4)

a) Output Shape : (None, 100, 50)
100 is the length of input sequence and 50 denotes the embedding dimension

b) Number of parameters : (shape_x) · (voc_size)

$= 50 \times 10000$

$= 500000$

# of params = 500000

c) Output shape : (None, 100, 40)
100 is the length of ~~input~~ return sequence which comes from the algorithm and 40 is the state dimension of LSTM.

d) Number of parameters = $4 \times ((\text{shape\_h}) \cdot (\text{shape\_h} + \text{shape\_x}) + \text{shape\_h})$

$= 4 \times (40 \cdot (40 + 50) + 40)$

$= 14560$

# of params = 14560
The number of params in LSTM is simply 4 times the number of params in RNN.

e) Output shape : (None, 100, 40)
Similar to the first LSTM layer.

f) Output shape : (None, 4000)
Flatten the output from the last LSTM layer which has shape (None, 100, 40)

g) Number of parameters = 0
Flatten layers have no parameters
# of params = 0

3) We know that,

$$E[f(X)] = \sum_{x \in X} p(x) f(x)$$

$$\pi(a|s) = P(A=a \mid S=s)$$

which means the probability of taking action a given the state s

$$\therefore \; V_\pi(s_t) = E_{a \sim \pi(\cdot|s_t)}\left[Q_\pi(s_t, a)\right]$$

$\pi(\text{"left"}|s_t) = 0.2$ $\quad Q_\pi(s_t, \text{"left"}) = 100$
$\pi(\text{"right"}|s_t) = 0.3$ $\quad Q_\pi(s_t, \text{"right"}) = 200$
$\pi(\text{"up"}|s_t) = 0.5$ $\quad Q_\pi(s_t, \text{"up"}) = 300$

$$V_\pi(s_t) = \sum_{a \in A} \pi(a|s_t) \cdot Q_\pi(s_t, a)$$

$$= \pi(\text{"left"}/s_t) \cdot Q_\pi(s_t, \text{"left"}) + \pi(\text{"right"}/s_t) \cdot Q_\pi(s_t, \text{"right"}) + \pi(\text{"up"}/s_t) \cdot Q_\pi(s_t, \text{"up"})$$

$$= (0.2)(100) + (0.3)(200) + (0.5)(300)$$

$= 20 + 60 + 150$

$\boxed{V_{\pi}(s_1) = 230}$

Ans : $\boxed{V_{\pi}(s_1) = 230}$

5)

a) W_q, W_K and W_V are trainable parameter matrices in our single - head attention layer. To get their dimension,

$$\tilde{Q} = W_q \; Q \qquad \therefore W_q \in R^{10 \times 20}$$

$$\begin{tikzcd}
10 \times 100 \arrow[r, leftrightarrow, bend right, "\text{should match}"'] & n \times m \arrow[r, leftrightarrow, bend right, "\text{should be equal}"'] & 20 \times 100
\end{tikzcd}$$

$$\tilde{K} = W_K \; K \qquad \therefore W_K \in R^{10 \times 30}$$

$$\begin{tikzcd}
10 \times 20 \arrow[r, leftrightarrow, bend right, "\text{should match}"'] & n \times m \arrow[r, leftrightarrow, bend right, "\text{same}"'] & 30 \times 20
\end{tikzcd}$$

$$\tilde{V} = W_V \; V \qquad \therefore W_V \in R^{10 \times 30}$$

$$\begin{tikzcd}
10 \times 20 \arrow[r, leftrightarrow, bend right] & n \times m \arrow[r, leftrightarrow, bend right] & 30 \times 20
\end{tikzcd}$$

Final trainable params : $10 \times 20 + 10 \times 30 + 10 \times 30$

$\boxed{\text{Ans} = 800}$

b) Output Shape : $C = \tilde{V}\ \text{softmax}(\tilde{K}^T \tilde{Q})$

$$= \underset{10\times 20}{\tilde{V}}\ \text{softmax}(\underset{20\times 10}{\tilde{K}^T}\ \underset{10\times 100}{\tilde{Q}})$$

Hence the resulting shape $= \underset{10\times 20}{\tilde{V}}\ \underset{20\times 100}{R}$

$$= \underset{10\times 100}{C}$$

→ Hence output shape $C \in R^{10\times 100}$

c) We can say that if m single head attention modules are connected, and each single head gives output of dimension $p\times q$ then the connected multihead gives $mp \times q$ dimension output.

∴ We know that

$C \in R^{10\times 100}$ output from 1 single head attention $\qquad Q \in R^{20\times 100}$ input

∴ $m = 20/10 = 2$, 2 single heads should be connected.

d) Total number of trainable params = m × (trainable params in single head)

= 2 × (800)

= 1600

# of params = 1600.

1)

a) False
b) False
c) False
d) True
e) False

2)

a) d
b) c
c) e
d) b
e) d
f) c
g) d
h) b
i) b
j) b, d

6)

a) In reinforcement learning, we define 's' as the state in which the given task is or it can also be defined as the current frame while we define action 'a' as the things which can be done by an agent at current state. Here 'π' can be defined as policy function which basically denotes the probability of taking action 'a' given current state 's'. Randomness means that the actions and state changes that are happening are not fixed. For example if action consists of ["up","down"], then any one of them can be performed randomly considering the policy function. The randomness comes from 'actions' and 'state changes' in reinforcement learning.

b) <u>Targeted attack</u>: The attacker tries to get an image classified as a particular target class. which is not the actual or true target class. Fake image is generated by setting a fake target and using network parameters. Here attacker tries to ~~fass~~ make the model err ~~to~~ in classifying the current image to desired false class. Example classify image of dog as image of lion.

Untargeted attack: The attacker just tries to make the model err, without any intention of using a particular target (fake) class. Attackers usually add some small deviations to images which are normally not visible to human eye, in order to make the model err. In general attacker tries to make model err and classify an image of dog as an image of any other class except dog.

→ The main difference is that attacker has no control over target class in the untargeted attacks while they do have some control in the case of targetted attacks.

c) Autoencoder: It is just a neural net that regenerates or reconstructs the given input. We can say that its a generalization of PCA. Training data should contain noisy inputs and clean targets. It consists of both encoder and decoder.

Variational Autoencoder: Similar to Autoencoder but it also has some probability techniques.

Difference is that AE has discontinuity and on the other hand VAE is continuous. and AE doesn't have generative capabilities as compared to VAE. We cannot train VAE with just one generation loss as it would just behave as AE.