A PROOF OF THEOREM 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:

$$\pi_v^{(t)} = (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{\pi_v^{(t-1)}}{d_w}\Big) + \frac{\epsilon}{N}. \tag{9}$$

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

$$Err(t) = \sum_v |\pi_v^{(t)} - \pi_v^*|. \tag{10}$$

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

$$\pi_v^* = (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{\pi_w^*}{d_w}\Big) + \frac{\epsilon}{N}. \tag{11}$$

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

$$\pi_v^{(t)} - \pi_v^* = (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{\pi_w^{(t-1)} - \pi_w^*}{d_w}\Big). \tag{12}$$

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

$$|\pi_v^{(t)} - \pi_v^*| \leq (1-\epsilon)\Big(\sum_{(w,v)\in E} \frac{|\pi_w^{(t-1)} - \pi_w^*|}{d_w}\Big). \tag{13}$$

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.

$$Err(t) = \sum_v |\pi_v^{(t)} - \pi_v^*| \leq (1-\epsilon)\Big(\sum_{(w,v)\in E} |\pi_w^{(t-1)} - \pi_w^*|\Big) \tag{14}$$

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

$$Err(t) \leq (1-\epsilon)Err(t-1) \tag{15}$$

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 CheAtt.

The self-attention matrix is unpredictable. As long as the self-attention matrix is fully connected, however, CheAtt 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 Theorem 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. For a better representation, we also incorporate linear layers as embedding layers for continuous features and utilize both continuous and categorical features as inputs for the Transformer layers.

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., 2022a)-based table representation learning model. 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.

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.
- News (Kelwin et al., 2015) is a regression dataset comprising a diverse range of attributes related to articles published by Mashable. It is aimed at predicting the quantity of shares these articles receive on social networks.
- 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
- News: https://archive.ics.uci.edu/ml/datasets/online+news+popularity
- 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[3] (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[4] (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[5] (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[6] (Arik & Pfister, 2021) combines decision trees and attention mechanisms to make accurate predictions on structured datasets.
- SAINT[7] (Somepalli et al., 2021) is a Transformer-based model which utilizes contrastive learning and performs attention over both rows and columns.
- MET[8] (Majmundar et al., 2022) is a table representation model based on masked autoencoders He et al. (2022a).

D.3 HYPERPARAMETERS

TabTransformer+CheAtt We use 7 hyperparameters including depth of Transformer, embedding dimensions, learning rate, the number of heads, the value of weight decay, coefficient for cross entropy loss and hidden dimension of mlp layer. Best hyperparameters are in Table 7.

Table 7: Best hyperparameters for TabTransformer+CheAtt

Datasets	depth	embedding dim	lr	# heads	weight decay	coef.	hidden dim
Income	3	32	1e-3	4	1e-3	0.3	32
Default	6	16	1e-3	8	1e-5	0.3	512
Phishing	3	16	1e-3	8	1e-3	0.3	32
Alphabank	6	64	1e-5	4	1e-5	0.5	64
Clave	6	64	1e-3	8	1e-9	0.1	1024
Contraceptive	6	32	1e-3	4	1e-5	0.5	256
Activity	6	64	1e-3	8	1e-5	0.1	256
Buddy	9	64	1e-3	4	1e-3	0.5	512
Medicalcost	6	64	1e-3	4	1e-5	0.3	256
Superconductivity	6	32	1e-4	8	1e-5	0.1	32

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

SAINT+CheAtt We use 6 hyperparameters including learning rate, embedding dimensions, the number of heads, cutmix augmentation probability p_{cutmix}, mixup parameter α, and temperature parameter τ. Best hyperparameters are in Table 7.

Table 8: Best hyperparameters for SAINT+CheAtt

Datasets	lr	embedding dim	# heads	p_{cutmix}	α	τ
Income	1e-3	32	4	0.1	0.1	0.7
Default	1e-3	32	4	0.1	0.6	0.7
Phishing	1e-5	12	6	0.1	0.3	0.7
Alphabank	1e-3	32	4	0.01	1.0	0.1
Clave	1e-3	32	4	0.1	0.3	0.7
Contraceptive	1e-3	16	1	0.1	0.1	0.1
Activity	1e-3	32	1	0.1	0.6	0.1
Buddy	1e-3	12	8	0.1	0.3	0.7
Medicalcost	1e-3	16	2	0.1	0.6	0.1
Superconductivity	1e-4	8	6	0.1	0.3	0.7

MET+CheAtt We use 7 hyperparameters including embedding dimensions, the number of attention heads, depth of encoder, depth of decoder, percentage of mask, learning rate and k. Best hyperparameters are in Table 7.

Table 9: Best hyperparameters for MET+CheAtt. Mask pct. means masking percentage of input.

Datasets	embedding dim	# heads	encoder depth	decoder depth	mask pct.	lr	k
Income	64	8	3	9	80	1e-3	3
Default	64	8	3	15	70	1e-3	2
Phishing	32	8	6	6	80	1e-3	2
Alphabank	32	2	3	9	80	1e-3	2
Clave	128	4	6	15	80	1e-3	2
Contraceptive	128	8	3	15	70	1e-3	10
Activity	64	2	6	9	80	1e-3	10
Buddy	32	8	3	3	80	1e-3	3
Medicalcost	64	4	3	12	80	1e-3	2
Superconductivity	32	2	3	3	80	1e-3	2

E Empirical Runtime Analysis

Table 10 and 11 show runtime evaluation results of applying our method to Transformer-based models, along with the results for each individual model for all datasets. We measure wall clock training time and wall clock time for generating output representation from test data for 5 times and report the average.

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	4.67s	4.54s	1.57s	4.24s	1.78s	0.70s	1.31s	2.78s	0.50s	8.62s
TabTransformer+CheAtt	5.65s	5.72s	1.95s	5.01s	2.45s	0.73s	1.68s	3.14s	0.64s	8.74s
SAINT	6.28s	7.48s	2.05s	5.27s	2.26s	0.67s	1.89s	2.84s	0.51s	14.20s
SAINT+CheAtt	7.56s	9.02s	2.28s	6.29s	2.63s	0.70s	2.06s	4.46s	0.55s	17.17s
MET	4.32s	4.33s	1.43s	3.98s	2.05s	0.32s	1.23s	2.97s	0.32s	5.83s
MET+CheAtt	4.58s	5.61s	1.70s	4.21s	2.08s	0.35s	1.51s	3.41s	0.52s	9.46s

Table 11: Wall clock time for generating 1,000 representations from data in milliseconds (↓) for all datasets

Methods	**Binary Classification**				**Multi-class Classification**				**Regression**	
	Income	Default	Phishing	Alphabank	Clave	Contra.	Activity	Buddy	Medical.	Super.
TabTransformer	8.58ms	5.99ms	8.08ms	8.90ms	7.93ms	9.26ms	9.42ms	7.68ms	7.46ms	7.09ms
TabTransformer+CheAtt	9.75ms	7.61ms	9.26ms	10.03ms	9.73ms	10.38ms	12.16ms	9.84ms	9.97ms	8.65ms
SAINT	1.88ms	1.66ms	2.75ms	1.97ms	3.38ms	4.04ms	2.20ms	2.25ms	4.49ms	1.75ms
SAINT+CheAtt	2.59ms	2.27ms	3.55ms	2.38ms	3.77ms	5.23ms	2.78ms	2.35ms	5.60ms	2.36ms
MET	3.22ms	2.95ms	2.43ms	3.50ms	2.36ms	2.50ms	2.36ms	2.32ms	2.59ms	2.79ms
MET+CheAtt	3.50ms	3.71ms	3.25ms	3.55ms	3.14ms	3.40ms	3.18ms	3.22ms	3.37ms	3.63ms