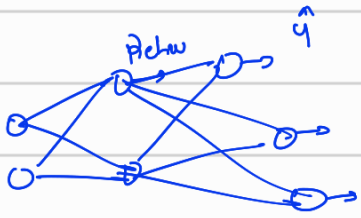
Tanh

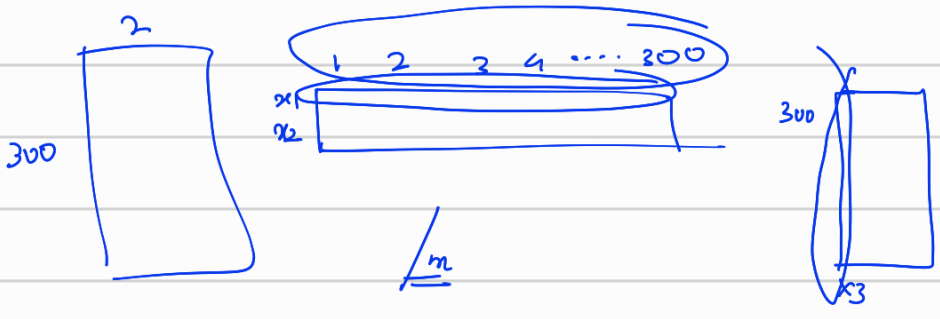


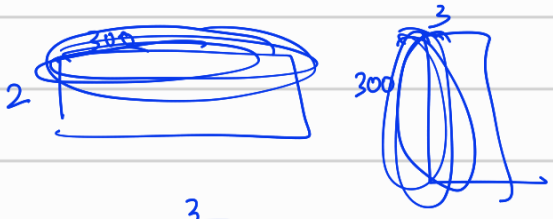



Tanh
+1
1
x<y
x
-1

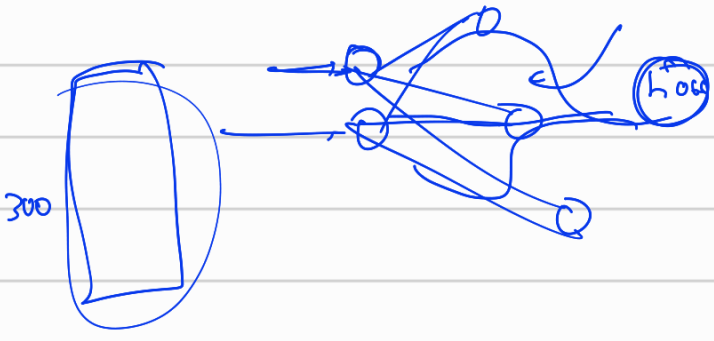

Olh
Ilh
ŷ
loss gradients, bp.

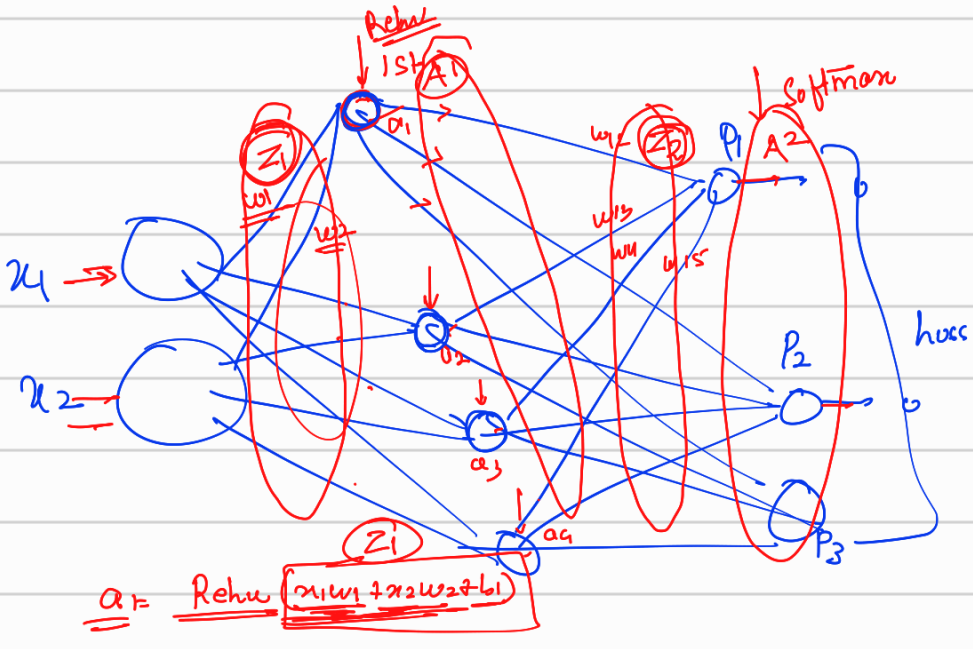

Olh
ŷ
Loss. gradients, bp.


2
300
1 2 3 4 300
x1
x2
m
300
x3



$$= 2 \begin{bmatrix} \phi_m & \phi_m & \phi_m \\ \phi_m & \phi_m & \phi_m \end{bmatrix} / m$$

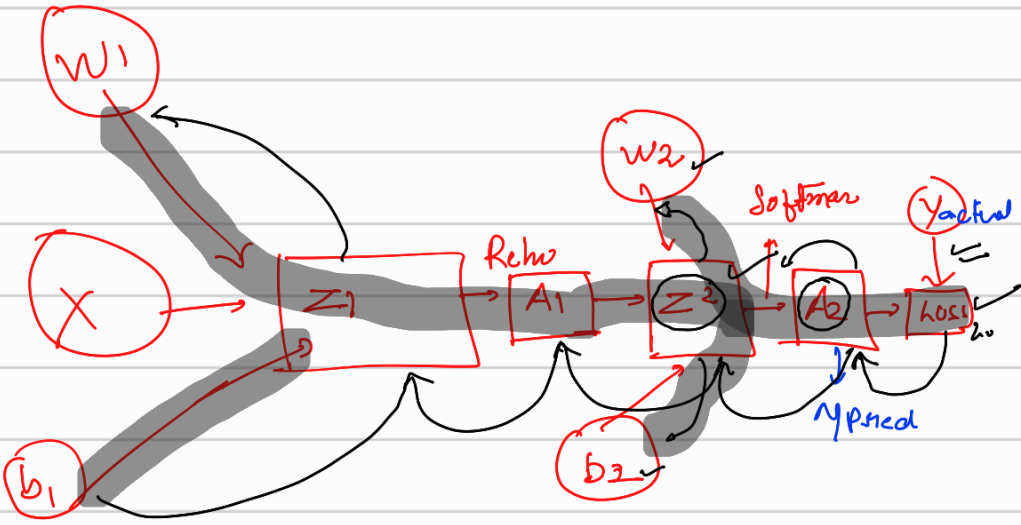




$a_1 = \text{Relu}(x_1 w_1 + x_2 w_2 + b_1)$

$a_i = \text{Relu}(z_i)$

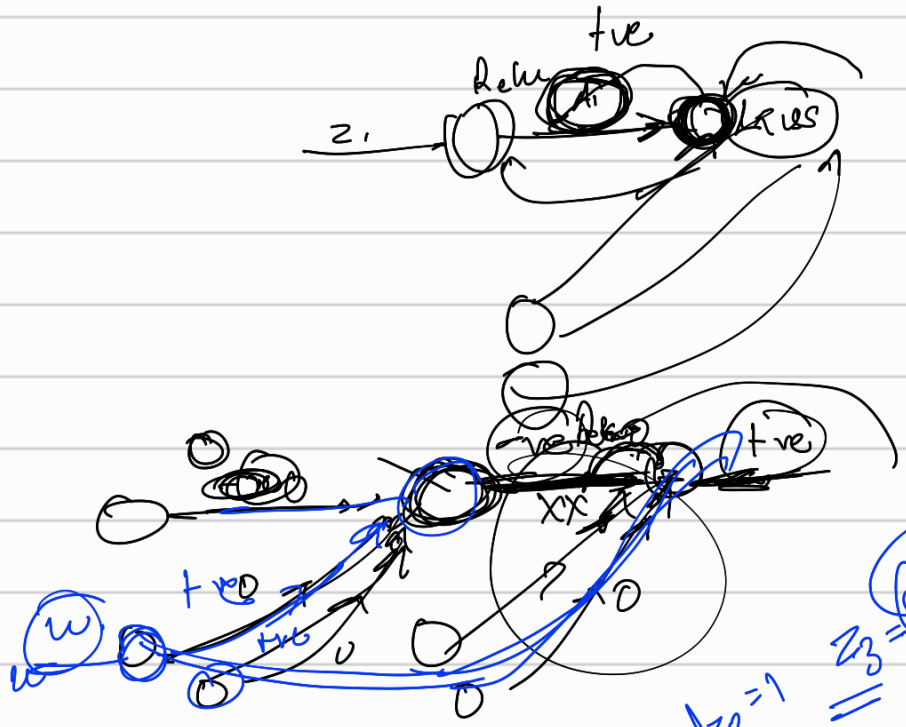
$z_2 = (w_{13} x_1 + w_{23} x_2 + w_{33} a_3 + w_{43} a_4 + b)$



$$loss = L$$

$$\frac{dL}{dA_2}$$

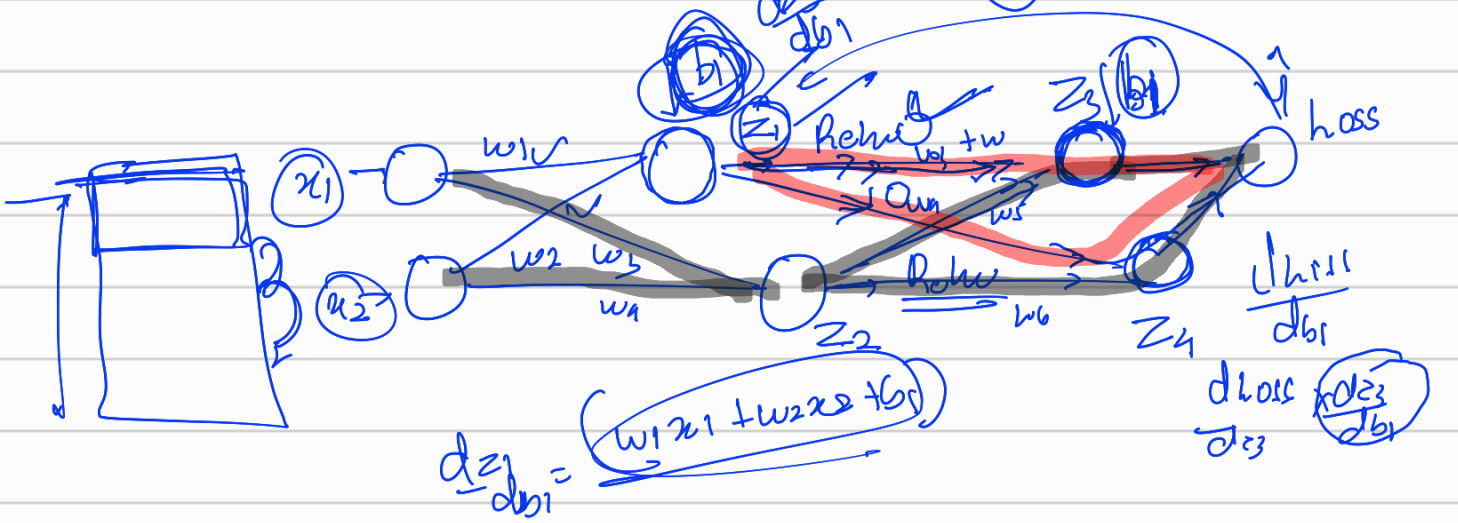
$$\frac{dL}{dz_2} = dz_2 = (P - y)$$



$$z_3 = (w_3 z_1 + w_5 x z_2) + b_1$$

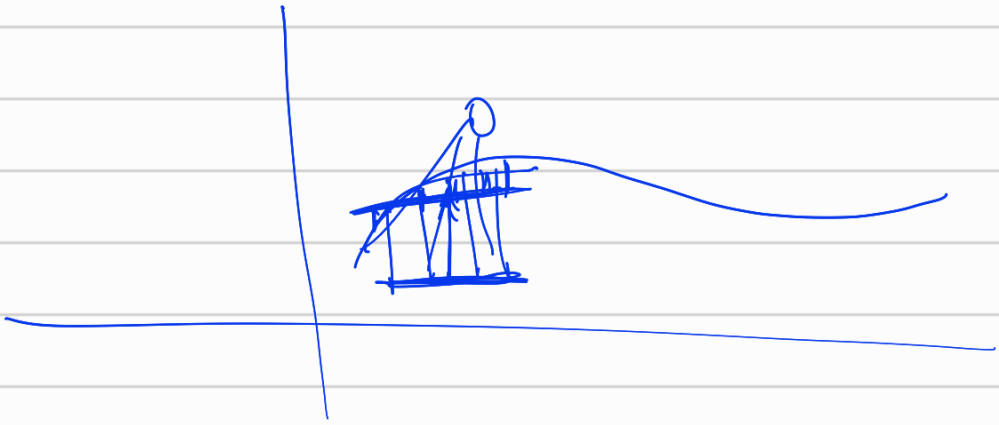
$$\frac{dloss}{dz_3} \times \frac{dz_3}{dw_3}$$

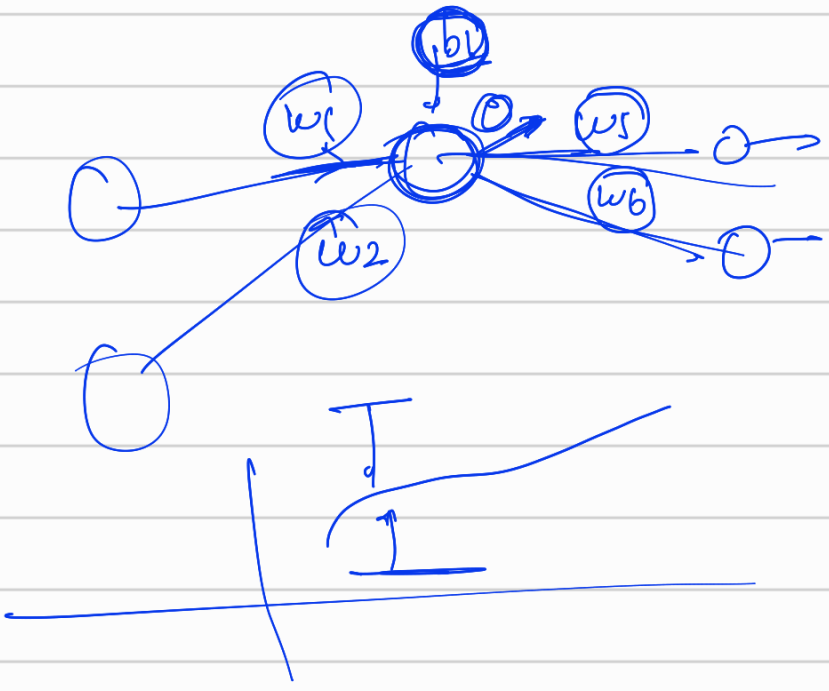
$$\frac{dz_3}{db_1} = 1$$

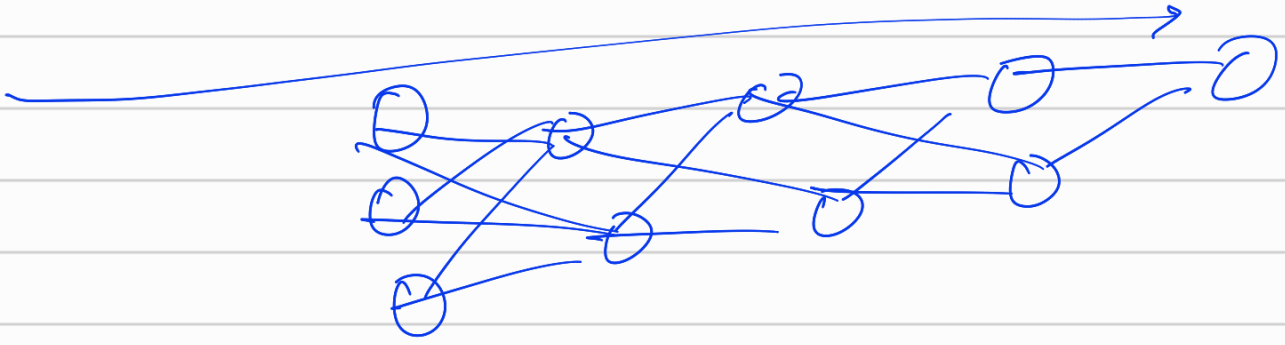


$$\frac{dz_1}{db_1} = (w_1 x_1 + w_2 x_2 + b_1)$$

$$\frac{dloss}{dz_3} \times \frac{dz_3}{db_1}$$

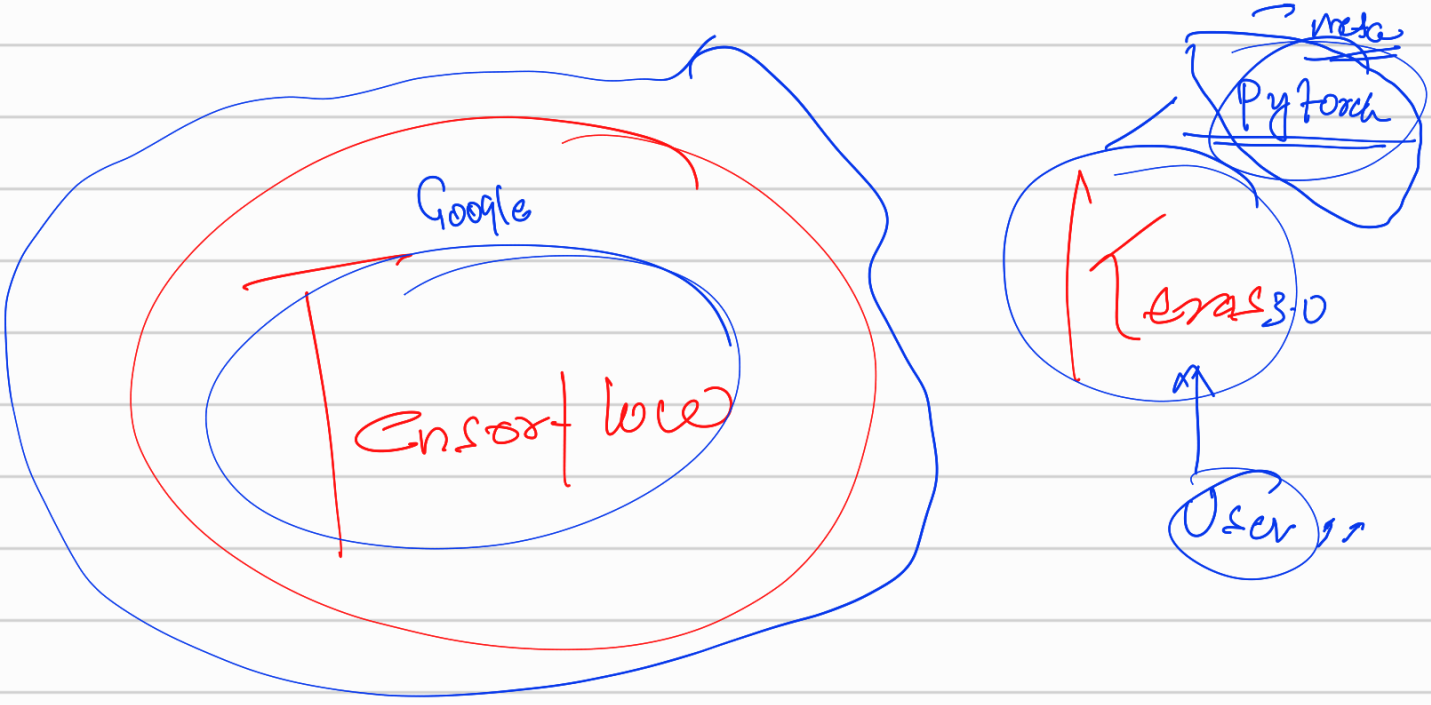






① Tensorflow 2 Keras

② Dropout, Batch normalization, Regularization, optimizers, Callbacks
Hyperparameter Tuning.



Class : { 0 : 5 , 1 : 70 }

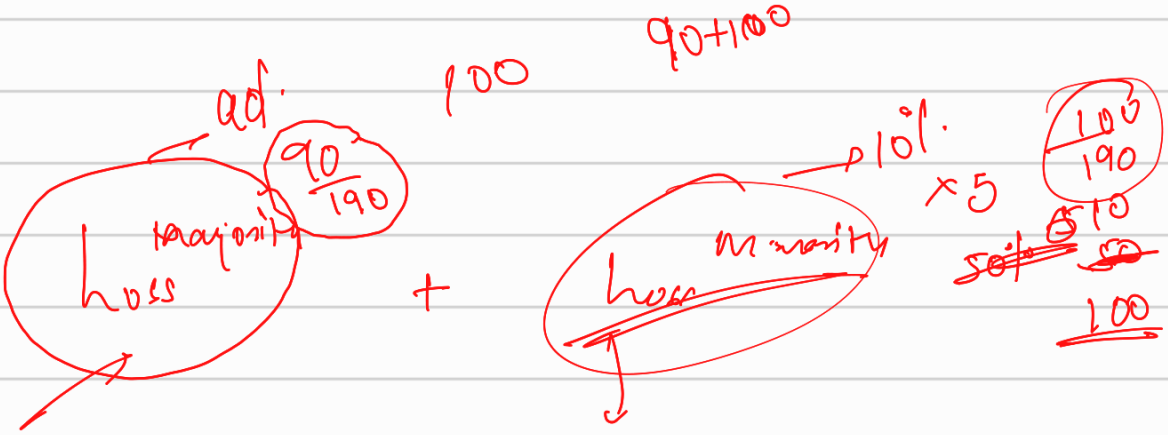
$$\sum_{i=1}^{n} \frac{1}{n} \left(y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i) \right) \times 5$$

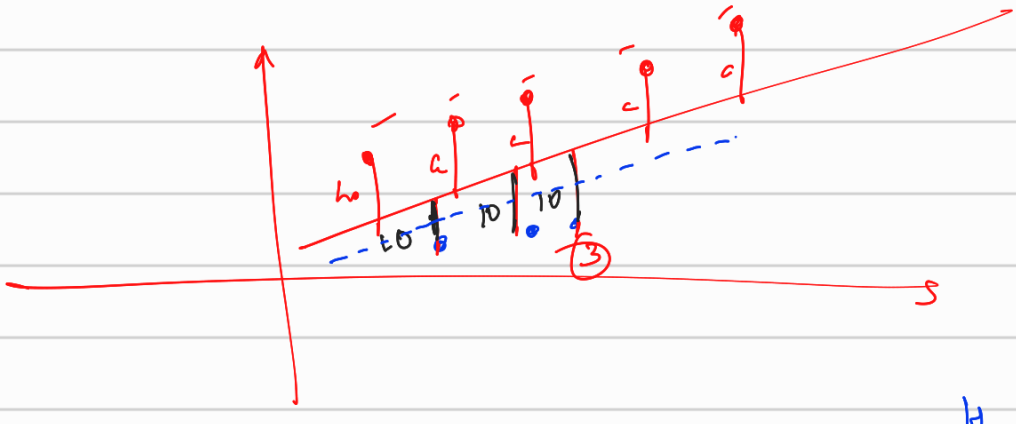
$$-\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{p} y_{ik} \log \hat{y}_{ik}$$

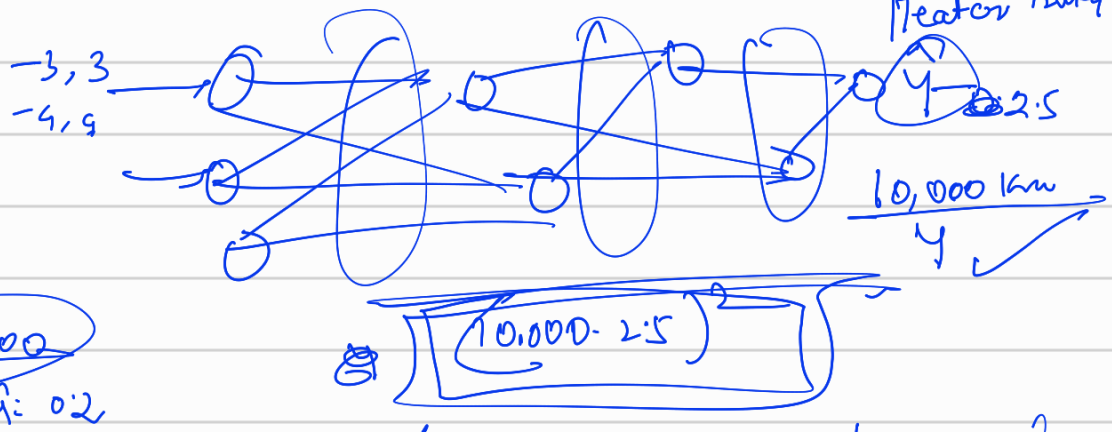
y_{i1} log $\hat{y}_{i1}$ + y_{i2} log $\hat{y}_{i2}$

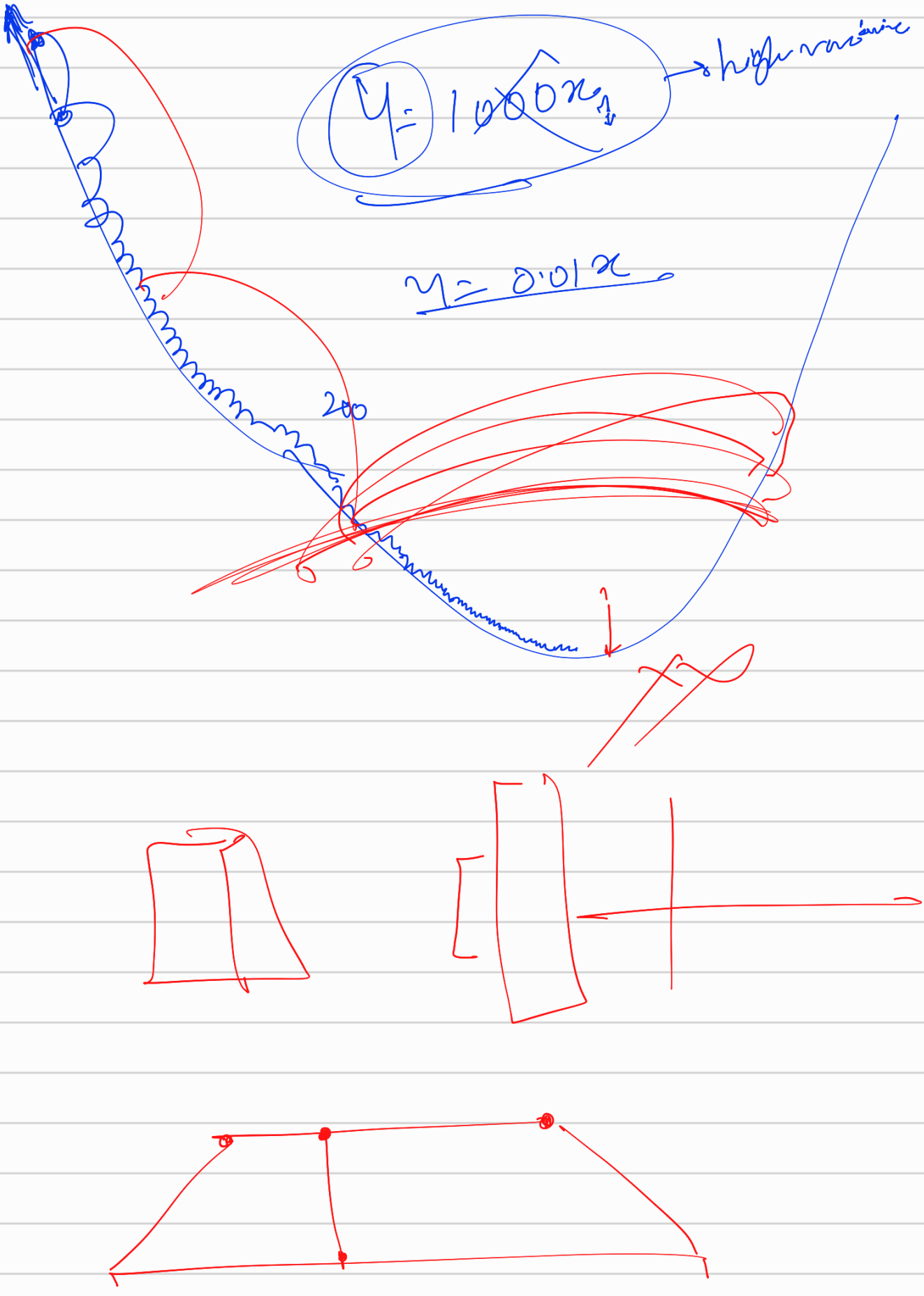
Break -

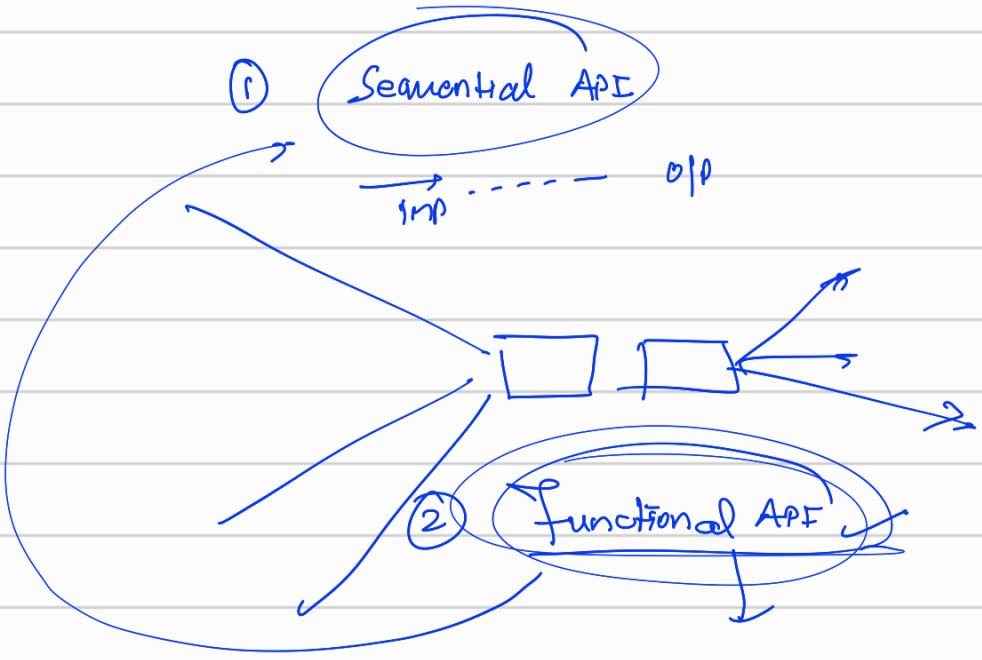
22:27 PM

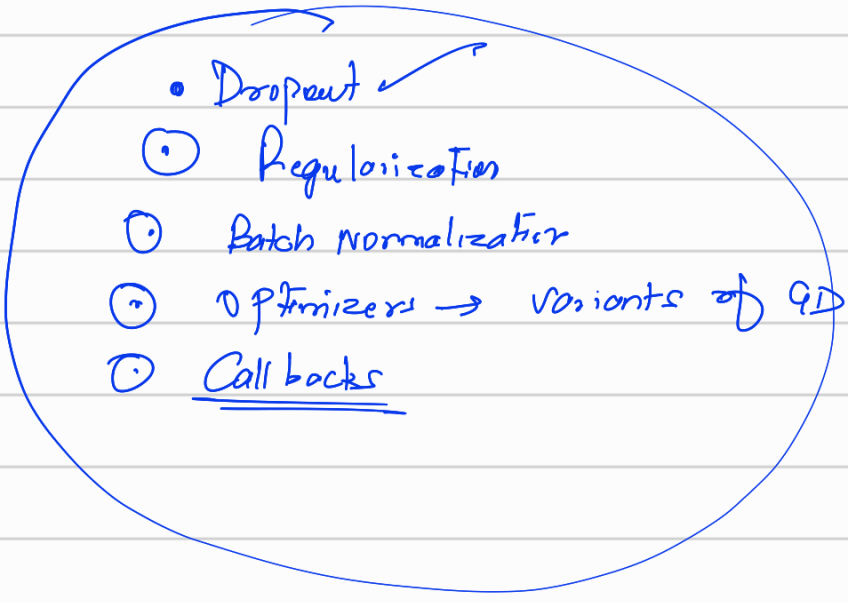







y = 1000x
y = 0.01x
200

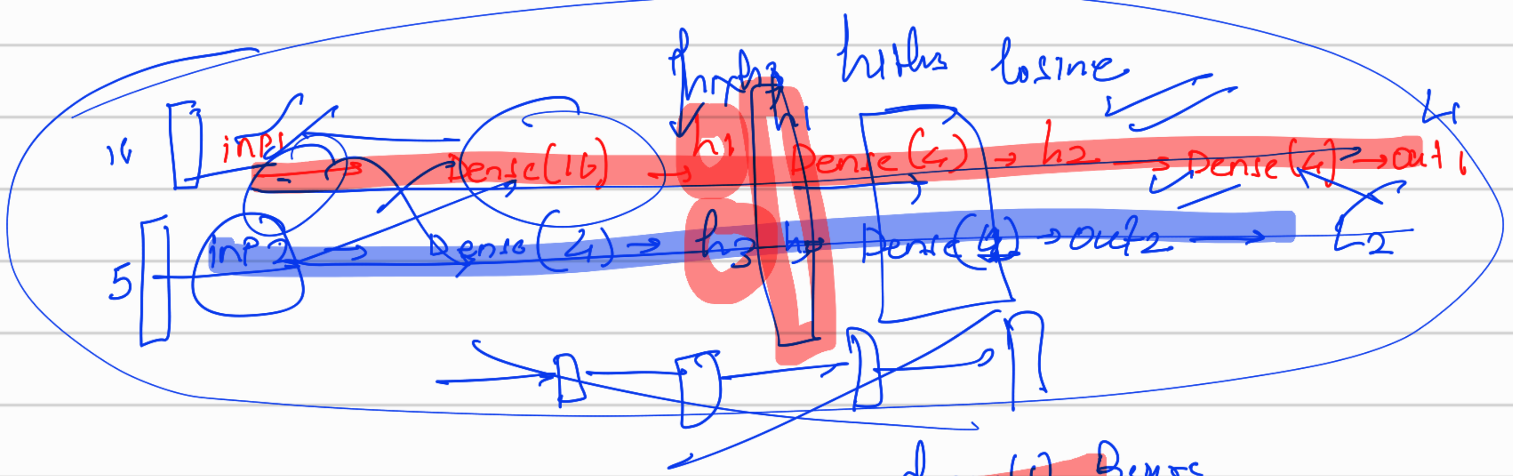


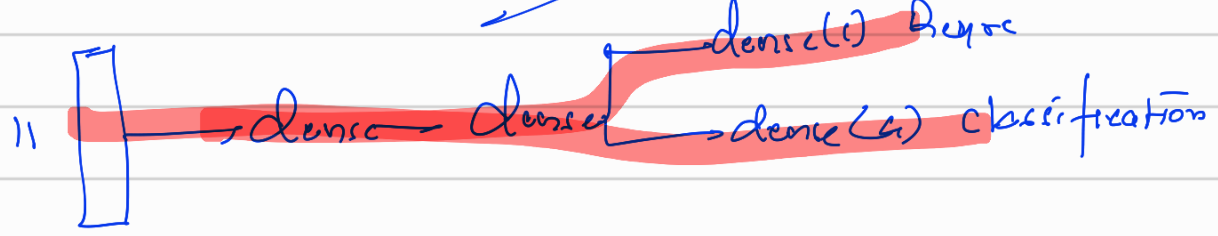


Functional API

```
inp1 = Input ( Shape=(11,) )
inp2 = Input ( Shape = (5,) )

h1 = Dense (16, activation = 'relu', name=' ' ) (inp1)
h2 = Dense (4, a..... , name... ) (h1)
h3 = Dense (4 .... ) (inp2)
out1 = Dense (4, activation='Softmax') (h2)
out2 = Dense (1,) (h3)
```


16
inp1
Dense(16)
h1
Dense(4)
h2
Dense(4)
out1
5
inp2
Dense(4)
h3
Dense(4)
out2
L2


11
dense
dense
dense(1)
dense(4)
classification

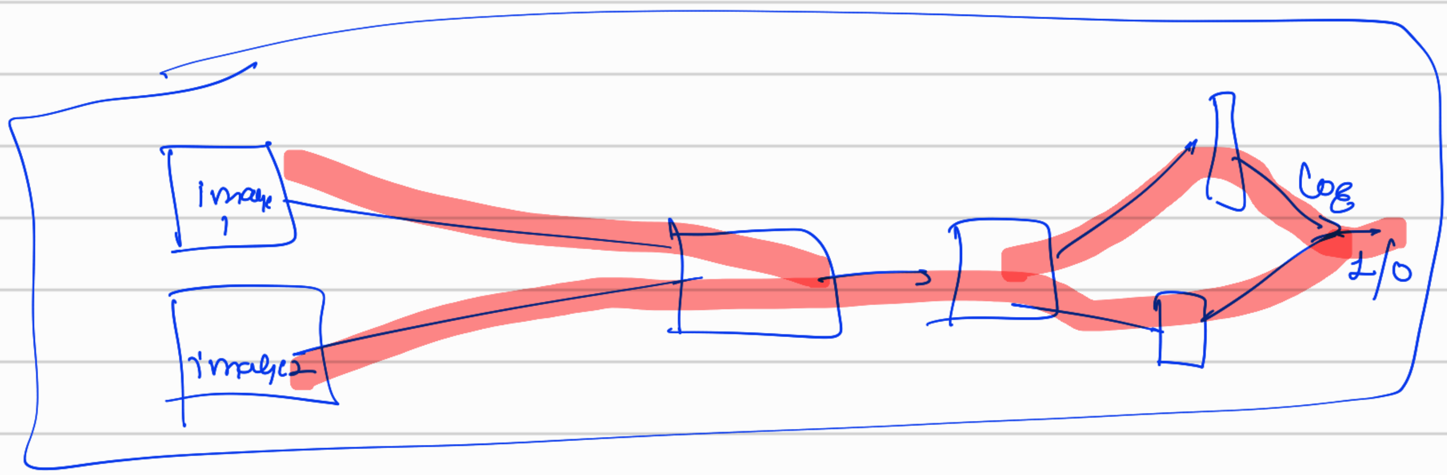


image 1
image2
1/0

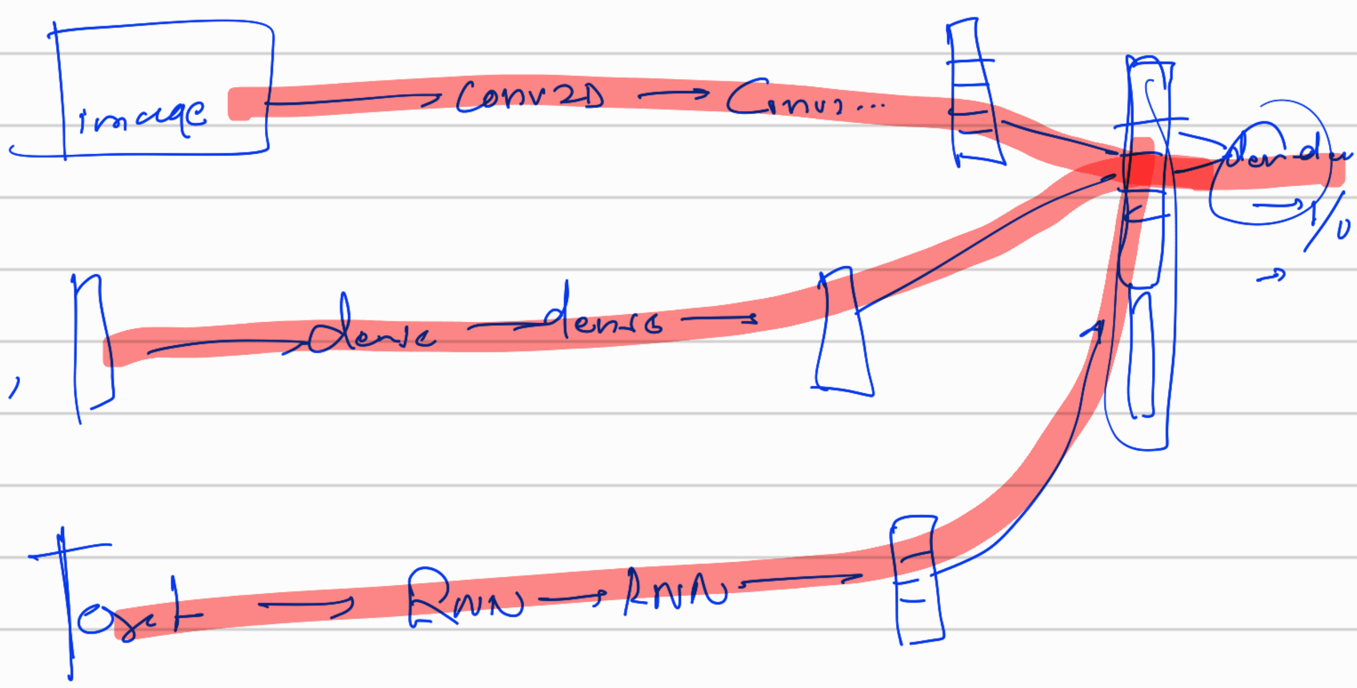
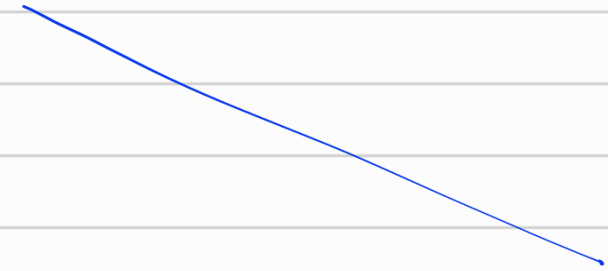
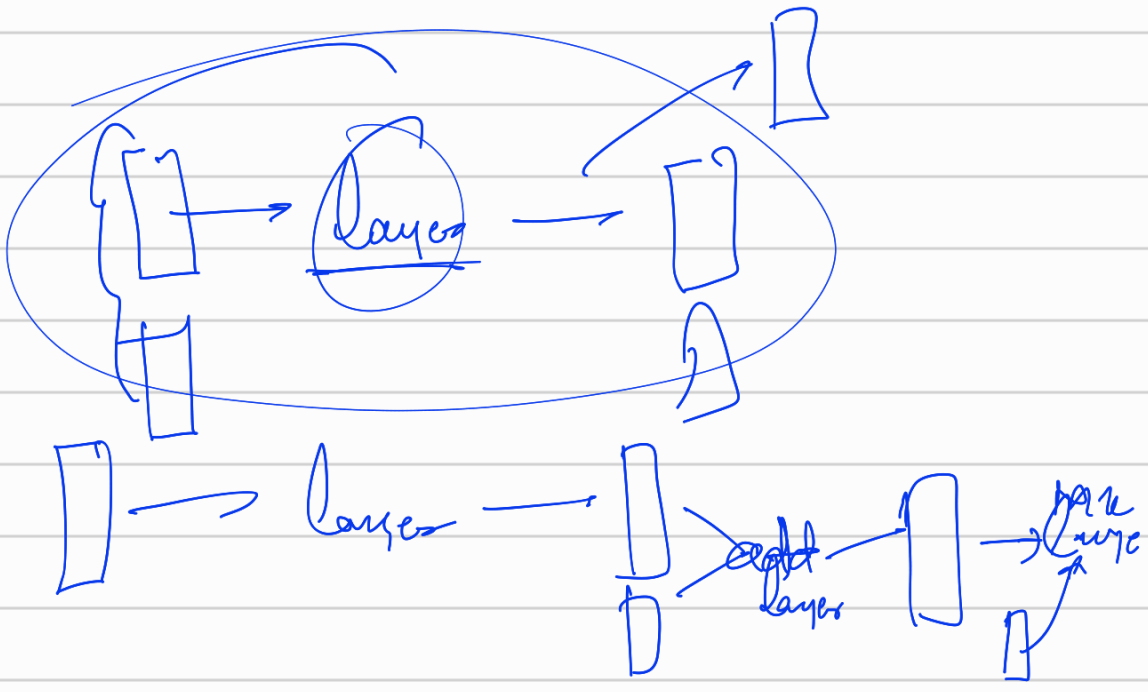


image
Conv2D
Conv2...
11,
dense
dense
Text
RNN
RNN
1/0



any [] → any layer

Combined → X



inp1 → dense → h1 → dense → h2 →

inp2 → dense → h2 →

Concat → dense →

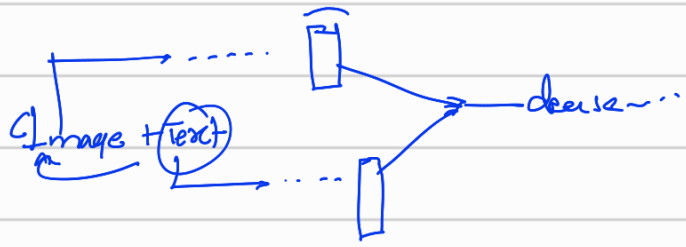
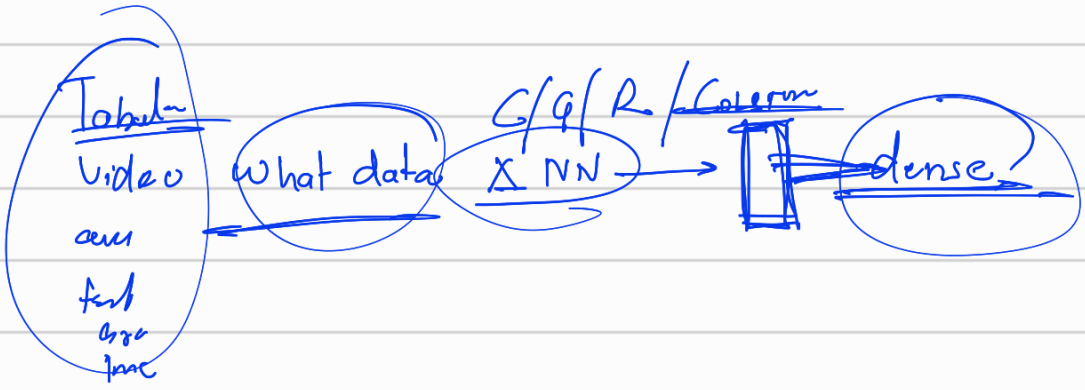
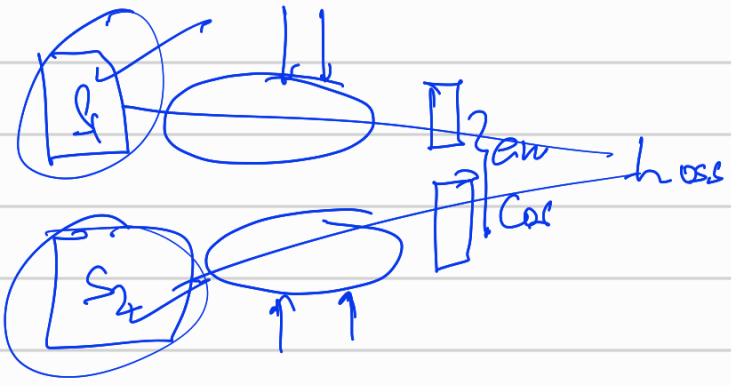


Image + text
dense


Tabular
Video
audio
text
Image
What data
NN
dense


loss

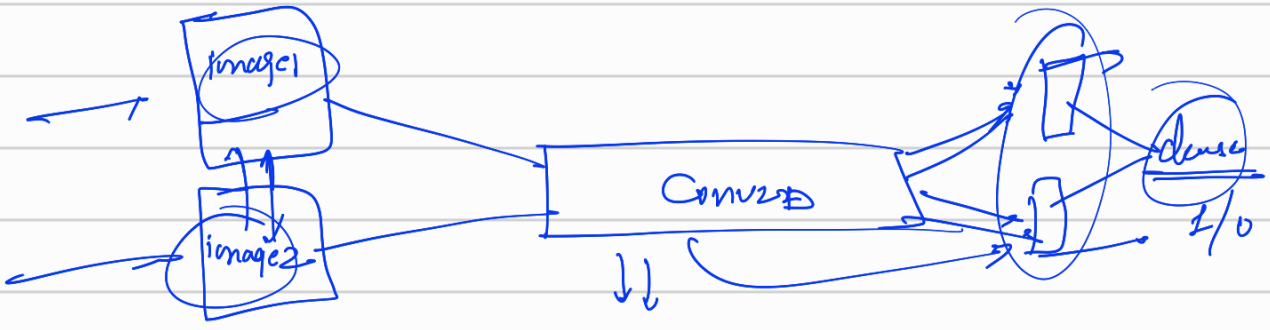


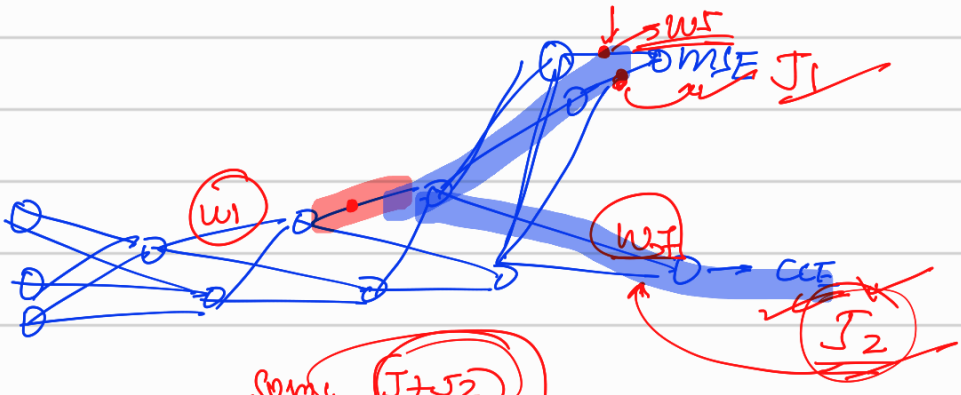
image1
image2
Concat
dense
1/0

22:36 Pros.

J1
J2

L = (J1 + J2)

Dropout:



Some (J1 + J2)

for all weights

d L = (J1 + J2)

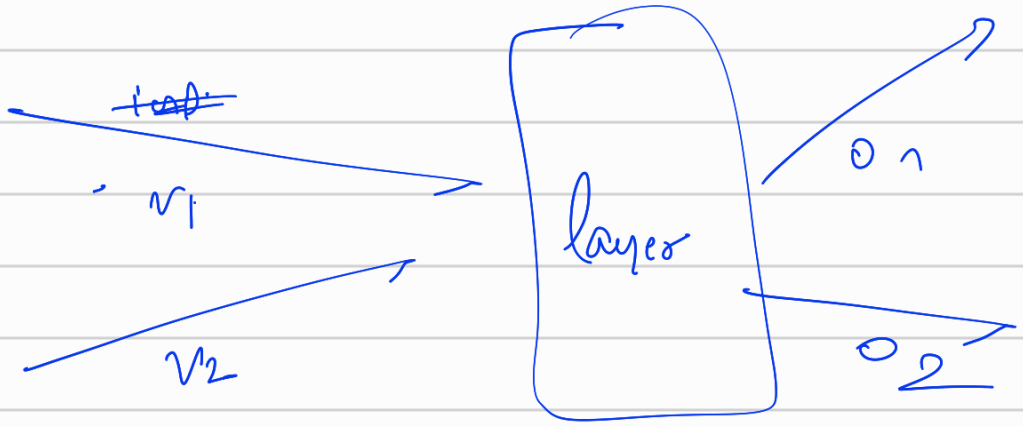
dJ1/dw1 + dJ2/dw1

$$L = J_1 + J_2$$

$$\frac{dL}{dw_5} = \frac{dJ_1}{dw_5} + \frac{dJ_2}{dw_5} = 0$$

inp1 = Input ( ) ——— dense( ) → D — dense

inp — (— or



10:36 pm

22:42 Pm

Loss

High ID | FB

Neurons / 47 students

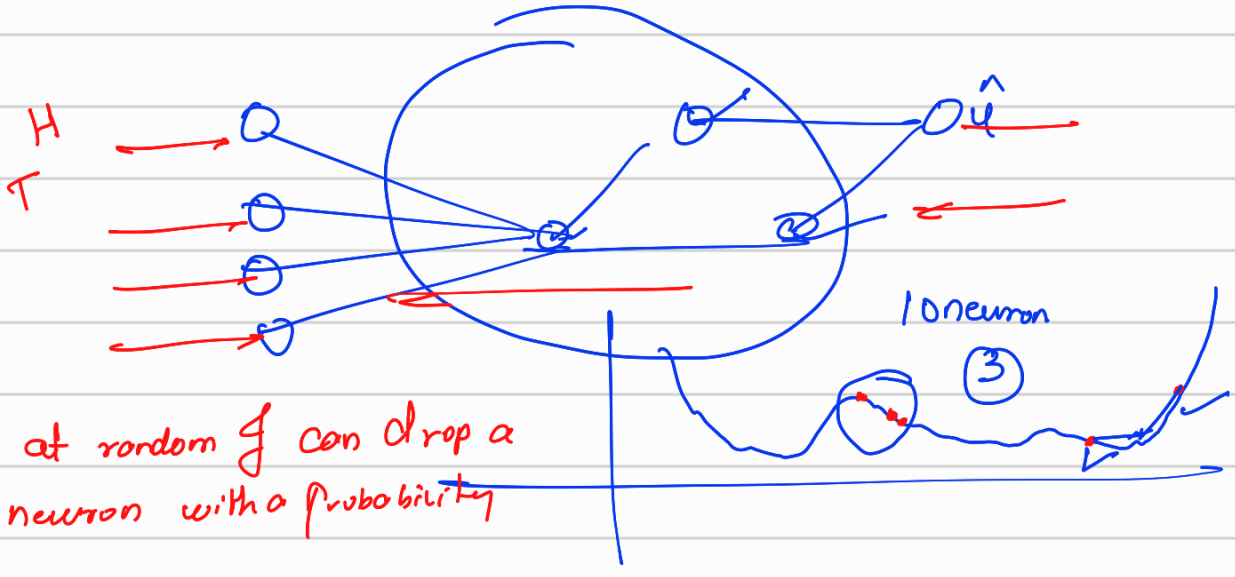
Dropout

Teacher 50 students → 50 students

3 students

asks

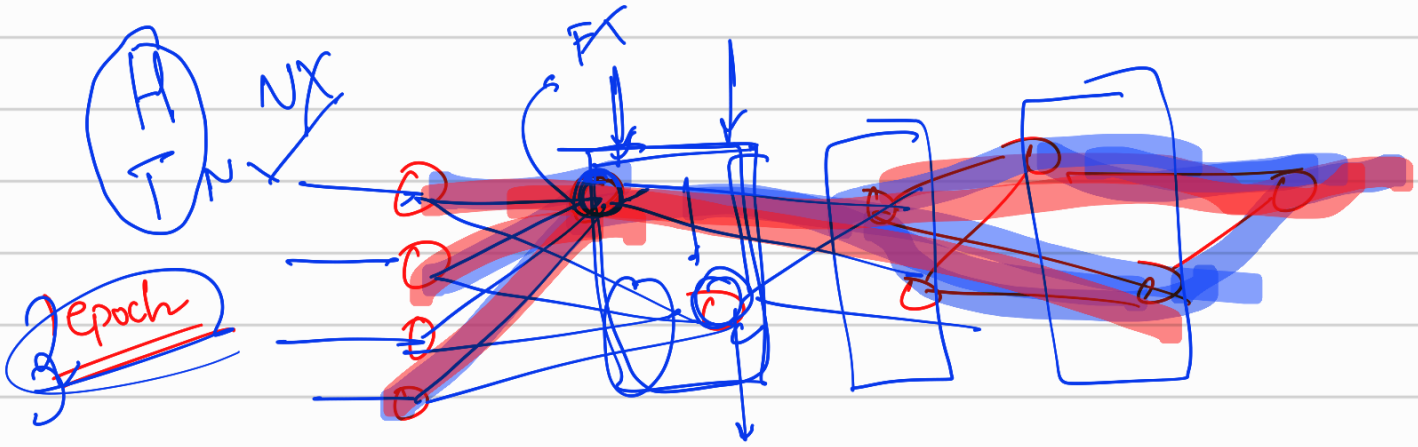
Teacher → cold calling Dropout

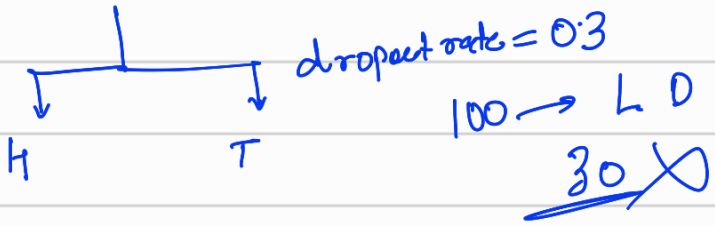


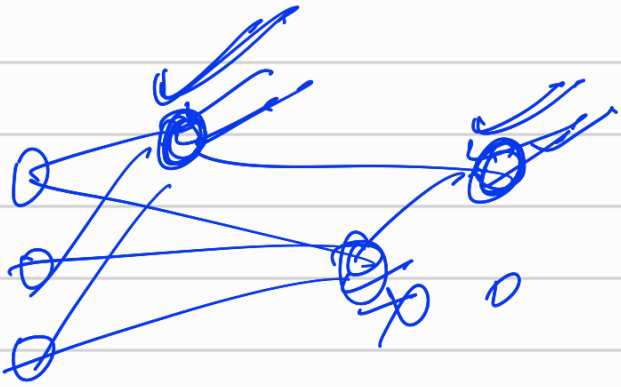
at random ∫ can drop a neuron with a probability

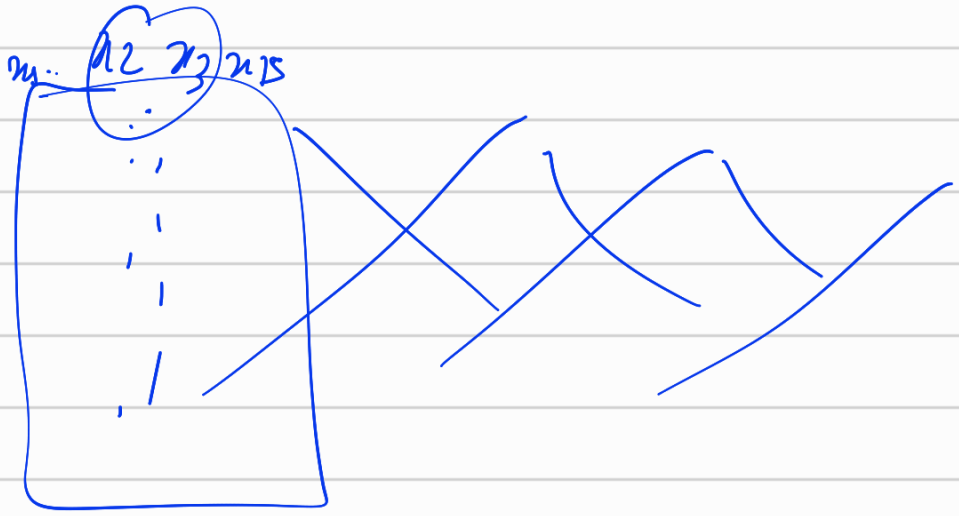
User / loss

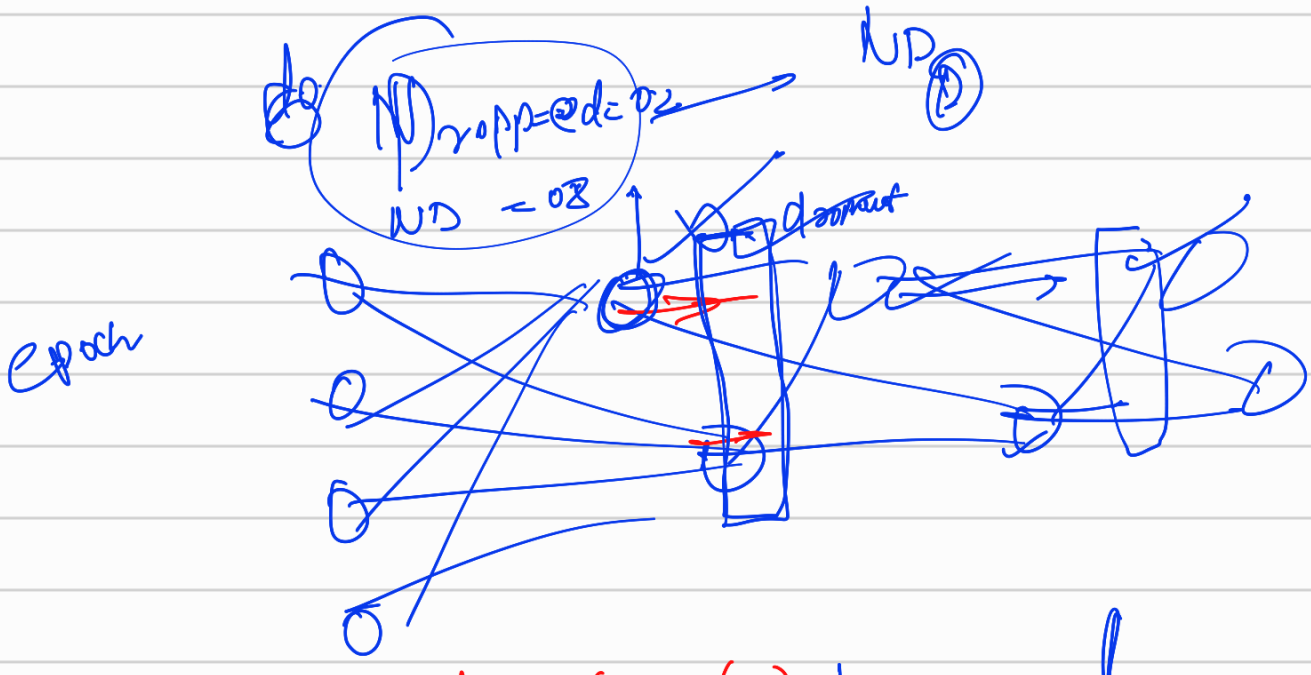
during that iteration the neuron will be dropped with a coin toss decision, if he is dropped then he will not (all weights incoming and outgoing) participate in the forward and backward propagation









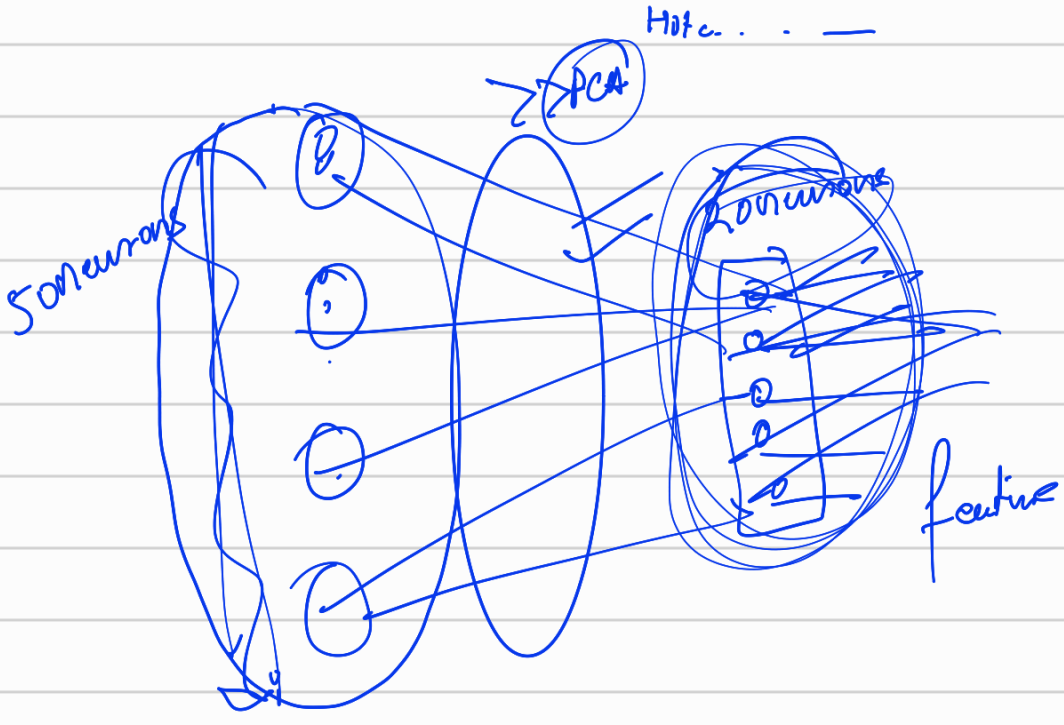


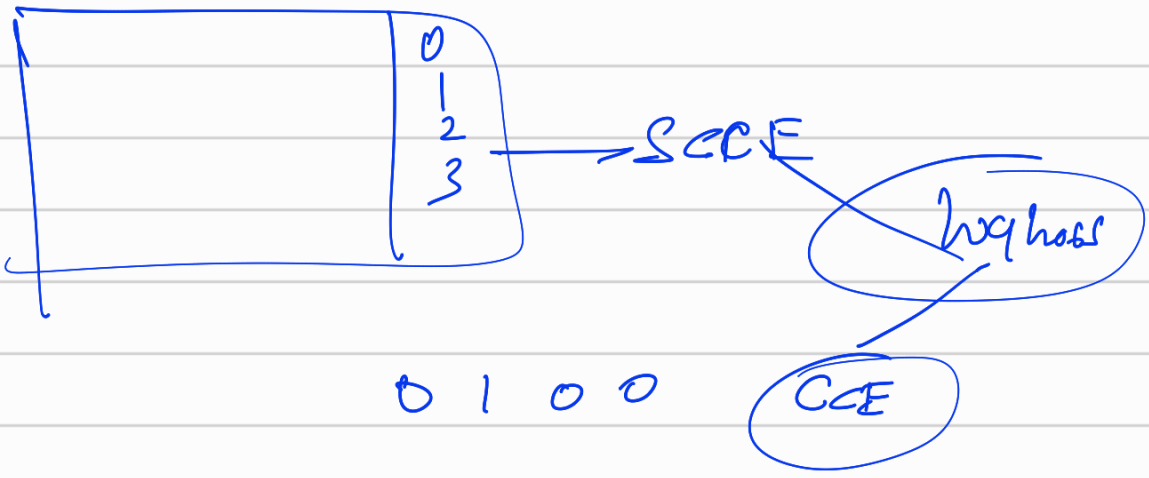
model.add(Dense(2))
model.add(Dropout(dropout_rate = 0.2))

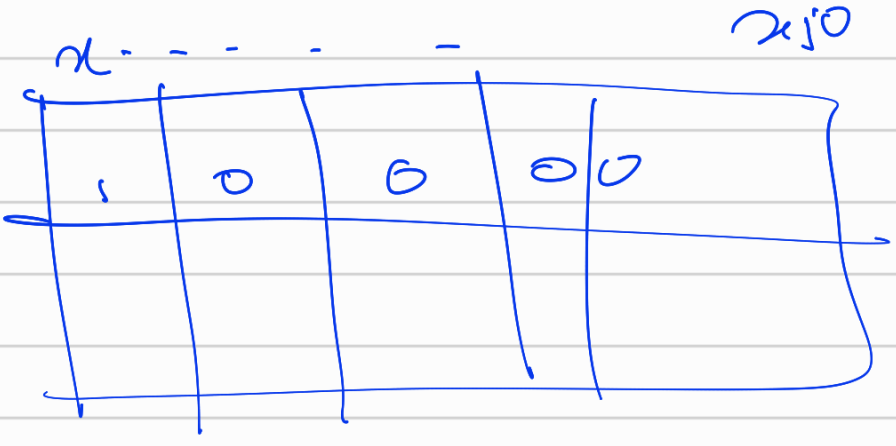
model.add Dense(2)

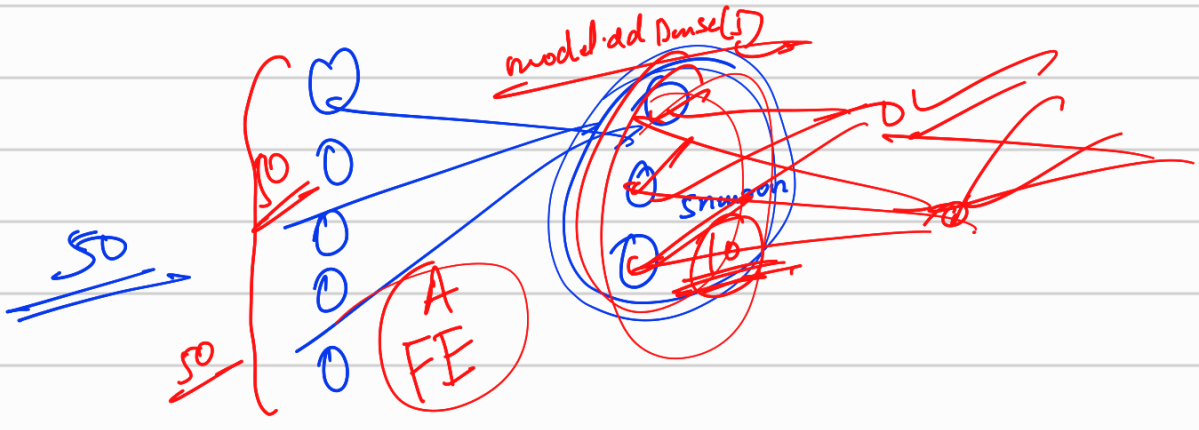
model.add Dropout

2 / 3 dense 1 Dr




0
1
2
3
SCCE
Ixq hoss
0 1 0 0
CCE


x
x10
1
0
0
0
0


model.add Dense(
10
50
50
A
FE