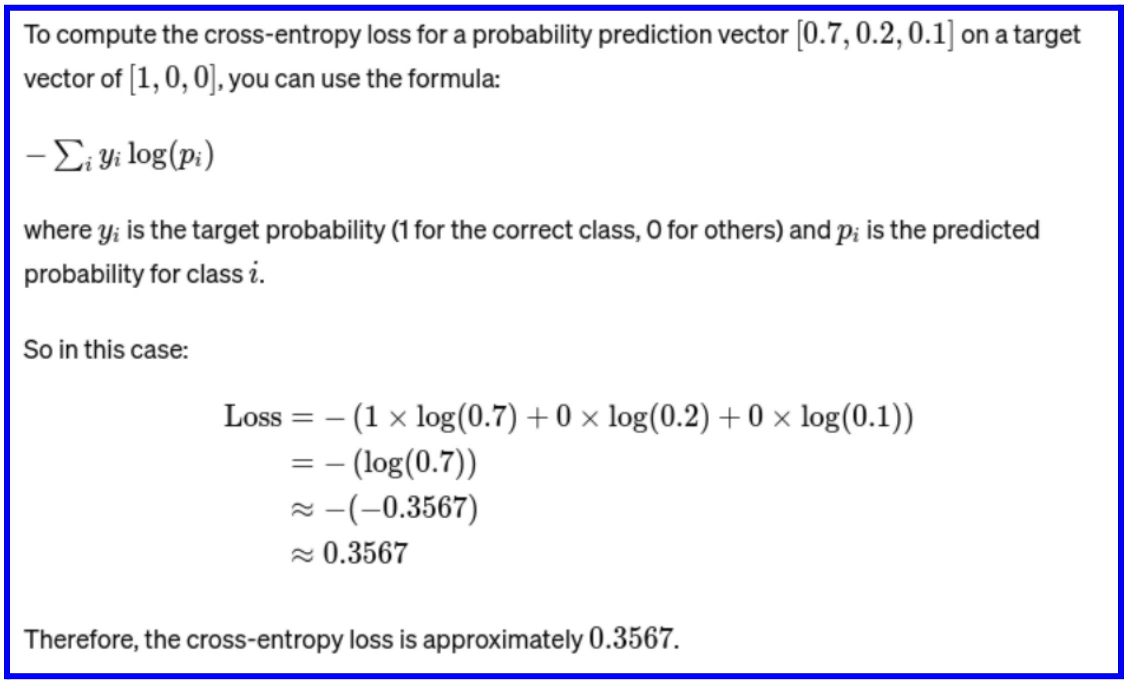
CS671: Mid-Sem Exam
(JAN-JUNE 2025)

Date: 04-Apr-2025

---

1. Compute the cross-entropy loss for a probability prediction vector [0.7,0.2,0.1] on a target vector of [1,0,0]. **[1 mark]**



To compute the cross-entropy loss for a probability prediction vector $[0.7, 0.2, 0.1]$ on a target vector of $[1, 0, 0]$, you can use the formula:

$-\sum_i y_i \log(p_i)$

where y_i is the target probability (1 for the correct class, 0 for others) and p_i is the predicted probability for class i.

So in this case:

$$\begin{aligned}\text{Loss} &= -(1 \times \log(0.7) + 0 \times \log(0.2) + 0 \times \log(0.1)) \\ &= -(\log(0.7)) \\ &\approx -(-0.3567) \\ &\approx 0.3567\end{aligned}$$

Therefore, the cross-entropy loss is approximately 0.3567.

---

2. Write the correct sequence of the steps to use a gradient descent algorithm. **[1 mark]**

a. Calculate the error between the actual value and the predicted value.
b. Reiterate until you find the best weights of the network.
c. Pass an input through the network and get values from the output layer.
d. Initialize the weights and biases of the model (network).
e. Go to each neuron that contributes to the error and change its respective value to reduce the error.

The correct sequence is: d, c, a, e, b

---

3. Suppose you are convolving an image using a square filter W. The convolution operation on each pixel computes the feature value Î(x, y) of pixel I(x, y) using the following formula: **[1 mark]**

$$\hat{\mathbf{I}}(x,y) = \sum_{i=-3}^{3}\sum_{j=-3}^{3} \mathbf{I}(x-i, y-j) * W(i,j)$$

What is the height/ width of the filter W? (Need NOT to show the computation.)

The filter considers three pixels left and three right (or three up and three down) of the centre pixel. Hence, the width/ height of the filter is 3+1+3=7.

---

4. The following is a jumble of steps to utilize an autoencoder for the classification task: **[1 mark]**

a. Train the added classification layer using the encoded features and class labels.
b. Detach the encoder and bottleneck layer from the decoder.
c. Add a classification layer on top of the pre-trained bottleneck layer.
d. Train an autoencoder on the input data without considering the class labels.
e. Fine-tune the decoder of the autoencoder after adding the classification layer on top.

The correct order is: d, b, c, a, e

---

5. What is the issue with the MSE loss function for reconstruction problems? *(Write precisely in not more than 2 lines.)* **[1 mark]**

MSE loss leads to blurry reconstructions because it minimizes pixel-wise differences, averaging multiple plausible outputs. It also penalizes larger errors more than smaller ones, making it sensitive to outliers.

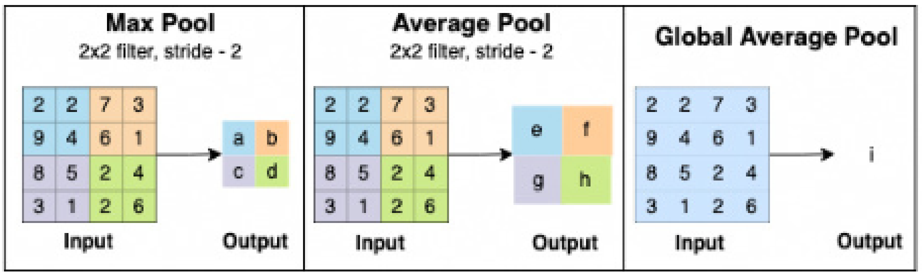
---

6. Given a kernel of size 5x5, what will be the padding size to keep the feature size the same as input. **[1 mark]**

The padding size should be 2 (on all sides).

---

7. Given the following Figure:



What is the value of a, b, c, d, e, f, g, h, i ? **[1 mark]**

a: 9 b: 6 c: 8 d: 6 e: 4 f: 4 g: 4 h: 3 i: 4

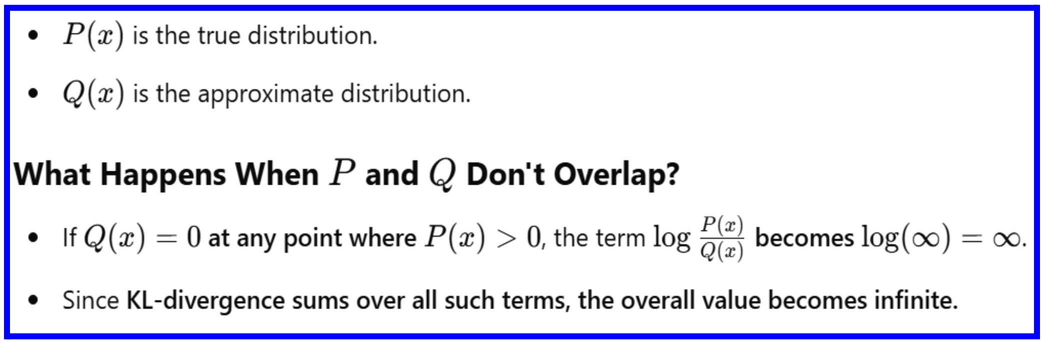
---

8. What is the difference in the outputs if on a 7x7x128 dimension image, 32 of 3x3x128 kernels are applied v/s 32 of 1x1x128 kernels are applied? (Consider padding that doesn't change the output spatial dimension.) **[1 mark]**

Output Size (Same Padding): $7 \times 7 \times 32$ in both the cases. Thus, 3×3 kernels focus on **spatial features**, while 1×1 kernels perform **channel-wise transformations** like in **depth-wise separable convolutions or bottleneck layers.**

---

9. What will be the KL divergence for 2 distributions which do not overlap at all? **[1 mark]**



- $P(x)$ is the true distribution.
- $Q(x)$ is the approximate distribution.

What Happens When P and Q Don't Overlap?

- If $Q(x) = 0$ **at any point where** $P(x) > 0$, the term $\log \frac{P(x)}{Q(x)}$ **becomes** $\log(\infty) = \infty$.
- Since **KL-divergence sums over all such terms, the overall value becomes infinite.**

---

10. What is the output of the discriminator when GAN has properly converged? **[1 mark]**

It outputs D(x)≈0.5 for both real and fake samples, meaning it is maximally uncertain.

11. Assume you have a dataset with 10,000 samples, and you are training a model using mini-batch gradient descent with a batch size of 125. In each epoch, how many times will the gradient be updated? **[1 mark]**

Ans: 10,000/125=80

---

12. Given a fully connected layer with 128 input neurons and 64 output neurons, calculate the number of floating-point operations (FLOPs) required for one forward pass (ignoring activation functions and biases). Assume a single floating-point operation is one multiplication or one addition. **[1 mark]**

Ans: Each output neuron requires 128 multiplications and 127 additions (for a total of 255 operations per neuron). Since there are 64 output neurons, the total number of FLOPs is 64×255=16,38464 \times 255 = 16,38464×255=16,384.

---

13. Consider a neural network where the output is computed as:

$$z=\text{softmax}(f_\theta(X))$$

where f_θ is an MLP. We want to discretise it into a one-hot vector before passing it to another fully connected layer g_ϕ, as follows:

$$y=g_\phi(\text{one\_hot}(\text{softmax}(f_\theta(X))))$$

Is there a problem with this formulation? Briefly justify your answer. **[1 mark]**

Ans:Yes, there is a problem with this formulation. The main issue is that the `one_hot(softmax(fθ(X)))` operation is non-differentiable, which means gradients cannot flow through it during backpropagation.

---

14. Consider a fully connected neural network (FCNN) with the following architecture designed for text classification:
Input: A 300-dimensional word embedding vector
Hidden layer 1: 512 output neurons, using ReLU activation
Hidden layer 2: 256 output neurons, using ReLU activation
Output layer: 5 neurons (for sentiment classification with 5 classes), using softmax activation
What is the total number of trainable parameters in this FCNN? **[2 mark]**

Ans: Input → Hidden 1:
Weights = `300 × 512` = 153,600

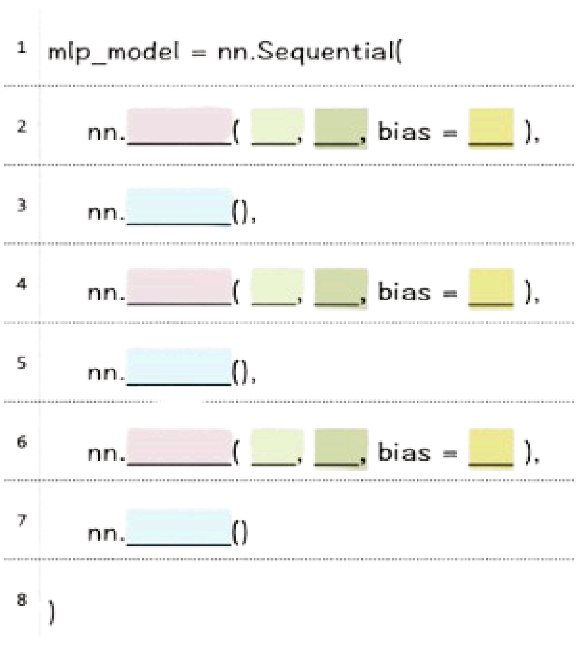
Biases = 512
Total = 153,600 + 512 = 154,112
Hidden 1 → Hidden 2:
Weights = 512 × 256 = 131,072
Biases = 256
Total = 131,072 + 256 = 131,328

Hidden 2 → Output:
Weights = 256 × 5 = 1,280
Biases = 5
Total = 1,280 + 5 = 1,285

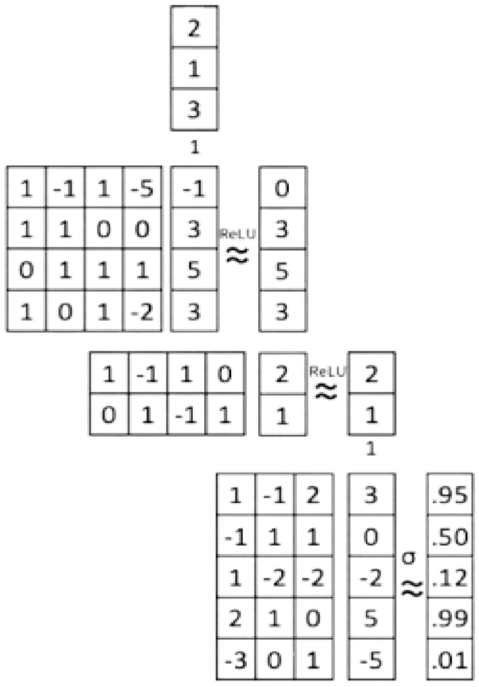
Total trainable parameters = 154,112 + 131,328 + 1,285 = 286,725

---

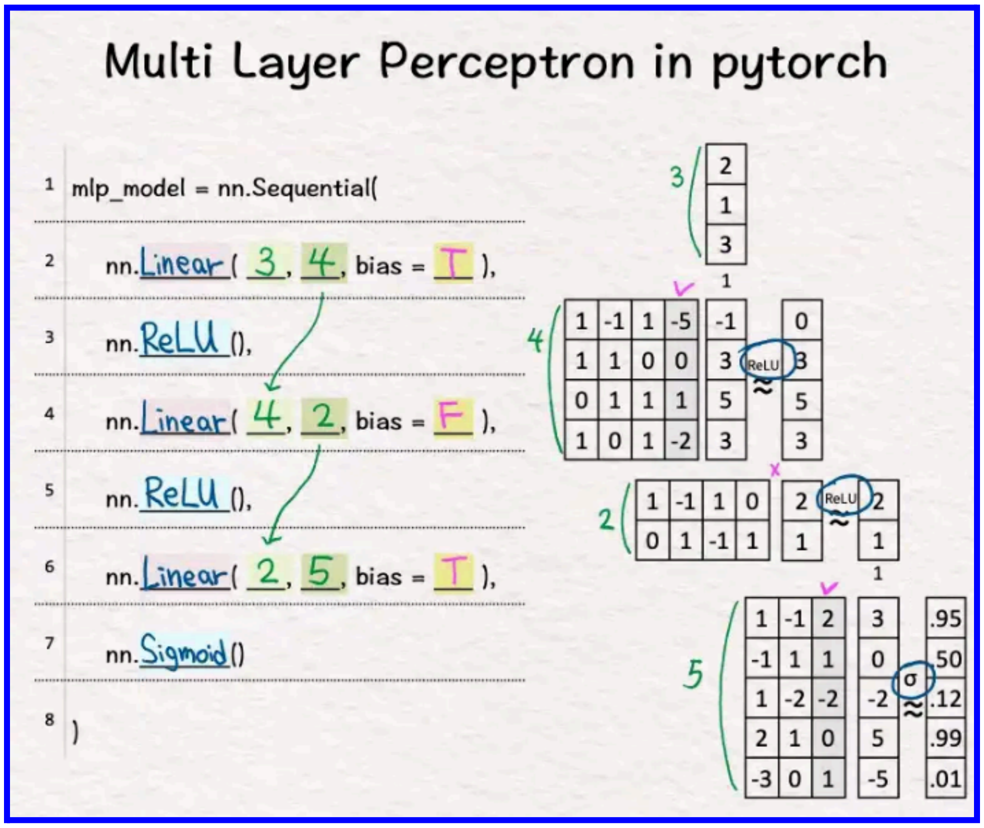
15. Given the following figure showing the forward pass of a Multi-Layer Perceptron (MLP) with three layers. Fill in the blanks which are the missing parts of the code (using the pytorch **nn** module) to correctly implement the forward pass. [Hint: Bias = T/F] **[5 mark]**



```
mlp_model = nn.Sequential(
    nn.________( ___, ___, bias = ___ ),
    nn.________(),
    nn.________( ___, ___, bias = ___ ),
    nn.________(),
    nn.________( ___, ___, bias = ___ ),
    nn.________()
)
```



Ans:



Multi Layer Perceptron in pytorch

```
mlp_model = nn.Sequential(
    nn.Linear( 3, 4, bias = T ),
    nn.ReLU (),
    nn.Linear( 4, 2, bias = F ),
    nn.ReLU (),
    nn.Linear( 2, 5, bias = T ),
    nn.Sigmoid()
)
```

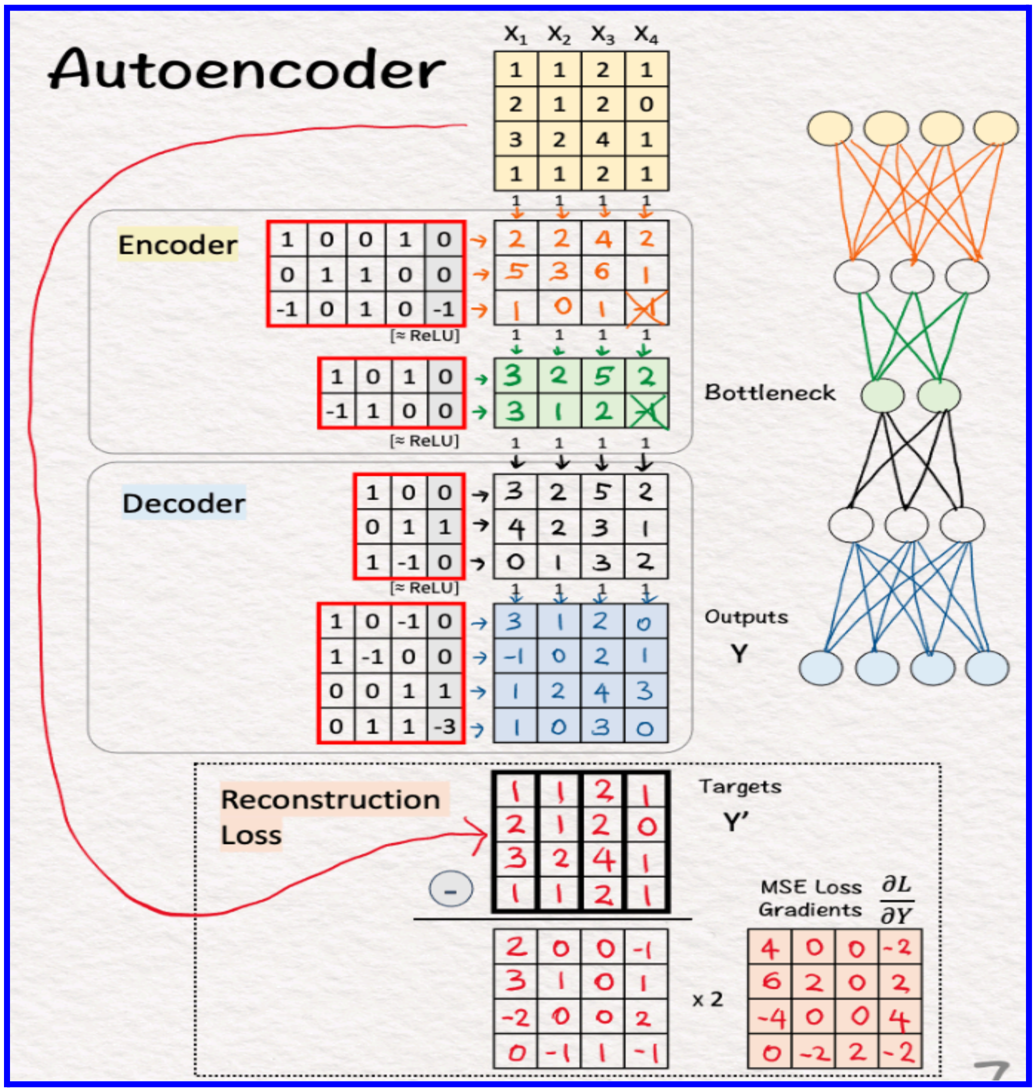
$$3\begin{pmatrix}2\\1\\3\end{pmatrix}\quad 1$$

$$4\begin{pmatrix}1 & -1 & 1 & \overset{\checkmark}{-5}\\ 1 & 1 & 0 & 0\\ 0 & 1 & 1 & 1\\ 1 & 0 & 1 & -2\end{pmatrix}\begin{bmatrix}-1\\3\\5\\3\end{bmatrix}\overset{\text{ReLU}}{\approx}\begin{bmatrix}0\\3\\5\\3\end{bmatrix}$$

$$2\begin{pmatrix}1 & -1 & 1 & \overset{\times}{0}\\ 0 & 1 & -1 & 1\end{pmatrix}\begin{bmatrix}2\\1\end{bmatrix}\overset{\text{ReLU}}{\approx}\begin{bmatrix}2\\1\end{bmatrix}\quad 1$$

$$5\begin{pmatrix}1 & -1 & \overset{\checkmark}{2}\\ -1 & 1 & 1\\ 1 & -2 & -2\\ 2 & 1 & 0\\ -3 & 0 & 1\end{pmatrix}\begin{bmatrix}3\\0\\-2\\5\\-5\end{bmatrix}\overset{\sigma}{\approx}\begin{bmatrix}.95\\.50\\.12\\.99\\.01\end{bmatrix}$$

16. Given four training examples X1, X2, X3, X4 and a set of weights, an autoencoder processes them through an encoder, bottleneck, and decoder. Compute the output Y by applying matrix multiplication and ReLU activation at each layer. Then, calculate the Mean Squared Error (MSE) loss with targets Y′ and find the gradient using 2×(Y−Y′). **[10 mark]**



17. Consider a Variational Autoencoder (VAE) trained to reconstruct input data X1, X2 and X3 using an encoder with ReLU activation, a latent space defined by mean μ and standard deviation σ, the reparameterization trick of z=μ+σ·ϵ and a decoder with two layers. The given network has 2-layered encoder and decoder. It is trained by minimizing the MSE-based reconstruction loss (say, L_1) and the KL divergence loss (say, L_2).

i) The closed form expression of the KL-divergence is as follows. Derive the expressions of the loss gradient with respect to μ and σ.

$$L_2 = KL\left(q(z|x_i)||p(z)\right) = KL\left(N(\mu,\Sigma)||N(0,1)\right)$$

$$= -\frac{1}{2}\sum_j (1 + log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

Ans:

For a batch of j samples gradient of $\mathcal{L}_2$ is computed with respect to each μ_j and σ_j, approximating the distribution of every sample j.

$$\therefore \frac{\partial \mathcal{L}_2}{\partial \mu_j} = \frac{\partial}{\partial \mu_j}\left[-\frac{1}{2}\sum_j \left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right)\right]$$

$$= \frac{\partial}{\partial \mu_j}\left[-\frac{1}{2}\left(0 - 2\mu_j - 0\right)\right]$$

[w.r.t. μ_j, loss gradient for index $\neq j$ is 0. $\therefore$ Summation is no more required

$$= \mu_j,$$

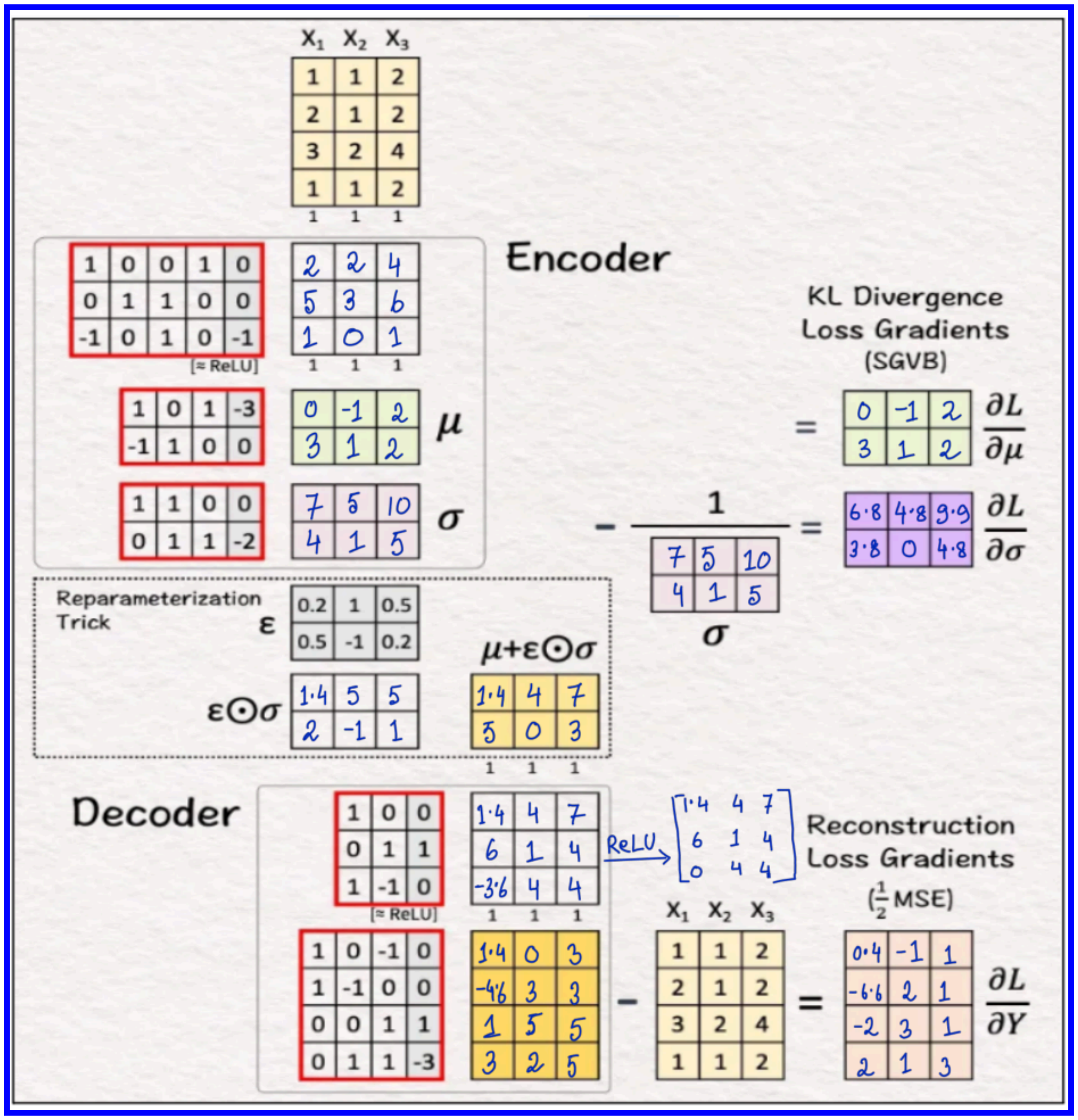
and $$\frac{\partial \mathcal{L}_2}{\partial \sigma_j} = \frac{\partial}{\partial \sigma_j}\left[-\frac{1}{2}\sum_j \left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right)\right]$$

$$= -\frac{1}{2}\left(\frac{2\sigma_j}{\sigma_j^2} - 0 - 2\sigma_j\right)$$

$$= -\frac{1}{\sigma_j} + \sigma_j = \sigma_j - \frac{1}{\sigma_j}.$$

ii) Perform the forward pass in the previous figure of the given VAE and compute the reconstructed output and evaluate the total loss for a given input. **[2+10 mark]**

Ans:



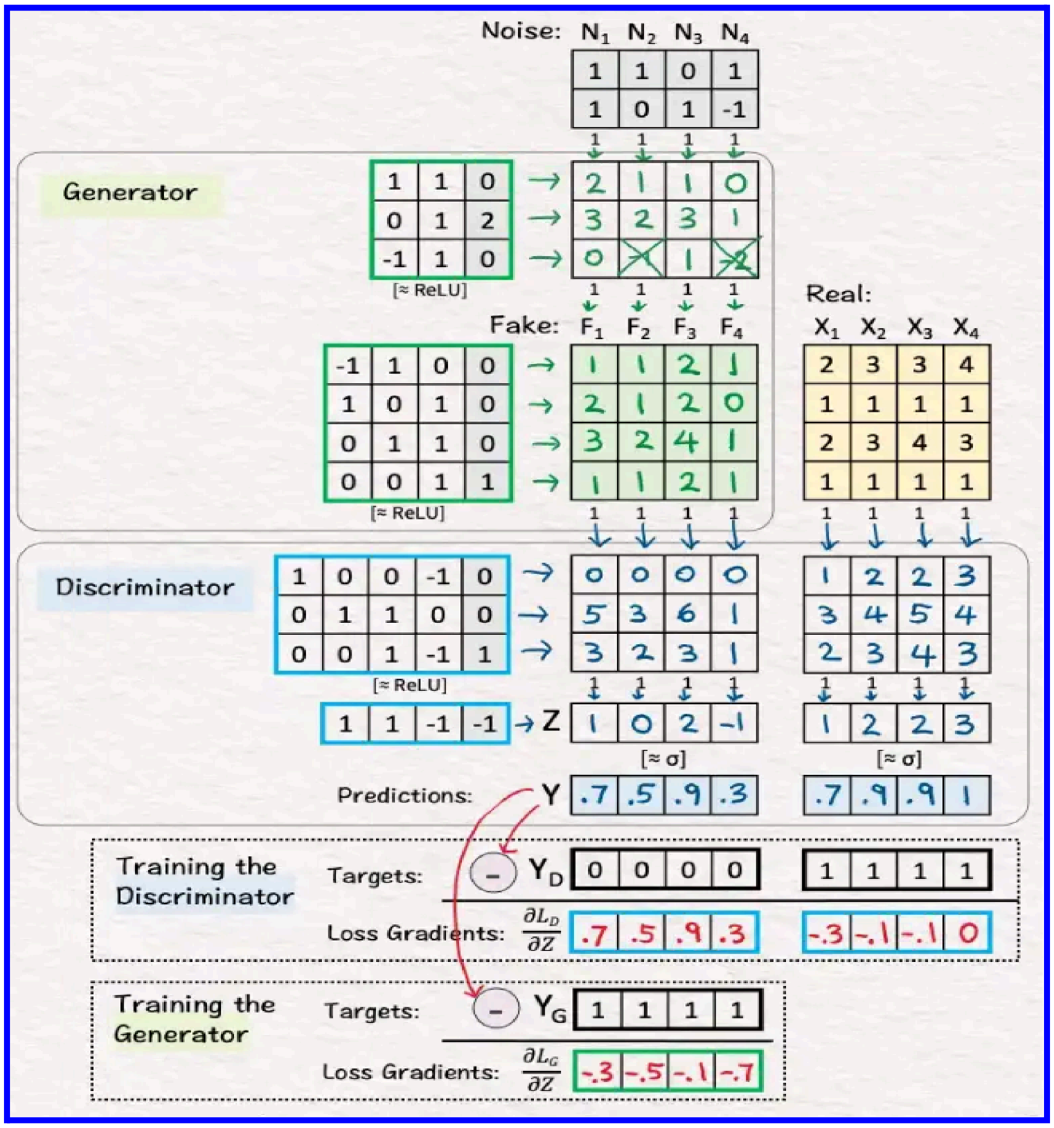
18. Given in the following figure is a Generative Adversarial Network (GAN) which is designed for the goal stated as:

To generate realistic 4D data from 2D noise.

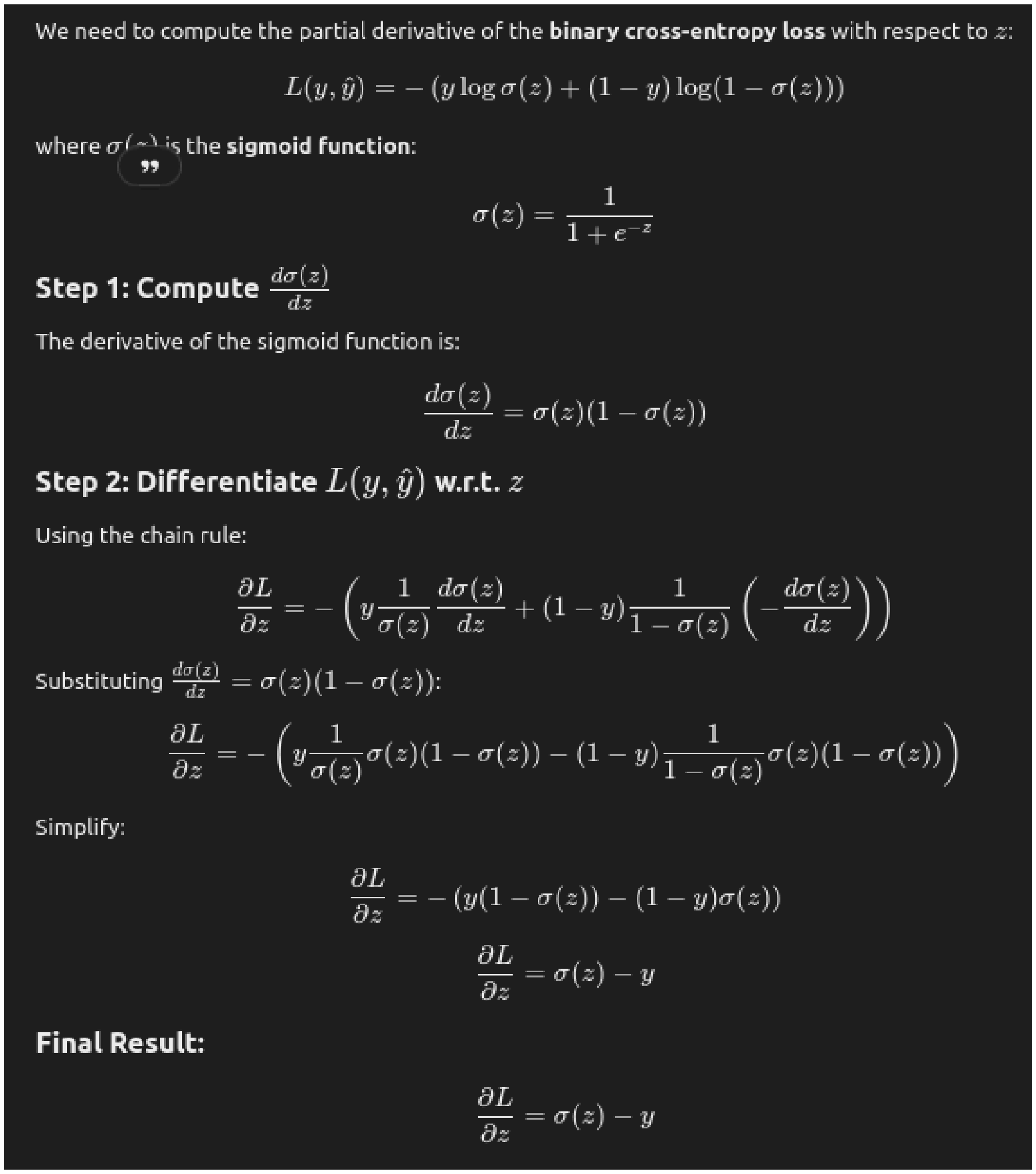
(Part-1): The Generator consists of two linear layers with ReLU activations, transforming 2D noise vectors into 4D "fake" data. The Discriminator consists of two linear layers followed by a sigmoid activation, classifying inputs as real or fake. Find the *predictions* generated by the Discriminator for the 4 real samples and 4 fake examples, given to the Discriminator. **[8 marks]**

(Part-II): During training, the Discriminator minimizes the binary cross-entropy loss by distinguishing real data from fake data, while the Generator aims to fool the Discriminator by maximizing its error. Given this setup, compute the loss gradients of the

Discriminator and Generator for the given batch of 4 fake and 4 real images. ***(Use Hint1, Hint2 given to solve. Please use the space and write the answer clearly in the figure only.)*** **[5+5 marks]**



(Part-II)



We need to compute the partial derivative of the **binary cross-entropy loss** with respect to z:

$$L(y,\hat{y}) = -\left(y\log\sigma(z) + (1-y)\log(1-\sigma(z))\right)$$

where $\sigma(z)$ is the **sigmoid function**:

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

Step 1: Compute $\frac{d\sigma(z)}{dz}$

The derivative of the sigmoid function is:

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$$

Step 2: Differentiate $L(y,\hat{y})$ w.r.t. z

Using the chain rule:

$$\frac{\partial L}{\partial z} = -\left(y\frac{1}{\sigma(z)}\frac{d\sigma(z)}{dz} + (1-y)\frac{1}{1-\sigma(z)}\left(-\frac{d\sigma(z)}{dz}\right)\right)$$

Substituting $\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$:

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

Simplify:

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

$$\frac{\partial L}{\partial z} = \sigma(z) - y$$

Final Result:

$$\frac{\partial L}{\partial z} = \sigma(z) - y$$

Source: https://www.byhand.ai/p/10-can-you-calculate-a-generative