Bachelor Thesis Assessment

Charles University, Faculty of Mathematics and Physics

June 18, 2024

Author: Tymofii Reizin
Name: Fast Algorithms for Attention Mechanism
Year: 2024
Assessment author: Ing. Uladzislau Yorsh

Transformer neural networks are renowned for their power and scalability given by their attention mechanism. But this mechanism also induces quadratic computational complexity relative to sequence length, which practically limits their ability to process long inputs. The aim of the work was to describe the problem in detail, introduce a theoretical motivation for the chosen method of a faster attention approximation and conduct an experiment to empirically assess the model.

The work can be split into two parts. In the introductory part, the student describes the base Transformer architecture, the potential difficulties of its faster approximation and a chosen method to build a faster attention mechanism. The second part is experimental and serves for verification of the approach on a synthetic task.

The first part is overall very good—it gives a necessary introduction into Transformers and provides a good motivation for the experimental core of the work. The writing is easy to follow, and the outline of the related theory with filled gaps in the proofs clearly demonstrates a good understanding of the complex topic. As a minor drawback, this part has numerous typos and small mistakes in formulas, which do not invalidate the conclusions.

The experimental part of the work consists in implementing the chosen model and assessing it on a synthetic task of reversing numerical sequences. This section is mostly successful—the student managed to design a benchmark, implement all the models and obtain good performance scores, on par with the approximated model. The student admits that he did not manage to demonstrate a latency reduction, arguing that the code needs to be optimized on a library level for the computations. I agree with him, but according to the results from the original paper, I am sure that the speedup could be demonstrated in the larger experimental setup (see Remarks for Section 5).

A more significant shortcoming of the experimental part is an absence of the variance suppression. The provided results are based on a single training run per setup, which combined with only a single training task does not allow to reason about different variants of the proposed approximation. The conclusions the student makes are adequate to the obtained results, but the results themselves could be significantly strengthened by considering this aspect.

The code used to execute experiments is short, well-written and self-documenting. The student demonstrates a good knowledge of the chosen implementation library. All the experiments are reproducible.

Given the complex task the author attempts to tackle, his good demonstrated knowledge of the underlying problem and the experimental results, I recommend this work for defense and suggest to grade it with **B**. The reason for not giving a higher grade is the evaluation part, which is not completely matching the stated problem.

Remarks

Overall writing

- In academic manuscripts full forms (”it is”, ”does not”) are preferred over shortened ones (”it’s”, ”doesn’t”).
- The figures 1.1, 1.2 and all Section 5 plots are of a low resolution. I advise to use vector images (e.g. .svg or generated by `tikz`) in the future.
- Since the majority of the Sections’ 2 and 4 material is cited from the reference papers [AS23] and [KMZ24], it is very recommended to provide a mapping between the definitions/lemmas/theorems there and in the source works.
- Some URLs in citations are broken, e.g. [AS23].

Section 1

- 1.1.3: *”...attention, [...], doesn’t distinguish tokens at different positions in the input sequence.”*

  Encoders/bidirectional models are indeed permutation-equivariant, but recent work shows that causal attention on its own can infer positional information thanks to one-sided context, see NoPE.
- 1.1.4, Eq. 1.1: the shape of M can be inferred from the context as $n \times d_{attn}$, but it is still worth to note it explicitly.

Section 2

- 2.1, Definition 1: *Notice that in equation (2.1) we divide by d instead of $\sqrt{d}$ inside of the exponential function, [...]. This is will be more convenient in the proof, and doesn’t make a difference for the definition since result of attention is invariant for different divisor inside the exponential.*

  This indeed should not affect the overall conclusion about the complexity and importance of the input boundedness (the $\sqrt{d}$ was replaced by d in the source article without explanation), but it is not true that the attention is scale-invariant. Consider $d \to 0$ and $d \to +\infty$ cases when we obtain hardmax and uniform scoring functions respectively.
- 2.1, Lemma 2:
  - *. . . bruteforce algorithm which works in* $\backslash^{\in - \mathcal{C}\gamma}$ (sic!).
    The $\mathcal{O}(n^{2-C\gamma})$ expression is broken there.
  - The expression $\lim_{c\to 0} 1 + c + \log \frac{C}{c} = 0$ is missing the c coefficient near logarithm and the RHS should be 1.
  - $\frac{1}{2d}\left(\frac{d}{2} + \frac{d}{2}||a_j - b_j||_0\right)$
    This and the last parts of the expression are incorrect, since it is missing the additional $\frac{d}{2}$ summand in the parenthesis and the $\frac{d}{2}$ coefficient near norm should not be there, see (2.4) where it is correct.
  - The inequality in (2.5) is in the wrong direction.
  - $\frac{n \cdot \exp(\frac{1}{4}\beta(1 - \frac{t}{d}))}{2n\tau} + \frac{2n}{\exp(\frac{\beta\epsilon t}{4d})} + \tilde{t}$
    There should be $\times$ instead of the left $+$.

Section 3

- 3.1: *... if we set sim*$(A, B) = \frac{\exp(AB^T)}{\sqrt{d_{attn}}}$ *in equation (3.3) we obtain equation (3.2).*

  The $1/\sqrt{d_{attn}}$ scalar should be inside the exponent.

- 3.2, Definition 4: if you are "overriding" a broader term of a mathematical kernel (which includes infinite m-s), it is worth to explicitly mention it there.

- 3.3, Eq. 3.7: the upper indices should be i instead of l_z, as well as in N_i and D_i.

Section 5

- The hardware setup for experiments is not mentioned.

- *We will focus only on accuracy because properly measuring speed isn't possible due to standard libraries not having optimizations for necessary functions.*

  Models using linear implementation of attention take more than 20 times the time it takes for models with quadratic implementation.

  The sequences we use consist of integers from 0 to 9 (inclusive) and have length 50.

  These three issues are likely to be related. The chosen sequence length of 50 is too short to demonstrate a speedup, since it is comparable with the model width and will be significantly dominated by it at the moment of computing the Kronecker expansion. If you take a look at the Fig.1 in [KMZ24], you will notice that they also have much higher training latency until the sequence length reaches 3000-4000 tokens. Note that the original JAX code provided by authors does not use any kind of special optimizations like custom kernels.

- *We train each model for 10000 iterations, with calculation of average loss of 10 samples each 100 iterations. Additionally, during the first 1000 iterations we calculate average loss each 10 iterations.*

  It is clear from the code, but not from the text, whether the periodical loss calculation is for visualization purposes only or applies on weight updating too. It is also not clear, do you mean actual samples or batches.

- *The final values of losses the models achieve after 10000 training steps are approximately 0.19621022, 0.01990822 and 0.00000408*

  Given the batch sizes and loss evaluation periods, the estimations are too noisy for numbers to have so many meaningful digits. It is better to use the scientific format and truncate them, e.g. as $1.96e{-}1, 1.99e{-}1$ and $4.08e{-}6$.

- The experimental setup is tiny and is runnable even on CPU, so it makes sense to make use of it and run multiple experiments per setup to suppress variance. The author clearly realizes that the initial seed has a significant influence on the result:

  From our tests we noticed that all of these functions except softmax and ELU are highly susceptible to initial conditions, sometimes performing very bad and sometimes very well.

  After rerunning the experiments myself I can confirm that the models are very non-robust. In this scenario it is crucial to run multiple instances of the same experiment, since one run is a very weak evidence of one variant being better than another, and certainly does not allow to rank them with confidence.

- *Plots suggest that in the best scenario one of the alternative attention functions manages to beat softmax, as in this case x^8 ends up with a smaller loss. But it is not clear how to reach those cases.*

  In this setup, all the three models end up with a tiny value of the loss function, corresponding to the near-perfect accuracy. I do not see much sense to rank them there, and it would be more clear if you were tracking accuracy along with the loss.

- There are some interesting results which might need a commentary, e.g. high magnitude of coefficients of recorded polynomials. Methodical investigation of their nature is clearly out of scope of the work, but it still would be nice to hear the author interpretation, why it can be so.

- It would also be nice to see some visualizations of the recorded polynomials vs the softmax kernel $\exp(xy^T)$. For example, by taking vectors $x, y \in R^d : |x| = |y| = 1$ and plotting the kernel value versus angle. Tracking gradients during training could also provide some insights as well, for example about the spiky loss behavior.