CT-BERT: Learning Better Tabular Representations Through Cross-Table Pre-training

Chao Ye[*]
Zhejiang University
Hangzhou, China
ye.chao@zju.edu.cn

Guoshan Lu[*]
Zhejiang University
Hangzhou, China
luguoshan@zju.edu.cn

Haobo Wang
Zhejiang University
Hangzhou, China
wanghaobo@zju.edu.cn

Liyao Li
Zhejiang University
Hangzhou, China
liliyao@zju.edu.cn

Sai Wu
Zhejiang University
Hangzhou, China
wusai@zju.edu.cn

Gang Chen
Zhejiang University
Hangzhou, China
cg@zju.edu.cn

Junbo Zhao[†]
Zhejiang University
Hangzhou, China
j.zhao@zju.edu.cn

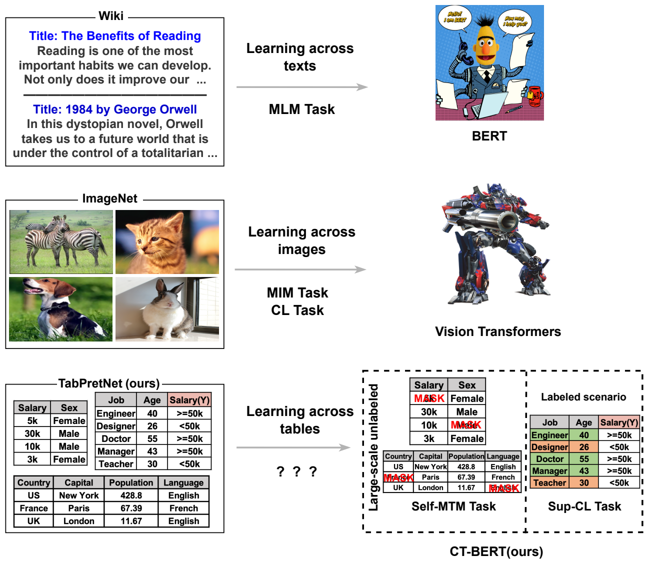
ABSTRACT

Tabular data — also known as structured data — is one of the most common data forms in existence, thanks to the stable development and scaled deployment of database systems in the last few decades. At present however, despite the blast brought by large pre-trained models in other domains such as ChatGPT [1] or SAM [34], how can we extract common knowledge across tables at a scale that may eventually lead to generalizable representation for tabular data remains a full blank. Indeed, there have been a few works around this topic. Most (if not all) of them are limited in the scope of a single table or fixed form of a schema. In this work, we first identify the crucial research challenges behind tabular data pre-training, particularly towards the cross-table scenario. We position the contribution of this work in two folds: (i)-we collect and curate nearly 2k high-quality tabular datasets, each of which is guaranteed to possess clear semantics, clean labels, and other necessary meta information. (ii)-we propose a novel framework that allows cross-table pre-training dubbed as CT-BERT. Noticeably, in light of pioneering the scaled cross-table training, CT-BERT is fully compatible with both supervised and self-supervised schemes, where the specific instantiation of CT-BERT is very much dependent on the downstream tasks. We further propose and implement a contrastive-learning-based and masked table modeling (MTM) objective into CT-BERT, that is inspired from computer vision and natural language processing communities but sophistically tailored to tables. The extensive empirical results on 15 datasets demonstrate CT-BERT's state-of-the-art performance, where both its supervised and self-supervised setups significantly outperform the prior approaches.

1 INTRODUCTION

With the extensive application of database management systems and the vigorous development of the internet industry, tabular data — also known as structured data — truly abounds. Indeed, the accumulation of scaled tables stored in databases has brought significant value to the industry or individuals, through tech stacks like data mining or the development of OLAP databases.

Notably, over the past decade, various large-scale collections of tabular datasets have been proposed [13, 15], and they were used for tasks like tableQA [19, 52], table interpretation [7, 22, 46], table expansion [2, 14, 38], etc. Despite that, how to enable a large-scale, distributed, and cross-table pre-training very much remains untapped.



Title: The Benefits of Reading
Reading is one of the most important habits we can develop. Not only does it improve our ...

Title: 1984 by George Orwell
In this dystopian novel, Orwell takes us to a future world that is under the control of a totalitarian ...

Salary	Sex
5k	Female
30k	Male
10k	Male
3k	Female

Job	Age	Salary(Y)
Engineer	40	>=50k
Designer	26	<50k
Doctor	55	>=50k
Manager	43	>=50k
Teacher	30	<50k

Country	Capital	Population	Language
US	New York	428.8	English
France	Paris	67.39	French
UK	London	11.67	English

Salary	Sex
MASK	Female
30k	Male
10k	MASK
3k	Female

Country	Capital	Population	Language
US	New York	428.8	English
MASK	Paris	67.39	French
UK	London	11.67	MASK

Job	Age	Salary(Y)
Engineer	40	>=50k
Designer	26	<50k
Doctor	55	>=50k
Manager	43	>=50k
Teacher	30	<50k

Figure 1: An example of our method compared with Natural language processing (NLP) and computer vision (CV). Our work pioneers the research of a relatively unexplored direction in tabular modeling: large-scale cross-table pre-training. MLM is short for masked language modeling, MLM is short for masked image modeling, CL is short for contrastive learning, Sup is short for supervised, and MTM is short for our proposed masked table modeling method.

[*]Chao Ye and Guoshan Lu are co-first authors of the article.
[†]Junbo Zhao is the corresponding author.

This, unfortunately, is in stark contrast to the other communities such as computer vision and natural language processing. In both of these domains, techniques like pre-training followed by fine-tuning have long established a dominant methodological status, such as BERT [20], CLip [43], ChatGPT [1], GPT4 [40], SAM [34], etc. In hindsight, the successes of these large-scale models lie in their ability to extract common semantic structure from the seen/unseen input and condense this knowledge/common sense into a vectorial representation. The emergence of this capacity stems from a scaled pre-training process on a gigantic amount of text or vision data across the domains.

Recently, a few works have attempted to learn contextualized representation from tabular data through neural networks, or more specifically the transformer model [56], such as TabTransformer [29], VIME [66], TabNet [4], SAINT [50], etc. While the concept is truly promising, these approaches are limited to single-table training with a fixed form of a schema. Most closely related to our work are TransTab [59] and PTab [36]. Both approaches note the importance of cross-table learning. However, they process the table to a proximal form of text data, for instance by converting a sample row in the table into a sentence, without doing much adaption specifically to the structured data. This weakened coupling of the data values in the tables with the schema/meta/column names has arguably obstructed these approaches to scale and absorb common knowledge.

1.1 Challenges

In what follows, we identify the core challenges that remained in scaled and cross-table pre-training.

C1. How can pre-training models accept inputs from heterogeneous tables as there are significant differences between different tables? For instance, the feature value "apple" appears under the column names "fruit" and "My_Laptop" in two different tables, conveying completely different meanings.

C2. Unlike image or text data where the pixels and word/character tokens are ordered, arbitrarily permuting any tables' rows or columns does not change its semantic meaning. We dub this property as *permutation invariance* uniquely to tabular data. Thus, how can the pre-training mechanism be compatible with this nature of tabular data?

C3. Still driven by the difference against common vision or text data, how to design a suitable cross-table pre-training task objective because there is no obvious context or spatial structure in the tabular data?

1.2 Key Idea behind CT-BERT

Ideally, in order for the pre-trained model to properly acquire the common knowledge from multiple heterogeneous tables, the model should be encouraged to learn the innate similarities or dissimilarities among the tabular data distribution. However, as we posited in the challenges, directly utilizing the original form of the data (or its corresponding embedding) may cause unentangleable confusion. Let us give a concrete example; given two tables with similar schema, two forms “10 meters" and “10 kg" are iconically identical. Despite that, directly converting them to embedding may inherently confuse and adversely impact the convergence or training difficulty. Abstracting away from this example, to cope with this challenge, the pre-training methodology must be capable to conform the different metrical systems or different notations. It is true that we can write heuristic rules to tackle this problem, but the amount of it would be surely insurmountable.

In that regard, we outline the core idea behind CT-BERT. In a nutshell, provided with any table, it can always be decomposed to **feature** that denotes the data curated column-wise, together with **token** drawn from the schema information such as the column name or other textual meta-information. Instead of following a normal embedding-based encoding approach, we proactively combine the feature with the token information, by casting them into a form of textual representation. For example, we convert the feature value "apple" combined with the schema information to "fruit is apple", which we dub as a **phrase**, as the atomic representation of the cell value in tabular data. This allows to distinguish the same feature value "apple" in column "fruit" and "My_Laptop" respectively.

We postulate that this manifests several merits. In particular, the challenge **C1** can be both theoretically and empirically solved, and this formation is rid of many heuristic rules, except the template for sticking the feature and token together.

1.3 Our Methodology: CT-BERT

Essentially, CT-BERT bases itself upon the *phrase* as the atomic representation of each unit in any provided table, in combination of the feature (column name/meta) with the feature value. We then process each atomic element similarly to word embedding in NLP. Towards the challenge **C2** of the permutation invariance property, we propose a novel transformer [56] encoding architecture that is adapted to cater to this nature of tabular data.

As a pioneer work to enable cross-table pre-training, we devise CT-BERT to be compatible with both supervised and self-supervised scenarios. In that regard, we profoundly categorized the available tables drawn from databases by a standard whether there exists a clear label column or not, that we direct it to supervised and self-supervised learning paradigms respectively. On one hand, for supervised learning, we propose a supervised contrastive learning-based objective to better cluster samples with the same label while allowing different labels to be uniformly distributed over the hypersphere of tabular representations. On the other hand, in order to take advantage of large-scale unsupervised data, we propose another pre-training method of masked table modeling (which we call MTM) — adapted from the MLM objective in the NLP community [20] — which facilitates to mask some features in the atomic then let the model predict the recovery (for challenge **C3**).

We believe that if the model can predict the masked features from the retained features, then the model can learn the underlying relationship between the features. Similar to CV or NLP, this relationship serves as the foundation to manifest the shareable knowledge that is migrated across tables.

1.4 Contributions

To wrap up, the contribution of this article is deemed two-fold. For one thing, we collect and curate nearly 2,000 tabular datasets, each of which is guaranteed to possess clear semantics, clean labels, and other necessary meta information. We treat these high-quality and labeled datasets as the foundation to launch large-scale pre-training. For another, we propose a generic and efficient cross-table pre-training solution, dubbed as **C**ross-**T**a**b**l**e** p**r**e-**T**raining framework (CT-BERT). CT-BERT promotes several novel development bullets including but not limited to: (i)-a novel paradigm compatible with both supervised and self-supervised objectives, (ii)-a contrastive learning and masked table modeling (MTM) objectives for pre-training tables, and a novel transformer architecture tailored to the permutation invariance nature of tabular data. Our pre-trained tabular model can support fine-tuning or few-shot learning for prediction on tables of any shape.

The remainder of the paper is organized as follows. In Section 4, we detail the table pre-training dataset TabPretNet we contributed. In Section 5, we present the proposed CT-BERT cross-table pre-training framework. In Section 6, we constructed extensive experiments to evaluate the effectiveness and superiority of CT-BERT.

2 RELATED WORKS

We provide a brief background on representation learning, models for tabular data, and self-supervised pre-training methods.

2.1 Representation Learning

In recent years, with the development of pre-trained large language models ("LLMs") like GPT-3 [12], the pre-training then fine-tuning and prompting paradigms have attracted attention. These methods typically train models with self-supervised representation learning methods from large-scale unstructured text and structured knowledge bases, and then fine-tune them or use them for various downstream tasks. In early work in natural language, including Word2Vec [39] and GloVe [42], pre-training distributed representations of words provided significant improvements over randomly initialized parameters. However, these methods cannot simulate the use of words in different linguistic contexts. This dilemma has prompted the development of vocabulary representations that can learn context and contextual relationships [20, 47, 64], and these pre-trained language models have achieved tremendous success and produced state-of-the-art results in various NLP tasks [57]. Similarly, self-supervised representation learning can also be used for tabular data, such as knowledge bases (KB) and databases, where entities and relationships in the KB can be embedded into continuous vector spaces and then utilized for various downstream tasks, such as KB completion [11, 60], relation extraction [45, 61], entity resolution [10], etc. Although representation learning on the text and KB has been successful, few works have explored directly learning self-supervised representations on large-scale tabular data for tabular modeling. In this work, we introduce CT-BERT, which is the first method for self-supervised pre-training on large-scale tabular data, and the pre-trained model can be fine-tuned for various downstream tabular prediction tasks.

2.2 Models for Tabular Data

For a long time, traditional machine learning (ML) methods such as tree-based methods [16] have dominated this field and have been the preferred choice for most practitioners and data mining competitions (e.g., Kaggle) [9]. Recently, many researchers have proposed new neural network-based architectures [4, 24, 25, 29, 50, 51, 59, 66] to model tabular data, attempting to challenge the dominance of tree-based models in this field. For example, TabNet [4] uses sequential attention to simulate the process of tree decision-making, TabTransformer [29] leverages transformers [56] to learn categorical features in tables, and AutoInt [51] utilizes attention mechanism [56] to model the relationship between user and item features in click-through rate prediction tasks. However, only very few of these neural network-based models work [36, 59] attempt to investigate how to handle heterogeneous tabular inputs. This leads to the advantage that deep learning methods can be pre-trained on large-scale datasets that cannot be fully exploited. As described in Section 5.2 and 5.3, our proposed CT-BERT not only can accept inputs from heterogeneous tables but also achieves permutation invariance of feature columns, and leverage semantic knowledge from table headers and textual features. These advancements pave the way for CT-BERT to be pre-trained on large-scale datasets for cross-table prediction.

2.3 Self-supervised pre-training

One of the key reasons for the great success of deep learning in computer vision and natural language processing is that knowledge on a large amount of unlabeled datasets is learned through a self-supervised pre-training task and then generalized to downstream tasks through fine-tuning. For instance, masked language modeling (MLM) self-supervised pre-text task [20, 49] is employed to learn contextual relationships in natural language processing. In computer vision, masked image modeling (MIM) [21, 49] and contrastive learning [32] have been used to train powerful image representations. Some studies have attempted to extend the success of self-supervised learning to tabular data. These approaches can be roughly categorized into three types: 1) reconstruction of masked inputs [55]; 2) contrastive learning similar to that in SimCLR [17]; 3) a combination of the first two. For example, VIME [66] utilizes autoencoders to reconstruct corrupted table inputs. SCARF [6] randomly selects and replaces certain features with corresponding empirical marginal distributions to construct different views of the same sample. We argue that contrastive learning methods similar to that in SCARF [6] are not applicable to large-scale unlabeled cross-table pre-training tasks. Assuming the existence of a priori true labels for these unlabeled samples, such contrastive learning methods are highly likely to distance samples with the same labels, especially for tables with unrich sample labels. We are more inclined to believe that methods like masked language modeling (MLM) and masked image modeling (MIM) have greater potential. Therefore, in this work, for the first time, we formalize this series of approaches as masked table modeling (MTM) tasks. Additionally, we propose a novel masked table modeling method that combines semantic cues from table headers, which is more suitable for learning cross-table knowledge.

3 PRELIMINARY

3.1 Problem Formulation

For a given tabular data $D = (\mathbf{x}_i, y_i)_{i=1}^{n}$ where n refers to the number of samples. $\mathbf{x}_i = \{\mathbf{x}_i^{cat}, \mathbf{x}_i^{num}\}$ where $\mathbf{x}_i^{cat} = \{x_i^1, x_i^2, \ldots, x_i^a\}$ denotes all a categorical features, and $\mathbf{x}_i^{num} \in \mathbb{R}^b$ denotes all b

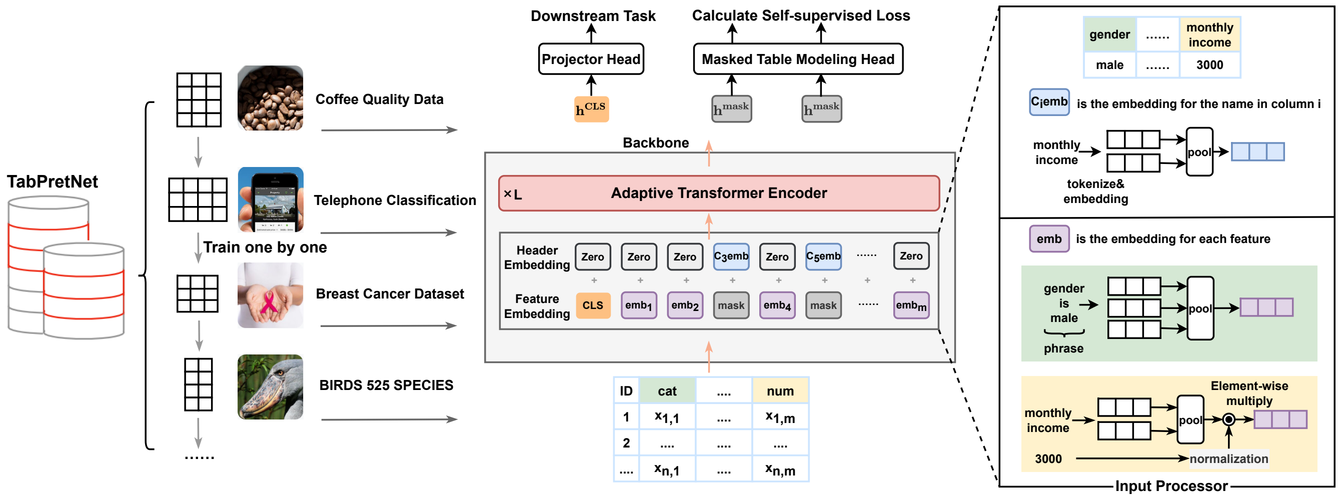


Figure 2: The architecture of the proposed CT-BERT. The number of rows and columns of the tables in TabPretNet (Details in Section 4) varies. First we use the input processor to accept these heterogeneous tables (Details in Section 5.1). Then, we use the L-layer transformer encoder to model the feature interactions (Details in Section 5.2). The obtained $\mathrm{h}^{\mathrm{CLS}}$ representations are used for prediction or pre-training task learning (Details in Section 5.4&5.3). In CT-BERT, we propose self-supervised masked table modeling and supervised contrastive learning to adapt to labeled and unlabeled scenarios, respectively. For self-supervised masked table modeling task (supervised contrastive learning task see Figure 3), we mask some proportion of features (gray features in the figure) and replace them with a shared, learnable vector. The pre-training task aims at recovering these masked features based on the retained features.

numerical features. $y_i \in \{1, 2, \ldots, T\}$ where T refers to the total classes of labels. All samples share the same table header descriptions (column names) $\mathrm{C} = \{c^1, c^2, \ldots, c^{a+b}\}$. Our goal is to find the best possible prediction function f_θ to model the mapping between features and labels:

$$f_\theta(\mathbf{x}_i; \mathrm{C}) = y_i, \tag{1}$$

where θ refers to all trainable parameters of the function f.

3.2 Pre-training then Fine-tuning Paradigm in Tabular Domain

Given a generic architecture, often called a *backbone* such as Transformer, and projection head for mapping to specific tasks, the model is first pre-trained on a large dataset by self-supervised or unsupervised tasks (e.g., Contrastive Learning or MLM). The individual feature columns of the dataset $\{\mathbf{x}^{cat}, \mathbf{x}^{num}\}$ are converted to the input format $\mathbf{x}_i = \{\mathbf{e}^{CLS}, \mathbf{e}^1, \mathbf{e}^2 \ldots, \mathbf{e}^{a+b}\}$, which is sent to the Transformer model, and the model is further optimized using self-supervised or unsupervised objectives. Then, in the downstream task-specific fine-tuning stage, the pre-trained *backbone* module is retained, the pre-trained projection head is discarded and the classification head for the new task is constructed, and the output $\mathbf{e}^{CLS}$ is used for multi-classification and optimized via cross-entropy loss [68], etc.

4 TABPRETNET: A LARGE-SCALE SEMANTIC TABULAR DATABASE

In recent years, the field of cross-table pre-training has been relatively underexplored. One major challenge lies in the lack of a clean and high-quality tabular dataset. Just as the proposal of ImageNet [18] has greatly propelled the advancement of computer vision representation learning and influenced various other domains, such as self-supervised learning and transfer learning, a similar catalyst is needed for the domain of tabular representation learning. Therefore, in this work we contribute a large-scale semantic tabular database, which we called TabPretNet, to better train our CT-BERT. TabPretNet is a large-scale tabular database with high quality built on various public tabular dataset websites and through our strict data cleaning. These tabular datasets are collected from OpenML[1], UCI[2], CATALOG[3], and Kaggle[4]. We have open-sourced TabPretNet[5] and hope to facilitate future research in the field of tabular representation learning.

With the advent of the Big Data era, the proliferation of database technologies has led to an explosion of tabular data on the Internet. These numerous tabular datasets can help more complex and powerful models and algorithms to learn more general tabular representations. And representations are the standard signal linking many machine learning applications in this day and age. This means that more novel AI techniques can be made accessible to databases, such as allowing large language models (e.g., ChatGPT [1]) to understand databases. However, the quality of tables in Internet databases is inconsistent greatly, which can significantly impact the learning performance of models. For example, column name information in some tabular datasets is usually anonymized

[1]https://www.openml.org/
[2]https://archive.ics.uci.edu/datasets
[3]https://catalog.data.gov/dataset
[4]https://www.kaggle.com/
[5]TabPretNet

or unclear to avoid compromising privacy (e.g., named *f1*, *f2*, etc.), which may lose important semantic knowledge to better understand the tabular data. In addition to this, some tabular datasets also suffer from too many missing values, redundant feature columns, lack of consistent formatting, etc. Therefore, in this work, we spent a lot of time filtering and cleaning the tabular data from the Internet database. Specifically for each table, our data cleaning includes the following steps:
(1) Check the semantic degree of the column names for each feature. For example, the column names {*user_age*, weight, *monthly_income*} have high semantic information, while the column names {*f1*, *f2*, *xyz*} have almost no semantic information. We compute the cumulative semantic relevance score for each table. In our cleaning protocol, we discard such tables that have less than 50% of the features having actual semantic information in the column names.
(2) Check the missing values. For example, the datasets with more than 40% missing values are discarded. Because too many missing values can easily lead to biased or inaccurate results. For the retained tables, we fill the missing values with the plural of the corresponding column.
(3) For categorical features in the tables, we aim to restore them to their original textual values. As for numerical features, we employ min-max normalization. This is done to mitigate the impact of inconsistent measurement units across different tables (e.g., kilograms vs. grams).
(4) For the table with labels and more than 100 features, feature filtering based on Random Forest importance [28] is performed, and the features with lower importance ranking are discarded.

At present, TabPretNet has contained about **17G** datasets, including approximately 1000 labeled datasets and 1000 unlabeled datasets. Usually high-quality and semantically rich labeled datasets are more difficult to obtain, while unlabeled tabular datasets are easier to obtain.

Therefore, in supervised pre-training, the theoretical upper bound of model performance is expected to be influenced by the quantity of available labeled tabular datasets at the data level. In contrast, self-supervised pre-training has the potential for a higher upper bound of performance. According to what is suggested in previous research [55], contrastive learning will not be adapted to tables that are not rich in label classed due to the chances of sampling negative samples are low, which is why we propose a novel self-supervised masked table modeling (MTM) pre-training approach. We believe that the contrastive learning-based pre-training approach will be more suitable for lightweight labeled scenarios, and the upper limit of the model will be determined by the number of its available tabular datasets. On the other hand, the self-supervised pre-training approach may require a large amount of data for model training and would also theoretically have more room for improvement.

5 METHODS

Previously proposed table pre-training methods [4, 6, 55, 66] have all been pre-trained on an individual tabular task dataset. As a result, these pre-trained models exhibit notably poor generalization performance on downstream tasks involving other tables. In this section, we detail our proposed novel cross-table pre-training framework CT-BERT, which improves the generalization ability of pre-trained models by learning shareable knowledge across different tables. The overall architecture is provided in Figure 2.

As we have discussed before, cross-table pre-training needs to address three key challenges **C1-C3**.

For **C1**, in Section 5.1 we propose to use a natural language-like approach to process the input of heterogeneous tables and enhance cross-table transfer learning by leveraging semantic knowledge in the schema. For **C2**, in Section 5.2 we use an adapted transformer encoder [56] without positional encoding to model feature-level interactions. For **C3**, in Section 5.3 we propose a novel masked table modeling (MTM) self-supervised pre-training task for large-scale unlabeled dataset scenarios and a contrastive learning-based supervised pre-training task for lightweight labeled dataset scenarios, respectively. At last, in Section 5.4 we introduce fine-tuning the pre-trained model on downstream tasks.

5.1 Input Processor on Heterogeneous Tables

Feature columns among tables from diverse domains often exhibit significant variations. Therefore the previous works [24, 50] often use the table-specific feature extractor which is also called "feature tokenizer" in their literature. This greatly hinders the model to perform cross-table learning. In CT-BERT, we analyze that the table is essentially a multimodal structured data, which contains both text (e.g., column names and discrete categorical values) and continuous values. Based on this observation, we use a natural language-like approach and combine the column name schema information to convert all features into a uniformly formatted *feature phrase*, e.g. [column name] is [value]. This design has two advantages. **First, our model can accept inputs from heterogeneous tables without any table-specific operation.** This serves as a necessary condition for enabling cross-table pre-training. **Second, the knowledge learned from pre-training can be maximized to transfer between similar features by semantic information in the schema across different tables.** For example, gender features are recorded in both tables. In one table, the column name is *gender* and the value is "male", and in the other table, the column name is "sex" and the value is "man". Our model can encode the two *feature phrases* "gender is male" and "sex is man" into two distance proximity embeddings (e.g., cosine similarity is high) based on semantic information.

For each *feature phrase*, we convert it into a low-dimensional embedding and employ it to model the feature interaction in the subsequent phase. The right part of Figure 2 illustrates the details about how we handle the categorical and numerical features separately to get the feature embedding.

Categorical Feature. For each sample x_i, each discrete category will have a corresponding text description (e.g., 1 for a man, 2 for a woman). We concatenate the column name and the original categorical description to form a *feature phrase*. Then, we use a pre-trained BERT [20] model to tokenize the phrase and generate the corresponding embedding for each token, where the pre-trained BERT model contains generic semantic knowledge. Further, we pool these token embeddings of the j-th feature into one feature embedding $\mathbf{e}_i^j \in \mathbf{R}^d$. In our experiments, we tried average, self-attention [48] and other pooling methods. See Section 6.4 for ablation experiments on these pooling strategies. Among them, the average pooling strategy performs the best. Therefore, without a special explanation, average pooling is used by default.

Numerical Feature. We know that at least for now pre-training token embedding of continuous values is ineffective [24]. For numerical features, we similarly process their column names as for categorical features to obtain the header embedding $\mathbf{c}^{j} \in \mathbf{R}^{d}$. Then we multiply the normalized numerical value with the corresponding header embedding to get the feature embedding$\mathbf{e}_i^j = x_i^j \times \mathbf{c}^{j} \in \mathbf{R}^{d}$. Note that the normalization of the numerical values is important here, as it helps the knowledge to transfer better across different tables. Because the same numerical features may have different measurement units across different tables. For example, the unit of height in one table is a meter, but in another table may be a centimeter.

We note that previous works [36, 59] have also tried to combine column names to convert each sample into a sequence of text tokens and the subsequent learning is built on the *token-level*. We think that such *token-level* interactions are more suitable for extracting textual semantic information from tables (e.g., TableQA task [27, 65]), but are not well-suited for our target column prediction task. For example, in a "work" column with the value "associate professor" in a table, this feature will first be converted into three token embeddings: $[work]$, $[associate]$ and $[professor]$. The subsequent model will learn the relationship between $[associate]$ token and $[professor]$ token in the same column, which is unreasonable. The experimental results in Section 6.4 also validate this observation. However, in our design, one column corresponds to one feature embedding, and the subsequent model learns at the *feature-level*. This is a straightforward but effective enhancement. At the same time, for tables with a large number of features, such a design can optimize computational efficiency and memory space usage.

5.2 Feature Interaction

There is no inherent order relationship among different columns in a table. In other words, tables possess permutation invariance in the column dimension. Previous tabular modeling works [24, 36] often overlooked this aspect by directly employing the transformer architecture [56]. Therefore, we have made certain modifications to the standard transformer encoder to adapt it to tabular data. Specifically, we 1) discard positional encoding and 2) use a shared-parameter fully connected feed-forward network at each transformer encoder block. Finally, our adapted transformer encoder block contains two sub-layers: a multi-head self-attention layer, and a shared-parameter fully connected feed-forward layer. In addition, a residual connection [53] is done for each sub-layer, followed by layer normalization [5]. The multi-headed self-attentive mechanism is the key to modeling feature interactions. It learns the relationship between features through Query, Key, and Value matrices. It is calculated as follows:

$$MultiHead(\mathbf{H}^{l}) = Concat(head_1, \dots, head_i, \dots, head_h))\mathbf{W}^{O}, \quad (2)$$

$$head_i = Attention(\mathbf{H}^{l}\mathbf{W}_i^{Q}, \mathbf{H}^{l}\mathbf{W}_i^{K}, \mathbf{H}^{l}\mathbf{W}_i^{V}), \quad (3)$$

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{d}})\mathbf{V}, \quad (4)$$

where $\mathbf{H}^{l} \in \mathbb{R}^{n \times d}$ is the input of the l-th layer; $\mathbf{W}^{O} \in \mathbb{R}^{d \times d}$ is parameter matrix; $\mathbf{W}_i^{Q}$, $\mathbf{W}_i^{K}$ and $\mathbf{W}_i^{V} \in \mathbb{R}^{d \times d_{head}}$. $d_{head} = \frac{d}{h}$ is the dimension of each attention head. Inspired by BERT [20], we add a special classification token ($\mathbf{e}^{CLS} \in \mathbb{R}^{d}$) to the first position of the input sequence in each layer. This special token is used as the aggregate sample representation and is then served for the subsequent pre-training and downstream tasks. As described in Section 5.1, we can obtain the processed feature embeddings $\mathbf{E} = \{\mathbf{e}^1, \mathbf{e}^2, \dots, \mathbf{e}^{a+b}\}$ from the raw tabular data . So we have the first layer of input $\mathbf{H}^0 = [\mathbf{e}^{CLS}, \mathbf{E}]$. Finally, we can model the higher-order feature interactions step by step through the following calculation:

$$\mathbf{H}^{l+1} = LayerNorm(\hat{\mathbf{H}} + linear(\hat{\mathbf{H}})), \quad (5)$$

$$\hat{\mathbf{H}} = LayerNorm(\mathbf{H}^{l} + MultiHead(\mathbf{H}^{l})). \quad (6)$$

5.3 Pre-training Across the Tables

Our work is the first to explore large-scale cross-table pre-training. Supervised and self-supervised pre-training are two major approaches in the field of deep learning. As described in Section 4, we contributed TabPretNet a cross-table pre-training dataset which is collected from various domains and includes approximately 1000 labeled tables and 1000 unlabeled tables. In this work, based on the nature of the collected dataset TabPretNet, we simultaneously explore supervised and self-supervised cross-table pre-training approaches. Firstly, for the relatively more easily learnable labeled tabular datasets, we propose a randomly subsampled supervised contrastive learning approach to adapt to the cross-table pre-training task. Secondly, for large-scale unlabeled tabular datasets, some studies have discussed the limitations of contrastive learning-based methods in unlabeled tabular scenarios [6, 55]. So in order to fully leverage the potential of shareable knowledge within unlabeled tabular data, in CT-BERT, we propose a novel masked table modeling (MTM) self-supervised cross-table pre-training method.

Details of the two cross-table pre-training approaches are as follows:

Supervised contrastive learning. In the labeled tabular scenario, we observe that samples with the same labels tend to have similar feature sets. Based on this observation we make a bold hypothesis: powerful representation should model the invariant factors of feature sets with the same label. We, therefore, propose a random overlapping subsampling method to construct positive and negative samples in contrastive learning.

Figure 3 illustrates how we randomly sample subsets and divide positive and negative pairs. Specifically, for each row $(\mathbf{x}_i, y_i)$ we randomly sample k feature subsets $\{\mathbf{s}_i^1, \mathbf{s}_i^2, \dots, \mathbf{s}_i^k\}$ and set all their labels to y_i. There will be a partial overlap of features between subsets. In this way, feature subsets with the same label form positive pairs, and subsets with different labels form negative pairs. Overall contrastive loss is:

$$\mathcal{L}_{pretrain}^{CL}(\mathbf{X}, \mathbf{y}) = \frac{1}{|B|} \sum_{i \in B} \frac{1}{|P(i)|} \sum_{p \in P(i)} \Psi(\mathbf{z}_i^{CLS}, \mathbf{z}_p^{CLS}), \quad (7)$$

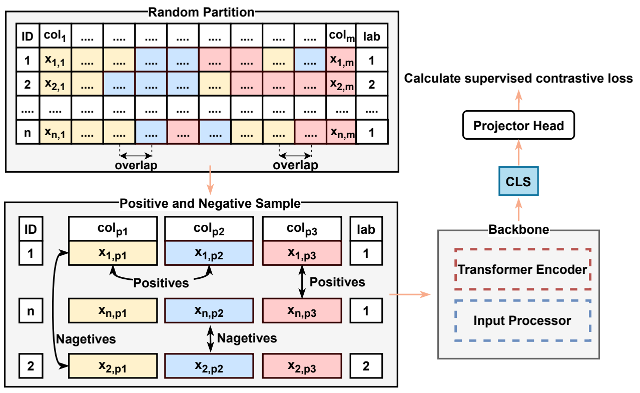


Figure 3: An illustration of contrastive learning supervised pre-training. For each row, we randomly sample K feature subsets (features with the same color in each row in the figure indicate that they are in the same feature subset), and each feature subset inherits the label of the original row. Note these feature subsets may overlap with each other. The feature subsets with the same label are positive samples, and those with different labels are negative samples.

$$\Psi(\mathbf{z}_i^{CLS}, \mathbf{z}_p^{CLS}) = -\log\left(\frac{\exp(sim(\mathbf{z}_i^{CLS}, \mathbf{z}_p^{CLS})/\tau)}{\sum_{i'\in B}\exp(sim(\mathbf{z}_i^{CLS}, \mathbf{z}_{i'}^{CLS})/\tau)}\right), \quad (8)$$

where B is the set of samples in a batch; $P(i) = \{p | p \in B, p \neq i, y_i = y_p\}$. The previous tabular contrastive learning work SCARF [6] focused only on constructing different views of the same samples, simply treating all different samples as negative pairs. This only applies when the sample label classes are very rich such that the sample labels in a batch are almost all different. Compared to the tabular vertical fixed-partitioned contrastive learning method [59], our method can learn more robust sample representations in richer feature subsets by random sampling.

Self-supervised MTM. For large-scale unlabeled scenarios, we propose a novel masked table modeling (MTM) self-supervised cross-table pre-training task. On each sample row in all tables, we mask some percentage of features, and then reconstruct them based on the retained features. We argue that if the model is able to successfully reconstruct the masked features from the retained features, then the model is able to learn the underlying relationships between features that can be transferred as shareable knowledge between different tables with similar feature columns, which will eventually indirectly bring closer the representations of samples with similar feature relationships.

The middle part of Figure 2 shows the overview of our self-supervised MTM pre-training method, which can be divided into three steps. **First step we select the features that are masked.** Given an input table, we first convert all features of each sample into feature embeddings, as described in Section 5.1. Then we mask approximately p^{mask} features for each row (p^{mask} is set to 35% in our experiments and further ablation results are shown in Section 6.5.2). Specifically, we generate a binary mask vector $\mathbf{m} = [m^1, m^2, \ldots, m^{a+b}] \in \{0,1\}^{a+b}$ where m_j is randomly sampled from a Bernoulli distribution with probability p^{mask}. The "1" in $\mathbf{m}$ indicates a masked feature and "0" indicates keeping the original feature. **Second step we replace the masked features with a shared, learnable vector $\mathbf{e}^{mask} \in \mathbb{R}^d$, which is also called mask token.** Note that here we will add additional header embedding, which is obtained by pooling the text token embeddings of the corresponding column name, for each mask token. Because there is no order relationship between the columns in tables. Here the role of header embeddings is like the position embeddings in masked language modeling (MLM) [49] and masked image modeling (MIM) [21, 63] tasks. **Third step we reconstruct these masked features.** We feed the masked sample row $\mathbf{x} = \{\mathbf{e}^j | m^j = 0\} \cup \{\mathbf{e}^{mask} + \mathbf{c}^j | m^j = 1\}$ into the L-layer transformer encoder to get the encoded representations $\mathbf{H} = \{\mathbf{h}^j\}_{j=1}^{a+b}$. For the masked numerical features, we pass it through a numerical projection matrix $\mathbf{M}_{pro}^{num} \in \mathbb{R}^{d\times 1}$ and then calculate the mean square error loss with the original feature values. For the masked categorical features, we pass it through a categorical projection matrix $\mathbf{M}_{pro}^{cat} \in \mathbb{R}^{d\times d}$ and then compute the cosine similarity with the original feature embedding $\mathbf{e}_j$. Here the feature embedding $\mathbf{e}^j$ is calculated in the same way as section 5.1 but with the column names removed. We formulate the masked table modeling pre-training loss as follows:

$$\mathcal{L}_{pretrain}^{mask}(\mathbf{X}) = \frac{1}{|B|}\sum_{i\in B}\Phi(\mathbf{x_i}, \mathbf{e_i}, \mathbf{z_i}), \quad (9)$$

$$\Phi(\mathbf{x_i}, \mathbf{e_i}, \mathbf{z_i}) = \frac{1}{N^{num}}\sum_{j=1}^{N^{num}}(x_i^j - z_i^j)^2 + \frac{1}{N^{cat}}\sum_{j'=1}^{N^{cat}}(1 - sim(\mathbf{e}_i^{j'}, \mathbf{z}_i^{j'})) \quad (10)$$

where B is the set of samples in a batch; $z_i^j = \mathbf{h}_i^j\mathbf{M}_{pro}^{num}$; $\mathbf{z}_\mathbf{i}^{\mathbf{j}'} = \mathbf{h}_i^{j'}\mathbf{M}_{pro}^{cat}$; N^{num} refers to the number of numerical features; N^{cat} refers to the number of categorical features. We do not compute the traditional cross-entropy loss for categorical features because the same category in the same feature column may be inconsistently labeled in different tables, which can lead to confusion when cross-table pre-training. For example, for the "gender" column, one table may have "man" corresponding to label "1" and "woman" corresponding to label "2", while another table might be the exact opposite, with "man" corresponding to label "2" and "woman" corresponding to label "1".

Rather than a completely random mask strategy, we think that the proportion of numerical and categorical features masked can be adjusted according to the downstream task. When the downstream scenario is a regression task, the model needs to predict a continuous value. In this case, the pre-training task to predict the masked numerical features will be more helpful. Similarly, for classification downstream tasks, it will be biased to mask more categorical features. The downstream tasks in our experiment are mainly classification prediction, so we set the mask ratio of categorical features and numerical features to 7:3 during pre-training. The overall mask rate is 35%.

5.4 Fine-Tuning on Downstream Tabular Tasks

After cross-table pre-training, we discarded the original projection header and added a new task layer on the Transformer encoder. We then fine-tuned the parameters on the downstream task datasets.

The downstream scenario in our experiments is mainly classification prediction tasks. So, we employ a simple linear classifier as the task layer. We use softmax [30] to calculate the probability of each label category and use cross-entropy loss as our empirical supervised loss.

$$\mathcal{L}_{task}(\mathbf{X}, \mathbf{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{T} y_{ij} \log(f_\theta(\mathbf{x}_i)), \tag{11}$$

where label y_i uses one-hot encoding; T is the total number of all label categories.

6 EXPERIMENTS

In this section, we evaluate the effectiveness and superiority of CT-BERT on several benchmark tabular datasets. Specifically, we conducted extensive experiments to demonstrate the following two points:

- How does our backbone, which can accept heterogeneous table inputs, compare with the current state-of-the-art tabular neural network framework when faced with a fixed single table downstream task without pre-training?
- **(key)** Our large-scale cross-table pre-training can help improve the effectiveness of downstream tasks by self-supervised masked table modeling pre-training in large-scale unlabeled scenarios and supervised contrastive learning pre-training in lightweight labeled scenarios, respectively.

6.1 Experimental Setup

6.1.1 Datasets. The experimental dataset consists of two parts: upstream large-scale cross-table pre-training datasets and downstream tabular tasks for evaluating the effectiveness of our model and pre-training.

Large-scale cross-table pre-training dataset: We collected more than 2000 high-quality datasets with semantic information of column names and performed some data cleaning, including 1000 labeled datasets and 1000 unlabeled datasets. We call this dataset TabPretNet and describe it in detail in Section 4.

Public downstream tabular tasks: We selected 15 common and high-quality tabular datasets from OpenML-CC18 [8] to evaluate the effectiveness of our model and pre-training method. These downstream datasets contain both binary and multi-class classification tasks. We included the details and source of each dataset in Table 4 & 5 in the Appendix 8.2.

6.1.2 Competing Methods. We conduct experiments on the following "shallow" (eg. tree-based) and neural network-based methods to show the efficacy and efficiency of CT-BERT on tabular learning.

Shallow baselines:

- *Logitic Regression* [62] is a linear classification algorithm that models the relationship between input variables and a binary outcome using a logistic function. It is widely used due to its simplicity, interpretability, and ability to handle large datasets efficiently.
- *Xgboost* [16] is an advanced implementation of gradient boosting algorithms. It has gained great popularity in machine learning competitions (e.g., Kaggle) and has been considered the dominant approach to modeling tabular data for a long time.
- *LightGBM* [31] is another gradient boosting tree framework. It employs a novel approach called "Gradient-based One-Side Sampling" (GOSS) to achieve faster training speeds and lower memory usage.

Neural network-based baselines:

- *MLP (Multilayer Perceptron)* [23] is a basic feed-forward fully connected artificial neural network architecture, but is considered a competitive neural network approach on tabular data.
- *TransTab* [59] is a newly proposed tabular framework that combines column description and table cells as the raw input to a transformer and is the current state-of-the-art tabular model.
- *FT-Transformer* [24] is a adaptation of the Transformer architecture [56] for the tabular data (Feature Tokenizer + Transformer).
- *TabNet* [4] uses sequential attention to simulate the process of tree decision-making, enabling interpretability and more efficient learning on tabular data.
- *VIME* [66] is a self- and semi-supervised learning framework specifically designed for tabular data.
- *SAINT* [50] is a newly proposed hybrid deep learning approach to solving tabular data problems and performs attention over both rows and columns.
- *DCN-v2* [58] is an improved version of Deep & Cross Network (DCN), and claimed to be able to automatically and efficiently captures feature interactions in tabular data.
- *AutoInt* [51] is a click-through prediction, which is a type of structured data task, model. It uses a multi-head self-attentive neural network to learn the high-order feature interactions of input features.

6.1.3 Metrics. We follow previous work [24, 59] using AUC [37] as the main evaluation metric and improve on it using 5-fold cross-validation [3] as the final result. Note that within each fold of the training set, we partitioned 20% as a validation set, which was utilized for hyperparameter selection and early stopping. For the sake of fairness, we employed the identical dataset splitting setting for all baseline algorithms and CT-BERT on all downstream task datasets.

6.1.4 Implementation Details. For details of all baseline implementations see Appendix 8.1, while the settings for all baselines remain consistent across all experiments unless otherwise specified. In the data pre-processing phase, we scale numerical features to $[0, 1]$ by min-max normalization in all methods. For classification features, we use ordinal codes to represent them in all baselines. However, note that in our CT-BERT, we use the raw textual values of the categorical features in order to better exploit their semantic information. CT-BERT uses a 4-layer transformer, where the embedding dimension of the token is 128, the hidden dimension of the middle dense layer is 256, and the self-attention module has 8 heads. We

Table 1: Comparison of CT-BERT with other shallow and NN-based methods. CT-BERT_NoPT is our supervised learning from scratch method (without pre-training), CT-BERT_P_M is our self-supervised MTM pre-training then fine-tuning method, and CT-BERT_P_S is our supervised contrastive learning pre-training then fine-tuning method.

Dataset	Shallow Methods			NN-Based Methods								Our Methods		
	LR	XGB	LightGBM	DCN-v2	AutoInt	MLP	FT-Trans	Saint	TabNet	VIME	TransTab	CT-BERT_NoPT	CT-BERT_P_M	CT-BERT_P_S
pc4	0.8621	0.7264	0.7938	0.8532	0.8741	0.8031	0.8990	0.8957	0.8449	0.8442	0.8882	0.8816	0.8837	**0.9030**
kc1	0.8004	0.6488	0.6623	0.7161	0.7710	0.7894	0.7957	**0.8459**	0.7917	0.7901	0.7945	0.7950	0.7846	0.7880
car	0.7393	0.9950	0.9134	0.9896	0.9664	0.9961	0.9981	0.9588	0.9713	0.9921	0.9039	0.9997	0.9996	**0.9998**
wilt	0.7098	0.8883	0.9249	0.9894	0.9601	0.6406	0.6978	0.9744	0.9934	0.9134	0.9850	0.9930	**0.9946**	0.9937
higgs	0.6346	0.6730	0.6935	0.6435	0.6237	0.6430	0.7063	0.7324	0.5474	0.6354	0.7284	0.6610	0.7002	**0.7348**
adult	0.8360	0.7894	0.8314	0.8923	0.8879	0.9023	0.9161	0.9152	0.9003	0.9128	0.9134	0.9150	**0.9155**	0.9143
climate	0.9449	0.7217	0.7014	0.8549	0.9097	0.4048	**0.9584**	0.8145	0.7951	0.8647	0.9345	0.9164	0.9204	0.9376
credit-g	0.7251	0.6755	0.7152	0.6912	0.7253	0.7370	0.7675	0.7817	0.6630	0.7659	0.7600	0.7701	0.7703	**0.7867**
vehicle	0.8912	0.9286	0.9305	0.9125	0.8883	0.9277	0.9231	0.8053	0.7877	0.7752	0.9178	0.9291	**0.9306**	0.9197
segment	0.9703	0.9929	0.9923	0.9746	0.9881	0.9858	0.9913	0.9809	0.9633	0.9752	0.9922	0.9908	0.9919	**0.9930**
amazon	0.5315	0.5231	0.6012	0.5564	0.5372	0.5461	0.5099	0.5550	0.5190	0.5081	0.5551	0.5698	**0.6092**	0.5313
satimage	0.9722	0.9889	0.9501	0.8023	0.9530	0.9863	0.9867	0.9838	0.9831	0.9126	0.9868	0.9856	**0.9897**	0.9888
phishing	0.9786	0.9669	0.9810	0.9389	0.9789	0.9943	0.9936	0.9923	0.9911	0.9913	0.8296	**0.9949**	**0.9949**	0.9942
mice-protein	0.9973	0.9993	0.9989	0.8894	0.9112	0.9997	0.9987	0.9973	0.9477	0.9579	0.9998	0.9981	**0.9999**	0.9998
cylinder-bands	0.7498	0.8197	0.7706	0.7465	0.7203	0.7070	0.8303	0.7415	0.5640	0.6916	0.8537	0.7629	0.8581	**0.8715**
mean	0.8229	0.8225	0.8307	0.8301	0.8463	0.8042	0.8648	0.8650	0.8175	0.8354	0.8695	**0.8775**	**0.8895**	**0.8904**

use a dropout of 0.3 in all attention layers and feed-forward layers. We choose ReLU for all activation functions. The supervised pre-training method is trained on 1000 labeled datasets, and the self-supervised pre-training method is trained on all 2000 datasets. We train CT-BERT using Adam [33] optimizer with a learning rate in {5e-5, 1e-4, 3e-4}, where the learning rate of the fine-tuning phase will be smaller than that of the pre-training phase. Batch size is in {64, 128, 256}. We use a pre-trained BERT-base-uncased [20] model on Hugging Face[6] to obtain token embeddings that are rich in semantic information. In the pre-training phase, we set the maximum training epoch to 500 for both the supervised contrastive learning and the self-supervised masked table modeling tasks. In the fine-tuning phase, the maximum training epoch is 200 and the patience value is set to 20 for early stopping. Experiments were conducted with 8 GPU V100, Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz, and 128GB RAM. We use the DeepSpeed [44] framework for parallel computation acceleration. DeepSpeed offers a range of optimization techniques, including model parallelism, data parallelism, and mixed-precision training. It can improve the efficiency of our large-scale cross-table pre-training which occupies a large portion of the computational resources in our experiments.

6.2 Overall Performance

In this section, we report the overall performance of CT-BERT. The results are shown in Table 1.

6.2.1 Supervised Learning from Scratch. As can be seen in Table 1, CT-BERT_NoPT outperforms all the existing works on standardized benchmarking datasets on average. Although TransTab [59] greatly outperforms the baseline method, CT-BERT_NoPT is still slightly higher than TransTab on avg. 0.8%. We analyze this due to the fact that CT-BERT_NoPT models at the *feature-level*, while TransTab [59] models at the *token-level*, which may be not reasonable on tabular data. And the experimental results also show that TransTab's performance drops abruptly on some datasets, such as *car* and *phishingweb*. In addition, we found that CT-BERT_NoPT is also comparable to FT-transformer [24] and SAINT [50]. We analyze and believe that CT-BERT_NoPT is essentially the same as these methods on single tabular data, which extract features from table data and then model feature interactions using a similar transformer encoder. However, the difference is that CT-BERT_NoPT can receive input from heterogeneous tables. This gives our approach a natural advantage in cross-table pre-training which is detailed in Section 5.3.

[6]https://github.com/huggingface

6.2.2 Cross-table Pre-training. We mainly compare with the supervised learning from scratch of CT-BERT.

Supervised: In labeled scenarios, our supervised contrastive learning cross-table pre-training model CT-BERT_P_S has achieved state-of-the-art average performance. As evident from the results in Table 1, CT-BERT_P_S outperforms the supervised training from scratch CT-BERT_NoPT by avg. **1.29%** and achieves better performance on 10 out of 15 diverse downstream tabular tasks. Moreover, we observed that CT-BERT_P_S achieves a comparatively competitive performance than masked table modeling self-supervised cross-table pre-training method on average. We analyze that the reason lies in CT-BERT_P_S's ability to fully leverage the label information, enabling the model to learn more powerful sample representations. And self-supervised methods may require a larger amount of training data to achieve significant advancements.

Self-supervised: In large-scale unlabeled scenarios, as can be seen in Table 1, our masked table modeling self-supervised cross-table pre-training model CT-BERT_P_M outperforms the supervised training from scratch CT-BERT_NoPT by avg. 1.2%. And CT-BERT_P_M achieves better performance on **13** out of 15 diverse downstream tabular tasks. It is noteworthy that our cross-table pre-trained model exhibits significant improvements on the *cylinder-bands*, *higgs*, and *Amazon* datasets. We hypothesize that this result can be attributed to the presence of certain tables in the pre-training data that bear close relevance to these downstream tasks. Therefore, we have reason to believe that masked table modeling cross-table pre-training approach on ultra-large-scale datasets

Table 2: Few-shot 5-fold AUC (%) on 6 datasets from the OpenML-CC18 [8].

Our Methods	Datasets						Mean
	vehicle	pc4	adult	phishing	cylinder	car	
shot=5							
CT-BERT_NoPT	0.6771	0.7459	0.7561	0.7389	0.6315	0.7527	0.7170
CT-BERT_P_M	0.7020	**0.7825**	0.8131	0.8435	0.6498	0.7260	0.7528
CT-BERT_P_S	**0.7126**	0.7684	**0.8778**	**0.8596**	**0.7246**	**0.8629**	**0.8010**
shot=10							
CT-BERT_NoPT	0.7440	**0.7721**	0.8082	0.8471	0.6694	0.8681	0.7848
CT-BERT_P_M	0.7514	0.7607	0.8499	0.9023	0.6652	0.8651	0.7991
CT-BERT_P_S	**0.7690**	0.7598	**0.8797**	**0.9424**	**0.7202**	**0.9186**	**0.8316**
shot=20							
CT-BERT_NoPT	**0.8312**	0.7695	0.8312	0.9253	0.6875	0.9591	0.8341
CT-BERT_P_M	0.8028	**0.7826**	0.8601	0.9512	0.6825	0.9464	0.8376
CT-BERT_P_S	0.8227	0.7601	**0.8805**	**0.9546**	**0.7734**	**0.9743**	**0.8609**

is a highly promising approach on the path toward a comprehensive universal table model.

CT-BERT is the first attempt at such large-scale cross-table pre-training. **Our experimental results demonstrate the feasibility of learning shareable knowledge across different tables through cross-table pre-training, which helps the model achieve better generalization on diverse downstream tasks.** Both supervised pre-training and self-supervised pre-training methods achieved good performance. We believe that supervised training requires higher dataset requirements but may be better suited for specific scenarios, while self-supervised training has the potential for greater scalability through larger pre-training datasets in the future.

6.3 Few-shot Learning

As widely recognized, a significant advantage of pre-trained models is that they still work well when the downstream task dataset is relatively scarce, commonly referred to as few-shot learning [67]. This capability stems from that the model can learn rich shareable knowledge from large-scale upstream datasets. In the domain of tabular tasks, there are numerous practical application scenarios characterized by limited data resources, such as medical diagnosis [35]. In such contexts, the exceptional few-shot learning ability of pre-trained models becomes invaluable. Therefore, we conducted extensive experiments to explore the practical effectiveness of CT-BERT in the context of few-shot learning settings.

Specifically, for each downstream classification tabular data, we randomly sampled 5/10/20 samples from each class to construct three new 5-shot/10-shot/20-shot tabular datasets. We then performed both supervised training from scratch and pre-training then fine-tuning on these new few-shot datasets. The experimental results are presented in Table 2. The self-supervised and supervised pre-trained models significantly outperformed the baseline of learning from scratch in the few-shot learning setting. In 5-shot case, CT-BERT_P_S outperforms the training from scratch CT-BERT_NoPT by avg. **8.4%**, and CT-BERT_P_M also surpassed by avg. **3.58%**. Furthermore, we can observe that the pre-trained model exhibits a greater improvement in performance when the number of samples is less. The improvement is most significant in the 5-shot case while is relatively weaker in the 20-shot case. We analyze this as a reasonable phenomenon. The shareable knowledge learned through cross-table pre-training is relatively more valuable when the training data is less. In conclusion, all these experimental results strongly demonstrate the tremendous potential of cross-table pre-training in the context of few-shot learning.

Table 3: Ablation studies of different pooling strategies (mean AUC %)

Pooling Strategy	CT-BERT_NoPT	CT-BERT_P_M	CT-BERT_P_S
No-Pooing	0.8556	0.8630	0.8633
Max	0.8695	0.8883	0.8774
Average	**0.8775**	**0.8895**	**0.8904**
Self-Attention	0.8681	0.8733	0.8710

6.4 Ablation Studies

In order to demonstrate that modeling at the feature level is more effective than previously used word token-level modeling in tabular data, we conducted ablation experiments. Specifically, we do not pool all word token embeddings into one feature embedding but feed them directly into the transformer layer for learning. The experimental result is presented in Table 3 and proves that feature-level modeling is significantly better than word token-level modeling. Additionally, we further evaluated different pooling strategies: average pooling, max pooling, and self-attention [56] pooling. The results are shown in Table 3. Among these strategies, average pooling gives the best results. We tried to analyze the reason that max-pooling may not be able to distinguish between different feature values in some cases. For example, the max value may come from the word token embedding in the column name, which is the same for all the sample rows. The self-attention mechanism may be too complex relative to this simple information extraction. And average pooling can do this task simply and efficiently.

6.5 Further Analysis

6.5.1 Convergence Curves. Figure 4 compares the convergence curves of two paradigms: "training from scratch" and "pre-training then fine-tuning". We observed that pre-training and then fine-tuning leads to faster convergence and better results. This demonstrates that CT-BERT has learned beneficial shareable knowledge for downstream tasks through cross-table pre-training. Furthermore, pre-training and then fine-tuning can achieve reasonable results within a short period of time. This significantly improves the efficiency of executing downstream tasks that do not require high precision. It also partially alleviates the longer training time issue associated with neural network training compared to traditional tree-based machine learning methods [4].

6.5.2 Masking Ratio. Previous research [26] has suggested that a higher mask rate is required to achieve better performance in masked image modeling tasks, whereas a lower mask rate is sufficient for masked language modeling tasks. In this experiment, we further investigate the impact of mask rates on masked table modeling tasks, as shown in Figure 5. We found that the model has high performance between 30% and 50%, with an excessively high mask rate leading to a steep descent, while an excessively low

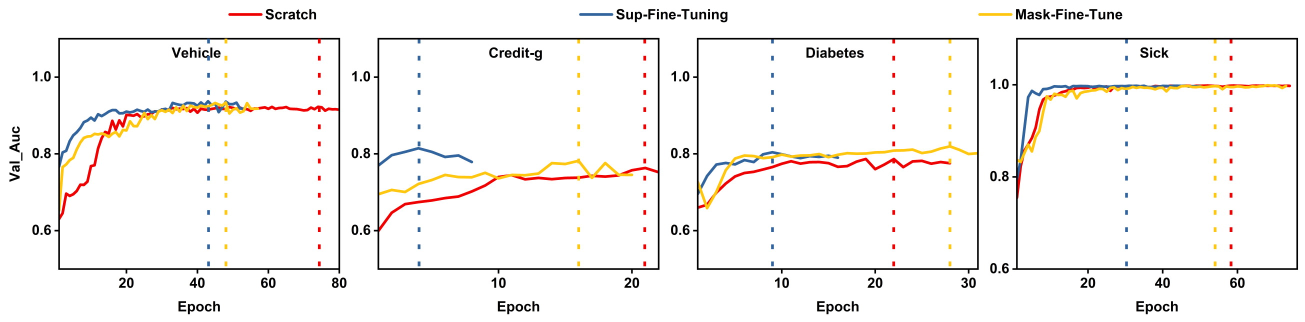


Figure 4: The convergence curves of two paradigms: *training from scratch* and *pre-training then fine-tuning*. The vertical line represents the epoch that reaches the maximum validation set auc.

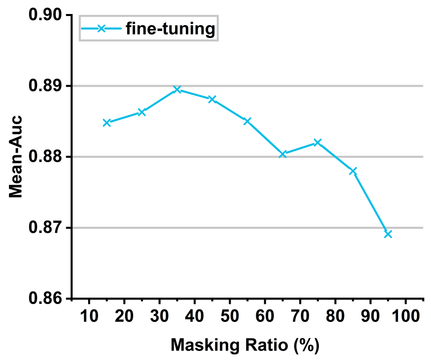
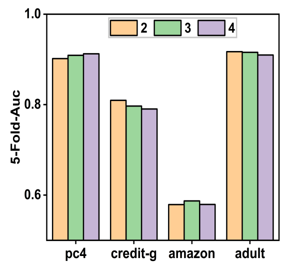
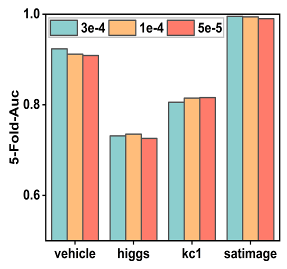


Figure 5: Masking ratio settings in our self-supervised masked table modeling method. We tried 9 different masking ratios, ranging from 15% to 95% with an interval of 10. The best results are achieved at a 35% masking ratio.



(a) Comparison of results with different number of randomly sampled partitions



(b) Comparison of results with different learning rate

Figure 6: Sensitivity analysis for hyperparameters.

mask rate leads to a more moderate descent. We analyze that table data exhibits high information density, where a change in a single feature value can significantly alter the meaning of a sample. So too high a mask rate will cause the model to have difficulty in learning the correct feature relationships.

6.5.3 Hyperparametric Sensitivity Analysis. We analyzed the sensitivity of the number of randomly sampled partitions and the learning rate. We randomly selected some datasets to experiment with the CT-BERT_P_S method. The experimental results are shown in Fig. 6. The settings are consistent with Section 6.1.4 except for the corresponding hyperparameters. It can be seen that CT-BERT is robust to the hyperparameters.

7 CONCLUSION

With CT-BERT and TabPretNet, we hope to initiate the scaled cross-table pre-training for the community of database and data mining community. Speaking humbly, we deem CT-BERT as a pioneer work to scale tabular data pre-training that it works in either a supervised and/or self-supervised manner. We empirically demonstrate that facilitating the pre-training procedure across large-scale tabular datasets indeed offers decent efficacy benefits.

Perceiving it through the lens of the development of current LLMs, our model is still small (50M), which is roughly the same size as BERT-base [20] in spite of CT-BERT being the largest-scaled pre-trained model in tabular modeling thus far. We think that for tabular data pre-training, we are still in the era of the BERT model in NLP tracking back a few years. That is to say, the size of the large model and the volume of the dataset still fall far behind the development of the LLMs, such as ChatGPT or its other rivals [54].

On the bright side, the volume of available tabular data is truly gigantic — wherever a database system is deployed there will be tabular data — but perhaps much more decentralized than the text and vision data. In the future, we hope to explore even further scaling CT-BERT and adapting it to more diversified data domains.

8 APPENDIX

8.1 Baseline architecture and implementation

The setup of our baseline follows the previous work [59] and includes the following methods:

- **Logistic Regression**: Use the default setting of the package Scikit-Learn. The maximum number of estimators is set to 1000.
- **XGBoost**: Implemented based on the XGBoost package. We set the maximum number of estimators in {50, 100, 300} and the max depth in {5, 8, 10}.

- **LightGBM**: Implemented based on the LightGBM. We set the maximum number of estimators in {50, 100, 300} and the max depth in {5, 8, 10}.
- **MLP**: Dense layers with hidden dimensions {256, 256}. Dropout with a rate of 0.1 is used. They are trained with batch size ∈ {16, 32, 64, 128}, learning rate ∈ {5e-5, 1e-4, 1e-3}, and early stopping patience of 5 with 100 maximum epochs.
- **TabNet**: Use the official implementation with the default recommended parameters[7]. Trained with batch size ∈ {16, 32, 64, 128}, learning rate ∈ {1e-4, 1e-3, 2e-2}, $n_a, n_b \in \{8, 16, 64, 128\}$, $\gamma \in \{1.3, 1.5, 1.8\}$, categorical embedding dimension $\in \{1, 8, 16\}$ and early stopping patience of 5 with 100 maximum epochs.
- **DCN-v2**: Use the implementation by paper [24][8]. The number of cross is 2. The dropout rate for the feedforward component is 0.1. MLP part has two dense layers of dimension {256, 128}. Trained with batch size ∈ {16, 32, 64, 128}, learning rate ∈ {5e-5, 1e-4, 1e-3}, and early stopping patience of 10 in 100 maximum epochs.
- **AutoInt**: Use the implementation by paper [24][8]. The attention layer number is set to 2. The attention head number is set to 2. MLP part has two dense layers of dimension 256, 128; dropout deactivated; trained with batch size ∈ {16, 32, 64, 128}, learning rate ∈ {5e-5, 1e-4, 1e-3}, and early stopping patience of 10 in 100 maximum epochs.
- **SAINT**: Use the official implementation[9]. The embedding size is 32 dimensions. 6 transformer layers are used. The number of heads of attention is $\in \{4, 8\}$. The dropout rate is 0.1 in all attention layers and feed-forward layers. Inside the self-attention layer, the q, k, and v vectors are of dimension 16, and in the intersample attention layer, they are of size 64.
- **FT-Transformer**: Use the official implementation[10]. Feed-forward component has 128 dimensions. 2 transformer layers are used. The number of heads of attention is $\in \{2, 4, 8\}$. The dropout rate is 0.1.
- **VIME**: We reproduce it by PyTorch [41] based on the original official implementation[11]. We train the model on all training data taking mask rate 0.3, batch size 128, learning rate 1e-4, and 10 epochs. During the fine-tuning phase, we add a classifier after the encoder with three dense layers of 100 dimensions and ReLU activations. Trained with batch size ∈ {16, 32, 64, 128}, learning rate ∈ {5e-5,1e-4,1e-3}, and early stopping patience of 10 in 100 maximum epochs.
- **TransTab**: Use the official implementation[12]. Token embedding has 128 dimensions. 2 transformer layers are used. The number of heads of attention is 8. We train the model on all downstream task data taking batch size 64, learning rate 1e-4, dropout rate 0, and early stopping patience of 10 in 100 maximum epochs. We run the pre-training, transfer learning, and vanilla supervised training methods in the paper, and take the highest score.

8.2 Details of the downstream task datasets

The downstream task datasets are mainly from the OpenML-CC18 benchmark [8].

Table 4: Statistical Information of Downstream Task Datasets

Dataset Name	Samples	Numerical	Categorical	label classes
pc4	1458	37	0	2
kc1	2109	21	0	2
car	1728	0	6	4
wilt	4839	5	5	2
higgs	2000	24	0	2
adult	48842	6	8	2
climate	540	20	0	2
credit-g	1000	7	13	2
vehicle	846	18	0	4
segment	2310	19	0	7
amazon	2000	0	9	2
satimage	6430	36	0	6
phishing	11055	0	30	2
mice-protein	1080	77	4	8
cylinder-bands	540	18	21	2

Table 5: Downstream Task Datasets Source

Dataset Name	Link
pc4	https://www.openml.org/d/1049
kc1	https://www.openml.org/d/1067
car	https://archive.ics.uci.edu/dataset/19/car+evaluation
wilt	https://archive.ics.uci.edu/dataset/285/wilt
higgs	https://www.openml.org/d/44422
adult	https://archive.ics.uci.edu/dataset/2/adult
climate	https://archive.ics.uci.edu/dataset/252/climate+model+simulation+crashes
credit-g	https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data
vehicle	https://archive.ics.uci.edu/dataset/149/statlog+vehicle+silhouettes
segment	http://archive.ics.uci.edu/dataset/50/image+segmentation
amazon	https://www.openml.org/d/44712
satimage	https://archive.ics.uci.edu/dataset/146/statlog+landsat+satellite
phishing	https://archive.ics.uci.edu/dataset/327/phishing+websites
mice-protein	https://archive.ics.uci.edu/dataset/342/mice+protein+expression
cylinder-bands	https://archive.ics.uci.edu/dataset/32/cylinder+bands

[7]https://github.com/dreamquark-ai/tabnet
[8]https://github.com/Yura52/tabular-dl-revisiting-models
[9]https://github.com/somepago/saint
[10]https://github.com/Yura52/rtdl
[11]https://github.com/jsyoon0823/VIME
[12]https://github.com/RyanWangZf/transtab

REFERENCES

[1] 2022. https://openai.com/blog/chatgpt.

[2] Ahmad Ahmadov, Maik Thiele, Julian Eberius, Wolfgang Lehner, and Robert Wrembel. 2015. Towards a hybrid imputation approach using web tables. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*. IEEE, 21–30.

[3] Davide Anguita, Luca Ghelardoni, Alessandro Ghio, Luca Oneto, and Sandro Ridella. 2012. The 'K'in K-fold cross validation. In *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. i6doc. com publ, 441–446.

[4] Sercan Ö Arik and Tomas Pfister. 2021. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 6679–6687.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[6] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. 2021. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147* (2021).

[7] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: Entity linking in web tables. In *The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*. Springer, 425–441.

[8] Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. 2017. Openml benchmarking suites. *arXiv preprint arXiv:1708.03731* (2017).

[9] Casper Solheim Bojer and Jens Peder Meldgaard. 2021. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting* 37, 2 (2021), 587–603.

[10] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data: Application to word-sense disambiguation. *Machine Learning* 94 (2014), 233–259.

[11] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[13] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1, 1 (2008), 538–549.

[14] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1, 1 (2008), 538–549.

[15] Michael J Cafarella, Alon Y Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. 2008. Uncovering the Relational Web.. In *WebDB*. Citeseer, 1–6.

[16] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[17] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[19] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* 51, 1 (2022), 33–40.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[22] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. 2017. Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16*. Springer, 260–277.

[23] Matt W Gardner and SR Dorling. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* 32, 14-15 (1998), 2627–2636.

[24] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems* 34 (2021), 18932–18943.

[25] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[26] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.

[27] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349* (2020).

[28] Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 8 (1998), 832–844. https://doi.org/10.1109/34.709601

[29] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. 2020. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678* (2020).

[30] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[31] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[32] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems* 33 (2020), 18661–18673.

[33] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[34] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643* (2023).

[35] Igor Kononenko. 2001. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine* 23, 1 (2001), 89–109.

[36] Guang Liu, Jie Yang, and Ledell Wu. 2022. PTab: Using the Pre-trained Language Model for Modeling Tabular Data. *arXiv preprint arXiv:2209.08060* (2022).

[37] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. 2008. AUC: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography* 17, 2 (2008), 145–151.

[38] Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. 2012. Chex: statically vetting android apps for component hijacking vulnerabilities. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 229–240.

[39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[40] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[42] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[44] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3505–3506.

[45] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*. 74–84.

[46] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2015. Matching html tables to dbpedia. In *Proceedings of the 5th international conference on web intelligence, mining and semantics*. 1–6.

[47] Justyna Sarzynska-Wawer, Aleksander Wawer, Aleksandra Pawlak, Julia Szymanowska, Izabela Stefaniak, Michal Jarkiewicz, and Lukasz Okruszek. 2021. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research* 304 (2021), 114135.

[48] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155* (2018).

[49] Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644* (2021).

[50] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. 2021. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342* (2021).

[51] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[52] Ningyuan Sun, Xuefeng Yang, and Yunfeng Liu. 2020. Tableqa: a large-scale chinese text-to-sql dataset for table-aware sql generation. *arXiv preprint arXiv:2006.06434* (2020).

[53] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[54] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[55] Talip Ucar, Ehsan Hajiramezanali, and Lindsay Edwards. 2021. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems* 34 (2021), 18853–18865.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[57] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

[58] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. , 1785–1797 pages.

[59] Zifeng Wang and Jimeng Sun. 2022. Transtab: Learning transferable tabular transformers across tables. *arXiv preprint arXiv:2205.09328* (2022).

[60] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28.

[61] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973* (2013).

[62] Raymond E Wright. 1995. Logistic regression. (1995).

[63] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. 2022. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9653–9663.

[64] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[65] Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. arXiv:2005.08314 [cs.CL]

[66] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. 2020. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems* 33 (2020), 11033–11043.

[67] Jian-Guo Zhang, Kazuma Hashimoto, Wenhao Liu, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2020. Discriminative nearest neighbor few-shot intent detection by transferring natural language inference. *arXiv preprint arXiv:2010.13009* (2020).

[68] Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems* 31 (2018).