Deep Learning Report

Nicolò Dentale (1868948), Stefano Iervese (2057309), Federico Raschiatore (2071132)

October 12, 2023

1 Introduction

This report aims to present the procedure and rationale employed in our project. Our primary objective was to develop a model capable of answering a wide range of mathematical and logical questions using the mathematical dataset referenced in the paper [3]. This achievement has been possible thanks to our model's proficiency in inferring, learning, and applying mathematical laws, axioms, and symbol manipulation rules.

We implemented various methods to compare different techniques and assess their performance. Initially, we constructed a Transformer model and utilized it as a baseline to create the TP-Transformer mentioned in the paper. Additionally, we opted to incorporate a distinct model, an LSTM, to conduct further evaluations.

2 Implementation

2.1 Implementation choices

In this section we will expose the specific implementation choices we have made during the project's development.

2.1.1 Pytorch lightning

As specified in the guidelines we used PyTorch Lightning. This framework provided us with a structured environment for training and testing our models, along with essential PyTorch functionalities.

2.1.2 Google Colab

All training and testing activities were conducted using the free version of Google Colab, leveraging the available GPU resources from the free tier. However, due to limited computational power, the training process was significantly slowed down.

2.2 Dataset

Our training and testing dataset originates from the mathematical dataset featured in the paper. This dataset comprises 56 files, each focusing on various mathematical areas, including algebra, probability, and more. It is categorized into different directories: train easy, train medium, train hard, interpolation, and extrapolation.

Given the computational constraints of the free version of Google Colab, we decided to create a scaled-down Dataset by selecting two files from the train easy directory for training (arithmetic add or sub.txt, arithmetic mul.txt) and we also tested on this one. For this reason we splited the Dataset in train and test (respectively 2/3 and 1/3 of the original Dataset). .

2.3 Metrics

As specified in the requirements, we employed accuracy as the primary evaluation metric. Our accuracy metric closely resembled the one outlined in the paper [3]. A prediction was considered correct only when it precisely matched the corresponding elements in the label data.

2.4 Dictionary and Tokenization

We chose to build our dictionary using the words and symbols present in the training dataset. This dictionary assigned a unique number to each word or symbol, facilitating tokenization. Each word or symbol was mapped to its respective number in the dictionary (e.g., "what" → 11, "is" → 12, "the" → 13, and so on). We also introduced a special symbol to denote the end ("@") of labels and with "&" serving as the padding symbol.

3 Models

3.1 LSTM

Our first model choice was an LSTM, as described in the paper [1]. This straightforward implementation consisted of a single LSTM cell. The input vectors were initialized with zero hidden state and cell state vectors, which were updated by the LSTM cell after each iteration.

To enhance performance, we introduced a teacher forcing mechanism during training with a 50% probability. This mechanism supplied either the correct value or the last predicted one to the LSTM cell during the training step, allowing us to control the model's behavior and mitigate potential training slowdowns.

We trained the model for 2 epoch, resulting in an accuracy of approximately 60%.

We used an Adam optimizer with a learning rate of 6e-4, and the cross-entropy function to compute the loss.

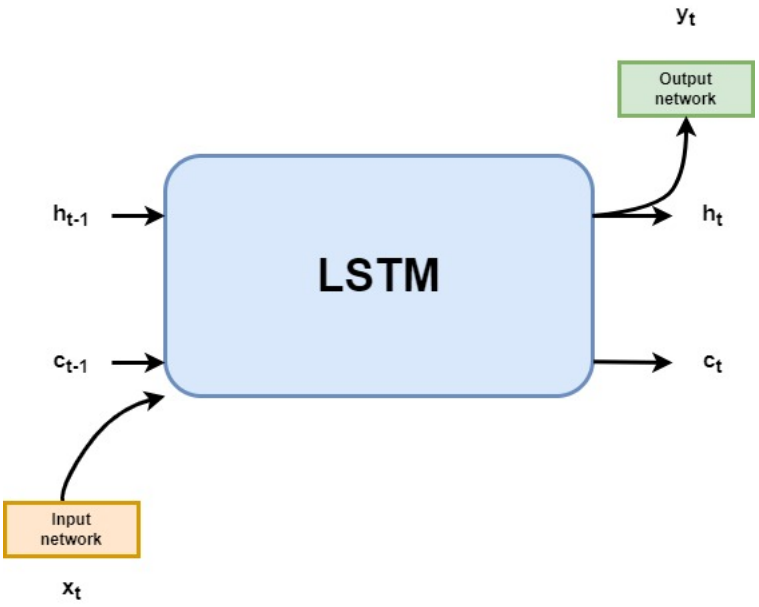

Figure 1: LSTM schema

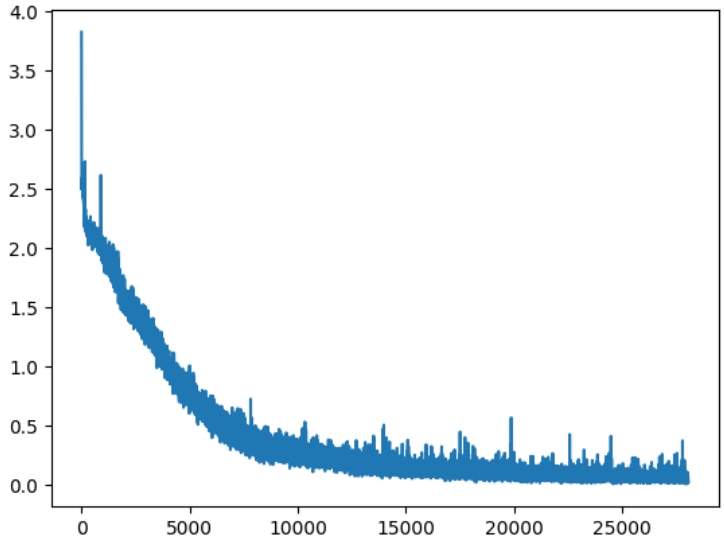

Figure 2: LSTM's accuracy graph in 2 epochs

3.2 Transformer

Our second model choice was a Transformer model, inspired by the paper "Attention Is All You Need"[2]. This model comprised a multi-head attention encoder and decoder, featuring 8 attention heads and an embedding dimension of 512. We utilized 6 encoders and 6 decoders as in the paper [2]. Additionally, we applied a dropout of 0.1 to avoid overfitting. We incorporated masking in the decoder layer to focus on past tokens and implemented a teacher forcing mechanism with a 90% probability to enhance model performance. The attention heads were generated by the same neural network. This approach optimized resource utilization, enabling the model to operate more efficiently in terms of computational power because need less GPU's capacity.

We trained the model for 3 epoch, resulting in an accuracy of approximately 90%.

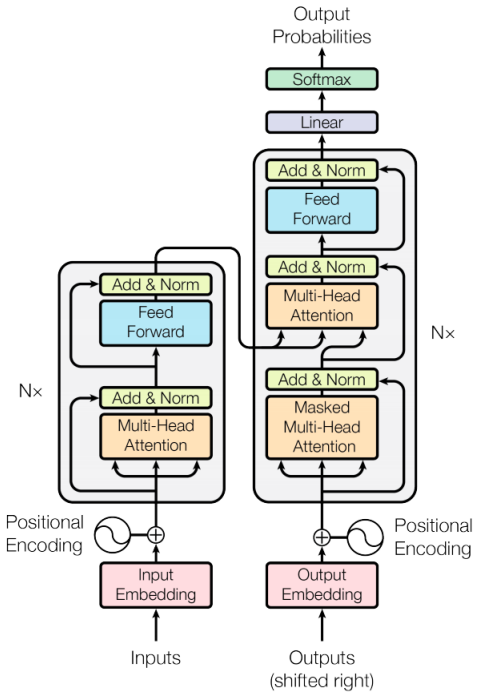

Figure 3: Transformer schema from "Attention is all you need"

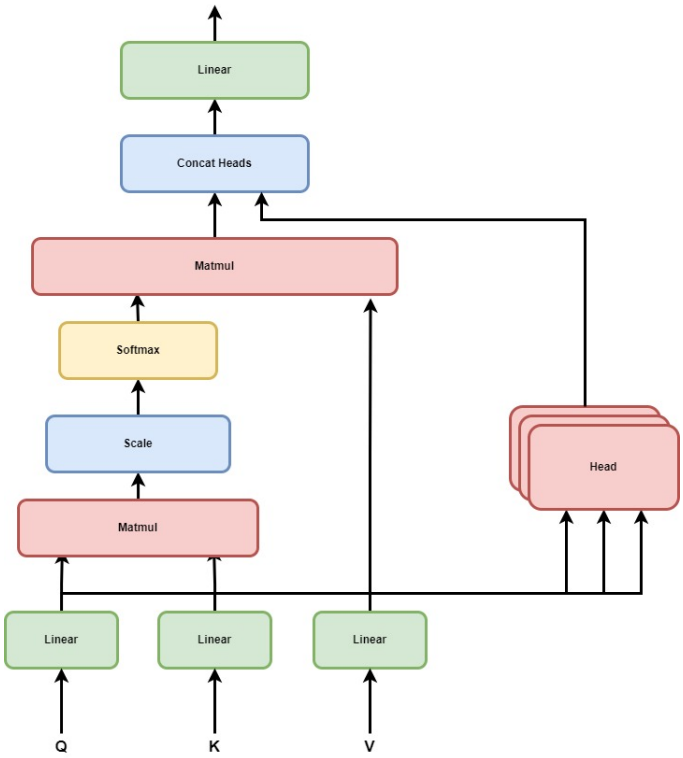

Figure 4: Multi-head attention schema

We used an Adam optimizer with a learning rate of 1e-4, and the cross-entropy function to compute the loss.

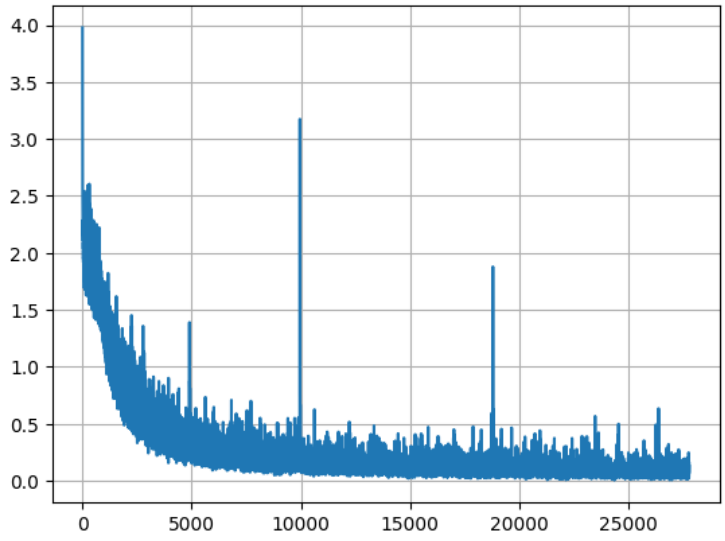

Figure 5: Transformer's accuracy in 2 epochs

3.3 SOTA: TP-Transformer

Lastly, we implemented the state-of-the-art model for the Mathematic Dataset, as detailed in the article "Enhancing the Transformer With Explicit Relational Encoding for Math Problem Solving" [3]. This model closely resembles the previously mentioned Transformer, with a notable difference: the product of the standard head layer was labeled as "Filler" and multiplied by another vector denoted

as "Role." The resulting product of "Filler" and "Role" was summed with other heads, following a similar structure. This model, known as the Tensor Product Transformer, introduced explicit relational encoding to improve performances.

We trained the model for 3 epoch, resulting in an accuracy of approximately 95%.

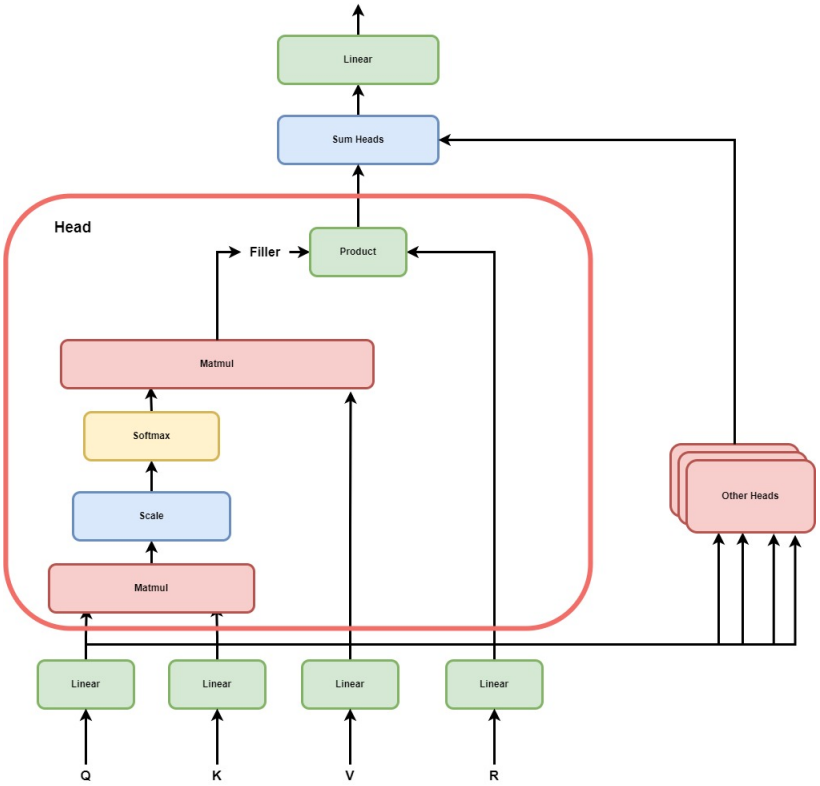

Figure 6: TP-Transformer multi-head attention schema

We used an Adam optimizer with a learning rate of 1e-4,and the cross-entropy function to compute the loss.

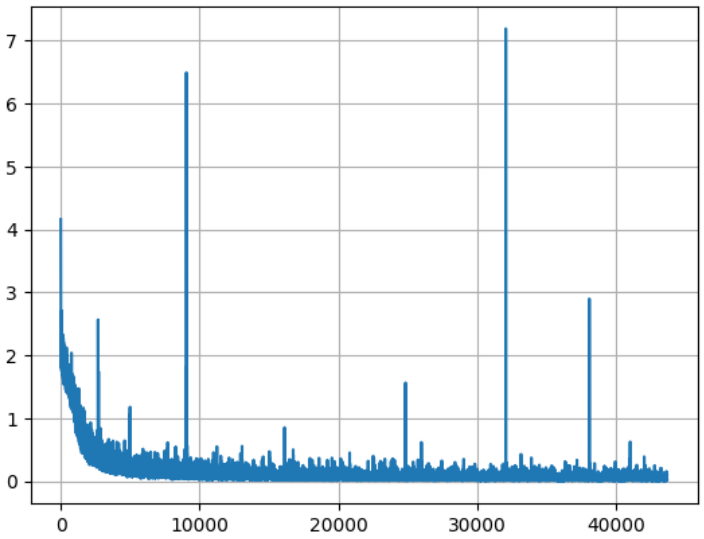

Figure 7: TP-Transformer's accuracy in 3 epochs

3.4 Evaluation

	Accuracy
LSTM	60%
Transformer	90%
TP Transformer	95%

Table 1: Accuracy on the models

3.5 Conclusion

We developed different models as LSTM, Transformer and the state-of-the-art TP Transformer to handle the Mathematic Dataset from paper [3]. We can assert that we are able to solve mathematical problems. TP-Transformer has emerged as the most accurate choice. However, there is room for further improvement and exploration in terms of dataset size, model complexity, and training strategies.

References

[1] "Analysing mathematical reasoning abilities of neural models" ICLR 2019
url: https://arxiv.org/pdf/1904.01557.pdf
[2] "Attention Is All You Need" url:https://arxiv.org/pdf/1706.03762.pdf
[3]"Enhancing the Transformer With Explicit Relational Encoding for Math Problem Solving" url:https://arxiv.org/pdf/1910.06611v2.pdf