Roll No: _____

CSE641: Deep Learning (2023)
Mid-Sem Examination

Duration: 1 hr **Total marks: 30**

PART A: Mandatory **15 marks**

1. In dropout, we randomly select neurons with dropout probability (d_p) to switch them off during the training phase. To compensate for the loss of *information* at the inference time, we either multiply y_i by the dropout probability (d_p) during the testing phase (normal dropout) or we multiply it by $1/d_p$ during the training phase (inverse dropout).

   Dropout:

   Training $\hat{y}_i^{(1)} = r \,.\, y_i^{(1)}$ Testing $\hat{y}_i^{(1)} = y_i^{(1)} * d_p^{(1)}$

   Inverse dropout:

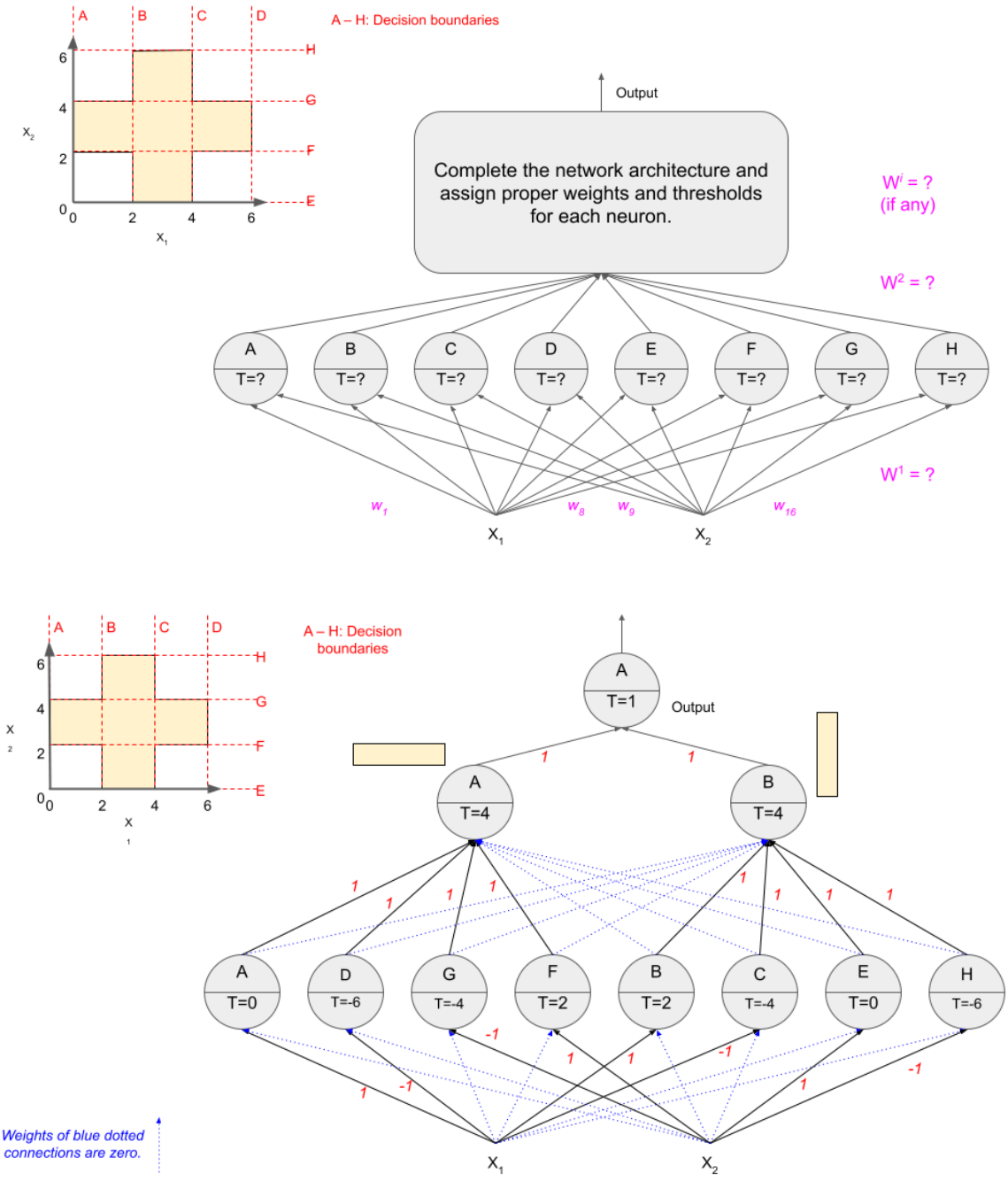
   Training $\hat{y}_i^{(1)} = \frac{r \,.\, y_i^{(1)}}{d_p}$ Testing $\hat{y}_i^{(1)} = y_i^{(1)}$

   Assuming we don't want to perform either operations, can you suggest an alternative, albeit an expensive one, to achieve the same goal? **[5]**

   A costly but equivalent setup is:
   Repeat a-c for each sample i in test set:
   a. Obtain different models all with different neurons set to dropout (m1, m2, m3,…,mN). After training save each individual model.
   b. During test load all the N models, for an input i, obtain the the output label y1,y2,...,yN from each nth model for the input i.
   c. Now the final label can be mode or mean of the label list.

2. For the following function (1 in the shaded regions; 0 elsewhere), complete the network architecture and assign proper weights and threshold for each neuron. **Note:** Accuracy of the network has preference over the size of the network. **[10]**

Roll No: _____



PART B: Choose any combination of questions for 15 marks. DO NOT mix different subparts to combine.

3. **History of DL:** Hebb, Minsky and Papert, Hinton, Aristotle, Rosenblat, McCulloch and Pitts, and Alexander Bain.
    a. Order the above scientists based on their groundbreaking theories/models in the year-wise fashion (Earlier the first). **[3]**
    b. Also, mention their groundbreaking theories/models (~1 line each). **[7]**

Roll No: _____

Ans:

Aristotle → Laws of association
Alexander Bain →Information is in the connections
McCulloch and Pitts → Mathematical model of neurons.
Hebb → "*Neurons that fire together wire together*" and the weight update rule (additive),
Rosenblat → PTA
Minsky and Papert → Perceptron can't compute non-linear functions.
Hinton → Backpropagation
[0.4m for each correct order, 3m for complete correct order]

4. Write steps of PTA with equations. Assume necessary variables. **[5]**
   - **a.** Randomly initialize w. *[1m]*
   - **b.** Repeat for n epochs or until the termination condition
     - **i.** For each sample <x, y>, where $y \in \{+1, -1\}$ [2m]
       - If ($ywx \leq 0$) // Error condition [1m]
         - $w = w + yx$ *[1m]*

5. The dying-negative gradient (zero gradient for the negative region) is a major issue with RELU **[max(x, 0)]**. To overcome it, a new activation function SILU [$\rho(x) = x * \sigma(x)$], where $\sigma(x)$ is the sigmoid function, has been introduced. **[1+1+2+1]**

   a. Why do we prefer **non-linear** activation functions for neural networks**?**

      A combination of linear functions no matter the depth can be reduced to a single linear function, limiting the learning ability of the model. We thus use non-linear activation functions to better capture the non-linearlity of the underlying function.

   b. Why do we prefer activation functions where derivatives can be written in terms of the function itself?

      During forward pass we already calculate the values for the activation function. If its derivate can be written in terms of the function itself, then by simply mathematically operations/combination on the activation function, we reduce the computation cost for backpropagation.

   c. Show step-wise how the derivative $\rho'(x)$ can be expressed in terms of $\sigma(x)$?

      Given that $\sigma'(x) = \sigma(x)(1-\sigma(x))$ By multiplication rule

      $\rho'(x) = x'*\sigma(x) + x*\sigma'(x)$

Roll No: _____

= 1*$\sigma(x)$ + x*$\sigma(x)$(1-$\sigma(x)$)

= $\sigma(x)$[1+x(1-$\sigma(x)$)] or

d. Intuitively, for large values of x, $\rho(x)$ **and** $\rho'(x)$ behave like which activation functions? **Note:** No derivation needed here.
   1. For large values of x, $\sigma(x)$ ~ 1, thus $\rho(x) \sim x$ and the function mimics RELU activation.
   2. For large values of x, $\sigma(x)$ ~ 1, or (1-$\sigma(x)$~ 0 and overall [1+x(1-$\sigma(x)$)] ~ 1, $\rho'(x)$ function mimics Sigmoid activation.

6. **Optimizers:**
   a. Mention the most fundamental difference between NAG and RMSProp. **[1]**

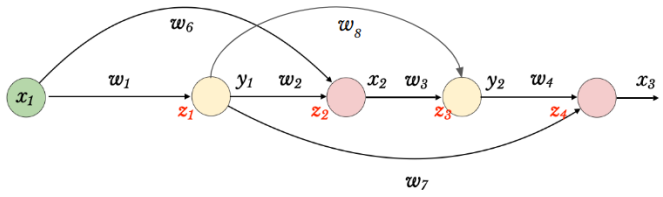
      Non-adaptive vs Adaptive: Single step-size for each parameter vs different (learnable) step-size for different parameters.

      Other valid differences will also considered correct.

   b. Considering the error(w) vs w graph of a convex function, discuss possible configurations based on the step size (η) vs optimal step size (η^{opt}). **[4]**

      Refer to optimizer slides: One-step convergence ($\eta = \eta^{opt}$), Monotonically convergence ($\eta < \eta^{opt}$), Oscillating convergence ($\eta^{opt} < \eta < 2\eta^{opt}$), and divergence ($\eta > 2\eta^{opt}$).

7. Assuming $\delta L/\delta x3$ is known, write the weight update equation for $w1$ ($\delta L/\delta w1$ should be in the expanded form). Input $x1$ and all weights are positive. All neurons have ReLU activation. **[5]**



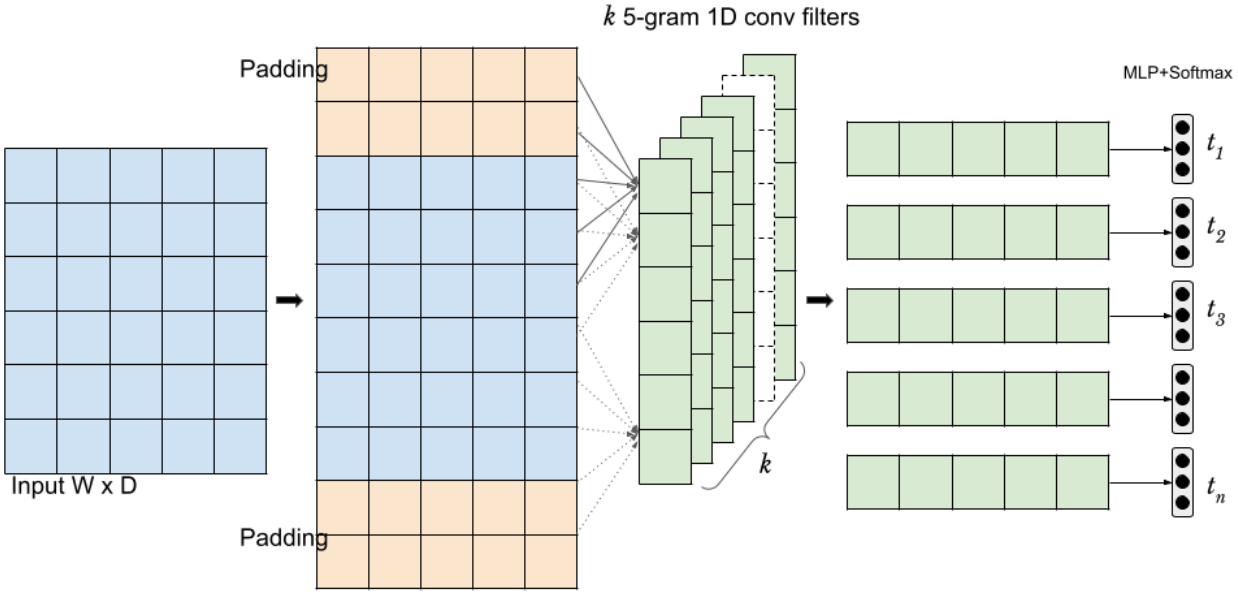
Roll No: _____

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial w_1}$$
$$= \{\frac{\partial L}{\partial z_4}\frac{\partial z_4}{\partial y_1} + \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial y_1} + \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial y_1}\}\frac{\partial y_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$
$$= \{\frac{\partial L}{\partial x_3}\frac{\partial x_3}{\partial z_4}\frac{\partial z_4}{\partial y_1} + \frac{\partial L}{\partial z_4}\frac{\partial z_4}{\partial y_2}\frac{\partial y_2}{\partial z_3}\frac{\partial z_3}{\partial y_1} + \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial x_2}\frac{\partial x_2}{\partial z_2}\frac{\partial z_2}{\partial y_1}\}.1.x_1$$
$$= \{\frac{\partial L}{\partial x_3}.1.w_7 + \frac{\partial L}{\partial x_3}.w_4.1.w_8 + \frac{\partial L}{\partial x_3}.w_4.w_3.1.w_2\}.x_1$$
$$= \frac{\partial L}{\partial x_3}(w_7 + w_4w_8 + w_4w_3w_2).x_1$$
$$\text{Given}\frac{\partial y_1}{\partial z_1} = \frac{\partial x_2}{\partial z_2} = \frac{\partial y_2}{\partial z_3} = \frac{\partial x_3}{\partial z_4} = 1$$

8. Design (Draw) a CNN-based architecture for the part-of-speech (PoS) tagger. Assume necessary hyperparameters. **[5]**

Input: A sentence S = $\{w_1, w_2, \dots, w_n\}$

Output: A tag sequence T = $\{t_1, t_2, \dots, t_n\}$



- Padding is required to maintain the input size (#of words) even after convolution.
- Similarly, we can have different n-gram filters. For each of then, padding will be different. Finally, for each word, the convoluted features of all filter sizes can be flattened before feeding into the MLP+Softmax, which computes probability distribution over the possible PoS tags.