Due Date: March 28th, 23:00 2025

Instructions

- For all questions that are not graded only on the answer, show your work! Any problem without work shown will get no marks regardless of the correctness of the final answer.
- Please try to use a document preparation system such as LaTeX. If you write your answers by hand, note that you risk losing marks if your writing is illegible without any possibility of regrade, at the discretion of the grader.
- Submit your answers electronically via the course GradeScope. Incorrectly assigned answers can be given 0 automatically at the discretion of the grader. To assign answers properly, please:
  - Make sure that the top of the first assigned page is the question being graded.
  - Do not include any part of answer to any other questions within the assigned pages.
  - Assigned pages need to be placed in order.
  - For questions with multiple parts, the answers should be written in order of the parts within the question.
- Questions requiring written responses should be short and concise when necessary.Unnecessary wordiness will be penalized at the grader's discretion.
- Please sign the agreement below.
- It is your responsibility to follow updates to the assignment after release. All changes will be visible on Overleaf and Piazza.
- Any questions should be directed towards the TAs for this assignment (theoretical part): *Alireza Dizaji, Pascal Tikeng.*

I acknowledge I have read the above instructions and will abide by them throughout this assignment. I further acknowledge that any assignment submitted without the following form completed will result in no marks being given for this portion of the assignment.

Signature: _____

Name: _____

UdeM Student ID: _____

Question 1 (6pts)**.** (**Normalization**)

Batch normalization, layer normalization and instance normalization all involve calculating the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma^2}$ with respect to different subsets of the tensor dimensions. Given the following 3D tensor, calculate the corresponding mean and variance tensors for each normalization technique: $\boldsymbol{\mu}_{batch}$, $\boldsymbol{\mu}_{layer}$, $\boldsymbol{\mu}_{instance}$, $\boldsymbol{\sigma}^2_{batch}$, $\boldsymbol{\sigma}^2_{layer}$, and $\boldsymbol{\sigma}^2_{instance}$.

$$\left[\begin{bmatrix}2,5,1\\4,3,4\end{bmatrix},\begin{bmatrix}1,2,4\\1,2,2\end{bmatrix},\begin{bmatrix}3,1,2\\3,2,4\end{bmatrix},\begin{bmatrix}3,4,3\\5,2,3\end{bmatrix},\right]$$

The size of this tensor is 4 x 2 x 3, which corresponds to the batch size, number of input channels, and number of features, respectively.

Question 2 (20pts)**.** (**Decoding**)

Suppose that we have a vocabulary containing N possible words, including a special token `<BOS>` to indicate the beginning of a sentence. Recall that in general, a language model with a full context can be written as

$$p(w_1, w_2, \ldots, w_T \mid w_0) = \prod_{t=1}^{T} p(w_t \mid w_0, \ldots, w_{t-1}).$$

We will use the notation $\boldsymbol{w}_{0:t-1}$ to denote the (partial) sequence $(w_0, \ldots, w_{t-1})$. Once we have a fully trained language model, we would like to generate realistic sequences of words from our language model, starting with our special token `<BOS>`. In particular, we might be interested in generating the most likely sequence $\boldsymbol{w}^\star_{1:T}$ under this model, defined as

$$\boldsymbol{w}^\star_{1:T} = \arg\max_{\boldsymbol{w}_{1:T}} p(\boldsymbol{w}_{1:T} \mid w_0 = \texttt{<BOS>}). \tag{1}$$

For clarity we will drop the explicit conditioning on w_0, assuming from now on that the sequences always start with the `<BOS>` token.

1. (2 pts) How many possible sequences of length $T + 1$ starting with the token `<BOS>` can be generated in total? Give an exact expression, without the O notation. Note that the length $T+1$ here includes the `<BOS>` token.

2. (2 pts) To determine the most likely sequence $\boldsymbol{w}^\star_{1:T}$, one could perform exhaustive enumeration of all possible sequences and select the one with the highest joint probability (as defined in equation 1). Comment on the feasibility of this approach. Is it scalable with vocabulary size?

3. (3 pts) In light of the difficulties associated with exhaustive enumeration, it becomes essential to employ practical search strategies to obtain a reasonable approximation of the most likely sequence.

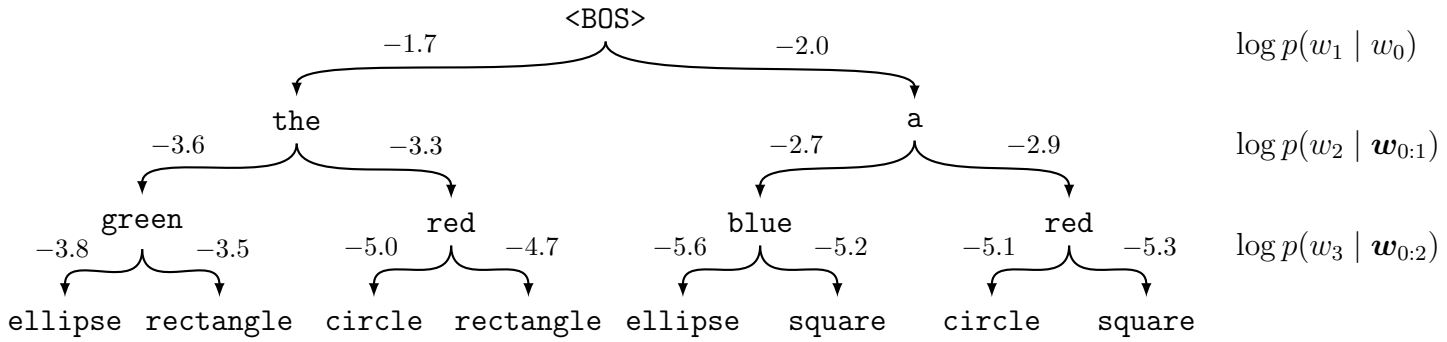
   In order to generate B sequences having high likelihood, one can use a heuristic algorithm called Beam search decoding, whose pseudo-code is given in Algorithm 1 below

Algorithm 1: Beam search decoding

Input: A language model $p(\boldsymbol{w}_{1:T} \mid w_0)$, the beam width B
Output: B sequences $\boldsymbol{w}_{1:T}^{(b)}$ for $b \in \{1, \dots, B\}$
Initialization: $w_0^{(b)} \leftarrow$ <BOS> for all $b \in \{1, \dots, B\}$
Initial log-likelihoods: $l_0^{(b)} \leftarrow 0$ for all $b \in \{1, \dots, B\}$
for $t = 1$ **to** T **do**
  for $b = 1$ **to** B **do**
    for $j = 1$ **to** N **do**
      $s_b(j) \leftarrow l_{t-1}^{(b)} + \log p(w_t = j \mid \boldsymbol{w}_{0:t-1}^{(b)})$
  for $b = 1$ **to** B **do**
    Find (b', j) such that $s_{b'}(j)$ is the b-th largest score
    Save the partial sequence b': $\widetilde{\boldsymbol{w}}_{0:t-1}^{(b)} \leftarrow \boldsymbol{w}_{0:t-1}^{(b')}$
    Add the word j to the sequence b: $w_t^{(b)} \leftarrow j$
    Update the log-likelihood: $l_t^{(b)} \leftarrow s_{b'}(j)$
  Assign the partial sequences: $\boldsymbol{w}_{0:t-1}^{(b)} \leftarrow \widetilde{\boldsymbol{w}}_{0:t-1}^{(b)}$ for all $b \in \{1, \dots, B\}$

What is the time complexity of Algorithm 1? Its space complexity? Write the algorithmic complexities using the O notation, as a function of T, B, and N. Is this a practical decoding algorithm when the size of the vocabulary is large?

4. (10 pts) The different sequences that can be generated with a language model can be represented as a tree, where the nodes correspond to words and edges are labeled with the log-probability $\log p(w_t \mid \boldsymbol{w}_{0:t-1})$, depending on the path $\boldsymbol{w}_{0:t-1}$. In this question, consider the following language model (where the low probability paths have been removed for clarity)



   (a) (2 pts) Greedy decoding is a simple algorithm where the next word $\overline{w}_t$ is selected by maximizing the conditional probability $p(w_t \mid \overline{\boldsymbol{w}}_{0:t-1})$ (with $\overline{w}_0 =$ <BOS>)

$$\overline{w}_t = \arg\max_{w_t} \log p(w_t \mid \overline{\boldsymbol{w}}_{0:t-1}).$$

   Find $\overline{\boldsymbol{w}}_{1:3}$ using greedy decoding on this language model, and its log-likelihood $\log p(\overline{\boldsymbol{w}}_{1:3})$.

   (b) (8 pts) Apply beam search decoding with a beam width $B = 2$ to this language model, and find $\boldsymbol{w}_{1:3}^{(1)}$ and $\boldsymbol{w}_{1:3}^{(2)}$, together with their respective log-likelihoods.

5. (3 pts) Please highlight the primary limitation that stands out to you for each of the discussed methods (greedy decoding and beam search).

Question 3 (17pts)**.** (**RNNs**)

Consider the following Bidirectional RNN:

$$\boldsymbol{h}_t^{(f)} = \tanh(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1}^{(f)})$$
$$\boldsymbol{h}_t^{(b)} = \tanh(\boldsymbol{W}^{(b)}\boldsymbol{x}_t + \boldsymbol{U}^{(b)}\boldsymbol{h}_{t+1}^{(b)})$$
$$\boldsymbol{y}_t = \boldsymbol{V}^{(f)}\boldsymbol{h}_t^{(f)} + \boldsymbol{V}^{(b)}\boldsymbol{h}_t^{(b)}$$

where $\boldsymbol{W}^{(f)}, \boldsymbol{W}^{(b)}, \boldsymbol{U}^{(f)}, \boldsymbol{U}^{(b)}, \boldsymbol{V}^{(f)}, \boldsymbol{V}^{(b)} \in \mathbb{R}^{d\times d}$, and $\boldsymbol{x}_i, \boldsymbol{h}_i^{(f)}, \boldsymbol{h}_i^{(b)} \in \mathbb{R}^d, \forall i \in [T]$ where the superscripts f and b correspond to the forward and backward RNNs respectively and tanh denotes the hyperbolic tangent function. Let $\boldsymbol{z}_t$ be the true target of the prediction $\boldsymbol{y}_t$ and consider the sum of squared loss $L = \sum_t L_t$ where $L_t = ||\boldsymbol{z}_t - \boldsymbol{y}_t||_2^2$.

In this question, our goal is to obtain an expression for the gradients $\nabla_{\boldsymbol{W}^{(b)}} L$ and $\nabla_{\boldsymbol{U}^{(f)}} L$.

1. (2 pts) First, complete the following computational graph for this RNN, unrolled for 3 time steps (from $t = 1$ to $t = 3$). Label each node with the corresponding hidden unit and each edge with the corresponding weight. Note that it includes the initial hidden states for both the forward and backward RNNs.

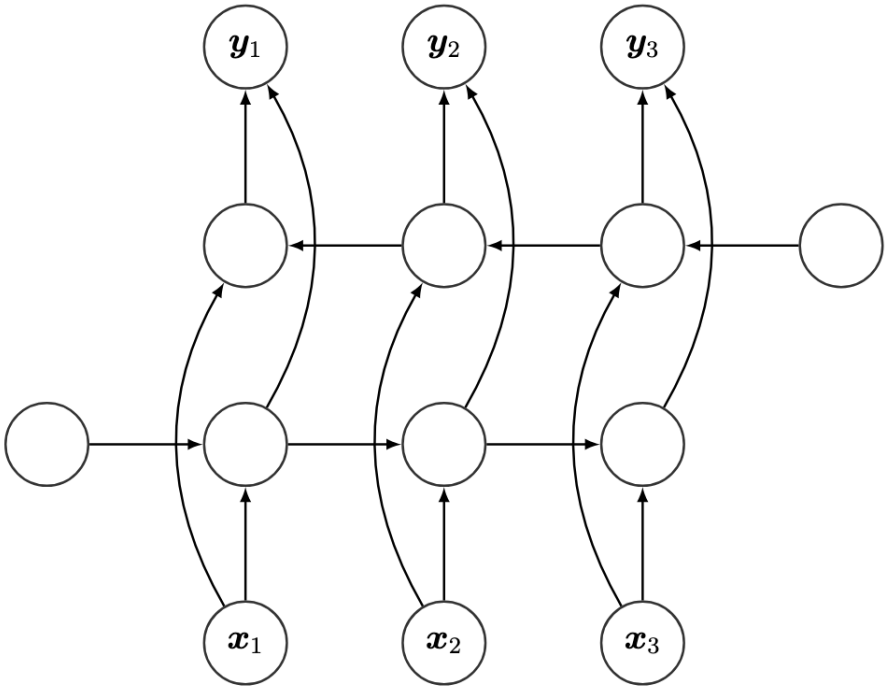


FIGURE 1 – Computational graph of the bidirectional RNN unrolled for three timesteps.

2. (10 pts) Using total derivatives we can express the gradients $\nabla_{\boldsymbol{h}_t^{(f)}} L$ and $\nabla_{\boldsymbol{h}_t^{(b)}} L$ recursively in terms of $\nabla_{\boldsymbol{h}_{t+1}^{(f)}} L$ and $\nabla_{\boldsymbol{h}_{t-1}^{(b)}} L$ as follows:

$$\nabla_{\boldsymbol{h}_t^{(f)}} L = \nabla_{\boldsymbol{h}_t^{(f)}} L_t + \left(\frac{\partial \boldsymbol{h}_{t+1}^{(f)}}{\partial \boldsymbol{h}_t^{(f)}}\right)^\top \nabla_{\boldsymbol{h}_{t+1}^{(f)}} L$$

$$\nabla_{\boldsymbol{h}_t^{(b)}} L = \nabla_{\boldsymbol{h}_t^{(b)}} L_t + \left(\frac{\partial \boldsymbol{h}_{t-1}^{(b)}}{\partial \boldsymbol{h}_t^{(b)}}\right)^\top \nabla_{\boldsymbol{h}_{t-1}^{(b)}} L$$

(a) (8 pts) Derive the expression for $\nabla_{\boldsymbol{h}_t^{(f)}} L_t$, $\nabla_{\boldsymbol{h}_t^{(b)}} L_t$, $\left(\frac{\partial \boldsymbol{h}_{t+1}^{(f)}}{\partial \boldsymbol{h}_t^{(f)}}\right)_{ij}$ and $\left(\frac{\partial \boldsymbol{h}_{t-1}^{(b)}}{\partial \boldsymbol{h}_t^{(b)}}\right)_{ij}$.

(b) (2 pts) Now using prev section, drive the expression for Jacobian matrices $\frac{\partial \boldsymbol{h}_{t+1}^{(f)}}{\partial \boldsymbol{h}_t^{(f)}}$ and $\frac{\partial \boldsymbol{h}_{t-1}^{(b)}}{\partial \boldsymbol{h}_t^{(b)}}$.

3. (5 pts) Now derive $\nabla_{\boldsymbol{W}^{(b)}} L$ and $\nabla_{\boldsymbol{U}^{(f)}} L$ as functions of $\nabla_{\boldsymbol{h}_t^{(b)}} L$ and $\nabla_{\boldsymbol{h}_t^{(f)}} L$, respectively.
Hint: It might be useful to consider the contribution of the weight matrices when computing the recurrent hidden unit at a particular time t and how those contributions might be aggregated.

Question 4 (18pts)**.** (**Low-rank adaptation**)

In this question, the goal is to study the low-rank adaptation techniques. Low-rank adaptation techniques are parameter-efficient fine-tuning methods that adapt large pre-trained models to specific tasks by adding small, trainable low-rank matrices to the frozen base weights, which is a common practice to reduce the memory usage with slight to zero changes on the performance. Here, you

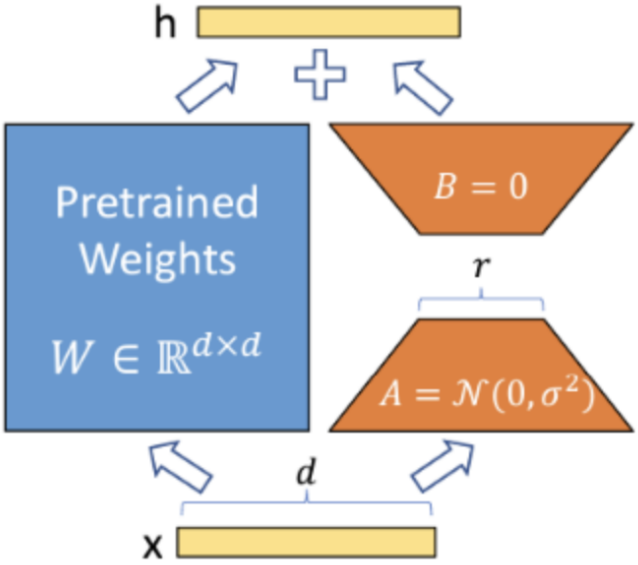


Figure 1: LoRA architecture

will study two well-known low-rank adaptation techinques, LoRA and QLoRA; You can access their papers by clicking here and here. Provide your answers for the following questions:

1. (2 pts) How do QLoRA and LoRA differ in their implementation of low-rank updates?
2. (2 pts) How does the choice of rank affect the performance and memory usage in both QLoRA and LoRA?
3. (2 pts) What are the trade-offs in speed and latency between QLoRA and LoRA during inference?

4. (12 pts) In this section, you will examine the differences in time and space complexities when applying LoRA and QLoRA for fine-tuning a Transformer architecture that includes Linear Feed-Forward network (LFFN) and multi-head attention (MHA) layers:

$$
\begin{aligned}
&\boldsymbol{H}^l = \text{LFFN}^{(l)}(\text{MHA}^{(l)}(\boldsymbol{H}^{l-1})) \\
&\text{LFFN}^{(l)}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W}^{F,l} + \boldsymbol{X}\boldsymbol{A}^{F,l}(\boldsymbol{B}^{F,l})^T \\
&\text{MHA}^{(l)}(\boldsymbol{H}) = [\text{head}^{l,1}(\boldsymbol{H}), \text{head}^{l,2}(\boldsymbol{H}), ..., \text{head}^{l,h}(\boldsymbol{H})]\boldsymbol{W}^{O,l} \\
&\text{head}^{l,i}(\boldsymbol{H}) = softmax(\frac{\boldsymbol{Q}^{l,i}(\boldsymbol{K}^{l,i})^T}{\sqrt{d}})\boldsymbol{V}^{l,i} \\
&\boldsymbol{Q}^{l,i} = \boldsymbol{H}\boldsymbol{W}^{Q,l,i} + \boldsymbol{H}\boldsymbol{A}^{Q,l,i}(\boldsymbol{B}^{Q,l,i})^T \\
&\boldsymbol{K}^{l,i} = \boldsymbol{H}\boldsymbol{W}^{K,l,i} + \boldsymbol{H}\boldsymbol{A}^{K,l,i}(\boldsymbol{B}^{K,l,i})^T \\
&\boldsymbol{V}^{l,i} = \boldsymbol{H}\boldsymbol{W}^{V,l,i} + \boldsymbol{H}\boldsymbol{A}^{V,l,i}(\boldsymbol{B}^{V,l,i})^T
\end{aligned}
$$

All pretrained parameters ($\boldsymbol{W}^*$) are frozen, where $\boldsymbol{W}^{Q,*}, \boldsymbol{W}^{K,*}, \boldsymbol{W}^{V,*}, \boldsymbol{W}^{F,*} \in \mathbb{R}^{d\times d}$, while $W^{O,*} \in \mathbb{R}^{dh\times d}$, and $\boldsymbol{A}^*, \boldsymbol{B}^* \in \mathbb{R}^{d\times r}$. By default, the precision used is 32-bit floating point (FP), while QLoRA first quantizes the pretrained parameters to m-bit integer precision for fine-tuning, where the quantization of a single matrix $\boldsymbol{W}$ is formulated as following:

$$
\tilde{\boldsymbol{W}} = \text{quant}(\boldsymbol{W}) = \left\lfloor \frac{\boldsymbol{W}}{\Delta} \right\rfloor, \Delta = \frac{max(|\boldsymbol{W}|)}{2^{(m-1)} - 1}
$$

and the dequantization of the matrix $\tilde{\boldsymbol{W}}$ to the original precision is computed by:

$$
\boldsymbol{W} = \text{dequant}(\Delta, \tilde{\boldsymbol{W}}) = \Delta\tilde{\boldsymbol{W}}
$$

Given the embedding dimension d, low-rank dimension $r \ll d$, h heads, and the context length n, in a single forward pass:

(a) (9 pts) Compute the time and space complexities of applying QLoRA and LoRA to FFN and MHA separately. In this problem, space complexity includes only the trainable parameters. *Hint: Ensure that operands at each stage are in the same precision.*

(b) (3 pts) If you were assigned to fine-tune using low-rank adapters and had the choice of applying either LoRA or QLoRA to each of these two modules, under the following conditions, which would you select and why? Fill in the table below based on your answer.

	MHA	LFFN
Memory limit		
Faster inference		
Both		