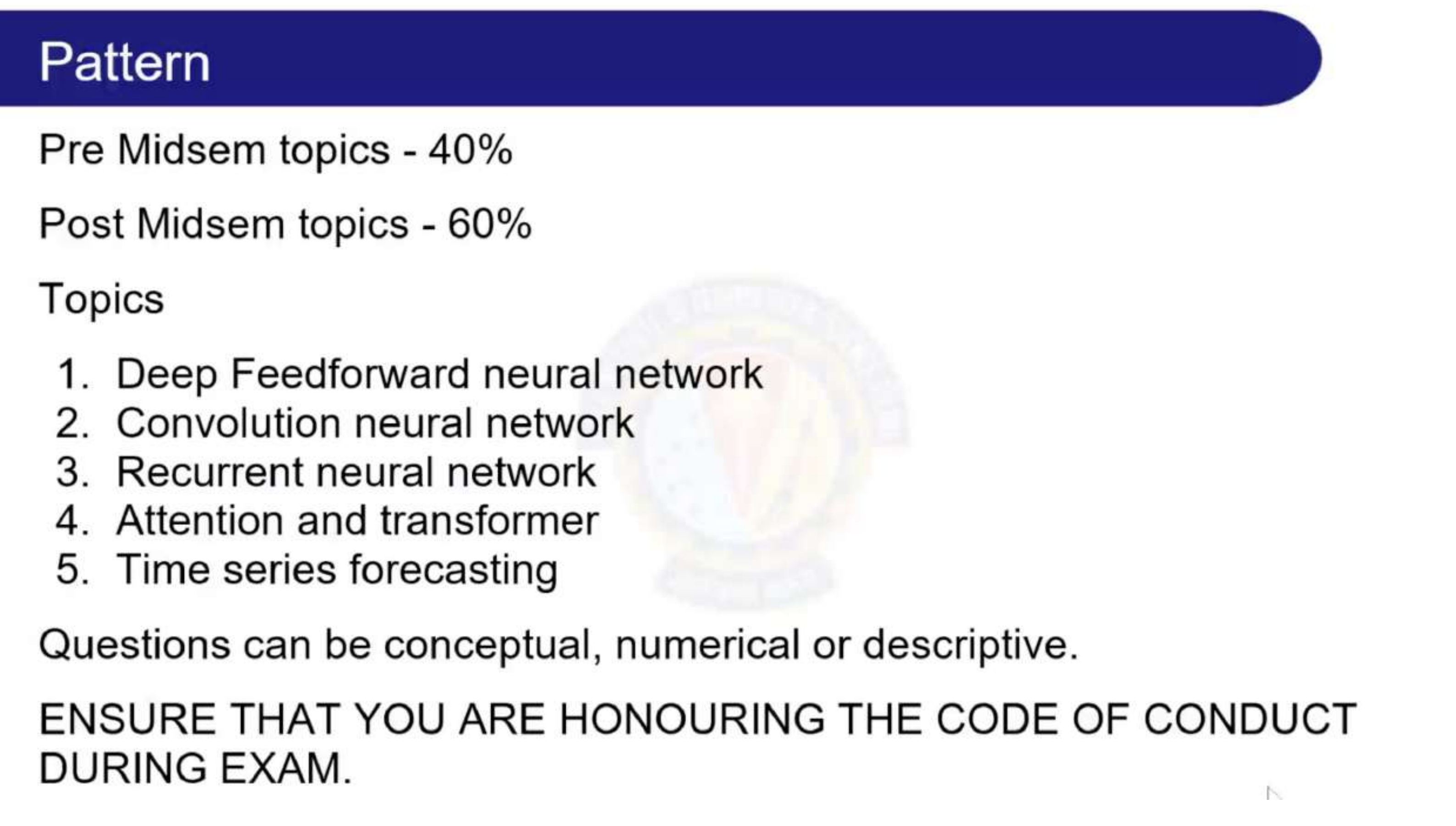


Pattern

Pre Midsem topics - 40%

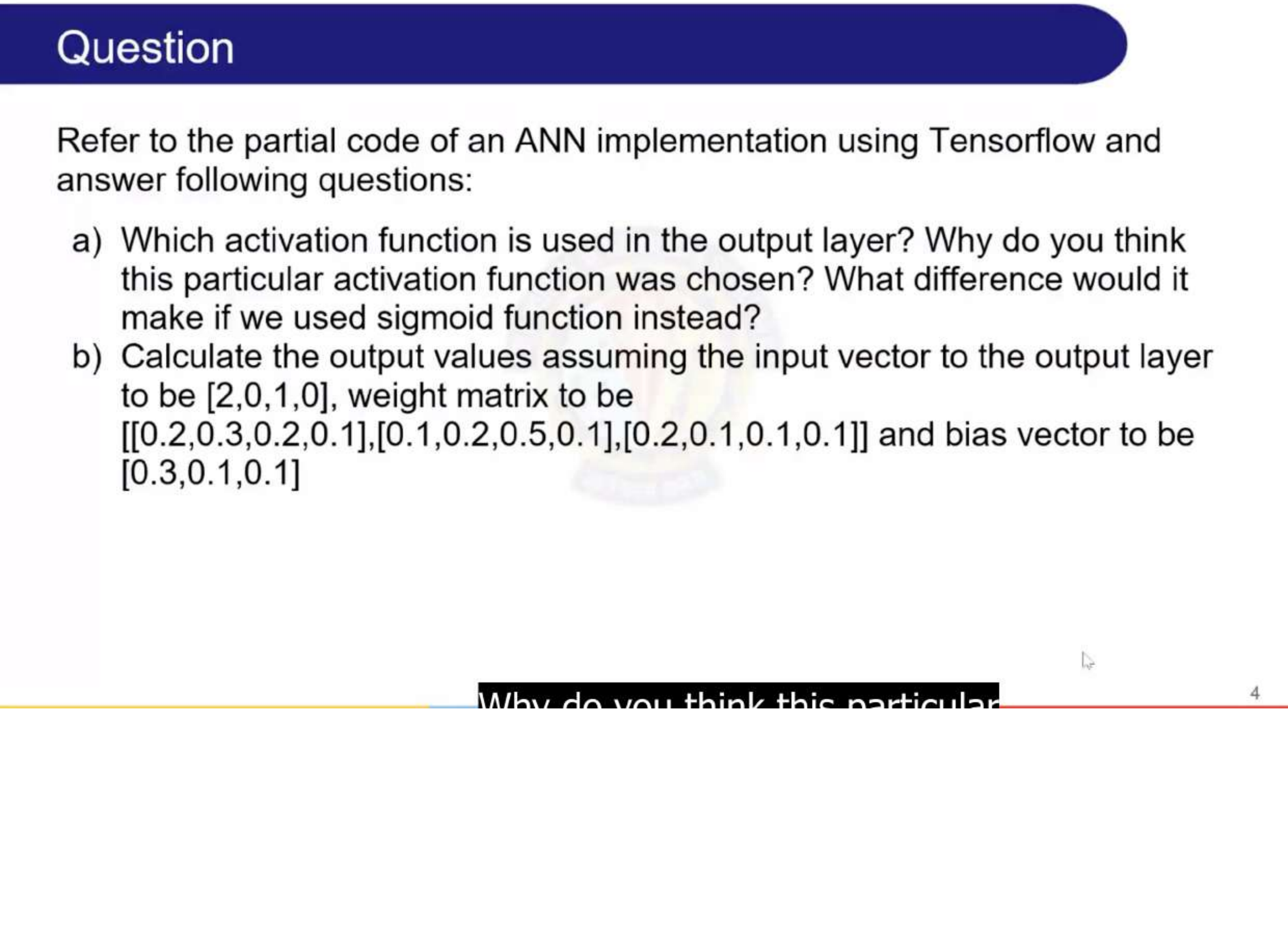
Post Midsem topics - 60%

Topics

1. Deep Feedforward neural network
2. Convolution neural network
3. Recurrent neural network
4. Attention and transformer
5. Time series forecasting

Questions can be conceptual, numerical or descriptive.

ENSURE THAT YOU ARE HONOURING THE CODE OF CONDUCT DURING EXAM.



Question

Refer to the partial code of an ANN implementation using Tensorflow and answer following questions:

a) Which activation function is used in the output layer? Why do you think this particular activation function was chosen? What difference would it make if we used sigmoid function instead?

b) Calculate the output values assuming the input vector to the output layer to be [2,0,1,0], weight matrix to be [[0.2,0.3,0.2,0.1],[0.1,0.2,0.5,0.1],[0.2,0.1,0.1,0.1]] and bias vector to be [0.3,0.1,0.1]

Why do you think this particular

Question

```
import tensorflow as tf
data = pd.read_csv('train.csv')
a = tf.placeholder(tf.float32, shape=[None, 16])
b = tf.placeholder(tf.float32, shape=[None, 3])
phi = { 'alpha': tf.Variable(tf.random_normal([16, 8])),
'beta': tf.Variable(tf.random_normal([8, 3])) }
omega = { 'alpha': tf.Variable(tf.random_normal([8])),
'beta': tf.Variable(tf.random_normal([3])) }
tl = tf.add(tf.matmul(x, phi['alpha']), omega['alpha'])
tl = tf.nn.relu(tl)
fl = tf.matmul(tl, phi['beta']) + omega['beta'])
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(fl, b))
optimizer = tf.train.AdamOptimizer(learning_rate=0.5).minimize(cost)
init = tf.initialize_all_variables()
```

Question

```
import tensorflow as tf
data = pd.read_csv('train.csv')
a = tf.placeholder(tf.float32, shape=[None, 16])
b = tf.placeholder(tf.float32, shape=[None, 3])
phi = { 'alpha': tf.Variable(tf.random_normal([16, 8])),
'beta': tf.Variable(tf.random_normal([8, 3])) }
omega = { 'alpha': tf.Variable(tf.random_normal([8])),
'beta': tf.Variable(tf.random_normal([3])) }
tl = tf.add(tf.matmul(x, phi['alpha']), omega['alpha'])
tl = tf.nn.relu(tl)
fl = tf.matmul(tl, phi['beta']) + omega['beta'])
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(fl, b))
optimizer = tf.train.AdamOptimizer(learning_rate=0.5).minimize(cost)
init = tf.initialize_all_variables()
```

Answer

a) Softmax is used in the output layer. The sum of output values from all the output nodes would be 1 so it is likely that the required output is probability distribution of a given variable. Sigmoid could also be used but the output values will not sum to 1.

b) w.x+b = [0.9, 0.8, 0.6]; softmax values = [0.38, 0.34, 0.28]

Question

Refer to the following code snippet.

$$\mu = \frac{1}{m}\Sigma h_i$$

$$\sigma = \left[\frac{1}{m}\Sigma (h_i - \mu)^2\right]^{1/2}$$

64 2
128 2
256 3
512 6

```
>>> from tensorflow.keras.applications.vgg16 import VGG16
>>> from tensorflow.keras.models import Model
>>> model = VGG16()
>>> model.layers.pop()
>>> model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
```

$$\rightarrow h_i = \gamma h_{i(norm)} + \beta \qquad h_{i(norm)} = \frac{(h_i - \mu)}{\sigma + \boxed{\epsilon}} \rightarrow \text{smoothing.}$$

a) If we add Batch Normalization after every convolution layer in the modified VGG16 model (popping last layer), what will be the total number of additional trainable parameters, beta's and gamma's? → re-scaling

Shifting.

b) What will be the total number of non-trainable parameters, i.e., means and variances in the first 3 Batch Normalization layers?

TP = no. of layers x no. of channels x 2 parameters (β, γ)

n.TP = no. of layers x no. of channels x 2 par (mean, var)

Answer

a) 8448

b) 512 (2*64*2 + 128*2)

Convolution Channels Conv layer Channels	Number Layers No. of layers	Trainable Parameters No. of (gamma and beta) Tp	Non-Trainable Parameters No. of layers x no. of (mean and variance)	Total Parameters
64	2	2*64*2 = 256	2*64*2 = 256	512
128	2	2*128*2 = 512	2*128*2 = 512	1,024
256	3	3*256*2 = 1,536	3*256*2 = 1,536	3,072
512	6	6*512*2 = 6,144	6*512*2 = 6,144	12,288
		8,448	**8,448**	**16,896**

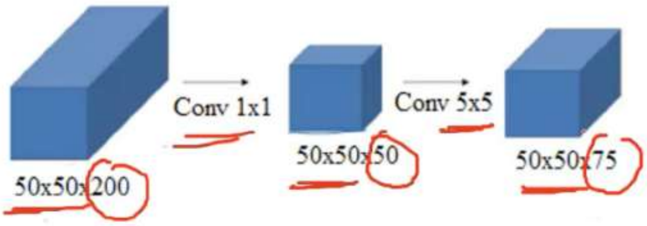
Question

In the following figure, 1x1 operators are used first to process the same 50x50 images of depth 200, and first output 50x50 images of depth 50, and then 5x5 operators are used to output 50x50 images of depth 75.

a) What is the padding size used in the first step and padding size in the last step?

b) How many multiplication operations are needed here?

ii.



Answer

a) For 1x1 convolution, padding size = 0.

For 5x5 convolution, padding size = 2.

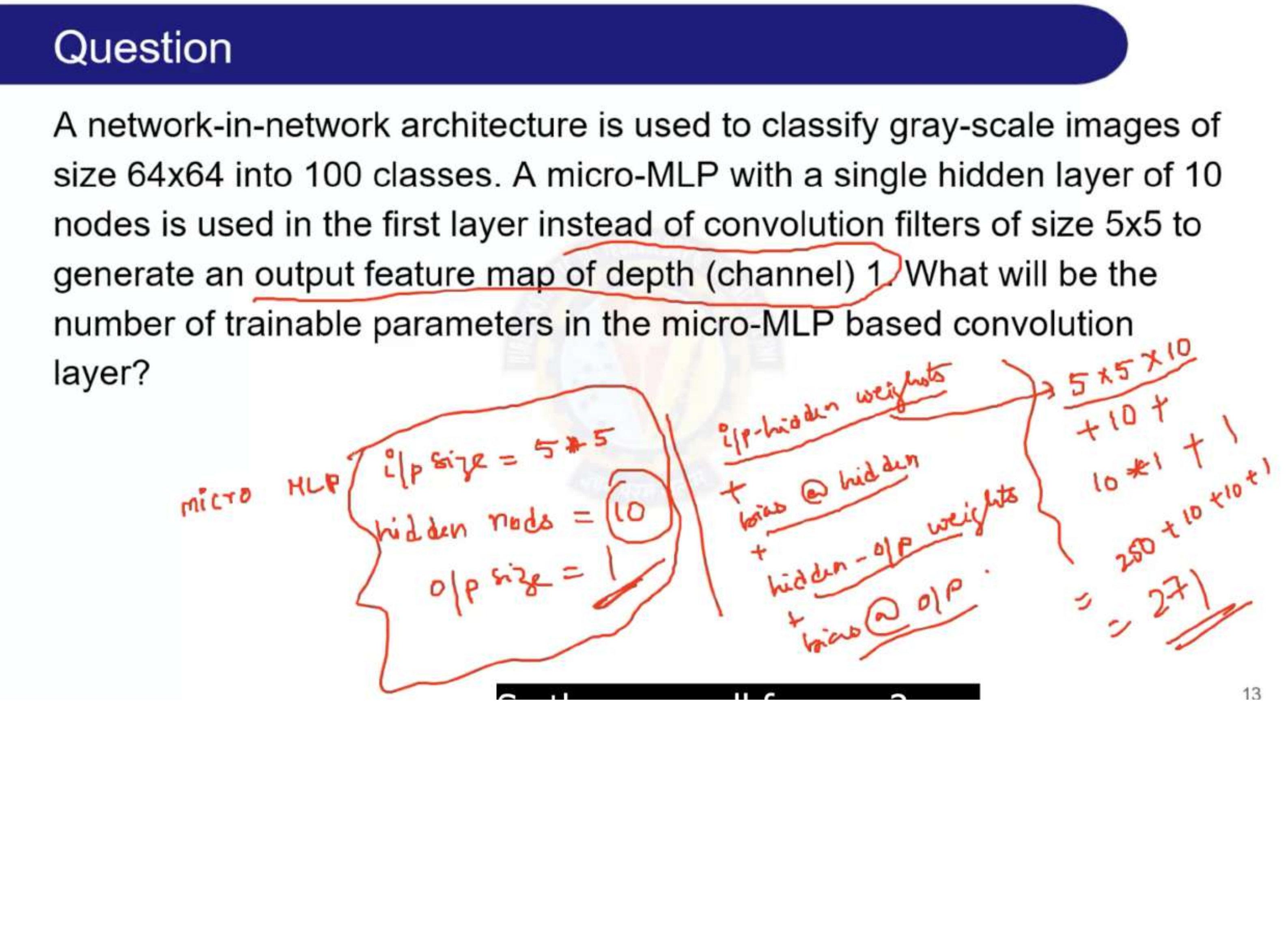
b) For 1x1 convolution,

# of multiplication operations = 50x50x50x200 = 2.5x107

For 5x5 convolution,

# of multiplication operations = 50x50x5x5x50x75 = 234375000

So, total # of operations = 236,875,000



Question

A network-in-network architecture is used to classify gray-scale images of size 64x64 into 100 classes. A micro-MLP with a single hidden layer of 10 nodes is used in the first layer instead of convolution filters of size 5x5 to generate an output feature map of depth (channel) 1. What will be the number of trainable parameters in the micro-MLP based convolution layer?

micro MLP

i/p size = 5 * 5

hidden nods = 10

o/p size = 1

i/p-hidden weights

+

bias @ hidden

+

hidden - o/p weights

+

bias @ o/p.

$\rightarrow 5 \times 5 \times 10$

$+ 10 +$

$10 * 1 + 1$

$= 250 + 10 + 10 + 1$

$= 271$

Answer

Micro-MLP input size: 5*5, Hidden nodes 10, output size 1

Total # of trainable parameters

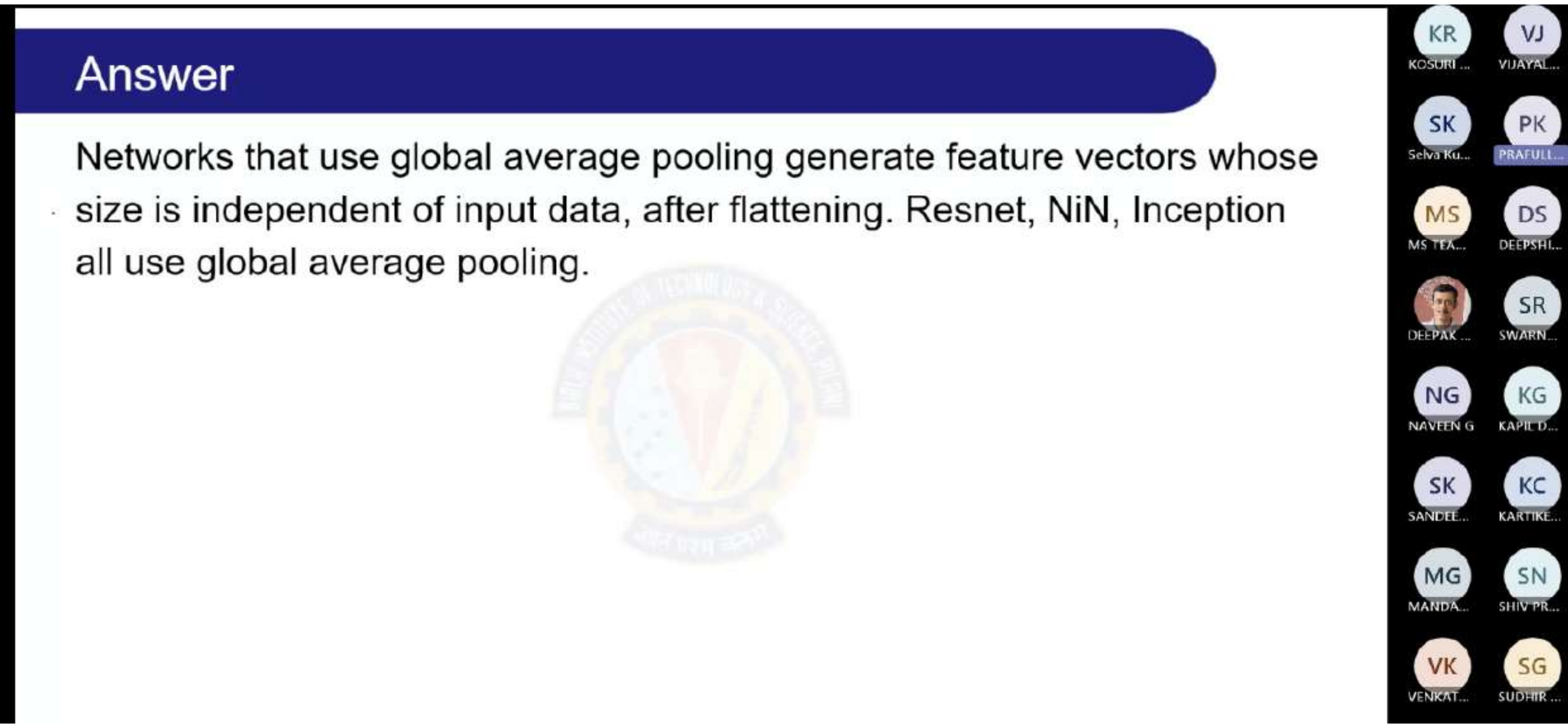
= 5*5*10 (input-hidden weights) + 10 (bias @ hidden) + 10*1 (hidden-to-output nodes) +1 (bias @ output)

= 271

Because the feature map is only

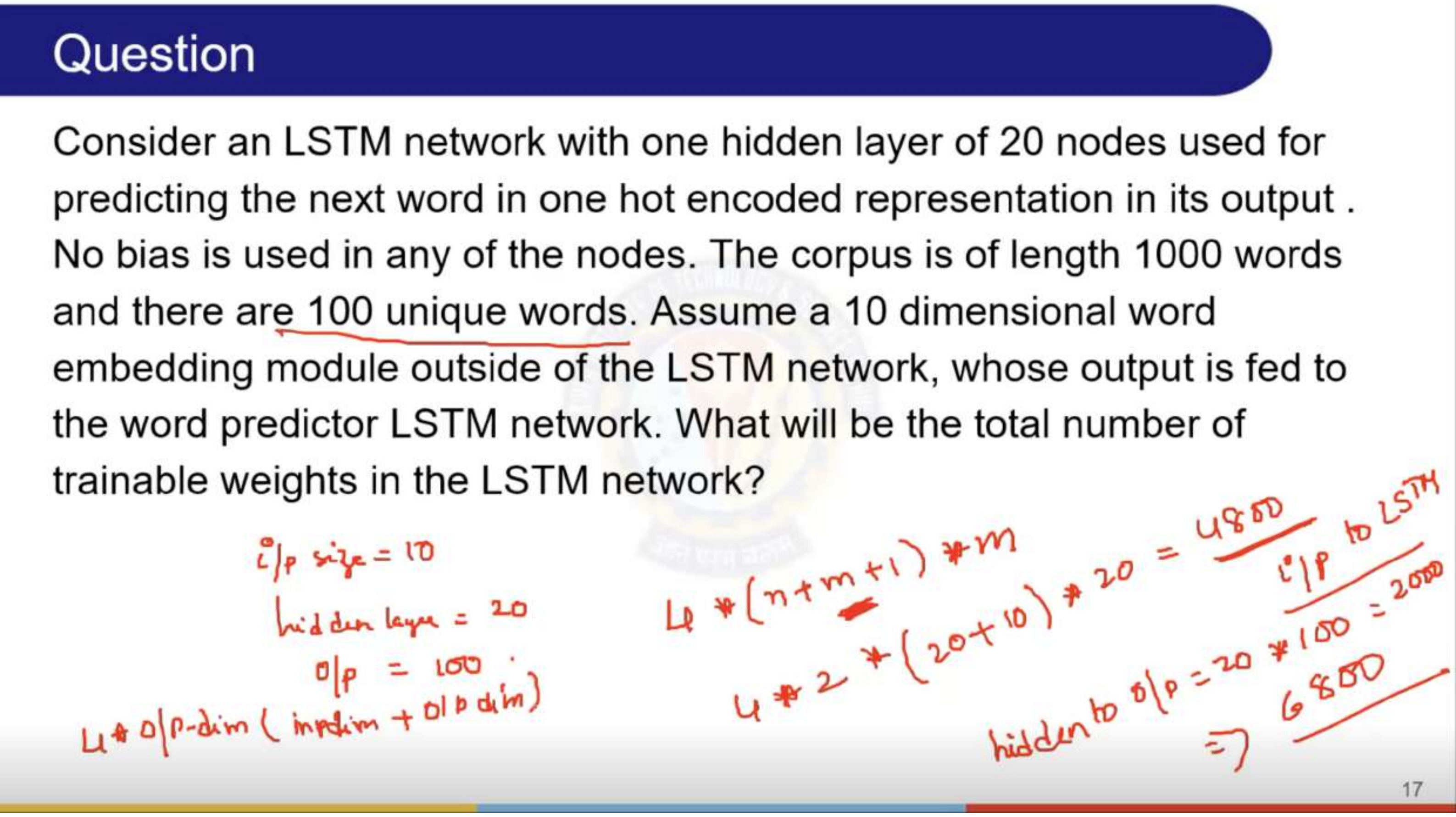
Question

Which of the following networks is (are) more suited architecturally for classifying images of varied size (e.g., the input image database has images of size 64x64, 96x96, 128x96, etc.)? No image rescaling is permitted.



Answer

Networks that use global average pooling generate feature vectors whose size is independent of input data, after flattening. Resnet, NiN, Inception all use global average pooling.



Question

Consider an LSTM network with one hidden layer of 20 nodes used for predicting the next word in one hot encoded representation in its output . No bias is used in any of the nodes. The corpus is of length 1000 words and there are 100 unique words. Assume a 10 dimensional word embedding module outside of the LSTM network, whose output is fed to the word predictor LSTM network. What will be the total number of trainable weights in the LSTM network?

i/p size = 10

hidden layer = 20

o/p = 100

4 * o/p-dim (inpdim + o/p dim)

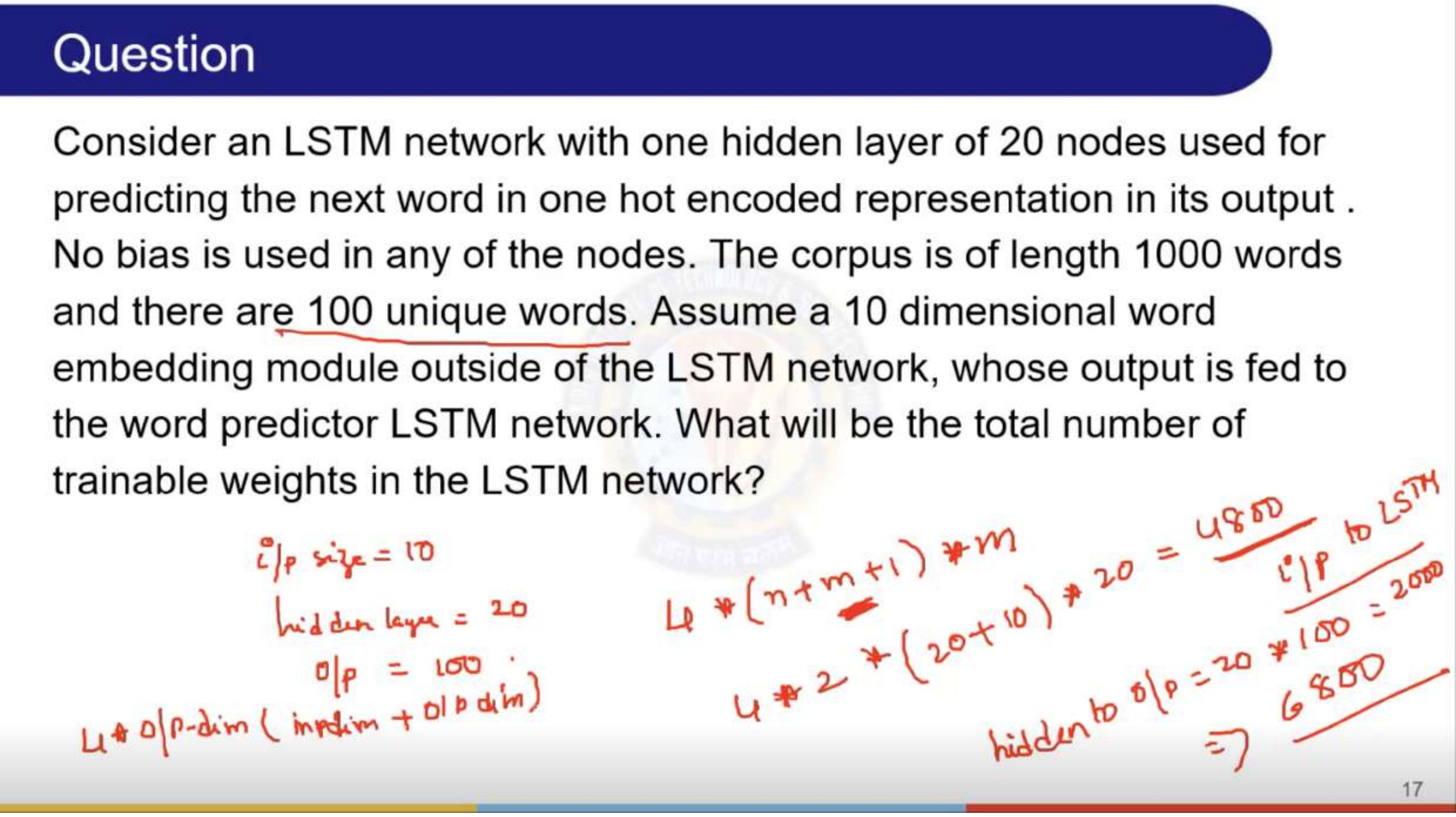
4 * (n + m + 1) * m

4 * 2 * (20 + 10) * 20 = 4800

i/p to LSTM

hidden to o/p = 20 * 100 = 2000

=> 6800



Question

Consider an LSTM network with one hidden layer of 20 nodes used for predicting the next word in one hot encoded representation in its output . No bias is used in any of the nodes. The corpus is of length 1000 words and there are 100 unique words. Assume a 10 dimensional word embedding module outside of the LSTM network, whose output is fed to the word predictor LSTM network. What will be the total number of trainable weights in the LSTM network?

i/p size = 10

hidden layer = 20

o/p = 100 .

4 * o/p-dim (inpdim + o/p dim)

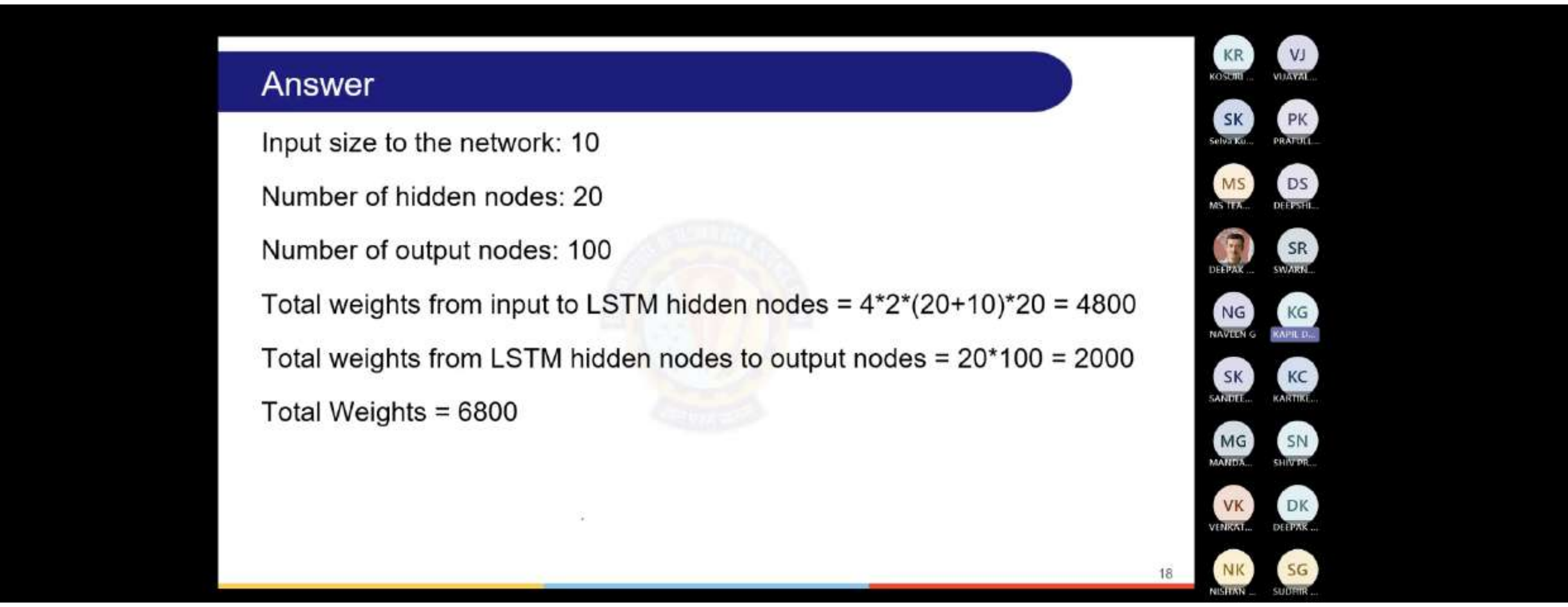
4 * (n + m + 1) * m

4 * 2 * (20 + 10) * 20 = 4800

i/p to LSTM

hidden to o/p = 20 * 100 = 2000

=> 6800



Answer

Input size to the network: 10

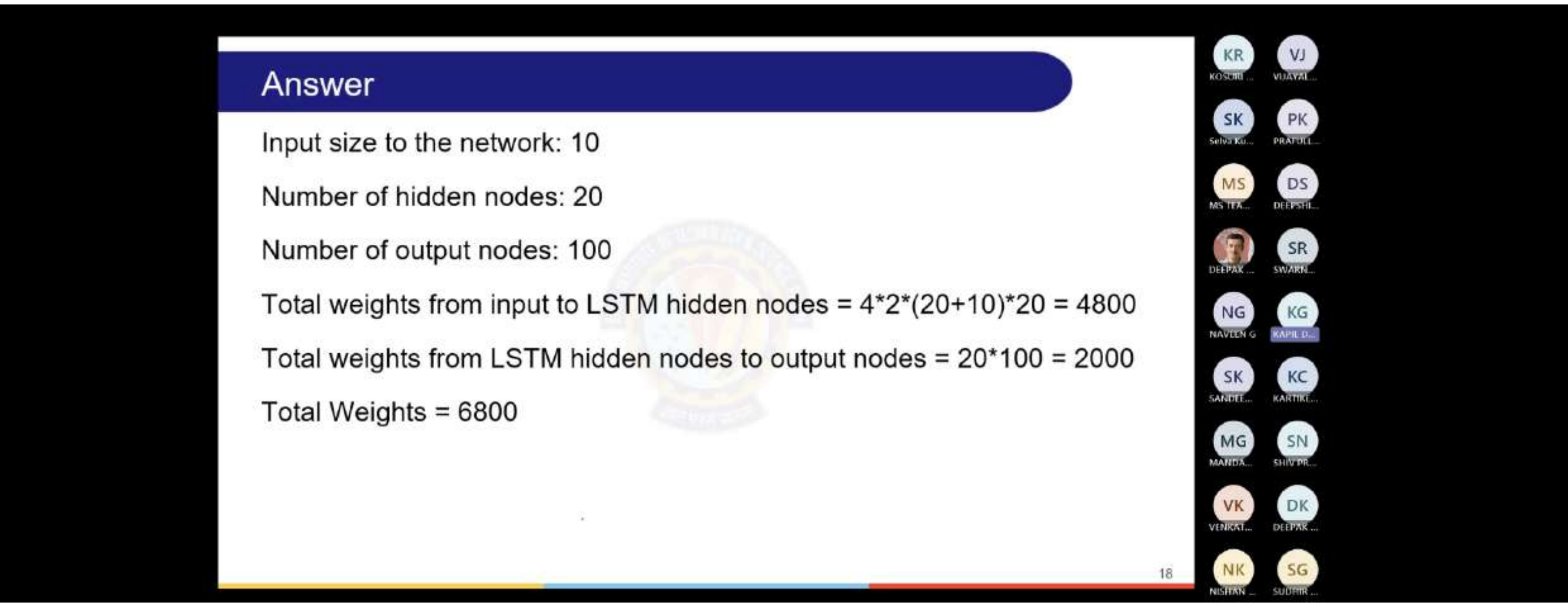
Number of hidden nodes: 20

Number of output nodes: 100

Total weights from input to LSTM hidden nodes = 4*2*(20+10)*20 = 4800

Total weights from LSTM hidden nodes to output nodes = 20*100 = 2000

Total Weights = 6800



Answer

Input size to the network: 10

Number of hidden nodes: 20

Number of output nodes: 100

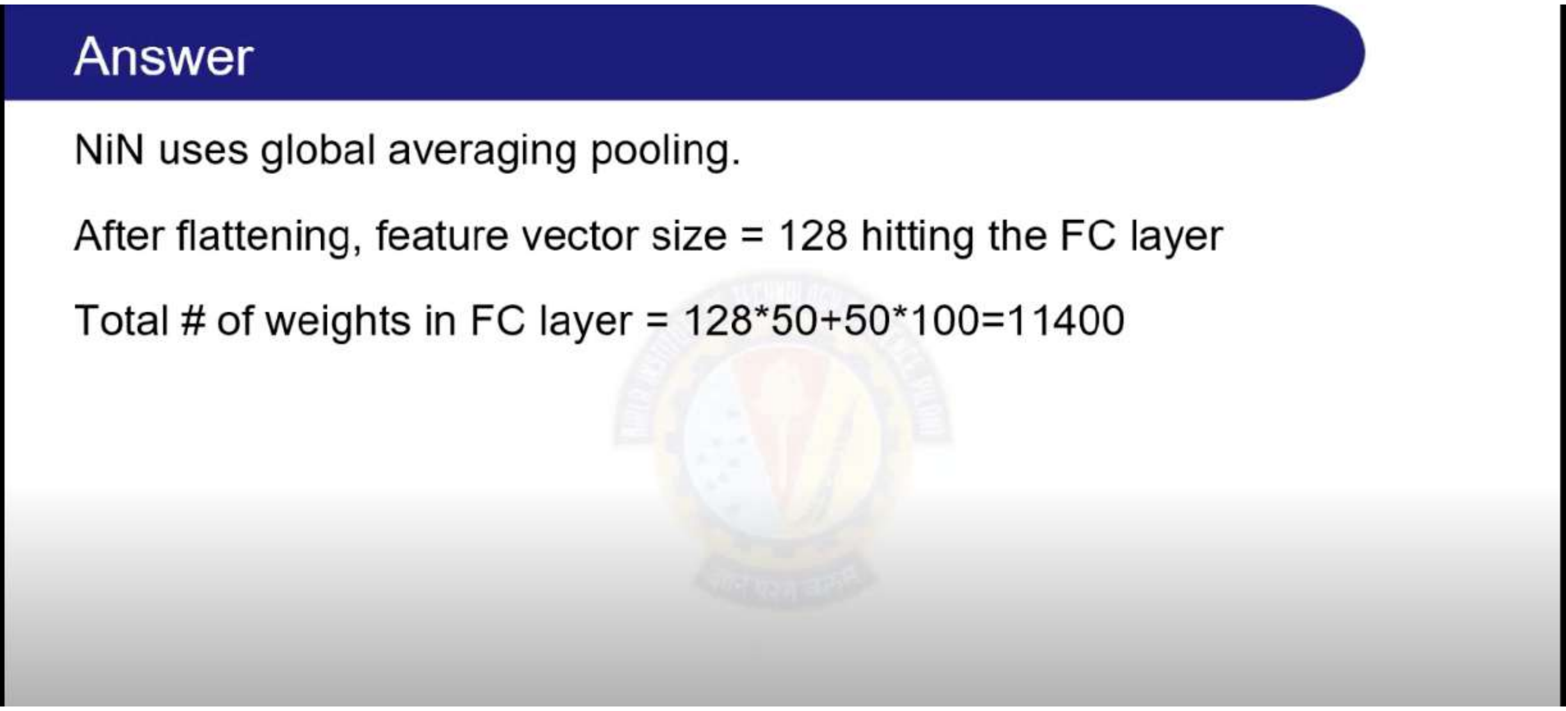
Total weights from input to LSTM hidden nodes = 4*2*(20+10)*20 = 4800

Total weights from LSTM hidden nodes to output nodes = 20*100 = 2000

Total Weights = 6800

Question

In a network in network architecture, one hot output encoding is used to classify 100 classes of objects. The last convolution layer generates 7x7x128 features maps (depth=128). Assuming the fully connected subnetwork has one hidden layer with 50 nodes, what will be the total number of trainable weights (excluding biases) in the fully connected subnetwork (from the last convolution layer to the output layer)?



Answer

NiN uses global averaging pooling.

After flattening, feature vector size = 128 hitting the FC layer

Total # of weights in FC layer = 128*50+50*100=11400

Consider a vanilla RNN network with one hidden layer of 20 nodes used for predicting the next word in a text corpus. No bias is used in any of the nodes. The corpus is of length 1000 words and there are 50 unique words. Assume a 10 dimensional word embedding module outside of the RNN network, whose output is fed to the word predictor RNN network. One hot encoding is used in the output. What will be the total number of trainable parameters in the RNN network?

Answer

Input-to-hidden weights = (20+10)*20

Hidden-to-output weights = 20*50

Total weights = 1600

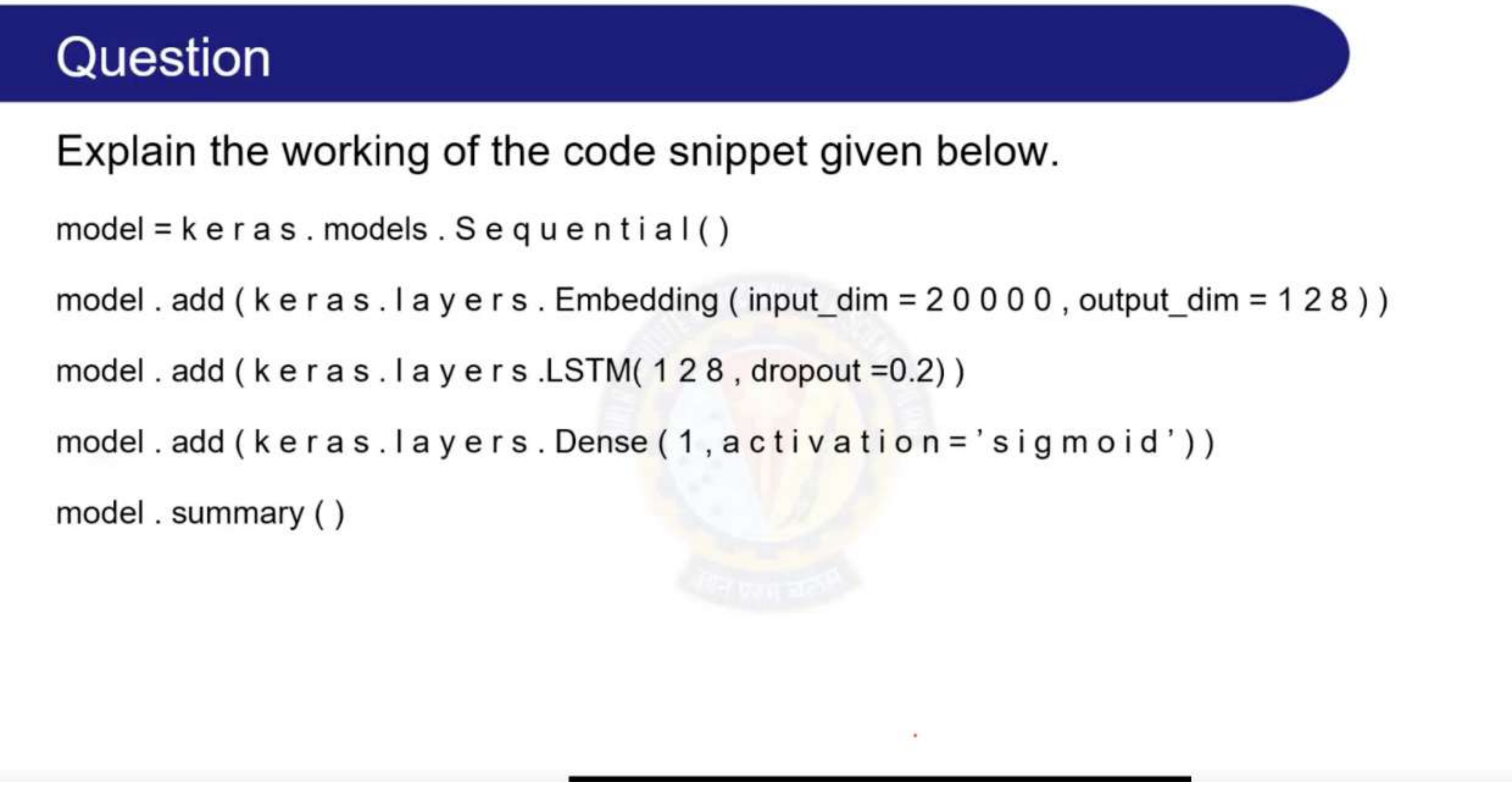
Question

For an image classification problem which classifies images into two categories, 128 by 128 pixel colour images is fed through four convolution layers with 32, 64, 128, 256 kernels of 5 by 5 respectively with max pooling for each layer. Then the tensors are fed through two layers of fully connected neurons with 1024 and 512 neurons.

(a) Draw a CNN for the above.

(b) Write Tensorflow-Keras code snippet for the above.

(c) Compute the number of parameters learned in each convolution and maxpooling layers.



Question

Explain the working of the code snippet given below.

```
model = keras . models . Sequential ( )

model . add ( keras . layers . Embedding ( input_dim = 20000 , output_dim = 128 ) )

model . add ( keras . layers .LSTM( 128 , dropout =0.2) )

model . add ( keras . layers . Dense ( 1 , activation = 'sigmoid' ) )

model . summary ( )
```

Question

Consider a RNN where the hidden state equation is h(t+1) = W h(t) + x (t+1) . Calculate and plot the output y(t) of a single hidden node and single input/output RNN, where activation function used in hidden and output nodes are, respectively, linear and sigmoid. Assume that the initial hidden state is 0, and the input sequence is of even length. Assume also that all biases are 0. Weight from input to hidden node is 1 and hidden to output node is 10. The recurrent weight in the hidden node is -1. Input is 1 if t=0 or even, and 0 for odd t.

$h_0 = h_{ini} + x_0 = 0 + 1 = 1$

$y_0 = sig(W_{out} * h_0) = sig(10 * 1) = sig(10) = 1$

$h_1 = -h_0 + x_1 = -1 + 0 = -1$

$y_1 = sig(10 * -1) = sig(-10) = 0$

$h_2 = -h_1 + x_2 = -1(-1) + 1 = \boxed{2}$

y_2

$y_1 = sig(10 * -1) = 0$

$y_2 = (10 * 2) = 1$

$y_3 = (10 * -2) = 0$

h_3 y_3 h_0 y_0 h_1

$h_3 = -h_2 + x_3 = -2 + 0 = -2$

0 to 1

Answer

$h0 = h(\text{initial}) + x0 = 0 + 1 = 1$

$y(0) = \text{sig}(\text{wout} \ast h0) = \text{sig}(10 \ast 1) = \text{sig}(10) = 1$

$h1 = -h0 + x1 = -1 + 0 = -1$

$y(1) = \text{sig}(\text{wout} \ast h0) = \text{sig}(10 \ast -1) = \text{sig}(-10) = 0$

$h2 = -h1 + x2 = -1(-1) + 1 = 2$

$y(1) = \text{sig}(\text{wout} \ast h0) = \text{sig}(10 \ast 2) = \text{sig}(20) = 1$

$h3 = -h2 + x3 = -2 + 0 = 2 = -2$

$y(1) = \text{sig}(\text{wout} \ast h0) = \text{sig}(10 \ast -2) = \text{sig}(-20) = 0$

The value of y will be close to 1 when t is even and close to 0 when t is odd. The value will oscillate from 1 to 0 and 0 to 1 for each time stamp starting from t=0. The graph will oscillate between 0 and 1.