1 (a) (i) (2 x 5) + 5 + (5 x 5) + 5 = 45

(ii) [[[1, 2, 3]], [[4, 5, 6]]]

(iii) It is not a good idea. Constant initialization can result in symmetrical weight updates, causing neuron to learn the same features. (or it can lead to exploding/vanishing gradient problem, an alternative better method is to use Xavier initialization)

(iv) Yes. High training loss suggests underfitting. A larger dataset may provide more diverse examples to help the model learn a better representation, reducing the training and test loss. (Essential to note that the quality, representativeness of the dataset is also crucial)

(v) 1. Inadequate model complexity as the model is too simple for the complexity of the given task. It is unable to capture any underlying pattern.

2. Learning rate choice is inappropriate. Updates are too small to provide any meaningful changes.

Editor's note: In the exam, I also mentioned the possibility of improper weight initialization to be 0 leading to no gradient updates during backpropagation.

(vi) 1. Using SGD / Mini Batch Gradient Descent

2. Using Adaptive Learning Rate optimizers such as Adam, Adagrad

(vii) Smaller batch size allows for more frequent weight updates which may lead to faster convergence. However, the gradient estimates used to update the model parameters are noisier and may conversely also lead to more erratic updates

(viii) Increase the number of parameters in the neural network and train the neural network for more epochs

(b) (i) $\nabla_y J = -2(d - y)$

$\nabla_W J = -2(d - y)x^T$

$\nabla_b J = -2(d - y)$

(ii) $\nabla_W J_1 = -2(d-y)x^T + 2\beta W$

$\nabla_b J_1 = -2(d-y)x^T$

2 (a)

$$W_1 = \begin{matrix} 3.2 & -2.4 & -2.1 \\ -1.2 & -1.4 & 2.6 \end{matrix} \qquad b_1 = \begin{matrix} 0.5 \\ 0.5 \\ 0.5 \end{matrix}$$

$$W_2 = \begin{matrix} 3.0 & 0 \\ -2.4 & -4.2 \\ -3.0 & 2.5 \end{matrix} \qquad b_2 = \begin{matrix} 0.5 \\ 0.5 \end{matrix}$$

$$W_3 = \begin{matrix} 2.0 \\ -1.0 \end{matrix} \qquad b_3 = 0.5$$

(b)

$$x_1 = \begin{matrix} 1.0 \\ 2.0 \end{matrix}$$

$$u = W^T x + b$$

$$u_1 = W_1^T \times x_1 + b_1 = \begin{matrix} 3.2 & -1.2 \\ -2.4 & -1.4 \\ -2.1 & 2.6 \end{matrix} \times \begin{matrix} 1.0 \\ 2.0 \end{matrix} + \begin{matrix} 0.5 \\ 0.5 \\ 0.5 \end{matrix} = \begin{matrix} 1.3 \\ -4.7 \\ 3.6 \end{matrix}$$

$$h_1 = g(u_1) = \tanh(u_1) = \begin{matrix} 0.86172 \\ -0.99983 \\ 0.99850 \end{matrix} = \begin{matrix} 0.86 \\ -1.00 \\ 1.00 \end{matrix}$$

$$u_2 = W_2^T \times h_1 + b_2 = \begin{matrix} 3.0 & -2.4 & -3.0 \\ 0 & -4.2 & 2.5 \end{matrix} \times \begin{matrix} 0.86 \\ -1.00 \\ 1.00 \end{matrix} + \begin{matrix} 0.5 \\ 0.5 \end{matrix} = \begin{matrix} 2.49 \\ 7.20 \end{matrix}$$

$$h_2 = g(u_2) = \tanh(u_2) = \begin{matrix} 0.99 \\ 1.00 \end{matrix}$$

. **(c)** $u_3 = W_3^T \times h_2 + b_3 = 1.48$

$y = f(u_3) = sigmoid(u_3) = 0.81$

Cross Entropy = -log(1-y) = 1.68

(d) $y = f(u) = \tanh(u)$

$f'(u) = 1 - y^2$

(e) $\nabla_{u_3} J = -\big(d - f(u)\big) = 0.81$

$$\nabla_{u_2} J = W_3 \nabla_{u_3} J \cdot h_2'(u) = \begin{matrix} 2.0 \\ -1.0 \end{matrix} \times 0.81 \cdot (1 - h_2^2) = \begin{matrix} 1.62 \\ -0.81 \end{matrix} \cdot \begin{matrix} 0.03 \\ 0 \end{matrix} = \begin{matrix} 0.04 \\ 0 \end{matrix}$$

$$\nabla_{u_1} J = W_2 \nabla_{u_2} J \cdot h_1'(u) = \begin{matrix} 3.0 & 0 \\ -2.4 & -4.2 \\ -3.0 & 2.5 \end{matrix} \times \begin{matrix} 0.04 \\ 0 \end{matrix} \cdot (1 - h_1^2) = \begin{matrix} 0.03 \\ 0 \\ 0 \end{matrix}$$

(f) $$\nabla_{W_3} J = h_2 \nabla_{u_3} J^T = \begin{matrix} 0.80 \\ 0.81 \end{matrix}$$

$\nabla_{b_3} J = \nabla_{u_3} J = 0.81$

$$\nabla_{W_2} J = h_1 \nabla_{u_2} J^T = \begin{matrix} 0.86172 \\ -0.99983 \\ 0.99850 \end{matrix} \times \begin{matrix} 0.04 & 0 \end{matrix} = \begin{matrix} 0.04 & 0 \\ -0.04 & 0 \\ 0.04 & 0 \end{matrix}$$

$$\nabla_{b_2} J = \nabla_{u_2} J = \begin{matrix} 0.04 \\ 0 \end{matrix}$$

$$\nabla_{W_1} J = x \nabla_{u_1} J^T = \begin{matrix} 1.0 \\ 2.0 \end{matrix} \times \begin{matrix} 0.03 & 0 & 0 \end{matrix} = \begin{matrix} 0.03 & 0 & 0 \\ 0.07 & 0 & 0 \end{matrix}$$

$$\nabla_{b_1} J = \nabla_{u_1} J = \begin{matrix} 0.03 \\ 0 \\ 0 \end{matrix}$$

<u>Editor's note:</u> I used 5 DP for my calculations, however I discussed with some of my friends who used the 2 DP values to calculate further steps and our answers when rounded differed by about +-0.01, so take that with a pinch of salt

3 (a) (i) $n_1 = 6, d_1 = 12, F_1 = 7, p_2 = 1$

(ii) Convolutional Layer 1 = 6 x 12 x 7 x 7 + 6 = 3534

Convolutional Layer 2 = 4 x 6 x 3 x 3 + 4 = 220

Fully Connected Layer = 4 x 25 x 25 x 128 x 1 + 128 = 320128

(b) (i) Depthwise convolution has 12 independent 1x7x7 filters and Pointwise convolution has 4 independent 12x1x1 filters

(ii) FLOPS in original Convolution Layer 1 = $(2 \times 12 \times 7^2) \times 6 \times 49 \times 49 = 16941456$

FLOPS in Depthwise = $2 \times 7^2 \times 12 \times 49 \times 49 = 2823576$

FLOPS in Pointwise = $2 \times 12 \times 6 \times 49 \times 49 = 345744$

FLOPS in Depthwise Separable Convolution = 2823576 + 345744 = 3169320

Ratio is 55:294

(iii) Using a very small mini-batch size with batch normalization can lead to poor estimates of the mean and variance, which are critical for the normalization process. This inaccuracy can result in unstable training and poor model generalization due to the high variance in the small sample statistics. Furthermore, small mini batches fail to leverage the computational efficiency of GPUs, potentially leading to longer training times and suboptimal learning.

Editor's note: There are multiple answers, this is just what I roughly wrote in the exams.

(c) (i) F

(ii) T

(iii) F

(iv) T

(v) F

(vi) F

4 (a)

$$x(1) = \begin{matrix} 1.5 \\ 1 \end{matrix}$$

$$x(2) = \begin{matrix} -3.0 \\ 2.0 \end{matrix}$$

$$U = \begin{matrix} 0.2 & 0.3 \\ 0.8 & 0.9 \end{matrix}$$

$$w = \begin{matrix} 2.0 & 1.0 \end{matrix}$$

$$V = \begin{matrix} 0.2 \\ -0.2 \end{matrix}$$

$$h(1) = \tanh\left(U^T x(1) + W^T y(0) + \begin{matrix} 0.2 \\ 0.2 \end{matrix}\right), where\ y(0) = 1$$

$$h(1) = \tanh\begin{pmatrix} 3.3 \\ 2.55 \end{pmatrix} = \begin{matrix} 0.9973 \\ 0.9879 \end{matrix}$$

$$y(1) = \sigma(V^T h(1) + 0.2) = 0.5503$$

$$h(2) = \tanh\left(U^T x(2) + W^T y(1) + \begin{matrix} 0.2 \\ 0.2 \end{matrix}\right)$$

$$h2 = \tanh\begin{pmatrix} 2.300 \\ 1.650 \end{pmatrix} = \begin{matrix} 0.9801 \\ 0.9289 \end{matrix}$$

$$y(2) = \sigma(V^T h(2) + 0.2) = 0.5524\ (4dp)$$

(b) (i) B (Statement 1 is True, Statement 2 is False)

(ii) C (Statement 1 is False, Statement 2 is True)

(iii) A (Both statements are True)

(c) (i) Sinusoidal positional encoding uses sine and cosine functions of to encode the positions.

$$if\ pos\ is\ even\ PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right),$$

$$else\ if\ pos\ is\ odd PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

This is beneficial as it each position has a unique encoding value and since the sinusoidal function is continuous, models are able to generalize to longer sequences.

(ii) Dimension 0: -0.9589

Dimension 1: 0.2837

Dimension 2: -0.9939

Dimension 3: -0.1107

Dimension 4: -0.9982

Dimension 5: -0.0595

Dimension 6: -0.9750

Dimension 7: -0.2221

Dimension 8: -0.9277

Dimension 9: -0.3733

Dimension 10: -0.8600

Editor's note: This answer is in Radian Mode in my calculator. You can convert to Degree mode, but I think both answers are accepted as long as it's consistent

(d) Mode collapse occurs in GANs when the generator starts producing a limited variety of outputs, often very similar or even identical to each other, despite variations in the input noise. This happens when the generator finds a particular type of output that consistently fools the discriminator and then overfits to these outputs.

The implications are a lack of diversity in the quality of the generated samples as all samples produced tend to be of a similar distribution. Additionally, outputs tend to lack realism and seem unrealistic compared to real data.

Solver: Puah Yi Hao (PUAH0021@e.ntu.edu.sg)