Polynomial-based Self-Attention for Table Representation Learning

Jayoung Kim [1] **Yehjin Shin** [1] **Jeongwhan Choi** [1] **Hyowon Wi** [1] **Noseong Park** [2]

Abstract

Structured data, which constitutes a significant portion of existing data types, has been a long-standing research topic in the field of machine learning. Various representation learning methods for tabular data have been proposed, ranging from encoder-decoder structures to Transformers. Among these, Transformer-based methods have achieved state-of-the-art performance not only in tabular data but also in various other fields, including computer vision and natural language processing. However, recent studies have revealed that self-attention, a key component of Transformers, can lead to an oversmoothing issue. We show that Transformers for tabular data also face this problem. To tackle the problem, we suggest a novel self-attention layer for tabular data, leveraging matrix polynomials. This proposed layer serves as a replacement for the original self-attention layer, contributing to the improvement of model scalability. In our experiments with three representative table learning models equipped with our proposed layer, we illustrate that the layer effectively mitigates the oversmoothing problem and enhances the representation performance of the existing methods, outperforming the state-of-the-art table representation methods.

1. Introduction

Out of the top 10 database management systems, 7 are relational databases, including Oracle, MySQL, and Microsoft SQL Server [1]. Likewise, structured data is one of the most common data types in the fields of data mining and machine learning. With the increasing focus on tabular data, several recent methods have demonstrated remarkable success in table representation, such as (Huang et al., 2020; Ucar et al., 2021; Somepalli et al., 2021; Majmundar et al., 2022), with many of them being Transformer-based methods.

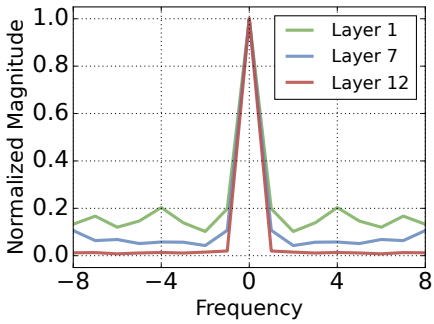


Figure 1: Spectral response of an attention map from TabTransformer (Huang et al., 2020)

Transformers have made significant advancements in deep learning, becoming state-of-the-art models in various domains, including computer vision and natural language processing (Vaswani et al., 2017; Radford et al., 2018; Devlin et al., 2019; Gulati et al., 2020; Ying et al., 2021; Dosovitskiy et al., 2021; Touvron et al., 2021; Yu et al., 2024; Shin et al., 2024). However, recent studies have raised concerns about the potential limitations of self-attention, a fundamental component of Transformers, specifically an issue of *oversmoothing* (Dong et al., 2021; Wang et al., 2022; Guo et al., 2023b; Choi et al., 2023; Xue et al., 2023). Gong et al. (2021) and Zhou et al. (2021) have highlighted that at deeper layers of the Transformer architecture, all token representations tend to become nearly identical (Brunner et al., 2019). The problem poses challenges when it comes to expanding the scale of training Transformers, especially in terms of depth, since Transformers rely on a simple weighted average aggregation method for value vectors.

In our preliminary experiments, we observe that Transformers designed for tabular data also exhibit the oversmoothing issue, as illustrated in Fig. 1. As we go deeper into the layers, TabTransformer (Huang et al., 2020), a model designed for tabular data, tends to focus more on low-frequency components in its attention mechanism, even though table column relationships could be represented using a wider range of components. (For a more detailed discussion, please refer to Sec. 2.3.) To address this challenge, we propose a redesigned self-attention for table representation in this paper.

[1]Yonsei University, South Korea [2]KAIST, South Korea. Correspondence to: Noseong Park <noseong@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]https://db-engines.com/en/ranking

Our design is inspired by graph signal processing (GSP). In a general sense, a graph filter on a graph $\mathcal{G}$ is typically expressed as a polynomial based on its adjacency or Laplacian matrix. In the context of our work, the conventional self-attention mechanism can be considered the most basic graph filter, utilizing only $\mathbf{A}$, where $\mathbf{A} \in [0,1]^{n \times n}$ represents a learned attention matrix that encodes relationships between columns of tabular data, and n is the number of input tokens. In other words, the proposed mechanism generalizes the original self-attention by allowing for more flexibility and customization. Building upon this notion, we replace the self-attention layer with our proposed polynomial-based layer, designed to approximate an optimal graph filter. In this context, we introduce our novel self-attention layer for table representation learning as *TabPSA* (**Tab**le Representation using **P**olynomial-based **S**elf-**A**ttention).

Our proposed self-attention mechanism is composed of coefficients α_k for each polynomial term $\mathbf{A}^k$, where k represents the order of the polynomial. It is crucial to emphasize that computing $\mathbf{A}^k$ may pose computational challenges when dealing with a substantial number of tokens, a common occurrence in natural language processing tasks. However, in the case of tabular data, the number of tokens is typically small because each table column is treated as a token. Consequently, tabular data do not require computationally expensive processes due to the smaller attention matrix, making it more feasible to compute $\mathbf{A}^k$ with a moderately large value of k. This approach allows us to design a more scalable graph filter that leverages the characteristic.

High order polynomials require multiple squares. Here, we make use of the property of PageRank. PageRank converges after a few iterations when a transition matrix satisfies three conditions: i) stochasticity, ii) irreducibility, and iii) aperiodicity. Surprisingly, attention matrices satisfy all three conditions, as discussed in Sec. 4.1. This means high order polynomial terms also converge, and thus we do not need to compute higher order polynomial terms.

Furthermore, our proposed graph filter is able to capture a wider range of frequency information as discussed in Sec. 5.2. To summarize, graph filter approximated by TabPSA encompasses both low and high-frequency components, while others often lack high-frequency signals. In summary, our contributions are as follows:

1. To the best of our knowledge, we present the first study on the self-attention in the field of tabular data.
2. We propose table representation learning based on Transformer with self-attention tailored to tabular data improves representation quality compared to existing deep learning methods.
3. We have developed a matrix polynomial-based self-attention mechanism that efficiently leverages properties of PageRank and self-attention matrix, without a substantial increase in computational cost.
4. Tree-based ensemble models outperform deep neural networks for tabular data in many cases (Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022). Our method outperforms them in 6 out of the 10 experimented datasets, which is a significant improvement.

2. Related Work

2.1. Representation Learning for Tabular Data

Representation learning focuses on learning meaningful features from raw data. Recently, there has been a growing focus on representation learning for tabular data. The challenges in table representation learning stem from the absence of common correlation structure in tabular data unlike the case of image and text data (Yoon et al., 2020). Yoon et al. (2020) proposed VIME which is an approach to self- and semi-supervised learning tailored for tabular data. It incorporates a unique pretext task focused on estimating mask vectors from corrupted tabular data, along with the reconstruction pretext task. SubTab (Ucar et al., 2021) is a self-supervised learning framework designed for tabular data, which partitions the input features into multiple subsets, enhancing its ability to capture more efficient latent representations.

Transformer-based models have emerged as dominant approaches for learning useful features for tabular data (Majmundar et al., 2022; Somepalli et al., 2021; Huang et al., 2020). TabTransformer (Huang et al., 2020) employs a Transformer encoder to acquire contextual embeddings for only categorical features. SAINT (Somepalli et al., 2021) maps both continuous and categorical features into an embedding space and then processes them through the Transformer blocks. SAINT utilizes contrastive learning and calculates attention scores over both rows and columns to get enhanced embeddings. MET (Majmundar et al., 2022) is a table representation model based on masked autoencoders. It employs encoders with random masking to acquire positional embeddings for individual feature coordinates, enabling the capture of latent structures among these coordinates. These days, representations of tables are used to improve the performance of downstream tasks for tabular data, such as classification and regression, where deep learning models are struggling to beat traditional machine learning approaches.

2.2. Self-Attention Mechanism in Transformers

Self-Attention mechanism is a key components of Transformer architecture. Each input embedding is projected onto three parametric matrices: key, query, and value matrices, denoted as $\mathbf{K} \in \mathbb{R}^{n \times d}$, $\mathbf{Q} \in \mathbb{R}^{n \times d}$ and $\mathbf{V} \in \mathbb{R}^{n \times d}$, respectively. The self-attention mechanism SA can be expressed

as follows:

$$SA(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V} = \mathbf{A}\mathbf{V}, \quad (1)$$

where d is the scale factor and n is the number of input tokens. The basic idea of self-attention is to establish correlations between tokens (features) by assessing similarity between their key and query representations. With the calculated attention matrix $\mathbf{A}$, which is equal to $softmax(\mathbf{Q}\mathbf{K}^T/\sqrt{d})$, a value matrix $\mathbf{V}$ is re-weighted through dot-product.

Self-Attention is known to have similar characteristics to those of a graph convolution network (GCN). GCNs are designed to process data that can be represented as graphs denoted as $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a node set and $\mathcal{E}$ is a set of edges connecting node pairs. Using graph convolutional layers, they learn representations of nodes within a graph, taking into account information from their local neighborhoods. Self-attention matrix used in Transformers can be seen as a normalized adjacency matrix of tokens (Guo et al., 2023b).

2.3. Oversmoothing in GCNs and Transformers

Oversmoothing is a phenomenon that can be observed in deep learning models, particularly in GCNs (Kipf & Welling, 2017). It describes a problem where a network excessively smooths node features during the aggregation process, potentially resulting in reduced discriminative capability in node representations (Oono & Suzuki, 2020; Zhou et al., 2020; Rusch et al., 2023).

Surprisingly, oversmoothing phenomenon is also observed in Transformer (Wang et al., 2022; Shi et al., 2022). Unlike convolutional neural networks (CNNs), Transformers do not show performance improvements by adding more layers beyond a specific threshold. This issue arises from attention matrices that are similar to GCNs. In other words, it is a fundamental problem in Transformers, and these problems occur in Transformer-based models across different domains. Dong et al. (2021) identifies the issue of "token uniformity", which diminishes the effectiveness of Transformer-based architectures by causing all token representations to be the same. Shi et al. (2022) explores hierarchical fusion strategies, which adaptively combine representations from various layers to introduce diversity into the output, thereby mitigating the oversmoothing issue.

Wang et al. (2022) provide a theoretical basis for the oversmoothing phenomenon in Transformers, particularly in self-attention. According to Theorem 1 in (Wang et al., 2022), they demonstrate that self-attention functions as a low-pass filter, continuously diminishing high-frequency information.[2] This inherent characteristic leads to the oversmoothing phenomenon, as unique high-frequency features are lost in the deeper layers of the network, further exacerbating the uniformity of token representations.

Through experiments, we observed the same issue occurring in Transformer-based table representation models. Therefore, we aim to propose an attention matrix from the perspective of graph filters that can enhance Transformer-based table representation models.

3. Preliminaries

3.1. Graph Signal Processing

Leveraging insights from graph signal processing (GSP), we designed our new attention method, TabPSA. GSP has a close connection to discrete signal processing (DSP). In DSP, a discrete signal with a length of n can be represented by a vector $\mathbf{x} \in \mathbb{R}^n$. Let $\mathbf{g} \in \mathbb{R}^n$ be a filter applying to $\mathbf{x}$. The convolution $\mathbf{x} * \mathbf{g}$ can be computed as follows:

$$\mathbf{y}_i = \sum_{j=1}^{n} \mathbf{x}_j \mathbf{g}_{i-j}, \quad (2)$$

where the index refers to the i-th element in each vector.

GSP can be viewed as a generalized case of DSP — in other words, DSP is a special case of GSP where a *ring graph with n nodes* is used and therefore, the graph Fourier transform of the line graph is identical to the discrete Fourier transform. In addition, the linear and shift-invariant graph convolution filter with n nodes can be written with a shift operator $\mathbf{S}$ (regardless of the diagonalizability of $\mathbf{S}$, i.e., $\mathbf{S}$ can be from directed graphs (Marques et al., 2020)) as follows:

$$\mathbf{y} = \sum_{k=0}^{n-1} w_k \mathbf{S}^k \mathbf{x}, \quad (3)$$

where $\mathbf{x} \in \mathbb{R}^n$ is a 1-dimensional graph signal and $w_k \in [-\infty, \infty]$ is a coefficient. $\mathbf{S}$ is an $n \times n$ matrix where (i, j)-th element is non-zero if and only if there is an edge from node i to j — its diagonal elements can also be non-zeros and therefore, two representative examples of $\mathbf{S}$ are adjacency and Laplacian matrices. We note that Eq. (3) is a generalization of Eq. (2) under the context of GSP. Eq. (3) can be simplified as follows:

$$\mathbf{y} = \mathbf{H}\mathbf{x}, \quad (4)$$

where the linear and shift-invariant graph filter $\mathbf{H}$ is the same as $\sum_{k=0}^{n-1} w_k \mathbf{S}^k$ in Eq. (3) which is called *matrix polynomial*.

[2] The Fourier Transform converts a signal from the time domain to the frequency domain, decomposing it into individual frequency components. Low-frequency signals change slowly and represent broader trends, while high-frequency signals change quickly and capture finer details.

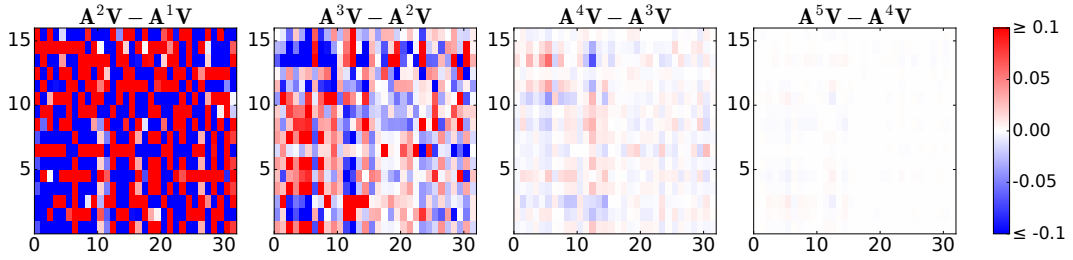


Figure 2: Convergence of $\mathbf{A}^k\mathbf{V}$, where $\mathbf{A}$ is an attention matrix and $\mathbf{V}$ is a value matrix for Phishing

We note that this graph filtering operation can be extended to d-dimensional cases. Therefore, the core part of the self-attention, i.e., $\mathbf{AV}$, can be considered as a d-dimensional graph filter with $\mathbf{A}$ only, where $\mathbf{H} = \mathbf{A}$ and $\mathbf{A}$ is a shift operator. Our goal in this paper is design an effective form of $\mathbf{H}$ considering the characteristics of tabular data.

3.2. PageRank

PageRank is an algorithm used to assess the significance of web pages by considering both the quality and quantity of links leading to them, which, in turn, influences their rankings in search engine results. We refer to the collection of web pages (or nodes) as W and the network of links (or directed edges) as E. If a page u has a link pointing to page v, then we say $(u, v) \in E$. We denote the number of links leading out of a page v as d_v, and the PageRank score of page v as π_v. To explain PageRank, we assume a random surfer who navigates web pages based on a transition probability matrix $M \in \mathbb{R}^{N\times N}$ and a visiting probability vector $\pi^{(t)} \in \mathbb{R}^N$, where N is the total number of pages and t is the current iteration. In the matrix M, M_{wv} is equal to $1/d_v$ if page v links to page w and 0 otherwise. The PageRank equation can be expressed as follows:

$$\pi_v^{(t)} = (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{\pi_w^{(t-1)}}{d_w}\Big) + \frac{\epsilon}{N}, \qquad (5)$$

where $\pi_v^{(t)}$ is the iterative PageRank score of page v after t iterations and ϵ is reset probability, representing the probability that the random surfer randomly jumps to another page. PageRank score can be computed iteratively as shown in Eq. (5), and the iterative method can be viewed as the power iteration. PageRank score converges quickly when its transition matrix M satisfies three conditions: i) stochasticity, ii) irreducibility, and iii) aperiodicity.

4. TabPSA

In this section, we present the details of our design in a sequential manner. We start by introducing the inspiring concept behind our design, PageRank. Following that, we delve into the matrix-polynomial for our self-attention layer. Finally, we introduce another key component of our design, Jacobi basis, and discuss our design from various angles.

4.1. PageRank

PageRank scores that contain the importance of pages, converge quickly when its transition matrix satisfies three conditions, as in Proposition 4.1. The three conditions are as follows: i) the transition matrix must be a stochastic, ii) irreducible, and iii) aperiodic matrix. Interestingly, attention matrix $\mathbf{A}$ in Transformers meet all the 3 conditions:

1. **Stochasticity**: The softmax function in Transformers ensures that attention scores are normalized, making the attention matrix stochastic because values in each column sum to 1.
2. **Irreducibility**: In Transformers, attention matrices assign a non-zero probability to focus on any part of the input sequence from any position in the output sequence — note that this is guaranteed by the softmax function (cf. Eq. (1)). This ensures the existence of a pathway, though not always direct, connecting any position to any other, satisfying the condition of irreducibility.
3. **Aperiodicity**: The aperiodicity in Markov chains condition denotes the lack of repeating patterns. In short, the irreducible chain is aperiodic if all states have a period of 1, which means that each state has at least one self-loop. This is the case in the self-attention since the attention matrix has non-zero elements, i.e., completely connected, although some are close to zeros after the softmax function — note that a negative infinite logit is required for the softmax function to produce a zero, which is not likely in neural networks.

Proposition 4.1 (Convergence of PageRank)**.** *Define the error term as a difference between the exact PageRank score π_v^* and the t-th PageRank score $\pi_v^{(t)}$: $Err(t) = \sum_v |\pi_v^{(t)} - \pi_v^*|$, where $\pi_v^{(t)} = (1-\epsilon)\left(\sum_{(w,v)\in E} \frac{\pi_w^{(t-1)}}{d_w}\right) + \frac{\epsilon}{N}$. Then, the total error converges within a small number of iterations. The proof is in Appendix A.*

According to Proposition 4.1, in other words, since the attention matrix $\mathbf{A}$ satisfies the conditions, $\mathbf{A}^k\mathbf{V}$, where

$k \in \mathcal{R}$, converges with a small k in matrix polynomial. More discussions are in the following section. Moreover, Fig. 2 shows the convergence of $\mathbf{A}^k\mathbf{V}$. The change of the result of $\mathbf{A}^k\mathbf{V} - \mathbf{A}^{k-1}\mathbf{V}$ quickly decreases to 0 as k increases. In Appendix B, we discuss the convergence of the attention matrix in detail.

4.2. Matrix Polynomial-based Transformer

Let $\mathbf{A} \in [0,1]^{n\times n}$, where n is the number of tokens[3], be a self-attention matrix, and $\mathbf{V}^{n\times d}$, where d is dimension of each token and $\mathbf{V}$ is a value matrix. Self-attention, which can also be viewed as a simplified version of graph filters, can be extended using matrix polynomial. By extending self-attention with matrix polynomial, the extended definition can be expressed as follows — as described earlier for Eq. (3), $\mathbf{A}$ can be a directed adjacency matrix:

$$\mathbf{HV} = \sum_{k=0}^{n-1} w_k \mathbf{A}^k \mathbf{V}, \tag{6}$$

where w are polynomial coefficients. The extended equation requires large computation of high-order power of $\mathbf{A}$. However, due to the nature of tables, which typically have only tens of columns (tokens), the computational cost becomes manageable. The expression of matrix polynomial-based self-attention is as follows:

$$\mathbf{HV} \approx w_0\mathbf{V} + w_1\mathbf{AV} + w_2\mathbf{A}^2\mathbf{V} + \cdots + w_j\mathbf{A}^j\mathbf{V}, \tag{7}$$

where j is a point where the convergence error is tolerable with respect to an enough low bound b, i.e., $\|\mathbf{A}^i\mathbf{V} - \mathbf{A}^j\mathbf{V}\|_F \leq b, \forall i \geq j$. Therefore, all terms higher than j are absorbed to $w_j\mathbf{A}^j\mathbf{V}$ (cf. Proposition 4.1 and Fig. 2). As known in the existing works, we can understand that self-attention inevitably dampens, as shown in Fig. 1, the high-frequency elements (Dong et al., 2021; Wang et al., 2022; Guo et al., 2023b; Xue et al., 2023). Consequently, the original self-attention is not suitable for tasks involving representation learning, which require capturing all forms of information from the data. Conversely, when we allow w to be learned and potentially take on negative values through the model learning, the graph filter will become capable of conveying high-frequency information, as we prove with our experiments in Sec. 5.

4.3. Jacobi Bases

Yet, the optimization of $w_k, 0 \leq k \leq j$, using Eq. (7) may encounter instability due to the non-orthogonality of the set of bases, denoted as $\{\mathbf{A}^k | 0 \leq k \leq j\}$. This lack of orthogonality implies an absence of guaranteed convergence. In light of this, Jacobi presents itself as a suitable alternative for the matrix polynomial discussed earlier. Jacobi basis,

[3]In the perspective of GSP, n represents the number of nodes.

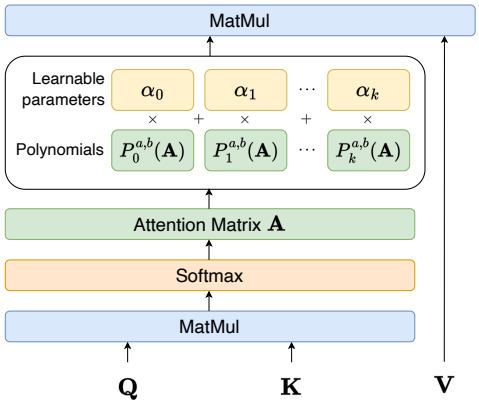


Figure 3: Architecture of the proposed TabPSA

denoted as $\mathbf{P}_k^{a,b}(x)$, can be recursively defined as follows:

$$\mathbf{P}_0^{a,b}(x) = 1, \tag{8}$$

$$\mathbf{P}_1^{a,b}(x) = \frac{a-b}{2} + \frac{a+b+2}{2}x. \tag{9}$$

For $k \geq 2$, it satisfies

$$\mathbf{P}_k^{a,b}(x) = (\theta_k x + \theta'_k)\mathbf{P}_{k-1}^{a,b}(x) - \theta''_k\mathbf{P}_{k-2}^{a,b}(x), \tag{10}$$

where

$$\theta_k = \frac{(2k+a+b)(2k+a+b-1)}{2k(k+a+b)}, \tag{11}$$

$$\theta'_k = \frac{(2k+a+b-1)(a^2-b^2)}{2k(k+a+b)(2k+a+b-2)}, \tag{12}$$

$$\theta''_k = \frac{(k+a-1)(k+b-1)(2k+a+b)}{(k(k+a+b)(2k+a+b-2)}. \tag{13}$$

$\mathbf{P}_k^{a,b}, k = 0, \cdots, j$ are orthogonal with a weight function $(1-x)^a(1+x)^b$ on $[-1,1]$ with $a, b > -1$. Therefore, we use this Jacobi polynomial to stabilize the training of the coefficients.

To conclude, the self-attention with our expended graph filter is as follows:

$$\mathbf{HV} \approx \alpha_0\mathbf{P}_0^{a,b}(\mathbf{A})\mathbf{V} + \alpha_1\mathbf{P}_1^{a,b}(\mathbf{A})\mathbf{V} \tag{14}$$

$$+ \cdots + \alpha_j\mathbf{P}_j^{a,b}(\mathbf{A})\mathbf{V}, \tag{15}$$

where α are the Jacobi polynomial coefficients. Eq. (14) can be rewritten to a polynomial of $\mathbf{A}$, since $\mathbf{P}_k^{a,b}(\mathbf{A}), \forall k$, is a function of $\mathbf{A}$. Thus, we utilize Jacobi polynomial of order j since the self-attention $\mathbf{A}^j\mathbf{V}$ converges rapidly with a small j (cf. Proposition 4.1).

Special cases of Jacobi bases. As a special case of Jacobi polynomial, Chebyshev and Legendre polynomials also are

orthogonal polynomials which can be applied to our proposed method. Specifically, Chebyshev polynomial is a special case of Jacobi polynomial when $a = b = -\frac{1}{2}$, and Legendre polynomial is a special case of Jacobi polynomial when $a = b = 0$.

4.4. Discussions

Graph filtering aspects of TabPSA. TabPSA enables better graph filter approximation through its Jacobi polynomial approximation. TabPSA involves iterative matrix powers as shown in Eq. (14). Extensive computational resources are necessary when dealing with attention matrices in tasks that involve large datasets, such as images and graphs, which can consist of tens of thousands of tokens. For this reason, studies aiming to alleviate oversmoothing with matrix polynomial-based graph filters often limit the use of Laplacian matrix powers or optimize substitute parameter(s) to approximate the graph filter (Chien et al., 2020; Gasteiger et al., 2018; He et al., 2021). In contrast, attention matrices for tables, typically containing fewer than 100 columns, involve a relatively small number of tokens, making computation more manageable. In this context, our design is more suitable for tabular data than datasets with large attention matrices.

Apply on Transformers. Our self-attention layer is designed by drawing inspiration from the core concepts of graph signal processing, where self-attention can be seen as a specialized form of matrix polynomial operations. It is noteworthy to emphasize that the seamless adoption of TabPSA framework into the Transformer architecture entails a simple step: the substitution of the conventional self-attention layer with our self-attention layer. This architectural substitution not only demonstrates the flexibility and compatibility of our approach but also underscores its potential to enhance the performance of existing Transformer-based models.

Representation performance on the existing methods. To validate the effectiveness of TabPSA, we apply it to the existing Transformer-based table representation models, specifically TabTransformer (Huang et al., 2020), SAINT (Somepalli et al., 2021), and MET (Majmundar et al., 2022), with minor modifications. Details are in Appendix C. The results are in the following section.

5. Experiments

5.1. Experimental Environments

Experimental settings. Our software and hardware environments are as follows: UBUNTU 20.04 LTS, PYTHON 3.8.2, PYTORCH 1.8.1, CUDA 11.4, and NVIDIA Driver 470.42.01, i9 CPU, and NVIDIA RTX A5000.

Evaluation methods. We use 10 datasets and 10 baselines for our experiment. Details of the datasets and baselines can be found in Appendices D.1 and D.2, respectively. To demonstrate the efficacy of TabPSA, three selected base models for table learning are compared with base models trained using TabPSA. After training the representation models as proposed in the original paper, we subsequently train auxiliary small MLP layers for classification/regression. For classification, AUROC scores are reported, while for regression, R^2 scores are presented. Each experiment is conducted five times, and the corresponding mean values and standard deviations are reported.

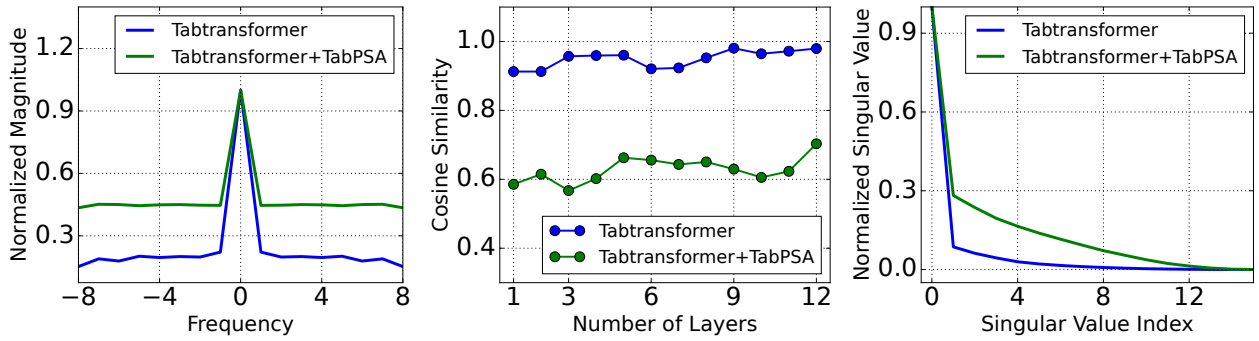
5.2. Experimental Results

Table 1: Relative score improvement over the base Transformers. TabTransf. represents TabTransformer.

	TabTransf.	SAINT	MET
Base model	76.9	84.5	79.4
Base model + TabPSA	**78.5**	**85.4**	**84.0**
Improvement	2.14%	1.02%	5.76%

Firstly, we discuss the efficacy of TabPSA. The experimental results are summarized in Table 1. As shown, TabPSA improves the base models in all cases, increasing performance by at least 1.02% in each instance. Notably, for MET, TabPSA is particularly effective, yielding an average performance increase of 5.76%. This enhancement can be attributed to TabPSA's ability to capture diverse signal frequencies. It is important to note that this suggests TabPSA has the potential to further enhance the performance of Transformers, including both existing models and those yet to be developed.

Fig. 4 is a visualization of feature maps from the trained TabTransformer. Each figure is plotted with the averaged values over the test dataset. In Fig. 4 (a), TabTransformer trained with TabPSA significantly retains high-frequency data compared to TabTransformer. In Fig. 4 (b), we present token-wise cosine similarity with respect to the layers. Greater cosine similarity indicates that the tokens in a layer become more similar, which is a symptom of oversmoothing. Compared to TabTransformer+TabPSA, TabTransformer exhibits higher cosine similarity in general, and as the layers get deeper, cosine similarity slightly increases, which can also indicate oversmoothing. In Fig. 4 (c), we present the normalized singular values of feature maps. The rapid decrease in singular values of TabTransformer indicates that the feature maps are approximately in an extremely low-rank. On the other hand, the slow decrease in singular values of TabTransformer+TabPSA indicates that the feature maps are more representative.



(a) Spectral response (b) Cosine similarity (c) Singular value

Figure 4: Visualization of spectral response, cosine similarity, and singular values of feature maps in Phishing. TabTransformer+TabPSA represents TabTransformer trained with TabPSA.

Table 2: Sensitivity experiment with respect to k. The reported scores are AUROC (↑) for classification, and R^2 (↑) for regression.

Datasets	k	TabTransf. + TabPSA	SAINT + TabPSA	MET + TabPSA
Alphabank	2	62.1±0.21	62.5±0.72	60.1±0.50
	3	**62.1±0.21**	63.2±0.97	**62.7±0.31**
	5	61.5±1.67	**63.4±0.87**	61.8±0.45
	10	61.5±1.55	63.0±0.50	61.7±0.59
Contraceptive	2	65.0±1.72	75.6±0.47	78.1±0.91
	3	65.3±1.95	77.0±0.85	**79.0±1.13**
	5	66.0±1.63	**77.4±1.51**	77.5±0.63
	10	**66.0±0.97**	76.9±0.83	78.0±1.41
Medicalcost	2	0.61±0.01	0.86±0.00	0.87±0.24
	3	**0.61±0.02**	0.87±0.00	**0.87±0.00**
	5	0.61±0.00	**0.87±0.00**	0.86±0.01
	10	0.61±0.00	0.87±0.00	0.86±0.00

Table 3: Experimental result w.r.t. matrix polynomial forms. The reported scores are AUROC (↑) for classification, and R^2 (↑) for regression.

Datasets	Polynomials	TabTransf. + TabPSA	SAINT + TabPSA	MET + TabPSA
Default	Power	**78.2±0.08**	77.1±0.39	77.9±0.43
	Chebyshev	78.0±0.06	77.4±0.71	**77.9±0.33**
	Legendre	78.0±0.06	**78.4±0.30**	77.2±0.81
	Jacobi	78.0±0.13	50.5±0.78	63.4±4.62
Buddy	Power	89.5±1.94	**94.8±0.73**	**87.6±4.16**
	Chebyshev	88.8±1.27	92.1±0.96	87.6±1.07
	Legendre	**90.0±1.20**	90.8±1.25	83.9±3.49
	Jacobi	89.0±1.29	53.0±2.96	73.2±20.4
Activity	Power	79.8±1.27	89.7±0.73	88.7±2.93
	Chebyshev	80.2±1.41	90.8±0.53	**90.9±2.39**
	Legendre	**80.9±0.67**	**91.1±0.47**	90.9±1.06
	Jacobi	80.3±1.59	49.7±0.44	70.2±9.27

5.3. Sensitivity on the Order of Polynomial

We perform a sensitivity experiment with respect to the order of polynomials, and the results are summarized in Table 2. We set the order of polynomial k to 2, 3, 5, and 10. In general, we get the best scores within 5 order of polynomials, except for one case. After a certain threshold of k, model performance tends to saturate for all models. This means that we do not need to use high-order polynomials to approximate the graph filter, as discussed above (cf. Section 4).

5.4. Exploring Different Polynomial Bases

We compare four polynomial bases: Power, Chebyshev, Legendre, and Jacobi, the latter three of which are orthogonal. The results are summarized in Table 3. Generally, we find that the representation performance is robust across different types of polynomials. Among the four bases, Chebyshev and Legendre polynomials provide better representation of datasets, achieving the highest scores in 6 out of 9 cases. Interestingly, the performance of the Jacobi polynomial deteriorates in some instances; for example, in all three datasets when combined with SAINT+TabPSA and MET+TabPSA. Moreover, in the case of MET+TabPSA, Jacobi polynomial introduces substantial standard deviation. This can be attributed to the tendency of the Jacobi polynomial to suppress mid-frequency signals, as noted by Guo et al. (2023a), while effective data representation requires capturing a diverse range of frequency signals.

5.5. Comparison to Other Methods

Table 4 presents the performances of various methods including machine learning models and deep learning models. In this comparison, machine learning models exhibit strong performance, which is widely acknowledged within our community (Grinsztajn et al., 2022). However, in 6 out of

Table 4: Comparison of base models with our proposed self-attention layer and other models. Contra., Medical., and Super. represent Contraceptive, Medicalcost, and Superconductivity, respectively. TabTransf. means TabTransformer. We report AUROC (↑) for classification and R^2 (↑) for regression. The best results are in **boldface**, and the second-best results are underlined. For Superconductivity, TabTransformer, which makes attention using categorical features only, cannot be used due to the absence of categorical features.

Methods	Binary Classification				Multi-class Classification				Regression	
	Income	Default	Phishing	Alphabank	Clave	Contra.	Activity	Buddy	Medical.	Super.
MLP	89.8±0.14	78.2±0.28	84.9±0.15	62.1±0.38	92.0±0.80	68.5±4.30	86.1±1.01	85.7±2.62	0.74±0.00	0.86±0.01
Decision Tree	89.5±0.07	76.2±0.00	83.1±0.00	60.4±0.06	84.8±0.17	75.8±0.00	88.5±0.23	82.1±0.04	0.87±0.00	0.84±0.00
Regression	57.3±0.00	65.1±0.00	85.2±0.00	61.5±0.00	91.0±0.00	73.6±0.00	68.8±0.00	50.0±0.00	0.75±0.00	0.72±0.00
XGBoost	**92.1±0.07**	77.5±0.21	82.3±0.43	60.5±0.54	95.9±0.09	75.0±0.56	**98.1±0.06**	93.5±0.30	0.81±0.01	0.90±0.00
Random Forest	91.2±0.02	**78.6±0.15**	85.0±0.15	59.8±0.15	93.3±0.17	77.3±0.08	98.0±0.04	88.5±1.34	0.87±0.00	**0.91±0.00**
TabNet	89.8±0.10	77.1±0.67	81.9±0.70	61.8±0.62	87.0±1.58	52.4±8.17	66.6±1.87	79.8±5.44	-1.18±0.02	0.88±0.00
VIME	87.3±44.5	78.0±0.26	83.3±0.56	60.8±0.88	95.8±0.21	69.1±1.82	76.5±1.41	80.6±2.43	0.80±0.06	0.87±0.01
TabTransformer	88.9±0.87	78.2±0.07	84.2±0.35	59.5±1.16	92.9±0.85	64.1±1.58	78.0±2.54	85.9±2.11	0.60±0.00	-
SAINT	91.0±0.07	78.4±0.23	85.3±0.11	60.9±1.60	96.5±0.19	75.4±0.91	89.2±1.32	94.7±0.57	0.86±0.01	0.88±0.00
MET	87.8±2.63	76.9±0.67	84.5±0.46	61.8±1.80	92.9±0.25	76.5±1.55	59.0±4.66	84.6±1.71	0.85±0.01	0.86±0.01
TabTransf.+TabPSA	89.0±0.94	78.2±0.08	84.6±0.20	62.1±0.21	94.6±0.63	66.0±0.97	80.9±0.67	90.0±1.20	0.61±0.02	-
SAINT+TabPSA	91.2±0.35	78.4±0.30	**85.7±0.34**	**63.4±0.87**	**96.8±0.16**	77.4±1.51	91.1±0.47	**94.8±0.73**	**0.87±0.00**	0.88±0.00
MET+TabPSA	89.7±0.16	77.9±0.33	85.5±0.31	62.7±0.31	93.1±0.11	**77.9±2.47**	90.9±2.39	87.6±4.16	0.87±0.00	0.88±0.00

10 datasets, Transformers with TabPSA achieves the highest score among all the evaluated models, outperforming the machine learning models. In the remaining 4 datasets, Transformer-based model with TabPSA is very close to the best model except for Activity. In Activity, ensemble models, XGBoost, and Random Forest perform better than other methods. However, among the remaining methods, excluding ensemble methods, SAINT and MET with TabPSA consistently demonstrate the highest performance. In case of Alphabank, Contraceptive, and Medicalcost, the Transformer-base models alone do not outperform the other methods. However, when TabPSA is incorporated into the base models, they outperform all other methods, which clearly demonstrates the effectiveness of TabPSA. In Alphabank, Clave, and Buddy, the Transformer-based model exhibit high performance surpassing that of ensemble models, and the addition of TabPSA to the base model further improve its performance. This indicates that enhancing the performance of the base model can lead to the creation of even better-performing models. In Appendix F, we summarize the results of various additional experiments, including combining TabPSA with Transformers for language modeling and performing sensitivity analyses with regularization on polynomial coefficients.

5.6. Computation Efficiency Analysis

Time complexity. The original attention mechanism has a time complexity of $\mathcal{O}(n^2 d)$, where n is the number of tokens and d is a dimension of each token. TabPSA adds complexity to compute A^k with $k-1$ matrix multiplications, resulting in a time complexity of $\mathcal{O}(n^2 d + (k-1)n^{2.371552})$, where we assume that we use algorithm in (Williams et al., 2024). Practically, when $d > (k-1)n^{2.371552}$, the time complexity of TabPSA becomes $\mathcal{O}(n^2 d)$, a condition met in almost all cases in our experiments.

Table 5: Training time per epoch in wall-clock seconds (↓) and memory usage for training models in milliseconds (↓)

	Training time (per epoch)	Memory usage
TabTransformer	1.84s	19.26MB
TabTransf.+TabPSA	1.87s (↑ 1.63%)	19.30MB (↑ 0.21%)
SAINT	3.81s	251.02MB
SAINT+TabPSA	4.55s (↑ 25.08%)	251.29MB (↑ 0.18%)
MET	2.11s	273.85MB
MET+TabPSA	2.32s (↑ 10.11%)	301.78MB (↑ 10.24%)

Memory usage. In Table 5, we summarize the wall-clock training time and memory usage required for training Transformers with our proposed method. These metrics are averaged across all datasets, with detailed results available in Appendix E. To ensure a fair comparison, the architecture of the base models and the +TabPSA models are maintained constant, and Jacobi polynomial of order 3 is employed for TabPSA. Our findings indicate that, on average, integrating TabPSA leads to an increase in training time of up to 25%. Regarding memory usage, the addition of TabPSA results in only a minimal increase, with the additional learnable

parameter count for incorporating TabPSA being $k + 1$.

6. Conclusion

Modern Transformers have revealed limitations related to oversmoothing, a phenomenon where as the depth of the Transformer model increases, hidden representations become similar for all tokens. For tabular data, we show that this problem also occurs. In order to address this phenomenon, we propose the use of polynomial-based self-attention, drawing inspiration from graph signal processing. In our experiments, which encompassed 10 datasets and 10 baseline models, Transformer-based table representation learning models trained with our proposed self-attention mechanism, demonstrated significant performance improvements in downstream tasks such as classification and regression. These improvements are substantial. We anticipate our proposed method, TabPSA, can used for other Transformers, if any, when the token numbers are not large.

Impact Statement

Given our study on the novel self-attention layer for Transformer-based methods for tabular data, the societal impacts could be substantial. This work has the potential to significantly improve data analyses across various fields, making processes more efficient and accurate. However, it also brings to the fore ethical considerations around data privacy and the fairness of automated decisions. As we push the boundaries of machine learning, it is vital to consider these implications to ensure that advancements benefit society equitably and responsibly.

Acknowledgements

This work was supported by Institute for Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2021-0-00231, Development of Approximate DBMS Query Technology to Facilitate Fast Query Processing for Exploratory Data Analysis).

References

Arik, S. Ö. and Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6679–6687, 2021.

Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., and Wattenhofer, R. On identifiability in transformers. *arXiv preprint arXiv:1908.04211*, 2019.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Choi, J., Wi, H., Kim, J., Shin, Y., Lee, K., Trask, N., and Park, N. Graph convolutions enrich the self-attention in transformers! *arXiv preprint arXiv:2312.04234*, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Dong, Y., Cordonnier, J.-B., and Loukas, A. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pp. 2793–2803. PMLR, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Fuller, D. Replication Data for: Using machine learning methods to predict physical activity types with Apple Watch and Fitbit data using indirect calorimetry as the criterion., 2020.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Gong, C., Wang, D., Li, M., Chandra, V., and Liu, Q. Vision transformers with patch diversification. *arXiv preprint arXiv:2104.12753*, 2021.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

Guo, J., Du, L., Chen, X., Ma, X., Fu, Q., Han, S., Zhang, D., and Zhang, Y. On manipulating signals of user-item graph: A jacobi polynomial-based graph collaborative filtering. *arXiv preprint arXiv:2306.03624*, 2023a.

Guo, X., Wang, Y., Du, T., and Wang, Y. Contranorm: A contrastive learning perspective on oversmoothing and beyond. *arXiv preprint arXiv:2303.06562*, 2023b.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

He, M., Wei, Z., Xu, H., et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34: 14239–14251, 2021.

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Kam, H. Superconductivty Data. UCI Machine Learning Repository, 2018. DOI: https://doi.org/10.24432/C53P47.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Levin, D. A. and Peres, Y. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Lim, T.-S. Contraceptive Method Choice. UCI Machine Learning Repository, 1997. DOI: https://doi.org/10.24432/C59W2D.

Majmundar, K., Goyal, S., Netrapalli, P., and Jain, P. Met: Masked encoding for tabular data. *arXiv preprint arXiv:2206.08564*, 2022.

Marques, A. G., Segarra, S., and Mateos, G. Signal processing on directed graphs: The role of edge directionality when processing and learning from network data. *IEEE Signal Processing Magazine*, 37(6):99–116, 2020. doi: 10.1109/MSP.2020.3014597.

Mohammad, R. and McCluskey, L. Phishing Websites. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C51W2X.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Shi, H., Gao, J., Xu, H., Liang, X., Li, Z., Kong, L., Lee, S., and Kwok, J. T. Revisiting over-smoothing in bert from the perspective of graph. *arXiv preprint arXiv:2202.08625*, 2022.

Shin, Y., Choi, J., Wi, H., and Park, N. An attentive inductive bias for sequential recommendation beyond the self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8984–8992, 2024.

Shwartz-Ziv, R. and Armon, A. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. Saint: Improved neural networks for tabular data via row attention and contrastive pretraining. *arXiv preprint arXiv:2106.01342*, 2021.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *ICML*, pp. 10347–10357. PMLR, 2021.

Ucar, T., Hajiramezanali, E., and Edwards, L. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34:18853–18865, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vurka, M. Firm-Teacher_Clave-Direction_Classification. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5GC9F.

Wang, P., Zheng, W., Chen, T., and Wang, Z. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. *arXiv preprint arXiv:2203.05962*, 2022.

Williams, V. V., Xu, Y., Xu, Z., and Zhou, R. New bounds for matrix multiplication: from alpha to omega. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2024.

Xue, F., Chen, J., Sun, A., Ren, X., Zheng, Z., He, X., Chen, Y., Jiang, X., and You, Y. A study on transformer configuration and training objective. 2023.

Yan, J., Zheng, B., Xu, H., Zhu, Y., Chen, D., Sun, J., Wu, J., and Chen, J. Making pre-trained language models great on tabular prediction. *arXiv preprint arXiv:2403.01841*, 2024.

Yeh, I.-C. default of credit card clients. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C55S3H.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Yoon, J., Zhang, Y., Jordon, J., and van der Schaar, M. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

Yu, Y.-Y., Choi, J., Cho, W., Lee, K., Kim, N., Chang, K., Woo, C., KIM, I., Lee, S., Yang, J. Y., YOON, S., and Park, N. Learning flexible body collision dynamics with hierarchical contact mesh transformer. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=90yw2uM6J5.

Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., Hou, Q., and Feng, J. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

A. Proof of Proposition 4.1

Proof. Let $\pi_v^{(t)}$ be a iterative PageRank of page v after t iterations, and $\pi_v^{(0)} = \frac{1}{N}$, where N is the total number of pages. Then, at every iteration of the algorithm, the following formula is used to compute the PageRank:

$$\pi_v^{(t)} = (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{\pi_v^{(t-1)}}{d_w}\Big) + \frac{\epsilon}{N}. \tag{16}$$

To prove that the convergence time is small, we define π_v^* as the true PageRank of v. Then we can define the total error at step t to be

$$Err(t) = \sum_v |\pi_v^{(t)} - \pi_v^*|. \tag{17}$$

Since π_v^* is the true solution, we know that it must satisfy the PageRank equations exactly:

$$\pi_v^* = (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{\pi_w^*}{d_w}\Big) + \frac{\epsilon}{N}. \tag{18}$$

To find the error, we subtract this from the iterative method equation, and optain:

$$\pi_v^{(t)} - \pi_v^* = (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{\pi_w^{(t-1)} - \pi_w^*}{d_w}\Big). \tag{19}$$

Using the Triangle Inequality, we get this experssion for the error in PageRank v at ste t:

$$|\pi_v^{(t)} - \pi_v^*| \leq (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{|\pi_w^{(t-1)} - \pi_w^*|}{d_w}\Big). \tag{20}$$

We can sum over all v to get the total error. Notice that the page w will occur d_w times on the right hand side, and since there s a d_w on the denominator, these will cancel.

$$Err(t) = \sum_v |\pi_v^{(t)} - \pi_v^*| \leq (1-\epsilon)\Big(\sum_{(w,v)\in E} |\pi_w^{(t-1)} - \pi_w^*|\Big) \tag{21}$$

We are left with $(1-\epsilon)$ times the total error at time $t-1$ on the right hand side.

$$Err(t) \leq (1-\epsilon)Err(t-1) \tag{22}$$

This shows the fast convergence, because the decease in total error is compounding. □

B. Convergence of Attention Matrix

The self-attention matrix and the PageRank matrix share one key common characteristic that their adjacency matrices are fully connected with many small values. PageRank uses a matrix of $(1-\alpha)\mathbf{A} + \alpha\frac{\mathbf{1}}{\mathbf{N}}$, where $\mathbf{A}$ is an adjacency matrix, N is the number of nodes, and α is a damping factor, and therefore, all elements have small non-zero values. In the self-attention matrix, this is also the case since Transformers use the softmax function. Because of this characteristic, in addition, PageRank converges and so does TabPSA.

The self-attention matrix is unpredictable. As long as the self-attention matrix is fully connected, however, TabPSA works. The softmax hardly produces zeros although some values are very small. Note that in PageRank, small values are also used when N is very large, i.e., a web-scale graph of billions of nodes.

We claim that an attention matrix that satisfies the three conditions converges quickly, as in Proposition 4.1. The three conditions are i) stochasticity, ii) irreducibility, and iii) aperiodicity. The first and second conditions are trivial, as attention matrices are normalized with the softmax function, and they hardly have zero values, as discussed earlier. To demonstrate

the satisfaction of the third condition, showing that self-attention matrices satisfy aperiodicity, we refer to (Levin & Peres, 2017).

Let $\mathcal{T}(x) := \{t \geq 1 : P^t(x, x) > 0\}$ be the set of times when it is possible for the chain to return to starting position x, where P is an irreducible chain if for any two states $x, y \in \mathcal{X}$ there exists an interger t such that $P^t(x, y) > 0$. The ***period*** of state x is defined to be the greatest common divisor (gcd) of $\mathcal{T}(x)$.

Lemma B.1. *If P is irreducible, then* $\gcd \mathcal{T}(x) = \gcd \mathcal{T}(y)$ *for all* $x, y \in \mathcal{X}$.

Proof. Fix two states x and y. There exist non-negative integers r and l such that $P^r(x, y) > 0$ and $P^l(x, y) > 0$. Letting $m = r + l$, we have $m \in \mathcal{T}(x) \bigcap \mathcal{T}(y)$ and $\mathcal{T}(x) \subset \mathcal{T}(y) - m$, whence $\gcd \mathcal{T}(y)$ divides all elements of $\mathcal{T}(x)$. We conclude that $\gcd \mathcal{T}(y) \leq \gcd \mathcal{T}(x)$. By an entirely parallel argument, $\gcd \mathcal{T}(x) \leq \gcd \mathcal{T}(y)$. □

For an irreducible chain, the period of the chain is defined to be the period that is common to all states. The irreducible chain will be called ***aperiodic*** if all states have a period of 1. This means a Markov chain is aperiodic if there is at least one self-loop. As discussed earlier, the self-attention matrix has non-negative values for all elements, including diagonal elements, making the self-attention matrix aperiodic.

C. Modification on the Existing Methods

TabTransformer (Huang et al., 2020) When pretraining TabTransformer-MLM, masks are applied only to the categorical features, which is impossible to do on datasets consisting solely of continuous features such as Superconductivity. Therefore, we also apply masks to continuous features and utilize a loss function that is a weighted sum of cross-entropy loss and mean squared error loss, note that we use coefficient for cross-entropy loss.

SAINT (Somepalli et al., 2021) SAINT is a Transformer-based table representation model. SAINT introduces column-wise and row-wise self-attention blocks into its Transformer architecture. The model undergoes self-supervised learning, employing techniques such as contrastive learning and denoising during pre-training. Following the pre-training phase, SAINT proceeds to fine-tune the model through supervised training on downstream tasks. For SAINT, we use the original form of the model without any modification.

MET (Majmundar et al., 2022) MET is a masked autoencoder (He et al., 2022)-based table representation learning model, which incorporates adversarial loss and reconstruction loss for self-supervised representation learning. In downstream evaluation tasks, the model omits the decoder and relies solely on the representations generated by the encoder. These representations are then used to train auxiliary small Multi-Layer Perceptron (MLP) layers to predict the classes or values of records, with the encoder held fixed. In our experiments, we fine-tune the encoder while training the auxiliary MLP layers. This approach contributes to enhancing the model's representation performance in a supervised training fashion.

D. Experimental Details

In this section, we provide details of our experiments, including datasets and baselines description, searched hyperparameters and so on.

Reproducibility Statement To reproduce the experimental results, we have made the following efforts: 1) Source codes used in the experiments are available in the supplementary material. By following the README guidance, the main results are easily reproducible. 2) All the experiments are repeated five times, and their mean and standard deviation values are reported. 3) We provide dataset and baseline details in Appendix D.

D.1. Dataset

We use 10 real-world tabular datasets. The general statistics of datasets are listed in Table 6.

- Income is a binary classification dataset used to determine whether individual earns an annual income exceeding $50K, using census data as the basis.

Table 6: Statistics of Datasets

Dataset	Task (# class)	# Features	# Continuous	# Categorical	Dataset Size	# Train set	# Valid set	# Test set
Income	Binary	14	6	8	45,222	22,632	7,530	15,060
Default	Binary	23	15	8	30,000	21,000	3,000	6,000
Phishing	Binary	19	3	16	7,032	5,450	527	1,055
Alphabank	Binary	7	1	6	30,477	21,333	4,572	4,572
Clave	Multi-class (4)	16	0	16	10,800	7,560	1,620	1,620
Contraceptive	Multi-class (3)	9	2	7	1,473	1,031	221	221
Activity	Multi-class (6)	17	15	2	6,264	4,384	846	1,034
Buddy	Multi-class (4)	9	6	3	17,357	12,149	2,084	3,124
Medicalcost	Regression	6	3	3	1,338	1,003	134	201
Superconductivity	Regression	81	81	0	21,263	14,884	2,552	3,827

- Default (Yeh, 2016) is a binary classification dataset describing data related to default payments among credit card clients in Taiwan.
- Phishing (Mohammad & McCluskey, 2015) is a binary classification dataset used to differentiate between phishing and ligitimate webpages.
- Alphabank is a binary classification dataset to determine whether the client subscribed to a long-term deposit.
- Clave (Vurka, 2015) is a multi-class classification dataset comprising binary attack-point vectors and their clave-direction class(es).
- Contraceptive (Lim, 1997) is a multi-class classification dataset used to predict a woman's choice of the current contraceptive method, taking into account her demographic and socio-economic characteristics.
- Activity (Fuller, 2020) is a multi-class classification dataset used to predict physical activity types, with indirect calorimetry serving as the reference standard.
- Buddy is a multi-class classification dataset comprising of attributes related to adoptable animals.
- Medicalcost is a regression dataset used to predict individual medical costs billed by health insurance with demographic features.
- Superconductivity (Kam, 2018) is a regression dataset encompassing 81 features derived from superconductors. Its primary objective is to predict the critical temperature.

The download links for each dataset are as follows:

- Income: `https://www.kaggle.com/lodetomasi1995/income-classification`
- Default: `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`
- Phishing: `https://archive.ics.uci.edu/ml/datasets/phishing+websites`
- Alphabank: `https://www.kaggle.com/raosuny/success-of-bank-telemarketing-data`
- Clave: `https://archive.ics.uci.edu/dataset/324/firm+teacher+clave+direction+classification`
- Contraceptive: `https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice`
- Activity: `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/ZS2Z2J`

- Buddy: https://www.kaggle.com/datasets/akash14/adopt-a-buddy
- Medicalcost: https://www.kaggle.com/mirichoi0218/insurance
- Superconductivity: https://archive.ics.uci.edu/ml/datasets/Superconductivty+Data

D.2. Baselines

To verify the effectiveness of our model, we compare our model with 10 baselines, which include deep learning models and ensemble algorithms in machine learning. For training Decision Tree, Regression and Random Forest, we use scikit-learn package.

- MLP is a abbreviation of multi-layer perceptron. We use 2-layer perceptron for baseline.
- Decision Tree partitions data into subsets based on features, creating a tree-like structure to make sequential decisions.
- Regression is a statistical method for tabular data. Note that we use logistic regression for classification task, and linear regression for regression task.
- XGBoost[4] (Chen et al., 2015) is a gradient boosting algorithm that excels in predictive modeling tasks by sequentially training decision trees to correct the errors.
- Random Forest is an ensemble machine learning algorithm that combines the predictions from multiple decision trees to improve predictive accuracy and reduce overfitting.
- TabTransformer[5] (Huang et al., 2020) is a Transformer-based model which uses a Transformer encoder to learn contextual embeddings on categorical features. Note that we use TabTranformer-MLM for self-supervised learning.
- VIME[6] (Yoon et al., 2020) proposed an semi- and self-supervised learning for tabular data with pretext task on estimating mask vectors, along with the reconstruction pretext task.
- TabNet[7] (Arik & Pfister, 2021) combines decision trees and attention mechanisms to make accurate predictions on structured datasets.
- SAINT[8] (Somepalli et al., 2021) is a Transformer-based model which utilizes contrastive learning and performs attention over both rows and columns.
- MET[9] (Majmundar et al., 2022) is a table representation model based on masked autoencoders (He et al., 2022).

D.3. Hyperparameters

TabTransformer+TabPSA We use 8 hyperparameters including depth of Transformer, embedding dimensions, learning rate, the number of heads, the value of weight decay, hidden dimension of mlp layer, polynomial type, and k. Best hyperparameters are in Table 7.

SAINT+TabPSA We use 8 hyperparameters including learning rate, embedding dimensions, the number of heads, cutmix augmentation probability p_{cutmix}, mixup parameter α, temperature parameter τ, polynomial type, and k. Best hyperparameters are in Table 8.

MET+TabPSA We use 7 hyperparameters including embedding dimensions, the number of attention heads, depth of encoder, depth of decoder, learning rate, polynomial type, and k. Best hyperparameters are in Table 9.

[4] https://github.com/dmlc/xgboost
[5] https://github.com/lucidrains/tab-transformer-pytorch
[6] https://github.com/jsyoon0823/VIME
[7] https://github.com/dreamquark-ai/tabnet
[8] https://github.com/somepago/saint
[9] https://github.com/google-research/met

Table 7: Best hyperparameters for TabTransformer+TabPSA

Datasets	Depth	Embedding dim	LR	# heads	Weight decay	Hidden dim	Polynomial Type	k
Income	6	32	2	1e-3	1e-3	128	Chebyshev	5
Default	3	32	4	2e-3	1e-3	256	Power	3
Phishing	6	32	4	1e-6	5e-3	512	Chebyshev	3
Alphabank	3	64	2	1e-3	1e-3	128	Chebyshev	3
Clave	6	64	4	1e-6	1e-3	64	Jacobi	10
Contraceptive	6	64	4	1e-7	1e-3	256	Jacobi	10
Activity	3	32	4	1e-6	1e-3	256	Legendre	3
Buddy	3	64	1	1e-6	5e-3	64	Legendre	3
Medicalcost	3	32	4	1e-6	5e-3	512	Chebyshev	3

Table 8: Best hyperparameters for SAINT+TabPSA

Datasets	LR	Embedding dim	# heads	p_{cutmix}	α	τ	Polynomial type	k
Income	8e-4	32	4	0.1	1.0	0.5	Legendre	5
Default	8e-4	32	4	0.1	1.0	0.5	Power	5
Phishing	1e-5	16	4	0.1	10.0	0.7	Chebyshev	5
Alphabank	8e-4	16	4	0.1	10.0	0.5	Power	5
Clave	8e-4	32	4	0.1	10.0	0.5	Legendre	10
Contraceptive	8e-4	16	4	0.1	10.0	0.5	Chebyshev	5
Activity	8e-4	32	4	0.1	1.0	0.5	Legendre	3
Buddy	5e-3	32	4	0.1	0.1	0.5	Power	3
Medicalcost	8e-4	16	4	0.1	10.0	0.7	Power	5
Superconductivity	8e-4	8	4	0.1	1.0	0.3	Legendre	3

Table 9: Best hyperparameters for MET+TabPSA

Datasets	Embedding dim	# heads	Encoder depth	Decoder depth	LR	Polynomial type	k
Income	8	8	3	3	8e-4	Power	5
Default	4	2	3	3	8e-4	Chebyshev	10
Phishing	64	4	3	3	5e-3	Jacobi	3
Alphabank	8	8	6	3	1e-3	Legendre	3
Clave	4	4	9	9	8e-4	Jacobi	3
Contraceptive	16	2	3	3	8e-4	Legendre	3
Activity	16	4	6	3	1e-3	Chebyshev	5
Buddy	32	2	3	3	5e-3	Power	3
Medicalcost	32	2	6	9	8e-4	Legendre	3
Superconductivity	16	4	6	3	8e-4	Legendre	3

E. Computational Efficiency Analysis

Tables 10 and 11 present the evaluation results for the computational efficiency of applying our method to Transformer-based models, detailing the results for each model across all datasets. Table 10 measures the wall-clock training time on five occasions and reports the average. Table 11 documents the memory usage required to train Transformer-based models with TabPSA.

Table 10: Wall-clock training time per epoch in seconds (↓) for all datasets

Methods	**Binary Classification**				**Multi-class Classification**				**Regression**	
	Income	Default	Phishing	Alphabank	Clave	Contra.	Activity	Buddy	Medical.	Super.
TabTransformer	4.10s	4.03s	1.03s	2.49s	1.03s	0.67s	0.92s	1.67s	0.59s	-
TabTrans.+TabPSA	4.16s	4.04s	1.06s	2.52s	1.07s	0.69s	0.97s	1.72s	0.63s	-
SAINT	5.49s	6.79s	1.57s	5.03s	1.88s	0.51s	1.42s	2.38s	0.41s	12.57s
SAINT+TabPSA	6.77s	7.64s	2.06s	6.31s	2.39s	0.58s	1.89s	4.00s	0.45s	13.43s
MET	3.89s	3.51s	1.07s	3.51s	1.40s	0.26s	0.92s	2.13s	0.38s	3.99s
MET+TabPSA	4.98s	3.63s	1.12s	3.56s	1.55s	0.29s	0.96s	2.20s	0.46s	4.44s

Table 11: GPU memory usage for training Transformers equipped with TabPSA in MB (↓) for all datasets

Methods	**Binary Classification**				**Multi-class Classification**				**Regression**	
	Income	Default	Phishing	Alphabank	Clave	Contra.	Activity	Buddy	Medical.	Super.
TabTransformer	19.08MB	19.33MB	20.98MB	18.77MB	20.92MB	18.75MB	18.40MB	18.69MB	18.45MB	-
TabTrans.+TabPSA	19.13MB	19.37MB	21.02MB	18.82MB	20.96MB	18.80MB	18.41MB	18.73MB	18.49MB	-
SAINT	86.42MB	177.25MB	128.75MB	37.70MB	101.41MB	48.59MB	110.29MB	48.83MB	34.60MB	1,736.35MB
SAINT+TabPSA	86.43MB	177.94MB	129.19MB	37.72MB	102.30MB	48.60MB	110.33MB	48.84MB	34.62MB	1,736.93MB
MET	132.22MB	316.09MB	54.72MB	140.28MB	121.37MB	157.64MB	120.91MB	56.55MB	85.72MB	1,552.95MB
MET+TabPSA	136.40MB	337.28MB	55.67MB	241.22MB	127.45MB	158.59MB	125.23MB	56.84MB	85.77MB	1,693.33MB

F. Additional Experiments

F.1. Evidence of the Proposed Mechanism in Other Domain

We integrated TabPSA with BERT$_{base}$ and conducted text classification tasks on the GLUE benchmark. The models were fine-tuned for 5 epochs with our self-attention layer, and a slight modification was introduced in the initialization of coefficients — specifically, $w_0 = 0$, $w_1 = 1$, and $w_i = 0$ for $i \geq 2$. The reason for this modification is that the pre-trained BERT$_{\text{base}}$ is trained with the original self-attention mechanism, which is the multiplication of the attention matrix and the value matrix. As shown in Table 12, we find that the proposed self-attention layer also improves transformer-based model for language modeling.

Table 12: Performance comparison of BERT$_{\text{base}}$ and BERT$_{\text{base}}$ + TabPSA across various datasets

Methods	CoLA	SST-2	MRPC	QQP	STS-B	MNLI-m	MNLI-mm	QNLI	RTE	Avg.
BERT$_{\text{base}}$	56.79	93.81	88.70	88.32	88.16	84.96	84.15	91.63	66.06	82.51
BERT$_{\text{base}}$ + TabPSA	59.43	93.58	91.78	88.46	89.10	82.19	83.08	91.67	69.68	83.22

F.2. Experiment on Recent Transformer-based Model for Tables

We summarize the performance comparison of TP-BERTa (Yan et al., 2024) and TP-BERTa + TabPSA in Table 13. TP-BERTa is a more recent Transformer-based model for tabular data, specifically pre-trained for table prediction. The results show a consistent performance improvement with TabPSA, as demonstrated thus far.

Table 13: Performance comparison of TP-BERTa and TP-BERTa + TabPSA across various datasets

Models	Parkinsons	Phishing	Contraceptive	Wine	Medicalcost
TP-BERTa	96.34±1.57	82.98±0.36	74.05±1.38	0.90±0.05	0.85±0.00
TP-BERTa + TabPSA	98.14±1.33	83.58±0.23	76.14±0.97	0.95±0.02	0.85±0.01
Increase (%)	1.86%	0.72%	2.83%	6.18%	0.31%

F.3. Sensitivity Analyses with Regularization on Polynomial Coefficient

For further analysis of sensitivity with respect to k, we conducted the same experiment as in Table 2, but with regularization on the polynomial coefficient w. This experiment aims to check for symptoms of overfitting in the coefficients. The experimental results shown in Table 14 suggest that training without regularization introduces mild overfitting to the coefficients.

Table 14: Sensitivity analyses with regularization over coefficient w

Datasets	k	TabTransformer+TabPSA	SAINT+TabPSA	MET+TabPSA
Alphabank	2	62.1±0.16	62.2±0.22	61.6±1.08
	3	62.1±0.19	62.4±0.52	61.2±0.62
	5	62.1±0.22	62.4±0.76	61.0±0.83
	10	62.1±0.20	62.5±0.50	60.3±1.56
Contraceptive	2	65.2±1.44	75.5±0.81	77.1±1.84
	3	65.6±1.80	76.1±0.36	76.5±0.97
	5	65.7±2.34	75.3±0.47	77.3±2.62
	10	66.6±0.94	75.9±0.65	77.4±0.83
Medicalcost	2	0.61±0.02	0.87±0.00	0.74±0.11
	3	0.61±0.04	0.86±0.01	0.87±0.00
	5	0.61±0.05	0.86±0.00	0.86±0.01
	10	0.61±0.26	0.86±0.00	0.87±0.01