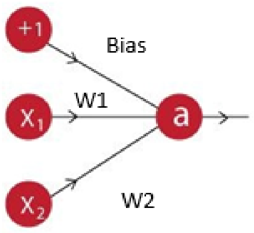
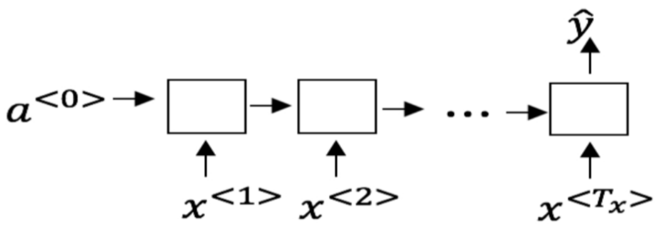
MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

B. Tech Vth Semester Mid-Sem Examination September 2024

Date: 25/09/2024 DSE3121 Deep Learning Techniques: Scheme Max. Marks: 30

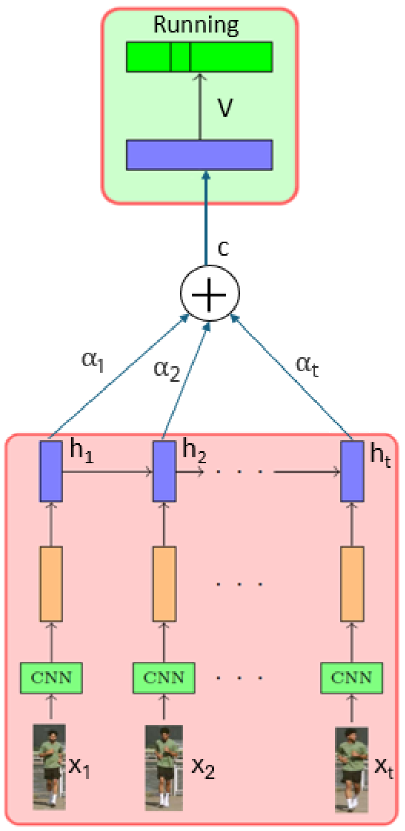
		M	CO	BL
	Section A: MCQ			
Q1	Consider a Sigmoid Neuron represented by "a", with the following architecture.  What are the weights and biases for implementing an AND gate with this neuron, where x1 and x2 are inputs to the AND gate. **1. Bias = -1.5, w1 = 1, w2 = 1** 2. Bias = 1.5, w1 = 2, w2 = 2 3. Bias = 1, w1 = 1.5, w2 = 1.5 4. Bias = 5, w1= 10, w2=10	**0.5**	**CO1**	**3**
Q2	Model capacity refers to the ability of a neural network to approximate complex functions. Which of the following is true about model capacity? 1. As the dropout rate increases, model capacity increases **2. As the number of hidden layers increases, model capacity increases** 3. As learning rate increases, model capacity increases 4. As the learning rate drops, model capacity increases.	**0.5**	**CO1**	**2**
Q3	What is the purpose of the backpropagation through time (BPTT) algorithm in a Recurrent Neural Network? 1. To adjust the learning rate during training **2. To compute the gradients and update the network's parameters** 3. To prevent overfitting by regularizing the model 4. To input test instances in random order	**0.5**	**CO3**	**2**
Q4	In an LSTM cell, the _________ gate controls and decides the part of the long term state to be removed **1. Forget** 2. Input 3. Output 4. Update	**0.5**	**CO3**	**2**
Q5	To which of the following tasks can we apply the following Recurrent Neural Network?  $\hat{y}$ $a^{<0>}$ $x^{<1>}$ $x^{<2>}$ $x^{<T_x>}$	**0.5**	**CO3**	**3**

	1. Machine Translation 2. Chatbot 3. Image Captioning **4. Sentiment Analysis**			
Q6	A convolutional neural network layer takes an input feature map of size 28 × 28 × 10 (width × height × depth). The layer uses 16 filters, each of size 3 × 3. How many trainable parameters are there in this convolutional layer (including the bias)? 1. 1600 2. 160 3. 14560 **4. 1456**	**0.5**	**CO2**	**3**
Q7	In a CNN, what is the analytical effect of using a stride greater than one in convolutional layers? 1. It increases the feature map size, capturing more details. **2. It acts similar to pooling by reducing the spatial dimensions.** 3. It has no effect on the receptive field of the network. 4. It enhances the network's ability to generalize.	**0.5**	**CO2**	**2**
Q8	An RGB-Image of width x height equal to 25 x 30, is convolved using two kernels of width x height = 3 x 5. What is the width x height x depth of the resultant feature map if padding=1 and, stride = 2. 1. 12 x 13 x 3 2. 12 x 13 x 2 3. 13 x 14 x 3 **4. 13 x 14 x 2**	**0.5**	**CO2**	**3**
Q9	If you replace the activation function in a Deep CNN's convolutional layers from ReLU to sigmoid, what is the most significant analytical impact on the network's training dynamics? 1. Improved convergence speed due to non-linearity. **2. Increased likelihood of vanishing gradients.** 3. Enhanced feature extraction capabilities. 4. Reduced computational requirements	**0.5**	**CO2**	**2**
Q10	In an Encoder-Decoder architecture for machine translation, how does attention mechanism improve the performance compared to a basic Encoder-Decoder model? 1. By reducing the number of parameters in the model. 2. By focusing on all input tokens equally during decoding. **3. By allowing the decoder to focus on relevant parts of the input sequence selectively.** 4. By eliminating the need for the encoder component entirely.	**0.5**	**CO3**	**2**

	Section B: Descriptive			
Q11	You are tasked with designing a deep learning model to classify videos into different action categories (E.g., "walking," "running," "jumping," etc.). Each video consists of a sequence of frames captured over time. With the help of a **block diagram and necessary equations**, design an **encoder-decoder framework with an attention mechanism** for the given task. Explain how the model can classify input videos by focusing on the relevant frames.	4	CO3	4

Answer:

o/p: Running (one of the defined actions)



i/p: Frames of a video

Data:

x^j_t : tth frame of jth video,

y^j: Action

where j=1: Number of Samples, t=1: no. of frames in a video

Encoder:

$$h_t = \text{RNN}(\text{CNN}(x_t), h_{t-1})$$

Decoder:

$$e_t = v_a^\top \tanh(W_a h_t + U_a s)$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{t'=1}^{T} \exp(e_{t'})}$$

$$c = \sum_{t=1}^{T} \alpha_t h_t$$

Where T is the number of frames in the video

$$\hat{y} = \text{Softmax}(Vc + b_c)$$

Parameters: V, b_c, W_{conv}, U_{enc}, W_{enc}, W_a, U_a, V_a

Loss: $\mathscr{L}(\theta) = -\log P(y = \ell | Video)$

Each frame of the input video is passed through a CNN to extract spatial features. These features are fed into an RNN to model the temporal dependencies between frames, producing a sequence of hidden states **h_t**

The attention mechanism computes an alignment score e_t between each encoder hidden state h_t and the current decoder state s. This score reflects the "importance" of each encoder hidden state for predicting the current output at time t. **Once the alignment scores e_t are computed for each time step t, they are normalized using the softmax function to produce attention weights α_t for each frame**. This ensures that the attention weights sum to 1. The model assigns higher attention weights to frames that are more relevant.

The attention mechanism aggregates the hidden states into a context vector c. This vector captures the most important information from the video sequence, allowing the model to focus on key frames while ignoring irrelevant ones.

The context vector is passed to a fully connected layer with a softmax activation to output the predicted action category.

Scheme:

- **Diagram: 1 mark**
- **Computations/Equations: 1.5 mark**
- **Explanation of the model and terms in the equation: 1.5 mark**

Q12	With a suitable diagram or example, Compare and Contrast the following: a) The Regression Loss functions of MSE and Huber Loss b) ReLU vs Softplus activation function c) Working of Simple RNN vs Bidirectional RNN	**3**	**CO1**	**2**

Answer:

a) Mean square error is the square of the difference between actual and predicted value. Huber loss is quadratic when error is smaller than threshold but linear when the error is larger than threshold. The linear function makes it less sensitive to outliers than the mean square error.

b) Rectified Linear Unit returns the input value ‘x’ when 'x' is positive, and returns 0 if 'x’ is negative. ReLU is continuous but not differentiable at x=0. Softplus is a smooth variant of ReLU, softplus is near 0, where the softplus is enticingly smooth and differentiable near 0.

c) At each time step a Simple RNN looks at past and present inputs to generate its output. In tasks like NLP , there is a need to look ahear at the next words before encoding a given word. So Bidirectional RNN has two recurrent layers on the same inputs, one reading words from left to right and the other reading words from right to left.

Scheme:

- **1 mark for each subdivision – 0.5 for explanation, 0.5 for suitable diagram/example**

Q13	Consider an RNN for stock price forecasting based on historical data. **The input time series is multivariate and consists of two values at each time step: stock price and trading volume.** The RNN consists of **one hidden layer with two neurons and a dense output layer with one neuron**, which predicts the next stock price. At **time step 1**, the input vector is **x_1 = [Stock Price_1, Trading Volume_1]**, and at **time step 2**, the input vector is **x_2 = [Stock Price_2, Trading Volume_2]**. The RNN uses the following parameters: • **U, W, and V** are weight matrices for the input, recurrent connections, and output layers, respectively. • **b and c** are biases for the hidden and output layers, respectively. Given the following values for the input and the parameters: $x_1 = \begin{bmatrix} 0.4 \\ 0.3 \end{bmatrix}$ $U = \begin{bmatrix} 0.6 & 0.1 \\ -0.2 & 0.5 \end{bmatrix}$ $x_2 = \begin{bmatrix} 0.5 \\ 0.2 \end{bmatrix}$ $W = \begin{bmatrix} 0.3 & -0.4 \\ 0.2 & 0.6 \end{bmatrix}$ $b = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$ $V = \begin{bmatrix} 0.7 & 0.3 \end{bmatrix}$ $c = 0.5$ a) Calculate the hidden state s_1 at time step 1 and the predicted stock price y_1 for the next time step.	**3**	**CO3**	**4**

b) Calculate the hidden state s_2 at time step 2 and the the predicted stock price y_2 for the next time step.

Note: Use ReLU activation function at the hidden layers and linear activation function at output layer. Additionally, the initial hidden state is a vector of zeros.

Answer:

a) **At time step t=1, hidden state s_1:**

$$s_1 = \mathrm{ReLU}(Ux_1 + Ws_0 + b)$$

Since $s_0 = \begin{bmatrix}0\\0\end{bmatrix}$, $Ws_0 = \begin{bmatrix}0\\0\end{bmatrix}$, so we only compute $Ux_1 + b$:

$$Ux_1 = \begin{bmatrix}0.6 & 0.1\\-0.2 & 0.5\end{bmatrix}\begin{bmatrix}0.4\\0.3\end{bmatrix} = \begin{bmatrix}0.27\\0.11\end{bmatrix}$$

$$Ux_1 + b = \begin{bmatrix}0.27\\0.11\end{bmatrix} + \begin{bmatrix}0.1\\0.2\end{bmatrix} = \begin{bmatrix}0.37\\0.31\end{bmatrix}$$

Applying ReLU:

$$s_1 = \mathrm{ReLU}\left(\begin{bmatrix}0.37\\0.31\end{bmatrix}\right) = \begin{bmatrix}0.37\\0.31\end{bmatrix}$$

Predicted stock price y_1 for the next time step:

$$y_1 = Vs_1 + c$$

$$Vs_1 = \begin{bmatrix}0.7 & 0.3\end{bmatrix}\begin{bmatrix}0.37\\0.31\end{bmatrix} = (0.7 \times 0.37) + (0.3 \times 0.31) = 0.259 + 0.093 = 0.352$$

$$y_1 = 0.352 + 0.5 = 0.852$$

b) **At time step t=2, hidden state s_2:**

$$s_2 = \mathrm{ReLU}(Ux_2 + Ws_1 + b)$$

$$Ux_2 = \begin{bmatrix}0.6 & 0.1\\-0.2 & 0.5\end{bmatrix}\begin{bmatrix}0.5\\0.2\end{bmatrix} = \begin{bmatrix}0.32\\0.06\end{bmatrix}$$

$$Ws_1 = \begin{bmatrix}0.3 & -0.4\\0.2 & 0.6\end{bmatrix}\begin{bmatrix}0.37\\0.31\end{bmatrix} = \begin{bmatrix}0.009\\0.297\end{bmatrix}$$

$$Ux_2 + Ws_1 + b = \begin{bmatrix}0.32\\0.06\end{bmatrix} + \begin{bmatrix}0.009\\0.297\end{bmatrix} + \begin{bmatrix}0.1\\0.2\end{bmatrix} = \begin{bmatrix}0.429\\0.557\end{bmatrix}$$

Applying ReLU:

$$s_2 = \mathrm{ReLU}\left(\begin{bmatrix}0.429\\0.557\end{bmatrix}\right) = \begin{bmatrix}0.429\\0.557\end{bmatrix}$$

Predicted stock price y_1 for the next time step:

$$y_3 = Vs_3 + c$$

$$Vs_3 = \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} \begin{bmatrix} 0.2609 \\ 0.8059 \end{bmatrix} = (0.7 \times 0.2609) + (0.3 \times 0.8059) = 0.18263 + 0.24177 = 0.4244$$

$$y_3 = 0.4244 + 0.5 = 0.9244$$

Scheme:

a) Hidden state computation + Predicted stock price computation: 1 + 0.5 = 1.5 marks

b) Hidden state computation + Predicted stock price computation: 1 + 0.5 = 1.5 marks

Q14	Explain the problem of Vanishing and Exploding Gradients in RNN with examples. How does gradient clipping help in solving unstable gradients?	**3**	**CO3**	**2**

Answer:

As the algorithm back propagates error from the output layer to the input layer, the gradients get smaller and approach zero which eventually leaves the weights of the unchanged. As a result, the gradient descent never converges to the global optima. This is known as the vanishing gradients problem.

In some cases, the gradients keep on getting larger as the backpropagation algorithm progresses. This causes very large weight updates and the gradient descent diverges. This is known as the exploding gradients problems.
One solution to deal with exploding gradient is to specify a cap on the upper limit of the gradient and the method is called as Gradient Clipping

Scheme:

- **1 mark for explanation of vanishing gradient with appropriate example**
- **1 mark for explanation of exploding gradient with example,**
- **1 mark for gradient clipping**

Q15	Consider a Convolutional Neural Network (CNN) applied to gridded weather data. The input consists of **4-channels** (corresponding to the value of 4 atmospheric variables) of size **8 x 12** corresponding to spatial grids in the region of interest. A convolution operation is performed on the input layer with **six filters** each of size **3 x 5**. The **stride** length is equal to **1**, and **no padding** is applied. Based on the given information, answer the following: a) Compute the number of parameters and the total number of connections (including the bias) between the input layer and the convolution layer. b) Explain the two main advantages of using a convolution layer instead of a fully connected layer, based on the results obtained in (a)).	**3**	**CO2**	**3**

Answer:

a) **Number of parameters: ((3*5*4)+1)*6 = 366**

Number of Connections:

To compute the total number of connections, we need to figure out the feature map size after the convolution.

- The input size is 8×12, the filter size is 3×5, the stride is 1, and there is no padding. Therefore, the feature map size is 6x8x6.

- Each neuron of one plane of the feature map (in the convolution layer) is connected to (3*5*4) +1 = 61 neurons in the input layer.
- Total connections for 1 plane of feature map in the convolution layer = 6*8*61 = 2928
- Total connections for 6 planes of feature maps in the convolution layer = 2928*6 =17568
- So, **Total no. of connections = ((3*5*4) +1) * (6*8) *6= 17568**

b) The two main advantages of using a convolutional layer instead of a fully connected layer are:

- **Sparse Connectivity**: Total no. of connections/parameters if it was a fully connected layer= ((8*12*4) + 1)*6*8*6 = 110880.
  However, the total no. of connections for a convolutional layer is = 17568, which means that there is a 84.15% decrease in the total no. of connections as compared to the fully connected layer.
- **Shared Weights:** In a typical ANN, the number of parameters between two layers is equal to the total number of connections. The parameters themselves are the weights associated with these connections. However, in the case of CNN, as the same convolutional filter is applied across the entire image, we need not update all the 17,568 connections of the network separately. As the weights are shared across the pixels/neurons of the image, we need to update only 366 parameters out of 17,568 connections. Compared to a fully connected layer, instead of updating 110880 parameters, we are now updating only 366 parameters.

Scheme:

- **No. of Parameters + No. of connections: 0.5 + 0.5 = 1 mark**
- **Two advantages with comparison against Fully Connected NN: 1 + 1 = 2 marks**

Q16	Given an Input Image I and a filter K as defined below, perform the following operations:	3	CO2	3

a) Apply the **2x2 convolution filter K** on the image I with a **stride of 1 in the horizontal direction** and **stride of 2 in the vertical direction,** and compute the resulting feature map (consider the top left pixel in a window as the reference).

b) Perform **Max Pooling** on the resulting feature map using a 2x2 window with a **stride of 2**, and compute the final output.

$$I = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Answer:

a) Resulting Feature map: $\begin{bmatrix} 7 & 9 & 11 \\ 23 & 25 & 27 \end{bmatrix}$

b) Feature map after Max. Pooling: $\begin{bmatrix} 25 \end{bmatrix}$

Scheme:

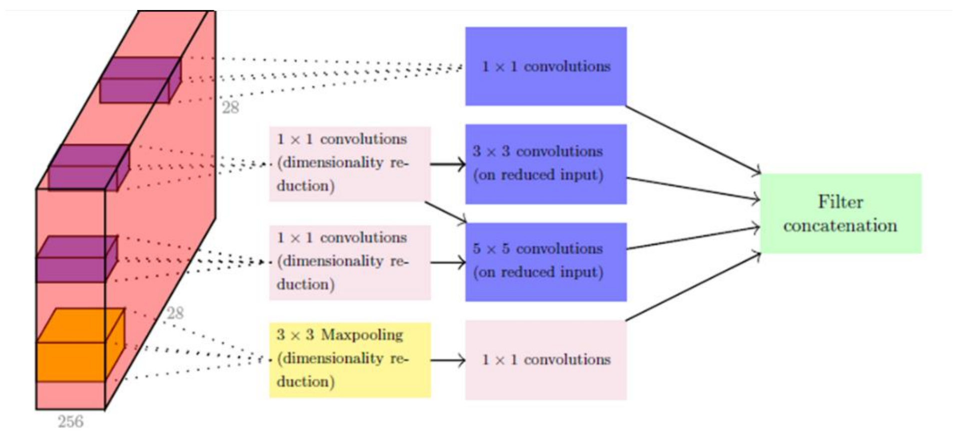
a) 1.5 mark

b) 1.5 mark

Q17	Explain how the use of 1x1 convolutions in the Inception module of Google's InceptionNet allows the model to examine input at multiple scales simultaneously and increase the depth of convolution layers, all while minimizing computational complexity and controlling the number of parameters. Support your explanation with an example.	**2**	**CO2**	**3**

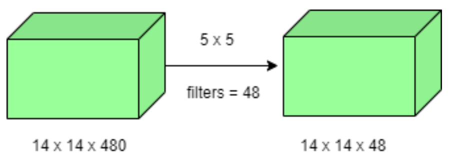
Answer:

Consider the following GoogleNet Inception Module architecture:


28
1 × 1 convolutions
1 × 1 convolutions (dimensionality reduction)
3 × 3 convolutions (on reduced input)
1 × 1 convolutions (dimensionality reduction)
5 × 5 convolutions (on reduced input)
28
3 × 3 Maxpooling (dimensionality reduction)
1 × 1 convolutions
Filter concatenation
256

The inception module of GoogleNet uses 1x1 convolution which aggregates the pixels along its depth. In the block diagram of the inception architecture, 1x1 convolution produces a feature map of size 28x28x1. This shrinks the depth of the input volume from 256 to 1. Now, one could apply K different 1x1 filters to produce a feature map of size 28x28xK (where, K<<256), over which filters of varying receptive fields (3x3 or 5x5) could be applied. This significantly reduces the number of parameters and computations when compared to applying these filters to the original input volume.

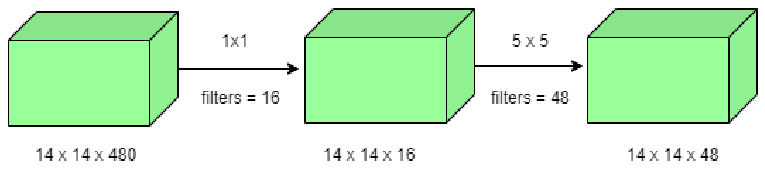
To illustrate this, consider the following example:



If we want to perform 5×5 convolution having 48 filters without using 1×1 convolution as intermediate:

Total no. of parameters: ((5*5*480) + 1)*48 = 5,76,048

However, if 1x1 convolution is used:



Total no. of parameters: ((1*1*480)+1)*16 + ((5*5*16)+1)*48 = 38,944 which is much lesser than 5,76,048. Similarly, the total no. of computations without using 1x1 convolution is : (14*14*480)*(5*5)*48 = 1,12,896,000 Similarly, the total no. of computations using 1x1 convolution is : (14*14*480)*(1*1)*16 + (14*14*16)*(5*5)*48 = 5,268,480.

Therefore, we can apply filters of multiple sizes and layers allowing us to examine the input at multiple scales and at examine features at an abstract level without worrying about the exponential increase in the no. of parameters/computations.

Scheme:

- **Explanation of with example: 1+1 = 2 marks**

Q18	Explain the differences in the features of the LSTM vs the GRU cell.	**2**	**CO3**	**2**

Answer:

A) In terms of Architecture, LSTM has a complex architecture an input gate, update gate, output gate, and forget gate, while GRU has an update gate and a reset gate.

B) In terms of State, LSTM has separate cell state for long term memory and hidden states for short term memory. GRU combines them into a single entity using mathematical operations.

C) LSTM has more parameters to train than GRU.

D) LSTM has a higher computational cost than GRU.

Scheme:

- **0.5 marks for each point**

Q19	Answer the following questions: a) In a fully connected neural network, explain with an example how overfitting is detected. b) Can a neuron without a bias term, using any activation function, learn to produce a non-zero output when all inputs are zero or when the inputs are linearly dependent? Justify your reasoning	**2**	**CO1**	**2**

Answer:

a) Overfitting- Model specialized on training data, it is not able to generalize to new data. A plot of learning curves shows overfitting if:
The plot of training loss continues to decrease with experience. The plot of validation loss decreases to a point and begins increasing again. The inflection point in validation loss may be the point at which training could be halted as experience after that point shows the dynamics of overfitting.

b) **Case-1: When all inputs are zero:**
The output of a neuron is typically given by the formula: $y = f(\sum w_i x_i + b)$

If there is no bias term (i.e., b=0) and all inputs x_i=0, the weighted sum becomes zero: $\sum w_i x_i = 0$

Thus, the neuron will receive zero as input to the activation function: $y = f(0)$

Most common activation functions like the sigmoid, ReLU, or tanh return specific values when their input is zero. Without the bias term, the neuron cannot "learn" to produce a non-zero output for zero inputs because there is no flexibility in shifting the output. The activation function is dependent purely on the weighted sum of the inputs, and with all inputs being zero, the weighted sum will always be zero.

Case-2: When inputs are linearly dependant:

If the inputs are linearly dependent (i.e., some inputs are scalar multiples of others), the neuron without a bias still faces a challenge. The output will be constrained by the linear relationships between the inputs. In this case, the neuron can only map linearly dependent inputs to linearly dependent outputs, which limits its ability to model more complex relationships.

Therefore, without a bias term, the neuron lacks the ability to introduce a constant shift in the output, which is necessary for breaking the dependence on the inputs. This limits its expressiveness, especially when the inputs are zero or linearly dependent.

Scheme:

a) **Suitable diagram of learning curve – 0.5 marks, explanation – 0.5 marks**

b) **Case-1: 0.5 marks and Case-2: 0.5 marks**