Deep learning Indexes

Ser no				
1	Introduction of DL	1	Recurrent Neural Network	81
2	ML vs DL	1	RNN Forward propation	85
3	Perceptron	2	Types of RNN	86
4	Loss function	7	Backpropagation in RNN	88
5	Gradient Descent	9	LSTM and GRU	92
6	Multi-layer-Perceptron Notation	11	Deep RNN-stacked RNN	96
7	Multilyer Perceptron	12	Bidirectional RNN	99
8	Forward Propagation	13	Encoder-decoder	102
9	Loss functions in DL	15	Attention Mechanism	105
10	Back propagation	19	Bandanau/Luong Attention	107
11	Backpropagation The Way	26	Transformer	111
12	What is convergence	29	Self Attention	112
13	Memorization in Backprop	29	Multi-Head Attention	120
14	Gradient Descent Types	30	Positional Encoding	121
15	Vanishing Gradient Desc. probe	32	Normalization layer	125
16	Hyperparameters in DL	35	Transformer deep	129
17	Early stopping	37	Masked-Self-Attention	131
18	Normalizing Input Data	37	Cross Attention	134
19	Dropout layers	38	Transformer decoder Archi	136
20	Regularization	40	Transformer Inference	139
21	What are active functions?	41		
22	Types of active functions	42		
23	Why weights initial important	46		
24	Batch Normalization	51		
25	Batch Normalization The How	53		
26	Optimizers	54		
27	Exponentially Weight Moving Ave (EWMA)	55		
28	Momentum Optimization	57		
29	NAG (Nesterov Gradient Descent	58		
30	ADAGRAD	59		
31	RMS Prop	61		
32	ADAM optimizer	62		
33	Convolutional NN (CNN)	63		
34	Padding and stride	66		
35	Backpropagation in CNN	71		
36	Data Augumentation	74		
37	Pre-Trained Models	75		

CamScanner

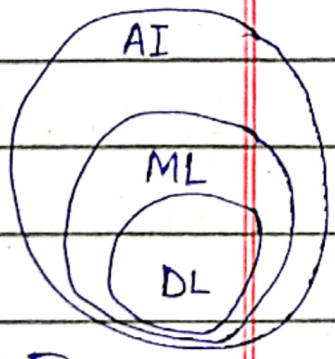
Date: __/__/2025 M T W T F S

Deep Learning

— Deep Learning is a subfield of AI and machine Learning that is ispired by the structure of human brain.

ANN → Simple Artificial NN

CNN, RNN, GAN.



— Why DL getting so famous?

① Applicability → Usage:-

: Computer Vision, Machine Traslation, Medical Image Analyzation, Web design, NLP, Speech Recognition, Self-driving Cars.

② Performance:- Its performance on these fields is awesome.

— Machine Learning vs Deep Learning

Machine learning algorithem works on statistical mind. (techniques).

Deep learning is works on logical structure based on human mind.

Deep learning works on more data.

① Data Dependency

Date: __/__/2025 M T W T F S

② Hardware Dependency

③ Training Time ⑤ Feature Selection

feature وہ خود بخود select کر لیتا ہے

⑥ Interaparability

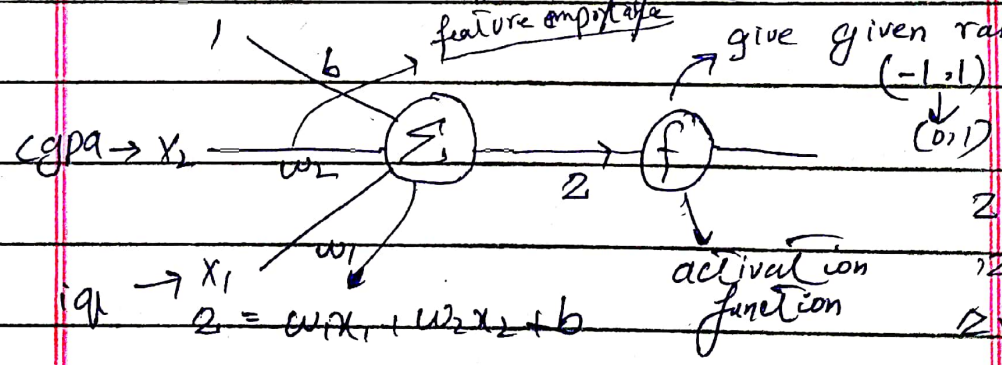
پتہ نہیں اس نے اسے reject کیوں کیا۔ اس نے کسی بچے پر اسے false کر دیا۔

Perceptron

fundamental of the ANN.

What is Perceptron?

It is algorithem used in supervised ml. You it call it a mathematical model/function.



Difference btw Neuron & perceptron.

① Neuron is very complex to perceptron.

② processing

③ Neuroplasticity بڑھتے رہتے ہیں کر پر Perceptron کے جیسے رہتے ہیں۔ Dendrites

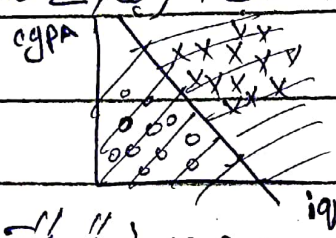
cgpa , iq | placement |

Date: __/__/2025

$$z = w_1x_1 + w_2x_2 + c$$

$$z = Ax_1 + Bx_2 + C$$

تو یہ ایک line کی equal ہے۔ بلکہ اگر چار data میں یہ points ہیں۔



تو یہ جو ہم کچھ نہیں کر رہے ہیں جو بتا رہے ہیں کہ placement ہوگا یا نہیں اگر z کی value صفر سے زیادہ ہوگی یا برابر ہوگی تو 1 اگر ورنہ 0 آئے گا۔

$$z = Ax_1 + Bx_2 + C \geq 0$$ placement

perceptron is nothing but a line جس کا تو create region کرنا۔ یعنی z کی value نکالتا ہے۔

perceptron is a binary classifier

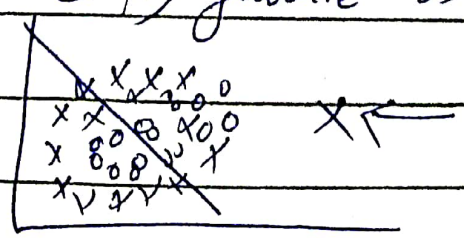
جو کہ آپ کے data کو 2 classes میں divide کرتا ہے۔

iq | cgpa | gender | placement :-

تو اب perceptron جو ہے وہ plane کو 2 حصوں میں line کی divide کرتا ہے۔

Disadvantage :-

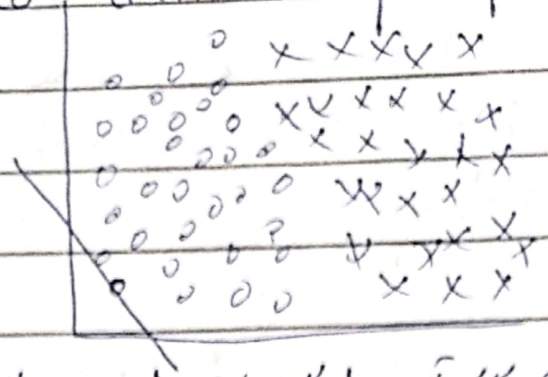
perceptron صرف linear یا linearly data sort of linear پر کام کرتا ہے۔ یا اس کی data پر کام کرے گا۔ جس کو یہ دو line میں divide کر سکتا ہے۔ مگر اس کے پاس ایسا data آ گیا تو وہ failure کر جائے گا۔



Perceptron Trick

Date: __/__/2025

How To Train a perceptrone?



سب سے پہلے آپ ایسے ہی b ، w_1 اور w_2 کی value پکڑ
کر دو۔ $b=0$ ، $w_2=1$ ، $w_1=4$ draw line کرنے کے
بعد آپ پوچھو گے point سے کہ آپ اچھی جگہ ہے یا نہیں۔
اور وہ کہے گا کہ میں تو اچھی جگہ ہوں پھر دوسرے p سے
پوچھو گے اور وہ کہیں گے کہ نہیں میں تو اچھی جگہ
پر نہیں ہوں تو آپ پھر اسے line کے اندر لانے
کی کوشش کریں گے اور تو ایسے طریقے سے آپ point کو
اندر لانے یعنی اچھی جگہ لانے کی کوشش کریں گے۔
یہ کام آپ loop کے ذریعے کریں گے۔

How To identify the possitive region and negative region?

اگر $z \geq 0$ تو وہ positive ہوگا۔

$$2x+3y+5>0$$

اگر آپ کو point کے بارے میں پتہ کرنا ہے کہ وہ positive region میں ہے یا negative میں۔

$$2x+3y+5<0$$

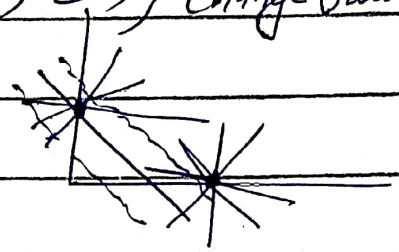
Date: __/__/2025 (M) (T) (W) (T) (F) (S)

How To move the line:-

اگر C کی value بڑھا دیں گے تو change line parallely کرو گے۔

اگر A کی change value کرو گے تو y پر rotation کرو گے۔

اگر B کی change value کرو گے تو x پر change rotation کرو گے۔



اگر C کی value بڑھا دیں گے تو line نیچے آئے گی اور اگر C کی value کو کم کرو گے تو line اوپر جائے گی۔

$$4x + 5y - 7 = 0$$

coefficients

$C_{new} = C_{old} - \eta$ coordinates

اگر آپ positive point کو neg. کی طرف جانا چاہتا ہے تو آپ coefficients میں سے coordinates کو remove کریں گے اور اگر negative point کو poss. کی طرف جانا چاہتا ہے تو آپ یہ کریں گے

$C_{new} = C_{old} + \eta$ coordinates

	cgpa	iq	placement
x_0	x_1	x_2	y
1	8.9	75	1

$$Ax + By + C = 0$$

$$w_0 + w_1x_1 + w_2x_2 = 0$$

Date: ___/___/2025 M T W T F S

$w_0 + w_1 x_1 + w_2 x_2 = 0$

$w_0 = c, \quad w_1 = A, \quad w_2 = B$

$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$

~~$w x_0 + w x$~~ $\begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$

$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$

اگر اس کی value زیادہ ہوا 0 سے تو 1 ورنہ 0 لگا دیں۔

epochs = 1000, $\eta = 0.01$

for i in range(epochs):-

* randomly select a point

if $x_i \in N$ but our $\sum_{i=1}^{n} w_i x_i \geq 0$

$w_{new} = w_{old} - \eta x_i$

if $x_i \in P$ but our $\sum_{i=1}^{n} w_i x_i < 0$

$w_{new} = w_{old} + \eta x_i$

Updated Code:-

عین ہے condition if نہیں لگانی بس بلکہ عین ہے

0 کی value کو update کرنا ہے۔

for i in range(1000):-

* select random number.

$w_{new} = w_{old} \pm \eta x_i$

according to our point location

Date: __/__/2025

Problem with Perceptron Trick

آپ line کے بارے میں 100% sure نہیں ہوتے کہ وہ درست ہیں یا نہیں یعنی آپ w, b کی value پر sure نہیں ہوتے کہ وہ ٹھیک ہے۔

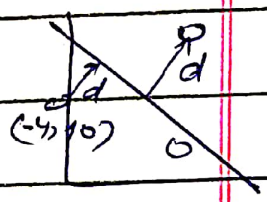
Is best or any other best line exists.

Loss Function :-

loss func. کا مطلب یہ ہے کہ ہماری predicted line جو ہے وہ best line سے کتنا دور ہے یا کتنے error پر ہے۔

by Distance formula:-

$$d_1 + d_2 = 10 + 12 = 22$$



but we did not use this:

we use simple perceptron loss:

we find coordinates of that point

اور اگر line کی equation ہے $3x + 4y - 7$

$$3(-4) + 4(10) - 7 = -12 + 40 - 7 = 21$$

اسی طرح دوسرے point کا error بھی معلوم کر لیں گے

Perceptron loss Function in Sklearn:-

$$L = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w_1, w_2)$$

Regularization, we just learn later

$$L(y_i, f(x_i)) = \max(0, -y_i f(x_i))$$

$$f(x_i) = w_1 x_1 + w_2 x_2 + b$$

$$L = \frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i f(x_i))$$

Date: __/__/2025 M T W T F S

we find a value of w_1, w_2, b

جس کے لیے اس expression کی minimum value ہو جائے

اس کے لیے ہم gradiant desant کا استعمال کریں گے۔

اب اس کا مطلب کیا ہے۔

$$f(x_i) = w_1 x_{11} + w_2 x_{12} + b$$

$$\max(0, -y_i f(x_i))$$

اس میں 0 اور $-y_i f(x_i)$ میں سے ایک value کو select کریں گے۔ اگر value 0 سے بڑی ہوگی

$$-y_i f(x_i) \geq 0$$

تب ہم اس value کو select کریں گے max کا مطلب یہی تو ہوتا ہے نا۔

سو guess اگر ہمارے پاس دو کام ہیں تو تب کیسے sklearn کی مدد سے calculate loss کیا جاتا ہے۔

$$L = \frac{1}{2}\left[\max(0, -y_1 f(x_1)), \max(0, -y_2 f(x_2))\right]$$

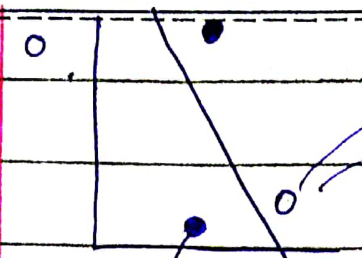
if $n = 3$

$$L = \frac{1}{3}\left[\max(0, -y_1 f(x_1)), \max(0, -y_2 f(x_2)), \max(0, -y_3 f(x_3))\right]$$

$$f(x_2) = w_1 + w_1 x_{21} + w_2 x_{22}$$

cgpa	iq	placedmant
7	8	1
6	8	-1
4	2	1
1	1	-1

Date: ___/___/2025 (M)(T)(W)(T)(F)(S)



$\hat{y} = 1$ $\max(0, -y_i f(x_i))$

$\max(0, -(+ive)(+ive))$

$\max(0, -ive)$

$= 0$

مطلب L میں یہ point کا 0 contribution

$\max(0, -y_i f(x_i))$

$= \max(0, -ive(-ive)(+ive))$

$= \max(0, +ive)$

$= +ive$

مطلب loss میں یہ contribute کرے گا۔

function loss تو classify کیا ہے model آپ کا net shell

اس کا contribution ہو گا وہ 0 ہو گا۔

Gradient Descent

اب ہمیں w_1, w_2, b کا ایسا value نکالنا ہے جس کے لیے minimum value کا $= \frac{1}{n}\sum_{i=1}^{n}(\max(0, -y_i f(x_i)))$ ہو۔

we randomly value of $w_1, w_2, b = 1$

for in epochs:

$$w_1 = w_1 + \eta \frac{dL}{dw_1}$$

$$w_2 = w_2 + \eta \frac{dL}{dw_2}$$

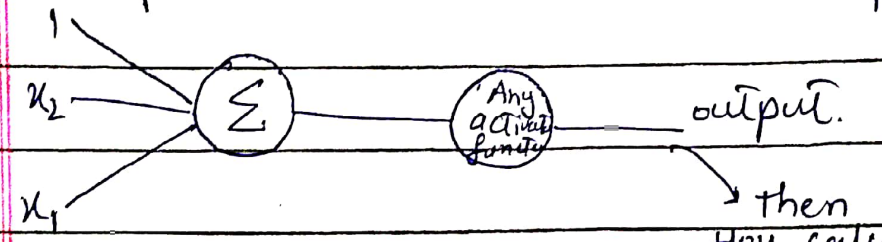
$$b = b + \eta \frac{dL}{db}$$

perceptron = Logistic Regression

perceptron is only work on linear data not work on nonlinear data.

Date: __/__/2025 M T W T F S

Perceptron is a mathematical formula.



then you calculate loss function.

perceptron is flexible by design.

اگر آپ کو output کی جگہ 0 اور 1 کی بجائے probability میں

چاہیے تو آپ activation fun کو change کر کے sigmoid کر دیں گے

binary crossentropy loss funct

$$L = -y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$$

sigmoid function $= \sigma(z) = \frac{1}{1+e^{-z}}$

For multiclass we also change

activation = softmax =

L = categorical-crossentropy

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

You can also use perceptron for Regression:

activation func = linear or nothing

L = mean-squared-error

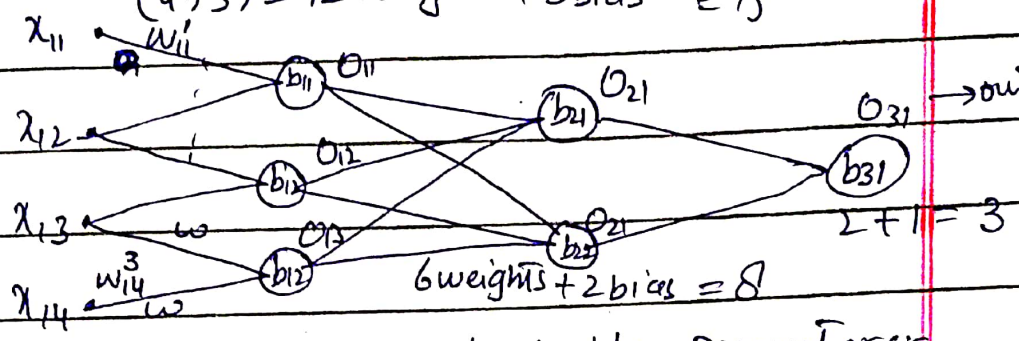
Loss function	activation	output
Hinge loss	step	perceptron → binary classif
Log-loss (binary cross entropy)	sigmoid	Logistic Regress
~~categorical~~ crossentropy	softmax	softmax regression (multi class classifing)
mse	linear	linear Regression (number)

Date: __/__/2025

MLP Notation:-

جب ہم backpropagation کر رہے ہوتے ہیں تو ہمیں اسے یاد کرنے میں بہت مسئلہ ہوتا ہے اس کا سب سے بڑا reason weights اور bias یعنی parameters کو اچھے سے denote نہیں کرتے۔

(4,3) = 12 weights + 3 bias = 15



how many trainable parameters:-

There are 26 parameters :-

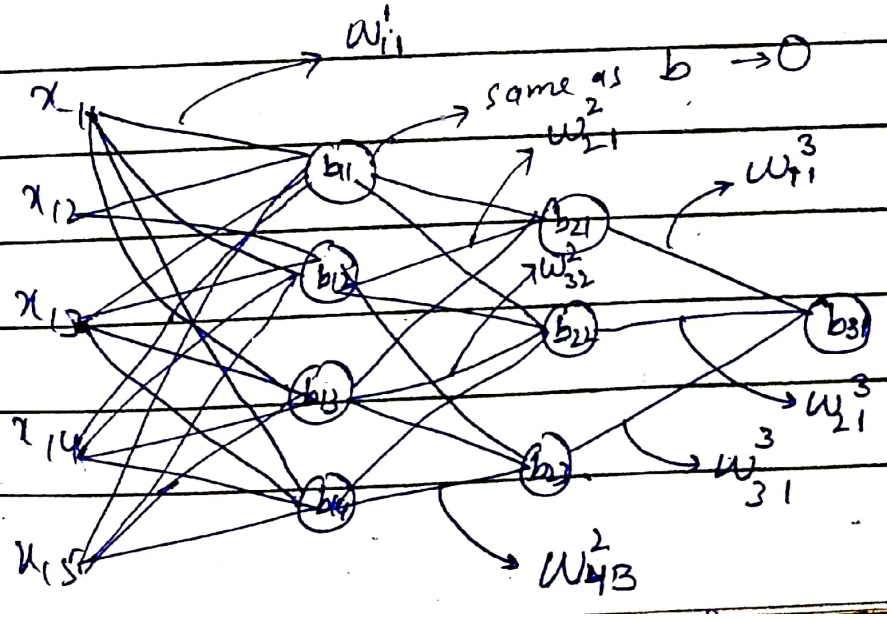
اب ہم w, b کو کیسے notate کرتے ہیں۔

for b = b_{ij}, b_{11} for first layer first node

for node = O_{ij} same as bias.

for weights, you need i, j, k => w^k_{ij}

k means = ~~node~~ Layer



The output of sigmoid is probability of True class =

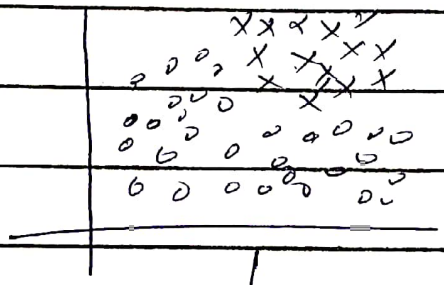
False class = 1 − True_probability

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

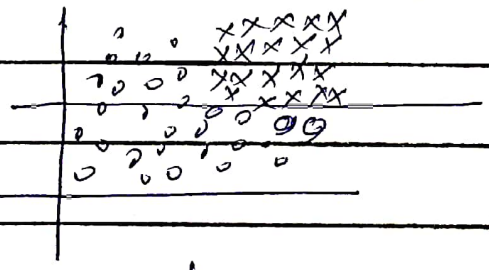
Multi-Layer Perceptron:-

perceptron صرف وہ کام کر سکتا ہے جو data linear کے ساتھ ہو۔ data non-linear کے ساتھ کام نہیں کر سکتا۔ تو اس کے لیے ہم ML perceptron کا استعمال کریں گے۔

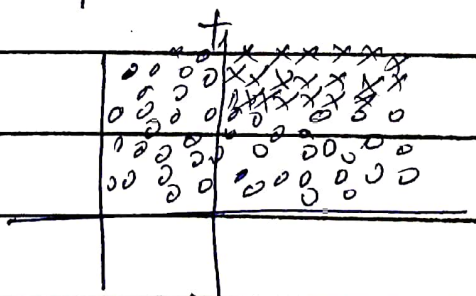
جتنے زیادہ perceptron add کرتے جائیں گے ہمارا model اتنے زیادہ data non-linear کے ساتھ اچھی performance دے گا۔

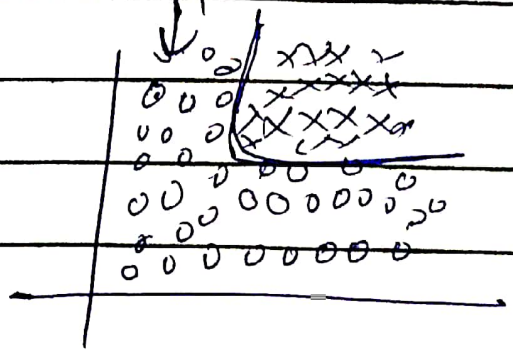


We Do this



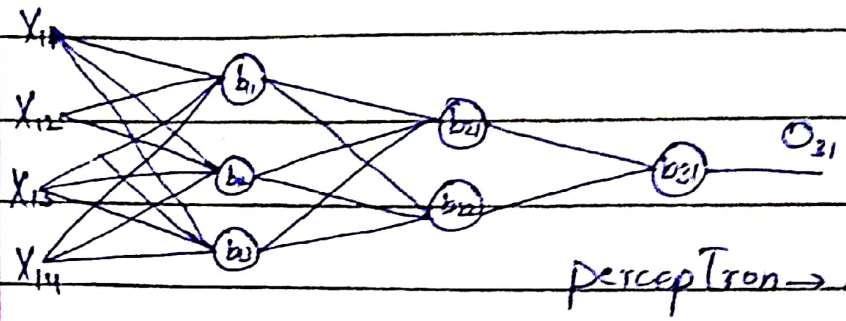
First we try one perceptron and then try another perceptron and then combine their result.





Date: ___/___/2025 M T W T F S

Forward Propagation



perceptron $\rightarrow \sigma(w^t x + b)$

Layer # 1

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}^t \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$$(4 \times 3)^t \cdot (4 \times 1) = (3 \times 1) + (3 \times 1)$$

$$a = \left(\begin{bmatrix} w'_{11} x_{11} + w'_{21} x_{12} + w'_{31} x_{13} + b_{12} \\ w'_{12} x_{11} + w'_{22} x_{12} + w'_{32} x_{13} + b_{12} \\ w'_{13} x_{11} + w'_{23} x_{12} + w'_{33} x_{13} + b_{13} \end{bmatrix} \right)$$

$$= \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix}$$

$\rightarrow$ output for first layer.

$\rightarrow a_0 \rightarrow$ output of zero layer

Then second layer:-

$$\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w^2_{31} & w^2_{32} \end{bmatrix}^t \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

$$(3 \times 2)^t \cdot (3 \times 3) \cdot (3 \times 1) = (2 \times 1) + (2 \times 1) =$$

$$= \sigma\left(\begin{bmatrix} \quad \end{bmatrix} \right) = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}$$

$a_1 \rightarrow$ output of 2nd layer

Date: ___/___/2025

For 3# layer

$$\begin{bmatrix} w_{11}^3 \\ w_{21}^3 \end{bmatrix}^T \cdot \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \\ \cancel{b_{32}} \end{bmatrix}$$

$$\begin{bmatrix} w_{11}^3 & w_{21}^3 \end{bmatrix} \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \\ \cancel{b_{32}} \end{bmatrix}$$

$$\begin{bmatrix} w_{11}^3 O_{21} + w_{21}^3 O_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \end{bmatrix}$$

$$\hat{y} = w_{21}^3 O_{21} + w_{21}^3 O_{22} + b_{31} = [O_{31}]$$

netshell میں آپ یہ کام کر رہے ہو۔

$$a^{[1]} = \sigma\left(a^{[0]} w^{[1]} + b^{[1]}\right)$$

$$a^{[2]} = \sigma\left(a^{[1]} w^{[2]} + b^{[2]}\right)$$

$$a^{[3]} = \sigma\left(a^{[2]} w^{[3]} + b^{[3]}\right)$$

$$\hat{y} = \underbrace{\sigma\Big(\underbrace{\sigma\Big(\underbrace{\sigma\left(a^{[0]} w^{[1]} + b^{[1]}\right)}_{a^{[1]}} w^{[2]} + b^{[2]}\Big)}_{a^{[2]}} w^{[3]} + b^{[3]}\Big)}_{a^{[3]}}$$

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

Loss Functions In Deep Learning

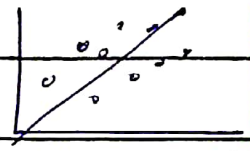
Loss function is a method of evaluating how well your algorithem is modeling your dataset:-

high loss function = bad performance.

"It also a mathematical function $f(x)$"

loss (parameter's)

loss (m, b)



$L = (y_i - \hat{y})^2$

$L(m, b)$

$= mx_i + b$

How is loss function important?

["You can't improve what you can't measure"] Peter Drucker

loss function is eye of ML algorithems.

$L = (y_i - \hat{y})^2$

for first, then second, then n-Times.

Loss Functions in DL:-

- Regression
  - mse, mae,
- classification
  - binary-crossentropy
  - category-crossentropy
- Object Detection
  - focal_loss.
- Autoencoders
  - KL divergence
- GAN
  - Discriminar loss
  - min-max
  - gan_loss

Date: ___/___/2025 M T W T F S

Loss function vs Cost function

y_i	$\hat{y}$
6.3	6.1
4.1	4
3.5	3.7
7.2	7

Loss function = single Training

$y_i = 6.3$, $\hat{y}_1 = 6.1$

Loss $= (y_i - \hat{y})^2$

Loss $= (6.3 - 6.1)^2 = 4$

Loss function or Error

But for all it called cost :-

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

Cost function $= \frac{1}{n}\left[(6.1 - 6.3)^2 + (4.1 - 4)^2 + (3.5 - 3.7)^2\right]$

$=$

— ① Mean Squared Loss / L2 Loss

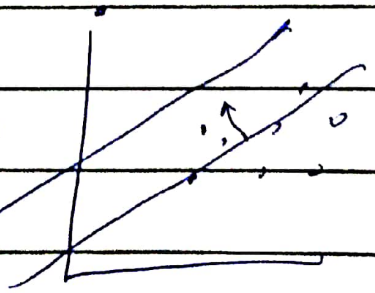
$(\text{true} - \text{predicted})^2$

$= (y - \hat{y})^2 = (6.1 - 6.3)^2$

ہم اس value کو minimize کرتے ہیں

اور weights کو update کرتے ہیں

Advantages:-	Disadvantage
i) Easy to interpret	error unit change $y = \hat{y}$ $(pa = 4pa^2)$
ii) Differentiable	not robust to outlier
iii) 1 local minima	



Last Dense Layer activation = 'linear'

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

(2) Mean absolute error

Loss function = $|y_i - \hat{y}_i|$

Cost function = $\frac{1}{2} \sum_{i=1}^{n} |y_i - \hat{y}_i|$

Disadvantage

i) Intuitive to understand

ii) Unit → same

iii) It is robust to outliers

Disadvantage

i) Graph

ii) It is not differentiable

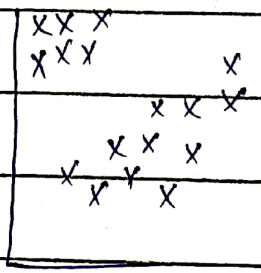
iii) Gradient Descent not applied.

(3) Huber Loss

$$L = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

75% = real data

25% = outliers



(i) ہمارا mae جو ہے وہ actual data کو دیکھتا ہے
اور outlier کو زیادہ
value نہیں دیتا ہے
طرف رہے کیونکہ وہ

(ii) ہمارا mse جو ہے 25% کو زیادہ value دے رہا ہے کیونکہ
وہ outliers کو prefer کرتا ہے۔

(iii) تیسرا ہمارا hubber جو ہے یہ line provide کر دیتا ہے۔
جس اور یہ بہت perform اچھی دیتا ہے جب آپ کے
پاس بہت سارے outliers نہیں ہیں۔

Date: __/__/2025 log-loss. M T W T F S

4. Binary crossentropy

→ classification

→ Two classes:-

Loss function = $-y \log \hat{y} - (1-y)\log(1-\hat{y})$

y = actual value

for binary-crossentropy you must give sigmoid activation function.

Cost function = $\frac{1}{n}\left[-y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i)\right]$

Advantage	Disadvantage
* Differentiable	Multiple local minimas

5. Categorical - Cross Entropy

→ Multi class classification

Loss = $-y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3) \ldots$

activation = softmax

Last layer Dense layer has Neurons equal to class length Dense(10, a-f='sof

$$f(z) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + \ldots}$$

Sparse - categorical Crossentropy

کا استعمال تب کریں جب ہمارے label میں

1,2,3,4 ہوں تو ہے نا

[0,1,0,0,0]

[0,0,0,1,0]

مطلب OHE نہیں ہوتا

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

Sparse CrosEropy is faster than Categorical Entropy
کیونکہ یہاں آپ کو data OHE میں نہیں ہوتا مطلب
آپ کے feature کم ہوتے ہیں۔

for one row:-

$$\text{Loss-function} = -\sum_{j=1}^{k} y_j \log(y_j)$$

$$\text{Cost-function} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\left(y_j \log(y_j)\right)$$

Back Propagation

It is an algorithem used to Train NN.

Steps:- 0 : initial b, w values

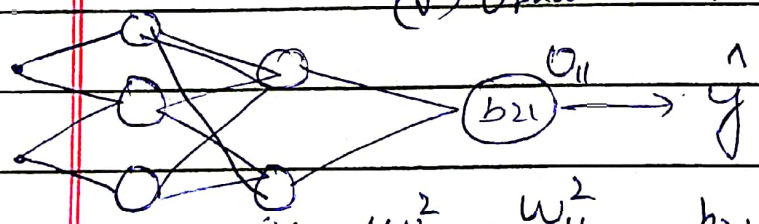
i) You select a point (row) → student

ii) Predicted Lpa → forward propagation

iii) Choose a loss function (mse)

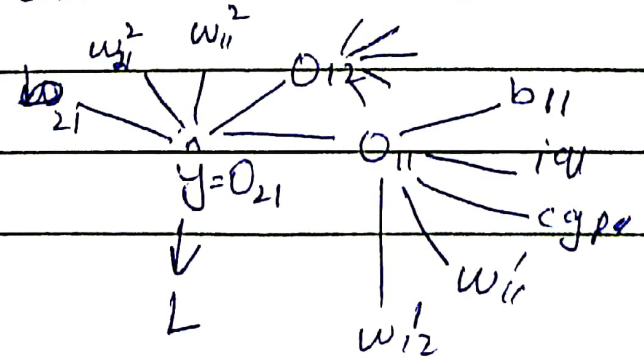
(iv) Calculate loss

(v) Updates bias and weights → Gradient Des



اب یہاں پر $\hat{y}$ ہمارا b_{21} ، w^2_{11} ، w^2_{12} پر منحصر ہے

$$O_{21} = w^2_{11} O_{11} + w^2_{12} O_{12} + b_{21}$$



partial Derivative = Gradient

$$\frac{\partial L}{\partial w/b}$$

Date: __/__/2025 M T W T F S

پھر ہم یہی 5 steps کو repeate کرتے رہیں گے۔
output نکالیں گے loss نکالیں گے پھر weights اور bias
according to Gradient Descent technique کو update کریں گے

Simple پھنڈا

جب ہمارے پاس $\hat{y}$ آ جائے گا اور ہم loss نکالیں گے۔
تو پھر weights اور bias کو update کریں گے۔

$$w_{11}^1 = w_{11\,old}^1 - \eta \, slop\left(\frac{dL}{dw_{11}}\right)$$

$\hat{y}$ depends on first $w_{11}^2, w_{21}^2, b_{21}$
تو ہم ان کی values کو پہلے update کریں گے۔

$$w_{21}^2 = w_{21\,old}^2 - (\text{learning rate}) \frac{\partial L}{\partial w_{21}}$$

directly کو $\frac{\partial L}{\partial w_{21}}$ پر ایک مسئلہ ہے ہم
نہیں کر سکتے

$$b_{21} = b_{21\,old} - \eta \boxed{\frac{\partial L}{\partial b_{21}}}$$

we find this one

$$\frac{\partial L}{\partial w_{11}^2}, \frac{\partial L}{\partial w_{21}^2}, \frac{\partial L}{\partial b_{21}} \rightarrow \frac{\partial L_1}{\partial w_{11}}, \frac{\partial L}{\partial w_{21}}, \frac{\partial L}{\partial b_{11}}$$

w_{11} پر depend کرتا ہے $\hat{y}$

$\hat{y}$ پر depend کرتا ہے L

$\hat{y} \rightarrow w_{11}$

Date: __/__/2025

$L(min) \rightarrow \hat{y}$

L کو min کرنے پر تلا ہوا ہے

اگر تو اب ہمیں کام کرنا ہے کہ جب میں w_{11}^2 میں change کر رہا ہوں تو L میں کیا change ہو رہا ہے۔

M T W T F S

جو چیز ایک دوسرے پر depend نہیں کرتے تو ان کے لیے اب ہم کیا کریں گے۔

when you calculate dy/dx

y تو یہ نکالتے ہیں کہ جب x میں چھوٹی سی change کیا تو y میں کیا change ہوا۔

You can't $\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_{11}^2}$

$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} (y - \hat{y})^2 = 2(y - \hat{y})(-1)$$

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

$$\frac{\partial \hat{y}}{\partial w_{11}^1} = \frac{\partial}{\partial w_{11}^1} \left(o_{11} w_{11}^2 + o_{12} w_{21}^2 + b_2 \right)$$

$$= o_{11}$$

$$\boxed{\frac{\partial L}{\partial w_{11}^1} = -2(y - \hat{y}) o_{11}}$$

$$\frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_{21}^2}$$

$$\frac{\partial \hat{y}}{\partial w_{21}^2} = \frac{\partial}{\partial w_{21}^2} \left[o_{11} w_{11}^2 + o_{12} w_{21}^2 + b_{21} \right]$$

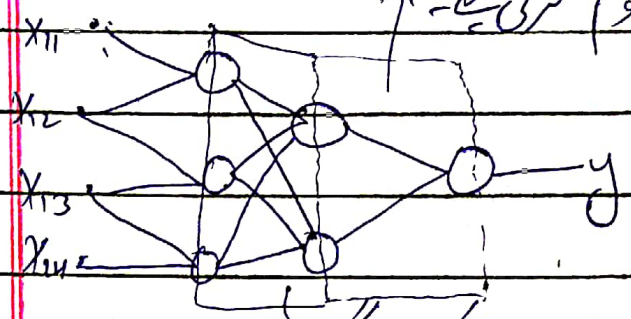
$$= o_{12}$$

$$\boxed{\frac{\partial L}{\partial w_{21}} = -2(y - \hat{y}) o_2}$$

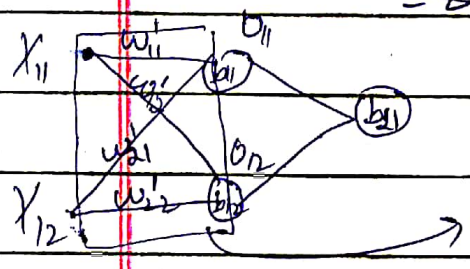
Date: ___/___/2025 M T W T F S

$$\boxed{\frac{\partial L}{\partial b_{21}} = -2(y-\hat{y})}$$

ہم نے یہ چیز معلوم کر لی ہے۔



اسی طرح اب ہم یہ چیز معلوم کریں گے۔



o_{11} سے w^1_{11} کی طرح

o_{12} o_{11} $\hat{y}$ L

$$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w^1_{11}}$$

$$\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w^1_{12}}$$

اسی طرح ہم o_{12} کے لیے بھی معلوم

$$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial b_{11}}$$

یہ ہم نے ابھی معلوم کر لی ہیں۔

یہ ہم معلوم کر سکتے ہیں۔

$$\frac{\partial \hat{y}}{\partial o_{11}} = \frac{\partial}{\partial o_{11}}\left[w^2_{11} o_{11} + w^2_{21} o_{21} + b_{21}\right]$$

$$= w^2_{11}$$

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

$$\frac{\partial L}{\partial w_{11}} = -2(y-\hat{y})\, w_{11}^{2}\, X_{i1}$$

$$\frac{\partial L}{\partial w_{21}} = -2(y-\hat{y})\, w_{11}^{2}\, X_{i2}$$

$$\frac{\partial L}{\partial b_{11}} = -2(y-\hat{y})\, w_{1}^{2}$$

Steps:-

0) weights / biases initialize → randomly, $b=0$, $w=1$

i) for i in range (4):-

1a) 1-~~student~~ forward - predict (Lpa)

1b) loss calculate (mse)

1c) ajust all weights and biases.

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

Backpropagation algorithem:-

for i in range (epochs):

for j in range (X.shape[0]):

→ select 1 row (random)

→ Predict (using forward propagat)

→ calculate loss (using loss function → mse)

→ update weights & bias using GD

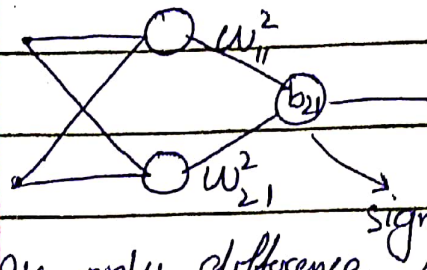
$$w_n = w_0 - \eta \frac{\partial L}{\partial w}$$

calculate avg loss for the epochs

Date: __/__/2025

M T W T F S

Now Backpropagation for Logistic Regression.



$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_{11}^2}$$

$$\frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_{21}^2}$$

$$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial b_{21}}$$

Its only difference.

$\hat{y} \to z \to w_{11}^2$

$$L = [-y \log(\hat{y}) - (1-y)\log(1-\hat{y})]$$

binary-cross-entropy

$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\left(-y\log(\hat{y}) - (1-y)\log(1-\hat{y})\right)$$

$$= \frac{-y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})} \qquad \because \frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1-\sigma(z)]$$

$$= \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})}$$

$$\frac{\partial \hat{y}}{\partial z} = \frac{\partial}{\partial z}\left(\sigma(z)\right) = \sigma(z)[1-\sigma(z)]$$

$$= \hat{y}(1-\hat{y})$$

$$\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} = \frac{-(y-\hat{y})}{\cancel{\hat{y}(1-\hat{y})}} \times \cancel{\hat{y}(1-\hat{y})}$$

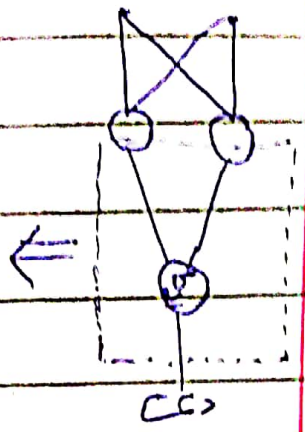
$$= -(y-\hat{y})$$

Date: __/__/2025

$$\frac{\partial L}{\partial w_{11}^2} = -(y-\hat{y})\,O_{11}$$

$$\frac{\partial L}{\partial w_{21}^2} = -(y-\hat{y})\,O_{12}$$

$$\frac{\partial L}{\partial b_{21}} = -(y-\hat{y})$$



Now first layer:- and first node:-

$$L \rightarrow \hat{y} \rightarrow z_f \rightarrow O_{11} \rightarrow z_{prev} \rightarrow w_{11}^1$$

$$\frac{\partial L}{\partial w_{11}^1} = -(y-\hat{y})\,w_{11}^2\,O_{11}(1-O_{11})\,X_{i1}$$

$$\frac{\partial L}{\partial w_{21}^1} = -(y-\hat{y})\,w_{11}^2\,O_{11}(1-O_{11})\,X_{i2}$$

$$\frac{\partial L}{\partial b_{11}} = -(y-\hat{y})\,w_{11}^2\,O_{11}(1-O_{11})$$

First layer and Second node:-

$$\frac{\partial L}{\partial w_{12}^1} = -(y-\hat{y})\,w_{21}^2\,O_{12}(1-O_{12})\,X_{i1}$$

$$\frac{\partial L}{\partial w_{22}^1} = -(y-\hat{y})\,w_{21}^2\,O_{12}(1-O_{12})\,X_{i2}$$

$$\frac{\partial L}{\partial b_{12}} = -(y-\hat{y})\,w_{21}^2\,O_{12}(1-O_{12})$$

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

Backpropagation : The WAY

→ loss function is a function of all trainable paraparameters w_1, w_2

$L(?) = L(y - y^2) \Rightarrow L = (c - y^2)$

$L(\text{parameters}) \Rightarrow L(w_{11}, w_{12}, \ldots, b_{31})$

→ Concept of Gradient

fancy word of derivative.

$y = f(x)$

پہلی چیز ہمارا function ایک ہی parameter پر depend کرتا ہے۔ تو اس سے ہم derivative بولا سکتے ہیں۔ دوسرا اگر

$z = f(x, y) = x^2 + y^2$

$\frac{\partial z}{\partial x^2} = 2x, \quad \frac{\partial z}{\partial y^2} = 2y$

دو یا دو سے زیادہ parameter پر مشتمل ہو تو اس سے ہم gradient پکارتے ہیں ← 9-dimentional ← parameter 9

Complex ← $L(w_{11}, w_{12}^1 \ldots w_{21}^2, w_{22}^2, b_2, b_1, b_2)$

gradient

— Concept of derievative →

$\frac{dy}{dx}$ ایک quantity کے change ہونے سے دوسری quantity پر کیا اثر پڑتا ہے۔

$\frac{dy}{dx} = 2 (+ive)$

$\frac{dy}{dx} = -2 (-ive)$

یہاں magnitude نہیں sign کی اہمیت ہے کہ ایک change ہونے سے دوسرے پر کیا اثر پڑ رہا ہے magnitude

مطلب کسی quantity میں بڑھ رہا ہے اور sign کا مطلب کس طرف بڑھ رہا ہے۔

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

→ Derivative at a point:-

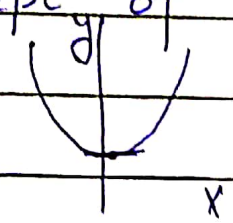
$y = x^2 + 2x$ at $x = 5$

$\frac{dy}{dx} = (2x+2)$

$= 10+2 = 12$

slope at $x = 5$ is 12

→ The concept of minima:-



for find minima

$\frac{dy}{dx} = 0$

$y = x^2$

$\frac{dy}{dx} = 2x = 0$

$2(4) = 0$

8

in 3-D

$z = x^2 + y^2$

$\frac{\partial z}{\partial x} = 0$; $\frac{\partial z}{\partial y} = 0$

L (9 - parameter)

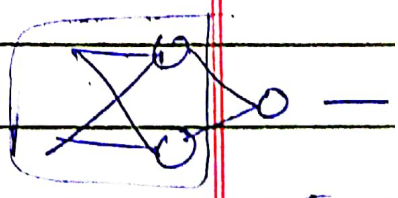
$\frac{\partial L}{\partial w_{11}} = 0$; $\frac{\partial L}{\partial w_{21}^2} = 0 \cdots \frac{\partial L}{\partial b_{21}} = 0$.

→ Back prop Initive:-

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

assumption $\eta = 1$

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



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

8 are constant

y = constant

$L = L(w, b)$

$L(b_{21})$

Date: __/__/2025 M T W T F S

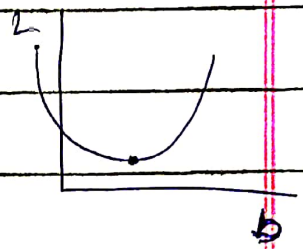
اگر $\hat{y}$ b_{21} پر منحصر ہے تو $\hat{y} = (b_{21})$

L بھی صرف اس پر ہوگا $= L(b_{21})$

$$b_{21} = b_{21} - \frac{\partial L}{\partial b_{21}}$$

Why we substract this.

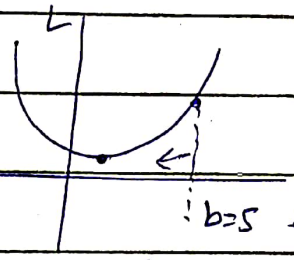
$L \propto b_{21}^2 \Rightarrow y^2 = L$



$\frac{\partial L}{\partial b_{21}}$ = L derivative with respect to b_{21}

اگر: $\frac{\partial L}{\partial b_{21}}$ = +ive

مثال کے طور پر:



تو ہم b کو کم کریں گے پر

ہماری تو آنکھیں ہیں پر

باقی آنکھیں نہیں ہیں تو ہم find slope کریں گے۔ اگر

slope جو ہے وہ +ive ہوا تو ہم b کی negative طرف جائیں گے

$\frac{\partial L}{\partial b}$ = slope | -ive of gradient $b_{21} = b_{21} \ominus \frac{\partial L}{\partial b}$

وہ مطلب میں ہی ہوتا ہے۔

— Why we need Learning Rate:-

آپ جو steps لیتے ہو - وہ ان کو smoth بناتا ہے۔

$$W_n = W_0 - \eta \frac{\partial L}{\partial b} \qquad \because \frac{\partial L}{\partial b} = -50$$

$$b = -5 - (-50) = 45$$

$$= b - \eta \frac{dL}{db}$$

$$b = -5 + (0.01 \times 50) = 0.5$$

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

What is convergence:-

اس کا مطلب ہم اپنا back prop کتنی بار چلاتے ہیں۔
اگر آپ کا نیا والا value پرانے والے value کے بہت
زیادہ close ہے تو break کر دو گے۔ کیونکہ تب
کوئی فائدہ ہو نہیں رہا ہوتا ہے۔

$$W_n = W_0 - \eta \frac{\partial L}{\partial w} \qquad \frac{\partial L}{\partial w} \simeq 0$$

$$W_n = W_0 - (0.01)(0.001)$$

$$W_n \cong W_0$$

تو بہت سے لوگ 100، 1000 بار لگا دیتے ہیں۔

Memoization in Backpropagation

What is this:-

Memoization ایک trick ہے جس میں آپ space کا خرچہ
کر کے time کو reduce کرتے ہو۔ اپنے langchain
میں vectors کو store کر لیتے ہو۔ اور بار بار ان
vectors سے result نکالتے رہتے ہو۔

In back propagation:-

ہمارے پاس کچھ derivat ایسے ہیں جو بار بار repeate
ہو رہے ہیں تو ہم انہیں کی
جگہ store کر سکتے ہیں۔ $\frac{dL}{dy}$ ، $\frac{dy}{dw_i}$
اور ضرورت ہوئی تو دوبارہ ہم انہیں نکال کر لیں گے۔

اچھا آپ نے ایک بات notice کی کہ جب batch میں ایک ساتھ سارے کے سارے rows کا y-yat calculate کر رہے ہو تو آپ dot-product کا استعمال کر رہے ہو۔ آپ loop کا استعمال کیوں نہیں کر رہے۔ کیونکہ dot کام کر دیتا ہے اور problem with big data vectorization کہتے ہیں۔

Date: __/__/2025 M T W T F S

Gradient Descent:-

→ optimization algorithem:-

① **Batch-Gradient Descent:-**

آپ epochs کے size ہونگے اور ہر بار آپ آپ پورے data کا استعمال کرونگے اور y-hat y میں سے subtract کرنے کے بعد آپ weights اور bias کو update کرونگے۔ آپ دوسری epochs میں دوبارہ جاؤنگے اور weights اور bias کو update کرونگے۔

epochs = # of updates

② **Stochastic Gradient Descent:-**

```
for i in range(epochs):
    epochs = 10
    for i in range (X.shape[0])
        ↳ random point
        y-hate → forward
        ↳ loss
        ↳ w, b, update,

    avg-loss
```

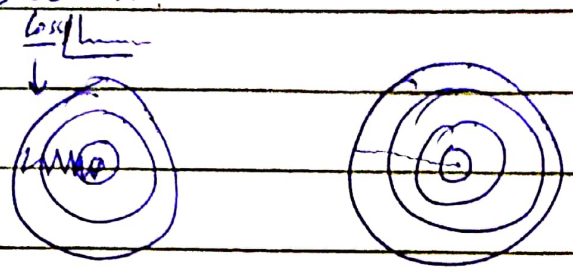
آپ پورے data کا استعمال نہ کرتے ہوئے صرف ایک point لیتے ہو اور اس پر کام کرتے ہو۔ اور یہی کام ہم دیکھ کرتے آئے ہیں۔

frequency of weight-update is higher.

Date: __/__/2025 M T W T F S

- Which is faster given same no. of epochs?
  batch-gradient descent (for time)
- Which is faster to converge (≠ same epochs).
  Stochastic (for solution) → accuracy

But in:



Stochastic — not smoth on result

Batch — Smothness

but: کبھی کبھی stochastic کی unsmothness آ سکتی ہے۔ جب local-min میں ہمارا point پھنس جائے تب۔ تو یہ ٹپک کر باہر نکل سکتا ہے۔ جبکہ batch میں ایسا نہیں ہوتا۔

Mini-batch Gradient Descent

300 rows → you make baches of 30

batch-size = 30

for in in epochs:-

for j in (length (baches)):

y-hate - loss - updates

no-of baches = no. of updates

More beneficial than stochastic and batch

Interview: Why batch-size is provided multiple of 2? (2, 4, 8, 16, 32, 64)

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

Answer:- use effective RAM کرنے کے لیے
It is faster کو code اپنے بناتے ہیں۔
an optimization technique.

— What if batch-size does not divide no. of rows properly:-

rows = 400

batch-size = 150

no. of batches = 2.66 = 3

150, 150, 100

⊖ Vanishing Gradient problems:

There are many villans in ANN and it is one of them.

یہ تب آتا ہے جب آپ ANN train کرنے جاتے ہیں۔
GD technique لگاتے ہیں۔ اور back propa. لگاتے ہیں۔
تو اگر ہمارے پاس deep NN ہے تو ہم weights کو update کرتے جائیں گے۔

$$w_{11}' = w_{11} - \eta \frac{\partial L}{\partial w_{11}}$$

تو اس کے لیے ہمیں derivative نکالنا پڑے گا۔
وہ بہت سارے derivative نکالنے پڑے گے۔

$$= \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial a_{11}} \times \frac{\partial a_{11}}{\partial w_{11}}$$

i) 0.1 × 0.01 × 0.001 = VGP
2) Deep NN → DDDD
3) Sigmoid / tanh

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

تو بھئی تو نے یہ sigmoid لگا لیا پھر نے گا۔ تو وہ
تو ۰، ۱ کے درمیان ہوں گے۔ تو یہ نہیں ہمارے
بہت چھوٹے ہوں جائے گے۔

$$\frac{dL}{\partial w_{11}} = 0.001 \times 0.1 \times 0.0001$$
$$= 0.00001$$

$$w'_{11} = w'_{11} - (0.1)(0.0001)$$
$$w'_{11} = w'_{11} - 0.0000$$
$$w'_{11} = w'_{11}$$

تو مسئلہ کیا ہے۔ تو ہمارا gradient بہت small ہو جائے گا
effectively preventing the weights from its actual value. In the worse case, this may completely stope the neural network from further training.

80s, 90s میں اسی وجہ سے deep Learn بہت دھیمے چلی گئی۔

— How to recognize?
i) Focus on loss (if no changes ≡ VGP)
ii) Weights - Graph value

— How to handle Vanishing Gradient Prob.
i) Reduce model complexity:-
مطلب آپ hidden layers کو کم کر سکتے ہو۔

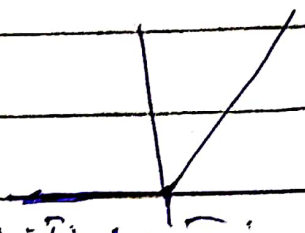
Date: __/__/2025 M T W T F S

ii) Using ReLU activation function.

formulat max(0,z)

(-1000, 1000)

(0, 1000)



iii) proper weight initialization

(iv) Batch Normalization Layer

(5) Residual Network.

— Exploiding Gradiant Decsent → RNN

opposite VGD

10, 10, 10, .. = 1000000

تو اس میں بھی same مسئلہ ہے۔

— How To Improve a Model performance

hyperparameters: Learning rate, hidden layers, neurons per layer, Optimizer, batch-size, activation function

By solving problems:-

→ vanishing / Exploiding gradient.

→ Not enough data

→ slow Training

→ Overfitting.

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

Hyperparameters:-

① No. of hidden layers:-

آپ ایک hidden ~~neuro~~ layer کے ذریعے بھی کام چلا سکتے ہو اور آپ بہت سارے add neurons کر سکتے ہو۔ پر اس سے اچھا کہ آپ زیادہ hidden layer لو اور اُس میں کم add neurons کر دو۔ کیونکہ deep Learning ہوتا وہ ایک technique استعمال کرتا ہے۔ Representation Learning۔ سب سے پہلے پہلی layer میں آپ edges لیتے ہو اور پھر اس edges کو ملا کر تھوڑی shape ~~the~~ بناتے ہیں اور آخری hidden layer پوری shape بناتے ہو۔ تو اس لیے آپ زیادہ hidden کا استعمال کرتے ہو۔

Overfitting

جہاں overfitting ہونے لگے وہاں stop ہو جانا چاہیے۔

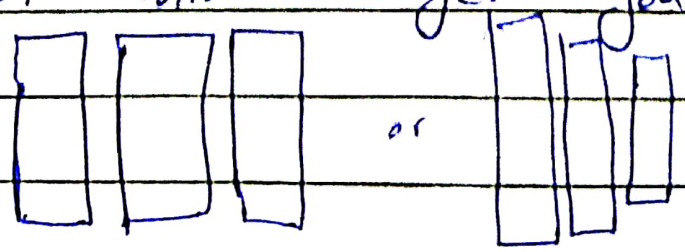
Transfer Learning : اس بھی مدد دیتا ہے

② Neuron / layer

Input = shape of your data columns

Output = how many classes in output.

In hidden layer you either



note

اتنے پس ہو آپ کی requirement کے sufficient ہوں

Date: __/__/2025 M T W T F S

③ Batch_size:

Smaller = 8-32 - ~~Trainis~~ Result better

larger = ≥32 = Trainy fast

Somaone sayes you either do:

learning_rate scheduler

پہلے وارمس کو کم اور پھر بڑھاتے جاؤ اور اگلے وارمس کو

called warning up lr. میں learning-rate کو بڑھاتے جاؤ

training start → learning-rate lower

than after you increase lr.

its fast and accurate:

④ Epochs:-

you can use early_stopping

The next Tutorial is also.

— Problems in Training

① Vanishing / Exploiding GD

i) Weight initializing (different values)

ii) activation function change

iii) Batch Normalization

iv) Gradient clipping

② Not enough data:-

i) Transfer learning

ii) Unsupervised pretraining

Date: __/__/2025 M T W T F S

(3) Slow Training
i) optimizers
ii) learning rate schedular

(4) Overfitting
i) l1 and l2 regularization
ii) Dropouts

— Early Stopping

اس کا مطلب آپ کتنی epochs کو Train کرو گے۔
تو اس میں ہم کیا کرتے ہیں جب بھی overfitting ہو رہی
ہوتی ہے دیس پر stop کر دیتے ہیں۔
اس کا مطلب ایک point کے بعد آپ کا فائدہ نہیں
نقصان ہو رہا ہے، loss آپ بڑھنے لگ رہا ہے۔
اس کے لئے ہم callbacks کا استعمال کریں گے۔

```
callbacks = EarlyStopping(
    monitor = 'valloss' ,  * which for you stope
    patience = 20 ,  * how many epochs you waight
    min_delta = 0.0001
    monitor = AUTO)
```

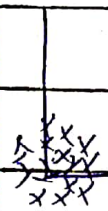
— Normalizing Inputs Data:-
For fast Training
آپ کا input اوپر نیچا ہے۔
کچھ ترازو میں سے کچھ اوپر نیچے ہے۔
loss-function = non-symmetrical

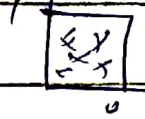
When To Normalize and when Standardize:-

اس میں آپ کو پتہ نہیں کہ زیادہ سے زیادہ value کتنی ہو سکتی ہے تو آپ
like price
Normalize کرتے ہیں - اور gpa یا میں آپ کا پتہ ہے کہ ہماری value 0-10 کے درمیان ہوگی -
تو آپ standard کرتے ہیں -

Date: __/__/2025 M T W T F S

Standard	Normalize
$\frac{x_i - \mu}{\sigma}$	$\frac{x_i - x_{min}}{x_{max} - x_{min}}$
$(-1, +1)$	same scale $(0, 1)$





Dropouts Layers

When overfitting we use dropouts

Solution of Overfitting:-

1. Add more data:
2. Reduce Complexity
3. Early stopping
4. Regularization → L1, → L2
5. Dropouts

For every epochs, we randomly switch off some of the nodes in input layer and output layer

اس کا مطلب کون سا node رہے گا اور کون سا node نہیں
رہے گا جب آپ دس epoch چلائیں گے تو آپ 10
different NN پر اپنے train data کر رہے ہوں گے

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

مثال کے طور پر آپ کے NN میں ایک particular node پر بہت depend کر رہا ہے تو dropout کے استعمال سے اگلی بار وہ نہیں ہوگا۔ اس کے بدلے کچھ اور ہوگا۔ تو ہمارا NN اب اس پر بھی برابر کا depend کرے گا۔

Random Forest Analogy:-

random-forest میں آپ trees بناتے ہیں randomly features کو select کیا پھر اگلے پانچ کو select کیا پھر بیچ والے پانچ کو select کیا اور ہر سے output نکلوا رہے ہیں اگر آپ نے random forest میں 100 trees لگائے اور سو میں سے 60 بار 1 آیا ہے اور 40 بار 0 آیا ہے تو آپ majority والے کو select کریں گے۔

اسی طرح dropout میں آپ الگ الگ NN کو train کر رہے اور اس کو ملا کر آپ result نکلوا رہے ہیں۔

پھر Testing میں سارے active nodes رہیں گے۔

$w = (1 - \text{dropout})$

weight test $= 1 - 0.25 = 0.75$

Practical Tips:-

① Overfitting dropout ↑, Underfitting dropout ↓

② سارے NN Architecture کے last-layer پر dropout لگائیں

(i)	CNN	average	40 – 50 ← (P)
(ii)	RNN		20 – 30
(iii)	ANN		10 – 50

Date: ___/___/2025 M T W T F S

Drawbacks :-

training slow

loss function vary karta rehta hy.

آگے پیچھے رہتا ہے اور debug کرنے میں مشکل ہوتی ہے۔

Regularization:-

- For Overfitting

complex NN سے بچنے کے لیے ایک کام تو یہ ہے کہ اپنے overfitting کے کچھ weights کو 0 کے برابر کر دو۔ اسے ہی regulari کہتے ہیں

weights $\cong 0$

- Ways To Solve Overfitting

i) adding more data (Data Augumentation)

ii) reduce complexity of your data.

(Dropout, Early Stopping, Regularization)

In Regularization (L1, L2 (99% chances), L1 & L2)

L2 :- ANN → weights / bias → purpose → min loss

$$C = \underbrace{\frac{1}{n}\sum_{i=1} L(y_i - \hat{y}_i)^2}_{L} + \text{penality}$$

Regularization

$$C = L + \underbrace{\frac{\lambda}{2n}\sum_{i=1} \|W_i\|^2}$$

(اور یہ کیا کام کرتا ہے آپ کے weights کو تقریباً 0 کر دیتا ہے۔

اور یہی ہم چاہتے ہیں۔)

λ is hyperparameter

L2: weights جو ہیں وہ zero نہیں ہوتے zero کے آس پاس رہتے ہیں
L1: میں ہو جاتے ہیں

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

$$L1 :- \quad L + \frac{\lambda}{2n} \sum_{i=1}^{n} \|w_i\|$$

you can't square.

ہم bias پر square نہیں لگاتے۔

— Intuition behind Regularization:

Now Loss function:

$$L' = L + \frac{\lambda}{2} \sum \|w_i\|^2$$

Now differentiate $w_1 = \frac{dL}{dw_1}$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial w_0} + \lambda w_0$$

$$w_{new} = w_{old} - \eta \lambda w_0 - \eta \frac{\partial L}{\partial w_0}$$

$$= (1 - \eta \lambda) w_0 - \eta \frac{\partial L}{\partial w_0}$$

L2 - regularization = weight - decay - kay.

— What are Activation function?

ANN میں ہر neuron اپنے input کا weighted sum بناتا ہے
اور scalar value کو function کے اندر پاس کر دیتا ہے۔
function کا کام بتانا ہے کہ neuron کتنا active ہونا ہے۔

Why we need

اس کا استعمال ہم non-linearity کو capture کرنے کے لیے کرتے ہیں۔
اگر ہم نہ کریں تو وہ linear کی طرح treat کرے گا۔
وہ LR یا perform logistic Reg کرے گا۔
سیدھی line کھینچے گا۔

Date: __/__/2025 M T W T F S

Ideal Activation Function

i) Non-linear $y = f(x)$ → ہونا چاہیے non-linear activation

ii) Activation function must be differentiable for Gradient Descent.

iii) Computationaly inexpensive:
- چاہیے fast, simple, calculate کا derivative

iv) Zero centered:- Normalized.

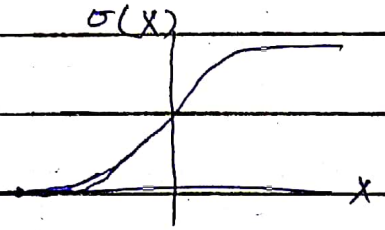
v) Non-saturating = input کو دبانا نہیں Vanishing problem

Sigmoid → $\boxed{0,1}$
tanh → $(-1,1)$
→ saturating

Non-saturating → non-saturating

(1) Sigmoid Activation Function

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



Advantage:-

→ output in [1,0] in range.

→ we Treat as probability

→ Non-linear function mean capture non-linearity.

→ differential, (Gradient Descent) To calculate derivatives).

Date: __/__/2025 M T W T F S

Disadvantage:-

1) saturating function means V-G-P

اگر آپ کا X میں value بہت بڑا ہے تو $\frac{dz}{dw}$ وہ تقریباً 0 کے برابر آتا ہے۔ ساری value کھو کے 0 کے لگ بھگ آ جائے گی۔ جس سے ہمارا model stop کر جائے گا۔

$$W_{new} = W_{old} - 0.0001$$

∵ That's why we didnot use sigmoid in hidden layer

2) Non-zero centered

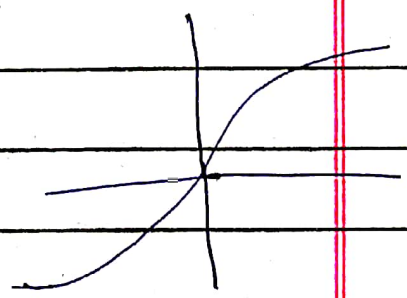
تو آپ بہت restricted طریقے سے minimize/calculate loss کرتے اور آپ کا training time بڑھ جاتا ہے۔

It's take time اور convergence نہیں کر پاتے۔

3) $z = \frac{1}{1+e^{-z}}$ = Computation

(2) Tan-h [-1, 1] It's similar to tanh

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



Derivative:

$$f(x) = (1 - \tanh^2(x))$$

→ Advantage:-

Non-linear, Differentiable

zero-centered (training faster.

→ Disadvantage:-

→ saturating function (Vanishing GP)

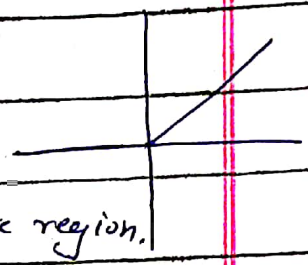
→ computationally expensive

Date: __/__/2025 M T W T F S

4 Relu Function

Its mostly use in hidden layer

$f(x) = \max(0, x)$



— Advantage:-

i) Non-linear

ii) Not saturated in the +ive region.

iii) Computationally inexpensive.

iv) Convergence faster

— Disadvantage:-

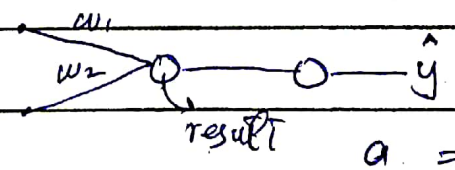
1. not zero-centered

2. not differentiable.

— Dying Relu Problem:-

جب آپ relu لگاتے ہیں۔ وہ کچھ neuron کو output 0 دیتا ہے اسے dead neuron بنا دیتے ہیں۔ مطلب وہ کام ہی نہیں کر رہے۔ اگر 50% سے زیادہ dead neuron ہو جائیں تو آپ اپنے data میں کچھ pattern کو دھونڈ نہیں پاتے۔

Why Dying Relu:-



$a = \max(0, z)$

$z_1 = w_1 x_1 + w_2 x_2 + b_1$

اگر اس کا deriv 0 سے چھوٹا ہو

$\max(0, z_1) = 0$

=

45 سارا سارا کام z کے -ive ہونے پر ہو رہا ہے۔ اگر -ive ہوا تو کمال کام غلط ہو جائے گا۔ کیونکہ relu 0 کر دے گا۔ اور weight update نہیں ہوگے

Date: __/__/2025 (M)(T)(W)(T)(F)(S)

تو پھر ہمارے weights update نہیں ہوں گے اور neuron کچھ نہیں سیکھے گا۔

- When $z = w_1x_1 + w_2x_2 + b_1 < 0$

i) when Lr (learning rate) is high

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$$

$$w_1 = w_1 - (0.5)(-0.5)$$

$$w_1 = w_1 + 1$$

تو یہ بہت بڑا ہو جائے گا اور اگلے cycle میں z = w1x1 -ive ہو سکتا ہے

ii) High -ive bias

$$z = w_1x_1 + w_2x_2 + b$$

- How To Solve

→ set low learning rate.

→ set +ive value To bias → 0.01

→ Don't use relu instead use variants

Variants → linear variants → leaky relu, Parametric relu

Variants → non-linear variants → selu, Elu

Leaky Relu:-

$$f(x) = \max(0.012, z)$$

$$z < 0 \; ; \; \frac{1}{100} \times z$$

$$-ive\ (z) \rightarrow \neq 0 \quad \because 0.012$$

Not dying relu problem.

Advantages:-

i) Close To 0 centered

ii) No-dying problem

Date: __/__/2025 M T W T F S

- Parameteric relu:-

$$f(x) = \begin{cases} x & \text{if } x>0 \\ ax & \text{otherwise} \end{cases}$$

$f(x) = (ax, \cancel{x})$

→ trainable parameter → from Training

- Elu - Exponential Linear Unit:-

$$ELU(x) = \begin{cases} z & \text{if } x>0 \\ a(e^{z}-1) & \text{if } z<0 \end{cases}$$

constant ←

Advantage:-

→ close zero centered (convergence faster)

→ always continues as well differentiable.

→ Better X-data performance.

→ dying problem solved.

Diadvantage :- Computational expensive.

- Selu :- $SELU(z) = \begin{cases} z & \text{if } z>0 \\ ae^{z}-a & \text{if } z \leq 0 \end{cases}$

- Why Weight Initializing is important?

اگر آپ کو weights کو غلط طریقے سے initialize کریں تو:

→ vanishing GD.

→ Exploiding GD

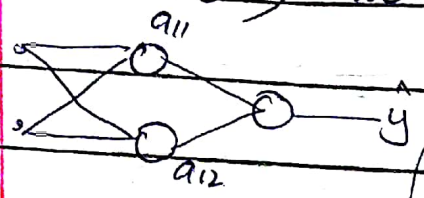
→ Slow converage

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

Wrong Weights Initiazation Technique:-

Case 1:- Zero Initialization:-

اگر ہم weights کو initialize کر دیں 0 سے



$a_{11} = max(0, z_{11})$

$z_{11} = \underbrace{w_{11}^{1}}_{=0} x_1 + \underbrace{w_{21}^{1}}_{=0} x_2 + b_{11} = 0$

- For Relu

$a_{11} = max(0, 0)$

$a_{11} = 0$

$a_{12} = 0$

$a_{11} = a_{12}$

Vanishing GD

اور عام طور پر update weights کی نہیں ہوتی

- For Tanh $= \frac{\overset{=0}{e^{x}} - \overset{=0}{e^{-x}}}{e^{x} + e^{-x}} = \frac{0}{0} = 0$

اس کے ساتھ بھی same prob. ہوگی۔

- For Sigmoid:-

اس میں تھوڑا سے مختلف ہے کہ اگر 10000 neuron add کر دو تو بھی یہ single neuron کی طرح treat کرے گا

For Second neuro:

$a_{21} = \sigma(z_{21}) = 0.5$

$a_{21} = 0.5$

$a_{21} = a_{22}$

for first Neuron:

$a_{11} = \sigma(z_{11}) = 0.5$

$a_{11} = 0.5$

$a_{11} = a_{12}$

in-short capture of non linearity نہیں کر پاتا

اگر آپ نے سارے weights = 0 کیے اور hidden sigmoid

تو سارے neurons = 1 perceptron

this not capture Non-linearity

Date: __/__/2025 M T W T F S

Case 2: Non-Zero ~~Ratio~~ values but constant

$w_1 = w_2 = b = 0.5$

اس سے بھی مسئلہ ایک جیسا ہے پہلے والے

node کے weight ایک جیسے ہونگے اور دوسرے والے node کے weight بھی

ایک جیسے ہونگے اور non linear feature پیدا نہیں کر پائیں گے۔

Case 3: Random Initialization:-

i) Small Random values:- → (-0.01, 0.01)

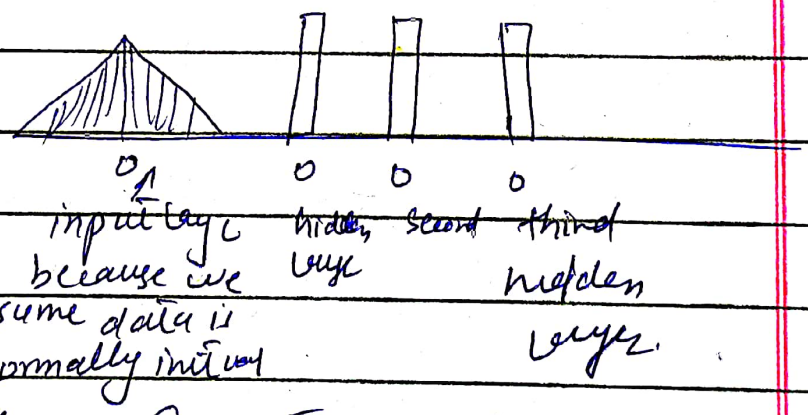
اگر آپ نے weight کی values بہت small رکھی ہیں

0.01 like تو backpropagation میں جب ان کے results

ایک دوسرے سے multiply ہوتے ہیں وہ almost equal

zero کو کر دیتا ہے یعنی weight update نہیں ہوتے

اور zero کے قریب ہی رہتے ہیں۔ اس کا graph



Vanishing Gradient Descent problem comes.

tanh relu کے ساتھ vanishing gradient بہت strong ہوگی پھر

sigmoid سے اتنی strong نہیں ہوگی۔ پھر ہوگی ضرور۔

range (-3, 3)

Date: __/__/2025 M T W T F S

Random Initialization (Large number)

slow traing

vanishing gradient Descent

تو اس سے derivat کی values بہت چھوٹی آئیں گی۔

تو saturation ہو گا slow conv ہو گا۔

In Sommary

zero X , non-zero-sam-value X

small random X , Large random X.

(تو ہمیں کیا کرنا چاہیے۔)

کچھ ایسی Technique آ چکی ہیں۔

Xavier/Glorot Initialization → Normal, → Uniform

Hei → Normal, → Uniform

ہمیں اپنے data کے حساب سے weight کو initize کرنا ہے وہ ہونے بھی random range چاہیے۔ ایک ایسی جس سے ہم gradient problem سے بچ سکیں۔

For only Xavier (tanh) use

For Hei (relu) Use

Xavier (Normal) $= \sqrt{\frac{1}{\text{fan-in}}}$

fan-in = no. of inputs coming to node.

Batch Normalization :-

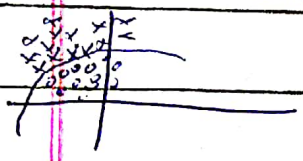
Batch normalization is an algorithem method which makes the training of Deep Neural Network faster and more stable.

Date: __/__/2025 M T W T F S

یہ کیا کرتا ہے آپ کے result کو normalize کر دیتا
ہے یعنی a_{11} کو mean = 0 کر دے گا std = 1 کر دے گا
اور ایسا کیا جاتا ہے کہ ایسا کرنے سے ہمارا DNN faster
بن جاتا ہے۔

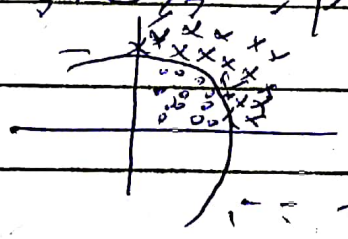
تو موگو ون نے کیا سوچا کہ اگر input کو scale کرنے سے
fast ہو سکتا ہے تو neuron کے input کو بھی scale کرنے
سے اور بھی fast ہو جائے گا۔

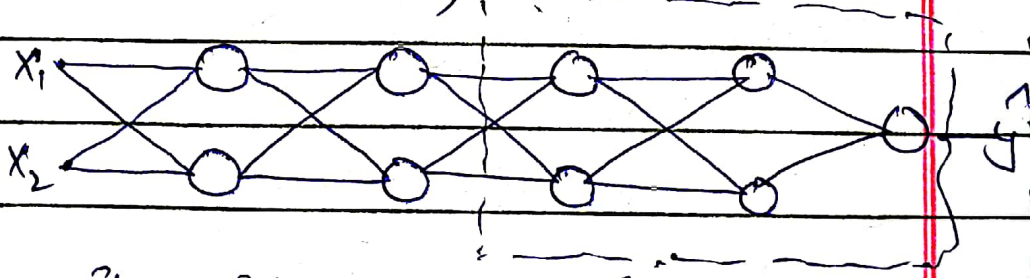
Interval Coverite Shift:-



Train-df آپ کا data ایسا ہے۔
اور model نے train کیا ہے

اور اگر test-df میں آپ کے data distribution مختلف ہے۔ مگر
model نے predict کیا تو پھر بھی آپ کو Retrain کرنا پڑے گا۔





اگر آپ پہلی تک neural Net لیتے ہیں تو اس میں جو
input آئیں گے وہ actual input سے بہت مختلف ہوں گے
تو آپ ہر neuron کے input کو gaussian dist میں convert کریں گے
$\mu = 0, \sigma = 1$
تو اب آپ یہ pass کریں گے۔

آپ اپنی choice کے مطابق کسی layer پر normalization کر سکتے ہیں یا آپ کی مرضی ہوتی ہے۔ اس میں ہر neuron پر individually کام کرتے ہو تو ہر neuron کا ایک رنگ γ, β کی value ہوتی ہے۔

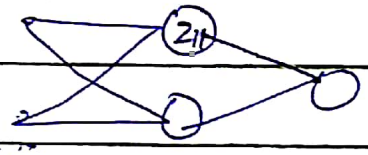
Date: __/__/2025 (M T W T F S)

Batch Normalization: The HOW

Steps: i) Mini-Batch GD

ii) layer-by-layer (مطلب اپنی layer پر apply کرنا)

اگر آپ کے پاس



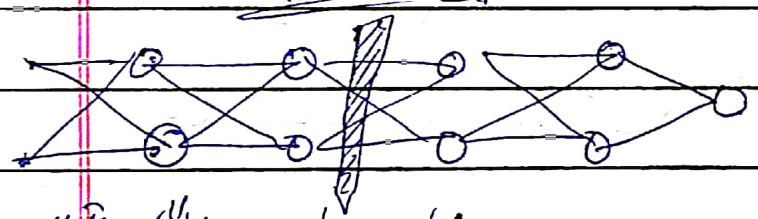
$$z_{11}^{BN} = \gamma z_{11}^{N} + \beta$$

z_{11} → calculated sum → find activation → normalize it (z_{11}^{N}) → multiply with γ and add with β → activation → a_{11}

$$z_{11} \rightarrow z_{11}^{N} \rightarrow z_{11}^{BN} \rightarrow g(z_{11}^{BN}) = a_{11}$$

یہ ساری چیز آپ اپنے سارے neuron کے ساتھ کریں گے

$$\cancel{z_{11} \rightarrow z_{11}^{N} \rightarrow z_{11}^{BN}}$$



یہاں پر اپنی layer لگا دیتے ہوئے

اس میں آپ γ اور β کی value بھی update کرو گے

$$\gamma = \gamma - \eta \frac{\partial L}{\partial \gamma}$$

On Test Data:-

آپ کے پاس تو ایک data ہے اور اس کی μ اور σ آپ کے پاس نہیں ہے اور batch-norm کے لیے اس کی ضرورت ہوتی ہے تو آپ ہمیں training کے batch سے نکالیں گے σ اور μ سے لیتے ہیں۔

For all batches: $\mu_{B1}, \mu_{B2}, \mu_{B2}, \mu_{B3}, \mu_{BN} = \mu$

$\sigma_{B1}, \sigma_{B2} \ldots \sigma_{B125} \ldots \sigma$

Date: __/__/2025 M T W T F S

Advantage:-

① stable → hyperparameter more range of value.

2) faster

③ Regularization (Sum effect کم کرتا ہے)

Some Over fiting reduce

④ ~~initialization weight~~ [illegible]

Weight - Initialization rejective Impact reduce.

Optimizer

performance of NN

↳ Fast Training

↳ Weight Initialization

↳ Batch Normalization.

Gradient Descent is also an optimizer

So why we need more optimizer.

Challenge with GD:-

① learning rate value decide

very small lr ↓ very slow

very large lr ↑ To steps away from minima.

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

Solution:- Learning-rate-schedule:-

3) Local minima problem:-

5) Saddle point:-

Exponentially Weighted Moving Average
EWMA

is a Technique in which you find trends in time-series data.

i) Time-series forcasting
ii) Singnal processing
iii) Company financial forcasting
iv) Deep Learning in Optimizers.

اس میں آپ بیچ میں آنے والے point کو زیادہ value دیتے ہو پہلے والے point سے۔

D1
D2 . less weight
D3
D4 . more weigtig

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

V_t ↓ current date

V_{t-1} ↓ less then one day

θ_t ↓ current data

$V_0 = 0$, $V_0 = \theta_0 = 2.5$

$V_1 = (0.9)(0) + (1-0.9)(1.3)$

$= 0 + 0.5$

What is the impact of β.

D1	2.5
D2	1.3
D3	1.7
D4	3.1
D5	4.3

Date: __/__/2025

اگر آپ نے $\beta = 0.9$ رکھی ہے تو اس کا مطلب ہے کہ

$$\frac{1}{1-\beta} = \frac{1}{1-0.9} = \frac{1}{0.1} = 10$$

اس line کا یا اس point کا جو value ہے وہ پچھلے دس دنوں کے data کے mean کے برابر ہے۔

Mathematical Intuition:-

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

$$V_0 = 0$$

$$V_1 = (1-\beta)(\theta_1)$$

$$V_2 = \beta V_1 + (1-\beta)\theta_2$$

$$V_2 = \beta(1-\beta)\theta_1 + (1-\beta)\theta_2$$

$$V_3 = \beta V_2 + (1-\beta)\theta_3$$

$$= \beta(\beta(1-\beta)\theta_1 + (1-\beta)\theta_2) + (1-\beta)\theta_3$$

$$= (1-\beta)\left[\beta^2\theta_1 + \beta^1\theta_2 + \theta_3\right]$$

$$\beta^2 < \beta_1 < \beta \qquad 0 < \beta < 1$$

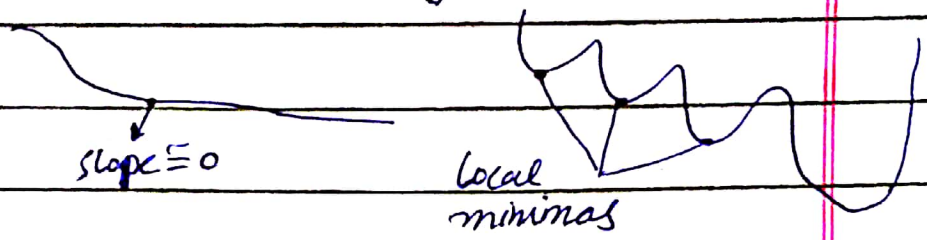
اس طریقے سے formula design ہے کہ وہ پرانی values کو weightage دیتا ہے اور نئی values کو زیادہ weightage دیتا ہے۔

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

Momentum Optimization - The Way:-

Non-Convex Optimizers:- جب ایسا seen ہو جائے تب

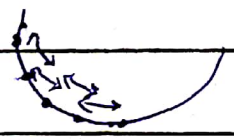
- higher curvature
- Consistent gradient
- Noisy gradient (local minima)



Momentum Optimization - The What?

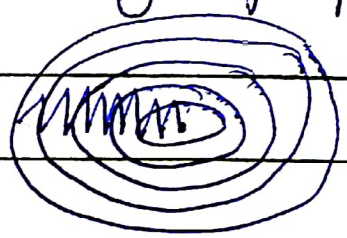
اگر A سے B کی طرف آپ نے جانا ہے اور آپ کو
رستہ معلوم نہیں ہے تو آپ نے چار بندوں سے پوچھا
تو انہوں نے آگے کی طرف point کر دیا۔ تو اب آپ confident
ہو گئے speed بڑھا دیں گے۔ اور یہی کام یہاں بھی ہوتا ہے۔
آپ پچھلے steps کو check کرتے ہو کر وہ کتنی speed سے
آئے تو اگلا step بھی اتنا ہی بڑا ہوگا۔

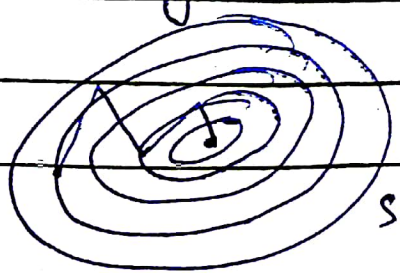
B رفتار کا مطلب ہے جتنی زیادہ value اتنی زیادہ speed



γ is speed
99% it is faster than other optimizers.

EWMA کی مدد سے آپ پچھلے steps کو یعنی میری as history of past velocity as a momentum





SGD momentum

یہ اتنی speed میں ہوتا کہ local minima میں پھنستا نہیں ہے اور
نیچے گر کر باہر نکل جاتا ہے اور global میں چلا جاتا ہے۔

Date: __/__/2025 It is faster

Effect of beta : in case of momentum with EWMA

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

$V_t = \beta V_{t-1} + \eta \nabla w_t$

beta کو ہم β کہیں گے جو کہ decaying factor ہے
decay سے speed کتنی velocity کی past
ہی جائے گی کہ
ہو رہی ہیں۔

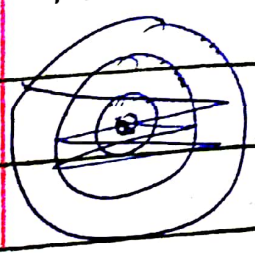
β کی value 0 کرنے سے وہ SGD بن جائے گا۔
اور β کی value 1 کرنے سے fraction نہیں ہوگی۔
اسی لیے ہم β کی value 0.95 رکھتے ہیں

Problem :-
momentum کا سب سے بڑا problem یہ ہے۔
کیونکہ جب global میں جاتا ہے تو تب بھی یہ
پھسلتا ہے مگر پھر رک جاتا ہے۔

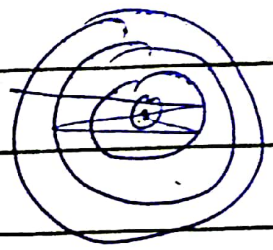
Nesterov Accelerated Gradient (NAG)

Updated Version of Momentum :-
اسی momentum technique کو implement کرنے کا طریقہ ہے۔
بس اس میں ایک improvement ہوتا ہے کہ آپ کے acceleration
کو آپ dump کر دیتے ہو۔ reduce کر دیتے ہو۔

Momentum



NAG



Optimizer

It is an algorithm جس میں آپ weights کی optimum value معلوم کرتے ہو

جس کے لیے minimum loss ہوتا ہے۔

Date: __/__/2025 M T W T F S

Momentum میں آپ کیا کرتے ہو آپ جس جگہ پر ہو

اسی جگہ کا gradient بھی اور momentum بھی۔

NAG میں آپ lookahead find کرتے ہو apply momentum

کر کے اور پھر جس جگہ پہنچے آپ اس جگہ پر آپ

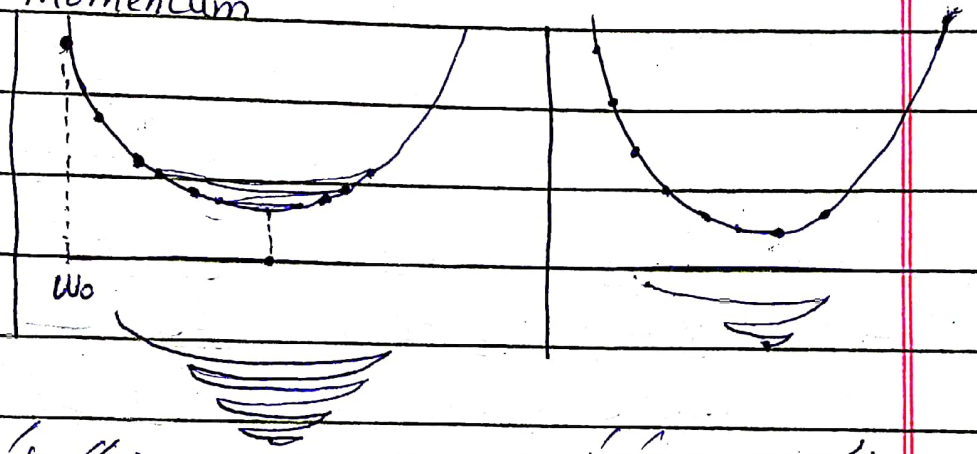
gradient معلوم کرتے ہو۔

Geometric Intuition:-

force gradient کرتا ہے جب آپ نے apply momentum

NAG میں کرنا ہوتا

Momentum



اس میں ہوتا کیا کہ NAG میں آپ minima سے آگے جا کر

gradient معلوم کرتے ہو تو وہ اب الٹا ہو جاتا ہے جس کی وجہ

سے آپ minima کی طرف move کرتے ہو۔

tf.keras.optimizers.SGD(decay-factor

learning-rate = 0.01, momentum = 0.9, nesterov = True).

ADAGrad

اس میں آپ learning-rate کو fix نہیں رکھتے بلکہ

adapt کرتے ہو۔

When to use:-

i) input-feature not scaled

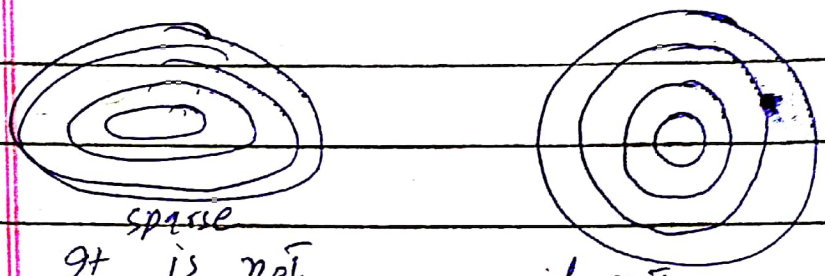
cgpa	salary

Date: __/__/2025 M T W T F S

ii) When features are sparse most of values are zero.

↳ Problem with sparse features:-

↳ Elongated bowl problem



sparse
It is not circular

if not sparse
it is circular.

اس میں کیا ہوتا ہے۔ بہت ساری values 0 ہونے کی وجہ سے $\frac{dy}{dx}$ 0 ہو جاتا ہے اور سب کچھ 0 ہونے کی وجہ سے negative کی طرف more کرنا شروع کر دیتا ہے۔

$$w = w - \eta \boxed{\frac{\partial L}{\partial w}} = 0$$

Adagrade mathematics

تو ہم کیا کریں گے کریں learning r کی different value کے لئے الگ الگ parameter کے لیے۔

جس کا gradient بڑا ہوگا اس کا small r کر دینا ہے۔ جس کا gradient چھوٹا ہوگا۔ r بڑا ہوگا۔

$$w_{t+1} = w_t - \frac{\eta \nabla w_t}{\sqrt{v_t + \varepsilon}}$$

Disadvantage:-

~~solution کے پاس تک تو پہنچتا ہے پر solution کے قریب کبھی نہیں پہنچ سکتا۔~~

You can't reach global minima.

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

RMSProp → Root mean squared propagation

It is an improvement over Adagrade

یہ Adagrad استعمال کرتے ہوئے جب sparse ہوتا ہے

اس سے آگے minima تک نہیں پہنچ پاتا کیونکہ

b کی value زیادہ ہوتی ہے almost 1 کے برابر اور

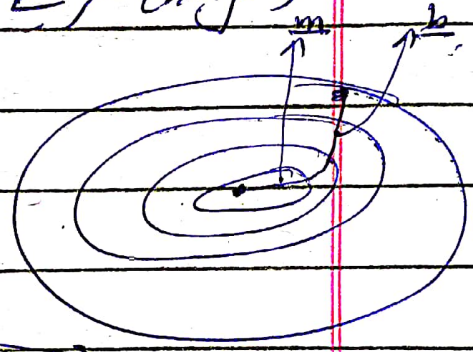
m کی value تقریباً zero ہو گی تو وہ پہلے b

کو change کرے گا اور اس کی نیچے کی طرف move ہو گا۔

AdaBoost :-

$$V_t = V_{t-1} + (\Delta w_t)^2$$

$$= w_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \nabla w_t$$



آگے past کے gradient کو calculate کرتا ہے اور

value بہت بڑی ہو جاتی ہے اور update/change

zero کے برابر ہو جاتا ہے۔

RMS Prop :-

$$V_t = \beta V_{t-1} + (1-\beta)(\Delta w_t)^2$$

$\because \beta = 0.95$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{x_t + \epsilon}} \nabla w_t$$

Disadvantage:-

This is no disadvantage. It is the competative of adam.

Adagrad work well in linear data.

RMS Prop has reach the global minima

Date: ___/___/2025 M T W T F S

Adam Optimizer:- → most popular.

Adoptive Moment Estimation

Last optimation:-

optimizers. ~~SGD | BGD | MGD~~	description
SGD | BGD | MGD	Very slow to reach
Momentum	It is fast اور کچھ آگے کی طرف جاتی ہے
NAG	یہ آگے پیچھے تھوڑا ہوتا ہے اور sparse data پر کام نہیں کرتا۔
Adgrad	یہ sparse پر کام کرتا ہے جب ملا جائے پیچھے آگے پیچھے
RMsgrad	یہ ان کا problems کو solve کرتا ہے۔

Ideas:-

1) Momentum ② Learning decay

merge → Adam — کیوں نہ ہم ان دونوں پر کام کریں۔

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} * m_t$$

$$\boxed{\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla w_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla w_t)^2 \end{aligned}}$$

momentum

Adagrad

$\beta_1 = 0.9$, $\beta_2 = 0.99$

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

- Convolutional Neural Network

It is highly inspired by our visual cortex.

Why not use ANN?

Problems:-

i) High Computation Cost

ii) Overfitting

iii) Loss of imp info like spacial arrangements of pixels.

- CNN Intuition:-

پہلے filters کو بتایا جائے گا کہ edges ڈھونڈے گا دوسرے میں corners اور تیسرے میں circles اور چوتھے میں پورا object

Applications:-

i) Image Classification

ii) Faces recognition and detection

iii) Object Detection

iv) Image Segmentation

v) pose activation

- CNN vs Visual Cortex:-

- Introduction:-

- پہلی layer والے فنڈ کرتے ہیں edges

premitive features

Date: ___/___/2025 (M)(T)(W)(T)(F)(S)

gray scale image 1 channels

RGB has 3 channels.

Convolution Operation

CNN → convolution layer
→ Pooling
→ Fully Connected layer.

starting layers permitive features کے لئے ہیں

How To detect Egde Detection:-

horizontal edge کو detect کرنے کے لئے

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255

find

(*) Convolution operator.

-1	-1	-1
0	0	0
1	1	1

filter (kernel) matrix [3x3]

=

Feature map

0	0	0	0
255	255	255	255
255	255	255	255
0	0	0	0

- اب آپ یہاں کیا کرو گے اور filter کو multiply کرو گے اور پھر کیا کرو گے sum کرو گے اور آگے جا کر لکھو گے اور پھر یہ result آگے move کرو گے
- horizontal edge کے لئے example کے لئے ہے
- اگر آپ feature کو change کرو گے تو vertical edge بھی

-1	0	1
-1	0	1
0	0	1

e.g. سامنے آ سکتے ہیں

آپ کو features کے اندر کی value بنانے کی ضرورت نہیں ہے۔

وہ آپ CNN پر چھوڑ دیتے ہیں اور random value دیتے ہیں اور

like weights in ANN backpropag میں وہ خود بخود ٹھیک کر لیتا ہے

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

سوال: جب آپ 28×28 پر 3×3 feature لگاتے ہیں تو feature map کتنے بنے گا۔

26×26

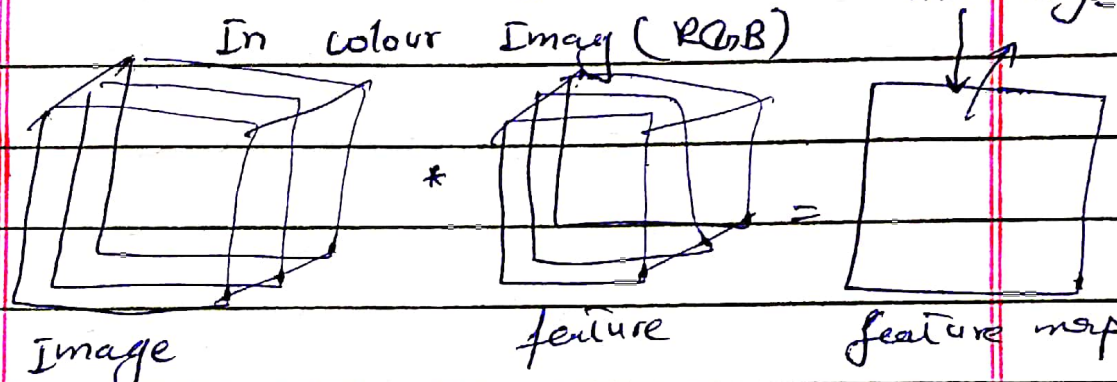
image shape , feature shape
n×n m×m

$= (n-m+1) * (n-m+1)$

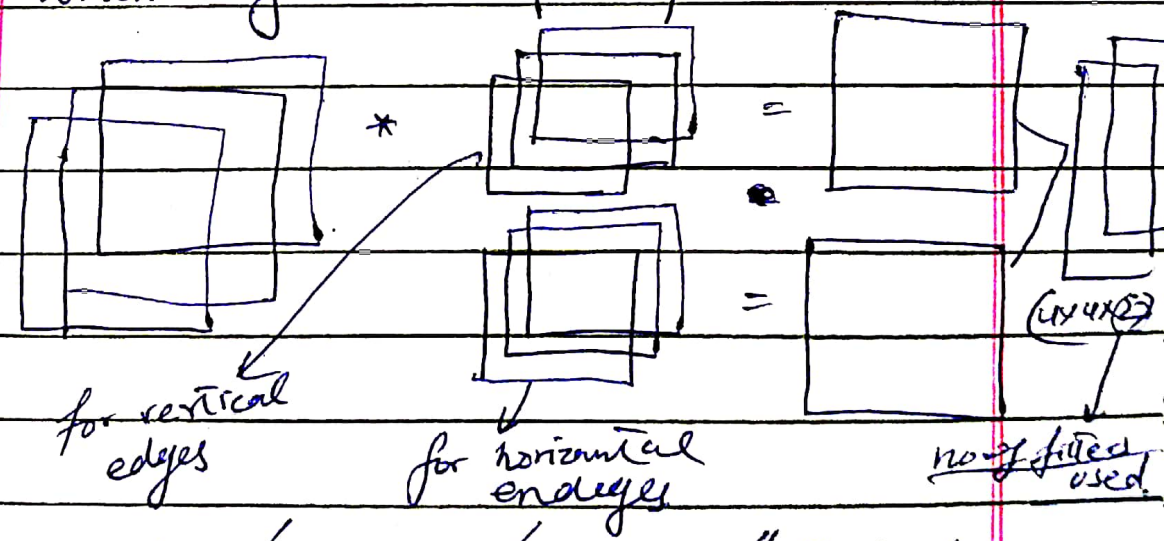
$= (28-3+1) \times (28-3+1)$

$= 26 \times 26$

In colour Image (RGB)



When you multiple features?



اب ہم ان filter map کو بھی merge کر دیتے ہیں۔
اور انہیں ایک volume کی طرح treate کر سکتے ہیں
جو آگے کی layer کے لیے input کا کام کرے گا۔

Date: __/__/2025

M T W T F S

Padding & Stride:-

(i) جب آپ Convolutional operation لگاتے ہیں تو ہر feature کے لیے آپ کو image کچھ نہ کچھ ترجانا پڑے اور آپ information loss کر سکتے ہیں۔

(ii) آپ کے corner والے pixels میں وہ کم part کو feature میں شامل ہیں center والے pixels کے مقابلے میں۔ تو ان کی درمیان والے pixels کو زیادہ importance دیں گے۔

اس میں ہم کیا کرتے ہیں ہم اپنے reduce size کو feature-map نہیں کرنا چاہتے۔ تو ہم کیا کرتے ہیں۔ ہم feature کو change نہیں کرتے بلکہ ہم image کو change کرتے ہیں۔ اس میں ایک ایک column اور row add کر دیتے ہیں۔ اوپر نیچے سے۔ اور padding کی values 0 ہوتی ہیں۔

New formula for feature-map

$$(n+2p-f+1) = (5+2(1)-3+1)$$

image size, padding, feature

$$= 5+2-3+1 = 5$$

Strides:-

عام طور پر stride (1,1) ہوتا ہے مطلب جب feature apply کرتے ہیں تو ایک ایک کر کے آگے move کرتے ہیں۔ پر آپ 2-2 کے آگے move کروا سکتے ہیں اور نیچے بھی دو دو کے move ہوگا۔

stride (2,2)

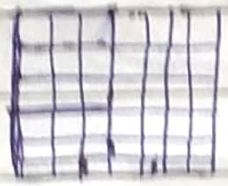
reduce size کو feature-map ہو جائے گا۔

$$\left\lceil \frac{n-f}{s}+1 \right\rceil$$

Date: ___/___/2025 (M)(T)(W)(T)(F)(S)

$$\frac{n+2p-f}{s}+1$$ if padding hai

اگر input کے پاس 1-p special case ہو تو
7×6 size کا filter map 3×3



first for row

$$\left\lceil \frac{n-f}{s}+1 \right\rceil = \left\lceil \frac{7-3}{2}+1 \right\rceil = 3$$

$$\left\lceil \frac{6-3}{2}+1 \right\rceil = 2.5 = (3\times 2)$$

Why we need strides:-

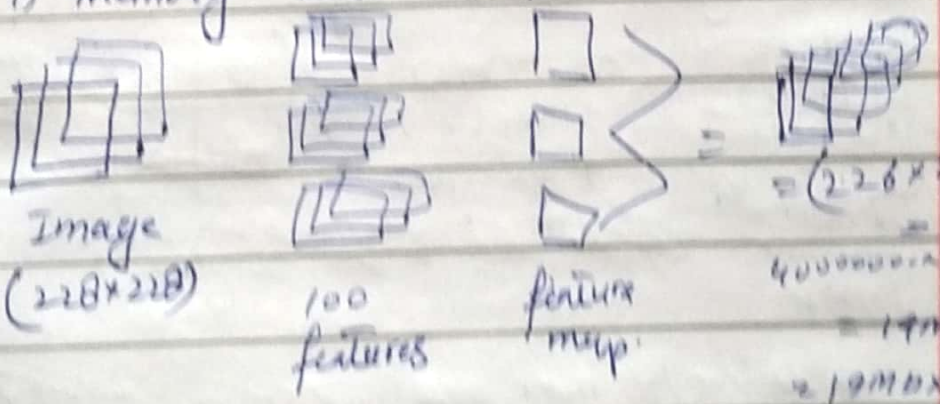
i) When you just want high level features.

ii) Computational power

- کہاں important ہے stride کو!

The problem with convolution:-

i) Memory issue ii) Translation variable.



$= (226\times 226)\times 100$

$= 4000000$

$4000000 \times 32\text{ bit}$

$= 19\text{MB}$

$= 19\text{MB}\times 32$ batch size

$= 1.55\text{GB}$

and our machine slow. size increased

تو اس سے بچنے کے لیے 16 stride کر سکتے ہو یا
پھر 16 layers pooling کی stride کر دیں
تو وہ ایک اور مسئلہ بھی solve کرتی ہے جو translation variance کا ہے

Date: ___/___/2025 M T W T F S

Convolution operation جب perform ہوتا ہے تو وہ translation variance کرتا ہے۔
features یہ edges کو detect کرتا ہے تو وہ direction پر depend کرتے ہیں۔ تو ہمیں direction پر depend نہیں ہونا چاہیے۔ کیونکہ بلی کا کان جہاں پر ہو ہے تو بلی کا کان ہی تو اسے translation variance کہتے ہیں۔ ہمیں تو translation-invariance چاہیے ہوتا ہے۔ تو pooling یہی کام کرتا ہے

You down sample features.

so that's why you add pooling.

تو آپ feature map کے size کو reduce کرتے ہو۔

تو آپ relu perform کرتے ہو۔ تاکہ non-linear آجائے۔

اس کے بعد آپ pooling perform کرتے ہو۔

Max pooling, Minpooling, ---

You decide: window-size, strides and type of pooling:-

3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

size (2,2)
stride(2)
type = max

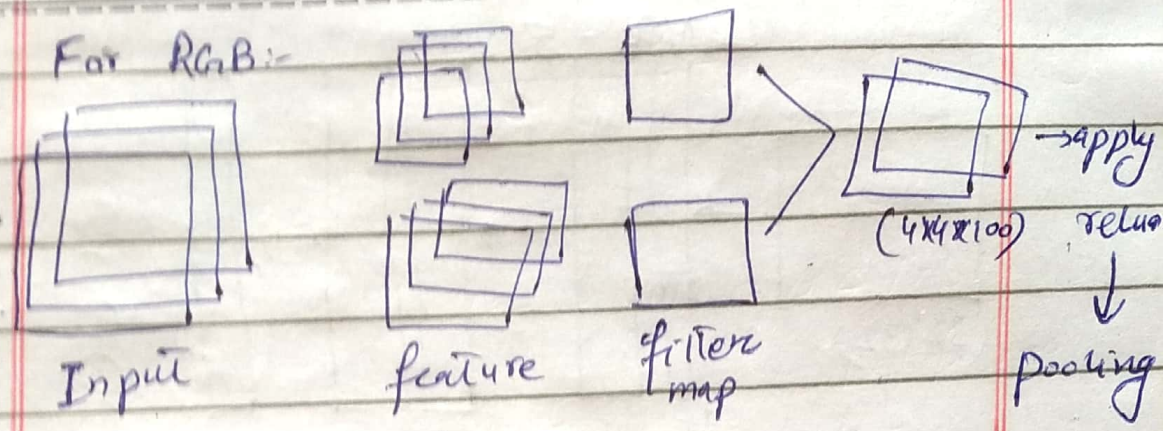
5	3
7	4

Now feature map

آپ window size کو max feature map پر چلا رہے ہو۔

تو اس میں وہ کیا کر رہا ہے وہ max نکال رہا ہے اور detail low-level کو values کی features strong کو eliminate کر رہے ہو۔

For RGB:-



اگر آپ کے پاس multiple feature-map ہیں تو آپ
Individual پر pooling perform کرو گے۔

Advantage :-

i) Reduced sized
(226x226x100) → 113x113x100

ii) Translation Invariance (does not depend on location)

iii) No need of training

Disadvantage:

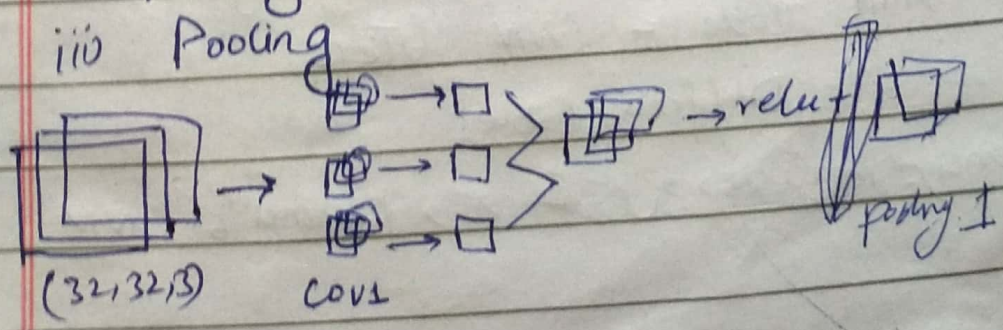
i) You loss a lot of information.

CNN Architecture:-

i) Convolution operation

ii) padding & stride

iii) Pooling

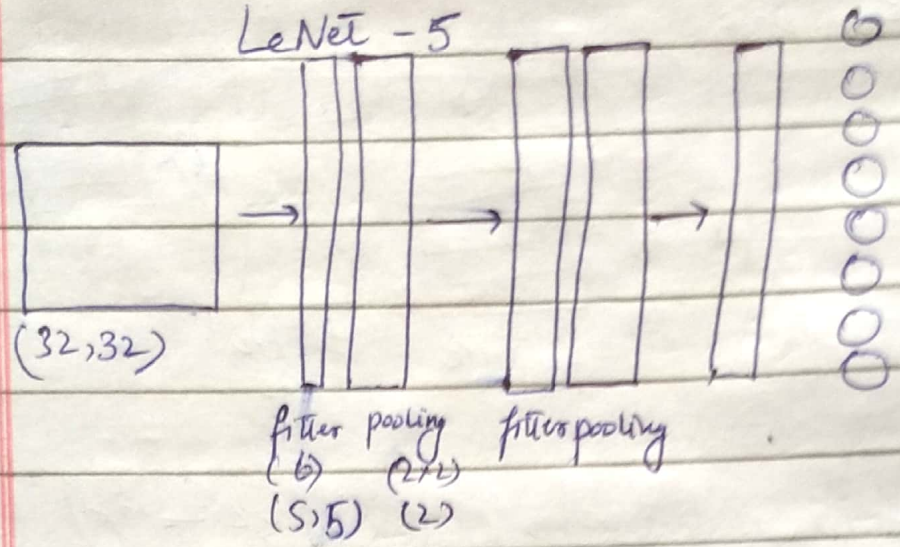


AlexNet, VGGNet, ResNet

Date: __/__/2025

M T W T F S

LeNet - 5



CNN vs ANN

1) Computation Cost
2) Overfitting
3) Loss of Important features
   ↳ like special arrangement of pixels.

Similarity in CNN & ANN:-

ANN میں آپ کیا کرتے ہیں particular node کے لیے weighted sum + bias add کرتے ہیں اور activation لگاتے ہیں۔

اسی طرح CNN میں بھی جب آپ feature-map نکالتے ہیں تو weighted sum + bias کرنے کے بعد activation لگاتے ہیں۔

یہ ہے similarity۔

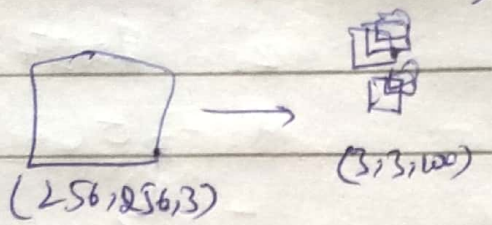
ANN node = CNN filters

Difference

CNN کا parameter جو ہوتے ہیں وہ input size پر depend

Date: ___/___/2025 M T W T F S

filter اور shape کی filter نہیں کرتے۔ کیونکہ وہ
کے size پر depend کرتے ہیں۔

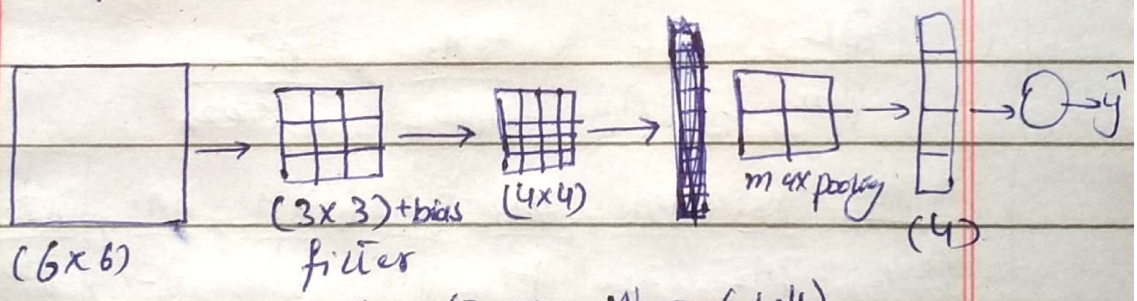


27×100
$= 2700 + 100 \text{ bias}$
$= 2800$

CNN میں input-shape پر depend کرتے ہیں۔ جتنے زیادہ
input-size ہوں گے اتنے زیادہ parameters ہوں گے۔

Backpropagation In CNN

Simple CNN Architecture:-

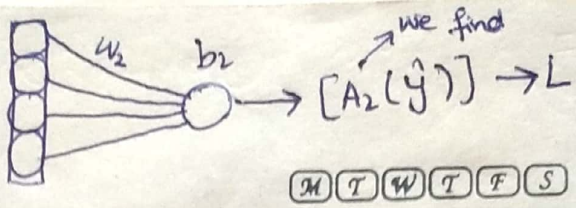


$W_1 = (3,3)$, $W_2 = (1,4)$
$b_1 = (1,1)$, $b_2 = (1,1)$ = 15 trainable parameters

Loss = binary-crossentropy

$$L = -y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)$$

$$X \longrightarrow \underset{+b}{\otimes} \longrightarrow Z \xrightarrow{\text{relu}} A_1 \xrightarrow[\text{max}]{\text{Pool}} P_1 \xrightarrow{\text{flatten}} F \xrightarrow{+\text{bias}} Z_2 \xrightarrow{\text{sigmoid}} A_2 \longrightarrow L$$



Date: ___/___/2025

M T W T F S

Forward (Prop)

$Z_1 = Conv(X, w_1) + b_1$, $F = flatten(P_1)$

$A_1 = relu(Z_1$ $\quad Z_2 = w_2 F + b_2$

$P_1 = maxpool(A_1)$ $\quad A_2 = \sigma(Z_2)$

$L(A_2)$

We have to apply Gradient Descent?

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}, \quad w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$

$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}, \quad b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial w_2}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial b_2}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_2} \times \frac{\partial A_1}{\partial Z_1} \times \boxed{\frac{\partial Z_1}{\partial w_1}}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_2} \times \boxed{\frac{\partial Z_1}{\partial w_1}}$$

$$\frac{\partial L}{\partial a_2} = \frac{\partial}{\partial a_2}\left(-y_i \log a_i - (1-y_i)\log(1-a_i)\right)$$

$$= \frac{-y_i}{a_i} - \frac{(1-y_i)}{(1-a_i)}$$

$$= \frac{-y_i(1-a_i) - (1-y_i)(a_i)}{a_i(1-a_i)}$$

Date: ___/___/2025 M T W T F S

$$\frac{a_2 - y_i}{a_2(1-a_2)}$$

$$\frac{\partial A_2}{\partial z_2} = \sigma(z_2)\left[1-\sigma(z_2)\right] = a_2\left[1-a_2\right]$$

$$\frac{\partial z_2}{\partial w_2} = \frac{\partial}{\partial w_2}\left(a_2(1-a_2)\right)$$

$\because z_2 = w_2 F + b_2$

$$\boxed{\frac{\partial z_2}{\partial w_2} = F} \quad , \quad \boxed{\frac{\partial z_2}{\partial b_2} = 1}$$

$$\frac{\partial L}{\partial w_2} = \frac{(a_2 - y_i)}{\cancel{a_2(1-a_2)}} \times F\,\cancel{(a_2(1-a_2))}$$

$$= F(a_2 - y_i) = (A_2 - Y)F$$

$$\frac{\partial L}{\partial b_2} = \frac{(a_2 - y_i)}{\cancel{a_2(1-a_2)}} \times \cancel{a_2(1-a_2)} \times 1$$

$$= (A_2 - Y)$$

$$(A_2 - Y)F$$

$$((1,1) - (1,1))\,(4,1)^t$$

$$(1,1) \times (4,1)^t$$

$$(1,1) \times (1,4)$$

$$(1,4)$$

$$\boxed{\frac{\partial L}{\partial w_2} = (A_2 - Y)F^t}$$

$$\boxed{\frac{\partial L}{\partial b_2} = (A_2 - Y)}$$

جہاں تو ایک image ہے۔ اگر آپ multiple images بھی
بھیجو گے تب بھی اُس کا same shape ہی آئے گا۔

A_2 ، A_1 ، F میں matrix کی شکل
(4,4) (4,4) (4,1)

Date: ___/___/2025

M T W T F S

For all back propagation :- the charts :-

$$X \longrightarrow \underset{\uparrow \atop w_1, b}{\otimes} \longrightarrow Z_1 \xrightarrow{relu} A_1 \xrightarrow[pool]{max} P_1 \xrightarrow{flatten} P_1 \downarrow F$$

$$L \longleftarrow A_2 \xleftarrow{\sigma} Z_2 \longleftarrow \underset{\uparrow \atop w_2, b_2}{\odot} \longleftarrow F$$

$\frac{\partial L}{\partial b_2}$, $\frac{\partial Z}{w_2}$ we already calculate,

$$\frac{\partial L}{\partial b_1} = conv\left(X, \frac{\partial L}{\partial Z_1}\right)$$

$$\frac{\partial L}{\partial b_1} = Sum\left(\frac{\partial L}{\partial Z_1}\right)$$

Data Augumentation

i) Reduce overfitting

You create new photoes from one photo.

اگر آپ کے پاس data نہیں ہے تو آپ data generate کر سکتے۔

medical field میں بہت زیادہ help کرتی ہے کیونکہ data کو gather کرنا بہت زیادہ costly ہوتا ہے اور deep learning is all about data.

Date: ___/___/2025 M T W T F S

We use Image Data Generator

i) rotation_range e.g 30

ii) shear_range 0-1

iii) zoom_range 0-1

iv) horizantal_flip True

v) width_shift_range 0-1

vi) height_shift_range 0-1

flow کا مطلب آپ ایک generate image کرنا چاہتے ہیں۔

flow_from_directory کا مطلب جب آپ کی direct میں Images پڑی ہوں 2 ہیں۔ اور ایک ساتھ سب process کرنا چاہتے ہیں۔

```
train_datagen.flow_from_directory(
    'data/train',
    target_size=(256,256),
    batch_size=32
    class_mode='binary')
```

note: آپ model.fit کے اندر model.fit_generator دیتے ہیں۔ اور training کے دوران آپ نے جو data کو generate کروایا ہے وہ use ہوگا نہ کہ آپ کا data real

Pre-Trained Models:-

Why use pre-Trained models?

Deep Learning is data hungry.

Date: ___/___/2025 M T W T F S

تو آپ کو بہت سارا data چاہیے اور
problem binary میں خوب سا data-label چاہیے۔

2) Time Consuming:-

So we use pre-Trained model.

- Image Net Dataset-
  ↳ Visual ~~Imag~~ database of images.

2006 میں deep Learn دوبارہ سے آگے جانے لگا۔ تو
اس وقت models پر بہت کام ہونے لگا تھا۔ تو ایک
data پر images 1.4 milliون Fei-fei Lie نے لے کے CS-
train کیا جس میں 20,000 classes تھے۔ اور اس میں
10 millions تو ایسے تھے جس میں object detection بھی کر سکتی تھی۔

- Image Net Challenge :-
  ↳ 2010 start

2012 میں deep-learning network نے
first time use CNN کر کے 16% لیا تھا۔
first time use relu.
first time use Deep-Learning.

2012 - Alex Net - 16.4%
2013 - ZFNet - 11.7%
2014 - VGG - 7.3%
2015 → GoogleNet - 6.7%
2016 → Resnet - 3.5%

Date: ___/___/2025 M T W T F S

Idea of pre-Trained Models:-

i) No need of Treening.

ii) No need of time

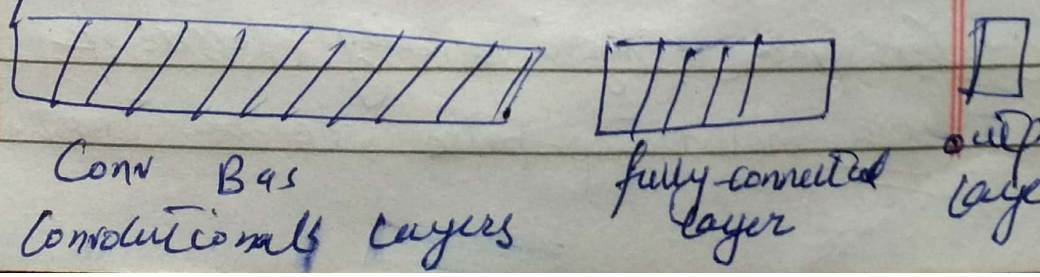
iii) No need of data

Transfer Learning

کسی ایک dataset پر Train کیے گئے model کو
کسی دوسرے dataset پر run کرتے ہیں۔

Pre-Trained model has an issue:-
اگر کوئی class ان 1000 classes میں نہیں
آتی ہے تو آپ پھر یہ models استعمال نہیں کر سکتے۔

For example:
Tablets / Mobile phone because this is not in 1000 classes of VGG16.
Thats why Transfer Learning comes.
Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying to a different but related problem. Structure:-



Date: __/__/2025 (M)(T)(W)(T)(F)(S)

Imagine there is no cat dog category in VGG16. So now how you apply Transfer Learning.

تو پھر ہم صرف conv Base والے part کو رکھ لیتے ہیں
اور fully-connected کو ہٹا دیتے ہیں۔ تو پھر اس کی جگہ
آپ اپنی dense layer لگا دیتے ہیں۔ اور آپ
اپنے conv base کو freeze کر دیتے ہیں۔ تو پھر
train conv Base نہیں ہوگا۔ بلکہ صرف dense والا
train ہوگا۔

This is called Transfer Learning

Why it works?

ہم جانتے ہیں کہ پہلے والی layers کو edges اور corners
fetch کرنے کے لیے ہوتے ہیں real-world میں ہر چیز کے
same corners اور edges ہوتے ہیں۔ تو اسی لیے
یہ بھی کام کرتا ہے اور ہم conv کو freeze کر دیتے ہیں۔
"بار بار پہیا نہ بناؤ، پہیا بن چکا ہے تو اسے چلانا سیکھو"۔

Ways to apply:

1) feature extraction 2) fine Tuning

جو ابھی آپ نے سیکھا ہے۔

اس کا استعمال آپ تب کرتے ہیں جب آپ کو پتا ہوتا
ہے کہ Imagenet already-classes میں train ہو چکی ہیں۔

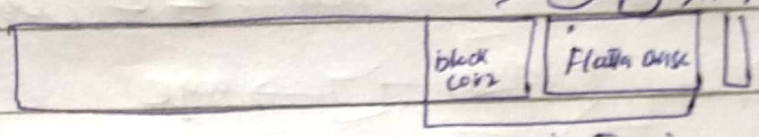
e.g include_Top=False , cov-baise.Trainable=False

Date: ___/___/2025

M T W T F S

Pre-Training:-

اس میں آپ کچھ conv layers کو بھی retrain کر دیتا ہے
جب imagenet کی classes میں آپ کے data جیسی
classes نہ ہوں تب۔



again-Train

```
conv_base.Trainable = True.
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_Trainable = True.
    if set_trainable:
        layer.Trainable = True
    else:
        layer.trainable = False
```

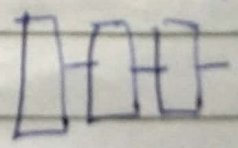
Keras Functional Model

Linear Topology وہ ابھی تک ہم نے جو بھی models پڑھے ہیں
جو follow کرتے۔

ANN-CNN ⟶ we use sequential model.

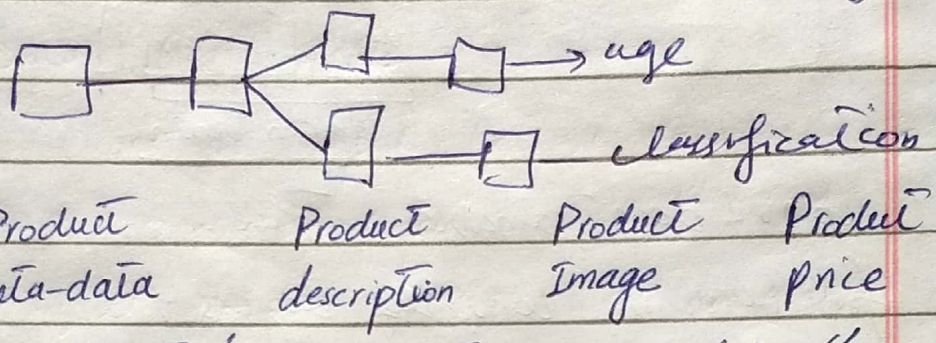
- 1- input
- 1- output
- Linear
- series

Linear Topology

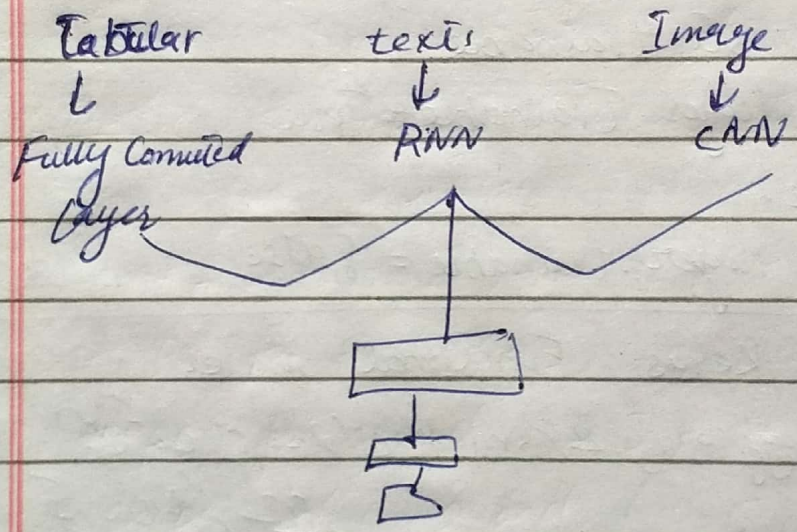


Date: __/__/2025 M T W T F S

e.g ہم نے دو output predict کرنے ہیں۔
کہ وہ sad ہے یا happy اور دوسرا age کی predict کرنا ہے۔
یا تو آپ دو linear model بنائے گے کہ ایک age کے دوسرا
classification کے لیے۔



اگر آپ کے پاس تین input ہیں اور ایک output ہے تو آپ
کیسے بنائے گے۔



Date: ___/___/2025 (M)(T)(W)(T)(F)(S)

. Recurrent Neural Network: Sequential Data
Which is specifically used in [Text NLP]
It is type of sequential model.

Non-Sequential Data:-
where sequence does not matter:
e.g cgpa, gender, marks => gender, marks, cgpa

Sequential Data:-
✓ :Hey, My name is Ali
✗ My Name, Hey is Ali.
:Like Time-Series is also sequential data.
:Speach is also their
:DNA Squance

— Why we need RNN?
اگر آپ data sequential کے لیے ANN استعمال کرنا
چاہتے تو پہلے اس text کو number میں convert کرنا
چاہیے گے اور پھر ANN لگاؤ گے۔ پھر اس میں ایک
problem ہے۔ کہ آپ ایک unnecessary computation کرو گے
zero-padding کرو گے تو پھر بھی یہی مسئلہ سامنے آئے گا۔
i) Input size length varying size (small, largen parag)
ii) zero padding with unnecessary computation.
iii) prediction is not correct.
iv) ANN has no memory (پچھلے کو یاد نہیں)

Date: ___/___/2025 M T W T F S

Applications:-

i) Sentiment Analysis:-

ii) Image Caption generator.

iii) Machine Translation

— Data for RNN:-

(timesteps, input_feature) Reviews | input

movies was good 1

..... bad 0

not good 0

In oneHEncoding

Uniquee Words:-

Movies was good not

[1 0 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]

t=1 t=2 t=3 t=4

- کیونکہ timesteps input میں RNN کو

for first input:-

time 1, time 2, time 3 | output

Shape for firs (3, 5) → total words

shape for second (3, 5)

shape for second (4, 5) movie was not good

In keras Simple RNN

batch-size, timsteps, input features

3 , 4 , 5

length of input , doing sentence, total unique words.

Date: ___/___/2025 M T W T F S

Who RNN works?:-

	Review : X	Sentiment
X_1	X_{11} movie X_{12} was X_{13} good	1
X_2	X_{21} movie X_{22} was X_{23} bad	0
X_3	X_{31} movie X_{32} was X_{33} not X_{34} good	0

RNN is similar to ANN but two differe

① In ANN input → hidden → output

In RNN input → hidden → output

② دوسرا difference یہ ہوتا ہے کہ آپ کس طریقے سے data input کر رہے ہیں۔ آپ ایک ایک review ایک time اور

At time=0 ۔ input کی basis پر کھینچتے ہیں

out ← you pass X_{11}

At time=1 output ← you pass X_{12}

آپ پورے input کو feed نہیں کرتے ایک ایک کر کے کرتے ہیں

③ ANN is a feed forward neural network

input to output → same to same.

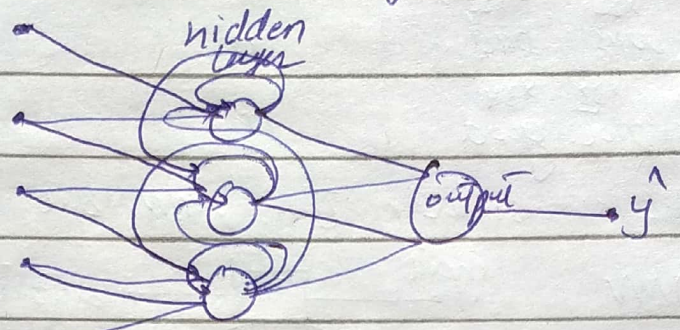
RNN is not same.

آپ کا جو hidden layer ہے وہ واپس پیچھے کر feedback کھینچتا ہے اور یہی difference ہے

input → h → output

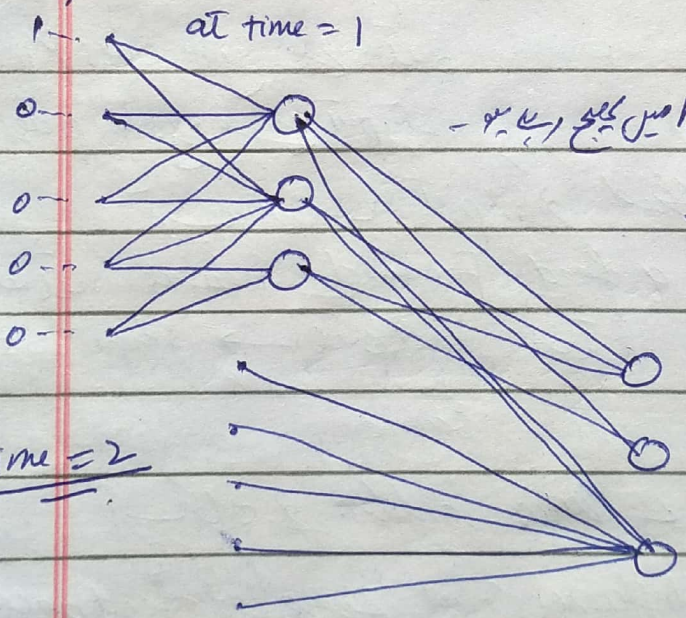
Date: ___/___/2025 M T W T F S

First to remmember, you input layer consiste of 5 nodes because every word consiste of 5 words.



کیونکہ یہ ایک recurrent hidden unit ہے تو اس کا output اس کا دوبارہ input بن جائے گا۔ مطلب یہ خود کو ہی feedback دے گا۔ مطلب ہر ایک node سے جو output نکلے گا وہ اس کو دوبارہ مل جائے گا۔ مطلب

For first movie first word



تو پھر آپ اس hid میں بھیج رہے ہو۔ at time=2 پر۔

سب سے بڑی بات یہ ہے:

آپ نے اپنا پہلا word input میں دیا at time=1 تو اب ہر node کا ایک input ہے "0"۔ تو آپ اسے "0"

Date: ___/___/2025 M T W T F S

hidden layer میں بھیج رہے ہیں time=2 کے لیے۔

time = 2 for second word "was"

تو اس timestep پر صرف was کا vector جائے گا۔

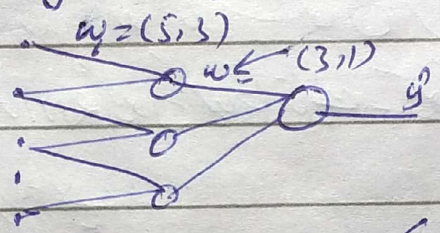
پچھلے timestep سے جو output آیا تھا وہ بھی جائے گا۔

اب وہی چیز آتی رہے گی continue ہوتی رہے گی۔

time = 0 ⟶ پہلا input ملے گا۔

تو اس جگہ پر کوئی random یا 0 input دے گا۔

How many weights & biases.



اور اس hidden کے بھی 9 weights ہوں گے۔ کیونکہ total weights 24 ہوں گے multiply سے بھی ایک دوسرے سے

RNN Forward Propagation

Review ~~Sentiment~~ | sentiment

we use concept unfolding through time.

- number input 5 node ہیں آپ کے پاس تین

activation ⟵ $\boxed{h}$ ⟶ O_1 (1,3)

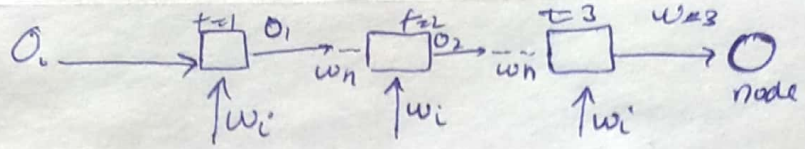
w_{11} time = 1

X_{11} (5,1)·(1,3)

= (5,3)

$= w_{11}X_{11}$ + bias

$f(x_{11}, w_{11}) = O_{11}$

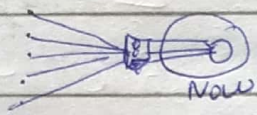


Date: ___/___/2025

now $t=2$ second word + output (t_1) O_1

$t=2$ $f(x_{12} w_i + O_1 w_n + b) = O_2 \ (1,3)$

$t=3$ $f(x_{13} w_i + O_2 w_n + b) = O_3 \ (1,3)$



Now this:-

$(w_{21} + w_{22} + w_{23})$

$(1,3) \times (3,1)$

$= (1,1)$ output.

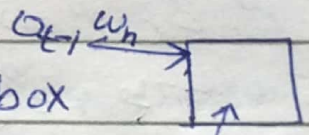
آپ ہر بار w_i کا استعمال کر رہے ہو اسے weight sharing کہتے ہیں

اسے RNN کیوں کہا جاتا ہے۔ کیونکہ اس کی hidden layer

recure کر رہی ہے۔

Summary:

You imagin it a box



two input - $X_{it} \cdot w_i + b$ X_{it}

$O_{t-1} \cdot w_n + b$ $\rightarrow f(\text{vector})$

$\hat{y} \leftarrow f(O_t \cdot w_o + b) \leftarrow O_t \leftarrow$

Types of RNN

Many To One

One To Many

One To One

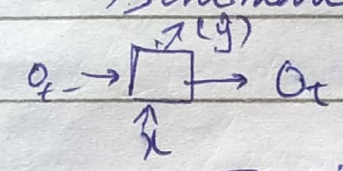
Many To One.

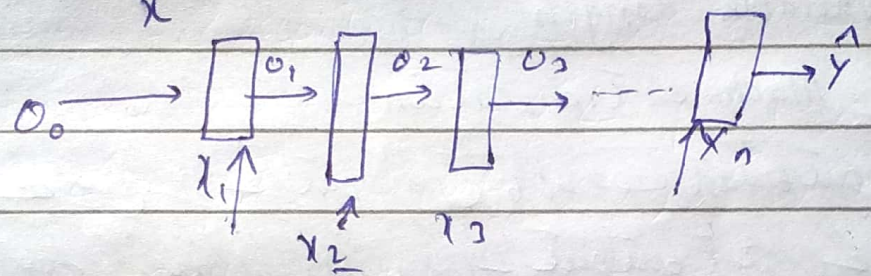
① Date: __/__/2025 Many To One. M T W T F S

input → sequence → character, time-series

output → non-seq → sentiment (0,1) → scaler

i) Sentiment Analysis ii) Rating Prediction

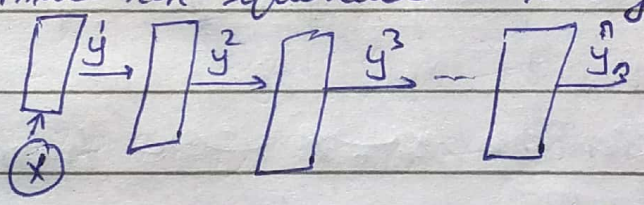




Multiple ~~outputs~~ inputs and only outputs.

② One To many

X → normal non-sequential → Images/number/



Send images and predict results (description of the image), Music Generation

③ Many To Many

X → sequential
Y → sequential.

seq-2-seq → Same-length, m-to-many → NER, POS TAGGING

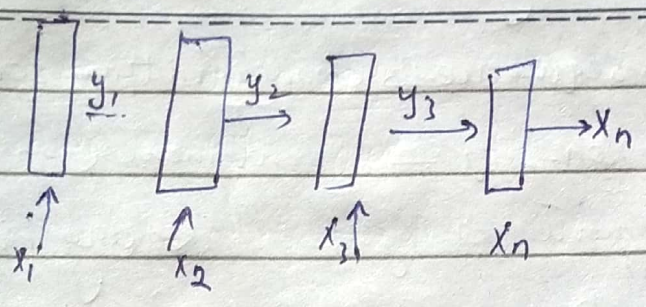
seq-2-seq → Variable length

len(input) = len(output)

POS TAGGING

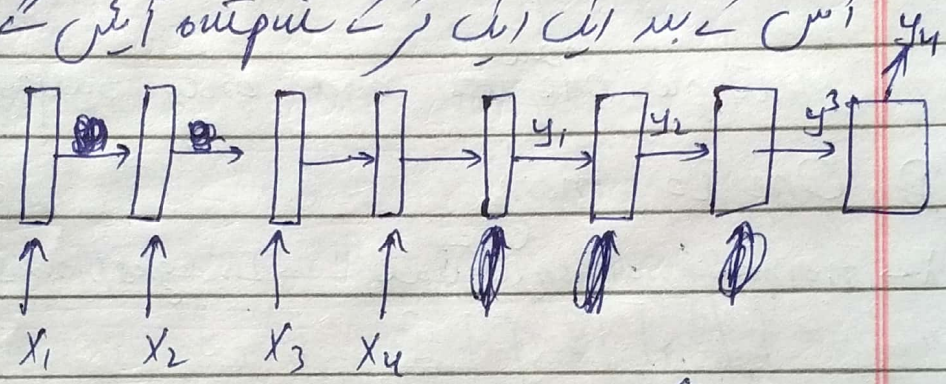
My	name	is	Ali
P	n	v	n

Date: ___/___/2025 M T W T F S



— Variable - Length Machine Translation

پہلے میں سارے input دوں گا اور جب last input دوں گا
اس کے بعد ایک ایک کر کے output آئیں گے۔



translation کے وقت آپ پورا sentence پڑھتے ہیں اور اس
کے بعد result نکالتے ہیں۔ انہیں encoder-decoder کہتے ہیں۔

— One - To - One

~~on~~ X	⟶ non-sequential	Image
Y	⟶ non - sequential	classification

Backpropagation In RNN :-

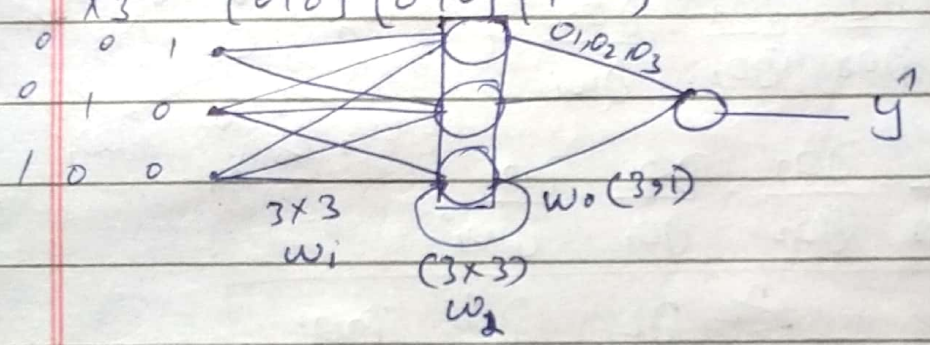
text	Output
cat mat rat	1
rat rate mate.	1
mate mate cat	0

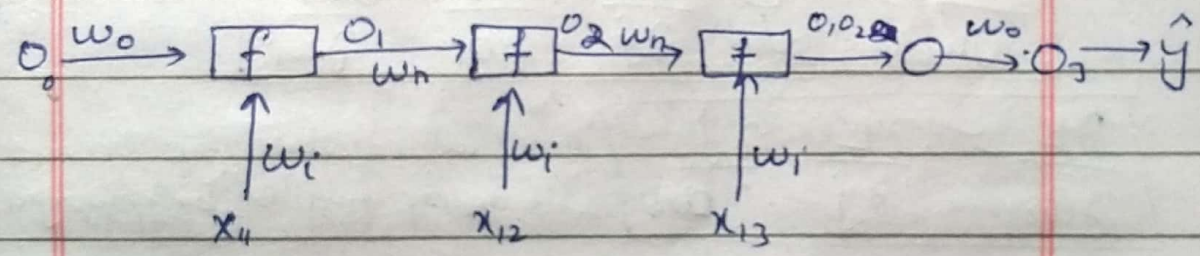
Date: ___/___/2025 M T W T F S

Vocab [iate, mat, rate] = 3

[1 0 0] [0 1 0] [0 0 1]

	X	y
X_1	[1 0 0] [0 1 0] [0 0 1]	1
X_2	[0 0 1] [0 0 1] [0 1 0]	1
X_3	[0 1 0] [0 1 0] [1 0 0]	0





$O_1 = f(x_{11} w_i + O_0 w_0)$

$O_2 = f(x_{12} w_i + O_1 w_h)$

$O_3 = f(x_{13} w_i + O_2 w_h)$

$\hat{y} = \sigma(O_3 w_0)$

we want to minimize the loss

apply back prop with Gradient Descent

$w_i = w_i - \eta \frac{\partial L}{\partial w_i}$ $\qquad L = -y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i)$

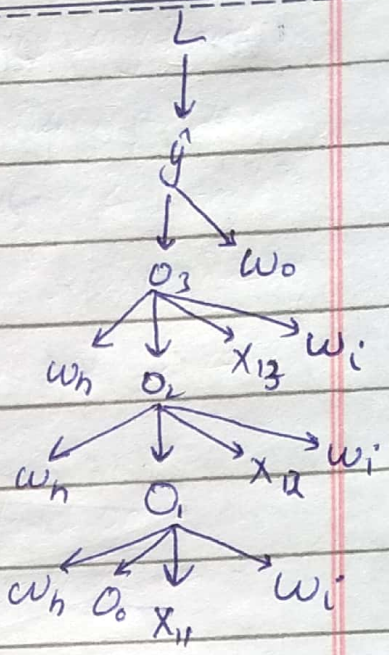
$w_0 = w_0 - \eta \frac{\partial L}{\partial w_0}$

$w_h = w_h - \frac{\eta \partial L}{\partial w_h}$

Date: ___/___/2025

M T W T F S

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_i}$$



$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial w_i} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial w_i} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial w_i}$$

$$\boxed{\frac{\partial L}{\partial w_i} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_j} \cdot \frac{\partial O_j}{\partial w_i}}$$

$$\boxed{\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_0}}$$

First we calculate this because it appear first in backpropagation.

$$\frac{\partial L}{\partial w_h} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial w_h} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial w_h} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial w_h}$$

$$\boxed{\frac{\partial L}{\partial w_h} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_j} \cdot \frac{\partial O_j}{\partial w_i}}$$

Date: ___/___/2025 M T W T F S

Problem with RNN

↳ Suitable for sequential data = Textual / Time-series

They are not use as much.

2-major problem:-

(1) Problem of long-term dependency.

(2) Unstable Training

RNN میں کیا ہوتا ہے کہ word پچھلے word پر depend
کرتا ہے۔ پھر اگر یہ data بڑھ جائے تو یہ پچھلے words کو
next-word-predictor بھول جاتا ہے۔

because of vanishing gradient descent
Exploiding gradient descent

1 Problem of long term dependency
In backpropagation:-

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_1} + \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_1} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_1} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$

یہاں تو آپ کے پاس input میں صرف 3 نمبرز تھے۔
پھر اگر 100 سے زیادہ ہوں تو ان کو multiply کرنے سے
value تو بہت کم ہو جائے گی اور loss میں ان کا contribution
بہت کم ہو جائے گا۔ اور short-term contribution زیادہ
رہے گا۔

Date: ___/___/2025 M T W T F S

تو Long Term کی value بہت کم لیتی ہو جاتی ہے like = 0.0001
کیونکہ ہم function لگاتے ہیں وہ tanh ہوتا ہے اور وہ
(0,1) کے درمیان value دیتا ہے۔ اور weight بھی ہم
بہت بار 1 اور 0 کے درمیان دیتے ہیں۔ تو اس سے value
ہوتی ہے وہ almost 0 ہو جاتی ہے۔

Solution:-

i) Different Activation function:-

ii) Different weights initialization

iii) LSTM

Problem #2

Exploiding Training:-

اگر learning rate یا b, w کی value زیادہ ہو جائے اور
تو لحیرہ بہت بڑا ہو جائے گا۔ اور اگر ہم activation fn
میں relu لگائے تو پھر possitive value سے بھی
بہت بڑا نمبر آ جائے گا۔

LSTM

For LSTM:

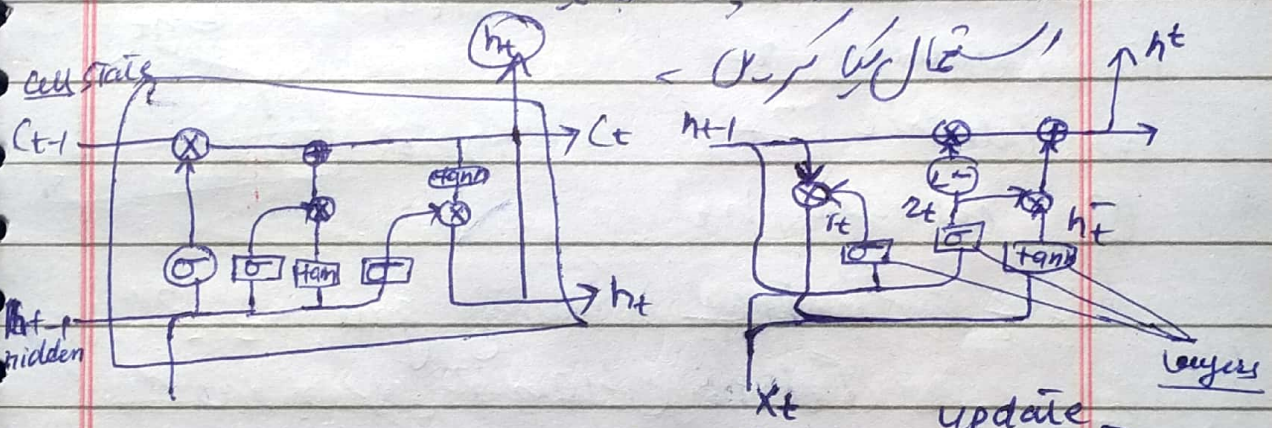
آپ کو دوسری notebook میں جانا ہو گا آخری pages پر۔

GRU

اگر آپ lstm کو دیکھو تو اس کے structure complex
ہونے کی وجہ سے اس کے trainable param بہت زیادہ

Date: ___/___/2025 M T W T F S

Training سے بڑھ جاتے ہیں۔ اور اگر بہت بڑا data ہے تو
میں بہت زیادہ problem ہو سکتی ہے۔ جس کی وجہ
سے gru آیا اور اس کا structure بہت زیادہ
complex نہیں اور جس کی وجہ سے parameters کم ہونے
کی وجہ سے training lstm کے مقابلے میں fast ہوتی
ہے اور دوسرا آپ کو چاہیے کہ دونوں models کو
استعمال کیا کریں۔



Theres two gates is Reset ii) ~~mahuplate~~ update

$\underbrace{h_{t-1}, h_t, X_t, r_t, z_t, \bar{h}_t}_{\text{vectors same dimensions}}$

without X_t

X_t = reset-gate

z_t = update gate.

$\bar{h}_t$ = candidate hidden state.

$\boxed{\sigma, \sigma, \tanh}$ are the nn layer which has same number of units.

h_{t-1}, اور dimensions کا ان NN layers کو اچھا ہے
same exactly dimen. کا $\bar{h}_t, r_t, z_t, X_t$ ہوتا ہے

Date: __/__/2025 M T W T F S

X_t is a vector why?

کیونکہ یہ آپ کا input ہوگا جا رہا ہے وہ text ہوگا جا رہا ہے

ہم vector میں convert کریں گے۔

steps:-

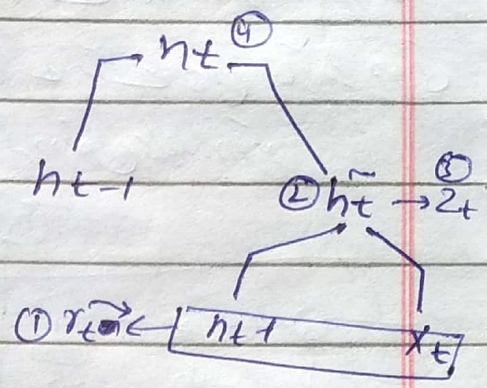
i) calculate r_t

ii) calculate $\tilde{h}_t$

iii) Calculate r_t

iv) Calculate Z_t

⑤ calculate h_t



اچھا یہ memory gru کو maintain کرتا ہے۔

کہانی کو یاد کرو۔ جو Matrix میں دی گئی تھی۔

4-factors= [Power, Conflict, Tragedy, Revenge]

اب gru میں hidden layer کیسے calculate ہوتی ہیں۔

first $t=1$ [0.7 0.7 0.4 0]

$t=2$ [0.9 0.6 0.3 0]

$t=3$ [0.8 0.8 0.6 0.5]

$t=4$ [1 0.9 0.5 0.7]

$t=5$ [0.7 0.8 0.3 1]

How to switch context

① How did GRU

$h_{t-1} \xrightarrow{①} \tilde{h}_t \xrightarrow{②} h_t$

مثال میں لے کر ہمارا h_{t-1} یہ ہے۔

$[0.5, 0.6, 0.7, 0.1] = h_{t-1}$

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

$x_t \rightarrow [0.8, 0.4, 0.5, 0.3] \rightarrow \tilde{h}_t$

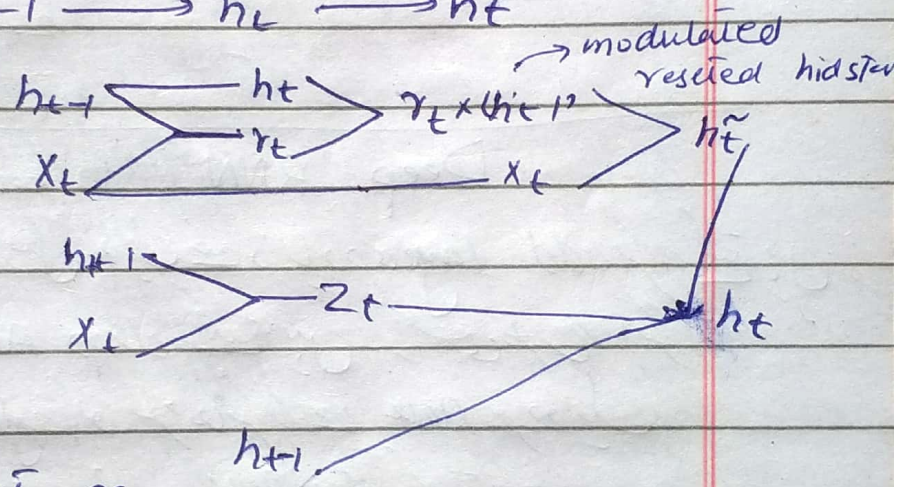
تو اس کی میں $\tilde{h}_t$ کو h_t نہیں کہہ سکتے کیونکہ
اس میں ہو سکتا ہے کہ x_t اتنا اہم نہ ہو۔
تو آپ ان balance کرنے ہو اور ان کے بیچ میں جاتے ہو۔

$$h_{t-1} \text{ — } \tilde{h}_t$$
$$h_t$$

you do that

$$h_{t-1} \xrightarrow{r_t} \tilde{h}_t \xrightarrow{z_t} h_t$$

Steps:- Charts:-



how to calculate r_t

r_t is a gate and vector.

جس کا کام ہے آپ کی past کی information والی کو reset کرنا۔

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$$

$$h_{t-1} * r_t = [0.48, 0.2, 0.07, 0.9]$$

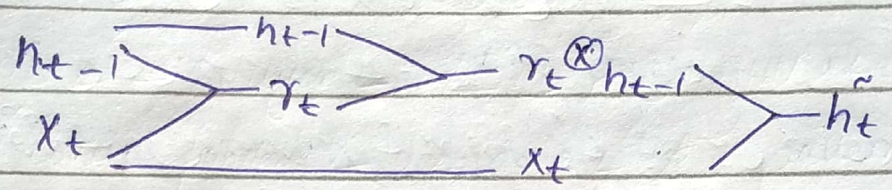
$$\tilde{h}_t = \tanh(W_c[h_{t-1} * r_t, x_t] + b_c)$$

h_{t-1}, x_t $\rightarrow$ z_t

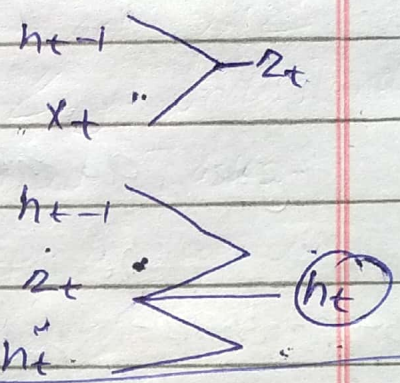
$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z)$$

Date: ___/___/2025 M T W T F S

$$h_t = (1 - z_t) \otimes h_{t-1} \oplus z_t \otimes \tilde{h}_t$$

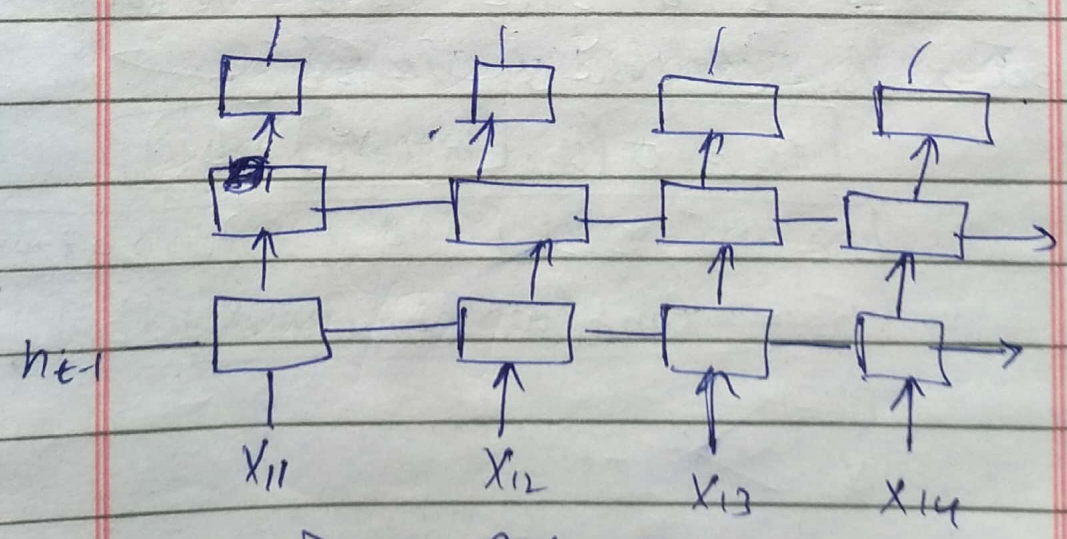


This is the flowchart to calculate GRU gates.



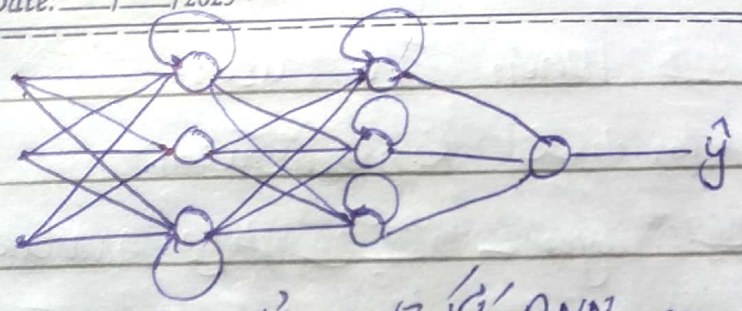
Deep RNN / Stacked RNN

جیسے ANN میں ایک سے زیادہ layers add کر سکتے ہیں۔ تو اسے deep ANN کہتے ہیں۔ اسی طرح اگر RNN کی layers کو بڑھا کر اسے stack کر دو ایک دوسرے کے ساتھ تو اسے deep RNN کہتے ہیں۔

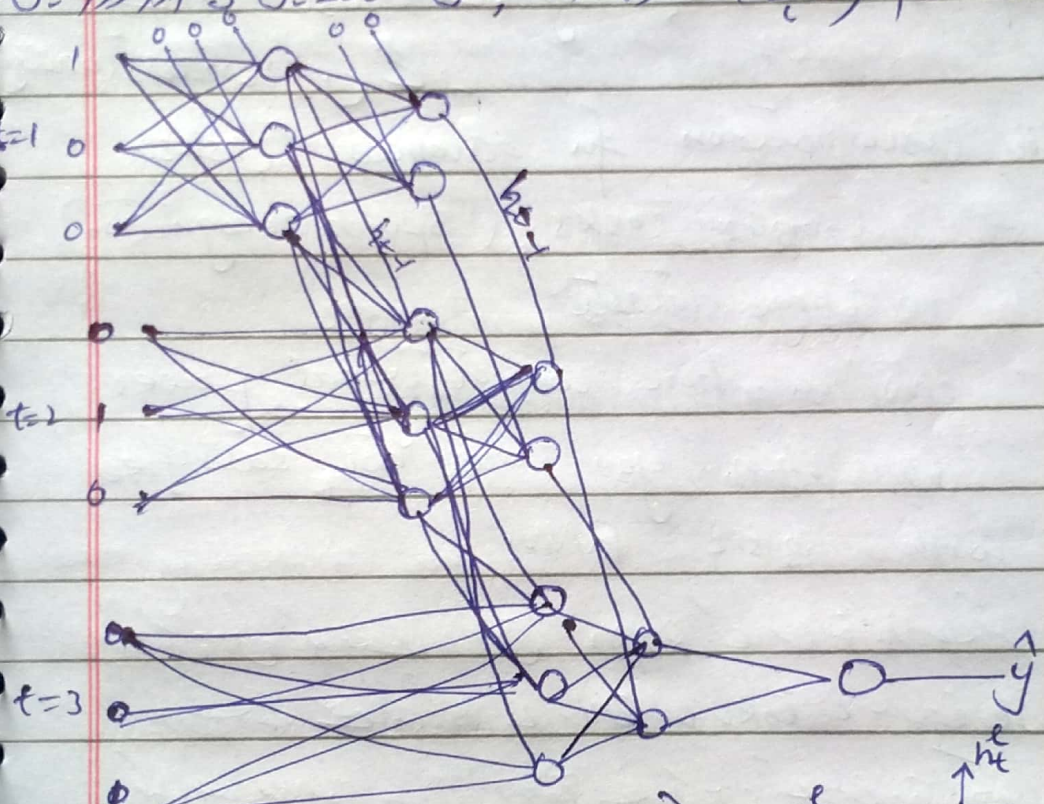


Deep RNN To improve accuracy

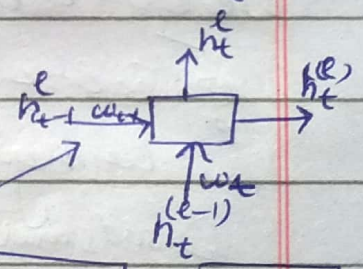
Date: ___/___/2025

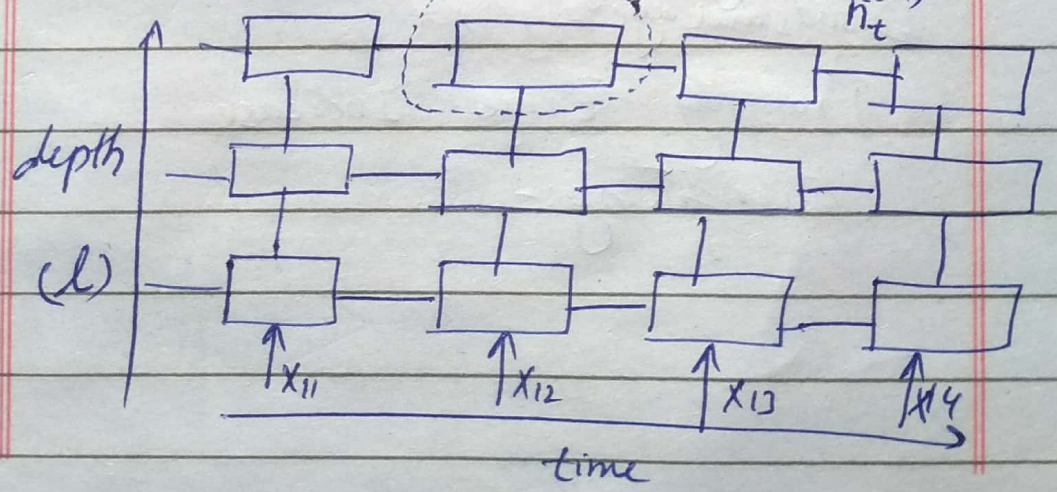


جیسے simple RNN کام کرتا ہے۔ اُسی طرح deep RNN بھی کام کرتا ہے۔ اگر ہمارے پاس تین تہیں ہیں اور دو کرنے ہیں



$$h_t^{(l)} = \left[\tanh\left(w_{t-1}(h_{t-1}^{l}) + w_t(h_t^{(l-1)})\right)\right]_2^{(3)}$$





"آپ کو return-sequence کو True رکھنا پڑے گا"
except your last year

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

Why & When To use:-

i) Heirarichal Representation:-

ANN like کی طرح پہلے والے layer آپ کے text کو cover fundamental کریں گے middle والے pattern deep کو data آپ کے layers آخر والے understanding لیں گے۔

ii) Customization for advanced Tasks:-

i) Complex Tasks (Speech recognition)

ii) Large dataset

(iv) You have computational power.

(v) When you are not satisfied with basic RNN.

"اس کو آپ باقی RNNs کے ساتھ ملا کر سکتے ہو۔ آپ اسے LSTM کے ساتھ یا GRU کے ساتھ use کر سکتے ہو۔" اور آپ اس کے ساتھ کچھ learning rate، Dropout like بھی لگا سکتے ہو

Disadvantage:-

Training Time Increase

Overfitting:-

BiLSTM کو بھی کسی بھی type کے RNN کے ساتھ rape کر سکتے ہیں۔ Bidirectional

Date: ___/___/2025 M T W T F S

Bidirectional RNN

RNN میں کیا ہوتا ہے کہ ہم left سے right کی طرف چلتے ہیں۔ اور inform. store کرتے ہیں۔

پھر اگر ایسا ہو جائے کہ آگے والا input پیچھے والے output کو effect کرے تو تب کیا کریں گے مثال کے طور پر: NER :- Named Entity Recognition

I love Amazon, its a great website

I love $\boxed{\text{Amazon}}$, its a beautiful river

ORG/LOC

تو ہمارا ماڈل کنفیوز ہو سکتا ہے

We can use Bidirection RNN.

Machine Translation —— Chatbots etc.

How Bidirection RNN solve this problem?

It use Two different RNNs

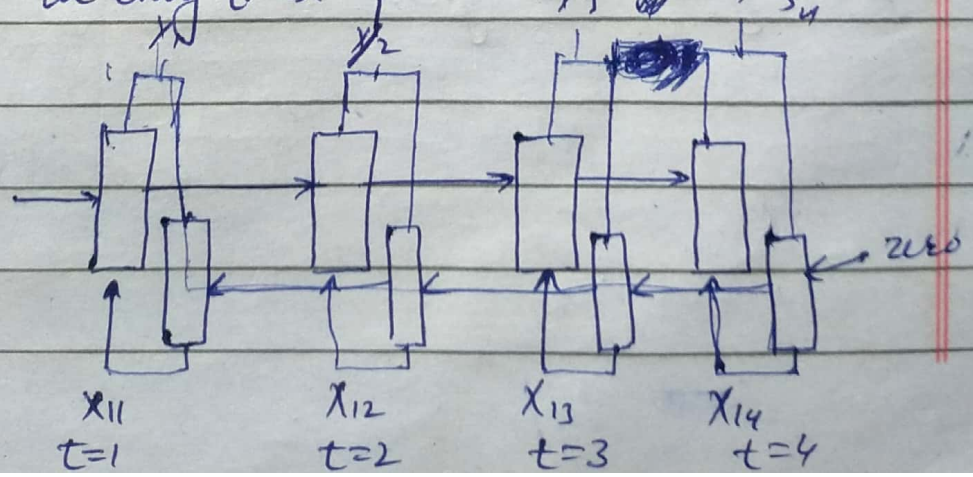
$\overrightarrow{RNN}$ $\overleftarrow{RNN}$

forward backward

left-Right Right-left

concatenate

at every timestamp



Date: __/__/2025 M T W T F S

Application and Drawbacks

i) NET ii) POS TAGGING

iii) Machine Translational.

iv) Time series forecasting

Drawbacks:-

i) Complexity → parameters - Increase.

ii) Overfitting

Embeddings

i) Keras embedding layers

input_dim * Vocabulary-size

output_dim * vector-size

trainable=True * Whether embedding are trainable.

When you use embedding in Keras:

i) It start with random numbers for each word. During training, it learns which word should be close together.

So good and great

Date: __/__/2025 M T W T F S

Glove Embedding:-

It is a pre-trained embedding learned on huge corpora (Wikipedia,) etc. Instead of learning yourself, you import - pretrained one.

i) You download embedding glove.6B.100d.txt

ii) Each word has a ready made vector of 100 numbers.

iii) These vectors already encode meaning and relationship.

ADvantage:-

Work well even on small data

capture strong semantic relationship

Fixed vocabulary, can't handle unknown words well.

Word 2Vec Embedding:-

CBOW | skipgram is two types.

Steps:-

Convert your dataframe into sentence.

Use word-2-vec class to ~~conv~~ make war-2-vec model using sentence

Date: __/__/2025 M T W T F S

3. Tokenize Text for Keras
4. Convert words to sequences.
5. Build word2Vec embedding matrix.
6. Build the LSTM model.

Encoder-Decoder

How much challenges we face previously?

i) Tabular Data → ANN → Deep Learning

ii) Image base → CNN

iii) Sequential data → RNN's → LSTM → GRU

iv) Sequence to sequence.

- جبکہ input میں بھی sequence ہو تا ہے اور output میں بھی -

~~i) Imag~~ e.g. Machine Translation

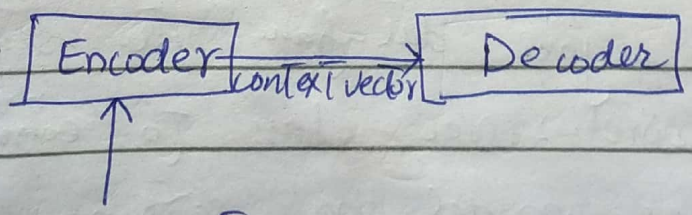
input → variable length

output → variable length.

3-words input → 300 words output

So we understand:- Encoder-decoder

- E کی Task ہے machine learning کی



Nice to meet you.

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

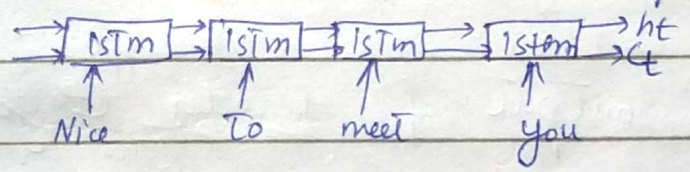
We sent one by one words:-

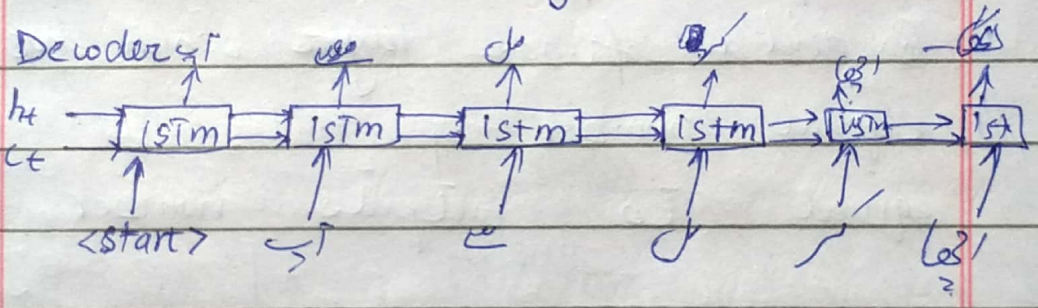
output has a vector

پھر اس کے بعد context vector میں اس vector کو Decoder میں بھیجے گا۔ اور Decoder اسے translate کرکے دے دے گا۔

Encoder:-

It is an lstm which unfold over Time.





How To train Encoder- Decoder:-

i) First Collect Data:-

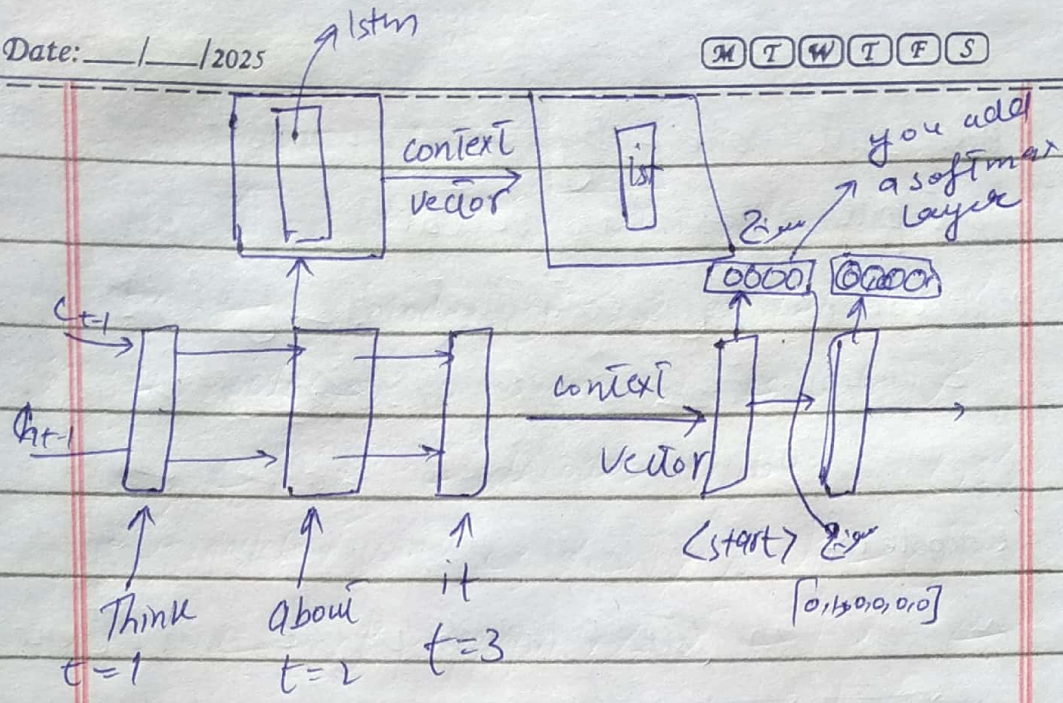
e.g

english	Urdu
Come in	اندر آجاؤ
Think about it	سوچ لو

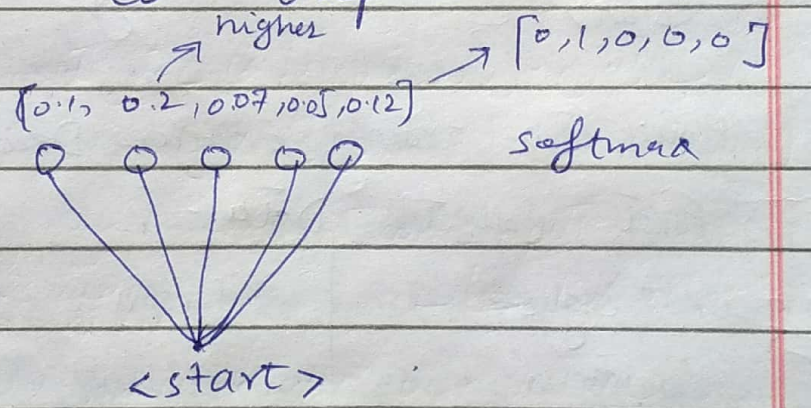
We apply one hot encoding:-

[(1,0,0,0),(0,1,0,0,0)]	<start>
(0,0,1,0,0),(0,0,0,1,0),(0,0,0,0,1)]	<end>

Date: ___/___/2025 M T W T F S



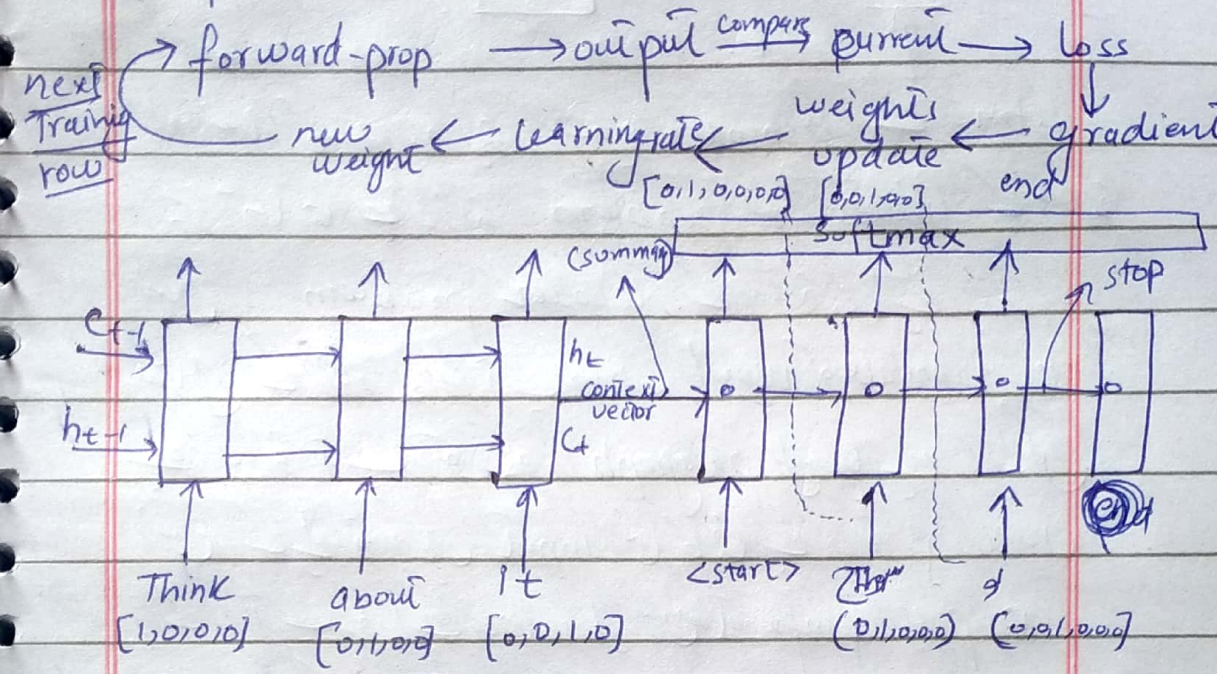
In first output of decoder you add a softmax layer which has equal number of nodes as equal to outputs number



آپ جب decoder میں جاتے ہو تو اپنا پہلا input <start> دیتے ہو اور output میں جو آئے گا غلط ہو گا یا صحیح ہو گا تو ہم اگلے timestamp میں صحیح input دیں گے۔ اور اگر ہمارا ماڈل غلطی کرتا رہتا ہے پھر بھی ہم صحیح input دیتے رہیں گے۔ اور sentence جب ختم ہوتا ہے تو آخر میں input end دیتے ہیں۔ اور پھر ہم دعا لگا لیتے ہیں۔

Date: ___/___/2025 M T W T F S

Steps



Improvements :-

i) Embedding → (Bag of words, tfidf, Word2Vec]
↳ low-dimensions | dense | sommary of words.

ii) Use Deep LSTM | Multilayer | Stacked LSTM
↳ useful for big paragraphs.
↳ learning capability increase.

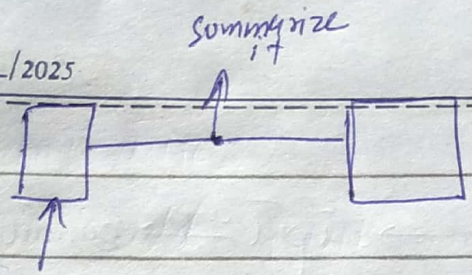
iii) Reverse the input [translation quality increase]

Attention Machanism

Why we need?

سمجھو کہ اگر بہت بڑا paragraph ہو تو اسے اگر آپ کو پہلے پڑھنا ہے اور پھر translate کرنا ہے تو کیسے کرو گے۔ یقیناً آپ نہیں کر پاؤ گے۔ ایسے ہی ہمارے endoder بہت بڑے paragraph کو summarize نہیں کر پاتے۔

Date: __/__/2025 M T W T F S



اور سارا کا سارا جو چیز encoder سے lstm
پر آتا ہے۔ اور problem صرف encoder کے ساتھ
نہیں۔ decoder problem کے ساتھ بھی ہے۔
مثال کے طور پر اگر آپ کو ایک word کو translate کرنا
ہے۔ تو آپ کو پورا paragraph مل رہا ہے۔ لیکن
اس کے لیے تو ایک input ہی کافی ہے۔ تو ہم کیا کریں
گے۔ آپ کو اپنے decoder میں information pass کرنا
پڑے گا۔ کہ timestamp 1 والا سب سے information سب سے
important ہے light کو print کرنے میں

This is called attention mechanism.

آپ اس میں دو input کی بجائے تین input pass کریں گے

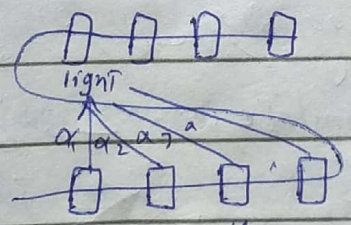
$(Y_1, S1) \Rightarrow [C_i, Y_{i-1}, S_{i-1}]$

$C_i \rightarrow$ attention input

$C_i \rightarrow$ vector / scalar / matrix

C_i دیکھتا کیسا ہے

تو یہ ایک vector ہوگا۔ کیونکہ ہم hidden state کو ڈال رہے ہیں۔
تو اس کی dimension کیا ہوگی۔
تو ہم sum weighted نکالیں گے



$C_2 = \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$

$C_i = \sum \alpha_{ij} h_j$

جتنی زیادہ کسی کے α کی value ہوگی۔ h-state زیادہ
اہم ہوگا۔

Date: ___/___/2025

Now how to calculate $\alpha_{21} \Rightarrow \alpha_{ij}$

$\alpha_{21} \longrightarrow h_j$, $\longrightarrow S_{i-1}$ $\quad \alpha_{ij} = f(h_j, S_{i-1})$

$C_i = \sum_{i=1}^{n} \alpha_{ji} h_j$ $\quad$ what function use

تو ہم کوئی بھی function لگا سکتے ہیں پر ہم
layer ANN لگائیں گے اور اس میں h_j اور S_{i-1} کو input
as a input دیں گے۔ تو اس میں α_{ij} مل جائے گا۔

Bahdanau Attension Vs Luong Attension.

یہ دونوں technique α کو calculate کرنے کے لیے
استعمال کی جاتی ہیں۔

Bahdanau attention :-

α کس پر depend کرتا ہے۔ h_1 کی اور S_0 پر۔
previous ended state

$$\alpha_{11} = f(h_1, S_0)$$

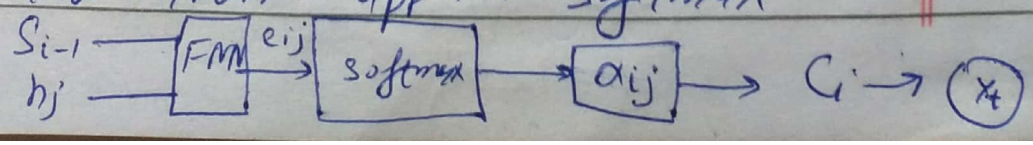
ابھی کا context سمجھتے ہوئے S_1 کا output ان
چار دن hidden layers میں سے کس پر depend کرتا ہے

$$\alpha_{ij} = f(h_j, S_{i-1})$$

allinment score is a mathematical funct
آپ کون سے func. استعمال کریں گے۔ جو
feed forward Ne کا استعمال کر دیں گے۔ آپ NN بنا دیں گے۔

input = h_j, S_{i-1}

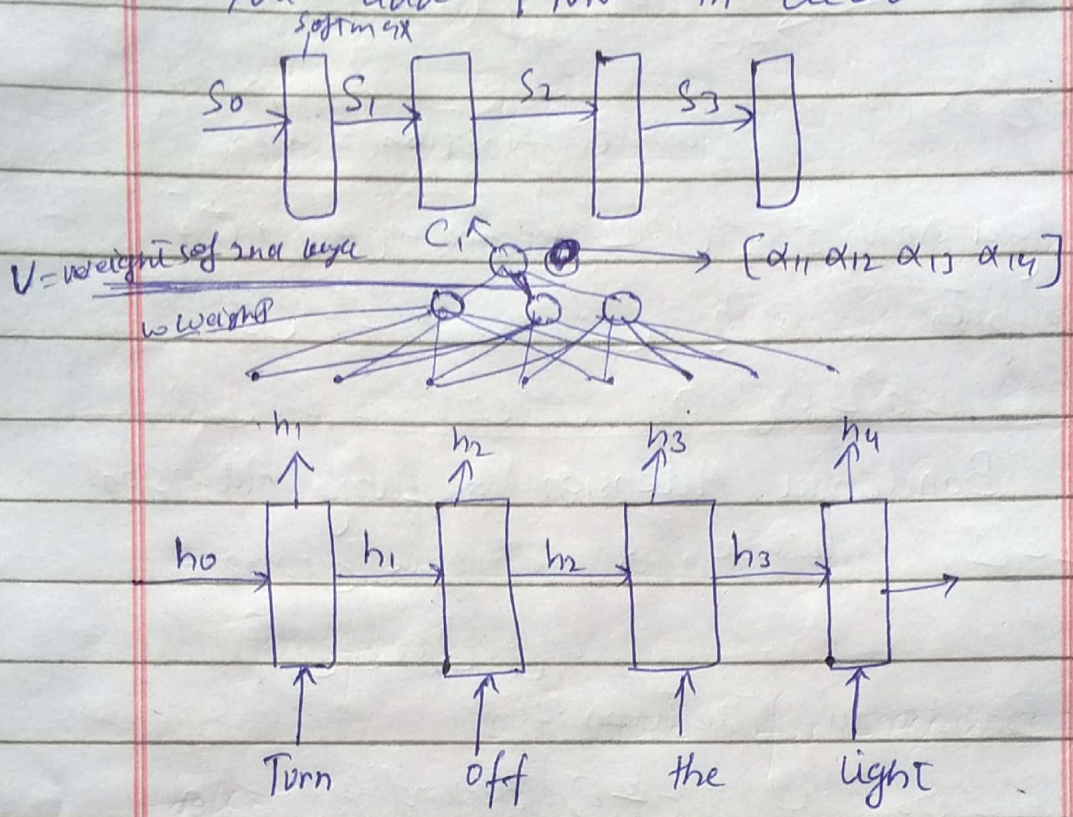
and then applied softmax



Date: ___/___/2025 M T W T F S

Architecture

You add FNN in decoder.



$$C_2 = \alpha_{21} h_1 + \alpha_{22} h_2 + \alpha_{23} h_3 + \alpha_{24} h_5$$

$$\begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} & h_{11} & h_{12} & h_{13} & h_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} & h_{21} & h_{22} & h_{23} & h_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} & h_{31} & h_{32} & & \\ S_{41} & S_{42} & S_{43} & S_{44} & h_{41} & h_{42} & & \end{bmatrix}$$

$$C_{ij} = \sum_{\alpha_{ji}} \alpha_{ji} h_j$$

$$\alpha_{ij} = \frac{e^{e_{ij}}}{\sum_{i=1}^{n} \exp(e_{ji})}$$

$$e_{ij} = \tanh(W[S_{i-j}, h_j] + b$$

$$e_{ij} = V(e_{ij})$$

Date: __/__/2025 (M) (T) (W) (T) (F) (S)

Luong Attention

Goal: We want to calculate attention score for every timestamp of decoder we have to find that which timestamp of encoder is most important.

$$c_i = \sum \alpha_{ij} h_j$$

$$\alpha_{ij} = f(s_{i-1}, h_j)$$

feed forward NN. previous encoder state

but here you depend on current state.

$$\alpha_{ij} = f(s_i, h_j)$$

تو یہ کیا کیا کہ جس feed forward NN کا استعمال کرنا کی جائے
s_i اور h_j کا dot product نکال لیں۔

because parameter added and work slow.

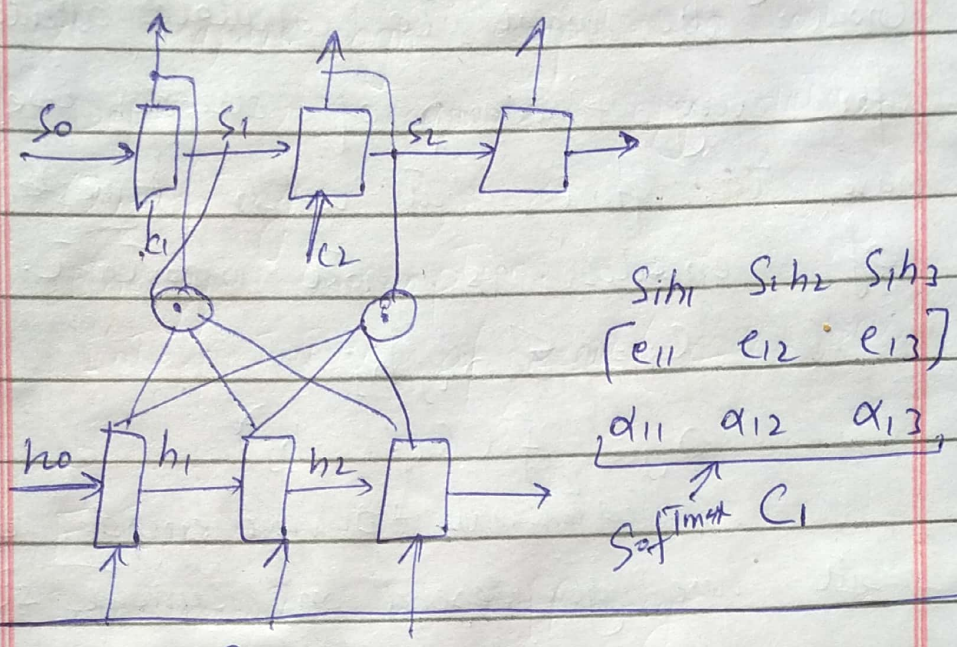
$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \end{bmatrix}$$

s_i h_j

Logic behid s_i → current use:
آپ ذیادہ dynamically اپنے output کو adjust کر سکتے ہیں

Date: ___/___/2025

Architecture:-



Transformer

It is build for seqq-2-seqq task.

- You can't use LSTM. You use self-attention.

It is developed in 2017 in paper "Attention is all you need."

- Impact of Transformer:-
  - Revolution in NLP:-
  - Dessovering AI → Bert, → GPT
  - Multimodal capability
    - ↳ Images
    - ↳ Speech
    - ↳ Text

Date: __/__/2025 M T W T F S

- Acceleration of Gen AI:-
  ↳ GANs (Images) ↳ Revolutionized text generat

- Unification of Deep Learning:-

— The Timeline:-

2000 - 2014 - RNN / LSTM's

2014 → Attention

2017 → Transformers

2018 → BERT / GPT (Transformer Learning)

2018-2020 → Vision Transformer / Alphafold 2

2021 → Gen AI

2022 - ChatGPT / Stable Diffusion.

— Advantage:-

i) Scalability (You can Train on huge data).

ii) Transfer Learning (fine Tunning on custom task).

iii) Multimodel Input / Output :- / Image - Image - Speech

iv) Flexible Architecture
  ↳ Encoder Only (BERT) ↳ decoder only (GPT)

v) Ecosystem:-
  Bawal ergia hy.
  ↳ huggingface ↳ videos / blogs.

vi) Easy Integerated with AI techniques:-

- GANS + Transformers
- RL + Transformer

Date: __/__/2025 M T W T F S

Disadvantage:-

i) It require high computation cost
ii) Data (It requires big data).
iii) Overfiting
iv) Energy consumption.
v) Interpretation of Result (اندر کیا چل رہا ہے)
vi) bias

Future:-

i) Improvement in Efficiency
ii) Multimodel capability
iii) Responsible
iv) Domain Specific Task (Doctor, Teacher) doctor GPT
v) Multi-Lingual
vi) Interperopability

Self Attention

What is self-attention

ایک ایسی technique ہے جو آپ کو word embedding سے
create-smart contextual embed کر سکتی ہے۔

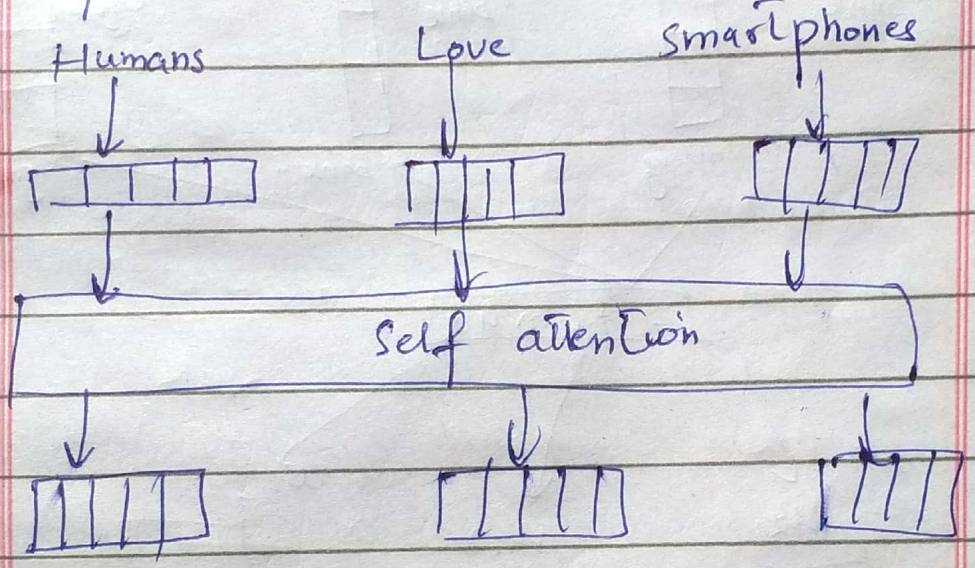
Contextual embedding :-

Company apple, fruit apple مطلب context کو ڈھونڈ سکتے ہیں۔

[0.9, 0.1], [0.1, 0.9]

Date: __/__/2025 M T W T F S

"Self attention is a mechanism that can take static embedding as input and can generate good contextual embedding which are much better used in any nlp techniques".



ہماری embedd جو ہیں وہ semantic meaning کو capture
کر لیتی ہیں جبکہ وہ contextual meaning capture
نہیں کر سکتی ہیں۔ تو ہم principal میں کیا کریں گے۔

money bank grows

اگر مجھے bank کا meaning نکالنا ہے تو یہ کیسا ہوگا۔

$e_{bank} = 0.1\ money + 0.7\ bank + 0.2\ grows$

$e_{grow} = 0.25\ money + 0.1\ money + 0.65\ grows$

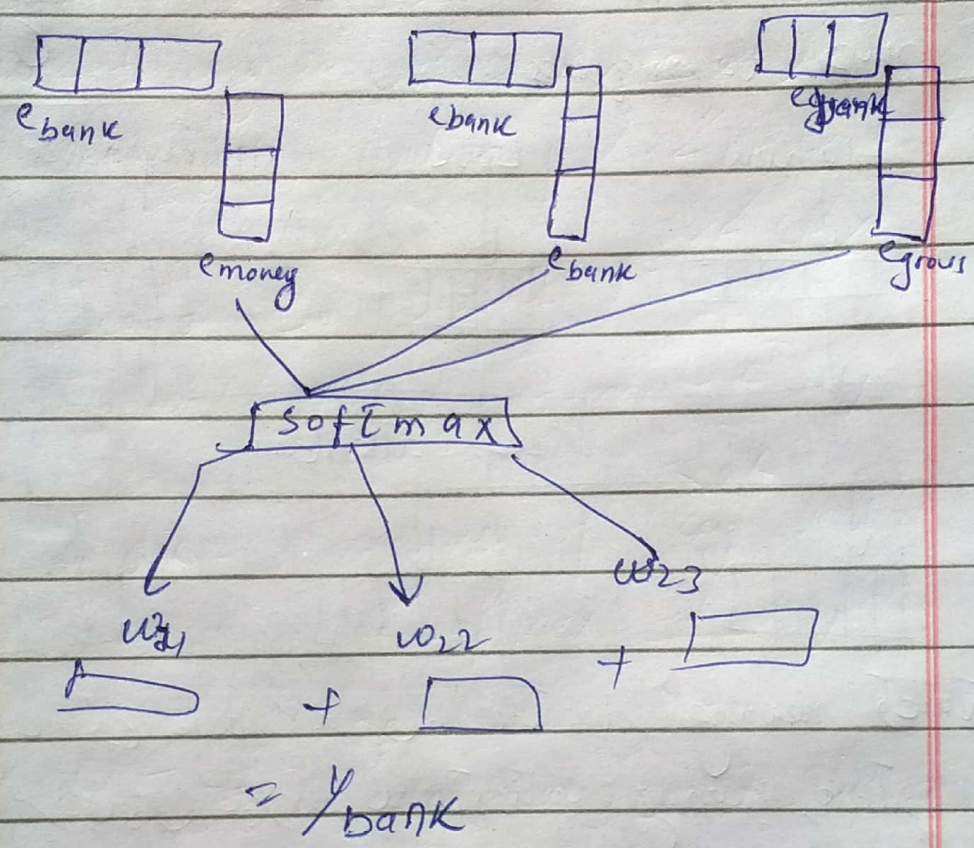
یہ نمبر کہاں سے آئیں گے اس کیلئے ہم

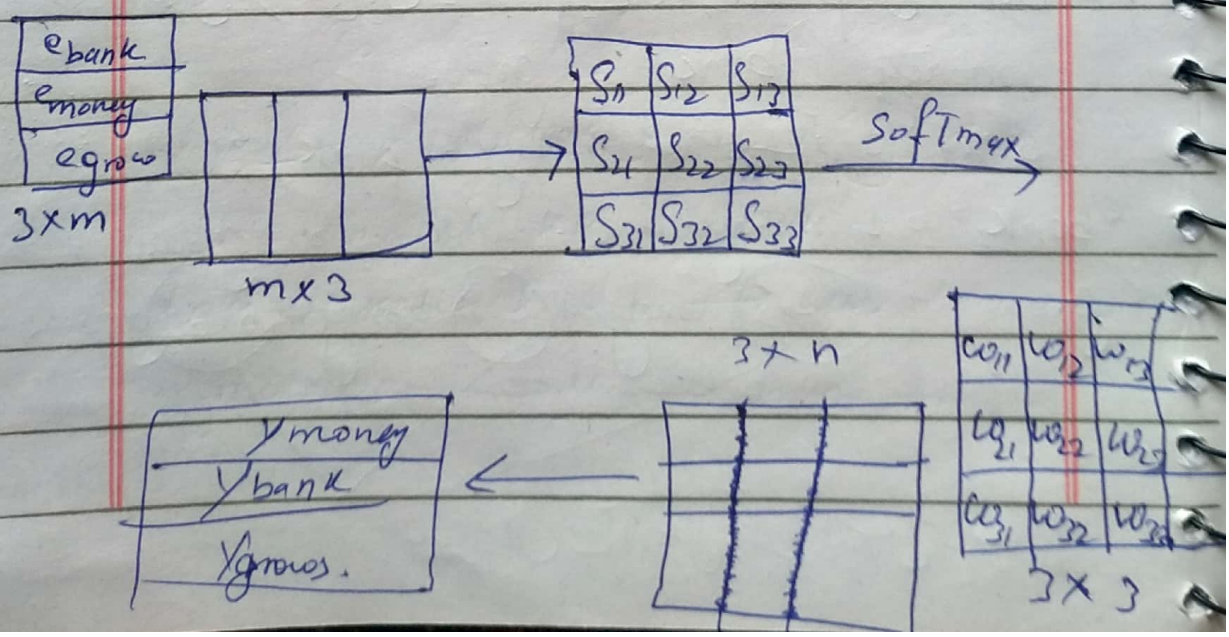
$e_{money} = 0.7\,money + 0.1\,bank$

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

$e_{bank} \cdot e_{money}^T$ → $money \cdot money^T$ → dot-product

$$e_{bank} = [e_{bank} \cdot e_{money}^T]\, e_{money} + [e_{bank} \cdot e_{bank}^T]\, bank + [e_{bank} \cdot e_{grous}^T]\, grous$$





Date: __/__/2025 (M) (T) (W) (T) (F) (S)

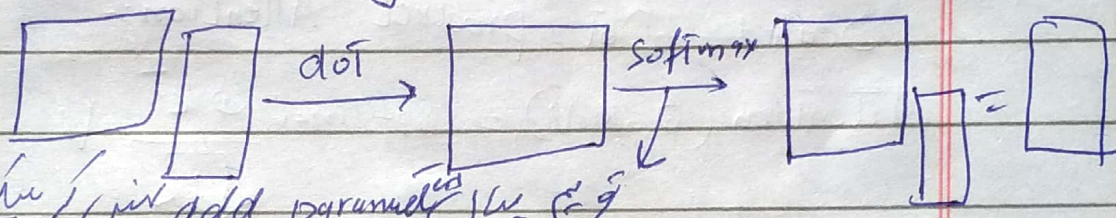
پھر اس میں بھی ایک مسئلہ ہے learning rate نہیں ہے۔
no lr = no learning ہو

کیونکہ contextual embedding تو دو طریقے کے ہوتے ہیں۔

① general ② task specific

اور ہم نے یہ realize کیا کہ specific-task result better دیں گی۔ تو ہمارا جو simple self-model ہے وہ صرف general ہی دے رہا تھا۔

"تو ہمیں اب learning-params ڈالنے ہونگے۔



تو ہم یہاں پر add parameters نہیں کر سکتے۔

پہلے آپ اپنے setup کو سمجھو۔ کہ actually ہو کیا رہا ہے۔

آپ ایک vector دوسرے vector سے پوچھتے ہیں کہ بتاؤ تم میرے لئے کتنے suitable ہو تو اگر زیادہ match کرتا ہے۔ نمبر بڑا ہوگا۔ اسی طرح ہم دوسرے نمبر سے پوچھیں گے۔

تو پھر ایک query vector کا کام بھی کر رہا ہے۔ اور جانے کا کام بھی کر رہا اور value کا کام بھی کر رہا ہے۔

پھر actually میں ایسا نہیں ہونا چاہیے، تو ہمیں کیا کرنا ہوگا۔ ہمیں اپنے پھر ایک vector کو تین اور vector میں transport کر دینا ہے۔ bank-key, bank-value, bank-value

تو ہم سمجھیں گے۔ shadi.com کے حساب سے۔

تو ہم ایک vector سے تین نئے vector نکالیں گے۔ تو اس کے بعد parallely
treat کیا جا سکتا ہے۔

Date: ___/___/2025 M T W T F S

تو ہم کریں گے جہاں ہم embedding لگا رکھی ہیں۔
تو اس جگہ query.vector ، key.v ، value.v بھیجیں گے
تو ہم یہ تین vector کیسے نکالیں گے اس کے لیے ہم
with some other vector linear Transform کریں گے۔
تو اس کے اندر number کون سے ہوں گے کیسے آئیں گے۔
اس کو سوچنے کی ضرورت نہیں وہ
time کے training
initialize weight - random ہو جائیں گے۔ تو ہم
کریں گے۔

Scaled Dot product Attention:-

$$\text{Attention}(Q, K, V) = \text{SoftMax}\frac{QK^T}{\sqrt{dk}}V$$

dk = dimensions of key vector

$$= \text{SoftMax}\frac{QK^T}{\sqrt{3}}V$$

تو جب میں نے ان دونوں کا vector نکالا تو اس کی scalar values ہوں گی۔
Q = 3-dimension , K = 3-dimensions
Total = 9 ~~dimensions~~ values.

تو ہم ان کا mean بھی calculate کر سکتے ہیں

Dot product ka Nuqsan:-
low dimentional d.t ⟶ low-variance.
high dimen ⟶ dot ⟶ high variance.

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

	2-dimention	3-dimensional
i	[1,2] • [2,3] = a	[1,2,3], [4,5,6] = a
ii	[ ] [ ] = b	[ ] [ ] = b
iii	[ ] [ ] = c	[ ] [ ] = c
iv	[ ] [ ] = d	[ ] [ ] = d
v	[ ] [ ] = e	[ ] [ ] = e

اس کا variance کم ہوگا کیونکہ dim کم ہیں۔

اس کا variance اس کی نسبت زیادہ ہوگا۔

high-variance is a big problem.

کیونکہ softmax apply کرتے ہیں اور اگر number بڑا ہوگا۔
تو softmax تو e^{number} اس کی probability زیادہ کر دے گا۔
تو جب back prop میں gradient لگائیں گے تو vanishing prob
آ جائے گی اور وہ parameter train ہی نہیں ہوں گے۔
اگر ہم ان کا reduce variance کر پائے تو ٹھیک ہو جائے گا
اور سبھی update ہوں گے۔

how to control variance?:

تو آپ کو اپنے dot product سے نکلی ہوئی
values کو کسی number سے divide کرنا پڑے گا۔
اور وہ number آپ کے vector کی dimension کے equal ہوگا

$d=1 \qquad y = \text{var}X$

$d=2 \qquad \frac{y}{\sqrt{2}} = \frac{2\,\text{var}X}{(\sqrt{2})^2} = \frac{\cancel{2}\,\text{var}X}{\cancel{2}} = \text{var}X$

$d=3 \qquad \frac{y}{\sqrt{3}} = \frac{\cancel{3}\,\text{var}X}{(\sqrt{\cancel{3}})^2} = \text{var}X$

Date: ___/___/2025 M T W T F S

Geometerical Intuition of : Self - Attention :

Sentence: bank money

embedding: bank money

dot product: W_a W_K W_V

q_{bank} K_{bank} V_{bank}

K_{mon} q_{money} V_{money}

K_{bank} K_{money} K_{bank} K_{bank}

S_{11} S_{12}

Scalling $\frac{QK^T}{\sqrt{d_k}} V$

Softmax

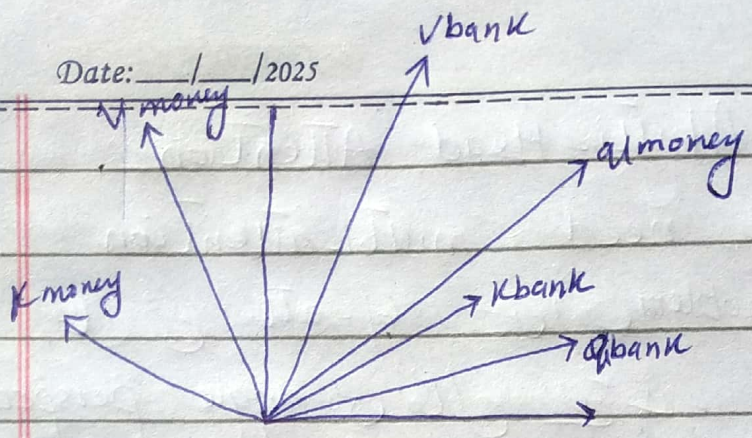
w_{11} w_{12}

$w_{11} \times$ money + $w_{12} \times$ V_{bank}

Y_{bank}

Date: ___/___/2025

M T W T F S



Dot product

$S_{21} = 10$; $S_{22} = 32$

Scalling

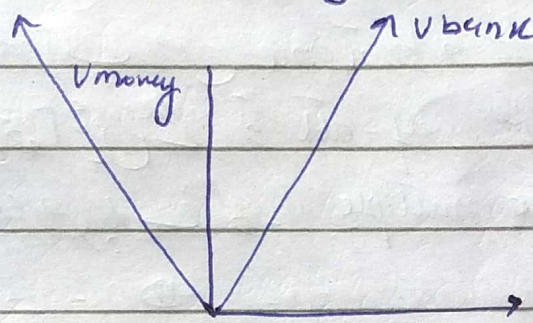
$S'_{21} = \frac{10}{\sqrt{2}} = 7.07$

$S'_{22} = \frac{32}{\sqrt{2}} = 22.6$

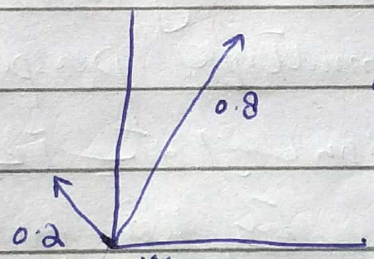
$0.2 \times V_{money} + 0.8 \times V_{bank}$

Sofimax

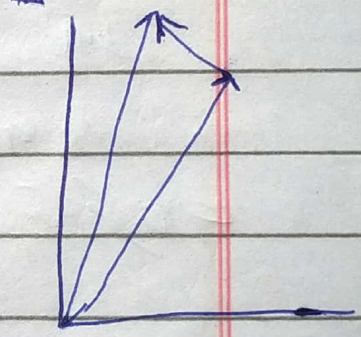
$\omega_{21} = 0.2$, $\omega_{22} = 0.8$

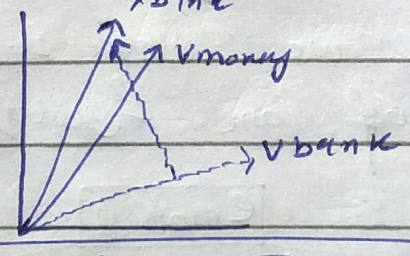


adding 2 vector



0.8 + 0.2





Why Self-Attention called "Self".

Self-attention & Luong Attention is same because mathematical formula is same.

یہ calculate attention کے بیچ میں same-sequence کے
its intra sequence - اس لیے اسے self کہا جاتا ہے

Date: ___/___/2025 M T W T F S

Multi-Head Attention

Why we need multi-attention

ہمارا self-att. میں ایک problem ہے کہ وہ single perspective ہی نکال سکتا ہے۔

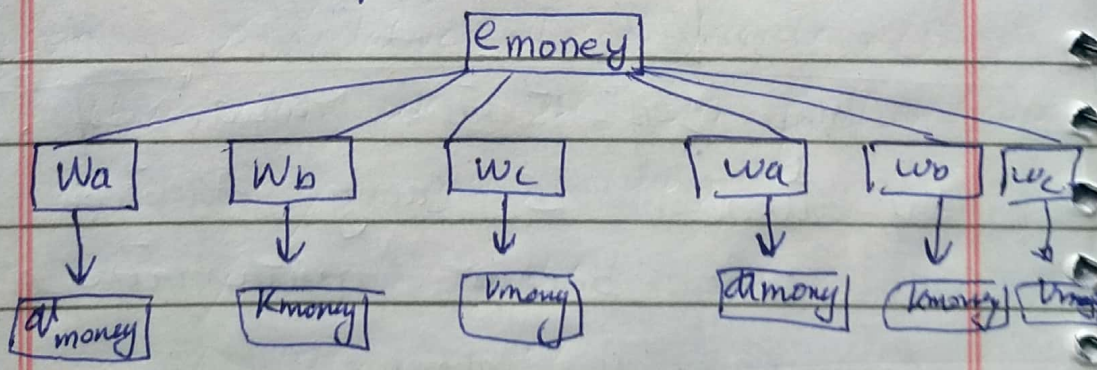
The man saw the astronomer with a Telescope.

تو اگر آپ اس sentence کو پڑھو تو آپ کو دو مطلب meaning سمجھ آئیں گے۔ تو self-att صرف ایک ہی perspective multiple نہیں نکال capture کر پائے گا۔

but in NLP many times there are multiple perspective.

تو اس لیے آپ کو ایسا mechanism تیار کرنا ہے تو multiple-per. نکال سکے۔

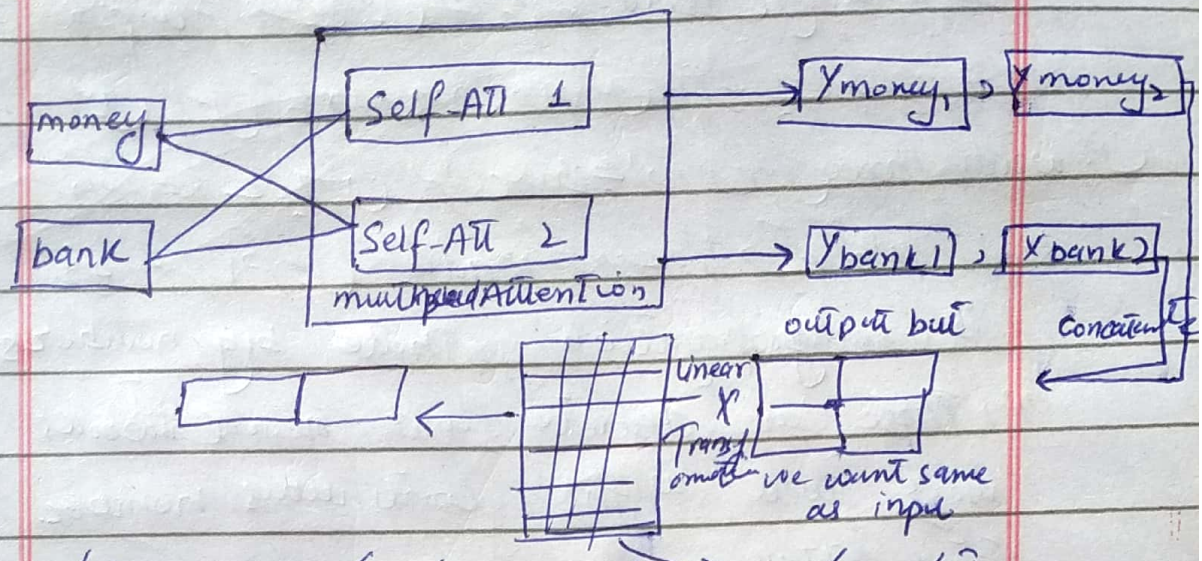
It says we use 2-self-attention instead of one. مطلب metrics کو بڑھا دیں گے



ہر کسی طرح 2 ہم word کے لیے multiple metrics نکالیں گے۔

Date: ___/___/2025

تو اب y-money بھی دو بار generate ہو گا۔



تو اس کو استعمال کر کے ہم دونوں کے درمیان یا تو کوئی ایک یا mixture of output دے دیں گے۔

Positional Encoding:

آپ کو پتہ ہے کہ self-attention execute parallel ہوتا ہے۔ مطلب آپ ایک پورا کا پورا paragraph کو دے سکتے ہو اور ایک ہی ٹائم execute ہو گا۔ جس سے fast training ہو جائے گی۔ مگر اس کا drawback ہے کہ word-order کو نہیں دیکھ پاتا۔ کون سا word پہلے آیا ہے اور کون سا word بعد میں آیا ہے۔

Ali Killed Lion

Lion killed Ali

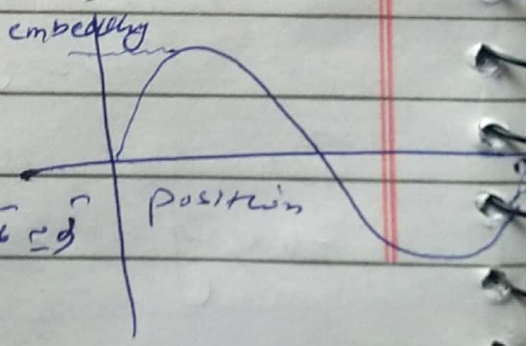
یہ اس کے لیے same ہو جائیں گے۔

This is where positional encoding comes in place.

Date: __/__/2025 M T W T F S

تو ایسے سمجھو کہ ہم بتا رہے ہیں Positional encoding
تو ہم کیا کر سکتے ہیں کہ ہم ہر word کی embedding کے آخر
میں اس کا no# index بنا دیں کہ یہ پہلے نمبر پر ہے
تو اگر ہمارے پاس بڑی book ہے اور اس میں 100000 words ہیں
تو یہ نمبر آخر میں add ہو کر بہت بڑا ہو جائے گا اور
i) backpropagation hate big numbers.
ii) Those are discrete and neural networks
work well with continuous number.
آپ کو بھی یہ position relative میں capture کرنا ہوگا
Nitish killed the Lion.
1 2 3 4
وہ یہ ہیں capture کر پائے گا کہ Nitish اور the
کے درمیان کتنی دوری ہے تو اسے position relative
کہتے ہیں۔

So we use predic function.
The answer is trignometric function
تو ہمیں ایسا value ڈھونڈنا ہے کہ وہ bounded
بھی ہو discrete بھی ہو اور relative بھی ہو۔ e.g. (sin)
How can we use sin function (curve
$y = \sin(\text{position})$
$y = \sin(1) = 0.84$
$y = \sin(2) = 0.87$



تو یہ تمام problem solve ہو جائیں گے

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

But there is one big problem:

کہ ہمیشہ different position ہونی چاہیے - اگر position same ہوگی تو وہ سمجھے گا - کہ دونوں word same position پر ہیں - اور function sin میں یہ بیماری ہے کہ وہ دوبارہ دوبارہ آتا رہتا ہے -

Solution :- You can use two different trignometric function

$y = \sin(\text{position})$
$y = \cos(\text{position})$ > (0.1, 0.5)

تو ہم دو number لیں گے کسی word کو represent کرنے کے لیے اور اگر دوبارہ بھی ایسا آتا ہے تو اور لگا دیں گے

$y = \sin(\text{pos}/2)$
$y = \cos(\text{pos}/2)$

Nitish = [0.1, 0.7, 0.65, 0.54]

embedding کے 60% same vector کے positional encodi

bank embedding = 6 dimen.

bank pos. encodding = 6 dimension

تو پھر آپ کو کیا کرنا ہے آپ ان دونوں vectors کو concatenate کر دیں گے means + کر دیں گے

سوال: تو اس کی بجائے ہم embedding کے آخر میں کیوں نہیں add کرتے pos. enc کو -

جواب: اگر ہم ایسا کریں گے تو double dimension ہو جائیں گی - جس سے training process بڑھ جائیں گے - اور train time بڑھ جائے گا

Date: ___/___/2025 M T W T F S

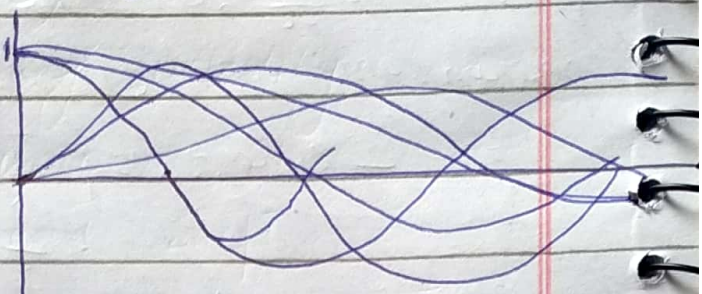
For calculating possitional encoding you use sin & cosine.

$i=0 \to \frac{6}{2}-1 <= i=0 \to \frac{d_{model}}{2}-1$

$i=0 \to 3-1$

$i=0$

Position of word



How To calculate:-

$P.E(pos, 2i) = \sin\left(pos / 10000^{2i/d_{model}}\right)$

$P.E(pos, 2i) = \cos\left(pos / 10000^{2i/d_{model}}\right)$

d_{model} = no. of dimension,

at $pos=0, i=0$

$P.E = (0,0) = \sin\left(0 / 10000^{0/6}\right) = 0$

$P.E(0,0) = \cos\left(0 / 10000^{0/0}\right) = 1$

at $pos=0, i=1$

$P.E(0,2) = \sin\left(0 / 10000^{2/6}\right) = 0$

$P.E(0,2) = \cos\left(0 / 10000^{2/0}\right) = 1$

~~P.E~~ at $pos=0, i=2$

$P.E(0,4) = \sin\left(0 / 10000^{2/3}\right) = 0$

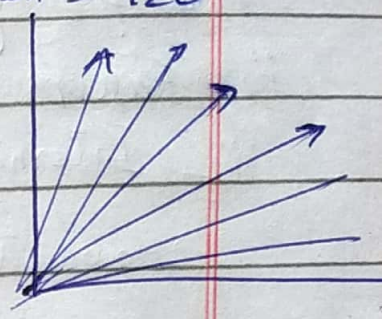
$P.E(0,4) = \cos\left(0 / 10000^{2/3}\right) = 0.71$

~~at pos = 0,~~

Date: __/__/2025

تو اب positional-enc میں ایک ایسی property آ چکی ہے
میں اسے explain کرتا ہے

$w = 50$, $v = 50$, dimen = 128



اگر میں کسی particular.ve پر
Linear-Transf لگاتا ہوں تو اس کا result
کیا ہوگا یہ metrix پر کے لیے ایک جیسا result ہوگا

$V_{10} \xrightarrow{\text{linear}} V_{30}$
$V_{30} \longrightarrow V_{50}$ } delta = 20

تو یہ جو فرق ہے اسے ہم delta لگا رہے ہیں۔
تو اس سے آپ کیا سمجھے کہ جو آگے والا result
"کا وہ پچھلے کے relative ہوگا۔

$V_{10} \longrightarrow V_{20}$ means V_{10} & V_{20} are relative to each other. This is mind blowing.

Normalization

Normalization in deep learning refers to the process of transforming data or model outputs to have specific statistical properties, typically a mean of zero and variance of one.

Date: __/__/2025 M T W T F S

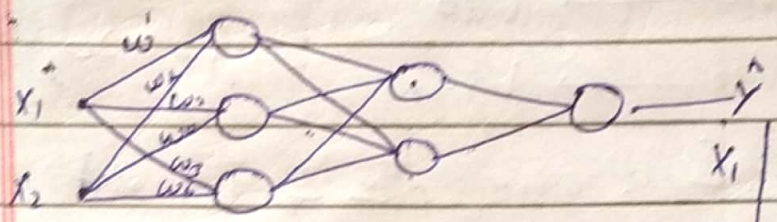
کیا جاتا ہے: activation کے پہلے input کو بھی normalized کرتے ہیں اور
values کو بھی normalize کرتے ہیں۔

i) Improving Training stability

ii) Faster Convergence

iii) Internal covanent Shift:

How To do Batch Normalization.



$z_1 = w_1x_1 + x_2w_2 + b$

$z_2 = w_2x_3 + w_4x_2 + b$

$z_3 = w_5x_1 + w_6x_2 + b$

x_1	x_2	z_1	z_2	z_3
5	4	0.14 ~~3~~	4	4
3	4	~~4~~ 0.14 0.36	3	5
7	6	~~7~~ 0.2	5	8
8	7	~~6~~	2	1

std $= z = \frac{x_1 - \mu}{\sigma} = \frac{3-5}{6} = 0.14$

add γ & β $\quad z_1' = 0.14\gamma + \beta \quad \because \gamma = 1, \because \beta = 0$

activation کے لئے z_2 کی طرح $z_1 = 0.14$

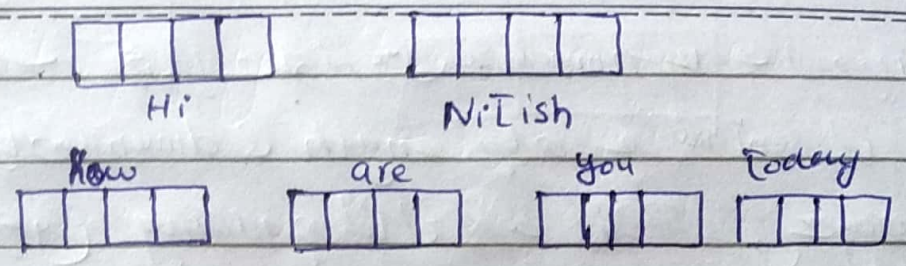
Why don't Use Batch Norm in Transf

self-attent کے ساتھ اچھے سے کام نہیں کر پاتا۔ Batch-Norm

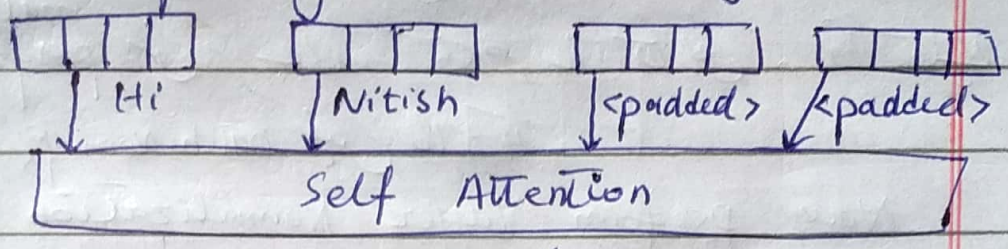
means don't work with sequence data

ہم ایک ساتھ دو sentence بھیج سکتے ہیں ایسا ہی ہے تو ہم
2-sentence بھیجے گا۔ وہ بھی batch کی مستقل نہیں

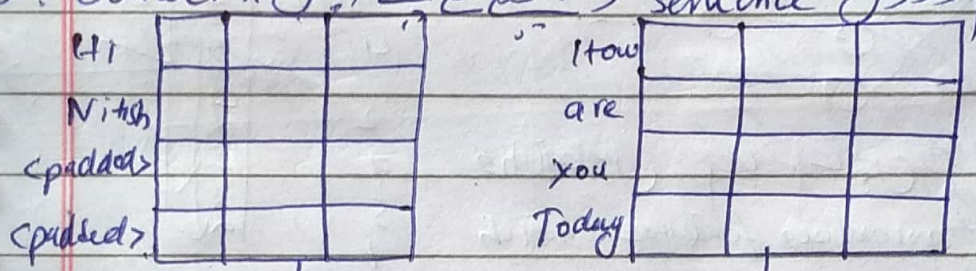
Date: ___/___/2025 M T W T F S



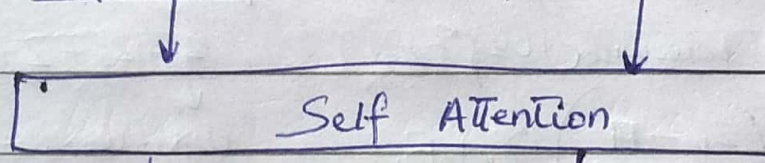
Add padding because length is not equal.

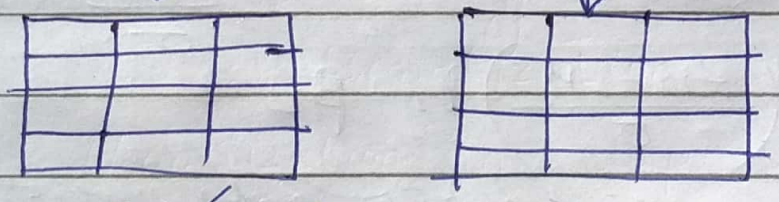


ہم دونوں sentence کو بھیجنے سے پہلے انہیں matrix میں تبدیل کرتے ہیں



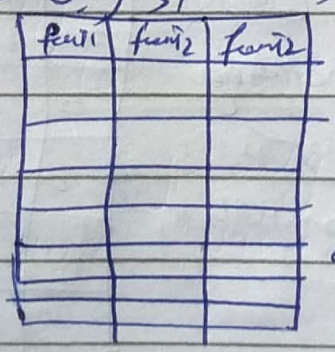
قوی ایک single tensor ہے تو مطلب - sentence ہے batch single





اب ہم Normalization batch perform کریں گے۔

ہم ان دونوں vector کو static کر دیں گے جو ڈر کر دیں گے



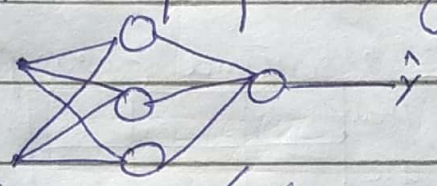
اب ہم سب کو ہم کے ساتھ batch
مطلب Perform Norm. کریں گے
normalize کریں گے mean نکالیں گے
σ نکالیں گے

w, γ, β

Date: ___/___/2025 M T W T F S

اب سمجھو بات کو کہ ہمیں normalize کرنا ہے اور وہاں
یا تو dimention کی length زیادہ ہے یا sentence کی۔ تو
ہمیں padding لگانی پڑے گی۔ جس سے unnecessary
zero آئیں گے۔ جس سے μ جو ہے وہ real-data کو
~~perform~~ match نہیں کرے گی۔ اسی طرح σ بھی۔ تو اس لیے
ہم batch-Norm نہیں لگا سکتے۔

Now perform layer Normalization



f_1	f_2	z_1	z_2	z_3
4	⋮	~~8~~	⋮	⋮
2	⋮	4	⋮	⋮
7	⋮	7	⋮	⋮
5	⋮	2	⋮	⋮

تو batch میں ہم کیا کرتے ہیں ہم ایک پورے
batch کے features کو normalize کرتے ہیں۔
اور ہم یہاں row-wise norm کریں گے

$$z_{11} = \left(\frac{z_{11} - \mu}{\sigma}\right)\gamma_1 + \beta_1$$

$$z_{21} = \left(\frac{z_{21} - \mu}{\sigma}\right)\gamma_2 + \beta_2$$

$$z_1 + z_2 + z_3$$

$$w_1x_1 + w_2x_2 + b$$

$$\frac{5 - \mu}{\sigma} = 0.4\gamma + \beta$$

اسی میں اگر دیکھو تو سمجھ آتی ہے کیوں ہم layer-Norm لگاتے ہیں۔

اس میں آپ اگر دیکھو کہ اگر ہم
padding والی rows کو normalize کریں گے تو وہ
دوسروں کو effect نہیں کر رہی اور value
zero سے normalize ہو رہی ہے۔

0.1	2.5	8	4.1	
7	1	2.5	0.5	
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

Transformer

ہم transformer کو word-simple میں سمجھنے کی کوشش کرتے ہیں

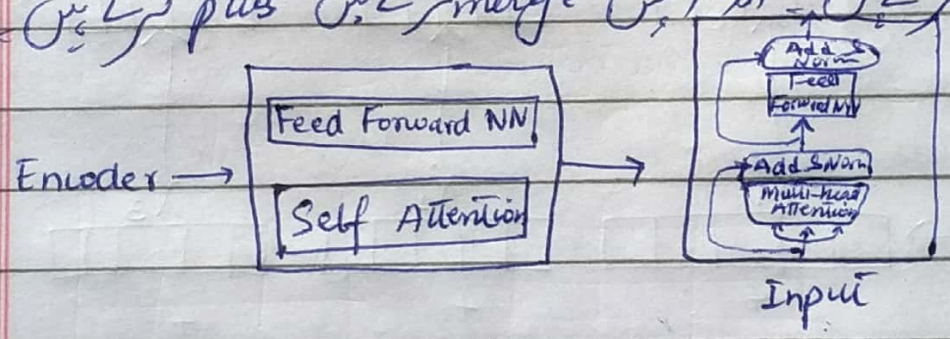
- Transformer has Two parts
  - Encoder
  - Decoder.

یہ encoder اور decoder ایک ایک نہیں بلکہ 6 ہیں۔

اور سارے ہی ایک جیسے ہیں۔

- Encoder has equal two parts:
  i) Self attion ii) Feed forward NN

ہم جیسے اپنا sentence encoder میں ڈالنے سے پہلے
جس input ڈالنا ہو گا۔ اس میں tokenization
کرتے ہیں۔ Embedding نکالتے ہیں۔ پھر pos.end
کرتے ہیں۔ اور انہیں merge کرتے ہیں plus کرتے ہیں

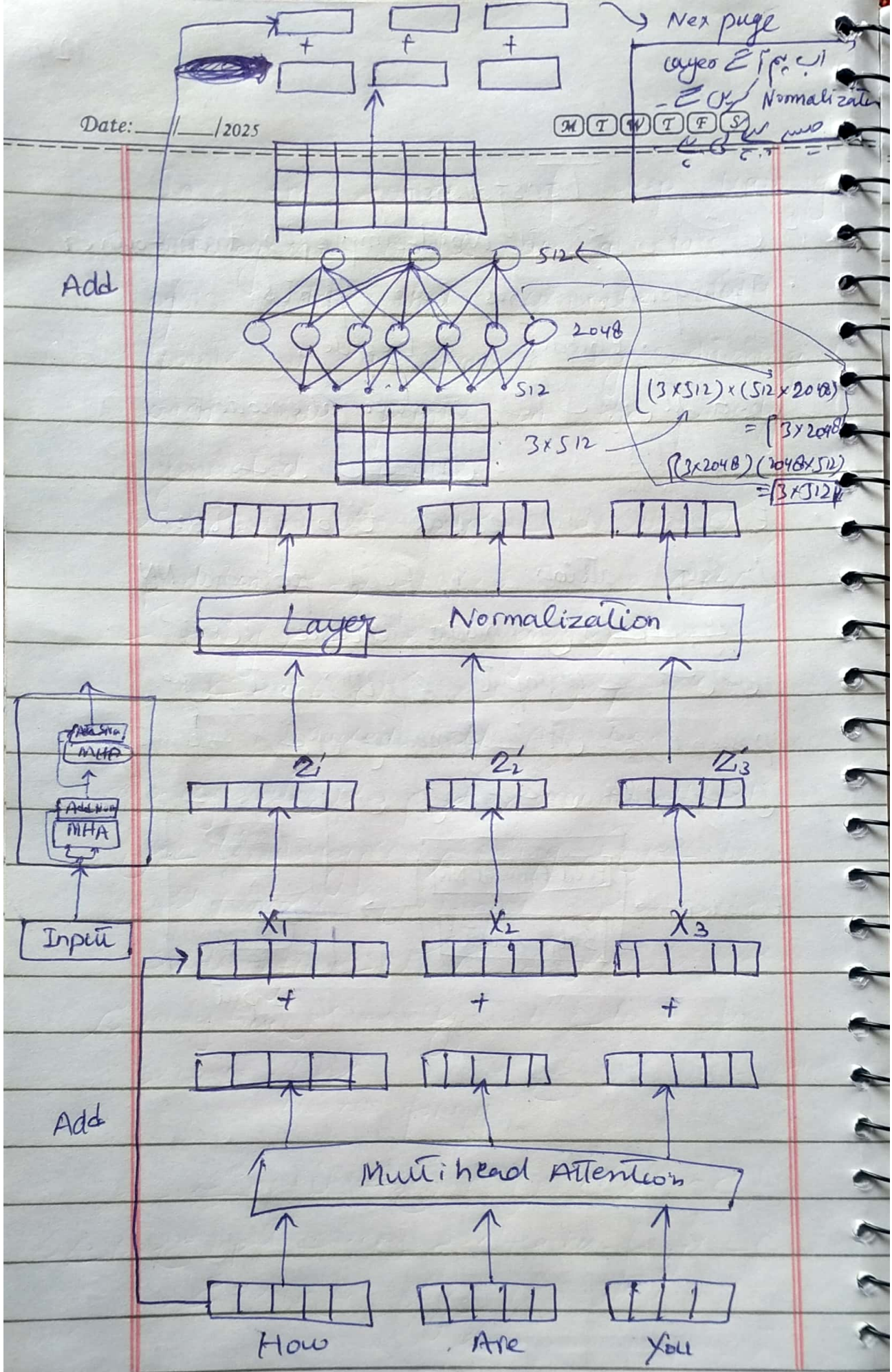


Date: ___/___/2025

M T W T F S

Next page

اب ہم آگے layer کی Normalization
کریں گے



Date: __/__/2025 (M) (T) (W) (T) (F) (S)

اچھا Total ہمارے پاس 6 encoder اور ہر کے پاس
weights ہیں - ہر اس کے اندر جو Multi-head کے weights ہیں
پھر Feed Forward NN کے weights ہیں - وہ مختلف ہوں گے
مگر function اندر same ہوں گے -

i) Questions: Why use residual connections?

اس کی original-explanation کو paper میں نہیں دیا گیا -
کہ ہم نے یہ vector-of-addition کیوں کیا؟
A provide stable training in DeepNN.

2. Why use a FFNN?

کیونکہ اس کے ذریعے ہم آپ تھوڑی سی
show non-linearity کروا رہے ہیں -

So we use relu to capture non-linearity.

3. Why we use more than one encoder block?

Language کو صحیح طریقے سے سمجھ پانے کے لیے ہم زیادہ
encoder لگاتے ہیں - اس کا جواب simple ہے - جتنے زیادہ
encoders ہوں گے آپ اتنا زیادہ in-depth جا سکو گے -

Masked Self-Attention

1. Autoregressive model:-

The transformer decoder is autoregressive at inference time and non-autoregressive at training time.

TF decoder

Prediction Time ← inference → Auto regressive

Training → Non-autoregressive

Date: __/__/2025 M T W T F S

In the context of deep learning autoregressive model are a class of models that generate data points in a sequence by conditioning each new points on the priously generated points.

اس کا مطلب یہ ہے کہ ہر output پچھلے output پر انحصار کرتی ہے۔ کہ پچھلا لفظ کیا تھا۔ تو یہ صرف prediction کے time پر autoregressive ہوتا ہے پر traing کے time پر نہیں ہوتا ہے۔ اس کا وجہ ہے۔

Masked-Self-Attention

اچھا ایسا assume کرو کہ ہم دونوں جگہ auto-regressive تھا رہے ہیں کہ traing پر بھی ہم auto-reg لگا رہے ہیں۔ کہ ایک ایک کر کے input آئیں گے encoder اور decoder میں جاتے رہیں گے۔ پھر اس سے training time بہت slow ہو جائے گا۔ ہم actually training time پر میں auto-regress نہیں کر سکتے because of teacher forcing.

عملی طور پر مجبوری نہیں کہ ہم پچھلے والا input بھیجیں۔ ہم نے تو teacher forcing کی وجہ سے درست input بھیج دیا ہے۔ جس تو data دیا گیا ہے وہ کا ہم وہی لگا دیں گے۔ تو parallely process ہو گا۔

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

تو اس سے بھی ہم ایک بہت بڑی problem میں پھنس جاتے ہیں۔ کہ training کے وقت کچھ اور کیا اور testing کے وقت کچھ اور کر دیا۔ training میں تو آپ بتا رہے ہوتے کہ یہ کام ایسے ہے پھر testing میں تو ایسے نہیں ہوتا۔

مثال کے طور پر جب ہم machine Translation میں پہلا input دے رہے ہوتے ہیں۔ تو اس کی embedding کہاں سے نکلتی ہے دوسرے دو لفظ ہیں۔ 0.7bank + 0.25river

تو ابھی سے تو دوسرے word کا پتہ ہی نہیں ہے تو یہ کیسے ہوگا۔ تو آپ training کے وقت data linkage کر رہے ہیں۔ تو آپ کے پاس extra time کے training

information ہے پر testing time اس کے پاس نہیں ہوگی۔

Solution = Masked - Multihead Attention

پھر کام سے problem آ رہی ہے approach autoregressive اپناتے ہیں تو بھی مسئلہ ہے اور نہیں اپناتے تو بھی مسئلہ تو اس کے حل کے لیے ہم کو پیچھے جانا پڑے گا۔

جہاں ہم self-att کی مدد سے contextual emb calculate کرتے ہیں۔

$$V_{bank} = V_{river} \times W_{11} + V_{head} + W_{22} + V_{bank}$$

یہ دو چیزیں ہمارا future information pass نہیں ہوتی ہمارے پاس نہیں ہے تو ہم ان کی value کو zero کر دیں گے وہ کیسی ہوگی اس کے لیے ہم ایک extra step لیں گے۔

Date: ___/___/2025

Mask

M T W T F S

$$\begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix} \xrightarrow[\sqrt{d_k}]{\text{scale}} \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix} + \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \text{Softmax}(-\infty) = 0$$

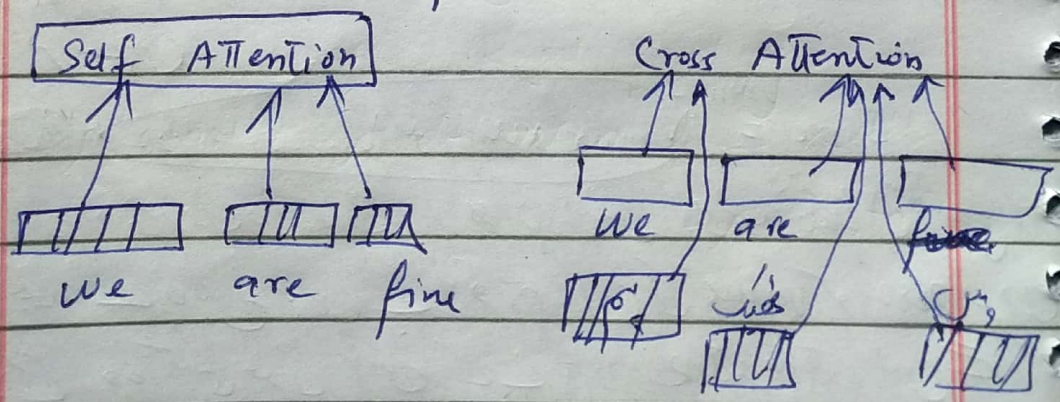
$$\begin{bmatrix} w_{11} & 0 & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

فوائد ہے values 0 ہو گئی ہیں

What is cross Attention:

Cross-attention is a mechanism used in Transformer architecture, particularly in tasks involving sequence-to-sequence data like translation and summarization. It allows a model to force on different parts of an input sequence while generating an output sequence.

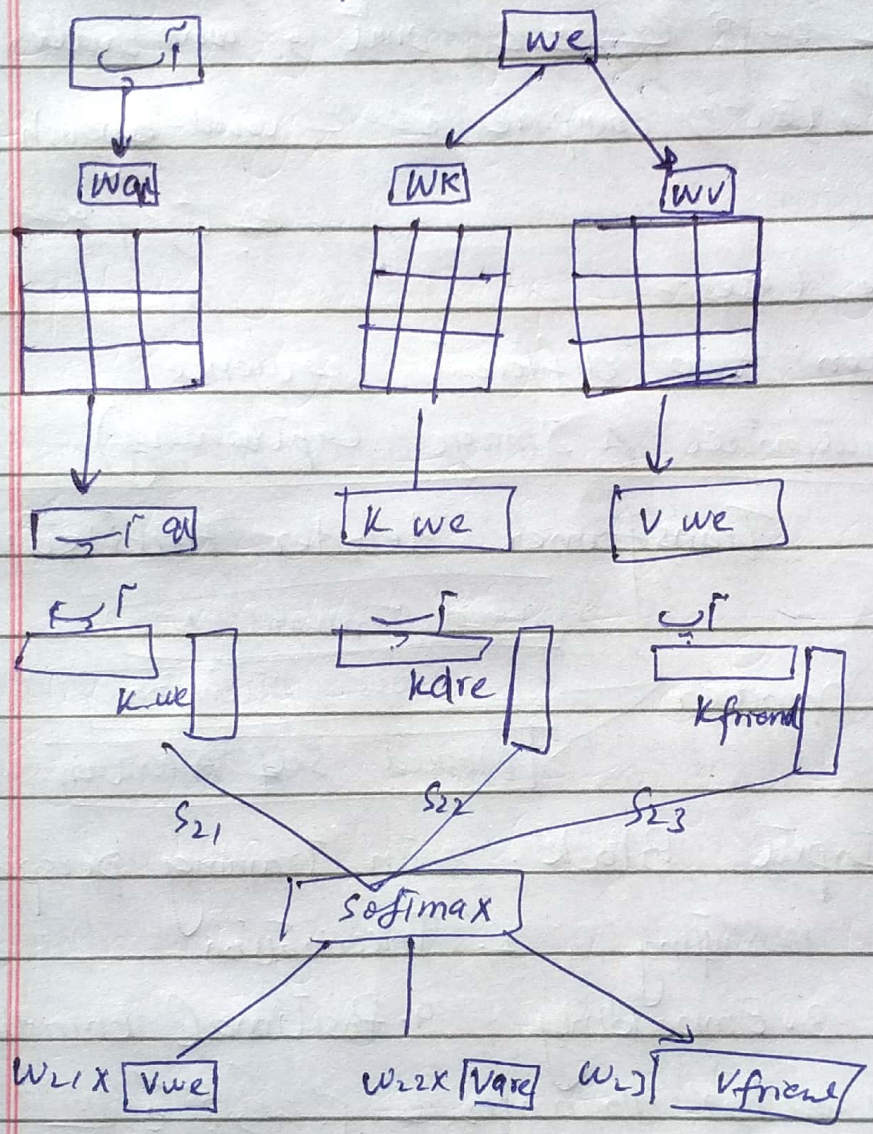
"Cross Attention is conceptually very similar to Self Attention."



Date: ___/___/2025

M T W T F S

Self-Attention میں ایک sequence کا input ہوتا ہے اور
cross-Attention میں 2 sequence کا input ہوتا ہے۔



	how	are	you
آپ	•	•	O
کیسے	O	•	•
ہیں	•	O	O

C.E آپ = 0.5 x e.we + 0.3 x e.are + 0.2 x friend

Date: __/__/2025 M T W T F S

Cross Attention / Friend Barahanda/Luong Attention

اس میں بھی ہم same کام کرتے تھے - ہم ہر اگلے
word order کو print کرنے سے پہلے ہم سارے کے سارے
relationship compare کر دیتا تھا - word-English کے
معلوم کرتا ہے ،

Use_Cases:-

i) When two different sequence.

ii) Multimodel (Image captioning)

Transformer decoder Architecture

Decoder
- Feed Forward NN
- Cross Attention
- Masked Self Attention

① Input Block. In Training perspective

i) Shifting 2. Tokenization

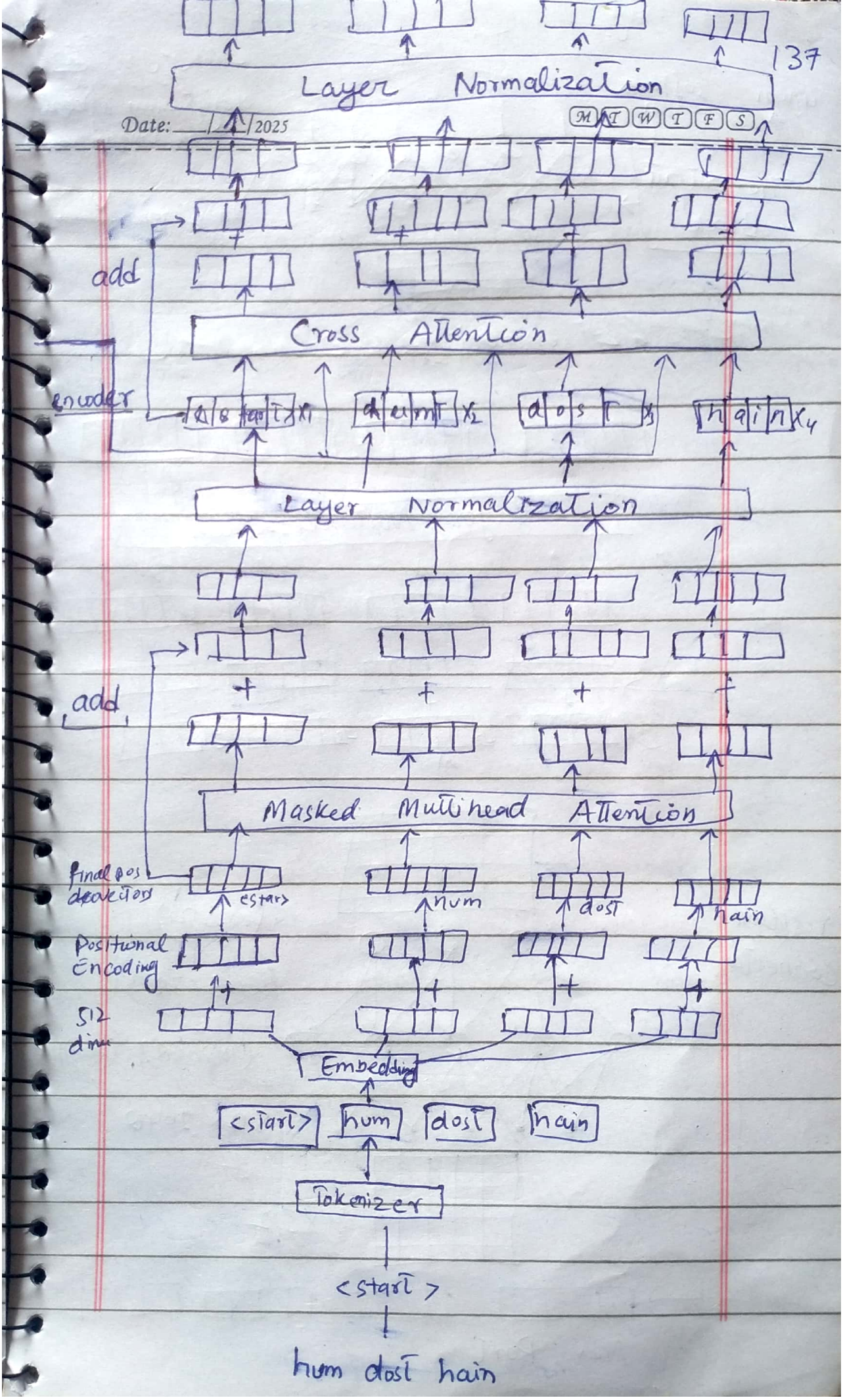
3. Embedding 4. Positional Encoding

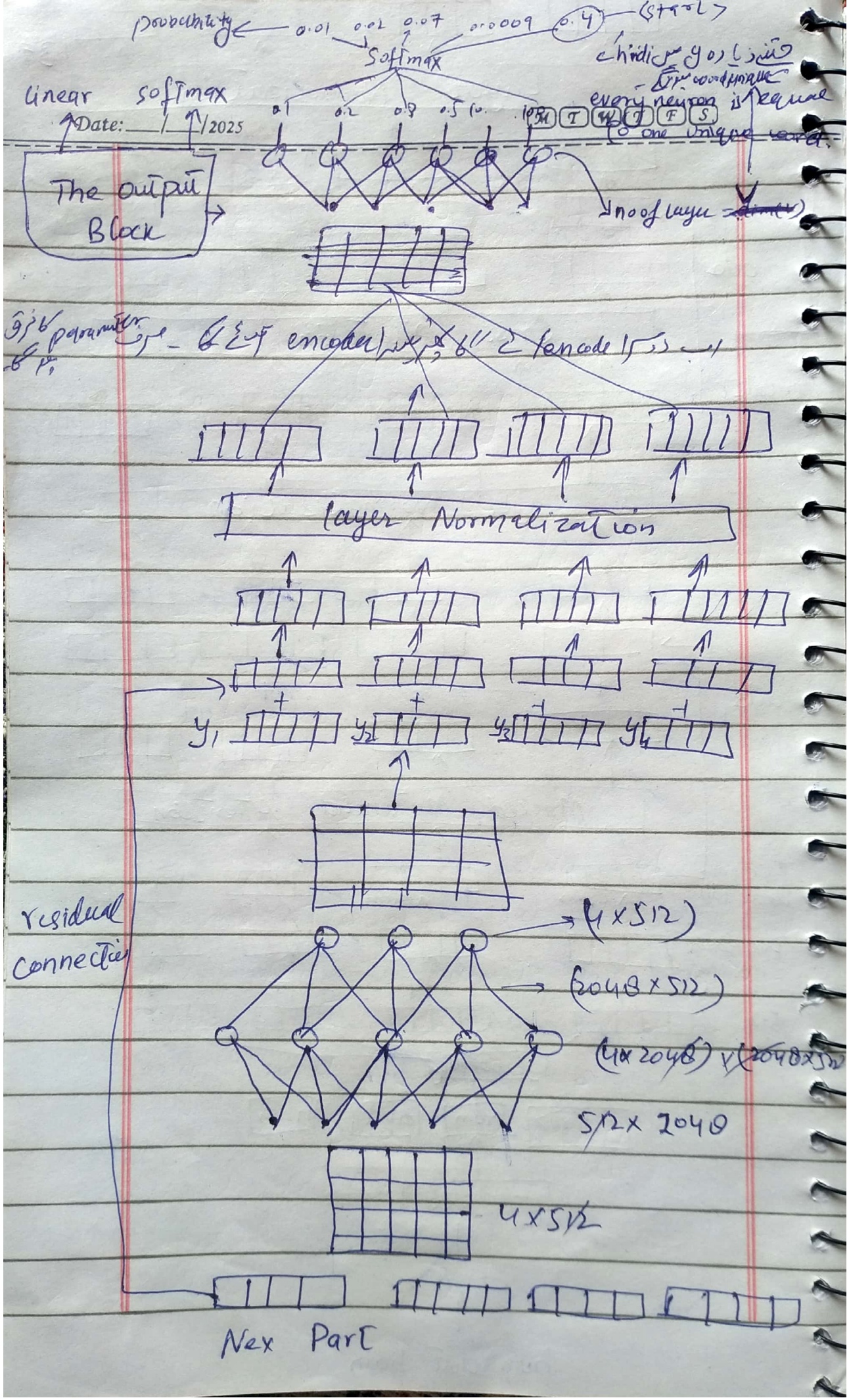
پہلے یہ سمجھو کہ ہم نے encoder والا پورا حصہ کر لیا
ہے اور اب ہمارے پاس hindi والا sentence ہے -

1st - process - Right shifting

مطلب آپ ایک add token کرتا ہو start پر
<start> پر -

Date: __/__/2025


Layer Normalization
add
Cross Attention
encoder
Layer Normalization
add
Masked Multihead Attention
final pos deviations
<start>
hum
dost
hain
Positional Encoding
512 dim
Embedding
<start>
hum
dost
hain
Tokenizer
<start>
hum dost hain


probability
0.01 0.02 0.07 0.0009 0.4
<stop>
Softmax
linear softmax
Date: / /2025
0.1 0.2 0.8 0.5 1.0 100
S M T W T F S
every neuron is equal to one unique word
The output Block
no of layer = dim(v)
layer Normalization
y1 y2 y3 y4
residual connection
(4×512)
(2048×512)
(4×2048) × (2048×512)
512×2048
4×512
Nex Part

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

Transformer Inference

Training ka waqt

encoder تو training کے وقت اور testing کے وقت
same behave کرتا ہے۔ پھر Testing میں decoder
same behave نہیں کرتا ہے۔ اس وقت وہ auto regressive
mode اپناتا ہے۔

3 differences ہیں جو آپ کو پتا ہونا چاہیے۔

(i) آپ کو training پر بھی اور testing پر بھی
masking کرنی پڑے گی۔

(ii) آپ کو cross-attention میں input دیتے ہیں۔
query وہ آپ decoder سے لیتے ہیں اور value
and key جو ہے وہ آپ encoder سے لیتے ہیں۔

(iii) decorder میں پہلا word جائے گا دوسرے
decorder میں پہلا اور دوسرا word جائے گا۔ اور تیسرے میں
تین جائیں گے۔

(iv) اور آخر میں جب آپ output نکالنے کے لیے layer
سے cross کرواتے ہیں تو input ڈالنے
کی بجائے آخری word کو input دیتے ہیں۔
اور پھر softmax لگاتے ہیں اور جس word کی
probability سب سے زیادہ ہوگی وہی دیتا ہوگا۔

Date: ___/___/2025 M T W T F S

Deep-Learning Project

- Neural Machine Translations

Steps:-

(1) Data Loading

(2) Data preprocessing
↳ Tokenization and padding

(3) Train Test Split

(4) Encoder Class. [LSTM/GRU/BiLSTM]

(5) Decoder Class → Attention Mechanism.

(6) Traing Set

(7) Inferrence

(8) Evaluation.

Data Loading

If you have a data of machine Translation like french-english. you have To convert this data into list and into typles of eng and fre sentences.

Data preprocessing:-

i) Unicode To ascii

convert special french words like é, ê = ev.

Date: ___/___/2025 M T W T F S

Remove funtuations
In fre sentence add <start>, <end> at start and end of the sentence.

Tokenization & padding

If your sentence take two words like ali ahmad you add integer-id to these numbers. you need a number max_unique_words = 10,000 ok and remaining words assign <unk>

In padding you need max_sent_length = 70 etc. If your sentence has two words than 68 are assign 0.

Train-Test-Split.

We train our model on fre_padding and our label is eng_padding. first we apply sikit_learn train-test-spite and then we apply tensorflow tensors mean we convert.

Date: ___/___/2025 M T W T F S

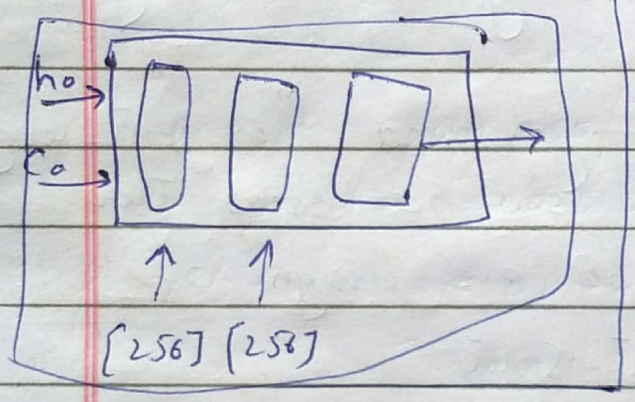
Encoder:

You have an input "[Le fancis shure]" [12, 45, 98, 0, 0, 0] padding

first you do embedding. you need embedding_dim = le = 256 number of vector.

~~[[256], [256], [256]]~~

Now hidden state h_0, cell state C_0 is all contain zeroes.



We are assuming our batch-size is 64. means we train 64 sentences of input max. length of 3, 4, 5, maximum 6

[64, 6]

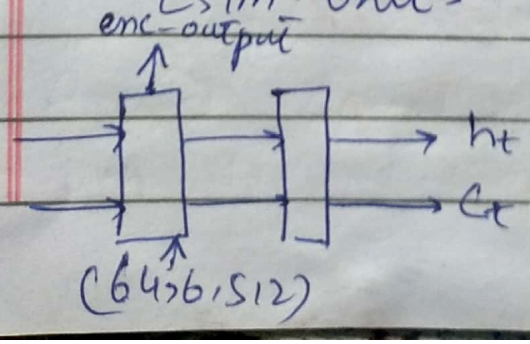
Embedding Layer:-

vocab-size = 10000, embedding_dim = 256

after Emb = (64, 6, 256)

③ Encoder (LSTM)

LSTM-Units = 512



enc-output = (64, 6, 512)

hidden, cell = (64, 512)

Date: __/__/2025 M T W T F S

Why enc-output is (64, 6, 512)
because of lstm units 512.

Decoder Initialization:

dec-input = (64, 512)

Decoder works on Timesteps,
at timestep 1, we give one
word and time = 2, second word.

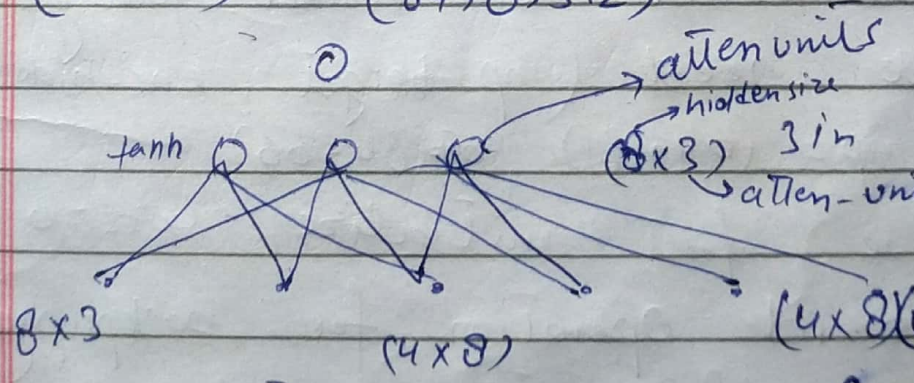
decoder_input = (64, 512)

After Embeding = (64, 1, 256)
Embedding of <start> word.

Now this size and enc-output
gives To attention.

Bahdanau Attention.

[. . . .] [[- -] [-] [-]
[ ] dic_hidden enc_output
[ ] (64, 512) (64, 6, 512)



Wi x enc-output 4 rows 8 colu

Date: ___/___/2025 M T W T F S

attention-weights = (64, 6, 1)

context-vector = atten-weights * hidden-size

= (64, 6, 1)(64, 512)

= (64, 512)

dec-embedding

expand-dim = (64, 1, 512) + (64, 1, 256)

= (64, 1, 768)

Decoder - LSTM.

LSTM-units = 512

dec-output = (64, 1, 512)

Dense.

When we apply softmax we need vocab-size.

output = (64, 1, 10,000)

Training Step

There is a difference between training step and Inference step and that is teacher forcing.

We need (inp, target, enc-hidden)

inp = (64, 71) , out = (64, 66)

we first initialize loss = 0

Record operations for backpropagation with tf.GradientTape() as tape:

Date: ___/___/2025 M T W T F S

tensorflow automatically record every operation inside this block. so it can compute gradient later.

- Encoder Forward Pass:-

encoder_out, hidden, enc_cell = encoder(inp, enc_hid)

enc_output = (64, 71, 512)

enc_hidden = (64, 512),

- Initialize Decoder Hidden:-

dec_hidden = (enc_hidden, enc_cell).

- Prepare decoder input:-

creates a Tensor of <start> Token IDs. =

- Teacher-forcing:-

~~we need from~~ We do not need start in the output That's why:-

for t in range(1, targ.shape[1])

enc_out, dec_input, dec_hidden, یہ decoder کو چلاتا ہے۔

predictions, dec_hidden, attention_weights. یہ 2 ''6'' اور 3 ''6'' دیتا ہے

Teacher-forcing کرتے ہیں۔

dec_input = tf.expand_dims(targ[:, t])

یہاں t کا مطلب ہے timestep کیا ہو گا۔

Date: ___/___/2025 (M) (T) (W) (T) (F) (S)

(4) Loss - Calculation:-

batchlost loss-function [target[:,t], prediction]

loss = loss + batch-loss

اور اس کے بعد میں next_word کے لیے

dec-input = tf.expand_dims(targ[:,t])

= (64,1)

batch-loss = loss / int(tag.shape[1])

(5) Backpropagation

اب ہم variables نکالیں گے سارے سارے۔ اس کے بعد

gradient نکالیں loss کے ذریعے اور پھر update کریں گے۔

variables = encoder.trainable_variable +

decoder.trainable_variables.

= tape.gradient(loss, variable)

= optimize.apply-gradient

(6) Training Loop:-

Epoch = 10

ہم ہر ایک Epoch میں encoder کے hidden اور cell state

total-loss = 0 کو reset کریں گے۔ کر دیں گے۔

~~totalloss~~ we call train-set function which

require inp, out, enc-hidden, inp, out

from train.ds.take (epoch_per_)

Total-loss = total-loss + batch-loss

Date: ___/___/2025 M T W T F S

Inference Step (Evaluation)

ایسا سمجھو نہیں ایک sentence دیا جاتا ہے French کا
اور تم ایک translator ہو۔ تو تم کیا کرو گے پہلے اس word
کو clean کرو گے words-unnecessary کو ہٹا دو گے اس کے
بعد model except. of sent: length دیں
کے training کے وقت کی تھی اور دوسرا طریقہ بھی except نہیں
کرتا تو آپ Tokenizer اور padding کے ذریعے
sent کے ہر word کو numb. assign کرنے کے بعد آپ padding کرو گے۔
training میں ہم نے tf.tensor دیا تھا تو یہ تو ایک number ہے
تو اسے tensor میں کرو گے۔ پھر آپ کے کوئی encoder کی
hidden-state نہیں ہو گی۔ تو پہلے آپ وہ بنا دیں گے 0 دے کر۔
اس کے بعد آپ decoder کا ~~output~~ input بنا دیں گے۔
اور پہلا input سب کو پتا ہے <start> ہوتا ہے پھر اس
کی dimension 1-D ہے decoder پر 2D لیتا ہے تو
2D میں convert کریں گے۔ جس پہلے encoder کو call
کرنے پر enc-h, enc-c مل چکا ہے اور ساتھ میں enc-out بھی
اس کے بعد ہم predict کریں time-step پر ایک word جائے گا۔
t=1 پر پہلا جائے گا۔ ہم decoder کو call کریں گے۔
جو dec-hidden prediction اور atten-weights دے گا۔ اب
prediction ایک tensor طریقے softmax کے ذریعے نکالا ہے تو max نکالیں گے
اور numpy() میں convert کریں گے۔ index سے "tokenizer" کو call
کر کے word نکال لیں گے۔ اور جب predicted word <end>
ہو گا تو result کو return ~~predict~~ کر دیں گے۔

Date: ___/___/2025 M T W T F S

Evaluation (BLEU-Score)

What is BLEU Score?

BLEU (Bilingual Evaluation understudy) ایک metric ہے جو model-machine Translation کی accuracy کو evaluate کرتا ہے۔

by comparing their predicted output (machine Translation) with one more reference Traslation. BLEU measures how close your model translation is to human Translation.

Smothing Function() method

اگر model کا output چھوٹا ہے اور reference بڑا ہے تو BLEU = 0 ہو جائے گا، اس لیے smothing استعمال کرتے ہیں