

HOMWORK 4: DEEP LEARNING

10-701 Introduction to Machine Learning
(PhD) (Spring 2024)

Carnegie Mellon University

pi piazza.com/cmu/spring2024/10701/home

OUT: Wednesday, Feb 28th, 2024*

DUE: Friday, Mar 15th, 2024 11:59 PM

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., “Jane explained to me what is asked in Question 2.1”). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section in our course syllabus for more information: <https://www.cs.cmu.edu/~hchai2/courses/10701/#Syllabus>
- **Late Submission Policy:** See the late submission policy here: <https://www.cs.cmu.edu/~hchai2/courses/10701/#Syllabus>
- **Submitting your work:**
 - **Gradescope:** There will be two submission slots for this homework on Gradescope: Written and Programming.
For the written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using the written submission slot. Please use the provided template. The best way to format your homework is by using the Latex template released in the handout and writing your solutions in Latex. However 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. Each derivation/proof should be completed in the boxes provided below the question, **you should not move or change the sizes of these boxes** as Gradescope is expecting your solved homework PDF to match the template on Gradescope. If you find you need more space than the box provides you should consider cutting your solution down to its relevant parts, if you see no way to do this, please add an additional page at the end of the homework and guide us there with a ‘See page xx for the rest of the solution’.
You are also required to upload your code, which you wrote to solve the final question of this homework, to the Programming submission slot. Your code will be run by TAs so please make sure it is in a workable state.
Regrade requests can be made after the homework grades are released, however this gives the

*Compiled on Wednesday 28th February, 2024 at 22:28

TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For \LaTeX users, use \blacksquare and \bullet for shaded boxes and circles, and don't change anything else. If an answer box is included for showing work, **you must show your work!**

1 Convolutional Neural Networks (8 Points)

1. **[2pts] Numerical answer:** Suppose the first layer of a CNN uses a $3 \times 5 \times 5$ convolutional filter and takes as input images of size 100×100 . It also uses a padding of 3 and a stride 2. What are the dimensions of the output of this layer?

2. **[2pts] Numerical answer:** Now suppose you have a CNN that takes inputs of shape $3 \times 100 \times 100$ and the first convolutional layer uses 5×5 filters and outputs 8 channels. How many parameters does this layer have (including a bias term for each output channel)?

3. **[2 pts] Select all that Apply:** Which of the following statements about pooling in CNNs is/are true?

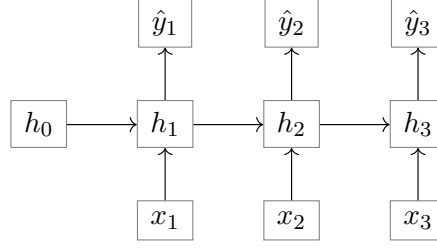
- ☐ Pooling is never used more than once in a single CNN because of image information loss
- ☐ Max pooling preserves the salient features of an image
- ☐ Average pooling is used significantly more often than max pooling in modern CNNs
- ☐ Max pooling is incompatible with backpropagation because the max function is not differentiable
- ☐ None of the above

4. **[2 pts] Select all that Apply:** Which of the following statements about 1×1 convolutions is/are true?

- ☐ 1×1 convolutions introduce non-linearity between two CNN layers
- ☐ 1×1 convolutions can be formulated as a fully connected layer between pixel-wise input channels and output channels
- ☐ 1×1 convolutions often reduce the overall number of parameters in a model
- ☐ 1×1 convolutions preserve the channel dimensionality but may impact other dimensions
- ☐ None of the above.

2 Recurrent Neural Networks (10 Points)

1. Consider the following simple RNN architecture:



where the layers and their corresponding weights are given below:

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

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

Here, σ is the **softmax** activation and ψ is the **identity** activation (i.e. no activation). J is the cross entropy loss and t indicates timesteps. Note that we have no intercept term in this architecture.

In this question, you will derive the steps of the backpropagation through time 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.

Homework 4: Deep Learning

- (a) **[2 pts]** 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 $G_{J_t} = \frac{\partial J}{\partial J_t}$.

$$\frac{\partial J}{\partial o_t}$$

Work

- (b) **[4 pts]** What is $G_{h_i} = \frac{\partial J}{\partial h_i}$ for an arbitrary $i \in [1, 3]$? Write your solution in terms of G_{o_t}, W_{hh}, W_{hy} in the first box, and show your work in the second.

$$\frac{\partial J}{\partial h_i}$$

Work

Homework 4: Deep Learning

- (c) **[4 pts]** 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.

$$\frac{\partial J}{\partial \mathbf{W}_{hh}}$$

Work

3 Attention & Transformers (12 Points)

1. **[4pts] Short Answer:** What is the primary limiting factor in a Transformer architecture for processing longer sequence lengths? Does this limitation also apply to RNNs? Why or why not?

2. **[2 pts] Select One:** Given an input sequence of length n , how does the time complexity of the self-attention mechanism scale with n ?

- ☐ $O(n)$
☐ $O(n \log n)$
☐ $O(n^2)$
☐ $O(n^3)$

3. **[6pts] Short Answer:** While training a decoder network in a Seq2Seq model, to pass an input to the i^{th} time step, would it be preferable to pass in the prediction generated at the previous time step (\hat{y}_{t-1}) or would it be preferable to pass in the ground truth token (y_{t-1})? What are the benefits and drawbacks of each?

4 Programming: RNN and Transformer in PyTorch [27 pts]

In this programming task, you will implement an RNN and a transformer encoder in PyTorch. The task we will be working on is sentiment analysis on the IMDB dataset. The IMDB dataset contains 50K text reviews of movies and your neural networks will predict whether they are generally positive or generally negative reviews, a binary classification task on text data.

Important Note on PyTorch: Unlike in HW3, in this programming task you are REQUIRED to use the automatic differentiation in PyTorch. You are allowed to use most of the built-in neural network modules in PyTorch (such as `nn.Linear`, `nn.ReLU`, `nn.Softmax`, etc) unless otherwise specified.

PyTorch modules that are NOT allowed in this task:

1. All recurrent layers: `nn.RNNBase`, `nn.RNN`, `nn.RNNCell`, `nn.LSTM`, `nn.LSTMCell`, `nn.GRU`, `nn.GRUCell`
2. All attention and transformer layers: `nn.MultiHeadAttention`, `nn.Transformer`, `nn.TransformerEncoder`, `nn.TransformerDecoder`, `nn.TransformerEncoderLayer`, `nn.TransformerDecoderLayer`,

4.1 The IMDB Dataset

The IMDB dataset is stored in `IMDB_Dataset.csv`: this file contains 50,000 entries and each entry has two columns. The first column contains the text reviews with lengths ranging from a few dozen words to more than a thousand words. The second column contains a string label being either “positive” or “negative”, specifying the overall sentiment of the review. For efficiency, we will only take the first $L = 50$ words in each review for training and testing. We provide a dataset class and function to initialize the dataset and the dataloader for you in `dataset.py`. **Please carefully read and make sure you understand all the code in this file.**

The `IMDB_Dataset` class preprocesses the text reviews by translating each word into an index representing that word. The word-to-index mapping is stored in the `self.word_dict` variable. Note that we include an additional padding token “PAD” that has index 0. We append the padding token to reviews that have lengths less than L , so that all reviews have the same lengths and can be batched together.

The dataloader will return a tuple containing the text tensor and the label tensor. If B is the batch size, then the text tensor has shape (B, L) , where each element represents the index of the word at that specific position. The label is a one-dimensional tensor with length B , where each element is either 1 or 0, with 1 representing positive reviews and 0 representing negative.

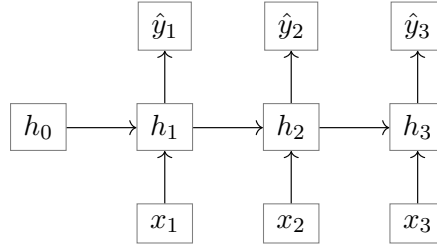
4.2 Implementing an RNN

4.2.1 Word Embedding

To convert the word indices into a word embedding, create an embedding layer using `torch.nn.Embedding`. The embedding layer takes in batched, word-index tensors with shape (B, L) and outputs an embedding X with shape (B, L, D_e) , where the hyperparameter D_e is the embedding dimension. The embedding layer maintains the embedding table tensor of shape (N, D_e) that will be optimized during training. N is the number of words in the dictionary. The k -th row in the embedding table will be the output vector corresponding to the input word at index k . Note that for the padding token “PAD” with index 0, we want to enforce its embedding to be a zero vector.

4.2.2 LayerNorm

Layer normalization (LayerNorm) is a common technique used in deep learning. It computes the mean and variance of a tensor across the feature dimension (as opposed to the batch dimension in BatchNorm) and normalizes the tensor with respect to the computed mean and variance. It then learns a new mean and variance for the output and scales the normalized tensor according to those learned parameters. You should use `nn.LayerNorm` to perform layer normalization in your networks.



4.2.3 RNN

Given a working dataloader and embedding layer, we will now implement a simple RNN with one hidden layer. The structure of the network is shown in the above diagram. x_i, h_i, y_i represents the input word embedding, hidden state, and output at the i^{th} recurrent step respectively. Your network should contain three linear layers: input-to-hidden, hidden-to-hidden, and hidden-to-output layer, with weights represented by W_{xh}, W_{hh} , and W_{hy} respectively. You should initialize h_0 as a zero vector. Given previous hidden state h_{i-1} and the i -th word embedding x_i , you can compute h_i and y_i as

$$h_i = \tanh(\text{LayerNorm}(W_{hh}h_{i-1} + W_{xh}x_i)),$$

$$y_i = \text{softmax}(W_{hy}h_i).$$

4.2.4 Loss and Prediction

Our RNN computes an output at every recurrent step. You need to compute the loss for all of the outputs and sum over the recurrent steps to compute your total loss for each iteration. Use `nn.CrossEntropy` as the loss criterion. However, for prediction, we only take the output at the very last recurrent step y_L , and predict the label by taking the argmax of the last output.

4.2.5 Train and Test your RNN

After finishing your RNN implementation, initialize your model and initialize a PyTorch optimizer that optimizes the parameters in your model. For this assignment you should use the Adam optimizer. Write a training loop over the training dataloader. For each batch of inputs, run the forward and backward passes, then update the model using the optimizer. Train your model using the hyperparameters specified in the starter code. For testing your model, loop over the test dataloader and run a forward pass with `torch.no_grad()`. For this step, **we highly recommend reviewing Recitation 5 on implementing the training loops in PyTorch.**

4.3 Implementing a Transformer Encoder

4.3.1 Transformer Embedding

For the transformer, we will use the same method as described above to convert word indices into embeddings. However, on top of that, we also need positional encodings to tell the model the location of each

embedding. We add the positional encoding to the word embedding as the input to the model. The class `TransformerEmbedding` does this and is provided for you in the starter code.

4.3.2 Multi-head Attention Layer

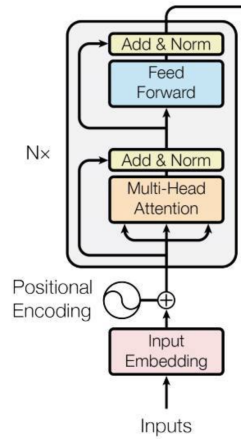
Implement the `AttentionLayer` class and then the `MultiHeadAttention` class. For single-head attention, given input X with shape (B, L, D_e) .

$$Q = W_Q X, K = W_K X, V = W_V X$$

where Q has shape (B, L, D_Q) and D_Q is the dimension of Q ; K and V behave similarly. Note that the dimensions of Q and K must be the same. Now we can compute the attention score and the final output of the attention layer as

$$score = \text{softmax} \left(\frac{QK^T}{\sqrt{D_Q}} \right), z = score V$$

where $score$ has shape (B, L, L) and z has shape (B, L, D_V) . Multi-head attention with N heads should contain N attention layers. It computes outputs for each attention head and concatenates the outputs z_1, z_2, \dots, z_N . The final output has shape $(B, L, N \times D_V)$.



4.3.3 Transformer Encoder

The structure of a transformer encoder layer is shown in the above figure. Given input X , it computes the intermediate tensor z and output y as

$$z = \text{LayerNorm}(X + \text{MultAttn}(X)), y = \text{LayerNorm}(z + FF(z))$$

where `MultAttn` represents multi-head attention and `FF` represents a feed-forward network, which consists of two linear layers and one ReLU activation after the first linear layer. In addition, note that we add back in the input X and intermediate tensor z before performing layer normalization, creating “residual connections”. In your implementations, z and y should have the same shape as input x . Implement the `FeedForward`, `Residual`, and the `EncoderLayer` classes. The transformer encoder contains multiple `EncoderLayer` modules and passes the input embedding through all the layers sequentially.

4.3.4 Loss and Prediction

For loss and prediction, we will use the output of the *first* word token “SOS”, a dummy token we add to each text review. We pass this output to one more linear layer to reduce the dimension to 2, the number of classes in our task, and apply a softmax to get the predicted probabilities. Again, use cross entropy for the loss and argmax for the prediction.

4.3.5 Train and Test your Transformer

We encourage you to reuse the same code you wrote for training and testing your RNNs to train and test your Transformers.

4.4 Empirical Questions

1. **[4 pts]** Train both the RNN and the Transformer encoder for 10 epochs using the hyperparameters specified in the code provided for you. Report the accuracy of both models on both the test dataset and the training dataset. Report values up to the 4th decimal place.

RNN Training Accuracy:

RNN Test Accuracy:

Transformer Training Accuracy:

Transformer Test Accuracy:

2. **[5 pts]** Describe the relationship between the values you reported in the previous question: does one model significantly outperform another? Is one more liable to overfit than the other? Provide an explanation for the relative performances of the two models using what you know of the task and how we preprocessed the dataset.

3. **[8 pts]** For each of the two models, plot a graph showing both the training and test loss vs epochs on the x-axis. Please make sure your graphs are properly labeled and include a legend.

RNN loss curves

Transformer encoder loss curves

4. **[10 pts]** Now we want to give you a chance to experiment with the two models by exploring their hyperparameters. For both the RNN model and the Transformer encoder model, produce a pair of plots that show the train and test loss for a few different values/settings of a hyperparameter; these should be similar to the ones you produced in the batch size experiment in HW3. In addition, note how changing the selected hyperparameter affects the time required to train your model (if at all). Experiment with at least 3 different values for the hyperparameters that you decide to modify. For your experiments, you must
- describe the hyperparameter you varied
 - list the values you set the hyperparameter to
 - detail the setting of all the other hyperparameters
 - report the time required to train the model for each setting of the hyperparameter, and
 - analyze your results in terms of the hyperparameter that you varied.

For your reference here are just some of the hyperparameters you could consider changing:

- Embedding dimension
- Number of recurrent/transformer layers
- Length of the text sequences considered

This question is for learning so feel free to experiment with whatever hyperparameters you wish to, as long you can provide a reasonable explanation for the observed behavior in the graphs. See if you can get the model that originally performed worse to outperform the other one or see how well you can preserve accuracy while decreasing the training time of a model; impress us!

RNN hyperparameter experiment

Transformer encoder hyperparameter experiment

5 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment? **Solution Yes / No.**
(b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

Solution

2. (a) Did you give any help whatsoever to anyone in solving this assignment? **Solution Yes / No.**
(b) If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

Solution

3. (a) Did you find or come across code that implements any part of this assignment? **Solution Yes / No.**
(b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).

Solution