# FATA-Trans: Field And Time-Aware Transformer for Sequential Tabular Data

Dongyu Zhang
dzhang5@wpi.edu
Worcester Polytechnic Institute
Worcester, Massachusetts, USA

Liang Wang
liawang@visa.com
Visa Research
Palo Alto, California, USA

Xin Dai
xidai@visa.com
Visa Research
Palo Alto, California, USA

Shubham Jain
shubhjai@visa.com
Visa Research
Palo Alto, California, USA

Junpeng Wang
junpenwa@visa.com
Visa Research
Palo Alto, California, USA

Yujie Fan
yufan@visa.com
Visa Research
Palo Alto, California, USA

Chin-Chia Michael Yeh
miyeh@visa.com
Visa Research
Palo Alto, California, USA

Yan Zheng
yazheng@visa.com
Visa Research
Palo Alto, California, USA

Zhongfang Zhuang
zzhuang@visa.com
Visa Research
Palo Alto, California, USA

Wei Zhang
wzhan@visa.com
Visa Research
Palo Alto, California, USA

## ABSTRACT

Sequential tabular data is one of the most commonly used data types in real-world applications. Different from conventional tabular data, where rows in a table are independent, sequential tabular data contains rich contextual and sequential information, where some fields are *dynamically* changing over time and others are *static*. Existing transformer-based approaches analyzing sequential tabular data overlook the differences between dynamic and static fields by replicating and filling static fields into each record, and ignore temporal information between rows, which leads to three major disadvantages: (1) computational overhead, (2) artificially simplified data for masked language modeling pre-training task that may yield less meaningful representations, and (3) disregarding the temporal behavioral patterns implied by time intervals. In this work, we propose FATA-Trans, a model with two field transformers for modeling sequential tabular data, where each processes static and dynamic field information separately. FATA-Trans is *field*- and *time*-aware for sequential tabular data. The *field*-type embedding in the method enables FATA-Trans to capture differences between static and dynamic fields. The *time*-aware position embedding exploits both order and time interval information between rows, which helps the model detect underlying temporal behavior in a sequence. Our experiments on three benchmark datasets demonstrate that the learned representations from FATA-Trans consistently outperform state-of-the-art solutions in the downstream tasks. We also present visualization studies to highlight the insights captured by the learned representations, enhancing our understanding of the underlying data. Our codes are available at https://github.com/zdy93/FATA-Trans.

## CCS CONCEPTS

• **Applied computing** → **Enterprise computing**; • **Computing methodologies** → **Knowledge representation and reasoning**.

## KEYWORDS

transformer, sequential tabular data, field and time aware

**ACM Reference Format:**
Dongyu Zhang, Liang Wang, Xin Dai, Shubham Jain, Junpeng Wang, Yujie Fan, Chin-Chia Michael Yeh, Yan Zheng, Zhongfang Zhuang, and Wei Zhang. 2023. FATA-Trans: Field And Time-Aware Transformer for Sequential Tabular Data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3614879


*CIKM '23, October 21–25, 2023, Birmingham, United Kingdom*

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614879

## 1 INTRODUCTION

Sequential tabular data is one of the most commonly used data types in real-world applications such as medical diagnosis[43], recommendation systems[44], click-through-rate (CTR) prediction[31], and transaction anomaly detection[46]. In a sequential tabular dataset, there are multiple rows and columns, where each row corresponds to a *record* and each column represents a *field*. An example of sequential tabular data is the credit card transaction data which captures purchasing activities of cardholders over time. Another example is the user review data from a website, which logs users'

comments and ratings for the products they rented or bought in the past.

Figure 1 depicts a screenshot of a modified sequential tabular dataset derived from a synthetic transaction dataset created by [1, 33]. Each row in the dataset represents a transaction record. The first six rows form a transaction sequence associated with User 0's card 0, while the last six rows represent another sequence linked to User 1's card 1. Within these transaction sequences, certain fields in the data are *dynamic*, capturing a user's *local* activities that are transient in nature. Conversely, other fields are *static*, reflecting a user's *global* identity that remains stable over time. For example, in Figure 1, dollar amounts are dynamic as they tend to vary from one transaction to another. On the other hand, card type (e.g., Debit Card) and issuer bank (e.g., Example Bank) remain constant throughout the transactions. Both dynamic and static fields play distinct but significant roles within a transaction sequence.

The concept of dynamic and static fields extends beyond the raw fields present in transaction data and can also include derived fields or features. For example, a derived static field could be the average dollar amount of each user's transactions over a specific time period. In the context of user review data, the ratings given by a user to various products can be considered dynamic since they can change over time. However, the user's average rating over a previous time period would be considered a static field, as it represents an aggregated value that remains constant within that specific period.

Furthermore, in sequential tabular datasets, capturing time and order information is crucial for understanding user behaviors. Figure 2 provides an example of transaction sequences associated with three different cardholders. Although the field values across the three sequences are identical, the timing and order of the transactions differ. The presence of time and order information becomes critial in identifying abnormal transaction behaviors. In the given example, the first sequence exhibits no abnormal transactions, but the other sequences contain abnormal transactions. Hence, considering time and order information becomes essential during model training to effectively capture user behaviors within sequential tabular datasets.

Extensive research has been dedicated to modeling sequential tabular data. In the early stage, recurrent neural networks (RNNs) were employed and demonstrated outstanding performance[18, 42, 47]. However, more recent efforts have primarily focused on transformer-based models due to their superior ability to capture long-range dependencies and effectively handle sequential data.

The pioneering work of Kang and McAuley[22] has inspired numerous subsequent studies, leading to the development of various innovative transformer-based model architectures specifically tailored for modeling sequential tabular data [6–8, 10, 33, 39, 41, 45, 48]. Among these architectures, TabBERT, initially developed by Padhi et al.[33] and further enhanced in subsequent works[17, 30], presents a powerful and comprehensive framework for processing sequential tabular data. TabBERT is a pre-trained transformer architecture trained using masked language modeling (MLM)[11], where it predicts masked tokens. It adopts a hierarchical approach to encode a series of transactions using two transformers. At the first level, the transformer processes individual tabular rows (transactions) by considering each field value as a token, thereby generating

| User | Card | Date | Time | Amount | Method | Zip | ... | Avg Amount | Card Type | Issue Bank |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2020-1-3 | 6:04 | $123.17 | Chip | 78335 | ... | $54.00 | Debit Card | Example Bank |
| 0 | 0 | 2020-1-5 | 17:35 | $4.85 | Online | | ... | $54.00 | Debit Card | Example Bank |
| 0 | 0 | 2020-1-6 | 6:16 | $121.30 | Chip | 78266 | ... | $54.00 | Debit Card | Example Bank |
| 0 | 0 | 2020-1-10 | 6:02 | $119.45 | Chip | 91750 | ... | $54.00 | Debit Card | Example Bank |
| 0 | 0 | 2020-1-16 | 9:52 | $34.19 | Chip | 91750 | ... | $54.00 | Debit Card | Example Bank |
| 0 | 0 | 2020-1-21 | 5:25 | $140.61 | Chip | 91750 | ... | $54.00 | Debit Card | Example Bank |
| ... | ... | ... | ... | ... | ... | | ... | ... | ... | ... |
| 1 | 1 | 2020-1-1 | 18:40 | $61.26 | Chip | 11363 | ... | $44.47 | Credit Card | Test Bank |
| 1 | 1 | 2020-1-11 | 6:41 | $163.55 | Chip | 11363 | ... | $44.47 | Credit Card | Test Bank |
| 1 | 1 | 2020-1-12 | 10:14 | $57.54 | Chip | 11420 | ... | $44.47 | Credit Card | Test Bank |
| 1 | 1 | 2020-1-15 | 12:01 | $15.75 | Chip | 11364 | ... | $44.47 | Credit Card | Test Bank |
| 1 | 1 | 2020-1-19 | 4:21 | $132.95 | Chip | 11363 | ... | $44.47 | Credit Card | Test Bank |
| 1 | 1 | 2020-1-20 | 13:11 | $1.00 | Swipe | 11363 | ... | $44.47 | Credit Card | Test Bank |

**Figure 1: A screenshot of a synthetic sequential tabular dataset showing transaction records. Each row represents a transaction associated with a user and card identifier (columns 1 and 2). A user can own multiple cards. The dataset contains dynamic fields (columns 3 to 7) that vary across transactions and static fields (last three columns) that remain constant. The dataset includes two distinct sequences of transactions: the first six rows and the last six rows.**

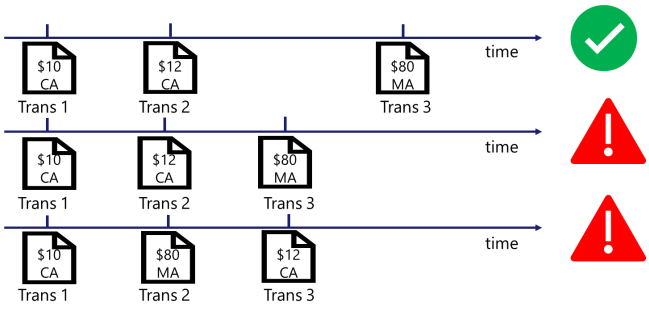



**Figure 2: Example of transaction record sequences with the same field values but different order and timing. Abnormal transactions in sequences 2 and 3 cannot be detected based solely on field values. This highlights the significance of time and order information in detecting abnormal transaction patterns.**

transaction embeddings. The second-level transformer takes these transaction embeddings as input and produces sequence embeddings.

While TabBERT is applicable to a wide range of sequential tabular data, it has two limitations. First, although it can accommodate both dynamic and static fields, it does not differentiate between them. Static fields are replicated in every record within a sequence, leading to computational overhead when multiple static fields or lengthy sequences are present. Additionally, since TabBERT employs MLM to predict masked tokens, predicting a masked static field in one record becomes relatively easy given the same field in other records. This can potentially hinder the model's ability to learn meaningful representations for sequential tabular data during pre-training. Second, although TabBERT leverages order information through the position embedding layer, it fails to consider the critical time information necessary to capture important user behavior patterns in a sequence, as depicted in Figure 2. It is important to

note that these limitations are not exclusive to TabBERT; they also exist in other recently proposed transformer-based architectures dealing with multivariate sequences[8, 41, 45, 48].

In general, previous methods have the following limitations:

- *Failure to distinguish static and dynamic fields*: Static and dynamic fields in sequential tabular datasets have distinct roles and impacts. However, previous works did not differentiate between these two types of fields, leading to an inability to capture their unique characteristics.
- *Negative effect caused by static field replication*: Previous approaches replicated static fields across records, resulting in unnecessary computational overhead and an excessive reduction in the complexity of the model's pre-training task.
- *Failure to utilize both time and order information*: Time and order information are crucial in tasks involving sequential tabular data. However, previous methods either neglected or only partially incorporated the influence of time and order information. A more effective design is necessary to fully leverage the importance of both time and order information.

To address these limitations, we present FATA-Trans, an innovative transformer-based architecture designed to handle sequential tabular datasets while considering field and time information. Our proposed method employs a hierarchical approach for processing input record sequences. At the first level, our method incorporates two field transformers: a static field transformer and a dynamic field transformer. These transformers independently encode the static fields and dynamic fields within each record, resulting in a static field embedding and a series of dynamic field embeddings. At the second level of the architecture, the generated embeddings from the first level are utilized to create sequence embeddings. Here, a field-type embedding is introduced to discern between static and dynamic fields. Additionally, a time-aware position embedding is employed in the second-level transformer to capture time and order information.

The contributions of our paper are as follows: We propose FATA-Trans, an innovative transformer-based architecture that learns representations of sequential tabular datasets. The architecture consists of three key components: (1) two field transformers at the first level that process static and dynamic fields separately, (2) a field-type embedding in the second-level transformer that can distinguish static and dynamic record embeddings, and (3) a customized time-aware position embedding that considers the impact of both time and order of records in a sequence.

Our experimental results demonstrate that the learned embeddings from FATA-Trans consistently yield improved performance compared to state-of-the-art solutions. Notably, the pre-training process of FATA-Trans outperforms TabBERT in terms of speed. Furthermore, we employed visualization techniques to explore the extracted embeddings from the pre-trained model, uncovering meaningful patterns within the data.

## 2 METHODOLOGY

### 2.1 Problem Definition

In this study, we investigate the task of learning effective representations for sequential tabular datasets using a pre-training framework based on transformer models. The process is visually depicted in Figure 3, which uses a record sequence as an illustrative example. Each record represents a row in the tabular dataset, and records associated with the same identifier form a sequence. In Figure 3, all these records are associated with a specific user. The sequence comprises multiple records, each containing both static and dynamic fields. Static fields maintain consistent values throughout the sequence, while dynamic fields exhibit variations over time. Additionally, the order and time information of the records are provided.

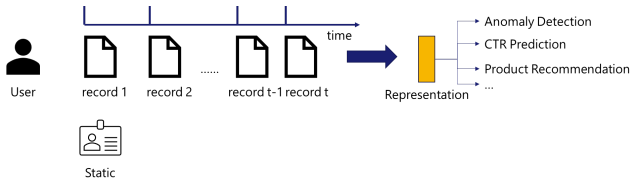



**Figure 3: Record sequence representation learning with both static and dynamic fields and time and order information. Given a sequence of records, the sequence is associated with both static fields (does not change over records) and dynamic fields (changes over records). The order and time information of records are also given. The goal is to learn a useful representation of the sequence of records that can be used in downstream tasks.**

More formally, given a sequential tabular dataset with $m$ records (rows), each record $x_i$ is composed of $n_s$ static fields and $n_d$ dynamic fields. An input, $X$, to our proposed method is represented as a windowed sequence of $l$ time-dependent rows (records) $x_i$,

$$X = [x_0, x_1, ..., x_{l-1}] \tag{1}$$

$$x_i = \{u_i^{s,0}, u_i^{s,1}, ..., u_i^{s,n_s-1}, u_i^{d,0}, u_i^{d,1}, ..., u_i^{d,n_d-1}\} \tag{2}$$

where $l$ ($\ll m$) is the number of consecutive records selected with a window offset (or stride), $u_i^{s,j}$ are static fields, and $u_i^{d,k}$ are dynamic fields. In the windowed sequence $X$, we assume that static fields always keep a constant value, that is $u_i^{s,j} = u^{s,j} \quad \text{for } i \in [0, l-1]$.

For each record $x_i$, we are provided with the creation time $t_i$. To simplify and maintain generality, we set the initial time $t_0$ as 0. $t_i$ represents the time interval between the creation time of record $x_i$ and $x_0$. These record sequences can be effectively utilized for specific tasks, such as detecting anomalies in credit card transactions, by leveraging the time and order information of the transactions.

Prior to training a model for specific tasks, pre-training can be conducted to generate useful representations for record sequences. The pre-trained model can then be fine-tuned for specific downstream tasks. The primary objective of the pre-training task is to train a transformer-based model $f_\theta : X \rightarrow R$, where $R$ denotes a sequence of representations for records,

$$R = [r_0, r_1, ..., r_{l-1}] \tag{3}$$

where $r_i \in \mathbb{R}^{n_e}$. Here, $n_e$ is the dimension of learned representations.

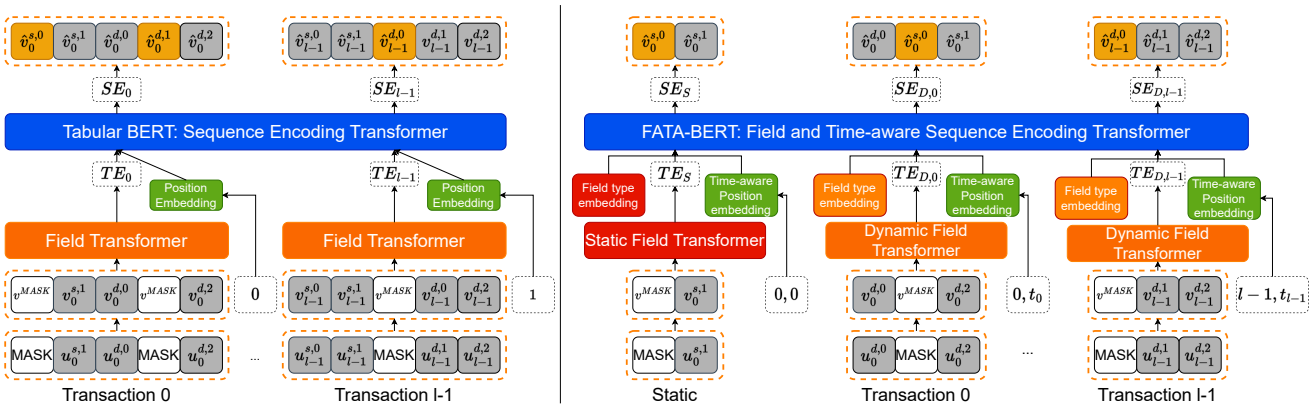



**Figure 4: TabBERT framework (left) and FATA-Trans framework (right). TabBERT framework processes record sequences in a hierarchical fashion, but it does not separate static and dynamic fields. Static fields are replicated over records. Also, position embedding in TabBERT does not consider time interval between records. FATA-Trans framework uses static and dynamic field transformers to process two types of fields separately. Static Fields are not replicated over records. Field type embedding distinguishes dynamic and static fields. Time-aware position embedding utilizes both time interval and sequence information.**

## 2.2 Previous Method: TabBERT

Our proposed method, FATA-Trans, is a variant of the TabBERT framework introduced in [33]. TabBERT was originally designed for modeling transaction sequences, where each *transaction* can be considered as a *record* in a general sequential tabular dataset. TabBERT defines each field on its own local vocabulary and quantizes numerical fields so that both categorical and numerical values can be represented by a finite vocabulary.

TabBERT encodes the sequence of transactions in a hierarchical fashion. As shown in the left panel of Figure 4, TabBERT first uses a field transformer to process each transaction individually, creating transaction embeddings (*TE*). Then these transaction embeddings are fed into the second-level transformer to create sequence embeddings (*SE*). In this setting, the field transformer takes both static and dynamic field tokens as input. Because the value of each static field remains unchanged across transactions, static field tokens are replicated $l$ times in every input. This replication of static field tokens can lead to substantial computational resource usage if $n_s$ or $l$ is large. TabBERT uses the MLM procedure proposed in [11] for pre-training. In MLM procedure, a certain percentage of the input tokens are masked at random, and then the model predicts those masked tokens. However, in TabBERT, if a static field token from one transaction is masked, the model can easily predict it by refernceing the same static field in other transactions. Consequently, the pre-training task becomes too simple, and the transformer fails to capture crucial relationships within transaction sequences. Furthermore, TabBERT does not incorporate time interval information. While the position embedding layer injects transaction order information into the second-level transformer, the influence of time intervals is overlooked. This omission prevents TabBERT from fully capturing the temporal aspect of the data.

## 2.3 Proposed Method: FATA-Trans

Our proposed method, Field And Time-Aware Transformer for Sequential Tabular Data (FATA-Trans), is depicted in the right panel of Figure 4. FATA-Trans consists of two levels: (1) At the first level, the static and dynamic fields are processed separately by the *Static Field Transformer* and *Dynamic Field Transformer* to generate record embeddings. (2) At the second level, a field type embedding distinguishes static and dynamic record embeddings, and a customized time-aware position embedding considers the impact of both time and order of records in a sequence. The record embedding, field type embedding, and time-aware position embedding are element-wise summed together and fed into the *field and time-aware sequential encoding transformer (FATA-BERT)* to generate sequence embeddings. The generated sequence embeddings can then be used for downstream tasks.

### 2.3.1 *Dynamic Field Transformer And Static Field Transformer.*

As shown in the right panel of Figure 4, FATA-Trans uses two field transformers to process static and dynamic fields separately. This distinction is crucial as static and dynamic fields represent different types of information in the record sequences. By processing them separately, FATA-Trans reduces computational overhead and enables the model to capture important patterns within the sequences. The transformation of raw field values into tokens follows a similar procedure to TabBERT. Both dynamic and static fields are tokenized. Subsequently, a random masking process is applied to some of these tokens, as described below:

$$v_i^{d,j} = \text{ConvertToVocab}(u_i^{d,j}), \text{ for } i \in [0, l-1], j \in [0, n_d - 1] \tag{4a}$$

$$\tilde{v}_i^{d,j}, \mathbb{I}_i^{d,j} = \text{RandomMask}(v_i^{d,j}), \text{ for } i \in [0, l-1], j \in [0, n_d - 1] \tag{4b}$$

$$v^{s,j} = \text{ConvertToVocab}(u^{s,j}), \text{ for } j \in [0, n_s - 1] \tag{4c}$$

$$\tilde{v}^{s,j}, \mathbb{I}^{s,j} = \text{RandomMask}(v^{s,j}), \text{ for } j \in [0, n_s - 1] \tag{4d}$$

In Equation 4, $\mathbb{I}$ indicates whether a token is masked or not. If $\mathbb{I} = 0$, the token is replaced by [MASK]. This random masking procedure is for model pre-training only. After that, the Dynamic field transformer processes each record individually with only dynamic

fields as input. The Static field transformer processes the static fields only once without replication. This helps to reduce computational overhead and prevents the model from seeing the masked static field tokens repeatedly. The Static field transformer ($f_{\theta_{sf}}$) produces the static record representation $TE_S$, and the Dynamic field transformer ($f_{\theta_{df}}$) produces dynamic record representations $TE_{D,0}, TE_{D,1}, ..., TE_{D,l-1}$ as follows:

$$TE_S = f_{\theta_{sf}}([\tilde{v}^{s,0}, \tilde{v}^{s,1}, ..., \tilde{v}^{s,n_s-1}]) \tag{5a}$$

$$TE_{D,i} = f_{\theta_{df}}([\tilde{v}_i^{d,0}, \tilde{v}_i^{d,1}, ..., \tilde{v}_i^{d,n_d-1}]), \text{ for } i \in [0, l-1] \tag{5b}$$

*2.3.2 FATA-BERT: Field and Time-Aware Sequential Encoding Transformer.* In the second level of the proposed architecture, we aim to capture the differences and relationships between static and dynamic fields, and utilize time interval information. To achieve this, we have customized the input representation for the second-level transformer, which we refer to as the Field and Time-Aware Sequential Encoding Transformer (FATA-BERT). FATA-BERT is mostly based on BERT-base [11] architecture. However, differing from the original BERT and TabBERT models, the input representation of FATA-BERT is constructed by adding the corresponding record, field type, and time-aware position embeddings.

The field type embedding is a lookup table that stores embeddings of both static and dynamic field types. For $TE_S$, the corresponding field type embedding $FE_S$ is the static type embedding. On the other hand, for each $TE_{D,i}$, the corresponding field type embedding $FE_D$ is the dynamic type embedding. The embeddings of each field type are updated during the training procedure.

The time-aware position embedding $P(i)$ is defined as a vector of length $d$, where each element $p_{(i,j)}$ is defined by:

$$TPos(i) = w_p * i + w_t * t_i + b \tag{6a}$$

$$p_{(i,j)} = \begin{cases} \sin \frac{TPos(i)}{10000(\frac{2j}{d})} & \text{if } j \text{ is even} \\ \cos \frac{TPos(i)}{10000(\frac{2j}{d})} & \text{if } j \text{ is odd} \end{cases} \tag{6b}$$

In this context, we use the variables $i$ and $j$ to represent the position index and time-aware position embedding dimension, respectively. The $TPos(i)$ function is used to merge information from both the position index $i$ and time interval $t_i$, which helps the model to capture time-aware position information through the trainable parameters $w_p$, $w_t$, and $b$, enabling the model to learn a flexible function.

It is important to note that for dynamic fields in the $i$th record, the corresponding time-aware position embedding is represented as $P(i)$. On the other hand, for static fields, we set $i$ to 0 and $t_i$ to $t_0$ to obtain the time-aware position embedding. This implies that the time-aware embedding for static fields is also represented as $P(0)$, which is the same as the time-aware embedding for dynamic fields in the 0th record.

The input representation for static fields ($IE_S$), and $i$th record's dynamic fields ($IE_{D,i}$) are defined by the following formula:

$$IE_S = TE_S + P(0) + FE_S \tag{7a}$$

$$IE_{D,i} = TE_{D,i} + P(i) + FE_D \tag{7b}$$

We feed $[IE_S, IE_{D,0}, IE_{D,1}, ..., IE_{D,l-1}]$ into FATA-BERT. The last layer of FATA-BERT generates a series of sequence embeddings $[SE_S, SE_{D,0}, SE_{D,1}, ..., SE_{D,l-1}]$ according to:

$$\begin{aligned}[SE_S, SE_{D,0}, SE_{D,1}, ..., SE_{D,l-1}] = \\ f_{\theta_{fata}}([IE_S, IE_{D,0}, IE_{D,1}, ..., IE_{D,l-1}])\end{aligned} \tag{8}$$

Here, $SE_S$ represents embeddings for the static fields, and $SE_{D,i}$ represents embeddings for the dynamic fields in the $i$th record. These sequence embeddings are then fed into the classification head ($f_{\theta_{cls}}$) to predict each token in the input record as follows:

$$\begin{aligned}&[\Pr_{\theta_{cls}}(v^{s,0}), \Pr_{\theta_{cls}}(v^{s,1}), ..., \Pr_{\theta_{cls}}(v^{s,n_s-1}), \\ &\Pr_{\theta_{cls}}(v_0^{d,0}), \Pr_{\theta_{cls}}(v_0^{d,1}), ... \Pr_{\theta_{cls}}(v_0^{d,n_d-1}), \\ &\Pr_{\theta_{cls}}(v_1^{d,0}), \Pr_{\theta_{cls}}(v_1^{d,1}), ..., \Pr_{\theta_{cls}}(v_{l-1}^{d,n_d-1}), ] \\ &= f_{\theta_{cls}}([SE_S, SE_{D,0}, SE_{D,1}, ..., SE_{D,l-1}])\end{aligned} \tag{9}$$

In Equation 9, $\Pr_{\theta_{cls}}(v)$ represents the prediction probability for each token. Note that when we calculate the cross-entropy loss for the MLM task, as defined by the following formula, we only consider tokens that have been masked (masked indicator $\mathbb{I} = 0$ in our setting):

$$\ell = -\sum_{j=0}^{n_s-1}(1-\mathbb{I}^{s,j})\log \Pr_{\theta_{cls}}(v^{s,j}) - \sum_{i=0}^{l-1}\sum_{j=0}^{n_d-1}(1-\mathbb{I}_i^{d,j})\log \Pr_{\theta_{cls}}(v_i^{d,j}) \tag{10}$$

## 3 EXPERIMENTS

We compared the performance of FATA-Trans with two powerful baseline models on three benchmark datasets. We also compared FATA-Trans to TabBERT which served as an inspiration for our study.

### 3.1 Datasets and Tasks

*Synthetic Transaction Dataset - Transaction Anomaly Detection Task.* This synthetic dataset was created by [1, 33] for credit card transactions [1]. It contains 24,386,900 transactions from 2,000 users' 6,139 cards, covering a period of 1991 through 2020. Each transaction has attributes such as transaction time, merchant category code (MCC), transaction location, transaction type (chip, swipe or online), transaction amount, and an anomaly label indicating abnormal transactions. Some static fields were derived from these attributes such as average and standard deviation of transaction amounts, the most frequent MCC observed in the card transaction history, and the most frequent transaction type for each user.

The objective of this study was to predict whether the last transaction in a windowed sequence of card transactions is abnormal or not. To create these windowed sequences for each card, 10 consecutive transactions were combined in a time-dependent manner, as described in [30].

*Amazon Product Reviews Dataset - Reviewer Rating Prediction Task.* This dataset, collected by [32], consists of product reviews and metadata from Amazon [2]. Each review record includes fields such as review rating (range from 1 to 5), verification status, review

[1] https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions
[2] https://nijianmo.github.io/amazon/index.html

time, reviewerID, asin (product ID). Static fields were created based on these attributes, including the average of historical ratings, the count of historical ratings, the percentage of low ratings (rating lower or equal to 3), and the percentage of high ratings (rating higher or equal to 4). This dataset contains 233.1 million reviews. In this study, we used two subsets under two categories: 5-core Movies and TV dataset (3,410,019 reviews) and 5-core Electronics dataset (6,739,590 reviews). Here, "5-core" means that all the remaining users and items in the dataset have at least 5 reviews each.

The objective of this task was to predict whether the last review in a reviewer's windowed sequence of reviews is a high rating or a low rating. The length of the windowed review sequence is set as 10.

## 3.2 Compared Methods

We compared FATA-Trans with the following methods:

*LightGBM.* LightGBM, developed by Microsoft[24], belongs to the family of gradient boosting trees [13] which have been the dominant solutions in various Kaggle and other industry competitions [21] and used by researchers as both inspiration and the standard by which to compare model performance [2, 12, 15, 16, 36]. We chose LightGBM as our first baseline because of its superior performance in modeling tabular data.

*RNN.* Recurrent neural networks (RNNs) have demonstrated remarkable performance in domains involving sequential data and the availability of user identifiers (e.g., credit card numbers or login IDs) [5, 18, 28, 29, 46, 47]. Thus, we selected RNNs as our second baseline due to the sequential nature of our data and the presence of user identifiers. Specifically, we chose the gated recurrent unit (GRU) RNN [9], which uses less memory and is faster to train compared with another commonly used RNN architecture known as long short-term memory (LSTM) [19].

*TabBERT.* The TabBERT model, proposed by [33], follows a hierarchical structure, where the first-level field transformer generates transaction embeddings from individual transactions. These embeddings are then used as input for the second-level transformer to generate sequence embeddings. However, it is important to note that TabBERT does not take into account time interval information when computing position embeddings, and it does not distinguish between static and dynamic fields. To ensure comparability, we included the time interval as a dynamic field in TabBERT, allowing it to leverage transaction time information.

## 3.3 Experimental Setup

*Data Preprocessing.* For the synthetic credit card transaction dataset, we divided all transactions occurring before 2018 into separate training and validation datasets. Transactions that occurred after 2018 were designated as the test (holdout) dataset. Within the training and validation datasets, we randomly selected 84% of the transaction sequences for training purposes, while the remaining sequences were used for validation.

For the Amazon product review dataset, we followed a similar approach as described in [6, 22, 27]. For each user, we included the last 10 reviews in the test dataset. The reviews from the 11th to the second-to-last review were placed in the validation dataset, and all the reviews before the second-to-last review were assigned to the training dataset. In cases where the sequence length was less than 10, we added a special token to the left of the sequence repeatedly until the length reached 10.

We created transaction/review sequences as sliding windows of 10 records, with a stride of 5 in training data and validation data, and a stride of 1 in test data. We followed the same procedure outlined in [33] to quantize numerical features and generate vocabularies.

*Model Pre-training and Downstream Task Training.* For our experiments, we utilized TabBERT and FATA-Trans as our pre-trained models. When working with the synthetic transaction dataset, we intentionally excluded the label column, which indicates whether a transaction is abnormal, to avoid any leakage of target information during the pre-training phase [23]. However, for the Amazon product review dataset, we included the label column representing a reviewer's rating on a product. This inclusion was based on the assumption that previous ratings can be informative for predicting future ratings. During our experiments, we followed a similar procedure as described in [11, 33]. We randomly masked 15% of the static field tokens and 15% of the dynamic field tokens in each sequence. These masked tokens were replaced with the [MASK] token 80% of the time, with random tokens 10% of the time, and left unchanged 10% of the time. The model learned to restore these masked tokens using the cross-entropy loss. We pre-trained all the models for three epochs, using the same parameter settings in [33].

After pre-training, we applied a linear layer as the classification head, taking the concatenated sequence embeddings as input. For the synthetic transaction dataset, we down-sampled the normal transaction sequences in the training dataset for the classification task training procedure. This was done to make the ratio of normal to abnormal transactions 20:1 since abnormal transactions are extremely rare. Note that during the pre-training procedures, models used the whole training dataset. For the Amazon product review datasets, we masked the rating column of the last review in each sequence to prevent target leaking [23]. This ensured that the model did not have access to the prediction label during training. We trained both models for 20 epochs and employed early-stop criteria when the AUC (area under the receiver operating characteristic curve) score for the validation dataset did not improve over three consecutive evaluation steps.

It should be noted that both LightGBM and RNN models were directly trained for the classification tasks without pre-training. Again, for the synthetic transaction dataset, we down-sampled the normal transaction sequences in the training dataset to achieve a 20:1 ratio between normal and abnormal transactions. For the LightGBM model, we utilized the LightGBM Python library[3]. We primarily used the default parameter settings recommended by the package but made adjustments to the learning rate and number of boosting rounds based on the AUC score of the validation dataset.

The RNN model was trained using the Adam optimizer [25] with a fixed learning rate of 0.001. A batch size of 64 was used for the synthetic transaction dataset, while a batch size of 128 was used for the two Amazon product review datasets. The model was trained for 100 epochs, and early-stop decisions were determined by the AUC score of the validation dataset. During our experimentation,

[3]https://pypi.org/project/lightgbm/

we evaluated both one-layer and two-layer GRU networks and found that they exhibited nearly identical performance. Therefore, we present the results obtained from a one-layer GRU network with 256 hidden nodes.

**Table 1: Performance comparison on three datasets: Synthetic Transaction Dataset, Amazon Movies and TV, and Amazon Electronics.**

| Methods | AUC |
|---|---|
| **Synthetic Transaction** | |
| LightGBM | 0.9624 |
| RNN | 0.9866 |
| TabBERT | 0.9985 |
| FATA-Trans | **0.9992** |
| **Amazon Movies and TV** | |
| LightGBM | 0.7593 |
| RNN | 0.7634 |
| TabBERT | 0.7964 |
| FATA-Trans | **0.8057** |
| **Amazon Electronics** | |
| LightGBM | 0.6962 |
| RNN | 0.7040 |
| TabBERT | 0.7098 |
| FATA-Trans | **0.7206** |

## 3.4 Experimental Results

We evaluated AUC scores on the testing dataset for both the transaction anomaly detection and review rating prediction tasks. As depicted in Table 1, our method consistently outperforms other methods across all three datasets. This indicates that FATA-Trans effectively captures more precise user behavior patterns by leveraging the time interval and field-type information incorporated within the specially designed embedding and transformer layers.

**Table 2: Pre-training time comparison on three datasets: Synthetic Transaction, Amazon Movies and TV, and Amazon Electronics. We used batch size 64 for the Synthetic Transaction dataset and batch size 128 for the other two datasets.**

| Methods | Pretraining Time |
|---|---|
| **Synthetic Transaction** | |
| TabBERT | 3d11h |
| FATA-Trans | **2d12h** |
| **Amazon Movies and TV** | |
| TabBERT | 11h |
| FATA-Trans | **6h33m** |
| **Amazon Electronics** | |
| TabBERT | 2d13h |
| FATA-Trans | **1d3h** |

We conducted a comparison of the pre-training time between TabBERT and FATA-Trans. Both models were implemented using PyTorch [34] and pre-trained on a single NVIDIA Tesla A100 GPU. The batch size was set to 64 for the synthetic transaction dataset and 128 for the other two datasets. According to Table 2, our method demonstrates significantly shorter pre-training times compared to TabBERT. This is attributed to the fact that FATA-Trans avoids redundant repetition of static fields in the sequence and only inputs them into the static-field transformer. As a result, this approach reduces memory usage and substantially saves training time.

**Table 3: Performance of the proposed FATA-Trans and its variations on three datasets: Synthetic Transaction, Amazon Movies and TV, and Amazon Electronics.**

| Methods | AUC |
|---|---|
| **Synthetic Transaction** | |
| FATA-Trans (w/o time-aware position embedding) | 0.9982 |
| FATA-Trans (w/o field type aware design) | 0.9966 |
| FATA-Trans (w/o pretraining) | 0.9970 |
| FATA-Trans | **0.9992** |
| **Amazon Movies and TV** | |
| FATA-Trans (w/o time-aware position embedding) | 0.8036 |
| FATA-Trans (w/o field type aware design) | 0.8038 |
| FATA-Trans (w/o pretraining) | 0.7819 |
| FATA-Trans | **0.8057** |
| **Amazon Electronics** | |
| FATA-Trans (w/o time-aware position embedding) | 0.7108 |
| FATA-Trans (w/o field type aware design) | 0.7174 |
| FATA-Trans (w/o pretraining) | 0.7072 |
| FATA-Trans | **0.7206** |

## 3.5 Ablation Study

We designed two variations of our proposed method to assess the impact of our customized time-aware position embedding, static field transformer, and field-type embedding. Each variant differs from FATA-Trans in either position embedding or field transformer and field-type embedding. We also compared FATA-Trans with a variant that did not utilize pre-training, but instead directly trained on the downstream classification task.

*FATA-Trans (w/o time-aware position embedding).* This variation replaced the time-aware position embedding with the regular position embedding proposed in [11]. It still used the time interval as a dynamic field to capture the time information.

*FATA-Trans (w/o field type aware design).* This variation removed the static field transformer and field type embedding. It replicated the static fields in every record in a sequence as TabBERT did.

*FATA-Trans (w/o pre-training).* This variation skipped the pre-training step and instead directly trained on the downstream task.

Table 3 shows the experimental results on all three datasets. FATA-Trans achieves better results against all three variations, which demonstrates that the time-aware position embeddings,

static field transformer, and field-type embedding can better exploit the time interval information and better learn the latent patterns across fields and records. Moreover, the pre-training procedure can help the method obtain more useful representations for the downstream tasks.

### 3.6 Representation Visualization

We used the learned sequence embeddings from FATA-Trans, which were not fine-tuned for any particular downstream task, to explore the insights captured by our model. To analyze these embeddings, we applied Principal Component Analysis (PCA) and generated 3D plots using the first three principal components. To create the plot, we either concatenated the sequence embeddings within each window or used a single sequence embedding and then applied PCA.

In Figure 5a, we present the distribution of anomaly labels for the synthetic transaction dataset. Each point in the figure represents a windowed transaction sequence. Notably, we observe a clear separation between abnormal and normal sequences, suggesting that our learned representation has captured this information even without explicitly training an anomaly detection model. Figure 5b presents the distribution of transaction types within the 3D space. Each point represents a transaction record. Again, we can observe a near-perfect separation between the two common transaction types: online and offline (or point-of-sale) involving swipe and chip transactions. Figure 5c illustrates the distribution of several top MCCs (the most frequent MCCs in a card's transaction history). Each point represents the sequence embedding of the static fields within a windowed transaction sequence. Notably, we can observe significant separation among these MCCs. It's important to note that transaction type is a dynamic field, while the top MCC is a static field. Additionally, we performed the same visualization task using the sequence embeddings learned by the pre-trained TabBERT model. Figure 5d displays the distribution of transaction sequence embeddings learned by the TabBERT model. It is worth noting that TabBERT does not differentiate between static and dynamic fields. When comparing Figure 5c (FATA-Trans) and Figure 5d (TabBERT), we observe that the embeddings from TabBERT do not effectively separate the top MCCs, which is a static field. This indicates that FATA-Trans excels at capturing information from static fields compared to the TabBERT model.

In Figure 5e, we visualize the distribution of reviewer ratings for the Amazon Electronics dataset. Each point represents a windowed review sequence, and we can observe a clear separation between high and low rating sequences. This indicates that our method successfully captures the underlying patterns associated with different rating levels. In Figure 5f, we illustrate the separation between positive and negative reviewers based on their review history in the Amazon Movie and TV dataset. The feature "is-pos" is a derived feature that is created based on a user's review history. It serves as a static field and reflects a user's overall preference. Each point in the figure corresponds to the sequence embedding of the static fields within a windowed review sequence. Notably, we can observe a significant separation between positive and negative reviewers, indicating that our approach effectively extracts patterns from both static and dynamic fields within record sequences.

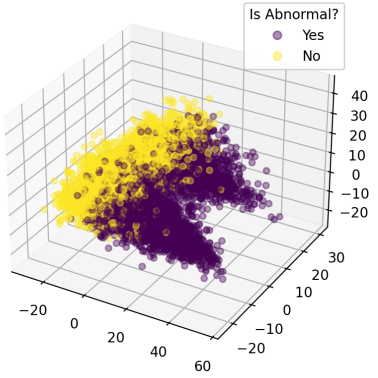
These figures collectively demonstrate the capability of our method to successfully capture and extract meaningful information from both static and dynamic fields within sequences.

## 4 RELATED WORK

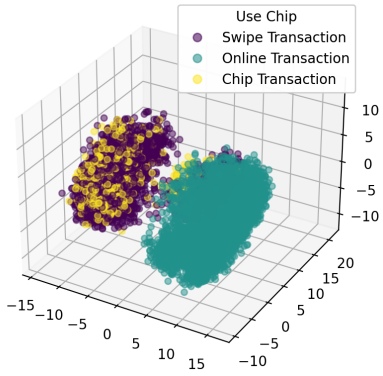
Sequential tabular data is ubiquitous across many industries. Different from attributed sequences [49], where each attribute is static, sequential tabular data has both static and dynamic feature fields. Recently, there has been growing interest in applying transformer-based models [11, 40] to tabular data. For example, SAINT[37] introduces an inter-sample attention strategy for modeling tabular datasets. TabNet[2] uses a sequential attention mechanism to choose a subset of semantically meaningful features to process at each decision step. TabTransformer[20] uses a transformer encoder to learn contextual embeddings on categorical features. AutoInt[38] automatically learns high-order feature interactions for CTR prediction using a self-attentive neural network. Shwartz-Ziv and Armon[36], Gorishniy et al. [14, 15], Rubachev et al. [35], and Levin et al. [26] conduct in-depth studies comparing the main families of deep learning architectures against gradient boosting trees. Borisov et al [4] and Badaro et al. [3] present extensive surveys in this field. However, most of these works are focused on non-sequential tabular data where rows in a table are independent and there are no temporal dependencies between the rows.

There is another line of research in sequential recommendation which leverages the sequential nature of a user's behavior to make better recommendations. Early work utilized RNN models and achieved state-of-art performance[18, 42, 47]. Recently, transformer-based models have become a proliferated approach. For example, SASRec[22] uses a self-attention mechanism combined with position embeddings to learn relevant items based on a user's past purchased items. TiSASRec[27] incorporates relative time intervals between any two items in a sequence into a self-attention mechanism to predict the next item that a user is likely to engage with. BERT4Rec[39] applies bidirectional attention to capture a users' sequential behavioral patterns. TLSRec[7] simultaneously models the global stability and local fluctuation of a user's preference with a hierarchical attention network. Rec-Denosier[6] adaptively eliminates the noisy items during the training process to remove irrelevant information in a user's behavior sequence. Transformers4Rec[10] performs an empirical analysis with broad experiments of various transformer architectures for the task of sequential recommendation. Despite encouraging performance, a common limitation of these approaches is their exclusive focus on item IDs within a univariate sequence, disregarding other valuable information associated with items and users, such as item category, item popularity, user past comments and ratings, and more.

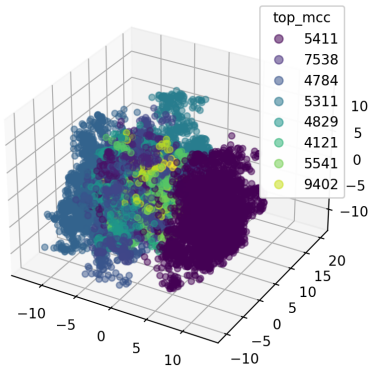
Several studies have addressed the limitation of exclusively relying on item IDs by incorporating other valuable information associated with items and users. For instance, FDSA [45] introduces a feature-level self-attention block to integrate detailed attribute information about items. BST [8] combines both item IDs and category IDs to construct user behavior sequences, which are then fed into a transformer layer. SSE-PT[41] incorporates user embeddings into a self-attentive neural network to personalize the transformer model. $S^3$-Rec[48] utilizes mutual information maximization



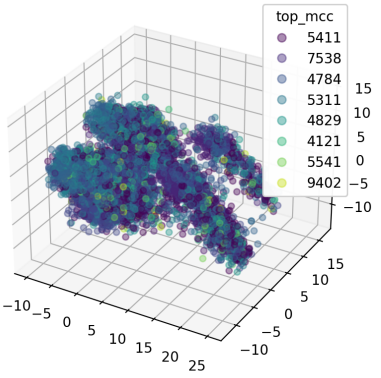

**(a) Scatter plot for abnormal transactions in the Synthetic Transaction Dataset. Each dot represents the first three principal components of the concatenated sequence embeddings of a windowed transaction sequence. The color of the dot indicates whether the last transaction in the sequence is abnormal or not.**



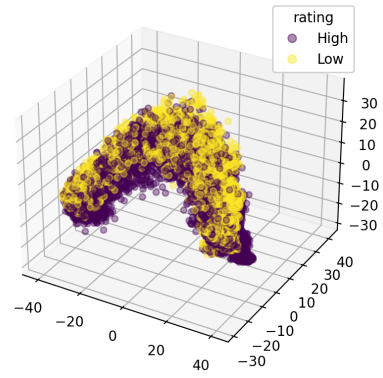

**(b) Scatter plot for the transaction types in the Synthetic Transaction Dataset. Each dot represents the first three principal components of a transaction's sequence embedding. The color of the dot represents transaction types: swipe, online, or chip.**



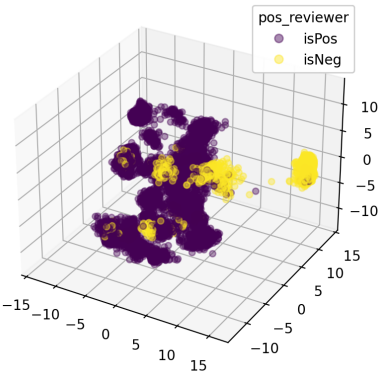

**(c) Scatter plot for top MCCs in the Synthetic Transaction Dataset. Each dot represents the first three principal components of a static fields' sequence embedding. Embeddings are generated by pre-trained FATA-Trans. The color of the dot represents the most frequent MCC in a card's transaction history.**




**(d) Scatter plot for top MCCs in the Synthetic Transaction Dataset. Each dot represents the first three principal components of a transaction's sequence embedding. Embeddings are generated by pre-trained TabBERT. The color of the dot represents the most frequent MCC in a card's transaction history.**




**(e) Scatter plot for reviewer ratings in Amazon Electronics. Each dot represents the first three principal components of the concatenated sequence embeddings of a windowed review sequence. Embeddings are generated by pre-trained FATA-Trans. The color of the dot represents if the last rating in the sequence is high (>=4) or low (<=3).**




**(f) Scatter plot for high or low rating reviewer in Amazon Movies and TV. Each dot represents the first three principal components of a static fields' sequence embedding. Embeddings are generated by pre-trained FATA-Trans. The color of the dot represents if a reviewer gave more high rating (>=4) or low rating (<=3) in the reviewer's review history.**

within a self-attentive architecture to capture correlations among attributes, items, subsequences, and sequences. TabBERT[33], from which we got inspiration, provides a comprehensive framework for modeling multivariate sequential tabular data. However, these approaches overlook the distinction between static and dynamic fields and do not account for time interval information in position embedding.

## 5 CONCLUSION

In this paper, we present FATA-Trans, a novel Field- and Time-Aware Transformer for modeling record sequences in sequential tabular data. Compared to previous works, FATA-Trans has a special design to process static and dynamic fields separately, and the time interval information is also incorporated into time-aware position embedding. We show that the representations learned by FATA-Trans provide consistent performance gain in both transaction anomaly detection and product review rating prediction tasks, achieved with substantially less training time. Visualization figures also show that FATA-Trans can capture important information from both static and dynamic fields.

# REFERENCES


[1] Erik Altman. 2021. Synthesizing credit card transactions. In *Proceedings of the Second ACM International Conference on AI in Finance*. 1–9.

[2] 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.

[3] Gilbert Badaro and Paolo Papotti. 2022. Transformers for tabular data representation: a tutorial on models and applications. *Proceedings of the VLDB Endowment* 15, 12 (2022), 3746–3749.

[4] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[5] Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana SC Almeida, João Tiago Ascensão, and Pedro Bizarro. 2020. Interleaved sequence rnns for fraud detection. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3101–3109.

[6] Huiyuan Chen, Yusan Lin, Menghai Pan, Lan Wang, Chin-Chia Michael Yeh, Xiaoting Li, Yan Zheng, Fei Wang, and Hao Yang. 2022. Denoising self-attentive sequential recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 92–101.

[7] Lihua Chen, Ning Yang, and Philip S Yu. 2022. Time Lag Aware Sequential Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 212–221.

[8] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[10] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 143–153.

[11] 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).

[12] James Fiedler. 2021. Simple modifications to improve tabular neural networks. *arXiv preprint arXiv:2108.03214* (2021).

[13] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[14] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2022. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems* 35 (2022), 24991–25004.

[15] 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.

[16] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815* (2022).

[17] Hongwei Han, Jialiang Xu, Mengyu Zhou, Yijia Shao, Shi Han, and Dongmei Zhang. 2022. LUNA: Language Understanding with Number Augmentations on Transformers via Number Plugins and Pre-training. *arXiv preprint arXiv:2212.02691* (2022).

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

[21] Dietmar Jannach, Gabriel de Souza P. Moreira, and Even Oldridge. 2020. Why are deep learning models not consistently winning recommender systems competitions yet? A position paper. In *Proceedings of the Recommender Systems Challenge 2020*. 44–49.

[22] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[23] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. 2012. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 4 (2012), 1–21.

[24] 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).

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

[26] Roman Levin, Valeriia Cherepanova, Avi Schwarzschild, Arpit Bansal, C Bayan Bruss, Tom Goldstein, Andrew Gordon Wilson, and Micah Goldblum. 2022. Transfer Learning with Deep Tabular Models. *arXiv preprint arXiv:2206.15306* (2022).

[27] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

[28] Xurui Li, Wei Yu, Tianyu Luwang, Jianbin Zheng, Xuetao Qiu, Jintao Zhao, Lei Xia, and Yujiao Li. 2018. Transaction fraud detection using gru-centered sandwich-structured model. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*. IEEE, 467–472.

[29] Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015).

[30] Simone Luetto, Fabrizio Garuti, Enver Sangineto, Lorenzo Forni, and Rita Cucchiara. 2023. One Transformer for All Time Series: Representing and Training with Time-Dependent Heterogeneous Tabular Data. *arXiv preprint arXiv:2302.06375* (2023).

[31] Aashiq Muhamed, Iman Keivanloo, Sujan Perera, James Mracek, Yi Xu, Qingjun Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul Chilimbi. 2021. CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*.

[32] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

[33] Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. 2021. Tabular transformers for modeling multivariate time series. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3565–3569.

[34] 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).

[35] Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. 2022. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208* (2022).

[36] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion* 81 (2022), 84–90.

[37] 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).

[38] 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.

[39] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[40] 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).

[41] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 328–337.

[42] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 729–732.

[43] Dongyu Zhang, Jidapa Thadajarassiri, Cansu Sen, and Elke Rundensteiner. 2020. Time-aware transformer-based network for clinical notes series prediction. In *Machine learning for healthcare conference*. PMLR, 566–588.

[44] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)* 52, 1 (2019), 1–38.

[45] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation.. In *IJCAI*. 4320–4326.

[46] Wei Zhang, Liang Wang, Robert Christensen, Yan Zheng, Liang Gou, and Hao Yang. 2021. Transaction sequence processing with embedded real-time decision feedback. US Patent 11,153,314.

[47] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[48] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management.* 1893–1902.

[49] Zhongfang Zhuang, Xiangnan Kong, Rundensteiner Elke, Jihane Zouaoui, and Aditya Arora. 2019. Attributed sequence embedding. In *2019 IEEE International Conference on Big Data (Big Data).* IEEE, 1723–1728.