Announcements

April 20, 2024

Remarks:

1. Poster presentations will be on Wednesday, April 24th between 12-1:30pm. Location is EECS atrium. There are 30 posters, so make sure that you have **2 minute pitch** to present your work.

2. We will provide easels and boards (3' x 4'). You need to print your poster. You will need 4 3M command poster strips to hang your poster. We will also try to accommodate this (but bring it with you if you have it).

3. Project report deadline is Sunday, April 28th.

4. **HW4&5:** You can reduce the context length to 32 if you are having trouble with the training time.

5. **HW4&5:** During test evaluation, note that positional encodings for unseen/long context are not trained. You are supposed to evaluate it as is. It is OK if it doesn't work well.

6. **HW4&5 (optional but strongly recommended):** You should add residual connection between blocks. This will likely improve the accuracy of your runs and will also speed up convergence.

    - Since results with and without residual connections are different, please clarify whether you used it or not at the very start of your HW report.

7. **HW4&5:** Comments are an important component of the HW grade. You are expected to explain the experimental findings. If you don't provide technically meaningful comments, you might receive a lower score even if your code and experiments are accurate.

Deadline: 11:59 PM, Thursday April 18, 2024 (no extension)

The objective of this assignment is comparing transformer architecture and SSM-type architectures (specifically Mamba [1]) on the associative recall problem. We provided an example code `recall.ipynb` which provides an example implementation using 2 layer transformer. You will adapt this code to incorporate different positional encodings, use Mamba layers, or modify dataset generation.

Background: As you recall from the class, associative recall (AR) assesses two abilities of the model: Ability to locate relevant information and retrieve the context around that information. AR task can be understood via the following question: Given input prompt $X = [a\ 1\ b\ 2\ c\ 3\ b]$, we wish the model to locate where the last token b occurs earlier and output the associated value $Y = 2$. This is crucial for memory-related tasks or bigram retrieval (e.g. 'Baggins' should follow 'Bilbo').

To proceed, let us formally define the associative recall task we will study in the HW.

Definition 1 (Associative Recall Problem) *Let $\mathcal{Q}$ be the set of target queries with cardinality $|\mathcal{Q}| = k$. Consider a discrete input sequence X of the form $X = [\dots\ q\ v\ \dots\ q]$ where the query q appears exactly twice in the sequence and the value v follows the first appearance of q. We say the model f solves AR(k) if $f(X) = v$ for all sequences X with $q \in \mathcal{Q}$.*

Induction head is a special case of the definition above where the query q is fixed (i.e. $\mathcal{Q}$ is singleton). Induction head is visualized in Figure 1. On the other extreme, we can ask the model to solve AR for all queries in the vocabulary.

Problem Setting

Vocabulary: Let $[K] = \{1, \dots, K\}$ be the token vocabulary. Obtain the embedding of the vocabulary by randomly generating a $K \times d$ matrix $\boldsymbol{V}$ with IID $\mathcal{N}(0, 1)$ entries, then normalized its rows to unit length. Here d is the embedding dimension. The embedding of the i-th token is $\boldsymbol{V}[i]$. Use numpy.random.seed(0) to ensure reproducibility.

Experimental variables: Finally, for the AR task, $\mathcal{Q}$ will simply be the first M elements of the vocabulary. During experiments, K, d, M are under our control. Besides this we will also play with two other variables:

- **Context length:** We will train these models up to context length L. However, we will evaluate with up to $3L$. This is to test the generalization of the model to unseen lengths.
- **Delay:** In the basic AR problem, the value v immediately follows q. Instead, we will introduce a delay variable where v will appear τ tokens after q. $\tau = 1$ is the standard.

Models: The motivation behind this HW is reproducing the results in the Mamba paper. However we will also go beyond their evaluations and identify weaknesses of both transformer and Mamba architectures. Specifically, we will consider the following models in our evaluations:

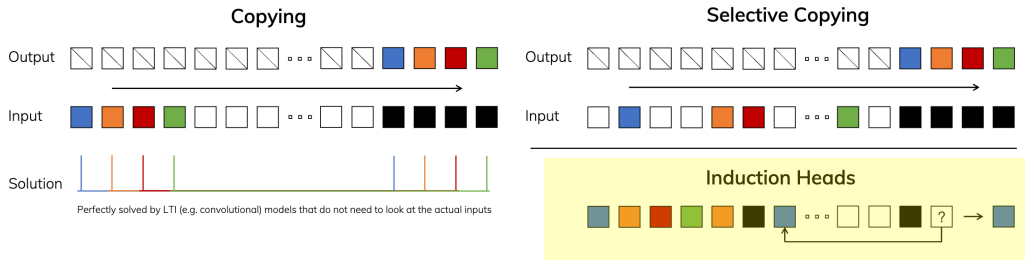

Figure 2: (*Left*) The standard version of the Copying task involves constant spacing between input and output elements and is easily solved by time-invariant models such as linear recurrences and global convolutions. (*Right Top*) The Selective Copying task has random spacing in between inputs and requires time-varying models that can *selectively* remember or ignore inputs depending on their content. (*Right Bottom*) The Induction Heads task is an example of associative recall that requires retrieving an answer based on context, a key ability for LLMs.

Figure 1: We will work on the associative recall (AR) problem. AR problem requires the model to retrieve the value associated with all queries whereas induction head requires the same for a specific query. Thus, the latter is an easier problem. The figure above is directly taken from the Mamba paper [1]. The yellow-shaded regions highlight the focus of this homework.

- **Transformer:** We will use the transformer architecture with 2 attention layers (no MLP). We will try the following positional encodings: (i) learned PE (provided code), (ii) Rotary PE (RoPE), (iii) NoPE (no positional encoding)
- **Mamba:** We will use the Mamba architecture with 2 layers.
- **Hybrid-A:** We will use an initial Mamba layer followed by an attention layer. No positional encoding is used.
- **Hybrid-B:** We will use an initial Mamba layer followed by an attention layer followed by a Mamba layer. No positional encoding is used.

Hybrid architectures are inspired by the Mamba paper as well as [2] which observes the benefit of starting the model with a Mamba layer. You should use public GitHub repos to find implementations (e.g. RoPE encoding or Mamba layer). Official repo is but you can also use ...

Generating training dataset: During training, you train with minibatch SGD (e.g. with batch size 64) until satisfactory convergence. You can generate the training sequences for AR as follows given (K, d, M, L, τ):

1. Training sequence length is equal to L.
2. Sample a query $q \in Q$ and a value $v \in [K]$ uniformly at random, independently. Recall that size of Q is $|Q| = M$.
3. Place q at the end of the sequence and place another q at an index i chosen uniformly at random from 1 to $L - \tau$.

4. Place value token at the index $i + \tau$.

5. Sample other tokens IID from $[K] - q$ i.e. other tokens are drawn uniformly at random but are not equal to q.

6. Set label token $Y = v$.

Test evaluation: Test dataset is same as above. However, we will evaluate on all sequence lengths from $\tau + 1$ to $3L$. Note that $\tau + 2$ is the shortest possible sequence.

Empirical Evidence from Mamba Paper: Table 2 of [1] demonstrates that Mamba can do a good job on the induction head problem i.e. AR with single query. Additionally, Mamba is the only model that exhibits length generalization, that is, even if you train it pu to context length L, it can still solve AR for context length beyond L. On the other hand, since Mamba is inherently a recurrent model, it may not solve the AR problem in its full generality. This motivates the question: What are the tradeoffs between Mamba and transformer, and can hybrid models help improve performance over both?

Your assignments are as follows. For each problem, make sure to return the associated code. These codes can be separate cells (clearly commented) on a single Jupyter/Python file.

Grading structure:

- Problem 1 will count as your **HW4 grade**. This only involves Induction Head experiments (i.e. $M = 1$).
- Problems 2 and 3 will count as your **HW5 grade**.
- You will make a single submission.

Problem 1 (50=25+15+10pts). Set $K = 16$, $d = 8$, $L = 64$.

- Train all models on the induction heads problem ($M = 1, \tau = 1$). After training, evaluate the test performance and plot the accuracy of all models as a function of the context length (similar to Table 2 of [1]). In total, you will be plotting 6 curves (3 Transformers, 1 Mamba, 2 Hybrids). Comment on the findings and compare performance of the models including length generalization ability.
- Repeat the experiment above with delay $\tau = 5$. Comment on the impact of delay.
- Which models converge faster during training? Provide a plot of convergence rate where x-axis is the number of iterations and y-axis is the AR accuracy over a test batch. Make sure to specify the batch size you are using (ideally use 32 or 64).

Problem 2 (30pts). Set $K = 16$, $d = 8$, $L = 64$. We will train Mamba, Transformer with RoPE, and Hybrids. Set $\tau = 1$ (standard AR).

- Train Mamba models for $M = 4, 8, 16$. Note that $M = 16$ is the full AR (retrieve any query). Comment on the results.

- Train Transformer models for $M = 4, 8, 16$. Comment on the results and compare against Mamba's behavior.
- Train Hybrid models for $M = 4, 8, 16$. Comment and compare.

Problem 3 (20=15+5pts). Set $K = 16$, $d = 64$, $L = 64$. We will only train Mamba models.

- Set $\tau = 1$ (standard AR). Train Mamba models for $M = 4, 8, 16$. Compare against the corresponding results of Problem 2. How does embedding d impact results?
- Train a Mamba model for $M = 16$ for $\tau = 10$. Comment if any difference.

References

[1] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[2] Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks. *arXiv preprint arXiv:2402.04248*, 2024.