A small survey of mathematical abilities of modern transformer architectures*

Bartosz Piotrowski

University of Warsaw, Poland, and Czech Technical University, Prague

Introduction Neural networks (NNs) are versatile tools which established state-of-the-art in multiple domains. In particular, one of the spectacular advances achieved with use of NNs has been in natural language processing (NLP). Today, the dominating kind of a neural model used in this domain is based on the transformer architecture [10]. It was also observed that neural architectures designed for NLP have ability to deal with tasks of symbolic (or algorithmic) nature. These include: recognizing propositional entailment [2], computing integrals [4], solving differential equations [1], normalizing polynomials [6], autoformalization [11], premise selection [5], differentiation, solving linear equations, number base conversion, and many others [9].

It is not well understood how neural models are able to perform algorithmic tasks well. It is also unclear what features of a neural architecture make it more suitable for such tasks. In this work, we make a step towards understanding this. We compare two different architectures – encoder-decoder *versus* decoder-only – and two different modes of training – starting from scratch *versus* fine-tuning a model pre-trained on a natural language dataset. We also want to see what is performance of a modern transformer model trained in a practical, limited setting: training for no more than two days on a single GPU.

Data We took 8 different datasets representing mathematical tasks of varied difficulty: addition, multiplication, differentiation, integration, solving linear equations, division, number base conversion, and normalizing polynomials. The first two were created for the purpose of this work and the remaining six were taken from other works [9, 4, 6]. Each dataset consists of *input-output* examples, where *input* is a query to the model and *output* in an answer that the model is trained to produce. For each of the datasets a hold-out testing set of 10000 examples was drawn. Below there are examples of *input-output* pairs for the linear equations dataset:

input	output
`Solve - 3 8 * h - 6 * h + 4 7 8 + 4 0 2 = 0 for h .`	`9`
`Solve 2 9 * i + 1 3 0 0 = - 3 * i + 4 1 * i - 7 4 * i for i .`	`- 2 0`
`Solve 1 0 4 9 * d = 4 3 1 2 + 5 1 2 9 for d .`	`- 4 5`

We experimentally established that treating single digits as tokens is better then taking whole numbers as tokens, and we preprocessed all the datasets accordingly.

Transformer models We compare two different state-of-the-art transformer architectures:

1. **GPT2** [7]: a decoder-only architecture with 124 million of trainable parameters.
2. **T5** [8]: an encoder-decoder architecture (closely following the original transformer model described in [10]). We use the `T5-small` version of this model with 60 million parameters.

Both GPT2 and T5 proved to perform very well on a range of NLP tasks. For both of them there are available high-quality pre-trained checkpoints released by the authors of the models.[1]

*The author was supported by the grant of National Science Center, Poland, no. 2018/29/N/ST6/02903.

[1] They are available in Huggingface: https://huggingface.co/gpt2, https://huggingface.co/t5-small

dataset	T5		GPT2	
	pretrained	untrained	pretrained	untrained
addition	86.74%	96.95%	98.60%	99.26%
multiplication	24.10%	47.58%	46.54%	68.00%
division	67.23%	70.98%	72.62%	77.16%
number base conversion	0.03%	2.58%	1.63%	3.52%
solving linear equations	37.56%	17.62%	45.57%	47.40%
differentiation	98.84%	95.05%	99.80%	99.75%
integration	26.65%	35.88%	79.70%	81.80%
polynomial normalization	58.13%	90.83%	89.35%	92.93%

Table 1: Final testing accuracy of neural language models tested on the eight datasets.

Experimental setup We perform the experiments using the Huggingface framework [12]. In each experiment we train with the Adam optimizer [3] with parameters: learning rate = 1e-5, $\beta_1 = 0.9$, $\beta_2 = 0.999$, ϵ =1e−8, weight_decay = 0. When we fine-tune a pre-trained model, we must use a tokenizer that comes along with the model – in cases of both GPT2 and T5 these are pre-trained byte pair encoding tokenizers. When training from scratch we use a simple tokenizer splitting on whitespaces. All trainings were performed using GeForce GTX 2080 Ti GPUs. We limit all the trainings to passing through a model 64 million training examples.[2] All data and scripts required to reproduce the results presented here are available at `https://github.com/BartoszPiotrowski/transformers-for-mathematics`

Results and conclusions Figure 1 shows training curves for one of the datasets – linear equations. Table 1 shows the final testing accuracy for all the tasks. There are two conclusions:

1. In almost all cases, the pre-trained versions of models performed worse than the models trained from scratch. It likely means that the data on which the models were pre-trained does not contain much information relevant for dealing with mathematical problems. There are, however, two exceptions: for T5 and datasets on differentiation and solving linear equations. Especially for the latter the difference is much in favour of the pre-trained version of the model. As for now, we do not have explanation for this.
2. GPT2 performed better than T5 for all the datasets. It means that decoder-only architectures are capable of learning mathematical tasks, despite the fact that in most of the cited related works encoder-decoder architectures were used. However, it is unclear whether the superior performance of GPT2 was due to the different architecture, or possibly because of larger number of trainable parameters. Further experiments would be needed.

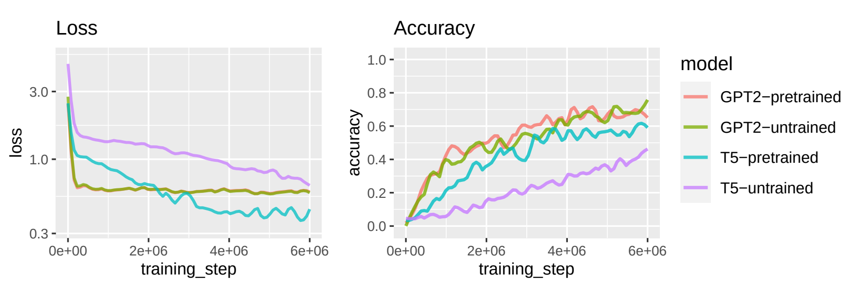

Figure 1: Training loss and accuracy on the linear equations dataset.

[2]This is a practical limit – full training takes then, depending on a dataset, between 4 and 50 hours.

References

[1] François Charton, Amaury Hayat, and Guillaume Lample. Learning advanced mathematical computations from examples. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[2] Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. Can neural networks understand logical entailment? In *International Conference on Learning Representations*, 2018.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[4] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[5] Bartosz Piotrowski and Josef Urban. Stateful premise selection by recurrent neural networks. In Elvira Albert and Laura Kovács, editors, *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, May 22-27, 2020*, volume 73 of *EPiC Series in Computing*, pages 409–422. EasyChair, 2020.

[6] Bartosz Piotrowski, Josef Urban, Chad E. Brown, and Cezary Kaliszyk. Can neural networks learn symbolic rewriting? *CoRR*, abs/1911.04873, 2019.

[7] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[9] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[11] Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. First experiments with neural translation of informal to formal mathematics. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *11th International Conference on Intelligent Computer Mathematics (CICM 2018)*, volume 11006 of *LNCS*, pages 255–270. Springer, 2018.

[12] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.