Arithmetic in Transformers

Jorrit Kruthoff, Sebastian Mizera

January 10, 2024

TL;DR Transformers are not that good at length generalisation in arithmetic tasks. We studied various ways of improving this. Padding and sampling seem to be crucial.

Outline Transformers have been used to tackle all kinds of reasoning problems. Here we study how transformers perform on simple arthimetic tasks or Dyck languages (closing brackets or other opening/closing symbols). These tasks can be learned well with the current transformers, but typically only for ID data and not for OOD data. This means that it can do very well on 10 digit addition, but fails on 11 digit addition. If we want to distill some sort of mathematical reasoning capabilities in the model, this is problematic as it should just learn an algorithm. Various techniques on how this could potentially be mitigated have been proposed, but some involve either introducing small amounts of OOD data in the training set (priming) or simply finetuning [1] or by constructing very specific architectures/algorithms for the task at hand [2–4]. We believe this is not necessarily beneficial, because specific architectures most likely hurts performance on other tasks.

Methodology We want to explore various positional embeddings, sampling techniques, padding/alignment strategies and how they affect the OOD generalisation. Furthermore, we want to do this for supervised learning of encoder and decoder-only transformers as well as a deep Q-network setting. Our base model is a six layer transformer but we sometimes also resort to much smaller models. The tasks we focus on are integer addition (5 for ID and 10-20 for OOD), multiplication and Dyck languages. These tasks are all generalizable to arbitrary length.

Related works There is a lot of work on generalisation capabilities of transformers. Regarding positional embeddings there is the original absolute positional embedding of [5] and the relative embeddings [6–8]. In [9] a comparative study was done, which also shows that no positional embedding (in the decoder only setting) works remarkably well. This is because with a causal structure, the model can learn various positional embeddings.[1] Priming and finetuning was studied in [1]. In [10] transformers where studied from a more computational perspective by asking what type of algorithms they could implement.

Results A few of our results. Let us focus first on integer addition.

1. Padding is crucial for generalisation. We see our models reaching 90+% accuracy on $5 \rightarrow 20$ digit addition sums. This is different when we make the padding random. Remarkably it can still learn addition with high accuracy (90+%), but it will completely fail to generalize. The fact that alignment or padding is important for generalizability, was also used in [11]. We, however find much higher accuracy.

2. Log sampling of the data is important. Giving the model batches that contain (on average) the same number of addition sums of a particular length, boosts performance. This makes intuitive sense, as the model will now get all kinds of sums at an equal rate, in contrast to when one uses linear sampling.

3. Positional embeddings are important, especially relative embedding, but there is a lot of variation. RoPE seems to work best in our setup. We played with various θ, but there was not really a θ that improved performance significantly.

4. Whereas for addition, the performance was boosted with log-sampling, for multiplication this wasn't the case. It did a little for ID, but not OOD. Other 'difficulty' based sampling also do not seem to help much. For instance, we tried assessing the difficulty of a multiplication sum by seeing how many products or each digit are more than 10 or not.

Comments

1. Besides the usual transformer architecture, we also studied the new Mambda architecture, which is different, but in their paper [12] they showed some impressive OOD generalisation results. The tasks were simple, but promising, so we started seeing whether the length generalisation also hold for integer addition. We did not find these models to perform better on OOD data over a range of hyperparameters.

References

[1] S. Jelassi, S. d'Ascoli, C. Domingo-Enrich, Y. Wu, Y. Li, and F. Charton, "Length generalization in arithmetic transformers," `arXiv:2306.15400 [cs.LG]`.

[1] This only seems to work when the model is big enough, otherwise there are not enough parameters to learn this embedding.

[2] L. Kaiser and I. Sutskever, “Neural gpus learn algorithms,” arXiv:1511.08228 [cs.LG].

[3] A. Banino, J. Balaguer, and C. Blundell, “Pondernet: Learning to ponder,” arXiv:2107.05407 [cs.LG].

[4] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, “Universal transformers,” arXiv:1807.03819 [cs.CL].

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., vol. 30. Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[6] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, “Roformer: Enhanced transformer with rotary position embedding,” arXiv:2104.09864 [cs.CL].

[7] R. Csordás, K. Irie, and J. Schmidhuber, “The neural data router: Adaptive control flow in transformers improves systematic generalization,” arXiv:2110.07732 [cs.LG].

[8] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” arXiv:1803.02155 [cs.CL].

[9] A. Kazemnejad, I. Padhi, K. N. Ramamurthy, P. Das, and S. Reddy, “The impact of positional encoding on length generalization in transformers,” arXiv:2305.19466 [cs.CL].

[10] H. Zhou, A. Bradley, E. Littwin, N. Razin, O. Saremi, J. Susskind, S. Bengio, and P. Nakkiran, “What algorithms can transformers learn? a study in length generalization,” arXiv:2310.16028 [cs.LG].

[11] S. Kim, H. Nam, J. Kim, and K. Jung, “Neural sequence-to-grid module for learning symbolic rules,” arXiv:2101.04921 [cs.LG].

[12] A. Gu and T. Dao, “Mamba: Linear-time sequence modeling with selective state spaces,” *arXiv preprint arXiv:2312.00752* (2023) .