ML Theory - worksheet 1

01/09/2025

(1.) o/p of last fc layer

$$\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \xrightarrow{\text{apply softmax}}$$

softmax formula

$$S(x) = \frac{e^{x_i}}{\sum_{i=1}^{} e^{x_i}} = \frac{7.39}{7.39 + 2.72 + 1.11}$$

$$\frac{7.39}{11.22}$$

$$\boxed{P(\hat{y}=0) = 0.65}$$

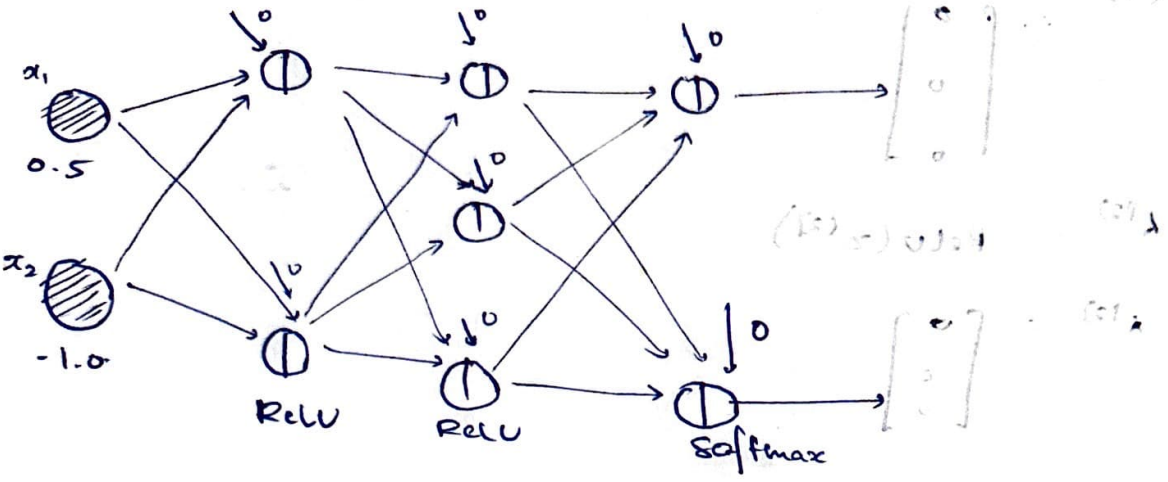
$$P(\hat{y}=1) = \frac{2.72}{11.22}$$

$$\boxed{P(\hat{y}=1) = 0.24}$$

$$P(\hat{y}=2) = \frac{1.11}{11.22}$$

$$\boxed{P(\hat{y}=2) = 0.11}$$

② FFN



$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.742 & 0.794 \\ 0.561 & 1.0 \end{bmatrix}$$

$$W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} \\ w_{31}^{[2]} & w_{32}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.725 & 0.613 \\ 0.416 & 0.092 \\ 0.874 & 0.590 \end{bmatrix}$$

$$W^{[3]} = \begin{bmatrix} w_{11}^{[3]} & w_{12}^{[3]} & w_{13}^{[3]} \\ w_{21}^{[3]} & w_{22}^{[3]} & w_{23}^{[3]} \\ w_{31}^{[3]} & w_{32}^{[3]} & w_{33}^{[3]} \end{bmatrix} = \begin{bmatrix} 0.382 & 0.124 & 0.233 \\ 0.112 & 0.132 & 0.534 \end{bmatrix}$$

$$Z^{[1]} = W^{[1]} X = \begin{bmatrix} 0.742 & 0.794 \\ 0.561 & 1.0 \end{bmatrix} \begin{bmatrix} 0.5 \\ -1.0 \end{bmatrix}$$

$$Z^{[1]} = \begin{bmatrix} -0.423 \\ -0.7195 \end{bmatrix}$$

$$A^{[1]} = \text{Relu}(Z^{[1]}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

(3) (1)

1st Layer

[forward Pass]

x_1	x_2	$z_1^{[1]}$	$a_1^{[1]}$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

(1)

x_1	x_2	$z_2^{[1]}$	$a_2^{[1]}$
0	0	-1	0
0	1	0	0
1	0	0	0
1	1	1	1

O/P Layer

$a_1^{[1]}$	$a_2^{[1]}$	$z^{[2]}$	$a^{[2]}$
0	0	0	0
1	0	1	1
1	0	1	1
1	1	-1	0

final o/p. $\rightarrow \hat{y}$

↓ Loss computation

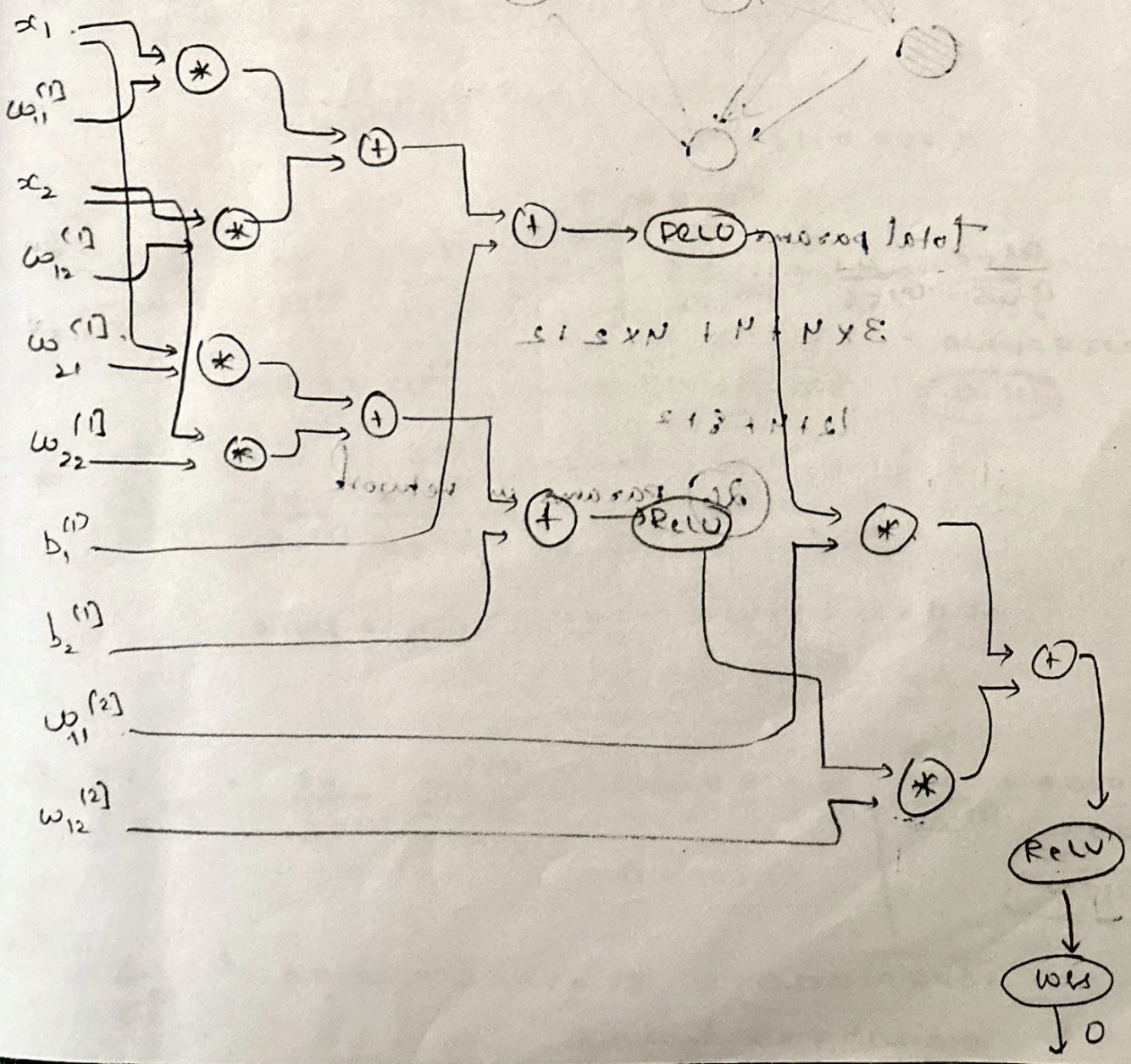
loss formula (MSE loss)

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

No	$\hat{y}$	y	
1	0	0	
2	1	1	
3	1		
4	0	0	0

total loss = 0

(backward pass)



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

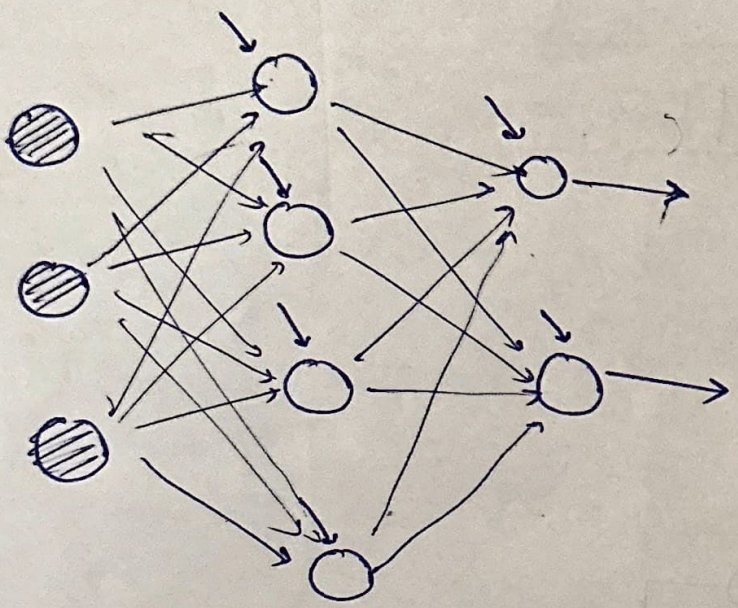
$$= 0.$$

$\frac{\partial L}{\partial y} = 0 \longrightarrow$ the gradients are backpropagated so all gradients could be zero

No update in parameters

Already minimum loss achieved

(9)



Total params

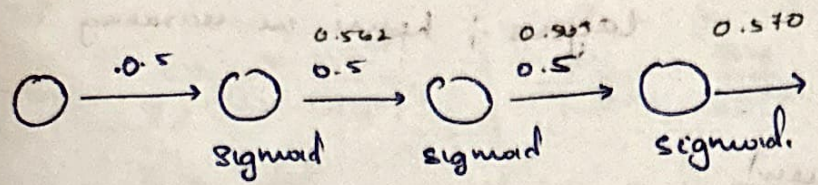
$3 \times 4 + 4 + 4 \times 2 + 2$

$12 + 4 + 8 + 2$

(26) Params in network

(5)

Vanishing gradient



Same applies for tanh

$z^{[1]} = 0.25$ | $z^{[2]} = 0.562 \times 0.5 = 0.28$ | $z^{[3]} = 0.285$

$a^{[1]} = 0.562$ | $a^{[2]} = 0.569$ | $a^{[3]} = 0.570$

$z^{[1]} = x a^{[0]} w^{[1]}$ $\quad a^{[1]} = \text{Sigmoid}(z^{[1]})$

$z^{[2]} = a^{[1]} w^{[2]}$ $\quad a^{[2]} = \text{sigmoid}(z^{[2]})$

$z^{[3]} = a^{[2]} w^{[3]}$ $\quad a^{[3]} = \text{Sigmoid}(z^{[3]})$ (y)

i) $da \quad \frac{\partial y}{\partial y} = 1$

ii) $\frac{\partial y}{\partial z^{[3]}} = \frac{\partial y}{\partial a^{[3]}} \frac{\partial a^{[3]}}{\partial z^{[3]}}$

$= 1 \times z^{[3]}(1 - z^{[3]}) = 0.285 \times (1 - 0.285)$

$= 0.204$

iii) $\frac{\partial y}{\partial a^{[2]}} = \frac{\partial y}{\partial z^{[3]}} \frac{\partial z^{[3]}}{\partial a^{[2]}}$ $\quad \frac{\partial y}{\partial w^{[3]}} = \frac{\partial y}{\partial z^{[3]}} \frac{\partial z^{[3]}}{\partial w^{[3]}}$

$= 0.204 \times w^{[3]} = 0.204 \times 0.5 = 0.1018$ $\quad = 0.204 \times 0.569 = 0.116$

iv) $\frac{\partial y}{\partial z^{[2]}} = \frac{\partial y}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}}$

$= 0.1018 \times z^{[2]}(1 - z^{[2]}) = 0.1018 \times 0.28 \times 0.72$

$= 0.0205$

v) $\frac{\partial y}{\partial a^{[1]}} = \frac{\partial y}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} = 0.0205 \times 0.5$ $\quad \frac{\partial y}{\partial w^{[2]}} = 0.0205 \times 0.562$

$= 0.01026$ $\quad = 0.01152$

vi) $\frac{\partial y}{\partial z^{[1]}} = 0.01026 \times 0.25 \times 0.75 = 0.0019240$

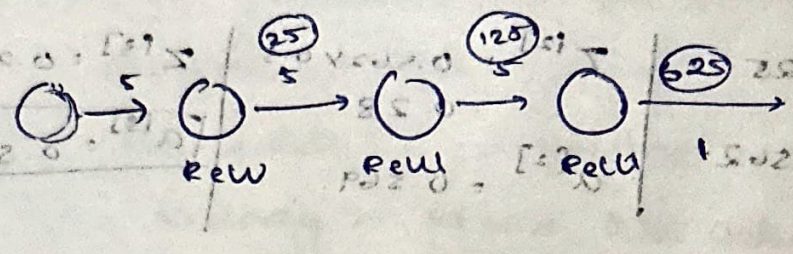
Gradient exp decreases &

$\frac{\partial y}{\partial w^{[1]}} = 0.0992 \times 0.5$

$= 0.00096.$

Eventually it vanishes with deeper layers ; hence no learning

Exploding Gradient



$z^{[1]} = w^{[1]} a^{[0]}$

$= 25$

$a^{[1]} = ReLU(z^{[1]})$

$= 25$

$z^{[2]} = w^{[2]} a^{[1]}$

$a^{[2]} = ReLU(z^{[2]})$

$= 125$

$z^{[3]} = w^{[3]} a^{[2]}$

$= 625$

$a^{[3]} = ReLU(z^{[3]})$

$= 625$

$y^{hat} = y$

$\frac{\partial y}{\partial y} = 1$; $\frac{\partial y}{\partial z^{[3]}} = 1 \times \frac{\partial a^{[3]}}{\partial z^{[3]}}$

$= 1$ (1)

$\frac{\partial y}{\partial a^{[2]}} = 1 \times w^{[3]}$; $\frac{\partial y}{\partial z^{[2]}} = \frac{\partial y}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}}$

$= 5$ (1)

$= 5 \times 1$

$\frac{\partial y}{\partial a^{[1]}} = 5 \times 5$; $\frac{\partial y}{\partial z^{[1]}} = 25 \times 1 = 25$

$= 25$

$\frac{\partial y}{\partial a^{[0]}} = 25 \times 5 = 125$

Exploding gradient

weights

$$\frac{\partial y}{\partial w^{[3]}} = \frac{\partial y}{\partial z^{[3]}} \times \frac{\partial z^{[3]}}{\partial w^{[3]}}$$

$$= 1 \times 1.25$$

$$= 125$$

$$\frac{\partial y}{\partial w^{[2]}} = \frac{\partial y}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w^{[2]}}$$

$$= 5 \times 25$$

$$= 125$$

$$\frac{\partial y}{\partial w^{[1]}} = \frac{\partial y}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial w^{[1]}}$$

$$= 25 \times .5$$

$$= 125$$

Explanation

During backpropagation the gradient at each layer is the product of the previous gradients and the derivative of activation function at that layer.

If a layer uses the sigmoid / tanh their derivatives are always less than 1 (maximum 0.25 for sigmoid, 1 for tanh but often much less) so the gradients can become exponentially smaller, as they traverse deeper layers, causing the early layer weights to change very little (vanishing gradient)

for ReLU, if weights are initialized with larger values, derivs can be greater than 1 causing gradients to multiply up as they propagate. becoming very large.

They can also encounter exploding ReLU problem for negative I/Ps

6. $X = [2, 4, 6, 8, 10]$

$\gamma = 2$

$\beta = 1$

X	$(x-\bar{x})^2$	$x_{norm} = \frac{x-\bar{x}}{\sqrt{var(x)}}$	$x_{new} = \gamma x_{norm} + \beta$
2	16	-1.265	-1.53
4	4	-0.633	-0.26
6	0	0	1
8	4	0.633	2.266
10	16	1.265	3.53
$\Sigma X = 30$	40		

$\frac{\Sigma X}{n} = \frac{30}{5} = 6 = \bar{x}$

$var(x) = \frac{1}{n-1}\Sigma(x-\bar{x})^2$

$\therefore \frac{40}{4} = 10.$

$var(x) = 10$

7. keep probability is the chance that neuron stays active during training

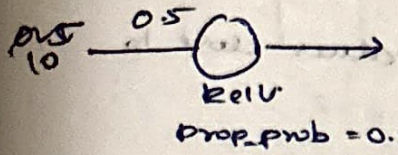
If keep_prob is 80%, then I can trust the o/p from the neuron only 80% of the time / The neuron has seen only 80% of the data

So we need to multiply each neuronal o/p with 0.8 during test → but this increases the time & computation during test stage which

is undesirable, so instead we can divide by keep prob during training stage itself

Example

training **test**



Sample 1 — i/p: 10, 0.5 — o/p: 5 — S1 → 5, S2 → 5

Sample 2 → dropped — S3 → 7.5

Sample 3 → dropped — S4 → 7.5

Sample 4 — 15

Total o/p = 12.5 — Total o/p = 25

mismatch.

↓

divide training by keep prob

$\frac{12.5}{0.5} = 25$

how it matches

8.

o/p of the convolution operation

6x6 * 3x3 = 4x4

30+0-30 = 0	30+0+0 = 30	30+0+0 = 30	0+0+0 = 0
0	30	30	0
0	30	30	0
0	30	30	0

This convolution filter extracts vertical edges in the input image. The filter emphasizes where there is a strong diff b/w left and right side of the window (such as the transition from 0s to 0s) producing high values at the boundaries

Flat or uniform regions produce o/p values close to 0

9.

As layers get deeper we encounter

→ vanishing gradient problem:- where gradients become very small during backpropagation. This makes it hard for early layers to learn effectively.

→ Because of vanishing gradients, the original image features might get diluted or lost over many layers which can hurt training and final performance

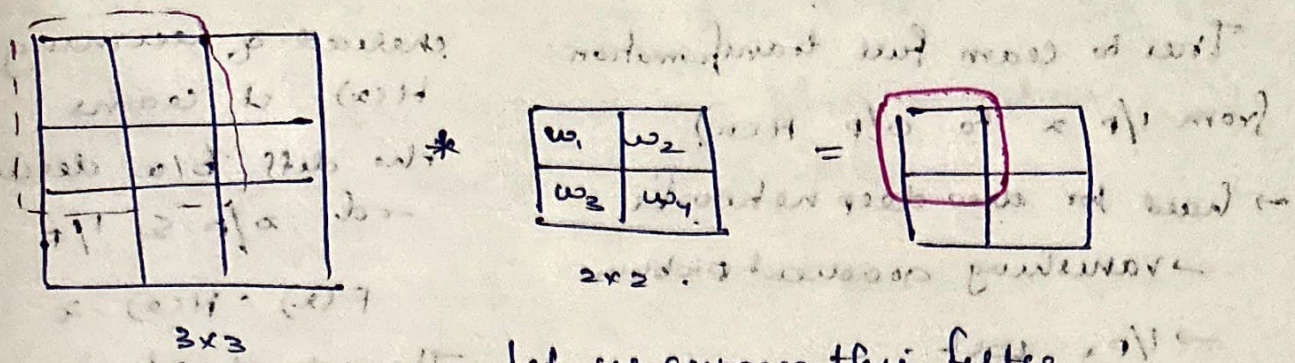
Positives

Can learn more detailed & complex features step by step, starting from simple edges in early layers to complex objects in deeper layers

It looks at larger parts of image (↑↑ Receptive field) gaining more context about patterns & shapes

Helps the But more layers, more computing power and careful training otherwise model could become harder to train effectively

10.



Let us suppose this filter can learn edges effectively

↓

then I superimpose it across the whole image to learn all the edges in the image

Here, the same set of weights is shared across diff spatial locations in the input.

- Translation invariant features:- It can recognise an abstract object regardless of where it appears in the image
- This concept of weight sharing dramatically reduces the number of parameters to be estimated

If we take one pixel in the o/p, it is obtained only from part of I/P image, not the entire image

(i.e) CNNs connect each neuron only to a small local region of the input called the Receptive field unlike FFN where every neuron is connected to every other neuron

- This is called Sparsity of connections
  - → helps focus on local spatial patterns effectively w/o having too many params
  - → It leverages the fact that pixels near each other are more related than pixels far away

Learning Deep Mapping

Tries to learn full transformation from i/p x to o/p H(x)

→ hard for very deep networks

→ vanishing gradient problem

→ i/p is lost

Learning Residuals

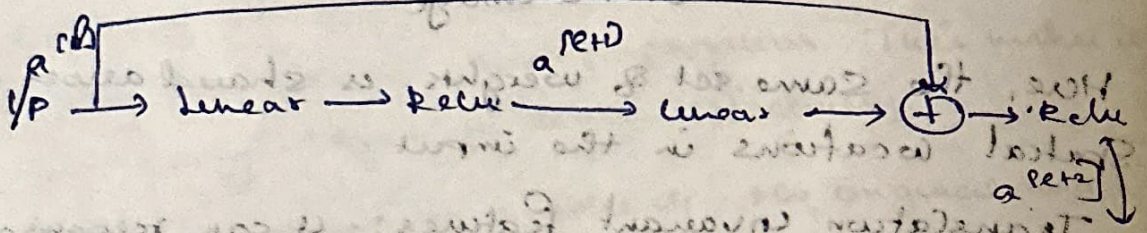
instead of learning H(x) - it learns the diff b/w desired o/p & i/p

$F(x) = H(x) - x$

The actual o/p is obtained by

$H(x) = F(x) + x$

Let us assume a network



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

If weights become very small then assume

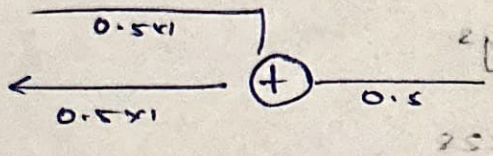
$a^{[l+1]} w^{[l+2]} + b^{[l+2]}$ becomes negligible

then the o/p becomes the i/p → if best solution cannot be found return the input

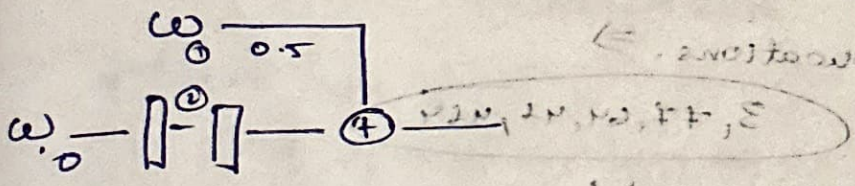
thus means the network can easily preserve i/ps w/o degradation.

12. Residual networks to solve the vanishing gradient problem, how does it solve.

As we saw network has skip connections which is basically an addition

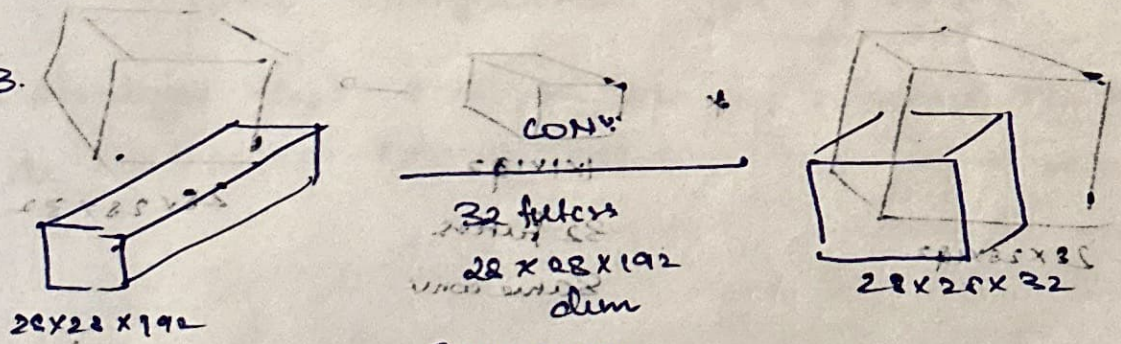


If its a plus gradient gets redistributed so even if one path becomes zero the other path still remains



$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial w}_{(1)} + \frac{\partial L}{\partial w}_{(2)}$$ → so gradient does not vanish.

13.



What should be my padding

$$\frac{f-1}{2} = p$$

$$\frac{28-1}{2} = p$$

$$p = \frac{27}{2} - 13.5$$

∴ which means padding should be asymmetric

top / bottom : 13 & 14

left / right : 13 & 14

final IP dimensions

55 x 55 x 192

Total no of overlays.

28 x 28 x 28 x 28

Multiplication in one overlay & across the layers across the filter

28 x 28 x 192 x 32

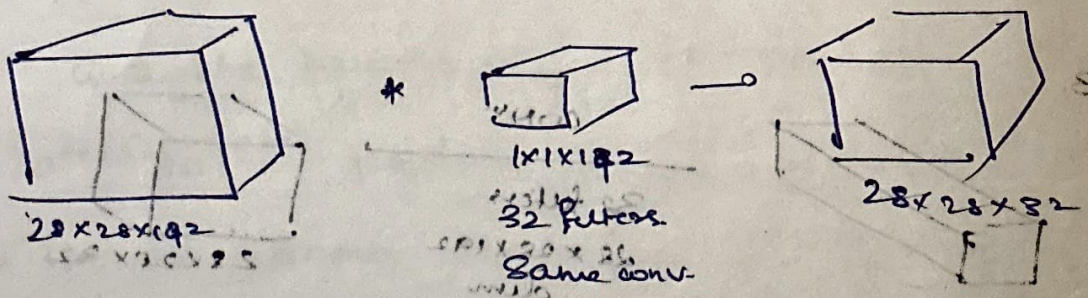
Totals reasons to be estimated as

Total multiplications. ⇒

3,77,64,41,414

∧ Cost !

1 by 1 conv



28 x 28 x 1 x 192 x 32

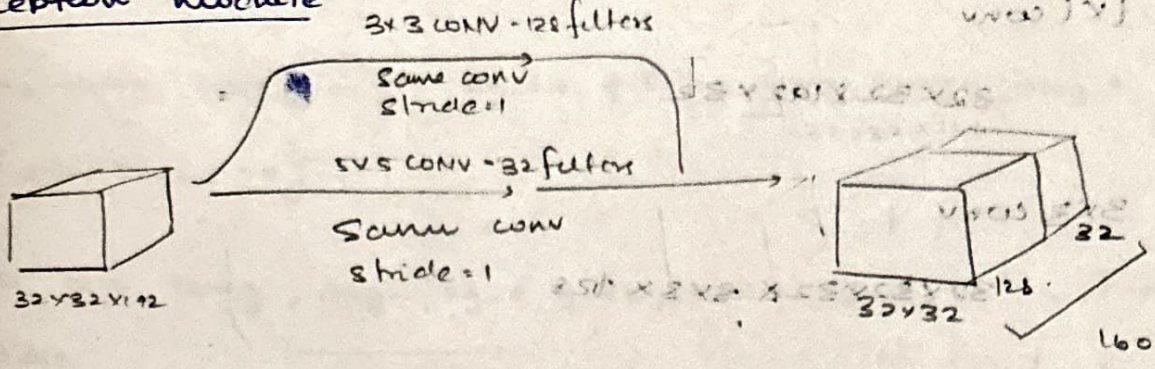
48,16,896

relatively very much less !!

→ so ideal to include another 3 x 3 filter

(14)

Inception module



1st 3×3 conv

no of multiplications ⇒

$32 \times 32 \times 3 \times 3 \times 128 \times 192$

$= 226492416$

2nd 5×5 conv

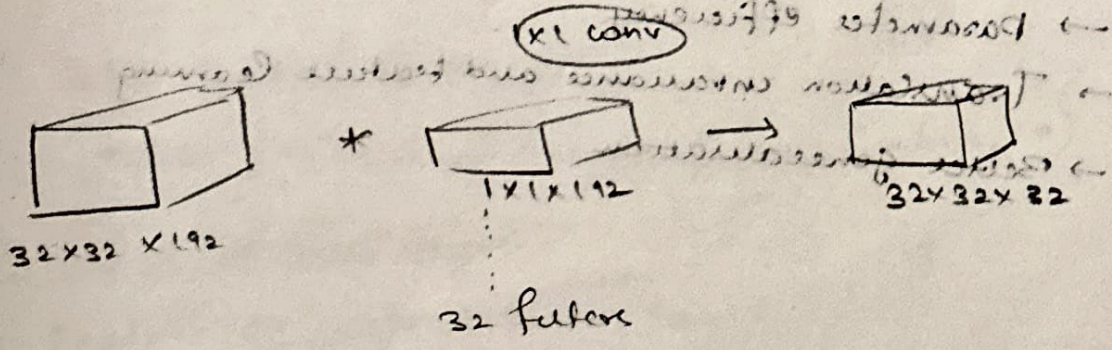
no of multiplications ⇒

$32 \times 32 \times 5 \times 5 \times 32 \times 192$

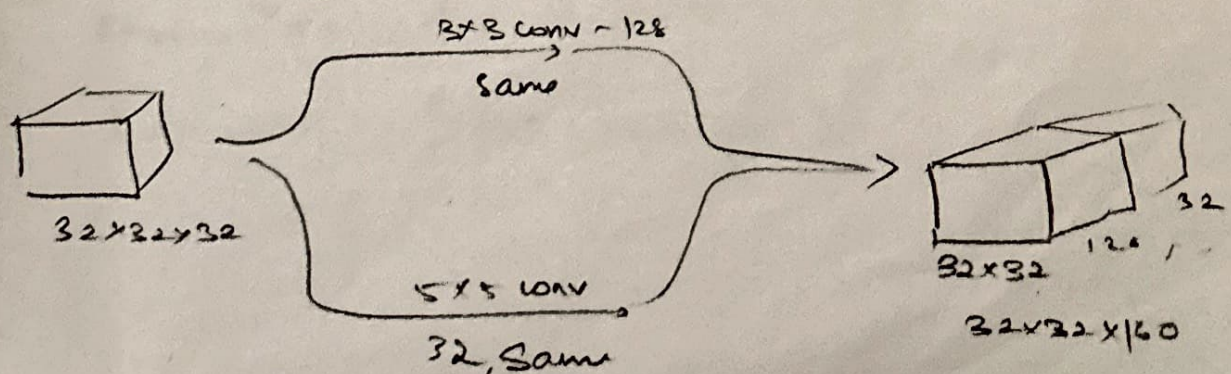
$= 157286400$

Total multiplications = 38,37,78,816.

The strategy that I would use is introduce 1×1 convolutions as bottleneck layers before expensive convolutions.



After this apply 3×3 & 5×5



No of multiplication

1X1 conv

32X32X192X32

3X3 conv

32X32X32X 3X3X 128

5X5 conv

32X32X32X5X5X32

add all to get 7,02,54,593

which is less by 81000000 multiplications

15. Between the two approaches using a CNN with 32X32 filters is better for object detection in images compared to flattening the image and using a FC layer

Reasons:-

→ preserves spatial structure

→ Parameter efficiency.

→ Translation invariance and feature learning

→ Better generalisation