Birla Institute of Technology and Science, Pilani

Work Integrated Learning Programmes Division

M. Tech. in AI & ML

II Semester 2022-2023

End-Semester Test
(EC3 - Regular)

Course Number	AIMLCZG51
Course Name	DEEP NEURAL NETWORK
Nature of Exam	Open Book
Weight-age for grading	40
Duration	2.5 hrs
Date of Exam	

* Pages	4
* Questions	4

---

1. (a) Consider a CNN architecture with an input image of size $256 \times 256 \times 3$. The architecture consists of two convolutional layers with 64 and 128 filters of size 5×5 respectively, followed by a max-pooling layer of size 2×2. Calculate the output dimensions after each layer. [3]

   (b) If the input image size is changed to $128 \times 128 \times 3$, how will it affect the number of trainable parameters? [1]

   (c) Compute the trainable parameters are there in the second convolutional layer for part (a). [1]

   (d) Write Python code to implement for part (a) architecture using TensorFlow Keras. Add one dense layer and an output layer for binary classification. [3]

   (e) I represents the top left corner pixel values of a much larger image (not shown in entirety). F is a filter that convolves over this image with stride 1. O is the result of this convolution, where only 3 resulting pixels at top left are shown. Importantly, there is no zero padding, which implies that each pixel in O is the dot product resulting from the filter being placed at the corresponding position on the image in I. Find the values of x, y and z. [3]

$$I = \begin{pmatrix} 3 & 0 & 3 & -3 & 0 & \dots \\ -3 & 2 & 0 & 3 & -2 & \dots \\ -5 & 0 & 3 & -2 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} \quad F = \begin{pmatrix} x & y & z \\ 1 & 0 & 2 \\ -1 & 1 & 0 \end{pmatrix} \quad O = \begin{pmatrix} 4 & 0 & -3 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

Rubrics and one solution:

(a)
- First Convolutional Layer (64 filters): [1 mark]
  Output dimensions: $(256 - 5 + 1)x(256 - 5 + 1)x64 = 252 \times 252 \times 64$
- Second Convolutional Layer (128 filters): [1 mark]
  Output dimensions: $(252 - 5 + 1)x(252 - 5 + 1)x64 = 248 \times 248 \times 128$

- Max-Pooling Layer (2x2): [1 mark]
  Output dimensions: $248/2x248/2x128 = 124 \times 124 \times 128$

(b)
- Number of Parameters per Filter will remain the same, since the filter size (5x5) and the number of input channels (3) remain the same. [0.5 mark]
- The number of filters in each layer (64 and 128) remains the same. [0.5 mark]

(c) Number of Parameters = (Filter Size * Number of Input Channels+1) * Number of Filters = $(5 \times 5 \times 64 + 1) \times 128 = 204,928$ parameters. [1 mark]

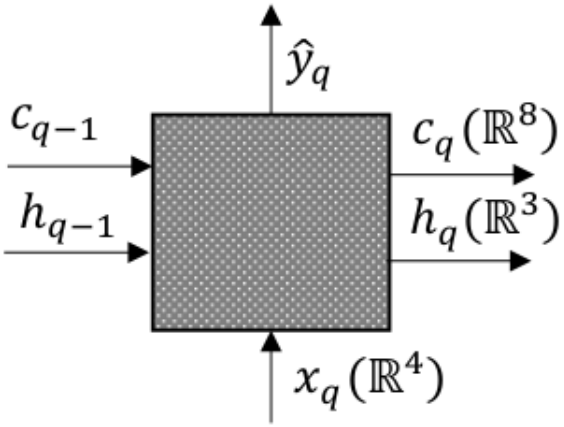
(d) 1 mark for each layer. If sequential model not declared and created give 0 marks for the entire answer.

```
# Define the CNN model
model = models.Sequential()
# First Convolutional Layer
model.add(layers.Conv2D(64, (5, 5), activation='relu',
input_shape=(256, 256, 3)))
# Second Convolutional Layer
model.add(layers.Conv2D(128, (5, 5), activation='relu'))
# Max pooling Layer
model.add(layers.MaxPooling2D((2, 2)))
```

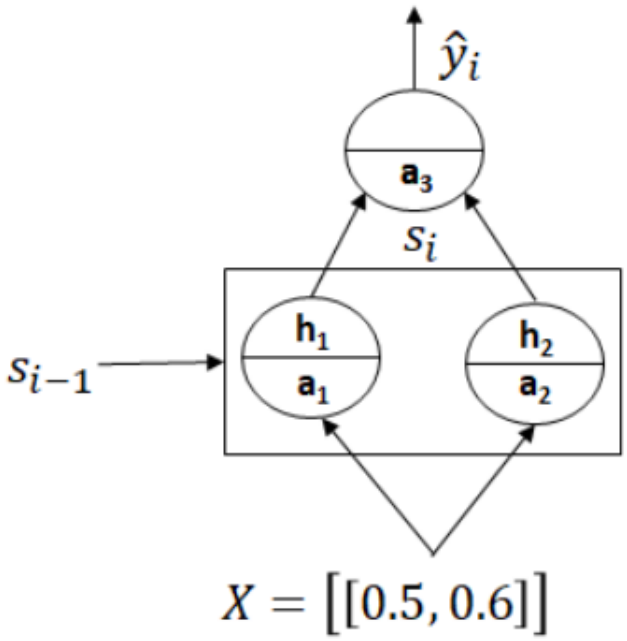
(e) 3 marks for correct equations and answers. No step marking.

$$\begin{aligned} \text{equations:} \quad & 3x + 3y = 12 \\ & 3y - 3z = -11 \\ & 3x - 3y = 6 \\ \text{solving:} \quad & x = 3 \\ & y = 1 \\ & z = 14/3 \end{aligned}$$

2. (a) A data science engineer proposed an LSTM network (shown below) for a deep learning problem. Calculate the parameters to be learnt in it. The dimensions are also shown in the parentheses. Output is one dimensional. [5]



(b) A RNN is shown below with one RNN layer of two tanh neurons and a fully connected output layer with one sigmoid neuron. A record of input with one time-step and two input features is processed with it. Calculate the output of the network assuming all weights and biases are initialized with value 0.2 and previous state as [0.3, 0.4]. The arrows shown in the diagram are representative. [5]



(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)
- Input (x_t): R^4
- Hidden state (h_t): R^3
- Cell state (c_t): R^8
- Output (y_t): R^1
- Input Gate (i_t)

  - Weight matrix for input gate (W_{xi}): $R^{4\times3}$
  - Weight matrix for hidden state (W_{hi}): $R^{3\times3}$
  - Bias for input gate (b_i): R^3
- Forget Gate (f_t)
  - Weight matrix for input gate (W_{xf}): $R^{4\times3}$
  - Weight matrix for hidden state (W_{hf}): $R^{3\times3}$
  - Bias for forget gate (b_f): R^3
- Output Gate (f_t)
  - Weight matrix for input gate (W_{xo}): $R^{4\times3}$
  - Weight matrix for hidden state (W_{ho}): $R^{3\times3}$
  - Bias for Output gate (b_o): R^3
- Cell State (c_t)
  - Weight matrix for input gate(W_{xc}): $R^{4\times8}$
  - Weight matrix for hidden state (W_{hc}): $R^{3\times8}$
  - Bias for cell state (b_c): R^8
- Output (y_t)
  - Weight matrix for output (W_{hy}): $R^{3\times1}$
  - Bias for output (b_y): R^1
- Total= $12+9+3+12+9+3+12+9+3+32+24+8+3+1 = 140$ parameters

(b) Two tanh neurons in the RNN layer. One sigmoid neuron in the output layer. Input features: [0.5, 0.6] Initial state: [0.3, 0.4]. All weights and biases are initialized with a value of 0.2. $W_{hh} = [0.2, 0.2]$; ($W_{xh} = [0.2, 0.2]; b_h = [0.2, 0.2]$

- Calculate the Hidden State (RNN Layer)

$$
\begin{aligned}
h_t &= \tanh(W_{hh} \times h_{t-1} + W_{xh} \times x_t + b_h) \\
h_1 &= \tanh([0.2, 0.2] \times [0.3, 0.4] + [0.2, 0.2] \times [0.5, 0.6] + [0.2, 0.2]) \\
h_1 &= \tanh([0.06, 0.08] + [0.1, 0.12] + [0.2, 0.2]) \\
h_1 &= [\tanh(0.06), \tanh(0.08)] \\
h_1 &\approx [0.059964, 0.079995]
\end{aligned}
$$

- Calculate the Output (Output Layer)

$$
\begin{aligned}
y &= \sigma(W_{hy} \times h_t + b_y) \\
y &= \sigma([0.2, 0.2] \times [0.059964, 0.079995] + 0.2) \\
&= \sigma(0.025993 + 0.2) \approx 0.556275
\end{aligned}
$$

3. (a) Consider a 1D CNN model for time series forecasting with a sequence length of 100. The model has a single convolutional layer with 64 filters, each of size 3, followed by a fully connected layer with 128 neurons. Batch normalization is applied after convolution layer. Calculate the total number of trainable parameters and non-trainable parameters in the model. [3]

   (b) Consider a Deep Neural Network with 3 hidden layers for time series forecasting. The input sequence has a length of 50, and the hidden layers have 64, 128, and 256 neurons, respectively. Calculate the total number of trainable parameters in the model. Assume the output sequence length. [3]

   (c) A data scientist wants to build a Deep Learning model using a single LSTM cell for forecasting a time-series. The training dataset has several records with 15 time steps each. Each time step consists of three-feature normalized numeric data. If total learnable parameters in this model are 180, what is the dimensionality of the short state? [3]
   (Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)
- Convolutional Layer: $64\,filters \times (3\,weights + 1\,bias) = 256$ parameters.
- Batch Normalization
  Trainable parameters: 64 filters × (2 parameters) = 128 trainable parameters
  Non-trainable parameters: 64 filters × (2 parameters) = 128 non-trainable
- Fully connected layer: $128 neurons \times (6400 weights + 1 bias) = 819,328$ trainable parameters.
- Total Trainable Parameters = 256 (Convolution) + 128 (Batch Norm) + 819, 328 (Fully Connected) = 819, 712
  Total Non-Trainable Parameters = 128 (Batch Norm)

(b)
- Input Layer: 0 trainable parameters
- First hidden layer: $50 \times 64 + 64 = 326450 \times 64 + 64 = 3264$ parameters
- Second hidden layer: $(64 \times 128) + 128 = 8320(64 \times 128) + 128 = 8320$ parameters
- Third hidden layer: $(128 \times 256) + 256 = 33024(128 \times 256) + 256 = 33024$ parameters
- Output Layer: Assume it has 1 neuron $(256 \times 1) + 1 = 257(256 \times 1) + 1 = 257$ parameters
- Total = 3264 + 8320 + 33024 + 257 = 44705 parameters

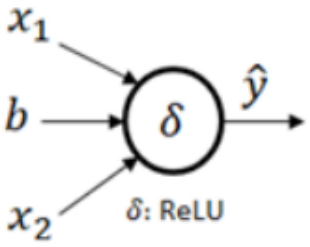
(c) Input dimension (D) is 3.
Weights for Input have the shape (D, H). Weights for Hidden State have the shape (H, H). Biases (b) has the shape (H,1).For each gate, $(D * H) + (H * H) + H$ parameters. For all gates and the cell state, there are $4 * [(D * H) + (H * H) + H]$

$$\begin{aligned} 4 * [(D * H) + (H * H) + H] &= 180 \\ 4 * [(3 * H) + (H * H) + H] &= 180 \\ 3H^2 + 7H - 45 &= 0 \\ H = \frac{-b + \sqrt{b^2 - 4ac}}{2a} &= 2 \end{aligned}$$

Dimensionality of the hidden state (H) is approximately 2.

4. (a) The training data $(x_1, y_1) = (3.5, 0.5)$ for a single Sigmoid neuron and initial values of $w = -2.0, b = -2.0, \eta = 0.10, \beta_1 = 0.90, \beta_2 = 0.99, \epsilon = 1e-8, s_W = 0$ and $s_B = 0$ are provided. Showing all the calculations, find out the values of w, b, s_W, s_B, r_W, r_B after one iteration of Adam. Use BCE as loss function. Apply bias-correction. [8]

   (b) For the training data $[x_1, x_2, y] = [0.65, 0.75, 2.50]$, an engineer selected an initial set of parameters for the single neuron as shown below. Assuming SSE loss function, evaluate his selection of parameters mathematically with proper calculations and reasoning. [3]



w_1	w_2	b	η
0.30	0.50	1.75	2.00

(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)

- Forward Pass and Calculate Loss

$$\hat{y} = \sigma(wx + b) = \sigma(-2.0 \times 3.5 - 2.0) \approx 0.0180$$
$$L = -[y \times \log(\hat{y}) + (1 - y) \times \log(1 - \hat{y})$$
$$L = -[0.5 \times \log(0.0180) + (1 - 0.5) \times \log(1 - 0.0180)] \approx 4.0076$$

- Calculate Gradients

$$\nabla_w L = (\hat{y} - y) \times x = (0.0180 - 0.5) \times 3.5 \approx -0.493$$
$$\nabla_b L = \hat{y} - y = 0.0180 - 0.5 \approx -0.482$$

- Update First Moments

$$s_W = \beta_1 \times s_W + (1 - \beta_1) \times \nabla_w L = 0.90 \times 0 - 0.10 \times (-0.493) \approx 0.04437$$
$$s_B = \beta_1 \times s_B + (1 - \beta_1) \times \nabla_b L = 0.90 \times 0 - 0.10 \times (-0.482) \approx 0.04382$$

- Update Second Moments

$$r_W = \beta_2 \times r_W + (1 - \beta_2) \times (\nabla_w L)^2 = 0.999 \times 0 - 0.001 \times (-0.493)^2 \approx 0.0001216$$
$$r_B = \beta_2 \times r_B + (1 - \beta_2) \times (\nabla_b L)^2 = 0.999 \times 0 - 0.001 \times (-0.482)^2 \approx 0.0001168$$

- Corrected First Moments

$$\hat{s}_W = \frac{s_W}{1 - \beta_1^1} \approx \frac{0.04437}{1 - 0.90^1} \approx 0.4437$$
$$\hat{s}_B = \frac{s_B}{1 - \beta_1^1} \approx \frac{0.04382}{1 - 0.90^1} \approx 0.4382$$

- Corrected Second Moments

$$\hat{r}_W = \frac{r_W}{1 - \beta_2^1} \approx \frac{0.0001216}{1 - 0.999^1} \approx 0.0608$$
$$\hat{r}_B = \frac{r_B}{1 - \beta_2^1} \approx \frac{0.0001168}{1 - 0.999^1} \approx 0.0584$$

- Update Weights and Biases

$$w = w - \frac{\eta}{\sqrt{\hat{r}_W} + \epsilon} \times \hat{s}_W = -2.0 - \frac{0.10}{\sqrt{0.0608} + 1e-8} \times 0.4437 \approx -2.081$$
$$b = b - \frac{\eta}{\sqrt{\hat{r}_B} + \epsilon} \times \hat{s}_B = -2.0 - \frac{0.10}{\sqrt{0.0584} + 1e-8} \times 0.4382 \approx -2.079$$

(b) Training data: $[x_1, x_2, y] = [0.65, 0.75, 2.50]$ Initial parameters: $w_1 = 0.3$, $w_2 = 0.5$, $b = 1.75$ Learning rate: $\eta = 2$

- Calculate the Predicted Output

$$\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$$
$$\hat{y} = \sigma(0.3 \times 0.65 + 0.5 \times 0.75 + 1.75) = \sigma(2.32) \approx 0.9106$$

- Calculate the Sum of Squared Errors (SSE) loss

$$\text{SSE} = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(0.9106 - 2.50)^2 = \frac{1}{2}(-1.5894)^2 = \frac{1}{2} \times 2.5245 = 1.2623$$

To improve the model's performance, the engineer may need to adjust the weights and bias using a learning algorithm (e.g., gradient descent) to minimize the SSE loss.