HOMEWORK 7: DEEP LEARNING

10-301/10-601 Introduction to Machine Learning (Fall 2023)
http://www.cs.cmu.edu/~mgormley/courses/10601/
OUT: Sunday, November 12th
DUE: Monday, November 20th
TAs: Haohui, Rakshith, Sahithya, Siva, Monica, Bhargav, Neural the Narwhal

Summary In this assignment you will implement the attention layer of the transformer and perform pre-training, fine-tuning, and performance evaluation. You will begin by going through some conceptual questions about CNNs, RNNs, transformers, pre-training, and fine-tuning that help build your intuition for the deep learning models and then use that intuition to build your own models.

START HERE: Instructions

- **Collaboration Policy**: Please read the collaboration policy here: http://www.cs.cmu.edu/~mgormley/courses/10601/syllabus.html
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/10601/syllabus.html
- **Submitting your work:** You will use Gradescope to submit answers to all questions and code. Please follow instructions at the end of this PDF to correctly submit all your code to Gradescope.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Each derivation/proof should be completed in the boxes provided. You are responsible for ensuring that your submission contains exactly the same number of pages and the same alignment as our PDF template. If you do not follow the template, your assignment may not be graded correctly by our AI assisted grader and there will be a **2% penalty** (e.g., if the homework is out of 100 points, 2 points will be deducted from your final score).
  - **Programming:** You will submit your code for programming questions on the homework to Gradescope. After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). You are only permitted to use the Python Standard Library modules, `numpy` and the modules already imported in the starter notebook. You are not permitted to import any other modules. You will not have to change the default version of your programming environment and the versions of the permitted libraries on Google Colab. You have 10 free Gradescope programming submissions, after which you will begin to lose points from your total programming score. We recommend debugging your implementation on Google Colab and making sure your code is running correctly first before submitting your code to Gradescope.

- **Materials:** The data and reference output that you will need in order to complete this assignment is posted along with the writeup and template on the course website.

Instructions for Specific Problem Types

For “Select One” questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

- ● Matt Gormley
- ○ Marie Curie
- ○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

- ● Henry Chai
- ○ Marie Curie
- ~~●~~ Noam Chomsky

For “Select all that apply” questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- □ I don’t know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- ~~■~~ I don’t know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

$\boxed{\text{10-601}}$ $\boxed{\text{10-}\cancel{6}\text{301}}$

Written Questions (51 points)

1 LaTeX Bonus Point and Template Alignment (1 points)

1. (1 point) **Select one:** Did you use LaTeX for the entire written portion of this homework?

   ◯ Yes

   ◯ No

2. (0 points) **Select one:** I have ensured that my final submission is aligned with the original template given to me in the handout file and that I haven't deleted or resized any items or made any other modifications which will result in a misaligned template. I understand that incorrectly responding yes to this question will result in a penalty equivalent to 2% of the points on this assignment.
   Note: Failing to answer this question will not exempt you from the 2% misalignment penalty.

   ◯ Yes

2 Convolutional Neural Network (14 points)

1. In this problem, consider a convolutional layer from a standard implementation of a CNN as described in lecture, without any bias term.

$$X = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 0 & -2 & 3 & 4 & 1 \\ \hline 2 & 9 & 5 & 6 & 0 & -1 \\ \hline 0 & -3 & 1 & 3 & 4 & 4 \\ \hline 6 & 5 & 2 & 0 & 6 & 8 \\ \hline -5 & 4 & -3 & 1 & 3 & -2 \\ \hline 4 & 1 & 2 & 8 & 9 & 7 \\ \hline \end{array} \qquad F = \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \qquad Y = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline e & f & g & h \\ \hline i & j & k & l \\ \hline m & n & o & p \\ \hline \end{array}$$

   (a) (1 point) Let an image X (6×6) be convolved with a filter F (3×3) using no padding and a stride of 1 to produce an output Y (4×4). What is value of j in the output Y?

   Your Answer

   (b) (1 point) Suppose you instead had an input feature map (or image) of size 6×4 (height $\times$ width) and a filter of size 2×2, using no padding and a stride of 2, what would be the resulting output size? Write your answer in the format: height $\times$ width.

   Your Answer

2. Parameter sharing is a very important concept for CNN because it drastically reduces the complexity of the learning problem and consequently that of the model required to tackle it. The following questions will deal with parameter sharing. Assume that there is no bias term in our convolutional layer.

   (a) (1 point) **Select all that apply:** Which of the following are parameters of a convolutional layer?

   ☐ Stride size

   ☐ Padding size

   ☐ Image size

   ☐ Filter size

   ☐ Weights in the filter

   ☐ None of the above

   (b) (1 point) **Select all that apply:** Which of the following are hyperparameters of a convolutional layer?

   ☐ Stride size

   ☐ Padding size

   ☐ Image size

   ☐ Filter size

   ☐ Weights in the filter

   ☐ None of the above

   (c) (1 point) Suppose for the convolutional layer, we are given grayscale images of size 22×22. Using one single 4×4 filter with a stride of 2 and no padding, what is the **number of parameters** you are learning in this layer?

   Your Answer

   (d) (1 point) Now suppose we do not do parameter sharing. That is, each output pixel of this layer is computed by a separate 4×4 filter. Again we use a stride of 2 and no padding. What is the **number of parameters** you are learning in this layer?

   Your Answer

(e) (1 point) Now suppose you are given a 40×40 colored image, which consists of 3 channels, each representing the intensity of one primary color (so your input is a $40 \times 40 \times 3$ tensor). Once again, you attempt to produce an output map without parameter sharing, using a unique 4×4 filter per output pixel, with a stride of 2 and no padding. What is the number of parameters you are learning in this layer?

Your Answer

(f) (1 point) In *one concise sentence*, describe a reason why parameter sharing is a good idea for a convolutional layer applied to image data, besides the reduction in number of learned parameters.

Your Answer

3. Neural the Narwhal was expecting to implement a CNN for Homework 5, but he is disappointed that he only got to write a simple fully-connected neural network.

(a) (2 points) Neural decides to implement a CNN himself and comes up with the following naive implementation:

```
# image X has shape (H_in, W_in), and filter F has shape (K, K)
# the output Y has shape (H_out, W_out)
Y = np.zeros((H_out, W_out))
for r in range(H_out):
    for c in range(W_out):
        for i in range(K):
            for j in range(K):
                Y[r, c] += X[___blank___] * F[i, j]
```

What should be in the *blank* above so that the output `Y` is correct? Assume that `H_out` and `W_out` are pre-computed correctly.

Your Answer

(b) (2 points) Neural now wants to implement the backpropagation part of the network but is stuck. He decides to go to office hours to ask for help. One TA tells him that a CNN can actually be implemented using matrix multiplication. He receives the following 1D convolution example:

> Suppose you have an input vector $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]^T$ and a 1D convolution filter $\mathbf{w} = [w_1, w_2, w_3]^T$. Then if the output is $\mathbf{y} = [y_1, y_2, y_3]^T$, $y_1 = w_1x_1 + w_2x_2 + w_3x_3$, $y_2 = \cdots$, $y_3 = \cdots$. If you look at this closely, this is equivalent to
>
> $$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$
>
> where the matrix $\mathbf{A}$ is given as $\cdots$

What is matrix $\mathbf{A}$ for this $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{w}$? Write only the final answer. Your work will *not* be graded.

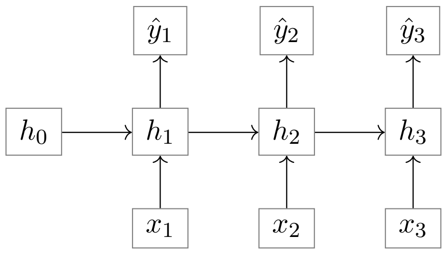
Your Answer

(c) (2 points) Neural wonders why the TA told him about matrix multiplication when he wanted to write the backpropagation part. Then he notices that the gradient is extremely simple with this version of CNN. Explain in *one concise sentence (or one short mathematical expression)* how you can compute $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ once you obtain $\mathbf{A}$ for some *arbitrary* input $\mathbf{x}$, filter $\mathbf{w}$, and the corresponding 1D convolution output $\mathbf{y}$ (so $\mathbf{A}$ is obtained following the same procedure as in part (b), but $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{w}$ can be different from the example). Write only the final answer. Your work will *not* be graded.

Your Answer

3 Recurrent Neural Network (14 points)

1. Consider the following simple RNN architecture:


$$\begin{tikzcd} & \hat{y}_1 & \hat{y}_2 & \hat{y}_3 \\ h_0 \arrow[r] & h_1 \arrow[r] \arrow[u] & h_2 \arrow[r] \arrow[u] & h_3 \arrow[u] \\ & x_1 \arrow[u] & x_2 \arrow[u] & x_3 \arrow[u] \end{tikzcd}$$

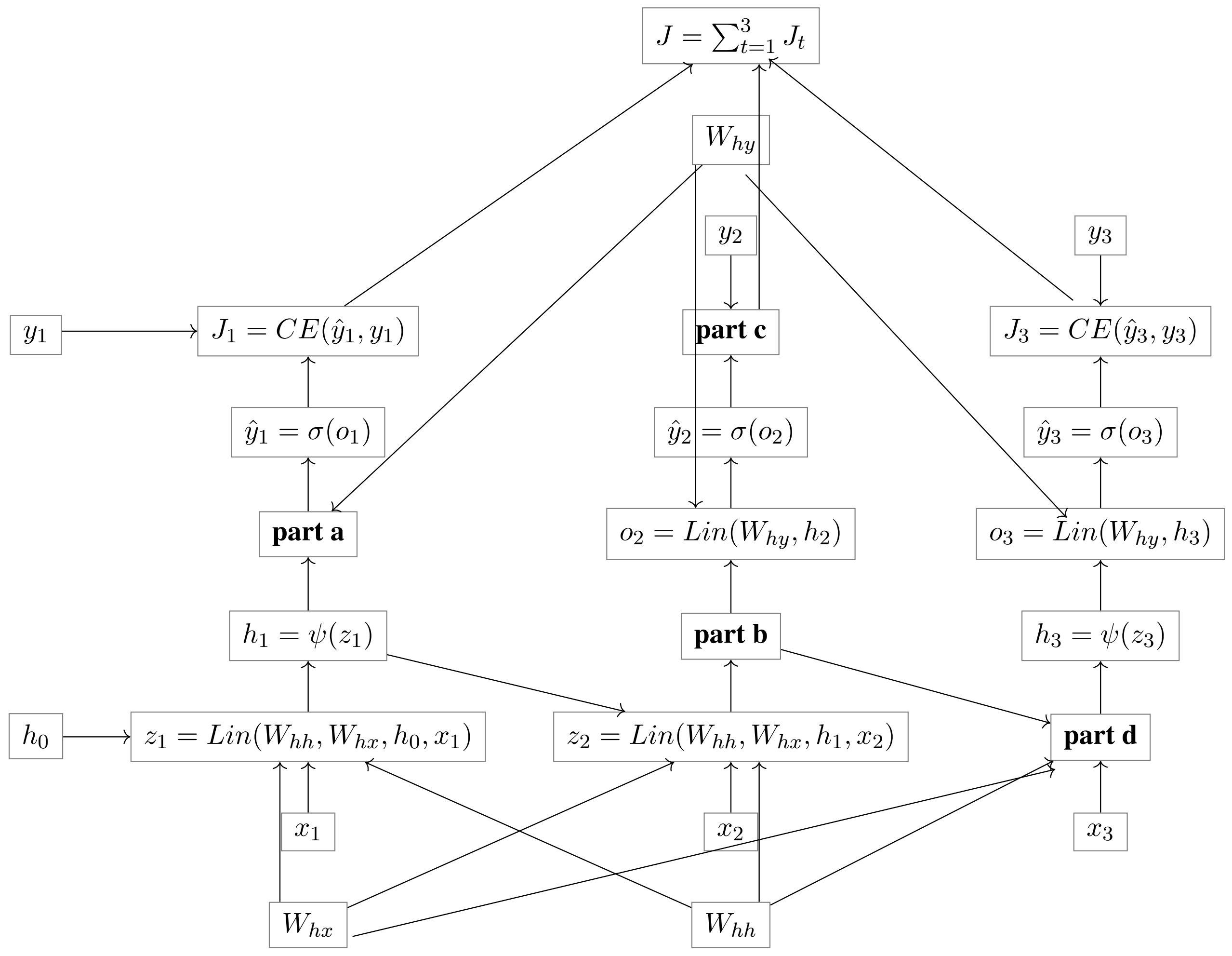
Where the layers and their corresponding weights are given below:

$$\begin{aligned} \mathbf{x}_t &\in \mathbb{R}^3 & \qquad \mathbf{W}_{hx} &\in \mathbb{R}^{4\times 3} \\ \mathbf{h}_t &\in \mathbb{R}^4 & \qquad \mathbf{W}_{hy} &\in \mathbb{R}^{2\times 4} \\ \mathbf{y}_t, \hat{\mathbf{y}}_t &\in \mathbb{R}^2 & \qquad \mathbf{W}_{hh} &\in \mathbb{R}^{4\times 4} \end{aligned}$$

$$\begin{aligned} J &= \sum_{t=1}^{3} J_t \\ J_t &= -\sum_{i=1}^{2} y_{t,i} \log(\hat{y}_{t,i}) \\ \hat{\mathbf{y}}_t &= \sigma(\mathbf{o}_t) \\ \mathbf{o}_t &= \mathbf{W}_{hy}\mathbf{h}_t \\ \mathbf{h}_t &= \psi(\mathbf{z}_t) \\ \mathbf{z}_t &= \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_t \end{aligned}$$

Where σ is the **softmax** activation and ψ is the **identity** activation (i.e. no activation). Note here that we assume that we have no intercept term. J here is computing the cross entropy loss with function $CE()$. t here indicates timesteps.

(a) (4 points) You will now construct the unrolled computational graph for the given model. Use input $\mathbf{x}$, label $\mathbf{y}$, and the RNN equations presented above to complete the graph by filling in the solution boxes for the corresponding blanks.



$J = \sum_{t=1}^{3} J_t$
W_{hy}
y_2
y_3
y_1
$J_1 = CE(\hat{y}_1, y_1)$
part c
$J_3 = CE(\hat{y}_3, y_3)$
$\hat{y}_1 = \sigma(o_1)$
$\hat{y}_2 = \sigma(o_2)$
$\hat{y}_3 = \sigma(o_3)$
part a
$o_2 = Lin(W_{hy}, h_2)$
$o_3 = Lin(W_{hy}, h_3)$
$h_1 = \psi(z_1)$
part b
$h_3 = \psi(z_3)$
h_0
$z_1 = Lin(W_{hh}, W_{hx}, h_0, x_1)$
$z_2 = Lin(W_{hh}, W_{hx}, h_1, x_2)$
part d
x_1
x_2
x_3
W_{hx}
W_{hh}

(a)	(b)

(c)	(d)

(b) Now you will derive the steps of the backpropagation algorithm that lead to the computation of $\frac{dJ}{dW_{hh}}$. For all parts of this question, please write your answer in terms of W_{hh}, W_{hy}, y, $\hat{y}$, h, and any additional terms specified in the question (note: this does not mean that every term listed shows up in every answer, but rather that you should simplify terms into these as much as possible when you can).

i. (2 points) What is $g_{J_t} = \frac{\partial J}{\partial J_t}$? Write your solution in the first box, and show your work in the second.

$\frac{\partial J}{\partial J_t}$

Work

ii. (2 points) What is $g_{o_t} = \frac{\partial J}{\partial o_t}$? Write your solution in the first box, and show your work in the second. Write your answer in terms of $\hat{y}_t$, y_t, and g_{J_t}(Hint: Think about how you can write J_t in terms of o_t, then use the chain rule. You may want to use a result from homework 5 to help here.)

$\frac{\partial J}{\partial \mathbf{o}_t}$

Work

iii. (2 points) What is $g_{\mathbf{h}_i} = \frac{\partial J}{\partial \mathbf{h}_i}$ for an arbitrary $i \in [1, 3]$? Write your solution in terms of $\mathbf{g}_{\mathbf{o}_t}$, W_{hh}, W_{hy} in the first box, and show your work in the second. (Hint: Find $\frac{\partial o_t}{\partial h_i}$, then use the chain rule. Also, for a given i, think about which o_t's h_i affects)

$\frac{\partial J}{\partial \mathbf{h}_i}$

Work

iv. (2 points) What is $g_{\mathbf{W}_{hh}} = \frac{\partial J}{\partial \mathbf{W}_{hh}}$? Write your solution in terms of $\mathbf{g}_{\mathbf{h}_i}$ and h in the first box, and show your work in the second (Hint: W_{hh} is in every timestep, so you need to consider that in the derivative).

$\frac{\partial J}{\partial W_{hh}}$

Work

2. (2 points) **Select all that apply:** Which of the following are true about RNN and RNN-LM?

- ☐ An RNN cannot process sequential data, whereas an RNN-LM is designed for sequential data processing such as in natural language processing.
- ☐ An RNN-LM is only exclusively used as an encoder, which can process sequential data and encode it into a fixed-size state vector.
- ☐ An RNN-LM includes additional layers and structures specifically designed to predict the next token in a sequence, making it more suited for tasks like text generation than a standard RNN.
- ☐ The RNN-LM is trained to maximize the probability of a sequence of tokens, given a previous sequence, which is not a typical training objective of a standard RNN.
- ☐ None of the above.

4 Transformers and AutoDiff (7 points)

```
global tape = stack()

class Module:

    method init()
        out_tensor = null
        out_gradient = 1

    method apply_fwd(List in_modules)
        in_tensors = [x.out_tensor for x in in_modules]
        out_tensor = forward(in_tensors)
        tape.push(self)
        return self

    method apply_bwd():
        in_gradients = backward(in_tensors , out_tensor , out_gradient)
        for i in 1,..., len(in_modules):
            in_modules[i].out_gradient += in_gradients[i]
        return self

function tape_bwd():
    while len(tape) > 0
        m = tape.pop()
        m.apply_bwd()
```

1. (1 point) **Select one:** This is a code snippet from lecture 18 slide 16. In the context of the `method apply_fwd()` inside the `Module` class, what is the primary role of the `tape.push(self)` command that pushes the module onto the tape?

   ◯ It records the current module onto the stack along with its parameters and tensors to ensure that the output tensor is saved for the backward pass.

   ◯ It pushes the current computation's gradient onto the stack for immediate use in the forward pass.

   ◯ It duplicates the module to allow for parallel computations in subsequent layers of the neural network.

   ◯ It activates the module for the forward pass, making it the only active computation in the network.

2. (2 points) **True or False:** We can replace a stack with a queue in Module-based AutoDiff. Explain your reasoning in no more than 2 sentences in the box below.

   ◯ True

   ◯ False

Your Answer

3. (2 points) We are looking to replace Neural the Narwhal with another mascot that is **cute and efficient**. We would want you to use a transformer that helps us decide a **mascot name** by searching in a large pool of possible mascots. How would the transformer help decide which mascot to choose? Suggest a possible **key**, **value** pair that it would consider. Be specific about which is the key and which is the value.
Hint: When you conduct a search on a web search engine (e.g. Google), the search engine compares your entered query with a set of **keys** (e.g. webpage titles, content, and metadata) of web pages in its database. It then displays the most relevant web pages as search results (**values**).

Your Answer

4. Consider a Transformer model employing a multi-headed self-attention mechanism. Suppose the input consists of a sequence of T tokens, each token represented by a d_{model}-dimensional embedding vector. This model utilizes H attention heads. During the attention process, each head generates key, query, and value matrices from the input embeddings. The dimensionality of the key and query vectors is d_k for each head, and the attention function produces an output vector of dimension d_v for each token and head.

   (a) (1 point) Which of the following represents the dimension of the key matrix for a single attention head?

   ◯ $T \times d_v$

   ◯ $H \times d_k \times d_{\text{model}}$

   ◯ $T \times d_k$

   ◯ $T \times d_{\text{model}} \times d_k$

   (b) (1 point) Which of the following represents the dimension of the output matrix of the multi-headed attention before any final linear transformation?

◯ $T \times H \times d_k$

◯ $T \times H \times d_v$

◯ $T \times d_{\text{model}}$

◯ $H \times d_k \times d_v$

5 Pre-training, Fine-tuning, In-context Learning (7 points)

1. (2 points) Select the correct statements about the pre-training and fine-tuning phases:

   ☐ The pre-training dataset usually requires a large amount of labeled data to guide the initial learning process and shape the weight configurations.

   ☐ The fine-tuning dataset is typically much larger than the pre-training dataset and contains a wide variety of examples from numerous domains to ensure that the model can generalize well to any task.

   ☐ Reconstruction error is a metric used during the fine-tuning process to measure the accuracy of a model's predictions, typically calculated as the percentage of correct labels the model assigns during supervised learning tasks

   ☐ None of the above.

2. (2 points) Select the correct statements about in-context learning:

   ☐ Zero-shot learning models usually make accurate predictions for tasks they have not explicitly been trained for, relying solely on a rich, pre-trained representation of data.

   ☐ One-shot learning models require only a single example to learn a new task, often leveraging this example to make analogies and predictions for similar tasks.

   ☐ Few-shot learning models typically require thousands of examples to learn a new task effectively and are not practical for data-scarce scenarios.

   ☐ In-context learning requires updating the model's parameters with gradient descent on each new example to make accurate predictions.

   ☐ None of the above.

3. (3 points) Llama2 (https://www.llama2.ai/) is Meta's open source large language model. Interact with llama2 with your own topic of choice (Please try to stick to simple mathematical operations to do your analysis). In the answer box below, put the prompt and the output from Llama2 for (1) a zero-shot scenario and (2) a few-shot in-context learning scenario in which the zero-shot provides poor feedback but the few-shot performs better. Summarize what you observe in one sentence.

Your Answer

6 Empirical Questions (8 points)

The following questions should be completed as you work through the programming component of this assignment. **Please ensure that all plots are computer-generated**.

1. (3 points) Plot the **training loss** against **number of iterations** during pre-training. The code provided automatically prints the loss every 2000 iterations.

Your Answer

2. (5 points) Plot the **validation accuracy** for training sizes [16, 32, 64, 128, 256, 512] for your fine-tuned model with pre-training and without pre-training. Please put them in the same plot.

Your Answer

7 Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details.
3. Did you find or come across code that implements any part of this assignment? If so, include full details.

Your Answer

8 Programming (30 points)

8.1 The Task

In the programming section you will partially implement a transformer, pre-train it on WikiText2 dataset and fine-tune it for sentiment analysis on movie reviews. On a technical level the main task is to implement the attention layer of the transformer, pre-train and fine-tune the model, and evaluate if pre-training helps improve performance or not.

The link to the python notebook `transformer.ipynb` is here. Use your CMU Google Account to access the file. Make a copy of the file by clicking on the File button on the top left corner (right below the file name) and select ”Save a Copy in Drive”. Work on that copy and upload your completed transformer.ipynb file to Gradescope.

8.2 The Dataset

WikiText2 dataset is a publicly available dataset used for pretraining in this notebook, which is a collection of over 2 million words extracted from Wikipedia articles and is commonly used for language modeling tasks.

8.2.1 Data Preparation

The dataset is prepared using SentencePiece, a text tokenizer and detokenizer mainly used for Neural Network Machine Translation applications. In this notebook, SentencePiece is utilized to generate a tokenizer model from the Wikitext training tokens, and this model is used for tokenizing the dataset into subword units (Byte Pair Encoding - BPE). The vocabulary size is set to 50,000, which is a common size for many language tasks.

8.2.2 Data Loading and Batching

A custom Corpus class is defined to handle the loading and numericalization of the tokens from the dataset files. The class reads and processes the following files:

```
wiki.train.tokens
wiki.valid.tokens
wiki.test.tokens
```

These files contain the training, validation, and test sets respectively, with tokens already preprocessed and separated by spaces.

For training purposes, the notebook defines functions to generate random and sequential batches of these tokens. The batch size and sequence length are configurable, but as an example, the notebook uses a batch size of 32 and a sequence length of 65 tokens.

8.2.3 Data Iteration

Two types of batch samplers are used:

random_batch_sampler: Generates random batches of tokens, which is often used in training to introduce stochasticity.
sequential_batch_sampler: Generates batches of tokens sequentially, which can be useful for validation and testing to preserve the sequence order.

Both samplers ensure that the tokens are appropriately batched and placed onto the correct device (CPU or GPU) for model training.

8.2.4 File Format

The text data in the dataset files is raw and unstructured, requiring the Corpus class to handle tokenization and numericalization. We use subword tokenization to ensure there are not out-of-vocabulary words.

8.3 Learning

Your first task is to implement the multi-headed attention layer to complete the implementation of the transformer model. This implementation should be designed to integrate with the existing model architecture provided in the notebook. The attention layer is critical for the model to weigh the importance of different tokens in the input sequence when predicting the next token.

This layer will compute dynamic weights for different parts of the input sequence, thereby allowing the model to prioritize which information to attend to during the learning process.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where

Q represents the matrix of queries.
K represents the matrix of keys.
V represents the matrix of values.
d_k is the dimensionality of the keys and queries.

Note: There are hints and TODOs to guide you through the implementation. While implementing if you plan to use any helper function, define it in the same cell where it is being called. The reference implementation takes around 1 hour to complete pre-training.

8.4 Transformer Architecture for Pre-training and Fine Tuning

8.4.1 Transformer Language Model (TransformerLM)

The Transformer class defines a standard transformer architecture primarily used for language modeling tasks. This model takes a sequence of token indices (representing words or subwords in a sentence) as input and predicts the next token in the sequence. Key components include:

Embedding Layer: Converts token indices (which are numerical representations of words) into dense vector representations. These dense vectors capture more nuanced information about words than mere indices, allowing the model to understand word meanings and relationships.

Positional Encoding: Adds information about the position of each token in the sequence. This helps the model understand the order of words in a sentence, which is crucial for interpreting the meaning correctly. It's essential because the model processes input in parallel and doesn't inherently capture sequential information.

Decoder Layers:
Multi-Head Attention layer allows the model to focus on different parts of the input sentence simultaneously, capturing different types of relationships between words. It's implemented through the MultiHeadAttention class, which splits the input into multiple heads, allowing the model to attend to different segments of the input simultaneously.
Position-Wise FeedForward layer applies a simple neural network to each position's output from the attention mechanism, further processing the information.

Output Layer: Transforms the final output of the decoder layers into a format that matches the size of the model's vocabulary.This layer is used for predicting the next word in the sequence, with each element in the output corresponding to a probability score for each word in the vocabulary.

Masking: The generate_mask function in the Transformer class creates a mask to hide future tokens. This is crucial during training to ensure that the prediction for a particular token doesn't depend on the future tokens

8.4.2 Classification Transformer (Fine-Tuning)

Architecture: The ClassificationTransformer class adapts the pre-trained TransformerLM for a classification task. It reuses the embedding and positional encoding layers from the pre-trained model and adds a final fully connected layer for classification.

Adaptations for Classification:
Reuses Pre-Trained Components: Embeddings and positional encodings from the pre-trained model are directly utilized, leveraging the knowledge acquired during the language modeling phase.
Final Classification Layer: A linear layer that takes the output of the last token (or a special classification token) and projects it to the number of classes for classification tasks.

8.5 Training and Evaluation:

8.5.1 Pre-Training

Forward Pass: The model processes the input data and generates predictions for the next token.
Loss Calculation: The Cross-Entropy Loss between the predictions and the actual next tokens is computed.
Backpropagation: The gradients of the loss are calculated and used to update the model's weights.
Optimizer: An optimizer like Adam is commonly used to adjust the weights to minimize the loss.

8.5.2 Fine-Tuning Process

Forward Pass: The input data goes through the model, which now predicts class labels instead of the next word.
Loss Calculation: Cross-Entropy Loss is calculated between the predicted labels and the true labels.
Backpropagation and Optimization: Similar to the pre-training phase, but the focus is more on fine-tuning the weights, especially in the newly added output layer.

Please make sure that you use `to(device)`
Need for `to(device)`: These calls ensure that the model and data are moved to the GPU, which provides faster computation compared to a CPU, especially beneficial for training large models like Transformers. Pre-training a Transformer model is a computationally intensive task due to the model's size and complexity, and the large size of the training datasets. Utilizing a GPU significantly speeds up this process.

8.6 Gradescope Submission

You should submit your `transformer.ipynb` to Gradescope. **Any other files will be deleted.** Please do not use other file names. Do not modify cells that does not explicitly say so. Some additional tips: For this programming assignment most of the implementation is provided to you and following the hints and TODOs carefully should help you get through them.