Report Deep Learning

Andrea Celestini, Lorenzo De Santis, Simone Di Rienzo

February 2023

Dataset

The dataset that has been used to train and test the models proposed is the **Mathematics Dataset**, which is a large collection of question and answer pairs, from a range of question types at roughly school-level difficulty. The main goal of this dataset is to investigate the capability of neural networks to reason formally. To achieve this goal, three main modules were chosen from the ones provided: algebra, calculus and probability.
Each module is composed by **2 million** (question, answer) pairs, in which the questions are limited to 160 characters in length, and answers to 30 characters in length. Note the training data for each question type is split into ”train-easy”, ”train-medium”, and ”train-hard”, but in this project they were mixed together uniformly; for the test set we have used the interpolate directory containing 10000 pairs for each module.
The dataset has been pre-processed in order to provide an encoding that is feasible with the networks we are working with.
We have extracted a character-level *vocabulary* of 58 symbols (including special symbols *start-of-sentence*, *end-of-sentence* and *padding*). Each character will correspond to a number and it will be needed to change the representation of each sentence of the dataset, adding also the special symbols; *padding* corresponds to 0, while *start-of-sentence* and *end-of-sentence* correspond respectively to 1 and 2.
The SOTA model was trained and tested on the 3 modules we have chosen (6 million of pairs for training set, 30000 pairs for test set), while the other two models were trained and tested only on the algebra module due to computational and time limits. For the same reason we have performed few epochs (2 for SOTA and first baseline, 3 for the additional baseline) and we have decided to not split the dataset into train and validation set, since we have no reason to perform early stopping (we have trained for few epochs). Besides the validation phase required

a lot of time because of no teacher forcing method can be used.
So at the end of each epoch we have computed the accuracy directly on the test set.

Baseline: Transformer

To implement our first baseline we have been inspired by the paper "Attention is all you need" and we have implemented a transformer, composed of six encoder blocks and six decoder ones. First we have embedded the input using an embedding layer with parameters that can be learned, then we add positional encoding to input embedding applying a sinusoidal embedding and we use the final result as input for the encoder part.
Each encoder block is composed of two layers, a multi head attention layer and a feed forward one , interleaved by layer normalization and residual connection.
The multi head attention layer is composed of 8 self attention layers whose outputs are concatenated and then used as input for a final linear layer.
Each self attention layer is composed of three linear layers that are used to create query, key and value matrices. We computed a scaled dot-product attention between queries and keys, then we apply softmax and we multiply by values. We have also applied a padding mask to the pre-softmax input setting to $-\infty$ the entries corresponding to pad that we were not interested in (in this way the output after softmax is 0).
The decoder takes as input the embedded target (with positional embedding) and the output of the encoder. Each decoder block is composed of three layers: a masked multi head attention in which attention of each character is computed only with previous characters setting to $-\infty$ the entries of pre-softmax input we were not interested in ; an encoder-decoder multi head attention that takes queries from the previous layer and keys and values from the last encoder block; a feed forward neural network with two linear layers and a ReLU; layers are interleaved with normalization and residual connection.
We have applied a final linear transformation to the output of the last decoder block using the same weight matrix we have used to embed the target; in this way the final output has the shape: batch size * sequence length * vocabulary size.
During the training phase we have used the teacher forcing: instead of using the previous output character to generate the following, we have used the correct target character, computing the loss between the expected correct character and the output of the model.
During inference we have not used teacher forcing. We have added also some regularization layers using dropout, but we have realized that accuracy decreased, so we have removed it

setting the probability to drop to 0.

SOTA: TP-Transformer

We have implemented the SOTA model following the paper "Enhancing the Transformer With Explicit Relational Encoding for Math Problem Solving".
With respect to the first baseline, we have changed self attention and multi head attention layers.
In self attention layer we have added also the role matrix and a final linear layer: we multiply the attention score matrix element wise by the role matrix and then we apply the linear layer.
In multi head attention instead of concatenate the output of the self attention layers, we sum directly since we have already applied the linear layer.
We have added a normalization layer and a residual collection as first operation to each encoder and decoder block.
We have built the TP-Transformer version C that uses the same dimension both for embedding layer and the hidden layer of the feed forward network (512).
We have used teacher forcing during training phase but not during inference.

Additional baseline: LSTM

We have implemented a recurrent neural network as additional baseline. It is composed of two Long Short Term Memory networks, one to encode the question and one to decode the answer.
First we have embedded question and answer, then we have used the question as input for the first LSTM and we have used the final hidden state and cell state as input for the second LSTM that takes as input one character at time.
The output of the LSTM is transformed using a linear layer to reshape it to batch size * vocabulary size shape.
We have used a probabilistic teacher forcing: with probability p we used the correct target character as input for the LSTM, otherwise we used the previous predicted character.
During inference we did not use teacher forcing.

Results

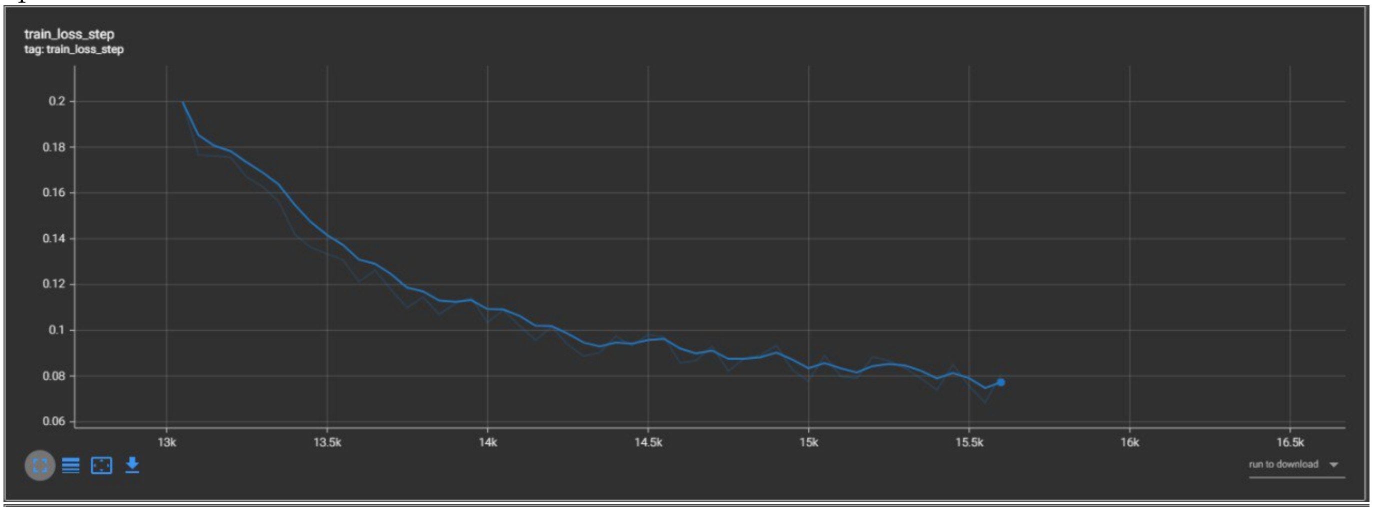
As we said before, we have not split the dataset into train and validation set, but we have used the entire module to train because we have trained for few epochs due to computational and time limits. So we have computed the accuracy at the end of each epoch on the test set (separated from the train set) and we have decided to stop after the second epoch for the two transformers and after the third for the recurrent neural network.

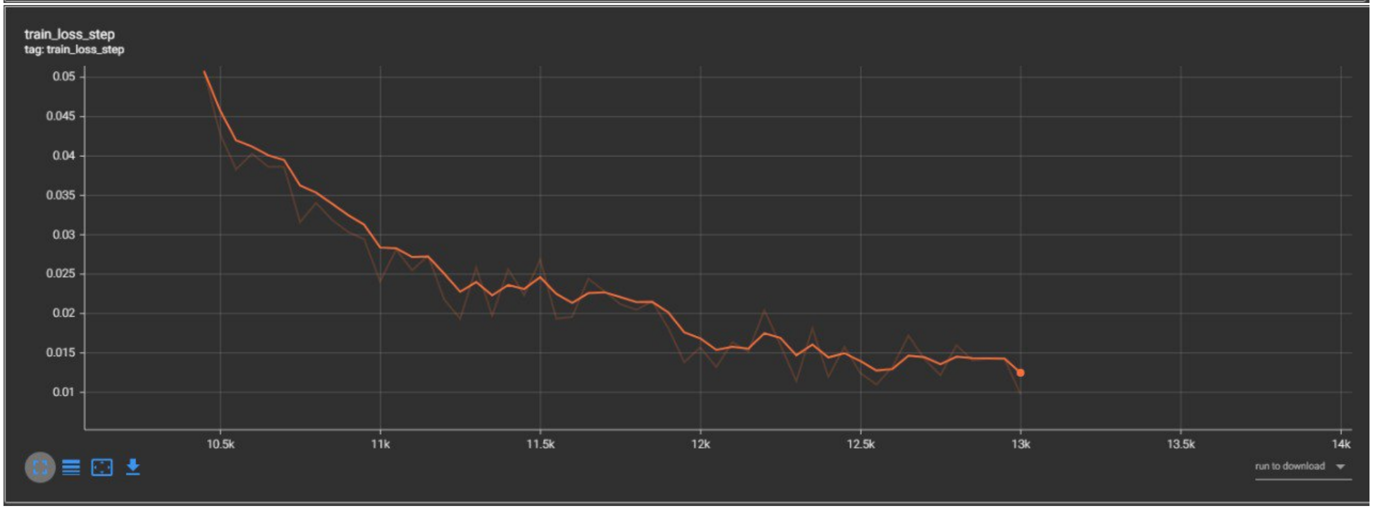
We report train loss and accuracy for each model we have trained and tested.

Baseline: Transformer

We have trained the first baseline on a single module, algebra, for two epochs.

We report the train loss starting from some steps after the first epoch and during the second epoch:



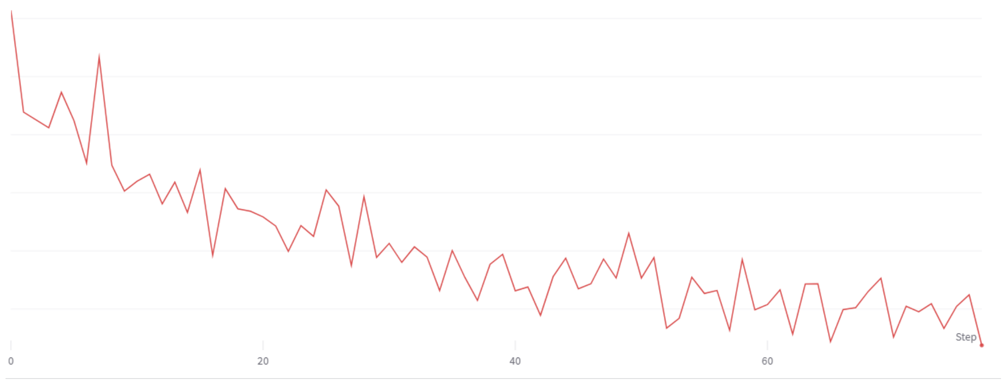


The accuracy at the end of the second epoch was about 64%

SOTA: TP-Transformer

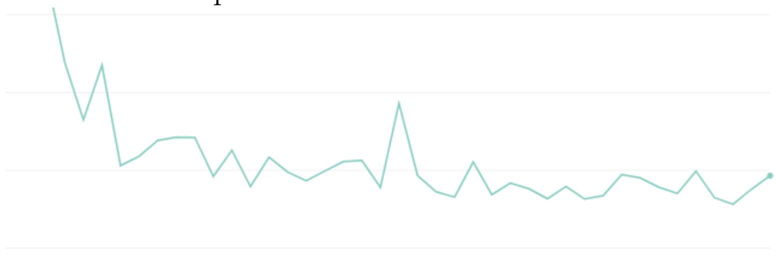
We have trained the SOTA model on 3 modules for two epochs.

The loss in the first epoch started at 1.37 and decreased until 0.12:



The accuracy at the end of the first epoch was around 55%

In the second epoch the loss decreased until 0.06:



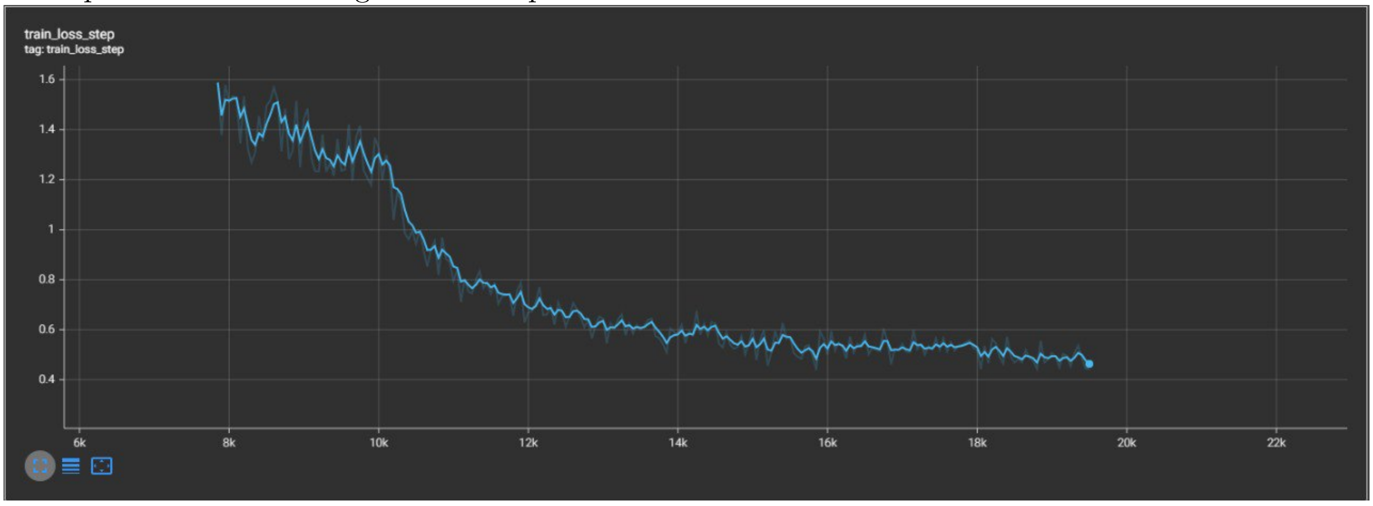
The accuracy at the end of the second epoch was around 67%

So it was a bit better than the baseline transformer but we have to consider that we have tested it on three different modules instead of testing only on algebra, accuracy increases when we consider a single module.

Additional Baseline: LSTM

We have trained the second baseline on a single module, algebra, for three epochs.

We report the train during the three epochs:



The accuracy at the end of the third epoch was about 48%