CSE 493/599 G1
Deep Learning
Spring 2023 Practice Midterm Exam

April 25, 2023

Full Name: _____

Net ID (Not Number): _____

Question		Score
True/False	(20 pts)	
Multiple Choice	(40 pts)	
Short Answer	(40 pts)	
Total	(100 pts)	

Welcome to the CSE 493/599 G1 Midterm Exam!

- The exam is 1 hour 20 minutes and is **double-sided**.
- No notes or electronic devices are allowed.

I understand and agree to uphold the University of Washington Honor Code during this exam.

Signature: _____ Date: _____

Good luck!

This page is left blank for scratch work only. DO NOT write your answers here.

1 True / False (20 points)

Fill in the circle next to True or False, or fill in neither. Fill it in completely like this: ●. No explanations are required.

Scoring: Correct answer is worth 2 points. To discourage guessing, incorrect answers are worth -1 points. Leaving a question blank will give 0 points.

1.1 Linear classifier can predict accurate result even when the classes in then dataset is not distributed evenly.

◯ True

◯ False

1.2 In order to normalize our data (i.e. subtract mean and divide by standard deviation), we typically compute the mean and standard deviation across the entire dataset before splitting the data into train/val/test splits.

◯ True

◯ False

1.3 Suppose we have trained a softmax classifier (with weights W) that achieves 100% accuracy on our dataset. If we change the weights to $2W$, the classifier will maintain the same accuracy while have a smaller loss (cross entropy loss without regularization).

◯ True

◯ False

1.4 You currently have a small dataset containing images of handwritten digits 0 to 9, and would like to train a model for digit recognition. Each digit has the same number of images in the dataset. It is a good idea to augment the data and increase the size of the dataset by randomly flipping each image horizontally and vertically and adding the resulting image to the dataset.

◯ True

◯ False

1.5 Recall that in Batch Normalization, the model has the capacity to learn the Beta and Gamma parameters such that at test time, these parameters can exactly undo the effects of mean centering and scaling by the standard deviation. However, in Layer Normalization, there does not exist Beta and Gamma parameters that can do this.

◯ True

◯ False

1.6 For an SVM classifier, initializing W and b to 0 will always result in all the elements in the final learned W to be the same.

◯ True

◯ False

1.7 For a binary classification task, using accuracy as a loss function directly over binary cross-entropy could yield better results in contexts where accuracy is your only metric of interest.

Note: We define accuracy as $\frac{\text{count}(\text{round}(\hat{y})==y)}{N}$, given a real-valued prediction vector $\hat{y} \in [0,1]^N$ and binary ground-truth vector $y \in \{0,1\}^N$.

◯ True

◯ False

1.8 Let $\vec{z}$ be an n-dimensional vector and $s(\cdot)$ be the softmax function. Since softmax normalizes predictions to yield a probability distribution, its output does not change due to a scaling of the input, i.e. for all real numbers c, $s(z) = s(cz)$.

◯ True

◯ False

1.9 Recall that if network activations become saturated or collapse to zero, learning becomes challenging. Such problem can be alleviated with a learned normalization scheme (batch normalization, layer normalization, etc.) without changing the weight matrix of convolution or fully-connected layer.

◯ True

◯ False

1.10 Only recurrent neural networks have vanishing gradients and exploding gradients problems.

◯ True

◯ False

2 Multiple Choices (40 points)

Fill in the circle next to the letter(s) of your choice (like this: ●). No explanations are required. **Choose ALL options that apply.**

Each question is worth 4 points and the answer may contain one or more options. Selecting all of the correct options and none of the incorrect options will get full credits. For questions with multiple correct options, each incorrect or missing selection gets a 2-point deduction (up to 4 points).

2.1 Suppose you are using k - Nearest Neighbor method with L1 distance to classify images, which of the following will **NOT** affect the prediction accuracy?

- ◯ A: Increase the size of training set.
- ◯ B: Increase the value of k.
- ◯ C: Use L2 distance instead.
- ◯ D: Horizontally flip each training image and test image.
- ◯ E: All of the above will affect the prediction accuracy.

2.2 Recall that the hinge loss on i-th input data is defined as $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$, where y_i is correct label of i-th input, s_j is the score for j-th class and Δ is the margin of some non-negative value. Which of the following is/are true about the hinge loss?

- ◯ A: As long as $L_i = 0$ holds for each input, increase the margin won't change the value of L_i.
- ◯ B: The bias term b in the score calculation won't affect the value of L_i,
- ◯ C: The more classes we would like to classify, the higher the hinge loss will be.
- ◯ D: None of the above.

2.3 Given n is the number of samples in the dataset, which of the following is the correct runtime complexity for test-time prediction for KNN and Linear SVM respectively?

- ◯ A: O(1), O(1)
- ◯ B: O(log n), O(1)
- ◯ C: O(n), O(n)
- ◯ D: O(1), O(n)
- ◯ E: O(n), O(1)

2.4 Suppose the input to a **normalization layer** has dimensions (N, H, W, C), where N is the number of data points (i.e. we have layer input $x_1, \ldots, x_N$ and corresponding layer output $y_1, \ldots, y_N$), and H, W, C are the height, width, and number of channels, respectively. For a specific input-output pair y_i, x_j where $i \neq j$, which of the following is/are true about the gradient matrix $\frac{\partial y_i}{\partial x_j}$?

- ◯ A: For a **batch** normalization layer, there are up to C nonzero terms.
- ◯ B: For a **batch** normalization layer, there are up to H^2W^2C nonzero terms.
- ◯ C: For a **batch** normalization layer, there are up to $H^2W^2C^2$ nonzero terms.
- ◯ D: For a **layer** normalization layer, there are up to C nonzero terms.

◯ E: For a **layer** normalization layer, there are up to H^2W^2C nonzero terms.

◯ F: For a **layer** normalization layer, there are up to $H^2W^2C^2$ nonzero terms.

2.5 We will be using gradient descent to optimize our losses in this question. Which of the following losses ($\mathcal{L}(x)$) converge for the given initialization (x_0) and step size ($step$)? Select all that apply. Note: Our update rule is $x_{new} = x_{old} - step * \mathcal{L}'(x_{old})$.

◯ A: $\mathcal{L}(x) = |x|$ for $x_0 = 1,\ step = 2$

◯ B: $\mathcal{L}(x) = x^2$ for $x_0 = 1,\ step = 1$

◯ C: $\mathcal{L}(x) = |x|$ for $x_0 = 1,\ step = 2.01$

◯ D: $\mathcal{L}(x) = x^2$ for $x_0 = 1,\ step = 0.99$

2.6 In mathematics, p-norm is a function from a vector space to the real numbers: $L_p(\mathbf{x}) = \left(\sum_{i=1}^{k} |x_i|^p\right)^{1/p}$, e.g. $L_1(\mathbf{x}) = \sum_{i=1}^{k} |x_i|$, $L_2(\mathbf{x}) = \sqrt{\sum_{i=1}^{k} |x_i|^2}$, $L_{0.5}(\mathbf{x}) = \left(\sum_{i=1}^{k} \sqrt{|x_i|}\right)^2$; specially, $L_\infty(\mathbf{x}) = \max_i |x_i|$
When $p = 1$ and $p = 2$, $L_p(\mathbf{x})^p = \sum_{i=1}^{k} |x_i|^p$ can give the $L1$ and $L2$ regularization respectively.
Which of the following is correct about the selection about regularization?

◯ A: $L1$ is always better than $L2$.

◯ B: $L2$ is always better than $L1$.

◯ C: Either $L1$ or $L2$ has its own strength. Sometimes we can choose by trying them empirically.

◯ D: Whatever p value, $L_p(\mathbf{x})^p$ can always be a good regularization choice

2.7 You are training a CNN model that, after a number of convolutional and pooling layers, generates a volume of size $(H, W, C) = (8, 8, 1024)$. This is flattened and then fed through a 4096-neuron hidden layer and a final 200-neuron output layer (you are doing 200-class classification). You wish to represent the fully connected layers as convolutional layers that perform the same exact computations using square $F \times F$ filters. What would be the filter width (as given by parameter F) in the convolutional versions of the hidden and output FC layers respectively?

◯ A: 64, 200

◯ B: 64, 1

◯ C: 4096, 200

◯ D: 8, 1

◯ E: 8, 200

2.8 Suppose we have a 5-layer ConvNet with layer configurations as follows (here we skip the input/output dimensions since they are irrelevant to this question; no need to consider dilation either[1].):

(i) Conv 3×3, with stride 1

(ii) Conv 3×3, with stride 2

(iii) Conv 3×3, with stride 2

[1]Don't worry if you don't know what dilation is; it is irrelevant to this question.

(iv) Conv 3×3, with stride 1

(v) Pool 2×2, with stride 2

Which of the following is true about the (effective) receptive field of a neuron at each layer? (As a reminder, for a particular layer, the effective receptive field of a neuron in the layer output is the number of input pixels that the neuron is affected by.)

◯ A: The receptive field area of a neuron at Layer (ii) is 6×6.

◯ B: The receptive field area of a neuron at Layer (iii) is 4 times the area at layer (ii).

◯ C: The receptive field area of a neuron at Layer (iv) is 17×17.

◯ D: The receptive field area of a neuron at Layer (v) is 4 times the area at layer (iv).

◯ E: The receptive field area of a neuron at Layer (v) is 33×33.

◯ F: None of the above.

2.9 Select all that are true about pooling layers.

◯ A: For a pooling layer, the gradients w.r.t. some inputs will always be zeroed.

◯ B: A 2×2 pooling layer has 4 parameters.

◯ C: Pooling layers do not change the depth of the output image.

◯ D: Pooling layers do not change the height and width of the output image.

2.10 Select all statements that are true about recurrent neural networks.

◯ A: Training recurrent neural networks can be impeded by the exploding gradient problem.

◯ B: Unlike standard feedforward networks, recurrent neural networks can learn from sequences of variable length.

◯ C: Gradient clipping might help if your RNN is troubled by vanishing gradients.

◯ D: None of the above.

3 Short Answers (40 points)

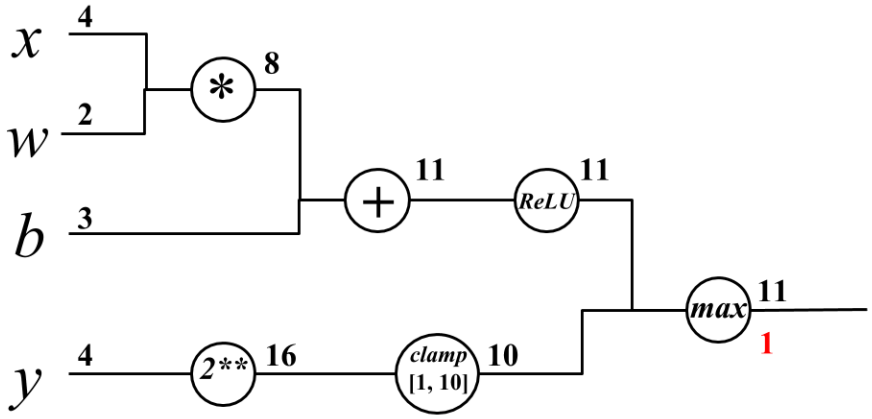
Please make sure to write your answer only in the provided space.

3.1 Computational Graphs (10 points)

1. (5 points) In the following computational graph, the activations for the forward pass have been filled out (numbers above the connections). Please fill out the gradient for each activation; the last gradient is provided for you (i.e. 1).

   - 2^{**}: exponential with base 2 (e.g. $2^{**3} = 8$)
   - $clamp_{[a,b]}$: clamping input values to be no smaller than a and no larger than b (e.g. $clamp_{[1,10]}(12) = 10$); i.e.

$$clamp_{[a,b]}(x) = \begin{cases} a & \text{if } x < a \\ x & \text{if } a \leq x \leq b \\ b & \text{if } x > b \end{cases}$$



2. (5 points) A feature map X is passed to a convolutional layer with 3×3 kernel K (i.e. operation $*$) with no padding followed by a 2×2 max pooling layer (i.e. operation max), resulting in a scalar output.

   Suppose the scalar output has a gradient of 1, what is the gradient at the position in X with value 1 (i.e. the underlined entry)?

$$\textbf{\textit{max}}\left(\begin{array}{|c|c|c|c|} \hline 4 & 6 & 3 & 4 \\ \hline \underline{\mathbf{1}} & 7 & 3 & 4 \\ \hline 5 & 8 & 4 & 4 \\ \hline 2 & 7 & 4 & 4 \\ \hline \end{array} \right)_{\boldsymbol{X}} * \begin{array}{|c|c|c|} \hline 0 & -1/4 & 0 \\ \hline -1/4 & 1 & -1/4 \\ \hline 0 & -1/4 & 0 \\ \hline \end{array}_{\boldsymbol{K}}$$

3.2 3D Convolution for Video Understanding (12 points)

We have previously seen 2D convolution networks being successfully used on image data. Video data are essentially temporal series of images, where each video as an input has dimensions (channels C, time T, height H, width W).

In this problem, we look at 3D convolutions for video data, where the convolution happens not only spatially but also temporally. 3D convolution operates on 4 dimensional input tensors of size $C \times T \times H \times W$ with 3D kernels, in contrast to the 2D case where we have input size $C \times H \times W$ and 2D kernels.

1. (6 points) Consider a 3D convolutional network with its layers specified in table 1. For each layer, fill in the table with the size of the activation volumes, and the number of weight and bias parameters. The INPUT layer is provided; this network takes in 8 frames of 64-by-64 pixel RGB images.
Please write your answer as a multiplication (e.g. 128 × 128 × 128) or points may deducted.
The layer descriptions are as follows:

- CONV3-N-p1-s2 is a convolutional layer with N $3 \times 3 \times 3$ kernels, padding 1 and stride 2 (in height, width, and temporal dimension) with ReLU activation. Similarly, CONV5-N-p2-s1 is a convolutional layer with N $5 \times 5 \times 5$ kernels, padding 2 and stride 1.
- POOL2 denotes a $2 \times 2 \times 2$ max-pooling layer, with stride 2 and no padding in all dimensions.
- FC-N denotes a fully-connected layer with N neurons, with ReLU activation.

Layer	Activation Volume Dimensions	# Weights	# Biases
INPUT	$3 \times 8 \times 64 \times 64$	N/A	N/A
CONV5-16-p2-s1			
POOL2			
CONV3-16-p1-s1			
POOL2			
CONV3-32-p1-s2			
FC-256			
FC-10			

Table 1: Dimensions and parameters.

2. (3 points) In a 3D convolutional network we can define *spatiotemporal batch normalization* similarly to operate on a batch of 4D feature maps of shape $N \times C \times T \times H \times W$, normalizing over the N, T, H, and W dimensions. If we now add one spatiotemporal Batch Normalization (BN) after each convolution layer, how many additional parameters will be introduced? Please put your answers in Table 2.

Layer	# BN parameters
CONV5-16-p2-s1	
CONV3-16-p1-s1	
CONV3-32-p1-s2	

Table 2: Number of BN parameters.

3. (3 points) 3D convolution network is computationally more expensive than its 2D counterpart. Assuming convolutions are implemented as matrix multiplications[2], let's analyze the computational complexity of convolutions in terms of the number of floating point multiplications they perform.

First calculate the complexity of 2D convolution, and then generalize to 3D convolution. Please write your answer as a multiplication (e.g. $128 \times 128 \times 128$) in Table 3.

Input tensor dimension	Convolutional layer type	# floating point multiplications
$3 \times 64 \times 64$	2D Conv, N=16, kernel=3×3, pad=1, stride=1	
$3 \times 64 \times 64$	2D Conv, N=32, kernel=5×5, pad=2, stride=1	
$3 \times 8 \times 64 \times 64$	3D Conv, N=32, kernel=$5 \times 5 \times 5$, pad=2, stride=1	

Table 3: Computation costs.

[2]Fun fact: there exist alternative ways of computing convolutions such as Fast Fourier Transform / Winograd convolutions, but don't consider them for this problem.

3.3 "Convolutions on Spatial Graphs" (18 points)

For many computer vision tasks such as image classification, it should not matter if an object is perfectly centered in an image or moved to the lower right corner. Convolutional neural networks achieve this so called translation-invariance with the help of 2D convolutions. Convolutions slide a filter W over a 2D input X to compute features independently of their location in the image. In order to extract features from a particular image patch $\widetilde{X}$ the dot product $\widetilde{X} \cdot W$ between $\widetilde{X}$ and the overlapping filter W is computed. Sometimes, however it might be needed to mask out selected pixels when applying a convolutional filter. One such case are convolutions over 2D spatial graphs which need to respect the node connectivity as depicted on the left side of Figure 1.

Spatial graph definition: In this exercise, we define 2D spatial graphs $G =< X, E >$ as follows. Each pixel on the image grid X defines a node x_{ij} in the graph. Edges E between pixel nodes define whether two nodes are connected and should influence each other or are unconnected and should not influence each other.

Spatial graph feature extraction: In order to extract features at node location x_{ij} from directly connected nodes, we center convolutional filter W over x_{ij} and compute the dot product, similarly to a 2D convolution. To prevent that unconnected nodes contribute to the feature extraction, we mask out nodes that are not connected to the center node x_{ij} with a binary mask M_{ij} (Figure 1, right). Specifically, we multiply filter W with mask M_{ij} element-wise before computing the dot product with $\widetilde{X}$ to extract features or in other words activations Y:

$$Y = f(X, W, M) = \widetilde{X} \cdot (W \odot M_{ij})$$

Each binary mask M_{ij} reflects the connectivity with respect to the center node and equals 1 if the overlapping pixel node is connected to the center node and 0 if it is not connected. Thus, while sliding the filter W across spatial graph locations ij, the binary masks M_{ij} change.

Now suppose we have the following 5×5 single-channel image X over which we define spatial graph $G =< X, E >$, a 3×3 filter W and 3×3 mask M_{ij}:

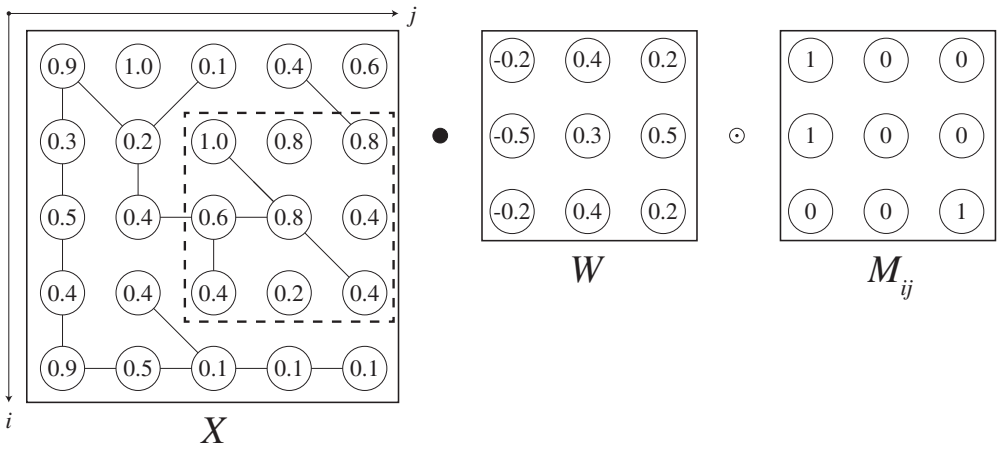


Figure 1: 5×5 single channel image X and 3×3 filter W and 3×3 mask M_{ij}.

1. (4 point) Calculate the activation at the dotted filter location by convolving filter W with image X (Ignore mask M_{ij} and the graph edges for now):

2. (4 point) Repeat the same calculation but this time only convolving over connected nodes in the graph with the help of M_{ij}:

3. (2 points) What would be the binary mask M_{ij} if we would move the dotted box one pixel to the left?

4. (1 point) What would a "1" at the center of M_{ij} indicate?

5. (2 points) Is the defined spatial graph feature extraction still translation-invariant in image-space X like a 2D convolution? Explain why or why not.

6. (2 points) The defined spatial graph feature extraction is translation-invariant within the space of spatial graphs, but still dependent on the relative position of connected nodes to the center node x_{ij}. Concretely, for example it matters whether a connected node is located to the right, left, bottom or top of a center node. How would you have to set the weights in filter W to remove this relative position dependency?

7. (3 points) The defined spatial graph feature extraction $Y = f(X, W, M)$ only takes directly connected nodes into account (e.g. in Figure 1, X: feature y_{11} is only influenced by nodes x_{21} and x_{22}). If we apply this operation again on the previously extracted features $Z = f(Y, W, M)$, we can compute over indirectly connected second-order nodes (e.g. in Figure 1, X: z_{11} is now also influenced by nodes x_{31}, x_{32}, and x_{13}). To compute over third-order connections, fourth-order connections and so on, we can keep stacking these operations until we have taken the entire graph, i.e. all nodes for each feature location into account.

   How many spatial graph feature extractors would we need to stack to take the entire graph in Figure 1 into account? What does the number of layers depend on and how could this be problematic? How could we potentially resolve this problem?

This page is left blank for scratch work only. DO NOT write your answers here.

This page is left blank for scratch work only. DO NOT write your answers here.