0.1 Introduction

From Table 0.4.1 and Table 0.3.2, 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 $\boldsymbol{X}$. The pseudotokens $\boldsymbol{I} \in \mathbb{R}^{M \times D}$ can then be used to perform cross-attention with the original tokens $\boldsymbol{X}$ thus reducing the memory complexity to $\mathcal{O}(MN_c + MM_t)$ making the model linear with respect to context and target set size.

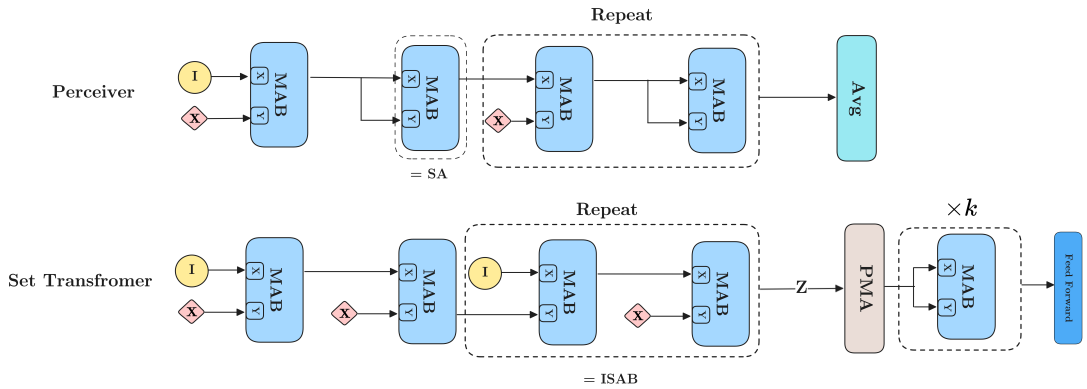

Figure 0.2.1: Perceiver vs Set Transformer. MAB is equivalent to Multi-Head Cross Attention, refer to [Ashman et al. 2024] for more details.

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 only really differ in the way they implement the cross-attention mechanism between the original and pseudo tokens. [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. Due to the similarity of the models, we expect both to have very similar performance on the tasks they are evaluated on.

0.3 Linear Transformer

[Katharopoulos et al. 2020] introduces a kernelized form of the self-attention mechanism that allows the model to be linear in the number of tokens by avoiding the softmax operation. 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]. It is clear that 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 $\boldsymbol{Q}_i$, thus reducing the complexity to $\mathcal{O}(N)$.

This can simply replace the transformer in the original TNP model, we will refer to this model as the 'Linear Transformer NP' (LinearTNP).

0.4 HyperMixer

Is self-attention required for a Transformer model to be effective? The majority of parameters in Transformers are in the MLPs and not the self-attention mechanism. [Tolstikhin et al. 2021] proposes a Transformer model that removes the self-attention mechanism and replaces with MLPs across rows and columns of the input, effectively 'mixing' the feature and data points dimensions in the input to learn patterns in a way akin to self-attention. This model is called the 'MLP-Mixer' and is shown to perform on par with the original Transformer models on image classification tasks. However, the 'MLP-Mixer' requires a **known fixed input size** which breaks the flexibility of the model if we apply it to a neural process.

To overcome this limitation in flexibility, the 'HyperMixer' [Mai et al. 2023] model 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 by using hypernetworks ($h_k, h_q : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times p}$) on the queries $\boldsymbol{Q}$ and keys $\boldsymbol{K}$ of the input (row wise).

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

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:

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

where σ is the activation function, the authors use the GELU function [Hendrycks and Gimpel 2023]. Cross attention can be 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 model to create the 'HyperMixNP' model.

Lack of Translation Equivariance

Both LinearTNP and HyperMixNP models are not translation equivariant. To introduce TE without using pseudotokens, we require the use of the pairwise differences' matrix $\mathbf{\Delta}$ as described in Equation 0.2.3, this 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

0.5.1 2D Gaussian Process

0.5.2 2D Sawtooth

0.5.3 Memory Complexity

Bibliography

Ashman, Matthew et al. (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 et al. (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 et al. (2021). *Perceiver: General Perception with Iterative Attention*. arXiv: 2103.03206 [cs.CV].

Katharopoulos, Angelos et al. (2020). *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*. arXiv: 2006.16236 [cs.LG].

Lee, Juho et al. (2019). *Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks*. arXiv: 1810.00825 [cs.LG].

Mai, Florian et al. (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].