

ML Theory - worksheet!

01/09/2025

Q1 O/P of last fc layer

$$\left[\begin{array}{c} \text{C1,00} \\ \text{C1,50} \\ \text{C2,00} \\ \text{C2,50} \end{array} \right] \times \left[\begin{array}{c} \text{F1,F0} = 100 \\ \text{F2,F0} = 100 \\ \text{F3,F0} = 100 \\ \text{F4,F0} = 100 \end{array} \right] \xrightarrow{\text{apply softmax}} \left[\begin{array}{c} \text{P1,F0} \\ \text{P2,F0} \\ \text{P3,F0} \\ \text{P4,F0} \end{array} \right] = \left[\begin{array}{c} 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{array} \right]$$

$$\frac{\text{C1,00}}{\text{C1,50}} = \frac{1.0}{1.5} = 0.6666666666666667 \quad \text{softmax formula}$$

$$\frac{\text{C1,00}}{\text{C1,50}} \times \frac{\text{F1,F0} = 100}{\text{F2,F0} = 100} = 0.6666666666666667 \quad \hat{y}(x) = \frac{e^{x_i}}{\sum_{i=1}^n e^{x_i}} = \frac{e^{2.72}}{e^{2.72} + e^{1.11} + e^{0.66} + e^{0.00}}$$

$$2.72 / 11.22 = 0.24 \quad \frac{e^{2.72}}{11.22} = \frac{0.24}{0.22}$$

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

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

$$P(\hat{y} = 2) = \frac{0.11}{11.22} = 0.01$$

$$P(\hat{y} = 3) = 0.01$$

$$P(\hat{y} = 0) = \frac{0.65}{1.0} = 0.65$$

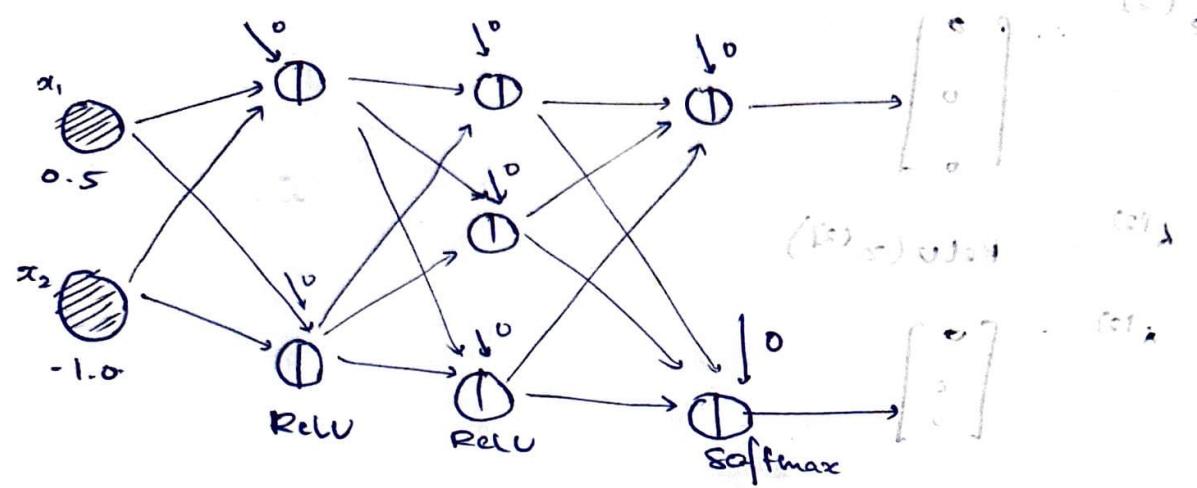
$$P(\hat{y} = 1) = \frac{0.24}{1.0} = 0.24$$

$$P(\hat{y} = 2) = \frac{0.11}{1.0} = 0.11$$

$$P(\hat{y} = 3) = \frac{0.01}{1.0} = 0.01$$

$$P(\hat{y} = 0) = \frac{0.65}{1.0} = 0.65$$

②

FFN

$$\omega^{(1)} = \begin{bmatrix} \omega_{11}^{(1)} & \omega_{12}^{(1)} \\ \omega_{21}^{(1)} & \omega_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} 0.742 & 0.794 \\ 0.561 & 1.0 \end{bmatrix}$$

$$\omega^{(2)} = \begin{bmatrix} \omega_{11}^{(2)} & \omega_{12}^{(2)} \\ \omega_{21}^{(2)} & \omega_{22}^{(2)} \\ \omega_{31}^{(2)} & \omega_{32}^{(2)} \end{bmatrix} = \begin{bmatrix} 0.725 & 0.613 \\ 0.416 & 0.092 \\ 0.876 & 0.590 \end{bmatrix}$$

(row 1 for $\omega_{11}^{(2)}$, row 2 for $\omega_{21}^{(2)}$, row 3 for $\omega_{31}^{(2)}$)

$$\omega^{(3)} = \begin{bmatrix} \omega_{11}^{(3)} & \omega_{12}^{(3)} & \omega_{13}^{(3)} \\ \omega_{21}^{(3)} & \omega_{22}^{(3)} & \omega_{23}^{(3)} \\ \omega_{31}^{(3)} & \omega_{32}^{(3)} & \omega_{33}^{(3)} \end{bmatrix} = \begin{bmatrix} 0.382 & 0.124 & 0.233 \\ 0.112 & 0.132 & 0.534 \end{bmatrix}$$

$$z^{(1)} = \omega^{(1)} x = \begin{bmatrix} 0.742 & 0.794 \\ 0.561 & 1.0 \end{bmatrix} \begin{bmatrix} 0.5 \\ -1.0 \end{bmatrix}$$

$$z^{(2)} = \begin{bmatrix} -0.423 \\ -0.4195 \end{bmatrix}$$

$$A^{(1)} = \text{ReLU}(z^{(1)}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

1st Layer

③ D

| 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 |

[forward pass]

④

| x_1 | x_2 | $z_2^{(2)}$ | $a_2^{(2)}$ |
|-------|-------|-------------|-------------|
| 0 | 0 | -1 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |

O/P layer

| $a_1^{(1)}$ | $a_2^{(1)}$ | $z^{(2)}$ | $a_2^{(2)}$ |
|-------------|-------------|-----------|-------------|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |

final O/P. $\rightarrow \hat{y}$

state of art for handwritten digit

loss computation



Loss formula (MSE case)

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

(\hat{y}, y)

16

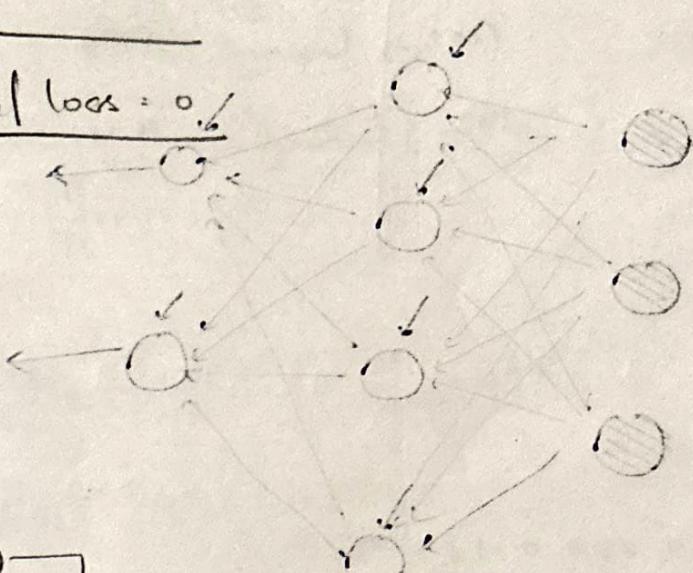
N^6

o :

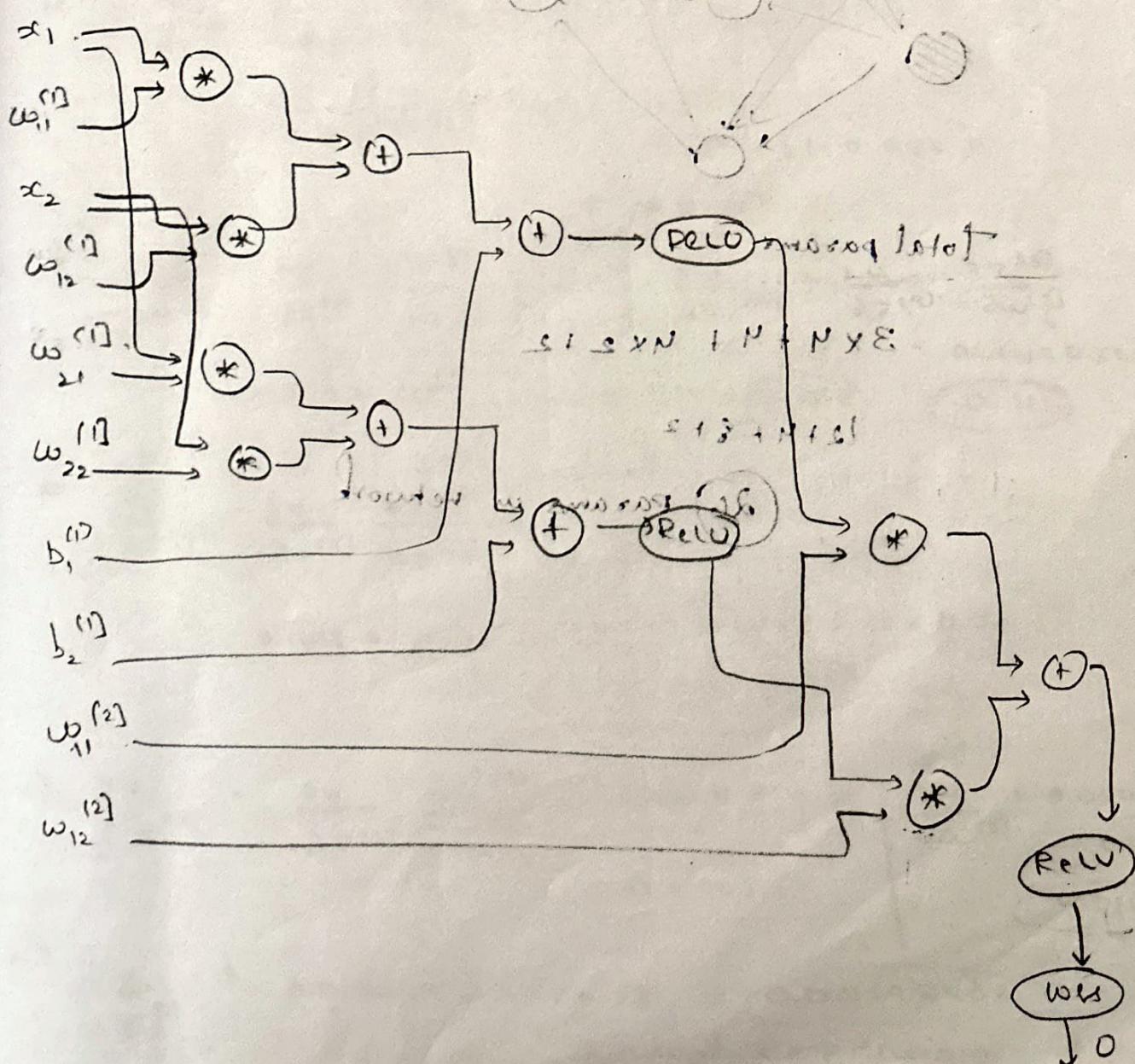
16

- No \hat{y} ~~represented~~ by y number of $\hat{y}_i = \frac{16}{16}$
1 o next \hat{y} was ~~represented~~ by
2 1 instances \hat{y}_i is shown off
3 , forward and backward pass
4 0 0

$$\text{Total loss} = 0.$$



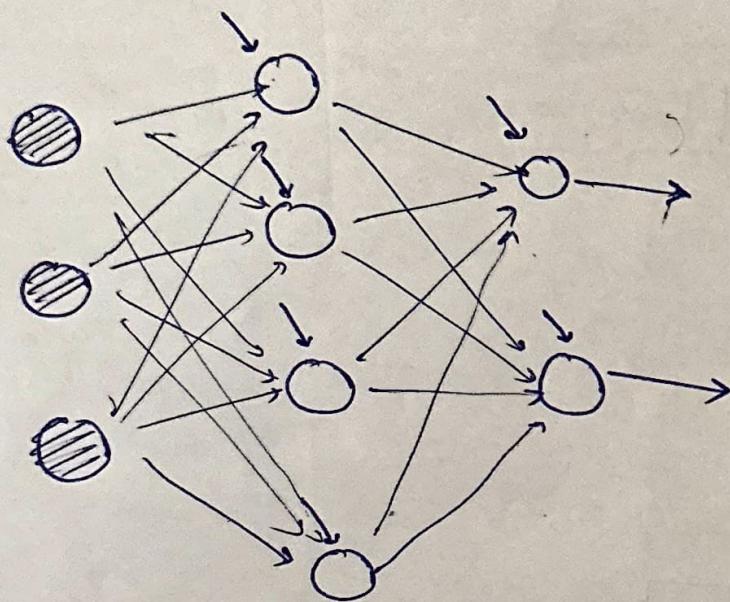
[backward pass]



$$\frac{\partial L}{\partial y} = 2(y - \hat{y}) \\ = 0.$$

$\frac{\partial L}{\partial y} = 0 \rightarrow$ the gradients are backpropagated so all gradients will be zero
 No update in parameters
 Already minimum loss achieved

(9)



[for backprop]

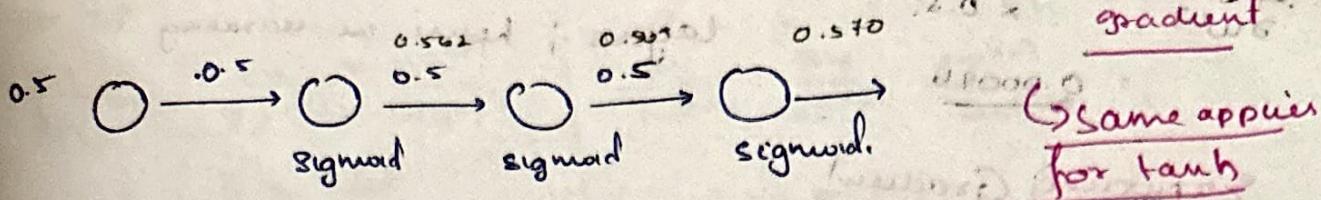
Total params

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

$$12 + 4 + 8 + 2$$

(26) Params in network

(5)



Vanishing gradient

↳ Same applies for tanh

$$\begin{array}{c|c|c} z^{(1)} = 0.25 & z^{(2)} = 0.562 \times 0.5 & z^{(3)} = 0.285 \\ a^{(1)} = 0.562 & a^{(2)} = 0.28 & a^{(3)} = 0.570 \end{array}$$

$$\begin{array}{l} z^{(1)} = \text{sigmoid}(a^{(0)} w^{(1)}) \\ z^{(2)} = a^{(1)} w^{(2)} \\ z^{(3)} = a^{(2)} w^{(3)} \end{array} \quad \begin{array}{l} a^{(1)} = \text{sigmoid}(z^{(1)}) \\ a^{(2)} = \text{sigmoid}(z^{(2)}) \\ a^{(3)} = \text{sigmoid}(z^{(3)}) \end{array} \quad \begin{array}{l} y = a^{(3)} \end{array}$$

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

$$ii) \frac{\partial y}{\partial z^{(3)}} = \frac{\partial y}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} = \frac{0.4}{0.285} = 1.42857 \quad \begin{array}{l} \text{Ans} \\ \text{Ans} \end{array}$$

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

$$iii) \frac{\partial y}{\partial a^{(2)}} = \frac{\partial y}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial a^{(2)}} = \frac{0.204}{0.285} = 0.7142857$$

$$= 0.204 \times w^{(3)} = 0.204 \times 0.5 = 0.1018 \quad \begin{array}{l} \text{Ans} \\ \text{Ans} \end{array}$$

$$iv) \frac{\partial y}{\partial z^{(2)}} = \frac{\partial y}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} = \frac{0.1018}{0.285} = 0.357142857 \quad \begin{array}{l} \text{Ans} \\ \text{Ans} \end{array}$$

$$= 0.1018 \times z^{(2)} (1 - z^{(2)}) = 0.1018 \times 0.28 \times 0.72 \quad \begin{array}{l} \text{Ans} \\ \text{Ans} \end{array}$$

$$= 0.0205 \quad \begin{array}{l} \text{Ans} \\ \text{Ans} \end{array}$$

$$v) \frac{\partial y}{\partial w^{(2)}} = \frac{\partial y}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} = 0.0205 \times 0.5 = 0.01025 \quad \begin{array}{l} \text{Ans} \\ \text{Ans} \end{array}$$

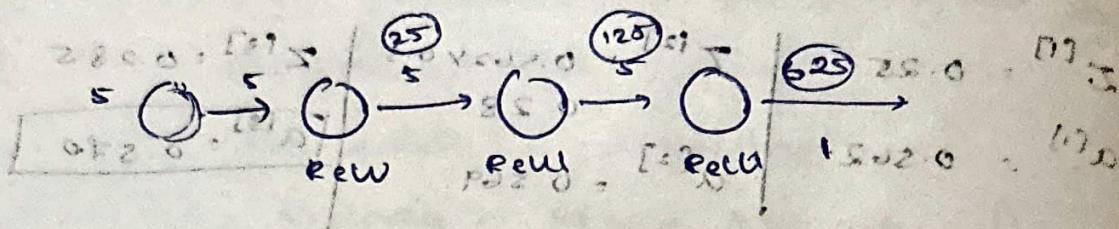
$$vi) \frac{\partial y}{\partial I^{(1)}} = 0.01025 \times 0.25 \times 0.45 = 0.00192402 \quad \begin{array}{l} \text{Ans} \\ \text{Ans} \end{array}$$

Gradient keep decreasing &

$$\frac{\partial y}{\partial w^{(1)}} = 0.0092 \times 0.5 \\ = 0.00096.$$

Eventually it vanishes with deeper layers.; hence no learning

Exploding Gradient



$$z^{(1)} = w^{(1)}a^{(0)} \\ = 25 \quad (\text{bias} = 125)$$

$$a^{(1)} = \text{relu}(z^{(1)}) \\ = 25$$

$$z^{(2)} = w^{(2)}a^{(1)} \\ = 125 \quad (\text{bias} = 125)$$

$$a^{(2)} = \text{relu}(z^{(2)}) \\ = 125$$

$$z^{(3)} = w^{(3)}a^{(2)} \\ = 625 \quad (\text{bias} = 125)$$

$$a^{(3)} = \text{relu}(z^{(3)}) \\ = 625$$

$$\frac{\partial y}{\partial y} = 1 \quad ; \quad \frac{\partial y}{\partial z^{(3)}} = 1 \times z^{(3)} \times (1 - z^{(3)}) \frac{\partial z^{(3)}}{\partial a^{(2)}} = \frac{\partial z^{(3)}}{\partial a^{(2)}} = 1$$

$$= 625(1 - 1) \quad (1)$$

$$(235.0 - 1) \times 235.0 = (234.0) \times 235.0 =$$

$$\frac{\partial y}{\partial a^{(2)}} = \frac{\partial y}{\partial z^{(3)}} \times \frac{\partial z^{(3)}}{\partial a^{(2)}} \\ = 1 \times 625 \times 1 = 625$$

$$125 \times 125 = 15625 = 25 \times 25 = 625 \times 1 =$$

$$\frac{\partial y}{\partial a^{(1)}} = 5 \times 5 \quad ; \quad \frac{\partial y}{\partial z^{(2)}} = 25 \times 1 = 25$$

$$= 25$$

$$25 \times 25 = 625 \times 1 = (25^2) \times 1 =$$

$$\frac{\partial y}{\partial a^{(0)}} = 25 \times 5 + 125$$

$$= 25 \times 25 + 125 = 625 + 125 = 750$$

Exploding gradient

$$= 750 \times 25 = 18750$$

weights

$$\frac{\partial y}{\partial w^{(3)}} = \frac{\partial y}{\partial z^{(2)}} \times \frac{\partial z^{(2)}}{\partial w^{(2)}}$$

$$= 1 \times 1.25$$

$$= 1.25$$

(0, 1, 2, 3, 4, 5) X

$$\frac{\partial y}{\partial w^{(2)}} \times \frac{\partial y}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial w^{(1)}}$$

$$= 5 \times 25$$

$$= 125$$

1 = 25

X

$$\frac{\partial y}{\partial w^{(1)}} = \frac{\partial y}{\partial z^{(0)}} \frac{\partial z^{(0)}}{\partial w^{(1)}}$$

$$= 125 \times 5$$

$$= 1.25$$

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, 0.5 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 large values, then gradients will be multiplied up as they propagate, becoming very large.

They can also encounter a slight ReLU problem for negative inputs. Just as first process 3.0 when input is -0.5, first process is 0.0 which is wrong.

$$b. \quad X = f(2, 4, 6, 8, 10)$$

| X | $(x - \bar{x})^2$ | x_{norm} | $x_{\text{new}} = \beta x_{\text{norm}} + \beta$ |
|-----|-------------------|-------------------------------------|---|
| | | $\frac{x - \bar{x}}{\text{Var}(x)}$ | $\frac{\beta x_{\text{norm}} + \beta}{\text{Var}(x)}$ |
| 2 | 16 | -1.265 | -1.53 |
| 4 | 4. | -0.633 | 0.26 |
| 6 | 0. | 0 | 2.126 |
| 8 | 4 | 0.633 | 2.82.266 |
| 10 | 16 | 1.265 | 3.53 |

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

treating all numbers as given

of neutrinos are known to have massiveness & therefore
and which have short lifespan and are called as f-

$\text{Var}(x) = \frac{1}{n-1} \sum (x - \bar{x})^2$ neutrinos have massless &
no mass & are called as p- (protons with no mass),
which are stable & have long life time,

so $\text{Var}(x) = 10$ called p-10 (protons with mass)
and $\text{Var}(x) = 10$ called f-10 (f neutrinos with mass)

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%. If the time / the neuron has

seen only 80% of the data

So we need to multiply each generated o/p with

0.8 during test → but this increases the

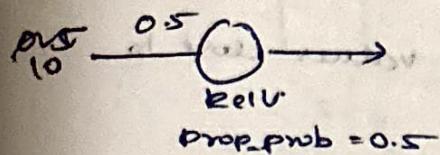
time & computation during test stage which

is considerable, so instead we can deviate by keep
boat clearing 'scouring' stage itself off. Goto 2 Aug 19

Example

to aiming

Test



Sample 1 1/p 0/p.
 0.5 5
 10

$s_1 \rightarrow s$

82 → 5

Sample 2 → dropped

$$53 \rightarrow 7.5$$

Sample 3. → dropped

SA → T

Sample 4 15th vent. & vessel of *Gymnophorus* per arched

Total O/P = 12.5 weeks Total O/P = 25 weeks

from two to ten species of leaves.

long hair mismatch. New victory

divide training by keep prob

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

How it matches

| | | | |
|-----------|----------|----------|---------|
| $30+0-30$ | $30+0+0$ | $30+0+0$ | $0+0+0$ |
| $= 0$ | $= 30$ | $= 30$ | $= 0$ |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |

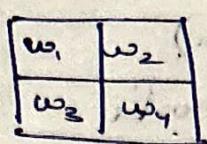
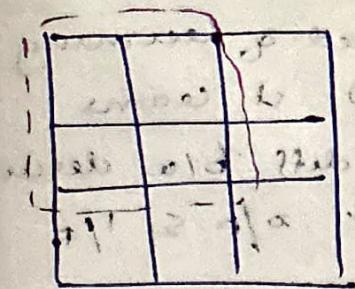
This convolution filter extracts vertical edges in the input image. The filter emphasizes where there is a strong diff b/w left and right side of the window (such as the transition from 100 to 00) producing high values at the boundaries. Flat or uniform regions produce 0% values close to.

- a. 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 deleted or lost over many layers which can hurt training and final performance.

Positives

- (22) - 2.01
- Can learn more detailed, & complex features step by step, starting from simple edges in early layers to complex objects in deeper layers
 - If looks at larger part of image (\uparrow receptive field) gaining more context about patterns & shapes
 - Helps \uparrow But more layers \uparrow computing power and careful training otherwise model could become harder to train effectively

10. Convolutional neural network



2×2 result of convolution

3×3 input let us suppose this filter can learn Edge effectively

Then we 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 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, its obtained only from part of the 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 Sparcity 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

Tree to learn full transformation
from $I/P \propto f_0 O/P H(x)$

→ hard for deep networks

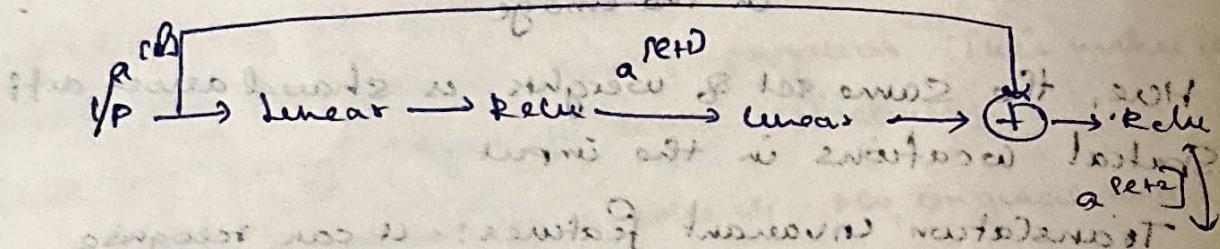
→ vanishing gradient problem

→ $I/P = \text{const}$

but mapped w.r.t.
generalized input word

it leads to vanishing gradient

Let us assume a Network



$$a^{(l+2)} = \text{ReLU}(\alpha^{(l+1)} w^{(l+2)} + b^{(l+2)})$$

If weights become very small then assume
 $w^{(l+2)} + b^{(l+2)}$ becomes negligible

then the output becomes the $I/P \rightarrow$ if best solution

cannot be fed retaining the input. In this way

this means the network can easily freeze

I/P w/o degradation.

more auto zeros at bottleneck

and hence for pt. loss \rightarrow hidden \rightarrow unit

existing storage loss no error added &

error won't pass through bottleneck

can store last few bits approx. to 1

and each row can be auto loss

Learning Residuals

instead of learning
 $H(x) \rightarrow$ it learns

the diff b/w desired
→ d $\approx I/P \approx 1/I/P$

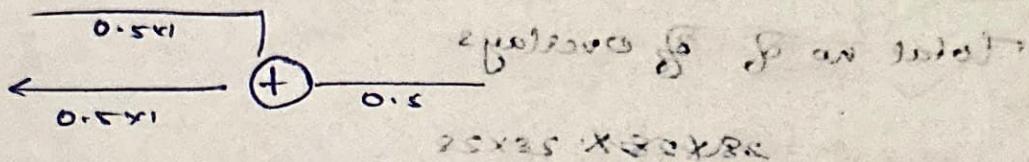
$$F(x) = H(x) - d$$

The actual O/P is
obtained by

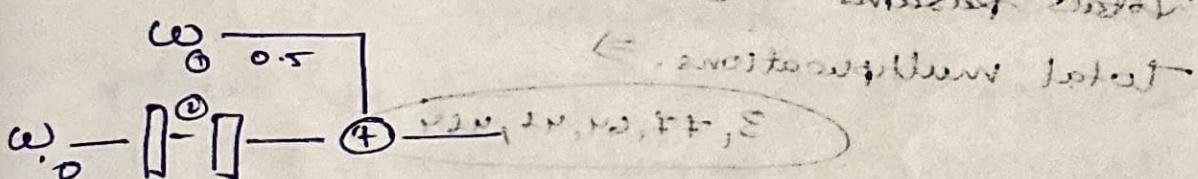
$$H(x) = F(x) + d$$

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

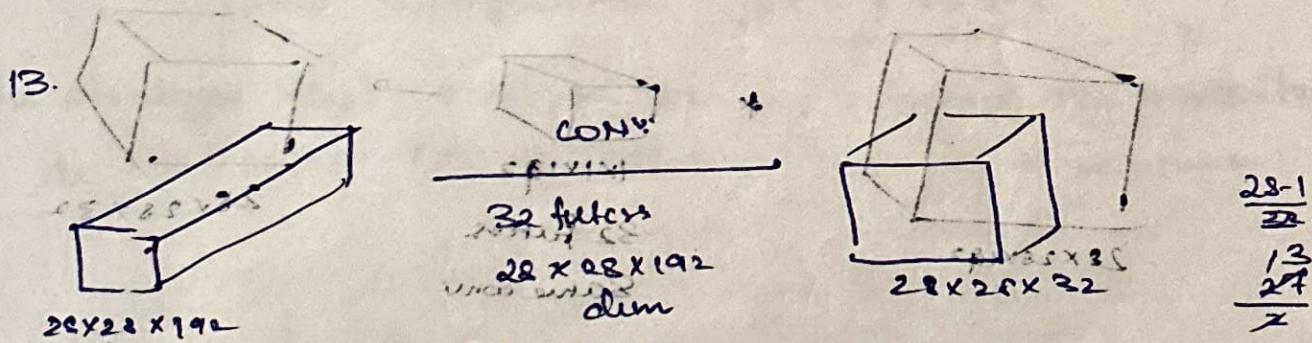
as we saw network has skip connections
which is basically an addition



If its a pure gradient gets distributed
So even if one path becomes zero the other
path still remains



$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial w} + \frac{\partial L}{\partial w} \rightarrow \text{so gradient does not vanish}$$



What should be my padding

$$f=1, p=1, s=1$$

$\frac{f+1}{2} - 1$ p. Just you calculate

$$\begin{aligned} f &= 2 \\ p &= 2 \\ s &= 2 \end{aligned}$$

$$\Rightarrow \frac{28+1}{2} = 15$$

$$\text{therefore } p = \frac{24}{2} = 12.5$$

which means padding should be asymmetric

$$\begin{aligned} f &= 1 \\ p &= 12 \\ s &= 1 \end{aligned}$$

$$\begin{aligned} \frac{28}{2} &= 14 \\ 14 &= 14 \end{aligned}$$

$$54-25-11$$

$$52-22-11$$

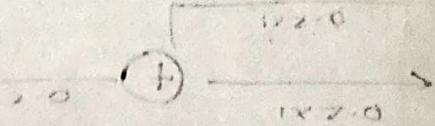
$$\begin{aligned} f &= 2 \\ p &= 11 \\ s &= 1 \end{aligned}$$

$$\begin{aligned} f &= 1 \\ p &= 12 \\ s &= 1 \end{aligned}$$

$$\begin{aligned} \frac{28}{2} &= 14 \\ 14 &= 14 \end{aligned}$$

top / bottom: 13x14 rows of 1x1 kernel
 left / right: 13x14 rows need, neighbors 3x3 step
 final output dimensions not finalized and are 20
 $55 \times 55 \times 192$ followed as shown

Total no of overlays.



$$28 \times 28 \times 1 \times 28 \times 28$$

Multiplication in one overlay & across the layers
also with $28 \times 28 \times 192 \times 32$ w.r.t. across the filter

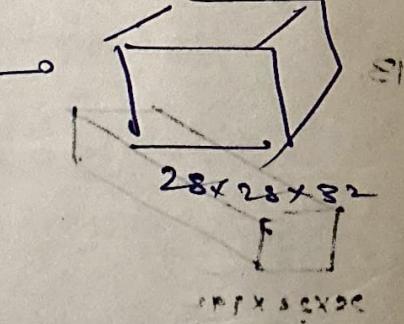
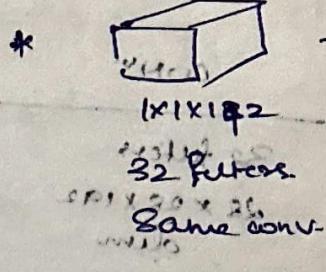
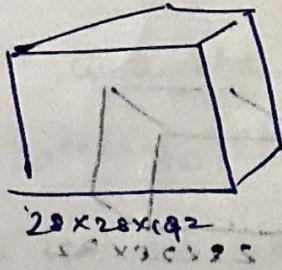
Total passes to be estimated at

Total multiplications. \Rightarrow

$$3,47,64,42,454$$

$\times \text{ cost}$!

for each layer pass $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2}$
1 by 1 conv



$$28 \times 28 \times 1 \times 192 \times 32$$

$$48,16,896$$

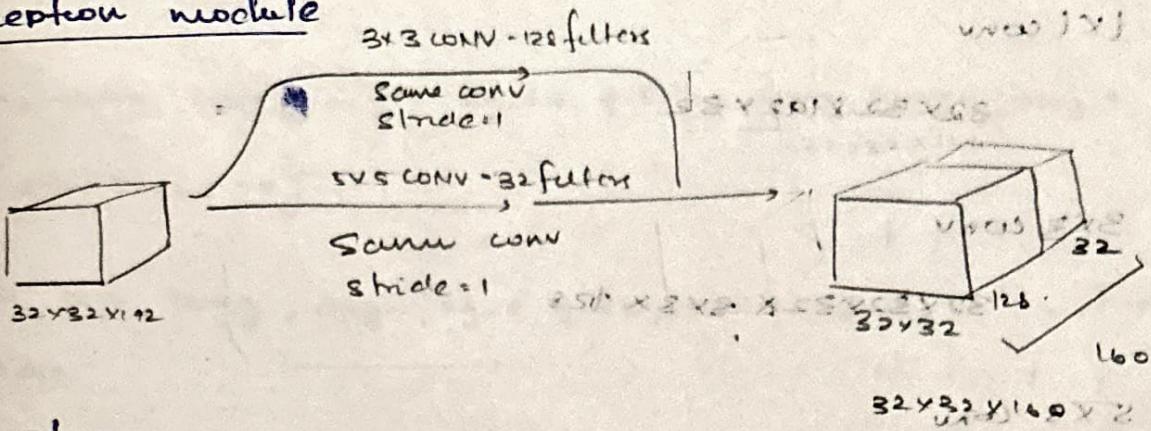
relatively very much less?

\rightarrow we need to
add 1 to 1, include
another convolution which needs a 3×3 filter
and so on

(14)

inference time of one

Inception module



1st 3x3 conv

no of multiplications $\Rightarrow 2 \times 2 \times 32 \times 32 \times 192$

$$32 \times 32 \times 32 \times 32 \times 192.$$

~~192² * 192 * 192~~
226492416 top of no 32x32

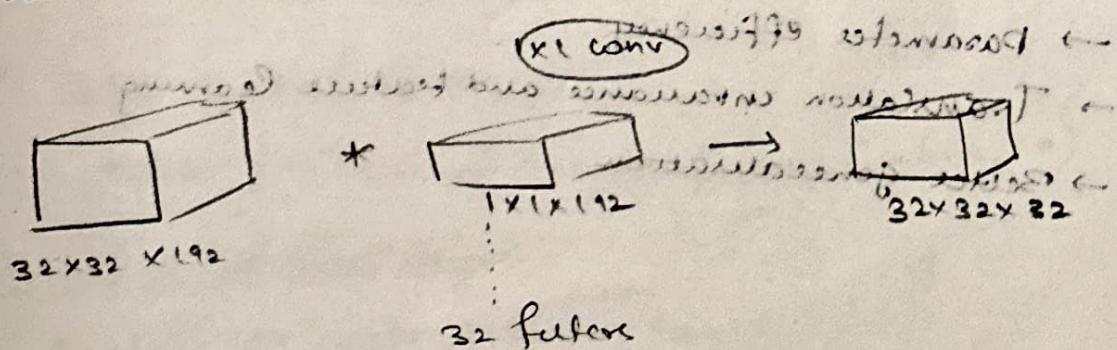
2nd 5x5 conv ~~0000000018~~ got next to nothingno of multiplications \Rightarrow when 4x4x8 pixels get mapped out not mapped \Rightarrow 32x32x192

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

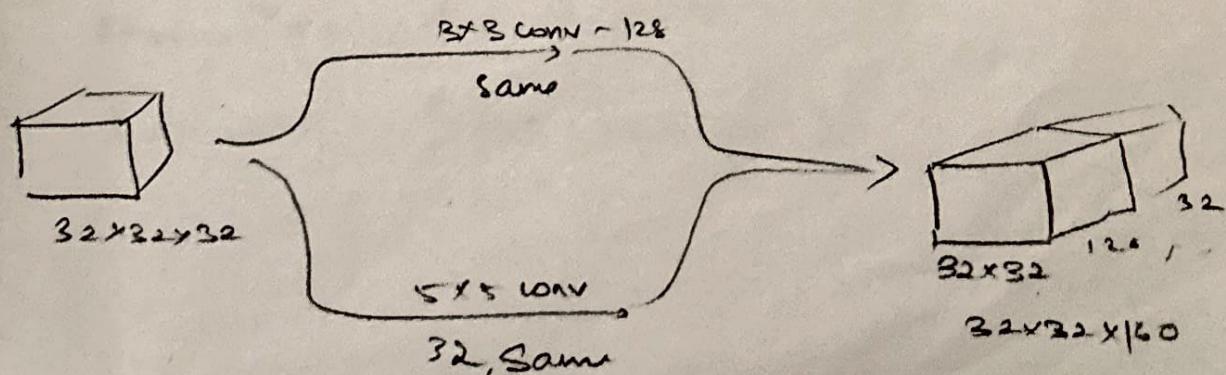
~~expansive is not taking too much not used as costly~~ $\Rightarrow 157286400$ ~~a pixel has spent all possibilities of being 192~~

$$\text{Total multiplications} = \boxed{38,34,78,816}$$

The strategy that I would use is introduce 1x1 convolutions as bottleneck layers ~~underlie for expensive convolutions~~.



After this apply 5x3 5x5



NO of multiplication

1x1 conv

$$32 \times 32 \times 1 \times 32 \times 32$$

3x3 conv

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

5x5 conv

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

add all to get $(7,02,54,592)$

which is less by $81,00,00,000$ multiplications

15. Between the two approaches using a CNN with 32×32 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 generalization