## Problem 4 Convolutional Neural Networks and Receptive Field (12 credits)

A friend of yours asked for a quick review of convolutional neural networks. As he has some background in computer graphics, you start by explaining previous uses of convolutional layers.

a) You are given a two dimensional input (e.g., a grayscale image). Consider the following convolutional kernels

$$C_1 = \frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \qquad C_2 = \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}.$$

0 1 2

What are the effects of the filter kernels $C_1$ and $C_2$ when applied to the image?

$C_1$: Local/box (0.5p) blur/smoothing (0.5p) kernel. Note: If only mean/arg is mentioned instead of blur then 0.5p
$C_2$: vertical (0.5p) edge detector (0.5p)

After showing him some results of a trained network, he immediately wants to use them and starts building a model in Pytorch. However, he is unsure about the layer sizes so you quickly help him out.

b) Given a Convolution Layer in a network with 5 filters, filter size of 7, a stride of 3, and a padding of 1. For an input feature map of $26 \times 26 \times 26$, what is the output dimensionality after applying the Convolution Layer to the input?

0 1 2

$8 \times 8 \times 5$ (2p) 1p for only kernel size computation $\frac{26-7+2\cdot 1}{3} + 1 = 7 + 1 = 8$

c) You are given a convolutional layer with 4 filters, kernel size 5, stride 1, and no padding that operates on an RGB image.

0 1 2

1. What is the shape of its weight tensor?
2. Name all dimensions of your weight tensor.

Shape: (3, 4, 5, 5) (1p)
Reasoning: input size/rgb channels, output size/channels, width, height (1p)

Note: -1p if only 3 dimensions are mentioned, -1p if 5's are simply described as filter size

Now that he knows how to combine convolutional layers, he wonders how deep his network should be. After some thinking, you illustrate the concept of receptive field to him by these two examples. For the following two questions, consider a grayscale 224x224 image as network input.

d) A convolutional neural network consists of 3 consecutive $3 \times 3$ convolutional layers with stride 1 and no padding. How large is the receptive field of a feature in the last layer of this network?

0 1

$1x1 \rightarrow 3x3 \rightarrow 5x5 \rightarrow 7x7$ (1p)
Note: -0.5p if no tuple

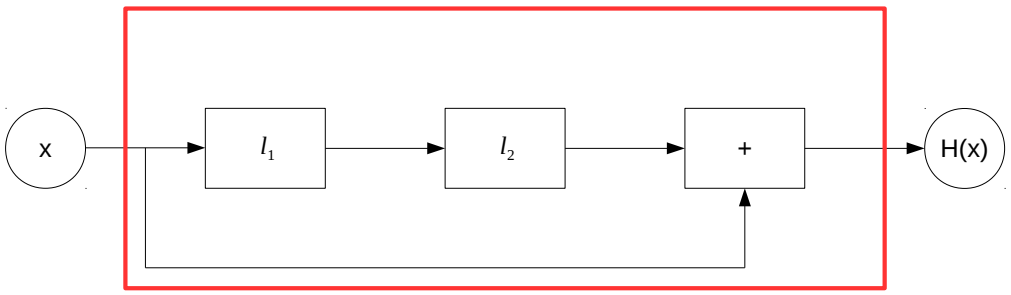
0 1 2

e) Consider a network consisting of a single layer.

1. What layer choice has a receptive field of 1?
2. What layer has a receptive field of the full image input?

1x1 convolution or identity (1p)
fully connected or conv/pooling layer with kernel size equals full input size (224x224) (1p)
If the answer is reasonable but incomplete: -0.5p

Blindly, he stacks 10 convolutional layers together to solve his task. However, the gradients seem to vanish and he can't seem to be able to train the network. You remember from your lecture that ResNet blocks were designed for these purposes.




0 1 2

f) Draw a ResNet block in the image above (1p) containing two linear layers, which you can represent by $l_1$ and $l_2$. For simplicity, you don't need to draw any non-linearities. Why does such a block improve the vanishing gradient problem in deep neural networks (1p)?

1p for correct drawing
Note: if image strucutre is correct but: i) arrow is missing or ii) "+" symbol is missing or not drawn correctly 0.5p
1p for highway of gradients
Note: if only forward pass is mentioned then no points

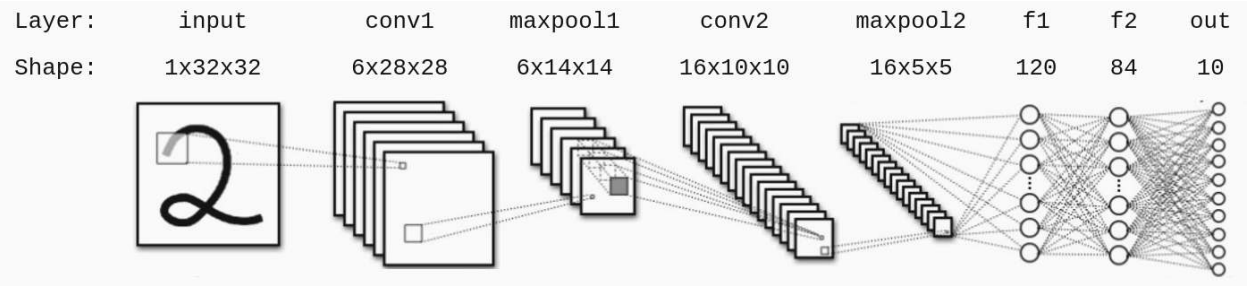
0 1

g) For your above drawing, given the partial derivative of the residual block $R(x) = l_2(l_1(x))$ as $\frac{\partial R(x)}{\partial x} = r$, calculate $\frac{\partial H(x)}{\partial x}$.

$\frac{\partial H(x)}{\partial x} = \frac{\partial x+R(x)}{\partial x} = \frac{\partial x}{\partial x} + \frac{\partial R(x)}{\partial x} = 1 + r$
1p for $\frac{\partial H(x)}{\partial x} = 1 + r$

## Problem 7 Convolutional Neural Networks (12 credits)

You are contemplating design choices for a convolutional neural network for the classification of digits. LeCun et. al suggest the following network architecture:




For clarification: the shape **after** having applied the operation 'conv1' (the first convolutional layer in the network) is 6x28x28.
All operations are done with stride 1 and no padding. For the convolution layers, assume a kernel size of 5x5.

0 1 2 3

a) Explain the term 'receptive field' (1p). What is the receptive field of one pixel after the operation 'maxpool1'(1p)? What is the receptive field of a neuron in layer 'f1' (1p)?

Receptive field is the size of the region in the input space that a pixel in the output space is affected by.
maxpool1: 6x6. One pixel after maxpool1 is affected by 4 pixels (2x2) in conv1. with 5x5 kernel and stride 1, a 2x2 output comes from a 6x6 grid.
(0.5p if only maxpool1 wrt to conv1(= 2x2) specified)
f1: whole image (32x32)

0 1

b) Instead of digits, you now want to be able to classify handwritten alphabetic characters (26 characters). What is the **minimal** change in network architecture needed in order to support this?

Change no. of output neurons from 10 to 26
(0.5p if only "change number of output neurons" specified)

0 1 2

c) Instead of taking $32 \times 32$ images, you now want to train the network to classify images of size $68 \times 68$. List two possible architecture changes to support this?

- Resize layer to downsample images to 32x32 / Downsample images to 32x32 (preprocess)
- conv 5x5 ($68 \rightarrow 64$) + maxpool 2x2 ($64 \rightarrow 32$) before the current architecture (0.5p if only specified conv+maxpool without parameters)
- Change input dimension of layer f1 to 16x14x14 (= 3136) (0.5p if only suggested changing input dimension layer without new dim)
- fully convolutional layers + global average pooling (0.5p if only fully conv layer suggested without global average pooling)

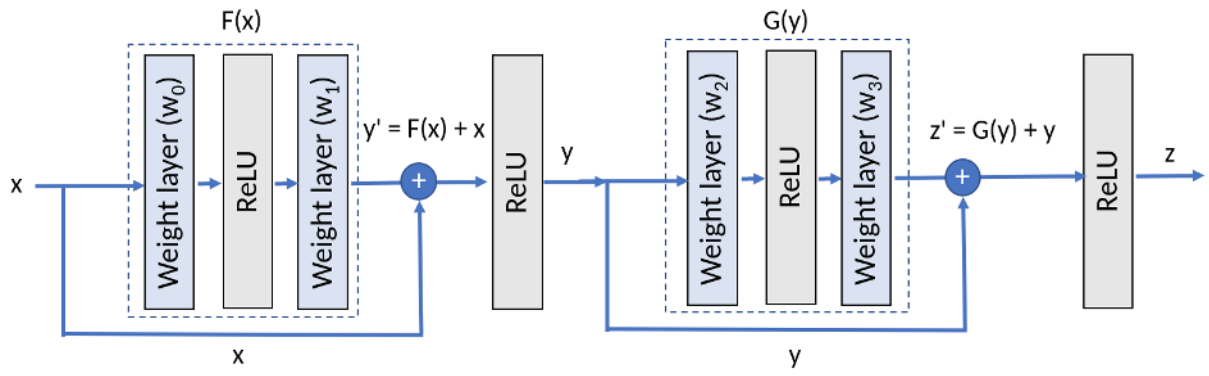
d) Your architecture works and you manage to classify characters fairly well. After reading many online blogs, you decide to try out a much deeper network to boost the network's capacity. Name 2 problems that you might encounter when training very deep networks.

0 1 2

- Vanishing gradients
- Memory issues
- Overfitting
- Increased training time

e) You read that skip connections are beneficial for training deep networks. In the following image you can see a segment of a very deep architecture that uses skip connections. How are skip connections helpful? (1p). Demonstrate this mathematically by computing gradient of output $z$ with respect to '$w_0$' for the network below in comparison to the case without a skip connection (3p). For simplicity, you can assume that gradient of ReLU, $\frac{d(ReLU(p))}{dp} = 1$.

0 1 2 3 4



Help prevent vanishing gradients / Provides highway for the gradients in backward pass (1p)

Let,

$$z' = G(y) + y$$
$$G(y) = ReLU(w_2 y) w_3$$
$$z = ReLU(z')$$
$$y' = F(x) + x$$
$$F(x) = w_1 ReLU(w_0 x)$$
$$y = ReLU(y')$$
$$\frac{dz}{dw_0} = \frac{dz}{dz'}\frac{dz'}{dy}\frac{dy}{dy'}\frac{dy'}{dw_0}$$
$$\frac{dz}{dw_0} = \frac{d(ReLU(z'))}{dz'}(\frac{dG(y)}{dy} + 1)\frac{d(ReLU(y'))}{dy'}\frac{dF(x)}{dw_0}$$
$$\frac{dz}{dw_0} = \frac{d(ReLU(z'))}{dz'}(w_3 w_2 \frac{d(ReLU(w_2 y))}{d(w_2 y)} + 1)\frac{d(ReLU(y'))}{dy'} w_1 x \frac{d(ReLU(w_0 x))}{d(w_0 x)})$$

Putting ReLU derivatives to 1

$\frac{dz}{dw_0} = (w_3 w_2 + 1) w_1 x$ (2 points for full expansion, 1pt if $\frac{dG(y)}{dy}$ and $\frac{dF(x)}{dw_0}$ are not expanded)

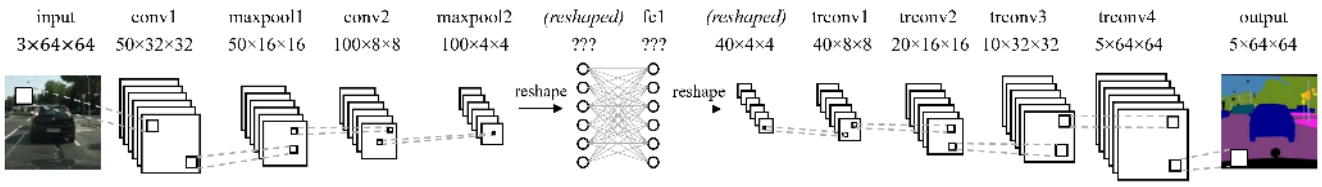
Comparing this to derivative without skip connection, which is

$$\frac{dz}{dw_0} = (w_3 w_2) w_1 x \text{ (1 point / 0.5p if not expanded)}$$

The extra '+1' term in the skip connection derivative help propagation of gradient flow, preventing vanishing gradients

# Problem 3 Convolutions (13 credits)

You are asked to perform **per-pixel** semantic segmentation on the Cityscapes dataset, which consists of RGB images of European city streets, and you want to segment the images into 5 classes (vehicle, road, sky, nature, other). You have designed the following network, as seen in the illustration below:
For clarification of notation: The shape **after** having applied the operation 'conv1' (the first convolutional layer in the network) is 50x32x32.
You are using 2D convolutions with: `stride` = 2 , `padding` = 1 , and `kernel_size` = 4 .
For the MaxPool operation, you are using: `stride` = 2 , `padding` = 0 , and `kernel_size` = 2 .




3.1 What is the shape of the weight matrix of the fully-connected layer 'fc1'? (Ignore the bias)

0 1 2

input: $100 \times 4 \times 4 = 1600$
output: $40 \times 4 \times 4 = 640$
weight matrix: $1600 \times 640$
(-0.5 if final output is not calculated, -1 if only input OR output is correct and explained, -1 if only input or output size is correct and not explained)

3.2 Explain the term 'receptive field' (1p). What is the receptive field of one pixel of the activation map. after performing the operation 'maxpool1'(1p)? What is the receptive field of a single neuron in the output of layer 'fc1' (1p)?

0 1 2 3

the region in the input space (0.5p) that a pixel in the output space is affected by / that affects a particular unit in the network (0.5). alternative: spatial extent of the connectivity of a convolutional filter (1p)
maxpool1: 6x6. One pixel after maxpool1 is affected by 4 pixels (2x2) in conv1. with 4x4 kernel and stride 2, a 2x2 output comes from a 6x6 grid. (1p)
fc1: whole image (64x64) (1p)

Receptive field is the total area of the image encoded in a single pixel after a convolution operation
Receptive field of one pixel of the activation map: It's the area from which each pixel has gotten its information from after applying the previous convolutions it's the same as the receptive field of its corresponding pixel

0 1

3.3 You now want to be able to classify finer-grained labels, which comprise of 30 classes. What is the **minimal** change in network architecture needed in order to support this without adding any additional layers?

1in - in trconv4/the last layer(0.5p) use 30 channels(0.5p) instead of 5 (trying to add or remove any layer or changing something in the fc layer 0 points as those are not the minimal change )

Use the weights as they are but change and retrain the fully connected layer to classify the 30 classes instead of just 5

0 1 2

3.4 Luckily, you found a pre-trained version of this network, which is trained on the original 5 labels. (It outputs a tensor of shape $5 \times 64 \times 64$ ). How can you make use of/build upon this pre-trained network (as a black-box) to perform segmentation into 30 classes.

- **add** conv or other layer at the end (with 30 output channels, that preserves size, e.g. 1x1) (1p) **after the pretrained network** to predict the class
Black-box: any attempts to change the pretrained network (e.g to only use part of the network) is incorrect (0p)

Variational auto encoder. By training the network to generate new photos of the same type it will also learn the features of these images

0 1 2 3

3.5 Luckily, you have gained access to a large dataset of city street images. Unfortunately, these images are not labelled, and you do not have the resources to annotate them. However, how can you still make use of these images to improve your network? Explain the architecture of any networks that you will use and explain how training will be performed. (Note: This question is independent of (3.3) and (3.4))

1. Choose the right architecture: autoencoder (or other methods e.g. U-net, VAE)(1p)
[Only answer transfer learning 0.5p, as the question asked about a specific architecture, not type of training method]
2. Training: Train it on this dataset for reconstructing the input image in an unsupervised way (1p).
3. Usage: Then use the trained encoder part to extract features for your network (1p).
-0.5 if feature extraction is not mentioned anywhere in the answer
[GAN is only accepted (1p) If with reasonable explanation for "how"] [Using your network/GAN/KNN to predict the label of the new image is not accepted for this subproblem.]
[Approaches which does not include a network architecture, e.g.K Means: 1p]

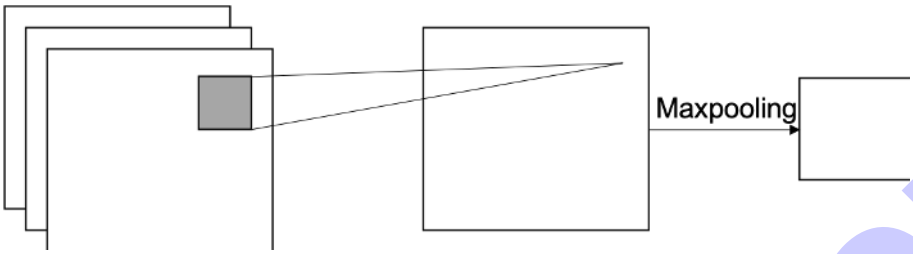
0 1 2

3.6 Instead of taking $64 \times 64$ images as input, you now want to be able to train the network to segment images of arbitrary size $> 64$. List, explicitly, two different approaches that would allow this. Your new network should support varying image sizes in run-time, without having to be re-trained. making it fully convolutional using 1x1 conv layers.

- fully convolutional: 1x1 conv or removing fc1, using FCN, U-Net (1p)
- preprocess: resize, crop, pre-process all the input images to 64x64 (1p)
- ADAPTIVE pooling (Average, Max etc) with fixed output size before the fc1 (1p)
- pooling with fixed output size and does not mention "ADAPTIVE" explicitly (0.5p)
- if approaches too similar (e.g. 1x1 conv + FCN) (1p in total)
- not the intended answer but ok: Graph Neural Network, RNN or LSTM, Transformer
- No: add conv in the beginning, cannot produce fixed size output for variable image size

## Problem 5 Backpropagation and Convolutional Layers (12 credits)

Your friend is excited to try out those "Convolutional Layers" you were talking about from your lecture. However, he seems to have some issues and requests your help for some theoretical computations on a toy example.
Consider a neural network with a convolutional (without activation) and a max pooling layer. The convolutional layer has a single filter with kernel size $(1, 1)$, no bias, a stride of 1 and no padding. The filter weights are all initialized to a value of 1. The max pooling layer has a kernel size of $(2, 2)$ with stride 2, and 1 zero-padding.




You are given the following input image of dimensions $(3, 2, 2)$:

$$x = \left( \begin{bmatrix} 1 & -0.5 \\ 2 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 \\ -1.5 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right)$$

0 1 2 a) Compute the forward pass of this input and write down your calculations.

Forward pass

$$\begin{bmatrix} 1 & -0.5 \\ 2 & -2 \end{bmatrix} + \begin{bmatrix} -2 & 1 \\ -1.5 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 \\ 0.5 & -1 \end{bmatrix} \quad (1p)$$

After max pooling,

$$\begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \quad (1p)$$

0 1 b) Consider the corresponding ground truth,

$$y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Calculate the binary cross-entropy with respect to the natural logarithm by summing over all output pixels of the forward pass computed in (a). You may assume $\log(0) \approx -10^9$. (Write down the **equation** and keep the logarithm for the final result.)

$$BCEloss = -\sum_i t_i \log s_i \qquad (0.5p \text{ for either his or the line below})$$
$$= -\log(2w_1 - 1.5w_2) - \log(-0.5w_1 + w_2)$$
$$= -\log(0.5) - \log(0.5) = 2\log 2 \qquad (1p)$$

0 ½ c) You don't recall learning the formula for backpropagation through convolutional layers but those $1 \times 1$ convolutions seem suspicious. Write down the name of a common layer that is able to produce the same result as the convolutional layer used above.

Fully-connected layer

d) Update the kernel weights accordingly by using gradient descent with a learning rate of 1. (Write down your calculations!)

0 1 2 3 4 5

Partial derivatives for w1/w2 (2p),

$$\frac{\partial BCE}{\partial w_1} = -\frac{\partial \ln(2w_1 - 1.5w_2) + \ln(-0.5w_1 + w_2)}{\partial w_1} = -\frac{2}{2w_1 - 1.5w_2} - \frac{-0.5}{-0.5w_1 + w_2} = -4 + 1 = -3$$

$$\frac{\partial BCE}{\partial w_2} = -\frac{\partial \ln(2w_1 - 1.5w_2) + \ln(-0.5w_1 + w_2)}{\partial w_2} = -\frac{-1.5}{2w_1 - 1.5w_2} - \frac{1}{-0.5w_1 + w_2} = 3 - 2 = 1$$

Update using gradient descent for w1/w2 (2p),

$$w_1^+ = w_1 - lr * \frac{\partial BCE}{\partial w_1} = 1 - 1 \times (-3) = 4$$

$$w_2^+ = w_2 - lr * \frac{\partial BCE}{\partial w_2} = 1 - 1 \times 1 = 0$$

Derivate and update for w3 (1p total):

$$\frac{\partial BCE}{\partial w_3} = 0$$

$$w_3^+ = w_3 - 0 = 1$$

1p if the person only wrote at least the gradient descent update rule

0 1 2

e) After helping your friend debugging, you want to showcase the power of convolutional layers. Deduce what kind of $3 \times 3$ convolutional filter was used to generate the output (right) of the grayscale image (left) and write down its $3 \times 3$ values.

Vertical edge detector (1p)

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} (1p)$$

Flipping & Scaling are OK

0 1

f) He finally introduces you to his real problem. He wants to find $3 \times 3$ black crosses in grayscale images, i.e., each pixel has a value between 0 (black) and 1 (white).

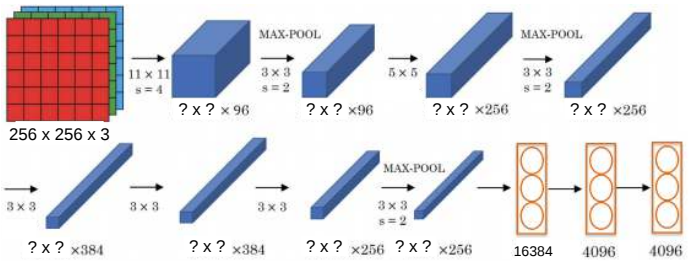
You notice that you can actually hand-craft such a filter. Write down the numerical values of a $3 \times 3$ filter that maximally highlights on the position of black crosses.

$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix} (2p)$$

Flipping & Scaling are OK, even though pixel values were given

## Problem 7 Convolutional Neural Networks (7.5 credits)

Your friend needs your expertise in classifying a set of RGB Images of size 256 x 256 pixels into a total of 1000 classes. Can you help him out?
Since you learned that CNNs are great to tackle these sort of tasks, you decide to start out with the following CNN architecture.




**Notes:**

- The values directly below the arrows indicate the filter sizes $f$ of the corresponding convolution and pooling operations.
- $s$ stands for stride. If no stride is specified, a stride of $s = 1$ is used.
- All convolutional and pooling layers use a padding of $p = \frac{f-1}{2}$ for a corresponding filter size of $f$.
- For each convolutional filter, we include a bias.

0 1 a) In the figure above, the output layers for classification are missing. Explain:

1. Which type of last layer you would use there and how would you choose it's dimension?
2. Which output function and loss function would you choose for this task?

1. Fully-connected layer (0.5 points) with 1000 neurons (0.5 points)
2. Softmax function combined with Cross-Entropy Loss (0.5 point)

0 1 b) Calculate the output dimension of the image after passing through the first convolutional layer. Make sure to include your calculations in the solution.

Formula to calculate the output dimension: $\dim_{out} = \frac{\dim_{in} - f + 2 \cdot p}{s} + 1$ Output Dimension $(\frac{256-11}{4} + 1) = 64 \times 64 \times 96$ (1 Point)

0 1 c) Calculate the number of learnable parameters in the first convolutional layer. Make sure to include your calculations in the solution. Un-multiplied answers are accepted (e.g., $3 * 3 * 16$)

Number of Parameters $(11 * 11 * 3 + 1) * 96 = 34944$ parameters (1 Point)

d) Calculate the size of the combined receptive field of the first two and of the first three layers. This corresponds resp. to the area of pixels in the input image that each neuron

0 1 2

1. after the second layer (right after the first MAX-POOL layer)
2. after the third layer (before the second MAX-POOL layer)

"sees".
**Hint:** Strides affect the total receptive field sizes of subsequent layers.

Size of total receptive field after $k > 1$: $r_k = r_{k-1} + \left(\prod_{i=1}^{k-1} s_i\right) \cdot (f_k - 1)$

- layer 1: $r_1 = 11$ ($11 \times 11$ filter)
- layer 2: $r_2 = 11 + (4 \cdot (3 - 1)) = 11 + 8 = 19$ (0.5p answer, 0.5p calculation)
- layer 3: $r_3 = 19 + (4 \cdot 2 \cdot (5 - 1)) = 19 + 8 \cdot 4 = 19 + 32 = 51$ (0.5p answer, 0.5p calculation)

Alternative solution for $r_3$ (from right to left, needs separate calculation for $r_2$!): $r_{i-j} = f_j + (r_{i-(j+1)} - 1) * s_j$)

- 3rd to 2nd layer: $r_{3-2} = 5$ ($5 \times 5$ filter)
- 3rd to 1st layer: $r_{3-1} = 3 + (5 - 1) \cdot 2 = 11$ ($3 \times 3$ filter, stride 2)
- 3rd layer to input: $r_3 = r_{3-0} = 11 + (11 - 1) \cdot 4 = 51$ ($11 \times 11$ filter, stride 4)

After a series of convolutional layers, the architecture has 3 fully connected layers that help process the spatial information obtained by the convolutions and prepare it for the output layer. Our friend just found an article about image segmentation and gets really excited about this. He decides to forget about the classification task and wants to work on image segmentation.

We don't need to reject our architecture: We first convert our architecture to a fully-convolutional network by ditching the fully connected layers. Next, we want to produce an output similar to the input size from this bottleneck where we would like to mirror the current architecture.

e) How would you replace the convolutional layers in the mirrored architecture to increase the the image size from our bottleneck onwards?

0 1

Accepted answers: upsampling/unpooling + convolution, transposed convolution.

Grading notes: upsampling alone (without conv) is 0.5p, "inverse convolution", "transformed convolution", and all other misspellings are 0.5p.

f) The new architecture is able to process an image of any input size. How about the original architecture that we started with, was it able to handle images of arbitrary sizes as input, too? Give an explanation for your answer.

0 1

No, the original architecture was not able to handle images of arbitrary size (0.5p).

Explanation: Fully-connected layers require fixed size and cannot handle variable output size of convolutional layers (0.5p).

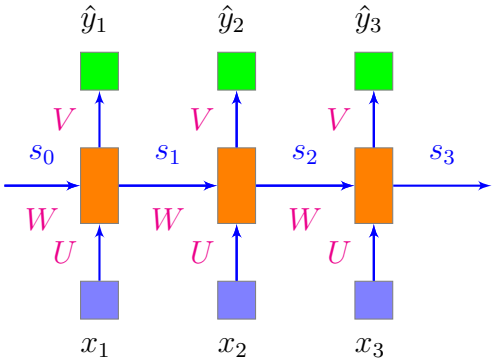
# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

---

## Sample RNN Question

Compute the outputs in each timestep and the state after timestep=3 for the Vanilla RNN given below. Assume the biases as zeros. [5]



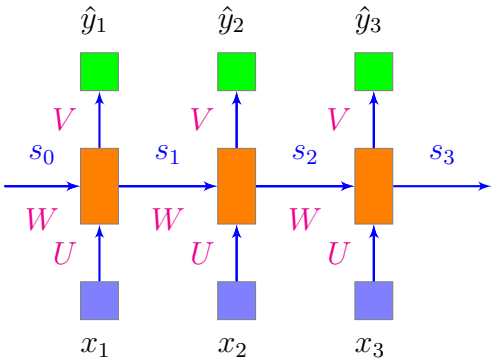

$$X = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^\top$$
$$W = \begin{bmatrix} 0.2 & 0.3 & 0.8 \end{bmatrix}^\top$$
$$U = \begin{bmatrix} 0.5 & 0.6 & 0.2 \end{bmatrix}^\top$$
$$V = \begin{bmatrix} 0.4 & 0.2 & 0.1 \end{bmatrix}^\top$$

Solution

$$X = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^\top$$
$$W = \begin{bmatrix} 0.2 & 0.3 & 0.8 \end{bmatrix}^\top$$
$$U = \begin{bmatrix} 0.5 & 0.6 & 0.2 \end{bmatrix}^\top$$
$$V = \begin{bmatrix} 0.4 & 0.2 & 0.1 \end{bmatrix}^\top$$
$$s_t = \sigma(Ux_t + Ws_{t-1} + b)$$
$$\hat{y}_t = Relu(Vs_t + c) \qquad \text{Relu is assumed}$$
$$s_0 = 0 \qquad b = 0 \qquad c = 0$$
$$s_1 = \sigma(0.5 * 1 + 0.2 * 0 + 0) = 0.6$$
$$\hat{y}_1 = \max(0, 0.4 * 0.6 + 0) = 0.24$$
$$s_2 = \sigma(0.6 * 1 + 0.3 * 0.6 + 0) = 0.68$$
$$\hat{y}_2 = \max(0, 0.2 * 0.68 + 0) = 0.136$$
$$s_3 = \sigma(0.2 * 0 + 0.8 * 0.68 + 0) = 0.63$$
$$\hat{y}_3 = \max(0, 0.1 * 0.63 + 0) = 0.063$$

# Sample LSTM Question

Compute the outputs in each timestep and the state after timestep=3 for the LSTM given below. Assume the biases as zeros. [5]



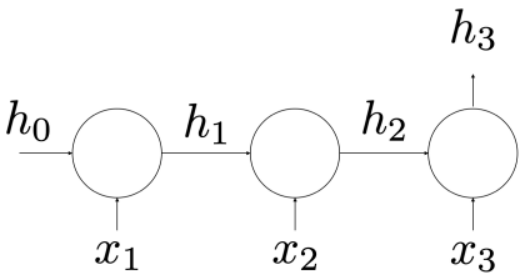

$$X = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^\top$$
$$W = \begin{bmatrix} 0.2 & 0.3 & 0.8 \end{bmatrix}^\top$$
$$U = \begin{bmatrix} 0.5 & 0.6 & 0.2 \end{bmatrix}^\top$$
$$V = \begin{bmatrix} 0.4 & 0.2 & 0.1 \end{bmatrix}^\top$$
$$W_o = \begin{bmatrix} 0.90.10.3 \end{bmatrix}^\top$$
$$U_o = \begin{bmatrix} 0.40.30.2 \end{bmatrix}^\top$$
$$W_i = \begin{bmatrix} 0.10.40.2 \end{bmatrix}^\top$$
$$U_i = \begin{bmatrix} 0.90.10.3 \end{bmatrix}^\top$$
$$W_f = \begin{bmatrix} 0.50.30.2 \end{bmatrix}^\top$$
$$U_f = \begin{bmatrix} 0.60.20.3 \end{bmatrix}^\top$$

Solution

$$s_t = \sigma(Ux_t + Ws_{t-1} + b)$$
$$\hat{y}_t = Relu(Vs_t + c) \qquad \text{Relu is assumed}$$
$$s_0 = 0 \qquad h_0 = 0 \qquad b = 0 \qquad c = 0$$
$$o_1 = \sigma(0.9 * 0 + 0.4 * 1 + 0) = 0.598$$
$$i_1 = \sigma(0.1 * 0 + 0.3 * 1 + 0) = 0.574$$
$$f_1 = \sigma(0.5 * 0 + 0.6 * 1 + 0) = 0.645$$
$$\hat{s}_t = \sigma(Wh_0 + Ux_1 + b)$$
$$\hat{s}_1 = \sigma(0.2 * 0 + 0.5 * 1 + 0) = 0.62$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \hat{s}_t$$
$$s_1 = f_1 \odot s_0 + i_1 \hat{s}_1 = 0.645 * 0 + 0.574 * 0.62 = 0.355$$
$$h_1 = o_1 \odot \sigma(s_1)$$
$$= 0.598 * \sigma(0.355) = 0.351$$
$$\hat{y}_1 = \sigma(0.4 * 0.355 + 0)$$

## Problem 6 Recurrent Neural Networks and Backpropagation (9 credits)

Consider a vanilla RNN cell of the form $h_t = \tanh(V \cdot h_{t-1} + W \cdot x_t + b)$. The figure below shows the input sequence $x_1$, $x_2$, and $x_3$.




0 1 a) Given the dimensions $x_t \in \mathbb{R}^3$ and $h_t \in \mathbb{R}^5$, what is the number of parameters in the RNN cell? (Calculate final number)

$3 \times 5 + 5 \times 5 + 5(bias) = 15 + 25 + 5 = 45$ (1p for 45, else 0)

0 1 b) If $x_t$ and $b$ are the 0 vector, then $h_t = h_{t-1}$ for any value of $h_t$. Discuss whether this statement is correct.

False: ( 0.5p)
After transformation with $V$ and non-linearity $x_t = 0$ does not lead to $h_t = h_{t-1}$ (0.5p) , i.e. $h_t$ can be changed.
Note: simply repeating the formula $h_t = \tanh(V \cdot h_{t-1})$ does not give any points. If you only mention $V$ or *tanh* then this is also correct, though giving an incorrect formula invalidates that half point.

Now consider the following **one-dimensional** ReLU-RNN cell without bias b.

$$h_t = \text{ReLU}(V \cdot h_{t-1} + W \cdot x_t)$$

(Hidden state, input, and weights are scalars)

0 1 2 c) Calculate $h_2$ and $h_3$ where

$$V = -3, \quad W = 3, \quad h_0 = 0, \quad x_1 = 2, \quad x_2 = 3 \quad \text{and } x_3 = 1.$$

$h_0 = 0$
$h_1 = \text{relu}(-3 \cdot 0 + 3 \cdot 2) = 6$
$h_2 = \text{relu}(-3 \cdot 6 + 3 \cdot 3) = 0$ (1 p)
$h_3 = \text{relu}(-3 \cdot 0 + 3 \cdot 1) = 3$ (1 p)

Note: Only points for correct solutions, no points for intermediate steps (even if you have an incorrect $h_1$)

d) Calculate the derivatives $\frac{\partial h_3}{\partial V}$, $\frac{\partial h_3}{\partial W}$, and $\frac{\partial h_3}{\partial x_1}$ for the forward pass of the ReLU-RNN where

$$V = -2, \quad W = 1, \quad h_0 = 2, \quad x_1 = 2, \quad x_2 = \frac{3}{2} \quad \text{and } x_3 = 4.$$

for the forward outputs

$$h_1 = 0, \quad h_2 = \frac{2}{3}, \quad h_3 = 1.$$

Use that $\frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=0} = 0$.

0 1 2 3

Generally:

$$\frac{\partial h_t}{\partial V} = h_{t-1} + V \cdot \frac{\partial h_{t-1}}{\partial V}$$
$$\frac{\partial h_t}{\partial W} = \frac{\partial \text{ReLU}(z_t)}{\partial z_t} \cdot \left( V \cdot \frac{\partial h_{t-1}}{\partial W} + x_t \right)$$
$$\frac{\partial h_t}{\partial x_\tau} = \frac{\partial \text{ReLU}(z_t)}{\partial z_t} \cdot \left( V \cdot \frac{\partial h_t}{\partial x_\tau} + W \cdot \delta_{t\tau} \right)$$

$$\frac{\partial h_3}{\partial V} = h_2 + V \cdot h_1 = \frac{2}{3} + 0 = \frac{2}{3} \quad \text{(1p)}$$
$$\frac{\partial h_3}{\partial W} = V \cdot x_2 + x_3 = -2 \cdot \frac{3}{2} + 4 = 1 \quad \text{(1p)}$$
$$\frac{\partial h_3}{\partial x_1} = 0 \text{ (dead ReLU )} \quad \text{(1p)}$$

Note: alternatively $\frac{\partial h_3}{\partial V} = \frac{3}{2}$ if student correctly identified that $h_2$ should have been flipped to be a correct forward pass.
For $\frac{\partial h_3}{\partial x_1}$, it's okay even if no formula, but some explanation is given (dead relu after first layer)

0 1 2

e) A Long-Short Term Memory (LSTM) unit is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

where $g_1$, $g_2$, and $g_3$ are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** $f$, **update** $u$, and **output** $o$ gates. (1p)
2) What does the value $c_t$ represent in a LSTM? (1p)

$g_1$ = output gate
$g_2$ = forget gate
$g_3$ = update gate/input gate
(1p for all three, zero otherwise)
$c_t$: cell state/memory (1p)
Note: if students interpreted $c_t$ as "what does it do?" half a point was awarded. Possible half point: “Intermediate value, check what to forget and what to add from input”

## Problem 10 Recurrent Neural Networks and Backpropagation (8 credits)

Recurrent neural networks, also known as RNNs, are a class of neural networks that allow an arbitrary number of inputs and, thus, are often used for sequences of data, e.g., in the fields of natural language processing and speech recognition.

a) Mathematically explain the reason for exploding and vanishing gradients when using a classic RNN, i.e., $A_t = \theta_c A_{t-1} + \theta_x x_t$, where both $\theta_c$ and $\theta_x$ are orthogonal. (2 points)

0 1 2

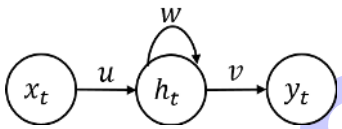
1. Show backpropagation explicitly to calculate gradients (1 point):

$$\frac{\delta h^{(t)}}{\delta h^{(1)}} = \frac{\delta h^{(t)}}{\delta h^{(t-1)}} \cdots \frac{\delta h^{(2)}}{\delta h^{(1)}}$$

2. After eigen-decomposition of $\theta_c$, the **largest** eigenvalue of $\theta_c > 1$ means explosion(0.5 point) and $< 1$ means vanishing (0.5 point).
   P.s. if the answer discussed eigenvalues of $A_t$ instead of $\theta_c$: 0 point
   if the answer discussed two cases of eigenvalues but didn't show explicitly which matrix should be decomposed: 0.5 point

b) Now consider the following RNN

0 1 2 3




which uses the one-dimensional ReLU-RNN cell

$$h_t = ReLU(u * h_{t-1} + w * x_t).$$

Compute the forward propagation $y_2, h_2$ and the gradient $\mathbf{dy}_2 := \frac{\delta y_2}{\delta u}$ where

$$h_0 = 3,\ w = 2,\ v = -1,\ u = 3,\ x_1 = 1,\ x_2 = 2.$$

Since there is a mismatch between the graph and the given equation, answers based either on the graph or on the equation are accepted. Solution based on the graph:

$$h_1 = ReLU(u * x_1 + w * h_0) = 9$$
$$h_2 = ReLU(u * x_2 + w * h_1) = 24(1p)$$
$$y_2 = v * h_2 = -24(1p)$$
$$\frac{\delta y_2}{\delta u} = v * w * x_1 + v * x_2 = -4(1p)$$

Solution based on the equation:

$$h_1 = ReLU(u * h_0 + w * x_1) = 11$$
$$h_2 = ReLU(u * h_1 + w * x_2) = 37(1p)$$
$$y_2 = v * h_2 = -37(1p)$$
$$\frac{\delta y_2}{\delta u} = v * (h_1 + u * h_0) = -20(1p)$$

Each equation and final result counts 0.5 points.

0 1 2

c) To circumvent the vanishing gradient problem, the Long-Short Term Memory (LSTM) unit was proposed. It is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

where $g_1$, $g_2$, and $g_3$ are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** $f$, **update** $u$, and **output** $o$ gates. (1p)
2) What does the value $c_t$ represent in a LSTM? (1p)

$g_1$ = output gate
$g_2$ = forget gate
$g_3$ = update gate/input gate
(1 pt)
$c_t$: cell state
(1 pt)

0 1

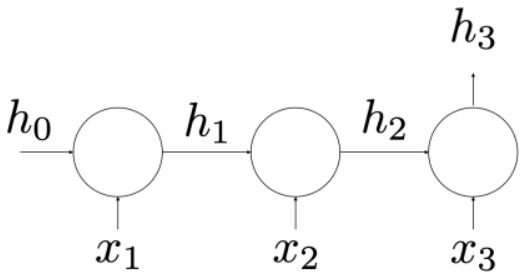
d) Why does the LSTM unit solve the vanishing gradient problem that is present in the default definition of an RNN cell?

gradient highway through the cell state (0.5pt) and gate system to remove or add information to the cell state (0.5pt)
(another alternative: from the perspectives of weights and activation functions as in the slides)

## Problem 6 Recurrent Neural Networks and LSTMs (12 credits)

a) Consider a vanilla RNN cell of the form $h_t = \tanh(V \cdot h_{t-1} + W \cdot x_t)$. The figure below shows the input sequence $x_1$, $x_2$, and $x_3$.

0
1
2




Given the dimensions $x_t \in \mathbb{R}^4$ and $h_t \in \mathbb{R}^{12}$, what is the number of parameters in the RNN cell? Neglect the bias parameter.

$4 \times 12 + 12 \times 12$ (1 pt) $= 48 + 144 = 192$ (1 pt)

b) ~~If $x_t$ is the 0 vector, then $h_t = h_{t-1}$.~~ Discuss whether this statement is correct.

0
1
2

False: ( 1 pt)
After transformation with $V$ and non-linearity $x_t = 0$ does not lead to $h_t = h_{t-1}$ (1 pt) . Full points require explanation, solely equation not sufficient.

0 1 2 3

c) Now consider the following **one-dimensional** ReLU-RNN cell.

$$h_t = \text{ReLU}(V \cdot h_{t-1} + W \cdot x_t)$$

(Hidden state, input, and weights are scalars)
Calculate $h_1$, $h_2$ and $h_3$ where $V = 1$, $W = 2$, $h_0 = -3$, $x_1 = 1$, $x_2 = 2$ and $x_3 = 0$.

$h_0 = -3$
$h_1 = \text{relu}(1 \cdot (-3) + 2 \cdot 1) = 0$ ( 1 pt)
$h_2 = \text{relu}(1 \cdot 0 + 2 \cdot 2) = 4$ (1 pt)
$h_3 = \text{relu}(1 \cdot 4 + 2 \cdot 0) = 4$ ( 1 pt)

d) Calculate the derivatives $\frac{\partial h_3}{\partial V}$, $\frac{\partial h_3}{\partial W}$, and $\frac{\partial h_3}{\partial x_1}$ for the forward pass of the ReLU-RNN Cell of (c). Use that $\frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=0} = 1$.

0 1 2 3

$h_t = \text{ReLU}(V \cdot h_{t-1} + W \cdot x_t) = \text{ReLU}(z_t)$

$\frac{\partial h_3}{\partial V} = \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_3} \cdot h_2 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot h_1 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_1} \cdot V^2 \cdot h_0 =$

$= 1 \cdot 4 + 1 \cdot 1 \cdot 0 + 0 \cdot 1 \cdot (-3) = 4$ (1 pt)

$\frac{\partial h_3}{\partial W} = \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_3} \cdot x_3 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot x_2 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_1} \cdot V^2 \cdot x_1 =$

$1 \cdot 0 + 1 \cdot 2 + 0 \cdot 0 = 2$ (1 pt)

$\frac{\partial h_3}{\partial x_1} = \text{ReLU}(x)\Big|_{x=z_3} \cdot V \cdot \text{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot \text{ReLU}(x)\Big|_{x=z_1} \cdot W = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 0 \cdot 2 = 0$ (1 pt)

Only correct and calculated result gives point.

0
1
2

e) A Long-Short Term Memory (LSTM) unit is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

where $g_1$, $g_2$, and $g_3$ are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** $f$, **update** $u$, and **output** $o$ gates. (1p)
~~2) What does the value $c_t$ represent in a LSTM? (1p)~~

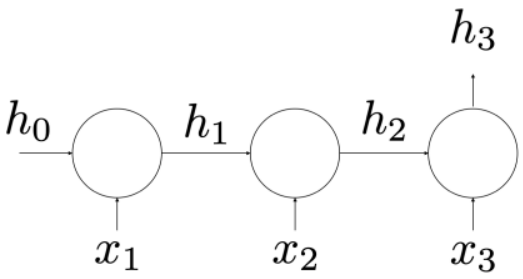
$g_1$ = output gate
$g_2$ = forget gate
$g_3$ = update gate
(1 pt)
$c_t$: cell state
(1 pt)

# RNN Questions

Introduction to Deep Learning (Technische Universität München)

## Problem 6 Recurrent Neural Networks and Backpropagation (9 credits)

Consider a vanilla RNN cell of the form $h_t = \tanh(V \cdot h_{t-1} + W \cdot x_t + b)$. The figure below shows the input sequence $x_1$, $x_2$, and $x_3$.




0 1 a) Given the dimensions $x_t \in \mathbb{R}^3$ and $h_t \in \mathbb{R}^5$, what is the number of parameters in the RNN cell? (Calculate final number)

$3 \times 5 + 5 \times 5 + 5(bias) = 15 + 25 + 5 = 45$ (1p for 45, else 0)

0 1 b) If $x_t$ and $b$ are the 0 vector, then $h_t = h_{t-1}$ for any value of $h_t$. Discuss whether this statement is correct.

False: ( 0.5p)
After transformation with $V$ and non-linearity $x_t = 0$ does not lead to $h_t = h_{t-1}$ (0.5p) , i.e. $h_t$ can be changed.
Note: simply repeating the formula $h_t = \tanh(V \cdot h_{t-1})$ does not give any points. If you only mention $V$ or *tanh* then this is also correct, though giving an incorrect formula invalidates that half point.

Now consider the following **one-dimensional** ReLU-RNN cell without bias b.

$$h_t = \text{ReLU}(V \cdot h_{t-1} + W \cdot x_t)$$

(Hidden state, input, and weights are scalars)

0 1 2 c) Calculate $h_2$ and $h_3$ where

$$V = -3, \quad W = 3, \quad h_0 = 0, \quad x_1 = 2, \quad x_2 = 3 \quad \text{and } x_3 = 1.$$

$h_0 = 0$
$h_1 = \text{relu}(-3 \cdot 0 + 3 \cdot 2) = 6$
$h_2 = \text{relu}(-3 \cdot 6 + 3 \cdot 3) = 0$ (1 p)
$h_3 = \text{relu}(-3 \cdot 0 + 3 \cdot 1) = 3$ (1 p)

Note: Only points for correct solutions, no points for intermediate steps (even if you have an incorrect $h_1$)

d) Calculate the derivatives $\frac{\partial h_3}{\partial V}$, $\frac{\partial h_3}{\partial W}$, and $\frac{\partial h_3}{\partial x_1}$ for the forward pass of the ReLU-RNN where

$$V = -2, \quad W = 1, \quad h_0 = 2, \quad x_1 = 2, \quad x_2 = \frac{3}{2} \quad \text{and } x_3 = 4.$$

for the forward outputs

$$h_1 = 0, \quad h_2 = \frac{2}{3}, \quad h_3 = 1.$$

Use that $\frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=0} = 0$.

0 1 2 3

Generally:

$$\frac{\partial h_t}{\partial V} = h_{t-1} + V \cdot \frac{\partial h_{t-1}}{\partial V}$$
$$\frac{\partial h_t}{\partial W} = \frac{\partial \text{ReLU}(z_t)}{\partial z_t} \cdot \left( V \cdot \frac{\partial h_{t-1}}{\partial W} + x_t \right)$$
$$\frac{\partial h_t}{\partial x_\tau} = \frac{\partial \text{ReLU}(z_t)}{\partial z_t} \cdot \left( V \cdot \frac{\partial h_t}{\partial x_\tau} + W \cdot \delta_{t\tau} \right)$$

$$\frac{\partial h_3}{\partial V} = h_2 + V \cdot h_1 = \frac{2}{3} + 0 = \frac{2}{3} \quad \text{(1p)}$$
$$\frac{\partial h_3}{\partial W} = V \cdot x_2 + x_3 = -2 \cdot \frac{3}{2} + 4 = 1 \quad \text{(1p)}$$
$$\frac{\partial h_3}{\partial x_1} = 0 \text{ (dead ReLU )} \quad \text{(1p)}$$

Note: alternatively $\frac{\partial h_3}{\partial V} = \frac{3}{2}$ if student correctly identified that $h_2$ should have been flipped to be a correct forward pass.
For $\frac{\partial h_3}{\partial x_1}$, it's okay even if no formula, but some explanation is given (dead relu after first layer)

0 1 2

e) A Long-Short Term Memory (LSTM) unit is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

where $g_1$, $g_2$, and $g_3$ are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** $f$, **update** $u$, and **output** $o$ gates. (1p)
2) What does the value $c_t$ represent in a LSTM? (1p)

$g_1$ = output gate
$g_2$ = forget gate
$g_3$ = update gate/input gate
(1p for all three, zero otherwise)
$c_t$: cell state/memory (1p)
Note: if students interpreted $c_t$ as "what does it do?" half a point was awarded. Possible half point: “Intermediate value, check what to forget and what to add from input”

## Problem 10 Recurrent Neural Networks and Backpropagation (8 credits)

Recurrent neural networks, also known as RNNs, are a class of neural networks that allow an arbitrary number of inputs and, thus, are often used for sequences of data, e.g., in the fields of natural language processing and speech recognition.

a) Mathematically explain the reason for exploding and vanishing gradients when using a classic RNN, i.e., $A_t = \theta_c A_{t-1} + \theta_x x_t$, where both $\theta_c$ and $\theta_x$ are orthogonal. (2 points)

0 1 2

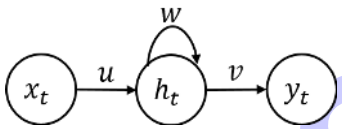
1. Show backpropagation explicitly to calculate gradients (1 point):

$$\frac{\delta h^{(t)}}{\delta h^{(1)}} = \frac{\delta h^{(t)}}{\delta h^{(t-1)}} \cdots \frac{\delta h^{(2)}}{\delta h^{(1)}}$$

2. After eigen-decomposition of $\theta_c$, the **largest** eigenvalue of $\theta_c > 1$ means explosion(0.5 point) and $< 1$ means vanishing (0.5 point).
   P.s. if the answer discussed eigenvalues of $A_t$ instead of $\theta_c$: 0 point
   if the answer discussed two cases of eigenvalues but didn't show explicitly which matrix should be decomposed: 0.5 point

b) Now consider the following RNN

0 1 2 3




which uses the one-dimensional ReLU-RNN cell

$$h_t = ReLU(u * h_{t-1} + w * x_t).$$

Compute the forward propagation $y_2, h_2$ and the gradient $\mathbf{dy}_2 := \frac{\delta y_2}{\delta u}$ where

$$h_0 = 3,\ w = 2,\ v = -1,\ u = 3,\ x_1 = 1,\ x_2 = 2.$$

Since there is a mismatch between the graph and the given equation, answers based either on the graph or on the equation are accepted. Solution based on the graph:

$$h_1 = ReLU(u * x_1 + w * h_0) = 9$$
$$h_2 = ReLU(u * x_2 + w * h_1) = 24(1p)$$
$$y_2 = v * h_2 = -24(1p)$$
$$\frac{\delta y_2}{\delta u} = v * w * x_1 + v * x_2 = -4(1p)$$

Solution based on the equation:

$$h_1 = ReLU(u * h_0 + w * x_1) = 11$$
$$h_2 = ReLU(u * h_1 + w * x_2) = 37(1p)$$
$$y_2 = v * h_2 = -37(1p)$$
$$\frac{\delta y_2}{\delta u} = v * (h_1 + u * h_0) = -20(1p)$$

Each equation and final result counts 0.5 points.

0 1 2

c) To circumvent the vanishing gradient problem, the Long-Short Term Memory (LSTM) unit was proposed. It is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

where $g_1$, $g_2$, and $g_3$ are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** $f$, **update** $u$, and **output** $o$ gates. (1p)
2) What does the value $c_t$ represent in a LSTM? (1p)

$g_1$ = output gate
$g_2$ = forget gate
$g_3$ = update gate/input gate
(1 pt)
$c_t$: cell state
(1 pt)

0 1

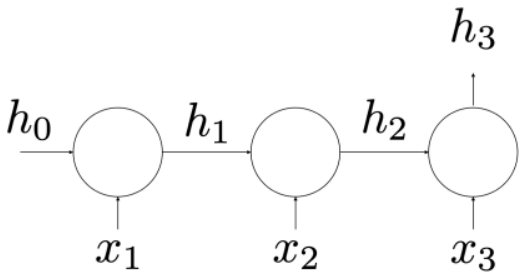
d) Why does the LSTM unit solve the vanishing gradient problem that is present in the default definition of an RNN cell?

gradient highway through the cell state (0.5pt) and gate system to remove or add information to the cell state (0.5pt)
(another alternative: from the perspectives of weights and activation functions as in the slides)

## Problem 6 Recurrent Neural Networks and LSTMs (12 credits)

a) Consider a vanilla RNN cell of the form $h_t = \tanh(V \cdot h_{t-1} + W \cdot x_t)$. The figure below shows the input sequence $x_1$, $x_2$, and $x_3$.

0
1
2




Given the dimensions $x_t \in \mathbb{R}^4$ and $h_t \in \mathbb{R}^{12}$, what is the number of parameters in the RNN cell? Neglect the bias parameter.

$4 \times 12 + 12 \times 12$ (1 pt) $= 48 + 144 = 192$ (1 pt)

b) ~~If $x_t$ is the 0 vector, then $h_t = h_{t-1}$.~~ Discuss whether this statement is correct.

0
1
2

False: ( 1 pt)
After transformation with $V$ and non-linearity $x_t = 0$ does not lead to $h_t = h_{t-1}$ (1 pt) . Full points require explanation, solely equation not sufficient.

0 1 2 3

c) Now consider the following **one-dimensional** ReLU-RNN cell.

$$h_t = \text{ReLU}(V \cdot h_{t-1} + W \cdot x_t)$$

(Hidden state, input, and weights are scalars)
Calculate $h_1$, $h_2$ and $h_3$ where $V = 1$, $W = 2$, $h_0 = -3$, $x_1 = 1$, $x_2 = 2$ and $x_3 = 0$.

$h_0 = -3$
$h_1 = \text{relu}(1 \cdot (-3) + 2 \cdot 1) = 0$ ( 1 pt)
$h_2 = \text{relu}(1 \cdot 0 + 2 \cdot 2) = 4$ (1 pt)
$h_3 = \text{relu}(1 \cdot 4 + 2 \cdot 0) = 4$ ( 1 pt)

d) Calculate the derivatives $\frac{\partial h_3}{\partial V}$, $\frac{\partial h_3}{\partial W}$, and $\frac{\partial h_3}{\partial x_1}$ for the forward pass of the ReLU-RNN Cell of (c). Use that $\frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=0} = 1$.

0 1 2 3

$h_t = \text{ReLU}(V \cdot h_{t-1} + W \cdot x_t) = \text{ReLU}(z_t)$

$\frac{\partial h_3}{\partial V} = \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_3} \cdot h_2 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot h_1 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_1} \cdot V^2 \cdot h_0 =$

$= 1 \cdot 4 + 1 \cdot 1 \cdot 0 + 0 \cdot 1 \cdot (-3) = 4$ (1 pt)

$\frac{\partial h_3}{\partial W} = \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_3} \cdot x_3 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot x_2 + \frac{\partial}{\partial x}\text{ReLU}(x)\Big|_{x=z_1} \cdot V^2 \cdot x_1 =$

$1 \cdot 0 + 1 \cdot 2 + 0 \cdot 0 = 2$ (1 pt)

$\frac{\partial h_3}{\partial x_1} = \text{ReLU}(x)\Big|_{x=z_3} \cdot V \cdot \text{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot \text{ReLU}(x)\Big|_{x=z_1} \cdot W = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 0 \cdot 2 = 0$ (1 pt)

Only correct and calculated result gives point.

0
1
2

e) A Long-Short Term Memory (LSTM) unit is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

where $g_1$, $g_2$, and $g_3$ are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** $f$, **update** $u$, and **output** $o$ gates. (1p)
~~2) What does the value $c_t$ represent in a LSTM? (1p)~~

$g_1$ = output gate
$g_2$ = forget gate
$g_3$ = update gate
(1 pt)
$c_t$: cell state
(1 pt)

Q1.

```
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)) izer(learning rate=0.5).min
optimizer = tf.train.AdamOptimizer(learning_rate=0.5).minimize(cost)
init = tf.initialize_all_variables()
```

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?

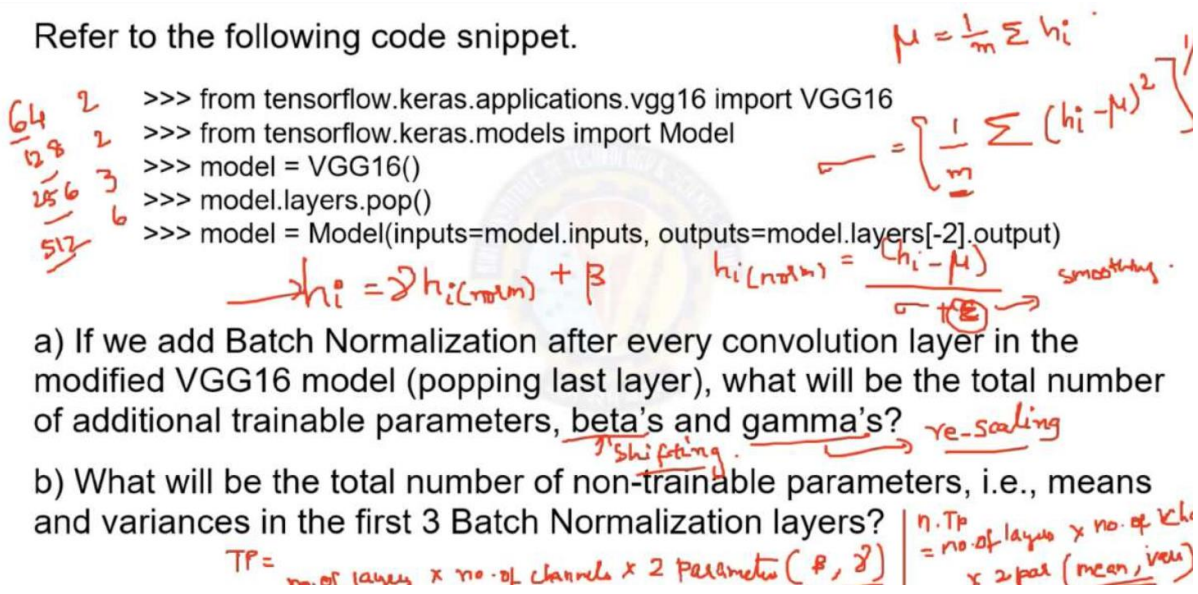
b) Calculate the output values assuming the input vector to the output layer to be [2,0,1,0], weight matrix to be [[0.2,0.3,0.2,0.1], [0.1,0.2,0.5,0.1], [0.2,0.1,0.1,0.1]] and bias vector to be [0.3,0.1,0.1]

**Solution:**

**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)** X.W^T+b = [0.9, 0.8, 0.6]; softmax values = [0.38, 0.34, 0.28]

**Q2.**



## Answer

a) 8448

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

| Conv layer Channels | No. of layers | Tp | | 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 |

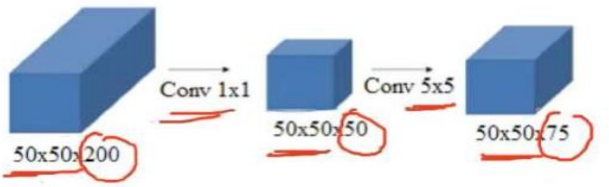
Q3.

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

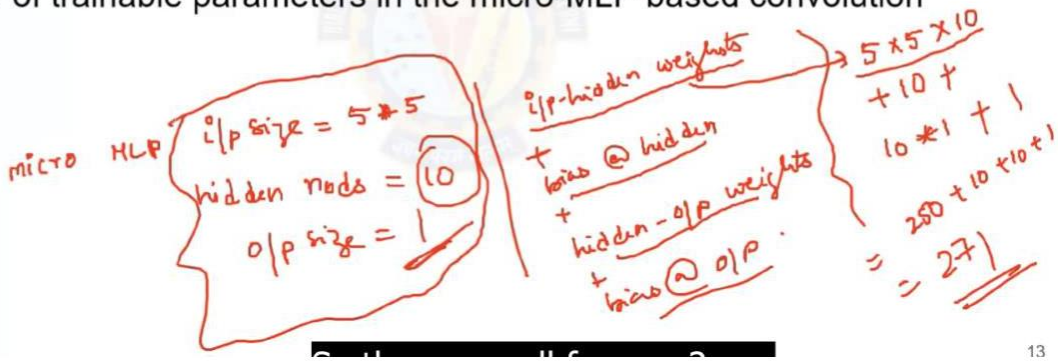
For 5x5 convolution,

# 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?






## 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

## 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 ( inp dim + 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



## 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

**Question:**

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

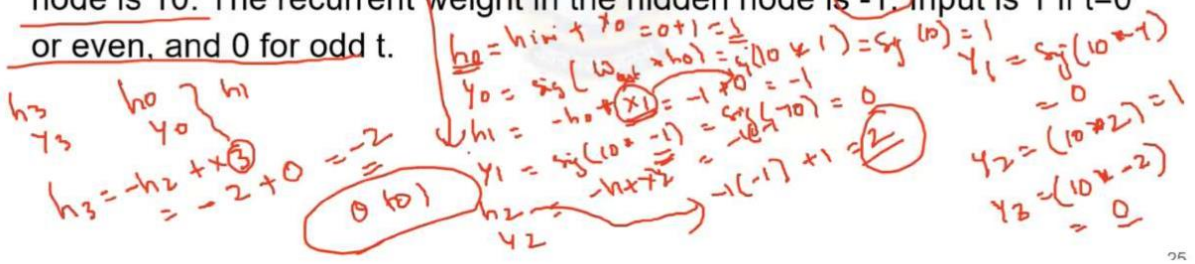
Input-to-hidden weights = (20+10)*20

Hidden-to-output weights = 20*50

Total weights = 1600

## Question

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.



## Answer

h0 = h(initial) + x0 = 0 + 1 = 1

y(0) = sig(wout * h0) = sig(10 * 1) = sig(10) = 1

h1 = −h0 + x1 = −1 + 0 = −1

y(1) = sig(wout * h0) = sig(10 * −1) = sig(−10) = 0

h2 = −h1 + x2 = −1(−1) + 1 = 2

y(1) = sig(wout * h0) = sig(10 * 2) = sig(20) = 1

h3 = −h2 + x3 = −2 + 0 = 2 = −2

y(1) = sig(wout * h0) = sig(10 * −2) = 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.

**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 ()
```

**Answer:**

This code snippet creates a sequential neural network model using the Keras API with the following architecture:

1. **Embedding Layer**:
   - The first layer added to the model is an Embedding layer. This layer is used for text processing tasks and is typically applied to sequences of integers representing words.

   - Parameters:
     - `input_dim`: Specifies the size of the vocabulary, i.e., the maximum integer index + 1. In this case, it's set to 20000, indicating that the model expects input sequences composed of integers ranging from 0 to 19999.
     - `output_dim`: Specifies the dimension of the embedding vector for each word. In this case, it's set to 128, meaning each word will be represented as a vector of length 128.

2. **LSTM Layer**:
   - The second layer added is an LSTM (Long Short-Term Memory) layer. LSTM is a type of recurrent neural network (RNN) architecture that is well-suited for sequence prediction problems, especially in natural language processing tasks.

   - Parameters:
     - `128`: Specifies the number of units (or neurons) in the LSTM layer.
     - `dropout`: Specifies the dropout rate, which is a regularization technique to prevent overfitting by randomly dropping a fraction of input units during training. Here, it's set to 0.2, indicating that 20% of the input units will be randomly dropped during training.

3. **Dense Layer**:
   - The final layer added is a Dense layer. This is a fully connected layer where each neuron is connected to every neuron in the previous layer.

   - Parameters:
     - `1`: Specifies the number of units in the Dense layer, which is 1 in this case. Since the task involves binary classification (sigmoid activation function is used), a single unit is sufficient to produce the binary output.
     - `activation='sigmoid'`: Specifies the activation function to be used in the Dense layer. The sigmoid activation function is chosen here, which squashes the output values between 0 and 1, making it suitable for binary classification tasks where the output represents probabilities.

4. **Model Summary**:
   - Finally, the `summary()` method is called on the model to display a summary of the model architecture, including the type of each layer, output shape, and number of parameters.

## 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.

**Answer:**

**Sure, let's tackle each part one by one:**

**(a) Here's a rough sketch of the CNN architecture described:**

**Input (128x128x3)**
**|**
**Conv2D (32 filters, 5x5)**
**|**
**MaxPooling2D**
**|**
**Conv2D (64 filters, 5x5)**
**|**
**MaxPooling2D**
**|**
**Conv2D (128 filters, 5x5)**
**|**
**MaxPooling2D**
**|**
**Conv2D (256 filters, 5x5)**
**|**
**MaxPooling2D**
**|**
**Flatten**
**|**
**Dense (1024 neurons)**
**|**
**Dense (512 neurons)**
**|**
**Output (2 classes)**

**(b) Here's the TensorFlow-Keras code snippet for the described architecture:**

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential()
```

```
# Convolutional layers
model.add(Conv2D(32, (5, 5), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Fully connected layers
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(512, activation='relu'))

# Output layer
model.add(Dense(2, activation='softmax'))  # Assuming binary classification

model.summary()
```

(c) Let's calculate the number of parameters learned in each convolution and maxpooling layers:

1. **Convolutional Layer 1**:
   - Parameters learned:
     - Number of filters: 32
     - Filter size: 5x5
     - Input channels: 3 (RGB)
   - Total parameters: (5 * 5 * 3 + 1) * 32 = 2432

2. **MaxPooling Layer 1**:
   - MaxPooling layers do not have any trainable parameters. They only perform downsampling.

3. **Convolutional Layer 2**:
   - Parameters learned:
     - Number of filters: 64
     - Filter size: 5x5
   - Total parameters: (5 * 5 * 32 + 1) * 64 = 51264

**4. **MaxPooling Layer 2**:**
**- No trainable parameters.**

**5. **Convolutional Layer 3**:**
**- Parameters learned:**
**- Number of filters: 128**
**- Filter size: 5x5**
**- Total parameters: (5 * 5 * 64 + 1) * 128 = 204928**

**6. **MaxPooling Layer 3**:**
**- No trainable parameters.**

**7. **Convolutional Layer 4**:**
**- Parameters learned:**
**- Number of filters: 256**
**- Filter size: 5x5**
**- Total parameters: (5 * 5 * 128 + 1) * 256 = 819456**

**8. **MaxPooling Layer 4**:**
**- No trainable parameters.**

**In summary, we calculated the number of parameters learned in each convolutional layer, while maxpooling layers don't have any parameters as they perform downsampling only.**

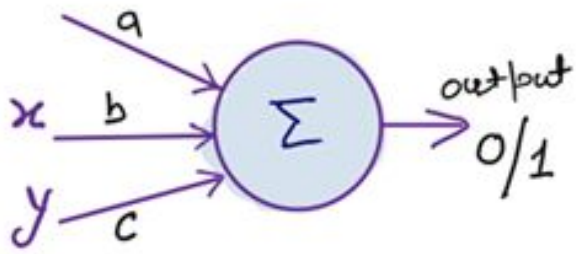
AIML- Cohort 1 -- C6
Compri MakeupExam
Max Marks : 30
---

Q1. Explain the following concepts in machine learning
**[2*3=6Marks]**
A. Xavier initialization.
B. Style GAN
C. Re-parameterization and its application

Answer:
Explanation using
A: Standard Definition
B: Standard Definition
C: Standard Definition

Q2.

Device weights a, b, c for the following neuron

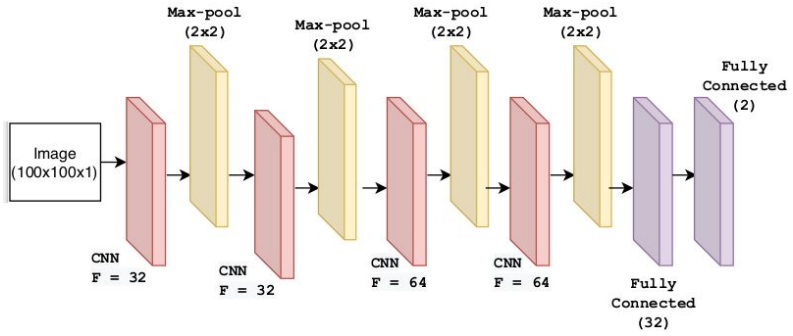



Such that it produces the following output.

| X | Y | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Answer:
Equations are a > 0 ; a+c>0; a+b<0; a+b+c <0
Solution is: a =1; c=1; b=-3;

Q3. A user has collected a lot of 100x100 size images. Some of them contain images of mountains and other of airplanes. The issue is that the number of imges is large and the user keeps adding more and more images. He did not want to manually label image as mountain or airplanes. He has devised a neural network to do the task. All activation functions to be RelU except at the final layer, where it is softmax. Kernel size for the CNN is throughout kept 5x5. Network architecture is given below (Note: CNN and Max pool layers are shown in pink and yellow respectively. Blue ones are fully connected layers).
Determine the number of trainable parameters for this architecture? Assume bias term is also need to be added in a CNN and image size reduces after an convolution operation.




Answer:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (5, 5), activation='relu',input_shape=(100, 100, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(2))
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 96, 96, 32)        832
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 44, 44, 32)        25632
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 22, 22, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 18, 18, 64)        51264
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 9, 9, 64)          0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 5, 5, 64)          102464
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 2, 2, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                8224
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 66
=================================================================
```

Total params: 188,482
Trainable params: 188,482
Non-trainable params: 0

Q4. Answer the below questions: **[1+2 = 3Marks]**

A. Explain weight decay.

B. How it leads to the regularization?

Answer:

A: Standard Definition

B: Standard Definition

Q5. Answer the following briefly. [1*3 = 3Marks]
A. What is the philosophical difference between GoogleLeNet and ResNet.
B. What was the biggest challenge in training a deeper network such as ResNet ?
C. How it was resolved in ResNet?

Answer:

A: GoogleLeNet aims for reducing computations and seeing things at multiple scales. However ResNet targets a simple architecture that is very deep and expects that the extralayers would automatically behave as identity mapping.
B: Biggest challenge that the standard solution of identity mappoint is just a single point in infinite space. So the chances of hitting the solution is very low.
C: ResNet applies skip connections to resolve the same.

Q6. Consider a random variable x, taking value in the range [1,6]. Consider two distributions, P and Q, as shown in the table below and answer the following questions: **[5+1 = 6Marks]**

| | P(x) | Q(x) |
|---|---|---|
| x=1 | 0 | 0.3 |
| x=2 | 0 | 0.1 |
| x=3 | 0.2 | 0.2 |
| x=4 | 0.3 | 0.1 |
| x=5 | 0.1 | 0.3 |
| x=6 | 0.4 | 0 |

A. Compute JS-Divergence between two distributions P and Q.
B. Without computation, comment on the possible value of KL-divergence of P and Q

Answer:
A: JSD is as above

| x | P | Q | M=(P+Q)/2 | log(P/M) | log(Q/M) | P*log(P/M) | Q*log(Q/M) |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0.3 | 0.15 | 0 | 0.301029996 | 0 | 0.0903089987 |
| 2 | 0 | 0.1 | 0.05 | 0 | 0.301029996 | 0 | 0.0301029996 |
| 3 | 0.2 | 0.2 | 0.2 | 0 | 0 | 0 | 0 |
| 4 | 0.3 | 0.1 | 0.2 | 0.176091259 | -0.301029996 | 0.0528273777 | -0.030103 |
| 5 | 0.1 | 0.3 | 0.2 | -0.301029996 | 0.176091259 | -0.030103 | 0.0528273777 |
| 6 | 0.4 | 0 | 0.2 | 0.301029996 | 0 | 0.1204119983 | 0 |
| | | | | | | **0.1431363764** | **0.1431363764** |
| | | | | | | | **0.14313638** |

NOTE: here log is at base 10. if someone takes log on another base (2 or e) they also will be awarded full marks if there are no calculation mistakes.

B: LK-divergence does not exist as neither P to Q not Q to P fully covers. So it is not possible to compute KLD.

# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

**M. Tech. in AI & ML**

**II Semester 2022-2023**

**End-Semester Test**
**(EC3 - Regular)**

| | |
|---|---|
| Course Number | AIMLCZG51 |
| Course Name | DEEP NEURAL NETWORK |
| Nature of Exam | Open Book |
| Weight-age for grading | 40 |
| Duration | 2.5 hrs |
| Date of Exam | |

| | |
|---|---|
| * Pages | 4 |
| * Questions | 4 |

---

1. (a) Consider a CNN architecture with an input image of size $256 \times 256 \times 3$. The architecture consists of two convolutional layers with 64 and 128 filters of size $5 \times 5$ respectively, followed by a max-pooling layer of size $2 \times 2$. Calculate the output dimensions after each layer. [3]

   (b) If the input image size is changed to $128 \times 128 \times 3$, how will it affect the number of trainable parameters? [1]

   (c) Compute the trainable parameters are there in the second convolutional layer for part (a). [1]

   (d) Write Python code to implement for part (a) architecture using TensorFlow Keras. Add one dense layer and an output layer for binary classification. [3]

   (e) I represents the top left corner pixel values of a much larger image (not shown in entirety). F is a filter that convolves over this image with stride 1. O is the result of this convolution, where only 3 resulting pixels at top left are shown. Importantly, there is no zero padding, which implies that each pixel in O is the dot product resulting from the filter being placed at the corresponding position on the image in I. Find the values of x, y and z. [3]

$$I = \begin{pmatrix} 3 & 0 & 3 & -3 & 0 & \dots \\ -3 & 2 & 0 & 3 & -2 & \dots \\ -5 & 0 & 3 & -2 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} \quad F = \begin{pmatrix} x & y & z \\ 1 & 0 & 2 \\ -1 & 1 & 0 \end{pmatrix} \quad O = \begin{pmatrix} 4 & 0 & -3 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

Rubrics and one solution:

(a)
- First Convolutional Layer (64 filters): [1 mark]
  Output dimensions: $(256 - 5 + 1)x(256 - 5 + 1)x64 = 252 \times 252 \times 64$
- Second Convolutional Layer (128 filters): [1 mark]
  Output dimensions: $(252 - 5 + 1)x(252 - 5 + 1)x64 = 248 \times 248 \times 128$

- Max-Pooling Layer (2x2): [1 mark]
  Output dimensions: $248/2x248/2x128 = 124 \times 124 \times 128$

(b)
- Number of Parameters per Filter will remain the same, since the filter size (5x5) and the number of input channels (3) remain the same. [0.5 mark]
- The number of filters in each layer (64 and 128) remains the same. [0.5 mark]

(c) Number of Parameters = (Filter Size * Number of Input Channels+1) * Number of Filters = $(5 \times 5 \times 64 + 1) \times 128 = 204,928$ parameters. [1 mark]

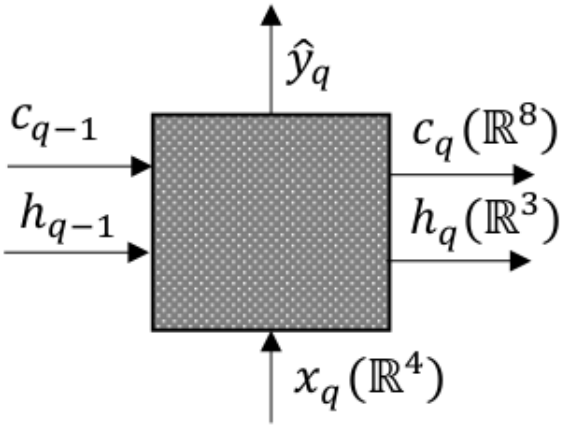
(d) 1 mark for each layer. If sequential model not declared and created give 0 marks for the entire answer.

```
# Define the CNN model
model = models.Sequential()
# First Convolutional Layer
model.add(layers.Conv2D(64, (5, 5), activation='relu',
input_shape=(256, 256, 3)))
# Second Convolutional Layer
model.add(layers.Conv2D(128, (5, 5), activation='relu'))
# Max pooling Layer
model.add(layers.MaxPooling2D((2, 2)))
```

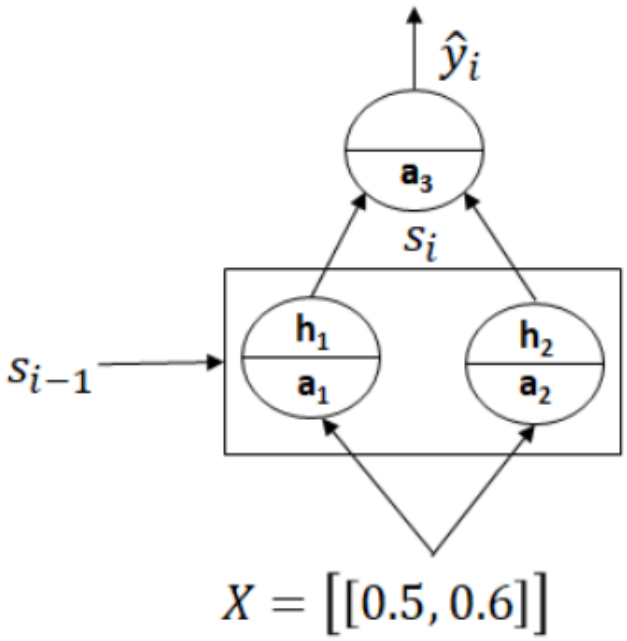
(e) 3 marks for correct equations and answers. No step marking.

$$\begin{aligned} \text{equations:} \quad & 3x + 3y = 12 \\ & 3y - 3z = -11 \\ & 3x - 3y = 6 \\ \text{solving:} \quad & x = 3 \\ & y = 1 \\ & z = 14/3 \end{aligned}$$

2. (a) A data science engineer proposed an LSTM network (shown below) for a deep learning problem. Calculate the parameters to be learnt in it. The dimensions are also shown in the parentheses. Output is one dimensional. [5]




(b) A RNN is shown below with one RNN layer of two tanh neurons and a fully connected output layer with one sigmoid neuron. A record of input with one time-step and two input features is processed with it. Calculate the output of the network assuming all weights and biases are initialized with value 0.2 and previous state as [0.3, 0.4]. The arrows shown in the diagram are representative. [5]




(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)
- Input ($x_t$): $R^4$
- Hidden state ($h_t$): $R^3$
- Cell state ($c_t$): $R^8$
- Output ($y_t$): $R^1$
- Input Gate ($i_t$)

  - Weight matrix for input gate ($W_{xi}$): $R^{4\times3}$
  - Weight matrix for hidden state ($W_{hi}$): $R^{3\times3}$
  - Bias for input gate ($b_i$): $R^3$
- Forget Gate ($f_t$)
  - Weight matrix for input gate ($W_{xf}$): $R^{4\times3}$
  - Weight matrix for hidden state ($W_{hf}$): $R^{3\times3}$
  - Bias for forget gate ($b_f$): $R^3$
- Output Gate ($f_t$)
  - Weight matrix for input gate ($W_{xo}$): $R^{4\times3}$
  - Weight matrix for hidden state ($W_{ho}$): $R^{3\times3}$
  - Bias for Output gate ($b_o$): $R^3$
- Cell State ($c_t$)
  - Weight matrix for input gate($W_{xc}$): $R^{4\times8}$
  - Weight matrix for hidden state ($W_{hc}$): $R^{3\times8}$
  - Bias for cell state ($b_c$): $R^8$
- Output ($y_t$)
  - Weight matrix for output ($W_{hy}$): $R^{3\times1}$
  - Bias for output ($b_y$): $R^1$
- Total= $12+9+3+12+9+3+12+9+3+32+24+8+3+1 = 140$ parameters

(b) Two tanh neurons in the RNN layer. One sigmoid neuron in the output layer. Input features: [0.5, 0.6] Initial state: [0.3, 0.4]. All weights and biases are initialized with a value of 0.2. $W_{hh} = [0.2, 0.2]$; ($W_{xh} = [0.2, 0.2]; b_h = [0.2, 0.2]$

- Calculate the Hidden State (RNN Layer)

$$
\begin{aligned}
h_t &= \tanh(W_{hh} \times h_{t-1} + W_{xh} \times x_t + b_h) \\
h_1 &= \tanh([0.2, 0.2] \times [0.3, 0.4] + [0.2, 0.2] \times [0.5, 0.6] + [0.2, 0.2]) \\
h_1 &= \tanh([0.06, 0.08] + [0.1, 0.12] + [0.2, 0.2]) \\
h_1 &= [\tanh(0.06), \tanh(0.08)] \\
h_1 &\approx [0.059964, 0.079995]
\end{aligned}
$$

- Calculate the Output (Output Layer)

$$
\begin{aligned}
y &= \sigma(W_{hy} \times h_t + b_y) \\
y &= \sigma([0.2, 0.2] \times [0.059964, 0.079995] + 0.2) \\
&= \sigma(0.025993 + 0.2) \approx 0.556275
\end{aligned}
$$

3. (a) Consider a 1D CNN model for time series forecasting with a sequence length of 100. The model has a single convolutional layer with 64 filters, each of size 3, followed by a fully connected layer with 128 neurons. Batch normalization is applied after convolution layer. Calculate the total number of trainable parameters and non-trainable parameters in the model. [3]

   (b) Consider a Deep Neural Network with 3 hidden layers for time series forecasting. The input sequence has a length of 50, and the hidden layers have 64, 128, and 256 neurons, respectively. Calculate the total number of trainable parameters in the model. Assume the output sequence length. [3]

   (c) A data scientist wants to build a Deep Learning model using a single LSTM cell for forecasting a time-series. The training dataset has several records with 15 time steps each. Each time step consists of three-feature normalized numeric data. If total learnable parameters in this model are 180, what is the dimensionality of the short state? [3]
   (Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)
- Convolutional Layer: $64\,filters \times (3\,weights + 1\,bias) = 256$ parameters.
- Batch Normalization
  Trainable parameters: 64 filters × (2 parameters) = 128 trainable parameters
  Non-trainable parameters: 64 filters × (2 parameters) = 128 non-trainable
- Fully connected layer: $128 neurons \times (6400 weights + 1 bias) = 819,328$ trainable parameters.
- Total Trainable Parameters = 256 (Convolution) + 128 (Batch Norm) + 819, 328 (Fully Connected) = 819, 712
  Total Non-Trainable Parameters = 128 (Batch Norm)

(b)
- Input Layer: 0 trainable parameters
- First hidden layer: $50 \times 64 + 64 = 326450 \times 64 + 64 = 3264$ parameters
- Second hidden layer: $(64 \times 128) + 128 = 8320(64 \times 128) + 128 = 8320$ parameters
- Third hidden layer: $(128 \times 256) + 256 = 33024(128 \times 256) + 256 = 33024$ parameters
- Output Layer: Assume it has 1 neuron $(256 \times 1) + 1 = 257(256 \times 1) + 1 = 257$ parameters
- Total = 3264 + 8320 + 33024 + 257 = 44705 parameters

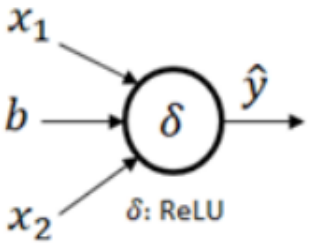
(c) Input dimension (D) is 3.
Weights for Input have the shape (D, H). Weights for Hidden State have the shape (H, H). Biases (b) has the shape (H,1).For each gate, $(D * H) + (H * H) + H$ parameters. For all gates and the cell state, there are $4 * [(D * H) + (H * H) + H]$

$$\begin{aligned} 4 * [(D * H) + (H * H) + H] &= 180 \\ 4 * [(3 * H) + (H * H) + H] &= 180 \\ 3H^2 + 7H - 45 &= 0 \\ H = \frac{-b + \sqrt{b^2 - 4ac}}{2a} &= 2 \end{aligned}$$

Dimensionality of the hidden state (H) is approximately 2.

4. (a) The training data $(x_1, y_1) = (3.5, 0.5)$ for a single Sigmoid neuron and initial values of $w = -2.0, b = -2.0, \eta = 0.10, \beta_1 = 0.90, \beta_2 = 0.99, \epsilon = 1e-8, s_W = 0$ and $s_B = 0$ are provided. Showing all the calculations, find out the values of $w, b, s_W, s_B, r_W, r_B$ after one iteration of Adam. Use BCE as loss function. Apply bias-correction. [8]

   (b) For the training data $[x_1, x_2, y] = [0.65, 0.75, 2.50]$, an engineer selected an initial set of parameters for the single neuron as shown below. Assuming SSE loss function, evaluate his selection of parameters mathematically with proper calculations and reasoning. [3]




| $w_1$ | $w_2$ | b | η |
|---|---|---|---|
| 0.30 | 0.50 | 1.75 | 2.00 |

(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)

- Forward Pass and Calculate Loss

$$\hat{y} = \sigma(wx + b) = \sigma(-2.0 \times 3.5 - 2.0) \approx 0.0180$$
$$L = -[y \times \log(\hat{y}) + (1 - y) \times \log(1 - \hat{y})$$
$$L = -[0.5 \times \log(0.0180) + (1 - 0.5) \times \log(1 - 0.0180)] \approx 4.0076$$

- Calculate Gradients

$$\nabla_w L = (\hat{y} - y) \times x = (0.0180 - 0.5) \times 3.5 \approx -0.493$$
$$\nabla_b L = \hat{y} - y = 0.0180 - 0.5 \approx -0.482$$

- Update First Moments

$$s_W = \beta_1 \times s_W + (1 - \beta_1) \times \nabla_w L = 0.90 \times 0 - 0.10 \times (-0.493) \approx 0.04437$$
$$s_B = \beta_1 \times s_B + (1 - \beta_1) \times \nabla_b L = 0.90 \times 0 - 0.10 \times (-0.482) \approx 0.04382$$

- Update Second Moments

$$r_W = \beta_2 \times r_W + (1 - \beta_2) \times (\nabla_w L)^2 = 0.999 \times 0 - 0.001 \times (-0.493)^2 \approx 0.0001216$$
$$r_B = \beta_2 \times r_B + (1 - \beta_2) \times (\nabla_b L)^2 = 0.999 \times 0 - 0.001 \times (-0.482)^2 \approx 0.0001168$$

- Corrected First Moments

$$\hat{s}_W = \frac{s_W}{1 - \beta_1^1} \approx \frac{0.04437}{1 - 0.90^1} \approx 0.4437$$
$$\hat{s}_B = \frac{s_B}{1 - \beta_1^1} \approx \frac{0.04382}{1 - 0.90^1} \approx 0.4382$$

- Corrected Second Moments

$$\hat{r}_W = \frac{r_W}{1 - \beta_2^1} \approx \frac{0.0001216}{1 - 0.999^1} \approx 0.0608$$
$$\hat{r}_B = \frac{r_B}{1 - \beta_2^1} \approx \frac{0.0001168}{1 - 0.999^1} \approx 0.0584$$

- Update Weights and Biases

$$w = w - \frac{\eta}{\sqrt{\hat{r}_W} + \epsilon} \times \hat{s}_W = -2.0 - \frac{0.10}{\sqrt{0.0608} + 1e-8} \times 0.4437 \approx -2.081$$
$$b = b - \frac{\eta}{\sqrt{\hat{r}_B} + \epsilon} \times \hat{s}_B = -2.0 - \frac{0.10}{\sqrt{0.0584} + 1e-8} \times 0.4382 \approx -2.079$$

(b) Training data: $[x_1, x_2, y] = [0.65, 0.75, 2.50]$ Initial parameters: $w_1 = 0.3$, $w_2 = 0.5$, $b = 1.75$ Learning rate: $\eta = 2$

- Calculate the Predicted Output

$$\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$$
$$\hat{y} = \sigma(0.3 \times 0.65 + 0.5 \times 0.75 + 1.75) = \sigma(2.32) \approx 0.9106$$

- Calculate the Sum of Squared Errors (SSE) loss

$$\text{SSE} = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(0.9106 - 2.50)^2 = \frac{1}{2}(-1.5894)^2 = \frac{1}{2} \times 2.5245 = 1.2623$$

To improve the model's performance, the engineer may need to adjust the weights and bias using a learning algorithm (e.g., gradient descent) to minimize the SSE loss.

# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

**M. Tech. in AI & ML**

**II Semester 2022-2023**

**End-Semester Test**
**(EC3 - Regular)**

| | |
|---|---|
| Course Number | AIMLCZG51 |
| Course Name | DEEP NEURAL NETWORK |
| Nature of Exam | Open Book |
| Weight-age for grading | 40 |
| Duration | 2.5 hrs |
| Date of Exam | |

| | |
|---|---|
| * Pages | 4 |
| * Questions | 4 |

---

1. (a) Consider a CNN architecture with an input image of size $256 \times 256 \times 3$. The architecture consists of two convolutional layers with 64 and 128 filters of size $5 \times 5$ respectively, followed by a max-pooling layer of size $2 \times 2$. Calculate the output dimensions after each layer. [3]

   (b) If the input image size is changed to $128 \times 128 \times 3$, how will it affect the number of trainable parameters? [1]

   (c) Compute the trainable parameters are there in the second convolutional layer for part (a). [1]

   (d) Write Python code to implement for part (a) architecture using TensorFlow Keras. Add one dense layer and an output layer for binary classification. [3]

   (e) I represents the top left corner pixel values of a much larger image (not shown in entirety). F is a filter that convolves over this image with stride 1. O is the result of this convolution, where only 3 resulting pixels at top left are shown. Importantly, there is no zero padding, which implies that each pixel in O is the dot product resulting from the filter being placed at the corresponding position on the image in I. Find the values of x, y and z. [3]

$$I = \begin{pmatrix} 3 & 0 & 3 & -3 & 0 & \dots \\ -3 & 2 & 0 & 3 & -2 & \dots \\ -5 & 0 & 3 & -2 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} \quad F = \begin{pmatrix} x & y & z \\ 1 & 0 & 2 \\ -1 & 1 & 0 \end{pmatrix} \quad O = \begin{pmatrix} 4 & 0 & -3 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

Rubrics and one solution:

(a)
- First Convolutional Layer (64 filters): [1 mark]
  Output dimensions: $(256 - 5 + 1)x(256 - 5 + 1)x64 = 252 \times 252 \times 64$
- Second Convolutional Layer (128 filters): [1 mark]
  Output dimensions: $(252 - 5 + 1)x(252 - 5 + 1)x64 = 248 \times 248 \times 128$

- Max-Pooling Layer (2x2): [1 mark]
  Output dimensions: $248/2x248/2x128 = 124 \times 124 \times 128$

(b)
- Number of Parameters per Filter will remain the same, since the filter size (5x5) and the number of input channels (3) remain the same. [0.5 mark]
- The number of filters in each layer (64 and 128) remains the same. [0.5 mark]

(c) Number of Parameters = (Filter Size * Number of Input Channels+1) * Number of Filters = $(5 \times 5 \times 64 + 1) \times 128 = 204,928$ parameters. [1 mark]

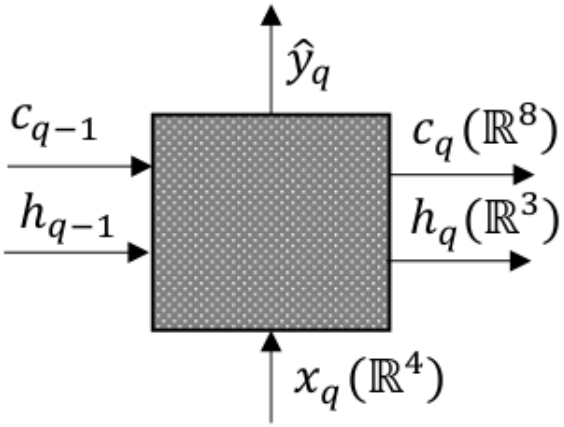
(d) 1 mark for each layer. If sequential model not declared and created give 0 marks for the entire answer.

```
# Define the CNN model
model = models.Sequential()
# First Convolutional Layer
model.add(layers.Conv2D(64, (5, 5), activation='relu',
input_shape=(256, 256, 3)))
# Second Convolutional Layer
model.add(layers.Conv2D(128, (5, 5), activation='relu'))
# Max pooling Layer
model.add(layers.MaxPooling2D((2, 2)))
```

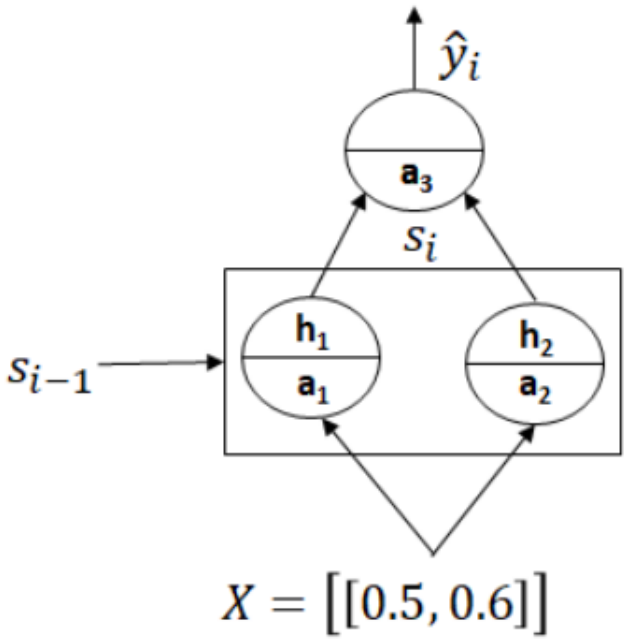
(e) 3 marks for correct equations and answers. No step marking.

$$\begin{aligned} \text{equations:} \quad & 3x + 3y = 12 \\ & 3y - 3z = -11 \\ & 3x - 3y = 6 \\ \text{solving:} \quad & x = 3 \\ & y = 1 \\ & z = 14/3 \end{aligned}$$

2. (a) A data science engineer proposed an LSTM network (shown below) for a deep learning problem. Calculate the parameters to be learnt in it. The dimensions are also shown in the parentheses. Output is one dimensional. [5]




(b) A RNN is shown below with one RNN layer of two tanh neurons and a fully connected output layer with one sigmoid neuron. A record of input with one time-step and two input features is processed with it. Calculate the output of the network assuming all weights and biases are initialized with value 0.2 and previous state as [0.3, 0.4]. The arrows shown in the diagram are representative. [5]




(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)
- Input ($x_t$): $R^4$
- Hidden state ($h_t$): $R^3$
- Cell state ($c_t$): $R^8$
- Output ($y_t$): $R^1$
- Input Gate ($i_t$)

  - Weight matrix for input gate ($W_{xi}$): $R^{4\times3}$
  - Weight matrix for hidden state ($W_{hi}$): $R^{3\times3}$
  - Bias for input gate ($b_i$): $R^3$
- Forget Gate ($f_t$)
  - Weight matrix for input gate ($W_{xf}$): $R^{4\times3}$
  - Weight matrix for hidden state ($W_{hf}$): $R^{3\times3}$
  - Bias for forget gate ($b_f$): $R^3$
- Output Gate ($f_t$)
  - Weight matrix for input gate ($W_{xo}$): $R^{4\times3}$
  - Weight matrix for hidden state ($W_{ho}$): $R^{3\times3}$
  - Bias for Output gate ($b_o$): $R^3$
- Cell State ($c_t$)
  - Weight matrix for input gate($W_{xc}$): $R^{4\times8}$
  - Weight matrix for hidden state ($W_{hc}$): $R^{3\times8}$
  - Bias for cell state ($b_c$): $R^8$
- Output ($y_t$)
  - Weight matrix for output ($W_{hy}$): $R^{3\times1}$
  - Bias for output ($b_y$): $R^1$
- Total= $12+9+3+12+9+3+12+9+3+32+24+8+3+1 = 140$ parameters

(b) Two tanh neurons in the RNN layer. One sigmoid neuron in the output layer. Input features: [0.5, 0.6] Initial state: [0.3, 0.4]. All weights and biases are initialized with a value of 0.2. $W_{hh} = [0.2, 0.2]$; ($W_{xh} = [0.2, 0.2]; b_h = [0.2, 0.2]$

- Calculate the Hidden State (RNN Layer)

$$
\begin{aligned}
h_t &= \tanh(W_{hh} \times h_{t-1} + W_{xh} \times x_t + b_h) \\
h_1 &= \tanh([0.2, 0.2] \times [0.3, 0.4] + [0.2, 0.2] \times [0.5, 0.6] + [0.2, 0.2]) \\
h_1 &= \tanh([0.06, 0.08] + [0.1, 0.12] + [0.2, 0.2]) \\
h_1 &= [\tanh(0.06), \tanh(0.08)] \\
h_1 &\approx [0.059964, 0.079995]
\end{aligned}
$$

- Calculate the Output (Output Layer)

$$
\begin{aligned}
y &= \sigma(W_{hy} \times h_t + b_y) \\
y &= \sigma([0.2, 0.2] \times [0.059964, 0.079995] + 0.2) \\
&= \sigma(0.025993 + 0.2) \approx 0.556275
\end{aligned}
$$

3. (a) Consider a 1D CNN model for time series forecasting with a sequence length of 100. The model has a single convolutional layer with 64 filters, each of size 3, followed by a fully connected layer with 128 neurons. Batch normalization is applied after convolution layer. Calculate the total number of trainable parameters and non-trainable parameters in the model. [3]

   (b) Consider a Deep Neural Network with 3 hidden layers for time series forecasting. The input sequence has a length of 50, and the hidden layers have 64, 128, and 256 neurons, respectively. Calculate the total number of trainable parameters in the model. Assume the output sequence length. [3]

   (c) A data scientist wants to build a Deep Learning model using a single LSTM cell for forecasting a time-series. The training dataset has several records with 15 time steps each. Each time step consists of three-feature normalized numeric data. If total learnable parameters in this model are 180, what is the dimensionality of the short state? [3]
   (Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)
- Convolutional Layer: $64\,filters \times (3\,weights + 1\,bias) = 256$ parameters.
- Batch Normalization
  Trainable parameters: 64 filters × (2 parameters) = 128 trainable parameters
  Non-trainable parameters: 64 filters × (2 parameters) = 128 non-trainable
- Fully connected layer: $128 neurons \times (6400 weights + 1 bias) = 819,328$ trainable parameters.
- Total Trainable Parameters = 256 (Convolution) + 128 (Batch Norm) + 819, 328 (Fully Connected) = 819, 712
  Total Non-Trainable Parameters = 128 (Batch Norm)

(b)
- Input Layer: 0 trainable parameters
- First hidden layer: $50 \times 64 + 64 = 326450 \times 64 + 64 = 3264$ parameters
- Second hidden layer: $(64 \times 128) + 128 = 8320(64 \times 128) + 128 = 8320$ parameters
- Third hidden layer: $(128 \times 256) + 256 = 33024(128 \times 256) + 256 = 33024$ parameters
- Output Layer: Assume it has 1 neuron $(256 \times 1) + 1 = 257(256 \times 1) + 1 = 257$ parameters
- Total = 3264 + 8320 + 33024 + 257 = 44705 parameters

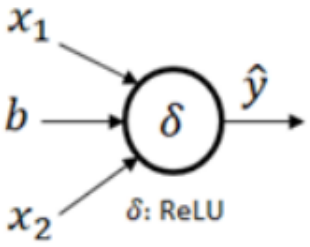
(c) Input dimension (D) is 3.
Weights for Input have the shape (D, H). Weights for Hidden State have the shape (H, H). Biases (b) has the shape (H,1).For each gate, $(D * H) + (H * H) + H$ parameters. For all gates and the cell state, there are $4 * [(D * H) + (H * H) + H]$

$$\begin{aligned} 4 * [(D * H) + (H * H) + H] &= 180 \\ 4 * [(3 * H) + (H * H) + H] &= 180 \\ 3H^2 + 7H - 45 &= 0 \\ H = \frac{-b + \sqrt{b^2 - 4ac}}{2a} &= 2 \end{aligned}$$

Dimensionality of the hidden state (H) is approximately 2.

4. (a) The training data $(x_1, y_1) = (3.5, 0.5)$ for a single Sigmoid neuron and initial values of $w = -2.0, b = -2.0, \eta = 0.10, \beta_1 = 0.90, \beta_2 = 0.99, \epsilon = 1e-8, s_W = 0$ and $s_B = 0$ are provided. Showing all the calculations, find out the values of $w, b, s_W, s_B, r_W, r_B$ after one iteration of Adam. Use BCE as loss function. Apply bias-correction. [8]

   (b) For the training data $[x_1, x_2, y] = [0.65, 0.75, 2.50]$, an engineer selected an initial set of parameters for the single neuron as shown below. Assuming SSE loss function, evaluate his selection of parameters mathematically with proper calculations and reasoning. [3]




| $w_1$ | $w_2$ | b | η |
|---|---|---|---|
| 0.30 | 0.50 | 1.75 | 2.00 |

(Note: Your answer must be supported by the calculation / justification / reasoning for all parts of this question. There are no marks for the direct answer or the output of any tool or program or for direct screenshots or images of textbook pages.)

(a)

- Forward Pass and Calculate Loss

$$\hat{y} = \sigma(wx + b) = \sigma(-2.0 \times 3.5 - 2.0) \approx 0.0180$$
$$L = -[y \times \log(\hat{y}) + (1 - y) \times \log(1 - \hat{y})$$
$$L = -[0.5 \times \log(0.0180) + (1 - 0.5) \times \log(1 - 0.0180)] \approx 4.0076$$

- Calculate Gradients

$$\nabla_w L = (\hat{y} - y) \times x = (0.0180 - 0.5) \times 3.5 \approx -0.493$$
$$\nabla_b L = \hat{y} - y = 0.0180 - 0.5 \approx -0.482$$

- Update First Moments

$$s_W = \beta_1 \times s_W + (1 - \beta_1) \times \nabla_w L = 0.90 \times 0 - 0.10 \times (-0.493) \approx 0.04437$$
$$s_B = \beta_1 \times s_B + (1 - \beta_1) \times \nabla_b L = 0.90 \times 0 - 0.10 \times (-0.482) \approx 0.04382$$

- Update Second Moments

$$r_W = \beta_2 \times r_W + (1 - \beta_2) \times (\nabla_w L)^2 = 0.999 \times 0 - 0.001 \times (-0.493)^2 \approx 0.0001216$$
$$r_B = \beta_2 \times r_B + (1 - \beta_2) \times (\nabla_b L)^2 = 0.999 \times 0 - 0.001 \times (-0.482)^2 \approx 0.0001168$$

- Corrected First Moments

$$\hat{s}_W = \frac{s_W}{1 - \beta_1^1} \approx \frac{0.04437}{1 - 0.90^1} \approx 0.4437$$
$$\hat{s}_B = \frac{s_B}{1 - \beta_1^1} \approx \frac{0.04382}{1 - 0.90^1} \approx 0.4382$$

- Corrected Second Moments

$$\hat{r}_W = \frac{r_W}{1 - \beta_2^1} \approx \frac{0.0001216}{1 - 0.999^1} \approx 0.0608$$
$$\hat{r}_B = \frac{r_B}{1 - \beta_2^1} \approx \frac{0.0001168}{1 - 0.999^1} \approx 0.0584$$

- Update Weights and Biases

$$w = w - \frac{\eta}{\sqrt{\hat{r}_W} + \epsilon} \times \hat{s}_W = -2.0 - \frac{0.10}{\sqrt{0.0608} + 1e-8} \times 0.4437 \approx -2.081$$
$$b = b - \frac{\eta}{\sqrt{\hat{r}_B} + \epsilon} \times \hat{s}_B = -2.0 - \frac{0.10}{\sqrt{0.0584} + 1e-8} \times 0.4382 \approx -2.079$$

(b) Training data: $[x_1, x_2, y] = [0.65, 0.75, 2.50]$ Initial parameters: $w_1 = 0.3$, $w_2 = 0.5$, $b = 1.75$ Learning rate: $\eta = 2$

- Calculate the Predicted Output

$$\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$$
$$\hat{y} = \sigma(0.3 \times 0.65 + 0.5 \times 0.75 + 1.75) = \sigma(2.32) \approx 0.9106$$

- Calculate the Sum of Squared Errors (SSE) loss

$$\text{SSE} = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(0.9106 - 2.50)^2 = \frac{1}{2}(-1.5894)^2 = \frac{1}{2} \times 2.5245 = 1.2623$$

To improve the model's performance, the engineer may need to adjust the weights and bias using a learning algorithm (e.g., gradient descent) to minimize the SSE loss.

# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

**M. Tech. in AIML**

**II Semester 2022-2023**

**Mid-Semester Test**
**(EC2 - Makeup)**

| | |
|---|---|
| Course Number | AIMLCZG511 |
| Course Name | Deep Neural Networks |
| Nature of Exam | Open Book |
| Weight-age for grading | 30 |
| Duration | 2 hrs |
| Date of Exam | |

| | |
|---|---|
| * Pages | 3 |
| * Questions | 5 |

---

1. A neural network designed using Tensorflow Keras is given below. Students are instructed to type the answers in the textbox of the portal.

```
net = tf.keras.layers.Sequential()
net.add(
        tf.keras.layers. InputLayer(input_shape = ((256*256*3),)),
        tf.keras.layers.Dense(1048, activation='relu'),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.4),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.6),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(16, activation='relu'),
        tf.keras.layers.Dense(8, activation='softmax'))
```

   (a) What is the objective of the neural network? What is the input given to the network? What is the expected output? How deep and wide is the network. [2]

   (b) Justify the choice activation function in output layer. Instead of Relu activation function, justify the choice of using Tanh activation function. [1]

   (c) Two dropout statements are added in the code. How many additional parameters are learned because of this? If drop out is added after the last statement in the given code, how will it affect the network? [1]

   (d) Write the code snippet for adding the optimizer of your choice. Justify the choice of the optimizer. Assume any other relevant information. [1]

   (e) What will the following code snippet do to the network. [1]

```
cb = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=7)
history = net.fit(epochs=200, batch_size=32, callbacks=[cb])
```

2. (a) Figure 1 below plots the loss when batch gradient descent is used for training. Which optimizers plots the loss in figure 2 and 3. [1]

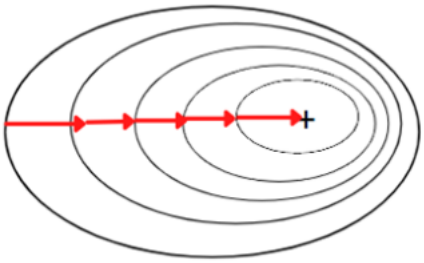


Figure 1

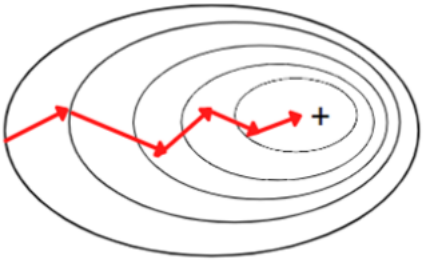


Figure 2

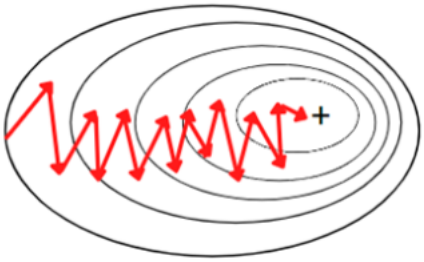


Figure 3

Figure 1

(b) In figure 2, assume that the learning rate used was 0.5. Redraw the plot to show the effect of increasing the learning rate to 2 and decreasing the learning rate to 0.01. [2]

(c) You are given a simple neural network with a single hidden layer containing two neurons, and an output layer containing one neuron. All neurons use the sigmoid activation function. The weights and biases of the network are as follows: [3]

- Weights from input to hidden layer: $w_1 = 0.5, w_2 = -0.6$
- Biases in hidden layer: b1 = 0.1, b2 = -0.2
- Weights from hidden layer to output: w3 = 0.7, w4 = -0.8
- Bias in output layer: b3 = 0.3

Given an input x = 0.75, calculate the output of the network and mean squared loss if the desired output is 1.25. Use the sigmoid activation function. What will be the effect in the loss if the loss function used is

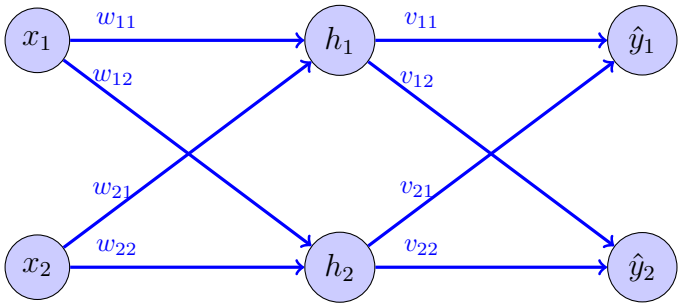
$$L(w, b) = \frac{1}{2}(d - \hat{y})^2 + \lambda||w^2||$$

3. (a) Consider a two input XNOR gate and simulate a perceptron algorithm for it, where, learning rate=0.02 and threshold = 0.2. [2]

(b) Represent using a multilayer neural network $A \odot B \odot C \odot D \odot E$ where $\odot$ represents XNOR. What will be the optimal depth and width for this network. [3]

(c) You'd like to train a fully-connected neural network with 7 hidden layers, each with 8 hidden units. The input is 30-dimensional and the output is a binary. What is the total number of trainable parameters in your network? [1]

4. Consider the following network structure. You can assume the initial weights. Assume bias to be zero for easier computations. Given that $< x_1, x_2, \hat{y}_1, \hat{y}_2 >=< 1, 1, 0, 1 >$ where $\hat{y}$ is the target. Assume $\beta = 0.9$ and $\eta = 0.01$.




(a) Compute the forward propagation and generate the output. Use Relu for hidden layers and Sigmoid activation function for output layer. [2]

(b) Compute the Softmax loss function for both outputs. [1]

(c) Let the initial weights that assumed be the weights [at time (t-1). Compute the weights $v_{21}$, $w_{12}$ and $w_{22}$ at time $t$ using SGD. [1.5]

(d) Let the weight at time $t$ be the ones computed in part (c). Compute the weights $v_{21}$, $w_{12}$ and $w_{22}$ at $(t+1)$ when momentum is used. [1.5]

5. (a) Represent the function

$$f(x, y) = x^4 - 32x^2 + y^4 - 18y^2$$

using a computation graph. Evaluating this function at x = 2, and y = 2. Find the first derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using computation graph through backpropagation. [3]

(b) Compute the Hessian of the function

$$f(x, y) = x^4 - 32x^2 + y^4 - 18y^2$$

You are given two point (0,0) and (4,3). Among these two points, which points are the local minima or local maxima or both. [3]

# Solution WS21 - Endterm

Introduction to Deep Learning (Technische Universität München)

Chair of Visual Computing & Artificial Intelligence
Department of Informatics
Technical University of Munich

Esolution
Place student sticker here

**Note:**

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

# Introduction to Deep Learning

| | | | |
|---|---|---|---|
| **Exam:** | IN2346 / endterm | **Date:** | Sunday 21st February, 2021 |
| **Examiner:** | Prof. Dr. Matthias Nießner | **Time:** | 08:00 – 09:30 |

| | P 1 | P 2 | P 3 | P 4 | P 5 | P 6 | P 7 | P 8 | P 9 | P 10 | P 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| I | | | | | | | | | | | |



## Working instructions

- This exam consists of **22 pages** with a total of **11 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 90 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources: None
- This is the blackened exam. Please fill out the boxes here.

Left room from _____ to _____ / Early submission at _____

## Problem 1 Multiple Choice (16 credits)

- For all multiple choice questions any number of answers, i.e. either zero (!), one or multiple answers can be correct.
- For each question, you'll receive 2 points if all boxes are answered correctly (i.e. correct answers are checked, wrong answers are not checked) and 0 otherwise.

**How to Check a Box Using Pen and Paper:**

- Please **cross** the respective box: ☒ (interpreted as **checked**)
- If you change your mind, please **fill** the box: ■ (interpreted as **not checked**)
- If you change your mind again, put a **cross** next to the filled box.

a) How will the Validation Loss behave, if you shuffle the validation set, compared to, if you don't shuffle it?

- ☐ 1) The validation loss will be higher if shuffled.
- ☐ 2) The validation loss will be lower if shuffled.
- ☒ 3) The validation loss will be the same if shuffled.
- ☐ 4) It's not possible to predict how shuffling will affect the Validation Loss.

b) Which one of the following layers have the same train and test time behavior?

- ☐ 1) Batch Normalization.
- ☒ 2) Linear Layer.
- ☐ 3) Dropout.
- ☒ 4) Sigmoid.

c) Which of the following layers does not have trainable parameters?

- ☐ 1) Parametric ReLU.
- ☒ 2) Global Average Pooling.
- ☐ 3) Convolution.
- ☒ 4) Dropout.

d) The following is true about the L1 and L2 parameter norm regularization techniques:

- ☒ 1) They introduce additional residuals in the cost function.
- ☒ 2) The L2 norm encourages a mean distribution over the weight values.
- ☐ 3) They improve training accuracy.
- ☐ 4) They aim to reduce the overall cost of the optimization.

e) Which of the following statements are true regarding second order derivative based optimization techniques

- ☒ 1) They converge to the minimum in lesser number of iterations when compared to first order techniques.
- ☐ 2) Each iteration step of second order technique are usually more faster than the first order technique.
- ☐ 3) RMS Prop is a second order optimization technique.
- ☒ 4) Each iteration step of second order technique are usually slower than the first order technique.

f) You start training your Neural Network but the train loss is almost completely flat and not decreasing. What could be the cause?

- [ ] 1) Class distribution is too even in the dataset.
- [x] 2) Bad initialization.
- [x] 3) Learning rate is too small.
- [ ] 4) Regularization strength is too small.

g) What can 1 × 1 convolutions be used for?

- [ ] 1) To perform feature selection.
- [x] 2) To reduce the number of channels before a costly operation.
- [x] 3) To make a network more complex when coupled with non-linearities.
- [x] 4) To replace fully-connected layers in order to make a network fully- convolutional.

h) What is true about Dropout?

- [x] 1) When using dropout, our network can be seen as an ensemble of multiple independent smaller networks.
- [ ] 2) Dropout makes training generally faster.
- [x] 3) Dropout acts as regularization.
- [ ] 4) Dropout can also be applied at test time, but the output has to be scaled by the number of training samples.

y **Problem 2** **Case Study: Road Signs (7 credits)**

Recently one of your friends wants to get a driving license in Germany, and since it is still in lock-down period, he would like to start to study the theoretical aspect first. He tries to memorize the meaning of each road signs, and after a while he feels tried and wants to have some fun with it, i.e., develop a deep learning model to classify the road signs and compete against it with his own memory to see whether he or the model can reach higher accuracy. So he approaches you for your deep learning expertise. He gets a hand-labelled dataset which has around 5,000 RGB images that contains both road signs taken at daytime and at night as well as 20 classes of different road signs.

0 1 a) Before building up the model, you need to split the dataset. How would you do this?

Train + Validation + Test set with 60%, 20%, 20% or 80%, 10%, 10%. (0.5 point for resonable percentages)
Important: answer should mention that the split needs to cover both daytime and nighttime data. (0.5 point)

0 1 2 b) You first trained a very simple model only on 200 samples, the train loss converges to a relative big value. Your friend thinks that if you train on more data and train longer, this issue can be solved. Do you agree? If not, what would you do?

No, I don't agree. (1 point)
The simple model has high training error which means the capacity of the model is too small such that this is a bias problem. Training on larger dataset will not help with this situation.
In this case, it would be better by adding more learnable parameters to increase the capacity of the model.
(1 point for increase model capacity, if only mention other possible solutions then 0.5 point)

0 1 2 c) After you solved the high training loss problem, you are training your model from scratch and getting a reasonable result. However, you would like to search for better hyperparameters. Name two hyperparameter search strategies and shortly discuss which procedure you would use to combine those.

Grid Search, Random Search. (0.5 point for each method)
Random search first to find reasonable region, then use grid search to refine. (1 point, if the order is the other way around then get 0.5 point)

d) Even after all this optimization, you accept that the dataset itself is not big enough. Here are some more data examples. To make your model generalize better, you decide to perform data augmentation.

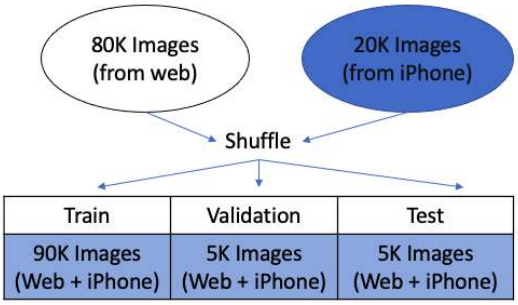
0 1 2

1. Explain how you could augment the data using mirroring (1p).
2. Give 3 examples of further data augmentation techniques that you can use in addition (1p).

It's important to note that some classes are mirror-inverted (e.g. left- and right-only-signs). Therefore, we must not only mirror the images X, but also need to adjust the label y. ()
Other types of regularization are: adding noise, small random rotations, translations, etc.

## Problem 3 Data Analysis (8 credits)

You want to build an iPhone app that can classify dog images taken from the iPhone-camera. To collect data, you walk through Munich and take pictures of dogs ( as well as some non-dog-objects for the negative class). This way you can collect 20.000 samples. However, according to your experience you need much more images to train a good model. You get the idea to simply write a Python script to download 80.000 images with a similar dog/non-dog ratio from the web. You then combine both datasets, randomly shuffle the data, and split the resulting dataset into train (90%), validation (5%) and test (5%) splits.

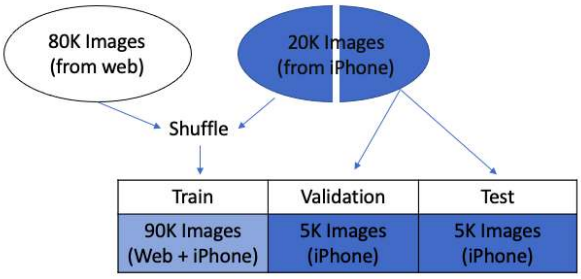



0 1 a) Why is it a bad idea, especially in terms of evaluation of your model, to first mix all the data and then create the training split from the mixed data?

The data comes from 2 different distributions (0.5pt). By the above approach, **only $\frac{1}{5}$ of the images in the validation (and test) set come from the target distribution** (iPhone images). Therefore, we won't be able to assess how well the model is actually performing in the task we're interested in (0.5pt).

Note: no points for shuffling-related answers or different ratio between web and iPhone images among training/val/test set.

0 1 2 b) You now decide to combine the datasets differently: You build a validation and test set that only consist of images from the target-distribution (5,000 images both) and mix the remaining 10,000 iPhone images with the internet-images to form the training data. Why will this approach lead to problems in the model evaluation as well?




Our validation set now has a good assessment about the performance of our model on the target data. However: Training distribution is different from the validation/test distribution. (1pt).

Therefore, if the validation error is high, we can't tell if the model performs bad because a) the model is overfitting, or b) there is a data mismatch error (1pt).

Note: 90-5-5 datasplit is not really problematic here, as we just have a simple binary classification problem. This answer didn't get any points.

c) Explain a better way to split the data by filling out the table analogous to the question b).
Hint: Think about using a 4th set to take into account the bridge between both distributions. Fill in their name, the distribution(s) they come from, and the number of samples.

0
1
2

| Train | Bridge / Train-Dev | Validation | Test |
|---|---|---|---|
| 85K Images (Web + Target) | 5K Images (Web + Target) | 5K Images (Target data only) | 5K Images (Target data only) |

The "bridge" (also: "train-dev") set is used, to evaluate whether the model is generalizing well to unseen samples or overfitting to the training data. Therefore, the data in this set comes from the same distribution as the data in the training-set. The validation- and test- set come from the target-distribution. This way we can evaluate, how well the model fits the target-distribution. (2 pt if all correct.)

"train (mixed) | val (web) | val (phone) | test (phone)": also 2 pt, even though val (web) should better be from same distr. as training to have more accurate information about overfitting.

"web | phone | phone | phone" IF mentioned that they first do pre-training on web-images and then transfer learning: also received 2 points.

"train (web) | val (web) | val (phone) | test (phone)": 1.5 pt - because not using any phone images for training.

"train (mixed) | val (mixed) | test (web) | test (phone)": 1.5 pt - not able to evaluate phone performance.

"mixed" in every set gave 0 points, as we want to train a classifier of phone images, web-data should not be in val/test.

d) You now train the network. You notice that the variance of the model is high. Fill in the table below, what sets will have a high and a low loss. Leave boxes empty for sets that aren't used during the whole training and optimization process.

0
1

| Train | Bridge / Train-Dev | Validation | Test |
|---|---|---|---|
| Low | High | High | (Not used) |

As the model doesn't generalize to unseen samples, the loss will be high for all sets except the training data (to which we are overfitting) (1pt)
Only get point if train-set has low loss and all non-train sets have high loss.

e) You now improve your model and solve the problem of high variance. However, you notice that there is a data-mismatch problem. How will the losses in this case look like? Leave boxes empty for sets that aren't used during the whole training and optimization process. (Fill in "high or "low")

0
1

| Train | Bridge / Train-Dev | Validation | Test |
|---|---|---|---|
| Low | Low | High | (Not used) |

As we could reduce the variance error (e.g. by regularization), the model performs well on unseen samples that come from the same distribution, as the training data. However, due to the data mismatch problem, it performs poorly on data from different distributions.

The grading of this subproblem depends on the student's solution to subproblem c). Only get point if for at least 3 datasets a loss was provided and datasets that have a different distribution than training set must have high loss.

Note: data mismatch *problem* implies that the loss for the other distribution from training is higher. If it would be lower (i.e., we would perform better on the target distribution in val/test than on training distribution), this would not be a problem but instead show that training with the web-data works great.

0
1

f) How could you solve the data mismatch problem?

We need to incorporate characteristics of the target distribution into the training set.

- collect more data from target distribution
- analyze the error to find out what are the difference in the datasets
- synthesize data, etc.

## Problem 4 Optimization (9 credits)

a) Write down the formula for the binary cross-entropy loss. Make sure to define all the variables in your equation.

0 1

$$\mathcal{L}(\hat{y}_i, y_i) = -(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)) \quad \text{or}$$

$$BCE = -\frac{1}{N} \sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)$$

Where $y_i$ is the binary label and $\hat{y}_i$ the prediction/score/probability of class i.
Formula: (0.5 points) (Also get the points without the minus at first place, since it's also in our slide.)
Explanation of variables: (0.5 points). (No point for: $\hat{y}_i$ is the predicted label.)

b) Why is computing the full-batch gradient often impractical during gradient descent? What do you do instead?

0 1

Impractical: update only after loop over entire data set / extremely expensive to compute / too slow / GPU limitation (0.5 points) (No point for: Only say the dataset size is too large.)
Use instead: Update over mini-batch / SGD (0.5 points)

c) What is a saddle point? What is the advantage/disadvantage of Stochastic Gradient Descent (SGD) in dealing with saddle points?

0 1 2

Saddle point - The gradient is zero (0.5 points), but it is neither a local minima nor a local maxima. (or:the gradient is zero and the function has a local maximum in one direction, but a local minimum in another direction). (0.5 points) SGD has noisier updates and can help escape from a saddle point. (1 point)
(- 0.5 point for : only say SGD helps escape from saddle point but without giving a reason.)

d) Given a learning rate of $\alpha$, gradient $\nabla$, and velocity $\nu$, write down the formula(s) for momentum update in SGD.

0 1 2

$$\nu^{k+1} = \beta \cdot \nu^k + \nabla_\Theta L(\Theta^k) \qquad \Theta^{k+1} = \Theta^k - \alpha \cdot \nu^{k+1}$$

or

$$\boldsymbol{v}^{k+1} = \beta \cdot \boldsymbol{v}^k - \alpha \cdot \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}^k\right) \qquad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \boldsymbol{v}^{k+1}$$

(1 point for each formula)
(- 0.5 points: only write $\nabla$ without the loss term, index wrong/missing)

e) What would be an advantage of a second order optimization method such as the Newton method besides taking less iteration steps? Why is it not commonly used in the context of neural networks?

0 1 2

Avoid choosing the learning rate(1P) Computing the inverse Hessian is expensive as it depends on the number of parameters (millions) / estimating the Hessian in a stochastic setting is difficult. (1P)

0
1

f) What is the difference between Newton and quasi-Newton methods, e.g., Broyden–Fletcher–Goldfarb–Shanno (BFGS)?

Quasi-Newton Methods do not compute the Hessian explicitely but find and approximation in $\mathcal{O}(n^2)$ or $\mathcal{O}(n)$. (1P)

## Problem 5 Batch Normalization (6 credits)

For an input $z = (z^{(1)}, z^{(2)}, ..., z^{(b)}) \in \mathbb{R}^b$ with $z^{(i)} \in \mathbb{R}^n$ ($b$ batch size, $n$ feature dimension) the output of a batch normalization layer $y \in \mathbb{R}^{b \times n}$ is given by

$$z_{\text{norm}} = \frac{z - \mu}{\sigma}$$

$$y = \gamma \cdot z_{\text{norm}} + \beta$$

where

$$\mu = \frac{1}{b} \sum_{i=1}^{b} z^{(i)}$$

$$\sigma = \sqrt{\frac{1}{b} \sum_{i=1}^{b} \left(z^{(i)} - \mu\right)^2}$$

Here, $\mu$ is the mean and $\sigma$ is the standard deviation of $z$. All additions and multiplications are elementwise and the trainable parameters $\gamma, \beta \in \mathbb{R}^n$ are replicated along the sample dimension.

a) Explain shortly the purpose of the learnable parameters in the batch normalization layer.

0
1

They allow the network to undo/revert the normalization. (1pt)

b) Now you want to use a neural network with batch normalization layers to build a classifier that can distinguish a cat from a dog. In one of the batch normalization layers, you forward propagate a batch of $b$ examples in your network. The input $z = [z_{(1)}^T, z_{(2)}^T, ..., z_{(b)}^T]^T$ of the batch normalization layer has shape ($b = 4, n_h = 3$), where $n_h$ represents the number of neurons in the pre-batchnorm layer:

0
1
2

$$z = \begin{bmatrix} 12 & 0 & -5 \\ 14 & 10 & 5 \\ 14 & 10 & 5 \\ 12 & 0 & -5 \end{bmatrix}$$

Calculate $\mu, \sigma$ and $z_{\text{norm}}$.

$\mu$ correct: (0.5 points)
$\sigma$ correct: (1 points)
$z_{\text{norm}}$ correct: (2 points)

$$\mu = \begin{bmatrix} 13 \\ 5 \\ 0 \end{bmatrix} \qquad \sigma = \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix} \qquad z_{\text{norm}} = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

0
1

c) Suppose the learnt parameters for scaling $\gamma = (1, 1, 1)$ and learnt parameters for offsetting $\beta = (0, -10, 10)$. What is the final result $y$ after applying the learnable parameters?

$$y = \begin{bmatrix} -1 & -11 & 9 \\ 1 & -9 & 11 \\ 1 & -9 & 11 \\ -1 & -11 & 9 \end{bmatrix}$$ (0.5 points for each correct column)

0
1

d) The above given definition of a batch normalization is applied in combination with fully-connected layers. Using a batch normalization layer in combination with a convolutional layer implies some changes to the original definition. Instead of an input dimension of $b \times n$, we have an input dimesnion of $N \times C \times H \times W$ where $N$ represents the number of samples, $C$ the number of channels and $H \times W$ the size of the input image.

What is the name of the altered layer? Explain shortly the difference to the original defined batch normalization procedure.

1. Spatial Batch Normalization (0.5 point), BatchNorm2d could be accepted as well
2. Instead of calculating the mean and variance across all input samples (0.5pt), the mean and variance will be calculated per channel of the input (0.5pt).

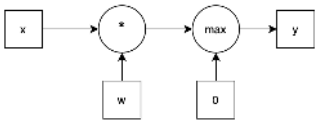
## Problem 6 Skip Connections and Computation Graph (6 credits)

Suppose you and your team want to train a very deep network, but the optimization gets more difficult as you go deeper. Thus, you decided to use skip-connections in your model. However, your team members are not convinced that skip-connections help with the optimization and they think it is a waste of time. You want to show them how skip-connections help with the gradient flow.

a) First, let's start with a simple network layer. Assume $x$, $w$ and $y$ are scalars and $*$ is the simple multiplication. Draw the computation graph of the function $y = max(w * x, 0)$ and write down the derivative **dy** $:= \frac{dy}{dx}$.

0 1 2

Computational Graph correct: 1 point
derivative correct: 1 point

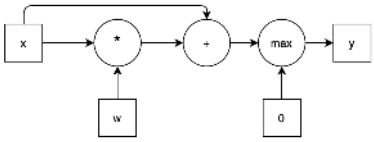



$dy/dx = 1(y > 0) * w$

b) Now, let's explore a simple skip-connection. Draw the computation graph of the operation $y = max(w * x + x, 0)$ and write down the the derivative **dy** $:= \frac{dy}{dx}$.

0 1 2

Computational Graph correct: 1 point
Derivative correct: 1 point




$dy/dx = 1(y > 0) * (w + 1)$

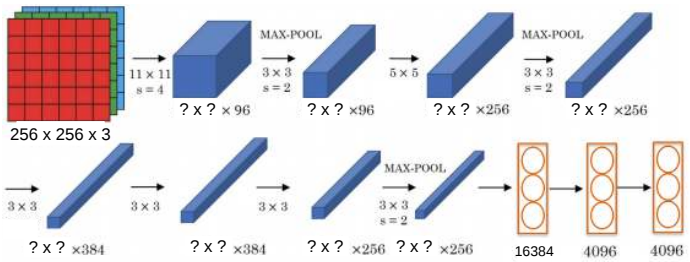
c) Evaluate both derivatives at $w = 0.001$, assuming $y > 0$. What are the values you get? Which problem did skip-connections solve?

0 1 2

a) dy/dx = 0.001 (0.5 point)
b) dy/dx = 1.001 (0.5 point)
The skip connection solved the vanishing gradient problem by allowing a larger gradient flow (1 point).

## Problem 7 Convolutional Neural Networks (7.5 credits)

Your friend needs your expertise in classifying a set of RGB Images of size 256 x 256 pixels into a total of 1000 classes. Can you help him out?
Since you learned that CNNs are great to tackle these sort of tasks, you decide to start out with the following CNN architecture.




**Notes:**

- The values directly below the arrows indicate the filter sizes $f$ of the corresponding convolution and pooling operations.
- $s$ stands for stride. If no stride is specified, a stride of $s = 1$ is used.
- All convolutional and pooling layers use a padding of $p = \frac{f-1}{2}$ for a corresponding filter size of $f$.
- For each convolutional filter, we include a bias.

0 1 a) In the figure above, the output layers for classification are missing. Explain:

1. Which type of last layer you would use there and how would you choose it's dimension?
2. Which output function and loss function would you choose for this task?

1. Fully-connected layer (0.5 points) with 1000 neurons (0.5 points)
2. Softmax function combined with Cross-Entropy Loss (0.5 point)

0 1 b) Calculate the output dimension of the image after passing through the first convolutional layer. Make sure to include your calculations in the solution.

Formula to calculate the output dimension: $\dim_{out} = \frac{\dim_{in} - f + 2 \cdot p}{s} + 1$ Output Dimension $(\frac{256-11}{4} + 1) = 64 \times 64 \times 96$ (1 Point)

0 1 c) Calculate the number of learnable parameters in the first convolutional layer. Make sure to include your calculations in the solution. Un-multiplied answers are accepted (e.g., $3 * 3 * 16$)

Number of Parameters $(11 * 11 * 3 + 1) * 96 = 34944$ parameters (1 Point)

d) Calculate the size of the combined receptive field of the first two and of the first three layers. This corresponds resp. to the area of pixels in the input image that each neuron

0 1 2

1. after the second layer (right after the first MAX-POOL layer)
2. after the third layer (before the second MAX-POOL layer)

"sees".
**Hint:** Strides affect the total receptive field sizes of subsequent layers.

Size of total receptive field after $k > 1$: $r_k = r_{k-1} + \left(\prod_{i=1}^{k-1} s_i\right) \cdot (f_k - 1)$

- layer 1: $r_1 = 11$ ($11 \times 11$ filter)
- layer 2: $r_2 = 11 + (4 \cdot (3 - 1)) = 11 + 8 = 19$ (0.5p answer, 0.5p calculation)
- layer 3: $r_3 = 19 + (4 \cdot 2 \cdot (5 - 1)) = 19 + 8 \cdot 4 = 19 + 32 = 51$ (0.5p answer, 0.5p calculation)

Alternative solution for $r_3$ (from right to left, needs separate calculation for $r_2$!): $r_{i-j} = f_j + (r_{i-(j+1)} - 1) * s_j$)

- 3rd to 2nd layer: $r_{3-2} = 5$ ($5 \times 5$ filter)
- 3rd to 1st layer: $r_{3-1} = 3 + (5 - 1) \cdot 2 = 11$ ($3 \times 3$ filter, stride 2)
- 3rd layer to input: $r_3 = r_{3-0} = 11 + (11 - 1) \cdot 4 = 51$ ($11 \times 11$ filter, stride 4)

After a series of convolutional layers, the architecture has 3 fully connected layers that help process the spatial information obtained by the convolutions and prepare it for the output layer. Our friend just found an article about image segmentation and gets really excited about this. He decides to forget about the classification task and wants to work on image segmentation.

We don't need to reject our architecture: We first convert our architecture to a fully-convolutional network by ditching the fully connected layers. Next, we want to produce an output similar to the input size from this bottleneck where we would like to mirror the current architecture.

e) How would you replace the convolutional layers in the mirrored architecture to increase the the image size from our bottleneck onwards?

0 1

Accepted answers: upsampling/unpooling + convolution, transposed convolution.

Grading notes: upsampling alone (without conv) is 0.5p, "inverse convolution", "transformed convolution", and all other misspellings are 0.5p.

f) The new architecture is able to process an image of any input size. How about the original architecture that we started with, was it able to handle images of arbitrary sizes as input, too? Give an explanation for your answer.

0 1

No, the original architecture was not able to handle images of arbitrary size (0.5p).

Explanation: Fully-connected layers require fixed size and cannot handle variable output size of convolutional layers (0.5p).

## Problem 8 General Training (8 credits)

0 1

a) When we normalize the data as a preprocessing step is the test set also normalized? If no, explain why you shouldn't do it. If yes, describe how do we normalize it.

Yes (0.5p), the test set is normalized with statistics calculated from the training set (0.5p).

Grading note: only explicit mentioning of training statistics gives full points, e.g., "yes, the test set is normalized similarly as train" is 0.5p.

0 1 2

b) When implementing a neural network layer from scratch, we usually implement a 'forward' and a 'backward' function for each layer. Explain what these functions do, which arguments they take, and what they return.

Forward Function: takes output from previous layer, performs operation, returns result (0.5 pt.), caches values needed for gradient computation during backprop (0.5 pt.)
Backward Function: takes the upstream gradient (0.5 pt.), returns partial derivatives or gradients (0.5 pt.)

If the answer has been written with respect to forward and backward pass through the network, instead of forward() and backward() functions of a layer, partial points have been provided if the answer contains the above key points.

0 1

c) You are training a neural network with 15 fully-connected layers using a *tanh* nonlinearity.

1. Explain the behavior of the gradient of the non-linearity with respect to very large positive inputs.
2. Suggest another non-linear activation function that does not have this behaviour.

1. Because the tanh is almost flat for very large positive values, its gradient will be almost 0. (or) vanishing gradients (or) gradients die off (0.5pt)
2. ReLu and it's variants (0.5pt)

No points only if saturation of tanh is mentioned without stating the gradients becoming zero.

0 1 2

d) Explain how K-fold Cross-Validation works, and how it is used to calculate validation scores for a given model.

We split the data into K parts. We train the model on K-1 parts and validate on the held out part. We repeat this process K times, so that each part was held out once, then we average results to obtain our final validation score. (1 pt for data split into K-1 train and 1 validation; 1 pt for result averaging)

0 1 2

e) You have 4000 cat and 100 dog images and want to train a neural network on these images to do binary classification. What problem(s) do you foresee with this dataset distribution? Name two possible solutions.

Network prefers cats as they are more likely or imbalance between classes (1pt)
leave out pics/reweight dataloader/reweight loss function/collect more dog images/data augmentation on dog images (0.5pt/sol)
No points for: dropout, regularization, batch norm, transfer learning, "get more data" or "data augmentation" without specifying which

## Problem 9 Unsorted Short Questions (8.5 credits)

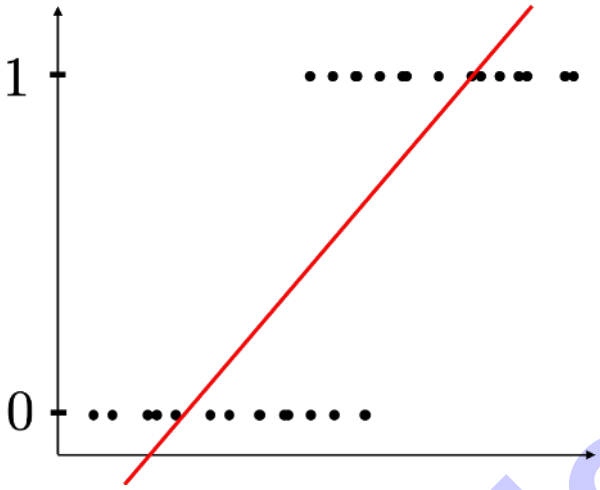
a) Given an input image with $50 \times 50$ pixels followed by a fully connected layer with 1024 neurons and a tanh activation. Compute the variance normalization coefficient of the Xavier initialization for the fully connected layer. How should the coefficient change if the tanh is replaced by a ReLU?

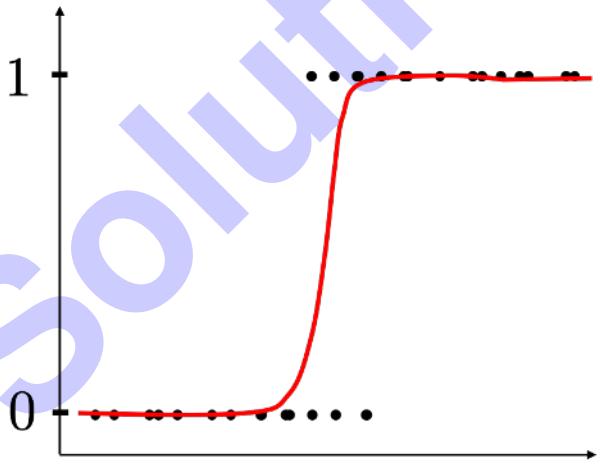
0 1 2

*Var* = 1/2500 (1 point)
ReLU: multiplied by 2, i.e. *Var* = 1/1250

b) Consider the following data points. Sketch a linear (0.5p) and logistic (0.5p) regression line into the figures. Which model is more suitable for this task (0.5p)?

0 1




Logistic regression (0.5 pt) For each drawing 0.5 pt. The logistic regression line needs to be a function and must not have multiple y values for one x value. The linear regression line must not have any bends or non-linear points.

0 1 2 3

c) Design a 2-layer neural network with 3 neurons to implement a ***XOR*** function. For each neuron,

$$y = f(wx + b), f(x) = \begin{cases} 0, x \leq 0 \\ 1, x > 0 \end{cases}$$

specify the proper weight and bias, where both weight and bias value must be chosen from $[-1, 0, 1]$.
**Hint: *XOR =*** $(A \cap \bar{B}) \cup (\bar{A} \cap B)$)

| | | | |
|---|---|---|---|
| $W_{11}$ | | $b_{11}$ | |
| $W_{12}$ | | $b_{12}$ | |
| $W_{21}$ | | $b_{21}$ | |

Layer 1:

1. $A \cap \bar{B} : AND(A, \bar{B}) = A - B - 0 \rightarrow S_1, \therefore W_{11} = [1, -1], b_{11} = 0$
2. $\bar{A} \cap B : AND(\bar{A}, B) = -A + B - 0 \rightarrow S_2, \therefore W_{12} = [-1, 1], b_{12} = 0$

Layer 2:

1. $S_1 \cup S_2 : OR(S_1, S_2) = S_1 + S_2 + 0 \rightarrow \therefore W_{21} = [1, 1], b_{21} = 0$

0.5pt if you only have $b_{21} = 0$ correct and 0.5pt if you only have $b_{11} = 0$ and $b_{12} = 0$ correct.

0 1 2

d) Suppose we have a grayscale image represented as an array, where larger values denote lighter pixels. What is the effect when we convolve it with the following kernel?

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & -4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (1)$$

If the input image is encoded as numbers between [0 and 255] or [0 and 1], the output will be completely black. 2pt (only 1pt for "darker") Or: If the input image is encoded as positive and negative numbers, such as numbers in [-1,1], the kernel will invert the image (1pt), blur it (1pt), and increase the inverted values. (1pt) NOT edge detection. NOT sharpening. NOT cross detection.

## Problem 10 Recurrent Neural Networks and Backpropagation (8 credits)

Recurrent neural networks, also known as RNNs, are a class of neural networks that allow an arbitrary number of inputs and, thus, are often used for sequences of data, e.g., in the fields of natural language processing and speech recognition.

a) Mathematically explain the reason for exploding and vanishing gradients when using a classic RNN, i.e., $A_t = \theta_c A_{t-1} + \theta_x x_t$, where both $\theta_c$ and $\theta_x$ are orthogonal. (2 points)

0 1 2

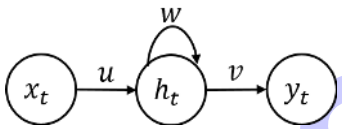
1. Show backpropagation explicitly to calculate gradients (1 point):

$$\frac{\delta h^{(t)}}{\delta h^{(1)}} = \frac{\delta h^{(t)}}{\delta h^{(t-1)}} \cdots \frac{\delta h^{(2)}}{\delta h^{(1)}}$$

2. After eigen-decomposition of $\theta_c$, the **largest** eigenvalue of $\theta_c > 1$ means explosion(0.5 point) and $< 1$ means vanishing (0.5 point).
   P.s. if the answer discussed eigenvalues of $A_t$ instead of $\theta_c$: 0 point
   if the answer discussed two cases of eigenvalues but didn't show explicitly which matrix should be decomposed: 0.5 point

b) Now consider the following RNN

0 1 2 3




which uses the one-dimensional ReLU-RNN cell

$$h_t = ReLU(u * h_{t-1} + w * x_t).$$

Compute the forward propagation $y_2, h_2$ and the gradient $\mathbf{dy}_2 := \frac{\delta y_2}{\delta u}$ where

$$h_0 = 3,\ w = 2,\ v = -1,\ u = 3,\ x_1 = 1,\ x_2 = 2.$$

Since there is a mismatch between the graph and the given equation, answers based either on the graph or on the equation are accepted. Solution based on the graph:

$$h_1 = ReLU(u * x_1 + w * h_0) = 9$$
$$h_2 = ReLU(u * x_2 + w * h_1) = 24(1P)$$
$$y_2 = v * h_2 = -24(1P)$$
$$\frac{\delta y_2}{\delta u} = v * w * x_1 + v * x_2 = -4(1P)$$

Solution based on the equation:

$$h_1 = ReLU(u * h_0 + w * x_1) = 11$$
$$h_2 = ReLU(u * h_1 + w * x_2) = 37(1P)$$
$$y_2 = v * h_2 = -37(1P)$$
$$\frac{\delta y_2}{\delta u} = v * (h_1 + u * h_0) = -20(1P)$$

0 1 2

c) To circumvent the vanishing gradient problem, the Long-Short Term Memory (LSTM) unit was proposed. It is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

where $g_1$, $g_2$, and $g_3$ are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** $f$, **update** $u$, and **output** $o$ gates. (1p)
2) What does the value $c_t$ represent in a LSTM? (1p)

$g_1$ = output gate
$g_2$ = forget gate
$g_3$ = update gate/input gate
(1 pt)
$c_t$: cell state
(1 pt)

0 1

d) Why does the LSTM unit solve the vanishing gradient problem that is present in the default definition of an RNN cell?

gradient highway through the cell state (0.5pt) and gate system to remove or add information to the cell state (0.5pt)
(another alternative: from the perspectives of weights and activation functions as in the slides)

## Problem 11 **Advanced Deep Learning (6 credits)**

a) Where do we use neural networks in Q-learning (1p) and why are they needed (1p)?

0 1 2

Function approximator for Q values: $Q^*(s,a) = Q(s,a;\theta)$ if $\theta$ are the network parameters. For even semi big problems, calculating all state action pairs is not tractable, but we can approximate it using neural networks.

b) State the Markov assumption in reinforcement learning both in words and as a formula for probability states $s_t$ at time $t$.

0 1 2

Multiple formulations possible including (1p):

$$\mathbb{P}(s_t|s_{t-1}) = \mathbb{P}(s_t|s_1, \dots, s_{t-1})$$

or

$$\mathbb{P}(s_t|s_{t-1}) = \mathbb{P}(s_t|s_{t-1}, \dots, s_1)$$

or

$$\mathbb{P}(s_{t+1}|s_t) = \mathbb{P}(s_{t+1}|s_1, \dots, s_t)$$

and: The state at time $t$ only depends on the previous state. / The next future state only depends on the present state. (1p) - Not mentioning the "state" subtracts 0.5 points unless the formula is also present.

c) What are the two key challenges when trying to define a neural network on graphs?

0 1 2

Variable sized inputs or variable number of nodes and edges (1P)
permutation invariance(1P)

**Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.**

# Curr Year Mid Sem Solution

BITS WILP M Tech Data Science & Engineering (Birla Institute of Technology and Science, Pilani)

**Birla Institute of Technology & Science, Pilani**
**Work Integrated Learning Programmes Division**
**Second Semester 2020-21**
**M.Tech. (Data Science and Engineering)**
**Midsem Examination (Regular)**

| | |
|---|---|
| Course No. | : DSECLZG524 |
| Course Title | : DEEP LEARNING |
| Nature of Exam | : Open Book |
| Weightage | : 30% |
| Duration | : 2 Hours |
| Date of Examination | : July 10th, 2020 |

No. of Pages = 3
*No. of Questions = 5*

Time of Exam: 10 AM – 12 PM

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.** [2 + 2.5 + 1.5 **=6 marks]**

Consider the following DNN for image classification of 10 classes. The dataset consists of colour images of size 16x16.

```
# Input Layer
inputs = keras.Input(shape=(##A##))

# Layer 1
x = layers.Dense(30, activation="relu", use_bias=False)(inputs)

# Layer 2
x = layers.Dense(60, activation="relu")(x)

# Layer 3
x = layers.Dense(120, activation="relu", use_bias=False)(x)

# Output Layer
outputs = layers.Dense(##B##, activation=##C##)(x)

model = keras.Model(inputs=inputs, outputs=outputs,
name="model")
model.compile(optimizer = 'adam', loss = ##D##,
metrics=['accuracy'])
```

A. Write the value or code of ##A##, ##B##, ##C## and ##D## to complete the network.

**A - (16*16*3,) or (768,)**
**B - 10**
**C - softmax**
**D - categorical cross-entropy**

B. If we add a dropout layer of value 0.4 after layer 2-

I.what will be the total number of active neurons in layer 2 during

i. model training

60*(1-0.4) = 36

ii. ii. model testing

60

II. what will be the total number of parameters in the network? What is the change in the number of parameters compared to the original network?

**Layer 1- 768*30 = 23,040**
**Layer 2- 30*60 + 60 = 1,860**
**Layer 3- 60*120 = 7,200**
**Output Layer - 120*10 + 10 = 1,210**
**Total = 23,040 + 1,860 + 7,200 + 1,210 = 33,310**
**No change in the number of parameters if dropout is added.**

C. Assume that the image dataset consists of 5000 train images and 1000 test images. We run for 10 epochs with batch size = 32, learning rate (LR) = 0.01 and sgd optimizer. During model training

I. What will be the size of the batch in the last iteration?

**5000 - (156x32) = 8**

II. What is the number of times we perform forward propagation?

**5000**

III. What is the number of times we perform backpropagation? And why?

**5000/32 = 156.25 i.e., 157 times backpropagation**

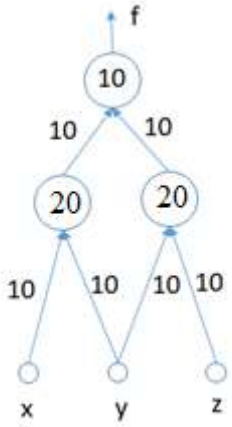
**We perform backprop one time per batch taking the average loss of all the images in that batch and finally update the parameters of the network.**

**Question 2.** [2+4**=6 Marks**]

A. Implement the following truth-table as a single-hidden layer MLP with the fewest perceptrons. x, y, z are input binary variables and f is the Boolean function, as specified in the truth table. Perceptrons use step activation function, i.e.,

output = +1 if total input >= bias,
= 0 otherwise.

f (x,y,z) = yz+xy

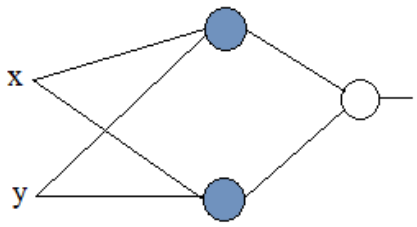



B. Specify all the weights and bias values of the network. Note, weights can be only +10 or -10 or 0, and bias values are multiples of 10 only.

| x | y | z | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Question 3.** [3+3=**6 Marks**]

A. Consider the MLP below where each perceptron has a linear activation function where the output z is related to its inputs x and y as z = ax + by + c. We claim that this network can be replaced by a single perceptron with a possibly different linear activation function. Is this claim true or not? Give quantitative justification for your answer.




Let x1, y1 be the outputs of the upper and lower hidden nodes, respectively. So, x1=ax+by+c, and y1=ax+by+c.
So, final output of network, z = ax1+by1+c=a(ax+by+c)+b(ax+by+c)+c
Or, z = $(a^2+ab)x+(b^2+ab)y+(ac+bc+c)$

Thus, a single perceptron with input x, y is enough to implement the given network.

B. We would like to minimize the following quadratic function $0.5*25x^2 + 16*x + 12$ using exactly two steps using gradient-descent with a learning rate $\eta = 1.0$ for the first step. What should the learning rate of the second-step be in order to obtain the minimum in two-steps?

Optimal learning rate $\eta = 1/25$

**Question 4.** [4 **Marks]**

Use RPROP to perform the minimization of $24 * x^2 + 16x + 12$ starting at x = 2.0 with initial step size 1.0. Take $\alpha = 1.1$ and $\beta = 0.8$. Find minimum of the function and corresponding value of x.

$\frac{df}{dx} = 48x+16$

$\frac{df}{dx}\Big|_{x0=2.0} = +ve$

x = 2.0-1.0=1.0

$\frac{df}{dx}\Big|_{x=1.0} = +ve$

x = 1.0-1.1=-0.1

$\frac{df}{dx}\Big|_{x=-0.1} = +ve$

x = -0.1-1.1*1.1=-1.31

$\frac{df}{dx}\Big|_{x=-1.31} = -ve$

x = -0.1-1.1*1.1*0.8=-1.068

$\frac{df}{dx}\Big|_{x=-1.068} = -ve$

x = -0.1-1.1*1.1*0.8*0.8=-0.8744

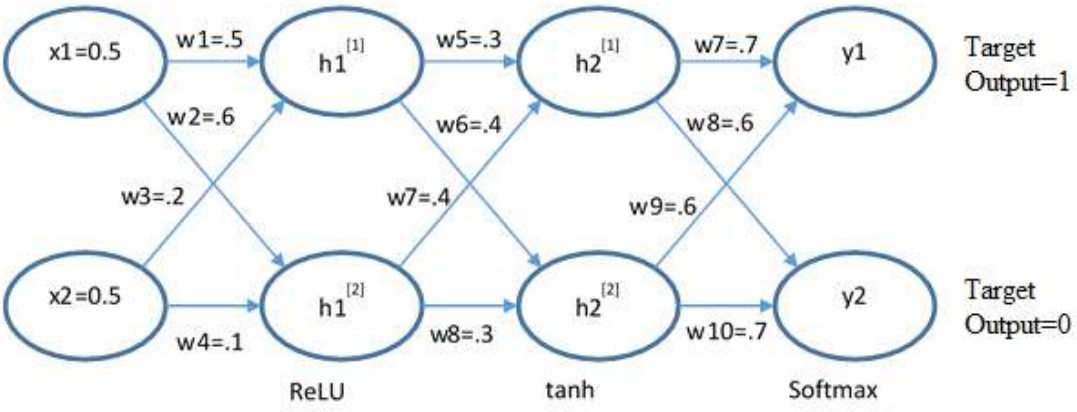
$\frac{df}{dx}\Big|_{x=-0.8744} = -ve$

Follow this process, until df/dx is close to 0.

The function reaches minimum of 12 at x=-0.33.

**Question 5.** [5+1+2=**8 Marks]**

A. The following network uses ReLU, tanh and softmax activation for hidden layer 1, hidden layer 2, and output layer, respectively. For the given input, calculate the Loss, backpropagate the error, and find new weights for w1 in the next iteration. No bias is used in any node and learning rate is 1.0.




$dh1^{[1]} = 0.35$ $dh1^{[2]} = 0.35$ $h2^{[1]} = 0.245$ $dh2^{[2]} = 0.245$

y1=exp(0.245*1.3)/(exp(0.245*1.3)+exp(0.245*1.3)=0.5

y2= exp(0.245*1.3)/(exp(0.245*1.3)+exp(0.245*1.3)=0.5

Loss L = -d1*log(y1)-d2*log(y2)=-log(y1)=0.693

$dL/dw1 = dL/dz1*dz1/dh2^{[1]} * dh2^{[1]} / dh1^{[1]} * dh1^{[1]} / dw1 +$
$dL/dz2*dz2/dh2^{[1]} * dh2^{[1]} / dh1^{[1]} * dh1^{[1]} / dw1 +$
$dL/dz1*dz1/dh2^{[2]} * dh2^{[2]} / dh1^{[1]} * dh1^{[1]} / dw1 +$
$dL/dz2*dz2/dh2^{[2]} * dh2^{[2]} / dh1^{[1]} * dh1^{[1]} / dw1$

$= dL/dz1*w7 * w5 * (1-h2^{[1]} * h2^{[1]}) * x1 + dL/dz2*w8 * w5 * (1-h2^{[1]} * h2^{[1]}) * x1 +$
$dL/dz1*w9 * w6 * (1-h2^{[2]} * h2^{[2]}) * x1 + dL/dz2*w10 * w6 * (1-h2^{[2]} * h2^{[2]}) * x1$

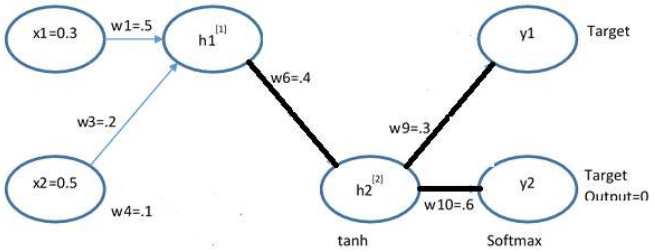
dL/dz1 = dL/dy1*dy1/dz1 = -1/y1*y1*(1-y1)=0.5-1=-0.50
dL/dz2 = dL/dy2*dy2/dz2 = 0

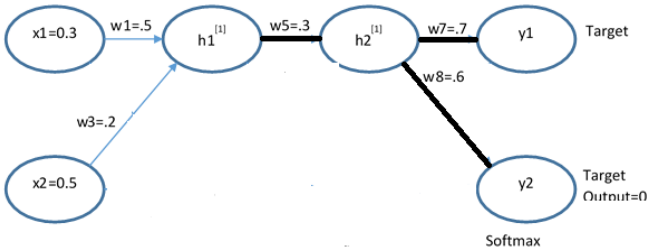
$w1 = 0.5 - (-0.5*w7 * w5 * (1-h2^{[1]} * h2^{[1]}) * x1-0.5*w9 * w6 * (1-h2^{[2]} * h2^{[2]}) * x1)$
=0.5+0.5*0.5*(0.21*(1-0.245*0.245)+0.24*(1-0.245*0.245))=0.6057

B. Assume you have a 50% dropout for Hidden layer 1 and Hidden layer 2.

I. How many unique paths are possible for the above network? 4

II. Identify two unique paths and explain how forward propagation and back propagation is calculated.




Forward calculation will be based on the above two network structure and backward gradient calculation and weight updates will be only based on the above two structures. Other weights will not chance in an iteration.

XXXXXXXXXXX

**Birla Institute of Technology & Science, Pilani**
**Work Integrated Learning Programmes Division**
**Second Semester 2020-21**
**M.Tech. (Data Science and Engineering)**
**Midsem Examination (Makeup)**

| | |
|---|---|
| Course No. | : DSECLZG524 |
| Course Title | : DEEP LEARNING |
| Nature of Exam | : Open Book |
| Weightage | : 30% |
| Duration | : 2 Hours |
| Date of Examination | : July 24th, 2021 |

No. of Pages = 3
*No. of Questions = 5*

Time of Exam: 10 AM – 12 PM

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.** **[2 + 2 + 1 + 1 + 1 =7 marks]**

A. While selecting different layers of a model for an image classification problem, what all factors do we consider to decide the total number of neurons? Select the correct/incorrect answers and explain.

I) Hardware capacity
II) Inter class variation in the dataset
III) Intra class variation in the dataset
IV) Total number of images in the dataset
V) Number of image augmentations applied on the dataset

Ans. A, B, C, E

A - Number of neurons cannot exceed the memory and hardware capacity.

B, C - More the inter and intra class variations, more the number of neurons required to capture the variation.

E - More the augmentations applied, more the number of neurons required to capture variations in original images and the augmented images.

Number of kernels does not depend on total number of images, as it can be handled by the batch size

B. An image classification problem consists of 4 classes (0, 1, 2, and 3). The output layer of the model consists of a one-hot encoding vector corresponding to each class. The values of each class before softmax activation are 10, 8, 12, and 5.

I) What is the output value of the predicted class post activation?

Class with the pre-activation value of 12 is fed to the loss function. Its output value is $(e^{12})/(e^{10} + e^{8} + e^{12} + e^{5})$ =

II) What is the difference between softmax values of class 1 and class 3?

Difference in softmax values of class 1 and 3 is $(e^8 - e^5)/(e^{10} + e^8 + e^{12} + e^5)$ =

C. Tuning hyperparameters using a test dataset. Is it preferred, if not then why?
Tuning model hyperparameters to a test set means that the hyperparameters may overfit to that test set. If the same test set is used to estimate performance, it will produce an overestimate. Using a separate validation set for tuning and test set for measuring performance provides unbiased, realistic measurement of performance.

D. If there are 8 classes and the classifier predicts each class with equal probability. What will be the cross-entropy loss on any single example?
Cross-Entropy loss simplifies to negative log of the predicted probability for the correct class. At the start of training, we have approximately uniform probabilities for the 8 classes, or 1/8 for each class. So, that means -log(1/8) is what the loss should approximately be at the start.

E. In a scenario while training a neural network for classification, the training loss comes much lower than the validation loss. What do you think is the reason and give two ways to resolve it?
The model is now overfitting, and all of these are valid techniques to address it.
1. Use a network with fewer layers
2. Increase L2 regularization weight
However, since dropout is a form of regularization, then decreasing it will have the opposite effect, as will increasing network size.
1. Decrease dropout probability
2. Increase the size of each hidden layer

**Question 2.** [2.5+2.5**=5 Marks**]

Consider the expression f(x1,x2) = 2x1x2 + 3x1 + 4x2.

A. Determine for the point (x1, x2) = (0, 0) whether it is (a) a local minima, (b) a local maxima or a saddle point, or none of these.
The point (0, 0) is either a local minimum, local maximum or saddle point since the gradient of f is zero at this point. To determine which of these it is we need to consider the Hessian H which evaluates to

$$\begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}$$

The eigenvalues of the above matrix turn out to be 1 and -1 respectively which means that (0, 0) is a saddle point.

B. Do the same exercise for the point (x1, x2) = (1, 1).

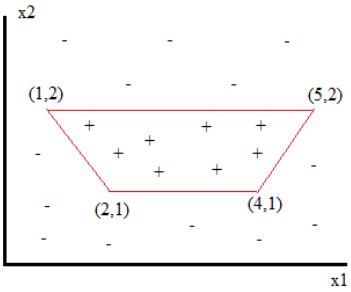
(1, 1) is none of these as the gradient of f is non-zero at this point.

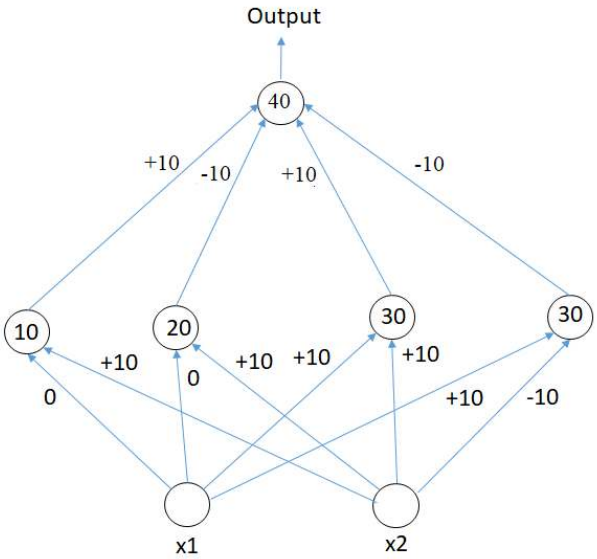
**Question 3.** [1+1+4=**6 Marks**]

A. What is the minimum number of hidden layers required to implement the following decision boundary?
1

B. What will be minimum number of hidden nodes required? Show the network architecture.
4

C. Organize the hidden nodes from left to right in ascending order of bias values. Use the input x1 as the left node in the input layer. Specify all the weights (only 0, 10 or -10 allowed) and bias (only 0 or multiples of 10). Hidden and Output units use step activation, i.e., output = 1 if total input >=bias, otherwise -1.




**Question 4.** [4 **Marks**]

Consider the following problem of L1-regularization, i.e., minimize for i=1 to n

$$L_R(\theta_i) = H_{ii}(\theta_i - \theta_i^*)^2 + \alpha \mid \theta_i \mid \text{ where all } H_{ii} > 0,$$

and α is the L1 regularization constant. Find the smallest value of α (in terms of $H_{ii}$ and $\theta_i^*$) so that the optimal regularized solution to the given minimization problem, $\theta_i^R = 0$ for all i.

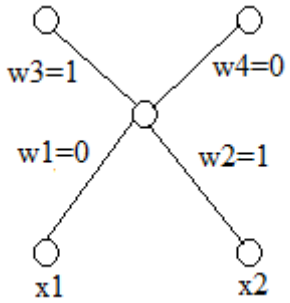
**Answer**: The regularised optimal solution is as follows: $(\theta_R^*)_i = max(\theta_i^* - \frac{\alpha}{H_{ii}}, 0), if \theta_i^* \geq 0$ and $(\theta_R^*)_i = min(\theta_i^* + \frac{\alpha}{H_{ii}}, 0), if \theta_i^* \leq 0$. If we select $\alpha$ such that $\theta_i^* < \frac{\alpha}{H_{ii}}$ and $-\theta_i^* < \frac{\alpha}{H_{ii}}$, then we can ensure that $(\theta_R^*)_i = 0$ for all $i$. This condition amounts to picking an $\alpha$ such that $\alpha \geq H_{ii}\|\theta_i^*\|$ for all $i$.

**Question 5.** [2+1+2+3=**8 Marks**]

A. Consider a fully connected multilayer perceptron (each hidden node is connected to all inputs and all outputs) with 2 dimensional binary input (0 or 1) and one hidden layer with ReLU activation function, and target output, same as the input (so, input [1,1] is associated with target output [1,1]). What activation function will you use at the output layer? What type of loss function will you use?
Sigmoid activation at output node since outputs are 0-1. Cross entropy is used as the loss function because the output is 0-1.

B. At iteration t, the weights are shown in the following architecture along with the input vector (x1=1, x2=0). What will be value of the loss function at iteration t?
Output of hidden node =0. Reconstructed output = (0.5, 0.5) Target output = (1.0,0.0)
So, loss = -ln(0.5)-ln(0.5)=1.386

C. What will be the weights w1 and w3 in iteration t+1 assuming gradient descent and gradient descent with momentum? Assume learning rate = 0.3 and momentum constant = 0.7. At (t-1), w1=-0.5, w2=0.5, w3=0.5 and w4=-0.5. Assume derivative of ReLU(z)=0 at z=0.




In ordinary gradient descent, change in w3 is proportional to hidden node output. So in this case, w3(t+1)=1.0 if ordinary gradient descent is used.
In ordinary gradient descent, change in w1is proportional to derivative of the hidden node activation function at hidden node output point. Deviative of ReLU(z)=0 at z=0. So in this case, w1(t+1)=0.0 if ordinary gradient descent is used.
For momentum based gradient update, w3(t+1)=1.0+0.7*0.5=1.35, w1(t+1) = 0.0+0.7*0.5=0.35

XXXXXXXXXXXX

# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

**M. Tech. in AI & ML**

**I Semester 2023-2024**

**Mid-Semester Test**
**(EC2 - Makeup)**

| | |
|---|---|
| Course Number | AIMLCZG51 |
| Course Name | DEEP NEURAL NETWORK |
| Nature of Exam | Closed Book |
| Weight-age for grading | 30 |
| Duration | 2 hrs |

| | |
|---|---|
| * Pages | 2 |
| * Questions | 6 |

---

**Instructions**

1. All questions are compulsory.
2. All answers must be directed to the question in short and simple paragraphs or bullet points; use visuals/diagrams wherever necessary.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.

---

1. Consider a single layer perceptron having 2 inputs and 1 output. Let threshold be 0.5, learning rate be 0.6, bias be -2 and weight values $w_1 = 0.3$ and $w_2 = 0.7$. Given the input patterns in the table, compute the value of the output and train using perceptron learning rule for one epoch. [5]

| E.g. # | $x_1$ | $x_2$ | $y$ |
|---|---|---|---|
| 1 | 1 | 1 | +1 |
| 2 | 1 | 0 | +1 |
| 3 | 0 | 1 | -1 |
| 4 | 0 | 0 | +1 |

Rubrics and answer

- z - 0.25, h - 0.25, $\Delta$ - 0.25, new - 0.25 marks
- Each row in table carries 1 mark each.
- Any wrong equation or wrong computation, reduce appropriately, but maintain a minimum of 0.25 for each row.

$$z = w_1x_1 + w_2x_2 + b$$
$$h = sign(z)$$
$$\Delta w_1 = \eta(t-h)x_1$$
$$\Delta w_2 = \eta(t-h)x_2$$
$$\Delta b = \eta(t-h)$$
$$w_{new} \leftarrow w_{old} + \Delta w$$

Epoch 1

| $x_1$ | $x_2$ | $y$ | $b$ | $w_1$ | $w_2$ | z | $h$ | $y, h$ | $\Delta$ | new b,$w_1$,$w_2$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | +1 | -2 | 0.3 | 0.7 | 0.3 + 0.7 - 2 =-1 | -1 | N | $\delta b = 0.6(1-(-1)) = 1.2$<br>$\delta w_1 = 0.6(1+1)1 = 1.2$<br>$\delta w_2 = 0.6(1+1)1 = 1.2$ | $b = -2 + 1.2 = 0.8$<br>$w_1 = 0.3 + 1.2 = 1.5$<br>$w_2 = 0.7 + 1.2 = 1.9$ |
| 1 | 0 | +1 | 0.8 | 1.5 | 1.9 | 1.5 + 0 + 0.8 =2.3 | +1 | Y | | |
| 0 | 1 | -1 | 0.8 | 1.5 | 1.9 | 0 + 1.9 + 0.8 = 2.7 | +1 | N | $\delta b = 0.6(-1-1) = -1.2$<br>$\delta w_1 = 0.6(-1-1)0 = 0$<br>$\delta w_2 = 0.6(-1-1)1 = -1.2$ | $b = 0.8 - 1.2 = -0.4$<br>$w_1 = 1.5$<br>$w_2 = 1.9 - 1.2 = 0.7$ |
| 0 | 0 | +1 | -0.4 | 1.5 | 0.7 | 0+0-0.4 = -0.4 | -1 | N | $\delta b = 0.6(1+1) = 1.2$<br>$\delta w_1 = 0.6(1+1)0 = 0$<br>$\delta w_2 = 0.6(1+1)0 = 0$ | $b = -0.4 + 1.2 = 0.8$<br>$w_1 = 1.5$<br>$w_2 = 0.7$ |

2. Derive the equation for the derivative of categorical cross-entropy loss $L$ with respect to the weighted sum $Z$, for a three-class classification problem. Assume single hidden layer and $d$ input neurons. [5]

Rubrics and answer

for class k, output $Z_k = \sum_{i=1}^{3} w_{jk} \cdot a_i$

softmax function $\hat{y}_k = \frac{e^{Z_k}}{\sum_{j=1}^{3} e^{Z_j}}$ 0.5 mark

categorical cross-entropy loss $L = -\sum_{k=1}^{3} y_k \cdot \log(\hat{y}_k)$ 1 mark

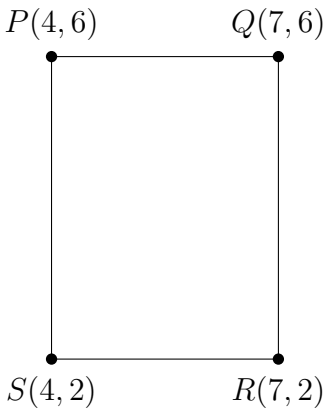
$$\frac{\partial L}{\partial Z_k} = \frac{\partial L}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial Z_k}$$ 1 mark

$$\frac{\partial L}{\partial \hat{y}_k} = -\frac{y_k}{\hat{y}_k}$$ 1 mark

$$\frac{\partial \hat{y}_k}{\partial Z_k} = \hat{y}_k \cdot (1 - \hat{y}_k)$$ 1 mark

$$\frac{\partial L}{\partial Z_k} = -y_k \cdot (1 - \hat{y}_k)$$ 0.5 mark

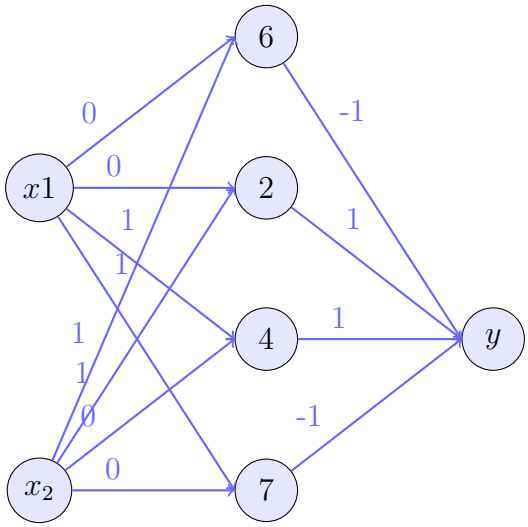
3. Construct an MLP for the given complex decision boundary. [6]




Rubrics and answer

- Each equation, 1 mark each.
- MLP, hidden layer with correct parameters - 1 mark
- MLP, last layer with correct parameters - 1 mark

$$PQ \rightarrow 0x_1 + 1x_2 = 6$$
$$SR \rightarrow 0x_1 + 1x_2 = 2$$
$$PS \rightarrow 1x_1 + 0x_2 = 4$$
$$QR \rightarrow 1x_1 + 0x_2 = 7$$




4. Find the minimum value of $p$ for the equation $t = (2p + 3)^2$ using SGD. Assume the initial value of $p$ as 6 and learning rate as 0.1. Do 3 iterations. [4]

Rubrics and answer

- Each equation, 1 mark each.

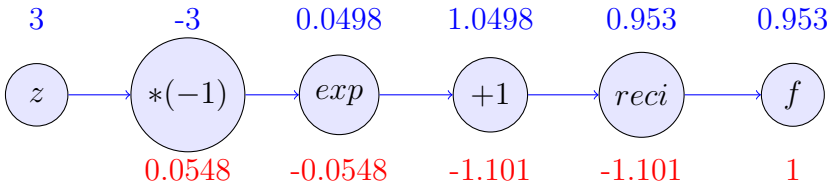
$$dt/dp = 2(2p+3)2 = 8p+12$$
$$p_1 \leftarrow 6 - 0.1(8*6+12) = 0$$
$$p_2 \leftarrow 0 - 0.1(8*0+12) = -1.2$$
$$p_3 \leftarrow (-1.2) - 0.1(8*(-1.2)+12) = -1.44$$

5. Draw the computational graph for the equation $f = 1/(1+e^{(-z)})$. Show the computations of derivatives of $f$ wrt $z$ in the graph. Using the graph, compute the value of $f$ and the derivatives if $z = 3$. [5]

Rubrics and answer

- FP graph 1 mark
- BP graph 1 mark
- Output computation 1 mark
- Gradient computation 1 mark

$$p = -z = -3 \qquad \frac{\partial p}{\partial z} = -1 \qquad \frac{\partial f}{\partial p} = (-1)(-0.0548) = 0.0548$$
$$q = e^p = e^{-3} = 0.0498 \qquad \frac{\partial q}{\partial p} = e^p \qquad \frac{\partial f}{\partial q} = e^{-3}(-1.101) = -0.0548$$
$$r = p + 1 = 0.0498 + 1 = 1.0498 \qquad \frac{\partial r}{\partial p} = 1 \qquad \frac{\partial f}{\partial p} = 1(-1.101) = -1.101$$
$$f = \frac{1}{r} = \frac{1}{1.0498} = 0.953 \qquad \frac{\partial f}{\partial r} = \frac{-1}{r^2} \qquad \frac{\partial f}{\partial r} = \frac{-1}{0.953^2} = -1.101$$




6. Given an error surface, compute the value that minimizes the error with respect to $(w_1, w_2, w_3)$. Compute the minimum possible value of error. [5]

$$E(w_1, w_2, w_3) = (w_1 - w_2)^3 - 2(w_1^2 - w_2) + w_1^2 + w_2^2$$

Rubrics and answer

$$\frac{\partial E}{\partial w_1} = 3(w_1 - w_2)^2 - 2w_1$$
$$\frac{\partial E}{\partial w_2} = -3(w_1 - w_2)^2 + 2w_2 + 2$$
$$\frac{\partial E}{\partial w_3} = 0$$

- Each equation 1 mark
- if the student writes or attempts that the equations have to be equated to zero and compute the value of parameters - 2 marks

# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

**M. Tech. in AI & ML**

**I Semester 2023-2024**

**Mid-Semester Test**
**(EC2 - Regular)**

| | |
|---|---|
| Course Number | AIMLCZG51 |
| Course Name | DEEP NEURAL NETWORK |
| Nature of Exam | Closed Book |
| Weight-age for grading | 30 |
| Duration | 2 hrs |

| | |
|---|---|
| # Pages | 2 |
| # Questions | 6 |

---

**Instructions**

1. All questions are compulsory.
2. All answers must be directed to the question in short and simple paragraphs or bullet points; use visuals/diagrams wherever necessary.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.

---

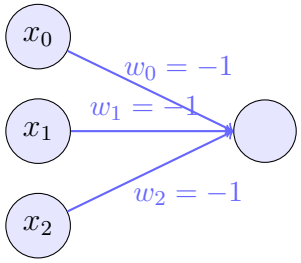
1. Given a truth table, design a perceptron and find the weights and threshold of the perceptron. [5]

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |

Rubrics and one solution

| $x_1$ | $x_2$ | $y$ | $h$ |
|---|---|---|---|
| -1 | -1 | 1 | $w_0 + w_1(-1) + w_2(-1) > 0$ |
| 1 | -1 | -1 | $w_0 + w_1(1) + w_2(-1) < 0$ |
| -1 | 1 | -1 | $w_0 + w_1(-1) + w_2(1) < 0$ |
| 1 | 1 | -1 | $w_0 + w_1(1) + w_2(1) < 0$ |

$$w_0 - w_1 - w_2 > 0 \quad (1)$$
$$w_0 + w_1 - w_2 < 0 \quad (2)$$
$$w_0 - w_1 + w_2 < 0 \quad (3)$$
$$w_0 + w_1 + w_2 < 0 \quad (4)$$
$$(2) + (3) \rightarrow 2w_0 < 0 \rightarrow w_0 < 0 \quad (5)$$
$$(2) + (4) \rightarrow 2w_0 + 2w_1 < 0 \rightarrow w_1 < 0 \quad (6)$$
$$(3) + (4) \rightarrow 2w_0 + 2w_2 < 0 \rightarrow w_2 < 0 \quad (7)$$




- Table [1]
- Equations 1 to 4 [1]
- Equations 5 to 7 [1]
- Any value of weight and threshold satisfying equations 5 to 7 [1]
- Perceptron diagram [1]

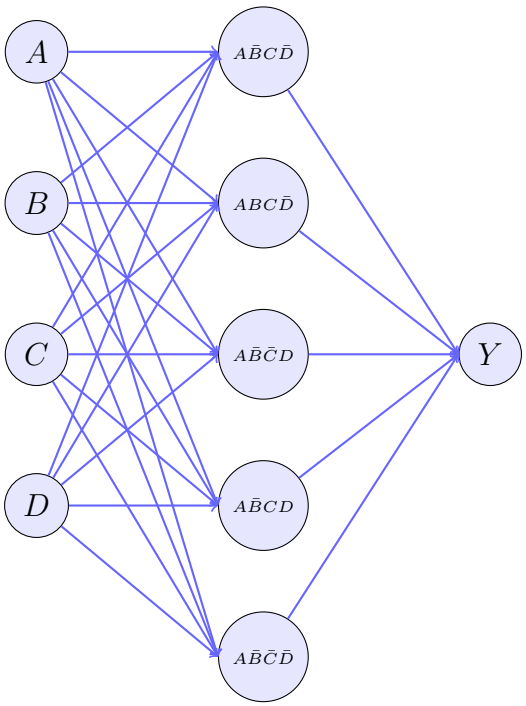
2. Construct an MLP for the given boolean expression. What is the depth and width of the MLP? [5]

| A | B | C | D | Y |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |

- Write the boolean expression [1]

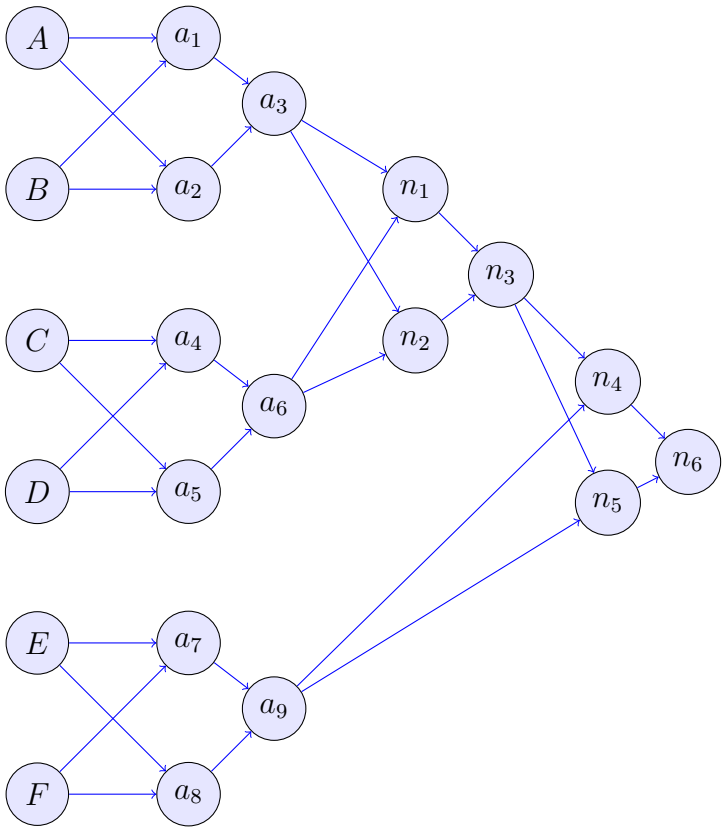
$$Y = A\bar{B}C\bar{D} + ABC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD + A\bar{B}\bar{C}\bar{D}$$

- Draw one node for each expression in layer 1 [1]
- Connect all nodes to a single node in layer 2 [1]
- Depth = 2 [1]
- Width = 5 [1]



3. Find the number of layers and number of perceptrons required for an MLP that can be used for computing the XOR of 6 parameters. Draw the MLP network. If a single hidden layer is used, how many perceptrons will be required? [5]

- n = 6
- Number of perceptrons $= 3(6-1) = 3(6-1) = 15$ [1]
- Number of layers $= 2\log_2 6 = 2\log_2 6 = 2 * \log 6/\log 2 = 2 * ceil(2.5) = 2 * 3 = 6$ [1]
- Network - multiple answers expected. [2]
- Single layer; number of perceptrons $= 2^{n-1} = 2^5 = 32$ [1]



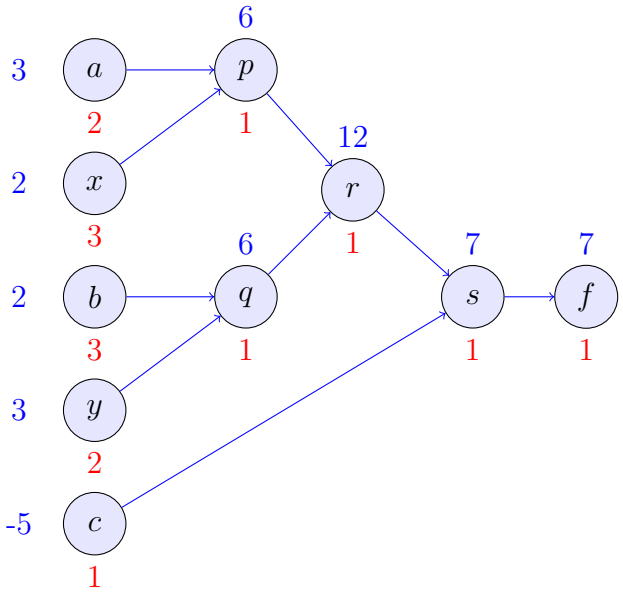

4. Derive the equation for the derivative of binary cross entropy loss $L$ with respect to the weighted sum $Z$, assuming activation $a = \sigma(Z)$ and the loss $L$ is computed from this activation $A$. [5]
Each equation, 1 mark.

$$
\begin{aligned}
L &= -[Y \cdot \log(A) + (1-Y) \cdot \log(1-A)] \\
\frac{\partial L}{\partial Z} &= \frac{\partial L}{\partial A} \cdot \frac{\partial A}{\partial Z} \\
\frac{\partial L}{\partial A} &= \frac{-y}{A} + \frac{1-Y}{1-A} \\
\frac{\partial A}{\partial Z} &= A \cdot (1-A) \\
\frac{\partial L}{\partial Z} &= \left(\frac{-Y}{A} + \frac{1-Y}{1-A}\right) A \cdot (1-A) = (A-Y)
\end{aligned}
$$

5. Draw the computational graph for the equation $f = relu(ax+by+c)$. Show the computations of derivatives of $f$ wrt $a, b, c$ in the graph. Using the graph, compute the value of $f$ and the derivatives if $a = 3, b = 2, c = (-5), x = 2$ and $y = 3$. [5]

- FP graph (blue digits) 1 mark
- BP graph (red digits) 1 mark
- Output computation 1 mark
- Gradient computation 1 mark

$$
\begin{array}{lll}
p = ax & \dfrac{\partial p}{\partial a} & = x \\
q = by & \dfrac{\partial q}{\partial b} & = y \\
r = p + q & \dfrac{\partial r}{\partial p} = \dfrac{\partial r}{\partial q} & = 1 \\
s = r + c & \dfrac{\partial s}{\partial r} = \dfrac{\partial s}{\partial c} & = 1 \\
f = relu(s) & \dfrac{\partial f}{\partial s} & = 1
\end{array}
$$




6. Given an error surface,

$$E(p, q, r) = 3p^3 + 3q^2 + 4r + 5$$

compute the optimal value of learning rate that will minimize the error surface, the largest learning rate for convergence and smallest learning rate for divergence. [5]

$$\frac{\partial E}{\partial p} = 9p \quad \frac{\partial E}{\partial q} = 6q \quad \frac{\partial E}{\partial r} = 4 \qquad 1 \text{ mark}$$

$$\eta_p = 1/9 \quad \eta_q = 1/6 \quad \eta_r = 1/4 \qquad 1 \text{ mark}$$

Optimal learning rate for convergence $\eta_{opt} = \min[\eta_p, \eta_q, \eta_r] = 1/9$ 1 mark

Largest learning rate for convergence $= \min[2\eta_p, 2\eta_q, 2\eta_r] = 2/9$ 1 mark

Smallest Learning rate for divergence $> 2\eta_{opt} = 2 * 1/9 = 0.22$ 1 mark

# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

**M. Tech. in AIML**

**II Semester 2022-2023**

**Mid-Semester Test**
**(EC2 - Makeup)**

| | |
|---|---|
| Course Number | AIMLCZG511 |
| Course Name | Deep Neural Networks |
| Nature of Exam | Open Book |
| Weight-age for grading | 30 |
| Duration | 2 hrs |
| Date of Exam | |

| | |
|---|---|
| * Pages | 3 |
| * Questions | 5 |

---

1. A neural network designed using Tensorflow Keras is given below. Students are instructed to type the answers in the textbox of the portal.

```
net = tf.keras.layers.Sequential()
net.add(
        tf.keras.layers. InputLayer(input_shape = ((256*256*3),)),
        tf.keras.layers.Dense(1048, activation='relu'),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.4),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.6),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(16, activation='relu'),
        tf.keras.layers.Dense(8, activation='softmax'))
```

   (a) What is the objective of the neural network? What is the input given to the network? What is the expected output? How deep and wide is the network. [2]

   (b) Justify the choice activation function in output layer. Instead of Relu activation function, justify the choice of using Tanh activation function. [1]

   (c) Two dropout statements are added in the code. How many additional parameters are learned because of this? If drop out is added after the last statement in the given code, how will it affect the network? [1]

   (d) Write the code snippet for adding the optimizer of your choice. Justify the choice of the optimizer. Assume any other relevant information. [1]

   (e) What will the following code snippet do to the network. [1]

```
cb = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=7)
history = net.fit(epochs=200, batch_size=32, callbacks=[cb])
```

Rubrics

(a) Objective: 8-class classification of 256 by 256 color images 0.5 marks
Input is 256 by 256 colour image and Output is 8 class classification 0.5 marks
Depth = 8 Width = 1048 1 marks

(b) Relu for hidden layers as input is image. Relu as computationally efficient. 0.5 marks
Tanh is computationally expensive when compared to Relu.

(c) 0 additional parameters. 0.5 marks
Adding dropout after last layer will add drop output neurons. Classification may be incorrect. 0.5 marks

(d) Correct code 0.5 marks
Justification of optimizer 0.5 marks

(e) Stops the training when last 7 historical loss values are converging.

2. (a) Figure 1 below plots the loss when batch gradient descent is used for training. Which optimizers plots the loss in figure 2 and 3. [1]

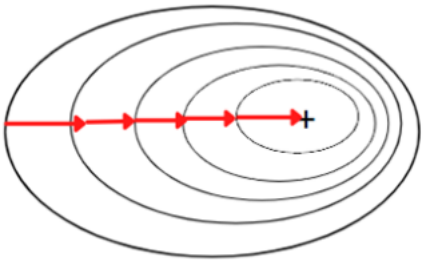


Figure 1

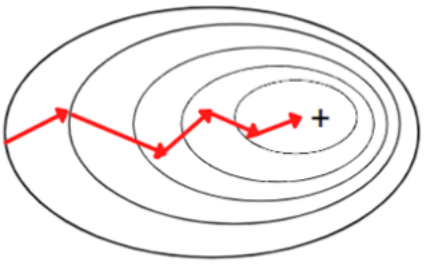


Figure 2

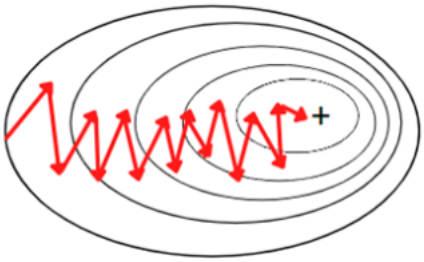


Figure 3

Figure 1

(b) In figure 2, assume that the learning rate used was 0.5. Redraw the plot to show the effect of increasing the learning rate to 2 and decreasing the learning rate to 0.01. [2]

(c) You are given a simple neural network with a single hidden layer containing two neurons, and an output layer containing one neuron. All neurons use the sigmoid activation function. The weights and biases of the network are as follows: [3]

- Weights from input to hidden layer: $w_1 = 0.5, w_2 = -0.6$
- Biases in hidden layer: b1 = 0.1, b2 = -0.2
- Weights from hidden layer to output: w3 = 0.7, w4 = -0.8
- Bias in output layer: b3 = 0.3

Given an input x = 0.75, calculate the output of the network and mean squared loss if the desired output is 1.25. Use the sigmoid activation function. What will be the effect in the loss if the loss function used is

$$L(w, b) = \frac{1}{2}(d - \hat{y})^2 + \lambda||w^2||$$

Rubrics

(a) Fig 2 mini batch sgd and fig 3 is sgd. 2*0.5 marks

(b) Increased learning rate: may not converge or diverge 0.5 marks
Decreased learning rate: slow learning, more oscillations 0.5 marks

(c)

$$
\begin{aligned}
h1 &= sigmoid(w1 * x + b1) = sigmoid(0.5 * 0.75 + 0.1) = 0.6168 \qquad 0.5 \text{ marks} \\
h2 &= sigmoid(w2 * x + b2) = sigmoid(-0.6 * 0.75 - 0.2) = 0.3427 \qquad 0.5 \text{ marks} \\
output &= sigmoid(w3 * h1 + w4 * h2 + b3) \\
&= sigmoid(0.7 * 0.6168 + (-0.8) * 0.3427 + 0.3) = 0.6125 \qquad 0.5 \text{ marks} \\
mse &= 0.5 * (1.25 - 0.6125)^2 = 0.2032 \qquad 0.5 \text{ marks} \\
L(w, b) &= 0.5 * (1.25 - 0.6125)^2 + \lambda[0.5^2 + (-0.6)^2 + 0.7^2 + (0.8)^2] \\
&= 0.2032 + \lambda\, 1.74 \qquad 0.5 \text{ marks}
\end{aligned}
$$

Assume Lambda value in the above equation. This introduces more loss, reducing overfitting. 0.5 marks

3. (a) Consider a two input XNOR gate and simulate a perceptron algorithm for it, where, learning rate=0.02 and threshold = 0.2. [2]

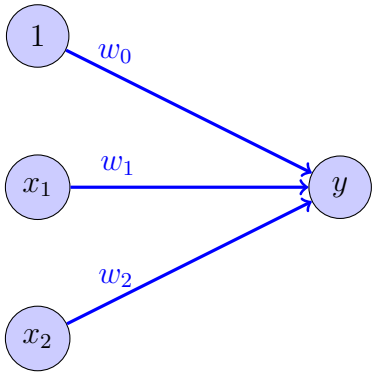
(b) Represent using a multilayer neural network $A \odot B \odot C \odot D \odot E$ where $\odot$ represents XNOR. What will be the optimal depth and width for this network. [3]

(c) You'd like to train a fully-connected neural network with 7 hidden layers, each with 8 hidden units. The input is 30-dimensional and the output is a binary. What is the total number of trainable parameters in your network? [1]

Rubrics

(a)

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

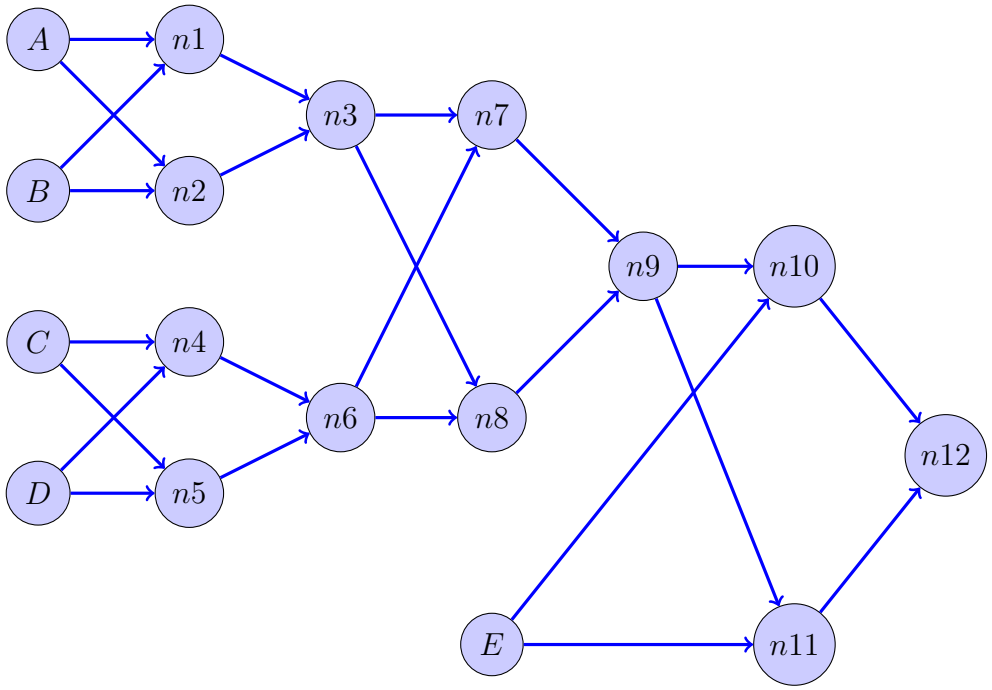


$$
\begin{aligned}
\text{Equations } y &= sign(w_1x_1 + w_2x_2 + w_0) \\
w_i &= w_i + \eta(d - y)x_i \\
\text{Assume } w_1 &= w_2 = 0 \\
\text{First input } y &= sign(0 + 0 - 0.2) = -1 \qquad \text{equals d(x). No parameter update} \\
\text{Second input } y &= sign(0 + 0 - 0.2) = -1 \qquad \text{not equals d(x). Parameter update} \\
w_1 &= 0 + 0.02 * (1 - (-1)) * (-1) = -0.04 \\
w_2 &= 0 + 0.02 * (1 - (-1)) * (+1) = 0.04 \\
w_0 &= -0.2 + 0.02 * (1 - (-1)) = -0.16 \\
\text{Third input } y &= sign(-0.04 * 1 + 0.04 * (-1) - 0.16) \\
&= -1 \qquad \text{not equals d(x). Parameter update} \\
w_1 &= -0.04 + 0.02 * (1 - (-1)) * (1) = 0.0 \\
w_2 &= 0.04 + 0.02 * (1 - (-1)) * (-1) = 0.0 \\
w_0 &= -0.16 + 0.02 * (1 - (-1)) = -0.12 \\
\text{Fourth input } y &= sign(0 * 1 + 0 * 1 - 0.12) = -1 \qquad \text{equals d(x). No Parameter update}
\end{aligned}
$$

(b) XNOR can be represented by 3 neurons in triangular pattern similar to XOR. This is one solution.

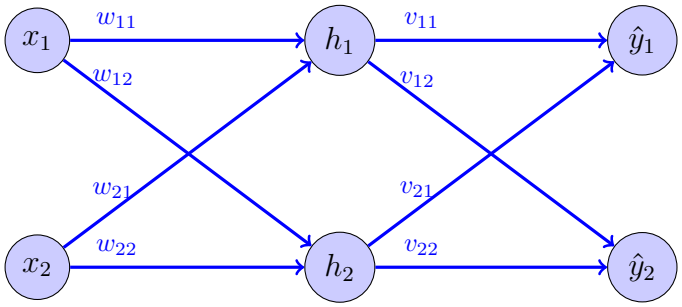
$$
\begin{aligned}
n &= 5 \\
width &= 3(n - 1) = 12 \\
depth &= 2log_2 n = 2 * 3 = 6
\end{aligned}
$$

Diagram 2 marks, width 0.5 and depth 0.5




(c) (30+1)*8 + (8+1)*8*7 + (8+1)*1 or (30+1)*8 + (8+1)*8*7 + (8+2)*2 = 761 or 772

4. Consider the following network structure. You can assume the initial weights. Assume bias to be zero for easier computations. Given that $< x_1, x_2, \hat{y}_1, \hat{y}_2 >=< 1, 1, 0, 1 >$ where $\hat{y}$ is the target. Assume $\beta = 0.9$ and $\eta = 0.01$.




(a) Compute the forward propagation and generate the output. Use Relu for hidden layers and Sigmoid activation function for output layer. [2]

(b) Compute the Softmax loss function for both outputs. [1]

(c) Let the initial weights that assumed be the weights [at time (t-1). Compute the weights $v_{21}$, $w_{12}$ and $w_{22}$ at time $t$ using SGD. [1.5]

(d) Let the weight at time $t$ be the ones computed in part (c). Compute the weights $v_{21}$, $w_{12}$ and $w_{22}$ at $(t+1)$ when momentum is used. [1.5]

Rubrics

(a) Award 1 mark if only equations are written. Substitute assumed weights and compute values, then award 2 marks in total. If assumed weights are all same among multiple students, report to IC.

$$h1 = relu(w_{11} * 1 + w_{21} * 1 + 0) \quad (1)$$
$$h2 = relu(w_{12} * 1 + w_{22} * 1 + 0) \quad (2)$$
$$\hat{y}_1 = sigmoid(v_{11}h_1 + v_{21}h_2) \quad \text{Substitute values of eqs: 1,2} \quad (3)$$
$$\hat{y}_2 = sigmoid(v_{12}h_1 + v_{22}h_2) \quad \text{Substitute values of eqs: 1,2} \quad (4)$$

(b) 1 marks for loss computation. Writing equations alone will be awarded 0 marks.

$$loss \quad L = \frac{e^{\hat{y}_1}}{e^{\hat{y}_1} + e^{\hat{y}_2}}; \frac{e^{\hat{y}_2}}{e^{\hat{y}_1} + e^{\hat{y}_2}}$$

(c) Computation of weights using SGD. Award 1 mark if only equations are written. If all

3 weights of equations 13, 14, 15 computed, then award 3*0.5 marks.

$$\frac{\partial L}{\partial \hat{z}} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial \hat{z}} = \frac{\partial L}{\partial \hat{y}} * 1 \qquad \text{Substitute value of eq: 6} \tag{5}$$

$$\frac{\partial L}{\partial v_{21}} = \frac{\partial L}{\partial \hat{z}} * \frac{\partial \hat{z}}{\partial v_{21}} = \frac{\partial L}{\partial \hat{z}} * h_2 \qquad \text{Substitute value of eq: 7} \tag{6}$$

$$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial \hat{z}} * \frac{\partial \hat{z}}{\partial h_2} = \frac{\partial L}{\partial \hat{z}} * v_{21} \qquad \text{Substitute value of eq: 6} \tag{7}$$

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial h_2} * \frac{\partial h_2}{\partial z_2} = \frac{\partial L}{\partial h_2} * 1 \qquad \text{Substitute value of eq: 9} \tag{8}$$

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial w_{12}} = \frac{\partial L}{\partial z_2} * x_1 \qquad \text{Substitute value of eq: 10} \tag{9}$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial w_{22}} = \frac{\partial L}{\partial z_2} * x_2 \qquad \text{Substitute value of eq: 10} \tag{10}$$

$$v_{21} = assumed\, v_{21} - \eta \frac{\partial L}{\partial v_{21}} \qquad \text{Substitute value of eq: 8} \tag{11}$$

$$w_{12} = assumed\, w_{12} - \eta \frac{\partial L}{\partial w_{12}} \qquad \text{Substitute value of eq: 11} \tag{12}$$

$$w_{22} = assumed\, w_{22} - \eta \frac{\partial L}{\partial w_{22}} \qquad \text{Substitute value of eq: 12} \tag{13}$$

(d) Computation of weights using Momentum. Award 1 mark if only equations are written. If all 3 weights of equations 16, 17, 18 computed, then award 3*0.5 marks.

$$v_2 = v_2(eqn13) - \eta \frac{\partial L}{\partial v_2} + \beta(v_2(eqn13) - assumed\, v_2) \tag{14}$$

$$w_{12} = w_{12}(eqn14) - \eta \frac{\partial L}{\partial w_{12}} + \beta(w_{12}(eqn14) - assumed\, w_{12} \tag{15}$$

$$w_{22} = w_{22}(eqn15) - \eta \frac{\partial L}{\partial w_{22}} + \beta(w_{22}(eqn14) - assumed\, w_{22} \tag{16}$$

5. (a) Represent the function

$$f(x, y) = x^4 - 32x^2 + y^4 - 18y^2$$

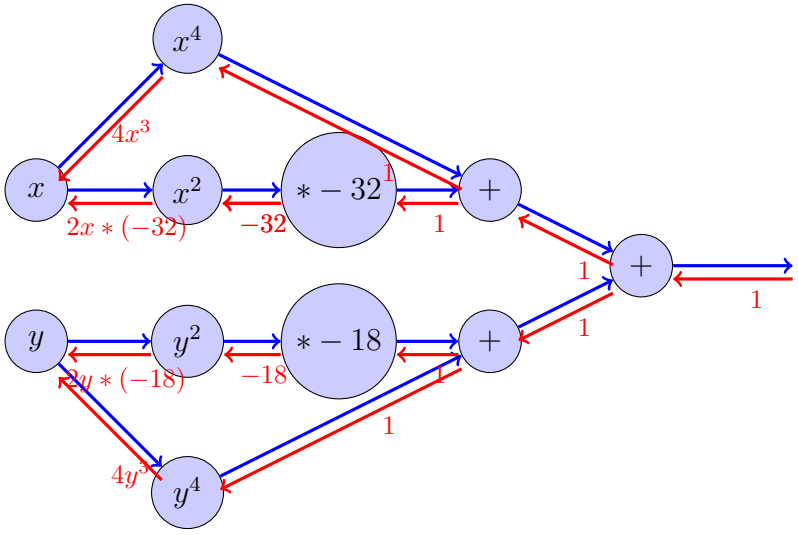
using a computation graph. Evaluating this function at x = 2, and y = 2. Find the first derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using computation graph through backpropagation. [3]

(b) Compute the Hessian of the function

$$f(x, y) = x^4 - 32x^2 + y^4 - 18y^2$$

You are given two point (0,0) and (4,3). Among these two points, which points are the local minima or local maxima or both. [3]

(a) Graph 1.5 mark computing of partial derivatives using graph 1.5 marks




$$f(x,y) = x^4 - 32x^2 + y^4 - 18y^2$$
$$\frac{\partial f}{\partial x} = 4x^3 - 64x$$
$$\frac{\partial f}{\partial y} = 4y^3 - 36x$$

$$f(x,y) = x^4 - 32x^2 + y^4 - 18y^2$$
$$H = \begin{bmatrix} 12x^2 - 64 & 0 \\ 0 & 12y^2 - 36 \end{bmatrix}$$
$$\nabla^2 f(0,0) = \begin{bmatrix} -64 & 0 \\ 0 & -36 \end{bmatrix}$$
$$\nabla^2 f(4,3) = \begin{bmatrix} 128 & 0 \\ 0 & 72 \end{bmatrix}$$

(b) local minimum is (4,3) and point of local maximum is (0,0)
Local minima 1.5 marks and local maxima 1.5 marks

# Birla Institute of Technology and Science, Pilani

**Work Integrated Learning Programmes Division**

**M. Tech. in AIML**

**II Semester 2022-2023**

**Mid-Semester Test**
**(EC2 - Regular)**

| | |
|---|---|
| Course Number | AIMLCZG511 |
| Course Name | Deep Neural Networks |
| Nature of Exam | Open Book |
| Weight-age for grading | 30 |
| Duration | 2 hrs |
| Date of Exam | |

| | |
|---|---|
| * Pages | 3 |
| * Questions | 4 |

---

1. (a) You'd like to train a fully-connected neural network with 5 hidden layers, each with 10 hidden units. The input is 20-dimensional and the output is a scalar. What is the total number of trainable parameters in your network? [1]

   (b) You would like to train a dog/cat image classifier using mini-batch gradient descent. You have already split your dataset into train, dev and test sets. The classes are balanced. You realize that within the training set, the images are ordered in such a way that all the dog images come first and all the cat images come after. A friend tells you: "you absolutely need to shuffle your training set before the training procedure." Is your friend, right? Justify. [1]

   (c) You want to evaluate the classifier you trained in part(b). Your test set $(X_{test}, Y_{test})$ is such that the first $m_1$ images are of dogs, and the remaining $m_2$ images are of cats. After shuffling $(X_{test}, Y_{test})$, you evaluate your model on it to obtain a classification accuracy $\alpha_1\%$. You also evaluate your model on $(X_{test}, Y_{test})$ without shuffling to obtain accuracy $\alpha_2\%$. What is the relationship between $\alpha_1\%$.and $\alpha_2\%$. $(\geq, \leq, =, <, >)$? Justify. [1]

   (d) You are designing a deep learning system to detect driver fatigue in cars. It is crucial that that your model detects fatigue, to prevent any accidents. Which of the following is the most appropriate evaluation metric: Accuracy, Precision, Recall, Loss Value. Justify your choice. [1]

   (e) You want to solve a classification task. You first train your network on 20 samples.Training converges, but the training loss is very high. You then decide to train this network on 10,000 examples. Is your approach to fixing the problem correct? If yes, explain the most likely results of training with 10,000 examples. If not, give a solution to this problem. [2]

   (f) Justify the use of first momentum and second moment in Adam optimizer. [1]
   (PS: Ensure that you are writing your own answer in one or two lines for all the parts. Inspired or copied answers will be penalized. )

Rubrics

(a) (20+1)*10 + (10+1)*10*4 + (10+1)*1 [1 mark]

(b) The friend is incorrect. The optimization is much harder with minibatch gradient descent because the loss function moves by a lot when going from the one type of image to another. [1 mark]

(c) $\alpha_1 = \alpha_2$. When evaluating on the test set, the only form of calculation that you do is a single metric (e.g. accuracy) on the entire test set. The calculation of this metric on the entire test set does not depend on the ordering. [1 mark]

(d) Recall and justification [0.5 + 0.5 mark]

(e) The model is suffering from bias. Increasing the amount of data reduces the variance, and is not likely to solve the problem. A better approach would be to decrease the bias of the model by maybe adding more layers/ learnable parameters. It is possible that training converged to a local optimum. Training longer/using a better optimizer/ restarting from a different initialization could also work. [2 marks]

(f) First Moment (Mean): It calculates the moving average of the gradients of the parameters over time. By considering the momentum, the optimizer can continue moving in the right direction, even when gradients change rapidly, leading to more stable and consistent updates.
Second Moment (Variance): It calculates the moving average of the squared gradients. Squaring the gradients emphasizes large gradients and dampens small ones. High variance in gradients needs smaller learning rates and low variance may indicate smoother regions where larger learning rates can be safely applied.

2. Given N training data points $\{(x^k, y^k)\}, k = 1 : N, x^k \in \mathcal{R}^d$, and labels in $y^k \in \{-1, 1\}$, we seek a linear discriminant function $f(x) = w \cdot x$ optimizing the loss function $L(z) = e^{-z}$ , for $z = yf(x)$.

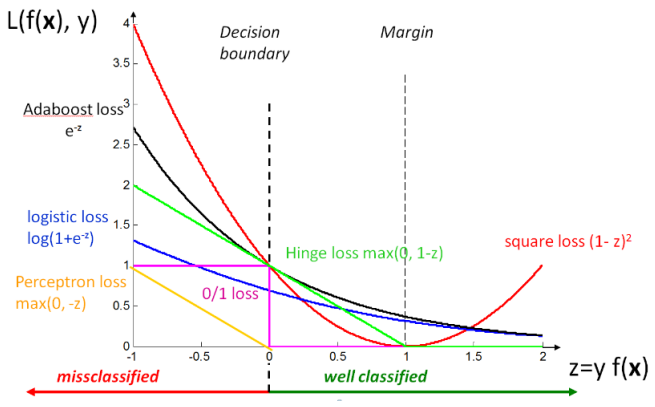
(a) Is $L(z)$ a large margin loss function? Justify your answer with a graphical representation. [2]

(b) Derive the stochastic gradient descent update $\Delta w$ for $L(z)$. [3]

(c) Represent the computation of the stochastic gradient descent using a computation graph, using general equations. [2]

## Rubrics

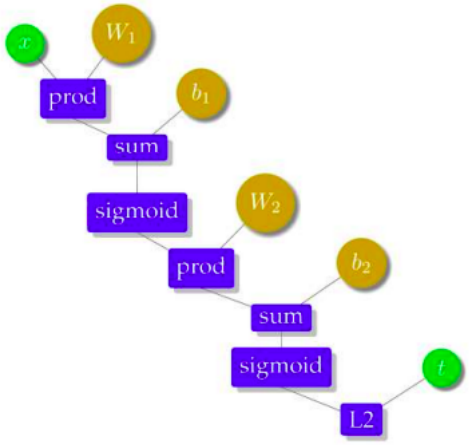
(a) Yes. This is because the loss penalizes even examples that are well classified, but the penalty decreases as you go away from the decision boundary. [1 mark]
Graph [1 mark]



(b) For a learning rate $\eta \geq 0$, and for $z = yf(x) = y\sum_{i=1}^{d} w_i x_i$

$$\begin{aligned} \nabla w_i &= -\eta \frac{\partial L}{\partial w_i} && \text{1 mark} \\ &= -\eta \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_i} && \text{1 mark} \\ &= \eta e^{-z} y x_i && \text{0.5 mark} \\ \nabla w &= \eta e^{-z} y x && \text{0.5 mark} \end{aligned}$$

(c) FP 1 mark and BP (reverse arrows) 1 mark.




3. (a) Consider the DNN model using Adam optimizer with the loss function $L = 3w_1^2 + 1.5w_2^3$ with the weights where $w_1 = 1.5$ and $w_2 = -2$ at time (t-1). Suppose the first moment vector $v = [0, 0]$ and second moment vector $s = [0, 0]$, learning rate = 0.01, decay rate 1=0.5 and decay rate 2 = 0.9 respectively with $epsilon\epsilon = 10^{-8}$. Calculate the weight vector $w$ for time $t$ after the first iteration. [3]

(b) Suppose that the first momentum and second moment updates can be written as the mean and variance of the gradients respectively for not requiring the bias correction in Adam optimizer. Then calculate the weight vector $w$ for time $t$ for the first iteration only using the above parameters mentioned in part a) with $w_1 = 1.5$ and $w_2 = -2$. (Assume the first momentum and second moment values by your own and mention the same while calculation.) [3]

(c) Investigate the differences in the weights $w_t$ obtained in parts a) and b) and provide your analysis. [2]

Rubrics

(a)

$$\nabla w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial(3w_1^2 + 1.5w_2^3)}{\partial w_1} = 6w_1 = 6 * 1.5 = 9$$
$$\nabla w_2 = \frac{\partial L}{\partial w_2} = \frac{\partial(3w_1^2 + 1.5w_2^3)}{\partial w_2} = 4.5w_2^2 = 4.5 * (-2)^2 = 18$$

First Moment [1 mark]

$$v_1 = \beta_1 * v_1 + (1 - \beta_1) * \nabla w_1 = 0.5 * 0 + (1 - 0.5) * 9 = 4.5$$
$$v_2 = \beta_1 * v_2 + (1 - \beta_1) * \nabla w_2 = 0.5 * 0 + (1 - 0.5) * 18 = 9$$
$$v_t = [4.5, 9]$$

Second Moment [1 mark]

$$s_1 = \beta_2 * s_1 + (1 - \beta_2) * \nabla w_1^2 = 0.9 * 0 + (1 - 0.9) * 9^2 = 8.1$$
$$s_2 = \beta_2 * s_2 + (1 - \beta_2) * \nabla w_2^2 = 0.9 * 0 + (1 - 0.9) * 18^2 = 32.4$$
$$s_t = [8.1, 32.4]$$

No bias correction. Only weight update. [1 mark]

$$w_{t-1} = [1.5, -2]$$
$$w_t = w_{t-1} - \frac{\eta v}{\sqrt{s + \epsilon}}$$
$$w_1 = 1.5 - \frac{0.01 * 9}{\sqrt{8.1 + 10^{-8}}} = 1.468$$
$$w_2 = -2 - \frac{0.01 * 18}{\sqrt{32.41 + 10^{-8}}} = -2.032$$
$$w_t = [1.468, -2.032]$$

(b) Assume initial v and s vectors and use that for computation.
First Moment [1 mark]

$$v = \beta_1 * v_0 + (1 - \beta_1) * \nabla w$$
$$v_t = [v_1, v_2]$$

Second Moment [1 mark]

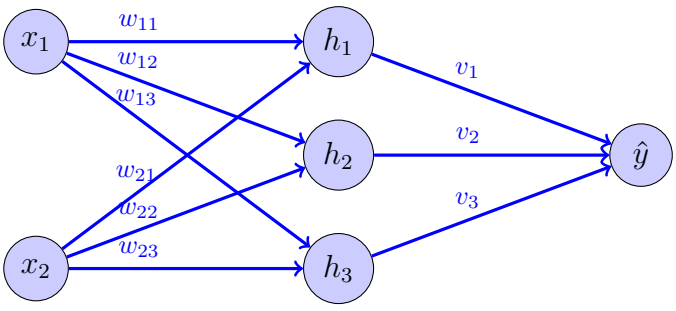
$$s = \beta_2 * s_0 + (1 - \beta_2) * \nabla w^2$$
$$s_t = [s_1, s_2]$$

Apply bias correction and then update weights. [1 mark]

$$w_{t-1} = [1.5, -2]$$
$$w_t = w_{t-1} - \frac{\eta v}{\sqrt{s + \epsilon}}$$
$$w_t = [w_1, w_2]$$

(c) May result in faster convergence when non-zero momentums are assumed, but cant be verified for all initial assumptions. [2 mark]

4. Consider the following network structure. You can assume the initial weights. Assume bias to be zero for easier computations. Given that $< x_1, x_2, y >=< 1, 1, 0 >$ where $y$ is the target. Assume $\eta = 0.01$.



(a) Compute the forward propagation and generate the output. Use Relu for hidden layers and Sigmoid activation function for output layer. [2]

(b) Compute the loss and its derivative. You can assume the loss function. [1]

(c) Let the initial weights that assumed be the weights [at time (t-1). Compute the weights $v_1$, $w_{11}$ and $w_{21}$ at time $t$ using SGD. [1.5]

(d) Assume you are applying drop out for the hidden layer. For odd BITS id, assume that h3 is dropped out and for even BITS id, assume that h2 is dropped out. Computing the weights $v_1$, $w_{11}$ and $w_{21}$ at time $t$ after dropping out the the neuron. [1.5]

(e) Investigate the differences in the weights $w_t$ obtained in parts c) and d) and provide your analysis. [2]

Rubrics

(a) Award 1 mark if only equations are written. Substitute assumed weights and compute values, then award 2 marks in total. If assumed weights are all same among multiple students, report to IC.

$$h1 = relu(w_{11} * 1 + w_{21} * 1 + 0) \tag{1}$$
$$h2 = relu(w_{12} * 1 + w_{22} * 1 + 0) \tag{2}$$
$$h3 = relu(w_{13} * 1 + w_{23} * 1 + 0) \tag{3}$$
$$\hat{y} = sigmoid(v_1 h_1 + v_2 h_2 + v_3 h_3) \quad \text{Substitute values of eqs: 1,2,3} \tag{4}$$

(b) Loss should be either RMSE or binary cross entropy. 0.5 marks for loss computation and 0.5 marks for derivative computation. Writing equations alone will be awarded 0 marks.

$$loss \quad L = \frac{1}{2}(y - \hat{y})^2 \quad \text{Substitute value of eq: 4} \tag{5}$$
$$gradient \quad \frac{\partial L}{\partial \hat{y}} = (y - \hat{y}) \quad \text{Substitute value of eq: 4} \tag{6}$$

OR

$$loss \quad L = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$
$$gradient \quad \frac{\partial L}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1 - y}{1 - \hat{y}}$$

(c) Computation of weights using SGD. Award 1 mark if only equations are written. If all

3 weights of equations 13, 14, 15 computed, then award 3*0.5 marks.

$$\frac{\partial L}{\partial \hat{z}} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial \hat{z}} = \frac{\partial L}{\partial \hat{y}} * 1 \qquad \text{Substitute value of eq: 6} \tag{7}$$

$$\frac{\partial L}{\partial v_2} = \frac{\partial L}{\partial \hat{z}} * \frac{\partial \hat{z}}{\partial v_2} = \frac{\partial L}{\partial \hat{z}} * h_2 \qquad \text{Substitute value of eq: 7} \tag{8}$$

$$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial \hat{z}} * \frac{\partial \hat{z}}{\partial h_2} = \frac{\partial L}{\partial \hat{z}} * v_2 \qquad \text{Substitute value of eq: 6} \tag{9}$$

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial h_2} * \frac{\partial h_2}{\partial z_2} = \frac{\partial L}{\partial h_2} * 1 \qquad \text{Substitute value of eq: 9} \tag{10}$$

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial w_{12}} = \frac{\partial L}{\partial z_2} * x_1 \qquad \text{Substitute value of eq: 10} \tag{11}$$

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial z_2} * \frac{\partial z_2}{\partial w_{22}} = \frac{\partial L}{\partial z_2} * x_2 \qquad \text{Substitute value of eq: 10} \tag{12}$$

$$v_2 = assumed\, v_2 - \eta \frac{\partial L}{\partial v_2} \qquad \text{Substitute value of eq: 8} \tag{13}$$

$$w_{12} = assumed\, w_{12} - \eta \frac{\partial L}{\partial w_{12}} \qquad \text{Substitute value of eq: 11} \tag{14}$$

$$w_{22} = assumed\, w_{22} - \eta \frac{\partial L}{\partial w_{22}} \qquad \text{Substitute value of eq: 12} \tag{15}$$

(d) Add dropout according to BITS id. If not matching with BITS id, award zero. Recompute the values of $\hat{y}$, loss, gradients and the weight updates.

(e) Check if part d is according to BITS id. If not matching with BITS id, award zero.