ANALYSING MATHEMATICAL REASONING ABILITIES OF MEMORY AUGMENTED NEURAL NETWORKS

DEEP LEARNING PROJECT PROPOSAL

Laurent Cavazzana
lcavazzana@student.ethz.ch

Alexis Stockinger
salexis@student.ethz.ch

Bernhard Walser
walserb@student.ethz.ch

December 9, 2019

DESCRIPTION

Mathematical reasoning is difficult even for humans. Mathematical concepts have to be remembered and chained together to solve different problems. In the Paper from Saxton et. al. [1], they analyse mathematical reasoning abilities of neural models based on (attentional) recurrent neural network and transformer architecture. As opposed to previous work, they cover a broad area of mathematics instead of narrowing to one specific subject, testing the general reasoning skills of the network. They obtain moderate performance overall, in particular areas requiring several intermediate calculations remain largely unsolved. Memory augmented networks could be well suited for this task as they might benefit from the ability to store intermediate values for later usage. We thus would like to extend this paper by analysing the performance of memory augmented neural networks on mathematical reasoning.

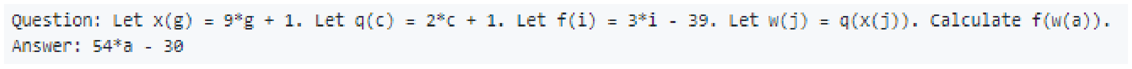
1 Method / Procedure

First, we need to acquire a good understanding of memory augmented neural networks and the different architectures used in the paper through the provided literature. We can then start implementing a model suited to our problem, using some available implementations[1][2]. We then train, test and optimize this model on the dataset and investigate the performance of it. When we can not find any more improvements to the model we compare the performance to the architectures given in the paper [1]. This concludes our project.

2 Dataset

The dataset contains mathematical problems based on a national school mathematics curriculum (up to age 16). The dataset was synthetically generated such that linguistic variation or complexity is not included. The problems are given as a Question and the corresponding Answer in string format. When predicting the Answer, it has to match the given Answer character-for-character to be regarded as correct (see figure 1). The dataset comes with two sets of tests: interpolation tests include question types occurring in the training set; and extrapolation tests which measure generalization to problems which difficulty goes beyond that seen during training. The dataset is provided by DeepMind in a GitHub repository[3], either as pre-generated dataset or as generator to have the possibility to generate more data if needed. For the generator they also make sure that data which is generated has a low probability to be generated twice, which is essential for the usability of the test data later.

[1]https://github.com/snowkylin/ntm
[2]https://github.com/MarkPKCollier/NeuralTuringMachine
[3]https://github.com/deepmind/mathematics_dataset

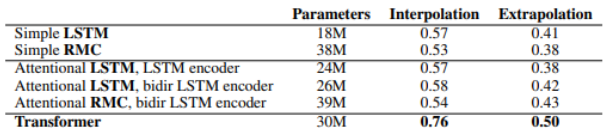


```
Question: Let x(g) = 9*g + 1. Let q(c) = 2*c + 1. Let f(i) = 3*i - 39. Let w(j) = q(x(j)). Calculate f(w(a)).
Answer: 54*a - 30
```

Figure 1: Question Format

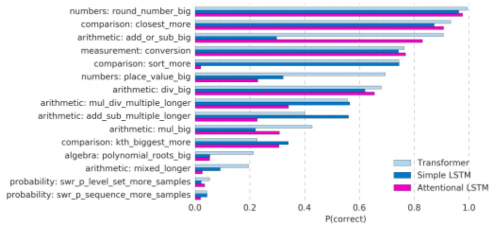
3 Baselines

There are different baselines involved: the baseline for the overall performance is set by an attentional/transformer architecture [2] (see Figure 2a). In the second baseline, we can analyse the performances across the different mathematical sub problems (see Figure2b). We will compare our model performance on interpolation and extrapolation tests to both the best performing transformer approach and the simple LTSM recurrent neural network architecture.



	Parameters	Interpolation	Extrapolation
Simple **LSTM**	18M	0.57	0.41
Simple **RMC**	38M	0.53	0.38
Attentional **LSTM**, LSTM encoder	24M	0.57	0.38
Attentional **LSTM**, bidir LSTM encoder	26M	0.58	0.42
Attentional **RMC**, bidir LSTM encoder	39M	0.54	0.43
Transformer	30M	**0.76**	**0.50**

(a) Overall Baseline



(b) Baseline Subproblems (Extrapolation)

Figure 2: Baselines

4 Literature

We have several papers to study for our project. First, the initial 'Neural Turing Machines' paper [3] and also extensions of it [4]. We also investigate the approach they tried and failed to use [5] in the reference paper [1]. Then, a paper about implementing Neural Turing Machines [6]. To understand the previous architectures used in [1] we have to gain good understanding of LSTM [7] and a extended version they used referred as 'Attentional LSTM' [8] and of course their best performing approach, an attentional/transformer architecture [2].

References

[1] Felix Hill Pushmeet Kohli David Saxton, Edward Grefenstette. Analysing mathematical reasoning abilities of neural models. *arXiv:1904.01557*, 2019.

[2] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N.Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. *arXiv:1706.03762*, 2017.

[3] Ivo Danihelka Alex Graves, Greg Wayne. Neural turing machines. *arXiv:1410.5401*, 2014.

[4] Matthew Botvinick Daan Wierstra Timothy Lillicrap Adam Santoro, Sergey Bartunov. One-shot learning with memory-augmented neural networks. *arXiv:1605.06065*, 2016.

[5] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Badia, Karl Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538, 10 2016.

[6] Joeran Beel Mark Collier. Implementing neural turing machines. *arXiv:1807.08518*, 2018.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[8] Yoshua Bengio Dzmitry Bahdanau, KyungHyun Cho. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2016.