T2G-FORMER: Organizing Tabular Features into Relation Graphs Promotes Heterogeneous Feature Interaction

Jiahuan Yan[1*], **Jintai Chen**[1*], **Yixuan Wu**[2], **Danny Z. Chen**[3], **Jian Wu**[4†]
[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]School of Medicine, Zhejiang University, Hangzhou, China
[3]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
[4]The First Affiliated Hospital, and Department of Public Health, Zhejiang University School of Medicine, Hangzhou, China
jyansir@zju.edu.cn, jtchen721@gmail.com, wyx_chloe@zju.edu.cn, dchen@nd.edu, wujian2000@zju.edu.cn

Abstract

Recent development of deep neural networks (DNNs) for tabular learning has largely benefited from the capability of DNNs for automatic feature interaction. However, the heterogeneity nature of tabular features makes such features relatively independent, and developing effective methods to promote tabular feature interaction still remains an open problem. In this paper, we propose a novel *Graph Estimator*, which automatically estimates the relations among tabular features and builds graphs by assigning edges between related features. Such relation graphs organize independent tabular features into a kind of graph data such that interaction of nodes (tabular features) can be conducted in an orderly fashion. Based on our proposed *Graph Estimator*, we present a bespoke Transformer network tailored for tabular learning, called T2G-FORMER, which processes tabular data by performing tabular feature interaction guided by the relation graphs. A specific *Cross-level Readout* collects salient features predicted by the layers in T2G-FORMER across different levels, and attains global semantics for final prediction. Comprehensive experiments show that our T2G-FORMER achieves superior performance among DNNs and is competitive with non-deep Gradient Boosted Decision Tree models. The code and detailed results are available at https://github.com/jyansir/t2g-former.

Introduction

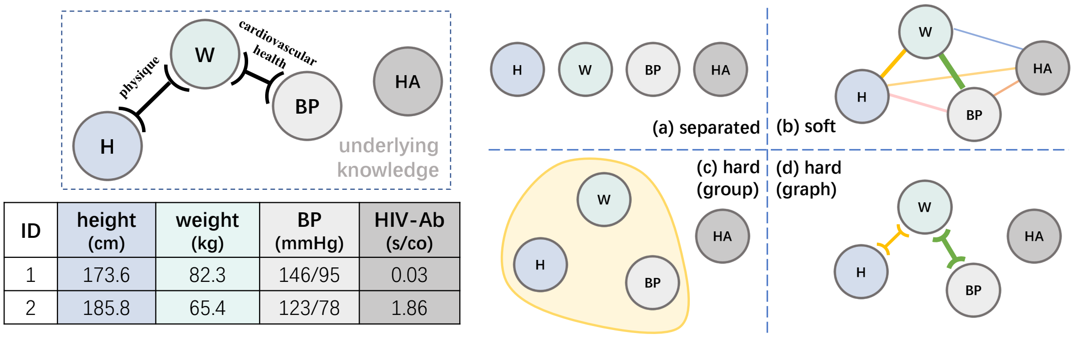
Data in the form of table structures are ubiquitous in many fields, e.g., medical records (Johnson, Pollard et al. 2016; Hassan, Al-Insaif et al. 2020) and click-through rate (CTR) prediction (Covington, Adams, and Sargin 2016; Song et al. 2019). It was observed that Gradient Boosted Decision Trees (GBDT) (Chen and Guestrin 2016; Ke, Meng et al. 2017; Prokhorenkova, Gusev et al. 2018) were dominating models for tabular data tasks in machine learning and industrial applications. Due to big successes of deep neural networks (DNNs) in various fields, there has been increasing development of specialized DNNs for tabular data learning (Popov, Morozov, and Babenko 2019; Arik and Pfister 2021; Wang, Shivanna et al. 2021; Gorishniy, Rubachev et al. 2021; Chen, Liao et al. 2022). Such studies either leveraged ensembling of neural networks (Popov, Morozov, and Babenko 2019; Arik and Pfister 2021; Katzir, Elidan, and El-Yaniv 2020) to build *differentiable tree models*, or explored diverse interaction approaches (Guo, Tang et al. 2017; Wang, Fu et al. 2017; Song et al. 2019; Wang, Shivanna et al. 2021; Gorishniy, Rubachev et al. 2021; Chen, Liao et al. 2022) to learn comprehensive features by fusing different tabular features.

However, different from images and texts, it is challenging for fusion-based models to handle tabular feature interaction due to the feature heterogeneity problem (Borisov et al. 2021). DANets (Chen, Liao et al. 2022) suggested the “selection & abstraction” principle that processes tabular data by first selecting and then interacting the selected features. Known neural feature selection schemes can be categorized into soft and hard versions. The soft selection essentially exerts fully connected interactions among features (see Fig. 1(b)), such as multiplicative interaction (Guo, Tang et al. 2017), feature crossing (Wang, Fu et al. 2017; Wang, Shivanna et al. 2021), and attention-based interaction (Song et al. 2019; Huang et al. 2020; Gorishniy, Rubachev et al. 2021). However, tabular features by nature are heterogeneous, and fully connected interaction is a suboptimal choice since it blindly fuses all features together. DANets (Chen, Liao et al. 2022) performed hard selection by grouping correlative features and then constraining interactions among grouped features (Fig.1(c)). Although DANets achieved promising results, its feature selection operation cannot thoroughly address intra-group interactions (see Fig. 1(c)), and thus features assigned in a same group are indiscriminately fused, making the model inferiorly expressive.

There are numerous daily applications that exemplify the significance of selective interaction for heterogeneous tabular features. The left part of Fig. 1 gives an example of a medical data table. Using underlying medical knowledge, a static graph can be formed to indicate relations of reasonable feature pairs. For instance, the relation of *height* and *weight* gives a probability representing a high-level semantic *physique*. Also, the relation between *weight* and *blood pressure* (*BP*) is likely to indicate a semantic *cardiovascular health*. Besides, there might be some “inert features” that are unrelated to any other features, such as the features rep-

[*] These authors contributed equally.

[†] The corresponding author.



ID	height (cm)	weight (kg)	BP (mmHg)	HIV-Ab (s/co)
1	173.6	82.3	146/95	0.03
2	185.8	65.4	123/78	1.86

Figure 1: An example of medical data tables. The values in different columns are located in heterogeneous feature spaces. Underlying medical knowledge sparsely links feature pairs. (a) Original separated features without any interactions, which are often used in non-deep models. (b) Fully connected interactions by softly selecting all the features. (c) Selective interactions among grouped features by hard selection. (d) Selective interactions according to a weighted relation graph. “BP” denotes blood pressure; “HIV-Ab” indicates the level of HIV antibody.

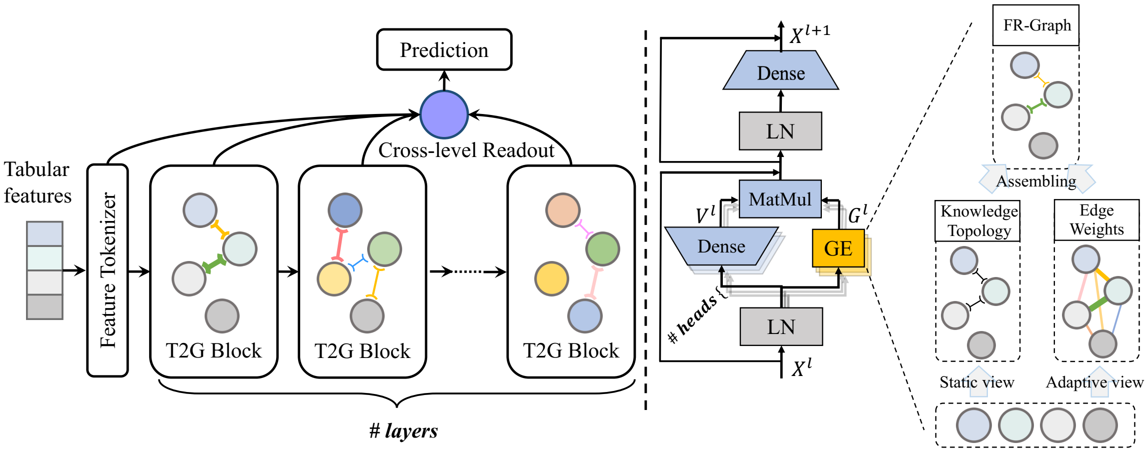
resenting the *level of HIV antibody* (*HIV-Ab*). In the right part, Fig. 1(a) presents the original tabular features whose relations are not specified, and higher-level semantics cannot be directly obtained if the feature relations are not determined. Fig. 1(b) illustrates the fully connected interactions of soft selection, which may introduce some noisy relations in feature fusion (e.g., the “inert feature” connects with the other features). Hard selection with a grouping operation (e.g., used in DANets) achieves partially selective interactions by grouping related features (see Fig. 1(c)), but is still likely to include noisy interactions. It can only group related features but fails to handle the feature relations within the same group. In Fig. 1(c), the grouping design can only put the features *height*, *weight*, and *BP* together for mutual interactions, but cannot exclude the meaningless *height-BP* pair. It is intuitive that a precise health condition assessment can be made based on both the data-specific record values (e.g., 173.6 cm for *height* in Fig. 1) and the underlying knowledge represented by the edges of the relation graph. For the first sample in the medical table (ID = 1), considering the values of *height* and *weight* jointly can suggest a symptom of overweight. Similarly, combining the values of *weight* and *BP* indicates a risk of cardiovascular problems. The second sample (ID = 2) directly indicates a risk of HIV infection solely based on the feature of *HIV-Ab*. Hence, we argue that an ideal way to handle such complex decision processes is to build a graph with adaptive edge weights. The edge weights (represented by different colors and widths in Fig. 1(d)) indicate the strengths of relations based on specific feature values, and the static graph topology represents the underlying knowledge to constrain meaningful relations.

Inspired by the above observations, in this paper, we propose to build graphs for tabular features to guide feature interaction. We develop a novel *Graph Estimator* (GE) for organizing independent tabular features into a feature relation graph (FR-Graph). Further, we present a bespoke Transformer network for tabular learning, called T2G-FORMER, by stacking GE-incorporated blocks for selective feature interaction. GE models an FR-Graph by assembling (i) a static graph topology depicting underlying knowledge of the task and (ii) data-adaptive edge weights for graph edges. The static graph depicts the underlying knowledge (the relations of feature pairs), while the data-adaptive edge weights represent the strengths of relations based on specific feature values. Using the FR-Graph, we can effectively capture more subtle interactions which may be mishandled by grouping strategies (as shown in Fig. 1(c)). In our proposed T2G-FORMER, each layer employs the FR-Graph to transform layer input features into graph data, and heterogeneous feature interactions are performed in an orderly fashion based on the specification of graph edges. Besides, a special *Cross-level Readout* collects salient features from each level and attains global tabular semantics for the final prediction.

The workflow of T2G-FORMER proceeds as follows. An FR-Graph, whose edges represent the static relations of features with data-adaptive weights (predicted by the GE module), guides the processing of the tabular feature interaction to predict higher-level features. Then another FR-Graph for higher-level tabular features is built to organize the feature interaction, and the process continues. T2G-FORMER can output comprehensive semantics from different feature levels by repeating the above process. The shared *Cross-level Readout* is used to aggregate semantics from different feature levels, and takes all these features into consideration in the final prediction.

Overall, the main contributions of our work are as follows:

- We first utilize feature relation graphs to handle heterogeneous feature interaction for tabular data, and propose a novel *GE* module for feature relation organization.
- We adapt feature relation graphs in the Transformer architecture, and build a specialized tabular learning Transformer T2G-FORMER for tabular classification and regression.
- Comprehensive experiments show that T2G-FORMER consistently outperforms state-of-the-art tabular DNNs on many datasets, and is competitive with GBDTs.



(a) T2G-Former Architecture

(b) Basic Block & GE Workflow

Figure 2: (a) The architecture of T2G-Former for tabular learning. Each T2G block builds an FR-Graph for a feature level and performs selective interaction. A global readout node collects salient features from each layer to form tabular semantics. (b) Illustrating a basic block in Sec. and GE in Sec. .

Related Work

DNNs for Tabular Learning

Tabular learning refers to machine learning applications on tabular data that conducts prediction based on categorical or continuous features (Dong, Cheng et al. 2022). Classical non-deep methods (Li et al. 1984; Friedman 2001; Zhang and Honavar 2003; Zhang, Kang et al. 2006; He, Pan et al. 2014) are prevalent choices for such tasks (Anghel et al. 2018), especially the ensemble methods of decision trees, such as GBDT (Friedman 2001), XGBoost (Chen and Guestrin 2016), LightBGM (Ke, Meng et al. 2017), and CatBoost (Prokhorenkova, Gusev et al. 2018).

Compared to their shallow counterparts, DNNs enjoy strong abilities of automatic feature learning (Thawani, Pujara et al. 2021), and hence offer a good potential to exploit hidden features. Recently, increasingly more studies applied DNNs to tabular data (Guo, Tang et al. 2017; Yang, Morillo, and Hospedales 2018; Song et al. 2019; Feng, Yu, and Zhou 2018; Hazimeh et al. 2020; Popov, Morozov, and Babenko 2019; Arik and Pfister 2021; Chen, Liao et al. 2022), which can be roughly categorized into **differentiable tree models** and **fusion-based models**.

Differentiable Tree Models. DNNs of this type (Popov, Morozov, and Babenko 2019; Arik and Pfister 2021; Katzir, Elidan, and El-Yaniv 2020) were inspired by the successes of the ensemble tree frameworks (Kontschieder, Fiterau et al. 2015; Feng, Yu, and Zhou 2018; Yang, Morillo, and Hospedales 2018). NODE (Popov, Morozov, and Babenko 2019) combined differentiable oblivious decision trees (Lou and Obukhov 2017) with multi-layer hierarchical representations and achieved competitive performances as GBDT. TabNet (Arik and Pfister 2021) employed an attention mechanism (Vaswani, Shazeer et al. 2017) to sequentially select salient features for tree-like decision. Net-DNF (Katzir, Elidan, and El-Yaniv 2020) introduced bias of a disjunctive normal form to select and aggregate feature subsets in each block. NODE and Net-DNF largely benefited from model ensembles but did not take advantage of the feature representation capability of DNNs (Chen, Liao et al. 2022). TabNet designed non-interactive transformer blocks for feature representation and selection without feature fusion. All these DNNs function as feature selectors and splitters, but neglect underlying interactions among tabular features.

Fusion-based Models. Fusion-based models (Guo, Tang et al. 2017; Song et al. 2019; Huang et al. 2020; Wang, Shivanna et al. 2021; Gorishniy, Rubachev et al. 2021; Chen, Liao et al. 2022) leveraged DNNs to fuse higher-level features via feature interaction. DeepFM (Guo, Tang et al. 2017) performed multiplicative interaction on encoded features for CTR prediction. DCN (Wang, Fu et al. 2017; Wang, Shivanna et al. 2021) combined DNNs with cross components to learn complex features with high-order interactions. Recently, attention module (Vaswani, Shazeer et al. 2017) became a popular choice due to its interactive bias and remarkable performance (Kenton and Toutanova 2019; Dosovitskiy, Beyer et al. 2020). AutoInt (Song et al. 2019) used multi-head self-attention to interact low-dimension embedded features. TabTransformer (Huang et al. 2020) directly transferred Transformer (Vaswani, Shazeer et al. 2017) blocks to tabular data but neglected interaction between categorical features and continuous ones. FT-Transformer (Gorishniy, Rubachev et al. 2021) addressed this problem by tokenizing these two types of features and processing them equally. DANets (Chen, Liao et al. 2022) selected correlative tabular features and attentively fused the selected features into higher-level ones.

Tabular Feature Interaction

Most of the previous fusion-based work simply transferred successful neural architectures (e.g., MLP (Guo, Tang et al. 2017), self-attention (Song et al. 2019), and Transformer (Huang et al. 2020; Gorishniy, Rubachev et al. 2021)) into tabular data and interacted features with soft selection. However, feature heterogeneity (Borisov et al. 2021; Popov, Morozov, and Babenko 2019) led to gap of inductive bias and made these models (which were designed for homogeneous data, e.g., images and texts) sub-optimal. DANets (Chen, Liao et al. 2022) first adapted selective feature interaction by hard selection, constraining interactions in a feature group, and achieved promising results; but, relations of intra-group features were still not managed well. Hence, this paper proposes feature relation graphs and adapts them into a tailored Transformer network.

Graph Estimator

We propose *Graph Estimator* (GE) (Fig. 2(b)) for automatically building ***Feature Relation Graphs*** **(FR-Graphs)**, which treats tabular features as nodes in a graph and estimates the feature relations as edges. The GE design is inspired by knowledge graph completion (KGC) (Shi and Weninger 2018; Wu et al. 2021) that might use semantical similarity of two entities to estimate their relation plausibility. A basic form to measure semantical similarity (Nickel et al. 2011) is:

$$f_r(h,t) = h^T M_r t, \tag{1}$$

where $h, t \in \mathbb{R}^n$ are an encoded head entity node and a tail one, and a learnable matrix $M_r \in \mathbb{R}^{n\times n}$ represents relation r in a knowledge graph (KG). Various following methods (Yang et al. 2015; Trouillon, Welbl et al. 2016; Nickel, Rosasco, and Poggio 2016) followed this idea, which differed from one another solely in relation embeddings and score functions.

Different from KGC models that only compute static relation plausibility for entities, GE estimates the feature relations by a static underlying graph topology with data-adaptive edge weights. We take each tabular feature as a node, and first perform semantic matching to estimate the soft plausibility of pair-wise interactions between tabular features, which are referred to as **data-adaptive edge weights** in this section. Second, a **static knowledge topology** is learned based on tabular column semantics to preserve interactions of salient feature pairs. At the end, edge weights are assembled with the knowledge topology to form an FR-Graph.

FR-Graph Structure Components

To mine the relations among tabular features, we build FR-Graph by treating tabular features as graph node candidates and predicting the edges among them. The edges were yielded from two perspectives: adaptive edge weights representing data-specific information, and static edge topology for all the data representing the underlying knowledge. Note that some features are isolated from the FR-Graph if no other nodes connected with them.

Adaptive Edge Weights. Given two tabular feature embedding vectors $x_i, x_j \in \mathbb{R}^n$ $(i, j \in \{1, 2, \dots, N\})$, where N is the number of input features (table columns), we evaluate their interaction plausibility using the following pair-wise score function:

$$G_w[i,j] = g_w(f_i^h, f_j^t) = {f_i^h}^T \mathrm{diag}(r) f_j^t, \tag{2}$$

$$f_i^h = W^h x_i, f_i^t = W^t x_i, \begin{cases} W^h \equiv W^t & \text{if symmetric,} \\ W^h \neq W^t & \text{if asymmetric,} \end{cases} \tag{3}$$

where two learnable parameters $W^h, W^t \in \mathbb{R}^{m\times n}$ denote transformations for a head feature and a tail one, and $\mathrm{diag}(r) \in \mathbb{R}^{n\times n}$ is a diagonal matrix parameterized by learnable relation vectors $r \in \mathbb{R}^n$ that semantically represent feature interaction relations. Here W^h and W^t share parameters if the pair-wise feature edge weights are symmetric (i.e., $G_w[i,j] \equiv G_w[j,i]$) and are parameter-independent in the asymmetric case (i.e., $G_w[i,j] \neq G_w[j,i]$). All bias vectors are omitted for notation brevity. Consequently, the adaptive weight scores g_w of all feature pairs constitute a fully connected weighted relation graph G_w. Note that the edge weight score is degraded to an attention score when r is filled with scalar value 1 (and $\mathrm{diag}(r)$ becomes an entity matrix), and thus it is able to measure weighted feature similarity.

Static Knowledge Topology. Although we introduce soft edge weights for all feature pairs, it is also important to globally consider the underlying knowledge of the tabular data. Thus, we use a series of column embeddings to represent the semantics of the tabular features, and a static relation topology score can be computed as follows:

$$G_t[i,j] = g_t(e_i^h, e_j^t) = \frac{{e_i^h}^T e_j^t}{\|e_i^h\|_2 \|e_j^t\|_2}, \tag{4}$$

$$e_i^h = E^h[:, i], \; e_i^t = E^t[:, i],$$

where $E \in \{E^h, E^t\}$ is learnable column embeddings categorized into the head view or tail view, $E = (e_1, e_2, \dots, e_N) \in \mathbb{R}^{d\times N}$, and d is the embedding dimension. Similarly, the relation topology score g_t has the symmetric and asymmetric counterparts, and E^h and E^t share parameters in the symmetric relation topology (i.e., $G_t[i,j] \equiv G_t[j,i]$) but are parameter-independent in the asymmetric case (i.e., $G_t[i,j] \neq G_t[j,i]$). We use L_2 normalization in the g_t score function to transform embeddings to be on a similar scale and improve the training stability.

We generate static relation topology based on the G_t scores in Eq. (4), as:

$$A = f_{top}(G_t) = \mathbb{1}\left[\sigma_1(G_t + b) > T\right], \tag{5}$$

where σ_1 is an element-wise activation parameterised by a learnable bias b (like the operation in PReLU (He, Zhang et al. 2015)), G_t is adjacency matrix scores composed of the relation topology score g_t, T is a constant threshold for signal clipping, and $\mathbb{1}$ denotes the indicator function. In this way, we obtain a global graph topology (an adjacency matrix A) to constrain feature interactions, and this topology can be regarded as static knowledge on the whole task.

Relation Graph Assembling

As we obtain "soft" adaptive edge weights from the data view and "hard" static relation graph topology from the knowledge view, we combine them to generate an FR-Graph, following the idea of "decision on both specific data and underlying knowledge". Specifically, we assemble the two components as follows:

$$G = \sigma_2(f_{\text{nsi}}(A) \odot G_w), \tag{6}$$

where σ_2 is a competitive activation (e.g., L_p normalization, softmax, entmax, sparsemax (Martins et al. 2016)) to restrict the indegree of each "feature node", and $\odot$ denotes the Hadamard product. The resulted relation graph G is a weighted graph based on both adaptive feature matching and static knowledge topology. To help the FR-Graph focus on learning meaningful interactions between different features, a "no-self-interaction" function f_{nsi} is performed to explicitly exclude self-loops in G. We use the FR-Graph to instruct subsequent feature interactions. Since both the edge weights and knowledge topology have the symmetric and asymmetric versions, there are four combinations of FR-Graph covering the complete relation graph. In experiments, we will further discuss the impact of the FR-Graph type.

T2G-FORMER

We incorporate GE into the attention-like basic block, and build T2G-FORMER by stacking multiple blocks for selective tabular feature interaction (see Fig. 2). T2G-FORMER uses estimated FR-Graphs to interact features and attain higher-level features layer by layer. A *Cross-level Readout* is sequentially transformed to the feature space of each layer, and selectively collects salient features for the final prediction. A shortcut path is added to preserve the information from the preceding layers, resulted in gated fusion in different feature levels that promotes the model capability.

Basic Block

A single block is built equipped with GE for selective feature interaction (see Fig. 2(b)). Given input features $X^l \in \mathbb{R}^{n \times N}$ to the l-th layer, we obtain higher-level features X^{l+1} as follows:

$$G^l = GE(X^l), \;\; V^l = W_v X^l, \tag{7}$$

$$H^l = G^l V^l + g(X^l), \;\; X^{l+1} = \text{FFN}(H^l) + g(H^l), \tag{8}$$

where $W_v \in \mathbb{R}^{m \times n}$ is learnable parameters for feature transformation, and V^l is transformed input features. FFN denotes a feed-forward network. As self-interaction is excluded in G^l (see Eq. (6)), a shortcut path g is added to protect the information from the preceding layers, which is a simple *dropout* layer in experiments. Notably, we yield and use the FR-Graph for feature interactions, and does not influence the intra-feature update conducted by the shortcut. In the first layer, we set X^0 as the input tabular data encoded by a simple feature tokenizer (Gorishniy, Rubachev et al. 2021). In this way, higher-level features can be iteratively obtained with the generated FR-Graphs and selective interaction. In implementation, layer normalization is performed (see Fig. 2(b)) for stable training.

Cross-level Readout

We design a global readout node to selectively collect salient features from each layer and attain comprehensive semantics for the final prediction. Specifically, we attentively fuse selected features at the current layer and combine them with the lower-level features from the preceding layers by a shortcut path. Given the current readout status $z^l \in \mathbb{R}^n$, the collection process at the l-th layer is defined by:

$$\alpha_i^l = g_w(h^l, f_i^t) \cdot f_{top}(g_t(e^l, e_i^t)), \;\; h^l = W^h z^l, \tag{9}$$

$$r^l = \text{softmax}(\boldsymbol{\alpha}^l)^T V^l + z^l, \tag{10}$$

$$z^{l+1} = \text{FFN}(r^l) + r^l, \tag{11}$$

where α_i^l denotes the weight of the i-th feature that constitutes a weight vector $\boldsymbol{\alpha}^l \in \mathbb{R}^N$, $e^l \in \mathbb{R}^d$ is a learnable vector representing the semantics of the readout node at the l-th layer, f_i^t is an encoded feature (Eq. (3)) of each layer, and e_i^t is a layer-wise column embedding (Eq. (4)). V^l is the transformed input features (Eq. (7)). Here we put z^l forward through the same FFN transformation to transform the current readout into the feature space at the $(l+1)$-th layer for the next round of collection. The shortcut paths are directly added without information drop. This collection process is repeated from the input features to the highest-level features, thus encouraging interactions among cross-level features.

The Overall Architecture and Training

Basic blocks are stacked in T2G-FORMER (Fig. 2(a)). If without special specification, in experiments we use 8-head *GE* in each block by default (Fig. 2(b)). Prediction is made based on the readout status after processing the final layer L, as:

$$\hat{y} = \text{FC}(\text{ReLU}(\text{LN}(z^L))),$$

where LN and FC denote layer normalization and a fully connected layer, respectively. As for optimization, we use the cross entropy loss for classification and the mean squared error loss for regression, as in previous DNNs. We tested various tasks and observed that continuing to optimize the static graph topology A in Eq. (5) across the whole training phase may lead to unstable performance on some easy tasks (e.g., binary classification, small datasets, or few input features). Thus, we freeze it after convergence for further training in a fixed topology manner.

Note that we introduce additional hyperparameters d (Eq. (4)) and T (Eq. (5)). In experiments, we adaptively set $d = 2\lceil \log_2 N \rceil$ which is for the minimal amount of information to present an adjacency matrix with N^2 binary elements, and keep $T = 0.5$ across all the datasets. We choose *sigmoid* as σ_1 and *softmax* as σ_2. Straight-through trick (Bengio, Léonard, and Courville 2013) is used to solve the undifferentiable issue of the indicator function in Eq. (5).

Experiments

In this section, we present extensive experimental results and compare with a wide range of state-of-the-art tabular learning DNNs and GBDT. Also, we conduct empirical experiments to examine the impacts of some key T2G-FORMER components, including comparison of the feature relation graph (FR-Graph) types, ablation study of self-interaction, and the effect of GE. Besides, we explore the model interpretability by visualizing the FR-Graphs and readout selection on two semantically rich datasets.

Experimental Setup

Datasets. We use twelve open-source tabular datasets. Gesture Phase Prediction (GE, (Madeo, Lima, and Peres 2013)), Churn Modeling (CH, Kaggle dataset), Eye Movements (EY, (Salojärvi, Puolamäki et al. 2005)), California Housing (CA, (Pace and Barry 1997)), House 16H (HO, OpenML dataset), Adult (AD, (Kohavi et al. 1996)), Helena (HE, (Guyon, Sun-Hosoya et al. 2019)), Jannis (JA, (Guyon, Sun-Hosoya et al. 2019)), Otto Group Product Classification (OT, Kaggle dataset), Higgs Small (HI, (Baldi, Sadowski, and Whiteson 2014)), Facebook Comments (FB, (Singh, Sandhu, and Kumar 2015)), and Year (YE, (Bertin-Mahieux, Ellis et al. 2011)). For each dataset, data preprocessing and train-validation-test splits are fixed for all the methods according to (Gorishniy, Rubachev et al. 2021; Gorishniy, Rubachev, and Babenko 2022). Dataset statistics are given in Table 1, and more details are in Appendix A.

Implementation Details. We implement our T2G-FORMER model using PyTorch on Python 3.8. All the experiments are run on NVIDIA RTX 3090. In training, if without special specification, we use FR-Graphs with symmetric edge weights and asymmetric graph topology in GE. The optimizer is AdamW (Loshchilov and Hutter 2018) with the default configuration except for the learning rate and weight decay rate. For DANet-28, we follow its QHAdam optimizer (Ma and Yarats 2018) and the pre-set hyperparameters given in (Chen, Liao et al. 2022) without tuning. For the other DNNs and XGBoost, we follow the settings provided in (Gorishniy, Rubachev et al. 2021) (including the optimizers and hyperparameter spaces), and perform hyperparameter tuning with the Optuna library (Akiba, Sano et al. 2019) and grid search (only for NODE). More detailed information of hyperparameters is provided in Appendix B.

Comparison Methods. In our experiments, we compare our T2G-FORMER with the representative non-deep method XGBoost (Chen and Guestrin 2016) and the known DNNs, including NODE (Popov, Morozov, and Babenko 2019), AutoInt (Song et al. 2019), TabNet (Arik and Pfister 2021), DCNv2 (Wang, Shivanna et al. 2021), FT-Transformer (Gorishniy, Rubachev et al. 2021), and DANets (Chen, Liao et al. 2022). Some other common DNNs such as MLP and SNN (an MLP network with SELU activation) (Klambauer et al. 2017) are taken into comparison as well.

Main Results and Analyses

Performance Comparison. The performances of the DNNs and non-deep models are reported in Table 2. T2G-FORMER outperforms these DNNs on eight datasets, and is comparable with XGBoost in most the cases. All the models are hyperparameter-tuned by choosing the best validation results with Optuna-driven tuning (Akiba, Sano et al. 2019).

The Effect of FR-Graph Types. We compare four types of FR-Graphs in GE. Table 3 reports the results, from which one can see that it is often better to choose symmetric edge weights and asymmetric knowledge topology. This suggests that mutual interactions between two tabular features are likely to be the same, and asymmetric topology offers a larger semantic exploration space that is more likely to yield useful features. The results on the other datasets are provided in Appendix C.

The Effect of Self-interaction. One of our key designs in GE is the "no self-interaction function" that explicitly excludes self-loops in FR-Graphs. Table 4 reports comparison results on several datasets with no self-loop FR-Graphs (ours) and self-loop FR-Graphs. The results show that in most the cases, removing self-loops and focusing on interactions with other features slightly benefit performances in both classification and regression. This may be because feature self-interaction affects the probabilities of interactions with other features (as we use competitive activation in Eq. (6)), while our shortcut paths have already preserved self-information.

The Effect of GEs. We explore the impact of including GEs at different layers of T2G-FORMER. Table 5 reports the performances of different model versions which differ solely in the positions and numbers of GEs used. As for the layers without GE, we use the ordinary attention score for substitution. Overall, the positions of GEs show bigger influence on regression tasks than on classification tasks. As one can see, in regression tasks, the model incurs larger performance drops when GEs are equipped in higher layers, while the drops do not seem so large related to GE positions in classification. Also, the model equipped with only attention score is better than the one with a single GE in a high layer (not in the first layer) for regression tasks, but is always sub-optimal in classification tasks. A probable explanation is that regression needs a smoother optimization space than classification, and thus the fully connected attention score provides the kind of interactions to cope with continuous feature values, while a single GE in a high layer is difficult to capture clear relations among features fused in the fully connected manner. Therefore, it is better to completely use attention score than a single GE in a high layer for regression. A single GE in the first layer shows the least performance drop in both regression and classification, which can be explained by the strength of GE in capturing underlying relations among tabular features with clear semantics.

In summary, the removal of GE in any layers is likely to cause performance drop, and the best results are achieved by applying GE to all the layers.

Comparison of Topology Learning Approaches. Apart from the column embedding approach proposed in Sec. , there are some other intuitive straightforward approaches to get knowledge topology of the RF-Graph, for example, performing threshold clipping on the adaptive edge weights directly (we call it "**adaptive** topology") or learning an N-by-N adjacency matrix (we call it "**free** topology"). Concretely, for learning **adaptive** topology, we substitute G_t in Eq. (5)

Dataset	GE	CH	EY	CA	HO	AD	OT	HE	JA	HI	FB	YE
# features	32	9+1	26	8	16	6+8	93	27	54	28	50+1	90
# samples	9873	10000	10936	20640	22784	48842	61878	65196	83733	98050	197080	515345
# classes	5	2	3	-	-	2	9	100	4	2	-	-
Metric	Acc.	Acc.	Acc.	RMSE	RMSE	Acc.	Acc.	Acc.	Acc.	Acc.	RMSE	RMSE

Table 1: Some details of the 12 public datasets. ”RMSE” denotes root mean squared error (for regression), and “Acc.” means accuracy (for classification). The number following each “+” in the row of “# features” is the number of categorical features.

	GE ↑	CH ↑	EY ↑	CA ↓	HO ↓	AD ↑	OT ↑	HE ↑	JA ↑	HI ↑	FB ↓	YE ↓
XGBoost	68.42	85.92	72.51	0.436	3.169	87.30	82.46	37.47	71.85	72.41	5.359	8.850
MLP	58.64	85.77	61.10	0.499	3.173	85.35	80.99	38.38	71.97	72.00	5.943	8.849
SNN	64.69	85.74	61.55	0.498	3.207	85.40	81.17	37.19	71.94	72.21	5.892	8.901
TabNet	60.01	85.01	62.08	0.513	3.252	84.84	79.06	37.86	72.26	71.97	6.559	8.916
DANet-28	61.63	85.10	60.53	0.524	3.236	85.00	81.04	35.45	70.72	71.47	6.167	8.914
NODE	53.94	85.86	65.54	0.463	3.216	85.77	80.37	35.33	72.78	72.51	**5.698**	**8.777**
AutoInt	58.33	85.51	61.07	0.472	3.147	85.66	80.11	37.26	72.08	72.51	5.852	8.862
DCNv2	55.72	85.68	61.37	0.489	3.172	85.48	80.15	38.61	71.56	72.20	5.847	8.882
FT-Transformer	61.25	86.07	70.84	0.460	**3.124**	85.72	81.30	**39.10**	73.24	73.06	6.079	8.852
T2G-FORMER	**65.57**	**86.25**	**78.18**	**0.455**	3.138	**85.96**	**81.87**	39.06	**73.68**	**73.39**	5.701	8.851

Table 2: Performance comparison on the 12 public tubular datasets. Each result reported is averaged over 15 random seeds. For standard deviations, see Appendix C. For each dataset, the top performances among the DNNs are marked in bold, and the second best results are underlined. We also report XGBoost results as a typical representation of GBDT models. ↓ represents the RMSE metric (the lower the better) and ↑ represents accuracy (the higher the better).

FR-Graph	EY ↑	HO ↓	FB ↓	YE ↓
$\mathrm{A^wS^t}$	77.34	3.171	5.736	8.886
$\mathrm{A^wA^t}$	77.59	3.145	5.718	8.861
$\mathrm{S^wS^t}$	76.46	3.151	5.723	8.885
$\mathrm{S^wA^t}$ (ours)	78.18	3.138	5.701	8.851

Table 3: Comparison of four FR-Graph types on several tasks and datasets. “A” means asymmetric, and “S” means symmetric. “$\mathrm{A^wS^t}$”, for example, is for asymmetric edge weights and symmetric graph topology. Likewise, “$\mathrm{A^wA^t}$”, “$\mathrm{S^wS^t}$”, and “$\mathrm{S^wA^t}$” denote the other three types of FR-Graphs.

	EY ↑	HO ↓	FB ↓	YE ↓
w/o SL (ours)	78.18	3.138	5.701	8.851
SL	77.89	3.152	5.691	8.856
SL − w/o SL	−0.29	0.014	−0.01	0.005

Table 4: Comparison of the effects of FR-Graphs without (w/o) self-loops and FR-Graphs with self-loops. “SL” means self-loops.

with G_w in Eq. (2). For learning the **free** topology, we directly represent G_t by an N-by-N matrix. Table 6 reports the comparison results of these topology learning strategies. One can see that, the static knowledge topology shared on the whole dataset (our approach) attains superior performances than the adaptive topology, implying the plausibility of our underlying knowledge assumption mentioned in Sec. . Besides, the completely free topology also achieves inferior performances, which is probably because of the excessive freedom given to the learnable matrix.

Comparison with DANet Grouped Interactions. As illustrated in Fig. 1, DANets (Chen, Liao et al. 2022) interacted tabular features in the group determined by the “entmax” operation. Here we compare our **graph-based** interaction with that **group-based** one to inspect the benefits of FR-Graph. Specifically, we substitute the knowledge topology A in Eq. (5) with DANet grouped selection mask. The results in Table 7 suggest that it of greater benefits to organize tabular features into a graph, since a graph topology is able to capture relation edges and provide more subtle interactions than a group structure.

Interpretability

In Fig. 3, we visualize the first-layer FR-Graph and the readout collecting strategy on the input features (i.e., features from the feature tokenizer; see Fig. 2(a)). On the CA dataset, it is reasonable to find that the median income (*MedInc*, *MI*) of the residents within a block group is related to the average number of the household members (*AveOccup*, *AvO*), and *AveOccup* can affect the average number of bedrooms

	CA (×100) ↓	JA ↑
All	45.53	73.68
# 1	45.78	73.40
# 2	45.96	73.31
# 3	46.06	73.37
None	45.84	73.23

Table 5: Performances of including GEs in different layers of T2G-FORMER. All the results are obtained with a 3-layer T2G-FORMER. "# i" means that only the i-th layer has GE while the other layers replace GE with the ordinary attention score, "All" means that all the layers are equipped with GE, and "None" means that all the layers use ordinary attention.

Topology	CA (×100) ↓	JA ↑	Complexity
ours	45.53	73.68	O($N \log N$)
adaptive	45.88	73.08	O(1)
free	45.87	73.46	O(N^2)

Table 6: Performances of different topology learning approaches. "Complexity" indicates the additional space computational complexity (the amount of extra model parameters) caused by the number of tabular features N.

(*AveBedrooms*, *AvB*). Also, there appear to be some relations such as *Longitude* (*Lon*)-*HouseAge* (*HoA*), *Longitude-AveRooms* (*AvR*), and *Longitude-Population* (*Pop*), which are probably derived from dataset bias. As for readout, one can see that solely *HouseAge* is collected that is a meaningful feature in house price prediction. On the CH dataset, there are reasonable relations between *Balance* (*Bal*, bank balance of a customer) and *EstimatedSalary* (*ESa*), as well as the age of the customer (*Age*) and *EstimatedSalary*. Also, it is interpretable that the credit score of a customer (*CreditScore*, *CrS*) is highly related to that customer's *Age* and *Balance*. The readout collects only *Age* in the current level for predicting whether a customer will leave the bank, which is intuitive as well.

Conclusions

In this paper, we proposed T2G-FORMER, a new bespoke Transformer model for tabular learning with a novel module *Graph Estimator* (GE) for promoting heterogeneous feature interaction based on estimated relation graphs. We adapted feature relation graphs into the basic blocks of T2G-FORMER in an attention-like fashion for simplicity and applicability. Experiments on extensive public datasets showed that T2G-FORMER achieves better performances than various DNNs and is comparable with XGboost. We expect that our T2G-FORMER will serve as a strong baseline in tabular learning studies and enhance research interest in handling feature heterogeneity of tabular data.

Interaction	CA (×100) ↓	HO ↓	JA ↑
graph (ours)	45.53	3.138	73.68
group (DANet)	45.88	3.215	73.08

Table 7: Comparison with DANet group-based interaction on several datasets.

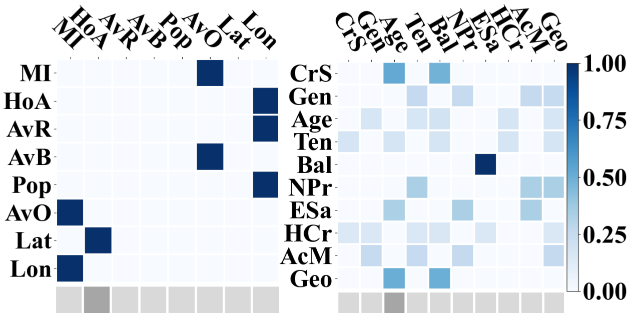


Figure 3: Visualization of the FR-Graph edge weights in the first layer (heat map) and the readout selection (dark bar) on the datasets CA (left) and CH (right). More details of the feature descriptions are given in Appendix D.

Acknowledgments

This research was partially supported by the National Key R&D Program of China under grant No. 2018AAA0102102 and National Natural Science Foundation of China under grants No. 62132017.

References

Akiba, T.; Sano, S.; et al. 2019. Optuna: A next-generation hyperparameter optimization framework. In *KDD*.

Anghel, A.; et al. 2018. Benchmarking and optimization of gradient boosting decision tree algorithms. In *NeurIPS*.

Arik, S. Ö.; and Pfister, T. 2021. TabNet: Attentive interpretable tabular learning. In *AAAI*.

Baldi, P.; Sadowski, P.; and Whiteson, D. 2014. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Bertin-Mahieux, T.; Ellis, D. P. W.; et al. 2011. The million song dataset. In *Proceedings of the International Society for Music Information Retrieval Conference*.

Borisov; et al. 2021. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*.

Chen, J.; Liao, K.; et al. 2022. DANets: Deep abstract networks for tabular data classification and regression. In *AAAI*.

Chen, T.; and Guestrin, C. 2016. XGBoost: A scalable tree boosting system. In *KDD*.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for YouTube recommendations. In *RecSys*.

Dong, H.; Cheng, Z.; et al. 2022. Table Pre-training: A survey on model architectures, pre-training objectives, and downstream tasks. In *IJCAI*.

Dosovitskiy, A.; Beyer, L.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.

Feng, J.; Yu, Y.; and Zhou, Z.-H. 2018. Multi-layered gradient boosting decision trees. In *NeurIPS*.

Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*.

Gorishniy, Y.; Rubachev, I.; and Babenko, A. 2022. On embeddings for numerical features in tabular deep Learning. *arXiv preprint arXiv:2203.05556*.

Gorishniy, Y.; Rubachev, I.; et al. 2021. Revisiting deep learning models for tabular data. In *NeurIPS*.

Guo, H.; Tang, R.; et al. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *IJCAI*.

Guyon, I.; Sun-Hosoya, L.; et al. 2019. Analysis of the AutoML Challenge Series. *Automated Machine Learning*.

Hassan, M. R.; Al-Insaif, S.; et al. 2020. A machine learning approach for prediction of pregnancy outcome following IVF treatment. *Neural Computing and Applications*.

Hazimeh, H.; et al. 2020. The tree ensemble layer: Differentiability meets conditional computation. In *ICML*.

He, K.; Zhang, X.; et al. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*.

He, X.; Pan, J.; et al. 2014. Practical lessons from predicting clicks on ads at Facebook. In *Proceedings of the International Workshop on Data Mining for Online Advertising*.

Huang, X.; et al. 2020. TabTransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

Johnson, A. E.; Pollard, T. J.; et al. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data*.

Katzir, L.; Elidan, G.; and El-Yaniv, R. 2020. Net-DNF: Effective deep modeling of tabular data. In *ICLR*.

Ke, G.; Meng, Q.; et al. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *NeurIPS*.

Kenton, J. D. M.-W. C.; and Toutanova, L. K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-normalizing neural networks. In *NeurIPS*.

Kohavi, R.; et al. 1996. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *KDD*.

Kontschieder, P.; Fiterau, M.; et al. 2015. Deep neural decision forests. In *ICCV*.

Li, B.; Friedman, J.; Olshen, R.; and Stone, C. 1984. Classification and regression trees (CART). *Biometrics*.

Loshchilov, I.; and Hutter, F. 2018. Decoupled weight decay regularization. In *ICLR*.

Lou, Y.; and Obukhov, M. 2017. BDT: Gradient boosted decision tables for high accuracy and scoring efficiency. In *KDD*.

Ma, J.; and Yarats, D. 2018. Quasi-hyperbolic momentum and Adam for deep learning. In *ICLR*.

Madeo, R. C.; Lima, C. A.; and Peres, S. M. 2013. Gesture unit segmentation using support vector machines: Segmenting gestures from rest positions. In *Proceedings of the Annual ACM Symposium on Applied Computing*.

Martins, A.; et al. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*.

Nickel, M.; Rosasco, L.; and Poggio, T. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Nickel, M.; et al. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Pace, R. K.; and Barry, R. 1997. Sparse spatial autoregressions. *Statistics & Probability Letters*.

Popov, S.; Morozov, S.; and Babenko, A. 2019. Neural oblivious decision ensembles for deep learning on tabular data. In *ICLR*.

Prokhorenkova, L.; Gusev, G.; et al. 2018. CatBoost: Unbiased boosting with categorical features. In *NeurIPS*.

Salojärvi, J.; Puolamäki, K.; et al. 2005. Inferring relevance from eye movements: Feature extraction. In *NeurIPS-W*.

Shi, B.; and Weninger, T. 2018. Open-world knowledge graph completion. In *AAAI*.

Singh, K.; Sandhu, R. K.; and Kumar, D. 2015. Comment volume prediction using neural networks and decision trees. In *IEEE UKSim-AMSS International Conference on Computer Modelling and Simulation*.

Song, W.; et al. 2019. AutoInt: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*.

Thawani, A.; Pujara, J.; et al. 2021. Representing number in NLP: A survey and a vision. In *NAACL-HLT*.

Trouillon, T.; Welbl, J.; et al. 2016. Complex embeddings for simple link prediction. In *ICML*.

Vaswani, A.; Shazeer, N.; et al. 2017. Attention is all you need. In *NeurIPS*.

Wang, R.; Fu, B.; et al. 2017. Deep & cross network for ad click predictions. In *ADKDD*.

Wang, R.; Shivanna, R.; et al. 2021. DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *WWW*.

Wu, L.; et al. 2021. Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090*.

Yang, B.; et al. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Yang, Y.; Morillo, I. G.; and Hospedales, T. M. 2018. Deep neural decision trees. In *ICML-W*.

Zhang, J.; and Honavar, V. 2003. Learning from attribute value taxonomies and partially specified instances. In *ICML*.

Zhang, J.; Kang, D.-K.; et al. 2006. Learning accurate and concise Naïve Bayes classifiers from attribute value taxonomies and data. *Knowledge and Information Systems*.