WORKSHEET SOLUTIONS

(1) Let us say the output of a last FC layer is [2.0 1.0 0.1]. If we need to convert this network into predicting class probabilities for 3 classes, how will you go about doing it and what are the probability values for 3 classes?

(A) For output layer, we should Softmax function: Softmax converts raw scores (logits) into probabilities by exponentiating each score.

$$S_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_i}}$$

$$S_1 = \frac{e^{x_1}}{\sum_{j=1}^{3}\left(e^{x_1} + e^{x_2} + e^{x_3}\right)} \Rightarrow \frac{e^2}{\left(e^2 + e^1 + e^{0.1}\right)} \Rightarrow \frac{7.3}{7.3 + 2.7 + 1.1} \Rightarrow \frac{7.3}{11.1} \Rightarrow 0.66$$

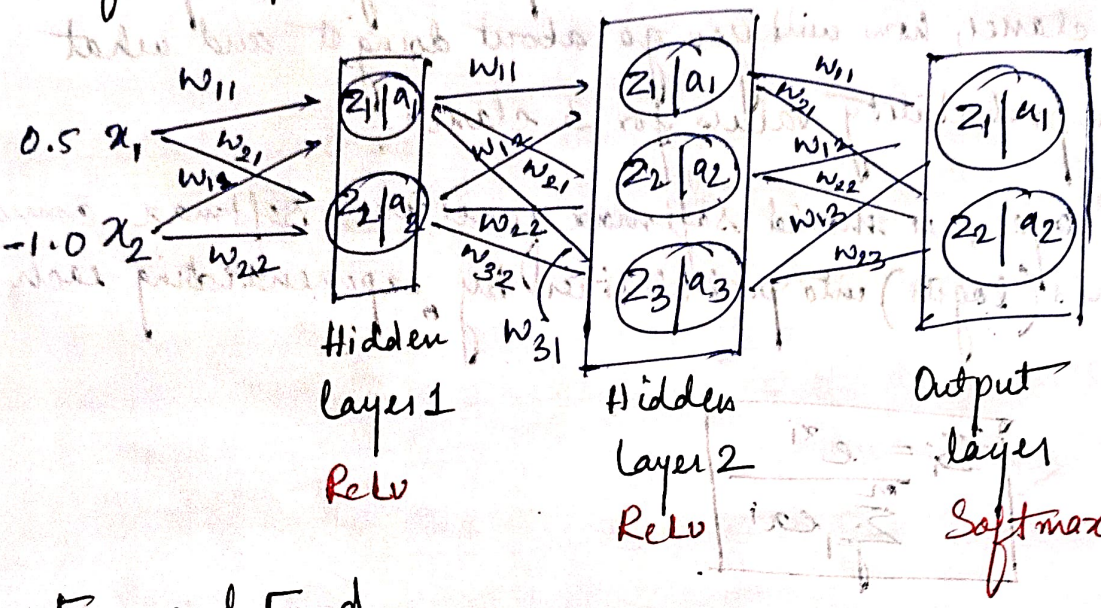
$$S_2 = \frac{e^1}{11.1} \Rightarrow \frac{2.7}{11.1} \Rightarrow 0.24$$

$$S_3 = \frac{e^{0.1}}{11.1} \Rightarrow \frac{1.1}{11.1} \Rightarrow 0.099 \simeq 0.1$$

$$S = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} 0.66 \\ 0.24 \\ 0.1 \end{bmatrix} \quad \text{(Ans)}$$

(2). Consider the following FFN. Input $X = [0.5 \; -1.0]$, two hidden layers with 2 and 3 neurons respectively. The output layer contains layer contains two neurons. ReLU non-linearity is applied for hidden layers and softmax operation is applied for the output layer activations. For simplicity assume that there are no bias terms connected to

neurons. Initialize the weight matrices randomly. Represent activations and weights as vectors and matrices and compute the forward pass output using a series of matrix multiplications.



Forward Feed

Layer 1

$z_1 = w_{11}x_1 + w_{12}x_2$

$= (0.2 \times 0.5) + (1.0 \times -1.0)$

$= 0.1 + (-1)$

$= 0.1 - 1 \Rightarrow -0.9$

taking $w_{11} = \begin{bmatrix} 0.2 & -0.4 \\ 1.0 & 0.1 \end{bmatrix}$

$z_2 = w_{21}x_1 + w_{22}x_2$

$= (0.4 \times 0.5) + (0.1 \times -1.0)$

$= -0.2 + (-0.1)$

$\Rightarrow -0.3$

$\therefore a_1 = [-0.9, -0.3]$

applying ReLu:- $\frac{x + |x|}{2} \Rightarrow a_1 = \frac{-0.9 + |-0.9|}{2}$

$a_2 = \frac{-0.3 + |0.3|}{2}$

$\Rightarrow \frac{-0.3 + 0.3}{2} \Rightarrow 0$

$\Rightarrow \frac{-0.9 + 0.9}{2}$

$\Rightarrow 0$

Layer 2

taking $W_{2e} = W^{[2]} = \begin{bmatrix} -0.3 & 0.8 & 0.5 \\ 0.7 & -0.6 & 0.2 \end{bmatrix}$

$z_1 = 0 \quad z_2 = 0 \quad z_3 = 0$

as a_1 and a_2 are zero

0 × multiplying with any weights will give zero.

Hence $a_1 = 0, \quad a_2 = 0, \quad a_3 = 0$ ← Relu layer.

Layer 3

taking $W^{[3]} = \begin{bmatrix} 0.4 & -0.2 \\ 0.3 & 0.9 \\ -0.5 & 0.7 \end{bmatrix}$

$z_1 = 0 \quad z_2 = 0$ → Similarly everything will be zero

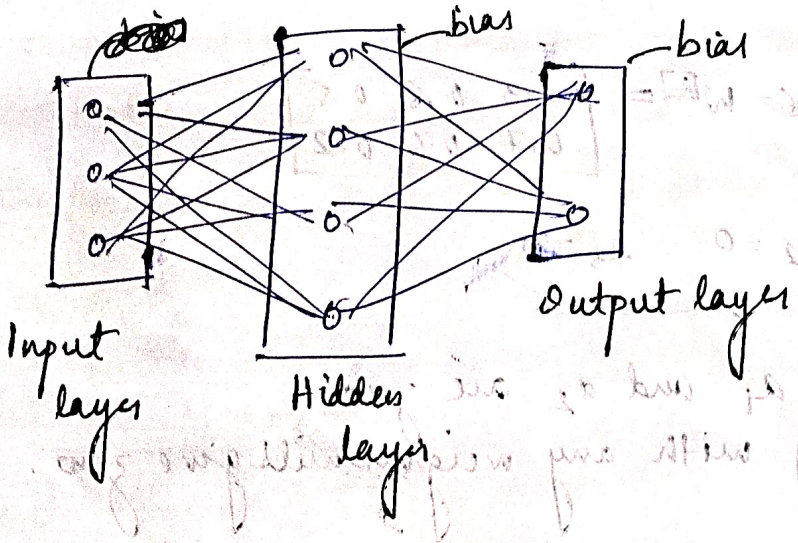
Applying softmax:

$a_1 = \frac{e^0}{e^0 + e^0} \Rightarrow \frac{1}{1+1} \Rightarrow \frac{1}{2} \Rightarrow 0.5$

$a_2 = 0.5$

$P = [0.5, 0.5]$

(4) Consider a 2 layer fully connected feedforward network with 3 neurons in the input layer, 4 neurons in the hidden layer and 2 neurons in the output layer. Bias terms are connected to all neurons in the hidden and output layers. What will be the total no. of parameters in the network?



Input to ~~Output~~ Hidden layer

$\Rightarrow 3 \times 4 = 12$

$\Rightarrow 12 + 4$ bias terms

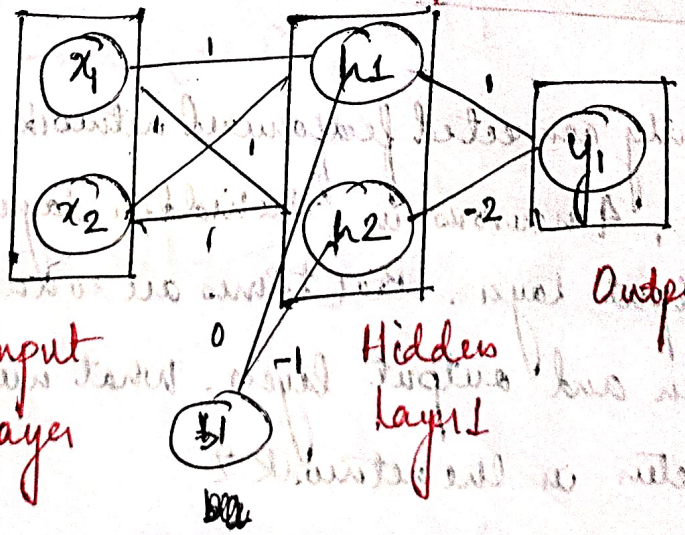
$\Rightarrow 16$

Hidden to Output $\rightarrow 4 \times 2 \rightarrow 8 + 2$ bias terms for Output layer

$\rightarrow 10$

Total $\rightarrow 16 + 10 \rightarrow$ 26 Parameters are there in the Network.

(3) Construct a computational graph, compute the gradients and update the weights, for a single forward and backward pass. The activations are ReLu for hidden layers. The input X is shown in the Table below. Use all the four samples as a single batch.



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

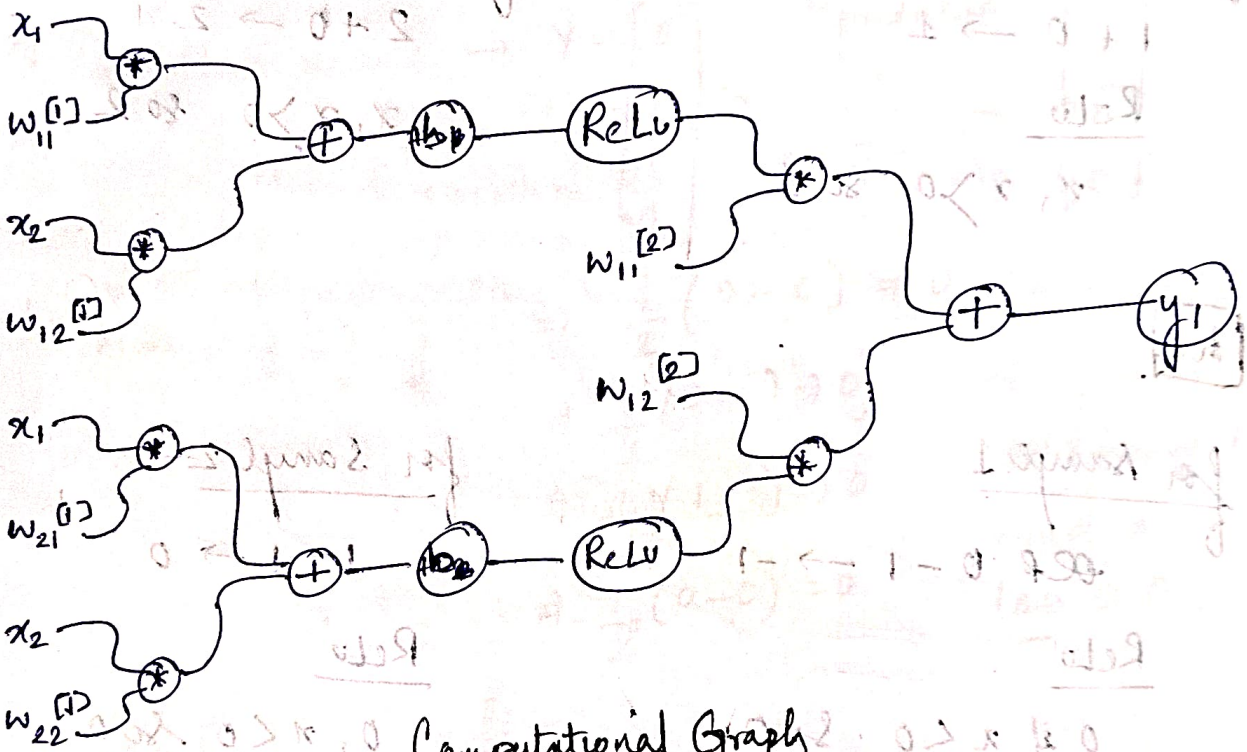
for the ~~computational~~ graph

~~1. Forward Pass~~

Hidden Layer W: $\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

Output layer $W = [w_{11} \; w_{12}]$

$= [1 \; -2]$



Computational Graph.

Forward Pass.

for $x_1 = 0$ $x_2 = 0$ $\quad h_1 = x_1 w_{11} + x_2 w_{12} \Rightarrow 0 + 0 \Rightarrow 0$

for $x_1 = 0$ $x_2 = 1$ $\Rightarrow 0 + 1(1) \Rightarrow 1$

for $x_1 = 1$ $x_2 = 0$ $\Rightarrow 1 + 0 \Rightarrow 1$

for $x_1 = 1$ $x_2 = 1$ $\Rightarrow 1 + 1 \Rightarrow 2$

h_2
$0 + 0 \Rightarrow 0$
$0 + 1 \Rightarrow 1$
$1 + 0 \Rightarrow 1$
$1 + 1 \Rightarrow 2$

h1

for sample 1

0 + bias → 0

ReLU

0 if $x \leq 0$. so 0

for sample 2

1 + 0 → 1

ReLU

x, $x > 0$ so 1

for sample 3

1 + 0 → 1

ReLU

x, $x > 0$ so 1

for sample 4

2 + 0 → 2

x, $x > 0$ so 2

h2

for sample 1

0 − 1 → −1

ReLU

0 if $x \leq 0$. so 0

for sample 2

1 − 1 → 0

ReLU

0, $x \leq 0$ so 0.

for sample 3

1 − 1 → 0

ReLU

0, $x \leq 0$ so 0

for sample 4

2 − 1 → 1

x, $x > 0$. so 1

Hence $H = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$ (After applying ReLU)

for Y.

$H \times W \Rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 \end{bmatrix}^T$

$\Rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ ← for Output layer.

Ground Truth values → $Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ $Y_{predicted} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

$MSE = \frac{1}{2}(\hat{y} - y)^2 \Rightarrow \frac{1}{2}(0-0) \Rightarrow 0$

$\Rightarrow \frac{1}{2}(1-1) \Rightarrow 0$

$\Rightarrow \frac{1}{2}(1-1) \Rightarrow 0$

$\Rightarrow \frac{1}{2}(0-0) = 0$

as y predicted and ground truth values are same Loss is 0.

Loss is 0. ~~$\frac{dL}{dy} = 0$~~.

with respect to output layer weights

$\frac{dL}{dw} = H^T \times \frac{dL}{dY_{true}} = H^T \times 0 \Rightarrow \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$= \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\frac{dL}{dH} = \frac{dL}{dy_{true}} \times W^T = 0 \times [1 \ -2] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

with respect hidden activations

(6) If $x = [2\ 4\ 6\ 8\ 10]$, gamma = 2 and $\beta = -1$ perform batch norm of all these 4 samples and write the output.

$x = [2\ 4\ 6\ 8\ 10]$

$\mu = \frac{2+4+6+8+10}{5} \Rightarrow 30/5 \Rightarrow 6$

$\sigma = \frac{1}{5}\sum_{i=1}^{5}(x-\mu)^2$

$(2-6)^2 = (-4)^2 = 16$

$(4-6)^2 = (-2)^2 = 4$

$(6-6)^2 = 0$

$(8-6)^2 = (2)^2 = 4$

$(10-6)^2 = (4)^2 = 16$

adding these

$16+4+0+4+16 = 40$

$\sigma = \frac{16+4+0+4+16}{5}$

$= \frac{40}{5} = 8$

$\hat{z}_1 = \frac{x_1-\mu}{\sigma} = \frac{2-6}{8} = \frac{-4}{8} = -0.5$

$\hat{z}_2 = \frac{4-6}{8} = \frac{-2}{8} = -0.25$

$\hat{z}_3 = \frac{6-6}{8} = 0$

$\hat{z}_4 = \frac{8-6}{8} = \frac{2}{8} = 0.25$

$\hat{z}_5 = \frac{10-6}{8} = \frac{4}{8} = 0.5$

calculating 2 scores for each sample.

Let $\gamma = 2$ $\beta = -1$ $\quad \tilde{z} = \gamma \hat{z} + \beta$

$\tilde{z}_1 = 2(-0.5) + (-1)$

$= -1 - 1 \Rightarrow -2$

$\tilde{z}_2 = 2(-0.25) - 1 \Rightarrow -0.5 - 1 \Rightarrow -1.5$

$\tilde{z}_3 = 2(0) - 1 \Rightarrow 0 - 1 \Rightarrow -1$

$\tilde{z}_4 = 2(0.25) - 1 \Rightarrow 0.5 - 1 \Rightarrow -0.5$

$\tilde{z}_5 = 2(0.5) - 1 \Rightarrow 1 - 1 \Rightarrow 0$

8. For the following image and a filter, apply the convolution operation and write the output image. What did the convolution operation achieve in this example?

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Input image

$6 \times 6 = 36$

$*$

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

Filter

$3 \times 3 = 9$

$=$

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

$\Rightarrow 10+0-10+10+0-10+10+0-10 \Rightarrow 0$

$\Rightarrow 10+0-0+10+0-0+10+0-0 \Rightarrow 30$

$\Rightarrow 10+0-0+10+0-0+10+0-0 \Rightarrow 30$

$\Rightarrow 0+0-0+0+0-0+0+0-0 \Rightarrow 0$

$\Rightarrow 10+0-10+10+0-10+10+0-10 \Rightarrow 0$

$\Rightarrow 10+0-0+10+0-0+10+0-0 \Rightarrow 30$

$\Rightarrow 10+0-0+10+0-0+10+0-0 \Rightarrow 30$

$\Rightarrow 0+0-0+0+0-0+0+0-0 \Rightarrow 0$

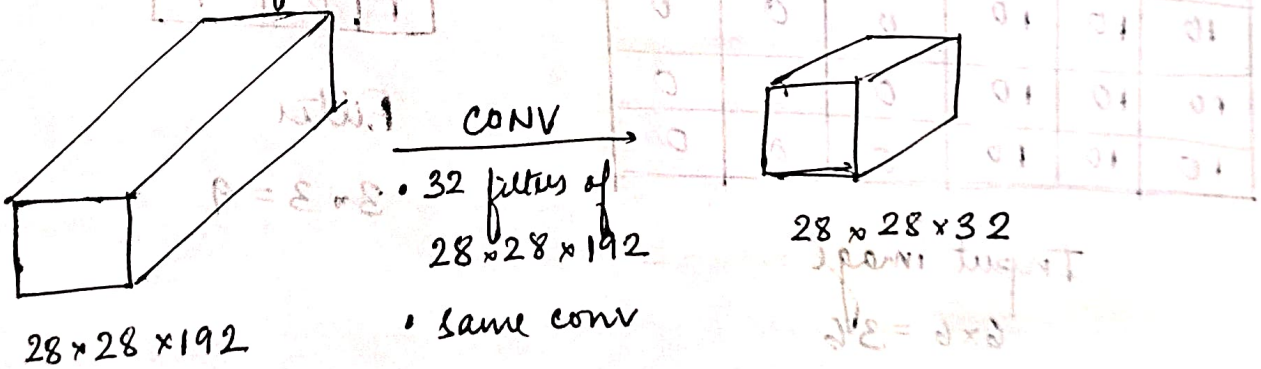
$\Rightarrow 10+0-10+10+0-10+10+0-10 \Rightarrow 0$

$\Rightarrow 10+0-0+10+0-0+10+0-0 \Rightarrow 30$

$\Rightarrow 10+0-0+10+0-0+10+0-0 \Rightarrow 30$

$\Rightarrow 0+0-0+0+0-0+0+0-0 \Rightarrow 0$.

(13) For the following CONV operation, compute the computational cost (number of multiplication for example)



Design a new network architecture using 1×1 conv layers to reduce the computational cost. Show the savings from your architecture. Please note that your new proposed architecture should take the same sized input image (28×28×192) and ~~output input~~ the image of the same size shown in the above diagram (28×28×32)

Input - 28x28x192

Filter - 32 each of size of 28x28x192

Output - 28x28x32

Each filter is applied across all 192 channels.

Each filter on one channel of the input image :- 28x28x28x28

= 614656

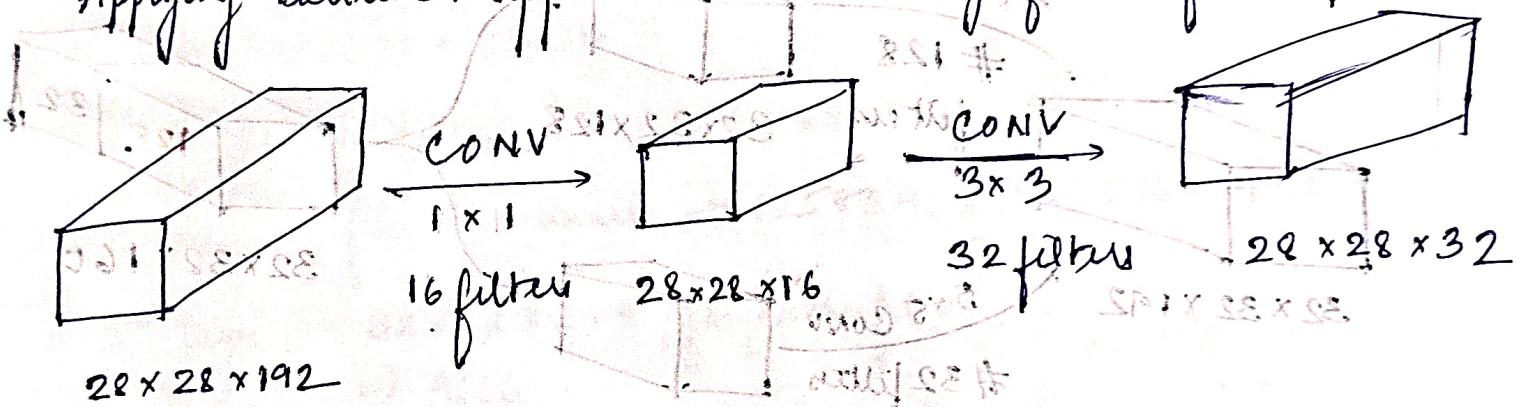
For 32 filters = 614656 x 32 ⇒ 19668992

for 192 channels :- 19668992 x 192 = 3,776,446,464

≃ 3 billion

Architecture with reduced multiplications:

Applying Bottleneck approach. 1x1 CONV layer for the first layer.



Calculations:-

First layer → 1x1 filter x 28x28 input image = 784

for 192 channels = 784 x 192 = 150528

for 16 filters = 2,408,448

2nd layer → 28x28 input image x 3x3 filter size = 7056

for 16 channels = 112,896

for 32 filters = 3,612,672

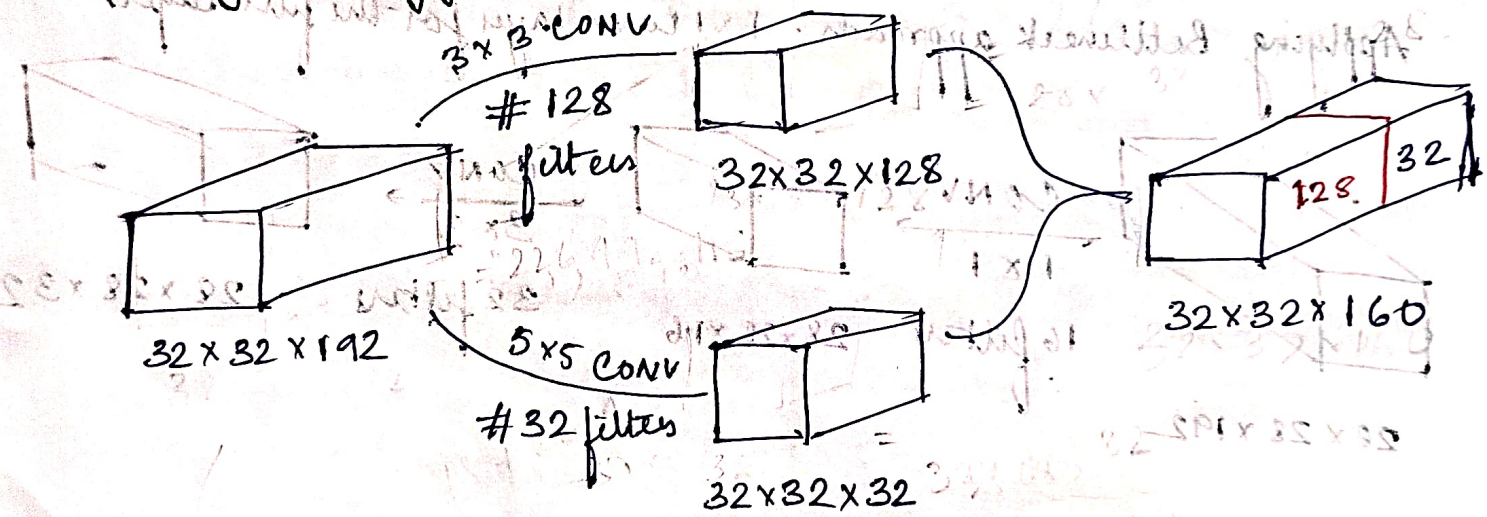
Adding the outputs = 2408448 + 3612672

= 6,021,120 ≃ 6 million

(14) I would like to construct an inception module with 128 × 3 × 3 CONV filters, 32 5×5 CONV filters, same convolutions, stride = 1. The input to the inception module (coming from previous layer output activation) size is 32 × 32 × 192 and the output image is 32 × 32 × 160.

a. What will be the number of multiplications in the above inception module

b. Provide a strategy to reduce the number of multiplications and write down the no. of multiplications after adopting your strategy.



First branch → 32×32× filter dimensions = 32×32×3×3
= 9216

for 128 filters = 9216 × 128
= 1179648

for 192 channels = 1179648 × 192
= 226,492,416

Second branch → 32×32×5×5 = 25,600

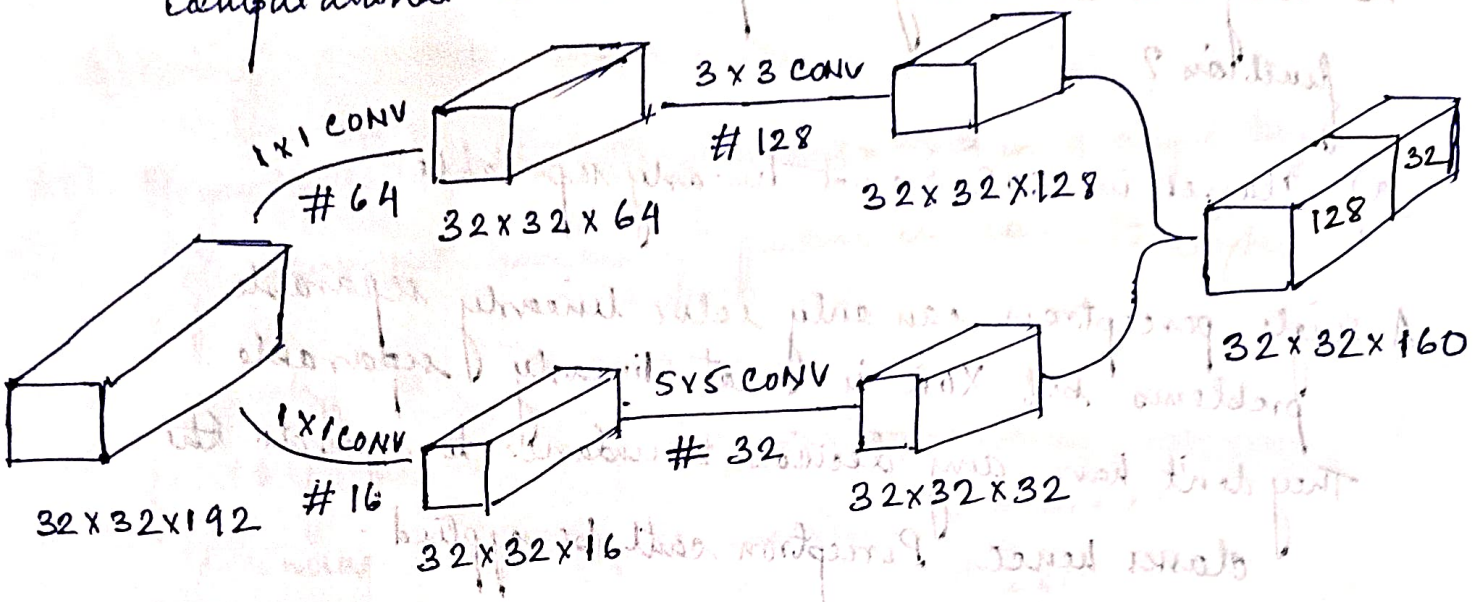
for 32 filters = 25600 × 32 = 819200

for 192 channels = 157,286,400

Concatenating the two = 226,492,416 + 157,286,400

⇒ 383,778,816 (Ans)

Constructing a structure to reduce the multiplications and computational cost.



Calculations :- 1st branch 1st layer :-

each filter → 32x32x1x1 = 1024

for 64 filters → 1024 x 64 ⇒ 65,536

for 192 channels → 12582912 (65536 x 192)

2nd layer :- 32x32x3x3 (for each filter)

⇒ 9216

for 128 filters ⇒ 9216 x 128 ⇒ 1179648

for 64 channels ⇒ 1179648 x 64 = 75497472

2nd branch 1st layer :- 32x32x1x1 ⇒ 1024

(1024 x 16) → for 16 filters ⇒ 16384 → 16384 x 192 channels ⇒ 3145728

2nd layer :- 5x5x32x32 ⇒ 25600

for 32 filters ⇒ 25600 x 32 ⇒ 819200

for 16 channels ⇒ 819200 x 16 ⇒ 13,107,200

Concatenate ⇒ 75,497,472 + 13,107,200 ⇒ 88,604,672 (Ans)

WORKSHEET CONTINUED :-

(7) What is the Role of Keep ~~Rate~~ Probability in ~~Dropout~~ the context of dropout & and why scaling the activations in "keep probability" is necessary?

Ans :- Keep Probabilities are used in Dropouts which is a type of regularization technique where we randomly drop neurons from a layer.

The fraction of neurons kept active :- Keep probability (P)

for example :- if $p=0.8$ then for each sample 80% of the neurons are kept and the rest 20% are dropped.

Draw the example from B2.

(9) What is the impact of increasing the number of convolutional layers in a CNN?

Ans :- Advantages :

(1.) Hierarchical Feature Learning

- Shallow CNN - detects very simple features (edges, corners, color blobs)
- Deep CNN - detects semantic structures
- Very Deep CNN (eg ResNet)
  - can learn highly abstract task specific features.

(2.) But Accuracy - More the layers better the performance of the model as it will capture more details.

More CNN layers increase the model's capacity to fit complex data distributions, potentially leading to higher accuracy.

Disadvantages

(1) Vanishing/Exploding Gradients — With many layers, gradients might shrink/explode while backpropagating. The gradients will eventually stop updating and the initial layers will stop learning.
(For this problem we use ResNet, skip connections)

(2) Overfitting — If the dataset is small, adding many layers makes the model learn only the training data causing overfitting (can also happen if proper regularization is not done).

(15) You have to develop a 5-way classifier to detect objects in an image. You are given a gray scale input image of size 32×32. Which of the following approaches is better and why?

a. Flatten the image (32×32) to form a 1024 vector and pass this to a fully connected layer with 5 neurons.

b. Directly pass the image to a CNN layer having five 32×32 filters.

1. Fully Connected Layer (Flattening)
The 32×32 grayscale image is flattened into a 1024-dimensional vector and passed to a layer with 5 neurons.
This treats all pixels equally without exploiting their spatial arrangement.
Results in large no. of parameters and higher risk of overfitting. (especially with small datasets or images)

2. CNN layer

- CNNs use parameter sharing and local connectivity allowing filters to learn local patterns (edges, textures, shapes) and build hierarchical feature representations.
- This gives better performance on images and fewer parameters compared to a fully connected layer for the same problem.

CNN is better :-

Preserves ~~better~~ spatial information

Parameter efficiency

Feature hierarchy.