0.1 Introduction

From Table 0.5.1 and Table 0.5.1, it is clear that the Transformer models are memory-intensive due to their quadratic complexity in terms of the dataset size. This is a significant bottleneck for scaling up the Transformer models to larger datasets with limited compute resources. To address this issue, several methods have been proposed to reduce the memory complexity of the Transformer models which we will explore in this chapter.

0.2 Pseudotokens

Pseudotokens (also known as Inducing Points) are a set of tokens that are used to approximate the full set of tokens, it can be thought of as a lower dimensional representation of the data set. They have been widely used in the context of Gaussian Processes (GPs) to reduce the complexity of the model with great success [Hensman, Fusi, and Lawrence 2013]. The original tokens $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ are projected onto into a lower-dimensional space $\boldsymbol{I} \in \mathbb{R}^{M \times D}$ where $M << N$ through some translation equivariant network [Ashman et al. 2024] giving us the pseudotokens $\boldsymbol{I}$ which are translation equivariant to the original tokens $\boldsymbol{X}$. We then perform cross-attention between the pseudotokens $\boldsymbol{I}$ and the original tokens $\boldsymbol{X}$, hence reducing the memory complexity to $\mathcal{O}(MN_c + MN_t)$.

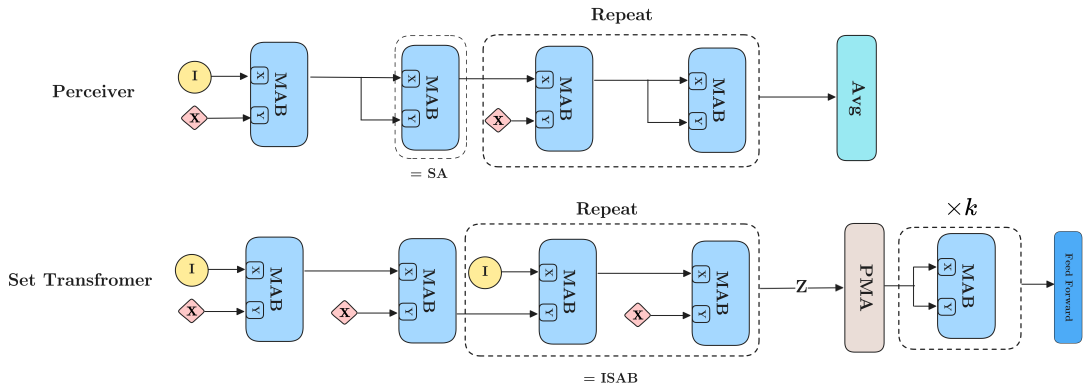

Figure 0.2.1: Perceiver vs Set Transformer. MAB is equivalent to Multi-Head Cross Attention, refer to [Jaegle et al. 2021; Lee et al. 2019] for more details.

We consider two implementations of a pseudotokens based Transformer, one is the Set Transformer [Lee et al. 2019] and the other is the Perceiver [Jaegle et al. 2021]. Both models are very similar and differ in the way they implement the cross-attention mechanism between the original and pseudo tokens Figure 0.2.1. Feng et al. 2023 implemented the Perciever model in the context of NPs creating the 'Latent Bottled Attention Neural Process' (LBANP) model. Ashman et al. 2024 implemented the Set Transformer model into a NP creating the 'Inducing Set Transformer' (IST) model. Ashman et al. 2024 found both models to perform very similarly. Both models will be encompassed under the 'PT-TETNP' term in the results' section since they are very similar in terms of performance.

0.3 Linear Transformer

Katharopoulos et al. 2020 introduces a kernelized form of the attention mechanism that allows the model to be linear with the number of tokens. The output of an attention head $\boldsymbol{H}$ is computed as follows:

$$\boldsymbol{H}_i = \frac{\phi(\boldsymbol{Q}_i)^T \sum_{j=1}^{N} \phi(\boldsymbol{K}_j)\boldsymbol{V}_j^T}{\phi(\boldsymbol{Q}_i)^T \sum_{j=1}^{N} \phi(\boldsymbol{K}_j)} \tag{0.3.1}$$

Where ϕ is a function that introduces non-linearities, the authors use the ELU function [Clevert, Unterthiner, and Hochreiter 2016]. We only compute $\sum_{j=1}^{N} \phi(\boldsymbol{K}_j)$ and $\sum_{j=1}^{N} \phi(\boldsymbol{K}_j)\boldsymbol{V}_j^T$ once for all the queries $\boldsymbol{Q}_i$, hence reducing the complexity to $\mathcal{O}(N)$. This can simply replace the transformer in the original TNP model, giving us the 'Linear Transformer NP' (LinearTNP).

0.4 HyperMixer

Is attention required for a Transformer model to be effective? The majority of parameters in Transformers are in the MLPs and not the attention mechanism. Tolstikhin et al. 2021 proposes the 'MLP-Mixer' - a Transformer model that replaces the attention mechanism with MLPs across rows and columns of the input. It can be viewed as 'mixing' the features across tokens, thus learning patterns in a way akin to attention. However, the MLP-Mixer requires a **known and fixed input size** which breaks the flexibility of the model if we apply it to a NP.

To overcome this limitation in flexibility, the 'HyperMixer' model [Mai et al. 2023] was proposed which is a variant of the MLP-Mixer that is designed to work with variable input sizes. The model uses hypernetworks [Ha, Dai, and Le 2016] to generate the weights of the MLPs. The hypernetworks $h_k, h_q : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times p}$ are applied on the queries $\boldsymbol{Q}$ and keys $\boldsymbol{K}$ of the input (row wise) to generate the weights:

$$\boldsymbol{W_k} = h_k(\boldsymbol{K}) = \begin{bmatrix} \text{MLP}_k(\boldsymbol{k}_k) \\ \vdots \\ \text{MLP}_k(\boldsymbol{k}_N) \end{bmatrix} \quad \boldsymbol{W_q} = h_q(\boldsymbol{Q}) = \begin{bmatrix} \text{MLP}_q(\boldsymbol{q}_k) \\ \vdots \\ \text{MLP}_q(\boldsymbol{q}_N) \end{bmatrix} \tag{0.4.1}$$

where $\text{MLP}_k, \text{MLP}_q : \mathbb{R}^d \to \mathbb{R}^p$ transforms the dimension of the input to a lower dimension p. The output of the model is computed as follows:

$$\boldsymbol{H} = \boldsymbol{W_k}\sigma(\boldsymbol{W_q}^T\boldsymbol{V}) \tag{0.4.2}$$

With σ being an activation function, the authors use the GELU function [Hendrycks and Gimpel 2023]. Cross attention is performed by generating the queries from one input and the keys and values from another input. We simply replace attention in the original TNP model with the HyperMixer giving us the 'HyperMixNP' model.

Lack of Translation Equivariance

Both the LinearTNP and HyperMixNP models are not translation equivariant. To introduce TE without using pseudotokens, we require the use of the pairwise differences matrix $\boldsymbol{\Delta}$ (Equation 0.2.3), which would increase the complexity of the model to $\mathcal{O}(N^2)$. It is possible to use pseudotokens in conjunction with these model to achieve translation equivariance however this was not explored in this work.

0.5 Experimental Results

Model	Linear	TE	GP Loss	Sawtooth Loss
HyperMixNP	✓	✗	1.144	-
LinearTNP	✓	✗	1.141	-
PT-TETNP	✓	✓	1.148	-
TETNP	✓	✓	**1.134**	**-1.407**
ConvNP	✗	✓	1.168	-0.8701

Table 0.5.1: Comparison of validation loss on 2D datasets using $N_c = 64$ $N_t = 128$ with 1 million parameter models. All models were trained for 4 hours. Fields with '-' indicate that the model was unable to learn the dataset.

Table 0.5.1 shows the validation loss of the models on the 2D datasets. All the linear models perform similarly to each other and outperform the ConvNP in the GP dataset. However on the Sawtooth dataset, none of the linear models are able to learn the dataset, even with hyperparameters tuning. Such a result is surprising, and potentially indicates that loss of expressive power by simplifying the attention mechanism to be linear. Ultimately, the TETNP massively outperforms all the models.

0.6 Computational Complexity

		Linear			Quadratic	
N_c	N_t	HyperMixNP	LinearTNP	PT-TNP	ConvNP	TETNP
10	10	16	16	19	24	13
100	10	16	16	19	29	23
1000	10	23	31	50	257	984
5000	10	69	109	190	1273	24082
10	1000	19	25	51	268	26
100	1000	19	31	51	268	113
1000	1000	24	32	51	268	985
5000	1000	70	109	191	1273	24083

Table 0.6.1: Memory usage of the models in MB under inference using N_c context points and N_t target points on the 2D datasets.

Table 0.6.1 demonstrates the drastic improvement in memory usage of the linear models compared to the quadratic models. The linear models are able to scale to larger datasets with a much smaller memory footprint, making them suitable for large-scale applications where data is smooth, and the quadratic models are infeasible to use.

0.7 Summary

The Linear Transformer models are a promising direction for scaling up the Transformer models to larger datasets. They were able to significantly reduce the memory complexity of the model, however the loss of performance on the Sawtooth dataset is concerning. More research is required to understand the limitations of the linear attention mechanism and potentially improve the performance of the models.

Bibliography

Ashman, Matthew, Cristiana Diaconu, Junhyuck Kim, Lakee Sivaraya, Stratis Markou, James Requeima, Wessel P. Bruinsma, and Richard E. Turner (2024). "Translation-Equivariant Transformer Neural Processes". In: *Forty-first International Conference on Machine Learning*. URL: https://openreview.net/forum?id=pftXzp6Yn3.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2016). *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. arXiv: 1511.07289 [cs.LG].

Feng, Leo, Hossein Hajimirsadeghi, Yoshua Bengio, and Mohamed Osama Ahmed (2023). *Latent Bottlenecked Attentive Neural Processes*. arXiv: 2211.08458 [cs.LG].

Ha, David, Andrew Dai, and Quoc V. Le (2016). *HyperNetworks*. arXiv: 1609.09106 [cs.LG].

Hendrycks, Dan and Kevin Gimpel (2023). *Gaussian Error Linear Units (GELUs)*. arXiv: 1606.08415 [cs.LG].

Hensman, James, Nicolo Fusi, and Neil D. Lawrence (2013). *Gaussian Processes for Big Data*. arXiv: 1309.6835 [cs.LG].

Jaegle, Andrew, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira (2021). *Perceiver: General Perception with Iterative Attention*. arXiv: 2103.03206 [cs.CV].

Katharopoulos, Angelos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret (2020). *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*. arXiv: 2006.16236 [cs.LG].

Lee, Juho, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh (2019). *Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks*. arXiv: 1810.00825 [cs.LG].

Mai, Florian, Arnaud Pannatier, Fabio Fehr, Haolin Chen, Francois Marelli, Francois Fleuret, and James Henderson (2023). *HyperMixer: An MLP-based Low Cost Alternative to Transformers*. arXiv: 2203.03691 [cs.CL].

Tolstikhin, Ilya et al. (2021). *MLP-Mixer: An all-MLP Architecture for Vision*. arXiv: 2105.01601 [cs.CV].