**Birla Institute of Technology & Science, Pilani**
**Work Integrated Learning Programmes Division**
**Second Semester 2019-20**
**M.Tech. (Data Science and Engineering)**
**Comprehensive Examination (Makeup)**

# SOLUTION

| | |
|---|---|
| Course No. | : DSECLZG524 |
| Course Title | : DEEP LEARNING |
| Nature of Exam | : Open Book |
| Weightage | : 50% |
| Duration | : 2 Hours |
| Date of Exam: | July 19, 2020 Time of Exam: 2:00 PM – 4:00 PM |

No. of Pages = 7
*No. of Questions = 6*

Note: Assumptions made if any, should be stated clearly at the beginning of your answer. Show your rough work to get partial credit, when appropriate.

**Question 1. [3+1+2+2+0.5+0.5+1=10 marks]**

A. Refer to the following code snippet of a CNN implementation using Keras and answer below questions:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
classifier = Sequential()
classifier.add(Conv2D(filters =16,
                      kernel_size = (4, 4),
                      strides=(2, 2),
                      input_shape = (32, 32, 3),
                      activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (3, 3), strides=(2, 2)))
classifier.add(Conv2D(filters =10,
                      kernel_size = (3, 3),
                      padding = 'same',
                      activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (3, 3),
                            strides=(1,1)))
classifier.add(Flatten())
classifier.add(Dense(units = 64, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

a) Calculate the dimension of the input matrix at each step of the network

CNN L1 – 3 matrices of shape 32x32

CNN L2 – 48 matrices of shape 15x15 {3*16 matrices of dimension 32-4/2 + 1}

CNN L3 – 48 matrices of shape 7x7 {15-3/2 + 1}

CNN L4 – 480 matrices of shape 7x7 {48*10 matrices; same dimension due to padding}

CNN L5 – 480 matrices of shape 5x5 {7-3/1 + 1}

FC Input layer – Vector of shape 12000 {flattening – 480*5*5}

FC Hidden layer – Vector of shape 64 {as per architecture}

FC Output layer – scalar of size 1

b) What would be the difference in the dimension of the input vector of the fully connected layer had we not performed convolution steps in this architecture?
Answer – input vector would be of shape 32*32*3 = 3072 vs 12000

c) Above code uses “adam” optimizer. What benefit does this optimization algorithm have over traditional stochastic gradient descent?
Answer –The biggest drawback of SGD is that it uses a fixed learning rate for the entire training whereas in adam the learning rate is adapted based on the running estimate of the recent magnitudes of the gradients (both first and second moments of the gradient).

d) You want to train a picture classifier that identifies your face on the above architecture and therefore you source pictures of yourself from various sources. What is the key data pre-processing step that you will need for the above code to work on your dataset and why? What are the disadvantages of that pre-processing step?
Answer – Changing the size of the pictures to 32*32*3 as that is the input dimension that the architecture expects. This amounts to image compression and will likely cause loss of information.

B. Which one of the following is not a pre-processing technique?
i. Stemming and Lemmatization
ii. Converting to lowercase
iii. Removing punctuations and stop words
iv. Parts of speech tagging
Answer: iv.

C. Refer to the following code snippet.

```
>>> from nltk.translate.bleu_score import corpus_bleu
>>> corpus_bleu(actual, predicted, weights=(0.25, 0.25,
0.25, 0.25)
```

a) In the above code, what does the parameter “weights” signify?
i. Weights for unigrams only
ii. Weights for bigrams only
iii. Weights for 4-grams only
iv. None of the above
Answer: iv. (weights for unigrams, bigrams, trigrams and so on)

```
>>> from nltk.translate.bleu_score import corpus_bleu
>>> references = [[['this', 'is', 'not' 'a', 'test'], ['this',
'is', 'a', 'test']]]
>>> hypothesis = [['this', 'is', 'a', 'test']]
>>> score = corpus_bleu(references, hypothesis, weights=(0.3,
0.3, 0.3, 0))
>>> print(score)
```

b) What will be the output of the above code snippet?

i. 0

ii. 0.75

iii. 0.83

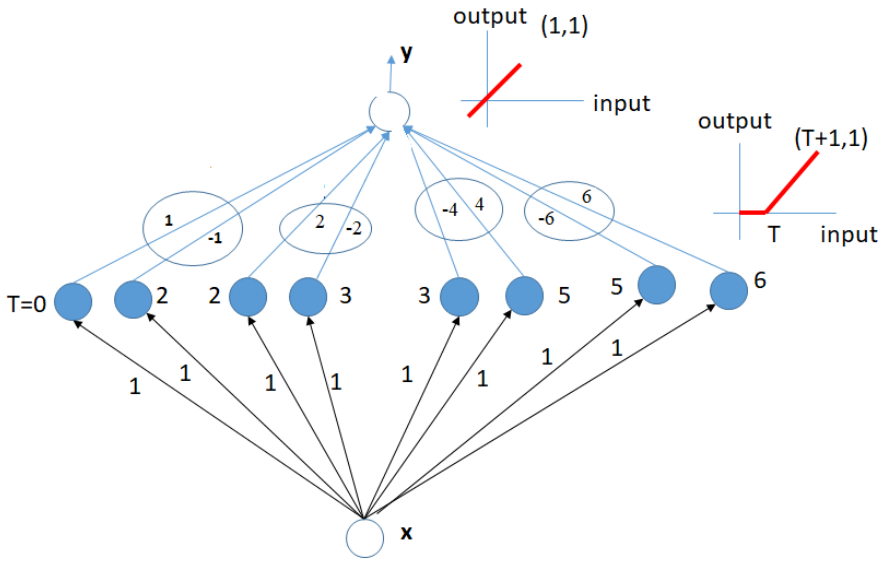
iv. 1.0

Answer: iv. (perfect score of 1.0 as the hypothesis matches one of the references exactly.)

**Question 2.** [**0.5+1+1+1.5+3+1=8 Marks**]

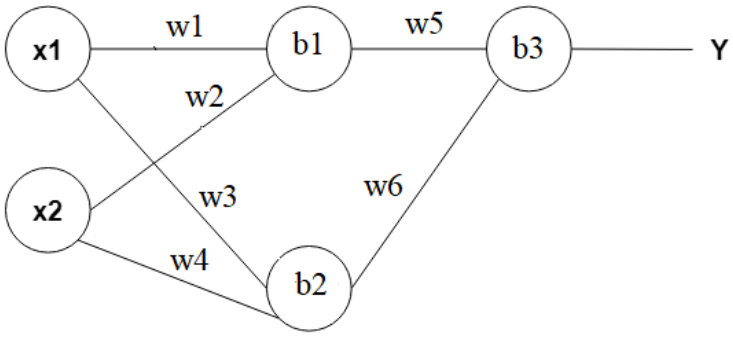
A. The hidden nodes in the network below uses ReLU activation functions with bias T, i.e., output = input-T if input >=T, else output is 0. The output node uses a linear activation function, i.e., output = input. For the specified weight configuration, find the output for

a. x=1.0
b. x=2.5
c. x=4
d. x=8




y(x=1.0)=1.0 (only the leftmost hidden node gives output 1, rest hidden node outputs are 0)
y(x=2.5)=2.5*1-1*(2.5-2)+2*(2.5-2) =3 (leftmost 3 hidden nodes output>0)
y(x=4)=4*1-1*(4-2)+2*(4-2)-2*(4-3)-4*(4-3) = 0 (leftmost 5 hidden nodes output>0)
y(x=8)=8*1-1*(8-2)+2*(8-2)-2*(8-3)-4*(8-3)+4*(8-5)-6*(8-5)+6*(8-6)=-10

B. The following network outputs a Y='1' when number of 1's in input binary string (x1,x2) of length 2 is either zero or even. For odd number of 1's in the input string, output is 0. Specify the weights w1, w2, w3, w4, w5, w6 and bias b1, b2, b3 in the following network. **The weights w's can be only +1 or -1 or 0**. The nodes use step threshold, i.e., output =1 if input >= bias, else 0.

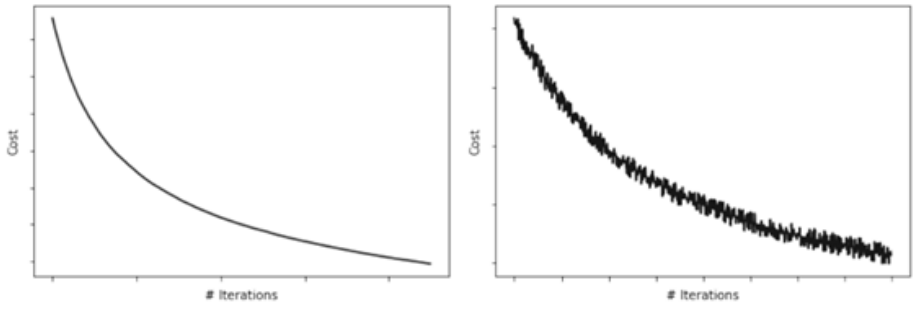



Output Y = (x1' AND x2') OR (x1 AND x2). The hidden nodes implement the AND function. Choose w1=w2=-1 and w3=w4=1. So, correspondingly, choose b1=-0.5 and b2 = 1.5. (in fact 1< b1 <=0 and 1< b2 <= 2)
Choose w5=w6=1 and correspondingly, b3=0.5 (actually 0< b3 <=1)

**Question 3. [**1+1+1+1+1+2+2+1=**10 Marks]**

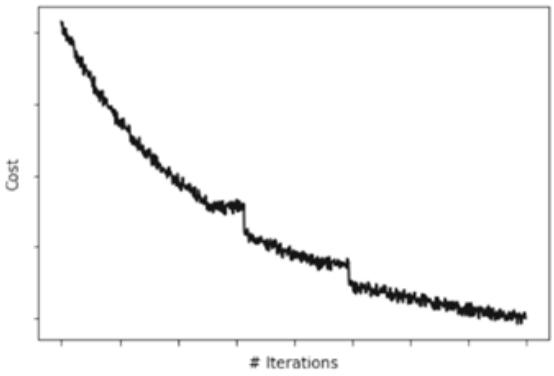
A. What is a saddle point? What is the advantage/disadvantage of using Stochastic Gradient Descent in dealing with saddle points?
Saddle point - The gradient is zero, but it is neither a local minima nor a local maxima. Also accepted - the gradient is zero and the function has a local maximum in one direction, but a local minimum in another direction. SGD has noisier updates and can help escape from a saddle point.

B. Figure below shows how the cost decreases (as the number of iterations increases) when two different optimization algorithms are used for training. Which of the graphs corresponds to using batch gradient descent as the optimization algorithm and which one corresponds to using mini-batch gradient descent? Explain.
Batch gradient descent - (a), Minibatch – (b). Batch gradient descent - the cost goes down at every single iteration (smooth curve). Mini-batch - does not decrease at every iteration since we are just training on a mini-batch (noisier)




(a) (b)

C. Figure below shows how the cost decreases (as the number of iterations increases) during training. What could have caused the sudden drop in the cost? Explain.




Learning rate decay

D. Consider an error function $E(w_1, w_2) = 0.05 + \frac{(w_1-3)^2}{4} + \frac{(w_2-4)^2}{9} - \frac{(w_1-3)(w_2-4)}{6}$. Different variants of gradient descent algorithm can be used to minimize this error function w.r.t. $w_1$, $w_2$. Assume $(w_1, w_2) = (1,1)$ at time *(t-1)* and after update $(w_1, w_2) = (1.5, 2.0)$ at time *t*. Assume, learning rate $\eta = 0.3$ and momentum update rate $\beta = 0.9$.

a. What is the value of $(w_1, w_2)$ that minimizes this error function? What is the minimum possible value of E?
   w1=3, w2=4. $E_{min}$ = 0.05
b. What is the value of $(w_1, w_2)$ at time (t+1) if standard gradient descent is used?
   δE/δw1=0.5*(w1-3)-(w2-4)/6 and δE/δw2=2/9*(w2-4)-(w1-3)/6
   So, w1(t+1)=1.5-0.3*0.5(1.5-3)+0.3*(2-4)/6 = 1.625, and
   w2(t+1)=2.0-0.3*2*(2-4)/9+0.3(1.5-3)/6=2.058
c. What is the value of $(w_1, w_2)$ at time (t+1) if momentum based gradient descent is used?
   w1(t+1)=1.625+(1.5-1.0)*0.9=2.075
   w2(t+1)=2.88+(2-1)*0.9=3.78

d. What is the value of $(w_1, w_2)$ at time (t+1) if Nestorov's accelerated gradient is used?
   $w1_{new}$ =1.5+0.9*(1.5-1)=1.95 $w2_{new}$ = 2.0+0.9*(2-1)=2.9
   At ($w1_{new}$=1.95, $w2_{new}$=2.9) δE/δw1 = 0.5*(1.95-3)-(2.9-4)/6 = -0.342 and
   δE/δw2=2/9(2.9-4)-(1.95-3)/6=-0.0694
   w1(t+1)=1.5+0.3*0.342=1.603 w2(t+1)=2.0+0.3*0.0694=2.021

e. Which among the above techniques results in highest reduction in E in this example?
   E(1.625,2.058)=0.497
   E(2.075,3.78)=0.235
   E(1.603,2.021)=0.512

   So, momentum based update gives the best reduction in E.

**Question 4. [1+3+2+2 = 8 Marks]**
**Answer A. OR B. Answer C.**

A. Consider a 2x2 convolution (as per Deep Learning terminology) operator O

| | |
|---|---|
| -1 | -1 |
| 1 | 1 |

Output is generated by stacking two such operators on the 2-D input data (i.e., the operator is applied twice on the input).

a) What is the size of the equivalent operator V that will give the same output, when applied only once on the input data?
3x3

b) Show the elements $V_{ij}$ of equivalent operator V.

$$\begin{pmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{pmatrix}$$

B. a) Assuming stride = 1 and padding =0 what is the size of transposed convolution of the following 2x2 kernel O

| | |
|---|---|
| 0 | 2 |
| 1 | 3 |

with the following input I?

| | |
|---|---|
| 0 | 2 |
| 1 | 3 |

b) Show the result of the transposed convolution.

Size is 3x3.

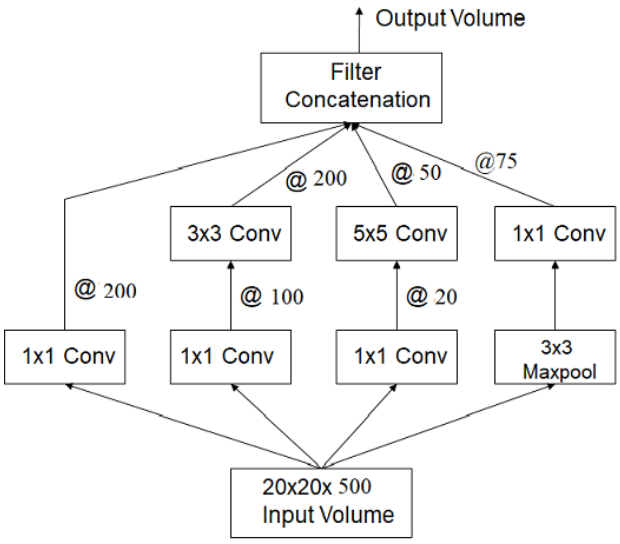
$$\text{Output} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 4 \\ 0 & 2 & 6 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 3 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 6 \\ 0 & 3 & 9 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 4 \\ 0 & 4 & 12 \\ 1 & 6 & 9 \end{pmatrix}$$

C. Consider the following configuration of a sample Google Lenet inception block. Appropriate padding is assumed. The output depth of the 1x1 convolution blocks are, respectively, 200, 100, 20 and 75, as shown in the figure.

a) What is the size of the output volume? What is the size of padding used during 1x1, 3x3 and 5x5 convolution operations?

Output volume size = 20x20x525
Padding size = 0, 1, 2

b) How many parameters are there in this block?
500x200+500x100+500x20+9x100x200+25x20x50+500x75
=500x320+180000+25000+500x75
=402500

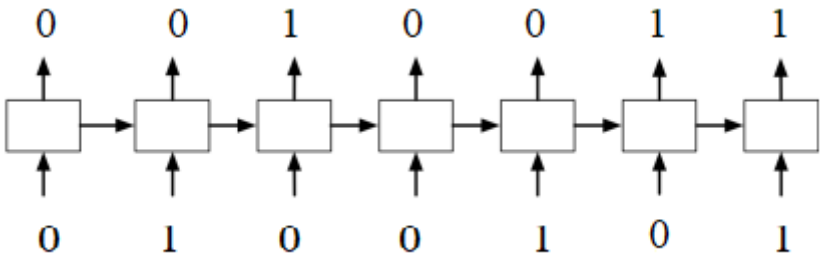



**Question 5. [1+4+5 = 10 marks]**

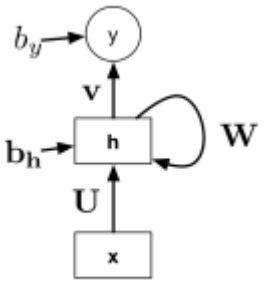
Design a recurrent neural network that outputs a parity bit for binary sequences of arbitrary length. The inputs are given as binary sequences from right to left and output of 1 is generated when number of '0's in the string seen so far is even. For instance, the input string 1010010 would generate an output as follows:

- Input: 0,1,0,0,1,0,1
- Correct output: 0,0,1,0,0,1,1

There is one input unit corresponding to the input bit, and one output unit. Therefore, the pattern of inputs and outputs for this example would be:




The RNN has one input unit **x**, two hidden units **h**, and one output unit y. All of the units use the hard threshold activation function, i.e., output is 1 if total weighted input is >=bias, else 0.




Note, at time t, $\mathbf{h}_t$ = step($\mathbf{W}\mathbf{h}_{t-1}$ + **U x** - $\mathbf{b}_h$) and $y_t$ = step($\mathbf{v}\mathbf{h}_t$ - $b_y$)

A. Specify the state representations for the underlying finite state machine in terms of outputs of 2 hidden nodes.
Choose [h1,h2]=[0,0] and [h1,h2]=[1,1] corresponding to y=0, and
[h1,h2]=[1,0] for y=1

B. For the chosen representation, express the current hidden states $\mathbf{h}_t$ as Boolean functions of past hidden states $\mathbf{h}_{t-1}$ and current input **x**. Express the current output $y_t$ as a Boolean function of current hidden states $\mathbf{h}_t$.
y(t) = h1(t) h2'(t)
h1(t) = h1(t-1)+h2'(t-1)+x'
h2(t) = h1(t-1)h2'(t-1)x'

C. For the obtained Boolean functions, specify weight matrices **U** (size 2x1), **v** (size 1x2), and **W** (size 2x2), bias vector $\mathbf{b}_h$ (2x1) and scalar bias $b_y$. Elements of **U**, **v** and **W** can be only -1, 1 or 0.

**v** = [1 -1] and $b_y$ = 0.5

Thus,

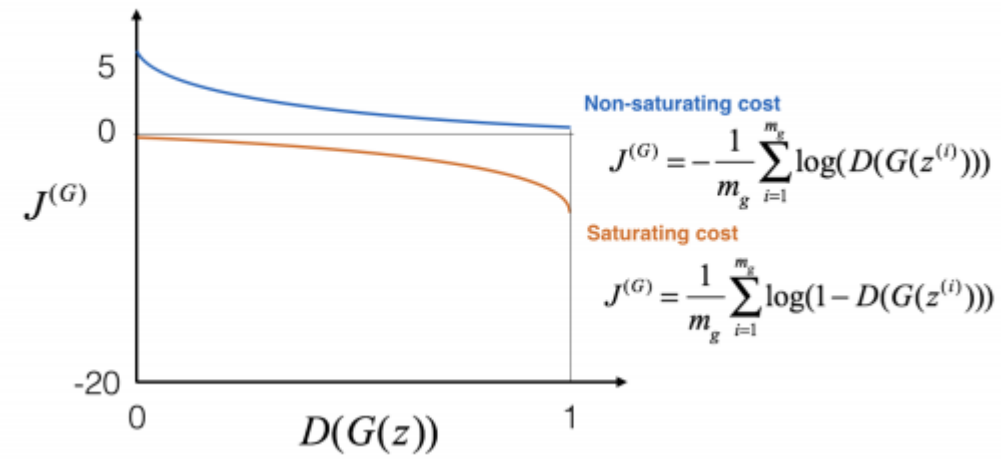
$$\mathbf{W}=\begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}$$

$$\mathbf{U}=\begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

$$b_h^T = [-0.5 \quad 0.5]$$

**Question 6. Answer the following questions [1+1+1+1 = 4 Marks]**

Consider the following graph representing the training procedure of a generative adversarial network. Cost function of the generator is plotted against the output of the discriminator, given a generated image G(z). The discriminator's output is 0 implies that the discriminator thinks the input has been generated by G (1 if it thinks the input is actually real data).




A. Early in the training, is the value of D(G(z)) closer to 0 or closer to 1? Yes/No. Why?

The value of D(G(z)) is closer to 0 because early in the training, D is much better than G. One reason is that G's task (generating images that look like real data) is a lot harder to learn than D's task (distinguishing fake images from real images).

B. Two cost functions are presented in the above figure, which one would you choose to train your network? Why?

Choose the "non-saturating cost" because it leads to much higher gradients early in the training and thus helps the generator learn quicker.

C. What types of convolution are used in the pooling layers of CNN based GANs?

Strided convolutions in discriminator and fractionally strided in generator.

D. Why is KL divergence used for regularization in variational autoencoders?

To ensure the latent feature space is regular so that points that are close in latent space are also close in original input space after decoding. This preserves the semantics during encoding process.