NLP Challenge: Polynomial Expansion

Adithya Sampath
Carnegie Mellon University
Pittsburgh, PA 15213
sampath.adithya96@gmail.com

1 Introduction

For the Polynomial Expansion challenge I went with the Transformer encoder-decoder model.

Transformers [1] were introduced in the context of machine translation with the purpose to avoid recursion in order to allow parallel computation (to reduce training time) and also to reduce drops in performance due to long dependencies. The main characteristics are:

1. Non sequential: sequences are processed as a whole rather than sequentially.
2. Self Attention: used to compute similarity scores between tokens in a sequence to gain additional context. Due to this, transformer models are more contextualized, and have a more holistic understanding of the sequence.
3. Positional embeddings: introduced to replace recurrence. The idea is to use fixed or learned weights to encode information related to the position of a token in a sequence.

The non-sequential nature is the main reason why transformer do not suffer from long dependency issues. The original transformers do not rely on past hidden states to capture dependencies with previous tokens. They instead process a sequences as a whole. That is why there is no risk to lose (or "forget") past information. Moreover, multi-head attention and positional embeddings both provide information about the relationship between different tokens. Due, to thse reasons I went with the Transformer encoder-decoder model.

I chose Transformer models over Recurrent neural networks (RNN), and Long-short term memory models (LSTM), since they have the following core properties:

1. Sequential processing: sequences must be processed token by token. This is the reason why RNN and LSTM can't be trained in parallel
2. Past information retained through past hidden states: sequence to sequence models follow the Markov property: each state is assumed to be dependent only on the previously seen state. Information in RNN and LSTM are retained using the context computed from hidden states. The encoding of a specific word is retained only for the next time step, which means that the encoding of a word strongly affects only the representation of the next word, so its influence is quickly lost after a few time steps. LSTM (and also GRU) can boost a bit the dependency range they can learn thanks to a deeper processing of the hidden states through specific units (which comes with an increased number of parameters to train) but nevertheless the problem is inherently related to recursion.

Empirically, as shown in the results section, I found that using 3 encoder-decoder layers, 10 attention heads, and hidden dim of 256 gives the best results. The number of Trainable model parameters is **4.032548M**. For all experiments I used the Adam optimizer, and Cross Entropy as the Loss function, along with the following hyperparameters:

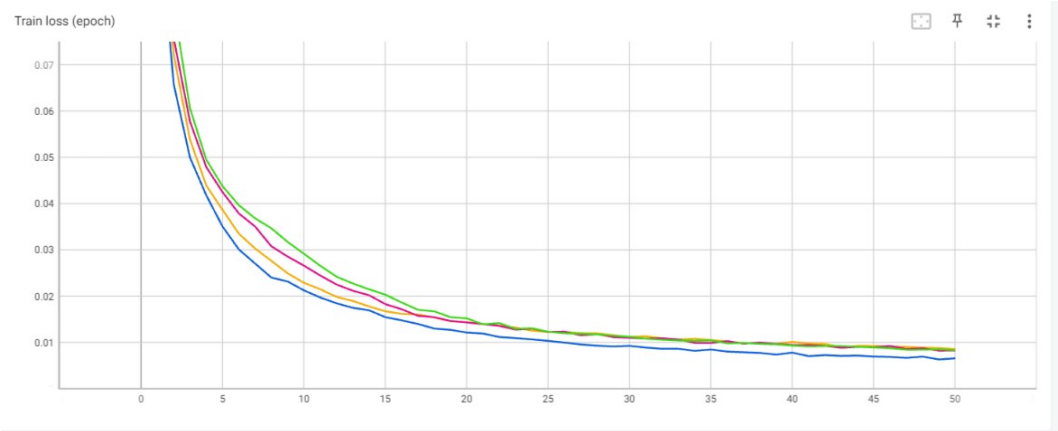
Hyperparameter	Value
Learning Rate	0.0005
Batch Size	128
Seed	1234
Num Workers	8
Max Input Sequence Size	29

2 Results

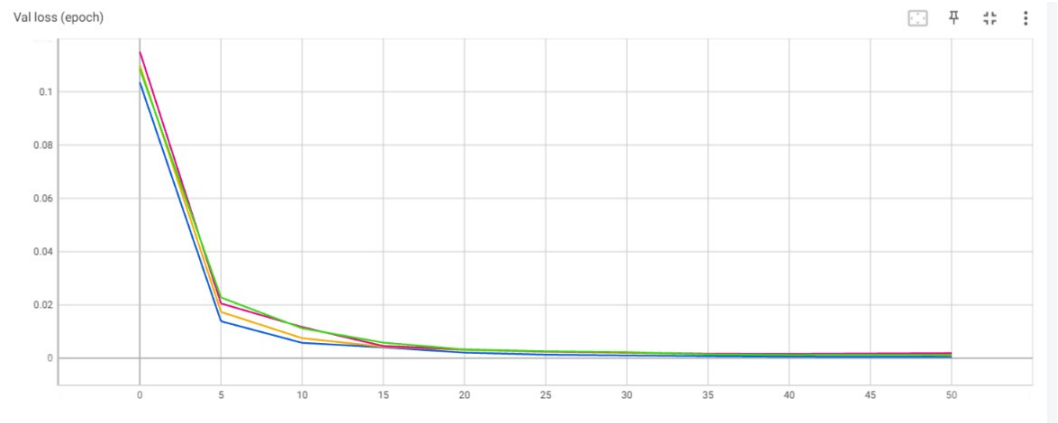
2.1 Experiments with different hidden dims

For this ablation study, I kept the number of encoder & decoder layers of the transformer model as 3, and the number of heads for each encoder & decoder layers as 8 for all experiments. Empirically, I found that using hidden dim of 256 and encoder & decoder hidden dim of 256 to give the best results.

Hidden dim	Enc & Dec Hidden dim	Test Accuracy	Validation Accuracy
256	**256**	**0.99762**	**0.99786**
512	512	0.99495	0.99478
256	512	0.99262	0.9925
512	256	0.9959	0.99566



(a) Train Loss



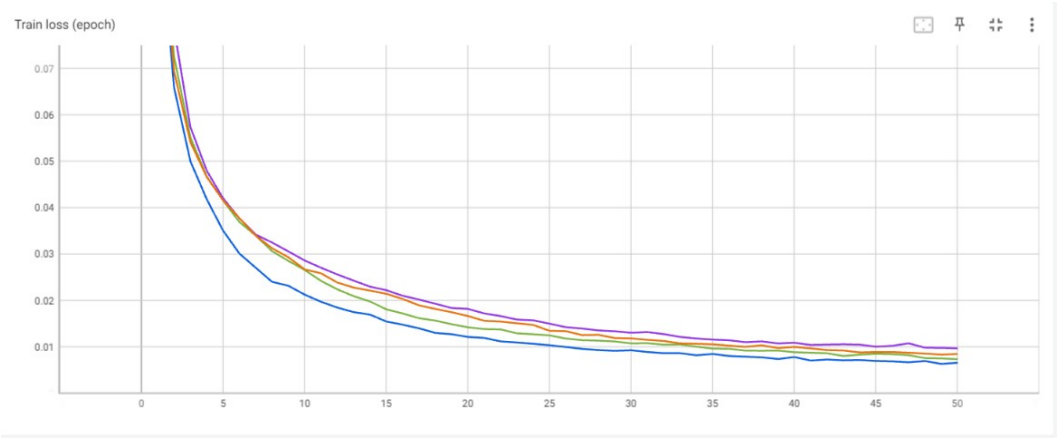
(b) Validation Loss

Figure 1: Loss plots for Transformer encoder-decoder model different hidden dims

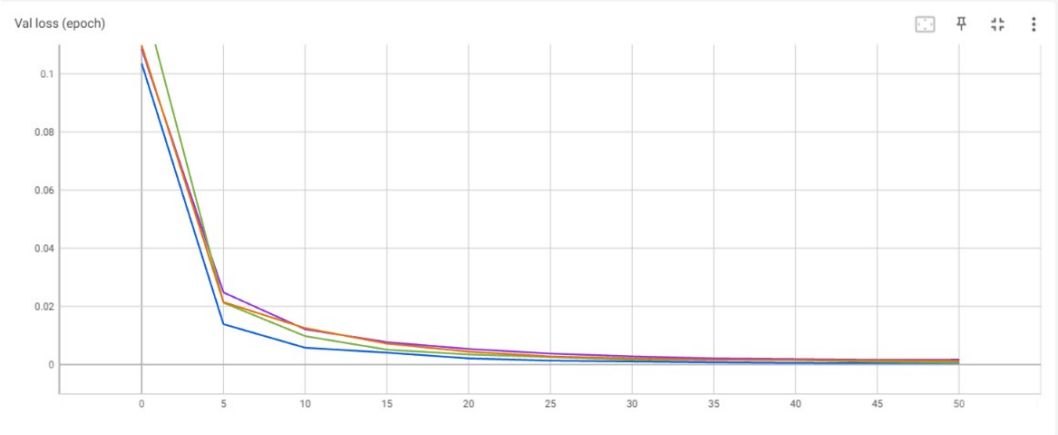
2.2 Experiments with different encoder decoder layers

For this ablation study, I kept the hidden dim as 256, and the number of heads for each encoder & decoder layers as 8 for all experiments. Empirically, I found that using 3 layers for encoder & decoder gave the best results. We can observe below that as we increase number of layers from 1 to 3, we see improvement in performance, however, increasing beyond 3 layes (i.e. 4 and above) shows a drop in performance.

Num Layers	Test Accuracy	Validation Accuracy
1	0.99333	0.99272
2	0.99688	0.99688
3	**0.99762**	**0.99786**
4	0.99461	0.99457



(a) Train loss



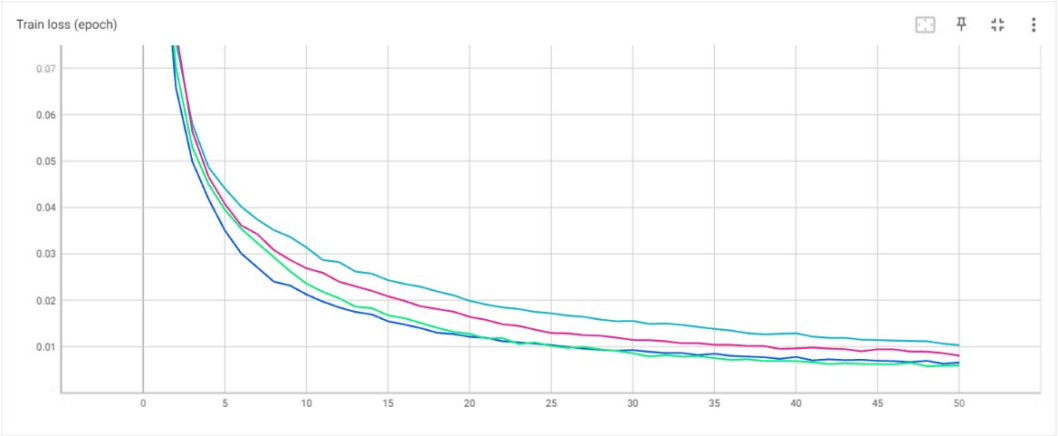
(b) Validation Loss

Figure 2: Loss plots for Transformer model different number of encoder & decoder layers

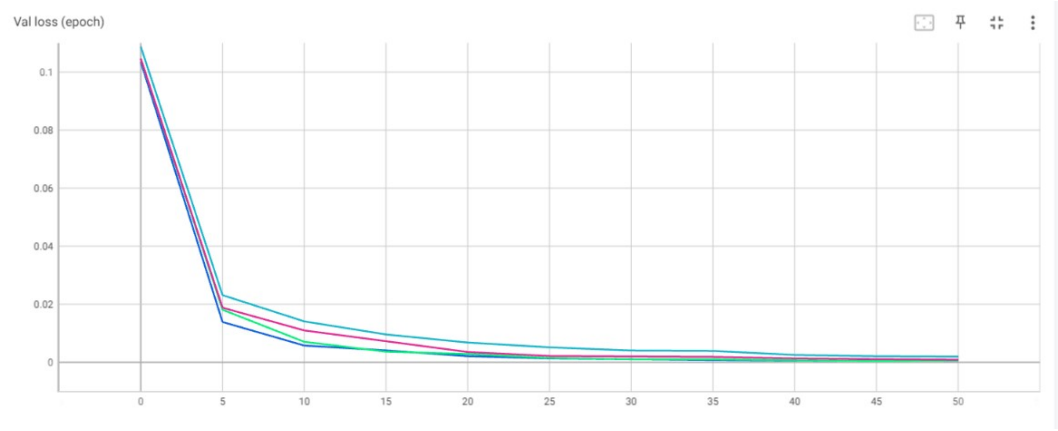
2.3 Experiments with different number of heads for encoder decoder layers

For this ablation study, I kept the hidden dim as 256, and the number of layers for each encoder & decoder layers as 3 for all experiments. Empirically, I found that using 10 heads for encoder & decoder layer gave the best results. We can observe from the results below that there's an improvement in performance as we increase the number of attention from 4 to 10.

Num heads	Test Accuracy	Validation Accuracy
4	0.99117	0.99074
6	0.99662	0.99663
8	0.99762	0.99786
10	**0.99841**	**0.99823**



(a) Train Loss



(b) Validation Loss

Figure 3: Loss plots for Transformer model different number of attention heads for encoder & decoder

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.