USN | | | | | | | | | | |

Internal Assessment Test II –November 2025

Sub:	DEEP LEARNING					Sub Code:	BCA701	Branch:	CSE AIML	
Date:		Duration:	90 min	Max Marks:	50	Sem/Sec:	VII C		OBE	
Answer any FIVE FULL Questions								**MARKS**	**CO**	**RBT**
1 a	You are designing a Convolutional Neural Network (CNN) to classify handwritten digits from noisy images. Illustrate how you would choose the kernel size, number of filters, pooling strategy, and activation function to handle noise effectively. Justify how each chosen CNN component contributes to improving accuracy in this scenario.							10	CO4	L3
2 a	Discuss the variants of the basic convolution function.							10	CO4	L2
3 a	Explain the various datatypes used with CNN.							5	CO4	L2
3 b	Differentiate between Stride and Pooling in CNN.							5	CO4	L2

USN | | | | | | | | | | |

Internal Assessment Test II – November 2025

Sub:	DEEP LEARNING					Sub Code:	BCA701	Branch:	CSE AIML	
Date:		Duration:	90 min	Max Marks:	50	Sem/Sec:	VII-C		Duration:	
Answer any FIVE FULL Questions								**MARKS**	**CO**	**RBT**
1 a	You are designing a Convolutional Neural Network (CNN) to classify handwritten digits from noisy images. Illustrate how you would choose the kernel size, number of filters, pooling strategy, and activation function to handle noise effectively. Justify how each chosen CNN component contributes to improving accuracy in this scenario.							10	CO4	L3
2 a	Discuss the variants of the basic convolution function.							10	CO4	L2
3 a	Explain the various datatypes used with CNN.							5	CO4	L2
3 b	Differentiate between Stride and Pooling in CNN.							5	CO4	L2

4	Explain Bidirectional RNN and Deep Recurrent Networks.	10	CO5	L2
5	Given a sequence where long-term dependencies are crucial, explain how each LSTM gate (input gate, forget gate, and output gate) would operate on the sequence during processing. Illustrate how gate activations change the cell state and hidden state at each step, and analyze how modifying the gate values would affect the model's ability to retain or eliminate information.	10	CO5	L2
6	Explain how Recurrent Neural Network (RNN) processes the data sequence using computational graphs.	10	CO5	L2

Faculty Signature CCI Signature HOD Signature

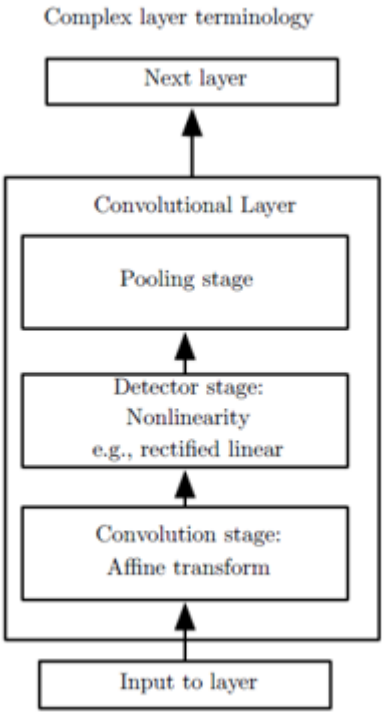
4	Explain Bidirectional RNN and Deep Recurrent Networks.	10	CO5	L2
5	Given a sequence where long-term dependencies are crucial, explain how each LSTM gate (input gate, forget gate, and output gate) would operate on the sequence during processing. Illustrate how gate activations change the cell state and hidden state at each step, and analyze how modifying the gate values would affect the model's ability to retain or eliminate information.	10	CO5	L2
6	Explain how Recurrent Neural Network (RNN) processes the data sequence using computational graphs.	10	CO5	L2

Faculty Signature CCI Signature HOD Signature

USN | | | | | | | | | | |

Internal Assessment Test II –November 2025
Answer Key

Sub:	DEEP LEARNING					Sub Code:	BCA701	Branch :	CSE AIML
Date:		Duration:	90 min	Max Marks:	50	Sem/Sec :	VII C		OBE

	Answer any FIVE FULL Questions	**MARKS**	**CO**	**RBT**
1 a	You are designing a Convolutional Neural Network (CNN) to classify handwritten digits from noisy images. Illustrate how you would choose the kernel size, number of filters, pooling strategy, and activation function to handle noise effectively. Justify how each chosen CNN component contributes to improving accuracy in this scenario. Answer :- Diagram 2M  Mention the stages and explain in detail, 1.Convolution Stage- usage of kernel or filters. (2M) 2.Detector Stage(2M) 3.Pooling Stage(2M) 4.Flattening and use of Softmax(2M)	10	CO4	L3
2 a	Discuss the variants of the basic convolution function. Answer:- Convolution means an operation that consists of many applications of convolution in parallel. This is because convolution with a single kernel can only extract one kind of feature, albeit at many spatial locations. Convolutional networks usually use multi-channel convolution, the linear operations they are based on are not guaranteed to be commutative, even if kernel-flipping is used We may want to skip over some positions of the kernel in order to reduce the computational cost (at the expense of not extracting our features as finely). We can think of this as downsampling the output of the full convolution function. If we want to sample only every *s* pixels in each direction in the output, we refer to	10	CO4	L2

	s as the **stride** of this downsampled convolution. Zero padding the input allows us to control the kernel width and the size of the output independently. Three special cases of the zero-padding setting - ➢ One is the extreme case in which no zero-padding is used whatsoever, and the convolution kernel is only allowed to visit positions where the entire kernel is contained entirely within the image. This is called **valid** convolution. In this case, all pixels in the output are a function of the same number of pixels in the input. However, the size of the output shrinks at each layer. If the input image has width m and the kernel has width k, the output will be of width $m - k + 1$. ➢ Another special case of the zero-padding setting is when just enough zero-padding is added to keep the size of the output equal to the size of the input. This is called **same** convolution. ➢ Other extreme case, which is referred to as **full** convolution, in which enough zeroes are added for every pixel to be visited k times in each direction, resulting in an output image of width $m + k - 1$. In some cases, we do not actually want to use convolution, but rather locally connected layers. This is sometimes also called **unshared convolution**, because it is a similar operation to convolution with a small kernel, but without sharing parameter across locations. **Tiled convolution** offers a compromise between a convolutional layer and a locally connected layer. Rather than learning a separate set of weights at *every* spatial location, we learn a set of kernels that we rotate through as we move through space. This means that immediately neighboring locations will have different filters, like in a locally connected layer, but the memory requirements for storing the parameters will increase only by a factor of the size of this set of kernels, rather than the size of the entire output feature map			
3 a	Explain the various datatypes used with CNN. Answer :- 3 types 1D -1M 2D -2M 3D-2M on Single channel and Multichannel **Dimension** / **Single channel** / **Multi channel** 1 D — **Audio waveform** - We discretize time and measure the amplitude of the waveform once per time step. — **Skeleton animation data** – Animations of 3-D computer-rendered characters are generated by altering the pose of a "skeleton" over time. At each point in time, the pose of the character is described by a specification of the angles of each of the joints in the character's skeleton. 2 D — **Audio data that has been preprocessed with a Fourier transform** - We can transform the audio waveform into a 2D tensor with different rows corresponding to different frequencies and different columns corresponding to different points in time. — **Color image data** - One channel contains the red pixels, one the green pixels, and one the blue pixels. 3 D — **Volumetric data** - A common source of this kind of data is medical imaging technology, such as CT scans. — **Color video data** - One axis corresponds to time, one to the height of the video frame, and one to the width of the video frame.	5	CO4	L2
3 b	Differentiate between Stride and Pooling in CNN. Scheme:- 5 Differences each (5X1=5M) Stride / Pooling	5	CO4	L2

	Parameter used in Convolution	Its an operation		
	Happens in Convolution layer	Happens in Pooling Layer		
	Determines how convolution filter slides	Aggregate max/avg		
	Can reduce feature map size	Reduce feature map size		
	Stride values can be 1, 2 etc.	Max and AveragePooling		

4	Explain Bidirectional RNN and Deep Recurrent Networks. Answer:- Bidirectional RNN with Diagram -5Marks	10	CO5	L2

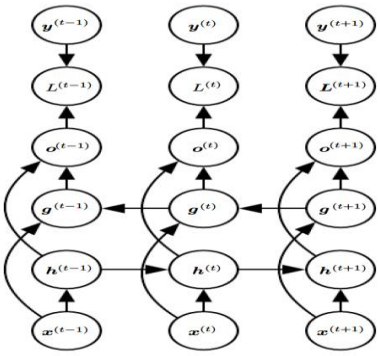


Figure 10.11: Computation of a typical bidirectional recurrent neural network, meant to learn to map input sequences $\boldsymbol{x}$ to target sequences $\boldsymbol{y}$, with loss $L^{(t)}$ at each step t. The $\boldsymbol{h}$ recurrence propagates information forward in time (towards the right) while the $\boldsymbol{g}$ recurrence propagates information backward in time (towards the left). Thus at each point t, the output units $\boldsymbol{o}^{(t)}$ can benefit from a relevant summary of the past in its $\boldsymbol{h}^{(t)}$ input and from a relevant summary of the future in its $\boldsymbol{g}^{(t)}$ input.

- However, in many applications we want to output a prediction of $y^{(t)}$ which may depend on the whole input sequence. For example, in speech recognition, because of the linguistic dependencies between nearby words: if there are two interpretations of the current word that are both acoustically plausible, we may have to look far into the future (and the past) to disambiguate them.
- Bidirectional recurrent neural networks (or bidirectional RNNs) were invented to address that need.
- They have been extremely successful in applications such as handwriting recognition, speech recognition and bioinformatics.
- **Bidirectional RNNs combine an RNN that moves forward through time beginning from the start of the sequence with another RNN that moves backward through time beginning from the end of the sequence.**

Deep Recurrent Networks with Diagram -5Marks

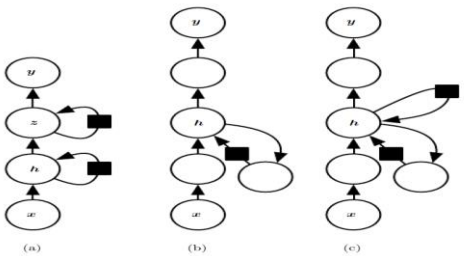
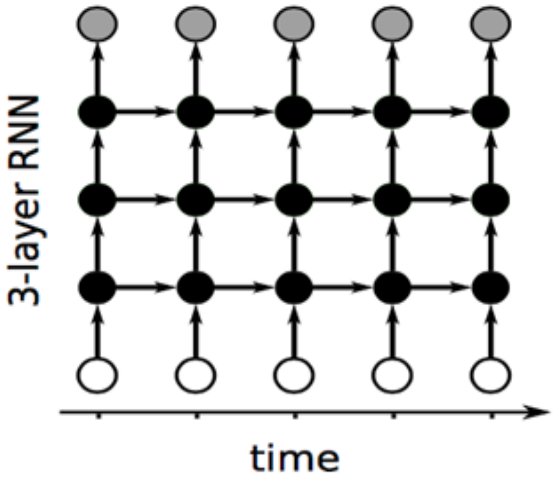
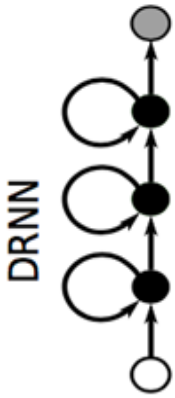
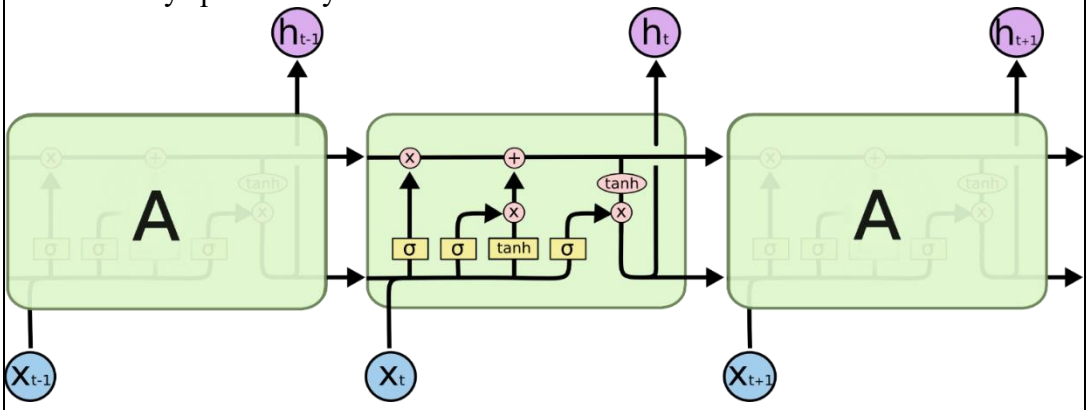
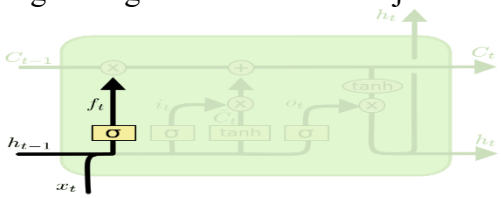
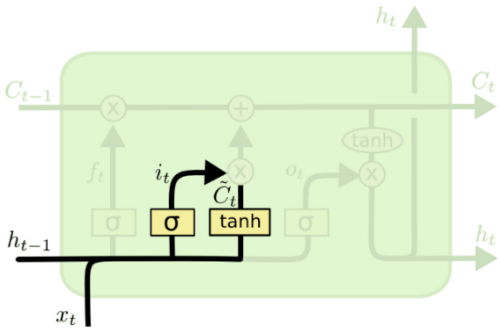
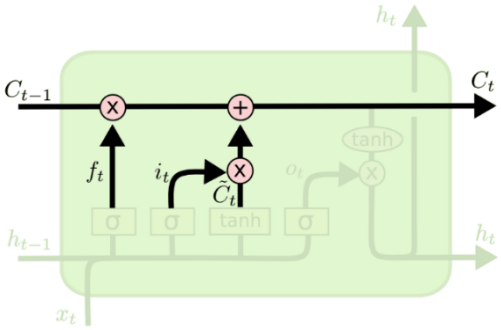
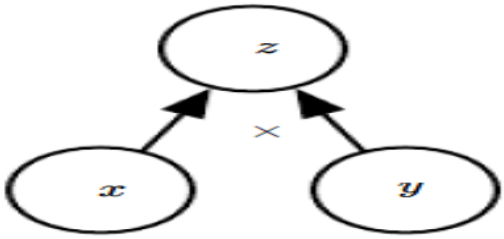


Figure 10.13: A recurrent neural network can be made deep in many ways (Pascanu *et al.*, 2014a). *(a)*The hidden recurrent state can be broken down into groups organized hierarchically. *(b)*Deeper computation (e.g., an MLP) can be introduced in the input-to-hidden, hidden-to-hidden and hidden-to-output parts. This may lengthen the shortest path linking different time steps. *(c)*The path-lengthening effect can be mitigated by introducing skip connections.

The computation in most RNNs can be decomposed into three blocks of parameters and associated transformations:
1. from the input to the hidden state,
2. from the previous hidden state to the next hidden state, and
3. from the hidden state to the output.

	 			
5	Given a sequence where long-term dependencies are crucial, explain how each LSTM gate (input gate, forget gate, and output gate) would operate on the sequence during processing. Illustrate how gate activations change the cell state and hidden state at each step, and analyze how modifying the gate values would affect the model's ability to retain or eliminate information. Answer • LSTMs also have this chain like structure, but the repeating module has a different structure. • Instead of having a single neural network layer, there are four, interacting in a very special way.  Forget Gate with Diagram and equations 4M The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely get rid of this." Let's go back to our example of a language model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject  $$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$ Input Gate with Diagram and equations 3M	10	CO5	L2

	The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state. In the example of our language model, we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.  $i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$ $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$ Output Gate with Diagram and equations 3M It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it. We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value. In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.  $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$			
6	Explain how Recurrent Neural Network (RNN) processes the data sequence using computational graphs. Answer:- Computational Graph 2M **Computational graph** It is a way to formalize the structure of a set of computations. ➢ Many ways of formalizing computation as graphs are possible. ➢ Here, we use each node in the graph to indicate a variable. The variable may be a scalar, vector, matrix, tensor, or even a variable of another type. ➢ If a variable *y* is computed by applying an operation to a variable *x*, then we draw a directed edge from *x* to *y (x->y)*. ➢ We sometimes annotate the output node with the name of the operation applied, and other times omit this label when the operation is clear from context.	10	CO5	L2

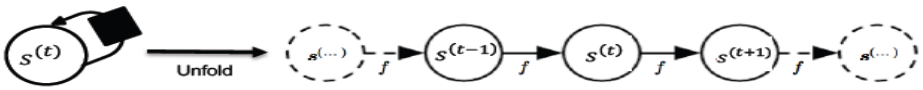


Equations 3M

➢ Ex) $s^{(t)} = f(s^{(t-1)}; \theta)$

Unfolding equation

$$\begin{aligned} s^{(t)} &= f(s^{(t-1)}; \theta) \\ &= f(f(s^{(t-2)}; \theta) \\ &= f(f(f(s^{(t-3)}; \theta) \\ &\vdots \\ &= f(f(f(\cdots(f(s^{(1)}; \theta)) \end{aligned}$$



For example, consider the classical form of a dynamical system:

$$s^{(t)} = f(s^{(t-1)}; \theta), \tag{10.1}$$

where $s^{(t)}$ is called the state of the system.

Equation 10.1 is recurrent because the definition of s at time t refers back to the same definition at time $t-1$.

For a finite number of time steps τ, the graph can be unfolded by applying the definition $\tau - 1$ times. For example, if we unfold equation 10.1 for $\tau = 3$ time steps, we obtain

$$s^{(3)} = f(s^{(2)}; \theta) \tag{10.2}$$

$$= f(f(s^{(1)}; \theta); \theta) \tag{10.3}$$

Unfolding the equation by repeatedly applying the definition in this way has yielded an expression that does not involve recurrence. Such an expression can now be represented by a traditional directed acyclic computational graph. The unfolded computational graph of equation 10.1 and equation 10.3 is illustrated in figure 10.1.

Figure 10.1: The classical dynamical system described by equation 10.1, illustrated as an unfolded computational graph. Each node represents the state at some time t and the function f maps the state at t to the state at $t+1$. The same parameters (the same value of θ used to parametrize f) are used for all time steps.

RNN process with explanation 5M

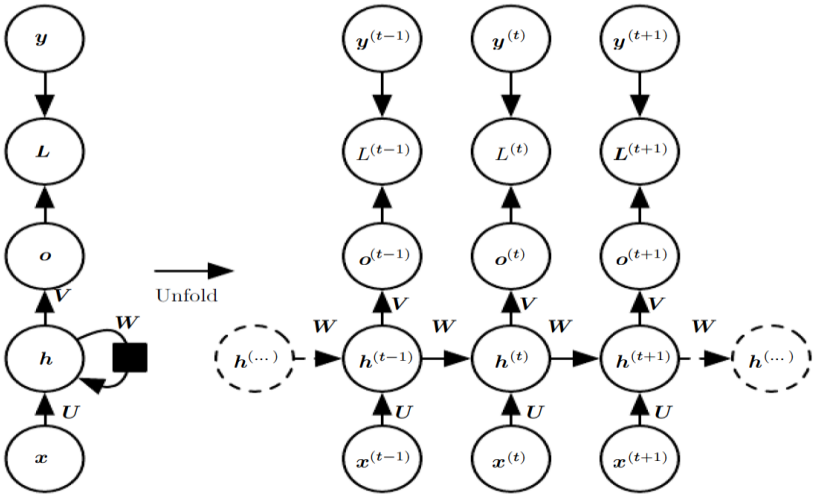


Figure 10.3: The computational graph to compute the training loss of a recurrent network that maps an input sequence of $\boldsymbol{x}$ values to a corresponding sequence of output $\boldsymbol{o}$ values. A loss L measures how far each $\boldsymbol{o}$ is from the corresponding training target $\boldsymbol{y}$. When using softmax outputs, we assume $\boldsymbol{o}$ is the unnormalized log probabilities. The loss L internally computes $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{o})$ and compares this to the target $\boldsymbol{y}$. The RNN has input to hidden connections parametrized by a weight matrix $\boldsymbol{U}$, hidden-to-hidden recurrent connections parametrized by a weight matrix $\boldsymbol{W}$, and hidden-to-output connections parametrized by a weight matrix $\boldsymbol{V}$. Equation 10.8 defines forward propagation in this model. *(Left)*The RNN and its loss drawn with recurrent connections. *(Right)*The same seen as an time-unfolded computational graph, where each node is now associated with one particular time instance.

1. Recurrent networks that produce an output at each time step and have recurrent connections **between hidden units**..
2. Recurrent networks that produce an output at each time step and have recurrent connections **only from the output at one time step to the hidden units at the next time step**.
3. Recurrent networks with **recurrent connections between hidden units**, that read an **entire sequence** and then **produce a single output**.

Faculty Signature CCI Signature HOD Signature