CSE 493 G1 / 599 G1
Deep Learning
Spring 2025 Exam

SOLUTIONS

May 20, 2025

Full Name: _____

UW Net ID: _____

Question		Score
Multiple Choice	(30 pts)	
Backpropagation	(22 pts)	
Convolution & Pooling	(16 pts)	
RNN Diagnostics	(12 pts)	
Tokenization	(20 pts)	
Total	(100 pts)	

Welcome to the CSE 493 G1 / 599 G1 Exam!

- The exam is 80 min and is **double-sided**.
- No electronic devices are allowed.

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

Mean: 79.7
Median: 81.3
Stdev: 10.8

1 Multiple Choices (30 points) - Recommended 20 Minutes

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 5 points and the answer may contain **exactly one or two** options. Selecting all of the correct options and none of the incorrect options will get full credit. For questions with multiple correct options, each incorrect or missing selection gets a 2.5-point deduction (up to 5 points).

1.1 (5 points) Given scores $z = [z_1, \ldots, z_d]$, let $\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$. Which statements are true?

- ◯ A: Adding a constant α to all z_i leaves the softmax output scores unchanged.
- ◯ B: The softmax function is non-differentiable at points where two components tie.
- ◯ C: Cross-entropy loss for multi-class classification is $-\sum_{i=1}^{N} \log \frac{e^{z_{y_i}}}{\sum_j e^{z_j}}$ (N is the size of training set).
- ◯ D: After applying softmax, the sum of the scores will exceed 1.
- ◯ E: None of the above.

SOLUTION:
A and C. (B) Softmax is smooth and differentiable everywhere, including at ties. (D) The sum of softmax scores should be exactly 1.

1.2 (5 points) When using batch gradient descent (full-batch) to minimize a differentiable loss function, selecting a learning rate that is too large will most likely result in:

- ◯ A: Very slow convergence toward the minimum.
- ◯ B: Divergence or oscillation around the minimum.
- ◯ C: Convergence to the global minimum.
- ◯ D: The gradient norm becoming exactly zero at each step.
- ◯ E: None of the above.

SOLUTION:
B. (A) A learning rate that is too large causes overshooting rather than taking small steps, so it does *not* produce very slow convergence. (C) A too-large learning rate typically prevents settling at the minimum; it does not guarantee convergence to the global minimum. (D) The gradient norm becoming exactly zero at each step would imply you've perfectly reached a stationary point, which a large learning rate does *not* enforce.

1.3 (5 points) Mark all valid statements:

- ◯ A: ReLU is defined as $f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$

◯ B: Leaky ReLU is defined as $f(x) = \max(0.01, x)$.

◯ C: Layer Norm uses batch statistics during inference.

◯ D: During inference, Batch Norm uses the running averages of mean and variance computed in training.

◯ E: None of the above.

SOLUTION:
A and D. (B) Leaky ReLU should be $f(x) = \max(0.01x, x)$. (C) Layer Norm uses per-sample statistics at both train and test time, not batch stats.

1.4 (5 points) Which of the following is a key reason to use convolutional layers instead of fully-connected layers when processing images?

◯ A: Convolutional layers require more parameters to learn complex patterns.

◯ B: Convolutional layers exploit spatial locality and share weights across locations.

◯ C: Convolutional layers enforce global connectivity between all pixels.

◯ D: Convolutional layers are non-differentiable.

◯ E: None of the above.

SOLUTION:
B. (C) Convolutional layers doesn't enforce connectivity between ALL pixels. (A) and (D) are not true obviously.

1.5 (5 points) Select all **incorrect** statements about RNNs and LSTMs:

◯ A: Standard RNNs suffer from vanishing gradients for long sequences.

◯ B: Gradient clipping can worsen exploding gradients in RNNs.

◯ C: LSTM forget-gate activations lie in $(0, 1)$ due to the sigmoid nonlinearity.

◯ D: Standard RNNs and LSTMs also need explicit positional encoding to help them "remember" position, like what Transformers do.

◯ E: None of the above.

SOLUTION:
(B) (D). All others are correct.

1.6 (5 points) In the "scaled dot-product" attention $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, which are **not true**?

- ◯ A: Scaling by $\sqrt{d_k}$ prevents softmax from becoming too peaked.
- ◯ B: Keys and queries must have the same dimensionality.
- ◯ C: Positional encoding cannot be learnable due to its inherent complexity.
- ◯ D: Multi-head attention concatenates parallel attention outputs.
- ◯ E: None of the above.

SOLUTION:
(C). All others are correct.

2 Backpropagation (22 points) - Recommended 20 Minutes

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

Consider the following 2-layer network with scalar input x, weights w_1, w_2, w_3, ReLU nonlinearity, and squared-error loss.

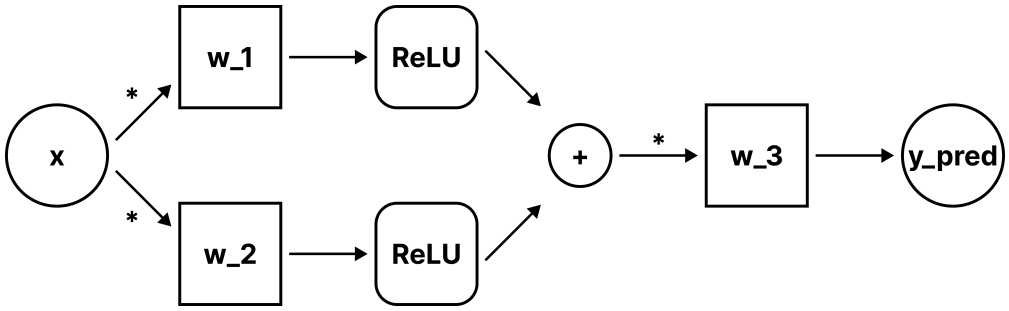

Figure 1: The model architectures of the 2-layer network.

Mathematically, the forward pass is defined as the following:

1. $z_1 = w_1 x$, $a_1 = \max(0, z_1)$
2. $z_2 = w_2 x$, $a_2 = \max(0, z_2)$
3. $y_{\text{pred}} = w_3 (a_1 + a_2)$
4. Loss $L = \frac{1}{2}(y_{\text{pred}} - y_{\text{true}})^2$

Given $x = 2$, $w_1 = 0.5$, $w_2 = -1.0$, $w_3 = 2.0$, and $y_{\text{true}} = 1.0$:

Given this setup, answer the following questions:

1. (16 points) Compute all forward-pass activations $(z_1, a_1, z_2, a_2, y_{\text{pred}})$ and then backpropagate to find gradients $\frac{\partial L}{\partial w_1}$, $\frac{\partial L}{\partial w_2}$, $\frac{\partial L}{\partial w_3}$. **Write your final answers on the labeled lines at the bottom.**

   Forward activations:

z_1: _____ a_1: _____ z_2: _____ a_2: _____ y_{pred}: _____

SOLUTION:

$$
\begin{aligned}
z_1 &= w_1 x = 0.5 \times 2 = 1.0, \\
a_1 &= \max(0, 1.0) = 1.0, \\
z_2 &= w_2 x = -1.0 \times 2 = -2.0, \\
a_2 &= \max(0, -2.0) = 0.0, \\
y_{\text{pred}} &= w_3 \, (a_1 + a_2) = 2.0 \cdot (1.0 + 0.0) = 2.0, \\
L &= \tfrac{1}{2}(2.0 - 1.0)^2 = \tfrac{1}{2}(1.0)^2 = 0.5.
\end{aligned}
$$

Backward pass:

$\frac{\partial L}{\partial w_1}$: _____ $\frac{\partial L}{\partial w_2}$: _____ $\frac{\partial L}{\partial w_3}$: _____

SOLUTION:

$$\frac{\partial L}{\partial y_{\text{pred}}} = y_{\text{pred}} - y_{\text{true}} = 2.0 - 1.0 = 1.0.$$

For w_3:

$$\frac{\partial y_{\text{pred}}}{\partial w_3} = a_1 + a_2 = 1.0, \frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial y_{\text{pred}}} \cdot \frac{\partial y_{\text{pred}}}{\partial w_3} = 1.0 \times 1.0 = 1.0.$$

For a_1:

$$\frac{\partial y_{\text{pred}}}{\partial a_1} = w_3 = 2.0, \frac{\partial L}{\partial a_1} = 1.0 \times 2.0 = 2.0.$$

Since $a_1 = \text{ReLU}(z_1)$ and $z_1 > 0$, $\frac{da_1}{dz_1} = 1$, so

$$\frac{\partial L}{\partial z_1} = 2.0 \times 1 = 2.0.$$

Then $z_1 = w_1 x$ gives

$$\frac{\partial z_1}{\partial w_1} = x = 2.0, \quad \frac{\partial L}{\partial w_1} = 2.0 \times 2.0 = 4.0.$$

For w_2:

$$a_2 = 0 \rightarrow \frac{da_2}{dz_2} = 0 \rightarrow \frac{\partial L}{\partial w_2} = 0.$$

Gradients:

$$\frac{\partial L}{\partial w_1} = 4.0, \quad \frac{\partial L}{\partial w_2} = 0.0, \quad \frac{\partial L}{\partial w_3} = 1.0.$$

2. (6 points) Using learning rate $\alpha = 0.1$, perform one SGD update on (w_1, w_2, w_3) and report the new weight values. **Remember to fill your final answers on the labeled lines at the bottom.**

w_1: _____ w_2: _____ w_3: _____

SOLUTION:

$$w_1^{\text{new}} = 0.5 - 0.1 \times 4.0 = 0.5 - 0.4 = 0.1,$$
$$w_2^{\text{new}} = -1.0 - 0.1 \times 0.0 = -1.0,$$
$$w_3^{\text{new}} = 2.0 - 0.1 \times 1.0 = 2.0 - 0.1 = 1.9.$$

3 Convolution & Pooling (16 points) - Recommended 15 Minutes

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

Let

$$I = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 2 \\ 0 & 1 & 2 & 1 \end{bmatrix}, \quad F = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}.$$

Given this setup, answer the following questions:

1. (4 points) What is the shape of the feature-map produced by convolving I with F using **no padding** and **stride = 1**?

   SOLUTION:
   General formula for a $H \times W$ input, filter $k \times k$, padding p, stride s:

   $$H_{\text{out}} = \left\lfloor \frac{H + 2p - k}{s} \right\rfloor + 1, \quad W_{\text{out}} = \left\lfloor \frac{W + 2p - k}{s} \right\rfloor + 1.$$

   Here $H = W = 4$, $k = 2$, $p = 0$, $s = 1$:

   $$H_{\text{out}} = W_{\text{out}} = \left\lfloor \frac{4 + 0 - 2}{1} \right\rfloor + 1 = \lfloor 2 \rfloor + 1 = 3.$$

   The feature-map is 3×3.

2. (4 points) Explain how setting **padding = 1** and **stride = 2** would change the output dimensions. Why might a model designer choose those settings?

   SOLUTION:

- Output dimensions:
$$H_{\text{out}} = \left\lfloor \frac{4 + 2 \cdot 1 - 2}{2} \right\rfloor + 1 = \lfloor 2 \rfloor + 1 = 3,$$
so the output is 3×3 instead of 3×3 with no padding/stride1.
- Why choose these settings? Padding preserves border information, and stride=2 reduces spatial dimensions (downsampling), lowering computation and memory while still covering the full input.

3. (4 points) Apply a 2×2 max-pool with stride $= 2$, no padding, to the original I. What is the resulting 2×2 matrix?

SOLUTION:
Partition I into $\left\{ \begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix} \right\}$, $\left\{ \begin{smallmatrix} 2 & 1 \\ 0 & 0 \end{smallmatrix} \right\}$, $\left\{ \begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix} \right\}$, $\left\{ \begin{smallmatrix} 1 & 2 \\ 2 & 1 \end{smallmatrix} \right\}$. Taking maxima:

$$\begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}.$$

4. (4 points) In practice, one can downsample a feature-map either via **stride $>$ 1** in a convolution or via a separate pooling operation. Give one advantage and one disadvantage of each approach.

SOLUTION:

Strided convolution

Pro: Combines feature extraction and downsampling in one learnable step (fewer layers, less compute).

Con: Harder to separate "what" vs. "where". The model must learn both simultaneously, which can introduce aliasing.

Pooling

Pro: Non-learnable pooling (e.g. max) enforces local translation invariance and highlights strongest activations.

Con: May discard useful information and adds an extra operation (more layers/overhead).

4 RNN Diagnostics (12 points) - Recommended 10 Minutes

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

1. (6 points) During training, you observe that the training loss steadily decreases, but the validation accuracy on sequences longer than 50 time steps collapses to random chance. List two potential reasons related to RNN behavior and outline specific debugging experiments or changes you would perform to identify and mitigate each cause.

   SOLUTION:
   Reason 1: Vanishing gradients on long sequences.

   Debug: Track gradient magnitudes across sequence positions; plot gradient norms for each time step to confirm exponential decay.

   Mitigation: Replace the vanilla RNN with an LSTM or GRU cell to introduce gating and preserve gradients over long horizons.

   Reason 2: Overfitting to short sequences or insufficient capacity.

   Debug: Evaluate performance on held-out short vs. long sequences separately; monitor training vs. validation gap.

   Mitigation: Increase hidden-state dimensionality or add regularization (dropout on recurrent connections, weight decay); implement curriculum learning by gradually increasing sequence length during training.

2. (6 points) You log the gradient norms of the hidden-state parameters at each time step and find that gradients for early time steps are nearly zero, while those for recent steps are large and unstable. Explain what this indicates about your model's training dynamics, and propose two modifications (architectural or optimization) to alleviate the problem, explaining how each modification addresses the issue.

   SOLUTION:

The pattern (near-zero gradients at early steps, large at recent steps) indicates *vanishing gradients* for early time steps and possible gradient concentration at the end.

Modification 1: Apply *gradient clipping* (e.g., clip by global norm) to prevent extreme gradients and help redistribute gradient flow more evenly across time steps.

Modification 2: Introduce *residual connections* or *layer normalization* in the recurrent update to stabilize training and reduce vanishing/exploding behaviors.

5 Tokenization (20 points) - Recommended 15 Minutes

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

Consider a greedy subword tokenizer processing the sentence:

```
"The dog that chased the dog was tired."
```

Then, after tokenization, a standard Transformer encoder will receive them as input.

Assume we first run self-attention *without* any positional encodings.

Moreover, assume the sentence is tokenized with a greedy longest-match (maximal-munch) algorithm over a fixed subword vocabulary. At each step, among all vocabulary entries that match the current prefix of the remaining string, choose the longest one (break ties arbitrarily or by smaller ID), emit its ID, remove it from the front, and repeat until the string is consumed.

Example: Suppose the vocabulary contains the tokens {`"he"`:1, `"hello"`:2, `"llo"`:3}. Then the input `"hello"` is tokenized as [2] ("hello") rather than [1,3] ("he","llo"), because the tokenizer greedily matches the longest possible subword.

Given this setup, answer the following questions:

1. (4 points) Suppose we use a greedy subword tokenizer (BPE) with the following dictionary (token → index):

$$\{\texttt{" "} : 1,\ \texttt{"The "} : 2,\ \texttt{"dog "} : 3,\ \texttt{"dog"} : 4,\ \texttt{"that"} : 5, ,\ \texttt{"chased"} : 6, \texttt{"the"} : 7,$$

$$\texttt{"dog."} : 8,\ \texttt{"was"} : 9, \texttt{"tired"} : 10,\ \texttt{"."} : 11\}.$$

   Describe step by step how this sentence will be tokenized using the longest-match rule, and list the resulting token indices.

   SOLUTION:

   (a) "The " → 2. Remaining: "dog that chased the dog was tired."
   (b) "dog " → 3. Remaining: "that chased the dog was tired."
   (c) "that" → 5. Remaining: " chased the dog was tired."
   (d) " " → 1. Remaining: "chased the dog was tired."
   (e) "chased" → 6. Remaining: " the dog was tired."
   (f) " " → 1. Remaining: "the dog was tired."
   (g) "the" → 7. Remaining: " dog was tired."
   (h) " " → 1. Remaining: "dog was tired."
   (i) "dog " → 3. Remaining: "was tired."
   (j) "was" → 9. Remaining: " tired."
   (k) " " → 1. Remaining: "tired."
   (l) "tired" → 10. Remaining: "."
   (m) "." → 11. Remaining: ""

Resulting token indices: $[2, 3, 5, 1, 6, 1, 7, 1, 3, 9, 1, 10, 11]$.

2. (4 points) Given the tokenization you came up with from the last question as input, how will the model differentiate between the two occurrences of the token "dog" when no positional encodings are used? Provide a brief conceptual explanation.

SOLUTION:
Without positional encodings, each "dog" token has the same embedding, so its query, key, and value vectors are identical. Self-attention uses only these vectors, so both positions compute the same attention weights and produce the same output.

3. (4 points) Suppose we use a greedy subword tokenizer (BPE) with a new dictionary (token → index), note that only index 4 is changed from "dog" to " dog" compared to the previous dictionary:

$$\{\texttt{" "} : 1,\ \texttt{"The "} : 2,\ \texttt{"dog "} : 3,\ \texttt{" dog"} : 4,\ \texttt{"that"} : 5, ,\ \texttt{"chased"} : 6, \texttt{"the"} : 7,$$

$$\texttt{"dog."} : 8,\ \texttt{"was"} : 9, \texttt{"tired"} : 10,\ \texttt{"."} : 11\}.$$

Describe step by step how this sentence will be tokenized using the longest-match rule, and list the resulting token indices.

SOLUTION:

(a) "The " → 2. Remaining: "dog that chased the dog was tired."
(b) "dog " → 3. Remaining: "that chased the dog was tired."
(c) "that" → 5. Remaining: " chased the dog was tired."
(d) " " → 1. Remaining: "chased the dog was tired."
(e) "chased" → 6. Remaining: " the dog was tired."
(f) " " → 1. Remaining: "the dog was tired."

(g) “the” → 7. Remaining: “ dog was tired.”

(h) “ dog ” → 4. Remaining: “was tired.”

(i) “was” → 9. Remaining: “ tired.”

(j) “ ” → 1. Remaining: “tired.”

(k) “tired” → 10. Remaining: “.”

(l) “.” → 11. Remaining: “”

Resulting token indices: $[2, 3, 5, 1, 6, 1, 7, 4, 9, 1, 10, 11]$.

4. (4 points) Given the tokenization you came up with from the last question as input, how will the model differentiate between the two occurrences of the token “dog” when no positional encodings are used? Provide a brief conceptual explanation.

SOLUTION:
Although without positional encodings, the two “dog” tokens are encoded into different embeddings (“dog ” and “ dog”), so their query, key, and value vectors are different. Self-attention will then produce different outputs, which means that the model is able to differentiate the two.

5. (4 points) Explain why positional encodings remain crucial for capturing sequence order and structural relationships in a Transformer, even when token-level embeddings differ between occurrences.

SOLUTION:
Transformers are inherently permutation-invariant: without positional encodings, they cannot infer

token order or relative distances. Positional encodings inject absolute (and via sinusoidals, relative) position information into embeddings, enabling the model to track sequence structure, learn n-gram patterns, and capture long-range dependencies that depend on token positions.