Question-1

The time gradient of the Heaviside function is given by,

$$\partial H(\xi) = \begin{cases} 0 & \text{if } \xi < 0 \\ 1 & \text{if } \xi > 0 \\ \text{undefined} & \text{if } \xi = 0 \end{cases}$$

Question - 2

A → Let's denote the output probabilities from the network as follows,

1. for the first approach (1 of (c+1) class label):

↳ Let P_i be the probability that the image region contains an object of class i (where $i = 1, 2, \ldots c$).

↳ Let $P_{background}$ be the probability that the image region does not contain an object of any of the defined class.

2. for the 2nd approach (binary variable for presence / absence and 1-of-c vector for class label):

↳ Let $P_{presence}$ be the probability that an object is present in the image region.

↳ Let Q_i be the probability that if an object is present, it belongs to class i (where $i = 1, 2, \ldots c$)

The relationship between these 2 sets of probability can be expressed as follows:

↳ $P_{presence} = 1 - P_{background}$

(Since either an object is present or the background

↳ for $i = 1, 2, \ldots c$:

$P_i = P_{presence} \times Q_i$ (Probability that an object is present and belongs to class i)

B

The convolution operation in a CNN is a discrete approximation of the continuous convolution operation. The discrete convolution operation can be seen as a sum of products, where each element in the output feature map is calculated by taking a weighted sum of the elements in the input feature map and the corresponding weights in the filter kernel.

Let's breakdown the components.

↳ $I(j, k)$ represents the pixel intensities of the input image at position (j, k).

$\hookrightarrow$ $K(l,m)$ represents the pixel values for the filter kernel at position (l,m)

$\hookrightarrow$ $C(j,k)$ represents the activation values for the feature map at position (j,k).

The activation value at position (j,k) in the feature map C is calculated by taking the sum of the product of pixels intensities in the input feature map I and the corresponding weights in the filter kernel K, with appropriate shifts:

$$C(j,k) = \sum_{l} \sum_{m} I(j+l, k+m)\, K(l,m)$$

This operation effectively computes the weighted sum of pixel intensities in the input feature map, with the weights determined by the filter kernel. This process is akin to the discrete approximation of the continuous convolution operation, where the filter kernel acts as the convolutional kernel $K(x-y)$ in the continuous case.

Question - 3

Start with the original formulation,

$$Y(x) = \text{Concat}[H_1, \ldots, H_H] \cdot W(o)$$

where,

- $Y(x)$ is the output of the multi-head attention mechanism.
- Concat $[H_1, \ldots, H_H]$ represents the concentration of the attention heads.
- $W(o)$ is the weight matrix for the output transformation.

Now re-write the expression in steps

Step-1

Define the attention heads individually

$$H_h = \text{Attention}(Q_h, K_h, V_h)$$

where Q_h, K_h, V_h are the Query, key and Value matrices for the hth attention-head.

Step-2

Define the transformation matrix for each attention head,

$$W(h) = W(r)_h, W(o)_h$$

where,

C_s $W(v)h$ is the weight matrix for the vertical segments of the concatenated attention matrix for the hth head, typically of dimension $D_v \times D$

C_s $W(o)h$ is the weight matrix for the horizontal segments of the concatenated attention matrix for the hth head.

Step-3

Rewrite the original expression using the defined attention heads and transformation matrices:

$$Y(x) = \text{Concat}[H_1, \ldots, H_H] \cdot W(o)$$

$$= \text{Concat}[\text{Attention}(Q_1, K_1, V_1), \ldots, \text{Attention}(Q_H, K_H, V_H)] \cdot W(o)$$

$$= \sum_{h=1}^{H} H_h \cdot W(o)$$

$$= \sum_{h=1}^{H} \text{Attention}(Q_h, K_h, V_h) \cdot W(o)$$

$$= \sum_{h=1}^{H} \text{Attention}(Q_h, K_h, V_h) \cdot (W(v)_h \cdot W(o)_h)$$

$$= \sum_{h=}^{H} \text{Attention}(Q_h, K_h, V_h) \cdot W(v)_h \cdot W(o)_h$$

Now the expression is rewritten in the form:

$$Y = \sum_{h=1}^{H} H_h \cdot X \cdot W(h)$$

where,

$$H_h = \text{Attention}(Q_h, K_h, V_h)$$

$$W(h) = W(v)_h \cdot W(o)_h$$

This shows that the original expression of multi-head attention can indeed be re-written in the specified form.

Question - 4

forward pass:—

(a)

We can compute the forward pass as,

$$h_1 = w \cdot (u_1 + 0) = w \cdot u_1$$

$$h_2 = w \cdot (u_2 + w \cdot u_1) = w \cdot (u_2 + w \cdot u_1)$$

$$h_3 = w \cdot (u_3 + w \cdot (u_2 + w \cdot u_1))$$

$$= w \cdot (u_3 + w \cdot (u_2 + w \cdot u_1))$$

Question-5

Given,

Image matrix,

$$\begin{bmatrix} 4 & 5 & 2 & 2 & 1 \\ 3 & 3 & 2 & 2 & 4 \\ 1 & 3 & 4 & 1 & 1 \\ 5 & 1 & 4 & 1 & 2 \\ 5 & 1 & 3 & 1 & 4 \end{bmatrix}$$

(5x5)

Convolutional Kernal,

$$\begin{bmatrix} 4 & 3 & 3 \\ 5 & 5 & 5 \\ 2 & 4 & 3 \end{bmatrix}$$

(3x3)

(i) values generated,

After forward propagation, we will be having a feature matrix having rows,

$5 - 3 + 1 = 3$

&

having columns,

$5 - 3 + 1 = 3$

So, 9 values will be generated after forward propagation.

(ii) let's denote the feature matrix as, F

So, $F[1,1] = 16 + 15 + 6 + 15 + 15 + 10 + 2 + 12 + 12$

$= 103$

$F[1,2] = 20+6+6+15+10+10+6+16+3$
$= 92$

$F[1,3] = 8+6+3+10+10+20+8+4+3$
$= 72$

$F[2,1] = 12+9+6+5+15+20+10+4+12$
$= 93$

$F[2,2] = 12+6+6+15+20+5+2+16+3$
$= 95$

$F[2,3] = 8+6+12+20+5+5+8+4+6$
$= 84$

$F[3,1] = 4+3+12+25+5+20+10+4+9$
$= 92$

$F[3,2] = 12+12+3+15+20+5+2+12+3$
$= 84$

$F[3,3] = 16+3+3+20+5+10+6+4+12$
$= 79$

So the final 'F' matrix is,

$$\begin{bmatrix} 103 & 92 & 72 \\ 93 & 95 & 84 \\ 92 & 84 & 79 \end{bmatrix}$$

B

Let's denote the weight matrix as W.

For each output input, we have a kernel of width 3. Since the input dimensionality is 5 and the stride is 1, we have 3 positions where the filter can slide over the input.

Each filter has 3 weights are shared among different positions in the input. Therefore we will replicate the weights accordingly.

Weight Matrix (W) :-

$$W = \begin{bmatrix} w_1 & w_2 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & 0 \\ 0 & 0 & 0 & w_1 & w_2 \end{bmatrix}$$

Here,

w_1 & w_2 represent the shared parameters for the connections.

- The weight matrix captures the sliding window operation of convolutional layer.
- Zero are essential for the missing connections as convolutional layer have sparse connection.

This weight matrix allows us to perform the convolutional operation by simply performing matrix multiplication between the input vectors and the weight matrix.

Therefore, the 1D Convolutional layer can be expressed as a special case of a fully connected layer using the constructed weight matrix W.

Part-C

after padding the input vector is,

$$I = \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ 0 \end{bmatrix}$$ and the filter is given by

$$f = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

let's evaluate the convolution assuming stride = 2,

$$I * f$$

$$= \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ 0 \end{bmatrix} * \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$= \begin{bmatrix} w_1 \cdot 0 + w_2 x_1 + w_3 x_2 \\ w_1 \cdot x_2 + w_2 \cdot x_3 + w_3 x_4 \end{bmatrix}$$

This is the output vector.

now lets evalute a matrix A, which should give the output matrix after multiplication with I.

$$A = \begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & 0 \end{bmatrix}$$

So, we can see that

$A \cdot I = A \cdot$ (Input vector)

gives the output vector.

Question-6

(a) Step-1 (Input representation)

↳ Input sentence: Sequence of 10 words.

↳ Each word is represented by a word vector of size [1x300].

↳ So, the input to the Conv Net would be a tensor of shape [10x300].

Step-2 (Pooling)

↳ apply convolutional filters over the input tensor.

↳ let's say we use 5 filters with a kernel of size 3. This will result in 5 feature maps.

↳ Apply ReLU activation.

↳ Perform max pooling over each feature map to reduce its size of 1, resulting in a tensor of shape [5x1].

Step-3 (Fully connected layer with softmax).

- Flatten the pooled tensor into a vector of length 5.
- pass this vector through a fully connected layer.
- Apply softmax activation to obtain the probability values for the possible 3 classes.

(b) Step-1

Input → 10 X 300

Step-2

Convolutional layer,

- Input → 10 X 300
- no of filters : 5
- kernel size : 3 X 300
- output after convolution : 8 X 1 X 5 (assuming padding is used)

→ Activation fn → ReLU.

→ Max pooling layer,

↳ pool size: [8x1]

↳ output size after pooling [1x1x5]

Step-3

→ flatten the pooled tensor into a vector of length 5.

→ fully connected layer:

Input size: 5

output size: 3 (no. of classes)

→ Activation → softmax.

→ Output [3x1].

(c) Small filter size:—

↳ focuses on capturing localized features.

↳ effective at capturing fine grained details.

↳ may not capture broader contextual information.

↳ typically results in fewer parameters.

Large filter size:

- have a wider receptive field, allowing them to capture global or broader features.
- may overlook fine-grained details.
- generally lead to models with more parameters, potentially increasing computational complexity.

Selecting filter size:-

- Involves the trade-off between capturing localized details (with smaller filter sizes) and capturing broader contextual information (with large filter sizes), based on the specific requirements of the classification task.